COMPUTER INPUT DEVICE FOR PEOPLE WITH SEVERE DISABILITIES

By

MOSSAB AWAD ALLAH ATTA RIZIG

INDEX NO. 064080

Supervisor

Dr. Sharief F. Babikir

A REPORT SUBMITTED TO

University of Khartoum

in partial fulfillment of the requirements for the degree of

B.Sc (HONS) Electrical and Electronic Engineering

(COMMUNICATION ENGINEERING)

Faculty of Engineering

Department of Electrical and Electronic Engineering

July 2011
DECLARATION OF ORIGINALITY

I declare that this report entitled “Computer input device for people with severe disabilities ” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Name: Mossab Awad Allah Atta Rizig

Signature:

Date: 03 Jul. 2011
Dedication

To my family who has been a source of encouragement and inspiration to me throughout my life, and without whom I would never gone this far, I thank them DEEPLY for the great support and love they gave me. I thank Allah for their presence in my life every day, may Allah give me the strength to repay them.
Acknowledgement

From the formative stages of this thesis, to the final draft, I owe an immense debt of gratitude to my supervisor, Dr. Sharief F. Babikir. His sound advice and careful guidance were invaluable.

My PARTNER Nader Tawfeeg without whom this project might not have been completed, and to whom I am greatly indebted.

A special thanks go to Abd El Karim Al Badri for his massive support.

Finally, I would be remiss without mentioning my friends and family, whose extreme generosity will be remembered always.

To each of the above, I extend my deepest appreciation.
Abstract

A computer input device for people with severe disabilities is designed. It allows them to use the functions of an ordinary mouse. It uses the head movement as a mechanism to move the cursor and intentional eye blinking as a selection mechanism. Head movement is detected using an accelerometer, while the blinking is detected using a reflective object sensor. Sensed data is processed using a microcontroller unit interfaced with the computer using the personal system/2 (PS/2) protocol. The sensors used have been bought from abroad, that delayed the work on the device. The final device did not work since it could not be recognized by the computer, but the functions that represent the PS/2 protocol have been written.
المستخلص

تم تصميم جهاز إدخال للحاسوب للأشخاص الذين يعانون من إعاقات بدنية جسيمة. فهو يسمح لهم

استخدام وظائف الفأرة (الماوس) العادية. يستخدم هذا الجهاز حركة الرأس كآلية لتحرك المؤشر ووضارات

العين المتعددة كآلية الاختيار. يتم الكشف عن حركة الرأس باستخدام حساس التسارع، في حين يتم الكشف

عن الوضادات بحساس الأجسام العاكسة. يتم معالجة الإشارات المحسوسة باستخدام وحدة متحكم دقيق

متصلة مع الحاسوب عن طريق بروتوكول الكمبيوتر الشخصي $2$ (PS/2). قد تم شراء أجهزة الاستشعار

المستخدم من الخارج، والتي أُورِدت العمل على الجهاز. لم يعمل الجهاز لعدم إمكانية التعرف عليه من قبل

الكمبيوتر، ولكن تم كتابة الوظائف التي تمثل البروتوكول $2$ / PS.
Table of contents

DECLARATION OF ORIGINALITY .................................................................................. II
DEDICATION .................................................................................................................. III
ACKNOWLEDGEMENT .................................................................................................... IV
ABSTRACT ..................................................................................................................... V
المستخلص ....................................................................................................................... VI
TABLE OF CONTENTS .................................................................................................... VII
LIST OF FIGURES .......................................................................................................... X
LIST OF TABLES .............................................................................................................. XII
1 CHAPTER ONE ........................................................................................................... 1

INTRODUCTION ............................................................................................................. 1
1.1 MOTIVATION: ........................................................................................................... 1
1.2 AIMS AND OBJECTIVES: ...................................................................................... 2
1.3 APPROACH: ............................................................................................................ 2
1.4 THESIS LAYOUT: .................................................................................................... 2

2 CHAPTER TWO .......................................................................................................... 4

HUMAN COMPUTER INTERACTION (HCI) TECHNIQUES FOR PEOPLE WITH SEVERE DISABILITIES ........................................................................................................ 4
2.1 OVERVIEW: ............................................................................................................ 4
2.2 MOUSE: ................................................................................................................... 4
   2.2.1 Mouse technologies: ....................................................................................... 5
   2.2.2 Connecting a computer mouse: ...................................................................... 6
2.3 PARALYSIS: ............................................................................................................ 6
   2.3.1 Types: ............................................................................................................. 6
   2.3.2 Common Causes: .......................................................................................... 6
   2.3.3 Paralysis Advice: ......................................................................................... 7
2.4 TECHNIQUES USED TO HELP PARALYZED PEOPLE USE COMPUTERS: .......... 7
   2.4.1 Hardware computer assistive devices: ......................................................... 8
   2.4.2 Computer vision-based interaction systems: .............................................. 12
CHAPTER THREE ........................................................................................................17

METHODOLOGY ......................................................................................................... 17

3.1 OVERVIEW .............................................................................................................. 17
3.2 PS/2 PROTOCOL .................................................................................................... 18
   3.2.1 Introduction ..................................................................................................... 18
   3.2.2 Mouse Inputs .................................................................................................. 18

THE STANDARD PS/2 MOUSE INTERFACE SUPPORTS FIVE INPUTS; X (RIGHT/LEFT) MOVEMENT, Y (UP/DOWN) MOVEMENT, LEFT BUTTON, MIDDLE BUTTON AND RIGHT BUTTON. THE MOUSE PERIODICALLY READS THESE INPUTS AND UPDATES VARIOUS COUNTERS AND FLAGS TO REFLECT MOVEMENT AND BUTTON STATES .................................................................................18
   3.2.3 Movement Data Packet .................................................................................... 18
   3.2.4 Resolution and Sample rate ............................................................................ 19
   3.2.5 PS/2 protocol .................................................................................................. 19

3.3 ACCELEROMETER ................................................................................................. 22
3.4 REFLECTIVE OBJECT SENSOR ........................................................................... 24
3.5 MICROCONTROLLER UNIT ................................................................................... 25

CHAPTER FOUR ............................................................................................................ 27

IMPLEMENTATION ......................................................................................................... 27

4.1 OVERVIEW .............................................................................................................. 27
4.2 THE HARDWARE COMPONENT ............................................................................. 28
   4.2.1 Procedures ...................................................................................................... 29

4.3 THE SOFTWARE COMPONENT ............................................................................. 29
   4.3.1 PS/2 emulation ................................................................................................ 30
   4.3.2 Hand shaking process (initialization process) .................................................. 32
   4.3.3 Running in stream mode .................................................................................. 32

CHAPTER FIVE .............................................................................................................. 37

RESULTS AND DISCUSSION ....................................................................................... 37

5.1 RESULTS .................................................................................................................. 37
5.2 DISCUSSION ............................................................................................................ 39

CHAPTER SIX ............................................................................................................... 40

CONCLUSION AND FUTURE WORK ........................................................................ 40

6.1 CONCLUSION ......................................................................................................... 40
6.2 FUTURE WORK

REFERENCES

APPENDIX A

APPENDIX B

APPENDIX C
List of figures

FIGURE 1.1: SOME POINTING DEVICES ............................................................................................................. 1
FIGURE 2.1: TYPES OF PARALYSIS, GREY PART ARE PARALYZED ................................................................. 7
FIGURE 2.2: TWO TYPES OF SWITCHES: ACTIVATED BY HEAD (LEFT) AND BY CHIN (RIGHT), ............... 8
FIGURE 2.3: EXAMPLES OF JOYSTICKS: ACTIVATED BY CHIN (LEFT) AND BY MOUTH (RIGHT)......... 9
FIGURE 2.4: EXAMPLES OF HEAD STICK POINTERS ......................................................................................... 9
FIGURE 2.5: EAR PRESSURE SENSOR PARTIALLY INSERTED INTO THE EAR CANAL TO DETECT
           SIGNALS FROM TONGUE MOVEMENTS ..................................................................................................... 10
FIGURE 2.6: AN INDIVIDUAL WITH MAGNETIC TONGUE PIERCING WEARING THE TONGUE DRIVE
           SYSTEM HEADSET ..................................................................................................................................... 11
FIGURE 2.7: MICROWIRES EMERGING FROM THE GREEN AND ORANGE TUBES CONNECT TO TWO
           ARRAYS OF 16 MICROELECTRODES ....................................................................................................... 12
FIGURE 2.8: THE FACIAL MOUSE’S WORKING ENVIRONMENT ......................................................................... 14
FIGURE 2.9: FLUORESCENT COLOUR MARKER SAMPLES ................................................................................. 15
FIGURE 2.10: SEQUENCE OF THE INTERNAL WEBCOLOR OPERATION ............................................................. 15
FIGURE 3.1: DEVICE TO HOST COMMUNICATION ............................................................................................ 21
FIGURE 3.2: HOST TO DEVICE COMMUNICATION ............................................................................................ 21
FIGURE 3.3: DETAILED HOST TO DEVICE COMMUNICATION .......................................................................... 22
FIGURE 3.4: SIMPLIFIED TRANSDUCER PHYSICAL MODEL .............................................................................. 24
FIGURE 3.5: A PHOTO TRANSISTOR ................................................................................................................ 24
FIGURE 3.6: PHOTOELECTRIC EFFECT .............................................................................................................. 25
FIGURE 4.1: GENERAL DESIGN ....................................................................................................................... 27
FIGURE 4.2: PS/2 SOCKET ................................................................................................................................. 28
FIGURE 4.3: FUNCTIONS TO SEND DATA TO HOST ........................................................................................ 31
FIGURE 4.4: HAND SHAKING PROCESS .......................................................................................................... 34
FIGURE 4.5: DETERMINATION OF X MOVEMENT ......................................................................................... 35
FIGURE 4.6: BLINK DETECTION ....................................................................................................................... 36
FIGURE 5.1: SIMULATION ENVIRONMENT ...................................................................................................... 37
FIGURE 5.2: ABOVE ARE THE FRAMES 0xAA AND 0x00, BELOW IS THE CLOCK ........................................... 38
FIGURE 5.3: SOME DATA FRAMES SENT TO THE HOST .................................................................................. 39
FIGURE 6.1: SUGGESTION FOR WIRELESS INPUT DEVICE ............................................................................ 41
List of Tables

TABLE 3.1: PS/2 MOUSE MOVEMENT DATA PACKET .......................................................... 19
TABLE 3.2: PS/2 BUS STATES ......................................................................................... 20
TABLE 3.3: PS/2 DEVICE TO HOST FRAME FORMAT ...................................................... 20
TABLE 4.1: PINS ASSIGNMENT OF PS/2 SOCKET ....................................................... 28
TABLE 4.2: ATMega32 PORT C CONFIGURATION ....................................................... 29
Chapter One

Introduction

The phenomenal growth of computers in modern society has been dramatic and spectacular. Almost everywhere computers are playing their roles efficiently. They are used for work, education, communication, entertainment, shopping and lots of other useful things. The use of computers has simplified many previously complex tasks to a mere touch of a few buttons on the keyboard and a few clicks on the mouse. They have become an indispensable tool in facilitating communication and work. The presence of internet has revolutionized the way people communicate and interact. Geographical distance has been overcome by the linkage opened up by computer networking or internet social networking. Business transactions, dissemination of information and interpersonal interactions have all been incorporated in the computerization process.

Computers can be designated as one of the most creative innovations of human beings. In coming days computers are even going to be more pervasive, because technology is getting advanced day by day. In fact the world wouldn’t be complete without computers and their applications.

1.1 Motivation

The main way to interact with a computer is to use pointing devices such as mice, trackballs, touch pads, etc. All these devices are used to control the movement of the cursor on the screen to navigate through the user interface and make selections and actions.

![Mouse](Mouse.png)  ![Trackball](Trackball.png)  ![Touchpad](Touchpad.png)

Figure 1.1: Some pointing devices
These pointing devices require the one who is using the computer to use his hand to control them, but how to interact with the computer if the hand isn’t functioning properly?

There are many people suffering from severe disabilities, like the amputees and those having paralysis and can’t use their hands. Those people find difficulties using computers, thus find themselves dependent on others to perform tasks they can do by themselves if they can use computers. They are also swamped by the technological revolution that is occurring and become isolated from the career life, so they passively affect the community being non-productive and others dependent.

1.2 Aims and objectives

The project aims to create a new input device that allows people with special needs interact with computer easily or, at least, use the computer as an educational or training tool. This will enrich their lives and help them build some sort of self-confidence and decrease their need for others. They may also be productive as well as creative since the computer is such a powerful tool.

1.3 Approach

The input device is based on the use of sensors to detect the head movement and eye blinking as a way to interact with computer. Head movement is detected using accelerometer while the eye blinking is detected using a reflective object sensor. The sensed signals are fed into a microcontroller unit which communicates with computer using the personal system/2 (PS/2) protocol and acts with it as a normal mouse.

1.4 Thesis layout

Chapter Two: Human Computer Interaction (HCI) techniques for people with severe disabilities

Gives brief information about paralysis and some of the methods and techniques used to help people with severe disabilities use computer.

Chapter Three: Methodology

Includes methods used to achieve the design, and the needed information about these methods to implement the project.
Chapter Four: Implementation

Contains hardware and software details of the design.

Chapter Five: Results and discussion

Discusses the results obtained.

Chapter Six: Conclusion

Conclusion and suggestion for future work.
Chapter Two

Human Computer Interaction (HCI) techniques for people with severe disabilities

2.1 Overview

Human computer interaction (HCI) is the study, planning and design of the interaction between people and computers. Interaction between users and computers occurs at the user interface, which includes both software and hardware; for example, characters or objects displayed by software on a personal computer's monitor, input received from users via hardware peripherals such as keyboards and mice, [1].

This chapter gives a description of the mouse as an important hardware tool for HCI. It also includes brief information about paralysis, its types, causes as well as an advice to avoid it, since the project aims to help paralyzed people.

Massive effort has been put on the field of developing devices and techniques that are friendly to people suffering from paralysis to ease their lives and make them close to normal as possible. Some of these techniques will be discussed on this chapter.

2.2 Mouse

It is a device that controls the movement of the cursor on a display screen through rolling along a hard, flat surface. As the mouse is moved, the pointer on the display screen moves in the same direction. It contains at least one button and sometimes as many as three, which have different functions depending on what program is running. Some mice also include a scroll wheel for scrolling through long documents. The mouse is important for graphical user interfaces because any user can simply point to options and objects and click a mouse button. Such applications are often called point-and-click programs, [2].
2.2.1 Mouse technologies

The mouse can use one of three different technologies; mechanical, optical or laser technologies.

2.2.1.1 Mechanical mouse

The first type of mouse around, the mechanical mouse (also known as a ball mouse), uses a moving ball to work. As the mouse is moved across a surface, the ball moves too. Also inside the mouse are two rollers that roll against two sides of the moving ball. One roller tracks the horizontal motion of the mouse, and the other roller tracks the vertical motion.

The motion of the two rollers is converted into electrical signals which are then sent to the computer through a cord. The software on the computer then converts these electrical signals into meaningful X and Y movement of the mouse cursor that is seen on-screen [2].

2.2.1.2 Optical mouse

The optical mouse uses a tiny camera to take 1,500 pictures every second. Able to work on almost any surface, the mouse has a small, red Light-Emitting Diode (LED) that bounces light off that surface onto a complimentary metal-oxide semiconductor (CMOS) sensor. The CMOS sensor sends each image to a digital signal processor (DSP) for analysis. The DSP, operating at 18 MIPS (million instructions per second), is able to detect patterns in the images and see how those patterns have moved since the previous image. Based on the change in patterns over a sequence of images, the DSP determines how far the mouse has moved and sends the corresponding coordinates to the computer. The computer moves the cursor on the screen based on the coordinates received from the mouse. This happens hundreds of times each second, making the cursor appear to move very smoothly [2].

Optical mice have many advantages over mechanical mice, some of them include:

- Less wear and lower chance of failure because there are no moving parts.
- Dirt or dust can't get into the mouse and interfere with the sensors.

2.2.1.3 Laser mouse

Laser technology has been the newest technology breakthrough in computer mice. The way in which a laser computer mouse works is similar to an optical mouse, but with better results. The laser workings inside of a laser mouse can deliver up to 20 times the performance of an optical mouse. This means more accurate tracking of motion, and smoother movement. Laser mice can also work flawlessly on almost any type of surface, whereas optical mice haven't got as much versatility. [2].
2.2.2 Connecting a computer mouse

1) Wired connections:
   a) PS/2 connection, where a PS/2 mouse is connected to PS/2 port.
   b) USB connection, where the mouse is connected to USB port in the computer.

2) Wireless connection:
   Wireless mice come with a USB, Bluetooth or Infrared receivers, which are simply plugged into a USB port on the computer and they are ready to use.

2.3 Paralysis

Paralysis is broadly characterized by the loss of muscle control, power and movement to one or more parts of the body. Nerve disease or injury is the most common reason for paralysis and is directly caused by a wide range of problems, Error! Reference source not found.

2.3.1 Types

There are basically two types of paralysis under which most paralyzing disorders occur: localized and generalized. Localized paralysis means that the affliction is limited to one region or side of the body. Generalized paralysis usually means that the problem exists over large parts of the body, or to the entire body, Error! Reference source not found.

2.3.2 Common Causes

There are two common causes of paralysis. The first is stroke and the second is trauma to the nervous system, brain, neck, back or spinal cord (automobile accidents, gunshots and other forms of violent action against the spinal structures can injure or sever the actual spinal cord). If serious damage occurs in the lower spinal cord, it may lead to paralysis of both legs (paraplegia paralysis), but if the damage occurs in the higher spinal cord, closer to the neck, paralysis of all four limbs may occur (quadriplegia paralysis), Error! Reference source not found.

Besides stroke and trauma, there are medical conditions and diseases that cause some degree of paralysis. These conditions typically result in varying degrees of either localized or generalized paralysis, Error! Reference source not found.
2.3.3 Paralysis Advice

The spinal cord is such a delicate structure and once it is damaged, it typically does not regenerate. It is for this reason that many spinal cord injuries are permanent in nature. There are some new hopes in the field of spinal cord injury treatment. Many new drugs and stem cell research offer a promise for one day reversing disability due to cord injury or other neurological damage. In the meantime, advances in mobility, support and patient comfort have improved the lives of affected individuals exponentially. Best of all, social and governmental recognition of a disabled person's rights have created an environment which is welcoming and accessible to everyone with a functional impairment.

2.4 Techniques used to help paralyzed people use computers

A lot of researches have been carried out to develop techniques to help paralyzed people use computers. Most of the recent computer assistive technology devices are based on the extraction of a stimulus generated by the users, usually through voluntary movement of any part of their body.

These techniques can be broadly divided into two groups:

- Hardware computer assistive devices.
- Computer vision-based interaction systems
2.4.1 Hardware computer assistive devices

These techniques are based on attaching a hardware component to a well-functioning moving part on the body, in order to sense a control signal to interact with the computer. An assistant software is used to translate these signals, so a computer can understand them.

2.4.1.1 Switch

The switch is one of the most simple and widely used computer access systems and consists of an electrical device that the user activates with the part of his/her body that has “movement”, for example the switch can be activated by head or chin. The switch is available in different sizes and sensibilities, [6].

In general, switches allow for a simple interaction and are usually combined with (screen) scan software. The scan is done in one dimension (as a list of menu options, icons in a window, rows and columns of the screen, etc.). The user activates the switch when the desired option is highlighted to execute the selection, [6].

Figure 2.2: Two types of switches: activated by head (left) and by chin (right), [6]

However, this technology has a technologic problem: its low bandwidth. This feature affects a normal communication between the user and the system; the pair action-reaction is not instantaneous (with action meaning “what the user does” and reaction “the system feedback”) and can frustrate the user with a lot of dead time, [6].

2.4.1.2 Joystick

The joystick is a well-known computer peripheral used mainly for gaming. In the ambit of disabilities it is used for controlling a mouse pointer or driving wheelchair, for example. It usually acts as a substitute for a mouse when the user has difficulties using one. There are joysticks that can be used with the chin, or mouth, [6].
Interaction is based on a succession of directional selections over time; this allows for greater bandwidth than the switch. But the time constraint remains present when emulating the mouse functionality. Thus the user must wait while the pointer is moving. [6]

2.4.1.3 Head Pointer

The head pointer is a pointing device controlled with the user’s head and is useful for people with good cephalic control. There are different kinds of head pointers:

- Head sticks are head-worn pointers: These can be used in a number of ways: for signaling pictures, words, communication board icons; as a keyboard aid; or as a pencil holder, for turning pages or drawing, for example. The only requirement is the user ability to move their head with certain precision. The user of these devices heavily depends on third party assistance in order to place the device on user’s head, [6].

- Light (laser) head pointers: These have evolved from the early head pointers. A laser emitter is mounted on a head-worn device and can be used for signaling functions. The
main difference from the preceding technology is that while the older pointers interact with physical touch, the new ones interact with a laser light, [4].

- Electronic head pointers: Based on infrared or ultrasonic technology, these devices usually require a mounted headset or reflective dot on the user’s head to make the system functional. The pointer is used mainly to control an onscreen pointer, [6].

All head pointing devices usually emulate a standard mouse, meaning they have a two dimensional interaction space equivalent to a standard mouse. The click generation is usually done through the mouse-over method (also called a dwell click) that consists of stopping the pointer over the desired location for a moment.

2.4.1.4 Pressure waves in the ear induced by tongue movements

This is carried out by a team led by Ravi Vaidyanathan from the Naval Postgraduate School, Southern Illinois University, Case Western Reserve University, the University of Southampton, and Think-A-Move, Ltd. They proved that by inserting a small microphone into the ear, the waves from the air pressure changes provoked by the tongue can be tracked down and distinguished for computer commands, Error! Reference source not found.

![Figure 2.5: Ear pressure sensor partially inserted into the ear canal to detect signals from tongue movements, Error! Reference source not found.](image)

- It’s obvious that this method requires quite a lot of research (a big team from many universities) and a design for the ear pressure sensor that should be used.
- The details of this work are not yet available for commercial use.

2.4.1.5 Tongue Drive System (TDS)

Maysam Ghovanloo, of the GT-Bionics Laboratory at Georgia Institute of Technology, and his research team have developed a tongue-operated Assistive technology called the Tongue Drive System (TDS) that is unobtrusive, wearable, and wireless and could substitute for many arm and hand functions, 0.
Figure 2.6: An individual with magnetic tongue piercing wearing the Tongue Drive System headset, [8]

Tongue movements change the magnetic field around the mouth, which is detected by small magnetic sensors mounted on the headset near the user’s cheeks. A wireless transmitter in the headset sends information from the sensors to the computer that translates the changes in magnetic field into the desired user commands, 0.

- This system has two main disadvantages:
  1- Requires much training for different commands.
  2- Few numbers of commands are available.

2.4.1.6 Easy input

Easy Input is a head-controlled keyboard and mouse input device for disabled users. This system was designed by students in Cornell University in the United States. The system uses accelerometers to detect the user’s head tilt in order to direct mouse movement on the monitor. The clicking of the mouse is activated by the user’s eye blinking through a reflective sensor. The keyboard function is implemented by allowing the user to scroll through letters with head tilt and with eye blinking as the selection mechanism, [7].

2.4.1.7 Neural Interfaces

Neural interfaces allow computers to pick up the user’s intention through sensors connected to nerve terminations on different parts of the body or directly to the brain. Because interaction is done solely through the voluntary control of user’s own mental activity this technology implies that no mobility is necessary for use, [6].

Although it’s not necessary to have a well-functioning moving part on the body, this is considered as a hardware technique, because of the use of sensors.
2.4.1.8 Microelectrodes on brain

A University of Utah study shows that brain signals controlling arm movements can be detected accurately using new microelectrodes that sit on the brain but don't penetrate it, [9]. For people who have lost a limb or are paralyzed, this device should allow a high level of control over a prosthetic limb or computer interface that decodes signals from the brain. A lot of work is needed to refine computer software that interprets brain signals so they can be converted into actions, like moving an arm, [9].

Figure 2.7: Microwires emerging from the green and orange tubes connect to two arrays of 16 microelectrodes, [9]

2.4.2 Computer vision-based interaction systems

Computer vision-based interaction systems process the images coming from one or more cameras to extract features that are interpreted for implementation by way of specialized software. These systems are extremely flexible because any modification detected in the video is
susceptible to be interpreted by the computer and used to unleash some action. There are many advantages to computer vision-based interaction systems. First, a user can interact at a distance, without physical contact with the device. Additionally, virtually any part of the body with mobility can be used to perform the interaction, which is especially important for people with severe physical disabilities. It is also possible to combine classic interaction peripherals, like keyboard or mouse, to improve interaction in a multimodal way. In addition, Webcam-based technologies, for instance, are particularly affordable compared to professional (industrial) computer vision devices. The idea of using computer vision-based interaction systems for people with severe movement restrictions is not new. [10].

There are two types of computer vision-based systems: one based on face tracking and the other on colour tracking. [11].

2.4.2.1 Facial Mouse

The Facial Mouse is a mouse emulator system based on the facial movement of the user. A Webcam is placed in front of the user, focusing on the user’s face. A motion extraction algorithm, which is user independent, is used to extract the facial motion from the video. This motion is used to move the mouse pointer that is controlled in a fashion relatively similar to standard mouse devices. This system can be used with great accuracy even when the user has exiguous cephalic motion control. [11].

The click can be generated through several mechanisms:

- **Built-in mechanisms:**
  - Dwell click: This click is automatically generated after stopping the pointer for a certain amount of time.
  - Sound click: The click is generated when the user emits a sound whose input level is greater than a configured threshold.

- **External mechanisms:** An external device can be used to send click commands to the computer, like a keyboard, a standard mouse or a mechanical switch for example.

The Facial Mouse’s working environment is composed of a camera, a computer, and the software.
The Webcam captures the user’s movements that a software program translates in order to position the cursor on the screen. [6]

A good example of the facial mouse technique is a computer program called “Camera Mouse” that enables people with disabilities to communicate just by moving their head. It essentially replaces the mouse so it can be used with all types of application programs and websites.

Support for this project – Camera Mouse - came from the National Science Foundation, the Philanthropy Committee of Mitsubishi Electronic Research Labs (MERL), Mitsubishi Electric America Foundation and the Accenture Fund at Boston College. Boston College and Boston University also provided generous support. [12].

This is a very good method, but it requires writing of a computer program that may be a little bit complicated. Also it requires a camera, that isn’t cheap.

Similar to facial mouse is the eye tracking system that consists of one or more cameras that focus on one or both eyes and record the movement as the user looks at an area. Eye tracking setups vary greatly: some are head-mounted, some require the head to be stable, and some automatically track the head. Eye tracking systems allow the use of a computer by people with eye-only motion. [13]
2.4.2.2 The WebColor Detector

This is software which uses a camera to gather data. It is able to detect in real-time the presence or absence of a distinctive color and to track its position. Color markers are usually fluorescent-colored pieces of paper that can be attached to any surface or to the user’s body, [6].

![Fluorescent colour marker samples, [6]](image)

Figure 2.9: Fluorescent colour marker samples, [6]

Figure (2.10) shows a sequence of the internal WebColor operation. The first step must train the software for the desired colour, which is then saved and reused in subsequent executions. The training is performed by making a single click over the desired color, [4].

![Sequence of the internal WebColor operation](image)

(a) After the color training process, WebColor detects a blue color, and then a red box encircles the marker. (b) The program “sees” only the white stain that corresponds to the marker, a red box
encircles the marker; (c) When the mark is removed, (d) the result is a black image because nothing is detected, [6]

Afterward, the application automatically finds and tracks that specific colour marker as a means of operating the system. The position of the marker in the video is coordinated with the screen pointer. So moving the marker in the web camera scene moves the pointer on the screen. The click is emulated with the dwell click technique, [6].
Chapter Three

Methodology

3.1 Overview

It is necessary to design a hardware computer assistive device that will provide the basic functions of a computer mouse and can be used with convenience by people with severe disabilities; i.e. people with no functioning limbs; e.g. people suffering from quadriplegia paralysis and amputees with no limbs at all.

Such people may use the basic functions of the mouse by another device that can be controlled with another part of the body. The device should detect the movement and provide it to a software module that acts as an interface to the computer, in order to translate the detected movement as a normal mouse commands.

This project uses the head movement as the key to design such a device. Since people suffering from quadriplegia paralysis symptoms may even face difficulties moving their heads as normal people do, the device that detects the head movement should be sensitive. The accelerometer can fit into this application.

In addition to cursor control, a person should be able to make selections, i.e. left button mouse click. Intentional eye blinking is used here for this purpose. A blink can be detected by a reflective object sensor.

The interfacing software module is a microcontroller unit that takes the movements sensed by the accelerometer and eye blinks sensed by the reflective object sensor and communicates with the computer through the PS/2 protocol typically used by the mouse.

This chapter includes information about the methods used to achieve the design and implement the project which are:

- PS/2 protocol as an interface to the computer.
- Accelerometer to detect the head movement used to control the mouse cursor on the screen of the computer.
- Reflective object sensor to detect the eye blink used for left button click of the mouse.
• Microcontroller unit (MCU) to perform analog to digital conversion (ADC) - since the signals generated by the sensors are analog and can’t be understood by computer - and provide PS2 emulation (sends/receives data packets to/from the computer).

3.2 PS/2 protocol

3.2.1 Introduction

All the computer peripherals need an interface to communicate with the computer. There are many interfaces available such as Universal Serial Bus (USB), Personal System/2 (PS/2) mouse interface, Apple Desktop Bus (ADB), RS-232 serial port and others.

The PS/2 mouse interface originally appeared in IBM's "Personal System/2" computers in the late 80's and it remains a widely supported interface, [1]. It utilizes a bidirectional serial protocol to transmit movement and button-state data from mouse to computer and commands from computer to mouse.

On the following sections the computer may be referred to as “host” and the mouse as “device”.

3.2.2 Mouse Inputs

The standard PS/2 mouse interface supports five inputs; X (right/left) movement, Y (up/down) movement, left button, middle button and right button. The mouse periodically reads these inputs and updates various counters and flags to reflect movement and button states, [14].

The standard mouse has two counters that keep track of movement; the X movement counter and the Y movement counter. These are 9-bit 2's complement values and each has an associated overflow flag. Their contents, along with the state of the three mouse buttons, are sent to the host in the form of a 3-byte movement data packet. The movement counters represent the mouse's offset relative to its position when the previous movement data packet was issued, [14].

When the mouse reads its inputs it records the current state of its buttons and increments/decrements the movement counters according to the amount of movement that has occurred since the last input sample. If either of the counters has overflowed, the appropriate overflow flag is set, [14].

3.2.3 Movement Data Packet

The standard PS/2 mouse sends movement/button information to the host using the 3-byte packet shown in table (3.1):
Table 3.1: PS/2 mouse movement data packet

<table>
<thead>
<tr>
<th></th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 1</td>
<td>Y overflow</td>
<td>X overflow</td>
<td>Y sign bit</td>
<td>X sign bit</td>
<td>Always 1</td>
<td>Middle button</td>
<td>Right button</td>
<td>Left button</td>
</tr>
<tr>
<td>Byte 2</td>
<td>X movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 3</td>
<td>Y movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The movement values are 9-bit 2’s complement integers, where the most significant bit appears as a “sign” bit in byte 1 of the movement data packet. Their value represents the mouse’s offset relative to its position when the previous packet was sent, in units determined by the current resolution. The range of values that can be expressed is -255 to +255. If this range is exceeded, the appropriate overflow bit is set, [14]

3.2.4 Resolution and Sample rate

3.2.4.1 Resolution

The resolution determines the amount by which the movement counters are incremented/decremented. The default resolution is 4 counts/mm and the host may change that value using the “Set Resolution” (0xE8) command, [14].

3.2.4.2 Sample rate

The mouse sends movement data when it detects a movement or a change in state of one or more mouse buttons. The maximum rate at which this data may be reported is known as the sample rate. This parameter ranges from 10-200 samples/sec, with a default value of 100 samples/sec. The host may set this value using the "Set Sample Rate" (0xF3) command, [14].

3.2.5 PS/2 protocol

The PS/2 protocol requires two lines to control the transmission of data between the host and the device. These two lines are the data line and clock line. They provide synchronous bidirectional serial communication protocol to send/receive data to/from the computer, [15].

The two lines can be referred to as the “bus”. According to the states of data line and clock line the bus can be found in one of three states:
Table 3.2: PS/2 Bus states

<table>
<thead>
<tr>
<th>Data</th>
<th>Clock</th>
<th>Bus State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
<td>Idle state</td>
<td>Device is able to send</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>Communication</td>
<td>The host ask the device to generate the clock in order to transmit data to device</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>Host Request-to-Send</td>
<td></td>
</tr>
</tbody>
</table>

The clock signal is always generated by the PS/2 device, but it is not continuous (it is only generated when data needs to be transmitted), [3]. The clock frequency is 10-16.7 kHz, [14].

The lines must be tri-state bi-directional, since at times they are driven by the mouse and at other times by the computer, [15].

The transmitted data is sent in frames, with each frame carrying a single byte of data.

3.2.5.1 Transmission from PS/2 device to host

The frame carrying data from device to host consists of 11 fields. The order of these fields is shown in table (3.3):

Table 3.3: PS/2 device to host frame format

<table>
<thead>
<tr>
<th>Start</th>
<th>Data 0</th>
<th>Data 1</th>
<th>Data 2</th>
<th>Data 3</th>
<th>Data 4</th>
<th>Data 5</th>
<th>Data 6</th>
<th>Data 7</th>
<th>Parity</th>
<th>Stop</th>
</tr>
</thead>
</table>

- Start: Indicates the start of a transmitted frame. This is always a '0'.
- Data 0-Data 7: The actual data bits. The least significant bit (Data 0) is always sent first.
- Parity: Used for error detection. Odd parity is used. Therefore, if the number of '1's in Data0-7 is even, the Parity bit is set (1). If the number of '1's is odd, the Parity bit is cleared (0).
- Stop: Indicates the end of the frame currently being transmitted. This is always a '1'.

The device writes a bit on the Data line when Clock is high, and it is read by the host when Clock is low,[14].

Figure (3.1) shows the timing for this frame transmission:
Chapter Three

3.2.5.2 Transmission from host to PS/2 device

Total control over transmission is ultimately in the hands of the host. If the host wants to send data, it must first put the Clock and Data lines in a "Request-to-send" state as follows:

- The clock line is first taken low for at least one clock period (entering Inhibit Transmission state).
- The data line is then taken low (providing the Start bit of the frame to be transmitted).
- The clock line is then released (still holding data low), [16].

When the device detects this state, it will begin generating Clock signals and clock in eight data bits and one stop bit. The host changes the Data line only when the Clock line is low, and data is read by the device when Clock is high. After the stop bit is received, the device will acknowledge the received byte by bringing the Data line low and generating one last clock pulse, then return to idle state, [14].

Figure (3.2) shows these steps graphically:

![Diagram of Host to device Communication](image-url)

**Figure 3.2: Host to device Communication, [13]**
Figure (3.3) separates the timing to show which signals are generated by the host, and which are generated by the PS/2 device. Notice the change in timing for the "ack" bit—the data transition occurs when the Clock line is high (rather than when it is low as is the case for the other 11 bits.), [14].

![Diagram of host to device communication](image)

**Figure 3.3: Detailed host to device communication**, [13]

Referring to figure (3.3), there are two time quantities the host looks for:

- (a) is the time it takes the device to begin generating clock pulses after the host initially takes the Clock line low, which must be no greater than 15 ms, [14].
- (b) is the time it takes for the packet to be sent, which must be no greater than 2 ms, [14].

Immediately after the "ack" is received, the host may bring the Clock line low to inhibit communication while it processes data. If the command sent by the host requires a response, that response must be received no later than 20 ms after the host releases the Clock line, [14].

For more information about PS/2 protocol see Appendix A.

### 3.3 Accelerometer

Acceleration is the rate of change of velocity (the derivative of velocity) with respect to time. It is a vector which has magnitude and direction relative to the axis of sensitivity. Its units are length/time², [17].

Accelerometer is an electromechanical device that will measure acceleration forces. These forces may be static, like the constant gravity force, or they could be dynamic; caused by moving or vibrating the accelerometer, [18]. An accelerometer senses motion of a surface to which it is attached, producing an electrical output signal precisely analogous to that motion, [19].
Accelorometers have two important specifications: dynamic range and sensitivity. Dynamic Range is the ± maximum amplitude that the accelerometer can measure before distorting or clipping the output signal. Typically specified in g's (1g is the acceleration due to the earth's gravity which is 9.8 m/sec²). Sensitivity is the output voltage produced by a certain force, its typical unit is mV/g. The output level will be proportional to the amplitude of the vibrations. Low output accelerometers are used to measure high vibrational levels while high output accelerometers are used to measure low level vibrations, [20].

There are many different types of accelerometers, such as capacitive micro-machined accelerometer, piezoelectric accelerometer, shear mode accelerometer and many others; each has its own principle of operation. The type that is used in this project is a capacitive micro-machined accelerometer.

- **Capacitive micro-machined accelerometer principle of operation**

  This accelerometer consists of two surface micro machined capacitive sensing cells (g-cell) and a signal conditioning Application Specific Integrated Circuits (ASIC) contained in a single integrated circuit package. The sensing elements are sealed hermetically at the wafer level using a bulk micro machined cap wafer, [21].

  The g-cell is a mechanical structure formed from semiconductor materials (polysilicon) using semiconductor processes (masking and etching). It can be modeled as a set of beams attached to a movable central mass that move between fixed beams. The movable beams can be deflected from their rest position by subjecting the system to acceleration, [21].

  As the beams attached to the central mass move, the distance from them to the fixed beams on one side will increase by the same amount that the distance to the fixed beams on the other side decreases. The change in distance is a measure of acceleration. The g-cell beams form two back-to-back capacitors. As the center beam moves with acceleration, the distance between the beams changes and each capacitor’s value will change, \( C = A\varepsilon/D \). Where A is the area of the beam, \( \varepsilon \) is the dielectric constant, and D is the distance between the beams, [21].

  The ASIC uses switched capacitor techniques to measure the g-cell capacitors and extract the acceleration data from the difference between the two capacitors. The ASIC also signal conditions and filters (switched capacitor) the signal, providing a high level output voltage that is proportional to acceleration, [21].
3.4 Reflective object sensor

A reflective object sensor is a device used to detect objects within its range. It consists of two modules:

- An emitting device that radiates light.
- A phototransistor that responds when light radiated by the emitting device is reflected from an in-range object.

The emitting device is usually a light emitting diode (LED) that produces infrared (IR) light. IR is used to maximize the signal-to-noise ratio as the silicon phototransistors have a good response to IR.

The phototransistor is a transistor that regulates current or switches it on and off based on the intensity of the light it is exposed to rather than an external electric signal, [22].

The phototransistor principle of operation is based on the photoelectric effect (exposure to high energy photons will produce current due to the electrons freed after absorbing the energy of photons).
So when the light strikes the phototransistor base, a current will flow and that’s how light is converted to current.

### 3.5 Microcontroller unit

A Microcontroller unit (MCU) is a computer-on-a-chip. It is a type of microprocessor emphasizing high integration, low power consumption, self-sufficiency and cost-effectiveness, in contrast to a general-purpose microprocessor (the kind used in a PC). In addition to the usual arithmetic and logic elements of a general purpose microprocessor, the microcontroller typically integrates additional elements such as read-write memory for data storage, read-only memory, such as flash for code storage, Electrically Erasable Programmable Read Only Memory (EEPROM) for permanent data storage, peripheral devices, and input/output interfaces. At clock speeds of as little as a few MHz or even lower, microcontrollers often operate at very low speed compared to modern day microprocessors, but this is adequate for typical applications. They consume relatively little power (milliwatts), and will generally have the ability to sleep while waiting for an interesting peripheral event such as a button press to wake them up again to do something. Power consumption while sleeping may be just Nano watts, making them ideal for low power and long lasting battery applications. Microcontrollers are frequently used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, appliances, power tools, and toys. By reducing the size, cost, and power consumption compared to a design using a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to electronically control many more processes, [23].

Microcontrollers were originally programmed only in assembly language, but various high-level programming languages are now also in common use to target microcontrollers. These
languages are either designed especially for the purpose, or versions of general purpose languages such as the C programming language, [10].
Chapter Four

Implementation

4.1 Overview

As mentioned on chapter three “Methodology chapter”, design of the project is based on two components; hardware component and software component. The hardware component contains four main parts:

- Microcontroller unit.
- Accelerometer.
- Reflective object sensor.
- PS/2 to USB converter.

While the software component performs two tasks:

- Takes data from the sensors – accelerometer and reflective object sensor- and performs analog to digital conversion (ADC).
- Puts the converted analog data into a format suitable for emulation of PS/2 protocol (exchange of movement data and commands with the computer).

On this chapter, the detailed information about the design of these two components is illustrated.

Head Movement detection

Eye blink detection

Figure 4.1: General design
4.2 The hardware component

The microcontroller unit used is Atmel AVR ATmega32. It provides analog to digital conversion with noise reduction so it gives accurate results for conversion of sensed signals. It operates with 4.5V - 5.5V. (For pin configuration see Appendix B).

Freescale Semiconductor MMA7260 accelerometer is used. It is a three axes accelerometer with four selectable dynamic ranges (1.5g, 2g, 4g & 6g). As noted by Freescale Semiconductor it is typically used for motion sensing applications. (For pin configuration see Appendix B).

1.5g dynamic range is used as it provides the highest sensitivity with an output of 800 mV/g giving accurate measurements for the head movement (The other sensitivities 2g, 4g and 6g give 600,300 and 200 mv/g respectively). For emulation of mouse movement only two axes are required, hence only X and Y axes are used. It operates with 2.2 V – 3.6 V.

The reflective object sensor used is Optek Technology OPB608A. It has a light emitting diode and a phototransistor output. It is used for near field applications. (For pin configuration see Appendix B).

PS/2 protocol is used as an interface. Since USB is more common on modern computers, PS/2 to USB signal converter is used. This signal converter uses an integrated circuit (pre-programmed chip) to actively translate the PS/2 signal and convert it into a USB signal. This allows the PS/2 device to be automatically recognized by the operating system as if it were a standard USB device. The use of converter makes the device hot-pluggable (i.e. the device is connected/disconnected while the computer is turned on).

The physical PS/2 port has the pins arrangement shown in figure (4.2):

![Figure 4.2: PS/2 socket](image)

With the pins assignment as in table (4.1):

<table>
<thead>
<tr>
<th>Pin 1</th>
<th>Pin 2</th>
<th>Pin 3</th>
<th>Pin 4</th>
<th>Pin 5</th>
<th>Pin 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Not connected</td>
<td>Ground</td>
<td>Vcc (+5V)</td>
<td>Clock</td>
<td>Not connected</td>
</tr>
</tbody>
</table>
4.2.1 Procedures

- The power required for operating the hardware unit is supplied from the computer through the PS/2 to USB converter; it provides a voltage of +5V (Vcc) and current of 100mA.

- The X axis and Y axis output pins of MMA7260 are connected respectively to pins 0 and 1 of PORT A on ATmega32 in order to perform analog to digital conversion (ADC). 1.5g sensitivity is selected by grounding g-select1 and g-select2. Sleep mode (low power consumption for battery saving) isn’t used, so sleep mode pin (active low) is connected to Vcc. The MMA7260 operates with 2.2V – 3.6V. The Vcc of +5V is down converted to 3V using a voltage divider using two resistors.

- Output of the reflective object sensor is taken from the collector of the phototransistor, amplified and connected to pin 2 on PORT A.

- PORT C on ATmega32 is used for the bidirectional data and clock lines used by PS/2 interface (refer to Chapter three “Methodology”, section 3.2.5) with the pin assignment shown in table (4.2):

<table>
<thead>
<tr>
<th>Pin 0</th>
<th>Data coming from host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>Data generated by device</td>
</tr>
<tr>
<td>Pin 2</td>
<td>Clock controlled by host</td>
</tr>
<tr>
<td>Pin 3</td>
<td>Clock generated by device</td>
</tr>
</tbody>
</table>

- The bidirectional data and clock lines are implemented using resistors.

- Two light emitting diodes (LEDs) are connected to pin 5 & 7 on PORT C for debugging purpose to ensure that the device is responding.

4.3 The software component

The software carries out three functions:

- Provides PS/2 emulation (exchange of movement data and commands with the computer) through functions to send data to host and functions to receive data from host.

- Initialize the connection with the computer through the hand shaking process (initialization process).
• Run in stream mode where the microcontroller takes data from the sensors – accelerometer and reflective object sensor- and performs analog to digital conversion (ADC), and use the converted values to implement the function of a real mouse.

The software is written in C programming language with CodeVisionAVR as a helping tool. It is recommended to refer to Chapter three “Methodology”, section 3.2.5 for information about the PS/2 protocol as well as Appendix A for the commands used by host and device.

4.3.1 PS/2 emulation

For the PS/2 emulation functions to exchange data and commands between the host and device are implemented. These functions are grouped into two types:

• Functions to send data to host.
• Functions to receive data from host.

The clock is always generated by the device and it isn’t continuous; it is only generated when data needs to be transmitted. The clock frequency in use is 10 KHz.

4.3.1.1 Functions to send data to host

These are five functions:

1- send_frame:
   • It is used to send data frame of 11 bits to host.
   • It checks the bus state , if the bus is idle for 50µs it will send the frame of 11 bits (1 start bit, 8 data bits, 1 parity bit, and 1 stop bit) to the host.
   • It contains the following functions (send_start_bit, send_data_byte, _send_parity, and send_stop_bit are all included within it)

2- send_start_bit:
   • Generates 1 clock cycle needed for transmission of start bit.
   • Generates the start bit (which is 0) on the middle of the high half of clock cycle (device writes on data line when clock is high as imposed by PS/2 protocol).

3- send_data_byte:
   • Generates 8 clock cycles needed for transmission of data byte which can be status byte, X movement, or Y movement.
   • Toggle the value of data line according to the bits of the data byte on the middle of the high half of each clock cycle.
   • Calculates the odd parity of transmitted data.
4- send_parity:
   • Generates 1 clock cycle needed for transmission of parity bit.
   • Sends parity bit calculated by send_data_byte function on the middle of the high half of clock cycle.

5- send_stop_bit:
   • Generates 1 clock cycle needed for transmission of stop bit.
   • Generates the stop bit (which is 1) on the middle of the high half of clock cycle.

Figure (4.3) shows the timing for each function.

Figure 4.3: Functions to send data to host

4.3.1.2 Functions to receive data from host

These are two functions:

1- read_data_parity:
   • Generates clock to allow the host to send the 8 bits command.
   • Reads each bit on the middle of the high half of each clock cycle (device reads data when clock is high as imposed by PS/2 protocol).

2- read_frame:
   • Generates clock for host.
   • Reads command sent by host using read_data_parity() function.
   • Acknowledge the host that the command is received.
- Return to idle state.

### 4.3.2 Hand shaking process (initialization process)

The device starts the hand shake process by sending two data frames to the host, the host receives these frames and starts sending commands to the device. The hand shake executes until the device receives the enable command from host.

The hand shake process is done using an external interrupt activated every time the host pulls the clock line low to inhibit the communication and releases it asking the device to start generation of clock in order to send its commands. When the device receives the enable command the external interrupt is disabled; since the hand shake process is finished and the device can enter the stream mode in which it sends data and performs the normal tasks of the mouse.

Figure (4.4) shows the steps of the hand shaking process.

### 4.3.3 Running in stream mode

In this mode, the device performs its main function of sending the movement data packets. The Microcontroller enters this mode when it receives the enable command from the host.

When data generated from accelerometer and reflective object sensor, the microcontroller converts them into digital signals using the function “read_adc()”. CodeVisionAVR provides this function for ADC from any of the 8 pins of PORT A. ATmega32 is able to convert an analog signal to a digital signal with two resolution options; 10bits and 8 bits. Since the PS/2 X and Y movement counters can handle a byte each, the 8 bits resolution is used.

What happens in stream mode can be summarized as follow:

- After entering the stream mode, the microcontroller takes the first hundred readings from the sensors and calculate the initial references as the average of these readings.
- When the accelerometer is tilted left/right or up/down the corresponding digital value is measured against the reference and compared with a threshold in order to:
  1- Update the movement counters to determine the amount of movement and speed state (fast, medium, or slow). If the difference between the present reading and the reference is less than the threshold they will be no movement.
  2- Update the status byte to indicate the direction of movement (left/right and up/down).
  3- Update the references for use in next reading.
• The signal detected from reflective object sensor is compared with the reference to decide if an intentional eye blink is made or not — also there is a threshold — and the status byte is updated accordingly (no update for reference).

• Send the status byte along with X movement and Y movement counters to the host.

Figure (4.5) shows the determination of the X movement, it is also applied to Y movement. Figure (4.6) shows the blink detection.

See Appendix C for the entire code listing.
Figure 4.4: Hand shaking process
Figure 4.5: Determination of X movement
Figure 4.6: Blink detection
Chapter Five

Results and discussion

5.1 Results

The final device didn’t give the expected results. There was a problem with the initialization process; the computer didn’t give the enable command needed to enter the stream mode, or the computer couldn’t detect it.

Although the final device faced difficulties establishing communication with computer, a simulation of the PS/2 protocol has been made between the device and another microcontroller that represents the computer.

The simulation environment is shown in figure (5.1).

![Simulation Environment Diagram]
The simulation is used to determine the shape of the data frame sent by the device. The sequence 0xAA and 0x00 is sent to start the hand shaking process, it has been captured as shown in figure (5.2).

Figure 5.2: Above are the frames 0xAA and 0x00, below is the clock

As shown in figure (5.2), the data value is changed when the clock is high as required by the PS/2 protocol. First the start bit (0) is sent, then the sequence 10101010 which represents the command 0xAA is sent (least significant bit 0), the parity bit (odd parity) is calculated correctly (1) since the number of 1's in the command is even. Then the stop bit (1) is sent and that determines that the bus is back to idle state. Same happens for sequence 0x00.

The same sequence was obtained practically using the circuit implemented for the device.

Also some analog signals represent the X and Y movements are taken and the status byte is calculated accordingly. It is shown in figure (5.3).
5.2 Discussion

- The final device couldn’t give the expected results; it couldn’t be recognized by the computer.
- The biggest obstacle that faced the implementation of the project is the massive delay due to the unavailability of the sensors which are the bottleneck components on the design of the project. These sensors have been bought from abroad.
- Another issue is limitation found on the reflective object sensor, its field view is very short; it’s only 9.525 mm. Hence it couldn’t be used; because it will not be convenient to have such a close object to the eye.
- The value of the move threshold was just estimated, the exact value can only be found using practical testing.

Figure 5.3: Some data frames sent to the host

To implement the whole protocol the microcontroller unit that represents the computer must be programmed.
Chapter Six

Conclusion and Future work

6.1 Conclusion

The project basically aims to help the people who are severely disabled such as amputees and people with quadriplegic paralysis use the computer; since it’s the basic tool nowadays for work, communication, education, and many other useful things. Those people face difficulties use the computer because there are no many hardware tools that they can use with ease to control and interact with the computer. So the project tries to implement a computer input device to improve their quality of lives and enrich it.

This input device provides the functions of the ordinary mouse; which is the control of the cursor on the computer’s display and making selections. Those severely disabled people may use their heads to move the cursor and blink their eyes to select an item on the screen.

The device uses two sensors, one for head movement detection, this is an accelerometer, and another sensor to detect the eye blinking, and this is a reflective object sensor. The sensed signals are passed to a microcontroller unit which converts them to digital signals and sends them to the computer using the personal system/2 (PS/2) protocol, so the computer recognizes these data as a usual PS/2 mouse. A PS/2 to USB (Universal Serial Bus) converter is used so as to use USB ports instead of PS/2 ones which are rarely found on the computers nowadays.

The microcontroller unit is programmed to perform analog to digital conversion of the sensed data and interact with the computer through the PS/2 protocol, which has its own requirements, e.g. data frame formats, frequency, etc. so, the functions that implements the protocol have been implemented.

The final device hasn’t worked as expected; it faced difficulties being recognized by the computer.

One of the limitations of the approach used to implement the project is that it is a wired device. It also uses the PS/2 protocol which is not supported on many recent computers. Another issue is that the reflective object sensor uses infrared light, although it may not be tense enough to damage the eye, it is still a risk to use it.
6.2 Future work

Suggestions for future work include:

- Use wireless transmission. The design may contain two units; a head unit contains the sensors, a microcontroller unit (for analog to digital conversion) as well as a wireless transmitter, and a ground unit contains a wireless receiver and a microcontroller (for PS/2 implementation).

This design will offer the comfort of wireless transmission but will put an extra cost and requires more software manipulations to design a wireless protocol.

![Figure 6.1: Suggestion for wireless input device](image)

Use the USB protocol instead of PS/2. This will give support for use with wider range of computers. But requires more complex coding.

- Use another selection mechanism rather than the eye blinking detection; e.g. sound.
- The device can be improved to be used with other devices, such as a motorized wheelchair. The microcontroller can be programmed to generate signals to direct the wheelchair, so the disabled people may even move without help of others.
References


   http://www.webopedia.com/TERM/M/mouse.html


   http://www.humanillnesses.com/original/Pan-Pre/Paralysis.html

   http://www.cure-back-pain.org/paralysis.html


[8] National institute of biomedical imaging and bioengineering.
   http://www.nibib.nih.gov/HealthEdu/eAdvances/26May10

   http://www.sciencedaily.com/releases/2009/06/090629081137.htm


   http://www.pva.org/site/News2?page=NewsArticle&id=14579

http://www.computer-engineering.org/ps2mouse/
Author: Adam Chapweske


[16] Altium PS2 controller, core reference, CR0109 (v1.0) February 04, 2004


[18] Dimension engineering.
http://www.dimensionengineering.com/accelerometers.htm


http://www.omega.com/prodinfo/accelerometers.html

[21] Freescale Semiconductor, Technical Data, “±1.5g - 6g Three Axis Low-g Micromachined Accelerometer”, Document Number: MMA7260QT, Rev 5, 03/2008

[22] The free dictionary.
http://www.thefreedictionary.com/phototransistor

http://www.bookrags.com/wiki/Microcontroller
Appendix A

PS/2 protocol

Mouse modes of operation:

Data reporting (sending of movement data packets) is handled according to the mode in which the mouse is operating. There are four modes of operation:

- Reset mode: This is the initial mode, in which the mouse performs initialization and self-diagnostics.
- Stream mode: The default operating mode, in which the mouse issues movement data packets when movement occurs or button state changes.
- Remote mode: The host must poll for movement data packets.
- Wrap mode: A purely diagnostic mode where the mouse echoes every received packet back to the host.

On this project only the Reset mode and stream mode are implemented as the other modes are of no need and they are rarely used.

Reset Mode:

The mouse enters reset mode at power-on or in response to the "Reset" (0xFF) command. Upon entering this mode, the mouse performs a diagnostic self-test called BAT (Basic Assurance Test) and sets the following default values:

- Sample Rate = 100 samples/sec
- Resolution = 4 counts/mm
- Scaling = 1:1
- Data Reporting = disabled

The mouse then sends a BAT completion code of either 0xAA (BAT successful) or 0xFC (Error). The host's response to a completion code other than 0xAA is undefined.

Following the BAT completion code (0xAA or 0xFC), the mouse sends its device ID of 0x00. This distinguishes it from a keyboard or nonstandard mouse.

Once the mouse has sent its device ID to the host, it enters stream mode.
Stream Mode:

In stream mode the mouse sends movement data when it detects movement or a change in state of one or more mouse buttons according to the sample rate of 100 samples/sec that was set on the reset mode. The host may change the sample rate using the "Set Sample Rate" (0xF3) command.

Note that reporting is disabled by default. The mouse will not actually issue any movement data packets until it receives the "Enable Data Reporting" (0xF4) command.

Stream mode is the default operating mode, and is otherwise set using the "Set Stream Mode" (0xEA) command.

 Commands set:

Commands sent to mouse:

<table>
<thead>
<tr>
<th>Commands Sent to Mouse</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset Mouse</td>
<td>FF</td>
</tr>
<tr>
<td>Mouse returns AA, 00 after self-test</td>
<td></td>
</tr>
<tr>
<td>Resend Message</td>
<td>FE</td>
</tr>
<tr>
<td>Set to Default Values</td>
<td>F6</td>
</tr>
<tr>
<td>Enable Streaming Mode</td>
<td>F4</td>
</tr>
<tr>
<td>Mouse starts sending data packets at default rate</td>
<td></td>
</tr>
<tr>
<td>Disable Streaming Mode</td>
<td>F5</td>
</tr>
<tr>
<td>Set sampling rate</td>
<td>F3, XX</td>
</tr>
<tr>
<td>XX is number of packets per second</td>
<td></td>
</tr>
<tr>
<td>Read Device Type</td>
<td>F2</td>
</tr>
<tr>
<td>Set Remote Mode</td>
<td>EE</td>
</tr>
<tr>
<td>Set Wrap Mode</td>
<td>EC</td>
</tr>
<tr>
<td>Mouse returns data sent by system</td>
<td></td>
</tr>
<tr>
<td>Read Remote Data</td>
<td>EB</td>
</tr>
<tr>
<td>Mouse sends 1 data packet</td>
<td></td>
</tr>
<tr>
<td>Set Stream Mode</td>
<td>EA</td>
</tr>
<tr>
<td>Status Request</td>
<td>E9</td>
</tr>
<tr>
<td>Mouse returns 3-bytes with current settings</td>
<td></td>
</tr>
<tr>
<td>Set Resolution</td>
<td>E8, XX</td>
</tr>
<tr>
<td>XX is 0, 1, 2, 3</td>
<td></td>
</tr>
<tr>
<td>Set Scaling 2 to 1</td>
<td>E7</td>
</tr>
<tr>
<td>Reset Scaling</td>
<td>E6</td>
</tr>
</tbody>
</table>
Messages sent by mouse:

<table>
<thead>
<tr>
<th>Messages Sent by Mouse</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resend Message</td>
<td>FE</td>
</tr>
<tr>
<td>Two bad messages in a row</td>
<td>FC</td>
</tr>
<tr>
<td>Mouse Acknowledge Command</td>
<td>FA</td>
</tr>
<tr>
<td>Sent by Mouse after each command byte</td>
<td></td>
</tr>
<tr>
<td>Mouse passed self-test</td>
<td>AA</td>
</tr>
</tbody>
</table>

Initialization:

The PS/2 mouse is normally detected/initialized only when the computer is booting up. That is, the mouse is not hot-pluggable and the computer must be restarted whenever a PS/2 mouse is added/removed. Adding/removing the PS/2 mouse while the computer is running may physically damage some motherboards. The following is the communication between computer and a standard PS/2 mouse during the boot process. It is fairly typical of how a PS/2 mouse is initialized.

Power-on Reset:
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host: FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host: FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host: FF Reset command
Mouse: FA Acknowledge
Mouse: AA Self-test passed
Mouse: 00 Mouse ID
Host: F3 Set Sample Rate: Attempt to Enter Microsoft
Mouse: FA Acknowledge: Scrolling Mouse mode
Host: C8 decimal 200:
Mouse: FA Acknowledge:
Host: F3 Set Sample Rate
Mouse: FA Acknowledge
Host: 64 decimal 100
Mouse: FA Acknowledge
Host: F3 Set Sample Rate
Mouse: FA Acknowledge
Host: 50 decimal 80
Mouse: FA Acknowledge
Host: F2 Read Device Type
Mouse: FA Acknowledge
Mouse: 00 Mouse ID: Response 03 if Microsoft scrolling mouse
Host: F3 Set Sample Rate
Mouse: FA Acknowledge
Host: 0A decimal 10
Mouse: FA Acknowledge
Host: F2 Read Device Type
Mouse: FA Acknowledge
Mouse: 00 Mouse ID
Host: E8 Set resolution
Mouse: FA Acknowledge
Host: 03 8 Counts/mm
Mouse: FA Acknowledge
Host: E6 Set Scaling 1:1
Mouse: FA Acknowledge
Host: F3 Set Sample Rate
Mouse: FA Acknowledge
Host: 28 decimal 40
Mouse: FA Acknowledge
Host: F4 Enable
Mouse: FA Acknowledge
Initialization complete...
Appendix B:

Data sheets

ATmega 32 pin configuration

MMA7260 Pin configuration
<table>
<thead>
<tr>
<th>Pin #</th>
<th>LED</th>
<th>Pin #</th>
<th>Transistor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Cathode</td>
<td>1</td>
<td>Collector</td>
</tr>
<tr>
<td>3</td>
<td>Anode</td>
<td>2</td>
<td>Emitter</td>
</tr>
</tbody>
</table>

OPB608A Pin configuration
Appendix C:

Program listing

C-1
{ //the interrupt occurred due to falling edge  TCNT0=0; //reset the counter, start counting from the falling edge  localMCUCR^=0x01;  //change the interrupt detection mode (Rising=0x03, Falling=0x02)  } //the interrupt occurred due to rising edge
else
{  current_time=TCNT0; //current time is time from last falling edge to this rising edge  period=current_time*MY_CLOCK_PERIOD; //Multiplied by 1 because the clock period is 1us, used timer frequency is 1MHz
  if (period>=100)
  { //that means the host pulled the clock low and inhibited connection  GICR=0x00; //Interrupt 0 is Disabled  data=read_frame(); //read data from host
    if (data==0xFF)
      { delay_ms(400); //device should respond between 300ms and 500ms as stated by the protocol, this is just for the reset command
        send_frame(0xFA); //Acknowledge
        send_frame(0xAA); //Self test passed
        send_frame(0x00); //Mouse ID
        GICR=0x04; //Enable Interrupt 0
      }
    else if ((data==0xF3)||(data==0xC8)||(data==0x64)||(data==0x50)||(data==0x0A)||(data==0xE8)||(data==0x03)||(data==0xE6)||
      { send_frame(0xFA); // Acknowledge
        GICR=0x04; // Enable Interrupt 0
      }
    } else if ((data==0xF2)
      { //get device ID
        send_frame(0xFA); //Acknowledgement
        send_frame(0x00); //Mouse ID
        x_ref=0b00000000; //reset the mouse
        y_ref=0b00000000; //movement counters
        GICR=0x00; //Enable Interrupt 0
      }
    else if (data==0xF4)
      {
        x_ref=0b00000000; //reset the mouse
        y_ref=0b00000000; //movement counters
        GICR=0x00; //Enable Interrupt 0
      }
    } //enable command  send_frame(0xFA); //Acknowledgement  for (i=0;i<5;i++)
      {  LED1=1;
        LED2=1;
        delay_ms(500);
        LED1=0;
        LED2=0;
      };
    GICR=0x00; //disable Interrupt 0, initialization completed, no more interrupts allowed
void send_start_bit()
{
    // function for sending start bit
    cout=high;
    delay_us(25); // Data line should be modified in the middle of each clock cell
    dout=low;     // start bit=0
    delay_us(25);
    cout=low;     // frequency used is 10KHz, so the clock should remain high for 50us & low for the same period
    delay_us(50);
}

int send_data_byte(unsigned char command){
    // function for sending data byte (can be X or Y counters, or the status byte, or a command)
    int i,p=1; // i used for the loop, p for parity calculation
    unsigned char byte,sb; // both variables are temporary variables used for shifting and sending data bits
    byte=command;
    for (i=0;i<8;i++)
    {
        sb=byte;
        sb = sb& 0x01;
        if(sb==0x01)
        {
            cout=high;
            delay_us(25);
            dout=high; // data bit = 1
            delay_us(25);
            cout=low;
            delay_us(50);
            p=p^1;
        } else
        {
            cout=high;
            delay_us(25);
            dout=low; // data bit = 0
            delay_us(25);
            cout=low;
            delay_us(50);
            p=p^0;
        } // end of else
        byte=byte>>1; // shifts the byte to right to read next bit
    } // end of for loop
    return (p); // end of "send data byte" function
}

void send_parity (int parity){
    // function to send parity bit
    if(parity==1)
    {
        cout=high;
        delay_us(25);
        dout=high;  // parity bit = 1
        delay_us(25);
        cout=low;
        delay_us(50);
    } else
    {
        cout=high;
        delay_us(25);
        dout=low;   // parity bit = 0
        delay_us(25);
        cout=low;
        delay_us(50);
    }
}

void send_stop_bit()
{
    // function for sending stop bit
    cout=high;
    delay_us(25);
}

void send_frame(unsigned char command){  // sends the whole frame consists of : start bit, 8 data bits, parity bit then stop bit
int parity;
if((cout==high)&(dout==high))
{
    // check for idle state
    delay_us(50);  //device should ensure that bus is idle for 50us before it starts transmission
    if((cout==high)&(dout==high))
    {
        LED1=1;
        send_start_bit();
        parity=send_data_byte(command);
        send_parity(parity);
        send_stop_bit();
        cout=high;
        LED1=0;
    }
}
}

/functions to send data to host
*******************************************************************************/
unsigned char read_data_parity()
{
    // reads sent data bits and calculates parity
    int i,p=1;
    unsigned char byte=0x00,temp;   // temp is a temporary variable
    for(i=0;i<8;i++)
    {
        cout=low;
        delay_us(50);
        cout=high;
        //mouse should read the data when clock is high
        delay_us(25);
        if (din==1)
        {
            // read data bit, if data=1 , do the following loop , else do the other
            temp=0x01;
            byte=byte+(temp<i);    // this line is used to read data using shifting
            p=p^1;               // to calculate the parity of data byte (p xor din)
            delay_us(25);
        } else
        {
            temp=0x00;
            byte=byte+(temp<i);    // this line is used to read data using shifting
            p=p^0;               // to calculate the parity of data byte
            delay_us(25);
        }
    }
    return (byte);   // data byte that will be returned to "read frame function"
}

unsigned char read_frame()
{
    // function to read the sent frame from host , the mouse should generate clock for the host
    unsigned char data;
    LED2=1;
    data= read_data_parity();  // a function that reads sent data from host
    cout=low;
    delay_us(50);  // start of parity bit
    cout=high;
    delay_us(50);
    cout=low;       // start of stop bit
    delay_us(50);
}
cout=high;
delay_us(25);
dout=low;                      //the device will acknowledge the received byte by bringing
delay_us(25);                 //the Data line low and generating one last clock pulse.
cout=low;
delay_us(50);
cout=high;                    //return to idle state
dout=high;

LED2=0;
return data;                   // data byte that will be returned to the main function
}

/**********************************************
ADC function done by wizard
***********************************************/
// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
    ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
    // Delay needed for the stabilization of the ADC input voltage
delay_us(10);
    // Start the AD conversion
    ADCSRA|=0x40;
    // Wait for the AD conversion to complete
    while ((ADCSRA & 0x10)==0);
    ADCSRA|=0x10;
    return ADCH;
}

/**********************************************
Main function
***********************************************/
// Declare your global variables here
void main(void)
{
    // Declare your local variables here
    unsigned char x_mov,y_mov, blink;          // these represent the X & Y counters
    char count=0;                       //used to calculate the initial reference
    // Input/Output Ports initialization
    // Port A initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTA=0x00;
    DDRA=0x00;
    // Port B initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTB=0x33;
    DDRB=0xFF;
    // Port C initialization
    // Func7=Out Func6=In Func5=Out Func4=In Func3=Out Func2=In Func1=Out Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTC=0xAA;
    DDRC=0xAA;
    LED2=0;
    // Port D initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTD=0x00;
    DDRD=0x00;
    // Timer/Counter 0 initialization
}
// Clock source: System Clock
// Clock value: 1000.000 kHz
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCR0=0x01;
TCNT0=0x00;
OCR0=0x80;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFF
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICRH=0x00;
ICRL=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Falling Edge
// INT1: Off
// INT2: Off
GICR|=0x40;
MCUCR=0x02;
MCUCSR=0x00;
GIFR=0x40;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC Clock frequency: 7.813 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x87;
// ADC initialization
// ADC Clock frequency: 500.000 kHz
// ADC Voltage Reference: AREF pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x81;
// Global enable interrupts
#asm("sei")

Initialization Initialization Initialization Initialization Initialization Initialization
*******************************************************************************/
delay_ms(500);                 //time needed to perform power-on self-test
GICR=0x00;                      //Interrupt 0 is Disabled
send_frame(0xAA) ;              // Self test passed
send_frame(0x00);               // Mouse ID
GICR=0x40;                      //Interrupt 0 is enabled
delay_ms(10000);                //stop the program from entering the while, until the host is allowed to initialize the mouse, 10
senconds should be enough
 Initialization complete Initialization complete Initialization complete
*******************************************************************************/
While loop : where the program stays forever
*******************************************************************************
if (data==0xFA) //don’t send movement data packets unless the enable command is received
{
    while (1)
    {
        x=read_adc(0);                                    // converted data from ADC.0 represents X-movements
        y=read_adc(1);                                    // converted data from ADC.1 represents Y-movements
        blink=read_adc(2);

        if(count < 100)
        {
            x_ref=x_ref+x;                               // AN
            y_ref=y_ref+y;                               // VALUE
            blink_ref=blink_ref+blink;
            count++;                                     // FOR
        }
        if(count==100)
        {
            x_ref = (int)(x_ref/100);                    // "INITIAL" REFERENCES
            y_ref = (int)(y_ref/100);                    // FROM 100
            blink_ref= (int)(blink_ref/100);
            count++;                                     // SUCCESIVE READINGS
        }
        if(count > 100)
        {
            if(x<x_ref)
            {
                status = status || 0b00011000 ;              // xsign=1
                if(x-x_ref > threshold*3)
                    x_mov = 9;                                    // fast movement
                else if(x-x_ref > threshold*2)
                    x_mov = 6;                                    // medium movement
                else if (x-x_ref>threshold)
                    x_mov=3;                                     // slow movement
                else x_mov = 0;                                // no movement
            }
            else if(x>x_ref)
            {
                status = status || 0b00001000 ;              // xsign=0
                if(x-x_ref > threshold*3)
                    x_mov = 9;                                    // fast movement
                else if(x-x_ref > threshold*2)
                    x_mov = 6;                                    // medium movement
                else x_mov = 0;                                // no movement
            }
        }
    }
}
x_mov = 6;
else if (x-x_ref>threshold)
    x_mov=3;
else x_mov = 0;

x_ref=x;
} else {
    status = status || 0b00001000 ;  //xsign=0
    x_mov=0x00;
}

if(y<y_ref) {
    // Y-axis
    status = status || 0b00011000 ;  // ysign=1
    if(y_ref-y > threshold*3)
        y_mov = 9;
    else if(y_ref-y > threshold*2)
        y_mov = 6;
    else if (y_ref-y>threshold)
        y_mov=3;
    else y_mov = 0;

    y_ref=y;
} else if(y>y_ref) {
    status = status || 0b00001000 ;  // ysign=0
    if(y-y_ref > threshold*3)
        y_mov = 9;
    else if(y-y_ref > threshold*2)
        y_mov = 6;
    else if (y-y_ref>threshold)
        y_mov=3;
    else y_mov = 0;

    y_ref=y;
} else {
    status = status || 0b00001000 ;  // ysign=0
    y_mov=0x00;
}

if (blink>blink_ref)                          //blink detected
    status= status||0b00000001;
send_frame(status);                          // send Status Byte
send_frame(x_mov);                          // send X-axis Byte
send_frame(y_mov);                          // send Y-axis Byte

} delay_ms(10) ;
}