COMPUTER INPUT DEVICE FOR USERS WITH SEVERE DISABILITIES

BY
NADIR TAWFIG MOHAMED MOHAMED ALI
INDEX NO. 064084

Supervisor
Dr. Sharief F. Babikir

A THESIS SUBMITTED TO
University of Khartoum
In partial fulfillment of the requirement for the degree of
B.Sc. (HON) in Electrical and Electronics Engineering
(ELECTRONICS AND COMPUTER ENGINEERING)
Faculty of Engineering
Department of Electrical and Electronics Engineering
July 2011
Abstract

This project aims to help people who lost the ability of using their hands, due to paralysis or amputation, to use the computer, which used to be impossible because of their disability to use the standard input unit, the mouse. In order to enable these users to use the computer, a sensor called Accelerometer that measures acceleration is used to detect the movement of the head of user by putting this sensor on his head. This movement will be an emulation of the mouse movement that directs the cursor on the screen, thereby enabling the user to move the cursor on the Graphical User Interface that provides icons and menus and many graphical features. In order to enable user to choose these icons, a reflective object sensor will be used to detect the blink of the user’s eye which will emulate clicks of the mouse.

Both of the Sensors – Accelerometer and reflective object sensor – will be connected to a microcontroller that will have the output of these sensors as inputs, which will use these inputs to create the required movement data packet for the PS/2 (Personal System 2) protocol that is used to communicate with the computer.
المستخلص

هذا المشروع يهدف إلى مساعدة الأشخاص الذين فقدوا القدرة على استخدام أدواتهم، وتمكينهم من استخدام الكمبيوتر، وذلك استنادًا إلى استخدامهم له بسباعتهم وكم قدرتهم على استخدام إحدى وسائل الإدخال الأساسية، وهي الفارة (الماوس). من أجل تمكينهم، الأشخاص من استخدام الكمبيوتر، يتم استخدام جهاز التسجيل للاستخدام بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامه. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامه. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدرتهم على استخدامهم. ومع ذلك، يتم استخدام الكمبيوتر بناءً على القدرة على استخدامهم. وسيكون هذا الجهاز هادفًا لاستخدام كمبيوتر بواسطة الأشخاص الذين يشعرون بببساطتهم وكم قدير
Dedication

I want to dedicate this report to:

My Mother…

My Father…

My siblings…

&My colleagues…
Acknowledgement

All thanks and gratitude and thanks to Allah for giving me the power and the knowledge to do this honorable task. I would like to thank my supervisor, Dr. Sharief Fadul Babikir, for his unlimited support and for his wise guidance.

I would also like to thank Mr. Abdulkareem Elbadri for his dedication and for his assistance through the whole phases of the project. I would like to thank my pair, Mossab Awad Allah, for being a great partner, colleague and friend.

Last but not least, I would like to thank Mustafa Ibrahim for his precious tips in writing this report, which were really valuable.
# Table of Contents

Abstract ........................................................................................................................................... i

Dedication ......................................................................................................................................... iii

Acknowledgement ............................................................................................................................. iv

Table of Contents ............................................................................................................................. v

List of Figures ..................................................................................................................................... viii

List of Tables ....................................................................................................................................... ix

List of Abbreviations .......................................................................................................................... x

1 CHAPTER 1 INTRODUCTION ........................................................................................................ 1
  1.1 Overview ................................................................................................................................... 1
  1.2 Motivation .................................................................................................................................. 1
  1.3 Objective .................................................................................................................................... 2
  1.4 Approach ................................................................................................................................... 2
  1.5 Thesis Layout .............................................................................................................................. 3

2 CHAPTER 2 HCI & USER INTERFACES FOR HANDICAPPED USERS ...................................... 4
  2.1 Overview ................................................................................................................................... 4
  2.2 Human- Computer Interaction (HCI) ......................................................................................... 4
    2.2.1 Definitions .......................................................................................................................... 4
    2.2.2 Goals of HCI ...................................................................................................................... 5
    2.2.3 Mouse ................................................................................................................................ 5
  2.3 Paralysis ...................................................................................................................................... 7
    2.3.1 Types ................................................................................................................................... 7
    2.3.2 Common Causes ................................................................................................................... 7
  2.4 Techniques used to help paralyzed people use computers ...................................................... 8
    2.4.1 Hardware computer assistive devices ................................................................................ 8
6.1.2 Limitations of the approach ................................................................. 38
6.2 Future work ............................................................................................ 38

Mouse modes of operation: ................................................................. 1
Reset Mode: ......................................................................................... 1
Stream Mode: ...................................................................................... 1
Commands set ....................................................................................... 2
Commands sent to mouse ..................................................................... 2
Messages sent by mouse: ................................................................. 3

<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: ............................................................................... 3

FREESCALE MMA7260Q ACCELEROMETER ........................................... 1
List of Figures

Figure 1.1: Usage of the accelerometer ................................................................. 2
Figure 2.1: A computer Mouse .............................................................................. 5
Figure 2.2: Types of paralysis, grey parts are paralyzed ......................................... 8
Figure 2.3: Two types of switches: activated by head (left) and by chin (right) .......... 9
Figure 2.4: Examples of head stick pointers .......................................................... 10
Figure 2.5: Ear pressure sensor ............................................................................. 10
Figure 2.6: An individual with magnetic tongue ....................................................... 11
Figure 2.7: Microwires emerging from the green and orange tubes ......................... 12
Figure 2.8: Quadriplegic user interacting with the computer using Easy Input system..... 13
Figure 2.9: The Facial Mouse’s working environment ............................................. 14
Figure 2.10: Fluorescent color marker samples ...................................................... 15
Figure 2.11: Web Color ......................................................................................... 16
Figure 3.1: Device to host communication ............................................................. 20
Figure 3.2: Host to device Communication .............................................................. 21
Figure 3.3: Detailed host to device communication ................................................ 21
Figure 3.4: Simplified Transducer Physical model .................................................. 23
Figure 3.5: A photo transistor .............................................................................. 24
Figure 3.6: Photocell effect ................................................................................. 24
Figure 4.1: General design ................................................................................... 26
Figure 4.2: PS/2 socket ....................................................................................... 27
Figure 4.3: Functions to send data to host .............................................................. 30
List of Tables

Table 3.1: PS/2 mouse movement data packet .................................................. 18
Table 3.2: PS/2 Bus states ................................................................................. 19
Table 3.3: PS/2 device to host frame format ......................................................... 20
Table 4.1: Pins assignment of PS/2 socket .......................................................... 27
Table 4.2: ATmega32 PORT C configuration ...................................................... 28
Table 4.4: Initialization process .......................................................................... 32
Table 4.5: Determination of X-movement ............................................................ 33
Table 4.6: Blink detection .................................................................................... 34
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS/2</td>
<td>Personal System 2</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>HCI</td>
<td>Human Computer Interface</td>
</tr>
<tr>
<td>TDS</td>
<td>Tongue Derive System</td>
</tr>
<tr>
<td>MCU</td>
<td>Microcontroller</td>
</tr>
<tr>
<td>ROM</td>
<td>Read Only Memory</td>
</tr>
<tr>
<td>EROM</td>
<td>Erasable Read Only Memory</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Electrically Erasable Programmable Read Only Memory</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computing</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuits</td>
</tr>
<tr>
<td>G</td>
<td>a unit of acceleration equal to Earth's gravity at sea level, it equals 32.2 ft/s(^2) or 9.81 m/s(^2)</td>
</tr>
</tbody>
</table>
CHAPTER 1 INTRODUCTION

1.1 Overview

The computer is considered as one of the most important inventions, not only in the 20th century, but it may be the most important invention of all times, and it gained all this attention due to its unlimited number of its uses in every aspect on daily basis. Computers are found everywhere. In fact modern world will be incomplete without computers and their applications. It’s almost impossible to even imagine the modern facilities without the use of computers. For many individuals computer means PC, on which they can see movies, play games, prepare office sheets and manage daily planners. But this is just a page of the book of computers.

Computer means much more than a PC. A computer can simply be defined as a machine which takes instructions and perform computations and operations accordingly. These commanded instructions are known as programs and computers execute these programs to do an operation. At a time, a set of instructions can be given to a computer to perform several operations, simultaneously. This feature is a point of distinction for the computers.

Another important advantage of the computer is that in spite of its architecture’s complexity, it is so easy to use that even young people can use it efficiently. Many types of computers are available these days, such as Supercomputers that are capable of doing trillions of calculations in fraction of seconds, there are also computers that are used in conducting simple operations like billing, ticket transactions, record maintenance, security analysis etc. And even computers (PC) that used to do home based general activities like office sheet maintenance, day planner, entertainment etc. So computers are involved in every sector of life with different forms and different applications.

1.2 Motivation

It is known that computers have become an important part in everyone’s life because of its unlimited uses in every aspect, so it is not restricted to a certain category, on the contrary, people from every age and every field can use computers, after its usage was restricted to people who were familiar with architecture and the components of the computers, but handicapped people were not able to use computers.

In every society, there is a large category that is marginalized due to their physical disabilities. Although many communities have provided a lot of services for these people, were they provide an assistant to handicapped people to help them doing their normal daily life activities, using computer have always been a problem to them; because the needed someone to use setting next to them and execute what is dictated from the disabled user.
The disability of a person does not necessarily mean his loss of ability to be a productive member in society; so there must some way that enables these people to use the computer so they can use in a productive way such as researches, typing and many other ways.

1.3 Objective

The objective of this project is to design a system that can be used by these handicapped people to interact with the computer directly without the assistance of another person, these handicapped users include amputees (people who lost their hands due to amputation) and quadriplegic people (the people who lost functions of muscles of hands and legs). This system is used as interface between the disabled user and the computer. This system should perform the mouse functions, which are moving the cursor on the screen and clicking to choose the desired object on screen.

1.4 Approach

This project will execute the functions of the mouse using two sensors: the first sensor is an accelerometer, which is a sensor that that measures acceleration, and this sensor will use the movement of the head of the user to move the cursor on screen. The second sensor is a reflective sensor, a sensor that will detect the blinking of the user eye to activate the clicking of the mouse. The accelerometer is placed on the head of the user; the user will move his head to the front and to the back, and from one side to the other as shown in figure (1.1). The reflective is placed in front of the eye to reflect the light emitted from the sensor off the eye. Signals generated from sensors are fed into a Microcontroller. Transmission of data between the these sensors and the computer occurs using the PS/2 (Personal System 2) Protocol.

![Figure 1.1: Usage of the accelerometer](image-url)
1.5 Thesis Layout

1. Chapter 2 (Paralysis and History of Human-Computer Interaction (HCI))
Contains information about paralysis, its types, causes, also some information about the mouse and its types and different operating mechanisms and some of the techniques that were used to help disabled users to interact with the computer.

2. Chapter 3 (Methodology)
Contains the theories implied in designing the system, the principle of operation of the accelerometer, reflective sensor major components used and the methods and procedures carried out to design the system.

3. Chapter 4 (Design)
Includes the detailed information of hardware of the system, and the main functions of software used in programming the MCU to implement the PS/2 Protocol.

4. Chapter 5 (Results & Discussion)
Contains the results obtained from the system and the discusses them.

5. Chapter 6 (Conclusion & Future Work)
Contains conclusion of the project and some suggested ideas for future work on the system.

6. Appendix A (PS/2 Protocol)
Describes the PS/2 Protocol and actions taken to transmit and receive data using this Protocol.

7. Appendix B (Program Code)
Contains the Code of the MCU used to initialize the necessary ports of the MCU and functions used to execute the PS/2 protocol.

8. Appendix C (Accelerometer)
Includes detailed information of the accelerometer, its structure, principle of operation, pins description and pins assignment.

9. Appendix D (Pins Description & Assignment)
Illustrates pins description and pin assignment of each chip used in designing the system.
CHAPTER 2 HCI & USER INTERFACES FOR HANDICAPPED USERS

2.1 Overview

A lot of researches have been conducted in creating new mechanisms used by humans to interact with computers, and these mechanisms include input devices, such as mice, trackballs and touchpads, and menus on the screen that are easy to use and can be understood easily. As computers become more pervasive in culture, designers are increasingly looking for ways to make interfacing with devices easier, safer and more efficient.

This chapter gives information about HCI (Human-Computer Interaction), which is a discipline that is concerned with the study, design, construction and implementation of human-centric interactive computer systems, also definition of Paralysis and its causes, and it also contains some techniques used to help disabled to interact with computers.

2.2 Human-Computer Interaction (HCI)

2.2.1 Definitions

Human–computer interaction (HCI) is the study, planning and design of the interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioral sciences, design and several other fields of study. Interaction between users and computers occurs at the user interface (or simply 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, and other user interactions with large-scale computerized systems such as aircraft and power plants. The Association for Computing Machinery defines human-computer interaction as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." [1].

Because human-computer interaction studies a human and a machine in conjunction, it draws from supporting knowledge on both the machine and the human side. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, linguistics, social sciences, cognitive psychology, and human factors such as computer user satisfaction are relevant. Engineering and design methods are also relevant. Due to the multidisciplinary nature of HCI, people with different backgrounds contribute to its success [1].
2.2.2 Goals of HCI

A basic goal of HCI is to improve the interactions between users and computers by making computers more usable and receptive to the user's needs. Specifically, HCI is concerned with:

1. Methodologies and processes for designing interfaces (i.e., given a task and a class of users, design the best possible interface within given constraints, optimizing for a desired property such as learnability or efficiency of use).
2. Methods for implementing interfaces (e.g., software toolkits and libraries; efficient algorithms)
3. Techniques for evaluating and comparing interfaces
4. Developing new interfaces and interaction techniques
5. Developing descriptive and predictive models and theories of interaction

One of the most popular applications of the HCI discipline is the pointing devices, these devices are almost used in every computer since every computer today supports the Graphical User Interface (GUI), one of its living examples is the mouse, that functions by detecting two-dimensional motion relative to its supporting surface.

2.2.3 Mouse

It is a device that controls the movement of the cursor or pointer on a display screen. A mouse is a small object that can be rolled along a hard, flat surface.

As the mouse is moved, the pointer on the display screen moves in the same direction. Mice contain at least one button and sometimes as many as three, which have different functions depending on what program is running. Some newer mice also include a scroll wheel for scrolling through long documents [2].

Figure 2.1: A computer Mouse
Invented by Douglas Engelbart of Stanford Research Center in 1963, and pioneered by Xerox in the 1970s, the mouse is one of the great breakthroughs in computer ergonomics because it frees the user to a large extent from using the keyboard. In particular, 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. The mouse is also useful for graphics programs that allow drawing pictures by using the mouse like a pen, pencil, or paintbrush [2].

2.2.3.1 Computer Mouse Technologies

Here are the different computer mouse technologies, starting with the old-type mechanical mice:

**Mechanical Mice**

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 seeing on-screen [2].

**Optical Mice**

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:

1. Less wear and lower chance of failure because there are no moving parts.
2. Dirt or dust can't get into the mouse and interfere with the sensors.
3. Smoother response due to increased tracking resolution.
4. Using a mouse pad isn’t required, optical mice can work on normal surfaces such as a desk.

Laser Mice

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. [2]

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].

2.2.3.2 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 paralyzation 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). 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.*

If serious damage occurs in the lower spinal cord, it may lead to paralysis of both legs, this is called Paraplegia. If the damage occurs in the higher spinal cord, closer to the neck, paralysis of all four limbs may occur, and this type is called Quadriplegia. *Error! Reference source not found.*

![Diagram of human spine and paralysis types]

*Figure 2.2: Types of paralysis, grey parts are paralyzed [3]*

This project aims to help people who lost the ability to use their hands, including quadriplegic people and people who their arms were amputated, to interact with the computer.

### 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 [4].

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 [4].

![Two types of switches: activated by head (left) and by chin (right) [4]](image)

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 [4].

2.4.1.2 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:

1. **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 [4].
2. **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].

3. **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 [4].

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.

### 2.4.1.3 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. 

![Figure 2.5: Ear pressure sensor partially inserted into the ear canal to detect signals from tongue movements](image)
1. 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.

2. The details of this work are not yet available for commercial use.

**2.4.1.4 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 [Error! Reference source not found.].

![Figure 2.6: An individual with magnetic tongue piercing wearing the Tongue Drive System headset [6].](image)

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 [Error! Reference source not found.]. This system has two main disadvantages:

1. Requires much training for different commands.
2. Few numbers of commands are available.

**2.4.1.5 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 [4].

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.
2.4.1.6 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 do not penetrate it [7]. 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 [7].

Figure 2.7: Microwires emerging from the green and orange tubes connect to two arrays of 16 microelectrodes [7]

2.4.1.7 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 [8].

Figure 2.8: Quadriplegic user interacting with the computer using Easy Input system [8].

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 [9].

There are two types of computer vision-based systems: one based on face tracking and the other on color tracking [10].

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 [10].

The click can be generated through several mechanisms:

A. Built-in mechanisms:

1. Dwell click: This click is automatically generated after stopping the pointer for a certain amount of time.

2. Sound click: The click is generated when the user emits a sound whose input level is greater than a configured threshold.

B. 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.

![Figure 2.9: The Facial Mouse's working environment.](image_url)
The Webcam captures the user’s movements that a software program translates in order to position the cursor on the screen [4].

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 [11].

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 [12].

### 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 [4].

Color markers are usually fluorescent-colored pieces of paper that can be attached to any surface or to the user’s body [4].

![Figure 2.10: Fluorescent color marker samples](image-url)

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

Where:
(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 [4].

Afterward, the application automatically finds and tracks that specific color 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 [4].

This project is similar to the Easy Input system of Cornell’s in that it detects the movement of the head to direct the mouse and the blinking of an eye as a mechanism for selection, but it differs in that no Keyboard mode is applied; because the function of the keyboard can be performed using the On-screen keyboard program that is supported in many operating systems such as Windows Vista, Windows 7, Linux and Mac OS.
CHAPTER 3 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).
CHAPTER 3

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, [13]. 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, [13]. 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, [13]. 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, [13].

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>
<thead>
<tr>
<th>Byte 1</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></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: PS/2 mouse movement data packet
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 [13].

### 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 [13].

#### 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 [13].

### 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 [14]. 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>
<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 Inhibited</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>

*Table 3.2: PS/2 Bus states*

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) [15]. The clock frequency is 10-16.7 kHz [13].
The lines must be tri-state bi-directional, since at times they are driven by the mouse and at other times by the computer [14]. 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>
<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>

#### Table 3.3: PS/2 device to host frame format

- **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 Data 0-7 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 [13]. Figure 3.1 shows the timing for this frame transmission:

![Figure 3.1: Device to host communication](image)

The PS/2 device is free to send data to the host when the bus is in the idle state. This means that both the clock and data lines are high. The PS/2 device must then wait for half a clock period before it can start its transmission [15]. It should be noted that only a single byte of data (therefore a single frame) can be sent to the host at any one time. To send additional bytes of data (further frames) the PS/2 device must wait until the bus returns to the idle state, before starting each additional transmission [15]. In the case of mouse the movement packet consists of three bytes as indicated earlier, this means the movement data packet requires three frames.
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) [15].

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 [13].

Figure 3.2 shows these steps graphically:

![Figure 3.2: Host to device Communication [13]](image)

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.) [13].

![Figure 3.3: Detailed host to device communication [1]](image)
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 [13].

(b) Is the time it takes for the packet to be sent, which must be no greater than 2 ms [13].

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 [13].

### 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 [16].

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 [17]. An accelerometer senses motion of a surface to which it is attached, producing an electrical output signal precisely analogous to that motion [18].

Accelerometers 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 [19].

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.

#### 3.3.1 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 [20].

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 [20].

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, \( C = \frac{A\varepsilon}{D} \). Where \( A \) is the area of
the beam, \( \varepsilon \) is the dielectric constant, and \( D \) is the distance between the beams [20].

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 [20].

![Figure 3.4: Simplified Transducer Physical model [8].](image)

### 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 [21].
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 [22].

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 [22].
CHAPTER 4 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.

![General design](image)

**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).

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’s 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 (4.1):

![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>

**Table 4.1: Pins assignment of PS/2 socket**

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

Table 4.2: ATmega32 PORT C configuration

• 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:

**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)

**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).

**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.

**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.

**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.

**4.3.1.2 Functions to receive data from host**

These are two functions:

**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).

**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 Initialization process (Hand shaking 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 Initialization 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 Initialization 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 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.
Table 4.3: Initialization process

- send_frame(0xAA)
- send_frame(0x00)

Enable external interrupt

Wait for Host to pull clock line low

Wait for host to release clock line

Disable external interrupt

data = read_frame()

send_frame(0xFA)

Enable external interrupt

send_frame(0x86)

Send frame(0xF2)

Enable external interrupt

Initialization completed
Allowed to enter stream mode

send_frame(0x0A)

data = 0xFF ?

Yes

data IN (0x03, 0xC8, 0x64, 0x50, 0x0A, 0x08, 0x03, 0x06, 0x25) ?

No

send_frame(0xFA)

send_frame(0x00)

Send frame(0xF2)

Enable external interrupt

data = 0xF2 ?

Yes

Initialization completed
Allowed to enter stream mode

No

send_frame(0xFA)

send_frame(0x00)

Send frame(0xF2)

Enable external interrupt

data = 0xFF ?

No
Table 4.4: Determination of X-movement
Table 4.5: Blink detection
CHAPTER 5 RESULTS AND DISCUSSION

5.1 Overview

This chapter includes the results obtained from the project. It discusses these results and shows to which extent they match the expected results.

5.2 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).

Figure 5.1: Simulation environment
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).

![Simulation of data frame](image)

**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).

![Analog signals](image)

**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.

5.3 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.
CHAPTER 6 CONCLUSION & FUTURE WORK

6.1 Conclusion

This project aims to help people with severe disabilities to use the computer in an efficient way, so that they can be able to use the computer freely without the aid of another person. A computer input device is designed so the disabled user can move the cursor on a Graphical User Interface, and they will be able to choose the various icons on that interface.

6.1.1 Suggested solution

The input device will use two sensors to emulate the operation of a real mouse. The first sensor, Accelerometer, detects the motion of the head of the user, and the other sensor, a reflective object sensor, is used to detect the blink of an eye of the user, which are used as a mechanism to emulate the clicks of the mouse. These sensors will generate analog signals that will be converted to digital signals. To establish the communication between the device and the computer, PS/2 protocol is used. The digital signals obtained from the conversion form the required Data Packet imposed by the PS/2 protocol.

Although the required Data frames that are imposed by the protocol were obtained, the designed device did not work; because there were some problems in recognition of the device by the computer.

6.1.2 Limitations of the approach

The major limitation of this project is the reflective object sensor that is used to detect the blinks uses infrared light that will be reflected from the eye of the user. Infrared light is harmful to the eye although its intensity is very high, and using it may be risky.

6.2 Future work

Many suggested enhancements can be made to the suggested solution to maximize its efficiency and to overcome some of the limitations faced in the designing of the computer input device.

Some of these enhancements are:

1. Using the USB to communicate directly with the computer instead of using a PS/2-to-USB converter. Although programming USB protocol is more complex, using a converter will make designing the device more expensive.
2. Making the device wireless instead of making wired. Wired device will limit the movement of the users. Wireless device will use two units, a head unit and ground unit. Head unit will be composed of a Microcontroller that does the conversion process, and the converted data will be sent using a transmitter to the ground unit, where a receiver on it and this unit will emulate the PS/2 protocol and performs the communication with the Computer.

This device can also be used in many other applications not only as a computer input device:

- This device can be in games as input device instead of gamepads and joysticks. Thus, disabled users in early ages will be able to use game consoles such as Play station and X box.
- This device can also be used to control the movement of a motorized wheel chair, where the direction of the tilt of the head will be used to direct the wheel chair.
REFERENCES


Author: Adam Chapweske


http://www.dimensionengineering.com/accelerometers.htm

[18] Dytran Instruments, INC, “Accelerometer Mounting Considerations”

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

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

http://www.thefreedictionary.com/phototransistor

[22] Book rags, microcontroller summary.
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>

Table F: Commands sent from computer
APPENDICES

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>Mouse passed self-test</td>
<td>AA</td>
</tr>
</tbody>
</table>

Table G: message sent from mouse.

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
OPB608A 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>
APPENDICES

APPENDIX C

FREESCALE MMA7260Q ACCELEROMETER

±1.5G - 6G THREE AXIS LOW-G MICRO-MACHINED ACCELEROMETER

C.1- Introduction:

The MMA7260Q low cost capacitive micro-machined accelerometer features signal conditioning, a 1-pole low pass filter, temperature compensation and g-Select which allows for the selection among 4 sensitivities. Zero-g offset full scale span and filter cut-off are factory set and require no external devices. Includes a Sleep Mode that makes it ideal for handheld battery powered electronics. Before getting into the features of the accelerometer, here are some of the terminologies that are used in describing the functions of the accelerometer:

C.1.1) Acceleration:

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 or other reference frame. The units are length/time^2. Gravity .g. is an acceleration. \[^{[3]}\]

C.1.2) g-level:

This refers to the acceleration value. +1 g is the acceleration measurement for gravity which is equal to 9.81m/s^2. \[^{[3]}\]

C.1.3) g-Select:

A feature on the accelerometer device that allows for the selection between more than one sensitivity. Depending on the logic of this input the internal gain is changed allowing the accelerometer to function with a higher or lower acceleration range. \[^{[3]}\]

C.1.4) Offset:

Offset refers to the DC output level of the accelerometer when no motion or gravity is acting on it, often called the 0g-offset. \[^{[3]}\]

C.1.5) Sensitivity:

The output voltage change per unit of input acceleration at nominal $V_{DD}$ and temperature, measured in mV/g (Voltage Output per g).\[^{[3]}\]

Figure (C.1) shows the Pin connections of the accelerometer, and Table (C.1) shows their description:
C.2 - Principle of operation:

The Freescale accelerometer is a surface-micromachined integrated-circuit accelerometer.

The device consists of two surface micromachined capacitive sensing cells (g-cell) and a signal conditioning ASIC contained in a single integrated circuit package. The sensing elements are sealed hermetically at the wafer level using a bulk micromachined capwafer.

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 an acceleration (Figure C.2).

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 (Figure C.2). 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.

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. \(^3\)

**Figure (C.2):** Simplified Transducer Physical model.

The accelerometer outputs a DC voltage from its output pins even when there is no motion or acceleration is acting on the accelerometer, which is called the static acceleration, and the value of this voltage depends on the position of the accelerometer. Obviously, the accelerometer output voltages vary according to the motion or acceleration acting on the accelerometer, which is called the Dynamic Acceleration. Figures (C.3) and figure (C.4) show Static acceleration and Dynamic acceleration respectively:
APPENDICES

C.3.1) g-Select:

The g-Select feature allows for the selection among 4 sensitivities present in the device. Depending on the logic input placed on pins 1 and 2, the device internal gain will be changed allowing it to function with a 1.5g, 2g, 4g, or 6g sensitivity (Table C.2). This feature is ideal when a product has applications requiring different sensitivities for optimum performance. The
sensitivity can be changed at anytime during the operation of the product. The g-Select1 and g-Select2 pins can be left unconnected for applications requiring only a 1.5g sensitivity as the device has an internal Pulldown to keep it at that sensitivity (800mV/g). \[3\]

<table>
<thead>
<tr>
<th>g-select 2</th>
<th>g-select 1</th>
<th>g-Range</th>
<th>Sensitivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1.5g</td>
<td>800mV/g</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2g</td>
<td>600mV/g</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4g</td>
<td>300mV/g</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>6g</td>
<td>200mV/g</td>
</tr>
</tbody>
</table>

**Table (C.2): g-select pin description**

**C.3.2- Sleep Mode:**

The 3 axis accelerometer provides a Sleep Mode that is ideal for battery operated products. When Sleep Mode is active, the device outputs are turned off, providing significant reduction of operating current. A low input signal on pin 12 (Sleep Mode) will place the device in this mode and reduce the current to 3uA typ. For lower power consumption, it is recommended to set g-Select1 and g-Select2 to 1.5g mode. By placing a high input signal on pin 12, the device will resume to normal mode of operation. \[3\]

**C.3.3- Ratiometricity:**

Ratiometricity simply means the output offset voltage and sensitivity will scale linearly with applied supply voltage. That is, as supply voltage is increased, the sensitivity and offset increase linearly; as supply voltage decreases, offset and sensitivity decrease linearly. This is a key feature when interfacing to a microcontroller or an A/D converter because it provides system level cancellation of supply induced errors in the analog to digital conversion process. \[3\]
APPENDIX D

PROGRAM LISTING

/*****************************************************/
Project : Input device for disabled users
Date : 6/29/2011
Author : MossabRizig& Nadir El Berair
Company : UOFK

Chip type : ATmega32
Program type : Application
Clock frequency : 1.000000 MHz
Memory model : Small
External SRAM size : 0
Data Stack size : 512

/*****************************************************/
# includes
#include <mega32.h>
#include <delay.h>
#include <stdio.h>

/*****************************************************/
# Defines
#define ADC_VREF_TYPE 0x20 // set by the wizard for ADC
#define din PINC.0             // din is the pin where host sends its commands
#define dout PORTC.1           // dout is the pin which the device sends data through
#define cin PINC.2             // clock controlled by host
#define cout PORTC.3           // clock generated by device
#define LED1 PORTC.5           // LEDs used for debugging
#define LED2 PORTC.7           // LED1 activated when sending data to host , LED2 activated when receiving data from host
#define high 1
#define low 0
#define threshold 5            //just estimated, needs to be checked when the device works
#define MY_CLOCK_PERIOD  1.0     /*Micro Secs */

/*****************************************************/
Variables declaration
unsigned int period, current_time;    //to detect the inhibit and request to send states
int x, y;                             //data read from accelerometer
unsigned char x_ref=0x00, y_ref=0x00, blink_ref=0x00, status=0b00001000;    //initial values for the references
unsigned char localMCUCR=0x02;        //used to control the interrupt sense of triggering
unsigned char data;                  //used to read data from host

/*****************************************************/
Local Functions
*****************************************************/
unsigned char read_frame() ;
void send_frame(unsigned char command) ;

This interrupt will just be used for initialization purpose
******************************************************************************

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    int i;
    // Place your code here
    PORTB=~PORTB ;
    if (localMCUCR==0x02)  
        //the interrupt occurred due to falling edge
        TCNT0=0;  //reset the counter, start counting from the falling edge
        localMCUCR^=0x01 ;
        MCUCR = localMCUCR ;  // 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)
                {  //reset command
                    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(0x00);  //Self test passed
                    send_frame(0x00);  //Mouse ID
                    GICR=0x40;  // Enable Interrupt 0
                } else if((data==0xF3)||(data==0xC8) ||(data==0x64)||(data==0x50) ||(data==0x0A)||(data==0xE8) ||(data==0x03)||(data==0xE6) ||(data==0x28))
                    {
                    send_frame(0xFA);  // Acknowledge
                    GICR=0x40;  // Enable Interrupt 0
                    }  //0xF3: set sample rate
                    //0xC8: decimal 200,0x64: decimal 100,0x50: decimal 50,0x28: decimal
                    //0xE8: set resolution
                    //0x03:8 counts/mm
                    //0xE6: set scaling 1:1
            else if (data==0xF2)
                {  // get device ID
                    send_frame(0xFA);  // Acknowledge
                    send_frame(0x00);  //Mouse ID
                    x_ref=0b00000000;  // reset the mouse
                    y_ref=0b00000000;  // movement counters
GICR=0x40;                // Enable Interrupt 0
} else if (data==0xF4)
{
    //enable command
    send_frame(0xFA);           //Acknowledge
    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
}
localMCUCR^=0x01 ;
MCUCR = localMCUCR ;               // change the interrupt detection mode( Rising=0x03,
Falling=0x02)
}

**********************************************************************************************
Functions to send data to host
***********************************************************************************************/
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);
delay_us(25);
    //frequency used is 10KHz , so the clock should remain high for 50us & low for the same period
cout=low;
}

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)
i=0;
       // i used for the loop . p for parity calculation
unsigned char byte,sb;                            // both variables are temporary variables used for shifting and
for (i=0;i<8;i++)
{/n    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);
    dout=high;       // stop bit = 1
    delay_us(25);
    cout=low;
    delay_us(50);
}

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
        // device should ensure that bus is idle for 50us before it starts transmission
        delay_us(50);   // device should ensure that bus is idle for 50us before it starts transmission
        if((cout==high)&&(dout==high))
        {
APPENDICES

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)
{
#ifdef
definition

    // 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
#endif
definition

    // 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=0x033;
    DDRB=0x33;
    // Port C initialization
APPENDICES

// Func7=Out Func6=In Func5=Out Func4=In Func3=Out Func2=In Func1=Out Func0=In
// State7=T State6=T State5=1 State4=T State3=1 State2=T State1=1 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
TCCR0=0x01;
TCNT0=0x00;
OCR0=0x80;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// 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;
ICR1H=0x00;
ICR1L=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 ***
**********************************************************/
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 ***
**********************************************************/

/**********************************************************
*** 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)
        {
            // FIND
x_ref=x_ref+x;                               //                             AN
y_ref=y_ref+y;                               //                           AVERAGE
blink_ref=blink_ref+blink;
count++;                                     //                            VALUE
}                                                //                             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_ref-x > threshold*3)
        x_mov = 9;                               //fast movement
      else if(x_ref-x > threshold*2)
        x_mov = 6;                              //medium movement
      else if (x_ref-x>threshold)
        x_mov=3;                                //slow movement
      else
        x_mov = 0;                            //no movement
      x_ref=x;
    } 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 if (x-x_ref>threshold)
        x_mov=3;                                //slow movement
      else
        x_mov = 0;                            //no movement
      x_ref=x;
    } else
    {
      status = status || 0b00001000 ;              //xsign=0
      x_mov=0x00;
    }

    if(y<y_ref)
    {
      status = status || 0b00011000 ;              //ysign=1
      if(y_ref-y > threshold*3)
        y_mov = 9;                               //fast movement
      else if(y_ref-y > threshold*2)
        y_mov = 6;                              //medium movement
      else if (y_ref-y>threshold)
        y_mov=3;                                //slow movement
      else
        y_mov = 0;                            //no movement
    }

if(y>y_ref)
{
    status = status || 0b00011000 ;              //ysign=1
    if(y_ref-y > threshold*3)
      y_mov = 9;                               //fast movement
    else if(y_ref-y > threshold*2)
      y_mov = 6;                              //medium movement
    else if (y_ref-y>threshold)
      y_mov=3;                                //slow movement
    else
      y_mov = 0;                            //no movement
}
APPENDICES

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 Status Byte
        send_frame(status);
        send_frame(x_mov);  // send X-axis Byte
        send_frame(y_mov);  // send Y-axis Byte

    y_ref = y;
}

delay_ms(10);