Haptically Enhanced Mouse

Heather Jones and Lauren Stadler
Advisor: Erik Cheever

30 November 2005

Abstract

We intend to design and build a haptically enhanced mouse for the visually impaired. A basic trackball mouse will be improved upon to include a brake to indicate when the pointer reaches the edge of the screen, a vibrating motor that alerts the user when the pointer passes over the edge of a window, and an audio feedback option that provides information about the window or icon the pointer is over. The mouse will connect through the USB port and operate in Linux. The mouse will initially be designed to interact with a GUI that we will create, but ultimately it will be able to interact with the window managing system. This project will involve building and wiring the mouse itself, programming control for the brake and motor, and writing a driver that will allow the mouse to interface with the computer.

Introduction

We propose to build a trackball mouse that will improve the experience of visually impaired users, or those with poor coordination, interacting with a desktop computer's visual environment. While several devices for blind computer users currently exist, including a mouse with similar functions to our design, most are quite costly. Our proposed device, however, will be made of very simple, off-the-shelf components, and therefore the mouse will be much less expensive to manufacture. The technical discussion below describes how we will implement the hardware on the mouse, the software on the host computer, and the communication between the mouse and the host computer; it also explains the reasoning behind the major design choices we have made thus far. The project plan provides a brief statement of the steps involved and a critical path method schedule detailing when these steps will be accomplished. The qualifications section outlines our experience in technical tasks related to this project and discusses our access to the necessary resources. The costs section provides a preliminary estimation of cost for different elements of this project.

Technical Discussion

There are many interfaces available for computers to communicate with peripheral devices. In most systems on the market today, however, the Universal Serial

Bus (USB) has become the dominant interface, with lower-cost systems phasing out the older “legacy” ports completely. Since we want our mouse to be a prototype of a device that would be useful to the mainstream blind user without requiring the purchase of additional expensive hardware, we chose to use USB. Additionally, the USB specification is controlled by the USB Implementer’s Forum, which includes over 900 companies in the technology industry. This means USB is not a proprietary technology: although a $1500 fee is required to obtain a vendor ID before selling any product including a USB interface, there is no licensing fee required to develop USB software.

The Universal Serial Bus was designed to connect peripheral devices to a computer; each instance of the bus includes a single host that controls communication with up to 127 devices. A USB cable transmits data serially using a differential signal and has physically different connectors on its upstream and downstream ends. The USB specification standardizes the connectors, cables and transfer types that comprise the USB interface. Two versions of the specification are currently in use. USB 1.1 supports a low-speed data transfer rate of 1.5 Megabits per second and a full-speed data transfer rate of 12 Megabits per second. The newest version of the specification, USB 2.0, also supports a high speed data transfer rate of 480 Megabits per second. For this project, because we will be taking input from and providing feedback to a human user, who necessarily operates at a much lower speed than the computer, low-speed data transfer is sufficient.

To create a USB device, we need

- A controller chip with a USB interface and the ability to control all other device functions
- Code on this chip to handle the USB interface and other device functions
- A device driver on the host computer
- A program on the host to test the device
- A program that allows the device to perform its desired functions, which in our case will be a modification of an existing window manager.

Considerations in choosing a controller chip include cost, availability, speed, reprogrammability, and difficulty of development. We chose to look at PIC microcontrollers because of their low cost and high availability. Since we only need low-speed data transfer, we first examined the PIC16C745/765, but these chips only come in one-time-programmable and UV-erasable PROM versions, neither of which would facilitate easy reprogramming. Since the 18F2455/2550/4455/4550 series chips that support full-speed data transfer also support low-speed data transfer, and they have Flash memory, we decided to work with these chips instead. There is a development board available for the 18F4550, and the Engineering Department should already have the software necessary to program the chip using this board. Programming is done in the C language, which we will also be using in other parts of this project; this is better than having to learn an additional programming language to work with the microcontroller.

In order to make our mouse usable for the greatest number of visually impaired users, we would ideally like it to work with the Windows™ operating system. Neither of us, however, have done any software development for Windows, so we chose to write a driver for the Linux operating system. Unlike Windows, Linux is open source, so there is a free USB driver skeleton we can use, as well as plenty of similar example code to examine. Should our device actually go into production, it would not be difficult for someone more familiar with Windows driver development to write a new driver for our device.

Before we attempt to have our device interact with windows and desktop icons, we need to make sure that it is working correctly and test its effectiveness, so we plan to write a simple test program. This will need to include a graphical user interface (GUI). Because JAVA has a good library for writing GUIs, and we want to spend as little time as possible on the GUI portion of the code, we have decided to implement the test program in JAVA.

Ultimately, we want the mouse to be a tool that can identify visual elements outside of our GUI program, such as icons on the desktop or the edge of a window. In order to do this, we must interact with the window manager. There are many window managers available for Linux – we want one that is simple and well documented. Twm has been suggested to us as a very simple window manager, so this will be the first that we examine.

We want our mouse to provide several functions:

- Brake the trackball when the pointer reaches the edge of the screen

- Vibrate when the pointer crosses over a feature of interest
- Provide a spoken description of a visual feature upon request

The first two of these will involve mechanical parts on the mouse itself. The third can be done on the host computer. To brake the trackball, we will use a small motor or solenoid to clamp the ball between two rubber pads. We examined the possibility of placing brakes on the wheels used to translate the motion of the trackball into electrical pulses, but we discovered that stopping these wheels did not necessarily stop the trackball. To vibrate the mouse, we will use a small asymmetrically weighted motor such as those found in cellular phones. Incorporating these additional components may significantly increase the size of the mouse. They may also require more power than can be drawn from the USB interface, in which case we will include a battery in our mouse. We have decided to build a trackball mouse instead of a traditional mouse. Because this design requires only the trackball to move, and not the entire device, the additional size and weight will not be a disadvantage. The user's direct contact with the trackball in this configuration also makes it much easier to detect when the device is preventing the trackball from moving.

Because computers generally have speakers already, we do not need to include speakers in our mouse in order to implement the speech-on-request function. Instead, this can be done by calling a text-to-speech function when a certain mouse button is pressed. There are a number of text-to-speech systems that exist for Linux. We will use the one we find most appropriate for this task.

Project Plan

Our project can be broken down into three main phases: research and design, building and programming, and debugging and testing. Research concerning the materials necessary for building the mouse has already begun and they will be purchased before the end of the fall semester. A preliminary list of devices has been compiled, including solenoids for the brake, a vibrating motor for alters, batteries, a USB microcontroller, a development board, and trackball mice, and a few parts have already been ordered. The first task of the project will involve constructing a brake for the trackball. In addition, the first weeks of the semester will be spent coming to understand how the microcontroller works, researching existing codes and device drivers for USB, and brushing up on our programming skills. Once the trackball brake is built and tested, the design and layout of other components, including a brake release button, vibrating motor, and the microcontroller chip will begin, and assembly of the mouse will follow immediately. The next major task will be to program the microcontroller and write additional device drivers for communication between the controller and computer. As we progress, we will test the mouse using simple GUIs created in JAVA. If we are able to successfully execute all of the desired functions of the mouse in the GUI, the next step is for the mouse to interact with the window manager.

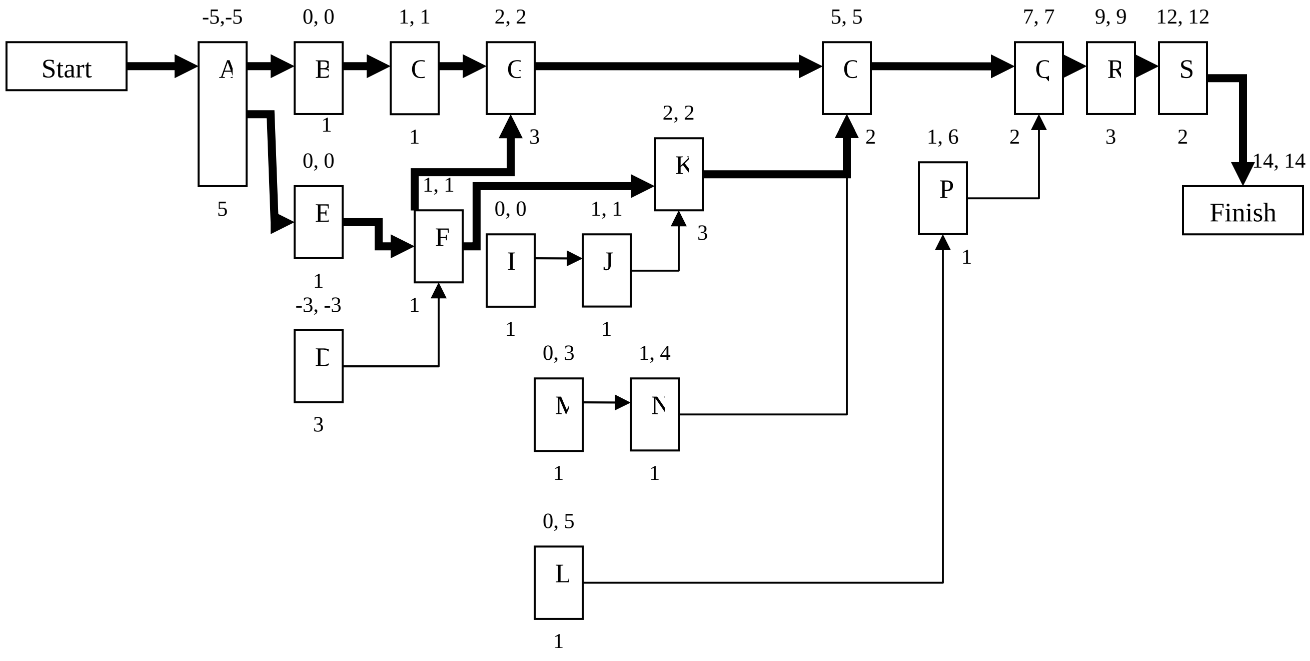
Below is a Critical Path Method (CPM) Network diagram and flow chart that show the significant activities of the project and their expected duration and effort. Appendix A is a Gantt milestone chart that indicates where the important benchmarks of the project fall on a schedule of the 14 weeks of the spring semester and displays the week by which each activity should be completed.

CPM Network Diagram

Activity	Needs	Feeds	Duration	Effort	Action
A	-	B	5 w	9 h	Research and purchase materials
B	A	C	1 w	10 h	Design track ball brake and test
C	B	G	1 w	8 h	Re-order devices if necessary and test
D	-	F	3 w	20 h	Learn C
E	A	F	1 w	8 h	Familiarize ourselves with the microcontroller 1) Read Development Kit exercise book and complete simple exercises 2) Practice using the PIC compiler and programming the microcontroller (detect push button, make LEDs blink)
F	D, E	G, K	1 w	5 h	Write simple test programs for communication between computer and controller 1) Activate LED on controller board from program on computer 2) Have computer beep/respond with input from controller board
G	C	O	3 w	30 h	Design and layout circuit and build mouse
I	-	J	1 w	5 h	Read and understand existing device drivers
J	I	K	1 w	5 h	Plan out what additions to the driver are necessary/what we need our code to do
K	F, J	O	3 w	25 h	Write and debug code for microcontroller – action complete when: 1) Brake mechanism activates with signal from computer 2) Vibrating motor activates with signal from computer 3) Brake release button releases braking mechanism 4) All mouse movements and button presses are sent to the computer
L	-	P	1 w	5 h	Research window manager
M	-	N	1 w	6 h	Familiarize ourselves with JAVA and creating GUIs – read tutorial and

					complete sample exercises
N	M	O	1 w	3 h	Design GUI test program
O	G, K, N	Q	2 w	30 h	Implement and test GUI (incrementally) – action complete when: 1) Mouse can vibrate and brake when positioned over visual objects within program 2) Audio feedback can identify object at mouse position 3) Blindfolded test subject can perform a designated sequence of selections of objects on screen after exploring the environment
P	L	Q	1 w	5 h	Plan interactions with window manager
Q	O	R	2 w	35 h	Write code to interact with window manager 1) Mouse vibrates when passing over an icon or edge of a window 2) Mouse brakes when it reaches the edge of the screen 3) Audio feedback identifies active window and icons on desktop
R	Q	S	3 w	40 h	Test and re-evaluate – action complete when blindfolded test subject can sketch a picture of the desktop environment after exploring with mouse
S	R	-	2 w	55 h	Final report and presentation

CPM Flow Chart



Qualifications

For our project in Digital Systems, we created a system that collected data from the physical world, transferred this data to a computer, made decisions based on this data, and sent control signals back to a physical device. Although that project was much smaller and simpler than the one that we propose, it included many of the same elements. In addition, one of us has experience with PIC microcontroller programming, coursework in JAVA, and familiarity with programming the Linux operating system. There are a number of computers in Hicks which are dual-boot Windows and Linux; these will allow us to develop our embedded code for the microcontroller in Windows and our host-based software in Linux. The major hardware aspects of this project will consist of soldering together the electrical circuits and physically mounting the components in a base structure. The former can be done in the electronics lab in Hicks, and the latter can be accomplished with tools that we own or, if necessary, tools borrowed from the engineering shop. Thus, we believe that we have the ability, experience, and resources necessary to for this project.

Costs and Equipment

We have already purchased a few PIC18F2550 microcontrollers for approximately $6 each. We are in the process of purchasing a PIC development board, which should cost $60-$120. We may be able to obtain several used trackball mice for free to use for research and mechanical component testing. If not, we should be able to purchase such mice for $15-25 each. The solenoids and motors that we will test for use in our device should cost approximately $4-12 each. Many of the electrical components we will use should be stocked in the electronics lab. Those that are not should not be unduly expensive. The body of the mouse itself can be built fairly cheaply from wood or metal machined in the engineering shop. We will need a computer on which we are allowed administrative access in order to load our driver.

References

Axelson, Jan. 2001. *USB Complete*. 2nd Ed. Lakeview Research. Madison, WI.

Custom Computer Services, Inc. http://www.ccsinfo.com. 11 Nov. 2005.

Jameco Electronics. http://www.jameco.com. 11 Nov. 2005.

Microchip Technology Inc. http://www.microchip.com. 11 Nov. 2005.

Mouser Electronics. http://www.mouser.com. 11 Nov. 2005.

Rinne, Karl. Lecture Slides. *ET4508: Computer Systems Architecture*. University of Limerick. Feb.-May 2005.

Appendix A: Gantt Milestone Chart

		0	1	2	3	4	5	6	Break	7	8	9	10	11	12	13	14
Activity	**(Pre-spring)**	**16-Jan**	**23-Jan**	**30-Jan**	**6-Feb**	**13-Feb**	**20-Feb**	**27-Feb**	**6-Mar**	**13-Mar**	**20-Mar**	**27-Mar**	**3-Apr**	**10-Apr**	**17-Apr**	**24-Apr**	**1-May**
A	■								▨								
B		■							▨								
C			■						▨								
D	■								▨								
E		■							▨								
F			■						▨								
G				■	■	■			▨								
I		■							▨								
J			■						▨								
K				■	■	■			▨								
L	■	■							▨								
M	■								▨								
N						■			▨								
O							■	■	▨								
P								■	▨	■							
Q									▨	■	■						
R									▨			■	■	■			
S									▨						■	■	