Custom Keyboard for Disabled Veteran

Jordan Birrenkott
jordan.t.birrenkott@ndsu.edu

Jacob Leerssen
jacob.leerssen@ndsu.edu

William Fowley
william.fowley@ndsu.edu

North Dakota State University
Electrical and Computer Engineering Department
Fargo, ND 58108-6050

December 2015

Keywords: Keyboard, Disabled Veteran, Wounded Veteran, Shortcut

Abstract

The goal of this project is to make using a computer keyboard easier for a wounded veteran or anyone without the use of their their hands. To make this possible the computer keys were enlarged to make them easier to press, and are assigned shortcuts to allow for common computer use at one press of a button.

This material is based in part upon work supported by the National Science Foundation (NSF) under Grant Number CBET-1401507. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

1. Introduction

The goal of this project is to make a custom keyboard that will allow common tasks on a computer to be done with a single button press by accomplishing the following: custom print larger keys to limit the precision required to operate, program the keys to do common tasks that would require multiple keystrokes or key and mouse click combinations.

To make the larger keys we decided to use 3-D printing technology. This allowed us to have complete control over the design of the key and also quickly produce prototypes for testing. In order for us to be able to implement the multiple keystrokes on a standard keyboard using only a single button push on ours, we will need to use a microprocessor. We decided to use an Arduino to allow us to recognize a button press as well as send the desired instructions to the computer. This will be discussed in greater detail in the programing section.

2. Previous Work

Standard Keyboard

A previous work that we will be drawing from for this project is a standard QWERTY keyboard. The primary function of a standard keyboard is to be an input device to a computer. A large part of the circuitry that makes up a keyboard is called the key matrix. The key matrix is a grid of circuits under the keys. When a key is pressed it pushes down on a switch that completes a circuit and allows a tiny amount of current to run through.

After the processor recognizes that a circuit is closed, it compares the location of the circuit to a character map stored in some ROM(read-only memory). The key press that is associated with the character mapping is then passed on to the computer. The corresponding action is then carried out by the computer. This signal is passed through a USB in modern computers, but older keyboards sent information through a PS/2 cable. Some of the newest keyboards communicate with computers wirelessly through infrared, radio frequency, or bluetooth technology.

Regardless of how the signal is passed, the computer has a keyboard controller that sits and waits for the input. This is an integrated circuit that processes all the data from the keyboard and then forwards it on to the operating system. The operating system first checks to see if a system command was sent, such as ctrl-alt-delete, and does what the command requires. If the command is not system level the computer checks for a general command, such as alt-f, and then does the required task that the program specifies. If the input is neither of the above the data is accepted as content if the program accepts data, otherwise the data is just ignored.

X-keys XK-24 Desktop keyboard

This is a shortcut keyboard developed by P.I. Engineering.The model that we specifically looked at was the one that had 24 user-programmable keys and blue or red LEDs for backlighting. What this keyboard does is allow the user to have a separate keyboard strictly dedicated to shortcuts, applications, etc. If the user reprograms the key they need to pull the key out and replace it with a different one if they want it to represent the new command. The keys on this shortcut keyboard are the same size as those on a standard keyboard.

3. Requirements

- Keyboard must communicate with the computer by USB cable
  - The vast majority of computers have USB ports, This will allow our keyboard to be used with most computers
- Powered by USB
  - Having the keyboard powered by a USB means there does not need to be a separate cord for power and there does not need to be an open power outlet near by.
- Large easy to press buttons
  - The buttons need to be large and easy to press so that using a sip and puff device with the keyboard is simple.
- Reprogrammable
  - The keyboard needs to be easily reprogrammable
- 3D printed enclosure
  - On the off chance that the enclosure is damaged, this will allow for a new one to be created very simply.

4. Design Options

Keyboard Layout

Option 1: Single Row of Keys

Advantages:

- No issue with having the user press a key in the row in front of or behind the desired key when pressing down on desired key

Disadvantages:

- The keyboard would end up being long and the user may have to rotate themselves to reach both ends of the keyboard

Option 2: Matrix of Keys

Advantages:

- The user will be able to reach all keys without having to rotate themselves
- Will be roughly the same size as a standard keyboard, so it can fit anywhere a standard keyboard can

Disadvantages:

- There will be the potential that a user could “misclick” a key when trying to reach the one before or behind it

Decision:

In the end we chose to do a matrix of keys. The single row of keys would take up a large amount of space on the user's desk. The matrix of keys would be more size efficient per button.

Key Switch Used

Option 1: Dome Switch

Advantages:

- They are inexpensive
- They aren't noisy when they are pressed
- They are easy to press

Disadvantages:

- We would have to have a fixed design where all of the keys have to be close to each other which may not be optimal to use with the "sip and puff"

Option 2: Push Button Switch

Advantages:

- They are inexpensive
- They would make the layout of the keyboard optimal since they can be placed anywhere
- They would be easier to program and reprogram

Disadvantages:

- They may be harder to press

Decision:

We decided to go with the push button switch. The ability to place the keys in any configuration we want is essential to this project.

Integrated Circuit Used

Option 1: Xilinx Spartan-6 LX9 Microboard

Advantages:

- USB powered
- Small
- Enough I/O

Disadvantages:

- Unfamiliar programming language
- Very little source material or examples

Option 2: Arduino Due

Advantages:

- USB powered
- Uses familiar programming language
- Small
- Numerous examples and source material

Disadvantages:

- Not as challenging from a learning standpoint

Decision:

We decided to go with the Arduino Due. The Spartan-6 was very complicated and was more advanced than what we needed. The Arduino Due was the perfect fit for our project.

5. Software

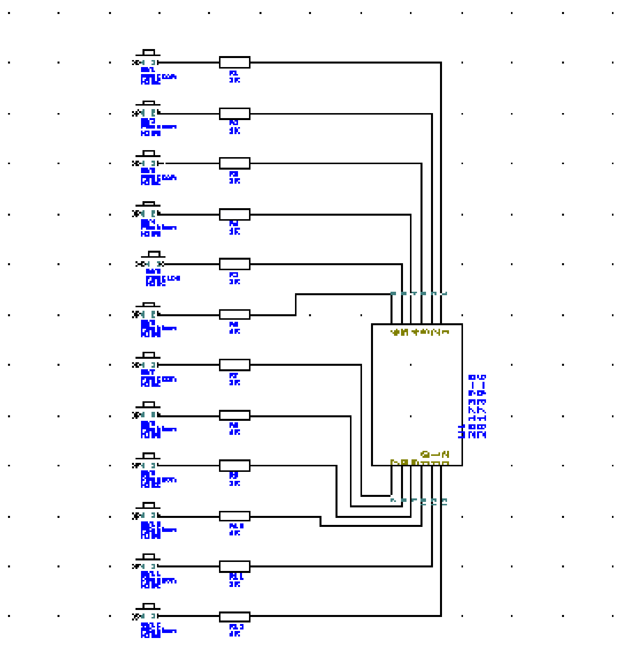
To program the Arduino Due we used the language C++. By using an Arduino Due and C++, we were able to implement some existing open source libraries in both of the main functions of our software. The two main functions of the software is to recognize a button is pressed and then simulate the desired keystroke and/or mouse movement of a standard keyboard and mouse setup.

The first thing the program does is assign pins one through twelve on the Arduino Due to be able to implement a class that will prevent registering false clicks. To do this the Bounce library was used. The Bounce library takes two parameters in order to be used. The first parameter declares what pin we want to associate with the Bounce class. The second parameter is the amount of time in ms we want to delay after receiving a button press. This is used to prevent chatter on our keys. Without this chatter prevention our keyboard would have the potential to register a single press as multiple presses causing the function to perform multiple times instead of the desired one. To be used as input we also need to declare what type of input each pin will be. For this project we decided to use pullup resistors. In order to implement this in the software we used the pinMode method built into Arduino software. In this method the first parameter declares what pin we are going to be using and the second declares that we are using INPUT_PULLUP as our pinmode type.

The second main function of the program is to send the desired instructions to the computer. This will be implemented in two ways: using keystroke combinations and batch files. The keystroke combination functions will utilize the press method in the Keyboard library. The Keyboard library is an open source library that was imported into our program. The press method will simulate a press and hold of a key whose name is passed as a parameter. After the desired keys have been pressed the releaseAll method will simulate the release of the keys. This is how the key combinations such as alt + tab were implemented. In order to implement functions such as “open the internet” a batch file was required. The batch files are all stored in a folder that will be located on the C drive. Since there are multiple batch files we created a few methods to be reused each time a batch file will be used. The first method opens the command prompt. This is done by sending a key press combination of the windows and ‘r’ key. This opens up the run menu. In the run menu the println method is utilized to send ‘cmd’ and press enter. This opens the command prompt. The second method created for reuse in the batch file commands is the changeDirectory method. This method works by sending the file path of the batch files to the command prompt line that was previously opened. After these two methods have executed the name of the batch file will be sent to the command prompt which will then execute the batch file. After this has been completed a final custom created method named exitCommandPrompt will be executed. This will send the string “exit” to the command prompt followed by the enter key which will close the command prompt.

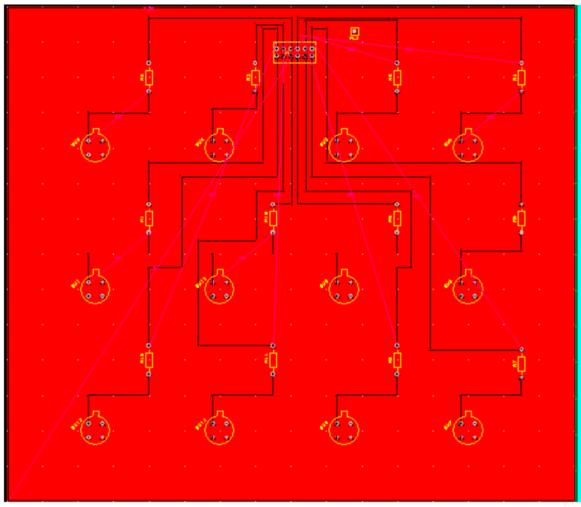
6. Hardware

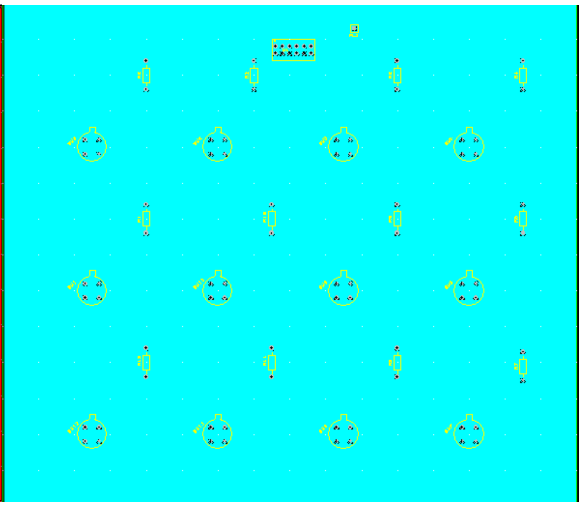
The PCB for this project was very simple. There are twelve buttons that connect to twelve pullup resistors that lead to a 2x6 socket. The value of the resistors don’t matter, but the ones we used were 1.5 kΩ. To connect the PCB to the Arduino, we ran wires from the socket to the I/O pins on the Arduino.



Schematic of the PCB

On the side of the button that wasn’t connected to the resistor, it was connected to a single socket connector. This side of the button was connected to the socket from the entire back plane of the PCB. A wire is run from the single socket connector to the ground pin on the Arduino.

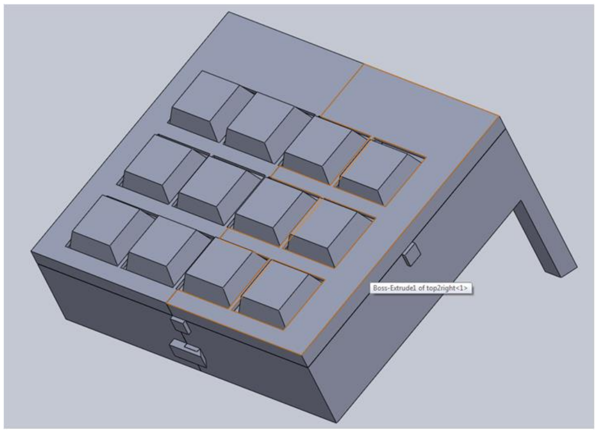

Top Layer of PCB


Bottom Layer of PCB

The PCB is the physical part of our keyboard. To get the spacing right for our keys, the PCB had to be large. The dimensions of the board are eight inches wide and seven inches tall.

7. Enclosure

The enclosure was made with a 3D printer. We were slightly limited by the 3D printers we had access to, as it was not quite large enough to print the individual pieces by themselves, So the pieces had to be printed in 2 parts, then glued together. The material it prints is a hard, durable, plastic. The enclosure did not need to do anything special, it just needed to hold the Arduino and the PCB. The keyboard is angled at 30 degrees. Here is a rendered picture of all the parts put together.



8. Item List

- 3D printed enclosure
- PCB board
- 12 buttons
- 12 resistors
- Arduino Due

- USB cord
- Wires

9. How to Use the System

To use this device please paste the folder titled "CustomKeyboard" to your C: drive. This folder contains all the batch files used in the operation of the keyboard. Then you simply need to plug the USB cable into your computer and it will be fully functional. To perform the desired task simply press a key on the keyboard.

Disclosure:

This material is based in part upon work supported by the National Science Foundation (NSF) under Grant Number CBET-1401507. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

Approved By

Advisor Name _______Samee_Khan, Ph.D.__________

Advisor Signature ______________________________ Date _______________