BREAKTHROUGH INNOVATIONS GROUP
Go BIG or Go Home

10/03/2016

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby British Columbia
V5A 1S6

Re: ENSC 305W/440W Design Specifications for the Omega Key

Dear Dr. Rawicz,

The following document will outline the design specifications for our dynamic display keyboard named Omega Key for capstone. The goal of our group is to design a keyboard with programmable key displays, allowing the user to fully customize the language and position of letters on a keyboard.

The design specification offers specific design and component choices including mechanical hardware and software designs to fulfill the goals outlined in the functional spec document. It will also include a plan to test the Omega Key to ensure everything is up to specifications.

Breakthrough Innovations Group (B.I.G) has six very talented engineering students: David Pallmann, Chase Kwak, Steven Liu, Steven Timotius, Steven Luu, and Frank Tran. If you have any questions or concerns about our functional specifications, please feel free to contact me by email or phone at dpallman@sfu.ca or (604)-928-9269.

Sincerely,

D Pallmann

David Pallmann
President and CEO
Breakthrough Innovations Group (BIG)

BREAKTHROUGH INNOVATIONS GROUP

Go BIG or Go Home

ENSC 305W/440W

Design Specification:

Omega Key

Team Members:

David Pallmann
Frank Tran
Chase Kwak
Steven Luu
Steven Timotius
Steven Liu

Contact Person:

Chase Kwak – ckwak@sfu.ca

Submit to:

Dr. Andrew Rawicz – ENSC440W
Steve Whitmore – ENSC305W
School of Engineering Science
Simon Fraser University

Issue Date: 10/03/2016

Revision: 1.0

Abstract

The design specification of the Omega Key dynamic display keyboard is aimed to provide a set of detailed description of the design and development of our proof-of-concept model of the Omega Key. Thus, all discussions and considerations mentioned in this document are pertaining for the functional requirements labelled I and II, as specified in the document Functional Specification for the Omega Key.

This document outlines the design specifications and process of the Omega Key. It shall provide a detailed description of our proof-of-concept model as well as justifications for parts chosen. Furthermore, we will discuss potential improvements for future designs.

The OLED displays chosen are 0.49" diagonal with 64x32 resolution. Microcontrollers used are an Arduino Mega and an Arduino Leonardo with the possible need of an expansion adaptor depending on future needs. Key casings and frame casing shall be done through 3D printing designed in SolidWorks. Each OLED shall be housed in an individual key casing which shall subsequently be housed inside the frame casing. Each OLED shall have its connections connected to the microcontroller to ensure display functions. Key cases shall also house a Cherry MX Blue switch connected to the microcontroller which shall be responsible for typing inputs. The microcontrollers shall be responsible for reading typing inputs and outputting it to the PC, as well as controlling the display status of each OLED display screen.

Detailed descriptions of the above resources and hardware are provided in this document. Software development and test plans are also found with this document.

Table of Contents

List of Figures

List of Tables

Glossary

Controller	The Microprocessor controlling the Keyboard
Firmware	The software running on the microcontroller
Host Software	The software running on the computer
Keyboard Layout	The mapping of character, symbol, macro and the corresponding images to physical keys
Macro	A series of pre-programmed instructions to be executed from a key press
LCD	Liquid Crystal Display
OLED	Organic Light Emitting Diode
BIG	Breakthrough Innovation Group
USB	Universal Serial Bus
HID	Human Interface Device
RAM	Random Access Memory
ROM	Read Only Memory
EEPROM	Electronically Erasable Programmable Read Only Memory
Leonardo	Arduino Leonardo microcontroller
Mega	Arduino Mega microcontroller

1. Introduction and Background

1.1 Introduction

The Omega Key dynamic display keyboard offers users the ability to customize and remap the function of each individual key. This include various languages, symbols, specialized characters, and macros. Each key possess an OLED screen visible to the user that shall display the current function of the corresponding key. This will offer users the ability to easily distinguish between different keys and their functions. Furthermore, the mechanical design of the keyboard offers typers unparalleled ergonomics that cannot be achieve with any type of touch-screen keyboards. Our aim is to break through from traditional QWERTY layouts and bring typing into the future for people of all backgrounds. The design specification describes the technical details and component choices of our proof-of-concept.

1.2 Background

The motive behind designing the Omega Key is to minimize the frustration of using a standard QWERTY keyboard for purposes that are not centered on the English language. For users with a preference in a different language where the alphabet is not the 26 found in English, professional who may prefer specialized symbols and characters, or video game enthusiasts who want to remap their control keys; the Omega Key shall provide such desired customizability.

1.3 Scope

This document describes the design specifications that need to be met by Omega Key. It will identify how and why our design decisions meet the requirements listed in the functional specification document. Unless specified otherwise, the design specifications will only apply to the proof-of concept model. The design specification will be divided into hardware, firmware, and software sections. Justification for each design choice will be given. A test plan will provide a list of high level tests for evaluating overall performance of the proof-of concept model. This document refers to our team's functional specification throughout the document.

1.4 Classification

Throughout this document, the following convention shall be used to refer to functional requirements from the functional requirements document:

[R**n-p**]

where **n** is the functional requirement number and **p** is the priority of the functional requirement as denoted by one of the following three values:

I	The requirement applies to the proof-of-concept system only.
II	The requirement applies to all versions of the system.
III	The requirement applies to the prototype and final production system only.

2. System Design

Overall Omega Key design is divided into hardware, firmware and software components. Software component discussed later in section 3.4 will only be for the final production model and will allow the user to interactively program the keyboard layouts. Due to limited time & cost, Omega Key prototype will feature in 3 pre-defined layouts that user can choose from. The detail of each component will be covered in section 3 of the document.

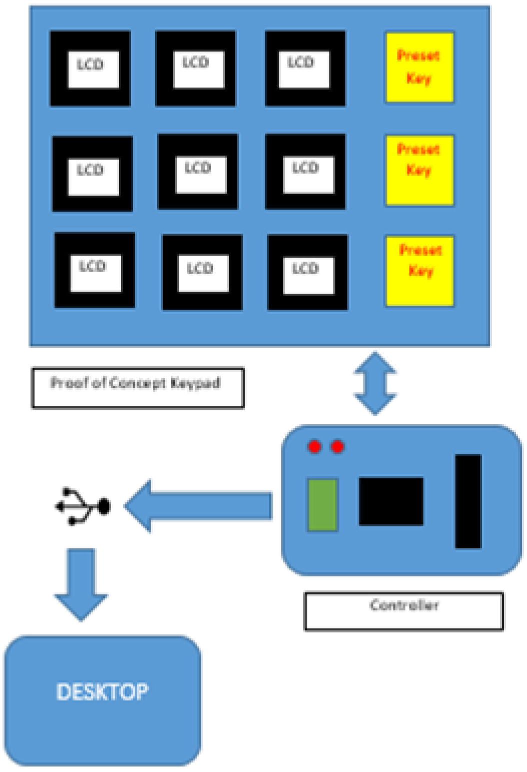

Figure 1: Proof of Concept's System Diagram

Omega Key is a customizable OLED keyboard (user customization function will only be applicable to the final product) which can display different screen layout depending on user's preference. If user wants to type in different language for example, they can press one of the pre-defined buttons, which will update the current display into a new display. This unique function provides users with options to type in different language / symbols that they prefer.

The main system components are consisted of: 2 microcontrollers, 12 64×32 OLEDs, 12 keyboard switches, customized buttons and casing. The main control unit of the Omega Key are the Arduino Mega and Arduino Leonardo, which will be responsible for communicating with the computer CPU and controlling the OLED displays. The OLEDs act as output devices and will display black & white images. Cherry MX Blue switches will be used to give users tactile feedback of the key presses. Each key press will be recognized by Arduino Leonardo.

Customized buttons will be 3D printed to hold each key switch and OLED in place. Casing will accommodate necessary wiring, buttons and keys, satisfying [R12-III].

3. Keyboard & Controller

3.1 Keyboard Mechanical Design

3.1.1 KeyCap Design

The key has to be designed with several critical requirements in mind. The key shall be comfortable to rest fingers on [R13 - III], the size of the key shall not have more than a 0.7' diagonal [FS Table 2], the user shall not be able to touch the screen, the key shall be able to click onto the mechanical switch [R33 - II], and the key shall allow easy access to wire the OLED.

With all of the aforementioned specifications in mind we decided that due to the small and complex nature of the key, the only way to accomplish these tasks were to 3D print. We completed the modelling of the key in SolidWorks.

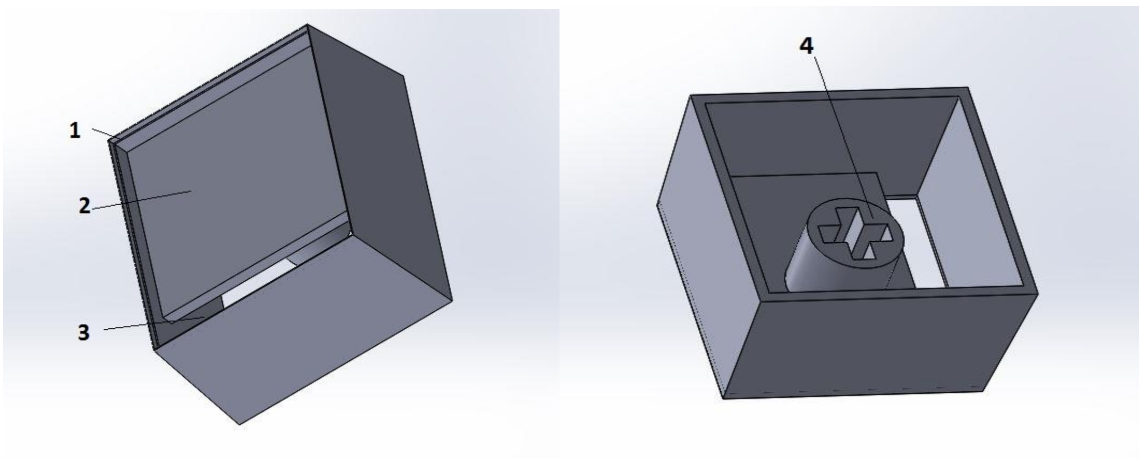

Figure 2: Key Cap Design in SolidWorks

Annotation #	Description of Design
1	Lip for piece of acrylic to sit in
2	Shelf for OLED screen to sit on
3	Hole for ribbon from OLED module to drop down
4	Connector between keycap and cherry MX switch

Table 1: Key Cap Annotations

3.1.2 Switch Bed Design

The switch bed, as the name suggests is a place for the mechanical switches to rest in. This is necessary as the OLEDs sit on top of the switches and we need to have room for wiring (of both the switches and OLED screens). It has 12 places to put in keyswitches [R19 - I] and three preset button holders [R7 - II][R21 - II]. This design of the switch bed allows the user to intuitively switch between presets [R29 -II].

Because of the complex nature of the number of cuts and complex nature of them we decided that 3D printing was also best for the switch bed.

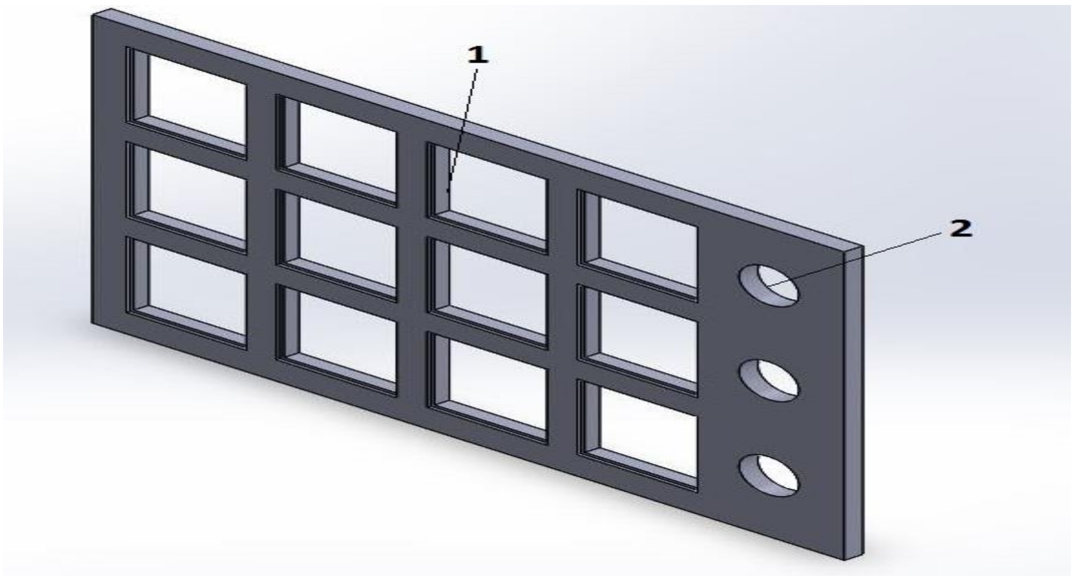

Figure 3: Switch Bed Design in SolidWorks

Annotation #	Description of Design
1	Hole for Cherry Mx Switch to sit in
2	Hole for Preset button to sit in

Table 2: Switch bed Annotations

3.1.3 Case Design

The case of the keyboard provides two major purposes. It allows the switch bed and by association the OLED screens to be raised to the optimal height. As well as acting as a place to hide the wiring for safety and easy access [R18 - II]. This component is by far the simplest piece of hardware and as such we decided to make it out of wood which also insulates the case [R50 - II]. Note that the height of the keyboard may be decreased to ease strain on users hands [R11 -III][R51 - II].The final production model shall be designed with more ergonomics in mind through study of previous keyboards and test user feedback. We shall stick rubber pads underneath the casing to prevent sliding and scratching [R14 -II].

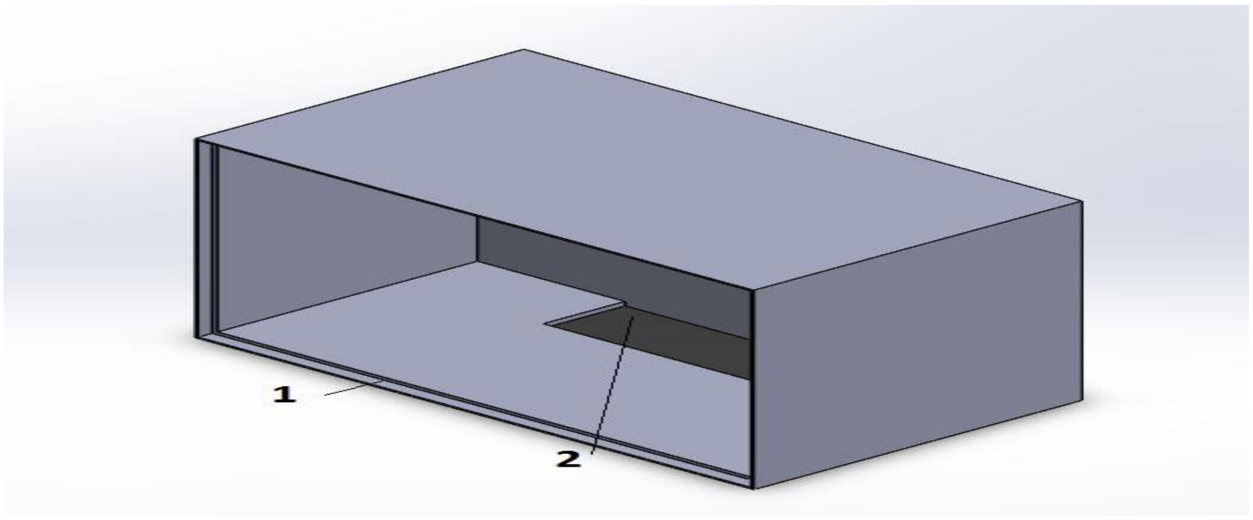

Figure 4: Case Design in SolidWorks

Annotation #	Description of Design
1	Lip for switch bed to sit in
2	Hole for extra wiring to come out of (power cables etc.)

Table 3: Casebed Annotations

3.1.4 Model of Completed assembly

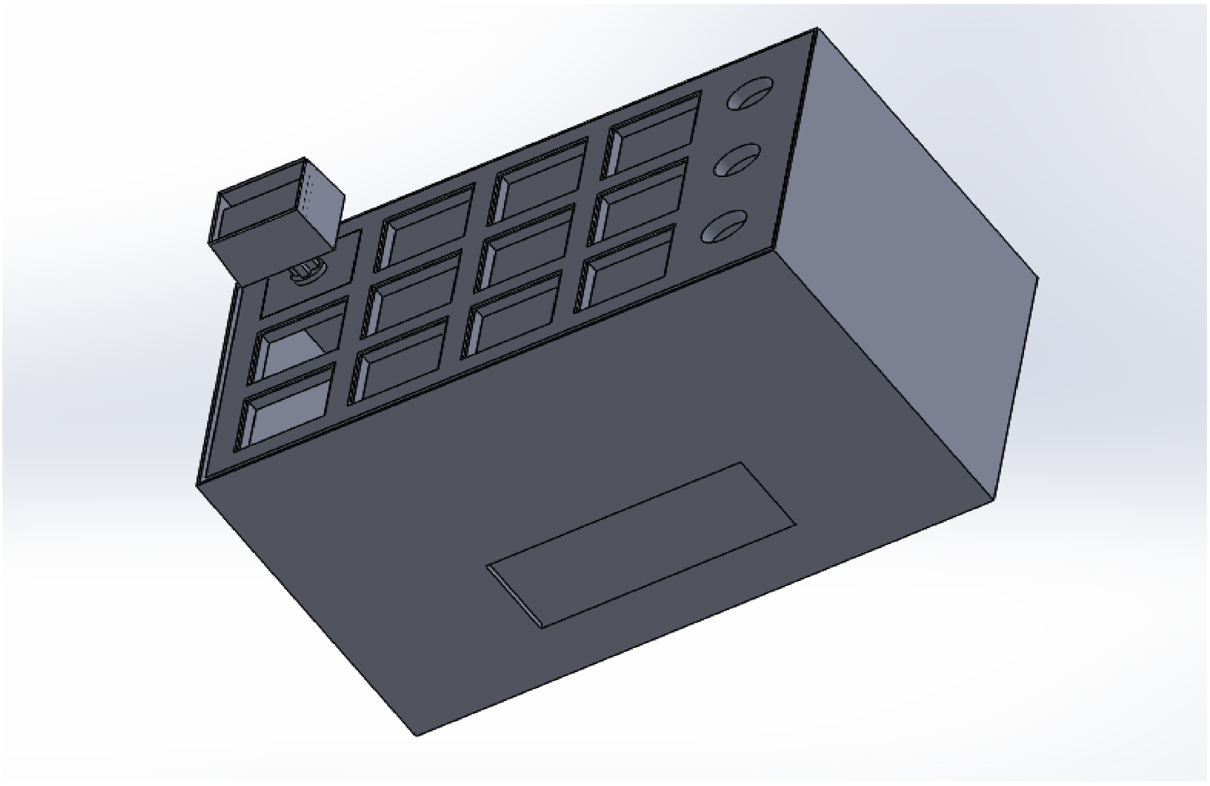

Figure 5: Completed assembly design with one keycap in SolidWorks

3.2 Keyboard Hardware Components

3.2.1 Key Switch

We decided to use a Cherry MX Blue for the key switch as it was the most recommended mechanical keyboard switch available[R31-II]. This is one of the most reliable switches on the market and as such satisfies all of our durability requirements [R32 - III].

Figure 6: Cherry MX Switch

3.2.2 OLED Display Module

For OLED displays, 0.49” OLED Display Module is chosen for the small size which will fit the size of the key casing. The OLED Display also uses I2C protocol, which will lead to less wiring needed for interfacing with the microcontroller, making it easier to satisfy [R16-II].

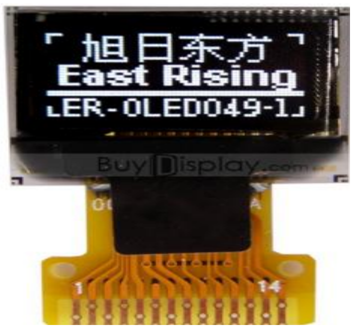

Figure 7: 0.49” OLED Display

Model	0.49” OLED Display
Appearance	White on Black
Number of Pixels	64 ×32
Interface	I2C
Connection	FPC - Soldering
Visual Area	12.58 ×6.98 mm
Power Supply	2.8 V
Number of pins	14
Microcontroller Compatibility	Arduino, Rasberry Pi, 8051, PIC, ARM

Table 4: OLED Specifications

Figure 8 shows a connection diagram between an OLED and Arduino Mega. Two pins from OLED are going to be hooked up to Arduino Mega to receive / transmit data. The other two pins from OLED will be used for power supply & ground connections.

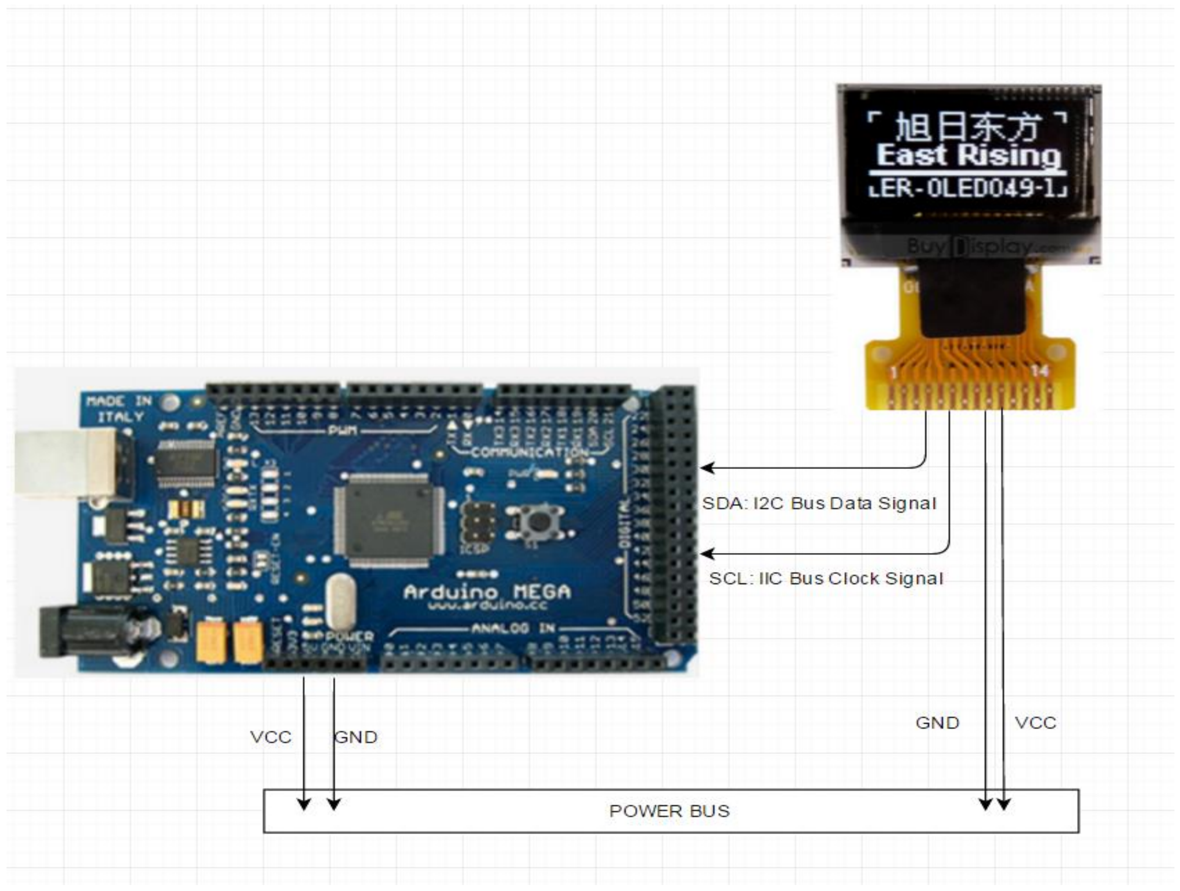

Figure 8: OLED Display Wring Diagram

3.3 Controller

3.3.1 General Design

For our proof-of-concept model, we will be using two Arduino boards, allowed by [R39-I]. Arduino board is also connected to computer through a USB 2.0 connection [R1-II] [R23-II]. The first board we are using will be the Arduino Leonardo, as it has better support for USB communications to satisfy [R38-II] [5]. This board will be used to handle detection of key strokes and communicate directly to the host computer. The second board will be the Arduino Mega. This is to accommodate [R35-II], which specifies the number of I/O ports; since the Leonardo by itself doesn't have enough while the Mega has many. As well, having two boards can modularize the project to facilitate people working concurrently. Both boards are readily available [R34-I].

Board	Arduino Leonardo [1]	Arduino Mega [2]
Clock Speed	16 MHz	16 MHz
Flash Memory	32 KB	256 KB

SRAM	2.5 KB	8 KB
Digital I/O Ports	20	54
EEPROM	1 KB	4 KB
Length	68.6 mm	101.52 mm
Width	53.3 mm	53.3 mm
Weight	20 g	37 g

Table 5: Basic Specifications for Arduino Leonardo and Mega

The proof of concept will have 4 predefined key layouts, with 12 keys each displaying a 64x32 black and white image. Therefore, about 17 KB of flash memory will be needed to store the image bitmap data, and the Mega has more than enough. The remaining memory will definitely be sufficient to hold the rest of the firmware [R37-II]. The two boards will communicate with each other via 4 digital lines connected to I/O ports. For the prototype of Omega Key, the two controllers may not be enclosed inside its casing due to the time constraints and cost. Operating temperature for Arduino Mega and Leonardo is between -40C to 85C [R22-II].

Figure 9 shows the Controller General Design for Omega Key. Once user presses a normal key (which is not a pre-defined key), Leonardo will recognize the key press input and retrieve the corresponding key code from its own memory unit. Then Leonardo will communicate with CPU to display the key on the screen. When the pre-defined button is pressed, Leonardo will prompt Mega to load a corresponding display layout and update each OLED.

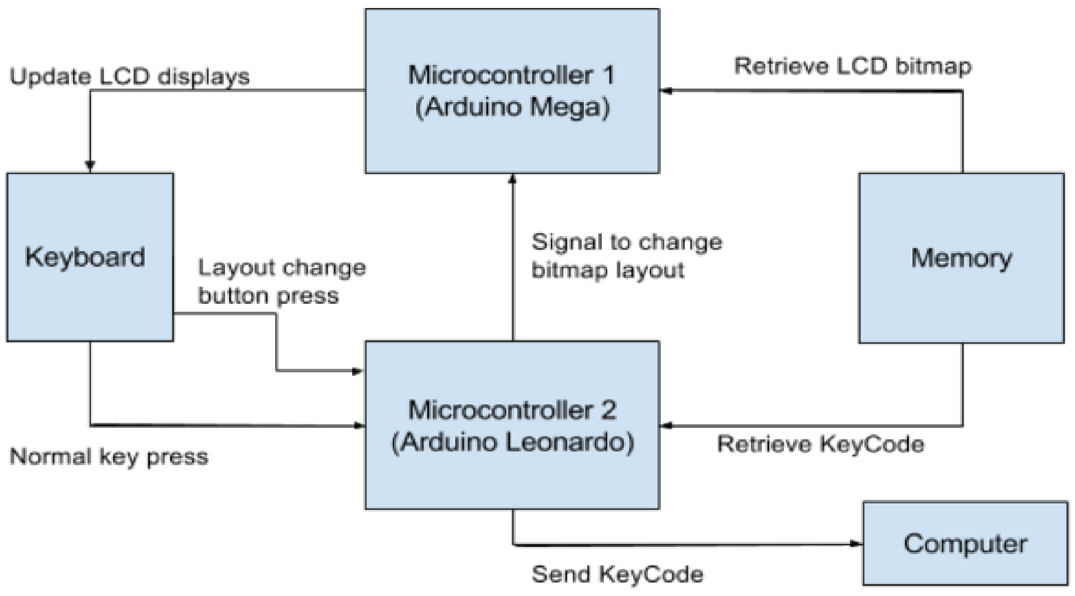
Update LCD displays
Microcontroller 1 (Arduino Mega)
Retrieve LCD bitmap
Keyboard
Layout change button press
Signal to change bitmap layout
Memory
Microcontroller 2 (Arduino Leonardo)
Normal key press
Retrieve KeyCode
Computer
Send KeyCode

Figure 9: Controller General Design (Proof of Concept)

Figure 10 shows the simple circuit design that will be used to connect the key switches to the Leonardo board. Key switch state will be determined by testing the voltage at the input pins.

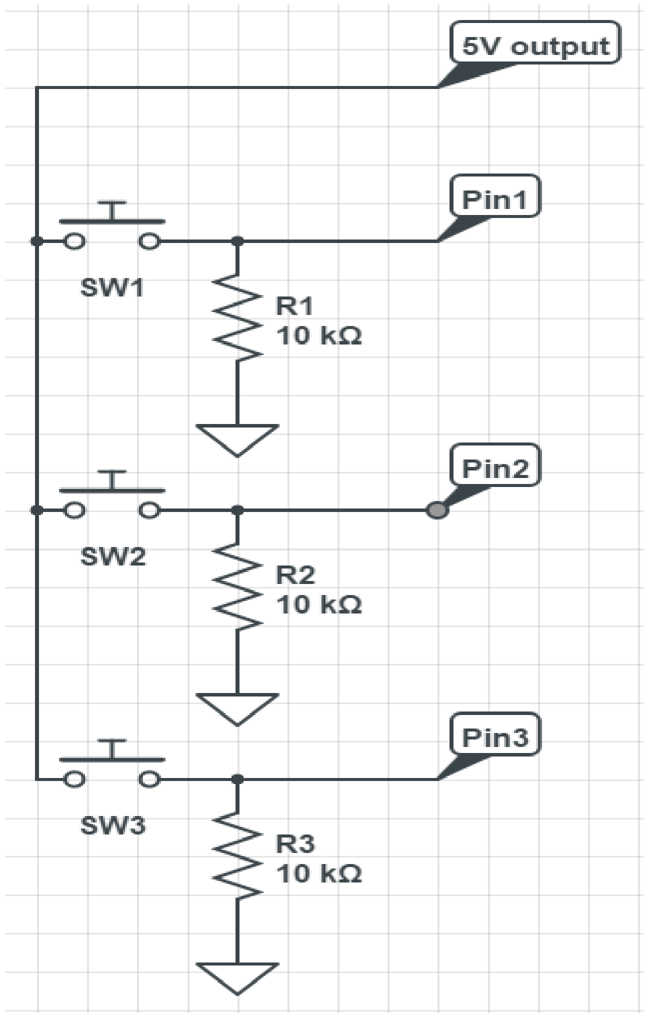

Figure 10: Basic circuit design for Leonardo board to connect to keys

(3 keys used as example, in reality would be 12 keys.
The ground would be connected to the ground pin)

The final product would not use an Arduino prototyping board. Instead, we would use a custom microprocessor and print our own circuit board [R36-III].

The following describes how the system will behave in the typical use cases: layout change and normal key press

- Layout change:
  - System is on standby
  - User presses layout change button X
  - The Leonardo recognizes button press and changes keycode map to layout X
  - The Leonardo then tells the Mega to change character map to layout X
  - The Mega then refreshes the OLED array displays
  - System goes back on standby
- Normal key press
  - User presses a key on the keyboard

- ○ Leonardo detects key press
- ○ Leonardo sends the corresponding character code/code sequence to the Computer
- ○ System goes back on standby

3.3.2 Firmware Design

3.3.2.1 Leonardo's Firmware

The Arduino Leonardo firmware will detect keystrokes and communicate with the host computer [R40-II]. The firmware will use a fast polling architecture, as shown in figure 11.

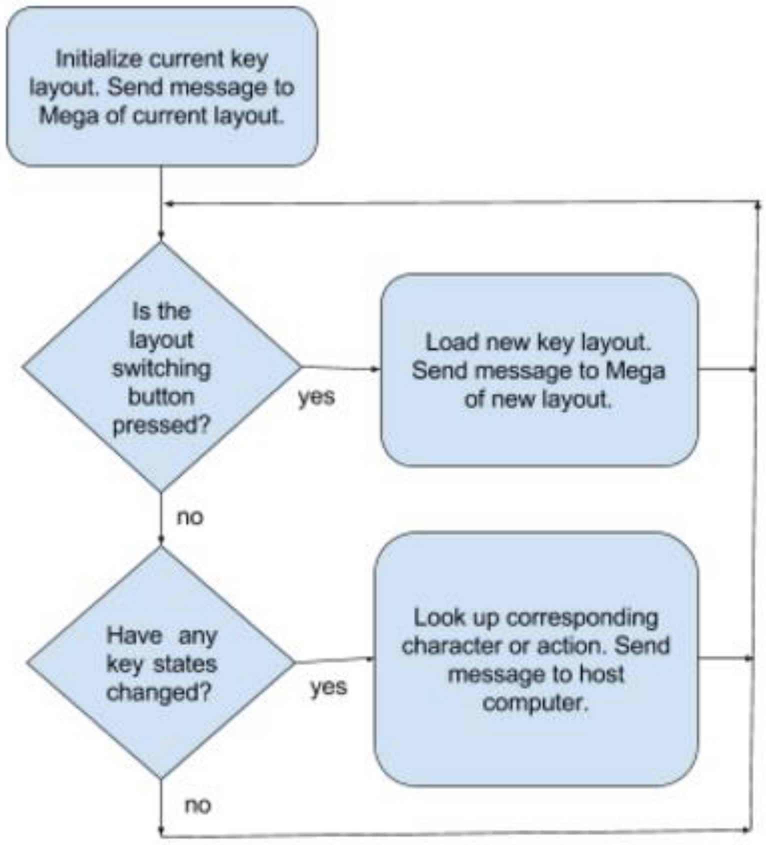
Initialize current key layout. Send message to Mega of current layout.
Is the layout switching button pressed?
yes
Load new key layout. Send message to Mega of new layout.
no
Have any key states changed?
yes
Look up corresponding character or action. Send message to host computer.
no

Figure 11: Simplified flowchart for Leonardo's firmware logic

The key layout will be preloaded onto the board's flash memory [R43-I]. The Windows operating system natively supports inputting special characters by key combinations, which can be triggered by the Arduino using the HID protocol [R8-II] [24-II] [R41-II]. The standard keyboard HID protocol is compatible with almost all modern consumer computers [R5-III]. However, the proof of concept model will be designed against a Windows desktop, allowed by [R4-I].

Low level communication using the HID protocol will be handled by a library natively distributed by Arduino [5]. On initialization, the Leonardo will transmit a keyboard report descriptor, the contents of which are laid out in Appendix A. When the key state is changed, the Leonardo will transmit a keyboard input report, which is described in Appendix B. [6]

3.3.2.2 Mega Firmware

The Arduino Mega firmware will not directly detect the user key input. Rather, it will be prompted by Arduino Leonardo to retrieve the new layout from Mega's memory unit when the user presses one of the four layout switching buttons. The main task of the Arduino Mega will be controlling the OLED array displays. The Mega will monitor communication from The Leonardo and refresh the OLED displays as needed. Figure 12 shows the Mega's firmware flowchart, which will satisfy [R7-II][R42-II]

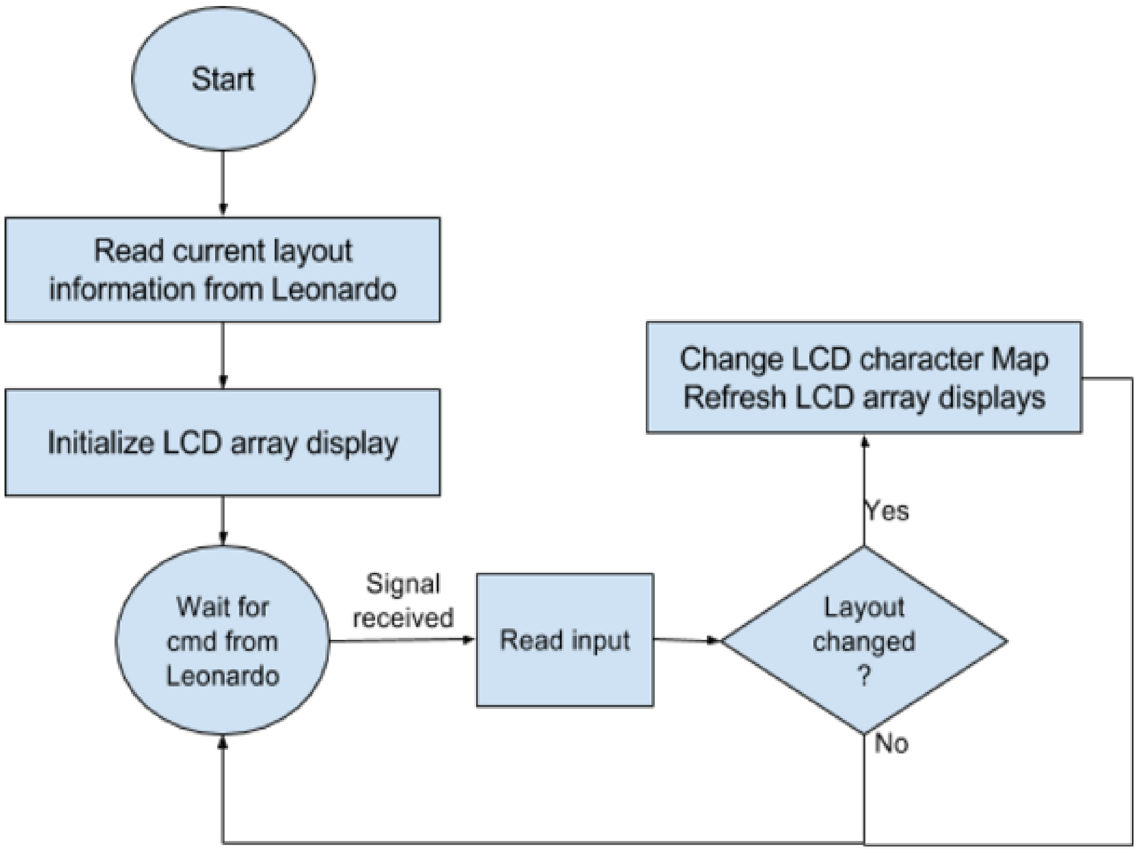
Start
Read current layout information from Leonardo
Initialize LCD array display
Wait for cmd from Leonardo
Signal received
Read input
Layout changed ?
Yes
No
Change LCD character Map Refresh LCD array displays

Figure 12: Simplified flowchart for Mega's firmware logic

3.3.2.3 Controllers Communication

The Arduino Leonardo will communicate the layout change information to the Arduino Mega using a simple 4 wires protocol as described below:

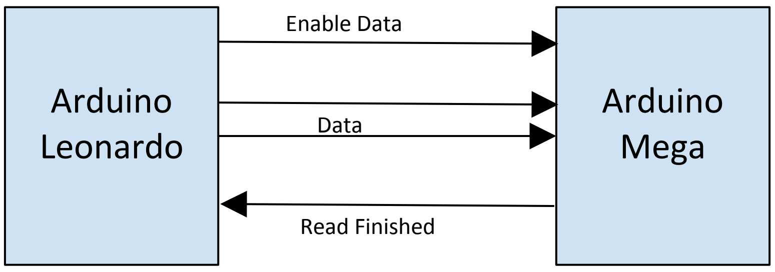

Figure 13: Simple Communication protocol between Leonardo and Mega

- The Leonardo will first send the layout number in parallel on the 2 data wires, then raise the 'Enable Data' signal. The Leonardo will then monitor the 'Read Finished' signal.
- The Mega will monitor the 'Enable Data' line, if it is raised, the Mega will read data from the 2 data wires.
- The Mega will then raise the 'Read Finished' signal after reading the data
- The Leonardo will then lower the 'Enable Data' and the Mega will also follow by lowering "Read Finished", finishing communication.

3.3.2.4 Layout Data definition

For the proof of concept, Keyboard layout data will be stored in the onboard flash memory. The data will be stored using the following format:

Address	Type (1 byte)	Data (15 bytes)
(Layout# * 12 + Key#) * 16	1: Regular Key 2: Symbol 3: Macro	Keycode Sequence

Table 6: Layout data stored on Leonardo (macros represent several key presses)

Address	BitMap (256 bytes)
(Layout# * 12 + Key#) * 256	64x32 bitmap array specifying pixel on/off

Table 7: Layout data stored on Mega

3.4 Host Software

Host software will not be required for the proof of concept model as stated in [R45-I]. As well, all versions of the product would be able to function with basic capabilities if no host software is present [R30-II]. However, below is the theoretical block diagram showing how host software would interact in the prototype and final product.

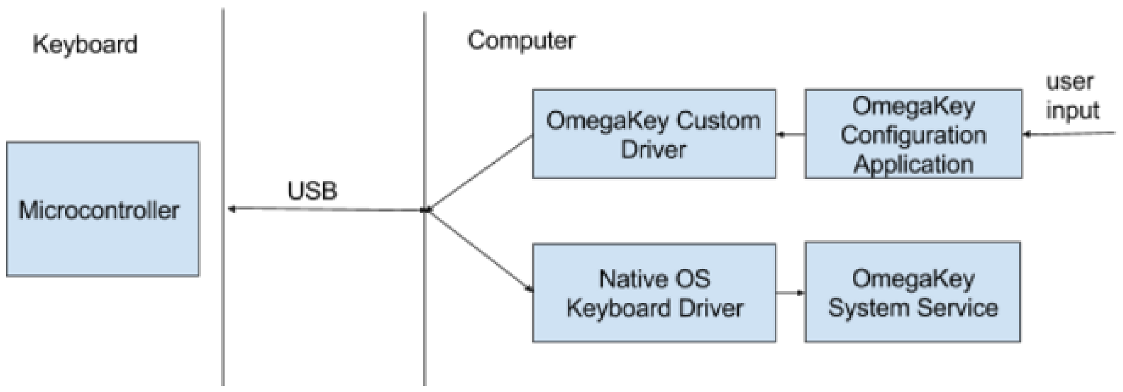

Figure 14: Block Diagram for Final Product Host Software

The user would interact with the configuration application, which would have an interface displaying an image of the full keyboard. When a key on the image is clicked, a popup would allow the user to configure an action for the button. A dropdown menu at the top would be used to select the layout being edited [R10-III] [R46-III]. The configuration application would communicate with the custom driver for low level communication to configure the keyboard [R47-III]. When a key is pressed, the keyboard sends messages that would be picked up by the normal keyboard driver directly. A system service running the background would further process the message if required, for extended functionality [R9-III] [R48-III]. All the buttons in the user interface would have floating text descriptions and distinct icons as well as a user manual would be distributed with the software [R49-III].

4. Test Plan

The Omega Key is targeted to a wide audience including professional and recreational usage. With this in mind the following test plan will ensure a high level of performance and usability. The integrated system testing will be performed on the final proof-of-concept model to confirm its functionality.

4.1 System Testing

4.1.1 Detection and output

The most basic function of the system is to be able to detect keystroke signals from the switches and output the corresponding character in Notepad.

Method: Press all the keys individually and see if the firmware is able to detect a keystroke signal from the corresponding keys.

Expected results: According to [R40 - II], the firmware will be able to detect a signal from the keys and outputs the appropriate characters in Notepad.

4.1.2 Layout

The system will need to be able to set a specific layout of characters on the keys and display the corresponding characters. The Omega Key will be able to store 3 layouts.

Method: Set the layout of the keys using the firmware and observe the results on the corresponding keys.

Expected results: The layout from the firmware is shown on the corresponding displays and each key shows the proper character with accordance to [R43 - I].

4.1.4 Switching Layouts

The integrated system is able to store 3 layouts and switch between them. The displays should update to the new layout in a short amount of time, and the system should output the appropriate character when the layout changes.

Method: Implement a set of layouts on the microcontroller. The preset buttons will then be pressed to cycle through the various layouts. For each layout each key will be pressed to see if the output character matches the character on the display. The accuracy of the layout, character accuracy, and the time it takes to update the display will be recorded.

Expected results: The keys of the system should be properly configured to match the characters of a new layout within 4 seconds of the preset button being pressed. All the keys will output the corresponding character on the display when the layout is changed.

4.1.5 Prolonged Usage

The Omega Key is intended for professional use so comfort and responsiveness of the keys needs to be tested.

Method: Spend roughly two minutes continuously typing on the Omega Key while observing the level of comfort and typing experience.

Expected results: The keys should feel secure, intuitive, and responsive while typing.

5. Conclusion

This document specifies all hardware and firmware design details of the Omega Key by Breakthrough Innovations Group. The corresponding functional specification of the system has been referenced throughout the document. During the development cycle the design specification document will be used as a guideline to guarantee the intended functionality of the proof-of-concept model. The included test plan will be used for quality assurances purposes and to ensure the system functions properly. The design specification clearly presents our methods and goals in the development of the proof-of-concept model.

Appendix A

Report Descriptor for HID Keyboard Protocol [6]

```
Usage Page (Generic Desktop),
Usage (Keyboard),
Collection (Application),
        Report Size (1),
        Report Count (8),
        Usage Page (Key Codes),
        Usage Minimum (224),
        Usage Maximum (231),
        Logical Minimum (0),
        Logical Maximum (1),
        Input (Data, Variable, Absolute), ;Modifier byte
        Report Count (1),
        Report Size (8),
        Input (Constant), ;Reserved byte
        Report Count (5),
        Report Size (1),
        Usage Page (LEDs),
        Usage Minimum (1),
        Usage Maximum (5),
        Output (Data, Variable, Absolute), ;LED report
        Report Count (1),
        Report Size (3),
        Output (Constant), ;LED report padding
        Report Count (6),
        Report Size (8),
        Logical Minimum (0),
        Logical Maximum(255),
        Usage Page (Key Codes),
        Usage Minimum (0),
        Usage Maximum (255),
        Input (Data, Array),
End Collection
```

Appendix B

Keyboard Input Report [6]

Byte	Description
0	Modifier keys
1	Reserved
2	Keycode 1
3	Keycode 2
4	Keycode 3
5	Keycode 4
6	Keycode 5
7	Keycode 6

Modifier Keys Byte Contents:

Bit	7	6	5	4	3	2	1	0
Key	LCtrl	LShift	LAlt	LGUI	RCtrl	RShift	RAlt	RGUI

(on Windows, the GUI key corresponds to the Windows key)

Appendix C

Test Plan

System Testing

Test	Test Method	Expected Results	Results
Detection and output	Press all the keys individually and see if the firmware is able to detect a keystroke signal from the corresponding keys.	According to [R40 - II], the firmware will be able to detect a signal from the keys and outputs the appropriate characters in Notepad.	Pass Fail
Comments			
Layout	Set the layout of the keys using the firmware and observe the results on the corresponding keys.	The layout from the firmware is shown on the corresponding displays and each key shows the proper character with accordance to [R43 - I].	Pass Fail
Comments			
Switching Layouts	Implement a set of 3 layouts on the microcontroller. The preset buttons will then be pressed to cycle through the various layouts. For each layout each key will be pressed to see if the output character matches the character on the display. The accuracy of the layout, character accuracy, and the time it takes to update the display will be recorded.	The keys of the system should be properly configured to match the characters of a new layout within 4 seconds of the preset button being pressed. All the keys will output the corresponding character on the display when the layout is changed.	Pass Fail
Comments			
Prolonged Usage	Spend roughly two minutes continuously typing on the OmegaKey while observing the level of comfort and typing experience.	The keys should feel secure, intuitive, and responsive while typing.	Pass Fail
Comments			

References

[1] Arduino Leonardo. [Online]. Available:
https://www.arduino.cc/en/Main/ArduinoBoardLeonardo [Accessed: March 7, 2016]

[2] Arduino Mega. [Online].
Available:https://www.arduino.cc/en/Main/ArduinoBoardMega2560
[Accessed: March 7, 2016]

[3] OLED Display Module [Online]. Available:
http://www.buydisplay.com/default/0-49-inch-oled-display-module-64x32-pixel-ssd1306-spi-i2c-white-on-black [Accessed: March 7, 2016]

[4] Cherry Mx Datasheet [Online]. Available:
http://cherryamericas.com/wp-content/uploads/2014/12/mx_cat.pdf
[Accessed: March 5, 2016]

[5] Mouse and Keyboard Libraries [Online]. Available:
https://www.arduino.cc/en/Reference/MouseKeyboard [Accessed: March 5, 2016]

[6] Device Class Definition for Human Interface Devices (HID) [Online]. Available:
http://www.usb.org/developers/hidpage/HID1_11.pdf [Accessed: March 5, 2016]