**Ωmega Vest**

**Functional Specification**

Version Number: 5

Revision Date‐Time: 12/4/2014 12:20:00 PM

Design Team: Ryan Foxworth, Omar Halabi, Juan Lopez Marcano, Justin Parker, Seng Loong Yu

Faculty Advisor: Dr. Andrew Raij

**Version Status Date Edited By Description of Change**

1 Draft 4/11/14 J. Parker Start of draft

2 Draft 4/11/14 J. Marcano Continuation of draft

3 Draft 4/16/14 R. Foxworth Completion of draft H. Yu

O. Halabi

4 Final 4/25/14 J. Parker Proofreading

J. Marcano

5 Final 12/4/14 J. Marcano Update

***Confidential***

***This document contains confidential and proprietary information of the*** *Ω****mega Vest. Any reproduction,***

***disclosure, or use in whole or in part is expressly prohibited, except as may be specifically authorized by prior***

***written agreement or permission.***

**Table of Contents**

**1 OVERVIEW**

1.1 Description

1.2 Scope

1.2.1 Objectives

1.2.2 Non-Objectives

1.3 References

**2 FUNCTIONALITY**

2.1 Existing Functionality Details

2.2 Proposed Design

2.2.1 Description of Functionality

2.2.2 User Interface Changes

2.2.3 External Interfaces

2.2.4 Process Flow

2.3 Alternate Design Options Considered

2.4 Design Decision Summary

**3 CONSIDERATIONS**

3.1 Dependencies and Assumptions

3.2 Risks

3.3 Managed Issues

**4 EFFORT**

4.1 Original Estimate from SOW

4.2 Latest Estimate

4.3 Reasons for Change in Estimate/Scope

**5 SIGN-OFF**

**1. Overview**

The Ωmega Vest is an entertainment device whose ultimate goal is to enhance user experience by reproducing effects of movies in a wearable vest. The system will address the user experience issues by implementing the following functionalities:

* Make a video player using for Windows OS using Microsoft Visual Studio 2010. It will also be able to read comma-separated-value (CSV) files that will determine how the effects will be played and send signals serially to the microcontroller.
* Synchronize the video player with the hardware inside the vest. To achieve this goal, we will use an Arduino Uno, which will receive signals serially from the computer to produce the effects.
* Make the vest vibrate. This effect can be activated, for instance, when the movie displays earthquakes, a car crash, etc.
* Blow air. This effect could be applied when the movie displays a fall, a person bicycling, etc.
* Spray scent. This effect can be activated when, i.e., the movie displays cinnamon rolls being baked.

**1.1 Description**

The Ωmega Vest system will be comprised of three distinct subsystems: video player, microcontroller that interfaces with the hardware of the vest, and the output devices – vibration motors and pneumatic scent and compression systems. The video player will be user friendly: it will use common icons for playing, resuming, stopping, exiting, and adjusting audio. It will read CSV files, parse them, and send string signals serially to the microcontroller to specify which effects will be on and which ones will be off.

**1.2 Scope**

The Ωmega Vest will be initially intended for use at home. As its popularity increases, it may be adapted to work in theaters and with gaming devices. Once the user opens the video player, he or she will be able to play movies and enable Ωmega Vests to synchronize with the movies. Unfortunately, the number of Ωmega Vests enabled per computer is dependent on the computer’s processing power and USB bus, which could be from 1 to 10.

**1.2.1 Objectives**

The purpose of this project is to address the need to improve user experience when watching movies. We believe that we have reached a point where impressive, colorful displays, and even 3D displays, are not as entertaining and breathtaking as they used to be. Thus, the objective is to exploit the power of other sensations to improve user experience.

As stated in the preferred requirements document, the software will be developed using Microsoft Visual Studio 2010 (C#, .NET framework) because it has multiple libraries that facilitate the creation of user interfaces, while also supporting robust communication with peripherals, such as USB-controlled or USB-powered devices. As microcontroller, we chose the Arduino Uno. We chose it because it has a large number of pins, which are used for sending signals. Thus, if we wanted to provide more effects, or have effects in more regions of the vest, we would be able to do that with the Arduino Uno. The other reason is that the Arduino required the smallest learning curve for the members of the team.

**1.2.2 Non-Objectives**

The Ωmega Vest is a product that is almost unique in its kind. Similar products have been created, but they are actually intended for gaming devices and rides in theme parks. In other words, the Ωmega Vest will not only be a new product, but also a new market. Thus, there are certain aspects that we will not address (non-objectives).

Video Format: As stated previously, the video player will send text files to the microcontroller and play the video. However, there should a video format that would encode video and text files. Just like there is a format for HD Video and 3D Video, which was created recently, we are assuming there is a format for Ωmega Vest.

Software Development Kit (SDK): Creating an SDK is not part of the scope of this project. However, an official SDK would allow developers and recording companies to create powerful interactive experiences by tailoring the effects that would be reproduced by the vest.

Routing of Ωmega Vests: As we have stated before, the product is initially intended for use at home. Making it available for theaters was beyond the scope of this project because it would exponentially increase the complexity of the algorithm to send signals to the microprocessors. Thus, we constrained the scope to use at home.

**1.3 References**

Preferred Requirements Document

**2 Functionality**

**Providing an Overview of the Functional Architecture**

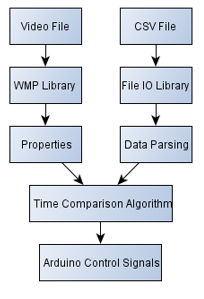
Currently, the Ωmega Vest currently comprises three subsystems

* Video player
* Microcontroller
* Vibration motors and pneumatic scent and compression system.

The video player is the interface the user will use throughout the Ωmega Vest experience. It is responsible for playing the video files, detecting one or multiple microcontrollers, sending data to the microcontroller, synchronizing video with sensory effects, and allowing the user to perform all the customizations that he or she could do while using any commercial or open source video player program. Thus, the video player is in charge of performing computations and data transmission.

The microcontroller is the interface between the video player and the devices that produce physical effects. It is in charge of receiving the data from the Omega Player and controlling the devices.

Lastly, the vibration motors and the air compression system are the relay devices. They perform the tasks assigned by the microcontroller. Refer to the Preferred Requirements document for information on how they work and how they will be integrated in the system.

* 1. **Existing Functionality Details**

The current design of the Ωmega Vest utilizes a custom video player for playback of video files and synchronization with the microcontroller. It decodes a CSV that specifies which will be turned on and how long. A time comparison algorithm is run every 50 milliseconds to determine the location of the video cursor and send signals to the microcontroller so that the effects can be felt as soon as they occur in the movie (see figure 2.1).

Figure 2.1: Ωmega Player flowchart

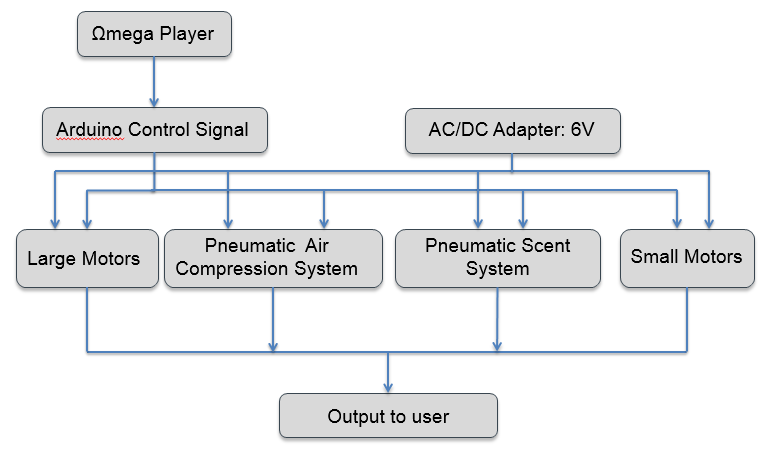


Figure 2.2: Block diagram description of the Ωmega Vest System.

At present time, the system meets all the requirements listed in the Preferred Requirements document and functions as specified in figure 2.2. The Ωmega Player handles synchronization with the movie, ensures that playback capabilities are supported, parses the CSV file, and sends the data to the Arduino. The Arduino is in charge of turning and off the vibration motors and making the servos for the pneumatic compression and scent system turn 35 and 65 degrees respectively. The servo that controls the air compression system will actuate a valve that will allow the allow air to come out of tubes. The servo that controls the scent system applies torque to a canister that sprays cinnamon scent. All the relay devices are powered by a 6V AC/DC adapter.

**2.2 Proposed Design**

Details on how the pneumatic system for air compression and the vibration modules can be found in the Preferred Requirements document. Similarly, the Preferred Requirements document provides a detailed explanation on the user interface.

To address synchronization after pausing or resuming, we propose a timer-based system that will be created once the text file is opened. The timer will tick every 50 milliseconds and an event listener will be called to handle this event. Figures 2.3, 2.4, and 2.5 show the algorithms implemented to synchronize the video.

**if(video.pause()==true){**

**turnOffSignals()**

**}**

Figure 2.3: Algorithm for handling pauses.

**if(video.resume()==true){**

**CurrentTime= PreviousTime**

**if(Microcontroller.isWorking()==true){**

**turnOnSignals();**

**}**

**}**

Figure 2.4: Algorithm for resuming effects

**if(video.playing()==true){**

**//video has been moved back**

**While(CurrentTime < PreviousTime){**

**If(effectPosition=0) break**

**effectPosition = effectPosition -1**

**PreviousTime = convertToTime(Effect[0])**

**FutureTime = convertToTime(Effect[1])**

**listContents = ParsedCSV[position];**

**}**

**//Video has been moved forward**

**While(FutureTime < CurrentTime){**

**If(effectPosition=max) break**

**effectPosition = effectPosition +1**

**PreviousTime = convertToTime(Effect[0])**

**FutureTime = convertToTime(Effect[1])**

**listContents = ParsedCSV[position];**

**}**

**}**

Figure 2.5: Algorithm for resuming effects

**2.2.1 Description of Functionality**

Functionalities have not changed from those of the previous specifications and those of the Preferred Requirements document. Refer to section 2.1 to find a high-level description of the functionalities and refer to the Referred to the Preferred Requirements document to obtain more detailed explanations on how the subsystems work.

**2.2.2 User Interface Changes**

There are no major changes in the user interface. We tested the interface with potential end-users and their suggestion is to make changes in the fonts and backgrounds of the multiple screens that can be reached while interacting with the video player. Refer to the Preferred Requirements document to understand how to interact with the user interface of the Ωmega Vest.

**2.2.3 External Interfaces**

Some changes have been made to the external interface. The Raspberry Pi used to be our choice, but we switched to the Arduino. We realized this was the better choice because we did not need a microcomputer to send and receive signals. Part of the original design was to allow the raspberry pi to perform some computational work, but we realized all the computational work can be performed in a single unit, which is the computer the Ωmega Player will be installed.

**2.2.4 Process Flow**

Since there are no changes in the functions of the system and how components interact with one another, the process flow will be as described in Figure 2.1.

*Functional Specification*

**2.4 Design Decision Summary**

Upon reviewing the proposed design changes, it is decided that the project must meet with the design specifications illustrated in Figure 2.2 and Figure 2.3. Additionally, it must also follow the user experience guidelines in sections 2.2.1 to 2.2.3.

**3 Considerations**

**3.1 Dependencies and Assumptions**

Please refer to section 2.5 of the requirements document for the discussion of dependencies and assumptions.

**3.2 Risks**

* + 1. **Safety Risks**

There are safety and health risks whenever a product contains electronic components. Currently, our design approach is first not use a battery, as it will significantly increase safety risks and delay our product’s working prototype. Also, to make sure our system does not explode while in use, we decided to have a separate housing specifically for high power consumption devices, that mainly being the air compressor.

* 1. **Managed Issues**
     1. **Video and Ωmega Vest Syncing**

Since the Ωmega Vest prototype will feature a computer application with video playback capabilities, synchronization between the vest and the video playback must be implemented. When the video pauses/resumes, the vest must perform accordingly as well.

To address this issue, Juan Lopez Marcano and Seng Loong Yu developed the time comparison algorithm explained in section 2.2. The code has been tested under average, best, and worst case scenarios, as dictated by the lifecycle of a program. The code has passed all our tests successfully, and thus, is the one the Ωmega Player currently uses.

* + 1. **Power Distribution for Different Components**

Since the Ωmega Vest is made of different electronic components, the task of distributing the right amount of power to each component is a daunting one. In order for each part of the Ωmega Vest to perform as expected, they each need the right voltage/current.

Ryan Foxworth and Justin Parker designed and tested a power distribution circuit that functions using Bipolar Junction Transistors (BJT) and diodes to protect the motors. The circuit was designed by hand and tested using PSpice, and we found that it provides the components with the voltages and currents needed. A circuit following these specifications was assembled in a prototyping board and is currently being used by the current version of the Ωmega Vest.

* + 1. **Arduino-USB Serial Crosstalk**

To verify that the Arduino was receiving the signals being sent serially, a logging algorithm was developed to keep track of incoming serial messages. Unfortunately, the logging information obtained via USB caused crosstalk between the USB and the Arduino. To address this issue, we switched to logging via Arduino IDE.