New Sneakers Final Project Report

Nick Ebert, Tim Echtenkamp, Ray Page, Matt Rosno
Department of Computer Science and Engineering

✦

1 INTRODUCTION

This project report introduces the Jarvis Home Automation System. Jarvis is an interactive information hub for your home. Users of the system can connect to home security cameras and view digital media, the weather, and other useful information. A unique and intuitive user interface, based on the Microsoft Surface platform, gives users a fun and interactive new way to view media and information.

Home automation is not necessarily a new concept. The concept of being able to speak aloud to an intelligent computer has been around as long as computers have been around. For awhile, this was only science fiction: reserved for Star Trek and similar titles. However, home automation has become a reality. Products that use X10, the de-facto standard format for home automation, have been on the market since 1978 [3]. The first systems only turned on and off lights based on a timer, but the evolution of home automation has brought advanced control to manage power within the home. Modern systems can regulate thermostats and water heaters based off of the homeowners schedule, which is done in an effort to save energy.

The Jarvis Home Automation system takes a different approach than traditional home automation systems. Instead of controlling lights, appliances, and heating and air conditioning units, Jarvis differentiates itself by placing the focus on how you would like to interact with your house. The Jarvis Home Automation system revolutionizes your living room in a fun and exciting way so you can sit back, relax, and let Jarvis do the work.

2 DESIGN CONCEPT

The team had originally intended to develop some form of a home automation system. The opportunity to work with the Microsoft Surface Platform greatly influenced the design of the system. The following sections describe the design of the system in more detail: *Departure from Traditional Home Automation, Integration of Different Control Mechanisms, and Living Room Centric Design.*

2.1 Departure from Traditional Home Automation

The team evaluated several approaches to traditional home automation. After evaluating typical home automation tasks such as light control, measuring energy consumption and lock automation, the team drifted away from the standard topics of home automation. The Jarvis Home Automation System automates information in the home. Instead of controlling lights and locks, Jarvis controls home media and other information sources around the home. The departure better suits the goal of the project - to make the life of the homeowner easier.

The main driver for the departure from traditional home automation was the availability of the Microsoft Surface, the main interaction point for the project. This motivated the group to present users with more information than traditional home automation.

2.2 Integration of Different Control Mechanisms

Since the team chose to work with the Microsoft Surface, the primary way to interact with the system is by using a touch screen. The development team had the goal of integrating more than one way to control the system. The following diagram depicts the three additional ways (other than touch) to interact with the system.

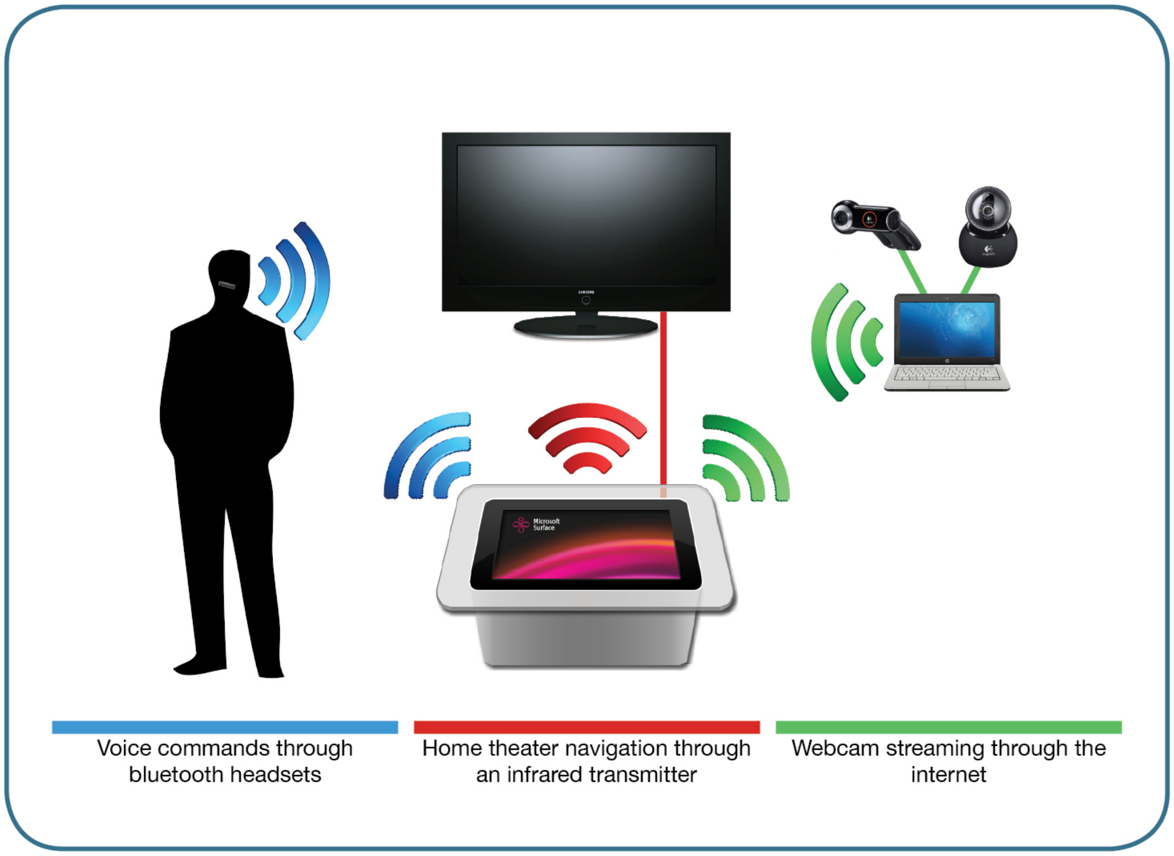
Voice commands through bluetooth headsets
Home theater navigation through an infrared transmitter
Webcam streaming through the internet
Microsoft Surface

Figure 1: Additional interactions in the Jarvis Home Automation System.

2.3 Living Room Centric Design

After the team discovered the capabilities of the Microsoft Surface, the design and layout of the system's components became more intuitive. The home screen of the Jarvis Home Automation system mimics the layout of a living room. This home screen is sometimes referred to as the "virtual living room". Each feature is accessible by touching a unique object in the virtual living room. For example, users can access the media center library by touching the bookshelf. This interface makes the system much easier to use.

The physical characteristics of the system also make it suitable to be placed in a homeowner's living room. Having the Microsoft Surface as a coffee table allows users to be within arms reach of information about their home.

3 Analysis of Design and Architecture

The design of the system changed considerably over the course of the project. Due to the multifaceted nature of the system, the architecture is not uniform across components. The following sections include more information that explain the design and architecture of the components: *Living Room Architecture, Video Analysis and Architecture, Data-Driven Architecture, and Universal Remote Architecture.*

3.1 Living Room Architecture

The virtual living room is the main screen in the application. From here, users can access all features of the system. The following image is a screenshot of the home screen.

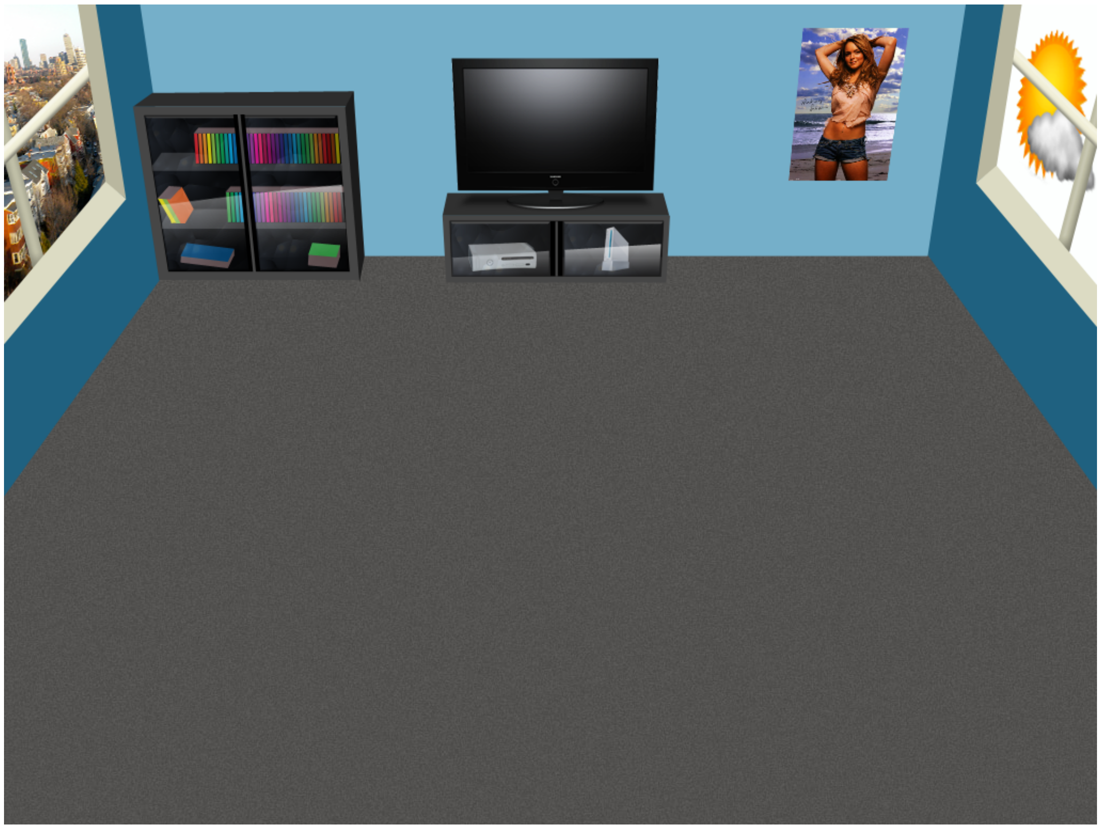

Figure 2: The main components of the system.

From the home screen, users can access:

- *Surveillance Video* - By touching on the left window, the user is taken to the main video screen.
- *Media Collection* - By touching the bookshelf to the left of the TV, the user can view items in the media collection.
  - *Media Center* - The user can drag items from the media collection onto the TV to view them on the connected display. The universal remote is also activated when this occurs.
- *Universal Remote* - By touching the TV in the center of the home screen, the user can view the universal remote and control connected devices.
- *Weather* - By touching on the right window, the user is taken to the five-day weather screen

The main screen handles the scatterview, a container that can display many different pieces of information in a single display. This allows the natural sorting of items in the media collection. Also on the main screen, controls for voice navigation are implemented. The system is designed to allow users to navigate using voice commands from anywhere in the system.

Having one main screen also has its drawbacks. All of the features of the system have to be working in order to show the living room screen correctly. If one component is not working, the living room main screen will not work as well. The team feels that this is an acceptable risk because of the organization that the interactive living room presents.

3.2 Video Analysis and Architecture

Video has always been a central component in the system. The architecture of video in the system has changed considerably from the first project update to the final system. In the beginning, webcams were attached directly to the Surface. This enabled the system to access high quality video. This approach was not very extensible, and it was also impractical because all cameras had to be physically connected to the Surface.

The intermediate approach to the video architecture was to use 3rd party applications to turn standard webcams into IP cameras that the Jarvis System could access. This solution was initially attractive because of the use of free, already built software. The team quickly discovered that this implementation would not work. Video performance from using third party software was unacceptable. The video lagged up to 20 seconds and the quality was degraded.

The final video architecture involved reworking a custom webcam streaming application into a Windows Service. This service is installed on computers with web cams, and these computers serve the video stream. The Surface gets the video stream directly through a socket connect to the web cam host computer. The following image depicts the final video architecture:

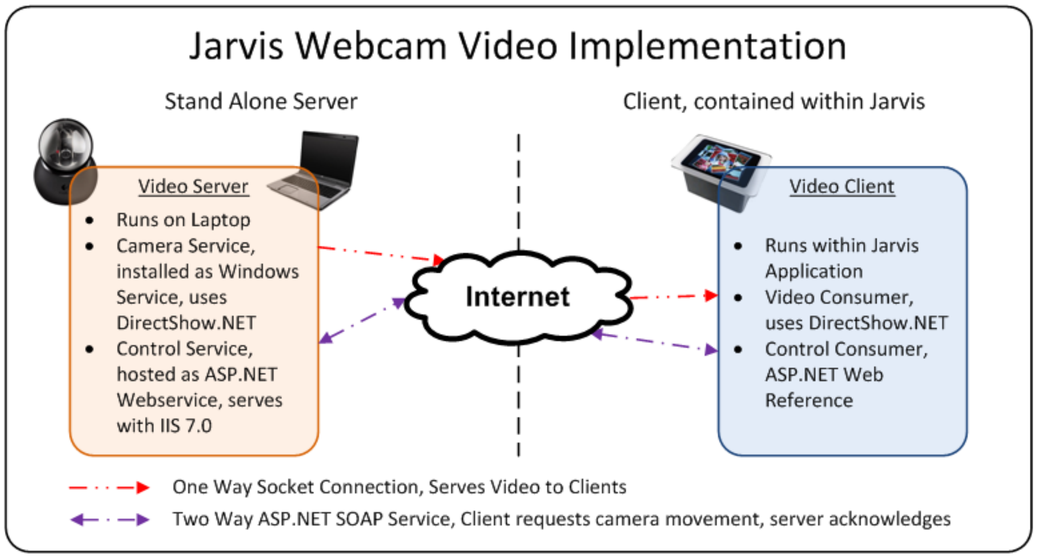
Jarvis Webcam Video Implementation
Stand Alone Server
Client, contained within Jarvis
Video Server
• Runs on Laptop
• Camera Service, installed as Windows Service, uses DirectShow.NET
• Control Service, hosted as ASP.NET Webservice, serves with IIS 7.0
Internet
Video Client
• Runs within Jarvis Application
• Video Consumer, uses DirectShow.NET
• Control Consumer, ASP.NET Web Reference
One Way Socket Connection, Serves Video to Clients
Two Way ASP.NET SOAP Service, Client requests camera movement, server acknowledges

Figure 3: Architecture of the video streaming feature of the system.

This architecture takes advantage of the DirectShowNET library for handling media in .NET. It allows the system to consume a high quality web cam stream with little modification to the application.

The primary motivation behind using a USB webcam instead of an IP camera is cost. IP cameras cost significantly more than webcams. The development team also wanted a more portable solution, and IP cameras are more complex than USB webcams to setup.

3.3 Data-Driven Architecture

An original goal of the development team was to integrate external data with the system to provide users the most current information available. The Jarvis Home Automation system pulls external data every time it is accessed. There is no caching of external data. The development team chose not to cache external data because of the policies that would need to be built into the system to expire cached data and pull new data.

The pulling of external data works like a visit to a website. For example, every time the media collection is accessed, DVD covers are pulled from the Netflix movie catalog. These covers do not exist in the Jarvis System; they are located in the Netflix catalog.

3.4 Universal Remote Architecture

A USB Universal Infrared Receiver/Transmitter (USB-UIRT) is used to interact with home theater components. The maker of this device provided a C# wrapper that was used in the WPF code-behind. Although the wrapper provided many different transmit formats, the UIRT-Raw format was used for the infrared codes.

Besides transmitting codes, the universal remote was also able to learn codes. This functionality was available in the C# wrapper and allowed the Jarvis Home Automation system store the learned codes directly into the database for immediate usage.

4 Implementation

Component implementation was dependent on the complexity of each component. Some components needed a lot of lead time and extensive research to implement. Others could be added once primary components were implemented. The following sections detail the implementation process: *Video Implementation, Audio Implementation, Media Center Implementation, Remote Implementation and Weather Implementation.*

4.1 Video Implementation

The surveillance video module of the Jarvis system required integrating several technologies and frameworks. The final implementation combines solutions developed over the course of several months into a seamless experience for the user. An overview of the inter-working of the video surveillance module is given below.

The video module uses web cameras, which are converted to a basic IP camera developed by the team. The Orbit AF is used by the team for the main video stream and another generic web cam is used as a secondary camera. Each of these cameras are connected to a computer, or laptop, through USB. The team built a Windows Service called WebCamService to interface with the camera through the Windows COM API and host the video stream on a specified port. Every camera has to be connected to a computer running this service. The Jarvis application, running on the Surface, consumes the stream directly from the socket, by fetching individual frames and displaying those in an image control. In this way, the application is able to show 10 to 15 frames per second, which makes a viewable, albeit a little choppy, video stream.

The video module also implements pan and tilt controls for the Orbit AF camera. The Orbit has mechanical pan and tilt. The pan capability is over a 180 degree-range and tilt is over a 140-degree range. The camera control is implemented in two parts. The first part is a DLL that enumerates the devices and issues movement commands to the camera through the IAMCameraControl COM interface. The second part is the ASP Web Service that abstracts the DLL functions and exposes directional controls as web service methods. The Jarvis application consumes the ASP Web Service and calls the method corresponding to the direction wanted (tilt up, tilt down, pan left, pan right). The IIS Web Server translates those method calls into method calls that the DLL provides, which ultimately move the Orbit AF.

The surveillance video screen supports up to three different camera streams at once, which are configurable through the admin screen. It also provides a still capture feature.

4.2 Audio Implementation

The audio implementation is handled by a speech recognition object that is instantiated on the background page of the application. This image pulls speech recognition information from a database when it is created, so it know to recognize specific phrases specified in a database. This phrase information is pulled together in Grammars, which are attached to the speech recognition object.

Speech recognition information is stored in a hierarchical database so the user knows which commands are invoked by phrases in the system The following image describes the structure of speech information:

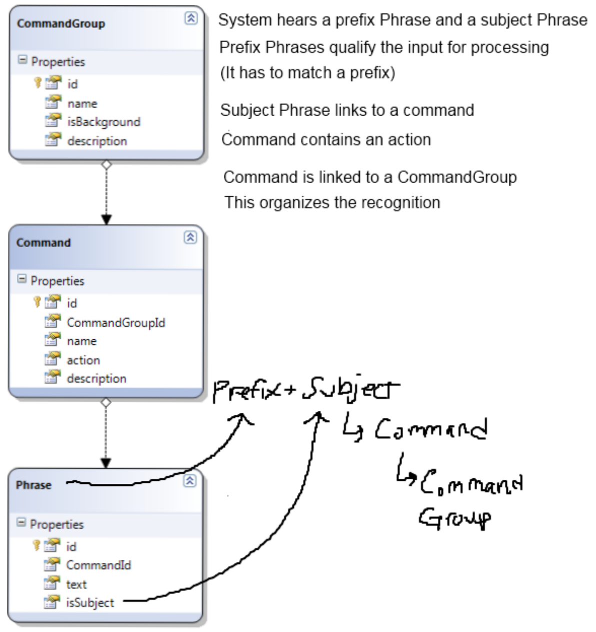
CommandGroup
Properties
id
name
isBackground
description
Command
Properties
id
CommandGroupId
name
action
description
Phrase
Properties
id
CommandId
text
isSubject
System hears a prefix Phrase and a subject Phrase
Prefix Phrases qualify the input for processing
(It has to match a prefix)
Subject Phrase links to a command
Command contains an action
Command is linked to a CommandGroup
This organizes the recognition
Prefix + Subject
↳ Command
↳ Command Group

Figure 4: Structure of the speech recognition database tables

The speech recognition feature is set up to recognize a prefix phrase and a command phrase. The speech recognition object on the background screen will respond to the combination of the prefix and command phrases. To be more extensible, many different prefix phrases can be paired with the same command phrase that will carry-out the same action.

4.3 Media Center Implementation

The Media Center consists of four primary actions: getting content-specific information from Netflix, displaying the content in a Surface LibraryBar, supporting drag-and-drop functionality between the LibraryBar and the home screen, and finally displaying content on the TV if items are dropped onto the TV icon.

First, movie information is retrieved using a service reference to the Netflix Data Catalog web service. Image URLs are retrieved and stored in the application. Then, a custom Thumbnail class was used to add both videos and images to a LibraryBar for a consistency within the LibraryBar. After that the ScatterView was switched to a DragDropScatterView class that was provided in the Surface SDK examples. By default, the Surface SDK provides drag and drop between multiple LibraryBars, but not between a LibraryBar and ScatterView. the DragDropScatter-View provides this functionality and enables features such as displaying content on the TV.

To display images and video on the TV, the center of the dropped item is detected. If it is within the coordinates of the TV icon, that item is removed from the DragDropScatterView, displayed on the TV, and the appropriate actions are displayed on the remote control. For images, this is only ”Eject” but for videos, ”Play/Pause”, video location, and ”Eject” are presented as options. Upon ejecting an item from the TV, it is added back to the DragDropScatterView and then the TV is available for a new item.

4.4 Remote Implementation

The remote is implemented as an object in the Scatter-view container that serves as the background of the application. Based on what the user selects, different information is displayed in the remote object. Since the object is a Scatter-view object, it can be manipulated through user input. The size of the remote is frozen to give the user a more consistent view of the remote interface.

Remote codes were programmed using a separate program. The development team captured all of the remote codes that correspond to each button of the remotes of the connected devices. These codes are stored in a remote database that the application has access to. The storage of the remote codes was implemented before the actual remote interface. This ensured that the remote would function correctly.

4.5 Weather Implementation

When the weather interface is loaded, the application calls a remote web-service provided by the Weather Channel. This information is used to build the weather interface. This was the first feature developed on the system.

5 TESTING

The development team used many different methods of testing the various components of the Jarvis Home Automation system. Components of the system were individually tested heavily; later, integration testing was done on the Surface after satisfying individual testing. Testing procedures are detailed in the following sections: *Video Testing, Audio Testing, Media Center Testing, Remote Testing, and Integration Test.*

5.1 Video Testing

The video component of the Jarvis system enabled the user to view surveillance cameras set up on the property from the Surface unit and capture images from those cameras. If the camera has mechanical pan and tilt controls, the user can also control those using the Jarvis system. The video integration was highly involved and required testing on several different levels. It also involved network programming which is prone to errors.

5.1.1 Testing for Windows Service

The Windows Service on the computer attached to the camera was in charge of getting input from the camera and making it available to the internet via a socket. This piece of the system leveraged the open source library DirectShowNet. The Windows service was tested first with a stand alone video client before a custom video client was built into the Jarvis application. In testing this service, it was proven to be very reliable. The team only ran into issues with the firewall blocking the necessary ports, which was easily resolved.

5.1.2 Testing for Video Client

The video client was built into the Jarvis application. It would read the video stream when given the network address and port number (both were configurable through the admin screen). The video client was tested on a private network by configuring the network addresses of different available cameras. The video client had to deal with network lag and loss of a network connection. After testing revealed these problems, the team implemented a network timeout and an error message to relay the failure information to the user.

5.1.3 Testing for Pan/Tilt Control

The pan and tilt control was implemented completely differently than the video stream. The computer hosting the camera was running an IIS Web Service which would receive commands to pan or tilt the camera in any of the four directions. The web server would process the service call and then call into the Logitech Orbit PTZ camera API to move the camera by five degrees in the desired direction. The team tested this in two steps. First, the call to the camera API was developed and tested with a stand alone program. The guts of that program eventually became a DLL. Next, the web service was developed to translate web requests into the corresponding methods provided by the DLL. The web service was tested by implementing it directly into the Jarvis system and experienced very few problems except response time. The team decided that a relatively long response time was acceptable because the camera would be stationary most of the time.

5.2 Media Center Testing

The primary control behind the media center component is the WPF Media Element, which was tested in external applications before its integration into the application. This testing helped the team through several issues, like having to install the correct codecs for video playback and the difficulty of the progress bar feature. Writing the quick, external application made the integration of the Media Element much smoother.

Once the media center was integrated into the Surface application, additional testing was needed to ensure that the Surface could play media. The Surface required additional codecs to play the media that the team was using to test the application. Additional research was needed to display media on a secondary monitor.

The development team planned on developing additional media center controls to increase the speed of video playback. The testing media was not standard, so the team could not implement additional control features. This testing was done late in the project, so the feature could have benefit from additional testing.

5.3 Remote Testing

The remote was tested to ensure that it could properly communicate with both the database and the electronic devices via the USB-UIRT. Every button on the remote is linked to a corresponding database entry. To ensure the remote was functioning correctly, all of the buttons were initialized to a null state. After capturing an infrared code with the USB-UIRT, the database was referenced to check for a valid entry. The team also programmed all of the remotes for devices that were available in the household. To test functionality, the team used the remote on these devices to determine if it could be used in a natural environment.

The team tested the universal remote incrementally by using a smaller, more portable TV. This allowed the team to discover problems in the application. Problems were the mismatching of remote codes in the database, the speed that the application can send codes and the sending of the same codes more than once. Checking the codes sent against the codes in the database allowed the team to eliminate duplicate or incorrectly matched codes. The team was not satisfied with the speed that the application could send codes to the TV, so performance was improved by additional refactoring of the system. The team analyzed calls to the universal remote DLL and reduced the number of calls made. This improved the performance greatly. The final issue discovered in testing of the remote still remains in the system. The universal remote cannot send the same signal consecutively. For example, sending the remote code for ”2” twice in a row will not function correctly. The system will only send one instance of the ”2” command.

5.4 Integration Testing

Integration testing happened in the later stages of the development of the Jarvis Home Automation system. The primary form of integration testing was completed on the Microsoft Surface unit. Primarily, the team was testing for issues that arose when switching to different components in the system. For example, issues with loading time plagued the video screen for awhile. The loading timeout when entering the video interface was too high, and the system would appear unresponsive and frozen.

Some integration testing was completed using the Surface Simulator, for example, the video surveillance screen was tested almost completely using the simulator. This made the final integration of the feature much easier because the team was able to iron out issues before final testing on the Surface.

Other integration testing was completed before the e-week demonstration. It was assumed that the system would be connected to a wireless network that is shared with computers that are streaming video for video surveillance. When the surface and the video providers are connected to different networks, error can occur. This is something that the team had to discover over time, because there was not more than one wireless network available.

The team had hoped to completed more integration earlier in the project but was not able to test on the Surface because of hardware issues with the Microsoft Surface unit.

6 Results (pictures, test results, performance results)

Overall, the development team feels that the project was a success. Each area of implementation experienced varying levels success. The results of the project is analyzed in the following sections: *Video Results, Audio Results, Media Center Results, and Universal Remote Results.*

6.1 Video Results

The team was able to completely implement video surveillance as was originally envisioned. The video screen provides up to three different video feeds, provides camera control for any camera that provides mechanical pan and tilt via the IAMCameraControl COM interface. It also provides the ability to capture pictures from the camera.

The performance of the video module was relatively poor. It was decided that performance in this module was a secondary concern. This is especially true for the performance of the the camera control, because it would be used infrequently. The camera control took 8 to 10 seconds the first time you moved the camera and about 1.5 seconds for subsequent adjustments. The performance of the actual video stream was also a bit slow. The video framerate is approximately 10 to 15 frames per second. The team did not conclusively determine where the bottleneck occurs in the system, but it is suspected to be primarily due to network lag.

6.2 Audio Results

The team was able to completely implement the voice navigation feature. Users have the ability to navigate to different parts of the system by saying a command phrase selected from a set of customizable commands. Commands are stored in a remote database and can be changed without having to recompile the application. The team feels that the initial voice recognition portion of the project was successful, but should have been expanded to other parts of the system.

Compared to the intended scope of implementation, the actual level of implementation of the voice feature was not successful. The team planned to integrate voice search with outside data sources, but was not able to do so. The main hindrance to the wider development of the audio recognition feature was the complexity of the voice recognition implementation. The wide scope of the project also diverted focus away from the voice recognition feature.

6.3 Universal Remote Results

The team was able to fully implement the remote feature. Users have the ability to control multiple infrared devices in the house. These devices included a TV, a cable box, an Xbox 360, and a receiver. Infrared codes are read by the USB-UIRT and stored in a database. These codes were then transmitted through the USB-UIRT to control devices. The database entries could be overwritten with a new code by holding a button on the remote instead of clicking it.

Compared to the intended scope of the remote, more was achieved than initially planned. The team initially planned to use the remote to just help control the television. Due to the success of this remote, more remotes were developed and implemented by the team.

6.4 Media Center Results

Similar to the universal remote, more was achieved than initially desired in the Media Center module. The initial intent was to take digital media and display it on the Surface. On top of that, movie information is acquired from the Netflix Data Catalog web service, the media items are grouped by type in the LibraryBar, and drag and drop functionality was added with the ScatterView. Near the end of the project, displaying digital content on the TV was also fully realized.

7 COST ANALYSIS

The development team was presented a unique opportunity to work with the Microsoft Surface computing platform at the beginning of the school year. This heavily influenced our decision, which resulted in a solution that lacks practicality because of the high cost of the Microsoft Surface. The following sections are detailed in the cost analysis of the project: *Microsoft Surface, Audio Input Devices, Video Devices, Universal Remote, and Project Presentation Materials.*

7.1 Microsoft Surface

The Microsoft Surface is the most expensive item in this project. At a cost of $15,000, this is not a feasible item to have in your living room. With touch technology making advances and becoming more popular, the team hopes that a cheaper alternative will be available in the near future. [4]

There is currently a guide to build a multi-touch-surface-PC that has a cost of about $350. While the Jarvis system is specifically designed for the Microsoft Surface, the core information is based on touch technology and it could be possible to adapt it to be used on a different multi-touch table. [5]

7.2 Audio Input Devices

The team used an Aliph Jawbone II for audio input on the project. The cost of the headset is $41.99. The team felt that this was a fair price for the benefit of bluetooth connectivity and noise cancellation built into the headset. [1]

7.3 Video Devices

The Logitech Orbit AF QuickCam was the main video device used. The cost of this unit is $112.42. This webcam was the best solution the team could find that had pan and tilt, and could also produce a high definition video stream. [2]

7.4 Universal Remote

The device used to transmit and receive infrared signals was the USB-UIRT. This unit costs $50. [6]

Overall Costs:
Surface - $15,000.00
Jawbone = $41.99
Orbit AF - $112.42
UIRT - $50.00
Total - $15,204.41

8 DIFFICULTIES ENCOUNTERED

The project team faced a variety of difficulties over the development of the Jarvis Home Automation system. The following difficulties are detailed in the following sections: *Microsoft Surface Issues, Development Environment Issues, Scope of the Project, and Video Design.*

8.1 Microsoft Surface Issues

The Microsoft Surface was the source of many difficulties with the project. The team faced both hardware and software issues.

The first issue faced by the team was the tight constraints for running the SDK and Surface Simulator. The simulator will only work on a 32-bit version of Windows Vista, unless additional configuration was done. Due to varying operating systems being used, not everyone could run the simulator. This made testing functionality difficult.

The team also worked through hardware issues with the Surface. Due to an unknown problem, the touch functionality of the Surface would stop working. The team worked with Microsoft to fix the issue and a replacement part was sent. When the replacement part arrived, it was discovered that the wrong item had been sent. The repairman looked at the Surface anyways and reseated internal components to resolve the issue.

8.2 Development Environment Issues

During the course of the project, the team encountered several issues in setting up the development environment. The issues experiences varied for each team member, depending on the operating system, whether the computer was 64-bit or 32-bit, and screen resolution, among other factors.

The first problems encountered were during the initial setup of the development environment. To develop on the Surface, the Surface SDK must be installed. Some team members had problems getting it to install properly if they were using a 64-bit platform, because it was developed for 32-bit Windows Vista. All teammates also faced the challenge of screen resolution somewhere along the way. In order to run the simulator program, the simulator window must be able to fit entirely on the screen, which meant that the computer had to be larger than the resolution of the Surface. This posed a problem for many team members while they were on laptops. This essentially restricted development to desktop computers.

Lastly, the environment setup went through many changes during development, which were difficult to manage across the team. Oftentimes, an added reference to an outside DLL would be committed without committing the DLL. This caused team members frustration when they updated from Subversion and could no longer build the program. Along the same lines, if the .project file was changed, those changes might either be real additions that need to be committed, or they may be side effects of something the person was toying around with. Since the changes to the .project file are mostly hidden from the user, it was difficult to manage these changes.

8.3 Scope of the Project

The team ran into difficulties defining the boundaries of what would be included in the project and what would not. This is because of the open-ended nature of the project. Many different ideas and pieces of functionality had potential to be incorporated into the application. The team wanted to be as open as possible when considering features, so it became difficult to say no to different parts of the system. This resulted in pieces of the system that were not developed as completely as they could have been.

Future projects should have a clearly defined scope that teams can commit to. The team believes that since the scope was so large, there was no chance of completing everything in the project, which hurt the overall motivation on the project. The new sneakers team would have benefit from a well defined scope from the beginning of the second semester. This would have increased the functionality of the completed system.

8.4 Video Design

The video surveillance module was a challenging aspect to the project. The team encountered difficulties when attempting to gain access to the camera's API in order to get the video stream or adjust the pan and tilt of the Orbit AF. These issues stemmed from a lack of understanding of Microsoft COM and DirectShow. The team was also challenged to find a way to make the video stream available over the network.

The early, failed attempts at solving these problems involved using DirectShow API directly with C++ and then attempting to package that code into a DLL which could be included into our project. These attempts ultimately failed because the team lacked the expertise needed to work directly with COM interfaces. The early, failed attempt at making the camera available over the network involved a Linksys Network Storage Link for USB 2.0 Disk Drive

(NSLU2), which had Linux installed. While the team was able to make some progress on that solution, it was eventually scrapped for a simpler approach.

The eventual solution was found in a library called DirectShowNet. This library wrapped the C++ interfaces needed, into C# interfaces which could be used directly with the Jarvis application. The DirectShowNet library also had dozens of example projects which greatly helped the team get off the ground with the video surveillance.

9 CONCLUSION

The Jarvis Home Automation application allow you to interact with your home in a new fun and useful way. The system provides convenient and intuitive access to information and entertainment that you would otherwise have to search out for yourself. The high level goals of this project was to make a system with a futuristic feel that could improve the home life of its users. The team believes they have accomplished these goals through the Jarvis system. The specific components: weather, video surveillance, universal remote control, and the media library all have immediate application, but also demonstrate that the system is highly flexible and extensible. Due to the high cost of the components, it is not expected that this specific system could be very popular, but instead the team hopes that it can help inspire others to think large about what is possible in the realm of home automation. Many products monitor and reduce energy consumption, which is a valuable endeavor, but far fewer systems aim to improve the home life by bringing together existing technology in creative and useful ways.

REFERENCES

[1] **Amazon.** Aliph Jawbone II Bluetooth Headset with NoiseAssassin (Black) - Bulk Packaging. *Amazon.com.* [Online] [Cited: 05 05, 2010.] http:www.amazon.comAliph-Jawbone-Bluetooth-Headset-NoiseAssassindpB002D9GQKOref=sr_1_1?ie=UTF8&s=electronics&qid=1273027426&sr=8-1.

[2] **Amazon.** Logitech QuickCam Orbit AF Auto Focus System (Black). *Amazon.com.* [Online] [Cited: 05 05, 2010.] http:www.amazon.comLogitech-QuickCam-Orbit-Focus-SystemdpB000UY1OMYref=sr_1_1?ie=UTF8&s=electronics&qid=1272492197&sr=8-1.

[3] **Edward B. Driscoll, Jr.** A Timeline for Home Automation. *eddriscoll.com.* [Online] Yablok & Associates, 2002. [Cited: 04 27, 2010.] http:www.eddriscoll.comtimeline.html.

[4] **Microsoft.** Purchasing Microsoft Surface. *Microsoft.com* [Online] [Cited: 05 05, 2010.] http:www.microsoft.comsurfaceenusPagesHowToBuyHowToBuy.aspx

[5] **Parrish, Kevin.** Build Your Own Multi-touch Surface Computer. *TomsHardware.com.* [Online] [Cited: 05 05, 2010.] http:www.tomshardware.comnewsmulti-touch-surface-PC-DIY,7484.html.

[6] **Rhees, Jon.** USB-UIRT. *USB-UIRT Home.*[Online] http:www.usbuirt.com.