Center for Embedded Computer Systems
University of California, Irvine

Iris – A Home Control System with an Implemented Home Improvement Application

Davit Hovhannisyan, Arno Abramyan

Derek Nham, Barry Thach

Center for Embedded Computer Systems

University of California, Irvine

Irvine, CA 92697-2620, USA

{dhovhann, aabramya, dnham, bthach} @uci.edu

CECS Technical Report 12-06

July 27, 2012

Table of Contents

List of Figures

Abstract

Project Iris explored the integration of different I/O peripherals—such as cameras, microphones, projectors and speakers—into a single compact unit based on a BeagleBoard. To demonstrate the integration, a stud detector system was implemented. The Iris unit was programmed to detect red light coming from a traditional stud finder and project the studs on the wall. The unit could be started manually or with a voice command. The user communicates via the voice recognition unit and initializes stud locations using the stud finder; these are later used to calculate projection coordinates independently from the distance of the device from the wall. Furthermore, the projector displays vertical lines on the wall, which resembles studs and allows the user to have an advantage over traditional mounting tools/applications. The project demonstrated the ability of the BeagleBoard to coordinate the separate devices and opened the door for numerous applications to be created on the platform.

Background/Problem

Many different home control systems have been envisioned and implemented throughout the past century. Some of these solution controlled home entertainment, security and basic control systems. Some of the most common implementations are television, radio, and projector, garage, and air conditioner controllers. On the other hand the sci-fi community offered other solutions that envisioned complete artificial intelligence systems which would be controlled by humans. Such systems would have the ability to change the course of daily lives; however, no known plausible solution has ever been proposed to implement a complete solution for all types of needs in the modern daily life.

On the other hand it is understandable why different solution have been implemented and no centralized universal system has been offered. Problems are associated with complexity of dealing with all possible structures. In fact just making a universal TV control is already a major difficulty. Moreover, making a simple control system that is able to make intelligent decisions universal to all users is on its own is close to impossible.

However, after careful analysis of existing implementation of one of the most common telecommunication system, the Apple IPhone revealed that it is possible to make a universal solution where users themselves are able to add their desired futures. This ability to extend the use of hardware opens new doors for software. This ability to add features as time goes on makes products easier to manufacture, hence, reasonable to engineer.

Vision

One way to engineer a feasible universal home control system is to make it as a platform that has capability to be connected to all known everyday devices and use available information from these devices to make intelligent decisions. Also, it shall allow external addition of software and addition of new and unknown peripherals.

Goal

To build an intelligent machine with numerous input and output devices which will possess a high degree of flexibility and adaptability for many possible uses. Vision is to create a system with the ability to see, hear, speak, and show. Once a base is complete first application will integrate its sight and projection capabilities to project light over the studs in a wall, making mounting your television onto the wall a lot easier. Of course this is a very specific application has a very small market, but Iris will be able to engage larger markets with the right software. The purpose of this application is to display the capabilities of the device. The purpose of the device is to eventually become a home based stationary assistant, doing things such as helping kids memorize multiplication tables, turning lights on and off, and much more all with vocal instructions.

Impact and Constraints

Economic

Environmental impact shall be viewed from different perspectives. It will be replace numerous products and stimulate the creation of better way to protect the environment. For instance a simple application that monitors electricity will decrease fire hazard potential. The product itself possess no harm to the environment, it has minimal power use.

Manufacturability and Sustainability

Device has the ability to easily replace its parts, hence allowing ease of maintenance. Also once different components are manufactured separately user driven design can be applied to have one sustainable product that won't need to be replaced but only added and extended with futures.

Environmental

Environmental impact shall be viewed from different perspectives. It will be replace numerous products and will allow creation of more secure ways for reductions of environmental harm by human factors. For instance a simple application that monitors electricity will decrease fire hazard potential. The product itself posses no harm to the environment, it has minimal power use.

Health and Safety

As modern day medical components are becoming increasingly interconnected, Iris will eventually become a platform or mediator of medical implants and other devices that monitor the health of local patients. Iris itself will not pose any health or safety risks, because it will be manufactured from safe materials, with the utmost quality.

Social, Political and Ethical

Having an all watching eye could raise a question of personal privacy and the political issues related to it. A production version of Iris will have necessary security measures to ensure the information gathered and available to Iris is kept private. The ethical issues in terms of user privacy are non-existent, because Iris will not share any data it collects with the users' specific permission or command. Specific application which would deal with life and death situations should be fail proof.

Individuals, Organizations and Society

Impact on these aspects will allow people to be better connected with their families and homes. Companies in the business construction might need to adapt some of their techniques to build homes with better technology. Society itself may be resistant to a shift towards a technologically advanced home.

Design

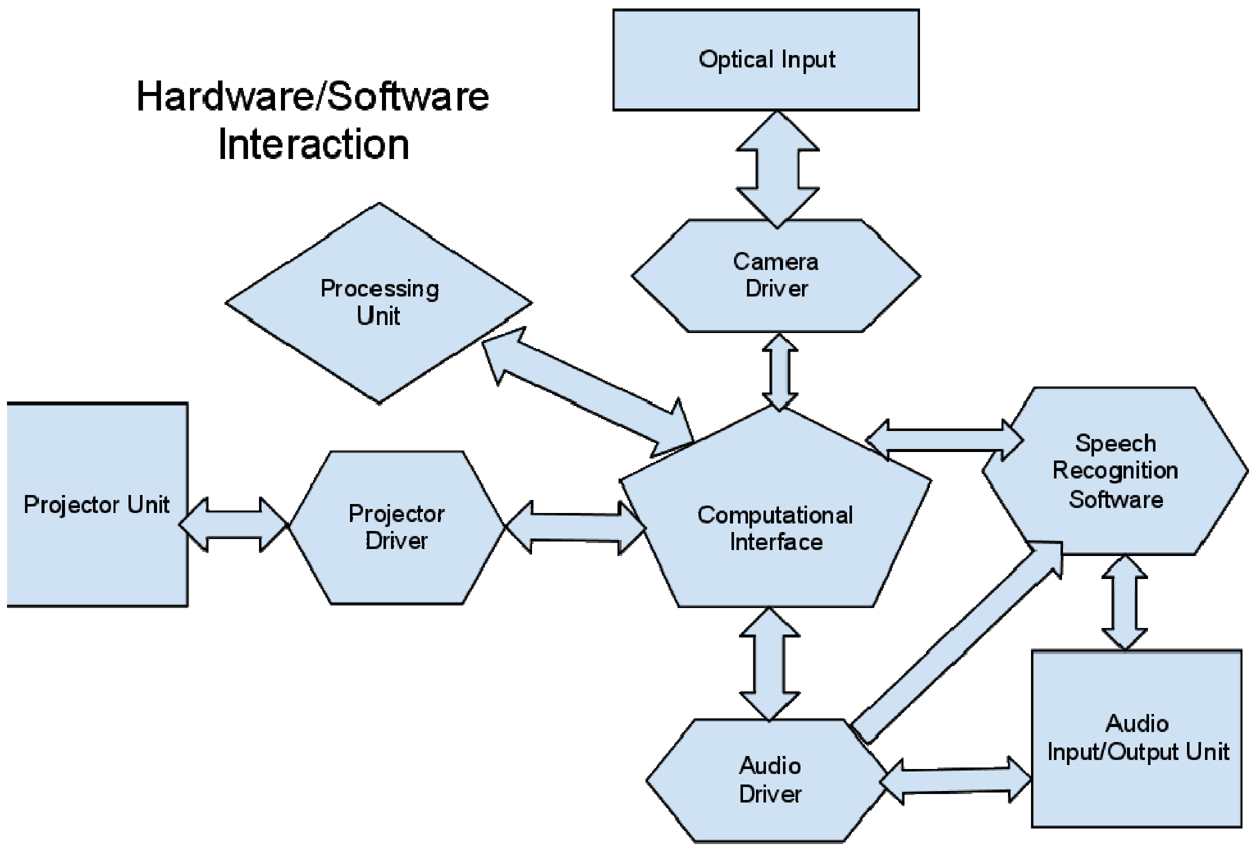


Figure 1 – Overview of Design

Devices Explained

- Optical Input Unit
    - Definition
        - A camera
    - Function
        - It will take in visual data, such as the coordinates of the studs
    - Testing
        - After calibration is done, it should project out the correct coordinates of studs, if not, recalibrate.
- Camera Driver
    - Definition
        - The driver will tell the operating system how to work with the camera hardware.
    - Function
        - It will take the data from the camera and input it to the display
    - Testing
        - Check display to see whether or not the camera captures the image
- Projector Unit
    - Definition
        - Its purpose is to project an image onto the wall/screen
    - Function
        - It will display the Linux terminal
        - In our case, it will display the studs on the wall.
    - Testing
        - Connect to beagle board, and see if it projects the image/terminal
- Audio Input/Output Unit
    - Definition
        - Microphone/Speaker
    - Function
        - Commands will be spoken into the microphone and it will send data to the beagle board on what to do
    - Testing
        - Command would be understood if it does the assigned task
- Processing Unit
    - Definition

        - We will be using a BeagleBoard which is a low-power circuit board with a processor, memory, input/output, and other features.
    - Function
        - The BeagleBoard has all the functionalities of a basic computer and it will be the backbone of our design. We will install our input and output devices and get the proper drivers for our hardware on our BeagleBoard.
    - Testing
        - We will test our components with this BeagleBoard after porting the Linux OS successfully.
- Computational Interface
    - Definition
        - An interface with public access methods allowing signals passed and received by drivers defined by these constraints.
        -
    - Function
        - Is the platform, base and shall be allowing public access to the components
    - Testing
        - Written app shall test all the specifications as well as pass on test parameters for other components
- Speech Recognition Software Unit
    - Definition
        - EasyVR Module which has a built-in speaker with simple recordable memory slots, microphone, and built-in voice recognition for 32 set patterns.
    - Function
        - EasyVR can be used with any host with an UART interface powered at 3.3V-5V.
        - Ideal for home applications such as voice-controlled light switches and locks.
        - Can playback up to 9 minutes of recorded audio or speech.
    - Testing
        - EasyVR Module has GUI tools for voice recognition training and testing.
- Audio Driver
    - Definition
        - It allows for the microphone to be used with the beagle board.
    - Function
        - Stores the audio from the microphone.
    - Testing
        - If microphone does not work, must re-install/re-write driver

Components

Figure 2 - BeagleBoard xM [1]

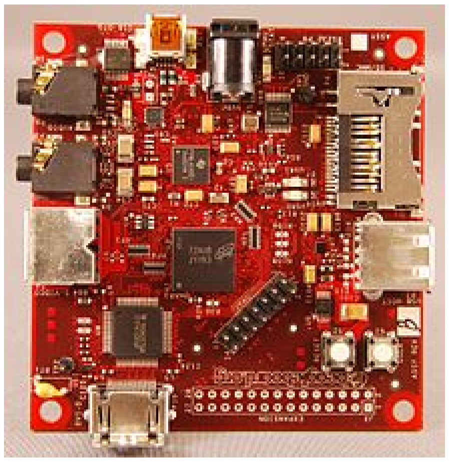
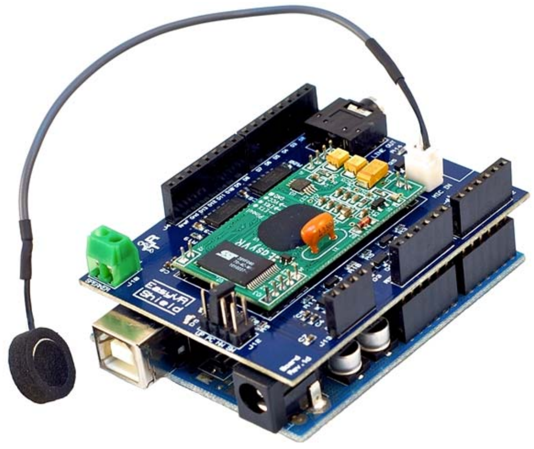


Figure 3 - TI Pico Projector & Power Supply [4]

Figure 4 - Leopard Imaging Camera [3]

Figure 5 - Arduino Voice Recognition Unit [2, 5]



Cost & Funding

Projected Cost

Item	Quantity	Price/item	Total
Beagle Board xM	1	$150	$150
Pico Projector (Acer C110 Portable Pico Projector)	2	$200	$400
Web Camera (Logitech 720p Webcam Pro 9000)	1	$51	$51
Microphone and Speaker (Easy VR Module)	1	$58	$58
Power Supply	1	$20	$20
Cables	*	$30(total)	$30
Paper and Supplies (Used for Symposium)	**	$150	$150
Sales Tax		8.75%	$76
Total			$935

Actual Cost

Item	Quantity	Price/item	Total
Beagle Board xM	1	$150	$150
Pico Projector (TI DLP)	1	$400	$400
Web Camera (Leopard Imaging 3M)	1	$51	$51
Arduino Board	2	$30	$60
EasyVR	1	$50	$50
Power Supply	1	$20	$20
Cables	*	*	$100
Paper and Supplies (Used for Symposium)	**	$150	$150
Sales Tax		8.75%	$89
Total			$1075

Software Development

Operating System

The BeagleBoard-xM traditionally uses either a Linux OS or an Android OS. Originally the Android OS was appealing, with a goal of integrating Iris with mobile phones running on Android. After conducting some research and speaking with people who had used the BeagleBoard, it was decided that Linux Angstrom was the ideal operating system to use because of its widespread use, finding information and resources was easier than it would be for the Android.

Installing the OS

Installing Linux on to the Android was not an easy task, first an image had to be built and downloaded. Second a micro SD card, the main memory of BeagleBoard, had to be correctly formatted and partitioned; finally the OS was copied and installed on the card. This step was difficult because of minor changes that were needed to allow the OS to work correctly, being unaware of these changes a great deal of time was spent on this step, involving several rebuilds and reinstalls.

Connecting the Peripherals

Initially we connected a keyboard using one of the BeagleBoard's USB ports, and a monitor using the HDMI port. After this initial setup we were able to connect the Pico projector which ported over without a hitch. The leopard imaging camera directly connects to a special port on the BeagleBoard. The Arduino board was connected via USB which emulated a serial port.

Peripheral Drivers and Software

The Pico projector and keyboard required no additional software; the camera required installing a special package on the OS. The Arduino board required no additional as its only purpose was to receive a signal via the serial emulated USB connection.

Peripheral Testing

The camera and projector were simultaneously tested by looping the input from the camera to project out.

Software

Choosing a Programming Language

The initial language of choice to write applications for our BeagleBoard was C, but after some debate we decided on using Java because of its vast libraries, portability, and user friendliness. The only requirement was installing the java packages onto the Angstrom OS.

Proof of Concept - Developing the Stud Finder Application

Background

A regular stud finder is slid across a wall until it encounters a stud in the wall, when it does a red LED on the devices turns on. The stud finder application needed to capture video and convert it to images using the camera, and then it had to go through the images to find where the red light appeared. Finally it had to build an image to project the studs on the wall. A simple diagram is of captured images and output is displayed in Figure 6.

How the software works

Image capture

The program starts out by running a command in the OS shell, which starts the camera on capturing and saving images. See figure (insert figure number here) for command. The camera captured images for 10 seconds, with 5 images per second.

Image analysis

The captured images are analyzed to find the location of the red light; images with red in them are put in a list. There needs to be two different locations which have a red light, so images which have red within a close proximity are grouped together and a X coordinate for the two groups of red lights.

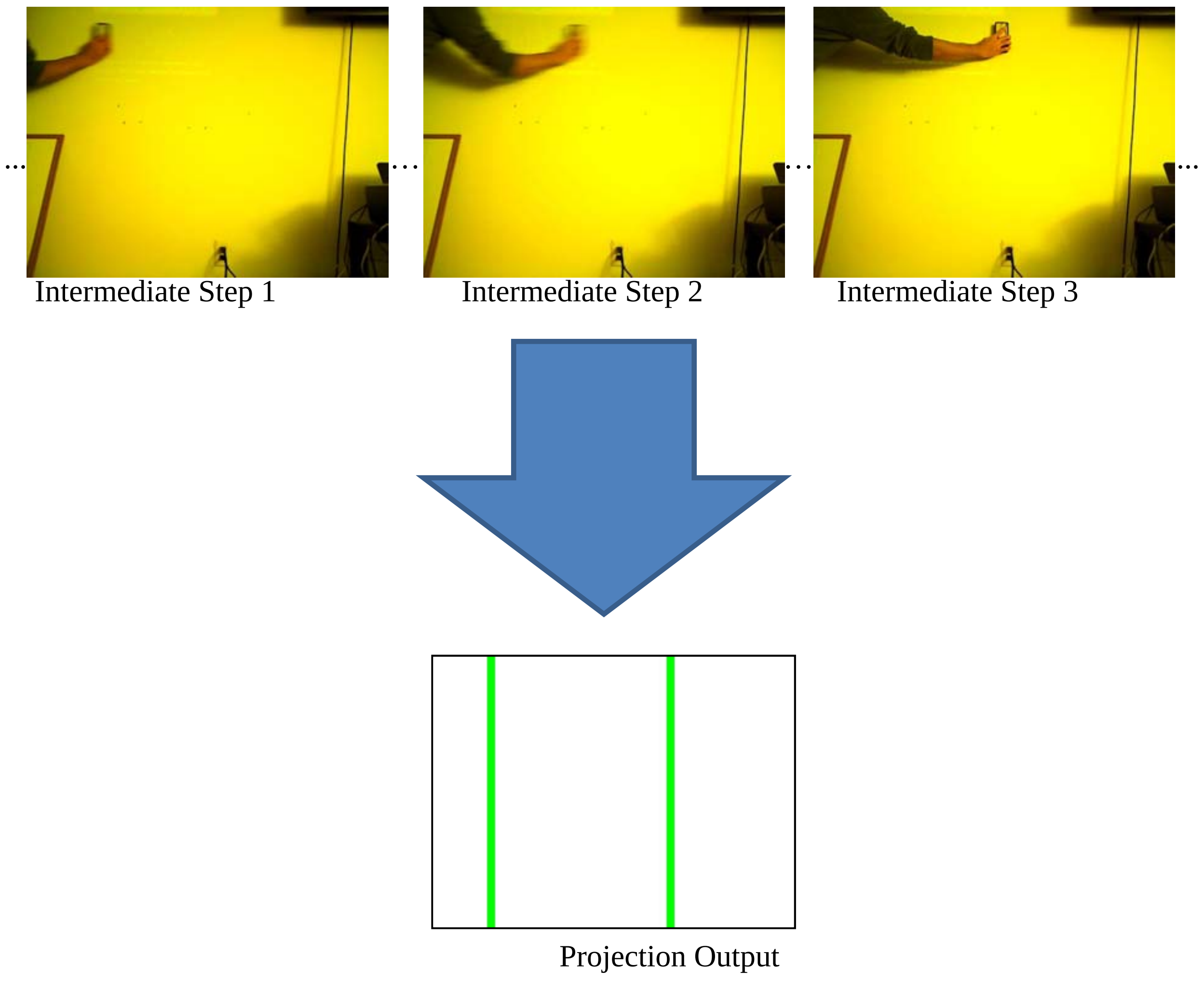
Output Image drawing

Using the two X coordinates provided by the Image Analysis, calculations are preformed on these coordinates to convert them to appropriate coordinates for the projector. This is necessary because location (0,0) on the camera is not the same as location (0,0). After the conversion a basic image is built which creates green beams 10 to 15 pixels wide spanning the length of the image.

Code Testing

The individual methods in our program were tested separately; after the individual components of the program were tested they were integrated and retested together. The conversion algorithm used required multiple test runs of the application to tweak and increase precision.

Figure 6 - Input to Output Diagram of Stud Finder Application.



Code Snippets

Classes

```
public class XY {
	public int X;
	public int Y;
	public XY(int xC, int yC){
		super();
		X=xC;
		Y=yC;
	}
	public String toString(){
		return Integer.toString(X) + "   " + Integer.toString(Y);
	}
}
```

```
public class RGB {
	public int RED;
	public int GREEN;
	public int BLUE;
	public RGB(int rED, int gREEN, int bLUE) {
		super();
		RED = rED;
		GREEN = gREEN;
		BLUE = bLUE;
	}
	public String toString(){return "("+RED+","+GREEN+","+BLUE+")";}
}
```

Interface

```
            //waits for serial signal
public static void wait4Signal();

            //returns array list of all file names from dir directory_name
public static ArrayList<String> FileNames(String dir);

            // stores duplicate images (img) with format and name base
public static void storeImages(BufferedImage img, String format, String name, int quantity);

            //generates an image with bars using Arraylist coordinates (list) with accuracy and height
public static BufferedImage genImage(ArrayList<XY> list, int accuracy, int height);

            //sorts list by X coordinates in ascending order
public static ArrayList<XY> sortX(ArrayList<XY> list);

            // returns a list of XY coordinates from images list where rgb was found with delta accuracy
public static ArrayList<XY> findRGBs(ArrayList<BufferedImage> images, RGB rgb, int delta);

            // returns a list of XY coordinates from the image where rgb was found with delta accuracy
public static ArrayList<XY> findRGB(BufferedImage image, RGB rgb, int delta);

            // returns a list of XY coordinates from the image where rgb was found
public static ArrayList<XY> findRGB(BufferedImage image, RGB rgb);

            // returns a list of XY coordinates from all images in the list images rgb was found with delta
            //accuracy
public static ArrayList<XY> findRGBs(ArrayList<BufferedImage> images, RGB rgb, int delta);

            // returns a list of XY coordinates where in image rgb was found with delta accuracy
public static ArrayList<XY> findRGB(BufferedImage image, RGB rgb, int delta);

            // returns a list of XY coordinates where in image rgb was found with delta accuracy
public static ArrayList<XY> findRGB(BufferedImage image, RGB rgb);
```

```
            // returns a list of XY coordinates from all images 1 by 1  found in the directory that match
            //with from names in the list of strings rgb was found with delta accuracy

public static ArrayList<XY> insertImages1by1(ArrayList<String> list,String dir, RGB rgb, int delta);

            // returns a list of XY coordinates from all images found in the directory that match with from
            //names in the list of strings rgb was found with delta accuracy

public static ArrayList<BufferedImage> insertImages(ArrayList<String> list, String dir) ;

            // insert an image from the file name using default directory path

public static BufferedImage insertImage(String str) ;

            ….

            ….

            //capture num number of Images perSecond times in a second

public static void captureImages(int perSecond, int num);

            //encode this number of images in this order

public static void encodeImages(int num, String result);

            //plays an encoded image from the file result

public static void playMovie(String result);
```

Span of these methods allows for complete object oriented java based application implementations. With popularity of java and Android market, it is expected that later implementations of Iris, might also use Android OS and other more sophisticate hardware.

In order to help future application developers, this interface also includes all standard Java libraries and numerous other not listed methods. However, it is important to mention that last three methods above have unique platform dependent solutions, which need to be modified with revisions or change of OS.

Dependent Method Implementations

```
    public static void captureImages(int perSecond, int num){

            String cmd = "mplayer -vf screenshot -fps "+ Integer.toString(perSecond)+ " tv:// -tv
driver=v4l2:device=/dev/video0 -frames "+ Integer.toString(num)+ " -vo jpeg";

            Runtime run = Runtime.getRuntime() ;

            try{

            Process pr1 = run.exec(cmd) ;

            pr1.waitFor();

            }catch(Exception e){

                    System.out.println("Exception thrown from Image capture"+ e.getCause());

            }

    }

    public static void encodeImages(int num, String result){

            String cmd = "ffmpeg -r " + Integer.toString(num) + " -b 1800 -i %05d.jpg " + result;

            Runtime run = Runtime.getRuntime();

            try{

            Process pr2 = run.exec(cmd);

            pr2.waitFor();

            }catch(Exception e){

                    System.out.println("Exception thrown from Image capture"+ e.getCause());

            }

    }
    public static void playMovie(String result){
            String cmd = "mplayer -vo fbdev " + result;
            Runtime run = Runtime.getRuntime() ;
            try{
            Process pr3 = run.exec(cmd) ;
            pr3.waitFor();
            }catch(Exception e){
                        System.out.println("Exception thrown from Image capture"+ e.getCause());
            }
}
```

These methods explicitly use Angstrom OS dependent instructions. It is envisioned that in a future revisions of this product only these methods will need to be rewritten to accommodate architecture. Hence, applications written for this implementation of Iris are platform independent as far as manufacturer provides up to date interface implementations.

Development plan

Task #	Task Name	Duration	Start	Finish
1	Acquiring Beagle Board	7 days	Mon 11/7/11	Tue 11/15/11
2	Installing Linux OS	4 days	Tue 11/15/11	Fri 11/18/11
3	Looking for Components and Appropriate Drivers	12 days	Tue 11/8/11	Thu 11/24/11
4	Ordering Necessary Sensors and IO Components	10 days	Thu 11/17/11	Wed 11/30/11
5	Independently looking Speech Recognition appropriate open source software.	11 days	Mon 11/7/11	Mon 11/21/11
6	Looking for Motion Detection Algorithms and Practices with Webcams	30 days	Mon 11/7/11	Fri 12/16/11
7	Synchronization algorithms for multiple projectors.	30 days	Mon 11/7/11	Fri 12/16/11
8	Identifying Solution for Applied Sensor Applications	25 days	Mon 12/19/11	Fri 1/20/12
9	Implementation of all necessary outlined software	27 days	Wed 12/7/11	Thu 1/12/12
10	Testing Debugging	31 days	Fri 1/6/12	Fri 2/17/12
11	Possible Application documentation	46 days	Mon 12/19/11	Mon 2/20/12
12	Presentation Board Making	5 days	Mon 1/16/12	Fri 1/20/12
13	Testing Presentation for Iris	24 days	Mon 1/23/12	Fri 2/24/12
14	Documentation	79 days	Mon 11/14/11	Thu 3/1/12

Deliverables

- Platform consisting of Camera, Microphone, Speakers, Beagle Board, that is running on a Linux Angstrom Operating System, and a Pico-Projector.
- Interface with implemented methods that allow intended functionality
- Drivers that allow the Interface to be performing its tasks.
- Symposium tool that supports applications demonstration
- A report that includes final and complete design characteristics and specifications

Acknowledgements

The authors would like to thank Dr. Fadi Kurdahi for his supervision, also Kiarash Amiri, Shih-Hsien Yang. This material is based upon work supported by the Undergraduate Research Opportunity Program of University of California, Irvine.

References

[1]. BeagleBoard xM: http://www.beagleboard.org/

[2]. Arduino: http://www.arduino.cc/

[3]. Camera: https://www.leopardimaging.com/uploads/LI_LBCM3M1_Camera_Board.pdf

[4]. Pico Projector: http://dlp.com/pico-projector/

[5]. EasyVr: http://www.veear.eu/products/easyvr/