중간 보고
5.2 프로젝트 문제 사항........................................................................................................18
1 Introduction

1.1 Purpose

증강 현실 기술을 통해 여러가지 위치기반 서비스들이 제공되고 있다. 유비쿼터스 환경에 대한 연구를 통해 여러 가전기기들이 하나의 기기로 컨트롤할 수 있는 환경이 구성되고 있다. 이러한 환경에서 멀티컨트롤 디바이스 플랫폼으로 대두되고 있는 안드로이드와 유비쿼터스의 기초적인 임베디드 환경인 적외선 통신을 결합하여 사용자에게 편리한 원격 제어 환경을 구축함은 물론 증강 현실 기술을 접목함으로서 User Friendly 환경을 구축하여 일반 사용자들도 안드로이드는 물론 유비쿼터스 환경의 필요성을 실감하게 하는 것이 본 프로젝트의 목적으로 한다.

1.2 Scope

1.2.1 Product Name

본 프로젝트의 이름은 AR Remote System으로 결정한다. 이는 증강현실과 기존 가전 원격 제어 환경의 이름을 결합한 것으로 사용자의 호기심을 불러 일으킴과 동시에 Convergence 환경에 대한 부담감도 줄이는 데 일조할 것이다.

1.2.2 Product Main Functions

본 프로젝트의 주요 기능은 다음과 같다.

1.2.2.1 안드로이드 기반의 모바일 환경에서 가전 디바이스를 인식한다

1.2.2.2 모바일 환경에서 홈 네트워크상의 PC를 통해 미디어 센터를 컨트롤 할 수 있다.

1.2.2.3 모바일 환경에서 직접 혹은 홈 네트워크 상의 PC를 통하여 비 네트워크 기기인 TV의 기본적인 제어가 가능하다.

1.2.3 Relevant benefits

본 프로젝트를 통해 안드로이드 소켓에 대한 응용 분야에 대한 파악이 가능하다. 또한 차후 안드로이드 스펙에 포함될 적외선 통신에 대한 분야가 확장된다면 별도의 윈도우 어플리케이션을 제외하고 안드로이드에서 직접 적외선 기반 기기를 제어 할 수 있는 환경을 구축 할 수 있다. 증강 현실 부문에서도 실생활 기기들을 직접 인식 함으로서 제어뿐만 아니라 유비쿼터스 감시 분야에서도 활용할 수 있다.
1.3 Terms of Use

1.3.1 AR : 증강현실을 지칭한다. 가상현실의 한 분야로 실제 환경에 가상 사물이나 정보를 합성하여 원래의 환경에 존재하는 사물처럼 보이도록 하는 컴퓨팅 기법이다. 이 문서에서 증강현실이라 함은 카메라 이미지를 특정 알고리즘에 입력하여 원하는 결과 값을 출력해주는 모듈을 의미한다.

1.3.2 UI : 안드로이드 어플리케이션의 UI를 지칭한다. 원도우 어플리케이션의 경우 UI에서 제공하는 기능이 미미 하므로 안드로이드 어플리케이션의 UI만을 UI라 지칭한다.

1.3.3 OpenCV : 오픈소스 컴퓨터 비전 C 라이브러리이다. 원도우 리눅스 등의 여러 플랫폼을 지원하며 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다. HCI(Human-Computer Interaction)의 한 분야이다.

1.3.4 원격 제어 : 모바일 디바이스에서 목표 디바이스로 신호를 전송하여 목표 디바이스의 무선 제어 환경을 구축하는 것을 의미한다.

1.3.5 아두이노 : 아두이노(Arduino)는 오픈소스를 기반으로 한 물리적 컴퓨팅 플랫폼으로, AVR을 기반으로 한 보드와 소프트웨어 개발을 위한 통합 환경(IDE)을 제공한다. 아두이노는 많은 스위치나 센서로부터 값을 받아들여, LED나 모터와 같은 것들을 통제함으로서 환경과 상호작용이 가능한 물건을 만들어낼 수 있다. 또한 Flash, Processing, Max/MSP와 같은 소프트웨어를 연동할 수 있다.

1.4 Overview

이 문서는 AR Remote System의 전반적인 기능을 포함하고 있다.

2장 Overall Description 에서는 기본적인 프로젝트의 기능, 사용자의 특성 그리고 개발에 관련 제약사항들을 제시하고, 3장에서는 2장의 내용을 기반으로 Functional Spec 과 Non-Functional Spec을 제시 할 것이다.
2 Overall description

2.1 Product Perspective

2.1.1 User interface
카메라 화상을 통해 인식

디바이스 인식부 화면(인식 상대)

디바이스 정보 등록창

디바이스 이름
디바이스 IP
(컴퓨터의 경우 활성화)
비밀번호 (추가 고려중)
비밀번호 확인 (추가 고려중)

디바이스 등록창
2.2 Product functions

2.2.1 앤드로이드 핸드폰에서의 증강현실 기능 제공

2.2.1.1 OpenCV 환경을 앤드로이드에 적용하여 구현 예정

2.2.1.2 다음과 같은 방식으로 이미지 인식을 구성하고 있다.

2.2.1.2.1 카메라 이미지 만으로 디바이스 인식

2.2.1.2.2 미리 저장해 놓은 디바이스 이미지와 카메라 이미지의 매칭으로 디바이스를 인식

2.2.1.2.3 마커를 디바이스에 부착한 후 마커를 인식하여 디바이스를 간접적으로 인식

2.2.2 인식된 컴퓨터 디바이스를 원격으로 컨트롤

2.2.2.1 C# 가상 키보드 방식을 활용하여 키 이벤트를 중으로서 윈도우 미디어 플레이어 뿐만 아니라 코드 수정이 가능하다면 다른 여러가지 PC환경을 컨트
2.2.3 인식된 적외선 디바이스를 원격으로 컨트롤

2.2.3.1 Arduino 혹은 안드로이드 IOIO 보드를 활용하여, 적외선 신호를 적외선 수신 디바이스에 전송하면 기초적인 제어가 가능한 환경이 구축된다. 물론 모바일 디바이스에서의 직접적인 제어가 아니므로 즉각적인 반응을 기대하기는 어렵다. (Performance의 문제)

2.3 User Characteristics

사용자의 예상 특성은 다음과 같다.

2.3.1 기초적인 네트워크 지식을 갖춘 사용자 임을 가정한다. 웹서버를 통해 전송하여 기초적인 네트워크 지식이 없는 사용자를 대상으로 제작 할 수도 있지만, 성능상의 문제와 총업작품의 규모를 생각하여 직접 IP를 검색하고 모바일 디바이스에 설정 할 수 있는 사용자임을 사전에 가정한다.

2.3.2 윈도우 미디어 센터의 기본 기능을 사용 할 수 있는 사용자임을 가정한다. 윈도우 미디어 센터를 실행 시키는 기능은 없으므로 윈도우 미디어 센터의 기본 기능을 사용 할 줄 아는 사용자에 한 해 본 프로젝트의 기능을 모두 활용할 수 있을 것이 다.

2.3.3

2.4 Constraints

2.4.1 인원 : 4명

2.4.2 사용언어 : C#, 안드로이드 Java, HDLC(미정)

2.4.3 구동환경 : 서버 : Window 7 Home // 클라이언트 : 안드로이드 2.3.3(Ginger Bread)

2.4.4 수행 기간 : 2011년 2분기

3 Specific Requirements

3.1 External interface requirement

3.1.1 User Interfaces
3.1.1.1 안드로이드 어플리케이션은 GUI 환경을 제공하고 윈도우 어플리케이션은 최대한 Background Process의 형태로 제공한다.

3.1.1.2 설정 창과 리모트 컨트롤 창을 분리하여 최대한 카메라 화면 상에서 디바이스를 컨트롤 한다는 느낌을 제공 하도록 한다.

3.1.1.3 리모트 컨트롤 제공의 범위는 기초적인 전원, 볼륨, 채널 등에 한정한다. 세부적인 컨트롤의 경우 적용되지 않는 디바이스가 존재 할 수 있기 때문이다.

3.2 Functional Requirement

3.2.1 실행

3.2.1.1 클라이언트는 안드로이드 apk파일의 형태로 제공한다.

3.2.1.2 중간 보고서 상의 실행은 고정된 IP를 통해 접속을 하도록 한다.

3.2.1.3 디바이스 인식은 아직 개발중이므로 디바이스를 인식 하였다는 전체 하에 실행 후 바로 리모콘 형태의 버튼을 제공하도록 한다.

3.2.1.4 윈도우 어플리케이션은 C#으로 제공하며 백그라운드 프로세스 형태로 제공 한다.

3.2.2 조작

3.2.2.1 기본적인 조작은 터치 만으로 이루어 진다.

3.2.2.2 디바이스가 인식되면 인식음과 함께 인식 버튼을 표시하고, 인식이 완료되면 해당 디바이스에 대한 컨트롤러가 표시된다.

3.2.2.3 인식된 디바이스가 TV일 경우 서버에서 IR 모듈을 통해 적외선을 송신하는 모드로 전환한다.

3.2.2.4 각 컨트롤러 버튼에 바인딩된 메시지를 컨트롤러 버튼 입력시마다 서버로 전송한다.

3.2.2.5 세션을 종료하는 버튼을 두어 인식 종료가 가능하게 한다.

3.2.3 디바이스 등록

3.2.3.1 컴퓨터의 경우 IP를 함께 등록해 준다. 이는 네트워크 기반으로 동작 하기 때문에 물체 인식 만으로는 데이터를 전송 할 수 없기 때문이다.
3.2.3.2 인식 방식에 따라 디바이스의 초기 인식상태 사전이 첨부 될 수도 있다.

3.2.3.3 보안 모듈의 추가 여부에 따라 디바이스마다 비밀번호를 추가하여 인식시 비밀번호를 입력하게 하는 기능도 추가 가능하다.

3.3 Performance Requirement

3.3.1 PC의 경우 무선 인터넷을 지원하여야 한다.

3.3.2 PC의 경우 C# 어플리케이션이 실행 가능한 환경을 갖추었다고 가정한다.

3.3.3 TV의 경우 적외선 제어를 지원 하여야 한다.

3.3.4 반응 속도의 경우 Wi-fi 환경(10Mbps 이상)이 지원되는 환경에서는 1초 이내에서 반응이 이루어져야 한다.

4 Development Progress

4.1 Remote Control

4.1.1 미디어 센터 컨트롤의 경우 가상 키보드에서 작동을 얻어, 직접 윈도우 키 이벤트를 발생시켜 컨트롤 하는 방식을 채택하였다.

4.1.2 적외선 컨트롤의 경우 다음의 방식들이 후보에 오르고 있다.

4.1.2.1 Arduino보드와 서버 PC의 USB 연결로 IR 컨트롤 제공

4.1.2.1.1 Arduino 보드에 IR 모듈을 추가하여 서버 PC에서 IR로 적외선 TV를 제어

4.1.2.2 Arduino보드와 안드로이드 디바이스의 블루투스 연결로 IR 컨트롤 제공

4.1.2.2.1 Arduino 보드에 블루투스 쉐드와 IR 모듈을 추가하여 컨트롤을 제공 하는 방식이다.

4.1.2.3 안드로이드 IOIO보드에 IRED를 추가하여 적외선 모듈로 활용 IR 컨트롤 제공

4.1.3 세가지 방안에 대하여 장단점을 파악해 본 결과는 다음과 같다.
<table>
<thead>
<tr>
<th>방식</th>
<th>Arduino with PC</th>
<th>Arduino with BlueTooth</th>
<th>Android IOIO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros.</strong></td>
<td>기초 예제들이 풍부하여 동작 방식을 제대로 이해하여 구현하기 쉽다.</td>
<td>블루투스 환경이기 때문에 안드로이드 뿐만 아니라 아이폰 등 다른 디바이스와도 연결 방식으로 수정이 가능하다.</td>
<td>안드로이드 OS 차원에서 지원하는 임베디드 보드이기 때문에 네트워크와 무관하게 적외선 모듈을 컨트롤 할 수 있다.</td>
</tr>
<tr>
<td><strong>Cons.</strong></td>
<td>상대적으로 고가인 Arduino 모듈의 장점들 제대로 살리지 못한다.</td>
<td>블루투스 쉴드가 Arduino보다 고가여서 비용이 높아진다.</td>
<td>안드로이드 이외의 디바이스에는 적용이 불가능하다. 아직은 자료가 풍부하지 않아 프로젝트 진행에 부담소로 작용 할 수 있다.</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>5만원선</td>
<td>10만원선</td>
<td>8만원선</td>
</tr>
</tbody>
</table>

4.1.4 미디어 센터 키보드 이벤트 발생 클래스

```c
public class MControl {
    [DllImport("user32.dll")]
    static extern uint keybd_event(byte bVk, byte bScan, int dwFlags, int dwExtraInfo);

    public void kDown() {
        keybd_event((byte)Keys.Down, 0, 0, 0);
    }

    public void kUp() {
        keybd_event((byte)Keys.Up, 0, 0, 0);
    }

    public void kEnter() {
        keybd_event((byte)Keys.Space, 0, 0, 0);
        keybd_event((byte)Keys.Space, 0, 2, 0);
    }

    public void kVUp() {
        keybd_event((byte)Keys.VolumeUp, 0, 0, 0);
        keybd_event((byte)Keys.VolumeUp, 0, 2, 0);
    }

    public void kVDown() {
        keybd_event((byte)Keys.VolumeDown, 0, 0, 0);
        keybd_event((byte)Keys.VolumeDown, 0, 2, 0);
    }
}
```
```java
public void kPlay()
{
    keybd_event((byte)Keys.MediaPlayPause, 0, 0, 0);
}

public void kNTrack()
{
    keybd_event((byte)Keys.MediaNextTrack, 0, 0, 0);
}

public void kPTrack()
{
    keybd_event((byte)Keys.MediaPreviousTrack, 0, 0, 0);
}
```

코드에서 보면 알 수 있듯이 User32.dll 파일을 임포트하여 시스템 함수를 직접 호출하여 이벤트를 발생 시킨다.

## 4.2 Connection Between Android and PC

### 4.2.1 앱드로이드와 서버PC의 연결에는 TCP/IP 소켓 통신을 사용하였다. 보통 앱드로이드에서는 http방식을 사용하여 통신하는 예제가 많았다. 그렇지만 http방식을 사용하기 위해서는 별도의 Apache 서버가 필요할 뿐만 아니라 만족할만한 퍼포먼스 (접속시 4초, 메시지 전송시 3초(10Mbps 환경)가 소요되었다.)도 보여주지 못하였기 때문에 소켓 통신을 채택하였다.

4.2.2 앱드로이드와 C# 서버간의 소켓 통신은 앱드로이드 별도 제공 API가 아닌 자바 API를 통해 TCP/IP 소켓 통신을 활용 하여야 했다. 다음은 앱드로이드 클라이언트의 접속 부분 소스이다.

```java
public void connect(String server, int port, String user) {
    try {
        cSocket = new Socket();
        SocketAddress remoteAddr = new InetSocketAddress(server, port);
        cSocket.connect(remoteAddr, 3000);
        streamOut = new PrintWriter(cSocket.getOutputStream(), true);

        // 출력용
        streamIn = new BufferedReader(new InputStreamReader(cSocket.getInputStream(), "UTF-8"); // 입력용 스트림
        sendMessage("Connected.");
    }
```
} catch (Exception ex) {
    logger("접속이 제대로 이루어 지지 않았습니다.");
}

4.2.3 안드로이드 테스트용 클라이언트

4.2.4 C# 서버는 SocketListener를 활용하여 들어오는 문자열을 감시하는 형태로 작성하였다. 다음은 C# 서버의 Listener 파트 코드이다.

```csharp
private void cmdListen_Click(object sender, System.EventArgs e)
{
    try
    {
        //create the listening socket...
        m_socListener = new
        Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        IPEndPoint iLocal = new IPEndPoint ( IPAddress.Any, 8221);
        //bind to local IP Address...
        m_socListener.Bind( iLocal );
        //start listening...
        m_socListener.Listen (4);
        // create the call back for any client connections...
        m_socListener.BeginAccept(new AsyncCallback
        ( OnClientConnect ),null);
        cmdListen.Enabled = false;
    }
    catch (Exception ex) {
        logger("접속이 제대로 이루어 지지 않았습니다.");
    }
}
```
catch (SocketException se)
{
    MessageBox.Show ( se.Message );
}

public void OnClientConnect(IAsyncResult asyn)
{
    try
    {
        m_socWorker = m_socListener.EndAccept (asyn);

        WaitForData(m_socWorker);

        m_socListener.BeginAccept (new AsyncCallback(OnClientConnect), null);
    }
    catch (ObjectDisposedException)
    {
        System.Diagnostics.Debugger.Log(0,"1","OnClientConnection: Socket has been closed");
    }
    catch (SocketException se)
    {
        MessageBox.Show ( se.Message );
    }
}
4.2.5 안드로이드 클라이언트와 C# 서버간의 통신 결과는 시연으로 보이도록 한다.

4.3 Augmented Reality

4.3.1 OpenCV를 활용한 이미지 프로세싱이 증강현실을 구현하는 데의 핵심이 될 것이 다. 본 장에서는 OpenCV 환경 구축과 그에 대한 간단한 예제 테스트를 통해 안드로이드에 대한 적용과 본 프로젝트에서 진행 방향에 대해 설명하도록 하겠다.

4.3.2 Package Needed for this Project

4.3.2.1 Android SDK

4.3.2.2 Android-NDK – Android CrystaX

4.3.2.3 Cygwin

4.3.2.4 Android-OpenCV Source

4.3.2.5 Eclipse IDE for Java EE User

4.3.3 Install Process

4.3.3.1 Install Android SDK

4.3.3.2 Install Cygwin – Cygwin 설치시 Devel 파트의 하위 항목을 모두 설치 하도록 한다.

4.3.3.3 Install Android NDK

4.3.3.3.1 Download – http://developer.android.com/sdk/ndk/index.html
4.3.3.2 Cygwin Home 설치

4.3.3.3 .bashrc 파일을 열어 다음 명령어 추가

export PATH=$PATH:/home/[Username]/android-ndk-r4-crytast

export ANDROID_NDK_ROOT=/home/[Username]/android-ndk-r4-crytast

4.3.3.4 Install Android-OpenCV Source

4.3.3.4.1 Install Eclipse SVN

4.3.3.4.2 http://android-opencv.googlecode.com/svn/trunk/ 에서 Download

4.3.3.5 C로 이루어진 OpenCV File을 Java에서 사용하기 위해 NDK를 이용해서 build

4.3.3.6 Cygwin에서 다음의 명령어를 수행한다.

4.3.3.6.1 OpenCV Source 설치 폴더로 이동

4.3.3.6.2 Sh build.sh

4.3.3.6.3 Build 결과물들을 실행 예제의 폴더 경로로 이동시킨다.

4.3.3.6.4 Android-OpenCV Source file을 예제 파일과 합친다.

4.3.3.6.5 합친 프로젝트를 Eclipse에 추가 시킨다.

4.3.3.7 예제 실행 결과는 시연으로 보이도록 한다.
5 Plan

5.1 진행상황

<table>
<thead>
<tr>
<th>사전조사</th>
<th>8월 2주차</th>
<th>8월 3주차</th>
<th>8월 4주차</th>
<th>9월 1주차</th>
<th>9월 2주차</th>
<th>9월 4주차</th>
<th>10월 1주차</th>
<th>10월 2주차</th>
</tr>
</thead>
<tbody>
<tr>
<td>안드로이드</td>
<td>디바이스 인식/UI</td>
<td>기본 UI 완성</td>
<td></td>
<td>OpenCV 활용 물체인식</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>모바일에서 PC로 메세지전송</td>
<td>안드로이드 소켓 통신 파트 완성</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>레시피수신</td>
<td>C# 소켓 통신 파트 완성</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>디바이스 제어</td>
<td>미디어 센서 컨트롤러 트롤러 완성</td>
<td>IR 컨트롤러 진행중</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>디버깅 &amp; 테스팅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>테스팅 계획 논의중</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 프로젝트 문제 사항

5.2.1 적외선 부분 장비 구입에 대한 논의가 아직 이루어지지 않았음

5.2.2 UI 부분에 대한 상세 항목에 대한 논의가 부족함

5.2.3 파트를 분담하여 개발하는 방식이어서 통합 시 문제점에 대한 파악이 되지 않았음
6 References

http://www.arduino.cc

http://cafe.naver.com/carroty.cafe?iframe_url=/ArticleRead.nhn%3Farticleid=137505&


손에 잡히는 아두이노 // 마시모 밴지 저 // 인사이트