

분류번호: 2016-02-4105-06

졸업작품 최종보고서

화재 탐사용 RC CAR

지도교수 : 이 예 훈 교수

제 출 일 : 2016년 10월 19일

제 작 자 : 13184354 박 승 진

13184415 이 정 섭

11384301 강 동 육

06107804 김 종 표

서울과학기술대학교 전자IT미디어공학과

목 차

| | |
|--|-----|
| └─ 요약 | i |
| └─ 표 복차 | ii |
| └─ 그림복차 | ii |
| └─ 사진복차 | iii |
| I. 개요 | |
| I. I. 작품의 목표 | 1 |
| I. II. 제작 배경 | 1 |
| I. III. 목표설정 풍기 | 1 |
| I. IV. 문제점 | 2 |
| I. V. 해결방안 | 2 |
| I. VI. 설계 진행일정 | 2 |
| II. 시스템의 설계 | 3 |
| II. I. 5대 설계 원칙 구현 | 3 |
| II. II. 설계절차 기준 | 4 |
| I. 요소설계 H/W | 4 |
| II. 요소설계 S/W | 8 |
| III. 시스템의 구성 도면 | 20 |
| III. 시스템의 제작 | 21 |
| III. I. 1학기 기본제작 | 21 |
| I. H/W 기본설계 | 21 |
| II. S/W 기본설계 | 23 |
| III. II. 2학기 보충제작 및 Feedback 제작 | 23 |
| I. H/W 보충설계 | 23 |
| II. S/W 보충설계 | 24 |
| III. 외형 제작 | 25 |
| III. III. 사용 부품 명세서 | 25 |
| IV. 시스템의 동작 검증 | 26 |
| IV. I. H/W, S/W 통합검증 | 26 |
| I. RaspberryPi (웨서비) | 26 |
| II. breadduino (보드제이) | 27 |
| III. 앱 어플리케이션 | 27 |
| IV. II. 완성된 전체 시스템의 목표 성능에 대한 평가항목 및 평가 결과 | 28 |
| V. 결론 | 30 |
| 참고문헌 | 31 |
| 간사의글 | 32 |

요 약

제 목 : 화재 탐사용 RC CAR by IoT

최근 세월호 사고나 지진 등 사회 안전망에 대한 우려가 많아진 만큼, '안전'에 대한 국민적 관심과 기대도 한층 높아져 있는 상태이다. 이러한 사회 동향에 따라 우리 팀은 새난 관리 IoT(Information and Communications Technologies)를 기반으로 여러 가지 센서와 카메라를 장착한 화재 재난용 RC Car를 기획하였다. 이 작품의 목적은 새난 현장에서 인명 피해를 최소화시키고 스마트 새난 지원 시스템을 구축하는 데 기여하여 좀 더 안전한 사회를 만드는데 이바지하는 것이다.

RC Car를 컨트롤하기 위한 방법으로, 사물인터넷(Internet Of Things)을 이용하였다. 사물인터넷이란 기존의 유선통신을 기반으로 한 인터넷이나 모바일 인터넷보다 진화된 단계로, 인터넷에 연결된 기기가 사람의 개입 없이 상호간에 알아서 정보를 주고받아 처리하는 것이다. 사물이 인간에 의존하지 않고 통신을 주고받는 전에서 기존의 유비쿼터스나 M2M(Machine to Machine: 사물시능통신)과 비슷하기도 하지만, 통신장비와 사람과의 통신을 주목적으로 하는 M2M의 개념을 인터넷으로 확장하여 사물은 물론이고 현실과 가상세계의 모든 정보와 상호작용하는 개념으로 진화한 단계라고 할 수 있다.

이를 구현하기 위한 기술 요소로는 유형의 사물과 주위 환경으로부터 정보를 얻는 '센싱 기술', 사물이 인터넷에 연결되도록 지원하는 '유무선 통신 및 네트워크 인프라 기술', 각종 서비스 분야와 형태에 적합하게 정보를 제공하고 처리하거나 각종 기술을 융합하는 '서비스 인터페이스 기술'이 핵심이며, 대량의 데이터 등 사물 인터넷 구성요소에 대한 해킹이나 정보 유출을 막기 위한 '보안 기술'도 필수적이다.

작품의 기대효과는 화재, 폭발 등 재난현장에 사람이 직접 들어가서 구조하지만, 이 작품을 이용하여 새난현장에 사람 대신 무인되어 내부 상황을 보다 정확하고 안전하게 파악할 수 있다. 손 안의 스마트폰이나 태블릿PC로 새난 현장의 내부 상황을 카메라와 센서를 통해 얻은 정보를 실시간으로 확인할 수 있다. 따라서 안전 분야를 새로운 창조산업으로 육성하고 있는 정부 정책에 따라 공직개발원조(PDA) 사업 연계 등 기술 집약형 중소·중견 안전기업의 재벌을 다양화시키는 데 일조할 것으로 기대된다.

이러한 성과 이외에도 이 작품을 조금 더 응용하면 두둥두진한 결과를 도출해 낼 수 있을 것이라 생각한다. 인터넷은 지구 반대편에서라도 쉽게 접속 가능하므로, 인터넷을 통해서 여러 기기들과 소통하고 제어할 수 있게 될 것이고, 이를 화재 현장이라는 특수한 상황을 넘어 다양한 분야에 효과적으로 쓰일 것이다.

표 목 차

| | |
|---------------------------------------|----|
| Table 1. 일정 계획표 | 3 |
| Table 2. 라즈베리파이(웹서버 보드) 사양도 | 7 |
| Table 3. 1학기 기본제작 진행일정 | 29 |
| Table 4. 2학기 보충제작 및 진행일정 | 32 |
| Table 5. 사용 부품 명세서 | 34 |
| Table 6. 목표 성능에 대한 평가항목 및 평가 결과 | 38 |

그림목차

| | |
|--------------------------------|----|
| Fig 1. 구상도 | 3 |
| Fig 2. 보터제어 전원부 | 5 |
| Fig 3. 보터 제어부(PWM) | 6 |
| Fig 4. 보터제어 전체 회로도 | 6 |
| Fig 5. 웹서버 전체 회로도 | 7 |
| Fig 6. 웹서버 보드 외형도 | 8 |
| Fig 7. 보터드라이버 회로도 (밀디파이) | 9 |
| Fig 8. GPIO Pin 매치도 | 9 |
| Fig 9. 시스템의 구성 도면 | 29 |
| Fig 10. MC와 웹서버 보드와의 연결 | 30 |

사진목차

| | |
|--|----|
| photo 1. 실계 사진 | 4 |
| photo 2. 밀디 파이 | 8 |
| photo 3. 밀디파이 결합모습 | 9 |
| photo 4. 리눅스에서의 무선환경 구축 | 12 |
| photo 5. 무선 구축 완료 (웹서버 환경) | 13 |
| photo 6. 웹페이지에서의 GPIO 작동/상태 확인 | 21 |
| photo 7. 이클립스 그래픽 레이아웃 | 23 |
| photo 8. 웹서버 보드와 MC와의 연결 모습 | 31 |
| photo 9. RC카 제어신호 및 카메라 상/하 제어신호 | 31 |
| photo 10. 내부 HAT 종 결합 모습 | 32 |
| photo 11. 앱으로 보는 영상 및 제어신호 확인 | 33 |
| photo 12. RC Car측 시스템과 Client 시스템의 외형 모습 | 33 |
| photo 13. 웹서버 GPIO단자 4,17,22,23,24 각 푸드 정상 출력 확인 | 34 |
| photo 14. 하(웹페이지) | 35 |
| photo 15. 하 | 35 |
| photo 16. 상(웹페이지) | 35 |
| photo 17. 상 | 35 |
| photo 18. 좌(웹페이지) | 36 |
| photo 19. 좌 | 36 |
| photo 20. 우(웹페이지) | 36 |
| photo 21. 우 | 36 |
| photo 22. 앱 작동 모습 | 37 |

I. 개요

1-1. 작품의 목표

IoT 기반 산업은 아직 자라고 있는 새로운 플레이그라운드이며 앞으로 많은 분야의 비즈니스 창출과 함께 기술의 발전이 기대되고 있다. 이에 따라 IoT 기술에 대해 폭넓게 이해하는 한편 이 기술을 통해 얼마나 삶의 질이 바뀔 수 있고 어떻게 생활을 바꿔나갈 수 있을지, 창의적인 사고를 바탕으로 공학적인 성과를 얻고자 하였다. 문제가 되었던 사물과 사람간의 소통의 유연성 또한 스마트폰을 통해 정보교환 과정의 간소화와 사물과의 상시 소통이 가능하게 된으로써 보다 능동적인 소통이 가능하게 되었다.

또한 현재 ‘안전’에 관한 문제가 굉장히 중요한 사안으로 떠오르고 있으며 우리는 화재 현장에서 ‘소방관 순직’이라는 뉴스를 실신치 않게 볼 수 있다.

따라서 작품 목표로는 위와 같은 IoT 기술을 기반으로하여 카메라와 여러 센서의 정보가 인터넷을 통해 안드로이드 모바일로 전송되어 사용자가 정보를 확인하는 것과 사용자가 모바일을 통해 인터넷에 접속하여 카메라를 제어하는 것이다. 이로 인해 위험한 상황 속에서 소방관들에게 조금 더 안전한 곳에서 화재 현장의 내부상황을 파악할 수 있도록 도움을 준다.

1-2. 제작 배경

2014년 4월 16일 전도 앞바다에서의 세월호 사고와 최근 2016년 9월 10일 경주 지진 등 사회 안전망에 대한 국민적 자성과 우려가 많아진 만큼 ‘안전’에 대한 국민적 관심과 기대도 한층 더 높아졌다. 특히 세월호 사고 이후 정부는 안전사회 확보와 안전 산업을 성장시키기 위한 안전투자를 (15년도 6004억 원 가량) 확대 시켜왔다. 이러한 사회적 동향에 따라 재난관리를 기반으로 고성능 센서와 카메라를 장착한 화재 재난용 RC Car를 만들게 되었다.

또한 요즘 세계 각국에서 ‘IoT’ 기반 산업이 주면 위로 떠오르고 있다. IoT(Internet of Things)란 교유하게 식별 가능한 ‘사물(Things)’이 만들어낸 정보를 인터넷을 통해 공유하는 환경을 의미한다. 인텔은 ‘IoT’를 통해서 2020년에는 40억 명의 인구가 310억 개의 디바이스를 통해서 인터넷에 연결될 것이라고 예상하고 있으며 현재 구글이 만들고 있는 ‘구글 글래스’와 애플이 준비 중이라는 ‘아이와치’는 기대가 되는 IoT 기반의 제품들이다. IoT 기술은 사물과 사람이 결합된 소셜 네트워크를 구성함으로써 사물 간 통신이나 사람의 개입 없이 자동적으로 사람을 도와주는 것이 가능하다. 또한 사람이 원할 때 사물을 연결하여 필요한 서비스를 제공받을 수 있게 만들어 줌으로써 휴식적인 생활환경을 조성하게 된다. 예를 들어, 누구나 간편하게 손안의 스마트폰으로 자신이 필요한 곳의 물체를 제어할 수 있으므로 그로인한 에너지절약과 고효율의 업무처리가 가능하다. IoT 산업은 국가 인프라인 유/무선 네트워크를 기반으로 성장하는 산업이므로 정부가 주도적으로 산업을 장려하고 있으며 스마트 단말기, 통신 모듈과 센서 등의 기술 발달 및 표준 확대로 인해 전 산업 분야에 걸쳐 영향력이 크게 확대될 전망이다. 미국 국가정보위원회(NIC)는 IoT를 2025년까지 국가경쟁력에 영향을 미칠 ‘혁신적인 파괴적 기술(Disruptive Civil Technology)’ 중 하나로 선정한 상태이고 유럽 및 중국 등 각국에서도 연구센터를 구축, 산업기금을 조성 중에 있다. 최근 개발된 지전력 네트워킹 칩으로 인해 더욱 손쉬운 IoT 기술 개발이 가능해졌다.

그리므로 우리 팀은 IoT를 기반으로 한 화재 탐사용 RC Car를 제작하게 되었다.

1-3. 목표 설정 초기

밀리 떨어진 장소에서도 손안의 스마트폰 혹은 태블릿PC등을 통해서 무언가를 확인하고 제어해 보고 싶었다. 처음에는 단순한 RC Car를 생각했지만, 디 나아가 작품을 설계하는 과정에서 여기에 여러 센서와 카메라를 부착해서 사람이 들어가지 못하는 위험한 장소를 대신 들어 갈 수 있는 RC Car를 생각하게 되었다.

1-4. 문제점

1-4-1. 통신 문제

공유기를 통해 무선통신을 할 때, 무선공유기의 거리상의 제한이 있었다. 즉, 와이파이 신호가 유지되는 일정 범위 안에서는 내부네트워크도 접속하여 영상과 제어가 딜레이 없이 무난하게 작동 하였지만, 무선공유기의 와이파이 신호가 유지되지 않는 범위 밖에서는 외부Wifi 또는 3G/4G를 통해 접속을 하여야 하는데, 이 경우 1~2초의 추가적인 딜레이가 생긴다.

1-4-2. 보안 문제

또한 웹서비스를 이용한 작품이기 때문에 보안이 취약한 문제가 있다. 이 부분은 특정 포드를 사용하여 위험도를 낮추는 방법을 사용하여 해킹에 대한 위험도를 낮추는 방법만 사용하고 있었지만 좀 더 나은 보안대책이 필요한 것 같다.

1-4-3. 동시 접속 문제

영상은 스트리밍할 때 웹서비스에 부하가 있기 때문에 동시에 접속하여 영상을 보려고 하면 웹서비스 보드가 다운되는 경우가 있다.

1-4-4. 온도 문제

화재 현장에서 사용하는 물건이다 보니 주위의 온도가 실온보다 높을 가능성이 많고, 이로 인해 차 내부의 온도가 높아서 문제가 생길 수 있다.

1-5. 해결방안

1-5-1. 통신 문제

이 문제를 해결하기 위해 도메인 서비스를 인터넷에서 신청하거나 공인 IP를 제공하는 프로그램 등을 사용해야 될 것으로 보인다.

1-5-2. 보안 문제

특정 포드를 사용하여 위험도를 낮추는 방법도 좋은 방법이지만, 관련 IP를 주기적으로 업데이트해주면서 접속자와 배정시켜주는 DDNS기능을 사용하는 것도 좋은 해결책이라고 본다.

1-5-3. 동시 접속 문제

스트리밍 서비스로 경유해서 영상을 전송하거나 NAS 서비스를 이용하는 방법 등이 해결책으로 보이며, 웹서비스 기능을 담당하는 Raspberry 사양을 좀 더 높인다면 해결될 것으로 보인다.

1-5-4 온도 문제

차량 외부를 최대한 열을 차단 할 수 있는 재료를 사용하여 감싸고 차량 내부에 열전화로와 콜러를 이용하여 내부의 열을 외부로 막출한다.

1-6. 설계 진행일정



Table 1. 일정 계획표

II. 시스템의 설계

2-1. 5대 설계 원칙 구현

2-1-1. 구상 설계 (Idea 도출)

시스템은 기본적으로 카메라와 센서에서 읽어드린 정보가 웹서비스에 올라가고 안드로이드 모바일을 통하여 사용자가 정보를 확인하고 모바일을 통해 인터넷에 접속하여 카메라, 센서, RC car를 제어하는 것이다.

2-1-2. 기능 설계

목표를 구현하기 위한 IoT 기술은 크게 센싱 기술, 통신 및 네트워크 기술, 서비스 인터페이스 기술도 나눌 수 있다. 외부 환경으로부터 정보를 얻는 센싱 기술에는 카메라의 영상처리기술과 여러 다양한 센서, 저전력으로 구동되는 회로가 필요하다. 통신 및 네트워크 기술에는 웹서비스를 사용한다. 그리고 서비스 기술에는 모바일 어플리케이션, 웹 서비스, 카메라 동작제어 등을 위해 안드로이드 개발환경 및 MCU 개발환경을 구축한다. 간편하게 손안의 스마트폰으로 필요한 곳으로 RC Car를 제어할 수 있으므로 그로인한 효율적이고 안전한 업무처리가 가능해진다.

| H/W | S/W |
|--------------------------|-----------------|
| Raspberry Pi (웹서버 보드) | 리눅스 기반(Webiopi) |
| Iteaduino (MCU 보드) | C언어(아두이노 스케치) |
| 모바일 어플리케이션 (APP) | 이클립스(자바 오픈소스) |

Fig 1. 구상도

2-1-3. 다목적 설계

화재현장의 내부 영상뿐만 아니라, 내부의 온도, 공기 중의 산소 농도 등의 센서를 RC카를 침투시켜서 이동성을 부가함으로써 사람이 들어가지 못하는 곳에서 필요한 정보 등을 얻을 수 있다.

2-1-4. 경제적 설계

작품을 구현하는데 있어서 들어간 총 비용은 40만원 정도이다.

주요 기능을 담당하는 웹서비 보드인 Raspberry Pi 가 4만원, MCI 보드가 3만원, 각종 센서들이 5만원, 아플리케이션은 무료제작이다. 이처럼 비용적인 측면에서는 매우 저렴하고 경제적이다. 웹서비 보드와 MCI(보더제어)의 운영전력 또한 3V~5V인 저전력이 사용되는 회로가 구성되기 때문에 에너지 측면에서도 효율성을 가진다.

2-1-5. 미관적 설계

이동성을 위한 RC카를 설계하는 과정에서 턱이 있거나 여러 가지 장애물에 있어서 유연한 장점을 지닌 캐터필러 유형의 전차모델을 선택하였다. 그리고 외형적으로 차지분해 보일 수 있는 H/W들과 선들은 프레임을 2층 구조로 설계하여 1층에 고정시키 놓음으로써 보기 좋게 해놓았다. 카메라의 위치는 영상을 확인하는 목적으로 충 실할 수 있게 하기 위해서 가장 위쪽에 설치하였으며 상하로 움직이면서 RC카의 방향 성의 제한이 되는 부분을 보완 해주도록 설계하였다.

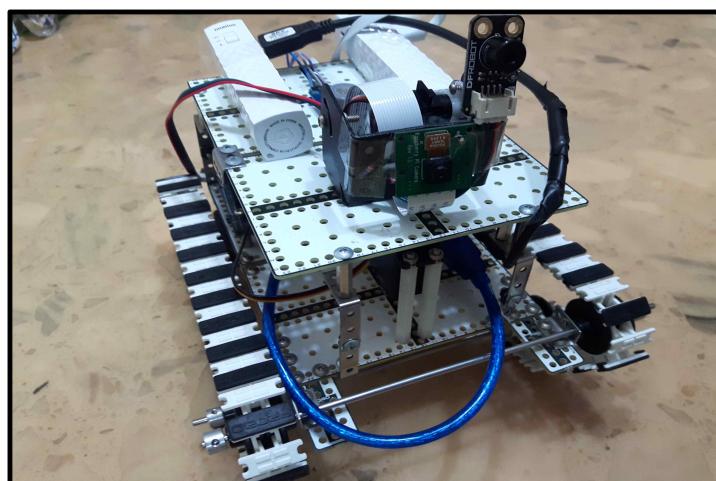


Photo 1. 설계 사진

2-2. 설계 실차 기술
 2-2-1. 요소 설계(H/W)
2-2-1-1 노디제어 전원부 설계

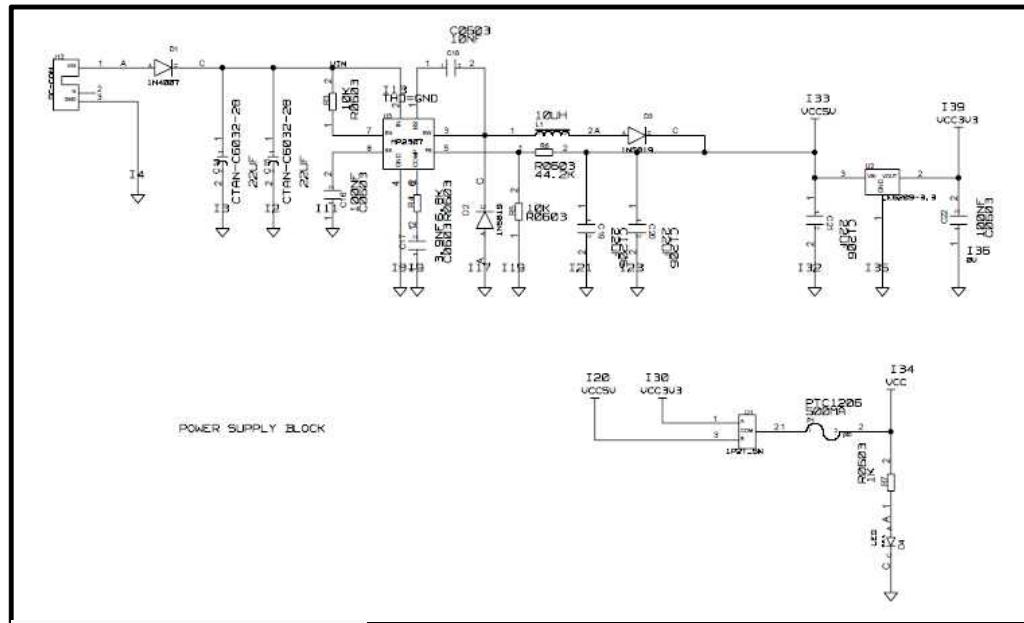


Fig 2. 노디제어 전원부

2-2-1-2. 보터 제어부 설계

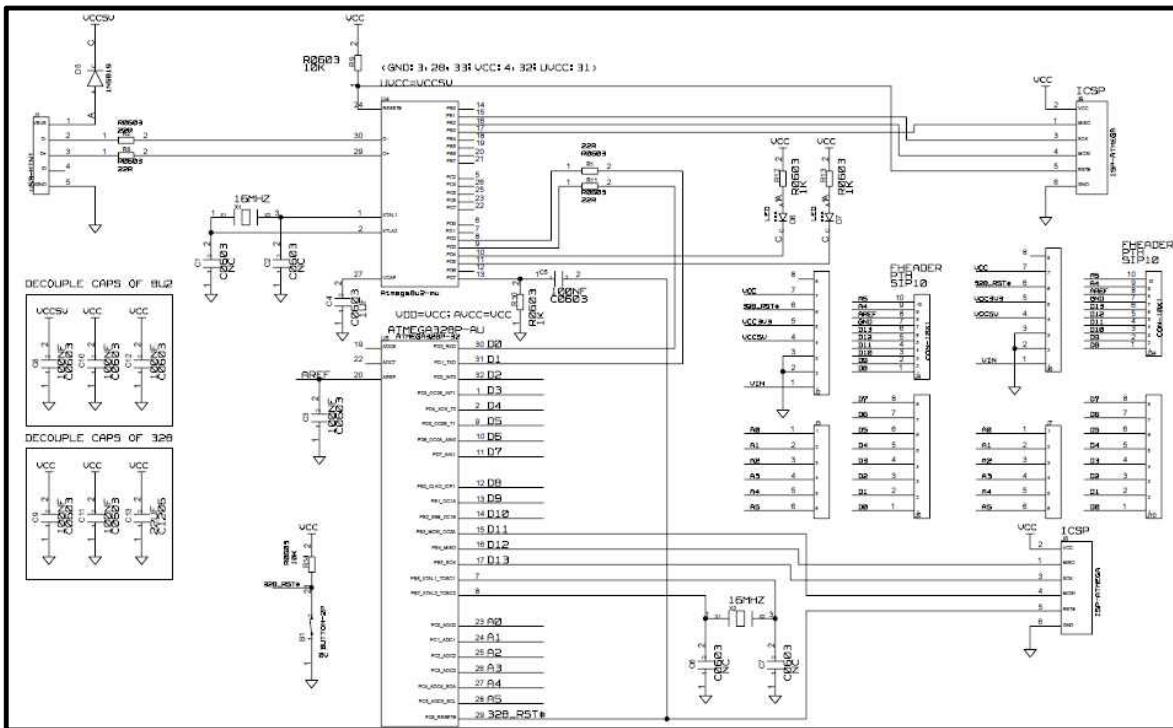


Fig. 3. 보터 제어부(PWM)

보터를 제어하기 위한 보드로 아두이노라는 사동 인터넷 기반의 개방형 플랫폼을 사용하였다. 오픈 소스가 공개되어 소프트웨어 개발이 빙용적이며 센서의 조작이 쉽고, 여러 하드웨어 장치들과의 연동성이 뛰어나다. 아두이노 스케치 프로그램을 이용하여 4개의 신호를 받았을 때 보터의 상하좌우 이동이 가능하게 프로그래밍 하였다.

회로도에서 PWM 부분으로써 디지털핀 D10, D9 PIN을 사용하였다.

PWM으로 코딩된 상하신호가 출력단자인 D10(상,하), D9(좌우)를 통해서 외부단자로 전송될 수 있게 프로그래밍 하였다.

A0핀을 이용하여 일산화탄소 센서의 아날로그 센서값을 읽어 올 수 있도록 하였고 SDA, SCL 핀을 이용하여 온도 센서의 온도 값을 읽을 수 있도록 하였다.

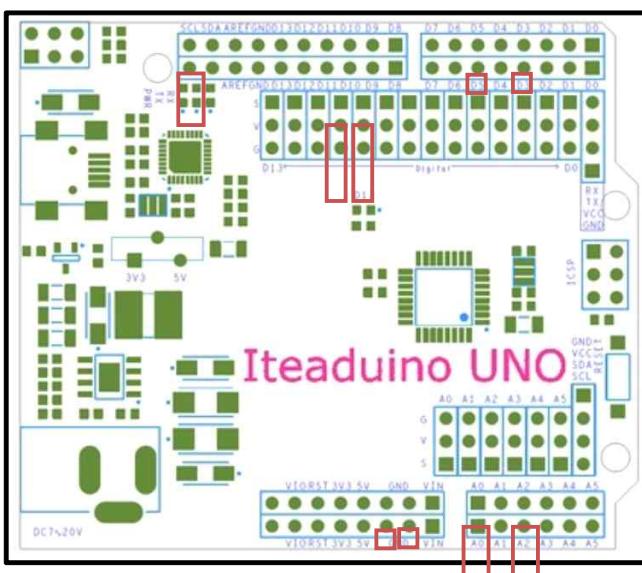


Fig. 4. 보터제어 전체 회로도

2-2-1-2. 웹서버 회로 설계

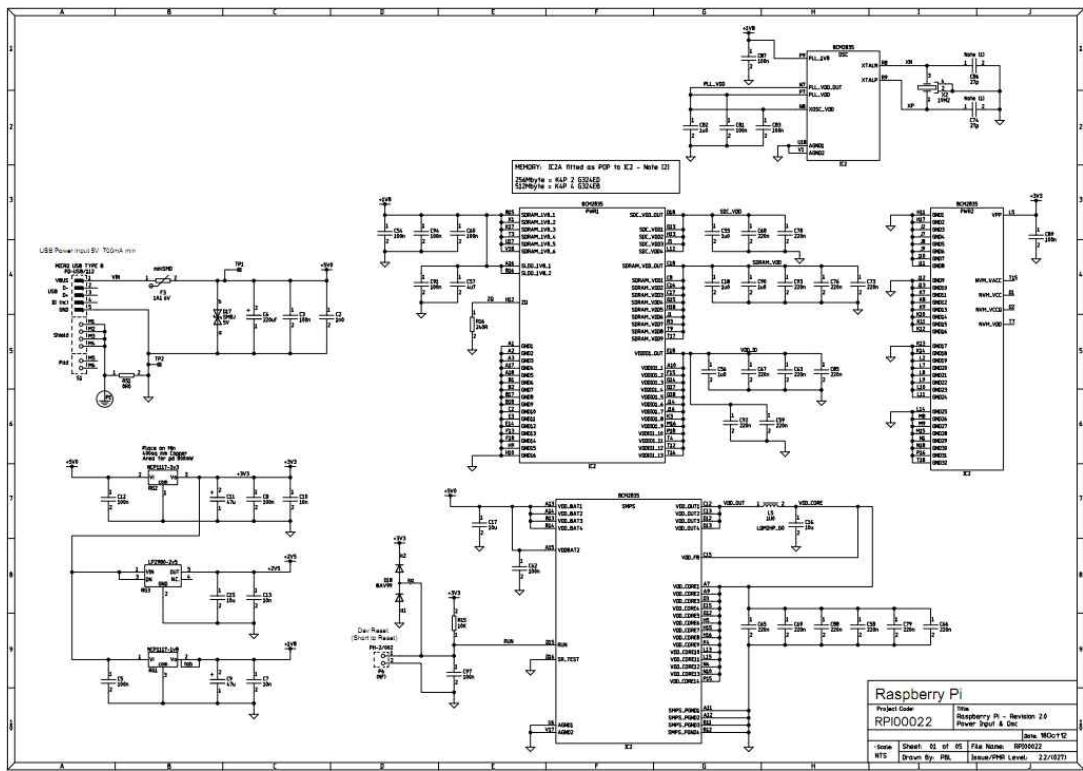


Fig. 5. 웹서버 전체 회로도

| | 보덴 A | 보덴 B |
|---------------|--|--|
| SoC | 브로드컴 BCM2835 (CPU, GPU, DSP, 그리고 SDR SDRAM) | |
| CPU | 700 MHz ARM1176JZF-S core (ARM11 계열) | |
| GPU | 브로드컴 비디오코어 IV, OpenGL ES 2.0, 1080p30 h.264/MPEG 4 AVC 높은 프로파일 디코더 | |
| 메모리 (SDRAM) | 256 MB (GPU와 같이 사용) as of October 15, 2012 | 512 MB (GPU와 같이 사용) as of October 15, 2012 |
| USB 2.0 ports | 1 | 2 (내장 USB 허브 하나 제외) |
| 비디오 출력 | 컴포지트 RCA (PAL & NTSC), HDMI (rev 1.3 & 1.4), 액정 디스플레이를 위한 디스플레이 시리얼 인터페이스 640x350 부터 1920x1200 이상의 14개의 HDMI 해상도와 여러 PAL, NTSC 표준들 | |
| 오디오 출력 | 3.5 mm 잭, HDMI | |
| 내장 저장장치 | SD / MMC / SDIO 카드 슬롯 | |
| 내장 네트워크 | None | 10/100 이더넷 (RJ45) |
| 로우레벨 주변장치 | 8 × GPIO, UART, I ² C 버스, 2개의 칩을 선택할 수 있는 SPI, 3.3 V, +5 V, 접지 | |
| 전력 | 500 mA (2.5 W) | 700 mA (3.5 W) |
| 지원 | MicroSD에서부터 5 볼트 혹은 GPIO 헤더 | |
| 크기 | 85.60×53.98mm | |
| 무게 | 45g | |
| 운영체제 | Debian GNU/Linux, Fedora, Arch Linux ARM, RISC OS | |

Table 2. 라즈베리파이(웹서버 보드) 사양도

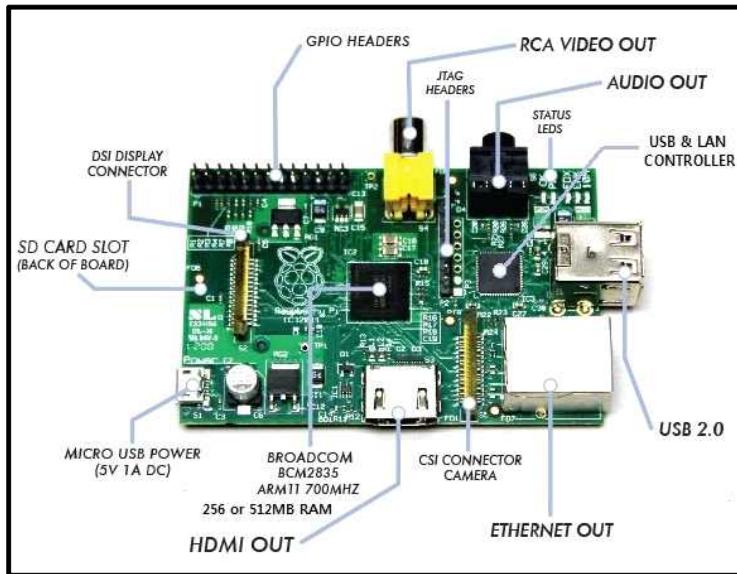


Fig. 6. 웹서버 보드 외형도

- 라즈베리파이(웹서버 보드)는 영국의 라즈베리 파이 재단이 학교에서 기초 컴퓨터 과학 교육을 증진시키기 위해 만든 싱글 보드 컴퓨터이다. 라즈베리 파이는 브로드컴의 BCM2835 단일 칩 시스템을 사용하여, 이 칩에는 ARM1176JZF-S 700MHz 프로세서, 비디오코어 IV GPU와 256 메가바이트 RAM이 들어 있다. 라즈베리파이는 하드 디스크 드라이브나 솔리드 스테이트 드라이브를 내장하고 있지 않으며, SD카드를 외부 기억 장치로 사용한다. 또한 라즈베리파이 재단 측에서는 라즈베리파이에 포팅한 대비안과 아치 리눅스, Qt on Pi 등의 리눅스 배포판을 제공하고 있다.

2-2-1-2. 멀티 파이 (RC카 호환보드)

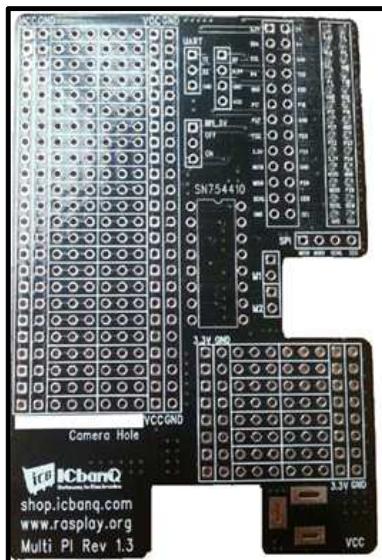


photo 2. 멀티 파이

멀티파이의 경우 라즈베리파이 보드 위에 그대로 장착할 수 있게 편의적으로 설계되었다.

오른쪽 상단에 보이는 26개의 포트가 GPIO(General Purpose Input Output)이며, 이것은 웹서비스 상에서 원격으로 사물의 제어를 가능하게 해주는 핀들이다.

멀티파이는 내부적으로 Pin들이 RC카 제작을 위해 상호적으로 편리하게 연결되어 있다.

즉, GPIO 4, 17, 27, 22 Pin들이 가운데 보이는 SN754410 모터드라이버의 VCC, GND, 모터로 연결되는 포트로의 연결이 자체적으로 되어있으나 때문에 IC칩만 꽂으면 구동이 가능하게끔 설계되어 있다.

RC카 모터제어를 위한 별도의 회로를 설계할 필요가 없다.

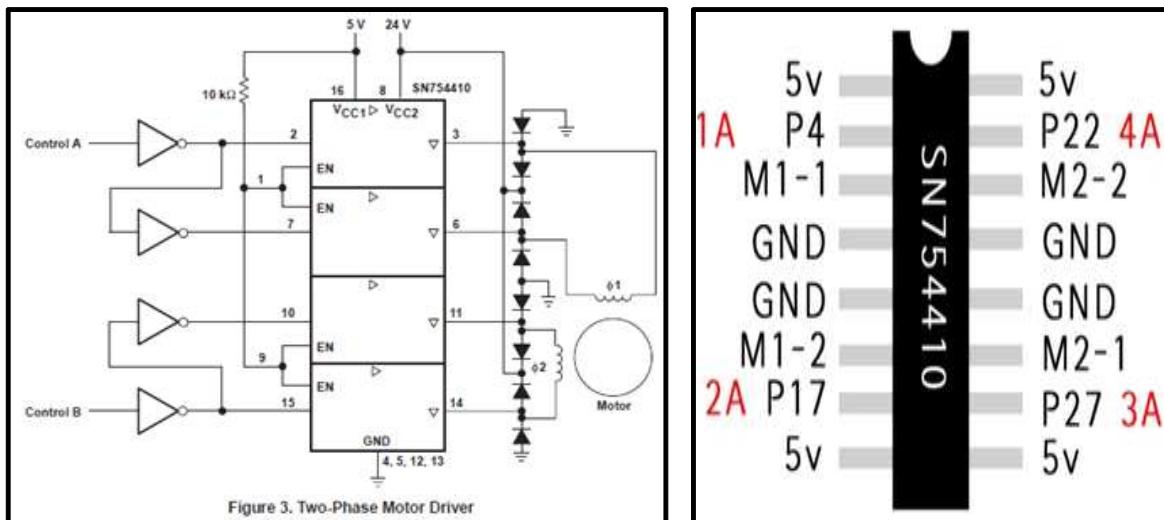


Fig 7. 모터드라이버 회로도 (밀티파이)

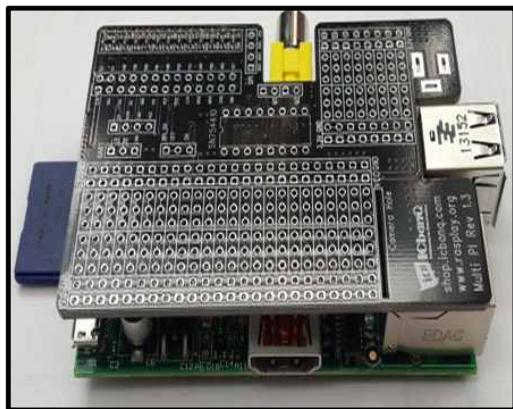


photo 3. 밀티파이 결합보습

| | | |
|-----------|-----|----------|
| 3.3V | ① ② | 5V |
| #2, SDA | ③ ④ | 5V |
| #3, SCL | ⑤ ⑥ | GND |
| #4 | ⑦ ⑧ | #14, TXD |
| GND | ⑨ ⑩ | #15, RXD |
| #17 | ⑪ ⑫ | #18 |
| #27 | ⑬ ⑭ | GND |
| #22 | ⑮ ⑯ | #23 |
| 3.3V | ⑰ ⑱ | #24 |
| #10, MOSI | ⑲ ⑳ | GND |
| #9, MISO | ㉑ ㉒ | #25 |
| #11, SCKL | ㉓ ㉔ | #8, CE0 |
| GND | ㉕ ㉖ | #7, CE1 |

Fig 8. GPIO Pin 배치도

2-2-2. 요소 설계 (S/W)

2-2-2-1. 모터제어(MCU) 프로그래밍

아두이노 스케치 프로그램을 이용하여 4개의 신호를 받았을 때 모터의 상하좌우 이동이 가능하게 프로그래밍 하였고, 온도 센서와 CO 농도 센서의 값을 읽어오는 코드를 프로그래밍 하였다. 아두이노는 프로그램 언어로 C, C++언어를 지원하고 있다.

< 모터 제어 >

```
#include <IR Thermometer Sensor MLX90614.h>
#include <Wire.h>
#include <Servo.h>

Servo myservo;
Servo myservol;
```

```

const int upbuttonPin = 4;
const int downbuttonPin = 5;
const int coGasPin = A0 ;
int pos = 30;
int pos1 = 90;
unsigned long checkTime;
int upbuttonState = 0;
int downbuttonState = 0;
void posMotor();
IR Thermometer Sensor MLX90614 MLX90614 = IR Thermometer Sensor MLX90614();

void setup()
{
    Serial.begin(9600);
    MLX90614.begin();

    myservo.attach(9);
    myservo1.attach(10);

    pinMode(upbuttonPin, INPUT);
    pinMode(downbuttonPin, INPUT);

    myservo.write(45);
    myservo1.write(90);

    checkTime = millis();
}

void loop()
{
    upbuttonState = digitalRead(upbuttonPin);
    downbuttonState = digitalRead(downbuttonPin);
    posMotor();

    if((millis() - checkTime)>2000){
        checkTime = millis();
        Serial.print("CO = "); Serial.println(analogRead(coGasPin));
        Serial.print("Ambient = ");
    }
}

```

```

Serial.print(MLX90614.GetAmbientTemp_Celsius());    Serial.println(" *C");
    Serial.print("Object                  =                 " );
Serial.print(MLX90614.GetObjectTemp_Celsius());    Serial.println(" *C");
}

void posSlider() {
    if (upbuttonState == HIGH && downbuttonState == LOW) {
        if(30 <= pos < 170) {
            pos=pos+1;
            myservo.write(pos);
            delay(45);
            if(170 <= pos){
                pos=170;
            }
        }
    }

    else if (upbuttonState == LOW && downbuttonState == HIGH) {
        if(30 < pos < 170) {
            pos=pos-1;
            myservo.write(pos);
            delay(45);
            if(pos <= 30){
                pos=30;
            }
        }
    }

    else if (upbuttonState == HIGH && downbuttonState == HIGH) {
        if(30 <= pos < 170) {
            pos=pos+1;
            myservo1.write(pos);
            delay(45);
            if(170 <= pos){
                pos=170;
            }
        }
    }
}

```

```
        }  
  
    else if (upbuttonState == LOW && downbuttonState == LOW) {  
        if(30 < pos < 170) {  
            pos=pos-1;  
            myservo1.write(pos);  
            delay(45);  
            if(pos <= 30){  
                pos=30;  
            }  
        }  
    }  
}
```

다음 화면은 아두이노 스케치를 통해 프로그래밍을 진행한 부분이다. 각 신호에 대해서 선언한 부분과, 신호를 줌으로써 모터가 30도에서 170도까지 상하/좌우로 움직이도록 하였다. 여기서 각도를 30~170로 설정한 이유는 서보모터를 지지하는 벤틸드 구조가 0~180도 까지 완전하게 조립되지 않기 때문에 시스템 구조상 어느 정도의 오차를 주었다. 센서 값은 2초마다 출력 되도록 코드를 작성하였다.

2-2-2-2. 웹서비 보드 프로그래밍

(1) 무선 환경 구축

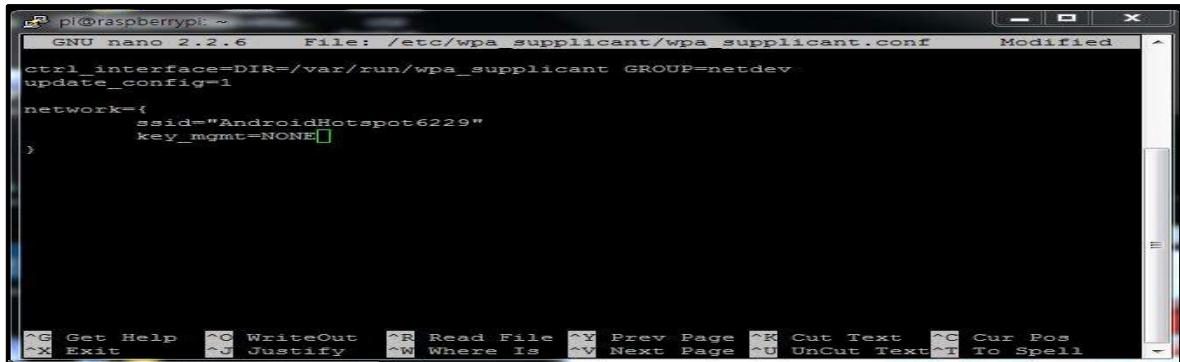


photo 4. 리눅스에서의 무선환경 구축

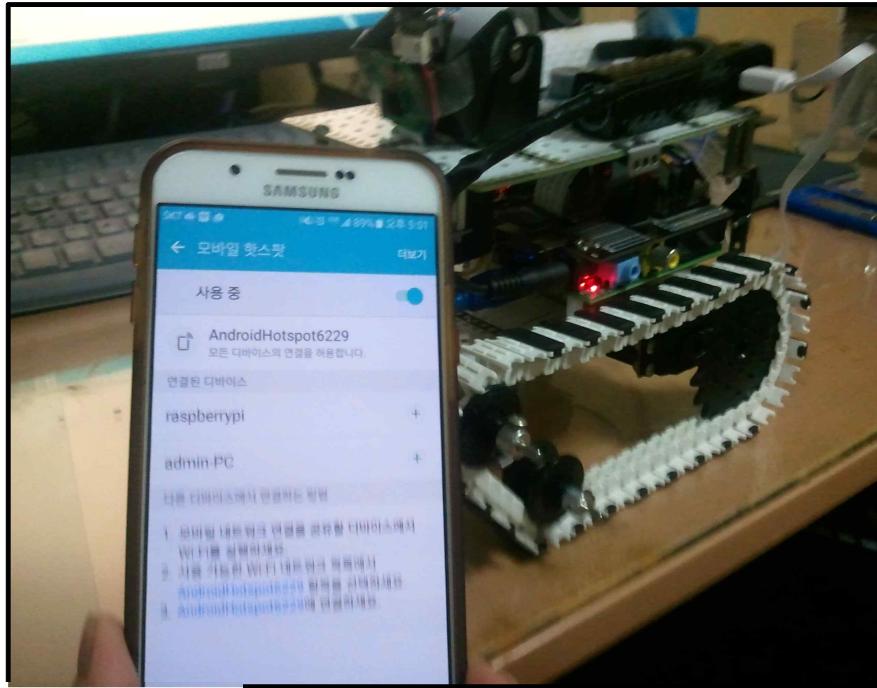


photo 5. 무선 구축 완료 (웹서버 환경)

(2) Webiopi

Webiopi는 이름 그대로 Web상에서 Pi의 GPIO에 관한 작업들을 할 수 있는 프레임워크이다. HTTP, JavaScript, Python 등으로 개발되어 있고, 쉽게 접근할 수 있는 라이브러리를 제공 중이다. 라즈베리파이에 GPIO포트를 웹 상에서 제어가 가능하고 구글 코드 사이트에 공개가 되어있다.

라즈베리파이의 GPIO를 제어하기 위한 많은 방법이 있지만 Web방식의 제어를 위해 Webiopi를 사용하기로 결정하였다. Webiopi는 이름에서 알 수 있듯이 Web Browser에서 GPIO를 제어할 수 있도록 제작되었고, Python으로 Web Server를 구동하고, HTML, JavaScript로 Web Pages를 디자인한다. Python으로 GPIO를 컨트롤 하고, JavaScript로 Python 함수를 호출한으로써 웹서버 모드의 입출력편이 제어가 가능하다.

다음은 Webiopi의 Feature이다.

- REST API over HTTP and CoAP(draft-14) with multicast support
- Server written in Python with zero dependency
- Supports GPIO, Serial, I2C, SPI, 1-Wire with zero dependency
- Supports more than 30 devices including DAC, ADC, sensors...
- Full Python library for the Server, GPIO, Serial, I2C, SPI and devices drivers
- Compatible with both Python 2 and 3
- Extensible and highly customizable
- Login/Password protection

- Mobile devices compatible
- Includes debug web apps
 - GPIO Header
 - GPIO List
 - Serial Monitor
 - Devices Monitor
- Javascript client library built on top of jQuery
- Python client library with HTTP and CoAP support

(2) 1. webiopi 프로그래밍 <Index.html>

다음은 Webiopi 프로그램을 기반으로 하여 보드를 제어하는 버튼을 파이썬 웹서버를 통해 웹페이지에 보여지도록 구현한 프로그래밍 소스이다. macro함수를 이용하여 webiopi.js의 신호구현 부분을 호출하여 버튼을 누름으로써 웹서버 보드의 GPIO에 신호가 발생하게 된다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <meta name="viewport" content = "height = device-height, width =
420, user-scalable = no" />
  <title>WebIOPi + Demo</title>
  <script type="text/javascript" src="/webiopi.js"></script>
  <script type="text/javascript">
    webiopi().ready(function() {
      var button;

      // create a button which call PrintTime
      button = webiopi().createMacroButton("macro", "L",
      "MoveLeft");
      $("#middle").append(button); // append button to middle div

      button = webiopi().createMacroButton("macro", "X", "Stop");
      $("#middle").append(button); // append button to middle div
    });
  </script>
</head>
<body>
  <div id="middle"></div>
</body>
</html>
```

```

        button = webiopi().createMacroButton("macro", "R",
"MoveRight");
        $("#middle").append(button); // append button to middle div

        button = webiopi().createMacroButton("macro", "F",
"MoveForward");
        $("#up").append(button); // append button to up div

        button = webiopi().createMacroButton("macro", "B",
"MoveBackward");
        $("#down").append(button); // append button to down div

        button = webiopi().createMacroButton("macro", "U",
"MoveUp");
        $("#content").append(button); // append button to content
div

        button = webiopi().createMacroButton("macro", "D",
"MoveDown");
        $("#content").append(button); // append button to content
div

        button = webiopi().createMacroButton("macro", "CL",
"CounterClock");
        $("#content").append(button); // append button to content
div

        button = webiopi().createMacroButton("macro", "CR",
"Clock");
        $("#content").append(button); // append button to content
div

        webiopi().refreshGPIO(true);
    });


```

</script>

<style type="text/css">

```

        button {
            margin: 5px 5px 5px 5px;
            width: 45px;
            height: 45px;
            font-size: 24pt;
            font-weight: bold;
            color: black;
        }

        .LOW {
            background-color: White;
        }

        .HIGH {
            background-color: Red;
        }

    </style>
</head>
<body>
    <div id="content" align="center">
        <div id="up"></div>
        <div id="middle"></div>
        <div id="down"></div>
    <div>

</body>
</html>

```

(2)-1. webiopi 프로그래밍 <script.py>

```

# Imports
import webiopi
import time

```

```

# Enable debug output
webiopi.setDebug()

# Retrieve GPIO lib
GPIO = webiopi.GPIO

# SN754410NE
IC1A = 4
IC1B = 17
IC2A = 27
IC2B = 22
IC3A = 23
IC3B = 24
IC4A = 18
IC4B = 25

# Called by WebIOPi at script loading
def setup():
    webiopi.debug("Script with macros - Setup")
    # Setup GPIOs
    GPIO.setFunction(IC1A, GPIO.OUT)
    GPIO.setFunction(IC1B, GPIO.OUT)
    GPIO.setFunction(IC2A, GPIO.OUT)
    GPIO.setFunction(IC2B, GPIO.OUT)
    GPIO.setFunction(IC3A, GPIO.OUT)
    GPIO.setFunction(IC3B, GPIO.OUT)

    GPIO.digitalWrite(IC1A, GPIO.LOW)
    GPIO.digitalWrite(IC1B, GPIO.LOW)
    GPIO.digitalWrite(IC2A, GPIO.LOW)
    GPIO.digitalWrite(IC2B, GPIO.LOW)
    GPIO.digitalWrite(IC3A, GPIO.LOW)
    GPIO.digitalWrite(IC3B, GPIO.LOW)

# Looped by WebIOPi
def loop():
    # Toggle LED each 5 seconds
    webiopi.sleep(5)

```

```

# Called by WebIOPi at server shutdown
def destroy():
    webiopi.debug("Script with macros - Destroy")
    # Reset GPIO functions
    GPIO.setFunction(IC1A, GPIO.IN)
    GPIO.setFunction(IC1B, GPIO.IN)
    GPIO.setFunction(IC2A, GPIO.IN)
    GPIO.setFunction(IC2B, GPIO.IN)
    GPIO.setFunction(IC3A, GPIO.IN)
    GPIO.setFunction(IC3B, GPIO.IN)
    GPIO.setFunction(IC4A, GPIO.IN)
    GPIO.setFunction(IC4B, GPIO.IN)

# A macro without args which return nothing
@webiopi.macro
def MoveLeft():
    GPIO.digitalWrite(IC1A, GPIO.HIGH)
    GPIO.digitalWrite(IC1B, GPIO.LOW)
    GPIO.digitalWrite(IC2A, GPIO.HIGH)
    GPIO.digitalWrite(IC2B, GPIO.LOW)
    GPIO.digitalWrite(IC3A, GPIO.LOW)
    GPIO.digitalWrite(IC3B, GPIO.LOW)
    GPIO.digitalWrite(IC4A, GPIO.LOW)
    GPIO.digitalWrite(IC4B, GPIO.LOW)

@webiopi.macro
def MoveRight():
    GPIO.digitalWrite(IC1A, GPIO.LOW)
    GPIO.digitalWrite(IC1B, GPIO.HIGH)
    GPIO.digitalWrite(IC2A, GPIO.LOW)
    GPIO.digitalWrite(IC2B, GPIO.HIGH)
    GPIO.digitalWrite(IC3A, GPIO.LOW)
    GPIO.digitalWrite(IC3B, GPIO.LOW)
    GPIO.digitalWrite(IC4A, GPIO.LOW)
    GPIO.digitalWrite(IC4B, GPIO.LOW)

@webiopi.macro
def MoveForward():
    GPIO.digitalWrite(IC1A, GPIO.LOW)
    GPIO.digitalWrite(IC1B, GPIO.HIGH)
    GPIO.digitalWrite(IC2A, GPIO.HIGH)

```

```
GPIO.digitalWrite(IC2B, GPIO.LOW)
GPIO.digitalWrite(IC3A, GPIO.LOW)
GPIO.digitalWrite(IC3B, GPIO.LOW)
GPIO.digitalWrite(IC4A, GPIO.LOW)
GPIO.digitalWrite(IC4B, GPIO.LOW)

@webiopi.macro
def MoveBackward():
    GPIO.digitalWrite(IC1A, GPIO.HIGH)
    GPIO.digitalWrite(IC1B, GPIO.LOW)
    GPIO.digitalWrite(IC2A, GPIO.LOW)
    GPIO.digitalWrite(IC2B, GPIO.HIGH)
    GPIO.digitalWrite(IC3A, GPIO.LOW)
    GPIO.digitalWrite(IC3B, GPIO.LOW)
    GPIO.digitalWrite(IC4A, GPIO.LOW)
    GPIO.digitalWrite(IC4B, GPIO.LOW)

@webiopi.macro
def Stop():
    GPIO.digitalWrite(IC1A, GPIO.LOW)
    GPIO.digitalWrite(IC1B, GPIO.LOW)
    GPIO.digitalWrite(IC2A, GPIO.LOW)
    GPIO.digitalWrite(IC2B, GPIO.LOW)
    GPIO.digitalWrite(IC3A, GPIO.LOW)
    GPIO.digitalWrite(IC3B, GPIO.LOW)
    GPIO.digitalWrite(IC4A, GPIO.LOW)
    GPIO.digitalWrite(IC4B, GPIO.LOW)

@webiopi.macro
def MoveUp():
    GPIO.digitalWrite(IC1A, GPIO.LOW)
    GPIO.digitalWrite(IC1B, GPIO.LOW)
    GPIO.digitalWrite(IC2A, GPIO.LOW)
    GPIO.digitalWrite(IC2B, GPIO.LOW)
    GPIO.digitalWrite(IC3A, GPIO.HIGH)
    GPIO.digitalWrite(IC3B, GPIO.LOW)
    GPIO.digitalWrite(IC4A, GPIO.LOW)
    GPIO.digitalWrite(IC4B, GPIO.LOW)

@webiopi.macro
def MoveDown():
```

```
GPIO.digitalWrite(IC1A, GPIO.LOW)
GPIO.digitalWrite(IC1B, GPIO.LOW)
GPIO.digitalWrite(IC2A, GPIO.LOW)
GPIO.digitalWrite(IC2B, GPIO.LOW)
GPIO.digitalWrite(IC3A, GPIO.LOW)
GPIO.digitalWrite(IC3B, GPIO.HIGH)
GPIO.digitalWrite(IC4A, GPIO.LOW)
GPIO.digitalWrite(IC4B, GPIO.LOW)
```

```
@webiopi.macro
def CounterClock():
    GPIO.digitalWrite(IC1A, GPIO.LOW)
    GPIO.digitalWrite(IC1B, GPIO.LOW)
    GPIO.digitalWrite(IC2A, GPIO.LOW)
    GPIO.digitalWrite(IC2B, GPIO.LOW)
    GPIO.digitalWrite(IC3A, GPIO.LOW)
    GPIO.digitalWrite(IC3B, GPIO.LOW)
    GPIO.digitalWrite(IC4A, GPIO.HIGH)
    GPIO.digitalWrite(IC4B, GPIO.LOW)
```

```
@webiopi.macro
def Clock():
    GPIO.digitalWrite(IC1A, GPIO.LOW)
    GPIO.digitalWrite(IC1B, GPIO.LOW)
    GPIO.digitalWrite(IC2A, GPIO.LOW)
    GPIO.digitalWrite(IC2B, GPIO.LOW)
    GPIO.digitalWrite(IC3A, GPIO.LOW)
    GPIO.digitalWrite(IC3B, GPIO.LOW)
    GPIO.digitalWrite(IC4A, GPIO.LOW)
    GPIO.digitalWrite(IC4B, GPIO.HIGH)
```

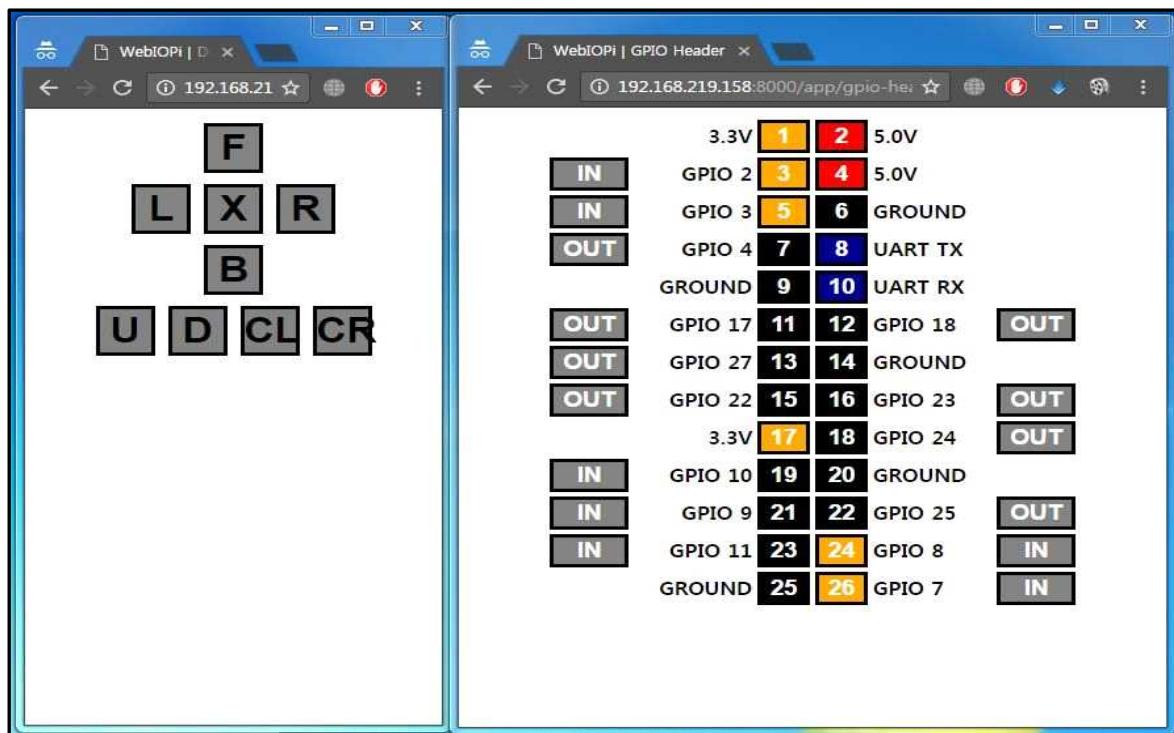


photo 6. 웹페이지에서의 GPIO 작동 상태 확인

위의 그림은 웹서버를 프로그래밍한 후에 웹페이지에서 해당 GPIO Pin이 제대로 출력이 되는지 확인하는 실현이다.

RC카의 전/후/좌/우 제어를 위한 4, 17, 22, 27번 핀이 output으로 신호가 주어진 것을 볼 수 있다.

(2) 2. mjpeg streamer

mjpeg-streamer는 리눅스 환경에서 구동 가능한 카메라로부터 JPG파일을 썩거나 Motion과 같이 프로그램 자체적으로 웹서버 기능이 존재하기 때문에 HTTP를 통해 MJPEG형식의 스트리밍을 제공하는 프로그램이다. Motion이나 opencv 등처럼 영상을 이용하여 제어시스템을 구현할 수는 있지만 cpu에 부하가 적게 걸리고 jpg포맷을 통해 스트리밍하기 때문에 웹으로의 데이터 전송이 빠른 장점이 있다.

인터넷을 통해 실시간으로 영상을 확인한다는 측면에서 이 프로그램이 가장 딜레이가 적은 영상을 보여주기 때문에 영상을 보여주는 프로그램으로 이것을 활용하는 것이 가장 좋은 방법이라고 생각하였다. 영상 신호를 통한 다른 제어 시스템과의 연계가 되지 않는 단점이 있지만 영상을 확인하는 프로그램으로 이것을 사용하고 motion 등의 영상제어 프로그램은 다른 시스템에 구축하여 제어하는 것도 다른 하나의 방편이 될 수 있다고 생각된다.

다음은 웹서비 보드의 전원이 켜졌을 때 MJPG-streamer가 구동이 가능하도록 설정해준 방법이다. etc 폴더의 rc.local 파일을 이용하여 리눅스가 시작할 때 MJPG streamer가 320x240 해상도에 25 프레임으로 동작할 수 있도록 하였다.

```
sudo nano /etc/rc.local
#!/bin/sh -e
#
# This script will be run by the rc daemon
# at startup after all other services have been started
# It is intended to start MJPG Streamer
# and output to http://:8080/mjpg-streamer/www

# LD_LIBRARY_PATH="/home/pi/mjpg-streamer/mjpg-streamer-experimental/"
# LD_PRELOAD=/usr/lib/uv4l/uv4l-ext/armv6l/libuv4l-ext.so
# /home/pi/mjpg-streamer/mjpg-streamer-experimental/mjpg_streamer \
# -i "/home/pi/mjpg-streamer/mjpg-streamer-experimental/input_uvc.so" \
# -d "/dev/video0" \
# -r 320x240 \
# -f 25 \
# -o "/home/pi/mjpg-streamer/mjpg-streamer-experimental/output_http.so" \
# -w "/home/pi/mjpg-streamer/mjpg-streamer-experimental/www"
```

2.2.2.3. 안드로이드 어플리케이션 프로그래밍

모바일 어플리케이션 개발을 위해 아름립스라는 자바 개발 오픈소스 프로그램을 사용하였다. 앱개발을 위한 코드는 다음과 같다.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_gravity="center/center_vertical"
        android:layout_height="20dp"
        android:text="@string/cammera_view" />

    <WebView
        android:id="@+id/WebView01"
        android:layout_width="match_parent"
        android:layout_height="248dp"
        android:layout_marginBottom="0dp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center/center_vertical"
        android:layout_marginBottom="0dp"
        android:text="@string/sensor_view" />

    <WebView
        android:id="@+id/webView02"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```

        android:layout_gravity="right"
        android:layout_marginBottom="0dp"
        android:layout_weight="1.00" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center/center_vertical"
        android:layout_marginBottom="0dp"
        android:text="@string/button_view" />

    <WebView
        android:id="@+id/webView03"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="0dp"
        android:layout_weight="1.00" />

</LinearLayout>

```

위의 코딩은 앱을 실행했을 때, 사용자에게 보여지는 웹뷰와 텍스트들의 레이아웃을 나타낸 것이다. 위의 결과는 아플리케이션 graphic layout을 통해 바로 확인할 수 있다.

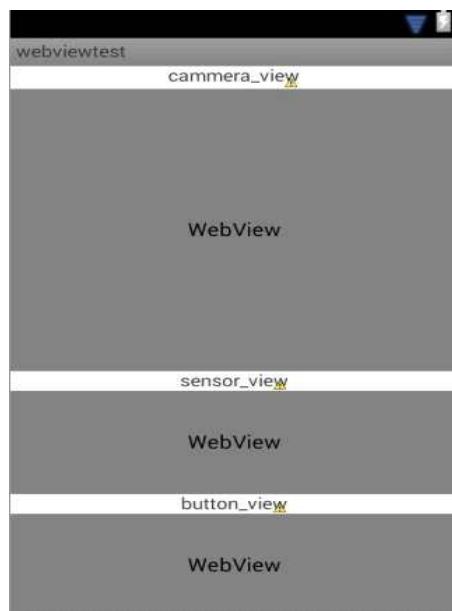


photo 7. 아플리케이션 레이아웃

위의 레이아웃은 java xml파일이고, 다음은 웹뷰를 앱 화면에 띄우기 위해 작성한 3개의 java script이다. 웹뷰를 띄우는 것은 주소(url)를 불러오는 방법으로 코딩했고, 자바 스크립트 파일은 세 개의 액티비티를 따로 생성하지 않고 하나의 네인 액티비티 (WebViewActivity)안에서 WebViewClient와 WebBrowserClient를 불러와서 실행되게 하였다. 앱으로 띄우는 화면은 총 3개가 있을 것이다. 카메라 정보와 센서정보, 비튼조절정보이다. 카메라와 센서 정보는 화면을 띄우는 것만으로 충분하지만, 비튼

조절은 주소를 불리울 뿐 아니라 웹에서 카메라와 RC카 방향전환을 할 수 있었던 것과 동일한 기능을 할 수 있게끔 만들어 주어야 한다. 이를 실현하기 위해서는 웹의 화면 뿐 아니라 기능을 그대로 앱으로 가져와야 하고, 그 과정을 MyChromeClient와 WebBrowserClient로 2개의 클래스를 이용해 나타낸 것이다.

```
package com.example.webviewtest;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Message;
import android.webkit.JsResult;
import android.webkit.WebChromeClient;
import android.webkit.WebView;

public class MyChromeClient extends WebChromeClient {

    @Override
    public boolean onCreateWindow(WebView view, boolean dialog,
        boolean userGesture, Message resultMsg) {
        // TODO Auto-generated method stub
        return super.onCreateWindow(view, dialog, userGesture,
            resultMsg);
    }

    @Override
    public boolean onJsAlert(WebView view, String url, String
        message, final JsResult result) {

        // TODO Auto-generated method stub
        //return super.onJsAlert(view, url, message, result);
        new AlertDialog.Builder(view.getContext())
            .setTitle("알림")
            .setMessage(message)
    }
}
```

```

        .setPositiveButton(android.R.string.ok,
                new AlertDialog.OnClickListener(){
                    public void onClick(DialogInterface dialog,
int which) {
                        result.confirm();
                    }
                })
        .setCancelable(false)
        .create()
        .show();

    return true;
}

@Override
public boolean onJsConfirm(WebView view, String url, String
message,
final JsResult result) {
// TODO Auto-generated method stub
//return super.onJsConfirm(view, url, message, result);
new AlertDialog.Builder(view.getContext())
        .setTitle("알림")
        .setMessage(message)
        .setPositiveButton("네",
                new AlertDialog.OnClickListener(){
                    public void onClick(DialogInterface dialog,
int which) {
                        result.confirm();
                    }
                })
        .setNegativeButton("아니오",
                new AlertDialog.OnClickListener(){
                    public void onClick(DialogInterface dialog,
int which) {
                        result.cancel();
                    }
                })
}

```

```
        }
    })
    .setCancelable(false)
    .create()
    .show();
return true;
}

}
```

〈MyChromeClient 클래스 자바스크립트〉

```
package com.example.webviewtest;

import android.webkit.*;

class WebBrowserClient extends WebViewClient {
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
    }
}
```

〈WebBrowserClient 클래스 자바스크립트〉

```
package com.example.webviewtest;

import android.os.Bundle;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.webkit.*;

@SuppressLint("SetJavaScriptEnabled")
public class WebviewActivity extends Activity {
    WebView webView;
```

```
public void onCreate(Bundle savedInstanceState) {  
  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_webview);  
  
    webView = (WebView)findViewById(R.id.webView01);  
    WebSettings webSettings = webView.getSettings();  
  
    webSettings.setPluginState(WebSettings.PluginState.ON);  
    webSettings.setLayoutViewPort(true);  
    webSettings.setSupportZoom(true);  
    webSettings.setBuiltInZoomControls(true);  
    webSettings.setJavaScriptEnabled(true);  
    webSettings.setSupportMultipleWindows(true);  
  
    webSettings.setJavaScriptCanOpenWindowsAutomatically(true);  
    webView.setWebViewClient(new WebBrowserClient());  
    MyChromeClient myChromeClient=new MyChromeClient();  
    webView.setWebChromeClient(myChromeClient);  
    String url =  
"http://192.168.1.137:8080/javascript/simple.html";  
    webView.loadUrl(url);  
  
    webView = (WebView)findViewById(R.id.webView02);  
    WebSettings webSettings1 = webView.getSettings();  
  
    webSettings1.setPluginState(WebSettings.PluginState.ON);  
    webSettings1.setLayoutViewPort(true);  
  
    webSettings1.setJavaScriptEnabled(true);  
    webSettings1.setSupportMultipleWindows(true);
```

```

webSettings1.setJavaScriptCanOpenWindowsAutomatically(true);
    webView.setWebViewClient(new WebBrowserClient());
    MyChromeClient myChromeClient1=new MyChromeClient();
    webView.setWebChromeClient(myChromeClient1);
    String url1 = "http://192.168.1.137:8000/app/serial-monitor";
    webView.loadUrl(url1);

    webView = (WebView)findViewById(R.id.webView03);
    WebSettings webSettings2 = webView.getSettings();

    webSettings2.setPluginState(WebSettings.PluginState.ON);
    webSettings2.setLayoutDeviceViewPort(true);

    webSettings2.setJavaScriptEnabled(true);
    webSettings2.setSupportMultipleWindows(true);

    webSettings2.setJavaScriptCanOpenWindowsAutomatically(true);
    webView.setWebViewClient(new WebBrowserClient());
    MyChromeClient myChromeClient2=new MyChromeClient();
    webView.setWebChromeClient(myChromeClient2);
    String url2 = "http://192.168.1.137:8000/app/1group";
    webView.loadUrl(url2);
}

```

〈메인 액티비티 역할을 하는 WebViewActivity 클래스〉

2-2-3. 시스템 구성 도면

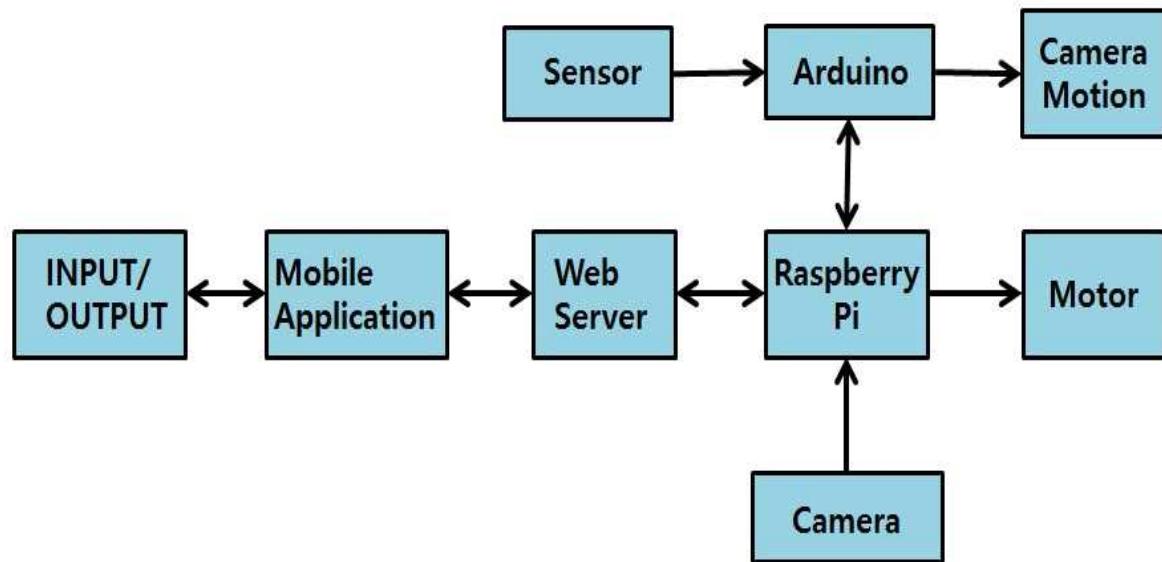


Fig 9. 시스템의 구성 도면

III. 시스템 제작 [합성, 제작]

3-1. 1학기 기본제작

| 세부계획 및 담당자 | 연구기간 | | | |
|----------------|------|---|---|---|
| | 3 | 4 | 5 | 6 |
| 서보모터 제어 및 센서 | 이정섭 | | | |
| 안드로이드 개발 | 박승진 | | | |
| 웹서버 구축 및 시스템구현 | 강동욱 | | | |
| 차체 디자인 및 강화 | 김종표 | | | |

Table 3. 1학기 기본제작 진행일정

1학기 기본 제작에서는 기술 선정 및 자료조사에 중점을 두고 IO와 웹서버, 안드로이드를 하나의 네트워크로 묶는 것에 집중하였다.

3-1-1. H/W 기본 설계

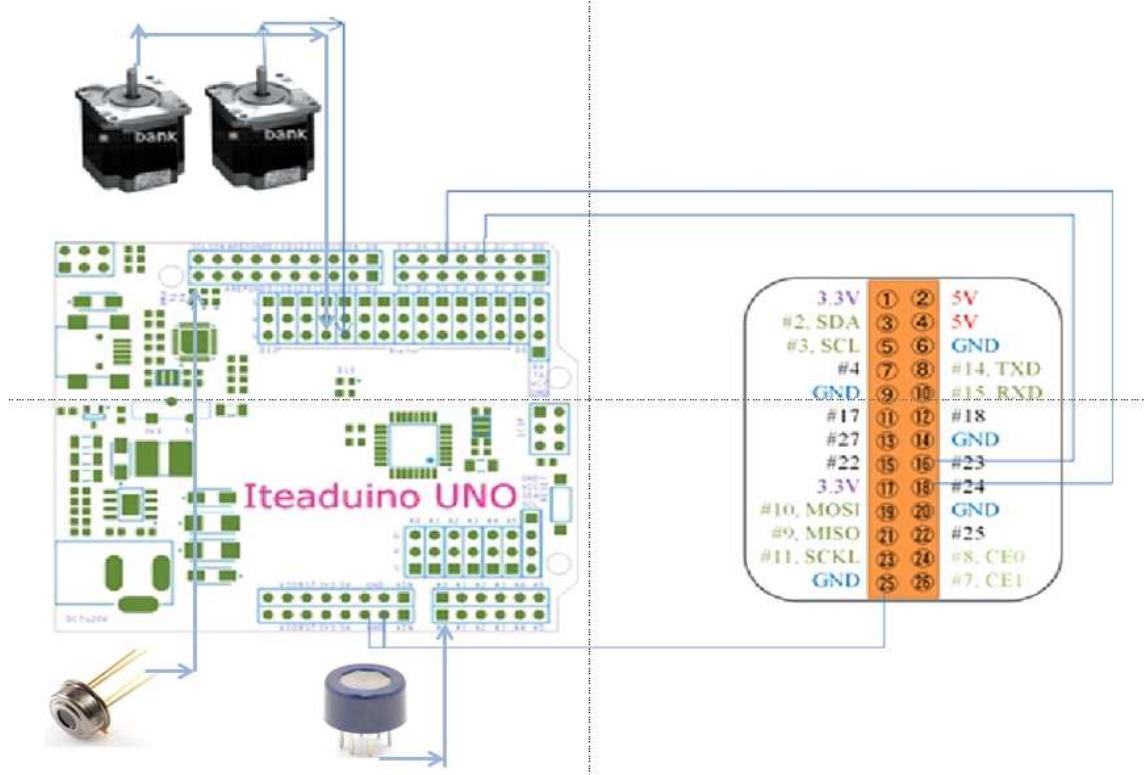


Fig 10. IIC와 웹서버 보드와의 연계

웹페이지에 카메라 상하 제어를 위한 신호를 받기 위해서 MCU보드와 웹서버 보드와의 연계를 실시하였다. 서보모터를 아두이노의 PWM 10,9번 단자에 연결하여 코딩을 하고, digital Pin 4,5번에 각각의 상/하, 좌/우 신호가 전달되게 하였다.

이 신호를 웹서버 보드의 GPIO단자 23,24번 Pin이 각각 받아서 Webiopi를 통하여 웹페이지에 신호가 갈 수 있도록 하였다.

센서의 경우 온도 센서와 일산화탄소 농도를 재는 센서를 각각 I2C, A0핀에 연결하였고 이 값을 시리얼 통신을 통해 라즈베리 파이로 보내는 방법을 사용하였다.

두 보드를 연결하는 방법으로, 직접적으로 usb선을 연결하는 방법을 사용하였다. 웹서버 보드의 GPIO(입출력핀)을 통해 나오는 신호를 보터제어 보드의 입력핀으로 연결하였다. 웹서버를 구축하는 라즈베리파이 보드에서 보터제어를 구현하는 아두이노 보드로 신호를 보내려면 사용전압이 다른 관계로 5V를 3.3V로 낮춰 필요가 있었지만, 아두이노의 업데이트 버전인 Iteaduino가 운영전압을 5V, 3.3V 둘 다 가질 수 있게 설계가 되었기 때문에 별도로 회로를 추가할 필요가 없었다.

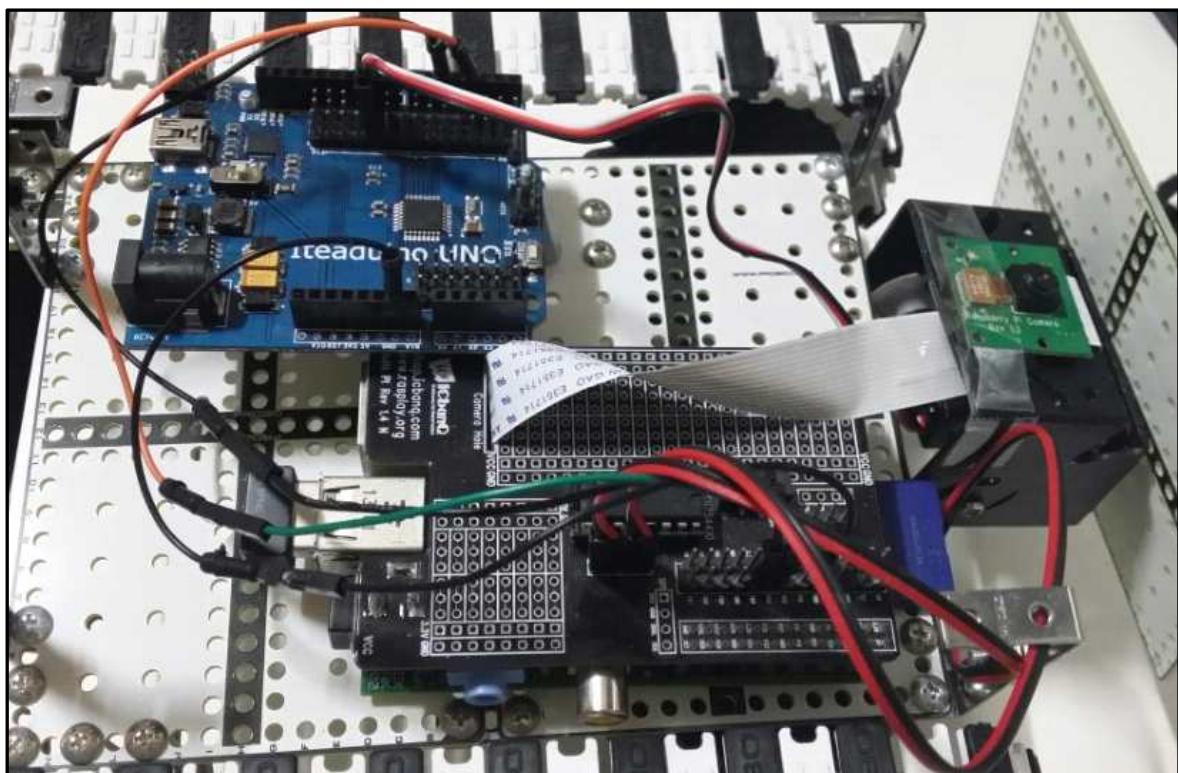


photo 8. 웹서비 보드와 MCI와의 연결 모습

3-1-2. S/W 기본 설계

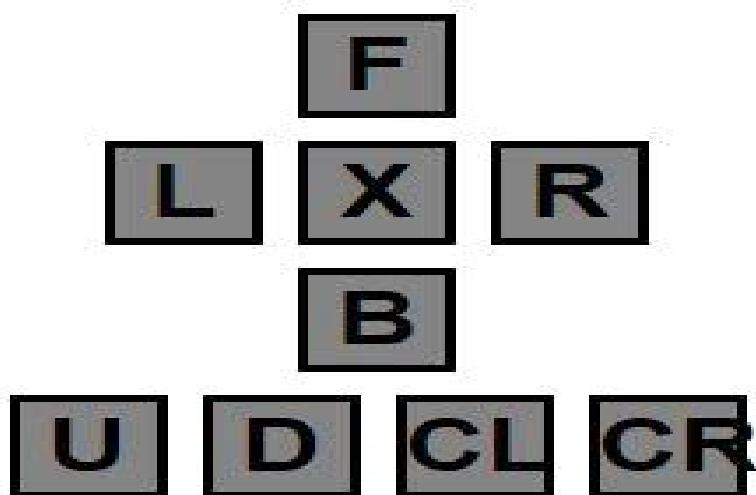


photo 9. RC카 제어신호 및 카메라 신호 제어신호

다음은 MCI와 웹서비 보드의 연계를 통하여 Webiopi를 이용함으로써 웹페이지에 구현한 모습이다.

3-2-1. 2학기 보충제작 및 Feedback 제작

| 세부계획 및 담당자 | 연구기간 | 7 | | 8 | | 9 | | 10 | |
|----------------|------|---|---|---|----|----|----|----|----|
| | | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 서보모터 제어 및 센서 | 이정섭 | | | | | | | | |
| 안드로이드 개발 | 박승진 | | | | | | | | |
| 웹서버 구축 및 시스템구현 | 강동욱 | | | | | | | | |
| 차체 디자인 및 강화 | 김종표 | | | | | | | | |

Table 4. 2학기 보충제작 및 진행일정

2학기 보충제작 기간에는 어플리케이션을 개발하는 것과 앱, MCU, 웹서비스를 하나의 네트워크로 묶는 것과 완전 무선환경을 구축하는 것을 목표로 하였다.

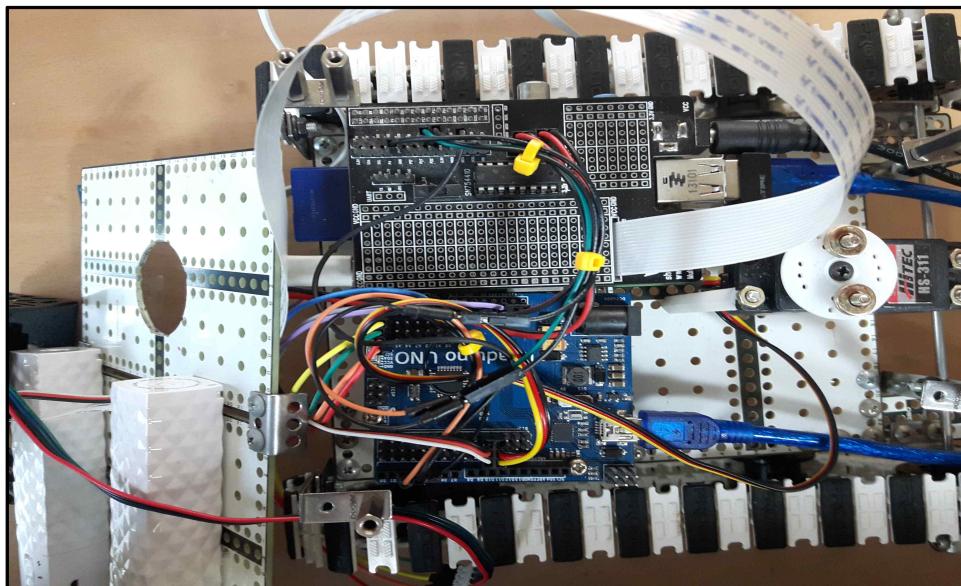


photo 10. 내부 HW 총 결합 모습

3.2.2. S/W 보충 설계

(1) 웹서버 무선 환경 구축

Raspberry Pi 보드에 리눅스 환경을 설치하고 Python 프로그램을 통해 웹서버를 구축하였다. 보드에 무선 공유기를 연결하고 포트포워딩 기능을 통해 스마트폰에서의

집속이 가능하게 하였다. 포트포워딩은 설정한 IP주소와 특정포트로 외부와 통신이 가능하게 해주는 서비스이다. 사전에 포트를 열어줘야하는 웹서버 보드의 고정아이피를 설정해 주는 것이 필수이다. 즉, 이것은 모바일 환경에서의 영상 전송과 보터제어 신호를 웹서버와 이어주는 역할을 한다.

단점으로는 무선공유기의 네트워크 범위 안에 있을 때만 딜레이가 0.5초 미만으로 자연스럽게 동작한다는 것이다. 만약 범위 밖에 있을 시에는 외부wifi나 3G/4G를 통하여 집속하여야 하는데, 이런 경우에는 딜레이가 내부 네트워크로 집속 했을 때보다 1~2초 이상의 딜레이가 자연된다.



photo 11. 앱으로 보는 영상 및 제어신호 확인

3 2 3. 외형 제작

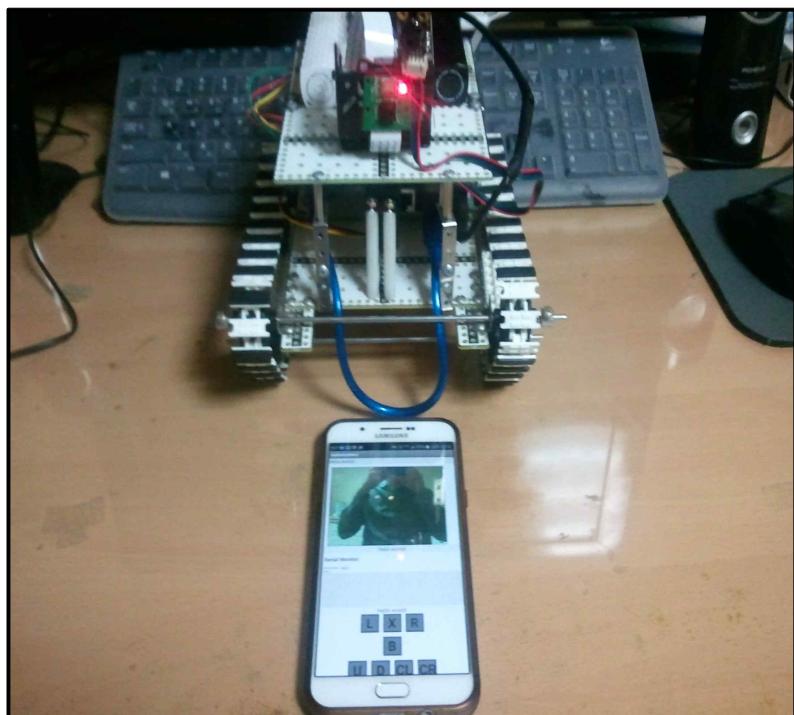


photo 12. RC Car측 시스템과 Client 시스템의 외형 모습

3-3. 사용 부품 명세서

사용 부품 명세서는 최종 완성된 작품에 들어있는 부품을 기준으로 작성하였습니다.

| 품명 | 수량 | 단가 | 금액 |
|-----------------------|----|--------|--------|
| TR B200 캐터필러 SET | 1 | 71,500 | 74,000 |
| SN754410NF | 1 | 3,170 | 3,170 |
| 실피 와이어 F/F형 10cm | 50 | 10,000 | 11,000 |
| 브레드보드 실피 와이어 M/M | 60 | 6,900 | 7,590 |
| 소형 시보모터 S형 | 2 | 12,000 | 24,000 |
| 2축 서보모터 구조돌 | 1 | 10,000 | 11,000 |
| 12mm 핀헤더 40핀 2.54mm간격 | 1 | 200 | 220 |
| ATmega328 tenduino | 1 | 37,400 | 37,400 |
| 라즈베리파이 카네라보드 | 1 | 38,000 | 41,800 |
| 메모리 카드 SD 8G | 1 | 8,500 | 9,350 |
| 라즈베리파이 본체 | 1 | 48,500 | 53,350 |
| MQ 7일산화탄소 센서 | 1 | 2,420 | 2,420 |
| 비 접촉식 온도센서 | 1 | 22,000 | 22,000 |
| PowerBank 보조배터리 | 5 | 20,000 | 20,000 |
| 밀티파이 V1.4 | 1 | 7,000 | 7,700 |

Table 5. 사용 부품 명세서

IV. 시스템의 동작 검증

4-1. H/W, S/W 동작 검증

(1) 웹서버 보드

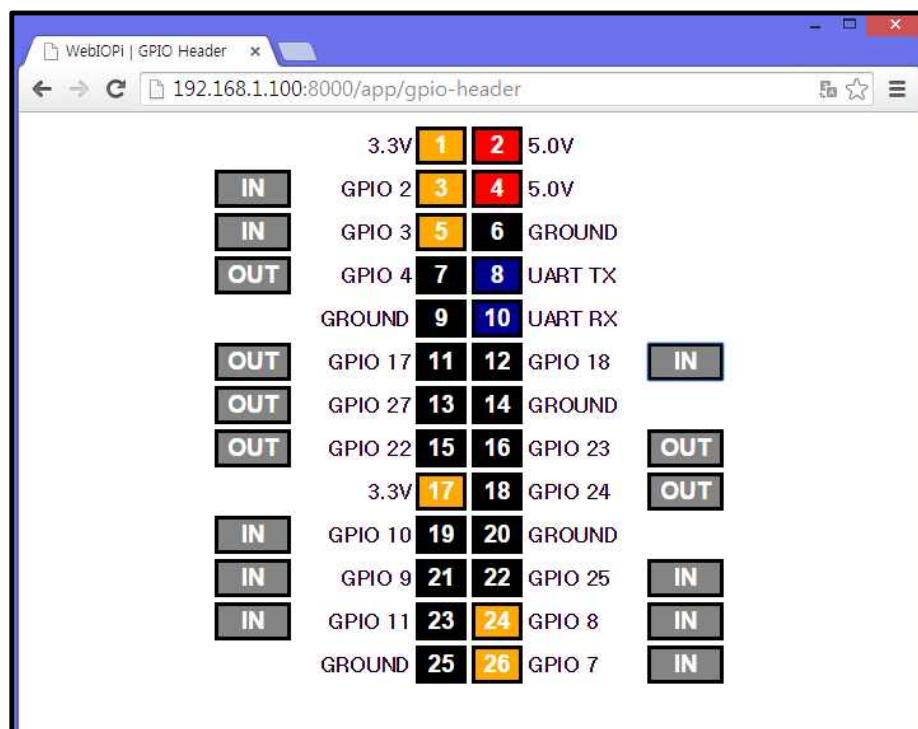


photo 13. 웹서버 GPIO단자 4, 17, 27, 22, 23, 24 각 포드 정상 출력 확인

| Webiopi를 기반으로 해서 웹페이지에 보여진 GPIO 출력 단자들을 확인할 수가 있다.
그림과 같이 4, 17, 22, 27 포트인 RC카 제어부와 23, 24 포트인 카메라 상/하
제어부가 정상적으로 출력되고 있음을 볼 수 있다.

|2) 모터제어 모드

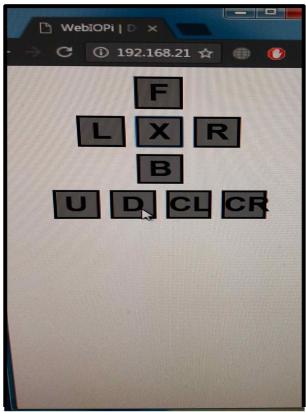


photo 14. 하(웹페이지)

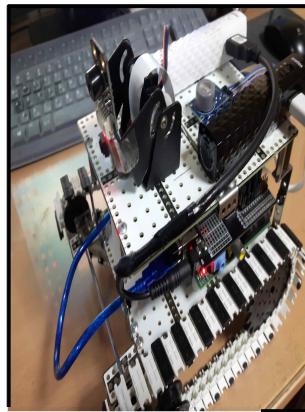


photo 15. 하

웹페이지에서 카메라의 Down버튼을 눌렀을 때, 카메라가 아래방향으로 움직이는 것을
볼 수 있다.

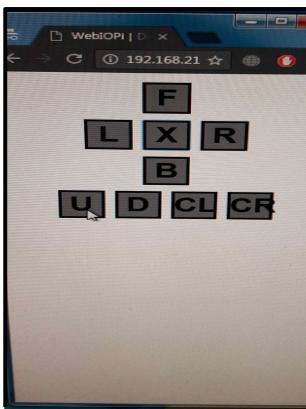


photo 16. 상(웹페이지)

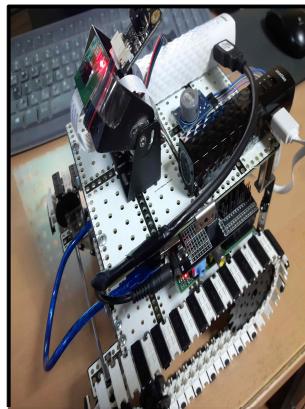


photo 17. 상

웹페이지에서 카메라의 Up버튼을 눌렀을 때, 카메라가 위방향으로 움직이는 것을 볼
수 있다.

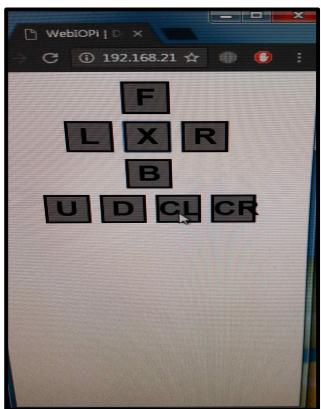


photo 18. 좌(웹페이지)

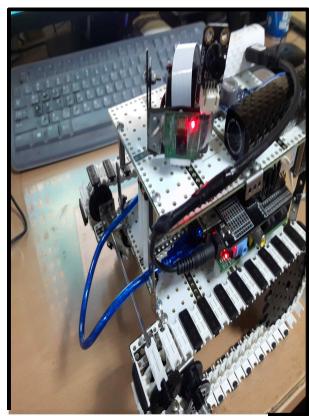


photo 19. 좌

웹페이지에서 카메라의 CL버튼을 눌렀을 때, 카메라가 왼쪽으로 움직이는 것을 볼 수 있다.

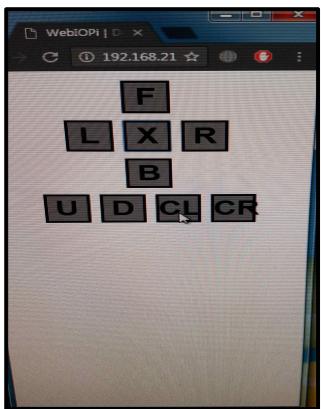


photo 20. 우(웹페이지)



photo 21. 우

웹페이지에서 카메라의 CR버튼을 눌렀을 때, 카메라가 오른쪽으로 움직이는 것을 볼 수 있다.

(3) 앱 어플리케이션

사진에서 보는 것과 같이 현장에 있는 카메라의 영상정보가 어플리케이션의 영상 부분에 실시간으로 출력이 되는 것을 볼 수 있고, 현장에 있는 사물을 제어하기 위한 상하좌우 제어버튼이 화면 아랫부분에 제대로 위치되어 있는 것을 볼 수 있다.



photo 22. 앱 작동 모습

- 카메라 영상 전송

영상 전송 부분에서는 실시간 스트리밍을 목표로 하고 있으며 안정적인 영상 전송과 낮은 지연율을 중점적으로 두었다. 이것을 이루어내기 위해 다양한 영상 제어 프로그램을 사용하고 시험하였다.

먼저 `mjpg-streamer`라는 프로그램은 카메라의 영상 지연이 대략 1초 정도로 낮은 지연율을 보였다. (해상도 1280×720 10프레임) 웹페이지를 통해서 볼 수 있으며 모바일 어플리케이션에서 `WebView`를 이용하여 영상 화면을 볼 수 있었다. 부하가 적고 속도가 빠르기 때문에 안정적으로 영상 전송이 가능하고 다른 프로그램들보다 낮은 지연율로 영상 전송이 이루어져 최종적으로 선택하였다.

`gstreamer`는 지연율이 1~2초 정도로 괜찮은 편이지만 부하가 심하여 웹서버 보드에 깊은 문제를 일으켰다. (1280×720 10프레임) 화질은 좋았지만 안정적이지 않았다. `ffmpeg`는 대략 2~5초의 지연율을 보였다. (1280×720 10프레임) 부하가 적지만 지연율이 높고 화질이 좋지 않았다.

`vlc`는 3초 정도의 지연율을 보였다. (1280×720 10프레임) 다만 이 프로그램은 사용하기가 쉽고 `rtsp`를 통한 영상 전송이 용이한 부분이 있었다. 마지막으로 `motion`은 다양하게 응용할 수 있는 부분이 많지만 프로그램 중 가장 높은 지연율을 보였다.

최종적으로 안드로이드 모바일 환경에서 영상을 출력하기 위해 `mjpg-streamer` 방식을 택하게 되었다.

4-2. 완성된 전체 시스템의 목표 성능에 대한 평가항목 및 평가 결과

| No | 평가항목 | 평가결과 |
|----|----------------------|------|
| 1 | 웹페이지에서의 영상 및 제어신호 확인 | o |
| 2 | 영상 딜레이 및 출력 화질 | o |
| 3 | 보드 제어 & 센서부 출력 | o |
| 4 | 안드로이드 앱 어플리케이션 구현 | o |
| 5 | 무선 환경 구축 | △ |

Table 6. 목표 성능에 대한 평가항목 및 평가 결과

작품 초기에 목표로 했던 부분들에 대해서 근접한 결과를 얻어낼 수 있었다.

보디제어 보드의 신호와 카메라 영상신호, 센서 값, RC카 제어 신호가 웹페이지 공간에서 문제없이 합쳐졌다.

영상 또한 라즈베리제단에서 라즈베리파이 보드에 최적화된 카메라보드를 출시함으로써 훌륭한 화질을 보여줬다. 화면을 실시간으로 확인하는 데 복석이 있는 스트리밍 방식으로써 `jpeg-streamer`를 채택함으로써 1초미만의 비교적 자연스러운 실시간 스트리밍 서비스를 보여줬다.

안드로이드 앱 어플리케이션은 현재 아주 기본적인 디자인을 한 것이지만, 좀 더 공부하고 연습한다면 더 훌륭한 앱을 만들 수 있을 거라고 생각한다.

다만, 문제점이 있다면 이번 프로젝트를 수행하는데 있어서 무선 환경을 구축하는 것이 가장 핵심이고 중요한 사항이었는데, 이 부분에서 한계가 있었다. 내부 네트워크에서는 원활한 작동을 보이지만, 외부 네트워크로 접속할 경우에는 기존의 내부 네트워크에서의 반응 속도보다 2배 정도의 딜레이를 보였다. 이것이 가장 아쉬운 점이다.

V. 결 과

5-1. 완성도 시술

프로젝트 초기에 목표로 했던 것과 같이 모바일 어플리케이션을 통해 카메라와 RC 카를 버튼으로 움직여 제어하고 영상을 재생하는 부분까지 성공하였다. 제어와 통신, 센서, 외부적 견고성이라는 작품제작 전에 목표로 하였던 초기의 목표와 거의 완벽히 구현하였다. 기본적으로 화재 탐사용을 목적으로 만들어진 기능을 가진 작품이지만 많은 발전 가능성이 있다고 생각하고 다양한 용도로 사용될 수 있다고 생각한다. 카메라 영상은 다양한 프로그램으로 실현해보고 가장 낮은 자연광과 안정적인 영상 스트리밍이 가능한 `mjpg-streamer`를 사용하였지만 카메라를 다양한 용도로 활용하는 측면을 생각할 때 `motion`이나 `opencv` 등의 프로그램을 사용하는 것도 나을 것 같다.

지금까지 개발한 작품에서 그치지 않고 조금 더 나아가 화재현장을 넘어 다양한 분야에 쓰임 받는 작품이 될 수 있을 것으로 기대한다.

5-2. 완성된 작품의 주요 활용분야

기본적으로 이 작품은 IoT 방식으로 제작되었기 때문에 누구나 간편하게 손안의 스마트 폰으로 자신이 필요한 곳의 물체를 제어할 수 있으므로 에너지 절약과 고효율의 입두치리가 가능하다. 던지는 화재 탐사용이라는 용도로 쓰임받기 이전에 이 작품은 전원 온오프 시스템과의 연계 등을 통해 지전력으로 시스템을 구축할 수 있으므로 대기전력 절감과 실생활의 편의또한 더할 수 있다. 이 작품은 화재 현장에 최적화된 것으로, 내부에 장착된 큐리와 대기 CO농도센서 등의 특수함을 가지고 있으므로 사람이 갈 수 없는 화재현장을 스마트폰으로 직접 제어하여 정보를 얻는 것을 가능케 할 것이다.

5-3. 주요 긴도사항 시술 (보완점)

조금 더 보완할 점으로 공유기를 통해 무선통신을 할 때 DDNS 기능이 동작되지 않는 문제가 있었다. DDNS는 유선으로 들어오는 공인 IP에서만 동작하기 때문에 작품을 시현할 때 유선으로 공유기를 연결해야 한다. 하지만 무선공유기를 사용할 때 포드포워딩 기능을 통해서 무선 환경을 구축할 수 있었지만, 내부 네트워크를 벗어나서 외부 네트워크에서 접속 시엔 딜레이가 신한 것을 볼 수 있었다. 이것을 해결하기 위해서는 DDNS기능이 필수적인데, 이 문제를 위해서 도메인 서비스를 인터넷에서 신청하거나 공인 IP를 제공하는 프로그램 등을 사용해야 될 것으로 보인다.

또한 웹 서비스를 이용한 작품이기 때문에 보안이 취약한 문제가 있다. 이 부분은 작품에서는 특정 포드를 사용하여 위험도를 낮추는 방법만 사용하고 있었지만 좀 더 나은 보안대책이 필요한 것 같다.

영상은 스트리밍 시 웹서비스에 부하가 있기 때문에 동시에 접속하여 영상을 보려고 하면 웹서비스 보드가 다운되는 경우가 있었다. 스트리밍 서비스로 경유해서 영상을 전송하거나 NAS 서버를 이용하는 방법 등이 해결책으로 모인다.

참 고 문 헌

민경식, “사물인터넷(Internet of Things)”, 한국인터넷진흥원, 인터넷 & 시큐리티
이슈, 제 2012-6호.

유동근, 박정민, “Android API bible”, 정보문화사

김태호, “Kunny's Android from scratch”, 인사이드

박혜란, “Master of android apps programming”, 웅진씽크빅

Oehlman, Damon, “Pro Android web apps”, 갈벗

이두진, “The WebApp Programming with HTML5 and PhoneGap”, 파씨북

채진욱, “Arduino for interactive music”, 인사이드

Margolis, Michael, “Arduino cookbook”, 제이펍

김정윤, 김경, “라즈베리 파이 스마트 라이프”, 디지털북스

사이먼 몽크, “파이썬으로 시작하는 라즈베리 파이 프로그래밍”, 제이펍

참고 사이트

<http://www.arduino.cc>

<http://www.raspberrypi.org>

<http://www.python.org>

<http://code.google.com/p/webiopi>

<http://cafe.naver.com/pipe>

감사의 글

본 연구는 서울과학기술대학교 전자it미디어공학과에서 이루어졌으며 이 프로젝트가 성공적으로 완료될 수 있도록 학문적 지도는 물론 진행 과정에서 생기는 문제점, 발표 등 많은 부분에서 깊은 관심으로 지도해 주신 이예훈 교수님께 먼저 감사의 말씀을 드립니다.

프로젝트의 주제 선정부터 제작과정까지 많은 부분에서 시행착오와 기술적 제약 등 의 문제점에 직면하여 진행이 늦어짐에도 격려와 조언으로 다른 분야 정보까지 정리하여 도움을 주시라고 온갖 노력을 기울여 주신 문도성 교수님 덕분에 계획된 일정으로 작품을 마무리 할 수 있었습니다.

또한, 학과의 여러 친구들과 함께 상호 협동적으로 이론적 지식을 공유함으로써 좀 더 폭넓은 활용을 통하여 프로젝트의 질을 향상시킬 수 있었으며, 졸업작품 프로젝트를 통해 이론으로 배우고 실제로 경험해 볼 수 없었던 부분을 직접 설계하고 제작함으로써 배운 외 쪽을 넓힐 수 있는 좋은 계기였습니다.

이러한 지식과 경험, 배움의 노력을 바탕으로 졸업 후 직장전선에서도 서울과학기술대학교 축신 엔지니어로써 많은 양을 벌치고 학교의 위상을 높일 수 있는 인재가 되도록 하겠습니다.

감사합니다.

2016년 10월 19일

06107804 김종표

11384301 강동욱

13184354 박승진

13184415 이정섭