개발완료보고서 [스마트 홈]

0. 작성 시 주의사항

※아래의 작성 양식(제출분량, 폰트, 크기, 줄 간격 등)을 미준수 시 서류 평가의 감점요인됨

※ 제출 분량 : A4 용지 상세내용 포함 30 page 이내
※ 작성 양식 (폰트 : 맑은 고딕 / 폰트 크기 : 10pt / 자간 : 0% / 장평 : 100% / 줄 간격 : 130%)
※ 제출 포맷 : pdf

1. 팀 정보

팀명	Voyager	팀장	김성원
팀원	문성신		

2. 개발완료보고서

0. 작품명 : RIDOS[라이도스, Real-type Integrated Device Operation System]

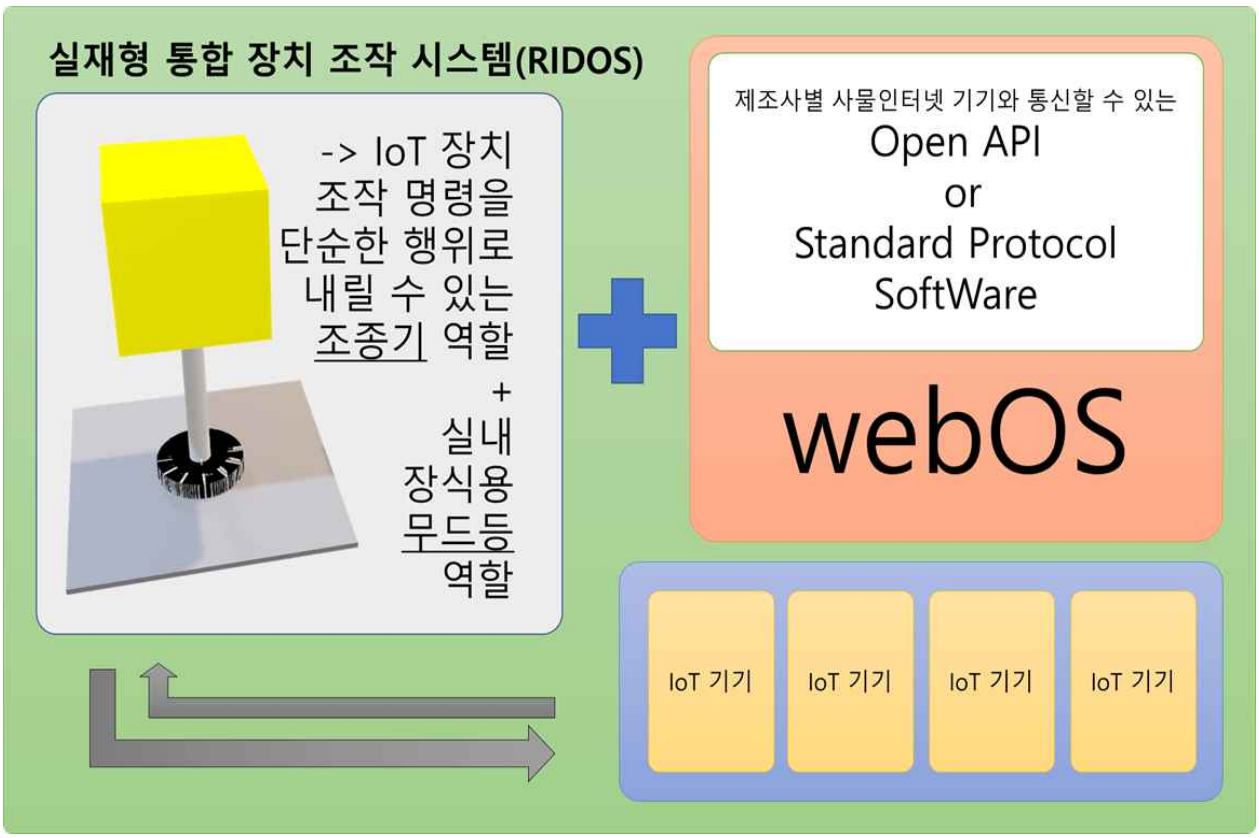
1. 개요

1.1. 작품 개요

- Real-type Integrated Device Operation System [실재형 통합 장치 조작 시스템, RIDOS(라이도스)]

- **요약** : 다양한 종류의 사물인터넷 기기들을 조작함에 있어서 실재적이면서 사용자 친화적인 인터페이스를 제공하는 시스템.

- 사물인터넷 기기를 조작하는 전용 단말기기로써 단순한 사용법을 통해 직관적인 사용자 경험을 제공할 수 있도록 하는 장치이다.



<작품 개괄도>

1.2. 개발 목표

- 개발 내용은 다양한 사물인터넷 기기들을 통합적으로 제어할 수 있고, 이용자들에게 실재형 사용자 경험을 제공하는 하나의 조작 시스템을 개발하는 것이다.

- 그에 따른 **초기 개발 목표**는 '집 안에 설치해두고 사용할 수 있는, 복잡하지 않고 단순한 행위만으로 IoT 기기들을 조작할 수 있는 시스템'이었으며, 작동 과정 설명은 아래와 같았다.

<조작법 설명 및 비교>

- 달성한 초기 개발 목표 :

> 다양한 사물인터넷 기기와 연동하여 그 기기들을 통합적으로 관리할 수 있는 하나의 사물인터넷 장치를 개발하는 것이 목표이다.

> RIDOS는 직관적이면서 단순해야 한다. 그렇기 때문에 키보드 등과 같은 사용자가 정보 입력 방식을 암기해야 하는 입력 매체는 조작에 적절하지 않다.

> 단순히 통합 조작 장치를 돌리거나 누르는 것으로 연동된 모든 사물인터넷 장치와 정보를 주고받을 수 있어야 한다.

> RIDOS는 연동된 장치 중 '어느 것을 조작할 것인지 선택하는 행위'와 그 장치를 '어떻게 조작할 것인지 선택하는 행위'를 정의하고, 그 행위들을 RIDOS에 있는 센서들이나 스위치를 통해 입력받아야 한다.

> 그리고 그 장치가 현재 어떠한 상태인지(미세먼지 센서의 경우 미세먼지 농도에 대한 정보, 전등의 경우 켜져 있는지 아닌지에 대한 정보)를 사용자에게 표시할 수 있는 출력장치(EX : 디스플레이) 또한 갖추어야 한다.

> 미려한 디자인의 실내장식 제품의 역할(무드등)도 갖추고 있도록 하여 동일한 기능을 할 수 있는 리모컨 및 스마트폰과 다른 차별성을 지닐 수 있도록 한다.

- 달성에 실패한 초기 개발 목표 및 이유 :

> RIDOS는 다양한 기기와의 연동 호환성을 갖추어야 한다. 또한, 다양한 방식으로 네트워크에 연결될 수 있어야 한다. ->> 표준 iotivity를 이용하여 연결 호환성과 안정성을 갖추고 기기 연결 방식 또한 다원화하려 시도하였으나 시간이 부족하여 iotivity를 사용한 개발에는 실패하였다.

정량적 목표

항목	목표
최대 기기 연동가능 수	7개의 장치
기기 작동 딜레이	0.5ms 이내
인디케이터 디스플레이	5인치 디스플레이 한 개
배터리 사용시 러닝타임	최소 3시간 이상

<달성된 정량적 세부 목표>

2. 개발 환경 설명 (최대한 자세하게 기술)

2.1. Hardware 구성

- 라즈베리파이가 삽입된 무드등 형태의 하드웨어 디바이스 및 각종 간이 사물인터넷 장치

2.2. Software 구성

- 하드웨어 제어 신호를 수집하며 서버의 기능도 수행하는 네이티브 서비스, main.cpp
- 서비스와 통신하여 LCD 화면에 정보를 표시할 웹 앱, index.html
- 간이 IoT 장치들용 클라이언트 소프트웨어

2.3. Software 설계도 (흐름도 및 클래스 다이어그램 등 (개발언어에 따라 선택))

-

2.4. Software 기능 (필요 시 알고리즘 설명 포함)

- 클라이언트 기기 연결 요청 수락
- 하드웨어적 제어 신호 수집
- 수집된 제어 신호 소켓통신으로 전송
- 클라이언트 기기 상태 수신 및 LCD 표기

2.5. UI 사용법 (구성 및 설명)

- 사용자 인터페이스는 RIDOS기기 상단에 부착되어있는 5인치 LCD 디스플레이에 연결된 사물인터넷 장치들의 상태 표시와 조작법 설명 등이 표기된다.

2.6. 개발환경 (언어, Tool, 사용시스템 등)

- OS : Windows 10 Pro, Elementary OS
- Target OS : webOS(RIDOS), Raspbian(클라이언트 기기)
- Programming Language : C, C++, Python3
- Tool : Bitbake, scons, gcc, clang, CLI

3. 개발 프로그램 설명 (최대한 자세하게 기술)

3.1. 파일 구성

- com.webos.service.gpio.control/main.cpp -> RIDOS 네이티브 서비스
- com.ridos.app/index.html -> RIDOS 웹 앱
- type_gasvalve.c -> 간이 IoT 가스 밸브
- type_rgblight.c -> 간이 IoT 삼색 등
- type_patternlight.c -> 간이 IoT 정원 등
- type_autodoor.c -> 간이 IoT 자동문
- type_airconditioner.c -> 간이 IoT 에어컨
- type_tv.py -> 간이 IoT TV

3.2. 함수별 기능

- void letLEDWork(int const option); -> 무드등 LED 색 변경
- void * gpioThread(void * data); -> 무드등 회전상태 파악용 스레드
- void * clickCheckThread(void * data); -> 오른쪽, 왼쪽 클릭 상태 파악용 스레드
- static bool deviceStatus3(LSHandle * sh, LSMessage * message, void * ctx); -> 웹 앱에 클라이언트 디바이스 상태 전달
- void * acceptThread(void * data); -> RIDOS 장치가 전원이 켜진 후로 계속 클라이언트를 연결하거나 연결 해제하도록 함
- int main(int argc, char * argv[]); -> OS에 서비스를 등록하고, TCP 서버를 실행하고, 스레드를 실행하고, 웹 앱을 켬

3.3. 주요 함수의 흐름도


Gpio 스레드
Pull Down 모드로 핀 설정
로터리 스위치 디지털 핀 값 읽기
값이 변경됐을 경우 무드등 색 변경
clickCheckThread가 생성한
왼쪽 및 오른쪽 버튼 입력 여부 확인
버튼 눌러졌을 시 클라이언트 기기로 제어 명령 전송
클라이언트 기기 상태 수신
반복 수행
clickCheck 스레드
Pull Down 모드로 핀 설정
왼쪽 및 오른쪽 스위치 디지털 핀 값 읽기
값이 변경됐을 경우
gpio스레드가 읽을 버튼 입력 여부 플래그 확성화
반복 수행
accept 스레드
스택으로부터 클라이언트 accept
클라이언트 기기 상태 정보 수신
수신이 불가능한 경우 클라이언트 연결 해제
반복 수행
연결이 해제된 클라이언트가 있으면 다시 accept

3.4. 기술적 차별성

- 네트워크 사정으로 인해 강제로 연결이 끊어지는 경우에 클라이언트가 응답이 없으면 연결을 해제하고 재 연결을 시도하게 함으로써 오류 가능성을 줄였다.

4. 개발 중 장애요인과 해결방안

- webOS 전체 빌드 과정에서 해결법을 찾기가 힘든 오류들이 계속적으로 다수 발생하여 해당 오류들을 수정하고 다시 빌드를 진행하는 작업을 약 3주간 진행하였으나 결국 전체 빌드는 불가능하다고 판단, 개별 컴포넌트만 빌드하는 방향으로 전환하였다.

- bitbake의 사용이 처음이라 네이티브 서비스에 이용되어야 하는 wiringPi 라이브러리를 빌드 프로그램의 링커에 추가하느라 장애를 겪었다. 또한, 개발 도중에 wiringPi 라이브러리의 git 서버가 닫히는 일이 발생하면서 빌드에 오류가 발생하는 것을 확인한 후 다른 mirror git으로 전환하기도 하였다.

- 구현해야 하는 기능의 특성상 멀티스레드를 활용해야 되는 상황이었으나, bitbake로 빌드 시에 pthread 라이브러리를 추가하는 방법에 대한 자료가 없어 시간이 많이 소요되었다. 결국 bb 파일에 'LDFLAGS= "-lpthread"'를 추가하는 방법으로 해결하였다.

- iotivity를 활용하여 사물인터넷 시스템을 개발하려 했으나 scons를 통한 webos용 iotivity 빌드에 끝내 성공하지 못하였고, scons를 사용하지 않고 다른 방법으로 빌드하기 위해 iotivity 소스를 수정하려 시도했으나 주석이 부실하고 인터넷에 올려진 도큐먼트도 부족하여 난항을 겪었다. 많은 시간을 소모한 끝에 iotivity 프레임워크를 사용하지 않고 TCP/IP 소켓 통신으로 전환하기로 하였다.

5. 개발결과물의 차별성

- 개발 완료된 기기는 기획단계부터 단순하게 기기를 조작하는 방법에 대해 많은 고려가 진행되었다. 오른쪽이나 왼쪽으로 꺾음으로써 기기에 신호를 보낼 수 있으며, 무드등 부분을 시계 방향, 반대방향으로 돌림으로써 현재 선택된 기기를 변경할 수 있다.

- 이처럼 단순한 방법으로 사물 인터넷기기들을 조작할 수 있는 실재형 사물인터넷 기기 조작 전용 장치는 너무 많은 용도를 지니는 스마트폰이나 음성 인식의 불안감을 지울 수 없는 인공지능 스피커와는 다른 방향성을 지닌다.

- 또한, 실내 미려한 디자인과 실내 무드등으로 활용할 수 있다는 점에서 충분한 가치를 지닌다고 생각된다.

6. 단계별 개발계획 및 실제 참여인원 및 업무 분장

단계별 개발계획 :

1. 초기 형태 제작 (7월)	- 설계했던 대로 초기 물리 모델을 실제로 제작한다.
2. webOS build (7월)	- webos 이미지 빌드 방법 및 bitbake 사용법 스터디 한다.
3. native service & web app 제작 (8월)	- RIDOS 기기에 사용될 서비스와 웹앱을 제작한다.
4. iotivity 스터디 (8월)	- iotivity에 관한 내용을 스터디한 뒤 프로그래밍한다.
5. IoT 클라이언트 제작 (9월)	- RIDOS 기기에 연결하여 조종할 클라이언트 디바이스들을 제작한다.
6. 최종 서류 작성 (9월)	- 최종 완성본에 대한 서류를 작성한다.

- 업무 분담 :

팀장(김성원)	- 초기 설계대로 3D 프린팅 등의 방식을 이용하여 물리 모델 제작 - 센서 입력값 처리 알고리즘 프로그래밍 - 사물인터넷 기기 조작 Open API를 활용하여 통합 조작 소프트웨어 제작 - API가 공개되지 않은 사물인터넷 장치들에 입력값을 보내는 방법에 대한 연구
팀원(문성신)	- 계획서 및 각종 서류 작성 - 디스플레이 장치 작동 부분 프로그래밍(webOS 활용 인디케이터 모델 모색) - 사물인터넷 기기 조작 Open API를 활용하여 통합 조작 소프트웨어 제작 - 네이티브 서비스 및 웹 앱 제작 - 정상 작동 확인 및 피드백 시 - RIDOS와 연동될 간단한 사물인터넷 기기 제작

- 실제 참여인원 :

1. 초기 형태 제작 (7월)	- 김성원(팀장)
2. webOS build (7월)	- 문성신(팀원)
3. native service & web app 제작 (8월)	- 문성신(팀원)
4. iotivity 스터디 (8월)	- 문성신(팀원), 김성원(팀장)
5. IoT 클라이언트 제작 (9월)	- 문성신(팀원)
6. 최종 서류 작성 (9월)	- 문성신(팀원)