**목차**

**1. 서론**

**1.1 목적과 배경**

**1.2 계획**

**2. 아두이노 응용에 관련된 이론**

**2.1 아두이노**

**(1) 아두이노 보드의 이해**

**(2) 아두이노 보드의 장점**

**(3) 아두이노 보드의 종류**

**2.2 릴레이 모듈**

**(1) 릴레이 모듈의 이해**

**(2) 릴레이 모듈의 이론**

**3. 스마트 멀티 탭의 구성**

**3.1 물리적 구조**

**(1) 물리 구성도**

**(2) 아두이노 보드**

**(3) 릴레이 보드**

**(4) 어댑터**

**3.2 논리적 구조**

**(1) 내부 수행 순서도**

**(2) 릴레이 설정 소스코드**

**(3) 릴레이 제어 소스코드**

**4. 하드웨어 조립 및 제작**

**4.1 하드웨어 구성요소**

**4.2 조립 및 제작 순서**

**4.3 하드웨어 제어**

**5. 소프트웨어 구성**

**5.1 아두이노에 대한 소스코드**

**6. 멀티탭 성능 테스트**

**7. 결론**

**7.1 문제점과 해결책**

**(1) 블루투스 기능**

**(2) 어플 이용**

**7.2 기대효과**

**1. 서론**

**1.1 목적과 배경**

IoT기술의 부상과 함께 스마트 홈 서비스가 함께 각광을 받기 시작했다. 스마트 홈 서비스는 IoT를 기반으로 여러 형태의 자동화 서비스를 제공하는 새로운 주거형태를 말한다. 가스 원격제어 냉난방 제어, 조명 제어 등이 가능하며, 사용자와 집 안의 여러 가정 제품들과의 소통이 가능하다. 따라서 사용자가 집안에 있지 않아도 자동으로 작동이 가능한 다양한 서비스들이 스마트 홈에는 포함되어있다. 현재 LG U+의 IoT@home, KT의 기가 IoT, 애플의 홈 키트라는 사업이 진행 중이다. 이러한 스마트 홈 서비스의 사업의 발전 가능성을 보아 아두이노를 응용하여 멀티탭을 제어를 할 수 있는 기능을 제작하게 되었다.

**1.2 계획**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **프로젝트 계획** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **아두이노 특징 이해** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **아두이노 응용 기술 확인** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **프로그램 설계 및 회로도 설계** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **제품 구성 및 제작** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **작동원리 이해** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **아두이노 프로그램** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **제품 테스트 및 오류 수정** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **관련학습 이행** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**2. 아두이노 응용에 관련된 이론**

**2.1 아두이노**

**(1) 아두이노의 정의**

아두이노는 오픈 소스를 기반으로 한 단일 보드 마이크로컨트롤러로 완성된 보드(상품)와 관련 개발 도구 및 환경을 말한다.

아두이노는 다수의 스위치나 센서로부터 값을 받아들여, LED나 모터와 같은 외부 전자 장치들을 통제함으로써 환경과 상호작용이 가능한 물건을 만들어 낼 수 있다. 임베디드 시스템 중의 하나로 쉽게 개발할 수 있는 환경을 이용하여, 장치를 제어할 수 있다.

**(2) 아두이노 보드의 장점**

아두이노는 마이크로컨트롤러를 쉽게 동작시킬 수 있다.

아두이노 통합 개발 환경(IDE)을 제공하며, 소프트웨어 개발과 실행코드 업로드도 제공한다. 그리고 어도비 플래시, 프로세싱와 같은 소프트웨어와 연동할 수 있다.

아두이노 IDE를 이용해서 컴파일된 펌웨어를 USB를 통해 쉽게 업로드 할 수 있다. 또한, 아두이노는 다른 모듈에 비해 비교적 저렴하고, 윈도를 비롯해 맥 OS X, 리눅스와 같은 여러 OS를 모두 지원한다. 아두이노 보드의 회로도가 CCL에 따라 공개되어 있으므로, 누구나 직접 보드를 만들고 수정할 수 있다.

**(3) 아두이노 보드의 종류**

아두이노 UNO : 아두이노의 표준보드로써 현제 가장 보편적으로 사용되는 보드 5V, 3.3V 모두 지원하며, USB 전원을 사용 가능하며 외부입력전운도 지원한다. 안드로이드 표준보드로, 핀 배열 등이 UNO를 기준으로 맞춰져 있으므로 확장 쉴드 등을 사용하는데 최적이다.

아두이노 Nano : UNO보드의 소형화 버전이다. UNO와 같은 ATmega328 칩을 사용하므로 UNO의 기능을 그대로 제공하고 mini-B 타입 USB도 내장되어있다. UNO 보드와 같은 칩을 사용하므로 개인적으로 UNO 계열의 보드라고 부르며 사용 방법도 동일하고, 라이브러리도 그대로 쓸수 있다.

아두이노 레오나르도 : UNO와 다르게 ATmega32u4 칩을 탑재하고 있다. ATmega32u4 칩은 내부에 USB 통신을 위한 기능을 갖추고 있기 때문에 UNO 보드처럼 별도의 변환기를 필요로 하지 않다. UNO 보드에서는 PC와 serial로 통신하기 위해 D0, D1(디지털 1번, 2번) 핀을 사용하는데 레오나르도 보드에서는 이게 필요없기 때문에 D0, D1 핀을 다른 모듈과의 serial 통신에 사용할 수 있다.

아두이노 Micro 보드 : 레오나르도 보드의 소형화 버전, 레오나르도 보드처럼 ATmega32u4 칩을 탑재하고 있기 때문에 특징도 레오나르도 보드와 거의 동일하다.

**2.2 릴레이 모듈**

**(1) 릴레이 모듈의 정의**

입력이 어떤 값에 도달하였을 때 작동하여 다른 회로를 개폐하는 장치로서 전기, 전자제품의 구동을 스위치만으로 제어하는 장치이다. 릴레이 장치로 다양한 전자기기의 제어를 할 수 있습니다.

**(2) 릴레이 모듈의 기능**

릴레이 모듈은 릴레이(Relay)를 1개 제어할 수 있는 모듈로써 기본적으로 5V에서 동작한다.

릴레이(Relay)는 전자석의 원리로 전류가 흐르면 자기장을 형성해 자기력으로 자석을 끌어당겼다가 전류가 흐르지 않으면 자석을 놓는 원리다.

릴레이에 센서, 모듈 또는 가정에서 사용하는 멀티탭이나 형광등 스위치 등을 연결하여 스위치로써 On/Off 제어를 할 수 있다.

**3. 스마트 멀티탭의 구성**

**3.1 물리적 구성요소**

**(1) 아두이노 보드 (Aduino UNO)**

|  |  |
| --- | --- |
| 전자 부품, 회로 구성요소, 패시브 회로 부품, 전자제품이(가) 표시된 사진  자동 생성된 설명 | 아두이노의 각 핀에 릴레이를 연결하여 릴레이를 제어한다. |

**(2) 릴레이 보드**

|  |  |
| --- | --- |
| 텍스트, 전자제품, 전자 공학, 전자 부품이(가) 표시된 사진  자동 생성된 설명 | 아두이노와 연결하여 멀티탭의 ON/OFF기능을 한다. |

**(3) 어댑터**

|  |  |
| --- | --- |
| 케이블, 정보기기, 전자제품, 헤드폰이(가) 표시된 사진  자동 생성된 설명 | 콘센트 부분을 분해하여 한쪽에는 멀티탭과 연결 다른 한쪽에는 콘센트와 연결한다. |

**(4) 멀티탭**

|  |  |
| --- | --- |
| 전원 플러그 및 소켓, 실내이(가) 표시된 사진  자동 생성된 설명 | 콘센트의 접지선과 멀티탭의 접지 연결, 한쪽에는 어댑터와 콘센트를 연결, 다른 한쪽에는 릴레이보드와 연결하여 릴레이보드로 ON/OFF를 제어한다. |

**(5) 콘센트**

|  |  |
| --- | --- |
| 전자제품, 플러그, 전원 플러그 및 소켓, 케이블이(가) 표시된 사진  자동 생성된 설명 | 멀티탭, 어댑터, 릴레이보드, 아두이노에 전원을 연결해주는 역활  각 멀티탭에 접지 연결하는 역활 |

도표, 라인, 평면도, 텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**(5) 구성도**

**3.2 논리적 구조**

**(1) 내부 수행 순서도**

**(2) 릴레이 설정 소스코드**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

각 릴레이보드를 아두이노 핀 7 ~ 10과 연결한 것을 설정해준다.

state를 통해서 각 릴레이의 상태를 확인한다.

초기설정은 모든 릴레이를 LOW로 설정하여 릴레이를 OFF상태로 한다.

**(3) 릴레이 제어 소스코드**

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

시리얼 모니터에서 값을 받을 수 있게 Serial.available()을 사용했고 입력된 값을 읽기 위해서 Serial.read()를 사용한다.

Serial.read()를 통해서 받은 값을 각각의 입력 값에 맞게 설정을 한다.

입력 값이 1 ~ 4인 경우는 위의 state값에 따라 false 상태일 때는 릴레이를 HIGH로 설정해주고 state를 true로 변경하게 해준다.

입력 값이 0인 경우에는 모든 릴레이의 state를 false로 하고 LOW로 설정하여 멀티탭의 모든 전원을 끊는다. 반대로 입력 값이 9인 경우에는 모든 릴레이의 state를 true로 하고 HIGH로 설정하여 멀티탭의 모든 전원을 연결한다.

**4. 하드웨어 조립 및 제작**

**4.1 하드웨어 구성요소**

실내, 목재이(가) 표시된 사진

자동 생성된 설명

재료 : 멀티탭 4구, 220V 전기선, 콘센트, 아두이노 UNO, 릴레이 모듈 4개, 어댑터, 케이스

**4.2 조립 및 제작 순서**

|  |  |  |
| --- | --- | --- |
| 1 | 전원 플러그 및 소켓, 실내이(가) 표시된 사진  자동 생성된 설명 | 각 멀티탭을 케이스에 부착시킨후 고정시켜준다. |
| 2 | 전기 배선, 케이블, 전기 공급, 전자 공학이(가) 표시된 사진  자동 생성된 설명 | 멀티탭에 전원에  각각 220V 전기선의  파랑색 전원과 갈색 전원을  연결해주고 초록색 선으로 접지를 연결해준다.  콘센트에도 각각의 선들을 위와 같이 연결한다. |
| 3 | 케이블, 전기 배선, 공구, 전자제품이(가) 표시된 사진  자동 생성된 설명 | 어댑터를 분해한 후 기존의 콘센트를 제거한 뒤  전원을 220V 전기선과 연결해준다 |
| 4 | 텍스트, 실내, 바닥이(가) 표시된 사진  자동 생성된 설명 | 멀티탭에 연결된 220V의 파란선과 콘센트의 파란선을 연결해주고 어댑터에 연결해준다.  각각의 접지선을 연결해준다. |
| 5 | 전자제품, 전자 공학, 회로 구성요소, 전자 부품이(가) 표시된 사진  자동 생성된 설명 | 멀티탭의 220V의 갈색선과 릴레이보드의 한 포트에 연결해주다.  다른 한 포트의 220V선과 콘센트의 갈색선을 어댑터에 연결해준다. |
| 6 | 전자 공학, 회로 구성요소, 전자 부품, 회로이(가) 표시된 사진  자동 생성된 설명 | 아두이노와 어댑터를 연결하고  아두이노와 릴레이보드의 전원을 연결하고 7~10 핀에 각각 릴레이보드와 연결한다. |
| 7 | 실내, 플라스틱, 케이블, 전자제품이(가) 표시된 사진  자동 생성된 설명 | 최종적으로 기판 전체의 납땜을 완료하고 케이스안에 정리한다. |

**4.3 하드웨어 제어**

|  |  |  |
| --- | --- | --- |
| 1 | 전자제품, 전자 공학, 전기 배선, 케이블이(가) 표시된 사진  자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진  자동 생성된 설명 | USB 시리얼 케이블을 이용해 아두이노에 소스코드를 업로드 한다. |
| 2 | 스크린샷, 텍스트, 소프트웨어, 디스플레이이(가) 표시된 사진  자동 생성된 설명 | 시리얼 모니터를 이용해서 입력한다.  각 입력값에 대한 상태변화를 확인할 수 있다. |

**5. 소프트웨어 구성**

**5.1 아두이노에 대한 소스코드**

* **별도 첨부**

**6. 멀티탭 성능 테스트**

|  |  |  |
| --- | --- | --- |
| 1 | 텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진  자동 생성된 설명전자제품, 전자 공학, 전기 배선, 회로 구성요소이(가) 표시된 사진  자동 생성된 설명 | 각 입력에 따라서 릴레이보드의 전원이 들어오는 것을  확인 할 수 있다. |
| 2 | 실내, 텍스트, 컴퓨터 모니터, 모니터이(가) 표시된 사진  자동 생성된 설명 | 릴레이보드와 연결된 멀티탭을 시리얼을 통해서 ON/OFF하는 것을  확인한 뒤  전기를 이용해야 하는 장비를 통해서 ON 상태에서 전기가 통하는 것을 확인한다. |
| 3 | 실내, 컴퓨터 모니터, 텍스트, 개인용 컴퓨터이(가) 표시된 사진  자동 생성된 설명 | 시리얼 모니터에서 연결된 릴레이에 OFF값을 입력하면 전원이 나가는 것을 확인할 수 있다. |
| 4 | 실내, 텍스트, 컴퓨터 마우스, 케이블이(가) 표시된 사진  자동 생성된 설명 | 이후에 모든 멀티탭에 연결하고 전체 ON/OFF를 하여 모든 명령 시 실행되는 것을 확인할 수 있다. |

<각 입력에 따른 상태변화 파악>

**7. 결론**

**7.1 문제점과 해결책**

**(1)블루투스 기능**

현재까지의 기능을 통해서 시리얼 입력에 따른 전원기능이 가능하지만 사용 시에는 시리얼모니터로 입력해야한다는 번거로움이 있다. 이러한 번거로움을 해결하기 위해서 블루투스모듈을 추가하여 원격으로 제어할 수 있게 해주면 사용 시에 시리얼 모니터를 이용할 필요가 없게 된다.

**(2)어플 이용**

현재 나온 블루투스 어플을 사용할 수 있지만 각 멀티탭에 어떤 장치가 연결됐는지 확인할 수 있게 이 멀티탭에 맞는 별도의 어플을 제작하려 한다.

**7.2 기대효과**

이 프로젝트를 진행하면서 중요한 점은 입력 값에 따른 릴레이의 상태변화와 전원공급 여부였다. 그리고 그 뒤에 번거로움을 제거하기 위해서 추가적으로 블루투스 모듈 연결이라는 해결책을 찾았다. 블루투스 연결이 가능하고 이를 어플로 제어한다는 가정을 한다면 이 프로젝트는 충분히 가능성이 있다고 판단됐다. 이를 통해서 이 프로젝트를 조금 더 발전시켜서 어플을 이용해 멀티탭을 제어할 수 있다면 이 프로젝트는 스마트 홈 제품으로서 대기업들의 제품보다 저렴한 가격으로 시장에서 경쟁력이 있을 것이라는 예상한다.

**<첨부 – 소스코드>**

int Relaypin1 = 7; // IN1

int Relaypin2 = 8; // IN2

int Relaypin3 = 9; // IN3

int Relaypin4 =10;

bool state1, state2 , state3, state4;

void setup()

{

Serial.begin(9600);

pinMode(Relaypin1,OUTPUT); // 릴레이 제어 1번핀을 출력으로 설정

pinMode(Relaypin2,OUTPUT); // 릴레이 제어 2번핀을 출력으로 설정

pinMode(Relaypin3,OUTPUT); // 릴레이 제어 3번핀을 출력으로 설정

pinMode(Relaypin4,OUTPUT); // 릴레이 제어 4번핀을 출력으로 설정

digitalWrite (Relaypin1, LOW); // 릴레이 OFF

digitalWrite (Relaypin2, LOW); // 릴레이 OFF

digitalWrite (Relaypin3, LOW); // 릴레이 OFF

digitalWrite (Relaypin4, LOW); // 릴레이 OFF

}

void loop()

{

if (Serial.available()) {

switch (Serial.read()) {

case '1': Serial.println("Relay1");

if(state1==false){

state1 = true;

digitalWrite (Relaypin1, HIGH);

}

else if(state1==true){

state1 = false;

digitalWrite (Relaypin1, LOW);

}

break;

case '2': Serial.println("Relay2");

if(state2==false){

state2 = true;

digitalWrite (Relaypin2, HIGH);

}

else if(state2==true){

state2 = false;

digitalWrite (Relaypin2, LOW);

}

break;

case '3': Serial.println("Relay3");

if(state3==false){

state3 = true;

digitalWrite (Relaypin3, HIGH);

}

else if(state3==true){

state3 = false;

digitalWrite (Relaypin3, LOW);

}

break;

case '4': Serial.println("Relay4");

if(state4==false){

state4 = true;

digitalWrite (Relaypin4, HIGH);

}

else if(state4==true){

state4 = false;

digitalWrite (Relaypin4, LOW);

}

break;

case '0': Serial.println("OFF");

state1=false;

state2=false;

state3=false;

state4=false;

digitalWrite (Relaypin1, LOW);

digitalWrite (Relaypin2, LOW);

digitalWrite (Relaypin3, LOW);

digitalWrite (Relaypin4, LOW);

break;

case '9': Serial.println("On");

state1=true;

state2=true;

state3=true;

state4=true;

digitalWrite (Relaypin1, HIGH);

digitalWrite (Relaypin2, HIGH);

digitalWrite (Relaypin3, HIGH);

digitalWrite (Relaypin4, HIGH);

break;

default: Serial.println("Unknown");

break;

}

}

}