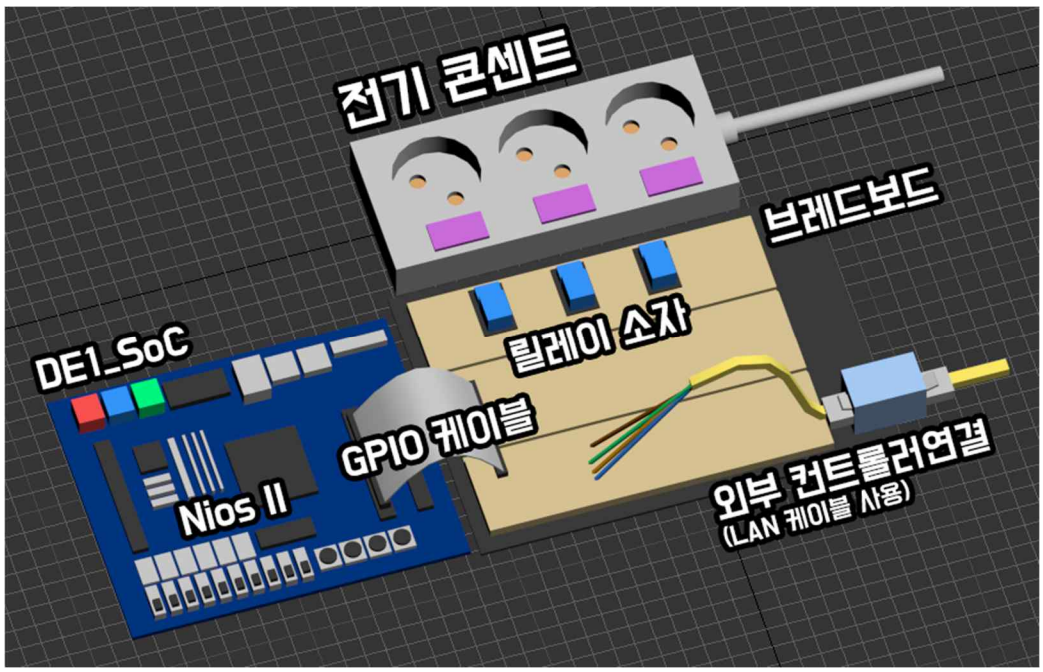
교과목 명		전자 하드웨어 설계						
설계 제목		다기능 전원 제어 장치 (Multi-Function Power Supply Device)						
설계 기간		2018 년도 2 학기						
지도교수		이재환						
팀원	이름	이민우	학번	2014122191	☎	010-6476-5164	E-mail	b727dlalsdn@naver.com
	이름	김영찬	학번	2013122041	☎	010-9357-8504	E-mail	kau_esc@naver.com
	이름		학번		☎		E-mail	

목표설정 — **설계 목표**

1. 초록

C 언어를 기반으로 3 구 멀티탭에 각각 연결된 물체의 전원을 제어하고 기능을 선택적으로 조작하는 controller 를 만든다. 하드웨어인 본체를 직접 제작해, 멀티탭의 각 구의 선을 스위치 역할을 하는 릴레이와 연결하여 사용자가 원할 때 멀티탭의 각 포트별로 전원을 제어할 수 있도록 한다. De1-SoC 보드에 있는 GPIO 포트의 VCC, GND 와 여러 Data Pin 을 외부에 연결되어 있는 controller 와 연결해서 사용자가 입력한 data 를 매 clock 마다 De1-SoC 에 받아온다. De1-SoC 는 Interrupt Handler 를 통하여 GPIO 포트에서 받아온 값을 임의로 만든 프로토콜을 기반으로 해석해 Input Buffer 에 저장하고, 구현한 프로그램은 Input Buffer 에 있는 값을 읽어 들여 수행하고자 하는 동작을 실행한다.



< 그림 1 - 다기능 전원 제어 장치의 설계 기획도 >

2. 기능

1) 본체 외부에 연결된 controller 로 멀티탭 제어

- Bread board 의 IC 소자를 기반으로 구성한 controller 에서 clock 과 data 값을 De1-SoC 에 전달하고, De1-SoC 는 이를 입력 받아 상황에 맞는 값을 출력해 멀티탭에 해당 구에 보내 작동시킴.
- controller 에서 보내는 data 값은 각 포트에 얼마나 작동시킬지 나타내는 타이머 값과 멀티탭의 어느 포트를 작동시킬지에 대한 값임.
- controller 가 8bit 크기의 데이터를 보낼 때, De1-SoC 가 해독함에 있어서 오류를 적게 하기 위해 Preamble 방식을 사용함.

2) 모니터에 현재 상황 출력
   - De1-SoC 의 VGA 포트를 통해 모니터에 해당 포트가 ON 이면 녹색, OFF 이면 적색으로 표시됨.
   - 입력해준 타이머 값을 초 단위로 모니터에 나타냄.

3) 외부 controller 가 없을 시 PUSH BUTTON 으로 제어
   - 위 그림에서 외부 controller 의 입력 단자에서 LAN 케이블을 제거할 시 De1-SoC 의 PUSH BUTTON 을 사용해 제어함.
   - 해당 구를 확인하기 위해 Switch 를 사용하여 01 이면 첫번째 구, 10 이면 두번째 구 그리고 11 이면 세번째 구를 가리키게 함.
   - KEY[0]는 전원을 TOGGLE 해 ON/OFF 시켜주며 KEY[1]는 10 초, KEY[2]는 20 초, KEY[3]은 30 초를 타이머로 설정함.

4) 릴레이 부품 사용한 전원 제어
   - De1-SoC 에 있는 GPIO 를 이용해 멀티탭의 릴레이 소자를 조작하여 220V 를 제어함.

설계 규격

1. GPIO 를 주로 사용하여 입출력을 받으므로, GPIO 의 규격을 설명함.
   1) FPGA 의 Parallel ports 를 사용함.
   2) De1-SoC 의 2 개의 GPIO0 와 GPIO1 은 각각 JP1 과 JP2 expansion parallel port 임.
   3) JP1 과 JP2 각각 32bit 의 양방향 핀으로 구성.
   4) GPIO 의 사용 가능한 각 핀은 input 이 될 수도 있고 output 이 될 수도 있음.
   5) JP1 과 JP2 의 address base 는 0xFF200060 과 0xFF200070 임.
   6) GPIO 의 address map 은 4 개로 구성 되어 있고 각각 32 bit 임..
      - base : pin 으로부터 데이터값을 읽어오거나 pin 에 데이터값을 쓰는 data register 으로 읽고 쓰기가 가능함.
      - base + 4 : pin 을 0 으로 설정하면 input 이고 1 으로 설정하면 output 으로 환경을 설정해 주는 Direction register 으로 읽고 쓰기가 가능함.
      - base + 8 : 각각의 pin 의 interrupt 를 enable 해주는 Interrupt Mask register 이며, 해당 pin 은 "base + 4"에서 0 으로 설정해 input 상태로 만들어 줘야함.
      - base + 12 : interrupt 발생시 어느 bit 가 변화했는지 확인하는 Edge capture register 으로, 해당 bit 를 1 을 써서 clear 해주고 전체 bit 를 clear 해주기 위해서는 0xFFFFFFFF 을 씀.
   7) Direction register 를 0 또는 1 로 설정해 초기치를 설정함.
   8) 어느 interrupt 가 enable 되면 입력 pin 의 값이 1 에서 0 으로 바뀌는 Negative edge 임.
   9) NIOS ||에서 JP1 과 JP2 의 interrupt level 은 11 과 12 임.
   10) Edge capture register 의 bit 가 high 값일 때 인터럽트가 트리거 되기 때문에 인터럽트를 enable 하거나 interrupt handler 를 종료하기 전에 Edge capture register 를 초기화 해 줘야함.
   11) JP1 과 JP2 모두 1 과 3 번 pin 은 clock in 을 하고 19 와 21 번 pin 은 clock out 을 하는 pin 임.
   12) 1 번 pin 은 5V 가, 29 번 pin 은 3.3V 가 인가되는 pin 이며 각각 다음 pin 인 12 와 30 번 pin 은 접지해주는 GND 임.

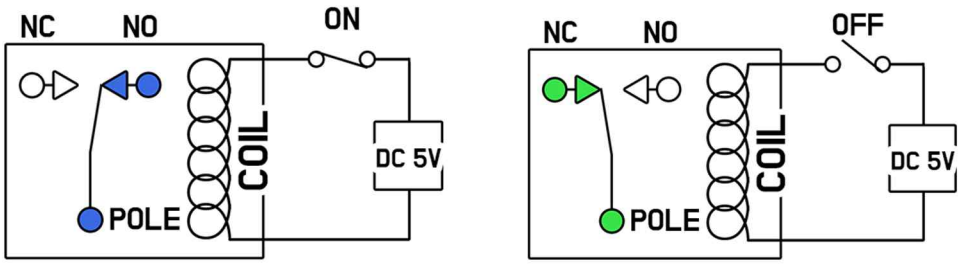
2. De1-SoC 의 VGA 포트를 통해 모니터에 나타내기 때문에 모니터에 나타내는 과정을 설명함.
   1) FPGA 의 Video-out-port 를 사용함.
   2) Pixel Buffer register 의 base 는 Buffer 으로 Buffer 의 시작 주소임.
   3) base + 4 는 Back buffer 주소임.
   4) make frame 과 draw frame 을 동기화 시키고 make frame 이 준비되면 buffer 에서 front buffer 와 back buffer 를 swap 해주어 V-Sync 와 Double buffering 을 해 줌.

합성/분석

관련 기술

1. De1-SoC 보드와 Nios II 프로세서
   1) 해당 프로젝트를 구현하기 위한 하드웨어로는 Terasic 사의 De1-SoC 보드를 사용함.
   2) 프로그램은 De1-SoC 의 FPGA 프로세서인 Nios II CPU 를 사용함.
   3) Nios II 프로세서는 다음과 같은 특징이 있음.
      - Reduced Instruction Set Computer (RISC) Architecture 임.
      - 레지스터와 메모리는 Load/Store Instruction 으로 구성되어 있음.
      - 산술 또는 논리 연산이 General Purpose Register 에서 이루어 짐.
      - 모든 Register 와 Instruction 이 4Byte, 즉 32bit 로 정형화되어 있음.
      - Nios II / fast , Nios II / Standard , Nios II / economy 라는 3 개의 설정이 존재함.
      - Little-Endian 방식으로 데이터를 저장함.

2. 릴레이 (Relay) 소자
   1) 장치의 제어, 회로의 보호 등을 위해 계전기를 사용하는 소자임.
   2) 미리 설정해 둔 전기량에 대응해서, 전기적 입력의 유무, 또는 대소 등의 형태를 식별하여 다른 전기 회로의 개폐를 제어하는 소자를 말함.
   3) 즉, 낮은 전압 (5V)을 이용하여 더 높은 전압(220V)를 제어하는 스위치임.



<그림 1 - 릴레이 소자의 내부 동작 >

   4) 릴레이는 내부에 전자석 코일을 포함하고 있어서 전류가 통하게 되면 자기장을 형성한다. 이 자기장은 릴레이 내부의 POLE 을 움직여 전자 회로의 개폐를 조작할 수 있음.
   5) 이 릴레이 소자를 통해 De1-SoC 의 GPIO 로 멀티탭을 제어함.

3. 프리앰블 비트 (Preamble Bit)
   1) 비트를 신호화 할 때, 비트의 타이밍을 송신 측과 수신 측 양쪽에서 일치시키기 위해 Data 의 선두에 지금부터 데이터 프레임이 시작된다는 의미의 Preamble Bit 를 추가하여 송신함.
   2) 주로 Ethernet 통신에서 Ethernet Frame 동기화를 위해 사용됨.
   3) 이번 프로젝트에서는 De1-SoC 와 사용자 controller 의 Bit 동기를 맞추기 위해 사용됨.

4. 입력 버퍼 (Input Buffer)
   1) 이번 프로젝트에서는 GPIO Interrupt, PushButton Interrupt 등 여러 Interrupt 를 사용하고, 해당 Interrupt 별로 이를 제어하기 위한 Interrupt Handler 가 필요함.
   2) 이 때, Interrupt Handler 에 해당되는 기능을 기술하는 것은 바람직하지 못한 프로그래밍 방식이라고 생각됨.
   3) 예를 들어, 키보드의 경우에도 키보드는 직접 필요한 기능을 수행하는 것이 아니라 단지 Buffer 에 해당하는 값을 넣는 입력 장치임. 키보드의 입력이 필요한 장치는 Buffer 에 저장된 값을 읽어 오는 것으로 동작함.
   4) 마찬가지로, 이번 프로젝트의 PushButton Interrupt Handler 와 GPIO Interrupt Handler 는 특수한 기능을 수행하는 것이 아닌, 사용자가 입력한 값을 Input Buffer 에 넣는 기능만을 수행함.
   5) 구현할 프로그램은, Loop 를 돌며 Input Buffer 가 비어 있는지 확인하고, 아니라면 Buffer 의 값을 읽어와

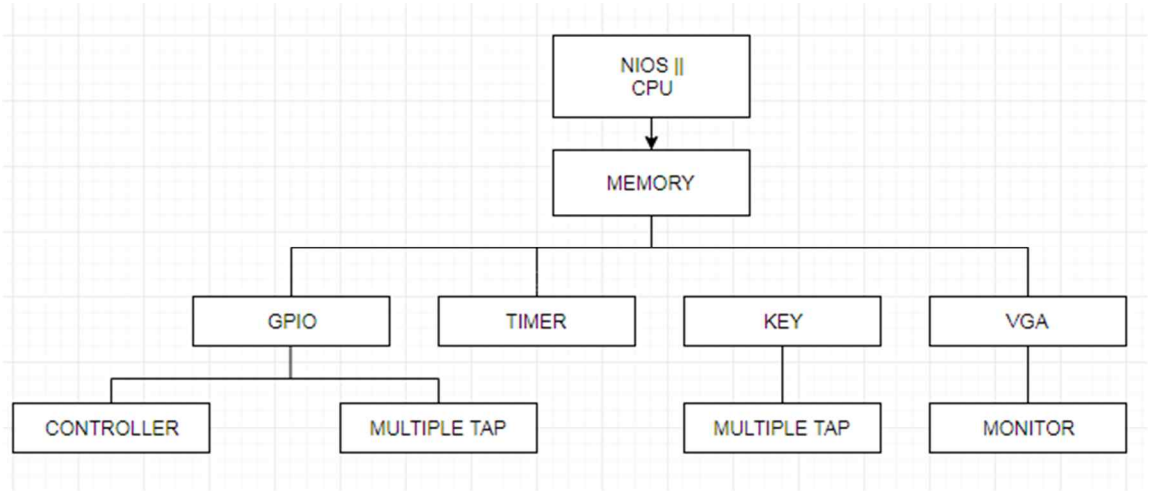
사용자가 입력한 기능을 수행하도록 설계함.

6) C 언어로 Buffer 를 구현하기 위해서는 데이터가 넣어질 위치인 in 이라는 함수와 데이터를 빼내야 할 위치인 out 이라는 변수를 사용해야함.

제작 **설계 계획**

1. 하드웨어 아키텍처 (Hardware Architecture)

1) 이번 프로젝트는 De1-SoC 를 이용한 프로젝트로서, Nios II 프로세서를 사용함.
2) 또한 각종 I/O 의 Interrupt 와 Data 값을 읽거나 쓰기 위해 장치에 접근해야 할 때에는 Memory Mapped I/O 를 사용하는 De1-SoC 의 특성에 맞게 해당 주소 값을 통해 데이터를 읽어 오기로 함.
3) 사용될 I/O 장치는 크게 4 분류로 아래와 같다.
   - 사용자 controller 의 입력을 받고, 릴레이를 제어하기 위한 GPIO
   - 장치의 Timer Toggle 기능을 구현하기 위한 타이머
   - 사용자 controller 없이도 다기능 전원 장치를 조작할 수 있게 하기 위한 Push Button
   - 다기능 전원 장치의 현재 상태를 가시적으로 확인할 수 있게 하는 VGA



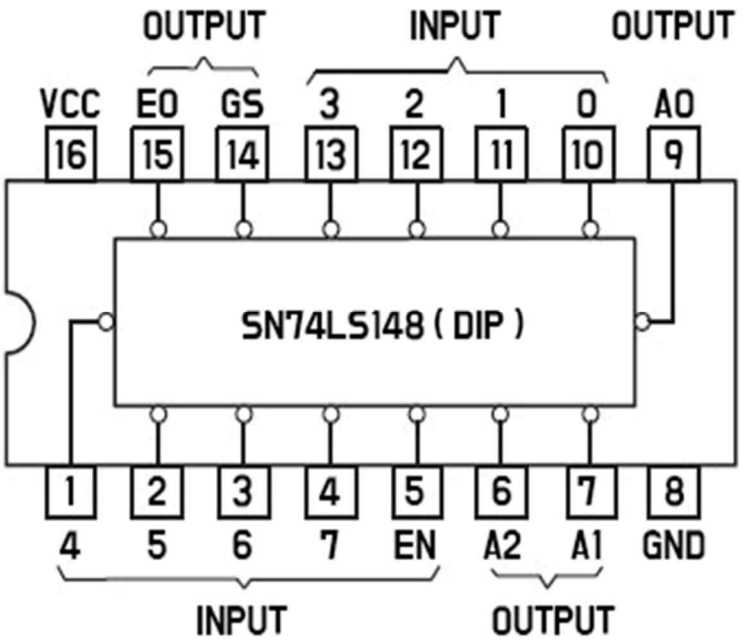
< 그림 2 - Hardware Architecture >

2. 멀티탭 파츠

1) 멀티탭은 3 포트를 각각 제어 할 수 있는 멀티탭을 구매하기로 함.
2) 이번 프로젝트는 멀티탭을 분해하여, 해당 포트의 스위치를 제거하고 그 자리에 relay 모듈을 장착하여 각 포트를 제어하는 방향으로 구현하기로 함.

3. 사용자 Controller

1) 다기능 전원 제어 장치의 여러 기능을 사용하기 위해서는 De1-SoC 의 Key 와 Switch 만으로는 부족하다고 판단됨.
2) 이를 위해 사용자가 좀 더 직관적으로 장치를 사용할 수 있도록 따로 특수한 controller 를 만들기로 함.
3) 해당 controller 는 De1-SoC 와 직접 통신이 가능해야 하는 전자 부품으로써, Bread Board 위에 IC 소자를 사용하여 구현하도록 함.
4) 해당 controller 는 0~9 까지의 숫자 버튼과, 확인/취소 버튼, 그리고 UP / DOWN / LEFT / RIGHT 의 4 방향 버튼으로 구성되어 있으며, 이 16 개의 버튼을 통해 사용자는 원하는 기능을 수행 할 수 있음.
5) 16 가지 버튼을 De1-SoC 에게 전달하기 위하여, 8-to-3 Encoder 인 SN74LS148(DIP) 2 개 사용하여, 16 가지 버튼의 경우의 수를 6Bit 로 전달함.



< 그림 - SN74LS147 의 Pin Arrangement >

6) controller 모듈은 어떤 버튼이 눌렸는지를 나타내는 6Bit Data 외에도 De1-SoC 의 해석을 도와 주기 위해 Preamble Bit 로써 11 의 2 Bit 를 합쳐서 같이 보내기로 함.
7) 따라서 De1-SoC 와 controller 는 VCC, GND, Data, Clock 의 4 Bit 로 통신을 함.

4. MFPSD 프로토콜 (Multi-Function Power Supply Device Protocol)

1) controller 가 입력한 버튼에 따라서 De1-SoC 에 입력되는 비트가 달라짐.
2) controller 는 이를 Clock 신호에 맞춰서 1 Bit 씩 보내는 Serial 통신으로 구현함.
3) 이때 송신되는 Bit 는 아래와 같은 프로토콜을 기반으로 보내 지며, De1-SoC 도 데이터를 수신하면 해당 프로토콜을 참고하여 데이터를 해석함.

번호	비트 (6 Bit)	내용	번호	비트 (6 Bit)	내용
0	000000	숫자 0 키	8	001000	숫자 8 키
1	000001	숫자 1 키	9	010000	숫자 9 키
2	000010	숫자 2 키	10	011000	확인/선택 키
3	000011	숫자 3 키	11	100000	취소/지움 키
4	000100	숫자 4 키	12	101000	LEFT 키
5	000101	숫자 5 키	13	110000	RIGHT 키
6	000110	숫자 6 키	14	111000	사용하지 않음
7	000111	숫자 8 키			

< 표 1 - MFPSD 프로토콜 표 >

4) 사용자 controller 에서 언급했듯이, 버튼을 누르면 controller 는 11 이라는 2 Bit Preamble 뒤에 버튼에 해당하는 6 Bit 코드를 추가하여 8 Bit 신호를 De1-SoC 에게 Clock 신호와 함께 전달함.
5) De1-SoC 는 전달받은 Clock 에 맞춰 8 Bit Data 를 해석하고, MFPSD 프로토콜 표를 참고하여 판단함.

5. 입력 버퍼 (Input Buffer)

1) 이번 프로젝트에서는 1024 Byte 의 링 형태의 Buffer 를 사용함.
2) 버퍼와 관련된 핵심 코드는 아래와 같음.

기본 선언

```
# define BUFFER_SIZE 1024;                    // 버퍼의 사이즈. 여기서는 1024Byte 로 정함.
char buffer inputBuffer;                      // Char 형 버퍼를 만듦
int in = 0; int out = 0;                      // 버퍼의 읽고 쓰는 위치를 기록
```

Buffer 를 채우는 함수

```
void putBuffer ( char n ) {
    if ( (in + 1) $ BUFFER_SIZE == out ) return;    // 버퍼가 꽉참
    inputBuffer [ in ] = n;                         // 버퍼에 값을 넣음
    in = ( in + 1 ) % BUFFER_SIZE;                  // 버퍼를 링 형태로 생각하여 in 값을 증가시킴
}
```

Buffer 에서 꺼내는 함수

```
char getBuffer () {
    if ( in == out ) return -1;                     // 버퍼가 비어있음
    char result = inputBuffer [ out ];              // 버퍼의 값을 가져옴
    out = ( out + 1 ) % BUFFER_SIZE;                // 버퍼를 링 형태로 생각하여 out 값을 증가시킴
    return result;                                  // 꺼내온 값을 반납함
}
```

< 표 2 - 입력 버퍼 구현에 관한 핵심 코드 >

3) in 와 out 변수는 버퍼를 읽거나 쓸 때마다 자신이 사용되면 1 증가함. 이때, BUFFER_SIZE 보다 크다면 0 으로 초기화 되어 링 형태의 버퍼를 구현 할 수 있음.
4) 프로그램은 Loop 를 돌며 Buffer 의 값을 가져온다.

6. 소프트웨어 아키텍처 (Software Architecture)

1) 해당 프로그램의 프로세스는 주로 main 안의 while(1)의 무한 Loop 를 돌며 VGA 를 통해 화면을 출력함.
2) Interrupt Handler 는 Interrupt 를 호출한 원인을 분석하여 TimerISR, PushButtonISR, GPIO ISR 을 호출함.
3) Timer Interrupt 는 멀티탭 포트의 Timer Toggle 기능이 Enable 되어 있다면 해당 포트의 Timer Value 를 1 감소시킴.
4) Push Button ISR 은 입력한 KEY 를 분석하여, 입력 버퍼에 저장함.
5) GPIO ISR 은 입력한 버튼을 분석하여, 입력 버퍼에 저장함.

시험/평가	검증 계획	1. GPIO Interrupt Handler 검증 1) De1-SoC 의 GPIO 와 Bread Board 를 연결함. 2) Bread Board 에 Pull Down 저항을 바탕으로 한 Push Button 스위치 회로를 구성함. 3) Bread Board 에 1K Ohm 저항과 LED 로 이루어진 LED 점멸 회로를 구성함. 4) Push Button 을 통해 LED 을 Toggle 시키는 GPIO ISR 을 구현함. 5) 이를 통해 GPIO 의 ISR 인 GPIO Interrupt Handler 의 기능을 검사함. 2. TImer Interrupt Handler 와 VGA 의 검증 1) VGA 를 사용하기 위해 C 언어 코드로 main.c 를 작성함. 2) VGA 를 이용해 모니터에 지정된 2 자리 숫자를 띄우는 코드를 작성함. 3) 이를 함수로 따로 만들어 지정된 숫자를 개발자가 원하는 위치에 원하는 크기로 출력하도록 구현함. 4) Timer Interrupt 를 사용하여 설정한 시간이 될 때 마다 Timer Interrupt Handler 를 호출시킴. 5) Timer Interrupt Handler 는 임의의 숫자는 1 감소시킴. 6) Timer Handler 에 의해 줄어든 숫자가 출력되는 것으로 Timer Interrupt 와 VGA 의 기능을 검사함. 3. Relay 소자의 검증 1) 검증 (1)을 완료함. 2) 검증 (1)의 회로에서, 1K Ohm 저항과 LED 를 제거하고, Relay 소자를 연결함. 3) Relay 소자에 De1-SoC 에 있는 GPIO 의 VCC 와 GND 를 연결함. 4) Push Button 은 검증 (1)과 마찬가지로, Relay 소자를 Toggle 하는 기능을 구현함. 5) 이를 통해 Relay 소자를 제어할 수 있음을 검사함. 4. Controller 의 검증 1) Bread Board 에 IC 소자를 배치하여 Controller 를 구현함. 2) N555 Timer IC 와 Capacitor, 저항을 이용하여 일정 주기마다 High 신호를 반복해 출력하는 Clock 모듈을 만듦. 3) LAN 선을 통해 De1-SoC 에 있는 GPIO 가 연결된 Bread Board 에 Controller 를 연결함. 4) 해당 Bread Board 에 VCC , GND , Serial Data , Clock 신호등을 전달함. 5) GPIO Interrupt Handler 에서, 데이터가 도착하면 MFPSD 프로토콜을 참조하여 입력한 값을 입력 버퍼에 저장함. 6) Altrera Monitor Program 의 Terminal 을 이용해 무한 Loop 를 돌며 입력 버퍼가 비어 있지 않을 때 마다 버퍼의 값을 printf 함수로 출력하는 코드를 작성함. 7) 버튼을 눌러서 해당 값이 옳게 출력됨을 확인하고 Controller 의 통신 기능을 검사함. 5. Input Buffer 의 검증 1) 검증 (4)를 완료함. 2) 검증 (4)의 코드에서, printf 함수로 버퍼의 내용을 출력하는 기능대신, 무한 LOOP 를 돌며 버퍼에 저장된 값이 있으면 해당 값에 따라 GPIO 를 제어하는 코드를 작성함. 3) 검증 (1)을 완료함. 4) 검증 (1)의 회로에서, 사용자가 입력한 값에 맞게 LED 를 제어하도록 수정함. 5) LED 가 Input Buffer 의 변화에 따라 옳게 변하는지를 확인하고 Input Buffer 의 기능을 검사함. 6. 최종 검증 1) 모든 검증을 완료함. 2) 검증 (3)의 회로에서, Push Button ISR 에서 GPIO 의 Relay 소자의 상태를 직접 조작하는 것이 아니라, 프로그램 상에서 선언된 status 변수에 의해 제어되도록 수정 3) VGA 에서 릴레이 상태에 해당하는 변수와 Timer Toggle 기능을 위한 Timer 출력이 정상적으로

작동되도록 수정함.

4) Timer ISR 이 각 릴레이별 Timer Toggle 기능이 Enable 일 경우, Timer Value 를 1 감소시키도록 수정함.
5) Main 함수에서, 각 릴레이별 Timer Value 가 0 일 경우, status 변수를 수정함. (ON 이면 OFF, OFF 면 ON)
6) Controller 와 KEY 는 PushButton ISR, GPIO ISR 을 이용하여 Main 함수에 선언된 입력 버퍼에 값을 추가함.
7) Main 함수에서, Loop 를 돌 때 마다, 입력 버퍼가 비어 있는지 확인함.
8) 만약 비어 있지 않으면, 입력 버퍼에서 값을 가져와 해당하는 동작을 수행함.
9) 이를 통해 다기능 전원 제어 장치의 모든 기능을 검사함.

일정

주차	진행 계획	예상 산출물 및 Demo 내용
1	- C 언어를 기반으로 KEY 를 눌렀을 때 GPIO 에 연결된 LED 를 ON/OFF 해서 GPIO 를 통해 전원 제어가 되게 한다. - GPIO 에 연결된 PUSH BUTTON 을 통해서 De1-SoC 의 HEX 에 불이 켜지는 것을 확인해 외부의 입력과 De1 -SoC 가 통신 되는 것을 확인한다.	- GPIO 를 통해 외부 controller 와 De1 - SoC 통신하는지 확인할 수 있음. - 실제로 값이 제대로 입력 또는 출력이 되는지 LED 나 HEX 를 통해 확인할 수 있음.
2	- Controller 자체 하드웨어를 설계 및 제작 해서 동작이 잘 되는지 확인한다. - 사용자가 주는 입력에 따라 3 구 멀티탭이 ON/OFF 되는 것을 De1-SoC 의 VGA 포트를 통해 모니터에 나타내는 것을 확인하고, 1 주차에 진행한 GPIO 에 연결된 LED 작동도 확인해 값이 동시에 잘 들어가도록 한다. - C 언어의 코드 상의 타이머를 모니터에 나타내 초 단위로 타이머가 움직이는 것을 확인한다.	- Controller 입력에 따라 LED 가 켜지는지 확인할 수 있음. - 입력해주는 값에 따라 모니터에 ON/OFF 값이 제대로 동기화 되고, 타이머가 초 단위로 작동하는지 확인할 수 있음.
3	- 멀티탭과 De1-SoC 보드와 이를 bridge 해주는 bread board 가 들어 있는 본체 하드웨어를 설계 및 제작한다. - 본체 하드웨어와 외부의 하드웨어인 Controller 를 LAN 케이블로 연결해 통신하도록 한다. - Controller 가 있을 때 Controller 에서 인가 해주는 입력에 따라 원하는 멀티탭의 해당 포트가 ON/OFF 또는 타이머가 잘 맞는지 모니터에 나타낸다. - Controller 가 없을 때 본체 하드웨어의 De1-SoC 의 KEY interrupt 를 사용해 발생 시 Switch 의 값을 읽어 멀티탭의 해당 포트를 ON/OFF 또는 타이머를 작동시켜 이를 모니터에 나타낸다.	- 본체 하드웨어 설계 및 제작해서 Controller 와 연결해 Controller 에서 입력해주는 data 가 clock 에 따라 잘 들어오는지 확인하고, De1-SoC 와 통신해 나오는 출력 값을 멀티탭에 입력하고 멀티탭의 해당하는 구에 작동시키고, 멀티탭의 현재 상황을 확인해 모니터에 나타나는 것을 확인한다.

역할 분담

▶김영찬 (2013122041)
- 팀장으로써 팀 전체를 조율하고 프로젝트의 방향을 조정한다.
- 프로젝트를 진행하면서 회의록 1,2,3 중에 회의록 2 를 작성한다.
- Controller 와 본체 하드웨어에 사용될 재료를 구매한다.
- De1-SoC 에 data 값과 clock 을 보내줌으로써 전원을 제어하는 Controller 을 설계 및 제작한다.
- 최종 발표를 진행할 때, 즉석으로 코딩을 수정하는 역할을 담당한다.
- Controller 와 본체 하드웨어 사이의 serial 통신이 잘 되는지 확인한다.
- Input buffer 를 사용해 Controller 에서 오는 값을 저장한다.
- Controller 의 하드웨어를 만들기 위해 Bread Board 위에 IC 소자를 배치한다.
- GPIO 와 De1-SoC 또는 멀티탭과 GPIO 의 연결을 확인한다.

	▶이민우 (2014122191) - 전체적인 프로젝트에 사용될 플로우 차트를 구상 후 설계한다. - 프로젝트를 진행하면서 회의록 1,2,3 중에 회의록 1 과 3 을 작성한다. - I/O Peripheral 에서 Interval timer, Pushbutton switch parallel port, JP1 Expansion parallel port 그리고 JP2 Expansion parallel port 에 대한 interrupt handler 코드를 구상하고 작성한다. - De1-SoC 와 멀티탭과 bridge 역할을 하는 bread board 가 들어있는 본체 하드웨어를 설계 및 제작한다. - 타이머를 모니터에 초 단위로 나타내고 멀티탭 각 구의 ON/OFF 상태를 나타낸다. - Controller 에서 오는 data 값 또는 KEY interrupt 를 발생시킬 때 이에 해당하는 작동을 하게 해준다. - 최종 발표에 사용할 PPT 를 제작한다. - 최종 발표를 진행할 때, 질의 응답을 담당한다.