|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **교과목 명** | | 전자하드웨어설계 | | | | | | | | |
| **설계 제목** | | 다기능전원제어장치( Multi-Function Power Supply Device ) | | | | | | | | |
| **설계 기간** | | 2018년도 2학기 | | | | | | | | |
| **지도교수** | | 이재환 | | | | | | | | |
| **팀원** | | **이름** | 김영찬 | | **학번** | 2013122041 | **☎** | 010-9357-8504 | **E-mail** | kau\_esc@naver.com |
| **이름** | 이민우 | | **학번** | 2014122191 | **☎** | 010-6476-5164 | **E-mail** | b727dlalsdn@naver.com |
| **이름** |  | | **학번** |  | **☎** |  | **E-mail** |  |
| **목표설정** | **설계** **배경** | 전자 회로 실험과 기초 회로 및 디지털 실험에서 사용했던 장비들에게 전원을 주기 위해서는 전기 콘센트에 Plug를 꽂고 전기 콘센트를 건물 어딘가에 있는 전원을 공급하는 콘센트에 꽂아야 한다. 이 때 전기 콘센트에서 각각의 PORT마다 스위치가 있는 경우 개별적으로 PORT를 제어할 수 있지만, 스위치가 없는 경우는 직접 Plug를 뽑아야 하는 상황이다. 직접 Plug를 뽑거나 스위치가 있는 전기 콘센트에서 해당 PORT를 끄는 경우 실험실 또는 직장과 같이 한 개의 장비만 있는 것이 아니라 여러 장비가 있는 경우 해당 Plug가 꽂힌 곳을 찾기란 쉽지 않을 것이다. 이런 경우를 대비해 DE1-SoC보드와 릴레이 전기 소자와 외부 컨트롤러를 연결해 직접적으로 전기 콘센트에 손을 대지 않아도 사용하고 있는 장비들의 전원을 제어함으로써 장비를 사용하는 사용자에게 편의성을 제공했다.  전기 콘센트는 전원을 ON/OFF하는 역할만 하므로, 타이머를 두어 사용자가 원하는 시간 동안만 전원을 ON하고 타이머가 끝나면 전원을 OFF해서 전원 낭비를 줄이고자 했다. 그리고 예약기능을 넣어 사용자가 자리를 비우고 다른 작업을 하는 경우 미리 컨트롤러나 DE1-SoC를 사용해 미리 설정을 해서 원하는 PORT를 몇 초 뒤에 킬 수 있게 하여 편의성을 제공하며, 컨트롤러 없이 직접적으로 제어를 하고 있은 경우 DE1-SoC 보드만으로도 제어를 할 수 있게 하여 원하는 스타일대로 작동을 할 수 있게 했다.  다기능전원제어장치를 통해 DE1-SoC 보드에 내재되어 있는 기능을 파악하고 이를 제어하기 위한 코딩을 통해 C언어를 사용하기 위해 알고리즘을 설계를 하고 이를 바탕으로 코드를 작성해 C언어와 DE1-SoC 보드를 이해 할 수 있다. 또한 외부 컨트롤러를 제작해야 하기 때문에 전자공학도로써 전자회로와 회로이론과 같은 회로에 대한 이론과 Bread board에 설계한 회로도 대로 옮기면서 전자 회로 실험과 기초 회로 및 디지털 실험과 같은 실험과목의 경험을 직접적으로 활용할 수 있고 점프선과 AND GATE등 전자 부품들을 사용하여 후에도 사회에서 이를 사용할 수 있다는 공학적 의의가 있다. | | | | | | | | |
| **목표** | 1. 목표    **<그림 1 - 다기능전원제어장치의 설계 계획도>**  DE1-SoC 보드에 내재 된 GPIO 포트에 VCC, GND가 할당된 pin과 값을 받고 넣어줄 Data Pin을 LAN 케이블을 사용해 C언어를 기반으로 3개의 PORT를 가진 멀티탭에 물체의 전원을 제어하고 원하는 기능을 선택해 조작하는 외부 controller를 만든다. 하드웨어인 본체를 직접 제작하고, 멀티탭의 각 PORT의 선을 스위치와 같은 역할을 하는 릴레이 소자와 연결해 사용자가 원할 때 멀티탭의 각 PORT별로 전원을 제어 할 수 있도록 한다. 외부 controller는 LAN 케이블을 통해 본체와 연결하고 controller 자체의 매 clock 마다 data를 DE1-SoC 보드에 전송하며 DE1-SoC는 C언어를 기반으로 작성한 interrupt Handler를 통해 GPIO PORT에서 받은 값을 임의로 만든 프로토콜을 기반으로 해석해서 어떤 동작을 수행할 지 파악 후 Input Buffer에 저장하고, 구현한 프로그램이 Input Buffer에 저장되어 있는 값을 읽어서 수행하고자 하는 동작을 실행한다.  2. 기능   1. 외부 controller로 멀티탭 제어  * IC 소자로 구성한 controller에서 clock와 data 값을 DE1-SoC에 전달하며, DE1-SoC는 이를 입력으로 받아 상황에 맞는 값을 출력해 멀티탭의 해당 PORT를 작동시킴. * 외부 Controller에서 보내는 data값은 멀티탭의 PORT에 어느 시간동안 작동 시킬 지 나타내는 타이머값과 어느 PORT를 가리키는지에 대한 값임. * Preamble방식을 사용해 8bit 크기의 data를 DE1-SoC가 data를 해석하는데 오류를 줄임.  1. 현재 상황을 모니터에 출력  * DE1-SoC의 VGA 포트를 통해 모니터에 멀티탭의 해당 PORT의 전원 상태를 ON이면 녹색, OFF이면 적색으로 표시하고, 입력해준 타이머 값을 초단위로 나타냄.  1. PUSH BUTTON으로도 전원 제어  * <그림 1>에서 controller가 연결되어 있지 않는 경우 DE1-SoC의 Switch와 PUSH BUTTON을 사용해서 제어함. * Switch 는 각 PORT를 가리키며 타이머를 증가시키고 초기화 및 OFF를 가리킴. * KEY[0]는 PORT3의 전원을 TOGGLE, KEY[1]은 PORT2의 전원을 TOGGLE, KEY[2]는 PORT1의 전원을 TOGGLE하며 KEY[3]은 Switch에서 지칭하는 동작을 수행하는 특수 KEY로 설정함. | | | | | | | | |
|  | **관련** **기술** **조사, 분석** | 1. DE1-SoC 보드의 GPIO   1. FPGA의 GPIO0와 GPIO1은 각각 JP1과 JP2 Expansion parallel port임. 2. JP1과 JP2는 32bit의 bidirection Pin으로 구성 됨. 3. GPIO0와 GPIO1은 각각 0xFF200060과 0xFF200070의 address를 가짐. 4. GPIO0의 Interrupt level은 11이고, GPIO1은 12임. 5. GPIO는 4개의 address map를 가지며 각각 32bit 임.  * 기존 base : Pin에서 data value를 읽거나 Pin에 data value를 write하는 data register이고 R/W 가능함. * base + 4 : 특정 Pin을 Input으로 사용하고 싶으면 0으로 두고 Output으로 사용하고 싶으면 1로 설정하는 Direction register이고 R/W 가능함. * base + 8 : 원하는 Pin을 interrupt enable로 설정해 해당 Pin이 변화할 때마다 Interrupt를 발생시켜주는 Interrupt Mask register이고, 해당 Pin은 Direction bit를 0으로 설정해 Input으로 두어야함. * base + 12 : Interrupt enable 해준 어느 Pin이 변했는지 확인하는 Edge capture register이며, clear를 해주기 위해서는 해당 Pin을 1을 넣어주고 전체를 clear해주기 위해서는 0xFFFFFFFF을 넣음.  1. 프로젝트를 진행하면서 사용할 VCC5는 11번 pin이고 그 옆의 12번 pin이 ground 임.   2. Relay 소자   1. 연결된 장치의 제어와 회로 보호를 위해 계전기를 사용함. 2. 5V의 낮은 전압을 통해 220V의 상대적으로 높은 전압을 제어하는 Switch임.     **<그림 2 - 릴레이 소자의 내부 동작>**   1. Relay는 전자석 코일이 내부에 있어 전류가 흐르면 자기장을 형성하며 이는 Relay 내부의 POLE을 움직이게 해서 회로의 ON/OFF를 조작함.   3. Input Buffer   1. Pushbutton, Timer, GPIO의 Interrupt를 사용하는데 이 Interrupt를 개별적으로 제어 할 Interrupt Handler를 작성해야함. 2. Interrupt Handler가 해당 기능을 제어하는 것이 아닌 Buffer를 사용해 Pushbutton와 GPIO Interrupt Handler는 사용자가 Input한 값을 Input Buffer에 넣는 것을 수행하게 함. 3. main.c에서 무한 Loop를 돌며 사용자가 입력하는 Input Buffer를 확인하고 비어있으면 넘기고 그렇지 않으면 Buffer에 있는 값을 읽어 해석해 해당 기능을 수행하도록 함. 4. 해당 프로젝트에서는 C언어를 기반으로 코드를 작성하므로 in과 out이라는 함수를 사용해 데이터가 넣어지는 위치와 빼는 위치를 지정해 Buffer를 구현함.   4. Preamble Bit   1. Input Buffer에 전송하는 bit를 보낼 때, 송신단과 수신단의 양쪽의 타이밍을 일치시키기 위해 Data bit의 선두에 값이 시작하는 의미의 Preamble Bit를 추가함. 2. 사용자의 외부 Controller와 DE1-SoC의 Bit 동기화 함. | | | | | | | | |
| **제작** | **설계** **내용** | 1. Hardware Architecture   1. 하드웨어는 DE1\_SoC , DE1\_SoC의 GPIO 브레드 보드, 사용자 컨트롤러로 구성되어 있음. 2. DE1\_SoC의 GPIO 브레드 보드에는 각 포트를 제어 할 수 있는 3개의 릴레이 소자와, 사용자 컨트롤러로부터 Clock 신호와 Data 신호를 받아오는 2개의 디지털 핀으로 구성되어 있음. 3. 컨트롤러와 DE1\_SoC GPIO 브레드 보드는 8 Pair UTP Cable로 연결하기로 함. 4. 사용자 컨트롤러의 버튼은 풀업 저항을 사용하여 누르면 GND신호를 전달함 5. 사용자 컨트롤어의 Clock은 N555 Timer 소자를 사용하였음. 해당 Timer 소자는 구형파를 출력할 때 저항과 캐패시터의 조합으로 주기를 조정 할 수 있음. 6. 사용자가 가시적으로 이용하기 쉽도록 Pixel Buffer와 Character Buffer를 사용하여 모니터 출력을 구현함. 7. 사용자 컨트롤러가 연결되어 있지 않아도 De1\_SoC의 Push Button으로도 조작 할 수 있도록 인터럽트 핸들러를 설계함. 8. 컨트롤러나 Push Button에 기능을 직접 추가하지 않고, 단순히 버퍼에 값을 넣는 역할만 수행함. 9. 자세한 동작은 Main Program에서 동작함. 10. Main Program은 Loop를 돌며 해당하는 Page를 출력함.     **< 그림3 – 다기능 전원 장치의 하드웨어 아키텍쳐 >**  2. Main Program   1. 무한 반복을 돌며 page변수에 저장된 값에 따라 해당되는 페이지를 출력함. 2. 버퍼에서 값을 읽어 옴. 3. Page변수에 저장된 값과 버퍼에서 읽어온 값에 따라 해당하는 동작을 진행함. 4. 동작을 모두 수행한 후, Power 전달에 수정사항이 있을 수 있으므로 확인함. 5. 사용자가 볼 수 있도록 모니터에 출력함.   3. Push Button   1. 사용자 컨트롤러 이외에도 기기를 조작 할 수 있는 수단임. 2. KEY0 : 확인 / KEY1 : 취소 / KEY2 : 왼쪽 이동 / KEY3 : 오른쪽 이동 3. Timer Set Page에서만 특수하게 KEY2 : 10초 추가 / KEY 3 : 1초 추가 로 바뀜.     **< 그림4 – 사용자 입력에 따른 Main Program Page Flowchart >**  4. 사용자 컨트롤러   1. 다음과 같은 IC 소자들이 사용되었다.  * 74HC165 <1EA> : 8bit SIPO Shift Register * 74HC04 <1EA~2EA> Hex Inverter * 74HC148 <2EA> : 8 TO 3 Encoder * 74HC30 <2EA> : Octal NAND * 74HC02 <1EA> : Quad Nor * 74HC08 <1EA~2EA> : Quad And   **https://user-images.githubusercontent.com/40852277/48850088-1fb73d00-edeb-11e8-8f98-04765ed83b25.png**  **< 그림 5 – 사용자 컨트롤러 설계도 >**   * 74HC273 <2EA> : 8 bit Register * 100KOhm Register <16EA> : 스위치가 16개 이므로 * Switch <16EA>: 숫자 10 / 상하좌우 4 / 확인취소 2 * N555 <1EA> : 타이머용 소자       **< 그림 6 - 사용자 컨트롤러의 구상도 >**  5. 사용자 입력 버퍼   1. 키보드는 컴퓨터에게 어떤 키가 눌렸는지를 알려주는 장치이지, 각 버튼마다 특수한 기능이 있는 건 아님. 2. 마찬가지로, KEY\_ISER이나 GPIO\_ISR은 특수한 기능을 가지는게 아니라, 어떤 버튼이 눌렀는지 Main Program에게 전달해주는 방식으로 설계 되어야 코드도 직관적이고 깔끔해 짐. 3. 따라서 모두 다같이 쓰는 버퍼를 하나 만들어서 ISR에서는 버퍼에 넣는 작업을, Main Program에서는 버퍼를 빼는 작업을 진행하도록 함. 4. 버퍼는 buffer\_in과 buffer\_out에 의해 관리되며, 크기가 BUFFER\_SIZE인 링 버퍼임.  |  | | --- | | Input Buffer \_ Write Function | | //==========================================================  // 버퍼에 넣는 함수  //==========================================================  void inputBuffer\_write(char item) {  if ( ( buffer\_in + 1 ) % BUFFER\_SIZE == buffer\_out )  return;    inputBuffer [ buffer\_in ] = item;  buffer\_in = ( buffer\_in + 1 ) % BUFFER\_SIZE ;  return;  } |  |  | | --- | | Input Buffer \_ Read Function | | //==========================================================  // 사용자 버퍼를 읽어오는 함수  //==========================================================  char inputBuffer\_read() {  if ( buffer\_in == buffer\_out )  return 0;    int result = inputBuffer[buffer\_out];  buffer\_out = (buffer\_out + 1) % BUFFER\_SIZE;  return result;  } |   6. 프리엠블   1. 사용자 컨트롤러로 데이터를 보낼 경우, 데이터의 시작을 알리는 프리엠블이 헤더처럼 추가됨. 2. 프리엠블은 0101로 구성되어 있고, N555 Timer 소자가 만드는 Clock 신호에 따라 전달됨 3. De1\_SoC 보드에서도 이 Clock에 맞춰서 데이터를 해석해야 함. 4. 추가적으로 사용자 컨트롤러 이외의 KEY 입력도 받을 수 있도록 상수를 추가하였음. 5. Main Program이나 ISR들이 버퍼에 저장하거나 불러올 때는 아래 표와 같은 값을 사용함.  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0번 키 | **0101** **0000** | 4번 키 | **0101 0100** | 8번 키 | **0101 1000** | LEFT 키 | **0101 1100** | | 1번 키 | **0101 0001** | 5번 키 | **0101 0101** | 9번 키 | **0101 1001** | RIGHT키 | **0101 1101** | | 2번 키 | **0101 0010** | 6번 키 | **0101 0110** | UP 키 | **0101 1010** | OK 키 | **0101 1110** | | 3번 키 | **0101 0011** | 7번 키 | **0101 0111** | DOWN키 | **0101 1011** | BACK 키 | **0101 1111** | | KEY\_0 | **1 000** | KEY\_1 | **1 0001** | KEY\_2 | **1 0010** | KEY\_3 | **1 0011** | | | | | | | | | |
| **설계** **과정** | 초반에 전원을 제어하기 위해 보조배터리 또는 건전지를 통해 전원을 주고 Input해주는 외부 Controller는 간단한 아날로그 회로로 2개의 Input을 가지는 AND 게이트를 사용해서 한 곳에는 계속 1을 주고 나머지 한 곳을 사용자가 원하는 입력을 주는 간단한 회로를 구상했다.  Controller를 구상하면서 5V의 신호로 220V를 제어할 수 있는 방법을 조사하다가 Relay소자를 사용해서 5V를 주면 220V의 신호를 내놓는 스위치 IC 소자를 찾았다.  DE1-SoC의 GPIO는 배우지 않은 부분이라 Altera의 manual을 통해 우리가 필요한 VCC와 Ground이므로 해당 Pin이 몇 번째 위치하는 지와 사용하지 않는 Pin은 어디에 위치하는지 파악해서 회로를 연결할 때 필요한 부분만 사용했다.  이를 통해 첫 번째 주의 DEMO에서 GPIO의 활용을 보이기 위해 Pull down 저항으로 Pushbutton 스위칭회로를 구성해 Polled I/O로 외부의 Bread board와 DE1-SoC의 특정 Pin을 연결해서 Bread board의 Pushbutton을 누르면 DE1-SoC의 LEDR0가 켜지는 것을 확인했다.  다음에는 GPIO를 Interrupt으로 받아 Pushbutton이 눌릴 때 마다 Interrupt가 발생해서 DE1-SoC의 LEDR0가 켜지는 것을 확인했는데, 이 때 Switch역할을 하는 Pushbutton 소자가 아날로그 소자이다 보니 누르는 힘과 방향에 따라서 ON/OFF가 반복적으로 발생해 GPIO ISR에서 printf를 사용해 ON이면 1, OFF이면 0을 나타내게 했더니 한 번 눌렀을 뿐인데 1,0,1,0이 연속적으로 발생해 한번의 입력에 대해 Interrupt가 연속적으로 나타나게 되었다. 이러한 이유를 통해 간단한 회로를 통해 외부 Controller를 제작하는 경우에는 원하는 입력을 주는 것이 제한되어 있고 많은 정보를 주기 위해서는 회로가 복잡해지기 때문에 규칙적인 입력을 bit단위로 주기로 했다. 이를 통해 외부 입력에 대해 DE1-SoC가 이 입력을 받아 작동하는 것을 확인했다.  외부 입력으로 DE1-SoC를 제어하는 것을 확인 했으므로 이번에는 DE1-SoC의 제어에 따라 외부의 Bread board에 연결된 LED이 ON/OFF 되는 것을 구현하기로 했다. DE1-SoC의 KEY0를 누르면 1의 값이 가서 LED가 ON이 되고, KEY1을 누르면 0의 값이 가서 OFF가 되도록 했다. 이번에는 흔들리는 값이 아닌 DE1-SoC에서 주는 정확한 1 또는 0 값을 주는 것이므로 LED가 사용자가 제어 하는 대로 정상 작동했다.  GPIO에 대해 이해를 하기위해 여러 경우를 따져 알아보고 이를 시각적으로 나타낼 방법을 생각하다가 VGA를 통해 해당 전자 제품의 ON/OFF 상태를 나타내고 이 뿐만 아니라 타이머 기능을 추가해서 주어진 시간이 끝나면 OFF가 되도록 기능을 추가해 Bitmap을 통해 0~9까지의 숫자를 굵기를 주어 VGA에 나타냈다. Timer는 1초마다 시간이 가기 위해 Timer address에 100MHz의 값을 넣어주었고 Time Out event가 발생 할 때마다 Interval Timer Interrupt가 발생해 해당 PORT의 TIMER Value들을 1씩 감소시켜 줬다.  또한 Controller가 연결되지 않으면 DE1-SoC의 Pushbutto과 Switch를 통해 멀티탭을 제어해야 하기 때문에 입력에 대해 값이 바뀌는 것을 구현하기 위해 Pushbutton의 KEY0~KEY2은 각각 PORT3~PORT1의 ON/OFF를 해주는 기능을 할당하고 KEY3은 특수기능으로 Switch의 값을 받아와서 sw0~sw2가 각각 high이면 PORT3~PORT1의 Timer Value를 10초씩 증가시키도록 했다. 또한 sw3과 함께 sw0~sw2가 high가 되어 있으면 해당 PORT의 Timer를 초기화하고 전원을 OFF하는 것을 우선적으로 VGA에 나타냈다.  외부 Controller를 완성하고 멀티탭을 분해해 -극은 그대로 두고 +극의 금속 띠를 개조해 각 PORT마다 wire를 납땜해서 원하는 입력을 줄 수 있도록 제작한 후 Relay 소자와 외부 Controller를 Bread board 상에서 연결 후 보조 배터리를 통해 직접 전원을 연결해 줬을 때 멀티탭에 헤어드라이기와 스탠드가 켜지는 것을 확인해 회로상에 문제가 없음을 확인했다.  회로가 정상 작동한 다는 것을 확인했으므로 보조 배터리로 전원을 주는 것이 아니라 DE1-SoC GPIO의 VCC 5V를 통해 Relay소자에 입력을 주어 220V를 출력하게 해서 멀티탭에 연결된 헤어드라이기와 스탠드가 작동하는 것을 확인했다.  외부 Controller는 Bread board에 많은 wire들과 IC 소자가 있어 외부의 충격에 약하기 때문에 전면부에 가시적으로 0~9의 숫자와 확인, 취소, 왼쪽, 오른쪽, 위쪽, 아래쪽을 나타내는 Pushbutton을 만들면서 하드보드지로 보호막을 만들어 외부 충격에 어느정도 내구성을 가지도록 했다.  기본 작동들을 모두 확인 했으므로 Input Buffer를 통해 외부 Controller의 값을 받는데, main의 무한 LOOP에서 Input Buffer를 계속 확인해서 비어있으면 현재 값을 계속 유지하고, 입력을 줘서 Input Buffer가 값이 있으면 이를 해석해서 외부 Controller에서 화살표를 통해 특정 PORT를 가리키고 30을 눌러 30초의 Timer Value를 주고 확인을 누르면 특정 PORT가 30초 동안만 작동되게 했다. 또한 외부 Controller가 없으면 DE1-SoC에서 원하는 PORT의 Switch를 올리고 KEY3을 눌러 Timer Value를 10초씩 증가시키고 해당 PORT가 1이면 KEY2를 눌러서 ON이 되게 해서 정상 작동을 하게 했다. | | | | | | | | |
| **시험 /평가** | **설계** **결과** **검증**, **분석**, **평가** | 1. GPIO interrupt Handler   1. DE1-SoC의 GPIO PORT와 Bread board를 연결함. 2. Bread board에 Pull Down 저항을 바탕으로 Push Button 스위치회로와 1K Ohm 저항과 LED로 LED 점멸회로를 구성함.  * 저항을 큰 것을 사용해 ON 상태에서 LED쪽으로 연결이 되게 해야함. * LED의 양/음극 방향을 정확한 위치에 꽂아야 LED에 불이 들어옴. * 아날로그 신호의 값을 받아오므로 소자가 고장이 나면 정상작동이 되지 않음.  1. PUSH BUTTON을 통해 LED를 Toggle 시키는 GPIO ISR을 구현함.  * GPIO\_ISR을 작성할 때 각 Pin의 input과 output 구분을 잘 해야함.  1. 위 순서를 통해 GPIO Interrupt Handler의 기능을 검사함.  * GPIO의 Edgecapture bit를 clear 하기 위해서는 모든 Edgecapture bit를 1로 설정해야지 다음 GPIO Interrupt를 받아 올 수 있음.   2. Timer Interrupt Handler   1. Timer I/O를 100MHz로 설정해 1초로 두고 시간이 다 될 때 마다 Time out이 발생해 Timer Interrupt Handler를 호출함. 2. Timer Interrupt Handler에서 각 PORT들의 Timer값을 1 감소시킴.  * 1초 단위로 설정을 해서 구현가능한 범위를 초단위로 설정함. * 초단위이므로 * 1초 단위가 아닌 분 또는 시 단위로 설정을 하고 싶으면 Timer I/O에 설정한 값을 100MHz가 아닌 각 단위에 필요한 값으로 설정함.   3. VGA   1. C 언어를 기반으로 코드를 작성해 main.c 만듦. 2. VGA를 통해 모니터에 초 단위에 2자리 숫자를 띄우는 코드를 작성함.  * 2자리 숫자를 띄우는 것은 0~9의 숫자를 나타내는 bitmap은 header 파일로 만들어서 main.c에서 참조하도록 함. * bitmap을 header 파일로 사용하지 않는 경우는 직접 main.c에 bitmap을 나타내는 함수를 만들어서 int main에서 참조하도록 함.  1. VGA는 2.의 Timer Interrupt Handler에서 Interrupt가 발생시 Timer값이 1 감소시키는 것을 계속 반영하기 위해 무한 루프를 돌고 있음.  * int main에서 타이머가 Tearing effect가 나타나지 않도록 V-Sync를 맞춰 주어 VGA에 매끄럽게 Timer가 작동 되게 하고 ON/OFF 상태를 잘 나타내줌.   4. Relay 소자   1. GPIO Interrupt Handler의 회로에서 GPIO가 제대로 들어가는 것을 확인 했으므로 1K Ohm 저항과 LED 제거 후 Relay 소자를 연결함. 2. Relay 소자에 wire를 사용해 GPIO의 VCC와 GND를 연결하여 회로에 추가함.  * Relay 소자는 5V의 값을 주어야 입력이 1이 왔을 때 Switch On 상태가 되어 220V를 내놓으므로 GPIO의 5V를 출력하는 Pin을 VCC로 연결한다. * Relay의 입력 단에 wire로 연결을 할 때 피복이 벗겨진 부분이 보이지 않도록 길이 조정을 하고 보일 시 절연 테이프로 막아줌.  1. Relay 소자의 입력 단에 1 또는 0을 주는 GPIO Pin을 연결해 1이면 ON, 0이면 OFF되게 하는 Toggle 기능을 구현함. 2. 이를 통해 Relay 소자의 작동을 파악하고 신호를 통해 제어 할 수 있음을 확인 함.   5. PUSH BUTTON Interrupt Handler와 Switch   1. 외부 Controller가 없는 경우 DE1-SoC의 PUSH BUTTON을 사용해 제어함. 2. PUSH BUTTON이 눌릴 때 마다 PUSH BUTTON Interrupt Handler를 호출함. 3. 각각의 KEY[0]~KEY[2]는 멀티탭의 각 PORT의 ON/OFF 기능을 담당함.  * 각 PORT의 ON/OFF는 VGA에 ON이면 녹색, OFF이면 빨강으로 나타냄.  1. KEY[3]은 외부 Controller에 있는 기능 몇가지를 가져와서 switch와 같이 사용함.  * sw[0]은 PORT3을 가리키며 high로 한 후 KEY[3]을 누르면 10초 증가함. * sw[1]은 PORT2을 가리키며 high로 한 후 KEY[3]을 누르면 10초 증가함. * sw[2]은 PORT1을 가리키며 high로 한 후 KEY[3]을 누르면 10초 증가함. * 위의 동작들은 동시에 작동 가능함. * sw[3]를 high로 두고 sw[0]~sw[2]중 각각 high이면 해당 PORT의 Timer 초기화 및 전원 OFF를 수행함.   6. Controller   1. Bread board에 IC 소자로 Controller 구현함. 2. N555 Timer IC를 기반으로 일정주기마다 High 신호를 반복해 출력하는 Clock 모듈을 제작함. 3. LAN선을 통해 Bread board에 DE1-SoC의 GPIO와 Controller를 연결함. 4. Bread board에 VCC, GND, pin data, Clock 신호 등을 전달함.  * 8bit의 값이 매 clock마다 1개씩 Bread board에 들어오는데 이는 사람 눈으로 확인 할 수 없이 매우 빠르게 나감.  1. GPIO Interrupt Handler에서 데이터가 도착하면 MFPSD 프로토콜을 참조하여 입력한 값을 입력버퍼에 저장함. 2. Monitor Program의 Terminal을 이용해 무한 루프를 돌며 Input Buffer가 비어있지 않을 때 마다 buffer의 값을 printf로 출력하는 코드를 작성함.  * 이 때 받은 Buffer를 해석해서 해당 값이 뜻하는 작업 수행함.  1. 버튼을 눌러 해당 값이 정상적으로 출력 되는 것을 확인하고 Controller의 통신 기능을 확인함.   7. Input Buffer   1. Controller 정상 작동을 확인 후 무한 루프에서 버퍼에 저장된 값이 있으면 해당 값에 따라 GPIO를 제어하는 코드를 작성함.  * Controller 확인 부분에서 printf 함수로 버퍼의 내용을 출력하는 기능 제거 후 실행함.  1. GPIO Interrupt Handler의 회로가 정상작동 된 것을 확인하고 사용자가 입력한 값에 맞게 멀티탭 PORT를 제어하도록 함. 2. 멀티탭 PORT가 Input Buffer의 변화에 따라 잘 변하는지를 확인하고 Input Buffer의 기능을 검사함.   8. 최종 검증 및 분석   1. Relay 소자가 연결 된 회로에서, Push Button Interrupt 발생 시 GPIO의 Relay 소자의 상태를 프로그램 상에서 선언된 status 변수에 의해 제어되도록 수정함.  * 매 clock마다 값을 받아 status 변수를 최신화 함.  1. VGA에서 Relay 소자의 상태에 해당하는 변수와 Timer Toggle 기능을 위한 Timer 출력이 정상적으로 작동 되도록 수정함. 2. Time Out이 발생 시 Timer ISR에서 각 Relay 별 Timer Toggle이 Enable일 경우 Timer 값을 1 감소시키도록 함. 3. Main에서 각 PORT의 Timer 값이 0일 경우, status 변수를 ON이면 OFF, OFF이면 ON의 되도록 수정함.  * Timer 값은 최대 99이므로 1분 39초를 상한선으로 잡음.  1. Controller와 KEY는 Push button 과 GPIO ISR을 이용해 Main에 선언 된 입력 버퍼에 값을 추가함. 2. Main의 무한 루프를 돌 때마다 Input Buffer가 비어있는지 확인함. 3. Input Buffer가 비어있지 않고 값이 있으면 값을 가져와 해당 동작을 수행함. 4. 위 순서를 통해 기능전원제어장치의 모든 기능 확인함. | | | | | | | | |
| **추진** **일정** **대비** **진행** **결과** | |  |  |  | | --- | --- | --- | | 주차 | 진행계획 | 해당 주차 Demo 내용 및 진행 결과 | | 1 | - Altera manual을 통해 GPIO의 JP1 expansion parallel port와 JP2 expansion parallel port의 각각의 address map이 무엇을 의미하는지 이해한다.  - Bread board에 Pull Down 저항을 바탕으로 Push Button 스위치회로와 LED 점멸회로를 구성함.  - C언어를 사용해 코드를 작성해 KEY를 눌렀을 때 Bread board를 통해 GPIO에 연결된 LED가 ON/OFF 되는 것을 확인해 GPIO를 통해 전원 제어가 되는 것을 확인한다.  - GPIO에 연결된 PUSH BUTTON을 통해서 누를 때 마다 De1-SoC의 Switch에 불이 켜지는 것을 확인해 외부의 입력과 De1-SoC가 통신 되는 것을 확인한다.  - 위를 통해서 GPIO를 Polled I/O와 Interrupt로 받는 것을 확인한다.  - 외부 Controller에 어느 기능을 넣을 것인지 결정하고 그에 대한 회로도를 설계한다. | - DE1-SoC의 KEY Interrupt가 발생하면 GPIO를 통해 Bread board에 연결 된 LED가 KEY가 눌리면 ON이 되고, 떼면 OFF 되는 것을 확인했다.  - DE1-SoC의 Switch가 GPIO의 Polled I/O를 통해 Bread board에 연결 된 Push button을 누르면 ON이 되고 떼면 OFF 되는 것으로 전원이 toggle 되는 것을 확인했다.  - Logic works 프로그램을 이용해서 외부 Controller가 매 clock마다 입력해준 값이 preamble과 함께 출력되는 것을 확인했다. | | 2 | - 설계한 외부 Controller를 Bread board에 직접 구현하여 입력해준 값이 연결된 LED를 통해서 매 clock 마다 나오는 것을 확인한다.  - 본체 하드웨어의 멀티탭이 short가 나지 않고 전자 제품을 꽂았을 때 정상작동 하는지 직접 전원을 제어해서 확인해 본다.  - 멀티탭에서 직접 입력을 주기 위해서 멀티탭을 분해 후 +를 주는 부분을 개조해서 Relay 소자와 연결을 해서 Relay 소자에 5V를 주면 멀티탭에 전원이 가도록 한다.  - 실제로 전자제품을 개조한 멀티탭에 꽂아서 전원이 켜지는 것을 확인한다.  - VGA에 나타낼 0~9까지의 숫자의 bigmap을 다른 파일에 저장 후 header 파일로 사용해 main.c에서 참조하도록 한다.  - 각 PORT의 ON/OFF 상태도 VGA에 나타내는데, ON이면 녹색, OFF이면 빨강으로 나타낸다.  - Switch를 사용해서 KEY Interrupt가 발생 시 Switch의 값을 읽어 해당 기능을 수행한다.  - 사용자가 DE1-SoC보드에서 KEY를 누르면 해당 KEY의 기능에 따라 VGA에 나타낸다.  - Timer Interrupt를 사용하여 Time Out이 발생할 경우 ON이 되어 있는 PORT의 Timer 값을 1 감소시킨다.  - KEY0를 누르면 PORT3의 전원을 ON/OFF하는 것으로 VGA에 해당 PORT3의 전원이 녹색/빨강으로 Toggle 되게 하며 Timer 또한 ON인 경우 시간이 흐르도록, OFF인 경우 멈추도록 한다.  - KEY1과 KEY2를 누르면 KEY0를 누른 것과 같이 각각 PORT2와 PORT1을 동작시킨다.  - KEY3은 Switch와 동기화해서 해당 Switch를 누르고 KEY3을 누르면 작동하도록 한다. | - 외부 Controller가 Bread board 상에서 입력 해준 값이 매 clock에 따라서 preamble 4 bit와 Input bit인 4 bit로 총 8 bit가 입력이 되어 연결된 LED를 통해 값이 잘 나오는지 확인했다.  - 초기에 설정해준 Timer 값이 VGA에 잘 나오는 것과 DE1-SoC의 KEY를 눌렀을 때 각각의 KEY가 가리키는 PORT가 ON이 되어 Timer가 흐르는 것을 확인했다.  - ON 상태에서 다시 KEY를 누르면 OFF가 되며 Timer가 멈추는 것을 확인했다.  - sw[0]~[2]는 각각 PORT3~1을 가리키며 해당 Switch가 high일 때 KEY3을 누르면 Timer가 10초 증가하며 sw[3]과 해당 Switch들이 high일 때 KEY3을 누르면 전원 OFF 및 Timer 초기화가 되는 것을 확인했다.  - 제작한 외부 Controller와 본체 하드웨어를 연결해서 전자 제품이 연결 되었을 때 전원이 켜지는 것을 확인했다. | | 3 | - 본체 하드웨어와 외부 Controller를 LAN 케이블로 연결해 통신하도록 한다.  - main.c에서 외부 Controller에서 입력해주는 값을 Input Buffer로 받아서 무한 루프를 돌 때 마다 이를 계속 확인해 주며 값이 있으면 값을 가져와 해당 동작을 수행한다.  - 외부 Controller를 LAN 케이블을 빼면 DE1-SoC의 PUSH BUTTON과 Switch를 통해 제어가 되도록 한다. | - 직접 멀티탭의 전원을 제어해서 ON/OFF를 시키는 것이 아니라 DE1-SoC의 값과 외부 Controller에 의해서 전원이 ON/OFF되고 Timer입력 값 또한 정상 작동이 되는 것을 확인했다.  - 모든 것을 합하고 VGA에 현재 상황이 잘 나타나는 것을 확인했다. | | | | | | | | | |
| **역할** **분담** **및** **개인별** **공헌** **내용** | | ▶김영찬( 2013122041 )  - 팀장으로써 팀의 역할 분담과 프로젝트의 방향을 결정한다.  - 외부 Controller와 본체 등 모든 하드웨어에 사용 될 재료를 구매 한다.  - Controller에서 DE1-SoC에 data값과 clock을 보내 전원을 제어하는 Controller을 설계하고 제작한다.  - 외부 Controller를 제작 후 입력해주는 값이 제대로 나오는지 검증한다.  - 최종발표 진행 시, 추가 코딩 문제를 해결하는 역할을 담당한다.  - 본체 하드웨어와 외부 Controller간 LAN선을 통해 통신이 정상적으로 되는지 확인한다.  - 외부 Controller에서 오는 값을 Input buffer를 통해 저장한다.  - Input buffer에서 오는 값을 해석해 main.c에서 해당 동작을 수행하게 한다.  - Bread board위에 IC소자를 배치해 외부 Controller를 만든다.  - 멀티탭과 GPIO의 연결 또는 외부 Bread board와 GPIO를 통한 DE1-SoC 연결을 확인한다.  - Relay 소자에 대해 조사를 하고 이를 Bread board 상에서 정상 작동하는 것을 확인한다.  ▶이민우( 2014122191 )  - 프로젝트에 사용 되는 전체적인 Flow Chart를 구상해 설계한다.  - Timer interrupt handler의 코드를 작성해 시간에 따른 Timer의 값을 감소시킨다.  - PUSH BUTTON interrupt handler의 코드를 작성해 각각의 KEY에 대해 어느 기능을 할당할지 정하고 이를 DE1-SoC의 보드상에서 동작시킨다.  - Switch를 PUSH BUTTON interrupt가 발생했을 때 값을 받아와 해당 Switch의 기능을 수행한다.  - JP1 Expansion parallel port와 JP2 Expansion parallel port에 대한 Interrupt handler 코드를 구상하고 작성해 멀티탭 또는 시험용 회로의 LED에 입출력하게 한다.  - 멀티탭과 DE1-SoC, 외부 Controller와 연결을 하는 Bridge인 Bread board가 내장되어 있는 본체 하드웨어를 설계하고 제작한다.  - Timer의 값을 모니터에 초 단위로 나타내고 DE1-SoC의 GPIO를 통해 연결된 멀티탭의 각 PORT의 ON/OFF 전원 상태를 나타낸다.  - 외부 Controller가 없을 시 DE1-SoC를 사용해 멀티탭을 제어할 수 있게 한다.  - 최종발표용 PPT를 제작하고 질의 응답을 담당한다. | | | | | | | | |
| **현실적** **제한** **요건** **반영** **결과** | | 경제요건 | | 외부 Controller를 제작하는데 많은 IC소자들이 필요하기 때문에 경제적으로 제한이 있었고, Relay 소자의 경우 제조사에 따라 필요한 경제요건이 다르기 때문에 성능대비 합리적인 소자를 선택해야 했다. | | | | | | |
| 사회적 영향 | | 현대에 주변에서 자주 사용되는 4차 산업혁명중 하나인 사물인터넷과 비슷한 원리로 직접 제어하지 않고 원격으로 제어를함으로써 간단한 원리로 4차 산업혁명을 이해한다. | | | | | | |
| 미적 요소 | | 외부 Controller의 회로도가 복잡하기 때문에 많은 wire를 사용해서 겉으로 보기에 복잡해 플라스틱으로 포장을 하기에는 제한적이었기에 하드보드지를 사용해 숫자 버튼과 화살표를 나타내고 본체 하드웨어 또한 상자로 제작해 PC같은 느낌을 받을 수 있었다. | | | | | | |
| 보건 및 안전 | | 멀티탭을 분해하고 각 PORT에 연결되어 있는 금속띠를 개조해서 wire를 하나씩 빼 사용자가 직접 전원을 제어할 수 있게해 멀티탭 내의 단선으로 인한 화재 또는 감전의 위험이 있어 전기가 통하지 않는 글루건으로 접착 및 감전의 위험을 줄였다. | | | | | | |
| 내구성 | | DE1-SoC의 GPIO, PUSHBUTTON, Switch를 사용해 제어를 하므로 내구성이 있지만 외부 Controller의 경우 외부 충격에 의해 소자가 빠지거나 wire가 끊어질 수 있어 조작판과 함께 보호판을 만들었다. | | | | | | |
| 산업 표준 | | 구형 멀티탭은 나사를 풀어 내부를 쉽게 확인할 수 있지만 신형 멀티탭은 밀봉되어 쉽게 열 수 있는 구조가 아니다. 또한 구형 멀티탭은 wire로 연결 되어있어 선을 하나씩 빼서 연결하기 쉽지만 신형 멀티탭은 금속띠로 연결이 되어 있다. | | | | | | |
| **토의** | | 우리의 목표는 직접 전원을 제어하지 않아도 외부 Controller를 통해 전원과 원하는 기능을 제어하는 다기능전원제어장치이다. 실험실이나 직장에서 전원을 공급해주는 콘센트를 직접 빼고 꽂는데, 물이 묻는 경우나 특수한 경우에 감전의 위험성이 있고 수동으로 해야한다는 것에 불편함이 있다. DE1-SoC의 Switch 또는 PUSHBUTTON같이 직접적으로 입력하는 제한적인 입력 방법에서 더 다양한 입력을 주고 싶어서 GPIO를 사용해 외부 Controller를 제작했다.  외부 Controller를 아날로그적으로 설계하지 않고 Keyboard같이 이미 있는 장치를 사용하면 더 편리하고 쉽게 접근할 수 있었지만 tv를 리모컨으로 제어하듯이 멀티탭을 키보드가 아닌 우리만의 Controller를 만들고 싶었고, 전자 하드웨어 설계를 하면서 기초 회로 및 디지털 실험이나 전자 회로 실험 같이 직접 하드웨어를 제작해 학기중에 배운 내용들을 더 이해하고 활용해 경험을 살리자는 의의를 두고 Logic Works를 사용해 회로도를 설계하고 Bread board에 직접 연결했다. Controller를 본체에 넣기에는 부피를 너무 많이 차지하고 미적 요소가 떨어지기 때문에 외부에 따로 만들어서 PC처럼 본체에 Controller를 꽂는 것처럼 만들자고 회의를 하고 본체와 LAN선으로 연결해 실제 기기처럼 보이도록 했다.  흔히 주변에서 전기적인 POWER을 줄 때 실험실에서는 POWER SUPPLY를 사용해서 전원을 주거나 일상 생활에서는 건물 내부의 콘센트에 PLUG를 꽂아 충분한 POWER가 갔지만, 설계를 하면서 DE1-SoC에 VCC 5V를 주는 Pin이 있는 것을 확인하였고, LED와 저항을 사용해 기본적인 회로를 구성해서 연결을 하려고 했다. 신호를 주어 입력하는게 아니라 직접 VCC 5V와 Ground를 통해 연결을 했지만 LED에 불이 들어오지 않았고 이유를 보니 단순히 LED 소자가 좋지 않은 소자였기에 5V가 와도 켜지지 않았던 것이다. 이를 해결하고 VCC가 정상 작동하는 것을 확인하고 Bread board에 Pull down 저항으로 PUSH BUTTON 스위치 회로를 연결해 PUSH BUTTON을 누르면 DE1-SoC의 LEDR0가 ON이 되는 것을 확인하고 마지막으로 GPIO Interrupt를 통해 LEDR0가 ON/OFF 되는 것을 확인하려고 했지만, PUSH BUTTON 소자가 제대로 눌리지 않으면 ON/OFF가 반복적으로 나타나는 노이즈가 발생했다. 이런 이유 때문에 아날로그적으로 구상하면 노이즈와 변수가 많기 때문에 외부 Controller를 통해정확한 값을 주자고 정하는 계기가 되었다.  또한 DE1-SoC의 GPIO에 대해 알아보기 위해 자료 조사를 했지만 실질적으로 DE1-SoC 보다는 라즈베리파이를 사용해서 코드를 작성하고 회로를 연결한 경우가 많았다. 라즈베리파이에 대해 알아보니 초소형/초저가의 교육용 목적의 컴퓨터이며 센서가 많아 여러 분야에서 적용가능한 보드였다. DE1-SoC와 비슷한 느낌을 받았지만 아두이노처럼 C언어로 작성하면 길어지는 코드가 하나의 명령어 안에 포함되어 있는 코드 구조였다. 이를 통해 C언어나 ASSEMBLY 코드 하나의 명령어에도 프로그래머가 플로우 차트를 구성하는 것을 이해하고 프로그래밍이란 넓은 분야에 적용되고 있다는 것을 확인했다.  00년대의 구형 멀티탭을 개조한 경험을 통해 다시 멀티탭을 개조하려고 했지만 구형 멀티탭에서는 분해를 할 수 있었지만 신형 멀티탭은 안전과 내구성을 생각해 분해하기 힘든 구조로 변했다는 것을 깨닫고 진행계획에 차질이 생길 뻔 했지만 멀티탭을 겨우 개봉을 했다. 또한 내부를 살펴보니 기술의 발전으로 인해 금속띠를 사용해서 전기가 더 잘 통하게 하고 이전의 전선이 끊어지는 걱정 없이 내구성을 증가시켰다. 이렇게 주변에서 흔히 볼 수 있는 기본적인 생활 용품들도 기술의 발전과 함께 변해간다는 것을 느꼈다.  전원을 제어하는 것이 목표이다 보니 설계자 입장에서는 직접 전원을 주면서 제대로 연결이 되어있는지 확인을 하고 이를 작성한 코드와 합쳐서 구현하는 과정을 겪었다. 전원을 제어하는데 합선 또는 감전의 위험성이 있어서 전기가 통하지 않는 글루건으로 접착을 하는 동시에 감전의 위험성을 줄이고 멀티탭은 실패의 경우 열어서 다시 손을 보기 위해 절연테이프로 재봉합하며 Relay 소자에 같은 신호를 주기 위한 wire들은 최대한 피복을 짧게 벗겨서 꼰 후 절연테이프로 감쌌다. 또한 코드가 아닌 직접 전원을 주어 헤어드라이기와 스탠드가 작동하는 것을 확인 할 때 모든 위험성을 줄였지만 불이 나거나 합선 되어 두꺼비 집이 내려가면 어떻게 할 지 걱정이 많았지만 정상적으로 회로가 연결되었을 때 전자공학도로써 배운 것을 통해 위험성이 있는 전원을 제어했다는 것에 성취감과 자부심을 느꼈다.  우리는 프로젝트를 진행하면서 전원을 제어하겠다는 목표를 너무 쉽게 설정했다고 느꼈다. 물론 Relay 소자와 AND게이트와 같은 IC 소자를 사용했지만 일상에서 당연하다고 생각한 필수 요소인 전기적 POWER를 제어한다는 익숙하지 않은 주제와 이전에 컴퓨터 프로그래밍 과목을 수강하면서 배웠던 C언어를 사용해 코딩을 해야한다는 것에 대해 부담이 작용한 것 같다. 이번 프로젝트를 진행하면서 아무리 기본적인 것이더라도 기본적일 수록 더욱 더 철저한 사전조사와 이를 이해하는 것에 대한 필요성을 느꼈다. 이번 프로젝트에서도 멀티탭이 구형에서 신형으로 바뀐 것을 미리 파악했더라면 급하지 않고 우리가 구상한 대로 회로가 작동하고 작성한 코드대로 작동하며 체계적인 설계를 통해 더 좋은 결과를 낼 수 있었을 것이라 생각된다. | | | | | | | | |