|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **교과목 명** | | 전자 하드웨어 설계 | | | | | | | |
| **설계 제목** | | 다기능 전원 제어 장치 ( Multi-Function Power Supply Device ) | | | | | | | |
| **설계 기간** | | 2018년도 2학기 | | | | | | | |
| **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  C언어를 기반으로 3구 멀티탭에 각각 연결된 물체의 전원을 제어하고 기능을 선택적으로 조작하는 controller를 만든다. 하드웨어인 본체를 직접 제작해, 멀티탭의 각 구의 선을 스위치 역할을 하는 릴레이와 연결하여 사용자가 원할 때 멀티탭의 각 포트별로 전원을 제어 할 수 있도록 한다. De1-SoC보드에 있는 GPIO 포트의 VCC, GND와 여러 Data Pin을 외부에 연결되어 있는 controller와 연결해서 사용자가 입력한 data를 매 clock마다 De1-SoC에 받아온다. De1-SoC는 Interrupt Handler를 통하여 GPIO 포트에서 받아온 값을 임의로 만든 프로토콜을 기반으로 해석해 Input Buffer에 저장하고, 구현한 프로그램은 Input Buffer에 있는 값을 읽어들여 수행하고자 하는 동작을 실행한다.    **< 그림 1 - 다기능 전원 제어 장치의 설계 기획도 >**  2. 기능   1. 본체 외부에 연결된 controller로 멀티탭 제어  * Bread board의 IC 소자를 기반으로 구성한 controller에서 clock과 data값을 De1-SoC에 전달하고, De1-SoC는 이를 입력 받아 상황에 맞는 값을 출력해 멀티탭에 해당 구에 보내 작동시킴. * controller에서 보내는 data값은 각 구에 얼마나 작동 시킬지인 타이머 값과 멀티탭의 어느 구를 작동 시킬지에 대한 값임. * controller가 8bit 크기의 데이터를 보낼 때, De1-SoC가 해독 함에 있어서 오류를 적게 하기 위해 Preamble 방식을 사용함.  1. 모니터에 현재 상황 출력  * De1-SoC의 VGA포트를 통해 모니터에 해당 포트가 ON이면 녹색, OFF이면 빨간색으로 표시가 됨 * 입력해준 타이머 값을 초 단위로 모니터에 나타냄.  1. 외부 controller가 없을 시 PUSH BUTTON으로 제어  * 위 그림에서 외부 controller의 입력 단자에서 LAN 케이블을 제거할 시 De1-SoC의 PUSH BUTTON을 사용해 제어 함. * 해당 구를 확인 하기 위해 Switch를 사용하여 01이면 첫번째 구, 10이면 두번째 구 그리고 11이면 세번째 구를 가리키게 함. * KEY[0]는 전원을 TOGGLE해 ON/OFF 시켜주며 KEY[1]는 10초, KEY[2]는 20초, KEY[3]은 30초를 타이머로 설정 함.  1. 릴레이 부품 사용한 전원 제어  * De1-SoC에 있는 GPIO를 이용해 멀티탭의 릴레이 소자를 조작하여 220V를 제어함. | | | | | | | |
| **설계 규격** | 1. GPIO를 주로 사용하여 입출력을 받으므로, GPIO의 규격을 설명함.   1. FPGA의 Parallel ports를 사용함. 2. De1-SoC의 2개의 GPIO0와 GPIO1은 각각 JP1 과 JP2 expansion parallel port 임. 3. JP1과 JP2 각각 32bit의 양방향 핀으로 구성. 4. GPIO의 사용 가능한 각 핀은 input이 될 수도 있고 output이 될 수도 있음. 5. JP1과 JP2의 address base는 0xFF200060과 0xFF200070 임. 6. GPIO의 address map은 4개로 구성 되어 있고 각각 32 bit임..  * base : pin으로 부터 데이터값을 읽어오거나 pin에 데이터값을 쓰는 data register으로 읽고 쓰기가 가능함. * base + 4 : pin을 0으로 설정하면 input이고 1으로 설정하면 output으로 환경을 설정해 주는 Direction register으로 읽고 쓰기가 가능함. * base + 8 : 각각의 pin의 interrupt를 enable해주는 Interrupt Mask register이며, 해당 pin은 “base + 4”에서 0으로 설정해 input 상태로 만들어 줘야함. * base + 12 : interrupt 발생시 어느 bit가 변화했는지 확인하는 Edge capture register으로, 해당 bit를 1을 써서 clear해주고 전체 bit를 clear해주기 위해서는 0xFFFFFFFF을 씀.  1. Direction register를 0또는 1로 설정해 초기치를 설정함. 2. 어느 interrupt가 enable되면 입력 pin의 값이 1에서 0으로 바뀌는 Negative edge임. 3. NIOS ||에서 JP1과 JP2의 interrupt level은 11과 12임. 4. Edge capture register의 bit가 high값일 때 인터럽트가 트리거 되기 때문에 인터럽트를 enable하거나 interrupt handler를 종료하기 전에 Edge capture register를 초기화 해줘야함. 5. JP1과 JP2 모두 1과 3번 pin은 clock in을 하고 19와 21번 pin은 clock out을 하는 pin임. 6. 1번 pin은 5V가, 29번 pin은 3.3V가 인가되는 pin이며 각각 다음 pin인 12와 30번 pin은 접지해주는 GND임.   2. De1-SoC의 VGA포트를 통해 모니터에 나타내기 때문에 모니터에 나타내는 과정을 설명함.   1. FPGA의 Video-out-port를 사용함. 2. Pixel Buffer register의 base는 Buffer으로 Buffer의 시작 주소임. 3. base + 4는 Back buffer 주소임. 4. make frame과 draw frame을 동기화 시키고 make frame이 준비되면 buffer에서 front buffer와 back buffer를 swap해주어 V-Sync와 Double buffering을 해줌. | | | | | | | |
| **합성/분석** | **관련 기술** | 1. De1-SoC 보드와 Nios II 프로세서   1. 해당 프로젝트를 구현하기 위한 하드웨어로는 Terasic사의 De1-SoC보드를 사용함. 2. 프로그램은 De1-SoC의 FPGA 프로세서인 Nios II CPU를 사용함. 3. Nios II 프로세서는 다음과 같은 특징이 있음.  * Reduced Instruction Set Computer (RISC) Architecture임. * 레지스터와 메모리는 Load/Store Instruction으로 구성되어 있음.. * 산술 또는 논리 연산이 General Purpose Register에서 이루어 짐. * 모든 Register와 Instruction이 4Byte, 즉 32bit로 정형화 되어 있음. * Nios II / fast , Nios II / Standard , Nios II / economy 라는 3개의 설정이 존재함. * Little-Endian 방식으로 데이터를 저장함.     2. 릴레이 ( Relay ) 소자   1. 장치의 제어, 회로의 보호등을 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 - 릴레이 소자의 내부 동작 >**     1. 릴레이는 내부에 전자석 코일을 포함하고 있어서 전류가 통하게 되면 자기장을 형성한다. 이 자기장은 릴레이 내부의 POLE을 움직여 전자 회로의 개폐를 조작할 수 있음. 2. 이 릴레이 소자를 통해 De1-SoC의 GPIO로 멀티탭을 제어함.     3. 프리앰블 비트 ( Preamble Bit )   1. 비트를 신호화 할 때, 비트의 타이밍을 송신 측과 수신 측 양쪽에서 일치시키기 위해 Data의 선두에 지금부터 데이터 프레임이 시작된다는의미의 Preamble Bit를 추가하여 송신함. 2. 주로 Ethernet 통신에서 Ethernet Frame 동기화를 위해 사용됨. 3. 이번 프로젝트에서는 De1-SoC와 사용자 controller의 Bit 동기를 맞추기 위해 사용됨.     4. 입력 버퍼 ( Input Buffer )   1. 이번 프로젝트에서는 GPIO Interrupt, PushButton Interrupt등 여러 Interrupt를 사용하고, 해당 Interrupt별로 이를 제어하기 위한 Interrupt Handler가 필요함. 2. 이 때, Interrupt Handler에 해당되는 기능을 기술하는 것은 바람직하지 못한 프로그래밍 방식이라고 생각됨. 3. 예를 들어, 키보드의 경우에도 키보드는 직접 필요한 기능을 수행하는 것이 아니라 단지 Buffer에 해당하는 값을 넣는 입력 장치임. 키보드의 입력이 필요한 장치는 Buffer에 저장된 값을 읽어 오는 것으로 동작함. 4. 마찬가지로, 이번 프로젝트의 PushButton Interrupt Handler와 GPIO Interrupt Handler는 특수한 기능을 수행하는 것이 아닌, 사용자가 입력한 값을 Input Buffer에 넣는 기능만을 수행함. 5. 구현할 프로그램은, Loop를 돌며 Input Buffer가 비어있는지 확인하고, 아니라면 Buffer의 값을 읽어와 사용자가 입력한 기능을 수행하도록 설계함. 6. C언어로 Buffer를 구현하기 위해서는 데이터가 넣어질 위치인 in이라는 함수와 데이터를 빼내야할 위치인 out이라는 변수를 사용해야함. | | | | | | | |
| **제작** | **설계 계획** | 1. 하드웨어 아키텍처 ( Hardware Architecture )   1. 이번 프로젝트는 De1-SoC를 이용한 프로젝트로서, Nios II 프로세서를 사용함. 2. 또한 각종 I/O 의 Interrupt와 Data값을 읽거나 쓰기 위해 장치에 접근해야 할 때에는 Memory Mapped I/O를 사용하는 De1-SoC의 특성에 맞게 해당 주소값을 통해 데이터를 읽어오기로 함. 3. 사용될 I/O 장치는 크게 4분류로 아래와 같다.  * 사용자 controller의 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 확인 할수 있게 하는 VGA         **< 그림 - 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-SoC의 Key와 Switch만으로는 부족하다고 판단됨. 2. 이를 위해 사용자가 좀 더 직관적으로 장치를 사용 할 수 있도록 따로 특수한 controller를 만들기로 함. 3. 해당 controller는 De1-SoC와 직접 통신이 가능해야하는 전자 부품으로써, Bread Board 위에 IC소자를 사용하여 구현하도록 함. 4. 해당 controller는 0~9까지의 숫자 버튼과, 확인/취소 버튼, 그리고 UP / DOWN / LEFT / RIGHT 의 4방향 버튼으로 구성되어 있으며, 이 16개의 버튼을 통해 사용자는 원하는 기능을 수행 할 수 있음. 5. 16가지 버튼을 De1-SoC에게 전달하기 위하여, 8-to-3 Encoder인 SN74LS148(DIP) 2개 사용하여, 16가지 버튼의 경우의 수를 6Bit로 전달함.         **< 그림 - SN74LS147의 Pin Arrangement >**     1. controller 모듈은 어떤 버튼이 눌렸는지를 나타내는 6Bit Data 외에도 De1-SoC의 해석을 도와주기 위해 Preamble Bit로써 11의 2 Bit를 합쳐서 같이 보내기로 함.. 2. 따라서 De1-SoC와 controller는 VCC, GND, Data, Clock의 4 Bit로 통신을 함.     4. MFPSD 프로토콜 ( Multi-Function Power Supply Device Protocol )   1. controller가 입력한 버튼에 따라서 De1-SoC에 입력되는 비트가 달라짐. 2. controller는 이를 Clock 신호에 맞춰서 1 Bit씩 보내는 Serial 통신으로 구현함. 3. 이때 송신되는 Bit는 아래와 같은 프로토콜을 기반으로 보내지며, De1-SoC도 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 8키 | | 1 | 000001 | 숫자 1키 | 9 | 010000 | 숫자 9키 | | 2 | 000010 | 숫자 2키 | 10 | 011000 | 확인/선택 키 | | 3 | 000011 | 숫자 3키 | 11 | 100000 | 취소/지움 키 | | 4 | 000100 | 숫자 4키 | 12 | 101000 | LEFT 키 | | 5 | 000101 | 숫자 5키 | 13 | 110000 | RIGHT 키 | | 6 | 000110 | 숫자 6키 | 14 | 111000 | 사용하지 않음 | | 7 | 000111 | 숫자 8키 |  |  |  |     **< 표 1 - MFPSD 프로토콜 표 >**     1. 사용자 controller에서 언급했듯이, 버튼을 누르면 controller는 11 이라는 2 Bit Preamble 뒤에 버튼에 해당하는 6 Bit 코드를 추가하여 8 Bit 신호를 De1-SoC에게 Clock 신호와 함께 전달함. 2. De1-SoC는 전달받은 Clock에 맞춰 8 Bit Data를 해석하고, MFPSD 프로토콜 표를 참고하여 판단함.     5. 입력 버퍼 ( Input Buffer)   1. 이번 프로젝트에서는 1024 Byte의 링 형태의 Buffer를 사용함. 2. 버퍼와 관련된 핵심 코드는 아래와 같음.      |  | | --- | | 기본 선언 | | # define BUFFER\_SIZE 1024; // 버퍼의 사이즈. 여기서는 1024Byte로 정함.  char buffer inputBuffer; // Char형 버퍼를 만듦  int in = 0; int out = 0; // 버퍼의 읽고 쓰는 위치를 기록 |      |  | | --- | | Buffer를 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 in값을 증가시킴  } |      |  | | --- | | Buffer에서 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 out값을 증가시킴  return result; // 꺼내온 값을 반납함  } |     **< 표 2 - 입력 버퍼 구현에 관한 핵심 코드 >**     1. in와 out 변수는 버퍼를 읽거나 쓸 때마다 자신이 사용되면 1 증가함. 이때, BUFFER\_SIZE보다 크다면 0으로 초기화 되어 링 형태의 버퍼를 구현 할 수 있음. 2. 프로그램은 Loop를 돌며 Buffer의 값을 가져온다.     6. 소프트웨어 아키텍처 (Software Architecture)   1. 해당 프로그램의 프로세스는 주로 main안의 while(1)의 무한 Loop를 돌며 VGA를 통해 화면을 출력함. 2. Interrupt Handler는 Interrupt를 호출한 원인을 분석하여 TimerISR, PushButtonISR, GPIO ISR을 호출함. 3. Timer Interrupt는 멀티탭 포트의 Timer Toggle 기능이 Enable되어 있다면 해당 포트의 Timer Value를 1 감소 시킴. 4. Push Button ISR은 입력한 KEY를 분석하여, 입력 버퍼에 저장함. 5. GPIO ISR은 입력한 버튼을 분석하여, 입력 버퍼에 저장함. | | | | | | | |
| **시험/평가** | **검증 계획** | 1. GPIO Interrupt Handler 검증   1. De1-SoC의 GPIO와 Bread Board를 연결함. 2. Bread Board에 Pull Down 저항을 바탕으로 한 Push Button 스위치 회로를 구성함. 3. Bread Board에 1K Ohm 저항과 LED로 이루어진 LED 점멸 회로를 구성함. 4. Push Button을 통해 LED을 Toggle 시키는 GPIO ISR을 구현함. 5. 이를 통해 GPIO의 ISR인 GPIO Interrupt Handler의 기능을 검사함.   2. TImer Interrupt Handler와 VGA의 검증   1. VGA를 사용하기 위해 C언어 코드로 main.c를 작성함. 2. VGA를 이용해 모니터에 지정된 2자리 숫자를 띄우는 코드를 작성함. 3. 이를 함수로 따로 만들어 지정된 숫자를 개발자가 원하는 위치에 원하는 크기로 출력하도록 구현함. 4. Timer Interrupt를 사용하여 설정한 시간이 될 때 마다 Timer Interrupt Handler를 호출시킴. 5. Timer Interrupt Handler는 임의의 숫자는 1 감소시킴. 6. Timer Handler에 의해 줄어든 숫자가 출력되는 것으로 Timer Interrupt와 VGA의 기능을 검사함.   3. Relay 소자의 검증   1. 검증 (1)을 완료함. 2. 검증 (1)의 회로에서, 1K Ohm 저항과 LED를 제거하고, Relay 소자를 연결함. 3. Relay 소자에 De1-SoC에 있는 GPIO의 VCC와 GND를 연결함. 4. Push Button은 검증 (1)과 마찬가지로, Relay 소자를 Toggle하는 기능을 구현함. 5. 이를 통해 Relay 소자를 제어 할 수 있음을 검사함.   4. Controller의 검증   1. Bread Board에 IC 소자를 배치하여 Controller를 구현함. 2. N555 Timer IC와 Capacitor, 저항을 이용하여 일정 주기마다 High 신호를 반복해 출력하는 Clock 모듈을 만듦. 3. LAN선을 통해 De1-SoC에 있는 GPIO가 연결된 Bread Board에 Controller를 연결함. 4. 해당 Bread Board에 VCC , GND , Serial Data , Clock 신호등을 전달함. 5. GPIO Interrupt Handler에서, 데이터가 도착하면 MFPSD 프로토콜을 참조하여 입력한 값을 입력 버퍼에 저장함. 6. Altrera Monitor Program의 Terminal을 이용해 무한 Loop를 돌며 입력 버퍼가 비어있지 않을 때 마다 버퍼의 값을 printf 함수로 출력하는 코드를 작성함. 7. 버튼을 눌러서 해당 값이 옳게 출력됨을 확인하고 Controller의 통신 기능을 검사함.   5. Input Buffer의 검증   1. 검증 (4)를 완료함. 2. 검증 (4)의 코드에서, printf 함수로 버퍼의 내용을 출력하는 기능대신, 무한 LOOP를 돌며 버퍼에 저장된 값이 있으면 해당 값에 따라 GPIO를 제어하는 코드를 작성함. 3. 검증 (1)을 완료함. 4. 검증 (1)의 회로에서, 사용자가 입력한 값에 맞게 LED를 제어하도록 수정함. 5. LED가 Input Buffer의 변화에 따라 옳게 변하는지를 확인하고 Input Buffer의 기능을 검사함.   6. 최종 검증   1. 모든 검증을 완료함. 2. 검증 (3)의 회로에서, Push Button ISR에서 GPIO의 Relay소자의 상태를 직접 조작하는 것이 아니라, 프로그램상에서 선언된 status 변수에 의해 제어되도록 수정 3. VGA에서 릴레이 상태에 해당하는 변수와 Timer Toggle 기능을 위한 Timer 출력이 정상적으로 작동 되도록 수정함. 4. Timer ISR이 각 릴레이별 Timer Toggle 기능이 Enable일 경우, Timer Value를 1감소시키도록 수정함. 5. Main함수에서, 각 릴레이별 Timer Value가 0일 경우, status 변수를 수정함. ( ON이면 OFF, OFF면 ON ) 6. Controller와 KEY는 PushButton ISR, GPIO ISR을 이용하여 Main함수에 선언된 입력 버퍼에 값을 추가함. 7. Main함수에서, Loop를 돌 때 마다, 입력 버퍼가 비어있는지 확인 함. 8. 만약 비어있지 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 | - C언어를 기반으로 KEY를 눌렀을 때 GPIO에 연결된 LED를  ON/OFF해서 GPIO를 통해 전원 제어가 되게 한다.  - GPIO에 연결된 PUSH BUTTON을 통해서 De1-SoC의  HEX에 불이 켜지는 것을 확인해 외부의 입력과 De1-SoC가  통신 되는 것을 확인한다. | - GPIO를 통해 외부 controller와 De1 -  SoC 통신하는지 확인 할 수 있음.  - 실제로 값이 제대로 입력 또는  출력이 되는지 LED나 HEX를 통해  확인 할 수 있음. | | 2 | - Controller 자체 하드웨어를 설계 및 제작 해서 동작이 잘  되는지 확인한다.  - 사용자가 주는 입력에 따라 3구 멀티탭이 ON/OFF 되는  것을 De1-SoC의 VGA포트를 통해 모니터에 나타내는 것을  확인하고, 1주차에 진행한 GPIO에 연결된 LED 작동도  확인해 값이 동시에 잘 들어가도록 한다.  - C언어의 코드 상의 타이머를 모니터에 나타내 초 단위로  타이머가 움직이는 것을 확인한다. | - Controller 입력에 따라 LED가  켜지는지 확인할 수 있음.  - 입력해주는 값에 따라 모니터에  ON/OFF 값이 제대로 동기화 되고,  타이머가 초 단위로 작동하는지 확인  할 수 있음. | | 3 | - 멀티탭과 De1-SoC보드와 이를 bridge해주는 bread  board가 들어 있는 본체 하드웨어를 설계 및 제작한다.  - 본체 하드웨어와 외부의 하드웨어인 Controller를 LAN  케이블로 연결해 통신하도록 한다.  - Controller가 있을 때 Controller에서 인가 해주는 입력에  따라 원하는 멀티탭의 해당 포트가 ON/OFF 또는 타이머가  잘 맞는지 모니터에 나타낸다.  - Controller가 없을 때 본체 하드웨어의 De1-SoC의 KEY  interrupt를 사용해 발생 시 Switch의 값을 읽어 멀티탭의  해당 포트를 ON/OFF 또는 타이머를 작동 시켜 이를  모니터에 나타낸다. | - 본체 하드웨어 설계 및 제작해서  Controller와 연결해 Controller에서  입력해주는 data가 clock에 따라 잘  들어오는지 확인하고, De1-SoC와  통신해 나오는 출력값을 멀티탭에  입력하고 멀티탭의 해당 하는 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 2를 작성한다.  - Controller와 본체 하드웨어에 사용될 재료를 구매한다.  - De1-SoC에 data값과 clock을 보내줌으로써 전원을 제어하는 Controller을 설계 및 제작한다.  - 최종 발표를 진행 할 때, 즉석으로 코딩을 수정하는 역할을 담당한다.  - Controller와 본체 하드웨어 사이의 serial 통신이 잘 되는지 확인한다.  - Input buffer를 사용해 Controller에서 오는 값을 저장한다.  - Controller의 하드웨어를 만들기 위해 Bread Board위에 IC소자를 배치한다.  - GPIO와 De1-SoC 또는 멀티탭과 GPIO의 연결을 확인한다.  ▶이민우 ( 2014122191 )  - 전체적인 프로젝트에 사용될 플로우 차트를 구상후 설계한다.  - 프로젝트를 진행하면서 회의록 1,2,3 중에 회의록 1과 3을 작성한다.  - I/O Peripheral에서 Interval timer, Pushbutton switch parallel port, JP1 Expansion parallel port 그리고 JP2  Expansion parallel port에 대한 interrupt handler 코드를 구상하고 작성한다.  - De1-SoC와 멀티탭과 bridge 역할을 하는 bread board가 들어있는 본체 하드웨어를 설계 및 제작한다.  - 타이머를 모니터에 초 단위로 나타내고 멀티탭 각 구의 ON/OFF 상태를 나타낸다.  - Controller에서 오는 data값 또는 KEY interrupt를 발생시킬 때 이에 해당하는 작동을 하게 해준다.  - 최종 발표에 사용할 PPT를 제작한다.  - 최종 발표를 진행 할 때, 질의 응답을 담당한다. | | | | | | | |