

교과목 명		전자 하드웨어설계						
설계 제목		가스밸브 안전 제어시스템						
설계 기간		2017년도 2학기						
지도교수		김태환						
팀원	이름	오헤빈	학번	2015124129	☎	01026995260	E-mail	5binnn@naver.com
	이름	이민호	학번	2012122192	☎	01029281686	E-mail	alsghdjrk@naver.com
	이름	임찬영	학번	2012122246	☎	01039703347	E-mail	Limchan1@naver.com
목표 설정	설계 배경	<p>Lab을 통해 학습한 DE1-SOC 보드의 peripheral과 추가적인 기능 활용, 하드웨어를 추가하여 설계하기로 한다. 또한, interrupt, polled IO방식을 적절하게 사용하여 peripheral을 control하기로 했다. 또, 이외의 외부 모듈(센서, 모터...)을 보드에 연결하여 사용한다.</p> <p>우리는 가스실에서 들어가 작업을 하는 임부와 그 임부에게 컨트롤 하는 커맨더가 하나 있다고 가정하고 프로젝트를 설계한다. 우선 가스실에서 안전은 필수 사항이다. 그렇기 때문에 가스실에서 문제가 될 만한 상황, 위험이 발생했을 경우 밸브를 제어하고 위험을 알릴 수 있는 시스템을 구현하기로 한다.</p> <p>가스 유출과 스파크를 감지 할 수 있도록 하드웨어를 구성하기로 계획 했다. 가스가 감지가 된다면 커맨더는 화면을 통해서 가스가 감지 되었다는 것을 확인하여 무전으로 임부에게 알릴 수 있고 뿐만 아니라 가스나 불꽃이 감지가 되면 위험을 알리는 역할을 하기 위해 경고음을 출력하기로 했다. 또한 가스가 감지가 되면 더 큰 위험을 막기 위해 servo motor를 통해 가스 밸브를 잠그는 것을 구현 했다 이를 위해 DE1-SOC보드의 gpio와 센서, 모터, 버저를 연결 하였다.</p> <p>그 다음 DE1-SOC에 가스실에서 작업자에게 필요한 것은 무엇이 있을지 생각을 해보았다. 우선 가스실에서 작업을 할 때 일정 시간만큼만 작업할 수 있게 구현을 했다. 작업 시간 setup은 KEY를 이용하여 시간 추가, 감소, reset, start/stop를 구현한다. 뿐만 아니라 DE1-SOC로 시간을 알 수 있는 기능 이 있다면 편리할 것이라고 판단이 되어서 현재 시간을 들어가기 전에 설정을 하여 현재 시간과 남은 작업 시간을 돌려가며 확인 할 수 있도록 하였다.</p> <p>그리고 만약 대화를 할 수 없는 작업 공간에 들어갔을 경우를 대비하여 키보드로 입력하여 상황실 커맨더에게 띄워 주거나 인수인계시 필요한 work log를 남길수 있는 기능을 구현 하였다. 키보드 입력방식은 수업시간에 학습한 것이 아니 었지만 수업을 통해 각 memory mapped I/O device의 reference를 보고 그것을 이용하는 방법을 익혔다. PS/2 port의 reference를 참조하여 interrupt방식으로 사용하도록 진행한 다.</p> <p>연결한 하드웨어가 많으므로 switch를 이용하여 세부적으로 control해준다. 예를 들어, key lock을 만들 필요가 있다. 어두운 곳에서 작업을 하는 임부가 만약 실수로 키를 누르거나 하면 작업을 미완성 할 수도 있고 위험 하다고 판단 할 수 있어서 이다. 이처럼 Switch와 key를 이용한 세부적인 기능들이 많기 때문에 우리가 설계한 것의 기본적인 기능들에 대한 간단한 설명을 모니터에 display하여 볼 수 있도록 구현 하였다.</p> <p>위에서 구현한 모든 상태들을 character buffer와 pixel buffer를 이용해 모니터에 display해주도록 한다.</p>						
	설계 목표	<p>이번 프로젝트는 DE1_SOC 과거 실습에서 배웠던 것들(interval timer, hex, led, pushbutton, pixel buffer)을 최대한 이용하고 그 외에 PS/2 port, gpio, character burred, slider switch를 이용한다.</p> <ol style="list-style-type: none"> 1. 설계하고자 하는 시스템은 C언어를 사용하여 구현을 하도록 한다. 위에서 말한 가스밸브 안전제어 시스템으로 위험을 감지하고 이에 따라 위험요소를 제거할 수 있도록 액추에이터를 동작시키고, 항상 그 상태를 display한다. 2. 현재시간과 타이머를 구현한다. 이는 interrupt handling방식으로 2가지 interrupt(timer, push button)와 polled I/O방식인 slider switch를 이용하여 세부적인 기능을 제어할 수 있다. Push button으로 interrupt를 발생시켜 타이머와 현재시간을 세부적으로 설정할 수 있도록한다. 3. gpio에 센서를 연결하여 센서의 digital out값을 이용해 시스템을 제어한다. 센서의 감지여부,타이머의 타임아웃에 따라 모터 작동시킨다. 센서와 마찬가지로 gpio를 통해 연결한다. PWM pulse를 생성해 이를 이용하여 모터를 제어할 수 있도록 한다. 이는 second interval timer를 사용하여 Delay_T() 함수를 						

구현하여 PWM pulse를 출력한다.

5. 위험이 감지되면(센서 감지시) 모니터에 위험이 감지됨을 (gas sensor | frame sensor) ON을 표시 하게 하고, 뿐만 아니라 버저로 경고음이 출력되도록 구현한다. Buzzer는 gpio에 연결하여 상황별로 사운드 ON/OFF 제어할 수 있도록 한다.

6. 키보드를 이용하여 작업 일지를 작성 할수있도록 구현한다. PS/2 port에 키보드를 입력하여 interrupt방식으로 작동할 수 있도록한다. int형으로 키보드를 입력했을 때 들어오는 값을 print로 확인을 하여 각 키 값을 파악한후 사용한다. 문자 27개와 space와 back space를 기능을 구현한다. backspace를 이용할 때 포인터를 이전 자리로 이동 공백을 채워주고 다시 포인터를 뒤로 가도록하여 back space가 정상 작동을 할 수 있도록 했다.

7. 위의 모든 상황을 디스플레이 해준다. Pixel buffer, character buffer를 이용하는 display함수를 이용하여 현재 시간, 타이머의 시간, 세부기능을 설명 등을 모니터에 표시 하도록 하고, 가스센서, 불꽃 센서의 감지여부도 모니터에 문자열로 출력한다. 6번의 키보드입력을 모니터에 character buffer를 이용하여 표시 하도록 한다. flickering현상이 없도록 하고 vertical sync를 맞춰준다.

-Memory mapped I/O

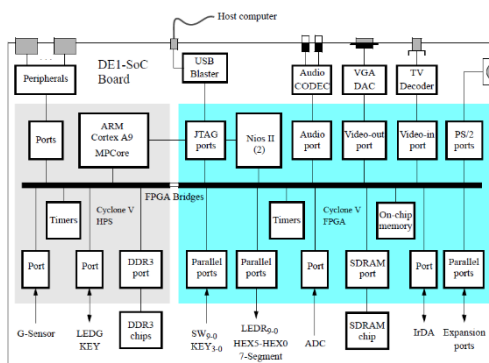


Figure 1. Block diagram of the DE1-SoC Computer.

I/O 작업을 수행하기 위해 메모리의 읽기, 쓰기 작업과 같은 방식으로 처리한다. 입출력 장치의 레지스터에 메모리와 같이 주소를 할당하고, 데이터를 읽고 쓴다. Memory mapped I/O 프로젝트를 진행하면서 NIOS2에서 기본으로 제공하는 address.map_nios2.h를 이용하여 memory mapped I/O를 진행한다. 이 프로젝트에서는 VGA-DAC, parallel ports(expansion ports, SW, KEY, HEX, LEDR), interval Timer, second interval timer, ps2 port를 사용했다.

인터럽트 기반으로 De1-soc 보드의 기능들을 사용하기 위해 I/O Peripheral의 값을 Ienable 부분에 Write 해주는 ctl3 register, STATUS에 인터럽트를 받을 수 있는 PIE의 값을 1로 초기화 해준다. 이 프로젝트에서는 Interval timer, PS/2 port, Pushbutton의 interrupt를 사용했다.

Register	Name	$b_{31} \dots b_2$	b_1	b_0
ctl0	status	Reserved	U	PIE
ctl1	estatus	Reserved	EU	EPIE
ctl2	bstatus	Reserved	BU	BPIE
ctl3	Ienable	Interrupt-enable bits		
ctl4	ipending	Pending-interrupt bits		
ctl5	cpuid	Unique processor identifier		

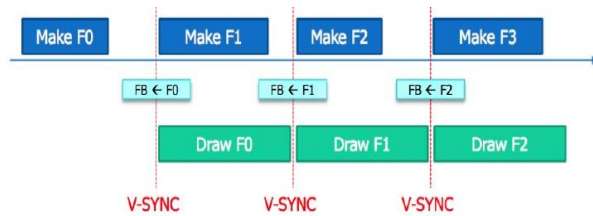
I/O Peripheral	IRQ #
Interval timer	0
Pushbutton switch parallel port	1
Second Interval timer	2
Audio port	6
PS/2 port	7
JTAG port	8
Serial port	10
JP1 Expansion parallel port	11
JP2 Expansion parallel port	12

- Interval Timer

Address	31	...	17	16	15	...	3	2	1	0
0xFF202000	Unused								RUN	TO
0xFF202004	Unused								STOP	START
0xFF202008	Counter start value (low)								CONT	ITO
0xFF20200C	Counter start value (high)									
0xFF202010	Counter snapshot (low)									
0xFF202014	Counter snapshot (high)									

인터럽트를 기반으로 한 Timer이다. 현재 타이머의 시간을 7segment에 나타내기 위해 interval timer 1을 인터럽트방식으로 사용했다. 서보모터와 센서들을 제어하기 위해 interval timer2 를 polled I/O 방식으로 사용했다. status register의 두번째 bit RUN은 TIMER가 동작하기 위해 1로 설정해야 하며 TO는 1초가 count되면 TO가 0에서 1로 바뀐다. 다음 시간을 count하기 위해 TO를 다시 0으로 초기화 해준다. Control register는 STOP, START, CONT, ITO가 있다. STOP은 현재 Interval Timer의 동작을 멈추게 하고 START는 타이머가 동작을 시작함을 나타내고 CONT는 타이머가 계속 동작하게 해준다. ITO는 Interrupt Time Out으로 인터럽트가 발생하면 ITO의 값이 1이 된다 다시 인터럽트로 Interval TIMER

-Double buffering



움직이는 이미지를 출력할 때, 하나의 버퍼를 동시에 읽고 쓰면 화면이 깜박이거나 깨지는 문제가 발생한다. 이를 해결해주기 위해 back buffer와 front buffer를 parallel하게 이용하는 double buffering 방법을 이용한다. Backbuffer에 그려진 후 S bit가 0이 되는 것을 기다린 후 이를 front

buffer에 옮겨주고, S bit가 1로 바뀐다. 이 후 다시 v-sync를 맞추어 화면에 출력해 준다.

-Slider Switch parallel port

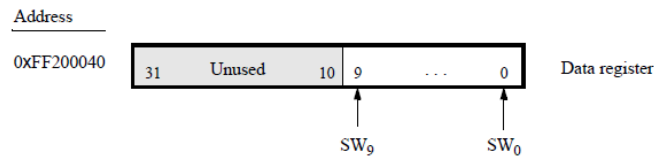


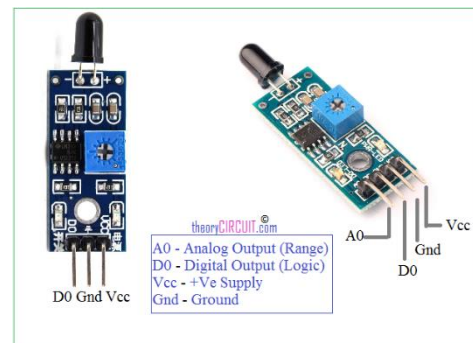
Figure 5. Data register in the slider switch parallel port.

switch는 key button과 다르게 인터럽트 방식이 아닌 polled I/O방식을 이용해 switch의 상태가 on인지 off인지 알 수 있다. 시스템의 세부기능을 구현하기 위해 스위치를 이용했다.

-MQ-2 Gas sensor, HS-FLAME sensor

가스를 감지하기 위해 MQ-2 sensor를 사용했다. MQ-2는 LPG, 부탄, 메탄, 알코올을 검출 할 수 있다. 센서 내부에 있는 히터가 가열이 되면 센서 내부의 금속막에 공기중의 성분이 달라붙게 된다. 이때, 금속막에 성분이 달라붙음에 따라 저항 값이 달라져서 가스를 감지할 수 있다. 위의 사진을 보면 핀이 4개가 있다. vcc에는 전원을 연결해주고, gnd에는 ground를 연결해준다. 그리고 AO 핀에서는 센서로 감지된 값이 아날로그로 출력되고, DO 핀에서는 센서로 감지된 값이 디지털로 출력된다. 가변저항으로 감도를 지정해주고 DO 핀과 DE1SOC보드의 gpio핀을 연결해주어 센서의 출력 값을 받을 수 있다.

스파크와 같은 불꽃을 감지하기 위해 flame sensor를 사용했다. 이 센서는 760nm~1100nm의 파장을 갖는 빛(불꽃)을 감지할 수 있다. 감지 각도는 60도 이다. 센서에 있는 가변저항으로 감도를 조절할 수 있다. 핀은 4개가 있다. 이는 위의 MQ-2 센서와 같다. DO 핀과 DE1SOC보드의 gpio핀을 연결해주어 센서의 출력 값을 받을 것이다.

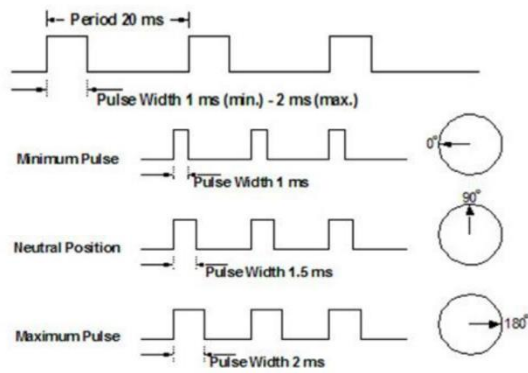


< <https://www.amazon.com/Wavesahre-MQ-2-Gas-Sensor-Detection/dp/B00NJOIB50>>

< <http://www.theorycircuit.com/arduino-flame-sensor-interface/>>

-servo motor

servo 모터는PWM 으로 회전 각도를 -90도에서 90도로 제어 할 수 있다. Servo motor의 각도는 이 펄스의 길이에 따라 제어된다. 약 20ms 마다 펄스를 받게 되고, 이 때 신호가 high인 시간에 따라 각도를 제어할 수 있다. -90도는 pulse의 폭이 1ms, 90도는 2ms이다. 이 PWM 신호를 gpio핀으로 출력하기 위해 출력되는 값 0, 1을 ms 단위로 변경하여 제어할 수 있어야한다.



<<https://www.inverseproblem.co.nz/Guides/index.php?n=ARM.ExTC>>

-PS/2 port

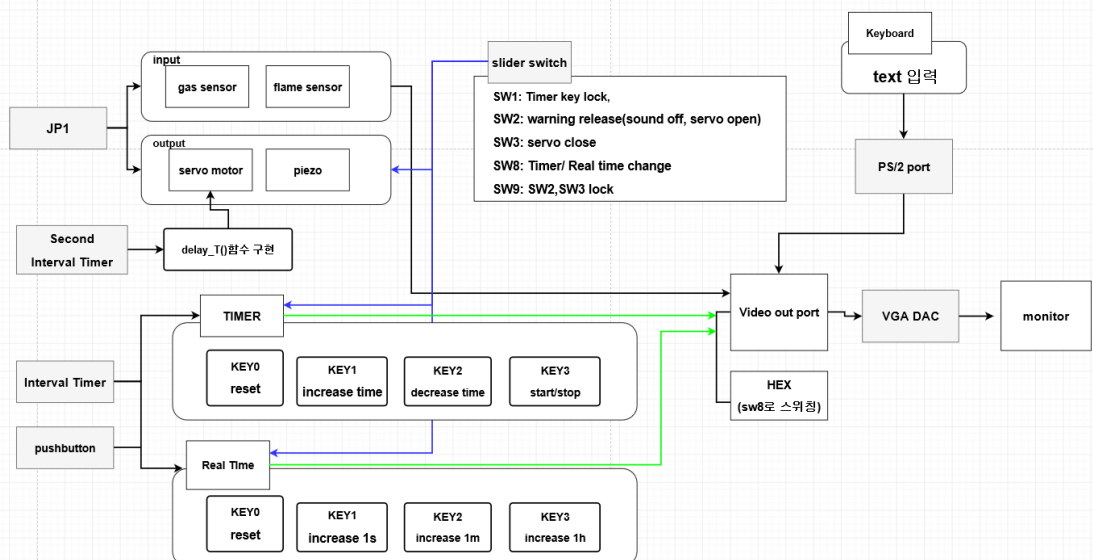
Address	31	...	16	15	...	10	9	8	7	...	1	0		
0xFF200100	RAVAIL				RVALID				Unused				Data	PS2_Data
0xFF200104							CE		RI				RE	PS2_Control

Figure 32. PS/2 port registers.

PS2 port는 DE1SOC보드와 키보드를 연결 할 수 있는 포트이다. 데이터를 저장할 256byte의 FIFO를 포함한다. PS2_data register, PS2_control register로 이루어져 있다. Data register는 읽기 쓰기가 가능하다. 15 번째 비트 RVALID가 1이면 FIFO의 head의 data와, RAVAIL에 있는 FIFO의 entrie를 읽는다. RAVLID가 1이 되면, 이 field를 1씩 감소시키며 데이터를 읽는다. Cotnrol register의 RE field를 1로 설정하고, RAVAIL>0 이면 PS2 port가 인터럽트를 발생시킨다. RI가 1이면 인터럽트가 pending된것이고, FIFO를 비워서 지울 수 있다.

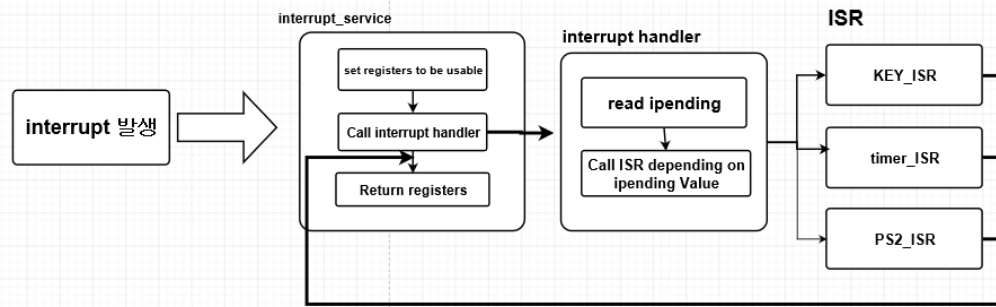
(위 그림들의 출처는 DE1-Soc_computer_Nios2pdf이다.)

다음은 최종적으로 구현한 시스템의 블록도이다. 사용한 peripheral과 추가한 하드웨어, 기능들을 나타냈다.



DE1SOC의 peripheral중 PS2port, interval timer, second interval timer, switch, pushbutton, vga, hex를 사용했다. 그 중 PS2 port, interval timer, pushbutton을 interrupt방식으로 사용했다. Interrupt를 사용하기 위해서는 ienable에 PS2 port, interval timer, pushbutton의 interrupt를 허용해주고 status를 1로 설정했다. Interrupt가 발생하면 program의 routine은 interrupt handler로 이동하게 된다. 여기서 어떤 장치가 interrupt를 발생시켰는지 ipending Value로 찾고 그에 따른 ISR로 이동하여 interrupt를 처리하게 된다. Interval timer의 주기인 1s가 지나면 timer interrupt가 발생하고 timer_ISR로 이동하여 타이머와 현재시간

을 제어한다. Key가 입력되면 interrupt가 발생하고 key_ISR로 이동하여 타이머설정, 현재시간 설정을 한다. 또, PS/2 port에 연결한 키보드의 키가 입력되면 interrupt가 발생하고 PS2_ISR로 이동하여 입력된 문자들을 문자열로 저장하게된다.



1.타이머 구현

Interval timer의 interrupt를 이용하여 1초 주기를 갖기 위해 counter start value를 0x5F5E100 로 설정했다. 초기 설정은 start=1, cont=1, ito=1로 했다. 먼저 타이머의 시간 변수인 total time의 초기값을 10분으로 설정 후, timer interrupt가 발생하면 timer ISR에서 run 변수가 1인 경우 타이머의 시간을 1씩 빼주어 타이머의 시간이 1초씩 줄어 들 수 있도록 했다. 여기서 사용한 run 변수는 타이머의 기능 중 start/stop을 위한 변수로 값이 1이라면 타이머의 상태가 stop이 아니라는 것이고, 값이 0이라면 타이머의 상태가 stop이므로 시간을 감소시키면 안된다. 그후 이 시간을 시, 분, 초 단위로 나누어 각각 timer_HH, timer_MM, timer_SS 변수에 저장했다.

SW8이 off일 때, Hex3to0(), Hex5to4() 함수를 호출하여 시간을 7segment형태로 display했다. Hex 함수 구현은 아래에서 설명한다. 케릭터 버퍼와 픽셀 버퍼를 이용하여 모니터에도 시간을 출력했다.

이 타이머는 SW8이 off일 때, KEY를 누르면 start/stop, reset, 설정시간 증가, 설정시간 감소가 되도록 4가지 기능을 구현했다. KEY에 따라 타이머가 동작하도록 하기위해 interrupt 방식을 사용했다. KEY를 누르면 interrupt가 발생하고 interrupt handler에서 KEY_ISR로 이동하게 된다. Key의 base address가 저장된 변수 KEY_ptr의 값을 통해 edge caputer register의 값을 읽어 4개의 key중 어떤 key가 눌렀는지 알아낸다. 먼저 타이머기능과 현재시간기능을 스위칭해주는 SW8의 비트값이 0인지 확인하여 맞다면, KEY0이 눌렀을 경우 타이머의 시간을 reset해준다. 그러면 초기값인 10분으로 설정된다. KEY1을 눌렀을 경우 total_time에 5분을 더해주어서 타이머의 시간을 5분 증가 시킨다. KEY2를 눌렀을 경우 total_time에 5분을 빼주어 타이머의 시간을 5분 감소시킨다. KEY3을 눌렀을 경우 타이머가 동작 중이라면 stop시키고, 타이머가 stop상태인 경우 다시 동작시키기 위해 run변수를 토글해준다.

SW1이 on인 경우, key를 눌러도 타이머가 변화 없도록 lock을 구현했다. 위의 KEY_ISR에서 SW1의 비트가 1이라면 return 되어 다시 main함수로 바로 돌아가도록 했다.

2. hex에 시간 display

SW8이 on인 경우 현재시간을 hex에 display해준다. Hex3to0(), HEX5to4()함수를 호출한다. 시, 분, 초를 각각 10으로 나누어 십의자리와 일의자리 숫자를 분리한다. 이 숫자들을 7segment 형태에 맞게 값을 변환해준다. 그 다음 Hex3_0, hex5_4에 맞게 시간을 저장한 각 변수들을 8bit씩 쉬프트시키고, or연산하여 합쳐준다. 이를 HEX의 base address에 저장한다.

3. 센서, 서보모터 gpio 연결

JP1 port의 Data register와 direction register를 이용했다. Jp1의 base address를 변수 *gpio에 저장했다. *(gpio)는 Dataregister이고, *(gpio+1)은 direction register이 된다. Gpio는 총 32pin을 연결 할 수 있는데, direction register에 값을 저장해주어 32개의 핀을 각각 input/output으로 설정할 수 있다. Input으로 사용할 경우 0, output으로 사용할 경우 1을 저장해주어야 한다. 편의상 0~15번 pin은 input으로, 16~31번 pin은 output으로 사용했다. 그러기 위해 direction register에 0xFFFF0000값을 저장했다. output으로 설정되어있는 pin에는 dataregister에 값을 저장해서 pin으로 그 값을 출력할 수 있고, input으로 설정되어 있는 pin은 받은 값을 dataregister에서 값을 읽어올 수 있다. 6번 pin아래가 5V vcc, 7번 pin아래가 ground이다. 하드웨어의 연결은 다음과 같다. Vcc와 ground를 브레드보드의 +, -에 연결하여 gas sensor, flame

sensor, speaker, servo motor의 전원 pin과 ground pin을 연결해주었다. input으로 사용할 Gas sensor와 flame sensor의 digital out pin을 각각 JP1의 4번, 5번 pin에 연결했고, output으로 값을 줘야하는 servo motor의 pwm pin과 buzzer의 전원 pin을 각각 JP1의 31,30번 pin과 연결했다.

4. 센서값 읽기

Gas sensor의 값을 읽어오기 위해 gas_sensor()함수를 구현했다. Gas sensor는 4번 핀에 연결했으므로 센서의 input값은 JP1의 dataregister의 5번째 bit에 저장된다. dataregister 값인 *(gpio)을 0b10000과 and 연산하여 변수 sensor1에 저장한다. 센서의 감지여부를 실험하여 printf 함수로 터미널에 그 값을 출력해 봤다. 그 결과 센서에 가스가 감지 되었을 때는 sensor1의 값이 0이었고, 감지되지 않았을 때 값은 16이었다. 센서의 값이 감지되었는지 여부만 필요하므로 sensor1의 값이 0인지 아닌지만 체크했다 그리고 값이 0 이라면 서보모터를 잠그고, 경고음이 출력도록 open=2, Onbuzzer=1로 바꿔준다.

Flame sensor의 값을 읽어오기 위해서 flame_sensor()함수를 구현했다. 5번핀과 digital out pin을 연결했으므로 센서의 input값은 JP1 dataregister의 6번째 bit에 저장된다. 이 값을 sensor2 변수에 저장해준다. gas sensor와 마찬가지로 센서에 감지되었을 때 0이었다. 불꽃이 감지되면 서보모터를 잠그고, 경고음을 출력하기 위해 open=2, Onbuzzer=1로 바꿔준다. 이 변수들은 아래에서 설명한다.

5. servo motor제어

Servo motor의 각도를 PWM pulse로 제어할 수 있다. Servo motor는 20ms 주기로 pulse입력을 기다리는데 이때 pulse가 high인 시간의 길이에 따라 각도가 정해진다. 이 시간을 구하기 위해 DE1SOC보드의 second interval timer를 사용하여 delay_T()함수를 구현했다. 이때 second interval timer의 시간단위는 ms로 설정했다. Delay 할 시간인 use_delay_time int형 변수를 parameter로 주면, 앞서 구현한 타이머와 같이 timer가 timeout되면 use_delay_time에서 1씩 빼준다. 이를 반복하여 use_delay_time이 0이 되면 return하게 되어 이 함수를 호출하면 일정시간(use_delay_time) 동안 delay 할 수있게 된다.

Servo motor를 제어하는 함수는 servo()이다. Servo motor의 각도를 제어하기 위해 변수 open을 사용한다. Open이 1이면 servo motor를 0도 위치로 회전시키고, open=2 인 경우 servo motor를 90도 위치로 회전시킨다. 0도로 회전시키기 위해 pwm pulse의 pulse width는 1ms이어야한다. 그러므로 31번 pin의 출력을 1로주고, delay_T(1)을 호출하여 1ms 만큼 시간이 지나게 한 후 pin의 출력을 0으로 바꿔준다. Open=2일때도 마찬가지로 pulse width를 2ms로 주기위해 pin의 출력을 1로 설정후, delay_T(2)를 호출하여 2ms 만큼 시간이 지난후 pin출력값을 0으로 바꿔준다.

6. 경고음 출력

Gas sensor또는 flame sensor에 위험이 감지되었을 경우 경고음을 출력해야한다. Buzzer의 전원 pin을 JP1의 30번 pin과 연결했으므로 이 pin의 출력값을 제어해주면 된다. Buzzer()함수를 구현했다. 변수 Onbuzzer값이 0이면 buzzer를 끄고, 1이면 buzzer를 켜는다. 원래의 *(gpio)값과 Onbuzzer를 30만큼 right shift한 값을 or연산해서 JP1의 30번 pin의 출력을 제어한다.

7. 모니터 출력

타이머의 시간, 현재시간, 센서의 감지여부 등을 모니터에 출력한다. 타이머의 시간과 현재시간을 character buffer를 이용하여 출력하게 된다면 문자의 크기가 너무 작아서 pixel buffrer를 이용한다.

타이머의 시간이 메인이므로 가장크게 출력한다. 0에서 9까지의 숫자를 각각 pixel buffer를 이용해 pixel 하나씩을 채워 그리기위해 number_display()함수를 구현했다. 이 함수는 0에서 9까지 숫자를 각각 7segment형태의 숫자로 그려준다. Draw_square()함수로 픽셀을 하나씩 채운다. 파라미터로 숫자를 그릴좌표인 X, Y와 크기를 조절하기 위한 변수 k를 받는다. 7segment를 그리기 위해 7개의 사각형을 X, Y의 좌표에 따라 채우도록 구현했다. 또, 타이머의 시간보다 조금 작게 현재시간도 출력해야 하므로 크기 변수를 두어 숫자의 크기를 제어할 수 있도록 했다.

나머지 "timer", "real time", "gas sensor on!", "flame sensor on!"과 같은 문자열들은 character buffer를 이용해 출력한다. 그러기 위해 display_char()함수를 구현했다. parameter로는 문자열의 첫 글자가 출력될 모니터상의 좌표 X,Y 그리고 출력해줄 문자열의 주소값을 받는다. Y좌표값을 7bit 만큼 shift하고 여기에 X

좌표 값을 더해서 offset으로 저장한다.character buffer base+offset에 문자열을 하나씩 저장해준다.

8. 현재 시간 구현

sw8번을 이용하여 1.타이머와 현재타임 모드를 구분할 수 있도록 했다.현재시간을 구현하기 위해서 보드 내의 interval timer가 필요했는데 앞서 1.타이머 구현과 모터제어에 사용되는 delay_T 함수 구현을 위해 2개의 타이머를 모두 사용했다. 그래서 1.타이머 구현에 이용했던 interval timer를 interrupt방식으로 사용했고, key button interrupt를 이용하여 시간을 설정할 수 있도록 구현하였다. 현재시간의 총 시간(단위 1sec)을 의미하는 Global 변수 real_time을 두어 값 timer interrupt가 발생시 timer_ISR에서 1씩 더하도록 하였다. 이 때 sw8을 키면 현재타임 설정 모드이므로 timer_ISR에서 1.타이머의 시간값인 total_time 값이 증가하지 않도록 예외처리를 하였다. KEY_ISR 에서도 마찬가지로 SW8이 켜져있는지 확인하고 켜져있다면 버튼이 1.타이머 제어가 아닌 현재시간 설정하도록 하였다. Key0은 reset, key1번은 1초증가이고 60이 넘었을 경우 0으로 바꿔주었고 key2, 3도 마찬가지로 1분증가 1시간증가를 의미하며 60, 24가 되었을 경우에는 0으로 초기화 해주었다. Key0번이 눌렀다면 reset이므로 rael_time을 0으로 만들었고, key1일경우 real_time을 1증가시켰다. 그러나 초가 60넘어갈수 없으므로 real_time을 60으로 나눈 나머지에 1증가한 값이 60이 됐을 경우 real_time에 60을 빼주어 전체값의 초를 초기화해주었다. 이와같은 원리로 key2, 3을 이용하여 분 과 시간을 증가시키도록 구현하였다.

이렇게 바뀌는 값은 main while(1)을 폴딩io방식으로 모니터와 hex0~5(sw8이 on일시)에 display하게 된다. 그 원리는 1.타이머의 구현에서 설명한 내용과 같다.

9. 키보드 입력으로 메모장 구현

Ps2 port를 이용하여 키보드입력을 받고 그 값을 ps2_interrupt를 이용하여 ch_text 문자열 변수에 값을 쓰도록 하였고 main함수에서 ch_text문자열값을 character buffer를 이용하여 모니터에 display하도록 구현하였다.

구현 방법을 자세히 보면 ps2_base address에 0xff를 store하여 내용값을 초기화해주고 ps2_base address+1 위치의 첫번째 비트에 1값을 주어 enable interrupt설정과 NIOS2_WRITE_IENABLE에 PS2의 level7에 해당 하는 8번째 비트에 값을 주어 초기설정을 하였다. 이후에 whlie(1)을 돌며 polledIO를 하며 ch_text 문자열값을 display하도록 하였다.

이때 키보드 입력이 들어와 키보드 인터럽트가 발생하면 ps2_BASE address 3번째 byte에 해당하는 register 부분에 눌린 키보드에 해당하는 0~256사이에 값이 입력받게 된다. 그 값을 mask하여 byte3 변수에 저장하게 된다. Ch_text의 문자열 커서위치를 buffer_cnt변수에 저장하고 키보드가 눌릴 때 마다 byte3에 해당하는 값을 ch_text buffer_cnt위치에 값을 쓰고 다음 값에 문자열구분을 위해 null값을 주었다. 이때 ch_text에 byte3을 저장하기 전에 byte3값에 해당하는 알파벳을 아스키코드 값에 맞춰 byte3값에 써줬다. 눌린키가 back space 눌린 경우 buffer_cnt값 감소 시키고 그 자리에 공백을 넣어주어 그전 알파벳이 지워지고 다음에 키보드가 눌리면 그 자리부터 글씨가 다시 써지도록 구현하였다.

10. 스위치를 이용한 세부기능 구현

switch의 경우 여러 key, timer, ps2 ISR에 사용 되므로 global 변수 sw에 상태값을 계속 저장하고 다른 ISR에서 sw변수를 extern하여 사용하도록 하였다.

switch1이 켜진경우 key button ISR에 첫부분에서 return하도록하여 Lock기능을 구현하였다.

switch2의 경우 서보모터를 열고 경고음을 끄는 기능으로 main polling 반복문에서 open의 값을 2로 설정하여 모터가 열리도록 설계하였고 OnBuzzer의 값을 0으로 바꿔주어 버저가 울리지 않도록 구현하였다.

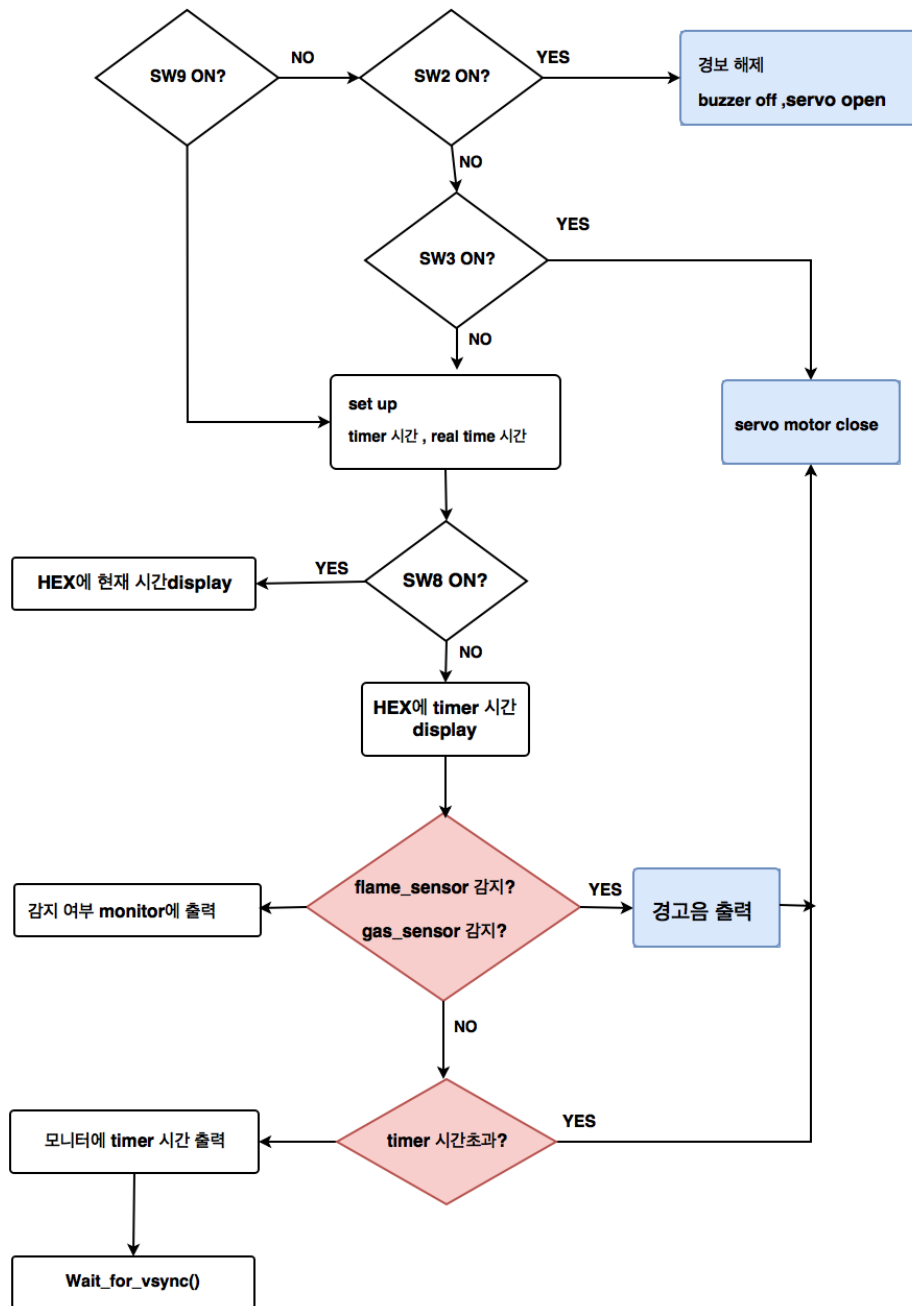
switch3의 경우 서보모터를 잠구는 기능으로 main polling 반복문에서 open의 값을 1로 설정하여 모터가 잠귀도록 설계하였다.

switch8의 경우 현재시간 설정모드로 Hex0~5에 실제시간인 real_HH, real_MS 변수의 값이 display되도록 하고 key ISR에서 real_time 변수가 값이 바뀌도록 컨디션널 해주었다. 마지막으로 timer에서는 시간설정이므로 real_time 변수가 증가하는 것을 막아주었다.

Switch9의 경우 모터제어 switch Lock으로 모터 스위치 해당하는 2,3에 nor연산하여 Lock기능을 하도록

		<p>구현하였다.</p> <p>main 함수 구현</p> <p>시스템이 전체적으로 상호작용하게 하기 위해서 main 함수에서 위에서 구현한 것들을 순서에 맞게 실행시켜야한다. 먼저, KEY, TIMER, PS2, pixel buffer의 초기값을 설정해준다. 그리고 인터럽트를 사용하기 위해 ienable에 interval timer, key, ps/2를 허용하고 Status를 1로 설정해준다. Double buffering을 위해 그려질 버퍼를 설정하고, vsync를 맞춘다. 그리고 JP1의 direction register값을 설정해준다.</p> <p>이 후부터 while(1)을 사용하여 1.센서의 감지, 2.모터, 버저의 제어, 3.모니터, hex에 출력을 계속 반복한다. While(1) 안의 routine 은 다음과 같다. 먼저 SW9가 on인경우 switch lock이기 때문에, SW8이 off 상태인지 확인한다. 그 후 SW2가 on이면 경보를 해제해야한다. 경고음을 끄기 위해 Onbuzzer=0, 서보모터를 열기 위해 open=1로 설정한다. SW3이 on이면 서보모터를 잠그기 위해 open=2로 설정한다.이 후 서보모터와 버저에 대한 변수의 값을 변경했으므로 buzzer(), servo()함수를 호출하여 변수에 맞게 동작하도록 한다.</p> <p>Interval timer에서 변경된 total_time과 Real_time의 값을 시, 분, 초 단위로 나누어서 각 변수에 저장해준다. SW8이 on인지 확인하여 맞다면 현재시간기능이므로 HEX()함수를 호출하여 현재시간을 hex에 display해주고, SW8이 0이면 타이머의 시간을 display해주도록한다.</p> <p>이 후 flame sensor, gas sensor의 감지여부를 파악하기 위해 gas_sensor(), flame_sensor()함수를 호출한다. 이때, 모니터에 센서의 감지여부에 따른 문자열을 display_char()함수를 호출하여 출력한다.</p> <p>그리고 타이머의 시간이 초과되었는지(0이 되었는지) 체크하고, 만약 맞다면 서보모터를 잠그기 위해 open=2로 설정한다.</p> <p>마지막으로 타이머의 시간과 현재시간을 모니터에 출력하도록 display_number()함수를 호출한다. 그 후 vsync를 맞춰준다. 다시 while문의 처음으로 돌아가 이를 무한 반복한다. KEY와 키보드 입력에 관한 동작은 KEY_ISR, PS/2_ISR에서 관리하므로 main함수에는 포함되지않는다. 다음은 main 함수내의 while(1)을 이용한 시스템의 루틴을 나타낸 flowchart이다.</p>
--	--	--

while(1)



설계 과정

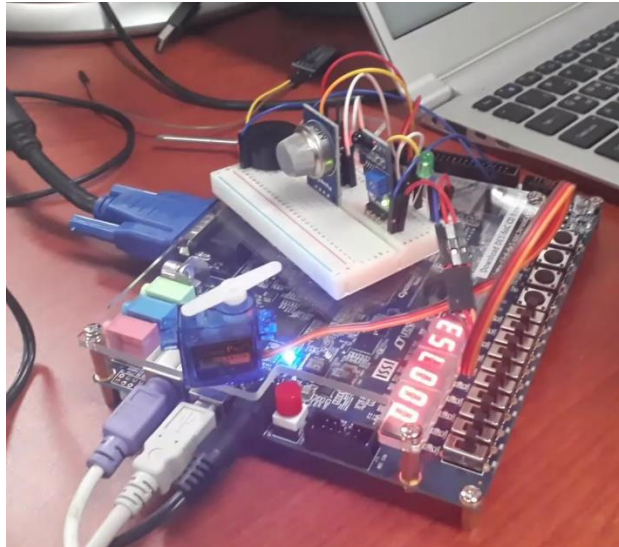
- 추가 외부모듈 연결이 필요없는 타이머부터 구현했다. 타이머는 interval timer를 사용해 시간을 count 하여 이를 시간, 분, 초 단위로 환산하여 hex에 시간, 분, 초 를 display했다.
 - KEY 입력으로 타이머의 세부동작을 control 할 수 있도록 구현했다. KEY0 입력시 timer의 시간이 reset 되도록했다. KEY1 입력시 timer의 시간이 5분 증가 하도록 했다. KEY2 입력시 timer의 시간이 5분 감소되도록했다.
 - SW1이 on인 경우 key lock이 되도록했다. 즉, KEY0~3을 눌러도 타이머의 상태가 변하지 않도록 했다.
- 모니터에 character buffer를 이용해 문자열을 출력하고. Pixel buffer를 이용해 숫자를 출력했다.
 - Timer의 숫자 위에 이것이 timer의 시간임을 표시하기 위해 문자열 "timer"를 출력했다.
 - flame sensor, gas sensor의 감지여부를 표시하기 위해 "flame sensor on!", "gas sensor on!"을 출력했다. 감지가 되지않은 상태일 경우 on 대신 off를 출력했다.
 - 시스템의 전체 KEY와 Switch의 기능 설명을 간단하게 화면에 표시했다.
 - pixel buffer를 이용해 timer의 숫자 시간, 분, 초를 7segment형태로 그려 모니터에 출력했다.
- gpio에 flame sensor, gas sensor, servo motor, buzzer를 연결하여 제어했다.
 - Gpio 4번 pin에 gas sensor를 연결하여 pin을 인풋으로 설정하여 센서의 값을 입력받았다.

Gpio 5번 pin에 flame sensor를 연결하여 pin을 인풋으로 설정하여 센서의 값을 입력받았다.

Gpio 30번 pin에 buzzer를 연결하여 pin을 아웃풋으로 설정하여 경고음을 출력할 수 있도록 했다.

Gpio 31번 pin에 servo motor를 연결하여 제어했다.

-Gas sensor와 flame sensor 둘 중 하나가 감지되었을 때, 타이머의 시간이 다 되었을 때 서보모터를 90도 위치로 회전하도록 했다. 원래 90도의 위치인 경우 아무 동작도 안한다. (벨브 잠김을 뜻함) 버저가 울리도록 했다.

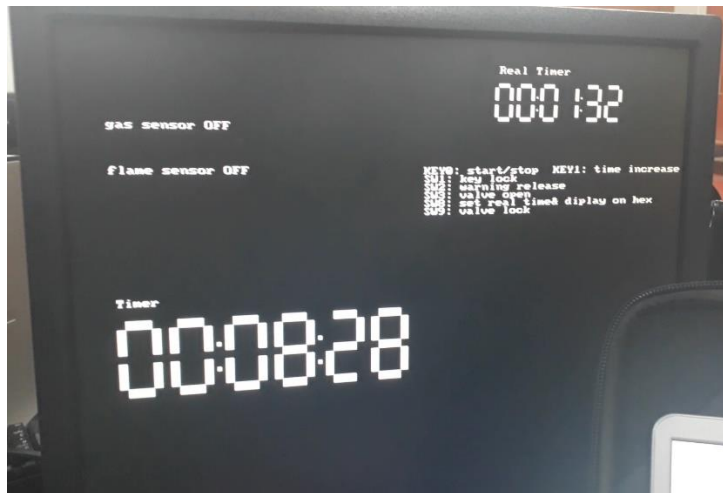


4. 현재 시간을 구현하여 모니터에 출력했다.

-Timer를 구현하기 위해 사용한 interval timer를 이용해 현재시간을 구현했다. 그리고 이 시간을 hex에 display했고, 모니터에도 출력했다.

-SW8이 on일 때 KEY0~3으로 현재시간을 설정할 수 있다.

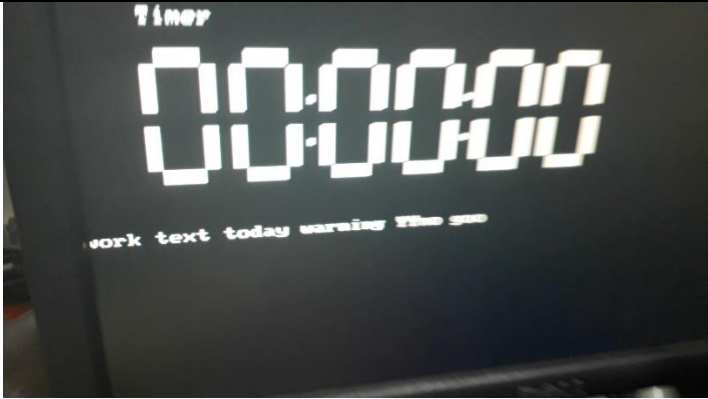
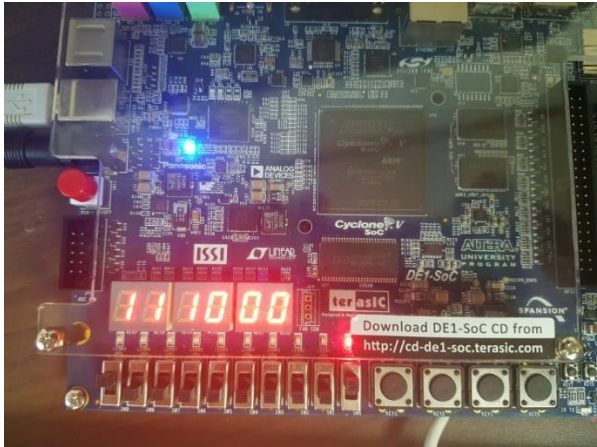
-KEY0: 시간 reset, KEY1: 1초 증가, KEY2: 1분 증가, KEY3: 1시간 증가



5. PS/2 port를 이용해 interrupt방식으로 키보드로 문자를 입력 받아 모니터에 출력했다.

-keyboard를 PS/2 port에 연결하여 key 입력시 interrupt가 발생하도록 했다.

-입력받은 문자 a~z로 문자열을 구성해 character buffer를 이용해 모니터에 출력했다. Space bar, back space 기능도 구현했다.

			
시험 / 평가	설계 결과 검증, 분석, 평가	<p>6. Switch를 이용하여 시스템을 세부적으로 사용자가 직접 control 할 수 있도록 구현했다.</p> <p>SW1: 타이머 key lock</p> <p>SW2: 경고음 끄기, servo motor 열기</p> <p>SW3: servo motor 잠그기</p> <p>SW8: on인경우 현재시간 설정, off인 경우 타이머 시간 설정</p> <p>SW9: servo motor lock</p>	
		<p>1주차</p> <p>1.계획</p> <ul style="list-style-type: none"> - 타이머 구현, hex에 display (HH:MM:SS Hex0~5표시) - Key button를 이용하여 타이머 제어(key0: 리셋, key1: 300초 증가, key2: 300초 감소, key3: 타이머 start/stop) - switch1를 이용해 key lock 기능 - sensor값 변수를 설정하여 LEDR 표시 <p>2. 결과</p> <ul style="list-style-type: none"> - 타이머 기능 구현 - Key button을 이용하여 timer 제어 구현 - key lock 기능 구현 - sensor변수 LEDR 표시 구현. <p>3. 검증</p> <p>사진과 같이 hex에 타이머의 시간(hh:mm:ss)을 디스플레이 했고 키 조작을 통하여 타이머의 동작을 제어하는데 key0을 누르면 reset, key1을 누르면 시간 증가, key2를 누르면 시간 감소, key3을 누르면 타이머의 작동 on/off toggle를 확인할 수 있었다. switch 1번을 on 했을 때 key버튼을 입력이 되도 타이머의 동작이 변화가 없음을 확인 가능했고 센서의 변수에 이상 검출 값을 주어 LEDR0이 작동하는지 확인하였다.</p>  <p>4. 분석 및 평가</p> <p>지난 실습과제에 배운 기술을 확장하여 쉽게 구현할 수 있었다. 가스안전 시스템에서 중요한 가스 사용</p>	

시기를 정하는 타이머를 구현하였고 KEY, TIMER 인터럽트를 이용하여 시간을 제어하고 스위치로 기능을 여러가지 기능을 만들 수 있도록 하였다.

2주차

1.계획

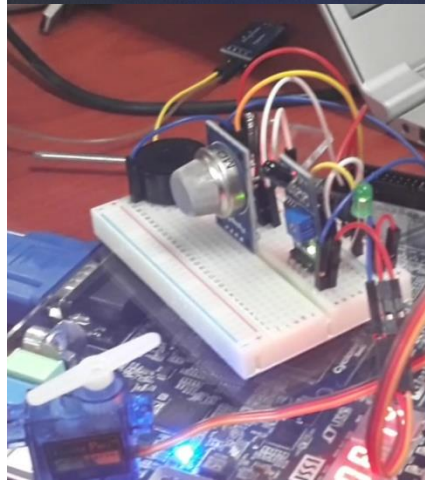
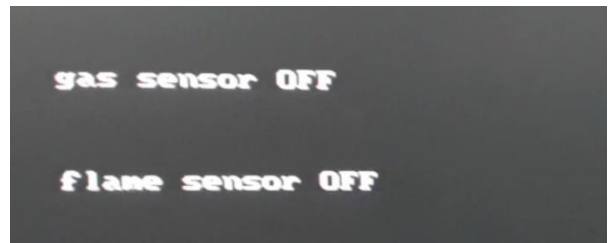
- gpio사용법과 센서 이용법을 공부하고, 이를 토대로 가스감지센서, 불꽃감지센서를 d1-soc보드에 연결한다.
- 케릭터 버퍼를 이용하여 모니터에 시간과, 불꽃 감지센서,가스 감지센서의 각 감지여부를 출력한다.

2.결과

- sensor들을 DE1-SOC보드의 gpio에 연결
- 모니터에 타이머의 시간을 출력하기위해 draw_square을 이용하여 숫자를 7segment방식으로 직접그리는 함수 구현
- timer, 불꽃감지센서, 가스감지센서의 각 감지여부를 모니터와 LEDR 출력 구현
- 오디오를 이용한 경고음을 버저로 변경하여 경고음 구현
- 서보모터를 원하는 위치로 제어하도록 PWM pulse를 생성할 수 있도록 코드 구현

3.검증

아래 사진은 타이머의 시간, 센서감지여부를 모니터로 출력한 결과이다. 그 아래 사진은 가스센서,불꽃센서, 피에조 스피커, 서보모터를 브레드보드에 구성하여 gpio에 연결한 것이다. 센서가 제대로 작동하는지 시험하기 위해 가스센서 근처에 라이터를 가까이 갖다대서 가스를 감지하게하고, 라이터를 켜서 불꽃센서가 불꽃을 감지하는지 알아보았다. 센서가 이를 감지하고 버저가 울림을 볼 수 있었다. 또 이러한 감지 여부가 LEDR과 모니터에 출력됨을 확인하였다.센서가 감지되었을 때는 sensor on!으로 출력되는 문자가 변한다. 서보 모터에 1ms펄스, 2ms펄스를 지속적으로 주었을 때 모터의 위치가 달라짐을 확인하였다.



4.분석 및 평가

센서들을 브레드보드판에 연결하였고 각 센서의 민감도를 가변저항을 조절하여 수정하며 적정수준의 민감도를 찾는 과정에서 가스센서의 문제가 없는데 불꽃 플레임센서의 경우 라이터의 불꽃보다는 스파크에

일시적으로 감지하는 모습을 보였다. 추가로 오디오를 통하여 경고음을 센서감지시 경고음을 낼려 했으나 경고음을 수시로 녹음해줘야하는 문제를 해결하지 못하여 버저로 구현하는 것이 더 좋은 시스템이라 판단하여 변경하였다. 가스 안전 시스템이 안전조치를 하기위해 센서를 이상없이 감지하고 유저가 시스템을 이용하기 위해 모니터에 출력하는 수준까지 왔다.

3주차

1. 계획

- 서보모터를 보드에 연결한다.(2주차 구현)
- 센서감지시 경고음이 나도록 한다.
- 타이머 시간종료시, 센서 감지시, 스위치 제어를 통해 서보모터를 이용한 동작 실행.

2. 결과 및 분석

- 센서감지시 경고음이 출력되도록 구현하였고 switch 2번을 껐을 때 경고음이 꺼지도록 하였다.
- stopwatch 시간종료, 센서 감지시, switch3번을 키면 서보모터 잠기도록 하였고 , 스위치 2번을 껐을 때 서보 모터를 열도록 구현하였다.

- switch 9번을 통하여 switch 3,4의한 서보모터 제어의 Lock을 만들었다.

(추가구현)

- 현재타임을 구현하여 모니터에 출력하였다. switch 8번을 껐을때 key button으로 현재시간을 설정할 수 있도록 하였고 hex에 현재시간이 display되도록 하였다.
- 키보드를 이용하여 모니터에 메시지를 남길 수 있도록 구현하였다.
- switch의 메뉴얼을 모니터에 출력하도록 하였다.

3.검증

모니터에 설정한 가스감지여부, stopwatch시간, realtime, switch메뉴얼, 메시지를 출력이 됨을 볼 수 있었다. 센서감지시 이상없이 경고음 이상표시와 경고음이 울리고 서보모터가 잠금을 확인하였고 스위치 2번을 껐을 때 서보모터가 열리고 3번을 껐을 때 서보모터가 잠금을 확인하였다. 스위치 9번을 껐을 때 스위치 3,4번의 제어가 먹지 않을 확인하였다. 스위치 8번을 껐을 때 hex에 현재 타임이 display됐고 key를 눌렀을 때 reset, 1초증가, 1분증가, 1시간증가가 됨을 확인하였다. 스위치 8번을 껐을 때 hex에 stopwatch시간이 display됐고 모니터에 리얼타임이 시간에 따라 증가함을 확인하였다. 키보드를 통하여 입력을 하였을때 message칸에 글씨가 입력되고 backspace를 눌렀을 때 원래대로 돌아감을 확인하였다.



4.분석 및 평가

3주차에 거쳐서 가스 안전시스템이 타이머,현재시간 표시, 스위치 제어, 센서감지와 그에 따른 서보모터 제어, 키 버튼의 타이머 제어, 키보드메시지 작성, 또 앞의 내용을 모니터,LED에 디스플레이등 전부 이상없이 작동하였고 시스템을 완성하였다. 또한 추가로 리얼타임을 이용하여 센서감지시, 메시지 작성시 리얼타임을 기록하려 하였으나 시간이 없어 추가 구현하지 못한 점이 아쉽다.

추진 일정
대비 진행
결과

전체 일정을 표로 나타내고 진행 계획 대비 추진 결과를 보임. 1페이지 이상 작성.

주차	진행 계획	추진 결과
1	<p>타이머 기능 구현</p> <ul style="list-style-type: none"> -interval timer를 이용하여 interrupt를 발생시켜 시간을 countr하고, 이를 바탕으로 타이머 구현. - Key 동작 구현(리셋, 시간증가, 시간감소, start/stop) -hex에 타이머 시간 diplay -switch를 이용해 timer_lock 기능 구현 (Switch가 on인 상태에서 key button을 눌러도 타이머의 동작에 변화가 없게 한다.) 	<ul style="list-style-type: none"> -key에 따른 세부사항 동작을 구현했다. ->KEY0을 눌렀을 때 timer의 시간이 reset되도록 했다. ->KEY1을 누를 때 마다 timer의 설정시간이 5분씩 증가하도록 했다. ->KEY2을 누를 때 마다 timer의 설정시간이 5분씩 감소하도록 했다. ->KEY3을 누르면 타이머의 동작이 start/stop 되도록 했다.(동작되고 있는 상태에서 누르면 멈추고, 멈춰진 상태에서 누르면 다시 동작을 하도록 했다.) -타이머의 시간을 hex0~5에 7segment로 display 했다. -SW1을 on했을 때 key0~3를 눌러도 timer의 동작에 변화가 없도록 key lock기능을 구현 했다.
2	<ul style="list-style-type: none"> - gpio사용법을 공부하고, 이를 이용하여 sensor, servo motor 연결 - 캐릭터 버퍼를 이용하여 모니터에 시간과, 센서의 감지여부를 출력한다. (어떤 센서에 감지 됐는지) 	<ul style="list-style-type: none"> -모니터에 타이머상태를 출력 ->character buffer를 이용해 "timer"와 key, switch기능 설명을 위한 문자열을 출력했다. ->pixel buffer를 이용해 타이머의 시간(hh:mm:ss)에 해당하는 숫자를 출력했다. ->모니터에 센서의 감지여부를 문자열로 출력했다.(ex, flame sensor on!) ->화면 깨짐, 깜박거림이 없도록 double buffering, vsync를 이용했다. -Gpio를 이용해 부가적 모듈 연결 ->gpio의 0~15번 pin을 input, 16~31 번 pin을 output으로 지정했다. ->gpio 4번 pin에 gas sensor, 5번 pin에 flame sensor, 31번 pin에 servo motor를 연결했다. ->sensor로 입력받은 값을 printf 함수를 이용하여 terminal에 출력해보고, 센서가 정상적으로 감지할 수 있는지 확인했다. ->servo motor를 제어를 위한 pwm pulse를 만들기 위해 second interval timer를 이용해 delay_T()함수를 구현했다.
3	<ul style="list-style-type: none"> - 센서의 값에 따른 servo motor를 이용한 동작 실행. -타이머 값에 따른 서보모터를 이용한 동작 실행. - 경고음 출력 - 현재 시간 구현 	<ul style="list-style-type: none"> -서보모터의 동작 확인 ->pwm pulse에 따라 제어가 되는지 확인했다. ->타이머의 시간이 0이 될 때, 센서가 감지했을 때 서보모터를 90도 위치로 회전하도록 했다.(벨브 잠금을 뜻한다.) ->SW3이 ON일 때 서보모터가 90도로 움직이도록 했다. (벨브 잠금을 뜻함) -경고음 출력 ->buzzer를 gpio에 연결했다.

			->센서가 감지됐을 때, buzzer를 이용해 경고음이 울리도록 했다. -SW2이 on이면 경보가 해제 되도록 했다. ->servo motor를 0도로 움직이도록 했다.(벨브 열림을 뜻함) ->경고음을 껐다. - 현재시간을 모니터에 표시했다. ->SW8이 off면 원래와 같이 타이머의 시간이 hex에 표시되고 key button으로는 타이머의 시간을 설정할 수 있도록 했다. SW8이 ON이면 현재시간이 hex에 표시되도록 했고, KEY0을 누르면 현재시간이 reset되고, KEY1~3으로는 현재시간을 설정할 수 있도록 했다. KEY1을 누르면 1초 증가, KEY2를 누르면 1분증가, KEY3을 누르면 1시간이 증가 되도록 했다. -계획외로 추가로 구현한 기능 - 메모기능을 구현했다. PS/2 port를 이용해 키보드로 문자를 입력 받아 모니터 하단에 출력했다.
역할 분담 및 개별 공헌 내용	<2015124129 오혜빈> DE1SOC보드에 연결하여 사용할 수 있는 센서와 액추에이터의 datasheet 조사. JP1에 센서, 버저, 서보모터를 연결하고 하드웨어 연결을 구성했다. -JP1 register를 제어하여 핀을 input/output으로 할당 sensor의값을 입력받고, 버저와 서보모터에 출력값을 주었다. -sensor의 입력값을 분석하고 input data를 사용할 수 있도록 code 구현 -서보모터를 제어하기 위해 second interval timer를 사용해서 display_T()함수를 구현. -display_T()함수와 JP1 register를 설정하여 pwm pulse를 출력할 수 있도록 구현 -센서 감지시에 모니터에 감지여부를 출력하도록 구현		
	<2012122 이민호> Timer 구현 -interval timer를 이용해 interrupt방식으로 시간을 count하여 이를 기준으로 timer를 구현했다. -key button으로 interrupt를 발생시켜 timer를 설정할 수 있도록 구현했다. Character buffer를 이용해 문자열을 출력할 수 있도록 했다. -display_char()함수 구현. 기존에 배운 픽셀 버퍼를 이용하여 숫자를 모니터에 출력할 함수 구현했다. -draw_line, draw_square, display_num 함수 구현 SW0~9의 상태에 따라 시스템을 세부적으로 control할 수 있도록 KEY_ISR, main함수에 구현		
	<2012122246 임찬영> 센서의 감지여부, 타이머의 시간에 따라 서보모터, 버저가 동작하도록 알고리즘을 구현했다. -Onbuzzer, open등의 변수 제어 현재 시간과 key로 그 시간을 설정할 수 있도록 구현했다. -timer에 사용되는 interval timer 를 이용해 현재시간을 구현했다. -SW8이 on인경우 현재시간을 KEY로설정 가능하도록 했고, hex에 display되도록 구현 -현재시간을 display_char, display_num 함수를 활용해 모니터에 출력 메모기능 구현 -PS/2 사용법을 조사		

		-PS/2 port를 이용해 keyboard로 문자를 입력받고 그 문자를 모니터에 출력하도록 구현했다. -PS/2 port를 interrupt 방식으로 사용하도록 구현 -PS2_ISR 구현
현실적 제한 요건 반영 결과	경제요건	이 가스 안전 시스템은 시장에 내놓아도 경제적 가치가 크다. 범용성이 크다. 실제로 사용한 센서와 모터의 가격은 저렴하지만 소대규모 시설의 가스실부터 가정집까지 가스가 사용되는 모든 곳에 필요와 단가에 맞게 하드웨어 성능, 필요한 기능을 수정하여 사용할 수 있다.
	사회적 영향	사회적으로 우수한 효과를 볼 수 있다. 앞으로 고령화 시대에 갈수록 노인 가구가 증가한다. 이때 나이가 많을수록 집중력과 순발력이 떨어지기 때문에 가스를 이용함에 있어 위험에 노출되기가 쉽다. 이 시스템을 이용한다면 자동으로 가스를 제어해주고 알림 알려줌으로써 많은 화재사고를 미연에 방지할 수 있다.
	미적 요소	하드웨어를 연결할때 브레드보드 하나에 연결하고, 모니터에 display시에도 단조롭게 구성하여 지금 당장은 미적 요소가 없지만 시장에 출시된다면 유저의 취향과 나이에 따라 모니터UI와 외적 디자인을 바꿀 필요가 있다.
	보건 및 안전	이 시스템은 안전을 위한 시스템이므로 보건 및 안정성에서는 우수하다. 그러나 만일에 하드웨어의 이상으로 위험요소가 제대로 감지가 안되는 것을 고려하여 주기적으로 점검하고 주요 시설에는 시스템을 분할하여 한 곳이 망가져도 다른 시스템이 작동하도록 확장이 가능하다.
	내구성	이 시스템은 여러 센서를 이용하는데 센서의 경우 내구성이 약하다. 그러므로 실제 제품을 만들 때 센서를 비전문가도 쉽게 교체할 수 있도록 설계하고 수시로 점검할 수 있도록 주기적으로 알림을 해주면 좋겠다.
	산업 표준	산업 표준에 범용성이 높을 수 있는 것이 가스 밸브를 제어하도록 이 시스템을 연결하면 되므로 어떤 소대규모 시설의 가스실이나 가정집에도 확장하여 사용이 가능하다.
토의	-어떤 센서를 사용할지에 대해 논의했다. 먼저 가스 밸브실에서의 위험상황이 뭐가 있을지 생각했다. 그 결과 스파크(불꽃), 가스를 감지하기로 했다. 그 중 DE1SOC에 연결하여 사용할 수 있는 센서를 찾았다. -모니터에 어떤 형태로 출력할지 논의했다. 타이머의 시간을 가장 크게 표시하기로 했다. 센서감지여부를 왼쪽 상단에 배치하고 타이머는 왼쪽 중간, 작업 메모내용은 화면 하단, 현재시간은 오른쪽상단, key와 SW에 따른 기능설명은 현재시간 밑에 나타내기로 했다. -센서에 감지됐을 경우, 경고음을 출력하기 위해 audio in/out port를 사용하려고 했었다. 하지만 녹음을 직접해주어 buffer를 채워야했다. 매번 녹음을 해주는 것은 비효율적이므로 buzzer를 이용해 JP1에 연결하여 경고음을 출력해주기로했다.	
	<임찬영>: 이번 프로젝트를 통하여 많은 것을 배운 것이 많고 또한 정말 즐거웠다. 첫째로, 프로젝트를 기획, 설계과정, 구현, 테스트 과정을 토의 하고 하나하나 자세히 문서화하는 과정에서 문서화의 중요성을 깨달았다. 기획 단계에서 아이디어 회의를 한 것을 계획서에 문서화하니 만들어야될 산출물에 대해서 명확해졌다. 설계과정에서 우리가 계획했던 진행사항을 파악하고 완성도를 체크하고 평가할 수 있었고 또 구현된 내용, 제한사항, 해결책등 논의한 내용을 정리하면서 팀원 간의 의견을 하나로 종합하여 방향성을 재설정할 수 있었다. 또 그를 토대로 구현하고 테스트해야 할 부분이 명확해져 시간을 단축할 수 있었다. 물론 이 과정들이 우리팀 팀원들 하나하나가 모두 협조적이고 참여하였기에 가능했다. 역할을 나눴지만 자기가 맡지 않은 부분까지 서로 도와가며 진행하였고 또 회의마다 서로 많은 아이디어를 냈고 그것을 서로 잘 종합할 수 있었다. 둘째로, 시스템을 계획할 때 많은 요소들 고려하여 만들어야 좋은 시스템임을 느꼈다. 많은 요소들 중에 몇가지 얘기하자면 확장성과 범용성에 대해서 말하고 싶다. 이 가스 안전시스템은 좋은 범용성과 확장성을 가지고 있다. 이 시스템은 가스밸브 연결을 제외한 다른 부분은 독립적이기 때문에 범용적으로 많은 곳에 가스밸브 제어 부분만 연결하면 사용이 가능하다. 그리고 이 시스템은 센서로 화재위험을 감지하고 위험요소를 차단하도록 설계되어있다. 그러므로 후에 확장된다면 센서 감지했을 때 많은 대처로 연결이 가능하다. 예를 들어, 화재 방화벽을 자동으로 내리고 위험 수준일시 시설에 있는 인력 피해가 없도록 사람을 대피시키는 안내하고 경고할 수 있다. 또 후에 기술이 발달하면 화재감지시 로봇을 컨트롤하여 초기	

에 화재를 신속하게 진압할 수 있다.

<이민호>: 이번 설계하면서 하면서 느낀 점은 단기간에 프로젝트를 계획하고 작업한 다는 것이 많이 어렵다는 것을 우선적으로 깨달았다. 계획할 시간이 1주일 밖에 없었기 때문에 무엇을 만들어야 할지 생각보다 고민이 많았다. 처음에는 지진 운동을 display하는 것을 만들 생각이었으나 조원들과 토의 후에 가스, 불꽃 감지 센서에 흥미가 생겨서 이것을 만들기로 하였다. 맨처음에는 키보드 구현을 필요 없었다고 생각해서 계획서에는 넣지 않았다. 프로젝트를 진행하면서 우리가 프로젝트 계획보다 빠르게 진행되는 것을 느끼고 추가 구현하면 좋은 것 중에 키보드를 추가 구현했다. 키보드 입력 값이 보통의 키보드와 다르게 이상한 문자가 들어오고 뿐만 아니라 키를 입력하는데 3개의 버퍼를 이용할 줄 모르고 막막했었다. 노력 한결과 좋은 결과를 냈다. 이번 프로젝트를 이용하면서 어려운 점은 키보드 구현과 모터 동작 구현방법이었다. 모터 동작 방법을 알고있는 조원이 있어서 생각보다 쉽게 풀렸지만 PWM을 이용한 방법으로 모터가 동작하는지 전혀 모르고 있던 나에게는 조금의 충격이 다가왔다. 나는 알고리즘 코딩하는 것을 많이 했었고 기계코딩은 드론 코딩 밖에 한적이 없었다. 이렇게 하드웨어 동작까지 관리하는 프로그램은 처음 만들어보았다. 하드웨어 프로그래밍은 생각보다 내게는 재미있었고 조금만 더 시간이 있으면 좀더 맛있는 display같은 것을 만들 수 있었는데 그 부분에 있어서는 아쉬웠다.

<오혜빈>: 수업 때 학습했던 timer와 hex에 display, video out을 이용한 모니터 출력 등을 기본으로 추가적으로 peripheral들을 연결하고 외부 모듈(센서, 모터 등)을 gpio로 연결하는 방식으로 설계를 진행했다. 계획단계에서는 우리가 원하는 기능을 추가하려면 어떤 모듈을 사용해야 하는지, DE1SOC보드와 연결할 때, 이 모듈이 보드와 호환이 될 것인지에 대해 고려했었다. 고민하며 여러 센서나 액추에이터의 성능을 찾아보면서 실제로 쓰이고 있는 제품들에 대해 알 수 있는 기회가 되었다.

실제로 설계 진행시에는 계획단계에서 예상했던 것 의외로 많은 문제가 생겼었다. 캐릭터버퍼를 이용한 것이 없었기 때문에 출력되는 문자의 크기를 예상하지 못했었다. 문자의 크기가 너무 작아 픽셀버퍼로 이전에 사용한 draw_square()함수를 이용해 숫자를 직접 그리기로 팀원들과 논의하면서 문제를 해결할 수 있었다. 이처럼 문제 발생시에 다른 대안을 찾고 기존에 배웠던 것을 응용하여 문제를 해결할 수 있었다.

설계과정에서 가장 어려웠던 점은 수업시간에 배우지 않은 memory mapped I/O device를 사용하는 것이었다. 우리는 PS/2 port를 사용했는데, 그동안 timer, pushbutton 등의 peripheral을 사용하는 방법을 배웠기 때문에 Nios2의 manual을 읽어보고 PS2의 register datasheet을 참조해서 interrupt 방식으로 키보드 입력을 받을 수 있었다.

또, 학교에서 배운것들을 종합적으로 활용해볼 수 있었던 프로젝트였다. 서보모터를 사용하려면 PWM pulse를 입력으로 줘야했는데, 이는 1학년 2학기 기초공학설계에서 배운 내용이었다. 또한, 컴퓨터구조에서 배운 것을 활용하여, 어떤식으로 코드를 작성해야 더 효율적이고 cpu를 낭비하지 않을지 하드웨어적인 관점에서 고려하여서 코드를 구성 할 수 있었다.