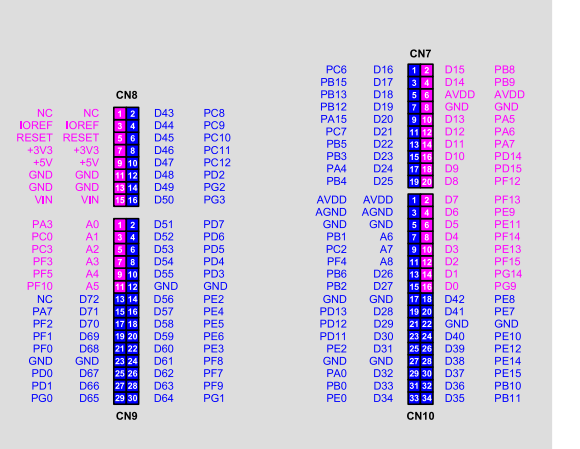
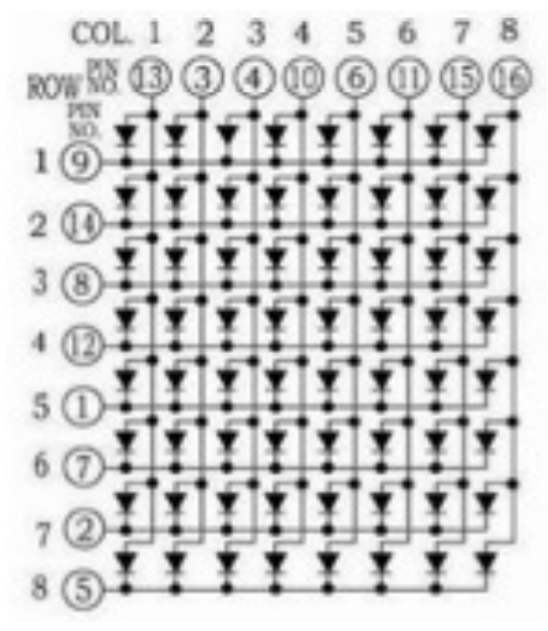
**임베디드 시스템 설계**

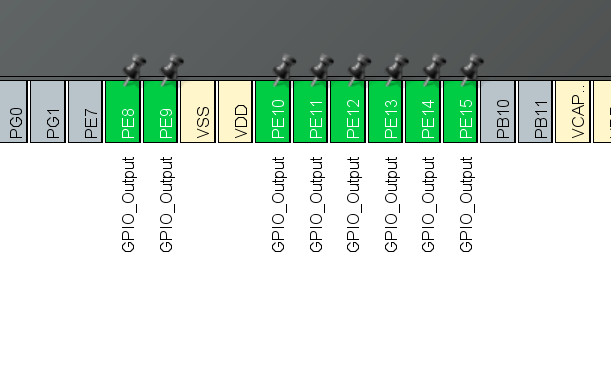
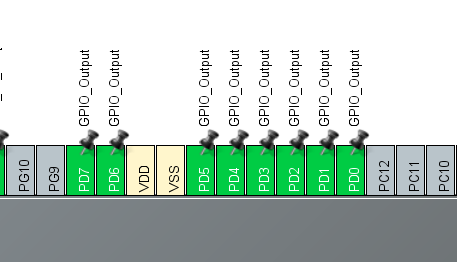
**학기말 설계과제**

20211606 한석기

**1. Dot Matrix Display를 Board에 연결하기 위한 초기 설정**



Dot Matrix Display의 제품번호가 표시된 왼쪽 끝부터 반시계방향으로 1 ~ 16까지의 PIN NO가 부여되어 있다. 하드웨어적으로 암암 점퍼케이블을 이용해 PD0 ~ PD7을 각각 PIN NO 9, 14, 8, 12, 1 7, 2, 5에 연결했고, PE8 ~ PE15를 각각 PIN NO 13, 3, 5, 10, 6, 11, 15, 16에 연결했다. 위 Dot Matrix Display 제품은 Column이 HIGH이고 Row가 LOW일 때, 해당 위치에 불이 들어온다.



소프트웨어적으로는 STM32CubeIDE에서 PD0 ~ PD7, PE8 ~ PE15에 대한 PINOUT 설정을 해주고 코드를 generate했다.

**2. ISR, Task**

(1) User Button ISR

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

이전에 User Button이 눌린 시간과 현재 User Button이 눌린 시간 사이의 차를 계산해 Debouncing을 처리했다. 만약 시간차가 80ms이상이라면 User Button 동작을 시행하고, 아니라면 시행하지 않는다.

mode 변수는 global 변수로 0이면 초기 상태, 5이면 5번 모드 중 발생한 Interrupt, 6이면 5번 모드에서 수정 중에 발생한 Interrupt, 8이면 실행할 모드의 번호를 고르는 단계를 의미한다. 따라서 각 모드의 실행동작에 따라 switch case문으로 처리했다.

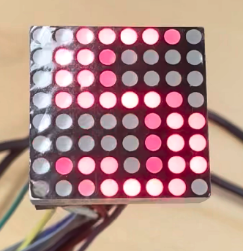
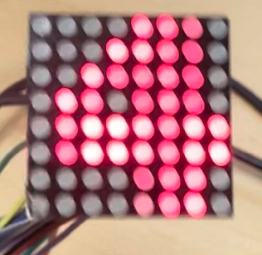
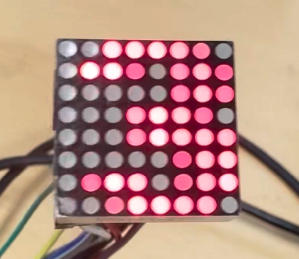
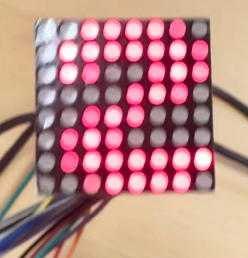
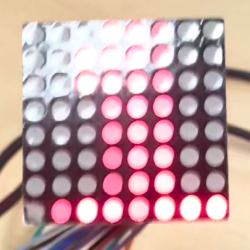
(2) SwitchTask

SwitchTask는 User Button Interrupt에 대한 동작을 모드 선택과 모드 5를 구분해 Task가 Unblock해야하는 상황이면 Task를 Unblock해주는 Task다. ISR에서 Semaphore를 통해 SwitchTask는 Unblock된다. 모드 선택에는 xQueue를 사용해, 모드 선택 Task를 Unblock해주고 모드 5번을 사용해 수정중일 때는, xFiveQueue에 보내야 할 상황일지, 아닐지를 구분하여 Unblock해준다.

(3) WriteNumTask

WriteNumTask는 번호를 띄워주고 선택하는 Task다. 해당 Task에서 xOneShotTimer를 켜주고 flag를 세운다. 번호가 켜진 채로 1초 이상이 지나면, 해당 번호를 선택해준다. 번호가 선택될 때, xSecQueue로 번호를 전송하여, ModeTask를 Unblock한다. 번호가 켜진 채로 1초 이상 지나지 않았는데, User Button Interrupt가 발생하면, 아까 세운 flag에 따라 Timer를 Reset한다.

Global 변수로 0~9까지에 대한 각 숫자의 8x8 Display를 배열로 저장했다. 따라서, switch case문으로 1에서 5까지 각 숫자를 선택해 PIN을 제어하면 아래 사진들과 같이 숫자를 Display에 띄울 수 있었다.



(4) ModeTask

WriteNumTask에서 선택된 번호를 처리하는 Task다. 1~4번 모드는 ModeactTask라는 하나의 Task로 관리된다. xOneQueue에 데이터를 전송함으로써, ModeactTask를 Unblock시켜준다. 또한 left scroll을 할 수 있게끔 하는 xAutoReloadTimer를 켜준다. 2번, 4번 모드는 각각 1번, 3번 모드의 2배 속도로 left scroll하게 Period를 변경해줬다. 이후에 있을 1번, 3번 모드 선택도 고려해 Period를 원래대로 되돌리는 작업 또한 넣어줬다.

5번 모드는 ModefiveTask를 통해 관리된다. 따라서 xFiveQueue에 데이터를 전송함으로써, ModefiveTask를 Unblock시켜준다. 이 때, 전송되는 데이터의 초기값은 0이다.

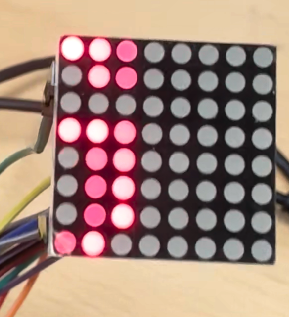
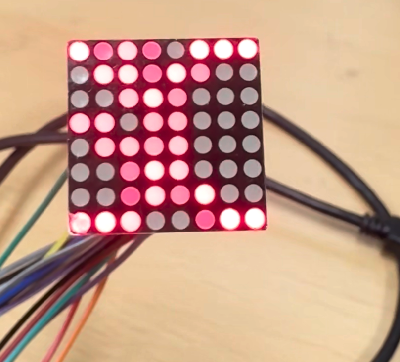
(5) ModeactTask

ModeactTask는 xOneQueue로 받은 데이터에 따라 포인터가 SOGANG배열과 STUDID배열을 선택해준다. Timer와 Sliding Window방식을 사용해, 특정 열 구간을 Display에 출력시켜주고 3번 left scroll할 때까지, xAutoReloadTimer를 돌린다. 3번 left scroll했다면, xAutoReloadTimer를 꺼주고 mode를 0으로 설정해, 초기상태로 만들어준다.

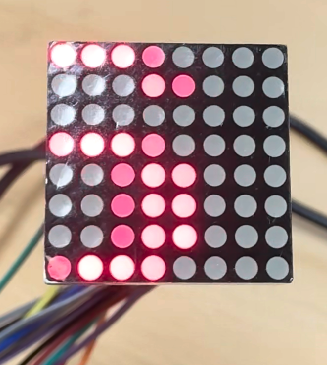
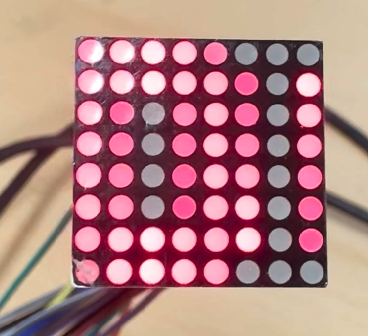
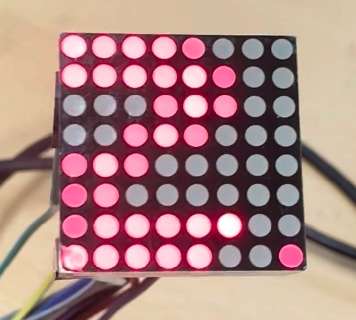
xAutoReloadTimer는 크기가 8인 Window의 시작위치를 변경시켜준다. Window는 moving\_point라는 Global 변수다. moving\_point가 배열을 넘어갔다면, 0으로 초기화시켜준다. moving\_point가 48, 즉, Blank 이전 문자 혹은 숫자까지 left scroll됐다면, shift\_count를 증가시킨다. shift\_count는 문자열 혹은 숫자열이 left scroll된 횟수다.

SOGANG이 left scroll되는 과정은 아래 사진들과 같다.

실내이(가) 표시된 사진

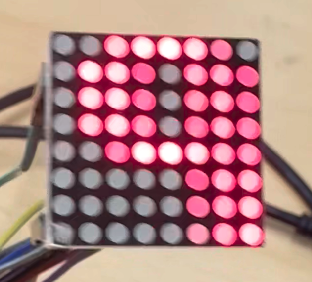
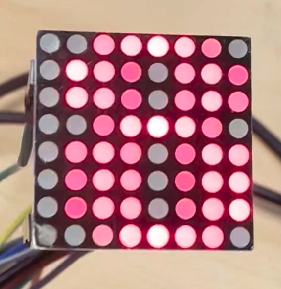
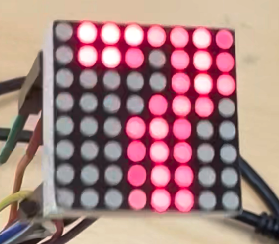
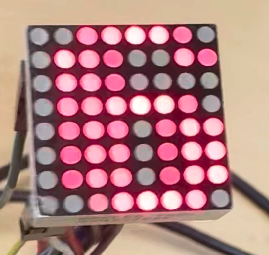
자동 생성된 설명

211606이 left scroll되는 과정은 아래 사진들과 같다.

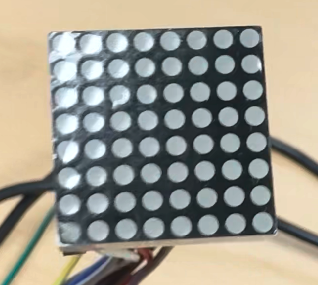


(6) ModefiveTask

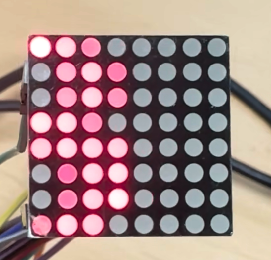
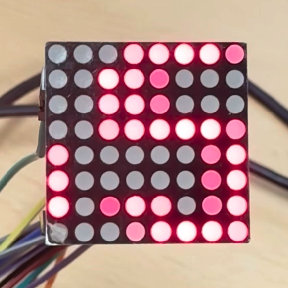
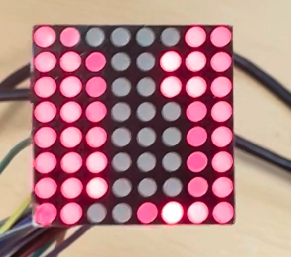
ModefiveTask는 xFiveQueue로 받은 데이터에 따라 숫자를 표시해준다. WriteNumTask와 마찬가지로, xModifyTimer라는 one shot timer를 켜주고 3초가 지나면, 저장한 자릿수에 따라 STUDID배열을 수정해준다. xModifyTimer 또한 flag를 통해 작업을 수행한다. 이 작업이 6번 진행됐다면, mode를 0으로 설정해, 초기상태로 만들어준다. 3초가 지나지 않았는데, User Button Interrupt가 발생할 시, 이전 Task에서 숫자를 바꿔주고 바뀐 숫자를 받아 출력한다. 이 때, Timer는 reset된다.



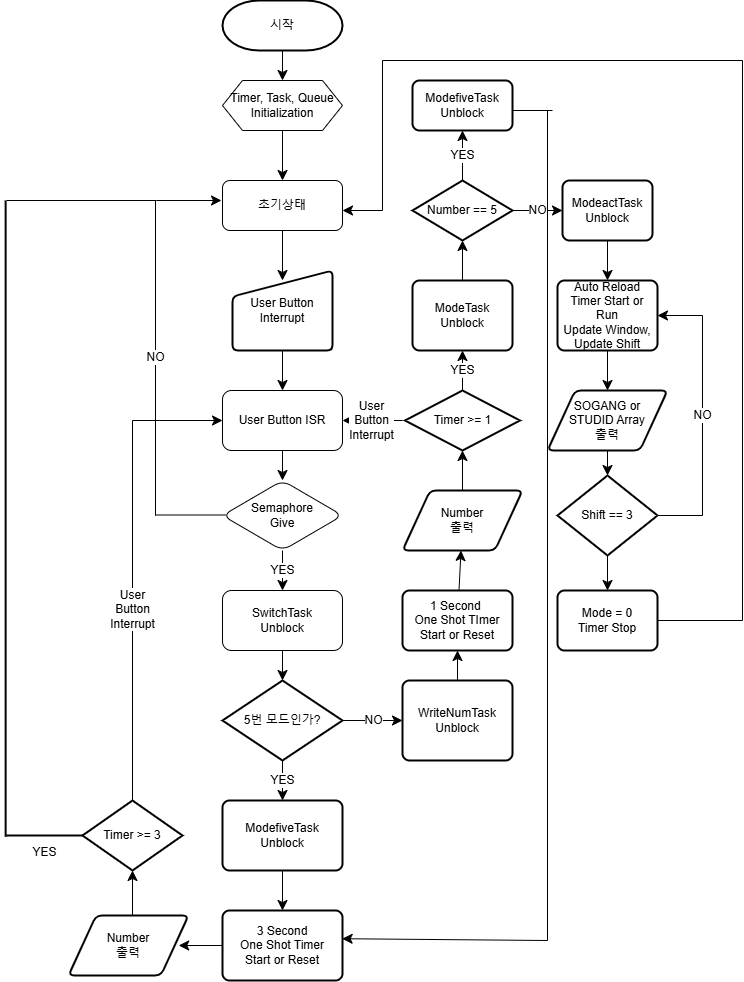
위 사진들은 모드 5에서 STUDID 수정 시에 나타나는 숫자들이다. 해당 숫자에 3초 이상 머무르면 아래와 같이 Blank상태로 간다.



원래는 211606이었던 숫자를 019563으로 수정하고 4번 모드를 실행하면 아래와 같이 동작한다.



**3. Flow Chart**



**4. 고려사항**

(1) Task 수행 중에 발생하는 Interrupt

Task 수행 중에 발생하는 User Button Interrupt가 Task에 영향을 줄 수 있다. 예를 들어, 숫자를 출력하는 무한반복문을 수행 중일 때, User Button Interrupt가 발생하여, 같은 Task를 수행하려 할 수 있다. 이를 고려하여, Task마다 Mode 값을 설정해, Interrupt에 대응했다.

(2) Timer에서의 값 업데이트

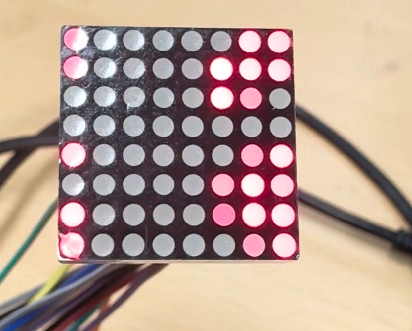
Timer는 Demon Task에게 의탁하기 때문에, Semaphore나 Queue와 같은 Block State에 의존하는 구조체는 사용하기 어렵다. 따라서 Global 변수로 다른 Task와 값을 공유했다. Auto Reload Timer의 경우, Write는 Timer만 할 수 있게 설정했고, 다른 Task들은 주로 Read를 수행했다. 그리고 초기화할 때만, 0으로 설정할 수 있게 했다.

One Shot Timer의 경우, Flag로 Timer가 실행된 뒤, 한번의 Write가 발생하면 처리되는 형식으로 수행했다. 이런 식으로 공유자원에 대한 Race Condition을 최대한 피하기 위해, 설계했다.

(3) 하나의 Task로 여러 작업을 수행

1번 모드, 2번 모드, 3번 모드, 4번 모드를 각각 하나의 Task로 만들 예정이었다. 코드를 작성하는 도중 네 가지 모드의 공통점을 발견하고 하나의 Task로 만들 수 있지 않을까 생각했다. 그래서 타이머만 조금씩 만지고, Queue에 전달하는 값을 통해 4가지 동작을 하나의 Task로 수행시켰다.

**5. 문제점**



위와 같이 1번 모드 ~ 4번 모드까지 마지막 BLANK가 출력되고 다시 첫번째 문자 혹은 숫자가 left scroll되기 시작할 때, 첫번째 column의 row들이 반짝인다. 이는 의도한 것이 아니며, 원래는 점등되지 않아야 할 부분이다.

ModeactTask 부분에서 위 점등을 처리하는데, 계산을 잘못했나 싶어, 계산을 다시 한번 확인하고, 삼항 연산자를 조건문으로 바꿔보고, Timer에서 Update하는 값의 범위를 바꿔보고 했다. 또한, 특정 Column의 문제이기에, 해당 Column에 대한 부분을 continue처리해봤다. 하지만 그렇게 했더니 다음 Column이 점등됐다.

직접 손으로 Debugging도 해봤지만 찾지 못했다. 이 부분은 계속 알아보고 해결해야 할 문제다.

**6. 결론**

하나의 문제점을 제외하고는 모든 기능이 정상적으로 수행됐다. ISR과 Task간의 상호작용, Task와 Task간의 상호작용을 깊게 이해할 수 있었고, Timer를 통해 특정 동작을 하게끔 할 수 있다는 점을 알았다. 특히, Test를 100회 이상 진행하는 동안, 공유자원 사용에 따른 Race Condition이 발생하지 않았다는 점이 괜찮은 설계였다.