논리회로설계실험 project

2016310936 우승민

코드 설명에 앞서 우리 조가 만든 atm을 설명을 하면, ppt에 있는 기본기능인 1일~31일 까지 24시간으로 표기하는 시계와 계좌 8개, A버튼으로 잔액확인, B버튼으로 입금 C버튼으로 출금 E버튼으로 확인 F버튼 초기화를 하였고 추가적인 기능으로는 1일이 지날 때 마다 10%씩 이자를 붙여주고 D버튼을 통해 계좌이체를 할 수 있다. 마지막으로 E버튼을 B버튼과 C버튼을 누르지 않은 상태일 때 계좌만 선택하여 누르면 통장의 내역을 확인할 수 있다.

우선 기본기능부터 설명하면

참고로 우리 조는 button_data를 받은 후 temp_data에 넣고 temp_data case에 따라 실행 상황을 나누었다. 이유는 button을 누른 상태를 유지해주기 위해서 하였다. (B나 C버튼을 누르고 E버튼을 다시 눌러야 하는데 누르기 전까지의 유지시간) (유지를 하지 않으면 button을 떼어내는 순간 바로 default로 이동하기 때문에)

선택된 계좌의 led에 빛이 나는 것은 key_data 입력을 받을 때 바로 해주게 하였다. n이 0일 때 key_data값을 입력 받고 아래 case문에서 key_data에 따라 led_data를 표시하게 해주었다. 참고할 점은 n==1일 때 case문이 작동하게 하였는데 key패드에서 손을 때는 순간 n이 2가되지만 case문에서 default일 때 key_data값을 유지시켜주는 방법을 사용하여 led에 빛이 꺼지지 않게 하였다.

시계 코드는 pre-lab의 방법을 인용하였는데 다만 day를 추가해주었다. day는 hour min sec와 달리 1부터 시작하여 31까지 표현해주었다. Seg_data로 바꾸는 방식도 마찬가지로 decode 함수를 이용하여 해주었다. 참고로 시계를 표시하는 것은 button이 눌리지 않았을 때 표시를 하기위해서 temp_data의 default 상태일때 시계 표시를 할 수 있게 해주었다.

다음으로 계좌를 초기화하는 방법을 설명하면 F버튼이 눌리면 모든 계좌에 0을 넣도록 하였고, 순간적으로 시계를 꺼지게 하기위해서 seg_com에 8'b11111111을 넣어주었다. 그 후 시계를 초기화 하는 방법이 까다로웠는데 이유는 시계를 표현하는 always문을 별개로 만들어주어서 서로 다른 always문에서 값을 변경할 수 없었기 때문이다. 그래서 우리는 새로운 변수인 m을 만들어 주었다. m의 초기값을 0으로 해주고 F버튼이 눌리면 계속 증가하는데 시계의 always 문들을 보면 각각 ||m이 추가 되어있는 것을 볼 수 있다. 여기서 m이 1보다 커질 때 모든 시간 값을 0으로 만

들어 주게 하였다. F버튼을 더 살펴보면 count가 499가 될 때 key_data에 10을 넣어주고 m은 다시 0으로 만들어주고 led_data도 꺼주는 것을 확인할 수 있다. Key_data에 10을 넣어준 이유는 default 상태로 가게 만들어 다른 버튼과 중첩이 되지 않게 하기 위해서 하였다. else문에서는 temp_data를 5로 계속 만들어주는 것을 확인할 수 있다. 이렇게 해야 F버튼 누른 case를 계속 유지해 줄 수 있다.(유지를 하지 않으면 바로 default를 넘어감)

다음 기본 기능인 A버튼을 설명하면 (처음에 뜬금없이 n을 6으로 만드는 코드가 있는데 이유는 B와 C 부분에 나와있다.) 우선 2자리를 표현해야 하기 때문에 seg_com을 2군데 옮기면서 잔상효과를 보일 수 있도록 c라는 변수를 두어 0일 때 첫번째 자리 1일 때 두번째 자리를 표현하도록 하였다. 그리고 각 case안에 led_data의 case를 나누어 각 계좌의 led가 켜져 있을 때 계좌의 값을 출력하도록 하였다. 이를 위해 계좌 내역 또한 decode 함수를 통해 seg_data로 변환해 주었다.

여기서는 5초동안 내역을 보여주어야 하므로 count 범위를 2499까지로 하였고 2499가 되는 순간 led가 꺼지면서 default문으로 이동하여 시계를 표현하도록 하였다. 또한 F버튼과 마찬가지로 else 문에 temp_data를 유지해 주었다.

B버튼과 C버튼은 같이 설명할 수 있는데, A버튼과 비슷한 방식으로 seg_com을 잔상효과를 주었고, 현재 입금/출금 하는 값을 answer이라는 변수로 만들어 주었다. answer 또한 seg_data로 표현해 주기 위해 decode 함수를 사용해 주었다. 여기서 b라는 변수에 1을 넣어주는데 key_data를 입력받는 코드를 확인해보면 n과 b와 d라는 변수에 따라 입력받도록 되어있다. 여기서 d는 D버튼에서 사용하는 것이고 n과 b를 B C 버튼에 따라 사용하게 만들었는데, 만약 n이 0인 상태에서 key패드를 누르면 계좌를 선택하는 것이고 계좌 선택 후 n이 1이 된다 그 후 key패드에서 손을 때면 default로 가서 n이 2로 갈지 0으로 갈지 b에 값에 따라 정해진다.

간단히 말하면 B버튼이나 C버튼을 누르지 않고 key패드를 누르면 n은 0인상태를 유지하게 되어 계속 계좌 선택을 하는 과정이 반복되는 것이고, key패드를 누른 후 B버튼이나 C버튼을 누르면 n이 2가 되어 answer 즉 입금/출금 하는 값을 입력 받을 수 있게 해주었다. 2자리 숫자를 받아야 하므로 answer <= answer *10 + a 방법을 이용했는데 처음에 숫자 하나를 누르고 나면 n은 4가 되면서 0*+a가 answer에 들어가고 그 후 숫자 하나를 더 받을 수 있는 상태가 된다.(n=4) 이때 숫자를 더 받으면 answer에 a *10 +b 가 들어가 두자리를 입력 받고 그 후 n이 5가 된다. 현재 seg_data는 내가 넣어준 answer값이 출력 되고 led_data는 처음에 고른 계좌가 보여질 것이다. 이후 E버튼을 누르면 b가 0이 아니므로 첫번째 if문으로 들어가고 입금/출금된 계좌의 내역이 seg로 보여지게 된다. 그 후 5초가 지나면(count ==2499) led를 꺼주고 n과 b값을 초기화 해주며 temp_data를 다시 8로 보내어 default인 시계를 표시할 수 있도록 만들어 준다. 여기서도 else(count!= 2499)상태일 때 temp_data를 유지해 준다.

C버튼과 B버튼의 차이점은 입금과 출금이기 때문에 출금은 내역을 벗어나는 범위를 적용할 수 없다. 그래서 if문을 통해 가지고 있는 금액보다 많은 값이 입력되면 ok에 1을 넣어주고 E버튼에서 X표시를 띄우면서 취소되고 원래 가지고 있던 내역을 표시하도록 해주었다. 범위 내의 숫자를 입력 받으면 ok표시에 0을 넣어주어 O 표시를 해준다. E 버튼을 누른 후 answer을 초기화 해주는 것도 잊지 않았다.

추가기능 3가지 중 이자에 대해 먼저 설명하면

Day가 1일이 지날 때 마다 모든 계좌의 내역에 + %10을 해주면 된다.(ex : a1 <= a1 + a1%10) 그냥 코드만 쓰면 될 것 같지만 시계와 같은 문제로 서로 always문이 달라서 day에서 코드를 작성할 수 없다. 그래서 day문에서 profit이라는 변수를 만들어 profit이 1이면 0 0이면 1이 되도록 만들어 준다. 그리고 계좌를 변경하는 always문에서는 (profit이 1이고 qwer이 0일 때, profit이 0이고 qwer이 1일 때) 이자를 붙여주고 각각 qwer을 1이면 0 0이면 1로 만들어준다. 과정을 설명하면 초기값은 둘 다 0이므로 작동을 하지 않고 1일이 지나면 profit만 1이되어 이자가 붙는다. 동시에 qwer을 1로 만들어준다. 이번엔 둘 다 1이므로 이자가 붙지 않고 1일이 지나야 profit이 0으로 되어 서로 다른값이 되어 이자가 붙는다. 동시에 qwer을 0으로 만들어준다.

정리하자면 하루가 지난 순간에만 서로 다른 값을 만들어주어 이자를 붙여주고 평소에는 둘의 값이 같게 유지해 주는 방법을 사용한 것이다.

다음으로 계좌이체에 대해 설명하면 D버튼을 이용하였다. 매커니즘은 B C 버튼과 같지만, 차이점은 계좌를 2개를 선택해야 한다는 점이다. 여기서 변수 d를 사용하였다. 처음에 target(입금 받는 계좌를 의미한다.) 에 아무 값도 입력 받지 않고 있으면 d에 1을 넣어준다. Key_data 입력 받는 코드를 보면 n이 0이고 d가 0이면 처음 송금할 계좌를 선택하는 것이고 D 버튼을 눌러 d를 1로 만든 후 key패드를 누르면 target에 key_data가 들어간다. 송금된 계좌를 선택하는 것이다. 그러면서 d를 2로 만들어준다. d가 2가 되면 B C에서 b가 1일때와 같은 방법으로 두 자리 숫자를 입력 받는다. 그 후 출금할 때처럼 송금할 계좌의 현재 내역이 입력한 숫자보다 작으면 ok에 1을 넣어 X를 표시하고 내역이 입력보다 크면 계좌에서 돈을 빼고 target 계좌에서 돈을 더해준 후 ok에 0을 넣어 E버튼에서 O를 표시하였다. 이후 처음 계좌의 잔액이 seg에 보이게 된다.

마지막으로 거래내역을 설명하면 각 계좌마다 array와 array에 값을 순서대로 받을 수 있게 또 다른 변수를 (ex : a01_1)를 만들어 금액에 변동이 있을 때 마다 값이 순서대로 들어가게 해주었다. 입금과 출금 계좌이체 이자까지 계좌에서 돈이 변동하면 a1[a1_01] <= a1 + answer 과 a1_01 <= a1_01 +1을 통해 array에 변동된 숫자를 넣어주고 계속 변할 것을 감안하여 각 array의 전용 정수 또한 증가시켜 주었다. 확인 방법은 계좌를 선택하고 다른 버튼을 누르지 않고 바로 E버튼을 누

르는 것인데 E버튼의 코드를 보면 처음 if문이 b와 d가 0이 아닐 때 작동한다. 그러면 else문은 둘다 0일 때 작동한다. (B와 C, D 버튼이 눌리지 않았을 때) else문 코드를 보면 seg에 xx를 표현한다. 여기서 xx는 선택한 계좌의 array의 i번째 정수이다. xx <= a01[i] 여기서 led_data의 case에 따라 계좌를 선택하도록 하였고 각 case문 안에서 i를 0에서 10까지 1초단위로 증가시켜주면서 표현을 해주었고 i가 10이 되는 순간 led를 꺼주고 i를 0으로 초기화 해주며 temp_data에 8을 넣어 다시 시계가 표시되도록 해주었다. 참고로 F버튼을 누를 때 array값과 계좌 고유의 정수도 0으로 초기화하도록 만들어 주었다.