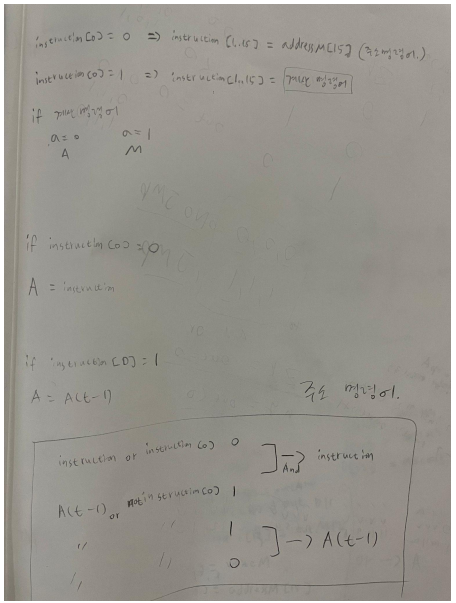
PORTFOLIO

박주한

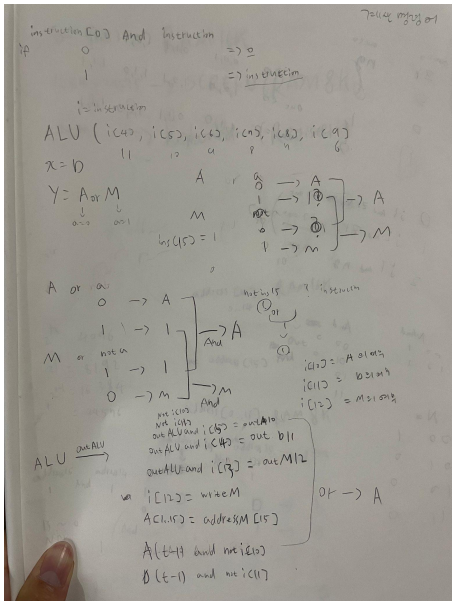
구현한 작품

1. 컴퓨터
2. 피카츄 배구 게임
3. 아날로그 시계

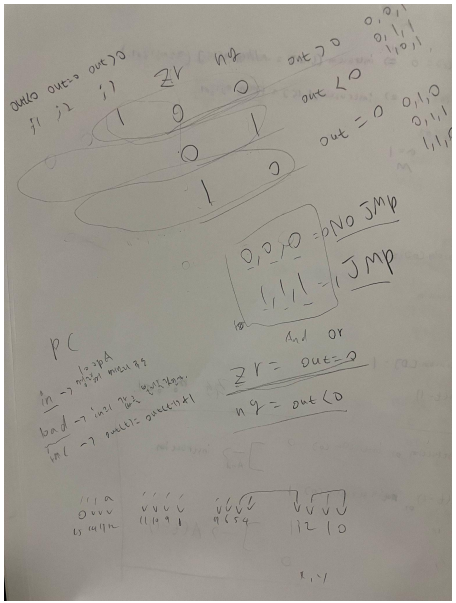
컴퓨터



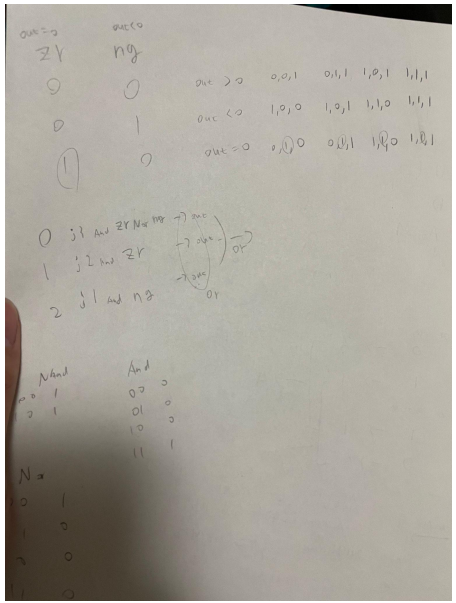
명령어 구분



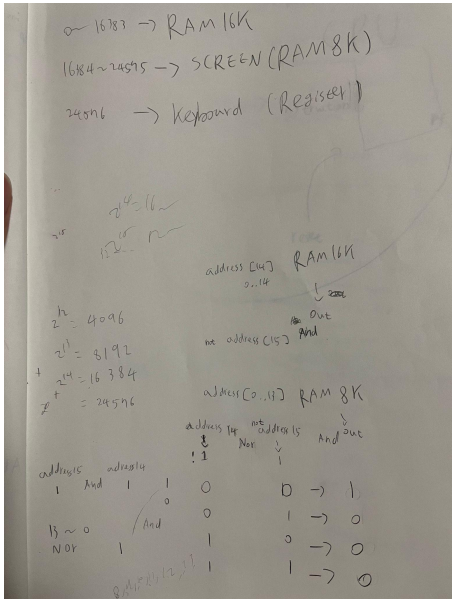
ALU 값처리



점프 처리 - 1



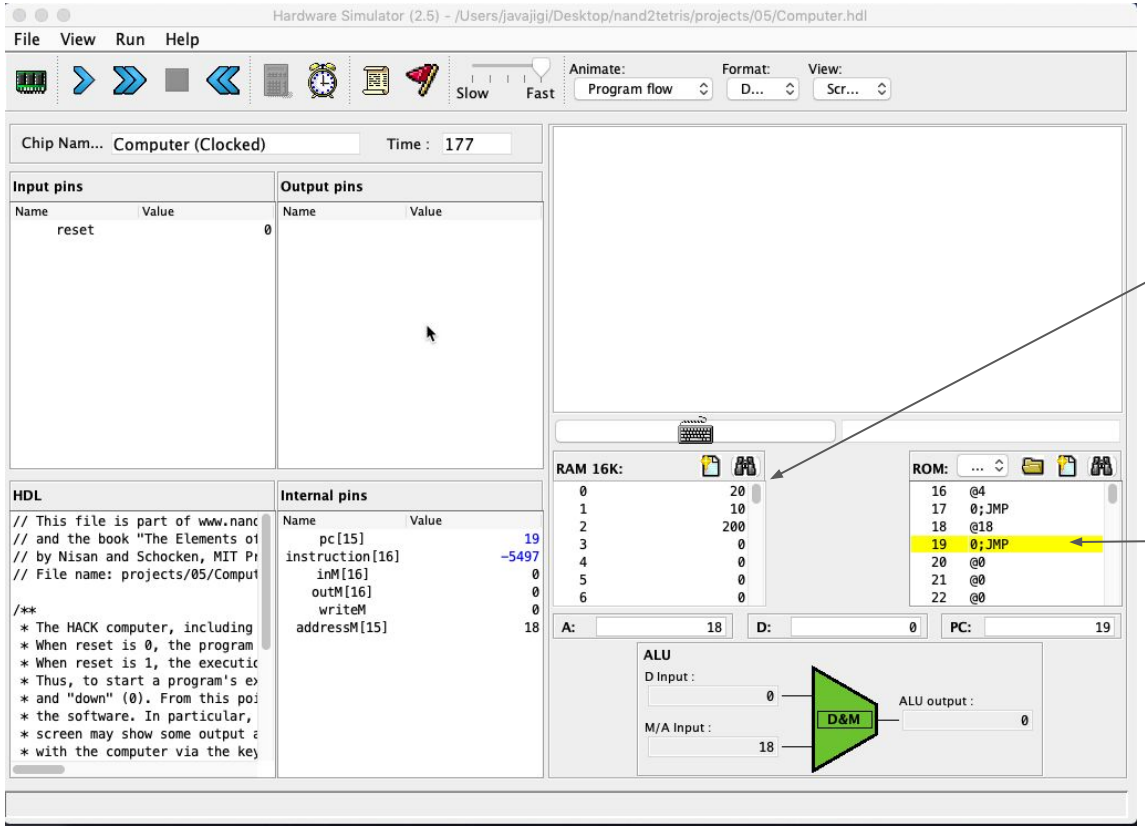
점프 처리 - 2



데이터 메모리 주소 설정

```
CHIP CPU {

    IN  inM[16],         // M value input  (M = contents of RAM[A])
        instruction[16], // Instruction for execution
        reset;           // Signals whether to re-start the current
                         // program (reset==1) or continue executing
                         // the current program (reset==0).

    OUT outM[16],        // M value output
        writeM,          // Write to M?
        addressM[15],    // Address in data memory (of M)
        pc[15];          // address of next instruction

    PARTS:
    Not(in=instruction[15], out=notins15);
    Or16-1(a=instruction, b=instruction[15], out=orins1);
    Or16-1(a=loopA, b=notins15, out=orloopA);
    And16(a=orins1, b=orloopA, out=andA);
    And16-1(a=andA, b=notins15, out=resultA);

    Not(in=instruction[12], out=notins12);
    Or16-1(a=loopA, b=instruction[12], out=orA);
    Or16-1(a=inM, b=notins12, out=orM);
    And16(a=orA, b=orM, out=orAM);
    ALU(x=loopD, y=orAM, zx=instruction[11], nx=instruction[10], zy=instruction[9], ny=instruction[8], f=instruction[7], no=instruction[6], out=outALU, zr=zrALU, ng=ngALU);
    And16-1(a=outALU, b=instruction[15], out=resultALU);
    And16-1(a=resultALU, b=instruction[3], out=outM);
    And(a=instruction[3], b=instruction[15], out=and315);
    And(a=and315, b=instruction[15], out=writeM);
    Or16(a=resultA, b=resultALU, out=inA);
    Or(a=notins15, b=instruction[5], out=loadA);
    And(a=instruction[4], b=instruction[15], out=loadD);
    ARegister(in=inA, load=loadA, out=loopA, out[0..14]=addressM);
    DRegister(in=resultALU, load=loadD , out=loopD);

    Nor(a=zrALU, b=ngALU, out=orzrng);
    And(a=instruction[0], b=orzrng, out=outj3);
    And(a=instruction[1], b=zrALU, out=outj2);
    And(a=instruction[2], b=ngALU, out=outj1);
    Or(a=outj3, b=outj2, out=orj3j2);
    Or(a=orj3j2, b=outj1, out=outloadPC);
    And(a=outloadPC, b=instruction[15], out=loadPC);
    PC(in=loopA, load=loadPC, inc=true, reset=reset, out[0..14]=pc);
}
```

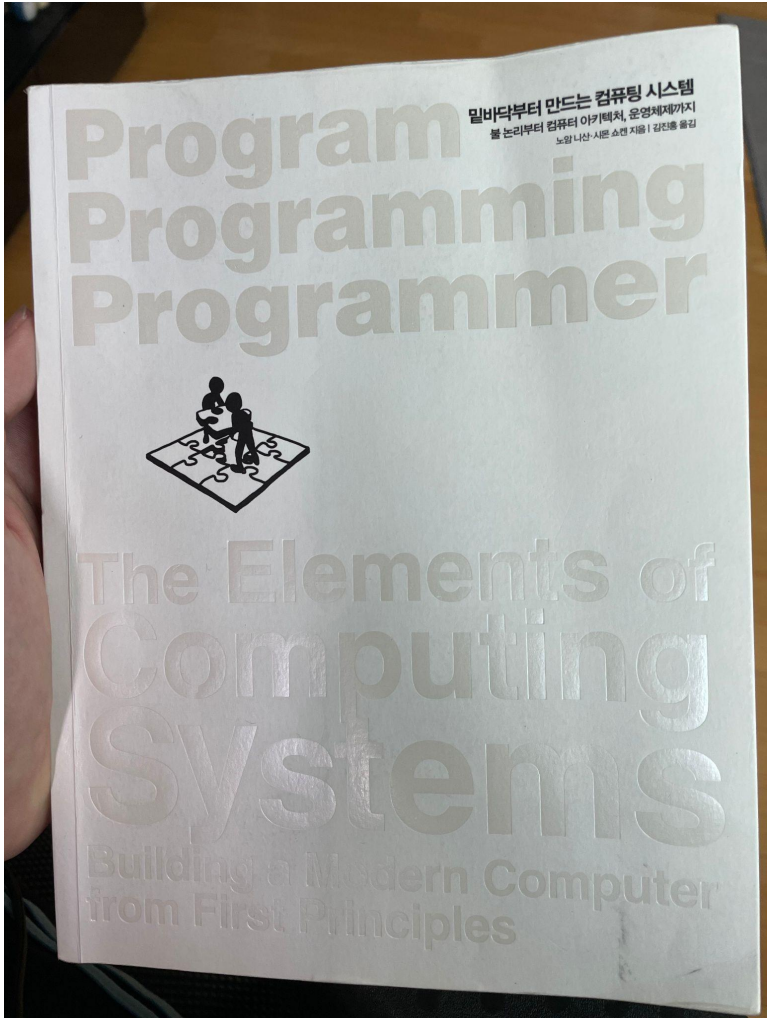
CPU 구현



미리 저장한 값과 어셈블리 코드에 의한 결과값을 확인할 수 있다.

R0과 R1(두 개의 가장 위 RAM 위치) 에 현재 저장된 값들을 곱하여 결과를 R2에 저장하는 어셈블리 코드이다.

구현한 CPU와 메모리를
이용하여 만든 컴퓨터



‘밑바닥부터 만드는 컴퓨팅 시스템’ 책의 5장까지 읽으며 구현한 컴퓨터 입니다.

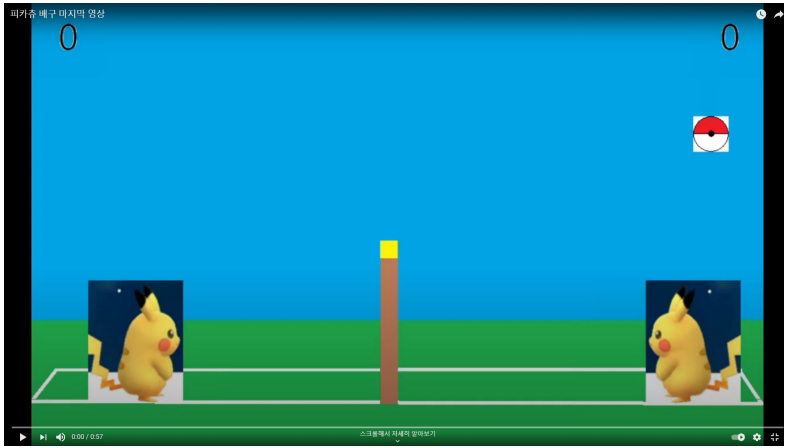
처음 접해보는 하드웨어 시스템 이기에 삽질도 많이 하고 상당히 지저분한 회로로 구현되어있습니다.

그래도 처음부터 끝까지 힌트 없이 제 스스로의 힘으로 구현했다는 점이 뿌듯합니다.

하드웨어 칩을 직접 만들며 제일 어려웠던 점은 python, java 등등 고수준 언어는 if문이 있어 조건에 따른 기능 수행을 쉽게 구현할 수 있지만 하드웨어 칩을 구현할때는 if문이 없었다는 점입니다.

평소 하드웨어에 관심이 많았고 컴퓨터가 어떤 방식으로 동작되는지 궁금했었는데 책을 읽고 직접 만들어보면서 컴퓨터는 무식한데 속도만 빠르다는 것을 다시한번 느꼈고 최초로 컴퓨터를 만들어낸 사람들과 지금의 컴퓨터로 발전 시킨 사람들이 새삼 대단하게 느껴졌습니다.

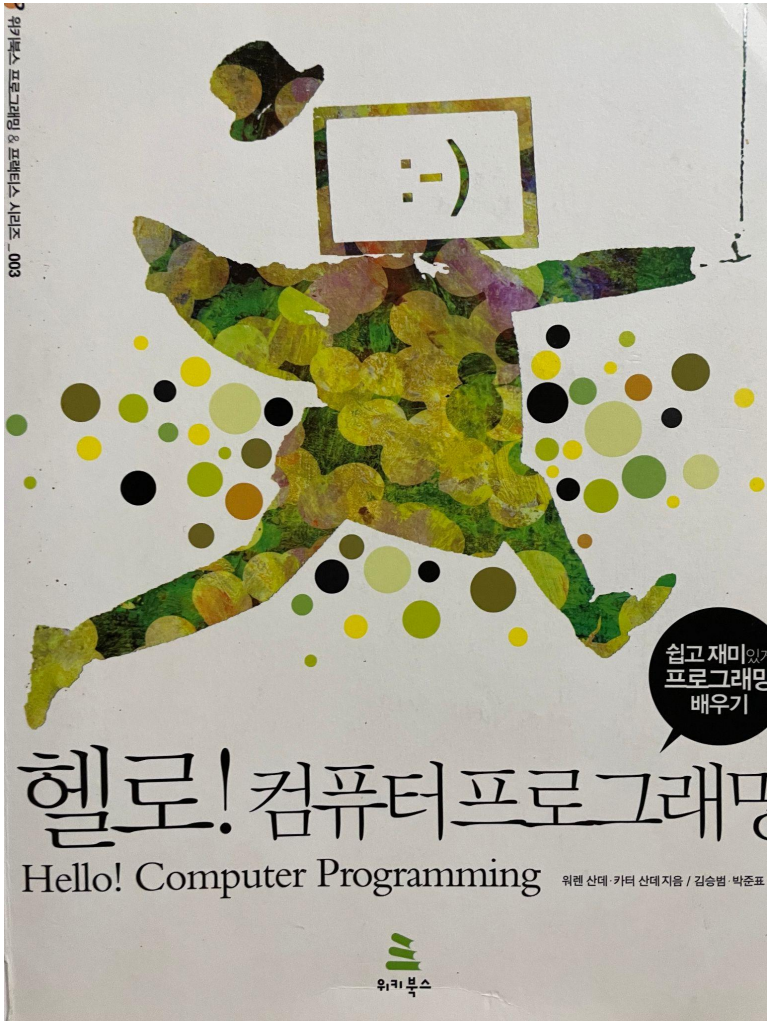
피카츄 배구 게임



피카츄 배구 게임 시작시



피카츄 배구 게임 한쪽 패배시



책에 나오는 Pygame 모듈을 이용하여 구현하였습니다.

물리 엔진을 적용하지 않아 공이 물리법칙을 무시한채 움직이는 점은 아쉽습니다.

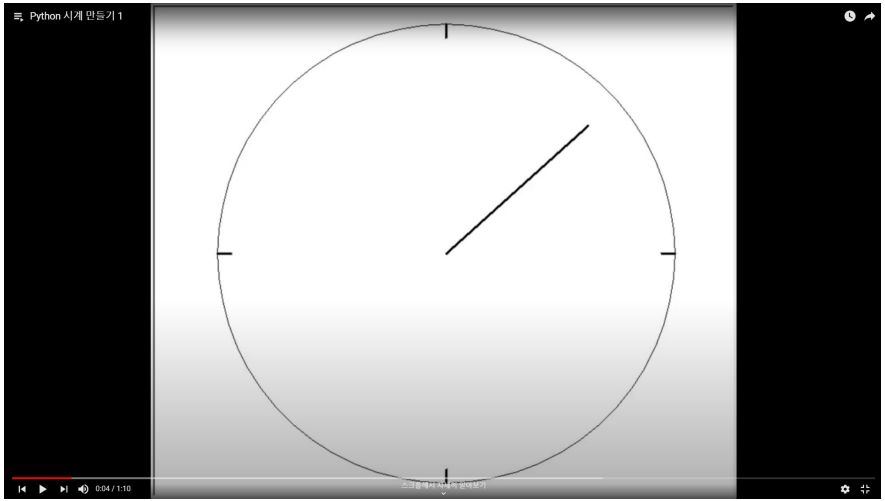
원작 게임에선 두가지 키를 동시에 눌러 공격 하는 기술이 존재하지만 두가지 키를 동시에 눌렀을 때 이벤트가 제대로 할당되지 않아 다양한 공격을 구현하기 위해 여러개의 키들을 따로 눌러야 했던 점 또한 아쉽습니다.

처음 만들어본 게임이여서 부족한 기능이 많고 잔버그도 많습니다.

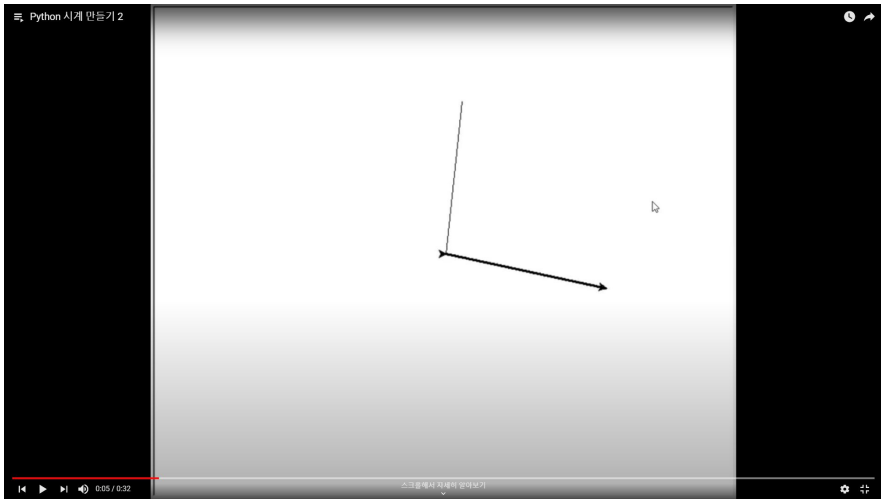
또한 아직 클래스를 잘 이해하지 못했고 잘 사용하지 못했기 때문에 Pygame모듈이 클래스를 이용해야 해서 이 부분에서 어려움을 느끼고 많은 삽질을 했습니다.

그래도 함수에 대해 제대로 이해할 수 있었고 아무리 어려운 문제가 주어져도 하나하나 천천히 해나가면 뭐든지 다 해결할 수 있겠다는 자신감이 생겼습니다.

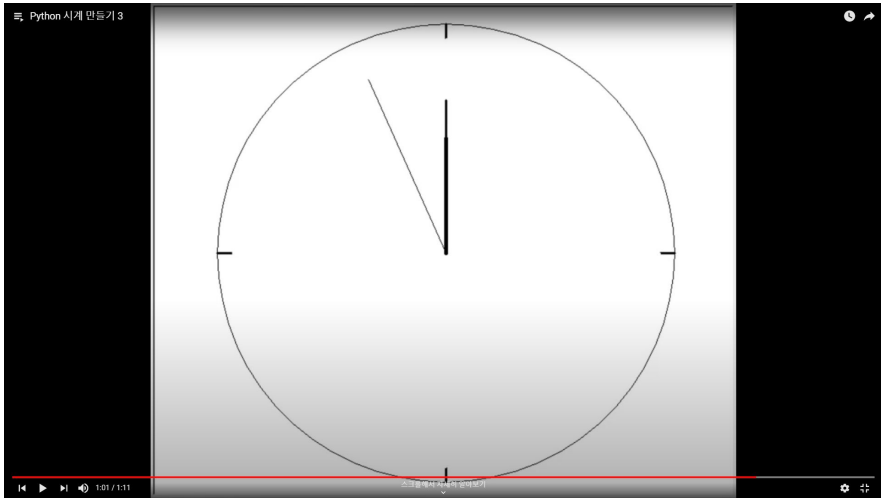
아날로그 시계



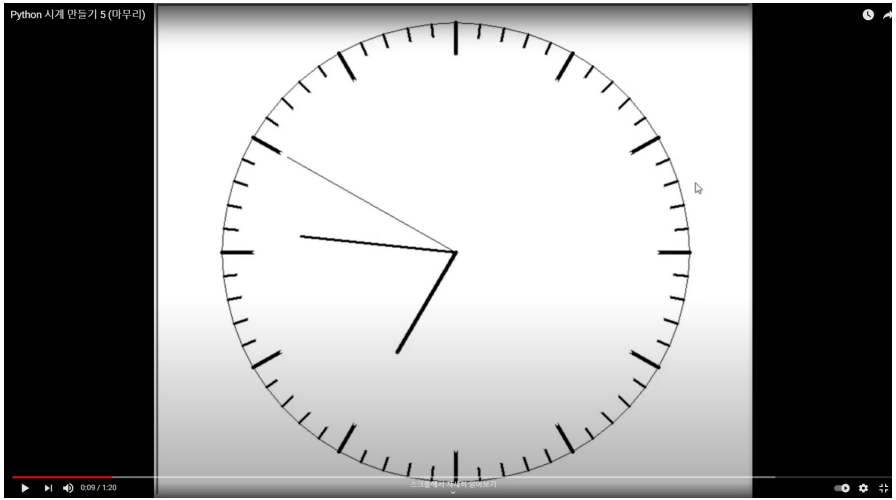
초침의 움직임 구현



분침의 움직임 구현



12시 부터 동작하는 시계 구현



현재 시간을 적용한 시계 구현



책에 나오는 터틀 모듈을 이용하여 구현했습니다.

프로그래밍을 배우고 처음으로 따라 작성한 코드가 아닌 직접 생각하여 작성한 프로그램이기 때문에 중복이 많고 변수명도 a, b 같이 이해하기 어려운 이름을 가지고 있습니다.

처음 if문을 사용하는 '초침이 한바퀴 돌았을 때 분침을 움직이는 기능'을 구현하는 것이 제일 어려워서 어떻게 구현해야 하나 많은 고민을 했습니다.

하루 종일 고민하다 밥을 먹을 때 갑자기 해결책이 떠올라서 방으로 달려가 기능을 마저 구현하고 오류까지 해결한 그날은 아직도 잊혀지지 않는 짜릿한 하루였습니다.

시계를 완성시키고 내가 만든 결과물을 지켜보며 프로그래머 들이 왜 코딩을 하고 프로그램을 만들어 가는지, 이렇게 재밌는걸 이제야 알게 됐다며 후회하고 다음엔 더 재밌고 어려운걸 만들어보고 싶다고 느꼈습니다.

감사합니다