

8. 1 자리 7 세그먼트 표시장치

1. 7 세그먼트 표시장치

7 세그먼트 표시장치(7-segment display)는 7개의 선분으로 숫자나 글자를 표시하기 위해 발광 다이오드(LED)를 사용하여 만든 출력 장치의 일종이다. 일반적으로 7 세그먼트 표시장치에는 그 이름과 다르게 7개 선분에 소수점(decimal point) 표시를 위한 LED가 추가되어 8개의 요소들을 1바이트의 데이터로 제어할 수 있도록 만들어져 있다. 7 세그먼트 표시장치에는 두 자리 이상의 숫자를 표시하도록 만들어진 경우도 많지만, 이 장에서는 한 자리 숫자 표시를 위한 7 세그먼트 표시장치를 살펴본다.

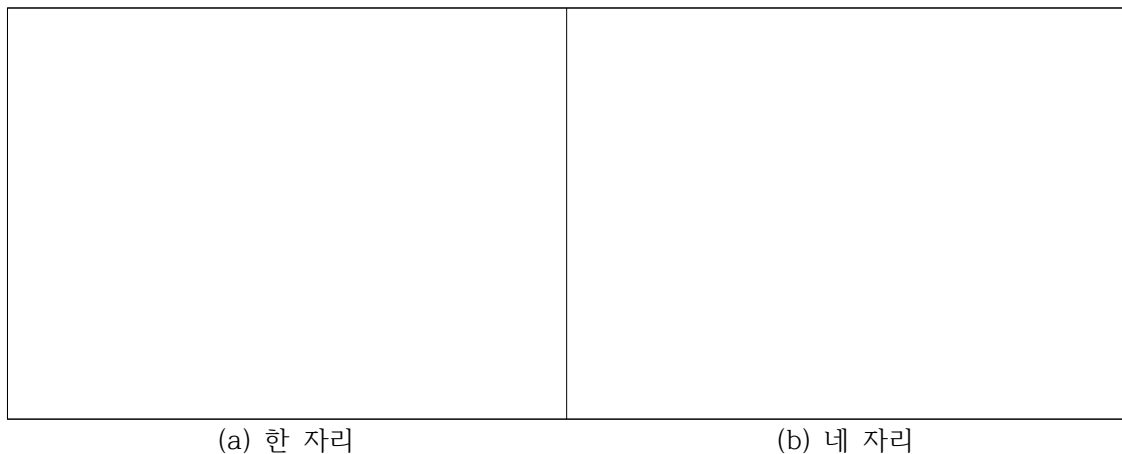


그림 1. 7 세그먼트 표시장치

한 자리 7 세그먼트 표시장치는 공통 양극(common anode) 방식과 공통 음극(common cathode) 방식의 두 가지가 있으며 핀의 개수와 배치는 동일하다. 핀 번호는 소수점이 있는 면의 가장 왼쪽 핀이 1번 핀에 해당하며 반시계 방향으로 번호가 증가한다.

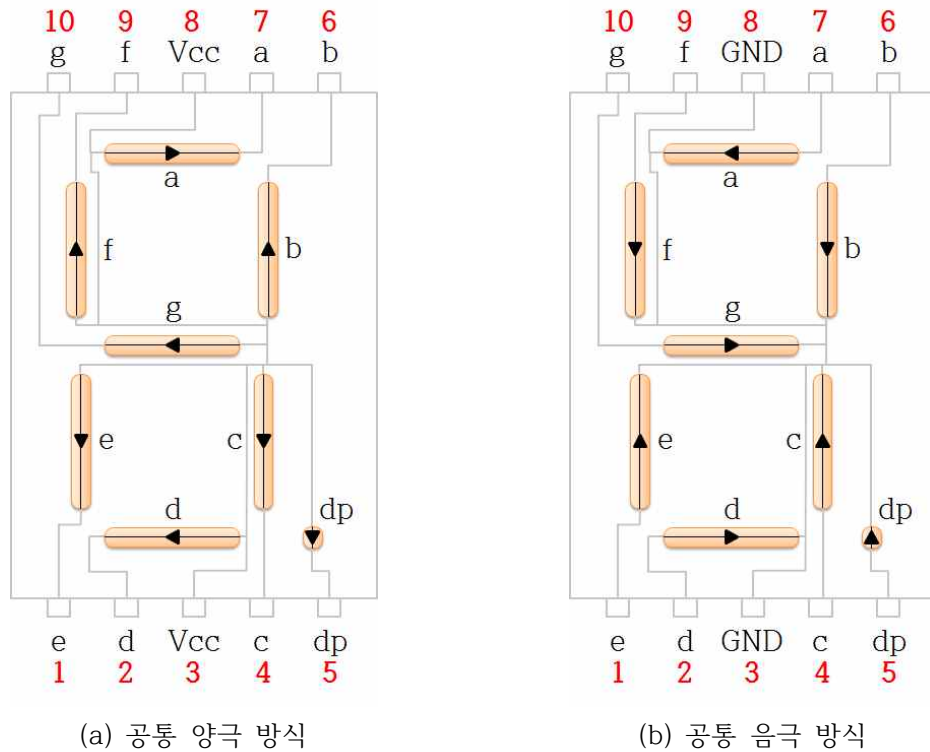


그림 2. 7 세그먼트 표시장치의 핀 배열

7 세그먼트 표시장치에서 10개의 핀 중 2개의 VCC나 GND를 연결하는 공통 핀이며 8개의 해당 요소를 제어하는 제어 핀이다. 공통 핀에 VCC를 연결하고 각 제어 핀을 GND에 연결하면 불이 들어오는 방식이 공통 양극 방식이며 공통 핀을 GND에 연결하고 각 제어 핀을 VCC에 연결하면 불이 들어오는 방식이 공통 음극 방식이다. 이처럼 공통 양극 방식과 공통 음극 방식은 해당 요소를 제어하는 방법이 반대이므로 7 세그먼트 표시장치를 사용하기 이전에 공통 양극 방식인지 공통 음극 방식인지 먼저 확인해야 한다. 한 자리 7 세그먼트 장치를 제어하기 위해서는 8개의 제어 선이 필요하므로 아두이노의 데이터 출력 핀 8개를 사용하여야 한다.

2. 1자리 7 세그먼트 모듈

실험에 사용할 모듈은 공통 양극 방식의 1자리 7 세그먼트 모듈이다. DM-UStyle EX 보드에는 1자리 7 세그먼트 표시장치가 포함되어 있지만 포함된 1자리 7 세그먼트 표시장치는 디지털 출력 확장을 위한 74595 직렬 입력 병렬 출력 (Serial In Parallel Out, SIPO) 칩을 통해 연결되어 있으므로 이 장에서는 아두이노의 디지털 핀을 이용하여 제어가 가능한 별도의 모듈

을 사용한다. 1자리 7 세그먼트 모듈의 회로도는 그림 3과 같다.

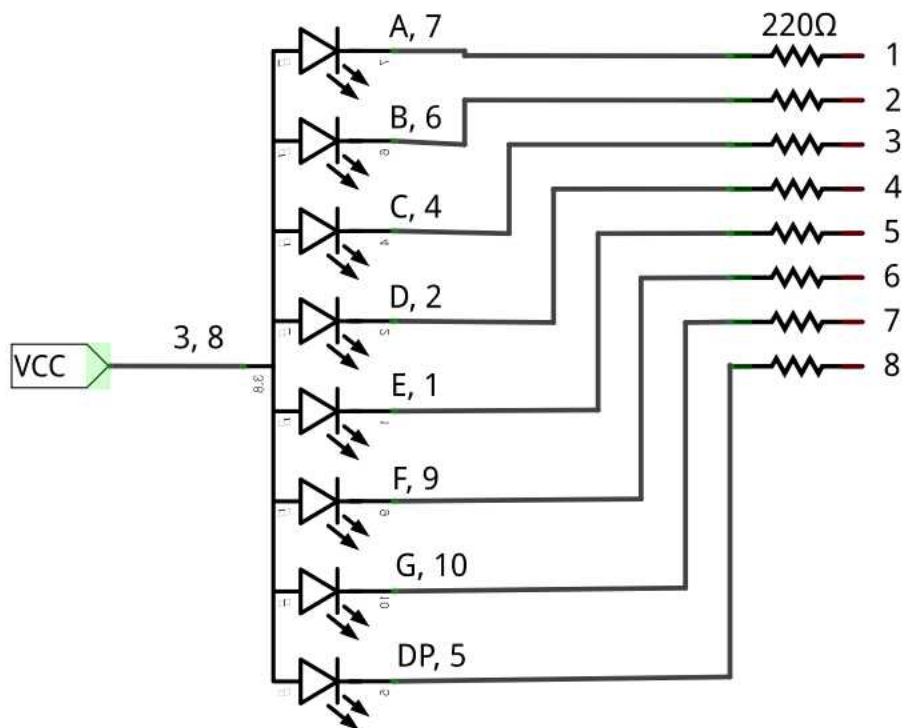


그림 3. 1자리 7 세그먼트 모듈 회로도

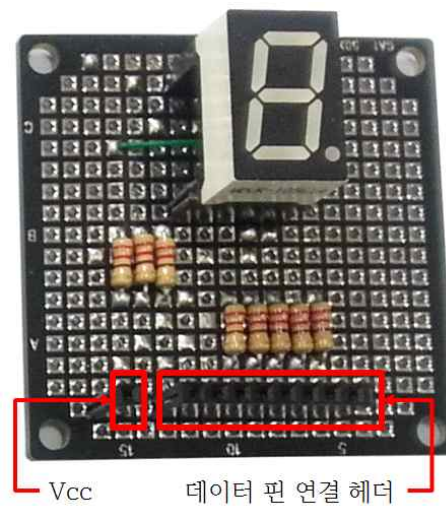


그림 4. 1자리 7 세그먼트 모듈

3. 숫자 표시하기

7 세그먼트 모듈을 디지털 5, 6, 7, 8, 10, 11, 12, 13번 핀에 연결해 보자.



그림 5. 1자리 7 세그먼트 모듈 연결

7 세그먼트 모듈은 공통 양극 방식이므로 각 요소 제어를 위한 제어 핀에 HIGH 값을 출력하면 꺼지고, LOW 값을 출력하면 켜진다. 스케치 1은 7 세그먼트 모듈의 요소를 a → b → c → d → e → f → g → dp 순서로 0.1초 간격으로 켜는 스케치의 예이다.

스케치 1. 1자리 7 세그먼트 테스트

```
int pins[] = {5, 6, 7, 8, 10, 11, 12, 13};    // 7세그먼트 모듈 연결 핀

void setup(){
  for(int i = 0; i < 8; i++){                // 7 세그먼트 모듈 연결 핀을 출력으로 설정
    pinMode(pins[i], OUTPUT);
  }
}

void loop(){
  for(int i = 0; i < 8; i++){
    for(int j = 0; j < 8; j++){
      if(i == j) digitalWrite(pins[j], LOW); // LOW 출력으로 켜
      else digitalWrite(pins[j], HIGH);      // HIGH 출력으로 끄
    }
  }
}
```

```
    delay(100);  
  }  
}
```

스케치 1은 기본적으로 LED 모듈에서 하나의 LED를 켜는 코드와 동일하다. LED 모듈의 회로도와 비교해 보면 LED 모듈의 경우 공통 음극 방식으로 연결되어 있는 것을 제외하면 1자리 7 세그먼트 모듈과 동일하다. 7 세그먼트에 숫자를 표시해 보자. 7 세그먼트에 숫자를 표시하는 예는 그림 6과 같다.

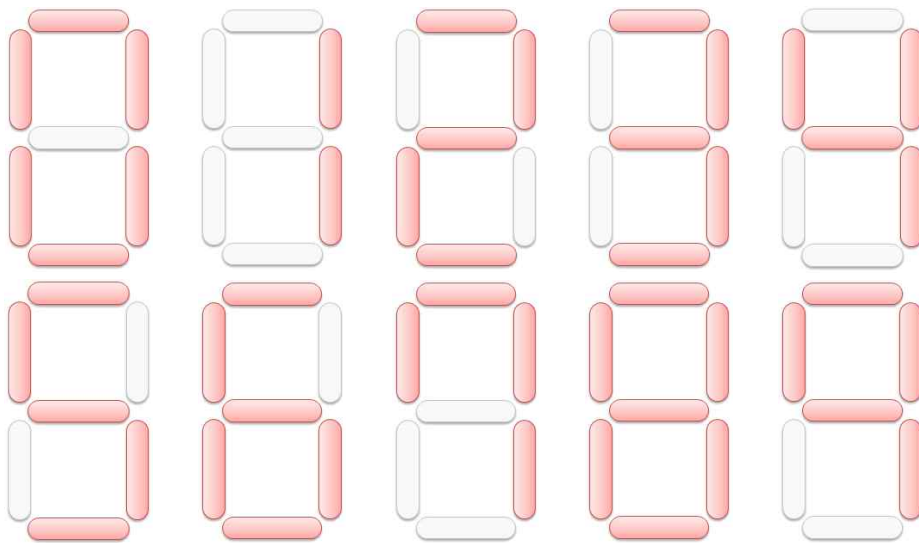


그림 6. 7 세그먼트 표시장치의 숫자 표현

그림 6의 각 표현에 해당하는 비트 표현은 다음과 같다. 비트 표현에서 켜지는 요소는 1로 꺼지는 요소는 0으로 표시하였다.

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, // 0b11111100
    0x60, // 0b01100000
    0xDA, // 0b11011010
    0xF2, // 0b11110010
    0x66, // 0b01100110
    0xB6, // 0b10110110
    0xBE, // 0b10111110
    0xE4, // 0b11100100
    0xFE, // 0b11111110
    0xE6 // 0b11100110
};
```

숫자 패턴을 이용하여 0.5초 간격으로 0에서 9까지의 숫자를 표현하는 스케치의 예가 스케치 2이다. 스케치 2에서 한 가지 유의할 점은 숫자 패턴에서 켜지는 요소가 1로 정의되어 있는 반면 실험에 사용하는 7 세그먼트 표시장치가 공통 양극이므로 비트값을 반전시켜 출력하여야 한다는 점이다.

스케치 2. 1자리 7 세그먼트에 숫자 표시하기 - 증가

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int pins[] = {5, 6, 7, 8, 10, 11, 12, 13}; // 7 세그먼트 모듈 연결 핀
int index;

void setup(){
    for(int i = 0; i < 8; i++){ // 7 세그먼트 모듈 연결 핀을 출력으로 설정
        pinMode(pins[i], OUTPUT);
    }
}
```

```

    index = 0;                                // 현재 출력 숫자
}

void loop(){
    index = (index + 1) % 10;                  // 10이면 다시 0으로 되돌림

    for(int i = 0; i < 8; i++){
        boolean on_off = !bitRead(patterns[index], i); // 반전 출력
        digitalWrite(pins[i], on_off);
    }

    delay(500);
}

```

스케치 2의 경우 0에서 9로 숫자가 증가되면서 표시된다. 숫자를 9에서 0으로 감소되는 방향으로 표시하기 위해서는 현재 표시되는 숫자를 나타내는 index 값을 감소시키면 된다. 단, 숫자가 증가하는 경우 9보다 큰 경우 다시 0으로 만들기 위해 스케치 2에서 나머지 연산자를 사용하였지만, 숫자가 감소하는 경우 0보다 작은 경우 다시 9로 만들기 위해 if 문을 사용하였다는 차이가 있다.

스케치 3. 1자리 7 세그먼트에 숫자 표시하기 - 감소

```

// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int pins[] = {5, 6, 7, 8, 10, 11, 12, 13}; // 7 세그먼트 모듈 연결 핀
int index;

void setup(){
    for(int i = 0; i < 8; i++){                // 7 세그먼트 모듈 연결 핀을 출력으로 설정
        pinMode(pins[i], OUTPUT);
    }
}

```

```

    }
    index = 0;                                // 현재 출력 숫자
}

void loop(){
    index = index - 1;
    if(index < 0) index = 9;                  // -1이면 다시 9로 되돌림

    for(int i = 0; i < 8; i++){
        boolean on_off = !bitRead(patterns[index], i); // 반전 출력
        digitalWrite(pins[i], on_off);
    }

    delay(500);
}

```

4. 타이머

1자리 7 세그먼트 표시장치를 이용하여 0에서 9까지를 반복하는 간단한 타이머를 만들어 보자. 기본적인 타이머는 스케치 2를 이용하여 구현 가능하며 숫자가 바뀌는 시간을 1초로 변경하면 초 단위의 타이머를 만들 수 있다. 하지만 delay 함수를 사용하는 경우 사용자의 입력을 받아들이는 등의 작업에 제한이 있으므로 스케치 3은 스케치 2에서 delay 함수를 제거하고 millis 함수를 이용하도록 작성한 초 단위 타이머의 예이다.

스케치 4. 초 단위 타이머

```

// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int pins[] = {5, 6, 7, 8, 10, 11, 12, 13}; // 7 세그먼트 모듈 연결 핀
int index;

```

```
unsigned long time_previous, time_current;
int INTERVAL = 1000;

void setup(){
    for(int i = 0; i < 8; i++){          // 7 세그먼트 모듈 연결 핀을 출력으로 설정
        pinMode(pins[i], OUTPUT);
    }
    index = 0;                          // 현재 출력 숫자
    time_previous = millis();
}

void loop(){
    time_current = millis();

    if(time_current - time_previous >= INTERVAL){
        time_previous = time_current;

        index = (index + 1) % 10;

        for(int i = 0; i < 8; i++){
            boolean on_off = !bitRead(patterns[index], i);    // 반전 출력
            digitalWrite(pins[i], on_off);
        }
    }
}
```

타이머에 시작과 정지 기능을 추가해 보자. 시작과 정지는 디지털 9번 핀에 연결된 버튼을 사용한다. 초기 상태는 정지 상태로 하며 버튼을 누르면 타이머가 동작하고 다시 버튼을 누르면 일시 정지하는 것으로 해 보자.

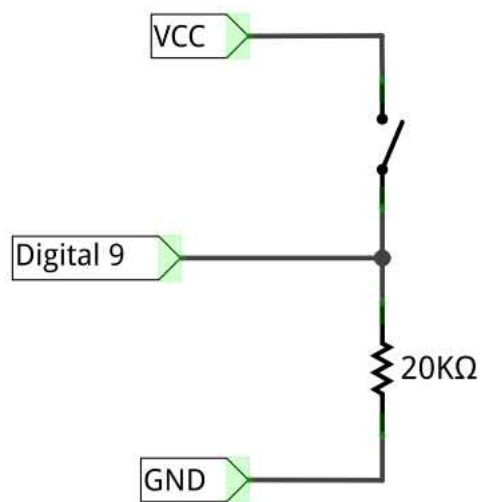


그림 7. 풀다운 저항 연결

스케치 5. 시작과 정지가 가능한 초 단위 타이머

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int pins[] = {5, 6, 7, 8, 10, 11, 12, 13}; // 7 세그먼트 모듈 연결 핀
int index;
int button_pin = 9; // 버튼 연결 핀
boolean going = false; // 정지 상태에서 시작

unsigned long time_previous, time_current;
int INTERVAL = 1000;

void setup(){
    for(int i = 0; i < 8; i++){ // 7 세그먼트 모듈 연결 핀을 출력으로 설정
        pinMode(pins[i], OUTPUT);
    }
    time_previous = millis();
    pinMode(button_pin, INPUT);
}
```

```
index = 0;                                // 현재 출력 숫자
for(int i = 0; i < 8; i++){                // 현재 숫자 표시
    boolean on_off = !bitRead(patterns[index], i);    // 반전 출력
    digitalWrite(pins[i], on_off);
}
}

void loop(){
    boolean pressed = digitalRead(button_pin);    // 버튼 입력
    if(pressed){
        delay(500);                                // 디바운스
        going = !going;                            // 시작과 정지 토글
    }
    else{
        if(!going) time_previous = millis();    // 정지 상태에서 이전 시간 갱신
    }

    if(going){                                      // 시작 상태인 경우
        time_current = millis();

        if(time_current - time_previous >= INTERVAL){
            time_previous = time_current;

            index = (index + 1) % 10;

            for(int i = 0; i < 8; i++){
                boolean on_off = !bitRead(patterns[index], i);    // 반전 출력
                digitalWrite(pins[i], on_off);
            }
        }
    }
}
```

5. 가변 저항으로 7 세그먼트 숫자 제어

아날로그 3번 A3 핀에 가변저항을 연결해 보자.

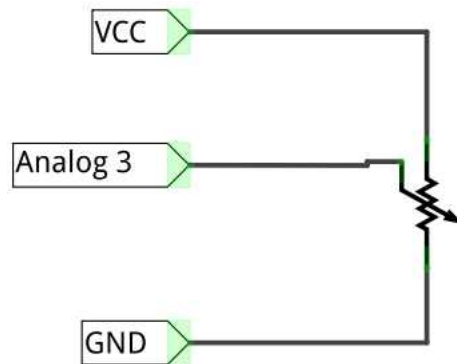


그림 8. 가변 저항 연결 회로도

가변저항의 값은 0에서 1023 사이의 값으로 나온다. 이 값을 0에서 9 사이의 값으로 선형 변환하여 7 세그먼트 표시장치에 나타내도록 스케치를 작성해 보자.

스케치 6. 가변저항으로 7 세그먼트 표시장치 제어

```
// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int pins[] = {5, 6, 7, 8, 10, 11, 12, 13}; // 7 세그먼트 모듈 연결 핀
int variableR = A3; // 가변저항 연결 핀

void setup(){
    for(int i = 0; i < 8; i++){ // 7 세그먼트 모듈 연결 핀을 출력으로 설정
        pinMode(pins[i], OUTPUT);
    }
}

void loop(){
    int R = analogRead(variableR); // 가변저항 읽기
```

```

int no = map(R, 0, 1023, 0, 9);           // 0에서 9 사이의 숫자로 선형 변환

for(int i = 0; i < 8; i++){
    boolean on_off = !bitRead(patterns[no], i);           // 반전 출력
    digitalWrite(pins[i], on_off);
}
}

```

스케치 6을 실행시키고 가변저항을 돌리면 7 세그먼트 표시장치의 값이 변하는 것을 확인할 수 있다.

6. 숫자 맞추기 게임

랜덤 함수를 이용하여 0에서 9 사이의 값을 생성하고 이를 시리얼 모니터로 보여준다. 7 세그먼트 표시장치에는 0.1초 간격으로 숫자가 바뀌도록 한 후 시리얼 모니터에 표시된 숫자가 표시되는 순간에 디지털 9번 핀에 연결된 버튼을 눌러 시리얼 모니터에 표시된 숫자와 일치하는지를 검사하는 게임을 만들어 보자.

스케치 7. 숫자 맞추기 게임

```

// 0에서 9까지 숫자 표현을 위한
// 세그먼트 a, b, c, d, e, f, g, dp의 패턴
byte patterns[] = {
    0xFC, 0x60, 0xDA, 0xF2, 0x66, 0xB6, 0xBE, 0xE4, 0xFE, 0xE6
};

int pins[] = {5, 6, 7, 8, 10, 11, 12, 13}; // 7 세그먼트 모듈 연결 핀
int button_pin = 9;                        // 풀다운 버튼 연결
int index;                                // 현재 출력 숫자

unsigned long time_previous, time_current;

int INTERVAL = 100;                       // 숫자 전환 시간 (ms)

```

```
void setup(){
    for(int i = 0; i < 8; i++){          // 7 세그먼트 모듈 연결 핀을 출력으로 설정
        pinMode(pins[i], OUTPUT);
    }
    pinMode(button_pin, INPUT);          // 버튼 연결 핀을 입력으로 설정

    index = 0;                          // 현재 출력 숫자
    Serial.begin(9600);
    randomSeed(analogRead(A5));          // 난수 생성기 초기화
}

void loop(){
    time_previous = millis();
    int number = random(0, 10);           // 난수 생성
    Serial.println();
    Serial.println("Hit " + String(number) + " !!"); // 생성된 난수 표시

    while(true){
        boolean button = digitalRead(button_pin);
        if(button){                      // 버튼을 누른 경우 난수와 현재 표시 숫자 비교
            delay(500);                  // 디바운스
            break;
        }

        time_current = millis();
        if(time_current - time_previous >= INTERVAL){
            time_previous = time_current;

            index = (index + 1) % 10;

            for(int i = 0; i < 8; i++){
                boolean on_off = !bitRead(patterns[index], i); // 반전 출력
                digitalWrite(pins[i], on_off);
            }
        }
    }
}
```

```
}

Serial.println("You pressed when " + String(index) + "...");
if(number == index){                                // 생성된 난수와 현재 표시 숫자가 일치
    Serial.println("Okay ! Good job !!");
}
else{
    Serial.println("Do some more practice !");
}

Serial.println("Press button to start again...");
while(!digitalRead(button_pin));                    // 다시 시도하기 위해 버튼 입력 대기
delay(500);                                           // 디바운스
}
```

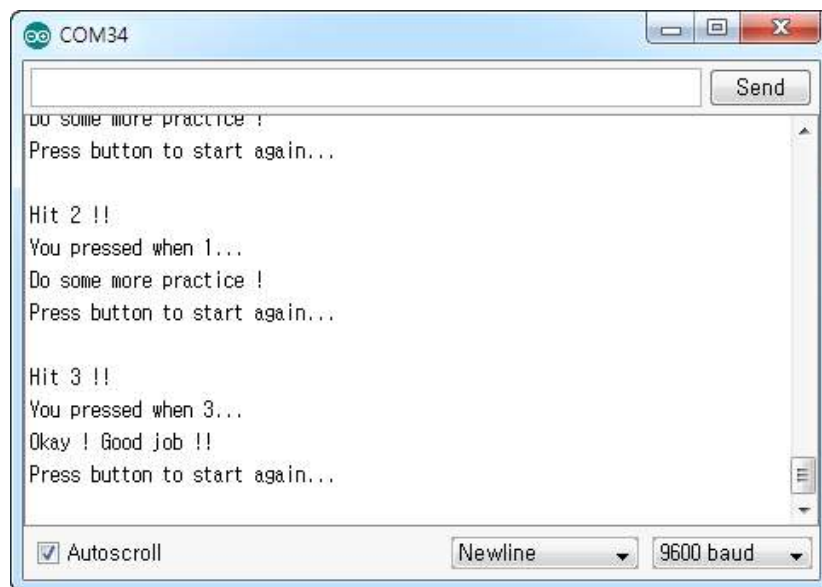


그림 9. 스케치 7 실행 결과