컴퓨터 구조 이론 및 실습 보고서

<아두이노 프로그램 응용>

동의대학교
DONG-EUI UNIVERSITY

학과 : 컴퓨터소프트웨어 공학과
과목 : 컴퓨터 구조와 이론
담당 교수님 : 김성우 교수님
학번 : 20212979
이름 : 임진호

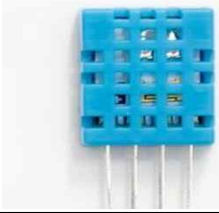
목 차

Ⅰ. 준비사항

1. 센서 (조도센서, 온습도 센서)
센서 : 빛, 소리, 화학물질, 온도 등과 같은 감각과 관련된 신호들을 수집하고 과학적인 방법으로 분석하여 외부의 상태를 알아내는 (종류:광센서, 유동센서, 압력센서, 온도센서 등)

- CDS조도 센서
저렴한 가격과 활용도 때문에 많이 사용, CDS 센서는 광에 쏘여지면 저항 값이 감소하는 광전도 효과를 이용한 반도체 포토 센서이다. CDS라고 불리는 이유는 CDS Photoresistor를 만들어주는 주 재료가 카드늄 CD와 황 S의 화합물인 황화카드뮴 CDS이기 때문이다.
주위가 밝으면 저항이 줄어들고 어두우면 저항이 커진다.
CDS 센서는 광에 쏘여지면 저항 값이 감소하는 광전도 효과를 이용한 반도체 포토 센서이다. 주위가 밝아지면 저항이 줄어들어 Analog Input 핀에 높은 전압이 걸리고, 주위가 어두어지면 저항이 커져서 Analog Input 핀에 낮은 전압이 걸린다

-DHT11 온습도 센서
LM35 온도 센서는 온도를 정밀하게 측정할 수 있는 아날로그 출력 기반의 반도체 온도 센서입니다. 이 센서는 높은 정확도와 사용 편리성으로 인해 온도 모니터링이 필요한 다양한 응용 분야에서 널리 사용됩니다. LM35는 주로 아두이노와 같은 마이크로컨트롤러와 함께 사용되며, 섭씨 온도를 아날로그 전압값으로 출력하는 특성을 가지고 있다. 디지털 펄스 값으로 출력된다.

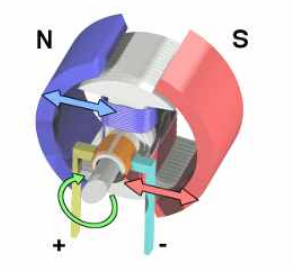
CDS조도 센서	DHT11 온습도 센서
	

2. 모터(DC 모터, 서보 모터, 스텝 모터)
모터 : 전기를 사용하여 회전력을 얻어 바퀴 등을 움직이도록 하는 장치
동작 원리 : 전원이 인가되면 브러시에 연결된 코일을 통해 자기장을 발생시켜 자석의 자기장과 서로 회전력을 만들어 코일을 회전시킨다.

-DC 모터
DC모터는 직류 전원으로 구동되는 모토이며 브러쉬 여부에 따라 분류가 된다 첫 번째로 브러쉬 DC모터는 브러쉬 마찰에 의해 에너지 손실, 제어가 간단, 가장 널리사용되는 종류이고
브러쉬 없는 DC모터는 브러쉬가 없으므로 반영구적, 제어가 복잡하고, 드론에 사용된다.
DC 모터는 전기적 에너지를 역학적 에너지로 바꾸는 장치이며, 공급 전압의 크기와 세기에 따라 속도를 제어할 수 있다.

-서보 모터
일반 DC 모터에 피드백 회로를 추가하여 모터의 위치를 제어하고 일반 DC모터에 비해 구조가 복잡하여 비싸지만 모터 드라이브가 내장되어 있어 구성이 간단하고 정밀 제어가 가능하다

-스텝모터
스텝모터는 펄스 형태의 입력전류에 의해 구동되는 모터이다. 입력되는 전원 펄스의 수에 따라 정해진 각도만큼 회전한다. 또 펄스 속도를 빠르게 입력하면 스테핑 모터의 회전속도가 빨라진다. 펄스 수로 회전량, 펄스 주파수로 회전속도가 정해진다. 모터의 구조가 간단해서 유지관리가 간단하다는 특징이 있다.



3. 적외선 통신
적외선 신호를 통해 통신하는 방식 -> 적외선 LED의 점멸 패턴에 따라 코드를 전송
IrDA표준으로 제정, 무선 리모콘 등으로 널리 사용한다
특징으로는 빛을 매개체로 하므로 전파규제가 없고 높은 대역폭을 획득.
빛의 직진성 때문에 송신기/수신기가 마주보고 있어야 하며 주변빛의 영향을 받는다.
통신 거리가 수 미터로 짧다, 적외선 통신은 송신부와 수신부 2가지로 나누어지고 송신부에서 적외선을 발산하면 수신부에서 포토다이오드를 이용해 적외선을 수신하는 방식으로 통신이 이루어진다.

적외선 리모콘

4. 직렬통신 방식(I2C, SPI)
직렬통신 방식은 한번에 한 비트씩 전송하는 통신방식이다. 병렬 통신에 비해 속도는 느리지만 필요한 선의 수가 적고 다양한 프로세서에서 직렬통신 방식을 선호한다 I2C, SPI, UART 등 다양한 방식이 있다 직렬 통신 방식은 동기식과 비동기 식이 있다. 동기식은 데이터 선 외에 동기를 위한 신호(주로 클럭)을 따로 보낸다 대표적으로 I2C, SPI 등이 있다 비동기식은 동기 신호를 따로 보내지 않는 특징이 있다.

I2C 통신

I2C(Inter-Integrated Circuit)는 두 개의 신호선(SDA, SCL)으로 데이터를 주고받는 직렬 통신 프로토콜입니다.

특징: 마스터-슬레이브 구조, 다중 장치 연결 가능, 저속(100~400kbps) 통신.

장점: 간단한 배선, 소형 디바이스에 적합.

사용 예: 온도 센서, RTC, LCD 디스플레이 등.

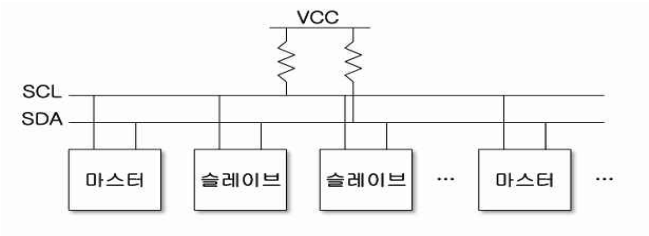
SPI 통신

SPI(Serial Peripheral Interface)는 4개의 신호선(MOSI, MISO, SCLK, CS)을 사용하는 고속 직렬 통신 프로토콜입니다.

특징: 마스터-슬레이브 구조, 고속 통신 지원(수 Mbps 이상).

장점: 빠른 속도, 안정적 데이터 전송.

사용 예: ADC/DAC, 메모리 칩, 센서, 디스플레이 모듈



5. RFID

RFID는 주파수를 이용하여 ID를 식별하는 방식이다. RFID는 전파를 사용하여 서로 통신하는태그와 리더로 구성되어 있다. 태그는 안테나에 부착된 직접회로로 이루어져있다.

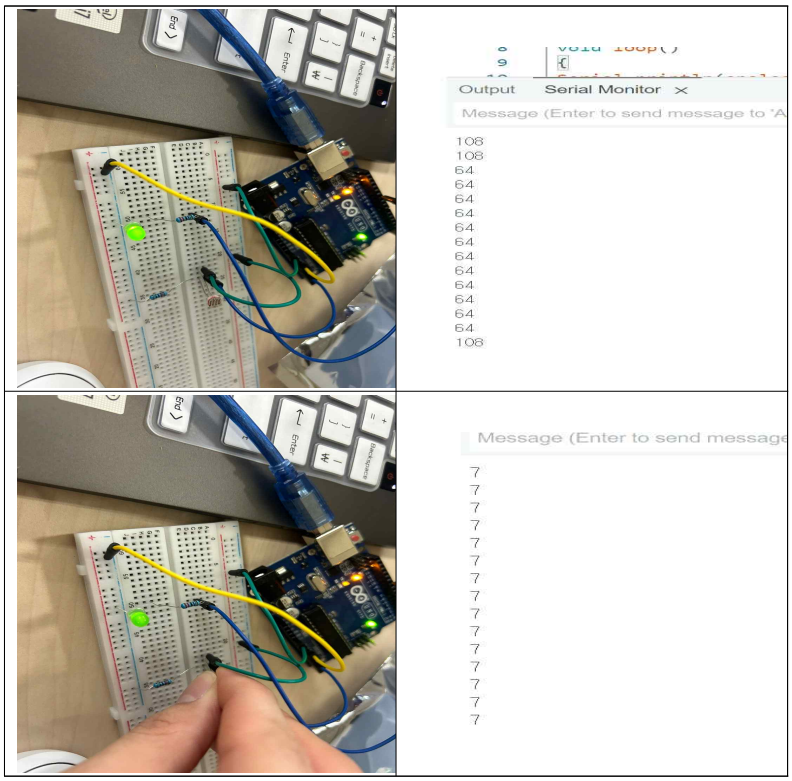
작동원리는 태그가 리더의 범위에 들어오면 ID정보를 전송하여 인식이 이루어지고 주로 유기견 식별, 도서관 책관리, 마트나 창고 물품관리등에 사용된다

장점으로는 반영구적으로 사용 가능하고 비접촉으로 빠르고 정확한 인식 높은 데이터 신뢰성 및 저장·변환 용이, 다수의 태그 정보를 동시에 인식 가능.

단점은 비교적 비용이 높고, 프라이버시 침해우려, 국가별 주파수 차이로 호환성문제, 전파의 범위 한정으로 적용 거리 제약 등의 문제점이 있다.

Ⅱ. 실습

1.(실습1번) 조도 센서를 이용하여 LED의 밝기를 조절



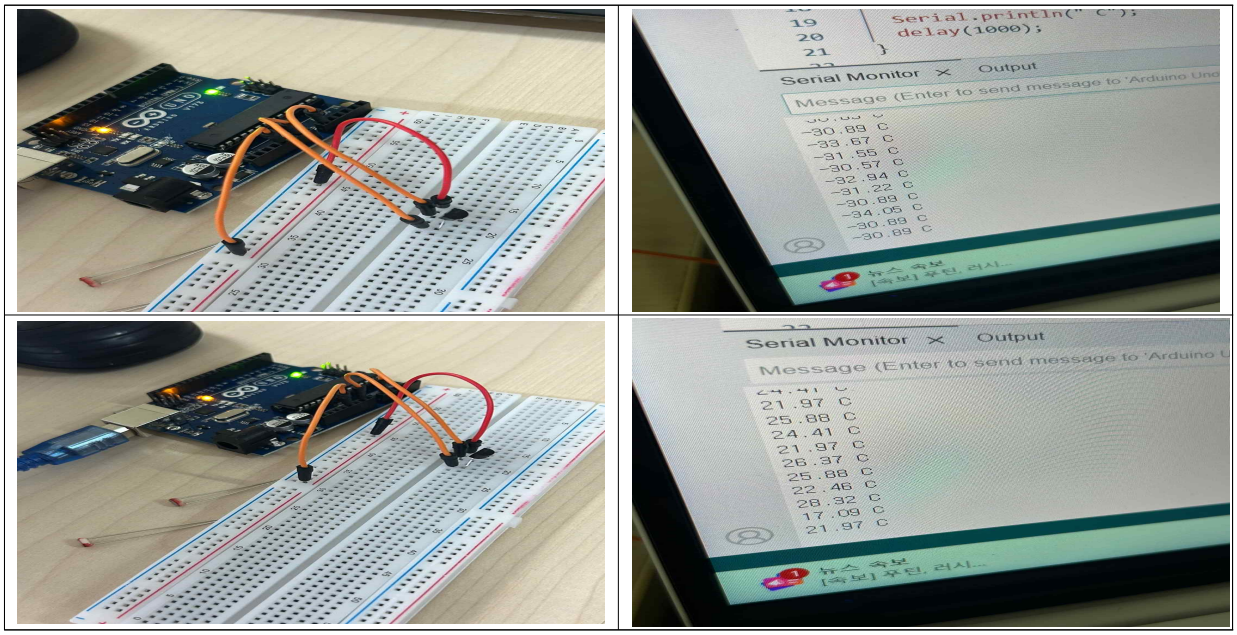
위의 사진이 밝을 때, 아래 사진이 어두울 때

<코드>

<코드>

```
int lightPin = 0; // define a pin for Photo resistor
int ledPin=11; // define a pin for LED

void setup()
{
    Serial.begin(9600); //Begin serial communcation
    pinMode( ledPin, OUTPUT );
}

void loop()
{
    Serial.println(analogRead(lightPin));
    analogWrite(ledPin, analogRead(lightPin)/2);
    delay(10); //short delay for faster response to light.
}
```

2.(실습2) LM35 온도 센서를 이용한 온도 값 측정 회로도



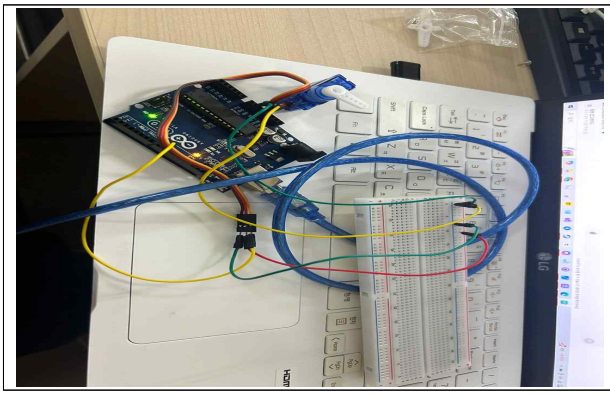
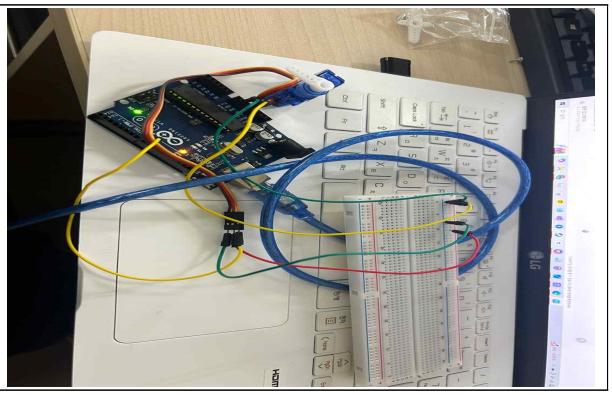
위에서부터, 2-1과 2-2의 실습 결과이다.

<코드>

```
int tempPin = 0; // define a pin for temperature sensor

double lm35(int RawADC) {
  double Temp;
  Temp = 5.0 * RawADC * 100 / 1024;
  return Temp;
}

void setup()
{
    Serial.begin(9600); //Begin serial communication
}

void loop()
{
    double temp = lm35(analogRead(tempPin));
    Serial.print(temp);
    Serial.println(" C");
    delay(1000);
}
```

```
int tempPin = 0; // define a pin for temperature sensor

double Thermistor(int RawADC) {
  double Temp;
  // See http://en.wikipedia.org/wiki/Thermistor for explanation of formula
  Temp = log(((10240000/RawADC) - 10000));
  Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp * Temp * Temp));
  Temp = Temp - 273.15;          // Convert Kelvin to Celcius
  return Temp;
}

void setup()
{
    Serial.begin(9600); //Begin serial communication
}

void loop()
{
    double temp = Thermistor(analogRead(tempPin));
    Serial.print(temp);
    Serial.println(" C");
    delay(1000);
}
```

3-1.(실습4번) 서보모터 제어

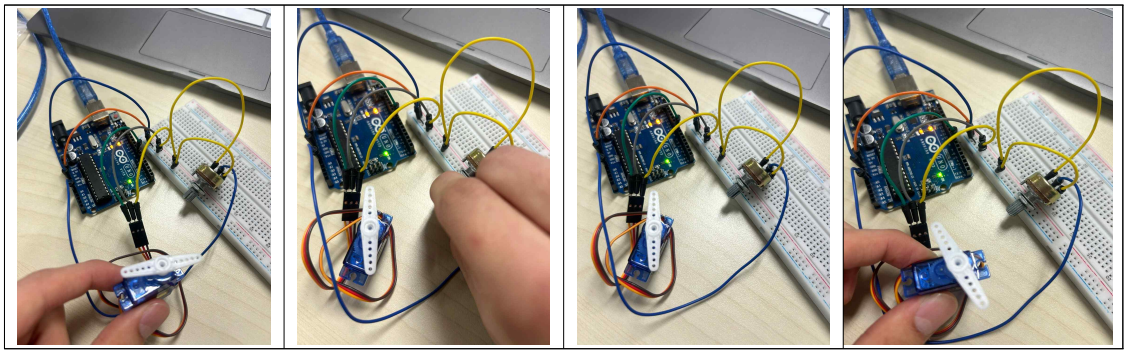



서버모터가 천천히 돌아가는 것을 볼 수 있다

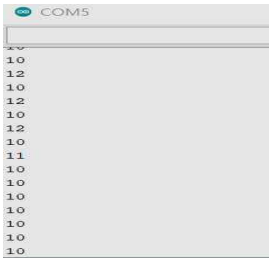
<코드>

```
#include <Servo.h> // 모터 사용 위해 헤더 선언
int motor_control = 8; // 모터 포트 번호
Servo servo;

void setup() {
servo.attach(motor_control);
}
void loop() {
int i;
servo.write(0); // 모터 초기화
delay(1000);
for (i = 0;i<90;i+=10) { // 모터 각도
servo.write(i); // 모터 작동
delay(20);
}
delay(1000);
}
```

3-2.(실습4) 가변저항과 서보모터 연결



가변저항 적을때 가변저항 클때

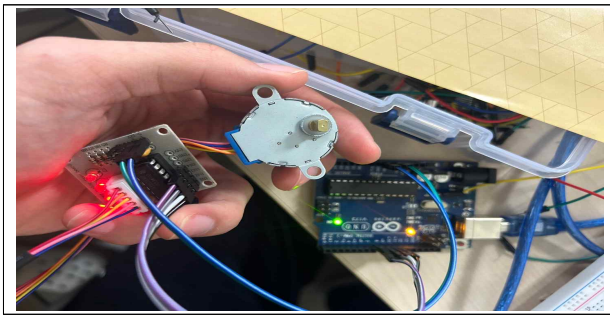
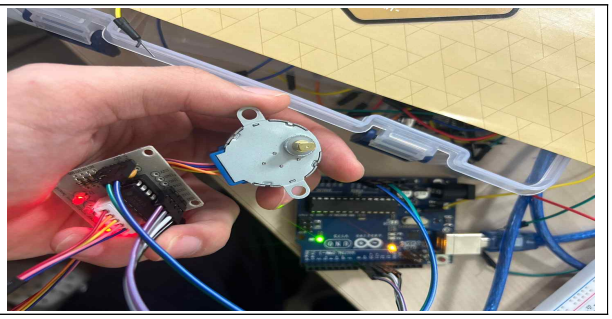


<코드>

```
#include <Servo.h>
Servo myServo;              // 서보 모터 객체 생성
const int servoPin = 8;   // 서보 모터 신호 핀
const int potPin = A0;    // 가변 저항 핀

void setup() {
  myServo.attach(servoPin); // 서보 모터 핀 설정
  Serial.begin(9600);         // 시리얼 통신 시작 (디버깅용)
}
void loop() {
  // 가변 저항 값 읽기 (0~1023)
  int potValue = analogRead(potPin);

  // 가변 저항 값을 서보 각도 값(0~180도)으로 변환
  int angle = map(potValue, 0, 1023, 0, 180);

  // 서보 모터에 각도 값 전송
  myServo.write(angle);

  // 디버깅용으로 시리얼 출력
```

```
  Serial.print("Potentiometer Value: ");
  Serial.print(potValue);
  Serial.print(" | Servo Angle: ");
  Serial.println(angle);
  delay(15); // 서보 모터 안정화 딜레이
}
```

4. (실습5번) 스테핑 모터제어 및 모터드라이버 연결

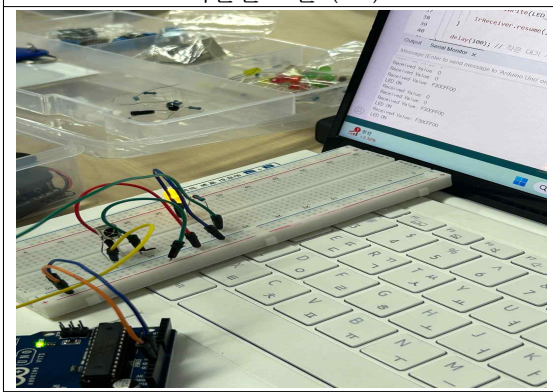
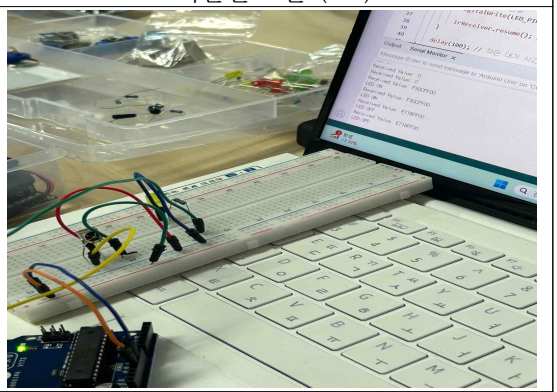
진동과 함께 모터가 돌아가는 것을 알 수 있다.

<코드>

```
const int motorPin1 = 11; // IN1
const int motorPin2 = 10; // IN2
const int motorPin3 = 9;  // IN3
const int motorPin4 = 8;  // IN4
int delayTime = 10;       // 각 스텝 사이의 지연 시간(ms)

// 한 스텝씩 회전하기 위한 시퀀스 배열
int stepSequence[4][4] = {
  {1, 0, 0, 0}, // Step 1
  {0, 1, 0, 0}, // Step 2
  {0, 0, 1, 0}, // Step 3
  {0, 0, 0, 1}  // Step 4
};

void setup() {
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
}

void loop() {
  // 정방향 회전: 한 바퀴 (200 스텝 기준)
```

```
  for (int i = 0; i < 1000; i++) {
    stepMotor(1); // 정방향으로 한 스텝 이동
    delay(delayTime);
  }

  delay(1000); // 1초 대기

  // 역방향 회전: 한 바퀴 (200 스텝 기준)
  for (int i = 0; i < 1000; i++) {
    stepMotor(-1); // 역방향으로 한 스텝 이동
    delay(delayTime);
  }

  delay(1000); // 1초 대기
}

// 모터를 한 스텝씩 이동시키는 함수
void stepMotor(int direction) {
  static int currentStep = 0; // 현재 스텝 상태 저장

  // 현재 스텝 출력
  digitalWrite(motorPin1, stepSequence[currentStep][0]);
  digitalWrite(motorPin2, stepSequence[currentStep][1]);
  digitalWrite(motorPin3, stepSequence[currentStep][2]);
  digitalWrite(motorPin4, stepSequence[currentStep][3]);

  // 다음 스텝 계산 (정방향: +1, 역방향: -1)
  currentStep = (currentStep + direction + 4) % 4;
}
```

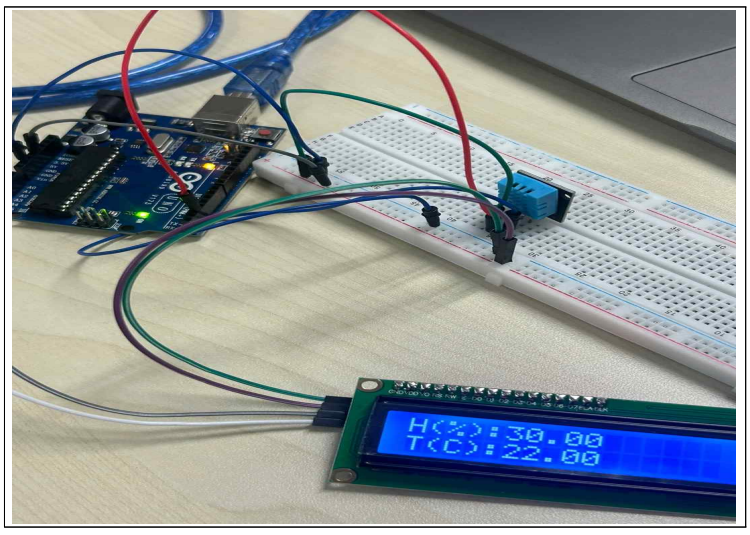
5. (실습7번) 적외선 수신 센서 리몬콘으로 수신

리몬콘 1번 (on)	리몬콘 2번 (off)
	

<코드>

```
#include <IRremote.hpp>

int RECV_PIN = 8; // 핀 번호 설정
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup(){
  irrecv.enableIRIn();
  pinMode(9,OUTPUT); // 출력 설정
  Serial.begin(9600); // 시리얼 모니터 설정
}
void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
      if(results.value==0xFF6897){ // 리모컨 0번 누르면 실행
        digitalWrite(9,HIGH); // LED 불 켜짐
      }
  else{
    digitalWrite(9,LOW); // LED 불 꺼짐
  }
  irrecv.resume();
  delay(100); //0.1초 정지
  }
}
```

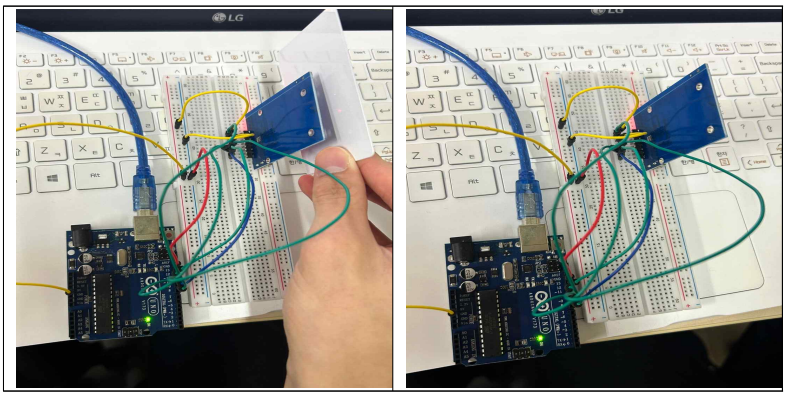
6. (실습8번) DHT11 온습도 센서 결과 LCD출력



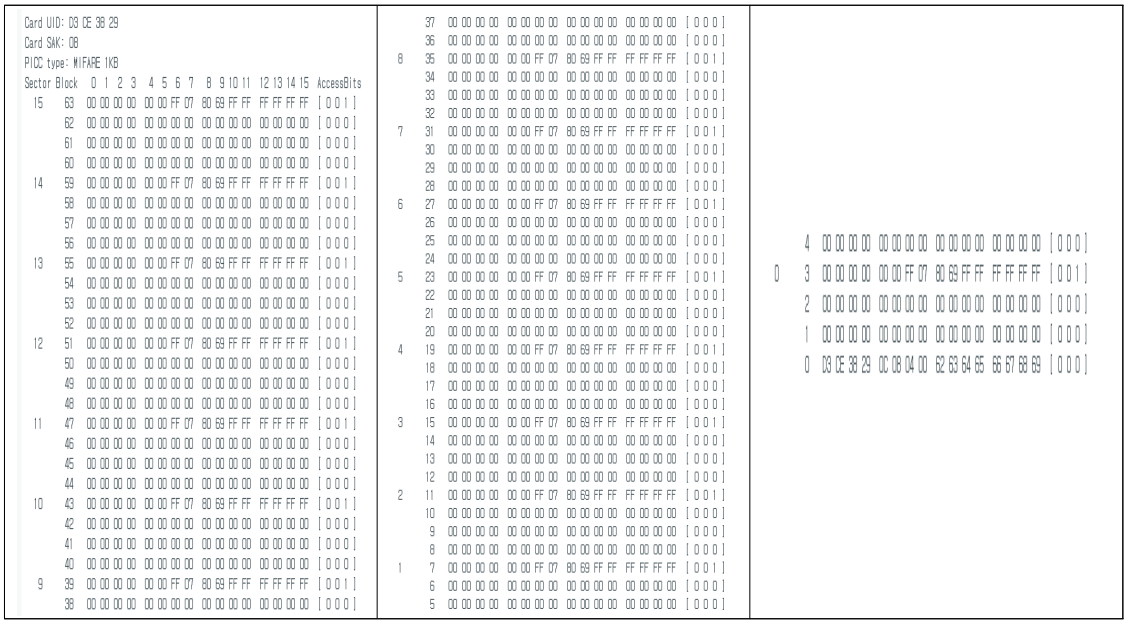
<코드>

```
#include <DHT.h> // 온습도 센서 사용 위해 헤더 선언
#include <DHT_U.h>
#include <LiquidCrystal_I2C.h> // LCD 사용 위해 헤더 선언
#include <Wire.h>
#define DHTPIN 2 // 온습도 센서 포트 핀 번호
#define DHTTYPE DHT11
LiquidCrystal_I2C lcd(0x27,16,2); // lcd 객체
DHT dht (DHTPIN,DHTTYPE); // dht 객체

void setup(){
  Serial.begin(9600); // 시리얼 모니터 설정
  lcd.init();
  lcd.backlight(); // lcd 백라이트 켬
  lcd.begin(16,2); // lcd 사용 및 크기 설정
  dht.begin();
}
void loop(){
  float h = dht.readHumidity(); // 습도 읽기.
  float t = dht.readTemperature(); // 온도 읽기.
  lcd.display(); // lcd 내용 표시
  lcd.setCursor(0,0); // 커서 0,0 위치
  lcd.print("H(%):");
  lcd.print(h); // 습도 출력
  lcd.setCursor(0,1); // 커서 0,1 위치
  lcd.print("T(C):");
  lcd.print(t); // 온도 출력
  delay(300); //0.3초 정지
}
```

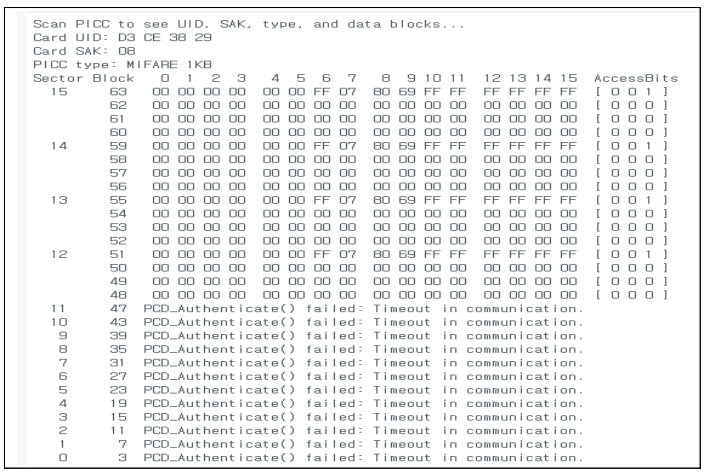
7. (실습9번) RFID 기판을 이용해 RFID 태그 인식



<카드를 인식할때>

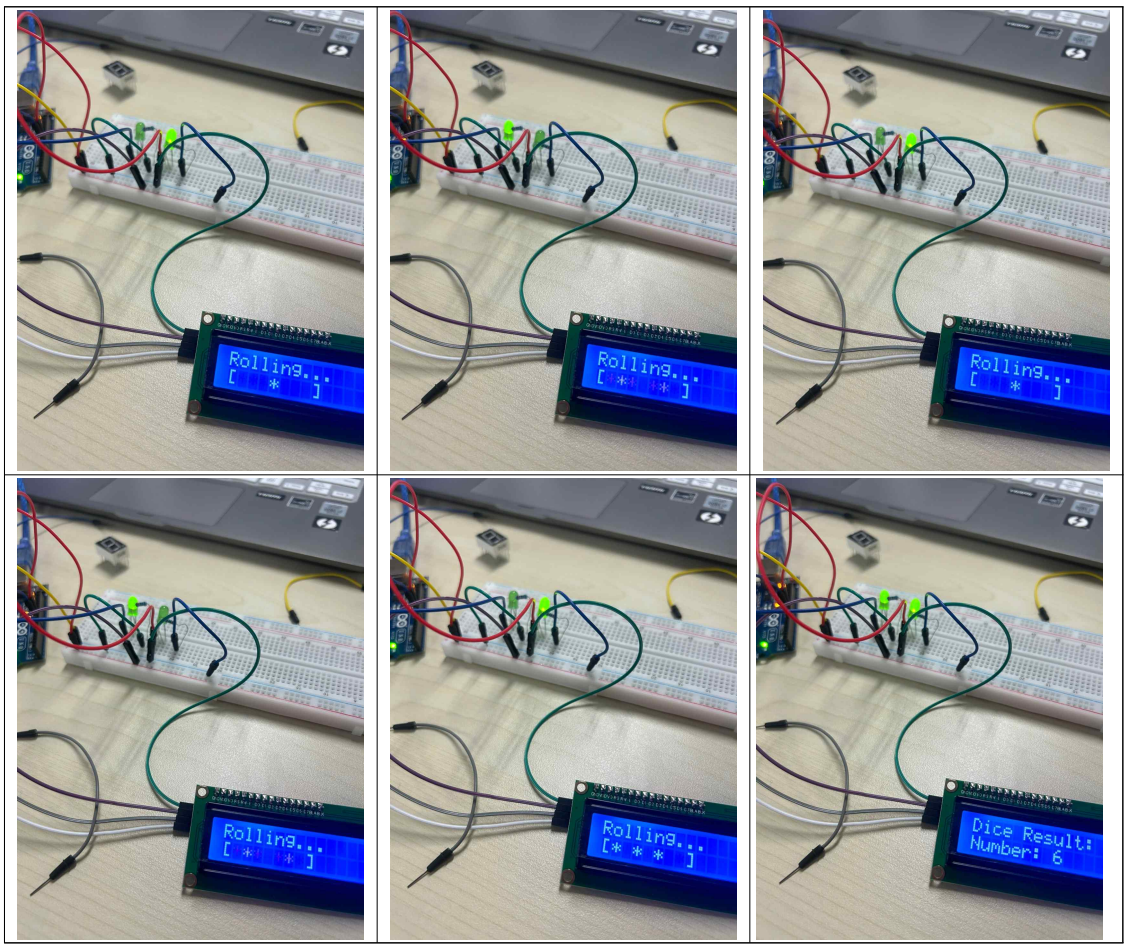


<카드를 뺐을때>



8. 개인프로젝트

(랜덤 주사위 프로젝트)



<코드>

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

// LCD 설정
LiquidCrystal_I2C lcd(0x27, 16, 2);

// LED 핀 번호 설정
int led1Pin = 8; // 첫 번째 LED 핀
int led2Pin = 9; // 두 번째 LED 핀

// 함수 선언 (프로토타입)
void displayDice(int number, bool isAnimating, int patternIndex = 0);

void setup() {
  // LCD 초기화
  lcd.init();
```

```
  lcd.backlight(); // LCD 백라이트 켬
  lcd.begin(16, 2); // LCD 초기화

  // LED 핀 초기화
  pinMode(led1Pin, OUTPUT);
  pinMode(led2Pin, OUTPUT);

  // 시작 메시지
  randomSeed(analogRead(0)); // 랜덤 시드 초기화
  lcd.setCursor(0, 0);
  lcd.print("Dice Simulator");
  delay(2000); // 시작 메시지 표시
  lcd.clear();
}

void loop() {
  // 주사위 굴리기 애니메이션
  for (int i = 0; i < 10; i++) { // 애니메이션 반복 횟수
    int animValue = random(1, 7); // 1부터 6까지 랜덤값
    displayDice(animValue, true, i % 4); // 애니메이션 눈 및 별 패턴 설정

    // LED 깜빡임 (번갈아 켜기)
    if (i % 2 == 0) {
      digitalWrite(led1Pin, HIGH);
      digitalWrite(led2Pin, LOW);
    } else {
      digitalWrite(led1Pin, LOW);
      digitalWrite(led2Pin, HIGH);
    }

    delay(200); // 애니메이션 및 LED 깜빡임 속도
  }

  // 최종 주사위 값
  int diceValue = random(1, 7); // 1부터 6까지 랜덤값
  displayDice(diceValue, false); // 최종 값 표시

  // LED 2개에 불 켜기
  digitalWrite(led1Pin, HIGH);
  digitalWrite(led2Pin, HIGH);

  delay(5000); // 5초 대기

  // LED 끄기
  digitalWrite(led1Pin, LOW);
  digitalWrite(led2Pin, LOW);
}

// 주사위 눈 또는 숫자 표시 함수
void displayDice(int number, bool isAnimating, int patternIndex) {
  lcd.clear();
```

```
  if (isAnimating) {
    lcd.setCursor(0, 0);
    lcd.print("Rolling...");

    // 별 패턴 반복
    lcd.setCursor(0, 1);
    switch (patternIndex) {
      case 0:
        lcd.print("[* * * *]");
        break;
      case 1:
        lcd.print("[* * *  ]");
        break;
      case 2:
        lcd.print("[ *   * ]");
        break;
      case 3:
        lcd.print("[   *   ]");
        break;
    }
  } else {
    lcd.setCursor(0, 0);
    lcd.print("Dice Result:");
    lcd.setCursor(0, 1);
    lcd.print("Number: ");
    lcd.print(number); // 주사위 숫자 출력
  }
}
```

Ⅲ. 실습 결과, 검토 및 요점

1. 실습1 조도센서를 이용해 LED의 밝기를 조절

센서와 LED의 핀을 설정하고 LED를 출력으로 설정하였다. 센서의 밝기를 모니터에 출력하여 상태를 확인할 수 있다. 이 센서는 밝기센서로 밝을 때 값이 크게 나타나고 LED가 밝게 빛나는 것을 확인할 수 있다. 그리고 손으로 센서를 가려 어둡게 만들면 모니터에 출력되는 값이 작게 나타나고 LDE또한 약하게 빛나는 것을 확인할 수 있다.

2. 실습2　LM35 온도 센서를 이용한 온도 값 측정 회로도

온도 센서의 핀을 설정하고 온도의 결과를 확인할 시리얼 모니터를 설정하였다. 측정한 온도를 계산하고 계산된 값을 읽어 시리얼 모니터에 출력되는 것을 확인할 수 있다.

3. 실습 4　서보모터 제어 (아두이노-서보모터, 브레이드보드- 가변저항)

실습4번은 2개의 실습으로 진행하였다 하나는 실습1에 있는 서버모터만 실습하였고 2번은 가변저항과 연결하여 진행하였다. 브레드 보드에 가변 저항을 연결해 서보모터를 조절할 때는 가변 저항의 저항값을 이용해 조절하였다. 즉, 가변저항을 돌리게 되면 서버모터가 같이 돌아가는 상황이다. 가변 저항에 연결된 5V의 값이 1023으로 굉장히 높기 때문에 map함수를 이용해 10~170까지로 조절해 모터의 각도가 변할 수 도록 작성하였다. 또한 #include <Servo.h> 라이브러리를 다운하였다

4. 실습5 스테핑 모터 제어

스테핑 모터 드라이버 핀 번호를 설정하고 출력을 설정하여주었다. 그리고 반복문을 이용해 모터가 계속 진동하며 돌아가는 것을 확인할 수 있다.

5. 실습7 적외선 수신 센서 리모콘

<IRremote.hpp>를 이용해 아두이노 리모콘을 사용할 수 있도록 하였다.
led를 출력으로 설정해주고 리몬콘 1번을 누르면 LED가 ON 2번을 누르면 LED가 OFF가 되어 LED가 켜지고 꺼지는 실습을 진행하였다.

6. 실습8 DHT11 온습도 센서 LCD출력

온습도 센서를 사용하기 위해 해당 라이브러리인 DHT sensor library를 다운받아 사용하였다. 그리고 온습도 센서의 핀 번호를 지정해주고 객체를 생성하여

readHumidity()함수로 습도를 읽고 readTemperature()함수를 이용하여 온도를 읽어 LCD의 화면에 그 온도와 습도를 출력하게 실습을 진행하였다.

7. 실습9 RFID 기판을 이용해 RFID 태그 인식

RFID 기판을 이용하여 태그가 인식되고 그 값이 시리얼 모니터에 출력되어 확인할 수 있는 프로그램을 만들었다. 카드를 계속 대고있으면 실습 맨 윗 사진처럼 정상적으로 출력이 되지만 카드를 중간에 빼게된다면 PCD_Authenticate() 와 같은 값이 뜨는 것을 확인할 수 있다

8. 실습10 자유주제(개인프로젝트) 주사위 게임

개인프로젝트로 주사위게임을 만들었습니다 LED와 LCD를 이용하여 1~6의 숫자가 랜덤으로 나오는 실습입니다 주사위가 돌아가는 시간동안은 LCD에 [* *] 형식으로 대괄호 안에 *가 반복되어 왔다 갔다 하면서 주사위가 돌아가고 있음을 표현하였고 브레드보드에도 LED 2개를 설치하여 2개가 각각따로 반복하여 불이 들어오고 꺼지게 만들어 주사위가 돌아가고 있음을 더욱 강조하고 결과가 나올 때 2개의 LED가 모드 켜진 상태로 다음 주사위가 돌아갈 때 까지 기다립니다.

결과가 나오면 Dice Result와 함께 1~6중 하나의 숫자가 출력된다.

추가로 delay를 이용하여 결과가 나온 후 5초동안 멈추게 만들었습니다.

느낀점

이번 실습을 하면서 실습9번과 같은 버스에서 카드를 찍을 때 내부에서 어떻게 동작하는지를 알 수 있었고 그것을 실제로 만들 수 있어 좋았습니다. 또한 지금까지의 Lab보고서에는 없었던 개인프로젝트를 통해 나만의 특별한 실습을 통해 색다른 경험을 할 수 있게 되었습니다.