타이머와 바람 세기 조절 기능을 갖는 선풍기의 구현

19010671 김택천

10cheon00@sju.ac.kr

1. 목표

아두이노의 DC 모터와 가변 저항을 이용하여 바람 세기를 조절할 수 있는 선풍기를 구현하고, 서보 모터를 이용하여 타이머를 나타내도록 구현한다. 그리고 타이머를 설정하기 위한 버튼과, 선풍기의 전원을 On/Off 할 수 있는 버튼을 구현한다.

2. 구현 과정

2-1. 전원 버튼과 타이머를 나타내는 서보 모터 회로 구현

전원 버튼을 눌렀을 때 전원 LED가 점등되면서 타이머를 나타내는 서보 모터가 작동하도록 한다. 그리고 타이머 버튼으로 타이머 값을 조절하도록 한다.

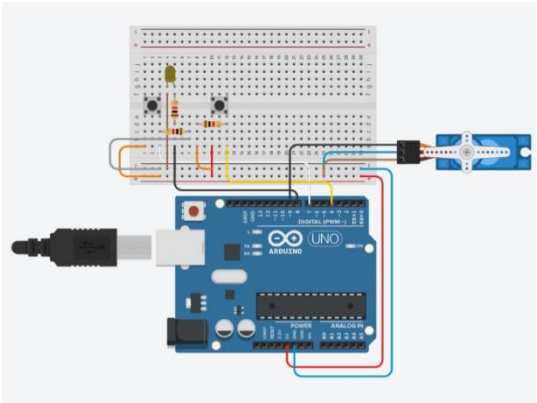

그림 1. 전원 LED 및 버튼, 타이머 버튼과 서보 모터 회로 구성

그림 1은 전원 버튼과 타이머 버튼 그리고 타이머를 나타내는 서보 모터 회로를 구현한 모습이다. 7번 핀에 연결된 전원 버튼을 누르게 된다면 8번 핀과 연결된 LED가 점등되어야 한다. 첨부 #1의 실행 코드에서 매 루프마다 `updatePowerBtnState()`함수를 통해 전원 버튼의 상태를 갱신한다. 전원 버튼을 눌렀을 때 `isPowerBtnDown()`함수를 통해 전원 버튼을 처음 누른 상태를 확인하여 전원을 켜도록 프로그래밍하였다. 이어서 매 루프마다 `isPowerOn()` 함수를 통해 전원이 켜 졌는지 확인하고, 만약 전원이 켜져 있다면 `turnOnPowerLED()` 함수를 통해 전원 LED를 점등한다. 전원이 점등된 후에는 타이머가 작동해야 하므로 `decreaseTimer()` 함수와 `delay(100)`을 통해 0.1초마다 타이머 값을 변경한다. 변경된 타이머는 매 루프마다 `updateTimerServo()` 함수를 통해 9번 핀에 연결된 서보 모터에 반영되어 타이머를 시각적으로 나타낸다.

아두이노 키트에 제공된 서보 모터는 0°부터 180°까지 조절이 가능하다. 그러므로 0°를 0초, 180°

를 타이머의 최대 시간으로 설정한다. 타이머의 최대 시간은 첨부 #1의 실행 코드에서 `const int TIMER_MAX = 10;`에 의해 10초로 설정되어 있다. 아래의 그림 2는 서보 모터가 나타내는 타이머의 값을 보여준다.

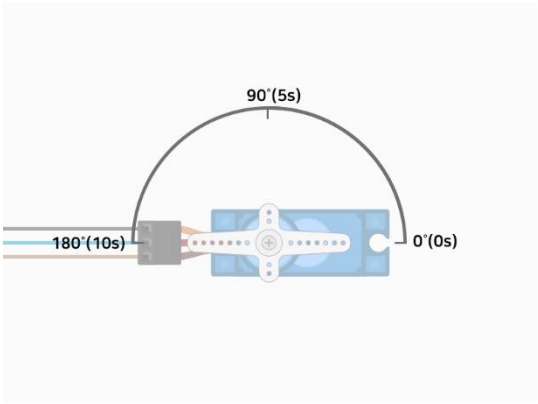

그림 2. 서보 모터의 각도가 나타내는 시간값

만약 타이머가 0이 되었다면 전원을 꺼야 하므로 `isTimerEnd()` 함수를 통해 타이머가 0이 되었는지 검사하고, 만약 그렇다면 `togglePowerState()` 함수를 통해 전원을 끄게 된다.

2-2. DC 모터와 가변 저항 회로 구현

5V 전압을 가하였을 때 DC 모터와 연결된 팬이 회전하도록 구현한다. 이 때 가변 저항을 연결하여 저항을 통해 바람 세기를 조절하도록 구현한다.

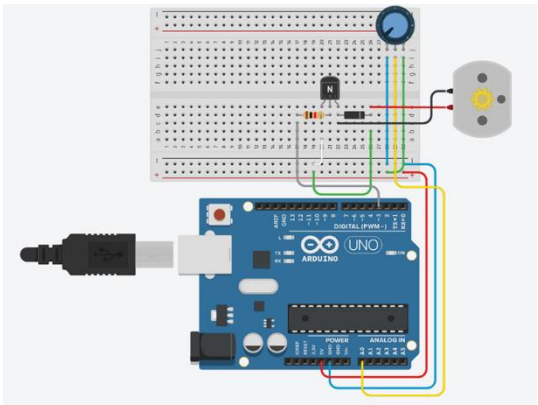

그림 3. 트랜지스터와 가변 저항을 연결한 DC 모터 회로 구성

그림 3은 DC 모터와 트랜지스터, 그리고 가변 저항을 통해 회로를 구현한 모습이다. 트랜지스터 없이 DC 모터와 서보 모터를 사용할 경우 아두이노 보드의 전류가 약하여 원활하게 활용할 수 없다. 따라서 트랜지스터를 통해 전류를 증폭시켜야 한다. 가변 저항을 A0핀에 연결하여 입력 값을 받은 후 3번 핀과 연결된 트랜지스터의 Base로 출력한다.

첨부 #1의 실행 코드에서 매 루프마다 `isPowerOn()`을 검사하여 전원이 켜져 있을 때에만 팬을 회전시키도록 하고 있다. `readDynamicResistance()` 함수를 실행하여 가변 저항 값을 입력 받은

후, `mapDynamicResistanceToMotorSpeed()` 함수를 실행하여 가변 저항 값을 0~255 범위로 매핑하여 속도 값으로 저장하고, `updateMotorSpeed()` 함수를 실행하여 조절된 속도 값을 3번 핀으로 출력하고 있다. 따라서 가변 저항 값이 트랜지스터의 Base로 출력되어 전류의 증폭량을 조절하게 되므로, 결과적으로는 DC 모터의 속도, 즉 바람 세기가 조절되는 셈이다.

타이머의 시간이 0이 될 경우 팬이 회전하면 안되기 때문에, `isPowerOn()` 함수가 false일 때 `setMotorSpeedZero()` 함수를 실행하여 `motorSpeed`의 값을 0으로 설정한 후 트랜지스터의 Base로 출력시킨다. 결과적으로 전원이 꺼진 상황에는 Base에 아무 전류도 흐르지 않게 되었으므로 팬이 회전을 멈추게 된다.

3. 결과

그림 1과 그림 3의 회로를 종합하면 아래의 그림 4와 같이 완성된다.

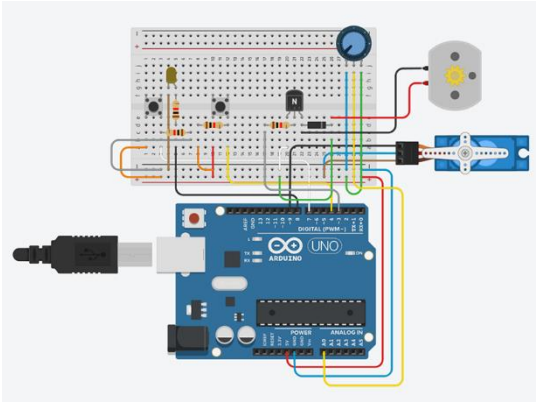

그림 4. 전체 회로 구성

사용자는 선풍기의 작동 시간을 조절하기 위해 타이머 버튼을 눌러 시간을 조절한다. 이 때 서보 모터를 통해 작동 시간을 시각적으로 확인할 수 있다. 사용자가 전원 버튼을 누르게 된다면 LED에 불이 들어오면서 팬이 회전하게 된다. 사용자는 선풍기의 전원이 켜져 있는 동안 가변 저항의 노브를 돌려서 세기를 조절할 수 있다. 작동 시간이 0이 될 경우 선풍기의 전원 LED가 꺼지고 팬이 회전을 멈춘다.

4. 결론

현실의 선풍기와 비슷하게 타이머 설정 기능과 바람 세기 조절 기능을 구현하였다. DC 모터와 가변 저항을 사용하여 선풍기를 구현한 후 서보 모터를 활용하려고 하니 전류가 충분하지 않아 팬이 약하게 돌았다. 이 문제를 해결하기 위해 많은 시간을 소모하였고, 결국 트랜지스터를 활용하여 문제를 해결했다. 테스트를 위해 오랫동안 팬을 회전시켰더니 트랜지스터의 발열이 심하여 화상을 입을 뻔했다. 테스트할 때마다 소자를 망가뜨리지 않게 주의가 필요함을 깨달았다.

첨부 #1 : 실행 코드

```
#include <Servo.h>

const int PIN_D_RSTN = A0;
const int PIN_MOTOR = 3;
const int PIN_BTN_TIMER = 4;
const int PIN_BTN_POWER = 7;
const int PIN_LED_POWER = 8;
const int PIN_SERVO = 9;

Servo timerServo;

const int TIMER_MAX = 10;
const int SERVO_ANGLE_MAX = 180;
const int TIMER_PRESET_COUNT = 3;
const int ARRAY_TIMER_PRESET[TIMER_PRESET_COUNT] = { 0, TIMER_MAX / 2, TIMER_MAX };
double timer = 0.0f;
int timerPresetIndex = 0;

bool power = true;

int powerBtnState = 0;
int prevPowerBtnState = 0;

int timerBtnState = 0;
int prevTimerBtnState = 0;

int dynamicResistance = 0;
int motorSpeed = 0;



void setup() {
```

```
    pinMode(PIN_BTN_TIMER, INPUT);
    pinMode(PIN_BTN_POWER, INPUT);
    pinMode(PIN_LED_POWER, OUTPUT);
    pinMode(PIN_MOTOR, OUTPUT);
    pinMode(PIN_D_RSTN, INPUT);

    timerServo.attach(PIN_SERVO);

#ifdef __DEBUG__
    Serial.begin(9600);
#endif
}

void loop() {
    updatePowerBtnState();
    if (isPowerBtnDown()) {
        togglePowerState();
    }

    updateTimerBtnState();
    if (isTimerBtnDown()) {
        changeTimer();
    }
    updateTimerServo();

    if (isPowerOn()) {
        turnOnPowerLED();

        readDynamicResistance();
        mapDynamicResistanceToMotorSpeed();
        updateMotorSpeed();
```

```
        decreaseTimer();
        if(isTimerEnd()) {
            togglePowerState();
        }

        delay(100);
    } else {
        turnOffPowerLED();
        setMotorSpeedZero();
    }
}

void updatePowerBtnState() {
    prevPowerBtnState = powerBtnState;
    powerBtnState = digitalRead(PIN_BTN_POWER);
}

bool isPowerBtnDown() {
    return !prevPowerBtnState && powerBtnState;
}

void togglePowerState() {
    power = !power;
}

bool isPowerOn() {
    return power;
}

void updateTimerBtnState() {
    prevTimerBtnState = timerBtnState;
    timerBtnState = digitalRead(PIN_BTN_TIMER);
```

```
}

bool isTimerBtnDown() {
    return !prevTimerBtnState && timerBtnState;
}

void changeTimer() {
    timerPresetIndex++;
    if (timerPresetIndex == TIMER_PRESET_COUNT) {
        timerPresetIndex = 0;
    }
    timer = ARRAY_TIMER_PRESET[timerPresetIndex];
}

void turnOnPowerLED() {
    digitalWrite(PIN_LED_POWER, HIGH);
}

void readDynamicResistance() {
    dynamicResistance = analogRead(PIN_D_RSTN);
}

void mapDynamicResistanceToMotorSpeed() {
    motorSpeed = map(dynamicResistance, 0, 1023, 0, 255);
}

void updateMotorSpeed() {
    analogWrite(PIN_MOTOR, motorSpeed);
}

void decreaseTimer() {
    if (timer >= 0.1f) {
```

```
        timer -= 0.1f;
    }
}

void updateTimerServo() {
    timerServo.write((timer / TIMER_MAX) * SERVO_ANGLE_MAX);
}

bool isTimerEnd() {
    return timer < 0.1f;
}

void turnOffPowerLED() {
    digitalWrite(PIN_LED_POWER, LOW);
}

void setMotorSpeedZero() {
    motorSpeed = 0;
    analogWrite(PIN_MOTOR, motorSpeed);
}
```

참고 문헌

[1] [05-1 푸시버튼으로 LED 켜고 끄기], https://wikidocs.net/30758, 2019년 5월 4일 17:02

[2] [07-3 DC모터 제어하기], https://wikidocs.net/30786, 2019년 5월 4일 19:05

[3] [08-2 서보모터(Servo motor) 사용하기], https://wikidocs.net/30882, 2019년 5월 4일 19:20

[4] [Tutorial 13: DC Motor], https://aaravpatel.com/2020/01/03/dc-motor/, January 3 2020

[5] [`servo.attach()` paralyzing (one) motor on L298N], https://arduino.stackexchange.com/a/31811, Dec 2 2016, 0:20