와이어링 파이

컴퓨터 네트워크 설계

청주대학교 전자공학과
한철수

목차

- 와이어링 파이
- 실습

와이어링 파이(Wiring Pi)

- 라즈베리 파이의 GPIO 포트를 제어하기 위해 작성된 GPIO 입출력 라이브러리임.
  - C 언어로 작성됨.
  - GNU LGPL v3.0 라이센스를 가짐.
- C와 C++ 언어에서는 바로 이용이 가능하고, 다른 언어들(Python, PHP, Ruby, Perl, Node 등)에서는 wrapper 형태로 이용 가능함.
- 와이어링 파이는 아두이노의 라이브러리와 매우 유사함.
  - 함수명, 매개 변수, 매크로 상수 등이 대부분 일치함.
  - 아두이노를 사용해 본 적이 있다면 큰 어려움없이 사용이 가능함.
- 와이어링 파이 웹사이트
  - 함수 사용법, 예제 등의 여러 가지 정보를 구할 수 있음.
  - http://wiringpi.com/

와이어링 파이의 주요 함수들 (1/2)

- int wiringPiSetup(void)
  - GPIO 포트를 초기화함.
- void pinMode(int pin, int mode)
  - pin을 입력 또는 출력으로 설정함.
- void digitalWrite(int pin, int value)
  - pin에 LOW 또는 HIGH를 출력함.
- int digitalRead(int pin)
  - pin의 상태 값(LOW 또는 HIGH)을 읽음.
- void analogWrite(int pin, int value)
  - pin에 유사 아날로그 값을 출력함.
- int analogRead(int pin)
  - pin의 아날로그 값을 읽음.
  - 라즈베리 파이에는 ADC가 내장되어 있지 않기 때문에, 별도의 ADC IC를 연결한 경우에만 함수가 정상적으로 동작함.

와이어링 파이의 주요 함수들 (2/2)

- void pwmWrite(int pin, int value)
  - pin에 PWM 신호를 출력함.
- void delay(unsigned int howLong)
  - howLong ms만큼 동작을 쉼.
- void delayMicroseconds(unsigned int howLong)
  - howLong us만큼 동작을 쉼.
- unsigned int millis(void)
  - 프로그램이 동작을 시작한 후부터 현재까지 얼마의 시간이 흘렀는지를 ms 단위로 반환함.
- unsigned int micros(void)
  - 프로그램이 동작을 시작한 후부터 현재까지 얼마의 시간이 흘렀는지를 us 단위로 반환함.

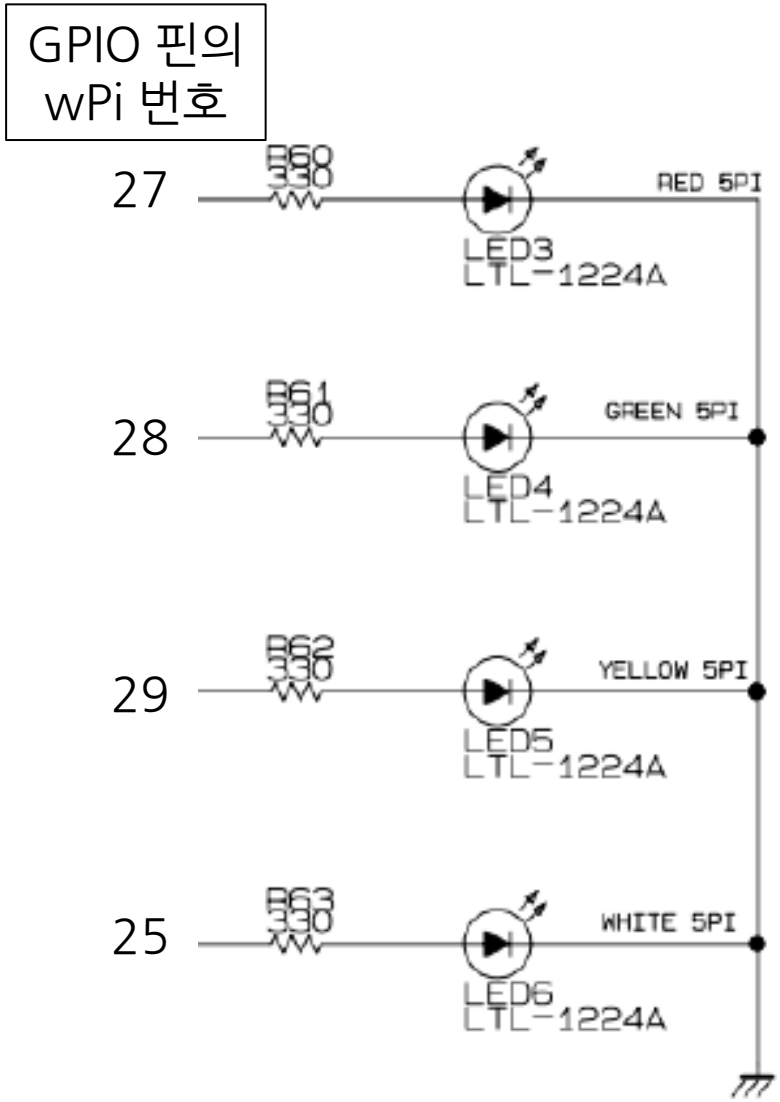
GPIO 포트를 제어하는 소스 코드의 작성 절차

1. 사용할 라이브러리를 추가함.
   - 예
     - #include <wiringPi.h> // GPIO 포트를 제어하기 위해 추가함.
     - #include <stdio.h> // 표준 입출력 함수를 사용하기 위해 추가함.
2. GPIO 포트를 초기화함.
   - int wiringPiSetup(void) 함수를 호출함.
   - 예
     - wiringPiSetup();
3. 핀 동작 모드를 설정함.
   - void pinMode(int pin, int mode) 함수를 이용함.
   - 예
     - pinMode(3, OUTPUT); // 3번 핀을 출력 핀(OUTPUT)으로 설정함.
     - pinMode(4, INPUT); // 4번 핀을 입력 핀(INPUT)으로 설정함.
4. 입출력 함수를 이용해 GPIO핀으로부터 값을 입출력함.
   - 예
     - digitalWrite(3, HIGH); // 3번 핀에 HIGH 신호를 출력함.
     - value=digitalRead(4); // 4번 핀의 상태 값을 읽어 변수 value에 저장함.

실습 과정

1. 회로를 구성함.
2. 소스 코드를 작성함.
   - GNU nano 에디터를 이용함.
3. 소스 코드를 빌드한 후, 실행 파일을 실행 시킴.
   - gcc 컴파일러를 이용함.

키트의 LED 회로


GPIO 핀의
wPi 번호
27
28
29
25
R60
330
R61
330
R62
330
R63
330
RED 5PI
GREEN 5PI
YELLOW 5PI
WHITE 5PI
LED3
LTL-1224A
LED4
LTL-1224A
LED5
LTL-1224A
LED6
LTL-1224A

코드 작성

- 파일 열기
  - nano 파일명
  - 예
    - nano led.c
- 저장
  - 컨트롤 키+O
- 나가기
  - 컨트롤+X

led.c

```
#include <wiringPi.h>     // GPIO 입출력 라이브러리를 추가함.

int main()
{
   const int led_pins[]={25,29,28,27};   // GPIO 핀의 wPi 번호
   const int n_led_pins=sizeof(led_pins)/sizeof(led_pins[0]);

   wiringPiSetup();       // GPIO 포트를 초기화함.

   for(int i=0; i<n_led_pins; ++i) {
      pinMode(led_pins[i], OUTPUT);     // 출력 핀으로 설정함.
      digitalWrite(led_pins[i], LOW);   // LED 끄기.
   }

   while(1) {
      for(int i=0; i<n_led_pins; ++i)
         digitalWrite(led_pins[i], HIGH); // LED 켜기.

      delay(1000);    // 1초 멈춤.

      for(int i=0; i<n_led_pins; ++i)
         digitalWrite(led_pins[i], LOW);  // LED 끄기.

      delay(1000);    // 1초 멈춤.
   }

   return 0;
}
```

빌드 및 실행

- 빌드
  - gcc -o 실행파일명 소스코드 -lwiringPi
  - 예
    - gcc -o led led.c -lwiringPi
      - gcc를 이용하여 led.c 파일을 빌드함.
      - wiringPi 라이브러리를 이용함. (-lwiringPi)
      - 생성되는 실행파일명을 led로 지정함. (-o led)
- 실행
  - ./실행파일명
  - 예
    - ./led
      - 현재 작업 디렉터리(.) 안에 있는 led라는 이름의 실행 파일을 실행함.

sizeof() 연산자

- 자료형의 크기를 반환함.
- 예
  - char a=10; int b=-10; char c[10]; int d[10];
  - sizeof(char) ➔ 1 // char형의 크기를 반환함.
  - sizeof(int) ➔ 4 // int형의 크기를 반환함.
  - sizeof(a) ➔ 1 // char형의 크기를 반환함.
  - sizeof(b) ➔ 4 // int형의 크기를 반환함.
  - sizeof(c[0]) ➔ 1 // char형의 크기를 반환함.
  - sizeof(c) ➔ 10 // 배열 c의 크기를 반환함. 1x10=10
  - sizeof(d) ➔ 40 // 배열 d의 크기를 반환함. 4x10=40
- 배열 d의 길이를 알아 내려면?
  - sizeof(d)/sizeof(d[0]) ➔ 10 // 40÷4=10
  - sizeof(d)/sizeof(*d) ➔ 10 // 40÷4=10
  - sizeof(d)/sizeof(int) ➔ 10 // 40÷4=10

질문

Q&A