IoT Platform 3rd Week

- Programming on RPi -

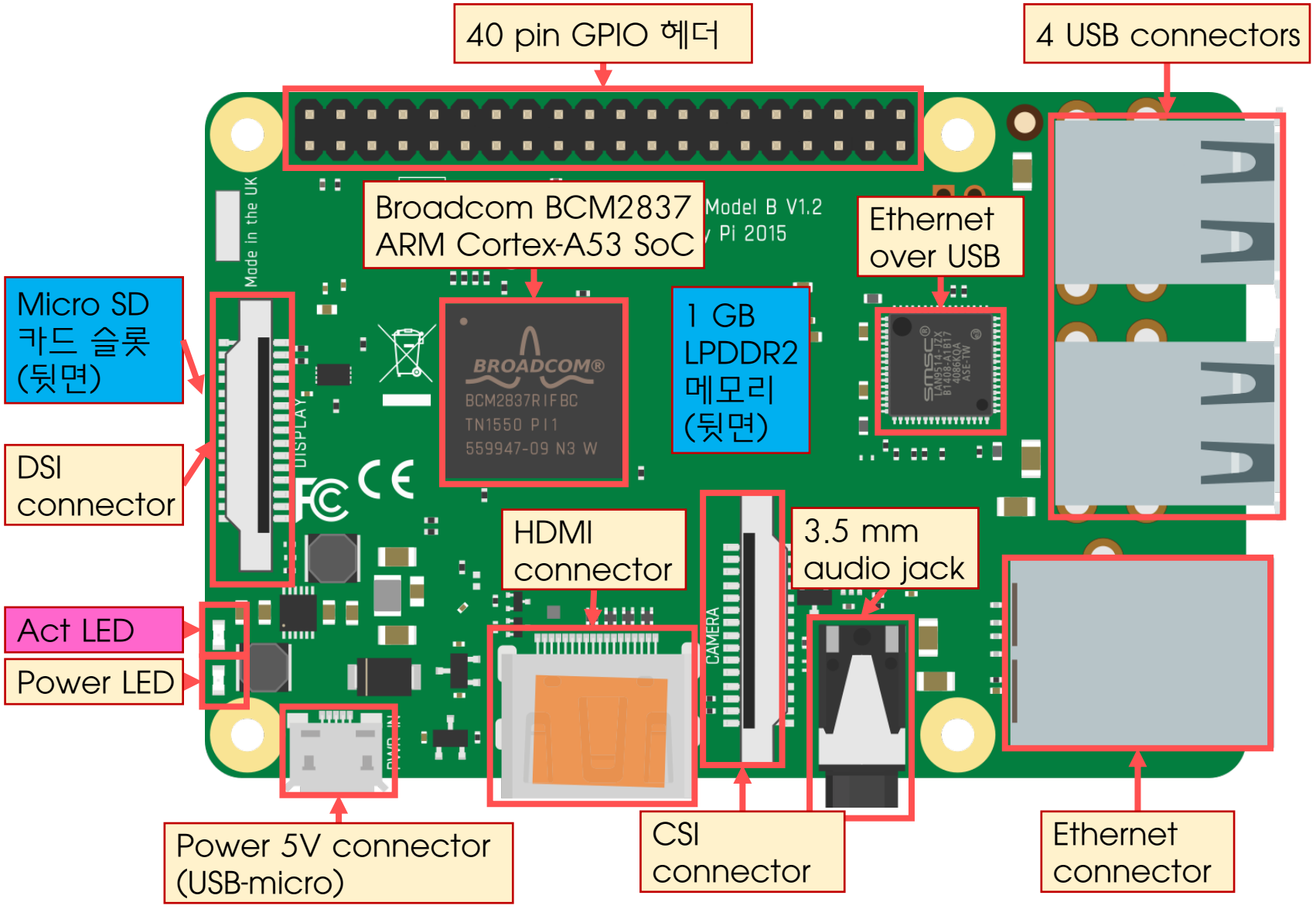
Jaeseok Yun

Soonchunhyang University

온보드 LED

sysfs

온보드 LED 다루기 sysfs



[1] 익스플로링 라즈베리 파이, p10

온보드 LED 다루기 sysfs

Practice

■ sysfs:

- 리눅스 커널 제공 **가상 파일 시스템**, 하드웨어와 디바이스 드라이버 **정보 접근** 가능
- 다양한 **기기와 커널 하부**에 대한 정보를 제공할 뿐만 아니라 **설정/제어**까지 가능

pi@raspberrypi: /sys/class/leds/led0

```
pi@raspberrypi:~ $ cd /sys/class/leds/
pi@raspberrypi:/sys/class/leds $ ls
input0::capslock  input0::numlock  input0::scrolllock  led0  led1
pi@raspberrypi:/sys/class/leds $ cd led0
pi@raspberrypi:/sys/class/leds/led0 $ ls
brightness  device  max_brightness  power  subsystem  trigger  uevent
pi@raspberrypi:/sys/class/leds/led0 $ cat trigger
none rc-feedback kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock kb
rllock kbd-altlock kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctrl
klight gpio cpu cpu0 cpu1 cpu2 cpu3 default-on input panic mmc1 [mmc0] r
pi@raspberrypi:/sys/class/leds/led0 $ sudo sh -c "echo none > trigger"
pi@raspberrypi:/sys/class/leds/led0 $ cat trigger
[none] rc-feedback kbd-scrolllock kbd-numlock kbd-capslock kbd-kanalock
ctrllock kbd-altlock kbd-shiftllock kbd-shiftrlock kbd-ctrlllock kbd-ctr
acklight gpio cpu cpu0 cpu1 cpu2 cpu3 default-on input panic mmc1 mmc0 rfkill-any rfkill0 rfkill1
pi@raspberrypi:/sys/class/leds/led0 $ sudo sh -c "echo 1 > brightness"
pi@raspberrypi:/sys/class/leds/led0 $ sudo sh -c "echo 0 > brightness"
pi@raspberrypi:/sys/class/leds/led0 $ sudo sh -c "echo timer > trigger"
pi@raspberrypi:/sys/class/leds/led0 $ ls
brightness  delay_off  delay_on  device  max_brightness  power  subsystem  trigger  uevent
pi@raspberrypi:/sys/class/leds/led0 $ cat delay_on
500
pi@raspberrypi:/sys/class/leds/led0 $ cat delay_off
500
pi@raspberrypi:/sys/class/leds/led0 $ sudo sh -c "echo 100 > delay_on"
pi@raspberrypi:/sys/class/leds/led0 $ sudo sh -c "echo 100 > delay_off"
pi@raspberrypi:/sys/class/leds/led0 $ sudo sh -c "echo mmc0 > trigger"
pi@raspberrypi:/sys/class/leds/led0 $
```

- cat 명령은 파일을 읽어서 출력
- Micro SD 카드 동작에 연결
- sh -c "문자열"은 문자열을 명령어로 실행하는 의미
- Micro SD 카드 동작 연결 해제
- ACT LED 켜기/끄기
- ACT LED 타이머 설정
- 기본 딜레이 확인 (0.5 초 on, 0.5 초 off)
- 0.1 초 on, 0.1 초 off 변경
- ACT LED를 다시 Micro SD 카드 동작에 연결

[1] 익스플로링 라즈베리 파이, p52

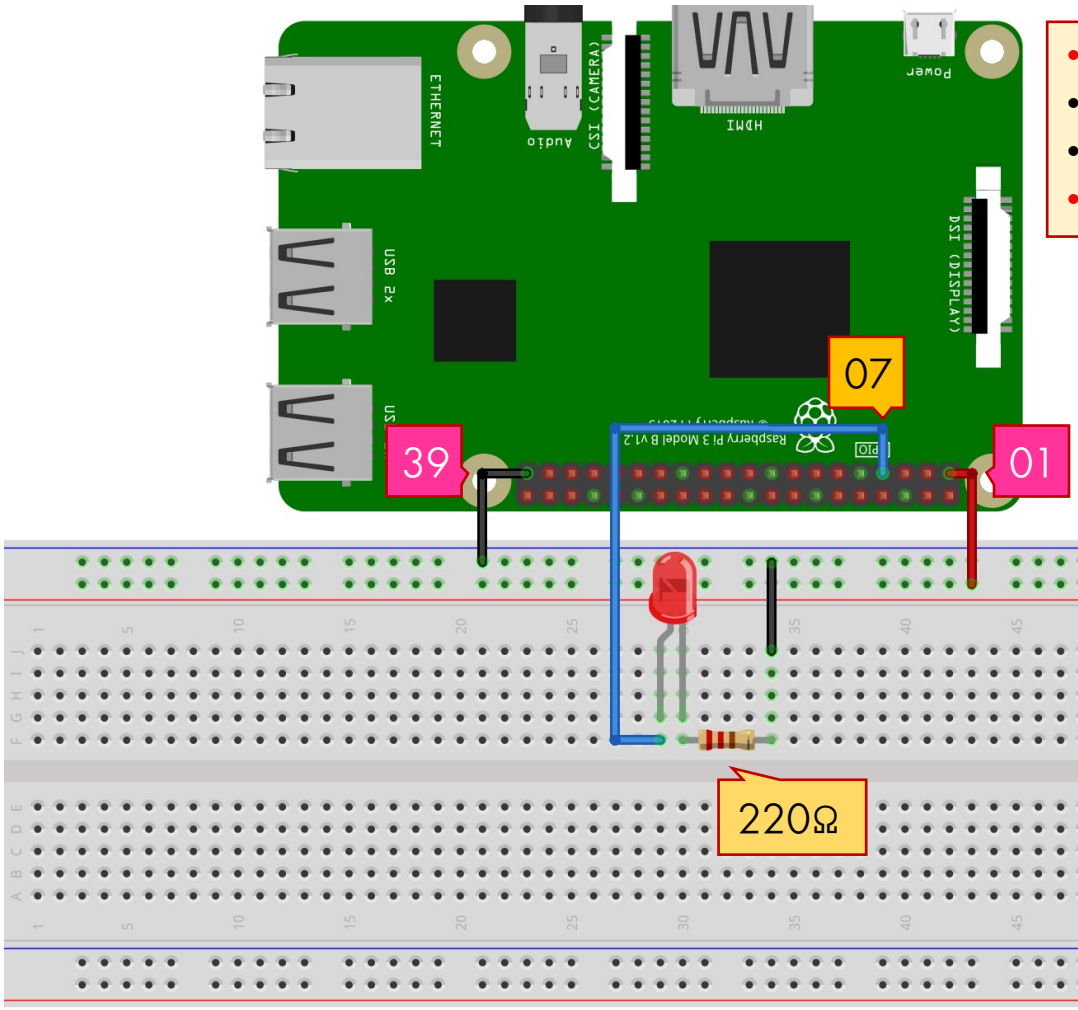
LED 동작 제어

sysfs

LED 제어 회로 구성

Practice

■ 리눅스 sysfs (파일 시스템) 활용 GPIO 직접 제어



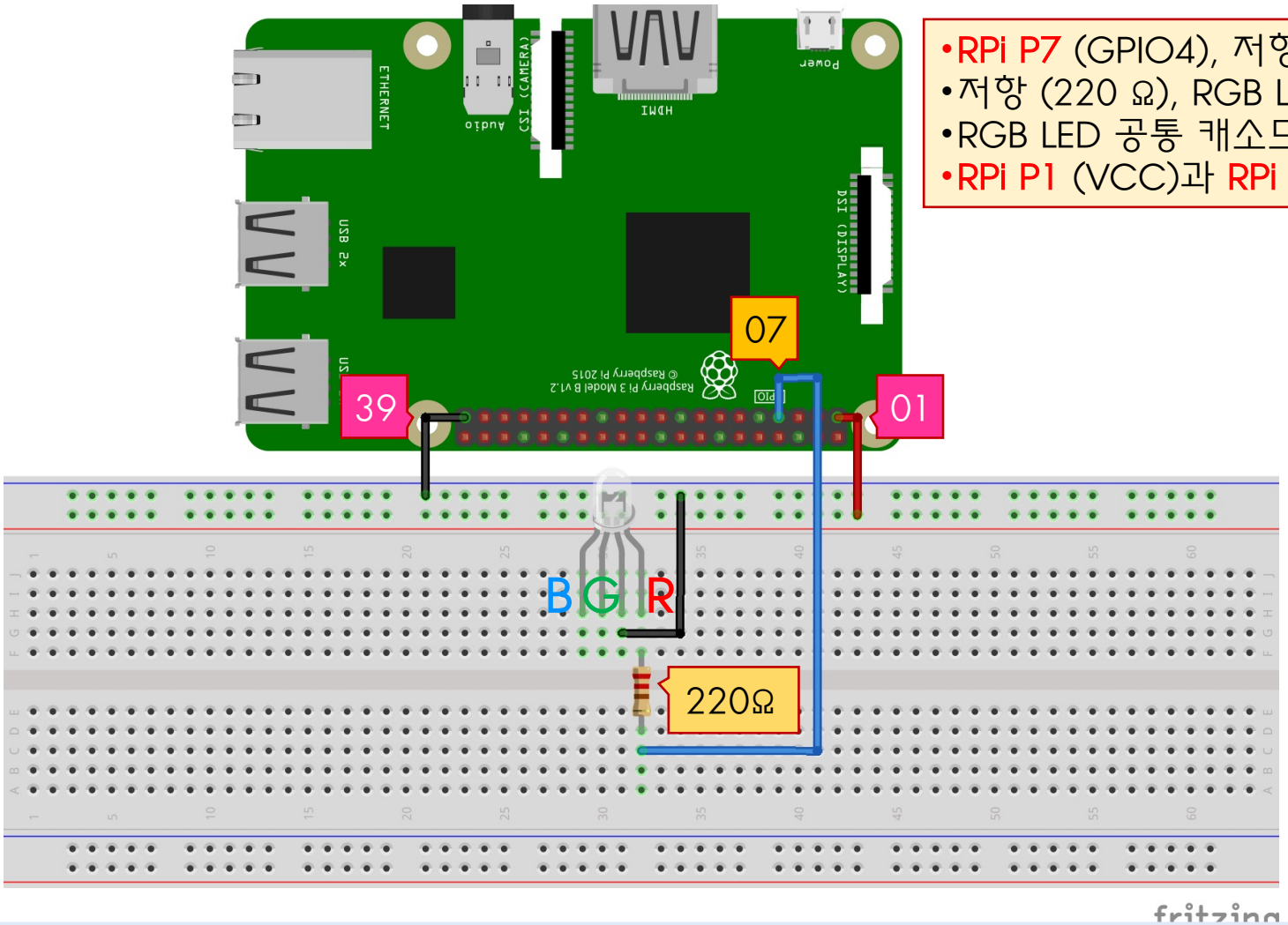
- RPi P7 (GPIO4), LED 애노드 (+) 연결
- LED 캐소드 (-), 저항 (220 Ω) 연결
- 저항 (220 Ω), GND 연결
- RPI P1 (VCC)과 RPi P39 (GND) 활용

[1] 익스플로링 라즈베리 파이, p173

LED 제어 회로 구성

Practice

■ 리눅스 sysfs (파일 시스템) 활용 GPIO 직접 제어



- RPi P7 (GPIO4), 저항 (220 Ω) 연결
- 저항 (220 Ω), RGB LED R/G/B 핀 연결
- RGB LED 공통 캐소드, RPi GND 연결
- RPi P1 (VCC)과 RPi P39 (GND) 활용

[1] 익스플로링 라즈베리 파이, p173

LED 제어 sysfs

Practice

■ 리눅스 sysfs (파일 시스템) 활용 GPIO 직접 제어

pi@raspberrypi: /sys/class/gpio

```
pi@raspberrypi:~ $ cd /sys/class/gpio/
pi@raspberrypi:/sys/class/gpio $ ls
export  gpiochip0  gpiochip100  gpiochip128  unexport
pi@raspberrypi:/sys/class/gpio $ echo 4 > export
pi@raspberrypi:/sys/class/gpio $ ls
export  gpio4  gpiochip0  gpiochip100  gpiochip128  unexport
pi@raspberrypi:/sys/class/gpio $ cd gpio4
pi@raspberrypi:/sys/class/gpio/gpio4 $ ls
active_low  device  direction  edge  power  subsystem  uevent  value
pi@raspberrypi:/sys/class/gpio/gpio4 $ echo out > direction
pi@raspberrypi:/sys/class/gpio/gpio4 $ echo 1 > value
pi@raspberrypi:/sys/class/gpio/gpio4 $ echo 0 > value
pi@raspberrypi:/sys/class/gpio/gpio4 $ cat direction
out
pi@raspberrypi:/sys/class/gpio/gpio4 $ cat value
0
pi@raspberrypi:/sys/class/gpio/gpio4 $ cd ..
pi@raspberrypi:/sys/class/gpio $ echo 4 > unexport
pi@raspberrypi:/sys/class/gpio $ ls
export  gpiochip0  gpiochip100  gpiochip128  unexport
pi@raspberrypi:/sys/class/gpio $
```

- /sys/class/gpio 는 RPi GPIO 접근 가상 파일 시스템
- RPi의 GPIO04를 활성화
- 활성화된 GPIO4 디렉토리 이동
- direction: 입력 (in), 출력 (out)
- value: 입출력 값 1, 0
- cat으로 direction과 value 확인 가능
- GPIO04 비활성화

[1] 익스플로링 라즈베리 파이, p173

LED 제어 프로그래밍

Java

exploringrpi/chp05/javaLED/LEDExample.java

LED 제어 Java

Practice

```
package exploringRPi;
import java.io.*;

public class LEDExample {

  private static String GPIO4_PATH = "/sys/class/gpio/gpio4/";
  private static String GPIO_SYSFS = "/sys/class/gpio/";

  private static void writeSysfs(String filename, String value, String path){
     try{
        BufferedWriter bw = new BufferedWriter(new FileWriter (path+filename));
        bw.write(value);
        bw.close();
     }
     catch(IOException e){
        System.err.println("Failed to access the RPi Sysfs file: " + filename);
     }
  }

  public static void main(String[] args) {
    System.out.println("Starting the LED Java Application");
    if(args.length != 1) {
       System.out.println("There is an incorrect number of arguments.");
       System.out.println("  Correct usage is: LEDExample command");
       System.out.println("where command is one of setup, on, off, status, or close");
       System.exit(2);
    }
```

- writeSysfs() 사용 예제
- writeSysfs("export", "4", GPIO_SYSFS)
- writeSysfs("direction", "out", GPIO4_PATH);
- writeSysfs("unexport", "4", GPIO_SYSFS);

- GPIO04 를 접근하기 위한 sysfs 디렉토리 경로를 변수에 저장

- sysfs를 이용해 GPIO4에 값을 출력해 LED를 제어하기 위한 함수 writeSysfs()

- 실행 인수가 하나가 아닐 때 오류 출력
- 예, java javaLED on off
  (Java 클래스 파일이 javaLED.class 라고 가정)

[1] 익스플로링 라즈베리 파이, p186

exploringrpi/chp05/javaLED/LEDExample.java

LED 제어 Java

Practice

```
  private static String GPIO4_PATH = "/sys/class/gpio/gpio4/";
  private static String GPIO_SYSFS = "/sys/class/gpio/";
```

```
    if (args[0].equalsIgnoreCase("On") || args[0].equalsIgnoreCase("Off")){
      System.out.println("Turning the LED " + args[0]);
      writeSysfs("value", args[0].equalsIgnoreCase("On")? "1":"0", GPIO4_PATH);
    }
    else if (args[0].equalsIgnoreCase("setup")) {
      System.out.println("Setting up the LED");
      writeSysfs("export", "4", GPIO_SYSFS);
      try{ Thread.sleep(100); } catch(InterruptedException e){} //sleep to ensure that gpio is exported
      writeSysfs("direction", "out", GPIO4_PATH);
    }
    else if (args[0].equalsIgnoreCase("close")) {
      System.out.println("Closing down the LED");
      writeSysfs("unexport", "4", GPIO_SYSFS);
    }
    else if (args[0].equalsIgnoreCase("status")){
      try {
        BufferedReader br = new BufferedReader(new FileReader(GPIO4_PATH+"value"));
        String line;
        while ((line = br.readLine()) != null) { System.out.println(line); }
        br.close();
      } catch(IOException e) { System.err.println("Failed to access the sysfs entry: /value");}
    }
    else {
      System.out.println("Invalid command");
    }
  }
}
```

- 실행 인수 'on', LED 켜기
- 실행 인수 'off', LED 끄기
- 실행 인수 'setup', GPIO 설정
- 실행 인수 'close', GPIO 해제
- 실행 인수 'status', 상태 출력

[1] 익스플로링 라즈베리 파이, p186

LED 제어 Java

Practice

pi@raspberrypi: ~/exploringrpi/chp05/javaLED

```
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ ls
build  exploringRPi  LEDExample.java  run
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ java exploringRPi.LEDExample setup
Starting the LED Java Application
Setting up the LED
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ java exploringRPi.LEDExample on
Starting the LED Java Application
Turning the LED on
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ java exploringRPi.LEDExample status
Starting the LED Java Application
1
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ java exploringRPi.LEDExample off
Starting the LED Java Application
Turning the LED off
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ java exploringRPi.LEDExample status
Starting the LED Java Application
0
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ java exploringRPi.LEDExample close
Starting the LED Java Application
Closing down the LED
pi@raspberrypi:~/exploringrpi/chp05/javaLED $
```

- 실행 인수 'setup', GPIO 설정
- 실행 인수 'on', LED 켜기
- 실행 인수 'status', LED 상태 확인
- 실행 인수 'off', LED 켜기
- 실행 인수 'status', LED 상태 확인
- 실행 인수 'close', GPIO 해제

- 최신 RPi OS (예, Buster) 에서는 Java 설치가 필요
- 상용 오라클 Java가 아닌 오픈 소스 Java 인 OpenJDK 8 설치[2]

```
$ sudo apt update
$ sudo apt install openjdk-8-jdk
$ java -version
```

[1] 익스플로링 라즈베리 파이, p186
[2] https://linuxize.com/post/install-java-on-raspberry-pi/

LED 제어 Java

Practice

pi@raspberrypi: ~/exploringrpi/chp05/javaLED

```
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ ls
build  exploringRPi  LEDExample.java  run
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ cat run
#!/bin/bash
echo "*** Setting up the LED"
java exploringRPi.LEDExample setup
echo "*** Turning the LED On"
java exploringRPi.LEDExample on
echo "*** Displaying the LED status"
java exploringRPi.LEDExample status
echo "*** On -- Sleeping for 2 seconds"
sleep 2
java exploringRPi.LEDExample off
echo "*** Off -- Sleeping for 2 seconds"
sleep 2
echo "*** Close the GPIO entry (the LED may remain on)"
java exploringRPi.LEDExample close
pi@raspberrypi:~/exploringrpi/chp05/javaLED $
```

- 셸 스크립트를 이용해 자동 프로그램
- LED를 2초간 켜고 끄기

[1] 익스플로링 라즈베리 파이, p186

LED 제어 Java

Practice

pi@raspberrypi: ~/exploringrpi/chp05/javaLED

```
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ ls
build  exploringRPi  LEDExample.java  run
pi@raspberrypi:~/exploringrpi/chp05/javaLED $ sh run
*** Setting up the LED
Starting the LED Java Application
Setting up the LED
*** Turning the LED On
Starting the LED Java Application
Turning the LED on
*** Displaying the LED status
Starting the LED Java Application
1
*** On -- Sleeping for 2 seconds
Starting the LED Java Application
Turning the LED off
*** Off -- Sleeping for 2 seconds
*** Close the GPIO entry (the LED may remain on)
Starting the LED Java Application
Closing down the LED
pi@raspberrypi:~/exploringrpi/chp05/javaLED $
```

- 셸 스크립트를 이용해 자동 프로그램
- LED를 2초간 켜고 끄기

[1] 익스플로링 라즈베리 파이, p186

LED 제어 프로그래밍

C

LED 제어 c

- 자신만의 작업 디렉토리를 만들어서 소스 코드를 복사하고 컴파일을 직접 해보기

Practice

```
$ cd
$ mkdir ex
$ cd ex
$ cp ~/exploringrpi/chp05/makeLED/makeLED.c ./
$ gcc ./makeLED.c –o makeLED
```

- writeGPIO() 사용 예제
- writeGPIO(GPIO_SYSFS "export", GPIO_NUMBER);
  writeGPIO(GPIO4_PATH "direction", "out");
- writeGPIO(GPIO4_PATH "value", "1");

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define GPIO_NUMBER "4"
#define GPIO4_PATH "/sys/class/gpio/gpio4/"
#define GPIO_SYSFS "/sys/class/gpio/"

void writeGPIO(char filename[], char value[]){
  FILE* fp;                          // create a file pointer fp
  fp = fopen(filename, "w+");        // open file for writing
  fprintf(fp, "%s", value);          // send the value to the file
  fclose(fp);                        // close the file using fp
}

int main(int argc, char* argv[]){
  if (argc!=2){                       // program name is argument 1
    printf("Usage is makeLEDC and one of:\n");
    printf("   setup, on, off, status, or close\n");
    printf(" e.g. makeLEDC on\n");
    return 2;                        // invalid number of arguments
  }
  printf("Starting the makeLED program\n");
```

- sysfs를 이용해 GPIO04에 값을 출력해 LED를 제어하기 위한 함수

- 실행 인수가 둘이 (프로그램 이름이 기본으로 포함되고 제어 문자열이 추가) 아닐 때 오류 출력

[1] 익스플로링 라즈베리 파이, p204

exploringrpi/chp05/makeLED/makeLED.c

LED 제어 c

Practice

```
#define GPIO_NUMBER "4"
#define GPIO4_PATH "/sys/class/gpio/gpio4/"
#define GPIO_SYSFS "/sys/class/gpio/"
```

```
if (strcmp(argv[1],"setup")==0){
  printf("Setting up the LED on the GPIO\n");
  writeGPIO(GPIO_SYSFS "export", GPIO_NUMBER);
  usleep(100000);                    // sleep for 100ms
  writeGPIO(GPIO4_PATH "direction", "out");
}
else if (strcmp(argv[1],"close")==0){
  printf("Closing the LED on the GPIO\n");
  writeGPIO(GPIO_SYSFS "unexport", GPIO_NUMBER);
}
else if (strcmp(argv[1],"on")==0){
  printf("Turning the LED on\n");
  writeGPIO(GPIO4_PATH "value", "1");
}
else if (strcmp(argv[1],"off")==0){
  printf("Turning the LED off\n");
  writeGPIO(GPIO4_PATH "value", "0");
}
else if (strcmp(argv[1],"status")==0){
  FILE* fp;         // see writeGPIO function above for description
  char line[80], fullFilename[100]; sprintf(fullFilename, GPIO4_PATH "/value");
  fp = fopen(fullFilename, "rt");         // reading text this time
  while (fgets(line, 80, fp) != NULL) { printf("The state of the LED is %s", line); }
  fclose(fp);
} else {
  printf("Invalid command!\n");
} printf("Finished the makeLED Program\n");
return 0;
```

- 실행 인수 'setup', GPIO 설정
- 실행 인수 'close', GPIO 해제
- 실행 인수 'on', LED 켜기
- 실행 인수 'off', LED 끄기
- 실행 인수 'status', 상태 확인

[1] 익스플로링 라즈베리 파이, p204

LED 제어 c

Practice

pi@raspberrypi: ~/exploringrpi/chp05/makeLED

```
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ls
build  makeLED  makeLED.c  makeLEDC  makeLED.cpp
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLED setup
Starting the makeLED program
The current LED Path is: /sys/class/gpio/gpio4/
Setting up the GPIO
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLED on
Starting the makeLED program
The current LED Path is: /sys/class/gpio/gpio4/
Turning the LED on
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLED status
Starting the makeLED program
The current LED Path is: /sys/class/gpio/gpio4/
The state is: 1
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLED off
Starting the makeLED program
The current LED Path is: /sys/class/gpio/gpio4/
Turning the LED off
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLED status
Starting the makeLED program
The current LED Path is: /sys/class/gpio/gpio4/
The state is: 0
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLED close
Starting the makeLED program
The current LED Path is: /sys/class/gpio/gpio4/
Unexporting the GPIO
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $
```

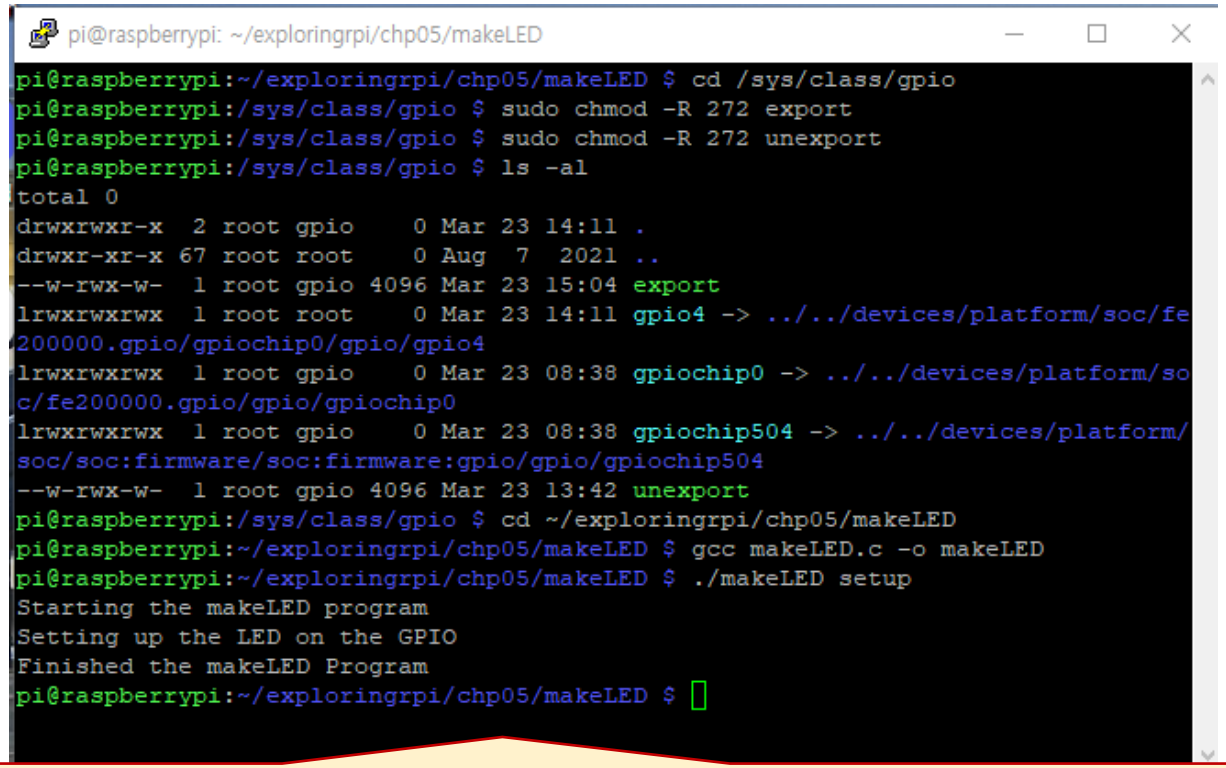
- 실행 인수 '**setup**', GPIO 설정
- 실행 인수 '**on**', LED 켜기
- 실행 인수 '**status**', 상태 확인
- 실행 인수 '**off**', LED 끄기
- 실행 인수 '**status**', 상태 확인
- 실행 인수 '**close**', GPIO 해제

[1] 익스플로링 라즈베리 파이, p204

LED 제어 c

Practice

Hotfix!
(2022/3)



- 'Segmentation fault' 발생시 대처법
- `$ cd /sys/class/gpio` (디렉토리 이동)
- `$ sudo chmod -R 272 export`
  (권한 부여 - 만약 위 방법으로 해결되지 않는다면 272 대신 777 사용 - 모든 사용자, 그룹에 권한 부여)
- `$ sudo chmod -R 272 unexport`
  (권한 부여 - 만약 위 방법으로 해결되지 않는다면 272 대신 777 사용 - 모든 사용자, 그룹에 권한 부여)
- `$ cd ~/exploringpi/chp05/makeLED` (디렉토리 이동)
- `$ gcc makeLED.c -o makeLED` (컴파일)
- `$ ./makeLED setup` (실행)

LED 제어 프로그래밍

C++ (not OPP)

LED 제어 C++ (not OPP)

Practice

- 자신만의 작업 디렉토리를 만들어서 소스 코드를 복사하고 컴파일을 직접 해보기

```
$ cd
$ mkdir ex
$ cd ex
$ cp ~/exploringrpi/chp05/makeLED/makeLED.cpp ./
$ g++ ./makeLED.cpp -o makeLEDC
```

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <unistd.h>
using namespace std;

#define GPIO_NUMBER "4"
#define GPIO4_PATH "/sys/class/gpio/gpio4/"
#define GPIO_SYSFS "/sys/class/gpio/"

void writeGPIO(string path, string filename, string value){
   fstream fs;
   fs.open((path + filename).c_str(), fstream::out);
   fs << value;
   fs.close();
}

int main(int argc, char* argv[]){
   if(argc!=2){
      cout << "Usage is makeLED and one of: " << endl;
      cout << "   setup, on, off, status, or close" << endl;
      cout << " e.g. makeLED on" << endl;
      return 2;
   }
   string cmd(argv[1]);
   cout << "Starting the makeLED program" << endl;
   cout << "The current LED Path is: " << GPIO4_PATH << endl;
```

- writeGPIO() 사용 예제
- writeGPIO(string(GPIO_SYSFS), "export", GPIO_NUMBER);
- writeGPIO(string(GPIO4_PATH), "direction", "out");
- writeGPIO(string(GPIO4_PATH), "value", "1");

- sysfs를 이용해 GPIO04에 값을 출력해 LED를 제어하기 위한 함수

- 실행 인수가 둘이 (프로그램 이름이 기본으로 포함되고 제어 문자열이 추가) 아닐 때 오류 출력

[1] 익스플로링 라즈베리 파이, p208

exploringrpi/chp05/makeLED/makeLED.cpp

LED 제어 C++ (not OPP)

Practice

```
#define GPIO_NUMBER "4"
#define GPIO4_PATH "/sys/class/gpio/gpio4/"
#define GPIO_SYSFS "/sys/class/gpio/"
```

```
if (cmd=="on"){
  cout << "Turning the LED on" << endl;
  writeGPIO(string(GPIO4_PATH), "value", "1");
}
else if (cmd=="off"){
  cout << "Turning the LED off" << endl;
  writeGPIO(string(GPIO4_PATH), "value", "0");
}
else if (cmd=="setup"){
  cout << "Setting up the GPIO" << endl;
  writeGPIO(string(GPIO_SYSFS), "export", GPIO_NUMBER);
  usleep(100000);
  writeGPIO(string(GPIO4_PATH), "direction", "out");
}
else if (cmd=="close"){
  cout << "Unexporting the GPIO" << endl;
  writeGPIO(string(GPIO_SYSFS), "unexport", GPIO_NUMBER);
}
else if (cmd=="status"){
  std::fstream fs;
  fs.open( GPIO4_PATH "value", std::fstream::in);
  string line;
  while(getline(fs,line)) cout << "The state is: " << line << endl;
  fs.close();
}
else { cout << "Invalid command!" << endl; }
cout << "Finished the makeLED Program" << endl;
return 0;
```

- 실행 인수 'on', LED 켜기
- 실행 인수 'off', LED 끄기
- 실행 인수 'setup', GPIO 설정
- 실행 인수 'close', GPIO 해제
- 실행 인수 'status', 상태 확인

[1] 익스플로링 라즈베리 파이, p208

LED 제어 C++ (not OPP)

Practice

pi@raspberrypi: ~/exploringrpi/chp05/makeLED

```
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ls
build  makeLED  makeLED.c  makeLEDC  makeLED.cpp
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLEDC setup
Starting the makeLED program
Setting up the LED on the GPIO
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLEDC on
Starting the makeLED program
Turning the LED on
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLEDC status
Starting the makeLED program
The state of the LED is 1
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLEDC off
Starting the makeLED program
Turning the LED off
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLEDC status
Starting the makeLED program
The state of the LED is 0
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $ ./makeLEDC close
Starting the makeLED program
Closing the LED on the GPIO
Finished the makeLED Program
pi@raspberrypi:~/exploringrpi/chp05/makeLED $
```

- 실행 인수 'setup', GPIO 설정
- 실행 인수 'on', LED 켜기
- 실행 인수 'status', 상태 확인
- 실행 인수 'off', LED 끄기
- 실행 인수 'status', 상태 확인
- 실행 인수 'close', GPIO 해제

[1] 익스플로링 라즈베리 파이, p208

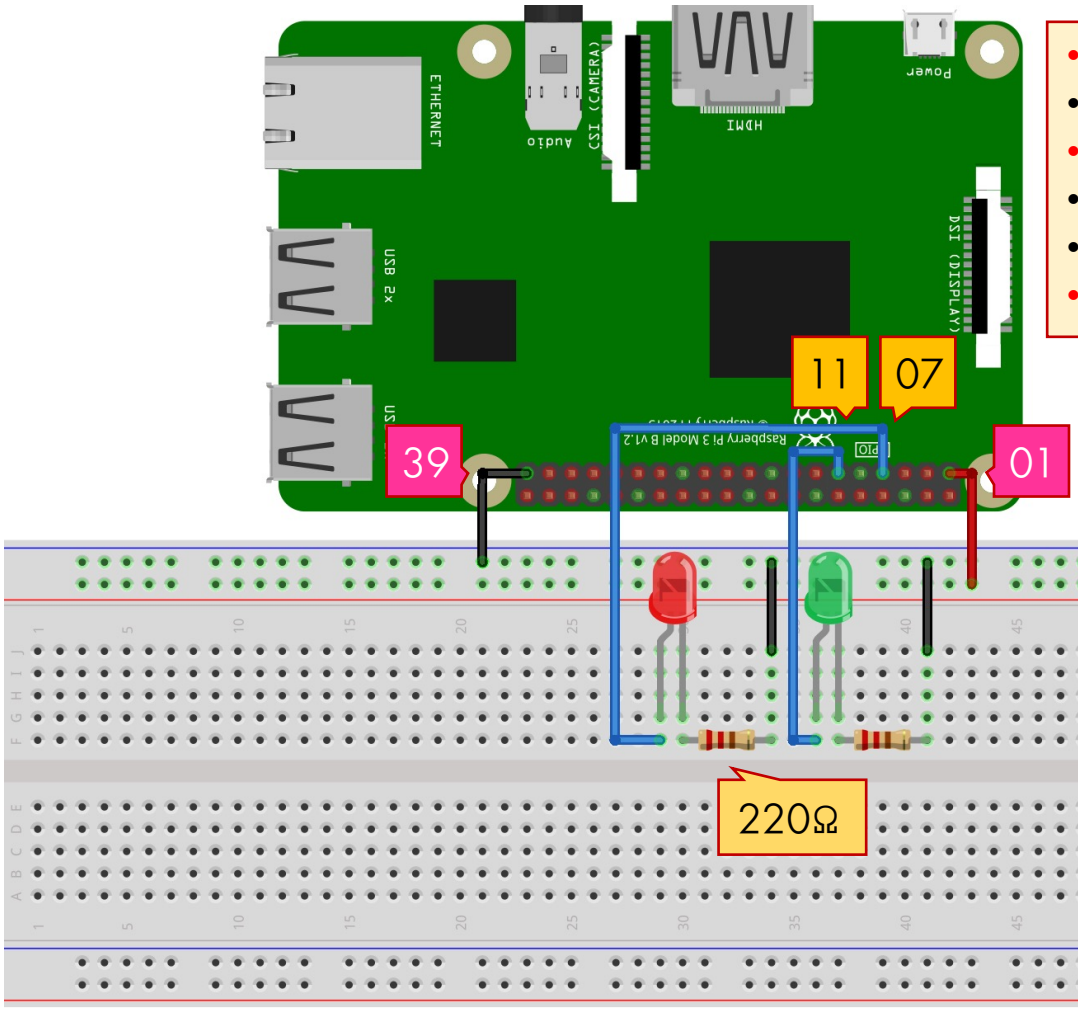
LED 제어 프로그래밍

C++ (OOP)

LED 제어 회로 구성

Practice

■ 리눅스 sysfs (파일 시스템) 활용 GPIO 직접 제어



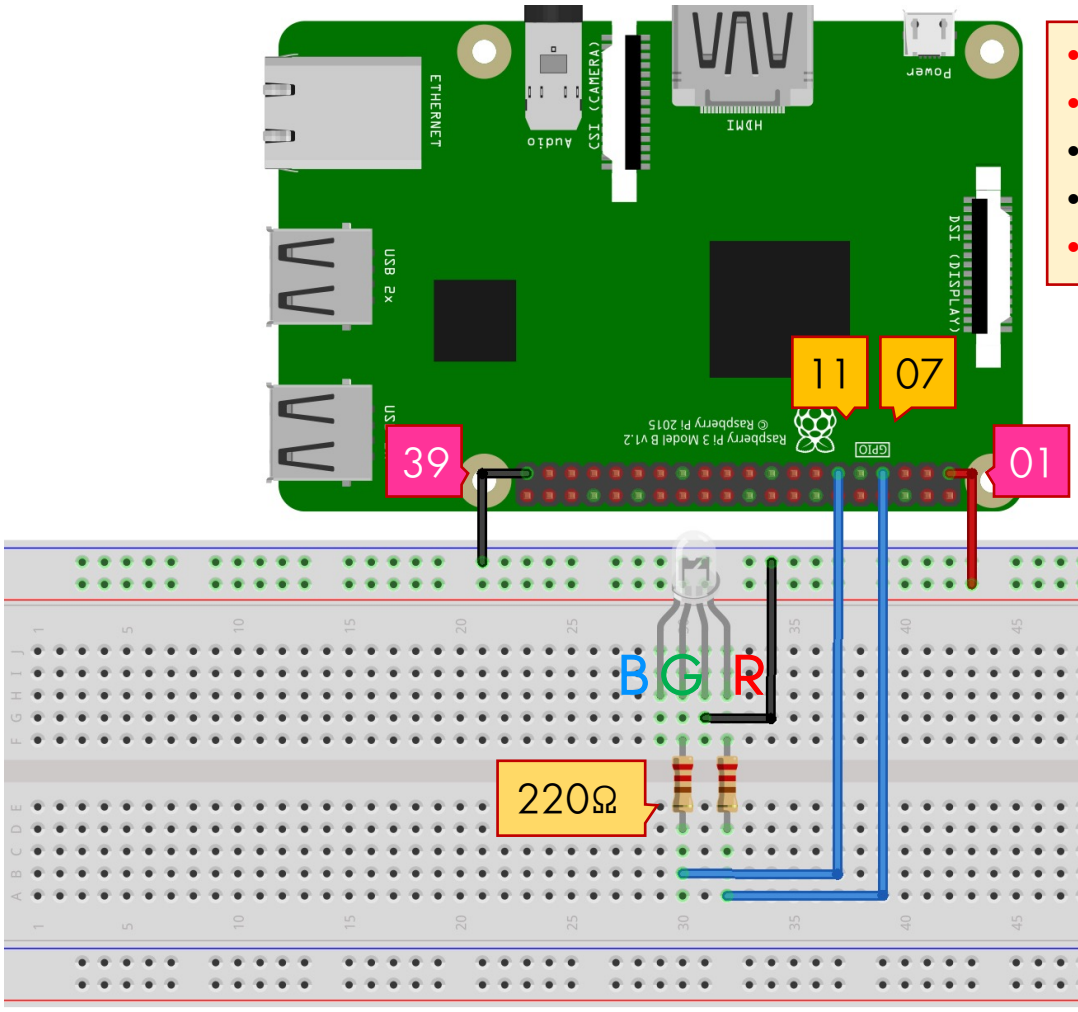
- RPi P7 (GPIO4), LED 애노드 (+) 연결
- LED 캐소드 (-), 저항 (220 Ω) 연결
- RPi P11 (GPIO17), LED 애노드 (+) 연결
- LED 캐소드 (-), 저항 (220 Ω) 연결
- 저항 (220 Ω), GND 연결
- RPI P1 (VCC)과 RPi P39 (GND) 활용

[1] 익스플로링 라즈베리 파이, p173

LED 제어 회로 구성

Practice

■ 리눅스 sysfs (파일 시스템) 활용 GPIO 직접 제어



- RPi P7 (GPIO4), 저항 (220 Ω) 연결
- RPi P11 (GPIO17), 저항 (220 Ω) 연결
- 저항 (220 Ω), RGB LED R/G/B 핀 연결
- RGB LED 공통 캐소드, RPi GND 연결
- RPi P1 (VCC)과 RPi P39 (GND) 활용

[1] 익스플로링 라즈베리 파이, p173

exploringrpi/chp05/makeLEDOOP/makeLEDs.cpp

LED 제어 C++ (OOP)

Practice

- 자신만의 작업 디렉토리를 만들어서 소스 코드를 복사하고 컴파일을 직접 해보기

```
$ cd
$ mkdir ex
$ cd ex
$ cp ~/exploringrpi/chp05/makeLEDOOP/makeLEDs.cpp ./
$ g++ ./makeLEDs.cpp -o makeLEDs
```

```
#include <iostream>
#include <fstream>
#include <string>
#include <unistd.h>          // for the mic
using namespace std;
#define GPIO         "/sys/class/gpio/"
#define FLASH_DELAY  50000 // 50 milliseconds

class LED {
  private:                  // the following is part of the implementation
    string gpioPath;      // private states
    int    gpioNumber;
    void writeSysfs(string path, string filename, string value);
  public:                   // part of the public interface
    LED(int gpioNumber); // the constructor -- create the object
    virtual void turnOn();
    virtual void turnOff();
    virtual void displayState();
    virtual ~LED();       // the destructor -- called automatically
};

LED::LED(int gpioNumber){  // constructor implementation
   this->gpioNumber = gpioNumber;
   gpioPath = string(GPIO "gpio") + to_string(gpioNumber) + string("/");
   writeSysfs(string(GPIO), "export", to_string(gpioNumber));
   usleep(100000);          // ensure GPIO is exported
   writeSysfs(gpioPath, "direction", "out");
}
```

- LED 클래스 정의
- 멤버와 메소드 정의

- LED 생성자 인수로 GPIO 핀 번호를 배정
- LED 객체 생성과 동시에 GPIO 핀 설정

- LED 클래스 생성자

[1] 익스플로링 라즈베리 파이, p214

exploringrpi/chp05/makeLEDOOP/makeLEDs.cpp

LED 제어 C++ (OOP)

Practice

```
class LED {
  private:
    string gpioPath;
    int    gpioNumber;
    void writeSysfs(string path, string filename, string value);
```

```
void LED::writeSysfs(string path, string filename, string value){
   ofstream fs;
   fs.open((path+filename).c_str());
   fs << value;
   fs.close();
}

void LED::turnOn(){
   writeSysfs(gpioPath, "value", "1");
}

void LED::turnOff(){
   writeSysfs(gpioPath, "value", "0");
}

void LED::displayState(){
   ifstream fs;
   fs.open((gpioPath + "value").c_str());
   string line;
   cout << "The current LED state is ";
   while (getline(fs,line)) cout << line << endl;
   fs.close();
}

LED::~LED(){  // The destructor unexports the sysfs GPIO entries
   cout << "Destroying the LED with GPIO number " << gpioNumber << endl;
   writeSysfs(string(GPIO), "unexport", to_string(gpioNumber));
}
```

- sysfs를 이용해 GPIO 에 값을 출력해 LED를 실제로 제어하기 위한 함수

- writeSysfs() 사용 예제
- writeSysfs(string(GPIO), "export", to_string(gpioNumber));
- writeSysfs(gpioPath, "direction", "out");
- writeSysfs(gpioPath, "value", "1");

- LED 객체를 켜는 메소드

- LED 객체를 끄는 메소드

- LED 객체의 상태를 보이는 메소드

- LED 객체가 연결된 GPIO 설정 해제

[1] 익스플로링 라즈베리 파이, p214

exploringrpi/chp05/makeLEDOOP/makeLEDs.cpp

LED 제어 C++ (OOP)

Practice

```
class LED {
  public:
    LED(int gpioNumber);
    virtual void turnOn();
    virtual void turnOff();
    virtual void displayState();
    virtual ~LED();
```

```
int main(int argc, char* argv[]){  // the main function sta
   cout << "Starting the makeLEDs program" << endl;


   LED led1(4), led2(17);          // create two LED objects


   cout << "Flashing the LEDs for 5 seconds" << endl;


   for(int i=0; i<50; i++){        // LEDs will alternate
      led1.turnOn();               // turn GPIO4 on
      led2.turnOff();              // turn GPIO17 off
      usleep(FLASH_DELAY);         // sleep for 50ms
      led1.turnOff();              // turn GPIO4 off
      led2.turnOn();               // turn GPIO17 on
      usleep(FLASH_DELAY);         // sleep for 50ms
   }


   led1.displayState();            // display final GPIO4 state
   led2.displayState();            // display final GPIO17 state
   cout << "Finished the makeLEDs program" << endl;

   return 0;
}
```

- LED 객체 led1, led2를 생성하고 GPIO4, GPIO17 로 연결
- led1를 켜고 led2를 끄기
- 50 ms 대기
- led1를 끄고 led2를 켜기
- 50 ms 대기
- 50회 반복
- led1, led2 마지막 상태 보이기

[1] 익스플로링 라즈베리 파이, p214

LED 제어 C++ (OOP)

Practice

- led1를 켜고 led2를 끄기
- 50 ms 대기
- led1를 끄고 led2를 켜기
- 50 ms 대기
- 50회 반복

[1] 익스플로링 라즈베리 파이, p214

Summary

- On-board LED control
- LED control in sysfs
- LED control in Java
- LED control in C
- LED control in C++ (Not OOP)
- LED control in C++ (OOP)

Thank you

Questions?

Contact: eclass.sch.ac.kr
(순천향대학교 학습플랫폼 LMS)