

# Raspberry Pi

많은 자료들을 특정한 규칙에 맞게 대용량의 저장장치에 보관하여 필요한 업무에 사용될 수 있는 것을 데이터베이스라고 할 수 있습니다. 프로그래밍에 있어 데이터베이스에 있는 자료를 검색하고 가공하고 저장할 수 있는 능력은 꼭 필요한 부분입니다. 이번 장은 데이터베이스에 대한 기본적인 이해를 통해서 필요한 자료를 검색하고, 가공하고, 저장하며 파이썬에서 라즈베리 파이에서 설치된 Maria 데이터베이스 서버에 접속하는 방법에 대해 알아봅시다.

CHAPTER  
**07**

# Maria 데이터베이스 활용하기

01 \_ 데이터베이스 설치

02 \_ Maria 데이터베이스 서버에 질의문을 이용한 회원데이를 작성하기

03 \_ 데이터베이스 서버 Connection

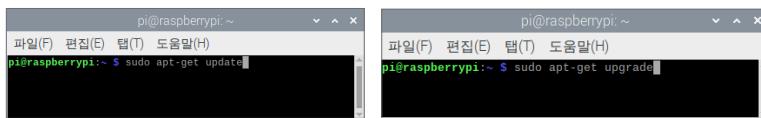
# 01 \_ 데이터베이스 설치

## 01-1 DBMS 설치하기

라즈베리 파이에서 Maria 데이터베이스 서버에 설치하는 방법은 간단합니다. 설치용 파일을 다운 받아서 설치하는 방법과 apt-get을 이용하는 방법이 있지만 apt-get를 이용하는 방법이 훨씬 간단하고 실용적이므로 본 교재도 Maria 데이터베이스 서버에 설치는 라즈베리 파이에서 apt-get방법을 이용해서 설치하도록 하겠습니다.

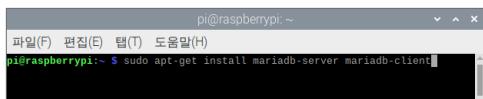
### Maria 설치

**01** 우선 시스템에 업데이트가 있나 먼저 확인을 한 후에 설치를 진행하도록 하겠습니다. 라즈베리 파이의 Terminal에서 다음과 명령어를 입력합니다.



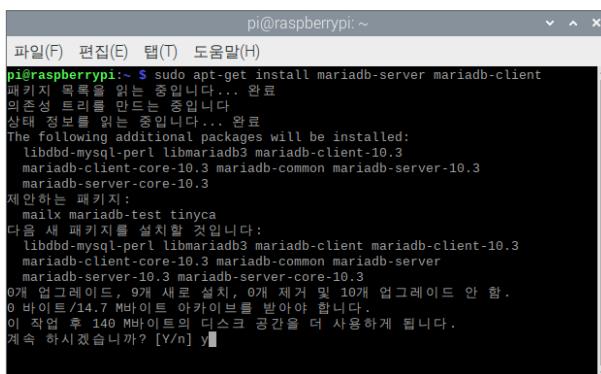
▲ 시스템 업데이트 및 업그레이드

**02** 라즈베리 파이의 Terminal에서 다음과 명령어를 입력하여 Maria 데이터베이스 서버를 설치합니다.



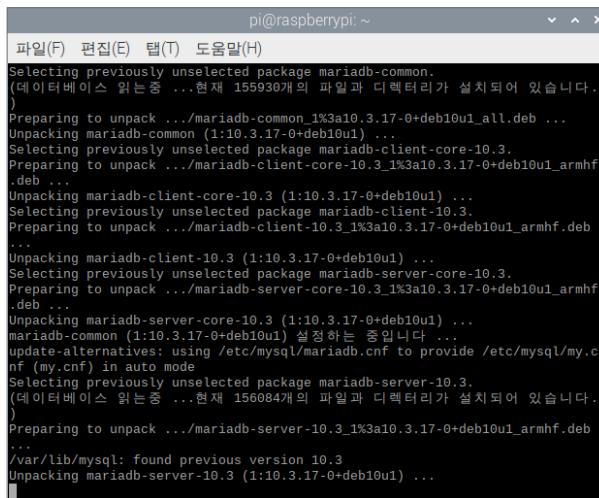
▲ 라즈베리 파이에서 Maria 데이터베이스 서버 설치

**03** 설치 중간에 다음과 같은 질문을 합니다. ‘y’라고 입력을 하고 계속 진행합니다.



▲ 라즈베리 파이에서 Maria 데이터베이스 서버 설치 시 진행 여부 질문

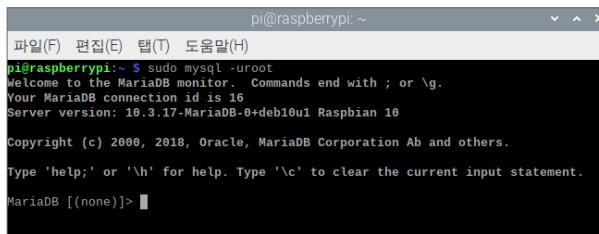
**04** 라즈베리 파이에서 Maria 데이터베이스 서버 설치 시에 보여 지는 Terminal 화면입니다. 별다른 오류가 없으면 다음과 같은 화면으로 설치가 됩니다.



```
pi@raspberrypi: ~
파일(F) 편집(E) 템(T) 도움말(H)
Selecting previously unselected package mariadb-common.
(데이터베이스 읽는 중 ...현재 155930개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack .../mariadb-common_1%3a10.3.17-0+deb10u1_all.deb ...
Unpacking mariadb-common (1:10.3.17-0+deb10u1) ...
Selecting previously unselected package mariadb-client-core-10.3.
Preparing to unpack .../mariadb-client-core-10.3_1%3a10.3.17-0+deb10u1_armhf.deb ...
Unpacking mariadb-client-core-10.3 (1:10.3.17-0+deb10u1) ...
Selecting previously unselected package mariadb-client-10.3.
Preparing to unpack .../mariadb-client-10.3_1%3a10.3.17-0+deb10u1_armhf.deb ...
...
Unpacking mariadb-client-10.3 (1:10.3.17-0+deb10u1) ...
Selecting previously unselected package mariadb-server-core-10.3.
Preparing to unpack .../mariadb-server-core-10.3_1%3a10.3.17-0+deb10u1_armhf.deb ...
Unpacking mariadb-server-core-10.3 (1:10.3.17-0+deb10u1) ...
mariadb-common (1:10.3.17-0+deb10u1) 설정하는 중입니다 ...
update-alternatives: using /etc/mysql/mariadb.cnf to provide /etc/mysql/my.cnf (my.cnf) in auto mode
Selecting previously unselected package mariadb-server-10.3.
(데이터베이스 읽는 중 ...현재 156084개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack .../mariadb-server-10.3_1%3a10.3.17-0+deb10u1_armhf.deb ...
...
/var/lib/mysql: found previous version 10.3
Unpacking mariadb-server-10.3 (1:10.3.17-0+deb10u1) ...
```

▲ 라즈베리 파이에서 Maria 데이터베이스 서버 설치 진행 화면

**05** 설치가 완료가 되었다면 라즈베리 파이 Terminal 화면에서 다음과 같은 명령어로 Maria 데이터베이스 서버에 접속을 합니다.



```
pi@raspberrypi: ~
파일(F) 편집(E) 템(T) 도움말(H)
pi@raspberrypi:~ $ sudo mysql -uroot
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 16
Server version: 10.3.17-MariaDB-0+deb10u1 Raspbian 10

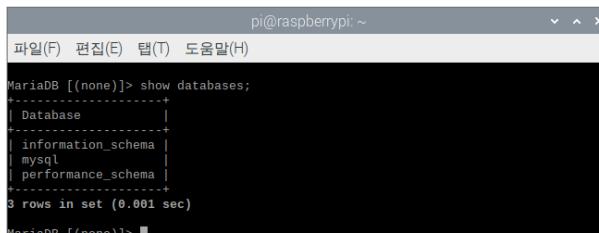
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> 
```

▲ 설치에 성공한 Maria 데이터베이스 서버에 접속

**06** 접속을 하고 나서 기본적으로 제공되는 Database을 목록을 보기 위해 다음과 같은 명령어를 입력합니다.

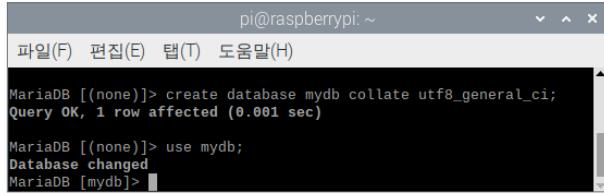


```
pi@raspberrypi: ~
파일(F) 편집(E) 템(T) 도움말(H)
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.001 sec)

MariaDB [(none)]> 
```

▲ 현재의 Database 목록

**07** 설치한 Maria 데이터베이스 서버에 예제로 사용할 mydb라는 database를 생성하고 해당 database를 사용하기 위해 다음과 명령어를 입력합니다. 7장에서 예제로 사용하게 되는 테이블(tblRegister)은 mydb라는 database에서 만들 겁니다. mydb 뒤에 있는 ‘collate utf8\_general\_ci’ 값은 한글 사용하기 위한 설정 값입니다.



```
pi@raspberrypi: ~
파일(F) 편집(E) 탭(T) 도움말(H)
MariaDB [(none)]> create database mydb collate utf8_general_ci;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use mydb;
Database changed
MariaDB [mydb]>
```

▲ mydb라는 Database를 생성하고 mydb 사용을 선택

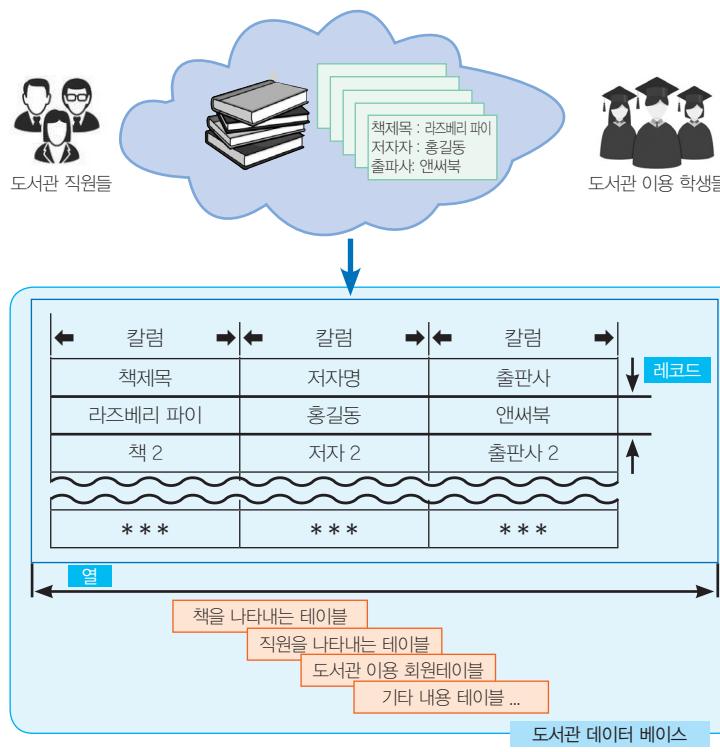
지금까지 라이베리파이에서 Maria 데이터베이스 서버를 설치하였고 mydb라는 database를 만들어 보았습니다. 이제부터는 Maria 데이터베이스 서버에서 사용할 기본적인 SQL문에 대해서 배워 보도록 하겠습니다.

## 02 \_ Maria 데이터베이스 서버에 질의문을 이용한 회원테이블 작성하기

데이터베이스에 저장된 데이터들을 다루기 위한 명령어들에 대해 하나씩 알아보도록 하겠습니다. 먼저 데이터베이스에 저장될 데이터들은 컬럼(Column), 레코드(Record), 테이블(Table), 데이터베이스(DataBase)로 나타내어집니다.

### 02-1 데이터베이스의 구성 '도서관'

도서관에는 수많은 책들이 있습니다. 그들 책은 하나하나 책이름과 저자, 출판사라는 특징을 가지고 있습니다. 좀 더 구체적으로 '라즈베리 파이' 입문'이란 책을 생각해 볼까요? 이 책은 '앤써북'이라는 출판사에서 출판하였고, 저자는 '홍길동' 이렇게 세 가지 특징을 갖고 있습니다. 이 외에도 이 도서관에는 다양한 제목을 가진 책이 많이 있습니다.



▲ 칼럼, 레코드, 테이블, 데이터베이스의 개념도

“ SQL : Structured Query Language

구조화된 질의어로 데이터베이스의 데이터들을 조작하는 명령들입니다.

예) select id, name, email, from tblRegister  
select는 어떤 데이터를 가져온다는 의미입니다. 그리고 from 뒤에 있는 부분은 어떤 공간을 지정합니다. 그래서 어떤 공간으로부터 select 하는데, select하는 부분들은 id, name, email, 이니까 이 부분을 select 하라는 의미입니다.

#### TIP

데이터베이스는 하나의 창고와 같습니다. 많은 자료를 저장할 창고가 됩니다. 이 창고를 만들고 이 창고 안에 테이블을 생성하고, 그리하여 이 테이블에 자료를 저장합니다. 이것이 가장 기본적인 데이터베이스. 테이블의 사용 순서가 된다고 하겠습니다.

- 열, 속성(Attribute) : 속성은 위의 예에서 책제목, 저자명, 출판사명 등이 될 수 있습니다. 실제 데이터베이스에서 책제목은 책제목이 들어갈 열에 저장되고, 저자명은 저자명이 들어갈 열에 저장되는 식으로 각 열에 맞는 값들이 들어가게 됩니다. 그림에서 각각의 세로로 구분되는 부분입니다. 여기서 하나의 행은 다수의 속성들로 이루어진다는 것을 알 수 있으며 여러 행들 중 각각의 행들을 구별할 수 있는 칼럼은 그 테이블의 키(key)가 됩니다.
- 레코드 : ‘라즈베리 파이 입문’이란 책은 ‘라즈베리 파이 입문’이라는 책제목, ‘홍길동’이란 저자명, ‘앤써북’이라는 출판사명을 가지는데 이렇게 여러 연관된 속성의 집합이 레코드가 됩니다. ‘라즈베리 파이 입문’, ‘홍길동’, ‘앤써북’이 ‘라즈베리 파이 입문’이라는 한 권의 책을 나타내는 한 레코드가 됩니다. 마찬가지로 ‘Python’을 나타내는 속성들로 구성된 레코드도 있을 것입니다. 결국 하나의 책은 하나의 레코드에 대응이 될 수 있습니다. 그림에서 가로의 행에 해당되는 부분입니다.

### TIP

Entity, Attribute에 대하여...

실제로 데이터베이스에 있어서 한 개체(Entity)는 하나의 테이블로 표현됩니다. 개체는 바로 책, 직원, 회원들이 됩니다. 다른 예를 든다면, 개체는 학생, 교수, 강좌 등 독립적으로 존재하는 대상을 말합니다. 하나의 개체는 자신의 특성을 가집니다. 이런 특성을 개체의 속성이라고 합니다. 좀 더 데이터베이스적인 용어를 빌려서 말한다면, ‘하나의 개체는 하나 이상의 속성을 가지고 있다’라고 할 수 있습니다.

### “ 관계형 데이터베이스 ”

관계형 데이터베이스는 모든 데이터들을 테이블과 같은 형태로 나타내어 저장하는 데이터베이스입니다. 즉 행과 열로써 데이터를 표현하는 데이터베이스입니다. 결국 우리가 사용하는 ‘표’의 개념을 이용한 데이터베이스입니다.

### “ 주키(Primary Key) ”

관계형 데이터베이스에서 주키는 아주 중요합니다. 그림의 책 테이블의 경우에는 그림 상으로는 책이름이 주키가 되고(중복된 책이름이 없는 경우), 도서관 직원 테이블의 경우의 주키는 직원 하나하나를 구별할 수 있는 칼럼(예를 들면 직원번호 정도가 될 것입니다.)을 주키로 정하고, 도서관 이용 회원 테이블의 경우는 회원의 ID를 주키로 정하면 될 것입니다.

- 테이블 : 도서관에 보관되어 있는 책은 그 수가 아주 많다는 것을 쉽게 알 수 있습니다. 그리고 그 각각의 책들은 하나하나의 속성이 모인 레코드로 표현할 수 있다는 것을 알았습니다. 각각의 책들은 책꽂이에 꽂혀져 있습니다. 이 책꽂이가 바로 테이블이 됩니다.
- 데이터베이스 : 책, 책꽂이가 있는 거대한 공간이 바로 도서관인데, 이 도서관이 바로 데이터베이스가 되는 것입니다.

도서관에는 소장된 책, 도서관 직원, 도서관 이용 회원들을 위한 테이블이 있습니다. 그 중 책을 나타내는 테이블에 대한 개념을 나타내었습니다. 책제목, 저자명, 출판사를 칼럼으로 표현할 수 있는 책 테이블은 도서관에 소장되어 있는 책 권수만큼의 레코드를 가지게 됩니다.

또한 도서관 직원, 도서관 이용회원에 대한 테이블도 그들을 표현할 수 있는 속성을 가질 수 있습니다. 이렇게 해서 도서관에 관계되는 여러 테이블이 모여서 전체적으로 도서관 데이터베이스를 구성하게 되는 것입니다. 테이블은 복수 개의 레코드로 구성이 되는데 그들 각각의 레코드를 구별하기 위해 쓰이는 주키(Primary Key)라는 개념이 있어 복수 개의 레코드를 구별할 수 있다고 했습니다. 즉 중복되는 책이름을 가진 책들이 있다고 해도 주키 값으로 그들 각각을 구별할 수 있습니다. 그럼에는 책제목으로 각각의 레코드를 구별할 수 있도록 되어 있습니다. 하지만 중복되는 책이름이 있는 경우도 있습니다. 이 경우에는 책 각권이 일련번호를 부여하는 식으로 유일한 주키 값을 배정할 수가 있습니다.

#### 잠깐만!!

- 관계형 데이터베이스에서 나타나는 여러 관계

관계형 데이터베이스는 말 그대로 테이블 간의 관계를 포함하고 있습니다. 이러한 관계에서 나타날 수 있는 경우는

- one to one (1:1) 관계 : 한 테이블에 있는 하나의 데이터는 다른 테이블에 있는 하나의 데이터와 연관

**예** 회원테이블과 주민번호테이블, 이때는 회원테이블에 있는 한 회원이 가지는 주민번호는 주민번호테이블에 있는 하나의 주민번호와 연관이 됩니다.

- one to N (1:N) 관계 : 한 테이블에 있는 하나의 데이터는 다른 테이블에 있는 여러 개의 데이터에 연관됩니다.

**예** 회원테이블과 주문테이블, 이 경우는 한 회원이 주문한 상품은 하나 이상이 될 수 있는 것을 말합니다. 가장 흔한 관계가 이 경우입니다.

- N to N (N:N) 관계 : 복수 개의 데이터는 복수 개의 데이터에 연관이 됩니다. 이 경우는 비정상적인 관계가 됩니다. 그래서 N : N 관계는 1 : N, N : 1의 관계를 가질 수 있게 하나의 테이블을 더 만들어야 합니다.

**예** 회원테이블과 동호회테이블 : 한 회원은 복수 개의 동호회에 해당될 수 있고, 한 동호회는 복수의 회원에 대한 정보를 가질 수 있습니다. 회원 테이블 1 : 동호회테이블 N, 동호회테이블 1 : 회원테이블 N인 경우가 합해지면, N : N 관계가 되어 버립니다. 해결책은 회원테이블 : 회원동호회테이블 : 동호회테이블 식을 구성이 되어서 1 : N : 1의 식으로 테이블이 구성되어야 합니다.

- 데이터형 : 테이블을 구성하는 하나하나의 칼럼들 중에는 앞서 말했던 책제목이나 저자명, 출판사명처럼 문자로 된 부분이 있을 수 있고, 직원테이블에서 각각의 직원의 나이나, 몸무게 등 수치상의 정보가 있을 것입니다. 또한 출판년도처럼 날짜형의 정보가 있을 수 있습니다. 이렇게 문자형태, 숫자형태, 날짜형태에 맞게끔 칼럼의 속성을 지정하는 것입니다. 데이터형의(정수형, 실수형, 문자형, 날짜형)은 다음과 같은 것들이 있습니다.

	데이터형	저장공간 크기	설명 및 특징
숫자형	INT(size)	4 bytes	숫자형 칼럼(정수)
	FLOAT	4 bytes	숫자형 칼럼(실수)
	DOUBLE	8 bytes	숫자형 칼럼(실수)
	REAL	8 bytes	숫자형 칼럼(실수)
날짜형	DATETIME	8 bytes	날짜형 칼럼
	DATE	3 bytes	날짜형 칼럼
	TIMESTAMP	4 bytes	날짜형 칼럼
문자형	CHAR	1~255까지 저장	문자형 칼럼
	VARCHAR	1~255까지 저장	문자형 칼럼
	BLOB	최대길이 65536	문자형 칼럼

## 02-2 회원테이블 만들기

개념을 실제로 Maria에서 구현을 해보도록 하겠습니다. 앞으로는 이 책에서 사용하기 위한 데이터베이스 이름은 'mydb'로 정하고, 이 책의 후반부에 나오는 실제 예제에서 사용하기 위한 데이터베이스로서 mydb에 들어갈 테이블은 회원테이블이 있습니다. 일반적으로 사이트에 가입하는 회원에 대한 내용을 저장하기 위한 회원테이블을 먼저 만들어 보기로 하겠습니다.

우선 Maria 데이터베이스 서버를 가동하는 것이 첫 번째로 해야 할 일입니다. 데이터베이스에 연동이 되는 프로그램을 만들기 위해서는 반드시 데이터베이스를 가동해야 합니다. 그런 다음 mydb라는 이름의 데이터베이스를 생성하도록 하겠습니다.

※ 실제 개발 시에는 데이터베이스용 서버를 따로 두어 서비스를 하고 있는 경우가 많습니다. 이때도 데이터베이스 서버로 접속을 할 경우 그 서버가 구동중인지를 먼저 확인한 후에 데이터베이스 서버로 접속해야 합니다.

### “ 데이터 모델링 ”

실제 프로젝트에는 데이터 모델이 필수입니다. 데이터 모델이 없이 프로젝트가 진행될 수는 없습니다. 그만큼 모델링은 프로젝트에서 중요하고, 많은 부분을 차지하고 있습니다. 이 모델링 단계를 거쳐서 각 개체(테이블) 간의 관계가 정의된 다이어그램이 작성되어집니다. 그런 다음에 이 모델을 토대로 프로그램 작업이 수행이 되어집니다. 프로그램이 아무리 중요하다고 해도 모델이 없다면, 프로젝트는 수행될 수 없는 정도입니다.

### (1) 데이터베이스 생성에서 테이블 생성까지

회원정보를 담아둘 회원테이블을 생성해야 할 차례입니다. 회원테이블의 구성에 대해서 생각해 보겠습니다. 여러분들이 사이트에서 회원가입을 할 때 입력했던 그런 내용들이 회원테이블의 속성이 될 수 있습니다. 우선 각각의 회원들이 사용할 아이디가 있어야 하고, 패스워드, 회원이름, 주민등록번호, 이메일, 전화번호, 주소, 직업 등이 있을 수 있습니다. 물론 상세한 정보를 담아두기 위해서 이보다 더 많은 속성을 두어서 회원에 대한 정보를 담을 수 있습니다.

결혼여부, 결혼기념일, 관심분야, 취미 등등 여러 가지가 있을 수 있습니다. 여기서는 기본 정보만을 담아 두도록 하겠습니다. 이렇게 하면 위에 말했던 속성을 가진 테이블이 구성됩니다. 테이블을 그림으로 나타내어 보겠습니다.

회원 테이블		tblRegister
회원아이디		id: VARCHAR(20)
패스워드		pwd: VARCHAR(20)
이름		name: VARCHAR(20)
주민등록번호앞자리		num1: CHAR(6)
주민등록번호뒷자리		num2: CHAR(7)
이메일		email: VARCHAR(30)
전화번호		phone: VARCHAR(30)
우편번호		zipcode: CHAR(5)
주소		address: VARCHAR(30)
직업		job: VARCHAR(30)

▲ 회원테이블 표

그럼은 회원테이블에 들어갈 속성들을 나타내는 그림입니다. 왼쪽의 '회원테이블'은 논리적으로 테이블에 들어갈 속성을 나타낸 부분이고, 오른쪽의 'tblRegister'는 실제 데이터베이스에 만들어진 속성들과 각각의 속성들이 가질 크기, 혹은 길이를 정의한 실제의 테이블입니다.

데이터베이스를 생성하기 위해서 필요한 질의문은 다음과 같습니다.

#### 데이터베이스 생성

```
CREATE DATABASE [DATABASE_NAME]
```

설명 : [DATABASE\_NAME]의 부분에 만들어질 데이터베이스 이름을 적습니다.

예 CREATE DATABASE myDB

#### TIP

Maria 서버는 리눅스, 유닉스, 윈도우용이 있습니다. 이때 Maria는 내부적으로 동작하는 방식은 운영체제와 연관이 되어 있습니다. 리눅스(라즈베리 파이)의 운영체제는 대소문자를 구별하기 때문에 Maria에서 사용되는 쿼리문(Query)들은 대소문자를 구별합니다.

#### TIP

database 삭제 명령

```
DROP DATABASE [DATABASE_NAME]
```

이 명령은 데이터베이스를 삭제하는 명령입니다. 삭제된 데이터베이스는 복구가 되지 않습니다. 이 명령을 사용할 때는 조심해서 사용하시길 바랍니다.

```
USE [DATABASE_NAME]
```

설명 : [DATABASE\_NAME] 사용할 데이터베이스 이름을 입력합니다.

#### TIP

실제로 데이터베이스를 사용하기 위한 명령어는 아주 많은 종류가 있습니다. 그런 명령어들을 잘 기억하고 어떻게 사용하는지에 대해서 잘 알아두면 데이터베이스 사용에 있어 부담을 훨씬 줄일 수 있습니다. 하지만 이 책에서는 데이터베이스에 관해 기초적인 지식만을 다루도록 하겠습니다. 데이터베이스에 대해 자세히 알고 싶다면 다른 서적이나 관련 자료를 참고하면 됩니다.

이 명령은 여러 데이터베이스를 사용할 때 필요한 명령입니다. 다른 데이터베이스에 있는 테이블을 사용할 때 USE 명령을 사용해서 사용할 테이블이 있는 데이터베이스를 먼저 지정하고 그 다음 테이블을 사용하며 됩니다.

현재는 mydb란 이름의 데이터베이스만 존재할 뿐 다른 테이블은 없습니다. use mydb 또는 mydb로 로그인을 했다면 현재 사용하는 데이터베이스는 mydb란 데이터베이스입니다.

**01** 이제 'tblRegister'란 이름을 가진 테이블을 생성하도록 하겠습니다. 라즈베리 파이 터미널에서 다음 질의문을 입력하고 난 뒤 실행을 시킵니다. 만약 오타나 잘못된 부분이 있을 경우 에러가 발생하므로 주의해서 입력하길 바랍니다.

#### 테이블 생성 명령

```
CREATE TABLE [TABLE_NAME](
    [COL_NAME1 TYPE][PRIMARY KEY][NOT NULL/NULL],
    [COL_NAME2 TYPE2],
    [COL_NAME1 TYPE3]...)
```

설명 : [TABLE\_NAME] 테이블 이름을 입력합니다.

[PRIMARY KEY]는 만들어질 테이블의 키를 설정합니다.

[NOT NULL/NULL]테이블의 속성(칼럼)에 들어갈 값 중에 NULL 값을 허용/비허용을 설정하는 부분입니다.

```
01 : CREATE TABLE tblRegister(
02 :     id          VARCHAR(20) NOT NULL,
03 :     pwd         VARCHAR(20) NOT NULL,
04 :     name        CHAR(6) NULL,
05 :     num1        CHAR(6) NULL,
06 :     num2        CHAR(7) NULL,
07 :     email       VARCHAR(30) NULL,
08 :     phone       VARCHAR(30) NULL,
09 :     zipcode     CHAR(5) NULL,
10 :     address     VARCHAR(60) NULL,
11 :     job         VARCHAR(30) NULL
12 : );
```

\* 아이디와 패스워드 이외에도 꼭 필요한 부분이 있다면 NOT NULL로 설정해도 됩니다. NOT NULL로 설정된 칼럼은 저장될 값이 꼭 있어야 합니다.

**02** : 회원의 아이디를 저장할 칼럼입니다. ID가 들어갈 공간은 20자 정도면 저장할 수 있기 때문에 VARCHAR(20)으로 설정하고, 아이디는 꼭 필요하기 때문에 NOT NULL로 설정합니다.

**03** : 패스워드 또한 20자 정도로 설정하고, 꼭 필요하기 때문에 NOT NULL로 설정합니다.

**04** : 이름은 최대 6자로 설정합니다. 회원테이블에서 꼭 필요한 부분이 아이디와 패스워드입니다. 나머지는 저장될 값이 없어도 가능한 NULL로 설정합니다.

**05** : 주민등록번호 중 앞부분 6자리가 들어갈 부분입니다.

**06** : 주민등록번호 중 뒷부분 7자리가 들어갈 부분입니다.

**07** : 이메일 주소가 들어갈 자리는 30자로 설정합니다. VARCHAR 타입의 경우는 실제로 들어갈 값이 설정된 크기보다 클 때는 실제 값만큼 길이가 증가합니다. 30자가 넘는 이메일 주소가 있을 경우를 대비해서 VARCHAR로 설정합니다.

**08** : 전화번호가 저장될 칼럼을 설정합니다.

**09** : 우편번호 '12345' 5자리가 저장될 칼럼을 설정합니다.

**10** : 주소를 저장할 칼럼을 설정합니다.

**11** : 직업을 저장할 칼럼을 설정합니다.

**TIP**

데이터베이스 이름, 테이블 이름, 한 테이블 안에 칼럼 이름은 중복이 되어서는 안 됩니다. 중복되는 부분이 있다면 에러가 발생합니다. 만약 에러가 발생하였다면 중복되는 이름이 있는지 확인하기 바랍니다.

**TIP**

리눅스용 Maria 데이터베이스 서버에서는 대소문자를 구별합니다. 또한 여러 개의 칼럼의 경우 각 칼럼을 구분하기 위해 '.'가 있어야 하고, 그리고 그 외 ')' 부분에서도 글자가 누락되는 경우에 질의가 수행이 되질 않습니다. 이러한 부분에 유의해서 질의를 작성합시다. 그러나 쿼리문에서 사용하는 예약어는 대소문자 구분을 하지 않습니다. SELECT나 select는 같습니다.

**TIP**

데이터 타입들에 있어서 CHAR형과 VARCHAR형은 다음 표와 같은 다른 점이 있습니다.

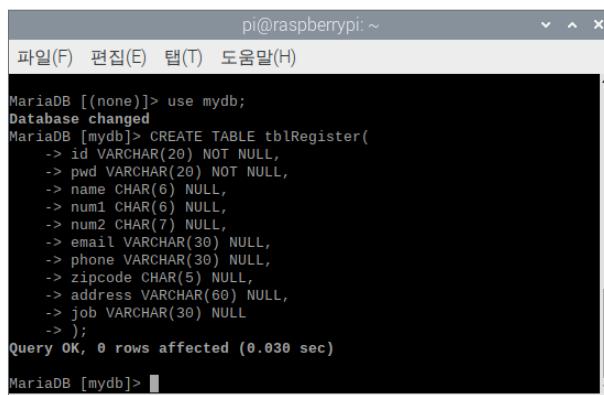
문자열	CHAR(4)		VARCHAR(4)	
	저장된 값	저장시 사용되는 공간	저장된 값	저장시 사용되는 공간
''	''	4 BYTES	''	1 BYTES
'123'	'123'	4 BYTES	'123'	4 BYTES
'12345'	'1234'	4 BYTES	'1234'	5 BYTES

▲ CHAR와 VARCHAR형의 비교

CHAR의 경우 저장될 값(문자, 문자열)의 길이가 확실한 경우에 설정합니다. 우편번호처럼 그 길이가 고정된 값일 경우에 사용하면 됩니다. 이 칼럼에 값이 저장될 때 남는 공간은 칼럼 길이에 맞게 공백이 채워집니다.

VARCHAR는 실제로 저장될 값의 길이(문자열 길이)가 변동적일 경우에 사용합니다. 주어진 길이보다 적은 길이의 문자열이 올 때 사용되는 공간은 <문자열크기 + 1 BYTES>만 사용됩니다. 결국, 실제 저장될 값의 길이에 따라 저장되는 형태가 다르다는 것입니다. 하지만 주의할 점은 CHAR나 VARCHAR 둘 다 처음에 설정한 길이보다 길이가 더 긴 값이 들어올 경우 넘어가는 길이만큼 뒷부분의 문자는 제거되어 저장된다는 것을 꼭 기억합니다.

## 02 다음과 같은 명령어로 입력하고 실행하면 테이블이 생성될 것입니다.



```
pi@raspberrypi: ~
파일(F) 편집(E) 템(T) 도움말(H)
MariaDB [(none)]> use mydb;
Database changed
MariaDB [mydb]> CREATE TABLE tblRegister(
-> id VARCHAR(20) NOT NULL,
-> pwd VARCHAR(20) NOT NULL,
-> name CHAR(6) NULL,
-> num1 CHAR(6) NULL,
-> num2 CHAR(7) NULL,
-> email VARCHAR(30) NULL,
-> phone VARCHAR(30) NULL,
-> zipcode CHAR(5) NULL,
-> address VARCHAR(60) NULL,
-> job VARCHAR(30) NULL
-> );
Query OK, 0 rows affected (0.030 sec)

MariaDB [mydb]>
```

▲ tblRegister 테이블 생성

## (2) 생성된 테이블 관리하기

데이터베이스 생성 후에 생성한 데이터베이스에 테이블을 생성했습니다. 현재의 데이터베이스에 어떤 테이블들이 있는가 하는 명령은 SHOW TABLES입니다. 이 명령어를 실행시켜보면 현재 데이터베이스에 생성되어 있는 테이블을 볼 수 있습니다.

### 테이블 전체 보기 명령

```
SHOW TABLES;
```

이상 없이 테이블이 생성되었다면 다음의 명령을 사용하여 테이블 속성을 확인합니다.

### 테이블 속성 보기 명령

```
DESC [TABLE_NAME]
```

설명 : [TABLE\_NAME]에 원하는 테이블 이름을 입력합니다.

만약 테이블을 생성했는데 데이터 타입을 바꾸고자 할 때, 특정 칼럼에 주키를 부여할 때에 사용되는 명령어에 대해서 알아보겠습니다.

01 tblRegister 테이블의 속성을 보기 다음과 같은 명령어를 입력합니다.

A screenshot of a terminal window titled 'pi@raspberrypi: ~'. The window shows the MySQL command-line interface. The user has run the command 'use mydb;' to select the database. Then, they have run the command 'CREATE TABLE tblRegister(' followed by a long list of columns with their respective data types and NULLability. The command ends with ')';'. After executing the command, the terminal displays 'Query OK, 0 rows affected (0.030 sec)'. At the bottom of the terminal window, there is a status bar with the text 'MariaDB [mydb]>'. A blue arrow points from the text '▶ tblRegister 테이블 속성' to the 'DESC' command in the terminal history.

### 테이블구조 변경 명령(특정 칼럼에 키를 부여할 때)

```
ALTER TABLE [TABLE_NAME]
```

```
ADD PRIMARY KEY(COL_NAME);
```

설명 : [TABLE\_NAME] 구조를 변경할 테이블 명을 입력합니다.

COL\_NAME 구조를 변경할 칼럼명을 입력합니다.

위의 명령을 사용해서 tblRegister 테이블 칼럼 중에 'ID'라는 칼럼을 주키로 설정하겠습니다. 앞에서 설명한 것처럼 주키란 테이블 안에 있는 여러 행들과 구분하기 위한 것이라고 했습니다.

```
001    ALTER TABLE tblRegister  
002        ADD PRIMARY KEY (ID);
```

### TIP

#### ALTER 명령

ALTER 명령은 새로운 칼럼을 추가하거나 삭제 또는 데이터 형을 변경하고자 할 때 사용하는 명령어입니다. 만약 칼럼 하나하나에 대한 변경이 불가능하다면 테이블을 삭제하고 다시 만들어야 하는 불편함이 있겠죠?

참고로 테이블을 삭제하는 명령어는 DROP TABLE [TABLE\_NAME]입니다. 삭제 명령을 사용할 경우 삭제한 테이블은 복구가 불가능하기 때문에 잘 생각해서 사용하시기 바랍니다.

02 다음과 같은 명령어로 입력하고 실행하면 tblRegister 테이블의 id 컬럼의 속성이 주키 설정이 됩니다.

```
pi@raspberrypi: ~  
파일(F) 편집(E) 템(T) 도움말(H)  
MariaDB [mydb]> ALTER TABLE tblRegister  
    -> ADD PRIMARY KEY(ID);  
Query OK, 0 rows affected (0.279 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
MariaDB [mydb]> desc tblRegister;  
+-----+-----+-----+-----+-----+  
| Field | Type  | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| id   | varchar(20)| NO  | PRI | NULL   |       |  
| pwd  | varchar(20)| NO  |     | NULL   |       |  
| name | char(6)   | YES |     | NULL   |       |  
| num1 | char(6)   | YES |     | NULL   |       |  
| num2 | char(7)   | YES |     | NULL   |       |  
| email | varchar(30)| YES |     | NULL   |       |  
| phone | varchar(30)| YES |     | NULL   |       |  
| zipcode | char(5) | YES |     | NULL   |       |  
| address | varchar(60)| YES |     | NULL   |       |  
| job   | varchar(30)| YES |     | NULL   |       |  
+-----+-----+-----+-----+-----+  
10 rows in set (0.003 sec)  
  
MariaDB [mydb]>
```

▲ tblRegister 테이블의 id 속성을 primary key로 변경

### (3) 데이터를 조회하는 명령

테이블 안에 있는 데이터를 조회하기 위한 명령입니다.

### TIP

#### 데이터 조회 명령

- ① select \* from table\_name
  - ② select col\_nm1, col\_nm2, col\_nm3... from table\_name
- ①, ②는 실행결과가 똑같습니다. 실제로 어떤 테이블에 있는 레코드를 조회하기 위한 명령어로 ①, ② 질의를 사용해서 실행시켜 본 결과는 동일합니다. 하지만, 실제로 ② 질의가 수행이 빠릅니다. 프로그램 작성 시 전체 칼럼을 뜻하는 '\*'보다, 칼럼 이름을 직접 기입하는 것이 더 좋습니다.

#### 데이터 조회 명령

```
SELECT * FROM [TABLE_NAME]
```

설명 : '\*'은 모든 칼럼을 불러온다는 뜻입니다. 필요한 칼럼만 불러올 경우에는 칼럼명을 입력하고 중간에 ','를 사용해서 구분하면 됩니다.

[TABLE\_NAME]에 조회대상 테이블이름을 입력합니다.

예) SELECT ID, PASSWD, JOB, FROM tblRegister

```
001      SELECT * FROM tblRegister;
```

위 명령을 실행하면 현재 테이블 안에는 아무런 데이터가 없기 때문에 칼럼 이름만 표시되고 안의 데이터는 공백으로 표시됩니다.

#### (4) 데이터 입력 명령

테이블에 실제 데이터를 입력하기 위한 명령입니다.

##### 데이터 입력 명령

```
INSERT INTO [TABLE_NAME] (COL_NAME1, COL_NAME2...)  
    VALUES(INPUT_VALUE1, INPUT_VALUE2...);
```

설명 : [TABLE\_NAME] 데이터를 입력할 대상 테이블 이름을 입력합니다.

[COL\_NAMEn] 데이터가 입력될 칼럼 이름을 입력합니다.

[INPUT\_VALUEn] 실제 데이터를 입력합니다.

VALUES 절 안에 실제로 입력될 데이터들이 있는데 이들 중 문자(문자열)의 경우는 ('문자열') 형식으로 감싸야 합니다. 하지만 숫자형의 경우는 "으로 감싸면 안 됩니다.

INSERT 명령에서 중요한 것은 칼럼이름과 대응한 데이터들의 개수와 순서가 같아야 하는 것입니다. 만약 모든 칼럼에 들어갈 데이터라면 INSERT INTO TABLE\_NAME 다음에 칼럼명은 생략하도록 됩니다. 이때 VALUES 다음에 나오는 데이터들은 전체 칼럼 개수와 같아야 합니다.(전체 칼럼에 데이터를 입력할 경우에는 테이블 이름 다음에 칼럼 명을 적는 부분을 생략해도 됩니다.)

또한, 전체 칼럼에 입력하는 경우가 아닌 몇 개의 칼럼에 대해서 데이터를 저장할 경우에는 칼럼 명을 분명히 적어주어야 합니다. 이때 칼럼 개수와 순서에 맞게 데이터를 입력해야 합니다. INTO 명령어는 생략 가능합니다. 구 버전에서 사용했던 명령어이지만 지금은 생략해도 무방합니다.

아래의 명령을 입력하고 나서 실행합니다.

```
INSERT INTO  
    tblRegister(ID, PWD, NAME)  
    VALUES('rorod', '1234', '이경미')  
ID, PASSWD, NAME의 개수, 순서에 맞게 'rorod', '1234', '이경미'처럼 입력해야 합니다.
```

다음은 INSERT 명령을 통해서 필요한 데이터를 입력하고 다시 테이블에 있는 데이터를 조회하기 위해서 SELECT 명령을 사용한 것입니다. Maria은 각 명령어가 끝나면 뒤에 ';'를 붙여서 명령의 끝을 알려줘야 합니다.

```

001      INSERT INTO
002          tblRegister(ID, PWD, NAME, NUM1, NUM2, EMAIL, PHONE,
003              ZIPCODE, ADDRESS, JOB)
004      VALUES('rorod', '1234', '이경미', '1234567', '1234567',
005          'rorod@jpsstudy.co.kr', '010-1111-1111', '1234', '부산 연제구',
006          '프로그래머');

```

```

pi@raspberrypi: ~
파일(F) 편집(E) 템(T) 도움말(H)
MariaDB [mydb]> INSERT INTO
->     tblRegister(ID, PWD, NAME, NUM1, NUM2, EMAIL, PHONE,
->     ZIPCODE, ADDRESS, JOB)
->     VALUES('rorod', '1234', '이경미', '1234567', '1234567',
->     'rorod@jpsstudy.co.kr', '010-5555-7890', '1234', '부산 연제구',
->     '프로그래머');
Query OK, 1 row affected (0.008 sec)

MariaDB [mydb]> select * from tblRegister;
+-----+-----+-----+-----+-----+-----+
| id   | pwd  | name | num1 | num2 | email        | phone
| zipcode | address | job |
+-----+-----+-----+-----+-----+-----+
| rorod | 1234 | 이경미 | 123456 | 1234567 | rorod@jpsstudy.co.kr | 010-5555-
7890 | 1234 | 부산 연제구 | 프로그래머 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)

MariaDB [mydb]>

```

▲ 테이블에 데이터 입력 및 출력

### TIP

#### Maria 터미널에서 실행할 때

도스창의 형태로 제공되는 Maria 터미널에서 명령을 실행시킬 때는 명령문 끝에 ';'를 반드시 입력해야 합니다. 앞에서 잠깐 언급했는데 ';'는 한번에 실행할 명령의 단위로 생각하면 됩니다.

터미널에서 실행할 때는 ';'를 붙이지 않으면 다른 행으로 넘어가면서 ';' 입력을 기다립니다.

그림을 보면, Maria 프롬프트에서 ';'이 입력될 때까지 계속 입력을 기다리게 됩니다. ';'가 입력이 되면 질의를 수행합니다. 터미널을 통한 접속 중 접속 해제를 위한 명령은 'quit' 또는 'exit'이며 명령어를 입력하면 터미널에서 빠져나옵니다.

## (5) 데이터 변경 명령

데이터를 수정할 필요가 있는 경우에 사용되는 명령입니다.

### 데이터 변경 명령

UPDATE [TABLE\_NAME] SET[COL\_NAME]=[VALUE1],.... WHERE [조건];

설명 : [TABLE\_NAME] 데이터를 바꿀 칼럼을 가지고 있는 테이블 명을 입력합니다.

[COL\_NAME]은 데이터를 바꿀 칼럼 명을 지정합니다.

[VALUE]은 실제 변경할 데이터를 입력합니다.

[조건] WHERE문 다음에 나오는 조건은 데이터를 바꿀 조건을 지정하는 것입니다. 해당 조건이 맞는 행만 값이 바뀌게 됩니다. 만약 WHERE문이 주어지지 않은 경우라면 테이블의 모든 행에 있는 칼럼 값이 바뀌게 됩니다.

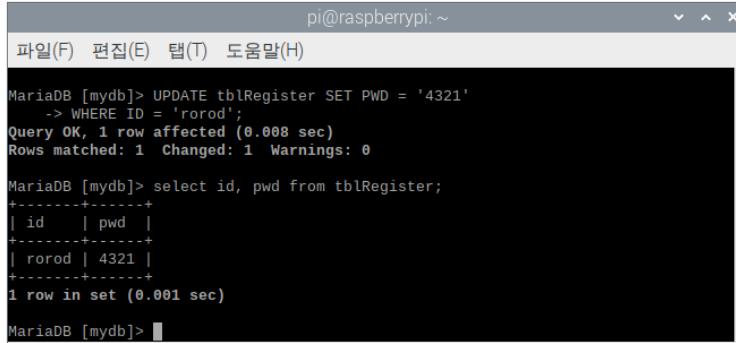
※ update문의 where절에서 지정한 조건에 맞는 행을 찾아서 값을 변경합니다. 이때 where절에서 정의한 조건에 해당되는 행이 없을 경우 update는 아무런 행에 영향을 미치지 않습니다.

예) UPDATE tblRegister SET PWD='4321' WHERE ID='simba'

ID가 'simba'인 행이 없기 때문에 UPDATE는 아무런 행에도 영향을 미치지 않습니다.

```
001 UPDATE tblRegister SET PWD = '4321'  
002 WHERE ID='rorod';
```

이 명령은 ID가 'rorod'인 행에서 PWD 칼럼의 값을 '4321'로 변경한다는 뜻의 명령어입니다.



```
pi@raspberrypi: ~  
파일(F) 편집(E) 템(T) 도움말(H)  
MariaDB [mydb]> UPDATE tblRegister SET PWD = '4321'  
-> WHERE ID = 'rorod';  
Query OK, 1 row affected (0.008 sec)  
Rows matched: 1 Changed: 1 Warnings: 0  
  
MariaDB [mydb]> select id, pwd from tblRegister;  
+----+----+  
| id | pwd |  
+----+----+  
| rorod | 4321 |  
+----+----+  
1 row in set (0.001 sec)  
  
MariaDB [mydb]>
```

▲tblRegister 테이블에 데이터 수정

## (6) 데이터 삭제 명령

저장되어 있는 데이터 중에 더 이상 필요 없거나, 삭제할 필요가 있는 데이터를 삭제하기 위한 명령입니다.

### 데이터 삭제 명령

DELETE FROM [TABLE\_NAME] WHERE [조건];

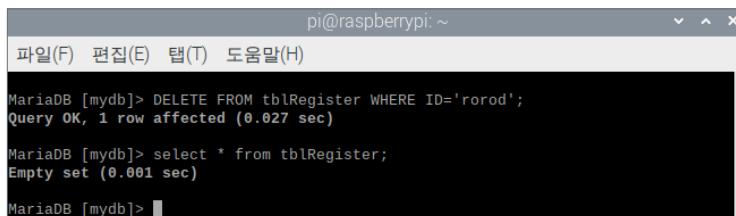
설명 : [TABLE\_NAME] 데이터를 삭제할 필요가 있는 테이블 명을 입력합니다.

[조건] WHERE절에 있는 조건문은 이 조건에 만족하는 행에 대해서 데이터를 삭제하기 위한 조건입니다. 만약 WHERE문이 주어지지 않는다면 테이블에 입력되어 있는 모든 행들이 삭제가 됩니다.

```
001 DELETE FROM tblRegister WHERE ID='rorod';
```

\* delete문에서 where절에 정의한 조건에 맞는 행이 없을 경우 삭제되는 행은 없습니다. 이는 다른 질의문 select, update문에서 where 절에 정의한 조건에 해당되는 행이 없는 경우 행을 조회하거나, 변경하는 행이 없는 것과 동일합니다.

현재 tblRegister라는 테이블에는 하나의 행(id가 'rorod'인 행)만이 저장되어 있습니다만, 실제 테이블에서 아주 많은 행들이 저장되어 있을 때, 그들 중 id가 'rorod'인 행을 삭제한다는 뜻입니다.



```
pi@raspberrypi: ~  
파일(F) 편집(E) 템(T) 도움말(H)  
MariaDB [mydb]> DELETE FROM tblRegister WHERE ID='rorod';  
Query OK, 1 row affected (0.027 sec)  
  
MariaDB [mydb]> select * from tblRegister;  
Empty set (0.001 sec)  
  
MariaDB [mydb]>
```

▲tblRegister 테이블에 id가 rorod를 삭제

## 03 \_ 파이썬과 Maria 데이터베이스 서버 Connection

라즈베리 파이에 설치한 Maria 데이터베이스 서버와 파이썬을 연결하고 tblRegister에 입력한 레코드를 출력하는 예제를 해보도록 보겠습니다. 앞에서 삭제를 했기 때문에 INSERT문은 다시 한번 실행을 하시기 바랍니다.

### (1) 파이썬에 설치하기

파이썬과 Maria 데이터베이스 서버를 연결하기 위해서는 먼저 파이썬 버전 체크와 파이썬 모듈을 설치해야 합니다. 다음과 같은 명령어를 Terminal에 실행하시기 바랍니다.

```
python -V
```

만약 버전 체크 후에 2.x 버전이면 3.x 버전으로 연결을 해야 합니다. 3.x 버전이면 할 필요가 없습니다.

```
ls -al /usr/bin/python  
sudo ln -f /usr/bin/python3.7 /usr/bin/python
```

```
pi@raspberrypi:~ $ python -V  
Python 2.7.16  
pi@raspberrypi:~ $ ls -al /usr/bin/python  
lrwxrwxrwx 1 root root 7 3월 5 2019 /usr/bin/python -> python2  
pi@raspberrypi:~ $ sudo ln -f /usr/bin/python3.7 /usr/bin/python  
pi@raspberrypi:~ $ python -V  
Python 3.7.3  
pi@raspberrypi:~ $
```

▲ 파이썬 버전 체크

```
python3 -m pip install PyMySQL
```

```
pi@raspberrypi:~ $ python3 -m pip install PyMySQL  
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting PyMySQL  
  Using cached https://files.pythonhosted.org/packages/ed/39/15045ae46f2a123019aa0968dfcba0396c161c20f855f11dea6796bc当地95/PyMySQL-0.9.3-py3-none-any.whl  
Installing collected packages: PyMySQL  
Successfully installed PyMySQL-0.9.3  
pi@raspberrypi:~ $
```

▲ 파이썬 모듈 설치

### (2) 파이썬과 Maria 연결에 필요한 설정

먼저 mysql 데이터베이스에 있는 user 테이블에 root 계정의 plugin 값을 그림과 같이 수정을 해야 합니다. 이렇게 빙값으로 설정을 해야지 파이썬에서 연결이 가능합니다.

user 테이블이 mysql 데이터베이스 안에 있는 테이블이기 때문에 먼저 use mysql을 실행하고 UPDATE을 해야 합니다.

```

pi@raspberrypi: ~
파일(F) 편집(E) 템(T) 도움말(H)
MariaDB [mysql]> UPDATE user SET plugin='';
Query OK, 1 row affected (0.006 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [mysql]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [mysql]>

```

▲ plugin 값 수정

그리고 Maria 데이터베이스 서버에서 root 계정 비밀번호 설정을 합니다. Maria 데이터베이스 서버 설치 시 root 계정의 비밀번호는 빈값으로 설정이 되어 있습니다. 그러나 파이썬 코드에서는 반드시 비밀번호가 입력되어 되어야 하므로 root 계정의 비밀번호는 '1234'로 설정합니다. 그리고 변경된 사항들을 저장합니다.

```

pi@raspberrypi: ~
파일(F) 편집(E) 템(T) 도움말(H)
MariaDB [mydb]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> UPDATE user SET password=password('1234') WHERE user='root';
Query OK, 0 rows affected (0.001 sec)
Rows matched: 1 Changed: 0 Warnings: 0

MariaDB [mysql]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [mysql]>

```

▲ root 계정의 비밀번호 변경

### (3) 파이썬과 Maria 데이터베이스 서버 연결 예제

**01** 다음과 같이 코드를 작성하고 저장합니다.

실습 파일 : webapps/ch07/mariaConnection.py

```

01 : import pymysql
02
03 : db = pymysql.connect(host='localhost', user='root', password='1234',
    Maria 연결에 필요한 host랑 id 및 비밀번호를 세팅을 합니다.
04 : db='mydb', charset='utf8')
    연결할 database명과 한글 출력을 위한 세팅을 합니다.
05 : cur = db.cursor()
06 : cur.execute("SELECT * FROM tblRegister")
    query을 선언하고 실행 합니다.
07 : rows = cur.fetchall()
08 : print(rows)
    실행한 query문의 결과값을 출력합니다.
09 : db.close()

```

The screenshot shows the VS Code interface. On the left is a code editor window titled 'mariadbConnection.py' containing the following Python code:

```
1 import pymysql
2
3 db = pymysql.connect(host='localhost', user='root', password='1234',
4 db='mydb', charset='utf8')
5 cur = db.cursor()
6 cur.execute("SELECT * FROM tblRegister")
7 rows = cur.fetchall()
8 print(rows)
9 db.close()
```

On the right is a terminal window titled '1: bash' showing the command-line output of running the script:

```
pi@raspberrypi:~/webapps/ch07 $ python -V
Python 3.7.3
pi@raspberrypi:~/webapps/ch07 $ python mariadbConnection.py
([('rorod', '1234', '이경미', '123456', 'rorod@jpsstudy.co.kr', '010-5555-7890', '123
4', '부산 연제구', '프로그래머')])
```

▲ VScode에서 mariadbConnection.py 입력하고 터미널에서 실행

mydb 데이터베이스에 생성된 tblRetgister 테이블에 저장된 레코드가 화면에 출력이 됩니다. 현재는 한 개의 레코드만 있기 때문에 하나만 출력되지만 여러 개의 레코드가 저장이 되었다면 모든 레코드가 다 출력이 됩니다.

# Raspberry Pi

블루투스 비콘에 대해 알아보고 라즈베리 파이 4를 별도의 추가 없이 비콘으로 변경하여 스마트폰에  
비콘스캐너 앱을 설치한 후 라즈베리 파이 근처에 도달했을 때 비콘의 기능을 활용해 라즈베리 파이의  
웹서버로 접속해 GPIO를 제어하는 방법을 알아보겠습니다.

CHAPTER  
**08**

# 블루투스 비콘(Beacon) 사용하기

- 01 \_ 블루투스(Bluetooth)와 비콘(Beacon)이란?
- 02 \_ 비콘(Beacon)이란?
- 03 \_ 라즈베리 파이 4를 블루투스 비콘으로 바꾸기

## 01 \_ 블루투스(Bluetooth)와 비콘(Beacon) 이란?

---

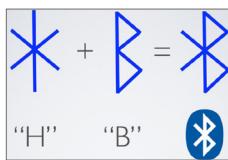


▲ 블루투스 로고

블루투스(Bluetooth)는 근거리 무선 통신 기술의 표준으로 무선 마우스, 키보드를 비롯해, 스마트 폰, 태블릿, 스피커 등에서 문자 및 음성 정보를 무선통신을 통해 주고 받는 용도로 채용되고 있습니다. 블루투스는 수 미터에서 수십 미터 정도의 거리를 둔 정보기기 사이에, 전파를 이용해서 간단한 정보를 교환하는데 사용됩니다.

1994년에 스웨덴의 통신 장비 회사인 에릭슨이 최초로 개발을 시작하고 곧이어 블루투스 SIG(Bluetooth Special Interest Group)라는 단체가 결성되어 본격적인 개발에 들어갔으며 1999년에 공식발표 되었습니다. 초기의 블루투스는 유선 케이블 통신인 RS-232를 무선으로 대체하기 위해 개발되었고 2.4~2.485 GHz 주파수 대역을 사용 합니다. 블루투스는 와이파이에 비해서 저전력으로 좁은 범위의 통신에 적합하게 설계 되었습니다.

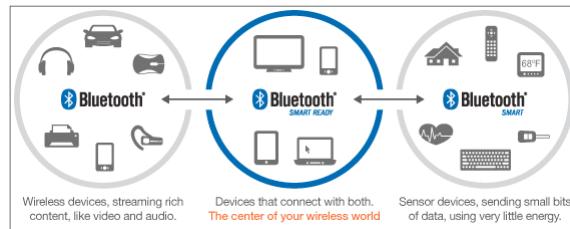
### 01-1 블루투스 이미지의 유래



▲ 블루투스 로고의 유래

블루투스의 로고는 북유럽의 룬 문자로 10세기경 처음으로 덴마크와 노르웨이를 통일한 하랄드 블라톤 국왕의 이름의 앞글자 'H' 와 'B'를 따서 해당하는 룬 문자를 결합한 모양에서 유래 합니다. 블루투스의 개발자는 블라톤 왕이 북유럽을 통일 했듯이 블루투스 기술로 전자제품의 무선구격을 통일시키겠다는 의미로 '블루투스'라고 이름 지었다고 합니다.

## 01-2 블루투스 클래식과 BLE



▲ 블루투스 4.0

블루투스 기술은 20년 전에 시작되어서 많은 발전이 있었습니다. 블루투스 4.0 버전부터 클래식 블루투스(Classic Bluetooth), 고속 블루투스(Bluetooth high speed) 와 함께 블루투스 LE (BLE) 가 발표 되었습니다.

기존의 블루투스 프로토콜을 그대로 사용하는 것이 클래식 블루투스이고 거기에 고속 전송을 지원하는 것이 고속 블루투스입니다. BLE는 기존의 클래식 블루투스와 호환이 되지 않지만, 클래식 블루투스에 비해 전력소모를 매우 적게 만든 저전력 (LE, Low Energy) 기술입니다. 연결이 되지 않을 때 절전 모드를 유지하고 대기 상태에 있다가 다시 연결하는 과정을 통해 매우 적은 전력으로 사용이 가능합니다.

일반 블루투스는 많은 데이터를 처리하지만 배터리 소모가 빠른데 비해 BLE는 저용량 데이터를 주로 처리하는 경우에 배터리 하나로 수년간 사용할 수 있도록 설계 되었습니다.

블루투스 4.0 이후부터 BLE가 추가되어 블루투스 스마트와 스마트 레디가 등장합니다.

- 블루투스 (Classsic Bluetooth) : 일반적인 기존의 블루투스 지원 기기
- 블루투스 스마트(Bluetooth SMART) : BLE 기기만 지원 (싱글모드)
- 블루투스 스마트 레디(Bluetooth SMART READY) : 블루투스 기기와 BLE 모두 지원 ( 듀얼모드 )

블루투스 클래식 제품으로는 보통 끊임없이 데이터를 전송하는 멀티미디어 스트리밍 장치들이 이에 해당하고 BLE 제품으로는 저전력으로 소량의 데이터만 전송하는 센서 등이 있습니다.

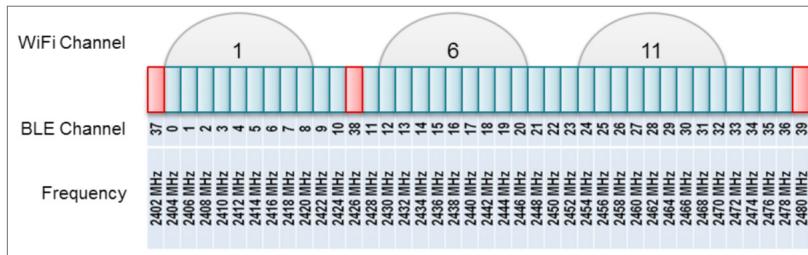
스마트폰, 태블릿, PC, TV 그리고 셋탑 박스 및 게임 콘솔 등은 모두 듀얼모드로 일반 블루투스와 BLE 기기와 모두 연결할 수 있습니다. 반면에 심박 모니터, 스마트 시계, 창문 및 현관 보안 센서, 혈압 밴드 등 많은 IOT 기기들은 BLE만 지원하는 싱글모드이기 때문에 이 기기들을 연결하기 위해서는 블루투스 스마트를 지원하거나 듀얼모드인 블루투스 스마트 레디가 필요합니다.

### 잠깐만!! 블루투스 페어링

블루투스 기기를 서로 연결하여 동작할 수 있도록 설정해주는 과정입니다.

한번 연결된 경우(페어링 성공)는 장치에서 연결 정보를 저장하고 있습니다. 그래서 기기가 전원이 켜져 있고 연결이 가능한 상태라면 자동으로 연결이 이뤄지게 됩니다.

### 01-3 주파수 간섭과 호핑(hopping)



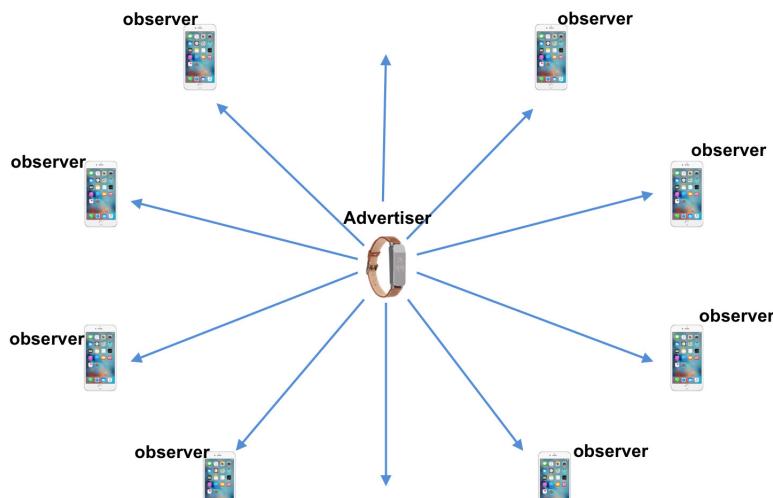
▲ 블루투스 주파수 간섭과 호핑

블루투스의 주파수는 WIFI 통신과 동일한 2.4Ghz 대역을 사용합니다. 따라서 동일한 주파수를 사용하는 다른 장치 또는 블루투스 간에 간섭현상이 생길 수 있습니다. 블루투스는 페어링 이후에 무선통신 상황에 따라 주파수를 일정간격으로 (클래식 블루투스 1Mhz , BLE 2Mhz) 건너뛰어 다른 채널의 주파수로 통신하는데 이를 호핑(Hopping) 기술이라고 합니다.

### 01-4 BLE 동작 방법

BLE 기기들은 기본적으로 Advertise(Broadcast)과 Connection이라는 방법으로 외부와 통신합니다.

#### ① Advertise(Broadcast) Mode



▲ advertise 모드

BLE의 브로드캐스트 모드는 특정한 디바이스와 연결 없이 주변의 모든 디바이스에게 데이터 패킷을 보냅니다. 일정한 주기로 주변의 모든 디바이스에게 일방적으로(한 방향 통신) 데이터를 보내는 모드입니다. 주기적으로 소량의 데이터를 여러 기기에 전달해야 하는 경우에 적합합니다.

- Advertiser (Broadcaster) : 연결 없이 데이터 패킷을 주기적으로 보내는 기기
- Observer : 데이터 패킷을 받기 위해 주기적으로 Scanning하는 기기

**TIP**

브로드캐스트 모드는 일반적인 연결모드에 비해서 보안이 취약해서 보안이나 개인정보 보호가 필요한 데이터를 사용하는 경우에는 적합하지 않습니다.

## ② Connection Mode

브로드캐스트 모드와는 달리 양방향으로 데이터를 주고받거나 데이터 패킷으로만 전달하기에는 많은 양의 데이터를 주고받아야 하는 경우에는 Connection Mode(연결 모드)로 통신을 합니다. ‘일대일’ 방식으로 기기들 간에 데이터 교환이 일어납니다.



▲ Central and Peripheral

## 02 \_ 비콘(Beacon)이란?

---

근거리에 있는 스마트 기기를 자동으로 인식하여 필요한 데이터를 전송할 수 있는 무선 통신 장치입니다. 블루투스 비콘(Bluetooth Beacon)이라고도 합니다. 근거리 무선 통신인 NFC가 10cm 이내의 근거리에서만 작동하고 일반적인 블루투스가 10m정도에서 사용한다면, 비콘(Beacon)은 최대 50m 거리에서 작동할 수 있습니다. 또한 비콘은 전원이 없는 장소에 배터리만으로 설치 가능하기 때문에 배터리 소모 속도가 매우 중요합니다. 보통 비콘 업체들이 자사 제품의 배터리 수명이 2년이라고 하지만 실제 수명은 비콘의 사용조건에 따라 다릅니다. 비콘의 데이터 신호 발송 간격 및 비콘의 실제 하드웨어적인 설계에 따라서 각각 다르기 때문에 정확한 확인이 필요합니다.

블루투스 비콘은 UUID(Universally Unique Identifier)가 포함된 정보 패킷을 보낼 수 있습니다. 이때 UUID는 해당 비콘에 특정한 이벤트를 유발할 수 있는데, 만약 그 이벤트가 특정 제품이나 브랜드에 관한 알림이라면 이는 광고의 목적으로 사용될 수 있습니다. 예를 들어 Apple의 iBeacon의 경우 UUID가 단말기의 앱에 의해 인식되고 이것이 광고의 형태로 나타납니다.

또한 비콘으로 위치를 나타낼 수 있는데 특정 건물의 내부에서 위치를 나타내고자 할 때 방마다 여러 개의 비콘을 설치하면 약 2미터 이내로 사용자의 위치를 파악할 수 있습니다. 블루투스 비콘은 RSSI (Received Signal Strength Indicator) 값을 전송할 수 있기 때문에 사전에 알려진 비콘의 출력 신호 세기와 신호강도를 사용하면 비콘과 단말기 사이의 거리를 추정할 수 있습니다. 하지만 이러한 근사값은 오차가 상당하여 정확한 위치추정의 경우 다른 기술들이 필요 합니다.

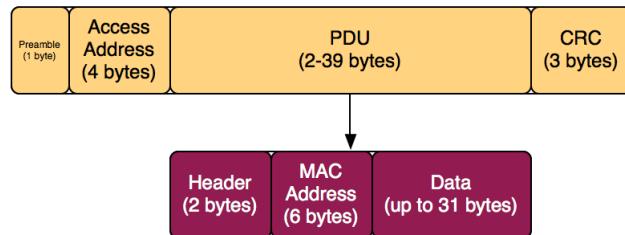
### 02-1 애플의 아이비콘(iBeacon)



▲ iBeacon

아이비콘은 2013년에 애플이 제시한 실내 위치 확인 시스템(indoor positioning system, IPS)으로, 애플은 “근처의 iOS 기기에 존재를 알릴 수 있는 새로운 차원의 저전력, 저비용 송신기”라고 설명합니다. 아이비콘 앱이 활성화된 단말기의 위치를 정확히 파악해서 이 단말기에 자동으로 관련 정보를 전송하는데 사용할 수 있습니다.

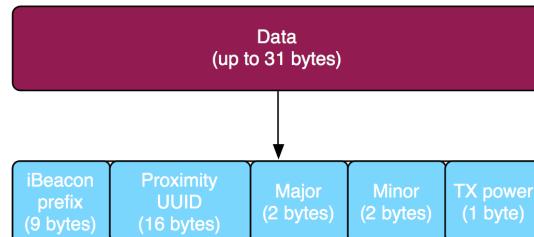
블루투스 4.0 LE(Bluetooth Low Energy)의 비콘 신호로 위치를 탐지하기 때문에, iOS는 물론 애플이 만들지 않은 블루투스 LE를 장착한 모든 기기(안드로이드 포함)에서 아이비콘을 이용할 수 있습니다.



▲ iBeacon 패킷구조

아이비콘의 데이터 패킷은 위의 그림과 같이 최대 47바이트까지 가능합니다.

- 4 바이트의 access address는 브로드캐스트 모드일 때 항상 0x8E89BED6입니다.
- 2~39 바이트의 PDU(Protocol Data Unit)에는 2바이트의 헤더(Header)가 포함되어 있는데, 헤더는 뒤의 Mac Address + Data의 길이를 나타냅니다. (최대 37바이트)
- 3 bytes CRC(cyclic redundancy check)



▲ 데이터 패킷 분석

비콘은 데이터 패킷을 주기적으로 보내는데 예를 들어서 Estimote사의 비콘이 보내는 데이터 패킷 속에 내용은 다음과 같습니다.

```
02 01 06 1A FF 4C 00 02 15 B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D 00 49 00 0A C5
```

데이터 패킷을 순서대로 정렬해 보면 다음과 같이 구조적으로 분석 할 수 있습니다.

❶ 02 01 06 1A FF 00 4C 02 15 :	iBeacon prefix
❷ B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D :	UUID
❸ 00 49 :	Major
❹ 00 0A :	Minor
❺ C5 :	TX power

- ❶ iBeacon Prefix : 아이비콘의 고정 고유 번호
- ❷ UUID( universally unique identifier ) : 범용 고유 식별자로 Major 와 Minor 식별번호와 조합하여 특정지역과 특정그룹안의 비콘을 식별 가능
- ❸ 2 byte의 비콘 그룹을 나타내는 식별 번호
- ❹ 2 byte의 비콘 그룹 내에서 고유 비콘을 나타내는 식별 번호
- ❺ 통신 세기를 나타내는 신호로써 이 신호로 비콘 거리를 측정

## 02-2 구글의 에디스톤



▲ 구글 에디스톤

영국의 등대의 이름을 따서 이름지은 에디스톤은 2015년 구글이 직접 발표한 비콘의 표준 규격입니다. 에디스톤은 오픈소스 형태로 Github에 공개되어 있어서 누구든지 사용할 수 있습니다. 플랫폼도 안드로이드 뿐 아니라 iOS에서도 사용가능하며 BLE를 감지할 수 있는 모든 기기에 에디스톤 비콘을 적용할 수 있습니다. 그리고 다양한 프레임을 지원하여 좀 더 독창적인 비콘 서비스를 구현 할 수 있습니다.

## (1) 패킷 유형에 따른 분류

에디스톤은 4종류의 패킷 유형이 있습니다.

### ① Eddystone-UID

UID는 Unique identifier 의 약자로, 아이비콘의 UUID와 비슷하게 설치된 앱을 실행할 때 알림을 보내는 등의 용도로 사용합니다. 16 byte로 구성되는데 2개로 구분됩니다.

- Namespace(10 byte) : 공통이름으로 설정
- Instance(6 byte) : 각 비콘을 구분하는 이름 부여

### ② Eddystone-URL

블루투스 신호를 통해 URL 정보를 압축해서 전송하여 크롬 브라우저로 바로 확인 가능하여 추가적인 어플 설치 없이도 즉시 관련 행사나 정보를 확인 가능하여 효율적이고 쉽게 정보를 전달 가능합니다. 또한 웹페이지를 기반으로 위치기반 서비스를 구현하는 ‘Physical Web’ 프로젝트의 근간이 되는 패킷유형입니다.

### ③ Eddystone-TLM

TLM은 원격측정(Telemetry)의 약자로 비콘의 배터리정보, 기기 온도, 패킷 전송 횟수 등의 정보를 전송합니다.

### ④ Eddystone-EID

EID는 임시 ID(Ephemeral Identifier)를 의미 합니다. 즉, ‘가상계좌’ 번호와 같이 일시적으로 유효한 ID를 제공하여 비콘을 짧은 시간 내 사용해야 하는 경우 유용하게 사용 할 수 있습니다. 관리자권한을 가진 사람만이 수정할 수 있어서 안전성과 보안이 강화된 패킷입니다.

## 03 \_ 라즈베리 파이 4를 블루투스 비콘으로 바꾸기

보통 비콘을 사용하기 위해서는 스마트 폰에 특정한 앱을 설치하고 특정한 웹사이트에 접속을 해야 비콘의 기능을 이용할 수 있었는데, 여기서는 Eddystone-URL을 이용하여 내가 가진 블로그나 홈페이지 등 특정 웹사이트를 전송하는 기능을 이용해 보겠습니다. LED를 제어하는 라즈베리 파이 웹서버를 만들고 웹서버 주소를 라즈베리 파이 4 비콘으로 바꾸고 구글의 Eddystone-URL 패킷으로 스마트폰에 전송하여 누구든지 스마트 폰을 가지고 블루투스를 스캔하여 근처의 라즈베리 파이 4 비콘이 전송하는 웹서버에 접속하여 LED를 제어해 보겠습니다.

라즈베리 파이 4를 비콘으로 바꾸는 것은 매우 쉽습니다.

- ① 라즈베리 파이의 블루투스 기능을 켭니다.
  - ② 블루투스 디바이스의 설정을 "advertise and not-connectable" 모드로 바꿉니다.
  - ③ 비콘이 전송할 패킷을 입력합니다.

3번째 과정이 가장 어려운데, 구글에서 만들어 놓은 에디스톤 URL 변환 사이트(Eddystone URL command calculator)에서 비콘의 전송할 URL 주소를 패킷으로 변화 시켜 줍니다.

TIP

Eddystone-URL에서 전송가능한 URL의 길이가 한계가 있기 때문에 URL의 길이가 상당히 길면, <https://goo.gl>에서 긴 URL을 짧은 URL로 줄여서 변환하면 됩니다.

- 에디스톤 URL 변환 사이트 : <https://yencarnacion.github.io/eddystone-url-calculator/>

그러면 에디스톤 URL 변환 사이트에서 Daum 포털 주소를 입력하여 비콘 패킷으로 변환 시켜 보겠습니다.

Eddystone URL command calculator

<https://www.daum.net/>

Your commands for "https://www.daum.net/" are

```
$ sudo hciconfig hci0 up
```

```
$ sudo hciconfig hci0 leady 3
```

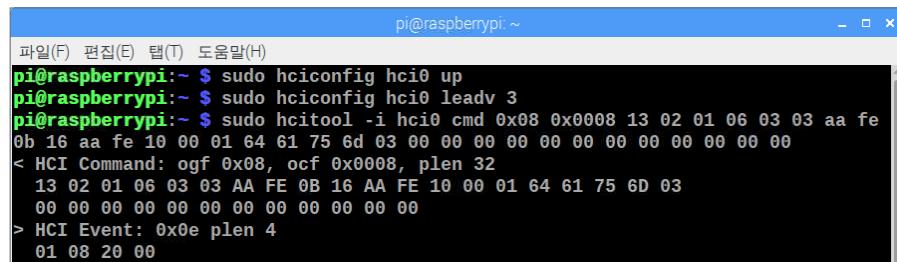
▲ URI 변환

소스를 분석해 보면 다음과 같습니다.

```
sudo hciconfig hci0 up
sudo hciconfig hci0 leadv 3
sudo hcitool -i hci0 cmd 0x08 0x0008 13 02 01 06 03 03 aa fe 0b 16 aa fe 10 00 01 64 61 75 6d 03 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

hciconfig hci0 up 과 leadv 3 은 라즈베리 파이의 블루투스 장치를 동작시키는 명령어이고 hcitool -i hci0 cmd 명령어로 비콘 패킷 메시지를 주변으로 브로드캐스트 전송을 합니다.

라즈비안의 CMD 창에 다음과 같이 그대로 블루투스 명령어를 넣으면 자체적으로 블루투스 장치를 가지고 있는 라즈베리 파이 4가 비콘처럼 주기적으로 Eddystone-URL 패킷 메시지를 전송 합니다.



```
pi@raspberrypi:~ $ sudo hciconfig hci0 up
pi@raspberrypi:~ $ sudo hciconfig hci0 leadv 3
pi@raspberrypi:~ $ sudo hcitool -i hci0 cmd 0x08 0x0008 13 02 01 06 03 03 aa fe
0b 16 aa fe 10 00 01 64 61 75 6d 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00
< HCI Command: ogf 0x08, ocf 0x0008, plen 32
  13 02 01 06 03 03 AA FE 0B 16 AA FE 10 00 01 64 61 75 6D 03
  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
> HCI Event: 0x0e plen 4
  01 08 20 00
```

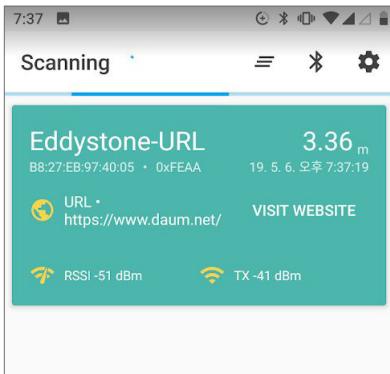
▲ 라즈베리 파이 4가 비콘처럼 동작

이제 스마트폰에 비콘 스캐너를 설치합니다. 같은 앱이 아니더라도 비슷한 종류의 어플도 괜찮습니다.



▲ Beacon Scanner 설치

자, 그러면 스마트폰에서 비콘 스캐너를 동작시켜서 스캐닝하면 라즈베리 파이 비콘이 전송하는 다음(Daum) 포탈의 URL 주소가 스캐닝에 잡히는 것을 확인할 수 있습니다.



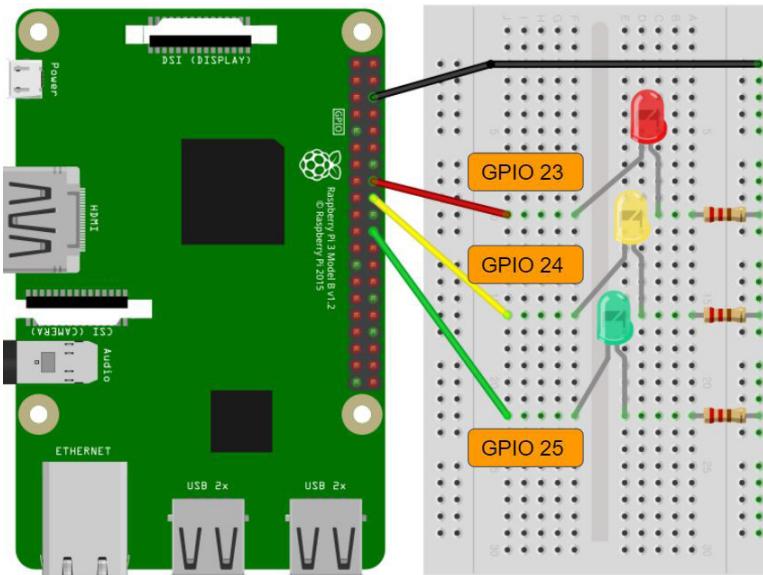
▲ 스마트폰에서 비콘 스캔결과

결과적으로 라즈베리 파이 4에서 보낸 URL 주소 메시지를 스마트폰에서 수신하여 URL 주소를 따라 어플에서 “VISIT WEBSITE”를 클릭하면 다음 포털로 이동하게 됩니다.

### 03-1 라즈베리 파이 LED 제어 웹서버 실행하기

#### (1) 브레드보드 연결

- 준비물 : led 3개 , 저항 220Ω , 전선



▲ 브레드보드 연결

지니(Geany)에서 homeLED.py 파일을 만들어 web-server 폴더 안에 저장합니다.

파이썬 프로그램은 (:) 기호 다음 아랫줄들은 들여쓰기가 정확해야 하므로 주의합니다.

## (2) 코드 작성하기

### ① 파이썬 코드 작성하기

01 다음과 같이 코드를 작성하고 저장합니다.

실행파일 : /home/pi/webapps/ch08/homeLED.py

```
#-*-coding:utf-8

# 필요한 라이브러리를 불러옵니다.
import RPi.GPIO as GPIO
from flask import Flask, render_template, request
app = Flask(__name__)

# 불필요한 warning 제거, GPIO핀의 번호 모드 설정
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# pins란 딕셔너리를 만들고 GPIO 23, 24, 25 핀을 저장합니다.
pins = {
    23 : {'name' : 'RED LED', 'state' : GPIO.LOW},
    24 : {'name' : 'Yellow LED', 'state' : GPIO.LOW},
    25 : {'name' : 'Green LED', 'state' : GPIO.LOW}
}

# pins 내에 있는 모든 핀들을 출력으로 설정하고 초기 LED OFF 설정
for pin in pins:
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.LOW)

# 웹서버의 URL 주소로 접근하면 아래의 main() 함수를 실행
@app.route("/")
def main():
    # pins 내에 있는 모든 핀의 현재 핀 상태(ON/OFF)를 업데이트
    for pin in pins:
        pins[pin]['state'] = GPIO.input(pin)
    # templateData 에 저장
    templateData = {
        'pins' : pins
    }
    # 업데이트 된 templateData 값을 homeLED.html로 리턴
    return render_template('homeLED.html', **templateData)

# URL 주소 끝에 “ /핀번호/<action> ” 을 붙여서 접근시에 action 값에 따라 동작
@app.route("/<changePin>/<action>")
def action(changePin, action):
    # 현재 핀번호를 URL 주소로 받은 핀번호로 설정
    changePin = int(changePin)
    # 핀번호에 설정된 이름값을 불러옴
    deviceName = pins[changePin]['name']
```

```

# action 값이 'on' 일때
if action == "on":
    GPIO.output(changePin, GPIO.HIGH)
# action 값이 'off' 일때
if action == "off":
    GPIO.output(changePin, GPIO.LOW)
# GPIO 핀의 ON/OFF 상태 저장
pins[changPin]['state'] = GPIO.input(changPin)
# 핀들의 값을 업데이트 해서 templateData에 저장
templateData = {
    'pins' : pins
}
# 업데이트 된 templateData 값을 homeLED.html로 리턴
return render_template('homeLED.html', **templateData)
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=False)

```

ch08 폴더 내에 templates 폴더를 만들고 그 안에 homeLED.html 파일을 만듭니다.

## ② html 코드 작성하기

### 01 다음과 같이 코드를 작성하고 저장합니다.

실습파일 : 실습파일: /home/pi/webapps/ch08/templates/homeLED.html

```

<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>GPIO TEST</title>
    <style>
        body {font-size:120%;}
        #main {display: table; margin: auto; padding: 0 10px 0 10px; }
        .button { padding:5px 5px 5px 5px; width:100%; font-size: 120%; }
        #but1 { background-color: mistyrose; }
        #but2 { background-color: honeydew; }
    </style>
</head>
<body>
    <div id='main'>
        <h2>웹서버 GPIO 제어</h2>
        {% for pin in pins %}
        <p>The {{ pins[pin].name }}
        {% if pins[pin].state == True %}
            <a href="/{{pin}}/off"><button id='but1'>끄기</button></a>
        {% else %}
            <a href="/{{pin}}/on"><button id='but2'>켜기</button></a>
        {% endif %}
    
```

```

</p>
{%
  endfor %}
</div>
</body>
</html>

```

### (3) 웹서버 실행하기

지니 프로그램에서 작성한 homeLED.py 파일을 실행합니다.

The screenshot shows the Geany IDE with the file 'homeLED.py' open. The code defines a function 'action' that takes 'changePin' and 'action' as parameters. It sets the current pin number from the URL query string, reads the pin name from the pins dictionary, and then toggles the GPIO pin state based on the action ('on' or 'off'). It also updates the pins dictionary with the current state. Finally, it returns the rendered template from 'homeLED.html'. If the name is 'main', it runs the Flask application on port 5000. Below the IDE is a terminal window showing the command 'cd /home/pi/web-server' and the execution of 'geany\_run\_script\_0MU01Z.sh'. The terminal output shows the Flask application running on port 5000 and receiving several GET requests from the IP address 192.168.0.8.

```

homeLED.py - /home/pi/web-server - Geany
파일(F) 편집(E) 찾기(S) 보기(V) 문서(D) 프로젝트(P) 제작(B) 도구(T) 도움말(H)
[File] [Edit] [Search] [View] [Document] [Project] [Build] [Tools] [Help]
homeLED.py ✘
40 def action(changePin, action):
41     # 현재 핀번호를 URL 주소로 받은 핀번호로 설정
42     changePin = int(changePin)
43     # 핀번호에 설정된 이름값을 불러옴
44     deviceName = pins[changePin]['name']
45     # action 값이 'on'일때
46     if action == "on":
47         GPIO.output(changePin, GPIO.HIGH)
48     # action 값이 'off'일때
49     if action == "off":
50         GPIO.output(changePin, GPIO.LOW)
51
52     # 현재 라즈베리파이의 사용하는 GPIO 핀들의 ON/OFF 상태 저장
53     pins[changePin]['state'] = GPIO.input(changePin)
54
55     # 핀들의 값을 업데이트 해서 templateData에 저장
56     templateData = {
57         'pins' : pins
58     }
59     # 업데이트 된 templateData 값을 homeLED.html로 리턴
60     return render_template('homeLED.html', **templateData)
61
62 if __name__ == "__main__":
63     app.run(host='0.0.0.0', port=5000, debug=False)
64
 상태 pi@raspberrypi:~/webapps/ch05 $ cd '/home/pi/web-server'
컴파일러 /bin/sh /tmp/geany_run_script_0MU01Z.sh
메시지 pi@raspberrypi:~/web-server $ /bin/sh /tmp/geany_run_script_0MU01Z.sh
    * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
터미널 192.168.0.8 - - [07/May/2019 06:29:36] "GET / HTTP/1.1" 200 -
192.168.0.8 - - [07/May/2019 06:29:37] "GET /favicon.ico HTTP/1.1" 404 -
192.168.0.8 - - [07/May/2019 06:29:45] "GET /25/on HTTP/1.1" 200 -

```

▲ 지니에서 Flask 웹서버 실행

이때 실행되는 웹서버의 주소는 현재 라즈베리 파이의 IP 주소에 :5000 (포트5000)을 추가한 주소입니다. => 라즈베리 파이 IP 주소 : 5000

### (4) 웹서버 접속하기

라즈베리 파이와 같은 공유기에 와이파이로 접속한 스마트 폰이나 같은 공유기에 유선으로 접속되어 있는 PC로 웹 브라우저를 열어서 라즈베리 파이 IP 주소 : 5000을 입력합니다.



▲ 스마트폰에서 웹서버 접속

## 03-2 라즈베리 파이 4를 비콘으로 바꿔서 웹서버 접속하기

Flask 웹서버로 만든 GPIO TEST 웹페이지 주소를 그대로 에디스톤 URL 변환사이트에 입력해서 변환합니다.

### Eddystone URL command calculator

```
http://192.168.0.9:5000

Your commands for "http://192.168.0.9:5000" are:
$ sudo hciconfig hci0 up
$ sudo hciconfig hci0 leadv 3
$ sudo hcitool -i hci0 cmd 0x08 0x0008 1e 02 01 06 03 03 aa fe 16 16 aa fe 10 00 02 31 39 32 2e 31 36 38 2e 30 2e 39 3a 35 30 30 00
```

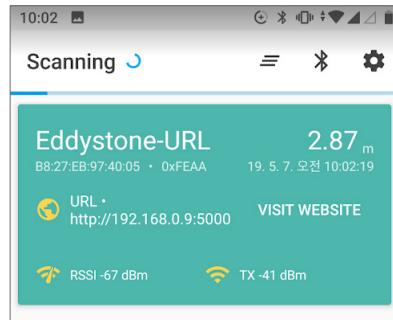
▲ Flask 웹서버 주소 변환

**01** 라즈비안의 CMD 창에 다음과 같이 변환한 명령어를 입력합니다.

```
pi@raspberrypi:~ $ sudo hciconfig hci0 up
pi@raspberrypi:~ $ sudo hciconfig hci0 leadv 3
LE set advertise enable on hci0 returned status 12
pi@raspberrypi:~ $ sudo hcitool -i hci0 cmd 0x08 0x0008 1e 02 01 06 03 03 aa fe
16 16 aa fe 10 00 02 31 39 32 2e 31 36 38 2e 30 2e 39 3a 35 30 30 30 00
< HCI Command: ogf 0x08, ocf 0x0008, plen 32
  1E 02 01 06 03 03 AA FE 16 16 AA FE 10 00 02 31 39 32 2E 31
  36 38 2E 30 2E 39 3A 35 30 30 30 00
> HCI Event: 0xe plen 4
  01 08 20 00
```

▲ 웹서버 주소 전송 패킷을 입력

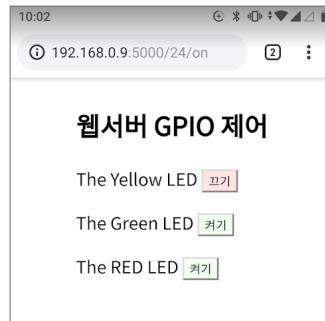
**02** 라즈베리 파이가 비콘처럼 주기적으로 웹서버 주소를 비콘 메시지로 주변에 전송하고 있으므로 스마트폰으로 비콘 스캐너 앱으로 스캐닝 하면 다음 그림과 같이 웹서버 URL 주소가 전송되어 나타납니다.



▲ 스마트폰으로 비콘 스캐너 확인

**03** “VISIT WEBSITE”를 클릭해서 웹서버에 접속해 라즈베리 파이로 LED를 제어해 봅니다.

이때 스마트폰이나 PC는 라즈베리 파이와 같은 유무선 공유기를 사용하고 있어야 합니다. 만약, 유무선 공유기에 상관없이 웹서버에 접속하기 위해서는 “11장 포트포워딩과 슈퍼디엠지”에서 공유기 설정으로 웹서버 IP를 외부 IP로 사용하는 방법을 보고 설정할 수 있습니다.



▲ 웹서버로 LED 제어 확인

# Raspberry Pi

이번 장에서는 라즈베리 파이 카메라를 세팅하고 사진 찍기, 동영상 촬영을 해 본 뒤 UV4L 모듈을 이용하여 웹 스트리밍을 구현해보고 Motion 모듈을 이용하여 모션 감지를 해보도록 하겠습니다.

CHAPTER  
**09**

# 라즈베리 파이 카메라 활용하기

- 01 \_ RPI 카메라
- 02 \_ 파이썬을 활용한 RPI 카메라
- 03 \_ UV4L 소개 및 설치
- 04 \_ WebRTC
- 05 \_ Motion 감지 프로그램

## 01 \_ RPI 카메라

---

곰돌이 눈에 라즈베리 파이 카메라가 달려있어요!!



▲ 곰돌이 눈에 라즈베리 파이 카메라

### 01-1 라즈베리 파이 카메라 세팅하기

파이 카메라는 풀HD급 1080p 해상도를 지원합니다. 파이 카메라를 사용하려면 플렉스 케이블(하얀 선 끝이 파란색)을 라즈베리 파이 보드의 CSI (Camera Serial Interface) 커넥터에 연결하면 됩니다. 연결 시 주의할 점은 라즈베리 파이에 전원이 꺼져 있어야 합니다.



▲ 라즈베리 파이 카메라

라즈베리 파이 카메라를 사용할 때 VNC 그래픽모드로 원격 접속하여 테스트하게 되면 GPU 용량이 부족하여 테스트가 불가능할 수 도 있으니 카메라를 사용할 때는 원격 접속을 하지 않는 것을 권장합니다.

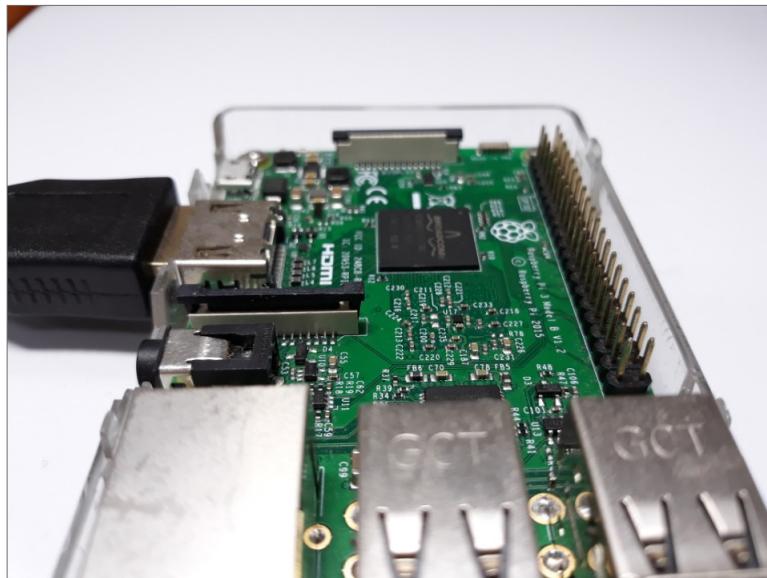
## (1) RPI 카메라 연결하기

01 라즈베리 파이 전원을 off합니다.



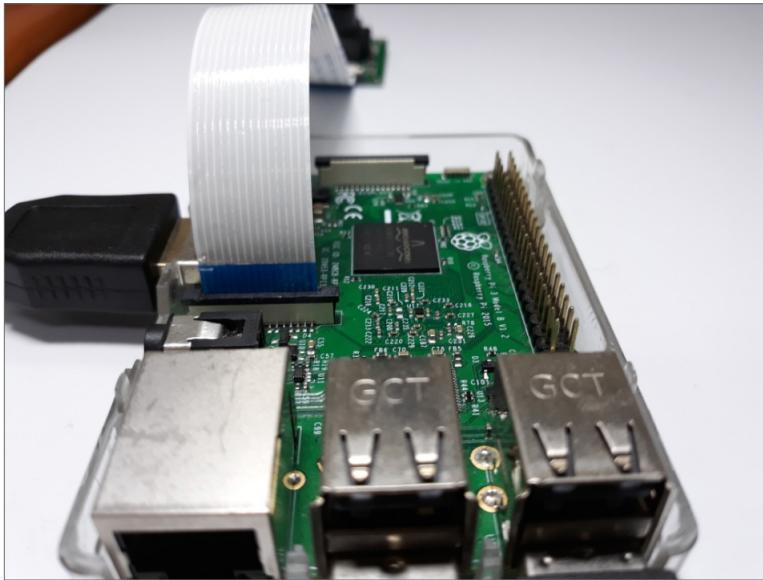
▲ 라즈베리 파이 전원 OFF

02 카메라 단자를 다음 그림과 같이 열어줍니다. 카메라 단자는 HDMI케이블과 오디오 연결 단자 사이에 있습니다.



▲ 라즈베리 파이 카메라 단자 열기

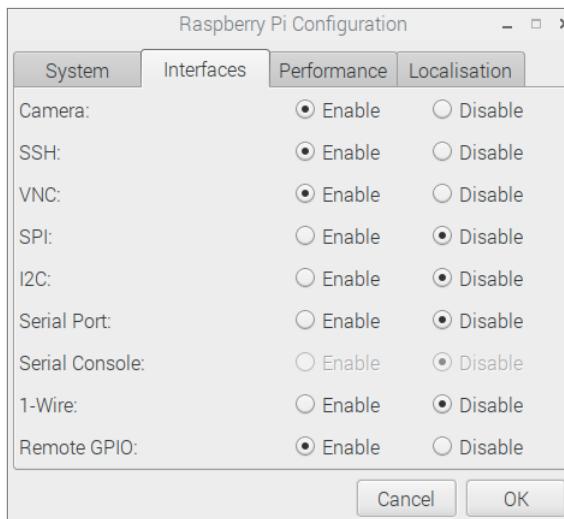
03 파란색 부분이 LAN포트를 향하도록 연결한 뒤 카메라 단자를 닫아줍니다.



▲ 라즈베리 파이 카메라 연결

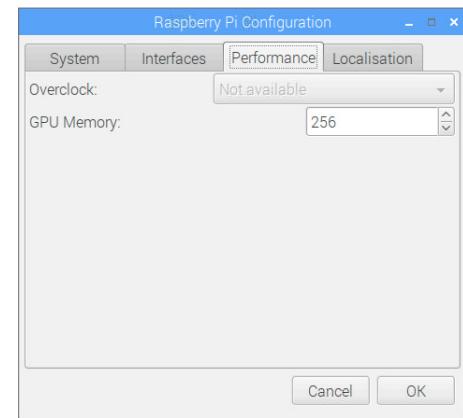
## (2) RPI Camera 설정하기

01 메뉴 – 기본설정 – Raspberry Pi Configuration – Interfaces – Camera – Enable을 선택합니다.



▲ 카메라 인터페이스 Enable 확인

**02** 메뉴 – 기본설정 – Raspberry Pi Configuration – Performance – GPU Memory 256을 선택합니다.



▲ GPU 메모리 확인

**03** 라즈비안 업데이트 및 재시작합니다.

```
$ sudo apt-get update  
$ sudo apt-get upgrade  
$ sudo rpi-update$ sudo reboot
```

## 01-2 라즈베리 파이 카메라 테스트하기

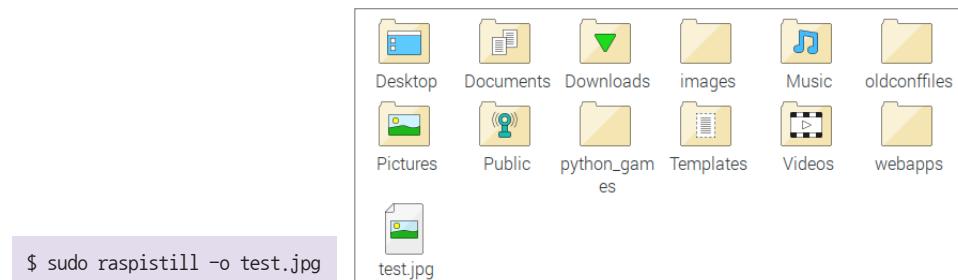
카메라 테스트 전에 최신버전으로 업데이트를 하길 바랍니다. 그 이유는 MMAL(Multi-Media Abstraction Layer)라이브러리를 활용하여 카메라 테스트를 해야 하는데 설치가 안 되어 있을 수 있기 때문입니다. 파이카메라는 세 가지 포트를 가지고 있습니다.

첫째, 프리뷰 포트입니다. 미리보기를 가능하게 해줍니다.

둘째, 스틸 포트입니다. 사진을 찍을 수 있게 해줍니다.

셋째, 비디오 포트입니다. 영상을 촬영할 수 있게 해줍니다.

### (1) 사진 촬영하기



```
$ sudo raspistill -o test.jpg
```

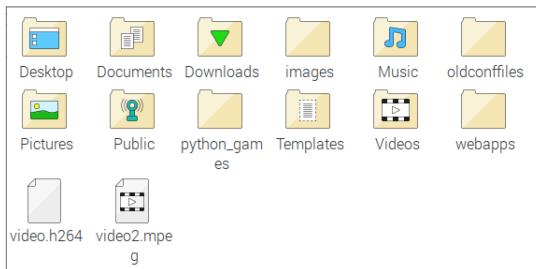
▲ test.jpg 확인

## (2) 영상 촬영하기

```
$ sudo raspivid -o video.h264  
$ sudo raspivid -o video2.mpeg -t 10000 -d
```

## (3) 영상 확인하기

```
$ sudo omxplayer video.h264  
$ sudo omxplayer video2.mpeg
```



▲ video.h264, video2.mpeg 확인

# 01-3 라즈베리 파일 카메라 명령어

## (1) help를 이용하여 명령어 살펴보기

```
$ raspistill --help  
$ raspivid --help
```

## (2) raspistill 옵션

- o 파일명 → 파일명으로 저장
- t 시간 → 시간 후에 촬영(단위 ms, 기본 5초)
- w, -h → 사진 크기(w넓이, h높이)
- br → 밝기 조정(0~100)

## (3) raspivid 옵션

- o 파일명 → 파일명으로 저장
- t 시간 → 시간 후에 촬영(단위 ms, 기본 5초)
- w.-h → 사진 크기 세팅(w 넓이, h 높이)
- q → 사진 파일 품질 조정(0~100)
- d → demo 모드
- s → signal 모드(신호가 들어오면 촬영)
- br → 밝기 조정(0~100)

## 02 \_ 파이썬을 활용한 RPI 카메라

### 02-1 python-picamera 모듈 활용하기

파이썬을 활용하여 파이카메라를 사용하기 위해서는 python-picamera 모듈을 활용하면 됩니다.

```
$ sudo apt-get install python-picamera
```

다음과 같이 이미 설치가 되어 있을 수 있습니다.

```
python-picamera is already the newest version (1.13).  
다음 패키지가 자동으로 설치되었지만 더 이상 필요하지 않습니다:  
  cgroupfs-mount  
Use 'sudo apt autoremove' to remove it.  
0개 업그레이드, 0개 새로 설치, 0개 제거 및 99개 업그레이드 안 함.  
▲ 이미 설치되어 있는 파이카메라 모듈
```

#### 01 다음과 같이 코드를 작성하고 저장합니다.

실행파일 : /home/pi/webapps/ch09/picam.py

```
import picamera  
  
with picamera.PiCamera() as camera:  
    camera.resolution = (640, 480)  
    camera.start_preview()  
    camera.start_recording('cos.h264')  
    camera.wait_recording(30)  
    camera.stop_recording()  
    camera.stop_preview()
```

- 자니 프로그램에서는 F5 로 실행합니다.
- 터미널에서는 다음과 같이 합니다.

```
$ cd /home/pi/webapps/ch09  
$ python3 picam.py
```

**02** 다음과 같이 코드를 작성하고 저장합니다.

실습파일 : /home/pi/webapps/ch09/picapture.py

```
import picamera  
import time  
  
with picamera.PiCamera() as camera:  
    camera.resolution = (640, 480)  
    camera.start_preview()  
    time.sleep(1)  
    camera.capture('cos.jpg')  
    camera.stop_preview()
```

- 지니 프로그램에서는 F5로 실행합니다.
- 터미널에서는 다음과 같이 합니다.

```
$ cd /home/pi/webapps/ch09  
$ python3 picapture.py
```

## 03 \_ UV4L 소개 및 설치

### 03-1 UV4L 소개

UV4L은 Userspace Video4Linux의 약자이며 단순한 프레임워크로서 V4L과 호환이 되는 실제 또는 가상의 비디오 장치들을 위한 사용자 영역 드라이버를 제공해주는 모듈입니다. UV4L을 설명하려면 먼저 V4L과 인터페이스에 대한 이해가 필요합니다.

#### (1) API

API란 Application programming Interface의 약자입니다. 특정 프로그램에서 운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 해주는 인터페이스입니다. 쉽게 말해서 내가 A라는 기능을 구현하고 싶은데 그 기능을 누군가가 제공해주면 나는 A라는 기능을 만들 필요 없이 사용하면 됩니다. 이때 A라는 기능을 제공해주는 것을 API라고 합니다.

인터페이스란 기보드나 디스플레이처럼 사람과 컴퓨터를 연결하는 장치라고 사전에 나와 있습니다. Inter는 ~사이라는 뜻입니다. Interchange라는 말을 많이 들어봤을 것입니다. Interchange는 교차로입니다. face는 직면하다는 뜻이 있습니다.

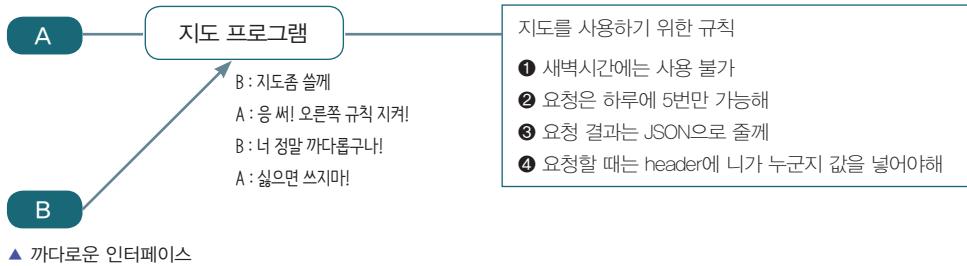
인터페이스는 A와 B를 연결해주는 매개체입니다. 컴퓨터 USB포트는 컴퓨터 하드웨어와 사람을 연결해주는 인터페이스입니다. 이때 USB포트는 이렇게 생겼으니까 USB는 이렇게 만들어야 한다는 표준이 만들어집니다.

여기서 중요한 개념이 나옵니다. "USB포트는 이렇게 생겼으니까 USB를 만드는 회사는 이 표준을 지켜" 이것은 프로토콜과는 조금 다른 의미입니다. 프로토콜 역시 서로간의 약속을 의미합니다. "친구야 오늘 모임 자리에서 귀를 두 번 만지면 지루하다는 뜻이니까 빨리 나가자는 뜻이야"라고 이야기 했을 때 친구가 "귀 두 번 만지는 것보다 기침을 두 번 할께"라고 약속하는 것을 프로토콜이라고 하는데 이 때 두 친구는 대등한 관계에 있습니다. 서로 약속을 할 때 일방적이지 않습니다. 하지만 Interface는 다릅니다. 서로 대등한 관계에 있지 않습니다. 상하관계가 존재하는 것이 인터페이스입니다.



▲ 인터페이스와 프로토콜의 차이

'A가 제공해주는 Application Programming을 B가 Interface(A가 일방적으로 정의한 약속)에 맞게 사용하는 것'을 API라고 합니다.



## (2) V4L

Video4Linux는 리눅스에서 비디오 디바이스를 제어하고 사용하기 위한 API입니다. 리눅스에서 비디오 장치를 제어하기 위한 프로그램을 만들어놨는데 너희가 사용하고 싶다면 사용할 수 있게 API를 만들어놨으니 규칙에 맞게 잘 사용하라는 뜻입니다. 최근에는 V4L이 아닌 향상된 버전인 V4L2를 사용합니다.

V4L2란 리눅스에서 카메라 입력을 받기 위한 표준 인터페이스로서 V4L2를 설치 후 활성화 할 때 사용자 프로그램이 커널을 통해 I/O요청을 확인하고 장치 드라이버로 전송이 이루어지는 것이 가능해집니다.

이러한 것처럼 사용자 프로그램이 커널을 통해 시스템 하드웨어에 접근 할 수 있도록 “dev” 디렉토리 내에 “video\*”라는 장치 파일 생성이 이루어지며 사용자는 이러한 “/dev/video”를 통해 자료를 읽거나 기타 장치로 자료 전송이 가능해집니다.

- Video4Linux : V4L은 Kernel에서 지원하는 기본 모듈로 TV 수신카드를 지원하기 위해 등장했습니다.
- Video4Linux2 : Video4Linux 1.0 Version이 TV 수신카드를 위해 등장했기 때문에 Web Cam에는 맞지 않습니다. V4L2는 USB Web Cam을 위해서 등장했습니다.

## (3) UV4L

UV4L은 Video4Linux2를 사용하면서 오랜 기간 사랑 받아 왔습니다. IoT 장치를 위한 스트리밍 서버 플러그인을 포함하고 있으며 이 플러그인은 오디오 및 비디오 스트리밍, 암호화 같은 것들을 웹을 통해 제공합니다. 또한 화상통신을 할 수 있도록 양방향 스트리밍을 지원합니다. 무엇보다 좋은 것은 커스터마이징을 원하는 개발자에게 RESTful API를 제공해 준다는 것입니다.

## 03-2 UV4L 설치하기

설치는 아래의 URL을 참고하였습니다.

<https://www.linux-project.org/uv4l/installation/>

### (1) 설치 시 주의할 점

#### Installation for ARM (Raspberry Pi)

*How to install or upgrade UV4L on Raspbian Wheezy, Raspbian Jessie & [Raspbian Stretch](#) for Raspberry Pi*

**IMPORTANT!** Packages for Raspbian Wheezy and Raspbian Jessie are no longer maintained, consider to upgrade your system to Raspbian Stretch instead.

The following instructions explain how to install UV4L on the official Raspbian Linux distributions available for any model of the Raspberry Pi boards: Zero, Zero W (Wireless), 1, 2, 3, Compute Module 1, Compute Module 3.

Other distributions than Raspbian and other ARM-based boards are known to work, but they are not officially supported.

As these instructions are updated and improved very frequently without notice, it is suggested to read them from scratch in case of problems and especially whenever a new UV4L module is announced. Important notes about specific drivers, modules, configurations, etc.. can be found at the bottom of this page.

#### ▲ UV4L 설치시 주의할 점

다음은 UV4L 공식 문서 내용을 번역하였습니다.

Raspbian Wheezy 및 Raspbian Jessie 용 패키지는 더 이상 유지 관리되지 않으며 시스템을 Raspbian Stretch로 업그레이드하는 것이 좋습니다.

다음 지침은 Raspberry Pi 보드의 모든 모델에 사용할 수 있는 공식 Raspbian Linux 배포판에 UV4L를 설치하는 방법을 설명합니다.

Zero, Zero W (무선), 1, 2, 3, Compute Module 1, Compute Module 3, Raspbian 및 기타 ARM 기반 보드 외의 다른 배포판도 작동하는 것으로 알려져 있지만 공식적으로 지원되지는 않습니다.

이 지침은 예고 없이 자주 업데이트 되고 개선되므로 문제가 발생한 경우 특히 새로운 UV4L 모듈이 발표될 때마다 처음부터 읽는 것이 좋습니다. 특정 드라이버, 모듈, 구성 등에 관한 중요한 정보는 이 페이지 하단에 있습니다.

- Raspbian Stretch를 권장
- USB캠, 라즈베리 파이 카메라 둘 다 사용 가능
- 예고 없이 업데이트되기 때문에 사이트에 자주 들어와서 문서를 읽어보는 것을 권장

## (2) UV4L 설치하기

**01** Stretch 저장소를 설정합니다.

```
$ curl http://www.linux-project.org/listing/uv4l_repo/lpkey.asc | sudo apt-key add -  
$ sudo nano /etc/apt/sources.list
```

가장 아래 부분에 추가해줍니다.

```
deb http://www.linux-project.org/listing/uv4l_repo/raspbian/stretch stretch main
```

**02** 변경된 저장소를 업데이트합니다.

```
$ sudo apt-get update
```

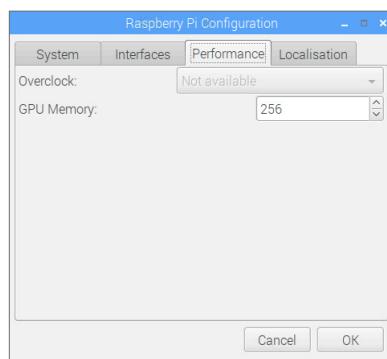
**03** UV4L 모듈을 설치합니다.

```
$ sudo apt-get install uv4l uv4l-raspicam
```

**04** 부팅시 자동으로 UV4L 드라이버를 로드하고 싶다면 아래 문장 추가합니다.

```
$ sudo apt-get install uv4l-raspicam-extras
```

**05** GPU Memory를 256MB로 설정합니다.



▲ GPU 메모리 256MB 설정

**06** 펌웨어 업데이트를 합니다.

```
$ sudo rpi-update
```

**07** UV4L 서비스를 재시작합니다.

```
$ sudo service uv4l_raspicam restart
```

**08** 사진 캡쳐 테스트를 진행합니다.

```
$ dd if=/dev/video0 of=snapshot.jpeg bs=1M count=1
```

캡쳐된 사진은 /home/pi에서 확인할 수 있습니다.

**09** uvrl-server를 설치합니다.

```
$ sudo apt-get install uv4l-server uv4l-uvc uv4l-xscreen uv4l-mjpegstream uv4l-dummy uv4l-raspidisp
```

**10** WebRTC Streaming Server를 설치합니다.(양방향 스트리밍)

```
$ sudo apt-get install uv4l-webrtc
```

WebRTC는 Web ReALTime Communication의 약자로 웹 브라우저 간에 실시간, 플러그인이 필요 없이 영상 및 음성, 데이터 통신에 대한 공개된 표준입니다.

즉, 별도의 프로그램 설치 없이 웹 브라우저 사이에 화상통신, 음성, 채팅을 가능하게 합니다.

P2P로 기기 간 Direct 통신이 가능하여 서버의 기능 및 성능이 크게 필요하지 않습니다.

구글이 처음으로 WebRTC를 제안한 이후 Google, Mozilla, Opera 및 MS까지 기술적 표준을 만들어 가고 있습니다.

**11** SSL를 구성합니다.

보안상의 이유로 일부 브라우저에서는 HTTP를 통한 WebRTC기능을 이용할 수 없습니다. 스트리밍 서버에서 보안 HTTPS를 대신 구성해야 하는데 이를 위해서 SSL을 구성하여야 합니다. HTTPS와 SSL은 이 책에서는 따로 설명하지 않습니다. 개인키와 서명된 인증서를 구성하기 위해 다음과 같이 입력합니다.

```
$ openssl genrsa -out selfsign.key 2048&& openssl req -new-x509 -key selfsign.key -out selfsign.crt -sha256
```

위와 같이 입력하면 Country를 설정하는 부분에는 KO라고 입력하고 나머지 부분은 굳이 설정하지 않으셔도 됩니다. 그래서 점(.)을 입력한 뒤 엔터를 쳐서 뒷부분은 모두 default값을 주게 합니다.

## 12 xmpp-bridge를 설치합니다.

넷상의 두 지점간의 통신 규격을 정의한 것으로 양 지점간의 메시징, 상태값들이 실시간으로 전달이 가능하게 하는 규격입니다. WebRTC를 통해 양방향 통신을 하기 위해서는 꼭 설치해야하는 모듈입니다.

```
$ sudo apt-get install uv4l-xmpp-bridge
```

## 13 모든 설치가 완료되었습니다. 재부팅 후 uv4l-server가 작동하고 있는지 확인하려면 8080포트 확인하면 됩니다.

```
$ sudo reboot  
$ netstat -nlpt
```

```
pi@raspberrypi:~ $ netstat -nlpt  
(Not all processes could be identified, non-owned process info  
will not be shown, you would have to be root to see it all.)  
Active Internet connections (only servers)  
Proto Recv-Q Send-Q Local Address          Foreign Address      State  
PID/Program name  
tcp        0      0 0.0.0.0:5900            0.0.0.0:*            LISTEN  
-  
tcp        0      0 0.0.0.0:8080            0.0.0.0:*            LISTEN  
-  
tcp        0      0 0.0.0.0:22              0.0.0.0:*            LISTEN  
-  
tcp6       0      0 ::::5900              ::::*               LISTEN  
-  
tcp6       0      0 ::::22                ::::*               LISTEN
```

▲ 사용중인 포트 확인

## 14 라즈베리 파이에서 인터넷을 열고 아래의 주소를 입력합니다.

- <http://localhost:8080>



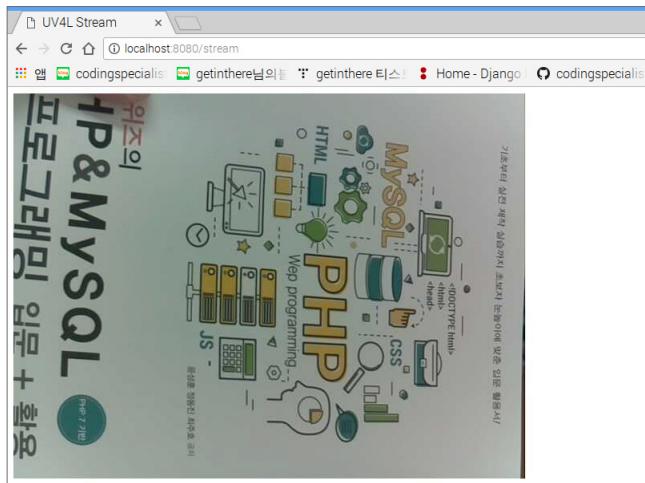
▲ UV4L 메인 화면

화질이 굉장히 높게 잡혀 있어서 화질을 낮추고 스트리밍을 확인해봅시다. Control Panel을 클릭합니다. 해상도를 가로(width)와 세로(height)를 각각 640 – 480으로 세팅한 뒤 가장 밑에 [apply] 버튼을 클릭하여 적용시켜줍니다.



▲ 웹 스트리밍 해상도 줄이기

다시 메인화면으로 돌아와서 MJPEG/Stills stream을 클릭합니다. 영상이 스트리밍 되고 있는 것을 확인할 수 있습니다.



▲ 웹 스트리밍 확인하기

**15** preview(미리보기) 설정을 해제합니다. 영상이 스트리밍 될 때마다 미리보기 화면이 나오는 것을 해제하기를 원한다면 다음과 같이 하면 됩니다. 라즈베리 파이 UV4L 카메라 설정파일로 이동합니다.

```
$ sudo nano /etc/uv4l/uv4l-raspicam.conf
```

nopreview = yes로 변경해줍니다.

```
### video overlay options:  
nopreview = yes  
fullscreen = no  
# osd-layer = 2  
# opacity = 255  
### preview window <x, y, w, h>:  
preview = 480  
preview = 240  
preview = 320  
preview = 240
```

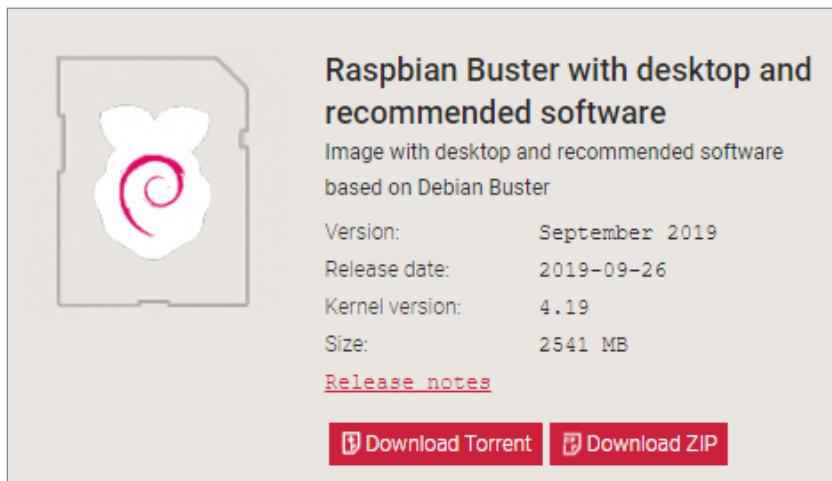
▲ 파이카메라 미리보기 해제

UV4L 서비스를 재시작합니다.

```
$ sudo service uv4l_raspicam restart
```

### (3) UV4L 설치 시 설치하지 않았던 것들에 대한 설명

라즈베리 파이 배포판은 buster(2019.7.10일 기준)입니다. 2019.4.8일까지 하더라도 stretch가 공식 배포판이었는데 정말 짧은 시간만에 변경되었습니다. UV4L에서 buster를 지원하지 않기 때문에 (2019.7 기준) stretch로 저장소를 지정하면 됩니다.



▲ 라즈비안 배포판 Stretch

- tc358743은 하드웨어입니다. 해당 칩은 HDMI로 입력된 신호 값을 MIPI로 변환해줍니다. MIPI는 "Mobile Industry Processor Interface"의 약자로 모바일 기기를 구성하는 각각의 구성 요소들 사이의 인터페이스를 규정하기 위하여 만들어졌습니다. 모바일 기기는 현재 성능이 빠른 속도로 향상되고 있으며, 다양한 기능과 성능의 구성 요소들이 결합됨으로 인하여 이들 간에 적절한 인터페이스에 대한 표준이 필요하게 되어서 만들어졌습니다. 우리는 현재 HDMI to MIPI가 필요하지 않기 때문에 설치하지 않습니다.
- demo는 UV4L로 구현할 수 있는 예제 소스가 포함되어 있습니다. 우리는 필요하지 않기 때문에 설치하지 않았습니다.

## 04 \_ WebRTC

01 다음 주소(URL)로 이동합니다.

- <http://localhost:8080>

02 WebRTC를 클릭합니다.



▲ UV4L 메인 화면

03 영상의 해상도를 640x480 15 fps로 설정합니다.

<b>Remote peer options</b>	
Video:	<input checked="" type="checkbox"/> force use of hardware codec for <b>640x480 15 fps</b>
NOTE: if your browser does not support the hardware codec yet, try Firefox with the codec plugin enabled or a recent version of Chrome.	

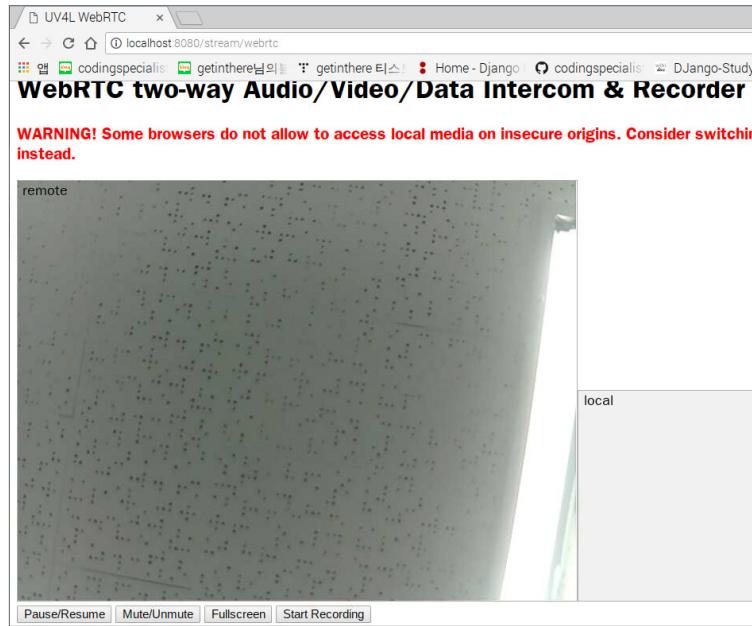
▲ 해상도 설정

04 스트리밍을 보려면 call!을 클릭합니다. 종료를 원하면 Hang up을 클릭하면 됩니다.

<b>▼ Advanced options</b>	
Remote Peer/Signalling Server Address: <input type="text" value="localhost:8080"/>	
Optional ICE Servers (STUN/TURN): <input type="text"/>	
Trickle ICE: <input checked="" type="checkbox" value="true"/>	
<b>Call!</b>	<b>Hang up</b>

▲ 스트리밍 시작 Call!

## 05 스트리밍이 되고 있는지 확인합니다.



▲ 스트리밍 확인

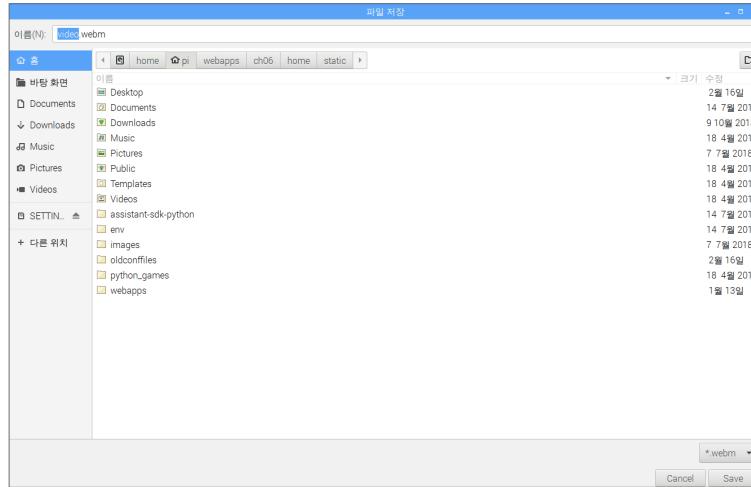
06 녹화를 시작하기 위해 Start Recording을 클릭합니다. 녹화를 종료하기 위해서는 Stop Recording을 클릭합니다.

07 녹화가 정상적으로 되었는지 [play] 버튼을 클릭하여 확인한 뒤 저장을 원하면 [Save as] 버튼을 클릭합니다.



▲ 녹화 시작

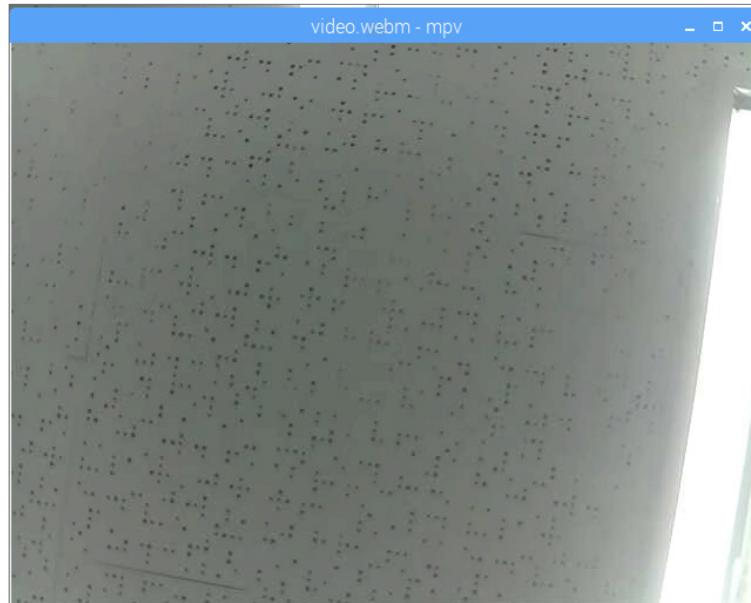
## 08 저장 위치를 home/pi 경로로 합니다.



▲ 녹화된 파일 동영상 파일 저장

## 09 탐색기를 열어서 /home/pi 폴더로 이동하여 정상적으로 저장되었는지 확인합니다.

10 video.webm 파일을 더블 클릭하여 실행합니다.



▲ 저장된 동영상 파일 실행

## 05 \_ Motion 감지 프로그램

### 05-1 Motion 감지 프로그램 세팅하기

라즈베리 파이 카메라를 활용하여 모션을 감지하고 감지된 영상을 순간적으로 캡쳐할 수 있는 Motion에 대해서 배워 보도록 하겠습니다. 영상은 사진의 연속입니다. 예를 들어 24frame은 1초에 24장에 사진이 출력된다는 의미입니다. Motion은 전 사진과 후 사진을 비교하여 사진의 다른 부분을 캐치하는 기술입니다.

Motion을 이용해서 웹 스트리밍을 해보고, 모션을 감지해서 사진으로 저장하고, 모션이 감지되는 객체를 추적해보겠습니다.

#### (1) 모션 설치하기

```
$ sudo apt-get install motion
```

#### (2) motion.conf 설정파일 수정하기

motion.conf 설정파일 수정은 다음 URL을 참고하였습니다.

- [https://motion-project.github.io/motion\\_config.html](https://motion-project.github.io/motion_config.html)

```
$ sudo nano /etc/motion/motion.conf
```

**01** daemon을 활성화 합니다. 기본 값은 off이고 on으로 설정합니다. 설정하고 motion을 시작하면 daemon 모드로 실행됩니다.

```
# Start in daemon (background) mode and release terminal (default: off)
daemon on
```

▲ daemon on 설정

**02** framerate는 2로 설정합니다. 이 값이 늘어날 수록 초당 캡처할 수 있는 사진이 늘어나게 됩니다. 최소값을 유지해줍니다. framerate가 늘어나면 늘어날수록 CPU부하가 많이 발생할 수 있습니다.

```
# Maximum number of frames to be captured per second.
# Valid range: 2-100. Default: 100 (almost no limit).
framerate 2
```

▲ framerate 2로 설정

#### 프레임 속도

- 유형 : 정수
- 범위 / 유형 : 2 - 100
- 기본값 : 15

카메라에서 초당 캡처 할 수 있는 최대 프레임 수입니다. 카메라에서 사진을 가져 오는 속도가 빨라질수록 더 많은 CPU 부하가 발생하고 동작이 깃거나 때 더 많은 그림이 포함됩니다. 프레임 속도가 2 미만으로 설정된 경우 모션이 사진 저장을 중지합니다. 이 매개 변수를 이미지 또는 동영상으로 저장하려는 초당 이미지의 최대 수로 설정하십시오. 간격을 1 초보다 길게 설정하려면 'minimum\_gap' 옵션을 대신 사용하십시오.

▲ 프레임 속도 설명

### 03 모션이 감지되었을 때 사진을 저장할 경로를 설정합니다.

```
# Target base directory for pictures and films  
# Recommended to use absolute path. (Default: current working directory)  
target_dir /home/pi/motion
```

▲ motion을 감지하면 저장될 경로 선택

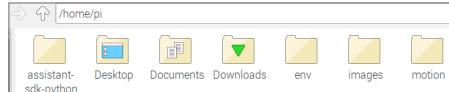
```
target_dir  
▪ 유형 : 문자열  
▪ 범위 / 유효 값 : 최대 4095자  
▪ 기본값 : 정의되지 않음 = 현재 디렉토리
```

저장할 그림 및 동영상 파일의 대상 디렉토리에 대한 전체 경로입니다. 기본값은 현재 디렉토리입니다. 모든 스냅샷, 그림 파일 및 동영상 파일의 대상 디렉토리입니다. 일반적으로 항상이 매개 변수를 절대 경로로 지정하려고 합니다.

▲ 저장될 타겟 경로 설명

### 04 /home/pi/motion 폴더를 생성해둡니다.

```
$ cd /home/pi  
$ mkdir motion
```



▲ motion 폴더 생성

### 05 locate\_motion\_mode를 on으로 설정합니다. 움직이는 객체를 감지할 때 네모 박스를 그려줍니다.

preview로 설정하면 움직이는 객체를 감지할 때 네모 박스를 그려주는 것은 동일하나 저장된 사진에는 네모 박스가 생기지 않습니다.

```
# Locate and draw a box around the moving object.  
# Valid values: on, off, preview (default: off)  
# Set to 'preview' will only draw a box in preview_shot pictures.  
locate_motion_mode on
```

▲ locate\_motion\_mode on 설정

```
locate_motion_mode  
▪ 유형 : 불연속 문자열  
▪ 범위 / 유효 값 : on, off, preview  
▪ 기본값 : off
```

움직이는 물체를 찾아 상자를 그립니다. '미리보기'값을 사용하면 저장된 미리보기 JPEG 이미지에만 상자가 그려지며 저장된 동영상에는 상자가 그려지지 않습니다.

▲ locate\_motion\_mode 설명

### 06 locate\_motion\_style을 redbox를 설정합니다. 움직이는 객체를 감지할 때 박스 색깔이 빨강으로 변경됩니다.

```
# Set the look and style of the locate box if enabled.  
# Valid values: box, redbox, cross, redcross (default: box)  
# Set to 'box' will draw the traditional box.  
# Set to 'redbox' will draw a red box.  
# Set to 'cross' will draw a little cross to mark center.  
# Set to 'redcross' will draw a little red cross to mark center.  
locate_motion_style redbox
```

▲ locate\_motion\_style redbox 설정

```
locate_motion_style
```

- 유형 : 불연속 문자열
- 범위 / 유효 값 : 상자, 빨강 상자, 십자가, 적색
- 기본값 : 상자

사용 가능한 경우 위치 지정 상자의 모양과 스타일을 설정하십시오.

▲ locate\_motion\_style 설명

**07** stream\_port를 8081로 설정합니다. http://localhost:8081로 이동하게 되면 웹 스트리밍 화면을 볼 수 있습니다.

- stream\_motion을 on으로 설정하면 웹 스트리밍 중에도 motion을 감지할 수 있습니다.
- stream\_localhost를 off로 설정하면 외부에서도 접근이 가능합니다. on이면 localhost로만 접근이 가능합니다.

```
stream_port 8081
# Quality of the jpeg (in percent) images produced (default: 50)
stream_quality 50

# Output frames at 1 fps when no motion is detected and increase to the
# rate given by stream_maxrate when motion is detected (default: off)
stream_motion on

# Maximum framerate for stream streams (default: 1)
stream_maxrate 1

# Restrict stream connections to localhost only (default: on)
stream_localhost off
```

▲ stream\_port 8081 설정

stream\_localhost  
• 유형 : 부울  
• 범위 / 유형 값 : on, off  
• 기본값 : 캐치  
스트리밍에 대한 액세스를 로컬 호스트로 제한합니다. 이 값을 on으로 설정하면 스트리밍이 Motion이 실행되는 동일한 시스템에서만 액세스 할 수 있습니다.

▲ stream\_port 설명

**08** webcontrol\_port를 8082로 설정합니다. webcontrol\_localhost를 off한 이유는 외부에서도 접근을 가능하게 하기 위함입니다.

```
# TCP/IP port for the http server to listen on (default: 0 = disabled)
webcontrol_port 8082

# Restrict control connections to localhost only (default: on)
webcontrol_localhost off

# Output for http server, select off to choose raw text plain (default: on)
webcontrol_html_output on
```

▲ webcontrol\_port 8082 설정

http://localhost:8082로 접속하게 되면 웹 브라우저에서 motion을 설정할 수도 있고 제어할 수도 있게 됩니다. 쉽게 말하면 motion.conf 파일에 접근하지 않고 웹을 통해서 motion.conf도 제어할 수 있고, motion을 종지시키는 등의 제어를 할 수 있게 됩니다.



← → ⌂ ⌄ ⓘ 주의 요함 | 192.168.0.80:8082/0/config/list  
앱 코딩스페셜리스트 getinthere님의블로그 getinthere 티스토리  
[← back](#)

**Camera 0**

- daemon = on
- process\_id\_file = /var/run/motion/motion.pid
- setup\_mode = off
- camera\_name = (not defined)
- logfile = /var/log/motion/motion.log
- log\_level = 6
- log\_type = all
- videodevice = /dev/video0

▲ webcontrol 웹 접근 화면

Ctrl + X, Y 를 통해 저장 후 nano 에디터를 빠져나옵니다.

## 05-2 Motion 감지 프로그램 실행하기

### (1) Motion 실행하기

#### 01 motion service를 시작합니다.

motion.conf 파일을 수정하면 motion service를 항상 재시작해야 변경된 설정이 적용됩니다. motion을 중지 시킨 뒤 시작하겠습니다. 시작 후 아무런 메시지가 없으면 정상적으로 작동된 것입니다.

```
$ sudo service motion stop  
$ sudo service motion start
```

- Start the Motion service sudo service motion start
- Stop the Motion service sudo service motion stop
- Restart the Motion service sudo service motion restart

▲ motion 명령어 3가지

#### 02 motion 백그라운드로 실행합니다. 옵션 -b를 이용해서 백그라운드 daemon mode로 실행시켜 보겠습니다.

```
$ sudo motion -b
```

```
SYNOPSIS motion [ -hbnsm ] [ -c config_file_path ] [ -d level ] [ -k level ] [ -p pid_file ] [ -l log_file ]  
◦ -c : Full path and filename of config file.  
◦ -h : Show help screen.  
◦ -b : Run in daemon mode  
◦ -n : Run in non-daemon mode  
◦ -s : Run in setup mode. Also forces non-daemon mode.  
◦ -d : Run with message log level 1 - 9  
◦ -k : Run with message log type 1 - 9  
◦ -l : Full path and file name for log file  
◦ -p : Full path and file name for the process id file  
◦ -m : Start in pause mode
```

▲ motion 명령어 옵션

#### 03 정상적으로 daemon0| 작동하고 있는지 확인해볼 필요가 있습니다.

```
$ netstat -nlpt
```

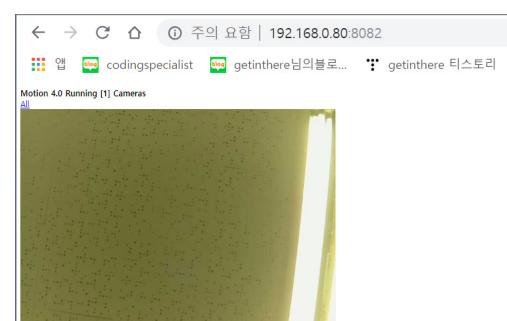
```
pi@raspberrypi:/etc/motion $ netstat -nlpt  
(Not all processes could be identified, non-owned process info  
will not be shown, you would have to be root to see it all.)  
Active Internet connections (only servers)  
Proto Recv-Q Send-Q Local Address           Foreign Address         State  
PID/Program name  
tcp        0      0 0.0.0.0:5900             0.0.0.0:*              LISTEN  
tcp        0      0 0.0.0.0:8081             0.0.0.0:*              LISTEN  
tcp        0      0 0.0.0.0:8082             0.0.0.0:*              LISTEN  
tcp        0      0 0.0.0.0:22              0.0.0.0:*              LISTEN  
tcp6       0      0 ::1:5900              :::*                  LISTEN  
tcp6       0      0 ::1:22               :::*                  LISTEN
```

▲ motion 포트 실행 중 확인

8081포트와 8082포트가 LISTEN 상태이면 완료입니다. 이제 확인해보겠습니다.

### (2) webcontrol 확인하기

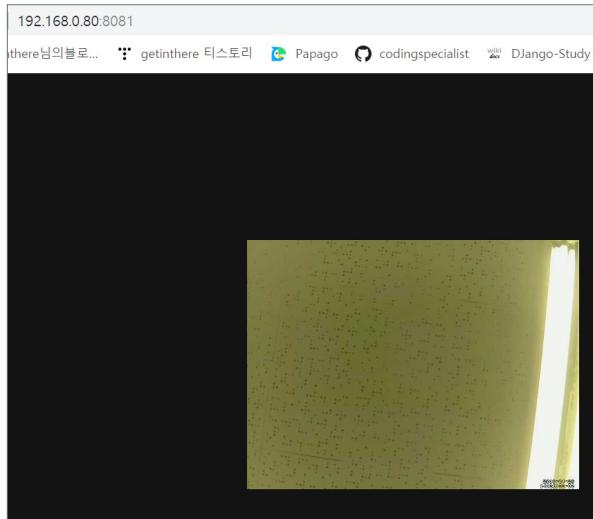
라즈베리 파이에서 브라우저를 열어서 확인한다면 localhost:8082를 입력합니다. 외부에서 작동시키려면 ifconfig 명령어를 통해서 ip를 확인한 뒤 ip를 입력하여 192.168.0.80:8082를 입력합니다.



▶ motion 스트리밍 확인 1

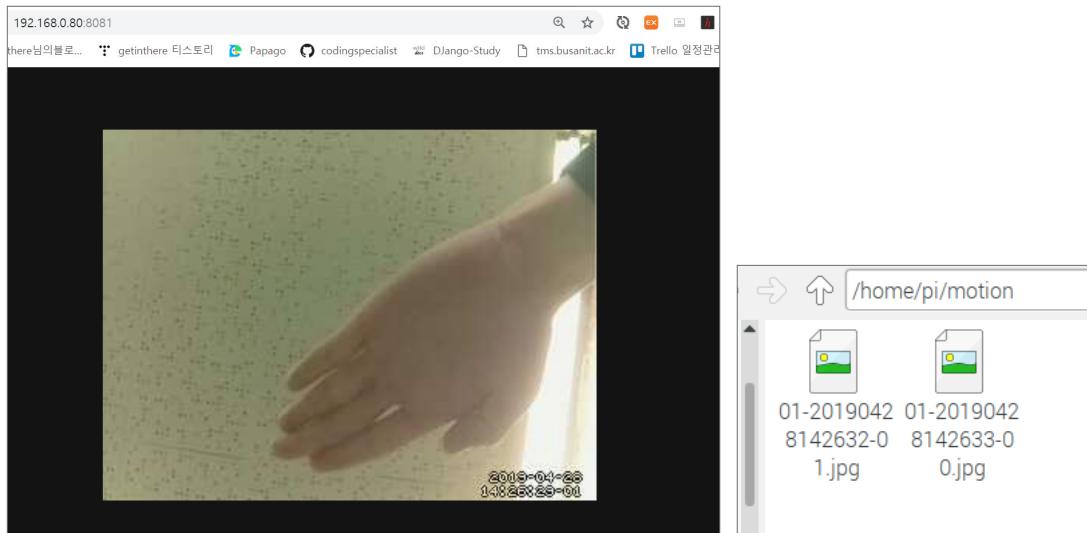
### (3) 영상 스트리밍을 확인하기

01 http://localhost:8081 주소로 이동합니다.



▲ motion 스트리밍 확인 2

02 locate\_motion\_mode가 off로 되어 있는 상태에서 motion 감지를 확인해보면 /home/pi/motion 폴더에 서 감지된 사진이 저장됩니다.



▲ motion 감지

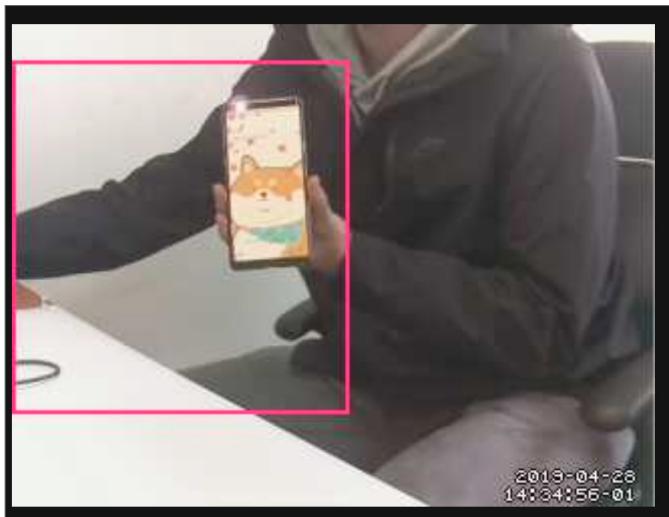
▲ motion 감지된 사진 파일

**03** 감지된 사진을 확인합니다.



▲ 감지된 사진 열기

**04** locate\_motion\_mode가 on으로 되어 있는 상태에서 motion 감지를 확인합니다.



▲ redbox 확인하기

**05** 모션이 확인되면 motion 폴더에 사진이 저장됩니다. 저장된 사진을 확인합니다.