PIAI 인공지능연구원
POSTECH Institute of Artificial Intelligence

AI with IoT

PART III. Raspberry Pi & Sensor BASICS

라즈베리 파이란?

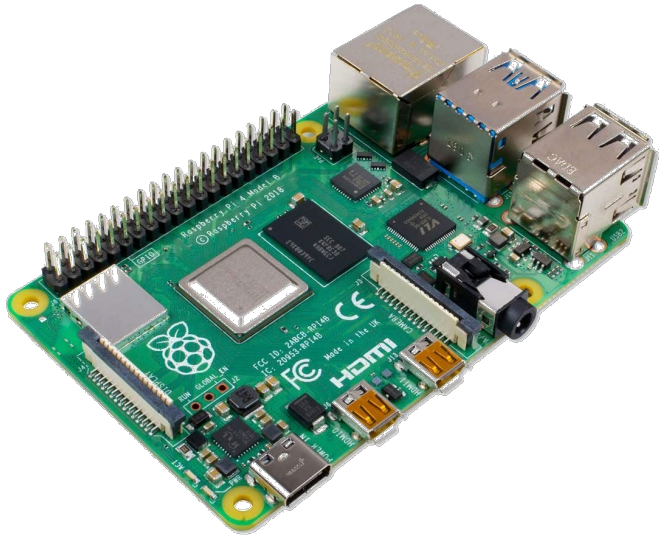
Raspberry Pi 4

싱글 보드 컴퓨터(Single-board computer, SBC)의 일종

- 단일보드(PCB) 위에 중앙처리장치(CPU), 메모리, I/O 장치 등이 장착된 컴퓨터
- 관리가 쉽고 성능대비 저렴한 가격, 저전력, 초소형의 특징을 지님

- 리눅스 계열의 OS 지원: Raspbian OS 사용

- GPIO (General-Purpose Input/Output) 지원
- 각종 센서 및 소자들을 C 또는 파이썬 언어를 통해 제어 가능



라즈베리 파이란?

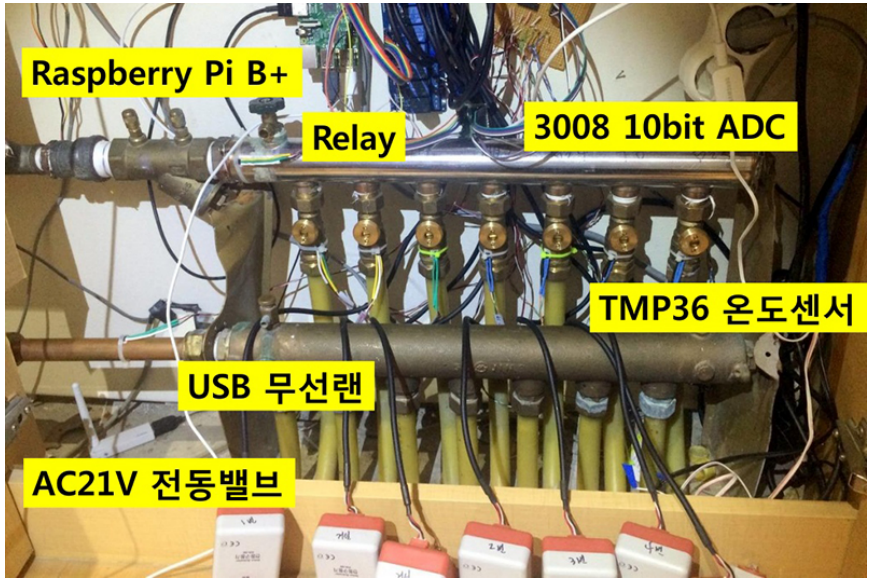
Raspberry Pi 4 H/W Spec.

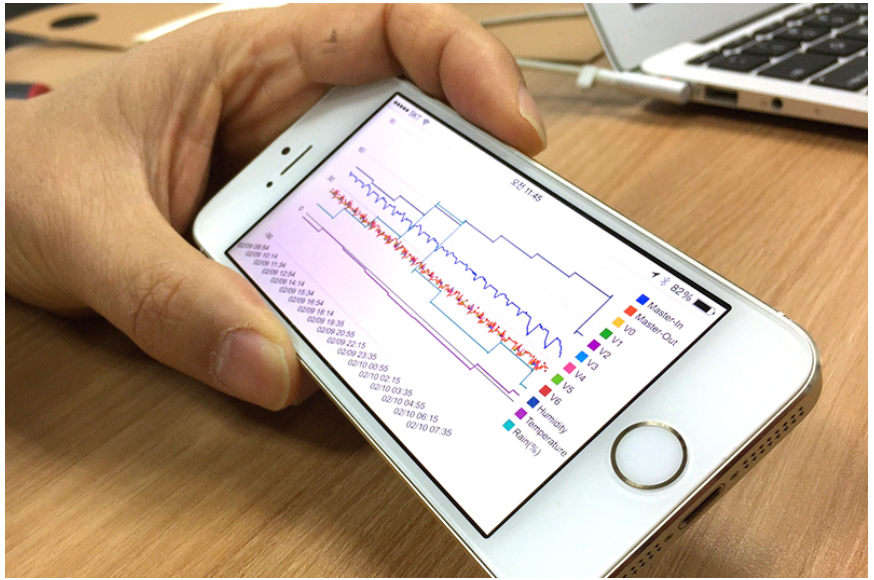
Raspberry Pi 4 Model B	H/W System	Broadcom BCM2711, Quad core Cortex-A72 (ARMv8) 64-bit SOC @ 1.5GHz
		1 GB LPDDR4-3200 SDRAM
		Extended 40-pin GPIO header
		5V/3A DC power input
	Network Interface	2.4 / 5 GHz IEEE 802.11ac wireless LAN
		Bluetooth 5.0, BLE / Gigabit Ethernet
	I/O ports	CSI camera port for connecting Raspberry Pi camera
		DSI display port for connecting a Raspberry Pi touchscreen
		2 micro-HDMI ports, 2 USB 3.0 & 2 USB 2.0 ports
		Micro SD card slot for loading OS and storing data
		4-pole stereo output and composite video port

라즈베리 파이의 활용 I

라즈베리 파이를 사용한 아파트 난방비 절감 시스템

- 난방 밸브 조절을 통한 물의 온기 효과적 사용
- 일기예보 데이터 획득 – 날씨에 따른 난방 정도 조절





< 출처: Bloter, 아파트 난방코딩: 이런게 IoT 아닌가요 >

라즈베리 파이의 활용 II

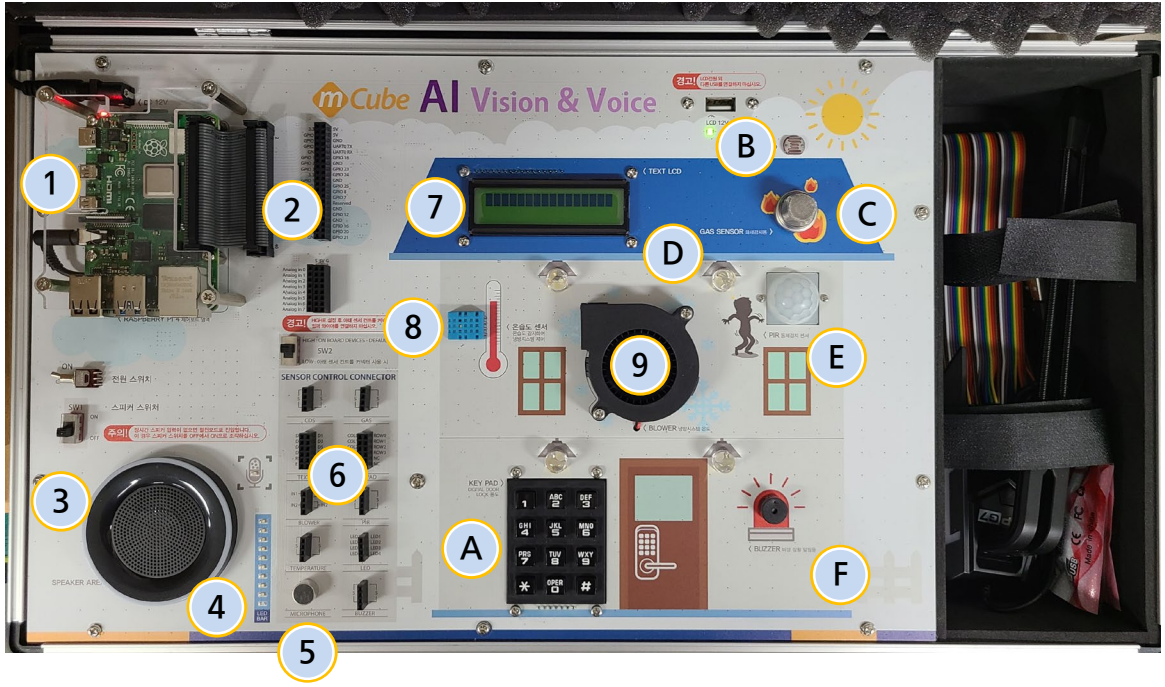
라즈베리 파이 기반 어항 관리 장치

- 어항 상태 모니터링 – 정보의 스마트 기기 전송
- 수온/산소량 관리 및 먹이 배급 장치 작동

< 출처: HAYDEN KIBBLE, The Internet Enabled Fishtank >

mCube-AI 키트 소개

mCube-AI 키트 구성



구분	장치
1	Raspberry Pi 4
2	Raspberry Pi 4 I/O Port
3	Speaker
4	Audio Level Meter
5	Microphone
6	Sensor Control Connector
7	Text LCD
8	온/습도 센서 (DHT 11)
9	Blower FAN
A	Keypad
B	조도 센서
C	가스 센서
D	LED (4 EA)
E	동작 감지 센서
F	Buzzer

인공 지능 음성 인식 시스템

mCube-AI 시스템 구성

Actions on Google

Google ASSISTANT

Speaker

Microphone

Raspberry Pi®

ACTUATOR

LED

KEYPAD

FAN

TEXT LCD

BUZZER

SENSOR

CDS

GAS

DHT11

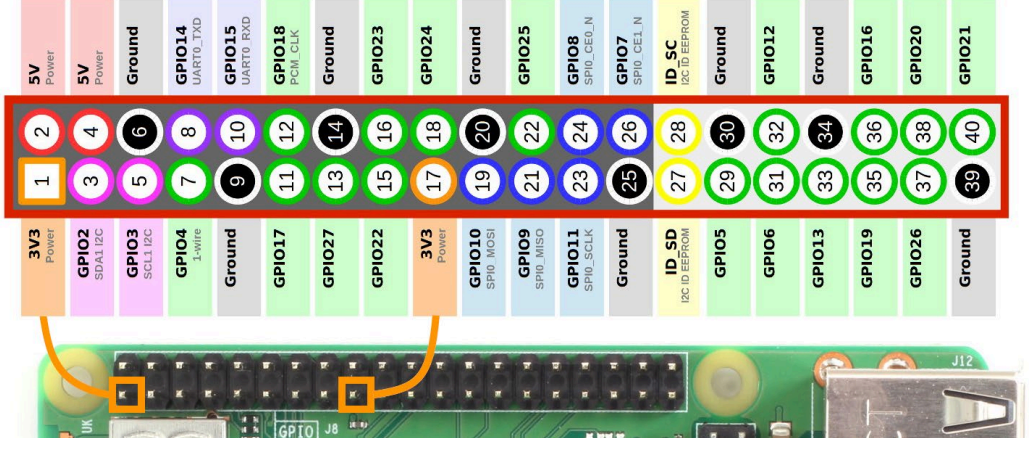
PIR

mCube-AI 키트 - GPIO

General-Purpose Input Output

범용 입출력 포트

- 주변장치 및 소자들을 동작시키는 인터페이스
- Ex. LED를 켜기 위해 해당 소자에 해당하는 포트로 신호를 인가
- 라즈베리파이는 보드 상 40개의 GPIO 핀을 제공
  (GPIO 핀을 제어할 수 있는 라이브러리는 raspbian OS에 기본 설치)



mCube-AI 키트 - GPIO

GPIO 기본 사용방법

- 가장 먼저 모듈을 import

```
>>> import RPi.GPIO as GPIO
```

- 핀 넘버를 이르는 방식을 선택

```
>>> GPIO.setmode(GPIO.BOARD)
```
// 라즈베리파이에 배열된 순서로 핀 넘버 사용

또는

```
>>> GPIO.setmode(GPIO.BCM)
```
// SOC 칩에서 사용하는 핀 넘버 사용

- 핀 모드 설정 (입력핀 또는 출력핀)

```
>>> GPIO.setup(4, GPIO.OUT)
>>> GPIO.setup([18, 19, 20], GPIO.IN, initial=GPIO.HIGH)
```
//list로 한번에 설정

- Input/Output 신호 인가

```
>>> GPIO.output(4, GPIO.HIGH)
```

- 프로그램 종료 전, 반드시 리소스 반납

```
>>> GPIO.cleanup()
```

IoT Example I – LED

발광 다이오드

Light Emitting Diode

- 순방향 전압을 가했을 때 발광하는 반도체 소자
- 전구, 형광등에 비해 전력 소모가 적고 수명이 긴 특징

LED 제어 방법

- GPIO 포트를 통해 LED가 연결된 GPIO 핀에 전압을 인가 했을 시, ON
- LED의 양극(Anode)에 전기적 신호를 통해 5V, 음극(Cathode)에 0V(접지, GND)가 유지 시, ON
- 라즈베리 파이의 CPU에서 해당 소자(GPIO 핀)에 Low(0) 또는 High(1)의 신호를 보내주는 방식
- GPIO 핀 할당 (LED 좌측 상단부터)

GPIO (BCM)	GPIO 4	LED_1
	GPIO 5	LED_2

GPIO (BCM)	GPIO 14	LED_3
	GPIO 15	LED_4

IoT Example I – LED

```python
import RPi.GPIO as GPIO
from time import sleep

LED_1 = 4

def main():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(LED_1, GPIO.OUT, initial=False)
    print("main() program running...")

    try:
        while True:
            GPIO.output(LED_1, GPIO.HIGH)
            sleep(0.5)
            GPIO.output(LED_1, GPIO.LOW)
            sleep(0.5)

    except KeyboardInterrupt:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

Exercise I – LED

LED 순차 점등 제어

1) 0개의 LED가 켜진 초기(initial) 상태
2) 0.5초 간격으로 1개의 LED가 차례로 점등(ON)
3) 4개의 LED가 켜졌을 때, 1개씩 차례로 소등(OFF)
4) 1-3의 과정이 반복

원하는 번호의 LED 제어

1) 0개의 LED가 켜진 초기(initial) 상태
2) 왼쪽 위 LED부터 시계 방향으로 번호를 매김 (1~4)
3) "LED NUMBER: " 을 출력하고, 1~4 중 입력을 받아 제어할 LED를 결정
4) "LED SET: " 을 출력하고, 'ON' 또는 'OFF' 입력을 받아 알맞은 번호의 LED 제어

IoT Example II – KeyPad

스위치(Switch)

전류의 흐름을 막거나 계속 흐르게 하는 용도의 장치

- 누름 버튼 스위치, 토글 스위치, DIP 스위치 등 다양한 종류
- mCube-AI의 키패드는 푸쉬 버튼 스위치 사용
- 버튼을 누르는 동안 전류를 흐르게 하여 신호 상태 제어

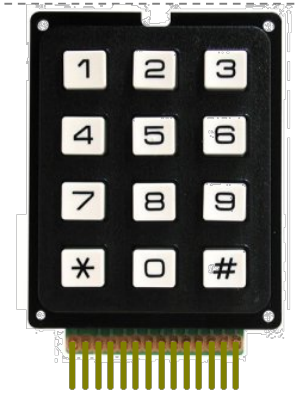
키패드 제어 방법

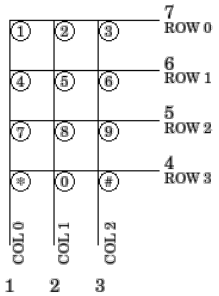
- 일반적 푸쉬 버튼 스위치: 눌림 상태에 HIGH 신호 인가, 눌리지 않은 상태에 LOW 신호 인가
- 키패드 각 버튼에 해당하는 스위치가 입력으로 작용: Input을 통해 MCU로 신호를 수신
  (<-> LED는 출력 신호를 인가하여 ON/OFF 제어)
- 푸쉬 버튼 스위치는 해당 출력 핀이 GND와 연결되어 있다가 스위치가 눌릴 경우 전류가 흘러 HIGH 상태
  MCU는 0과 1의 신호로 받아들여 동작을 제어

IoT Example II – KeyPad

매트릭스(Matrix) 구조 키패드

- 키패드의 모든 버튼을 스위치로 할당하는 경우
  : 사용되는 포트 수가 너무 많아짐!
- 포트의 수를 줄이기 위해 행렬 기반 매트릭스 구조 사용
- 각 행 또는 열 단위로 출력을 설정
  -> 행이 하나 선택되면, 몇 번째 열인지 구분
  -> 선택된 행 + 입력된 열에 해당하는 스위치 입력 판단





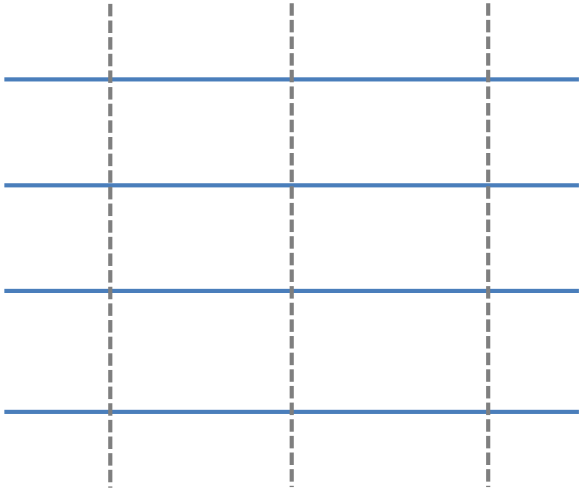
		SW_C		
		100	010	001
SW_R	1000	SW 1	SW 2	SW 3
	0100	SW 4	SW 5	SW 6
	0010	SW 7	SW 8	SW 9
	0001	SW *	SW 0	SW #

IoT Example II – KeyPad

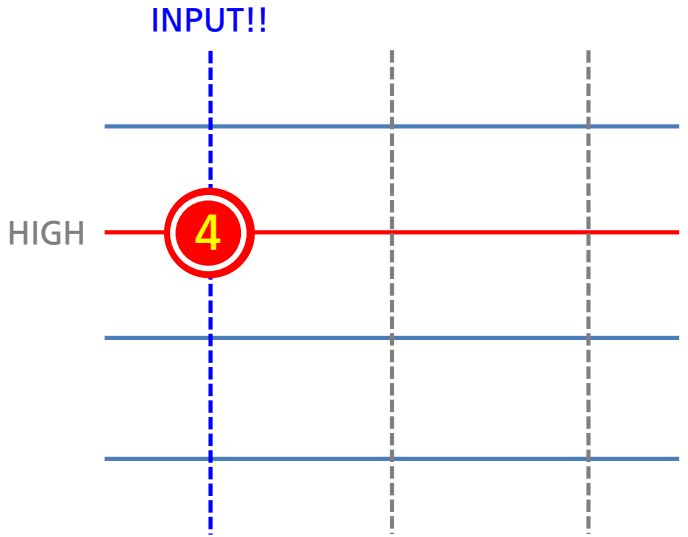
매트릭스(Matrix) 구조 키패드

입력 버튼 검출 알고리즘 구조

- 각 행에 순차적으로 HIGH 신호를 인가 (반복)



- 열 버튼에 입력된 신호 검출: 숫자 맵핑



IoT Example II – KeyPad

GPIO_EX

부족한 GPIO 핀을 확장하여 사용

- 라즈베리 파이의 GPIO v핀 2개를 사용하여 7개의 입력을 받을 수 있는 EXPANDER 사용
- 키패드의 각 행(ROW)와 각 열(COL)은 Expander에 연결되어 있음
- RPi.GPIO와 GPIO_EX 모듈을 import 해야 함!

라즈베리파이 확장 GPIO PIN (IO EXPANDER)	KEYPAD
P0	ROW0
P1	ROW1
P2	ROW2
P3	ROW3
P4	COL0
P5	COL1
P6	COL2

IoT Example II – KeyPad

```
import RPi.GPIO as GPIO
from time import sleep
import GPIO_EX

ROW0_PIN = 0
ROW1_PIN = 1
ROW2_PIN = 2
ROW3_PIN = 3
COL0_PIN = 4
COL1_PIN = 5
COL2_PIN = 6

COL_NUM = 3
ROW_NUM = 4

g_preData = 0

colTable = [COL0_PIN, COL1_PIN, COL2_PIN]
rowTable = [ROW0_PIN, ROW1_PIN, ROW2_PIN, ROW3_PIN]
```

IoT Example II – KeyPad

```
def initKeypad():
    for i in range(0, COL_NUM):
        GPIO_EX.setup(colTable[i], GPIO_EX.IN)
    for i in range(0, ROW_NUM):
        GPIO_EX.setup(rowTable[i], GPIO_EX.OUT)


def selectRow(rowNum):
    for i in range(0, ROW_NUM):
        if rowNum == (i + 1):
            GPIO_EX.output(rowTable[i], GPIO_EX.HIGH)
            sleep(0.001)
        else:
            GPIO_EX.output(rowTable[i], GPIO_EX.LOW)
            sleep(0.001)
    return rowNum
```

IoT Example II – KeyPad

```
def readCol():
    keypadstate = -1
    for i in range(0, COL_NUM):
        inputKey = GPIO_EX.input(colTable[i])
        if inputKey:
            keypadstate = keypadstate + (i + 2)
            sleep(0.5)
    return keypadstate
```

IoT Example II – KeyPad

```
def readKeypad():
    global g_preData
    keyData = -1

    runningStep = selectRow(1)
    row1Data = readCol()
    selectRow(0)
    sleep(0.001)
    if (row1Data != -1):
        keyData = row1Data

    if runningStep == 1:
        if keyData == -1:
            runningStep = selectRow(2)
            row2Data = readCol()
            selectRow(0)
            sleep(0.001)
```

IoT Example II – KeyPad

```
            if (row2Data != -1):
                keyData = row2Data + 3

    sleep(0.1)

    if keyData == -1:
        return -1

    if g_preData == keyData:
        g_preData = -1
        return -1
    g_preData = keyData

    print("\r\nKeypad Data : %d" % keyData)

    return keyData
```

PREVIOUS CODE

IoT Example II – KeyPad

```
def main():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)

    initKeypad()
    print("setup keypad pin")
    try:
        while True:
            keyData = readKeypad()


    except KeyboardInterrupt:
        GPIO.cleanup()


if __name__ == '__main__':
    main()
```

Exercise II – KeyPad & LED

KeyPad 입력 모듈 완성하기

1) 키패드의 3행, 4행의 입력이 나오도록 코드 작성
2) *, #의 입력이 제대로 동작하도록 작성
3) Hint-1: runningStep에 따른 행 선택(selectRow)와 열 입력 감지(readCol)
   Hint-2: 문자입력(*, #)은 keyData 변수 저장과 출력에 문자열에 맞게 작성

KeyPad 입력 숫자에 따른 LED 제어

1) 키패드 각 입력 [1, 2, 3, 4]에 따라 할당된 LED 점등
2) 키패드 1 입력 시, LED_1이 ON
3) LED_i 가 ON인 상태에서 키패드 i 입력 시, OFF
4) [1, 2, 3, 4] 외의 키패드 입력 시, 현재 점등된 모든 LED OFF

IoT Example III – CDS

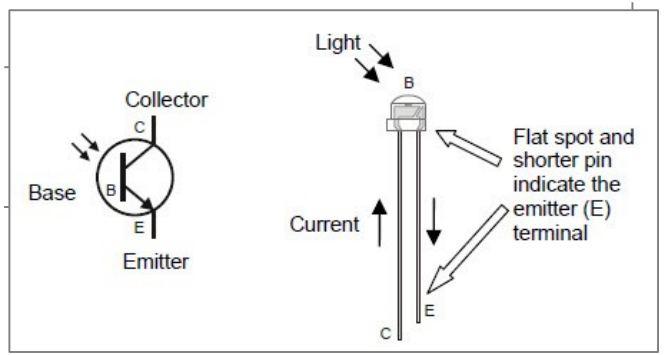
조도 센서

황화카드뮴(Cadmium Sulfide, CdS) 광 측정 센서

- 빛의 양이 어느 정도인지 확인하기 위한 용도
- Ex. 스마트폰에서 자동으로 화면 밝기를 조절하는 기능에 사용
- 기본적으로 포토 트랜지스터(Photo Transistor)의 원리로 동작

CDS 광센서

포토 트랜지스터



빛을 비추면 조도에 비례하여 광전류가 흐르는 수광소자

- 바이폴라(Bi-polar) 트랜지스터 구조
- 베이스 단자가 전기적으로 단절
- 광량이 많아지는 경우, 베이스 전류가 공급: 컬렉터 – 에미터 간 전류 양이 증가
- 광량에 비례하여 변화하는 전류량을 측정

IoT Example III – CDS

CDS 조도 센서 제어 방법

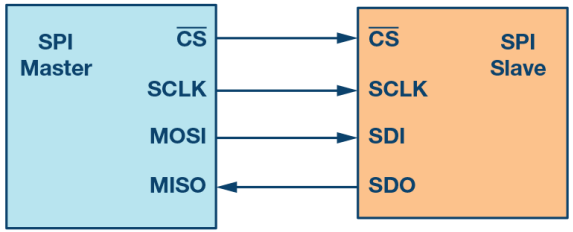
- 측정된 조도를 해당하는 핀으로 출력
- 출력 값은 아날로그(Analog)
- ADC (Analog-to-Digital Converter)를 통해 디지털 값으로 변환하여 확인 가능
- 별도의 ADC (MCP3208)을 통해 변환된 디지털 전압 값을 라즈베리 파이에서 수신
- 빛의 양이 많을 경우, CDS value (즉, 변환 전압)이 높게 측정

MCP3208

SPI 인터페이스

Serial Peripheral Interface

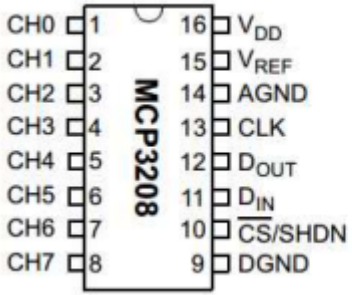
- 두 개의 주변장치 간 직렬통신으로 데이터를 교환하는 인터페이스
- Full-duplex 방식(양방향 동시전송)으로 동작
- CPU – 주변 디바이스/프로세서 간 통신에 활용
- MCP320 ADC 프로세서와 라즈베리 파이가 통신하기 위해 사용



IoT Example III – CDS

MCP3208 ADC [1/2]

- 8 CH (8개의 장치 연결 가능), 12 bit 데이터 통신
- 2개의 제어 모드 (SingleEnded, Diff)
  SingleEnded: 하나의 장치에서 값을 수신
  Diff.: 두 장치의 차이 값을 수신
- 원하는 장치로부터 값을 수신하기 위해 연결 채널 설정



Name	Function
V_{DD}	+2.7V to 5.5V Power Supply
DGND	Digital Ground
AGND	Analog Ground
CH0-CH7	Analog Inputs
CLK	Serial Clock
D_{IN}	Serial Data In
D_{OUT}	Serial Data Out
$\overline{CS}$/SHDN	Chip Select/Shutdown Input
V_{REF}	Reference Voltage Input

MCP3208 Pin 배열 및 기능

CONTROL BIT SELECTIONS				INPUT CONFIGURATION	CHANNEL SELECTION
SINGLE/$\overline{DIFF}$	D2	D1	D0		
1	0	0	0	single ended	CH0
1	0	0	1	single ended	CH1
1	0	1	0	single ended	CH2
1	0	1	1	single ended	CH3
1	1	0	0	single ended	CH4
1	1	0	1	single ended	CH5
1	1	1	0	single ended	CH6
1	1	1	1	single ended	CH7

MCP3208 채널 선택 제어 비트

IoT Example III – CDS

MCP3208 ADC [2/2]

- 전송 데이터 포맷: 3 바이트로 구성

StartBit

1st Byte	0	0	0	0	0	1	S/D	D2
2nd Byte	D1	D2	X	null	B11	B10	B9	B8
3rd Byte	B7	B6	B5	B4	B3	B2	B1	B0

```
uint8_t buf[3];

buf[0] = 0x06 | ((aChAddr & 0x07) >> 2)
buf[1] = ((aChAddr & 0x07) << 6)
buf[2] = 0x00
```

- 채널 선택 이후, 2번째 바이트 하위 니블과 3번째 바이트에 수신 데이터 존재

```
buf[1] = 0x0F & buf[1]          // masking lower nibble
Val = (buf[1] << 8) | buf[2]    // 12 bits by concatenation
```

IoT Example III – CDS

CDS 기본 예제

```
import RPi.GPIO as GPIO
from time import sleep
import spidev

spi = spidev.SpiDev()
CDS_CHANNEL = 0


def initMcp3208():
    spi.open(0, 0)  # open(bus,device), device 0 - CE0(GPIO8), device 1 - CE1(GPIO7)
    spi.max_speed_hz = 1000000  # set 1MHz
    spi.mode = 3  # set 0b110
```

- spi.open: 라즈베리 파이와 MCP3208은 하나의 버스(0) 공유
- spi.mode: spi 동작 모드 설정 (상승 엣지, 하강 엣지, 클럭 전단 혹은 후단)

IoT Example III – CDS

CDS 기본 예제

```
def buildReadCommand(channel):

    startBit = 0x04
    singleEnded = 0x08

    configBit = [startBit | ((singleEnded | (channel & 0x07)) >> 2), (channel & 0x07) << 6, 0x00]

    return configBit


def processAdcValue(result):
    '''Take in result as array of three bytes.
       Return the two lowest bits of the 2nd byte and
       all of the third byte'''
    byte2 = (result[1] & 0x0F)
    return (byte2 << 8) | result[2]
```

IoT Example III – CDS

CDS 기본 예제

```
def analogRead(channel):
    if (channel > 7) or (channel < 0):
        return -1

    r = spi.xfer2(buildReadCommand(channel))
    adc_out = processAdcValue(r)
    return adc_out


def controlMcp3208(channel):
    analogVal = analogRead(channel)
    return analogVal


def readSensor(channel):
    return controlMcp3208(channel)
```

IoT Example III – CDS

```
def main():
    GPIO.setmode(GPIO.BCM)
    initMcp3208()
    print("Setup pin as outputs")

    try:
        while True:
            readVal = readSensor(CDS_CHANNEL)

            voltage = readVal * 4.096 / 4096
            print("CDS Val=%d\tVoltage=%f" % (readVal, voltage))
            sleep(0.5)
    except KeyboardInterrupt:
        GPIO.cleanup()
        spi.close()


if __name__ == '__main__':
    main()
```

Exercise III – CDS & LED

CDS 광량 측정에 따른 LED 제어

1) CDS 측정 voltage 또는 측정 CDS value를 4단계 threshold로 구분
2) Threshold 범위에 따라 LED 점등 개수 결정
3) 가장 어두운 광량 범위: 4개의 LED 점등
   가장 밝은 광량 범위: 0개의 LED 점등

IoT Example IV – FAN

Blower FAN

바람이 나오는 출구를 두어 한쪽 방향으로 바람을 만들어 냄
- 전력을 공급하여 날개를 회전
- 출구로 바람을 내보내어 열을 식히는 용도

Blower FAN 

FAN 제어 방법

- GPIO 포트를 통해 FAN에 연결된 IN1 핀에 전압을 인가했을 시, ON
- IN2 핀에는 항상 Low 신호를 보내주어야 함 (GND 역할)
- LED 동작과 동일한 구조
  : GPIO 핀에 OUT 할당 후, 신호 인가

GPIO (BCM)	GPIO 18	FAN _IN1
	GPIO 27	FAN_IN2

IoT Example IV – FAN

```python
import RPi.GPIO as GPIO
from time import sleep

ON = 1
OFF = 0

FAN_PIN1 = 18
FAN_PIN2 = 27

def onFan():
    GPIO.output(FAN_PIN1, GPIO.HIGH)
    GPIO.output(FAN_PIN2, GPIO.LOW)

def offFan():
    GPIO.output(FAN_PIN1, GPIO.LOW)
    GPIO.output(FAN_PIN2, GPIO.LOW)
```

IoT Example IV – FAN

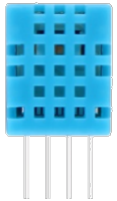
```
def main():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(FAN_PIN1, GPIO.OUT, initial=False)
    GPIO.setup(FAN_PIN2, GPIO.OUT, initial=False)
    print("Setup FAN pin as outputs")
    print("main() program")
    try:
        while True:
            onFan()
            sleep(3.0)
            offFan()
            sleep(1.0)
    except KeyboardInterrupt:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

IoT Example V – DHT11

DHT11 온/습도 측정 센서

온도 센서 동작 원리

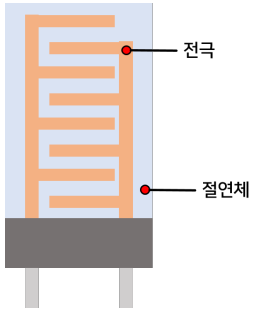
- 써미스터(Thermistor)를 사용한 온도 측정
- 온도에 따라 저항이 변화하는 소자
- 일정 전압이 인가될 때, 변화하는 저항에 따른 전류 측정 후 온도 값으로 변환



DHT11

습도 센서 동작 원리

- 저항 감지형 습도 측정 센서
- 양 전극이 서로 연결이 되어 있지 않으나 수분을 통해 미세 전류가 흐르는 구조
- 습도가 오를수록 미세 전류량에 따라 변화된 저항 값으로 습도를 측정



저항 감지형 습도 측정계

Adafruit(제조사)의 라이브러리 사용

- 전류 변화 및 저항 변화를 각각 온도/습도로 치환해주는 라이브러리

IoT Example V – DHT11

DHT11 제어 방법

```
라이브러리 및 필수 패키지 설치 확인
>>> sudo apt update
>>> sudo apt upgrade
>>> sudo apt install python3-pip
>>> sudo apt install –y python-smbus
>>> sudo apt install –y i2c-tools
>>> sudo raspi-config (인터페이스 옵션: I2C enable 확인)
>>> pip3 install RPI.GPIO
>>> pip3 install adafruit-blinka
>>> pip3 install adafruit-circuitpython-dht
>>> pip3 install adafruit-circuitpython-charlcd
>>> sudo apt install libgpiod2
```

IoT Example V – DHT11

```
import board
import adafruit_dht
from time import sleep

dhtDevice = adafruit_dht.DHT11(board.D17)

def main():
    print("main() program")
    while True:
        try:
            temperature_c = dhtDevice.temperature
            humidity = dhtDevice.humidity
            print("Temp: {:.1f}C    Humidity: {}%".format(temperature_c, humidity))
        except RuntimeError as error:
            print(error.args[0])
        sleep(2.0)

if __name__ == '__main__':
    main()
```

Exercise IV – FAN & DHT11

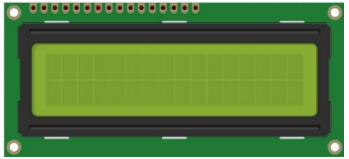
온도 및 습도 조건에 따른 팬 구동

1) DHT11에서 온도와 습도를 1초마다 측정
2) 온도가 28도 이상이거나(or) 습도가 60% 이상일 시, FAN 구동
3) 다시 온도와 습도가 일정치 이하로 내려가면, FAN OFF

IoT Example VI – Text LCD

Text LCD

문자를 표시하는 시각적 전달 장치
- 액정을 이용하여 화소에 도달하는 빛을 선택적으로 투과시키거나 차단함
- 간단한 문자 표시가 필요한 임베디드 장치에서 사용됨



Text LCD 제어 방법

Text LCD를 초기화하고 제어신호 타이밍에 맞추어 명령 및 데이터 신호 전송
- 제어 신호는 RS, R/W, E 핀을 사용
- 데이터 신호는 DB0-DB7번 핀을 사용
- GPIO를 통해 명령을 전달

IoT Example VI – Text LCD

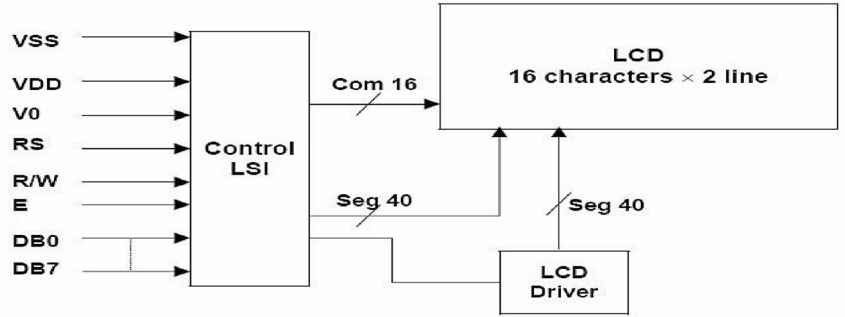
Text LCD 구조

Cotrol LSI: 다양한 Input에 맞추어 LCD 제어 정보를 저장하는 대규모 집적 회로

- 명령 레지스터: 화면 초기화, 커서 초기화 등의 제어 명령을 저장
- 데이터 레지스터: DDRAM과 CGRAM에 쓰고 읽은 데이터를 일시적으로 저장
- BF: 1이면 LCD Controller가 동작 중으로 명령 수행 불가능함을 알리고, 0이면 다음 명령 수행 가능함을 알림
- DDRAM (Data Display RAM): 표시될 문자의 아스키코드가 저장되어 있는 메모리
- CGRAM (Character Generator RAM): 사용자가 원하는 문자를 만들기 위해 사용하는 메모리

LCD: 정해진 문자를 출력하는 표시부

LCD Driver: 문자열 출력과 LCD 표시부 제어를 위한 명령을 수행



IoT Example VI – Text LCD

출력 예정 Text 저장 위치 결정

디스플레이 (Display) 화면의 각 행과 열의 위치에는 고유한 주소 (Address) 값이 부여됨

- 실습에 사용되는 장비는 16x2 디스플레이로 구성됨
- 행과 행 사이의 어드레스가 연속되어 있지 않음

Display Position	1	2	3	4	5	.....	15	16
DDRAM Address (hexadecimal)	00	01	02	03	04	···..	0E	0F
	40	41	42	43	44	···..	4E	4F

Text 출력 문자 변환과 저장

- 출력하고자 하는 문자를 ASCII 코드로 변환하여 DDRAM에 저장

Lower 4 Bits \ Upper 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	@	P	`	p				—	9	ミ	α	p
xxxx0001	(2)		!	1	A	Q	a	q			。	ア	チ	ム	ä	q

IoT Example VI – Text LCD

GPIO Mapping

제어 방법이 다소 복잡하고 소스 코드의 양이 많아서, 실습 과정에서는 라이브러리를 활용할 예정

- 제어와 데이터 전송에 필요한 PIN을 GPIO를 통해 연결
- 16열로 구성되어 있으므로, 4개의 Data bus만 활용 (D4 – D7)

라즈베리파이 GPIO PIN (BCM)	TEXT LCD
GPIO 22	LCD_RS
GPIO 23 (GND)	LCD_RW [option]
GPIO 24	LCD_E
GPIO 19	LCD_D4
GPIO 20	LCD_D5
GPIO 26	LCD_D6
GPIO 21	LCD_D7

IoT Example VI – Text LCD

```
import board
import digitalio
import adafruit_character_lcd.character_lcd as character_lcd
from time import sleep

# Raspberry Pi pin setup
lcd_rs = digitalio.DigitalInOut(board.D22)
lcd_en = digitalio.DigitalInOut(board.D24)
lcd_d7 = digitalio.DigitalInOut(board.D21)
lcd_d6 = digitalio.DigitalInOut(board.D26)
lcd_d5 = digitalio.DigitalInOut(board.D20)
lcd_d4 = digitalio.DigitalInOut(board.D19)

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 16
lcd_rows = 2

lcd = character_lcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7, lcd_columns, lcd_rows)
```

IoT Example VI – Text LCD

```
def initTextlcd():
    lcd.clear()
    lcd.home()
    lcd.cursor_position(0,0)
    sleep(1.0)      # Wait 1 seconds

def displayText(text='',col=0,row=0):
    lcd.cursor_position(col, row)
    lcd.message = text

def clearTextlcd():
    lcd.clear()
    lcd.message = 'clear LCD\nGoodbye!'
    sleep(2.0)
    lcd.clear()
```

IoT Example VI – Text LCD

```
def main():
    initTextlcd()
    print("start textlcd program ...")

    try:
        while(1):
            line = 'Hello World\n'
            lcd.clear()
            displayText(line,0,0)
            sleep(3)

    except KeyboardInterrupt:
        clearTextlcd()


if __name__ == '__main__':
    main()
```

Exercise V – Text LCD

입력 받은 숫자 출력 및 비밀번호 일치 여부 확인

1) 입력 받은 4자리 숫자를 첫 줄에 출력
2) 두 번째 줄에 입력 받은 숫자가 미리 설정된 비밀번호와 일치하는 지 확인
3) 일치할 경우 CORRECT를, 불일치할 경우 FAIL을 출력

온도 및 습도 출력

1) DHT11을 통해 구한 온도와 습도 값을 Text LCD에 출력
2) 첫 줄에 Temp. (온도) C
3) 둘 째 줄에 Humidity (습도) %