
임베디드 시스템

ADC 및 가변저항 제어

In-Hyeok Kang

M23522@hallym.ac.kr

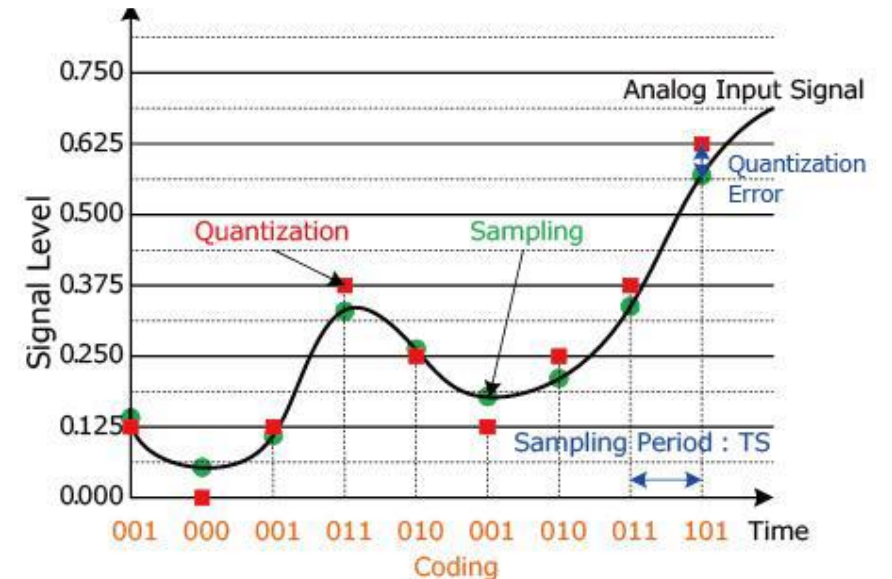
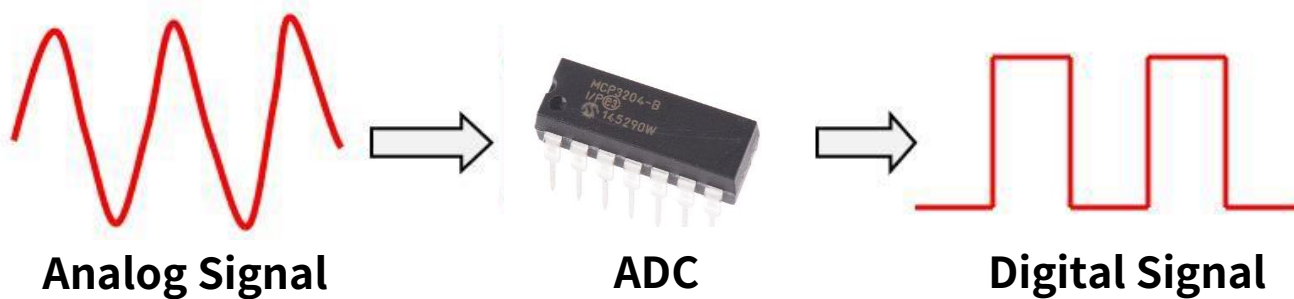
연구실: 공학관 1321호

Contents

1. ADC (Analog to Digital Converter)
2. Raspberry Pi SPI 인터페이스 활성화
3. ADC 및 가변저항 연결
4. ADC 및 가변저항 제어

ADC (Analog to Digital Converter)

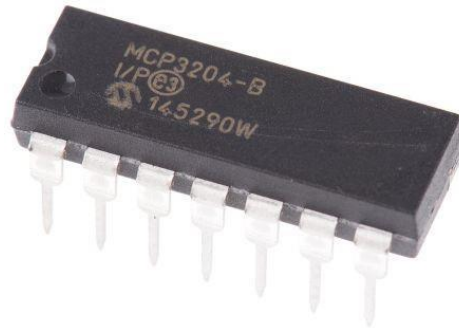
- ADC (Analog to Digital Converter)
 - 연속적인 아날로그 신호를 디지털 신호로 변환하는 장치로, 집적 회로(Integrated Circuit, I/C)로 구현됨
 - 아날로그 신호를 디지털 신호로 변환하는 과정
 - ✓ 표본화(Sampling): 시간을 기준으로 연속적인 아날로그 신호를 이산값으로 변환하는 과정
 - ✓ 양자화(Quantization): 진폭을 기준으로 연속적인 아날로그 신호를 이산값으로 변환하는 과정
 - ✓ 부호화(Encoding): 표본화 및 양자화를 통해 얻어진 값을 2진수로 바꾸는 과정



ADC (Analog to Digital Converter)

- MCP3204 – Datasheet 참고

- **아날로그 신호를 12-bit의 디지털 신호로 변환**하는 ADC 칩 ①
- 4-channel 12-bit ADC ②
ex) 0 ~ 3.3V의 센서 출력값을 0~4095(12-bit)의 디지털 값으로 변환
- **SPI 시리얼 통신 지원** ③
- 2.7V ~ 5.5V 동작 ④



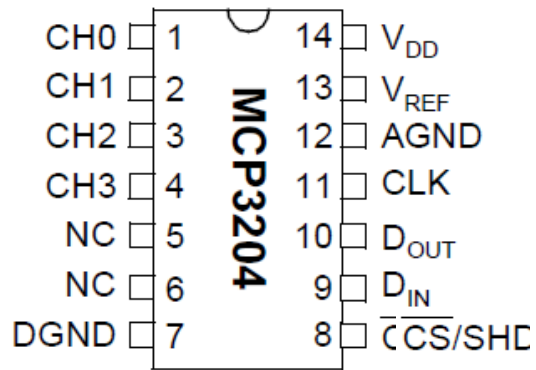
MCP3204

FEATURES

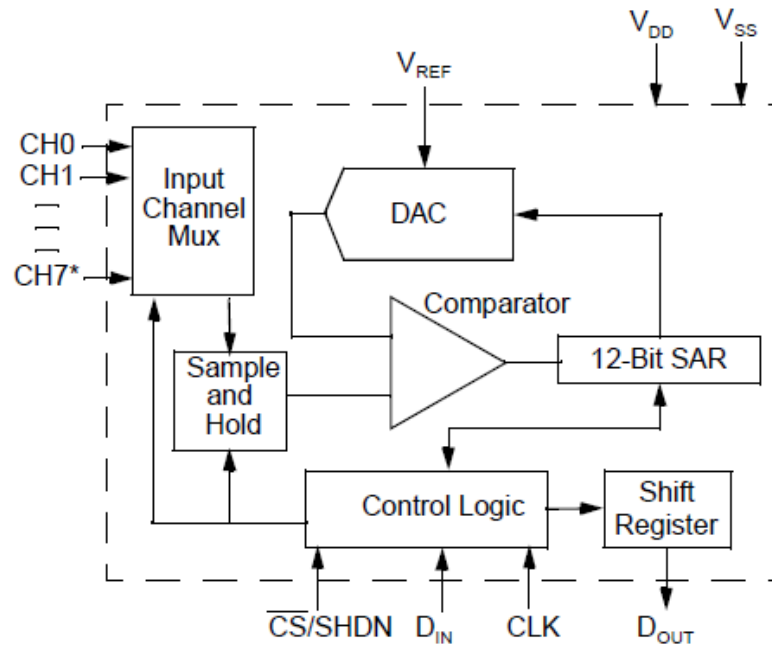
- 12-bit resolution ①
- ± 1 LSB max DNL
- ± 1 LSB max INL (MCP3204/3208-B)
- ± 2 LSB max INL (MCP3204/3208-C)
- 4 (MCP3204) or 8 (MCP3208) input channels ②
- Analog inputs programmable as single-ended or pseudo differential pairs
- On-chip sample and hold
- SPI[®] serial interface (modes 0,0 and 1,1) ③
- Single supply operation: 2.7V - 5.5V ④
- 100ksps max. sampling rate at $V_{DD} = 5V$
- 50ksps max. sampling rate at $V_{DD} = 2.7V$
- Low power CMOS technology
 - 500 nA typical standby current, 2 μ A max.
 - 400 μ A max. active current at 5V
- Industrial temp range: -40°C to +85°C
- Available in PDIP, SOIC and TSSOP packages

ADC (Analog to Digital Converter)

- MCP3204 Pin Layout/Block Diagram/Function Table– Datasheet 참고



Pin Layout



*Note: Channels 5-7 available on MCP3208 Only

Block Diagram

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

Function Table

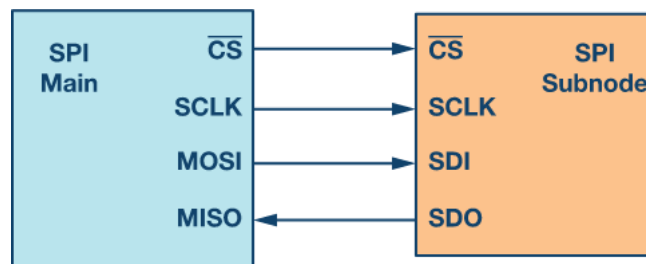
SPI (Serial Peripheral Interface)

- SPI(Serial Peripheral Interface)

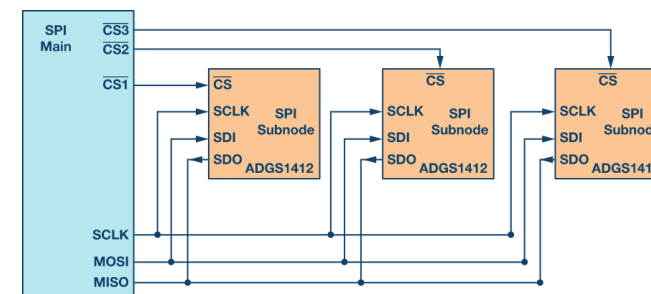
- 주변장치 간의 시리얼 통신을 위한 규약으로, Motorola에서 개발한 통신 방식
- Master – Slave 구조로 동작
 - ✓ 1 Master – 1 Slave (일 대 일)
 - ✓ 1 Master – N Slave (일 대 다)
- 전이중 통신(Full Duplex)
 - ✓ 송신과 수신을 동시에 진행
- 전송되는 비트에 대한 프로토콜 유연성
 - ✓ 최대 16-bit까지 길이 조정 가능함
- 매우 단순한 하드웨어 인터페이스 처리
- 4개의 핀 사용(CS, SCLK, MOSI, MISO)
- I2C보다 낮은 소비 전력

Signal Name	Alternative Names	Description
CS	nCs, SS, nSS, STE, CE	마스터가 특정 슬레이브를 선택
SCLK	SCK, CLK	클럭 신호 생성
MOSI	SDI, DI, SI	마스터의 출력 포트(슬레이브로 송신)
MISO	SDO, DO, SO	슬레이브의 출력 포트(마스터에서 수신)

SPI 통신을 위한 4개의 Pin



1 Master – 1 Slave



1 Master – 3 Slave

MCP3204 Serial Communications

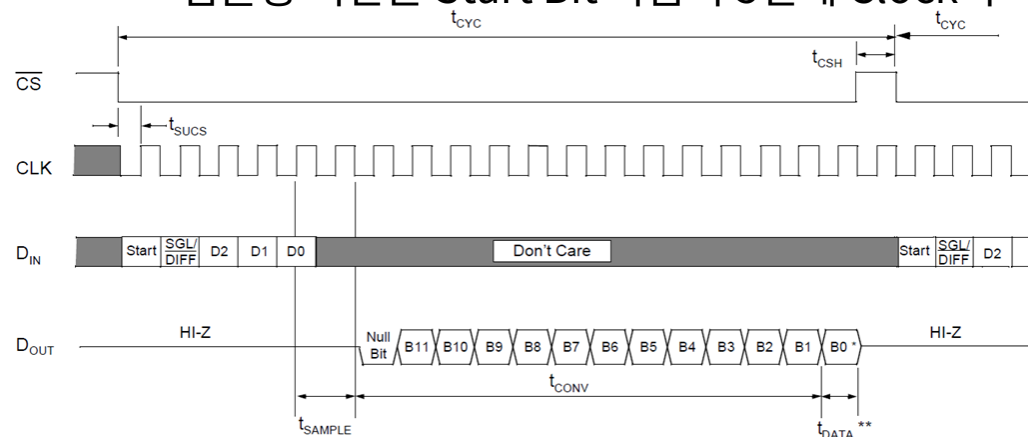
• MCP3204 시리얼 통신

- MCP3204와의 통신은 SPI 호환 시리얼 인터페이스를 사용해 수행됨
- 두 장치 간의 통신은 CS line을 Low로 설정함으로써 시작됨
- CS가 Low, D_{IN} 이 High일 때 수신된 1번째 Clock은 Start Bit를 구성함
- SGL/DIFF Bit는 Start Bit 다음에 위치하며, 변환 모드를 Single Ended or Differential Input으로 결정
- Start Bit 다음의 3bits(D0, D1, D2)는 Input 채널 설정을 위해 사용됨
- Slave는 Start Bit 수신 이후 Clock의 4번째 상승 Edge에서 아날로그 Input을 샘플링하기 시작함
- 샘플링 기간은 Start Bit 다음의 5번째 Clock의 하강 Edge에서 끝남

CONTROL BIT SELECTIONS				INPUT CONFIGURATION	CHANNEL SELECTION
SINGLE/DIFF	D2*	D1	D0		
1	X	0	0	single ended	CH0
1	X	0	1	single ended	CH1
1	X	1	0	single ended	CH2
1	X	1	1	single ended	CH3
0	X	0	0	differential	CH0 = IN+ CH1 = IN-
0	X	0	1	differential	CH0 = IN- CH1 = IN+
0	X	1	0	differential	CH2 = IN+ CH3 = IN-
0	X	1	1	differential	CH2 = IN- CH3 = IN+

*D2 is don't care for MCP3204

MCP3204 비트 구성

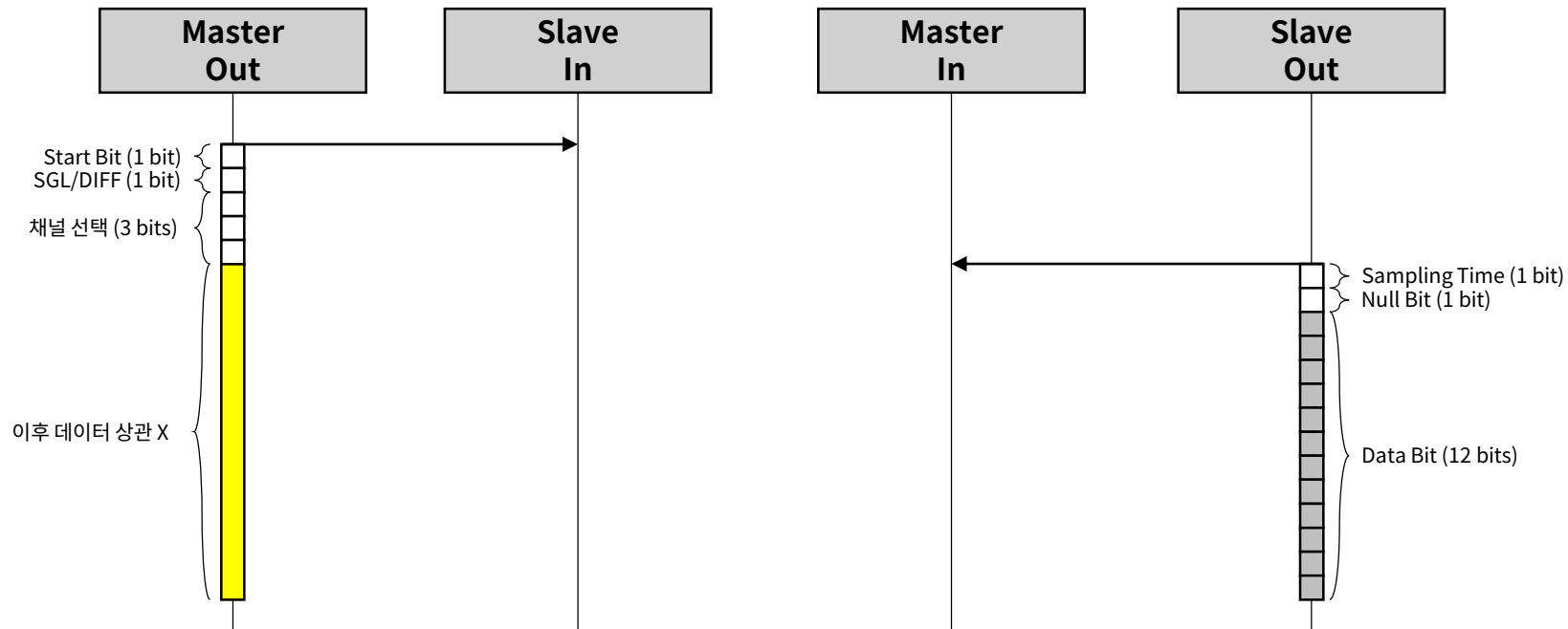


* After completing the data transfer, if further clocks are applied with \overline{CS} low, the A/D Converter will output LSB first data, then followed with zeros indefinitely. See Figure 5-2 below.

** t_{DATA} : during this time, the bias current and the comparator power down while the reference input becomes a high impedance node, leaving the CLK running to clock out the LSB-first data or zeros.

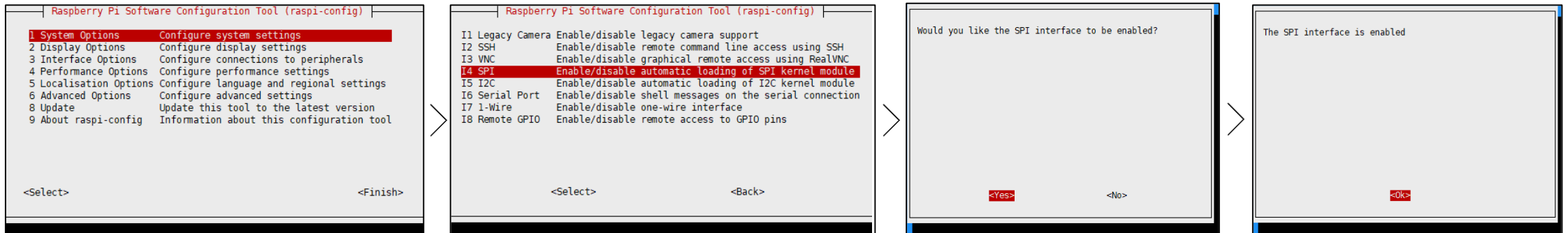
MCP3204 Serial Communications

- MCP3204 시리얼 통신
 - 한 번에 주고받는 데이터 양: 19bits
 - 한 번에 Raspberry Pi에 수신되는 데이터 크기: 19bits
 - Start Bit에서 Null Bit까지의 7bits를 제외한 12bits 사용



Raspberry Pi SPI 인터페이스 활성화

- SPI 인터페이스 활성화
 - XSHELL을 통해 Raspberry Pi에 원격 접속 후, 아래 명령어 입력
 - ✓ **sudo raspi-config**
 - **[3 Interface Options]** 선택
 - **[I4 SPI Enable/disable automatic loading of SPI kernel module]** 선택
 - **[Yes]** 선택
 - **[Finish]** 선택

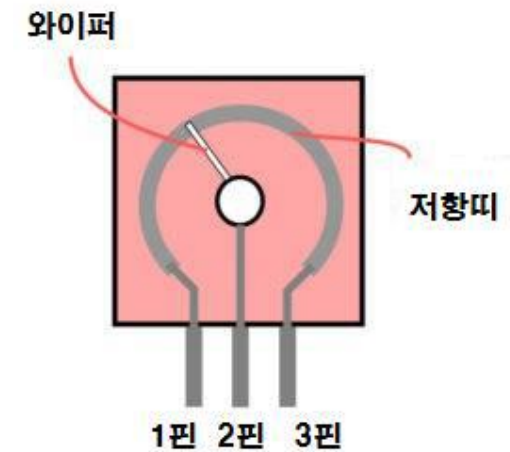


가변저항

- 가변저항(Potentiometer)
 - 가변저항은 저항 값을 임의로 바꿀 수 있으며, 저항 값에 따라 전류의 크기가 바뀜
 - 가변저항 내 와이퍼를 탄소로 구성된 저항띠를 따라 회전시킴으로써 저항값이 조절되도록 하는 구조

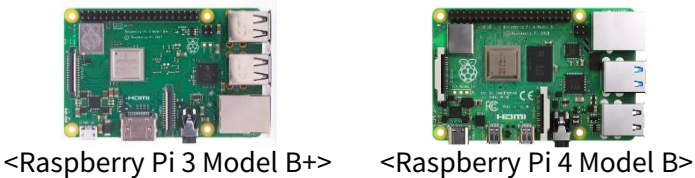





가변저항



ADC 및 가변저항 연결

- 1. 구성품 준비

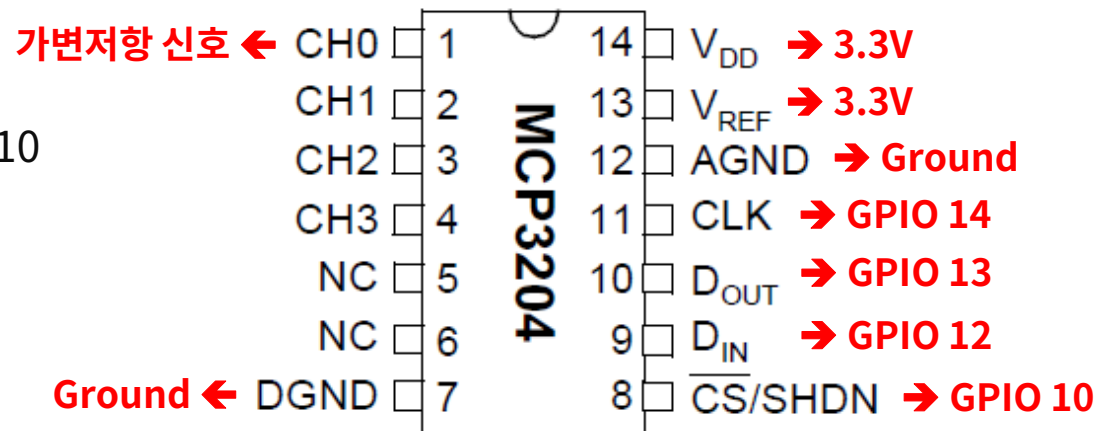
번호	구성요소	사진
1	Raspberry Pi 본체	 <p><Raspberry Pi 3 Model B+> <Raspberry Pi 4 Model B></p>
2	점프 와이어(M/F 6개, M/M 7개)	
3	ADC(MCP 3204)	
4	가변저항	

ADC 및 가변저항 연결

Raspberry Pi: Master
ADC: Slave

- ADC 연결 (MCP3204 – 라즈베리 파이)

- CH0 – MCP3204 Pin 2
- DGND – GND
- VDD – 3.3V
- VREF – 3.3V
- AGND – GND
- CLK – GPIO 14
- DOUT – GPIO 13
- DIN – GPIO 12
- CS/SHDN – GPIO 10



- 가변저항 연결 (가변저항 – 라즈베리 파이)

- Pin 1 – VCC or GND
- Pin 2 – MCP3204 CH0
- Pin 3 – VCC or GND

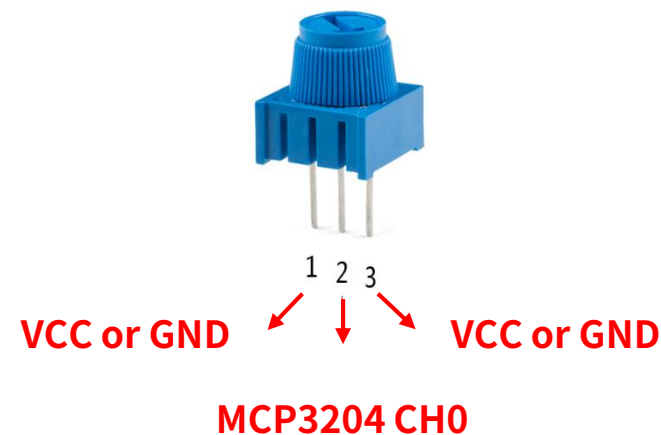
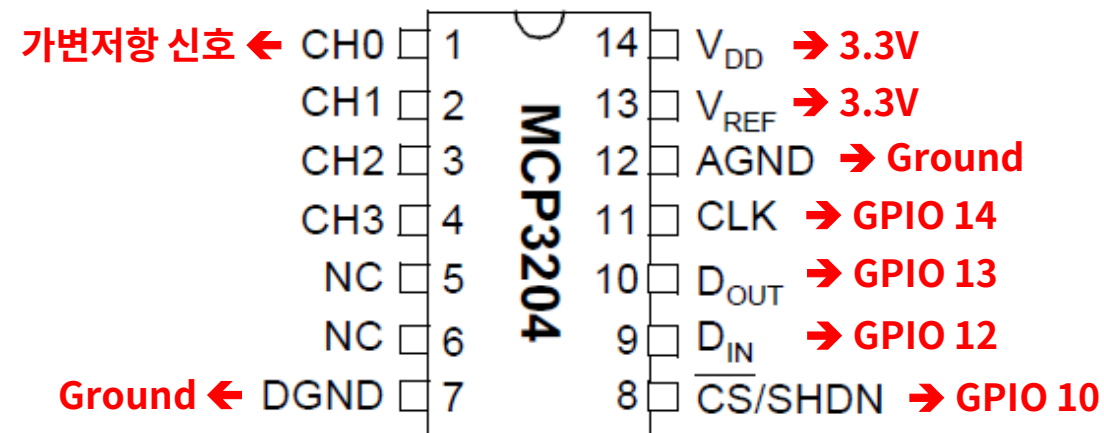


ADC 및 가변저항 연결

Raspberry Pi: Master
ADC: Slave

- ADC 및 가변저항 연결 (MCP3204 및 가변저항 – 라즈베리 파이)

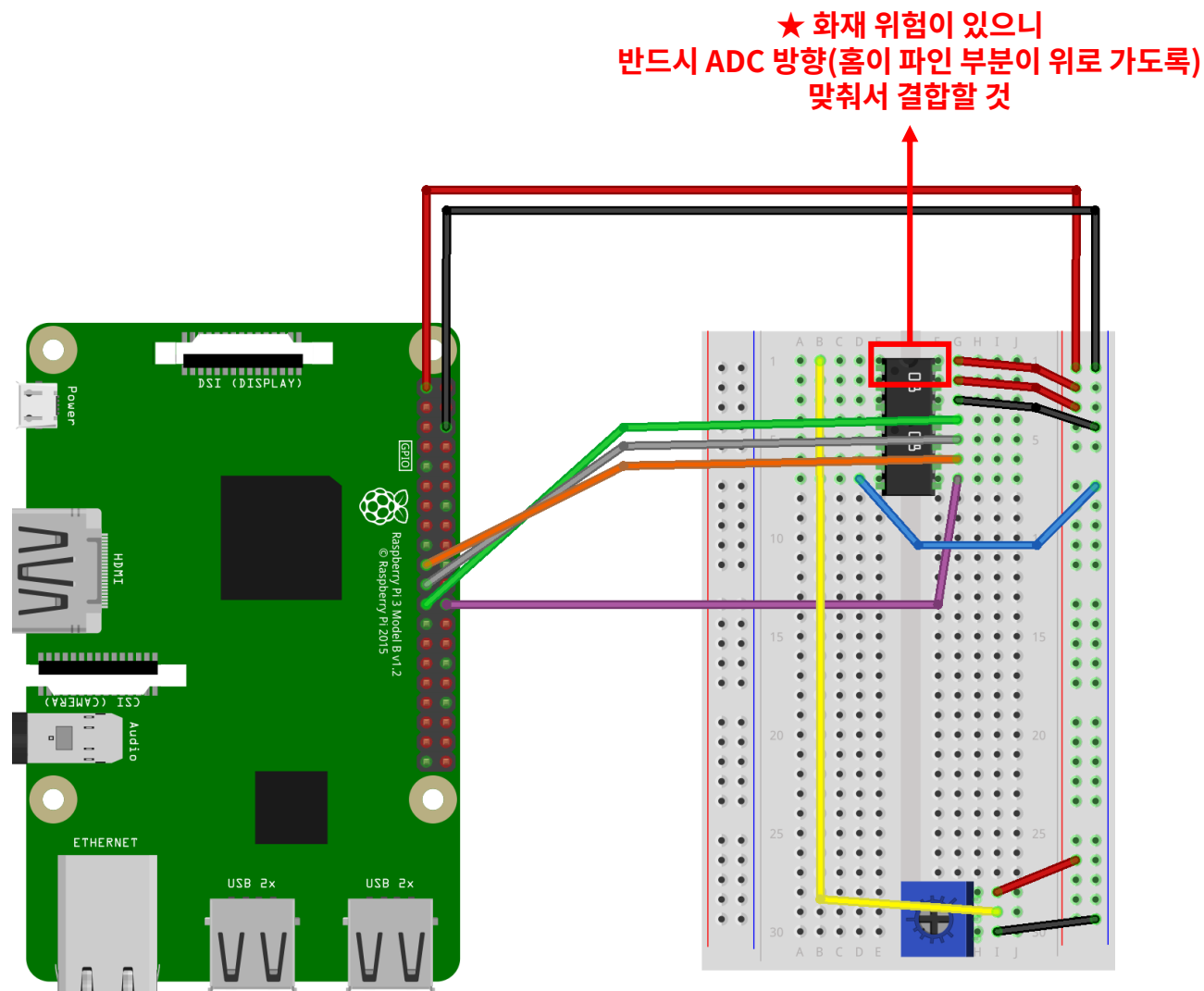
GPIO#	NAME		NAME	GPIO#
	3.3 VDC Power	1	5.0 VDC Power	2
8	GPIO 8 SDA1 (I2C)	3	5.0 VDC Power	4
9	GPIO 9 SCL1 (I2C)	5	Ground	6
7	GPIO 7 GPCLK0	7	GPIO 15 TxD (UART)	15
	Ground	9	GPIO 16 RxD (UART)	16
0	GPIO 0	11	GPIO 1 PCM_CLK/PWM0	1
2	GPIO 2	13	Ground	14
3	GPIO 3	15	GPIO 4	4
	3.3 VDC Power	17	GPIO 5	5
12	GPIO 12 MOSI (SPI)	19	Ground	20
13	GPIO 13 MISO (SPI)	21	GPIO 6	6
14	GPIO 14 SCLK (SPI)	23	GPIO 10 CE0 (SPI)	10
	Ground	25	GPIO 11 CE1 (SPI)	11
30	SDA0 (I2C ID EEPROM)	27	SCL0 (I2C ID EEPROM)	31
21	GPIO 21 GPCLK1	29	Ground	30
22	GPIO 22 GPCLK2	31	GPIO 26 PWM0	26
23	GPIO 23 PWM1	33	Ground	34
24	GPIO 24 PCM_FS/PWM1	35	GPIO 27	27
25	GPIO 25	37	GPIO 28 PCM_DIN	28
	Ground	39	GPIO 29 PCM_DOUT	29
				40



ADC 및 가변저항 연결

• 2. 구성품 연결

GPIO#	NAME		NAME	GPIO#
	3.3 VDC Power	1	2	5.0 VDC Power
8	GPIO 8 SDA1 (I2C)	3	4	5.0 VDC Power
9	GPIO 9 SCL1 (I2C)	5	6	Ground
7	GPIO 7 GPCLK0	7	8	GPIO 15 TxD (UART)
	Ground	9	10	GPIO 16 RxD (UART)
0	GPIO 0	11	12	GPIO 1 PCM_CLK/PWM0
2	GPIO 2	13	14	Ground
3	GPIO 3	15	16	GPIO 4
	3.3 VDC Power	17	18	GPIO 5
12	GPIO 12 MOSI (SPI)	19	20	Ground
13	GPIO 13 MISO (SPI)	21	22	GPIO 6
14	GPIO 14 SCLK (SPI)	23	24	GPIO 10 CE0 (SPI)
	Ground	25	26	GPIO 11 CE1 (SPI)
30	SDA0 (I2C ID EEPROM)	27	28	SCL0 (I2C ID EEPROM)
21	GPIO 21 GPCLK1	29	30	Ground
22	GPIO 22 GPCLK2	31	32	GPIO 26 PWM0
23	GPIO 23 PWM1	33	34	Ground
24	GPIO 24 PCM_FS/PWM1	35	36	GPIO 27
25	GPIO 25	37	38	GPIO 28 PCM_DIN
	Ground	39	40	GPIO 29 PCM_DOUT



ADC 및 가변저항 제어

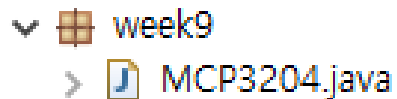
- 1. MCP3204.java 소스 코드

```
package week9;

import java.io.IOException;
import com.pi4j.io.spi.*;

public class MCP3204 {
    public static SpiDevice spi = null;
    public MCP3204() {
        try {
            //SPI 객체 선언
            spi = SpiFactory.getInstance(SpiChannel.CS0, SpiDevice.DEFAULT_SPI_SPEED, SpiDevice.DEFAULT_SPI_MODE);
        } catch (Exception e) {
            System.out.println("Fail to create a SPI instance");
        }
    }

    public static String byteToBinaryString(byte n) {
        // Byte의 binary 값을 String으로 반환
        StringBuilder sb = new StringBuilder("00000000");
        for (int bit = 0; bit < 8; bit++) {
            if (((n >> bit) & 1) > 0) {
                sb.setCharAt(7 - bit, '1');
            }
        }
        return sb.toString();
    }
}
```

A small rectangular window with a red border showing a file explorer. It displays a directory named 'week9' with a dropdown arrow to its left. Inside the directory, there is a file named 'MCP3204.java' with a document icon to its left and a right-pointing arrow to its right.

ADC 및 가변저항 제어

- 1. MCP3204.java 소스 코드(이어서 작성)

- readMCP3204(): ADC 칩의 특정 채널에서 아날로그 신호를 읽어 디지털 값으로 변환하는 메서드

- ✓ MCP3204에 전송할 데이터 준비

- 데이터를 담을 배열을 생성함
 - 첫 번째 바이트에 Start Bit, SGL/DIFF 비트, D2 비트를 설정함
 - adcChannel이 2 이상이면 D1 비트를 설정함
 - adcChannel이 홀수이면 D0 비트를 설정함

- ✓ 데이터 전송 및 수신

- SPI 통신을 통해 데이터를 송수신함
 - Full-duplex이므로 데이터를 전송하면서 동시에 데이터를 수신함

- ✓ 수신 데이터 처리

- 수신된 세 바이트 데이터를 이진 문자열로 변환함
 - 변환된 문자열에서 앞 7비트를 제외한 12비트를 읽어 정수 값으로 변환함
 - 해당 값이 MCP3204에서 읽어온 아날로그 값이 됨

- ✓ 결과 반환

- 읽어온 아날로그 값을 반환함

```
public int readMCP3204(int adcChannel) throws IOException {
```

readMCP3204()

```
}
```

ADC 및 가변저항 제어

- 1. MCP3204.java 소스 코드(이어서 작성)

```
public static void main(String[] args) {  
    MCP3204 obj = new MCP3204();  
    while(true) {  
        try {  
            int value = obj.readMCP3204(0); // CH0  
            System.out.println(value); // 가변저항 값 출력  
            Thread.sleep(1000);  
        } catch (Exception e) {  
            System.out.println(e);  
        }  
    }  
}
```

ADC 및 가변저항 제어

- 2. JAR 파일 생성 후 XFTP를 통해 Raspberry Pi로 전송
- 3. Raspberry Pi에서 JAR 파일 실행
 - `sudo java -jar mcp3204.jar`
- 4. 결과
 - 가변저항을 조절(회전)함에 따라 저항 값의 변화를 확인할 수 있음

```
pi@raspberrypi:~/ES_proj $ sudo java -jar mcp3204.jar
83025
83057
80300
81411
81547
83751
25306
396
112
80
```



감사합니다

Thank You

