라즈베리 파이와 MCP3008의 연결 및 테스트

MCP3008을 라즈베리 파이에 연결하기 위해서는 프로토타입 도구인 파이 코블러와 브레드보드를 사용해야 한다. 그림 6-10은 연결 회로도를 나타낸 것이고, 그림 6-11은 이를 실제로 연결한 모습을 나타낸 것이다.

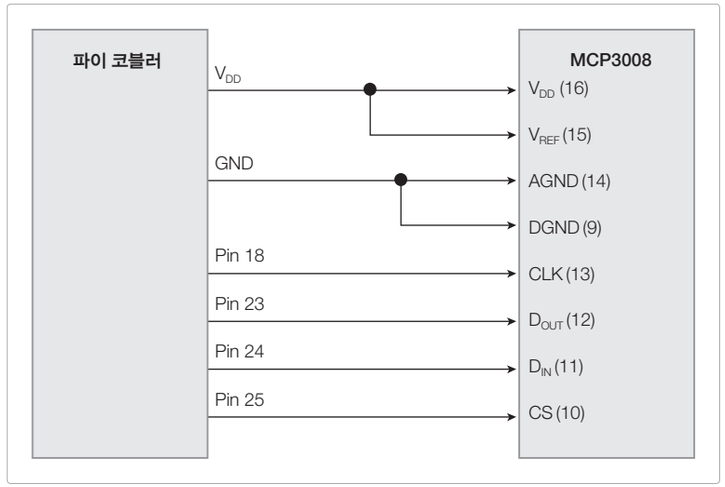

그림 6-10 *라즈베리 파이와 MCP3008의 연결 회로도*

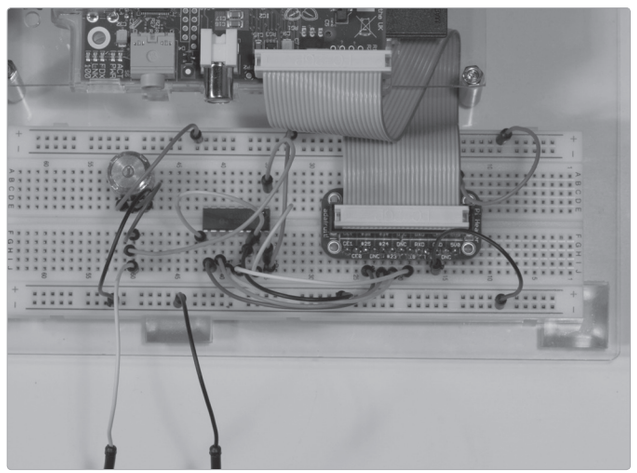

그림 6-11 *라즈베리 파이와 MCP3008의 실제 연결*

브레드보드의 왼쪽에 임시로 구성한 테스트 회로는 3.3V와 접지가 연결된 가변 저항을 포함하고 있다. ADC 채널은 가변 저항의 남은 한 핀에 이어져 테스트용 가변 전압을 받아들인다. 테스트용 소프트웨어인 Test_ADC.py는 ADC값의 연속적인 스트림을 만들어낸다. 이 책의 웹사이트(www.mhprofessional.com/raspi)에서 내려받을 수 있다. 아래의 코드는 위에서 설명한 ADC 및 SPI 프로토콜을 설정하기 위한 것이며, MCP3008 회로의 설명(Learn.Adafruit.com)에 수록된 예제 코드를 기반으로 짜여졌다.

Test_ADC.py

```
import time
import os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
DEBUG = 1
#MCP3008 ADC값을 읽기 위한 함수 작성
def adc(chan, clock, mosi, miso, cs):
    if((chan < 0) or (chan > 7)):
        return -1
    GPIO.output(cs, True)
    GPIO.output(clock, False)
    GPIO.output(cs, False)
    cmd = chan
    cmd |= 0x18
    cmd <<= 3
    for i in range(5):
        if(cmd & 0x80):
            GPIO.output(mosi, True)
        else:
            GPIO.output(mosi, False)
        cmd <<= 1
        GPIO.output(clock, True)
        GPIO.output(clock, False)
    result = 0
    for i in range(12):
        GPIO.output(clock, True)
        GPIO.output(clock, False)
```

```
        result <<= 1
        if(GPIO.input(miso)):
            result |= 0x1
    GPIO.output(cs, True)
    result >>= 1
    return result
#테스트 회로를 작동하기 위한 핀 정의
SPICLK = 18
SPIMISO = 23
SPIMOSI = 24
SPICS = 25
GPIO.setup(SPIMOSI, GPIO.OUT)
GPIO.setup(SPIMISO, GPIO.IN)
GPIO.setup(SPICLK, GPIO.OUT)
GPIO.setup(SPICS, GPIO.OUT)
channel = 0
while True:
    #ADC값 읽기
    adc_value = adc(channel, SPICLK, SPIMOSI, SPIMISO, SPICS)
    if DEBUG:
        print "value = ", adc_value
    #1초 대기 후 반복
    time.sleep(1)
```

그림 6-12는 ADC값을 500으로 조정한 아날로그 전압을 출력하는 프로그램의 출력 화면이다. 실제 전압은 보정하지 않은 멀티미터(전압, 전류, 저항값을 측정하는 종합 계측기)로 측정했을 때 1.629V라는 값을 가진다. 이는 ADC를 통한 계산치인 1.612에 꽤 근접한 값이다. 계산치와 측정치의 차이는 주로 이론적인 전원 전압보다 살짝 높은 3.32V 정도의 공급 전압에 의한 것일 수 있다. 이를 실제로 계산에 반영하면 그 결과는 1.621V로 원래의 계산치보다 0.008V 낮아진다. 그 결과는 ADC값 3, 혹은 +0.3% 정도의 오차를 보이며, 이런 종류의 ADC에서는 일반적인 오차이다.

예리한 독자들은 위 프로그램이 내장 SPI 기능이 아닌, 1장에서 언급한 '비트 충돌' 인터페이스를 구현하였음을 알 수 있을 것이다. 이러한 접근법을 사용한 이유는 이 프로젝트에서 사용한 리눅스 버전이 직접적으로 SPI 프로토콜을 구현하지 않았기 때문이다. 4개의 GPIO 핀만을

사용하여 비트 충돌을 구현하였기 때문에 이 핀이 다른 용도로 쓰일 수 없을 뿐, 내장 SPI와 기능적인 차이는 없다.

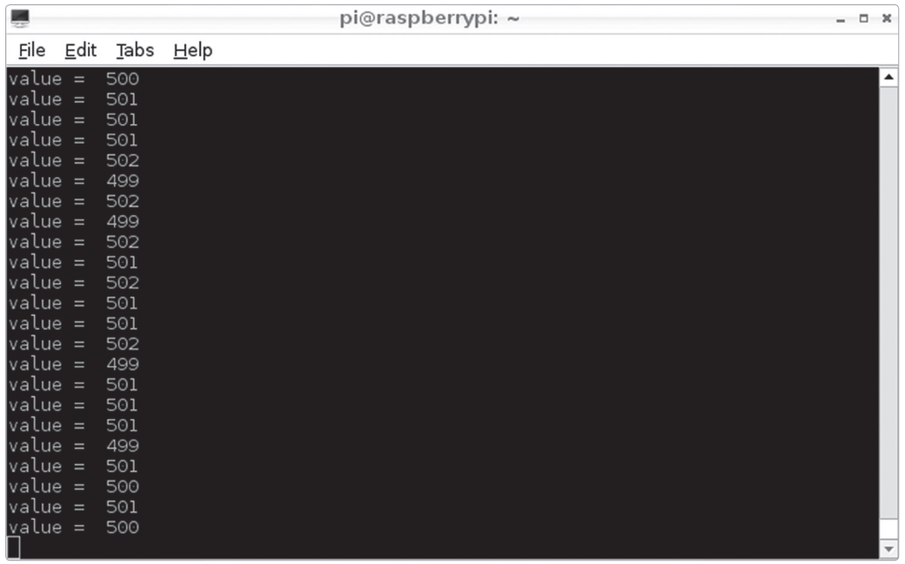

그림 6-12 *Test_ADC.py 프로그램의 출력 화면*

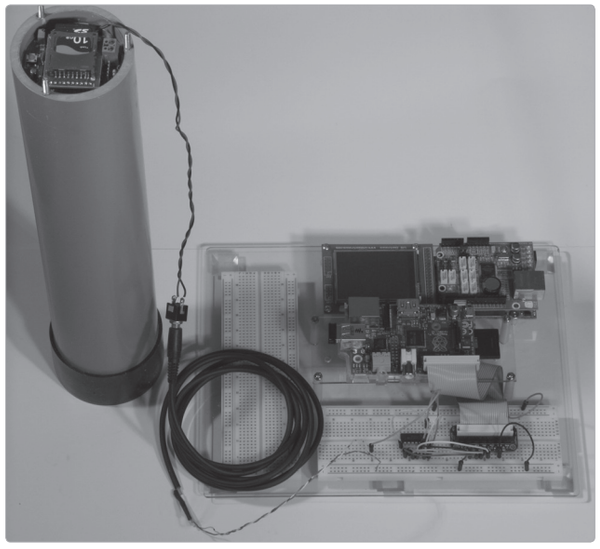

그림 6-13 *파이 코블러 및 라즈베리 파이와 지진계, MCP3008의 연결*

지진계 연결하기

가난뱅이용 지진계를 MCP3008 회로와 연결하기는 매우 쉽다. 우선 앞서 사용한 테스트 회로를 제거한다. 그리고 지진계의 OP 앰프 출력과 접지를 MCP3008의 채널 0과 접지에 연결한다. 그림 6-13은 지진계를 MCP3008 회로에 연결하고, 그 출력을 파이 코블러를 통해 라즈베리 파이에 연결한 것이다.

테스트를 완료했다면 이제 지진계 센서로부터 실제 데이터를 받아야 하기 때문에, 소프트웨어에 약간의 수정이 필요하다. 이 코드는 'Test_ADC.py Segment'라 불리며, 앞에 나왔던 코드에서 아래의 구문 이후를 대체한다.

```
channel = 0.
```

새로운 코드로 대체한 프로그램의 이름은 Test_File_ADC.py이다. 마찬가지로 이 책의 웹사이트(www.mhprofessional.com/raspi)에서 내려받을 수 있다. 이 프로그램은 1,200개의 adc_value를 작업 디렉터리인 pi 안에 있는 파일인 'myData'에 입력한다. 물론 경로를 바꾸지 않았을 때의 이야기다. 각각의 ADC값에는 줄바꾸기 문자를 붙여서 데이터 파일을 쉽게 엑셀로 옮겨서 분석할 수 있도록 했다.

Test_ADC.py Segment(실제 데이터의 수집을 위한 수정)

```
file = open("myData", 'w')
for i in range(1200):
    adc_value = adc(channel, SPICLK, SPIMOSI, SPIMISO, SPICS)
    #str() 함수는 adc_value의 수치를 문자열로 변환한다. "n"은 줄바꾸기 문자이다.
    adcStr = str(adc_value) + "\n"
    file.write(adcStr)
    if DEBUG:
        print "value = ", adc_value
   #1초에 10회 샘플을 취한다.
    time.sleep(.1)
file.close()
print("All done")
```

파이썬에서는 데이터를 파일에 저장할 때 두 가지 방식, 즉 문자열 혹은 이진 형식을 지원한다.

이 프로그램에서는 일련의 문자열로 데이터를 저장하도록 하여, 데이터를 엑셀 스프레드시트에 입력하거나 추가 분석이 가능하게끔 되어 있다. 단, 데이터를 문자열로 저장하면 이진 형식으로 저장할 때보다 더 큰 파일 공간을 소모하게 된다. 다음의 코드는 어떻게 데이터를 이진 형식으로 저장하는지 보여준다.

```
file.write(adc_value, "wb")
```

여기에서 wb는 이진 쓰기write binary를 줄인 말이다. 마찬가지로 데이터를 읽을 경우에는 이 자리에 'rb'를 써야 한다. 그뿐 아니라, '쓰레기' 값을 읽거나 순서를 섞지 않기 위해 바이트 수를 세려면 프로그램을 재구성해야 한다. 이진 쓰기를 사용하면 다음과 같은 구문은 필요하지 않을 것이다.

```
adcStr = str(adc_value) + "\n"
```

그 이유는 이진 데이터는 프로그래머가 판독해야 의미를 알 수 있는 연속적인 바이트 스트림으로 쓰여지기 때문이다. 그런 측면에서, 문자열을 사용하면 파일의 내용을 디버깅하기가 쉽다.

샘플 사이의 간격이 0.1초로 설정되었기 때문에, 프로그램은 2분 동안 돌아간다. 이 샘플 주파수는 앞에서 언급한 지진계의 이상적인 값인 10Hz에 해당된다는 사실을 기억하자. 또한, 'DEBUG = 0'이 설정되어 있을 때에는 화면에 1,200개의 값이 나타나게 된다.

Test_File_ADC.py를 실행할 때는 지진계를 조금 흔들어서 지진파에 가까운 활동을 만들어내야 한다. 하지만 지나치게 의욕적으로 흔들어서 진자가 PVC 파이프 벽을 건드리게 하지는 말자. 센서가 꽤 민감하기 때문에 과도한 힘을 가하면 안 된다. 파일을 만들고 나면 데이터 분석을 위한 준비가 끝난 것이다.

지진 데이터 분석

myData 파일의 내용을 복사하여 엑셀 스프레드시트에 붙여 넣자. 하나의 열에 정확히 1,200개의 값이 들어 있어야 한다. 각각의 값은 지진계가 디지털 값으로 변환한 샘플을 나타낸다. 각각의 샘플을 0.1초 간격으로 취했기 때문에 기록된 데이터의 전체 길이는 2분 정도에 해당한다. 그