예를 들어, 4방향 버튼의 위쪽 버튼을 누르면 2048이라는 값이 블루투스를 통해 라즈베리 파이로 전달된다. 마찬가지로 '–' 버튼을 누르면 16의 값이 전달된다. 버튼을 조합하면 그에 해당하는 각 값을 더한 값이 전달된다. '1' 버튼과 홈 버튼을 동시에 누르면 130의 값이 전달된다. '1' 버튼의 값인 2와 홈 버튼의 값인 128을 합친 값이 전달되는 것이다.

Wiimote_Test.py라는 이름으로 작성된 간단한 테스트 프로그램을 통해 자동차와 위모트 사이에 블루투스 연결을 맺어 보자. 테스트를 위해 위모트의 4방향 버튼을 통해 서로 다른 4가지 값을 보낼 수 있다. 책의 웹사이트(www.mhprofessional.com/raspi)에서 프로그램을 내려받은 후, 아래 명령을 입력하여 프로그램을 실행하자.

```
sudo python Wiimote_Test.py ↵
```

그림 13–12는 연결을 맺은 후 4개의 방향 버튼을 눌러 나타난 결과이다.

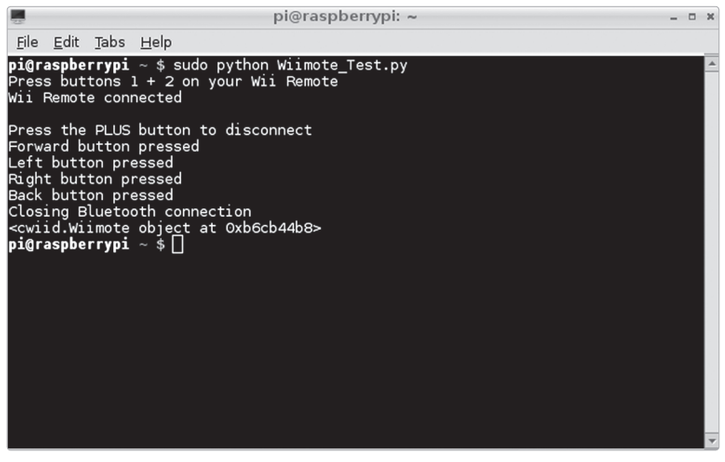

그림 13-12 *Wiimote_Test.py 프로그램의 결과*

Wiimote_Test.py의 실제 코드는 아래와 같다.

Wiimote_Test.py

```
#!/usr/bin/env python
import time
import cwiid
print 'Press buttons 1 + 2 on your Wii Remote'
time.sleep(3)
wm = cwiid.Wiimote()
print 'Wii Remote connected'
print '\nPress the PLUS button to disconnect'
time.sleep(1)
wm.rpt_mode = cwiid.RPT_BTN
while (wm.state['buttons'] < 4096):
    if wm.state['buttons'] == 2048:
        print('Forward button pressed')
        time.sleep(1)
    if wm.state['buttons'] == 1024:
        print('Back button pressed')
        time.sleep(1)
    if wm.state['buttons'] == 512:
        print('Right button pressed')
        time.sleep(1)
    if wm.state['buttons'] == 256:
        print('Left button pressed')
        time.sleep(1)
print 'Closing Bluetooth connection'
time.sleep(1)
exit(wm)
```

'wm = cwiid.Wiimote()' 구문은 위모트의 논리적 객체를 선언하는 역할을 한다. 그리고 'wm.rpt_mode = cwiid.RPT_BTM' 구문은 이 객체가 상태값을 내보낼 수 있도록 설정한다. 그 뒤에는 'wm.state['buttons']' 구문을 통해 위모트의 현재 상태값을 받아 오면 된다.

'+' 버튼을 누르면 위모트 연결이 끊어지며 프로그램이 종료된다. 'while (wm.state['buttons'] < 4096):' 구문은 상태값이 4096 이상이 될 때까지 반복하는 루프를 형성하는 반복문이기 때문에, '+'에 할당된 값 4096은 반복문을 빠져나가 프로그램을 종료시키게 된다. 이렇게 프로그램 상태를 바꾸는 데 아주 큰 값을 사용하면, 버튼 조합을 통해 만들어질 수 있는 큰 값이 프로그램을 의도치 않게 종료시키는 것을 막을 수 있다. 자신만의 프로그램을 작성할 때에도 이 방법을 참조하도록 하자.