

Raspberry Pi Technical Documentation

scwook*

Rare Isotope Science Project
Institute for Basic Science, Daejeon, South Korea

March 19, 2015

Abstract

본 기술문서는 Raspberry Pi에 대한 기본 설치 및 설정방법을 포함하여 다양한 센서들을 이용한 테스트 및 EPICS Integration 방법에 대하여 설명하였다. 기본적으로 사용된 Model은 Raspberry Pi Model B+ 이며 OS는 Raspbian을 사용하였다.

1 Introduction

Raspberry Pi(RPi)는 교육용 프로젝트의 일환으로 개발된 소형 컴퓨터로 가격이 아주 저렴하고 신용카드 정도의 크기를 가지고 있다. RPi는 하드웨어적으로 ARM기반의 CPU를 장착하고 있으며 5V의 Micro USB를 통해 전원을 공급받는다. 확장 포트로는 USB, Ethernet Port, HDMI를 지원하며, 특히 입출력 신호를 제어하기 위한 GPIO(General Purpose Input Output)포트를 지원하는데 SPI, I2C, UART통신이 가능하다. 결과적으로 다양한 Device 및 Sensor를 RPi를 통해 제어 및 모니터링 가능하다. 본 기술문서에서 다루고 있는 Device 및 Sensor는 다음과 같다.

- PIR Motion Sensor[1]
- PM1001 Dust Sensor[2]
- DHT11 Temperature and Humidity Sensor[3]
- DHT22 Temperature and Humidity Sensor[4]
- DS1820 Temperature Sensor[5]

*@ibs.re.kr

- L298 Dual H-Bridge Motor Driver[6]
- MD5-DH14 Motor Driver[7]

RPi는 ARM 아키텍처를 기반으로 하기 때문에 이를 지원하는 OS는 거의 설치가능하다. 현재 공식 홈페이지에서 제공하는 OS는 5가지가 있으며, 이 중 Debian 계열의 Raspbian이 가장 많이 사용되고 있다.

1.1 Installation

Raspbian을 설치하는 방법은 2가지가 있다.

- New Out Of the Box Software(NOOBS) 설치
- Raspbian Image 설치

Raspberry Pi에 설치되는 OS는 Raspbian외에 몇가지가 더 있는데 NOOBS는 이러한 OS를 Package로 묶은 것으로 하나 또는 그 이상의 OS를 한번에 설치할 수 있다. 만약 하나의 OS만 설치하고자 하는 경우에는 Image파일을 이용하여 설치하면 되는데 초보자에게는 다소 어려울 수 있다. RPi 공식 홈페이지에서는 NOOBS를 이용하는 것을 추천하므로 여기에서도 NOOBS를 이용하여 설치를 진행한다.

Download

Raspbian 설치를 위해 다음 홈페이지에서 NOOBS 파일을 다운 받는다

<http://www.raspberrypi.org/downloads/>

다운로드한 파일의 압축을 해제하고 Micro SD Card에 파일을 전부 복사한다.

First Boot

Raspberry Pi전원을 연결 하면 NOOBS Install Manager가 나오는데 Raspbian을 선택한 후 Install 버튼을 누르면 설치가 진행된다. 설치가 완료되고 재부팅을 하면 Raspberry Pi Software Configuration Tool이 나타나는데 Finish를 누르면 기본적인 Raspbian 설치는 완료된다.

1.2 Configuration

Password

Raspbian의 기본 ID 및 Password는 각각 pi와 raspberry로 설정되어 있으며 passwd 명령을 통해 Password 변경이 가능하다.

```
pi@raspberrypi# passwd
Changing password for pi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Networking

네트워크는 기본적으로 DHCP로 설정되어 있다. 고정 IP로 변경할 경우 `/etc/network/interfaces` 파일을 수정한다 .

```
auto lo

iface lo inet loopback

allow-hotplug eth0
iface eth0 inet static
    address 10.1.4.206
    netmask 255.255.255.0
    broadcast 10.1.4.255
    gateway 10.1.4.254
    dns-nameservers 10.1.2.240
```

무선 네트워크를 사용할 경우 다음과 같이 무선 네트워크 정보를 추가해 준다.

```
auto lo

iface lo inet loopback

allow-hotplug wlan0
iface wlan0 inet static
    address 10.1.4.207
    netmask 255.255.255.0
    network 10.1.4.0
    broadcast 10.1.4.255
    gateway 10.1.4.254
    dns-nameservers 10.1.2.240
    wpa-scan-ssid 1
    wpa-ap-ssid 1
    wpa-key-mgmt WPA-PSK
    wpa-proto RSN WPA
    wpa-pairwise CCMP TKIP
    wpa-group CCMP TKIP
    wpa-ssid "CTRLTEAM"
    wpa-psk "asdf12345"
```

여기서 `wpa-ssid`와 `wpa-psk`는 사용하고자 하는 무선네트워크 `ssid`와 `password`를 넣으면 된다. 만약 DHCP로 무선을 설정 할 경우 다음과 같이 `wpa-ssid`와 `wpa-psk`만 설정해 주면 된다.

```
auto lo

iface lo inet loopback

allow-hotplug wlan0
    iface wlan0 inet dhcp
    wpa-ssid "CTRLTEAM"
    wpa-psk "asdf12345"
```

UART

Raspberry Pi의 UART(Universal asynchronous receiver/transmitter)는 일반적으로 RS-232와 같은 시리얼 통신을 5V Level의 TTL신호로 전송한다. 따라서 Raspberry Pi나 Arduino와 같이 UART를 지원하는 보드에서는 바로 연결하여 사용이 가능하다. 여기서

주의할 사항은 PC와 같이 Serial 통신의 전압 Level이 다른 센서를 사용할 경우 MAX232와 같은 Level Convert를 사용하여 사용하려는 센서의 전압 Level에 맞게 변경해야 한다. 기본적으로 Raspberry Pi의 UART는 Consol접속용으로 설정되어 있다. 따라서 일반적인 UART 통신을 하기 위해서는 다음과 같이 /boot/cmdline.txt 파일과 /etc/inittab 파일을 수정해야 한다.

/boot/cmdline.txt 파일을 열어 "console=ttyAMA0,115200" 부분을 삭제 한다.

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=de$
```

/etc/inittab 파일을 열어 마지막 라인에 있는 "T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100" 앞에 '#'을 넣어 주석 처리한다

```
# /etc/inittab: init(8) configuration.
# $Id: inittab,v 1.91 2002/01/25 13:35:21 miquels Exp $

# The default runlevel.
id:2:initdefault:

# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6
# Normally not reached, but fallthrough in case of emergency.
z6:6:respawn:/sbin/sulogin

# What to do when CTRL-ALT-DEL is pressed.
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now

# Action on special keypress (ALT-UpArrow).
#kb::kbrequest:/bin/echo "Keyboard Request--edit /etc/inittab to let this work."

# What to do when the power fails/returns.
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop

# /sbin/getty invocations for the runlevels.
#
# The "id" field MUST be the same as the last
# characters of the device (after "tty").
#
```

```
#
# Format:
#   <id>:<runlevels>:<action>:<process>
#
# Note that on most Debian systems tty7 is used by the X Window System,
# so if you want to add more getty's go ahead but skip tty7 if you run X.
#
1:2345:respawn:/sbin/getty --noclear 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
4:23:respawn:/sbin/getty 38400 tty4
5:23:respawn:/sbin/getty 38400 tty5
6:23:respawn:/sbin/getty 38400 tty6

# Example how to put a getty on a serial line (for a terminal)
#
#T0:23:respawn:/sbin/getty -L ttyS0 9600 vt100
#T1:23:respawn:/sbin/getty -L ttyS1 9600 vt100

# Example how to put a getty on a modem line.
#
#T3:23:respawn:/sbin/mgetty -x0 -s 57600 ttyS3

#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

설정을 마쳤으면 재부팅 한다.

```
pi@raspberrypi~$ sudo reboot
```

재부팅이 완료되면 UART를 사용할 수 있다. 참고로 Raspberry Pi의 UART device는 /etc/ttyAMA0에 할당되어 있으며 8번 Pin을 Transmit, 10번 Pin을 Receive로 사용한다.

1.3 Library

wiringPi

WiringPi는 Arduino에서 사용되는 Wiring Library를 RPi에 맞게 변경한 것으로 GPIO access를 포함하여 PiFace, GertBoard와 같은 확장 보드 및 HD44780U와 같은 Device Chip을 지원한다. 보다 자세한 사항은 홈페이지[8]를 참고하기 바란다.

기본적인 설치 는 git 서버로 부터 파일을 복사한 후 빌드하면 된다.

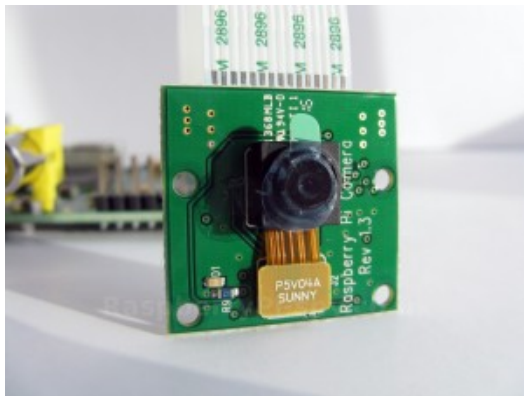
```
pi@raspberrypi# git clone git://git.drogon.net/wiringPi
Cloning into 'wiringPi'...
remote: Counting objects: 657, done.
remote: Compressing objects: 100% (599/599), done.
remote: Total 657 (delta 476), reused 95 (delta 58)
Receiving objects: 100% (657/657), 247.61 KiB | 94 KiB/s, done.
Resolving deltas: 100% (476/476), done.
pi@raspberrypi# cd wiringPi
pi@raspberrypi:/wiringPi# ./build
```

Pin Layout은 'readall' 명령으로 확인 가능하다. Wiring Pi는 자체적인 Pin Map을 사용하는데 wPi로 표시된 부분은 Wiring Pi에서 사용하는 GPIO 핀 번호이다.

```
pi@raspberrypi# gpio readall
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```


2.1 Pi Camera

Raspberry Pi에서 제공하고 있는 카메라는 2종류가 있다. 하나는 일반적으로 사용하는 카메라로 기판 색이 초록색으로 되어 있으며 가시광선 영역의 파장을 받는다. 다른 하나는 NoIR(No Infrared) 카메라로 기판색이 검은색이며 가시광선을 포함하여 적외선 영역의 파장까지 인식한다. 즉, NoIR 카메라의 경우 적외선 LED와 함께 사용하면 어두운 장소에서도 촬영이 가능하다. 반면 낮에는 실제 색감 및 밝기가 일반 카메라와 다르게 보이는 단점이 있다.



(a) Normal Camera



(b) NoIR Camera

Figure 2 Raspberry Pi Camera Module

Installation

RPi 카메라 모듈은 Figure 3과 같이 Camera 전용 Port를 사용하며 RPi Configuration을 통해 Port를 활성화 시켜 준다.

```
pi@raspberrypi# sudo raspi-config
```

설정을 마쳤으면 재부팅 한다.

```
pi@raspberrypi# sudo reboot
```

Test

기본적인 카메라 작동은 Shell Command를 사용하며, 보다 다양한 기능의 작동은 Raspberry Pi 홈페이지 [9]를 참고하기 바란다. 사진 캡처는 raspistill을 사용한다.

```
pi@raspberrypi# raspistill -o cam.jpg
```

상하 좌우 반전을 하고 싶으면 vf, hf 옵션을 설정한다.

```
pi@raspberrypi# raspivid -vf -hf -o cam2.jpg
```

동영상 촬영은 raspivid를 사용한다.

```
pi@raspberrypi# raspivid -o vid.h264
```

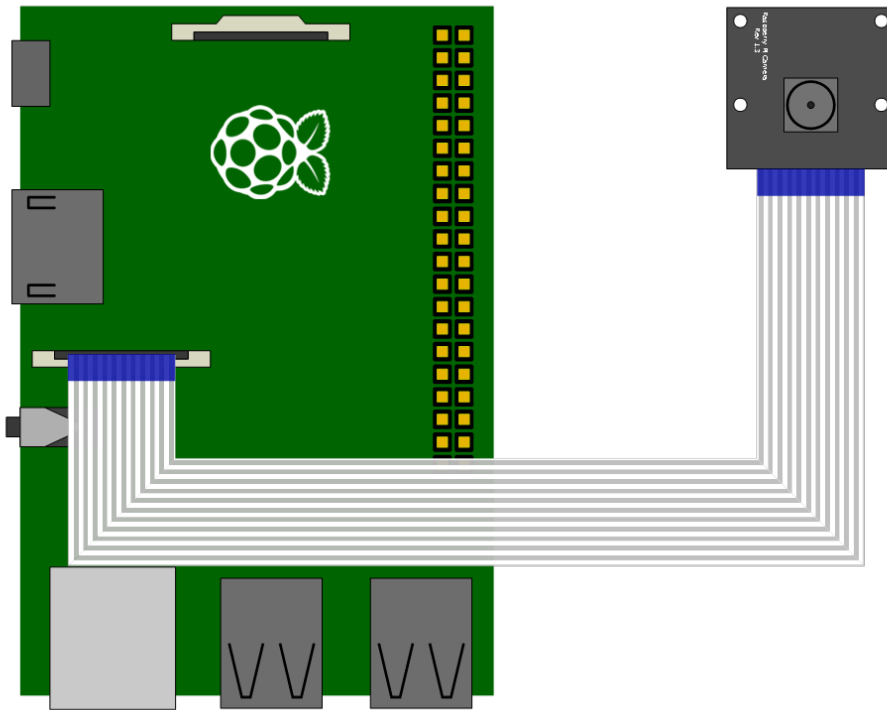


Figure 3 Camera Installation

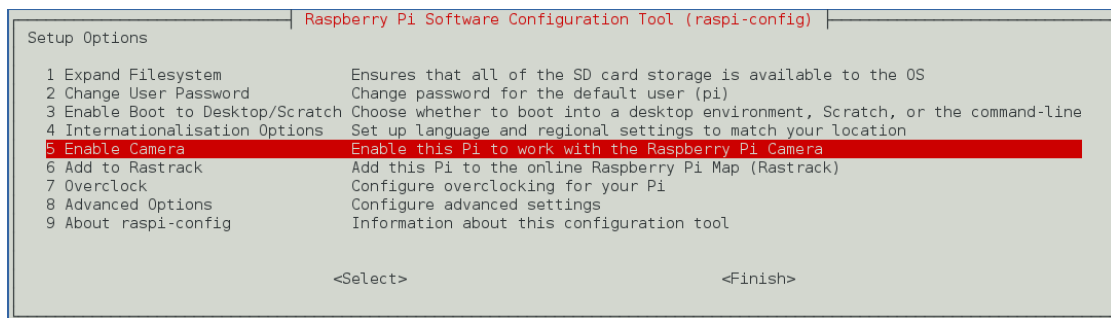


Figure 4 Camera Installation

t 옵션을 사용하면 시간 설정이 가능하다.(기본은 5초) 다음은 10초동안 촬영한다.

```
pi@raspberrypi# sudo raspi-config
```

카메라가 작동할 때 LED가 켜지지 않게 하려면 /boot/config.txt 파일에 disable_camera_led=1 을 추가한 후 재부팅 한다.

```
...
...
```

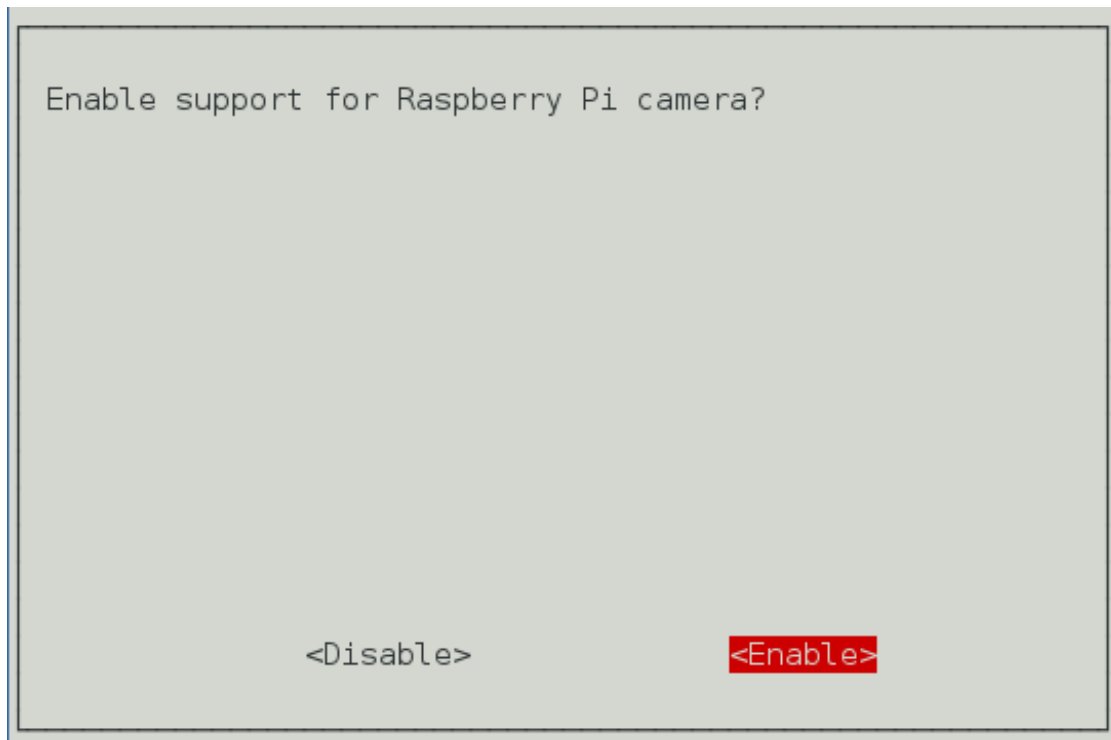



Figure 5 Camera Installation

```
...  
  
# NOOBS Auto-generated Settings:  
hdmi_force_hotplug=1  
config_hdmi_boost=4  
overscan_left=24  
overscan_right=24  
overscan_top=16  
overscan_bottom=16  
disable_overscan=0  
start_x=1  
gpu_mem=128  
disable_camera_led=1
```

Web Streaming

mjpg streamer Library를 이용하면 Web Streaming을 통해 영상을 볼 수 있다. Library를 설치하기 앞서 Web Streaming에 필요한 파일을 설치한다.

```
pi@raspberrypi# sudo apt-get install git cmake libjpeg8-dev imagemagick -y
```

파일 설치가 완료되면 videodev.h 헤더파일을 videodev2.h파일로 링크한다.

```
pi@raspberrypi# sudo ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h
```

기본 준비가 끝났으면 git에서 mjpg streamer를 복사한다.

```
pi@raspberrypi# git clone https://github.com/jacksonliam/mjpg-streamer
```

mjpg-streamer/mjpg-streamer-experimental 폴더로 이동한 후 make를 실행한다.

```
pi@raspberrypi# cd mjpg-streamer/mjpg-streamer-experimental
pi@raspberrypi:/mjpg-streamer/mjpg-streamer-experimental# make
```

make가 완료되면 다음과 같은 실행 스크립트를 만든다.

```
export STREAMER_PATH=$HOME/mjpg-streamer/mjpg-streamer-experimental
export LD_LIBRARY_PATH=$STREAMER_PATH
$STREAMER_PATH/mjpg_streamer -i "input_raspicam.so -x 640 -y 480 -fps 30" -o "output_http.so -w
$STREAMER_PATH/www"
```

마지막으로 스크립트를 실행 한다.

```
pi@raspberrypi# sh mjpg.sh
MJPEG Streamer Version: svn rev:
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 118): argv[0]=raspicam
input plugin
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 118): argv[1]=-x
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 118): argv[2]=640
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 118): argv[3]=-y
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 118): argv[4]=480
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 118): argv[5]=-fps
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 118): argv[6]=30
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 175): case 2,3
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 181): case 4,5
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
input_init(), 187): case 6, 7
i: fps.....: 30
i: resolution.....: 640 x 480
i: camera parameters.....:

Sharpness 0, Contrast 0, Brightness 50
Saturation 0, ISO 400, Video Stabilisation No, Exposure compensation 0
Exposure Mode 'auto', AWB Mode 'auto', Image Effect 'none'
Metering Mode 'average', Colour Effect Enabled No with U = 128, V = 128
Rotation 0, hflip No, vflip No
o: www-folder-path...: /home/pi/mjpg-streamer/mjpg-streamer-experimental/www/
o: HTTP TCP port.....: 8080
o: username:password.: disabled
o: commands.....: enabled
i: Starting Camera
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
worker_thread(), 553): Host init, starting mmal
stuff
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
worker_thread(), 681): Camera enabled, creating
encoder
Encoder Buffer Size 81920
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,
worker_thread(), 764): Encoder enabled, creating
pool and connecting ports
```

```
DBG(/home/pi/mjpg-streamer/mjpg-streamer-experimental/plugins/input_raspicam/input_raspicam.c,  
worker_thread(), 880): Starting video o
```

스크립트가 실행되면 다음 주소를 통해 영상을 확인할 수 있다.

http://[IP Address]:8080

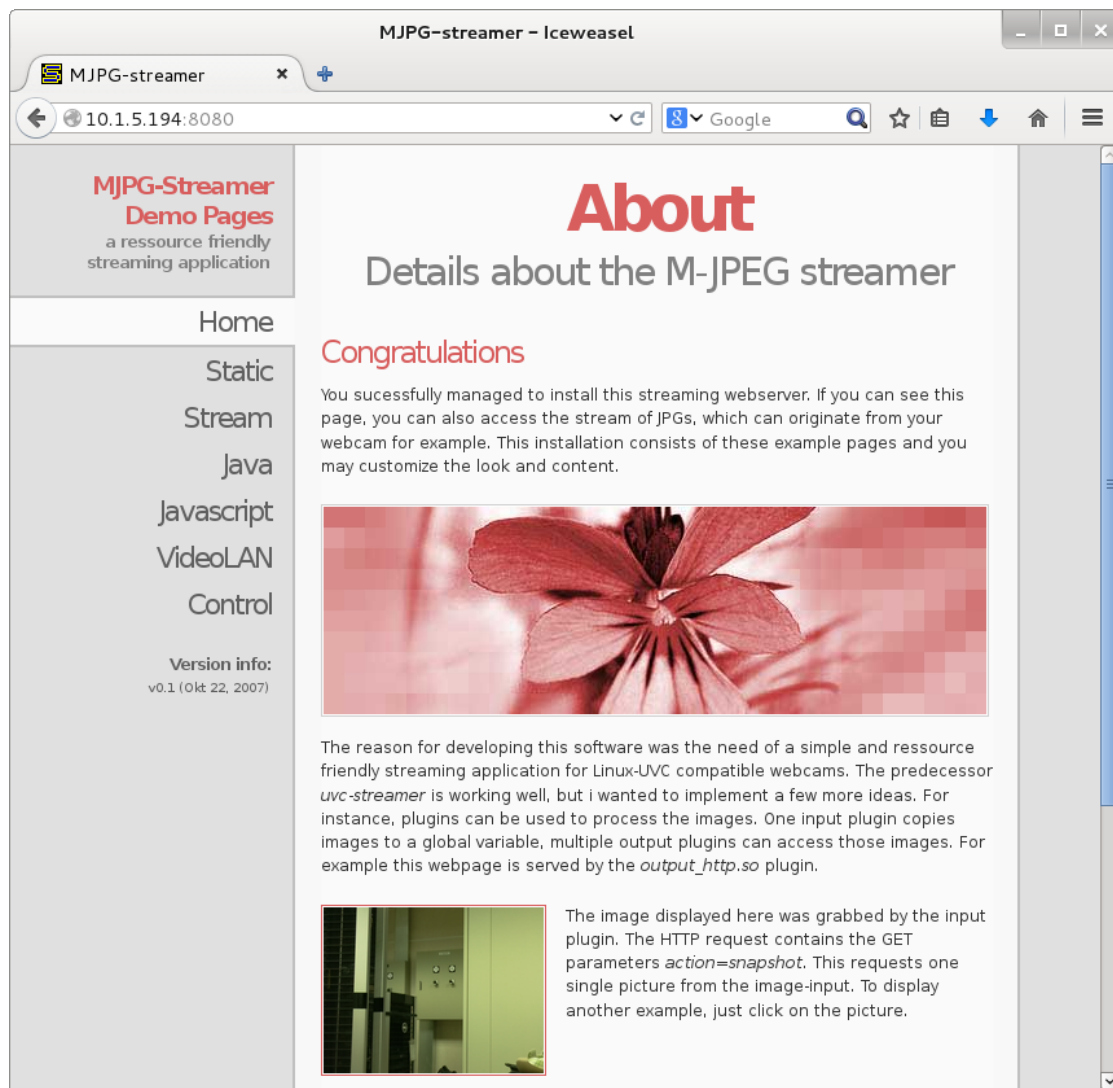


Figure 6 Camera Installation

wget 명령어를 사용하면 Web Streaming으로 부터 이미지를 저장할 수 있다.

```
scwook@scwook# wget http://10.1.5.194:8080/?action=snapshot -O image.jpg
```

Script를 만들면 주기적으로 이미지를 저장할 수 있다. 다음은 2초 간격으로 이미지를 저장하는 script이다.

```
while :
do
    DATE=$(date +"%Y-%m-%d_%H%M%S")
    wget -nv http://10.1.5.194:8080/?action=snapshot -O ./camera/$DATE.jpg
    sleep 2
done
```

camera 폴더를 만들고 script를 실행한다.

```
scwook@scwook# mkdir camera
scwook@scwook# sh capture.sh
```

mencoder를 이용하면 앞서 만든 여러장의 이미지를 하나의 Time-Lapse 동영상으로 만들 수 있다. 파일을 만들기 전에 우선 mencoder를 설치한다.

```
scwook@scwook# sudo aptitude install mencoder
```

동영상으로 만들 이미지 리스트를 stills.txt 파일로 dump 시킨다.

```
scwook@scwook# ls *.jpg > stills.txt
```

mencoder를 이용하여 이미지들을 동영상으로 변환 한다.

```
scwook@scwook# mencoder -nosound -ovc lavc -lavcopts vcodec=mpeg4:aspect=4/3:vbitrate=8000000 -vf
scale=640:480 -o timelapse.avi -mf type=jpeg:fps=24 mf://@stills.txt
```

2.2 GPIO

LED and Button Test

버튼을 누르면 LED가 켜지는 테스트를 해보자. 그림7과 같은 회로를 구성한다. 소스코드는 다음과 같다.

```
1 #include <stdio.h>
2 #include <wiringPi.h>
3
4 #define LED 4
5 #define BUTTON 1
6
7 int main(void)
8 {
9     if(wiringPiSetup() == -1)
10         return 1;
11
12     pinMode(LED, OUTPUT);
13     pinMode(BUTTON, INPUT);
14
15     digitalWrite(LED, 0);
16     int input = 0;
17
18     for(;;)
19     {
20         if(digitalRead(BUTTON))
21             digitalWrite(LED, 1);
22         else
23             digitalWrite(LED, 0);
24     }
```

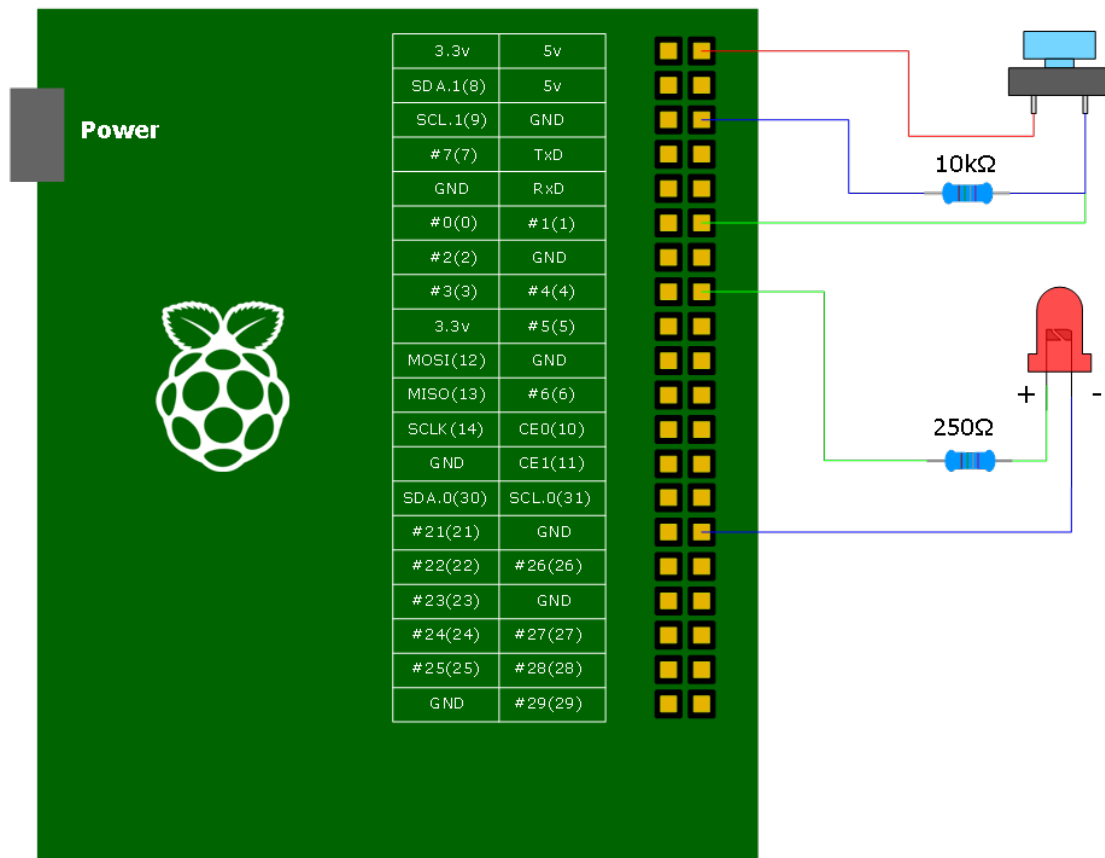


Figure 7 GPIO In/Out Test

```

25     delay(100);
26 }
27
28     return 0;
29 }

```

Listing 2.1 gpio.c

위 코드에서 pinMode함수를 통해 LED는 4번 Output, Button은 1번 Input으로 설정했음을 알 수 있다. 따라서 Button을 누르면 1번 Pin이 High 상태가 되고 이 값을 digitalRead함수를 통해 읽어진다. 반대로 digitalWrite 함수는 설정된 Pin 상태를 1(High) 또는 0(Low)상태로 설정할 수 있다. 위 코드에서는 Button의 상태에 따라 LED가 ON/OFF됨을 알 수 있다.

코드를 실행하기 위해 wiringPi Library를 추가한 후 컴파일 한다. 최종적으로 프로그램을 실행한 상태에서 Button을 눌렀을 때 LED에 불이 들어오는지 확인한다.

```

pi@raspberrypi# gcc -o buttonTest buttonTest.c -lwiringPi
pi@raspberrypi# sudo ./buttonTest

```

PIR Motion Sensor

PIR Motion 센서는 적외선을 이용하여 움직임을 감지하는 센서로 센서의 Detection Area에 움직임이 감지될 경우 3V의 출력을 내보낸다. 따라서 출력신호를 GPIO에 연결하여 센서의 상태를 읽을 수 있다. 센서에 대한 기본 사양은 다음과 같다.

- Input Voltage: 3.3 ~ 5V, 6V Maximum
- Working Current: 15uA
- Working Temperature: -20 ~ 85 °C
- Output Voltage: High 3V, low 0V
- Output Delay Time(High Level): About 2.3 to 3 Seconds
- Detection angle: 100 °
- Detection distance: 7 meters
- Output Indicator LED(When output HIGH, it will be ON)
- Pin limit current: 100mA

테스트를 위해 그림8과 같은 회로를 구성한다. 소스코드는 다음과 같다.

```
1 #include <stdio.h>
2 #include <wiringPi.h>
3
4 #define PIR 1
5
6 int main(void)
7 {
8     if(wiringPiSetup() == -1)
9         return 1;
10
11     pinMode(PIR, INPUT);
12
13     int input = 0;
14
15     for(;;)
16     {
17         if(digitalRead(PIR))
18             printf("Motion Detected!\n");
19
20         delay(100);
21     }
22
23     return 0;
24 }
```

Listing 2.2 pri.c

PIR Motion Sensor는 3V의 출력 신호가 나온다. 따라서 pinMode를 Input으로 설정하였으며 digitalRead 함수를 통해 값을 Monitoring한다. 위 코드에서는 Motion이 감지될 때 "Motion Detected!" 문자가 출력되어 Sensor의 작동을 확인할 수 있다. 코드를 실행하기 위해 wiringPi Library를 추가한 후 컴파일 한다. 최종적으로 프로그램을 실행한 상태에서 Motion이 감지되는지 확인한다.

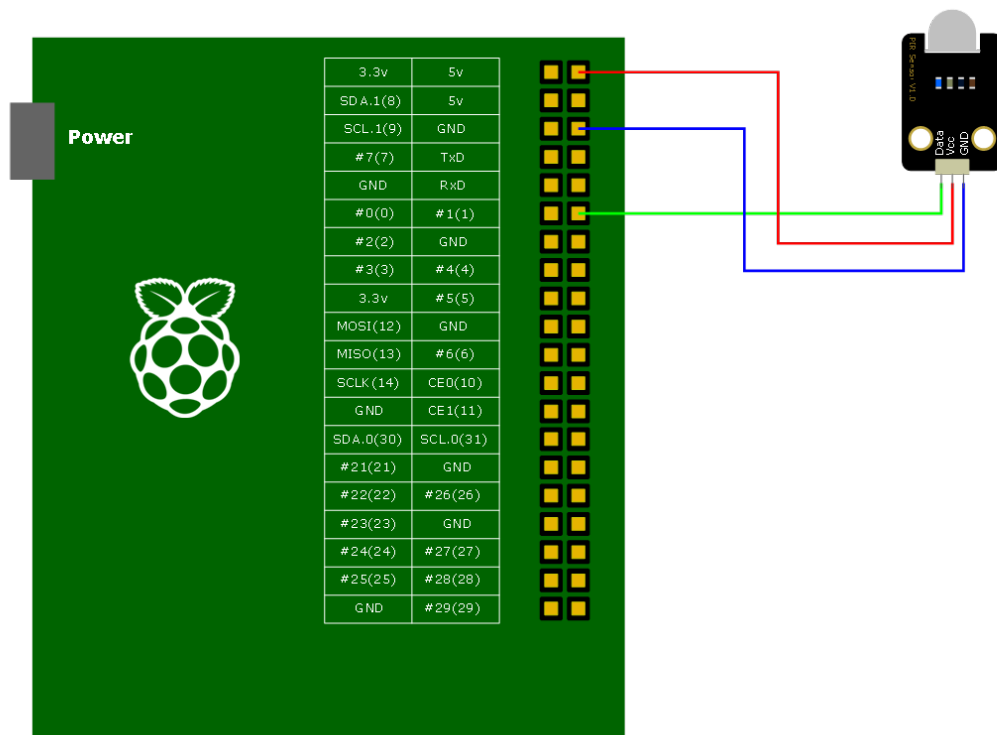


Figure 8 PIR Motion Sensor Test

```
pi@raspberrypi# gcc -o pir pir.c -lwiringPi
pi@raspberrypi# sudo ./pir
```

2.3 Humidity and Temperature Sensor

DHT11

DHT11 센서를 이용하여 온도와 습도를 읽어 보자. 그림9와 같은 회로를 구성한다. 소스코드는 다음과 같다.

```
1 #include <wiringPi.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <stdint.h>
5
6 #define MAX_TIME 85
7 #define DHT11PIN 1
8
9 int dht11_val[5]={0,0,0,0,0};
10
11 void dht11_read_val()
12 {
```

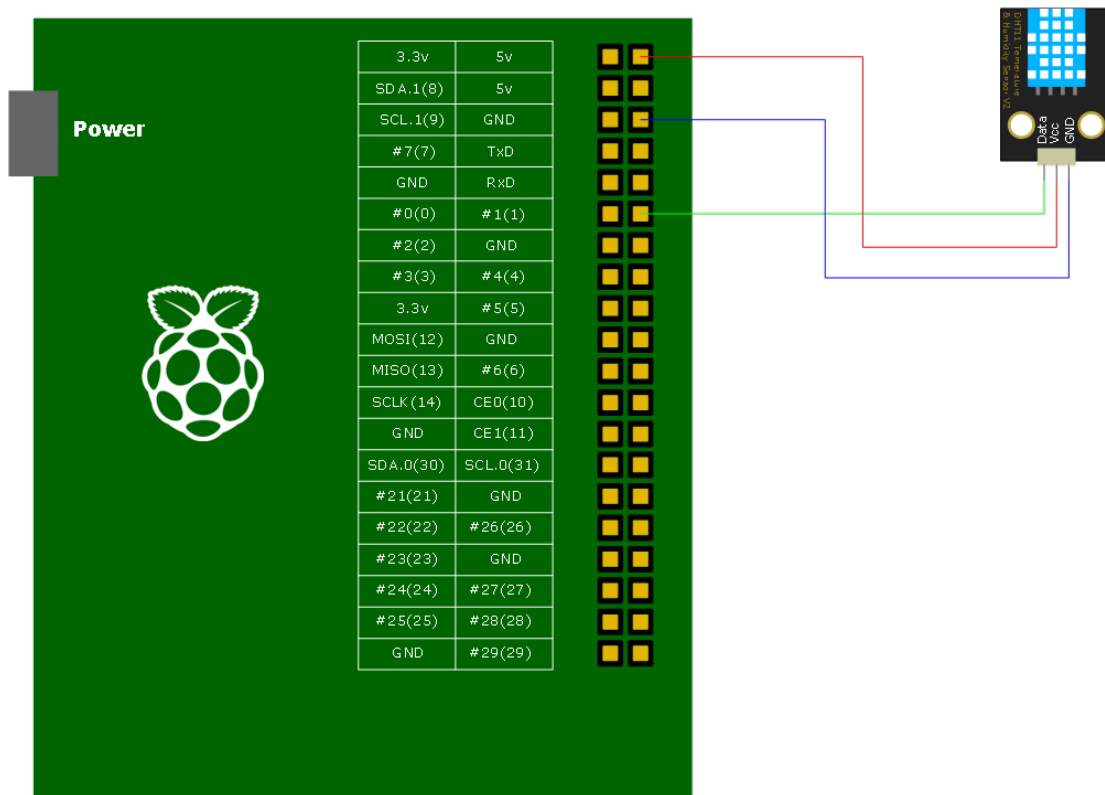


Figure 9 DHT11 Sensor Test

```

13 uint8_t lststate=HIGH;
14 uint8_t counter=0;
15 uint8_t j=0,i;
16
17 for(i=0;i<5;i++)
18     dht11_val[i]=0;
19
20 pinMode(DHT11PIN,OUTPUT);
21 digitalWrite(DHT11PIN,LOW);
22
23 delay(18);
24
25 digitalWrite(DHT11PIN,HIGH);
26
27 delayMicroseconds(40);
28
29 pinMode(DHT11PIN,INPUT);
30 for(i=0;i<MAX_TIME;i++)
31 {
32     counter=0;
33     while(digitalRead(DHT11PIN)==lststate){
34         counter++;
35         delayMicroseconds(1);
36         if(counter==255)
37             break;

```



```

38     }
39
40     lststate=digitalRead(DHT11PIN);
41
42     if(counter==255)
43         break;
44
45     // top 3 transistions are ignored
46     if((i>=4)&&(i%2==0)){
47         dht11_val[j/8]<=<=1;
48         if(counter>16)
49             dht11_val[j/8] |=1;
50         j++;
51     }
52 }
53
54 // verify cheksum and print the verified data
55 if((j>=40)&&(dht11_val[4]==((dht11_val[0]+dht11_val[1]+dht11_val[2]+dht11_val[3])& 0xFF)))
56 {
57     fahrenheit=dht11_val[2]*9./5.+32;
58     printf("H = %d.%d\nT = %d.%d\n",dht11_val[0],dht11_val[1],dht11_val[2],dht11_val[3]);
59 }
60 else
61     printf("Invalid Data!!\n");
62 }
63
64 int main(void)
65 {
66     if(wiringPiSetup()==-1)
67         exit(1);
68
69     while(1)
70     {
71         dht11_read_val();
72         delay(1000);
73     }
74
75     return 0;
76 }

```

Listing 2.3 dht11.c

read_dht11_val함수는 센서로 부터 data를 읽어 온습도값을 출력해주는 함수로 컴파일 후 실행한다.

```

pi@raspberrypi# gcc -o dht11 dht11.c -lwiringPi
pi@raspberrypi# sudo ./dht11

```

온습도 값이 출력되면 성공!

DS1820

DS1820 센서를 이용하여 온도를 읽어 보자. 소스코드는 다음과 같다.

```

1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <stdint.h>
5
6 #include <wiringPi.h>
7

```

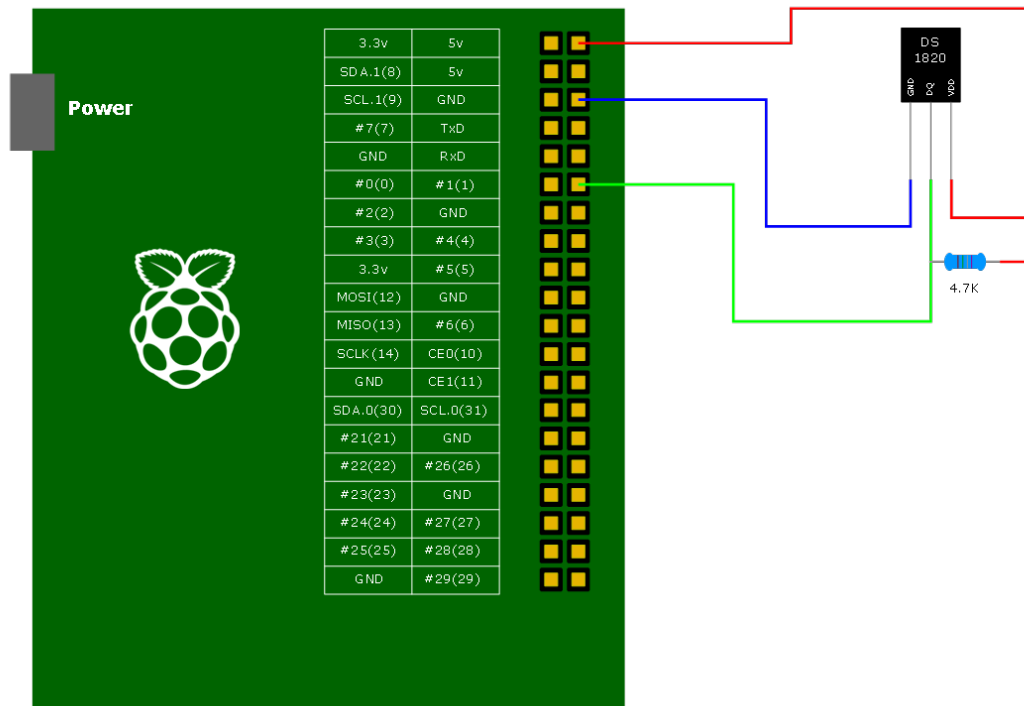


Figure 10 DS1820 Temperature Sensor Test

```

8  #define PIN_NUM 1
9
10 float ds1820_read();
11 int onewire_reset();
12 void onewire_write(uint8_t data);
13 void onewire_write_bit(int bit);
14 uint8_t onewire_read();
15 int onewire_read_bit();
16 uint8_t crc_read();
17 uint8_t crc_cal(uint8_t crc, uint8_t data);
18
19 int main()
20 {
21     if(wiringPiSetup() == -1)
22         return 1;
23
24     float temp = 0.0f;
25
26     while(1)
27     {
28         temp = ds1820_read();
29         printf("%.1f\n", temp);
30
31         delay(1000);
32     }

```

```

33 }
34 float ds1820_read()
35 {
36     uint8_t busy = 1;
37
38     onewire_reset();
39     onewire_write(0xCC);
40     onewire_write(0x44);
41
42     delay(750);
43     while(busy == 0)
44     {
45         busy = onewire_read();
46         printf("busy: %d\n", busy);
47     }
48
49     onewire_reset();
50     onewire_write(0xCC);
51     onewire_write(0xBE);
52
53     uint8_t lsb, msb, th, tl, reserved1, reserved2, count_remain, count_per_c, crc;
54     float real_temp = 0.0f;
55     signed char temp_read = 0;
56
57     lsb = onewire_read();
58     msb = onewire_read();
59     th = onewire_read();
60     tl = onewire_read();
61     reserved1 = onewire_read();
62     reserved2 = onewire_read();
63     count_remain = onewire_read();
64     count_per_c = onewire_read();
65     crc = onewire_read();
66
67     uint8_t data[] = {lsb, msb, th, tl, reserved1, reserved2, count_remain, count_per_c};
68
69     onewire_reset();
70
71     if(crc_read(data) == crc)
72     {
73         temp_read = (signed char)(lsb>>1);
74
75         if(msb == 255)
76             temp_read = temp_read | 0x80;
77
78         real_temp = (float)temp_read + 0.85f - (float)count_remain/(float)count_per_c;
79         real_temp = (int)(real_temp * 10) / 10.0f;
80     }
81     else
82         printf("CRC Error ");
83
84     return real_temp;
85 }
86
87 int onewire_reset()
88 {
89     int result;
90
91     pinMode(PIN_NUM, OUTPUT);
92
93     digitalWrite(PIN_NUM, LOW);
94     delayMicroseconds(480);

```

```

95
96     pinMode(PIN_NUM, INPUT);
97     delayMicroseconds(70);
98
99     result = digitalRead(PIN_NUM);
100
101     delayMicroseconds(410);
102
103     return result;
104 }
105
106 void onewire_write(uint8_t data)
107 {
108     int loop;
109
110     for(loop=0; loop<8; loop++)
111     {
112         onewire_write_bit(data & 0x01);
113
114         data >>= 1;
115     }
116 }
117 void onewire_write_bit(int bit)
118 {
119     pinMode(PIN_NUM, OUTPUT);
120
121     if(bit)
122     {
123         digitalWrite(PIN_NUM, LOW);
124         delayMicroseconds(6);
125         digitalWrite(PIN_NUM, HIGH);
126         delayMicroseconds(64);
127     }
128     else
129     {
130         digitalWrite(PIN_NUM, LOW);
131         delayMicroseconds(60);
132         digitalWrite(PIN_NUM, HIGH);
133         delayMicroseconds(10);
134     }
135 }
136 }
137
138 uint8_t onewire_read()
139 {
140     int loop, result=0;
141
142     for(loop=0; loop<8; loop++)
143     {
144         result >>= 1;
145
146         if(owewire_read_bit())
147             result |= 0x80;
148     }
149
150     return result;
151 }
152
153 int onewire_read_bit()
154 {
155     int result;
156

```

```

157     pinMode(PIN_NUM, OUTPUT);
158
159     digitalWrite(PIN_NUM, LOW);
160     delayMicroseconds(6);
161
162     pinMode(PIN_NUM, INPUT);
163     delayMicroseconds(9);
164
165     result = digitalRead(PIN_NUM) & 0x01;
166     delayMicroseconds(55);
167
168     return result;
169 }
170
171 uint8_t crc_read(uint8_t *data)
172 {
173     uint8_t i, crc;
174
175     crc = 0x00;
176
177     for(i=0; i<8; i++)
178         crc = crc_cal(crc, data[i]);
179
180     return crc;
181 }
182
183 uint8_t crc_cal(uint8_t crc, uint8_t data)
184 {
185     int j;
186     for(j=0; j<8; j++) {
187         if ((data & 0x01) ^ (crc & 0x01)) {
188             // DATA ^ LSB CRC = 1
189             crc = crc>>1;
190             // Set the MSB to 1
191             crc = crc | 0x80;
192             // Check bit 3
193             if (crc & 0x04) {
194                 crc = crc & 0xFB; // Bit 3 is set, so clear it
195             } else {
196                 crc = crc | 0x04; // Bit 3 is clear, so set it
197             }
198             // Check bit 4
199             if (crc & 0x08) {
200                 crc = crc & 0xF7; // Bit 4 is set, so clear it
201             } else {
202                 crc = crc | 0x08; // Bit 4 is clear, so set it
203             }
204         } else {
205             // DATA ^ LSB CRC = 0
206             crc = crc>>1;
207             // clear MSB
208             crc = crc & 0x7F;
209             // No need to check bits, with DATA ^ LSB CRC = 0, they will remain unchanged
210         }
211         data = data>>1;
212     }
213
214     return crc;
215 }

```

Listing 2.4 ds1820.c

ds1820_read함수는 bit data를 읽어 온도값을 return하는 함수로 크게 4개의 함수를 호출한다. onewire_reset함수는 command전송하기 위한 초기화 과정을 수행한다. 초기화가 완료되면 sensor는 ROM Command를 받을 준비가 되며 Family Code, Alarm Condition등과 같은 정보를 읽을 수 있다. onewire_write 함수는 sensor로 데이터를 전송하는 함수로 내부적으로 onewire_write_bit함수를 이용하여 1bit씩 전송한다. 위 코드에서는 Skip Command를 전송하여 다음 Mode로 진행하였다. ROM Command Process가 끝나면 Control Command Mode로 넘어간다. Control Command 상태에서는 Scratch Pad에 Write 또는 Read할 수 있는 명령 설정이 가능한데 Temperature Convert는 44h명령을 통해 이루어 진다. sensor가 Temperature Conver 명령을 전송받으면 9byte의 온도 데이터를 Scratch Pad에 저장할 하게 되고 Read Scratch Pad명령(BEh)과 onewire_read함수를 통해 데이터를 읽게 된다. 앞서 전송된 9byte의 데이터 중 상위 8byte는 온도계산을 위한 데이터 이며 마지막 1byte는 오류검사를 위한 CRC데이터가 들어있다. 따라서 CRC데이터는 상위 8byte의 데이터를 이용하여 계산할 수 있으며 계산된 값과 마지막 1byte의 CRC값을 비교함으로써 데이터의 유효성을 판단할 수 있다. 최종적인 온도값은 다음 8byte의 정보에 따라 계산되어 진다.

BYTE 0: Temperature LSB
 BYTE 1: Temperature MSB
 BYTE 2: TH/USER Byte 1
 BYTE 3: TL/USER Byte 2
 BYTE 4: Reserved
 BYTE 5: Reserved
 BYTE 6: Count Remain
 BYTE 7: Count Per C

$$\text{Temperature} = \text{Temp Read} + (\text{Count Per C} - \text{Count Remain}) / \text{Count Per C}$$

여기서 Temp Read값은 LSB와 MSB의 bit연산을 이용하여 만든 값으로 LSB의 8bit를 bit shift연산을 이용하여 1번 bit를 자른다. 여기서 MSB는 부호bit으로 온도가 음수일 경우 255(bit로는 11111111) 양수면 0(bit로는 00000000)이다. 따라서 255또는 0값을 판단하여 LSB에서 남은 상위1bit를 1또는 0으로 설정한 값이 Temp Read값이 된다. 코드를 실행하기 위해 wiringPi Library를 추가한 후 컴파일 한다. 최종적으로 프로그램을 실행하여 온도 값이 출력되는지 확인한다.

```
pi@raspberrypi# gcc -o ds1820 ds1820.c -lwiringPi
pi@raspberrypi# sudo ./ds1820
```

2.4 Dust Sensor

PM1001

PM1001 Dust Sensor는 공기중에 존재하는 1um이상의 입자를 검출하여 농도 값을 출력해 주는 센서로 UART 통신을 통해 데이터를 전송하므로 Raspberry Pi의 UART 포트와 연결이 가능하다. 센서에 연결하기 전에 1.2를 참고하여 UART Consol 설정을 해제한다.

테스트 회로는 그림 11와 같다. 메뉴얼에 따르면 PM1001의 통신 명령어는 다음과 같다.

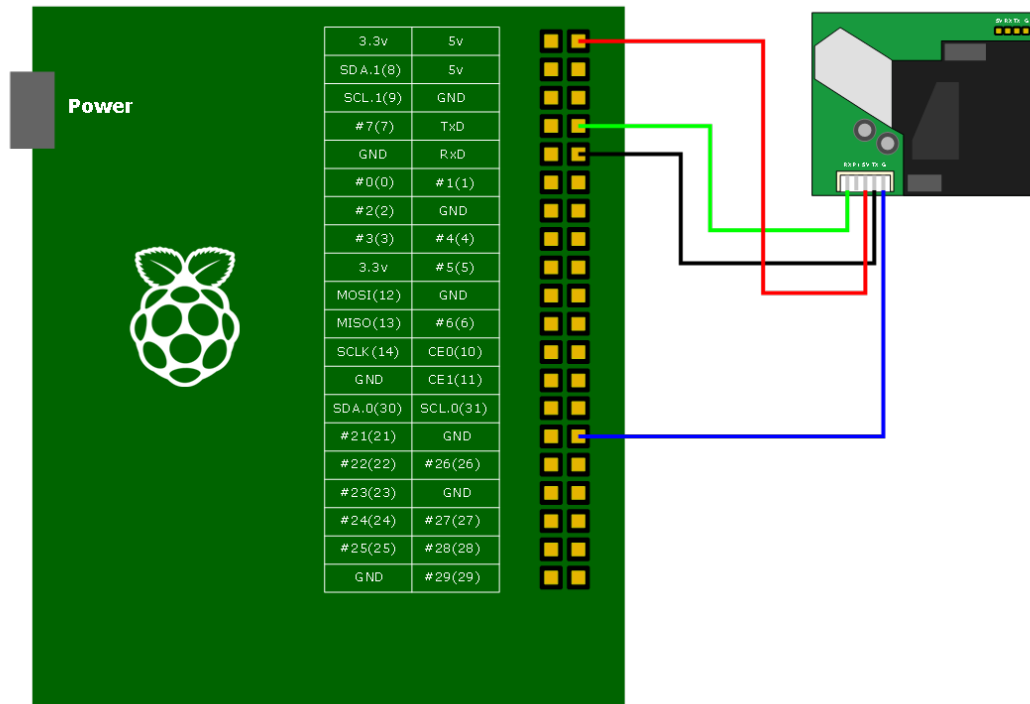


Figure 11 PM1001 Dust Sensor

SEND: [IP] [LB] [CMD] [DF] [CS]

RESPONSE: [ACK] [LB] [CMD] [DF] [CS]

여기서 각 명령에 대한 의미는 다음과 같다.

[IP]: address(fixed as 0x11)

[LB]: byte length followed does not include CS

[CMD]: command

[DF]: parameter items with command, optional

[CS]: CS = -(IP + LB + CMD + DF)

[ACK] 0x16 right command

예를 들어 PM1001로 부터 먼지 값을 읽는 명령어는 다음과 같다.

SEND: 0X11, 0X01, 0X01, 0XED

RESPONSE: 0x16, 0x0D, 0x01, 4BytePM값, 4BytePM값, 4BytePM값, [CS]

여기서 PM값은 4Byte(DF0, DF1, DF2, DF3)로 구성된 먼지 데이터 값으로 측정 값은 다음과 같다.

Measured value = DF0 * 256 * 256 * 256 + DF1 * 256 * 256 + DF2 * 256 + DF3

기본 단위는 PCS/L 로 농도($\mu\text{g}/\text{m}^3$)값으로 변환 하고자 할 경우 다음 식을 사용한다.

농도($\mu\text{g}/\text{m}^3$) = ((수량PCS/L값) * 3,528) / 100,000

전체 코드는 다음과 같다.

```
1 #include <stdio.h>
2 #include <wiringPi.h>
3 #include <wiringSerial.h>
4
5 int main(void)
6 {
7     int fd;
8
9     if((fd = serialOpen("/dev/ttyAMA0", 9600)) < 0)
10     {
11         printf("Unable to open serial device\n");
12         return 1;
13     }
14
15     if(wiringPiSetup() == -1)
16     {
17         printf("Unable to start wiringPi\n");
18         return 1;
19     }
20
21     serialFlush(fd);
22
23     unsigned char c;
24     int df[4];
25     int dust;
26
27     c = 0x11;
28     serialPutchar(fd, c);
29
30     c = 0x01;
31     serialPutchar(fd, c);
32     serialPutchar(fd, c);
33
34     c = 0xED;
35     serialPutchar(fd, c);
36
37     printf("%d ", serialGetchar(fd));
38     printf("%d ", serialGetchar(fd));
39     printf("%d ", serialGetchar(fd));
40
41     int i=0, j;
42     for(i; i<3; i++)
43     {
44         j=0;
```



```

45     for(j; j<4; j++)
46         df[j] = serialGetchar(fd);
47
48     dust = df[0]*256*256*256 + df[1]*256*256 + df[2]*256 + df[3];
49     printf("%d ", dust);
50 }
51
52 printf("%d\n", serialGetchar(fd));
53 printf("%d %d %d %d\n", df[0], df[1], df[2], df[3]);
54
55 serialFlush(fd);
56 serialClose(fd);
57
58 return 0;
59 }

```

Listing 2.5 pm1001.c

UART는 Serial통신이므로 wiringPi에서 제공하는 Serial Library를 이용하면 쉽게 코드를 만들 수 있다. sensor와 데이터를 주고 받기 위해 serialOpen함수를 이용하여 Serial Connection작업을 수행한다. Connection이 성공적으로 완료되면 serialFlush함수를 이용하여 Buffer를 Clear하고 Data를 전송할 준비를 한다. 이 후 serialPutchar함수를 통해 Data Request명령을 전송하고 serialGetchar함수를 이용하여 Data를 읽는다. 마지막으로 Buffer를 Clear하고 Serial Connection을 Close한다.

코드를 실행하기 위해 wiringPi Library를 추가한 후 컴파일 한다. 최종적으로 프로그램을 실행하여 Dust 값이 출력되는지 확인한다.

```

pi@raspberrypi# gcc -o pm1001 pm1001.c -lwiringPi
pi@raspberrypi# sudo ./pm1001

```

2.5 Motor

L298 Dual H-Bridge

L298 Dual H-Bridge를 이용하여 2Phase Motor를 작동시켜 보자. 그림12와 같은 회로를 구성한다. 소스코드는 다음과 같다.

```

1  #include <stdio.h>
2  #include <wiringPi.h>
3
4  #define TRUE 1
5  #define FALSE 0
6  #define DELAY 1800
7
8  #define EA 15
9  #define EB 4
10 #define IN1 16
11 #define IN2 1
12 #define IN3 5
13 #define IN4 6
14
15 void setStep(int a, int b, int c, int d)
16 {
17     digitalWrite(IN1, a);
18     digitalWrite(IN2, b);
19     digitalWrite(IN3, c);

```

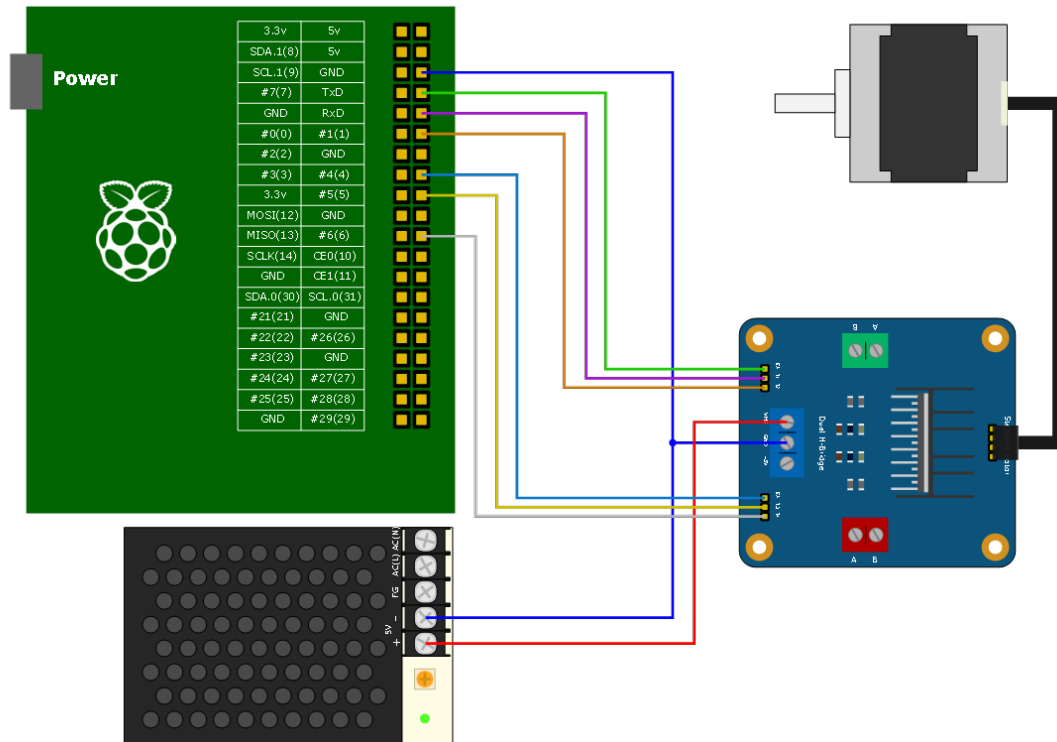


Figure 12 L298 Dual H-Bridge 2Phase Step Motor

```

20     digitalWrite(IN4, d);
21 }
22
23 int main(void)
24 {
25     if(wiringPiSetup() == -1)
26     {
27         printf("Init Error\n");
28         return 1;
29     }
30
31     pinMode(EA, OUTPUT);
32     pinMode(IN1, OUTPUT);
33     pinMode(IN2, OUTPUT);
34     pinMode(EB, OUTPUT);
35     pinMode(IN3, OUTPUT);
36     pinMode(IN4, OUTPUT);
37
38     digitalWrite(EA, TRUE);
39     digitalWrite(EB, TRUE);
40
41     int i;
42     int loop;
43
44     for(;;)

```

```

45 {
46     for(i=0; i<500; i++)
47     {
48         setStep(1,0,1,0);
49         delayMicroseconds(DELAY);
50         setStep(0,1,1,0);
51         delayMicroseconds(DELAY);
52         setStep(0,1,0,1);
53         delayMicroseconds(DELAY);
54         setStep(1,0,0,1);
55         delayMicroseconds(DELAY);
56     }
57
58     delay(1000);
59
60     for(i=0; i<500; i++)
61     {
62         setStep(1,0,0,1);
63         delayMicroseconds(DELAY);
64         setStep(0,1,0,1);
65         delayMicroseconds(DELAY);
66         setStep(0,1,1,0);
67         delayMicroseconds(DELAY);
68         setStep(1,0,1,0);
69         delayMicroseconds(DELAY);
70     }
71
72     delay(1000);
73
74 }
75
76 digitalWrite(EA, FALSE);
77 digitalWrite(EB, FALSE);

```

Listing 2.6 l298.c

위 코드에서 setStep함수는 IN신호를 만드는 함수로 총 4번의 Step이 1Cycle이 된다. 1Step당 1.8도씩 회전하며 총 회전수는 반복문을 통해 제어 가능하다. 따라서 모터를 1회전 하고자 하면 반복 횟수를 $50(360 / 1.8 / 4)$ 으로 하면 된다. 모터 속도는 DELAY시간에 따라 바뀌는데 시간이 너무 짧은 경우 회전하지 않는다. 보통 모터마다 최대 응답속도가 있으므로 그에 맞게 조절해야 한다. 테스트한 모터의 경우 1.8ms(대략 167rpm)보다 짧은 경우 불규칙적인 회전을 보인다. Step신호를 반대로 주면 순서가 반대로 작용하므로 모터 방향이 바뀐다. 모터의 방향은 Step순서를 반대로 하면 된다.

코드를 실행하기 위해 wiringPi Library를 추가한 후 컴파일 한다. 최종적으로 프로그램을 실행하여 모터가 작동하는지 확인한다.

```

pi@raspberrypi# gcc -o l298 l298.c -lwiringPi
pi@raspberrypi# sudo ./l298

```

MD5-DH14

MD-DH14 Motor Driver를 이용하여 5Phase Pentagon방식의 Step Motor를 작동시켜 보자. 그림 13과 같이 회로를 구성한다. 소스코드는 다음과 같다.

```

1 #include <stdio.h>
2 #include <wiringPi.h>
3

```

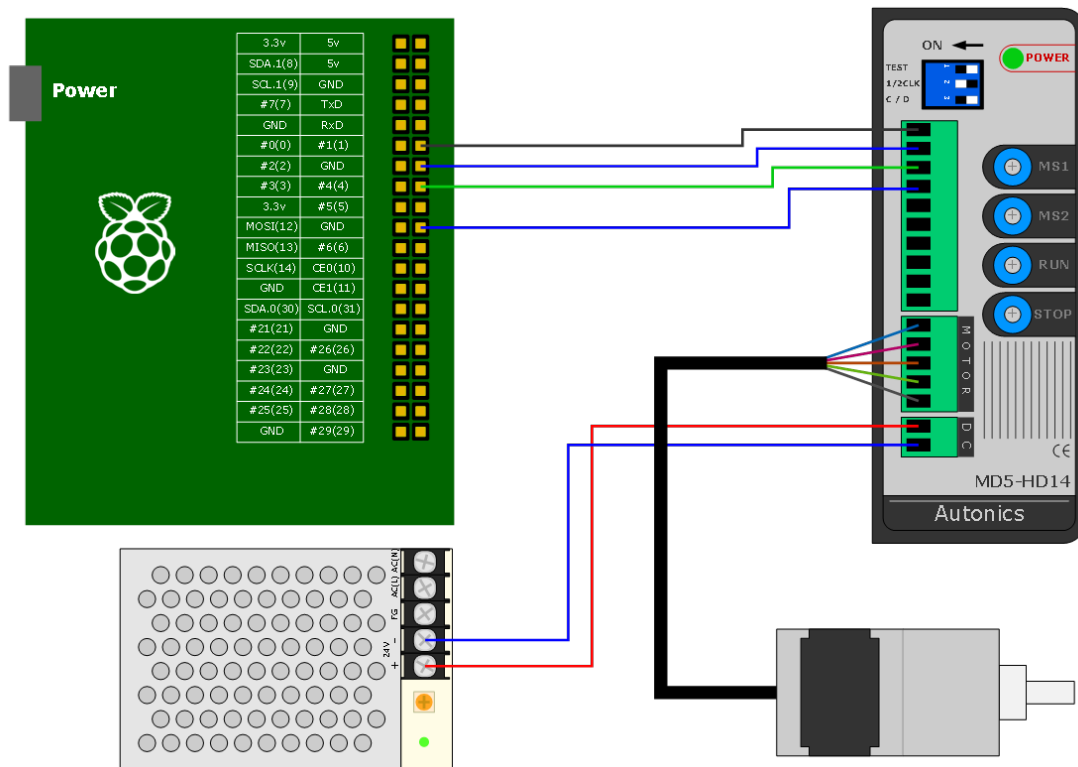


Figure 13 5Phase Motor Test

```

4 #define PULSE 5000
5
6 int main(void)
7 {
8     if(wiringPiSetup() == -1)
9     {
10         printf("Init Error\n");
11         return 1;
12     }
13
14     pinMode(1, OUTPUT);
15     pinMode(4, OUTPUT);
16
17     int pulse;
18     for(;;)
19     {
20         digitalWrite(4, 0);
21         for(pulse=0; pulse<PULSE; pulse++)
22         {
23             digitalWrite(1, 1);
24             delayMicroseconds(500);
25             digitalWrite(1, 0);
26             delayMicroseconds(500);
27         }
28     }

```

```

29     digitalWrite(4, 1);
30     for(pulse=0; pulse<PULSE; pulse++)
31     {
32         digitalWrite(1, 1);
33         delayMicroseconds(500);
34         digitalWrite(1, 0);
35         delayMicroseconds(500);
36     }
37 }
38 }

```

Listing 2.7 md5dh14.c

코드를 실행하기 위해 wiringPi Library를 추가한 후 컴파일 한다. 최종적으로 프로그램을 실행하여 모터가 작동하는지 확인한다.

```

pi@raspberrypi# gcc -o dm5dh14 dm5dh14.c -lwiringPi
pi@raspberrypi# sudo ./dm5dh14

```

3 EPICS Integration

여기서는 앞서 테스트한 Sensor 및 Device를 EPICS에 Integration하고 Channel Access를 통해 값을 Monitoring 또는 Control하는 방법에 대하여 기술하였다. 본 Chapter에서 사용된 EPICS는 3.14.12.4 버전이며 전체적으로 다음과 같은 구조를 바탕으로 하고 있다.

3.1 Installation

위 구조에 맞게 EPICS를 설치하기 위해 git으로 부터 script파일을 받는다.

```

pi@raspberrypi# git clone https://github.com/jeonghanlee/scripts_for_epics

```

3.2 Library

synApps

3.3 GPIO

section 2.2에서 진행하였던 GPIO 테스트를 EPICS로 Integration 하여 테스트해 본다. 최종 목표는 그림 7과 같이 회로를 구성하고 다음 epics db를 통해 GPIO를 제어하는 것이다.

```

record(bi, "inp1")
{
    field(DTYP, "GPIO")
    field(SCAN, "1 second")
    field(INP, "@1")
}

record(bo, "out4")
{

```

```

    field(DTYP, "GPIO")
    field(OUT, "@4")
}

```

코드 작성에 앞서 테스트 할 기본 폴더 및 EPICS Application 구조를 생성한다.

```

pi@ctrlpi3 cd ../epics/R3.14.12.4/siteApps
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps# mkdir gpio
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps# cd gpio
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/gpio# makeBaseApp.pl -t ioc gpio
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/gpio# makeBaseApp.pl -i -t ioc gpio

```

```

Using target architecture linux-arm (only one available)
The following applications are available:
    gpio
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed above.
Application name? gpio
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/gpio# ls
configure  gpioApp  iocBoot  Makefile

```

gpioApp/src 폴더로 이동한 후 devGPIO.c 파일을 만들어 다음과 같이 코드를 작성한다.

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  #include <epicsExport.h>
6  #include <devSup.h>
7  #include <boRecord.h>
8  #include <biRecord.h>
9
10 #include <wiringPi.h>
11
12 static long bo_init_record(boRecord *pbo);
13 static long bi_init_record(biRecord *pbi);
14
15 static long write_bo(boRecord *pbo);
16 static long read_bi(biRecord *pbi);
17
18 struct Pin_Info
19 {
20     int pin_num;
21 };
22
23 static long bo_init_record(boRecord *pbo)
24 {
25     struct Pin_Info *pin_info = malloc(sizeof(struct Pin_Info));
26
27     if(wiringPiSetup() == -1)
28         return 1;
29
30     int pin_num = 0;
31     pin_num = atoi(pbo->out.value.instio.string);
32
33     pinMode(pin_num, OUTPUT);
34
35     pin_info->pin_num = pin_num;
36
37     pbo->dpvt = pin_info;
38
39     return 0;
40 }

```

```

41
42 static long bi_init_record(biRecord *pbi)
43 {
44     struct Pin_Info *pin_info = malloc(sizeof(struct Pin_Info));
45
46     if(wiringPiSetup() == -1)
47         return 1;
48
49     int pin_num = 0;
50     pin_num = atoi(pbi->inp.value.instio.string);
51
52     pinMode(pin_num, INPUT);
53
54     pin_info->pin_num = pin_num;
55
56     pbi->dpvt = pin_info;
57
58     return 0;
59 }
60
61
62 static long write_bo(boRecord *pbo)
63 {
64     struct Pin_Info *pin_info = pbo->dpvt;
65
66     int pin = pin_info->pin_num;
67     int val = pbo->rval;
68
69     digitalWrite(pin, val);
70
71     return 0;
72 }
73
74 static long read_bi(biRecord *pbi)
75 {
76     struct Pin_Info *pin_info = pbi->dpvt;
77
78     int pin = pin_info->pin_num;
79     int val = digitalRead(pin);
80
81     pbi->rval = val;
82
83     return 0;
84 }
85
86 struct
87 {
88     long num;
89     DEVSUPFUN    report;
90     DEVSUPFUN    init;
91     DEVSUPFUN    init_record;
92     DEVSUPFUN    get_ioint_info;
93     DEVSUPFUN    write_bo;
94     DEVSUPFUN    special_linconv;
95 } devBoGpioAsync = {
96     6,
97     NULL,
98     NULL,
99     bo_init_record,
100     NULL,
101     write_bo,
102     NULL

```

```

103 };
104
105 struct
106 {
107     long num;
108     DEVSUPFUN    report;
109     DEVSUPFUN    init;
110     DEVSUPFUN    init_record;
111     DEVSUPFUN    get_ioint_info;
112     DEVSUPFUN    read_bi;
113     DEVSUPFUN    special_linconv;
114 } devBiGpioAsync = {
115     6,
116     NULL,
117     NULL,
118     bi_init_record,
119     NULL,
120     read_bi,
121     NULL
122 };
123
124 epicsExportAddress(dset,devBoGpioAsync);
125 epicsExportAddress(dset,devBiGpioAsync);

```

Listing 3.1 devGPIO.c

코드 작성이 완료되면 devGPIO.dbd 파일을 만든다.

```

device(bo, INST_IO, devBoGpioAsync, "GPIO")
device(bi, INST_IO, devBiGpioAsync, "GPIO")

```

앞서 작성한 코드를 Build하기 위해 Makefile에 다음과 같이 추가한다.

```

TOP=../..

include $(TOP)/configure/CONFIG

USR_INCLUDES += -I/home/pi/wiringPi/wiringPi
wiringPi_DIR += /home/pi/wiringPi/wiringPi /home/pi/wiringPi/devLib

#-----
#  ADD MACRO DEFINITIONS AFTER THIS LINE
#=====

#=====
# Build the IOC application

PROD_IOC = gpio
# gpio.dbd will be created and installed
DBD += gpio.dbd

# gpio.dbd will be made up from these files:
gpio_DBD += base.dbd

# Include dbd files from all support applications:
#gpio_DBD += xxx.dbd
gpio_DBD += devGPIO.dbd

# Add all the support libraries needed by this IOC
#gpio_LIBS += xxx
gpio_LIBS += wiringPi

# gpio_registerRecordDeviceDriver.cpp derives from gpio.dbd

```



```

gpio_SRCS += gpio_registerRecordDeviceDriver.cpp
gpio_SRCS += devGPIO.c

# Build the main IOC entry point on workstation OSs.
gpio_SRCS_DEFAULT += gpioMain.cpp
gpio_SRCS_vxWorks += -nil-

# Add support from base/src/vxWorks if needed
#gpio_OBJS_vxWorks += $(EPICS_BASE_BIN)/vxComLibrary

# Finally link to the EPICS Base libraries
gpio_LIBS += $(EPICS_BASE_IOC_LIBS)

#=====

include $(TOP)/configure/RULES
#-----
# ADD RULES AFTER THIS LINE

```

db파일을 만들기 위해 gpioApp/Db 폴더로 이동한 후 다음과 같은 gpio.db 파일을 만든다.

```

record(bi, "inp1")
{
    field(DTYP, "GPIO")
    field(SCAN, "1 second")
    field(INP, "@1")
}

record(bo, "out4")
{
    field(DTYP, "GPIO")
    field(OUT, "@4")
}

```

gpio.db에서는 GPIO 1번을 입력으로 4번을 출력으로 설정했음을 알 수있다. Makefile에 db파일을 추가해 준다.

```

TOP=../..
include $(TOP)/configure/CONFIG
#-----
# ADD MACRO DEFINITIONS AFTER THIS LINE

#-----
# Optimization of db files using dbst (DEFAULT: NO)
#DB_OPT = YES

#-----
# Create and install (or just install) into /db
# databases, templates, substitutions like this
#DB += xxx.db
DB += gpio.db

#-----
# If .db template is not named *.template add
# _template =

include $(TOP)/configure/RULES
#-----
# ADD RULES AFTER THIS LINE

```

마지막으로 앞서 작성한 파일들을 컴파일 하기 위해 gpio폴더로 이동한 후 make를 실행한다.

```
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/gpio# make
```

make가 완료되면 bin/linux-arm 폴더에 gpio 파일과 db 폴더에 gpio.db 파일이 만들어진다. 이제 ioc를 실행시키기 위해 iocBoot/iocgpio 폴더로 이동 후 st.cmd파일에 gpio.db를 Load하는 코드를 추가해 준다.

```
#!/../bin/linux-arm/gpio

## You may have to change gpio to something else
## everywhere it appears in this file

< envPaths

cd ${TOP}

## Register all support components
dbLoadDatabase "dbd/gpio.dbd"
gpio_registerRecordDeviceDriver pdbname

## Load record instances
#dbLoadTemplate "db/userHost.substitutions"
#dbLoadRecords "db/dbSubExample.db", "user=piHost"
dbLoadRecords "db/gpio.db"

## Set this to see messages from mySub
#var mySubDebug 1

## Run this to trace the stages of iocInit
#traceIocInit

cd ${TOP}/iocBoot/${IOC}
iocInit

## Start any sequence programs
#seq sncExample, "user=piHost"
```

마지막으로 st.cmd파일을 실행파일로 변경한 후 실행시킨다.

```
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/gpio/iocBoot/iocdht11 $ chmod 755 st.cmd
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/gpio/iocBoot/iocdht11 $ sudo ./st.cmd
```

output 테스트를 위해 다음과 같이 out4 출력값을 1로 하고 LED에 불이 들어오는지 확인한다.

```
epics> dbpf out4 1
```

input 테스트는 버튼을 눌렀을 때 inp1 값을 읽어 확인한다.

```
epics> dbpr inp1
ASG:          DESC:          DISA: 0          DISP: 0
DISV: 1       NAME: gpio:inp1  RVAL: 0          SEVR: NO_ALARM
STAT: NO_ALARM SVAL: 0        TPRO: 0          VAL: 1
```

3.4 Dust Sensor

PM1001

section 2.4에서 진행하였던 PM1001 Sensor를 EPICS로 Integration 하여 테스트해 본다. 최종 목표는 그림 11와 같이 회로를 구성하고 다음 epics db를 통해 먼지값을 읽는 것이다.

```
record(ai,"PI:DUST")
{
  field(DTYP, "stream")
  field(INP, "@sensor.proto get_dust UART")
  field(SCAN, "1 second")
}
```

코드 작성에 앞서 기본 폴더 및 EPICS Application 구조를 생성한다.

```
pi@raspberrypi ~/epics/R3.14.12.4 $ cd siteApps
pi@raspberrypi ~/epics/R3.14.12.4/siteApps $ mkdir pm1001
pi@raspberrypi ~/epics/R3.14.12.4/siteApps $ cd pm1001
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/pm1001 $ makeBaseApp.pl -t ioc pm1001
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/pm1001 $ makeBaseApp.pl -i -t ioc pm1001
Using target architecture linux-arm (only one available)
The following applications are available:
  pm1001
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed above.
Application name? pm1001
```

pm1001/configure/RELEASE 파일에 asyn과 stream Library 위치를 추가해 준다.

```
# RELEASE - Location of external support modules
#
# IF YOU MAKE ANY CHANGES to this file you must subsequently
# do a "gnumake rebuild" in this application's top level
# directory.
#
# The build process does not check dependencies against files
# that are outside this application, thus you should do a
# "gnumake rebuild" in the top level directory after EPICS_BASE
# or any other external module pointed to below is rebuilt.
#
# Host- or target-specific settings can be given in files named
# RELEASE.$(EPICS_HOST_ARCH).Common
# RELEASE.Common.$(T_A)
# RELEASE.$(EPICS_HOST_ARCH).$(T_A)
#
# This file should ONLY define paths to other support modules,
# or include statements that pull in similar RELEASE files.
# Build settings that are NOT module paths should appear in a
# CONFIG_SITE file.

TEMPLATE_TOP=$(EPICS_BASE)/templates/makeBaseApp/top

# If using the sequencer, point SNCSEQ at its top directory:
#SNCSEQ=$(EPICS_BASE)/../modules/soft/seq

# EPICS_BASE usually appears last so other apps can override stuff:
EPICS_BASE=/home/pi/epics/R3.14.12.4/base

# Set RULES here if you want to take build rules from somewhere
```

```
# other than EPICS_BASE:
#RULES=/path/to/epics/support/module/rules/x-y
```

```
ASYN=$(EPICS_PATH)/siteLibs
STREAM=$(EPICS_PATH)/siteLibs
```

pm1001App/src 폴더로 이동하면 pm1001Main.cpp와 Makefile이 있다. Makefile에 다음 코드를 추가한다.

```
TOP=../..
```

```
include $(TOP)/configure/CONFIG
#-----
# ADD MACRO DEFINITIONS AFTER THIS LINE
#=====

#=====
# Build the IOC application

PROD_IOC = pm1001
# pm1001.dbd will be created and installed
DBD += pm1001.dbd

# pm1001.dbd will be made up from these files:
pm1001_DBD += base.dbd

# Include dbd files from all support applications:
#pm1001_DBD += xxx.dbd

pm1001_DBD += stream.dbd
pm1001_DBD += drvAsynSerialPort.dbd

# Add all the support libraries needed by this IOC
#pm1001_LIBS += xxx

pm1001_LIBS += stream
pm1001_LIBS += asyn

# pm1001_registerRecordDeviceDriver.cpp derives from pm1001.dbd
pm1001_SRCS += pm1001_registerRecordDeviceDriver.cpp

# Build the main IOC entry point on workstation OSs.
pm1001_SRCS_DEFAULT += pm1001Main.cpp
pm1001_SRCS_vxWorks += -nil-

# Add support from base/src/vxWorks if needed
#pm1001_OBJS_vxWorks += $(EPICS_BASE_BIN)/vxComLibrary

# Finally link to the EPICS Base libraries
pm1001_LIBS += $(EPICS_BASE_IOC_LIBS)

#=====

include $(TOP)/configure/RULES
#-----
# ADD RULES AFTER THIS LINE
```

db파일을 만들기 위해 pm1001App/Db 폴더로 이동한 후 다음과 같은 pm1001.db 파일을 만든다.

```
record(ai,"PM1001:DUST")
{
```

```

    field(DTYP, "stream")
    field(INP, "@sensor.proto get_dust UART")
    field(SCAN, "1 second")
}

```

Makefile에 db파일을 추가해 준다.

```

TOP=../..
include $(TOP)/configure/CONFIG
#-----
#  ADD MACRO DEFINITIONS AFTER THIS LINE

#-----
#  Optimization of db files using dbst (DEFAULT: NO)
#DB_OPT = YES

#-----
#  Create and install (or just install) into /db
#  databases, templates, substitutions like this
#DB += xxx.db
DB += pm1001.db

#-----
#  If .db template is not named *.template add
#  _template =

include $(TOP)/configure/RULES
#-----
#  ADD RULES AFTER THIS LINE

```

protocol 파일을 만들기 위해 폴더를 생성한 후 다음과 같은 pm1001.proto 파일을 만든다.

```

pi@raspberrypi ~/epics/R3.14.12.4/siteApps/pm1001 $ mkdir proto
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/pm1001 $ cd proto

```

```

get_dust{
    out "\x11\x01\x01\xED";
    in "%*3r%4r%*4r%*4r%*1r";
}

```

get_dust 함수는 out을 통해 먼지 값을 읽어오는 명령을 전송한다. pm1001 센서는 응답 값으로 총 16byte의 값을 리턴하는데 이 중 먼지 데이터는 처음 3byte 이후 4byte 씩 3번 반복되므로 첫 4byte만 저장하고 checksum을 포함한 나머지 byte는 무시한다. 참고로 읽고자 하는 값을 무시 하고 싶을 때는 '*'를 앞에다 붙이면 된다. 여기에서는 4byte를 읽으므로 앞서 말한 Measured value를 계산하기 위해 256을 곱하지 않아도 된다. 만약 DF0 DF3을 따로 읽고자 하면 다음과 같이 1byte씩 읽으면 된다.

```

get_d0{
    out "\x11\x01\x01\xED";
    in "%*3r%1r%*3r%*4r%*4r%*1r";
}

get_d1{
    out "\x11\x01\x01\xED";
    in "%*3r%*1r%1r%*2r%*4r%*4r%*1r";
}

get_d2{
    out "\x11\x01\x01\xED";
}

```

```

    in "%*3r%*2r%1r%*1r%*4r%*4r%*1r";
}

get_d3{
    out "\x11\x01\x01\xED";
    in "%*3r%*3r%1r%*4r%*4r%*1r";
}

```

마지막으로 앞서 작성한 파일들을 컴파일 하기 위해 pm1001폴더로 이동한 후 make를 실행한다.

```
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/pm1001 $ make
```

make가 완료되면 bin/linux-arm 폴더에 pm1001파일과 db 폴더에 pm1001.db 파일이 만들어 진다. 이제 ioc를 실행시키기 위해 iocBoot/iocpm1001 폴더로 이동 후 st.cmd 파일에 serial 연결을 위한 설정 및 pm1001.db를 Load하는 코드를 추가해 준다.

```

#!/../bin/linux-arm/pm1001

## You may have to change pm1001 to something else
## everywhere it appears in this file

< envPaths

cd ${TOP}

epicsEnvSet "STREAM_PROTOCOL_PATH" "../proto"

## Register all support components
dbLoadDatabase "dbd/pm1001.dbd"
pm1001_registerRecordDeviceDriver pdbname

drvAsynSerialPortConfigure "UART" "/dev/ttyAMA0"

asynSetOption("UART", 0, "baud", "9600")
asynSetOption("UART", 0, "bits", "8")
asynSetOption("UART", 0, "parity", "none")

## Load record instances
#dbLoadRecords("db/xxx.db","user=piHost")
dbLoadRecords("db/pm1001.db")

cd ${TOP}/iocBoot/${IOC}
iocInit

## Start any sequence programs
#seq sncxxx,"user=piHost"

```

마지막으로 st.cmd를 실행파일로 변경한 후 실행시킨다.

```

pi@raspberrypi ~/epics/R3.14.12.4/siteApps/pm1001/iocBoot/iocpm1001 $ chmod 755 st.cmd
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/pm1001/iocBoot/iocpm1001 $ sudo ./st.cmd
#!/../bin/linux-arm/pm1001
## You may have to change pm1001 to something else
## everywhere it appears in this file
< envPaths
epicsEnvSet("ARCH","linux-arm")
epicsEnvSet("IOC","iocpm1001")
epicsEnvSet("TOP","/home/pi/epics/R3.14.12.4/siteApps/pm1001")
epicsEnvSet("EPICS_BASE","/home/pi/epics/R3.14.12.4/base")
cd /home/pi/epics/R3.14.12.4/siteApps/pm1001
epicsEnvSet "STREAM_PROTOCOL_PATH" "../proto"

```

```
## Register all support components
dbLoadDatabase "dbd/pm1001.dbd"
pm1001_registerRecordDeviceDriver pdbbase
drvAsynSerialPortConfigure "UART" "/dev/ttyAMA0"
asynSetOption("UART", 0, "baud", "9600")
asynSetOption("UART", 0, "bits", "8")
asynSetOption("UART", 0, "parity", "none")
## Load record instances
#dbLoadRecords("db/xxx.db", "user=piHost")
dbLoadRecords("db/sensor.db")
cd /home/pi/epics/R3.14.12.4/siteApps/pm1001/iocBoot/iocpm1001
iocInit
Starting iocInit
#####
## EPICS R3.14.12.4 $Date: Mon 2013-12-16 15:51:45 -0600$
## EPICS Base built Oct 4 2014
#####
iocRun: All initialization complete
## Start any sequence programs
#seq sncxxx, "user=piHost"
```

먼저 값이 제대로 읽혀지는지 확인해 본다.

```
epics> dbpr PM1001:DUST
ASG:          DESC:          DISA: 0          DISP: 0
DISV: 1        NAME: SS:DUST   RVAL: 673   SEVR: NO_ALARM
STAT: NO_ALARM SVAL: 0        TPRO: 0        VAL: 673
```

3.5 Temperature & Humidity Sensor

DHT11

2.3에서 진행하였던 DHT11 센서 EPICS로 Integration 하여 테스트해 본다. 최종 목표는 그림9과 같이 회로를 구성하고 다음 epics db를 통해 온습도 값을 읽는것이다.

```
record(ai, "tem")
{
    field(DTYP, "DHT11")
    field(SCAN, "1 second")
    field(INP, "@1 temperature")
}

record(ai, "hum")
{
    field(DTYP, "DHT11")
    field(SCAN, "1 second")
    field(INP, "@1 humidity")
}
```

코드 작성에 앞서 테스트 할 기본 폴더 및 EPICS Application 구조를 생성한다.

```
pi@ctrlpi3 cd ../epics/R3.14.12.4/siteApps
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps $ mkdir dht11
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps $ cd dht11
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/dht11 $ makeBaseApp.pl -t ioc dht11
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/dht11 $ makeBaseApp.pl -i -t ioc dht11
```

```
Using target architecture linux-arm (only one available)
The following applications are available:
```

```

dht11
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed above.
Application name?

pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/dht11 $ ls
configure dht11App iocBoot Makefile

```

dht11App/src 폴더로 이동한 후 devDHT11.c 파일을 만들어 다음과 같이 코드를 작성한다.

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  #include <epicsExport.h>
6  #include <devSup.h>
7  #include <recSup.h>
8  #include <recGbl.h>
9  #include <dbAccess.h>
10 #include <callback.h>
11 #include <aiRecord.h>
12
13 #include <wiringPi.h>
14
15 #define MAX_TIME 85
16
17 int dht11_val[5];
18
19 typedef struct _DHT_INFO
20 {
21     CALLBACK callback;
22
23     int pin_num;
24     int pin_mode;
25
26     float val_h;
27     float val_t;
28
29     float pre_val_h;
30     float pre_val_t;
31 }DHT_INFO;
32
33 void readDHT11(DHT_INFO *dht_info);
34
35 static long ai_init_record(aiRecord *pai);
36 static long read_ai(aiRecord *pai);
37
38 static void myCallback(CALLBACK *pcallback)
39 {
40     aiRecord *precord;
41     struct rset *prset;
42
43     callbackGetUser(precord, pcallback);
44     prset = (struct rset *) (precord->rset);
45
46     dbScanLock((dbCommon*)precord);
47     (*prset->process)(precord);
48     dbScanUnlock((dbCommon*)precord);
49 }
50
51 static long ai_init_record(aiRecord *pai)

```



```

52 {
53     DHT_INFO *dht_info = malloc(sizeof(DHT_INFO));
54
55     callbackSetCallback(myCallback, &dht_info->callback);
56     callbackSetPriority(priorityLow, &dht_info->callback);
57     callbackSetUser(pai, &dht_info->callback);
58
59     if(wiringPiSetup() == -1)
60         return 1;
61
62     char *para;
63     char *sensor;
64     int pin_num = 0;
65     int mode = 0;
66
67     para = pai->inp.value.instio.string;
68
69     pin_num = atoi(strtok(para, " "));
70     sensor = strtok(NULL, " ");
71
72     if(strcmp(sensor, "humidity") == 0)
73         mode = 0;
74     else
75         mode = 1;
76
77     int i;
78     for(i=0;i<5;i++)
79         dht11_val[i] = 0;
80
81     dht_info->val_h = 0.0f;
82     dht_info->val_t = 0.0f;
83     dht_info->pre_val_h = 0.0f;
84     dht_info->pre_val_t = 0.0f;
85
86     dht_info->pin_num = pin_num;
87     dht_info->pin_mode = mode;
88
89     pai->dpvt = dht_info;
90
91     return 0;
92 }
93
94 static long read_ai(aiRecord *pai)
95 {
96     DHT_INFO *dht_info = pai->dpvt;
97
98     if(pai->pact)
99     {
100         readDHT11(dht_info);
101
102         if(dht_info->pin_mode == 0)
103             pai->val = dht_info->val_h;
104         else
105             pai->val = dht_info->val_t;
106
107         pai->udf = FALSE;
108
109         return 2;
110     }
111
112     pai->pact = TRUE;
113     callbackRequestDelayed(&dht_info->callback, pai->disv);

```

```

114
115     return 0;
116 }
117
118 struct
119 {
120     long num;
121     DEVSUPFUN    report;
122     DEVSUPFUN    init;
123     DEVSUPFUN    init_record;
124     DEVSUPFUN    get_ioint_info;
125     DEVSUPFUN    read_ai;
126     DEVSUPFUN    special_linconv;
127 } devAiDHT11Async = {
128     6,
129     NULL,
130     NULL,
131     ai_init_record,
132     NULL,
133     read_ai,
134     NULL
135 };
136
137 epicsExportAddress(dset,devAiDHT11Async);
138
139 void readDHT11(DHT_INFO *dht_info)
140 {
141     int pin = dht_info->pin_num;
142     int mode = dht_info->pin_mode;
143
144     int i=0;
145     for(i=0;i<5;i++)
146         dht11_val[i]=0;
147
148     epicsUInt8 lststate=HIGH;
149     epicsUInt8 counter=0;
150
151     pinMode(pin,OUTPUT);
152     digitalWrite(pin,LOW);
153     delay(18);
154     digitalWrite(pin,HIGH);
155     delayMicroseconds(40);
156     pinMode(pin,INPUT);
157
158     int j=0;
159     for(i=0;i<MAX_TIME;i++)
160     {
161         counter=0;
162         while(digitalRead(pin)==lststate)
163         {
164             counter++;
165             delayMicroseconds(1);
166             if(counter==255)
167                 break;
168         }
169
170         lststate=digitalRead(pin);
171         if(counter==255)
172             break;
173         // top 3 transistions are ignored
174         if((i>=4)&&(i%2==0))
175         {

```

```

176     dht11_val[j/8]<=1;
177     if(counter>16)
178     dht11_val[j/8]=1;
179     j++;
180 }
181 }
182
183 float val = 0.0f;
184 char tmp[10];
185
186 if((j>=40)&&(dht11_val[4]==((dht11_val[0]+dht11_val[1]+dht11_val[2]+dht11_val[3])& 0xFF)))
187 {
188     sprintf(tmp, "%d.%d", dht11_val[0], dht11_val[1]);
189     val = atof(tmp);
190
191     dht_info->val_h = val;
192     dht_info->pre_val_h = val;
193
194     sprintf(tmp, "%d.%d", dht11_val[2], dht11_val[3]);
195     val = atof(tmp);
196
197     dht_info->val_t = val;
198     dht_info->pre_val_t = val;
199 }
200 else
201 {
202     if(mode == 0)
203         dht_info->val_h = dht_info->pre_val_h;
204     else
205         dht_info->val_t = dht_info->pre_val_t;
206 }
207 }

```

Listing 3.2 devDHT11.c

코드 작성이 완료되면 devGPIO.dbd 파일을 만든다.

```
device(ai, INST_IO, devAiDHT11Async, "DHT11")
```

앞서 작성한 코드를 Build하기 위해 Makefile에 다음과 같이 추가한다.

```

TOP=../..

include $(TOP)/configure/CONFIG
#-----
#  ADD MACRO DEFINITIONS AFTER THIS LINE
#=====

#=====
# Build the IOC application

USR_INCLUDES += -I/home/pi/wiringPi/wiringPi
wiringPi_DIR += /home/pi/wiringPi/wiringPi /home/pi/wiringPi/devLib

PROD_IOC = dht11
# dht11.dbd will be created and installed
DBD += dht11.dbd

# dht11.dbd will be made up from these files:
dht11_DBD += base.dbd

# Include dbd files from all support applications:
#dht11_DBD += xxx.dbd

```

```

dht11_DBD += devDHT11.dbd

# Add all the support libraries needed by this IOC
#dht11_LIBS += xxx

# dht11_registerRecordDeviceDriver.cpp derives from dht11.dbd
dht11_SRCS += dht11_registerRecordDeviceDriver.cpp
dht11_SRCS += devDHT11.c

# Build the main IOC entry point on workstation OSs.
dht11_SRCS_DEFAULT += dht11Main.cpp
dht11_SRCS_vxWorks += -nil-

# Add support from base/src/vxWorks if needed
#dht11_OBJS_vxWorks += $(EPICS_BASE_BIN)/vxComLibrary

# Finally link to the EPICS Base libraries
dht11_LIBS += $(EPICS_BASE_IOC_LIBS)
dht11_LIBS += wiringPi

#=====

include $(TOP)/configure/RULES
#-----
# ADD RULES AFTER THIS LINE

```

db파일을 만들기 위해 dht11App/Db 폴더로 이동한 후 다음과 같은 gpio.db 파일을 만든다.

```

record(ai, "tem")
{
    field(DTYP, "DHT11")
    field(SCAN, "1 second")
    field(INP, "@1 temperature")
}

record(ai, "hum")
{
    field(DTYP, "DHT11")
    field(SCAN, "1 second")
    field(INP, "@1 humidity")
}

```

Makefile에 dht11.db를 추가해 준다.

```

TOP=../..
include $(TOP)/configure/CONFIG
#-----
# ADD MACRO DEFINITIONS AFTER THIS LINE

#-----
# Optimization of db files using dbst (DEFAULT: NO)
#DB_OPT = YES

#-----
# Create and install (or just install) into /db
# databases, templates, substitutions like this
#DB += xxx.db
DB += dht11.db

#-----
# If .db template is not named *.template add

```

```
# _template =
```

```
include $(TOP)/configure/RULES
```

```
#-----
```

```
# ADD RULES AFTER THIS LINE
```

마지막으로 앞서 작성한 파일들을 컴파일 하기 위해 dht11폴더로 이동한 후 make를 실행한 다.

```
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/dht11 $ make
```

make가 완료되면 bin/linux-arm 폴더에 dht11 파일과 db 폴더에 dht11.db 파일이 만들어 진 다. 이제 ioc를 실행시키기 위해 iocBoot/iocdht11 폴더로 이동 후 st.cmd파일에 dht11.db를 Load하는 코드를 추가해 준다.

```
#!../bin/linux-arm/dht11
```

```
## You may have to change dht11 to something else
## everywhere it appears in this file
```

```
< envPaths
```

```
cd ${TOP}
```

```
## Register all support components
dbLoadDatabase "dbd/dht11.dbd"
dht11_registerRecordDeviceDriver pdbname
```

```
## Load record instances
#dbLoadRecords("db/xxx.db", "user=piHost")
dbLoadRecords("db/dht11.db")
```

```
cd ${TOP}/iocBoot/${IOC}
iocInit
```

```
## Start any sequence programs
#seq sncxxx, "user=piHost"
```

마지막으로 st.cmd파일을 실행파일로 변경한 후 실행시킨다.

```
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/dht11/iocBoot/iocdht11 $ chmod 755 st.cmd
```

```
pi@ctrlpi3 ~/epics/R3.14.12.4/siteApps/dht11/iocBoot/iocdht11 $ sudo ./st.cmd
```

```
#!../bin/linux-arm/dht11
```

```
## You may have to change dht11 to something else
## everywhere it appears in this file
```

```
< envPaths
```

```
epicsEnvSet("ARCH", "linux-arm")
epicsEnvSet("IOC", "iocdht11")
epicsEnvSet("TOP", "/home/pi/epics/R3.14.12.4/siteApps/dht11")
epicsEnvSet("EPICS_BASE", "/home/pi/epics/R3.14.12.4/base")
cd /home/pi/epics/R3.14.12.4/siteApps/dht11
```

```
## Register all support components
dbLoadDatabase "dbd/dht11.dbd"
dht11_registerRecordDeviceDriver pdbname
```

```
## Load record instances
#dbLoadRecords("db/xxx.db", "user=piHost")
dbLoadRecords("db/dht11.db")
```

```
cd /home/pi/epics/R3.14.12.4/siteApps/dht11/iocBoot/iocdht11
iocInit
```

```
Starting iocInit
```

```
#####
```

```
## EPICS R3.14.12.4 $Date: Mon 2013-12-16 15:51:45 -0600$
```

```
## EPICS Base built Aug 29 2014
#####
iocRun: All initialization complete
## Start any sequence programs
#seq sncxxx,"user=piHost"
epics>
```

온도와 습도값이 제대로 읽어지면 끝!

```
epics> dbpr tem
ASG:          DESC:          DISA: 0          DISP: 0
DISV: 1       NAME: tem      RVAL: 0          SEVR: NO_ALARM
STAT: NO_ALARM SVAL: 0       TPRO: 0          VAL: 26
epics> dbpr hum
ASG:          DESC:          DISA: 0          DISP: 0
DISV: 1       NAME: hum      RVAL: 57         SEVR: NO_ALARM
STAT: NO_ALARM SVAL: 0       TPRO: 0          VAL: 57
```

DS1820

2.3에서 진행하였던 DHT11 센서 EPICS로 Integration 하여 테스트해 본다. 최종 목표는 그림 10과 같이 회로를 구성하고 다음 epics db를 통해 온도 값을 읽는것이다.

```
record(ai, "ds1820")
{
  field(DTYP, "DS1820")
  field(SCAN, "1 second")
  field(INP, "@1")
}
```

코드 작성에 앞서 테스트 할 기본 폴더 및 EPICS Application 구조를 생성한다.

```
pi@raspberrypi cd ../epics/R3.14.12.4/siteApps
pi@raspberrypi ~/epics/R3.14.12.4/siteApps $ mkdir ds1820
pi@raspberrypi ~/epics/R3.14.12.4/siteApps $ cd ds1820
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/ds1820 $ makeBaseApp.pl -t ioc ds1820
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/ds1820 $ makeBaseApp.pl -i -t ioc ds1820
```

```
Using target architecture linux-arm (only one available)
The following applications are available:
  ds1820
What application should the IOC(s) boot?
The default uses the IOC's name, even if not listed above.
Application name?
```

```
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/ds1820 $ ls
conigure ds1820App iocBoot Makefile
```

ds1820App/src 폴더로 이동한 후 devDS1820.c 파일을 만들어 다음과 같이 코드를 작성한다.

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <stdint.h>
5
6 #include <epicsExport.h>
7 #include <devSup.h>
8 #include <recSup.h>
9 #include <recGbl.h>
10 #include <dbAccess.h>
```

```

11 #include <callback.h>
12 #include <aiRecord.h>
13
14 #include <wiringPi.h>
15
16 typedef struct _DS1820_INFO
17 {
18     CALLBACK callback;
19
20     int pin_num;
21
22     float temper;
23     float pre_temper;
24 }DS1820_INFO;
25
26 void ds1820_read(DS1820_INFO *ds1820_info);
27 int onewire_reset(int pin);
28 void onewire_write(int pin, uint8_t data);
29 void onewire_write_bit(int pin, int bit);
30 uint8_t onewire_read(int pin);
31 int onewire_read_bit(int pin);
32 uint8_t crc_read();
33 uint8_t crc_cal(uint8_t crc, uint8_t data);
34
35 static long ai_init_record(aiRecord *pai);
36 static long read_ai(aiRecord *pai);
37
38 static void myCallback(CALLBACK *pcallback)
39 {
40     aiRecord *precord;
41     struct rset *prset;
42
43     callbackGetUser(precord, pcallback);
44     prset = (struct rset *) (precord->rset);
45
46     dbScanLock((dbCommon*)precord);
47     (*prset->process)(precord);
48     dbScanUnlock((dbCommon*)precord);
49 }
50
51 static long ai_init_record(aiRecord *pai)
52 {
53     DS1820_INFO *ds1820_info = malloc(sizeof(DS1820_INFO));
54
55     callbackSetCallback(myCallback, &ds1820_info->callback);
56     callbackSetPriority(priorityLow, &ds1820_info->callback);
57     callbackSetUser(pai, &ds1820_info->callback);
58
59     if(wiringPiSetup() == -1)
60         return 1;
61
62     char *para;
63     int pin_num = 0;
64
65     para = pai->inp.value.instio.string;
66
67     pin_num = atoi(para);
68
69     ds1820_info->temper = 0.0f;
70     ds1820_info->pre_temper = 0.0f;
71
72     ds1820_info->pin_num = pin_num;

```

```

73     pai->dpvt = ds1820_info;
74
75     return 0;
76 }
77
78 static long read_ai(aiRecord *pai)
79 {
80     DS1820_INFO *ds1820_info = pai->dpvt;
81
82     if(pai->pact)
83     {
84         ds1820_read(ds1820_info);
85
86         pai->val = ds1820_info->temper;
87
88         pai->udf = FALSE;
89
90         return 2;
91     }
92
93     pai->pact = TRUE;
94     callbackRequestDelayed(&ds1820_info->callback, pai->disv);
95
96     return 0;
97 }
98
99 struct
100 {
101     long num;
102     DEVSUPFUN    report;
103     DEVSUPFUN    init;
104     DEVSUPFUN    init_record;
105     DEVSUPFUN    get_ioint_info;
106     DEVSUPFUN    read_ai;
107     DEVSUPFUN    special_linconv;
108 } devAiDS1820Async = {
109     6,
110     NULL,
111     NULL,
112     ai_init_record,
113     NULL,
114     read_ai,
115     NULL
116 };
117
118 epicsExportAddress(dset,devAiDS1820Async);
119
120 void ds1820_read(DS1820_INFO* ds1820_info)
121 {
122     uint8_t busy = 1;
123     int pin = ds1820_info->pin_num;
124
125     onewire_reset(pin);
126     onewire_write(pin, 0xCC);
127     onewire_write(pin, 0x44);
128
129     delay(750);
130     while(busy == 0)
131     {
132         busy = onewire_read(pin);
133         printf("busy: %d\n", busy);
134     }

```



```

135     }
136
137     onewire_reset(pin);
138     onewire_write(pin, 0xCC);
139     onewire_write(pin, 0xBE);
140
141     uint8_t lsb, msb, th, tl, reserved1, reserved2, count_remain, count_per_c, crc;
142     float real_temp = 0.0f;
143     float pre_real_temp = ds1820_info->temper;
144     signed char temp_read = 0;
145
146     lsb = onewire_read(pin);
147     msb = onewire_read(pin);
148     th = onewire_read(pin);
149     tl = onewire_read(pin);
150     reserved1 = onewire_read(pin);
151     reserved2 = onewire_read(pin);
152     count_remain = onewire_read(pin);
153     count_per_c = onewire_read(pin);
154     crc = onewire_read(pin);
155
156     uint8_t data[] = {lsb, msb, th, tl, reserved1, reserved2, count_remain, count_per_c};
157
158     onewire_reset(pin);
159
160     if(crc_read(data) == crc)
161     {
162         temp_read = (signed char)(lsb>>1);
163
164         if(msb == 255)
165             temp_read = temp_read | 0x80;
166
167         real_temp = (float)temp_read + 0.85f - (float)count_remain/(float)count_per_c;
168         real_temp = (int)(real_temp * 10) / 10.0f;
169
170         ds1820_info->temper = real_temp;
171         ds1820_info->pre_temper = real_temp;
172     }
173     else
174         ds1820_info->temper = pre_real_temp;
175
176 }
177
178 int onewire_reset(int pin)
179 {
180     int result;
181
182     pinMode(pin, OUTPUT);
183
184     digitalWrite(pin, LOW);
185     delayMicroseconds(480);
186
187     pinMode(pin, INPUT);
188     delayMicroseconds(70);
189
190     result = digitalRead(pin);
191
192     delayMicroseconds(410);
193
194     return result;
195 }
196

```

```

197 void onewire_write(int pin, uint8_t data)
198 {
199     int loop;
200
201     for(loop=0; loop<8; loop++)
202     {
203         onewire_write_bit(pin, data & 0x01);
204
205         data >>= 1;
206     }
207 }
208
209 void onewire_write_bit(int pin, int bit)
210 {
211     pinMode(pin, OUTPUT);
212
213     if(bit)
214     {
215         digitalWrite(pin, LOW);
216         delayMicroseconds(6);
217         digitalWrite(pin, HIGH);
218         delayMicroseconds(64);
219     }
220     else
221     {
222         digitalWrite(pin, LOW);
223         delayMicroseconds(60);
224         digitalWrite(pin, HIGH);
225         delayMicroseconds(10);
226     }
227
228 }
229
230 uint8_t onewire_read(int pin)
231 {
232     int loop, result=0;
233
234     for(loop=0; loop<8; loop++)
235     {
236         result >>= 1;
237
238         if(owread_bit(pin))
239             result |= 0x80;
240     }
241
242     return result;
243 }
244
245 int onewire_read_bit(int pin)
246 {
247     int result;
248
249     pinMode(pin, OUTPUT);
250
251     digitalWrite(pin, LOW);
252     delayMicroseconds(6);
253
254     pinMode(pin, INPUT);
255     delayMicroseconds(9);
256
257     result = digitalRead(pin) & 0x01;
258     delayMicroseconds(55);

```

```

259     return result;
260 }
261
262
263 uint8_t crc_read(uint8_t *data)
264 {
265     uint8_t i, crc;
266
267     crc = 0x00;
268
269     for(i=0; i<8; i++)
270         crc = crc_cal(crc, data[i]);
271
272     return crc;
273 }
274
275 uint8_t crc_cal(uint8_t crc, uint8_t data)
276 {
277     int j;
278     for(j=0; j<8; j++) {
279         if ((data & 0x01) ^ (crc & 0x01)) {
280             // DATA ^ LSB CRC = 1
281             crc = crc>>1;
282             // Set the MSB to 1
283             crc = crc | 0x80;
284             // Check bit 3
285             if (crc & 0x04) {
286                 crc = crc & 0xFB; // Bit 3 is set, so clear it
287             } else {
288                 crc = crc | 0x04; // Bit 3 is clear, so set it
289             }
290             // Check bit 4
291             if (crc & 0x08) {
292                 crc = crc & 0xF7; // Bit 4 is set, so clear it
293             } else {
294                 crc = crc | 0x08; // Bit 4 is clear, so set it
295             }
296             } else {
297                 // DATA ^ LSB CRC = 0
298                 crc = crc>>1;
299                 // clear MSB
300                 crc = crc & 0x7F;
301                 // No need to check bits, with DATA ^ LSB CRC = 0, they will remain unchanged
302             }
303             data = data>>1;
304         }
305
306     return crc;
307 }

```

Listing 3.3 devDS1820.c

코드 작성이 완료되면 devDS1820.dbd 파일을 만든다.

```
device(ai, INST_IO, devAiDS1820Async, "DS1820")
```

앞서 작성한 코드를 Build하기 위해 Makefile에 다음과 같이 추가한다.

```
TOP=../../
```

```

include $(TOP)/configure/CONFIG
#-----
# ADD MACRO DEFINITIONS AFTER THIS LINE

```

```

#####

#####
# Build the IOC application

USR_INCLUDES += -I/home/pi/wiringPi/wiringPi
wiringPi_DIR += /home/pi/wiringPi/wiringPi /home/pi/wiringPi/devLib

PROD_IOC = ds1820
# dht11.dbd will be created and installed
DBD += ds1820.dbd

# dht11.dbd will be made up from these files:
ds1820_DBD += base.dbd

# Include dbd files from all support applications:
#ds1820_DBD += xxx.dbd
ds1820_DBD += devDS1820.dbd

# Add all the support libraries needed by this IOC
#ds1820_LIBS += xxx

# ds1820_registerRecordDeviceDriver.cpp derives from ds1820.dbd
ds1820_SRCS += ds1820_registerRecordDeviceDriver.cpp
ds1820_SRCS += devDS1820.c

# Build the main IOC entry point on workstation OSs.
ds1820_SRCS_DEFAULT += ds1820Main.cpp
ds1820_SRCS_vxWorks += -nil-

# Add support from base/src/vxWorks if needed
#ds1820_OBJS_vxWorks += $(EPICS_BASE_BIN)/vxComLibrary

# Finally link to the EPICS Base libraries
ds1820_LIBS += $(EPICS_BASE_IOC_LIBS)
ds1820_LIBS += wiringPi

#####

include $(TOP)/configure/RULES
#-----
# ADD RULES AFTER THIS LINE

```

db파일을 만들기 위해 ds1820App/Db 폴더로 이동한 후 다음과 같은 ds1820.db 파일을 만든다.

```

record(ai, "ds1820")
{
    field(DTYP, "DS1820")
    field(SCAN, "1 second")
    field(INP, "@1")
}

```

Makefile에 ds1820.db를 추가해 준다.

```

TOP=../..
include $(TOP)/configure/CONFIG
#-----
# ADD MACRO DEFINITIONS AFTER THIS LINE

#-----
# Optimization of db files using dbst (DEFAULT: NO)

```

```
#DB_OPT = YES

#-----
# Create and install (or just install) into /db
# databases, templates, substitutions like this
#DB += xxx.db
DB += ds1820.db

#-----
# If .db template is not named *.template add
# _template =

include $(TOP)/configure/RULES
#-----
# ADD RULES AFTER THIS LINE
```

마지막으로 앞서 작성한 파일들을 컴파일 하기 위해 ds1820폴더로 이동한 후 make를 실행한 다.

```
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/ds1820 $ make
```

make가 완료되면 bin/linux-arm 폴더에 ds1820 파일과 db 폴더에 ds1820.db 파일이 만들어 진 다. 이제 ioc를 실행시키기 위해 iocBoot/iocds1820 폴더로 이동 후 st.cmd파일에 ds1820.db를 Load하는 코드를 추가해 준다.

```
#!../bin/linux-arm/ds1820

## You may have to change dht11 to something else
## everywhere it appears in this file

< envPaths

cd ${TOP}

## Register all support components
dbLoadDatabase "dbd/ds1820.dbd"
ds1820_registerRecordDeviceDriver pdbname

## Load record instances
#dbLoadRecords("db/xxx.db","user=piHost")
dbLoadRecords("db/ds1820.db")

cd ${TOP}/iocBoot/${IOC}
iocInit

## Start any sequence programs
#seq sncxxx,"user=piHost"
```

마지막으로 st.cmd파일을 실행파일로 변경한 후 실행시킨다.

```
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/ds1820/iocBoot/iocds1820 $ chmod 755 st.cmd
pi@raspberrypi ~/epics/R3.14.12.4/siteApps/ds1820/iocBoot/iocds1820 $ sudo ./st.cmd
#!../bin/linux-arm/ds1820
## You may have to change ds1820 to something else
## everywhere it appears in this file
< envPaths
epicsEnvSet("ARCH","linux-arm")
epicsEnvSet("IOC","iocdht11")
epicsEnvSet("TOP","/home/pi/epics/R3.14.12.4/siteApps/ds1820")
epicsEnvSet("EPICS_BASE","/home/pi/epics/R3.14.12.4/base")
cd /home/pi/epics/R3.14.12.4/siteApps/ds1820
## Register all support components
```

```

dbLoadDatabase "dbd/ds1820.dbd"
dht11_registerRecordDeviceDriver pdbbase
## Load record instances
#dbLoadRecords("db/xxx.db", "user=piHost")
dbLoadRecords("db/ds1820.db")
cd /home/pi/epics/R3.14.12.4/siteApps/ds1820/iocBoot/iocds1820
iocInit
Starting iocInit
#####
## EPICS R3.14.12.4 $Date: Mon 2013-12-16 15:51:45 -0600$
## EPICS Base built Aug 29 2014
#####
iocRun: All initialization complete
## Start any sequence programs
#seq sncxxx, "user=piHost"
epics>

```

온도와 습도값이 제대로 읽어지면 끝!

```

epics> dbpr ds1820
ASG:          DESC:          DISA: 0          DISP: 0
DISV: 1       NAME: tem      RVAL: 0          SEVR: NO_ALARM
STAT: NO_ALARM SVAL: 0       TPRO: 0          VAL: 26

```

Bibliography

- [1] PIR Motion Sensor Web Document, 2015.
http://www.dfrobot.com/wiki/index.php/PIR_Motion_Sensor_V1.0_SKU:SEN0171#Document (accessed Feb 23, 2015).
- [2] PM1001 Dust Sensor Manual, 2015.
http://diwellshop.cafe24.com/web/DATASHEET/01_subtitle/1_Sensor/4_Gas_Sensor/Dust/PM1001/PM1001UserManual.pdf (accessed Feb 23, 2015).
- [3] DHT11 Humidity & Temperature Sensor Manual, 2015.
<http://www.micro4you.com/files/sensor/DHT11.pdf> (accessed Feb 23, 2015).
- [4] DHT22 Humidity & Temperature Sensor Manual, 2015.
http://www.dfrobot.com/image/data/SEN0137/AM2302_manual.pdf (accessed Feb 23, 2015).
- [5] DS1820 Temperature Sensor Manual, 2015.
<http://www.systronix.com/Resource/ds1820.pdf> (accessed Feb 23, 2015).
- [6] L298 Dual H-Bridge Motor Driver, 2015.
http://gobotics.com/datasheets/L298_Data_Sheet.pdf (accessed Feb 23, 2015).
- [7] MD5-DH14 Motor Driver Manual, 2015.
http://download.autonics.com/upload/data/1420685639_MD5-HD14_EN_EP-KE-14-0008C_20141111.pdf (accessed Feb 23, 2015).
- [8] Wiring Pi Web Page, 2015.
<http://wiringpi.com/> (accessed Feb 23, 2015).
- [9] Pi Camera Web Page, 2015.
<http://www.raspberrypi.org/documentation/usage/camera/README.md> (accessed Feb 23, 2015).