



프로세서응용종합설계

프로젝트-디지털 온도계

담당교수 : 황신환 교수님

제출일자 : 20.12.13

학 과 : 전자공학과

학 년 : 3학년

이 름 : 배준성

학 번 :



한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

목 차

- | | |
|-------------------|--------------------------|
| 1 프로젝트 목표 | 6 드라이버·라이브러리
설치 및 분석 |
| 1.1 프로젝트 목표 및 산출물 | |
| 2 개발 절차 | 6.1 Raspberry Pi 4 |
| 3 필요 리소스 분석 | 6.2 DHT11 |
| 3.1 하드웨어 | 6.3 SSD1306 |
| 3.2 소프트웨어 | 6.4 Pillow |
| 4 근거리 통신 분석 | 6.5 한글 글꼴 |
| 4.1 I2 | 7 온도계 제작 및 시연 |
| 4.2 SPI | 7.1 부품 테스트 및
라이브러리 분석 |
| 4.3 UART | 7.2 로고출력 |
| 5 사용 부품 분석 | 7.3 온도측정 |
| 5.1 DHT11 | 7.4 디스플레이 출력 |
| 5.2 SSD1306 | 7.5 종료화면 출력 |
| | 8 하드웨어 연결구조 |
| | 9 최종 코드 |

프로젝트 목표

1.1. 프로젝트 목표 및 산출물

본 프로젝트는 Raspberry Pi 4, 0.96" Display SSD1306 그리고 온·습도 측정센서 DHT11을 이용하여 디지털 온도계를 제작하고 시연하는 것을 목표로 한다. 또한 제작 과정을 통하여 각 부품의 물리적 특성을 학습하고 그들을 동작 시키기 위하여 필요한 각종 드라이버와 라이브러리를 분석하고 학습하도록 한다. 다양한 부품들을 합성하여 동작 시키기 위해서는 근거리 통신방식 또한 필요하므로 각종 근거리 통신 방식을 분석하고 특히 I2C통신 방식을 이용하여 OLED 디스플레이와 Raspberry Pi 보드를 연결한다.

프로젝트 산출물의 경우 시작로고 출력, 온도, 습도 측정하고 OLED 디스플레이를 통해서 출력, 인터럽트를 이용하여 사용 종료 등의 기능을 가진 디지털 온습도계를 만드는 것을 목표로 한다.

개발 절차

1. 필요 리소스 분석

디지털 온도계 제작에 필요한 하드웨어 및 소프트웨어 파악

2. 하드웨어 사양 분석

주요 하드웨어 부품의 성능, 규격 파악

3. 하드웨어 구조 설계

하드웨어 부품의 연결 구조 설계

4. 드라이버 및 라이브러리 분석

하드웨어 부품 사용에 필요한 드라이버 및 라이브러리 설치 및 분석

5. 소프트웨어 설계

구현에 필요한 소프트웨어 설계

필요 리소스 분석

3.1. 하드웨어

- Raspberry Pi 4
초소형 컴퓨터
- DHT11
온도센서
- SSD1306
OLED 디스플레이

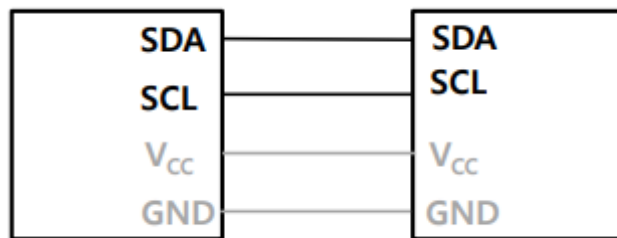
3.2. 소프트웨어

- Adafruit_DHT
온도센서 라이브러리
- Adafruit_SSD1306
디스플레이 라이브러리
- PIL
이미지 편집 라이브러리
- time
시간관련 라이브러리

근거리 통신 분석

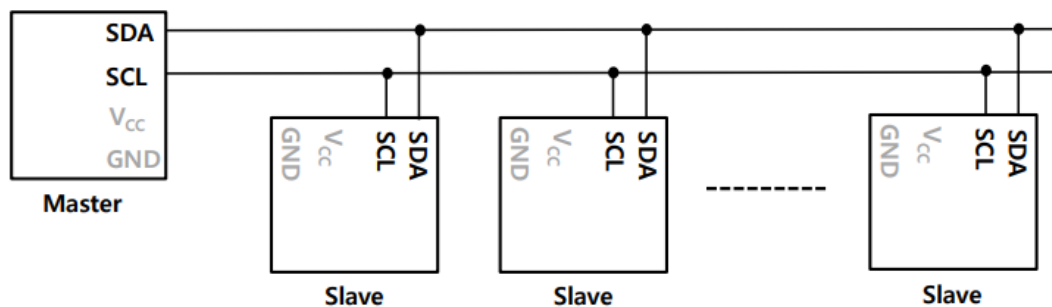
4.1. I2C

- Inter Integrated circuit
- 2선을 이용하는 근거리 통신 방식
SDA : 데이터 선
SCL : 클럭 신호선



Vcc, GND 는 전원

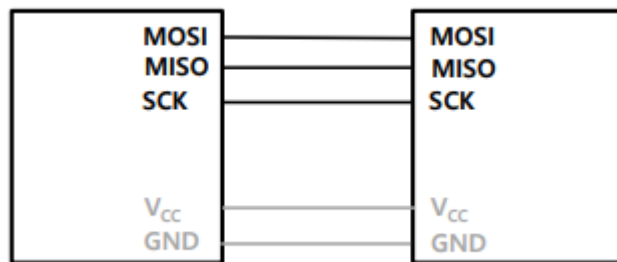
- 일대다 통신 가능
- Master, slave 구조



- 마스터가 슬레이브에게 입력, 출력을 요청하여야 슬레이브가 응답을 한다.
- 따라서 소자를 구분하기 위해 주소값을 메시지보다 먼저 전송한다.

4.2. SPI

- Serial Peripheral Interface
- 3선을 이용하는 근거리 통신 방식
MOSI(Master Out Slave In) : 데이터 선
MISO(Master In Slave Out) : 데이터 선
SCK : 클럭 신호 전송용 선



- 일대다 통신 가능
- Chip Selection 방식 이용, CS핀을 이용하여 사용할 소자를 선택

4.3. UART

- Universal Asynchronous Receiver Transmitter
- 2개의 선 이용
RX : 데이터 수신
TX : 데이터 송신
RT와 TX교차 연결
- Baud rate 설정(초당 전송 신호) – 비동기식으로 두 프로세서 간의 속도를 맞춰주어야 된다.

사용 부품 분석

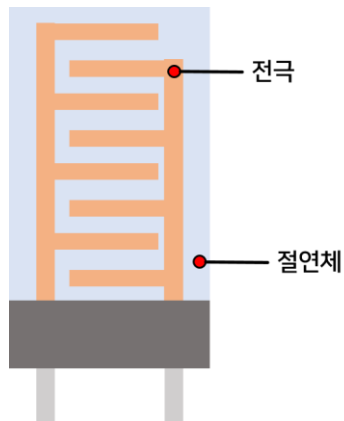
5.1. Raspberry Pi 4

- Raspberry Pi 4
Cortex_A72 CPU 기반 BCM2711 SoC 탑재
이더넷, 무선랜, 블루투스5.0 탑재
I2C, SPI, UART통신 지원
5.0V, 3.3V전원 핀 사용가능

5.2. DHT11

- DHT11
작동전압: 3.3V~5V
측정 시 작동 전류: 0.3mA
대기 시 작동 전류: 60uA
- 온도 사양
분해능: 1 °C
정확도: ± 2 °C
측정범위: 0°C ~ 50 °C
- 습도 사양
분해능: 1% RH
정확도: ± 2 % RH
측정범위: 20% ~ 90% RH (25°C)

- 저항식 습도센서



공기와 닿는 부분을 가지고 있으며 센서 외부의 작은 구멍들을 통해서 통풍되는 구조이다. 저항식 습도센서로서 두 전극 사이가 연결되어 있지 않다. 공기중의 수분을 통해서 미세하게 전류가 흐르는 방식으로 습도를 측정하며 공기중의 습도에 따라 전류가 변화한다. 이를 통해서 변화된 저항값으로 습도를 측정한다.

5.3. SSD1306

- SSD1306

해상도: 128 X 64

동작전압: 1.65 ~ 3.3V

밝기: 256단계 제어

저장장치: 128 X 64 bit SRAM 디스플레이 버퍼

인터페이스: I2C

핀 기능: VCC: 3.3V 전원 인가

GND: 접지

SCL: 클럭 신호 제어

SDA: 데이터 전송 채널

(수업에서 제공하는 부품을 수령하는 것이 제한되어 같은 이름의 부품을 구매해서 사용하였음.)

드라이버 · 라이브러리 설치 및 분석

6.1. I2C

- I2C 사용 세팅

```
LXTerminal
File Edit Tabs Help
root@raspberrypi:~# sudo raspi-config
```

```
LXTerminal
File Edit Tabs Help
Raspberry Pi 4 Model B Rev 1.2

Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the 'pi' user
2 Network Options       Configure network settings
3 Boot Options          Configure options for start-up
4 Localisation Options  Set up language and regional settings to match your
5 Interfacing Options   Configure connections to peripherals
6 Overclock             Configure overclocking for your Pi
7 Advanced Options      Configure advanced settings
8 Update                Update this tool to the latest version
9 About raspi-config    Information about this configuration tool
```

```
LXTerminal
File Edit Tabs Help

Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using
P3 VNC         Enable/Disable graphical remote access to your Pi using Rea
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial conn
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins
```

```
The ARM I2C interface is enabled
```

- I2C 연결 확인

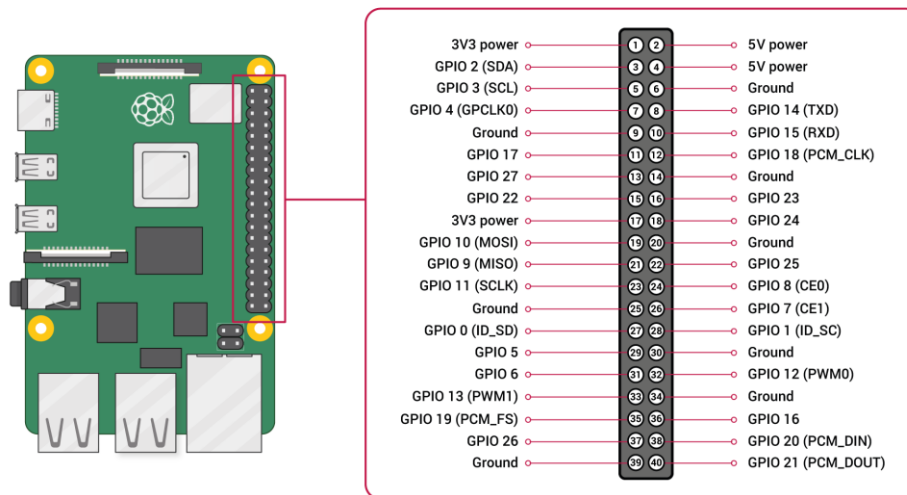
```
pi@raspberrypi: ~
File Edit Tabs Help

pi@raspberrypi:~ $ sudo apt-get install i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
i2c-tools is already the newest version (4.1-1).
i2c-tools set to manually installed.
The following packages were automatically installed and are no longer required:
  libxiv2-14 libgfortran3 libgmime-2.6-0 libncurses5 libssl1.0.2 uuid-dev
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

```
pi@raspberrypi:~ $ ls /dev/*i2c*
/dev/i2c-1
```

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

I2C핀에 데이터 선이 제대로 연결되지 않음



- I2C

- Data: (GPIO2); Clock (GPIO3)
- EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)

GPIO2번, GPIO3번핀을 데이터선과 클럭 신호선으로 사용(3번 5번 핀)

- 다시 I2C 연결 확인

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  3c  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

I2C가 제대로 연결된 것을 확인

```
pi@raspberrypi:~ $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  3c  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi:~ $ git clone https://github.com/iliapenev/ssd1306_i2c.git
Cloning into 'ssd1306_i2c'...
remote: Enumerating objects: 7, done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 7
Unpacking objects: 100% (7/7), done.
```

SSD1306의 I2C통신 라이브러리까지 설치 완료

6.2. DHT11

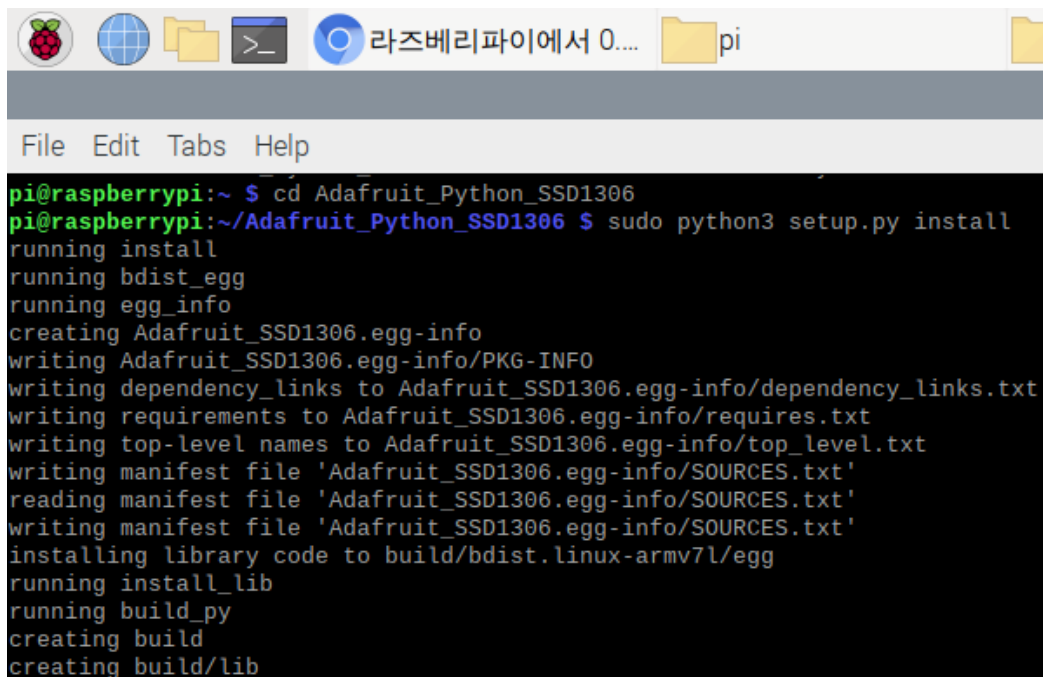
- Pip 명령을 이용하여 라이브러리 설치
`sudo pip3 install Adafruit_D`
- `Adafruit_DHT.read_retry(sensor, pin)`명령어를 이용하여 온도와 습도 값을 측정하여 저장할 수 있다.

6.3. SSD1306

- SSD1306 라이브러리 설치

```
pi@raspberrypi:~ $ git clone https://github.com/adafruit/Adafruit_Python_SSD1306
.git
Cloning into 'Adafruit_Python_SSD1306'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 137 (delta 0), reused 1 (delta 0), pack-reused 134
Receiving objects: 100% (137/137), 43.42 KiB | 230.00 KiB/s, done.
Resolving deltas: 100% (69/69), done.
```

깃허브로부터 Adafruit_Python_SSD1306 라이브러리 셋업 파일을 복사



```
pi@raspberrypi:~ $ cd Adafruit_Python_SSD1306
pi@raspberrypi:~/Adafruit_Python_SSD1306 $ sudo python3 setup.py install
running install
running bdist_egg
running egg_info
creating Adafruit_SSD1306.egg-info
writing Adafruit_SSD1306.egg-info/PKG-INFO
writing dependency_links to Adafruit_SSD1306.egg-info/dependency_links.txt
writing requirements to Adafruit_SSD1306.egg-info/requirements.txt
writing top-level names to Adafruit_SSD1306.egg-info/top_level.txt
writing manifest file 'Adafruit_SSD1306.egg-info/SOURCES.txt'
reading manifest file 'Adafruit_SSD1306.egg-info/SOURCES.txt'
writing manifest file 'Adafruit_SSD1306.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-armv7l/egg
running install_lib
running build_py
creating build
creating build/lib
```

복사 디렉토리로 이동 후 설치

- 설치 완료

```
Installed /usr/local/lib/python3.7/dist-packages/Adafruit_PureIO-1.1.8-py3.7.egg
Searching for spidev==3.4
Best match: spidev 3.4
Adding spidev 3.4 to easy-install.pth file

Using /usr/lib/python3/dist-packages
Finished processing dependencies for Adafruit-SSD1306==1.6.2
```

6.4. Pillow

- 로고 이미지 출력을 위한 이미지 프로세싱 라이브러리 설치

```
Finished processing dependencies for Adafruit-SSD1306==1.6.2
pi@raspberrypi:~/Adafruit_Python_SSD1306 $ pip3 install Pillow
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: Pillow in /usr/lib/python3/dist-packages (5.4.1)
```

6.5. 한글 글꼴 설치

- 한글 출력 위한 한글 글꼴 설치

```
pi@raspberrypi:~ $ sudo apt install fonts-nanum
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libexiv2-14 libgfortran3 libgmime-2.6-0 libncurses5 libssl1.0.2 uuid-dev
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  fonts-nanum
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 9,606 kB of archives.
After this operation, 29.5 MB of additional disk space will be used.
Get:1 http://ftp.kaist.ac.kr/raspbian/raspbian buster/main armhf fonts-nanum all
  20180306-1 [9,606 kB]
Fetched 9,606 kB in 4s (2,147 kB/s)
Selecting previously unselected package fonts-nanum.
(Reading database ... 162175 files and directories currently installed.)
Preparing to unpack .../fonts-nanum_20180306-1_all.deb ...
Unpacking fonts-nanum (20180306-1) ...
Setting up fonts-nanum (20180306-1) ...
Processing triggers for fontconfig (2.13.1-2) ...
```

온도계 제작 및 시연

7.1. 부품테스트 및 라이브러리 분석(테스트 코드 실행)

■ DHT11 테스트 코드

```
import time
import Adafruit_DHT

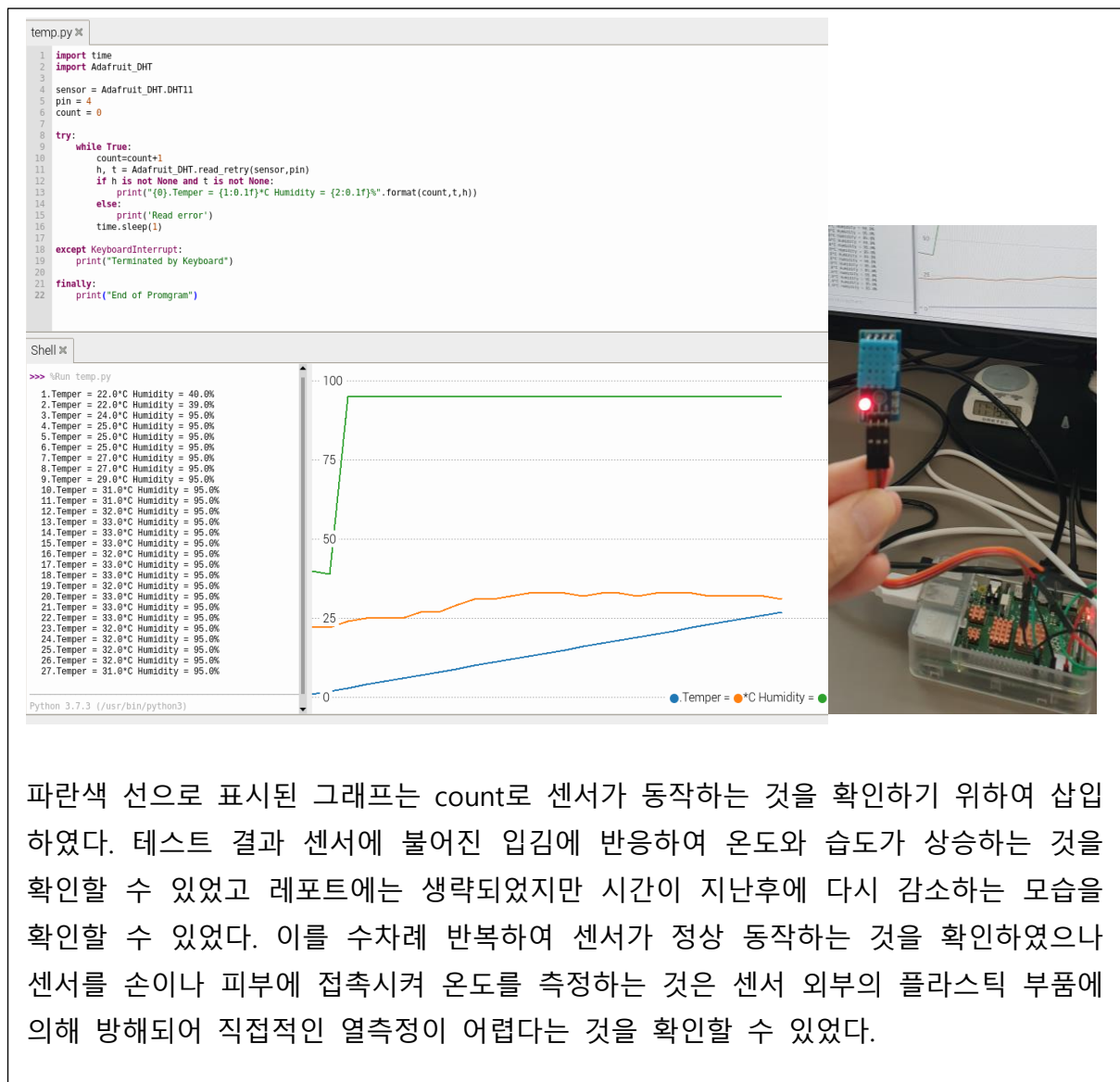
#기본 세팅
sensor = Adafruit_DHT.DHT11
pin = 4
count = 0

try:
    while True:
        count=count+1
        h, t = Adafruit_DHT.read_retry(sensor,pin) #온도 습도 값 가져옴
        if h is not None and t is not None:
            print("{0}.Temper= {1:0.1f}*C Humidity= {2:0.1f}%".format(count,t,h))#출력
        else:
            print('Read error')
            time.sleep(1)

except KeyboardInterrupt:
    print("Terminated by Keyboard")

finally:
    print("End of Promgram")
```

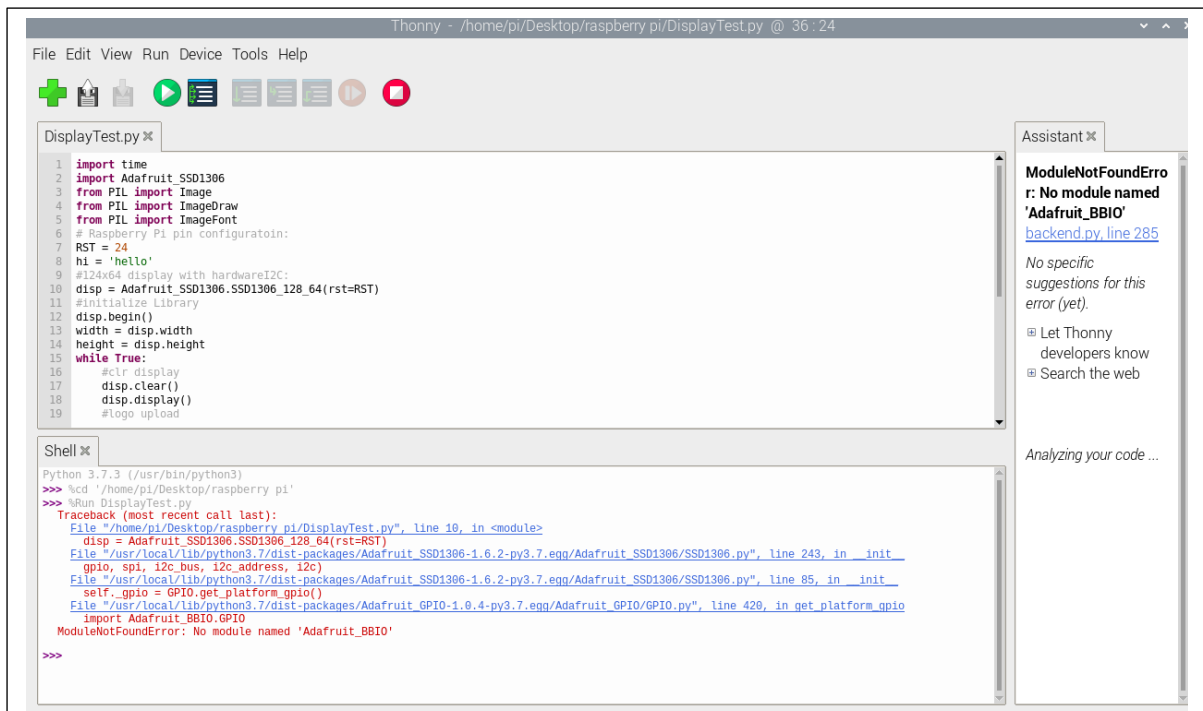

■ DHT11 테스트 결과



■ SDH1306 테스트 코드

```
import time
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
# Raspberry Pi pin configuratoin:
RST = 24
hi = 'hello'
#124x64 display with hardwareI2C:
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)
#initialize Library
disp.begin()
width = disp.width
height = disp.height
while True:
    #clr display
    disp.clear()
    disp.display()
    #logo upload
    image=Image.open('/home/pi/Desktop/logo.png').resize((width,height),Image.ANTIALIAS).convert('1')
    disp.image(image)
    disp.display()
    time.sleep(1)
    top=10
    font=ImageFont.truetype('Pillow/Tests/fonts/FreeMono.ttf',20)
    for i in range(40,-10,-10):
        #화면크기 빈 이미지생성
        image=Image.new('1',(width,height))
        draw=ImageDraw.Draw(image)
        #이미지위에 텍스트 출력
        draw.text((0,0),hi,font=font,fill=255)
        draw.text((i,top+10),'World',font=font,fill=255)
        #이미지 OLED로 출력
        disp.image(image)
        disp.display()
        time.sleep(0.3)
```

■ SDH1306 테스트 중 오류 발생



```
Thonny - /home/pi/Desktop/raspberry pi/DisplayTest.py @ 36.24
File Edit View Run Device Tools Help

1 import time
2 import Adafruit_SSD1306
3 from PIL import Image
4 from PIL import ImageDraw
5 from PIL import ImageFont
6 # Raspberry Pi pin configuration:
7 RST = 24
8 hi = 'hello'
9 #128x64 display with hardware I2C:
10 disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)
11 #Initialize library
12 disp.begin()
13 width = disp.width
14 height = disp.height
15 while True:
16     #clear display
17     disp.clear()
18     disp.display()
19     #logo upload

Shell
Python 3.7.3 (/usr/bin/python3)
>>> %cd '/home/pi/Desktop/raspberry pi'
>>> %Run DisplayTest.py
Traceback (most recent call last):
  File "/home/pi/Desktop/raspberry pi/DisplayTest.py", line 10, in <module>
    disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)
  File "/usr/local/lib/python3.7/dist-packages/Adafruit_SSD1306-1.6.2-py3.7.egg/Adafruit_SSD1306/SSD1306.py", line 243, in __init__
    gpio, spi, i2c_bus, i2c_address, i2c)
  File "/usr/local/lib/python3.7/dist-packages/Adafruit_SSD1306-1.6.2-py3.7.egg/Adafruit_SSD1306/SSD1306.py", line 85, in __init__
    self._gpio = GPIO.get_platform_gpio()
  File "/usr/local/lib/python3.7/dist-packages/Adafruit_GPIO-1.0.4-py3.7.egg/Adafruit_GPIO/GPIO.py", line 420, in get_platform_gpio
    import Adafruit_BBIO.GPIO
ModuleNotFoundError: No module named 'Adafruit_BBIO'

>>>
```

Assistant

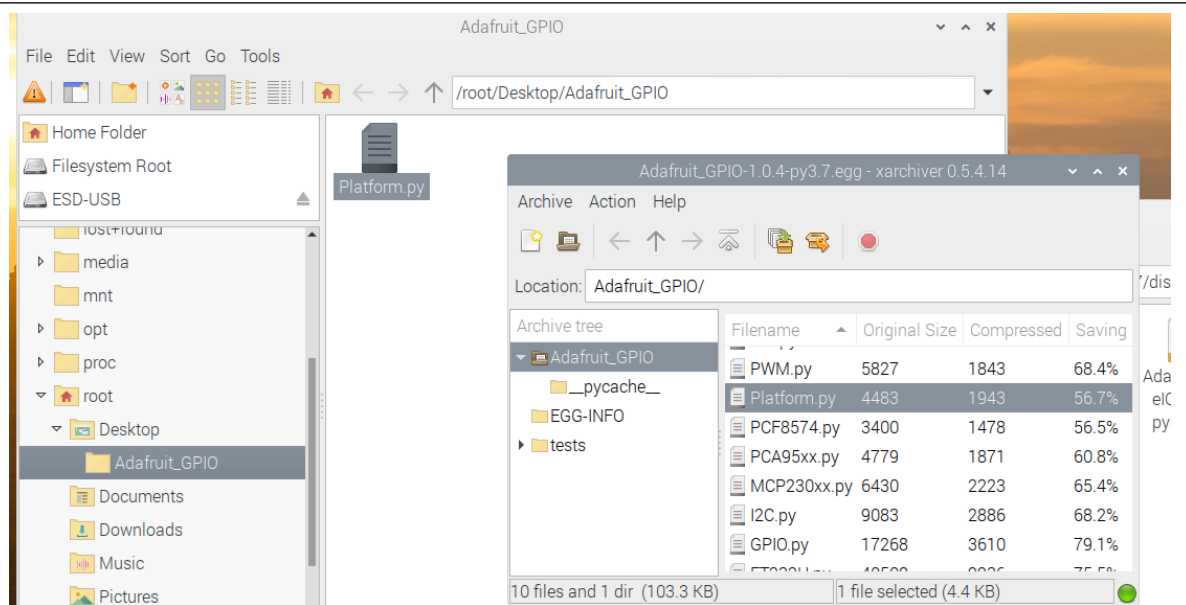
ModuleNotFoundError: No module named 'Adafruit_BBIO'
[backend.py, line 285](#)

No specific suggestions for this error (yet).

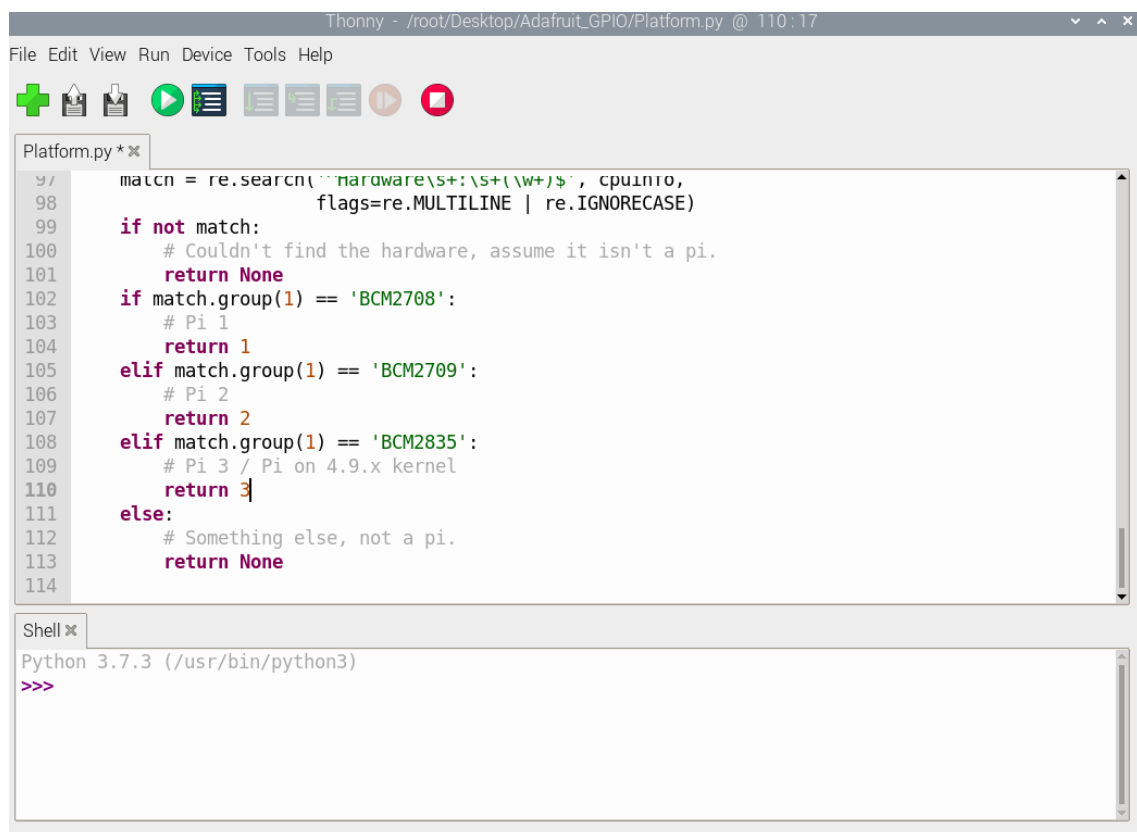
- Let Thonny developers know
- Search the web

Analyzing your code ...

소자를 테스트하는 도중 오류가 발생하였는데 오류를 분석해보니 코드상의 오류가 아닌 라이브러리 내부에서 오류가 발생 듯하였다. 라즈베리 파이로 인식되어야 하는 보드가 비글본 블랙이라는 다른 보드로 인식되어 BBIO라는 모듈을 불러오게 된 것으로 보였다. 이와 관련된 정보를 검색해보던 도중에 adafruit라이브러리 정보를 확인해 보았는데 라즈베리 파이 3까지 지원하는 라이브러리임을 확인하였다. 이것을 해결하기 위해서 라즈베리 파이 루트 계정을 이용하여 로컬 라이브러리의 플랫폼을 판별하는 부분을 수정해 주었는데 이 과정 중에서 라이브러리를 자세히 분석해 볼 수 있었다. Adafruit는 라즈베리파이 뿐만 아니라 아두이노, 위의 비글본 블랙 등 다양한 보드를 지원하기 때문에 이러한 오류가 발생하였고 비교적 간단한 수정 후에 라이브러리가 정상적으로 동작 할 수 있었다.



오류를 분석하여 라이브러리의 platform.py 파일까지 거슬러 올라갔다.



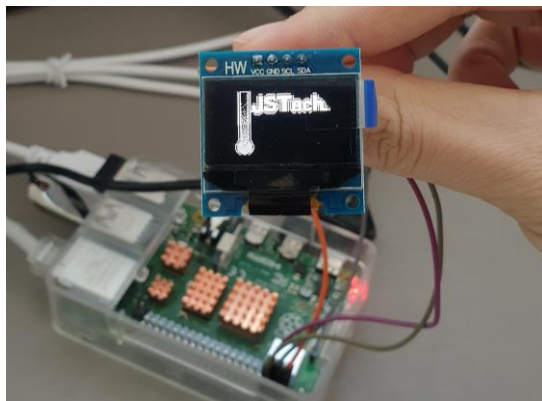
문제가 되는 부분을 분석해보면 라즈베리파이 3의 칩 까지만 제대로 분류하고 있고 우리가 사용하는 4의 칩에 대한 분류가 없다. 따라서 이부분을 수정하여 강제로 라즈베리파이 3로 인식되도록 만들어 오류를 해결하였다.

```
Thonny - /root/Desktop/Adafruit_GPIO/Platform.py @ 111:1
File Edit View Run Device Tools Help

Platform.py x
100 # Couldn't find the hardware, assume it isn't a pi.
101 return None
102 if match.group(1) == 'BCM2708':
103     # Pi 1
104     return 1
105 elif match.group(1) == 'BCM2709':
106     # Pi 2
107     return 2
108 elif match.group(1) == 'BCM2835':
109     # Pi 3 / Pi on 4.9.x kernel
110     return 3
111 elif match.group(1) == 'BCM2711':
112     # Pi 4
113     return 3
114 else:
115     # Something else, not a pi.
116     return None
117

Shell x
Python 3.7.3 (/usr/bin/python3)
>>>
```

라즈베리 파이4의 BCM2711를 추가해 라즈베리 파이로 인식되도록 수정하였다.
이후 정상적으로 동작함을 확인할 수 있었다.



사진은 디스플레이 테스트 결과로 정해진 이미지를 디스플레이로 출력하는 모습을 볼 수 있다.

7.2. 로고출력



시작시에 출력되는 개인 로고

위와 같은 로고를 만들어 시작과 동시에 출력되도록 하였다.

```
#logo upload
```

```
image=Image.open('/home/pi/Desktop/logo.png').resize((width,height)).convert('1')
```

```
disp.image(image)
```

```
disp.display()
```

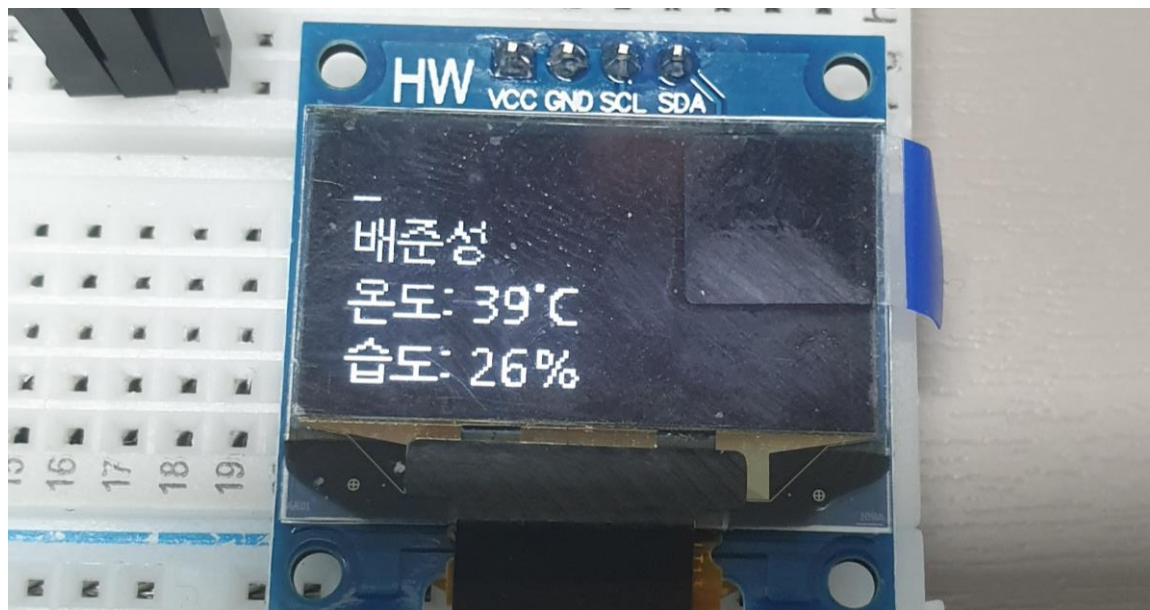
```
time.sleep(1.75)
```

```
start=1
```

출력될 이미지의 로고의 위치를 지정해주고 해당 로고 이미지를 디스플레이에 맞게 PIL라이브러리를 이용하여 리사이징 하여 출력해주도록 하였다. 해당 로고 이미지는 1.75초 동안 출력되고 다음으로 넘어간다. 마지막의 start변수는 시작시에 한 번만 로고를 출력하도록 해준다.



7.3. 온도 측정



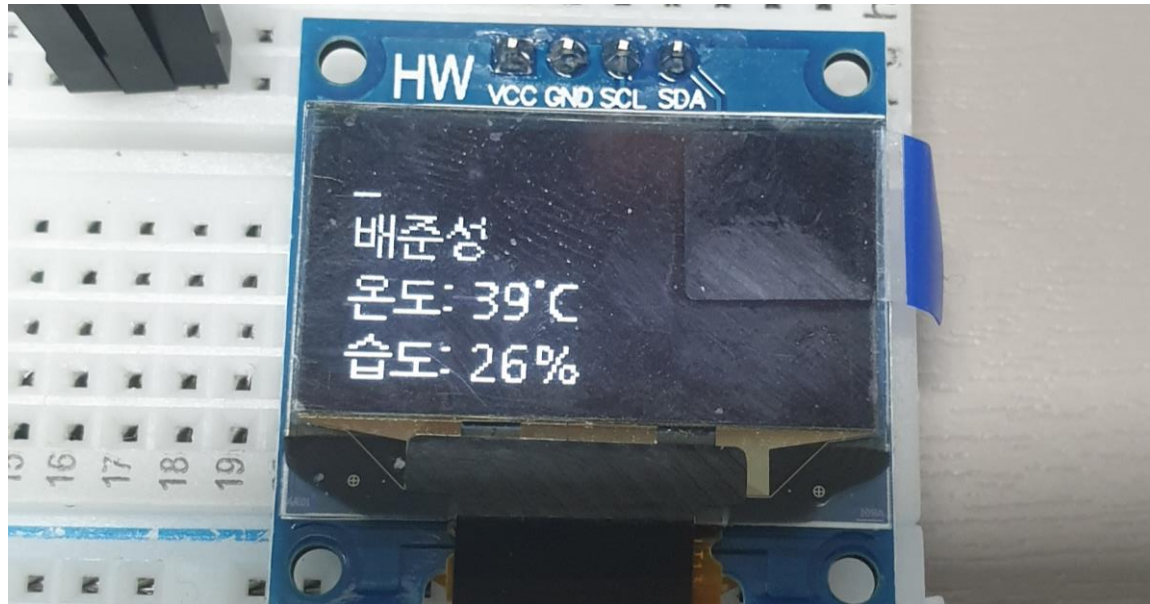
```
if count == '_':count = ''
    else:count = '_'

    h, t = Adafruit_DHT.read_retry(sensor,DHT11_pin)
    if h is not None and t is not None:
        temp="{0}\n{1}\n온도: {2}°C \n습도: {3}%".format(count,name,int(t),int(h))
    else:
        print('Read error')
```

온도 측정부의 경우 위와 같은 코드로 구성되어 있는데 count변수의 경우 맨위의 '_'를 출력해 주는 것으로 해당 온도계의 값이 갱신되는 동안 '_'와 ' '가 번갈아 가면서 들어가게 되는데 이를 통해서 온도, 습도 값에 변화가 없더라도 값이 정상적으로 갱신됨을 확인할 수 있다.

다음으로 온도, 습도 값을 Adafruit_DHT라이브러리의 read_retry 명령어를 이용하여 측정해 주고 각각의 값을 변수 h와 t에 입력한다. 이를 토대로 count, h, t를 조합하여 출력될 문장을 이름과 함께 원하는 형태로 만들어 temp 변수에 입력하였다. 후에 이 변수를 디스플레이에 출력하여 위와 같이 값을 출력하게 된다.

7.4. 디스플레이 출력



```
#화면크기 빈 이미지생성
image=Image.new('1',(width,height))
draw=ImageDraw.Draw(image)
#이미지위에 텍스트 출력
draw.text((0,0),temp,font=font,fill=255)
#이미지 OLED로 출력
disp.image(image)
disp.display()
time.sleep(0.5)
```

위에서 만들어진 값을 이 코드를 이용하여 출력한다. 우선 PIL라이브러리의 Image.new와 ImageDraw를 이용하여 화면크기의 빈 이미지를 만들어주고 그 위에 draw.text를 이용하여 원하는 텍스트를 입력해준다. 이렇게 만들어진 이미지는 disp를 이용하여 출력해준다. 0.5초동안 이 이미지를 출력하고 그후 새로운 값을 받아와 이미 지화 시켜 디스플레이에 출력해준다.

7.5. 종료화면 출력



```
except KeyboardInterrupt:
```

```
    print("Terminated by Keyboard")
```

```
finally:
```

```
    print("End of Promgram")
```

```
    image=Image.open('/home/pi/Desktop/A.png').resize((width,height)).convert('1')
```

```
    draw=ImageDraw.Draw(image)
```

```
    draw.text((0,0),' 프로그램을 종료합니다',font=font,fill=255)
```

```
    disp.image(image)
```

```
    disp.display()
```

```
    time.sleep(5)
```

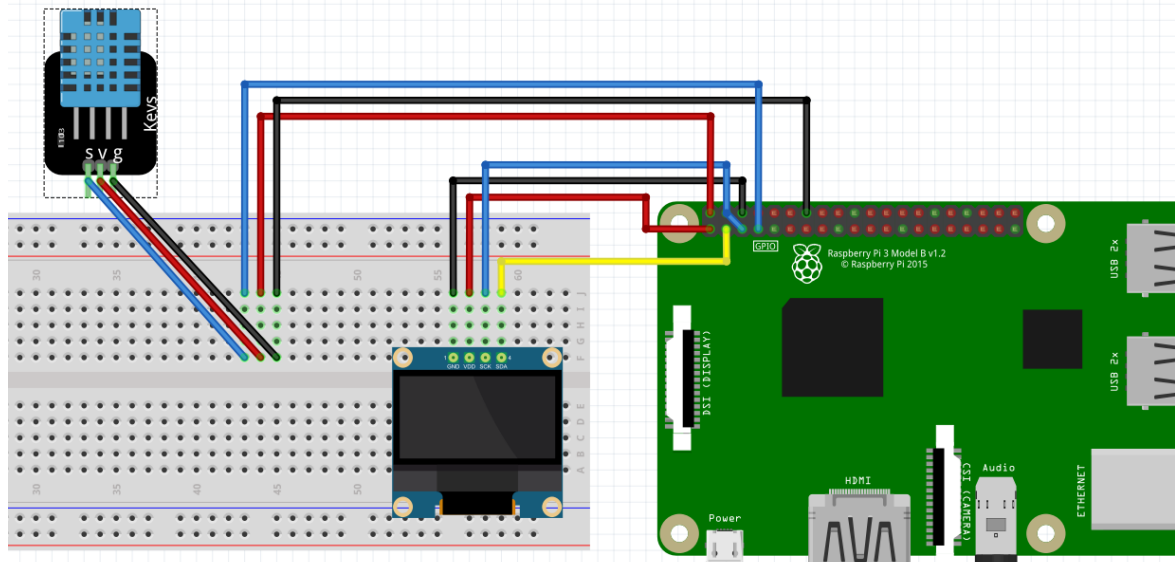
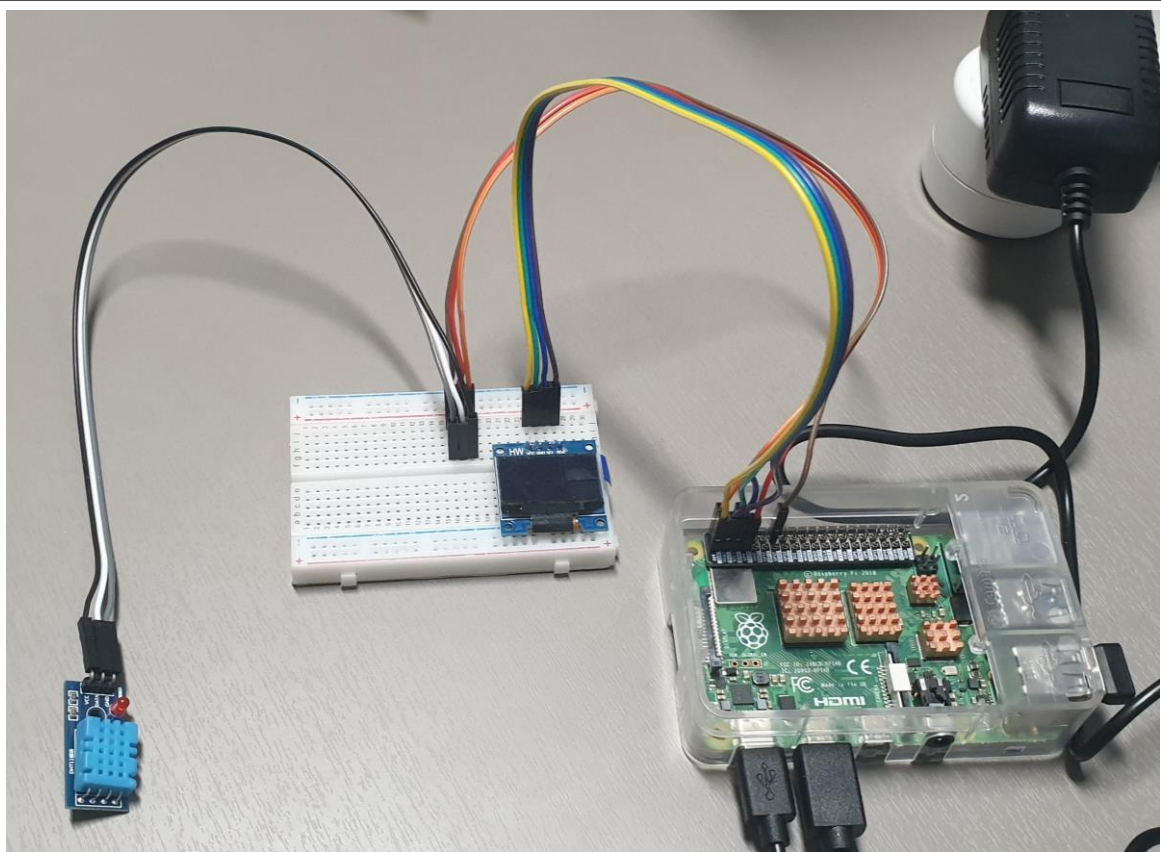
```
    #clr display
```

```
    disp.clear()
```

```
    disp.display()
```

추가적으로 인터럽트를 이용하여 컨트롤+c가 입력되면 위와 같은 이미지를 출력하고 프로그램의 루프를 중단하는 부분을 만들어 주었다.

하드웨어 연결구조



소자들 간의 연결은 위와 같다.

최종코드

```
import time
import Adafruit_DHT
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

#Raspberry Pi DHT11 sensor setting:
sensor = Adafruit_DHT.DHT11
DHT11_pin = 4
count = ' '
name = '배준성'
# Raspberry Pi pin configuratoin:
RST = 24
#124x64 display with hardwareI2C:
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)
#initialize Library
disp.begin()
width = disp.width
height = disp.height
top=10
font=ImageFont.truetype('/usr/share/fonts/truetype/nanum/NanumGothic.ttf',13)

start=0
try:
    while True:
        while start==0:

            #clr display
            disp.clear()
            disp.display()

            #logo upload
            image=Image.open('/home/pi/Desktop/logo.png').resize((width,height)).convert('1')
            disp.image(image)
            disp.display()
            time.sleep(1.75)
            start=1

        if count == ' ':count = ' '
        else:count = ' '

        h, t = Adafruit_DHT.read_retry(sensor,DHT11_pin)
        if h is not None and t is not None:
            temp="{0}\n{1}\n온도: {2}°C \n습도: {3}%".format(count,name,int(t),int(h))
        else:
            print('Read error')

        #화면크기 빈 이미지생성
        image=Image.new('1',(width,height))
        draw=ImageDraw.Draw(image)
        #이미지위에 텍스트 출력
        draw.text((0,0),temp,font=font,fill=255)
        #이미지 OLED로 출력
        disp.image(image)
        disp.display()
        time.sleep(0.5)

except KeyboardInterrupt:
    print("Terminated by Keyboard")

finally:
    print("End of Promgram")
    image=Image.open('/home/pi/Desktop/A.png').resize((width,height)).convert('1')
    draw=ImageDraw.Draw(image)
    draw.text((0,0),' 프로그램을 종료합니다',font=font,fill=255)
    disp.image(image)
    disp.display()
    time.sleep(5)
    #clr display
    disp.clear()
    disp.display()
```