AHT20

HƯỚNG DẪN SỬ DỤNG AHT20 DRIVER

TÁC GIẢ

- LÊ QUỐC DUY
- NGUYỄN QUỐC CƯỜNG
- LỮ ĐÌNH VĂN
- ĐỖ PHƯƠNG NAM

ASAIR

Driver của cảm biến AHT20 giúp tạo cầu nối giữa phần cứng cảm biến và phần mềm điều khiển, cho phép thực hiện các chức năng quan trọng như khởi tạo, đọc dữ liệu, chuyển đổi dữ liệu và kiểm tra trạng thái. Điều này giúp đảm bảo rằng các ứng dụng sử dụng cảm biến có thể hoạt động một cách ổn định và chính xác.
Driver của cảm biến AHT20 của chúng tôi tạo ra sẽ hỗ trợ bạn có thể đọc được dữ liệu nhiệt độ và độ ẩm với chỉ một lệnh duy nhất. Nhờ vậy bạn có thể ứng dụng dữ liệu đó phục vụ cho các mô hình cá nhân.

Made in China

MỤC LỤC

1. Giới thiệu

Figure 1. AHT20 module

Hướng dẫn này cung cấp thông tin về cách sử dụng driver AHT20 để đọc dữ liệu nhiệt độ và độ ẩm từ cảm biến AHT20 sử dụng giao tiếp I2C.

Driver AHT20 là một module kernel Linux tương tác với cảm biến nhiệt độ và độ ẩm AHT20 thông qua giao thức I2C. Nó cung cấp giao diện IOCTL để dễ dàng tương tác từ user space.

2. Kết nối vật lý

Phần cứng: Raspberry Pi 3B+ và cảm biến AHT20

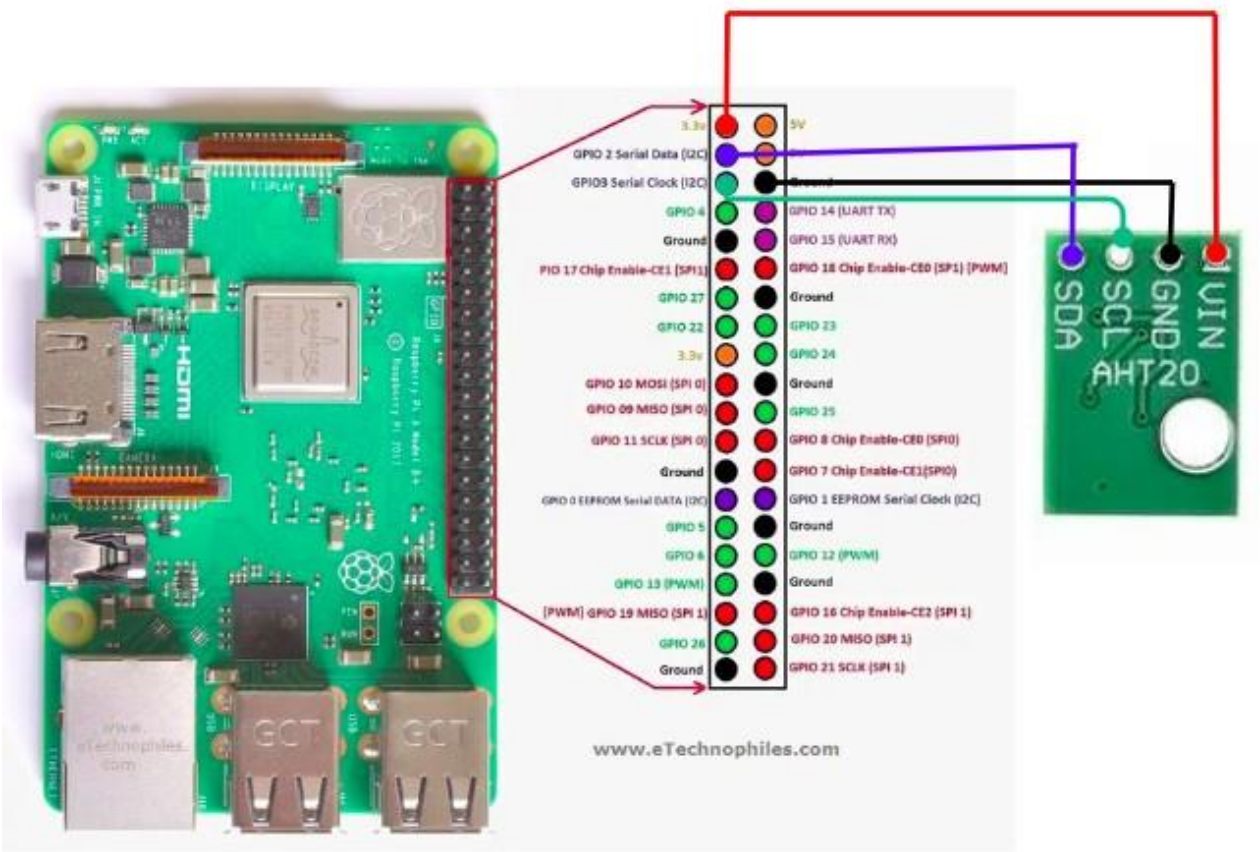

Figure 2. Sơ đồ kết nối module với raspberry

3. Cài đặt Driver

Xây dựng driver

→ Lưu mã driver vào một tệp có tên “aht20_driver.c”.
→ Tạo một Makefile với nội dung sau:

```
obj-m += aht20_driver.o
KDIR = /lib/modules/$(shell uname -r)/build

all:
   make -C $(KDIR) M=$(shell pwd) modules
clean:
   make -C $(KDIR) M=$(shell pwd) clean
```

→ Biên dịch driver bằng cách chạy lệnh sau trong thư mục chứa “aht20_driver.c” và Makefile bằng terminal:

```
pi@raspberrypi:~/Embeded_System/DO_AN_AHT20 $ make
make -C /lib/modules/6.1.21-v7+/build M=/home/pi/Embeded_System/DO_AN_AHT20 modules
make[1]: Entering directory '/usr/src/linux-headers-6.1.21-v7+'
  CC [M]  /home/pi/Embeded_System/DO_AN_AHT20/aht20_driver.o
  MODPOST /home/pi/Embeded_System/DO_AN_AHT20/Module.symvers
  CC [M]  /home/pi/Embeded_System/DO_AN_AHT20/aht20_driver.mod.o
  LD [M]  /home/pi/Embeded_System/DO_AN_AHT20/aht20_driver.ko
make[1]: Leaving directory '/usr/src/linux-headers-6.1.21-v7+'
pi@raspberrypi:~/Embeded_System/DO_AN_AHT20 $
```

Figure 3. Trình biên dịnh driver

Tải Driver

Tải module driver đã biên dịch vào kernel bằng lệnh “insmod”: sudo insmod aht20_driver.ko

Kiểm tra xem driver đã được tải: demsg

Kiểm tra số đăng ký của thiết bị: cat /proc/devices

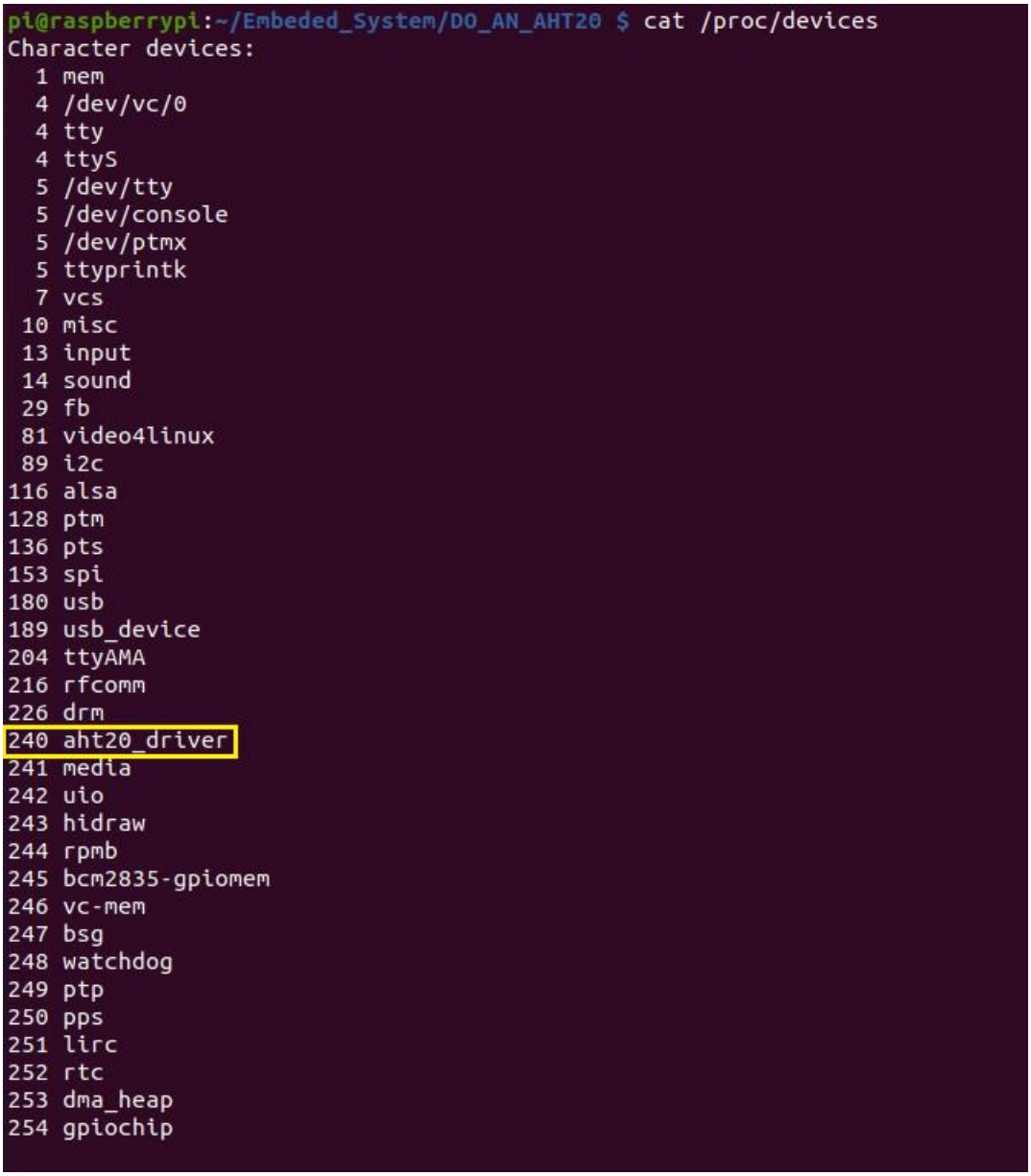

Figure 4. Kiểm tra số major number của thiết bị AHT20

Tạo file aht20_dev để kết nối với driver của thiết bị AHT20 trên hệ thống linux: Ví dụ như hình 4: sudo mknod /dev/aht20_dev c 240 0

Gỡ cài đặt driver:

Tải module driver đã biên dịch vào kernel bằng lệnh "insmod": sudo rmmod aht20_driver.ko

Kiểm tra xem driver đã được xóa: demsg

4. Hàm chức năng của Driver

a. Hàm đọc nhiệt độ từ cảm biến

aht20_read_temperature()

static int aht20_read_temperature(struct i2c_client *client,
uint32_t *temperature)

Chức năng : Đọc nhiệt độ từ cảm biến AHT20

Thông số :

*client con trỏ đến cấu trúc i2c client

*temperature con trỏ đến biến nơi lưu trữ nhiệt độ

Kết quả trả về : Trạng thái

Nếu thành công: trả về giá trị 0

Nếu thất bại: sẽ thấy báo lỗi

Ghi chú : Không

b. Hàm đọc độ ẩm từ cảm biến

aht20_read_humidity()

static int aht20_read_temperature(struct i2c_client *client,
uint32_t *humidity)

Chức năng : Đọc độ ẩm từ cảm biến AHT20

Thông số :

*client con trỏ đến cấu trúc i2c client

*temperature con trỏ đến biến nơi lưu trữ độ ẩm

Kết quả trả về : Trạng thái

Nếu thành công: trả về giá trị 0

Nếu thất bại: sẽ thấy báo lỗi

Ghi chú : Không

c. Hàm start cảm biến

aht20_start()

aht20_start(struct i2c_client *client)

Chức năng: Khởi tạo cảm biến aht20

Thông số:

*client con trỏ đến cấu trúc i2c client

Kết quả trả về : Trạng thái

Nếu thành công: trả về giá trị 0

Nếu thất bại: báo lỗi “Failed to start sensor”

Ghi chú : Không

d. Hàm Stop cảm biến

aht20_stop()

aht20_stop(struct i2c_client *client)

Chức năng: Khởi tạo cảm biến aht20

Thông số:

*client con trỏ đến cấu trúc i2c client

Kết quả trả về : Trạng thái

Nếu thành công: trả về giá trị 0

Nếu thất bại: báo lỗi “Failed to stop sensor”

Ghi chú : Không

5. Tương tác với Driver trên user space

Mở Thiết bị

Mở tệp thiết bị được tạo bởi driver. Tệp thiết bị này được đặt ở đường dẫn /dev/aht20_dev.

#define DEVICE_PATH "/dev/aht20_dev"

```
// Open the device
fd = open(DEVICE_PATH, O_RDWR);
if (fd == -1) {
    perror("Failed to open the device");
    return -1;
}
```

Sử dụng IOCTL để tương tác với user space. Cấu trúc của hàm Ioctl:

Ioctl(struct file *filep, unsigned int cmd, unsigned long arg)

filep: tệp thiết bị được tạo bởi driver.

cmd: tên hàm đã được định nghĩa.

Arg: biến để lưu output.

Hàm AHT20_READ_TEMPERATURE dùng để đọc nhiệt độ của cảm biến AHT20.

Khởi tạo AHT20_READ_TEMPERATURE:

#define AHT20_IOCTL_MAGIC 'A'

#define AHT20_READ_TEMPERATURE _IOR(AHT20_IOCTL_MAGIC, 1, uint32_t)

```
// Read temperature from sensor
if (ioctl(fd, AHT20_READ_TEMPERATURE, &temperature) < 0) {
    perror("Failed to perform ioctl");
    close(fd);
    return -1;
}
```

Hàm AHT20_READ_HUMIDITY dùng để đọc độ ẩm của cảm biến AHT20

Khởi tạo AHT20_READ_HUMIDITY:

#define AHT20_IOCTL_MAGIC1 'B'

#define AHT20_READ_HUMIDITY _IOR(AHT20_IOCTL_MAGIC1, 0, uint32_t)

```
// Read humidity from sensor
if (ioctl(fd, AHT20_READ_HUMIDITY, &humidity) < 0) {
    perror("Failed to perform ioctl");
    close(fd);
    return -1;
}
```

Đóng thiết bị:

```
// Close the device
close(fd);
```

6. Ứng dụng Mẫu

Sử dụng driver AHT20 đã cài đặt, chạy một chương trình để đọc nhiệt độ và độ ẩm của AHT20 thông qua IOCTL và hiển thị nhiệt độ lên led 7 đoạn max 7219 (sử dụng thư viện wiringPi cho max 7219)

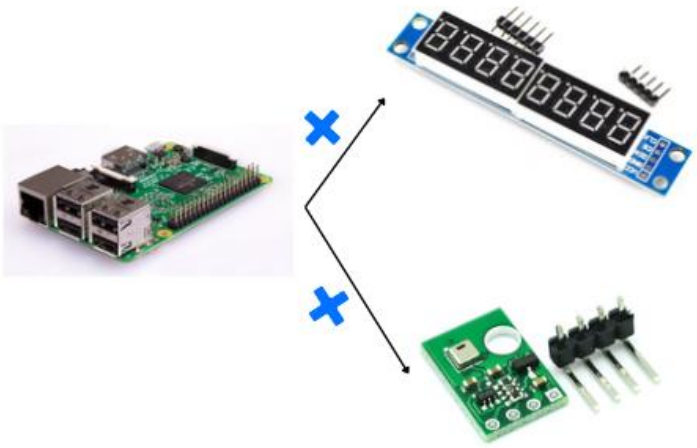

Figure 5. Kết hợp hiện thị trên max 7219

Chương trình mẫu:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <stdint.h>
#include <errno.h>
#include <string.h>
#include <wiringPi.h>

#define spi0 0

#define AHT20_IOCTL_MAGIC 'A'
#define AHT20_IOCTL_MAGIC1 'B'

#define AHT20_READ_TEMPERATURE _IOR(AHT20_IOCTL_MAGIC , 1, uint32_t)
#define AHT20_READ_HUMIDITY _IOR(AHT20_IOCTL_MAGIC1, 0, uint32_t)
#define AHT20_START _IO(AHT20_IOCTL_MAGIC, 2)
```

```
#define AHT20_STOP _IO(AHT20_IOCTL_MAGIC, 3)


#define DEVICE_PATH "/dev/aht20_dev"

#ifdef _WIN32
#define CLEAR_COMMAND "cls"
#else
#define CLEAR_COMMAND "clear"
#endif

uint8_t charTo7Segment(char c) {
   switch(c) {
      case 0 : return 0b01111110;
      case 1 : return 0b00110000;
      case 2 : return 0b01101101;
      case 3 : return 0b01111001;
      case 4 : return 0b00110011;
      case 5 : return 0b01011011;
      case 6 : return 0b01011111;
      case 7 : return 0b01110000;
      case 8 : return 0b01111111;
      case 9 : return 0b01111011;
      case '-' : return 0b00000001;
      default: return 0x00; // Blank for unsupported characters
   }
}

// Function to send data to MAX7219
void sendData(uint8_t address, uint8_t value) {
   uint8_t data[2];
   data[0] = address;
   data[1] = value;
   wiringPiSPIDataRW(spi0, data, 2);
}

// Function to setup MAX7219
void setup_max7912(){
   if (wiringPiSPISetup(spi0, 1000000) < 0) {
      fprintf(stderr, "SPI Setup failed: %s\n", strerror(errno));
      exit(1);
   }

   // Setup MAX7219
   sendData(0x09, 0x00); // no decode mode
   sendData(0x0A, 0x08); // intensity
```

```
    sendData(0x0B, 0x07); // scan limit
    sendData(0x0C, 0x01); // normal operation mode
    sendData(0x0F, 0x00); // turn off display test
}

// Function to display temperature on MAX7219
void displayTemperature(int temperature) {
    // Convert temperature to integer for display
    int data[4];

    //sendData(i + 5, charTo7Segment('-'));
    if (temperature < 0)
    {
        temperature = temperature*(-1);
        data[1] = temperature/100;
        data[2] = (temperature/10)%10;
        data[3] = temperature%10;

        for(int i = 0; i < 4 ; i++)
        {
            if(i == 3 ) sendData(i + 5, charTo7Segment('-'));
            else if (i == 1) sendData (i+5, charTo7Segment(data[3 - i]) |
0b10000000);
            else sendData(i+5, charTo7Segment(data[3 - i]));

        }
    }
    else
    {
        data[1] = temperature/100;
        data[2] = (temperature/10)%10;
        data[3] = temperature%10;
        for(int i = 0; i < 4 ; i++)
        {
            if(i == 3 ) sendData(i + 5, 0);
            else if (i == 1) sendData (i+5, charTo7Segment(data[3 - i]) |
0b10000000);
            else sendData(i+5, charTo7Segment(data[3 - i]));

        }
    }

}

// Function to display humidity on MAX7219
void displayHumidity(int humidity) {
```

```
    // Convert humidity to integer for display
    int data[4];
    data[0] = 0;
    data[1] = humidity/100;
    data[2] = (humidity/10)%10;
    data[3] = humidity%10;


   for(int i = 0; i < 4 ; i++)
   {
       if (i == 3) sendData (i+1, 0);
       else if (i == 1) sendData (i+1, charTo7Segment(data[3 - i]) |
0b10000000);
       else sendData(i+1, charTo7Segment(data[3 - i]));

   }
}
void display_clear() {
    // Convert humidity to integer for display
    for (int i = 1 ; i < 9 ; i++)
    {
       sendData(i,0);
    }
}
int main() {
    int fd;
    int humidity;
    int temperature;

    // Open the device
    fd = open(DEVICE_PATH, O_RDWR);
    if (fd == -1) {
       perror("Failed to open the device");
       return -1;
    }


    ioctl(fd, AHT20_START);

    // Setup MAX7219
    setup_max7912();
    display_clear();


    while (1){
       // Read temperature from sensor
```

```
            if (ioctl(fd, AHT20_READ_TEMPERATURE, &temperature) < 0) {
                perror("Failed to perform ioctl");
                close(fd);
                return -1;
            }
            ioctl(fd, AHT20_STOP);
            // Read humidity from sensor
            if (ioctl(fd, AHT20_READ_HUMIDITY, &humidity) < 0) {
                perror("Failed to perform ioctl");
                close(fd);
                return -1;
            }

            // Convert temperature and humidity to float
            float temp = (float) temperature / 10.0;
            float hum = (float) humidity / 10.0;
            // temperature = -200; // nhiet do am
            // Clear the screen
            system(CLEAR_COMMAND);

            // Display temperature and humidity
            printf("Temperature: %.1f°C\n", temp);
            printf("Humidity: %.1f%%\n", hum);

            // Display temperature and humidity on MAX7219
            displayTemperature(temperature);
            displayHumidity(humidity);

            // Delay for one second
            usleep(500000); //0.5s
        }

        ioctl(fd, AHT20_STOP);
        // Close the device
        close(fd);

        return 0;
    }
```

Kết quả:

- Hiển thị trên terminal:

```
Temperature: 27.7°C
Humidity: 50.4%
^C
pi@raspberrypi:~/Buoi1/Code_VND/Project $
```

- Hiển thị trên led 7 đoạn max 7219:

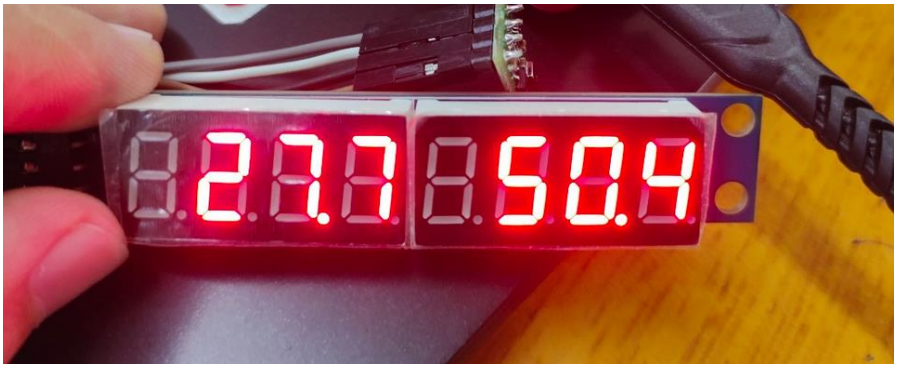

Figure 6 . Kết quả hiển thị

7. Cài đặt thư viện

Tạo một file Makefile có nội dung như sau:

```
CC = gcc
CFLAGS = -Wall -Iinclude

SRC = src/aht20.c
OBJ = $(SRC:.c=.o)

TARGET = libaht20.a
.PHONY: all clean
all: $(TARGET)
$(TARGET): $(OBJ)
	$(AR) rcs $@ $^
%.o: %.c
	$(CC) $(CFLAGS) -c -o $@ $<
clean:
	rm -f $(OBJ) $(TARGET)
```

Sau đó cài thư viện bằng lệnh make:

```
pi@raspberrypi:~/Embeded_System/AHT20_lib $ make
gcc -Wall -Iinclude -c -o src/aht20.o src/aht20.c
ar rcs libaht20.a src/aht20.o
pi@raspberrypi:~/Embeded_System/AHT20_lib $
```

Sau đó tạo một file code c "test_lib" để chạy thử thư viện, thêm #include "aht20.h":

```c
#include <stdio.h>
#include "aht20.h"

int main() {
    int file;
    uint32_t temperature;
    uint32_t humidity;

    if (aht20_init(&file) < 0) {
        return -1;
    }

    if (aht20_read_temperature(file, &temperature) == 0) {
        printf("Temperature: %d.%d°C\n", temperature / 10, temperature % 10);
    } else {
        printf("Failed to read temperature\n");
    }

    if (aht20_read_humidity(file, &humidity) == 0) {
        printf("Humidity: %d.%d%%\n", humidity / 10, humidity % 10);
    } else {
        printf("Failed to read humidity\n");
    }

    aht20_close(file);
    return 0;
}
```

Sử dụng lệnh: **gcc -o test test_lib.c -L. -laht20 -Iinclude** để biên dịch chương trình

Sau đó chạy lệnh: **./test** để chạy thử chương trình.

```
pi@raspberrypi:~/Embeded_System/AHT20_lib $ ./test
Temperature: 31.8°C
Humidity: 81.0%
```

8. Các hàm trong thư viện

a. Hàm aht20_init()

Cấu trúc: aht20_init(int *file)

Chức năng: Khởi tạo kết nỗi I2C

Set địa chỉ I2c

Thông số đầu vào: con trỏ *file dùng để lưu trữ địa chỉ của i2c-1

Kết quả trả về : Trạng thái

Nếu thành công: trả về giá trị 0

Nếu thất bại: sẽ thông báo lỗi.

Ghi chú: không

b. Hàm aht20_read_temperature()

Cấu trúc: aht20_read_temperature(int file, uint32_t *temperature)

Chức năng: Đọc nhiệt độ từ cảm biến AHT20

Thông số đầu vào: int file và uint32_t *temperture

Kết quả trả về : Trạng thái

Nếu thành công: trả về giá trị 0

Nếu thất bại: sẽ thông báo lỗi.

Ghi chú: không

c. Hàm aht20_read_humidity()

Cấu trúc: aht20_read_temperature(int file, uint32_t *humidity)

Chức năng: Đọc nhiệt độ từ cảm biến AHT20

Thông số đầu vào: int file và uint32_t *humidity

Kết quả trả về : Trạng thái

Nếu thành công: trả về giá trị 0

Nếu thất bại: sẽ thông báo lỗi.

Ghi chú: không

d. Hàm aht20_close()

Cấu trúc: aht20_close(int file)

Chức năng: dùng để đóng file đã mở.

Thông số đầu vào: int file

Ghi chú: không