How to use a Digital Sensor with RIOT

using an STM32 Nucleo-64 F401RE development board

Ioannis Chatzigiannakis

https://github.com/ichatz/riotos-apps

Measure the ambient temperature and humidity with an STM32 Nucleo-64 F401RE development board and the RIOT operating system.

Required hardware components:

- STM32 Nucleo-64 F401RE

Measure the ambient temperature and humidity with an STM32 Nucleo-64 F401RE development board and the RIOT operating system.

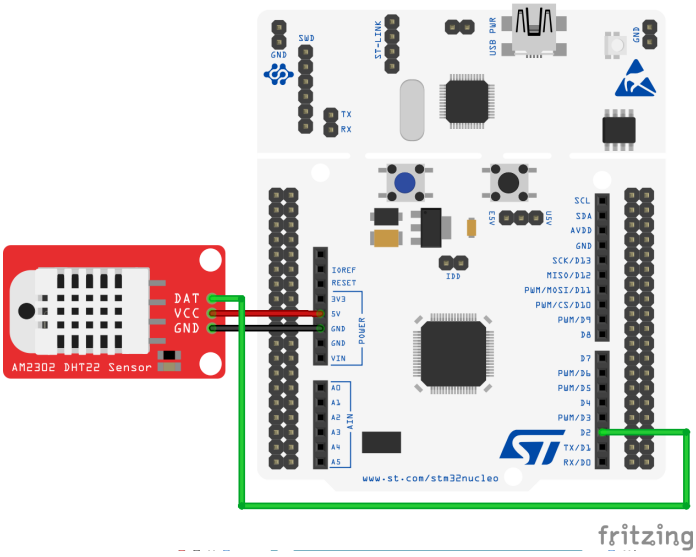
Required hardware components:

- STM32 Nucleo-64 F401RE
- DHT22 digital sensor

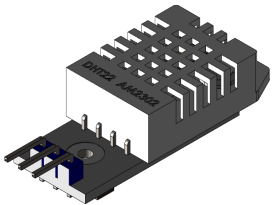
Measure the ambient temperature and humidity with an STM32 Nucleo-64 F401RE development board and the RIOT operating system.

Required hardware components:

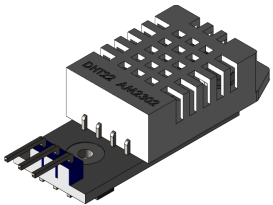
- STM32 Nucleo-64 F401RE
- DHT22 digital sensor
- 3 Female to male jumper wires


DAT
VCC
GND
AM2302 DHT22 Sensor
SWD
GND
ST-LINK
USB PWR
GND
TX
RX
SCL
SDA
AVDD
GND
SCK/D13
MISO/D12
PWM/MOSI/D11
PWM/CS/D10
PWM/D9
D8
D7
PWM/D6
PWM/D5
D4
PWM/D3
D2
TX/D1
RX/D0
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
AIN
IDD
E5V
VIN
www.st.com/stm32nucleo
fritzing

- The DHT22 digital sensor is a widely diffused component
  - Measures temperature and relative humidity
  - Uses a custom protocol which use a single wire/bus for communication.

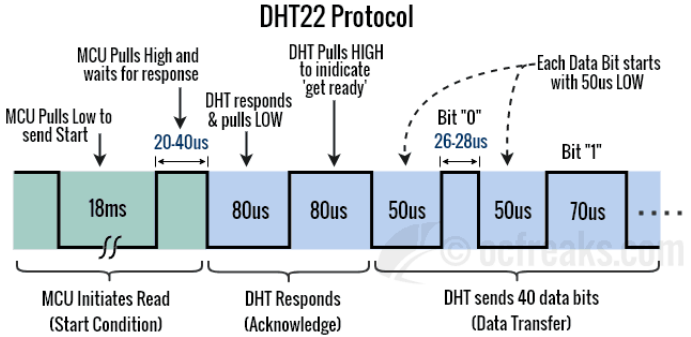


- The DHT22 digital sensor is a widely diffused component
  - Measures temperature and relative humidity
  - Uses a custom protocol which use a single wire/bus for communication.
- The DATA wire used for communication between STM32 MCU and the DHT22.
  - A $4.7K\Omega$ or $10K\Omega$ pull-up resistor is used to bring the bus in an IDLE state when there is no communication taking place.
  - A continuous HIGH on the line denote an IDLE state.
  - The STM32 MCU acts as the bus controller and hence is responsible for initiating communication (i.e., read).



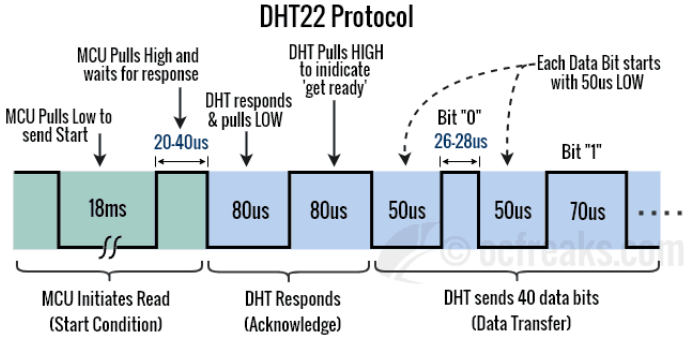
DHT22 Communication Protocol

1. The STM32 MCU pulls it to a LOW for 18ms and HIGH for around $20 \ldots 40 \mu s$.



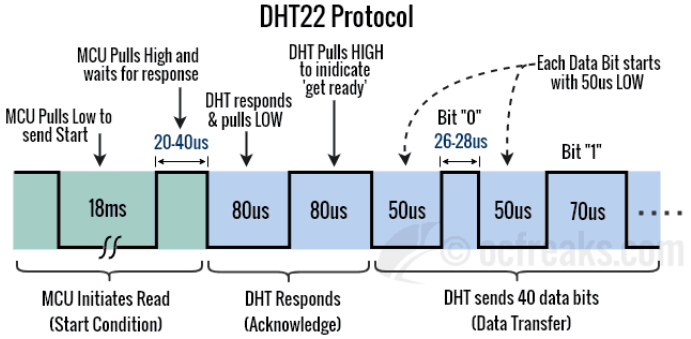
DHT22 Communication Protocol

1. The STM32 MCU pulls it to a LOW for 18ms and HIGH for around $20 \ldots 40\mu s$.
2. DHT22 detects a START and responds by pulling the line LOW for 80μs.



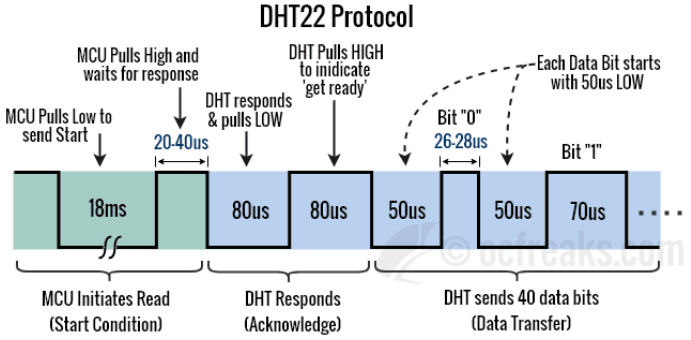
DHT22 Communication Protocol

1. The STM32 MCU pulls it to a LOW for 18ms and HIGH for around $20 \ldots 40\mu s$.
2. DHT22 detects a START and responds by pulling the line LOW for 80μs.
3. DHT22 pulls it HIGH for 80μs which indicates that it is ready to send 40 bits data.



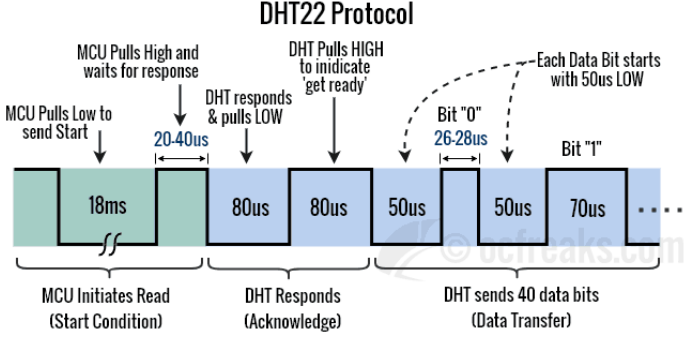
DHT22 Communication Protocol

1. The STM32 MCU pulls it to a LOW for 18ms and HIGH for around $20 \ldots 40\mu s$.
2. DHT22 detects a START and responds by pulling the line LOW for 80μs.
3. DHT22 pulls it HIGH for 80μs which indicates that it is ready to send 40 bits data.
4. A bit starts with a 50μs LOW followed by $26 - 28\mu$s for a "0" or 70μs for a "1".



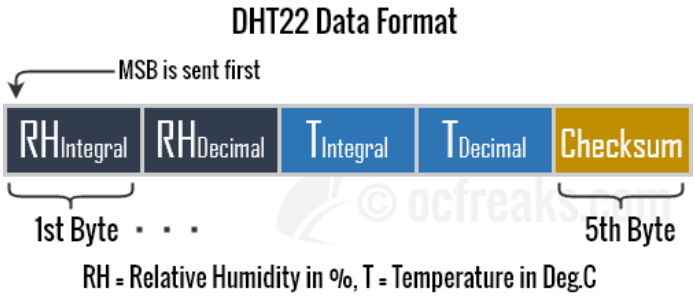
DHT22 Communication Protocol

1. The STM32 MCU pulls it to a LOW for 18ms and HIGH for around $20 \ldots 40\mu s$.
2. DHT22 detects a START and responds by pulling the line LOW for 80μs.
3. DHT22 pulls it HIGH for 80μs which indicates that it is ready to send 40 bits data.
4. A bit starts with a 50μs LOW followed by $26 - 28\mu$s for a "0" or 70μs for a "1".
5. To ends, the Line is pulled HIGH by the pull-up resistor and enters IDLE state.



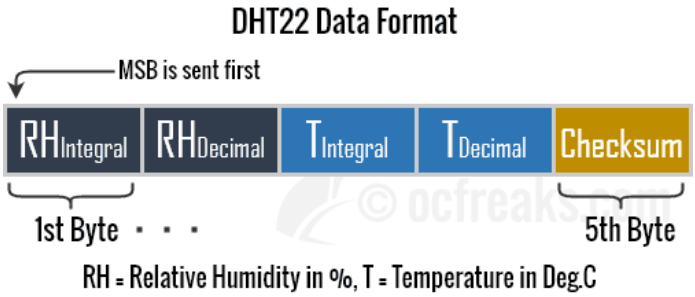
DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)



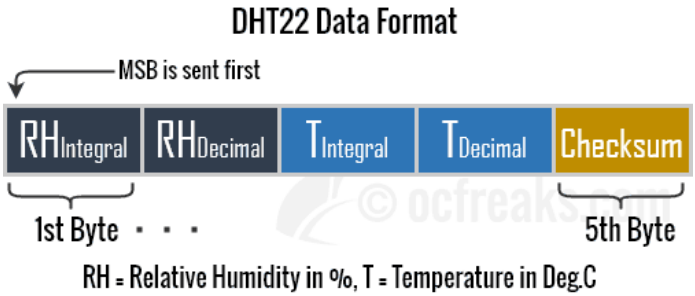
DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part)



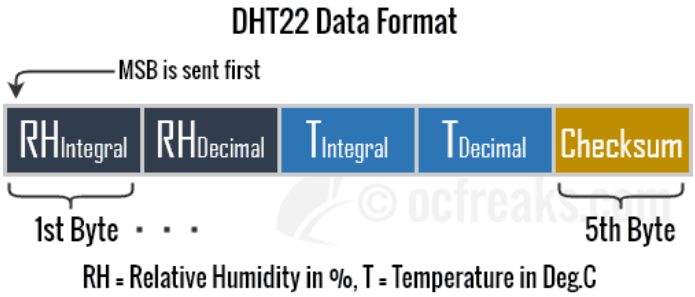
DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part)
3. 3rd Byte: Temperature Integral in Degree Celsius (Integer Part)



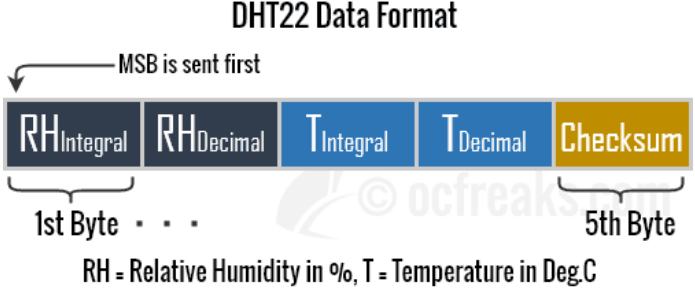
DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part)
3. 3rd Byte: Temperature Integral in Degree Celsius (Integer Part)
4. 4th Byte: Temperature in Decimal Data in % (Fractional Part)



DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part)
3. 3rd Byte: Temperature Integral in Degree Celsius (Integer Part)
4. 4th Byte: Temperature in Decimal Data in % (Fractional Part)
5. 5th Byte: Checksum (Last 2 bits of 1st Byte + 2nd Byte + 3rd Byte+ 4th Byte)



Creating a new application in RIOT OS – Step 1

1. Identify location of RIOT folder
   e.g., /home/ichatz/RIOT

Makefile

```
APPLICATION = myapp

BOARD ?= nucleo-f401re

RIOTBASE ?= $(CURDIR)/../RIOT

QUIET ?= 0
DEVELHELP ?= 1

include $(RIOTBASE)/Makefile.include
```

Creating a new application in RIOT OS – Step 1

1. Identify location of RIOT folder
   e.g., /home/ichatz/RIOT
2. Create a new folder
   e.g., /home/ichatz/myapp

Makefile

```
APPLICATION = myapp

BOARD ?= nucleo-f401re

RIOTBASE ?= $(CURDIR)/../RIOT

QUIET ?= 0
DEVELHELP ?= 1

include $(RIOTBASE)/Makefile.include
```

Creating a new application in RIOT OS – Step 1

1. Identify location of RIOT folder
   e.g., /home/ichatz/RIOT
2. Create a new folder
   e.g., /home/ichatz/myapp
3. Create a file named Makefile using your favorite editor.

Makefile

```
APPLICATION = myapp

BOARD ?= nucleo-f401re

RIOTBASE ?= $(CURDIR)/../RIOT

QUIET ?= 0
DEVELHELP ?= 1

include $(RIOTBASE)/Makefile.include
```

Creating a new application in RIOT OS – Step 1

1. Identify location of RIOT folder
   e.g., /home/ichatz/RIOT
2. Create a new folder
   e.g., /home/ichatz/myapp
3. Create a file named Makefile using your favorite editor.
4. Insert the contents provided here.

Makefile

```
APPLICATION = myapp

BOARD ?= nucleo-f401re

RIOTBASE ?= $(CURDIR)/../RIOT

QUIET ?= 0
DEVELHELP ?= 1

include $(RIOTBASE)/Makefile.include
```

Creating a new application in RIOT OS – Step 1

1. Identify location of RIOT folder
   e.g., /home/ichatz/RIOT
2. Create a new folder
   e.g., /home/ichatz/myapp
3. Create a file named `Makefile` using your favorite editor.
4. Insert the contents provided here.
5. Change QUIET from 0 → 1 to see the compiler invocation commands.

Makefile

```
APPLICATION = myapp

BOARD ?= nucleo-f401re

RIOTBASE ?= $(CURDIR)/../RIOT

QUIET ?= 0
DEVELHELP ?= 1

include $(RIOTBASE)/Makefile.include
```

Creating a new application in RIOT OS – Step 1

1. Identify location of RIOT folder
   e.g., /home/ichatz/RIOT
2. Create a new folder
   e.g., /home/ichatz/myapp
3. Create a file named `Makefile` using your favorite editor.
4. Insert the contents provided here.
5. Change QUIET from 0 $\rightarrow$ 1 to see the compiler invocation commands.
6. Change DEVELHELP from 1 $\rightarrow$ 0 to remove debug information.

Makefile

```
APPLICATION = myapp

BOARD ?= nucleo-f401re

RIOTBASE ?= $(CURDIR)/../RIOT

QUIET ?= 0
DEVELHELP ?= 1

include $(RIOTBASE)/Makefile.include
```

Creating a new application in RIOT OS – Step 2

7. Create the file named `main.c` file using your favorite editor.

main.c

```c
#include <stdio.h>

int main(void) {
    printf("RIOT empty app.\n");
    return 0;
}
```

Creating a new application in RIOT OS – Step 2

9. Create the file named `main.c` file using your favorite editor.
10. Insert the contents provided here.

main.c

```c
#include <stdio.h>

int main(void) {
    printf("RIOT empty app.\n");
    return 0;
}
```

Creating a new application in RIOT OS – Step 3

11. Open a terminal and go to the folder of the RIOT application e.g., /home/ichatz/myapp

Command line

```
ichatz:~/# cd myapp
ichatz:~/myapp# make
ichatz:~/myapp# make flash
ichatz:~/myapp# make term
```

Creating a new application in RIOT OS – Step 3

11. Open a terminal and go to the folder of the RIOT application e.g., `/home/ichatz/myapp`
12. Compile the code using `make`

Command line

```
ichatz:~/# cd myapp
ichatz:~/myapp# make
ichatz:~/myapp# make flash
ichatz:~/myapp# make term
```

Creating a new application in RIOT OS – Step 3

11. Open a terminal and go to the folder of the RIOT application e.g., `/home/ichatz/myapp`
12. Compile the code using `make`
13. Program the STM32 board using `make flash`

Command line

```
ichatz:~/# cd myapp
ichatz:~/myapp# make
ichatz:~/myapp# make flash
ichatz:~/myapp# make term
```

Creating a new application in RIOT OS – Step 3

11. Open a terminal and go to the folder of the RIOT application
    e.g., `/home/ichatz/myapp`
12. Compile the code using `make`
13. Program the STM32 board using `make flash`
14. Connect to the STM32 board through the USB to check debug output using `make term`

Command line

```
ichatz:~/# cd myapp
ichatz:~/myapp# make
ichatz:~/myapp# make flash
ichatz:~/myapp# make term
```

Using the DHT22 Driver of RIOT

Makefile

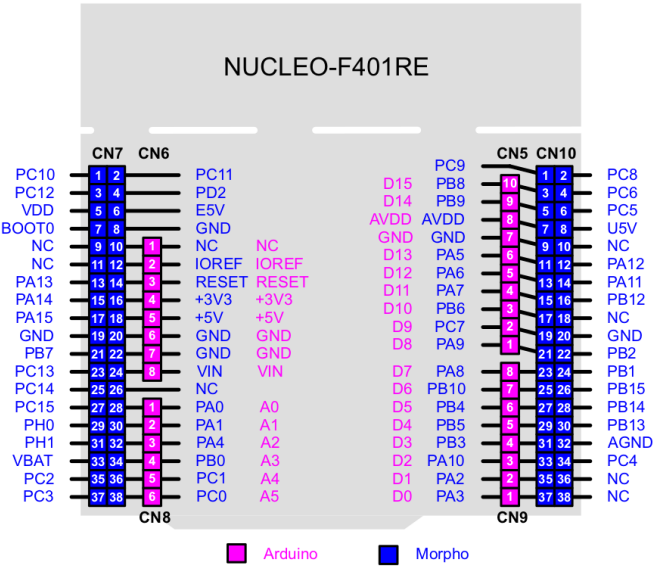
```
APPLICATION = myapp
BOARD ?= nucleo-f401re
RIOTBASE ?= $(CURDIR)/../RIOT

# Modules to include:
USEMODULE += dht
USEMODULE += fmt

QUIET ?= 0
DEVELHELP ?= 1

include $(RIOTBASE)/Makefile.include
```

← We wish to use the DHT module.
← FMT = Format Module to help with the DHT protocol.


NUCLEO-F401RE
CN7
CN6
CN8
CN5
CN10
CN9
PC10 PC12 VDD BOOT0 NC NC PA13 PA14 PA15 GND PB7 PC13 PC14 PC15 PH0 PH1 VBAT PC2 PC3
PC11 PD2 E5V GND NC IOREF RESET +3V3 +5V GND GND VIN NC PA0 PA1 PA4 PB0 PC1 PC0
NC IOREF RESET +3V3 +5V GND GND VIN A0 A1 A2 A3 A4 A5
PC9 D15 PB8 D14 PB9 AVDD AVDD GND GND D13 PA5 D12 PA6 D11 PA7 D10 PB6 D9 PC7 D8 PA9
D7 PA8 D6 PB10 D5 PB4 D4 PB5 D3 PB3 D2 PA10 D1 PA2 D0 PA3
PC8 PC6 PC5 U5V NC PA12 PA11 PB12 NC GND PB2 PB1 PB15 PB14 PB13 AGND PC4 NC NC
Arduino
Morpho

main.c

```c
#include "fmt.h"
#include "dht.h"
#include "dht_params.h"

int main(void) {
    dht_params_t my_params;
    my_params.pin = GPIO_PIN(PORT_A, 10);
    my_params.type = DHT22;
    my_params.in_mode = DHT_PARAM_PULL;

    dht_t dev;
    if (dht_init(&dev, &my_params) == DHT_OK) {
        printf("DHT sensor connected\n");
    }
```

```
    int16_t temp, hum;
    if (dht_read(&dev, &temp, &hum) != DHT_OK) {
        printf("Error reading values\n");
    }

    char temp_s[10];
    size_t n = fmt_s16_dfp(temp_s, temp, -1);
    temp_s[n] = '\0';

    char hum_s[10];
    n = fmt_s16_dfp(hum_s, hum, -1);
    hum_s[n] = '\0';

    printf("DHT values - temp: %s°C - relative humidity: %s%%\n",
           temp_s, hum_s);
}
```

Command line

```
ichatz:~/# cd myapp
ichatz:~/myapp# make flash
ichatz:~/myapp# make term
/home/ichatz/RIOT/dist/tools/pyterm/pyterm -p "/dev/ttyACM0" -b "115200"
Twisted not available, please install it if you want to use pyterm's JSON
# Connect to serial port /dev/ttyACM0
Welcome to pyterm!
Type '/exit' to exit.
# DHT sensor connected
# DHT values - temp: 21.8°C - relative humidity: 46.9%
```