How to use a Digital Sensor with RIOT
using an STM32 Nucleo-64 F401RE development board

Ioannis Chatzigiannakis

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

Low-power measurement of 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

Low-power measurement of the ambient temperature and humidity with an STM32 Nucleo-64 F401RE development board and the RIOT operating system.
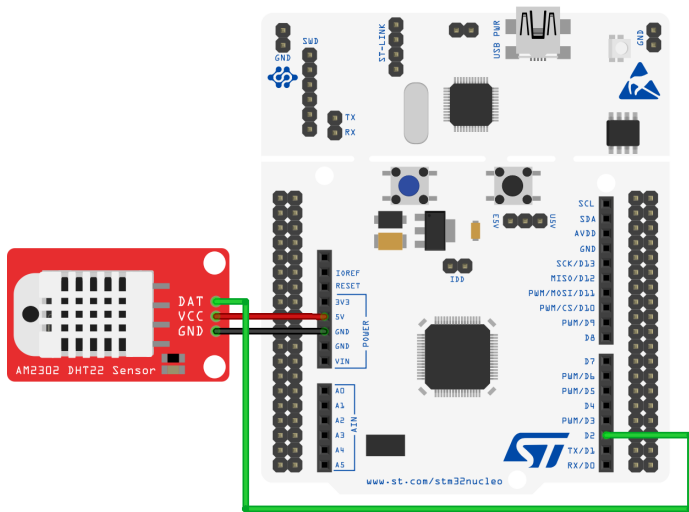
Required hardware components:

- STM32 Nucleo-64 F401RE
- DHT22 digital sensor

Low-power measurement of the ambient temperature and humidity with an STM32 Nucleo-64 F401RE development board and the RIOT operating system.
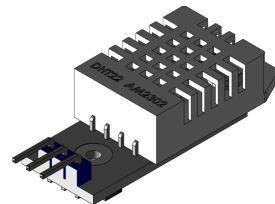
Required hardware components:

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

- 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.
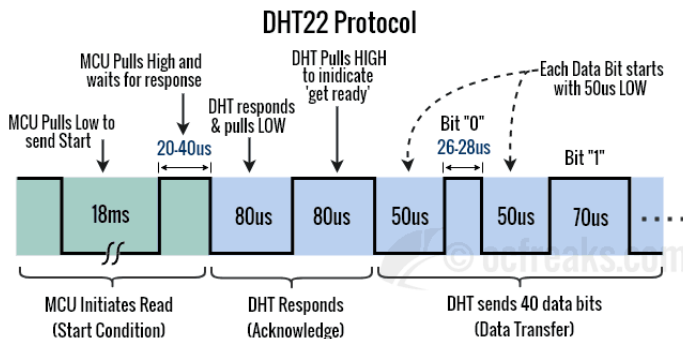
- The DHT22 digital sensor is a widely diffused component
  - Measures temperature and relative humidity
  - Uses a custom protocol which use a single wire/bus for communication.
- The DATA wire used for communication between STM32 MCU and the DHT22.
  - A 4.7$K\Omega$ or 10$K\Omega$ pull-up resistor 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).
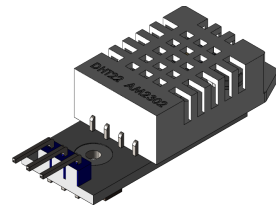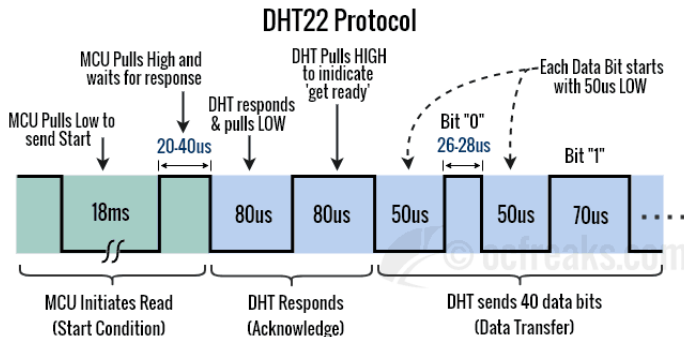
DHT22 Communication Protocol

1. The STM32 MCU pulls it to a LOW for 18ms and HIGH for around $20\ldots40\mu s$.

DHT22 Protocol

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 $80\mu s$.

DHT22 Protocol

MCU Pulls High and waits for response

DHT Pulls HIGH to inidicate 'get ready'

Each Data Bit starts with 50us LOW

MCU Pulls Low to send Start

20-40us

DHT responds & pulls LOW

Bit "0"

26-28us

Bit "1"

18ms   80us   80us   50us   50us   70us   · · · ·

MCU Initiates Read (Start Condition)

DHT Responds (Acknowledge)

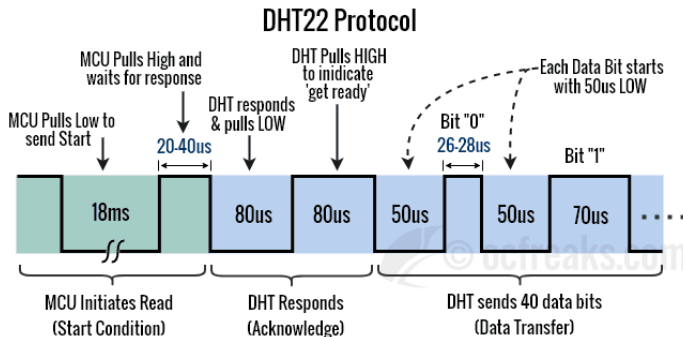DHT sends 40 data bits (Data Transfer)

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 $80 \mu s$.
3. DHT22 pulls it HIGH for $80 \mu s$ which indicates that it is ready to send 40 bits data.

DHT22 Protocol

MCU Pulls High and
waits for response

DHT Pulls HIGH
to inidicate
'get ready'

Each Data Bit starts
with 50us LOW

MCU Pulls Low to
send Start

DHT responds
& pulls LOW

20-40us

Bit "0"
26-28us

Bit "1"

18ms   80us   80us   50us   50us   70us   . . . .

© ucfreaks.com

MCU Initiates Read
(Start Condition)

DHT Responds
(Acknowledge)

DHT sends 40 data bits
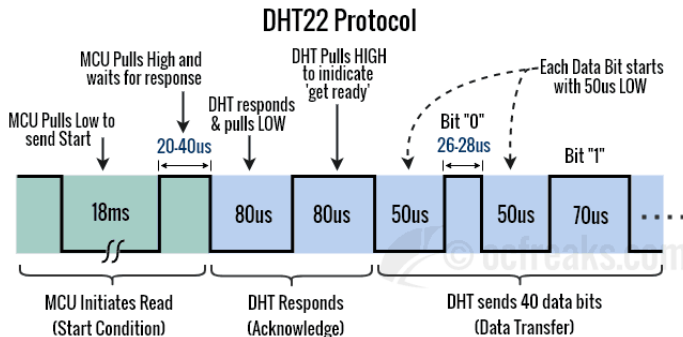(Data Transfer)

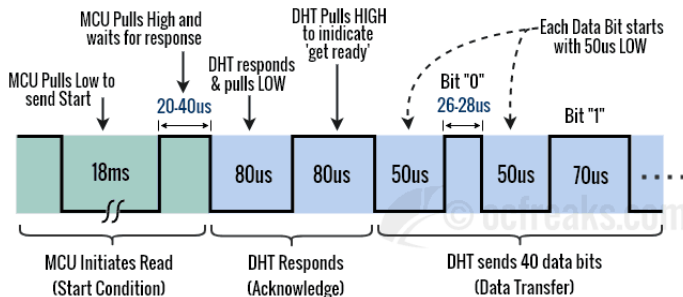RIOT   STM32

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 $80\mu s$.
3. DHT22 pulls it HIGH for $80\mu s$ which indicates that it is ready to send 40 bits data.
4. A bit starts with a $50\mu s$ LOW followed by $26 - 28\mu s$ for a "0" or $70\mu s$ for a "1".

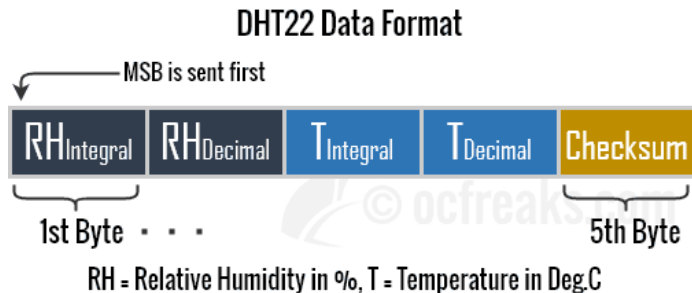DHT22 Protocol

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 $80\mu s$.
3. DHT22 pulls it HIGH for $80\mu s$ which indicates that it is ready to send 40 bits data.
4. A bit starts with a $50\mu s$ LOW followed by $26 - 28\mu s$ for a "0" or $70\mu s$ for a "1".
5. To ends, the Line is pulled HIGH by the pull-up resistor and enters IDLE state.

DHT22 Protocol

DHT22 Data Format

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

DHT22 Data Format

MSB is sent first

| RH$_{Integral}$ | RH$_{Decimal}$ | T$_{Integral}$ | T$_{Decimal}$ | Checksum |

1st Byte · · · 5th Byte

RH = Relative Humidity in %, T = Temperature in Deg.C

DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part) – Zero for DHT11
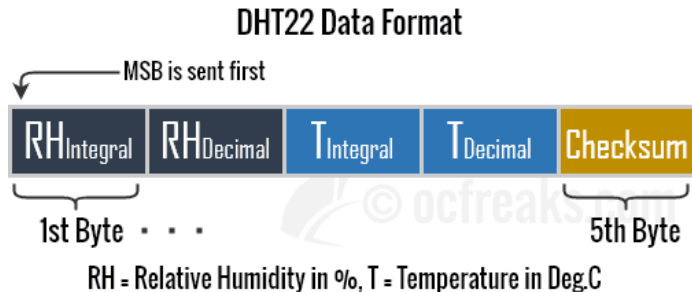

DHT22 Data Format
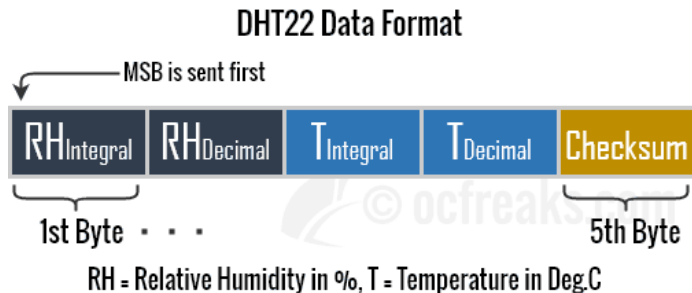
RH = Relative Humidity in %, T = Temperature in Deg.C

DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part) – Zero for DHT11
3. 3rd Byte: Temperature Integral in Degree Celsius (Integer Part)

DHT22 Data Format

MSB is sent first

| RH$_{Integral}$ | RH$_{Decimal}$ | T$_{Integral}$ | T$_{Decimal}$ | Checksum |

1st Byte · · · 5th Byte

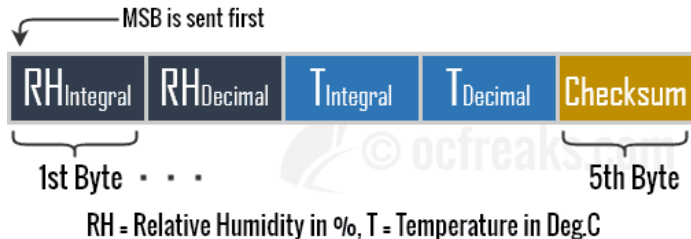RH = Relative Humidity in %, T = Temperature in Deg.C

DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part) – Zero for DHT11
3. 3rd Byte: Temperature Integral in Degree Celsius (Integer Part)
4. 4th Byte: Temperature in Decimal Data in % (Fractional Part) – Zero for DHT11

DHT22 Data Format

MSB is sent first

| RH$_{Integral}$ | RH$_{Decimal}$ | T$_{Integral}$ | T$_{Decimal}$ | Checksum |

1st Byte · · ·   5th Byte

RH = Relative Humidity in %, T = Temperature in Deg.C

DHT22 Data Format

1. 1st Byte: Relative Humidity Integral Data in % (Integer Part)
2. 2nd Byte: Relative Humidity Decimal Data in % (Fractional Part) – Zero for DHT11
3. 3rd Byte: Temperature Integral in Degree Celsius (Integer Part)
4. 4th Byte: Temperature in Decimal Data in % (Fractional Part) – Zero for DHT11
5. 5th Byte: Checksum (Last 8 bits of 1st Byte + 2nd Byte + 3rd Byte+ 4th Byte)

DHT22 Data Format

RH = Relative Humidity in %, T = Temperature in Deg.C

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

DEVELHELP ?= 1
QUIET ?= 0

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

How to use a Digital Sensor with RIOT
○○○○○●○○○○○○○○
The RIOT operating system

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

DEVELHELP ?= 1
QUIET ?= 0

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

DEVELHELP ?= 1
QUIET ?= 0

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

DEVELHELP ?= 1
QUIET ?= 0

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

CRIOT

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.

Makefile

```
APPLICATION = myapp

BOARD ?= nucleo-f401re

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

DEVELHELP ?= 1
QUIET ?= 0

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

DEVELHELP ?= 1
QUIET ?= 0

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;
}
```

How to use a Digital Sensor with RIOT
○○○○○○●○○○○○○
The RIOT operating system

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
```

How to use a Digital Sensor with RIOT
○○○○○○○○○●○○○○○
The RIOT operating system

Hardware Independent elements

Makefile

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

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

DEVELHELP ?= 1
QUIET ?= 0
include $(RIOTBASE)/Makefile.include
```
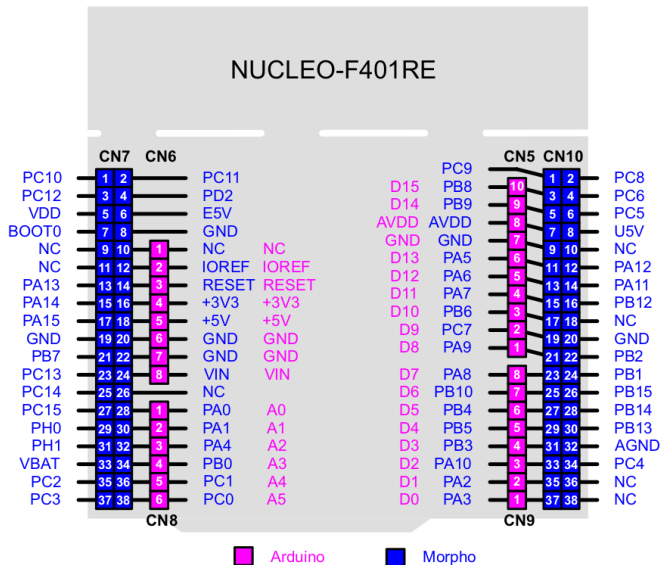
← We wish to use the DHT module.
← FMT = Format Module to help with the DHT protocol.

main.c

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

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

- Use the `struct` `dht_params_t` to provide the PIN used.

```c
int main(void) {
    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%
```