How to use a Digital Sensor with RIOT

using an STM32 Nucleo-64 F401RE development board

Ioannis Chatzigiannakis

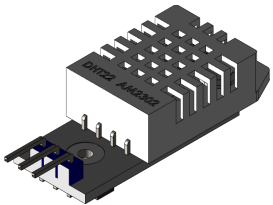
https://github.com/ichatz/riotos-apps

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:

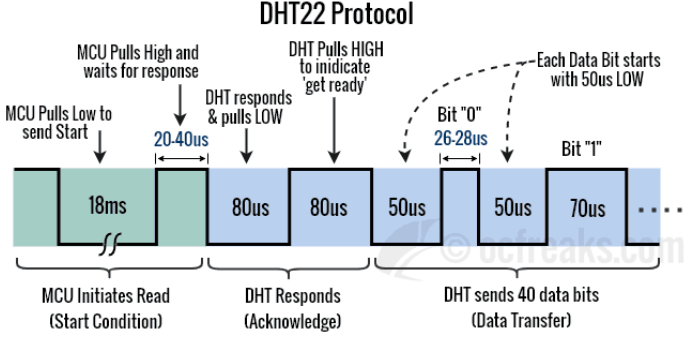
- STM32 Nucleo-64 F401RE
- DHT22 digital sensor
- 3 Female to male jumper wires

- 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 DATA wire used for communication between STM32 MCU and the DHT22.
  - A 4.7K or 10K 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).



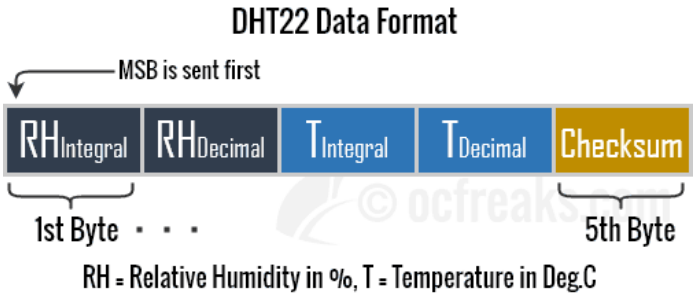
DHT22 Communication Protocol

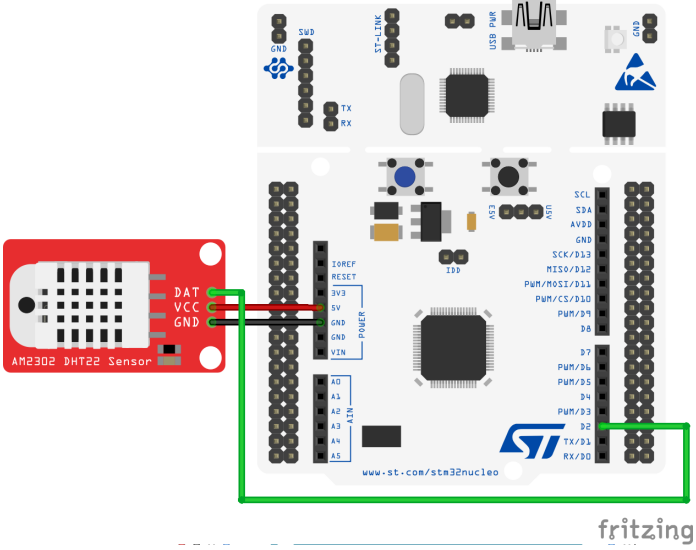
1. The STM32 MCU pulls it to a LOW for 18ms and HIGH for around 20 to 40us.
2. DHT22 detects a START and responds by pulling the line LOW for 80μs.
3. DHT22 pulls it HIGH for 80μs which indicates that it is ready to send 40 bits data.
4. Each bit starts with a 50μs LOW followed by 26-28μs for a "0" or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




DAT
VCC
GND
AM2302 DHT22 Sensor
SWD
GND
TX
RX
ST-LINK
USB PWR
GND
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
AIN
IDD
E5V
U5V
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
www.st.com/stm32nucleo
fritzing

Hardware Independent elements

Makefile

```
# name of application
APPLICATION = photocell

# Path to the RIOT base directory:
RIOTBASE ?= $(CURDIR)/../../RIOT

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

# RIOT features
FEATURES_OPTIONAL += periph_rtc
```

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

The RTC module used for power management.

main.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);
}
```

```c
int main(void) {
    const int mode = 0;
    const int delay = 5;

    printf("Setting wakeup from mode %d in %d seconds.\n", mode, delay);
    fflush(stdout);

    /* Setting a timer on the RTC */
    struct tm time;
    rtc_get_time(&time);
    time.tm_sec += delay;
    rtc_set_alarm(&time, callback_rtc, "Wakeup alarm");

    /* Enter deep sleep mode */
    pm_set(mode);
}
```

- Setting an alarm to the RTC requires a callback function.

main.c

```
/**
 * Call-back function invoked when RTC alarm triggers wakeup.
 */
static void callback_rtc(void *arg)
{
    puts(arg);
}
```