Atmel 328-Processor for RaspberryPi

DHT22, DHT11

Gerhard Hepp, April 2015

Content

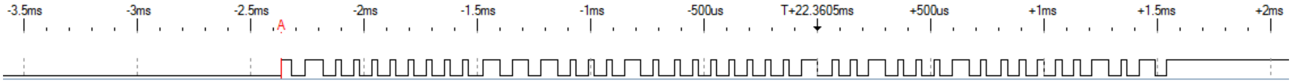
Overview

The temperature, humidity sensor DHT22 is a quite inexpensive sensor, well suited for microcontroller applications.
It is connected by a single wire, needs 5ms for a read cycle, but a quite challenging protocol where the pulse width gives '0' or '1' bit values.

This is a typical application for a coprocessor for raspberrypi. For an atmel328, it is not a challenge to handle this protocol.

Implementation

The protocol is quite simple:
The controller puts a '0' to the bus, min 1 ms. When the bus is free and pulled high by the resistor, the DHT22 answeres by a sequence of pulses.



The initial high to low-edge is not shown.
The transmission contains 40 bits.

Sample code for Raspberry Pi is provided in python. Integration to scratch is achieved with the scratchClient-Software from heppg.de.

The firmware needed can be programmed from the RPi, and the device can be easily implemented on a breadboard.

Disadvantage is the lengthy installation procedure. But with simple steps and verification points this is manageable.

The firmware for the controller is ready to use in the samples, no coding required in this area. As programming the firmware into the Atmel328 is done with the RPi, no extra programming device is needed.
The programming of the atmel device is no topic of this article.

Prerequisite is that the contoller is using the internal 8MHz RC oscillator. This is default for factory new devices.
If you get a preprogrammed device with e.g. a bootloader for arduino, the internal oscillator can be switched off. In this case you need to use a programming device, or attach an external clock signal. In the appendix, the procedure is described for this.

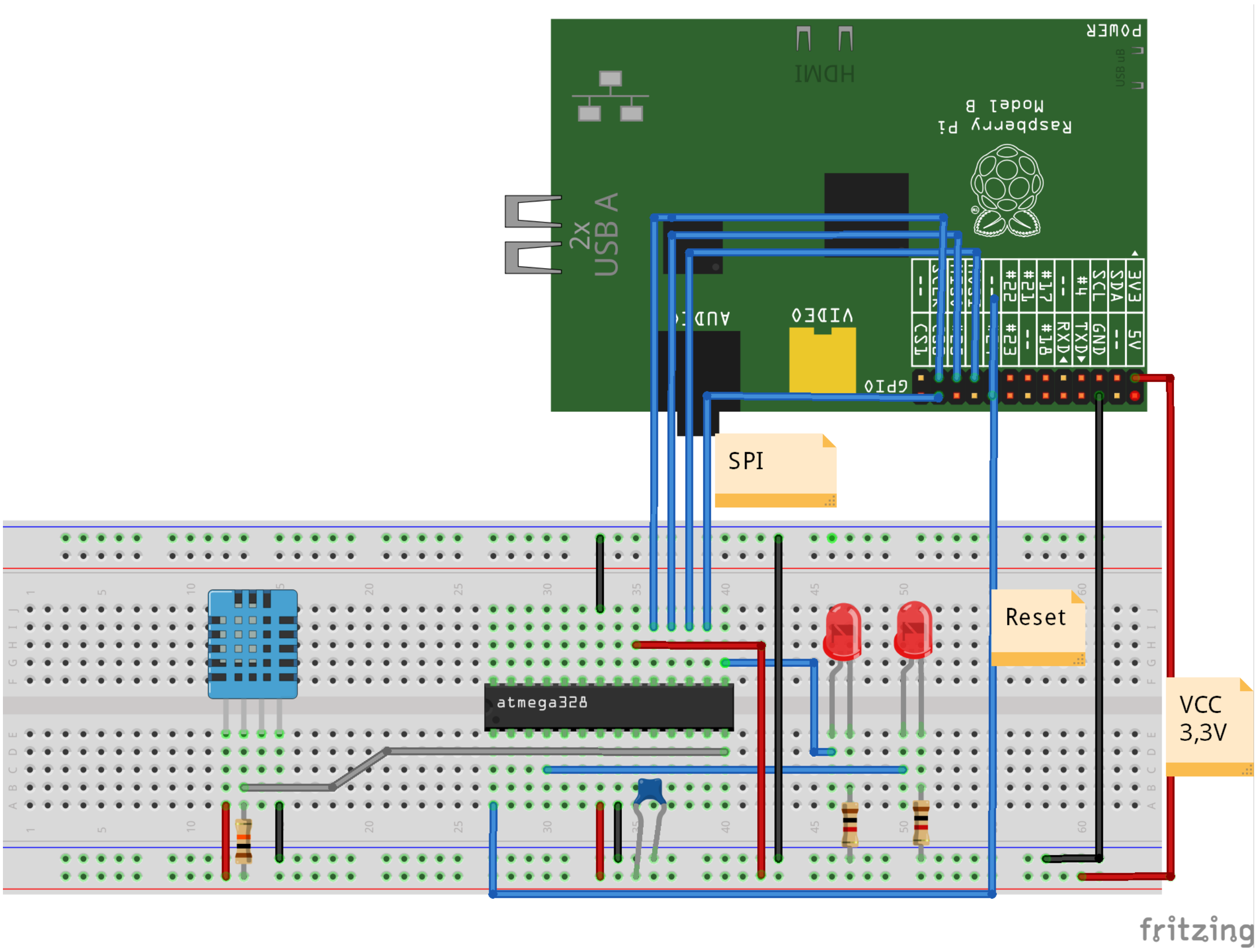
Setup

Parts list

- ATmega 328 P-PU (DIL-package)
- breadboard
- (optional) precision socket 28 pol, DIL 28 pol, 0.3 large. The socket is useful to protect the pins of the processor from bending. Set the socket into the breadboard and then insert the 328.

- LED, standard 20mA
- 1kOhm resistor for LED
- Capacitor 100nF, 10V min, ceramic (recommended; if not available it runs also without it)
- DHT22 / DHT11, up to four devices
- Resistor 10kOhm
- Patch cables m-f, 5 pieces and some extra to connect RPI and breadboard
- wires for breadboard

Of course you need a RPi.



Setup procedure

Power off Rpi
Insert controller in breadboard.

VCC of the controller is from 3.3V from RPi (black, red).
SPI-connections MISO, MOSI, SCK and SS an CS0 des RPI (blue).
RESET of controller Pin 1 to GPIO24 (blue)
LED with series resistor is from controller PB1, Pin 15 to GND..

Data input of the controller is PB0, pin 14.

Install software on Raspberry Pi

The procedures are for Raspbian distribution. You need root access to the system.

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install python-dev pip
sudo pip install spidev intelhex
```

Activate SPI driver. Can be done with raspi-config (enable SPI) permanently.
Or with

```
sudo modprobe  spi_bcm2708
```

which needs to be repeated after each reboot.

Copy the software archive to /home/pi and unpack.

```

tar xzvf  dht22_328.tar.gz
```

Verify hardware and programming software.

Read Atmega fuses

```
cd ~/dht22_328
sudo python src/program.py -rf
```

The output should read like this

```
PROGRAMMING_READ_CALIBRATION_BYTE   b0 10110000
PROGRAMMING_READ_EXTENDED_FUSE_BITS ff 11111111
   BODLEVEL0          1
   BODLEVEL1          1
   BODLEVEL2          1
PROGRAMMING_READ_FUSE_BITS        e2 11100010
   CKSEL0           0 ENABLED
   CKSEL1           1
   CKSEL2           0 ENABLED
   CKSEL3           0 ENABLED
   SUT0           0 ENABLED
   SUT1           1
   CKOUT            1
   CKDIV8          0
PROGRAMMING_READ_FUSE_HIGH_BITS     d9 11011001
   BOOTRST           1
   BOOTSZ0           0 ENABLED
   BOOTSZ1           0 ENABLED
   EESAVE           1
   WDTON            1
   SPIEN           0 ENABLED
   DWEN           1
   RSTDISBL          1
PROGRAMMING_READ_LOCK_BITS        ff 11111111

```

If there are error messages (e.g. device not in sync), then possibly there are wrong connections or the processor has fuses already programmed. Check wiring first.

Read current controller program

When output is correct, next step is to read out current flash program. Is empty on a new device, but either way useful to verify communication.

```
cd ~/dht22_328
sudo python src/program.py -r
```

Output is expected like this.

```
root@raspberrypi:/home/pi/dht22_328# python src/program.py -r
('read', 'out.hex')
programming_readCode
programming_enable
('PROGRAMMING_ENABLE', [172, 83, 0, 0])
(0, [255, 255, 83, 0])
programming_enable end
programming_disable
programming_readCode ende
ok
```

Flash 'blink' program

If successful, then load the first 'blink code' program into the controller. It will blink the LED.

```
cd ~/dht22_328
sudo python src/program.py -p 328/steckbrett_328_blink.hex
```

Takes a few seconds, and the LED should blink.

Program fuses to final settings

Everythink ok, then flash the fuses to have the 8MHz oscillator running.

```
cd ~/dht22_328
sudo python src/program.py -wf
```

Blinking will stop during the flash procedure, and restart noticeably faster, 5 times a second.
The controller has the internal clock divider disabled now, runs at 8MHz and wiring and software is ok.

Flash the firmware

Flash the firmware. It supports the various functions of the device.

```
cd ~/dht22_328
sudo python src/program.py -p 328/dht22_328.hex
```

Firmware Overview

The firmware features need to be enabled by configuration commands before using them.

These settings are not persistent, so after reset or reboot, these command need to be issued again.

For each Feature, there are SET_CONFIG and GET_CONFIG-commands.

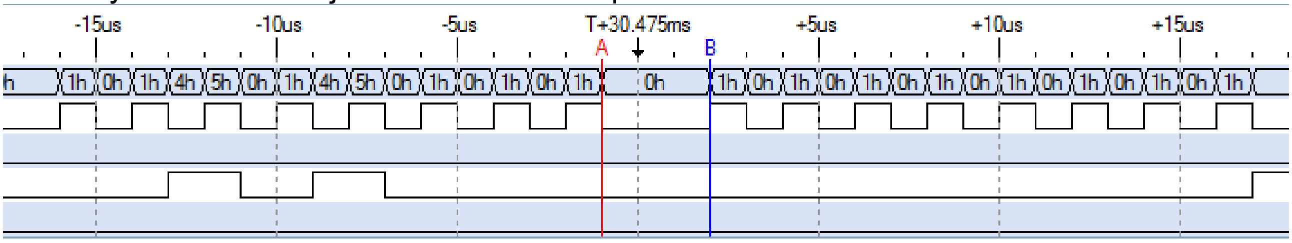
The LED-commands are available without prior activation.

After reset, the LED will blink 8 times.

A note on the commands for the controller. When a response from controller is expected, there is the need to shift the according number of dummy bytes into the SPI. These bytes are indicated by trailing '0' after the commands.

SPI clock speed is set to 240.000 Hz for the 8MHz version of the firmware. For this speed, the spidev library leaves enough time between the bytes for the interrupt program to provide responses. At a higher speed, there will be communication problems. Lower speed is no problem.

Communication on ATMEL is interrupt driven, programmed in assembler. The interrupt code is state based and was written using a custom code generator, combining the flexibility needed for adjustments with the speed of assembler.



The chart shows two byte transfers by SPI, 500kHz. The gap between the bytes is 3 us, at 8mHz CPU frequency these are 24 assembler instructions.

General Commands

Firmware, read version

GET_VERSION,0,0	0x4e	Controller responds with version number.

Version of the firmware is 0x41, minor version depends on release.

```
cd ~/dht22_328
sudo python src/test_get_version.py
```

Display should be '0x41', minor version is e.g. 0x05 or higher.

Firmware: LED-Commands

SET_LED_0_0	0x44	drive Port PB1 (Pin15) low.
SET_LED_0_1	0x48	drive Port PB1 (Pin15) high.
SET_LED_1_0	0x47	drive Port PD2 (Pin4) low.
SET_LED_1_1	0x49	drive Port PD2 (Pin4) high.

Test program.
Led blinking is controlled from the raspberry. Rythm is long low, short dark.

```
cd ~/dht22_328
sudo python src/test_blink.py
```

Led on, off

```
cd ~/dht22_328
sudo python src/test_led_on.py
sudo python src/test_led_off.py
```

Firmware: Read Sensor Values

There is one DHT11/ DHT22 supported.
The device is triggered (default) every 2 sec.

Data provided on SPI are
- data bytes 0 to 3 read from DHT22
- error byte 0x00 is no error.

GET_RESULT , 0,0,0,0,0	0x40	get device 0 results

The program reads 10 values and prints to console.

```
cd ~/dht22_328
sudo python src/test_get_result.py
```

Error codes

0x00	no error
0x01	no data aquired yet
0x02	checksum error
0x09 to 0x5c	each edge has its own error code. See dht22.cpp for details.

Firmware, auxiliary, manage delay values

SET_DELAY_SEC

SET_DELAY_SEC, 0	0x42	Write measurement delay. Values 2..255.

This value is the current delay time between measurements.

```
cd ~/dht22_328
sudo python src/test_set_delay.py
```

Display is initially 2 sec.

Connection to Scratch

Main purpose of this device is to interface with scratch. Download and install scratch client software from heppg.de.

Startup

```
cd ~/scratchClient
sudo python src/scratchClient.py -c config/config_dht22_atmel328.xml
```