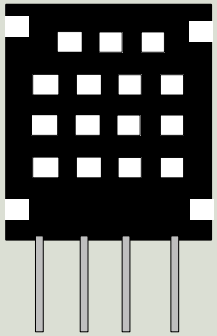
https://www.halvorsen.blog



Raspberry Pi and AM2320

Temperature and Humidity Sensor with I2C Interface

Hans-Petter Halvorsen

Contents

Introduction

Hans-Petter Halvorsen

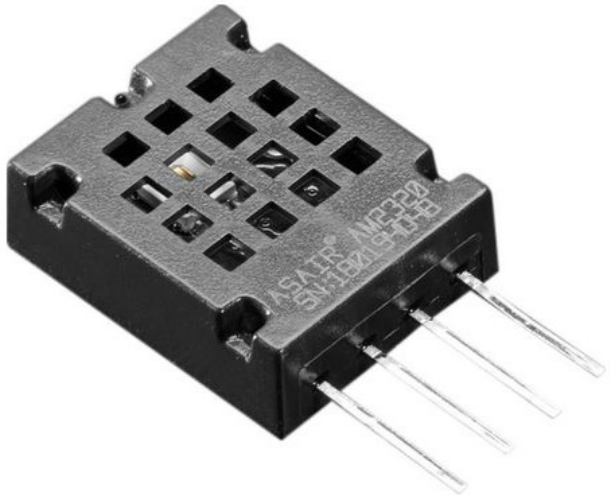
Table of Contents

Introduction

- Different AM2320 Python Libraries/Examples exists
  - These are typically available from https://pypi.org or https://github.com
- In this Tutorial Python Examples will be created from “scratch” by:
  - Reading the **Datasheet** carefully
  - Using the low-level **smbus** Python Library for I2C Communication

Introduction

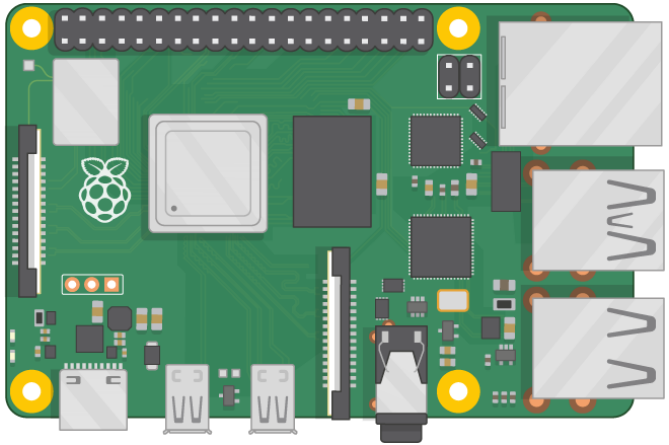
This Tutorial will demonstrate the use of a **AM2320** Temperature and Humidity Sensor in combination with **Raspberry Pi** and **Python**

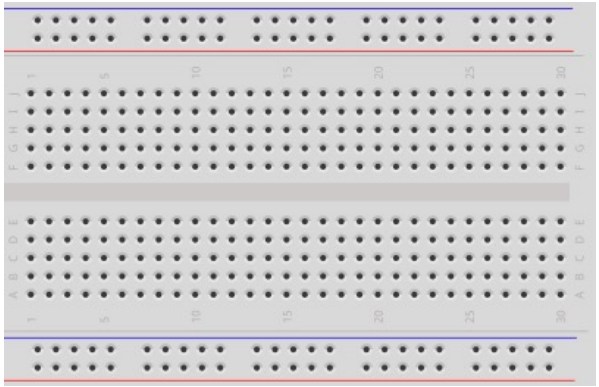


https://www.adafruit.com/product/3721

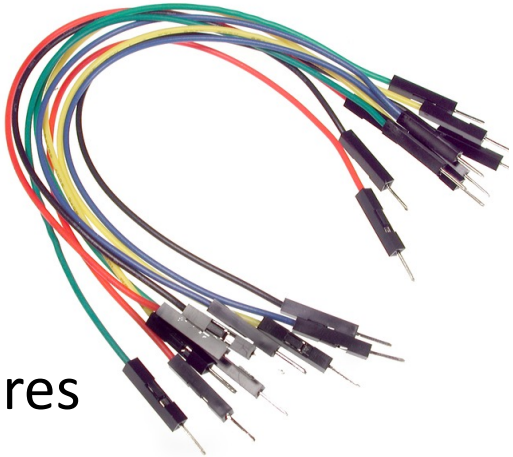
Hardware

Raspberry Pi



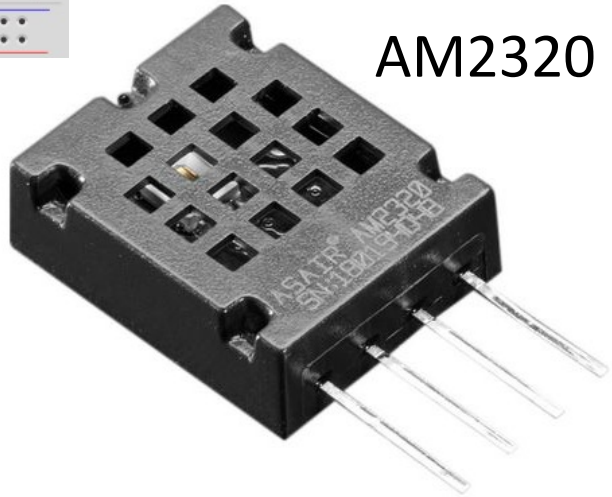


Breadboard



Wires

AM2320



AM2320

Temperature and Humidity Sensor

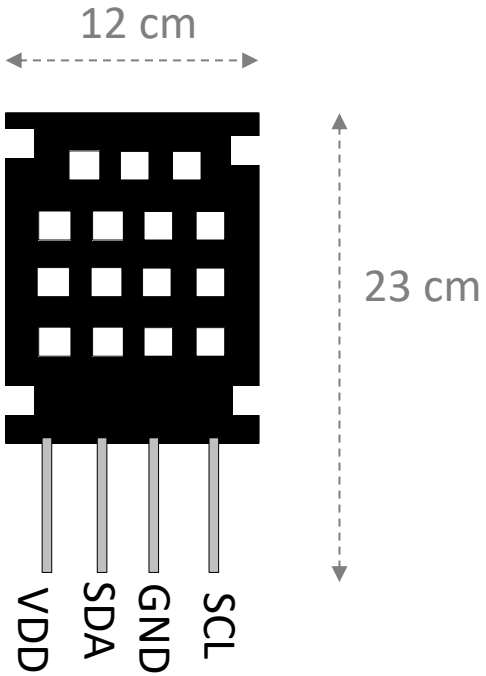
Hans-Petter Halvorsen

Table of Contents

AM2320 Sensor

- Temperature and Humidity Sensor
- **I2C** Interface
- Range: $-40°C$ to $+80°C$ and 0 to $100\%RH$
- Accuracy: Temperature ± 0.5℃ and Humidity $\pm 3\%RH$ according to the Datasheet
- Sampling Rate: $0.5Hz$, this means the minimum interval between readings is 2 seconds
- I2C address: **0x5C** (cannot be changed)
- Price: about $4
- Sensor Overview: https://learn.adafruit.com/adafruit-am2320-temperature-humidity-i2c-sensor
- Datasheet: https://cdn-shop.adafruit.com/product-files/3721/AM2320.pdf

AM2320 Sensor



I2C Interface

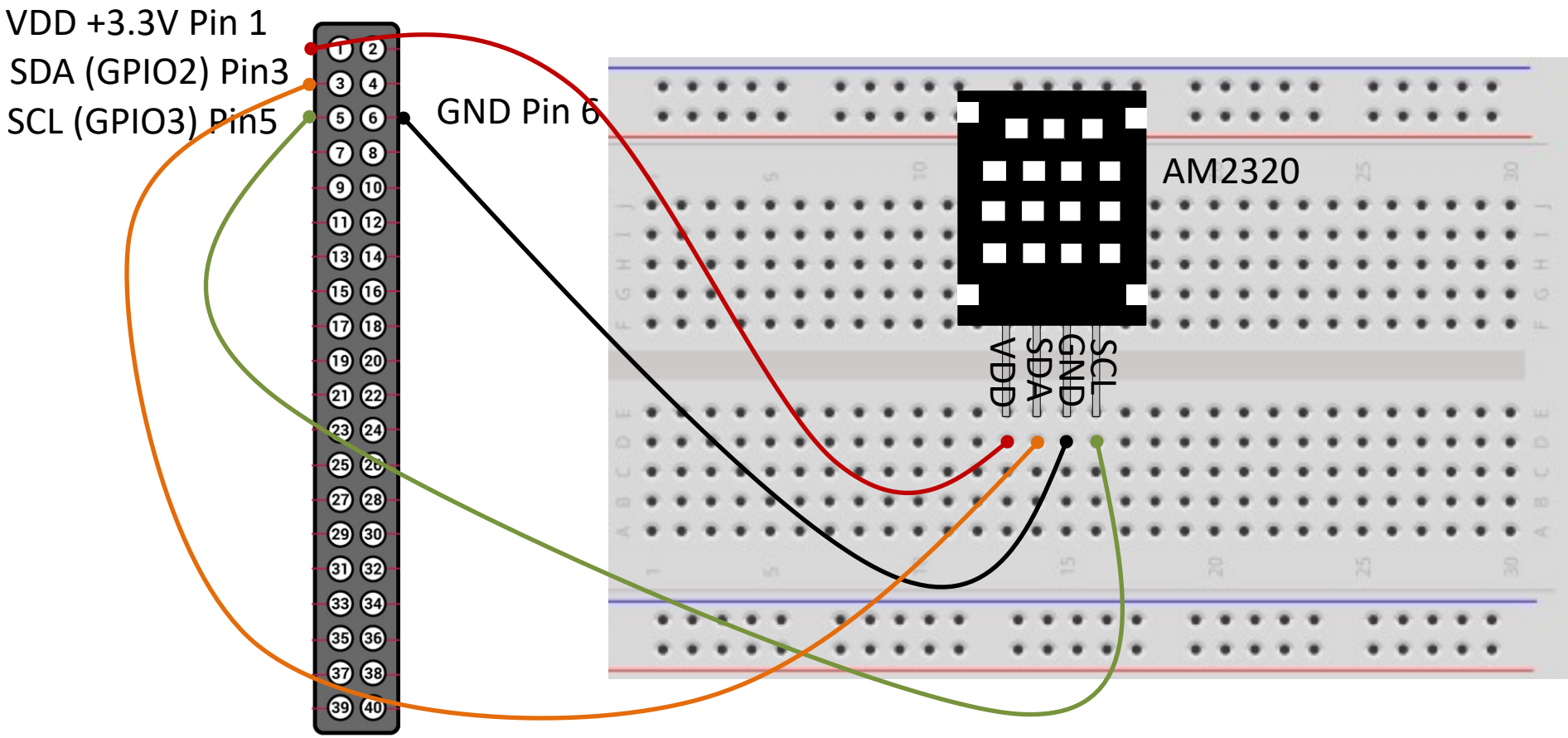
Pin Overview:

- **VDD** – Power, $3 - 5VDC$
- **SDA** - I2C data in/out, requires a pullup resistor of $2 - 10K\Omega$ to VDD
- **GND** - Ground
- **SCL** - I2C clock in, requires a pullup resistor of $2 - 10K\Omega$ to VDD

Note! The Raspberry Pi has built-in pull up resistors on SDA/SCL, so there is no need to add external pullup resistors

https://learn.adafruit.com/adafruit-am2320-temperature-humidity-i2c-sensor/pinouts

AM2320 Wiring



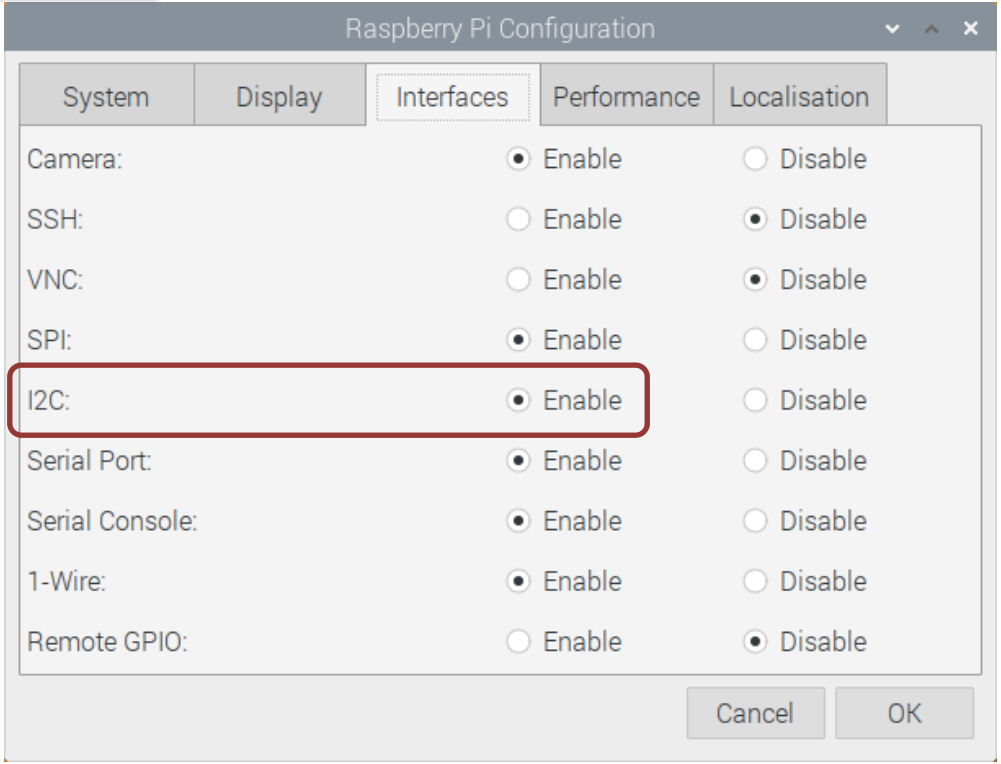
Raspberry Pi I2C Interface

Hans-Petter Halvorsen

Table of Contents

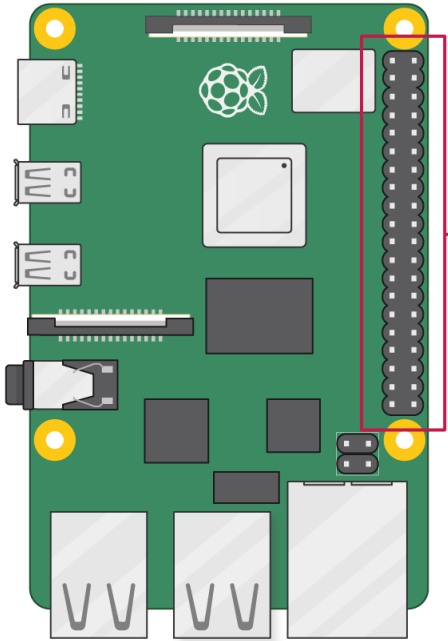
Access I2C on Raspberry Pi

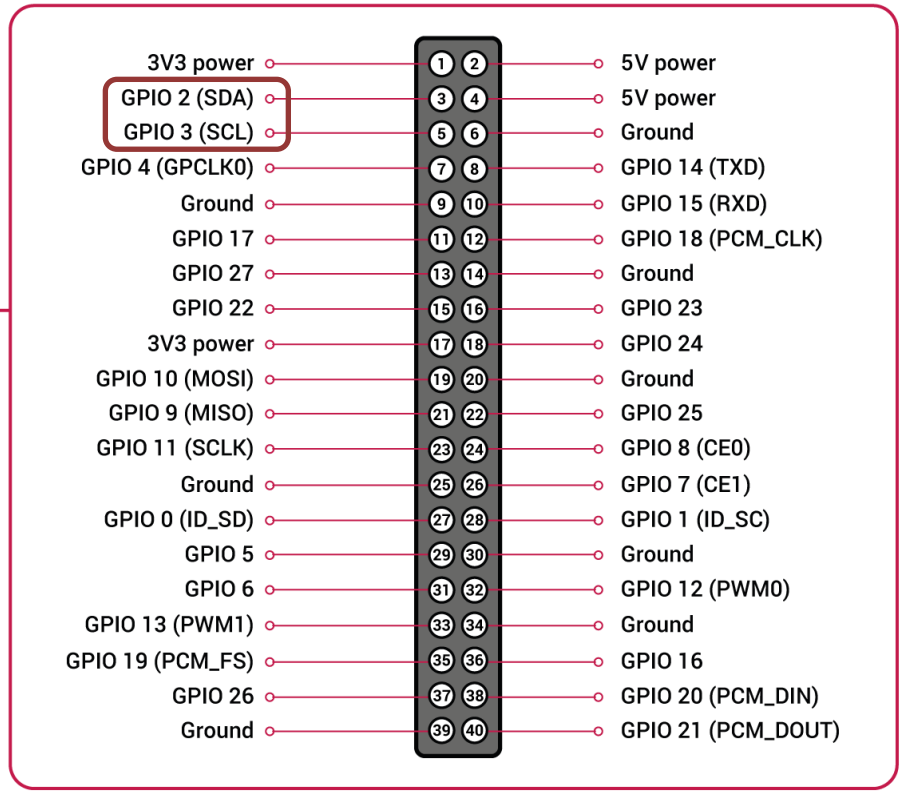
You need to Enable I2C on the Raspberry Pi



I2C Wiring on Raspberry Pi

GPIO 40 pins Connector





Note! The I2C pins include a fixed 1.8 kΩ pull-up resistor to 3.3v.

Detecting I2C Devices

Install I2C Tools on the Raspberry Pi:

```
sudo apt-get install -y i2c-tools
```

Detecting and Find the Address of the I2C Device using the `i2cdetect` command:

```
sudo i2cdetect -y 1
```

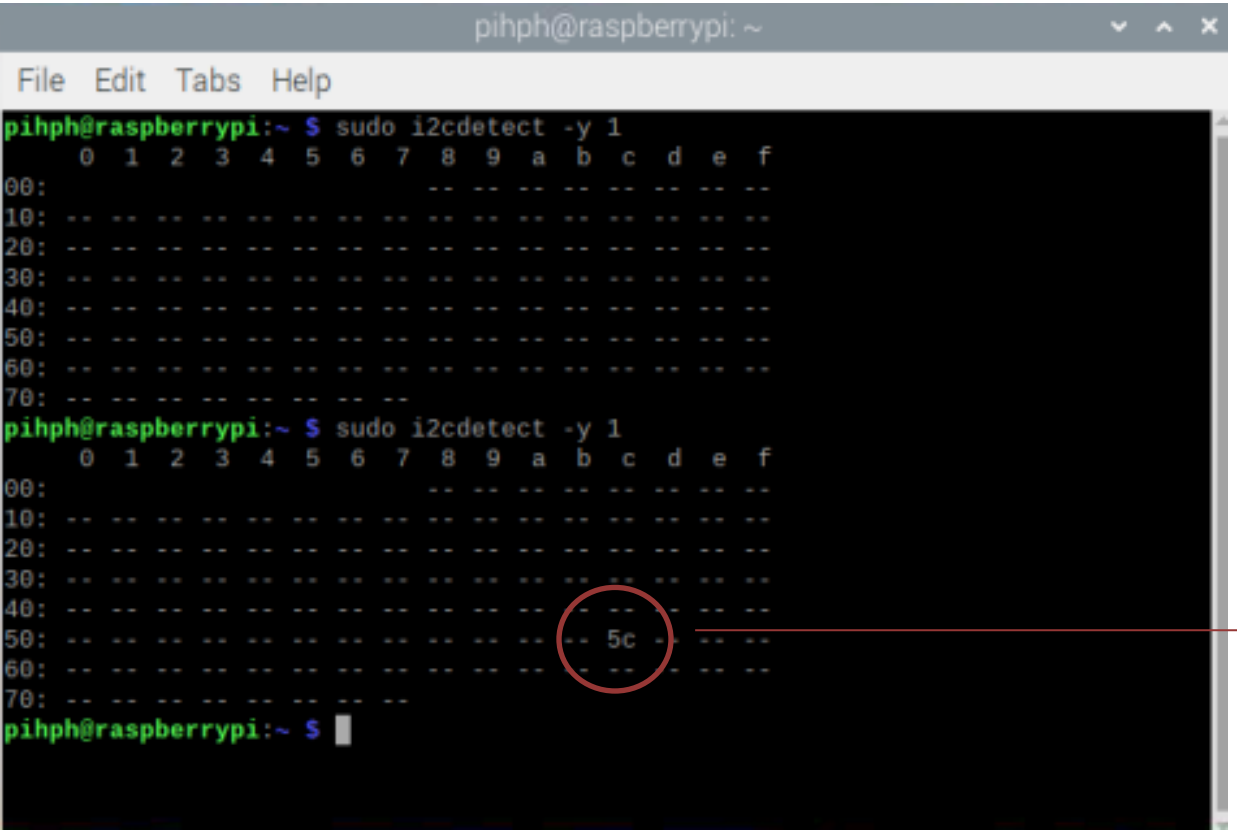
We can read and write its registers using `i2cget`, `i2cset` and `i2cdump`

Example:

```
sudo i2cget -y 1 0x5C
```

AM2320 Device Address

Detecting I2C Devices




```
sudo i2cdetect -y 1
```

Sometimes you need to run the command twice because the sensor goes into sleep mode

0x5C is the I2C address for the AM2320 Sensor

Python Examples

AM2320 Temperature and Humidity Sensor

Hans-Petter Halvorsen

Table of Contents

Python Examples

- Different AM2320 Python Libraries/Examples exists
- In this Tutorial Python Examples will be created from “scratch” by reading the Datasheet and using the low-level smbus Python Library for I2C Communication
- AM2320 Datasheet: https://cdn-shop.adafruit.com/product-files/3721/AM2320.pdf
- smbus: https://pypi.org/project/smbus/

smbus Python Library

SMBus (System Management Bus) is a subset from the I2C protocol

You can access I2C devices from Python using the smbus library:

```
import smbus
i2cbus = 1 #Default I2C Bus on Raspberry Pi
addr = 0x15 #am2320
bus = smbus.SMBus(i2cbus) # Initialize

#Write Data
bus.write_i2c_block_data(addr,cmd,vals[])
#Read Data
data = bus.read_i2c_block_data(addr,cmd)
```

https://pinout.xyz/pinout/i2c

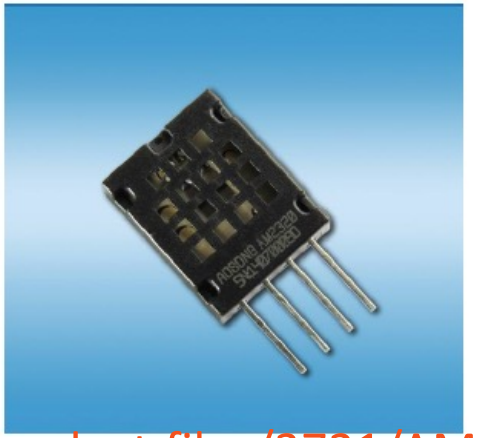
https://raspberry-projects.com/pi/programming-in-python/i2c-programming-in-python/using-the-i2c-interface-2

AM2320 Datasheet

AOSONG

Digital Temperature and Humidity Sensor

AM2320 Product Manual



AM2320 Datasheet: https://cdn-shop.adafruit.com/product-files/3721/AM2320.pdf

AM2320 Datasheet

Reader sample:			
Function	Function Code	Start address	Frame data content
Read the temperature and humidity	0x03	0x00	Send: (SLA+W) +0x03+0x00+0x04
			Return: 0x03 +0 x04 + humidity + high + low temperature and humidity high ~~temperature low~~ + CRC
Read the temperature	0x03	0x02	Send: (SLA+W) +0x03+0x02+0x02
			Return: 0x03+0x02+High temperature + low temperature+ CRC
Read humidity	0x03	0x00	Send: (SLA+W) +0x03+0x00+0x02
			Return: 0x03+0x02+High humidity+ Low humidity + CRC

In the Datasheet for the given sensor, you find all information you need. Here is some important excerpts

◎**Temperature output format**

Temperature resolution is 16Bit, temperature highest bit (Bit15) is equal to 1 indicates a negative temperature, the temperature highest bit (Bit15) is equal to 0 indicates a positive temperature; temperature in addition to the most significant bit (Bit14 ~ Bit0) indicates the temperature sensor string value. Temperature sensor value is a string of 10 times the actual temperature value.

AM2320 Datasheet: https://cdn-shop.adafruit.com/product-files/3721/AM2320.pdf

am2320sensor.py

```
import smbus
import time

i2cbus = 1 #Default
address = 0x5C #AM2020 I2C Address
bus = smbus.SMBus(i2cbus)

def WakeSensor():
  ..
def ReadTemperature():
  ..
def ReadHumidity():
  ..
def ReadTemperatureHumidity():
  ..
```

WakeSensor()

```
def WakeSensor():
    while True:
        try:
            bus.write_i2c_block_data(address, 0x00, [])
            break
        except IOError:
            pass

    time.sleep(0.003)
```

We just send an empty string to “wake up” the sensor from “sleep mode”

ReadTemperature()

```
def ReadTemperature():
    WakeSensor()
    while True:
        try:
            bus.write_i2c_block_data(address, 0x03, [0x02, 0x02])
            break
        except IOError:
            pass

    time.sleep(0.015)

    try:
        block = bus.read_i2c_block_data(address, 0, 4)
    except IOError:
        pass

    temperature = float(block[2] << 8 | block[3]) / 10
    return temperature
```

From Datasheet: Wait at least 1.5ms for result

Read Temperature Details

function code (0x03) | Start Address | Number of Registers (High and Low Temperature register)

```
bus.write_i2c_block_data(address, 0x03, [0x02, 0x02])
```

From Datasheet:

Read the temperature	0x03	0x02	Send: (SLA+W)+0x03+0x02+0x02
			Return: 0x03+0x02+High temperature + low temperature+ CRC

Blocks: [0][1] [2] [3]

```
block = bus.read_i2c_block_data(address, 0, 4)
```

Read 4 blocks

Each blocks is 8 bits

We put low + high part of Temperature value together:

```
temperature = float(block[2] << 8 | block[3]) / 10
```

<< is bitwise left shift operator

| is bitwise OR operator

From Datasheet: Temperature sensor value is a string of 10 times the actual temperature value

ReadHumidity()

```
def ReadHumidity():
    WakeSensor()
    while True:
        try:
            bus.write_i2c_block_data(address, 0x03, [0x00, 0x02])
            break
        except IOError:
            pass

    time.sleep(0.015)

    try:
        block = bus.read_i2c_block_data(address, 0, 4)
    except IOError:
        pass

    humidity = float(block[2] << 8 | block[3]) / 10
    return humidity
```

From Datasheet: Wait at least 1.5ms for result

Read Humidity Details

function code (0x03) | Start Address | Number of Registers (High and Low Humidity register)

```
bus.write_i2c_block_data(address, 0x03, [0x00, 0x02])
```

From Datasheet:

Read humidity	0x03	0x00	Send: (SLA+W)+0x03+0x00+0x02
			Return: 0x03+0x02+High humidity+ Low humidity + CRC

Blocks: [0][1] [2] [3]

```
block = bus.read_i2c_block_data(address, 0, 4)
```

Read **4** blocks

Each blocks is 8 bits

We put low + high part of Temperature value together:

```
temperature = float(block[2] << 8 | block[3]) / 10
```

<< is bitwise left shift operator

| is bitwise OR operator

From Datasheet: Humidity sensor value is a string of 10 times the actual humidity value

```
def ReadTemperatureHumidity():
    WakeSensor()
    while True:
        try:
            bus.write_i2c_block_data(address, 0x03, [0x00, 0x04])
            break
        except IOError:
            pass

    time.sleep(0.015)

    try:
        block = bus.read_i2c_block_data(address, 0, 6)
    except IOError:
        pass

    humidity = float(block[2] << 8 | block[3]) / 10
    temperature = float(block[4] << 8 | block[5]) / 10
    return temperature, humidity
```

This Function reads both Temperature and Humidity

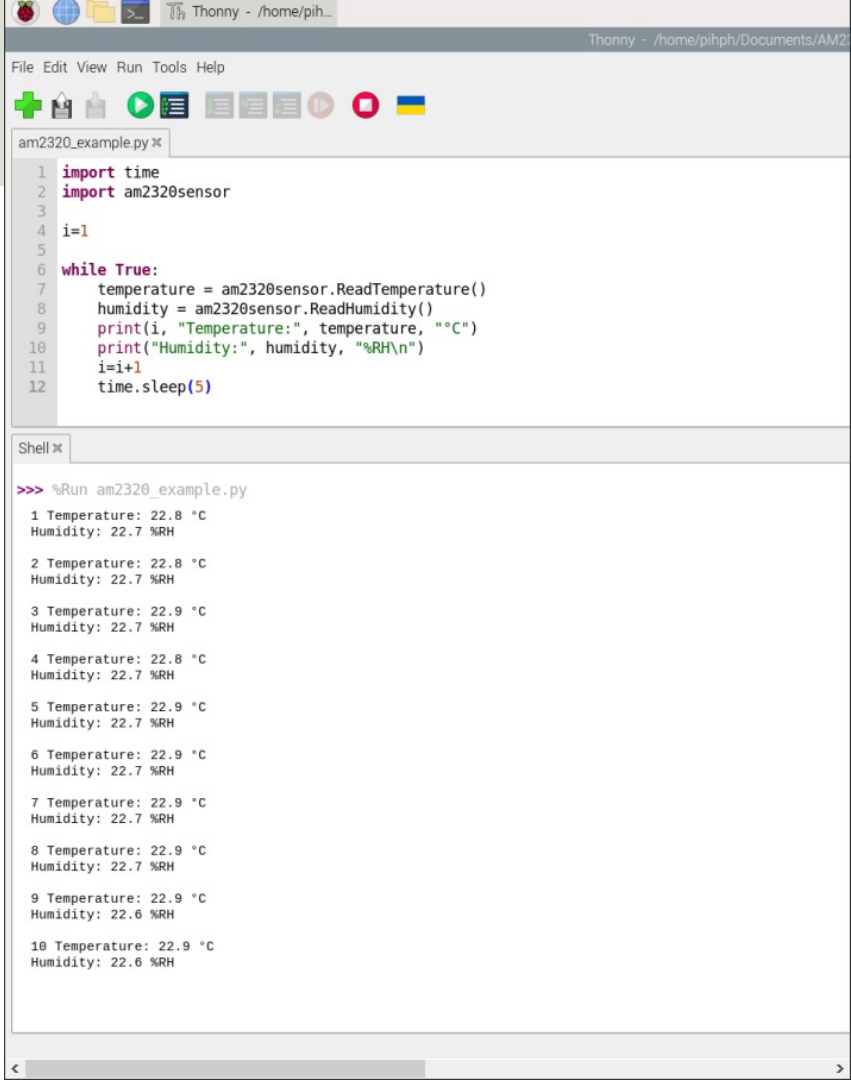
Python Code Example

```
import time
import am2320sensor

while True:
    temperature = am2320sensor.ReadTemperature()
    print(temperature)

    humidity = am2320sensor.ReadHumidity()
    print(humidity)

    time.sleep(5)
```

Improved Formatting

```
import time
import am2320sensor

i = 1
while True:
    temperature = am2320sensor.ReadTemperature()
    humidity = am2320sensor.ReadHumidity()
    print(i, "Temperature:", temperature, "°C")
    print("Humidity:", humidity, "%RH\n")
    i = i + 1
    time.sleep(5)
```

Results


Thonny - /home/pih...
Thonny - /home/pihph/Documents/AM2
File Edit View Run Tools Help
am2320_example.py
1 import time
2 import am2320sensor
3
4 i=1
5
6 while True:
7 temperature = am2320sensor.ReadTemperature()
8 humidity = am2320sensor.ReadHumidity()
9 print(i, "Temperature:", temperature, "°C")
10 print("Humidity:", humidity, "%RH\n")
11 i=i+1
12 time.sleep(5)
Shell
>>> %Run am2320_example.py
1 Temperature: 22.8 °C
Humidity: 22.7 %RH
2 Temperature: 22.8 °C
Humidity: 22.7 %RH
3 Temperature: 22.9 °C
Humidity: 22.7 %RH
4 Temperature: 22.8 °C
Humidity: 22.7 %RH
5 Temperature: 22.9 °C
Humidity: 22.7 %RH
6 Temperature: 22.9 °C
Humidity: 22.7 %RH
7 Temperature: 22.9 °C
Humidity: 22.7 %RH
8 Temperature: 22.9 °C
Humidity: 22.7 %RH
9 Temperature: 22.9 °C
Humidity: 22.6 %RH
10 Temperature: 22.9 °C
Humidity: 22.6 %RH

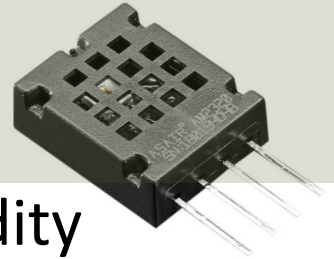
ReadTemperatureHumidity() Example

```
import time
import am2320sensor

i = 1
while True:
    t, h = am2320sensor.ReadTemperatureHumidity()

    print(i, "Temperature:", t, "°C")
    print("Humidity:", h, "%RH\n")

    i = i + 1
    time.sleep(5)
```

Discussions

- In this Tutorial Python Examples have bee created from “scratch” by reading the Datasheet and using the low-level smbus Python Library for I2C Communication
- It has been implemented as a Python Module with functions for reading Temperature and Humidity
- There are still several improvements to make
- No CRC check (error check code) has been implemented
- A Python Class and Library could have been made
- It could have been deployed to https://pypi.org to make it easy to install by using “pip install xxx” or from Thonny Python Editor (Tools -> Manage packages…)
- +++

Summary



- In this Tutorial an AM2320 Temperature and Humidity Sensor has been used in combination with Raspberry Pi
- Many different Python Libraries and Examples exists
  - These are typically available from https://pypi.org or https://github.com
- In this Tutorial Python Examples have been created from “scratch” by reading the Datasheet and using the low-level smbus Python Library for I2C Communication
  - It has been implemented as a Python Module with functions for reading Temperature and Humidity
  - So far it is **not** available from https://pypi.org or https://github.com
  - But you can download it for free from my Website/Blog

Resources

- AM2320 Sensor Overview: https://learn.adafruit.com/adafruit-am2320-temperature-humidity-i2c-sensor
- Datasheet: https://cdn-shop.adafruit.com/product-files/3721/AM2320.pdf
- CircuitPython: https://learn.adafruit.com/circuitpython-on-raspberrypi-linux
- Adafruit am2320 Library: https://docs.circuitpython.org/projects/am2320/en/latest/index.html
- Gozem/am2320: https://github.com/Gozem/am2320
- am2320-driver: https://pypi.org/project/am2320-driver/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog