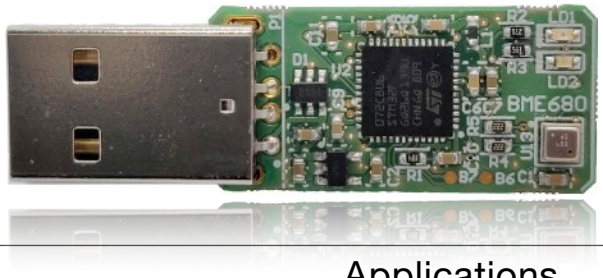
uThing::VOC™

Datasheet

Air Quality USB dongle with configurable output format



Features

- Measures ambient temperature, humidity, barometric pressure and VOC gas
- Direct indoor air quality (IAQ) index output
- Based on Bosch BME680 sensor
- Integrates Bosch proprietary Air-Quality Index algorithm (BSEC)
- USB Virtual Comm Port interface
- No special driver or software installation required (Linux, MacOS and Windows compatible)
- Configurable output formats (JSON, CSV and human readable) through command line
- Configurable sampling rate

Applications

- Quickly measure air-quality in a home, office or warehouse.
- BME680 sensor module evaluation
- Home automation control (e.g. HVAC)
- IoT rapid prototyping. Capture directly data into a CSV file without software installation
- Measure Air-Quality from a PC, gateway, RaspberryPi, etc.

Specifications

- Power input: 5V from USB port, 25 mA
- Operating range: -40-+85 °C, 10-90 %r.H., 300-1100 hPa
- IAQ Range: 0-500.
- Sensor-to-sensor IAQ deviation: ±15%
- Temperature resolution: 0.01 °C
- Absolute temperature accuracy @25 °C: ±0.5 °C
- Humidity accuracy tolerance: ±3% r.H.
- Humidity resolution: 0.008% r.H
- Dimensions: 48x15x5mm
- Weight: ~5 g
- Data rate: configurable between 3 seconds and 1 hour

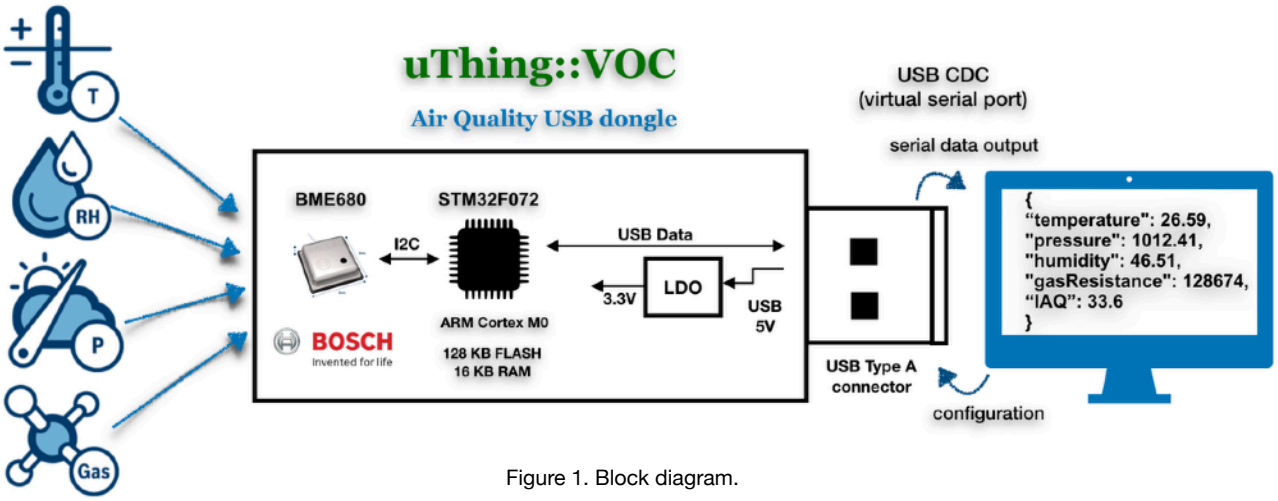


Figure 1. Block diagram.

Contents

1. Description

uThing::VOC™ is an Open Hardware USB dongle useful to evaluate or quickly integrate the Bosch BME680 air quality sensor.
The gas sensor within the BME680 detects a range of gases to measure indoor air quality (IAQ) index. Gases that can be detected include Volatile Organic Compounds (VOC) from paints (such as formaldehyde), lacquers, paint strippers, cleaning supplies, furnishings, office equipment, glues, adhesives and alcohol.

The integrated **STM32F072** MCU captures raw data from the sensors over the I2C port and uses the Bosch's proprietary Air Quality calculation algorithms (BSEC) to obtain equivalent air-quality, temperature, relative humidity and atmospheric pressure values. These values are then output over a simple Virtual Serial Comm on the USB connector in different configurable formats (JSON, CSV or Human readable).

Figure 1 shows the block diagram. An LDO regulator is used to convert the USB 5V power into 3.3V, used to power the MCU and **BME680** sensor. Two LEDs are used to display the status of the sampling process.
The PCB exposes a series of pins from the MCU into solderable pads (shown on figures 2 and 3). These pads can be used to expand the board functionalities by using either the USB or SWD ports to program a custom firmware image.

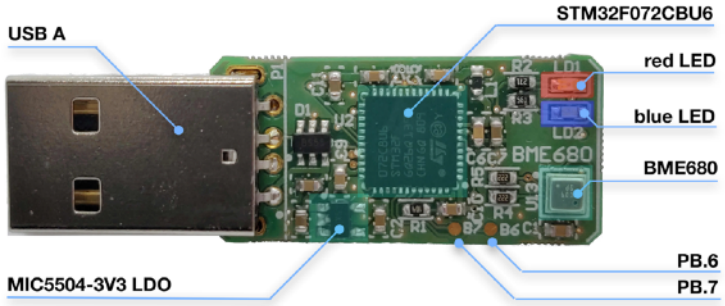


Figure 2. Front view

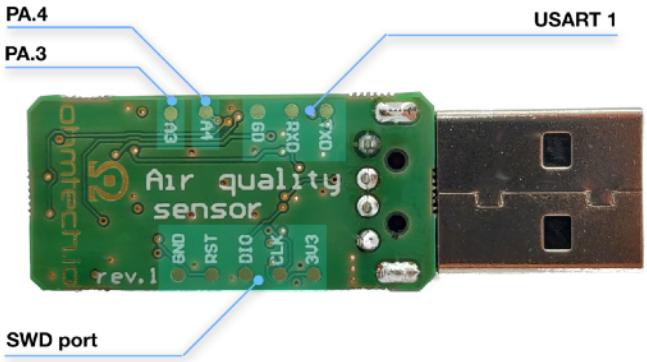


Figure 3. Back view

2. Compatibility

The device can be used with any USB capable host (laptop, embedded PCs, gateways, etc.). The firmware running in the *STM32F072* MCU integrates a standard **USB CDC**[1] (Communication Device Class). In particular, this application uses the **CDC-ACM** (Abstract Control Mode) to emulate a serial port. This implementation is also known as USB **VCP** (Virtual Communications Port).
The USB VCP interface is supported “out-of-the-box” in *Linux, MacOS* and *Windows* systems without the need of any extra driver installation since this driver comes integrated by default in those operating systems.

[1] The USB CDC is a generic composite class that can include more than one interface such as a custom control interface, data interface, audio, or mass storage related interfaces.

3. Bosch BSEC algorithm

The BME680 provides raw measurements of temperature, humidity, barometric pressure and VOC gas resistance over the I2C interface.
The **BSEC** (*Bosch Sensortec* Environmental Cluster) fusion library running on the MCU has been conceptualized to provide higher-level signal processing and fusion for the BME680. The library receives compensated sensor values from the sensor API and processes the BME680 signals in combination with optional additional sensors to provide the requested sensor outputs. The BSEC library then provides:

- Precise calculation of ambient air temperature outside the device[2]
- Precise calculation of ambient relative humidity outside the device
- Precise calculation of pressure outside the device
- Precise calculation of air quality (IAQ) level outside the device

Bosch uses the ISO16000-29 standard "Test methods for VOC detectors" to calibrate the BME680 sensors[3]. Besides ethanol (EtOH) as a target test gas, the sensors are also tested with **breath-VOC** (b-VOC), which composition is detailed in table 1.

Molar fraction	Compound	Production tolerance	Certified accuracy
5 ppm	Ethane	20 %	5 %
10 ppm	Isoprene /2-methyl-1,3 Butadiene	20 %	5 %
10 ppm	Ethanol	20 %	5 %
50 ppm	Acetone	20 %	5 %
15 ppm	Carbon Monoxide	10 %	2 %

Table 1. Breath-VOC testing gases composition

The BSEC algorithm outputs as a result of the combination of the input measurement the **Indoor Air Quality (IAQ)** value. This value is an index that can have values between 0 and 500 to indicate or quantify the quality of the air available in the surroundings[4].
Table 2 lists the IAQ classification and color coding:

IAQ Index	Air Quality
0 – 50	good[10]
51 – 100	average
101 – 150	little bad
151 – 200	bad
201 – 300	worse[2]
301 – 500	very bad

Table 2. Indoor air quality (IAQ) classification and color-coding.

The library also outputs an "Accuracy" indicator, which indicates the performance of the calculated IAQ, a value of 1 or more indicates that the sensor values are stable, providing a valid IAQ index.

[2] The BSEC library accepts a temperature offset input to account for the system self-heating (from other components in the PCB as the MCU, LDO, etc.). The uThing::VOC has been calibrated for ambient temperature while connected to a USB hub. If the dongle is plugged into a hotter system (PC, gateway, etc.) the output temperature is expected to be higher due to the heat transferred to the device.

[3] Please refer to the Bosch BME680 datasheet for more details.

[4] According to the guidelines issued by the German Federal Environmental Agency, exceeding 25 mg/m^3 of total VOC leads to headaches and further neurotoxic impact on health.

4. LEDs status indication

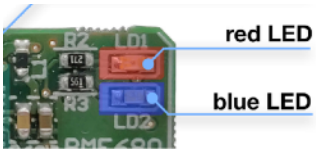


Figure 4. Status LEDs

The two LEDs on the top right side of the dongle are used to indicate the status of the BSEC algorithm and IAQ accuracy state:

- **Blue** LED **blinking** every 1 second: IAC Accuracy = 0
- **Blue** LED steady **ON**: IAC Accuracy = 1 (IAQ value is valid)
- **Red** LED short **blink**: BME680 data read
- **Red** LED steady **ON**: Error! Problem communicating with the sensor.

Note: The BSEC algorithm usually takes about 5 minutes to provide a valid IAQ value (Accuracy = 1).

5. Output formats

uThing::VOC provides three different output formats for the processed BSEC data. Check the configuration section below for instructions.

5.1. Output data units

The output sensor data is scaled in the following units:

- **Temperature**: Degrees celsius [°C]
- **Humidity**: percentage of the relative air humidity[5] [%]
- **Atmospheric pressure**: hectoPascal [hPa]
- **IAQ**: is a relative index, unit-less, range: 0-500
- **iaqAccuracy**: unit-less, indicate the sensor stability

5.2. Output formats

JSON [“J”]:
The JSON[6] format is the default output due to its common use on IoT applications. It’s also human readable. An output example is shown below.

```
{
        "temperature": 25.77,
        "pressure": 1005.78,
        "humidity": 33.20,
        "gasResistance": 278728,
        "IAQ": 26.8,
        "iaqAccuracy": 1
}
```

[5] Relative humidity refers to the ratio of the amount of moisture in the air at a certain temperature to the maximum amount of moisture that the air can retain at the same temperature. In other words, relative humidity measures how much of the moisture capacity of the air is used.

[6] JavaScript Object Notation (JSON) is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value).

CSV [“C”] (Comma Separated Values):
CSV is a plain text, comma delimited format. The values in one measurement are separated by the “,” comma symbol and the line break (CR/LF) character separates measurements[7]. The output follows the format below:

```
[temperature], [pressure], [humitidy], [gasResistance], [IAQ], [accuracy]
```

A measurement output example:

```
26.10, 1005.78, 32.62, 277787, 40.6, 1
```

Human readable [“M”]:
This option outputs a longer, but more user-friendly output which includes the measurement units. An example below:

```
Temperature: 26.17 C, Pressure: 1005.94 hPa, Humidity: 32.43 rH, Gas resistance: 277319
ohms, IAQ: 41.7, IAQ Accuracy: 1
```

6. Connecting to a Host

When the board is connected to a USB port (either directly or through a USB Hub), the *uThing::VOC*™ firmware enumerates itself as a CDC-ACM device, so the operating system automatically detects the VCP capability and loads the corresponding CDC-ACM driver.
The result of this is a new “Virtual Comm Port” ready on the Host PC.

6.1. Finding the VCP registered name

The exact name of the registered virtual device depends on the system, but below is a guideline to find the new device for different OS:

6.1.1. Linux

The exact name depends on the *Linux* distribution and number of VCP devices connected.
For instance, for Debian systems, then the first ACM device is connected, the device is registered as ***/dev/ttyACM0***.
Another common name is ***/dev/ttyUSBx*** (where x is the number of the VCP, starting with 0).

The device name can be found with the commands:

```
ls /dev/ttyACM*
```

or

```
ls /dev/ttyUSB*
```

depending on the distribution.

Note 1: In some Linux distributions, the system assumes by default that every new device enumerated as a VCP device is a modem. This is, the “ModemManager” service tries to capture the recently plugged device and send commands trying to configure it as a modem. In some situations, this could be an issue since the device will constantly receive strange commands. This can be avoided by adding a *udev* rule as described in this **post**. The *Vendor:Product ID* is **0483:5740**. If ModemManager is not needed (i.e. no external USB Modem will be used), this daemon can be disabled with "**sudo systemctl disable ModemManager.service**”.

Note 2: It was found that also in some distributions, the character devices (console) have the “echo” feature enabled by default. This is evident if the “cat /dev/ttyACM0” command is used to read the output before disabling the echo, the dongle will handle the echo received as false commands, resulting in undefined behaviour. To avoid this, “**stty -F /dev/ttyACM0 -echo**” can be used to disable the echo.

[7] The format is compliant with the *text/csv* MIME type (RFC4180)

6.1.2. MacOS

In *MacOS*, the exact name depends on the number of VCP devices connected.
For the 1st device the default name is ***/dev/cu.usbmodem1A1***

To find the device name use:

```
ls /dev/cu.usbmodem*
```

6.1.3. Windows

Open the **Device Manager** (Windows logo key and search for "device"). The VCP should be shown under *Ports (COM & LPT)* as a new USB Serial Device. The COM# number depends on the system.

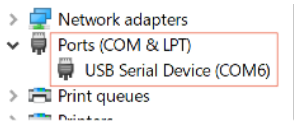


Figure 5. Virtual Serial Comm Device under Windows

6.2. Obtaining sensors' data

As soon as the device is plugged-in and the OS registers the VCP, the device automatically starts sending data over the virtual port with the default configuration of JSON format and sampling rate of 3 seconds.

There are basically 3 ways to interact with the Serial Port:

a) Interactively, using a GUI or command line terminal application[8]. Baud-rate and control lines are irrelevant since they aren't used in ACM, just open the VCP in the terminal and start interacting. Use the keyboard to explore options and change configuration.

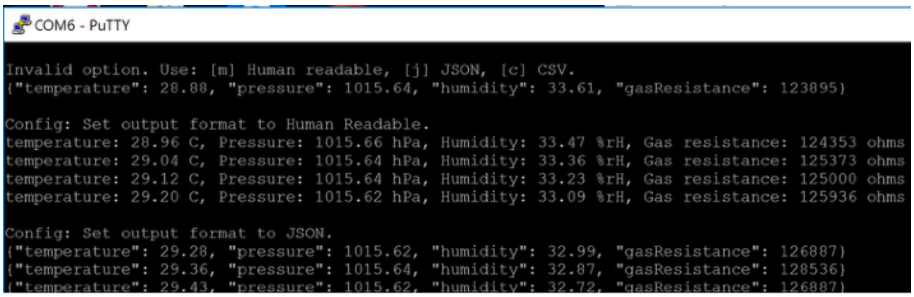


Figure 6. Viewing sensor output data with the Putty application.

b) Using the command line: No software installation is needed in Linux and MacOS (explained below).
c) Programatically: Most programming languages provide libraries that can be used to read and write data from a serial port.

6.3. Interacting with the device

In Linux and MacOS, the easiest way to start getting sensor data and changing the configuration, without installing any serial terminal application, is through the command line:
Open the Terminal[9] and find the device name . We will use ***/dev/cu.usbmodem1A1*** (default in MacOS) in our examples.

---

[8] Popular terminal applications for *Windows* are Putty, Teraterm and RealTerm. For *Linux and MacOS* the CLI apps *screen* and *minicom* are recommended, and as GUI based, some options are CoolTerm and Zoc for MacOS, and Cutecom for *Ubuntu.*

[9] Opening the terminal in MacOS *here*. In *Ubuntu* use "Ctrl+Alt +T".

6.3.1. Read sensor output

Simply by using the command:

```
cat /dev/cu.usbmodem1A1
```

will start printing the BSEC processed sensor output into the console every 3 seconds (default sampling rate).

> **Note:** If a Linux host is used, the console "echo" needs to be disabled (check notes on section 6.1.1)

6.3.2. Storing data into a file

Similarly, the output data can be sent to a plain text file with the following command:

```
cat /dev/cu.usbmodem1A1 > airQuality.log
```

> **Note:** if the "**&**" symbol is added at the end of the command, the process will be forked, and the system will keep storing the data until the user logs out or the dongle is disconnected.
> **Note 2:** If a Linux host is used, the console "echo" needs to be disabled (it's not needed in MacOS, check notes on section 6.1.1)

6.3.3. Sending a command to the dongle

For simplicity, all commands are 1 character long. Commands can be sent in an interactive way when a terminal application is being used (screen, minicom, Putty, etc.) just by pressing the correspondent key. The response is shown on the output.
As an alternative, directly from the command line[10], use the following command, where "***X****" is the desired command to send (listed on section 6).*

```
echo X > /dev/cu.usbmodem1A
```

> **Note:** For Linux users, it may be required to change the user's serial device permissions[11] in order to write in the serial port.

7. Configuration commands

In the actual firmware version (v1.0), two parameters can be configured: the data **sampling rate** (or more precisely, the data presentation rate[12]) and the **output format** previously described in section 5. Section 6.3 describes how to send commands.

As explained below, all the commands are a single character. The reception of commands is acknowledged on the output, so it's advised, if needed, to change the configuration **before** starting to collect data in order to simplify the data parsing or importing into an application.

7.1. Format configuration

The output format can be changed by sending the following characters:

- **"j" or "J"** for **JSON**
- **"c" or "C"** for **CSV**

---

[10] If the output is being printed already, a new terminal can be opened with "*cmd+n*" or "*cmd+t*".

[11] One way is by executing "sudo chmod 666 /dev/ttyACM0"

[12] The actual sensor sampling rate is **fixed** by the BSEC operation mode to 1/3 Hz (LP mode). The uThing::VOC firmware provides the option of downsampling this rate in order to reduce the amount of data to collect in applications where the storage or transmission bandwidth is constrained.

- **“m” or “M”** for **Human readable**

7.2. Sampling period

The data output period can be configured between 3 seconds (the minimum defined by the internal operation of the BSEC algorithm) and 1 hour with the following commands:

- **“1”: 3 seconds**
- **“2”: 10 seconds**
- **“3”: 30 seconds**
- **“4”: 1 minute**
- **“5”: 10 minutes**
- **“6”: 30 minutes**
- **“7”: 1 hour**

7.3. Extra commands

The **‘h’** command displays the available configuration commands as below:

```
-------------------------------------------------------
 Use:    [m] Human readable, [j] JSON, [c] CSV, [s] Show configuration,
 Period: [1] 3 sec, [2] 10 sec, [3] 3c, [4] 1 min, [5] 10 min, [6] 30 min, [7] 1 hour.
-------------------------------------------------------
```

Finally, the “**s**” command can be used to show the application status.
An example below:

```
-------------------------------------------------------
***  Status:
 Samplig period = 3 sec, Format = HUMAN, Uptime = 4787016 ms
-------------------------------------------------------
```

which shows the data output every 3 seconds in the long format, and the dongle has been plugged for approx. 1 hour 20 minutes (4787016 milliseconds).

PCB mark	Signal name	Comment
GND	**Ground**	System ground reference
RST	**nRESET**	Target reset, JTAG RESET line
DIO	**SWDIO**	corresponds to the JTAG TMS line
CLK	**SWCLK**	corresponds to the JTAG TCK line
3V3	**VCC/VTref**	Target reference voltage, it’s used by most debug probes to check target power and drive level converter references.

8. Flashing custom firmware

If a new firmware version is available, or in case the user wants to program their own application firmware on the MCU, there are 2 ways of re-programming the firmware:

8.1. Using the SWD interface

There are 5 pads at the bottom side of the PCB connected to the SWD (Single Wire Debugging)[13] interface of the STM32 MCU[14]:

Table 3. SWD interface

[13] SWD is a standard debugging/programming interface, a subset of the JTAG standard

[14] Various programming interfaces can be used (*Segger J-Link*, *ST-Link*-based or *FT2232*-based programmers, etc.)

This interface is more appropriate for firmware development due to the debug capabilities, as well as production firmware flashing with an appropriate spring-loaded test jig.

8.2. Using USB-DFU

If the intention is to simply update the firmware, or just program a slightly modified version without the need for extensive debugging / testing, the MCU can be re-flashed via the USB port by using the on-board USB-DFU[15] capability.
The programming process via USB requires two steps:

8.2.1. Booting the dongle into USB-DFU mode

The MCU has a special **Bootloader** in ROM memory, which provides the functionality of DFU. In order to start this process, the MCU has to be reset into this Bootloader mode. This is accomplished by powering-up the device while **holding** the **BOOT0** line (pin #44) into **logic-high** level (3.3V / VCC).

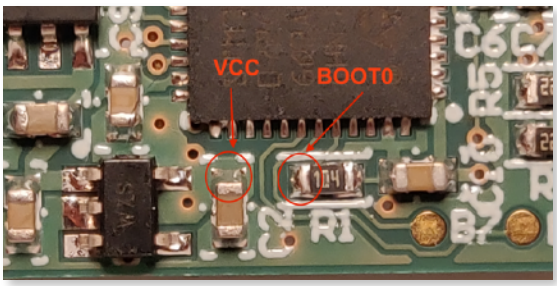


Figure 7. BOOT0 and VCC signals location.

To do this, unplug the dongle, hold a jumper (conductive material, like a piece of wire, screwdriver, paper-clip, etc.) between the VCC and BOOT0 pins with the precaution of not short-circuit any other pin, and plug the device into the USB while shorting these 2 pins. In this case, both status LEDs should stay OFF. The jumper can be then released.

To verify if the Bootloader enumerated the MCU this time as a USB-DFU capable device, use the command **`lsusb`**

This is the displayed information in *MacOS*:

```
Bus 020 Device 008: ID 0483:df11 STMicroelectronics STM32  BOOTLOADER  Serial: FFFFFFFEFFFF
```

In a *Debian* distribution:

```
Bus 001 Device 004: ID 0483:df11 STMicroelectronics STM Device in DFU Mode
```

In *Windows*, the *Device Manager* show a “STM Device in DFU Mode” under USB devices.

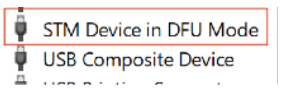


Figure 8. DFU-Bootloader recognised in Windows’ Device Manager

8.2.2. DFU-UTIL command

DFU-UTIL is a CLI application available for *MacOS, Linux* and *Windows.* In the Unix-based systems it’s available in the usual packet managers (Homebrew, apt-get, etc.). The version v0.9 is the latest one at the moment of writing and it’s been tested successfully on *Debian*[16] and *MacOS*.

---

[15] *Device Firmware Upgrade* (DFU) is a vendor- and device-independent mechanism for upgrading the firmware of USB devices with improved versions provided by their manufacturers.

[16] In *Linux*, the dfu-util needs to be run as root, or a udev rule and permissions should be added.

To perform the firmware programming, issue the following command, where “USBthingVOC.bin” is the application to flash in **.bin** format (no HEX or ELF supported):

```
dfu-util -a 0 -D USBthingVOC.bin --dfuse-address 0x0800C000 -d 0483:df11
```

9. Troubleshooting

9.1. Linux hosts

As mentioned in section 6.1.1, in some Linux distributions, the behaviour of the dongle could be undefined with the default configuration of the OS, more specifically, the “ModemManager” daemon and the console “echo” enabled by default in some distros can cause the dongle to output messages uncontrolled since the device interpret incoming characters as commands.

9.1.1. ModemManager

In some Linux distributions, the system assumes by default that every new device enumerated as a VCP device is a modem. This is, the “ModemManager” service tries to capture the recently plugged device and send commands trying to configure it as a modem. In some situations, this could be an issue since the device will constantly receive strange commands. This can be avoided by adding a *udev* rule as described in this **post**. The *Vendor:Product ID* is **0483:5740**. If ModemManager is not needed (i.e. no external USB Modem will be used), this daemon can be disabled with "**sudo systemctl disable ModemManager.service**”.

9.1.2. console “echo”

It was also found that in some distributions, the character devices (console) has the “echo” feature enabled by default. This is evident if the “**cat** /dev/ttyACM0” command is used to read the output before disabling the echo, the dongle will handle the echo received as false commands, resulting in undefined behaviour. To avoid this, “**stty -F /dev/ttyACM0 -echo**” can be used to disable the echo before reading the device with “cat”.

> **Note:** This is not an issue if a serial port application is used (screen, minicom, picocom) since they disable the echo by default. Or the port is opened by a library (c, python, java, etc.)

10. Schematic

10.1. Exposed pads

The board was designed with an Open Source philosophy in mind, therefore a few relevant pins are exposed on the pads illustrated in Figures 2 and 3. A user can solder wires to these pads and customise the **uThing::VOC** board to extend the default functionalities, or prototype new ideas.

Table 4 details the connection of the exposed pads and the alternative GPIO functions available[17]:

PCB mark	Pin #	Signal name	Comment
B6	**42**	**PB.6**	I2C1_SCL (a separate I2C bus to avoid any interfering with the BME680)
B7	**43**	**PB.7**	I2C1_SDA
A3	**13**	**PA.3**	exposes an ADC channel and a timer (PWM out or counter in)
A4	**14**	**PA.4**	mapped to an ADC, DAC & Timer channels
TXD	**30**	**PA.9**	USART1_TXD, TIM1_CH2 (PWM)
RXD	**31**	**PA.10**	USART1_RXD, TIM1_CH3 (PWM)
GD	**8,23,35,47**	**GND**	System reference ground

Table 4. Exposed pads description

10.2. Circuit Schematic

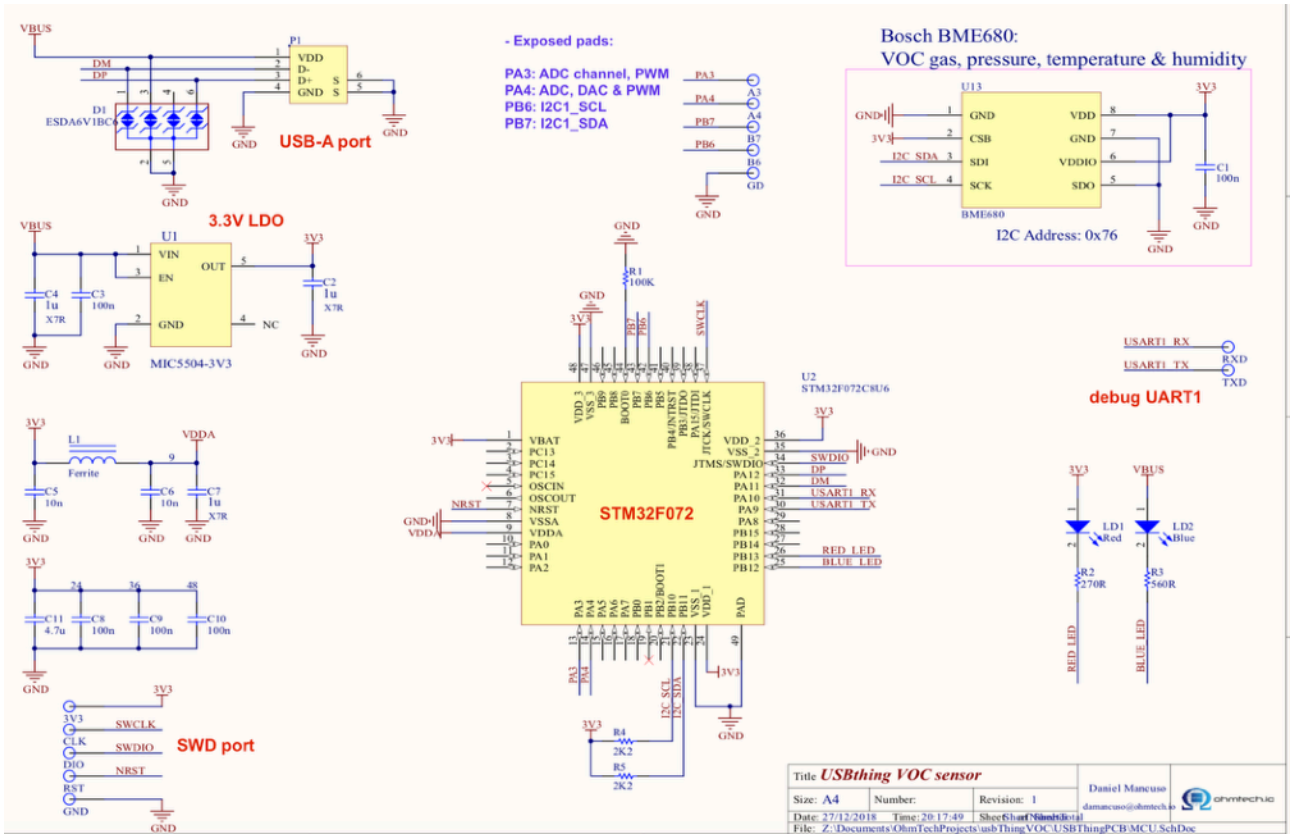


Figure 9. **uThing::VOC** revision 1 schematic

[17] Please refer to the STM32F072CB datasheet for extra information

11. Design files - source code

The design files (Gerbers, Bill of Materials, Schematics), along with the Firmware source code can be found in the project's GitHub repositories:

- Source code repository:
  https://github.com/ohmtech-io/uThingVOC

- Manufacturing files repository:
  https://github.com/ohmtech-io/uThingVOC-PCB.git

IMPORTANT NOTICE – PLEASE READ CAREFULLY

This USB dongle is provided "as is" without any guarantees or warranty. In association with the product, OhmTech.io makes no warranties of any kind, either express or implied, including but not limited to warranties of merchantability, fitness for a particular purpose, of title, or of noninfringement of third party rights. Use of the product by a user is at the user's risk. This product is no warranted against any manufactured defect from date of purchase. It is not fit for use in life-sustaining or security sensitive systems. Security sensitive systems are those for which a malfunction is expected to lead to bodily harm or significant property damage. OhmTech.io does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information.

NO REPRESENTATIONS OR WARRANTIES, EITHER EXPRESS OR IMPLIED, OF MERCHANTABILITY, FITNESS FOR A SPECIFIC PURPOSE, THE PRODUCTS TO WHICH THE INFORMATION MENTIONS MAY BE USED WITHOUT INFRINGING THE INTELLECTUAL PROPERTY RIGHTS OF OTHERS, OR OF ANY OTHER NATURE ARE MADE WITH RESPECT TO INFORMATION OR THE PRODUCT TO WHICH INFORMATION MENTIONS. IN NO CASE SHALL THE INFORMATION BE CONSIDERED A PART OF OUR TERMS AND CONDITIONS OF SALE."

Modifications reserved
Preliminary - specifications subject to change without notice.