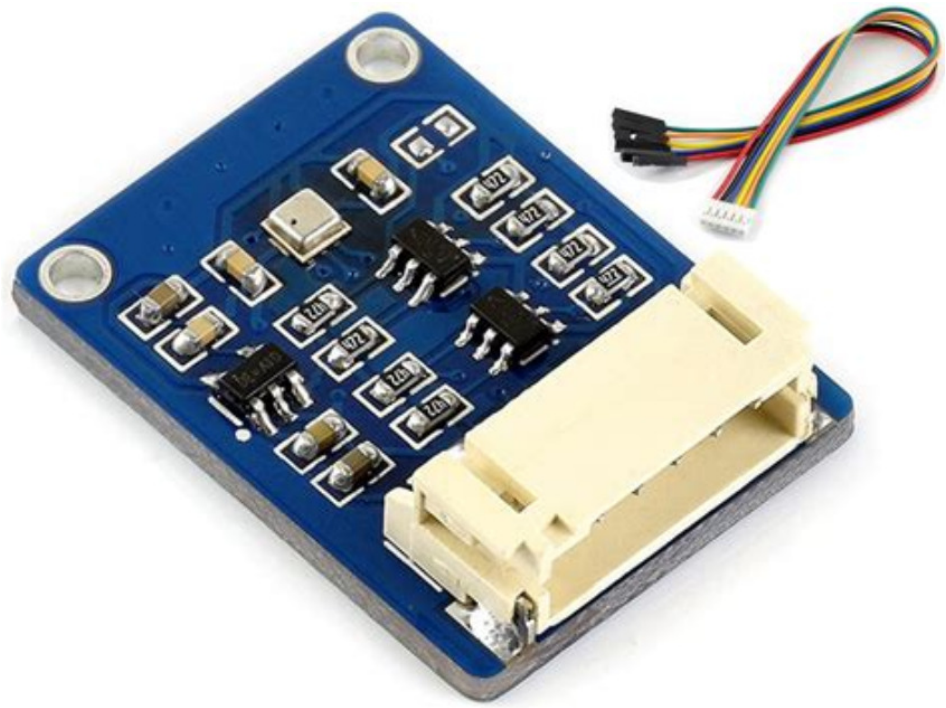
Pico-BME280-Module



Firmware Version 1.00
User Guide
Updated May 20th, 2025

Add-on C-Language module to merge with your existing (C-Language) project to support a BME280 sensor, using I2C communication protocol.

IMPORTANT :

This User Guide is about Pico-BME280-Module Firmware Version 1.00 from Andre St. Louys. To get the full potential of this module, you must NOT simply clone the GitHub repository. Read this User Guide and carefully follow the indicated procedure.

Join our Pico-BME280-Module discussion group on:
https://github.com/astlouys/Pico-BME280-Module/discussions

Pico-BME280-Module User Guide

Index

Introduction

The Pico-BME280-Module is a C-Language add-on module that will allow your existing C-language Pico program or project to use a BME280 sensor to read ambient temperature, humidity and atmospheric pressure.

Be aware that some companies announce their product as a BME280, whereas what they sell is a BMP280. The main difference is that the BMP280 does not read relative humidity. The unit that I bought from Waveshare can be seen from the front page. It is identified on the back of the small PC board as a BME280 and can read the three parameters (temperature, relative humidity and atmospheric pressure).

To help you figure out the details on how to use the Pico-BME280-Module, an example is included in the repository: « Pico-BME280-Example ». This is a basic C-Language program making use of the Pico-BME280-Module. This simple program is also described later in this User Guide.

To get the full potential of the Pico-BME280-Module, make sure you carefully follow the instructions given in this User Guide.

1) The « Setup part 1 » covers the steps required to install the Pico-BME280-Example to your system. Make sure everything is installed as required and that you can build and run the program without problem. You can them go on with the following step to merge the add-on module to your own program.

2) The « Setup part 2 » covers the steps to follow when you want to add the Pico-BME280-Module to one of your own existing C-Language program.

You can join other users of the Pico-BME280-Module on the discussion group on:
https://github.com/astlouys/Pico-BME280-Module/discussions
I can also be reached with my personal email address at:
astlouys@gmail.com

Pico-BME280-Module setup

Setup part 1: Installing / copying the Pico-BME280-Module to your system.

NOTE: The instructions below assume that your development system is a Raspberry Pi running the Raspberry Pi O.S. (previously called "Raspbian") and that you're using Visual Studio Code. It also assumes that you followed the recommended naming and structure conventions for the directory structure on your development system. If this is not the case, you will have to adapt the instructions accordingly.

1) Create the project directory on your system:
   ```
   mkdir  /home/pi/pico/Pico-BME280-Module
   ```

2) Clone / copy all files from the GitHub repository to the « /home/pi/pico/Pico-BME280-Module » project directory created in step 1 above.

3) While in the « /home/pi/pico/Pico-BME280-Module » project directory create a symbolic link pointing to the pico_sdk_import.cmake file. This way, if / when you update the Pico SDK, your project will always use the latest version when it is built:
   ```
   ln  -s  /home/pi/pico/pico-sdk/external/pico_sdk_import.cmake
   ```

4) Create a « UTILITIES » directory in the pico directory and move the file « baseline.h » from the Pico-BME280i-Module project directory to the « /home/pi/pico/UTILITIES » directory. The baseline.h file contains basic definitions that I use in most projects. You may want to take a look at it and add your own pieces of code that you use often. When you update the « baseline.h » file in the UTILITIES directory, it is automatically updated in all your projects through the symbolic link.

5) Create a symbolic link from the project directory to the « baseline.h » file:
   ```
   ln  -s  /home/pi/pico/UTILITIES/baseline.h
   ```

6) This completes the first part of the setup. At this point, make sure you are able to build and use the Pico-BME280-Example application. Refer to the section of this User Guide on how to use it.

Setup part 2: Integrating the Pico-BME280-Module to one of your projects.

NOTE: The instructions below assume that you development system is a Raspberry Pi running the Raspberry Pi O.S. (previously called « Raspbian ») and that you're using Visual Studio Code. It also assumes that you followed the recommended naming and structure conventions for the

directory structure on your development system. If this is not the case, you will have to adapt the instructions accordingly.

Make sure you carefully followed the instructions in the « Setup part 1 » above, since they are a prerequisite for this section.

1) From your project directory: « /home/pi/pico/Pico-My-Project », create a symbolic link pointing to both the « .c » and « .h » files of the Pico-BME280-Module:
```
ln  -s  /home/pi/pico/Pico-BME280-Module/Pico-BME280-Module.c
ln  -s  /home/pi/pico/Pico-BME280-Module/Pico-BME280-Module.h
```

2) From your project directory: « /home/pi/pico/Pico-My-Project », create a symbolic link pointing to the « baseline.h » file:
```
ln  -s  /home/pi/pico/UTILITIES/baseline.h
```

3) In your project file: « /home/pi/pico/Pico-My-Project.c », add the include file « Pico-BME280-Module.h » file:
```
#include "Pico-BME280-Module.h"
```

4) In your project file: « /home/pi/pico/Pico-My-Project.c », declare the structure defined in the Pico-BME280-Module.h file:
```
struct  struct_bme280  StructBME280;
```

5) You have to set a few definitions in the « Pico-NTP-Module.h » file as indicated in the next step.

6) You must define the GPIOs used to communicate with the BME280 sensor. The Pico-BME280-Module uses the I2C communication protocol. It can be seen as a « drop address » kind of protocol. So, as long as the devices have different addresses, they can coexist on the same communication line. The lines « #define SDA » and « #define SCL » must be set for the GPIOs you want to use (respectively « data » and « clock » line), as well as the I2C_PORT. I did use the same GPIOs than those used in the Pico-RGB-Matrix for the communication with the DS3231 real time clock IC.

7) Depending on your program, you may also want to copy-paste other parts / functions of « Pico-BME280-Example.c » to your own program to use them: « get_pico_unique_id() », « input_string() », « log_info() », « util_to_lower() »... Those functions are used in « Pico-BME280-Module », so either you copy-paste them to your own program, either you get rid of / replace them in the Pico-BME280-Module.

Pico-BME280-Example

The « Pico-BME280-Example » application is very basic... Its main purpose is to show how to integrate the « Pico-BME280-Module » to your current project. Nonetheless, here is how to use it.

Build the Pico-BME280-Example application and download it to your Pico. The application will automatically be launched when uploaded to the Pico.

Once started, the application gives you some time to start a terminal emulator program and make a connection with the Pico through the USB port. Since the purpose of the application is to read and display the BME280 weather parameters (temperature, humidity and atmospheric pressure) to a terminal emulator, it is not very useful if you do not connect such a terminal.

Once this is done, you will see a few messages over the screen as the Firmware executes and see what is going on.

The Firmware finally ends up in displaying the temperature (in both Celsius and Fahrenheit), the relative humidity (in percentage), and the atmospheric pressure. It will then wait for en entry on the keyboard. You may press « Enter » to loop back and proceed with another BME280 read cycle, or press « ESC » (followed by « Enter ») to switch the Pico in upload mode (in case you want to modify and try changes in the code).

If you have either a Pico-Green-Clock and / or a Pico-RGB-Matrix, a BME280 sensor with a standalone PicoW could be used to « broadcast » (« transmit ») the weather parameters to those devices. I'm currently working on a Smart Home system based on the Pico-RGB-Matrix. If there is interest among GitHub users, I could eventually publish an updated Firmware supporting the MQTT protocol (Message Queuing Telemetry Transport), along with an ecosystem based on peripherals working together...