finitespace / **BME280**

Provides an Arduino library for reading and interpreting Bosch BME280 data over I2C, SPI or Sw SPI.

246 commits **7** branches **14** releases **15** contributors GPL-3.0

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

finitespace Merge pull request #65 from coelner/heatindex … Latest commit `40fdb74` 2 days ago

docs	generic optimized formula (save ram and execution time). added explan…	13 days ago
examples	generic optimized formula (save ram and execution time). added explan…	13 days ago
src	generic optimized formula (save ram and execution time). added explan…	13 days ago
LICENSE	Moving license for github recognition.	3 months ago
keywords.txt	Fixing compilation. Fixing documentation. Fixing naming. Reorg abs al…	3 months ago
library.properties	Updating…	4 months ago

README.md

BME280

Provides an Arduino library for reading and interpreting Bosch BME280 data over I2C, SPI or Sw SPI. Additional environment calculation functions are provided. ESP and BRZO are now supported.

Table of Contents

Summary

Reads temperature, humidity, and pressure. Calculates altitude, equivalent sea level pressure and dew point. Provides functions for english and metric. Also reads pressure in Pa, hPa, inHg, atm, bar, torr, N/m^2 and psi.

Installation

To use this library download the zip file, decompress it to a folder named BME280. Move the folder to {Arduino Path}/libraries.

Usage

Include the library at the top of your Arduino script. `#include <BME280>` Create a global or local variable. `BME280 bme` In your start up call `bme.begin()`. Read the temperature, humidity, pressure, altitude and/or dew point.

`float pres, temp, hum` `bme.read(pres, temp, hum)`

or

`temp = bme.temp()` `hum = bme.hum()` `pres = bme.pres()`

Enumerations

TempUnit Enum

- TempUnit_Celsius
- TempUnit_Fahrenheit

PresUnit Enum

- PresUnit_Pa
- PresUnit_hPa
- PresUnit_inHg
- PresUnit_atm
- PresUnit_bar
- PresUnit_torr
- PresUnit_psi

OSR Enum

- OSR_Off
- OSR_X1
- OSR_X2
- OSR_X4
- OSR_X8
- OSR_X16

Mode Enum

- Mode_Sleep
- Mode_Forced
- Mode_Normal

StandbyTime Enum

- StandbyTime_500us
- StandbyTime_62500us
- StandbyTime_125ms
- StandbyTime_250ms
- StandbyTime_50ms
- StandbyTime_1000ms
- StandbyTime_10ms
- StandbyTime_20ms

Filter Enum

- Filter_Off
- Filter_1
- Filter_2
- Filter_4
- Filter_8
- Filter_16

ChipModel Enum

- ChipModel_Unknown
- ChipModel_BME280
- ChipModel_BMP280

Settings

BME280::Settings Struct

```
  * Temperature Oversampling Rate (tempOSR): OSR Enum, default = OSR_X1

  * Humidity Oversampling Rate (humOSR): OSR Enum, default = OSR_X1

  * Pressure Oversampling Rate (presOSR): OSR Enum, default = OSR_X1

  * Mode (mode): Mode Enum, default = Mode_Forced

  * Standby Time (standbyTime): StandbyTime Enum, default = StandbyTime_1000ms

  * Filter (filter): Filter Enum, default = Filter_16

  * SPI Enable: SpiEnable Enum, default = false
    values: true = enable, false = disable
```

BME280I2C::Settings Struct

- Includes all fields in BME280 settings.

```
 * BME 280 Address (bme280Addr): uint8_t, default = 0x76
```

BME280Spi::Settings Struct

- Includes all fields in BME280 settings.

```
 * SPI Chip Select Pin (spiCsPin): uint8_t
   values: Any pin 0-31
```

BME280Spi::Settings Struct

- Includes all fields in BME280 settings.

```
  * SPI Chip Select Pin (spiCsPin): uint8_t
    values: Any pin 0-31

  * SPI Master Out Slave In Pin (spiMosiPin): uint8_t
    values: Any pin 0-31

  * SPI Master In Slave Out Pin (spiMisoPin): uint8_t
    values: Any pin 0-31

  * SPI Serial Clock Pin (spiSckPin): uint8_t
    values: Any pin 0-31
```

Methods

BME280I2C(const BME280I2C::Settings& settings)

Constructor used to create the I2C Bme class. All parameters have default values.

BME280Spi(const BME280Spi::Settings& settings)

Constructor used to create the Spi Bme class. All parameters have default values except chip select.

BME280SpiSw(const BME280SpiSw::Settings& settings)

Constructor used to create the software Spi Bme class. All parameters have default values except chip select, mosi, miso and sck.

bool begin()

Method used at start up to initialize the class. Starts the I2C or SPI interface. Can be called again to re-initialize the mode settings.

```
 * return: bool, true = success, false = failure (no device found)
```

void setSettings(const Settings& settings)

Method to set the sensor settings.

const Settings& getSettings() const

Method to get the sensor settings.

float temp(TempUnit unit)

Read the temperature from the BME280 and return a float.

```
   return: float = temperature

   * unit: tempUnit, default = TempUnit_Celsius
```

float pres(PresUnit unit)

Read the pressure from the BME280 and return a float with the specified unit.

```
   return: float = pressure

   * presUnit: uint8_t, default = PresUnit_hPa
```

float hum()

Read the humidity from the BME280 and return a percentage as a float.

```
* return: float = percent relative humidity
```

void read(float& pressure, float& temp, float& humidity, TempUnit tempUnit, PresUnit presUnit)

Read the data from the BME280 with the specified units.

```
return: None, however, pressure, temp and humidity are changed.

* Pressure: float, reference
  values: reference to storage float for pressure

* Temperature: float, reference
  values: reference to storage float for temperature

* Humidity: float, reference
  values: reference to storage float for humidity

* tempUnit: tempUnit, default = TempUnit_Celsius

* presUnit: uint8_t, default = PresUnit_hPa
```

ChipModel chipModel()

```
* return: [ChipModel](#chipmodel-enum) enum
```

Environment Calculations

float Altitude(float pressure, AltitudeUnit = AltitudeUnit_Meters, float seaLevelPressure = 1013.25, outsideTemp = 15.0, TempUnit = TempUnit_Celsius)

Calculate the altitude based on the pressure with the specified units.

```
Return: float altitude

* Pressure: float
  values: unit independent

* AltitudeUnit: default = AltitudeUnit_Meters
  values:  AltitudeUnit_Meters, AltitudeUnit_Feet

* Sea Level Pressure: float, default = 1013.25
  values:  unit independent

* outsideTemp: float, default = 15.0
  values:  any float related to TempUnit

* TempUnit: default = TempUnit_Celsius
  values: TempUnit_Celsius, TempUnit_Fahrenheit

  Note: The formula evaluates the height difference based on difference of pressure.
 - May be used to evaluate altitude over MSL. (default set)
           (default referencePressure, default outsideTemp parameters ~ ISA standard)
           The altitude is derived from QNH, used in aviation.
 - May be used to evaluate height over MSL
           (referencePressure should be equal to QFF read in meteorologic synoptic maps,
           outsideTemp should be equal to local temperature)
 - May be used to evaluate the height difference between two points
           (referencePressure should be set to the pressure on the lower point.)
```

float EquivalentSeaLevelPressure(float altitude, float temp, float pres, AltitudeUnit altUnit, TempUnit tempUnit)

Convert current pressure to equivalent sea-level pressure.

```
Return: float equivalent pressure at sea level.

* altitude: float
```

```
      values: meters

    * temp: float
      values: celsius

    * pres: float
      values: unit independent

    * AltitudeUnit: default = AltitudeUnit_Meters
      values:  AltitudeUnit_Meters, AltitudeUnit_Feet

    * TempUnit: default = TempUnit_Celsius
      values: TempUnit_Celsius, TempUnit_Fahrenheit

      Note: To get correct EquivalentSeaLevel pressure (QNH or QFF) the altitude value should be independent
      on measured pressure. It is necessary to use fixed altitude point e.g. the altitude of barometer
      read in map.
```

float DewPoint(float temp, float hum, TempUnit = TempUnit_Celsius)

Calculate the dew point based on the temperature and humidity with the specified units.

```
    return: float dew point

    * Temperature: float
      values: any float related to TempUnit

    * Humidity: float, unit = % relative humidity
      values: any float

    * TempUnit: TempUnit, default = TempUnit_Celsius
      values: TempUnit_Celsius = return degrees Celsius, TempUnit_Fahrenheit = return degrees Fahrenheit
```

float AbsoluteHumidity(float temperature, float humidity, TempUnit = TempUnit_Celsius)

Calculate the absolute humidity based on the temperature and humidity with the specified units.

```
    return: float absolute humidity

    * Temperature: float
      values: any float related to TempUnit

    * Humidity: float, unit = % relative humidity
      values: any float

    * TempUnit: TempUnit, default = TempUnit_Celsius
      values: TempUnit_Celsius = return degrees Celsius, TempUnit_Fahrenheit = return degrees Fahrenheit
```

float HeatIndex(float temperature, float humidity, TempUnit tempUnit = TempUnit_Celsius)

Calculate the heat index based on the temperature and humidity with the specified units. The U.S. NWS algorithm is used.

```
    return: float heat index in TempUnit

    * Temperature: float
      values: any float related to TempUnit

    * Humidity: float, unit = % relative humidity
      values: any float

    * TempUnit: TempUnit, default = TempUnit_Celsius
      values: TempUnit_Celsius = return degrees Celsius, TempUnit_Fahrenheit = return degrees Fahrenheit
```

Contributing

1. Fork the project.
2. Create your feature branch: `git checkout -b my-new-feature`
3. Commit your changes: `git commit -am 'Add some feature'`

4. Push to the branch: `git push origin my-new-feature`
5. Submit a pull request.

History

- Jan 1, 2016 - Version 1.0.0 released
- Sep 19, 2016 - Version 2.0.0 released (Restructure for I2C and SPI)
- Nov 21, 2016 - Version 2.0.1 released (Set mode support)
- Dec 19, 2016 - Version 2.1.0 released (Support for SPI)
- Dec 21, 2016 - Version 2.1.1 released (Bugs)
- Feb 17, 2017 - Version 2.1.2 released (Docs)
- Sept 9, 2017 - Version 2.1.3 released (Formatting, reorg)
- Sept 13, 2017 - Version 2.1.4 released (Examples update, bug fixes)
- Oct 7, 2017 - Version 2.2.0 released (Enums, begin restructure)
- Oct 10, 2017 - Version 2.2.1 released (Bug fixes)
- Nov 21, 2017 - Version 2.3.0 released (Examples updates, env calc fixes, bugs)

Credits

Written by Tyler Glenn, 2016.

Special thanks to Mike Glenn for editing and reviewing the code.

License

GNU GPL, see License.txt