Oximeter & Heart Rate Module

BMH08002-4
Arduino Library V1.0.2 Description

Revision: V1.10 Date: October 19, 2023

www.bestmodulescorp.com

Contents

Introduction

The BMH08002-4 is an oximeter and heart rate module from Best Modules, which uses the UART communication method. This document provides the description of the BMH08002-4 Arduino Lib functions and how to install the Arduino Lib. The example uses the BMH83M002 module to demonstrate the function of obtaining blood oxygen, heart rate and perfusion index.

Applicable types:

Type	Description
BMH08002-4	Oximeter and Heart Rate Module
BMH83M002	Integrated BMH08002-4 Module

Arduino Lib Functions

Arduino Lib Name: BMH08002-4		Lib Version: V1.0.2
Constructors & Initialisation		
1	BMH08002_4(uint16_t enPin, HardwareSerial *theSerial=&Serial)	
	Description	Constructor, uses hardware UART
	Parameter	enPin: Enable pin *theSerial: Serial parameter
	Return Value	—
	Note	—
2	BMH08002_4(uint16_t enPin, uint16_t rxPin, uint16_t txPin)	
	Description	Constructor, uses software UART
	Parameter	enPin: Enable pin rxPin: UART data receive line txPin: UART data transmit line
	Return Value	—
	Note	—
3	void begin()	
	Description	Module initialisation
	Parameter	—
	Return Value	void
	Note	—
Performance Functions		
4	uint8_t beginMeasure()	
	Description	Start the measurement
	Parameter	—
	Return Value	Execution result: 0: Succeeded 1: Failed
	Note	Using the oximeter module is divided into 3 steps: Step 1: set the operating mode using setModeConfig; Step 2: start measuring using this function; Step 3: use the isInfoAvailable+readInfoPackage function or requestInfoPackage function to read the measuring result according to the operating mode.

5	uint8_t endMeasure()	
	Description	Complete the measurement
	Parameter	—
	Return Value	Execution result: 0: Succeeded 1: Failed
	Note	—
6	uint8_t sleep()	
	Description	Enter the sleep mode
	Parameter	—
	Return Value	Execution result: 0: Succeeded 1: Failed
	Note	Wake up using beginMeasure()
7	void powerDown()	
	Description	Enter powerDown mode
	Parameter	—
	Return Value	void
	Note	Wake up using beginMeasure()
8	uint8_t requestInfoPackage(uint8_t buff[])	
	Description	In the Inquire-respond mode, read the measured data
	Parameter	buff[]: Store the read measured data buff[0]: blood oxygen, ranging from 35~99% buff[1]: heart rate, ranging from 30 to 250, unit: BPM buff[2]: perfusion index, ranging from 0~200, 1 represents 0.1% buff[3]: heart rate variability, ranging from 0~255, unit: ms buff[4]: pulse, ranging from 0~16 buff[5]: systolic time interval, ranging from 0~255, unit: ms buff[6]~buff[7]: PPG waveform 1 buff[8]~buff[9]: PPG waveform 2
	Return Value	State of the measured data 0: No data was measured 1: Data is being generated 2: Data has been generated and can be read
	Note	—
9	bool isInfoAvailable()	
	Description	In the Periodic Sending mode and Continuous Uploading mode, query whether there is a receipt of information automatically output by the module
	Parameter	—
	Return Value	Has received information automatically output by the module true: Has received false: Not received
	Note	—
10	void readInfoPackage(uint8_t buff[])	
	Description	In the Periodic Sending mode and Continuous Uploading mode, read the measured data
	Parameter	buff[]: used to store module information (15 bytes)
	Return Value	void
	Note	Used after “isInfoAvailable()==true”

11	uint8_t updateWave()	
	Description	Update waveform control parameters
	Parameter	—
	Return Value	Execution result: 0: Succeeded 1: Failed
	Note	Used in PPG waveform mode, such as waveform display with OLED/TFT screen
12	uint8_t calSensitivity()	
	Description	Calibrate the finger detection sensitivity
	Parameter	—
	Return Value	Execution result: 0: Succeeded 1: Failed
	Note	—
13	uint16_t getTimeInterval()	
	Description	Obtain the time interval for Periodic Sending data
	Parameter	—
	Return Value	Time interval for Periodic Sending data
	Note	1 represents 4ms, if the obtained data is 5, the time interval for Periodic Sending data is 5×4=20ms
14	uint8_t getModeConfig();	
	Description	Obtain the operating mode
	Parameter	—
	Return Value	Operating mode: 0x00: Periodic Sending mode, the red LED is off when detecting a finger 0x01: Periodic Sending mode, the red LED is blinking when detecting a finger 0x02: Inquire-respond mode, the red LED is off when detecting a finger 0x03: Inquire-respond mode, the red LED is blinking when detecting a finger 0x04: Continuous Uploading (PPG waveform) mode, the red LED is off when detecting a finger 0x05: Continuous Uploading (PPG waveform) mode, the red LED is blinking when detecting a finger
	Note	—
Parameter Configuration		
15	uint8_t setTimeInterval(uint16_t Time)	
	Description	Set the time interval for Periodic Sending data
	Parameter	Time: time interval for Periodic Sending data, 1 represents 4ms, if enter 5, the time interval is 5×4=20ms
	Return Value	Execution result: 0: Succeeded 1: Failed
	Note	Used for Periodic Sending mode

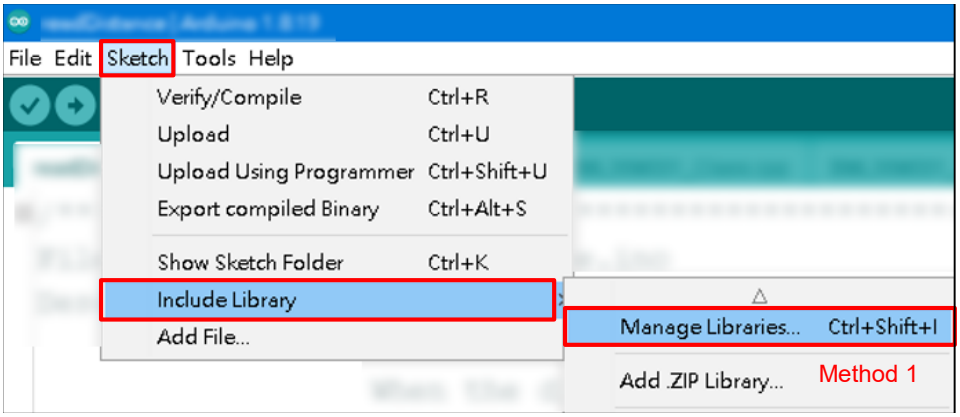
16	uint8_t setModeConfig(uint8_t mode_code)	
	Description	Configure the operating mode
	Parameter	mode_code: Operating mode 0x00: Periodic Sending mode, the red LED is off when detecting a finger 0x01: Periodic Sending mode, the red LED is blinking when detecting a finger 0x02: Inquire-respond mode, the red LED is off when detecting a finger 0x03: Inquire-respond mode, the red LED is blinking when detecting a finger 0x04: Continuous Uploading (PPG waveform) mode, the red LED is off when detecting a finger 0x05: Continuous Uploading (PPG waveform) mode, the red LED is blinking when detecting a finger
	Return Value	Execution result: 0: Succeeded 1: Failed
	Note	—

Arduino Lib Download and Installation

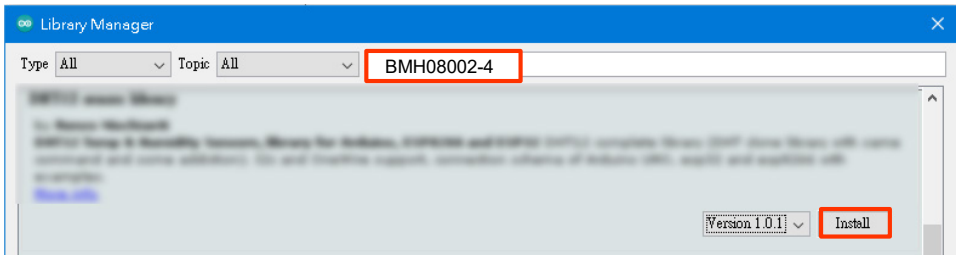
BMH08002-4 Library: Refer to the following two methods to install the BMH08002-4 Arduino Library

Method 1: Search for installation

Search for installation: Arduino IDE→Sketch→Include Library→Manage Libraries...→Search BMH08002-4→Install



Search for Installation Step 1

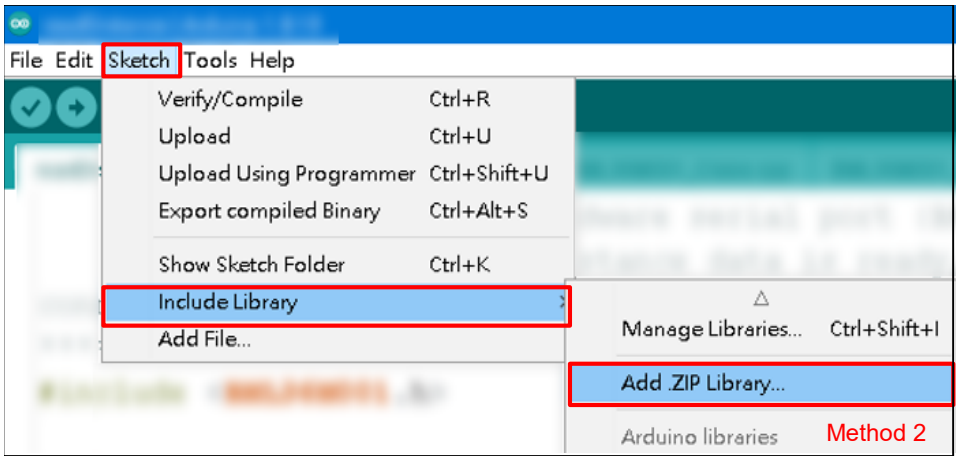


Search for Installation Step 2

Method 2: Download before adding a ZIP library

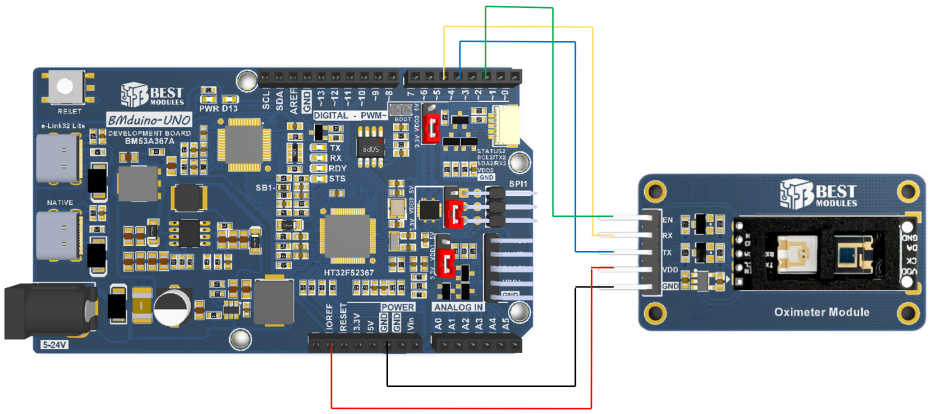
Download method: Open the Best Modules official website (https://www.bestmodulescorp.com/bmh08002-4.html) and download the BMH08002-4 Library from "Arduino example program" under the "DOCUMENTS" menu.

Add .ZIP library: Arduino IDE→Sketch→Include Library→Add .ZIP Library....



Arduino Example

Example: readDataInfo



Physical Connection Diagram

Example function: Obtain the module measured data and display it on the serial monitor

1. Open the example: File→Examples→Select Lib (BMH08002-4)→Select example (readDataInfo)

2. Example description:

a. Create object & initialise object

```
#include <BMH08002-4.h>
BMH08002_4 mySpo2(2,5,4);    // Software UART, EN=D2, RX=D5, TX=D4
uint8_t Mode=0;              // Set the operating mode
uint8_t rBuf[15]={0};        // Store the measured data
uint8_t Status=0;            // Measuring status
uint8_t flag=0;// Whether there is a finger close to the measured place
void setup()
{
  Serial.begin(9600);// Initialise the serial monitor with a baud rate
                     // of 9600
  mySpo2.begin();    // Initialise the module
  mySpo2.setModeConfig(0x02);// Configure the operating mode: Inquire -
                             // respond mode, the red LED is blinking
                             // when detecting a finger
  Serial.println("Please place your finger");  // Prompt to place your
                                               // finger
  delay(2000);                 // Wait for the finger to be placed
  mySpo2.beginMeasure();       // Enter the start measurement state
  Mode = mySpo2.getModeConfig();// Query the operating mode
  if(Mode == 0x02 || Mode == 0x03)
  {
    Mode = 1;                  // Inquire - respond mode
  }
  else Mode = 0;// Periodic Sending mode and Continuous Uploading mode
}
```

b. Obtain the measured data, print the SpO2, PI and heart rate data and display it in the serial monitor

```
void loop()
{
  switch(Mode)
  {
    case 1:
    Mode_ask();              // Inquire - respond mode
    break;
    default:
    Mode_continuous_timing();// Periodic Sending mode and Continuous
                             // Uploading mode
  }
}
void Mode_ask()
{
  Status= mySpo2.requestInfoPackage(rBuf);
  if (Status==0x02) // Determine if there is a 15-byte of data that
                    // has been measured
  {
    Serial.println("Measurement completed,Can remove fingers");
    Serial.print("SpO2:");
    Serial.print(rBuf[0],DEC);// Print the SpO2,heart rate and PI data
    Serial.println("%");
    Serial.print("Heart rate:");
    Serial.print(rBuf[1],DEC);
    Serial.println("BMP");
    Serial.print("PI:");
```

```
    Serial.print((float)rBuf[2] / 10);
    Serial.println("%");
    mySpo2.endMeasure();     // Stop measuring
    mySpo2.sleep();          // Enter the sleep mode
  }
  if (Status==0x01&&flag!=1) // Detected that a finger has been close to
                             // the measured place and the data is
                             // being generated
  {
    Serial.println("Don't move your fingers");
    flag=1;
  }
  if (Status==0x00&&flag!=0)// No fingers have been placed. Prompt again
  {
    Serial.println("Please reposition your fingers");
    flag=0;
  }
}
void Mode_continuous_timing()
{
  if (mySpo2.isInfoAvailable() == true) //
  {
    mySpo2.readInfoPackage(rBuf);
    for(uint8_t i=0;i<15;i++)
    {
      Serial.print(i);
      Serial.print(":");
      Serial.println(rBuf[i]);
    }
  }
}
```

3. Open the serial monitor and select the baud rate to be 9600. The serial monitor will display as follows.