Laser Dust Digital Sensor

BM25S3221-1
Arduino Library V1.0.1 Description

Revision: V1.00 Date: September 15, 2023

www.bestmodulescorp.com

Contents

Introduction

The Best Modules BM25S3221-1 is a laser dust digital sensor, which uses the UART communication method. This document provides the description of the BM25S3221-1 Arduino Lib functions and how to install the Arduino Lib. The example demonstrates the function of configuring the module parameters and reading the dust concentrations.

Applicable types:

Part No.	Description
BM25S3221-1	Laser dust digital sensor
BME25K322	On-board BM25S3221-1 sensor

Arduino Lib Functions

Arduino Lib Name: BM25S3221-1		Lib Version: V1.0.1
Constructor & Initialisation		
1	BM25S3221_1(uint8_t statusPin, HardwareSerial *theSerial=&Serial)	
	Description	Constructor, uses the hardware serial
	Parameter	statusPin: PWM input pin, connected to the BM25S3221-1 PWM pin or the BME25K322 STA pin *theSerial: Select hardware serial port (default serial port)
	Return Value	—
	Note	—
2	BM25S3221_1(uint8_t statusPin, uint8_t rxPin, uint8_t txPin)	
	Description	Constructor, uses the software serial
	Parameter	statusPin: PWM input pin, connected to the BM25S3221-1 PWM pin or the BME25K322 STA pin rxPin: RX pin, connected to the BM25S3221-1 or the BME25K322 TX pin txPin: TX pin, connected to the BM25S3221-1 or the BME25K322 RX pin
	Return Value	—
	Note	—
3	void begin()	
	Description	Module initialisation
	Parameter	—
	Return Value	void
	Note	Communication baud rate: 9600bps
4	void preheatCountdown()	
	Description	Wait for the module to finish warm-up
	Parameter	—
	Return Value	void
	Note	The warm-up time is about 30 seconds
Performance Functions		
5	uint16_t readPM25Value()	
	Description	Obtain the PM2.5 concentration values through the PWM
	Parameter	—
	Return Value	PM2.5 concentration, unit: μg/m^3
	Note	—

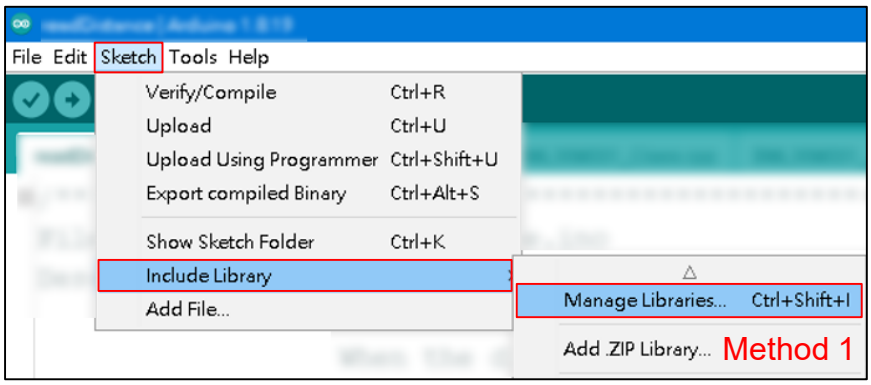
6	uint8_t readDustValue(uint16_t array[])	
	Description	Obtain the dust concentration value
	Parameter	array[]: Store the dust concentration value, array[0]/array[1]/array[2]:PM1.0/PM2.5/PM10, unit: μg/m^3
	Return Value	Execution result: 0: Read succeeded 1: Module reply verification failed 2: Module reply timeout
	Note	—
7	bool isInfoAvailable()	
	Description	Query whether received the automatic output information by the module
	Parameter	—
	Return Value	Reception result: true: Received false: Not received
	Note	Used in active upload mode
8	void readInfoPackage(uint8_t array[])	
	Description	Read the module active output information
	Parameter	array[]: Used to store the module information (18 bytes)
	Return Value	void
	Note	This function needs to be used after the “if (isInfoAvailable()==true)”. Refer to the datasheet for the each byte meaning
9	void setUploadMode(uint8_t modeCode)	
	Description	Set the module data upload mode
	Parameter	modeCode: Mode code 0x40 (AUTO): Active upload mode 0x41 (CMD): Command query mode
	Return Value	void
	Note	—
10	uint8_t sleep()	
	Description	Enter the sleep mode
	Parameter	—
	Return Value	Execution result: 0: Setup succeeded 1: Module reply verification failed 2: Module reply timeout 3: Setup failed
	Note	—
11	uint8_t wakeUp()	
	Description	Exit the sleep mode
	Parameter	—
	Return Value	Execution result: 0: Setup succeeded 1: Module reply verification failed 2: Module reply timeout 3: Setup failed
	Note	—

Arduino Lib Download and Installation

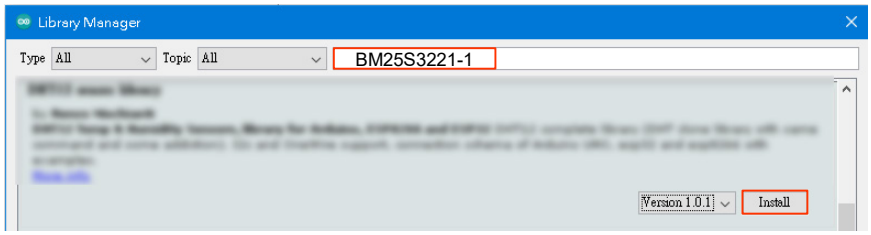
BM25S3221-1 Library: Refer to the following two methods to install the BM25S3221-1 Arduino Library.

Method 1: Search for installation

Arduino IDE → Sketch → Include Library → Manage Libraries... → Search BM25S3221-1 → Install



Search for Installation Step 1

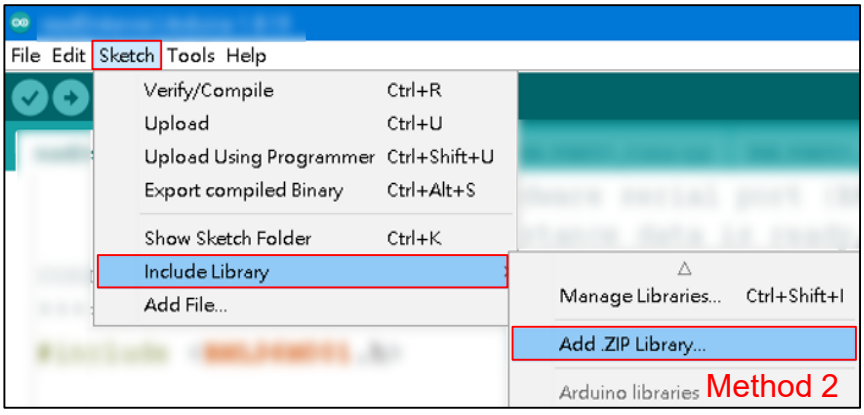


Search for Installation Step 2

Method 2: Download the .ZIP library before adding it

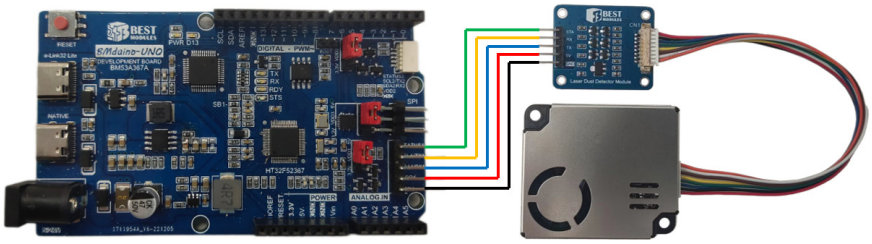
Download the Arduino example (BM25S3221-1 Library) under the DOCUMENTS menu from the Best Modules website (https://www.bestmodulescorp.com/bm25s3221-1.html).

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



Arduino Example

Example: readDustConcentration



Physical Connection Diagram

Connect VDD1 and 5V through a jump cap.

Function: Obtain the PM1.0, PM2.5 and PM10 concentration.

1. Open the example: Arduino IDE → File → Examples → Select Lib (BM25S3221-1) → Select example (readDustConcentration)
2. Example Description:

a. Create object & initialise object

```
#include <BM25S3321-1.h>
#define STA_PIN 22                    // Input pin
#define PM1_0 dustValue[0]
#define PM2_5 dustValue[1]
#define PM10 dustValue[2]
uint8_t dataBuf[32] = {0}, command = 0;
uint16_t dustValue[3] = {0};
/* Bmduino-UNO */
BM25S3221_1 dust(STA_PIN, &Serial1); // Hardware serial: Serial1
void setup()
{
  dust.begin(); // Initialise module, set the baud rate to be 9600bps
  Serial.begin(9600);       // Initialise serial interface,
                            // set the baud rate to be 9600bps
  Serial.println("Module preheating...(about 30 second)");
  dust. preheatCountdown(); // Wait for the module to finish warm-up
  Serial.println("End of module preheating.");
  Serial.println();
  Serial.println("Perform initial setup.");
  displayMenu();            // Displays the mode selection menu
}
```

b. Print mode selection menu

```
void displayMenu()
{
  Serial.println("==== Enter the serial number to run the
                      corresponding command ====");
  Serial.println("1. Setup the module to command query mode.");
  Serial.println("2. Setup the module to active upload mode.");
```

```
  Serial.println("=========================================
                  ==================");
  Serial.println();
}
```

c. Send "1", "2", "3", "4" through the serial monitor (set the monitor end with "Enter"), and select the corresponding working mode

```
void selectMode()
{
  uint8_t tmp[2] = {0};
  while (Serial.available() > 0)
  {
    tmp[0] = Serial.read();
    tmp[1] = Serial.read();
    if (tmp[1] == 13) // The ASCII code for "Enter" is 13
    {
      command = tmp[0] - 48;
    }
    else
    {
      command = 0;
    }
    switch (command)
    {
      case 0:
        Serial.println("Please enter the correct serial number and
                        end with carriage return.");
        break;
      case 1:
        Serial.println("1. Setup the module to command query mode.");
        dust.setUploadMode(CMD);
        break;
      case 2:
        Serial.println("2. Setup the module to active upload mode.");
        dust.setUploadMode(AUTO);
        delay(1200);
        break;
    }
    Serial.println();
  }
}
```

d. According to the selected upload mode, get the dust concentration value and print it to the serial monitor.

```
void loop()
{
  selectMode(); // Default select command query mode
  delay(30);
  if (command == 1)
  {
    if (dust.readDustValue(dustValue) != 0)
    {
      Serial.println("read failed!");
    }
    delay(1000);
  }
```

```
  if (command == 2)
  {
    if (dust.isInfoAvailable() == true)
    {
      dust.readInfoPacket(dataBuf);
      PM1_0 = ((uint16_t)dataBuf[10] << 8) + dataBuf[11];
      PM2_5 = ((uint16_t)dataBuf[12] << 8) + dataBuf[13];
      PM10 = ((uint16_t)dataBuf[14] << 8) + dataBuf[15];
    }
    else
    {
      Serial.println("read failed!");
    }
  }
  if (command > 0)
  {
    Serial.print("PM1.0: ");
    Serial.print(PM1_0);
    Serial.println("μg/m³");
    Serial.print("PM2.5: ");
    Serial.print(PM2_5);
    Serial.println("μg/m³");
    Serial.print("PM10: ");
    Serial.print(PM10);
    Serial.println("μg/m³");
    Serial.println();
    delay(1000);
  }
}
```

3. Open the serial monitor and set the baud rate to be 9600, send "1" through the UART after the module is preheated. The serial monitor will display as follows.