USB to I²C
SYSINNO Sensor Carrier Board

EZU2-I301

User Manual

Rev 1.0

Revision History

Revision	Date	Description
1.0	2023/07/20	Initial Release

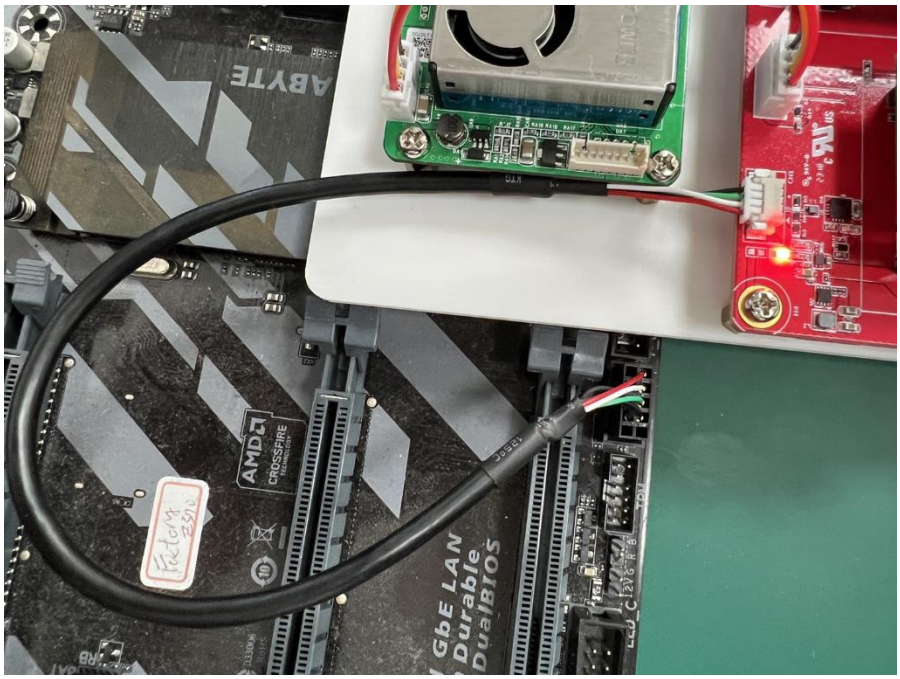
Table of Contents

Revision History ii
Table of Contents iii
1. Hardware Installation 1
1.1. Carrier Board 1
1.2. PM2.5 1
1.3. Other Sensor 2
2. Windows OS 3
2.1. Driver Installation 3
2.2. Sample Code 4
3. Linux OS 6
3.1. Driver Installation 6
3.2. Sample Code 6
4. Software API 8
4.1. UISMShowVer 8
4.2. UISMDebugLevel 8
4.3. UISMOpenDevice 9
4.4. UISMCloseDevice 9
4.5. UISMSensorInfo 9
4.6. UISMInit 10
4.7. UISMQuery 11
4.8. Error Codes 11
Contact us 13

1. Hardware Installation

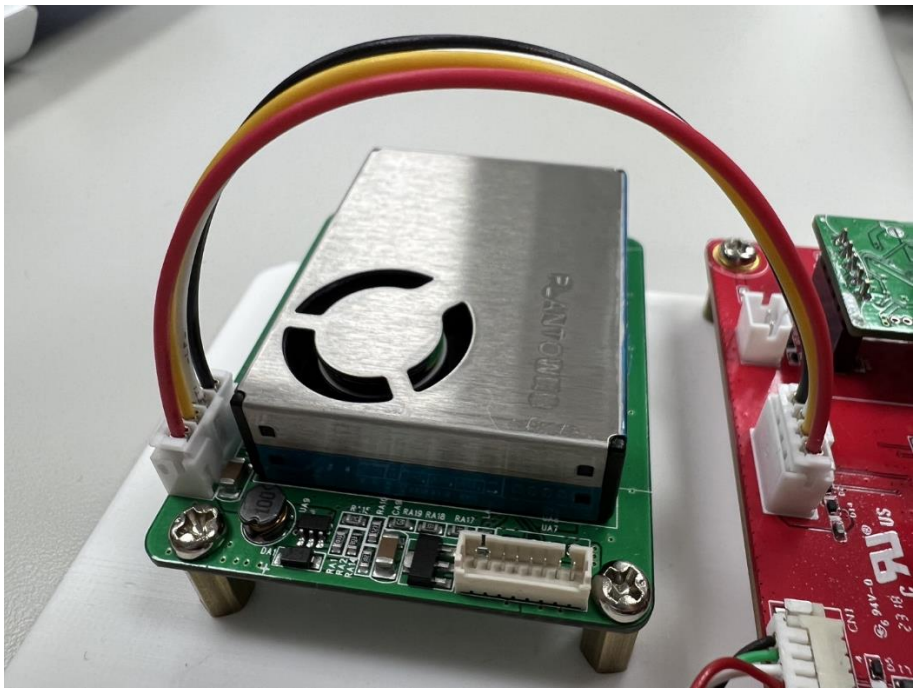
1.1. Carrier Board

The board comes with a USB cable (9pin to 4pin) can connect USB 9pin connector on motherboard to CN1.



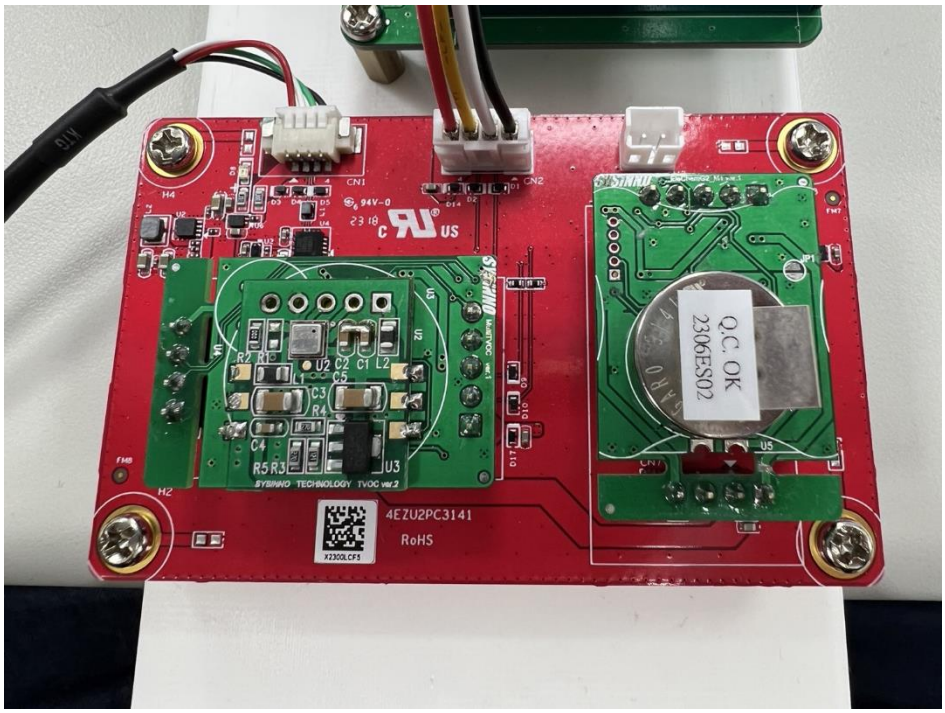
1.2. PM2.5

PM 2.5 sensor comes with a cable can connector to CN2 of sensor carrier board.



1.3. Other Sensor

Other sensors, which have two rows pins (4pin + 5pin) can install on the sensor carrier, board directly.

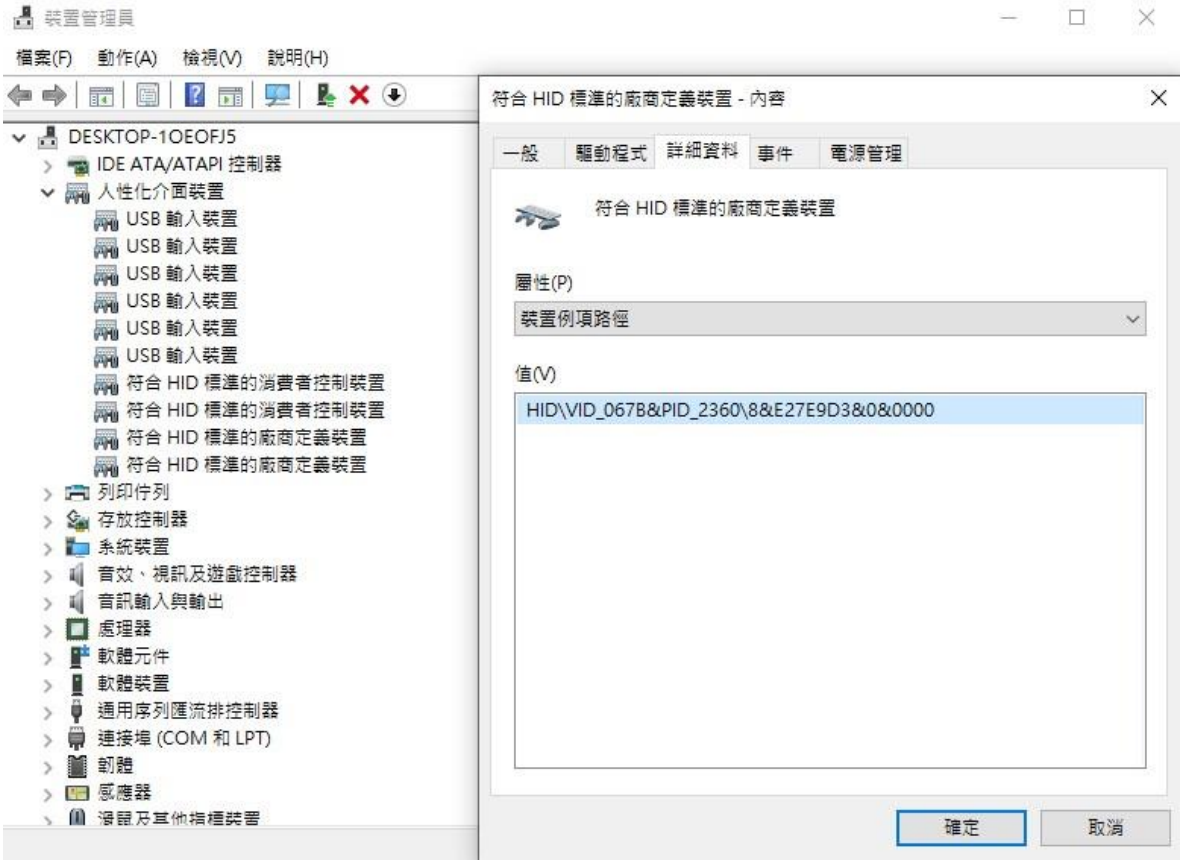


2. Windows OS

2.1. Driver Installation

Sensor carrier board uses OS native HID driver.

The device vendor ID “0x067B” can be found in “Device Manager”.



2.2. Sample Code

Adjust ***"Setup.ini"*** according to the needs of the test. The details of individual parameters are as follows.

debug_level:

Adjust different levels of debug level according to requirements.
(0=DBG_LVL_NONE, 1=DBG_LVL_FUNC, 2=DBG_LVL_HID)

Interval:

Set the data update interval of the selected monitoring sensor.
(unit: millisecond)

auto_scan:

Whether to use the Query function in the SDK to automatically detect the existing sensor module on the device.
When the "auto_scan" is enabled, the "sensor" section will be ignored.
(0=disable, 1=enable)

check_status:

Whether to provide the status information of the sensor module at the same time when returning data.
(0=disable, 1=enable)

"sensor option" section:

When the 'auto_scan' item is disabled, use this section to specify the sensor module to be selected.

The detailed and latest information can be found in the description of *sample\setup.ini*.

setup.ini

```
;
; some setting for testing
;
[setting]
debug_level=0           ; 0=DBG_LVL_NONE, 1=DBG_LVL_FUNC, 2=DBG_LVL_HID

;
; monitor infomation
;
[monitor]
interval=1000           ; unit: [millisecond]. Set the data update interval of the selected monitoring sensor.
auto_scan=1             ; 0=disable, 1=enable. When the "auto_scan" is enabled, the "sensor" section will be ignored.
check_status=1          ; 0=disable, 1=enable.

;
; sensor option
;
[sensor]
HCHO=1                  ; 0=disable, 1=enable.
PM=1                    ; 0=disable, 1=enable.
CO2=1                   ; 0=disable, 1=enable.
CO=1                    ; 0=disable, 1=enable.
O3=1                    ; 0=disable, 1=enable.
TVOC=1                  ; 0=disable, 1=enable.
TH=1                    ; 0=disable, 1=enable.
```

The execution diagram of the sample code.

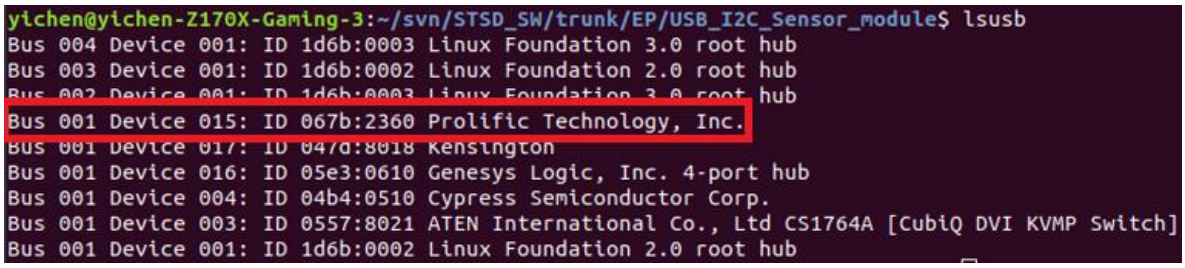
```
C:\USB_I2C_Sensor_module\sample
λ .\uism_sample.exe

Config
---------------------------------
[setting]
debug_level : 0

[monitor]
interval     : 1000
auto_scan    : True
check_status : True

[sensor]
HCHO         : True
PM           : True
CO2          : True
CO           : True
O3           : True
TVOC         : True
TH           : True

---------------------------------

UISMDebugLevel() successfully !
================================================
UISMOpenDevice() successfully !
================================================
UISMShowVer() successfully !
LIB ver: V1.0.0
================================================
UISMQuery() successfully !
HCHO : Not Found
PM   : Not Found
CO2  : Found
CO   : Not Found
O3   : Not Found
TVOC : Not Found
TH   : Found
================================================

[CO2 sensor information]
UISMSensorInfo() successfully !
PID        : IAGCO2
FW version : 0038
SN         : SENCO22304ES01
Status     : 0
================================================

[TH sensor information]
UISMSensorInfo() successfully !
SN         : 0F8FB045
================================================
UISMInit() successfully !
================================================
CO2   [ unit: 1       ppm ] : 1804  (status: 0)
TEMP  [ unit: 0.01      C ] : 2509
RH    [ unit: 0.01      % ] : 6289
```

3. Linux OS

3.1. Driver Installation

Sensor carrier board uses OS native HID driver. Supports Linux kernel 2.6.38 and above.

Use “lsusb” command to make sure the device is recognized by OS.



3.2. Sample Code

Adjust ***“Setup.ini”*** according to the needs of the test. The details of individual parameters are as follows.

debug_level:

Adjust different levels of debug level according to requirements.

(0=DBG_LVL_NONE, 1=DBG_LVL_FUNC, 2=DBG_LVL_HID)

Interval:

Set the data update interval of the selected monitoring sensor.

(unit: millisecond)

auto_scan:

Whether to use the Query function in the SDK to automatically detect the existing sensor module on the device.

When the "auto_scan" is enabled, the "sensor" section will be ignored.

(0=disable, 1=enable)

check_status:

Whether to provide the status information of the sensor module at the same time when returning data.

(0=disable, 1=enable)

“sensor option” section:

When the ‘auto_scan’ item is disabled, use this section to specify the sensor module to be selected.

The detailed and latest information can be found in the description of *sample\setup.ini*.

setup.ini

sample > setup.ini

```
;
; some setting for testing
;
[setting]
debug_level=0           ; 0=DBG_LVL_NONE, 1=DBG_LVL_FUNC, 2=DBG_LVL_HID

;
; monitor infomation
;
[monitor]
interval=1000           ; unit: [millisecond]. Set the data update interval of the selected monitoring sensor.
auto_scan=1             ; 0=disable, 1=enable. When the "auto_scan" is enabled, the "sensor" section will be ignored.
check_status=1          ; 0=disable, 1=enable.

;
; sensor option
;
[sensor]
HCHO=1                  ; 0=disable, 1=enable.
PM=1                    ; 0=disable, 1=enable.
CO2=1                   ; 0=disable, 1=enable.
CO=1                    ; 0=disable, 1=enable.
O3=1                    ; 0=disable, 1=enable.
TVOC=1                  ; 0=disable, 1=enable.
TH=1                    ; 0=disable, 1=enable.
```

The execution diagram of the sample code.

```
yichen@yichen-Z170X-Gaming-3:~/USB_I2C_Sensor_module/sample$ sudo ./UISM

Config
--------------------------------
[setting]
debug_level : 0

[monitor]
interval     : 1000
auto_scan    : True
check_status : True

[sensor]
HCHO         : True
PM           : True
CO2          : True
CO           : True
O3           : True
TVOC         : True
TH           : True

--------------------------------

UISMDebugLevel() successfully !
============================================
UISMOpenDevice() successfully !
============================================
UISMShowVer() successfully !
LIB ver: V1.0.0
============================================
UISMQuery() successfully !
HCHO : Not Found
PM   : Not Found
CO2  : Found
CO   : Not Found
O3   : Not Found
TVOC : Not Found
TH   : Found
============================================

[CO2 sensor information]
UISMSensorInfo() successfully !
PID        : IAGCO2
FW version : 0038
SN         : SENCO22304ES01
Status     : 0
============================================

[TH sensor information]
UISMSensorInfo() successfully !
SN         : 0F8FB045
============================================
UISMInit() successfully !
============================================
CO2   [ unit: 1       ppm ] : 1840  (status: 0)
TEMP  [ unit: 0.01      C ] : 2446
RH    [ unit: 0.01      % ] : 6583
```

4. Software API

UISM API is based on a dynamic library (DLL) in Windows/Linux to control USB to I^2C SYSINNO Sensor Carrier Board.

Please refer to the following table for UISM APIs:

ID	Function Name
4.1	UISMShowVer
4.2	UISMDebugLevel
4.3	UISMOpenDevice
4.4	UISMCloseDevice
4.5	UISMSensorInfo
4.6	UISMInit
4.7	UISMQuery

4.1. UISMShowVer

int UISMShowVer (VER_INFO *ver_info)

Function: get version information.

Parameters:

ver_info: version information.

```
typedef struct
{
    char api[VER_LEN];
}
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

4.2. UISMDebugLevel

int UISMDebugLevel (int level)

Function: Set Debug level of SDK.

Parameters:

level: Debug level Number

```
enum
{
    DBG_LVL_NONE = 0,
    DBG_LVL_FUNC,
    DBG_LVL_HID,
```

```
};
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

4.3. UISMOpenDevice

int UISMOpenDevice (void)

Function: Open HID device.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

4.4. UISMCloseDevice

int UISMCloseDevice (void)

Function: Close HID device.

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

4.5. UISMSensorInfo

int UISMSensorInfo (SENSOR_INFO *sensor_info)

Function: Get the specified sensor module information.

Parameters:

sensor_info: The pointer of the sensor module information structure.

```
typedef struct
{
    /*----- input variable from user to API    -----*/
    int                 name;

    /*----- output variable from API to user -----*/
    unsigned char PID[PID_LEN + 1];
    unsigned char FW_ver[FW_LEN + 1];
    unsigned char SN[SN_LEN + 1];

    unsigned char status;
} SENSOR_INFO;
```

```
/* Sensor name */
#define SENSOR_NAME_HCHO    0x00000001
#define SENSOR_NAME_PM      0x00000002
#define SENSOR_NAME_CO2     0x00000004
#define SENSOR_NAME_CO      0x00000008
#define SENSOR_NAME_O3      0x00000010
#define SENSOR_NAME_TVOC    0x00000020
#define SENSOR_NAME_TH      0x00000040
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

4.6. UISMInit

int UISMInit (bool status, INIT_INFO *init_info)

Function: The data return of starting or stopping the SDK and setting the initial parameters.

Parameters:

status: Set the data return of starting or stopping the SDK.

```
/* INIT */
enum
{
    INIT_STS_STOP = 0,
    INIT_STS_START,
};
```

Init_info: The pointer of the initial information structure.

```
typedef struct
{
    unsigned int interval;  /* unit: ms */
    int          monitor_opt;
    bool         check_status;
    DATA_CB      data_cb;
} INIT_INFO;
```

monitor_opt: The value after performing the OR operator on the sensor module names to be returned (for example, if you want to return CO2 and O3, that is SENSOR_NAME_CO2 | SENSOR_NAME_O3 = 0x00000014), or you can use the UISMQuery() API to obtain the results of all current sensor modules for use.

```
typedef void (*DATA_CB) (int error_code, int name, DATA_INFO *info);
typedef struct
```

```
{
    unsigned char *data1;
    int           data1_len;

    unsigned char *data2;
    int           data2_len;

    unsigned char  status;
} DATA_INFO;
```

Return Value:

If the function succeeds, the return value is zero. If the function fails, the return value is nonzero and the error code generated by the API. Refer to section **Error Codes** to find more information.

4.7. UISMQuery

int UISMQuery (int *result)

Function: Obtain the results of all current sensor modules.

Parameters:

result: The pointer used to store the query result.

Return Value:

If the function succeeds, the return value is filter total count - 1 (0-14, MAX: 15). If the function fails, the return value is error code generated by the API. Refer to section **Error Codes** to find more information.

4.8. Error Codes

The below table lists errors that functions API returns in response to calls. The detailed and latest information can be found in the description of library\header\liblism.h.

Error Code	Description
0	Success
1	Found HID device error
2	HID device can't found (Device count is 0)
3	Open HID device error
4	Close HID device error
5	Initial HID device error
6	Exit HID device error
11	Set I^2C frequency error
12	Set I^2C address error

13	I^2C get information error
21	This function is only supported when the device is opened
22	This function is only supported when the device is closed
23	This function is only supported when the device is started
24	This function is only supported when the device is stopped
25	Input invalid variable to API
31	Invalid sensor name

Contact us

Headquarters (Taiwan)

5F., No. 237, Sec. 1, Datong Rd., Xizhi Dist., New Taipei City 221, Taiwan

Tel: +886-2-77033000

Email: sales@innodisk.com

Branch Offices:

USA

usasales@innodisk.com

+1-510-770-9421

Europe

eusales@innodisk.com

+31-40-3045-400

Japan

jpsales@innodisk.com

+81-3-6667-0161

China

sales_cn@innodisk.com

+86-755-21673689

www.innodisk.com

July 20, 2023