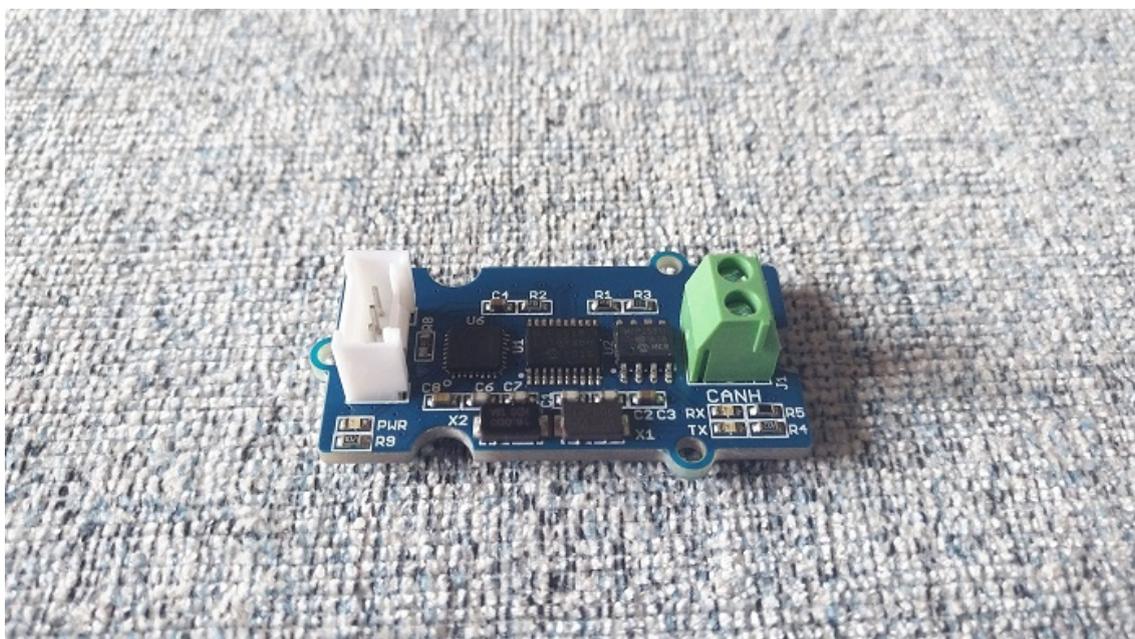


# I2C CAN BUS MODULE

## *Introduction*



We launched the [Serial CAN Bus Module](#) years ago, which is very practical and easy to use. Now we have slightly modified the Serial CAN Bus Module, wrote a special firmware, and launched this I2C CAN Bus Module.

The I2C CAN Bus Module uses I2C for communication. The board is small and flexible, which makes it can be quickly used in any system with an I2C interface.

The I2C CAN Bus Module is based on the high-performance MCP2515 CAN Bus controller and MCP2551 CAN Bus transceiver, provides a CAN Bus communication rate of up to 1Mb/s.

In addition, there is an Atmega168PA microcontroller on the board, you can also program it through a USB to Serial board, modify the firmware or write your application directly.

 **Note**

The default masks and filters is for standard can frame, if you need to receive a ext can frame, please set the mask and filt first.

## CAN BUS PRODUCTS LIST OF LONGAN-LABS



We have made a lot of can bus products, you can get more information through the following list, so as to choose a product suitable for you.

PRODUCT NAME	LINK	PRICE	MCU	CHIP
Serial CAN Bus Module	<a href="#">LINK</a>	\$19.9	ATMEGA168PA	MCP2515
I2C CAN Bus Module	<a href="#">LINK</a>	\$19.9	ATMEGA168PA	MCP2515
OBD-II Serial CAN Bus Dev Kit	<a href="#">LINK</a>	\$20.9	ATMEGA168PA	MCP2515

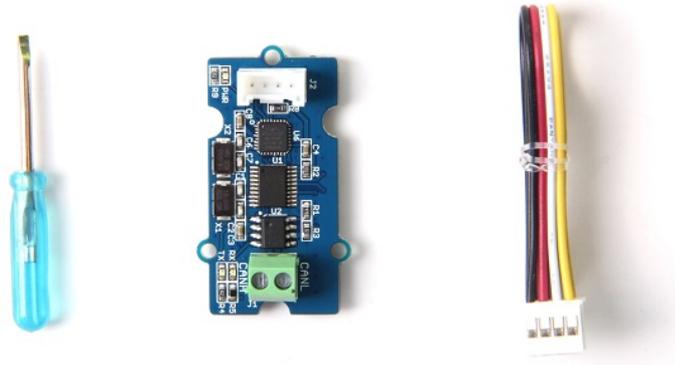
PRODUCT NAME	LINK	PRICE	MCU	CHIP
OBD-II CAN Bus GPS Dev Kit	<a href="#">LINK</a>	\$29.9	ATMEGA32U4	MCP2515
OBD-II CAN Bus Basic Dev Kit	<a href="#">LINK</a>	\$24.9	ATMEGA32U4	MCP2515
CAN-FD Shield	<a href="#">LINK</a>	\$19.9	NO MCU	MCP2517F
CAN Bus Shield	<a href="#">LINK</a>	\$9.9	NO MCU	MCP2515
CANBed	<a href="#">LINK</a>	\$24.9	ATMEGA32U4	MCP2515
CANBed-FD	<a href="#">LINK</a>	\$29.9	ATMEGA32U4	MCP2517F
CANBed M4	<a href="#">LINK</a>	\$49.9	ATSAME51	-
OBD-II RF Dev Kit	<a href="#">LINK</a>	\$19.9	ATmega168PA	MCP2515

 **Note**

The above price may not be the latest price, please refer to the price on the product page.

## Partlist

---



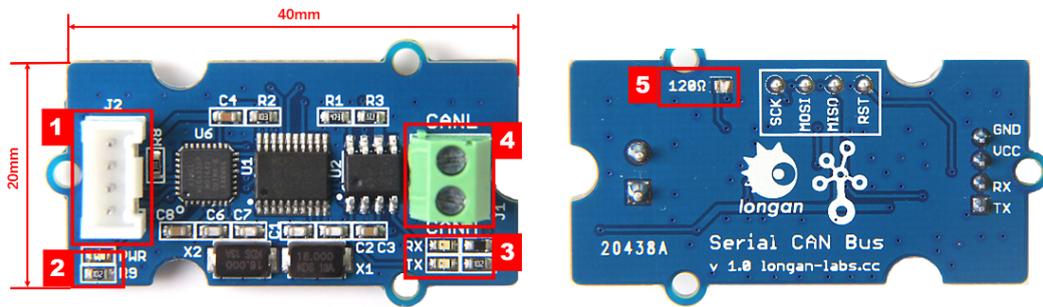
## Features

---

- I2C to CAN Bus Conversion
- Default I2C Address 0x25
- Max I2C Speed: 400k
- Work with Arduino/BeagleBone board/Pi or any MCU that integrated with I2C.
- Up to 1Mb/s CAN Bus baud rate (default 500k)
- TX and RX led indicator for CAN Bus
- 4pin HY connector
- 3.3 / 5V working voltage
- Easy-to-use Arduino library
- Small size: 20x40 mm

## Hardware Overview

---



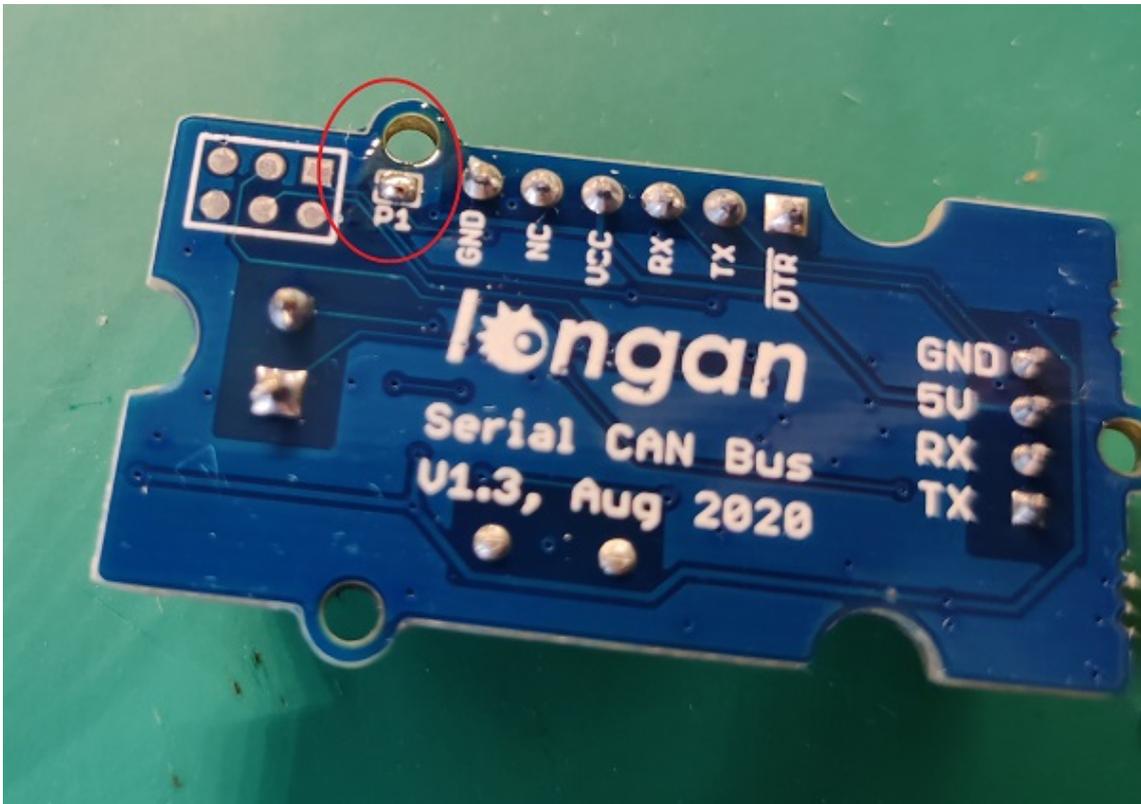
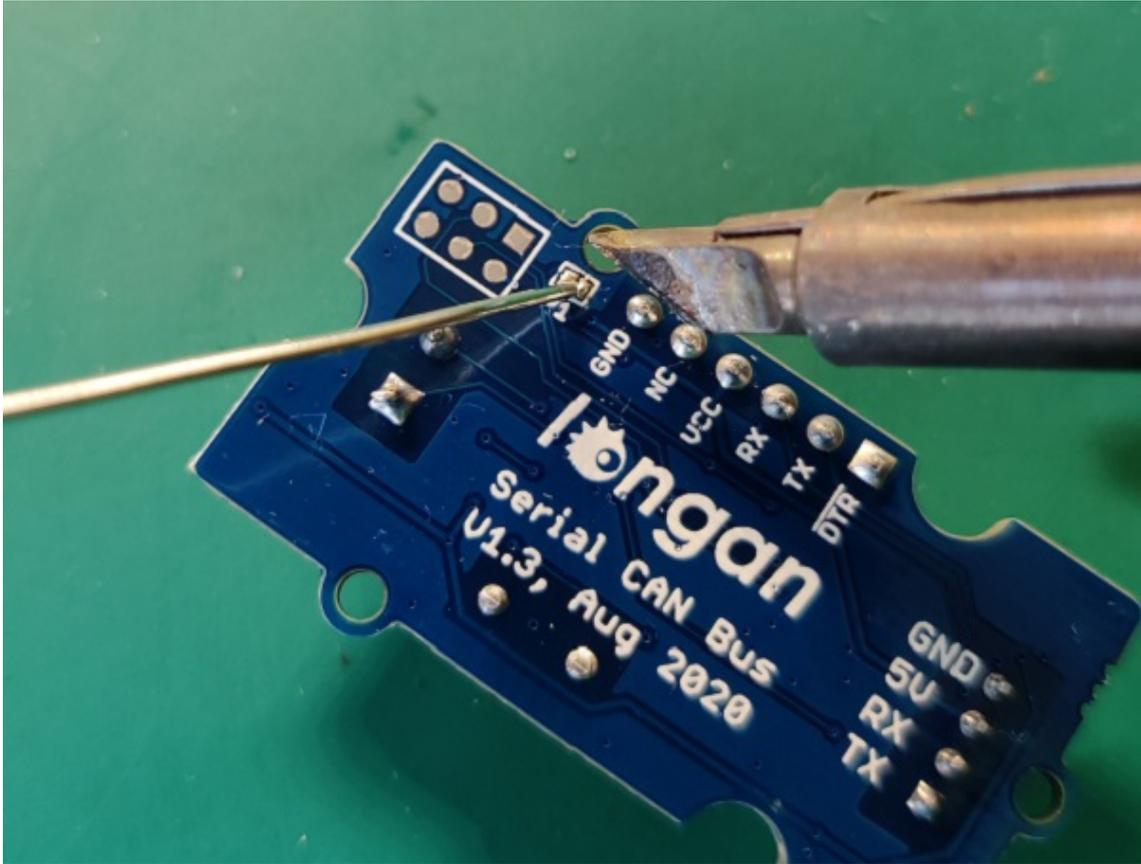
1. 4 pin 2.0mm Connector (Same as Grove connector from Seedstudio)
2. Power and status led indicator
  - Blink all the time: CAN Bus init fail, Maybe the board was damaged
  - When receive data from I2C, blink one time
3. Send and Recv led indicator
4. 3.5mm terminal to connect to CAN Bus (CAN\_H & CAN\_L)
5. 120Ω resistor.

#### Note

The picture shows the Serial CAN Bus Module, but it is also applicable to the I2C CAN Bus Module

## 120 Terminal resistor

If you need the 120Ω terminal resistor, you need a soldering iron, as shown below:



 **Note**

The picture shows the Serial CAN Bus Module, but it is also applicable to the I2C CAN Bus Module

## I2C Register

Address	Name	Length	Type	Default Value
0x01	ADDR	1 Byte	W	0x25
0x02	DNUM	1 Byte	R	0
0x03	BAUD	1 Byte	W/R	16(500kb/s)
0X30	SEND	1 Byte	W	-
0x40	RECV	1 Byte	R	-
0x60	MASK0	5 Byte	W/R	0
0x65	MASK1	5 Byte	W/R	0
0x70	FILT0	5 Byte	W/R	0
0x80	FILT1	5 Byte	W/R	0

Address	Name	Length	Type	Default Value
0x90	FILT2	5 Byte	W/R	0
0xA0	FILT3	5 Byte	W/R	0
0xB0	FILT4	5 Byte	W/R	0
0xC0	FILT5	5 Byte	W/R	0
0X30	SEND	16 Byte	W	-
0x40	RECV	16 Byte	R	-



### ADDR (0x01)

Set the I2C slaver address, default 0x25.

### DNUM (0x02)

The I2C CAN Bus Module can store up to 16 CAN Frames. The number of CAN Frames currently stored in the device can be read this register. If the number of CAN frames stored in the device exceeds 16, the new CAN frame will overwrite the ones that were not read in time.

### BAUD (0x01)

Set the baud rate of CAN Bus, which can be set to the following values.

---

value	01	02	03	04
rate(kb/s)	5	10	20	25

## SEND (0x30), RECV (0x40)

Send/receive a CAN Frame, the length is 16 bytes, defined as follows:

Byte	15	14	13	12
Define	ID3	ID2	ID1	ID0

- ID3~ID0: CAN ID, 11/29 bit
- EXT:
  - 1: Extended CAN Frame
  - 0: Standard CAN Frame
- RTR:
  - 1: Remote CAN Frame
  - 0: Standard CAN Frame
- Dlen: Length of a CAN Frame
- D7~D0: Data of a CAN Frame
- CheckSum: Check sum, get it as below,

```

unsigned char makeChecksum(unsigned char *dta, int len)
{
    unsigned long sum = 0;
    for(int i=0; i<len; i++)sum += dta[i];

    if(sum > 0xff)
    {
        sum = ~sum;
    }
}

```

```

        sum += 1;
    }

    sum = sum & 0xff;
    return sum;
}

```

## MASKn

Set Maskers, the length is 5 bytes, defined as follows:

Byte	4	3	2	1
Define	EXT	D3	D2	D1

- EXT:
  - 1: Extended CAN Frame
  - 0: Standard CAN Frame
- D3~D0: Value for masker, 11/29

## FILTn

Set Filters, the length is 5 bytes, defined as follows:

Byte	4	3	2	1
Define	EXT	D3	D2	D1

- EXT:
  - 1: Extended CAN Frame
  - 0: Standard CAN Frame

- D3~D0: Value for filter, 11/29

## Work with Arduino

---

### 120Ω Terminal Resistor

As there're only 2 serial can bus device, it need the 120Ω resistor, there's a P1 on the back side, please solder P1 to get the 120Ω resistor, as shown below,



#### Note

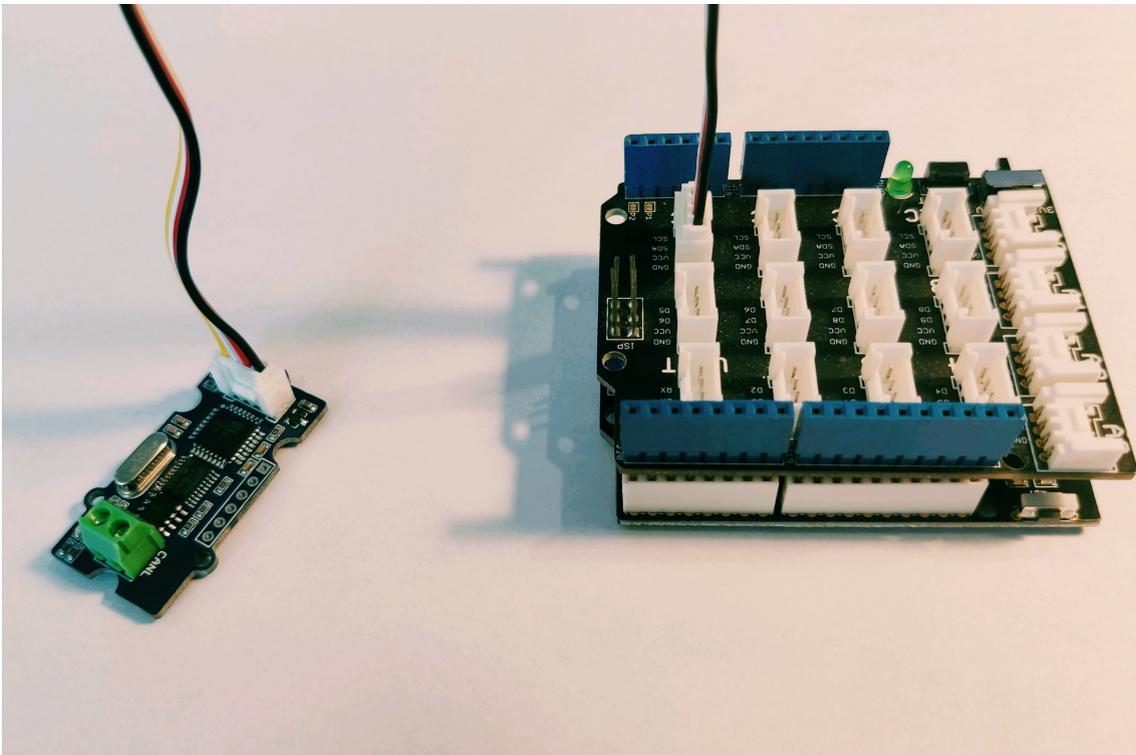
The picture shows the Serial CAN Bus Module, but it is also applicable to the I2C CAN Bus Module

### Hardware Connection

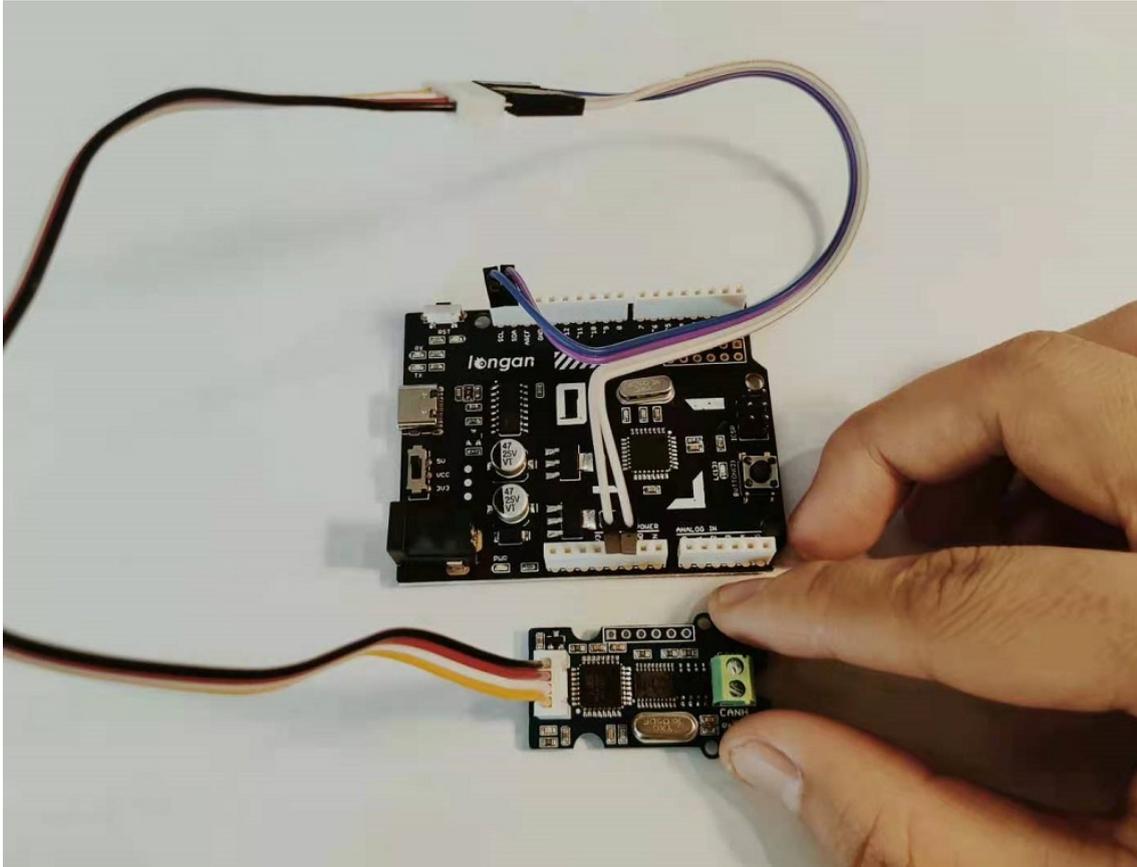
There are 4 pins on the board, they are:

Pin Name	Function	Color
SCL	SCL of I2C	Yellow
SDA	SDA of I2C	White
VCC	3.3 or 5V	Red
GND	Ground	Black

If you got a **Base Shield** from Seedstudio, the connection is easy, like below,



If there's no Base Shield for you, you need some dupont cable, and connect SCL/SDA to the IO of your Arduino, here we connect SCL/SDA to A5/A4:



## Software

We provide an Arduino Library for the board.

Please download it at [Github](#)

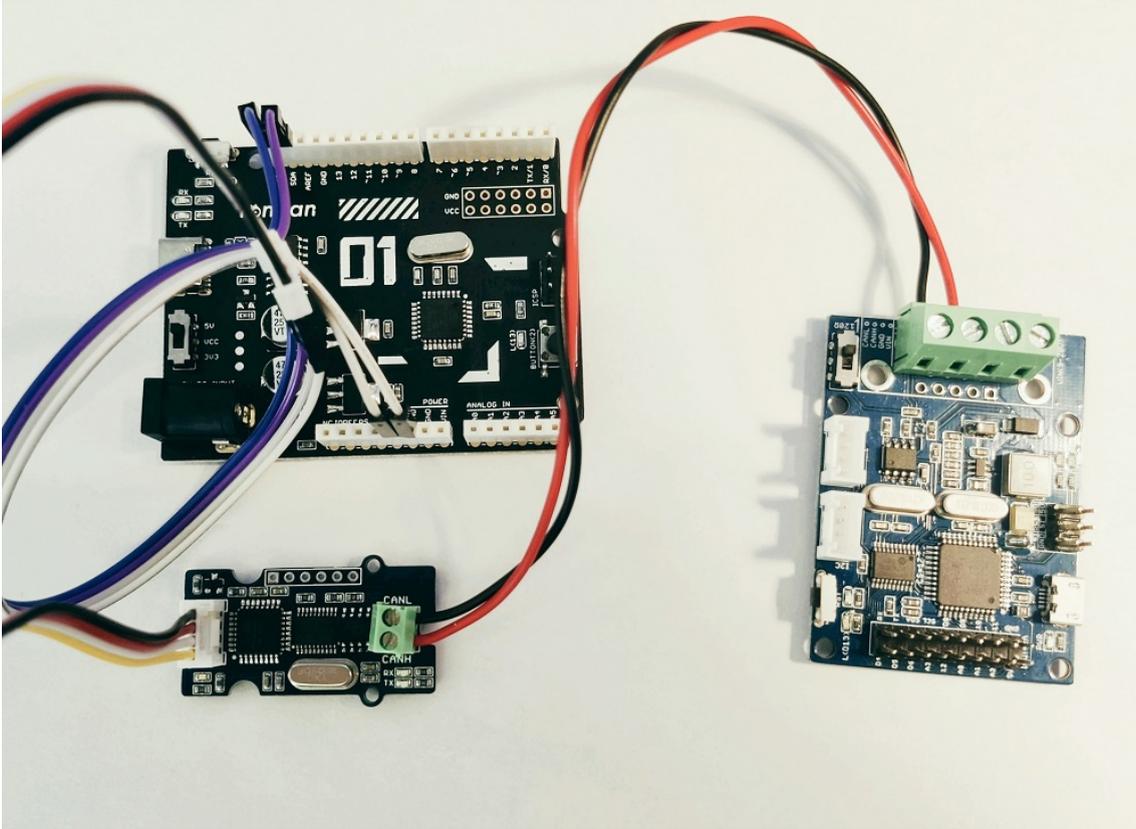
There're many examples for the library, which is consist of,

- ***send*** - How to send a frame to CAN Bus
- ***recv*** - How to recv a frame from CAN Bus
- ***debug*** - debug mode, you can send a cmd to the module
- ***set\_can\_baudrate*** - set can bus baudrate
- ***set\_mask\_filt*** - set mask and filt of the module

## Test the board

Here we make an example to show you how the board work with CAN Bus.

The connection like below, here we use **Longan Core 01** + I2C CAN bus module as a receiver, and a **CANBed V1** as CAN Bus sender.



Upload the **recv example** to Longan Core 01

```

/* receive a frame from can bus
support@longan-labs.cc

CAN Baudrate,

#define CAN_5KBPS           1
#define CAN_10KBPS          2
#define CAN_20KBPS          3
#define CAN_25KBPS          4
#define CAN_31K25BPS        5
#define CAN_33KBPS          6
#define CAN_40KBPS          7
#define CAN_50KBPS          8
#define CAN_80KBPS          9
#define CAN_83K3BPS         10
#define CAN_95KBPS          11
#define CAN_100KBPS         12
#define CAN_125KBPS         13
#define CAN_200KBPS         14

```

```

#define CAN_250KBPS      15
#define CAN_500KBPS     16
#define CAN_666KBPS     17
#define CAN_1000KBPS    18
*/

#include <Wire.h>

#include "Longan_I2C_CAN_Arduino.h"

I2C_CAN CAN(0x25);           // Set I2C
Address

void setup()
{
    Serial.begin(115200);
    //while(!Serial);

    Serial.println("begin to init can");

    while (CAN_OK != CAN.begin(CAN_500KBPS))    // init can bus :
    baudrate = 500k
    {
        Serial.println("CAN BUS FAIL!");
        delay(100);
    }
    Serial.println("CAN BUS OK!");
    pinMode(13, OUTPUT);
}

void loop()
{
    unsigned char len = 0;
    unsigned char buf[8];

    if(CAN_MSGAVAIL == CAN.checkReceive())      // check if
    data coming
    {
        CAN.readMsgBuf(&len, buf);    // read data, len: data
    length, buf: data buf

        if(len)
        {
            unsigned long canId = CAN.getCanId();

            Serial.print("Get ID: ");

```

```
        Serial.println(canId, HEX);

        for(int i = 0; i<len; i++)    // print the data
        {
            Serial.print(buf[i], HEX);
            Serial.print("\t");
        }
        Serial.println();
    }

    delay(10);
    blink();
}

void blink()
{
    static unsigned long timer_s = millis();
    if(millis()-timer_s < 100)return;
    timer_s = millis();
    digitalWrite(13, 1-digitalRead(13));
}
// END FILE
```

Then, upload [send example](#) to CANBed.

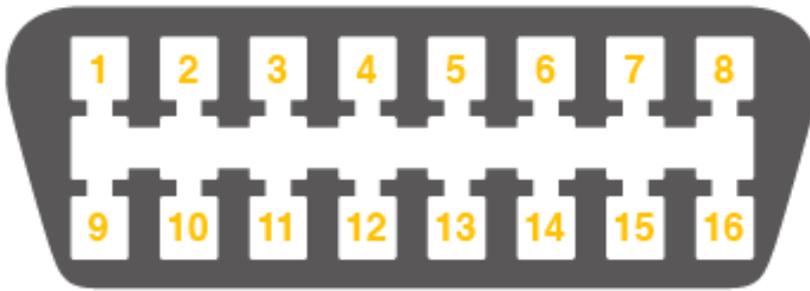
Open Serial monitor of Longan Core 01, you will get the can frame from CAN Bus.

## Get data from a Vehicle

We can use this module to get data from a vehicle, we take the vehicle speed for an example here.

You can use our products to read data from cars. Here we provide a simple example by which you can read the speed and revs from a car. This is the [OBD-based PIDs protocol](#). Regarding the deeper technology of OBD, we can't provide support at present. You may need to have some understanding of the car's protocol. After all, we are more of a hardware supplier.

The interface of OBD is as follows,



PIN	DESCRIPTION	PIN	DESCRIPTION
1	Vendor Option	9	Vendor Option
2	J1850 Bus +	10	j1850 Bus
3	Vendor Option	11	Vendor Option
4	Chassis Ground	12	Chassis Ground
5	Signal Ground	13	Signal Ground
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 Low
8	Vendor Option	16	Battery Power

### OBD-II Connector and Pinout

There're 4 pins we need to connect.

OBD Pin	I2C CAN
6. CAN(J-2234) High	CANH
14. CAN(J-2234) Low	CANL

You can also use our [OBD Connector](#), which is very convenient to connect to OBD.

Upload the following code to the development board, then open the serial monitor, you get the speed from the car now.

```

#include <Wire.h>
#include "Longan_I2C_CAN_Arduino.h"

I2C_CAN CAN(0x25); // Set I2C
Address

#define PID_ENGIN_PRM      0x0C
#define PID_VEHCILE_SPEED 0x0D
#define PID_COOLANT_TEMP  0x05

#define CAN_ID_PID        0x7DF

void set_mask_filt()
{
    // set mask, set both the mask to 0x3ff

    CAN.init_Mask(0, 0, 0x7FC);
    CAN.init_Mask(1, 0, 0x7FC);

    // set filter, we can receive id from 0x04 ~ 0x09

    CAN.init_Filt(0, 0, 0x7E8);
    CAN.init_Filt(1, 0, 0x7E8);

    CAN.init_Filt(2, 0, 0x7E8);
    CAN.init_Filt(3, 0, 0x7E8);
    CAN.init_Filt(4, 0, 0x7E8);
    CAN.init_Filt(5, 0, 0x7E8);
}

void sendPid(unsigned char __pid) {
    unsigned char tmp[8] = {0x02, 0x01, __pid, 0, 0, 0, 0, 0};
    CAN.sendMsgBuf(CAN_ID_PID, 0, 8, tmp);
}

bool getSpeed(int *s)
{
    sendPid(PID_VEHCILE_SPEED);
    unsigned long __timeout = millis();

    while(millis()-__timeout < 1000) // 1s time out
    {
        unsigned char len = 0;
        unsigned char buf[8];
    }
}

```

```

        if (CAN_MSGAVAIL == CAN.checkReceive()) {
// check if get data
            CAN.readMsgBuf(&len, buf);    // read data, len:
data length, buf: data buf

            if(buf[1] == 0x41)
            {
                *s = buf[3];
                return 1;
            }
        }

        return 0;
    }

void setup() {
    Serial.begin(115200);
    while(!Serial);

    while (CAN_OK != CAN.begin(CAN_500KBPS)) {                //
init can bus : baudrate = 500k
        Serial.println("CAN init fail, retry...");
        delay(100);
    }
    Serial.println("CAN init ok!");
    set_mask_filt();
}

void loop() {

    int __speed = 0;
    int ret = getSpeed(&__speed);
    if(ret)
    {
        Serial.print("Vehicle Speed: ");
        Serial.print(__speed);
        Serial.println(" kmh");
    }
    delay(500);
}

// END FILE

```

If you want to get other data by PIDs, you can refer to the above example, and refer to [this link](#)

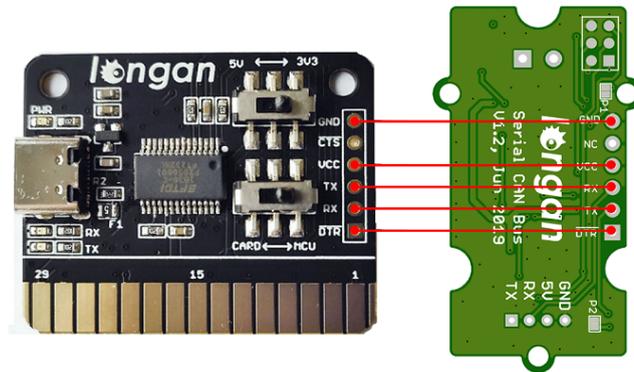
## Working Standalone

---

You can program the on-board MCU (Atmega168PA) with Arduino IDE.

First, you need a USB to Serial board, something like the [Programmer Card](#)

Here we will take Programmer Card as an example,



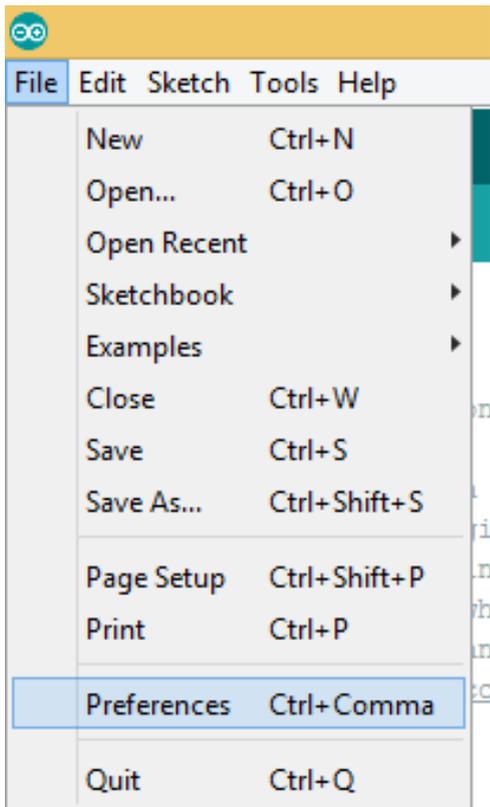
### Note

The picture shows the Serial CAN Bus Module, but it is also applicable to this I2C CAN Bus Module

Then, we need to setup your Arduino IDE.

Start the Arduino IDE.

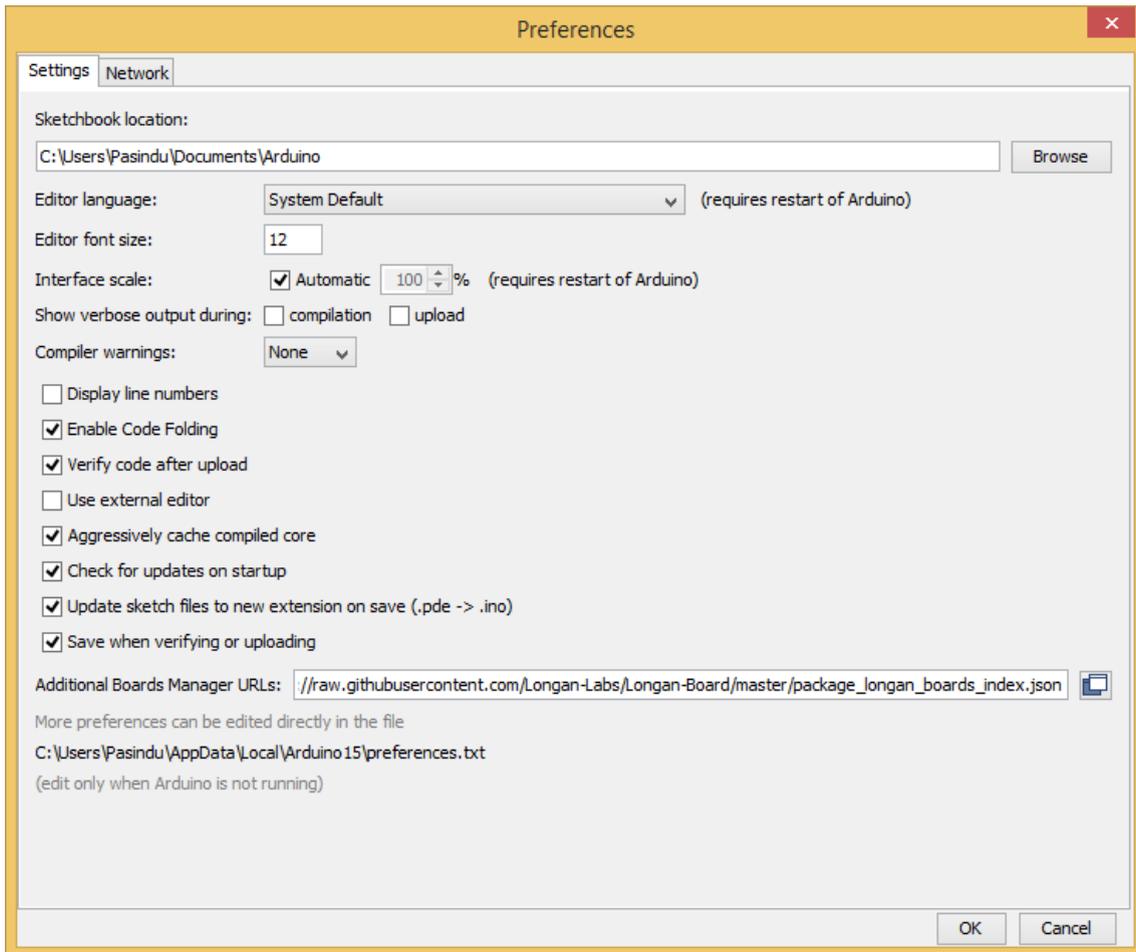
Copy and paste the link below into the **Additional Boards** Manager URLs option in the **Arduino IDE preferences (File > Preferences)**.



[https://raw.githubusercontent.com/Longan-Labs/Longan-Board/master/package\\_longan\\_boards\\_index.json](https://raw.githubusercontent.com/Longan-Labs/Longan-Board/master/package_longan_boards_index.json)

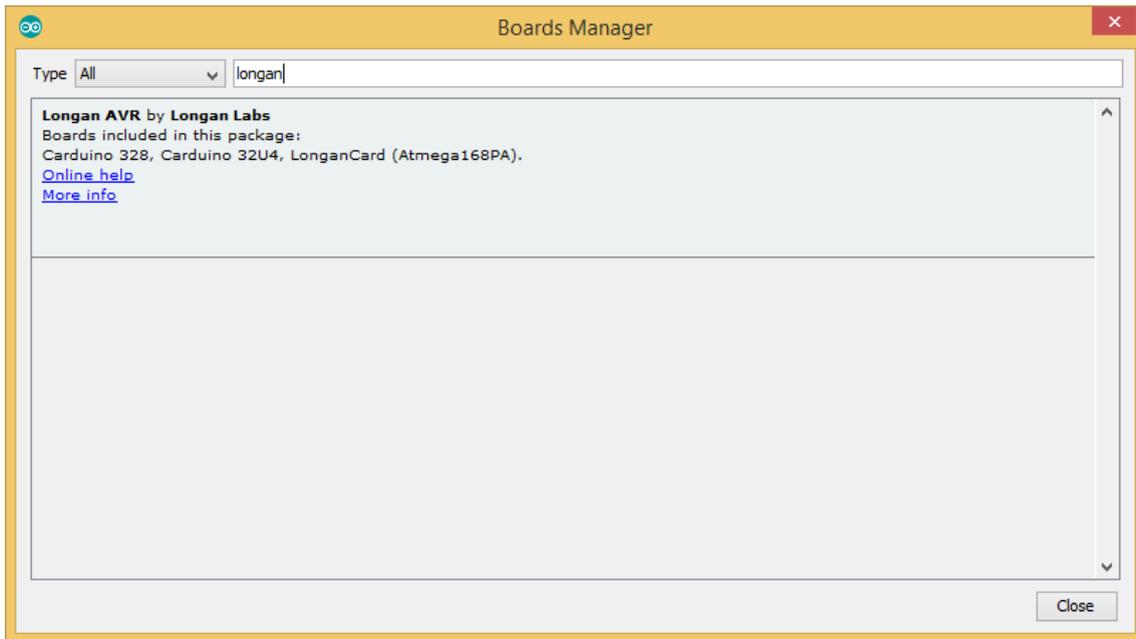
The Longan AVR boards supplied package includes supports for Longan Card, Carduino 328, Carduino 32U4.

Once done, click **OK** button to save the new preference settings.

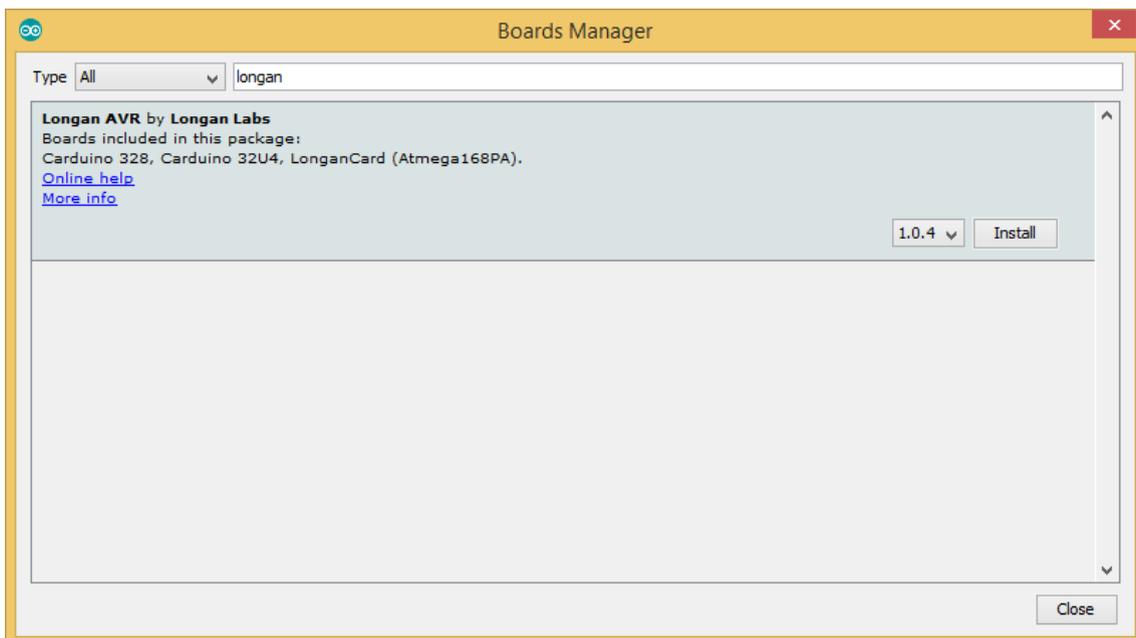


Now open the Boards Manager by navigating to the **Tools -> Board** menu.

Select **All** from the Type drop-down menu. Then type **longan** in the top search bar. While typing, you will see the **Longan AVR package**.



Click on the **Longan AVR by Longan Labs** and then click on **Install** button.



Once installed, close the **Boards Manager** window.

Then you can find **Serial CAN Bus Module** in the board list.

## ***Update the Firmware***

First we need to read the content of the [previous chapter](#).

1. Download the Firmware for [I2C CAN Bus Module](#)
2. You may need this [CAN Bus library](#)
3. Upload the Firmware to I2C CAN Bus module, then it work.

## FAQ

### The RX and TX leds light up instead of blinking

- Please check the baudrate of CAN Bus
- If the serial can bus module is in the end, the 120Ω terminal resistor is needed, please solder P1 on the back side of PCBA

### How to find the tech support

Please contact [support@longan-labs.cc](mailto:support@longan-labs.cc) for technical support. Our technical support engineer will usually reply you within 24 hours on working days. In order to get faster support, we hope that when you send us an email, you need to include at least the following content,

- When and how to buy the product
- Product version information
- Take a high-definition picture of the product you use, including the connection
- Describe in detail the problem you encountered and what kind of help you would like to get

## *Schematic Online Viewer*



## ***Reference***

---

- [Arduino Library](#)
- [Firmware](#)
- [How to get ONE](#)
- [Schematics in Eagle File](#)