Welcome!

Thank you for purchasing our *AZ-Delivery MCP2515 CAN Bus Module*. On the following pages, you will be introduced to how to use and set-up this handy device.

Have fun!

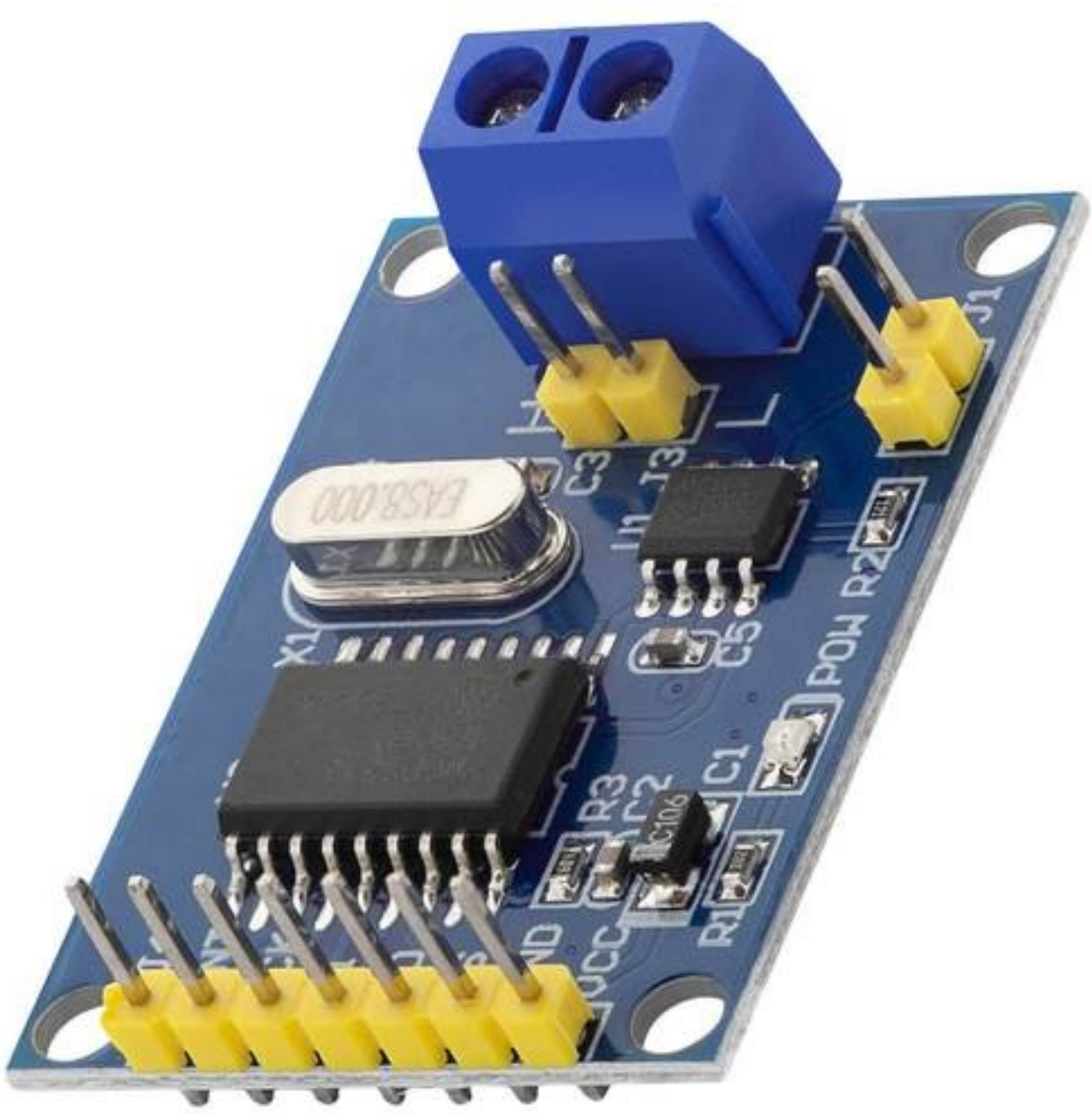


Table of Contents

Introduction

The MCP2515 CAN Bus Controller is a simple module that supports CAN Protocol version 2.0B and can be used for communication up to 1Mbps.

The module is made around MCP2515 chip which is a CAN bus controller and TJA1050 chip which is a CAN protocol tranceiver.

MCP2515 chip is the main controller that internally consists of three main subcomponents such as: CAN Module, Control Logic and SPI Block.

The TJA1050 chip acts as an interface between MCP2515 CAN Controller and the physical CAN Bus. This chip is responsible for taking the data from the controller and relaying it on to the bus.

CAN BUS is a two-wire, half-duplex communication protocol that is widely used in Automotive industry. One of its advantages is that it can be connected to any number of electronic controllers (or microcontrollers) in car through the two-wire bus, reducing the weight of wires that could be gained by using point-to-point communication between electronic controllers.

Specifications

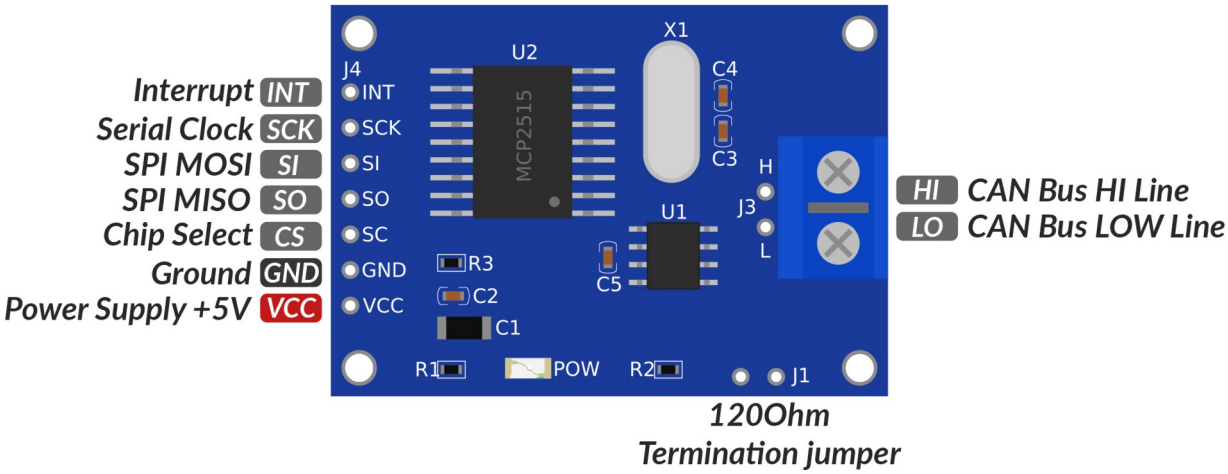
Operating voltage	5V
Operating current	5mA
Stand-by current	1µA
Interface	CAN, SPI
Operating temperature	-40 to 85°C
Dimensions	40x28x10mm (1.5x1.1x0.3in)

To use the MCP2515 CAN Bus module with a Raspberry Pi, Logic Level Converter has to be used because the pins on the Raspberry Pi are not 5V tolerant.

The module can be modded for use with a Raspberry Pi and this requires some soldering skills. Testing and modding will not be covered in this eBook.

The pinout

The MCP2515 CAN Bus Controller module has eleven pins. The pinout is shown on the following image:



The J3 jumper connector shares the same connection as the screw terminal connector on the right side of the image (CANL and CANH). The J1 jumper is connected to CANL and through a 120Ohm resistor to CANH.

How to set-up Arduino IDE

If the Arduino IDE is not installed, follow the *link* and download the installation file for the operating system of choice. The Arduino IDE version used for this ebook is 1.8.13.

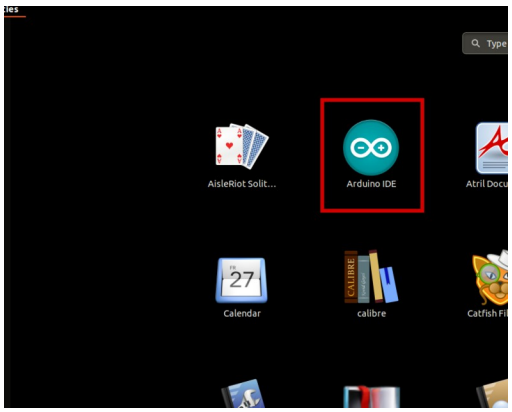
For *Windows* users, double click on the downloaded *.exe* file and follow the instructions in the installation window.

For *Linux* users, download a file with the extension *.tar.xz*, which has to be extracted. When it is extracted, go to the extracted directory and open the terminal in that directory. Two *.sh* scripts have to be executed, the first called *arduino-linux-setup.sh* and the second called *install.sh*.

To run the first script in the terminal, open the terminal in the extracted directory and run the following command:
sh arduino-linux-setup.sh user_name
user_name - is the name of a superuser in the Linux operating system. A password for the superuser has to be entered when the command is started. Wait for a few minutes for the script to complete everything.

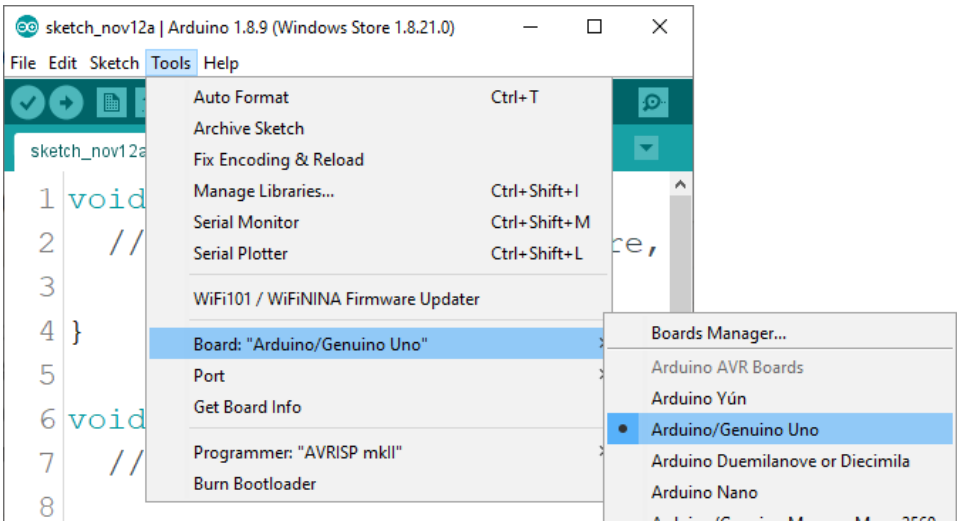
The second script, called *install.sh*, has to be used after the installation of the first script. Run the following command in the terminal (extracted directory): **sh install.sh**

After the installation of these scripts, go to the *All Apps*, where the *Arduino IDE* is installed.



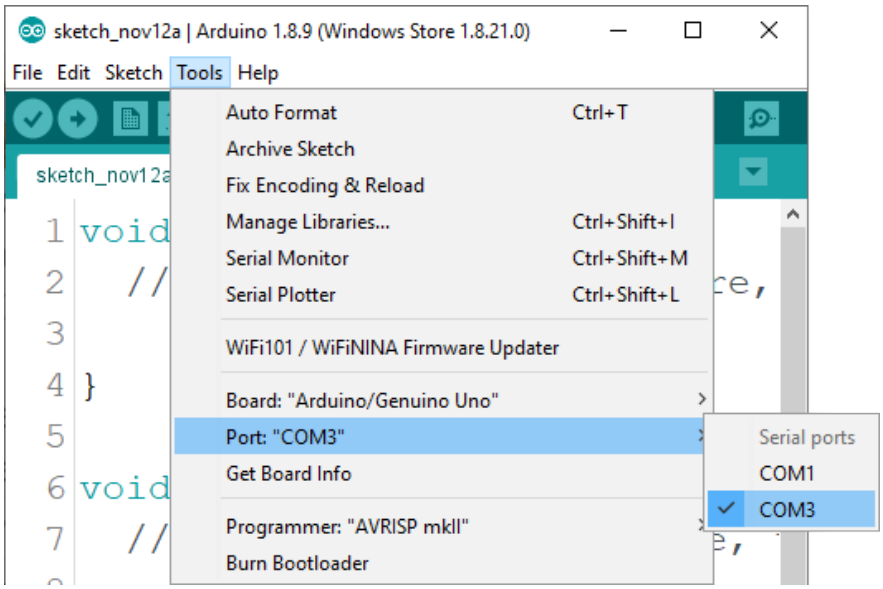
Almost all operating systems come with a text editor preinstalled (for example, *Windows* comes with *Notepad*, *Linux Ubuntu* comes with *Gedit*, *Linux Raspbian* comes with *Leafpad*, etc.). All of these text editors are perfectly fine for the purpose of the eBook.

Next thing is to check if your PC can detect an Atmega328p board. Open freshly installed Arduino IDE, and go to:
Tools > Board > {your board name here}
{your board name here} should be the *Arduino/Genuino Uno*, as it can be seen on the following image:



The port to which the Atmega328p board is connected has to be selected. Go to: *Tools > Port > {port name goes here}*
and when the Atmega328p board is connected to the USB port, the port name can be seen in the drop-down menu on the previous image.

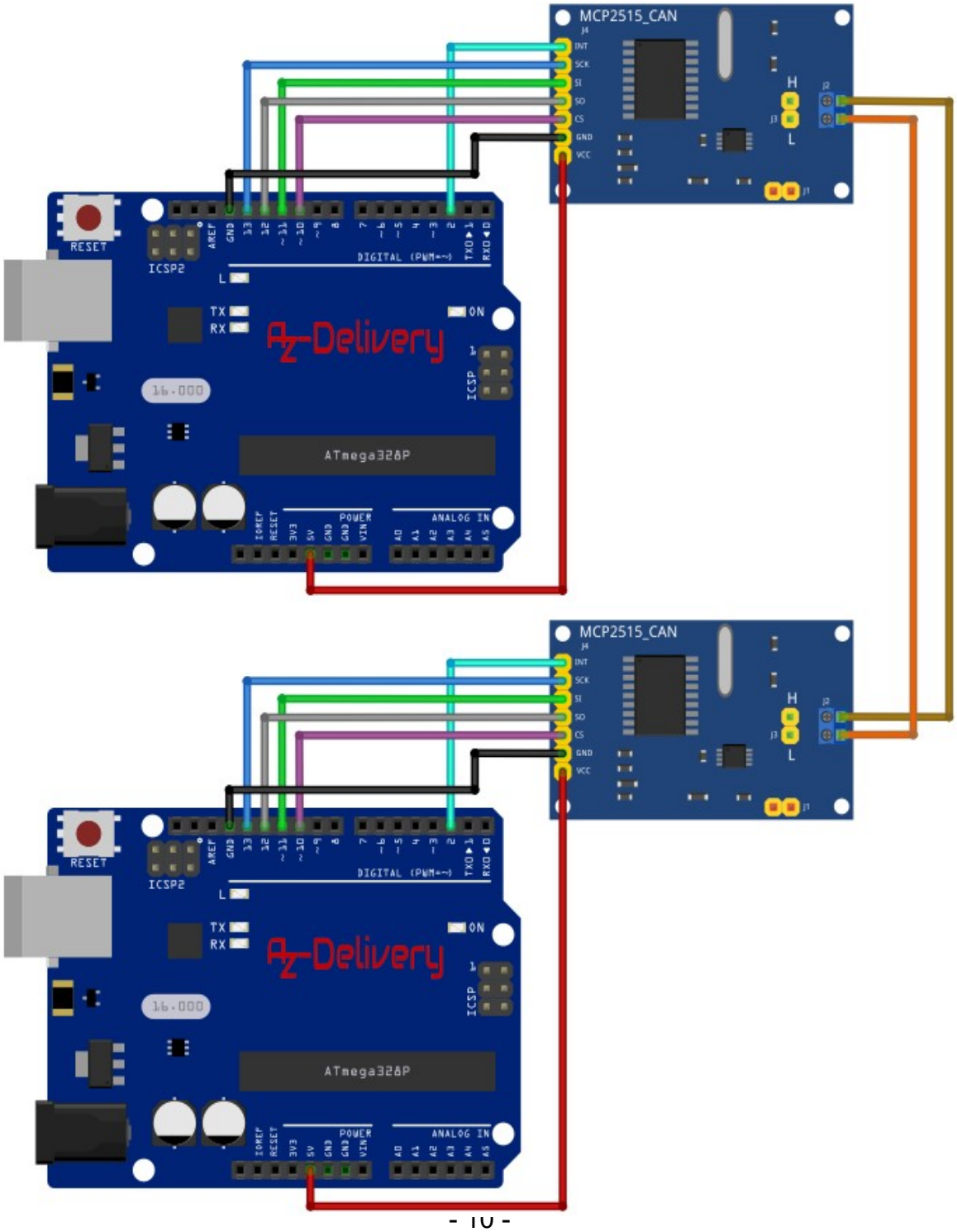
If the Arduino IDE is used on Windows, port names are as follows:



For *Linux* users, for example port name is */dev/ttyUSBx*, where *x* represents integer number between *0* and *9*.

Connecting the module with Atmega328p

Connect the module with the Atmega328p as shown on the following connection diagram:



Module pin	Mc pin	Wire color
INT	D2	Cyan wire
SCK	D13	Blue wire
SI	D11	Green wire
SO	D12	Gray wire
SC	D10	Purple wire
GND	GND	Black wire
VCC	5V	Red wire
H(CAN Bus HI)	To the 2nd module HI	Ochre wire
L(CAN Bus LOW)	To the 2nd module LOW	Orange wire

Library for Arduino IDE

To use the module with Atmega328p it is recommended to download an external library. The most simple library (that is recommend) is called the `CAN` library, which can be downloaded from the following link.

When you download the *.zip* file, open Arduino IDE and go to:
Sketch > Include Library > Add .ZIP Library
and add the downloaded zip file.

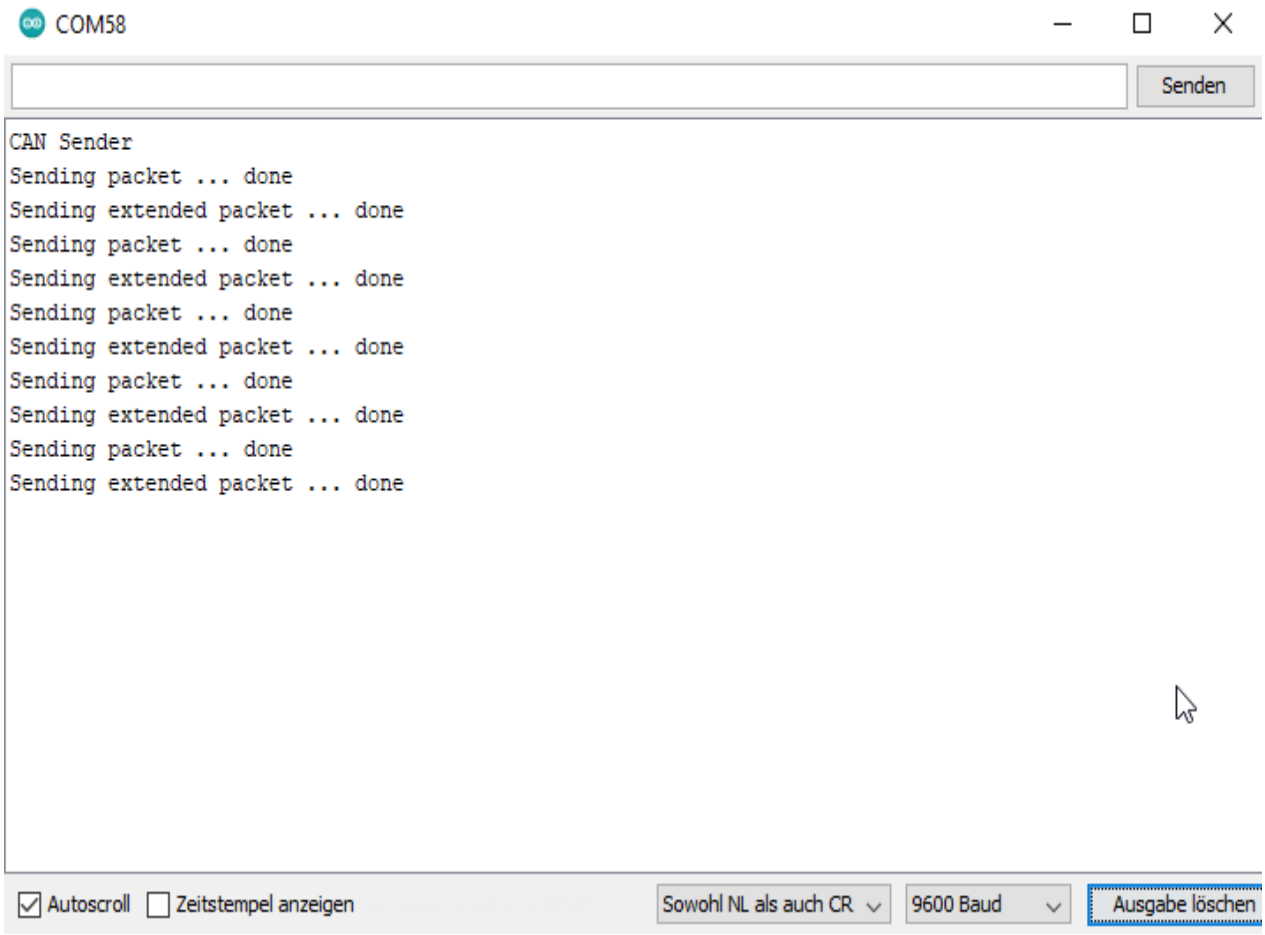
Sketch examples

Transmitter sketch

```
#include <CAN.h>

void setup() {
 Serial.begin(9600);
 while (!Serial);
 Serial.println("CAN Sender");
 // start the CAN bus at 500 kbps
 if (!CAN.begin(500E3)) {
  Serial.println("Starting CAN failed!");
  while (1);
 }
}
void loop() {
 // send packet: id is 11 bits, packet can contain up to 8 bytes of data
 Serial.print("Sending packet ... ");
 CAN.beginPacket(0x12);
 CAN.write('h');
 CAN.write('e');
 CAN.write('l');
 CAN.write('l');
 CAN.write('o');
 CAN.endPacket();
 Serial.println("done");
 delay(1000);
 /* send extended packet: id is 29 bits, packet can contain up to 8 bytes of
data*/
 Serial.print("Sending extended packet ... ");
 CAN.beginExtendedPacket(0xabcdef);
 CAN.write('w');
 CAN.write('o');
 CAN.write('r');
 CAN.write('l');
```

```
 CAN.write('d');
 CAN.endPacket();
 Serial.println("done");
 delay(1000);
}
```

Upload the sketch to the Atmega328p and run the Serial Monitor (*Tools* > *Serial Monitor*). The result should look like as on the following image:

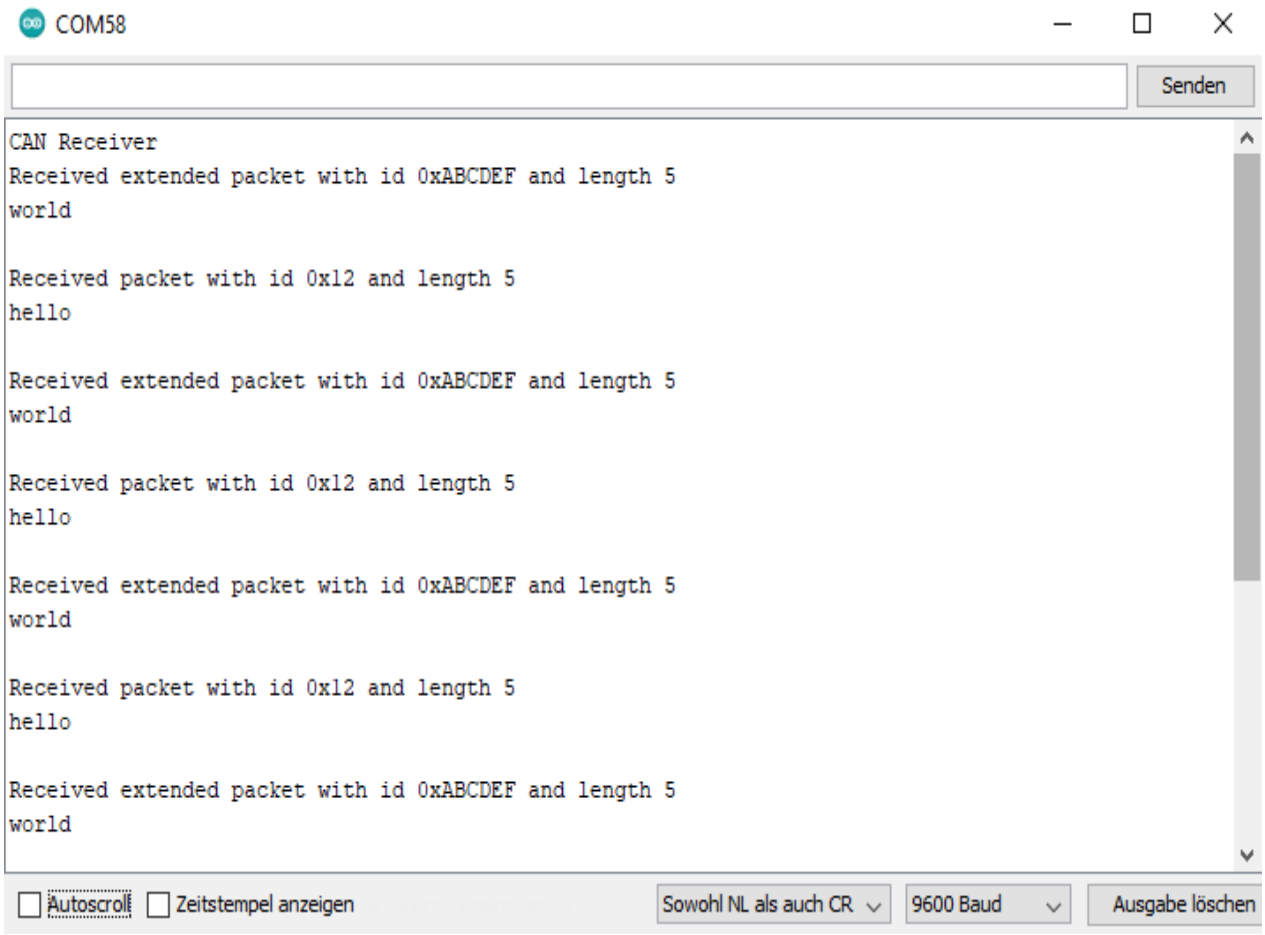


Receiver sketch

```
#include <CAN.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  Serial.println("CAN Receiver");

  // start the CAN bus at 500 kbps
  if (!CAN.begin(500E3)) {
    Serial.println("Starting CAN failed!");
    while (1);
  }
}
void loop() {
  // try to parse packet
  int packetSize = CAN.parsePacket();
  if (packetSize) {
    // received a packet
    Serial.print("Received ");
    if (CAN.packetExtended()) {
      Serial.print("extended ");
    }
    if (CAN.packetRtr()) {
      // Remote transmission request, packet contains no data
      Serial.print("RTR ");
    }
    Serial.print("packet with id 0x");
    Serial.print(CAN.packetId(), HEX);
    if (CAN.packetRtr()) {
```

```
      Serial.print(" and requested length ");
      Serial.println(CAN.packetDlc());
    } else {
      Serial.print(" and length ");
      Serial.println(packetSize);
      // only print packet data for non-RTR packets
      while (CAN.available()) {
        Serial.print((char)CAN.read());
      }
      Serial.println();
    }
    Serial.println();
  }
}
```

Upload the sketch to the Atmega328p and run the Serial Monitor (*Tools* > *Serial Monitor*). The result should look like as on the following image:



Now it is the time to learn and make your own projects. You can do that with the help of many example scripts and other tutorials, which can be found on the Internet.

If you are looking for the high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us