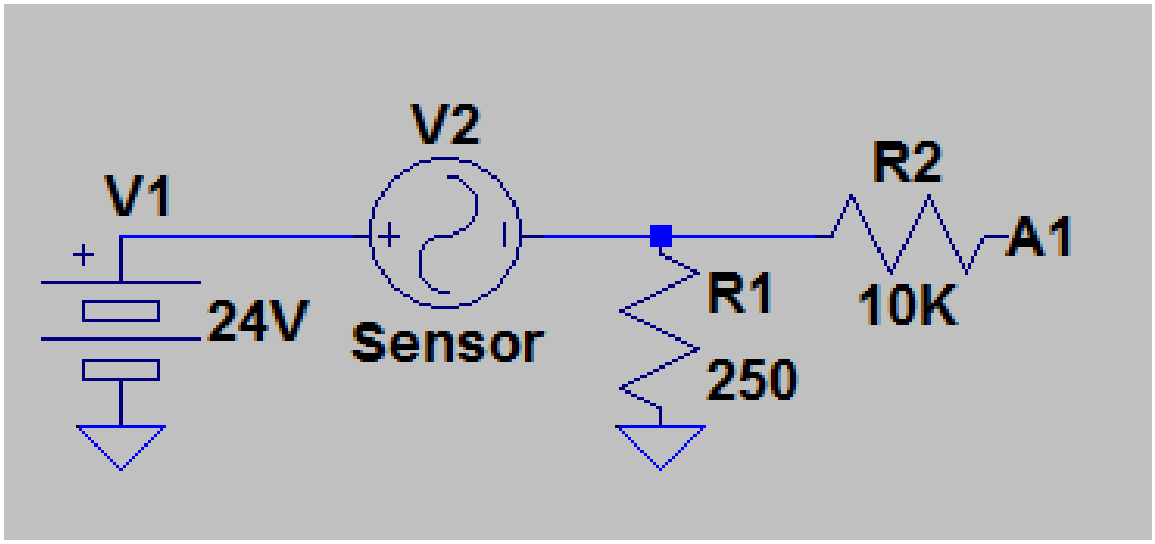
Reading 4-20mA Current Loop Sensors using Arduino

The sensor is designed to drive 4 to 20 mA through the output circuit regardless of the sensor supply voltage (within the allowed limits).

So, using a 250 Ohm resistor, Ohm's Law gives:
0.004A*250R = 1V
0.020A*250R = 5V

A better circuit includes a 10K resistor for overvoltage protection at the input, and has the negative sensor power supply connected to Arduino ground, as follows:



Reading 4-20mA Current Loop Sensors using Arduino

Reading 4-20mA current loop sensors using Arduino is much easier than you might think. Follow this simple guide and we will show you a few tips to make it fast and easy.

4-20mA current loop is most common and widely used communication method in an industrial environment. This 4-20mA current loop interface is also known as 2 wire interface technology. Despite being one of the oldest industry standards a lot of users have difficulty understanding the working of this technology. This 4-20mA current loop technology is used in temperature sensors, pressure sensors, current sensors, distance sensors, magnetic field sensors and much more.

A typical 4-20mA current loop setup contains 3 things

1. power supply — Most of the devices work at 24V DC but there are other voltage standards available as well
2. 4-20mA current loop sensor — this is the device which works as 4-20mA standard. for example, it could be a temperature sensor which gives the temperature value in the form of 4-20mA
3. 4-20mA current loop receiver — this is the device which will be used for readings 4-20mA current loop signal and will convert into digital or real world values

How two wire or 4-20mA current loop works
The 4-20mA current loop output devices give data output in the form of current consumption, to measure the change in the current we will need to put the correct receiver circuit in the series. so, for example, let's say you have a pressure sensor which works at 4-20mA current loop principle and it is capable of measuring 0-50psi. so when you connect the 4-20mA current receiver circuit in the series and read the sensor it will read 4ma when the pressure is 0 psi and it will read 20mA when the pressure is 50 psi. This conversion will be a linear conversion, so you will be able to calculate the real pressure value at any given point.

This 4-20mA current loop method is really popular in automation and sensing industry due to high reliability, easy installation, fewer components, and wires can be long. 4-20mA receiver devices could be really expensive but it can be done at a much lower cost using ncd.io 4-20mA current loop receiver boards.

Interfacing 4-20mA current loop receiver with Arduino

Hardware setup — For reading 4-20mA current loop sensors using Arduino you will need the following hardware

1.

[1-Channel 4-20mA Current Loop Receiver 16-Bit ADS1115 I2C Mini Module]

2. Arduino

The 4-20mA current loop receiver board has a 16 bit ADC, whit this high-resolution ADC you can get the best possible readings from your sensor. this board works over i2c communication, so using this board is really easy. it has one address bit, so you can connect up to two of these boards with one i2c master. The board has a voltage boost circuit and usages INA196 to monitor the current values. If you are looking for a high accuracy low-cost 4-20mA current loop receiver boards, this is your board.

To set up the hardware you will need to connect the Arduino to the i2c shield and use an i2c cable to connect 4-20mA current loop receiver board with Arduino i2c shield.

Software Setup–

The Arduino code for reading 4-20mA current loop is really easy and you can download the ADS1115 Arduino lib from here ADS1115 Arduino

```
1. /*************************************************************************/
2. /*
3.       This code can be used to read 4-20mA signal using ncd 4-20mA current
   loop board and arduino.
4.       this 4-20mA current loop board comes with 1,2,3,4 channels and 16 bit
   resolution
```

```
5.         Distributed with a free-will license.
6.         Use it any way you want, profit or free, provided it fits in the
   licenses of its associated works.
7.         ADS1115
8.         This code is designed to work with the ADS1115_I2CADC I2C Mini
   Module available from ControlEverything.com.
9.         https://www.controleverything.com/content/Analog-Digital-
   Converters?sku=ADS1115_I2CADC#tabs-0-product_tabset-2
10.       */
11.       /**********************************************************************
   ******/
12.
13.       #include <Wire.h>
14.       #include <ADS1115.h>
15.
16.       ADS1115 ads;
17.
18.       void setup(void)
19.       {
20.           Serial.begin(9600);
21.
22.           // The address can be changed making the option of connecting
   multiple devices
23.           ads.getAddr_ADS1115(ADS1115_DEFAULT_ADDRESS);   // 0x48, 1001 000
   (ADDR = GND)
24.           // ads.getAddr_ADS1115(ADS1115_VDD_ADDRESS);    // 0x49, 1001 001
   (ADDR = VDD)
25.           // ads.getAddr_ADS1115(ADS1115_SDA_ADDRESS);    // 0x4A, 1001 010
   (ADDR = SDA)
26.           // ads.getAddr_ADS1115(ADS1115_SCL_ADDRESS);    // 0x4B, 1001 011
   (ADDR = SCL)
27.
```

```
28.          // The ADC gain (PGA), Device operating mode, Data rate
29.          // can be changed via the following functions
30.
31.          ads.setGain(GAIN_TWO);          // 2x gain   +/- 2.048V  1 bit =
  0.0625mV (default)
32.          //ads.setGain(GAIN_TWOTHIRDS);  // 2/3x gain +/- 6.144V  1 bit =
  0.1875mV
33.          // ads.setGain(GAIN_ONE);       // 1x gain   +/- 4.096V  1 bit =
  0.125mV
34.          // ads.setGain(GAIN_FOUR);      // 4x gain   +/- 1.024V  1 bit =
  0.03125mV
35.          // ads.setGain(GAIN_EIGHT);     // 8x gain   +/- 0.512V  1 bit =
  0.015625mV
36.          // ads.setGain(GAIN_SIXTEEN);   // 16x gain  +/- 0.256V  1 bit =
  0.0078125mV
37.
38.          ads.setMode(MODE_CONTIN);       // Continuous conversion mode
39.          // ads.setMode(MODE_SINGLE);    // Power-down single-shot mode
  (default)
40.
41.          ads.setRate(RATE_128);          // 128SPS (default)
42.          // ads.setRate(RATE_8);         // 8SPS
43.          // ads.setRate(RATE_16);        // 16SPS
44.          // ads.setRate(RATE_32);        // 32SPS
45.          // ads.setRate(RATE_64);        // 64SPS
46.          // ads.setRate(RATE_250);       // 250SPS
47.          // ads.setRate(RATE_475);       // 475SPS
48.          // ads.setRate(RATE_860);       // 860SPS
49.
50.          ads.setOSMode(OSMODE_SINGLE);   // Set to start a single-
  conversion
51.
```

```
52.          ads.begin();
53.      }
54.
55.      void loop(void)
56.      {
57.          byte error;
58.          int8_t address;
59.
60.          address = ads.ads_i2cAddress;
61.          // The i2c_scanner uses the return value of
62.          // the Write.endTransmisstion to see if
63.          // a device did acknowledge to the address.
64.          Wire.beginTransmission(address);
65.          error = Wire.endTransmission();
66.          if (error == 0)
67.          {
68.              int16_t adc0, adc1, adc2, adc3;
69.
70.              Serial.println("Getting Single-Ended Readings from AIN0..3");
71.              Serial.println(" ");
72.              adc0 = ads.Measure_SingleEnded(0);
73.              Serial.print("Digital Value of Analog Input at Channel 1: ");
74.              Serial.println(adc0);
75.              float mACurrent = adc0 * 0.000628;
76.              Serial.print("Current Loop Input at Channel 1: ");
77.              Serial.println(mACurrent,3);
78.      //        adc1 = ads.Measure_SingleEnded(1);
79.      //        Serial.print("Digital Value of Analog Input at Channel 2:
   ");
80.      //        Serial.println(adc1);
81.      //        adc2 = ads.Measure_SingleEnded(2);
```

```
82.       //        Serial.print("Digital Value of Analog Input at Channel 3:
   ");
83.       //        Serial.println(adc2);
84.       //        adc3 = ads.Measure_SingleEnded(3);
85.       //        Serial.print("Digital Value of Analog Input at Channel 4:
   ");
86.       //        Serial.println(adc3);
87.       //        Serial.println(" ");
88.       //        Serial.println("        ************************
   ");
89.       //        Serial.println(" ");
90.           }
91.           else
92.           {
93.               Serial.println("ADS1115 Disconnected!");
94.               Serial.println(" ");
95.               Serial.println("        ************        ");
96.               Serial.println(" ");
97.           }
98.
99.           delay(1000);
100.      }
```

Converting ADC values into 4-20mA values

```
1.          float mACurrent = adc0 * 0.000628;
2.          Serial.print("Current Loop Input at Channel 1: ");
3.          Serial.println(mACurrent,3);
```

This is how we calculate the 4-20mA multiplication factor — we set the loop simulator at 4mA and connect it to the 4-20mA current loop receiver board and take the analog readings, in this case at 4mA the ADC reads around 6392. by using this two known entity we can calculate the 4-20mA calculation factor by simply dividing 4mA by 6392.

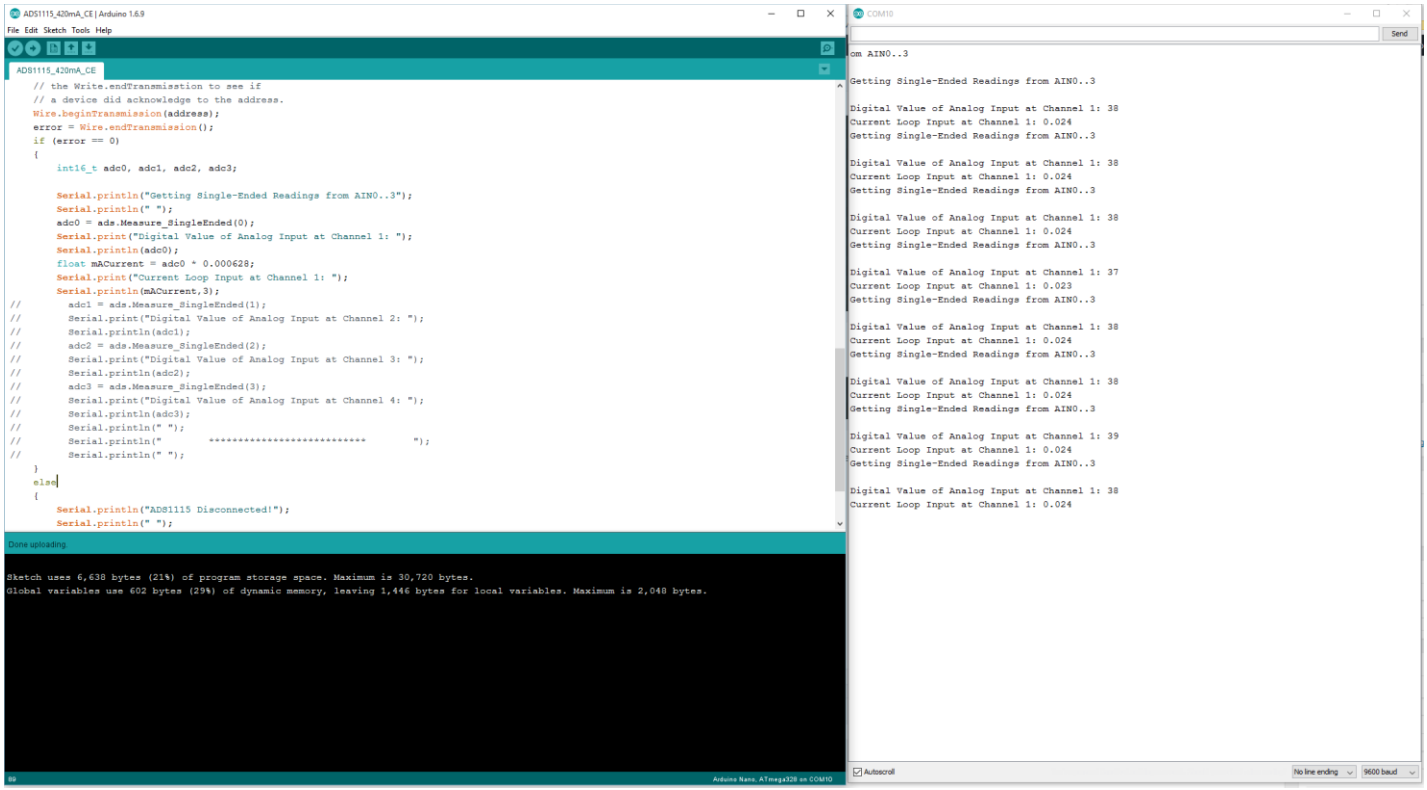
current multiplication factor = 4mA/6392 === 0.000628.

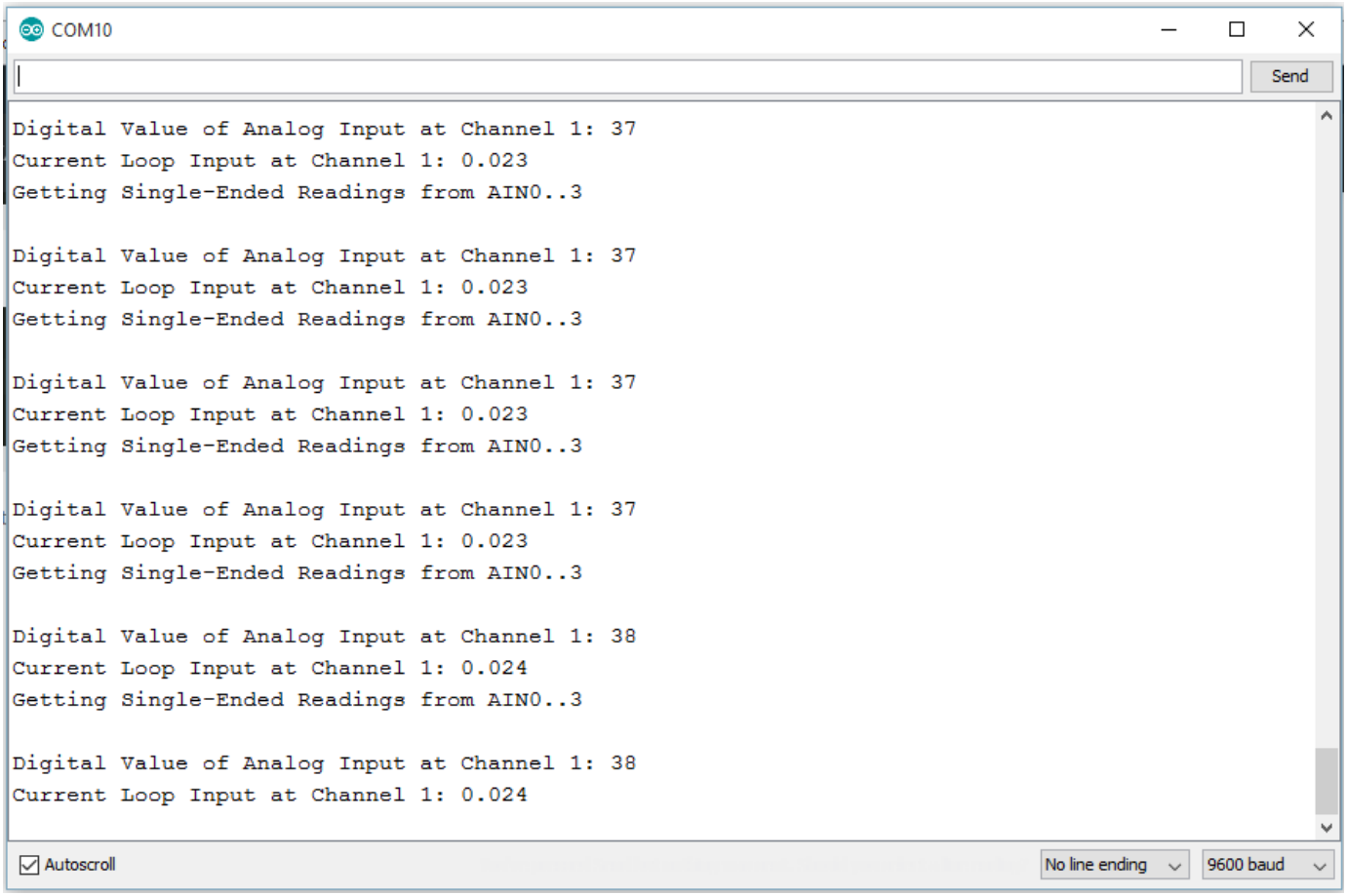
Testing 4-20mA current loop receiver board using Loop simulator —

Loop simulator is one of the most common tool used to debug 4-20mA devices. We will use this tool to verify the accuracy of our setup and to tune it.

Step 1. Turn on the loop simulator and select Current 2-Wire option

Once you select that, it will bring the twp-wire window. When it's not connected to the 4-20mA receiver board it will display open wire.

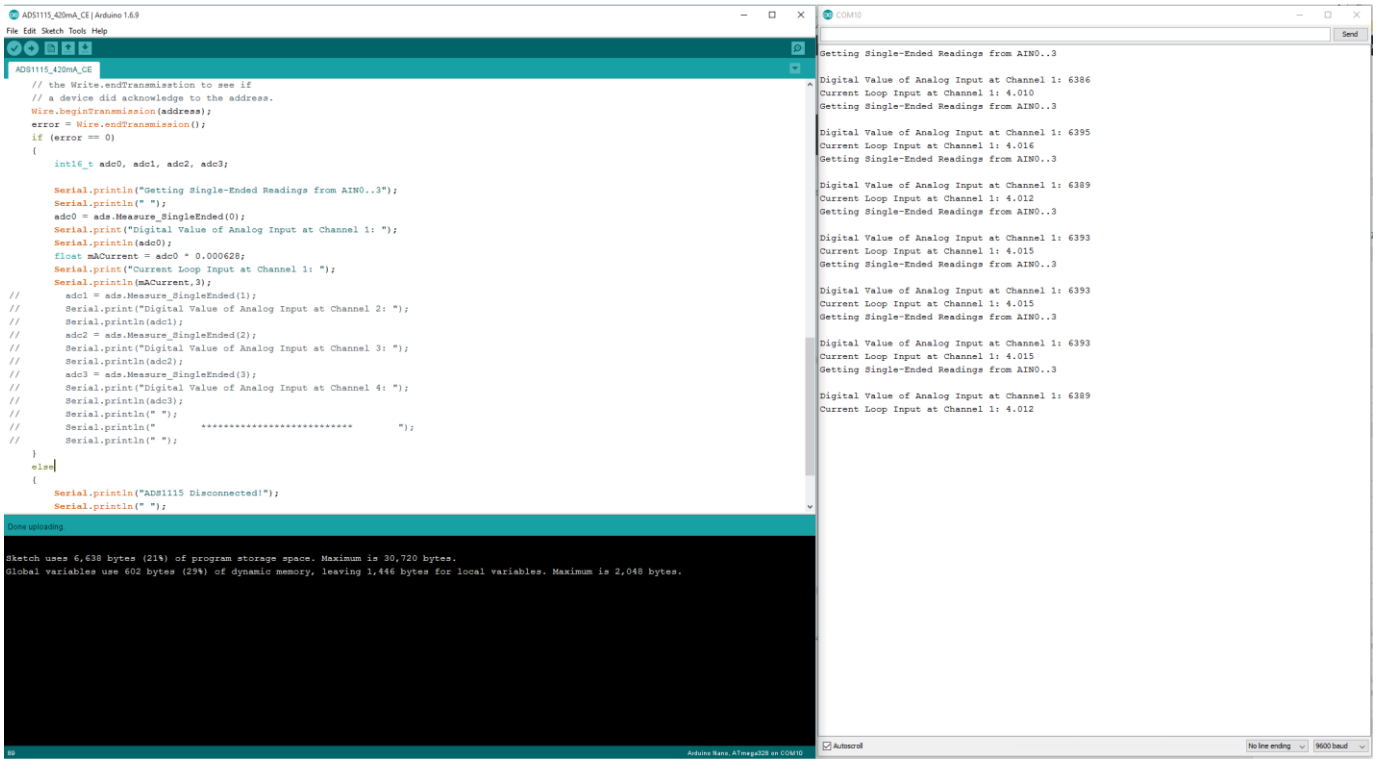


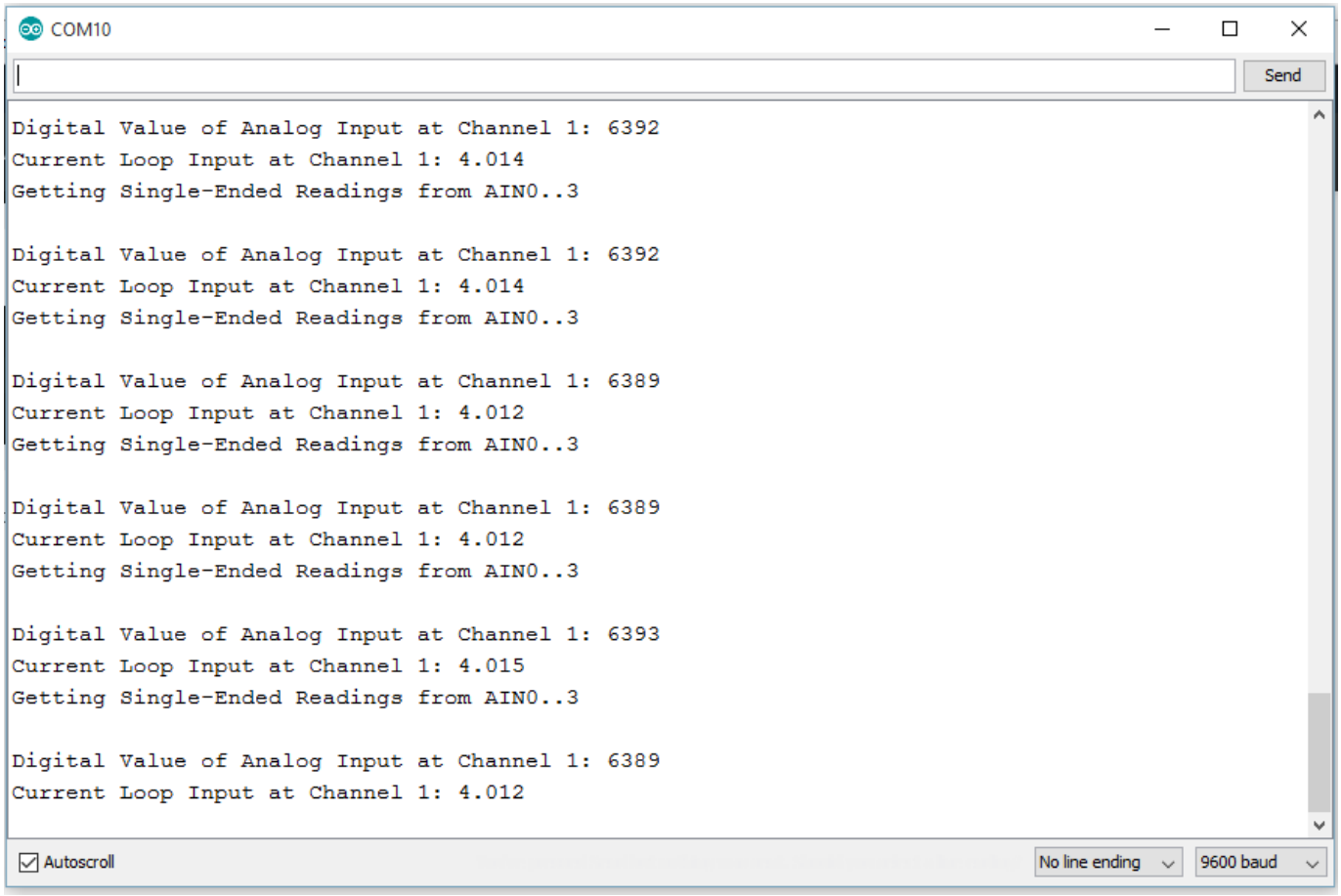


When the loop simulator is not connected to the receiver board, the Arduino will read 0mA, this is also used to detect is the sensor wiring is broken. This is one of most important thing you will need to understand while reading 4-20mA current loop sensors.

Step 2. The loop simulator has two probes, the black probe is named as Ext.PS+ and the red probe is named as PLC input. In our setup, we will connect the black probe to in IN1 terminal on the 4-20mA receiver board and the red probe to gnd. This is very important if you make the wrong connection it could damage your 4-20mA current loop receiver board. Once you have the correct connection the 4-20mA current loop board will start readings 4-20mA current loop signal.

By default, the loop simulator will be set at 4mA and when you connect to Arduino, it will read 4mA


ADS1115_420mA_CE | Arduino 1.6.9
File Edit Sketch Tools Help
ADS1115_420mA_CE
// the Write.endTransmisstion to see if
// a device did acknowledge to the address.
Wire.beginTransmission(address);
error = Wire.endTransmission();
if (error == 0)
{
int16_t adc0, adc1, adc2, adc3;
Serial.println("Getting Single-Ended Readings from AIN0..3");
Serial.println(" ");
adc0 = ads.Measure_SingleEnded(0);
Serial.print("Digital Value of Analog Input at Channel 1: ");
Serial.println(adc0);
float mACurrent = adc0 * 0.000628;
Serial.print("Current Loop Input at Channel 1: ");
Serial.println(mACurrent,3);
// adc1 = ads.Measure_SingleEnded(1);
// Serial.print("Digital Value of Analog Input at Channel 2: ");
// Serial.println(adc1);
// adc2 = ads.Measure_SingleEnded(2);
// Serial.print("Digital Value of Analog Input at Channel 3: ");
// Serial.println(adc2);
// adc3 = ads.Measure_SingleEnded(3);
// Serial.print("Digital Value of Analog Input at Channel 4: ");
// Serial.println(adc3);
// Serial.println(" ");
// Serial.println(" ********** ");
// Serial.println(" ");
}
else
{
Serial.println("ADS1115 Disconnected!");
Serial.println(" ");
Done uploading.
Sketch uses 6,638 bytes (21%) of program storage space. Maximum is 30,720 bytes.
Global variables use 602 bytes (29%) of dynamic memory, leaving 1,446 bytes for local variables. Maximum is 2,048 bytes.
Arduino Nano, ATmega328 on COM10
COM10
Send
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6386
Current Loop Input at Channel 1: 4.010
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6395
Current Loop Input at Channel 1: 4.016
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6389
Current Loop Input at Channel 1: 4.012
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6393
Current Loop Input at Channel 1: 4.015
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6393
Current Loop Input at Channel 1: 4.015
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6393
Current Loop Input at Channel 1: 4.015
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6389
Current Loop Input at Channel 1: 4.012
Autoscroll
No line ending
9600 baud


COM10
Send
Digital Value of Analog Input at Channel 1: 6392
Current Loop Input at Channel 1: 4.014
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6392
Current Loop Input at Channel 1: 4.014
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6389
Current Loop Input at Channel 1: 4.012
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6389
Current Loop Input at Channel 1: 4.012
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6393
Current Loop Input at Channel 1: 4.015
Getting Single-Ended Readings from AIN0..3
Digital Value of Analog Input at Channel 1: 6389
Current Loop Input at Channel 1: 4.012
Autoscroll
No line ending
9600 baud

Interfacing Isolated 4-20mA Current Loop Transmitter Arduino

In our last post we interfaced 4-20mA current loop receiver with the arduino, in this article we will learn how you can interface isolated 4-20mA current loop transmitter board with arduino. 4-20mA transmitter devices are used to provide 4-20mA signal which can be used to control pumps, valves, switches and transducers. For example lets say you want to want to control a sprinkler which works on 4-20mA current loop signal, in that case you will need a 4-20mA current loop transmitter board. To control such a sprinkler you will change the current output using arduino and this current output will control the sprinkler.

The 4-20mA current loop is also known as 2 wire protocol, in such setups there are two wires which control the device as well as provide the power using same two wires. The sensor or device values are changed by varying the current value.

Hardware

1. Arduino
2. Arduino I2C shield
3. 4-20mA current Loop Transmitter

This isolated 4-20mA current loop transmitter has a 12 bit DAC which provides high resolution 4-20mA output. On the one channel 4-20mA transmitter the DAC is MCP4725. In this post we will be using this one and the 4 channel will work in same way. This one channel isolated 4-20mA current loop transmitter board has 1kV power isolation and also has i2c communication isolation(2500Vrms). The board works on i2c communication and it has 1 address bit. By chaining the address bit its possible to connect two 4-20mA current loop transmitter boards together.

It is real simple to interface these isolated 4-20mA current loop receiver boards with arduino, raspberry pi, usb or any other computer, You can follow the exact same steps and can make it work with any of the platforms. To connect the arduino with 4-20mA current loop transmitter board all you will need to do is insert the arduino module into the arduino i2c shield and use an i2c cable to connect these two boards together.

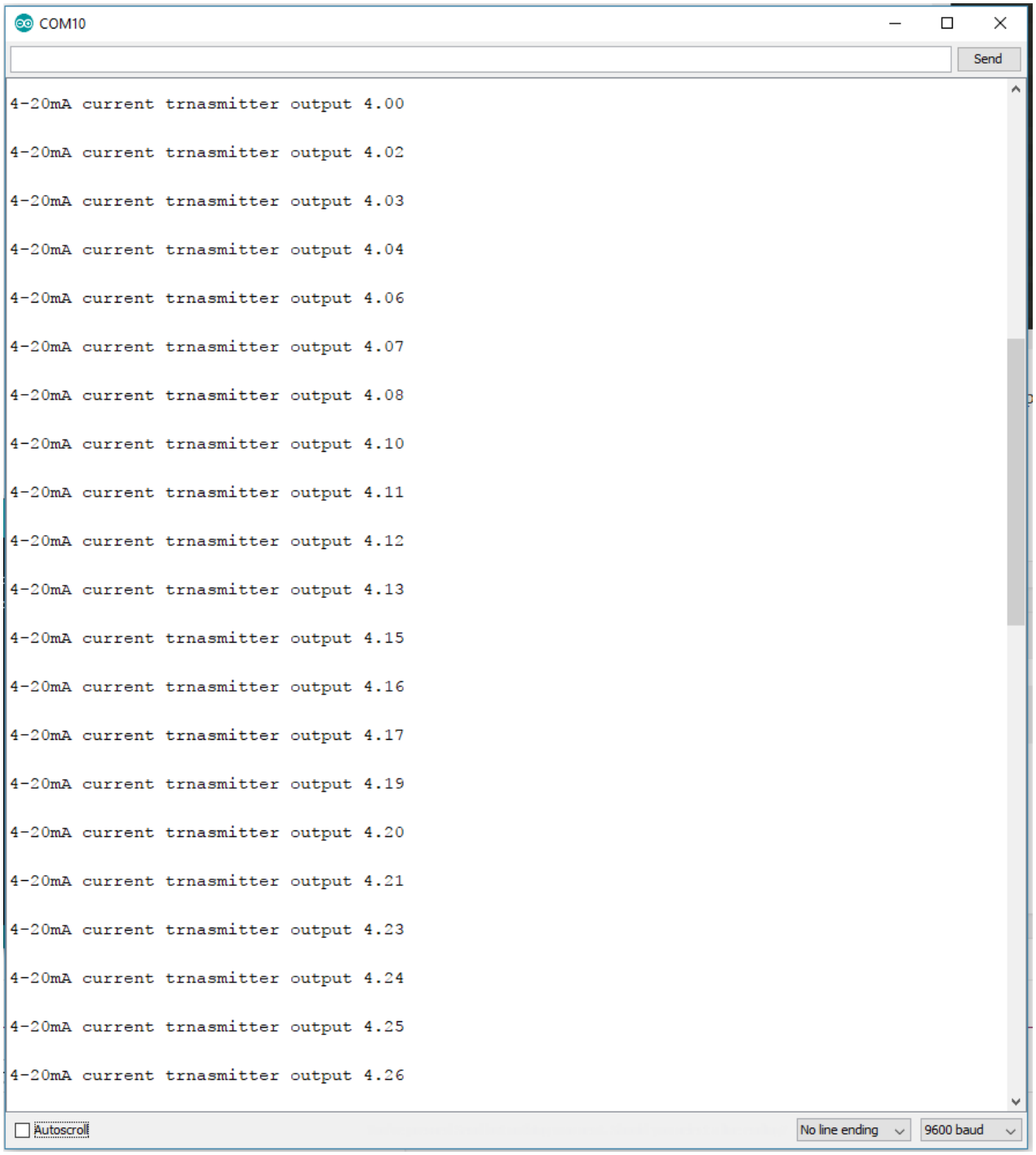
The isolated 4-20mA current loop transmitter board has a 2 connection screw terminal which provides 4-20mA output, over here you can connect the device or sensor you want control. You will also need a power supply to power up the device. Once you have all this setup, we can start working on the code.

The Arduino code can be found here

```
#include <Wire.h>
void setup()
{
  Serial.begin (9600);
  Wire.begin();

}
void loop()
{
      /// at DAC vlaue 290 the current output will be around 4mA and
      ///at DAC vlaue 1500 the current output will be around 20mA
      /// you can change these values to tune the 4-20mA output
       for (int i=290; i <= 1500; i++)
        {
        Wire.beginTransmission(0x60 );
        Wire.write(64);
       // Wire.write(64);
        Wire.write(i >> 4);        // 8 MSB
        Wire.write((i & 15) << 4); //4 LSB
        delay(100);
        Serial.print("4-20mA current trnasmitter output ");
        Serial.println(i*0.013);
        Serial.print("n");
        Wire.endTransmission();
        }
      }
```

This is real simple example code, in this code we are changing the DAC values and the current loop output will change accordingly. You can tune these values according to your application requirements.


COM10
Send
4-20mA current trnasmitter output 4.00
4-20mA current trnasmitter output 4.02
4-20mA current trnasmitter output 4.03
4-20mA current trnasmitter output 4.04
4-20mA current trnasmitter output 4.06
4-20mA current trnasmitter output 4.07
4-20mA current trnasmitter output 4.08
4-20mA current trnasmitter output 4.10
4-20mA current trnasmitter output 4.11
4-20mA current trnasmitter output 4.12
4-20mA current trnasmitter output 4.13
4-20mA current trnasmitter output 4.15
4-20mA current trnasmitter output 4.16
4-20mA current trnasmitter output 4.17
4-20mA current trnasmitter output 4.19
4-20mA current trnasmitter output 4.20
4-20mA current trnasmitter output 4.21
4-20mA current trnasmitter output 4.23
4-20mA current trnasmitter output 4.24
4-20mA current trnasmitter output 4.25
4-20mA current trnasmitter output 4.26
Autoscroll
No line ending
9600 baud