AMS 5915 AN01
Data readout of AMS 5915 pressure sensors with Arduino Uno

This application note describes a simple method to read data from AMS 5915's I2C bus with Arduino Uno. A special PCB is used to interface the AMS 5915 [1] to Arduino Uno [2] as well as a short demo C source code and a library for Arduino's integrated development environment (IDE) [3].

Arduino Uno is a popular microcontroller development board featuring an Atmega microcontroller and several I/O-pins including an SPI as well as an I2C port and is often used to read data from sensors. As Arduino Uno's I2C port operates at 5 V it is not easy to read date from AMS 5915, a board-mount pressure sensor operating at 3.3 V, without further external components for level conversion.

To simplify the connection of the AMS 5915 pressure sensor with an Arduino board Analog Microelectronics offers the shield *AMS 5915 – Arduino PCB* (available at www.analog-micro.com), which can easily be mounted onto Arduino Uno's pin sockets and provides all the connections needed to power AMS 5915, do the level conversion on the SDA and SCL line and read data from the sensor. As soon as an AMS 5915 is mounted into the PCB's socket and the PCB is connected to Arduino Uno the system is ready to use (see *Figure 1*).

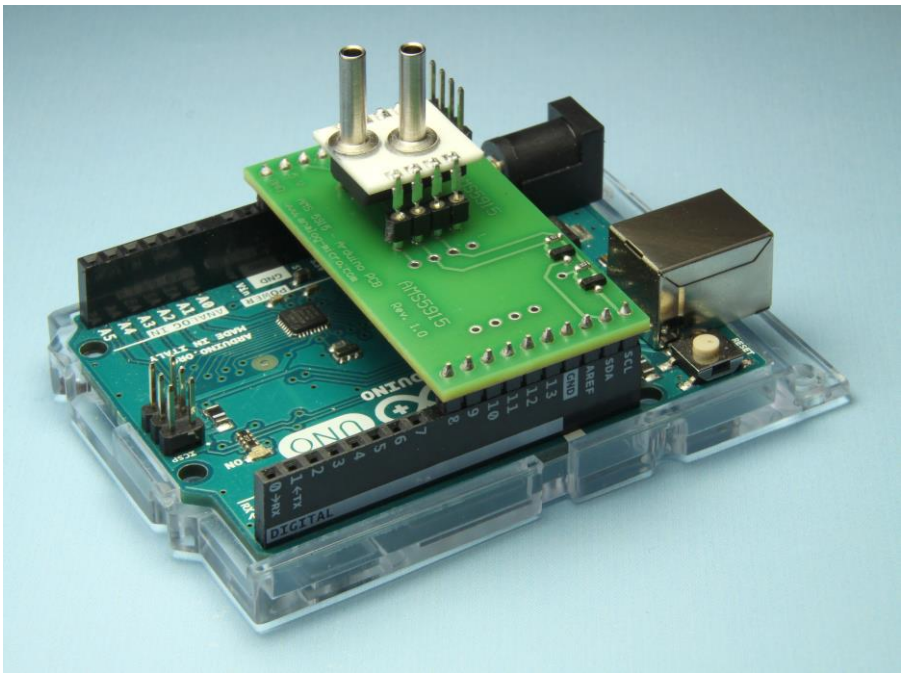

Figure 1: **Arduino Uno with AMS 5915 - Arduino PCB and an AMS 5915 pressure sensor**

Furthermore the *AMS 5915 – Arduino PCB* provides pins to connect another device to the 3.3 V I2C bus on the PCB and a second solder position for a second AMS 5915.

Please note: If two or more AMS 5915 pressure sensors are connected to the same I2C bus the sensors have to respond to different I2C addresses. Individual I2C addresses can be programmed using AMS 5915's starter kit [4] or the pressure sensors can be purchased with pre-programmed individual addresses from Analog Microelectronics.

Figure 2 illustrates the principle electrical connections which are needed for readout of AMS 5915 using an Arduino and established by the *AMS 5915 – Arduino PCB*. Arduino's 3.3 V voltage source and GND are connected to AMS 5915's corresponding pins directly. As AMS 5915's I2C bus operates at 3.3 V and Arduino Uno's bus works at 5 V a level converter has to be used to separate both voltage levels and additional pull-up resistors have to be added on the 3.3 V side. Due to Arduino's internal pull-up resistors its SDA and SCL ports can be connected to the level converter directly.

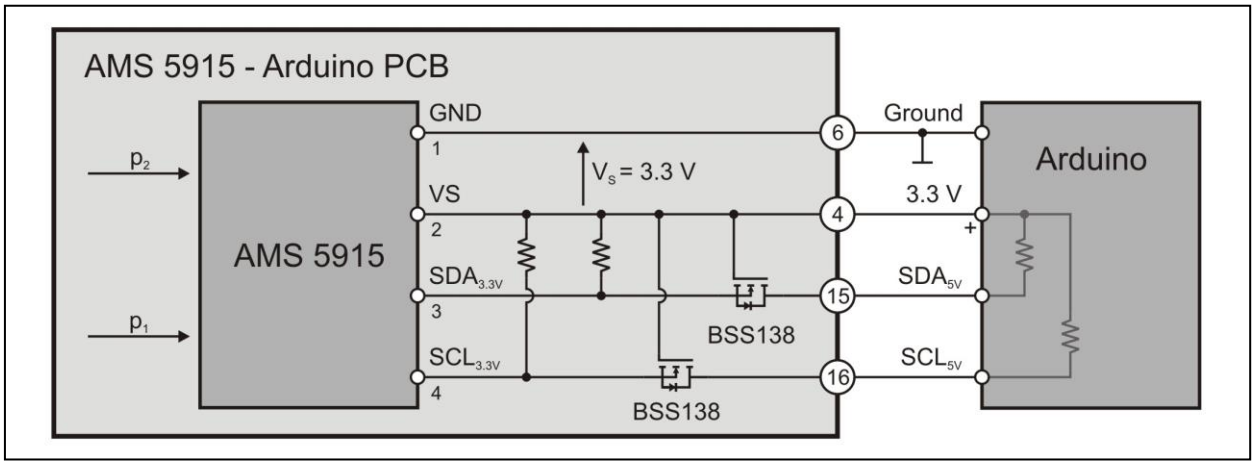

Figure 2: **Connecting AMS 5915 to Arduino**

A simple bidirectional level shifter can be realized using an N-channel enhancement MOSFET (e.g. BSS138) with a threshold voltage $V_{GS(th)}$ < 3.3 V. The gate is connected to Arduino's 3.3 V supply voltage, the source is connected either to AMS 5915's SCL or SDA pin and the drain to Arduino Uno's respective SCL or SDA pin. Therefore two MOSFETs are needed: one for the SCL line and one for the SDA line (for more information see [5]).

For bidirectional level conversion using N-channel enhancement MOSFETs three cases have to be considered:

1. As long as both the 3.3 V and the 5 V part of the level converter are pulled up to their respective HIGH level V_{GS} is 0 V leading to the MOSFET being not conducting and both sides staying on their HIGH level.
2. If the level converter's 3.3 V side is pulled down to LOW level V_{GS} becomes 3.3 V, the MOSFET starts to conduct and the 5 V side is also pulled down to LOW level.
3. In the case that the 5 V side is pulled down to LOW level the drain substrate diode pulls the 3.3 V side down until V_{GS} is larger than $V_{GS(th)}$ and the MOSFET begins to conduct leading to both sides being pulled down to LOW level.

Using those two MOSFET level shifters the I2C communication between AMS 5915 and Arduino Uno can be established although they operate at different voltages.

C Source Code for Data Readout

Using the following source code pressure and temperature data can be read from AMS 5915. The code uses the AMS library for Arduino's IDE provided by Analog Microelectronics for an easy readout of AMS 5915's data and their conversion into physical units (e.g. mbar, Pa, PSI…). This library's reference can be found on page 4.

```
//include the AMS library

#include <AMS.h>


//declaration of the used variables. readSensor, readPressure and readTemperature return a float value containing the current pressure value in the given pressure unit and the current temperature in degree Celsius, so Pressure and Temperature have to have the type float.

float Pressure;

float Temperature;

String DataString;

char PrintData[48];


//define the sensor's instance with the sensor's family, sensor's I2C address as well as its specified minimum and maximum pressure

AMS AMSa(5915, 0x28,0,2068);

//initialization function which will be executed only once, when the Arduino is powered up

void setup() {

	//set the serial port's baud rate. The COM port's standard parameters are: Data bits: 8, Parity: none, Stop bits: 1, Flow control: none, Baud rate: 9600

	Serial.begin(9600);

}


//main function, which will be repeated continuously until Arduino is resetted or removed from its power source and which contains the data readout from AMS 5915

void loop() {

	//Option 1: read pressure and temperature values

	//check if AMS 5915 responds to the given I2C address

	if (AMSa.Available() == true) {

		//read the sensor's temperature and pressure data. Please note that the temperature is measured inside the sensor's package and contains the sensor's self heating as well as the ambient temperature

		AMSa.readSensor(Pressure, Temperature);

		//check if an error occurred leading to the function returning a NaN

		if (isnan(Pressure) || isnan(Temperature)) {

			//write an error message on the serial port
```

```
                Serial.write("Please check the sensor family name.");
        } else {
                //put the data into a string
                DataString = String(Pressure) + " mbar " + String(Temperature) + " "
                + (char)176 + "C \n";
                //convert string into CharArray
                DataString.toCharArray(PrintData, 48);
                //write the sensor's data on the serial port
                Serial.write(PrintData);}
} else {
        //tell the user that there is something wrong with the communication be-
        tween Arduino and the sensor.
        Serial.write("The sensor did not answer.");}
//wait for 1 s until reading new data from the Sensor
delay(1000);

//Option 2: read pressure values only
if (AMSa.Available() == true) {
        //read AMS 5915's pressure data only
        Pressure = AMSa.readPressure();
        if (isnan(Pressure)) {
                Serial.write("Please check the sensor family name.");
        } else {
                DataString = String(Pressure) + " mbar \n";
                DataString.toCharArray(PrintData, 48);
                Serial.write(PrintData);}
} else {
        Serial.write("The sensor did not answer.");}
delay(1000);

//Option 3: read temperature values only
if (AMSa.Available() == true) {
        //read AMS 5915's temperature data in degree celsius
        Temperature = AMSa.readTemperature();
        if (isnan(Temperature)) {
                Serial.write("Please check the sensor family name."); //
        } else {
                DataString = String(Temperature) + " " + (char)176 + "C \n";
```

```
                    DataString.toCharArray(PrintData, 48);
                    Serial.write(PrintData);}
        } else {
              Serial.write("The sensor did not answer.");}
        delay(1000);
}
```

Reference for the AMS Arduino library

Although AMS 5915's data can be read from its I2C interface using Arduino's wire library, this method is prone to errors especially during readout and the conversion of the received data from arbitrary units into physical units. Therefore Analog Microelectronics provides the library "AMS" for Arduino's IDE. To use this library it has to be downloaded from www.analog-micro.com, unpacked into the Arduino/library folder and Arduino's IDE has to be restarted. Afterwards the library will be available at "Sketch -> Include Library -> Contributed libraries".[1]

The library has to be included into the source code with:

```
#include <AMS.h>
```

The "AMS" library contains the following elements:

The mandatory initialization sequence:

```
AMS SensorName(int sensor family, int I2C address, int minimum pressure in mbar or psi, int maximum pressure in mbar or psi);
```

contains the sensor family, the I2C address as well as the minimum and maximum pressure, which have to be given in the same physical units (e.g. `AMS AMSa(5915, 0x28, 0, 100);`), otherwise the conversion will not work correctly.

and the following functions:

```
bool Available()
```

is used to check, if AMS 5915 responds to the given I2C address. This function returns false, if the sensor does not answer correctly, or true, if the communication with the sensor works fine.

```
void readSensor(float pressure, float temperature)
```

returns the current pressure measured by AMS 5915 in the pressure unit given during initialization as well as the temperature measured at AMS 5915's pressure sensing element in °C. If an error occurred pressure and temperature will contain the value NaN.

```
float readPressure()
```

is used to read AMS 5915's pressure data only. The function returns the current pressure in the pressure unit given during initialization or NaN if an error occurred.

```
float readTemperature()
```

is used to read AMS 5915's temperature data only. The returned float value contains AMS 5915's sensing element's current temperature in °C or NaN if an error occured.

[1] The library "AMS" supports the pressure and temperature readout from the sensor families AMS 5812, AMS 5915, and AMS 6915.

References:

1.) AMS 5915's data sheet (see http://www.analog-micro.com)
2.) Arduino Uno (https://www.arduino.cc/en/Main/ArduinoBoardUno)
3.) Aduino's IDE (https://www.arduino.cc/en/Main/Software)
4.) AMS 5915's starter kit (see http://www.analog-micro.com)
5.) AN10441 (https://www.nxp.com/docs/en/application-note/AN10441.pdf)
6.) AMS Arduino library "AMS.zip" (see http://www.analog-micro.com)
7.) AMS 5915 – Arduino PCB (available at http://www.analog-micro.com)