AMS 5812 AN01
Interfacing AMS 5812 to an Arduino Uno

This application note describes, how to interface the OEM pressure sensor AMS 5812 [1] with an Arduino Uno development board [2]. For easy data readout via I2C Analog Microelectronics presents a special PCB connecting the AMS 5812 to an Arduino Uno as well as a short demo C source code and a library for Arduino's integrated development environment (IDE) [3].

The 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. To interface an AMS 5812 pressure sensor with an Arduino board Analog Microelectronics has designed a special PCB, the AMS 5812 – Arduino PCB (available at www.analogmicro.de). This PCB can easily be mounted onto Arduino Uno's pin sockets and features all the connections needed to power AMS 5812 and read data from its I2C port without soldering. After inserting an AMS 5812 into the PCB's socket the system is ready to use (see *Figure 1*).

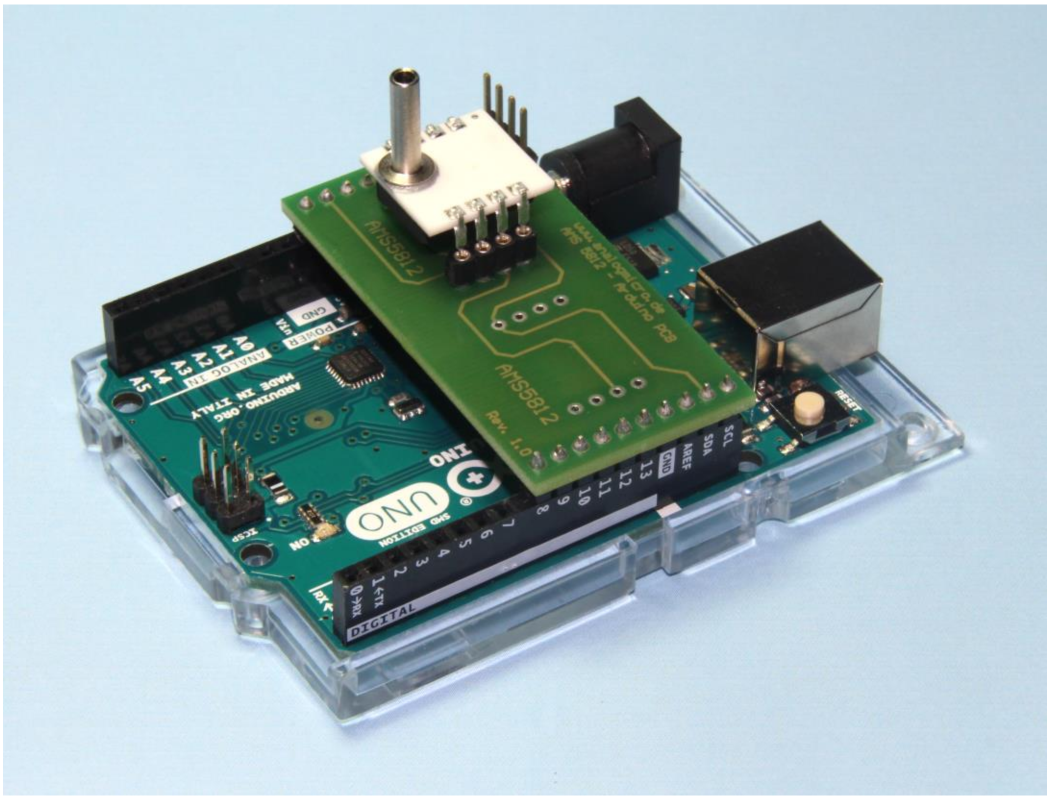

Figure 1: **Arduino Uno with AMS 5812 - Arduino PCB and an AMS 5812 pressure sensor**

Furthermore the AMS 5812 – Arduino PCB provides pins to connect another device to the I2C bus via cable and a second port, which can be used to solder a second AMS 5812 to the PCB.

Please note: If two AMS 5812 are used in parallel they have to respond on different I2C addresses (individual I2C addresses can be set using AMS 5812's starter kit [4] or sensors with pre-programmed individual addresses can be purchased from Analog Microelectronics. 0x78$_{HEX}$ cannot be used for either of the sensors then).

Figure 2 illustrates the principle electrical connections needed for readout of AMS 5812 using an Arduino and which are established by the AMS 5812 – Arduino PCB. Due to Arduino's internal pull-up resistors AMS 5812's SDA and SCL pins can be connected directly to Arduino's corresponding SDA and SCL ports on the PCB. Arduino's 5 V voltage source and GND are also connected to AMS 5812's corresponding pins directly.

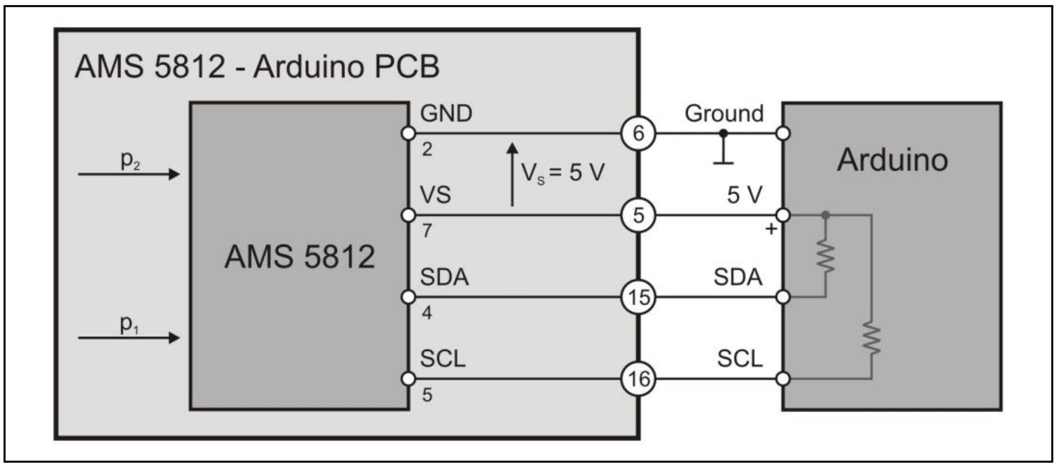

Figure 2: **Connecting AMS 5812 to Arduino**

C Source Code for Data Readout

The following source code can be used to read pressure and temperature data from AMS 5812. For an easy readout of AMS 5812's data and their conversion into physical units (e.g. mbar, Pa, PSI…) the code uses the AMS library for Arduino's IDE provided by Analog Microelectronics. The reference for this library can be found on page 4.

```
//include the AMS library
#include <AMS.h>


//declaration of the used variables, the variables pressure and temperature have to be
float since readSensor, readPressure and readTemperature return a float value containing
the current pressure value in the given pressure unit and the current temperature in de-
gree celsius.
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(5812, 0x78,0,2068);
//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 5812
void loop() {
	//Option 1: read pressure and temperature values
	//check if AMS 5812 responds to the given I2C address
	if (AMSa.Available() == true) {
		//read the sensor's temperature and pressure data. Please note that the tem-
		perature 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 between
		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 5812'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 5812'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 5812's data can be readout 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.analogmicro.de, 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, otherwise the conversion will not work correctly. (e.g. `AMS AMSa(5812, 0x78, 0, 100);`)

and the following functions:

```
bool Available()
```

is used to check, if AMS 5812 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.

[1] In principle the library "AMS" supports the pressure and temperature readout from the sensor families AMS 5812, AMS 5915 and AMS 6915. But AMS 5915 and AMS 6915 can only be connected to Arduino Uno, if level shifters are used.

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

returns the current pressure measured by AMS 5812 in the pressure unit given during initialization as well as the temperature measured at AMS 5812'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 5812'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 5812's temperature data only. The returned float value contains AMS 5812's sensing element's current temperature in °C or NaN if an error occured.

References:

1.) AMS 5812's data sheet (see http://www.analogmicro.de)
2.) Arduino Uno (https://www.arduino.cc/en/Main/ArduinoBoardUno)
3.) Aduino's IDE (https://www.arduino.cc/en/Main/Software)
4.) AMS 5812's starter kit (see http://www.analogmicro.de)
5.) AMS Arduino library "AMS.zip" (see http://www.analogmicro.de)
6.) AMS 5812 – Arduino (available at http://www.analogmicro.de)