Application Note AN-102: Arduino I2C Interface to K-30 Sensor

Introduction

The Arduino UNO, MEGA 1280 or MEGA 2560 are ideal microcontrollers for operating SenseAir's K-30 CO2 sensor. The connection to the K-30 is referred to I2C or TWI (Two Wire Interface). We recommend using the Arduino software Graphical User Interface (GUI).

If you are new to Arduino these low cost development boards are available from many sources. We recommend you start with genuine Arduino products for dependable results.

Run the Blink Example

The best way to become familiar with the GIU or to verify your Arduino board is operating properly is to create an Arduino project and run the example **Blink**. This simple test program confirms that a number of connection details and the GUI are working properly.

Caution: Do not connect the Arduino board to the USB port until the Arduino software is installed. Otherwise Windows will install a generic driver and the Arduino will not operate.

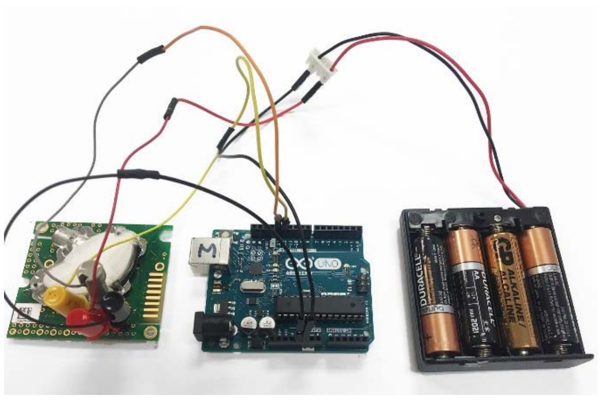
Step 1: Install Arduino software on your computer. From this page select the **Windows Installer.** https://www.arduino.cc/en/Main/Software

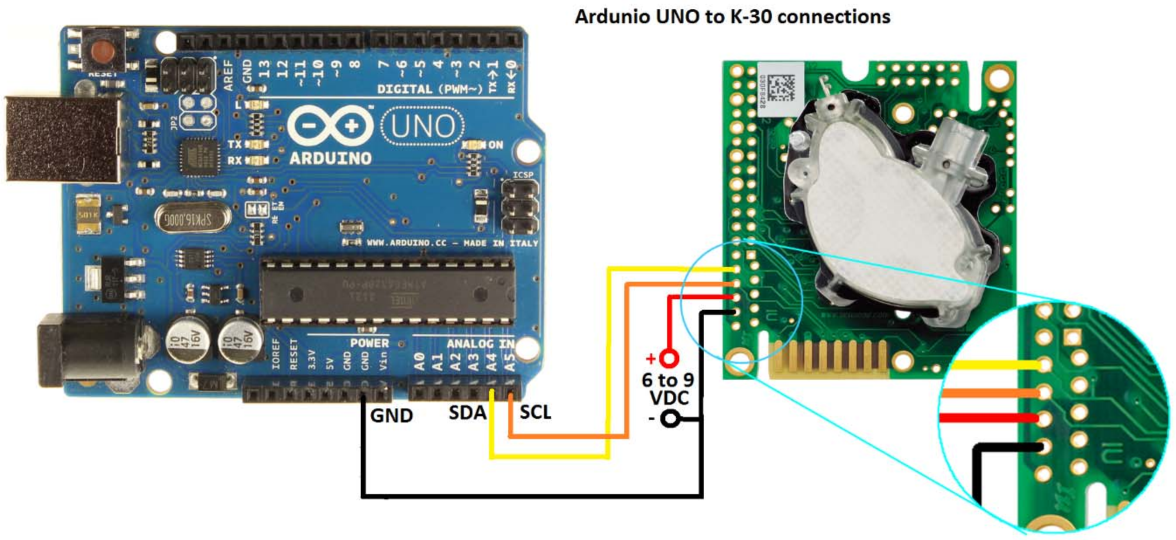
Step 2: To run the Blink example follow these instructions: https://www.arduino.cc/en/Tutorial/Blink

Connect the K30 Sensor

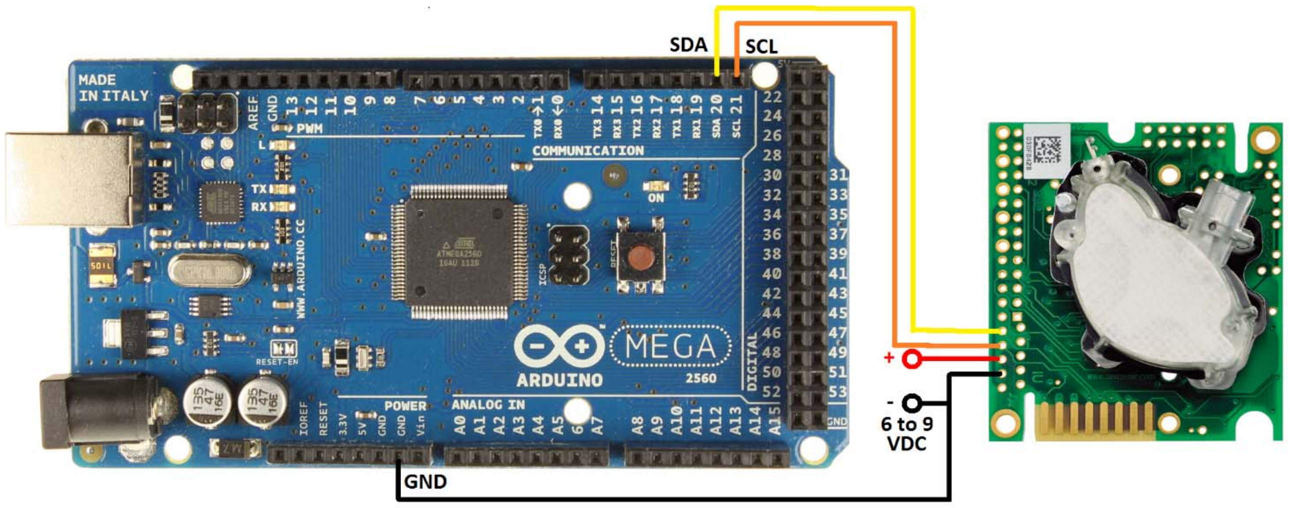
Refer to the wiring diagrams below for the Arduino UNO or MEGA. The connections for Arduino MEGA 1280 are identical to the Mega 2560.

Important: A 6-9 VDC, 500 ma or greater, external power supply is required. The K-30 sensor draws 300+ mA when sampling. Using the voltage off the Arduino board from USB will give erratic results. Connect the power supply as shown.





K-30 sensor I2C connection to Arduino UNO



K-30 sensor uses same pin connections to Arduino Mega 2560 or 1280

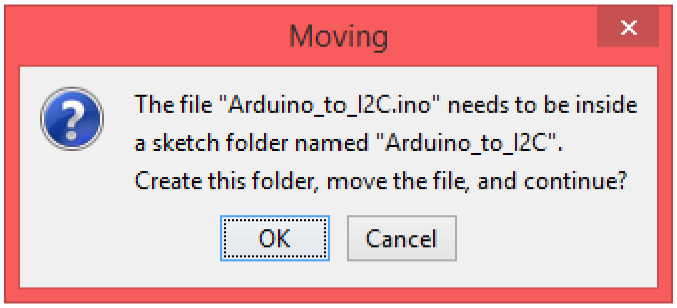
CO2Meter.com
CO2 Measurement Specialists

Creating an Arduino Project

1. Check that your Arduino is connected to your USB port and on-board LED indicates power is on.
2. Click on or copy/paste this link to the zip file that contains the **Arduino AN-102.ino** file.

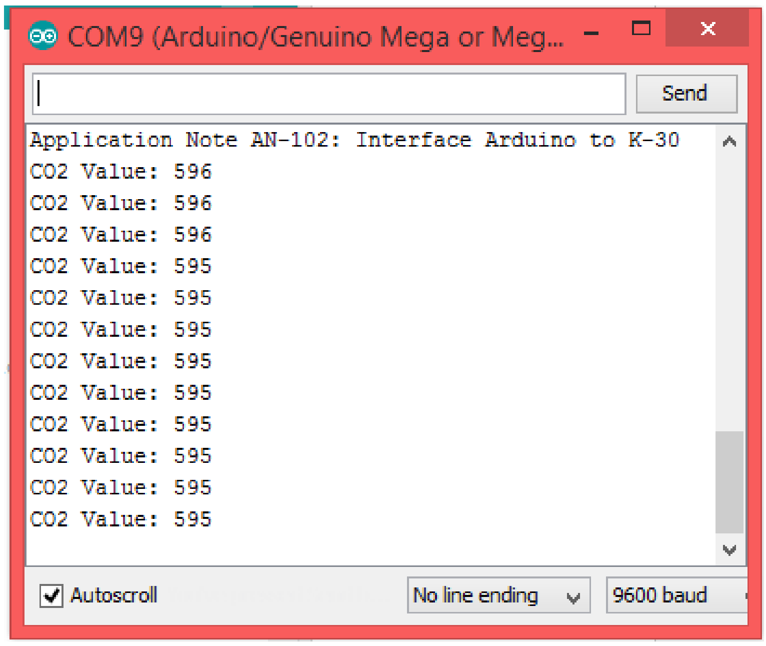
http://www.co2meters.com/Documentation/AppNotes/AN102-K30-Sensor-Arduino-I2C.zip

3. The Arduino GUI will start and ask the following;



The file "Arduino_to_I2C.ino" needs to be inside a sketch folder named "Arduino_to_I2C". Create this folder, move the file, and continue?

4. Click on OK. Observe the Arduino project code is displayed.
5. Click on Sketch >> Verify/Compile. The project should compile without errors.
6. Verify that your Arduino board is recognized correctly:
   a. Click on Tools. Set Board to Arduino UNO or MEGA.
   b. Confirm that Processor matches your Arduino: UNO or MEGA or MEGA 2560.
7. Click on Upload. When done uploading, your project is now running in the Arduino board.
8. To view program operation, click on Tools >> Serial Monitor. Observe the following:



```
Application Note AN-102: Interface Arduino to K-30
CO2 Value: 596
CO2 Value: 596
CO2 Value: 596
CO2 Value: 595
CO2 Value: 595
CO2 Value: 595
CO2 Value: 595
CO2 Value: 595
CO2 Value: 595
CO2 Value: 595
CO2 Value: 595
CO2 Value: 595
```

CO2Meter.com
CO2 Measurement Specialists

Appendix A:

The source code for the .ino and .txt files are below:

```
// CO2 Meter K-series Example Interface
// Revised by Marv Kausch, 7/2016 at CO2 Meter <co2meter.com>
// Talks via I2C to K30/K33 Sensors and displays CO2 values
#include <Wire.h>
// We will be using the I2C hardware interface on the Arduino in
// combination with the built-in Wire library to interface.
// Arduino analog input 5 - I2C SCL
// Arduino analog input 4 - I2C SDA
/*
  In this example we will do a basic read of the CO2 value and checksum
verification. For more advanced applications see the I2C Comm guide.
*/
int co2Addr = 0x68;
// This is the default address of the CO2 sensor, 7bits shifted left.
void setup() {
  Serial.begin(9600);
  Wire.begin ();
  pinMode(13, OUTPUT); // address of the Arduino LED indicator
  Serial.println("Application Note AN-102: Interface Arduino to K-30");
}
///////////////////////////////////////////////////////////////////
// Function : int readCO2()
// Returns : CO2 Value upon success, 0 upon checksum failure
// Assumes : - Wire library has been imported successfully.
// - LED is connected to IO pin 13
// - CO2 sensor address is defined in co2_addr
///////////////////////////////////////////////////////////////////
int readCO2()
{
  int co2_value = 0;  // Store the CO2 value inside this variable.

  digitalWrite(13, HIGH);  // turn on LED
  // On most Arduino platforms this pin is used as an indicator light.

  //////////////////////////
  /* Begin Write Sequence */
  //////////////////////////

  Wire.beginTransmission(co2Addr);
  Wire.write(0x22);
  Wire.write(0x00);
  Wire.write(0x08);
  Wire.write(0x2A);

  Wire.endTransmission();

  /////////////////////////
  /* End Write Sequence. */
  /////////////////////////

  /*
    Wait 10ms for the sensor to process our command. The sensors's
    primary duties are to accurately measure CO2 values. Waiting 10ms
    ensures the data is properly written to RAM

  */

  delay(10);

  /////////////////////////
  /* Begin Read Sequence */
  /////////////////////////

  /*
    Since we requested 2 bytes from the sensor we must read in 4 bytes.
    This includes the payload, checksum, and command status byte.

  */

  Wire.requestFrom(co2Addr, 4);

  byte i = 0;
  byte buffer[4] = {0, 0, 0, 0};

  /*
    Wire.available() is not necessary. Implementation is obscure but we
    leave it in here for portability and to future proof our code
  */

  while (Wire.available())
  {
    buffer[i] = Wire.read();
    i++;
  }

  ///////////////////////
  /* End Read Sequence */
  ///////////////////////
```

```
  /*
    Using some bitwise manipulation we will shift our buffer
    into an integer for general consumption
  */

  co2_value = 0;
  co2_value |= buffer[1] & 0xFF;
  co2_value = co2_value << 8;
  co2_value |= buffer[2] & 0xFF;

  byte sum = 0; //Checksum Byte
  sum = buffer[0] + buffer[1] + buffer[2]; //Byte addition utilizes overflow

  if (sum == buffer[3])
  {
    // Success!
    digitalWrite(13, LOW);
    return co2_value;
  }
  else
  {
    // Failure!
    /*
      Checksum failure can be due to a number of factors,
      fuzzy electrons, sensor busy, etc.
    */

    digitalWrite(13, LOW);
    return 0;
  }
}
void loop() {

  int co2Value = readCO2();
  if (co2Value > 0)
  {
    Serial.print("CO2 Value: ");
    Serial.println(co2Value);
  }
  else
  {
    Serial.println("Checksum failed / Communication failure");
  }
  delay(2000);
}
```