K-Series sensors
Arduino UART Example

Goal:
This is a guide to getting started using the K-series sensors with an Arduino/Atmel microcontroller. This example can be ported over to many MCU types, but the code and wiring is specifically for use with an Arduino. When the example is finished we will have a device that can get a co2 reading from a k-30 and report the value back to the computer over USB.

Previous Knowledge:
While it is possible to follow this guide with no experience, it would be helpful to be familiar with the following:

- Basic Arduino use
- The "SoftwareSeraial.h" Library for arduino
- K-Series UART communication

Library Dependencies:
This guide will give two examples for code. One will use the standard "SoftwareSerial.h" library that comes with the Arduino IDE, and the other will use a Library developed by Co2Meter.com ("kSeries.h"). To use this library just download the zip file and unzip it to the 'libraries' folder in your main Arduino directory (typically in MyDocuments) . If this is your first time using a 3rd party library you may have to create the 'libraries' folder first.

Once you have unzipped the files to your 'libraries' directory, restart the Arduino IDE. Now both code examples can be found under File >> Examples >> kSeries . When you wish to use the kSeries library In the future, just include it with the line:

```
#include "kSeries.h"
```

Using the kSeries.h Library: *(some functions not available on all sensors)*

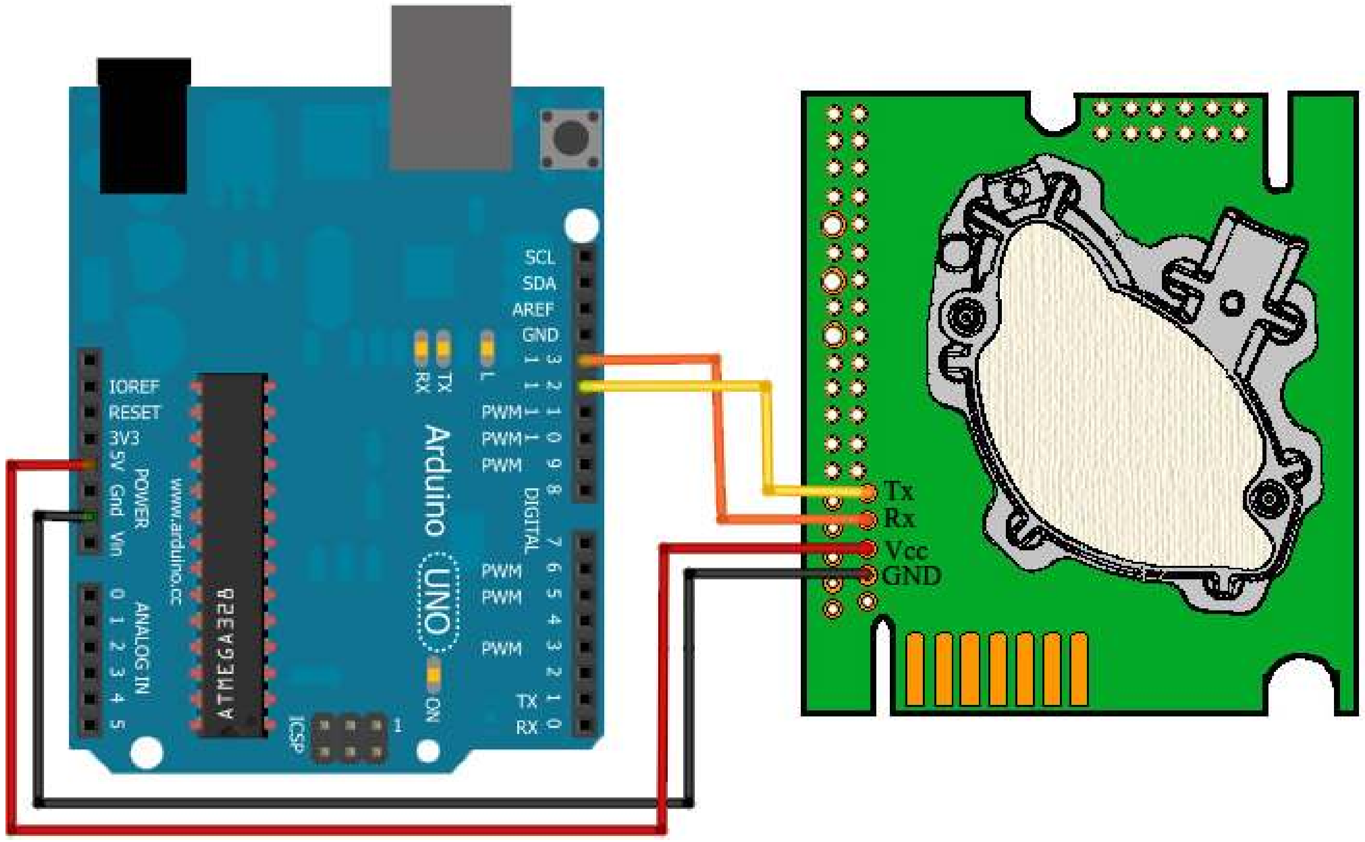
kSeries **kSeries(int Rx, int Tx)**
    Initializes a sensor on a virtual Serial port at Rx,Tx
double **getCO2(char format)**
    Returns a co2 reading in either ppm('p') or percent ('%') depending on the char passed to it
double **getTemp(char unit)**
    Returns a temperature reading in either farenheit('f') or Celsius ('c')
double **getRH()**
    Returns the humidity in RH

Wiring:

The first step is to wire up the sensor to your Arduino. This example shows the Arduino UNO with a K-30, but any of the Arduino series will work. Just make sure to select the proper board under Tools>>Board>> in your Arduino IDE. For use with another k-series model other than the K-30 look you must get the correct pins from the datasheet

1) Make the following Connections:

Wire	**Arduino**	**Sensor**
Red	5V	Vcc/G+ (do NOT connect Vcc for K-33)
Black	GND	Ground/G0
Yellow	Digital pin #12	UART TxD
Orange	Digital pin #13	UART RxD



Writing the Code:

The next step is to write the code to drive the device. First I will show a quick and simple sketch that will report the co2 value, and then I will show a sketch that does the same job but implements the kSeries Arduino library made by Co2Meter.com .

EXAMPLE #1

This example shows a sketch written from scratch using only the SoftwareSerial Library that is standard with the Arduino IDE.

```
/*
  Basic Arduino example for K-Series sensor
  Created by Jason Berger
  Co2meter.com
*/


#include "SoftwareSerial.h"

SoftwareSerial K_30_Serial(12,13);  //Sets up a virtual serial port
                                    //Using pin 12 for Rx and pin 13 for Tx


byte readCO2[] = {0xFE, 0X44, 0X00, 0X08, 0X02, 0X9F, 0X25};  //Command packet to read Co2 (see app note)
byte response[] = {0,0,0,0,0,0,0};  //create an array to store the response

//multiplier for value. default is 1. set to 3 for K-30 3% and 10 for K-33 ICB
int valMultiplier = 1;

void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600);         //Opens the main serial port to communicate with the computer
  K_30_Serial.begin(9600);    //Opens the virtual serial port with a baud of 9600
}

void loop()
{
  sendRequest(readCO2);
  unsigned long valCO2 = getValue(response);
  Serial.print("Co2 ppm = ");
  Serial.println(valCO2);
  delay(2000);

}
```

```
void sendRequest(byte packet[])
{
 while(!K_30_Serial.available())  //keep sending request until we start to get a response
 {
  K_30_Serial.write(readCO2,7);
  delay(50);
 }

 int timeout=0;  //set a timeoute counter
 while(K_30_Serial.available() < 7 )              //Wait to get a 7 byte response
 {
  timeout++;
  if(timeout > 10)    //if it takes to long there was probably an error
    {
      while(K_30_Serial.available())  //flush whatever we have
        K_30_Serial.read();

        break;                      //exit and try again
    }
    delay(50);
 }

 for (int i=0; i < 7; i++)
 {
  response[i] = K_30_Serial.read();
 }
}

unsigned long getValue(byte packet[])
{
   int high = packet[3];                      //high byte for value is 4th byte in packet in the packet
   int low = packet[4];                       //low byte for value is 5th byte in the packet


   unsigned long val = high*256 + low;              //Combine high byte and low byte with this formula to get value
   return val* valMultiplier;
}
```

END CODE EXAMPLE #1

EXAMPLE #2

This example shows a sketch with the same functionality as Example #1 , but simplified by implementing the kSeries.h library by CO2Meter.com.

```
/*
  Reports values from a K-series sensor back to the computer
  written by Jason Berger
  Co2Meter.com

*/

#include "kSeries.h"   //include kSeries Library

kSeries K_30(12,13);  //Initialize a kSeries Sensor with pin 12 as Rx and 13 as Tx

void setup()
{
  Serial.begin(9600); //start a serial port to communicate with the computer
  Serial.println("Serial Up!");

}

void loop()
{
  double co2 = K_30.getCO2('p'); //returns co2 value in ppm ('p') or percent ('%')

  Serial.print("Co2 ppm = ");
  Serial.println(co2);        //print value
  delay(1500);               //wait 1.5 seconds
}
```

END CODE EXAMPLE #2