```
// Based on Ethernet's WebClient Example
// Based on CO2 Meter K - series Example Interface
// Based on Andrew Robinson's code for CO2 Meter <co2meter.com>

// Talks via I2C to K33 - ELG/BLG Sensors for Host - Initiated Data Collection
// Submits collected data to myserver via HTTP POST

// Using the I2C hardware interface on the Arduino in
// combination with the built - in Wire library to interface.
//  Arduino analog input 5  -  I2C SCL  [el cable green]
//  Arduino analog input 4  -  I2C SDA  [el cable yellow]
//  basic read of the CO2 value and checksum verification.

#include <avr/wdt.h>
#include <Wire.h>
#include "WiFly.h"
#include "Credentials.h"

Client client("myserver.com", 80);        //Provide Server and Port to connect

String feedtemp="3m793";//"3m793";//"3m749";              //PROVIDE THE FEED
NAME like you would do on pachube.com
String feedco2="3m78h";//"3m78h";//"3m748";             //PROVIDE THE FEED
NAME pachube.com
String feedrh="3m794";//"3m794";//"3m74i";              //PROVIDE THE FEED
NAME

double actual = 107.43;
```

```
double tempValue = 0;

double rhValue = 0;

double co2Value = 0;


int co2Addr = 0x68;   // This is the default address of the CO2 sensor, 7bits shifted left.

int secondstowait = 0;

int cycletoreset = 0;


unsigned long time1=0;

unsigned long time2=0;

unsigned long reboot=10000;  //reboots arduino every REBOOT milli seconds (+8 seconds)

unsigned long start;

unsigned long duration;


void setup() {


  Serial.begin(9600);

  Wire.begin (); delay(300);

  WiFly.begin(); delay(300);                //START WIFLY

  WiFly.join(ssid, passphrase); delay(300);    //CONNECT TO WIFI USING NETWORK NAME AND PASSWORD FOUND IN CONFIG.H

//  wdt_enable(WDTO_8S);                    //Set up the watchdog timer. Si perro no pateado reset every 8 seconds

//  wdt_reset();                          //Good practice to reset the watchdog in setup

  pinMode(13, OUTPUT);                    // We will use this pin as a read - indicator

  Serial.println("A RESET HAS BEEN MADE.....QUE PASO GASPAR? THE SERIAL,WIRE, AND WIFLY ARE TURNED (IN THEORY) ON!");
```

```
  start=millis();
}


void loop() {


//while (millis()-start<40000){


wdt_enable(WDTO_8S);
//----------------  //READ DATA FROM SENSOR ----------------------------------------------
-
wakeSensor();
initPoll();
delaysinreset(16); //DELAY(16000);
wakeSensor();


double tempValue = readTemp();    delay(20); wakeSensor();


double rhValue = readRh();        delay(20);  wakeSensor();


double co2Value = readCo2();
```

```
//  if(co2Value >= 0) {                  //DISPLAY SENSOR VALUES TO THE SERIAL
PORT (PARA DEBUGEAR)
      Serial.print("CO2: ");
      Serial.print(co2Value);
      Serial.print("ppm Temp: ");
      Serial.print(tempValue);
      Serial.print("C Rh: ");
      Serial.print(rhValue);
      Serial.println("%");
//  }


// else {
//       Serial.println("Checksum failed / Communication failure");
//  }


//-----------------  //POST DATA TO my server ----------------------------------------------------
--
    wdt_reset();
    char dummy[32] = {0};           //BUFFER TO STORE THE CO2 READING IN AN
ARRAY OF CHARACTERS


    dtostrf(co2Value, 10, 2, dummy);  //converts doubles to chars : dtostrf(floatVar,
minStringWidthIncDecimalPoint, numVarsAfterDecimal, charBuf);
    Serial.println("Valor a mandar, Co2:");
    Serial.println(dummy);
    postear(feedco2,dummy); wdt_reset();


    dtostrf(tempValue, 10, 2, dummy);
    Serial.println("Valor a mandar, Temperatura:");
```

```
    Serial.println(dummy);
    postear(feedtemp,dummy); wdt_reset();

    dtostrf(rhValue, 10, 2, dummy);
    Serial.println("Valor a mandar,Rh:");
    Serial.println(dummy);
    postear(feedrh,dummy);  wdt_reset();

    //NOTE POSSIBLE ERRRORS:
    // posting 00.xxx or 0x.xx  results in http 500 error. Cant start with two zeros !
    // posting the co2 value from the sensor someimes we get HTTP/1.1 503 Service Unavailable even when the format is correct


//}//END WHILE

//Serial.println("RESEEEETT");
//  Serial.println(millis()-start);
//  wdt_enable(WDTO_2S);
//  delay(3000);



}

////////////////////////////////////////////////////////////////
```

```
// Function : void delaysinreset()

// Executes : Uses delay() and wdt_reset() to wait secondstowait without allowing the microprocessor to reset

// Note     : If we use delay() alone for more than 8 seconds, the watchdog resets the micro

//////////////////////////////////////////////////////////////

void delaysinreset(int secondstowait){

    for (int i=0; i <= secondstowait; i++){
       wdt_reset();
       delay(1000);
       Serial.print(i);
    }
}

//////////////////////////////////////////////////////////////

// Function : void mydelay()

// Executes : Uses millis to delay

// Note     : When we were using delay() the code was giving trouble

//////////////////////////////////////////////////////////////
```

```
void mydelay(unsigned long duracion)
{

  if (millis() - start > duracion)
  {
    Serial.print("me voy a dormir");
    delay(9000);
  }
}
```

```
//////////////////////////////////////////////////////////////////

// Function : void wakeSensor()

// Executes : Sends wakeup commands to K33 sensors.

// Note     : THIS COMMAND MUST BE MODIFIED FOR THE SPECIFIC AVR YOU ARE USING

//          THE REGISTERS ARE HARD‐CODED

//////////////////////////////////////////////////////////////////
```

```
void wakeSensor() {

  // This command serves as a wakeup to the CO2 sensor, for K33 - ELG/BLG Sensors Only

  // You'll have the look up the registers for your specific device, but the idea here is simple:

  // 1. Disabled the I2C engine on the AVR

  // 2. Set the Data Direction register to output on the SDA line

  // 3. Toggle the line low for ~1ms to wake the micro up. Enable I2C Engine

  // 4. Wake a millisecond.


  TWCR &= ~(1<<2); // Disable I2C Engine

  DDRC |= (1<<4); // Set pin to output mode

  PORTC &= ~(1<<4); // Pull pin low

  delay(1);

  PORTC |= (1<<4); // Pull pin high again

  TWCR |= (1<<2); // I2C is now enabled

  delay(1);


}


////////////////////////////////////////////////////////////////


// Function : void initPoll()


// Executes : Tells sensor to take a measurement.


// Notes    : A fuller implementation would read the register back and
```

```
//          ensure the flag was set, but in our case we ensure the poll

//          period is >25s and life is generally good.

////////////////////////////////////////////////////////////////

void initPoll() {

  Wire.beginTransmission(co2Addr);
Wire.send(0x11);
Wire.send(0x00);
Wire.send(0x60);
Wire.send(0x35);
Wire.send(0xA6);
  Wire.endTransmission();

delay(20);

Wire.requestFrom(co2Addr, 2);

  byte i = 0;
  byte buffer[2] = {0, 0};

  while(Wire.available()) {
     buffer[i] = Wire.receive();
     i++;
```

```
 }

}

//////////////////////////////////////////////////////////////////

// Function : double readCo2()

// Returns  : The current CO2 Value, -1 if error has occured

//////////////////////////////////////////////////////////////////

double readCo2() {

 int co2_value = 0;

 // We will store the CO2 value inside this variable.

 digitalWrite(13, HIGH);

 // On most Arduino platforms this pin is used as an indicator light.

//////////////////////////

/* Begin Write Sequence */

 //////////////////////////
```

```
 Wire.beginTransmission(co2Addr);
Wire.send(0x22);
Wire.send(0x00);
Wire.send(0x08);
Wire.send(0x2A);
 Wire.endTransmission();

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

delay(20);

//////////////////////

/* 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 nessessary. Implementation is obscure but we leave

    it in here for portability and to future proof our code

  */

while(Wire.available()) {

     buffer[i] = Wire.receive();

     i++;

  }

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 ((double) co2_value / (double) 1);
 }

else {
  // Failure!
  /*   Checksum failure can be due to a number of factors,
       fuzzy electrons, sensor busy, etc.
 */
    digitalWrite(13, LOW);
    //return (double)  - 1;
 }

}

//////////////////////////////////////////////////////////////////

// Function : double readTemp()

// Returns  : The current Temperture Value,  - 1 if error has occured
```

```
////////////////////////////////////////////////////////////////

double readTemp() {

  int tempVal = 0;
  digitalWrite(13, HIGH);

  Wire.beginTransmission(co2Addr);
Wire.send(0x22);
Wire.send(0x00);
Wire.send(0x12);
Wire.send(0x34);
  Wire.endTransmission();

delay(20);

Wire.requestFrom(co2Addr, 4);

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

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

```
tempVal = 0;

 tempVal |= buffer[1] & 0xFF;
 tempVal = tempVal << 8;
 tempVal |= buffer[2] & 0xFF;
 byte sum = 0;                    //Checksum Byte

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

 if(sum == buffer[3]) {
    digitalWrite(13, LOW);
    return ((double) tempVal / (double) 100);
 }
 else {
    digitalWrite(13, LOW);
   // return  - 1;
 }
}

//////////////////////////////////////////////////////////////////

// Function : double readRh()

// Returns  : The current Rh Value,  - 1 if error has occured
```

```
////////////////////////////////////////////////////////////////

double readRh() {

  int tempVal = 0;
  digitalWrite(13, HIGH);

 Wire.beginTransmission(co2Addr);
Wire.send(0x22);
Wire.send(0x00);
Wire.send(0x14);
Wire.send(0x36);
Wire.endTransmission();

delay(20);
Wire.requestFrom(co2Addr, 4);

 byte i = 0;
 byte buffer[4] = {0, 0, 0, 0};
 while(Wire.available()) {
    buffer[i] = Wire.receive();
    i++;
 }

tempVal = 0;

 tempVal |= buffer[1] & 0xFF;
```

```
  tempVal = tempVal << 8;
  tempVal |= buffer[2] & 0xFF;

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

  if(sum == buffer[3]) {
     digitalWrite(13, LOW);
     return (double) tempVal / (double) 100;
  }
  else {
     digitalWrite(13, LOW);
    // return  -1;
  }
}

//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
//
//FUNCTION "postear" POSTS TO server
//
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
void postear(String feed,String datapoint){
   Serial.println("El valor de datapoint");
   Serial.println(datapoint);
```

```
  client.connect();
  delay(300);       //WAIT FOR CONNECTION TO BE ESTABLISHED



  String s1 = "POST /networks/40/devices/";
  String s2 = "/data_items.json HTTP/1.1";
  String address = s1 + feed + s2;         //SEND "POST
/networks/40/devices/3m5bf/data_items.json HTTP/1.1"
  client.println(address); delay(50);
  client.println("Host: myserver.com"); delay(50);
  client.println("Accept: * / *"); delay(50);
  client.println("Content-Type: application/json"); delay(50);


  String j1 = "{\"user_key\":\"dha0\",\"data_item\":{\"body\":";
  String j2 = ",\"sent_at\":\"2012-01-29T12:16:27-08:00\"}}";
  j1.concat(datapoint);
  j1.concat(j2);


  String k1 = "Content-Length: ";
  String largo = j1.length();
  k1.concat(largo);


  client.println(k1);    delay(50);              //SEND "Content-Length:82";
  client.println();     delay(50);          //UNA ESPECIE DE ENF OF LINE
  client.println(j1);    delay(50);             //send
"{\"user_key\":\"dha0\",\"data_item\":{\"body\":459,\"sent_at\":\"2012-01-
29T12:16:27-08:00\"}}"
  client.println();     delay(50);          //UNA ESPECIE DE ENF OF LINE
```

```
  delay(500);


  Serial.println("Lo que se mando:----------");
  Serial.println(j1);
  Serial.println(k1);
  Serial.println("----------");


  //Serial.println("myserver's response to the POST is:");
 while (client.available()) {          //READS servers'S RESPONSE:WE HOPE FOR AN "HTTP/1.1 200 OK"
  char c = client.read();
  Serial.print(c);
  delay(5);                    //GIVE THE PORT SOME TIME OR IT GETS STUCK
 }


  //Serial.flush();
  client.stop();    //CLOSING THE CONECTION INCREASED THE FUNCTION'S RELIABILITY
  client.flush();


  //Serial.println("Desconectando del servidor usando: client.stop()");
  //wdt_disable(); //turn off watchdog timer - if sketch gets this far it hasn't hung
  //datapoint="";
}
//END OF postear()
//////////////////////////////////////////////////////
//////////////////////////////////////////////////////
```

```
//--------------------CODE BACKUP
//time1 = millis();
//time2 = millis();
//  Serial.print("leer y desplegar: ");
//  Serial.println((time2-time1));
```