

VISA R/W

NI-VISA Read/Write Challenge 6

1. INTRODUCTION

One of the many applications that LabView has is an I/O software known as VISA (Virtual Instrument System Architecture). A standard tool to create interfaces that allow us to control (configure, program, debug) instruments/equipment or acquire data from external sensors. All the above tasks are done through any communication port like Ethernet, GPIB, USB, and more.

2. GOAL

Follow the directions to complete all exercises using the advantages of VISA to control and communicate with an external device (ARDUINO), measure ambient CO₂ using a CCS811 sensor, and turn on/off LEDs to indicate the state of measures (over, low, or on the range).

After completing this activity, you will be able to connect, acquire, and send data through the USB serial port with LabView and show the behavior graph of the CO₂ values on the user interface.

3. MATERIAL

- LabView software
- Arduino UNO + USB Cable
- Protoboard
- Jumper Kit
- Resistors (120W)
- LED's
- CSS811 sensor

4. METHODOLOGY

Use the knowledge acquired in the previous sessions and self-study activities:

- Creating an interface in the front panel, variables, data processing, error handling.

Previous and Autonomous tasks:

- Exercises 6.1-6.5 from NI-VISA Read/Write and Serial Port

5. CHALLENGE

VISA: External data acquisition

In this challenge you will learn to get data from an external CCS811 sensor and show it on the user interface.

✓ Challenge: connecting the sensor

The aim is to get the CO₂ concentration values by the serial port (USB) and show the data on the screen.

- 1.- Open a new Arduino Sketch: a) type the following code (*figure 1*), b) verify if there are any errors, and c) upload the code to the Arduino board.

CODE:

<https://learn.sparkfun.com/tutorials/ccs811-air-quality-breakout-hookup-guide/all#example-basic-reading>

NOTE: You can also find the Arduino Sketch on the UB Campus Virtual, as well as the sensor hookup (“How to...” section).

```
#include <Wire.h>
#include "SparkFunCCS811.h" //Click here to get the library: http://librarymanager/All#SparkFun\_CCS811
#define CCS811_ADDR 0x5B //Default I2C Address
int variable;
CCS811 mySensor(CCS811_ADDR);

void setup()
{
  Serial.begin(115200); //set-up the baud rate

  pinMode(13, OUTPUT); // pIN_13 LED to know serial activation
  Serial.println("CCS811 Basic Example");

  Wire.begin(); //Initialize I2C Hardware. It's recommended to check run status on .begin()

  if (mySensor.begin() == false)
  {
    Serial.println("CCS811 error. Please check wiring. Freezing...");
    while (1);
  }
}

void loop()
{
  //Check to see if data is ready with .dataAvailable()
  if (mySensor.dataAvailable())
  {
    //If so... sensor read and calculate the results.
    //Get them later
    mySensor.readAlgorithmResults();

    Serial.print(mySensor.getCO2()); //Returns calculated CO2 reading
    Serial.print("\n"); //send "\n" // note: serial println or send "\n" (wait for changes to send the
  }
  delay(15); //Don't spam the I2C bus
}
```

2.- Verify that the data acquisition through the serial port is done correctly. To check it, blow close to the sensor and make sure that the CO2 acquisition value varies. Once verified, close the app (figure 2).

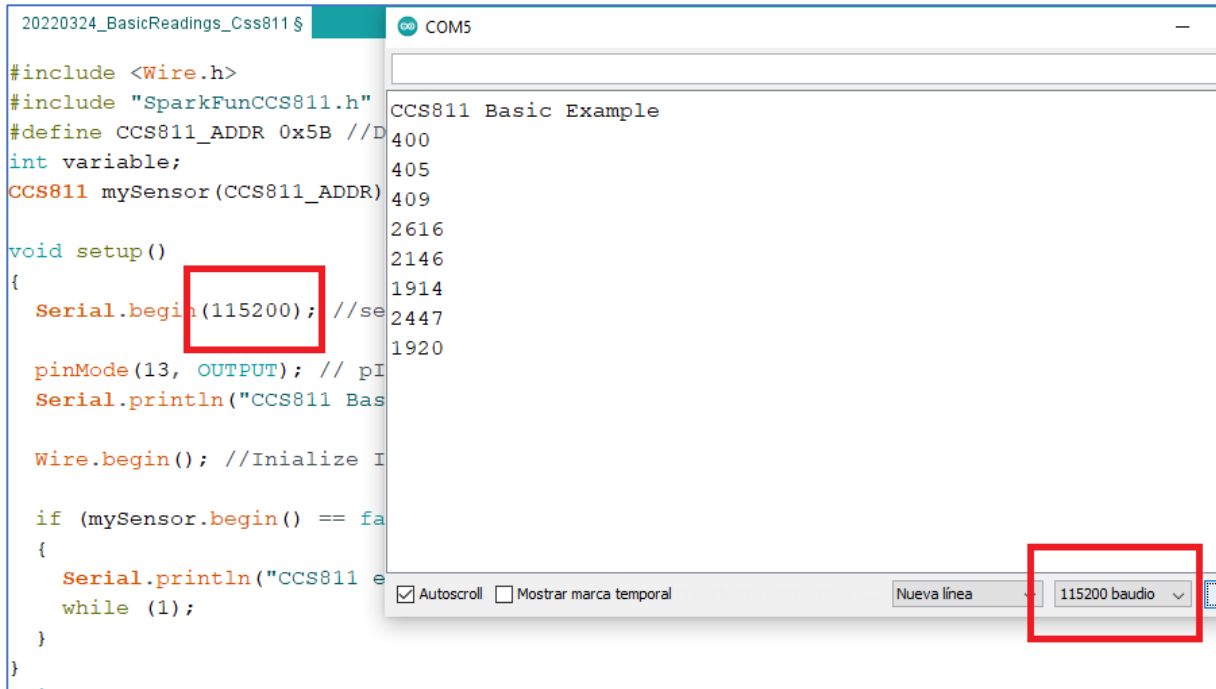


Figure 2. Arduino Sketch: Check the data transfer.

3.- On you Control Panel: Create a simple user interface as follows. This ensures that LabView is receiving data properly (Figure 3).

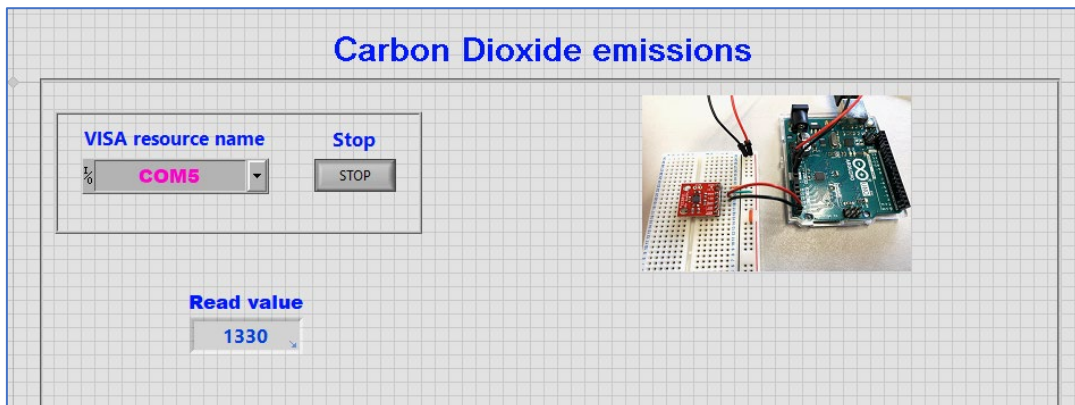


Figure 3. Arduino Sketch: Check the data transfer.

4.- On the Block Diagram: (Figure 4)

- 4.1 Configure de Serial Port indicating the resource name and the baud rate.
- 4.2 Use the VISA Read function to know the information coming from the Serial.
- 4.3 Close VISA.

Run the program to verify that the information transfer is correct.

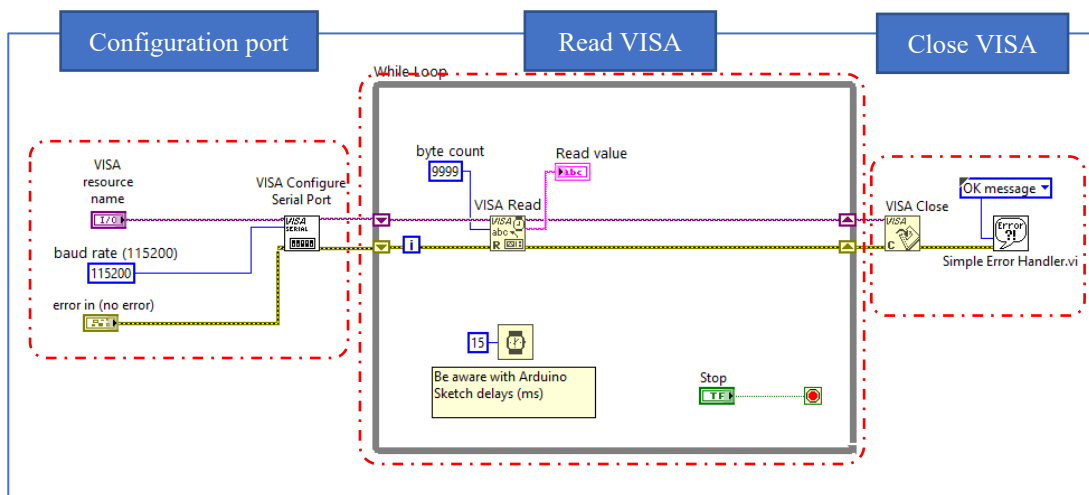


Figure 4. Block diagram view

✓ **Challenge: Process and display the input data on the user interface**

5.- Let's start by changing the *Configuration port* section (figure 5) to make sure that the incoming data is received line by line.

- Add the *Property Node* function and choose *Termination Character* and *Termination Character Enable* options to identify the character that will do the line brake action.
- Use the "Type Cast" function so the *Property Node* recognize Backslash character.

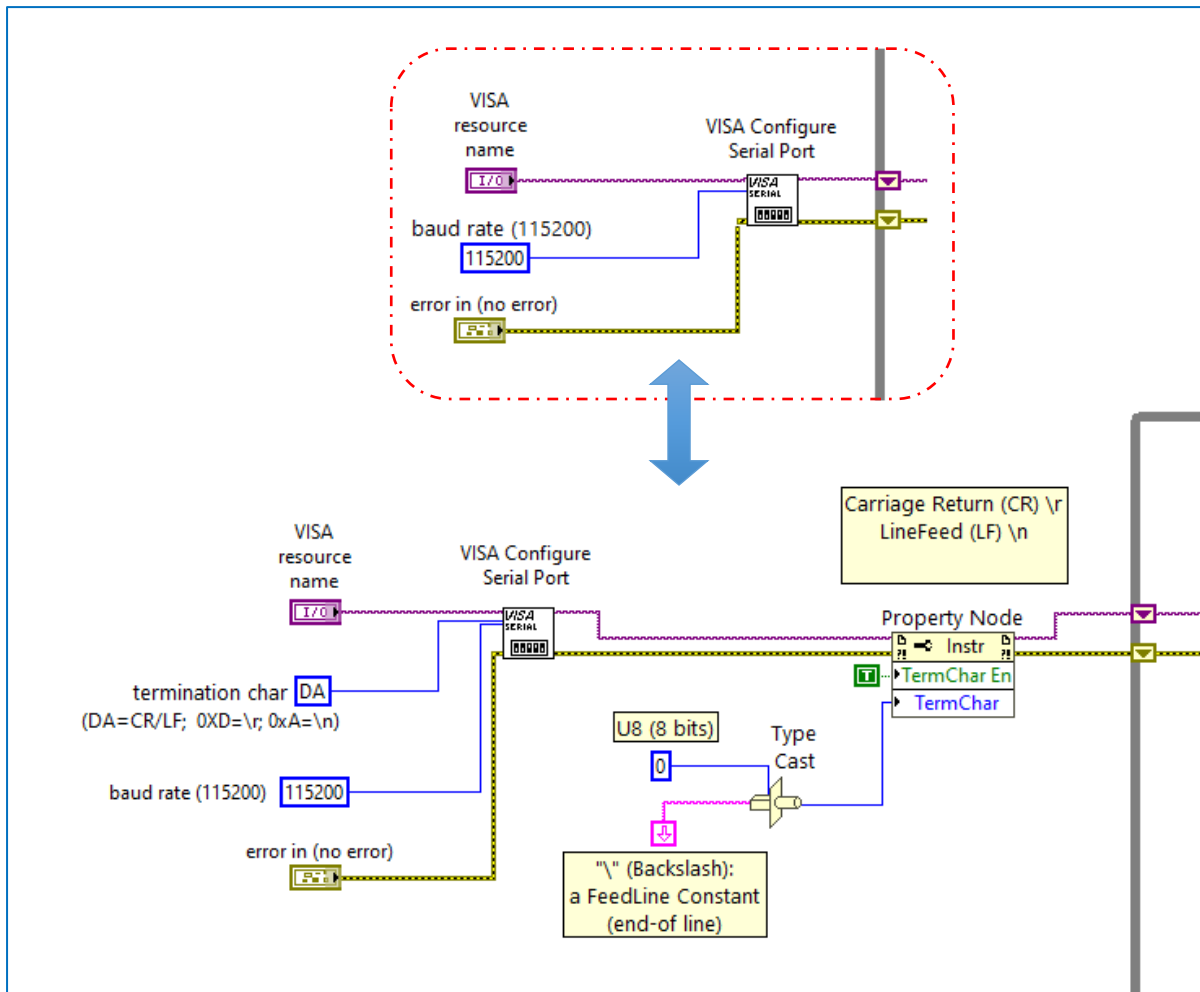


Figure 5. Block Diagram view: Configuration port section

6.- Now, the information within the While Loop is changed to compare the input value with a normal measurement range. In this way, three LED indicators will switch their status on/off indicating it the data is: "On Range", "Under Range", "Over Range".

On the front panel

6.1.- Use the *Number Control* function to create **two inputs** that the user can change. These represent the **upper and lower limits** of the normal operating range.

6.2.- Add three **LED** indicators to show the Status of the measurements.

6.3.- Add an **ENUM** to show the Acquisition Status (Calibrate, Measurement).

6.4.- Include a Waveform Chart to show the input values as they enter through the Serial Port as shown in *Figure 6*.

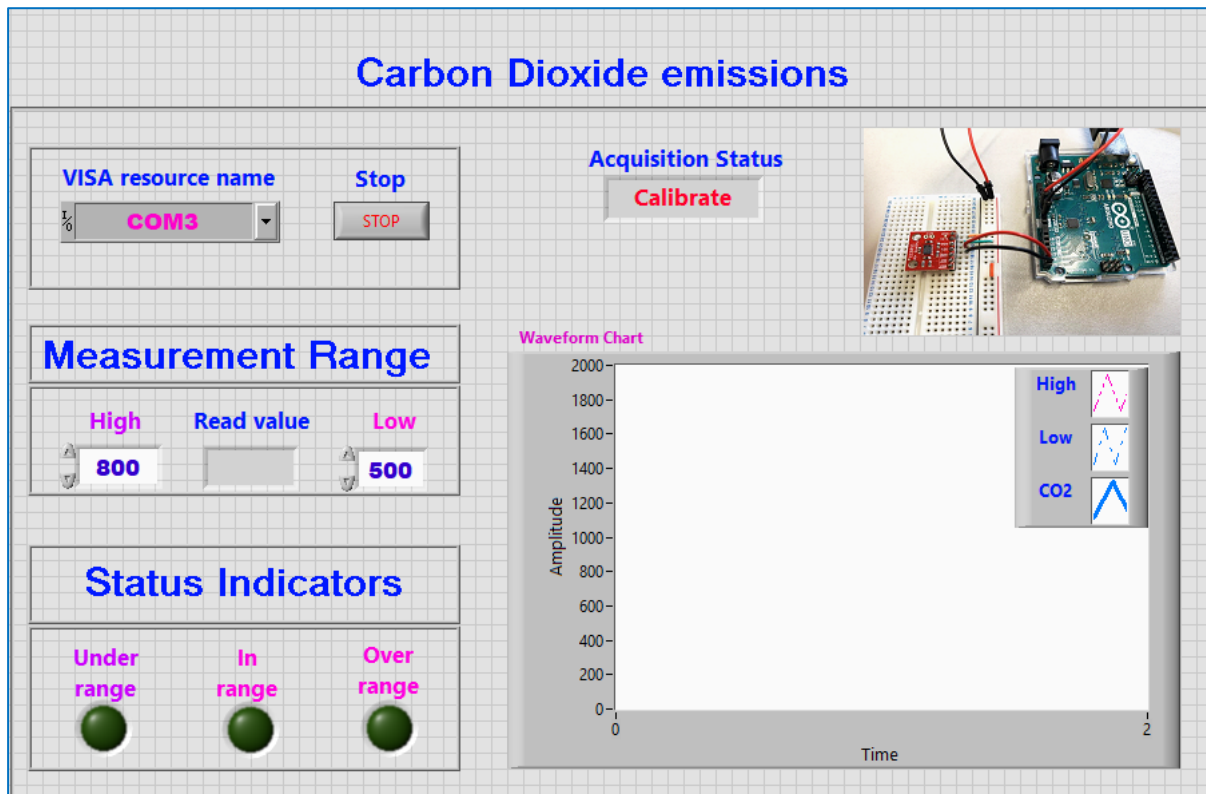


Figure 6. Front Panel View

On the Block Diagram

6.5.- Since the data coming from the Serial Port arrives as a String, use the format change function, *Frac/Exp String to Number*, for performing numerical operations.

6.6.- Moreover, use the *In Range and Coerce* function to determines whether the value incoming from the Serial Port remains within the range (between Low and High value). The response goes into the Case Structure (*Figure 6*).

6.7.- Case 1.- TRUE, the input value is within the range.

-Select the char to send (Visa Write) to turn on a Led on the protoboard.

-Choose the corresponding Boolean value for the virtual LEDs (In, Over, Under the range)

-Choose the state of the Acquisition Status (Enum).

a) Measure: When input is on or over the range

b) Calibrate: When input is under the range.

-Chose a plot color for the input using the corresponding Property Nodes as shows in *Figure 7*.

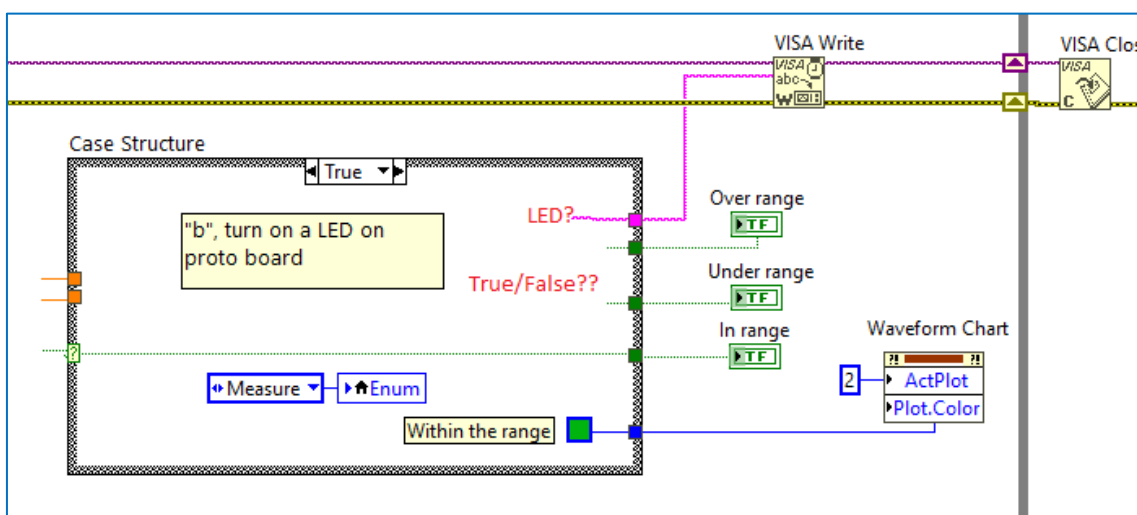
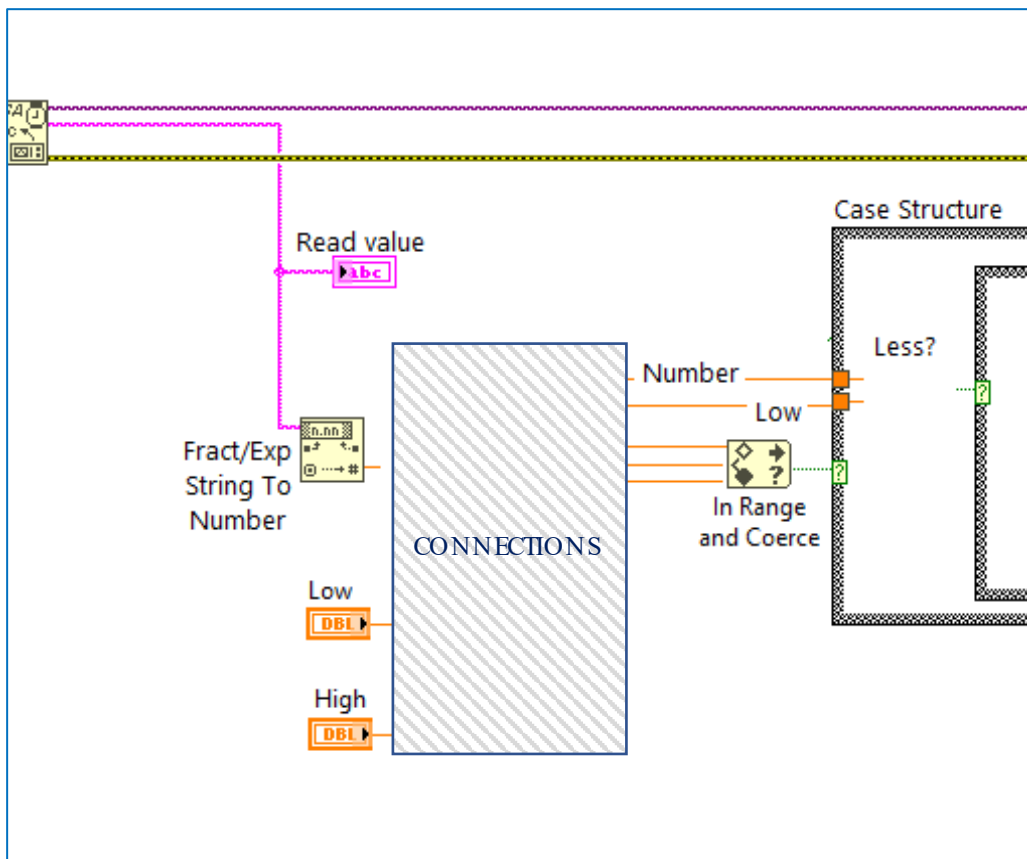


Figure 7. Block diagram view: Connections within the Case Structure (true)

6.8.- For the following cases

Case 2.- FALSE/FALSE, the sensing data is out of range and is over the range.

Case 3.- FALSE/TRUE, the sensing value is out of range and is under the range.

- Select the char to send (Visa Write) to turn on a Led on the protoboard.
- Choose the corresponding Boolean value for the virtual LEDs (In, Over, Under the range)
- Choose the state of the Acquisition Status (Enum).
- Chose a plot color for the input using the corresponding Property Nodes as shows in *Figure 8*.

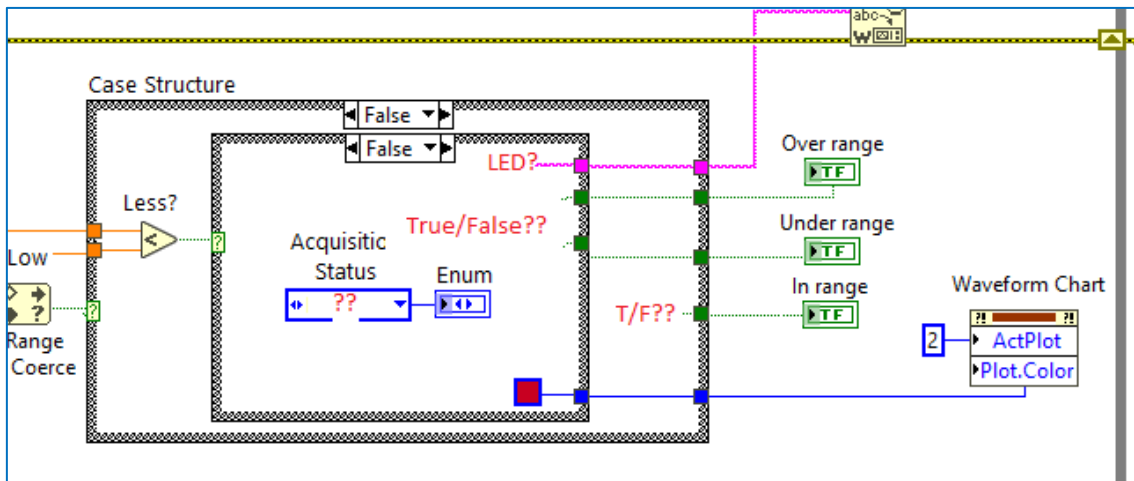


Figure 8. Block diagram view: Connections within the Case Structure (false/false)

6.9.- Connect the range values as well as the data from the sensor to a Bundle and show it on the Waveform Chart (*Figure 9*).

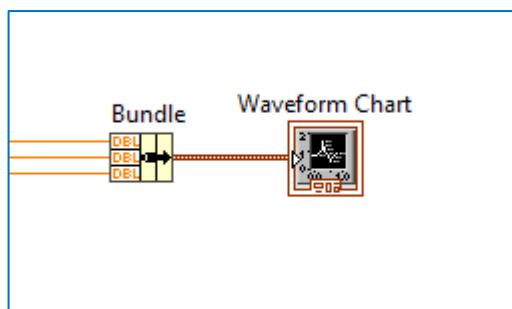


Figure 9. Block diagram view: Waveform Chart connection

6.10.- Make the proper connections as in *Figure 10* to:

- a) Specify a timer (wait)
- b) Connect the stop button

c) Create a Property Node from the Waveform Chart and choose the History Data. Clean history data from the Waveform Chart.

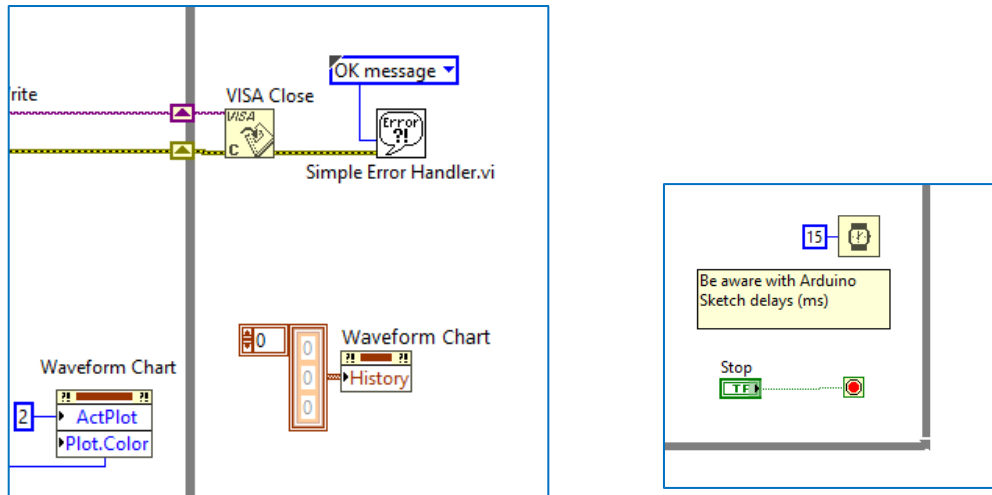


Figure 10. Block diagram. Delete history data, send error, add timer.

Run the program and check data acquisition, LEDs on/off functions, Acquisitions status changes, and plot colors variations according to the input data values (Figure 11).

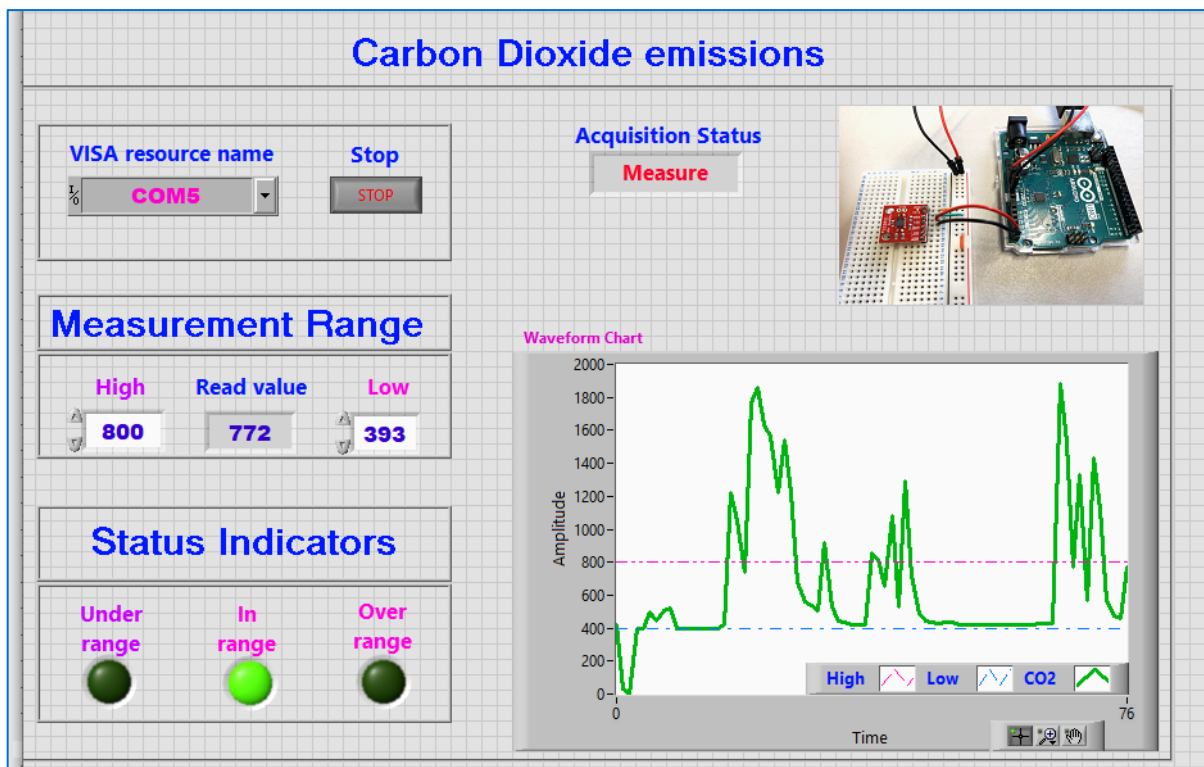
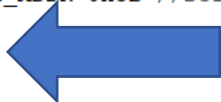


Figure 11. Testing program: User interface

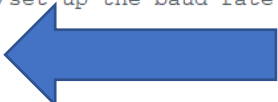
TIPS:

Remember to modify the Arduino Sketch to turn the LEDs on/off on the protoboard.

1.- Initialize the variables of the three LEDs. See the example below for one of them.

```
*****  
  
#include <Wire.h>  
#include "SparkFunCCS811.h" //Click here to get the library  
#define CCS811_ADDR 0x5B //Default I2C Address  
  
int LED9 = 9;   
  
int variable;  
CCS811 mySensor(CCS811_ADDR);  
  
void setup()  
{  
  Serial.begin(115200); //set-up the baud rate
```

2.- Declare variables as outputs on the Void Setup () section.

```
int variable;  
CCS811 mySensor(CCS811_ADDR);  
  
void setup()  
{  
  Serial.begin(115200); //set-up the baud rate  
  
  pinMode(LED9, OUTPUT);   
  
  pinMode(13, OUTPUT); // pIN_13 LED to know serial activation  
  Serial.println("CCS811 Basic Example");
```

3.- Specify the action that will be carried out repeatedly on Void loop () section.

```
}  
void loop()  
{  
  //Check to see if data is ready with .dataAvailable()  
  if (mySensor.dataAvailable())  
  {  
    //If so... sensor read and calculate the results.  
    //Get them later  
    mySensor.readAlgorithmResults();  
  
    Serial.print(mySensor.getCO2()); //Returns calculated CO2 reading  
    Serial.print("\n"); // Send "/n" (wait for changes to send the value = equivalent)  
  }  
  delay(15); //Don't spam the I2C bus  
  
  if (Serial.available()){  
    variable=Serial.read();  
  
    if (variable=='b')
```