

ARDUINO: THE VISUINO PROJECT - PART 2 BY BOIAN MITOV COLLECT DATA AND WORK WITH IT IN DELPHI



In the previous issue you learned how easy it is to program Arduino with Visuino and how to receive data from Arduino and visualize it. Acquiring, and visualizing this data is great, but it is not worth much, if we can't use it in our Delphi applications.

So let's fire up Delphi, and see how we can get that data into our Delphi applications. As I mentioned in part one, to develop Visuino, I needed a serial communication component. To solve the problem, I could have used one of the existing components such as AsynchronPro or any other. I actually spent some time reviewing and testing them, but I got disappointed fairly quickly. Most are designed so the data processing is done in the main thread.

This is an advantage, when they are used for smaller amounts of data, or speeds, by less experienced developers. But in case of large data exchange - if the main thread gets busy rendering graphical data or any other time consuming GUI related tasks - the communication will be blocked with a number of potential issues arising ranging from processing delay, memory over-usage, all the way to even potential data loss.

This led me to develop a new ComPort Serial component, and as with most components I develop, I made it OpenWire enabled, so it can directly connect to other OpenWire components.

The component worked well, and I wrote the necessary code to receive data from it, and properly populate a TMemo simulating a serial terminal.

While doing this and debugging the communication, I often needed to create test Delphi applications. I discovered that, time and again, I need to write small portions of data collecting and converting code to populate TMemos, so this led to the creation of dedicated terminal component, thus speeding up my development and debugging significantly.

In the last article I also demonstrated the ability to collect data from multiple sensors simultaneously, and send it to Visuino for visualization.

To achieve this, I needed to develop Data Packaging Arduino component, and its Delphi unpackaging counterpart. It was only natural to create the corresponding Package component in Delphi and Un-Package component in Arduino. This now allowed bidirectional packaged multichannel communication.

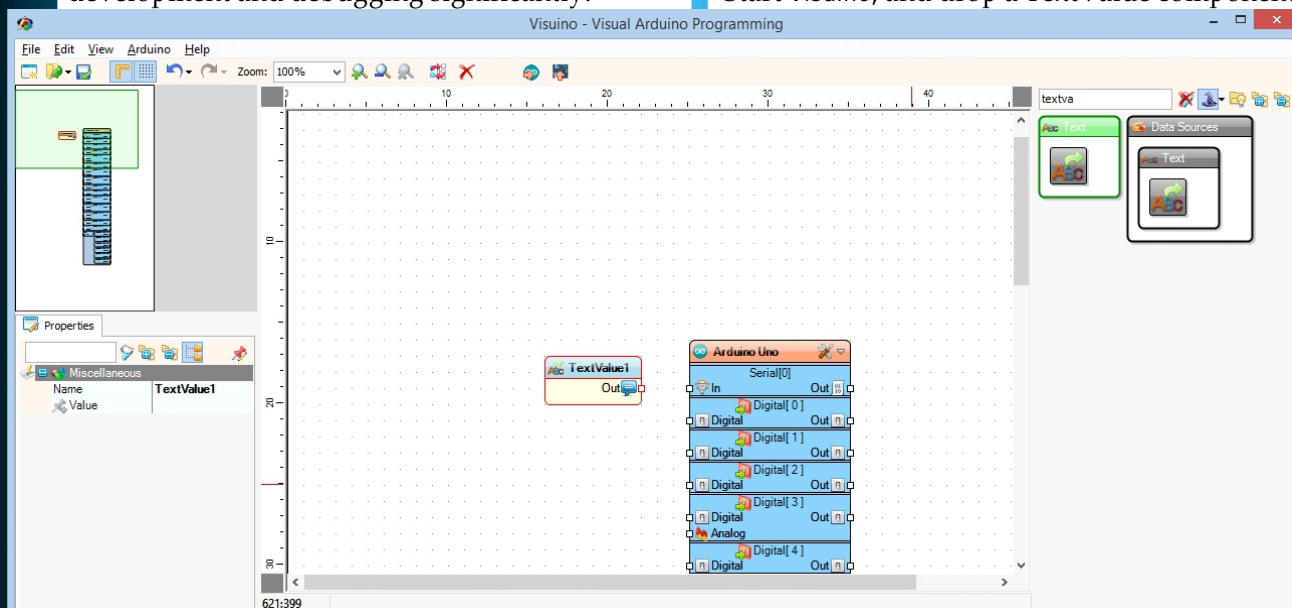
The ComPort, the Terminal, the Package and the UnPackage components formed the basis of a new upcoming library CommunicationLab. At present we are working to expand it with more components, and we are planning a Beta release soon. Here we will use the new components to demonstrate how easy it is to communicate with Arduino.

It is an old tradition to start with a "Hello World!" program. We will however start with a "Hello Delphi!" program instead.

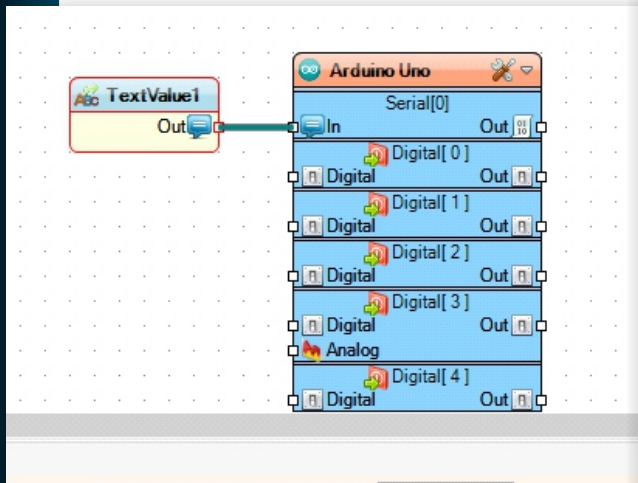
Before you start, make sure you have installed the Arduino IDE from <http://www.arduino.cc>, and Visuino from <http://www.visuino.com> as shown in the previous issue.

Visuino will automatically generate all the necessary Arduino C++ code, and the free open source Arduino IDE comes with all the necessary compiler and libraries. You do not need to know or understand any C++ code in order to program Arduino. The Visuino will handle all of that under the hood, as demonstrated in the previous issue.

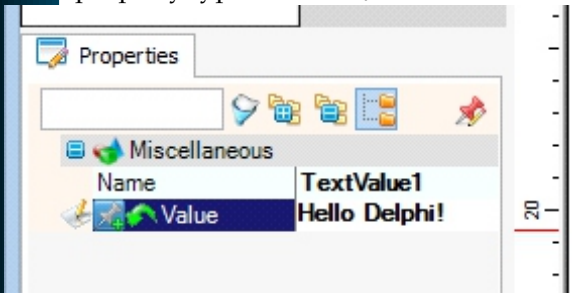
Start Visuino, and drop a TextValue component:



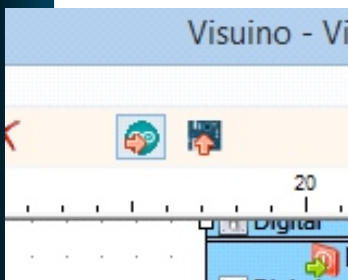
Next connect the OutputPin of the TextValue component to the InputPin of the Arduino Serial



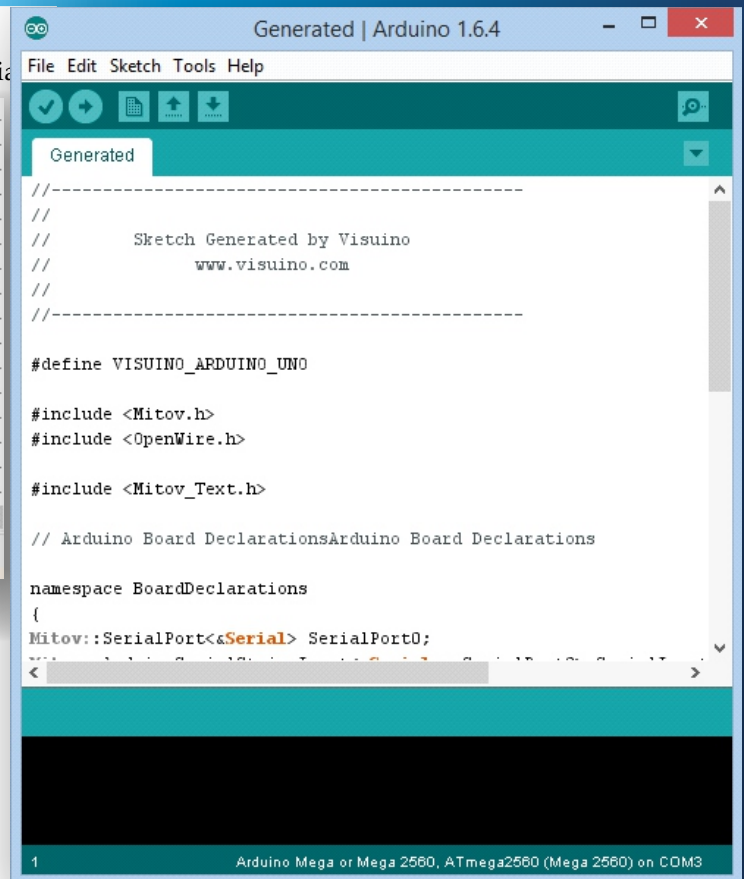
In the Object Inspector, for the "Value" property type "Hello Delphi!":




Click on the "Send to Arduino IDE for Compilation" button (or pressing F9):



Visuino will automatically generate the Arduino C++ code, and will open it in the Arduino IDE, where you can compile and upload it:

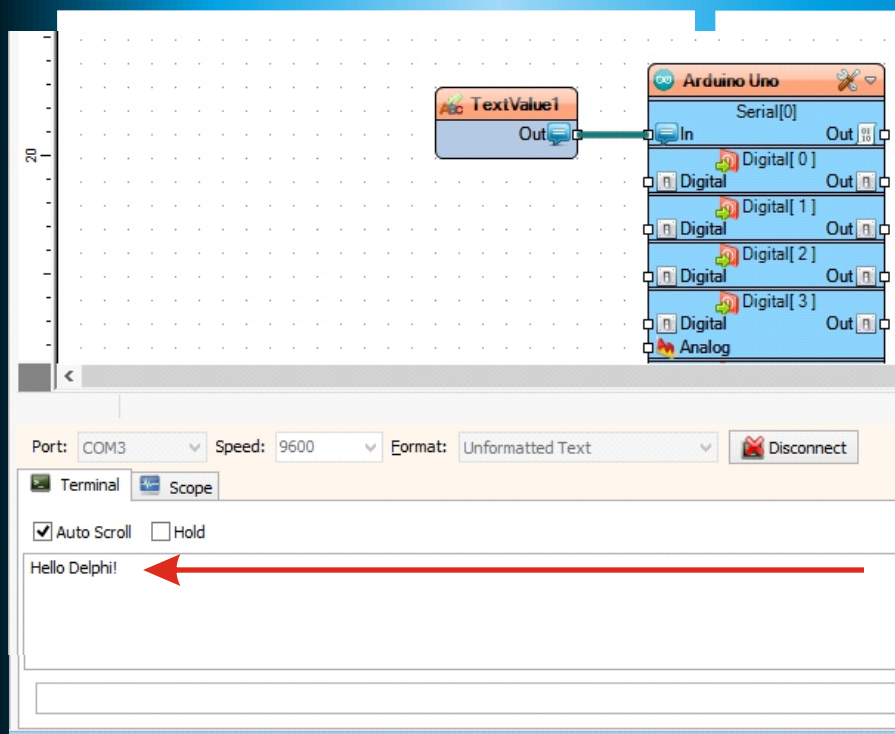


If you have your Arduino board connected, all you need is to click on the  button, and the code will be compiled and uploaded.

If you connect to Arduino with Visuino after the upload, by clicking on the "Connect" button:

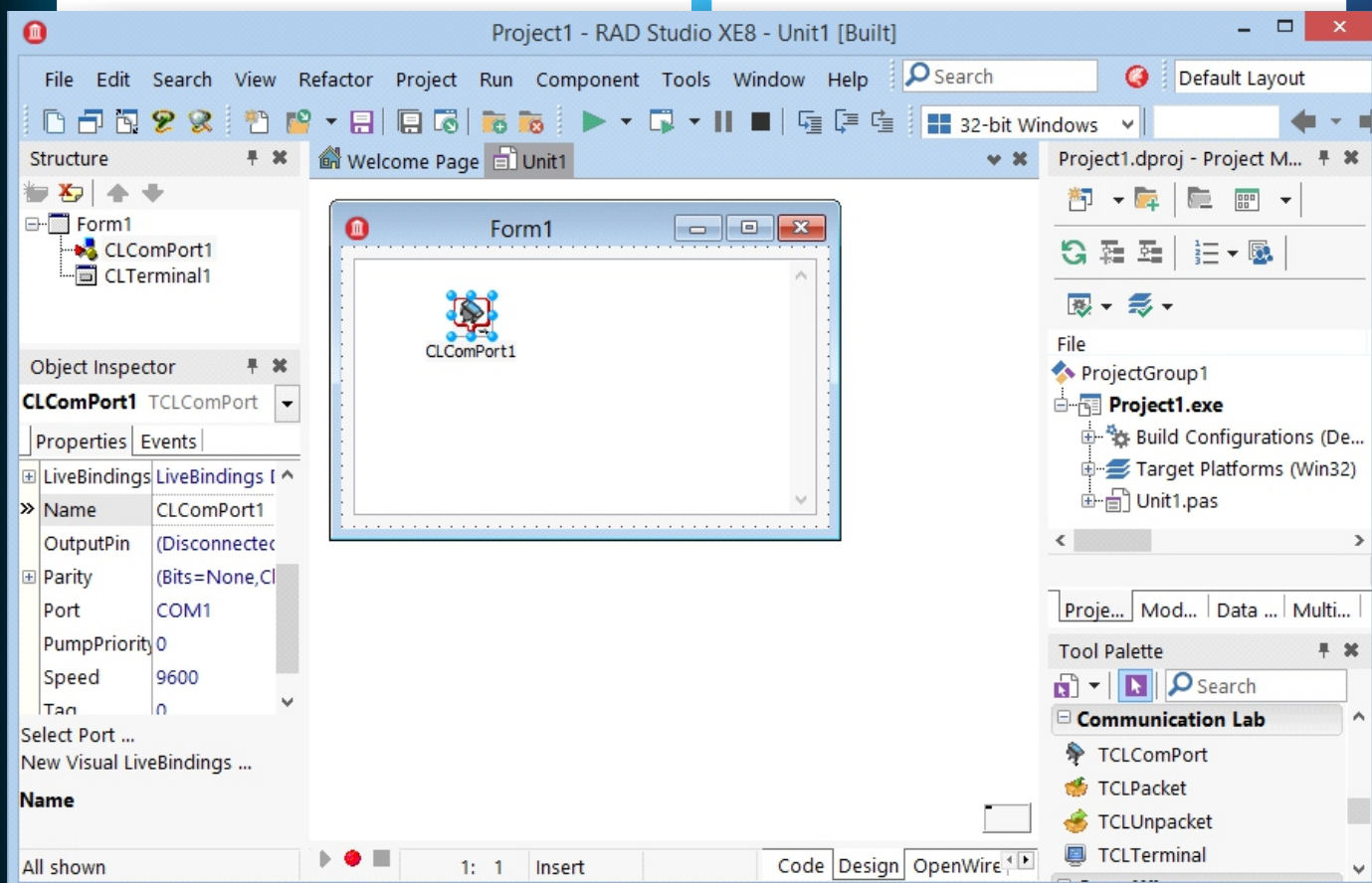


and press the reset button on the Arduino board, you will receive "Hello Delphi!" in the terminal

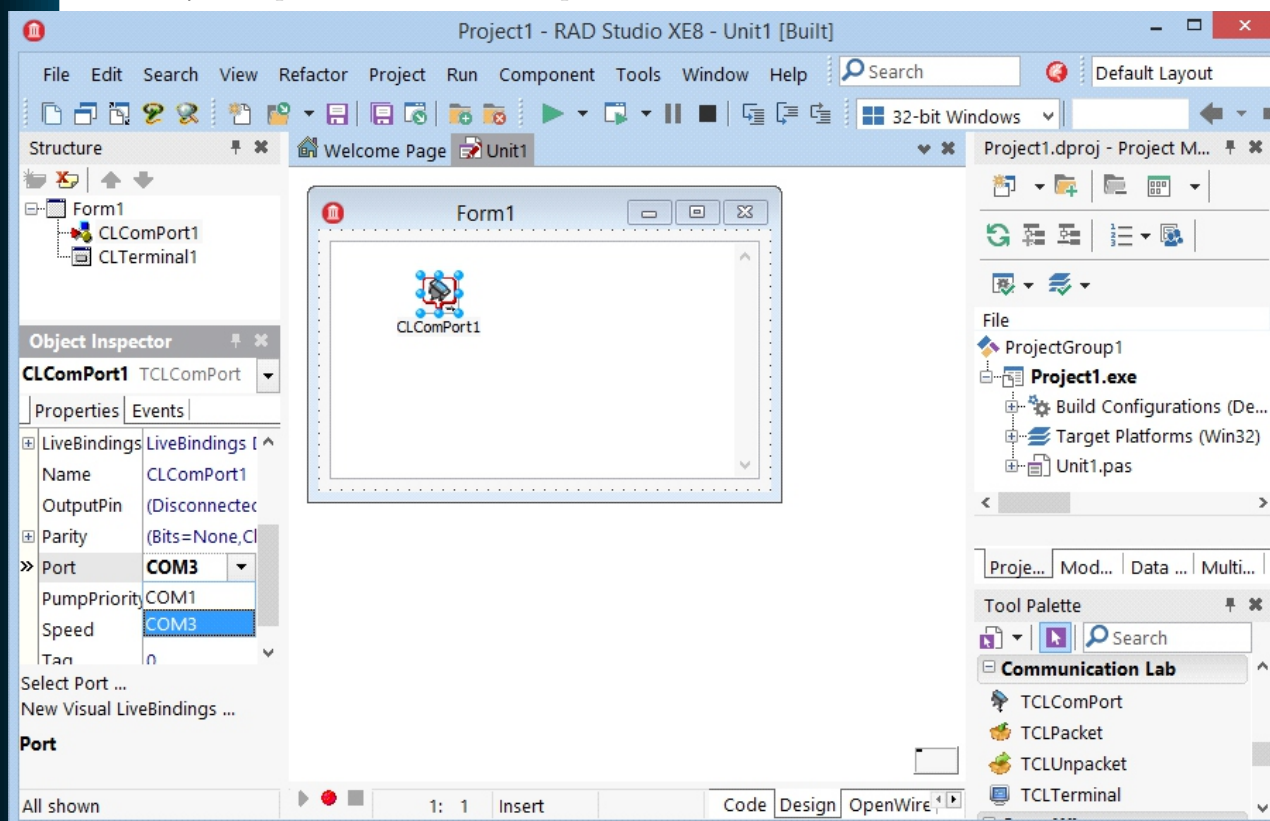


Now that we have the Arduino code working, it is time to see how we can connect to it from Delphi. Start Delphi.

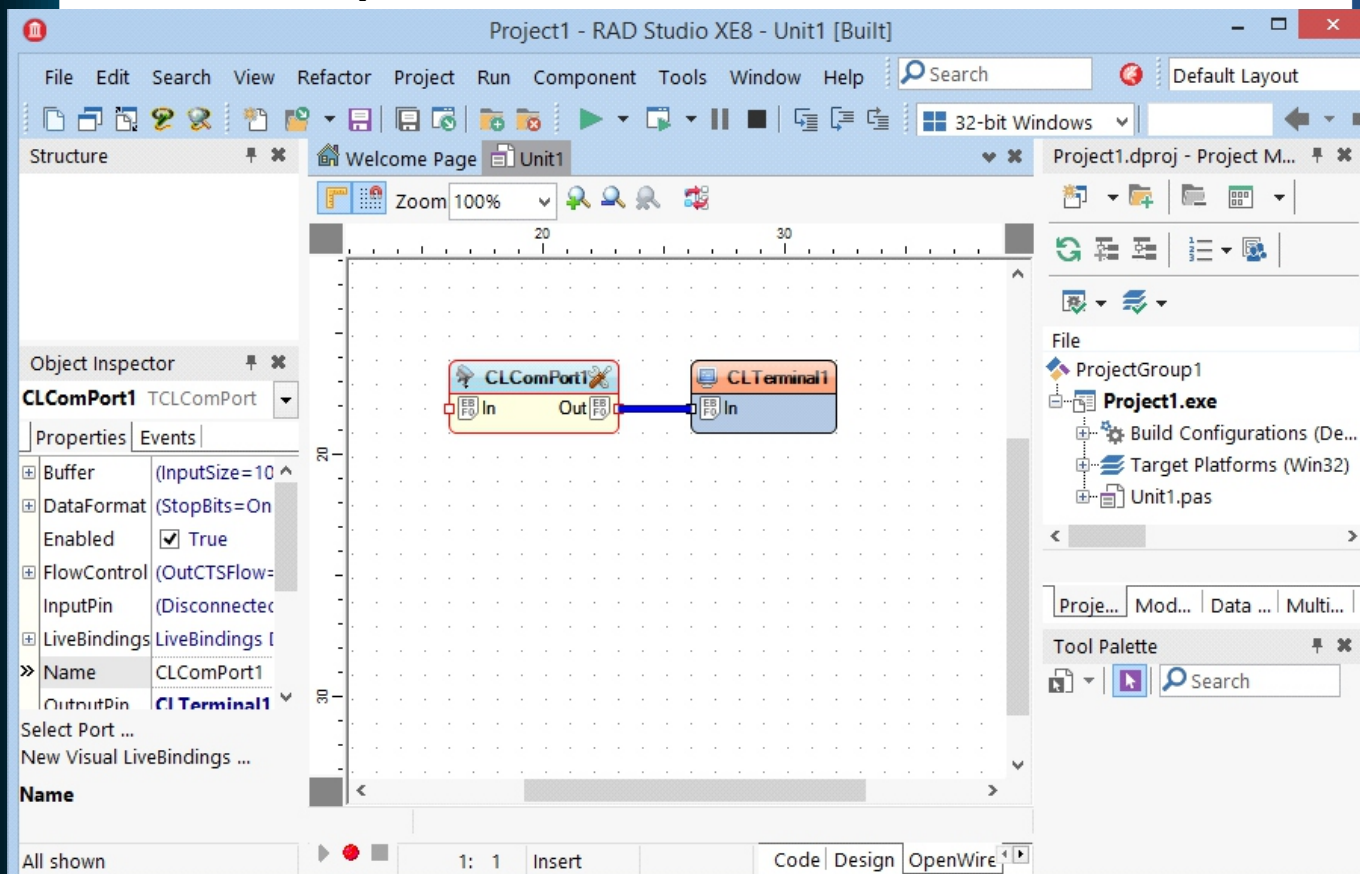
From the Component Palette, drop
TCLComPort, and TCLTerminal:



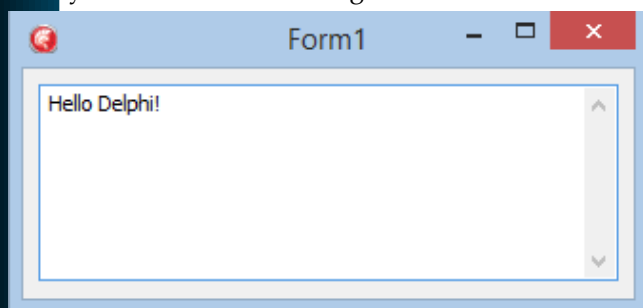
In the Object Inspector, select the COM port to which Arduino is connected:



Switch to the **OpenWire** view, by clicking on the OpenWire tab and connect the OutputPin of the CLComPort1 to the InputPin of the CLTerminal1:



Compile and run the application. If you press the reset button on the **Arduino** board, you will see this message:

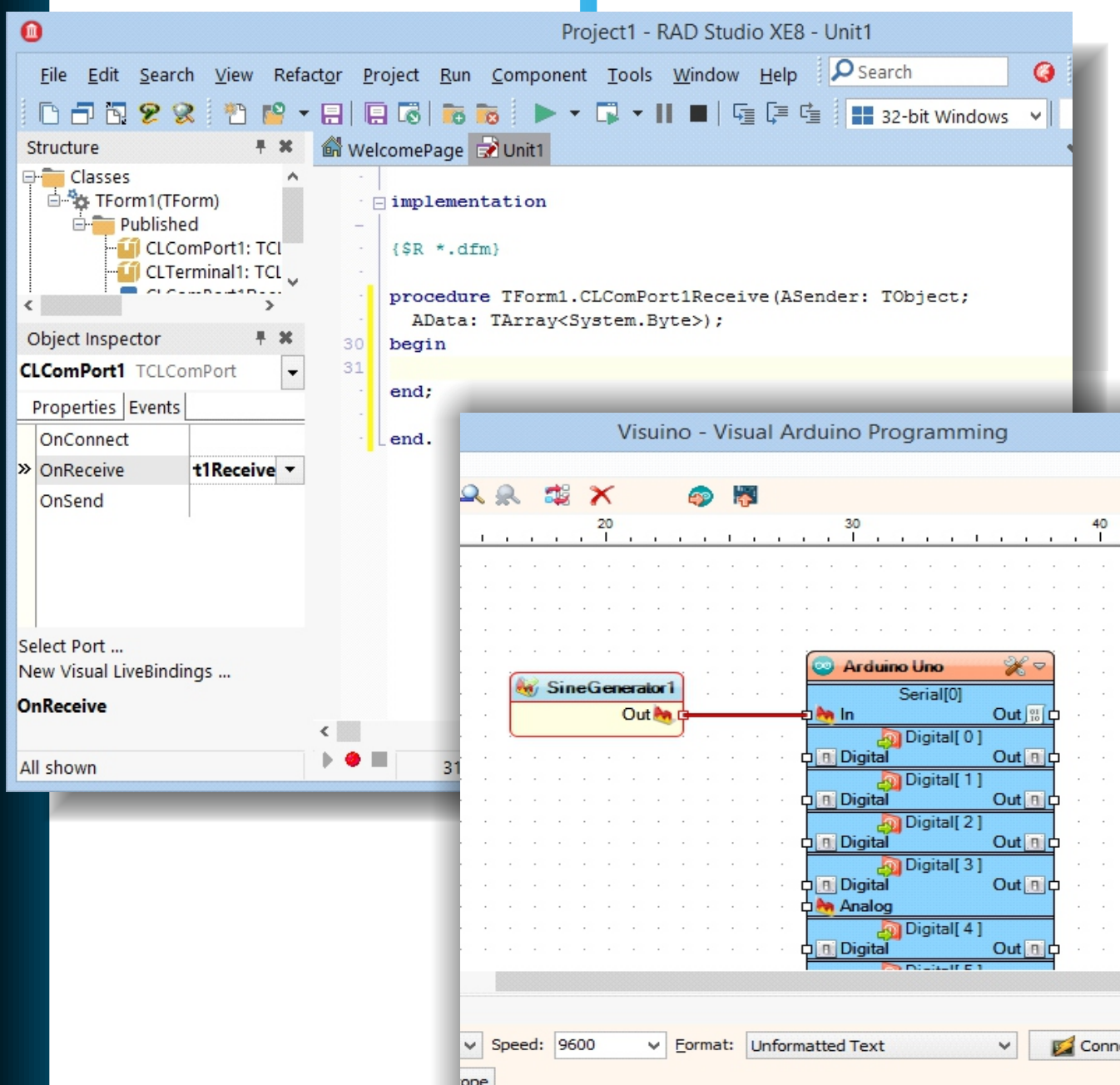


The `TCLComPort` also has `OnReceive` event where you can receive and process the data from your code:

The data arrives in binary form. Since in this case we know that we are sending text, we can convert the data to text:

```
procedure TForm1.CLComPort1Receive (
    ASender: TObject; AData:
    TArray<System.Byte>);
var AText : String;
begin
    AText := TEncoding.ASCII.GetString ( AData
);
end;
```

If you want to receive data from Arduino, we can connect a data source such as one of the analog pins, or a Sine Generator. We will use the generator, as it is the easiest to experiment with. From the toolbar, drop a Sine Generator, and connect its output pin, to the input pin of the Arduino Serial Port:



Press **F9** to automatically generate the C++ code and automatically open the Arduino IDE, then compile and upload the sketch, as we did earlier.

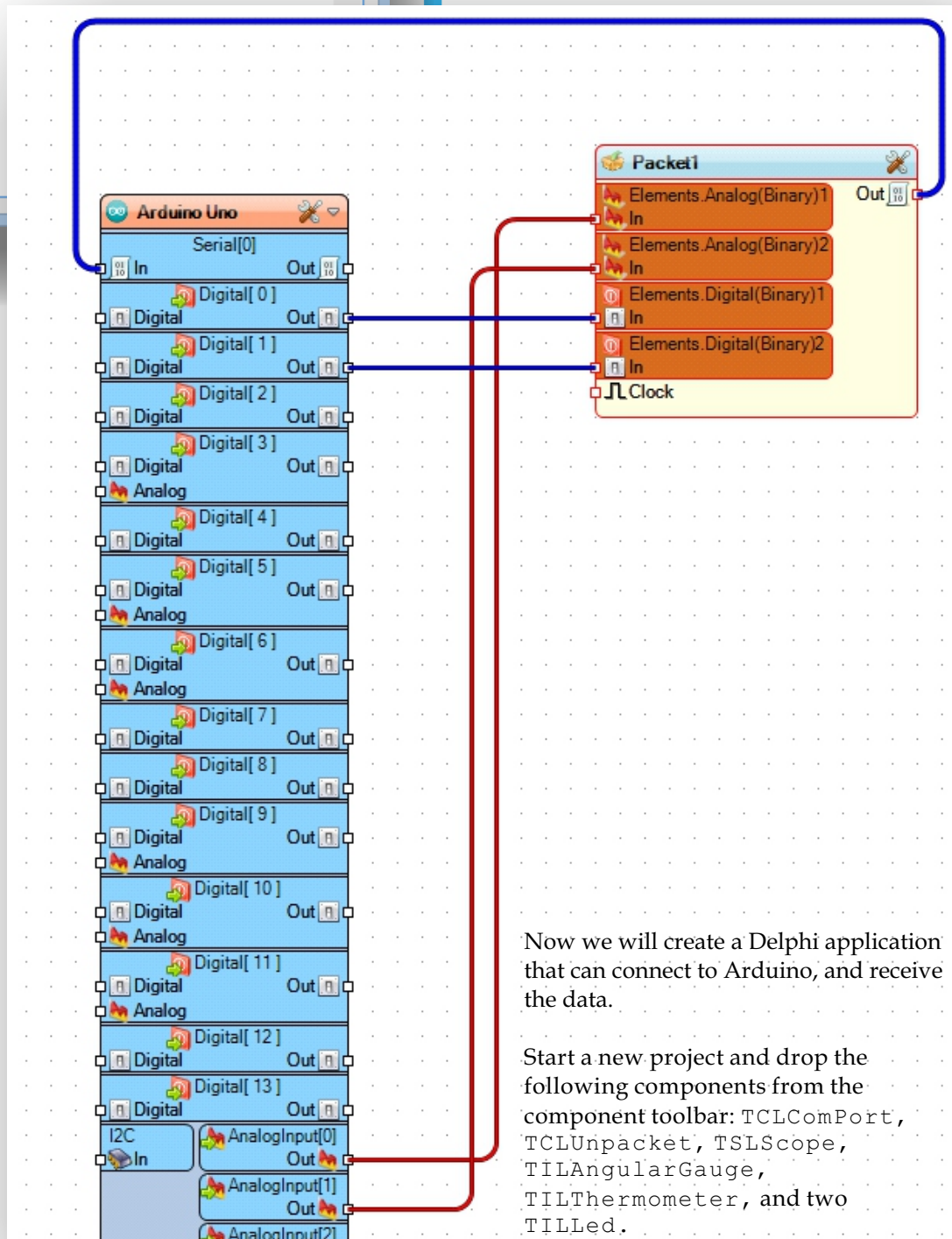
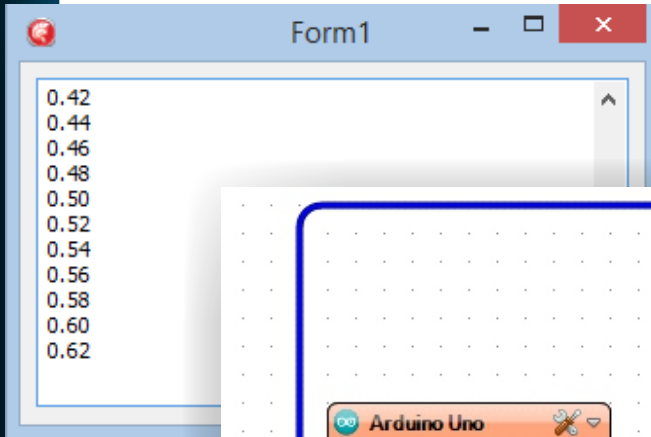
If you run your Delphi application now, you will receive the data from Arduino:

Receiving and visualizing data from one channel is fine, but the data arrived in text format, and is difficult to use directly for calculations, and processing.

It also limits us to a single data channel.

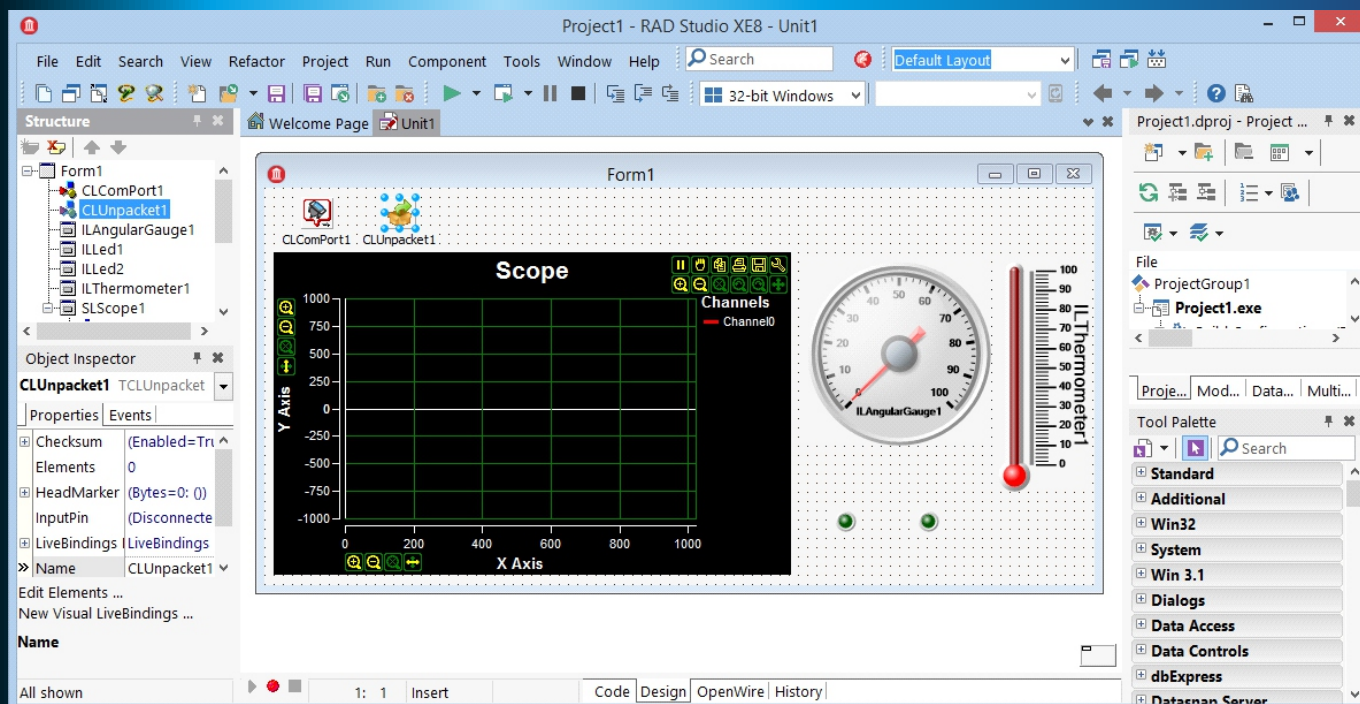
In the previous issue, we learned how we can use the Package component, to receive data from multiple sensors at the same time.

Here is the project we did:

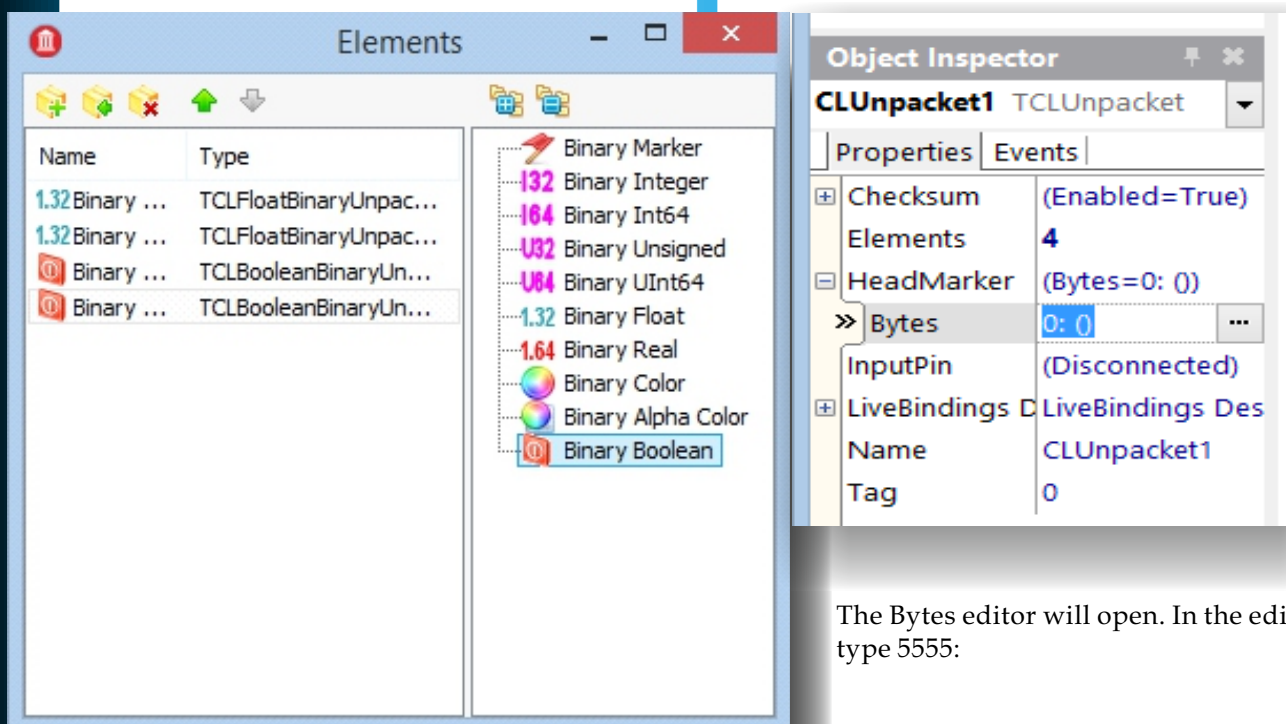


Now we will create a Delphi application that can connect to Arduino, and receive the data.

Start a new project and drop the following components from the component toolbar: TCLComPort, TCLUnpack, TSLScope, TILAngularGauge, TILThermometer, and two TILLed.



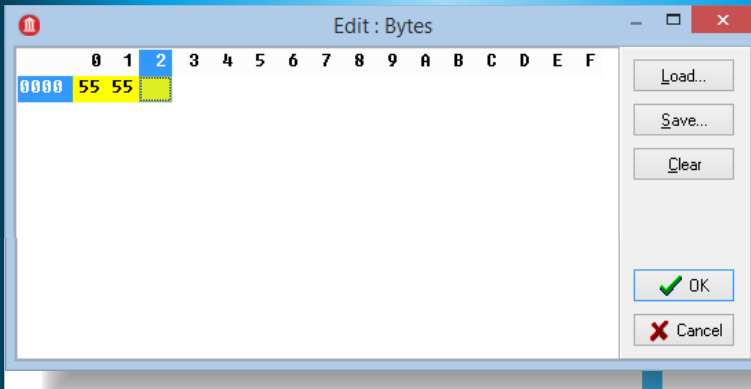
Set the COM port as we did earlier in the previous Delphi project. Then double click on the CLUnpacket1, then add 2 Binary Float channels, and 2 Binary Boolean channels:



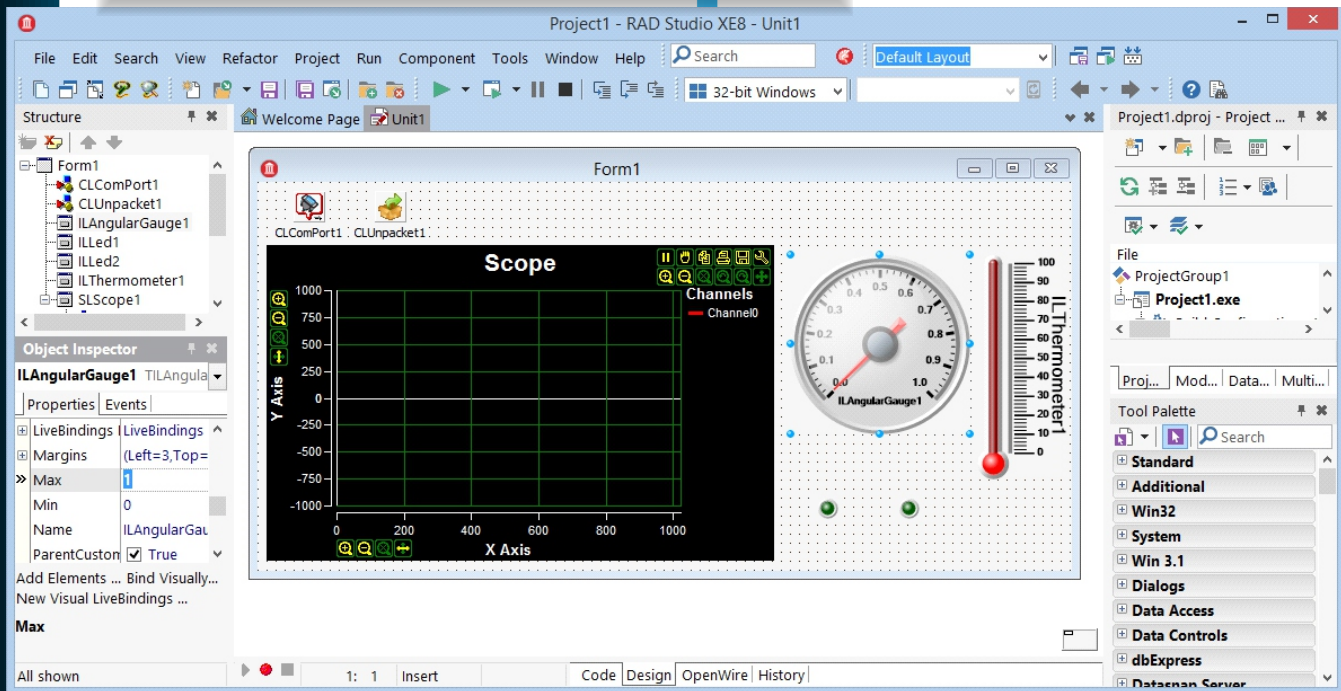
The Bytes editor will open. In the editor, type 5555:

Close the window.

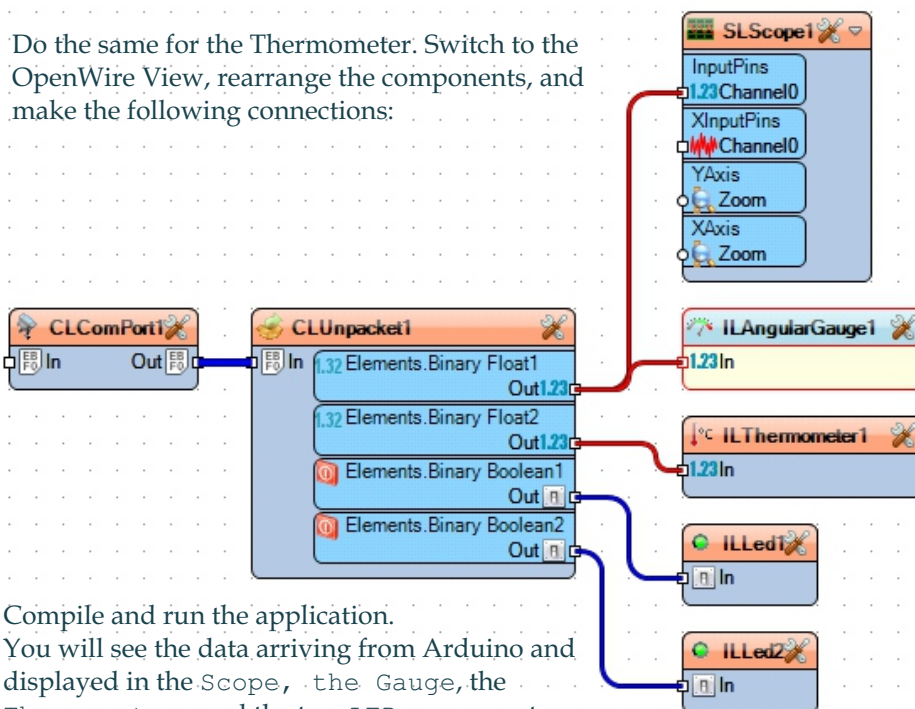
For the CLUnpacket1, in the Object Inspector, expand the HeadMarker property, and for the Bytes, click on the "..." elipsis-button.



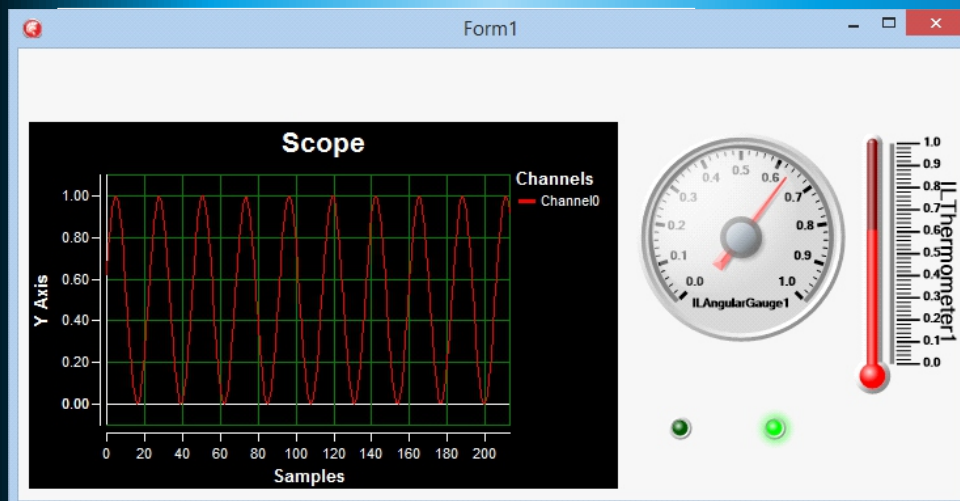
Click OK to close the editor.
Select the Angular Gauge, and set its Max value to 1, since we expect the data we receive to be between 0.0, and 1.0 (*The Visuino Analog Inputs, and Outputs are normalized*):



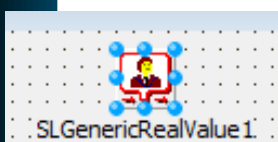
Do the same for the Thermometer. Switch to the OpenWire View, rearrange the components, and make the following connections:



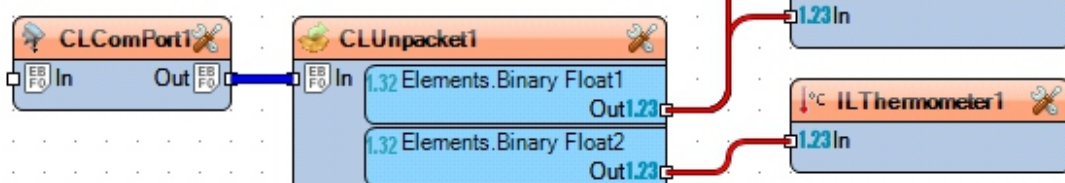
Compile and run the application.
You will see the data arriving from Arduino and displayed in the Scope, the Gauge, the Thermometer, and the two LED components:



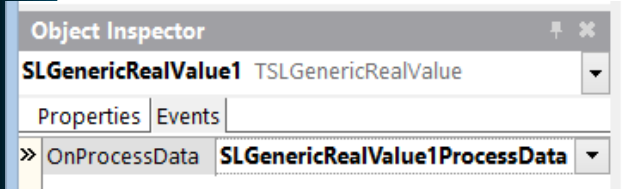
We already know how to visualize the data easily in Delphi. Often however we need to work with the data inside our code. To get the Analog data in the code, add a `TSLGenericRealValue` component:



You can connect the component to one of the Analog Floating Point channels:



The `SLGenericRealValue1` has `OnProcessData` event, where we can write our code:



As example we can assign the value to the position of a Progress Bar:

```
procedure TForm1.SLGenericRealValue1ProcessData(
  Sender: TObject; InValue: Real;
var OutValue: Real;
var SendOutputData: Boolean);
begin
  ProgressBar1.Position := Round( InValue * 100 );
end;
```

There are similar components available for Boolean and other data types as well.

You have learned for you can create Arduino code, that collects and send data, and how to create Delphi application that receives and processes that data over serial channel. In the following issues we will show you how you can do the same over the network, and how you can have multiple Arduino devices talk to each other and to Delphi.

In short we will introduce you to the Internet Of Things with Delphi and Visuino.

