ARDUINO: THE VISUINO PROJECT - PART 1 PAGE 1

BY BOIAN MITOV

starter expert Delphi

The Arduino board: a guide to quickly get you started using Delphi ...
or: How a new product comes into being.
How to quickly and efficiently start using Arduino, and connect to it from Delphi, and how a new product is born.

Today as Delphi and FPC developers, we feel masters of our Desktops, Servers, and Mobile devices, but we lack that mastery once we are outside this digital world. Whenever we need to control some other equipment, or collect some interesting data from sensors, we feel that lack of power, that we have grown to enjoy.

This was the feeling that drove me at the end of the last year, to buy myself an Arduino board with a small starter sensors and peripherals kit.

Arduino is an open source hardware platform, developed initially primarily by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino and David Mellis. The Arduino team/teams also manufacture and sell Arduino boards, however since the platform is Open Source, there is a huge number of other manufacturers of Arduino compatible boards ranging in prices from ~$1 to ~$100, and with huge range of sizes, some smaller than a quarter, and a huge range of capabilities, from few digital and analog pins, all the way to WI/FI, and GSP enabled boards, with high number of digital, and analog pins.

This great diversity, in capabilities, sizes, and prices, makes Arduino a very attractive development platform, for almost any project that requires monitoring, interfacing or controlling the world outside our comfortable Delphi/FPC controlled boxes, or at least that is what I thought...

I received my Arduino KIT, anxiously unpacked it, and hooked it to my system.
So far so good, it started to blink.
Then I started digging the web to learn how to program it, and my fantasy to control the world with it came crashing down on me. The Arduino boards are usually programmed using a very simple to use but extremely rudimentary IDE called Arduino IDE, in C/C++, with very low level code, that requires fairly deep hardware, and firmware knowledge to do even relatively simple tasks.

It took me hours to put even fairly simple code together - reading from one sensor, and controlling a motor. Hardly impressive achievement.
To become the master of the world with my Arduino, I would need a better approach.
Since obviously, nobody else had solved the problem I was facing, I decided to do it myself, and got down to work. Here is what my goal was:

1. In order a development tool to be attractive to inexperienced developers it must be very easy to use. **It had to be easy enough for a kid to learn it, and it should not require you to learn programming language or programming techniques.** In short, it should be intuitive visual design tool.

2. In order for it to be attractive to experienced and expert developers, it must produce compact, and highly efficient, scalable code.

3. Since Arduino didn't have an operating system, it must allow components to be designed as a collaborative plug-in framework, so they can work together, and not interfere with each other. In essence a small rudimentary, component oriented OS core was needed.

Fortunately for me, I was already experienced in developing high performance component frameworks for Delphi, C++ Builder, and .NET, as well as haaving a solid hardware background.

I also had a ready graphical development IDE for Windows called OpenWire Studio. OpenWire Studio is a very flexible and open architecture Graphical IDE, and can easily be adapted to program almost anything.
At least in theory, now here was the chance for it to prove that it is up to the task solving my little Arduino problem.
I started with the most difficult of the problems. Writing a component framework and very rudimentary scheduling functionality. For this I used a very rudimentary lightweight flavor of the good old OpenWire, implemented in C++.
Next I created few Delphi components and using the new Mitov.Runtime RTTI started to generate corresponding C++ code for Arduino from them.
Finally I cloned the OpenWire Studio, and installed the newly created components in it.
And the first version of Visuino was born. All that was needed was to add buttons to generate the Arduino code, and to start the Arduino IDE so I could compile the code.

This happened exactly in the middle of the Delphi Week, celebrating 20 years of Delphi, and so as I did a brief live interview about my experience with Delphi over the years, David I and Jim McKeeth, suggested to show the Visuino as example of what can be achieved with Delphi.
So it was, that the first people to see the product live in action (*bugs and all*), were the Delphi fans watching the Delphi Live broadcast!
At this point I already had achieved all the goals I had in mind, when I started the project.

All I needed was to write more components, and play with it, but soon I started to discover more shortcomings. I was able to program my Arduino with great ease, but my Arduino and my PC were living in separated worlds.

I wanted to see on my screen, what my Arduino was collecting as data, or processing. As a minimum, I needed a terminal window, so I created an OpenWire serial port component, and hooked a terminal window, with the necessary user interface. I was able to see my data, but it was in text format. What if I need to see it in a plot?

So using PlotLab, I naturally added a Scope component and hooked it to the serial port.
The Visuino was shaping very well. Not only could I program my boards with it, but I could also monitor and plot the data from one channel.
But what if I need to monitor more channels?

I still was not satisfied. To send data from multiple channels, over a single communication channel, I needed to package the data in some form of structure, so I designed a package and un-package components that allow the data to be packaged and transmitted as a structured packet.

Now I was able to plot multiple channels easily. I went even further, allowing Visuino to automatically configure the scope from the package format in my Arduino design.
Scope piloting was nice, but sometimes we want to see the data in Gauges, and LEDs so I decided to add instrumentation view as well, using the InstrumentLab component package. My Visuino was feature complete, I was happy, and I already had Delphi components developed that allowed me to easily communicate with the board.

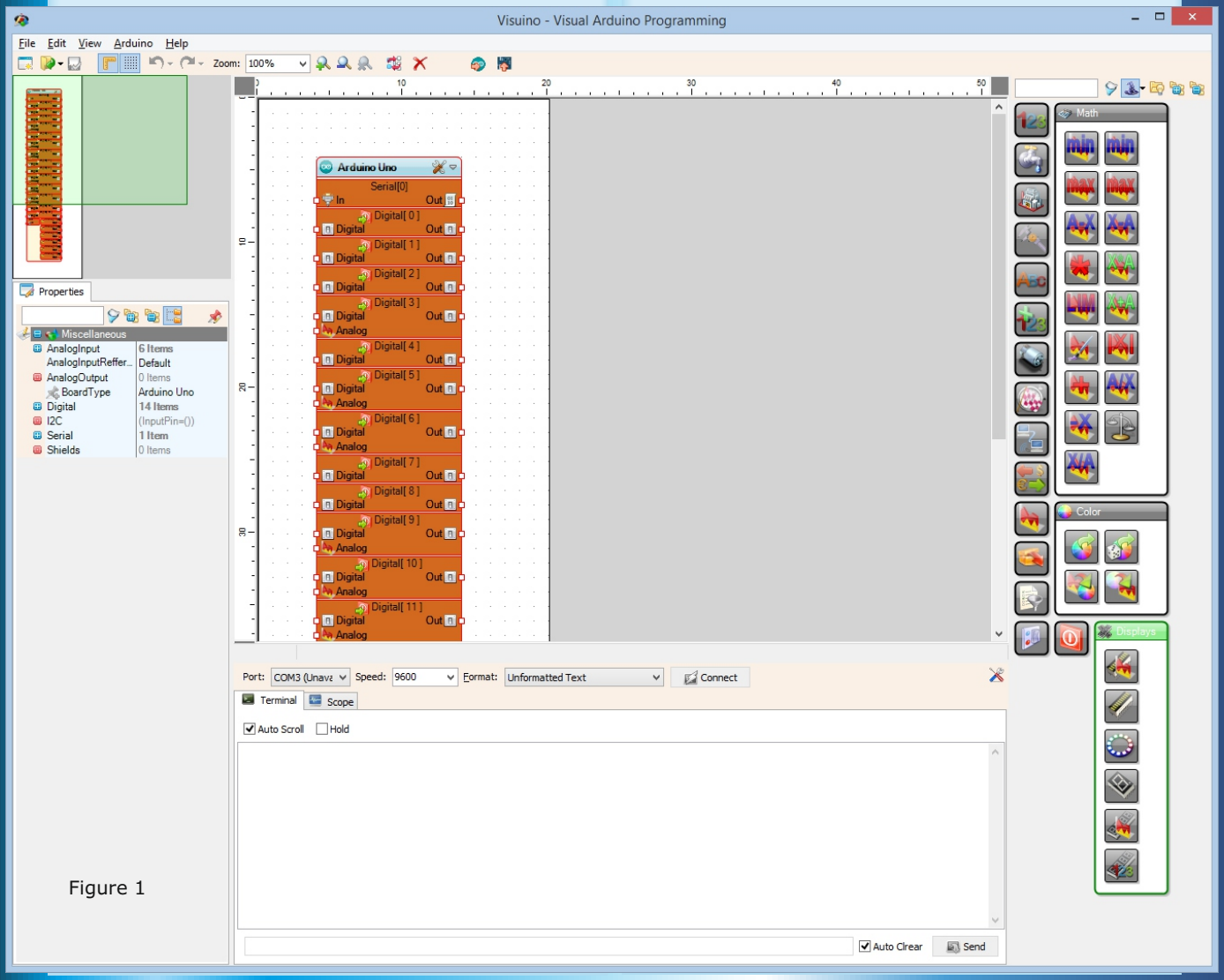


Figure 1

The next logical step was to package and release them, so other developers can communicate with Arduino from their Delphi applications.

Now that we are trough with the short Visuino history, it is time to take a real look at it.
You can download the **Visuino Beta** by visiting **www.visuino.com** .

Before installing make sure you have the latest version of the Arduino IDE installed.
Currently this is 1.6.4 and is available for download here :
http://www.arduino.cc/en/Main/Software .
The Arduino IDE comes with all standard Arduino libraries that will be needed to compile the code generated by Visuino.
After installing the Arduino IDE, you can go ahead with the Visuino installation.
If you have done a default installation of your Arduino IDE, there will be no need to do any configuration of Visuino.
Once installed you can start Visuino.

In the center you will have the **Visuino's design area**. This is where you will visually design your project.

On the right of figure 1 (page before) is the component toolbar where the components are organized in different categories and sub-categories.

In the top left corner of figure 1 (page before) is the overview navigation area, and bellow it is the property editor.
Below the Design Area is located the Serial Terminal the Scope and - as we will see later - the Instruments Panel.

The best place to start learning is to watch the Visuino video tutorials:
https://www.youtube.com/watch?v=v-yMtIzgIeU,

and

https://www.youtube.com/watch?v=wKKlhgKtDoI

The next step is opening the included demo projects. You can easily access them from the menu by selecting |File|Open Demo...|

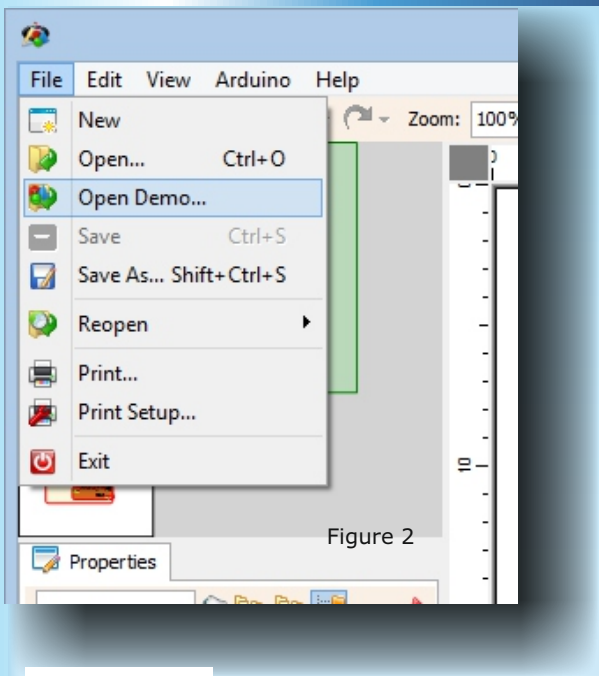


Figure 2


Figure 3

Once you have familiarized yourself with the demos, you can try to create a new project yourself. The design area always contains an Arduino component.
You can select the Arduino board type by setting the "BoardType" property or by clicking on the setup icon of the Arduino component.

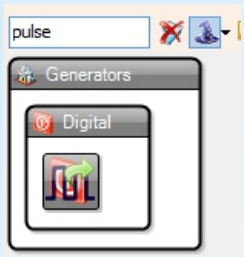


Figure 4

The simplest project is a blinking **LED. Drop a Pulse Generator** from the toolbar.

Connect it to pin 13 of the Arduino component.(figure 5)

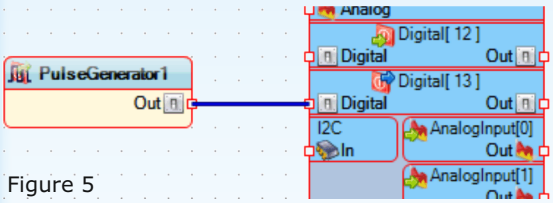


Figure 5

Now your design is ready, and you can generate the Arduino code by clicking on the "Send to Arduino IDE for Compilation" button:

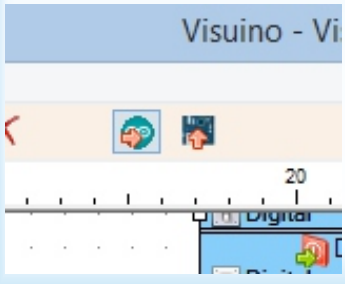


Figure 6

This will generate the Arduino code, and launch the Arduino IDE where you can compile and upload the compiled code into your Arduino board:

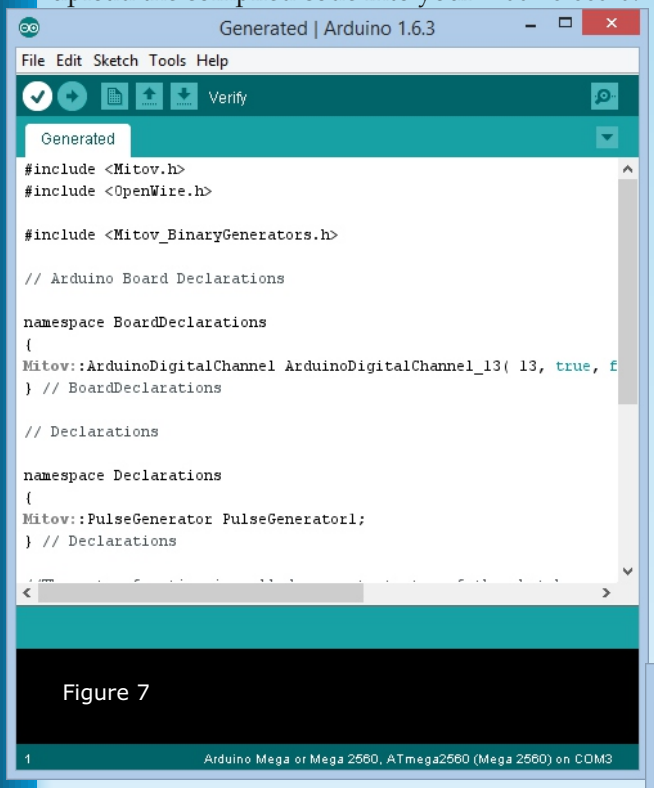
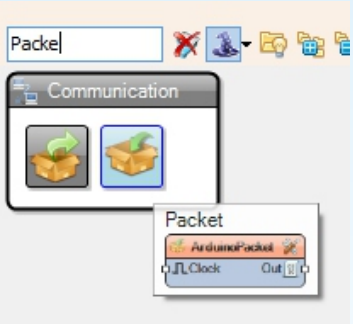
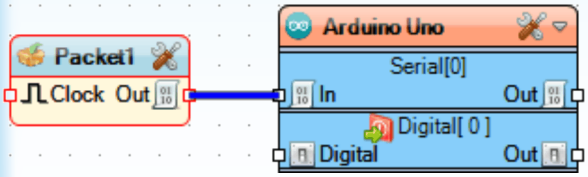


Figure 7

Next we can connect a few sensors to the Arduino pins, and monitor them in Visuino.
Start a new Visuino project, by selecting |File|New|.
Add a "Packet" component to the Design:



And connect its output to the Input of the Serial Port (*The software input for the serial receives data that will be sent as serial output from the hardware.*)



In the property editor expand the HeadMarker and click on the "..."ellipsis button:

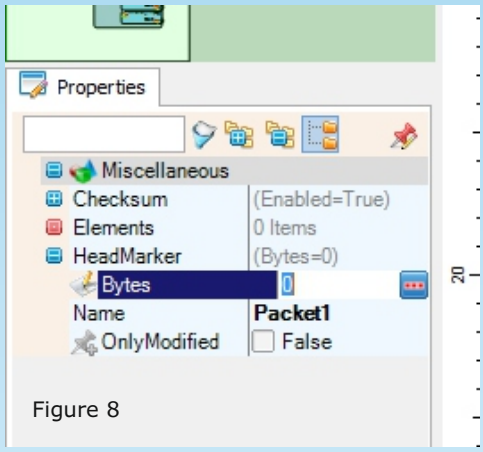


Figure 8

In the Bytes editor type `55 55` . This will be used to identify the starting point of a package.
You can use any number of bytes, with any values, but 2 bytes are a good choice, and common values such as `00 00` should be avoided as they often appear in data. The component makes sure the header is properly recognized even if 55 55 is present in the data, by special encoding.

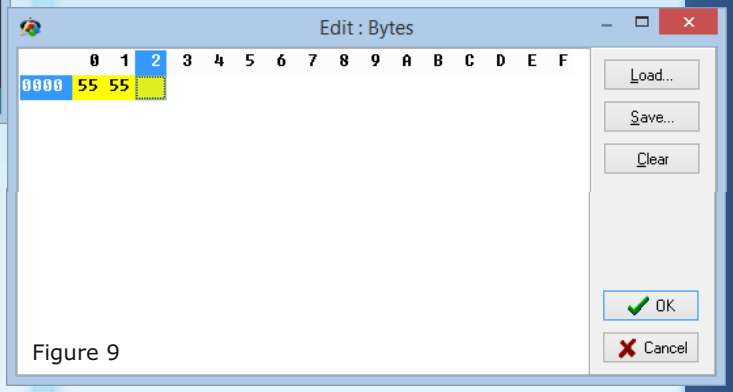


Figure 9

You can close the editor by clicking on the OK button.

Double-click on the Packet1 to open the elements editor:

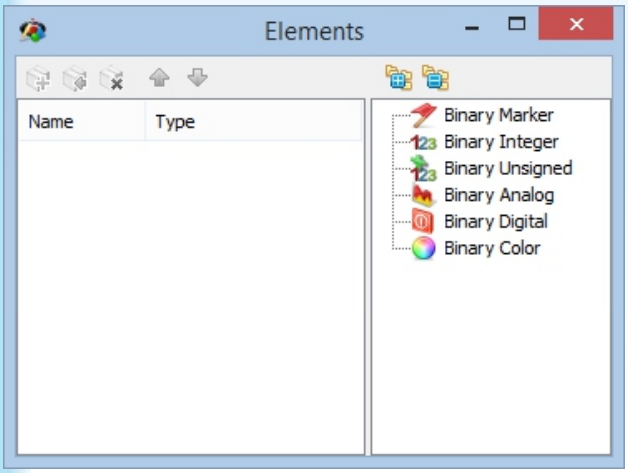


Figure 10

With this editor you can add data channels to the packet. For this example I will add 2 Analog and 2 digital channels.
Now we can connect the packet elements to the pins where our sensors are connected:

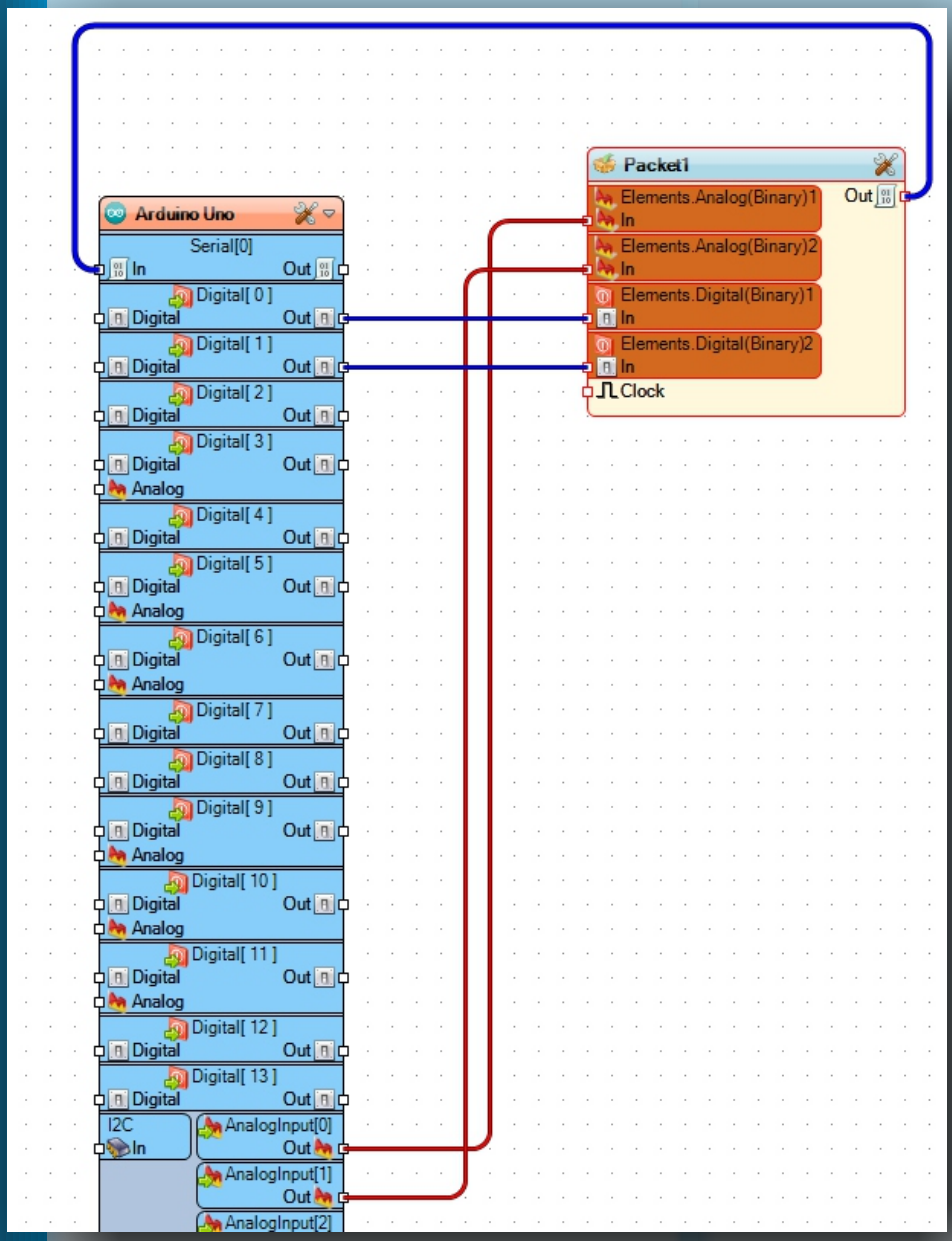


Figure 11

Your design is ready. You can generate, compile, and upload the code to your Arduino. Now you can use the Visuino Scope and Instrumentation Panel to view the data. Select the com port to which the Arduino is connected, and from the "Format" drop-down select Packet1: Then click Connect.

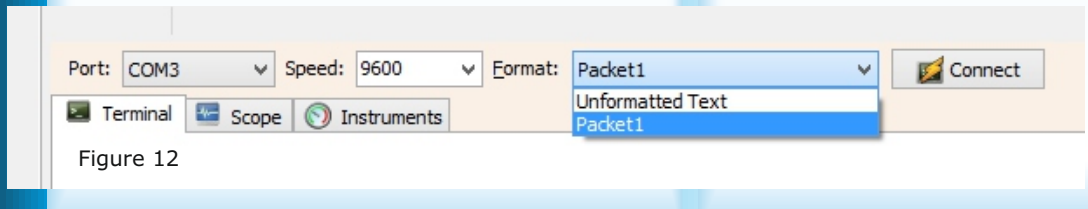


Figure 12

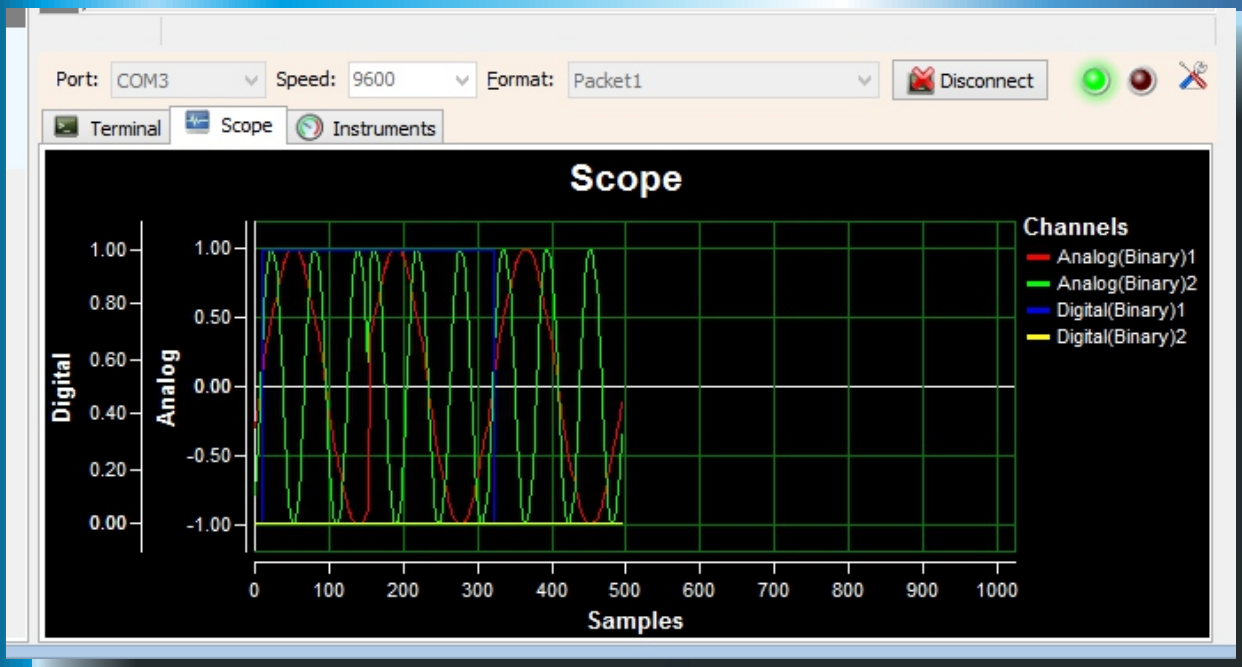


Figure 13

In the scope you can see the data arriving from the sensors: see figure above

And the same data can be seen in the Instrument Panel:

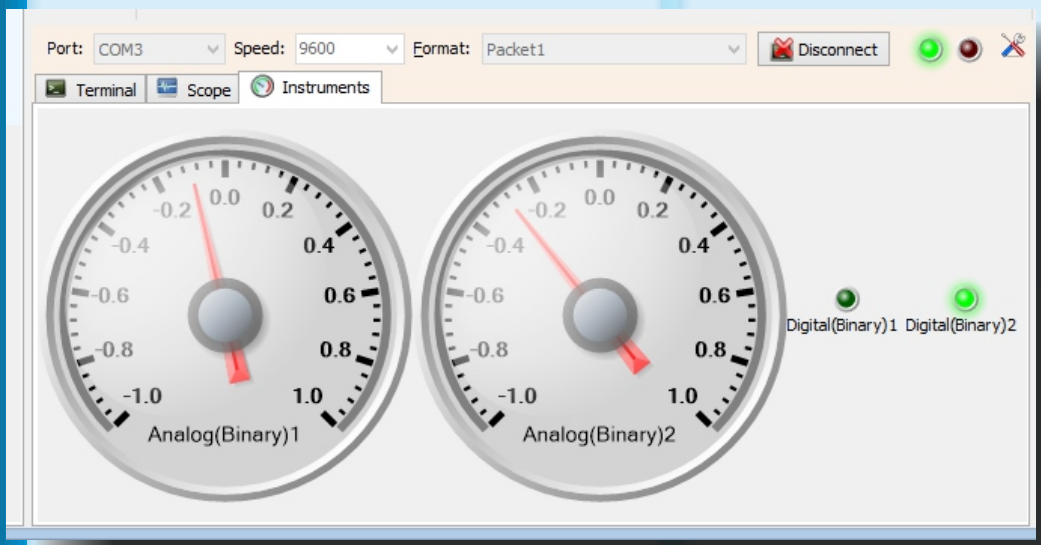


Figure 14

You have seen how you can receive and visualize data from sensors in the Visuino. In the next issue you will learn how to receive data from Arduino into your Delphi applications.