SCADA

Getting Started Guide

Contents

Introduction to SCADA

Supervisory Control And Data Acquisition (SCADA) is a control system architecture that uses computers with graphical user interfaces to provide high-level supervisory management of processes or machinery. It uses networked data protocols to communicate with peripheral devices including sensors, programmable logic controller (PLC) or discrete PID controllers, that interface directly with process plant or machinery.

SCADA computer systems provide operator interfaces that enable monitoring and the issuing of process commands, such as controller setpoint changes. Most commonly, the real-time control logic or process calculations are performed by networked modules that connect to the field sensors and actuators.

Matrix SCADA

Matrix SCADA is a low-cost and simple way of allowing visual and feature-rich PC control and data acquisition for electronic and electromechanical systems. It allows users to quickly and easily create PC-based control and data acquisition programs with graphical Human Machine Interfaces (HMI) based on flowcharts.

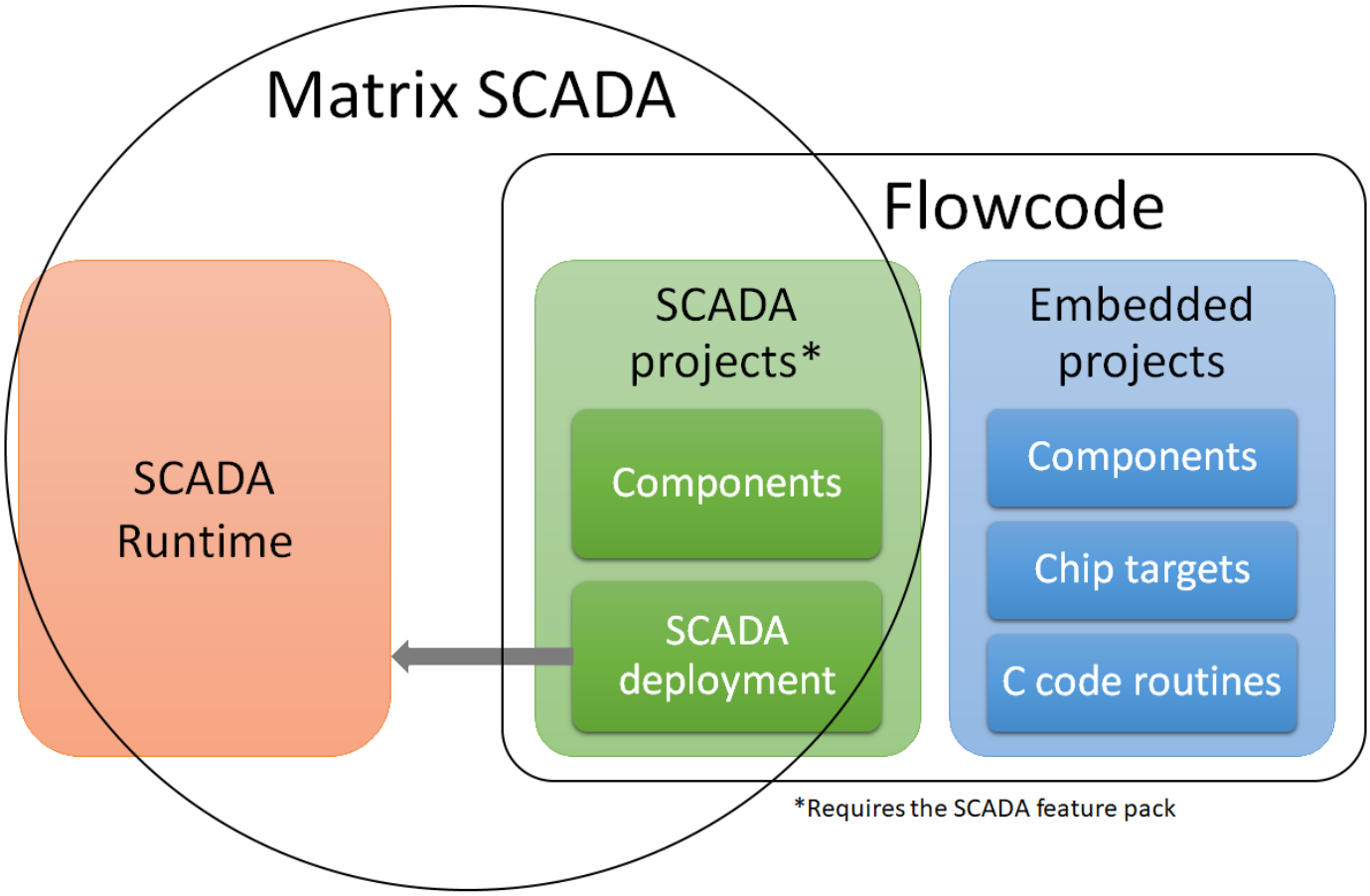
Matrix SCADA is comprised of two components: Flowcode SCADA, and the Matrix SCADA runtime.

As a developer, you can create a Flowcode SCADA program, and then export it as a Matrix SCADA runtime application. This Matrix SCADA runtime application can run standalone, on any Windows PC, without a license. This allows for great portability and lets you create sellable items that you can roll out to customers without writing a dedicated application from scratch.

The Matrix SCADA runtime version of your program is protected from editing, and its code is hidden. You can provide your end-user access to the traditional Flowcode panels, such as the 3D/2D, Console or Data recorder panels.

The main goal of Matrix SCADA is to remotely control hardware from your PC, whether it's on your bench or on the other side of the world. Applications may include the development of software to control the following: production batch testing, home automation, IoT, robotics / embedded systems control, and industrial systems control.

Breakdown of Matrix SCADA



Matrix SCADA is the name of the technology that allows you to create SCADA projects in Flowcode, and then run them independently as standalone applications built on top of the Matrix SCADA runtime.

Users of applications built on top of the Matrix SCADA runtime do not require a license. Users of Flowcode SCADA require the SCADA feature pack.

Overview of Matrix SCADA application creation

Users write programs in Flowcode SCADA using the following programming views: flowcharts, pseudocode, or blocks. Simply drag icons onto the views, double-click to see their properties, and set parameters to create the program. A number of premade visual components including meters, 7-segment displays, dials, switches, and indicators can be dragged onto the 2D/3D panels to show relevant information and allow user control of various interfaces. Flowcode is supplied with a large number of prewritten macros for control, data processing, communications, and visual data representation. It is designed to interface well to other analysis tools.

Summary of Matrix SCADA application creation:

- Create a new Flowcode SCADA project
- Drag and drop visual components onto the 2D/3D panels
- Drag and drop program icons onto the programming views
- Test directly by running the application
- Deploy the project as a standalone Matrix SCADA runtime application

Here is a summary of some of the visual and communication components included with Matrix SCADA.

- HMI components (Operator GUI)
  - Switch – push to make or latching
  - LED indicators
  - 7 segment display – 1 to 5 digit
  - Meter – variable axis
  - Meter - circular
  - Dial
  - Slider
  - File I/O
  - Speech / Audio output

- Network / Internet communications
  - Webserver
  - MQTT
  - Modbus
  - vNet

- COM port communications
  - USB
  - RS232
  - RS485
  - Bluetooth
  - Modbus

Developers can also write programs that create message dialog windows, and create dialogs that allow users to input text in the traditional way, all using the built-in Flowcode API.

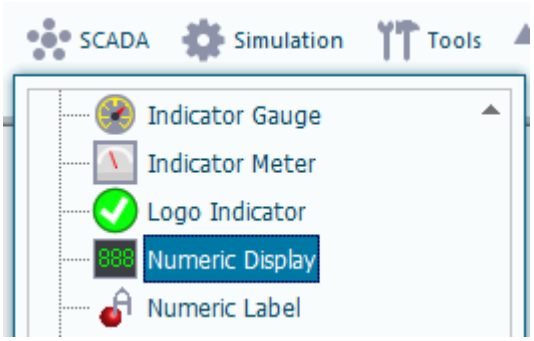
Creating a Flowcode SCADA application

This walkthrough will demonstrate how to create a Flowcode SCADA application that retrieves and displays weather information from a provider host such as the BBC weather RSS feed. It is assumed that the user has a prior working knowledge of general Flowcode use.

Run Flowcode and select “New Project” -> “SCADA”. At the Project Options dialog Select “GENERIC” and click “Ok”.

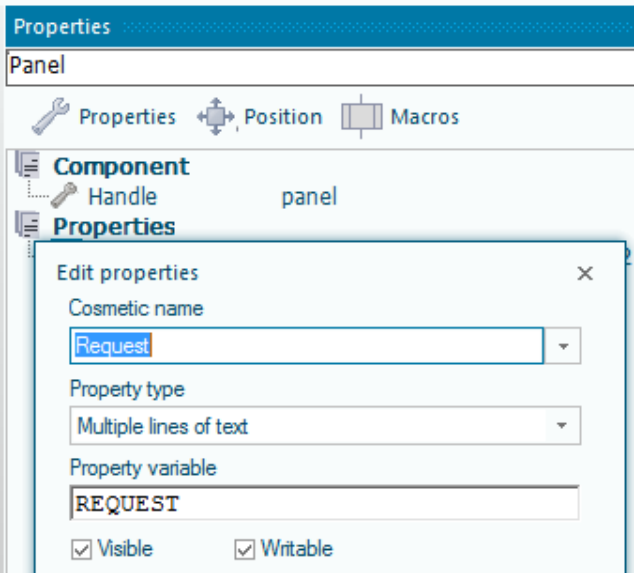
Select the “Comms” component category and drag a “Network Communications” component onto either the 2D or 3D panel.

In the “Component Properties” pane, set the “Handle” property to “network”



Select the “SCADA” component category and drag a “Numeric Display” component onto the same panel as the “Network Communications” component..

In the “Component Properties” pane, set the “Number of Digits” property to “3”. For our convenience we will also rename the “Handle” property to “TemperatureDisplay”



In the “Component Properties” pane, ensure that “Panel” is selected in the drop down at the top. Add a new property and name it “Request”. Set the “Property type” to “Multiple lines of text”. Finally, set the “Property variable” field to “REQUEST”. Click “Ok”.

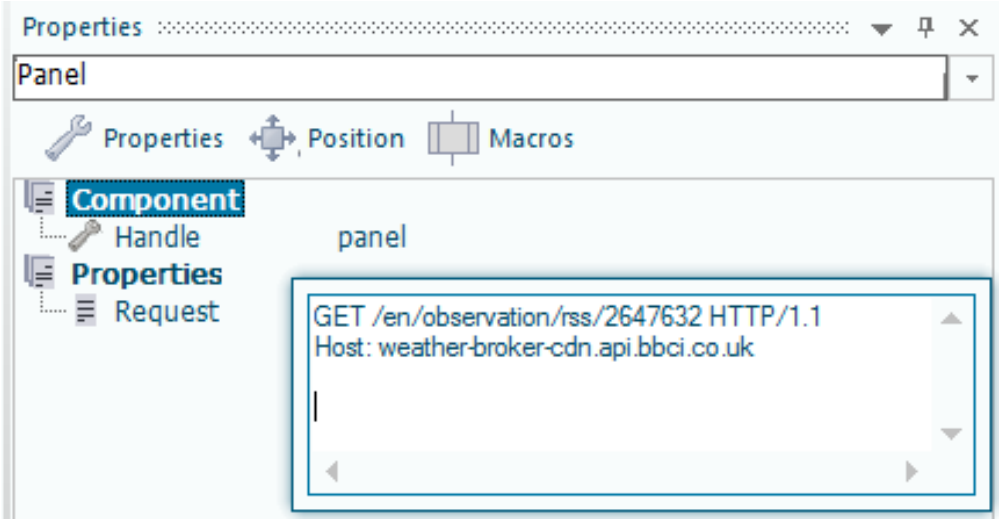
We will use this property to hold the GET request sent to the server.

The actual request data is as follows:

GET /en/observation/rss/2647632 HTTP/1.1

Host: weather-broker-cdn.api.bbci.co.uk

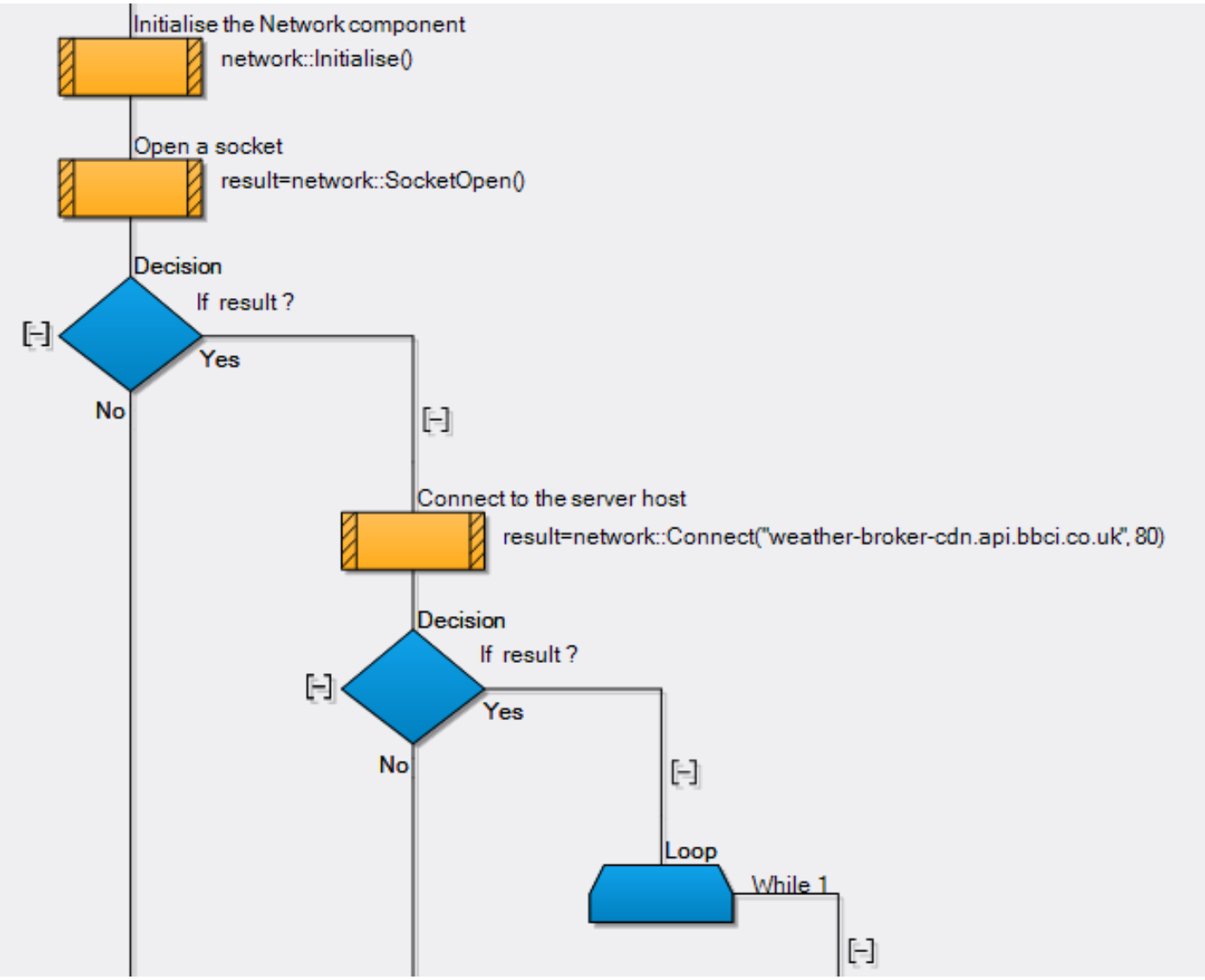
The request text includes the 2647632 code for Halifax, other towns and city codes can be found on the BBC website.



Add this text to the “Request” property's second column, ensure that enter is pressed twice after the “co.uk”: this adds two blank lines.

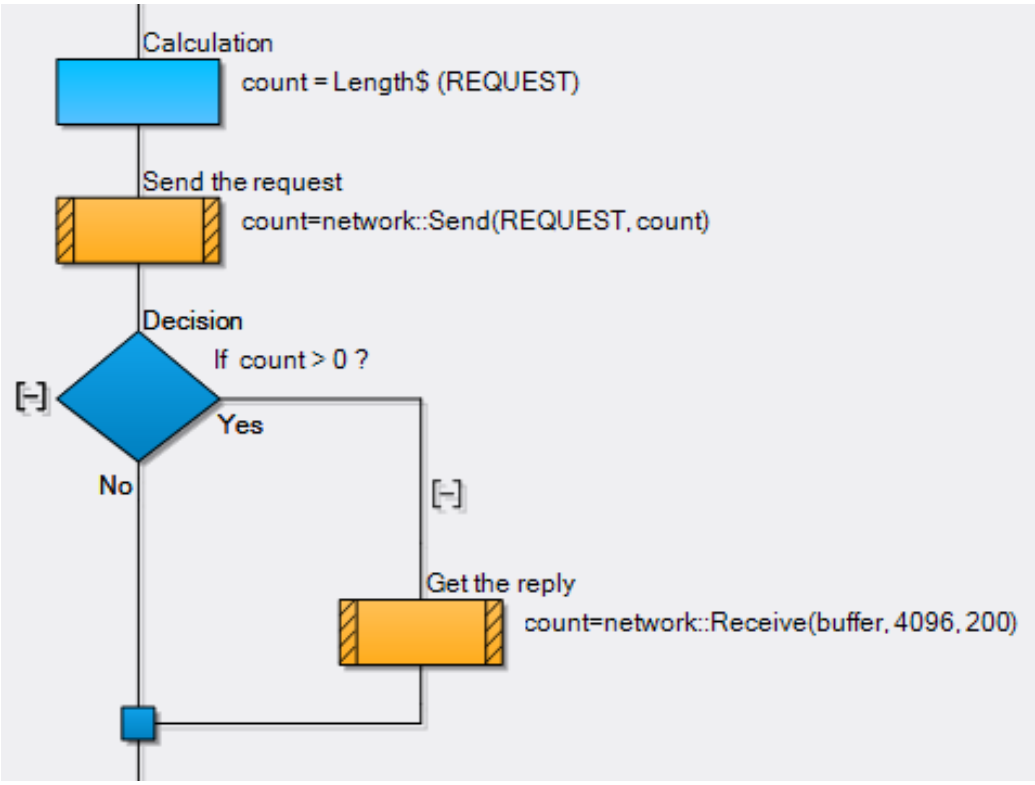
Please note that use of the BBC Weather content is subject to their Terms of Use. For example, if the project is not limited to personal testing, you might need to add a text label "BBC Weather" or "bbc.co.uk/weather".

We are now ready to create the controlling flowchart. Add the following macro calls belonging to the “network“ component: “Initialise” and “SocketOpen”. Add a “Connect” macro call to “network” and set the parameters “Address” to "weather-broker-cdn.api.bbci.co.uk", and “Port” to “80”, as shown below.



For the “Yes” branch of the “Connect” result we will enter an infinite loop requesting data from the server.

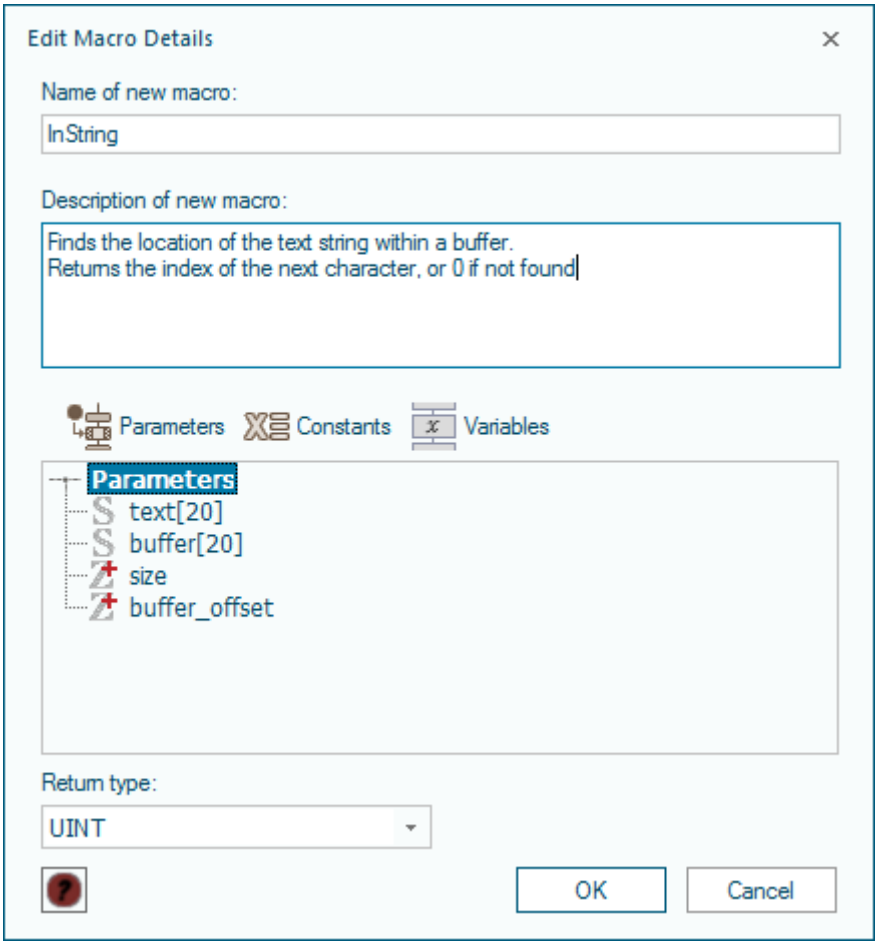
We show the contents of this infinite loop below



First we determine the request text length.
Then, using the “Send” macro, we send the GET request, which is the contents of our REQUEST property.

If the “Send” is successful then we call “Receive” to get the reply into a global buffer of size 4096 bytes.

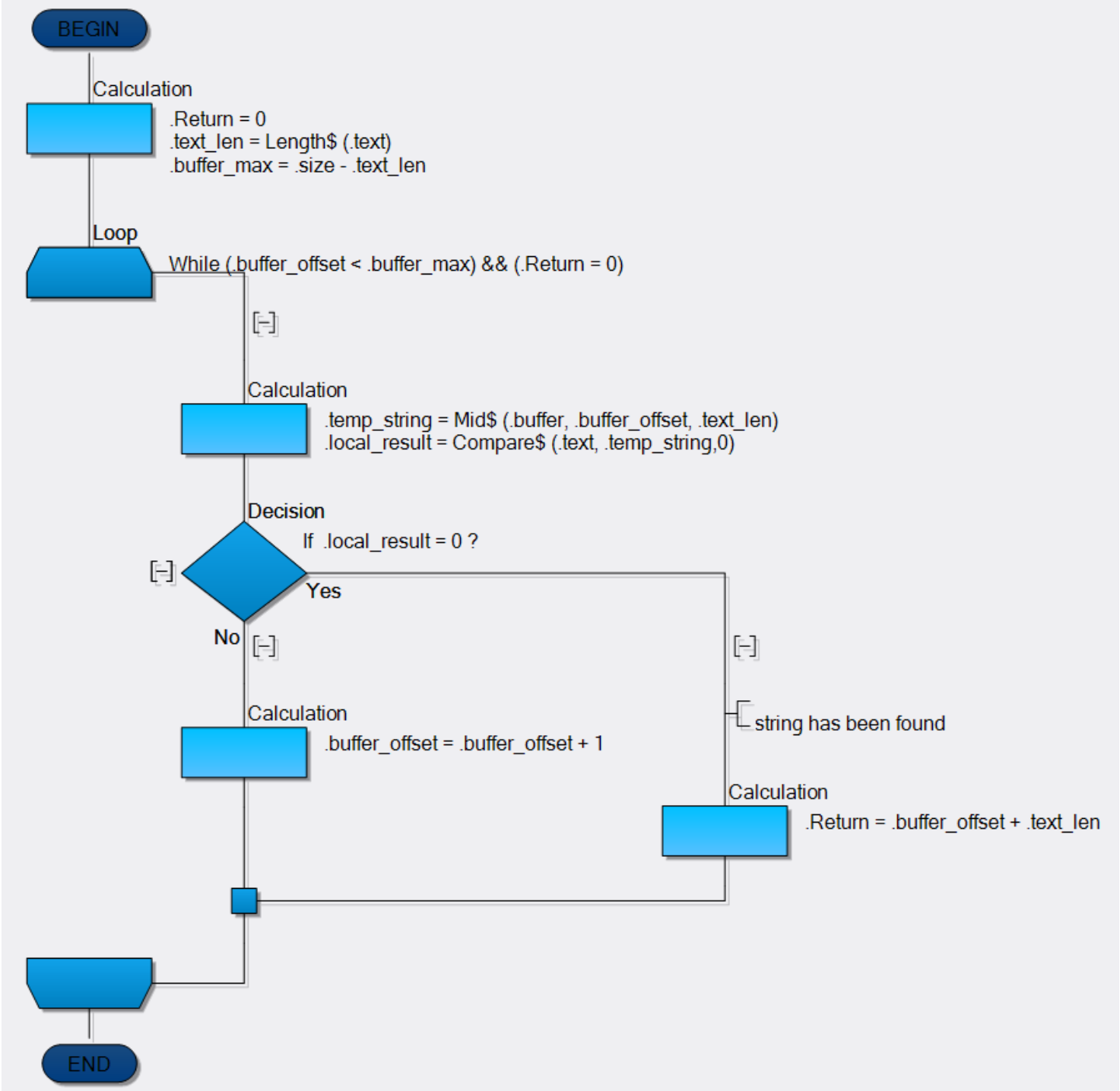
The next stage is to create some Flowcode functions that will enable us to read the temperature data from the received XML text. We could parse the XML tags and retrieve much data, but for this initial demonstration we will simply look for the word “Temperature” within the buffer text.



Here we create a new macro called “InString” that takes several parameters:

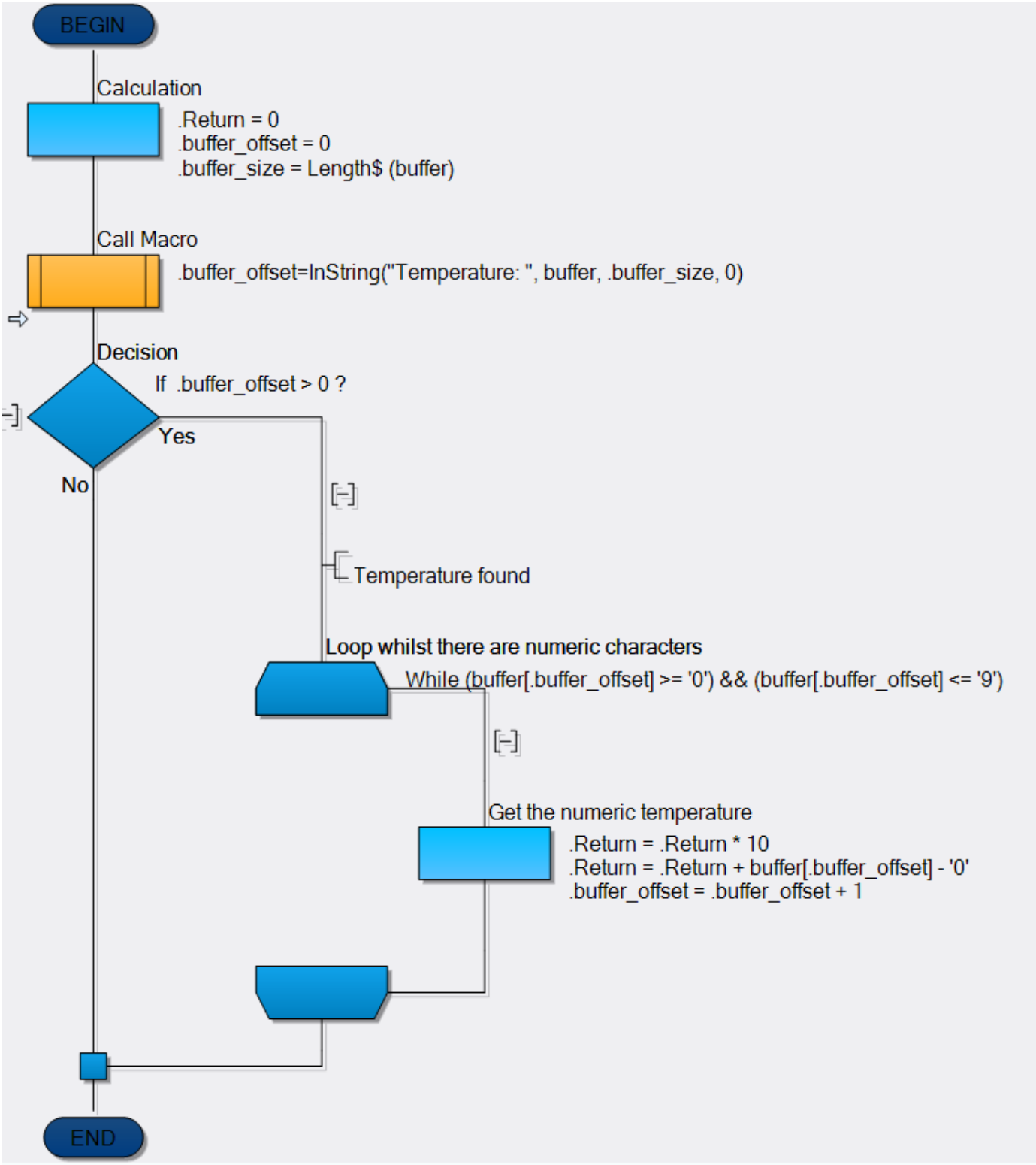
1) “text” which is the text string we are searching for,
2) “buffer” which is the text buffer in which we are searching,
3) “size” which is the total character count size of the buffer,
4) “buffer_offset” which is the starting index within the buffer from which point the search is done. This is such that multiple searches can be performed.

The macro returns the index offset of the next character following the end of the searched text string, if it is found, otherwise it returns zero.



Above is the flowchart for the “InString” macro, which has some local variables:

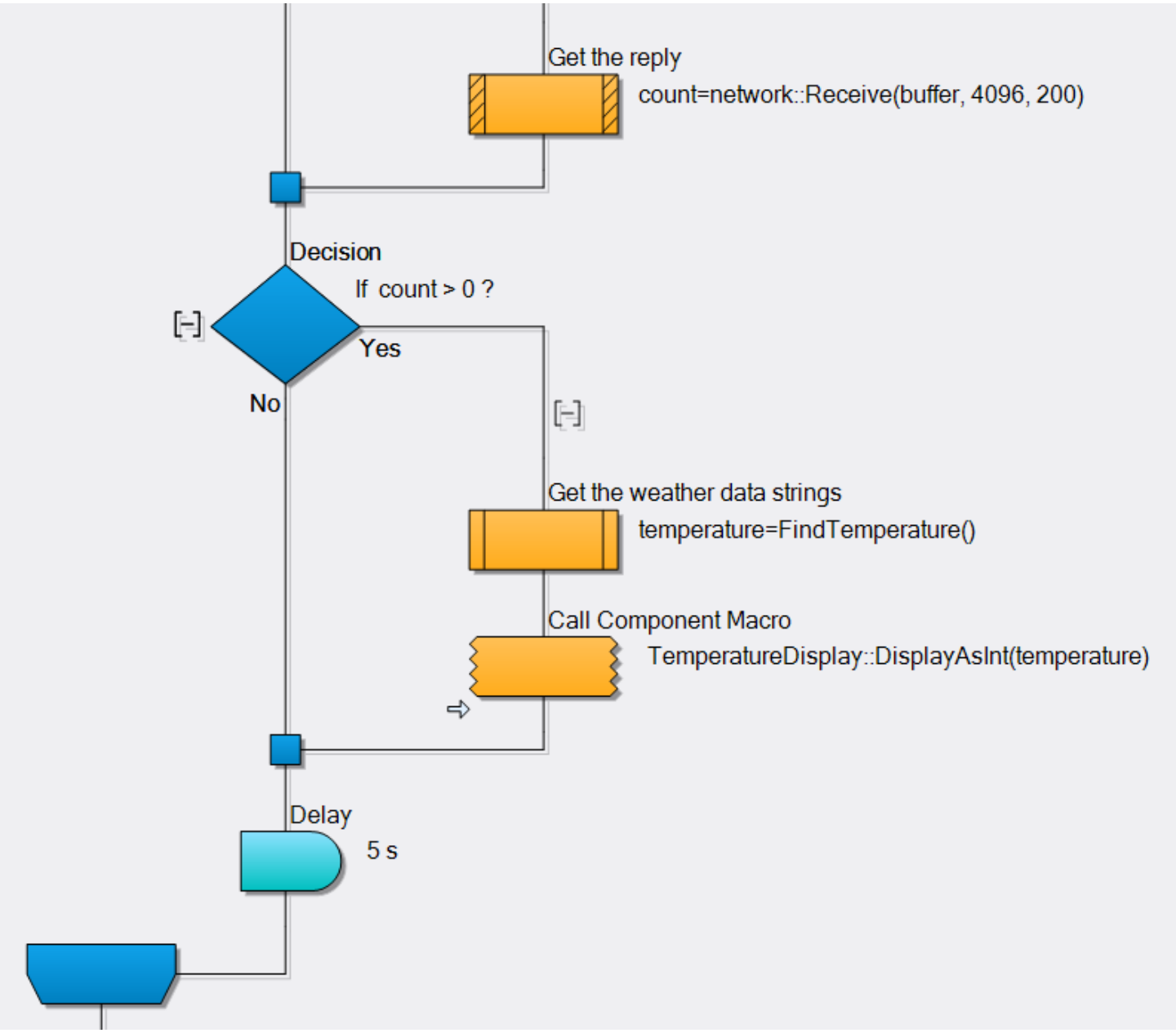
Byte local_result
String temp_string
UInt buffer_max
UInt text_len



We will also create a new macro called “FindTemperature” that searches the global text buffer for the word “Temperature:” and, if found, return the numeric value that follows, as an integer. It has some local variables:

UInt buffer_size
UInt buffer_offset

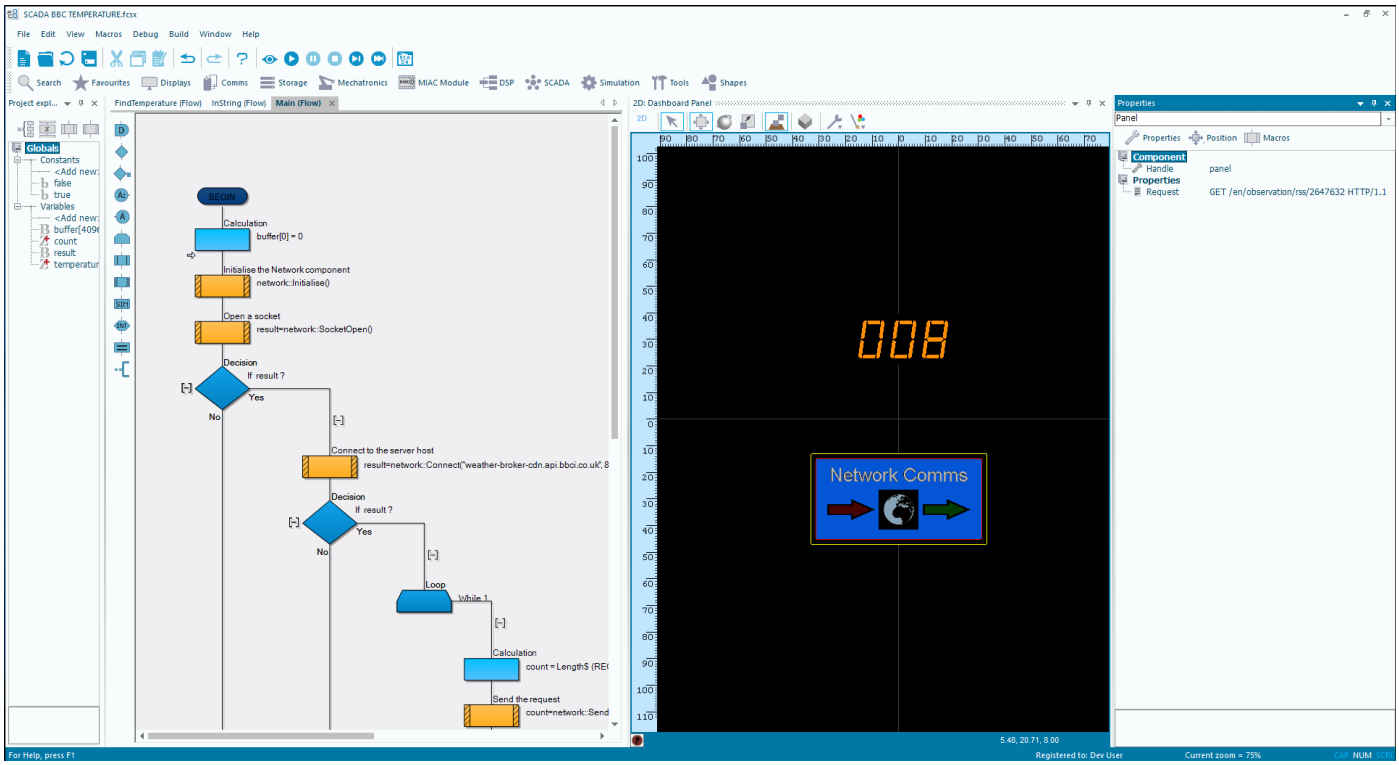
Now add the “FindTemperature” macro function into our main loop flowchart and display the result onto the temperature display on the panel. The image below shows the new “Decision” icon, associated macro calls, and “Delay” icon that we have added to the “Main” flowchart at the bottom of our infinite loop.



The program can now be run and tested. The Console display can be used to view the GET request and the server reply. The project is now ready to be deployed as a Matrix SCADA runtime application.

Deploying a Matrix SCADA runtime application

This walkthrough assumes that you have followed the previous section, and created a simple application that reads the temperature from the BBC Weather RSS feed. It also assumes that the application has been tested, and verified to run correctly. It will also assume that you added the two components to the 2D Dashboard Panel and your view of Flowcode is something similar to below.

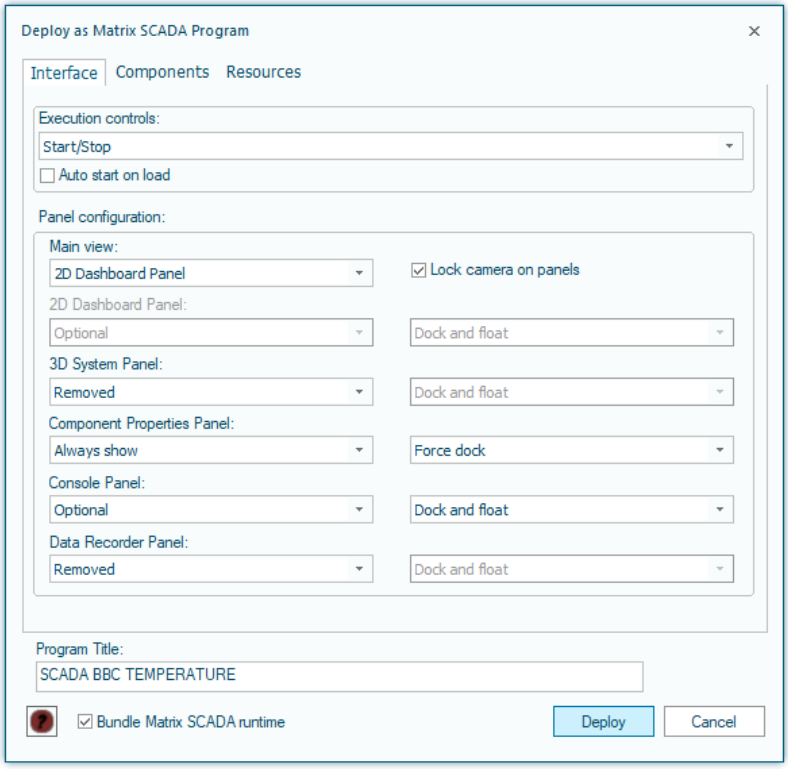


When a user accesses your application, it could be on any machine with any number of network adapters. To allow them to configure the correct network interface when they are running the application: we will grant them access to the “Network Interface” property of the “Network Communications” component.

In the “Component Properties” pane, select “network” from the drop-down menu at the top (recall that this is the name we gave our instance of the “Network Communications” component). Now, select the drop down arrow to the left of “Network Interface”, and click “Expose to top level”. That’s all that we need to do to grant the user access to this property.

It is now time to deploy your Flowcode SCADA application as a Matrix SCADA runtime application. Select the button on the toolbar indicated below.

You will be greeted with the following deployment configuration screen.

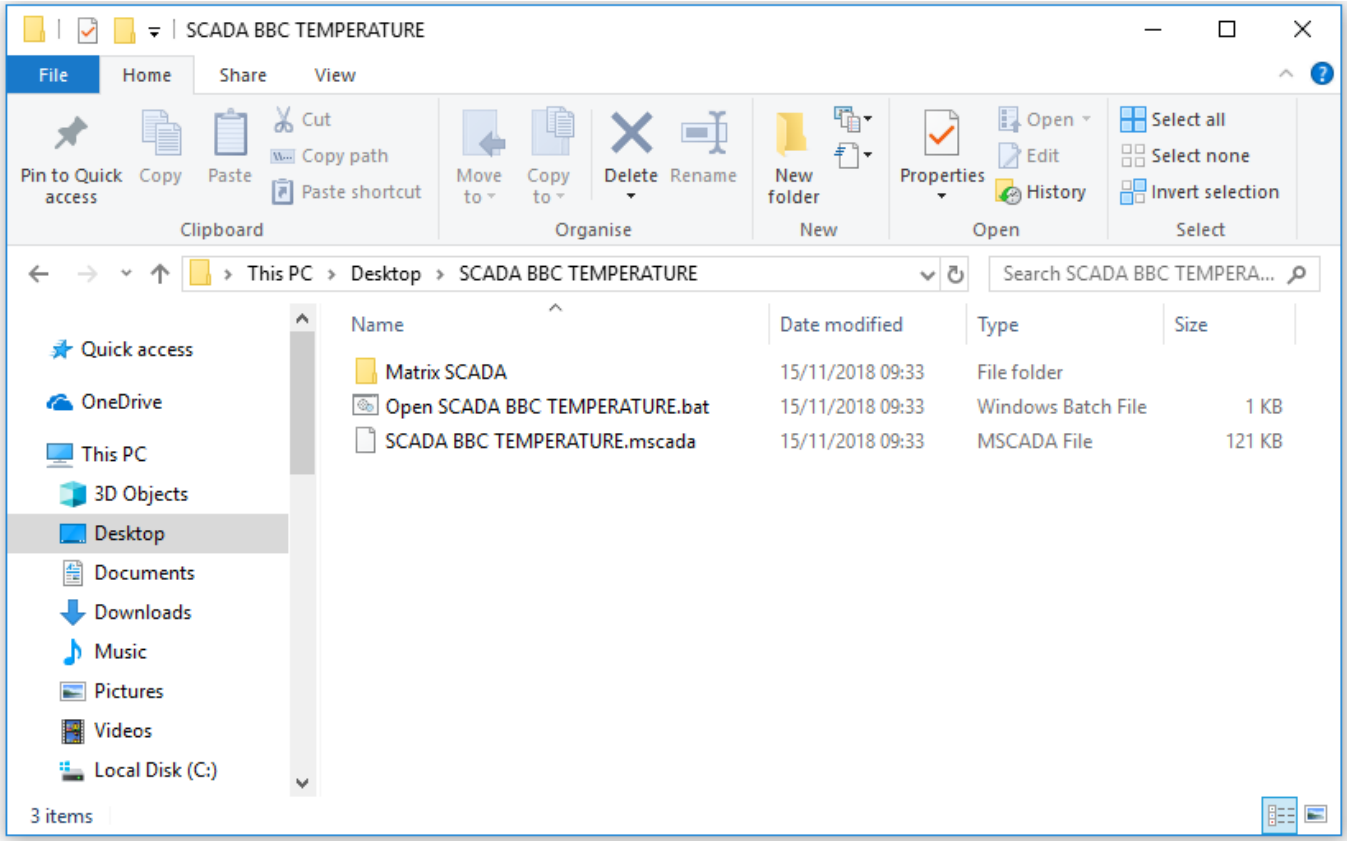


In this guide, we have opted to only give the user access to the Start/Stop control buttons. We have also made the 2D Dashboard panel the “main” view of the application and locked the camera into position. We have also removed the 3D System and Data Recorder panels, as these are not needed in our program. We have made it so that the Component Properties panel is always visible and always docked. We have made the Console panel optional and we allow the user to dock or float it freely.

All necessary components and dll files required for communications will be automatically bundled with the program. Nothing for you to do!

Finally, give your program an official name, select “Bundle Matrix SCADA runtime” and click “Deploy”. You will be asked to select a location for your program. Flowcode will then create your Matrix SCADA runtime application and tell you where it can be located.

In this example, we gave our Matrix SCADA runtime application the name “SCADA BBC TEMPERATURE” and opted to place the final program on the desktop. Flowcode has created a folder called “SCADA BBC TEMPERATURE” on the desktop, and included all necessary files in that folder. We show its contents below.



This folder is entirely portable, and can be given to anyone to run on their Windows-based computer. The user does not require Flowcode, any licenses, or any installed dependencies. In order to run the program, the user simply needs to double click “Open SCADA BBC TEMPERATURE.bat”. Doing so will automatically load the bundled copy of the Matrix SCADA runtime (shown above in its own folder) and load the program.

The above image shows the result of clicking the .bat file.
Just as we configured in the deployment options in Flowcode, the 2D Dashboard is the main view, and the camera cannot be moved. The Component Properties panel is forcibly docked and the Console is also optional. No other panels are available. The user has access only to “Start” or “Stop”.

Access to the code is not possible and the program cannot be modified. The program file itself is heavily encrypted and cannot be modified.

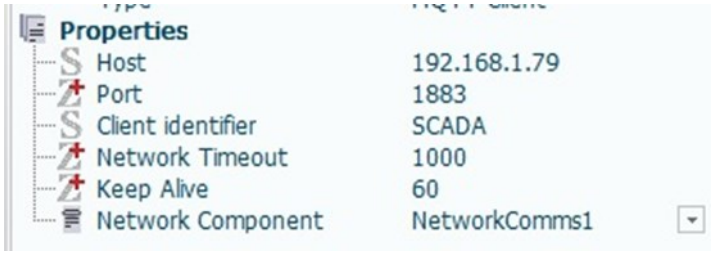
Note that the “Network Interfaces” option can be configured by the user in the Component Properties panel, as we chose to expose this to the user prior to deployment.

This was just a simple example of deployment. In more complex situations, you may choose to hide certain components or embed additional resources into the program.

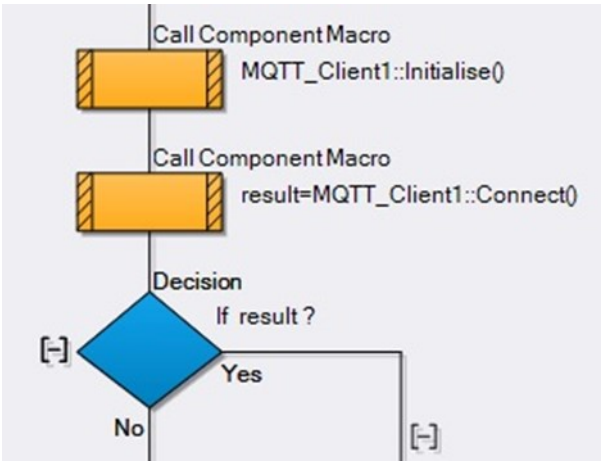
Using Communication Components

The MQTT communication component.

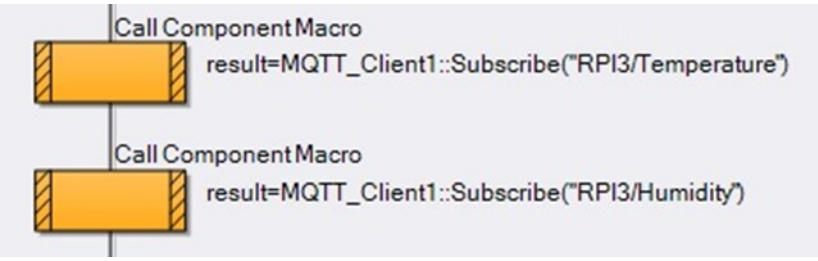
MQTT is a machine-to-machine IoT connectivity protocol that uses a broker as intermediary between clients that publish messages, and clients that subscribe to messages. The Flowcode MQTT component greatly simplifies use of this protocol and has the following properties.



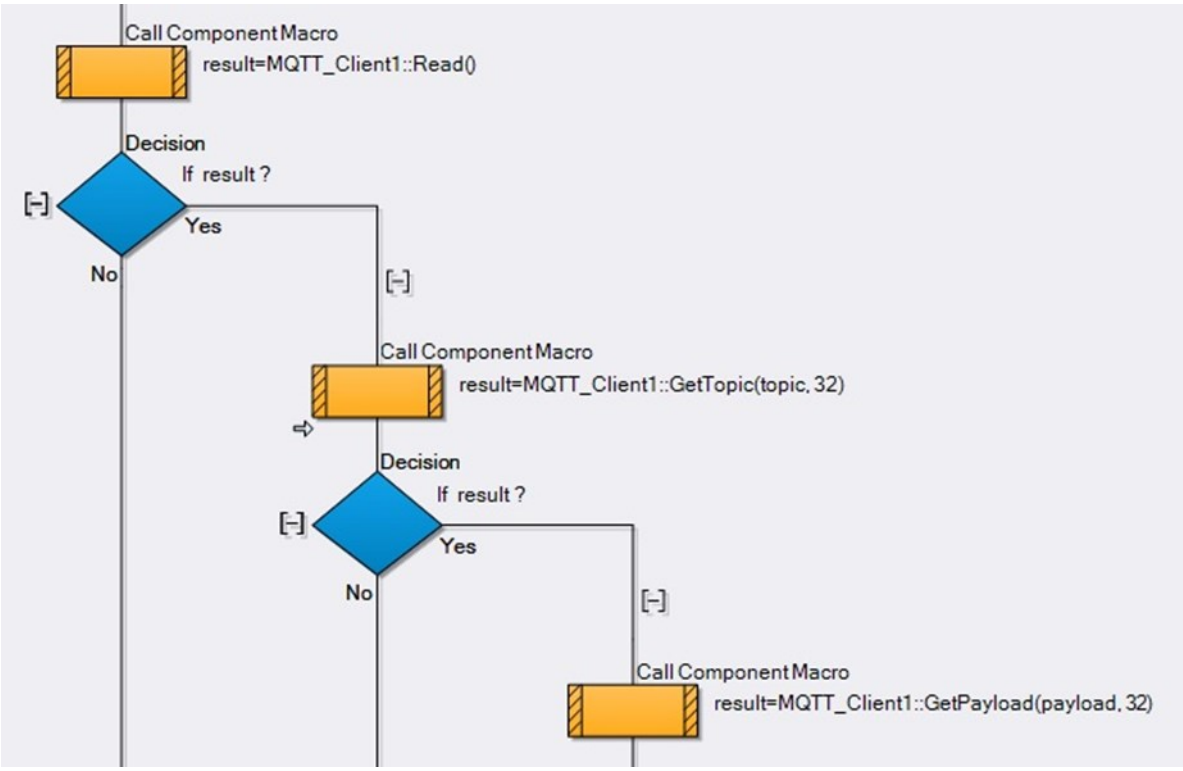
Here we see the setting for the MQTT broker host address, such as a Mosquitto broker. The Client Identifier can be set to any name that identifies us to the broker. The Keep Alive time is set here to 60 seconds, and this is the maximum time between our communications with the broker before it will assume communication loss and disconnect. So we need to either request information from the broker at a rate faster than this time period, or to send a Ping to keep the connection alive. The MQTT component connects with and uses a NetworkComms component. Any project begins by first initialising the MQTT component and attempting connection with the broker, as shown below.



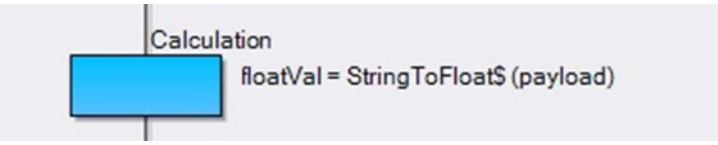
If the return result from the “Connect” request is true (non-zero) then this indicates that the connection is successful and we can continue. The next step is to subscribe to a topic (information stream) on the broker.



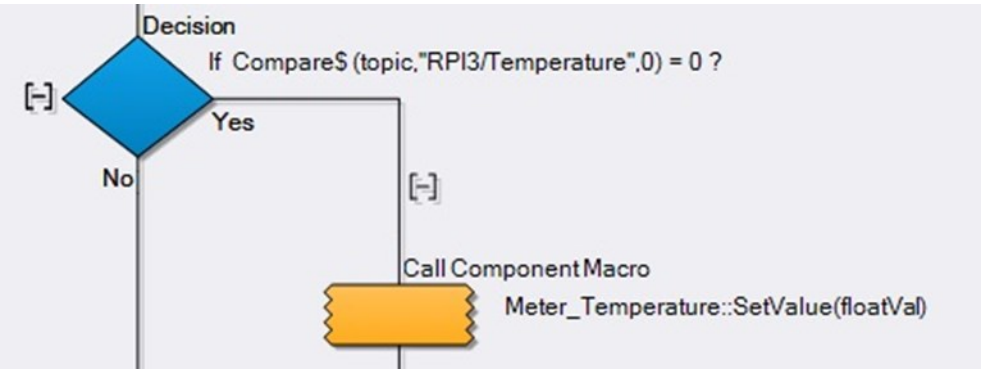
Here is an example of how we subscribe to information on the broker that is being published by another client, for example a remote Raspberry Pi-based environmental sensor.



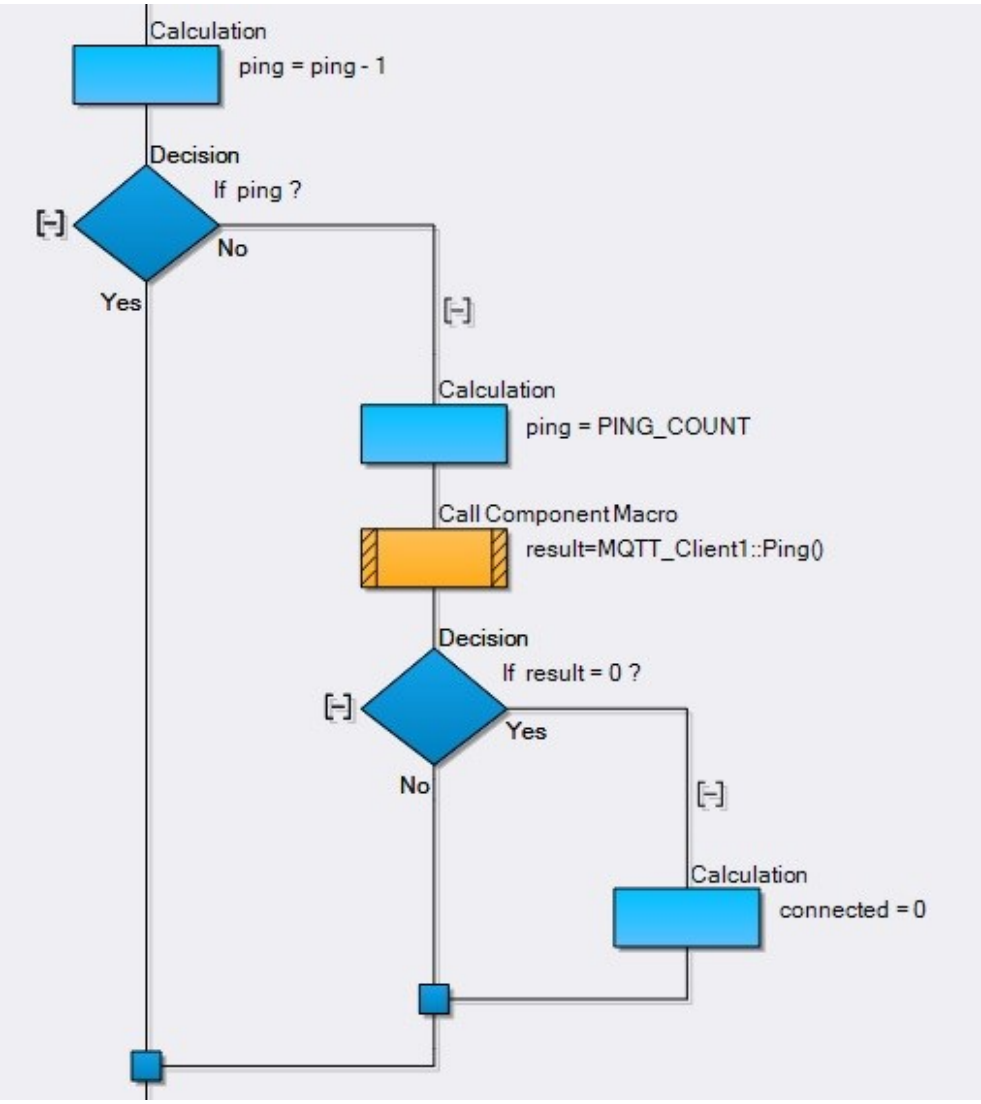
We can now enter a periodic loop where we check for any available data from the broker. We can do this with the MQTT component's "Read" macro. This returns true (non-zero) if a new message has been received from the broker.
The MQTT component's "GetTopic" macro gets the topic name from the incoming message and copies it into the string variable "topic", with the macro returning true (non-zero) if it successfully parsed the topic from the message. In the above example, if this is the case, then we go on to get the value data from the message that is contained in the payload of the message, using the "GetPayload" macro. This populates the "payload" string with the value for this topic.



We subscribed to two topics so this message could be either. We retrieve the value as a floating point number to pass to a panel meter.



We check the topic name, using the string compare function and update the appropriate panel meter.



For a number of reasons connection can be lost with the broker and our reading loop is passive, so we need to keep the connection alive using the Ping feature.

So in our reading loop, that includes a delay, we keep a loop count. Once the loop count expires we send a Ping to the broker to maintain the connection. This also enables us to test if the connection is still established.

Interfacing industrial sensors

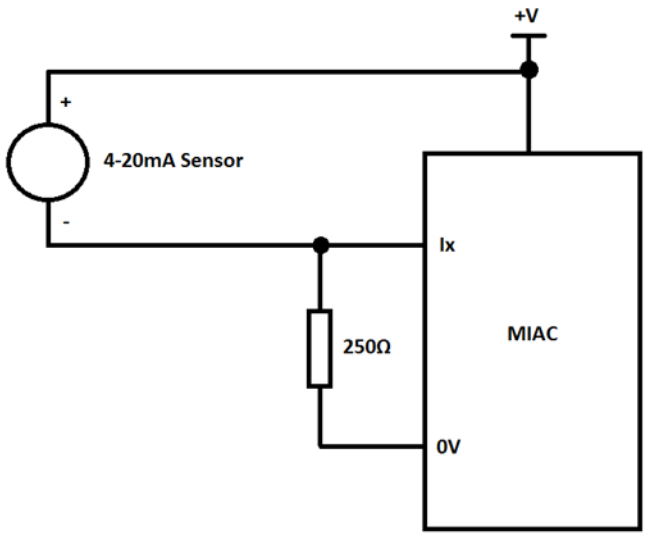
The Flowcode communication components allow a wide range of sensors and control modules to be connected to a SCADA application.

Sensors can provide information to a SCADA application via protocol-based commercial modules or embedded programmable devices, such as the Matrix MIAC range.

For example the Flowcode Modbus component provides access to a wide range of industry standard devices.

Many applications can be created using simple serial communication via commands sent over a hardware or virtual communication port for which Flowcode provides UART based components such as RS232 and RS485

Many sensors provide output information either in the form of a voltage or a current source proportional to the sensor value. Current sources are usually of the 4-20mA variety. These sensors can provide information to a SCADA application via protocol-based commercial modules or embedded programmable devices with Analogue to Digital convertors, such as the Matrix MIAC range. For example, an embedded application running on the MIAC could simply return the integer value from the input ADC when requested by the SCADA application, via commands sent over the virtual communication port. Calculations can then be done to convert the raw ADC value into a voltage, or directly to temperature or pressure based on the value range of the sending device.



This circuit demonstrates how a 2-wire current loop device can be connected to an ADC input, such as that of the MIAC.

The 250Ω resistor will result in a maximum voltage of 5V at 20mA.

This will give a maximum ADC value of 400 when reading in 10bit mode on a MIAC.

For 3-wire devices that need to be powered at 5V maximum then the resistor needs to be of lower value, such that the full range 20mA induced voltage is lower than 5V.

Matrix Technology Solutions Ltd
33 Gibbet Street
Halifax
HX1 5BA

t: +44 (0)1422 252380
e: sales@matrixtsl.com
www.matrixtsl.com