Simulated Cutting System

Creating modern interfaces to industrial systems

Technical specifications

February 2013

REVISION HISTORY

Revision	*Date*	*Author*
1.0.0 - DRAFT	7 February 2013	Claudio Vivante
1.0.1 – DRAFT	8 February 2013	Claudio Vivante
1.0.2	10 February 2013	Claudio Vivante

DEFINITIONS

This documents contains the technical specifications for the Simulated Cutting System (SCS) interface. The SCS project is composed by a simulator, also called *server* which simulated a simple production plant and published real-time data to connected clients, through the SCCT (Smartphone and Cross-platform Communication Toolkit) communication protocol. Clients (also called subscribers) receive different types real-time data: analog signals, digital signals, text messages with different rates. When a client connects to the SCS it exchange data with the server, data elements are called *data packages* or briefly *packages*. The time a client stays connected to the server is defined *session*.

GOAL

Purpose of the project is to use the SCS as useful case study for the development of *smart* interfaces at client side. SCS server can modulate the data throughput allowing interface designers to measure the reliability of their solutions and meter resource consumed by their interfaces. Clients can be implemented as simple observers of SCS status, it means they do not send data to the server. To provide a better experience to client's users, SCS allows clients to simulate some actions on the production plant. On real industrial machineries is not possible to switch some parts ON and OFF for a remote user, due to violation of basic safety rules. Anyway, this case study is focused to the creation of smart interfaces at client side and is a useful exercise for high school students and passionate programmers who want to design their dynamic multiuser environment. For this reason, some phenomena are simulated much faster than real systems do. [1]

CUTTING SYSTEM DESCRIPTION

SCS simulate the following production plant:

SCS is a plant composed by these units

- Motor controls the speed production is done: higher speed means more products/unit of time.
- Multiple-blade cutter, driven by the motor
- Injection circuit, to pump hot fluid
- Cooling system to control the plant temperature.
- Hardware controller with some modules for I/O purposes.

An industrial controller with some modules for I/O tasks

The plant produces different objects from a fluid hot plastic, the plastic is warmed by a specific unit. Fluid temperature and pressure are monitored. Fluid temperature and pressure cannot exceed their upper values otherwise pressure and temperature alarms are fired. The electrical motor drives a cutter, used to shape different products. The system can produce different products, generally indicated with name *product type 1*, *product type 2*, etc. Fluid is injected into moulds of different types and the cutter shapes the products. For the purposes of the case study, it' not very

[1] For example the system temperature changes dramatically and the cooling system, when activated, is capable to cold the plant better any real cooling machine!

important that such production system exists. Real systems are probably more complex, but for this case study we need a simple model capable to generate fast data with a certain degree of coherence. The plant works as follow:

System power indicator must be ON or OFF. If power is OFF nothing can be done.

Motor state can be in four states: **OFF** , **accelerating**, **decelerating**, **Ready**. Production can be done only if motor state is ready which means it has reached its target speed. When motor is changing its speed to reach the new target speed, production is not possible.

Cutter can be **armed** or **disarmed**. Production requires that cutter is armed.

Injection circuit, which let the how fluid go into the moulds, can be **open** or **closed**. Fuild flows when circuit is open. If circuit is closed, production cannot be done.

Controller's modules can be removed occasionally to do some maintenance to the plant. When one or more of them are removed from the controller, the production is stopped immediately. An alarm fires if a module is not installed.

Production can be done if the following conditions are true:

- Power is **ON**
- Motor is **ready** (motor speed greater than zero)
- cutter is **armed**.
- Injection circuit is **open**
- **Hardware module are installed properly**

If not all units indicated above are in the proper state, production doesn't start or, if started, stops immediately.

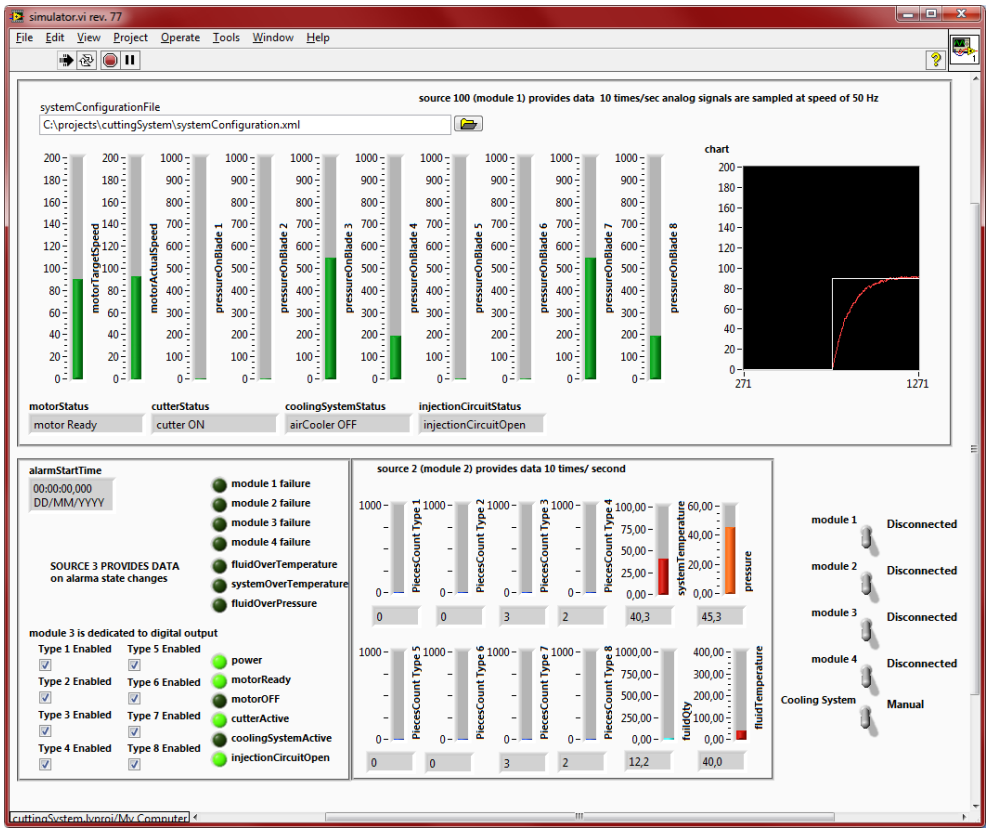
Described units when running, contribute to increase the system temperature[2], which is monitored with a thermocouple. When temperature rises up to a upper limit, a block alarm fires and the production is stopped. To prevent that alarm, a cooling system (a fan) is available. Cooling system can be controlled automatically or manually. When cooling system is in automatic mode, it switches on and off according to system temperature. When production is active, product counters increase as soon as a new piece has been completed. Counters cannot be reset, they start from zero when SCS application is started. Products can be made when fluid temperature is in a range RP different for every product type; the time required to complete a piece is calculated as

KP * motor speed

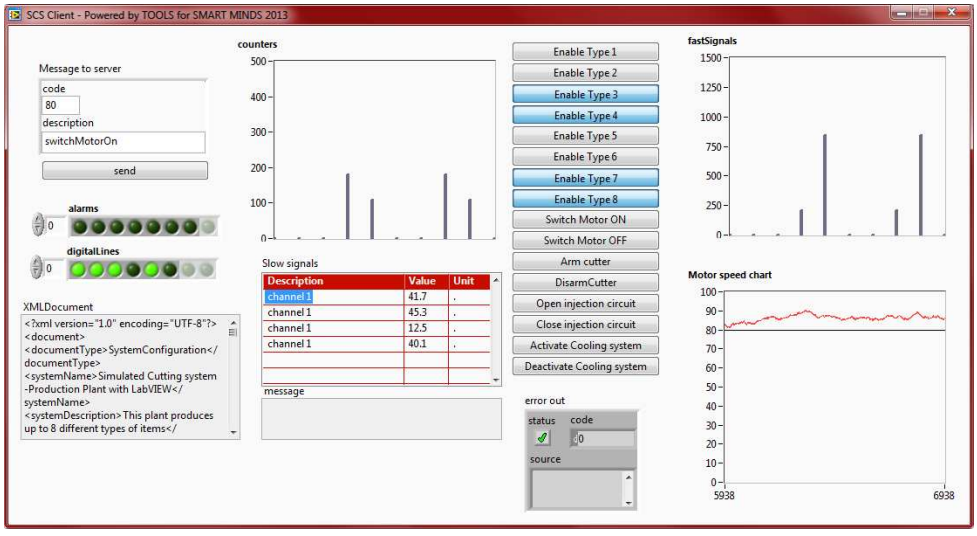
where KP is a constant that changes for every product type.

During the fluid injection, a pressure signal changes quickly up to a product specific value and quickly goes back to its lower value.

[2] Energy balance is calculated according to equations not indicated in this document.



SCS server main window



SCS client created with LabVIEW

SYSTEM CONFIGURATION

When a client connect to the server, it receives an XML document that describes the SCS systems. This document must be used by the client to dynamically build the interface. No other information are provided to clients during the session: SCS configuration cannot change if one or more sessions are active. The XML document contains the following TAGS:

- **documentType** specify the type of XML document. Its value is set to *SystemConfiguration*. Clients have to check that **documentType** is equal to *SystemConfiguration*. Future extension of SCS can include other XML documents transmitted in the middle of the session.
- **systemName** indicates the server name
- **systemDescription** contains a brief description of the server. It can be empty value.
- **availableProductTypes** indicates the number of types of products SCS can make
- **hardwareDescription** contains information about the HW devices used by SCS to acquire and generate signals. It include the following TAGS:
  - **controller contains details about the type of HW controller used to manage SCS. It includes the following TAG:**
    - **model indicates the name of the controller**
  - **installedModules indicates the number of modules attached to the controller.**
  - **modules contains information about the modules (i.e. physical devices) used to acquired** and control signals. Every module contains the following tags:
    - **name**
    - **slot**
- **sourceCount** indicates the number of sources used by the server
- **sources** contains information about the data published with the source tags. Every source contains the following tags
  - **id**
  - **description**
  - **digitalDataCount** indicates the number of digital lines published by this source
  - **digitalData** contains information about each digital line. Every digital line contains the following tags:
    - **name**
    - **description**
  - **analogDataCount** indicates the number of analog channels published by this source
  - **analogData** contains information about each analog signal. Every analog signal contains the following tags:
    - **name**
    - **description**
    - **minValue**
    - **maxValue**
    - **units**
    - **samplingRate**

SOURCES[3]

SCS publishes real-time data through SCCT communication libraries (in the follow SCCT only) grouped into different sources. Data sources are logical groups of information of different type with a kind of relation among them. Every data package received by a client includes a tag composed by a numeric field, the SOURCE ID and a text field called the SOURCE DESCRIPTION. Sources refers to logical units of the production plant or to a specific acquisition subsystem, typically an acquisition device or measurement module.

SCS uses the following sources:

SOURCE ID 1 [4]

This source is used for alarm indicated by Boolean values. These packages are sent when an alarm changes its state. Values = false mean NO ALARM, values = true mean ALARM.

SOURCE ID 2

This source is used for messages and digital values. These packages are sent to notify some events, for instance, if a component of SCS changes its state, the whole SCS state is notified.

SOURCE ID 3

This source is used for enable states of available product types.

SOURCE ID 100

This source is used for motor speed data acquired at sampling rate of 50 Hz[5]. The signals are published 10 times per seconds so every package contains 5 samples per channel. Sampling rate cannot change during SCS, it means that when a client connects and receives details about signals into source 1, these features don't change during the session.

SOURCE ID 101

This source is used for analog data streams acquired at sampling rate of 50 Hz[6]. The signals are published 10 times per seconds so every package contains 5 samples per channel. The channel's details are provided to clients with a XML document when they connect to the server. Sampling rate cannot change during SCS, it means that when a client connects and receives details about signals into source 1, these features don't change during the session.

SOURCE ID 200

This source is used to publish slow rate signals. These analog values are sampled 10 times per seconds.

SOURCE ID 201

This source is used to publish counters of available product types. Every counter is the number of pieces made since SCS server started. These analog values are sampled 10 times per seconds.

[3] Clients must connect to SCS server with the option Filter Enabled = False.

[4] Sources 1…99 are related to events and are not notified with a specific refresh rate.

[5] The sampling rate can be changed at SCS, clients must be able to adapt to changes of signal sampling rate.

[6] The sampling rate can be changed at SCS, clients must be able to adapt to changes of signal sampling rate.

CONNECTING TO THE SIMULATOR

To connect to the server, a client has to follow the sequence indicated below:

OPENING THE CONNECTION

With the SCCT command **OpenConnection**(serverAddress, serverPort, serverAPI-Key, timeout, clientName, sourceFilter=False)[7] a client request a connection to the server.

The parameters are described in the follow:

- ServerAddress is an IP4 address in the form xxx.xxx.xxx.xxx or the corresponding DNS name.
- serverPort is the TCP port number used by the server.
- serverAPI-Key is the password required to be successfully authenticated by the server.
- Timeout is the time in seconds, used by SCCT server and client to manage the keep-alive messages in their background tasks. Minimum value is 1 second, suggested value for Internet connections is 10 seconds
- clientName is the name of the user. If not specified, SCCT uses client's IP address+ TCP port as client name, to identify the connection.
- SourceFilter is an advanced feature of SCCT communication library to filter data at server side and save communication band when connecting to multiple sources server. In this case, client DON'T HAVE to restrict the number of sources, so this parameter must be set to FALSE.

If connection fails, client can retrieve the *failure reason*: a text message that describe why connection has not been established.

RECEIVING SYSTEM CONFIGURATION AND SETTING UP USER INTERFACE

If connection is established, the server transmits to the client the following packages:

an **XMLData** package called *systemConfiguration* with a complete description of digital and analog signals.

A **digitalData** package with the alarms on the simulated machine

A **DigitalData** package with the digital Line states (Motor ON/OFF, cutter ARMED/DISARMED, etc.)

An **DigitalData** package with the actual states of EnabledTypes checks, indicating which product types are currently enabled for production.

At this point, client can build the its user interface according to the data received and can setup all required graphical objects. Server doesn't transmit other information.

RECEIVING REAL-TIME DATA

When client is ready to receive real-time data, sends to the server a request to start real-time transmission with the SCCT function **play.** If client is not able to process all data coming from the server, it can use SCCT function **pause** to ask to server to stop data transmission. Refer to SCCT user guide for more details.

[7] The syntax may change according to the language. The meaning of the parameters is the same among all edition of SCCT libraries.

AVAILABLE COMMANDS AT CLIENT SIDE

These actions or client requests are sent to the server with the message package. Messages are composed by two elements:

- **Code** is a numeric (integer I32) value
- **Message** is a text value (not empty)

In the following table are indicated the action recognized by the server.

Message value	***Code value***	***description***
switchMotorOn	Target speed	Switches the motor on. Target speed (in RPM) must be indicated in code value. Valid speed range is indicated In system configuration XML document. Requested with invalid range are discarded.
SwitchMotorOff	Not used	Switches motor off
switchCoolingSystemOn	Not used	Switches the cooling system on
SwitchCoolingSystemOff	Not used	Switches the cooling system off
armCutter	Not used	Arms the cutter
disarmCutter	Not used	Disarms the cutter
openInjectionCircuit	Not used	Opens the injection circuit
closeInjectionCircuit	Not used	Closes the injection circuit
enableType	Index	enables the production of a specific type of products (1 based index)
disableType	Index	disables the production of a specific type of products (1 based index)
powerOn	Not used	Switches ON the power plant (no effect if power is ON yet)
powerOff	Not used	Switches OFF the power of the plant (No effect if power is Off yet)

CREDENTIALS

SCS online uses with the following credentials:

Port	8089
API-Key	SCS
Address	t4sm.blogdns.com

When used in *localhost* SCS refers to credentials indicated into setup.ini file.

Clients created with LabVIEW use credentials and server address indicated in setup.ini file located in the same folder of client.exe

RESOURCES

The following resources are available to developers

SERVER

SCS is available online at the following address

SCS stand-alone application is available for free download at

http://www.toolsforsmartminds.com/case-studies/SCS/simulator.zip

SCS server with Installer and LabVIEW Runtime Engine is available for free download at

http://www.toolsforsmartminds.com/case-studies/SCS/SCS-simulator+installer.zip

CLIENTS

Clients are available at the following links:

http://www.toolsforsmartminds.com/case-studies/SCS/client.zip

SCS client with Installer and LabVIEW Runtime Engine is available for free download at

http://www.toolsforsmartminds.com/case-studies/SCS/SCS-client+installer.zip

SCCT LIBRARIES

SCCT libraries for different platforms and development languages are available at

http://www.toolsforsmartminds.com/products/SCCT.php

an introduction to SCCT libraries is available in the SCCT White paper at the following link:

http://www.toolsforsmartminds.com/pdf/White%20paper%20SCCT.pdf

OTHER RESOURCES

3D drawings of Industrial controllers are available at the following links:

http://digital.ni.com/hardref.nsf/websearch/F0E51BDF84664F7386257A1D00536E08?OpenDocument&node=200533_US

http://digital.ni.com/hardref.nsf/websearch/129593ca1c70b78c862572f8004ec502

http://digital.ni.com/hardref.nsf/websearch/16d0d2454828bb71862572f4004f23b5

CONTACTS

To get more details, contact info@toolsforsmartminds.com

INDEX