

FeSCADA – using Joysticks and Bar Code Readers

Introduction

Human Interface Devices (HIDs) are computer input devices that allow users to have access to raw input data, such as keyboards, mice, *joysticks*, *gamepads*, *bar code readers*, etc. On Microsoft Windows PCs, the raw input API allows applications to receive raw input from any HID.

In this paper we present applications with FeSCADA using bar code readers, joysticks and gamepad controllers.

1. Description
2. Hardware
3. WM_INPUT Windows messages and RAWDATA
4. USB hardware serial ID. GetRawInputDeviceInfo() - the complete ID string
5. Bar code reader application
6. Joystick application
7. Gamepad application
8. Conclusions

1) Description

In the following pages three applications are developed using HID input devices in FeSCADA: a bar code reader, a joystick, and a gamepad. These are off the shelf products that can be found in many computer stores or on the Internet. They are important because they are affordable and easily replaceable.

The bar code reader is working like a keyboard, generating an array of characters in a fast and repeatable way. What is the purpose, in automation, of having the input of a 12-digit number from a bar code reader? One can connect the string from a bar code reader with a database. Based on the string ID, one can read very fast from the database different data like a recipe, to load a new set of parameters on a machine, or the first empty location in an automated storage and retrieval system (ASRS).

The joystick (gamepad) is an input device that can give access to 8-12 digital inputs (buttons) and 2-4 analog inputs. The purpose of using it is to easily replace a whole panel with buttons.

Any instance of FeSCADA can have access to one joystick and 2 bar code readers. FeSCADA can run only in one instance from a given folder. But another copy of FeSCADA, in another folder, can run another instance. In this way one can run multiple FeSCADA projects on the same computer, if they are in different folders.

That means that one can use multiple bar code readers and joysticks on the same computer, limited only by the available USB ports.

2) Hardware

The hardware in our applications is composed of:

- Logitech X30 joystick with 12 digital buttons, an 8 way hat switch, and 4 analog inputs: X, Y, tilt and speed.
- Marvo GT-006 gamepad controller, with 12 buttons and an 8 way hat switch.
- NADAMOO model 3094 bar code reader.



Joystick Logitech X30



Gamepad Marvo GT-006



Bar code reader NADAMOO – model 3094

Hardware prices.

Name	Unit price	Qty	Price	Description
Logitech X30	\$35.00	1	\$35.00	Joystick
Marvo GT-006	\$20.00	1	\$20.00	Gamepad controller
NADAMOO - 3094	\$30.00	2	\$60.00	Bar code reader
TOTAL =			\$115.00	

3) WM_INPUT Windows messages and RAWDATA

The raw input model is different from the original Windows input model for the keyboard and mouse. In the original input model, an application receives device-independent input in the form of messages that are sent or posted to its windows, such as WM_CHAR, WM_MOUSE_MOVE, and WM_APPCOMMAND. In contrast, for raw input an application must register the devices it wants to get data from. Also, the application gets the raw input through the WM_INPUT message.

There are several advantages to the raw input model:

- An application does not have to detect or open the input device.
- An application gets the data directly from the device, and processes the data for its needs.
- An application can distinguish the source of the input even if it is from the same type of device. For example, two exactly the same mouse devices on two different USB inputs.
- An application manages the data traffic by specifying data from a collection of devices or only specific device types.
- HID devices can be used as they become available in the marketplace, without waiting for new message types or an updated OS to have new commands in WM_APPCOMMAND.

By default, no application receives raw input. To receive raw input from a device, an application must register the device.

Note that an application can register a device that is not currently attached to the system. When this device is attached, the Windows Manager will automatically send the raw input to the application.

An application receives raw input from any HID whose top level collection (TLC) matches a TLC from the registration. When an application receives raw input, its message queue gets a WM_INPUT message and the queue status flag QS_RAWINPUT is set.

An application can receive data when it is in the foreground and/or when it is in the background position.

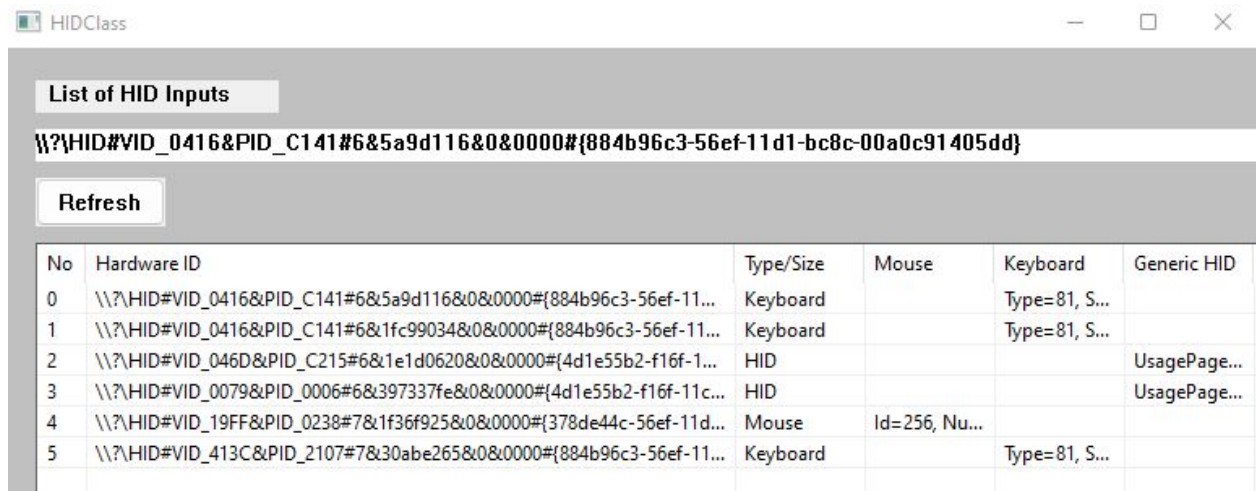
To interpret the raw input, detailed information about the HID is required. For example, for joysticks and gamepads the RAWDATA structure starts with a byte that specifies the size of data bytes, followed by the data bytes. The user has to map all the bits to the right buttons or analog inputs. For the bar code reader the RAWDATA is an array of characters that is ending with CR (carriage return) and/or LF (line feed).

4) USB hardware serial ID. GetRawInputDeviceInfo() - the complete ID string.

Every HIDClass device has a unique hardware serial ID. If the same HID device is plugged in another USB port it will have a different serial ID.

This hardware serial ID is important if we want to use a specific joystick in a specific USB port, and not to use the same joystick if it is plugged in a different USB port.

To find the complete hardware serial ID of your HID device one can use the program FeMODBUS. Go to 'Utils→HIDs' menu. The program will use the function GetRawInputDeviceInfo(...) to retrieve the complete serial ID of all the HID devices connected to the computer. See figure below.



The screenshot shows a window titled 'HIDClass' with a 'List of HID Inputs' section. Below this, there is a text field containing a long hexadecimal string representing a hardware ID. A 'Refresh' button is located below the text field. Below the button is a table with 6 columns: No, Hardware ID, Type/Size, Mouse, Keyboard, and Generic HID. The table lists 6 HID devices.

No	Hardware ID	Type/Size	Mouse	Keyboard	Generic HID
0	\\?\HID#VID_0416&PID_C141#6&5a9d116&0&0000#{884b96c3-56ef-11...	Keyboard		Type=81, S...	
1	\\?\HID#VID_0416&PID_C141#6&1fc99034&0&0000#{884b96c3-56ef-11...	Keyboard		Type=81, S...	
2	\\?\HID#VID_046D&PID_C215#6&1e1d0620&0&0000#{4d1e55b2-f16f-1...	HID			UsagePage...
3	\\?\HID#VID_0079&PID_0006#6&397337fe&0&0000#{4d1e55b2-f16f-1...	HID			UsagePage...
4	\\?\HID#VID_19FF&PID_0238#7&1f36f925&0&0000#{378de44c-56ef-11d...	Mouse	Id=256, Nu...		
5	\\?\HID#VID_413C&PID_2107#7&30abe265&0&0000#{884b96c3-56ef-11...	Keyboard		Type=81, S...	

One can plug and unplug the HID device to identify the new hardware serial ID. Press *Refresh* button to see the changes.

In the following tables we show the hardware IDs for our HID devices in 4 different USB ports.

Bar code readers. (both have the same ID in the same USB)

USB	Hardware serial ID
1	\\?\HID#VID_0416&PID_C141#6&328340d9&0&0000#{884b96c3-56ef-11d1-bc8c-00a0c91405dd}
2	\\?\HID#VID_0416&PID_C141#6&d0fdf8f&0&0000#{884b96c3-56ef-11d1-bc8c-00a0c91405dd}
3	\\?\HID#VID_0416&PID_C141#6&1fc99034&0&0000#{884b96c3-56ef-11d1-bc8c-00a0c91405dd}
4	\\?\HID#VID_0416&PID_C141#6&5a9d116&0&0000#{884b96c3-56ef-11d1-bc8c-00a0c91405dd}

Joystick

USB	Hardware serial ID
1	\\?\HID#VID_046D&PID_C215#6&7f59d63&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
2	\\?\HID#VID_046D&PID_C215#6&1e1d0620&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
3	\\?\HID#VID_046D&PID_C215#6&30d6b6c5&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
4	\\?\HID#VID_046D&PID_C215#6&b63557b&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}

Gamepad controller

USB	Hardware serial ID
1	\\?\HID#VID_0079&PID_0006#6&397337fe&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
2	\\?\HID#VID_0079&PID_0006#6&29e3d2aa&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
3	\\?\HID#VID_0079&PID_0006#6&102b948&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}
4	\\?\HID#VID_0079&PID_0006#6&172a2205&0&0000#{4d1e55b2-f16f-11cf-88cb-001111000030}

How FeSCADA works with the HID inputs?

The files: *Joystick.txt*, *Barcode1.txt* and *Barcode2.txt*, have to be created by the user and located in the configuration folder *./CONF/*. The hardware serial ID for the devices has to be copied in these files. If no text saved in a file, there will be no readings from that device.

The hardware serial ID string can be used with full length or shorter (the first part of the string). If it is shorter, any device of that type (that matches the string) can be recognized as valid and messages from it will be processed for input data. For example, the string *\\?\HID#VID* is common for any keyboard, bar code reader or joystick, of different brands and models.

A full string for hardware serial ID will provide security that the accepted data will come only from a certain device type, which is plugged in a certain USB port of the computer.

Once FeSCADA is receiving raw data inputs from a HID device, it will process and transfer the result in some system tags. The user has to define the following tags for data to be transferred in them.

Tag name	type	Description
<i>SysBarcode1</i>	string	
<i>SysBarcode2</i>	string	
<i>SysJ1</i>	integer	First 16 bits from the raw data - Bytes [1, 0]
<i>SysJ2</i>	integer	Next 16 bits from the raw data - Bytes [3, 2]
<i>SysJ3</i>	integer	Next 16 bits from the raw data - Bytes [5, 4]
<i>SysJ4</i>	integer	Next 16 bits from the raw data - Bytes [7, 6]
<i>SysJ5</i>	integer	Next 16 bits from the raw data - Bytes [9, 8]
<i>SysJ6</i>	integer	Next 16 bits from the raw data - Bytes [11, 10]
<i>SysJ7</i>	integer	Next 16 bits from the raw data - Bytes [13, 12]

For *SysBarcode1* and *SysBarcode2* the data is a string of 4-20 digits (or digits and letters, if it is coming from a keyboard).

If for the bar code readers the data is clear, for the joystick and the gamepad controller the user has to further process the raw data in order to obtain useful, meaningful results. See below the application examples.

Tags Setup

Tags List

No	Tag Name	DDE N...	DDE Ch...	Data Type	Update T...	Value	
13	SysDate	0	0	String	Read	10/17/2024	
14	var3	0	0	Integer	Read	5	
15	SysBarcode1	0	0	String	Read	9780857382313	
16	SysBarcode2	0	0	String	Read	KEYBOARD INPUT	
17	SysAlarms	0	0	Integer	Read	1	
18	SysJ1	0	0	Integer	Read	32639	
19	SysJ2	0	0	Integer	Read	32768	
20	SysJ3	0	0	Integer	Read	3968	
21	SysJ4	0	0	Integer	Read	0	
22	SysJ5	0	0	Integer	Read	0	
23	SysJ6	0	0	Integer	Read	0	
24	SysJ7	0	0	Integer	Read	0	
25	X_Left	0	0	Integer	Read	127	

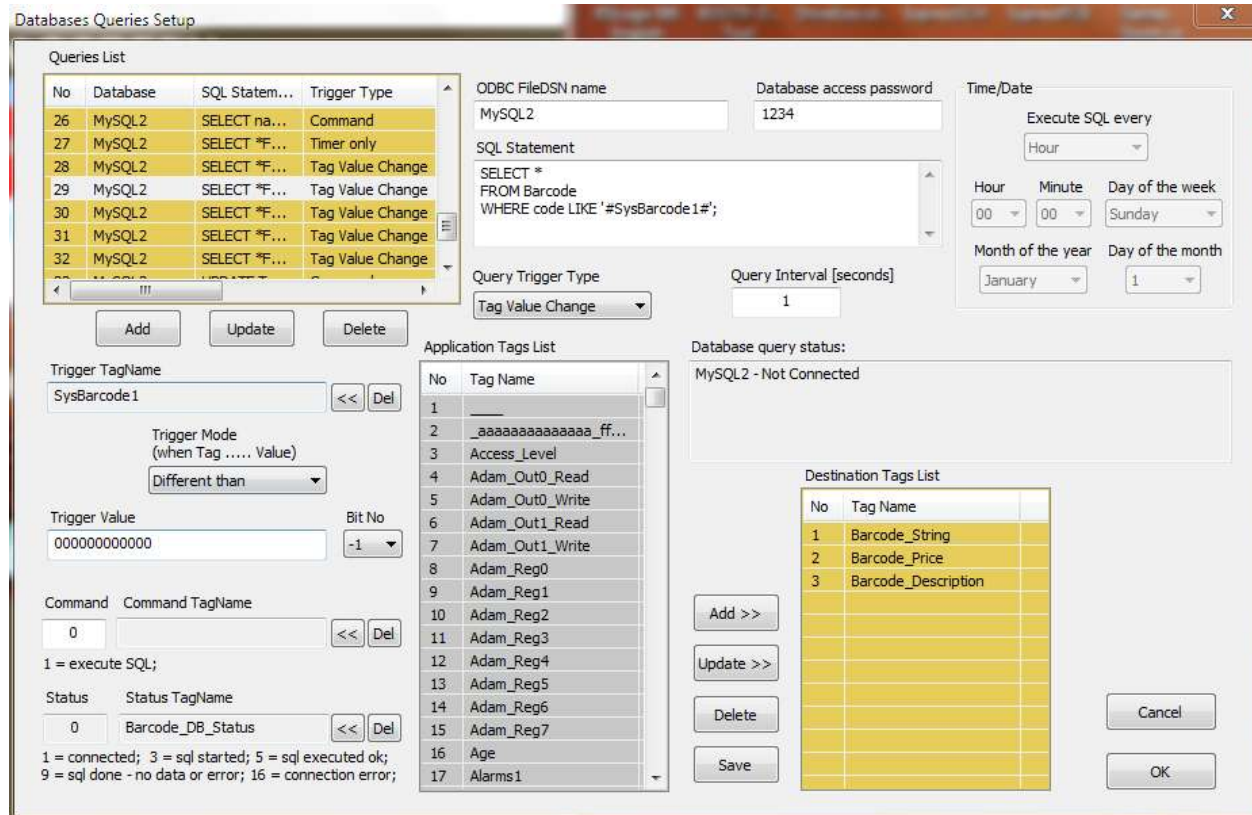
Tag Name: SysJ1 Data Type: Integer Update Type: Read Add

DDE Name: 0 Initial Value: 0 Update Delete Search

DDE Channel: 0

5) Bar code reader application

The most common application is to read from a database after a bar code string ID. FeSCADA can do that with the feature "Database Query". We did set up in a database a table named "Barcode" with the fields: *code*, *price*, and *description*. We did insert different data in this table and we did set up a database query that executes whenever the *SysBarcode1* value is changing. The result of the SELECT... database query is saved in the tags: *Barcode_String*, *Barcode_Price*, *Barcode_Description* (see below).



The results can be visualized in a user defined window, with different text or numerical display controls.

These tags are local. Are not linked with a remote PLC or controller.

Other possible applications are to read in the same way and to send the tags data to remote PLCs/controllers.

Database with bar codes

Barcode PC

655036945520

Database ID:

5

Barcode database

655036945520

Description

Power supply - 2 x 5VDC, 2.5A

Price:

9.50

\$

6) Joystick application

Using the joystick raw data in FeSCADA is a little more complicated because the raw data has to be processed. The raw data is coming in a series of bytes (8 bits) grouped in words (16 bits) and the data has to be extracted from these words.

For example, for our device, Logitech X30, the X and Y axes both have a resolution of 10 bits. That means that more than one byte is used to define one axis value.

FeSCADA is grouping the bytes in words in the order they are coming: first byte is the less significant byte (LSB) of the first word, second byte is the most significant byte (MSB) of the first word, etc.

To process the data we did use more local tags and a logic program. See below.

Logic Setup

Logic List

No	Name	Type	Cyclic	Trigger M...
0	Math	Timer	Yes	Equal
1	Joystick1	Timer	Yes	Equal
2	Joystick2	Timer	Yes	Equal
3	Button	Timer	No	Equal
4	Email	Timer	No	Equal
5	PID	Timer	No	Equal
6	Alarm ke...	Timer	No	Equal
7	Remote_...	Timer	Yes	Equal
8	Ramp	Timer	Yes	Equal

Program edit

```

X_Right = SysJ1 & 1023;
Y_Right = (SysJ1 >> 10) | (SysJ2 & 15) << 6;
Buttons = (SysJ3 & 255) | SysJ4 << 8;
TopHat = (SysJ2 >> 4) & 15;
Speed = SysJ3 >> 8;
Tilt = SysJ2 >> 8;
  
```

Application Tags List

No	Tag Name
1	Buttons
2	Cb0

Logic Setup Details:

- Name:** Joystick2
- Program status:** Loaded: 12, Prepared: 12, Executed: 9
- Buttons:** X_Right, Y_Right, Buttons, TopHat, Speed, Tilt

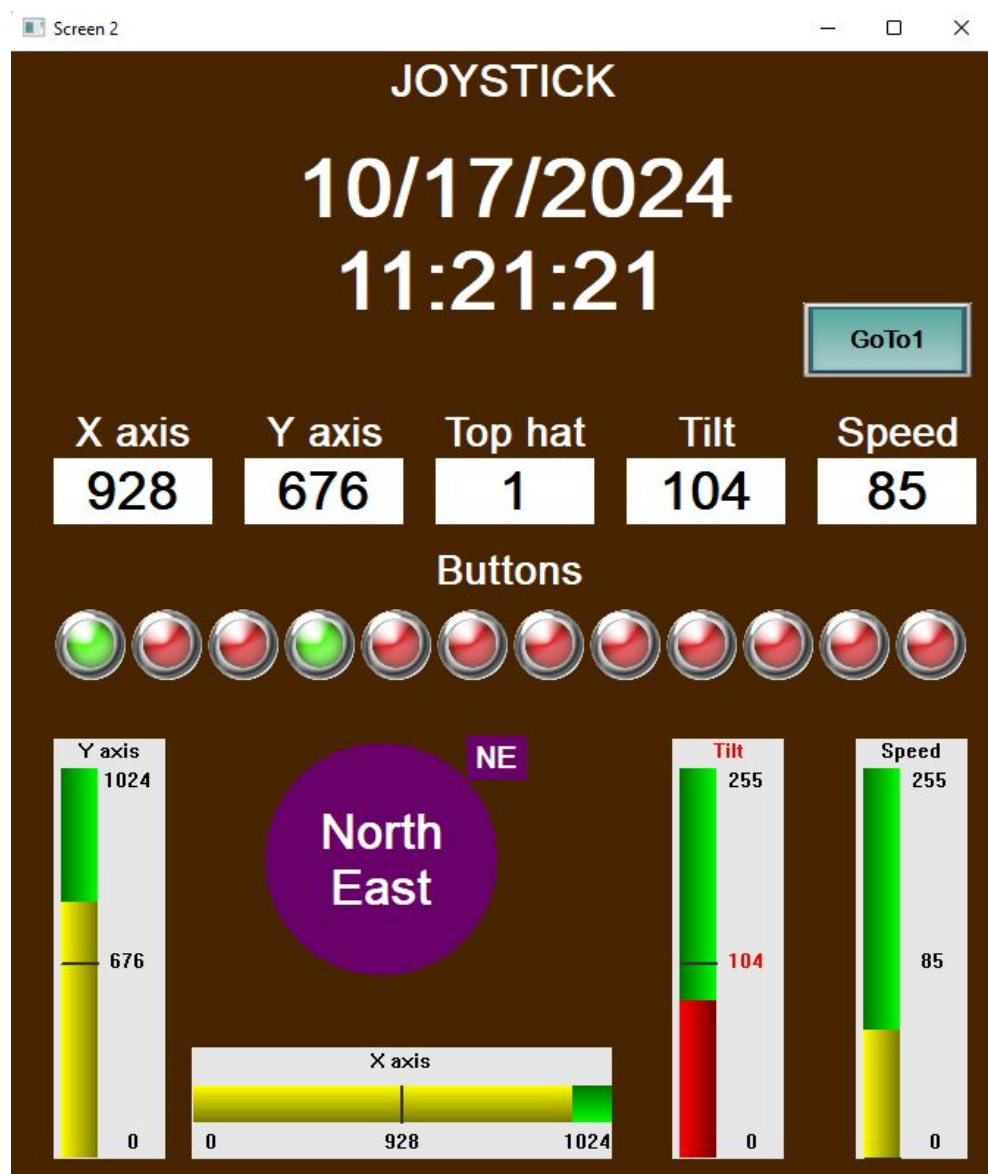
Tag name	type	Description
<i>Right_X</i>	integer	10 bits resolution. The first 10 digits from the first word (byte 0 plus the next 2 bits from byte 1)
<i>Right_Y</i>	integer	10 bits resolution. The last 6 bits from the first word (byte 1) plus the next 4 bits from the second word (byte 2)
<i>Buttons</i>	integer	12 bits extracted from words 3 and 4 (see below)
<i>TopHat</i>	integer	4 bits from word 2 (byte 3)
<i>Tilt</i>	integer	Last 8 bits from word 2
<i>Speed</i>	integer	Last 8 bits from word 3

In the tables below the description of the raw data from Logitech X30 is presented.

	Byte 1 (MSB)								Byte 0 (LSB)							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	0	1	1	1	0	1	1	0	0	1	0	1	1	0	0	1
Word 2	0	1	1	1	0	1	1	0	0	1	0	0	1	1	0	0
Word 3	0	1	1	0	0	0	1	0	0	1	1	1	0	0	1	0
Word 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	X Axis
	Y Axis
	Top Hat
	Tilt
	Speed
	Buttons

In the picture on the right we used the tags defined above to present the data with numerical and graphical controls.



7) Gamepad application

As with the joystick, the data from the gamepad has to be processed. For our device, Marvo GT-006, we did use a logic program to extract the useful data in new local tags. See below.

Logic Setup

No	Name	Type	Cyclic	Trigger M...
0	prog1	Timer	No	Equal
1	Gamepad	Timer	No	Equal

Add

Update

Delete

Name

Gamepad

Program edit

```

X_Left = SysJ1 & 255;
Y_Left = SysJ1 >> 8;

X_Right = SysJ2 & 255;
Y_Right = SysJ2 >> 8;

Buttons1 = SysJ3 >> 12;

Buttons2 = SysJ4;

```

Save

Reload

Execute

Program status

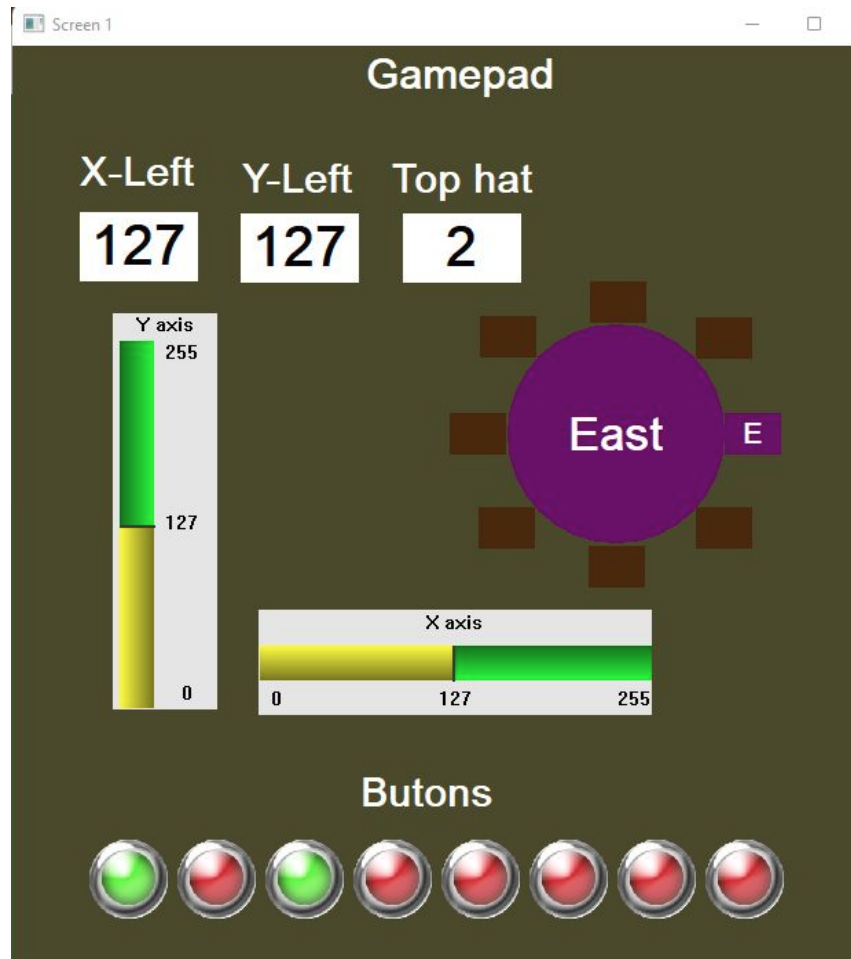
Loaded: 6, Prepared: 6, Executed: 6

Application Tags List

No	Tag Name
1	reg0
2	reg1

Tag name	type	Description
X_Left	integer	Left X 3 positions: 0-127-255 (left Top Hat)
Y_Left	integer	Left Y 3 positions: 0-127-255 (left Top Hat)
Buttons1	integer	The 4 action buttons in 4 bits (right Top Hat)
Buttons2	integer	The other 8 buttons on the gamepad

[illegible]



8) Conclusions

This paper presented the possibilities of FeSCADA in working with bar code readers and joysticks, HID devices affordable and readily available in computer stores.

In automation applications, the bar code readers are very useful in conjunction with a database.

The joysticks (or game pad controllers) can simulate and replace panels with digital button and analog sliders.