ScratchClientArduinoStartup

Inhaltsverzeichnis

1 Purpose, Usecase

The purpose of this tool is a more intuitive way to start scratchClient with the „arduinoUNO“ setup.
In this setup, an arduino uno or nano is used as an IO-expansion board. The arduino can provide an IDENT string which is used to provide only matching configurations for specific arduino.
The usage scenario is especially for classroom setups with multiple, different scratchClient configurations.

A sample use case is:
In a school classroom, there are some general purpose arduino available, but also two fixed setups where one arduino is connected to an elevator simulation and another is connected to a pan-tilt-servo assembly.

The arduino used will be identified by an IDENT code.
The general purpose arduino should receive an empty IDENT code.
The elevator could receive an IDENT code ‚ELEVATOR‘.
The pan-tilt could receive an IDENT ‚PANTILT‘.

2 Step by Step Explanation of Functionality

On startup of the ScratchClientArduinoStartup-Tool, the config file is read and the available arduino are scanned. Based on acquired information, a selection tool is displayed and from this the scratchClient with correct configuration can be launched.

The explanation follows the data in the provided config file sample.

2.1 Step 1: scan the USB devices for connected arduino.

The tool is not checking all possible serial connections, but narrows the devices to a well known list of connections given in a config file.

The config file defines the available hardware devices:

```
  "devices":  [
                    { "os": "win",    "device" : "COM6"  },
                    { "os": "win",    "device" : "COM7"  },
                    { "os": "win",    "device" : "COM19" },
                    { "os": "linux2", "device" : "/dev/ttyUSB0" },
                    { "os": "linux2", "device" : "/dev/ttyUSB1" }
                ],
```

The ‚os‘ attribute is checked with python `sys.platform` value.

When the os-value matches, then a connection for the devive is established and the IDENT code is requested from arduino.

In the internal data storage, the IDENT code and an ‚active‘-flag are added to the devices. As an example, on a linux machine the first USB-device retrieves an IDENT code. The data then are

os	device	active (when an arduino is responding)	IDENT (read from an arduino	comment
win	*COM6*	FALSE		*‚os‘ not matching*
win	*COM7*	FALSE		*‚os‘ not matching*
win	*COM19*	FALSE		*‚os‘ not matching*
linux2	*/dev/ttyUSB0*	TRUE	NANO_000	*Found connection*
linux2	*/dev/ttyUSB1*	FALSE		*No connection available*

Data in blue are from the config file, the IDENT and active-flag are read/calculated from available arduino connections.

Please note that multiple entries could be active, when two or more arduino are connected.
Setups with multiple arduino require unique IDENT for the arduino (scope: host computer).

2.2 Step 2: identify appropriate configurations for available arduino

With the available IDENT code from Step 1, it is checked which of the configurations ‚configs‘ match the available IDENT codes.

```
  "configs":    [
                     { "description": "example 1, start here",
                       "file": "config_appl_001.xml",
                       "idents": [
                                  {  "ident":   "NANO_000" , "alias": "DEVICE_A" }
                                 ]
```

For this snippet, the first entry „`example 1`“ has one ident entry which matches the available arduino IDENT ‚`NANO_000`‘.

So this configuration entry defines a valid scratchClient configuration file ‚config_appl_001.xml‘.

```
"configs":    [
                    ...,
                    {
                      "description": "example 2, need two arduino",
                      "file": "config_appl_002.xml",
                      "idents": [
                                 {  "ident":   "NANO_001" , "alias": "DEVICE_A" } ,
                                 {  "ident":   "NANO_003" , "alias": "DEVICE_B" }
                                ]
                     },
```

The second entry in the config file uses two ident NANO_001, NANO_003 which are not available. This scenario is not applicable.

The last entry in the config file uses regular expression for matching the ident code

```
                   {
                    "description": "example 5",
                    "file": "config_appl_004.xml",
                    "idents": [
                               {  "ident":   ".*" , "alias": "DEVICE_A" }
                              ]
                  }
```

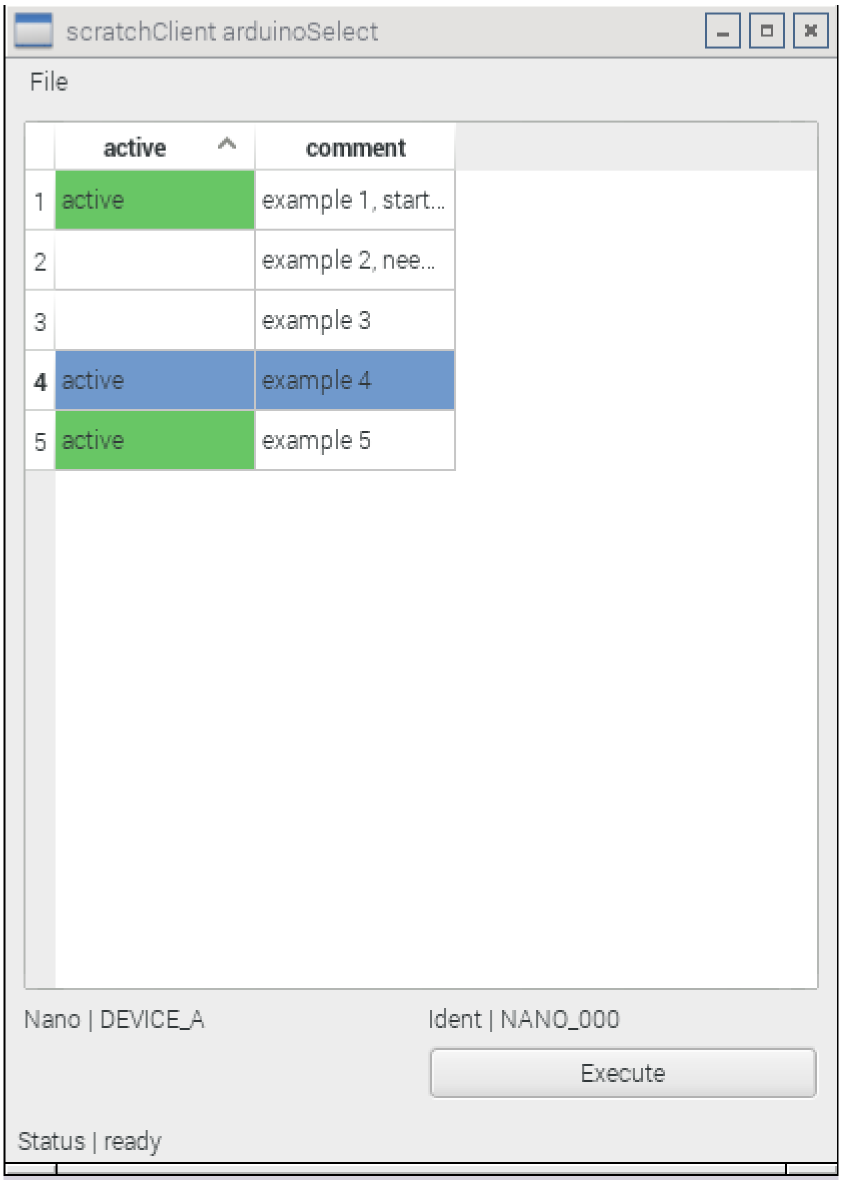
The regular expression „.*“ matches all possible ident codes, so it is available for all arduino.

A regular expression for matching two distinct IDENT is „NANO_003|ELEVATOR“. Look for a tutorial on ‚regular expressions in python‘ for the syntax.

The result of step 2 is a flag assigned to each ‚configs‘-entry active=TRUE or active=‘FALSE‘.

2.3 Step3: display a selection dialog

The configurations are displayed in a table display. Active configurations (which match the IDENT code found in connected arduino) are marked as ‚active‘ and the color green.



	active	comment
1	active	example 1, start...
2		example 2, nee...
3		example 3
4	active	example 4
5	active	example 5

Nano | DEVICE_A

Ident | NANO_000

Status | ready

The cursor is positioned on entry #4, hence the blue ‚selected' color'.

2.4 Step 4: Execute scratchClient with the configuration

2.4.1 Step 4A: patch the configuration

The available configuration files are separate from scratchClient configuration in scratchClientArduinoStartup/config.

The config files have no static device configured, but use a placeholder for this setting.

ScratchClient config snippet:

```
        <parameter name='serial.device' value='${DEVICE_A}' />
```

The syntax is ${somevalue} in ANT parameter style to clearly distinct this from static values.

The value ‚DEVICE_A‘ needs to match the the ‚alias‘-Name from the json config file.

For the example, an IDENT ‚NANO_000‘ was found on a linux machine on /dev/ttyUSB0. The config ‚alias‘ connects ‚DEVICE_A‘ with this value „/dev/ttyUSB0“.

The parameter

```
        <parameter name='serial.device' value='${DEVICE_A}' />
```

gets patched to

```
        <parameter name='serial.device' value='/dev/ttyUSB0' />
```

The patch process copies the config file to a temp directory and during copy it replaces the parameter.

2.4.2 Step 4B: start scratchClient

ScratchClient is started with the patched scratchClient config file.

3 Operations

Installation
The python code needs to have python-qt4 libraries:

```
 sudo apt-get install python-qt4
```

3.1 Start of the tool

Start the tool from a terminal window on desktop.

```
cd scratchClientArduinoStartup
python2 src/main.py
```

4 Best practice to set up configurations

Experiments

Define the experiments to be executed in classroom. This is an ordered list as

- blink LED
- switch LED with button
- Dim LED
- Servo control
…

The tool maintains the order of experiments from the config file into the selection dialog.

scratchClient config file

For each of the experiments, provide a scratchClient config file. It is valid to use scratchClient config file for multiple experiments.

In the config files, each <adapter class='adapter.arduino.UNO_Adapter'-section is responsible for one connected arduino. When multiple arduino are needed, there are multiple adapters.

In each config file, provide unique names for the parameter.value (scope: scratchClient config file) for the

```
        <parameter name='serial.device' value='${DEVICE_A}' />
```

USB connection

Define the available USB connections for the operation system used.

The example defines three devices for a linux machine. When arduino are disconnected and reconnected, it may happen that the current device is blocked (e.g. "/dev/ttyUSB0") and the next one (e.g. "/dev/ttyUSB1") is used on reconnect. So it is a good practice to provide some extra connections.

```
  "devices":  [
                    { "os": "linux2", "device" : "/dev/ttyUSB0" },
                    { "os": "linux2", "device" : "/dev/ttyUSB1" },
                    { "os": "linux2", "device" : "/dev/ttyUSB2" }
                ],
```

Arduino IDENT

Identify ‚general purpose‘ arduino devices and special arduino devices statically connected to experiments. The statically connected arduino are only operable with specific scratchClient configurations, whereas the general purpose arduino could handle multiple experiments and possibly multiple scratchClient configurations.

Program the arduino IDENT code as empty (all blank) or ‚GENERAL‘ or whatever is appropriate for the general usage devices.

Name the special purpose devices with unique names (scope: classroom).

Remark; Build the arduino IDENT from letters, chars and underscore. Example are

NANO_000
ELEVATOR
PAN_TILT

Avoid special chars like: [ ] { } ? * + - . \ ^ ( )

These special chars will make the regular expressions in the configuration configs.idents.ident quite difficult.

Prepare configuration sections.

For each of the experiments, create a setup in the configs-section.

Example:

```
"configs":    [
                {
                  "description": "example 1, start here",
                  "file": "config_appl_001.xml",
                  "idents": [
                              {  "ident":  "NANO_000" , "alias": "DEVICE_A" }
                             ]
                },
```

Declare the arduino IDENT in idents.ident. This is needed to make the connection to dedicated hardware. The idents.ident value is used as a regular expression to match the arduino IDENT code found on USB serial connections.

Doublecheck the scratchClient config file (here "config_appl_001.xml") to use the alias ‚DEVICE_A‘ for the connection parameter..