H2O Beakens definition

1. Introduction

Beakens are assumed to be some reference point for technical adjustment of a process which controls this manner 2 things or more features.
The first 2 approached are possible to implement.
→ H2O irrigation of the land.
→ H2O Energy consumption gas.
→ Plankton control, bread plankton at border of Beakens.
→ Beakens provision (check provision and load/unload area) + debarking area.

2. Approach

Beakers have to be filled in location of harbour to put reservoir in place,
can be dragged to the docking platform, where attached.

This with 2 batteries:
1 - Level batteries to level the reservoir holder.
2 – Measure batteries and energies connection.
3 – Battery level must be full before drift level has to be adjusted
4 – Regeneration of the power with current sources.

3. Hardware approach mini model

1 – Schematics drifter

2 – Schematics platform connection

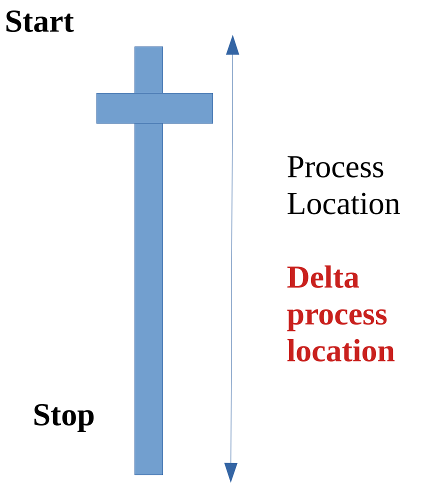
3 – Irrigation schematics

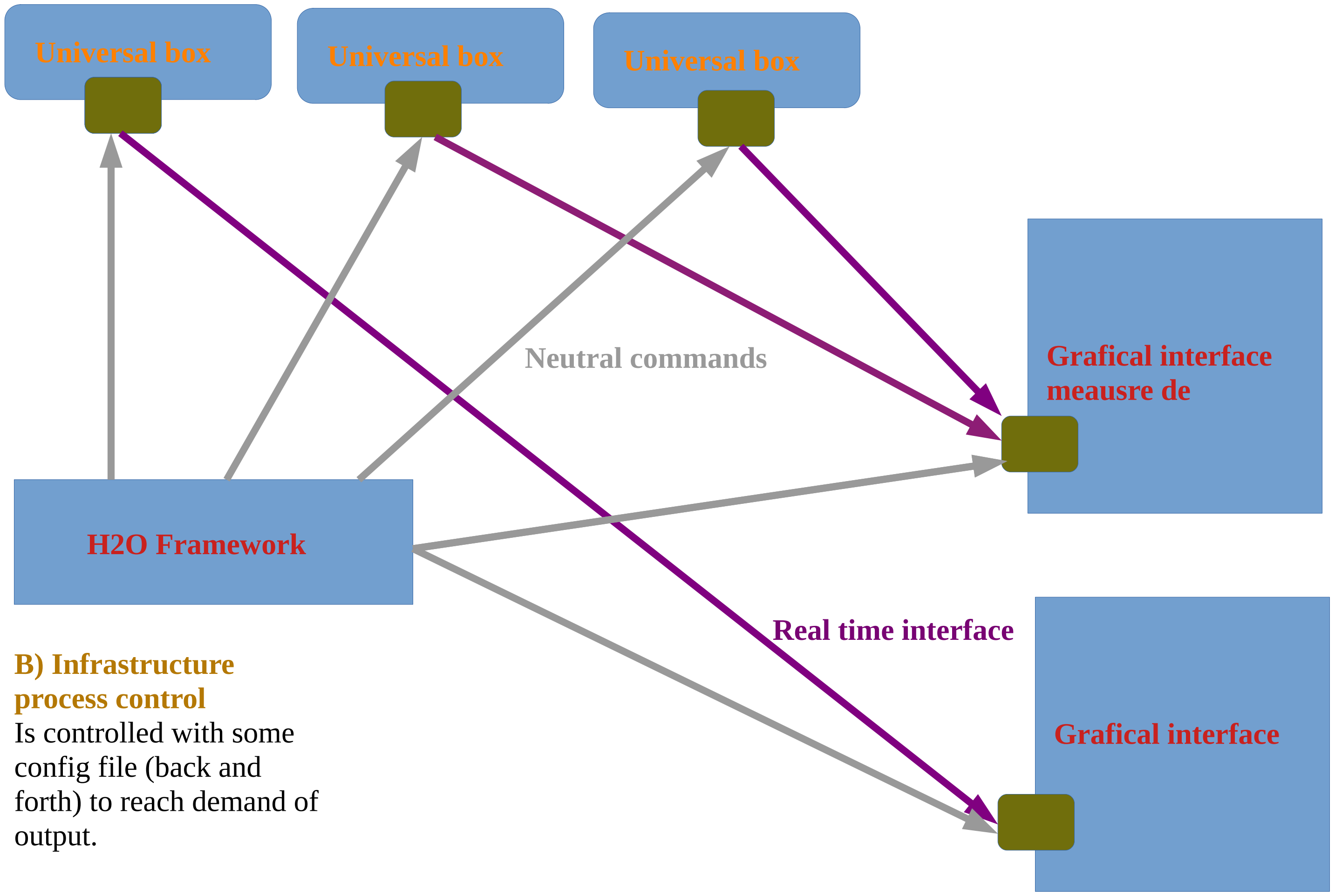
4 – Attached environment schematics

→ (To be milled)

4. Hardware map general with software elements

A) Main process control
Is controlled with some config file (back and forth) to reach demand of output





B) Infrastructure process control
Is controlled with some config file (back and forth) to reach demand of output.

5. Configuration driven approach

Main process control form start till stop **A** in previous step can be controlled with config files, this according state of the process with all measure equipment included or just for one measure equipment. This to set the state in the process. From stop still start (back and forth).

The graphical interfaces are programmed to feedback the out of tolerances in the graphical user interface. When reached this can be done over the neutral command interface. Also supporting the graphical interface. This in optional command to report the out of bounce between certain tolerances.

In the neutral command interface we have also a optional command which contains a socket switch. This when server is down and measure equipment has to be taken over.
The second optional command is holding the real time data input, this comes form the measure equipment.

In the neutral command interface the procedure section can be pre compiled and plugged in into the neutral access of the commands. This to update the procedure while active with certain measure equipment. (Should be done in test case area before updated into the real time environment).

In the neutral command interface the procedure section can be pre compiled and plugged in into the neutral access of the commands. This to update the procedure while active with certain measure equipment. (Should be done in test case area before updated into the real time environment).

Dimensions for the graphical user interface and layout which define the amount of measure equipment. Tolerances according measure equipment and combined measure equipment.

Timing of the systems to make equal which define the amount of measure equipment. Tolerances according measure equipment and combined measure equipment.

Optional command to update the configuration file for the graphical user interfaces equal which define the amount of measure equipment. Tolerances according measure equipment and combined measure equipment.

6. Technical configuration approach

Unit list:

Unit list: **units.csv** for configuration graphical user interface.
TIME;seconds;s;millisecond,10-3;microsecond,10-6;shake,10-8;nanoscond,10-9;picosecond,10-12 yoctosecond,10-24 NEWTON;newton;N; … VOLUME_GAS;m3; … ...

Configuration approach:

Configuration file **graphical** user interface (multiple graphical user interfaces supported)
<root> **<Graphical_user_interface_01>** <measureDevices> <!-- **chromatography (for gas, frequencies)** → <!-- **style** --> <!-- bar, freq, dots --> <!-- **style combined** -->

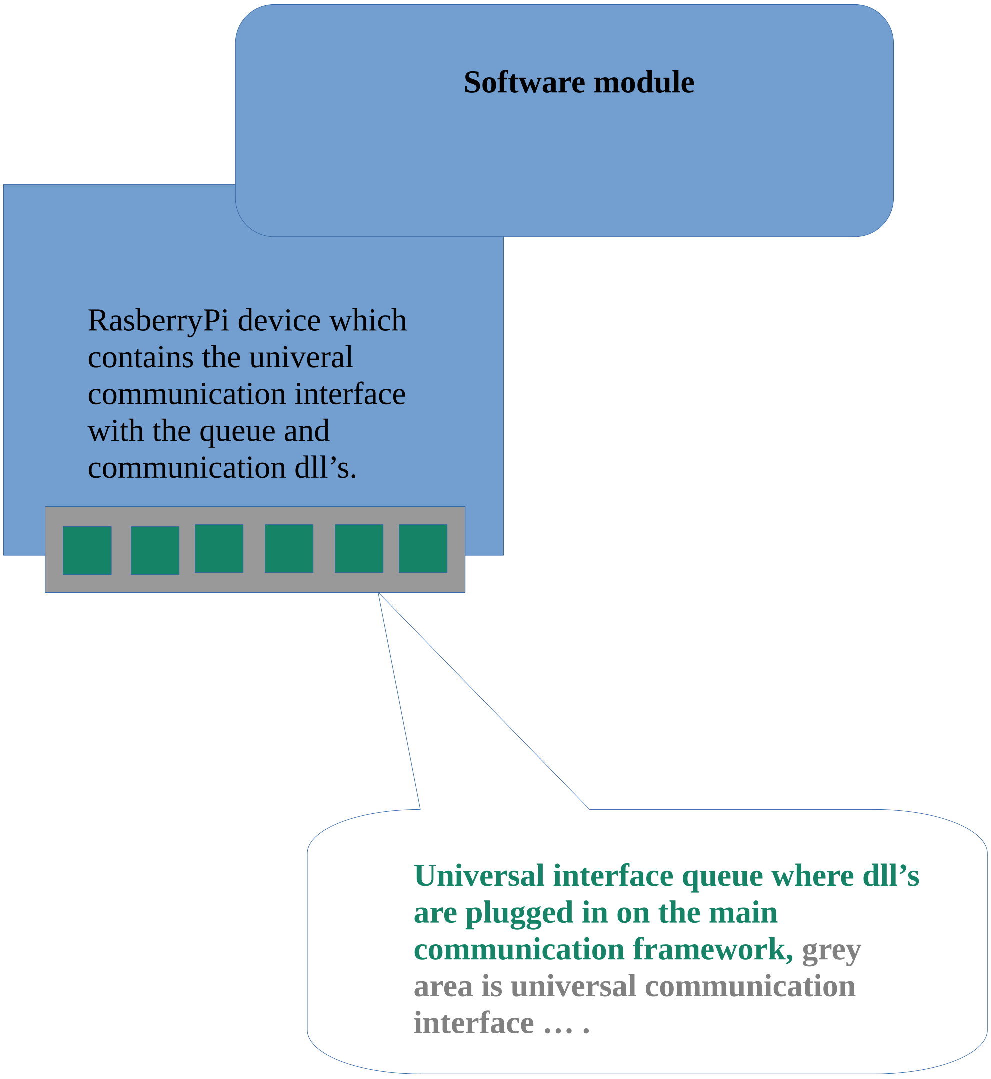
```
<!-- plane, freq, dots, average straight, average chromatography →
<!-- update time -->
<!-- update timing of the process for each and combined devices with unit and timing interval,
reference is seconds  update_time=”10-6” - ->
<!-- Combined devices -->
<combined    name=”Main_Production_output_01” color=”#333333” style=”freq”
update_time=”10-6” unit_definition=”TIME” unit_category=”10-3”>
<tolerances   name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Orange_Upper_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Orange_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Red_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Red_Upper_Lower_Limit_01” feedback=”true”>
</tolerance>
</tolerances>
</combined>
<measureDevice    name=”Produced_Volume_01” color=”#333333” style=”freq”
update_time=”10-6” unit_definition=”VOLUME_GAS” unit_category=”10-9”>
<tolerances   name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Orange_Upper_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Orange_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Red_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Red_Upper_Lower_Limit_01” feedback=”true”>
</tolerance>
</tolerances>
</measureDevice>
<!-- separeted devices  -->
<measureDevice    name=”Buffer_Volume_01” color=”#333333” style=”freq”
update_time=”10-6” unit_definition=”VOLUME_GAS” unit_unit_category=”10-12”>
<tolerances   name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Orange_Upper_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Orange_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Red_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance   name=”Red_Upper_Lower_Limit_01” feedback=”true”>
</tolerance>
```

```
</tolerances>
</measureDevice>
<!-- separeted devices  -->
<measureDevice   name=”Ouput_Volume_01” color=”#333333” style=”freq”
update_time=”10-6” unit_definition=”Newton” unit_category=”10-9”>
<tolerances  name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Orange_Upper_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Orange_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Red_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Red_Upper_Lower_Limit_01” feedback=”true”>
</tolerance>
</tolerances>
</measureDevice>
</Graphical_user_interface_01>
<Graphical_user_interface_02>
<measureDevice   name=”Input_Volume_01” color=”#333333” style=”freq”
update_time=”10-6” unit_definition=”Newton” unit_category=”10-9”>
<tolerances  name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Orange_Upper_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Orange_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Red_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Red_Upper_Lower_Limit_01” feedback=”true”>
</tolerance>
</tolerances>
</measureDevice>
</Graphical_user_interface_02>
<Graphical_user_interface_03>
<measureDevice   name=”Buffer_Volume_01” color=”#333333” style=”blocks”
update_time=”10-6” unit_definition=”Newton” unit_category=”10-9”>
<tolerances  name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Green_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Orange_Upper_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Orange_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Red_Lower_Limit_01” feedback=”true”>
</tolerance>
<tolerance  name=”Red_Upper_Lower_Limit_01” feedback=”true”>
```

</tolerance> </tolerances> </**measureDevice**> **</Graphical_user_interface_03>** </root>
For each state of the process all user interfaces need a state of config file. This means for: **START** to **STOP** of the process we need a certain configuration according the **process state**. They can be transferred over the interface after **different time-spans**. On each timespan we transfer the configuration according the wanted state and wanted output (including the buffer for reaching the target output).

7. Software map

A. Universal communication interface



RasberryPi device which contains the univeral communication interface with the queue and communication dll’s.

Universal interface queue where dll’s are plugged in on the main communication framework, grey area is universal communication interface … .

B. Used technology’s for this implementation:

- Raspberry PI
- Debian
- For the universal queue you can use **wine** and **Linux**

C. Framework

- **State diagram sub** in the normal **sub cycle** of the process.
- **State diagram main** in the **main cycle** of the process.
- **Graphical** user **interface** to **create** the **state process**.

1) Cycle object

- Cycle (state diagram according living objects)
- Defined choice on the properties of the living object

2) Living object

- Event driven objects waiting for operation creation
- Actions which output over the universale interface

3) Dead object

- Properties of the object which can only intput information

4) State object

- State object to remember the state of the cycle object

Living object 1

Living object 2

Living object 3

- ***Tree structure*** with the ***cycle objects***.
- Cycle objects apply to a level.

- ***Living object*** are added into the **group** of the **Cycle Object**. This to provide the process. ***Sub cycle*** are also added into the **group** of the **Cycle objects**.

- ***Each Cycle object*** contains ***living objects***

Main cycle objects:

- Cycle name ‘**main cycle nr. 1 process**‘ level 1
    - Property containing
        * Living object
        * Cycle level 2
        * Cycle level 2
        * Living object
            - operations with tolerances
        * Living object
            - operations with tolerances
    - Properties dead objects
    - Property timer
    - Property scheme
        - Living object operation tolerances
        - Living object state
        - Living object properties

- Cycle name ‘**main cycle nr. 1 process**‘ level 1

….. .

D. Graphical user interfaces

Measure device: **Name_01**

……….

Upper = + 18.14 delta
Lower = - 18.14 delta

Upper warning = + 19.14 delta
Lower warning = - 19.14 delta

Upper max = + 20.14 delta
Upper min = - 20.14 delta

Measure device: **Name_02**

- - - - - - -

Upper = + 18.14 delta
Lower = - 18.14 delta

Upper warning = + 19.14 delta
Lower warning = - 19.14 delta

Upper max = + 20.14 delta
Upper min = - 20.14 delta

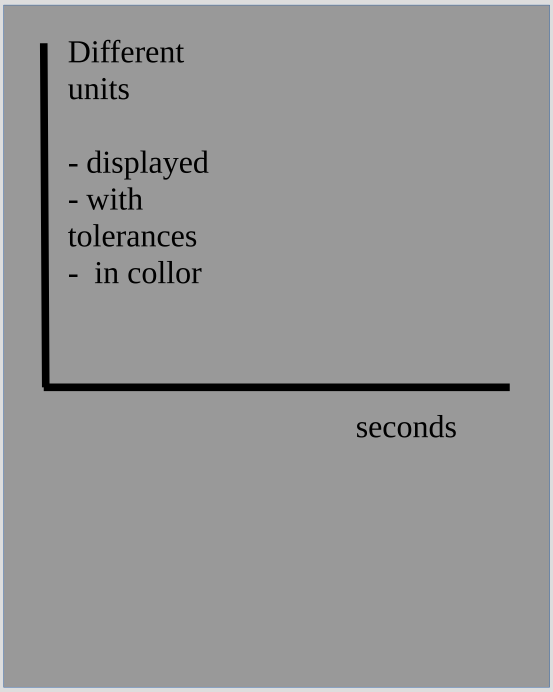
Measure device: **Name_03**

*** * * * * ***

Upper = + 18.14 delta
Lower = - 18.14 delta

Upper warning = + 19.14 delta
Lower warning = - 19.14 delta

Upper max = + 20.14 delta
Upper min = - 20.14 delta



Display of graphs (proposition with config attributes)

*** Style**
- freq
- blocks
- plane

*** Axes**
- style “2D - 3D”

E. Universal communication interface in detail

Definition neutral commands and optional commands:
(HW for the graphical pc's high PC memory cause of binary queue)

- Connect: **Connection command (Neutral socket)**
- Diss Connect: **(Neutral socket)**
- Version request: **(Version request)**
- State Request: **Battery level, Running-state, Online, Wait-state, timestamp (Neutral socket)**
- Error messages queue: **(Neutral socket)**
- Optional: **(Neutral socket)**
- Get binary queue: **(Neutral socket)** ***Representing the incoming real time data***
- *Return* of real time data: **Packet definition (real time data socket)**
- *Incomming* of real time data: **Packet definition (real time data socket)**
- Socket switch (for neutral of real time): **Socket switch for replay on different port (Neutral socket, real time data socket)**
- Platform switch for the dll: **Platform switch (unix for the moment) future plans**

Packet definition for real time data over sockets (dll,s for measure equipment are customer related):

(Packet start definition)
- String of characters '**UNIVERSAL_COMMUNICATION_START**'

(Header contains the neutral dll plugin interface name)
- Name interface, name dll
(Header contains the neutral dll plugin interface name)
- Time stamp
(Header contains the neutral dll plugin interface name) included in mormal packet
- Source ip / destination IP

(Body content with data format)
-Unit Definition
- Unit value
- Time Definition
- Time Value

(Body content optional data format)
-Unit Definition
- Unit value
- Time Definition
- Time Value

(Packet end definition, stop)
- String of characters '**UNIVERSAL_COMMUNICATION_STOP**'

(Wine 4.5 support) for cross platform

Action elements	**comment**
SUB	Element to substract material
MES	Element to take a measurement
BUF	Elements acting as buffer
MAG	Elements acting as magazine
ROB	Elements acting as robot
ADD	Element to substract material

Neutral	**Commands**	**Replay**	**Comment**
STA	**SUBSTA**	STA-->> **1,0,1,345.890**	Satus update of periphery: running/waiting/online/timing
CON	**MESCON**	CON-->>**OK**/**NOK**	Activate interface with/or without pass
DIS	**MESDIS**	DIS-->>**OK**/**NOK**	
VER	**BUFVER**	VER-->>##########.###,######	Return the version command of the periphery
ERR	**BUFERR**	ERR-->> #######, translated message	Return the ERROR code with translation
OPT	**BUFOPTGB**	**OPTGB**-->>**OK***RETURN_STRING/ **NOK***	
BINQUE	**BUFBINQUE**	BINQUE-->’packet header’ + ’packet content’	Return the received data over the REC socket. Return from queue for each command FIFO logic. Memory region.
SEND	**BUFSEN**	SEND-->>’packet content’	Sending the real time data over socket
SWITCH	**BUFREALIN/** **BUFREALOUT/** **BUFNUET**,10.10.1.16	SWITCH-->>**OK**/**NOK**	Switching from the socket: this for the socket real time in data, real time out data, real neutral
POS	**BUFPOS**	POS-->>X21.000,Y23.123,Z12.0,**C0.2333**	Location
REC	**BUFREC**	REC-->>**OK**/**NOK**	Receive, filling up the received data (FIFO) in the BINQUE

8. Modules

External apps	IMPORTANT TODO
Incoming bottleneck request	(connecting the mining software)
Database Editor	
Log_server	
Graphical application	
Config file manager	(Export and import the config files into the graphical user interfaces)
State machine + slider in process	(Displaying the current state of the framework)
Interface access to the framework	(socket with xml support)
Module plugin to communicate to neutral	(plugin in graphical/framework)

9. Database requirements

* Adding columns or objects to the database means we have to match the tables trough update.
* Adding columns or objects to the database means we have to match the tables trough update.
* Stored procedures have to be updated and adjusted in this process.
* Stored procedures have to be transferred to the other server.
* Replication has to be restarted after update of the database configuration.
* Adding columns or objects to the database means we have to match the tables trough update.
* Database solution: Multiple connectors should be used, currently taking the MariaDB for testing.

10. PLC requirements

***** PLC double implemented: if the ladder is performing a cycle at once location the PLC is also performing a cycle in parallel at the other side. This when a module or plc crash the other takes over. Ladder must be in the same state for both in the process.

***** Ethernet: would be possible to define own packet protocol on the PLC side, besides OPC_UA protocol.

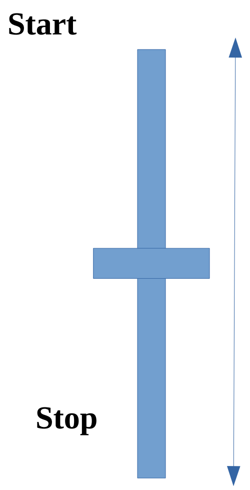
11. Main configuration of the universal interface

Configuration file

```
<root>
<main_configuration>
<name></name>
<dll_location></dll_location>
< - - wine (this could be 1 or 0 according) use or not, 1 is use - - >
<wine>0</wine>
<neutral_socket></neutral_socket>
< - - output (this could be 1 or 0 according) use or not, 1 is use of operations, 0 is use of real time data - - >
<real_time_socket_out></real_time_socket_out>
<real_time_socket_out_timeout></real_time_socket_out_timeout>
<real_time_socket_in></real_time_socket_in>
<real_time_socket_in_timeout></real_time_socket_in_timeout>
</main_configuration>
<dll_specific_configuration>
< - -  Protocol connection, serial = 0, modbus = 1, socket = 2, usb = 3, pipe = 4, OPCUA = 5,  parrallel = 6
- - >
<protocol_selection>4</protocol_selection>
<time_out_general>4</time_out_general>
<time_out_realtime_packet>4</time_out_realtime_packet>
<commands>
<command name=”STA”>
</command_name>
<command name=”CON”>
</command_name>
<command name=”DIS”>
</command_name>
<command name=”VER”>
</command_name>
<command name=”ERR”>
</command_name>
<command_name=”OPT” subcommand_name=“GB”>
</command>
<command_name=”OPT” subcommand_name=“SB”>
</command>
<command name=”BINQUE”>
</command_name>
<!-- sending the command over the
<command name=”SEN”>
</command_name>
<command name=”SWITCH” subcommand_name=”REALIN”>
</command_name>
<!-- BUFREALIN/BUFREALOUT/BUFNUET,10.10.1.16 -->
<command name=”SWITCH” subcommand_name=”REALOUT”>
</command_name>
<command name=”SWITCH” subcommand_name=”NUET”>
</command_name>
```

```
<command name=”POS”>
</command_name>
<! - -  0=UNC, 1=FTP, 2= …  - - >
<command name=”SENDFILE”  send_option=”0”>
</command_name>
<command name=”REC”>
</command_name>
</commands>
</dll_specific_configuration>
</root>
```

12. General prepared config files for process.

Process meter	File name with **time stamp** converted date
 **Current timig: 132456789896**	GeneralConfig_Mes0001_V0001_**132456789123**.xml GeneralConfig_Mes0002_V0001_**132456789456**.xml GeneralConfig_Mes0003_V0001_**132456789789**.xml GeneralConfig_Mes0001_V0001_**132456789888**.xml GeneralConfig_Mes0002_V0001_**132456789891**.xml GeneralConfig_Mes0003_V0001_**132456789893**.xml GeneralConfig_Mes0001_V0001_**132456789894**.xml GeneralConfig_Mes0002_V0001_**132456789896**.xml