LEONARDO
AIRBORNE & SPACE SYSTEMS

DOCUMENT CODE:

DATE:

DOCUMENT TYPE:

APPLICABILITY: **AIRBORNE & SPACE SYSTEMS DIVISION**

LEONARDO PLATFORM SIMULATOR FOR DEVELOPERS

DRAFT

SUMMARY / ABSTRACT:

The purpose of this document is to define a guide on how to project the LEONARDO PLATFORM SIMULATOR software communication interfaces .

	Responsibility / Unit	Name / Signature
Author[s]	SW Capaniliby and Innovation Manager/ Sofware Discipline	Caterina Calcagno
Owner [s]	TBD	TBD
Authority	TBD	TBD

Amendment Summary

Rev.	Date	BMSCP No.	Change Summary	Author[s]
00		-	Initial Issue	

SOMMARIO

1 INTRODUZIONE 4
1.1 Scopo 4
1.2 Applicabilità 4
2 DOCUMENTAZIONE RELATIVA E STRUMENTI 4
2.1 Documentazione 4
2.2 Templates/Forms/Checklists 4
2.3 Strumenti 4
3 DEFINIZIONI E ABBREVIAZIONI 5
3.1 Definizioni 5
3.2 Abbreviazioni 5
4 GUIDA PER LA PROGETTAZIONE DI INTERFACCE PER IL PLATFORM SIMULATOR 5
4.1 File INI 8
4.2 ConfigFile 9
4.3 BusDriver 11
4.4 UiFile 12
4.5 Plugin 13

LIST OF TABLES

No table of figures entries found.

LIST OF FIGURES

No table of figures entries found.

LIST OF APPENDIXES

No table of figures entries found.

LIST OF ANNEXES

No table of figures entries found.

1 INTRODUZIONE

'Leonardo Platform Simulator' è un applicativo rivolto alle attività di test e qualifica di apparati che utilizzano protocolli di comunicazione dati.

Poichè tali attività sono spesso effettuate utilizzando medesime tipologie di bus o metodologie di comunicazione, si è cercato di realizzare un ambiente software atto a soddisfare tali esigenze con il proposito di minimizzare la scrittura di nuovo codice.

1.1 Scopo

Lo scopo di questo documento è definire una guida che aiuti lo sviluppatore a costruire un applicativo basato sul 'CorePlatFormSimulator' in grado di comunicare e quindi operare con il target di cui si vuole testare il funzionamento.

Allo sviluppatore verranno messi a disposizione gli strumenti per la progettazione dell'insieme dei protocolli e della messaggistica per la comunicazione, che per semplicità chiamiamo ***interfacce sw.***

L'applicativo comprenderà tante ***interfacce sw*** quante saranno necessarie per soddisfare tutti i canali di comunicazione richiesti verso il target.

1.2 Applicabilità

Il documento è applicabile a tutti i CSCI SW sviluppati in Leonardo.

2 DOCUMENTAZIONE RELATIVA E STRUMENTI

2.1 Documentazione

Doc Code	Title

2.2 Templates/Forms/Checklists

Doc Code	Title

2.3 Strumenti

Name	Description

3 DEFINIZIONI E ABBREVIAZIONI

3.1 Definizioni

Definition	Description
LA&SSD	Leonardo Airborne & Space Systems Division
SVN	SUBVERSION

3.2 Abbreviazioni

Abbreviation	Description

4 GUIDA PER LA PROGETTAZIONE DI INTERFACCE PER IL PLATFORM SIMULATOR

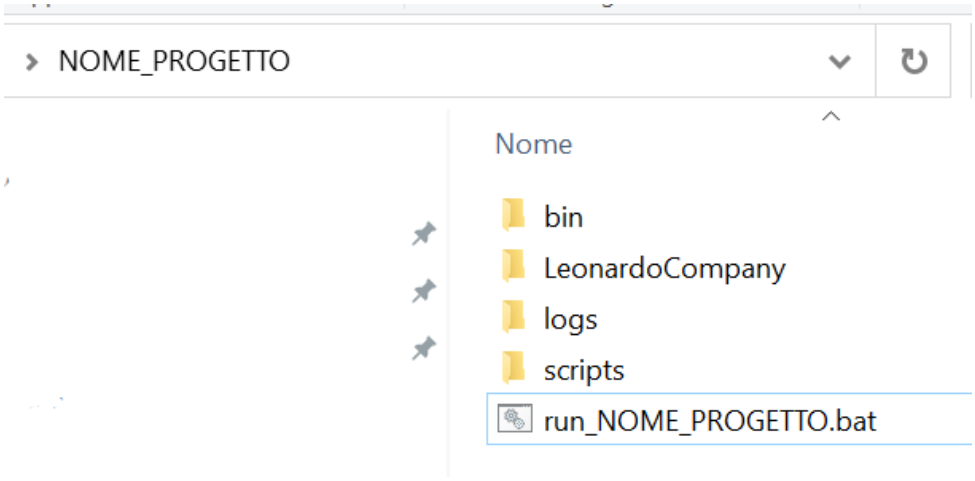
In questo documento verranno date le linee guida per costruire le ***interfacce sw*** per il Platform Simulator: l'alberatura delle cartelle, i nomi, la posizione dei file nel progetto *sono a* ***discrezione dello sviluppatore***, quindi ciò che viene mostrato in queste pagine è a titolo di esempio.

In un'interfaccia possono essere presenti:

- drivers: (.py/.dll)
- plugins: (.py/.dll)
- icd: (.xml)
- GUI: (.ui)

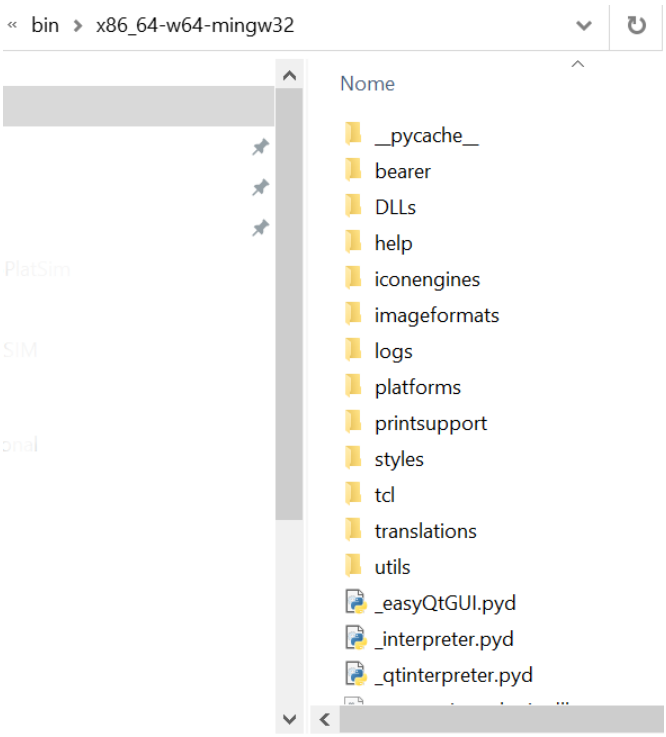
Vediamo nel dettaglio i vari passi:

1. Installazione di qtCreator versione 5 o successive ***se o solo se*** si devono sviluppare plugins che necessitano di oggetti grafici (plotter, etc).
2. Creazione della cartella del progetto **<NOME_PROGETTO>** e della relativa alberatura per la progettazione delle interfacce sw.
   Vedi figura sottostante.



Nel dettaglio:

- **<NOME_PROGETTO>** contiene intero progetto del simulatore con interfacce.
- **bin** contiene la sottocartella x86_64-w64-mingw32 il cui contenuto è il checkout della cartella al link
  https://sassvn.sas.itn/repos/exo-navigation/navigation/tools/PlatformSimulators/General_Purpose_Simulator/PlatformSimulator/tags/REL_055_trunk/bin/x86_64-w64-mingw32/release



- **LeonardoCompany** contiene la sottocartella

  \x86_64-w64-mingw32, che a sua volta contiene:

  - **<NOME_APPARATO>**, cartella che contiene i files dei drivers di comunicazione (BusDriver), i files di configurazione (ICD) in formato XML, il file **.ui** creato in automatico dal PlatSim ed eventuale file **.gss** se lo sviluppatore riterrà opportuno aggiungerlo.
  - Il **FileEsempio.ini**

- **Scripts** contiene i files di test automatici.
  Sarà presente un file <init_interfaccia_1>.py per la inizializzazione dell'interfaccia, come nell'esempio della figura seguente:

init_interfaccia_1.py | test_interfaccia1.py.txt

```
1  interface1=None
2  def startIntFace1():
3    global interface1
4    interface1=interpreter.PyInterfaceManager().getInterface(1)
5    return interface1.start()
```

Nei files <test1_interfaccia_1>.py, < test2_interfaccia_1>.py, etc vengono svolti i tests automatizzati: si deve procedere con la chiamata di inizializzazione dell'interfaccia su cui si eseguono i test e successivamente si devono usare i comandi (API) per scrittura e lettura dei campi dei messaggi (file .XML) del protocollo di comunicazione dell'interfaccia inizializzata come è mostrato nella figura sottostante.

init_interfaccia_1.py | test_interfaccia1.py.txt

```
1  import time
2  import os
3  import sys
4  exec(open('C:/Users/nomeutente/Desktop/NOME_PROGETTO/scripts/init_interfaccia_1.py').read())
5  interface1 = startIntFace1()
6
7  if interface1 == True:
8     print ("Interface1 started...")
9     ret1=interface1.assignMessageFieldValue('PLATFORM COMMAND MESSAGE','ACTIVE WAVEFORM','SBW')
10    interface1.commitChanges()
11    print ("commit done...")
12    time.sleep(1.5)
13    ret2=interface1.getMessageFieldValue('MESSAGE RX_30','COUNTER')
14
15
16 else:
17    print ("Interface1 not started...")
```

- Run_NOME_PROGETTO.bat è un file ***opzionale*** che permette di lanciare il simulatore senza aprire una cmd prompt e agire da riga di comando.
  Di seguito un esempio di file.bat:

```
@ECHO OFF

SET SESSION=TEST

SET ARCH=x86_64-w64-mingw32
SET PYTHON=C:\Python37
SET PLATSIMDIR=%~dp0\bin\%ARCH%\

SET APPDATA=%~dp0

IF NOT EXIST %PYTHON% GOTO PY_ERROR

@ECHO ON
%PLATSIMDIR%\uPlatSim.exe --session=%SESSION% --appdata=%APPDATA% --python=%PYTHON%
GOTO END

:PY_ERROR
@ECHO ######################################
@ECHO WRONG PYTHON PATH.
@ECHO %PYTHON% DOES NOT EXIST
@ECHO ######################################
PAUSE
:END
```

Python 3.7 e' distribuito insieme all'applicativo. Se si preferisce utilizzare una distribuzione di Python equivalente ma con librerie aggiuntive si puo' specificare il path come argomento da passare all'applicativo sia da riga di comandao sia nel file .BAT.

4.1 File INI

Il file che definisce globalmente l'operativita' (sessione) dell'applicativo verso un dato sistema e' un file .INI.

Il file .INI utilizza il seguente formalismo:

```
[Globals]
(IniScript=<path_to_file>{.py})
(StyleSheet=<path_to_file>)
(Icon=<path_to_file>)
(LegacyName=<legacy_code_string>)
(Font=<string>)
(Geometry=<string>)
InterfaceSize=<n>

[Interface_(i=0..n-1)]
Name=First Interface
ConfigFile=<path_to_file>{.xml}
BusDriver=<path_to_file>{.dll|.so}
UiFile=<path_to_file>{.ui}
(Plugin=<path_to_file>{.dll|.so})

[Tools]
(Plugin_0=<path_to_file>{.dll|.so})
```

Di seguito un esempio di file INI che contiene le informazioni necessarie per l'inizializzazione di due interfacce *(Interfaccia1 e Interfaccia2)*:

```
[Interface_0]
Name=<nome_interfaccia_0>
ConfigFile=./<NOME_APPARATO>/icd_interfaccia_0.xml
BusDriver=./<NOME_APPARATO>/interfaccia_0_busdriver.py
UiFile=./<NOME_APPARATO>/nome_interfaccia_0.ui

[Interface_1]
Name=<nome_interfaccia_1>
ConfigFile=./<NOME_APPARATO>/icd_interfaccia_1.xml
BusDriver=./<NOME_APPARATO>/interfaccia_1_busdriver.py
UiFile=./<NOME_APPARATO>/nome_interfaccia_1.ui

[Globals]
InterfaceSize=1
LegacyName=TEST FOO BAR
LegacyVersion=x.x.x
Geometry=@Rect(181 87 1616 870)
Font=@Variant(\0\0\0@\0\0\0\x1c\0M\0S\0 \0S\0h\0\x65\0l\0l\0 \0\x44\0l\0g\0 \0\x32@
\x80\0\0\0\0\0\xff\xff\xff\xff\x5\x1\0\x32\x10)

[<nome_interfaccia_0>]
CurrentLogPath=C:/Users/nome_utente/Desktop/<NOME_PROGETTO>/logs
```

Nel dettaglio:

`[Globals]`

`InterfaceSize=2` → *numero di Interfacce presenti sulla Gui*

`[Interface_0]`

`Name=Interfaccia0` → *nome Interfaccia*

`ConfigFile=./Interfaccia0/Interfaccia0.xml` → *nome e percorso file database ICD*

`BusDriver=./libraries_Interfaccia0/DummyBusDriver.dll` → *nome e percorso file driver di comunicazione*

`UiFile=./Interfaccia0/Interfaccia0.ui` → *nome e percorso file interfaccia grafica*

`Plugin=./Interfaccia0/ProvaPlugin.dll` → *nome e percorso file plugin*

`[nome_Interface_0]`

`CurrentLogPath=./NOME_PROGETTO/logs`

4.2 ConfigFile

E' un file che descrive la composizione dei messaggi di interscambio tra il simulatore ed il target.

Utilizza un formalismo XML le cui regole di composizione sono tracciate dal seguente documento DTD:

```
<?xml version='1.0' encoding='utf-8' ?>
<!DOCTYPE ICDHandler [
<!ELEMENT ICDHandler (HiLevel,LowLevel,Mapping)>
<!ELEMENT HiLevel (Message+)>
<!ELEMENT Message (Field+)>
<!ELEMENT Field (Values*|#CDATA)>
<!ELEMENT Values (Value+)>
<!ELEMENT Value (#CDATA)>
<!ATTLIST ICDHandler
Name CDATA #REQUIRED
Version CDATA #REQUIRED
Endianness (rawbyte|bigendian16|bigendian32|littleendian16|littleendian32) "rawbyte"
Ui (all_tab|all_vertical|all_horizontal|by_iotype) "by_iotype"
>
<!ATTLIST Message
Id CDATA #REQUIRED
Name CDATA #REQUIRED
Width CDATA #REQUIRED
IOType (tx|rx) #REQUIRED
```

```
uSecPeriodTime CDATA #IMPLIED
>
<!ATTLIST Field
Name CDATA #REQUIRED
Width CDATA #REQUIRED
Offset CDATA #REQUIRED
Min CDATA #IMPLIED
Max CDATA #IMPLIED
Weight CDATA "1"
OffsetValue CDATA "0"
Type (spare|boolean|short|ushort|int|uint|long|ulong|double|float|string|enum) #REQUIRED
Unit CDATA #IMPLIED
Display (binary|decimal|hexadecimal) "decimal"
Default CDATA #IMPLIED
Encoding (sign_module|bcd|binary|real|=<formula>(see notes)) "binary"
Resolution CDATA "0.000001"
Reserved (true|false) false
Ui
(default|hidden|entry|spinbox|combobox|radiobox|checkbox|listbox|slider|progressbar|knob)
"default"
>
<!ATTLIST Value
Name CDATA #REQUIRED
>
]>
```

Di seguito un esempio di configfile <interfaccia_1.xml> che comprende 4 messaggi di cui 1 in ricezione (RX) e 3 in trasmissione (TX):

Interfaccia1.xml

```
<ICDHandler Name='Interfaccia1' Version='{9C60369A-DBEE-4DC7-8A36-F9FB3463BC47}' Endianness='BigEndian16'>
  <HiLevel>
    <Message Id='1' Name='MESSAGE 30-RX' Width='128' IOType='RX' >
      <Field Name='COUNTER' Width='16' Offset='0' Type='UINT'  Display='HEXADECIMAL' Default='0' />
    </Message>

    <Message Id='2' Name='MESSAGE 30-TX' Width='128' IsPeriodic='TRUE' uSecPeriodTime='20000' IOType='TX'>
      <Field Name='COUNTER' Width='16' Offset='0' Type='UINT'  Display='HEXADECIMAL' Default='0' />
    </Message>

    <Message Id='3' Name='MESSAGE 31-RX' Width='128' IsPeriodic='TRUE' uSecPeriodTime='20000' IOType='TX'>
      <Field Name='BROADCAST-RESERVED' Width='128' Offset='0' Type='SPARE' />
    </Message>

    <Message Id='4' Name='PLATFORM COMMAND MESSAGE' Width='128' IsPeriodic='FALSE' IOType='TX' >
      <Field Name='ACTIVE WAVEFORM' Width='4' Offset='0' Type='ENUM' Default='VULOS WAVEFORMS' >
        <Values>
          <Value Name='VULOS WAVEFORMS'>0</Value>
          <Value Name='NARROWBAND LOS WAVEFORMS'>1</Value>
          <Value Name='NARROWBAND SATCOM WAVEFORMS'>2</Value>
          <Value Name='SBW'>3</Value>
        </Values>
      </Field>
      <Field Name='RESERVED FOR FUTURE GROWTH' Width='2' Offset='4' Type='SPARE' />
      <Field Name='IBIT ENABLE' Width='1' Offset='6' Type='ENUM'  Default='DISABLED' >
        <Values>
          <Value Name='DISABLED'>0</Value>
          <Value Name='ENABLD'>1</Value>
        </Values>
      </Field>
      <Field Name='RESERVED' Width='2' Offset='7' Type='SPARE' />
      <Field Name='TX BLANKING' Width='1' Offset='9' Type='BOOLEAN'  Default='FALSE' />
      <Field Name='RESERVED' Width='6' Offset='10' Type='SPARE' />

      <Field Name='RESERVED' Width='16' Offset='16' Type='SPARE' />
```

4.3 BusDriver

E' un file, tipicamente una libreria dinamica (.dll, .so, ...) o uno script python, con il quale si costruisce un canale di comunicazione tra l'applicativo ed il target.

E' dipendente dal protocollo utilizzato (RS232, Tcp/IP , GPIB, Arinc 429, MIL 1553 ....).

Puo' far riferimento a file di configurazione esterni.

Puo' ottenere informazioni relazionate al database ICD tramite i nodi XML `LowLevel e Mapping`, i quali sono lasciati a libera implementazione.

Per la costruzione del busdriver si fa riferimento:

- al file ***proxybusdriver.py*** al seguente link:

  https://sassvn.sas.itn/repos/exo-navigation/navigation/tools/PlatformSimulators/General_Purpose_Simulator/BusDrivers/ProxyPythonBusDriver/tags/REL_004_trunk/src/python/proxybusdriver.py

- ai files ***GenericBusEqp.h***, ***genericBusInterface.h*** e ***GenericLoggerInterface.h*** al seguente link:
  https://sassvn.sas.itn/repos/exo-navigation/navigation/tools/PlatformSimulators/General_Purpose_Simulator/interface/tags/REL_012_trunk

Per protocolli di comunicazione **standard** avionici sono disponibili implementazioni di driver di alcuni fornitori (Excalibur, NI,..) al seguente link:

https://sassvn.sas.itn/repos/exo-navigation/navigation/tools/PlatformSimulators/General_Purpose_Simulator/BusDrivers

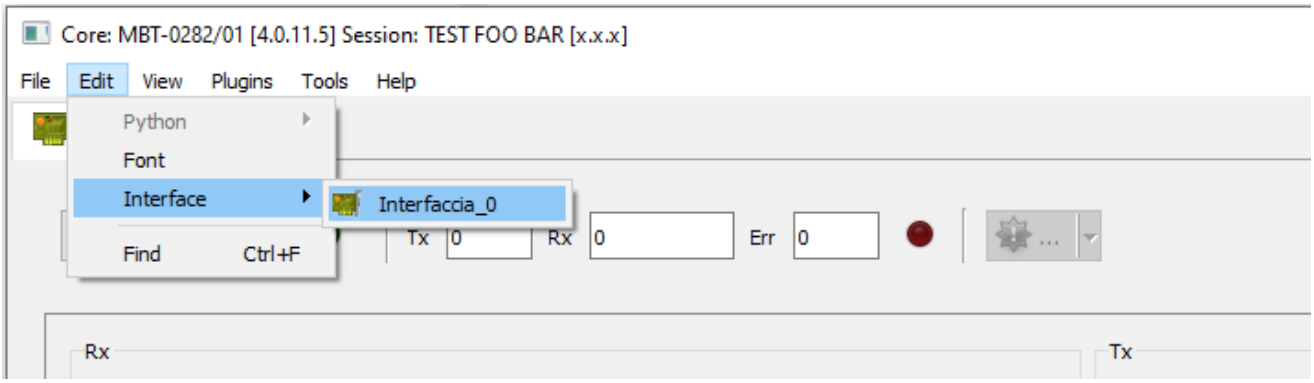
4.4 UiFile

E' un file testuale con il quale si definisce la modalità grafica dell'interfaccia.

Non è obbligatorio modificarlo a meno che non si voglia apportare modifiche particolari all'interfaccia grafica automaticamente costruita con XML.

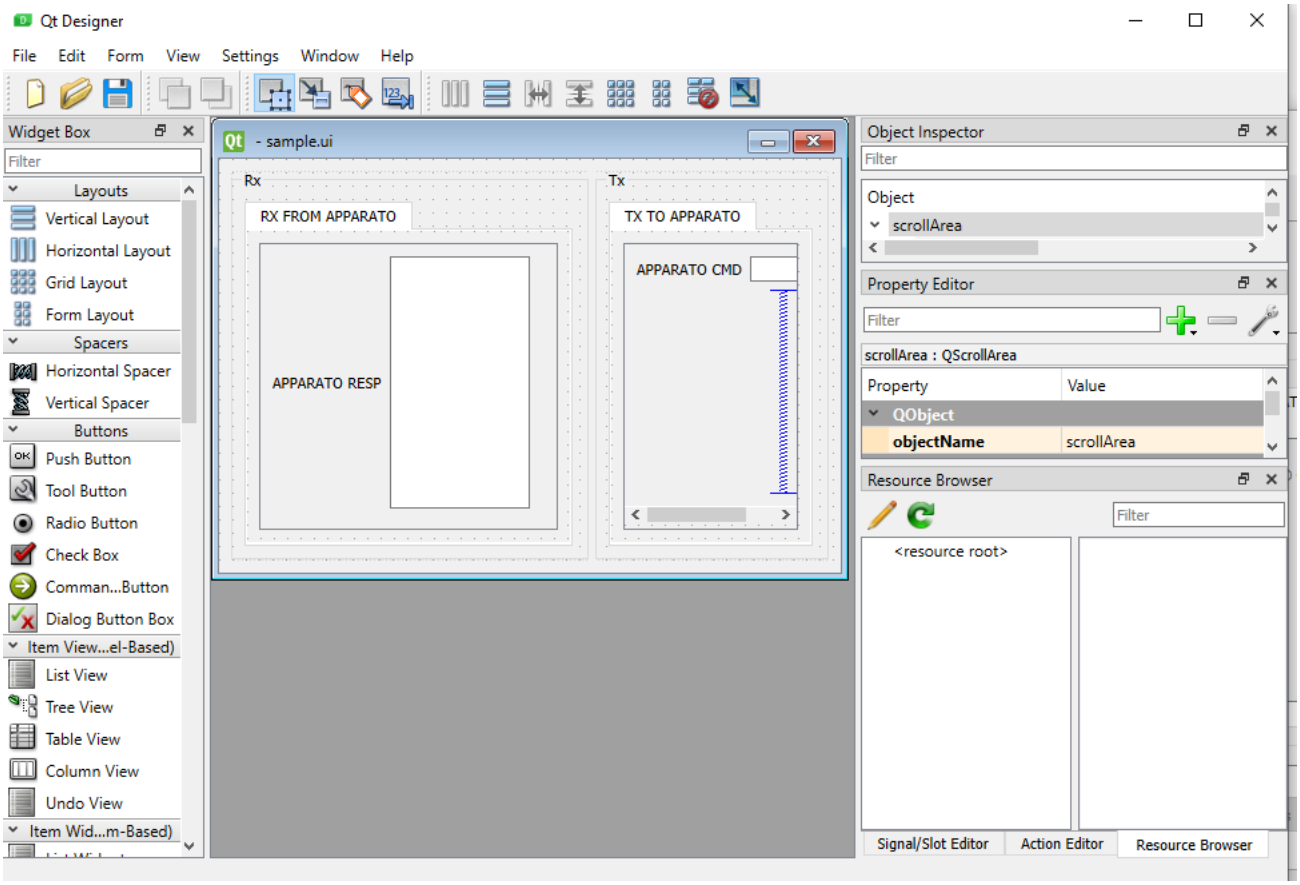
Il PlatFormSimulator mette a disposizione il "designer" come supporto per modificare la struttura della GUI evitando di modificare manualmente il file.ui.

Tale supporto si trova nel menu dell'applicativo, vedi figura sottostante:



Per ogni interfaccia presente si avrà nel menu un "designer" dedicato, con il quale lo sviluppatore può modificare la struttura della GUI.

Nella figura seguente un esempio di designer dell'interfaccia_0:



Si può intervenire anche sull'aspetto estetico della GUI utilizzando un file .QSS, che contiene informazioni su elementi tipo colore, font, icone, etc. Il file va creato e inserito nella cartella LeonardoCompany \x86_64-w64-mingw32\<NOME_APPARATO>

4.5 Plugin

E' un file, tipicamente una libreria dinamica o uno script python, con il quale vengono definite azioni specifiche dell' interfaccia (automatismi , trigger tra messaggi, extra funzionalita') le quali seguono un formalismo descritto nell' SDK (al link https://sassvn.sas.itn/repos/exo-navigation/navigation/tools/PlatformSimulators/General_Purpose_Simulator/PlatformSimulator/trunk/proxyPythonPlugin/sdk).

Al link seguente ci sono i files che definiscono l'interfacccia sw per creare un plugin in cpp:

https://sassvn.sas.itn/repos/exo-navigation/navigation/tools/PlatformSimulators/General_Purpose_Simulator/PlatformSimulator/trunk/plugin/include.

Per i plugins in python si deve accedere al link seguente:

https://sassvn.sas.itn/repos/exo-navigation/navigation/tools/PlatformSimulators/General_Purpose_Simulator/PlatformSimulator/trunk/proxyPythonPlugin.