After loading libraries, setting up GPIO, initializing global variables and preparing log, the application sends a turn off command to signal generator to be sure is going to be prepare for later. Then, main function is executed, step 1.

## Main:

* Fill combobox input with all documents .csv in folder SPECS
* Load wester automation logo
* Link some controls for the application with globals to be able to use them in any place in code.
* Link buttons with functions for actions
* Initialize interlock
* Define interruptions link to GPIOs
* Create Modbus communication
* OPTIONAL: read which boards are inside tester (function: identification\_boards\_hwid)

## Wait for combobox action (Funct: combo\_callback):

Once one product in combobox input is chosen

* Try to open the settings file for that product
  + If there is not such file, ask to choose another product
* Copy that file in the main folder call it settings.py
* Try to open file of the chosen product .csv
* Check the format of the file. Table with 25 columns
  + Format incorrect show an error message and ask for a new choose of product
* OPTIONAL: Check all the boards in .csv are in the tester
* Active next control (user ID)
* Give focus to that new control

## Wait for User ID action (Funct: combo\_callback):

Once ID is introduced and enter key is pressed

* Active next control (GNR)
* Give focus to that new control

## Wait for GNR action (Funct: combo\_callback):

Once ID is introduced and enter key is pressed

* Active next control (Next button)
* Give focus to that new control

## Wait for NEXT Button action (Funct: buttons\_callback):

Once next button is pressed

* Active the screen to show the test section for 4 duts
* Initialize controls for test (Remove old results, part numbers, etc)
* Look at interlock position
  + IF lid is open: show message to close lid (function: win\_scaning)
  + If lid is closed: show message to open lid (function: win\_error)

## Windows OPEN or CLOSED Lid (Funct: win\_error or win\_scaning):

* + Create widow
  + Fill with image and text
  + Wait for Lid action

## Wait for LID action (Funct: my\_callback):

If Lid was closed and now is open:

* Go to thread: threading\_1
  + Close windows
* Wait for closing Lid

If Lid was open and now is closed:

* Go to thread: threading\_1
  + Close window
* Go to scan barcodes (function: my\_BarCode)
* After step 8. Show results in their controls
* Active to use start button in nest
* Show message to press play with thread: thread\_1

## Reading BarCodes (Funct: my\_BarCode):

* Send each board in nest to active coil 31 (active barcode reader)
* Wait 1 second to catch the reading
* Ask to every board in nest the barcode
  + If no barcode is scanned fill variable with “0”
* Save them in an array called settings.barcode
* Return to step 7

## Wait Start button action (Funct: my\_startButton):

Only this is able from end of step 7.

* Close message to press play
* Call function Apply\_test

## Start test sequence (Funct: Apply\_test):

1. Turn off every board and turn off all led for results in nest
2. Get data for report such as date, time, ...
3. If previous test sequence used same specifications just takes the previos data
4. If it is first test sequence or different product: Take data form csv file (num test, ..)
5. Initialize signal generator (Function: start\_sgen)
6. Make loop up to total number of test
   1. The current test is **SERIAL**?
      1. Make a loop up to 4 (4 duts) and only make test if there was a barcode in that position
      2. Call function “start\_tests” (Test\_sequence\_1. py) Step 11
      3. Check type of the result variable to arrange results or keep as they are
      4. Fill variable of report with the result of the test
      5. Check if the result was fail, pass or error
         1. Fail and status=Stop. It will stop the sequence and fill the variable of report with “0” for the rest of test. Go to step 10.7
         2. Error: It will stop the sequence and fill the variable of report with “0” for the rest of test. Go to step 10.7
         3. PASS or (FAIL and status not equal to STOP): update screen with results using thread: Threading\_UpdateTestSequence
   2. The current test is **PARALLEL**?
      1. Call function “start\_tests” (Test\_sequence\_1. py) Step 11
      2. Update screen with results using thread: Threading\_UpdateTestSequence
      3. Loop for 4 DUTs
      4. Fill report variable
7. Turn off all boards
8. Update the big labels for total result of sequence using thread: Threading\_UpdateTestSequence
9. Write report file with the variable report
10. Try to send report to the shared folder
11. Wait for user opens Lid step 12.

## Funct: Start\_testets:

* Look if status is SKIP, if so, exit and return to step 10.c
* Check if for that test there is a calibration done in settings.py. If not return to step 10.c
* Look for the setup of the current test, and load commands for instruments
* Save in dur, the maxima duration of the test
* Active all required relays for that test
* Perform test
* Compare result with pass criteria of that test
* Return to step 10.6.1.3 / 10.6.2.2

## Open Lid (Funct: my\_callback):

When the operator opens the Lid, there is an interruption in one GPIO.

* Go to Threading\_1
* Call NEXT Button action (Funct: buttons\_callback) step 5