Saba

Release 1.0

milad

Jul 30, 2022

Contents

DORSA

WWW.DORSA-CO.IR

Contetns:

1 Electrical

All electronic equipment is in this section

Contetns:

1.1 Plc

1.1.1 About PLC

Category	
Model	Siemens S7 1200c DC/DC/DC
Installation Location	Main Frame
Type of communication	Rj 45 Cooper Cable
Power	DC12v

1.1.2 connections

Name	Count	Description
temperature sensor	2	Taking the temperature of the structure
Limit Switch	4	The integrity of the structure
Projectors	12 block	Enlighten the environment
Cameras	24	Imaging the sheet

1.1.3 Instruction of PLC

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

1.1.4 Dynamic System Connection

سیستم حرکتی : برای اطمینان از صحت قرار گیری سازه در محل خود دو عدد لیمیت سوییچ در انتهای بخش های بالا و پایین و دو عدد در ابتدای بخش های بالا و پایین قرار داده شده است که توسط پی ال سی مقادیر آنها خوانده میشود و در زمان هایی که سیستم به درستی در جای خود قرار ندارد و یا در حال حرکت میباشد به علت جلوگیری از بروز حادثه سیستم به صورت کامل خاموش میشود

1.1.5 Lighting system

سیستم روشنایی : به این صورت میباشد که در ابتدا ۶ بلوک ابتدایی روشن میشوند و پس از آن سیستم پردازشی با توجه به عرض ورق که ازسیستم های خط تولید دریافت کرده در صورت نیاز فرمان روشن شدن پرژکتور های دیگر را صادر میکند برای این کار یک عدد بین ۰ تا ۳ در خانه حافظه مربوطه نوشته میشود , سیتم روشنایی همواره با فرکانس ۶۶ هرتز مشغول به کار میباشد .. note:: به علت وجود عرض سنج پرژکتور های بالا و پایین مقداری تاخیر نسبت به هم دارند و به این علت میباشد که نور پرژکتور ها تصاویر دوربین های کناری را در هنگام اندازه گیری عرض دچار اختلال میکند

1.1.6 Imaging system

سیستم تصویر برداری : نیز به این صورت میباشد که سیستم پردازشی تعداد دوربین های مورد نیاز برای تصویر برداری را در خانه حافظه مربوط به پی ال سی نوشته (یک عدد بین ۰ تا ۶) و پی ال سی با توجه به این عدد دوربین های بالا و پایین را تریگ میزند و تصاویر توسط سیستم پردازشی دریافت میشود

1.1.7 Temperature control System

کنترل دما : برای این امر در ابتدا توسط سیستم پردازشی حد آستانه بالا و پایین و مدت زمان روشن ماندن کولر تنظیم میشود . عملکرد به این صورت میباشد که دما در لحظه در حال چک شدن توسط پی ال سی میباشد در صورتی که دما از آستانه بالا بیشتر شود سیستم خنک کاری مشغول روشن میشود و تا رسیدن به نقطه حداقل روشن میماند حال به این علت که ممکن است در این مدت فشار زیادی به کولر ها وارد شود در صورتی که مدت زمان روشن ماندن کولر بیشتر از زمان تنظیم شده شود سیستم خنک کاری قطع میشود

1.2 Panels

2 Mechanics

3

Softwares

Contetns:

3.1 Main Software

3.2 Train Software

3.3 Setting Software

3.3.1 Description

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

3.3.2 Directories

3.3.2.1 App Logs

Description

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک

میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

3.3.2.2 Backend

Description

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

Contetns:

add_defult_database_records module

`oxin.backend.add_default_database_records.`**`create_default_records`**`(`*ui_obj*, *api_obj*`)`

this function is used to create default records in database, if not exist

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`api_obj`** – (_type_) main api object

camera_funcs module

`oxin.backend.camera_funcs.`**`apply_soft_calibrate_on_image`**`(`*ui_obj*, *image*, *camera_calibration_params*, *pxcalibration=False*`)`

this function is used to apply soft calibration params to camera image

Parameters

- **`ui_obj`** – main ui object
- **`image`** – input camera image
- **`camera_calibration_params`** – input camera calibration params (as a dict)
- **`pxcalibration`** – a boolean determining wheater in pixel calibration step or not

Returns
image: result image that is soft calibrated

`oxin.backend.camera_funcs.`**`assign_existing_serials_to_ui`**(*ui_obj*, *db_obj*, *camera_id*, *available_serials*)

this function is called on every camera selection on camera settngs page, it takes as input available camera serials list, and current camera id, and those serial that not assigned to any camera, and the current camera serial are added to serial combobox on ui

Parameters

- **`ui_obj`** – main ui object
- **`db_obj`** – database object
- **`camera_id`** – current camera id
- **`available_serials`** – list of available camera serals (list of strings)

Returns
None

`oxin.backend.camera_funcs.`**`connect_disconnect_camera`**(*ui_obj*, *db_pbj*, *serial_number*, *connect=True*, *current_cam_connection=None*, *calibration=False*)

this functions is used to connect/disconnect from camera. in connect mode, a connection to camera with input serial number is returned, while in dissconnect mode, the input camera connection is closed

Parameters

- **`ui_obj`** – main ui object
- **`db_pbj`** – database object
- **`serial_number`** – camera serial number (in string)
- **`connect`** – a boolean determning wheter to create a new connection to camer, or disconnect from current camera
- **`current_cam_connection`** – current camera connection object
- **`calibration`** – a boolean determining if camera connection is for camera calibration page

Returns
on conect: camera_connection: the stablished camera connection, if faled, return None message: a meesage determinig the error occured while connecting to camera

Returns
on desconnect: None

`oxin.backend.camera_funcs.`**`convert_cv2_to_qt_image`**(*image*)

this function is used to converte a cv2 image to qt format image

Parameters
`image` – (_type_) image in cv2 format

Returns
qt_image: image in qt format

`oxin.backend.camera_funcs.`**`draw_grid`**`(`*image*, *crosshair=True*`)`

this function is used to draw align grids on input image

Parameters

- **`image`** – (_type_) _description_
- **`crosshair`** – (bool, optional) a boolean determining wheather draw cross-hair or grid. Defaults to True.

Returns

image: image with grid

`oxin.backend.camera_funcs.`**`get_available_cameras_list_serial_numbers`**`()`

this function is used to get available camera serials that are connected to network

Returns

serial_list: list of available camera serials (in string)

`oxin.backend.camera_funcs.`**`get_camera_calibration_params_from_db`**`(`*db_obj*, *camera_id*`)`

this function is used to get camera calibration params from database, given camera id

Parameters

- **`db_obj`** – (_type_) database object
- **`camera_id`** – (_type_) _description_

Returns

camera_calibration_params: in dict

`oxin.backend.camera_funcs.`**`get_camera_calibration_params_from_ui`**`(`*ui_obj*`)`

this function returns the camera sot calibration params from ui

Parameters

`ui_obj` – main ui object

Returns

camera_calibration_params: in dict

`oxin.backend.camera_funcs.`**`get_camera_checkbox_values`**`(`*ui_obj*`)`

this function returns a value determining wheareas to apply camera settings/params to only current camera, or multiple cameras in ui, there are two checkboxes for bottom and top cameras. enabling each of them means to apply current settings to all of the cameras on top/bottom

Parameters

`ui_obj` – main ui object

Returns

a number in range [0, 3], determining wheareas to apply camera settings/params to only current camera, or multiple cameras 0: apply to current camera only 1: apply to top cameras 2: apply to bottom cameras 3: apply to all cameras

`oxin.backend.camera_funcs.`**`get_camera_id`**`(`*camera_name_label*`)`

this function is used to get camera id, using camera name label in ui camera settings page

Parameters
`camera_name_label` – in string

Returns
camera_id: in string

`oxin.backend.camera_funcs.`**`get_camera_params_from_db`**(*db_obj*, *camera_id*)
this function is used to get camera params from database, using camera id

Parameters

- **`db_obj`** – database object
- **`camera_id`** – id of the camera (in string)

Returns
camera_params: a dict containing camera parameters

`oxin.backend.camera_funcs.`**`get_camera_params_from_ui`**(*ui_obj*)
this function is used to get camera parameters from ui (camera settings page)

Parameters
`ui_obj` – main ui object

Returns
camera_params

`oxin.backend.camera_funcs.`**`get_picture_from_camera`**(*camera_connection*)
this function is used to get picture from camera, using its camera connection

Parameters
`camera_connection` – (_type_) _description_

Returns
live_image: image

`oxin.backend.camera_funcs.`**`ip_validation`**(*ui_obj*, *ip_address*)
this function is used to validate ip to be in right format

Parameters

- **`ui_obj`** – main ui object
- **`ip_address`** – input ip address (in string)

Returns
message: a text message determining if the ip validation is ok or not, ‘True’ for validation ok

`oxin.backend.camera_funcs.`**`rotate_calibration_image`**(*image*, *angle*)
this function is used to rotate image along center by input angle

Parameters

- **`image`** – input image
- **`angle`** – input angle to rotate image (in degree)

Returns
rotated_image:

`oxin.backend.camera_funcs.`**`set_camera_calibration_params_to_db`**(*db_obj*, *camera_id*, *camera_calibration_params*)

this function is used to set camera calibration params to database

Parameters

- **`db_obj`** – (_type_) database object
- **`camera_id`** – (_type_) _description_
- **`camera_calibration_params`** – (_type_) in dict

Returns
resault: boolean determining update ok

`oxin.backend.camera_funcs.`**`set_camera_calibration_params_to_ui`**(*ui_obj*, *camera_calibration_params*)

this functino is used to set camera calibration params returned from dataabse, to ui

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`camera_calibration_params`** – (_type_) in dict

Returns
None

`oxin.backend.camera_funcs.`**`set_camera_params_to_db`**(*db_obj*, *camera_id*, *camera_params*, *checkbox_values*)

this function is used to update camera params on database, given camera id(s)

Parameters

- **`db_obj`** – database object
- **`camera_id`** – current camera id (in string)
- **`camera_params`** – dict of camera params
- **`checkbox_values`** – value of camera select checkboxes determing wheareas apply setting to current camera only or to multiple cameras

Returns
result: a boolean value determining if the settings are applied to database or not

`oxin.backend.camera_funcs.`**`set_camera_params_to_ui`**(*ui_obj*, *db_obj*, *camera_params*, *camera_id*, *available_serials*)

this function is used to set input camera params to ui (camera settings page)

Parameters

- **`ui_obj`** – main ui object

- **db_obj** – main database object
- **camera_params** – input camera parameters (in dict)
- **camera_id** – input camera id (in string)
- **available_serials** – available camera serials (list of strings)

Returns
None

oxin.backend.camera_funcs.**set_camera_picture_to_ui**(*ui_image_label*, *image*, *with_zoom=False*, *zoom_min=False*)

this function is used to set an image to ui label

Parameters

- **ui_image_label** – (_type_) ui lable name
- **image** – (_type_) input image
- **with_zoom** – (_type_) boolean determining wheather to zoom image

Returns
None

oxin.backend.camera_funcs.**set_camera_serial_to_ui**(*ui_obj*, *assigned_serial*)

this function takes as input a camera serial and update the serial combobox current value

Parameters

- **ui_obj** – main ui object
- **assigned_serial** – camera serial (in string)

Returns
None

oxin.backend.camera_funcs.**set_widjets_enable_or_disable**(*ui_obj*, *api_obj*, *names*, *enable=True*)

this function is used to get ui element names in a list, and enable/disable them

Parameters

- **ui_obj** – main ui object
- **api_obj** – main api object
- **names** – ui element names (list of strings)
- **enable** – a boolean determinnig whereas enable or disable ui elements

Returns
None

oxin.backend.camera_funcs.**shift_calibration_image**(*image*, *shifth*, *shiftw*)

this function is used to shift image along y or x (vertical or horizintal)

Parameters

- **`image`** – input image
- **`shifth`** – value to shift image horiintaly
- **`shiftw`** – value to shift image vertically

Returns
shifted_image:

`oxin.backend.camera_funcs.`**`show_calibration_summary`**(*ui_obj*, *db_obj*)

this function is used to set/update calibration summary params on ui dashboard page

Parameters
`ui_obj` – (_type_) main ui object

Returns
None

`oxin.backend.camera_funcs.`**`show_cameras_summary`**(*ui_obj*)

this function is used to set/update cameras summary params on ui dashboard page

Parameters
`ui_obj` – (_type_) main ui object

Returns
None

`oxin.backend.camera_funcs.`**`update_available_camera_serials_on_db`**(*db_obj*, *available_serials*)

this function is used to update available camera serials on database, it takes as input available camera serial, and checks the database, for each camera, if serial in database not in available cameras, assign 0 as its serial

Parameters

- **`db_obj`** – (_type_) database object
- **`available_serials`** – (_type_) list of available camera serials (in string)

Returns
None

`oxin.backend.camera_funcs.`**`update_ui_on_camera_connect_disconnect`**(*ui_obj*, *api_obj*, *connect=True*, *calibration=False*)

this function is used to update ui buttons on camera setting and calibration pages, on camera connect/dissconnect on every camera connect, camera take picture button must be enable, while camera connect button must be changed to disconnect on camera disconnect, camera take picture button must set to disable, while camera connect button must be changed to connect

Parameters

- **`ui_obj`** – main ui object
- **`api_obj`** – main api object
- **`connect`** – a boolean determinnig if camera is connected or disconnected
- **`calibration`** – a boolean determining if current page on ui is calibration page

Returns
None

`oxin.backend.camera_funcs.`**`validate_camera_ip`**(*ui_obj*, *db_obj*, *camera_id*, *camera_params*)

this function is used to validate camera ip to be valid and not used by oter cameras

Parameters

- **`ui_obj`** – main ui object
- **`db_obj`** – database object
- **`camera_id`** – current camera ip
- **`camera_params`** – camera parameters (dict)

Returns
result: a boolean determining ip validation is ok or not

Returns
message: the error message of ip validation not ok

`oxin.backend.camera_funcs.`**`zoom_in_calibration_image`**(*ui_obj*)

this function is used to zoom in calibration image on button click

Parameters
`ui_obj` – (_type_) main ui object

Returns
None

`oxin.backend.camera_funcs.`**`zoom_out_calibration_image`**(*ui_obj*)

this function is used to zoom out calibration image on button click

Parameters
`ui_obj` – (_type_) main ui object

Returns
None

camera_connection module

class `oxin.backend.camera_connection.`**`Collector`**(*serial_number*, *gain=0*, *exposure=70000*, *max_buffer=20*, *trigger=True*, *delay_packet=100*, *packet_size=1500*, *frame_transmission_delay=0*, *width=1000*, *height=1000*, *offet_x=0*, *offset_y=0*, *manual=False*, *list_devices_mode=False*, *trigger_source='Software'*)

Bases: `object`

`eror_window`(*msg*, *level*)

`getPictures`(*time_out=50*)

`get_cam`(*i*)

`listDevices`()

Lists the available devices

`serialnumber`()

`start_grabbing`()

`start_grabbing_error_handling`(*error*)

`stop_grabbing`()

`tempreture`()

`trigg_exec`()

`oxin.backend.camera_connection.`**`get_threading`**(*cameras*)

chart_funcs module

`oxin.backend.chart_funcs.`**`create_drive_barchart_on_ui`**(*ui_obj*, *frame_obj*, *chart_title='Chart'*)

this function is used to create bar-chart on storage managment page

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`frame_obj`** – (_type_) ui frame name to create chart in
- **`chart_title`** – (str, optional) _description_. Defaults to ‘Chart’.

Returns

None

`oxin.backend.chart_funcs.`**`update_drive_barchart`**(*ui_obj*, *drives_info*, *storage_thrs*, *warn_storage_thrs*)

this function is used to update drive satues barchart on storage management page

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`drives_info`** – (_type_) statues of the drive (in dict)
- **`storage_thrs`** – (_type_) an int determining thrshold of storage using in bas statues(for chart colors)
- **`warn_storage_thrs`** – (_type_) an int determining thrshold of storage using in warning statues(for chart colors)

Returns
None

colors_pallete module

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

date_funcs module

`oxin.backend.date_funcs.`**`get_date`**(*persian=True*, *folder_path=False*)

this function retrns current date, wheter in persian or miladi.

Parameters

- **`persian`** – a bolean value determining the foramt of date (in persian or miladi)
- **`folder_path`** – a boolean value determiningn if the date will be used as a folder name or not

Returns
current date (in string)

`oxin.backend.date_funcs.`**`get_datetime`**(*persian=True*, *folder_path=True*)

this function returns both curent date and time in wheater persian or miladi format

Parameters

- **`persian`** – a bolean value determining the foramt of date (in persian or miladi)
- **`folder_path`** – a boolean value determiningn if the date will be used as a folder name or not

Returns
date and time: current date and time (in string)

`oxin.backend.date_funcs.`**`get_time`**(*folder_path=False*)

this functionn :returns: current time

Parameters
`folder_path` – a boolean value determiningn if the date will be used as a folder name or not

Returns
time: current time (in string)

defect_management_funcs module

`oxin.backend.defect_management_funcs.`**`add_new_defect_to_db`**(*db_obj*, *new_defect_info*, *defect_group=False*)

this function is used to add a new defect/defect-group to database

Parameters

- **`db_obj`** – (_type_) database object
- **`new_defect_info`** – (_type_) new defect/defect-group info
- **`defect_group`** – (bool, optional) a boolean determining wheather to add defect-group. Defaults to False.

Returns
resault: a message determining if the add to dabase is done “True”: adding ok

`oxin.backend.defect_management_funcs.`**`assign_existing_defect_colors_to_ui`**(*ui_obj*, *db_obj*, *current='None'*)

this function is used to set/update existing defect colors to ui combobox. for a color, if the color dont used for any defects, or be the white, it will be added to combo

Parameters

- **`ui_obj`** – main ui object
- **`db_obj`** – database object
- **`current`** – None, or the id of a defect

Returns
None

`oxin.backend.defect_management_funcs.`**`change_defect_group_id_to_name`**(*db_obj*, *defects_list*, *reverse=False*, *single=False*) this function is used to translate defect-group-ids to defect-group-names or reverse

Parameters

- **`defects_list`** – list of defect_infoes (list of dict)
- **`reverse`** – a boolean determinig wheather to reverse translate or not
- **`single`** – a boolean determining if the input is only one defect record or not

Returns
translated_defects_list: the same defects_list with defect-group-ids translated

`oxin.backend.defect_management_funcs.`**`generate_defect_colors`**(*db_obj*)

this function is used to generate defect colors by number as needed

Parameters
`db_obj` – (_type_) database object

Returns
None

`oxin.backend.defect_management_funcs.`**`get_defect_info_from_ui`**(*ui_obj*, *db_obj*, *defect_group=False*, *is_filter=False*)

this function is used to get defect/defect-group info from ui it can be used to get new defect/defect-group info from ui, or get info from filter/search forms

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`db_obj`** – (_type_) database object
- **`defect_group`** – (bool, optional) a boolean detrtmines wheather to get defect-group info from ui. Defaults to False.
- **`is_filter`** – (bool, optional) a boolena determining wheather to get info from filter form in ui. Defaults to False.

Returns
defect/defect-group info: in dict

`oxin.backend.defect_management_funcs.`**`get_defects_from_db`**(*db_obj*, *defect_groups=False*)

this function is used to get and return defects/defect-groups list from database

Parameters
`defect_groups` – a boolean determining wheather to get defect-groups list or defects list

Returns
if defect_groups==False: defects_list: list of dicts

Returns
if defect_groups==True: defect_groups_list: list of defect groups

`oxin.backend.defect_management_funcs.`**`get_filtered_defects_from_db`**(*db_obj*, *filter_params*, *defect_groups=False*)

this function is used to get filtered defects/defect-groups from database

Parameters

- **`db_obj`** – (_type_) database object
- **`filter_params`** – (_type_)
- **`defect_groups`** – (bool, optional) a boolean determining whaeather to search for defect-groups. Defaults to False.

Returns

message: a text message 'all': no filter 'filtered': return filterd resualts

Returns

defects/defect-groups list:

`oxin.backend.defect_management_funcs.`**`get_selected_defect_groups`**(*ui_obj*, *defect_groups_list*)

this function is used to get selected defect-groups from ui defect-groups table

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`defect_groups_list`** – (_type_) list of defect groups rteturned from darabse

Returns

selected_defect_groups: list of selected defect-group ids

`oxin.backend.defect_management_funcs.`**`get_selected_defects`**(*ui_obj*, *defects_list*)

this function is used to get selected defects from ui defects table

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`defects_list`** – (_type_) defects list returned from databse

Returns

selected_defects: list of selected defects ids

`oxin.backend.defect_management_funcs.`**`load_defects_from_db`**(*db_obj*, *defect_id*, *defect_group=False*, *defect_group_id=False*)

this function is used to load defects/defect-groups from database. the function can be used to get defects, get defect-groups, or get those defects with specified defect-group-id

Parameters

- **db_obj** – (_type_) database object
- **defect_id** – (_type_) defect ids list
- **defect_group** – (bool, optional) a boolean determining wheather to load defect-groups. Defaults to False.
- **defect_group_id** – (bool, optional) a boolean to determine wheather to load defects with a specified defect-group-id (send as defect_id). Defaults to False.

Returns
defect_info: list of defects (in dict)

oxin.backend.defect_management_funcs.**new_defect_info_validation**(*ui_obj*, *db_obj*, *defect_info*, *on_edit=False*, *defect_group=False*)

this function is used to validate new defect/defect-group params to have right format and be unique

Parameters

- **ui_obj** – (_type_) main ui object
- **db_obj** – (_type_) database object
- **defect_info** – (_type_) _description_
- **on_edit** – (bool, optional) a boolean to determine if input defect to validate is in edit mode. Defaults to False.
- **defect_group** – (bool, optional) a boolean to determine wheather to validate a defect-group. Defaults to False.

Returns
message: validation message('True': validation ok)

Returns
level: the level of message (in int)

oxin.backend.defect_management_funcs.**remove_defects_from_db**(*db_obj*, *defects_list*, *defect_group=False*, *defect_group_id=False*)

this is used to remove defect or defect groups from database. we can determine whethere to remove one or multiple defect groups by id, single or multiple defect by id, or all defects with a specified defect-id

Parameters

- **db_obj** – database object
- **defects_list** – list of ids to remove, for removing defects with specified defect group-id, defect-group-id is the input

- **`defect_group`** – a boolean for determining to remove a defect-group or defects
- **`defect_group_id`** – a boolean determining wheather to remove all defects with input defect group id

Returns
resault: a boolean detrtmining if removing from database is done or not

`oxin.backend.defect_management_funcs.`**`set_defect_group_info_on_ui`**(*ui_obj*, *defect_group_info*)

this function is used to set input defect info to ui (for edit defect)

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`defect_group_info`** – (_type_) dict of selected defedct-groups infoes

Returns
None

`oxin.backend.defect_management_funcs.`**`set_defect_groups_on_combo`**(*ui_obj*, *defect_groups_list*)

this function is used to update defect-groups combobox according to available defect-groups

Parameters
`defect_groups_list` – list of defect-groups (in dict)

Returns
None

`oxin.backend.defect_management_funcs.`**`set_defect_groups_on_ui`**(*ui_obj*, *defect_groups_list*)

this function is used to set defect-groups list on ui defect-groups table

Parameters
`defect_groups_list` – list of defect-groups (in dict)

Returns
None

`oxin.backend.defect_management_funcs.`**`set_defect_info_on_ui`**(*ui_obj*, *db_obj*, *defect_info*)

this function is used to set input defect info to ui (for edit defect)

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`db_obj`** – (_type_) database object
- **`defect_info`** – (_type_) dict of selected defedct infoes

Returns
None

`oxin.backend.defect_management_funcs.`**`set_defects_on_ui`**(*ui_obj*, *defects_list*, *defect_group_name='None'*)

this function is used to set input defects list to defects table on ui

Parameters

- **`ui_obj`** – main ui object
- **`defects_list`** – list of defects (in dict)
- **`defect_group_name`** – if not None and have value (in string), those defect s with same defect-group-name will be highlithed

Returns
None

`oxin.backend.defect_management_funcs.`**`show_defects_summary_info`**(*ui_obj*, *db_obj*)

this function is used to show summera info from defects/defect-groups on dashboard page

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`db_obj`** – (_type_) database object

Returns
None

`oxin.backend.defect_management_funcs.`**`update_combo_color`**(*ui_obj*)

this function is used to update color combobox

Parameters
`ui_obj` – (_type_) main ui object

Returns
None

`oxin.backend.defect_management_funcs.`**`update_defects_to_db`**(*db_obj*, *defects_list*, *defect_group=False*)

this function is used to update a defect/defect-group on database

Parameters

- **`db_obj`** – (_type_) databasae object
- **`defects_list`** – (_type_) defect/defect-groups list
- **`defect_group`** – (bool, optional) a boolean determining wheather to update defect-group. Defaults to False.

Returns
resault: a boolean to detrtmine if update on database is ok

logging_funcs module

class oxin.backend.logging_funcs.**app_logger**(*name='saba_setting-app_logger'*, *log_mainfolderpath='./app_logs'*, *console_log=True*)

Bases: object

create_dailyfolder()

this function creates day by day folders in the main folder, to sotring the log files of each day

Returns
None

create_mainfolder()

this function creates the main folder to store log files

Returns
None

create_new_log(*message='nothing'*, *level=1*)

this function creates a log with input message and log level

Parameters
- **message** – the log message (in string)
- **level** – the log level (in int), an int value between [0, 5] specifing the log level) 0: debug 1: info 2: warning 3: error 4: critical error 5: excepion error

Returns
None

set_current_user(*current_username=None*)

this function sets the input username as the current user of the app and logging

Parameters
current_username – current username logged-in the app (in string)

Returns
None

mainsetting_funcs module

oxin.backend.mainsetting_funcs.**apply_appearance_params_to_program**(*ui_obj*, *confirm_ui_obj*, *login_ui_object*, *appearance_params*)

this function is used to apply apearnace params in setting page to app

Parameters
- **ui_obj** – (_type_) main ui object
- **confirm_ui_obj** – (_type_) _description_

- **`appearance_params`** – (_type_) in dict

Returns
appearance_params['window_color']: color of the app

Returns
appearance_params['font_size']: font-size of the app

Returns
appearance_params['font_style']: font-style of the app

`oxin.backend.mainsetting_funcs.`**`assign_appearance_existing_params_to_ui`**(*ui_obj*)

this function is used to assign default apearance params to ui (combobox contents in main-setting page)

Parameters
`ui_obj` – (_type_) main ui object

Returns
None

`oxin.backend.mainsetting_funcs.`**`get_appearance_params_from_ui`**(*ui_obj*)

this function is used to get app appearance params from ui seting page

Parameters
`ui_obj` – (_type_) main ui object

Returns
appearance params: in dict

`oxin.backend.mainsetting_funcs.`**`get_calibration_params_from_ui`**(*ui_obj*)

this function is used to get calibration params from main-setting page

Parameters
`ui_obj` – (_type_) main ui object

Returns
calibration_params: in dict

`oxin.backend.mainsetting_funcs.`**`get_defects_params_from_ui`**(*ui_obj*)

this function is used to get defect params from main-setting page

Parameters
`ui_obj` – (_type_): main ui object

Returns
defects_params: in dict

`oxin.backend.mainsetting_funcs.`**`get_image_procesing_params_from_ui`**(*ui_obj*)

this function is used to get image-preprocessing params from main-setting page

Parameters
`ui_obj` – (_type_) main ui object

Returns
image_procesing_params: in dict

`oxin.backend.mainsetting_funcs.`**`get_mainsetting_params_from_db`**(*db_obj*, *mode='all'*)

this function is used to get mainsetting params from database

Parameters

- **db_obj** – (_type_) database object
- **mode** – (str, optional) select mode to return specific parameters from database. Defaults to 'all'.

Returns
depending on mode 'all': all_params, multitasking params 'px_calibration': rect_areas, rect_acc

oxin.backend.mainsetting_funcs.**get_multitasking_params_from_ui**(*ui_obj*)

this function is used to get multitasking params from main-setting page

Parameters
ui_obj – (_type_) main ui object

Returns
multitasking_params: in dict

oxin.backend.mainsetting_funcs.**set_appearance_params_to_ui**(*ui_obj*, *appearance_params*, *multitask_params=None*)

this function is used to set input apearance params to ui setting page elements

Parameters

- **ui_obj** – (_type_) main ui object
- **appearance_params** – (_type_) in dict
- **multitask_params** – (_type_, optional) if not none, set multtalsk params. Defaults to None.

Returns
None

oxin.backend.mainsetting_funcs.**set_mainsetting_params_to_db**(*db_obj*, *apperance_params*, *is_multitask_params=False*)

this function is used to update/set mainsetting params to database

Parameters

- **db_obj** – (_type_) daabase object
- **apperance_params** – (_type_) params, could be appearance, calibration, image-preprocessing and …
- **is_multitask_params** – (bool, optional) a boolean determining wheather the input params are belonge to multitasking or not. Defaults to False.

Returns
resault: resualts of updating on database

oxin.backend.mainsetting_funcs.**update_combo_color**(*ui_obj*)

this function is used to update setting page color combobox background color by current color

Parameters
ui_obj – (_type_) main ui object

Returns
None

oxin.backend.mainsetting_funcs.**update_combo_fontsize**(*ui_obj*)

this function is used to update setting page fontsize-combobox font according to current app fontsize

Parameters
ui_obj – (_type_) main ui object

Returns
None

oxin.backend.mainsetting_funcs.**update_combo_fontstyle**(*ui_obj*)

this function is used to update setting page fontstyle-combobox font acoading to current app fontstyle

Parameters
ui_obj – (_type_) main ui object

Returns
None

plc_managment module

class oxin.backend.plc_managment.**management**(*ip*, *ui_obj*)

Bases: object

this class is used to create and manage opc/plc object

Parameters

- **ip** – plc ip (in string)
- **ui_obj** – main ui object

Returns
PLC object

connection()

this function is used to connect to plc

Returns
resault: a boolean deermining if connected or not

disconnect()

this functino is used to disconnect from plc

Returns
None

get_value(*path*)

this function is used to get value of a logic from plc using its path

Parameters
path – (_type_) plc logic path (in string)

Returns
value: value stored in path, if failed to load, return '-'

Returns
data_value: if failed to load, return message error

set_file_name(*name*)

this function is used to set json file name to store plc params

Returns
None

set_value(*path*, *value*)

this function is used to set/update value of a logic, using its path on plc

Parameters
- **path** – (_type_) path of the logic (in string)
- **value** – (_type_) input value to update (digit or boolean)

Returns
None

write(*value*)

this function is used to write plc values on json file

Parameters
value – (_type_) in dict

pxvalue_calibration module

oxin.backend.pxvalue_calibration.**apply_pxvalue_calibration**(*ui_obj*, *api_obj*, *db_obj*, *image*, *next=True*)

this function is used in pixel value calibration. the pixel value calibration is done during some steps, in every call of this function, one step (next/prev) is done and the results are updated on ui. this way, we can change between steps and tune parameters to get pixel value results

Parameters
- **ui_obj** – main ui object
- **api_obj** – main api object
- **db_obj** – database object
- **image** – input calibration image
- **next** – a boolean value determninig wheater take to next step or previous step

Returns
None

oxin.backend.pxvalue_calibration.**draw_contour**(*gray*, *cnts*)

this function is used to draw nput contours on image

Parameters
- **gray** – (_type_) image in gray format
- **cnts** – (_type_) contours

Returns
image: image with drawed contours

`oxin.backend.pxvalue_calibration.`**`draw_rect`**(*gray*, *cnts*, *areas*)

this function is used to draw input recangle contours on image

Parameters

- **`gray`** – (_type_) image in gray format
- **`cnts`** – (_type_) contours
- **`areas`** – (_type_) list of areas of rectangles (in mm)

Returns
image: image with drawed contours

`class` `oxin.backend.pxvalue_calibration.`**`extract_info`**(*gray*, *areas_mm*, *min_area=2000*, *max_area=50000*, *accuracy=0.9*, *gray_thrs=100*)

Bases: `object`

this class is used to get pixel-value of camera, using the Dorsa calibrator plate with 6 rectangles (3 pairs)

Parameters

- **`gray`** – input image in gray format
- **`areas_mm`** – list of areas of rectangles (in mm), containing 6 area value, first 3 for large rects, and last 3 for small rects
- **`min_area`** – min area of contours (in pixel)
- **`max_area`** – max area of contours (in pixel)
- **`accuracy`** – min rectangular accuracy for contours
- **`gray_thrs`** – gray threshhold for thresholding

Returns
None

`draw_rects`(*cnts*)

this function is used to draw rectangular contours on image
Parameters
`cnts` – (_type_) input contours
Returns
img: image with drawed countours
Returns
rects: list of 6 rectangle countours

`filter_acc`(*x*)

this function is used to filter a countour by its accuracy to be rectangular
Parameters
x – (_type_) _description_
Returns
type: _description_

`filter_contours_by_accuracy`(*cnts*)

this function is used to filter countours by their accuracy to be rectangular

Parameters

`cnts` – (_type_) input contours

Returns

img: image with drawed countours

Returns

cnts: rectangle accuracy filtered counturs

`filter_contours_by_area`(*cnts*)

this function is used to filter founded contours by min and max area

Parameters

`cnts` – (_type_) input contours

Returns

img: image with drawed countours

Returns

cnts: area filtered counturs

`final_decision`(*cnts*, *rects*)

this function is used to get pixel-values for each of rrectangle pairs

Parameters

- **`cnts`** – (_type_) input contours
- **`rects`** – (_type_) input 6 rectangle contours

Returns

resault: determining if done

Returns

infoes: array of rectangle pair centers and pixel values

Returns

infoes_final: array of rectangle pair centers and pixel values

`find_contours`(*mask*)

find countours of threshold mask

Parameters

`mask` – (_type_) threshold mask

Returns

img: image with drawed countours

Returns

cnts: foundeed counturs

`solve_equation`(*inputs*)

this function is used to solve equation for finding pixel value parameters

Parameters

`inputs` – (_type_) _description_

Returns

pixel_value_parameters: array of 3 parameters

`thrs_map`()

get thresholded/mask from input image

Returns

mask: threshold mask of input image

storage_funcs module

`oxin.backend.storage_funcs.`**`get_available_drives`**`()`

this function is used to get system available drives list

Returns
available_drives: in list

`oxin.backend.storage_funcs.`**`get_camera_live_drive_parameters_from_db`**`(`*db_obj*`)`

this function is used to get camera live drive parameters from database

Parameters
`db_obj` – (_type_) database object

Returns
drive_infoes: app general parameters (in dict)

`oxin.backend.storage_funcs.`**`get_camera_live_drive_parameters_from_ui`**`(`*ui_obj*`)`

this function is used to get defeault storage setting params from ui

Parameters
`ui_obj` – (_type_) main ui object

Returns
resaule: a boolean determining if the parameters are validated or not

`oxin.backend.storage_funcs.`**`get_drivename`**`(`*driveletter*`)`

this function is used to get drive name using its letter

Parameters
`driveletter` – (_type_) in string

Returns
drive_name: in string

`oxin.backend.storage_funcs.`**`get_files_in_path`**`(`*dir_path*, *reverse=False*`)`

this function is used to get all files in a path, sorted by date (old to new)

Parameters

- **`dir_path`** – (_type_) _description_
- **`reverse`** – (bool, optional) a boolean to reverse sorting to new to old. Defaults to False.

Returns
file_paths: list of file pathes

`oxin.backend.storage_funcs.`**`get_storage_status`**`(`*disk_path*`)`

this function is used to get storage statues of one drive

Parameters
`disk_path` – (_type_) drive path (in string)

Returns
drive_info: in dict

`oxin.backend.storage_funcs.`**`remove_old_files_in_directory`**`(`*api_obj*, *ui_obj*, *drive_path*, *dir_path*, *start_ratio*, *stop_ratio*, *reverse=False*`)` this function is used to remove old files in a directory

Parameters

- **`api_obj`** – (_type_) _description_
- **`ui_obj`** – (_type_) main ui object
- **`drive_path`** – (_type_) _description_
- **`dir_path`** – (_type_) directory of the folder in drive
- **`start_ratio`** – (_type_) _description_
- **`stop_ratio`** – (_type_) drive usage threshold to stop removing files
- **`reverse`** – (bool, optional) boolean to reverse sorting files in directory. Defaults to False.

Returns
None

`oxin.backend.storage_funcs.`**`set_camera_live_drive_parameters_to_db`**(*db_obj*, *drive_infos*)

this function is used to set/update drive setting params on database

Parameters

- **`db_obj`** – (_type_) database object
- **`drive_infos`** – (_type_) in dict

Returns
resault: boolean deermining whather set to database is ok

`oxin.backend.storage_funcs.`**`show_storage_status`**(*ui_obj*, *db_obj*)

this functionis used tp update storage info summary on dashboard

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`db_obj`** – (_type_) database object

Returns
None

`oxin.backend.storage_funcs.`**`update_camera_live_drive_combo`**(*ui_obj*, *available_drives*)

this function is used to update existing drives combobox on storage setting age

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`available_drives`** – (_type_) list of available drives

texts module

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

user_login_logout_funcs module

oxin.backend.user_login_logout_funcs.**authenticate_user**(*ui_obj*, *login_ui_obj*, *login_api_obj*, *api_obj*)

this function is used to authenticate the user, it takes the autintication results from login API, and if user be autenticated, enables/unlocks ui for user to work with

Parameters

- **ui_obj** – main ui object
- **login_ui_obj** –
- **login_api_obj** –
- **api_obj** – main API object

Returns
None

oxin.backend.user_login_logout_funcs.**logout_user**(*ui_obj*, *confirm_ui_obj*, *login_api_obj*)

this function is used to logout user from the app, and disable/lock ui after logout

Parameters

- **ui_obj** – main ui object
- **confirm_ui_obj** –
- **login_api_obj** –

Returns
None

oxin.backend.user_login_logout_funcs.**run_login_window**(*ui_obj*, *login_ui_obj*, *confirm_ui_obj*)

this function is used to run/show login window for user login

Parameters

- **ui_obj** – the main ui object
- **login_ui_obj** –
- **confirm_ui_obj** –

Returns
None

`oxin.backend.user_login_logout_funcs.`**`set_app_buttons_enable_or_disable`**(*names*, *enable=True*)

this function is used to enable/disable ui elements, by taking input elements as list of names

Parameters

- **`names`** – ui element names (list of strings)
- **`enable`** – a boolean determining wheather to enable/disable the elements

Returns
None

user_management_funcs module

`oxin.backend.user_management_funcs.`**`add_new_user_to_db`**(*db_obj*, *new_user_info*)

this function is used to add a new user to database

Parameters

- **`db_obj`** – (_type_) database object
- **`new_user_info`** – (_type_) in dict

Returns
resault: a boolean determining if the user is added to database

`oxin.backend.user_management_funcs.`**`get_selected_users`**(*ui_obj*, *users_list*)

this function is used to get selected users from users table in ui

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`users_list`** – (_type_) list of users (in dict)

Returns
selected_users: list of selected users user_names

`oxin.backend.user_management_funcs.`**`get_user_info_from_ui`**(*ui_obj*)

this funcion is used to get user info from ui add user fileds

Parameters
`ui_obj` – (_type_): main ui object

Returns
user_info: in dict

`oxin.backend.user_management_funcs.`**`get_users_from_db`**(*db_obj*)

this function is used to get users list from database

Returns
users_list: list of users (in dict)

`oxin.backend.user_management_funcs.`**`new_user_info_validation`**(*ui_obj*, *db_obj*, *user_info*, *default_user=False*)

this function is used to validate new user info, to be in right format and be unique

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`db_obj`** – (_type_) database object
- **`user_info`** – (_type_) input user_info (in dict)
- **`default_user`** – (bool, optional) a boolean to determine if input user info is for default admin user. Defaults to False.

Returns
message: the text message of validating user_info

Returns
message_level: an int value in range [0, 2] determioning the level of messege

`oxin.backend.user_management_funcs.`**`remove_users_from_db`**(*db_obj*, *users_list*)

this function is used to remove input users from database

Parameters

- **`db_obj`** – (_type_) database object
- **`users_list`** – (_type_) list of user_names

Returns
results: a boolean determining if the removing is ok

`oxin.backend.user_management_funcs.`**`set_users_on_ui`**(*ui_obj*, *users_list*)

this function is used to set input users list to ui users table

Parameters

- **`ui_obj`** – main ui object
- **`users_list`** – list of users (in dict)

Returns
None

`oxin.backend.user_management_funcs.`**`show_users_summary_info`**(*ui_obj*, *db_obj*)

this function is used to show user infoes summary on dashboard

Parameters

- **`ui_obj`** – (_type_) main ui object
- **`db_obj`** – (_type_) database object

3.3.2.3 Calibrational Cal

Description

> کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

Contetns:

Division module

oxin.calibrationCal.Division.**ImageDivision**(*img*, *dim*, *ol*)

load_recent_images module

oxin.calibrationCal.load_recent_images.**load_recent_images**(*path*, *image_count=3*)

main module

Noise module

oxin.calibrationCal.Noise.**NoiseDetection**(*img*, *dim*, *ndim*)

Preprocessing module

oxin.calibrationCal.Preprocessing.**EdgeDetection**(*img*)

oxin.calibrationCal.Preprocessing.**ImageEnhancement**(*img*)

oxin.calibrationCal.Preprocessing.**ImageSmoothness**(*img*)

oxin.calibrationCal.Preprocessing.**SmallNoiseRemoval**(*img*)

SteelSurfaceInspection module

oxin.calibrationCal.SteelSurfaceInspection.**CreateHeatmap**(*gray*, *img*)

oxin.calibrationCal.SteelSurfaceInspection.**FindDefectiveBlocks**(*gray*, *block_size='small'*, *defect_th=0*, *noise_th=7*, *noise=True*, *heatmap=False*)

oxin.calibrationCal.SteelSurfaceInspection.**SSI**(*img*, *block_size='Small'*, *defect_th=0*, *noise_th=7*, *noise=True*, *heatmap=False*)

Variance module

oxin.calibrationCal.Variance.**ImageBlockVariance**(*img*, *dim*, *ol*)

oxin.calibrationCal.Variance.**ThresholdCalculator**(*variance*)

oxin.calibrationCal.Variance.**VarianceCalculator**(*blocks*, *dim*)

3.3.2.4 Database

Description

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

3.3.2.5 Utils

Description

کارکرد پی ال سی : به این صورت میباشد که در ابتدا سنسور تشخیص ورق که متصل میباشد فرمان ورود ورق را صادر میکند پس از ان با فاصله زمانی اندکی سیستم نور پردازی (پرژکتورها)و تریگ زدن به دوربین ها آغاز میشود . وظیفه دیگر پی ال سی مدیریت کردن دمای محفظه های بالا و پایین میباشد برای این امر دو سنسور دما تعبیه شده که در لحظه چک میشوند و در صورت نیاز فرمان روشن شدن کولر های هر بخش صادر میشود , در صورتی که دما بیش از حد باشد فرمان خاموش شدن دوربین ها صادر میشود .

Contetns:

move_on_list module

class `oxin.utils.move_on_list.`**`moveOnList`**

Bases: `object`

this function is used to create a list of elements, with option to go next or preivious on list and get current objet/element

Returns
moveOnList class

`add`(*mylist*, *name*)

this function is used to add a list or elements with name/key

Parameters
- **`mylist`** – (_type_) _description_
- **`name`** – (_type_) name of list

Returns
None

`build_next_func`(*name*)

this function is used to get a next object fot moving nect on a list

Parameters
name – (_type_) name/key of list

Returns
next_on_list oject

`build_prev_func`(*name*)

this function is used to get a previous object fot moving nect on a list

Parameters
name – (_type_) name/key of list

Returns
prev_on_list oject

`check`(*name*)

this function is used to check if a key/name is in class

Parameters
name – (_type_): input name

Returns
resault: boolean detetmining if the name if avilable

`get_count`(*name*)

this function is used to get count of elements in a list

Parameters
name – (_type_) name/key of list

Returns
len_list: _description_

`get_current`(*name*)

this function is used to get curent element in a list

Parameters
name – (_type_) name/key of list

Returns
current_element of list:

get_list(*name*)

this function is used to get a list using its name/key

Parameters

name – (_type_) name/key of list

Returns

list: _description_

next_on_list(*name*)

prev_on_list(*name*)

3.3.3 Files

3.3.3.1 app_settings module

class oxin.app_settings.**Settings**

Bases: `object`

3.3.3.2 confirm_UI module

class oxin.confirm_UI.**UI_main_window**(*language='en'*)

Bases: `QMainWindow`, `Ui_confirm_window`

activate_()

this function connects the close button to its functionality

Returns

None

buttonClick()

this function is used to connect each button to its functionality, on button click

Returns

None

close_win()

this function is used for closing login window

Returns

None

staticMetaObject = <PySide6.QtCore.QMetaObject object>

3.3.3.3 confirm_window module

class `oxin.confirm_window.`**`UI_confirm_window`**

Bases: `QMainWindow`, `Ui_confirm_window`

`activate_``()`

`close_win``()`

`set_language``()`

`set_text``(`*msg=''*`)`

`staticMetaObject = <PySide6.QtCore.QMetaObject object>`

`yes``()`

3.3.3.4 database module

class `oxin.database.`**`dataBase`**`(`*username*, *password*, *host*, *database_name*, *logger_obj=None*`)`

Bases: `object`

this class is used to connect and working with database

Parameters

- **`username`** – username to connect to database
- **`password`** – password to connect to databse
- **`host`** – host of the database
- **`database_name`** – name of the database to work with
- **`logger_obj`** – the logger object to take logs

Returns
None

`add_record``(`*data*, *table_name*, *parametrs*, *len_parameters*`)`

this function is used to add a new record a specified table of database

Parameters
- **`data`** – data to be added to database
- **`table_name`** – in string
- **`parametrs`** – list of parameters (column names) of the database
- **`len_parameters`** – number of parameters

Returns
None

`check_connection``()`

this function is used to check if the connection to databse can be esablished

Returns
a boolean value determining if the connecton is stablished or not

`connect`()

this function is used for connecting to database

Returns
cursor: the object that is used to work with database by queries

Returns
connection: ?

`delete`(*db_name*, *table_name*)

`execute_quary`(*quary*, *cursor*, *connection*, *need_data=False*, *close=False*)

this function is used to execute a query on database

Parameters
- **`quary`** – the input query to execute
- **`cursor`** –
- **`connection`** –
- **`need_data`** – a bolean value
- **`close`** –

Returns
None

`get_all_content`(*table_name*)

this function is used to get/return all contents of a table

Parameters
`table_name` – in string

Returns
table_content: list of records in table (in dict)

`get_col_name`(*table_name*, *param_name*, *value*)

`get_log`(*message='nothing'*, *level=1*)

this function is used to get log from database tasks

Parameters
- **`message`** – (str, optional) _description_. Defaults to 'nothing'.
- **`level`** – (int, optional) level of log. Defaults to 1.

Returns
None

`remove_record`(*col_name*, *id*, *table_name*)

this function is used to remove a record from table acourding to specified column value

Parameters
- **`col_name`** – name of the column to check for (in string)
- **`id`** – value of the column (in string)
- **`table_name`** – name of the table (in string)

Returns
results: a boolean determining if the record is removed or not

`report_last`(*table_name*, *parametr*, *count*)

`search`(*table_name*, *param_name*, *value*, *multi=False*)

this function is used to search in table accoarding to one or multiple specifiic parameter (column name)

Parameters
- **`table_name`** – in string
- **`param_name`** – parameter (column) name in string, for multiple parameters, a list of strings

- **value** – value of the parameter to be (in string), for multiple values, a list of strings
- **multi** – a boolean value determining if the search is according to one parameter or multi parameters

Returns

result: a list containing the returned/searched row (record) in table, if failed to connect to database or nothing was found in table, an empty list will be returned

update_record(*table_name*, *col_name*, *value*, *id*, *id_value*)

this function is used to update a parameter (column) in a table record, detrtmingn by record id

Parameters

- **table_name** – name of the table in database (in string)
- **col_name** – column name of table to update (in string)
- **value** – value will be assigned to column ((in string))
- **id** – name of id column in table, its used to determine which record to update
- **id_value** – value of the id column

Returns

result: a boolean determining if the update on table is done or not

3.3.3.5 database_utils module

class oxin.database_utils.**dataBaseUtils**(*logger_obj=None*)

Bases: `object`

this class is used as an API to work with database

Parameters

logger_obj – the logger object to take loggs

Returns

database object

add_defect(*parms*)

this function is used to add new defect to database

Parameters

parms – (_type_) new defect infoes

Returns

message: determinig if ok or not

add_defect_group(*parms*)

this function is used to add new defect-group to database

Parameters

parms – (_type_) new defect-group infoes

Returns

message: determinig if ok or not

add_user(*parms*)

this function is used to add a new user to users table

Parameters

parms – (_type_) user infoes (in dict)

Returns
resault: a text message determining if the user is added "True": 'Databas Eror':

`get_dataset_path`()

`get_image_processing_parms`()

this function is used to set input image processing params for Miss.Abtahi algo to database

Parameters
`data` – (_type_) image processing params

Returns
image_procesing_params

`load_cam_params`(*input_camera_id*)

this function is used to load camear parameters from camera tables, using the camera id

Parameters
`input_camera_id` – id of camera (in string)

Returns
camera_params: a dict containing camera parameters

`load_defect_groups`()

this function is used to load defect-groups from table

Returns
defect_groups: list of defect-groups (in dict)

`load_defects`()

this function is used to get all defects from defects table

Returns
defects: list of defects (in dict)

`load_general_setting_params`(*is_mutitaskiing_params=False*)

this function is used to get general-settings params from table

Parameters
`is_mutitaskiing_params` – (bool, optional) a boolean determining wheather to load multitasing params from multitasking table. Defaults to False.

Returns
record: list of one dict

`load_plc_ip`()

this function is used to load plc ip from table on dataabase

Returns
record: plc ip (in string), if failed return False

`load_plc_parms`()

this function is used to load plc params from table

Returns
plc_params: in dict, if failed to load from dataabse, return None

`load_users`()

this function is used to load users list from database

Returns
users_list: list of users (in dict)

`remove_defect_groups`(*defect_ids*)

this function is used to remove defect groups from database by their ids

Parameters

`defect_ids` – list of input defect-group-ids (in string)

Returns

resault: a boolean determining if the removing is done

`remove_defects`(*defect_ids*)

this function is used to remove one or multiple defects from database, using their ids

Parameters

`defect_ids` – list if defect-ids (in string)

Returns

results: a boolean determining if the removing is done

`remove_defects_by_group_id`(*defect_ids*)

this function is used to remove all defects with a specific defect-group-id

Parameters

`defect_ids` – input defect-group-id (in string)

Returns

resault: a boolean determining if removig defects is done

`remove_users`(*users_name*)

this function is used to remove input users from database

Parameters

`users_name` – (_type_) list of user_names

Returns

None

`search_camera_by_ip`(*input_camera_ip*)

this function is used to search camera by its ip

Parameters

`input_camera_ip` – (_type_) in string

Returns

record: dict f camera params of camera with input ip

`search_camera_by_serial`(*input_camera_serial*)

this function is used to search camera by its serial

Parameters

`input_camera_serial` – (_type_) in string

Returns

record: dict f camera params of camera with input serila

`search_defect_by_color`(*input_color*)

this function is used to search a defect in database by its color

Parameters

`input_color` – string html code

Returns

defect_list: a list with single defect (in dict)

`search_defect_by_filter`(*parms*, *cols*)

this function is used to search/filter defects by filter params

Parameters

- **`parms`** – (_type_) value of columns to filter
- **`cols`** – (_type_) columns to filter

Returns
defect_info: list of filterd defects

`search_defect_by_group_id`(*input_defect_id*)

this function is used to search defects with specific defect-group-id

Parameters
`input_defect_id` – (_type_) input defect-group-id

Returns
defects_info: a list of defect_infoes (in dict)

`search_defect_by_id`(*input_defect_id*)

this function is used to serach a defect in database, according to its defect-id

Parameters
`input_defect_id` – (_type_) in string

Returns
defect_info: a list with single record (in dict)

`search_defect_by_name`(*input_defect_name*)

this function is used to search a defect by its name in database

Parameters
`input_defect_name` – in string

Returns
defects_list: a list of one defect record (in dict)

`search_defect_by_short_name`(*input_defect_name*)

this function is used to search a defect by its short-name in database

Parameters
`input_defect_name` – in string

Returns
defects_list: a list of one defect record (in dict)

`search_defect_group_by_filter`(*parms*, *cols*)

this function is used to search/filter defect-groups by filter params

Parameters
- **`parms`** – (_type_) value of columns to filter
- **`cols`** – (_type_) columns to filter

Returns
defect_info: list of filterd defect-groups

`search_defect_group_by_id`(*input_defect_group_id*)

this function is used to search a defect-group in database with its id

Parameters
`input_defect_group_id` – in string

Returns
defect_group: list of returned defect groups (since the ids are unique, its a list of one record in dict format)

`search_defect_group_by_name`(*input_defect_group_name*)

this function is used to search a defect-group in table by its name

Parameters
`input_defect_group_name` – (_type_) _description_

Returns
record: list of defects with this name (list of dicts)

`search_user`(*input_user_name*)

this funcion is used to search if any user is available in users table with input username, if username is vailable, user info are returened, else an empty list is returned

Parameters
`input_user_name` – input username to search (in string)

Returns
user_info: a dict containing user info: {user_name: username in string, password: password in string}

`search_user_by_user_name`(*input_user_name*)

this function is used to search a user by its usrnam

Parameters
`input_user_name` – (_type_) in string

Returns
record: user_info (in dict)

`set_image_processing_parms`(*data*)

this function is used to get input image processing params for Miss.Abtahi algo to database

Parameters
`data` – (_type_) image processing params

Returns
None

`update_cam_params`(*input_camera_id*, *input_camera_params*)

this function is used to update camera params of input camera id on table

Parameters
- **`input_camera_id`** – id of crrent camera (in string)
- **`input_camera_params`** – camera parameters (in dict)

Returns
result: a bolean value determining if the settings are updated on database or not

`update_defect`(*input_defect_params*)

this function is used to update a defect on table

Parameters
`input_defect_params` – (_type_) in dict

Returns
resaults: in boolean to deternmine if update is ok

`update_defect_group`(*input_defect_params*)

this function is used to update a defect-group on table

Parameters
`input_defect_params` – (_type_) in dict

Returns
resaults: in boolean to deternmine if update is ok

`update_general_setting_params`(*input_setting_params*, *is_mutitaskiing_params=False*)

this function is used to update general-setting params on table

Parameters
- **`input_setting_params`** – (_type_) _description_
- **`is_mutitaskiing_params`** – (bool, optional) a boolean to determine if params are belong to multitask params. Defaults to False.

Returns
resault: a boolean determining if the update is done

`update_plc_ip`(*ip*)

this function is used to update plc ip on table

Parameters

`ip` – (_type_) plc ip (in string)

Returns

resalt: a boolean determining wheather database updated

`update_plc_parms`(*plc_parms*)

this function is used to update plc params on database

Parameters

`plc_parms` – (_type_) in dict

Returns

resault: a boolean determining wheather update is done

3.3.3.6 eror_window module

`class` `oxin.eror_window.`**`UI_eror_window`**

Bases: `QMainWindow`, `Ui_MainWindow`

`activate_`()

`close_win`()

`mouseMoveEvent`(*self*, *event: PySide6.QtGui.QMouseEvent*) → None

`mousePressEvent`(*self*, *event: PySide6.QtGui.QMouseEvent*) → None

`mouseReleaseEvent`(*self*, *event: PySide6.QtGui.QMouseEvent*) → None

`set_text`(*msg='□□□□ '□□□□□*, *level=3*)

`staticMetaObject = <PySide6.QtCore.QMetaObject object>`

3.3.3.7 login_UI module

`class` `oxin.login_UI.`**`UI_main_window`**(*language='en'*)

Bases: `QMainWindow`, `Ui_MainWindow`

`activate_`()

this function connects the close button to its functionality

Returns

None

`buttonClick`()

this function is used to connect each button to its functionality, on button click

Returns

None

`close_win`()

this function is used for closing login window, also on closing, the password and username fileds are cleared

Returns

None

get_user_pass()

this function is used to get/return entered username and password from fields

Returns
username: in string

Returns
password: in string

set_login_message(*text=''*, *level=0*, *clearable=True*, *prefix=True*)

this function is used to show input message in input label, also there is a message level determining the color of label, and a timer to clear meesage after a while

Parameters

- **label_name** – label element name to show the message in
- **text** – input message to show (in string)
- **level** – level of the message (in int), its a value betweem [0, 2] determining the bakground color of message label
- **clearable** – a boolean value determining whater to clear the message after timeout or not
- **prefix** – a boolean value determinign wheater to show the message prefix or not

Returns
None

showPassword(*show*)

this functino is used for showing/hiding password text in password lineedit

Returns
None

staticMetaObject = <PySide6.QtCore.QMetaObject object>

3.3.3.8 login_api module

class oxin.login_api.**API**(*ui*, *logger_obj=None*, *language='en'*)

Bases: object

this class is used as the API for login window, to take login infoes from user and authenticate the user

Parameters

- **ui** – login ui object
- **logger_obj** – logger object to take logs of user authenticating and logging in
- **language** – the langage to show notifacations of the login

Returns
None

button_connector()

function to connect buttons to their functions

login()

this function is used to authenticate an login the user to app

Returns
result: a boolean value detrmining if the authentication done or not

Returns
user_info: a dict containing infoes of the user {user_name: username in string, password: password in string}

3.3.3.9 notif_UI module

class `oxin.notif_UI.`**`UI_main_window`**(*order=0*)

Bases: `QMainWindow`, `Ui_confirm_window`

`activate_`()

this function connects the close button to its functionality

Returns
None

`buttonClick`()

this function is used to connect ui buttons to their functions

Returns
None

`check_appear_done`()

`close_win`()

this function is used for closing login window

Returns
None

`close_win_2`()

this function is used for closing login window, also stoping progressbar and apear timers and start disapear timers

Returns
None

`progressbar`()

this function us used to update the progressbar value, by a timer. progressbar determines the remained time to finish and close the notfication

Returns
None

`staticMetaObject = <PySide6.QtCore.QMetaObject object>`

`unlock_move_flag`()

`update_current_position`()

`win_appear`(*use_current_pos=False*)

this function is used to appear/show the notification window with an sliding animation, notification window will be appeared from top left of the screen in sliding way

Returns
None

`win_disappear`(*use_current_pos=False*)

this function is used to disappear/hide the notification window with an sliding animation, notification window will be disappeared from top left of the screen in sliding way

Returns
None

`win_move_down`()

this function is used to move the notification down vertically, on any new notification is created

Returns
None

`win_move_down_run_timer`(*reverse=False*)

this function is used to move notification verticaly, if a new notification is created or a previous notification is closed on defalt, it is used to move down the notifications, but it can be used to move up the notifications by the reverse flag

Parameters
`reverse` – a boolean value deermining if the movement is reversly (move to top)

Returns
None

`win_move_top`()

this function is used to move the notification up vertically, if any top notification is closed

Returns
None

`win_startpoint`()

this function is used to detemine the startpoint of the notification window (showing from top right of the screen)

Returns
None

class `oxin.notif_UI.`**`notification_manager`**

Bases: `object`

this class is used to create and handle pop-up notifications of the app, it has functions to create new notification, and manage actived notificaions

Returns
None

`check_active_notifs`()

on every notification creation, this function is called to check the states of previous notifications, and if the last notification is deactived/finished, it most be removed from the actived notifications list

Returns
None

`create_new_notif`(*massage=''*, *win_color=None*, *font_size=None*, *font_style=None*, *level=0*)

this function is used to create a new pop-up notification, by taking as input the notification message and some other params

Parameters

- **`message`** – the notification message (in string)
- **`win_color`** – color of the window (same as the main app default color)
- **`font_size`** – font size of the messsage (same as the main app default)
- **`font_style`** – font style of the messsage (same as the main app default)
- **`level`** – the level of the message, in range of [0, 2], determinnig statues and importance of the message: 0: good statues, only notification 1: warning message 2: error message

Returns
None

`oxin.notif_UI.`**`rearange_active_notifes`**`()`

on every call of this function, all notifications in notification list are checked, and if a notifiacton is deactived (finished), the other actived notifications rearranged and moved to take right position

Returns
None

3.3.3.10 setting_UI module

`class` `oxin.setting_UI.`**`UI_main_window`**

Bases: `QMainWindow`, `Ui_MainWindow`

`activate_``()`

This function will activate ui operating buttons and connect theme to their functions

Returns
None

`animation_move``(`*label_name*, *lenght*`)`

this function is used to shiw/hide an element (frame) with an sliding effect

Returns
None

`buttonClick``()`

this funcion will connect each button in ui to its function

Returns
None

`change_camera_btn_icon``(`*camera_id*, *active=False*`)`

this function is used to change the current camera icon in camera settings page

Parameters
- **`camera_id`** – id of the cameras (in string)
- **`active`** – a boolean value determinging wheater the camera is selected or deselected

Returns
None

`check_box_state``(`*b*`)`

this function is used to change checkbox text to enable/disable by checkbox state

Parameters
b – checkbox element

Returns
None

`clear_line_edits`(*line_edits*)

this function is used to clear the lineedit texts

Returns
None

`close_app_force`(*message='An Error occured while runnung the app'*, *change_language=False*)

this function closes the app in force situations (app errors or excetions), also a log will be written determining the cause for closing the app, and an alert window will be appeared to warn the app closing

Parameters

- **`message`** – message to log on app close (in string)
- **`change_language`** – a boolean determines if the app close is for changing the app language

Returnes
None

`close_win`()

this function closes the app

Returns
None

`combo_image_preccess`(*s*)

`disable_camera_settings`()

this function will disable all camera params fileds in camera setting page, on camera disable/change or stackwidjet change

Returns
None

`get_image_proccessing_parms`()

this function is used to take and return entered image calibration parms of Miss.Abtahi algo from ui

Returns
dict{block_size, defect, noise, noise_flag}

`get_label`(*label_name*)

this function is used to take and return the text content of a label elemnt

Parameters
`label_name` – name of label element

Returns
None

`get_plc_ip`()

this function takes anf returns input PLC IP from ui

Returns
PLC ip: (in string)

`get_plc_parms`()

this function will take and :returns: the input PLC parameters and addreses from ui

Returns
dict: {limitswitch_top_plc, limitswitch_bottom_plc, thermometer_min_plc, thermometer_max_plc, cooler_uptime_plc, system_operating_plc, air_valve_plc, camera_limit_plc':[camera_limit_path, -1, -1], camera_frate_plc, projector_limit_plc, detect_sensor_plc

`get_user_pass`()
this function is used to get and return entered username and password from login window
Returns
username: in string
Returns
password: in string

`get_width_guage_parms`()
this function will returns the user slected camera in calibration page

Return: camrera id (in string)

`leftmenu`()
this function s used to show/hide the left side bar with an sliding effect
Returns
None

`maxmize_minimize`()
this function chages the window size of app
Returns
None

`minimize_win`()
this function minimizes the app to taskbar
Returns
None

`mouseMoveEvent`(*self*, *event: PySide6.QtGui.QMouseEvent*) → None

`mousePressEvent`(*self*, *event: PySide6.QtGui.QMouseEvent*) → None

`mouseReleaseEvent`(*self*, *event: PySide6.QtGui.QMouseEvent*) → None

`selected_camera`(*s*)
this function is used to change the camrea icon in calibration page
Parameters
`s` – id of camera (in int)
Returns
None

`set_button_enable_or_disable`(*names*, *enable=True*)
this function will enable or disble all the ui elements in the input list
Parameters
- **`names`** – ui elements (in list)
- **`enable`** – a boolean value determining wheather to enable/diable the elements

`set_checkboxes`()
this function is used to connect checkboxes in ui to their functions

Returns
None

`set_combo_boxes`()

this function is used to set the content of comboboxes in ui

Returns
None

`set_default_image_proccess`(*value*)

`set_image_label`(*label_name*, *img*)

this function is used to set/fit an image to a label element

Parameters
- **`label_name`** – name of the label element
- **`img`** – input image to fit/set to label

Returns
None

`set_image_proccessing_parms_to_ui`(*image_processing_params*)

this function is used to take and return entered image calibration parms of Miss.Abtahi algo from ui

Returns
dict{block_size, defect, noise, noise_flag}

`set_label`(*label_name*, *msg*, *color='black'*)

this funcion will set a text message to a label element, with text color

Parameters
- **`label_name`** – label element name
- **`msg`** – input message (in string)
- **`color`** – message/text color (in string, html code or color name)

Returns
None

`set_login_message`(*text*, *color*)

this function is used to set login message on login window

Parameters
- **`text`** – message to show (in string)
- **`color`** – color of the message text (in string, html code without #, or color name)

Returns
None

`set_plc_ip`(*text*)

this function will set input PLC IP from database to ui field

Parameters
`text` – PLC ip (in string)

Returns
None

`set_size`(*frame_name*, *size*, *minimum=False*, *maximum=False*)

this function is used to set maximum or minimum height for an element (frame)in ui

Parameters
- **`frame_name`** – name of frame element
- **`size`** – height/size of elemen

- **minimum** – a boolean value determning wheater the input height/size is minimumheight or not
- **maximum** – a boolean value determning wheater the input height/size is maximumheight or not if both minimum and maximum be False, the size will be applied as both minimumheight and maximumheight

Returns
None

set_sliders()

this function is used to connect siders in ui to their functions

Returns
None

show_mesagges(*label_name*, *text=''*, *level=0*, *clearable=True*, *prefix=True*)

this function is used to show input message in input label, also there is a message level determining the color of label, and a timer to clear meesage after a while

Parameters

- **label_name** – label element name to show the message in
- **text** – input message to show (in string)
- **level** – level of the message (in int), its a value betweem [0, 2] determining the bakground color of message label
- **clearable** – a boolean value determining whater to clear the message after timeout or not
- **prefix** – a boolean value determinign wheater to show the message prefix or not

Returns
None

show_value(*value*)

this function is used to show slider value in an label/textbox

Parameters
value – value of the slider (in int)

Returns
None

staticMetaObject = <PySide6.QtCore.QMetaObject object>

translate_headers_list(*header_list*)

this function is used to translate table headers or generally, all texts in and list, to ui default language

Parameters
header_list – a list of texts that will be translated

Returns
header_list: translated list of texts

translate_ui()

This function translate ui to selected language in settings page

Returns
None

3.3.3.11 setting_api module

class `oxin.setting_api.`**`API`**(*ui*)

Bases: `object`

the API class has the main functionalities of oxin setting app, it takes as input the ui object, and other ui objects like login window, alert window and notification windows are initialized in this class

Parameters
`ui` – the ui file of the app

Returns
None

`add_defect`(*default_defect={}*)

this function is used to add a new defect returned from ui to database. it is also used to add/update edited defect on dataabse

Parameters
`default_defect` – (dict, optional) if not empty, it is used as new defect to add to database, else the new defect info is returned from ui. Defaults to {}.

Returns
None

`add_defect_group`(*default_defectgroup={}*)

this function is used to add a defect group returned from ui/user to database.

Parameters
`default_defectgroup` – (dict, optional) if not empty, it is used as input to add to database if not, the info returned from ui is used. Defaults to {}.

Returns
None

`add_user`(*default_user={}*)

this function is used to add new user to database. the user info is returned from ui, and used as input to add to database

Parameters
`default_user` – (dict, optional) if not empty, this dict is use as input to add to database. if empty, new user_infoes are get from ui

Returns
None

`apply_calibration_on_image`(*image*)

this function is used to apply soft-calibration on image and then update results on ui

Parameters
`image` – (_type_) input calibration image from camera

Returns
None

`apply_changed_appearance_params`(*mode='appearance'*)

this functino is used to apply returned setting parameters in setting page, to app/database according to mode. we can select which parameters to apply/set

Parameters
mode – (str, optional): it is used to select which parameters to apply/set. Defaults to 'appearance'. 'appearance': apply appearance params like font, color or … 'calibration': apply calibration params 'imageprocessing': apply image preprocessing parasms 'multitasking': 'defects':
Returns
None

button_connector()
this function is used to connect ui buttons to their functions
Returns
None

check_all_plc_parms()
this function is used to check all plc logic pathes values
Returns
values: a dict of plc values

check_plc_parms(*name*)
this function is used to check/get value of a path on plc
Parameters
name – (_type_) check botton name of the path
Returns
value: value stored in path

check_storage_status()
this function is used to check storage statues
Returns
None

confirm_yes()
this function is the event for confirm window yes button, accoarding to message of the confirm window, the function decides to take right action
Returns
None

connect_dissconnect_to_camera(*calibration=False*)
this functon is used to connect/disconnect to camera
Parameters
calibration – a boolean determining if the current page is calibration page
Returns
None

connect_plc()
this function is used to connect to plc
Returns
None

control_list_image(*input_img_path*)
this function is used to load image procesing directory contatiing images
Parameters
input_img_path (*str*) – inpput image directory
Returns
None

`disconnect_camera_on_ui_change`()

this function is used to disconnect camera if any of camera parameters in camera seting page are changed, or stackwidjet current page change

Returns
None

`disconnect_plc`(*on_close=False*)

this function is used to disconnect from plc

Parameters
`on_close` – (bool, optional) a boolean deermining if function is called on app close. Defaults to False.

`edit_defects`(*defect_group=False*)

this function is used to edit selected defect/defect-group and change its parameters

Parameters
`defect_group` – (bool, optional) a boolean determining whather to edit defect-group. Defaults to False.

Returns
None

`filter_defects`(*defect_group=False*)

this function is used to filter/search in defect table

Parameters
`defect_group` – (bool, optional) a boolean determining whather to search in defect-groups. Defaults to False.

Returns
None

`force_clear_camera_live_storage`()

this function is used to makes True the flag for force clearing storage

Returns
None

`image_processing_calibration`(*params_changed=False*)

this function is used to apply image processing algo on input image

Parameters
`params_changed` – (bool, optional) a boolean to detemine if algo params changed. Defaults to False.

Returns
None

`load_appearance_params_on_start`(*mainsetting_page=False*)

this function is used to load appearance params from database and apply to program on start-up or function call

Parameters
`mainsetting_page` – (bool, optional) a boolean determining wheather on mainsetting page or not. Defaults to False.

Returns
None

`load_camera_params_from_db_to_UI`()

this function is used every time a camera is selected in camera settings page, and tries to load camera settings and parameters of that camera from database. at every camera selection, the previous camera will disconnected if it is connected

Returns
None

`load_plc_parms`()
this function is used to load plc params from database, and set to ui plc page
Returns
resault: a boolean determining if params loaded from database

`next_image_precessing`()
this function is used to load next image for image processing calibration
Returns
None

`on_close_operations`()
this function is used to check/do some functions before closing the app
Returns
None

`previous_image_precessing`()
this function is used to load prev image for image processing calibration
Returns
None

`refresh_dashboard_page`()
this function is used to do some tasks that are related to dashboard page. the taks are almost the dashboard parameters
Returns
None

`refresh_defects_table`(*only_defects=False*, *only_defect_groups=False*)
this function is used to refresh defect/defect-group tables from database to ui tables
Parameters
- **`only_defects`** – a boolean determining only update defect table
- **`only_defect_groups`** – a boolean determining only update defect-groups table

Returns
None

`refresh_storege_page`(*only_chart=False*)
this function is used to refresh storage page
Parameters
`only_chart` – (bool, optional) if true, only the storage chart is updated. Defaults to False.

`refresh_users_table`()
this function is used to refresh users table on ui
Returns
None

`remove_defects`(*defect_group=False*)
this function is used to remove selected defects/defect groups from database
Parameters
`defect_group` – (bool, optional) a boolean determining wheather to remove defect_groups or not. Defaults to False.
Returns
None

`remove_users``()`

this function is used to remove selected users in ui users table, from database

Returns
None

`run_storage_check_timer``(`*storage_check_interval=60*, *stop=False*`)`

this function is used to initailize and run timer for checking storage statues

Parameters
- **`storage_check_interval`** – (int, optional) check interval (in seconds). Defaults to 60.
- **`stop`** – (bool, optional) a boolean determining to stop the timer. Defaults to False.

Returns
None

`save_changed_calibration_params``()`

this function is used to update camera calibration params to database. the input params are returned from ui

Returns
None

`save_changed_camera_params``(`*apply_to_multiple=False*`)`

save input camera parameters entered on UI camera setting page to database

Parameters
`apply_to_multiple` – a boolean determining wheter apply settings to multiple cameras or only current camera

Returns
None

`save_image_processing_parms``()`

this function is used to save image processing params from Miss.Abtahi algo to database

Returns
None

`save_plc_ip``()`

this function is used to get plc ip from ui and update on database

Returns
None

`save_plc_parms``()`

this function is used to save plc params to database

Returns
None

`select_image_procesing_directory``()`

this function is used to select image processing drectory contaiing images to fix image processing params with

Returns
None

`set_plc_ip_to_ui``()`

this function is used to get plc ip from database and set to ui

Returns
None

`set_plc_value`()

this function is used to update/set a path calue on plc

Returns
None

`show_camera_picture`(*calibration=False*)

this function is used to start image grabbing from camera, and update image on ui

Parameters
`calibration` – a boolean detrmining if current page is calibration or not

Returns
None

`show_related_defects`()

this function is used to show related defects to a selected defect-group

Returns
None

`tabledefectgroups_onHeaderClicked`(*logicalIndex*)

this function is used to sort items accoading to one column, if clicked on that column

Parameters
`logicalIndex` – (_type_) _description_

Returns
None

`tabledefects_onHeaderClicked`(*logicalIndex*)

this function is used to sort items accoading to one column, if clicked on that column

Parameters
`logicalIndex` – (_type_) _description_

Returns
None

`things_to_do_on_stackwidject_change`()

this function performs tasks needed to done on ui stackwidjet (page) change

Returns
None

`update_camera_live_storage_parms`()

this function is used to update/set defalt storage params returned from ui, to database

Returns
None

`update_path_plc`()

this function is used to get value of a path on plc, everytime pathes-combobox has changed

Returns
None

`update_plc_dashboard_parms`()

this function is used to update plc summary satues on dashboard

Returns
None

`write_parms`()

3.3.3.12 translate_ui module

oxin.translate_ui.**translate_ui**(*language='fa'*, *ui_file_path_en='main_window.ui'*, *ui_file_path_fa='main_window_fa.ui'*)

This function takes as input the default english version ui file, and translate it to input language

Parameters

- **language** – input language to translate ui to (in string), default is fa (stands for farsi/persian)
- **ui_file_path_en** – path of the default english ui file (in string)
- **ui_file_path_fa** – path of the output translated ui file to save (in string)

Returns
None

4

Others

5 Indices and tables

- genindex
- modindex

Python Module Index

Index