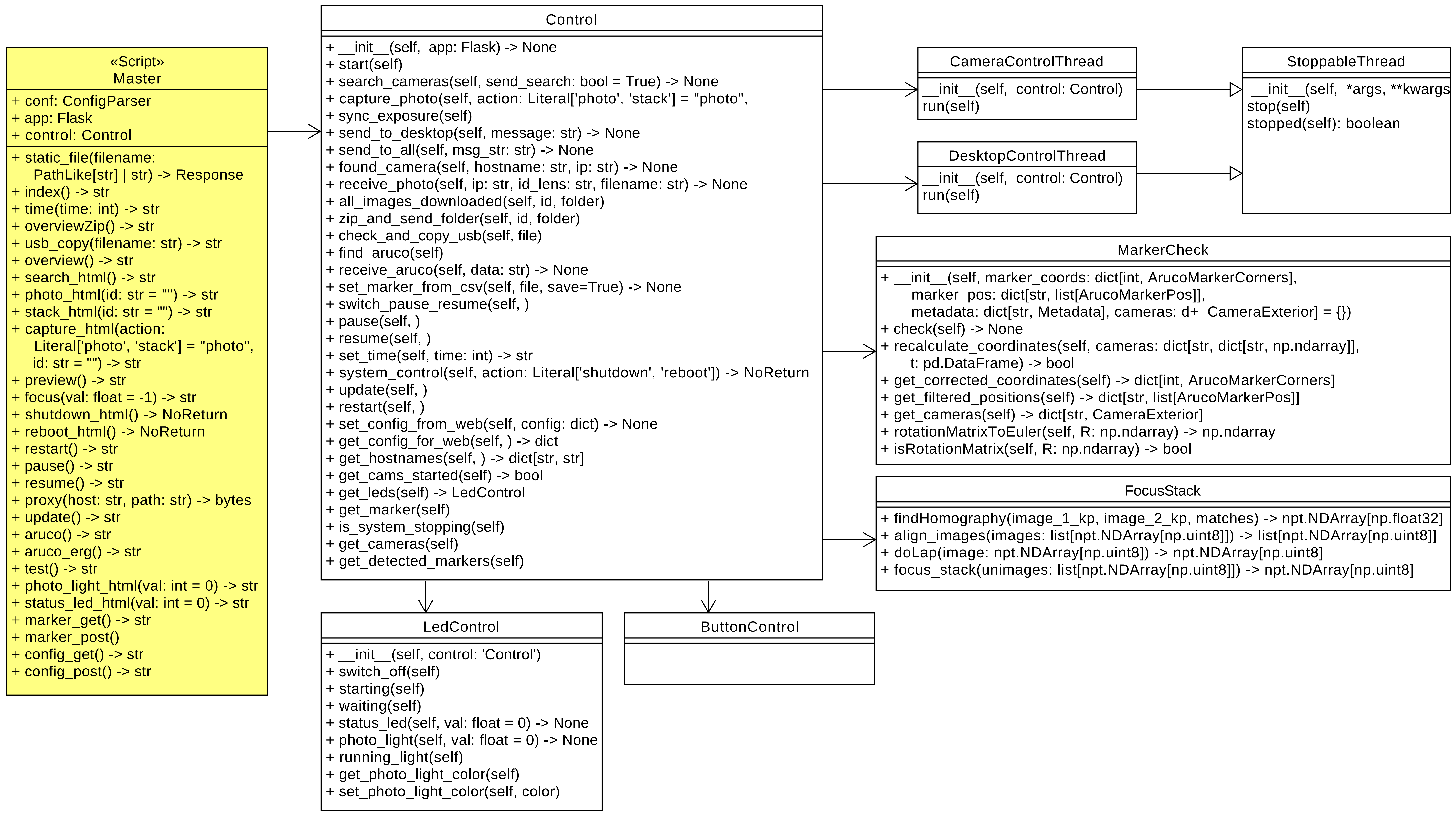

```
«Script»
Master
+ conf: ConfigParser
+ app: Flask
+ control: Control
+ static_file(filename:
    PathLike[str] | str) -> Response
+ index() -> str
+ time(time: int) -> str
+ overviewZip() -> str
+ usb_copy(filename: str) -> str
+ overview() -> str
+ search_html() -> str
+ photo_html(id: str = "") -> str
+ stack_html(id: str = "") -> str
+ capture_html(action:
    Literal['photo', 'stack'] = "photo",
    id: str = "") -> str
+ preview() -> str
+ focus(val: float = -1) -> str
+ shutdown_html() -> NoReturn
+ reboot_html() -> NoReturn
+ restart() -> str
+ pause() -> str
+ resume() -> str
+ proxy(host: str, path: str) -> bytes
+ update() -> str
+ aruco() -> str
+ aruco_erg() -> str
+ test() -> str
+ photo_light_html(val: int = 0) -> str
+ status_led_html(val: int = 0) -> str
+ marker_get() -> str
+ marker_post()
+ config_get() -> str
+ config_post() -> str
```

```
Control
+ __init__(self,  app: Flask) -> None
+ start(self)
+ search_cameras(self, send_search: bool = True) -> None
+ capture_photo(self, action: Literal['photo', 'stack'] = "photo",
+ sync_exposure(self)
+ send_to_desktop(self, message: str) -> None
+ send_to_all(self, msg_str: str) -> None
+ found_camera(self, hostname: str, ip: str) -> None
+ receive_photo(self, ip: str, id_lens: str, filename: str) -> None
+ all_images_downloaded(self, id, folder)
+ zip_and_send_folder(self, id, folder)
+ check_and_copy_usb(self, file)
+ find_aruco(self)
+ receive_aruco(self, data: str) -> None
+ set_marker_from_csv(self, file, save=True) -> None
+ switch_pause_resume(self, )
+ pause(self, )
+ resume(self, )
+ set_time(self, time: int) -> str
+ system_control(self, action: Literal['shutdown', 'reboot']) -> NoReturn
+ update(self, )
+ restart(self, )
+ set_config_from_web(self, config: dict) -> None
+ get_config_for_web(self, ) -> dict
+ get_hostnames(self, ) -> dict[str, str]
+ get_cams_started(self) -> bool
+ get_leds(self) -> LedControl
+ get_marker(self)
+ is_system_stopping(self)
+ get_cameras(self)
+ get_detected_markers(self)
```

```
CameraControlThread
__init__(self,  control: Control)
run(self)
```

```
DesktopControlThread
__init__(self,  control: Control)
run(self)
```

```
StoppableThread
 __init__(self,  *args, **kwargs
stop(self)
stopped(self): boolean
```

```
MarkerCheck
+ __init__(self, marker_coords: dict[int, ArucoMarkerCorners],
      marker_pos: dict[str, list[ArucoMarkerPos]],
      metadata: dict[str, Metadata], cameras: d+  CameraExterior] = {})
+ check(self) -> None
+ recalculate_coordinates(self, cameras: dict[str, dict[str, np.ndarray]],
      t: pd.DataFrame) -> bool
+ get_corrected_coordinates(self) -> dict[int, ArucoMarkerCorners]
+ get_filtered_positions(self) -> dict[str, list[ArucoMarkerPos]]
+ get_cameras(self) -> dict[str, CameraExterior]
+ rotationMatrixToEuler(self, R: np.ndarray) -> np.ndarray
+ isRotationMatrix(self, R: np.ndarray) -> bool
```

```
FocusStack
+ findHomography(image_1_kp, image_2_kp, matches) -> npt.NDArray[np.float32]
+ align_images(images: list[npt.NDArray[np.uint8]]) -> list[npt.NDArray[np.uint8]]
+ doLap(image: npt.NDArray[np.uint8]) -> npt.NDArray[np.uint8]
+ focus_stack(unimages: list[npt.NDArray[np.uint8]]) -> npt.NDArray[np.uint8]
```

```
LedControl
+ __init__(self, control: 'Control')
+ switch_off(self)
+ starting(self)
+ waiting(self)
+ status_led(self, val: float = 0) -> None
+ photo_light(self, val: float = 0) -> None
+ running_light(self)
+ get_photo_light_color(self)
+ set_photo_light_color(self, color)
```

```
ButtonControl
```