Dưới đây là giải thích chi tiết về toàn bộ dự án dựa trên file `main.py` bạn cung cấp. Đây là một ứng dụng kiểm tra, giám sát và ghi nhận kết quả kiểm tra sản phẩm bằng AI, sử dụng giao diện đồ họa PyQt5, camera, AI model (YOLO), điều khiển thiết bị ngoại vi (IO, đèn, scanner, v.v.), và lưu trữ dữ liệu vào cơ sở dữ liệu SQLite.

---

## 1. \*\*Tổng quan chức năng\*\*

- \*\*Giao diện người dùng (UI):\*\*

Sử dụng PyQt5 để tạo giao diện chính, các tab chức năng (Auto, Teaching, Data, IO Monitor), các nút điều khiển, bảng hiển thị kết quả, log, v.v.

- \*\*Kết nối thiết bị ngoại vi:\*\*

- \*\*Camera:\*\* Mở, đóng, lấy ảnh từ camera công nghiệp hoặc webcam.

- \*\*Đèn chiếu sáng:\*\* Điều khiển bật/tắt, chỉnh cường độ qua cổng COM.

- \*\*IO Controller:\*\* Điều khiển các cổng xuất/nhập (input/output) để tương tác với dây chuyền sản xuất.

- \*\*Máy quét mã vạch (Scanner):\*\* Đọc mã sản phẩm, xác nhận nhân viên.

- \*\*Giao tiếp với server:\*\* Kết nối tới server để gửi/nhận dữ liệu.

- \*\*AI & Xử lý ảnh:\*\*

- Sử dụng mô hình YOLO để nhận diện vật thể trên ảnh sản phẩm.

- Định nghĩa vùng kiểm tra (ROI) trên ảnh.

- So sánh kết quả nhận diện với cấu hình chuẩn (số lượng, vị trí, nhãn vật liệu).

- \*\*Lưu trữ & truy xuất dữ liệu:\*\*

- Lưu kết quả kiểm tra, ảnh gốc, ảnh kết quả, mã sản phẩm, trạng thái vào database SQLite.

- Tìm kiếm, lọc, xem lại lịch sử kiểm tra.

- \*\*Ghi log & báo lỗi:\*\*

- Ghi log các sự kiện, trạng thái, lỗi.

- Hiển thị cảnh báo, xác nhận khi có lỗi hoặc kết quả không đạt.

---

## 2. \*\*Luồng hoạt động chính\*\*

### \*\*A. Chế độ Auto (Tự động kiểm tra sản phẩm)\*\*

1. \*\*Khởi động:\*\*

Người dùng nhấn "Start Auto", hệ thống kết nối các thiết bị (camera, IO, server, scanner, đèn...).

2. \*\*Chờ cảm biến:\*\*

Khi cảm biến phát hiện sản phẩm, hệ thống bật đèn, lấy ảnh từ camera.

3. \*\*Nhận diện AI:\*\*

Ảnh được đưa vào model YOLO, nhận diện các vật thể, so sánh với cấu hình chuẩn (số lượng, nhãn, vị trí).

4. \*\*Xử lý kết quả:\*\*

- Nếu đạt: Ghi nhận PASS, xuất tín hiệu IO tương ứng, lưu log, gửi dữ liệu về server.

- Nếu không đạt: Hiển thị cảnh báo, yêu cầu xác nhận lại (có thể cần quét thẻ nhân viên để xác nhận).

5. \*\*Lưu trữ:\*\*

Lưu ảnh gốc, ảnh kết quả, mã sản phẩm, trạng thái vào database.

6. \*\*Lặp lại:\*\*

Quay lại bước chờ cảm biến cho sản phẩm tiếp theo.

### \*\*B. Chế độ Teaching (Dạy AI, cấu hình mô hình)\*\*

- Người dùng có thể:

- Thêm/xóa/lưu mô hình kiểm tra.

- Định nghĩa vùng kiểm tra (ROI) trên ảnh.

- Chọn model AI, file nhãn, cấu hình thông số.

- Mở camera, chụp ảnh, kiểm tra thử, xem kết quả nhận diện.

### \*\*C. Chế độ Data (Xem lại dữ liệu)\*\*

- Xem, lọc, tìm kiếm lịch sử kiểm tra sản phẩm theo thời gian, trạng thái, model, từ khóa.

- Xem lại ảnh gốc, ảnh kết quả, mã sản phẩm, lỗi.

### \*\*D. IO Monitor\*\*

- Hiển thị trạng thái các cổng input/output.

- Đổi màu label tương ứng khi có tín hiệu vào/ra.

---

## 3. \*\*Các thành phần chính trong code\*\*

### \*\*A. Lớp MainWindow (QMainWindow)\*\*

- \*\*Khởi tạo:\*\*

Kết nối các nút, signal-slot, khởi tạo các thành phần UI, load dữ liệu ban đầu.

- \*\*Các hàm điều khiển thiết bị:\*\*

- `open\_camera\_teaching`, `close\_camera\_teaching`, `start\_camera\_teaching`, `stop\_camera\_teaching`

- `open\_light\_teaching`, `set\_light\_value`, `update\_light\_value`

- `connect\_io\_controller`, `close\_io\_controller`, `set\_io\_result`

- `connect\_scanner`, `close\_scanner`, `read\_serial\_data`

- \*\*Các hàm xử lý AI:\*\*

- `load\_model`, `detect\_objects`, `is\_inside\_any\_shape`

- \*\*Các hàm xử lý dữ liệu:\*\*

- `load\_database`, `find\_database`, `find\_series\_number`

- `write\_log\_async` (ghi log vào database và lưu ảnh)

- \*\*Các hàm UI:\*\*

- `load\_classes\_ai`, `load\_result\_classes`, `set\_shapes`, `get\_shapes`

- `update\_label\_status`, `update\_label\_result`, `set\_text\_label`

- \*\*Các hàm xác nhận, cảnh báo:\*\*

- `show\_sn\_failed`, `hide\_sn\_failed`, `show\_message\_scan\_gen`, `hide\_message\_scan\_gen`, `recheck\_result`

### \*\*B. Các lớp, module phụ trợ\*\*

- \*\*Camera, IO, Light Controller, Scanner:\*\*

Được import từ các file trong thư mục `libs` và `cameras`, cung cấp API điều khiển thiết bị.

- \*\*Logger:\*\*

Ghi log sự kiện, trạng thái, lỗi.

- \*\*DatabaseManager:\*\*

Quản lý kết nối, truy vấn, thêm dữ liệu vào SQLite.

- \*\*HandleJsonPBA:\*\*

Đọc/ghi cấu hình model từ file JSON.

---

## 4. \*\*Luồng dữ liệu & xử lý AI\*\*

- \*\*Cấu hình model:\*\*

Được lưu dưới dạng JSON, gồm thông tin về vùng kiểm tra, model AI, thông số phần cứng, v.v.

- \*\*File nhãn (classes):\*\*

Định nghĩa các nhãn vật liệu, số lượng cần kiểm tra.

- \*\*Khi kiểm tra:\*\*

- Ảnh được lấy từ camera, đưa vào model YOLO.

- Kết quả nhận diện được so sánh với cấu hình chuẩn (số lượng, nhãn, vị trí).

- Nếu thiếu vật liệu, sai vị trí, hoặc số lượng không đúng → FAIL.

- Nếu đạt tất cả điều kiện → PASS.

- Kết quả, ảnh, mã sản phẩm được lưu vào database.

---

## 5. \*\*Các điểm nổi bật\*\*

- \*\*Tự động hóa kiểm tra sản phẩm bằng AI, tích hợp với dây chuyền sản xuất.\*\*

- \*\*Giao diện trực quan, dễ sử dụng, hỗ trợ nhiều chế độ (auto, teaching, data, IO monitor).\*\*

- \*\*Có khả năng xác nhận lại khi kết quả FAIL (quét thẻ nhân viên).\*\*

- \*\*Lưu trữ đầy đủ lịch sử kiểm tra, hỗ trợ truy xuất, tìm kiếm, xem lại ảnh.\*\*

- \*\*Có thể mở rộng, tùy biến cấu hình model, vùng kiểm tra, thiết bị ngoại vi.\*\*

---

## 6. \*\*Công nghệ sử dụng\*\*

- \*\*Python 3\*\*

- \*\*PyQt5\*\*: Giao diện người dùng

- \*\*OpenCV, NumPy\*\*: Xử lý ảnh

- \*\*YOLO (ultralytics)\*\*: Nhận diện vật thể AI

- \*\*SQLAlchemy, sqlite3\*\*: Quản lý database

- \*\*Serial, threading\*\*: Điều khiển thiết bị ngoại vi, đa luồng

- \*\*QSettings\*\*: Lưu trạng thái giao diện, cấu hình

---

## 7. \*\*Tóm tắt\*\*

\*\*Dự án này là một hệ thống kiểm tra sản phẩm tự động bằng AI, tích hợp với dây chuyền sản xuất, có giao diện trực quan, khả năng cấu hình linh hoạt, lưu trữ và truy xuất dữ liệu đầy đủ, hỗ trợ xác nhận lại khi có lỗi, và dễ dàng mở rộng cho nhiều loại sản phẩm, model AI, thiết bị ngoại vi khác nhau.\*\*

Nếu bạn cần giải thích chi tiết về từng hàm, từng module, hoặc muốn biết về cách mở rộng, tích hợp thêm thiết bị, hãy hỏi thêm nhé!

Dưới đây là giải thích chi tiết về các hàm chính trong file `main.py` của dự án. Các hàm này thực hiện các chức năng quan trọng như khởi tạo giao diện, điều khiển thiết bị, xử lý AI, lưu trữ dữ liệu, v.v.

---

## 1. \*\*Hàm `\_\_init\_\_`\*\*

```python

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.camera = Camera()

self.server = Server()

self.scanner = SerialReceiver()

self.send\_data\_com = SerialReceiver()

self.light = None

self.count\_product\_total = 0

self.count\_product\_ok = 0

self.count\_product\_ng = 0

self.is\_connect\_server = False

self.is\_open\_camera = False

self.is\_connect\_io\_controller = False

self.is\_connect\_scanner = False

self.is\_connect\_send\_data = False

self.is\_confirm\_ret = True

self.confirm\_lock = threading.Lock()

self.is\_check\_sn = False

self.is\_check\_gen\_id = False

self.is\_sensor\_on = False

self.active\_output = OutPorts.Out\_3

self.io\_controller = None

self.trigger\_on = False

self.handle\_file\_json = HandleJsonPBA()

self.input\_labels = [self.label\_input\_1, self.label\_input\_2, self.label\_input\_3, self.label\_input\_4,

self.label\_input\_5, self.label\_input\_6, self.label\_input\_7, self.label\_input\_8]

self.main\_logger = Logger('MainWindow')

self.db = DatabaseManager(db\_name=f'res/Database/{self.line\_edit\_database\_path.text()}')

self.log\_table = self.db.create\_table("users", Model=String, SeriesNumber= String, Source=String, Destination=String,

Return=String, Error=String)

```

- \*\*Mục đích:\*\*

Khởi tạo giao diện, các biến trạng thái, kết nối thiết bị, logger, database.

- \*\*Chi tiết:\*\*

- `self.setupUi(self)`: Load giao diện từ file UI.

- `self.camera`, `self.server`, `self.scanner`, `self.send\_data\_com`: Khởi tạo các đối tượng điều khiển thiết bị.

- `self.count\_product\_total`, `self.count\_product\_ok`, `self.count\_product\_ng`: Biến đếm số lượng sản phẩm.

- `self.is\_connect\_server`, `self.is\_open\_camera`, v.v.: Biến trạng thái kết nối thiết bị.

- `self.confirm\_lock`: Lock để đồng bộ hóa khi xác nhận kết quả.

- `self.input\_labels`: Danh sách label hiển thị trạng thái input.

- `self.main\_logger`: Logger ghi log sự kiện.

- `self.db`: Kết nối database, tạo bảng `users` để lưu kết quả kiểm tra.

---

## 2. \*\*Hàm `init\_main\_window`\*\*

```python

def init\_main\_window(self):

self.setWindowIcon(QIcon("res/Style/icon.ico"))

self.setWindowTitle('AI Vision')

self.image\_dst = Canvas()

self.image\_bin = Canvas()

self.layout\_image\_dst\_canvas.addWidget(WindowCanvas(self.image\_dst))

self.layout\_image\_bin\_canvas.addWidget(WindowCanvas(self.image\_bin))

self.action\_group = QActionGroup(self)

self.action\_group.setExclusive(True)

self.action\_group.addAction(self.actionLight\_mode)

self.action\_group.addAction(self.actionDark\_mode)

self.action\_group.addAction(self.actionAnime\_mode)

self.action\_group.addAction(self.actionWood\_mode)

with open("res/Style/light\_mode.qss", "r", encoding="utf-8") as file:

app.setStyleSheet(file.read())

self.statusBar().addPermanentWidget(self.progressBar)

self.progressBar.setVisible(False)

self.menuView.addAction(self.dockWidget\_2.toggleViewAction())

self.menuView.addAction(self.dockWidget\_3.toggleViewAction())

actions = self.menuView.actions()

if actions:

ui\_action = actions[0] # Action từ giao diện (ví dụ là action đầu)

self.menuView.removeAction(ui\_action) # Xóa nó khỏi vị trí ban đầu

self.menuView.addAction(ui\_action) # Thêm lại xuống cuối

self.is\_resetting = False

self.log\_model = QStandardItemModel()

self.list\_log\_view.setModel(self.log\_model)

```

- \*\*Mục đích:\*\*

Khởi tạo giao diện chính, canvas hiển thị ảnh, menu, thanh trạng thái, log.

- \*\*Chi tiết:\*\*

- `self.setWindowIcon`, `self.setWindowTitle`: Đặt icon, tiêu đề cửa sổ.

- `self.image\_dst`, `self.image\_bin`: Canvas hiển thị ảnh gốc, ảnh kết quả.

- `self.action\_group`: Nhóm các action chuyển đổi theme (light, dark, anime, wood).

- `app.setStyleSheet`: Load file CSS cho giao diện.

- `self.progressBar`: Thanh tiến trình, mặc định ẩn.

- `self.log\_model`: Model hiển thị log.

---

## 3. \*\*Hàm `init\_ui\_auto`\*\*

```python

def init\_ui\_auto(self):

self.auto\_canvas = Canvas()

self.layout\_auto\_canvas.addWidget(WindowCanvas(self.auto\_canvas))

self.but\_stop\_auto.setDisabled(True)

```

- \*\*Mục đích:\*\*

Khởi tạo giao diện tab Auto, canvas hiển thị ảnh, nút dừng.

- \*\*Chi tiết:\*\*

- `self.auto\_canvas`: Canvas hiển thị ảnh trong chế độ Auto.

- `self.but\_stop\_auto.setDisabled(True)`: Nút dừng mặc định bị vô hiệu hóa.

---

## 4. \*\*Hàm `init\_ui\_teaching`\*\*

```python

def init\_ui\_teaching(self):

self.is\_open\_camera = False

self.is\_showing\_camera = False

self.file\_open\_image = None

self.teaching\_canvas = Canvas()

self.timer = QTimer()

self.timer.timeout.connect(self.read\_serial\_data)

set\_roi = QAction(QIcon("res/Style/icon.ico"), "Set Roi", self)

self.teaching\_canvas.contextMenu.addSeparator()

self.teaching\_canvas.contextMenu.addAction(set\_roi)

self.layout\_teaching\_canvas.addWidget(WindowCanvas(self.teaching\_canvas))

self.but\_start\_camera\_teaching.setDisabled(True)

self.but\_open\_camera\_teaching.setProperty("status", "Open")

self.but\_start\_camera\_teaching.setProperty("status", "Open")

self.but\_open\_light.setProperty("status", "Open")

self.but\_open\_io\_controller.setProperty("status", "Open")

self.but\_connect\_server.setProperty("status", "Open")

self.but\_connect\_scanner.setProperty("status", "Open")

self.but\_open\_com\_send\_data.setProperty("status", "Open")

```

- \*\*Mục đích:\*\*

Khởi tạo giao diện tab Teaching, canvas hiển thị ảnh, menu ngữ cảnh, nút điều khiển.

- \*\*Chi tiết:\*\*

- `self.teaching\_canvas`: Canvas hiển thị ảnh trong chế độ Teaching.

- `self.timer`: Timer đọc dữ liệu từ scanner.

- `set\_roi`: Action đặt vùng kiểm tra (ROI) trên canvas.

- Các nút điều khiển (`but\_open\_camera\_teaching`, `but\_start\_camera\_teaching`, v.v.) được đặt trạng thái mặc định.

---

## 5. \*\*Hàm `init\_ui\_data`\*\*

```python

def init\_ui\_data(self):

self.file\_path\_database = None

self.data\_image\_input\_canvas = Canvas()

self.data\_image\_output\_canvas = Canvas()

self.layout\_image\_input\_canvas.addWidget(WindowCanvas(self.data\_image\_input\_canvas))

self.layout\_image\_output\_canvas.addWidget(WindowCanvas(self.data\_image\_output\_canvas))

self.but\_find\_database.setDisabled(True)

header = self.table\_widget\_database.verticalHeader()

header.setDefaultAlignment(Qt.AlignCenter)

```

- \*\*Mục đích:\*\*

Khởi tạo giao diện tab Data, canvas hiển thị ảnh gốc, ảnh kết quả, bảng dữ liệu.

- \*\*Chi tiết:\*\*

- `self.data\_image\_input\_canvas`, `self.data\_image\_output\_canvas`: Canvas hiển thị ảnh gốc, ảnh kết quả.

- `self.but\_find\_database.setDisabled(True)`: Nút tìm kiếm database mặc định bị vô hiệu hóa.

- `header.setDefaultAlignment(Qt.AlignCenter)`: Căn giữa header của bảng dữ liệu.

---

## 6. \*\*Hàm `init\_ui\_io\_monitor`\*\*

```python

def init\_ui\_io\_monitor(self):

self.input\_labels = [self.label\_input\_1, self.label\_input\_2, self.label\_input\_3, self.label\_input\_4,

self.label\_input\_5, self.label\_input\_6, self.label\_input\_7, self.label\_input\_8]

for label in self.input\_labels:

label.setStyleSheet("background-color: gray; color: white; padding: 5px;")

```

- \*\*Mục đích:\*\*

Khởi tạo giao diện tab IO Monitor, hiển thị trạng thái các cổng input.

- \*\*Chi tiết:\*\*

- `self.input\_labels`: Danh sách label hiển thị trạng thái input.

- `label.setStyleSheet`: Đặt màu nền, màu chữ, padding cho label.

---

## 7. \*\*Hàm `on\_click\_but\_start\_auto`\*\*

```python

def on\_click\_but\_start\_auto(self):

self.combo\_box\_model\_name\_teaching.setCurrentIndex(self.combo\_box\_model\_name\_auto.currentIndex())

model\_name = f'Model/{self.combo\_box\_model\_name\_auto.currentText()}'

self.config\_auto = self.get\_config\_auto(model\_name)

self.on\_click\_but\_connect\_server()

self.on\_click\_but\_connect\_scanner()

self.on\_click\_but\_open\_com\_send\_data()

self.on\_click\_but\_connect\_io\_controller()

self.on\_click\_but\_open\_camera\_teaching()

if not self.is\_open\_camera or not self.is\_connect\_server or not self.is\_connect\_scanner or not self.is\_connect\_io\_controller or not self.is\_connect\_send\_data:

self.on\_click\_but\_close\_camera\_teaching()

self.on\_click\_but\_close\_io\_controller()

self.on\_click\_but\_close\_server()

self.on\_click\_but\_close\_scanner()

self.on\_click\_but\_close\_com\_send\_data()

return

else:

self.set\_shapes(self.auto\_canvas, self.config\_auto.shapes)

self.but\_start\_auto.setDisabled(True)

self.but\_stop\_auto.setDisabled(False)

self.start\_process()

```

- \*\*Mục đích:\*\*

Bắt đầu quá trình kiểm tra tự động, kết nối các thiết bị, load cấu hình, khởi động luồng xử lý.

- \*\*Chi tiết:\*\*

- `self.combo\_box\_model\_name\_teaching.setCurrentIndex`: Đồng bộ model giữa tab Teaching và Auto.

- `self.config\_auto = self.get\_config\_auto(model\_name)`: Load cấu hình model từ file JSON.

- Kết nối các thiết bị (server, scanner, IO, camera, đèn).

- Nếu không kết nối được, đóng tất cả thiết bị, thoát.

- Nếu kết nối thành công, đặt vùng kiểm tra (shapes) lên canvas, vô hiệu hóa nút Start, bật nút Stop.

- `self.start\_process()`: Khởi động luồng xử lý tự động.

---

## 8. \*\*Hàm `on\_click\_but\_stop\_auto`\*\*

```python

def on\_click\_but\_stop\_auto(self):

self.stop\_process()

self.set\_disable\_auto(False)

self.on\_click\_but\_close\_server()

self.on\_click\_but\_close\_io\_controller()

self.on\_click\_but\_close\_camera\_teaching()

self.on\_click\_but\_close\_scanner()

self.on\_click\_but\_close\_com\_send\_data()

self.but\_start\_auto.setDisabled(False)

self.but\_stop\_auto.setDisabled(True)

```

- \*\*Mục đích:\*\*

Dừng quá trình kiểm tra tự động, đóng các thiết bị, reset trạng thái nút.

- \*\*Chi tiết:\*\*

- `self.stop\_process()`: Dừng luồng xử lý tự động.

- `self.set\_disable\_auto(False)`: Bật lại các nút điều khiển.

- Đóng các thiết bị (server, IO, camera, scanner, đèn).

- Reset trạng thái nút Start, Stop.

---

## 9. \*\*Hàm `on\_click\_but\_test\_auto`\*\*

```python

def on\_click\_but\_test\_auto(self):

self.count\_product\_total = 0

self.count\_product\_ok = 0

self.count\_product\_ng = 0

self.label\_ok.setText('0')

self.label\_ng.setText('0')

self.label\_total.setText('0')

self.label\_rate.setText('0')

```

- \*\*Mục đích:\*\*

Reset số lượng sản phẩm, tỷ lệ đạt/không đạt.

- \*\*Chi tiết:\*\*

- Reset các biến đếm `count\_product\_total`, `count\_product\_ok`, `count\_product\_ng`.

- Reset text trên các label hiển thị số lượng, tỷ lệ.

---

## 10. \*\*Hàm `on\_click\_but\_add\_model`\*\*

```python

def on\_click\_but\_add\_model(self):

name\_model = self.combo\_box\_model\_name\_teaching.currentText()

if name\_model and self.combo\_box\_model\_name\_teaching.findText(name\_model) == -1:

reply = QMessageBox.question(self, 'Question', f'Bạn có muốn thêm mô hình {name\_model} không',

QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No)

if reply == QMessageBox.StandardButton.No:

return

self.combo\_box\_model\_name\_teaching.addItem(name\_model)

self.combo\_box\_model\_name\_auto.addItem(name\_model)

self.combo\_box\_model\_name\_data.addItem(name\_model)

config = self.get\_config\_from\_ui()

config\_dict = json.loads(json.dumps(config, default=lambda x: vars(x)))

self.handle\_file\_json.add(name\_model, config\_dict)

QMessageBox.information(self, 'Information', f'Mô hình {name\_model} đã được thêm thành công',

QMessageBox.StandardButton.Close)

self.combo\_box\_model\_name\_teaching.clearEditText()

elif name\_model == '':

QMessageBox.warning(self, 'Warning', f'Chưa nhập tên mô hình', QMessageBox.StandardButton.Close)

else:

QMessageBox.warning(self, 'Warning', f'Mô hình {name\_model} đã tồn tại', QMessageBox.StandardButton.Close)

```

- \*\*Mục đích:\*\*

Thêm mô hình mới vào danh sách, lưu cấu hình vào file JSON.

- \*\*Chi tiết:\*\*

- Lấy tên mô hình từ combo box.

- Kiểm tra tên mô hình đã tồn tại chưa.

- Nếu chưa tồn tại, hiện hộp thoại xác nhận.

- Nếu đồng ý, thêm tên mô hình vào các combo box, lưu cấu hình vào file JSON.

- Hiện thông báo thành công, xóa text trong combo box.

- Nếu tên mô hình trống hoặc đã tồn tại, hiện cảnh báo.

---

## 11. \*\*Hàm `on\_click\_but\_delete\_model`\*\*

```python

def on\_click\_but\_delete\_model(self):

name\_model = self.combo\_box\_model\_name\_teaching.currentText()

index\_name\_model = self.combo\_box\_model\_name\_teaching.findText(name\_model)

if index\_name\_model == -1:

QMessageBox.warning(self, 'Warning', f'Mô hình {name\_model} không tồn tại',

QMessageBox.StandardButton.Close)

return

elif name\_model == 'Default':

QMessageBox.warning(self, 'Warning', f'Mô hình {name\_model} là mặc định, không thể xóa',

QMessageBox.StandardButton.Close)

return

reply = QMessageBox.question(self, 'Delete', f'Bạn có muốn xóa {name\_model} không?',

QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No)

if reply == QMessageBox.StandardButton.No:

return

self.handle\_file\_json.delete(name\_model)

index = self.combo\_box\_model\_name\_teaching.currentIndex()

self.combo\_box\_model\_name\_auto.removeItem(index)

self.combo\_box\_model\_name\_teaching.removeItem(index)

QMessageBox.information(self, 'Information', f'Mô hình {name\_model} đã được xóa',

QMessageBox.StandardButton.Close)

```

- \*\*Mục đích:\*\*

Xóa mô hình khỏi danh sách, xóa cấu hình từ file JSON.

- \*\*Chi tiết:\*\*

- Lấy tên mô hình từ combo box.

- Kiểm tra mô hình có tồn tại không, có phải mô hình mặc định không.

- Nếu không tồn tại hoặc là mô hình mặc định, hiện cảnh báo.

- Nếu tồn tại, hiện hộp thoại xác nhận xóa.

- Nếu đồng ý, xóa mô hình khỏi file JSON, xóa tên khỏi các combo box.

- Hiện thông báo thành công.

---

## 12. \*\*Hàm `on\_click\_save\_model`\*\*

```python

def on\_click\_save\_model(self):

name\_model = self.combo\_box\_model\_name\_teaching.currentText()

index\_name\_model = self.combo\_box\_model\_name\_teaching.findText(name\_model)

if index\_name\_model == -1:

QMessageBox.warning(self, 'Warning', f'Mô hình {name\_model} không tồn tại',

QMessageBox.StandardButton.Close)

return

config = self.get\_config\_from\_ui()

config\_dict = json.loads(json.dumps(config, default=lambda x: vars(x)))

reply = QMessageBox.question(self, 'Question', f'Bạn có muốn lưu mô hình {name\_model} không',

QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No)

if reply == QMessageBox.StandardButton.No:

return

self.handle\_file\_json.save(name\_model, config\_dict)

QMessageBox.information(self, 'Information', f'Mô hình {name\_model} đã thay đổi thành công',

QMessageBox.StandardButton.Close)

```

- \*\*Mục đích:\*\*

Lưu cấu hình mô hình hiện tại vào file JSON.

- \*\*Chi tiết:\*\*

- Lấy tên mô hình từ combo box.

- Kiểm tra mô hình có tồn tại không.

- Nếu không tồn tại, hiện cảnh báo.

- Nếu tồn tại, lấy cấu hình từ UI, chuyển đổi thành dict.

- Hiện hộp thoại xác nhận lưu.

- Nếu đồng ý, lưu cấu hình vào file JSON.

- Hiện thông báo thành công.

---

## 13. \*\*Hàm `on\_click\_but\_open\_camera\_teaching`\*\*

```python

def on\_click\_but\_open\_camera\_teaching(self):

if self.but\_open\_camera\_teaching.property("status") == "Open":

self.open\_camera\_teaching()

self.but\_open\_camera\_teaching.setProperty("status", "Close")

self.but\_open\_camera\_teaching.setText('Close')

self.style().unpolish(self.but\_open\_camera\_teaching)

self.style().polish(self.but\_open\_camera\_teaching)

self.style().unpolish(self.but\_start\_camera\_teaching)

self.style().polish(self.but\_start\_camera\_teaching)

self.but\_open\_camera\_teaching.clicked.disconnect()

self.but\_open\_camera\_teaching.clicked.connect(self.on\_click\_but\_close\_camera\_teaching)

```

- \*\*Mục đích:\*\*

Mở camera, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Open Camera".

- Nếu đang "Open", gọi `open\_camera\_teaching()`, đổi trạng thái nút thành "Close".

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (đóng camera).

---

## 14. \*\*Hàm `on\_click\_but\_close\_camera\_teaching`\*\*

```python

def on\_click\_but\_close\_camera\_teaching(self):

if self.but\_open\_camera\_teaching.property("status") == "Close":

self.close\_camera\_teaching()

self.but\_open\_camera\_teaching.setText('Open')

self.but\_open\_camera\_teaching.setProperty("status", "Open")

self.but\_start\_camera\_teaching.setProperty("status", "Open")

self.but\_start\_camera\_teaching.setText('Start')

self.style().unpolish(self.but\_open\_camera\_teaching)

self.style().polish(self.but\_open\_camera\_teaching)

self.style().unpolish(self.but\_start\_camera\_teaching)

self.style().polish(self.but\_start\_camera\_teaching)

self.but\_open\_camera\_teaching.clicked.disconnect()

self.but\_open\_camera\_teaching.clicked.connect(self.on\_click\_but\_open\_camera\_teaching)

```

- \*\*Mục đích:\*\*

Đóng camera, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Open Camera".

- Nếu đang "Close", gọi `close\_camera\_teaching()`, đổi trạng thái nút thành "Open".

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (mở camera).

---

## 15. \*\*Hàm `on\_click\_but\_start\_camera\_teaching`\*\*

```python

def on\_click\_but\_start\_camera\_teaching(self):

if self.but\_start\_camera\_teaching.property("status") == "Open":

self.but\_start\_camera\_teaching.setProperty("status", "Close")

self.but\_start\_camera\_teaching.setText('Stop')

self.start\_camera\_teaching()

else:

self.but\_start\_camera\_teaching.setText('Start')

self.but\_start\_camera\_teaching.setProperty("status", "Open")

self.stop\_camera\_teaching()

self.style().unpolish(self.but\_start\_camera\_teaching)

self.style().polish(self.but\_start\_camera\_teaching)

```

- \*\*Mục đích:\*\*

Bắt đầu/dừng hiển thị camera, thay đổi trạng thái nút.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Start Camera".

- Nếu đang "Open", đổi trạng thái thành "Close", gọi `start\_camera\_teaching()`.

- Nếu đang "Close", đổi trạng thái thành "Open", gọi `stop\_camera\_teaching()`.

- Cập nhật style nút.

---

## 16. \*\*Hàm `on\_click\_but\_open\_light`\*\*

```python

def on\_click\_but\_open\_light(self):

if self.but\_open\_light.property("status") == "Open":

self.but\_open\_light.setProperty("status", "Close")

self.but\_open\_light.setText('Close')

self.open\_light\_teaching()

elif self.but\_open\_light.property("status") == "Close":

self.set\_light\_value()

self.but\_open\_light.setText('Open')

self.but\_open\_light.setProperty("status", "Open")

self.style().unpolish(self.but\_open\_light)

self.style().polish(self.but\_open\_light)

```

- \*\*Mục đích:\*\*

Bật/tắt đèn, thay đổi trạng thái nút.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Open Light".

- Nếu đang "Open", đổi trạng thái thành "Close", gọi `open\_light\_teaching()`.

- Nếu đang "Close", đổi trạng thái thành "Open", gọi `set\_light\_value()` (tắt đèn).

- Cập nhật style nút.

---

## 17. \*\*Hàm `on\_click\_but\_connect\_io\_controller`\*\*

```python

def on\_click\_but\_connect\_io\_controller(self):

if self.but\_open\_io\_controller.property("status") == "Open":

self.but\_open\_io\_controller.setProperty("status", "Close")

self.but\_open\_io\_controller.setText('Close')

self.connect\_io\_controller()

self.style().unpolish(self.but\_open\_io\_controller)

self.style().polish(self.but\_open\_io\_controller)

self.but\_open\_io\_controller.clicked.disconnect()

self.but\_open\_io\_controller.clicked.connect(self.on\_click\_but\_close\_io\_controller)

```

- \*\*Mục đích:\*\*

Kết nối IO Controller, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Connect IO Controller".

- Nếu đang "Open", đổi trạng thái thành "Close", gọi `connect\_io\_controller()`.

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (đóng IO Controller).

---

## 18. \*\*Hàm `on\_click\_but\_close\_io\_controller`\*\*

```python

def on\_click\_but\_close\_io\_controller(self):

if self.but\_open\_io\_controller.property("status") == "Close":

self.but\_open\_io\_controller.setText('Connect')

self.but\_open\_io\_controller.setProperty("status", "Open")

self.close\_io\_controller()

self.style().unpolish(self.but\_open\_io\_controller)

self.style().polish(self.but\_open\_io\_controller)

self.but\_open\_io\_controller.clicked.disconnect()

self.but\_open\_io\_controller.clicked.connect(self.on\_click\_but\_connect\_io\_controller)

```

- \*\*Mục đích:\*\*

Đóng IO Controller, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Connect IO Controller".

- Nếu đang "Close", đổi trạng thái thành "Open", gọi `close\_io\_controller()`.

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (mở IO Controller).

---

## 19. \*\*Hàm `on\_click\_but\_connect\_server`\*\*

```python

def on\_click\_but\_connect\_server(self):

if self.but\_connect\_server.property("status") == "Open":

self.but\_connect\_server.setProperty("status", "Close")

self.but\_connect\_server.setText('Close')

self.connect\_server\_teaching()

self.style().unpolish(self.but\_connect\_server)

self.style().polish(self.but\_connect\_server)

self.but\_connect\_server.clicked.disconnect()

self.but\_connect\_server.clicked.connect(self.on\_click\_but\_close\_server)

```

- \*\*Mục đích:\*\*

Kết nối server, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Connect Server".

- Nếu đang "Open", đổi trạng thái thành "Close", gọi `connect\_server\_teaching()`.

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (đóng server).

---

## 20. \*\*Hàm `on\_click\_but\_close\_server`\*\*

```python

def on\_click\_but\_close\_server(self):

if self.but\_connect\_server.property("status") == "Close":

self.but\_connect\_server.setText('Open')

self.but\_connect\_server.setProperty("status", "Open")

self.close\_server\_teaching()

self.style().unpolish(self.but\_connect\_server)

self.style().polish(self.but\_connect\_server)

self.but\_connect\_server.clicked.disconnect()

self.but\_connect\_server.clicked.connect(self.on\_click\_but\_connect\_server)

```

- \*\*Mục đích:\*\*

Đóng server, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Connect Server".

- Nếu đang "Close", đổi trạng thái thành "Open", gọi `close\_server\_teaching()`.

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (mở server).

---

## 21. \*\*Hàm `on\_click\_but\_open\_image\_teaching`\*\*

```python

def on\_click\_but\_open\_image\_teaching(self):

if self.but\_start\_camera\_teaching.property("status") == "Close":

self.is\_showing\_camera = False

self.but\_start\_camera\_teaching.setProperty("status", "Open")

self.but\_start\_camera\_teaching.setText('Start')

self.style().unpolish(self.but\_start\_camera\_teaching)

self.style().polish(self.but\_start\_camera\_teaching)

options = QFileDialog.Options()

self.file\_open\_image, \_ = QFileDialog.getOpenFileName(self, "Chọn Ảnh", "",

"Images (\*.png \*.jpg \*.jpeg \*.bmp \*.gif)",

options=options)

if self.file\_open\_image:

self.teaching\_canvas.load\_pixmap(QPixmap(self.file\_open\_image))

```

- \*\*Mục đích:\*\*

Mở ảnh từ file, hiển thị lên canvas.

- \*\*Chi tiết:\*\*

- Nếu camera đang hiển thị, dừng camera.

- Hiện hộp thoại chọn file ảnh.

- Nếu chọn được ảnh, load ảnh lên canvas.

---

## 22. \*\*Hàm `on\_click\_but\_capture\_teaching`\*\*

```python

def on\_click\_but\_capture\_teaching(self):

if self.is\_showing\_camera:

self.file\_open\_image = self.mat.copy()

self.main\_logger.log\_image(self.combo\_box\_model\_name\_teaching.currentText(),

self.file\_open\_image, capture=True)

return

if self.is\_open\_camera:

self.file\_open\_image = self.camera.get\_frame()

self.main\_logger.log\_image(self.combo\_box\_model\_name\_teaching.currentText(),

self.file\_open\_image, capture=True)

self.teaching\_canvas.load\_pixmap(QPixmap(self.file\_open\_image))

return

```

- \*\*Mục đích:\*\*

Chụp ảnh từ camera, lưu ảnh, hiển thị lên canvas.

- \*\*Chi tiết:\*\*

- Nếu camera đang hiển thị, lấy ảnh từ biến `self.mat`.

- Nếu camera đang mở, lấy ảnh từ camera.

- Lưu ảnh vào log.

- Hiển thị ảnh lên canvas.

---

## 23. \*\*Hàm `on\_click\_but\_out\_all`\*\*

```python

def on\_click\_but\_out\_all(self, port, state):

if self.io\_controller:

if self.io\_controller.is\_open():

self.io\_controller.write\_out(port, PortState.On if state else PortState.Off)

```

- \*\*Mục đích:\*\*

Điều khiển cổng output của IO Controller.

- \*\*Chi tiết:\*\*

- Kiểm tra IO Controller đã mở chưa.

- Nếu đã mở, ghi trạng thái (On/Off) vào cổng output.

---

## 24. \*\*Hàm `on\_click\_but\_connect\_scanner`\*\*

```python

def on\_click\_but\_connect\_scanner(self):

if self.but\_connect\_scanner.property("status") == "Open":

if not self.connect\_scanner():

return

self.but\_connect\_scanner.setProperty("status", "Close")

self.but\_connect\_scanner.setText('Close')

self.style().unpolish(self.but\_connect\_scanner)

self.style().polish(self.but\_connect\_scanner)

self.but\_connect\_scanner.clicked.disconnect()

self.but\_connect\_scanner.clicked.connect(self.on\_click\_but\_close\_scanner)

```

- \*\*Mục đích:\*\*

Kết nối scanner, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Connect Scanner".

- Nếu đang "Open", gọi `connect\_scanner()`.

- Nếu kết nối thành công, đổi trạng thái nút thành "Close".

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (đóng scanner).

---

## 25. \*\*Hàm `on\_click\_but\_close\_scanner`\*\*

```python

def on\_click\_but\_close\_scanner(self):

if self.but\_connect\_scanner.property("status") == "Close":

self.but\_connect\_scanner.setText('Open')

self.but\_connect\_scanner.setProperty("status", "Open")

self.close\_scanner()

self.style().unpolish(self.but\_connect\_scanner)

self.style().polish(self.but\_connect\_scanner)

self.but\_connect\_scanner.clicked.disconnect()

self.but\_connect\_scanner.clicked.connect(self.on\_click\_but\_connect\_scanner)

```

- \*\*Mục đích:\*\*

Đóng scanner, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Connect Scanner".

- Nếu đang "Close", đổi trạng thái thành "Open", gọi `close\_scanner()`.

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (mở scanner).

---

## 26. \*\*Hàm `connect\_scanner`\*\*

```python

def connect\_scanner(self):

port = self.combo\_box\_com\_controller\_3.currentText()

baudrate = int(self.combo\_box\_baudrate\_controller\_3.currentText())

if self.scanner.connect(port, baudrate):

self.timer.start(100) # đọc mỗi 100ms

self.is\_connect\_scanner = True

return True

else:

self.is\_connect\_scanner = False

return False

```

- \*\*Mục đích:\*\*

Kết nối scanner qua cổng COM, baudrate.

- \*\*Chi tiết:\*\*

- Lấy port, baudrate từ combo box.

- Gọi `self.scanner.connect(port, baudrate)`.

- Nếu kết nối thành công, bật timer đọc dữ liệu, đặt `is\_connect\_scanner = True`.

- Nếu thất bại, đặt `is\_connect\_scanner = False`.

---

## 27. \*\*Hàm `close\_scanner`\*\*

```python

def close\_scanner(self):

self.timer.stop()

self.scanner.disconnect()

self.is\_connect\_scanner = False

```

- \*\*Mục đích:\*\*

Đóng scanner, dừng timer.

- \*\*Chi tiết:\*\*

- Dừng timer đọc dữ liệu.

- Gọi `self.scanner.disconnect()`.

- Đặt `is\_connect\_scanner = False`.

---

## 28. \*\*Hàm `read\_serial\_data`\*\*

```python

def read\_serial\_data(self):

try:

data\_sn = self.scanner.read\_data()

return data\_sn

except Exception as e:

print(e)

```

- \*\*Mục đích:\*\*

Đọc dữ liệu từ scanner.

- \*\*Chi tiết:\*\*

- Gọi `self.scanner.read\_data()`.

- Trả về dữ liệu đọc được.

- Nếu có lỗi, in lỗi.

---

## 29. \*\*Hàm `on\_click\_but\_test\_teaching`\*\*

```python

def on\_click\_but\_test\_teaching(self):

if self.file\_open\_image:

name\_model = self.combo\_box\_model\_ai.currentText()

classes\_ai = self.combo\_box\_classes\_ai.currentText()

model\_ai = self.load\_model(f'res/Model\_AI/{name\_model}')

image = cv2.imread(self.file\_open\_image)

shapes\_box = self.get\_shapes(self.teaching\_canvas)

name\_model = self.combo\_box\_model\_name\_teaching.currentText()

result = self.detect\_objects(name\_model, model\_ai, classes\_ai, image, shapes\_box)

self.load\_result\_classes(self.table\_widget\_classes\_ai\_teaching,

result.label\_counts, result.ret, result.error)

self.show\_image(self.teaching\_canvas, result.output\_image)

```

- \*\*Mục đích:\*\*

Kiểm tra thử mô hình AI trên ảnh đã chọn.

- \*\*Chi tiết:\*\*

- Kiểm tra ảnh đã mở chưa.

- Lấy tên model, file nhãn từ combo box.

- Load model AI.

- Đọc ảnh từ file.

- Lấy vùng kiểm tra (shapes) từ canvas.

- Gọi `detect\_objects()` để nhận diện vật thể.

- Hiển thị kết quả lên bảng, canvas.

---

## 30. \*\*Hàm `on\_click\_but\_open\_com\_send\_data`\*\*

```python

def on\_click\_but\_open\_com\_send\_data(self):

if self.but\_open\_com\_send\_data.property("status") == "Open":

if not self.open\_com\_send\_data():

return

self.but\_open\_com\_send\_data.setProperty("status", "Close")

self.but\_open\_com\_send\_data.setText('Close')

self.style().unpolish(self.but\_open\_com\_send\_data)

self.style().polish(self.but\_open\_com\_send\_data)

self.but\_open\_com\_send\_data.clicked.disconnect()

self.but\_open\_com\_send\_data.clicked.connect(self.on\_click\_but\_close\_com\_send\_data)

```

- \*\*Mục đích:\*\*

Mở cổng COM gửi dữ liệu, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Open COM Send Data".

- Nếu đang "Open", gọi `open\_com\_send\_data()`.

- Nếu mở thành công, đổi trạng thái nút thành "Close".

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (đóng cổng COM).

---

## 31. \*\*Hàm `on\_click\_but\_close\_com\_send\_data`\*\*

```python

def on\_click\_but\_close\_com\_send\_data(self):

if self.but\_open\_com\_send\_data.property("status") == "Close":

self.but\_open\_com\_send\_data.setText('Open')

self.but\_open\_com\_send\_data.setProperty("status", "Open")

self.close\_com\_send\_data()

self.style().unpolish(self.but\_open\_com\_send\_data)

self.style().polish(self.but\_open\_com\_send\_data)

self.but\_open\_com\_send\_data.clicked.disconnect()

self.but\_open\_com\_send\_data.clicked.connect(self.on\_click\_but\_open\_com\_send\_data)

```

- \*\*Mục đích:\*\*

Đóng cổng COM gửi dữ liệu, thay đổi trạng thái nút, kết nối signal-slot.

- \*\*Chi tiết:\*\*

- Kiểm tra trạng thái nút "Open COM Send Data".

- Nếu đang "Close", đổi trạng thái thành "Open", gọi `close\_com\_send\_data()`.

- Cập nhật style nút.

- Ngắt kết nối signal-slot cũ, kết nối signal-slot mới (mở cổng COM).

---

Tiếp tục giải thích các hàm còn lại:

## 32. \*\*Hàm `open\_com\_send\_data`\*\*

```python

def open\_com\_send\_data(self):

port = self.combo\_box\_com\_controller\_4.currentText()

baudrate = int(self.combo\_box\_baudrate\_controller\_4.currentText())

if self.send\_data\_com.connect(port, baudrate):

self.is\_connect\_send\_data = True

return True

else:

self.is\_connect\_send\_data = False

return False

```

- \*\*Mục đích:\*\* Kết nối cổng COM để gửi dữ liệu

- \*\*Chi tiết:\*\*

- Lấy port và baudrate từ combo box

- Thử kết nối qua SerialReceiver

- Cập nhật trạng thái kết nối

- Trả về True/False tương ứng với kết quả kết nối

## 33. \*\*Hàm `close\_com\_send\_data`\*\*

```python

def close\_com\_send\_data(self):

self.send\_data\_com.disconnect()

self.is\_connect\_send\_data = False

```

- \*\*Mục đích:\*\* Đóng kết nối cổng COM gửi dữ liệu

- \*\*Chi tiết:\*\*

- Ngắt kết nối SerialReceiver

- Cập nhật trạng thái kết nối thành False

## 34. \*\*Hàm `connect\_database`\*\*

```python

def connect\_database(self):

self.file\_path\_database = os.path.join(os.getcwd(), 'res', 'Database', 'database.db')

self.load\_database(self.file\_path\_database)

self.but\_find\_database.setDisabled(False)

```

- \*\*Mục đích:\*\* Kết nối và khởi tạo database

- \*\*Chi tiết:\*\*

- Tạo đường dẫn đến file database

- Load dữ liệu từ database

- Kích hoạt nút tìm kiếm database

## 35. \*\*Hàm `on\_click\_but\_find\_database`\*\*

```python

def on\_click\_but\_find\_database(self):

self.find\_database()

```

- \*\*Mục đích:\*\* Xử lý sự kiện khi nhấn nút tìm kiếm database

- \*\*Chi tiết:\*\*

- Gọi hàm find\_database() để thực hiện tìm kiếm

## 36. \*\*Hàm `on\_click\_but\_load\_database`\*\*

```python

def on\_click\_but\_load\_database(self):

self.load\_database(self.file\_path\_database)

```

- \*\*Mục đích:\*\* Xử lý sự kiện khi nhấn nút load database

- \*\*Chi tiết:\*\*

- Gọi hàm load\_database() để load lại dữ liệu

## 37. \*\*Hàm `on\_click\_action\_theme`\*\*

```python

def on\_click\_action\_theme(self, path):

with open(path, "r", encoding="utf-8") as file:

app.setStyleSheet(file.read())

```

- \*\*Mục đích:\*\* Thay đổi theme của ứng dụng

- \*\*Chi tiết:\*\*

- Đọc file CSS từ đường dẫn được chỉ định

- Áp dụng style mới cho ứng dụng

## 38. \*\*Hàm `ndarray2pixmap`\*\*

```python

def ndarray2pixmap(self, mat: np.ndarray):

height, width, channel = mat.shape

bytes\_per\_line = channel \* width

qimage = QImage(mat.data, width, height, bytes\_per\_line, QImage.Format.Format\_RGB888)

pixmap = QPixmap.fromImage(qimage)

return pixmap

```

- \*\*Mục đích:\*\* Chuyển đổi mảng numpy thành QPixmap

- \*\*Chi tiết:\*\*

- Lấy kích thước và số kênh của ảnh

- Tạo QImage từ dữ liệu mảng

- Chuyển QImage thành QPixmap

- Trả về QPixmap

## 39. \*\*Hàm `pixmap2ndarray`\*\*

```python

def pixmap2ndarray(self, pixmap: QPixmap) -> np.ndarray:

image = pixmap.toImage()

image = image.convertToFormat(QImage.Format\_RGB888)

width, height = image.width(), image.height()

ptr = image.bits()

ptr.setsize(image.byteCount())

arr = np.array(ptr).reshape(height, width, 3)

return arr

```

- \*\*Mục đích:\*\* Chuyển đổi QPixmap thành mảng numpy

- \*\*Chi tiết:\*\*

- Chuyển QPixmap thành QImage

- Chuyển đổi định dạng ảnh

- Lấy kích thước ảnh

- Tạo mảng numpy từ dữ liệu ảnh

- Trả về mảng numpy

## 40. \*\*Hàm `load\_model`\*\*

```python

def load\_model(self, model\_path):

start\_time = time.perf\_counter()

try:

model = YOLO(model\_path)

model.predict(np.zeros((640, 640, 3), dtype=np.uint8))

end\_time = time.perf\_counter()

msg = f"- Loading the network took {end\_time-start\_time:.2f} seconds."

print(msg)

except Exception as ex:

print(str(ex))

model = None

return model

```

- \*\*Mục đích:\*\* Load model YOLO

- \*\*Chi tiết:\*\*

- Đo thời gian load model

- Khởi tạo model YOLO từ đường dẫn

- Chạy thử model với ảnh trống

- In thời gian load

- Xử lý lỗi nếu có

- Trả về model hoặc None nếu lỗi

## 41. \*\*Hàm `load\_label`\*\*

```python

def load\_label(self, file\_path):

labels = set()

with open(file\_path, "r", encoding="utf-8") as f:

for line in f:

line = line.strip()

if not line:

continue

parts = line.split("\_", 1)

if len(parts) > 1:

label\_part = parts[1].split()[0]

labels.add(label\_part)

return labels

```

- \*\*Mục đích:\*\* Load danh sách nhãn từ file

- \*\*Chi tiết:\*\*

- Đọc file theo từng dòng

- Tách nhãn từ định dạng "index\_label"

- Thêm nhãn vào set

- Trả về set các nhãn

## 42. \*\*Hàm `is\_inside\_any\_shape`\*\*

```python

def is\_inside\_any\_shape(self, x1, y1, x2, y2, shapes\_dict):

for key in shapes\_dict:

sx, sy, sw, sh = shapes\_dict[key]

sx2 = sx + sw

sy2 = sy + sh

if x1 >= sx and y1 >= sy and x2 <= sx2 and y2 <= sy2:

return True

return False

```

- \*\*Mục đích:\*\* Kiểm tra xem một box có nằm trong bất kỳ shape nào không

- \*\*Chi tiết:\*\*

- Duyệt qua các shape trong dictionary

- Tính toán tọa độ góc phải dưới của shape

- Kiểm tra box có nằm hoàn toàn trong shape không

- Trả về True nếu có, False nếu không

## 43. \*\*Hàm `detect\_objects`\*\*

```python

def detect\_objects(self, name\_model, model\_ai, classes\_ai, image, shapes, seri\_number=None):

# ... (code dài, xem file gốc)

```

- \*\*Mục đích:\*\* Nhận diện vật thể trong ảnh

- \*\*Chi tiết:\*\*

- Chạy model YOLO trên ảnh

- Vẽ box và nhãn lên ảnh

- Kiểm tra vị trí vật thể

- So sánh với cấu hình chuẩn

- Xác định kết quả PASS/FAIL

- Vẽ thông tin lên ảnh

- Trả về kết quả

## 44. \*\*Hàm `start\_camera`\*\*

```python

def start\_camera(self, camera: Camera):

self.b\_start = True

threading.Thread(target=self.loop\_live\_camera, args=(camera, ), daemon=True).start()

```

- \*\*Mục đích:\*\* Bắt đầu luồng camera

- \*\*Chi tiết:\*\*

- Đặt flag b\_start = True

- Tạo thread mới để chạy loop\_live\_camera

- Thread chạy ở chế độ daemon

## 45. \*\*Hàm `stop\_camera`\*\*

```python

def stop\_camera(self):

self.b\_start = False

```

- \*\*Mục đích:\*\* Dừng luồng camera

- \*\*Chi tiết:\*\*

- Đặt flag b\_start = False để dừng vòng lặp

## 46. \*\*Hàm `loop\_live\_camera`\*\*

```python

def loop\_live\_camera(self, camera: Camera):

while True:

self.mat = camera.get\_frame()

if self.mat is not None:

self.showDstSignal.emit(self.teaching\_canvas, self.mat)

if not self.is\_showing\_camera:

break

time.sleep(0.04)

```

- \*\*Mục đích:\*\* Vòng lặp lấy và hiển thị ảnh từ camera

- \*\*Chi tiết:\*\*

- Lấy frame từ camera

- Nếu có frame, emit signal để hiển thị

- Kiểm tra flag is\_showing\_camera

- Delay 0.04s giữa các frame

## 47. \*\*Hàm `set\_canvas`\*\*

```python

def set\_canvas(self, canvas: Canvas, mat: np.ndarray):

mat\_rgb = cv2.cvtColor(mat, cv2.COLOR\_BGR2RGB)

pixmap = self.ndarray2pixmap(mat\_rgb)

canvas.load\_pixmap(pixmap)

```

- \*\*Mục đích:\*\* Hiển thị ảnh lên canvas

- \*\*Chi tiết:\*\*

- Chuyển ảnh từ BGR sang RGB

- Chuyển mảng numpy thành QPixmap

- Load pixmap lên canvas

## 48. \*\*Hàm `show\_image`\*\*

```python

def show\_image(self, canvas, mat):

self.set\_canvas(canvas, mat)

```

- \*\*Mục đích:\*\* Hiển thị ảnh lên canvas

- \*\*Chi tiết:\*\*

- Gọi hàm set\_canvas để hiển thị ảnh

## 49. \*\*Hàm `add\_log`\*\*

```python

def add\_log(self, message):

item = QStandardItem(message)

self.log\_model.appendRow(item)

self.list\_log\_view.scrollToBottom()

```

- \*\*Mục đích:\*\* Thêm log vào danh sách

- \*\*Chi tiết:\*\*

- Tạo QStandardItem từ message

- Thêm item vào model

- Cuộn xuống dòng cuối

## 50. \*\*Hàm `saveSettings`\*\*

```python

def saveSettings(self):

settings = QSettings("settings.ini", QSettings.IniFormat)

settings.setValue("geometry", self.saveGeometry())

settings.setValue("windowState", self.saveState())

settings.setValue("combo\_box\_index", self.combo\_box\_model\_name\_teaching.currentIndex())

settings.setValue("combo\_box\_index\_auto", self.combo\_box\_model\_name\_auto.currentIndex())

```

- \*\*Mục đích:\*\* Lưu cài đặt vào file

- \*\*Chi tiết:\*\*

- Lưu kích thước và vị trí cửa sổ

- Lưu trạng thái cửa sổ

- Lưu index của các combo box

## 51. \*\*Hàm `loadSettings`\*\*

```python

def loadSettings(self):

settings = QSettings("settings.ini", QSettings.IniFormat)

geometry = settings.value("geometry", QByteArray())

windowState = settings.value("windowState", QByteArray())

if isinstance(geometry, QByteArray) and not geometry.isEmpty():

self.restoreGeometry(geometry)

if isinstance(windowState, QByteArray) and not windowState.isEmpty():

self.restoreState(windowState)

self.combo\_box\_model\_name\_teaching.setCurrentIndex(int(settings.value("combo\_box\_index", 0)))

self.combo\_box\_model\_name\_auto.setCurrentIndex(int(settings.value("combo\_box\_index\_auto", 0)))

```

- \*\*Mục đích:\*\* Load cài đặt từ file

- \*\*Chi tiết:\*\*

- Load kích thước và vị trí cửa sổ

- Load trạng thái cửa sổ

- Load index của các combo box

## 52. \*\*Hàm `resetLayout`\*\*

```python

def resetLayout(self):

confirm = QMessageBox.question(self, "Reset Layout",

"Bạn có chắc chắn muốn đặt lại bố cục về mặc định không?",

QMessageBox.Yes | QMessageBox.No)

if confirm == QMessageBox.Yes:

self.is\_resetting = True

settings = QSettings("settings.ini", QSettings.IniFormat)

settings.clear()

QMessageBox.information(self, "Thông báo", "Đã đặt lại bố cục. Ứng dụng sẽ khởi động lại.")

self.restartApp()

```

- \*\*Mục đích:\*\* Reset layout về mặc định

- \*\*Chi tiết:\*\*

- Hiện hộp thoại xác nhận

- Nếu đồng ý:

- Đặt flag is\_resetting = True

- Xóa file cài đặt

- Hiện thông báo

- Khởi động lại ứng dụng

## 53. \*\*Hàm `upload\_model\_ai`\*\*

```python

def upload\_model\_ai(self):

model\_dir = "res/Model\_AI"

model\_names = [name for name in os.listdir(model\_dir) if name.endswith((".pt", ".pth"))]

classes\_names = [name for name in os.listdir(model\_dir) if name.endswith((".txt"))]

self.combo\_box\_model\_ai.clear()

self.combo\_box\_model\_ai.addItems(model\_names)

self.combo\_box\_classes\_ai.clear()

self.combo\_box\_classes\_ai.addItems(classes\_names)

```

- \*\*Mục đích:\*\* Load danh sách model AI và file classes

- \*\*Chi tiết:\*\*

- Tìm các file .pt, .pth trong thư mục Model\_AI

- Tìm các file .txt trong thư mục Model\_AI

- Thêm tên model vào combo box model\_ai

- Thêm tên file classes vào combo box classes\_ai

## 54. \*\*Hàm `upload\_model\_names`\*\*

```python

def upload\_model\_names(self):

model\_dir = "Model"

model\_names = [name for name in os.listdir(model\_dir)]

self.combo\_box\_model\_name\_teaching.clear()

self.combo\_box\_model\_name\_teaching.addItems(model\_names)

self.combo\_box\_model\_name\_auto.clear()

self.combo\_box\_model\_name\_auto.addItems(model\_names)

self.combo\_box\_model\_name\_data.addItems(model\_names)

```

- \*\*Mục đích:\*\* Load danh sách tên model

- \*\*Chi tiết:\*\*

- Tìm tất cả file trong thư mục Model

- Thêm tên model vào các combo box tương ứng

## 55. \*\*Hàm `upload\_feature\_camera`\*\*

```python

def upload\_feature\_camera(self):

feature\_dir = "res/Camera"

feature\_names = [name for name in os.listdir(feature\_dir) if name.endswith((".ini"))]

self.combo\_box\_feature\_camera.clear()

self.combo\_box\_feature\_camera.addItems(feature\_names)

```

- \*\*Mục đích:\*\* Load danh sách file cấu hình camera

- \*\*Chi tiết:\*\*

- Tìm các file .ini trong thư mục Camera

- Thêm tên file vào combo box feature\_camera

## 56. \*\*Hàm `update\_label\_status`\*\*

```python

def update\_label\_status(self, status):

self.label\_result.setText(status)

self.label\_result.setProperty("status", status)

self.label\_result.style().polish(self.label\_result)

```

- \*\*Mục đích:\*\* Cập nhật trạng thái hiển thị

- \*\*Chi tiết:\*\*

- Set text cho label

- Set property status

- Cập nhật style

## 57. \*\*Hàm `update\_label\_result`\*\*

```python

def update\_label\_result(self, ret):

self.count\_product\_total += 1

if ret == "PASS":

self.count\_product\_ok += 1

else:

self.count\_product\_ng += 1

if self.count\_product\_total != 0:

rate = round(self.count\_product\_ok / self.count\_product\_total, 2)

self.label\_ok.setText(str(self.count\_product\_ok))

self.label\_ng.setText(str(self.count\_product\_ng))

self.label\_total.setText(str(self.count\_product\_total))

self.label\_rate.setText(str(rate))

```

- \*\*Mục đích:\*\* Cập nhật kết quả kiểm tra

- \*\*Chi tiết:\*\*

- Tăng tổng số sản phẩm

- Tăng số OK/NG tùy kết quả

- Tính tỷ lệ OK

- Cập nhật các label hiển thị

## 58. \*\*Hàm `refresh\_ports`\*\*

```python

def refresh\_ports(self):

ports = serial.tools.list\_ports.comports()

self.combo\_box\_com\_controller.clear()

self.combo\_box\_com\_controller\_2.clear()

self.combo\_box\_com\_controller\_3.clear()

self.combo\_box\_com\_controller\_4.clear()

self.combo\_box\_com\_controller.addItems([port.device for port in ports])

self.combo\_box\_com\_controller\_2.addItems([port.device for port in ports])

self.combo\_box\_com\_controller\_3.addItems([port.device for port in ports])

self.combo\_box\_com\_controller\_4.addItems([port.device for port in ports])

self.combo\_box\_baudrate\_controller.clear()

self.combo\_box\_baudrate\_controller\_2.clear()

self.combo\_box\_baudrate\_controller\_3.clear()

self.combo\_box\_baudrate\_controller\_4.clear()

self.combo\_box\_baudrate\_controller.addItems(list(map(str, serial.Serial.BAUDRATES)))

self.combo\_box\_baudrate\_controller\_2.addItems(list(map(str, serial.Serial.BAUDRATES)))

self.combo\_box\_baudrate\_controller\_3.addItems(list(map(str, serial.Serial.BAUDRATES)))

self.combo\_box\_baudrate\_controller\_4.addItems(list(map(str, serial.Serial.BAUDRATES)))

devices = get\_camera\_devices()

self.combo\_box\_id\_camera.clear()

self.combo\_box\_id\_camera.addItems(list(devices.keys()))

```

- \*\*Mục đích:\*\* Refresh danh sách cổng COM và camera

- \*\*Chi tiết:\*\*

- Lấy danh sách cổng COM

- Thêm vào các combo box

- Lấy danh sách baudrate

- Thêm vào các combo box

- Lấy danh sách camera

- Thêm vào combo box camera

## 59. \*\*Hàm `get\_config\_from\_ui`\*\*

```python

def get\_config\_from\_ui(self):

config = SimpleNamespace(

name\_model = self.combo\_box\_model\_name\_teaching.currentText(),

shapes = self.get\_shapes(self.teaching\_canvas),

model\_ai=SimpleNamespace(

model\_name=self.combo\_box\_model\_ai.currentText(),

classes\_name=self.combo\_box\_classes\_ai.currentText(),

threshold=self.line\_edit\_threshold.text(),

),

hardware=SimpleNamespace(

server=SimpleNamespace(

ip=self.line\_edit\_ip\_server.text(),

port=self.line\_edit\_port\_server.text()

),

camera=SimpleNamespace(

id=self.combo\_box\_id\_camera.currentText(),

feature=self.combo\_box\_feature\_camera.currentText(),

name=self.combo\_box\_name\_camera.currentText()

),

lighting=SimpleNamespace(

comport=self.combo\_box\_com\_controller.currentText(),

baudrate=self.combo\_box\_baudrate\_controller.currentText(),

channel\_value\_0=self.spin\_box\_channel\_value\_0.value(),

channel\_value\_1=self.spin\_box\_channel\_value\_1.value(),

channel\_value\_2=self.spin\_box\_channel\_value\_2.value(),

channel\_value\_3=self.spin\_box\_channel\_value\_3.value(),

controller=self.combo\_box\_light\_controller.currentText(),

delay=self.spin\_box\_delay\_controller.value()

),

system=SimpleNamespace(

log\_dir=self.line\_edit\_log\_dir.text(),

log\_size=self.spin\_box\_log\_size.value(),

auto\_start=self.check\_box\_auto\_start.isChecked(),

database\_path=self.line\_edit\_database\_path.text()

),

io=SimpleNamespace(

comport=self.combo\_box\_com\_controller\_2.currentText(),

baudrate=self.combo\_box\_baudrate\_controller\_2.currentText(),

delay=self.line\_edit\_time\_delay\_io.text()

),

scanner=SimpleNamespace(

comport=self.combo\_box\_com\_controller\_3.currentText(),

baudrate=self.combo\_box\_baudrate\_controller\_3.currentText()

),

send\_data=SimpleNamespace(

comport=self.combo\_box\_com\_controller\_4.currentText(),

baudrate=self.combo\_box\_baudrate\_controller\_4.currentText()

)

)

)

return config

```

- \*\*Mục đích:\*\* Lấy cấu hình từ giao diện

- \*\*Chi tiết:\*\*

- Tạo SimpleNamespace chứa tất cả cấu hình

- Lấy giá trị từ các widget trên giao diện

- Trả về cấu hình

## 60. \*\*Hàm `set\_config\_teaching`\*\*

```python

def set\_config\_teaching(self, config):

self.set\_shapes(self.teaching\_canvas, config.get("shapes", {}))

model\_ai = config.get("model\_ai", {})

self.combo\_box\_model\_ai.setCurrentText(model\_ai.get("model\_name", ""))

self.combo\_box\_classes\_ai.setCurrentText(model\_ai.get("classes\_name", ""))

self.line\_edit\_threshold.setText(model\_ai.get("threshold", ""))

hardware = config.get("hardware", {})

server\_config = hardware.get("server", {})

self.line\_edit\_ip\_server.setText(server\_config.get("ip", ""))

self.line\_edit\_port\_server.setText(server\_config.get("port", ""))

camera\_config = hardware.get("camera", {})

self.combo\_box\_id\_camera.setCurrentText(camera\_config.get("id", ""))

self.combo\_box\_feature\_camera.setCurrentText(camera\_config.get("feature", ""))

self.combo\_box\_name\_camera.setCurrentText(camera\_config.get("name", ""))

lighting\_config = hardware.get("lighting", {})

self.combo\_box\_com\_controller.setCurrentText(lighting\_config.get("comport", ""))

self.combo\_box\_baudrate\_controller.setCurrentText(lighting\_config.get("baudrate", ""))

self.spin\_box\_channel\_value\_0.setValue(lighting\_config.get("channel\_value\_0", 0))

self.spin\_box\_channel\_value\_1.setValue(lighting\_config.get("channel\_value\_1", 0))

self.spin\_box\_channel\_value\_2.setValue(lighting\_config.get("channel\_value\_2", 0))

self.spin\_box\_channel\_value\_3.setValue(lighting\_config.get("channel\_value\_3", 0))

self.combo\_box\_light\_controller.setCurrentText(lighting\_config.get("controller", ""))

self.spin\_box\_delay\_controller.setValue(lighting\_config.get("delay", 0))

system\_config = hardware.get("system", {})

self.line\_edit\_log\_dir.setText(system\_config.get("log\_dir", ""))

self.spin\_box\_log\_size.setValue(system\_config.get("log\_size", 0))

self.check\_box\_auto\_start.setChecked(system\_config.get("auto\_start", False))

self.line\_edit\_database\_path.setText(system\_config.get("database\_path", ""))

io\_config = hardware.get("io", {})

self.combo\_box\_com\_controller\_2.setCurrentText(io\_config.get("comport", ""))

self.combo\_box\_baudrate\_controller\_2.setCurrentText(io\_config.get("baudrate", ""))

self.line\_edit\_time\_delay\_io.setText(io\_config.get("delay", ""))

scanner\_config = hardware.get("scanner", {})

self.combo\_box\_com\_controller\_3.setCurrentText(scanner\_config.get("comport", ""))

self.combo\_box\_baudrate\_controller\_3.setCurrentText(scanner\_config.get("baudrate", ""))

send\_data\_config = hardware.get("send\_data", {})

self.combo\_box\_com\_controller\_4.setCurrentText(send\_data\_config.get("comport", ""))

self.combo\_box\_baudrate\_controller\_4.setCurrentText(send\_data\_config.get("baudrate", ""))

```

- \*\*Mục đích:\*\* Set cấu hình lên giao diện

- \*\*Chi tiết:\*\*

- Set shapes lên canvas

- Set các giá trị cấu hình lên các widget tương ứng

- Xử lý từng phần cấu hình (model\_ai, server, camera, lighting, system, io, scanner, send\_data)

## 61. \*\*Hàm `get\_config\_auto`\*\*

```python

def get\_config\_auto(self, model\_name):

config\_data = self.handle\_file\_json.load(model\_name)

model\_ai\_config = config\_data.get('model\_ai', {})

hardware = config\_data.get("hardware", {})

server\_config = hardware.get("server", {})

camera\_config = hardware.get("camera", {})

lighting\_config = hardware.get("lighting", {})

system\_config = hardware.get("system", {})

io\_config = hardware.get("io", {})

scanner\_config = hardware.get("scanner", {})

send\_data\_config = hardware.get("send\_data", {})

config = SimpleNamespace(

name\_model = config\_data.get("name\_model", {}),

shapes = config\_data.get("shapes", {}),

model\_ai=SimpleNamespace(

model\_name=model\_ai\_config.get("model\_name", ""),

classes\_name=model\_ai\_config.get("classes\_name", ""),

threshold=model\_ai\_config.get("threshold", ""),

),

hardware=SimpleNamespace(

server=SimpleNamespace(

ip=server\_config.get("ip", ""),

port=server\_config.get("port", "")

),

camera=SimpleNamespace(

id=camera\_config.get("id", ""),

feature=camera\_config.get("feature", ""),

name=camera\_config.get("name", "")

),

lighting=SimpleNamespace(

comport=lighting\_config.get("comport", ""),

baudrate=lighting\_config.get("baudrate", ""),

channel\_value\_0=lighting\_config.get("channel\_value\_0", 0),

channel\_value\_1=lighting\_config.get("channel\_value\_1", 0),

channel\_value\_2=lighting\_config.get("channel\_value\_2", 0),

channel\_value\_3=lighting\_config.get("channel\_value\_3", 0),

controller=lighting\_config.get("controller", ""),

delay=lighting\_config.get("delay", 0),

),

system=SimpleNamespace(

log\_dir=system\_config.get("log\_dir", ""),

log\_size=system\_config.get("log\_size", 0),

auto\_start=system\_config.get("auto\_start", False),

database\_path=system\_config.get("database\_path", "")

),

io=SimpleNamespace(

comport=io\_config.get("comport", ""),

baudrate=io\_config.get("baudrate", ""),

delay=io\_config.get("delay", "")

),

scanner=SimpleNamespace(

comport=scanner\_config.get("comport", ""),

baudrate=scanner\_config.get("baudrate", "")

),

send\_data=SimpleNamespace(

comport=send\_data\_config.get("comport", ""),

baudrate=send\_data\_config.get("baudrate", "")

)

)

)

return config

```

- \*\*Mục đích:\*\* Load cấu hình từ file JSON cho chế độ Auto

- \*\*Chi tiết:\*\*

- Load dữ liệu từ file JSON

- Tạo SimpleNamespace chứa cấu hình

- Trả về cấu hình

## 62. \*\*Hàm `load\_model\_change\_teaching`\*\*

```python

def load\_model\_change\_teaching(self):

name\_model = self.combo\_box\_model\_name\_teaching.currentText()

if name\_model and self.combo\_box\_model\_name\_teaching.findText(name\_model) == -1:

QMessageBox.warning(self, 'Warning', 'Tên mô hình không đúng', QMessageBox.StandardButton.Close)

return

config = self.handle\_file\_json.load(file\_path=f'Model/{name\_model}')

self.set\_config\_teaching(config)

```

- \*\*Mục đích:\*\* Load cấu hình khi thay đổi model

- \*\*Chi tiết:\*\*

- Lấy tên model từ combo box

- Kiểm tra model có tồn tại không

- Load cấu hình từ file JSON

- Set cấu hình lên giao diện

## 63. \*\*Hàm `open\_camera\_teaching`\*\*

```python

def open\_camera\_teaching(self):

name\_camera = self.combo\_box\_name\_camera.currentText()

id\_camera=self.combo\_box\_id\_camera.currentText()

feature\_camera=self.combo\_box\_feature\_camera.currentText()

success = self.camera.open\_camera(name\_camera, config={

'id': id\_camera,

'feature': f'res/Camera/{feature\_camera}'

})

if not success:

return

self.is\_open\_camera = True

self.but\_start\_camera\_teaching.setDisabled(False)

```

- \*\*Mục đích:\*\* Mở camera trong chế độ Teaching

- \*\*Chi tiết:\*\*

- Lấy thông tin camera từ combo box

- Mở camera với cấu hình

- Nếu thành công:

- Set flag is\_open\_camera = True

- Kích hoạt nút Start Camera

## 64. \*\*Hàm `close\_camera\_teaching`\*\*

```python

def close\_camera\_teaching(self):

self.camera.close\_camera()

self.is\_open\_camera = False

self.is\_showing\_camera = False

self.but\_start\_camera\_teaching.setDisabled(True)

```

- \*\*Mục đích:\*\* Đóng camera trong chế độ Teaching

- \*\*Chi tiết:\*\*

- Đóng camera

- Set các flag về False

- Vô hiệu hóa nút Start Camera

## 65. \*\*Hàm `start\_camera\_teaching`\*\*

```python

def start\_camera\_teaching(self):

try:

self.is\_showing\_camera = True

threading.Thread(target=self.loop\_live\_camera, args=(self.camera, ), daemon=True).start()

self.main\_logger.info('Camera is starting')

except Exception as ex:

self.main\_logger.error(f'Failed to show camera. Error{ex}')

```

- \*\*Mục đích:\*\* Bắt đầu hiển thị camera

- \*\*Chi tiết:\*\*

- Set flag is\_showing\_camera = True

- Tạo thread mới để chạy loop\_live\_camera

- Ghi log

## 66. \*\*Hàm `stop\_camera\_teaching`\*\*

```python

def stop\_camera\_teaching(self):

try:

self.is\_showing\_camera = False

self.main\_logger.info('Camera was stoped')

except Exception as ex:

self.main\_logger.error(f'Failed to stop camera. Error{ex}')

```

- \*\*Mục đích:\*\* Dừng hiển thị camera

- \*\*Chi tiết:\*\*

- Set flag is\_showing\_camera = False

- Ghi log

## 67. \*\*Hàm `connect\_server\_teaching`\*\*

```python

def connect\_server\_teaching(self):

ip = self.line\_edit\_ip\_server.text()

port = self.line\_edit\_port\_server.text()

self.server.start\_server(ip, int(port))

if not self.server.is\_connected:

return

self.is\_connect\_server = True

```

- \*\*Mục đích:\*\* Kết nối server trong chế độ Teaching

- \*\*Chi tiết:\*\*

- Lấy IP và port từ line edit

- Khởi động server

- Nếu kết nối thành công, set flag is\_connect\_server = True

## 68. \*\*Hàm `close\_server\_teaching`\*\*

```python

def close\_server\_teaching(self):

self.server.stop\_server()

self.is\_connect\_server = False

```

- \*\*Mục đích:\*\* Đóng server trong chế độ Teaching

- \*\*Chi tiết:\*\*

- Dừng server

- Set flag is\_connect\_server = False

## 69. \*\*Hàm `open\_light\_auto`\*\*

```python

def open\_light\_auto(self):

controller = self.config\_auto.hardware.lighting.controller

comport = self.config\_auto.hardware.lighting.comport

baudrate = self.config\_auto.hardware.lighting.baudrate

channel\_0 = self.config\_auto.hardware.lighting.channel\_value\_0

channel\_1 = self.config\_auto.hardware.lighting.channel\_value\_1

channel\_2 = self.config\_auto.hardware.lighting.channel\_value\_2

channel\_3 = self.config\_auto.hardware.lighting.channel\_value\_3

delay = self.config\_auto.hardware.lighting.delay

if controller == 'DCPController':

self.light = DCPController(comport)

else:

self.light = LCPController(comport)

self.light.light\_logger.signalLog.connect(self.add\_log)

if not self.light.open():

return False, []

return True, [channel\_0, channel\_1, channel\_2, channel\_3, delay]

```

- \*\*Mục đích:\*\* Mở đèn trong chế độ Auto

- \*\*Chi tiết:\*\*

- Lấy cấu hình đèn từ config\_auto

- Khởi tạo controller đèn tương ứng

- Kết nối signal log

- Mở đèn

- Trả về kết quả và giá trị các kênh

## 70. \*\*Hàm `open\_light\_teaching`\*\*

```python

def open\_light\_teaching(self):

controller = self.combo\_box\_light\_controller.currentText()

comport = self.combo\_box\_com\_controller.currentText()

baudrate = self.combo\_box\_baudrate\_controller.currentText()

channel\_0 = self.spin\_box\_channel\_value\_0.value()

channel\_1 = self.spin\_box\_channel\_value\_1.value()

channel\_2 = self.spin\_box\_channel\_value\_2.value()

channel\_3 = self.spin\_box\_channel\_value\_3.value()

delay = self.spin\_box\_delay\_controller.value()

if controller == 'DCPController':

self.light = DCPController(comport)

else:

self.light = LCPController(comport)

self.light.light\_logger.signalLog.connect(self.add\_log)

if not self.light.open():

return False

self.set\_light\_value(channel\_0, channel\_1, channel\_2, channel\_3)

time.sleep(delay / 1000)

return True

```

- \*\*Mục đích:\*\* Mở đèn trong chế độ Teaching

- \*\*Chi tiết:\*\*

- Lấy cấu hình đèn từ giao diện

- Khởi tạo controller đèn tương ứng

- Kết nối signal log

- Mở đèn

- Set giá trị các kênh

- Delay

- Trả về kết quả

## 71. \*\*Hàm `set\_light\_value`\*\*

```python

def set\_light\_value(self, channel\_0=0, channel\_1=0, channel\_2=0, channel\_3=0):

self.light.set\_light\_value(0, channel\_0)

self.light.set\_light\_value(1, channel\_1)

self.light.set\_light\_value(2, channel\_2)

self.light.set\_light\_value(3, channel\_3)

```

- \*\*Mục đích:\*\* Set giá trị các kênh đèn

- \*\*Chi tiết:\*\*

- Set giá trị cho từng kênh đèn

## 72. \*\*Hàm `update\_light\_value`\*\*

```python

def update\_light\_value(self):

if hasattr(self, 'light') and self.but\_open\_light.property("status") == "Close":

channel\_0 = self.spin\_box\_channel\_value\_0.value()

channel\_1 = self.spin\_box\_channel\_value\_1.value()

channel\_2 = self.spin\_box\_channel\_value\_2.value()

channel\_3 = self.spin\_box\_channel\_value\_3.value()

self.light.set\_light\_value(0, channel\_0)

self.light.set\_light\_value(1, channel\_1)

self.light.set\_light\_value(2, channel\_2)

self.light.set\_light\_value(3, channel\_3)

```

- \*\*Mục đích:\*\* Cập nhật giá trị các kênh đèn khi thay đổi trên giao diện

- \*\*Chi tiết:\*\*

- Kiểm tra đèn đã mở chưa

- Lấy giá trị từ spin box

- Set giá trị cho từng kênh đèn

## 73. \*\*Hàm `connect\_io\_controller`\*\*

```python

def connect\_io\_controller(self):

com = self.combo\_box\_com\_controller\_2.currentText()

baud = self.combo\_box\_baudrate\_controller\_2.currentText()

self.io\_controller = IOController(com=com, baud=int(baud))

self.io\_controller.io\_logger.signalLog.connect(self.add\_log)

self.io\_controller.inputSignalOnToOff.connect(self.on\_to\_off)

self.io\_controller.inputSignalOffToOn.connect(self.off\_to\_on)

self.io\_controller.checkSignalOn.connect(self.set\_trigger\_on)

self.io\_controller.checkSensorOn.connect(self.sensor\_on)

self.io\_controller.checkSensorOff.connect(self.sensor\_off)

if self.io\_controller.open():

self.is\_connect\_io\_controller = True

else:

self.is\_connect\_io\_controller = False

```

- \*\*Mục đích:\*\* Kết nối IO Controller

- \*\*Chi tiết:\*\*

- Lấy port và baudrate từ combo box

- Khởi tạo IOController

- Kết nối các signal

- Mở kết nối

- Cập nhật trạng thái kết nối

## 74. \*\*Hàm `close\_io\_controller`\*\*

```python

def close\_io\_controller(self):

self.close\_all\_io\_controller()

self.set\_trigger\_off()

self.set\_io\_result(StepResult.WAIT.value)

self.showResultStatus.emit(StepResult.WAIT.value)

self.io\_controller.close()

```

- \*\*Mục đích:\*\* Đóng IO Controller

- \*\*Chi tiết:\*\*

- Đóng tất cả cổng IO

- Reset trigger

- Set kết quả về WAIT

- Đóng kết nối

## 75. \*\*Hàm `close\_all\_io\_controller`\*\*

```python

def close\_all\_io\_controller(self):

self.io\_controller.write\_out(OutPorts.Out\_1, PortState.Off)

self.io\_controller.write\_out(OutPorts.Out\_3, PortState.Off)

self.io\_controller.write\_out(OutPorts.Out\_2, PortState.Off)

self.io\_controller.write\_out(OutPorts.Out\_4, PortState.Off)

self.io\_controller.write\_out(OutPorts.Out\_5, PortState.Off)

self.io\_controller.write\_out(OutPorts.Out\_6, PortState.Off)

self.io\_controller.write\_out(OutPorts.Out\_7, PortState.Off)

self.io\_controller.write\_out(OutPorts.Out\_8, PortState.Off)

```

- \*\*Mục đích:\*\* Đóng tất cả cổng output

- \*\*Chi tiết:\*\*

- Set tất cả cổng output về Off

## 76. \*\*Hàm `set\_io\_result`\*\*

```python

def set\_io\_result(self, ret):

if ret == StepResult.PASS\_.value:

self.io\_controller.write\_out(OutPorts.Out\_2, PortState.Off)

time.sleep(0.02)

self.io\_controller.write\_out(OutPorts.Out\_3, PortState.Off)

time.sleep(0.02)

self.io\_controller.write\_out(OutPorts.Out\_1, PortState.On)

self.active\_output = OutPorts.Out\_1

elif ret == StepResult.WAIT.value:

self.io\_controller.write\_out(OutPorts.Out\_1, PortState.Off)

time.sleep(0.02)

self.io\_controller.write\_out(OutPorts.Out\_2, PortState.Off)

time.sleep(0.02)

self.io\_controller.write\_out(OutPorts.Out\_3, PortState.On)

self.active\_output = OutPorts.Out\_3

else:

self.io\_controller.write\_out(OutPorts.Out\_1, PortState.Off)

time.sleep(0.02)

self.io\_controller.write\_out(OutPorts.Out\_3, PortState.Off)

time.sleep(0.02)

self.io\_controller.write\_out(OutPorts.Out\_2, PortState.On)

self.active\_output = OutPorts.Out\_2

```

- \*\*Mục đích:\*\* Set kết quả lên cổng output

- \*\*Chi tiết:\*\*

- Nếu PASS: Bật Out\_1

- Nếu WAIT: Bật Out\_3

- Nếu FAIL: Bật Out\_2

- Cập nhật active\_output

Tiếp tục giải thích các hàm còn lại:

## 78. \*\*Hàm `on\_to\_off`\*\*

```python

def on\_to\_off(self, commands):

for cmd in commands:

index = list(InPorts).index(InPorts[cmd])

self.input\_labels[index].setStyleSheet("background-color: gray; color: white; padding: 5px;")

```

- \*\*Mục đích:\*\* Xử lý khi input chuyển từ On sang Off

- \*\*Chi tiết:\*\*

- Duyệt qua các command

- Lấy index của input từ enum InPorts

- Set style của label tương ứng thành màu xám

## 79. \*\*Hàm `off\_to\_on`\*\*

```python

def off\_to\_on(self, commands):

for cmd in commands:

index = list(InPorts).index(InPorts[cmd])

self.input\_labels[index].setStyleSheet("background-color: green; color: white; padding: 5px;")

```

- \*\*Mục đích:\*\* Xử lý khi input chuyển từ Off sang On

- \*\*Chi tiết:\*\*

- Duyệt qua các command

- Lấy index của input từ enum InPorts

- Set style của label tương ứng thành màu xanh lá

## 80. \*\*Hàm `sensor\_on`\*\*

```python

def sensor\_on(self):

self.is\_sensor\_on = True

print('sensor on')

```

- \*\*Mục đích:\*\* Xử lý khi sensor bật

- \*\*Chi tiết:\*\*

- Set flag is\_sensor\_on = True

- In thông báo

## 81. \*\*Hàm `sensor\_off`\*\*

```python

def sensor\_off(self):

self.is\_sensor\_on = False

print('sensor off')

```

- \*\*Mục đích:\*\* Xử lý khi sensor tắt

- \*\*Chi tiết:\*\*

- Set flag is\_sensor\_on = False

- In thông báo

## 82. \*\*Hàm `close\_light\_teaching`\*\*

```python

def close\_light\_teaching(self):

if not self.light.close():

return False

return True

```

- \*\*Mục đích:\*\* Đóng đèn trong chế độ Teaching

- \*\*Chi tiết:\*\*

- Gọi hàm close() của light controller

- Trả về kết quả đóng đèn

## 83. \*\*Hàm `read\_signal\_scanner`\*\*

```python

def read\_signal\_scanner(self, code):

if not self.is\_check\_sn:

if code == "CheckOCDU":

self.set\_trigger\_on()

return

if code is not None and code.startswith("S") and len(code) == 10:

if self.find\_series\_number(code):

self.data\_sn = code

self.hide\_sn\_failed()

else:

self.show\_sn\_failed(check\_repeat=True)

return

else:

self.show\_sn\_failed()

return

else:

if not self.check\_gen\_id(code):

self.show\_message\_scan\_gen()

self.is\_check\_gen\_id = False

return

self.is\_check\_gen\_id = True

```

- \*\*Mục đích:\*\* Xử lý tín hiệu từ scanner

- \*\*Chi tiết:\*\*

- Nếu không đang check SN:

- Nếu code là "CheckOCDU": Set trigger

- Nếu code là SN hợp lệ: Kiểm tra trùng lặp

- Nếu code không hợp lệ: Hiện thông báo lỗi

- Nếu đang check SN:

- Kiểm tra ID nhân viên

- Hiện thông báo nếu không hợp lệ

## 84. \*\*Hàm `show\_sn\_failed`\*\*

```python

def show\_sn\_failed(self, check\_repeat=False):

if hasattr(self, 'msg\_box\_show\_sn\_failed') and self.msg\_box\_show\_sn\_failed is not None:

self.hide\_sn\_failed()

self.msg\_box\_show\_sn\_failed = QMessageBox(self)

self.msg\_box\_show\_sn\_failed.setIcon(QMessageBox.Warning)

self.msg\_box\_show\_sn\_failed.setWindowTitle("Warning")

if check\_repeat:

self.msg\_box\_show\_sn\_failed.setText("Mã S/N đã được sử dụng, vui lòng thực hiện lại")

else:

self.msg\_box\_show\_sn\_failed.setText("Scan sai định dạng S/N, vui lòng thực hiện lại")

self.msg\_box\_show\_sn\_failed.setStandardButtons(QMessageBox.Cancel)

self.msg\_box\_show\_sn\_failed.show()

```

- \*\*Mục đích:\*\* Hiển thị thông báo lỗi SN

- \*\*Chi tiết:\*\*

- Đóng thông báo cũ nếu có

- Tạo QMessageBox mới

- Set icon và tiêu đề

- Set nội dung tùy theo loại lỗi

- Hiển thị thông báo

## 85. \*\*Hàm `hide\_sn\_failed`\*\*

```python

def hide\_sn\_failed(self):

if hasattr(self, 'msg\_box\_show\_sn\_failed') and self.msg\_box\_show\_sn\_failed is not None:

self.msg\_box\_show\_sn\_failed.close()

self.msg\_box\_show\_sn\_failed = None

```

- \*\*Mục đích:\*\* Ẩn thông báo lỗi SN

- \*\*Chi tiết:\*\*

- Đóng QMessageBox nếu có

- Set về None

## 86. \*\*Hàm `show\_message\_scan\_gen`\*\*

```python

def show\_message\_scan\_gen(self):

if hasattr(self, 'msg\_box\_show\_scan\_gen') and self.msg\_box\_show\_scan\_gen is not None:

self.hide\_message\_scan\_gen()

self.msg\_box\_show\_scan\_gen = QMessageBox(self)

self.msg\_box\_show\_scan\_gen.setIcon(QMessageBox.Warning)

self.msg\_box\_show\_scan\_gen.setWindowTitle("Warning")

self.msg\_box\_show\_scan\_gen.setText("Scan QR trên thẻ nhân viên để xác nhận")

self.msg\_box\_show\_scan\_gen.show()

```

- \*\*Mục đích:\*\* Hiển thị thông báo yêu cầu scan thẻ nhân viên

- \*\*Chi tiết:\*\*

- Đóng thông báo cũ nếu có

- Tạo QMessageBox mới

- Set icon, tiêu đề và nội dung

- Hiển thị thông báo

## 87. \*\*Hàm `hide\_message\_scan\_gen`\*\*

```python

def hide\_message\_scan\_gen(self):

if hasattr(self, 'msg\_box\_show\_scan\_gen') and self.msg\_box\_show\_scan\_gen is not None:

self.msg\_box\_show\_scan\_gen.close()

self.msg\_box\_show\_scan\_gen = None

```

- \*\*Mục đích:\*\* Ẩn thông báo scan thẻ nhân viên

- \*\*Chi tiết:\*\*

- Đóng QMessageBox nếu có

- Set về None

## 88. \*\*Hàm `recheck\_result`\*\*

```python

def recheck\_result(self):

msg\_box = QMessageBox(self)

msg\_box.setWindowTitle('Question')

msg\_box.setText('Kết quả là FAIL, bạn có muốn tiếp tục không?')

ok\_button = QPushButton('Xác nhận OK')

ng\_button = QPushButton('Xác nhận NG')

ng\_button.setStyleSheet("""

QPushButton {

background-color: red;

color: white;

}

QPushButton:hover {

background-color: darkred;

}

""")

msg\_box.addButton(ok\_button, QMessageBox.YesRole)

msg\_box.addButton(ng\_button, QMessageBox.NoRole)

msg\_box.setDefaultButton(ng\_button)

msg\_box.exec\_()

if msg\_box.clickedButton() == ok\_button:

self.is\_confirm\_ret = True

else:

self.is\_confirm\_ret = False

self.is\_check\_sn = True

self.show\_message\_scan\_gen()

```

- \*\*Mục đích:\*\* Xử lý khi cần xác nhận lại kết quả FAIL

- \*\*Chi tiết:\*\*

- Tạo QMessageBox với 2 nút

- Set style cho nút NG

- Nếu chọn OK: Set is\_confirm\_ret = True

- Nếu chọn NG: Set is\_confirm\_ret = False

- Set is\_check\_sn = True

- Hiện thông báo yêu cầu scan thẻ

## 89. \*\*Hàm `set\_text\_label`\*\*

```python

def set\_text\_label(self, label: QLabel, text):

label.setText(text)

```

- \*\*Mục đích:\*\* Set text cho label

- \*\*Chi tiết:\*\*

- Set text cho label được truyền vào

## 90. \*\*Hàm `check\_gen\_id`\*\*

```python

def check\_gen\_id(self, code):

if code is not None and re.fullmatch(r'(0[8-9]|[1-9][0-9])\d{6}', code):

self.hide\_message\_scan\_gen()

confirm = 'PASS' if self.is\_confirm\_ret else 'FAIL'

self.main\_logger.info(f'Gen: {code} xác nhận {confirm}')

return True

else:

return False

```

- \*\*Mục đích:\*\* Kiểm tra ID nhân viên

- \*\*Chi tiết:\*\*

- Kiểm tra code có đúng định dạng không

- Nếu đúng:

- Ẩn thông báo

- Ghi log

- Trả về True

- Nếu sai: Trả về False

## 91. \*\*Hàm `set\_trigger\_on`\*\*

```python

def set\_trigger\_on(self):

self.trigger\_on = True

```

- \*\*Mục đích:\*\* Set trigger bật

- \*\*Chi tiết:\*\*

- Set flag trigger\_on = True

## 92. \*\*Hàm `set\_trigger\_off`\*\*

```python

def set\_trigger\_off(self):

self.trigger\_on = False

```

- \*\*Mục đích:\*\* Set trigger tắt

- \*\*Chi tiết:\*\*

- Set flag trigger\_on = False

## 93. \*\*Hàm `start\_process`\*\*

```python

def start\_process(self):

time.sleep(0.2)

self.b\_start = True

threading.Thread(target=self.thread\_loop\_auto, daemon=True).start()

```

- \*\*Mục đích:\*\* Bắt đầu quá trình xử lý

- \*\*Chi tiết:\*\*

- Delay 0.2s

- Set flag b\_start = True

- Tạo thread mới để chạy thread\_loop\_auto

## 94. \*\*Hàm `stop\_process`\*\*

```python

def stop\_process(self):

self.b\_start = False

```

- \*\*Mục đích:\*\* Dừng quá trình xử lý

- \*\*Chi tiết:\*\*

- Set flag b\_start = False

## 95. \*\*Hàm `convert\_path`\*\*

```python

def convert\_path(self, relative\_path):

base\_path = os.getcwd()

absolute\_path = os.path.join(base\_path, relative\_path.replace("/", "\\"))

return absolute\_path

```

- \*\*Mục đích:\*\* Chuyển đổi đường dẫn tương đối thành tuyệt đối

- \*\*Chi tiết:\*\*

- Lấy đường dẫn hiện tại

- Nối với đường dẫn tương đối

- Thay dấu / thành \

- Trả về đường dẫn tuyệt đối

## 96. \*\*Hàm `get\_selected\_items`\*\*

```python

def get\_selected\_items(self):

selected\_row = self.table\_widget\_database.currentRow()

if selected\_row != -1:

item\_col2 = self.table\_widget\_database.item(selected\_row, 2)

item\_col3 = self.table\_widget\_database.item(selected\_row, 3)

src = item\_col2.text() if item\_col2 else "None"

dst = item\_col3.text() if item\_col3 else "None"

if not src or not dst:

return

img\_src = cv2.imread(src)

img\_dst = cv2.imread(dst)

self.show\_image(self.data\_image\_input\_canvas, img\_src)

self.show\_image(self.data\_image\_output\_canvas, img\_dst)

```

- \*\*Mục đích:\*\* Xử lý khi chọn item trong bảng database

- \*\*Chi tiết:\*\*

- Lấy hàng được chọn

- Lấy đường dẫn ảnh từ cột 2 và 3

- Đọc ảnh từ đường dẫn

- Hiển thị ảnh lên canvas

## 97. \*\*Hàm `load\_database`\*\*

```python

def load\_database(self, file\_path=None):

if not file\_path:

return

try:

conn = sqlite3.connect(file\_path)

cursor = conn.cursor()

cursor.execute(f"SELECT \* FROM users")

rows = cursor.fetchall()

columns = [desc[0] for desc in cursor.description]

if "id" in columns:

id\_index = columns.index("id")

columns.pop(id\_index)

rows = [tuple(value for i, value in enumerate(row) if i != id\_index) for row in rows]

self.table\_widget\_database.setRowCount(len(rows))

self.table\_widget\_database.setColumnCount(len(columns))

self.table\_widget\_database.setHorizontalHeaderLabels(columns)

for row\_idx, row\_data in enumerate(rows):

for col\_idx, col\_data in enumerate(row\_data):

self.table\_widget\_database.setItem(row\_idx, col\_idx, QTableWidgetItem(str(col\_data)))

conn.close()

except Exception as e:

print(f"Lỗi khi load dữ liệu: {e}")

```

- \*\*Mục đích:\*\* Load dữ liệu từ database

- \*\*Chi tiết:\*\*

- Kết nối database

- Lấy tất cả dữ liệu từ bảng users

- Bỏ cột id

- Hiển thị dữ liệu lên table widget

## 98. \*\*Hàm `load\_classes\_ai`\*\*

```python

def load\_classes\_ai(self, table: QTableWidget, file\_path: str):

file\_path = f'res/Model\_AI/{file\_path}'

try:

with open(file\_path, 'r', encoding='utf-8') as f:

self.lines\_classes = f.read().splitlines()

self.row\_count = len(self.lines\_classes)

table.setColumnCount(6)

table.setHorizontalHeaderLabels(["STT", "Label", "Số lượng", "Check", "Result", "Error"])

table.setRowCount(self.row\_count)

for i, line in enumerate(self.lines\_classes):

try:

left, count = line.rsplit(" ", 1)

index, label = left.split("\_", 1)

count = int(count)

except ValueError:

print(f"⚠️ Dòng sai định dạng: {line}")

continue

item\_index = QTableWidgetItem(index)

item\_index.setFlags(item\_index.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 0, item\_index)

item\_label = QTableWidgetItem(label)

item\_label.setFlags(item\_label.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 1, item\_label)

item\_count = QTableWidgetItem(str(count))

item\_count.setFlags(item\_count.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 2, item\_count)

item\_check = QTableWidgetItem("")

item\_check.setFlags(item\_check.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 3, item\_check)

item\_return = QTableWidgetItem("")

item\_return.setFlags(item\_return.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 4, item\_return)

item\_error = QTableWidgetItem("")

item\_error.setFlags(item\_error.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 5, item\_error)

table.setSpan(0, 5, self.row\_count, 1)

except Exception as e:

print(f"❌ Lỗi khi đọc file: {e}")

```

- \*\*Mục đích:\*\* Load danh sách classes AI

- \*\*Chi tiết:\*\*

- Đọc file classes

- Tạo table widget với 6 cột

- Parse từng dòng để lấy index, label và count

- Thêm dữ liệu vào table

- Set các item không thể edit

- Merge cột Error

## 99. \*\*Hàm `load\_result\_classes`\*\*

```python

def load\_result\_classes(self, table, label\_counts: dict, ret: str, error: str):

labels = list(label\_counts.keys())

count\_status\_list = [(info["count"], info["status"]) for info in label\_counts.values()]

missing\_materials = []

for i, line in enumerate(self.lines\_classes):

item\_check = QTableWidgetItem(str(count\_status\_list[i][0]))

item\_check.setFlags(item\_check.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 3, item\_check)

item\_return = QTableWidgetItem(count\_status\_list[i][1])

item\_return.setFlags(item\_return.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 4, item\_return)

item\_error = QTableWidgetItem(error)

item\_error.setFlags(item\_error.flags() & ~Qt.ItemIsEditable)

table.setItem(i, 5, item\_error)

if count\_status\_list[i][1].upper() == "FAIL":

item = table.item(i, 4)

if item:

item.setBackground(QBrush(QColor("#FF3B30")))

item.setForeground(QBrush(QColor("#FFFFFF")))

if item\_return:

item\_return.setForeground(QBrush(QColor("#FFFFFF")))

item\_return.setBackground(QBrush(QColor("#FF3B30")))

missing\_materials.append(labels[i])

else:

item = table.item(i, 4)

if item:

item.setBackground(QBrush(QColor("#34C759")))

item.setForeground(QBrush(QColor("#FFFFFF")))

if item\_return:

item\_return.setForeground(QBrush(QColor("#FFFFFF")))

item\_return.setBackground(QBrush(QColor("#34C759")))

table.setSpan(0, 5, self.row\_count, 1)

with self.confirm\_lock:

if not self.is\_confirm\_ret:

if missing\_materials:

QMessageBox.warning(

self,

"Thiếu vật liệu",

"Thiếu các vật liệu sau:\n- " + "\n- ".join(missing\_materials) +

"\n\nVui lòng kiểm tra lại và so sánh với WI."

)

```

- \*\*Mục đích:\*\* Load kết quả kiểm tra lên table

- \*\*Chi tiết:\*\*

- Lấy danh sách labels và kết quả

- Thêm dữ liệu vào table

- Set màu cho các item:

- FAIL: Màu đỏ

- PASS: Màu xanh lá

- Merge cột Error

- Hiện thông báo nếu có vật liệu thiếu

## 100. \*\*Hàm `set\_disable\_auto`\*\*

```python

def set\_disable\_auto(self, ret):

self.but\_open\_camera\_teaching.setDisabled(ret)

self.but\_connect\_server.setDisabled(ret)

self.but\_open\_io\_controller.setDisabled(ret)

self.but\_connect\_scanner.setDisabled(ret)

self.but\_open\_com\_send\_data.setDisabled(ret)

self.combo\_box\_feature\_camera.setDisabled(ret)

self.combo\_box\_id\_camera.setDisabled(ret)

self.combo\_box\_name\_camera.setDisabled(ret)

self.combo\_box\_com\_controller\_2.setDisabled(ret)

self.combo\_box\_com\_controller\_3.setDisabled(ret)

self.combo\_box\_com\_controller.setDisabled(ret)

self.combo\_box\_com\_controller\_4.setDisabled(ret)

self.combo\_box\_baudrate\_controller.setDisabled(ret)

self.combo\_box\_baudrate\_controller\_2.setDisabled(ret)

self.combo\_box\_baudrate\_controller\_3.setDisabled(ret)

self.combo\_box\_baudrate\_controller\_4.setDisabled(ret)

self.combo\_box\_model\_ai.setDisabled(ret)

self.combo\_box\_classes\_ai.setDisabled(ret)

self.line\_edit\_ip\_server.setDisabled(ret)

self.line\_edit\_port\_server.setDisabled(ret)

self.line\_edit\_threshold.setDisabled(ret)

self.line\_edit\_time\_delay\_io.setDisabled(ret)

```

- \*\*Mục đích:\*\* Enable/Disable các widget trong chế độ Auto

- \*\*Chi tiết:\*\*

- Set trạng thái disabled cho các button

- Set trạng thái disabled cho các combo box

- Set trạng thái disabled cho các line edit

## 101. \*\*Hàm `write\_log\_async`\*\*

```python

def write\_log\_async(self, mat, result, code):

mat\_copy = copy.deepcopy(mat)

result\_copy = copy.deepcopy(result)

code\_copy = code

def log\_task():

try:

src\_path = self.main\_logger.log\_image('Source', mat\_copy, code\_copy, result\_copy.ret)

src\_convert = self.convert\_path(src\_path)

path\_dst = self.combo\_box\_model\_name\_teaching.currentText()

dst\_path = self.main\_logger.log\_image(path\_dst, result\_copy.output\_image,

code\_copy, result\_copy.ret, log\_csv=True)

dst\_convert = self.convert\_path(dst\_path)

self.db.add\_entry(self.log\_table, Model=path\_dst, SeriesNumber=result\_copy.seri,

Source=src\_convert, Destination=dst\_convert,

Return=result\_copy.ret, Error=result\_copy.error)

except Exception as e:

self.main\_logger.error(f"Error writing log: {e}")

threading.Thread(target=log\_task, daemon=True).start()

```

- \*\*Mục đích:\*\* Ghi log bất đồng bộ

- \*\*Chi tiết:\*\*

- Tạo bản sao của dữ liệu

- Tạo thread mới để ghi log

- Ghi log ảnh gốc

- Ghi log ảnh kết quả

- Thêm entry vào database

## 102. \*\*Hàm `get\_shapes`\*\*

```python

def get\_shapes(self, canvas: Canvas):

shapes: list[MyShape] = canvas.shapes

shape\_config = {}

for s in shapes:

shape\_config[s.label] = s.cvBox

print(shape\_config)

return shape\_config

```

- \*\*Mục đích:\*\* Lấy cấu hình shapes từ canvas

- \*\*Chi tiết:\*\*

- Lấy danh sách shapes từ canvas

- Tạo dictionary với key là label và value là box

- Trả về dictionary

## 103. \*\*Hàm `set\_shapes`\*\*

```python

def set\_shapes(self, canvas: Canvas, shape\_config):

canvas.clear()

for label in shape\_config:

s = self.new\_shape(label, shape\_config[label])

canvas.shapes.append(s)

```

- \*\*Mục đích:\*\* Set shapes lên canvas

- \*\*Chi tiết:\*\*

- Xóa canvas

- Duyệt qua shape\_config

- Tạo shape mới cho mỗi label

- Thêm shape vào canvas

## 104. \*\*Hàm `new\_shape`\*\*

```python

def new\_shape(self, label, box):

s = MyShape(label)

x, y, w, h = box

s.points = [

QPointF(x, y),

QPointF(x + w, y),

QPointF(x + w, y + h),

QPointF(x, y + h)

]

return s

```

- \*\*Mục đích:\*\* Tạo shape mới

- \*\*Chi tiết:\*\*

- Tạo MyShape với label

- Tính toán 4 điểm từ box

- Set points cho shape

- Trả về shape

## 105. \*\*Hàm `find\_database`\*\*

```python

def find\_database(self):

try:

conn = sqlite3.connect(self.file\_path\_database)

cursor = conn.cursor()

query = "SELECT \* FROM users WHERE Time BETWEEN ? AND ?"

params = [

self.date\_time\_from.dateTime().toString("yyyy-MM-dd HH:mm:ss"),

self.date\_time\_to.dateTime().toString("yyyy-MM-dd HH:mm:ss")

]

result\_filter = self.combo\_box\_result\_data.currentText()

if result\_filter != "ALL":

query += " AND Return = ?"

params.append(result\_filter)

model\_filter = self.combo\_box\_model\_name\_data.currentText().strip()

if model\_filter != "ALL":

query += " AND Model = ?"

params.append(model\_filter)

keyword\_filter = self.line\_edit\_keyword\_data.text().strip()

if keyword\_filter:

query += " AND ("

keyword\_conditions = []

for column in ["SeriesNumber", "Source", "Destination", "Return", "Error"]:

keyword\_conditions.append(f"{column} LIKE ?")

params.append(f"%{keyword\_filter}%")

query += " OR ".join(keyword\_conditions) + ")"

cursor.execute(query, params)

rows = cursor.fetchall()

columns = [desc[0] for desc in cursor.description]

if "id" in columns:

id\_index = columns.index("id")

columns.pop(id\_index)

rows = [tuple(value for i, value in enumerate(row) if i != id\_index) for row in rows]

self.table\_widget\_database.setRowCount(len(rows))

self.table\_widget\_database.setColumnCount(len(columns))

self.table\_widget\_database.setHorizontalHeaderLabels(columns)

for row\_idx, row\_data in enumerate(rows):

for col\_idx, col\_data in enumerate(row\_data):

self.table\_widget\_database.setItem(row\_idx, col\_idx, QTableWidgetItem(str(col\_data)))

conn.close()

except Exception as e:

print(f"Lỗi khi tìm kiếm dữ liệu: {e}")

```

- \*\*Mục đích:\*\* Tìm kiếm trong database

- \*\*Chi tiết:\*\*

- Tạo câu query với các điều kiện:

- Thời gian

- Kết quả

- Model

- Từ khóa

- Thực thi query

- Hiển thị kết quả lên table widget

## 106. \*\*Hàm `find\_series\_number`\*\*

```python

def find\_series\_number(self, series\_number):

try:

conn = sqlite3.connect(self.file\_path\_database)

cursor = conn.cursor()

query = "SELECT \* FROM users WHERE SeriesNumber = ?"

cursor.execute(query, (series\_number,))

rows = cursor.fetchall()

conn.close()

if not rows:

return True

else:

return False

except Exception as e:

print(f"Lỗi khi tìm kiếm SeriesNumber: {e}")

return False

```

- \*\*Mục đích:\*\* Kiểm tra SN có tồn tại trong database không

- \*\*Chi tiết:\*\*

- Tìm SN trong database

- Nếu không tìm thấy: Trả về True

- Nếu tìm thấy: Trả về False

## 107. \*\*Hàm `show\_effect`\*\*

```python

def show\_effect(self, dt=3):

self.progressBar.setValue(0)

self.progressBar.setVisible(True)

for i in range(1, 100):

QTimer.singleShot(dt\*i, lambda value=i: self.progressBar.setValue(value))

```

- \*\*Mục đích:\*\* Hiển thị hiệu ứng loading

- \*\*Chi tiết:\*\*

- Reset progress bar

- Hiển thị progress bar

- Tăng giá trị progress bar dần dần

## 108. \*\*Hàm `hide\_effect`\*\*

```python

def hide\_effect(self, timeout=500):

QTimer.singleShot(10, partial(self.progressBar.setValue, 100))

QTimer.singleShot(timeout, partial(self.progressBar.setVisible, False))

```

- \*\*Mục đích:\*\* Ẩn hiệu ứng loading

- \*\*Chi tiết:\*\*

- Set progress bar lên 100%

- Ẩn progress bar sau timeout

## 109. \*\*Hàm `reload\_qss`\*\*

```python

def reload\_qss(self):

with open("res/Style/light\_mode.qss", "r", encoding="utf-8") as file:

app.setStyleSheet(file.read())

```

- \*\*Mục đích:\*\* Load lại file CSS

- \*\*Chi tiết:\*\*

- Đọc file CSS

- Áp dụng style cho ứng dụng

## 110. \*\*Hàm `closeEvent`\*\*

```python

def closeEvent(self, event):

reply = QMessageBox.question(self, 'Question', 'Are You want to quit?',

QMessageBox.Yes|QMessageBox.No, QMessageBox.No)

if reply == QMessageBox.Yes:

if self.is\_open\_camera:

self.close\_camera\_teaching()

if self.is\_connect\_server:

self.close\_server\_teaching()

if self.io\_controller:

if self.io\_controller.is\_open():

self.close\_all\_io\_controller()

self.set\_io\_result(StepResult.WAIT.value)

self.io\_controller.close()

if not self.is\_resetting:

self.saveSettings()

else:

self.restartApp()

event.accept()

else:

event.ignore()

```

- \*\*Mục đích:\*\* Xử lý khi đóng ứng dụng

- \*\*Chi tiết:\*\*

- Hiện hộp thoại xác nhận

- Nếu đồng ý:

- Đóng camera nếu đang mở

- Đóng server nếu đang kết nối

- Đóng IO controller nếu đang mở

- Lưu cài đặt hoặc khởi động lại

- Nếu không: Bỏ qua sự kiện

## 111. \*\*Hàm `restartApp`\*\*

```python

def restartApp(self):

python = sys.executable

subprocess.Popen([python] + sys.argv)

sys.exit(0)

```

- \*\*Mục đích:\*\* Khởi động lại ứng dụng

- \*\*Chi tiết:\*\*

- Lấy đường dẫn Python

- Tạo process mới với các tham số hiện tại

- Thoát process hiện tại

Đây là toàn bộ các hàm trong file main.py. Mỗi hàm đều có một mục đích và chức năng riêng, góp phần tạo nên một ứng dụng hoàn chỉnh với nhiều tính năng như:

- Giao diện người dùng đa dạng

- Xử lý ảnh và AI

- Kết nối với các thiết bị ngoại vi

- Quản lý database

- Ghi log và báo cáo

- Xử lý lỗi và ngoại lệ