PLC Automation System Design Overview

Introduction

The automated cell under consideration is designed to dispense, sort and dispatch workpieces of different sizes, colors and materials. A Programmable Logic Controller (PLC) orchestrates the sequence of operations by actuating a series of pneumatic grippers, linear actuators, a turntable and a press while monitoring sensors and push buttons. To achieve safe and deterministic control, the system is decomposed into functional modules connected through a centralized I/O mapping. The following report describes the major components, the I/O mapping, subsystem organization and the modular control strategy implemented.

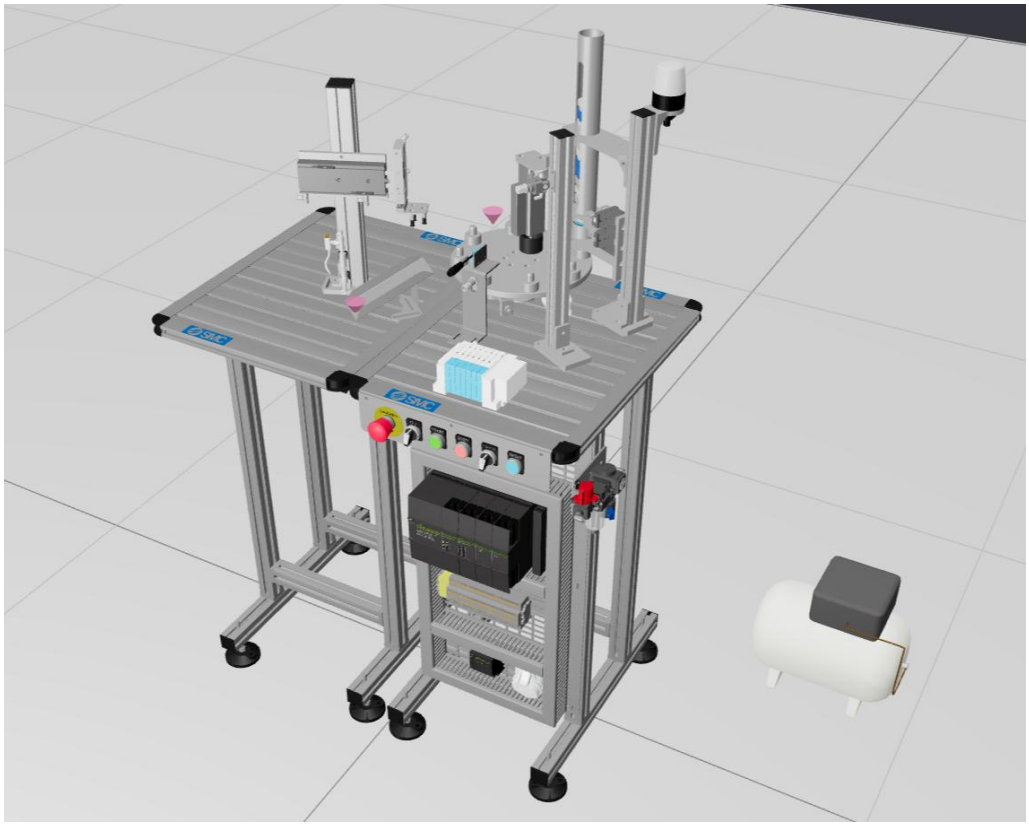


Figure 1: SMC System 3D Model

Component Identification

PLC Hardware

- **PLC Rack** – provides the backplane and power for input/output cards.
- **Digital input card** – 16-channel input (in_00 → in_15) used for reading push buttons, switches and sensors.
- **Digital output card** – 16-channel output (out_00 → out_15) used to drive solenoids and coils controlling pneumatic valves and actuators.
- **Analogue input card** – four channels (C1 → C4) measuring integer values; here the press height sensor is connected to channel IW200.
- **Analogue output card** – four channels (out_00 → out_03) generating integer values; the beacon light uses channel QB200.

Mechanical Components

- **Valve block** – distributes compressed air to actuators based on coil activation.
- **A_1_Actuator and A_2_Actuator** – double-acting pneumatic grippers forming the dispenser.
- **B_Actuator** – double-acting stroke-reading cylinder (CE1B20-50) used by the press module to measure workpiece height.
- **C_Actuator** – double-acting linear actuator controlling part of the turntable.
- **D_Actuator_1** – double-acting linear actuator (CDQSB12-30DCM) cooperating with C_Actuator to rotate the turntable.
- **E_Actuator and F_Actuator** – double-acting linear actuators (CXSM15-100 and CXSM10-50) used by the evacuation manipulator; E_Actuator provides horizontal movement, F_Actuator provides vertical movement.
- **Vacuum pad** – picks and drops workpieces when pressurized or depressurized.
- **Inductive sensor** – detects metallic workpieces (grey).
- **Diffuse sensor** – detects dark colours (blue).

Workpieces

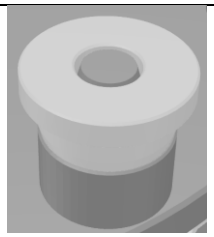
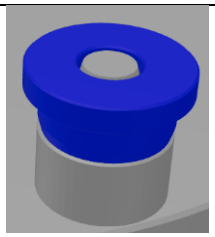
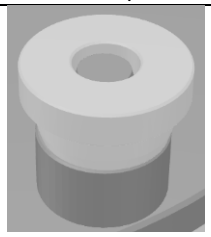
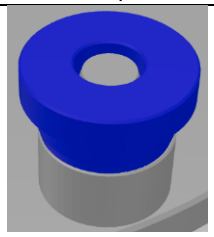
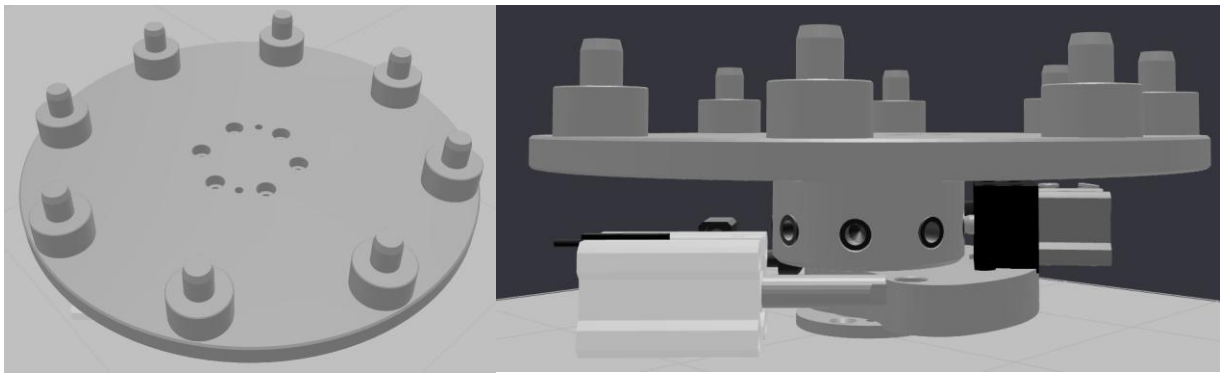
 1. White Small (Non-metal)	 2. Blue Small (Non-metal)	 3. Grey Small (Metal)
 4. White Large (Non-metal)	 5. Blue Large (Non-metal)	 6.Grey Large (Metal)

Table 1: Workpiece 3D Models

Sub-System Identification

1. **Turntable subsystem** – moves workpieces to different stations using C_Actuator and D_Actuator_1.



A. Top View **B. Top View**

Figure 2: Turntable 3D Model

2. **Dispenser subsystem** – uses A_1 and A_2 grippers to drop one workpiece onto the turntable.

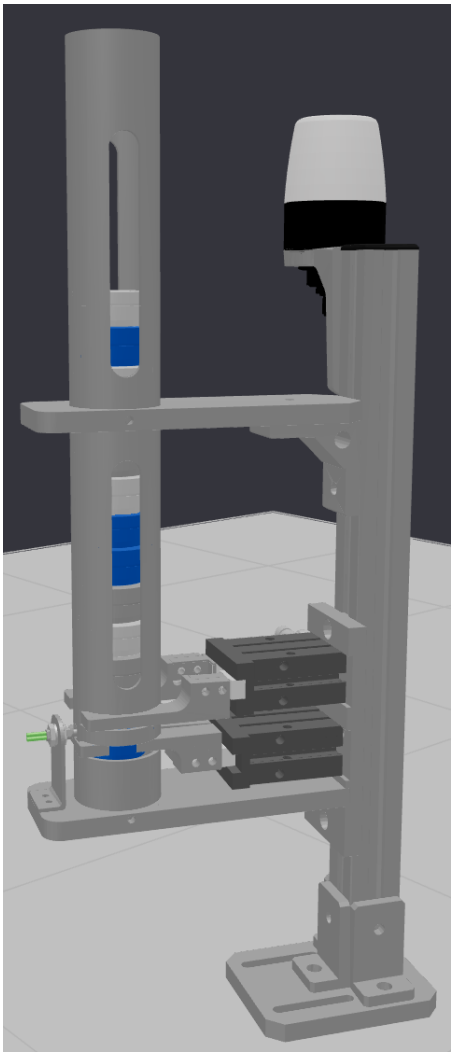


Figure 3: Dispenser 3D Model

3. **Press subsystem** – lowers the B_Actuator to measure height; the measured analogue value determines large versus small size.

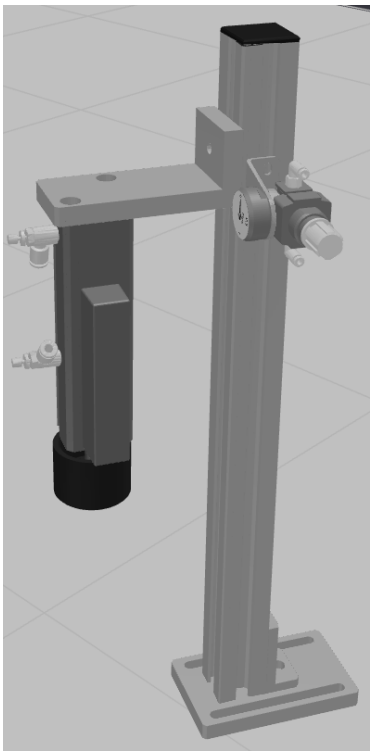


Figure 4: Press 3D Model

4. **Evacuation manipulator subsystem** – E_Actuator moves horizontally, F_Actuator moves vertically and a vacuum pad picks or drops workpieces.

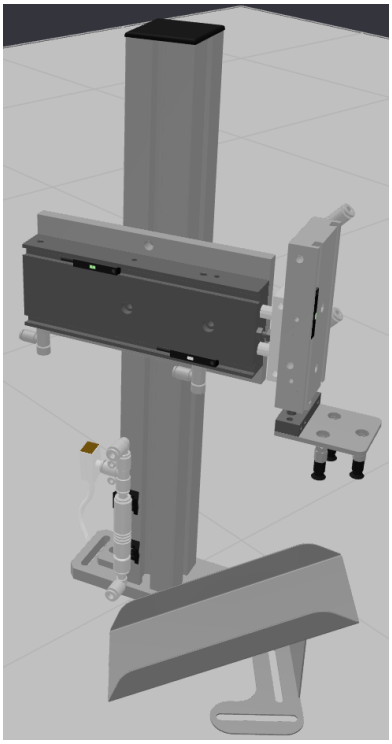


Figure 5: Evacuation Manipulator 3D Model

5. **Sensor subsystem** – inductive sensor identifies metallic (grey) workpieces; diffuse sensor identifies dark colours (blue); if neither sensor activates, the workpiece is white by default.

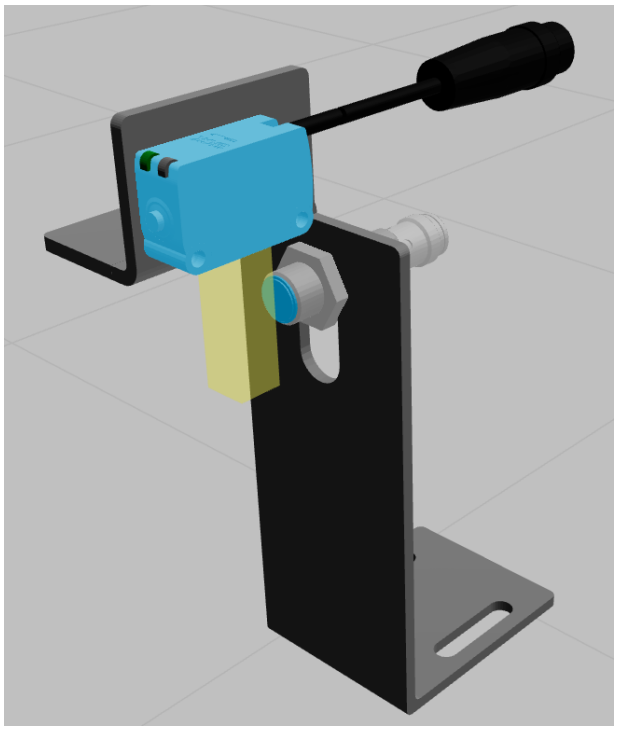


Figure 6: Sensors 3D Model

I/O Mapping

A clear mapping of digital and analogue signals is crucial for deterministic control. Digital inputs and outputs are grouped in a word and accessed as bits; similarly, analogue signals are mapped to integer channels.

Digital Inputs

Bit (in_xx)	Function	Notes
in_00	**Start push button**	Starts a cycle
in_01	**Stop push button**	Stops/halt cycle
in_02	**Auto–manual switch**	Mode selection
in_03	**Reset push button**	Resets system
in_05	**Inductive sensor**	Detects metal
in_11	**Emergency button**	Safety interlock
in_12	**Diffuse sensor**	Detects dark colour
Others	Unused or reserved for expansion	—

Table 2: Digital Inputs (IW0 – bytes inverted)

Analogue Input

- **IW200 (INT)** – linear voltage signal from the press (B_Actuator) representing piston stroke; used to measure workpiece height.

Digital Outputs

Bit (out_xx)	Function
out_01	**Open/close dispenser** (controls grippers A_1 & A_2)
out_02	**Press/release press** (controls B_Actuator)
out_03	**Extend/retract C_Actuator** (turntable)
out_04	**Extend/retract D_Actuator_1** (turntable)
out_05	**Extend E_Actuator**
out_06	**Retract E_Actuator**
out_07	**Extend/retract F_Actuator**
out_08	**Pressurise vacuum pad** (pick workpiece)
out_09	**Depressurise vacuum pad** (release workpiece)

Table 3: Digital Outputs (QW0 – bytes inverted)

Analogue Output

- **QB200 (INT)** – controls beacon light indicating system mode.

Modular Control Design

The control program is organised into independent modules, each encapsulating the logic for a specific subsystem. Modules expose triggers and return completion flags, enabling a higher-level scheduler to coordinate their execution. Timers enforce non-retriggerable windows to avoid conflicting commands.

Output Controller Module

To centralise and protect all actuator commands, an output controller provides a single interface for writing to solenoid outputs. It checks the emergency button (EB) and higher-level interlocks before energising coils. Parallel requests from different modules are merged, and conflicting commands (e.g. extend and retract signals simultaneously) are prevented. This ensures that safety conditions are globally enforced and simplifies debugging.

Dispense Module

When triggered, the dispense module opens the dispenser for a predefined window time, allowing exactly one workpiece to drop onto the turntable. It then closes the dispenser. A non-retriggerable timer prevents overlapping activations; subsequent triggers are ignored until the module has completed its window.

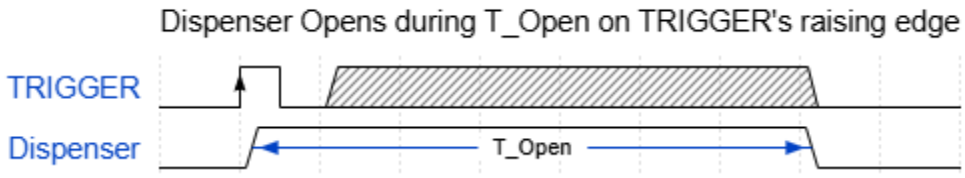


Figure 7: Dispenser Module Timing Diagram

Press Module

The press module controls the B_Actuator to measure workpiece height:

1. **Activation:** When triggered, a timer prevents reactivation during the measurement window.

2. **Lowering:** The B_Actuator extends until either it reaches its stroke limit or it encounters a workpiece; detection is implied by change in the analogue signal's rate of change or a predefined timeout.

3. **Hold:** The press holds for a specified period to allow analogue stabilisation.

4. **Measurement:** The analogue input IW200 is sampled. A calibration table relates raw integer values to physical height; the value is compared to thresholds to classify the workpiece as **large**, **small**, or **none** if not fall in into any thresholds.

5. **Retract:** The B_Actuator retracts and the module resets, ready for the next trigger.

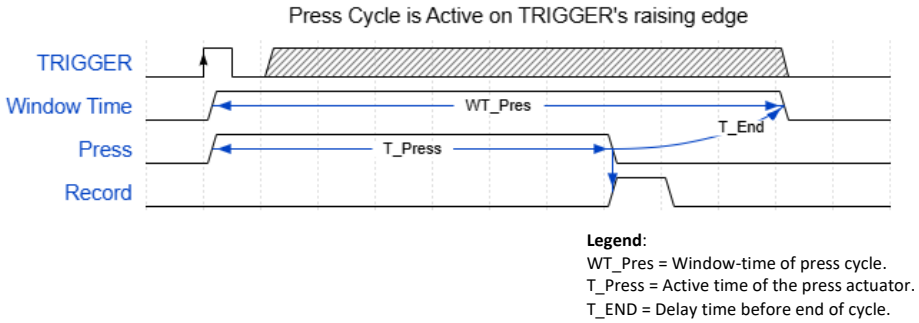


Figure 8: Timing Diagram for Press Cycle.

Workpiece Colour Identifier Module

This module determines the colour of the workpiece after measurement:

- **Inductive sensor active:** indicates a metallic workpiece; therefore the colour is **grey**.
- **Diffuse sensor active (no inductive):** indicates a dark colour; the workpiece is **blue**.
- **No sensor active:** by default, the workpiece is **white**.

The module outputs the colour code and a ready flag.

Workpiece Type Identifier Module

Using the size result from the press module and the colour result from the colour identifier module, this module assigns the workpiece to one of six types (White large, White small, Blue large, etc.). The result is used later to decide whether the piece should be picked or left on the turntable.

Pick-and-Drop Module

This module executes a pick-and-drop cycle using E_Actuator, F_Actuator and the vacuum pad. The steps are time-sequenced and controlled by internal timers:

1. **Extend E_Actuator** (horizontal movement).

2. **Extend F_Actuator** for a preconfigured `f_time` to lower the vacuum pad.

3. **Pressurise vacuum pad** to pick the workpiece.

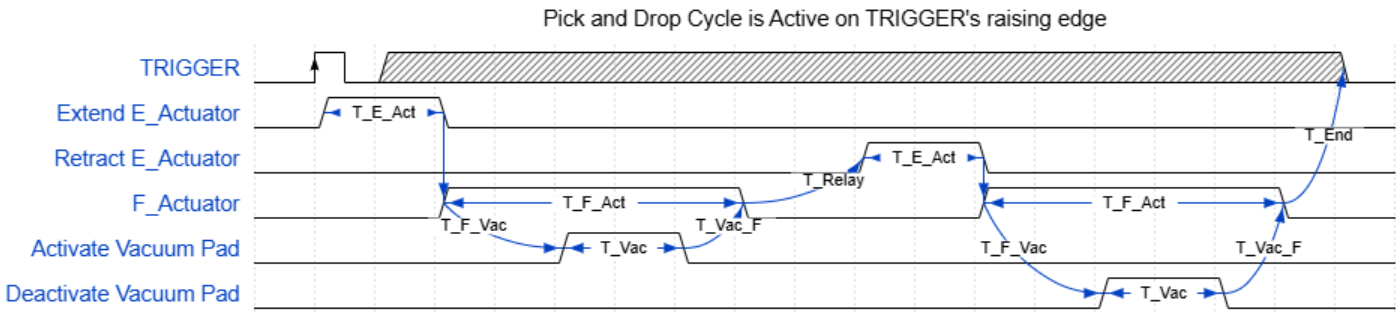
4. **Retract F_Actuator** after `f_time`.

5. **Retract E_Actuator** to bring the workpiece to the drop location.

6. **Extend F_Actuator** again and **depressurise vacuum pad** to drop the workpiece.

7. **Retract F_Actuator** and set a completion flag.

A running flag prevents reentry until the entire sequence finishes.



Legend:
T_E_Act = Time to extend E_Actuator.
T_F_Act = Time to extend F_Actuator.
T_F_Vac = Relay time before activating vacuum pad.
T_Vac = Time to activate/deactivate vacuum pad.
T_Vac_F = Relay time before retracting F_Actuator.
T_Relay = Relay time before retracting E_Actuator.
T_End = Relay time before cycle end.

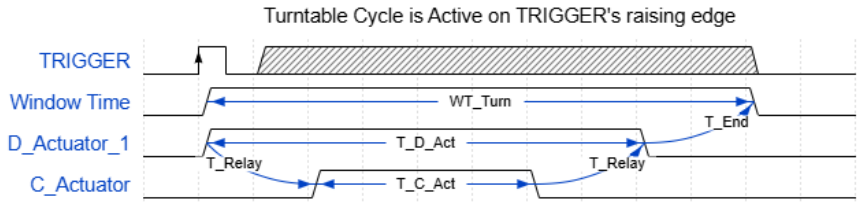
Figure 9: Timing Diagram of Pick and Drop Cycle.

Turntable Module

To rotate the turntable, C_Actuator and D_Actuator_1 must be energised in sequence. Two timers manage overlapping windows:

- **D_Actuator_1** is energised first and remains on for the entire rotation window.
- **C_Actuator** is energised inside this window to provide the mechanical rotation.
- After both timers expire, the actuators are de-energised in the reverse order.

The module cannot be retriggered until the rotation cycle completes.



Legend:
WT_Turn = Window-time of turntable cycle.
T_Relay = Relay latency between two activators.
T_D_Act = Active time of D_Actuator_1.
T_C_Act = Active time of C_Actuator.
T_END = Delay time before end of cycle.

Figure 10: Timing diagram of turntable cycle.

System Cycle Module

The system cycle module orchestrates the entire process. After system startup and reset, it enters a cyclic state machine where the following actions occur on each “tick” (coordinated by delay timers):

1. **Turntable rotation** is triggered first to present an empty station beneath the dispenser.

2. **Dispense** is triggered to drop a workpiece.

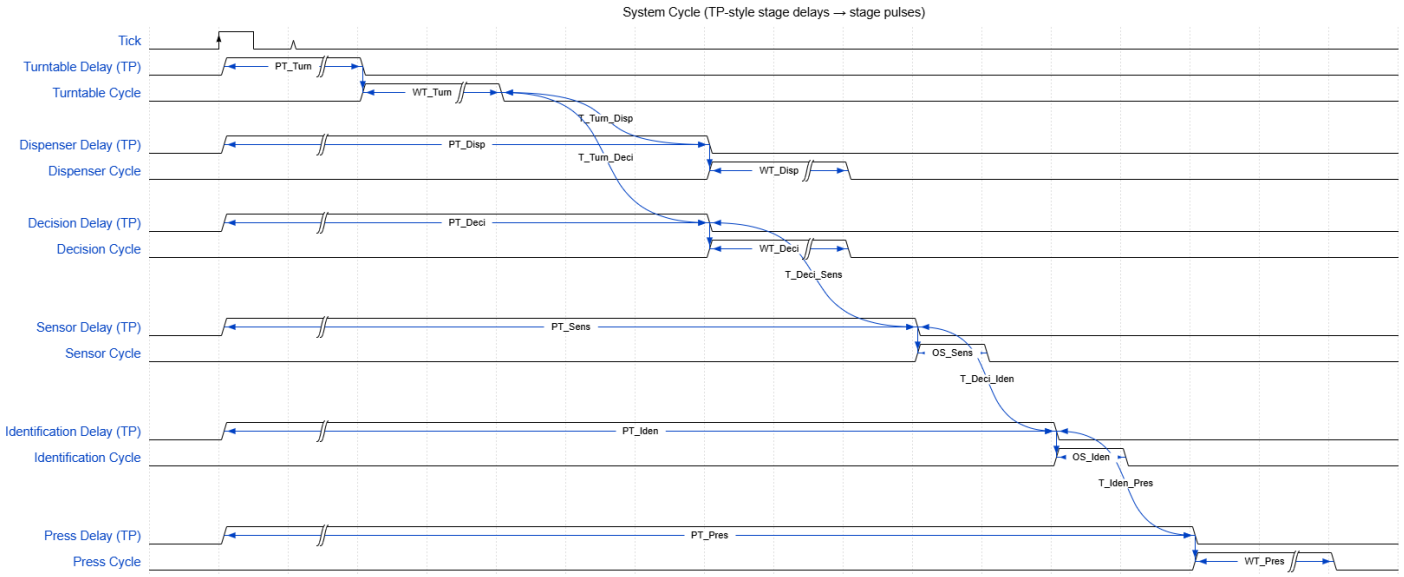
3. **Press measurement** is triggered after the piece arrives beneath the press.

4. **Colour identification** is triggered to determine the workpiece’s colour.

5. **Type identification** is triggered based on size and colour.

6. **Pick-and-drop** is triggered if the workpiece type is on a whitelist (e.g. only certain types are evacuated).

After all delays expire, the cycle restarts. Inter-module delays ensure sequential execution and avoid collisions. The system runs continuously as long as it remains in **RUNNING** mode.



Legend:
TP = pulse timer (Output high for preset-time (PT) once on rising edge.
PT_* = preset-time of each stage delay.
WT_* = window-time of each stage cycle.
S_* = one-shot stage.
T_X_Y = latency from X stage pulse to Y stage delay start.

Figure 11: Timing Diagram for the Entire System Cycle

Reset Module

When activated, all modules are reset to their initial state: timers are cleared, actuators are commanded to safe positions (retracted or off), the pick-and-drop running flag is cleared and internal variables are set to default. This module ensures the system can recover from faults or manual interventions.

Safety Module and State Machine

Safety is enforced through a state machine controlling the system's high-level mode and beacon light. The following states include two additional intermediate states to handle orderly resetting and stopping:

- **START_UP:** initial state; the system is powered but cannot start. The beacon blinks red. Pressing the reset button does not immediately start the system; instead it enters **RESETTING**.
- **RESETTING:** runs the reset module to return variables and actuators to their initial state. During this timed period the beacon can blink yellow to indicate reset-in-progress. Once the reset completes the state machine automatically moves to **IDLE**.
- **IDLE:** the system awaits a start command. The beacon shows solid yellow. Pressing the start button transitions to **RUNNING**. Pressing the reset button again moves to **RESETTING**, after which it returns to **IDLE**.
- **RUNNING:** normal operation. The beacon is green. Pressing the stop button does not halt immediately; it transitions to **HALTING**, a timed state that allows the current cycle to finish safely.
- **HALTING:** a delay timer runs to let the ongoing cycle complete. The beacon can blink yellow slowly to show stop-in-progress. After the timer expires the system enters **HALT**.
- **HALT:** the cycle has stopped in a controlled manner. The beacon blinks yellow slowly. Pressing the start button returns the system to **RUNNING**. Pressing the reset button moves to **RESETTING**, after which the state machine returns to **IDLE**.
- **WARNING:** abnormal conditions (e.g. sensor mismatch) cause entry to this state. The beacon blinks yellow quickly. Pressing the stop button moves to **HALTING**, and after the delay the machine stops in **HALT**.
- **FAULT:** triggered by the emergency button. The beacon shows solid red. All outputs (except the beacon) are de-energised immediately. Only resetting the emergency and pressing reset will restore the system to **START_UP**; a subsequent reset will pass through **RESETTING** before reaching **IDLE**.

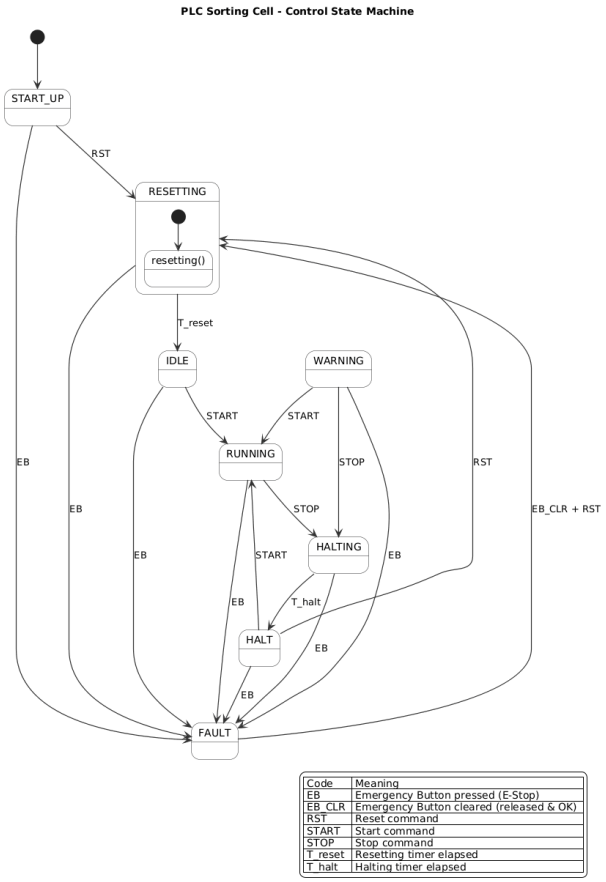


Figure 12: State Machine Diagram

Implementation Considerations

- **Programming language:** The PLC logic can be implemented in Structured Text (ST) or Ladder Diagram (LD). Modules are best expressed in ST as functions or function blocks with clear inputs and outputs. A global variable list (GVL) defines I/O and state machine variables.
- **Timers:** Each module uses on-delay timers (TON) to enforce activation windows and off-delay timers (TOF) to cleanly retract actuators. Non-retriggerable behaviour is achieved by gating triggers through timer enable conditions.
- **Calibration:** The analogue value ranges for “large” versus “small” workpieces must be calibrated experimentally and stored in constants. Hysteresis can be added to avoid misclassification due to noise.
- **Whitelisting:** The system cycle module references a lookup table of allowed workpiece types. Only whitelisted types trigger the pick-and-drop operation; others stay on the turntable for subsequent cycles.
- **Debugging aids:** The output controller module simplifies debugging by centralising coil commands. Each module returns status bits (running, completed, error) that can be displayed on an HMI for diagnostics.
- **Future expansion:** Unused inputs and outputs on the digital cards allow additional sensors or actuators to be added without hardware changes.

Demonstration Links

1. **Task 1 – Sorting metallic objects**

- **Goal:** Evacuation manipulator picks only metallic objects from the turntable and places them on the slide.
- **Demo URL**
- **Acceptance check:** Inductive sensor active ⇒ picked; otherwise skipped.

2. **Task 2 – Sorting non-metallic objects**

- **Goal:** Evacuation manipulator picks only non-metallic objects and places them on the slide.
- **Demo URL**
- **Acceptance check:** Inductive sensor inactive ⇒ picked.

3. **Task 3 – Sorting large blue objects**

- **Goal:** Evacuation manipulator picks only **large blue** objects and places them on the slide.
- **Demo URL**
- **Acceptance check:** Size==Large AND Colour==Blue.

4. **Task 4 – Sorting small blue objects**

- **Goal:** Evacuation manipulator picks only **small blue** objects and places them on the slide. *(fixed wording)*
- **Demo URL**
- **Acceptance check:** Size==Small AND Colour==Blue.

5. **Task 5 – Sorting large metallic objects**

- **Goal:** Evacuation manipulator picks only **large metallic** objects and places them on the slide.
- **Demo URL**
- **Acceptance check:** Size==Large AND Inductive==True.

6. **Task 6 – Sorting small metallic objects**

- **Goal:** Evacuation manipulator picks only **small metallic** objects and places them on the slide.

- **Demo URL**
- **Acceptance check:** Size==Small AND Inductive==True.

7. **Task 7 – Sorting small white objects**

- **Goal:** Evacuation manipulator picks only **small white** objects and places them on the slide.
- **Demo URL**
- **Acceptance check:** Size==Small AND Colour==White.

8. **Task 8 – Sorting large white objects**

- **Goal:** Evacuation manipulator picks only **large white** objects and places them on the slide.
- **Demo URL**
- **Acceptance check:** Size==Large AND Colour==White.

Conclusion

The automation solution is built around a modular PLC program that interacts with a range of pneumatic actuators and sensors to dispense, measure, classify and evacuate workpieces. By clearly mapping inputs and outputs, isolating subsystem logic into reusable modules and implementing a robust safety state machine, the system achieves reliable, deterministic operation. This structured approach facilitates maintenance, debugging and future upgrades while ensuring that safety interlocks and timing constraints are always respected.

Simulation & PLC Development Environment

- **3D System Simulation:** The system is simulated using **Simumatik**, which provides a virtual 3D environment for devices, sensors, and actuators.
- **PLC Programming:** The control logic is developed in **CODESYS Development System V3 (v3.5)**.
- **Communication Interface: OPC UA** is used as the common protocol between Simumatik and the CODESYS runtime to exchange I/O data and status in real time.

Notes:

- Configure the OPC UA server (endpoint, security mode, namespace mapping) on the PLC/runtime side and the OPC UA client within Simumatik to bind signals to PLC variables.
- Ensure consistent data types and address mapping for inputs/outputs (e.g., bytes, bools, analog values) to avoid quality errors.
- Time synchronization and scan rates should be aligned to maintain deterministic behavior in closed-loop tests.