Group 1
Lars Bach Sørensen (S235648) ,
Lasse Manicus (S235655),
Marius Millington (S235659)

INDUSTRIAL PROGRAMMING

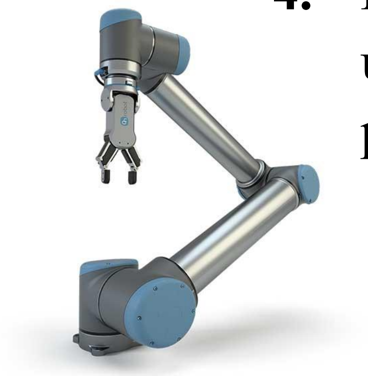
Group 1
Lars Bach Sørensen (S235648) ,
Lasse Manicus (S235655),
Marius Millington (S235659)

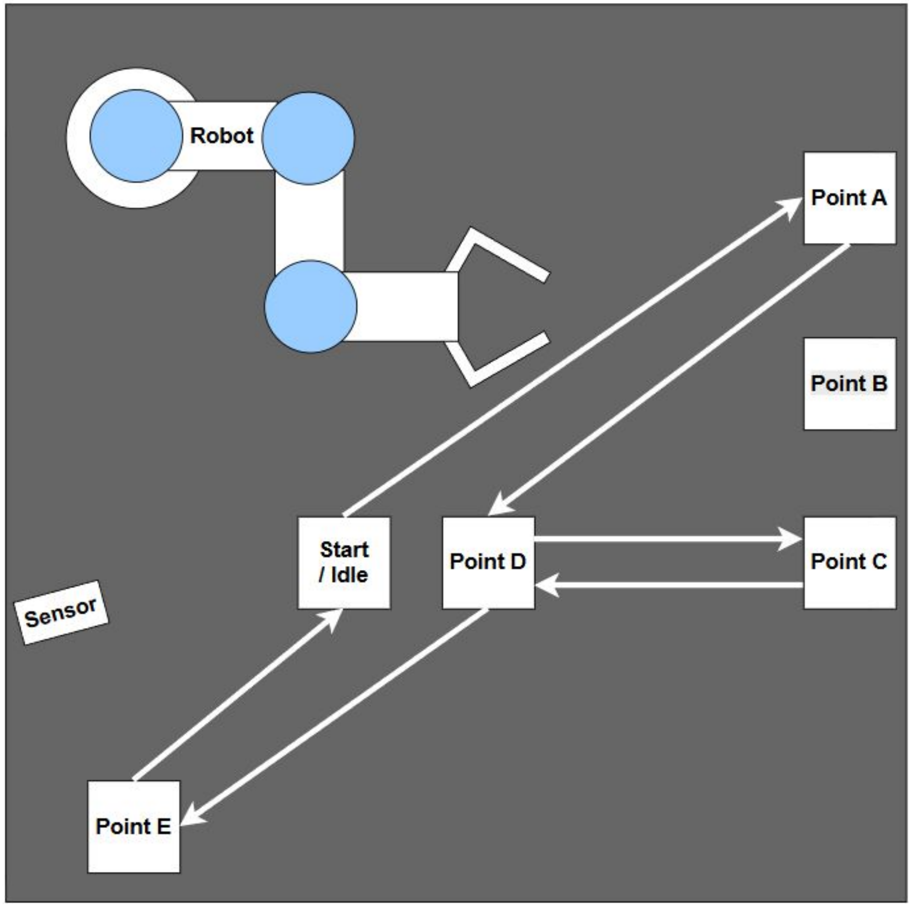
INDUSTRIAL PROGRAMMING

TOY FACTORY

Introduction / Problem (Scope)

1. **Order to robot action:** Converting customer orders (color variant and amounts) into concrete robot tasks and a set of sequences.
2. **Reliable robot communication:** Establishing communication between the PC application and the robot controller using URScript and IP.
3. **Process sequencing and handling:** Ensuring that assembly steps are executed in the correct order, and that the system can safely transition between idle, processing and completion.
4. **Data consistency:** Maintenance between graphical user interface, the database, and the physical production process (Robot)

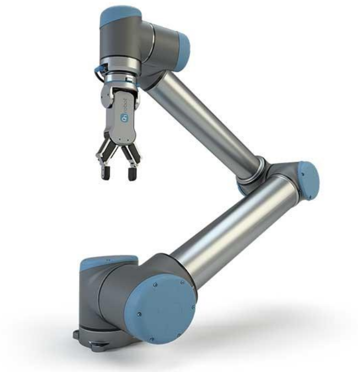




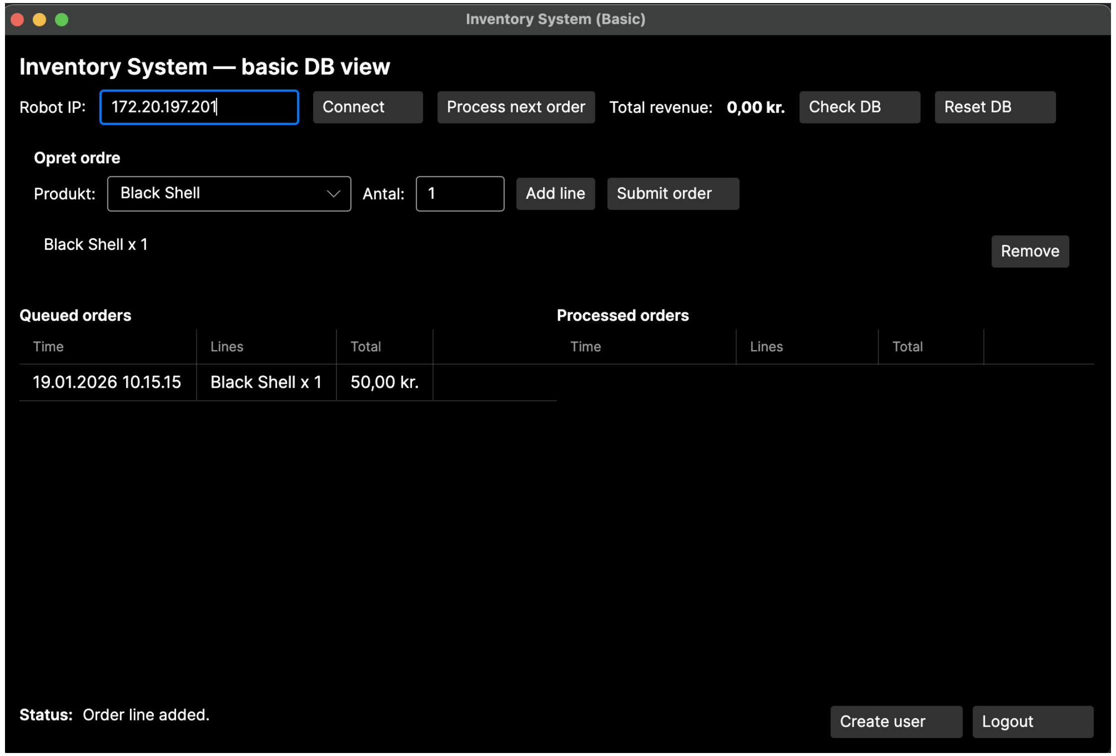
VIDEO

LINK:

https://drive.google.com/file/d/14DXHzHSYnXVVVRLE-rbOtciuuxSQrx-S/view?usp=drive_link



System Architecture



SCOPE - > Design principles

GUI / Operator Station

- Create & monitor production orders
- Start / control production

Application Layer (Control Logic)

- Reads next queued order
- Maps order → predefined robot sequence
- Executes robot + updates database

Data Layer (EF Core + SQLite)

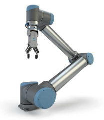
- Orders (Queued / Processed)
- Inventory & quantities
- Persistent system state

Robot Integration Layer

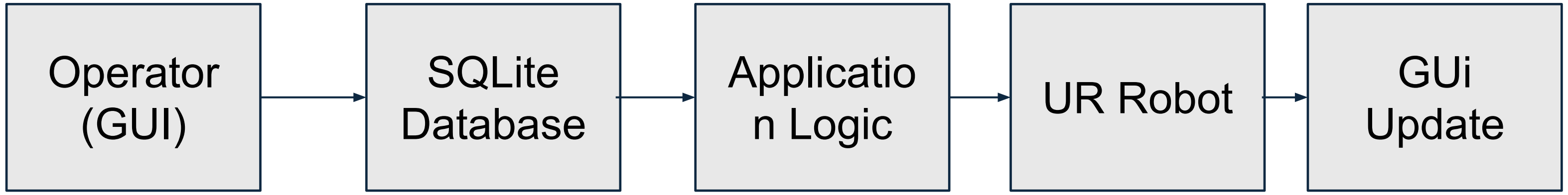
- URScript via TCP/IP
- Centralized motion sequences & positions
- Deterministic execution per item variant

Design Principles

- Single source of truth: Database
- Clear separation of concerns
- Traceability & restart-safe operation
- Robot logic decoupled from UI & data



Production flow:



Domain / Inventory

Domain Model – Core Business Logic: What is produced and in which order, rather than robot assembly. Independent of gui and robot.

Inventory Model

- Represents physical components used in production
- Inventory = collection of items + quantities
- Common item abstraction with optional specialization
- Tracks stock an updates when orders are processed.

OrderBook & Order States

- Central production flow controller

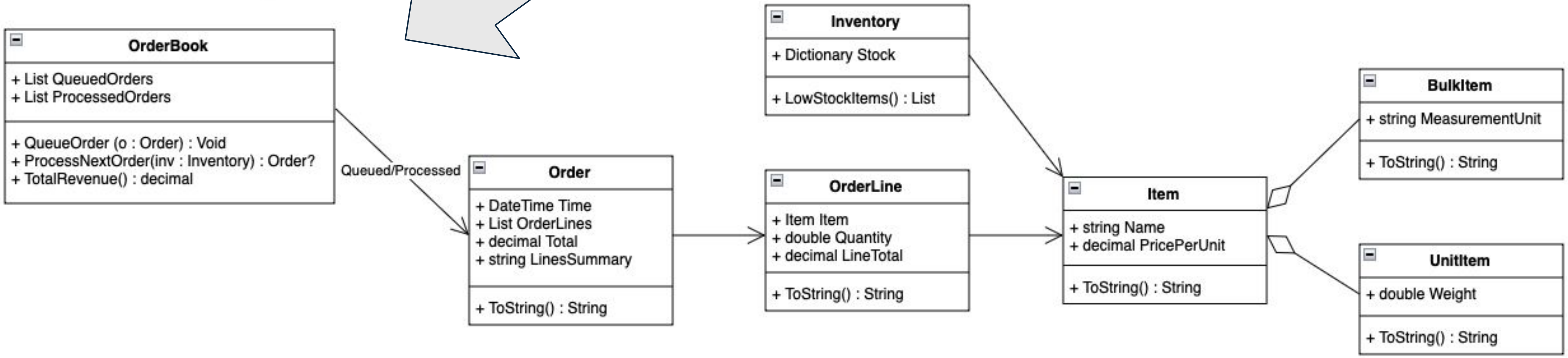
Two states

- Queued Orders (pending)
- Processed Orders (completed)

Characteristics

- **Operator choice:** the GUI lets the operator *compose* an order (White/Black shell lines) and submit it.
- **Queue rule:** once submitted, orders are **queued** and executed **FIFO**.
- **Traceability:** queue/processed state is stored in the database (single source of truth).

Domain Class Diagram



Domain / Inventory

Design Rationale

- Domain logic isolated from UI and robot execution
- Operator control improves transparency, safety, and testing
- Deterministic execution with option for future automation

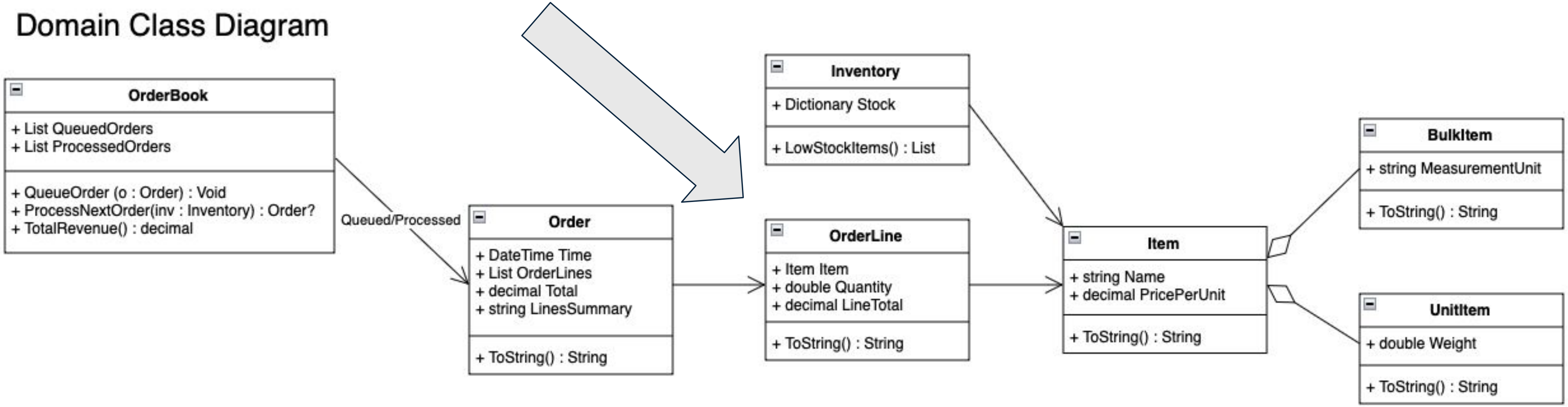
Result:
A simplified but realistic production control model combining
human supervision + data-driven automation

Order Model

- **Order** = timestamp + list of **OrderLines**
- **OrderLine** = item + quantity
- Supports multi-item orders and future product expansion

Technical Example: Process Next Order -> Boolean (order queued condition)-> if else (stop or continue) -> Loop (based on quantity). Strings for items (reference text)

Domain Class Diagram



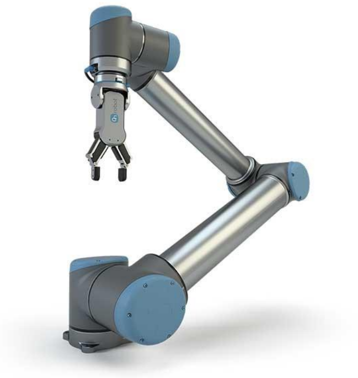
Database Design (MARIUS)

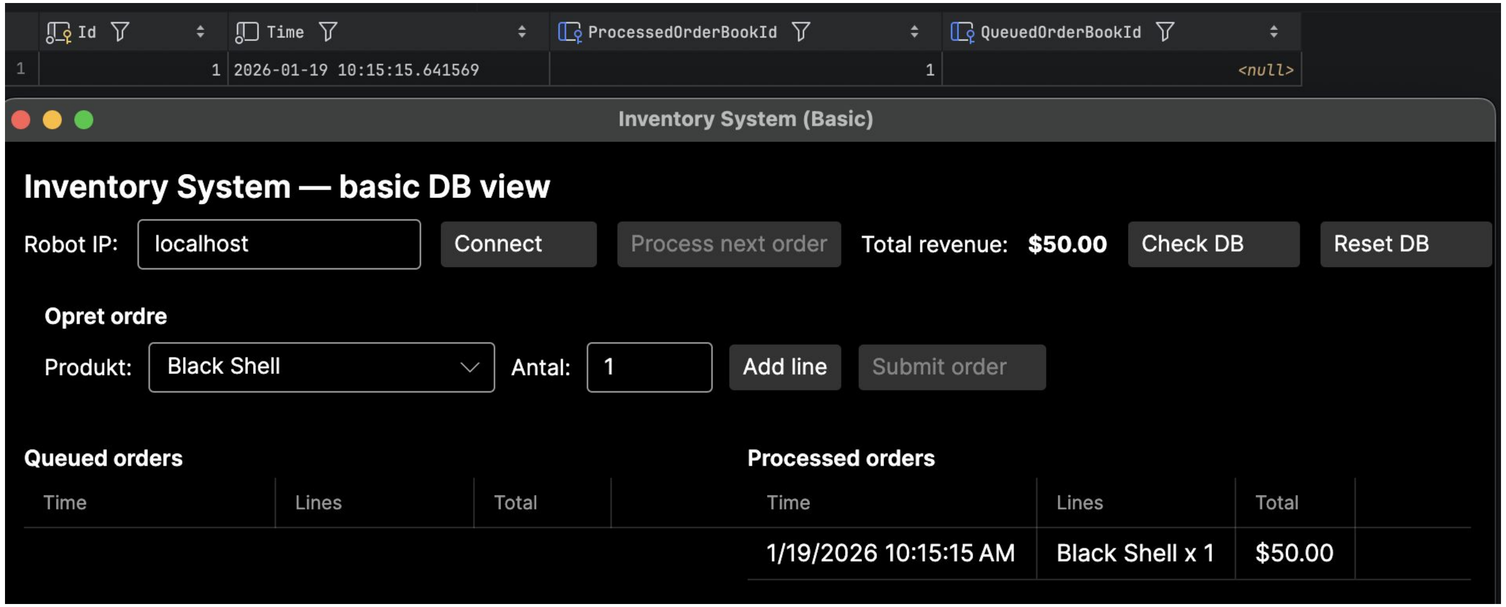
Database Technology and Access

- Ensures consistency between GUI, Application logic and physical robot execution.
- SQLite
- Entity Framework Core to persist production state
- Seeding & Reset

Data Flow & Consistency

1. All production-related actions follow a database-driven workflow.
2. Orders are created through the GUI and stored in the database.
3. The application retrieves the next queued order.
4. Order data is translated into a robot motion sequence.
5. After successful execution, the database is updated: the order is moved to processed and inventory quantities are reduced.”





Security

Login feature

- Database controlled

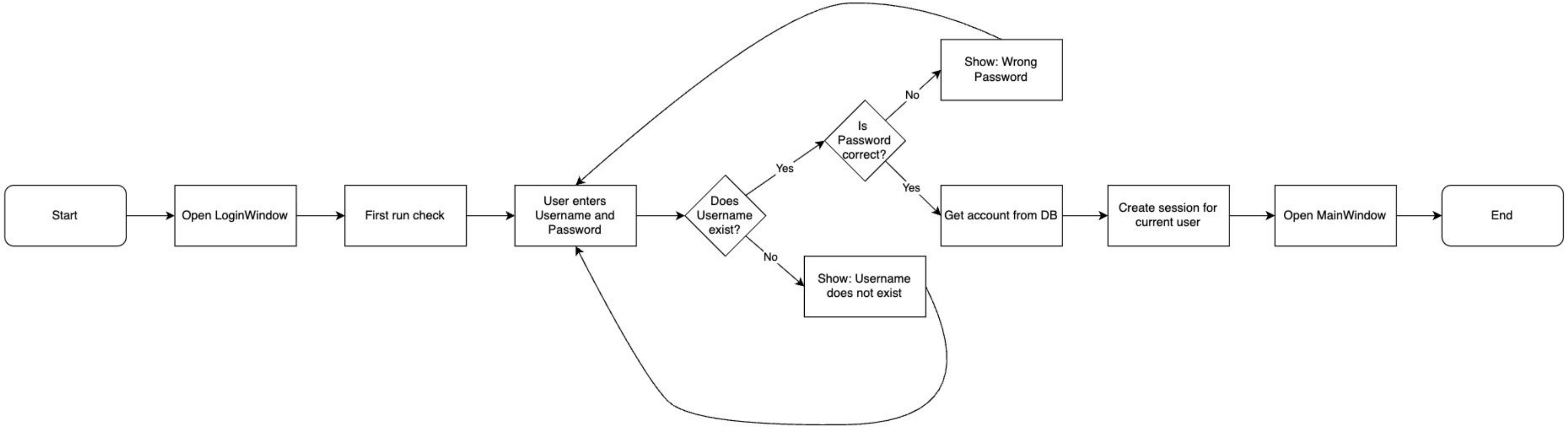
- Has multiple verification levels
  - Does Username exist
  - Is password correct

- Creates a session for the logged in user

Password Salting

- The system uses salting in database

- Protects against precomputed attacks like Rainbow table



Security

User

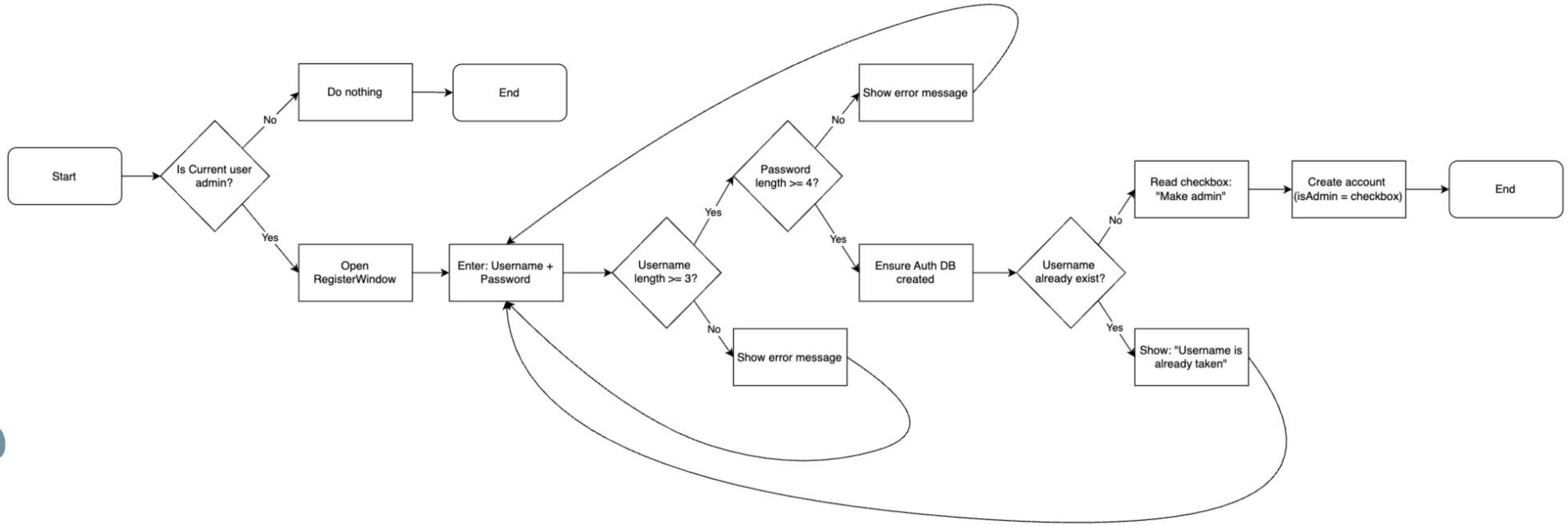
Logout

Admin User

Create user | Logout

Create User (Admin)

- Create new user is limited to admins
- Account requirements
  - Username length => 3?
  - Password length => 4?
- Option to make admin
- Creates account in database

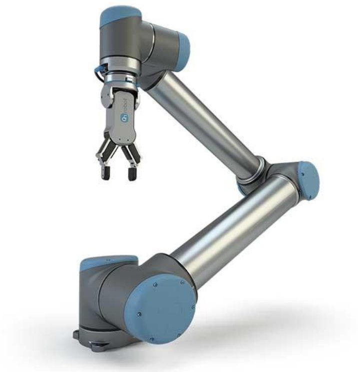


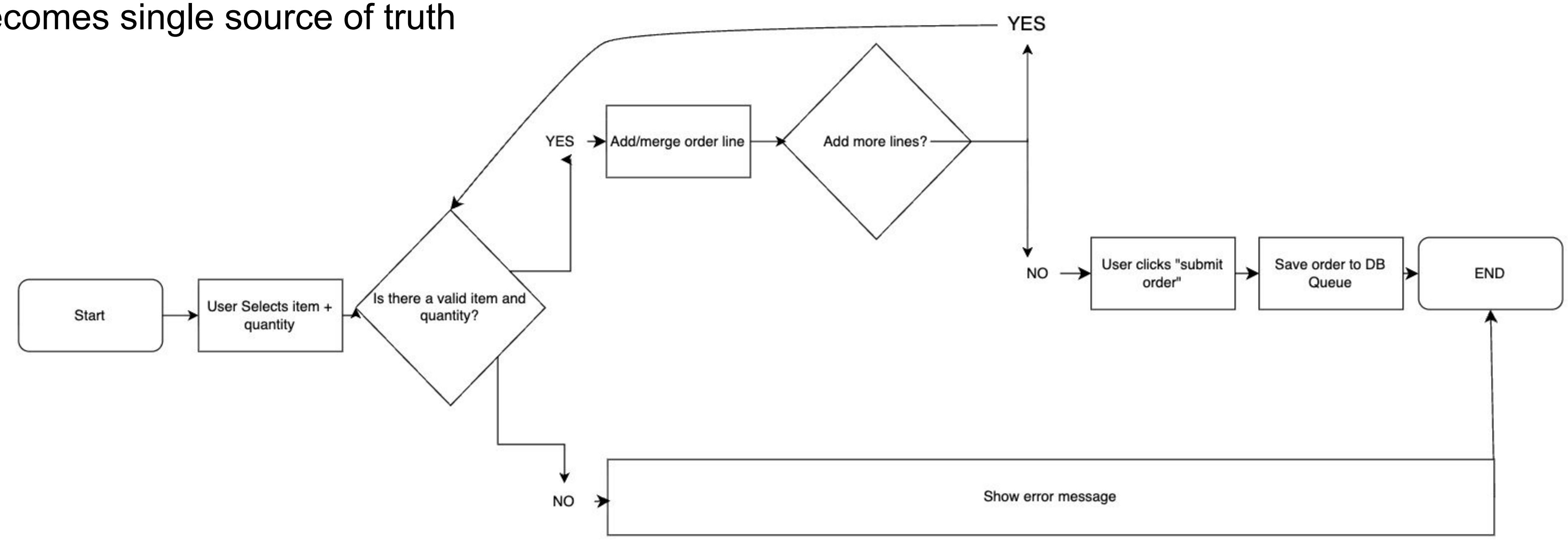
GUI and ViewModel (MVVM)

Interactive GUI able to both create orders and process orders.

Create order

- User able to select both item and quantity
- The order is first validated by the system
- Database stores as a queued order
  - Database becomes single source of truth





GUI and ViewModel (MVVM)

Interactive GUI able to both create orders and process orders.

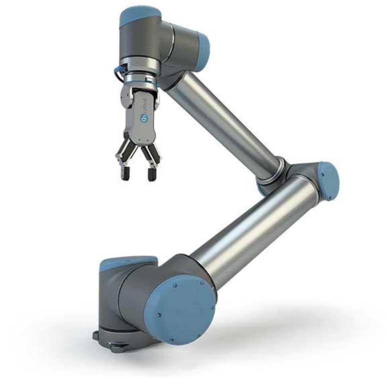
Process order

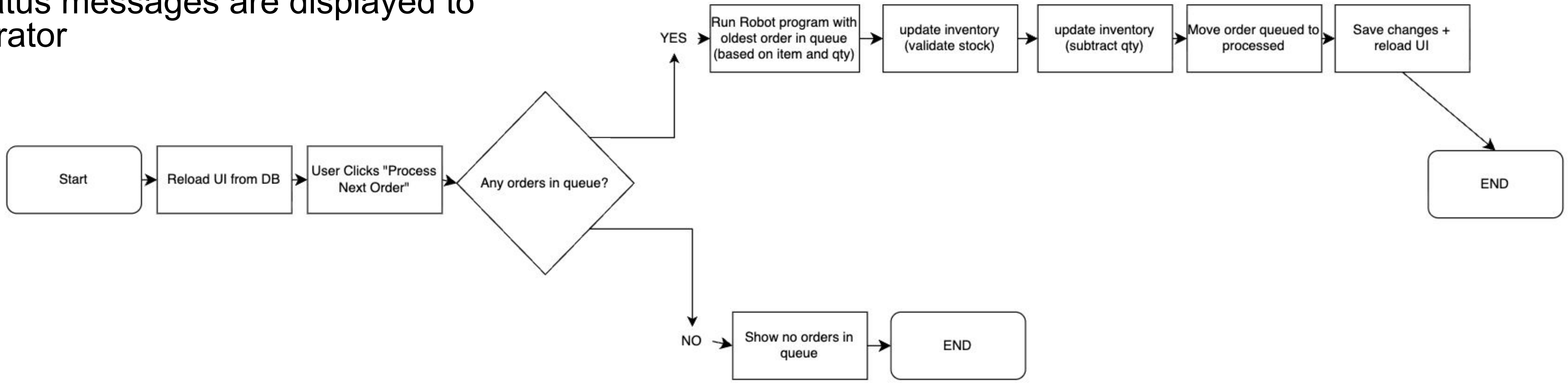
- *Process Next Order* retrieves oldest queued order in database

- The order is translated to the robot

- When robot is done - Database is updated
  - Inventory updated
  - Queued → Processed

- Continuous status messages are displayed to inform the operator

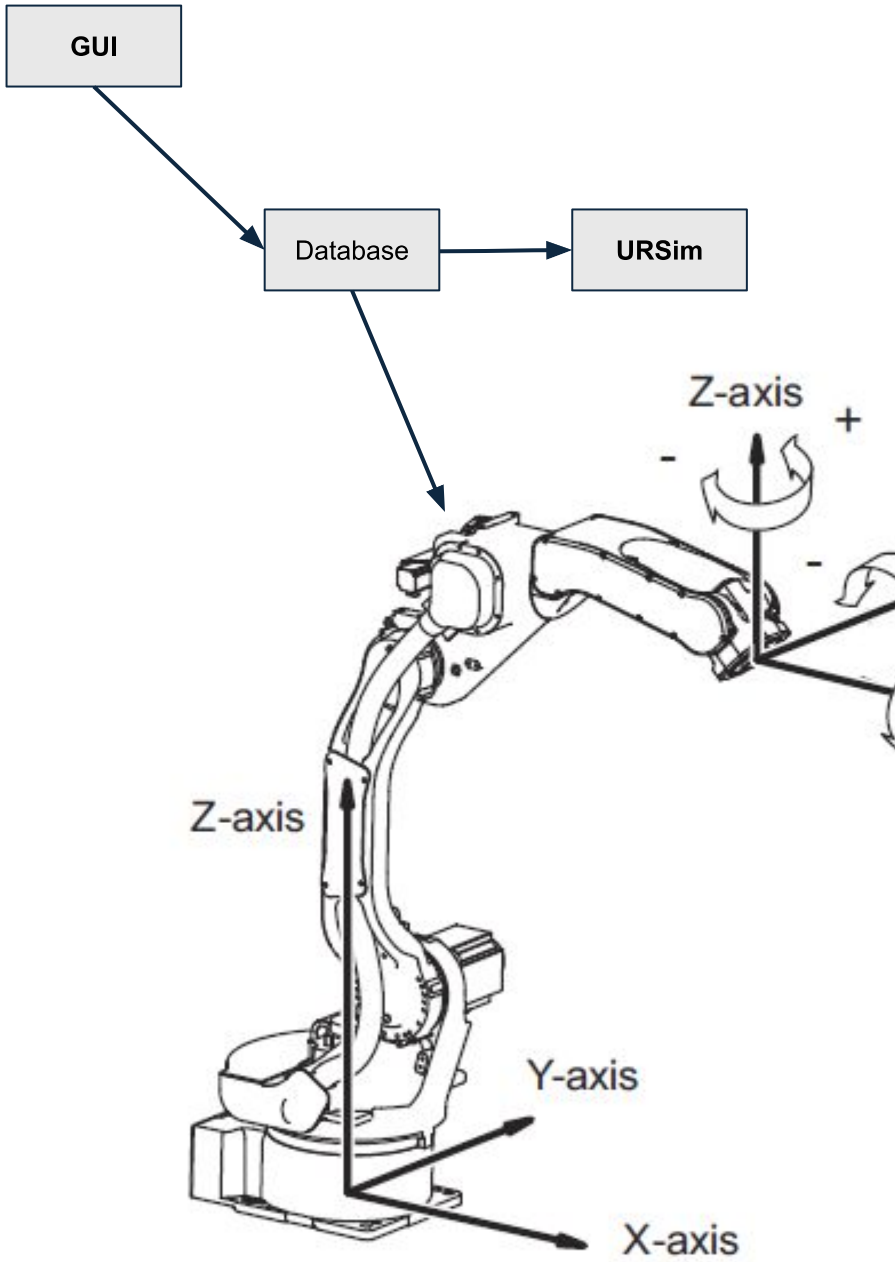


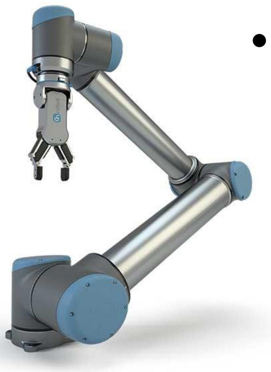


Robot Integration

- Database driven
- Orders created in GUI
- Stored in Database
- Communicating via URScript - TCP/IP
- Connects to URSim & real robot

- Start position (Tool Center Point)
  - X = 0.125
  - Y = -0.300
  - Z = 0.100
- Start rotation (Tool Center Point)
  - RX = 3.14 (180° nedad)
  - RY = 0.00
  - RZ = 0.00

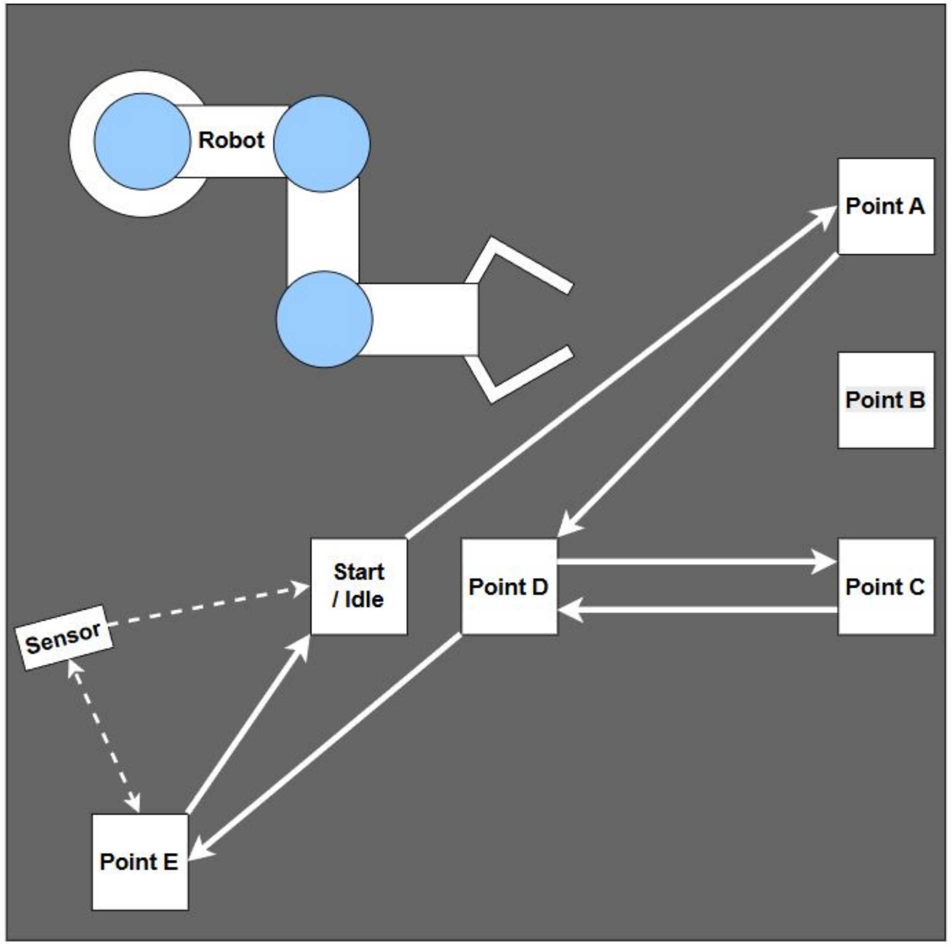


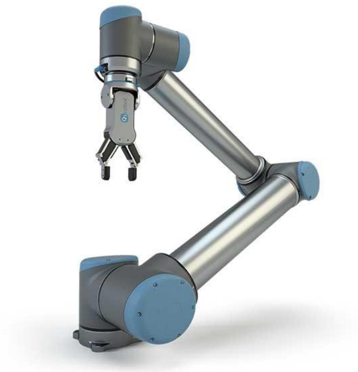


Robot Integration

- Robot motion structure
  - Determined motion from A→E
    - Using Move L
  - Pick & place - Height
    - Gripper - RG2
      - open_mm = 60
      - open_pick_mm = 85
      - close_mm = 31
      - f_open = 30
      - f_close = 30

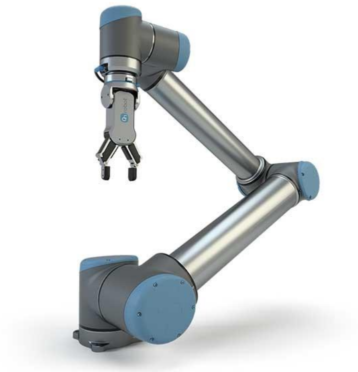
- Predefined predictable sequences

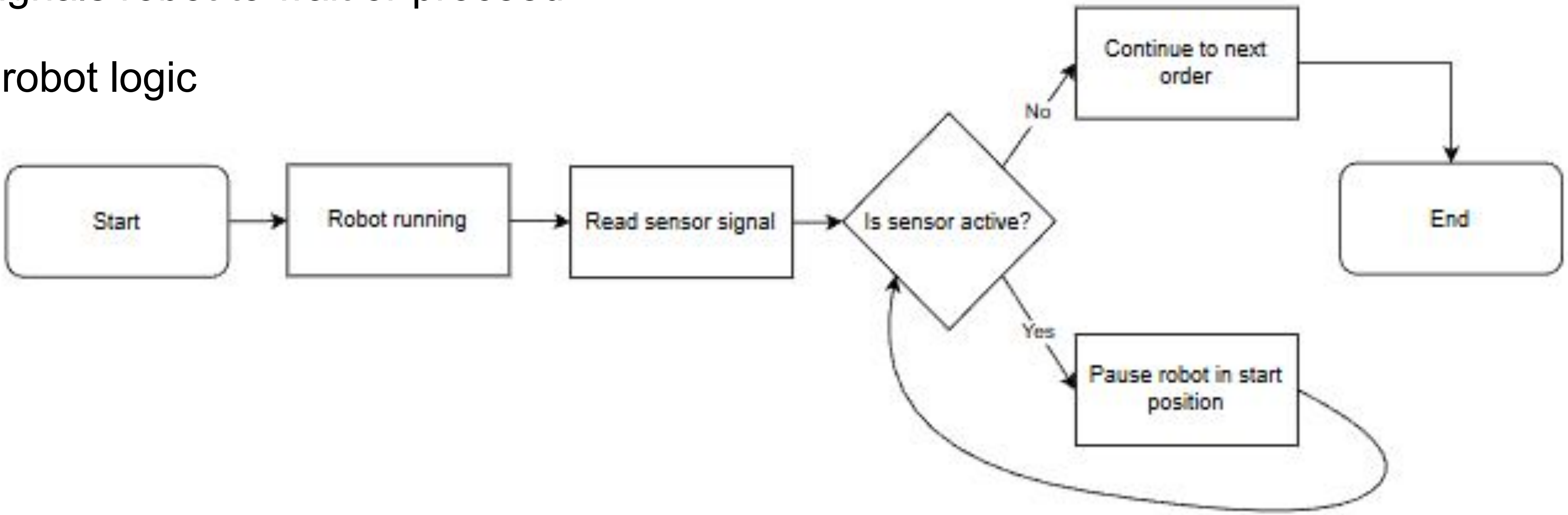




Sensor (Safety)

- Why safety is needed?
  - Due to human interaction
  - Error minimizing in production loop
- Sensor role
  - Photoelectric
  - Detects products
  - Signals robot to wait or proceed
- Uses robot logic





Discussion

What worked well:

- Easy operator-controlling
- Stable robot communication
- Predictable robot behavior due to fixed positions
- Clear separation between GUI, database, and robot logic
- Easy testing using URSim

Challenges:

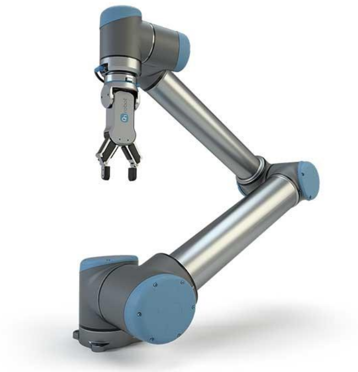
- Robot calibration required significant testing
- Fixed positions reduce flexibility
- System relies on operator-controlling

Trade-offs:

- Manual operator control chosen over full automation
- Fixed positions instead of vision system
- Safety prioritized over throughput

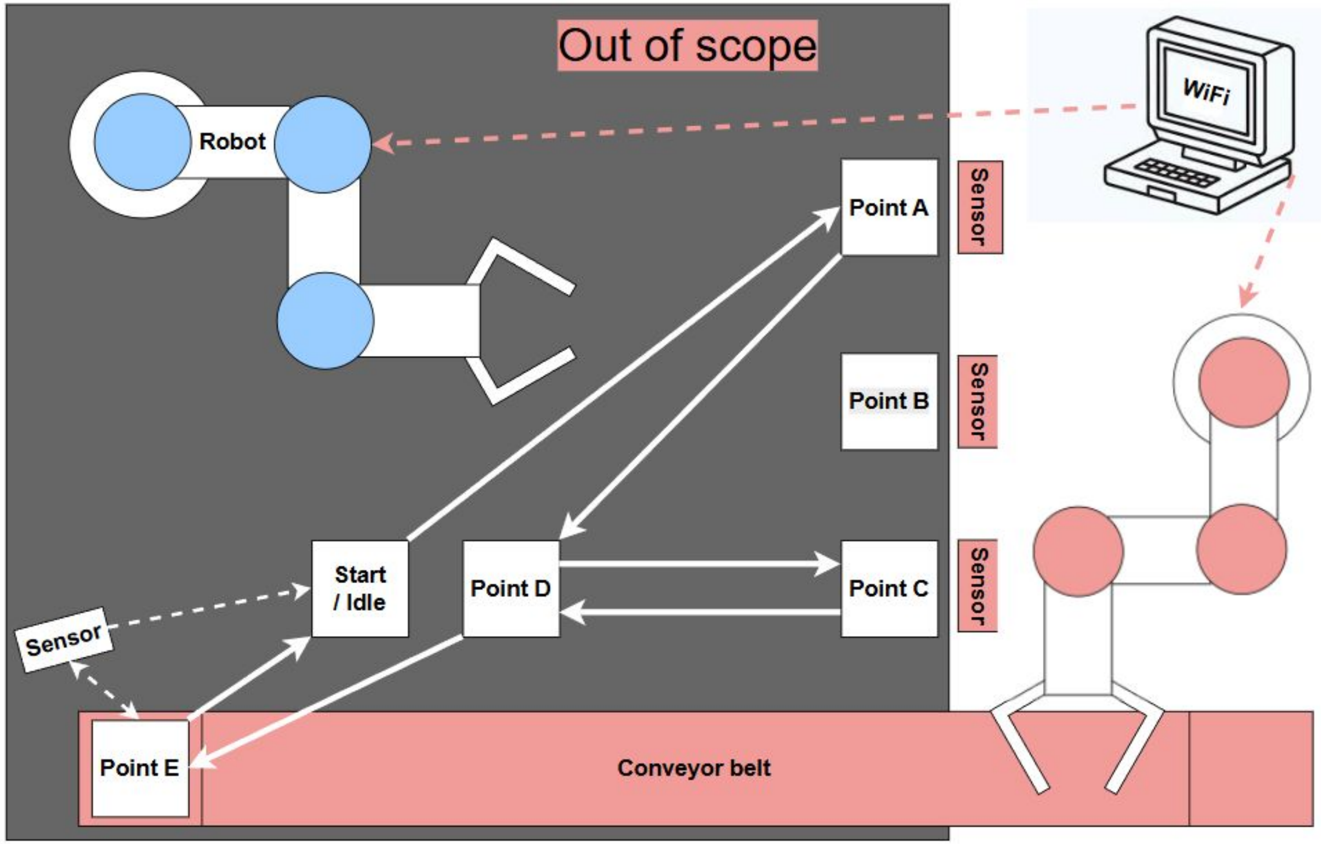
Solution

- Functional automated assembly prototype
- End-to-end digital flow:
  - GUI → Database → Robot → Database
- Minimal human interaction during execution
- Modular and maintainable architecture



Future implementation

- Wireless connection
- Additional robots
- Conveyor belt
- Machine vision



Conclusion

- Project objectives were met
- System behaves like a realistic industrial prototype
- Demonstrates principles from:
  - Industrial programming
  - Industry 4.0 & 5.0
  - Automation
  - Security & safety
  - Modular software design