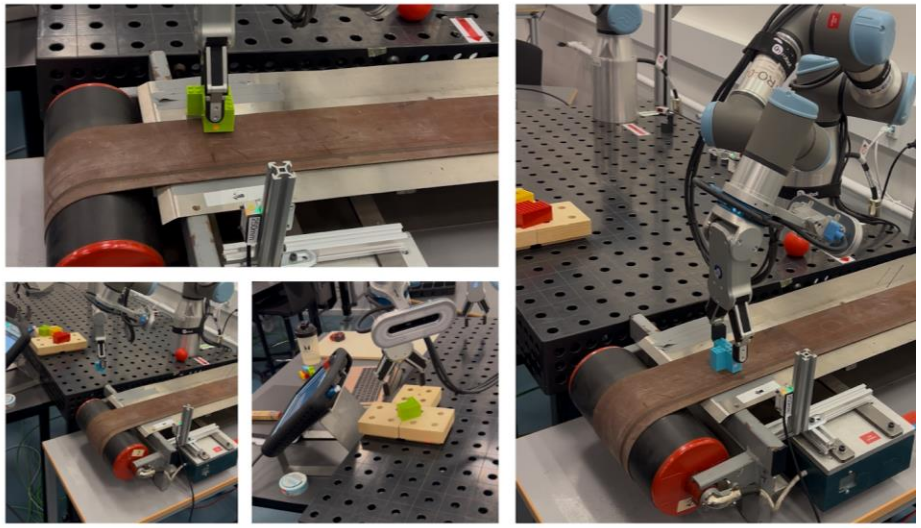
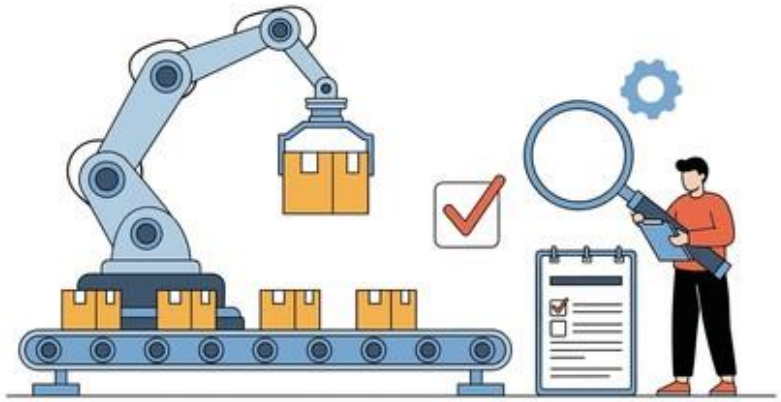
Group 3
Industrial Programming

Christian Olesen s235657
Francisco Noppenau s235669
Simon B. Rasmussen s235654

Project case:



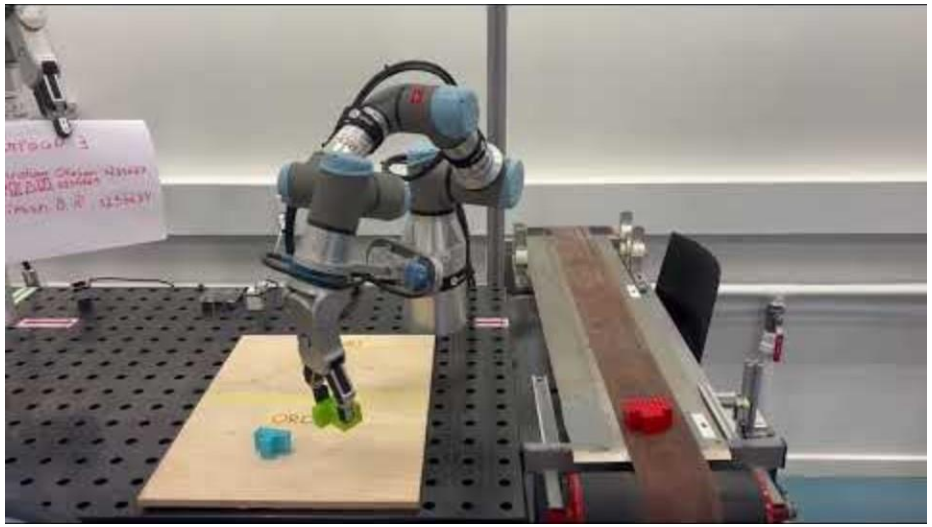
This project solves cases where production lines require automatic, reliable sorting of products for different customers. It replaces manual handling with a camera-guided robotic system that improves efficiency, accuracy, and scalability in industrial workflows.



Video Link

https://www.youtube.com/watch?v=uZIzbW6h2qI’

Group 3 - Industrial Programming Robot sorter



How did we solve it ?

Day 1-3 “Rookie Mistakes”

Day 1:

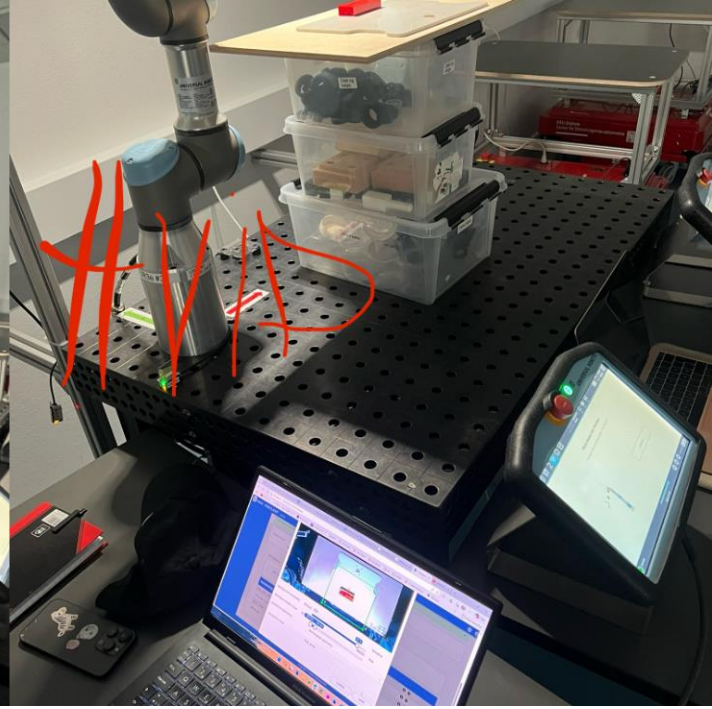
- Figuring things out
- Initial Camera setup; stack 3 boxes

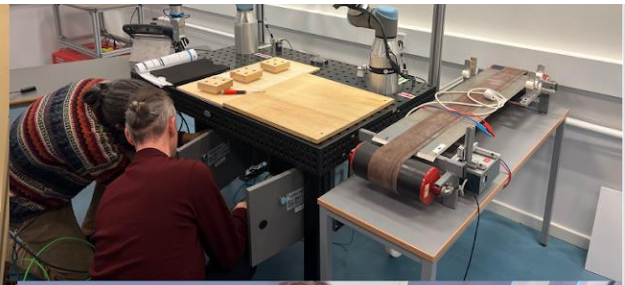
Day 2:

- Conveyor up and running – programmed with pendant
- Camera on arm
- Time not wasted; Lessons learned about reliable detection
- Pendent programming

Day 3:

- Finished programming the movement
- Inverse kinematics error – alternative pick paths
- Consistency is key





Day 4-6 ”Progress”

Day 4:

- Trying to make sense of exported file from pendant
- Fewer waypoints

Day 5:

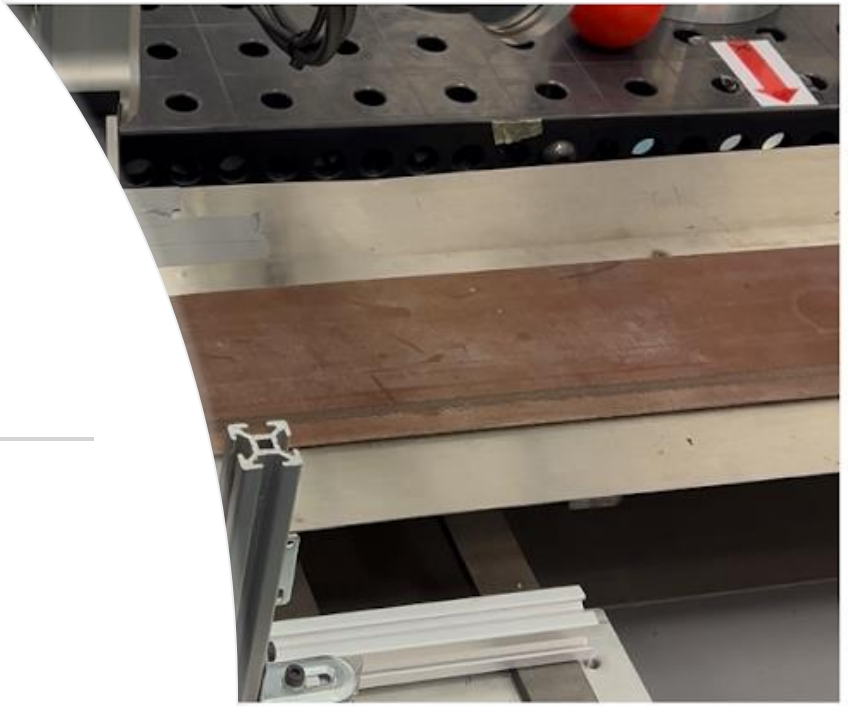
- Choosing to be pragmatic
  - A lot of redundent code left

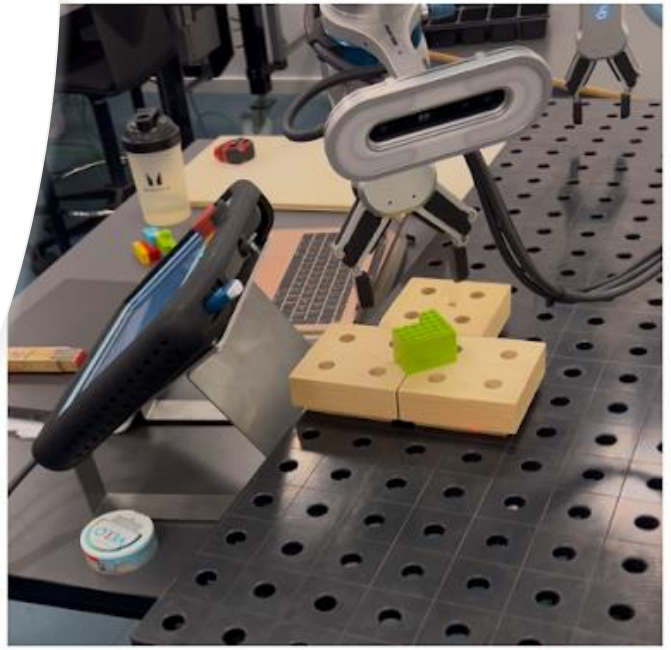
Day 5:

- Extension of Database
  - Customer profiles
  - Orders
- GUI Customization

Day 6:

- Eyes Locate: 15 seconds
- Multiple quantities of the same brick
- Automatic Log out





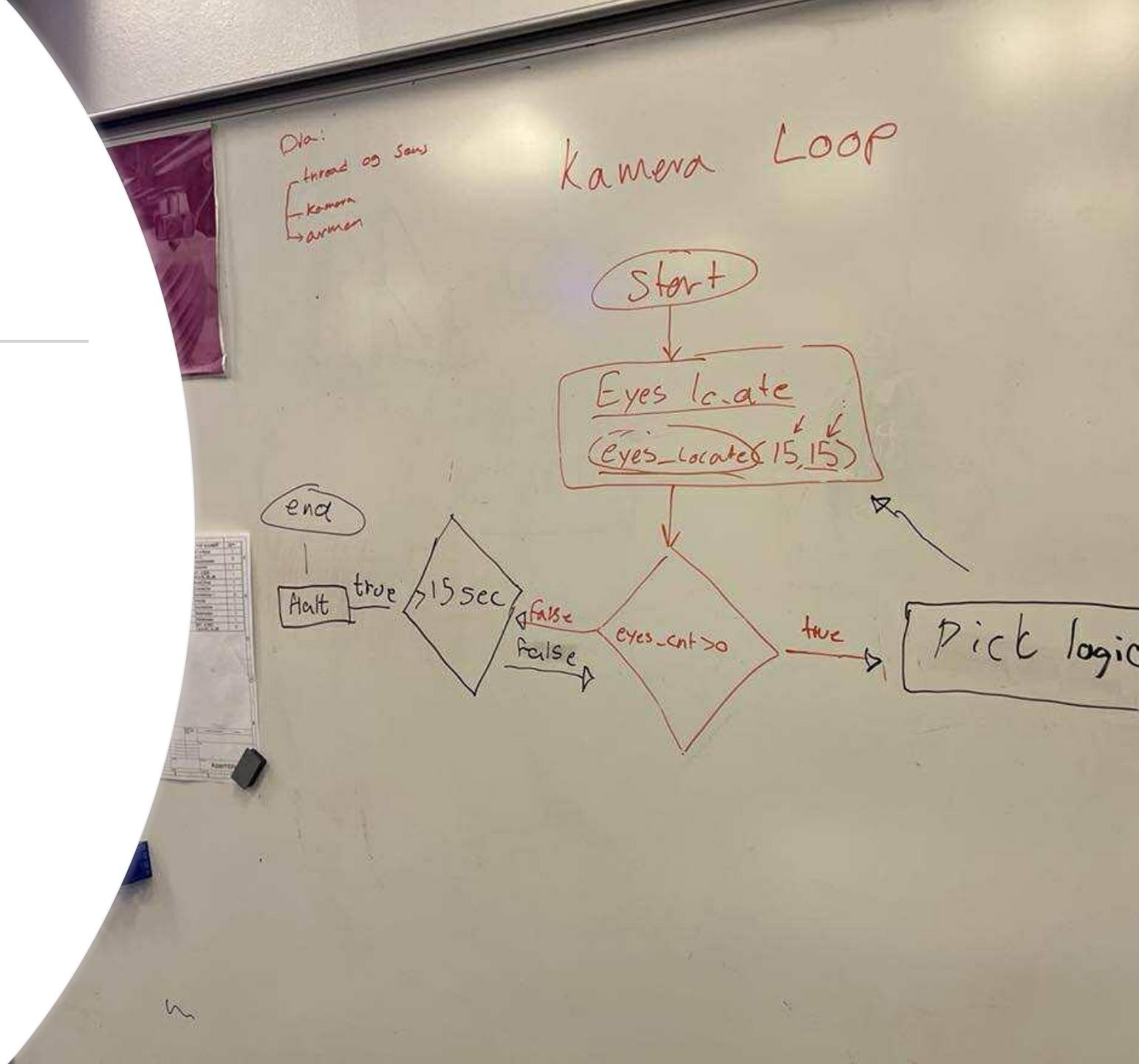
Day 7-8 ”KISS”

Day 7:

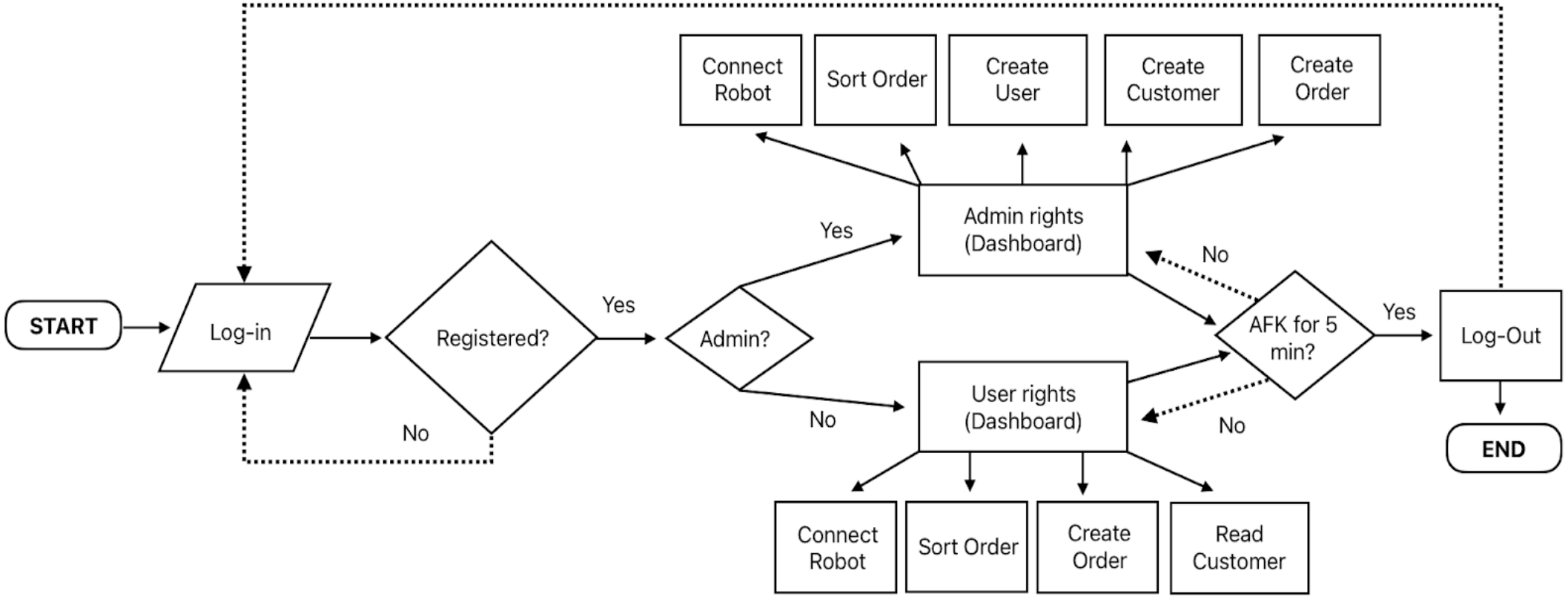
- Details (KISS); halt
  - No need for restart button
  - UI cleanup for better UX
  - Removed old .cs test files

Day 8:

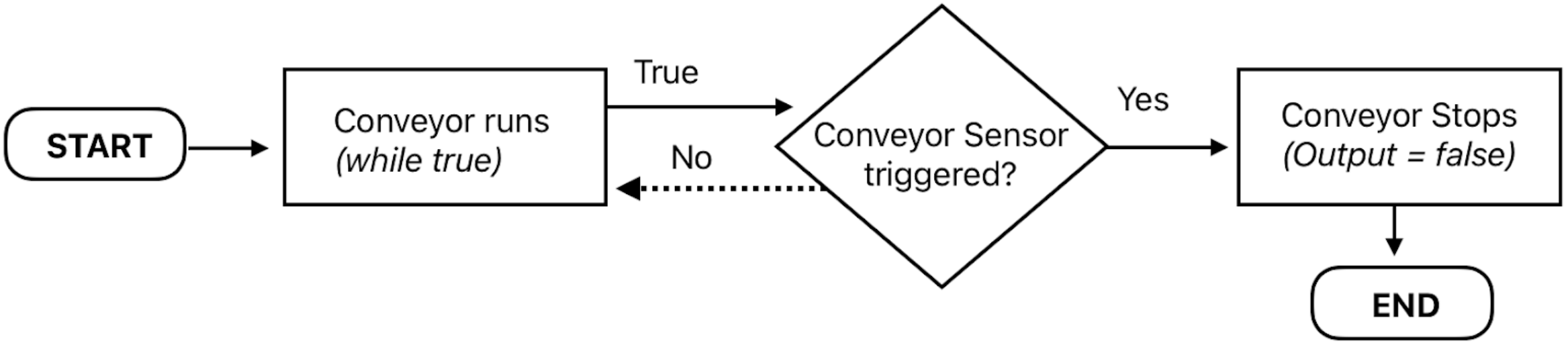
- Flow charts and video – Final product



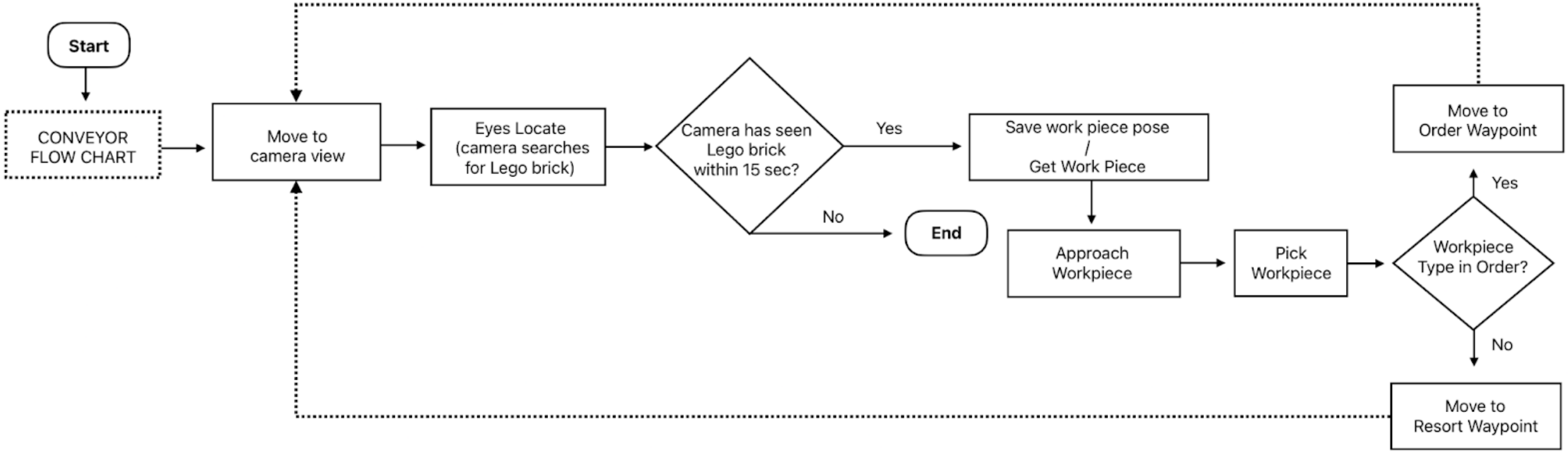
GUI Interaction Flow Chart



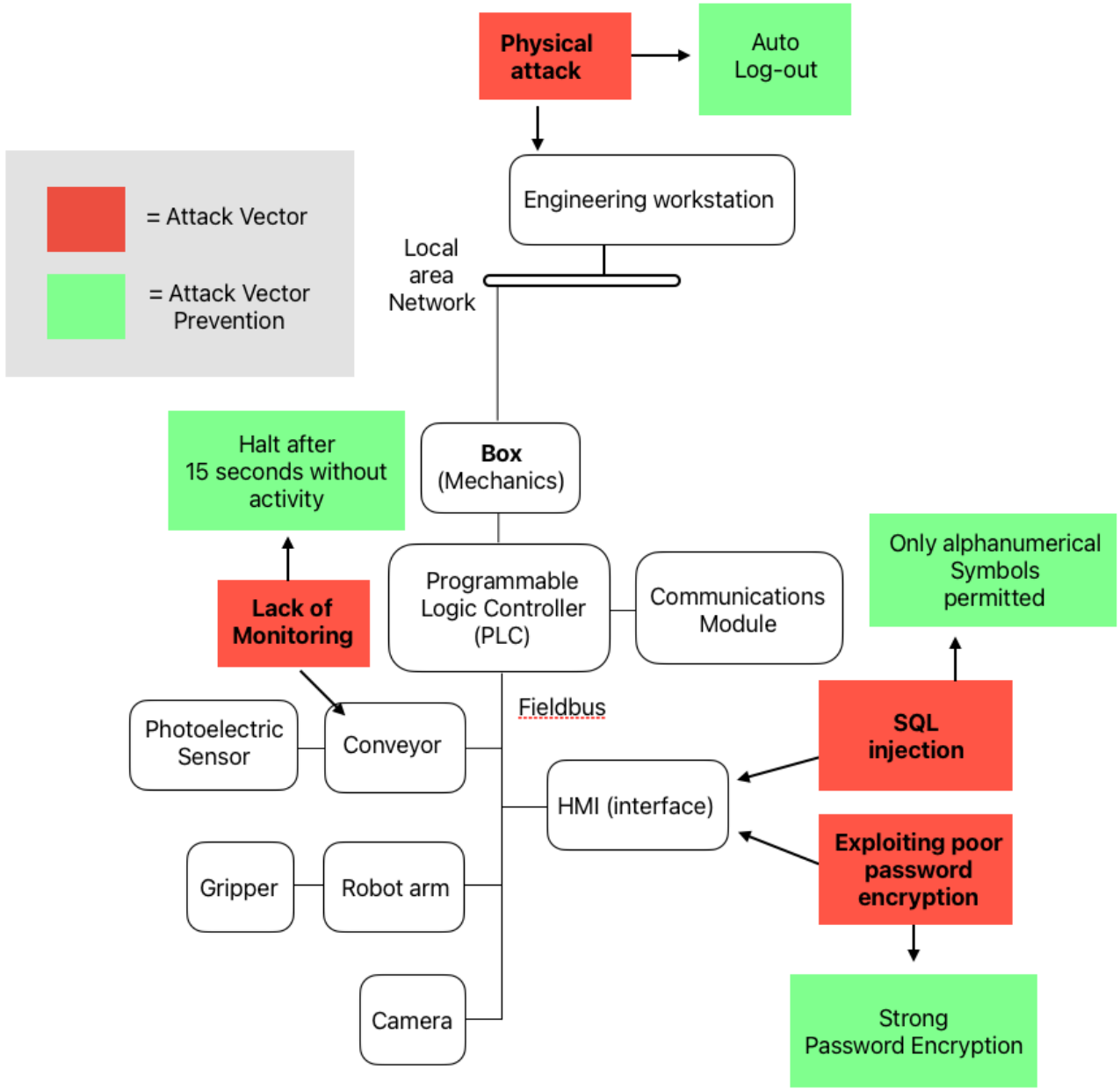
Conveyor Flow Chart



Sorting Flow Chart

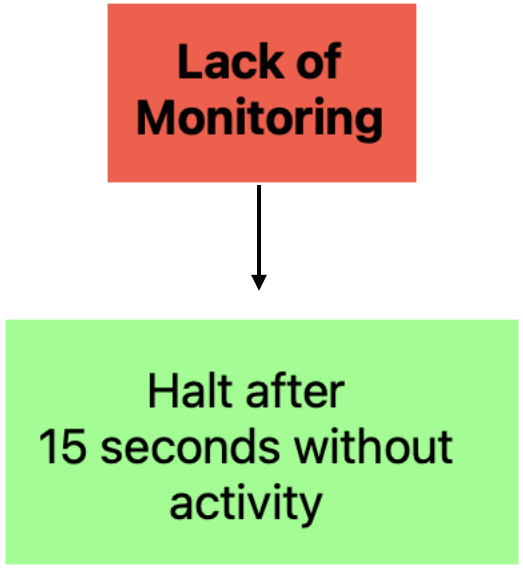


Program Diagram



Attack Vector and Prevention: Conveyor safety stop

```
1730        $ 6 "Eyes Locate"
1731        eyes_cnt = 0
1732    eyes_cnt = eyes_locate(15, 15)
1733    if (eyes_cnt > 0):
1734      eyes_workp_cnt = eyes_cnt
1735      # existing pick logic continues
1736    else:
1737      set_standard_digital_out(7, False)  # stop conveyor
1738      halt
1739      end
```



Attack Vector and Prevention: Auto Log-Out

```
_inactivityLogoutTimer = new InactivityLogoutTimer(TimeSpan.FromMinutes(5), HandleInactivityTimeout);
```

```
4 references
private void ResetInactivityTimer()
{
    if (_currentUser == null)
        return;

    _inactivityLogoutTimer.Reset();
}
```

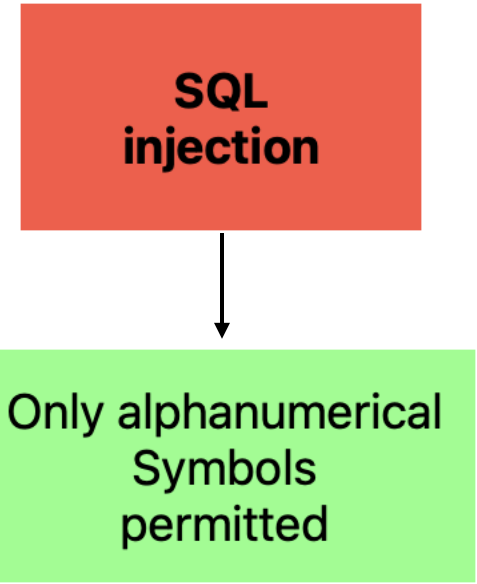
```
1 reference
private void HandleInactivityTimeout()
{
    if (_currentUser == null)
        return;

    _log("No activity detected. Logging out.");
    LogoutButton_OnClick(this, new RoutedEventArgs());
}
```

```
1 reference
private void StopInactivityTimer()
{
    _inactivityLogoutTimer.Stop();
}
```

Attack Vector and Prevention: SQL Injection

```
4 references
public static bool IsCredentialValid(string? value) =>
    !string.IsNullOrWhiteSpace(value) && value.All(char.IsLetterOrDigit);
```



Attack Vector and Prevention: Password Policies

```
public static bool IsPasswordValid(string? value) =>
    IsCredentialValid(value) && value!.Length > 8;
```

Length	Possible combinations
4 chars	14,776,336
9 chars	13,537,086,546,263,552

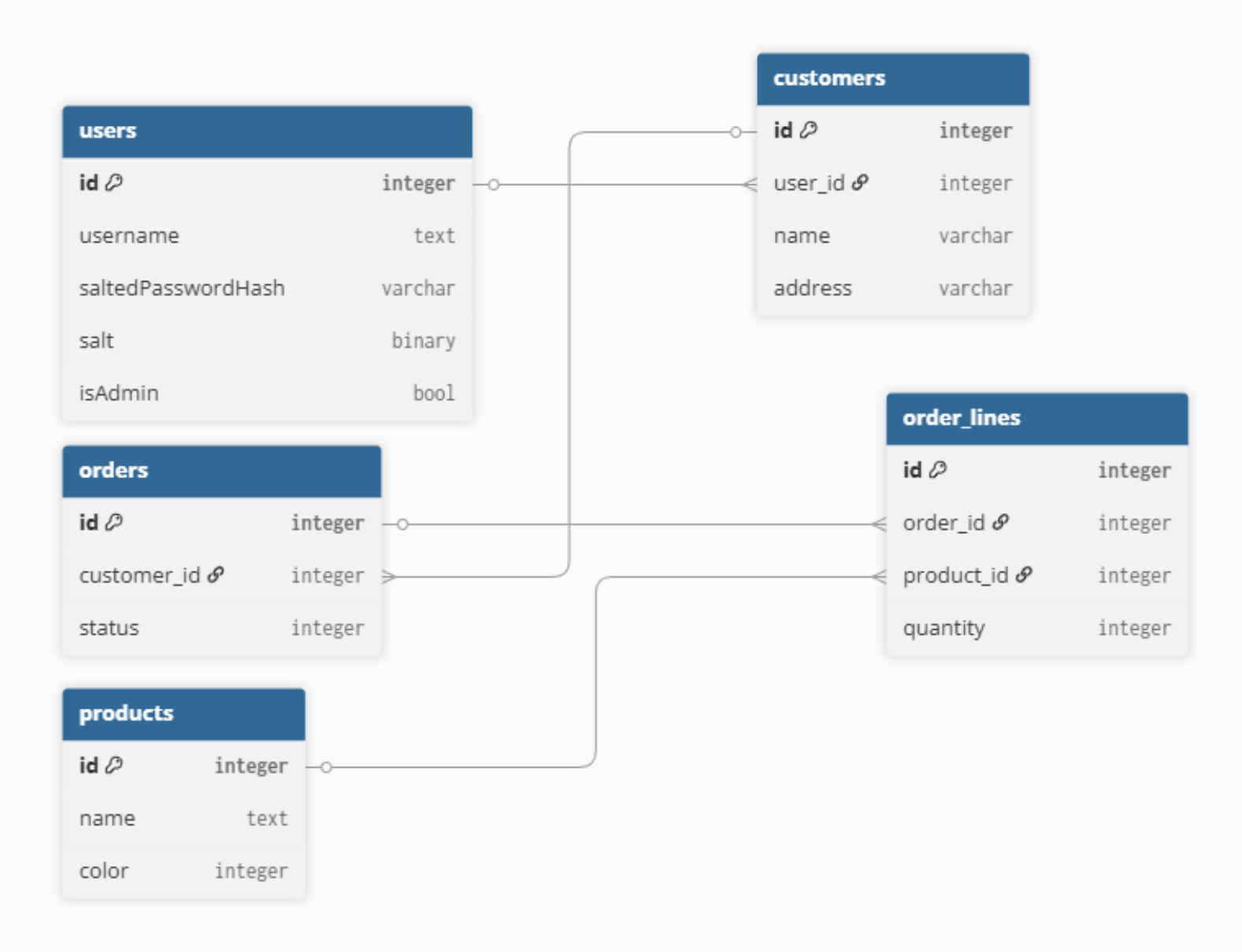
A 9-character code is ~916 MILLION times stronger than a 4-character one

SQLite Database Tables

One-to-Many:
User => Customers
Customers => Orders
Orders => Order_lines
Product => Order_lines

Many-to-Many:
Orders => products by order_lines

Why?
PLP
Better UI/UX
Multiple Qty and products



```
Console.WriteLine("End of presentation");
```

Appendix: Main Program flow

Setup: global state + waypoints + order counters

```
global Waypoint_1_p = p[...]
global OrderDropPose = p[...]
global ResortDropPose = p[...]
global BlueRemaining=0; global
RedRemaining=0; global GreenRemaining=0;
global YellowRemaining=0
```

Thread running in Parralel

```
thread Thread_1():
 while (True):
  set_standard_digital_out(7, True)
while (get_standard_digital_in(4) == False):
sync()
  end
  set_standard_digital_out(7, False)
while (get_standard_digital_in(4) == True):
   sync()
  end
 end
end
run Thread_1()
```

Main loop starts from safe/home waypoint

```
while (True):
 movej(get_inverse_kin(Waypoint_1_p,
qnear=Waypoint_1_q), a=..., v=...)
```

Vision scan: locate pieces (timeout) + fail-safe stop

```
eyes_cnt = eyes_locate(15, 15)
if (eyes_cnt > 0):
 eyes_workp_cnt = eyes_cnt
else: set_standard_digital_out(7, False)  # stop conveyor
 halt
 end
```

Appendix: Main Program flow

Per-color pipeline (Blue/Red/Green/Yellow): get pose + type

```
eyes_workp_pose = eyes_getworkpiecepose(1)
eyes_workp_type = eyes_getworkpiecetype()
eyes_workp_cnt  = eyes_getworkpiececount()
if (eyes_workp_type != -1):
  global EyesWorkpPose = eyes_workp_pose
```

Camera-guided pick sequence (approach → linear pick → retract)

```
if (pose_dist(EyesWorkpPose, p[0,0,0,0,0,0]) > 0.005):
  EyesPickPose = EyesWorkpPose
  EyesPickAppr = pose_trans(EyesWorkpPose, p[0,0,-0.0,0,0,0])

  movej(EyesPickAppr, a=..., v=...)
  movel(EyesPickPose, a=1.2, v=0.25)
```

Grip control (close to pick, open to drop)

```
rg_grip(0.0, 12.0, tool_index=0, blocking=True)     # CLOSE (pick)
...
rg_grip(100.0, 40.0, tool_index=0, blocking=True)    # OPEN
(release)
```

Order logic: decide bin + decrement remaining counters

```
drop_pose = ResortDropPose
if (BlueRemaining > 0):
  drop_pose = OrderDropPose
  BlueRemaining = BlueRemaining - 1
end
movel(drop_pose, a=1.2, v=0.25)
```

UI structure: Tab-based app + log panel (XAML)

```
<TabControl Name="TabControl">
  <TabItem Header="Login" Name="LoginTab">...</TabItem>
  <TabItem Header="Robot" Name="RobotTab">...</TabItem>
  <TabItem Header="Orders" Name="OrdersTab">...</TabItem>
  <TabItem Header="Users" Name="UsersTab">...</TabItem>
  <TabItem Header="Customers"
Name="CustomersTab">...</TabItem>
</TabControl>

<TextBlock Name="LogOutput" FontFamily="Consolas" />
<Button Grid.Row="2"
Click="ClearLogButton_OnClick">Clear</Button>
```

Startup wiring: FindControl + hook button events (C# constructor)

```
_robotIpTextBox = this.FindControl<TextBox>("RobotIpTextBox");
_connectRobotButton =
this.FindControl<Button>("ConnectRobotButton");
if (_connectRobotButton != null) _connectRobotButton.Click +=
async (_, __) => await ConnectRobotAsync();
if (_createOrderButton != null) _createOrderButton.Click += async
(_, __) => await CreateOrderAsync();
```

Login gates the entire UI (tabs hidden until auth)

```
foreach (TabItem item in TabControl.Items) item.IsVisible =
false;
LoginTab.IsVisible = true;

// after successful login:
LoginTab.IsVisible = false;
RobotTab.IsVisible = true;
OrdersTab.IsVisible = true;
CustomersTab.IsVisible = true;

if (user.IsAdmin) UsersTab.IsVisible = true;
```

Database + data loading on window load

```
Loaded += OnLoaded;

private async void OnLoaded(...)
{
  if (await EnsureDatabaseCreatedWithExampleDataAsync())
    _log("Customer data did not exist. So I created default
customers.");

  await _appDbContext.EnsureSchemaAsync();
  await LoadCustomersAsync();
  await LoadOrdersAsync();
}
```

Appendix GUI code

Orders UI: create order + select + sort button enabled only when valid

```
<dg:DataGrid x:Name="OrdersGrid" IsReadOnly="True"
SelectionMode="Single">...</dg:DataGrid>
<Button x:Name="SortOrderButton" IsEnabled="False"
Content="Sort Selected Order"/>
_ordersGrid.SelectionChanged += (_, __) => UpdateOrderSelection();

private void UpdateOrderSelection()
{
 _selectedOrder = _ordersGrid?.SelectedItem as OrderViewModel;
 var enable = _selectedOrder?.Status == OrderStatus.Pending;
 _sortOrderButton.IsEnabled = enable == true;
}
```

Robot connection page: connect/disconnect + status label

```
<TextBox x:Name="RobotIpTextBox" Text="172.20.254.203"/>
<Button x:Name="ConnectRobotButton" Content="Connect"/>
<Button x:Name="DisconnectRobotButton" Content="Disconnect"
IsEnabled="False"/>
<TextBlock x:Name="RobotConnectionStatusText" Text="Connection:
Disconnected"/>
await Task.Run(() => _robotConnection.Connect());
UpdateRobotConnectionUi(true);
private void UpdateRobotConnectionUi(bool connected)
{
 _robotConnectionStatusText.Text = connected ? "Connection: Connected" :
"Connection: Disconnected";
 _connectRobotButton.IsEnabled = !connected;
 _disconnectRobotButton.IsEnabled = connected;
}
```

"Sort order" flow: build URScript from DB order → send to robot

```
await _orderOperation.UpdateStatusAsync(orderId,
OrderStatus.Processing);
var orderEntity = await _orderOperation.GetOrderDetailsAsync(orderId);
var sortingOrder = BuildSortingOrder(orderEntity);
var options = SorterColorOptions.FromOrder(sortingOrder) with
{
 OrderDropPoseOverride = "p[...]",
 ResortDropPoseOverride = "p[...]"
};
var script = SorterColorScriptBuilder.Build(options);
await Task.Run(() => _robotConnection.SendUrscript(script));
await _orderOperation.UpdateStatusAsync(orderId,
OrderStatus.Completed);
```

Logging + idle timeout auto-logout

```
private void _log(string s)
{
 var now = DateTime.Now.ToString("yy-MM-dd HH:mm:ss");
 LogOutput.Text += $"{now} | {s}\n";
}

_idleTimer = new DispatcherTimer { Interval =
TimeSpan.FromMinutes(5) };
_idleTimer.Tick += (_, __) => AutoLogout();
```