RMIT UNIVERSITY — School of Science, Engineering & Technology

ABB

Scan for more information

| Team Members | Supervisors | Project Title |
|---|---|---|
| **1BPad**<br>Nguyen Quang Son<br>Phan Nguyen Nhat Duy<br>Duong Hai Nam<br>Le Vo Nhat Minh<br>Pham Dinh Kien | Dr Nguyen Hai Nguyen<br>**(Academic)**<br>Mr Dien Duc Uy Man<br>**(Industry)** | Vision Distribution Enhancement for PickMaster Twin 3<br>**Pick & Place Applications** |

## Background and Motivation

With the advancement of Industry 4.0, robotic automation systems have become increasingly critical in enhancing production efficiency, precision, and adaptability. Pick & Place technologies are particularly vital in high-throughput sectors such as food processing, pharmaceuticals, and logistics, where rapid and accurate object handling is essential. **ABB's PickMaster Twin 3** system addresses these demands through integrated spatial recognition and vision-based control, enabling real-time object detection and placement. However, practical deployment often necessitates system-level customization to meet specific operational constraints. This project investigates such adaptations, focusing on algorithmic refinement and external vision integration to optimize system performance in real-world applications.

## Objectives

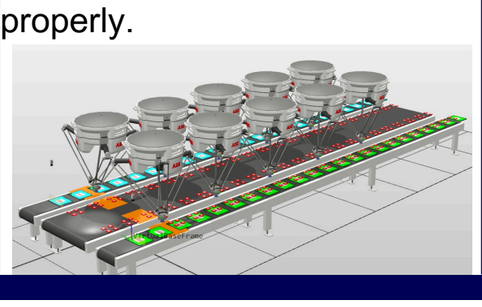
This project aims to boost the efficiency and flexibility of the ABB **IRB 360 Delta Robot** for high-speed pick-and-place applications by using **PickMaster Twin 3**. The goal is to develop custom **Python-based** algorithms to improve item distribution, filtering, and sorting.

## Technologies

## Methodology

### Simulation Plan

Before the simulation phase, a plan for how to place the robots has to be decided. The robots would have to showcase their ability to avoid collisions by the end of the project as well as an acceptable cycle time, space saving. Therefore, we have come up with **2 plans** to position the robots: 1 with **10 robots separated from each other** and 1 with **5 columns of 2 robots**, expecting a higher risk of collision if not handled properly.



### Virtual Robot Creation

In the following step, we would have to use **RobotStudio** to create **10 Virtual IRB360 Robot** Controller, which would then be used in the **Pickmaster** app for simulation. In this simulation foundation phase, all the installation configuration has to be done correctly so that the robots can function properly later on.

### Production Line Simulation

In this phase, the production line would be simulated using **PickMaster PowerPacs**. All the basic functions of the robots, like picking the objects and placing them in containers, up to the flow of items, the conveyor speeds are all simulated inside PickMaster 3.



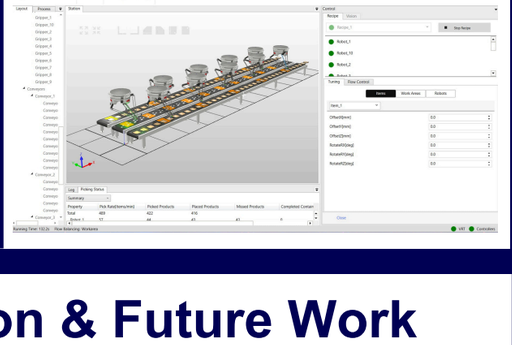
### Algorithm Development

After integrating the virtual production line, we would then have to use the user script function written in **Python** so that the robots would act as we want. Moreover, we would have to make the robots pick in the most optimized way possible - **smallest cycle time, no collision, all robots are utilized, and no items are unintentionally left behind**.

```python
    # Loop through each item in the dictionary to find the one closest to Y=0.
    for key, item in items.items():
        # The 'Time' key is also in the dictionary, so we skip it.
        if key == 'Time':
            continue

        # Calculate the absolute distance from the centerline (Y=0)
        distance_from_middle = abs(item['Y'])

        if distance_from_middle < smallest_y_distance:
            smallest_y_distance = distance_from_middle
            middle_item_key = key

# --- Filter the items ---
# Now, loop through the items again.
# Keep the 'middle_item' and discard all others by setting their 'Level' to 0.

if middle_item_key is not None:
    for key, item in items.items():
        if key == 'Time':
            continue

        # If the current item is NOT the middle item we found...
        if key != middle_item_key:
            # Set its 'Level' to 0.
            # Level 0 = Discarded (Robot ignores it)
            # Level 1 = Rejected
            # Level 2 = Accepted (Robot will try to pick it)
            items[key]['Level'] = 0
```

### Deployment & Evaluation

In this final phase, we would only have to do to debugging, check if the robot works as what we expect. Moreover, we would have to repeat the previous steps, create numerous test cases and test to make sure that the function is optimized in all cases. In our project, we have up to **7 item flow cases and 2 robot set up cases**, numerous test codes to compare and have the final best result.



## Experiment & Result

We have tried all the test cases, none of the item flow case that the robot miss, no robots collide each other during simulation since they are all given separated task, the cycle time is optimized and the robots are coded so that all of them are involved.





```python
        distance_from_middle = abs(item['Y'])

        if distance_from_middle < smallest_y_distance:
            smallest_y_distance = distance_from_middle
            middle_item_key = key

# --- Filter the items ---
# Now, loop through the items again.
# Keep the 'middle_item' and discard all others by setting their 'Level' to 0.

if middle_item_key is not None:
    for key, item in items.items():
        if key == 'Time':
            continue
```

## Conclusion & Future Work

- PickMaster Twin 3 made it much easier for us to build a stable and efficient pick-and-place line.
- In the future, the team hopes to create a complete "Digital Twin" for real-time monitoring, and connect the robots to a factory's MES/ERP management systems for total automation.
- More work could be done to have smaller cycle time.