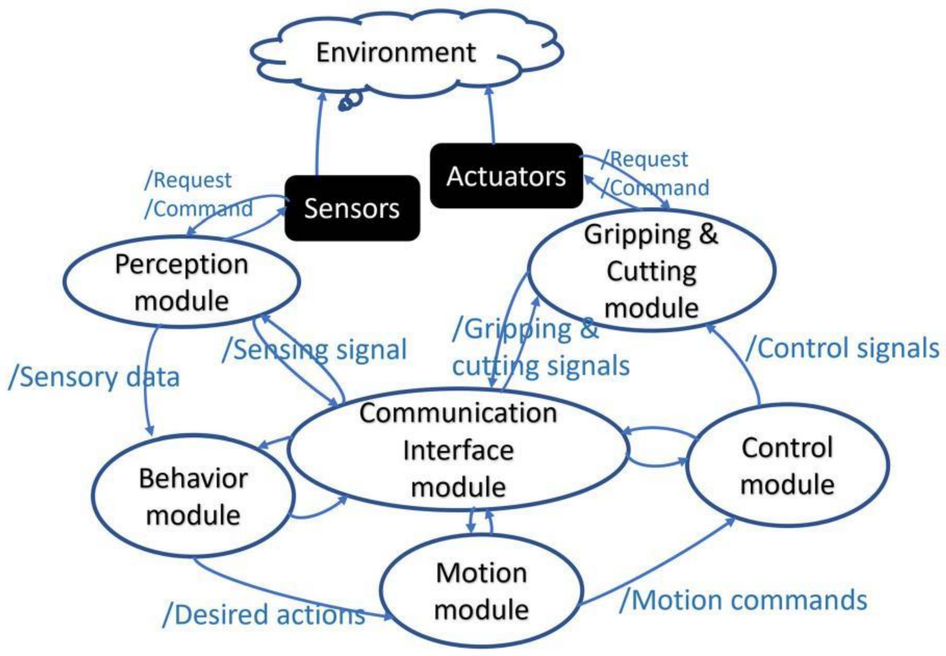

Fig. 1. Proposed behavior-based software architectural pattern for greenhouse harvesting robot.

uncertainties and variations in the greenhouse environment.

However, it comes with drawbacks, such as: 1) potentially less effective in long-term planning, 2) might struggle with complex decision-making scenarios.

Hence, a deliberative architecture could also be chosen that excel in precision and long-term planning. They can potentially optimize harvesting routes, taking into account the varying ripeness of tomatoes and the layout of the greenhouse. However, the tradeoff could be a potentially slower response time to immediate environmental changes. Deliberative architectures might struggle with rapid, reactive responses needed for unexpected events, which behavior-based architectures are designed to handle.

B. Food Processing Robots

The food-processing use-case addresses poultry processing and more specifically, the composition of product (chicken fillets, wings, thighs) placed densely in a bin and the robot should be able to pick products one-by-one at a time to place them on conveyor belt at right pose. This use-case addresses the challenge of how to deal with variation in shape and size of products, and deformability of objects.

In the case of pick and place of chicken pieces with a robot arm in the food processing industry, the software architecture must be designed to ensure efficient and accurate handling of the chicken pieces. The robot arm needs to have a reliable and accurate sensor system to detect the position and orientation of the chicken pieces. The software should be able to interpret the sensor data and plan the motion of the robot arm to pick up and place the chicken pieces in the desired location. The software must also take into account the safety of the workers around the robot and avoid collisions with other objects or machinery.

In this case study, the software architecture could be designed based on a layered architecture, with the application layer at the top, service layer at the middle and infrastructure layer at the bottom as explained in the following.

1) *Application layer:* This layer represents the topmost layer of the architecture and includes components related to application-specific business logic and user interfaces. It handles high-level tasks, such as user interaction, command execution, and overall coordination, of the pick and place operations.
2) *Service layer:* The service layer contains modules responsible for specific functionalities required for the pick and place task. This includes motion planning, object detection (to locate and recognize chicken pieces), and collision avoidance (to prevent collisions with objects or workers). These modules process data and events received from lower layers and generate relevant outputs.
3) *Infrastructure layer:* The infrastructure layer provides the necessary infrastructure components to support the functionality of higher layers. This includes sensor interfaces to communicate with sensors for precise detection of chicken pieces, actuator interfaces to control the robot arm's movements, communication modules for intermodule communication, and integration with external systems if required.

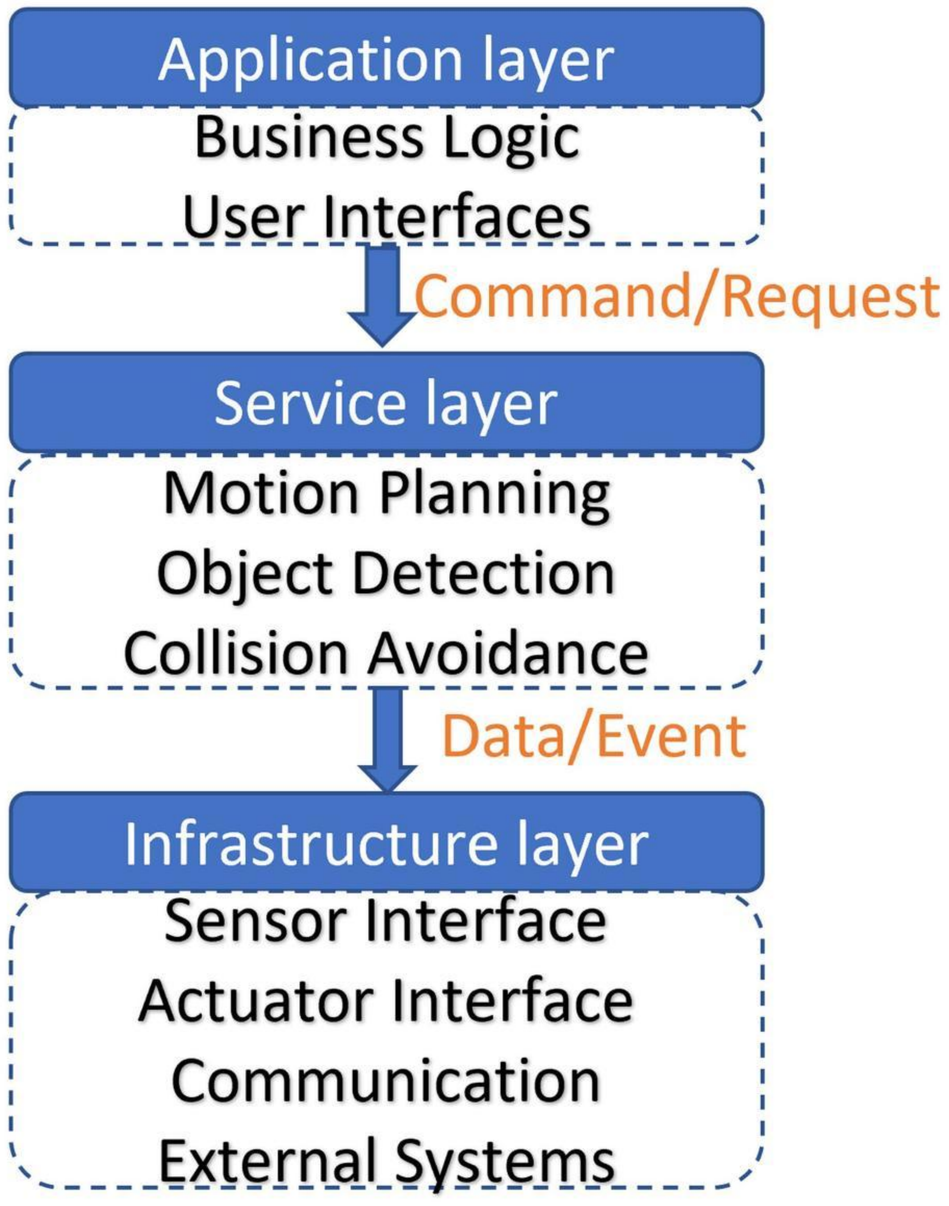

Fig. 2. Proposed layered software architecture for pick and place robot to process food.

The layered architecture provides a clear separation of concerns and modularization of functionalities shown in Fig. 2. Each layer focuses on specific aspects of the system and can be developed, tested, and maintained independently. The layers communicate through well-defined interfaces, allowing for flexibility, modifiability, and reusability.

The application layer handles high-level tasks and user interactions, while the service layer incorporates the core functionalities related to motion planning, object detection, and collision avoidance. The infrastructure layer provides the necessary infrastructure components for sensor integration, actuator control, communication, and integration with external systems.