TABLE I
REQUIREMENTS OF SOFTWARE ARCHITECTURE FOR THE AGRICULTURAL APPLICATIONS

Requirements	Description
Agility	The ability to respond quickly to a constantly changing environment. This feature is extremely important as the agricultural environment is dynamic in nature, that is it frequently changes over time.
Flexibility	The ability to be flexible enough to accommodate changes and modifications to the system over time. Patterns such as MVC and microservices provide the flexibility to modify and add functionality.
Complexity	A good architectural pattern should not introduce unnecessary complexity into the system. Patterns such as monolithic architecture can be simple, but can become complex as the application grows. Patterns, such as Microservices can be more complex to manage, but provide greater flexibility.
Real-time processing	The software architecture should be capable of processing data in real-time to enable quick decision-making and response to changing conditions.
Deployment	This can be a problem depending on how a pattern is implemented, especially for larger applications. A small change in a component can affect the application partially or completely, necessitating a partial or complete redeployment.
Adaptability	the ability to quickly adapt to complex, dynamic, and often chaotic environments.
Maintainability	It is the ability of a system to adapt to changes with ease.
Reusability	It is the ability of components and subsystems to be reused in other applications.
Testability	It is the ability to easily test components belonging to specific layers in the architecture, other layers can be mocked or stubbed. To isolate testing within a business component, a developer can mock a presentation component or screen, as well as mock the business layer to test specific screen functionality. Patterns such as MVC and layered architecture provide clear separation of concerns, making it easier to test code.
Interoperability	It is the ability of a system or systems to communicate with and exchange information with other external systems written and run by third parties.
Manageability	It refers to how simple it is for system administrators to manage the application.
Reliability	It is the ability of a system to remain operational over time.
Performance	It indicates the responsiveness of a system to execute any action within a given time interval.
Scalability	It is a property of a system that allows it to handle an increasing amount of work by adding resources to the system, i.e. the ability to change the size or scale of the system.
Modularity	The software architecture should be modular, allowing for easy integration of new components and functionalities as needed.
Cost	Cost of the system in terms of time to market, expected project lifetime, and legacy usage.
Security	It refers to a system's ability to prevent malicious or unintentional actions that are not intended.
Portability	It is the ability of a system to run in a variety of computing environments.
Fault tolerance	It is the ability to continue operating after a fault has occurred.
Marketability	It refers to how a system is used in relation to market competition.

The robot can be equipped with various end-effectors, such as grippers, cutting tools, and sprayers, to perform specific tasks.

III. Requirements and Challenges

The level of uncertainty in the agricultural environment is significantly higher than in other robotics applications. The agricultural climate is very complex in nature, meaning that it changes at all times. As a result, agricultural robots rely not only on advanced technology, but also on appropriate robot architecture, supported infrastructures, and services, such as communication between subsystems and data sharing and reuse. The unavailability of suitable robot architecture that can handle the existence of uncertainty in the agricultural environment may be one of the causes of this phenomenon.

A. Requirements for Agricultural Robot Software Architecture

The overall requirements for the agricultural applications are listed in Table I below, which should be considered while making a robot architectural pattern.

B. Challenges in Developing Agricultural Robot Software Architecture

There are some challenges listed in the Table II in developing agricultural robot software architecture considering all the requirements are as follows.

IV. Characterization of Current Robotic Architectures and Principles

Robot architectural patterns define the overall structure and organization of a software system within the robot(s), and provide a blueprint for the development team to follow. In general, each architectural pattern has its own tradeoffs, and the choice of pattern will depend on the specific requirements of the system being developed. It is necessary to look for specific characteristics in architectural patterns that are appropriate for agricultural applications. Because warehouse or manufacturing-related architectural patterns may not be appropriate for agricultural applications, as it is highly dependent on environmental conditions, specific tasks, and goals, and because the agricultural field's environment is highly dynamic, uncertain, and unpredictable in nature.

A. Categorization Based on Functionality

There are several ways to categorize robotic software architectural patterns, depending on the criteria or characteristics that are being used. For example, it can be categories based on functionality of the robot and how it performs its tasks. Examples include:

1) Control Architecture Versus Cognitive Architecture: Control architecture focuses on the control of robotic systems, including low-level motor control and high-level planning and decision-making, while cognitive architecture focuses on the