SECTION-5

9)

a)Why we need ROS for the development of Robots, give some examples.

- **ROS is general** :The base code and knowledge can be applied to many different kinds of robots .
  examples: robotic arms, drones, mobile bases
- **ROS packages for everything**: many ROS packages are available for almost any robotic application.
- **ROS is language-agnostic** :You can easily communicate between a Python node and a C++ node(interconnection between two devices).
- **ROS has great simulation tools :** You can't always get your robot running for real, so you need simulation tools. ROS has many great tools, such as Rviz and Gazebo
- **ROS can control multiple robots :** It means that you can have many independent robots, each with its own ROS system, and all robots can communicate between each other.
- **ROS is light** :The core base of ROS doesn't take much space and resources.
- **ROS is an open source project with a permissive license:** Most of the core packages are released under a BSD license. A BSD license allows you to modify and use the code for commercial purposes, without having to release your code with an open source license.

B) How to develop simple self balancing robot system.

Here is an outline of the steps that could be taken to develop a simple self-balancing robot system:

1. **Design the hardware:** The first step in building a self-balancing robot is to design the hardware. This includes selecting the motors, wheels, and other components that will be used to build the robot.

2. **Assemble the robot:** Once the hardware has been selected, the next step is to assemble the robot. This involves physically putting the robot together, including mounting the motors, wheels, and other components.
3. **Configure the control system**: The self-balancing robot will need a control system to maintain its balance. This can be done using a microcontroller, such as an Arduino, or a more sophisticated control system, such as a PLC. The control system will need to be configured and programmed to maintain the balance of the robot.
4. **Test and refine the control system:** Once the control system is set up, it will be necessary to test the robot and refine the control system as needed. This may involve adjusting the control algorithms or changing the hardware to improve the performance of the robot.
5. **Add additional features:** Depending on the goals of the project, it may be desirable to add additional features to the robot, such as sensors, cameras, or other capabilities. These can be incorporated into the system as needed.

10)

a) How to test the functionality and specific parameters of an 3 axis robotic hand build using ROS.

- **Define the testing goals:** Before testing begins, it will be important to define the specific parameters and functionality that need to be tested. This could include things like the range of motion, precision of movement, and force capabilities of the hand.
- **Set up a testing environment:** The next step is to set up a testing environment that will allow the hand to be tested under a variety of conditions. This could include a test stand or other equipment to hold the hand in place and allow it to move through its range of motion
- **Calibrate the hand:** Before testing begins, it will be important to calibrate the hand to ensure that it is operating correctly. This could involve adjusting the control parameters or fine-tuning the hardware as needed.

- **Test the basic functionality:** Once the hand is calibrated, the next step is to test the basic functionality of the hand. This could include testing the range of motion, precision of movement, and force capabilities of the hand
- **Analyze the results:** After testing is complete, it will be important to analyze the results to determine whether the hand is functioning as expected. Any problems or issues that are identified can be addressed and resolved as needed.

b) Give the application of ROS over microcontroller program.

Microcontrollers are used in almost every robotic product.

Typical reasons are:

- Sensors cannot be directly connected with the ROS. So, a microcontroller is used as an interfacing device for the sensor
- By using microcontroller in ROS power can be saved, the hardware is accessible and Hard low latency real time
- Another important reason is safety, but note that micro-ROS is not developed according to any safety standard.

Key features:

- Microcontroller-optimized client API(Application Programming Interface) supporting all major ROS concepts
- Seamless integration with ROS 2.
- Extremely resource-constrained but flexible middleware.
- Permissive license
- Vibrant community and ecosystem

there are three applications:-

- **Coordination**: ROS provides a central master node that can coordinate communication between different parts of a robot system. This allows for more complex and sophisticated interactions between the various components of the system.
- **Reusable code**: ROS includes a large number of libraries and tools for common robotics tasks, such as localization, path planning, and object

recognition. This means that developers can reuse existing code and algorithms, rather than having to implement these tasks from scratch for every new application.

- **Scalability**: ROS is designed to be scalable, which means that it can be used for small, simple robot systems as well as large, complex ones. This makes it a good choice for a wide range of robotics applications.