**ROS 2 Allgemein**

1. **Was ist ROS 2, und welche Vorteile bietet es im Vergleich zu ROS 1?**
   * ROS 2 ist die Weiterentwicklung von ROS 1 mit Verbesserungen in Echtzeitfähigkeit, Multithreading, Sicherheitsfunktionen und nativer Unterstützung für mehrere Plattformen. Es verwendet DDS (Data Distribution Service) für die Kommunikation, was flexibler und robuster ist.
2. **Was ist ein ROS 2 Node, und welche Aufgaben kann er übernehmen?**
   * Ein Node ist eine grundlegende Einheit in ROS 2, die spezifische Aufgaben übernimmt, wie Sensordaten erfassen, Befehle an Aktoren senden oder Daten analysieren. Jeder Node kommuniziert mit anderen Nodes über Topics, Services oder Actions.
3. **Wie werden ROS 2 Nodes erstellt, gestartet und verwaltet?**
   * Nodes werden in Python oder C++ erstellt und mit dem rclcpp- bzw. rclpy-Modul implementiert. Sie werden über Befehle wie ros2 run package\_name node\_name gestartet und können mit Tools wie ros2 node list überwacht werden.
4. **Welche Kommunikationsarten gibt es in ROS 2 (Topics, Services, Actions), und wie unterscheiden sie sich?**
   * **Topics**: Für die Veröffentlichung/Abonnement von Daten (asynchron).
   * **Services**: Für synchronen, einmaligen Datenaustausch.
   * **Actions**: Für lang andauernde Aufgaben mit Feedback und Möglichkeit zur Vorzeitigen Beendigung.
5. **Wie wird eine Nachricht in ROS 2 definiert, und wie können benutzerdefinierte Nachrichten erstellt werden?**
   * Nachrichten werden in .msg-Dateien definiert, die Felder mit Datentypen enthalten. Benutzerdefinierte Nachrichten werden durch Erstellen eines Pakets mit einem msg-Verzeichnis und der entsprechenden Definition erzeugt.
6. **Welche Rolle spielt der DDS (Data Distribution Service) in der Kommunikation von ROS 2?**
   * DDS ist ein Protokoll, das die Middleware von ROS 2 bildet. Es sorgt für zuverlässige und flexible Kommunikation zwischen Nodes, unabhängig davon, ob sie auf demselben Rechner oder über ein Netzwerk verteilt laufen.
7. **Was sind ROS 2 Namespaces, und wie helfen sie bei der Organisation von Nodes und Topics?**
   * Namespaces sind hierarchische Namensräume, die Nodes, Topics oder Parameter gruppieren, um Namenskonflikte zu vermeiden. Sie ermöglichen eine klare Strukturierung in komplexen Systemen.
8. **Welche Funktionen erfüllt ein ROS 2 Parameter, und wie wird er in einem Node verwendet?**
   * Parameter dienen dazu, Konfigurationen dynamisch zu laden, ohne den Code zu ändern. Sie können mit declare\_parameter in einem Node definiert und mit get\_parameter abgerufen werden.
9. **Wie können ROS 2 Launch-Dateien genutzt werden, um komplexe Systeme zu starten?**
   * Launch-Dateien ermöglichen es, mehrere Nodes, Parameter und Umgebungsvariablen zu starten. Sie werden in Python oder XML definiert und mit ros2 launch package\_name launch\_file ausgeführt.
10. **Welche Rolle spielen ROS 2 Lifecycle Nodes, und wie unterscheiden sie sich von regulären Nodes?**
    * Lifecycle Nodes haben definierte Zustände (z. B. Unconfigured, Inactive, Active), die kontrolliertes Aktivieren und Deaktivieren ermöglichen. Sie sind nützlich für deterministische Systeme.

**ROS 2 Kommunikation und Integration**

1. **Wie wird die Kommunikation zwischen ROS 2 Nodes sichergestellt, wenn diese auf verschiedenen Rechnern laufen?**
   * DDS ermöglicht Netzwerkkonfigurationen über Multicast oder Unicast. Die Umgebungsvariable ROS\_DOMAIN\_ID wird verwendet, um Kommunikationsbereiche zu isolieren.
2. **Wie werden Actions in ROS 2 implementiert, und in welchen Szenarien sind sie sinnvoll?**
   * Actions bestehen aus einer Zielnachricht, Feedback und einem Ergebnis. Sie werden verwendet, wenn eine Aufgabe Zeit benötigt, z. B. das Bewegen eines Roboterarms.
3. **Welche Herausforderungen können bei der Synchronisation von Daten zwischen Nodes auftreten, und wie lassen sich diese lösen?**
   * Probleme wie Datenlatenz oder nicht synchronisierte Zeitstempel können auftreten. Sie werden durch Time Synchronization Services oder die Verwendung von use\_sim\_time gelöst.
4. **Wie können Sensor- und Aktordaten effizient in ROS 2 übertragen werden?**
   * Über Topics mit optimierten Nachrichtenformaten wie sensor\_msgs oder durch Komprimierung der Daten.
5. **Welche Debugging-Werkzeuge stehen zur Verfügung, um ROS 2-Kommunikationsprobleme zu beheben?**
   * Tools wie ros2 topic echo, ros2 node info, und rqt\_graph helfen bei der Fehlersuche. Für tiefergehende Analysen kann Wireshark mit DDS-Protokollierung verwendet werden.

**Gazebo und Simulation**

1. **Was ist Gazebo, und wie unterstützt es die Entwicklung und Simulation von Robotern?**
   * Gazebo ist eine 3D-Simulationsumgebung, die physikalisch realistische Modelle, Sensoren und Umgebungen bereitstellt, um Roboter zu testen und zu entwickeln.
2. **Wie wird ein Roboter in Gazebo modelliert, und welche Rolle spielt eine URDF- oder xacro-Datei dabei?**
   * URDF/xacro-Dateien definieren die kinematische Struktur, Masse und Materialien eines Roboters. Gazebo nutzt diese Dateien, um das Modell und seine Physik darzustellen.
3. **Welche Schritte sind notwendig, um Gazebo mit ROS 2 zu integrieren?**
   * Ein Plugin wie gazebo\_ros2\_control wird hinzugefügt, um ROS 2-Kommunikation mit Gazebo zu ermöglichen. Launch-Dateien starten die Simulation und laden das Robotermodell.
4. **Welche physikalischen Parameter können in einer Gazebo-Simulation angepasst werden, und wie beeinflussen sie das Verhalten eines Roboters?**
   * Parameter wie Masse, Reibung, Dämpfung und Elastizität beeinflussen die Stabilität und Genauigkeit der Simulation.
5. **Wie können Gazebo-Simulationen genutzt werden, um reale Robotersysteme zu testen und zu validieren?**
   * Simulationen helfen, Bewegungspläne zu testen und Sensordaten zu analysieren, bevor ein Roboter physisch eingesetzt wird. Dies reduziert Kosten und Risiken.