**ROS**

**İÇİNDEKİLER**

|  |  |
| --- | --- |
|  | **Sayfa No** |
| ROS Nedir ve Ne İçin Kullanılır ? | **1** |
| ROS İçin Uygun Versiyon Seçimi | **2** |
| Gazebo Nedir ve Ne İçin Kullanılır ? | **3** |
| ROS için Kurulum Ortamı | **4** |
| ROS Avantaj ve Dezavantajları | **7** |
| Gazebo Nedir ? | **8** |
| ROS Örnek API Kullanımı | **9** |
| Gazebo ve ROS ile Birlikte Simülasyon Hazırlama | **10** |
| ROS Donanım Seçimi Nasıl Yapılır? | **11** |
| Sensör ve Aktüatör Seçimi | **12** |
| IoT Nedir? | **13** |
| ROS ile IoT Uygulamaları | **14** |
| Haberleşme Protokolü Seçimi | **15** |
| Örnek Proje Devre Şeması | **23** |
| Örnek Proje Çıktıları | **24** |
| Kaynakça | **25** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**ROS Nedir ve Ne İçin Kullanılır ?**

ROS (Robot Operating Systems) bir robot yazılım platformudur. ROS, birçok robot uygulama için kullanılabilen birçok araç ve kütüphane içermektedir. Bu araçlar; hareket planlama, görüntü işleme, sensör verisi işleme ve robot kontrolü gibi önemli robotik alanlarda kullanılabilir.

ROS, açık kaynak kodlu bir yazılım olduğu için, kullanıcılar tarafından geliştirilen eklentiler ve modüller de mevcuttur. Bu sayede, kullanıcılar kendi özel ihtiyaçlarına göre ROS'u genişletebilir ve kişiselleştirebilirler.

ROS, birden fazla robot sisteminde kullanılabilir. Örneğin, birçok insansı robot, mobil robot ve UAV (İnsansız Hava Aracı – Unmanned Air Vehicle) projesinde ROS kullanılmaktadır. Ayrıca, ROS, çeşitli sensör ve aktüatörlerle birlikte kullanılabilir. Bu sayede robotlar, görüntü, ses, doku ve hatta koku gibi farklı sensörlerle donatılabilir.

ROS, robotların kontrolü için kullanılan yazılımın bir parçası olarak kullanılabilir. Örneğin, hareket planlama modülü, robotun nereye gitmesi gerektiğini belirlerken, robot kontrol modülü ise robotun nasıl gitmesi gerektiğini belirler. Ayrıca, ROS, robotların sensör verilerini işlemek için de kullanılabilir. Örneğin, görüntü işleme modülü, robotun çevresindeki nesneleri tanımasına ve takip etmesine olanak tanır.

Sonuç olarak, ROS, robot yazılımının geliştirilmesi için idealdir. Özellikle, hareket planlama, görüntü işleme, sensör veri işleme ve robot kontrolü gibi alanlarda kullanılabilir. Ayrıca, açık kaynak kodlu olması nedeniyle kullanıcılar tarafından geliştirilen eklentiler ve modüller mevcuttur. Bu sayede, kullanıcılar kendi özel ihtiyaçlarına göre ROS'u genişletebilir ve kişiselleştirebilirler. ROS, birden fazla robot sisteminde kullanılabilir ve çeşitli sensör ve aktüatörlerle birlikte kullanılabilir. ROS, profesyonel robotik uygulamaların yanı sıra, araştırma ve öğrenme amaçlı kullanım için de idealdir. Özellikle, robotik eğitiminde, ROS kullanımı, öğrencilerin robotik konuları daha iyi anlamalarına ve uygulamalarına olanak tanır.

**ROS için Uygun Versiyon Seçimi ?**

ROS (Robot Operating System) için versiyon seçimi, projenin ihtiyacına ve hedeflerine göre yapılmalıdır. Öncelikle, projenizde kullanmak istediğiniz robotik ekipmanların ve sensörlerin ROS destekleme durumunu kontrol etmelisiniz. Örneğin, birçok insansı robot, mobil robot ve UAV (Uçuş Arabası) projesinde ROS kullanılmaktadır, ancak tüm versiyonları tüm ekipmanlarla uyumlu olmayabilir.

ROS versiyonları, iki yılda bir yayınlanmaktadır ve her versiyon, önceki versiyonlarla uyumlu olmayabilecek değişiklikler içerebilir. Örneğin, ROS Kinetic Kame, ROS Lunar Loggerhead ve ROS Melodic Morenia gibi versiyonlar arasında API (Application Programming Interface) değişiklikleri olabilir. Bu nedenle, projenizde kullanmak istediğiniz ekipmanların ve sensörlerin desteklediği versiyonu kontrol etmelisiniz.

Daha eski versiyonlar, daha stabil ve daha az hata içerebilir, ancak daha yeni versiyonlar daha fazla özellik ve daha iyi destek sunabilir. Örneğin, ROS Kinetic Kame, ROS Lunar Loggerhead ve ROS Melodic Morenia gibi versiyonlar arasında özellikler açısından farklılıklar olabilir. Bu nedenle, projenizin ihtiyacına göre versiyon seçimi yapmalısınız.

Sonuç olarak, ROS için versiyon seçimi, projenin ihtiyacına ve hedeflerine göre yapılmalıdır. Projenizde kullanmak istediğiniz robotik ekipmanların ve sensörlerin destekleme durumunu kontrol etmeli ve projenizin ihtiyacına göre versiyon seçmelisiniz. Ayrıca, eski versiyonlar daha stabil olabilirken, yeni versiyonlar daha fazla özellik sunabilir. Bu nedenle, projenizin ihtiyacına göre versiyon seçimi yapmalısınız.

1. ROS Noetic Ninjemys: Bu versiyon Mayıs 2020'de yayınlandı ve Ubuntu 20.04 LTS (Focal Fossa) temel alınmıştır. Hem Python 2 hem de Python 3 ile uyumludur. Bu versiyondaki yeni özellikler arasında, OpenCV, PCL ve Gazebo gibi popüler kütüphanelerin son sürümlerini kullanabilme, ARM tabanlı mimarilerle ROS kullanabilme gibi özellikler bulunmaktadır.
2. ROS Melodic Morenia: Bu versiyon Mayıs 2018'de yayınlandı ve Ubuntu 18.04 LTS (Bionic Beaver) temel alınmıştır. Python 2 ve Python 3 ile uyumludur. Bu versiyondaki yeni özellikler arasında, derinlik kameraları ve lidarlar gibi sensörlerle ROS kullanabilme, dronlar ve sualtı robotlar gibi robotik platformlarla ROS kullanabilme gibi özellikler bulunmaktadır.
3. ROS Lunar Loggerhead: Bu versiyon Ocak 2018'de yayınlandı ve Ubuntu 17.04 (Zesty Zapus) temel alınmıştır. Python 2 ve Python 3 ile uyumludur. Bu versiyondaki yeni özellikler arasında, dronlar ve sualtı robotlar gibi robotik platformlarla ROS kullanabilme, derinlik kameraları ve lidarlar gibi sensörlerle ROS kullanabilme gibi özellikler bulunmaktadır.
4. ROS Kinetic Kame: Bu versiyon Mayıs 2016'da yayınlandı ve Ubuntu 16.04 LTS (Xenial Xerus) temel alınmıştır. Hem Python 2 hem de Python 3 ile uyumludur. Bu versiyondaki yeni özellikler arasında, dronlar ve sualtı robotlar gibi robotik platformlarla ROS kullanabilme, derinlik kameraları ve lidarlar gibi sensörlerle ROS kullanabilme gibi özellikler bulunmaktadır.

**Gazebo Nedir ve Ne İçin Kullanılır ?**

Gazebo, bir robotik simülasyon ortamıdır. Gazebo, fiziksel dinamikler, sensörler ve robot kontrolü gibi robotik sistemleri simüle etmek için kullanılır. Bu sayede, robotik sistemlerin tasarımı, test edilmesi ve eğitilmesi için ideal bir ortam sağlar. Örneğin, Gazebo kullanarak bir mobil robotun nasıl davranacağını çeşitli senaryolarda simüle edebilirsiniz.

Gazebo, ROS ile birlikte kullanılabilir. ROS, robotik sistemlerin yazılımını yönetirken, Gazebo ise fiziksel dinamikleri simüle eder. Bu sayede, Gazebo ile gerçek dünyadaki robotik sistemleri simüle edebilir ve ROS ile yazdığınız yazılımı bu simüle edilmiş ortamda test edebilirsiniz.

Gazebo, açık kaynak kodlu bir yazılım olduğu için, kurulumu oldukça kolaydır. Öncelikle, Gazebo'yu kurmak için gerekli olan tüm bağımlılıkları yüklemelisiniz. Daha sonra, Gazebo'yu kurmak için gerekli olan kurulum dosyasını indirmelisiniz ve kurulumu gerçekleştirmelisiniz. Ayrıca, Gazebo'yu ROS ile birlikte kullanacaksanız, ROS'un da kurulu olması gerekir.

Sonuç olarak, Gazebo, robotik sistemlerin tasarımı, test edilmesi ve eğitilmesi için ideal bir simülasyon ortamıdır. Gazebo, ROS ile birlikte kullanılabilir ve kurulumu oldukça kolaydır. Bu sayede, gerçek dünyadaki robotik sistemleri simüle edebilir ve yazılımınızı bu simüle edilmiş ortamda test edebilirsiniz.

**ROS İçin Kurulum Ortamı**

ROS’u kurmak için birkaç yöntem mevcuttur. Bu yöntemler sırasıyla:

* WSL Tabanlı Ubuntu Üzerine ROS
* Sanal Makine Tabanlı Ubuntu Üzerine ROS
* Mini Bilgisayar Tabanlı Ubuntu Üzerine ROS

Bu yöntemler için ilk yapılması gereken işlem Ubuntu Linux dağıtımını indirmektir. İndirme işlemine başlamadan önce muhakkak yönteme karar verilmelidir.

WSL ve Sanal Makine yöntemleri için ekstra hiçbir donanıma ihtiyacımız yoktur. Yalnızca Ubuntu Linux dağıtımının iso dosyası yani disk görüntüsünü indirmek yeterlidir.

Mini bilgisayar tabanlı yöntem için muhakkak bir hafıza kartına ihtiyacımız vardır. Bu hafıza kartının amacı seçilen mini bilgisayar için bir depolama birimi oluşturabilmektir. Bu yüzden minimum 8 GB’lık bir hafıza kartına ihtiyaç duymaktadır. Sırasıyla kurulum işlemleri:

WSL:

1. Öncelikle sistemimiz için windows subsystem linux’u aktif etmemiz gerek. Bunun için: Denetim Masası > Programlar ve Özellikler > Windows Özelliklerini Aç veya Kapat üzerinden Alt Sistemler için Linux’u aktif edip sistemimizi yeniden başlatıyoruz.
2. Ardından Windows üzerinden sanal makineyi PowerShell üzerinden etkinleştirmemiz gerekiyor. Bunun için: “dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart” komutunu yönetici olarak çalıştırdığımız PowerShell’e giriyoruz.
3. Ardından sırasıyla aşağıdaki komut ile wsl sürümümüzü 2 olarak güncelliyoruz.

“wsl --set-default-version 2”

1. WSL ve sanal makineyi ayarladıktan sonra Microsoft Store üzerinden Ubuntu’yu indiriyoruz.
2. Ardından ROS kurulumu için aşağıdaki komutları kullanıyoruz:

* sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
* sudo apt install curl
* curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
* sudo apt update
* sudo apt install ros-noetic-desktop-full
* apt search ros-noetic
* source /opt/ros/noetic/setup.bash
* sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
* sudo apt install python3-rosdep
* sudo rosdep init
* rosdep update

Sanal Makine:

1. Öncelikle aşağıdaki bağlantı üzerinden ubuntu linux dağıtımını indiriyoruz.

“https://ubuntu.com/download/alternative-downloads”

1. Ardından sanal makine kurulumu için aşağıdaki bağlantıdan VirtualBox’ı indiriyoruz.

<https://www.virtualbox.org/>

1. VirtualBox’ı kurduktan sonra daha önce indirdiğimiz linux dağıtımı iso dosyası ile birlikte bir sanal makine oluşturuyoruz. Artık sanal makine olarak bir ubuntuya sahibiz.
2. Ubuntuya eriştikten sonra gerekli kullanıcı işlemlerini yapıp terminali açıyoruz.
3. Ardından ROS kurulumu için aşağıdaki komutları kullanıyoruz:

* sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
* sudo apt install curl
* curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
* sudo apt update
* sudo apt install ros-noetic-desktop-full
* apt search ros-noetic
* source /opt/ros/noetic/setup.bash
* sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
* sudo apt install python3-rosdep
* sudo rosdep init
* rosdep update

Mini Bilgisayar:

1- Öncelikle aşağıdaki bağlantı üzerinden Raspberry Imager programını indiriyoruz.

“https://www.raspberrypi.com/software/”

1. İndirdiğimiz programı kullanarak bilgisayarımıza takmış olduğumuz hafıza kartına Imager programı üzerinden Ubuntuyu kuruyoruz.
2. Ardından hafıza kartını raspberry pi mini bilgisayarımıza takıp raspberry pi’ı enerji kaynağına bağlayıp ubuntu için gerekli kullanıcı kurulumunu tamamlıyoruz.
3. Ardından ROS kurulumu için aşağıdaki komutları kullanıyoruz:

* sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
* sudo apt install curl
* curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
* sudo apt update
* sudo apt install ros-noetic-desktop-full
* apt search ros-noetic
* source /opt/ros/noetic/setup.bash
* sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
* sudo apt install python3-rosdep
* sudo rosdep init
* rosdep update

**ROS Avantaj ve Dezavantajları Nelerdir ?**

ROS (Robot Operating System) robotik sistemler için yazılım platformudur. ROS, robotik sistemlerin tasarımı, test edilmesi, eğitilmesi ve üretimi için birçok avantaj sağlar. Ancak, herhangi bir yazılım platformu gibi, ROS'un da belli dezavantajları vardır.

ROS'un avantajları arasında:

* Modüler yapısı: ROS, birçok modül ve paketlerden oluşur. Bu sayede, kullanıcılar sadece ihtiyaçları olan modülleri kullanabilirler ve ROS'u kişiselleştirebilirler.
* Açık kaynak kodlu olması: ROS, açık kaynak kodlu bir platformdur. Bu sayede, kullanıcılar kodları inceleyebilirler, değiştirebilirler ve kendi modüllerini oluşturabilirler.
* Çoklu platform desteği: ROS, farklı işletim sistemleri ve farklı robotik platformlar için destek sağlar. Bu sayede, robotik sistemlerin üretimi için platform seçimine gerek kalmaz.
* Yüksek seviyeli API: ROS, yüksek seviyeli bir API sunar. Bu sayede, robotik sistemlerin tasarımı ve programlaması kolaylaşır.

ROS'un dezavantajları arasında:

* Öğrenme eğrisi yüksek: ROS, modüler ve yüksek seviyeli bir platform olduğu için öğrenmesi zaman alabilir. Kullanıcılar, ROS kullanmadan önce belli bir seviyede programlama bilgisi ve robotik bilgisi olması gerekir.
* Sistem kaynaklarını yüksek kullanır: ROS, modüler yapısı nedeniyle birden fazla işlemi aynı anda gerçekleştirebilir. Bu sayede, sistem kaynaklarını yüksek seviyede kullanabilir. Bu, daha az güçlü bilgisayarlarda veya daha az güçlü robotik sistemlerde sorunlar yaratabilir.
* API değişiklikleri: ROS, iki yılda bir yeni versiyon yayınlar. Bu versiyonlar arasında API değişiklikleri olabilir. Bu nedenle, eski versiyonlarla yazılmış kodlar yeni versiyonlarla uyumlu olmayabilir.
* Destek yetersizliği: ROS, açık kaynak kodlu bir platform olduğu için destek yetersizliği olabilir. Özellikle, daha az popüler modüller veya paketler için destek daha az olabilir. Bu nedenle, kullanıcılar kendilerine ait çözümleri geliştirmek veya başka kaynaklardan destek almak zorunda kalabilir.

Sonuç olarak, ROS robotik sistemler için yazılım platformudur ve birçok avantaj sağlar. Ancak, öğrenme eğrisi yüksek, sistem kaynaklarını yüksek kullanması, API değişiklikleri ve destek yetersizliği gibi dezavantajları da mevcut. Bu nedenle, kullanıcılar, kendi ihtiyaçlarına ve sistemlerine göre ROS'un avantajlarını ve dezavantajlarını dikkatli bir şekilde değerlendirmelidir.

**Gazebo Nedir ?**

Gazebo, bir robotik simülasyon ortamıdır. Gazebo, fiziksel dinamikler, sensörler ve robot kontrolü gibi robotik sistemleri simüle etmek için kullanılır. Bu sayede, robotik sistemlerin tasarımı, test edilmesi ve eğitilmesi için ideal bir ortam sağlar. Örneğin, Gazebo kullanarak bir mobil robotun nasıl davranacağını çeşitli senaryolarda simüle edebilirsiniz.

Gazebo, birçok fizik motoru ve sensör modeli desteği sunar. Örneğin, Gazebo kullanarak bir robotun nasıl yürüyeceği veya nasıl yükseklik algılayacağı gibi senaryolarda simüle edebilirsiniz. Ayrıca, Gazebo, gerçek dünyada mevcut olan robotik sistemlerin fiziksel modellerini kullanabilir ve bu modelleri simülasyonda kullanabilirsiniz.

Gazebo, ROS (Robot Operating System) ile birlikte kullanılabilir. ROS, robotik sistemlerin yazılımını yönetirken, Gazebo ise fiziksel dinamikleri simüle eder. Bu sayede, Gazebo ile gerçek dünyadaki robotik sistemleri simüle edebilir ve ROS ile yazdığınız yazılımı bu simüle edilmiş ortamda test edebilirsiniz. Bu nedenle, Gazebo ve ROS birlikte kullanıldığında, robotik sistemlerin tasarımı, test edilmesi ve eğitilmesi için çok etkili bir ortam oluşur.

Gazebo, açık kaynak kodlu bir yazılımdır ve kullanımı ücretsizdir. Bu nedenle, herkes tarafından kullanılabilir ve katkıda bulunulabilir. Ayrıca, Gazebo topluluğu, kullanıcıların sorularını ve problemlerini çözmek için her zaman yararlı olacaktır. Örneğin, mobil robotlar, quadrotors, manipulatorlar, uydu veya uzay platformları, veya gerçek zamanlı 3D simülasyonlar için Gazebo kullanılabilir.

**ROS Örnek API Kullanımı**

import rospy  
  
def talker():  
 pb = rospy.Publisher('chatter', String, queue\_size=10)  
 rospy.init\_node('talker', anonymous=True)  
 rate = rospy.Rate(10)  
 while not rospy.is\_shutdown():  
 hello\_str = "[Onder Grup] Saat: %s" % rospy.get\_time()  
 rospy.loginfo(hello\_str)  
 pub.publish(hello\_str)  
 rate.sleep()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 try:  
 talker()  
 except rospy.ROSInterruptException:  
 pass

Bu kod "chatter" üzerinde mesajları yayınlayan bir düğümü çalıştırır ve saniyede 10 kez yapar (rate = 10). rospy kütüphanesi, ROS ile etkileşim kurmak için kullanılır ve başlangıçta içe aktarılması gerekir.

rospy.Publisher, yayıncı nesnesi oluşturmak için kullanılır ve ilk argüman mesajların yayınlanacağı konunun adıdır, ikinci argüman yayınlanacak mesajın türüdür ve son argüman isteğe bağlı olarak kuyruk boyutudur.

rospy.init\_node, düğümü başlatmak için kullanılır ve ilk argüman düğümün adıdır ve ikincisi isteğe bağlıdır, düğümün anonim olup olmadığını belirtir.

rospy.Rate, mesajların yayınlanma hızını ayarlamak için kullanılır, bu durumda saniyede 10 mesaj. rospy.is\_shutdown() düğüm hala çalışıyor mu kontrol etmek için kullanılır.

rospy.loginfo, mesajı günlüğe kaydetmek için kullanılır ve pub.publish(hello\_str), mesajı konuda yayınlamak için kullanılır.

rate.sleep(), yayınlanan mesajların hızının sabit kalmasını sağlamak için rate time için uyku moduna geçirir.

**Gazebo ve ROS ile Birlikte Simülasyon Hazırlama**

Gazebo ve ROS, robot simülasyonlarının oluşturulması ve test edilmesi için popüler bir araç olarak kullanılmaktadır. Gazebo, robotların fiziksel dünyasını simüle etmek için kullanılan bir 3D dinamik simülasyon ortamıdır. ROS ise, robotlar arası bilgi alışverişi ve kontrolü için kullanılan bir robot işletim sistemidir.

Gazebo ve ROS'un birlikte kullanılması, gerçek dünya sensörleri, aktüatörler ve yazılımının simülasyon ortamında test edilmesini sağlar. Bu, robotların gerçek dünyada karşılaşabilecekleri problemleri önceden tahmin etme ve çözme imkanı verir. Aynı zamanda, robotların fiziksel dünyada gerçekleştiremeyecekleri görevleri simülasyon ortamında deneyebilmenizi de sağlar.

Gazebo ve ROS'un birlikte kullanılması için, Gazebo'nun ROS plugin'lerini kurmanız gerekir. Bu plugin'ler, Gazebo ile ROS arasında bilgi alışverişini sağlar. Örneğin, Gazebo'da oluşan sensör verileri ROS'a gönderilir ve ROS tarafından işlenir. Aynı şekilde, ROS tarafından verilen kontrol komutları Gazebo tarafından uygulanır.

Gazebo ve ROS'un birlikte kullanılmasının bir diğer avantajı, mevcut ROS paketlerinin veya kütüphanelerin kolayca simülasyon ortamına entegre edilmesidir. Bu, mevcut ROS projelerinin simülasyon ortamında test edilmesini ve geliştirilmesini kolaylaştırır. Ayrıca, Gazebo'da mevcut olan fizik motorları, kinematik modeller, sensör modelleri ve diğer araçlar, ROS projeleriniz için kullanılabilir.

Gazebo ve ROS'un birlikte kullanımı, robotik projelerinizi gerçek dünya koşullarına benzer bir ortamda test etmenizi ve geliştirmenizi sağlar. Bu sayede, projenizin gerçek dünyada daha iyi performans göstereceğine dair güven sağlar. Ayrıca, gerçek dünyada gerçekleştiremeyecekleri görevleri simülasyon ortamında deneyebilmeniz ve problemleri çözmeniz mümkündür. Bu nedenle, Gazebo ve ROS'un birlikte kullanımı, robotik projelerinizin geliştirilmesi ve test edilmesi için önemli bir araçtır.

**ROS Donanım Seçimi Nasıl Yapılır ?**

ROS (Robot Operating Systems) donanım seçimi, robotik projelerinizin performansını ve ölçeklenebilirliğini doğrudan etkileyebilecektir. Öncelikle, ROS ile çalışacak donanımın ROS destekli olması gerekir. Bu, donanımın ROS paketleriyle birlikte çalışmayabileceği anlamına gelir. Ayrıca, projenizin ihtiyaçlarına uygun bir donanım seçmeniz gerekir. Örneğin, bir mobil robot için bir LIDAR veya bir kameralı sensör gerekli olabilirken, bir manipulator robot için bir force torque sensörü gerekebilir.

ROS için sensör seçimi yaparken, öncelikle sensörlerin ölçüm aralığı, ölçüm hassasiyeti, ölçüm hızı ve güncelleme hızı gibi özelliklerine dikkat etmelisiniz. Ayrıca, sensörlerin çalışma aralığı, çalışma sıcaklığı, güç tüketimi ve boyutları gibi fiziksel özelliklerine de dikkat etmelisiniz.

ROS için popüler sensör seçenekleri arasında, LIDAR sensörleri, kameralar, IMU sensörleri, GPS sensörleri, force torque sensörleri ve hız sensörleri bulunmaktadır. LIDAR sensörleri, robotlar için çevre tarama veya yol takip gibi uygulamalar için ideal olarak kullanılır. Kameralar ise, görüntü işleme, nesne takibi veya yüz tanıma gibi uygulamalar için kullanılabilir. IMU sensörleri ise robotların pozisyonunu, hızını ve yönünü ölçmek için kullanılır. GPS sensörleri ise, robotların harita veya yer bilgisi kullanarak navigasyon yapmasını sağlar. Force torque sensörleri ise, robotların kuvvet ve moment ölçümü yapmasını sağlar. Hız sensörleri ise, robotların hız veya ivme ölçümü yapmasını sağlar.

Son olarak, ROS için sensör seçimi yaparken, sensörlerin ROS desteği, fiyat, ticari destek, doğruluk ve güvenilirlik gibi diğer faktörleri de dikkate almalısınız. Örneğin, bir sensörün ROS desteği olması önemliyken, aynı sensörün fiyatı veya ticari desteği yetersiz olabilir. Bu nedenle, projenizin ihtiyaçlarını ve bütçenizi dikkate alarak en uygun sensörü seçmelisiniz.

**Sensör ve Aktüatör Seçimi**

Sensör ve aktüatörler, otomatik kontrol sistemlerinin temel bileşenleridir. Sensörler, fiziksel değişkenleri (örneğin, sıcaklık, basınç, hız vb.) ölçer ve bunları elektrik sinyali olarak çevirir. Aktüatörler ise elektrik sinyallerini fiziksel hareketlere dönüştürür.

Sensör seçimi, sistemin gerektirdiği ölçüm aralığı, ölçüm hassasiyeti, ölçüm hızı, ölçüm aralığının değişebilirliği, çalışma koşulları (örneğin, sıcaklık, nem, aşırı şok vb.) ve maliyet gibi faktörleri dikkate almak gerekir. Örneğin, sıcaklık sensörü seçimi için, ölçüm aralığı (-200°C ila +1200°C arasında), ölçüm hassasiyeti (±0.1°C), ölçüm hızı (1 ölçüm/saniye) ve çalışma koşulları (ölçüm yapılan ortamda sıcaklık değişebilir) gibi faktörler dikkate alınmalıdır.

Aktüatör seçimi için, sistemin gerektirdiği hareket aralığı, hareket hızı, hareket hassasiyeti, çalışma koşulları (örneğin, sıcaklık, nem, aşırı şok vb.) ve maliyet gibi faktörler dikkate alınmalıdır. Örneğin, bir kapı açma/kapama sistemi için, hareket aralığı (0-90 derece), hareket hızı (1 saniye), hareket hassasiyeti (±1 derece) ve çalışma koşulları (kapının açılacağı ortamda hava koşulları değişebilir) gibi faktörler dikkate alınmalıdır.

Sonuç olarak, sensör ve aktüatör seçimi, sistemin gerektirdiği ölçüm/hareket aralığı, hassasiyeti, hızı ve çalışma koşulları gibi faktörler dikkate alınarak yapılmalıdır. Ayrıca, maliyet ve çevresel faktörler de dikkate alınmalıdır. Örneğin, sensörler ve aktüatörlerin çevre dostu olması, uzun ömürlü olması ve kolay bakım/değiştirme imkanlarına sahip olması gibi faktörler de seçim sürecinde dikkate alınmalıdır. Ayrıca, sensörler ve aktüatörlerin kompatibilitesi de önemlidir. Ölçülen/hareket ettirilen sistemle sensör/aktüatör arasında uyumlu bir bağlantı sağlanmalıdır.

Sonuç olarak, sensör ve aktüatör seçimi, otomatik kontrol sistemlerinin performansını ve güvenliğini etkileyen önemli bir faktördür. Bu seçim işlemi, sistemin gerektirdiği özellikleri ve çalışma koşullarını dikkate alarak yapılmalı ve uygun sensörler ve aktüatörler seçilerek, sistemin performansı ve güvenliği en yüksek seviyede tutulmalıdır.

**IoT Nedir ?**

"Internet of Things" (IoT) genellikle fiziksel nesnelerin internete bağlanmasını ve bu nesneler arasında veri alışverişini ifade eder. IoT, birçok farklı alanda kullanılmaktadır ve geniş bir yelpazeyi kapsamaktadır: sağlık, enerji, otomotiv, ev ve bina otomasyonu, tarım, güvenlik vb.

IoT sistemleri, çeşitli sensörler, aktüatörler ve cihazlar aracılığıyla gerçekleştirilir. Bu sensörler ve aktüatörler, fiziksel nesnelerin çevresinde yer alır ve nesnelerin durumunu veya çevresindeki değişiklikleri ölçerler veya nesnelerin davranışını değiştirirler. Bu sensörler ve aktüatörler, internete bağlanabilir veya internete bağlı bir merkezi sisteme bağlanabilirler. Bu merkezi sistem, sensör ve aktüatörler tarafından toplanan verileri işler ve bu verileri kullanıcılara veya diğer sistemlere iletebilir.

IoT teknolojisi, çeşitli faydalar sağlamaktadır. Örneğin, IoT sistemleri, endüstriyel süreçlerin otomatikleştirilmesi, enerji verimliliği, sağlık ve güvenlik, ev ve bina otomasyonu gibi alanlarda verimliliği arttırmaktadır. Ayrıca, IoT sistemleri, çevre izleme ve kontrol, trafik yönetimi, üretim takibi gibi alanlarda kullanılmaktadır.

Ancak, IoT teknolojisi aynı zamanda bazı güvenlik ve gizlilik sorunlarını da beraberinde getirmektedir. Örneğin, IoT cihazlarının internete bağlı olması, bu cihazların hacklenmesine veya yanıltılmasına açık hale getirmektedir. Ayrıca, IoT cihazları tarafından toplanan verilerin kullanılması veya paylaşılması, kişisel bilgilerin gizliliğini tehlikeye atabilir. Bu nedenle, IoT sistemlerinin tasarımı ve kullanımı sırasında güvenlik ve gizlilik önlemleri alınması önemlidir. Örneğin, IoT cihazlarının güncel parolalar kullanılması, cihazların internete bağlı olmadıkları sürelerde kapatılması, cihazların sadece güvenli kaynaklardan yazılım güncellemelerini alması gibi önlemler alınabilir. Ayrıca, IoT verilerinin işlenmesi ve paylaşılması sırasında, kişisel bilgilerin korunması için gerekli önlemler alınmalıdır.

Sonuç olarak, Internet of Things (IoT) teknolojisi, birçok farklı alanda verimliliği arttırmaktadır. Ancak, IoT sistemlerinin tasarımı ve kullanımı sırasında güvenlik ve gizlilik önlemleri alınması gereklidir. IoT sistemlerinin tasarımı ve kullanımı sırasında, güvenlik ve gizlilik önlemleri alınarak, IoT sistemlerinin potansiyel faydalarından maksimum düzeyde yararlanılabilir.

**ROS ile IoT Uygulamaları**

Robot Operating System (ROS) ve Internet of Things (IoT) teknolojileri, birçok farklı alanda birlikte kullanılmaktadır. ROS, birçok sensör ve aktüatörün arasındaki veri alışverişini ve bu sensörler ve aktüatörlerin yönetimini sağlar. IoT ise, fiziksel nesnelerin internete bağlanmasını ve bu nesneler arasında veri alışverişini sağlar. Bu nedenle, ROS ve IoT teknolojileri, birçok farklı alanda bir arada kullanılarak, verimliliği arttırmaktadır.

Örneğin, bir ev otomasyon sistemi için, sensörler aracılığıyla evdeki sıcaklık, nem, gün ışığı vb. ölçülebilir ve bu veriler ROS aracılığıyla işlenerek, aktüatörler aracılığıyla ısıtma, havalandırma vb. sistemlerinin otomatik olarak yönetilmesi sağlanabilir. Bu şekilde, enerji tasarrufu ve konfor sağlanabilir.

Ayrıca, bir tesis otomasyon sistemi için, sensörler aracılığıyla tesisdeki sıcaklık, basınç, hız vb. ölçülebilir ve bu veriler ROS aracılığıyla işlenerek, aktüatörler aracılığıyla süreçlerin otomatik olarak yönetilmesi sağlanabilir. Bu şekilde, verimliliği arttırmak ve hata oranını azaltmak mümkündür.

Aşağıda, bir ev otomasyon sistemi için ROS ve IoT teknolojilerinin kullanımı örneği verilmektedir. Bu örnekte, evdeki sıcaklık sensörü aracılığıyla ölçülen sıcaklık verileri, ROS aracılığıyla işlenir ve ısıtma sistemi aktüatörü aracılığıyla otomatik olarak yönetilir.

import rospy  
from std\_msgs.msg import Float32  
  
temperature = 0  
  
def temp\_callback(data):  
 global temperature  
 temperature = data.data  
  
rospy.init\_node('home\_automation')  
rospy.Subscriber('temperature\_sensor', Float32, temp\_callback)  
heating\_pub = rospy.Publisher('heating\_system', Float32, queue\_size=10)  
  
desired\_temp = 22  
  
while not rospy.is\_shutdown():  
 if temperature < desired\_temp:  
 heating\_pub.publish(1)  
 else:  
 heating\_pub.publish(0)  
rospy.sleep(1)

Bu örnekte, ROS aracılığıyla sıcaklık sensöründen gelen veriler işlenir ve ısıtma sistemi aktüatörü aracılığıyla otomatik olarak yönetilir. Bu şekilde, evdeki sıcaklık istenen seviyede tutulur ve enerji tasarrufu sağlanır. Aynı şekilde, bu prensip farklı alanlarda da kullanılabilir ve verimliliği arttırmak için IoT ve ROS teknolojileri bir arada kullanılabilir.

**Haberleşme Protokolü Seçimi**

LoRaWAN: LoRaWAN, geniş alan ağları için kullanılan bir haberleşme protokolüdür. İnternet of Things (IoT) cihazları için tasarlanmıştır ve uzun mesafelerde haberleşme ihtiyacı olan cihazlar için kullanılır. Genellikle kırsal alanlarda 15 km'ye kadar ve şehir içi alanlarda 2-5 km'ye kadar mesafede haberleşebilir. LoRaWAN, alt-gigahertz frekans bölgesini kullanır ve cihazlar arasındaki gürültüyü azaltır. LoRaWAN ağları genellikle bir ağ sağlayıcı tarafından işletilir ve cihazlar ağda haberleşebilmek için sağlayıcıya kaydedilmelidir. LoRaWAN'ın avantajları arasında uzun mesafe, düşük güç tüketimi ve düşük maliyet yer alır. Bir dezavantajı ise diğer haberleşme protokollerine göre daha yavaş olması ve yüksek bant genişliği gerektiren uygulamalar için uygun olmamasıdır.

import lorawan

lora = lorawan.LoRaWAN()

# Set the AppEUI and AppKey for the device

lora.set\_app\_eui("your\_app\_eui")

lora.set\_app\_key("your\_app\_key")

lora.send("[OnderGrup] Communication protocools send test code !")

#lorawan library: <https://github.com/brocaar/lorawan>

DALI: DALI (Digital Addressable Lighting Interface), bir aydınlatma kontrol sistemi için kullanılan bir haberleşme protokolüdür. Bir bina veya diğer tesis içinde aydınlatma donanımlarını kontrol etmek ve izlemek için kullanılır. DALI, RS-485 standartına dayanır ve haberleşme için iki telli bir bus kullanır. Ağdaki her cihazın benzersiz bir adresi vardır, bu her cihazın bireysel olarak kontrol edilmesini sağlar. DALI'nın avantajları arasında esneklik, ölçeklenebilirlik, tek tek aydınlatmaların kontrol edilebilmesi ve kurulumun kolaylığı yer alır. DALI'nın bir dezavantajı ise diğer haberleşme protokollerine göre daha az desteklenmemesi ve mevcut sistemlere daha kolay entegre edilememesidir.

from pydali import DALI

# bus connection

dali = DALI(bus="COM3")

dali.send\_broadcast("[OnderGrup] Communication protocools send test code !")

dali.close()

DALI2 (Digital Addressable Lighting Interface 2): Aydınlatma kontrol sistemleri için kullanılan bir haberleşme protokolüdür. DALI protokolünün ikinci versiyonudur ve sahne kontrolü, renk sıcaklığı, hareket sensörleri ve diğer gelişmiş özellikleri içerecek şekilde genişletilmiştir.

DALI2, orijinal DALI protokolünün ilkelerine dayanır. İki telli bir bus kullanır ve her cihaz ağda benzersiz bir adrese sahiptir böylece her cihaz bireysel olarak kontrol edilebilir. Ancak, DALI2 aynı zamanda birden fazla ışığın grubu olarak kontrol edebilme, sahnelerin oluşturulması ve renk sıcaklığının ayarlanması gibi özellikleri de içerir. Ayrıca, DALI2, hareket sensörleri desteğini de içerir. Bu sensörler sayesinde, mekanların doluluk durumuna göre ışıklar otomatik olarak açılıp kapatılabilir.

DALI2 ayrıca "over the air firmware upgrade" adlı bir özellik içerir, bu özellik sayesinde cihazlar uzaktan güncellenebilir. Bu özellik cihazların güvenliğini ve performansını güncel tutmak için önemlidir.

DALI2 esnek, ölçeklenebilir ve kurulumu kolay bir aydınlatma kontrol protokolüdür ve ticari, endüstriyel ve yerleşim alanlarında geniş bir yelpazede kullanılabilecek gelişmiş özelliklere ve özelliklere sahiptir.

MQTT (Message Queuing Telemetry Transport): Makine-makine (M2M) ve İnternet of Things (IoT) cihazları için kullanılan bir haberleşme protokolüdür. Düşük hesaplama gücüne ve düşük ağ bant genişliğine sahip cihazlar için tasarlanmıştır. MQTT, yayın-abone modelini kullanır, cihazlar bir merkezi aracıya veri yayınlayabilir ve diğer cihazlar veriye abone olabilir. MQTT'nin avantajları arasında küçük kod yazılımı, düşük güç tüketimi ve düşük ağ bant genişliği gereksinimleri yer alır. Bir dezavantajı ise diğer haberleşme protokollerine göre daha az güvenli olması ve yüksek güvenlik gerektiren uygulamalar için uygun olmamasıdır.

import paho.mqtt.client as mqtt  
  
def on\_connect(client, userdata, flags, rc):  
 print("Connected with result code "+str(rc))  
 client.subscribe("onderGrup/communication")  
  
def on\_message(client, userdata, msg):  
 print(msg.topic+" "+str(msg.payload))  
  
client = mqtt.Client()  
client.on\_connect = on\_connect  
client.on\_message = on\_message  
  
client.connect("your\_mqtt\_server", 1883, 60)  
client.loop\_start()  
  
client.publish("onderGrup/communication", "[OnderGrup] Communication protocools send test code !")

OPC UA (Open Platform Communications Unified Architecture): endüstriyel otomasyon sistemleri için kullanılan bir haberleşme protokolüdür. Platform bağımsız olması için tasarlanmıştır ve çeşitli programlama dilleriyle kullanılabilir. OPC UA, bir istemci-sunucu mimarisini kullanır, cihazlar merkezi bir sunucuya bağlanarak ağdaki cihazları kontrol edebilir ve erişebilir. OPC UA'nın avantajları arasında platform bağımsızlığı, ölçeklenebilirlik ve gerçek zamanlı veri desteği yer alır. Bir dezavantajı ise uygulamak için karmaşık olması ve etkili bir şekilde çalışması için daha güçlü cihazlar gerektirmesi olabilir.

from opcua import Client  
  
client = Client("opc.tcp://your\_opcua\_server:4840")  
client.connect()  
  
root = client.get\_root\_node()  
  
var\_node = root.get\_child(["0:Objects", "2:MyDevice", "2:MyVariable"])  
  
var\_node.set\_value("[OnderGrup] Communication protocools send test code !")  
  
client.disconnect()

DDS (Data Distribution Service): Gerçek zamanlı veri haberleşmesi için kullanılan orta seviye bir protokoldür. Gömülü sistemler ve diğer düşük gecikme ve yüksek veri akışı gerektiren cihazlar için tasarlanmıştır. DDS, yayın-abone modelini kullanır, cihazlar bir merkezi aracıya veri yayınlayabilir ve diğer cihazlar veriye abone olabilir. DDS'nin avantajları arasında düşük gecikme, yüksek veri akışı ve gerçek zamanlı veri desteği yer alır. Bir dezavantajı ise uygulamak için karmaşık olması ve etkili bir şekilde çalışması için daha güçlü cihazlar gerektirmesi olabilir.

import OpenDDS.DCPS.transport as transport  
from OpenDDS.DCPS.DataWriter import DataWriter  
from OpenDDS.DCPS.TypeSupport import TypeSupport  
  
ts = TypeSupport()  
ts.register\_type(MyMessageType)  
  
dw = DataWriter()  
dw.create\_datawriter(ts, "MyTopic")  
  
msg = MyMessageType()  
msg.data = "[OnderGrup] Communication protocools send test code !"  
dw.write(msg)

NFC (Yakın Alan İletişimi): Cihazlar arasında 10 cm'nin altındaki mesafede veri alışverişi yapmaya olanak tanıyan kısa mesafeli bir kablosuz iletişim teknolojisidir. Temaslı ödemeler ve cihazlar arası veri aktarımı için yaygın olarak kullanılmaktadır. NFC'nin ana avantajı yüksek güvenlik seviyesidir, yakınlık gereksinimi nedeniyle yetkisiz cihazların veri aktarımına erişmesi zor hale gelir. Ancak, NFC'nin temel dezavantajı sınırlı mesafedir ve uzun mesafeli uygulamalar için uygun değildir.

#include <SPI.h>

#include <MFRC522.h>

#define SDA\_DIO 45

#define RESET\_DIO 47

#define buzzerPin 49

MFRC522 mfrc522(SDA\_DIO, RESET\_DIO);

void setup() {

Serial.begin(9600);

pinMode(buzzerPin, OUTPUT);

SPI.begin();

mfrc522.PCD\_Init();

Serial.println("Approximatethe card to the reader...");

Serial.println();

}

void loop() {

if ( ! mfrc522.PICC\_IsNewCardPresent()) {

return;

}

if ( ! mfrc522.PICC\_ReadCardSerial()) {

return;

}

String content= "";

Serial.print("UID tag :");

byte letter;

for (byte i = 0; i < mfrc522.uid.size; i++) {

Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");

Serial.print(mfrc522.uid.uidByte[i], HEX);

content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));

content.concat(String(mfrc522.uid.uidByte[i], HEX));

}

Serial.println();

Serial.print("Message : ");

content.toUpperCase();

if (content.substring(1) == "5D B6 00 31") {

Serial.println("Access granted :))");

} else {

Serial.println("Access denied");

digitalWrite(buzzerPin, HIGH);

delay(500);

digitalWrite(buzzerPin, LOW);

}

mfrc522.PICC\_HaltA();

mfrc522.PCD\_StopCrypto1();

}

Z-Wave: Ev otomasyonu ve akıllı ev uygulamaları için öncelikli olarak kullanılan bir kablosuz iletişim protokolüdür. 908.42 MHz frekansında çalışır ve 100 metre menziline kadar ulaşabilir. Z-Wave'in ana avantajlarından biri güvenilirliği ve düşük güç tüketimi olmasıdır, bu da pille çalışan cihazlar için uygundur. Ancak, relatif olarak düşük veri transfer hızı bazı uygulamalar için bir dezavantaj olabilir.

from pyzwave.pyzwave import ZWave  
  
zwave = ZWave("COM3")  
  
zwave.send\_broadcast("[OnderGrup] Communication protocools send test code !")  
  
zwave.close()

Thread: Ev otomasyonu ve akıllı ev uygulamaları için tasarlanmış bir kablosuz ağlama protokolüdür. IPv6 temel alınmıştır ve 2.4 GHz frekans aralığında çalışır. Thread'in ana avantajı, düşük güç tüketimi, güvenilirlik ve güvenlik sağlayan ve birçok cihazı destekleyebilen ağlar oluşturma yeteneğidir. Ancak, relatif olarak yeni bir teknolojidir ve diğer listedeki protokoller kadar yaygın olarak kabul edilmeyebilir.

Bluetooth Low Energy (BLE): Düşük güç tüketimi olan cihazlar için tasarlanmış bir kablosuz iletişim protokolüdür. 2.4 GHz frekans aralığında çalışır ve 100 metre menziline kadar ulaşabilir. BLE'nin ana avantajlarından biri düşük güç tüketimi olmasıdır, bu da uzun dönemler boyunca pillerle çalışması gereken cihazlar için uygun hale gelir. Ancak, relatif olarak düşük veri transfer hızı bazı uygulamalar için bir dezavantaj olabilir.

import pygatt  
  
adapter = pygatt.GATTToolBackend()  
adapter.start()  
device = adapter.connect("XX:XX:XX:XX:XX:XX")  
  
device.char\_write("XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX", "[OnderGrup] Communication protocools send test code !")  
  
adapter.stop()

Zigbee: Ev otomasyonu ve akıllı ev uygulamaları için tasarlanmış bir kablosuz iletişim protokolüdür. 2.4 GHz frekans aralığında çalışır ve 100 metre menziline kadar ulaşabilir. Zigbee'nin ana avantajlarından biri düşük güç tüketimi olmasıdır, bu da uzun dönemler boyunca pillerle çalışması gereken cihazlar için uygun hale gelir. Ancak, relatif olarak düşük veri transfer hızı bazı uygulamalar için bir dezavantaj olabilir.

from xbee import ZigBee  
import serial  
  
ser = serial.Serial('COM3', 9600)  
  
xbee = ZigBee(ser)  
  
xbee.send("tx", dest\_addr\_long = "XXXXXXXXXXXXXXXX", data = "[OnderGrup] Communication protocools send test code !")  
  
ser.close()

6LoWPAN: İnternet of Things (IoT) uygulamaları için tasarlanmış bir kablosuz iletişim protokolüdür. IPv6 temel alınmıştır ve 2.4 GHz frekans aralığında çalışır. 6LoWPAN'ın ana avantajlarından biri, düşük güç tüketimi, güvenilirlik ve güvenlik sağlayan ve birçok cihazı destekleyebilen ağlar oluşturma yeteneğidir. Ancak, relatif olarak yeni bir teknolojidir ve diğer listedeki protokoller kadar yaygın olarak kabul edilmeyebilir.

import py6lowpan  
  
lowpan = py6lowpan.SixLowpan()  
  
lowpan.send("[OnderGrup] Communication protocools send test code !")  
  
lowpan.close()

Wi-Fi: İnternet erişimi ve yerel alan ağları (LAN) için yaygın olarak kullanılan bir kablosuz iletişim protokolüdür. 2.4 GHz ve 5 GHz frekans aralıklarında çalışır ve 100 metre menziline kadar ulaşabilir. Wi-Fi'nin ana avantajlarından biri yüksek veri transfer hızlarıdır, bu da hızlı bir şekilde büyük miktarda veri aktarılması gereken uygulamalar için uygun hale gelir. Ancak, relatif olarak yüksek güç tüketimi bazı uygulamalar için bir dezavantaj olabilir.

import socket  
  
s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
s.connect(("wifi\_server", 80))  
  
message = "[OnderGrup] Communication protocools send test code !"  
s.send(message.encode())  
  
s.close()

Sigfox: İnternet of Things (IoT) uygulamaları için tasarlanmış bir düşük güç, geniş alan ağı (LPWAN) protokolüdür. 868 MHz frekans aralığında çalışır ve 15 km menziline kadar ulaşabilir. Sigfox'un ana avantajı uzun menzilli özellikleri ve düşük güç tüketimi olmasıdır, bu da uzak mesafelerde iletişim kurması gereken cihazlar için uygun hale gelir. Ancak, relatif olarak düşük veri transfer hızı bazı uygulamalar için bir dezavantaj olabilir.

Lora: Diğer LPWAN protokollerinden farklı olarak, daha yüksek veri transfer hızlarına sahip olabilen bir kablosuz iletişim protokolüdür. 868 MHz ve 915 MHz frekans aralıklarında çalışır ve 15 km menziline kadar ulaşabilir. Lora'nın ana avantajları arasında uzun mesafe yetenekleri, yüksek veri transfer hızları ve düşük güç tüketimi yer almaktadır. Ancak, geniş kapsama alanına sahip olması nedeniyle, bazı bölgelerde frekans kirliliği sorunları oluşabilir.

NB-IoT (Dar Bant IoT): İnternet of Things (IoT) uygulamaları için tasarlanmış bir düşük güç, geniş alan ağı (LPWAN) protokolüdür. Lisanslı hücresel frekanslarda çalışır ve 35 km menziline kadar ulaşabilir. NB-IoT'nin ana avantajı, birçok cihazı destekleyebilme ve uzak bölgelerde kapsama sağlama yeteneğidir. Ancak, relatif olarak yeni bir teknolojidir ve diğer listedeki protokoller kadar yaygın olarak kabul edilmeyebilir.

LTE-M (Makinelere Yönelik Uzun Dönemli Evrim): İnternet of Things (IoT) uygulamaları için tasarlanmış bir hücresel iletişim protokolüdür. Lisanslı hücresel frekanslarda çalışır ve 35 km menziline kadar ulaşabilir. LTE-M'nin ana avantajı, birçok cihazı destekleyebilme ve yüksek hızlı veri transferi sağlama yeteneğidir. Ancak, diğer LPWAN protokollerine göre güç verimliliği yüksek olmayabilir.

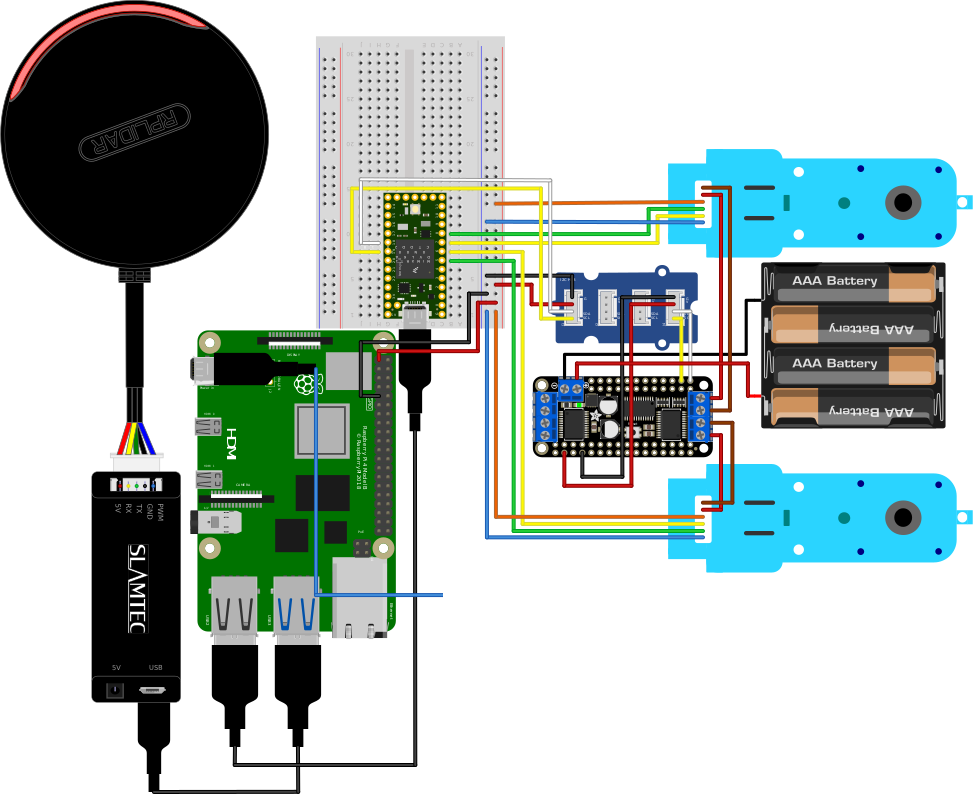
4G: internet erişimi ve mobil iletişim için yaygın olarak kullanılan bir hücresel iletişim protokolüdür. Lisanslı hücresel frekanslarda çalışır ve 35 km menziline kadar ulaşabilir. 4G'nin ana avantajı yüksek veri transfer hızları ve birçok cihazı destekleyebilme yeteneğidir. Ancak, diğer LPWAN protokollerine göre güç verimliliği yüksek olmayabilir ve tüm IoT uygulamaları için uygun olmayabilir.

Aşağıdaki tablo bize her bir iletişim protokolü arasındaki belirli farkları vermektedir. Bu farklar sırasıyla Veri hızı, Menzil, Güvenlik, Güç tüketimi, Başlık boyutu, Gecikme ve Frekans’tır. Tüm bu değerleri tek tablo üzerinden kontrol edip projeye uygun iletişim protokolü seçmek gayet mümkündür.

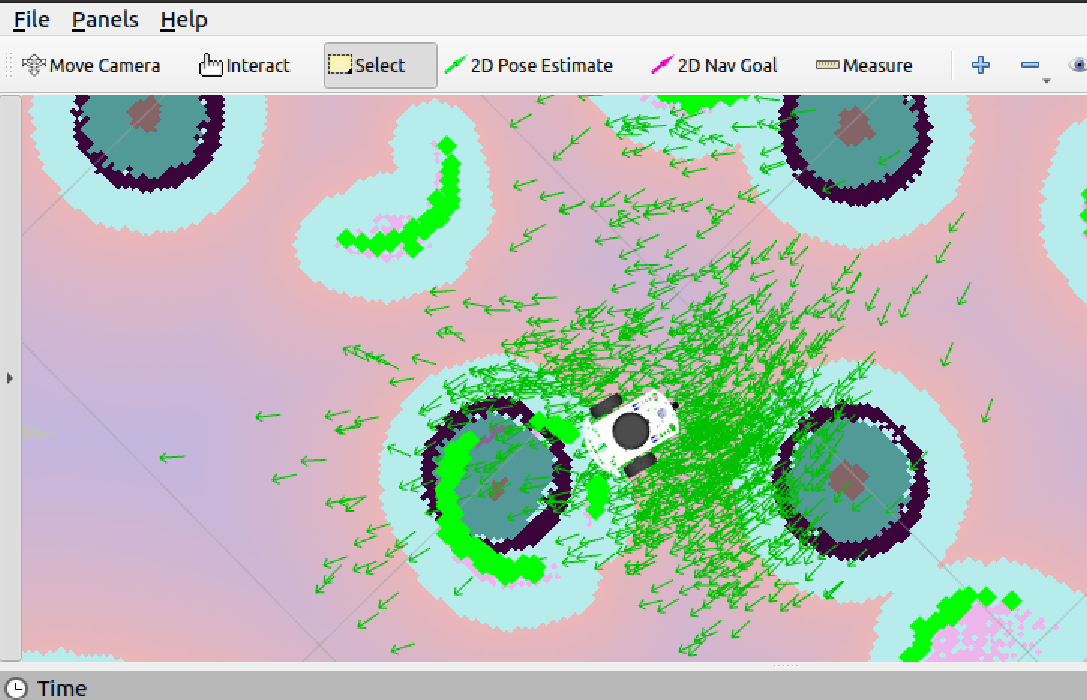
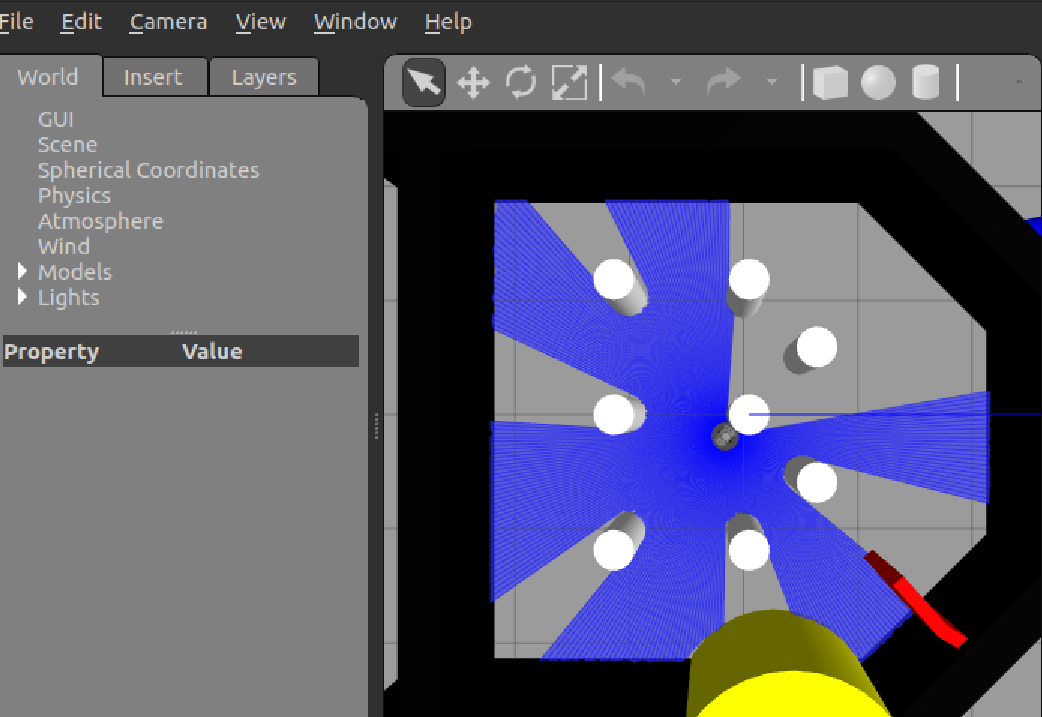
Table

Description automatically generated

**Örnek Proje Devre Şeması**



**Örnek Proje Çıktıları**



**Kaynakça**

* <https://github.com/ros-mobile-robots/diffbot>
* <https://data-flair.training/blogs/iot-technology/>
* <http://wiki.ros.org/>
* <https://www.aselsan.com.tr/tr/inovasyon/haber-detay/ros-nedir-gunumuzden-ornekler-ve-kullanim-alanlari-3423>
* <https://mqtt.org/mqtt-specification/>
* https://www.pubnub.com/blog/what-is-mqtt-use-cases/