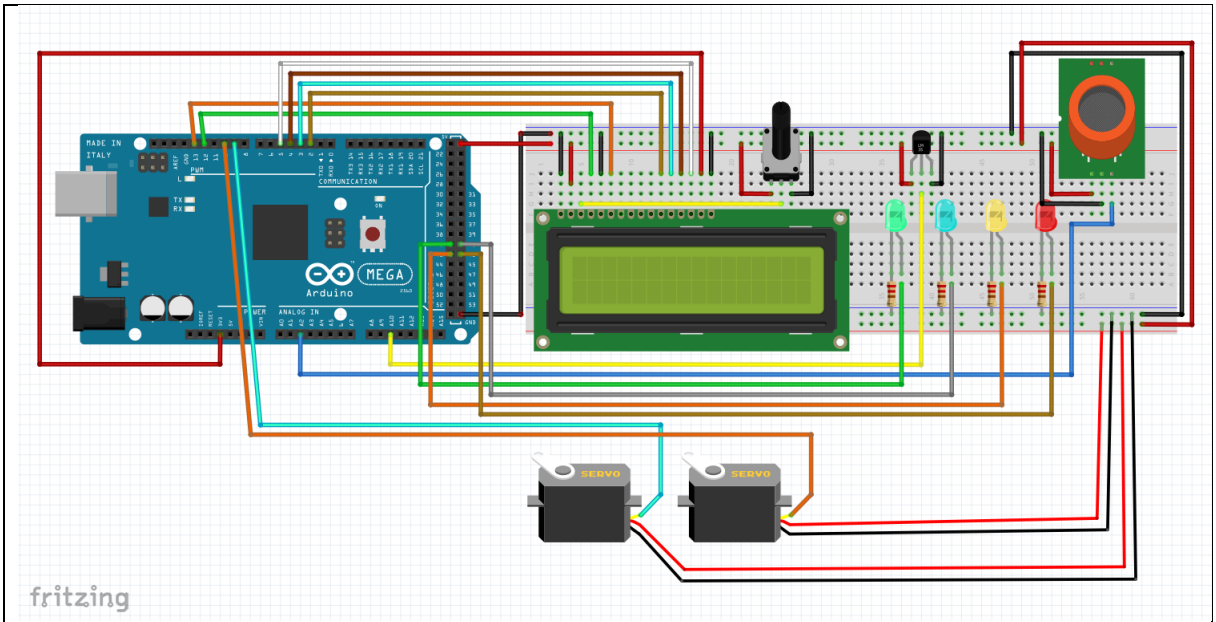
Derste yapılan örneğe bir adet sensör ekleyerek farklı bir görev ile kontrol ediniz.

Soru1. Derste yapılan örnek ile birlikte devre şemasını bağlantı noktalarını göstererek çiziniz. Tinkercad çizimleri kabul edilir. İstediğiniz şekilde çizebilirsiniz.



Devrenin Videosu: https://youtu.be/jYRXmQKLBk4
FreeRTOS Kodu: Google Drive
Yapısal Programlama Kodu: Google Drive

Soru 2. FreeRTOS ile gerçekleştirilen Arduino programının çalışmasını detaylı bir şekilde anlatınız. Sensör eklediğinizde farklı durumla karşılaştınız mı? Açıklayınız.

Arduino'ya güç verildiğinde;
Ledlerin sırayla yanması için led görevlerine ardışık 1, 2, 3 ve 4 saniye task delay verilir, sonra 3 saniyede bir ledler 1 saniye yakılıp söndürülür.
LM35'den analog değer alınır ve sıcaklık değerine dönüştürülür. Sıcaklık değeri xQueue_Sicaklik kuyruğuna bekletilmeden gönderilir ve 1sn sonra başa döner. Bu görev ilk çalıştırıldığında 1sn task delay alır.
Gaz sensöründen analog değer alınıp xQueue_Gaz kuyruğuna bekletilmeden gönderilir ve 1sn sonra başa döner. Bu görev ilk çalıştırıldığında 1sn task delay alır.
LCD ekranda yazılı değerler silinir. İlk if bloğunda xQueue_Sicaklik kuyruğuna değer geldiği gibi alır, ilk satıra "Sicaklik:" ve değerini yazar. İkinci if bloğunda xQueue_Gaz kuyruğuna değer geldiği gibi alır, ikinci satıra analog gaz değeri 550'den büyükse "Gaz Var" küçükse "Gaz Yok" yazar. 1sn bekler ve başa döner.
Öncelik sıralamasını görevlere ardışık dağıttığımda bazı görevler çalışmadı. Bazı görevlere aynı öncelik değeri vererek sorun çözüldü. LM35 sıcaklık sensörü stabil değerler vermiyordu 1sn task delay ile sorun çözüldü. Gaz sensörü kuyruğa çok değer gönderdiği için kuyruk doluyordu ve değer gönderemiyordu 1sn task delay ile sorun çözüldü.

Soru3. Eğer derste yapmış olduğumuz örneği yapısal programlama ile yapılmış olsaydı ne tür avantaj ve dezavantajlar sağlardı?

Yapısal programlamada bir görev bitene kadar diğer görevler beklemek zorun kalırdı. Uzun bekleme süresi olan görev varsa FreeRTOS kullanmak daha mantıdır. Servo 2. Konuma gelmesi için 1sn daha delay istiyor, bu yüzden led söndüğü gibi diğer ledi yakamıyor ve lcd ekrandaki değerler 1sn daha geç geliyor.

Soru4. Yapısal programlama ile FreeRTOS çalışmasındaki çevre birimlerin (LEDler, LCD, sensörler) okuma ve yazmaları arasındaki gecikmelerini bulunuz veya hesaplayınız.

Örneğin; Sıcaklık sensörünün ölçüm değerlerinin her okuması FreeRTOS kullanıldığında 10 mikrosaniye gecikme oluşmuştur. Yapısal programlama ile gerçekleştirildiğinde 20 mikrosaniye gecikme oluşmuştur.

Led1 yanma gecikmesi: freeRTOS-340ms yapısal programlama-40ms
Led2 yanma gecikmesi: freeRTOS-334ms yapısal programlama-36ms
Led3 yanma gecikmesi: freeRTOS-337ms yapısal programlama-36ms
Led4 yanma gecikmesi: freeRTOS-341ms yapısal programlama-48ms
Servonun gecikmesi: freeRTOS-154ms yapısal programlama-32ms
LM35'den sıcaklığın gönderilmesi: freeRTOS-87ms yapısal programlama-26ms
Gaz sensörü değerinin gönderinin gönderilmesi: freeRTOS-73ms yapısal programlama-32ms
LCD'nin değerleri alıp yazdırma gecikmesi: freeRTOS-62ms yapısal programlama-1ms
freeRTOS yada yapısal programlamanın üstüne tıklayarak hesaplamayı görebilirsiniz.

Soru5. FreeRTOS ile ilgili genel bir değerlendirme yapınız. En az 20 cümle içerecek)

FreeRTOS'da işlemin ne zaman başlayıp ne zaman biteceği bellidir. Zamanın önemli olduğu sistemlerde kullanılır. Kullanıcı gözünde bütün görevler aynı anda çalışıyor gibi görülür. Tek bir çekirdek aynı zamanda tek bir görev çalıştırabilir. Sistem bütün görevleri belli kısa zaman aralıklarıyla çalıştırır. Uzun süreli çalıştırmalarda zaman kaymaları olabilir. Kritik hayati projelerde sistem düzenli olarak resetlenmelidir. Görevlerin öncelikleri vardır. Görev akışında daha yüksek önceliğe sahip görev gelirse işlemci ilk önce o göreve yönelir. Görevlerin öncelik değerleri dikkatli verilmezse bazı görevler çalışmayabilir. Gerçek zamanlı sistemler kaynak kullanımı en iyi yöneten sistemlerdir. Kullanıcıyı hayati etkileyebilecek sistemlerde FreeRTOS kullanılmalıdır. Çünkü, yapısal programlamada bir görev bitmeden diğer göreve geçmez. Gelen girdiye FreeRTOS daha hızlı tepki gösterir. Düşük kapasiteli işlemcilerde çalışabildiği için maliyeti azdır. Multitasking özelliğiyle çoğu IoT uygulamalarında kullanılır. FreeRTOS'da bir görevden gelen bilgi diğer göreve anında gönderilebilir. Görevler arası haberleşme queue kuyruk yapısıyla olur. FreeRTOS'da klasik delay fonksiyonu kullanılırsa bütün sistemi etkiler. Etkilenmemesi için vTaskDelay fonksiyonu kullanılmalıdır klasik delay fonksiyonu ise kullanılmamalıdır.