T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNELERİN İNTERNETİ PROJE RAPORU

Internet of Things – Kameralı Araba Projesi

IoT

B211210031 - Eren KARA - 1/A

SAKARYA

Aralık, 2023

Nesnelerin İnterneti 1-A

İçindekiler:

1. Problemin Tanımı ve Çözüm Yöntemleri

Kargoların veya herhangi teslimatların daha hızlı, daha ucuz ve sürekli olmaması.

Bunun için maliyeti bir eleman çalıştırmaktan daha ucuz olan ve koskoca arabaları bu iş için yormaktan daha mantıklı olan küçük küçük çok sayıda kargo arabası kullanmak.

Kargo taşıma işinin günün her saati yapılabilmesi ve kullanıcı memnuniyetinin sağlanabilmesi için geliştirmiş olduğum bu araç ister uzaktan kontrol ile iste arabanın daha çok geliştirilerek sağlanabilecek olan kontrolün yapay zekaya bırakılması ile kargoların bir tuşa basarak teslim edilmesini sağlamayı hedefliyor.

2. Kullanılan Teknolojiler

Arabayı tasarlamak için 4 adet DC motor, bir adet esp32-cam kamera modülü, 1 adet L298N motor sürücü kartı kullandım. Bu aletleri beslemek için iki adet 8'li pil kullandım ve bu pilleri parallel bağladım, böylece gerekli olan akım miktarını elde etmiş oldum.

Esp32-cam modülünü kodlayabilmek için Arduino kullandım.

Esp32-cam modülünün çekmiş olduğu görüntüleri kullanıcıya aktarabilmek için bir tane web sitesi oluşturdum. Bu web sitesi aracılığı ile ve web socket kullanarak kullanıcıya çekmiş olduğum fotoğrafları yolladım. Ayrıca yine web socket kullanarak kullanıcıdan gitmek istediği yön bilgisi alarak motorları kontrol ettim.

Site üzerinden istenilen bir wifi'a bağlanılabilmesi için AsyncWebServer kütüphanesi aracılığı ile request karşıladım ve wifi'a bağlandım.

Kameranın çekmiş olduğu fotoğraflardan her 60 tanede bir tanesini Firebase cloud real time database kullanarak depoladım.

3. Tasarım ve Gerçekleme

Öncelikle koda ulaşmak istiyorsanız : https://github.com/EreenKara/Esp32camCar

Kafamda arabayı kurgularken kamera ile yönümüzü tayin edebildiğimiz oturduğumuz yerden kontrol ettiğimiz bir araba olmasını istemiştim ve yaparkende bunu hedefledim.

Bunun için gereken şeyin bir web sitesi ya da mobil uygulama olacağına karar verdim ancak ben bilgisayardan da kullanmak istediğim için mobil seçeneği kolaylıkla elenmiş oldu. Bu karardan sonra gereken kütüphaneleri araştırmaya başladım. RNT (Random Nerd Tutorial) kanalında bir çok şey ile ilgili hazır yapılmış kodlar ve anlatımlar halihazırda mevcuttu. Gereken

tek şey okuma yapmaktı. Kodları direkt olarak kopyalamaktan daha ziyade gerçek bir okuma yapıp kendime göre şekillendirip, optimizasyon yapmak istedim. RNT'den yolumu tayin edince bir sürü kütüphanenin github reposu çıktı karşıma ve okuma yapmaya başladım.

ESPAsyncWebServer bu kütüphane bize asenkron bir şekilde çalışan web server'ı için API sağlıyordu. Klasik bir request-response ilişkisine dayalı web server kütüphanesi olduğundan öğrenmem oldukça kısa sürdü. İlk önce hangi porttan iletişim kuracağını belirtiyoruz ki bu http olduğundan 80 portu ile başlatacağız demek oluyor. Ondan sonra ise url'lerden gelen verileri nasıl karşılaması gerektiğini belirliyoruz. Zaten bir tane anasayfa [handleRoot()] için bir tane de internet bağlantısını değiştirmek için iki ayrı route hazırladıktan sonra geriye kalan bunların içini doldurmaktı. Anasayfa olan için hemen bir websitesi tasarladım. Ancak bir çok hata ile karşılaştım ve istediğim gibi bir çıktı alamayınca internet üzerinden başka bir tane hazır web sitesi kullanmak zorunda kaldım. İkinci olarak wifi değiştirme [reconnectAnotherWifi()] işlemini yapmak kaldı. Onu da WiFi kütüphanesi ile kolaylıkla yapabildim. Wifi disconnect ve wifi begin demem yeterliydi.

Ana şablonu hazırladıktan sonra geriye kalan kullanıcı ile sürekli bir haberleşme içerisinde kalabilmekti. Bunun için ise Websocket kullanmak mantıklı geldi. Bunun etkili olmasındaki sebepler: haberleşmek için http portunu kullanıyor olması. Bir kez bağlantı kurulduktan sora bir daha bağlantı için istek atmıyor oluşumuz ve internet üzerinde yapılmış birçok örneğinin oluşu. Websocket için kaynakça 15'te belirttiğim github kütüphane linkinden ve RNT adresinden yararlandım. Kodda hemen Websocket nesnelerini hazırladım hangi url (route) üzerinden haberleşmelerini istediğimi belirttim ve geriye gelen verileri karşılamak ve veri göndermek kalıyordu. Veri karşılama işlemini github'da belirtildiği gibi yaptım. Gelen veriyi karşıladım[onCarInputWebSocketEvent()] ve ',' e göre ayırma işlemi yaptım. Ondan sonra gelen key'e göre ilgili fonksiyona yönlendirdim. Fonksiyonlardan motor ile ilgili olan için önceden gerekli pinlere output işlemi atadım[setupPinModes()].

Sıra geldi kamera işlemlerini halletmeye . Bu kısmı yapmak asıl zamanımı alan kısım diyebilirim. Kamerayı ilk önce başlatmak gerekiyordu. Başlatma işlemine biraz kaynakça 14'te bahsettiğim github linkinden biraz ise RNT üzerinden baktım. Çünkü bendeki model için gerekli konfigurasyonlar değişmiş ve dokümantasyonu bunları belirtmiyor ya da ben böyle anladım. Kamera konfigürasyonlarını [cameraConfiguration()] ile yaptım. Websocket'e yollama işleminde ise camera'nın bize sağlamış olduğu buffer'dan verileri binary bir şekilde web kullanıcısına aktardım. Ancak aktarırken bu binary veriler üzerinde herhangi başka bir işlem yapmaya kaltığımda sürekli null pointer hatası aldım. Bu işlemlerde ısrarcı olduğumdan - ki bu işlemler asenkron olarak firebase sunucusuna veri aktarmamda gerekliydi – asıl zamanımı çalan kısım burası oldu. İlk önce kameranın bize sağlamış olduğu buffer'ın ne tip veri içerdiğini bulmam kaynak ve dokümantasyonlardaki eksiklikten dolayı çok uzun zamanımı aldı. Binary array veri tipini içerdiğini öğrendikten sonra bile kopyalama yapmak istediğimde veya aynı anda thread çalıştırdığımda bu buffer üzerinde sürekli hatalar aldım. Peki neden asenkron işlem gerekliydi.

Asenkron işlemin gerekli olmasının sebebi firebase sunucusuna veri aktarmanın web servera veri aktarmaktan 20 kat daha uzun sürmesi ve bu süre zarfında web serverdaki stream'in takılması, donmasıydı. Bunun önüne geçmek için kaynakça 13'te bahsettiğim ReactESP kütüphanesini kullandım. Ancak bu kütüphane gördüğüm en amatör kütüphanelerden biri olduğundan ve bu asenkron işlemini Observer deseni ile yaptığından ve loop içerisinde sürekli notify ettiğimizden istediğim sonucu alamadım ve bu kütüphane zamanımı çalmaktan öteye gidemedi.

Asenkron denemekten vazgeçmedim ve ikinci bir yol olarak linux thread'lerini (pthread.h) denedim. Bu sefer çalışacağından emin olmama rağmen eş zamanlı erişmeye kalktığım her seferinde bir hata ile karşılaştım. Sorunun eş zamanlılıktan mı yoksa başka bir şeyden mi kaynaklandığını bulamadım. Bu da olmayınca mecburen asenkrondan vazgeçmem gerekti.

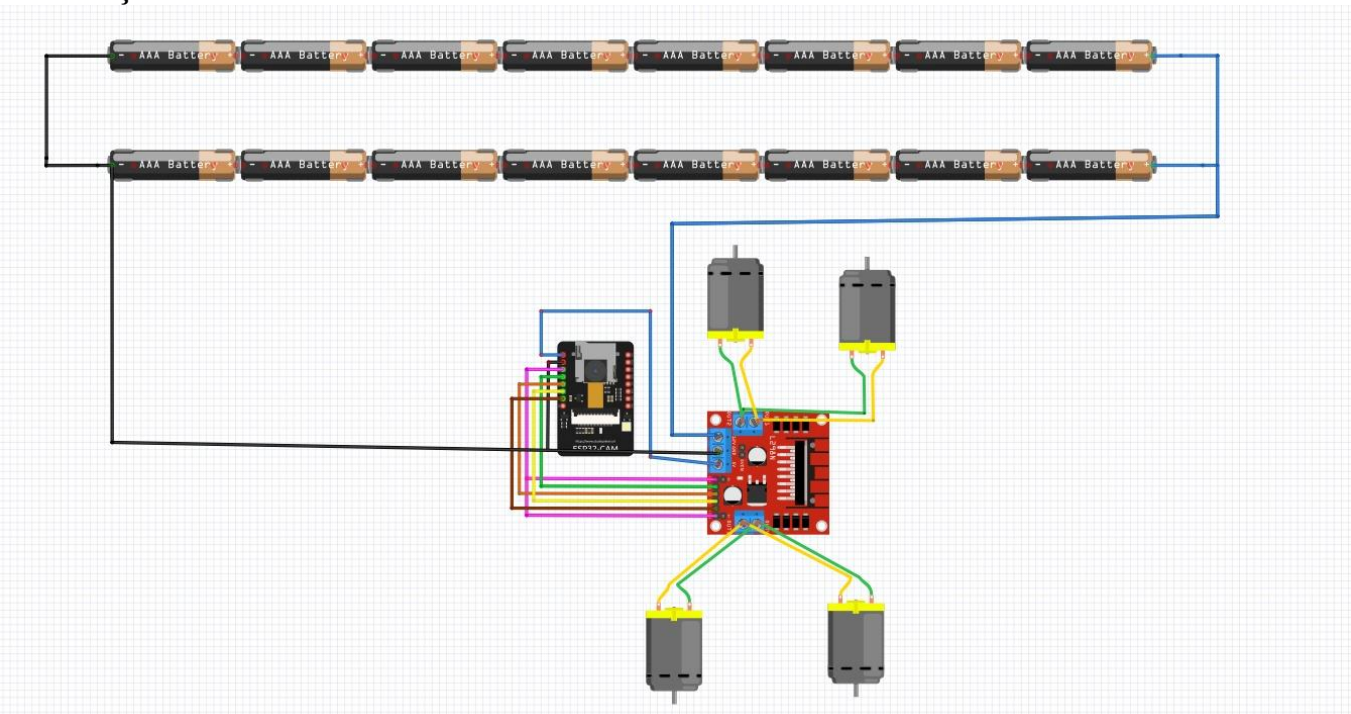
Kamera görüntüsünün yavaşlamasını istemediğimden her bir fotoğrafı firebase'e atmaktan vazgeçtim ve 60 tane fotoğrafı web sunucusuna attıktan sonra 1 tanesini de firebase'e atmaya karar verdim. Bu işlemleri ise web [loop()] içerisinde gerçekledim.

Firebase kısmını ise kaynakça 12'de bahsettiğim Firebase_ESP_Client.h kütüphanesini kullanarak yaptım. Ayarlamaları yani autherization kısmını kaynakça 18'deki videodan yardım alarak yaptım.

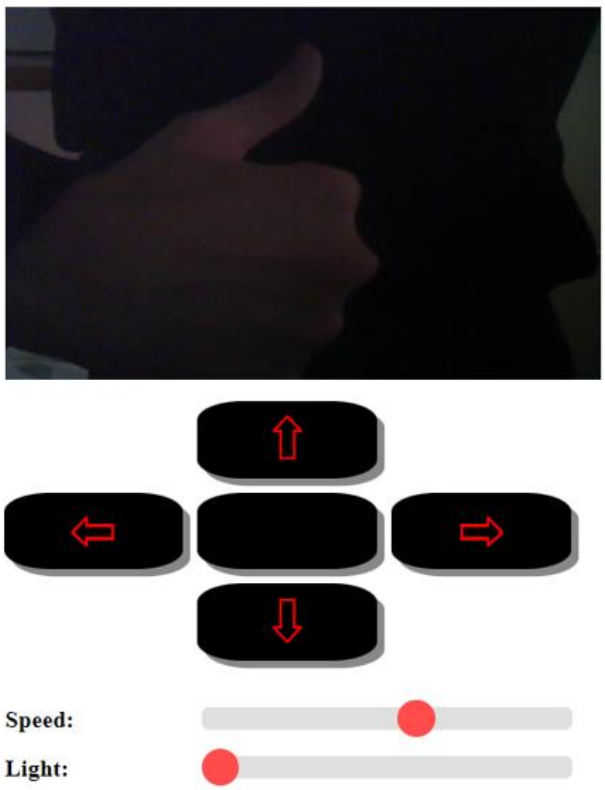
4. Maliyet Analizi

Ürün İsmi	Ürün Fiyati
Rex Chassis Serisi 4WD	272 TL
L298N Motor Sürücü Kartı	70 TL
ESP32-CAM + OV2640	300 TL
16 Adet Pil	80 TL
2 Adet Pil Yatağı	60 TL
Bant, vida vb.	80 TL
1 Adet Switch	6 TL
Toplam	868 TL

5. Devre Şeması



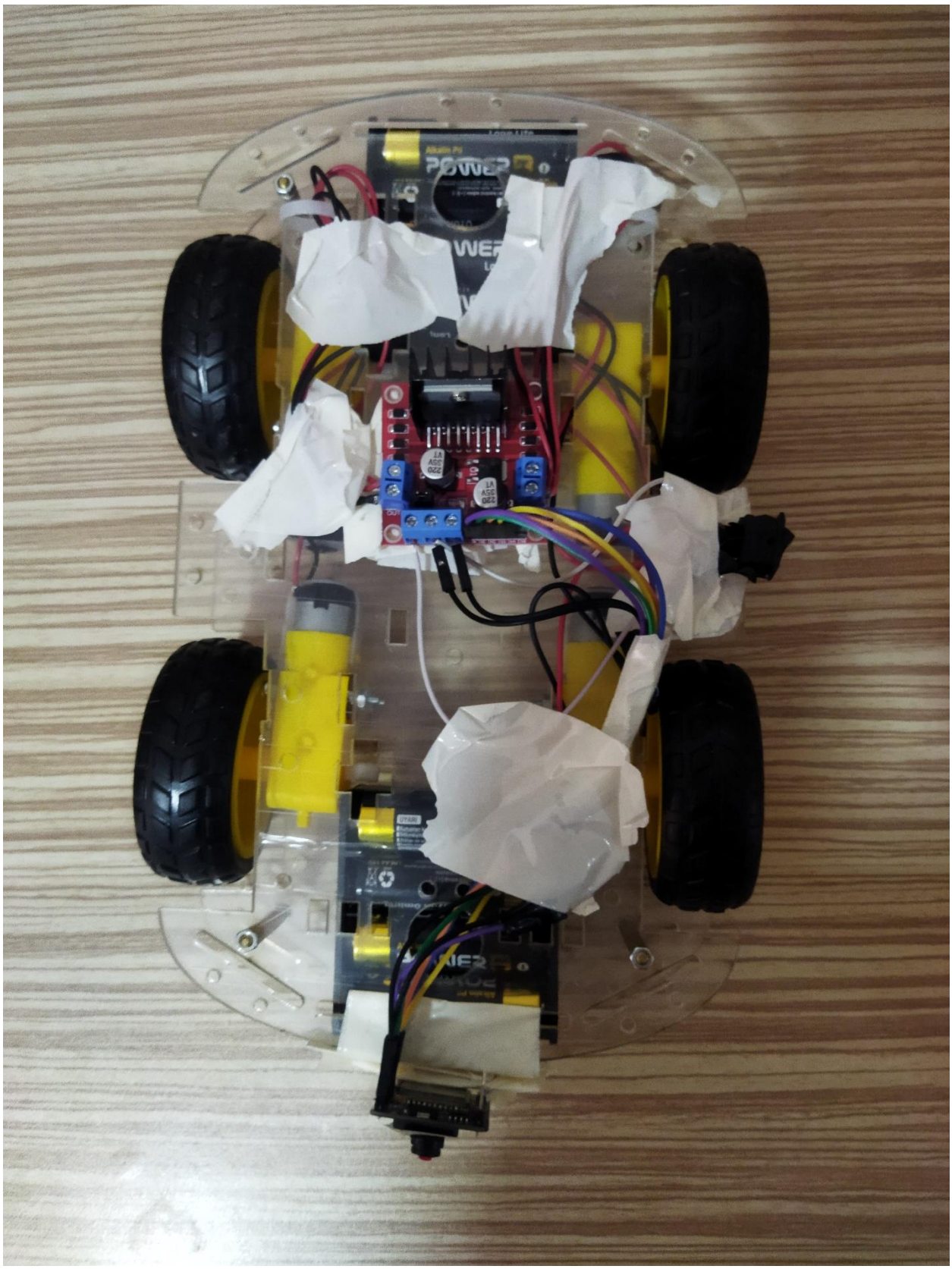
6. Uygulamaya Ait Görseller



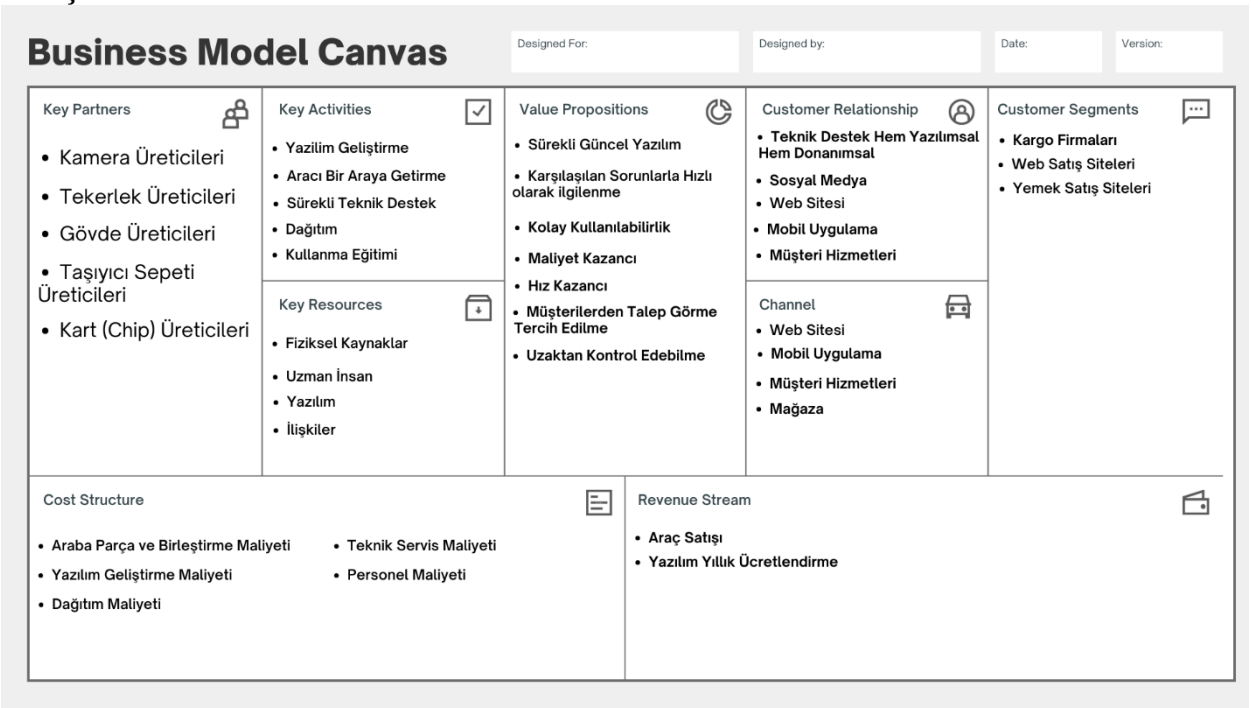
Wifi: Password: Baglan

Firebase Realtime Database

Fotograf: "blob,base64,/9j/4AAQSkZJRgABAQEAAAAAAAD/2wBDAAoHCAkIBgoJCAkLCwoMDxkQDw4ODx8WFxIZJCAmJiQgIyIoLToxKCs2KyIjMkQzNjs9QEFAJzBHTEY/S



7. İş Modeli



8. Big Data

Gönderdiğim fotoğraf verileri ve ileride gps modülü ilave edilerek elde edilebilecek aracın bulunduğu konum verisi ile yol durumu analizi gerçekleştirilebilir. Yolda herhangi bir göçük, engel, yıpranma var mı diye. Ayrıca otonom araçlar için aracın hareketleri ve yol verisi eşleştirilerek otonom araçların veri setine katkıda bulunabilinir.

Kullanıcı profili analizi gerçekleştirilebilir. O an o kargoyu teslim alan insanın özellikleri ve yaşadığı çevre değerlendirilerek kullanıcı profili çıkartılabilir.

Bir olay mahalinde anlık olarak o noktada bulunan bir aracın veya daha önce o yoldan geçmiş olan bir aracın kamera görüntüleri güvenlik amacı ile kullanılabilir.

9. Kaynakça

1. https://randomnerdtutorials.com/esp32-async-web-server-espasyncwebserver-library/
2. https://randomnerdtutorials.com/esp32-wi-fi-manager-asyncwebserver/
3. https://randomnerdtutorials.com/esp32-websocket-server-sensor/
4. https://randomnerdtutorials.com/esp32-cam-save-picture-firebase-storage/
5. https://randomnerdtutorials.com/esp32-cam-display-pictures-firebase-web-app/
6. https://randomnerdtutorials.com/esp8266-nodemcu-http-get-post-arduino/
7. https://randomnerdtutorials.com/esp8266-nodemcu-write-data-littlefs-arduino/
8. https://randomnerdtutorials.com/esp8266-adc-reading-analog-values-with-nodemcu/
9. https://randomnerdtutorials.com/program-upload-code-esp32-cam/
10. https://randomnerdtutorials.com/sending-data-from-an-arduino-to-the-esp8266-via-serial/
11. https://github.com/un0038998/CameraCar/blob/main/Camera_Car/Camera_Car.ino
12. https://github.com/mobizt/Firebase-ESP-Client
13. https://github.com/mairas/ReactESP/tree/master
14. https://github.com/espressif/esp32-camera
15. https://github.com/me-no-dev/ESPAsyncWebServer

16. https://www.arduino.cc/reference/en/language/variables/utilities/progmem/
17. https://arduino-esp8266.readthedocs.io/en/latest/PROGMEM.html
18. https://www.youtube.com/watch?v=aO92B-K4TnQ
19. https://forum.arduino.cc/t/read-the-data-buffer-of-esp32-cam-as-binary/917641/11
20. https://www.canva.com/
21. https://www.esp32.com/viewtopic.php?t=19135
22. https://esp32.com/viewtopic.php?t=20215
23. https://www.w3schools.com/jsref/event_onmousedown.asp
24. https://forum.arduino.cc/t/esp8266-disconnect-from-router/529495/6
25. https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/
26. https://codepen.io/kaypooma/pen/mdyyoK
27. https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/ledc.html
28. https://www.quora.com/How-do-I-send-the-data-from-Tx-to-Rx-How-does-serial-monitor-control-the-data
29. https://github.com/Emrehanoglu/Esp32Cam-Firebase/blob/main/Esp32Cam-Firebase/Esp32Cam-Firebase.ino
30. https://www.youtube.com/playlist?list=PLSuhOGv534vRjqidiUiCHSEhbbX3ZZD-p
31. https://www.youtube.com/watch?v=HfQ7lhhgDOk