Öncelikle biraz zaman geçtiği için bazı detayları tam hatırlayamayabilirim.

Bu belgeyi %100 doğru kabul etmeyin.

**Genel açıklama:**

1) Bu kodun yaptığı iş, belirtilen sayıda resmi teker teker aralıklarla veya burstler olarak çekip içindeki SD karta yazmak. Aslında RAM daha geniş olsaydı SD kart gerekmeyebilirdi ama tek seferde çekip RAMde tutulabilen fotoğraf sayısı çok az (sanırım 3 veya 4 jpgdi ama denemek lazım). Resimlerle yapılmak istenen işler sonrasında SD karttan okuyarak yapılabilir.

2) Bu kamerada neden olduğunu anlamam uzun süren bir sorun vardı. Cihaz ilk açıldığında hemen çektiği ilk resim koyu yeşil çıkıyordu.



Bunun sebebi olarak internette de bulduğum sınırlı şeyden yaptığım çıkarım, sanırım kameranın ya otomatik ışık ayarı ya da otomatik renk ayarının bitmesine zaman verilmemesinden kaynaklandığı. Bunu düzgün şekilde çözmek çok uzadığı için başta birkaç tane kaydetmeden resim çekip kameranın kendine gelmesini beklemek benim bulduğum en kolay çözümdü.



3) İlk başta bütün gereken setup / config işleri yapılıyor (paramatreler, sensör vs, kamera testi, sd testi). Sonra yukarıda bahsettiğim boş resimleri çekiyor. Sonra mainloop içinde resim çekip, başarılı olduysa kaydedip “pic\_batch\_count” tane çekene kadar devam ediyor. Serial üzerinden de debuglamak için işe yarayabilecek bazı bilgiler gönderiyor.

**NOT 1:**

* Bu libraryler ve pin def'leri internetten bu boardın örnek kodlarından buldum. (dikkatimi SD kart için bir libary olmaması çekiyor ve arada kaybolmuş olabilir, veya başka bir librarynin içinde geçiyor da olabilir)
* Her birisinin tek tek ne işe yaradığını %100 bilmiyorum ama hatırladığım kadarıyla bazılarını çıkarınca sıkıntı yaratıyordu.
* "brownout": supply voltage drop

**NOT 2:**

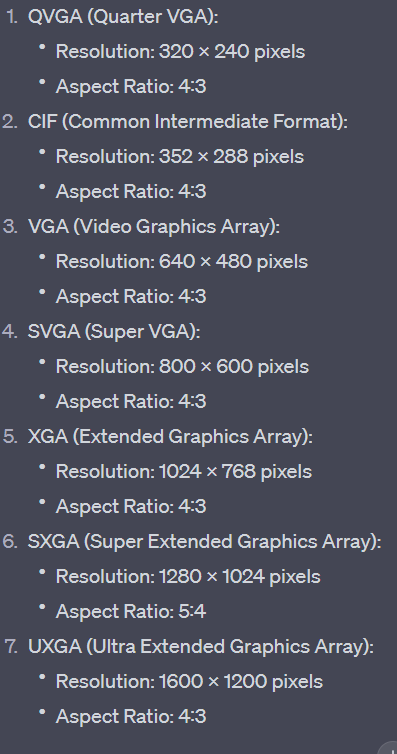
* burst\_size: SD’ye kaydetmeden kaçar tane resim çekileceği
* pic\_batch\_count: Toplam kaç resim çekileceği
* pictureNumber: SD’ye kaydederken resmin ismine verilecek sayı, her burstte burst\_size kadar arttırılıyor.
* time\_capture/ time\_capture\_temp: her burstte resimlerin çekilmesinin kaç ms sürdüğünü tutmak için
* time\_write/ time\_write\_temp: her burstte resimlerin SD’ye kaydedilmesinin kaç ms sürdüğünü tutmak için
* init\_time: uptime hesaplamak için
* error\_flag: SD write mode ile açılamazsa programı durdurmak için
* burst\_index: kaçıncı burst olduğunu tutmak için
* temp2: istenirse sensör parametrelerini bir runtime boyunca değiştirmek için.
* config: kamera librarysinin kullandığı, kameranın configi için kullanılan bir struct.
* s: sensör librarysinin kullandığı, sensörün configi için kullanılan bir struct.

**NOT 3:**

* Bu fonksiyon ESP’nin bütün RAM usagelarını görmek için

**NOT 4:**

* Bunları da internetten buldum, ama camera librarysine bakarak bunların ne işe yaradığı anlaşılabiliyor.
* Birkaç değiştirilebilecek parametre:
  + pixel\_format: Çekilen resmin nasıl saklanacağı için (raw png olarak da tutulabiliyor ama kalitede kayda değer bir farklılık yokken resim boyutu korkunç yükseliyor)
  + jpeg\_quality: Sanırım boyut-kalite konusunda optimum 4, ama kurcalamayı öneririm yine de.
  + frame\_size: çekilecek resmin boyutları (pixel olarak)



* + fb\_count: “fb” frame buffer anlamına geliyor, resimleri çekince ramde tutulması için için kaç tane resimlik allocation yapılacağı. (Bunu video stream tarzı işlerde 1den fazla tutmak daha iyi sonuç verir diyen bir internet sitesi görmüştüm ama neden emin değilim)

**NOT 5:**

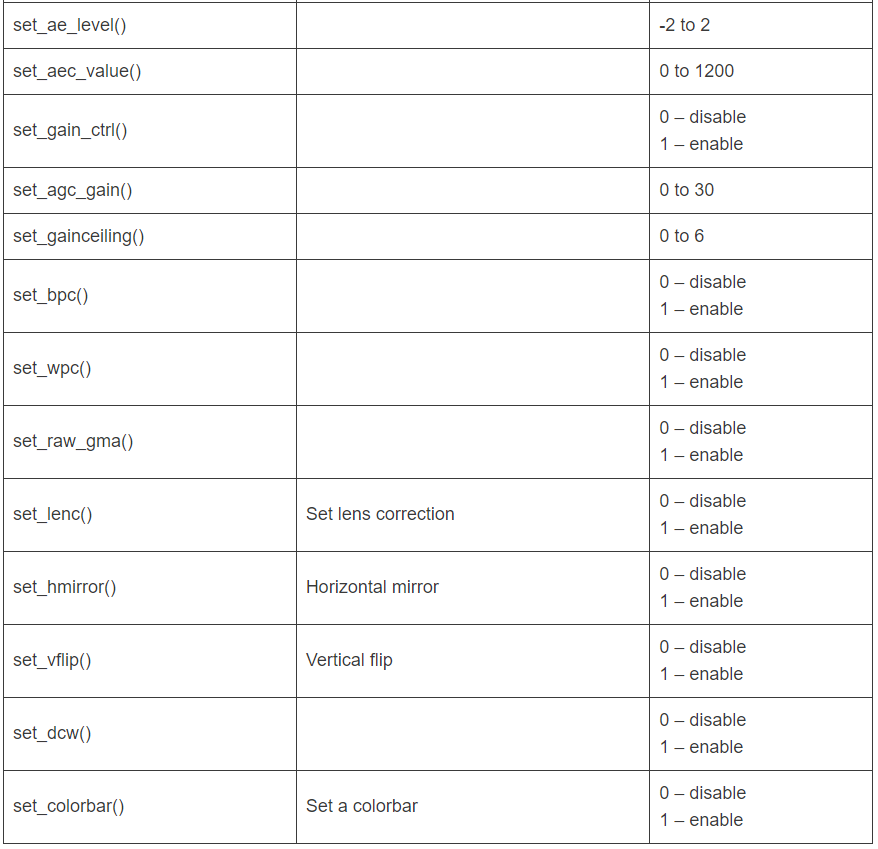
* “esp\_deep\_sleep\_start()” esp’nin kendi fonksiyonlarından birisi ama esp librarysinin olması lazım bunun için. İşlevi kendini softwarele kapatmak.

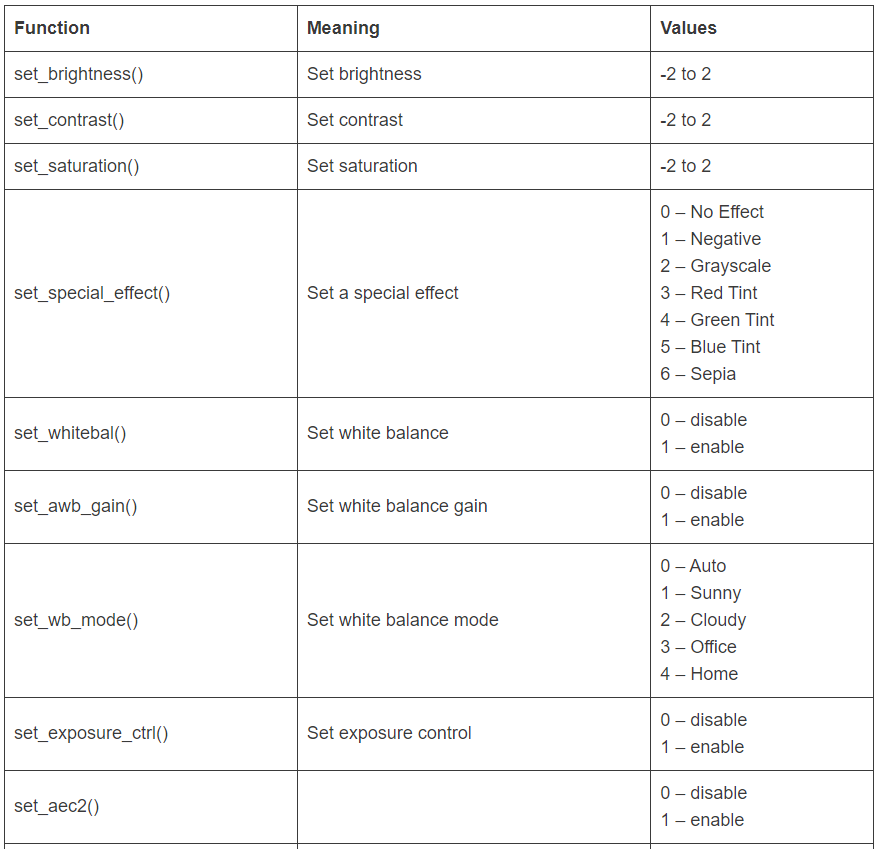
**NOT 6:**

* “dX cap” gibi ifadelerdeki “d” dummy manasında, yanındaki sayı da kaçıncısı olduğu. Bunlar debuglamak için kullandığım şeyler ama yine de çıkarmadım.
* İleride “pX” ve “cX” de var, onlar da kaydedilen fotoğraf manasında capture, indexi de burstteki kaçıncı resim olduğu.
* Burada yeşil fotoğraf sorunu için bekleye bekleye 4 tane resim çekilip geri allocationlar releaseleniyor.

**NOT 7:**

* Burası her 50 burstte bir automatic gain control – gain’i 1 arttırarak etkisini görmek içindi ama kolaylıkla aynı fikirle başka parametrelere de uygulanır.





Kapanış notları:

* Genel olarak kod biraz karışık ve variable isimleri biraz tutarsız, yazarken beynimin çoğunu o sırada imkansız gibi görünen olayları (mesela yeşil resimler) anlamlandırmakya kullandığım için buraya çok odaklanamamışım, onun için özür dilerim.
* Kesinlikle camera ve sensor librarylerini internetten bulup (<https://github.com/espressif/esp32-camera>)( <https://github.com/espressif/esp32-camera/tree/master/driver>) bir bakıma manual olarak el altında tutmayı öneririm.
* Bu ESP’yi ben Arduinonun bilgisayar uygulaması ile kullandım. Onunla alakalı gerekebilecek bilgiler:

metin, ekran görüntüsü, sayı, numara, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bir de sanırım klon arduinolarda da kullanılan USB-Serial BUS IC’si (CH340’dı sanırım) için driver lazım.

* Kendi kullandığım hardware setupunun şemasını da ekliyorum. O ESP boardını bölerek kullanmamın sebebi pinleri serbestçe kullanabilmekti. Şuanda neden 10k var devrede emin değilim ama sanırım bir sorun oluyordu o değerlerde bir direnç koymayınca, bir de üşenmeyip şemaya dahil etmişsem bir sebebi vardır diye tahmin ediyorum.
* Boardın kamera flaşı olarak kullanılan LEDi (sanırım GPIO4) camera librarysi tarafından kontrol edilyor sanırım, ve gözümü almasın diye üstünü bantlayınca siyah elektrik bandını eritmişti, dikkat edin. Softwarele kapatmaya çalıştım ama ben başaramamıştım.