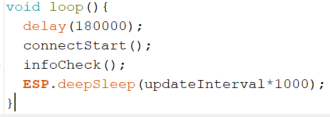
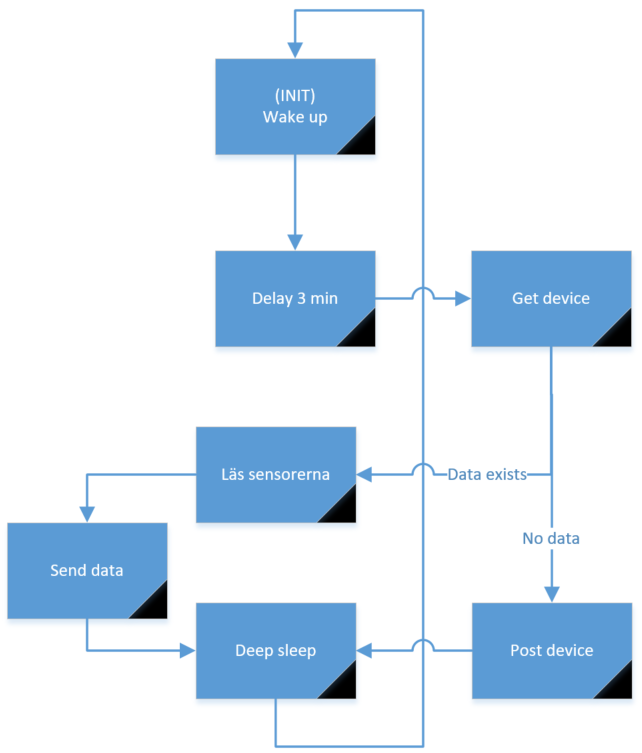
## Energioptimering

Enheten kunde energi optimeras genom flera metoder. Kollar vi på hårdvaran kunde vi optimera en sensor själva medan de andra kunde bli testade och ersatt. Den sensorn var LDR som vi med större resistans kunde minska strömmen genom sensorn. Mjukvaran kunde optimeras genom att ha en mindre delay i början då jag testade under de tre rekommenderande minuterna läsa av co2 sensorn vilket efter 1 minut gav mig värden med liten avvikelse från varandra. Man kunde även använda sig utav mqtt då jag märkte att sändningen av data tog några sekunder. Dessutom kunde man minnas om man hämtade enheten och då i framtiden hoppa över den funktionen och följaktligen ha på enheten ca 5 sekunder mindre varje vakning. Problemet förekommer då vi ändrar i databasen genom att antingen ta bort registationen eller ändrar sleep intervall då koden tror den är registrerad eller har rätt sleep intervall när den inte har det. Dessa energioptimeringar ändrar enhetens funktion lite. Men det finns optimeringar som att kolla ljusstyrkan eller tiden och köra programmet följaktligen tex bara när det är ljust, dag eller vardag. Det jag använde mig utav i min energioptimering var endast deep sleep då annan energioptimering skulle ändra vår datas intervall eller nogranhet vilket skulle förstöra data vi gav vår AI. Här är koden som visar användningen av deep sleep efter koden har körts som den alltid vid vakning ska pga konsekventa skäl.

Deep sleep ligger på 27 minuters intervallet med ca 3 minuter av delay och några sekunder av körning som är försumbara, då hamnar våran effekt 3/30 av körningen på 0.1A och den andra 27/30 av körningen på 0.03A Vilket annars med delay skulle vara på konstant 0.1A vi kan räkna ut effekten till att vara i genomsnitt: (27)/(30)\*0.03+(3)/(30)\*0.1 = 0.037A vilket under delay skulle vara 0.1A konstant.