NTNU

elsys
ELEKTRONISK SYSTEMDESIGN & INNOVASJON

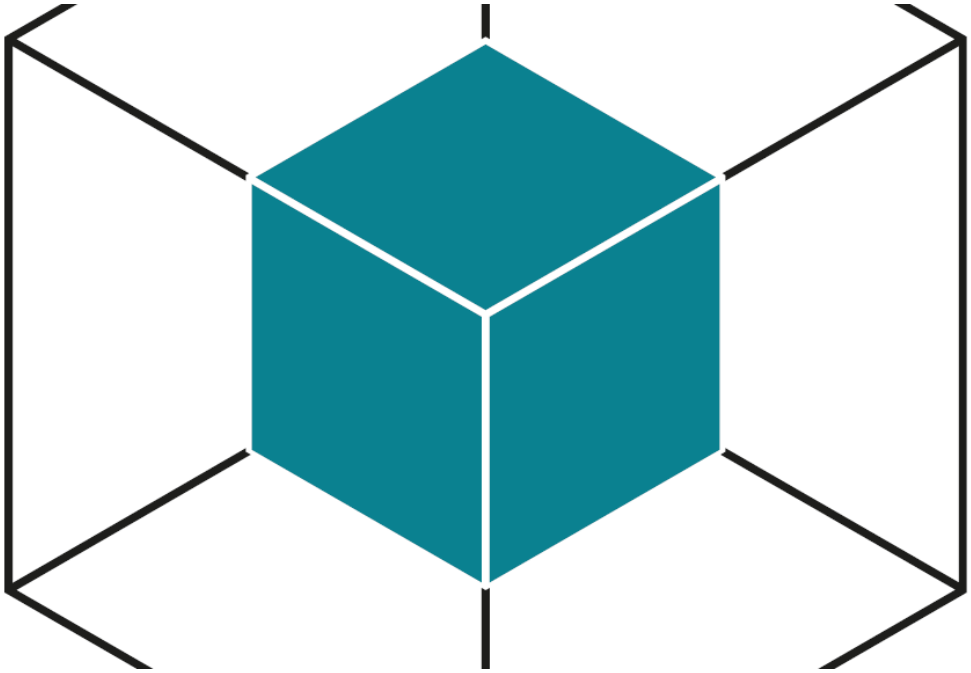
TTT4255 Elektronisk systemdesign, grunnkurs

I2: Arduino IDE

Elektronisk systemdesign og innovasjon

Ida Bjørnevik, Sven Amberg, Amalie Fridfeldt Hauge og Peter Magerøy

28.06.2023



Innhold

Introduksjon

I denne modulen skal du lære å bruke Arduino IDE.
- Installere programmet
- Installere bibliotek
- Konfigurere for riktig type kort
- Skrive ditt første program og laste det opp
- Lære å bruke seriell overvåker

For å få ESP32 til å oppføre seg på ønsket måte, det vil si gi spenning på rett pins og ta imot signal på pins, må vi skrive kode som vi laster opp til ESP32. Vi kobler mikrokontrolleren til en PC med USB-kabelen, det er her kommunikasjonen mellom PC og ESP32 foregår.

Ved å bruke en IDE (Integrated Development Environment) kan vi enkelt skrive kode og laste opp til ESP32, det er gjort ved noen få tastetrykk. I emnet TTT4255 har vi valgt å bruke Arduino IDE. Arduino IDE er en IDE laget for en annen mikrokontroller som heter Arduino. Denne ble tidligere brukt i dette emnet. Kortet deres heter ESP32-S2-Saola-1. ESP32 støtter også samme IDE, noe som er veldig fint.

NB: Dere har et kort basert på ESP32-S2. Søker dere opp problemer bør dere skrive dette og ikke bare ESP32, ettersom dere da sannsynligvis vil få opp resultater for en annen mikrokontroller.

Det finnes mange forskjellige programmeringsspråk og mange forskjellige programmer til å skrive i. Vi valgte Arduino IDE for ESP32 fordi det er så utbredt i maker-verdenen. Du finner eksempler og hjelp veldig enkelt, og det er veldig enkelt å bruke.

Programmeringsspråket som brukes i Arduino IDE er en forenklet type av C++ språket. Så det er veldig godt egnet for nybegynnere, men også for de mer viderekomne. Det fine med å bruke dette programmet er at du da også kan lett bruke kode ment til Arduino. Arduino er mye brukt i DIY-prosjekter og for å lære opp nybegynnere om mikrokontrollere.

Du vil lære mer om IDE-er og programmeringsspråk i ITGK.

Installering lenke

Trykk på lenken under, last ned Arduino IDE og følg instruksjonene.
- Link for Arduino installasjon

Her er noen nyttige Arduino lenker.
- Arduino sin hjemmeside
- Arduino getting started guide
- Installere Arduino IDE for Windows
- Installere Arduino IDE for Mac
- Installere Arduino IDE for Linux

Prosjektet

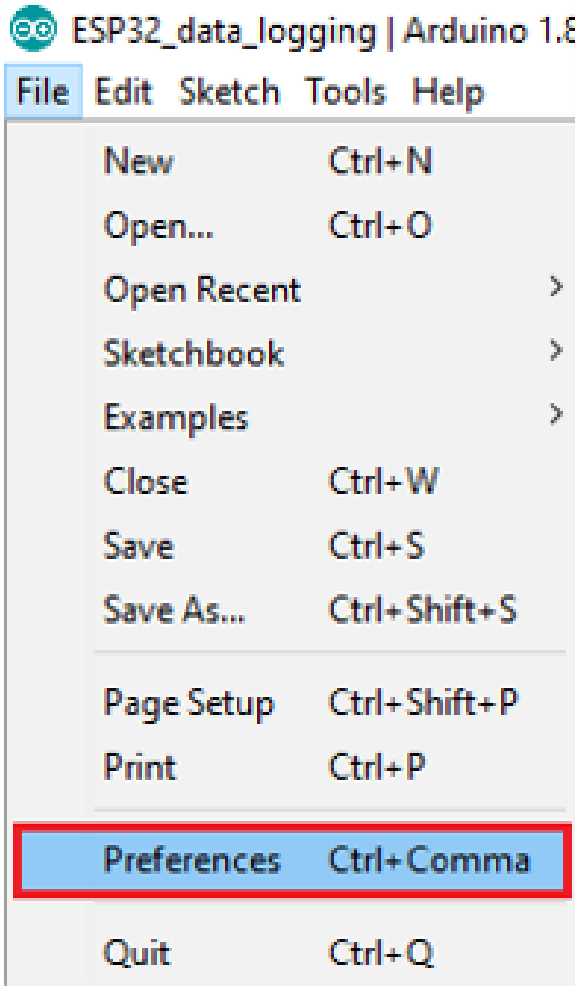
Utstyrsliste

- PC med Arduino IDE installert

Steg 1: Sett opp Arduino IDE

Etter installasjonen kan du koble til ESP32 med USB ledningen og starte programmet. Før du kan starte å programmere for ESP32 må vi installere kort alternativet og biblioteker. Dette gjør Arduino IDE kompatibel med ESP32.

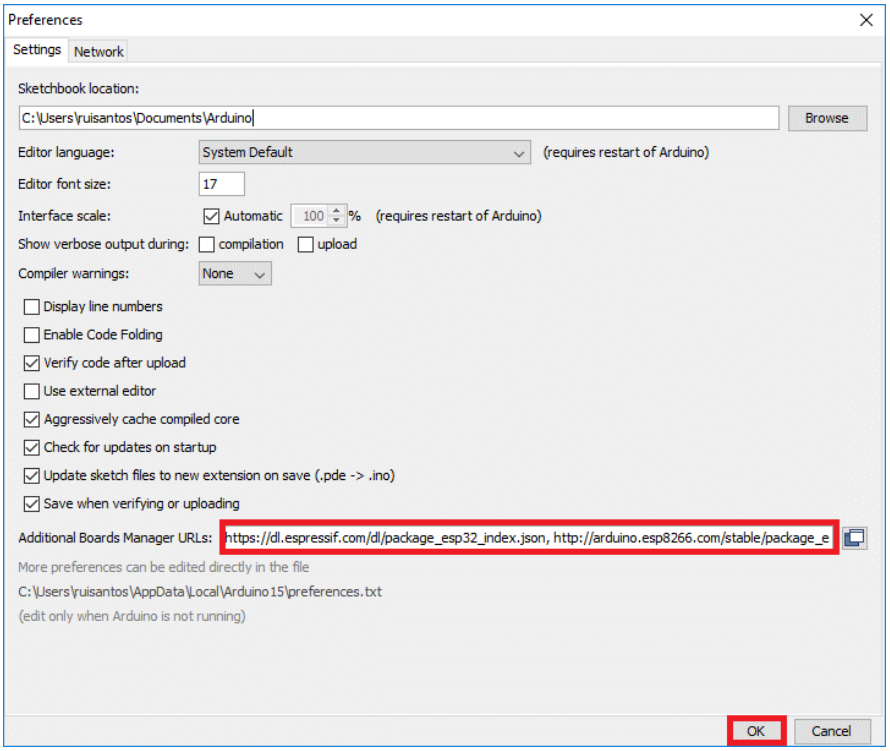
Gå til File > Preferences.



Figur 1: Navigering til Preferences.

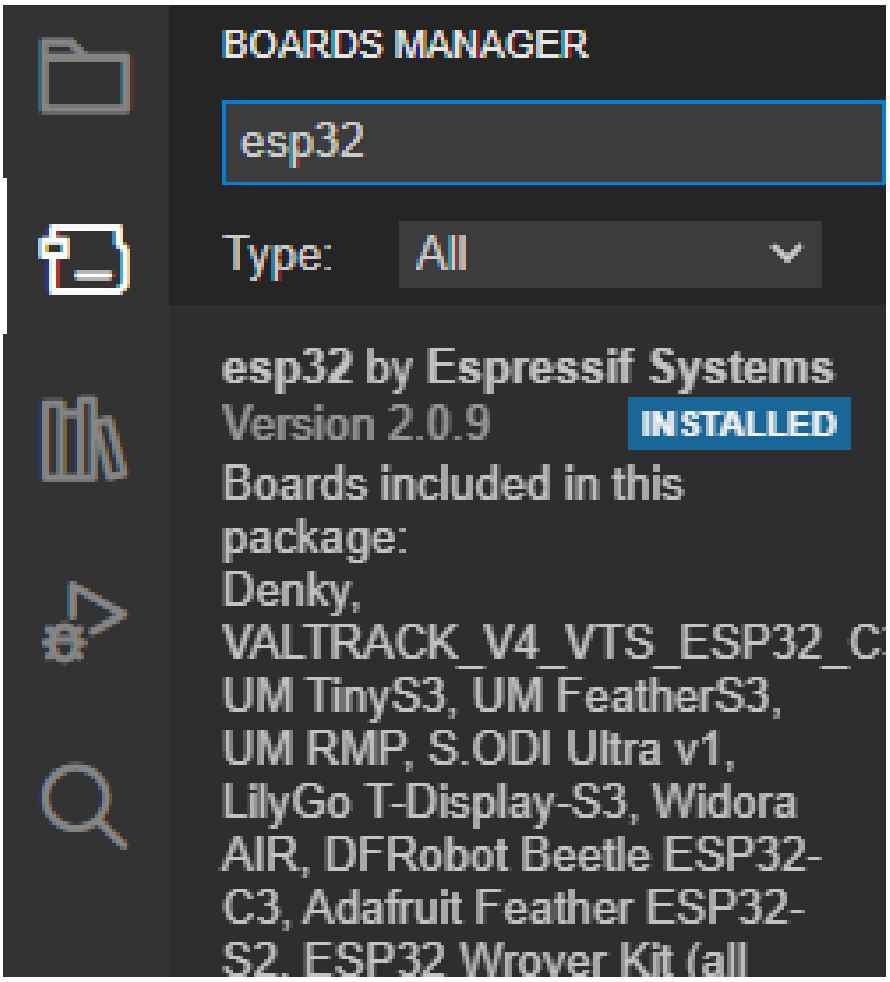
Legg til `https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json` inne i "Additional Board Manager URLs" feltet som vist i

figur 2. Her bør du også endre til dark theme om du ønsker en kulere IDE. Trykk så på OK.



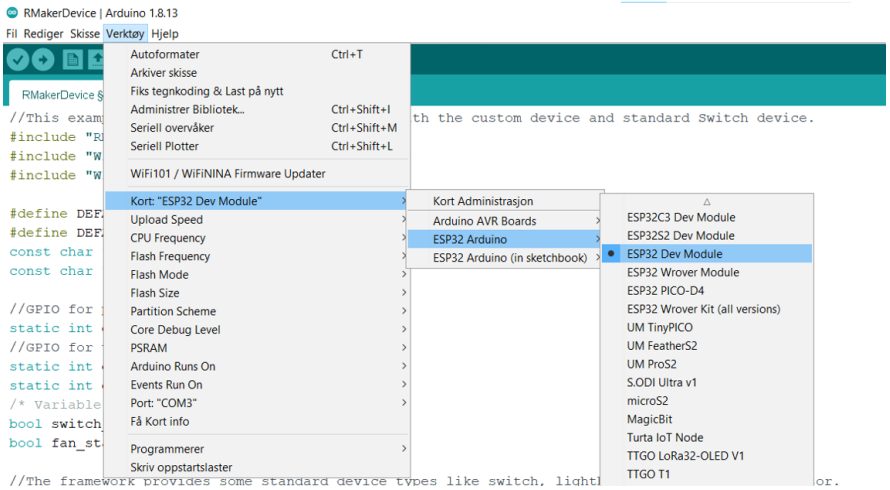
Figur 2: Inne i Preferences.

Gå til Tools > Board > Boards Manager. Søk på ESP32 og trykk på Install.



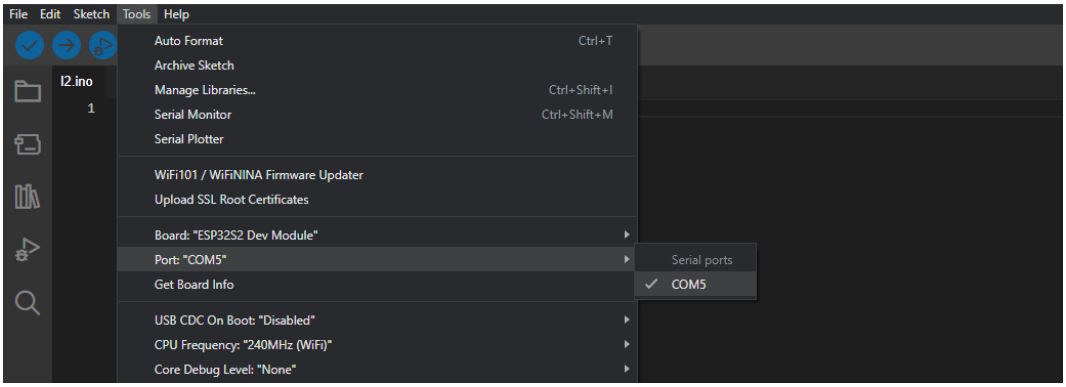
Figur 3: Navigering til kortadministrator.

Gå til Tools > Board og kontroller at *ESP32S2 Dev Module* er valgt som vist i figur 4.



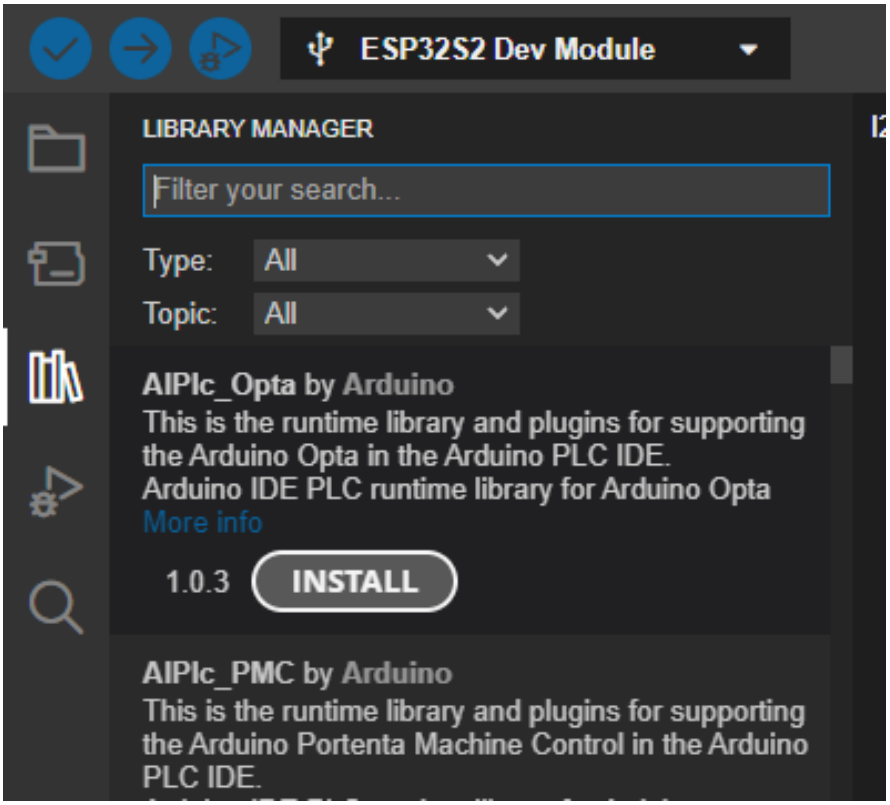
Figur 4: Velg riktig kort.

Sjekk at riktig Port er valgt, det ser du ved å gå til Tools > Port (Kan hende du må installere CP210x USB to UART Bridge VCP Drivers for at PC-en kan se at USB med ESP32 er koblet til). Porten skal stå som COM<nummer>. Spør læringsassistentene dersom du trenger hjelp.



Figur 5: Velg riktig port.

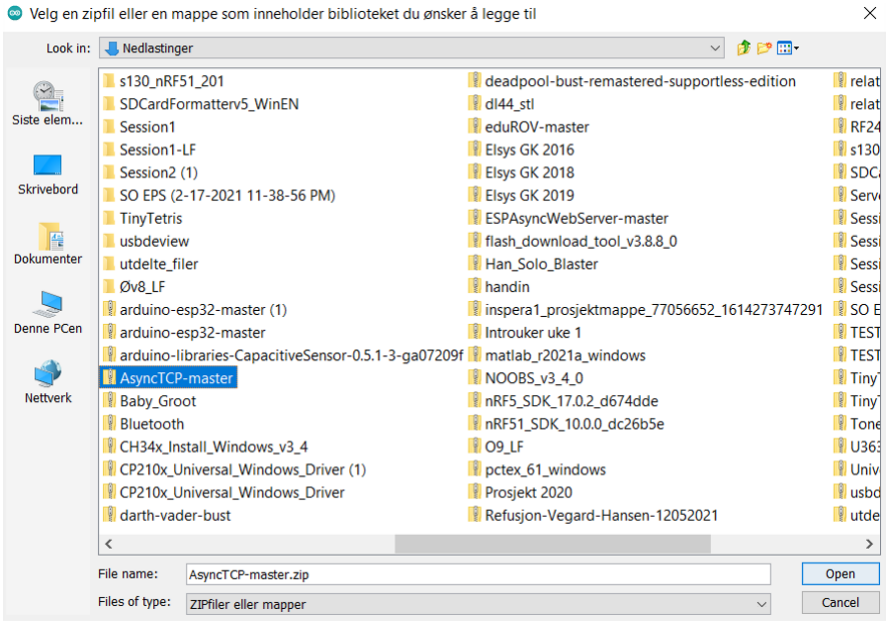
Noen ganger vil du måtte installere biblioteker. Dette er kode noen allerede har skrevet for oss og som kan hjelpe i vårt eget prosjekt. For å installere et bibliotek går du inn på Sketch > Include Library og velger enten **Manage Libraries** eller **Add .ZIP Library**. I modulene er det forklart nøye dersom du må installere noen biblioteker. Prøv først å gå til bibliotek administrasjon ved å navigere til **Manage Libraries**. Du skal få opp et vindu som vist i figur 6.



Figur 6: Dette er bibliotekadministrasjonen.

Som oftest kan du finne bibliotekene i bibliotekadministrasjonen. Der søker du bare på navnet og trykker på installer.

Dersom biblioteket ikke finnes i bibliotekadministrasjonen, så kan du finne de på nettet. Ofte er de på GitHub og du trenger da å laste ned Git'en som .zip-fil. Etter å ha gjort dette, naviger til **Add .ZIP Library**. Da åpner det seg et nytt vindu. Gå til mappen der du finner dine nedlastede filer. Velg .zip-mappen du lastet ned og trykk på åpne. Da vil biblioteket bli installert.



Figur 7: Velg .zip-bibliotek og installer det.

For å bruke et installert bibliotek i programmet ditt, gjør som i eksempelet under. Dersom det blir gjort i modulene er det vist i koden eller forklart hvordan du gjør det.

```
#include <WiFi.h>
```

Figur 8: Inkludere et bibliotek i koden.

Steg 2: Prøv ditt første program

Nå er alt klart for at du skal kunne skrive et program og laste det opp til ESP-en. Opprett et nytt prosjekt i Arduino IDE og skriv inn koden under.

```
#define led 17 // Gir pin 17 navnet led

void setup() {
  // Koden her kjøres kun ved oppstart
  pinMode(led,OUTPUT);
}

void loop() {
  // Koden her kjører igjen og igjen
  digitalWrite(led, HIGH); // Sender signalet "høy" til (led),
     LED-lyset vil da lyse
  delay(500); // Venter 500 ms
  digitalWrite(led, LOW); // Sender signalet "lav" til (led),
     LED-lyset blir skrudd av
  delay(500); // Venter 500 ms
}
```

Figur 9: Din første kode. Blink LED.

Teksten med `//` foran er kommentarer i programmet. Dette betyr at du kan lese det, men datamaskinen kan ikke. Når programmet blir kjørt blir kommentarene ignorert av datamaskinen. Kommentarene er der for å hjelpe et menneske å lese koden. Dette lærer du mer om i TDT4109 - Informasjonsteknologi grunnkurs. Merk at `//` kun kommenterer ut for den linjen og du vil ikke kunne kompilere om kommentaren går over flere linjer.

Steg 3: Beskrivelse av koden

Første linjen forteller programmet at GPIO pinne 17 på ESP-en skal ha navnet **led** i programmet. Så hvis du skal bruke GPIO 17 senere i programmet kan du skrive **led** i stedenfor tallet 17. Dette gjør koden mer leservennlig, men har ikke noe å si for hva som skjer når koden blir kjørt.

```
#define led 17
```

Figur 10: Define.

Setup()-funksjonen i Arduino IDE er den delen av programmet som kjører først og bare en gang. I vårt program sier vi at **led** skal ha modusen **OUTPUT**. Altså skal GPIO 17 sende ut et signal. ESP32 har bare digitale pins. Det betyr at spenningen alltid er høy/1 (3.3 V) eller lav/0 (0 V). Et digitalt signal består av enere og nullere, altså høy eller lav spenning på pinnen.

```
void setup() {
  pinMode(led, OUTPUT);
}
```

Figur 11: Setup()-funksjonen.

Loop()-funksjonen vil kjøre om og om igjen i programmet. Så alt som skrives inni den vil gjenta seg. I vårt program sier vi at **led** skal skrive ut et digitalt signal som er høyt. Det betyr at GPIO 17 skal sende ut en spenning på 3.3 V konstant. **delay(500)** sier til programmet at det skal stoppe og vente i 500 ms. Så sier vi at **led** skal gå lav, at spenningen som sendes ut skal være 0 V. Så skal programmet vente igjen i 500 ms.

Dette vil da gjenta seg hele tiden mikrokontrolleren har strøm.

```
void loop() {
  digitalWrite(led,HIGH);
  delay(500);
  digitalWrite(led,LOW);
  delay(500);
}
```

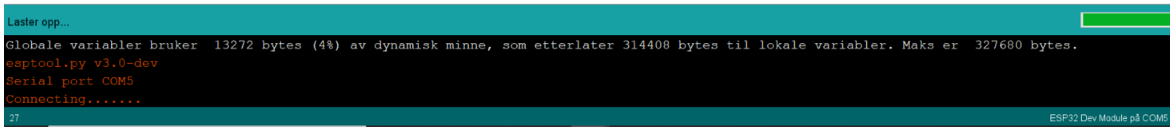
Figur 12: Loop()

For å sjekke at koden kan kompileres kan du prøve å trykke på **Verify** i IDE-en øverst i venstre hjørne. Dette trenger du ikke gjøre for å laste opp ettersom Arduino IDE gjør dette hver gang du skal laste opp koden. For å laste opp koden trykker du på **Upload**, ved siden av **Verify**. Nå må ESP-en være koblet til. Du må ha valgt riktig kort og riktig port.

NB: For at koden skal lastes opp til mikrokontrolleren må du holde inne knappen merket **BOOT** på kortet mens koden lastes opp. For at koden skal kjøre må du trykke knappen **RST** etter koden er ferdig lastet opp.

Figur 13: Trykk på knappen for å laste opp.

Nå vil du kunne se i konsoll-vinduet helt nederst at den starter å laste opp koden til ESP32. Det vil da stå **connecting....connecting....**.

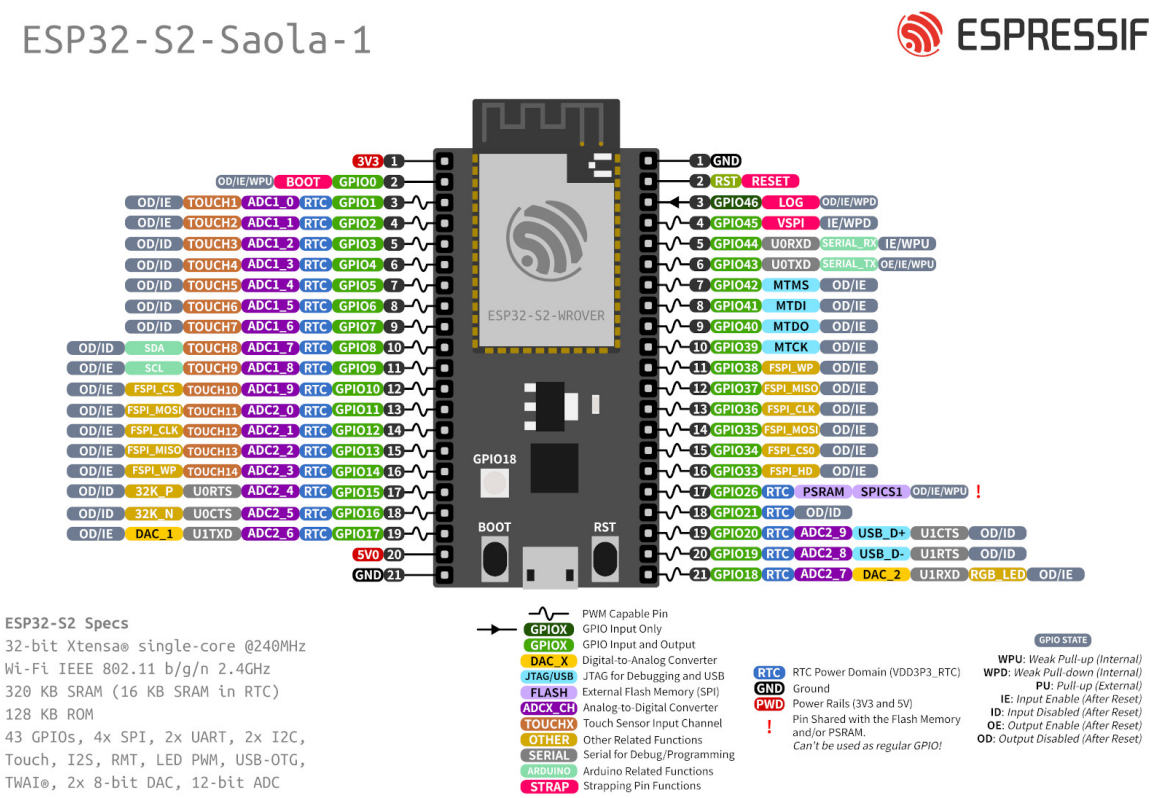


Figur 14: Hold inne boot knappen når dette vises.

Steg 4: Oppkobling

Etter å ha lastet opp koden legger du kanskje merke til at ingenting skjer. Dette er fordi vi ikke har koblet opp mikrokontrolleren enda. Nå sender den logisk høyt og lavt til GPIO 17, men det er ingenting koblet til denne pinnen. Du kan nå prøve å holde en LED med den lange pinnen på LED-en inntil GPIO 17 og den korte inntil GND. Disse pinnene skal være merket med hvit skrift på mikrokontrollerkortet med henholdsvis tallet 17 og skriften GND. Da skal du se LED-en lyse.

Vi har enda ikke forklart hva en GPIO er. Dette er pinner på kortet som ESP-en kan bruke for digital input og output. Som du har sett bruker vi nummeret på GPIO-pinnen i koden når vi ønsker å bruke den. I figur 15 er det vist pinout til kortet deres. Pinouten viser hvilke pinner på kortet som har hvilke muligheter og denne er veldig nyttig å ha når du skal koble opp ESP-en.

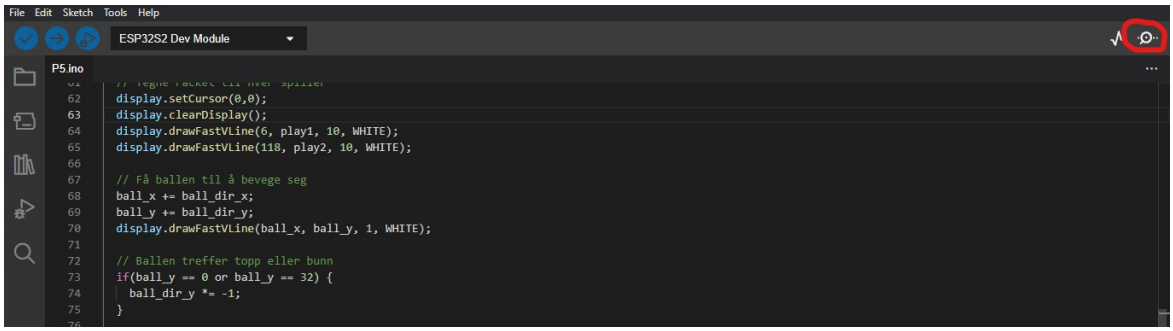


Figur 15: Pinout til ESP32-S2-Saola-1.

Legg merke til at GPIO pinnenummer ikke er det samme som pinnenummeret på kortet. For eksempel er pinne nummer 2 på kortet GPIO 0. Nummerene markert på selve kortet er GPIO nummer, ikke pinnenummer.

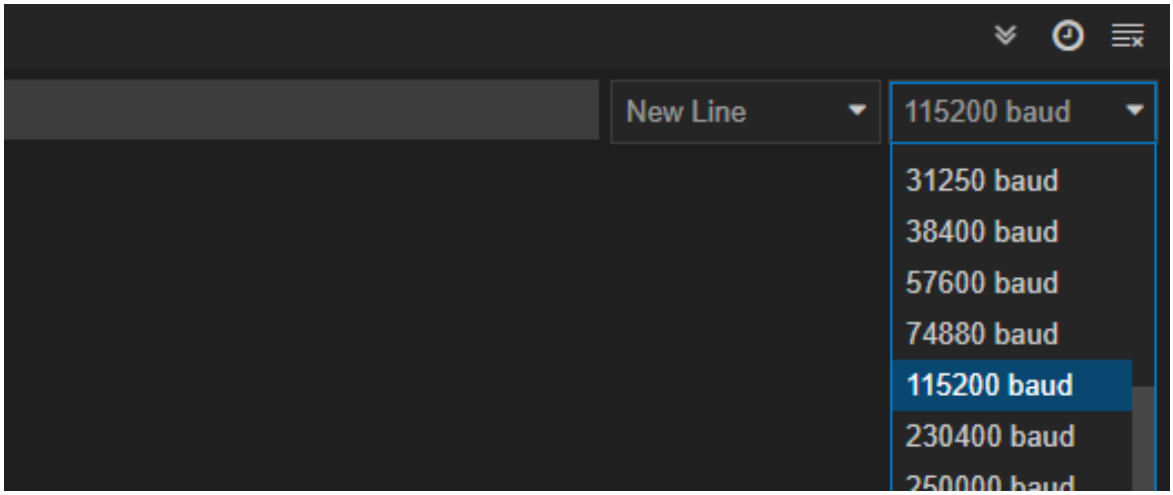
Steg 5: Seriell overvåker

Det kan være nyttig at ESP-en kan sende informasjon til PC-en mens den kjører. For å gjøre dette har Arduino IDE innebygd en seriell overvåker. I programmet vårt kan vi legge inn at mikrokontrolleren skal sende informasjonen. For å åpne seriell overvåker, trykk på forstørrelsesglasset øverst i høyre hjørnet i IDE-en mens programmet kjører på mikrokontrolleren og den er koblet til PC-en med USB.



Figur 16: Slik åpner du seriell overvåker.

Helt til høyre i seriell overvåker kan dere velge baudraten. Denne må være lik som slik den er definert i programmet som kjører på ESP-en. Dette er hastigheten på kommunikasjonen mellom PC og ESP.



Figur 17: Slik endrer du baudrate.

Hvordan du sender informasjon fra ESP-en kommer du til å se senere i modulene.

Konklusjon

I denne modulen lærte du hvordan du setter opp Arduino IDE for din mikrokontroller, samt å installere bibliotek og lage ditt første program.