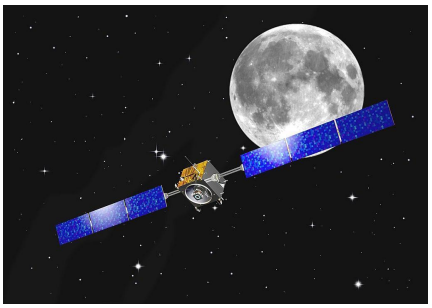
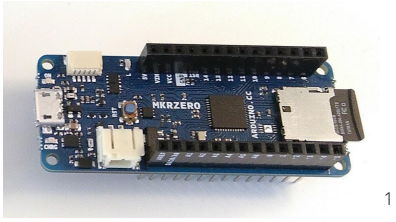
Laboration 1 - Börja använda Arduino

Grundläggande elektronik och programmering





[1]

Maskiner och datorer finns nästan överallt idag- allt från kylskåp till rymdsonder klassas som datorstyrda maskiner. I den här laborationen får du lära dig att kommunicera med en dator och få den att utföra uppgifter. Till det ska vi ska använd ett mikrokontrollerkort, en Arduino, d.v.s. ett *kretskort* med en inbyggd mikrokontroller (processor), som är lättåtkomlig både för programmering och för att bygga elektronikprojekt. Arduinon kommunicerar med en dator via USB och kan både ta emot och läsa både digitala och analoga signaler via ett antal kopplingspunkter, så kallade *pins*.

Material

- Arduino MKR ZERO
- Kopplingsplatta
- USB-sladd för att koppla Arduino till datorn
- Dator
- Resistor
- Diod
- Sladdar för koppling

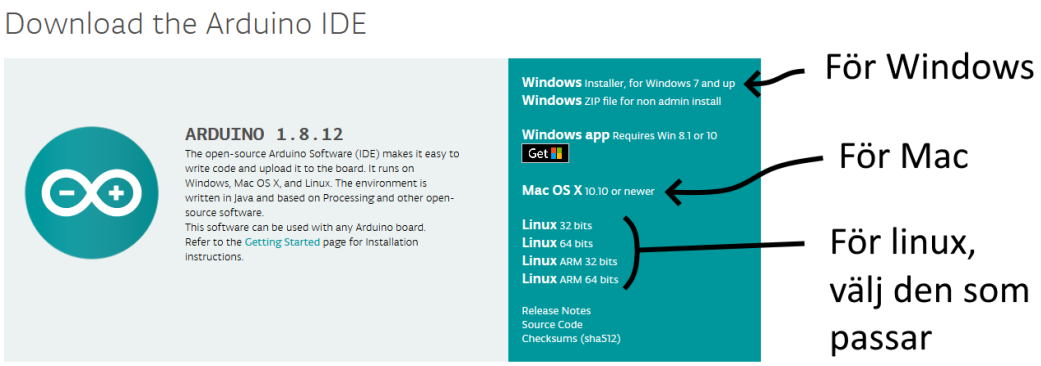
[1] SMART-1, By ESA, CC BY-SA 3.0-igo, https://commons.wikimedia.org/w/index.php?curid=59274312 Kylskåp, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=539461

E-post: cansat@au.se
Telefon: 070-000 90 56
Senast uppdaterad: 3/10 22

Installation

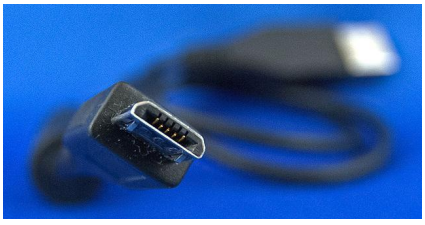
För att börja använda en Arduino: ladda ner programmet från Arduino Software (IDE)[2] eller använd online-verktyget.

Notera: Denna labbinstruktion kommer utgå ifrån det nedladdade programmet.



När programmet är nedladdat och installerat: starta programmet och anslut Arduinon till datorn med en USB-sladd.

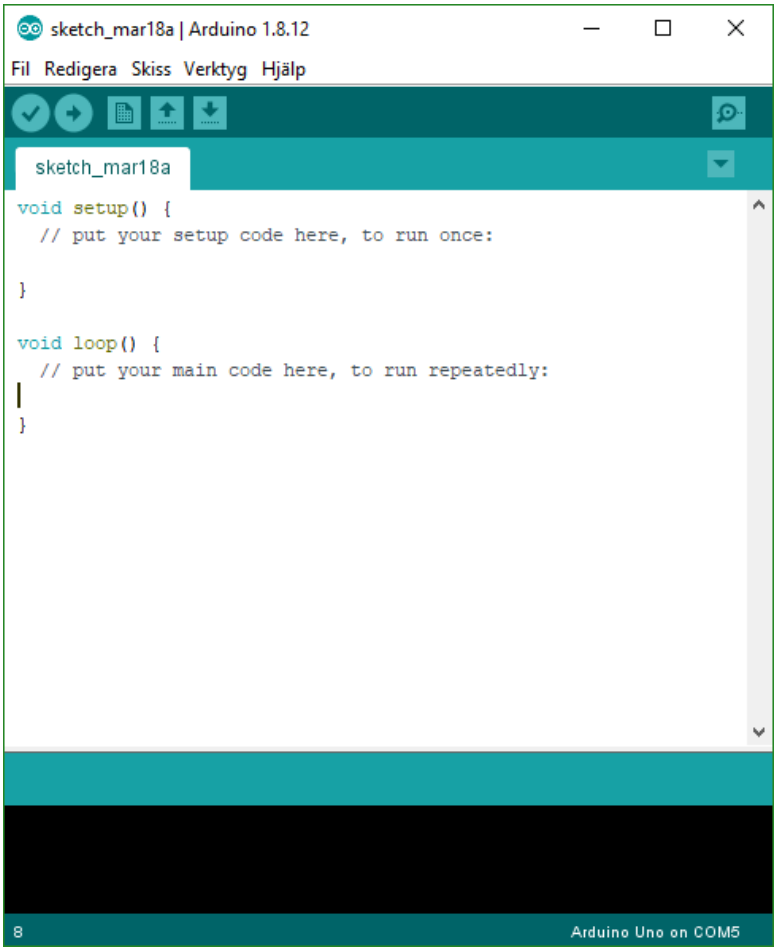
Om du använder en **Mac** kan du få ett konstigt meddelande om att din dator har hittat en okänd enhet (ett tangentbord eller ett modem), stäng det fönstret. Det uppkommer eftersom att Macen är förvirrad och tror att Arduinon är något annat men det gör inget.

 [3]

[2] https://www.arduino.cc/en/Main/Software

[3] By User:masamic - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7573019

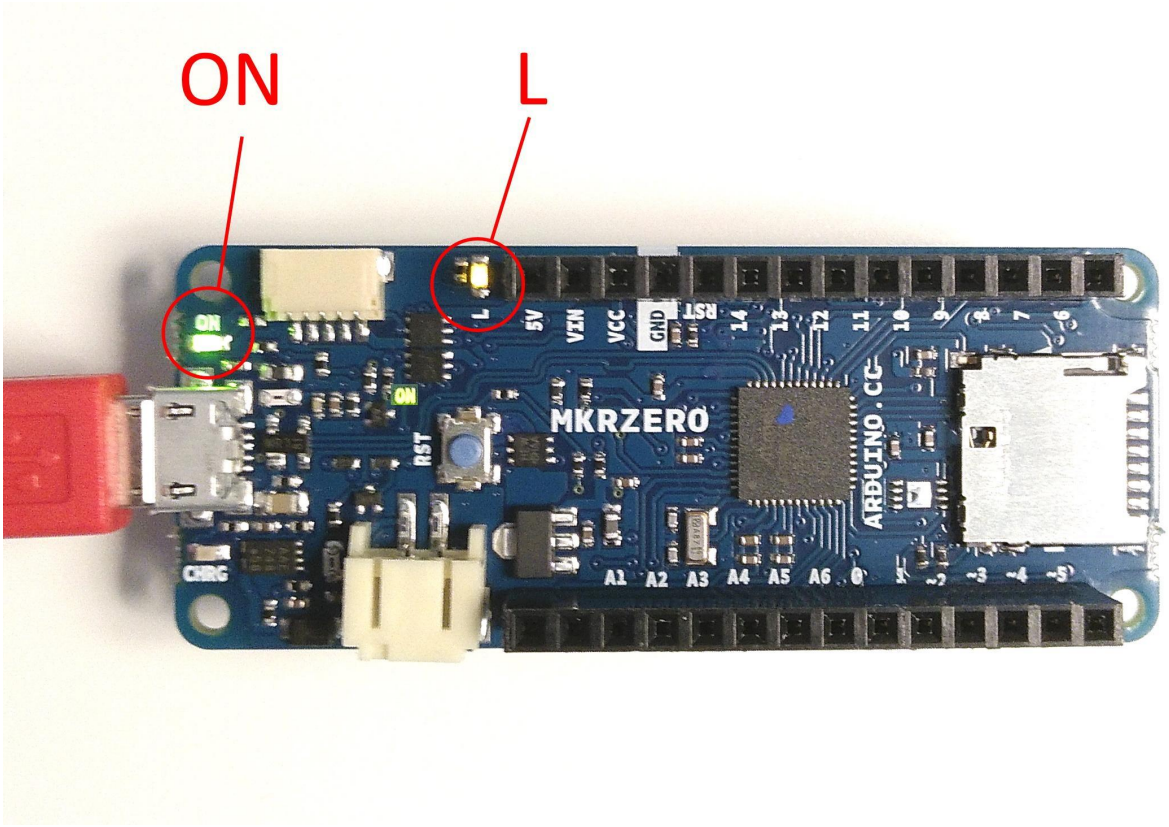
När du programmet är uppstartat får du upp denna ruta:



Att programmera är på många sätt likt att använda ett ordbehandlingsprogram (exempelvis Word, pages eller liknande). Det går att kopiera delar av filer och använda på andra ställen och det finns funktioner som att *öppna*, *spara* och *spara som*.

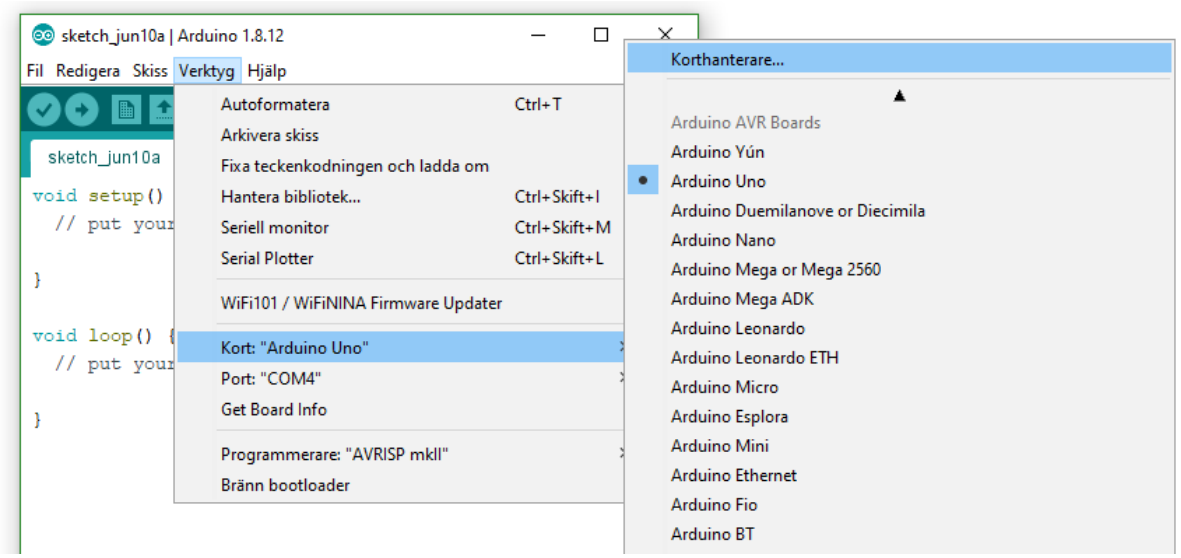
En fil med instruktioner till Arduinon (d.v.s. en kod) kommer vi att kalla för *en skiss*. En sådan tom skiss visas i bilden ovan. Man kan ändra teckenstorlek i programmet om man önskar, under Fil -> Inställningar och Redigeringsstorlek.

Arduinons ON-diod ska lysa när den är inkopplad i en dator. Dess L-diod kommer antagligen att blinka, det är det förinstallerade programmet som kör. Dioden kommer även lysa när arduinon skickar eller tar emot data via USB.

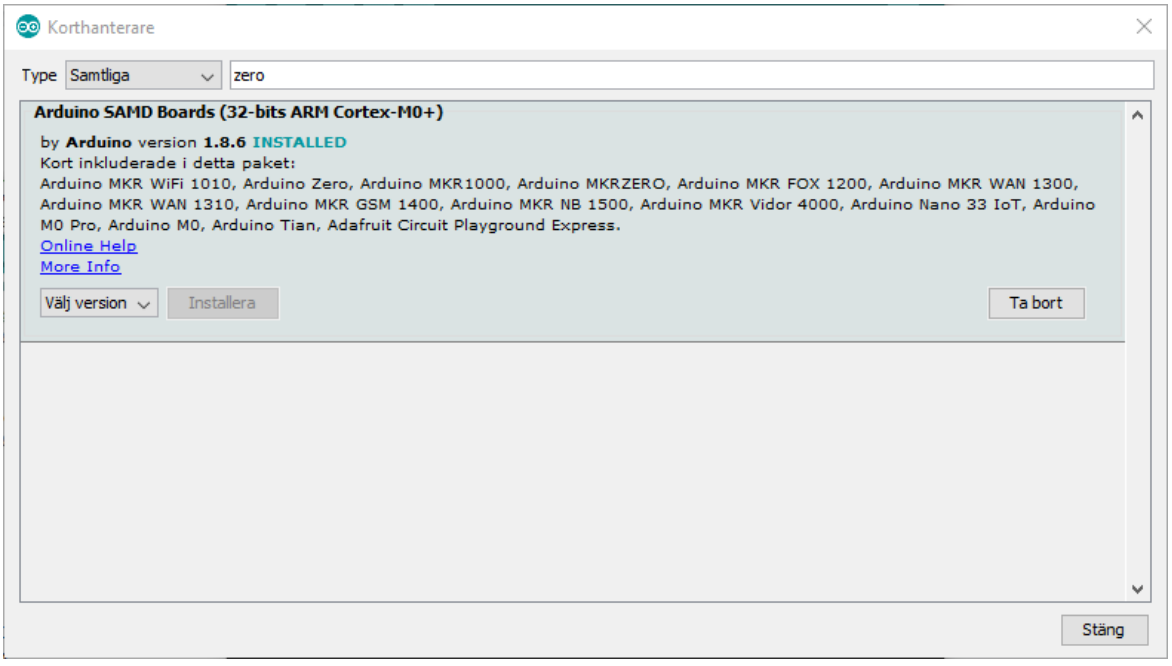


Ladda upp kod till Arduinon

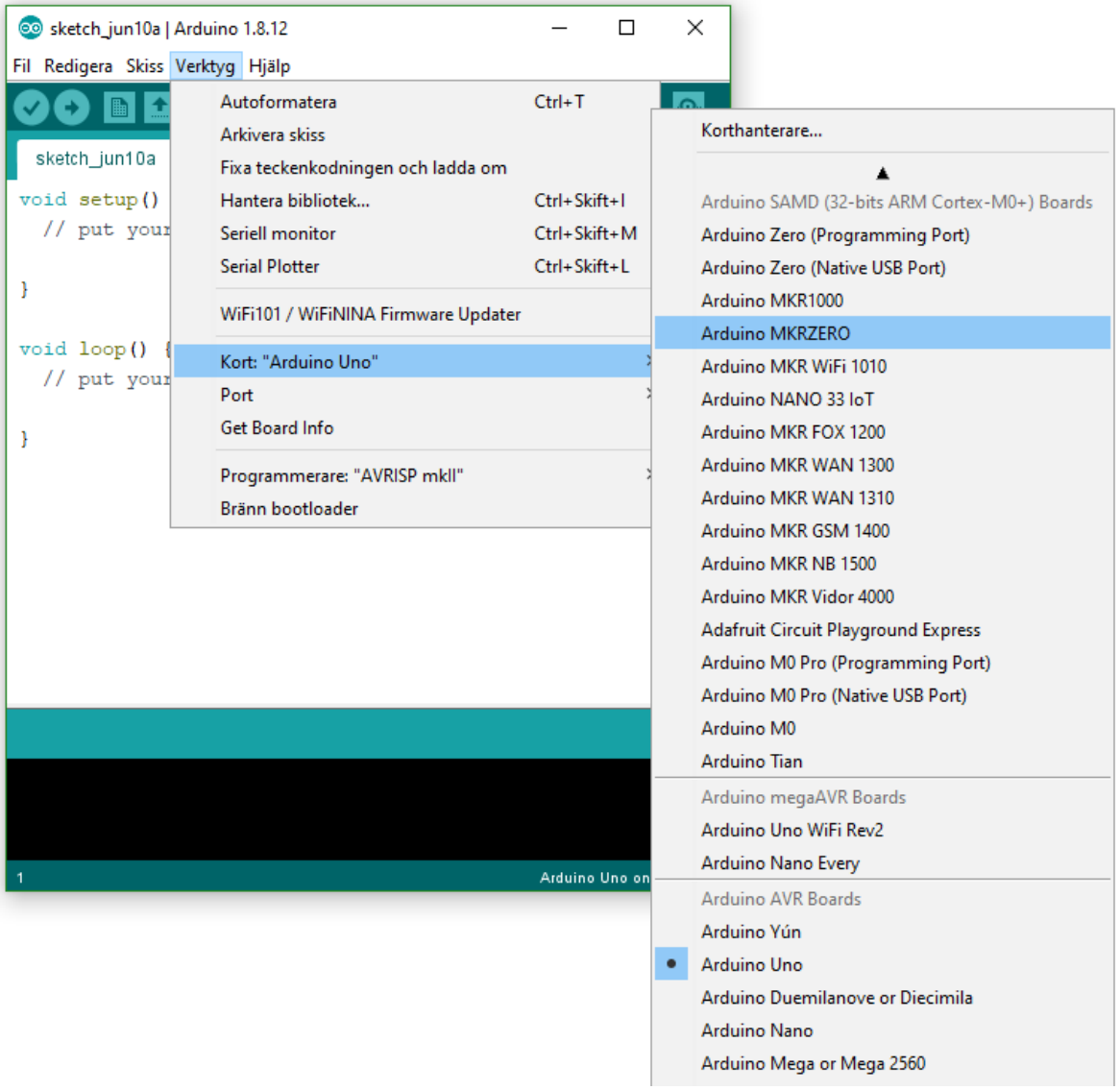
Nu ska vi ladda upp en skiss till Arduinon. Vi måste först tala om för programmet vilken typ av Arduino vi använder. Arduinon vi använder i den här labbintruktionen är en MKR ZERO och drivrutinerna för det kortet finns inte installerade från början, men vi kan lägga till dem genom att gå till Verktyg -> Kort -> Korthanterare och söka på zero.



Det vi behöver installera heter "**Arduino SAMD Boards (32-bits ARM Cortex-M0+)**" och det är det som borde komma upp. Sedan kan vi välja kortet.

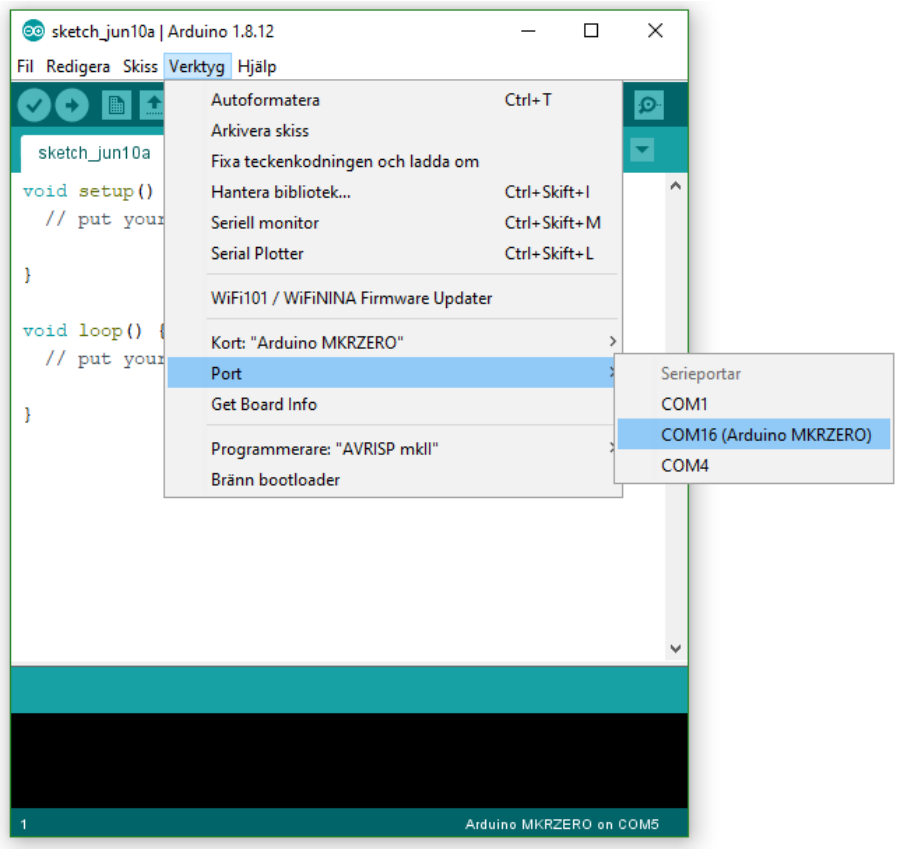


Det gör vi i programmets meny under Verktyg -> Kort. Det står antagligen *Arduino Uno*, eller något annat kort, och där vill vi istället välja **Arduino MKR ZERO.**

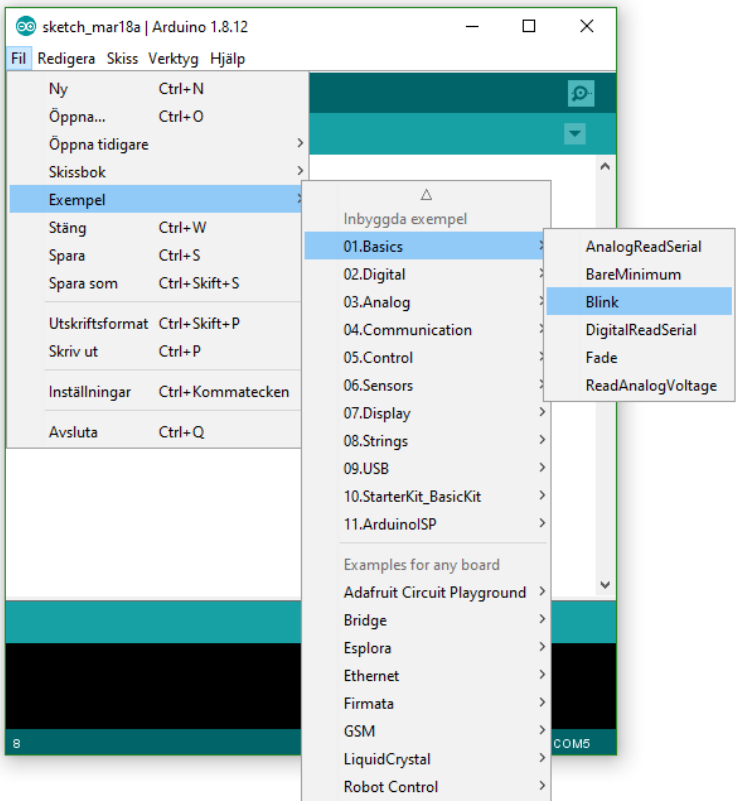


Vi måste även tala om för datorn vilken port vi ska kommunicera med Arduinon via. Detta görs genom programmets meny verktyg -> Port och välj rätt port.

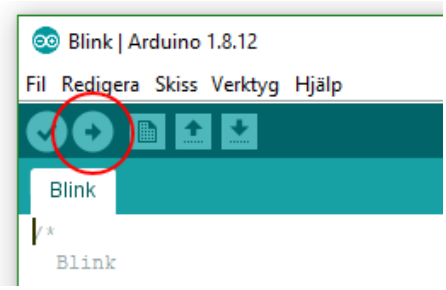
Notera: På en windowsdator kommer porten du ska välja heta COM följt av en siffra och Arduinons namn, på en Linux eller Mac kommer det finnas fler alternativ men det rätta alternativet kommer heta något med Arduino i slutet. Fråga din lärare om du är osäker.



Nu är vi redo för att ladda upp en skiss och prova kortet och programmet. I Arduino-programmets meny, gå till Fil -> Exempel -> 0.1 Basics -> Blink och välj.



Detta öppnar ett nytt fönster med en enkel exempel-skiss. Ladda upp skissen till Arduinon med knappen som visas i bilden.



När du trycker på knappen kommer det vara en kort fördröjning medans programmet förbereds och skickas iväg, sedan ska L-dioden på Arduinon börja blinka. Programmet ska visa “Uppladdning färdig.” när den är klar.

Om det inte funkar: kontrollera att rätt kort och port är valda.

Ändra kod

Nu är du redo för att modifiera programmet! Det finns två rader där det står:

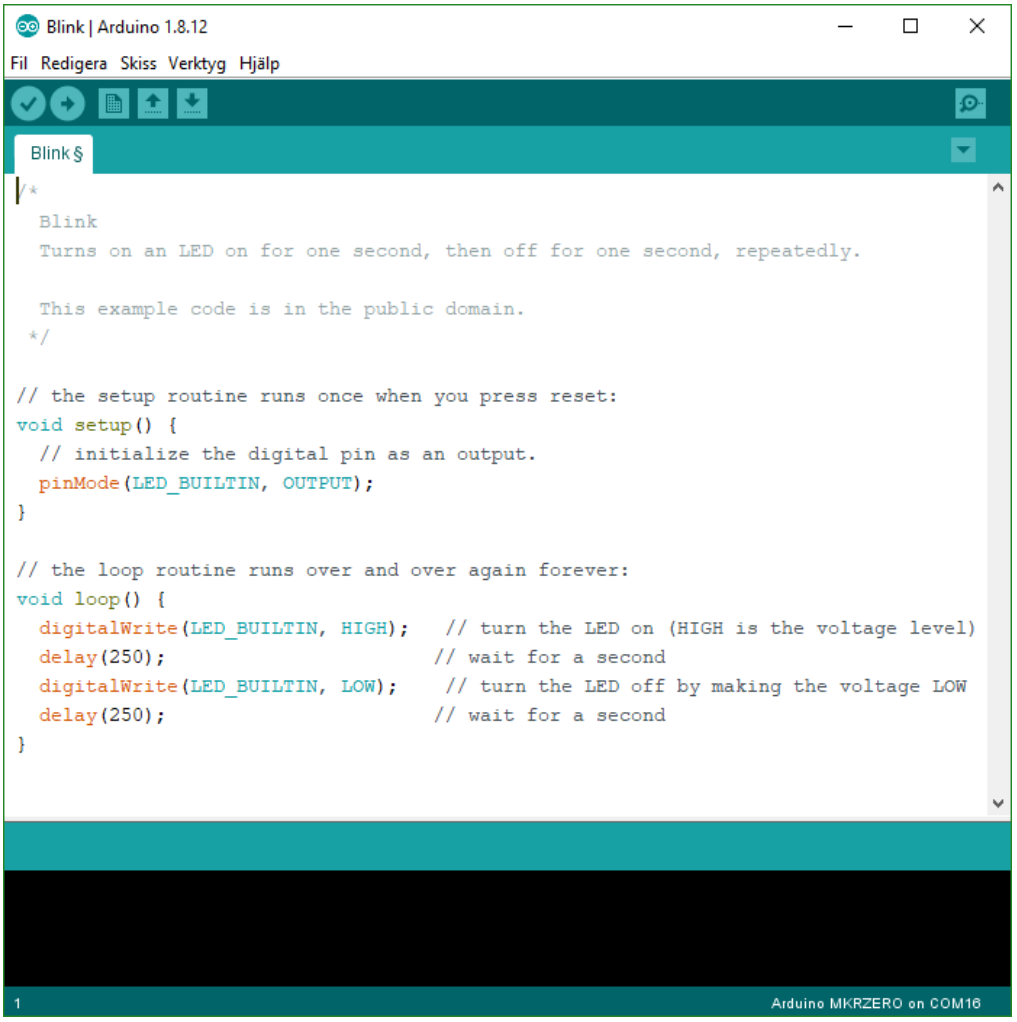
- `delay(1000);`

vilket gör att programmet väntar (eng. "delay") 1000 ms (1 sekund) efter att det slår på och stänger av lampan.

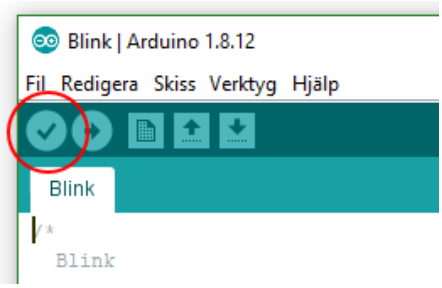
Ändra *båda* raderna till:

- `delay(250);`

. Nu har du reducerat väntetiden till en fjärdedels sekund!



Nu när du har ändrat dinMåltiderna är utformade efter ca 25 pers men mängden kan behöva varieras och ingredienser behöver såklart anpassas efter specialkost.
skiss är det en bra idé att låta programmet kontrollera det du har skrivit innan du skickar iväg det igen, så det inte kommer med några misstag. Detta kan du göra med *verifiera-knappen*, som ligger bredvid *ladda-upp-knappen*.

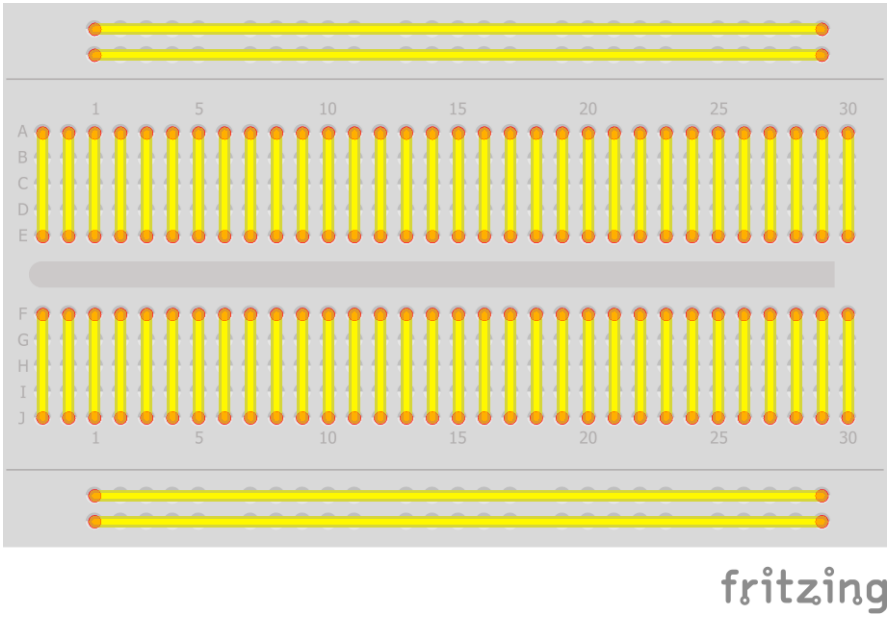


Nu kan du ladda upp den ändrade skissen. Du borde nu se lampan blinka dubbelt så fort. Spara ditt program med ett tydligt namn, förslagsvis MyBlink.

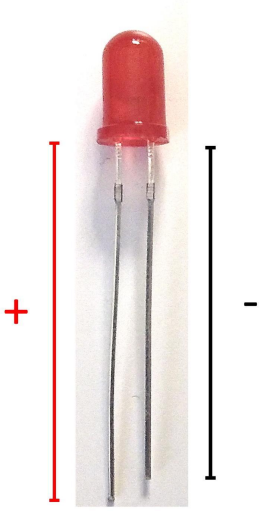
Notera: namn inte får innehålla mellanslag.

Kod som ligger mellan `/* */` eller efter dubbla snedstreck `//` är kommentarer. Kommentarer är text som programmet inte läser och används av programmerare för att skriva meddelanden eller påminnelser. Dvs. kan du skriva ner påminnelser och förklara den kod du skrivit. Det är smart att kommentera vad du ändrat i koden, så att du minns vad du ändrade till nästa gång du vill använda programmet.

Vi kan enkelt utöka vår installation till att styra flera dioder med hjälp av en *kopplingsplatta*. En kopplingsplatta fungerar genom att hålen på samma rad är kopplade till varandra så man kan sticka in de elektroniska komponenternas “ben” och göra en krets utan att behöva löda något. Bilden nedan visar hur hålen på en kopplingsplatta är sammankopplade. De långa skenorna längst ut på sidorna är användbara för att koppla spänning och jord till flera komponenter.

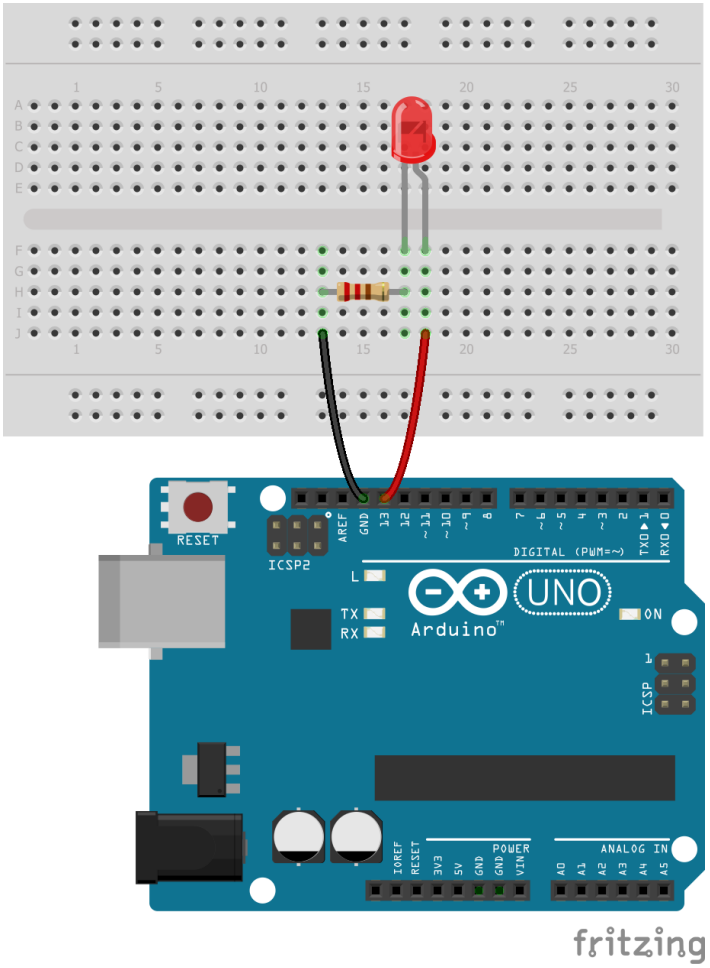


En *lysdiod* är en enkel komponent som lyser när ström går igenom den. Lysdioder är "enkelriktade", d.v.s. de kommer bara låta strömmen gå genom dem åt ett håll. Man kan se vilket håll dioden ska vara kopplad åt genom att jämföra längden på benen. Kort ben mot minus (jord) och långt ben mot plus. Lysdioden kommer även sakna en bit av kanten som går runt den längst ner på jord-benet.



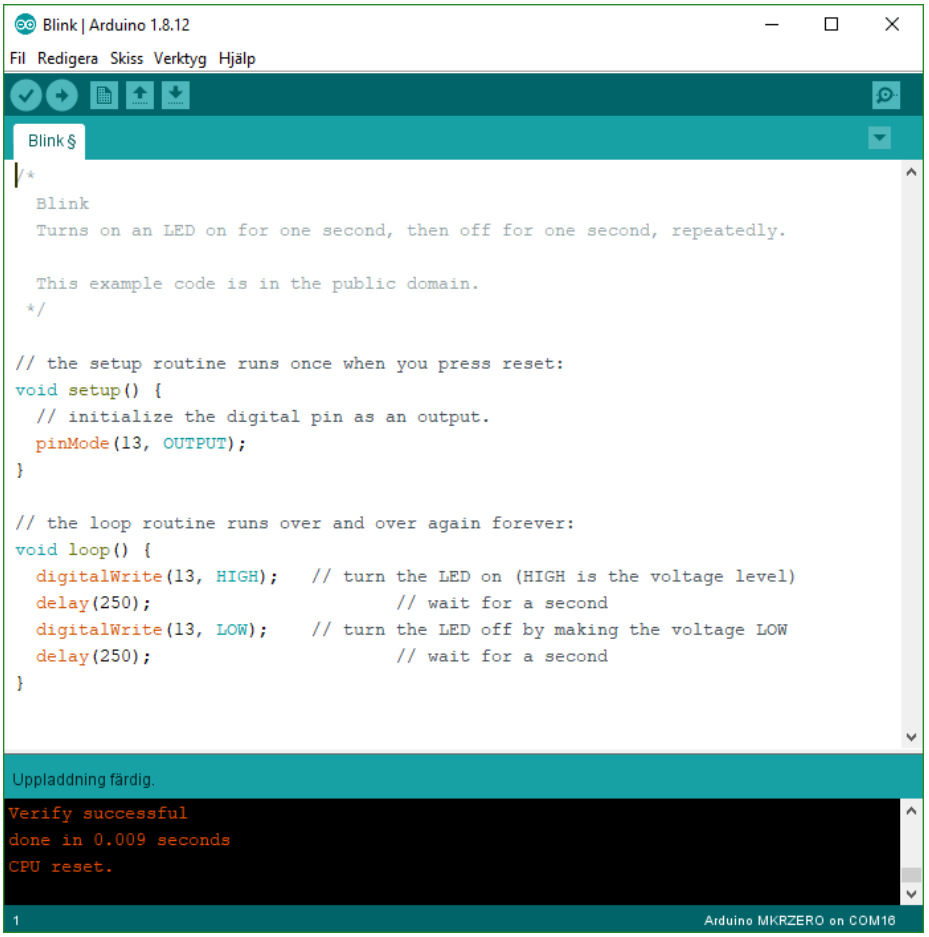
Vi behöver tala om vilken pin vi ska koppla dioden på. Den pin som heter 13 på Arduinon kommer vi att använda och bli den som matar spänning ut till en diod. Den pin som heter 13 på Arduinon kommer bli den som matar spänning ut till en diod.

Att göra: Koppla pin 13 till plus och GND (jord) till minus på en kopplingsplatta. Koppla sedan en resistor med ca. 200 Ω och en lysdiod i serie på kopplingsplattan mellan plus och minus.



En Arduino har två huvudtyper av pins. Det finns analoga pins vars nummer föregås av ett A och digitala pins som inte har det. Det huvudsakliga syftet med analoga pins är att läsa av analoga sensorer men analoga pins kan även göra allt som digitala pins kan göra. Vi kommer att använda

digitala pins om inte annat sägs. För att se till att Arduinon vet vilken pin som ska skicka ut ström, byt ut alla tre ställen i koden där det står `LED_BUILTIN` till `13` i koden.



Programmering

En Arduino programmeras med ett språk som är baserat på ett språk som kallas C. I de flesta programmeringsspråk, inklusive C, är det väldigt noga med hur kod skrivs, så kallad *syntax*. Till exempel så är C skiftlägeskänsligt, d.v.s. det är skillnad på om man använder stora eller små bokstäver. När vi använde `digitalWrite` tidigare så är det inte samma sak som `DigitalWrite`.

När en Arduino läser kod gör den det rad för rad och utför instruktionerna på en rad innan den går vidare till nästa. I C avslutas en rad med ett semikolon (`;`), vilket talar om för programmet att raden är slut.

I programmering använder man ofta *variabler*. En variabel kan ses som en behållare som innehåller ett värde som sedan kan användas. Det är väldigt praktiskt om man vill använda samma värde på flera ställen. Vi kan till exempel förbättra vårt modifierade Blink-program med en variabel för hur lång pausen ska vara, så behöver man bara ändra på ett ställe i framtiden om man vill ändra längden på pausen igen.

För att skapa en variabel i C behöver man tala om vilken typ av variabel det är, vilket namn den ska ha och vilket värde den ska ha.

Vår variabel ska vara siffran 500 så vi skriver

- `int`

först för att tala om att vi variabeln är ett heltal (eng. integer), följt av namnet på variabeln (använd inte åäö) förslagsvis

- `int waitTime`

följt ett likamedtecken och värdet variabeln ska ha,

- `int waitTime = 500`

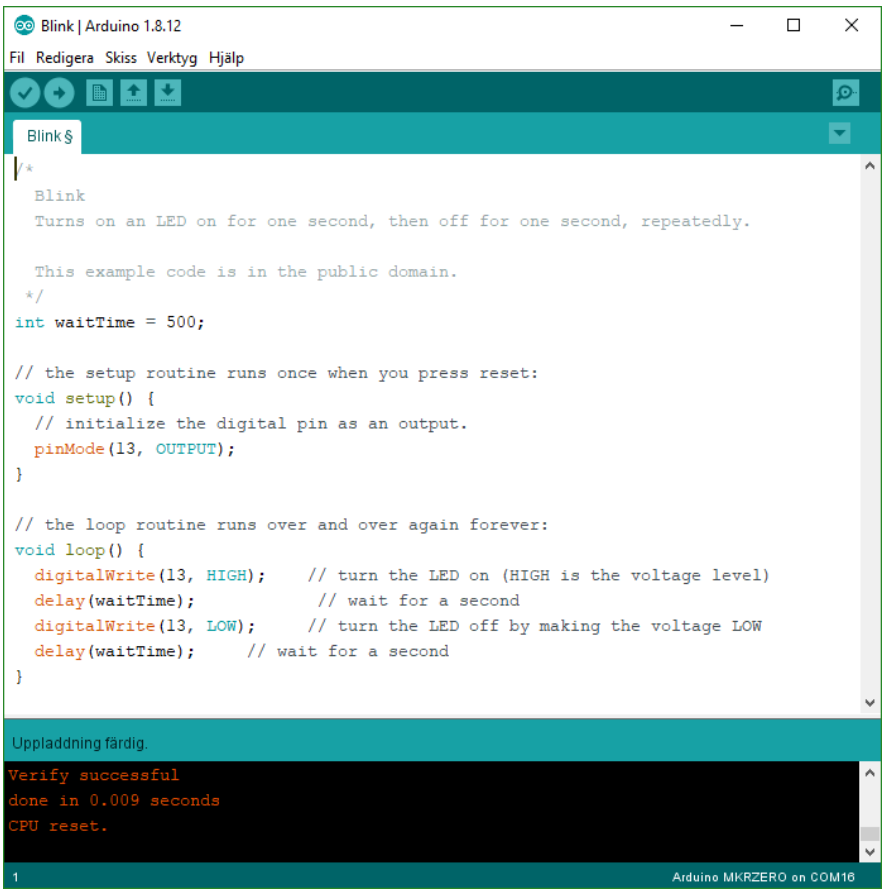
och avsluta raden med ett semikolon.

- `int waitTime = 500;`

Detta måste skrivas först i koden, innan `void setup()`. Slutligen byter vi ut 500 efter `delay` till `waitTime`.

- Från `delay(500);` till `delay(waitTime);`

Verifiera nu programmet och ladda upp det för att testa. Det ska bete sig på samma sätt som innan.



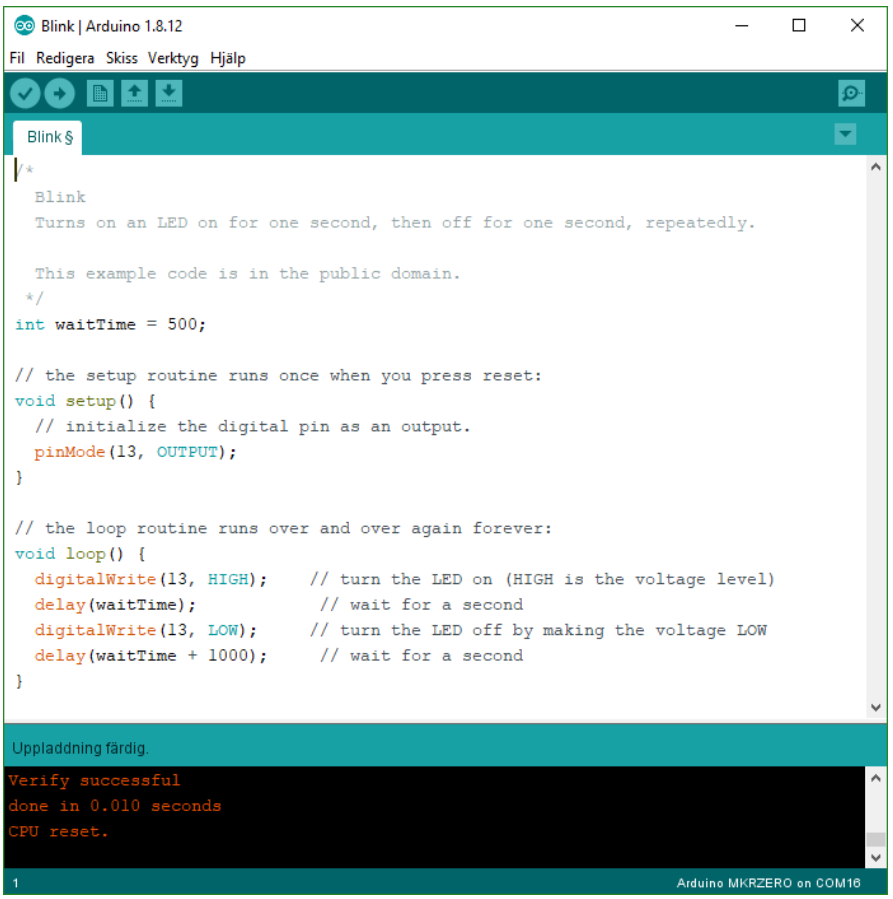
Variabler funkar på samma sätt som de värden de representerar. Man kan till exempel räkna med variabler och siffror. Om man i `delay` skriver `waitTime + 1000` så blir väntetiden 500 + 1 000 = 1500 ms. Prova att addera eller subtrahera något från en av tiderna i en `delay`, verifiera och ladda upp. Skriv om

- `delay(waitTime);`

till

- `delay(waitTime + 1000);`

Observera den ändrade rytmen i blinkningarna.



Extra övning

Återskapa den enkla kretsen med en blinkande diod fast genom att löda komponenter på ett kretskort.