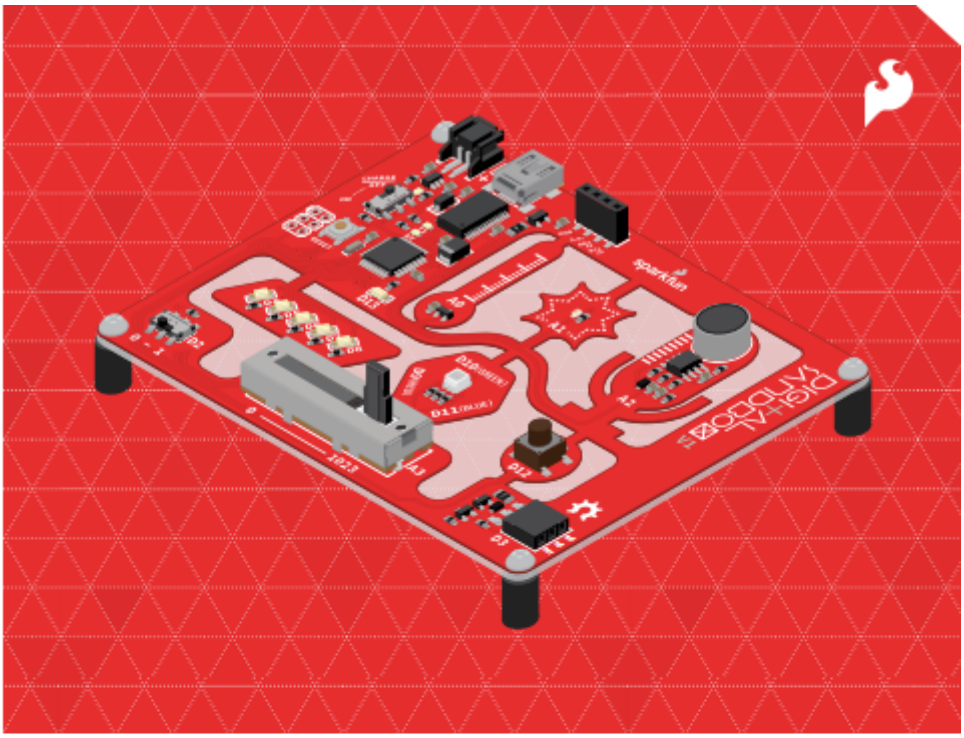
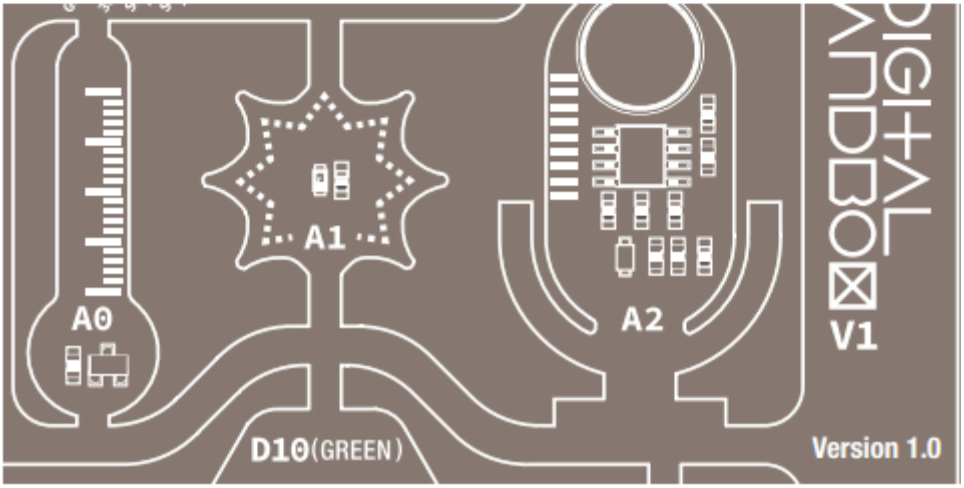


# DS GUIDE

**Din guide till SparkFun Digital Sandbox**




Denna guide är översatt av Alega Skolmateriel

www.alega.se

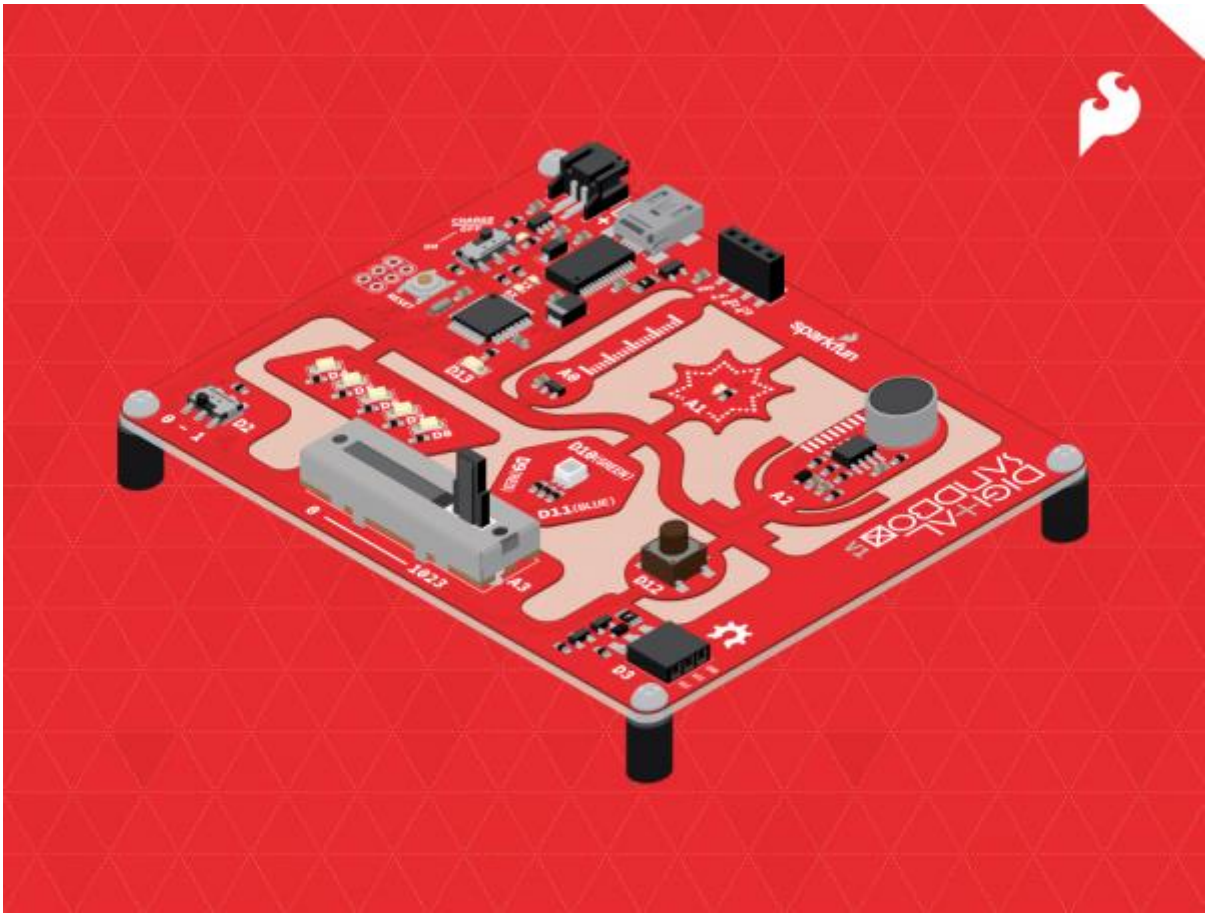
info@alega.se

## Översättarens förord

Även om DS-Guiden är översatt till svenska så är menyer i Arduino IDE och ArduBlock på engelska. Dessutom görs all textkodning med engelska uttryck, t.ex. ”*setup*” ”*loop*” och ”*if*”. Av dessa anledningar finns många engelska uttryck i denna guide som inte har kunnat översättas. Dessutom finns en hel del tekniska uttryck som kan behöva en förklaring. Vår förhoppning är att lärare eller annan vuxen kan hjälpa till med nödvändiga förklaringar.

# Välkommen till Digital Sandbox!

Digital Sandbox är ett programmeringskort som ger dig insikt inom både mjuk- och hårdvara. Den drivs av en ”mikrokontroller” som kan kommunicera med verkliga ingångar, som ljus- eller temperatursensorer, samtidigt som man styr LED, motorer och andra utgångar. Digital Sandbox är utrustad med allt du behöver för att slutföra 13 experiment, inklusive kontroll av en LED, mätning av hur höga saker är, detekterar temperaturen med mera.




Denna handledning går igenom en serie experiment/lektioner som visar hur man programmerar Digital Sandbox med hjälp av ArduBlock, ett grafiskt programmeringsspråk för Arduino.

Om du är intresserad av att programmera din Sandbox med det vanliga Arduino programmeringsspråket, kolla in vår parallella handledning: ”Digital Sandbox Arduino Companion” som du når via hemsidan www.sparkfun.com/products/12651.

## Innehållsförteckning:



Observera att experiment 14, 15 och 16 kräver Digital Sandbox Add-On – tillbehörssatsen som kan köpas separat.

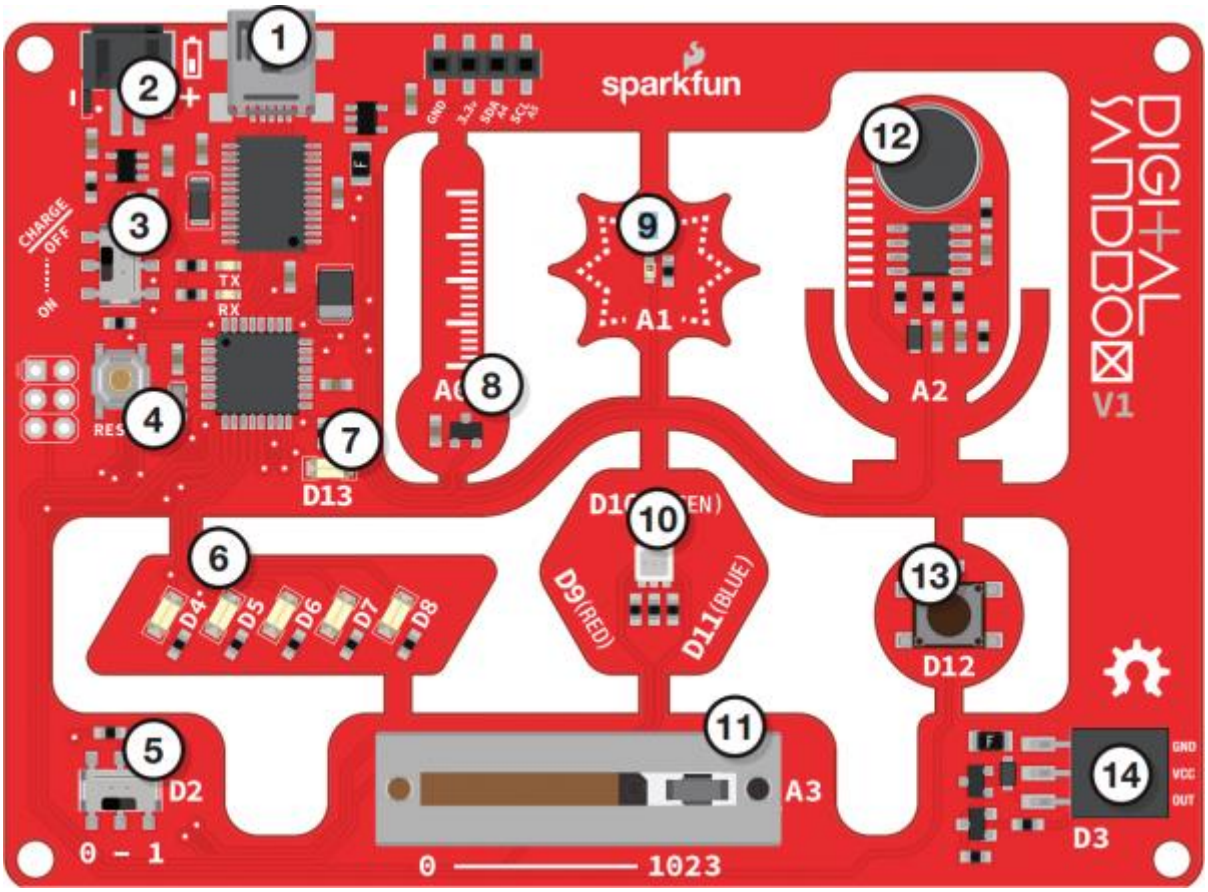
# Vad är Digital Sandbox?

Digital Sandbox (DS) är en inlärningsplattform som behandlar både programmering (mjukvara) och elektroniska komponenter (hårdvara). Den drivs av en så kallad mikrokontroller som kan samverka med verkliga ingångar, som ljus- eller temperatursensorer, samtidigt som man styr LED, motorer och andra utgångar.

Genom att koppla Sandboxen till din dator eller Mac via en USB-kabel kan den programmeras med den populära programmeringsmiljön från Arduino. För att ytterligare förenkla inlärnings-upplevelsen har vi utformat Sandboxen och den här guiden runt med hjälp av ett enkelt tillägg till Arduino för blockprogrammering som heter Ardublock.

## Uppbyggnaden av Digital Sandbox

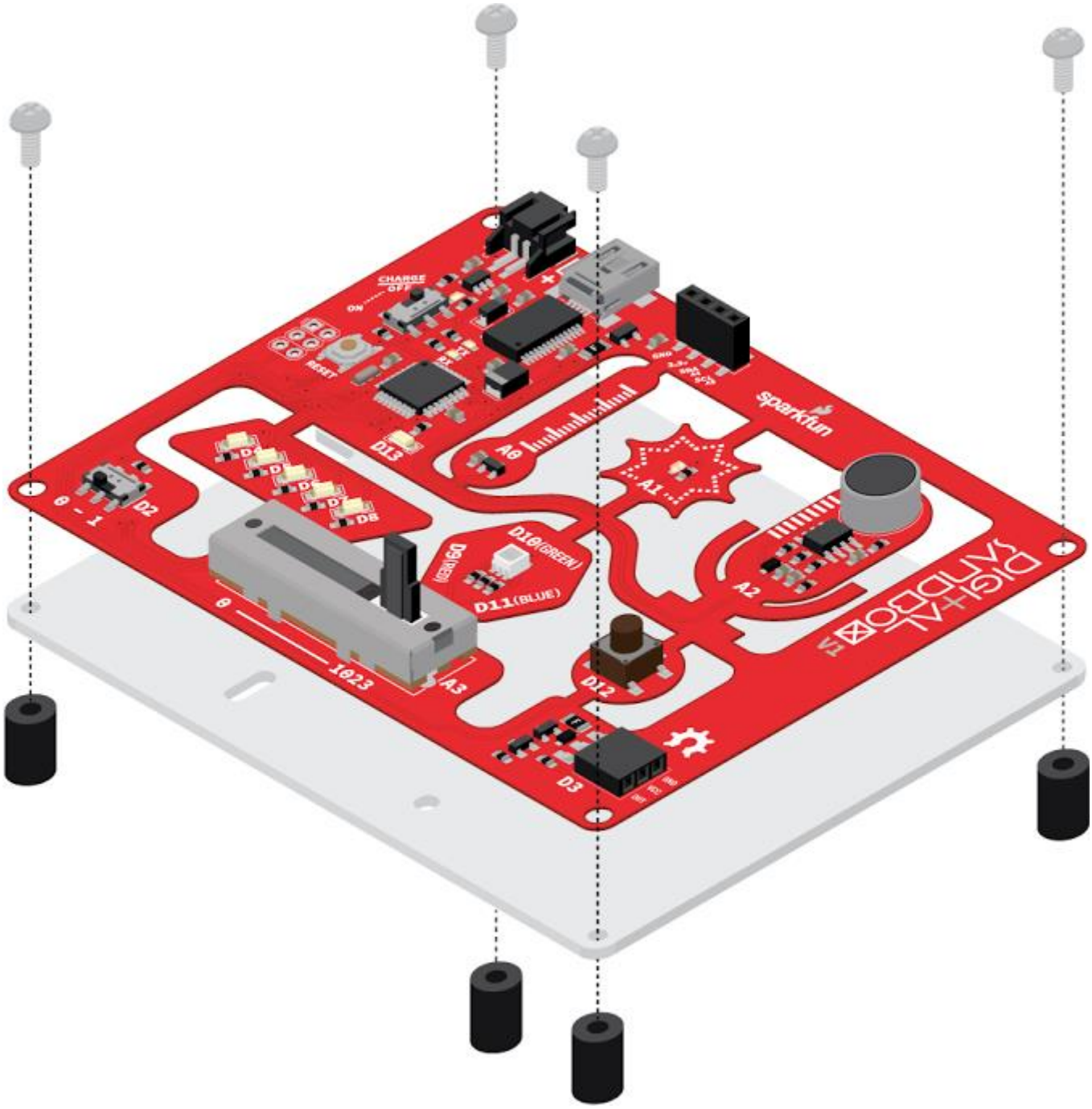
Digital Sandbox har en mängd ingångar och utgångar som är vanligt förekommande på elektroniska produkter. På nästa sida kan du se vilka som finns på DS-kortet:




1. USB Mini-B - kontakt – För att ansluta DS till din dator.
2. JST Högervinklad kontakt – Används för att ge ström till DS (om inte USB är ansluten).
3. Skjutomkopplare för laddning – Används för att ladda ett litiumpolymerbatteri som är anslutet till JST-kontakten med två stift, medan den digitala sandboxen är ansluten till en dator och skjutomkopplaren är i läge "ON".
4. Reset-knapp – Detta är ett sätt att manuellt återställa ditt DS-kort, som då kommer att starta om din kod från början.
5. Strömbrytare (Pin 2) – Omkopplare för På eller Av.
6. LED (Pins 4-8) – Använd en eller alla LED-lampor (lysdioder) för att lysa upp ditt projekt!
7. LED (Pin 13) – Används till att visa om ditt program körs korrekt.
8. Temperatursensor (Pin A0) – Mäter omgivande temperatur.
9. Ljussensor (Pin A1) – Mäter mängden ljus som registreras av sensorn.
10. RGB LED (Pins 9-11) – RGB (röd/grön/blå) LED har tre olika färgdioder som kan kombineras för att skapa många färger.
11. Skjutpotentiometer (Pin A3) – Ändra värden genom att skjuta fram och tillbaka.
12. Mikrofon (Pin A2) – Mäter hur högt något låter.
13. Tryckknapp (Pin 12) – En knapp är en digital ingång. Den kan vara antingen "på" eller "av".
14. Extrakontakt (Pin 3) – Trestiftskontakt för extrautrustning. Exempel på detta är servo, motorer och summer.

## Montera basplattan på din Digital Sandbox

Skruva fast DS-kortet på basplattan med de medföljande Phillips-skruvarna.




Skruva fast skruvarna med fingrarna för lätt borttagning senare.

# Ställa in Arduino och ArduBlock

Den här sidan hjälper dig att förbereda datorn så att du kan programmera din Digital Sandbox. Detta omfattar nedladdning och installation av programvaran, eventuell installation av drivrutiner på din dator samt inställning av Arduino-miljön för att få den att fungera med ditt DS-kort. Följ med, och du kommer att ha blinkande lysdioder på nolltid!

## Ladda ner Arduino, ArduBlock, och exempelkoden

*Översättarens anmärkning: När Arduino IDE uppdateras händer det att ArduBlock inte fungerar. För att det garanterat ska fungera kan du istället ladda ner Alegas eget mjukvarupaket som du når här:* DS_alegapack.zip

*Instruktioner finns här:* alegasstartsparkfunds.pdf

*Nu fortsätter originalet av DS-guiden:*

Först och främst måste du ladda ner lite programvara. Det finns två alternativ här: om du redan har Arduino installerad, följ anvisningarna för installing the Digital Sandbox Arduino addon. Om du inte har Arduino, följ anvisningarna nedanför.

### För dig som inte har Arduino IDE redan?

Vi har packat en anpassad version av Arduino-programvaran (version 1.6.9), som innehåller ArduBlock och Sandbox-exemplen som används i hela den här guiden. Klicka på en av länkarna nedan för att ladda ner programvaran, se till att du får den version som matchar ditt system:

- Arduino 1.6.9 - Windows (161 MB)
- Arduino 1.6.9 - Mac OS X (149 MB).

Arduino-mjukvaran kommer förpackad i komprimerat ZIP-format. När du har laddat ner ZIP-filen måste du extrahera den (packa upp den). Både Windows (använd den inbyggda extraheringsguiden) och Mac (dubbelklicka för att öppna) ska ha program för uppackning av ZIP-filer

Windows-användare ska flytta mappen arduino-1.6.9-SFEardublock till C:\Program Files (x86). Din dator kan ge dig en varning.

Mac-användare kan enkelt köra Arduino-programmet från den extraherade mappen, eller flytta den till en annan katalog (t ex Program) och kör den sedan.

Windows-användare kan flytta Arduino-mappen till en föredragen plats, till exempel "C:\Arduino". Sandboxens programexempel ingår också i en mapp som heter “Digital Sandbox Examples.”

När du har installerat Arduino fortsätter du nedan med installing the Sandbox drivers.

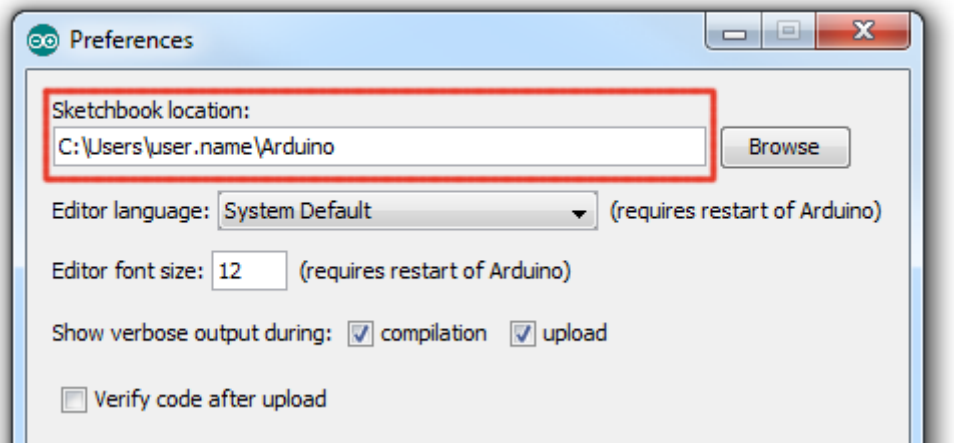
## För dig som redan har Arduino IDE installerat

Om du redan har Arduino installerat, kan du bara ladda ner Sandbox addon, som innehåller ArduBlock, drivrutinerna för Digital Sandbox-hårdvaran och exempelfilerna. Klicka på länken nedan för att ladda ner mappen Arduino addon:

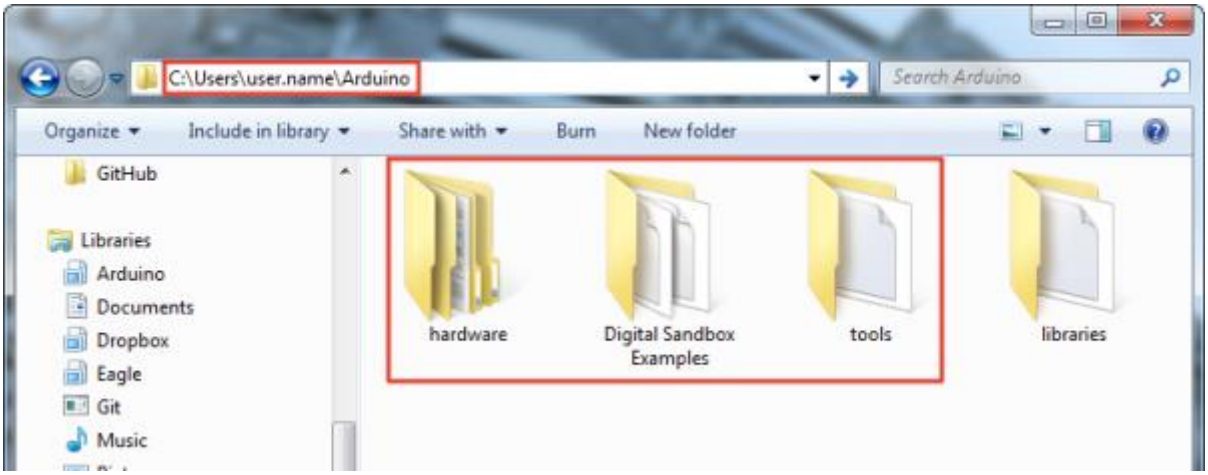
**DOWNLOAD THE SPARKFUN ARDUBLOCK AND SANDBOX ADDON**

OBS! Tillägget i länken ovan fungerar endast med Arduino versioner 1.6.0 och uppåt. Om du använder en äldre version av Arduino (1.0.6 eller tidigare) använd our previous version of the addon istället. Installationsanvisningarna kommer att vara desamma.

Alla addon-objekten finns i en ZIP-mapp. För att installera, extrahera ZIP-filen i datorns Arduino skissbokskatalog. Det här är en mapp på din dator där dina skisser (sketches) och bibliotek sparas som standard. För att hitta din skissboksmapp, kör Arduino och öppna Preferences genom att gå till Arkiv> Inställningar. Innehållet i den övre textrutan anger platsen för din skissbok. Notera den platsen och stäng Arduino.

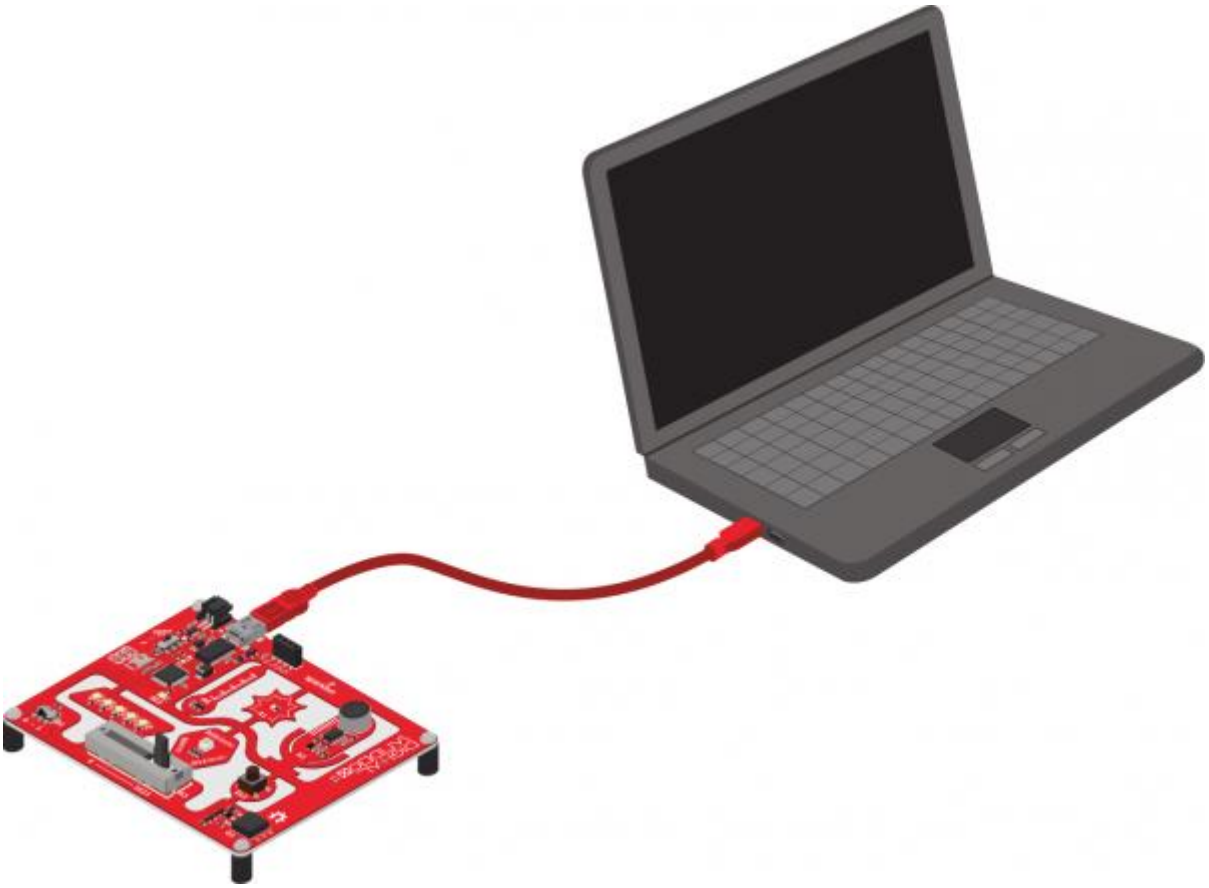



Packa därefter upp innehållet i filen Sandbox_Addons.zip till den platsen.



## Installera drivrutiner

När du har laddat ner och och packat upp Arduino-programvaran ska du koppla din Digital Sandbox till din dator.



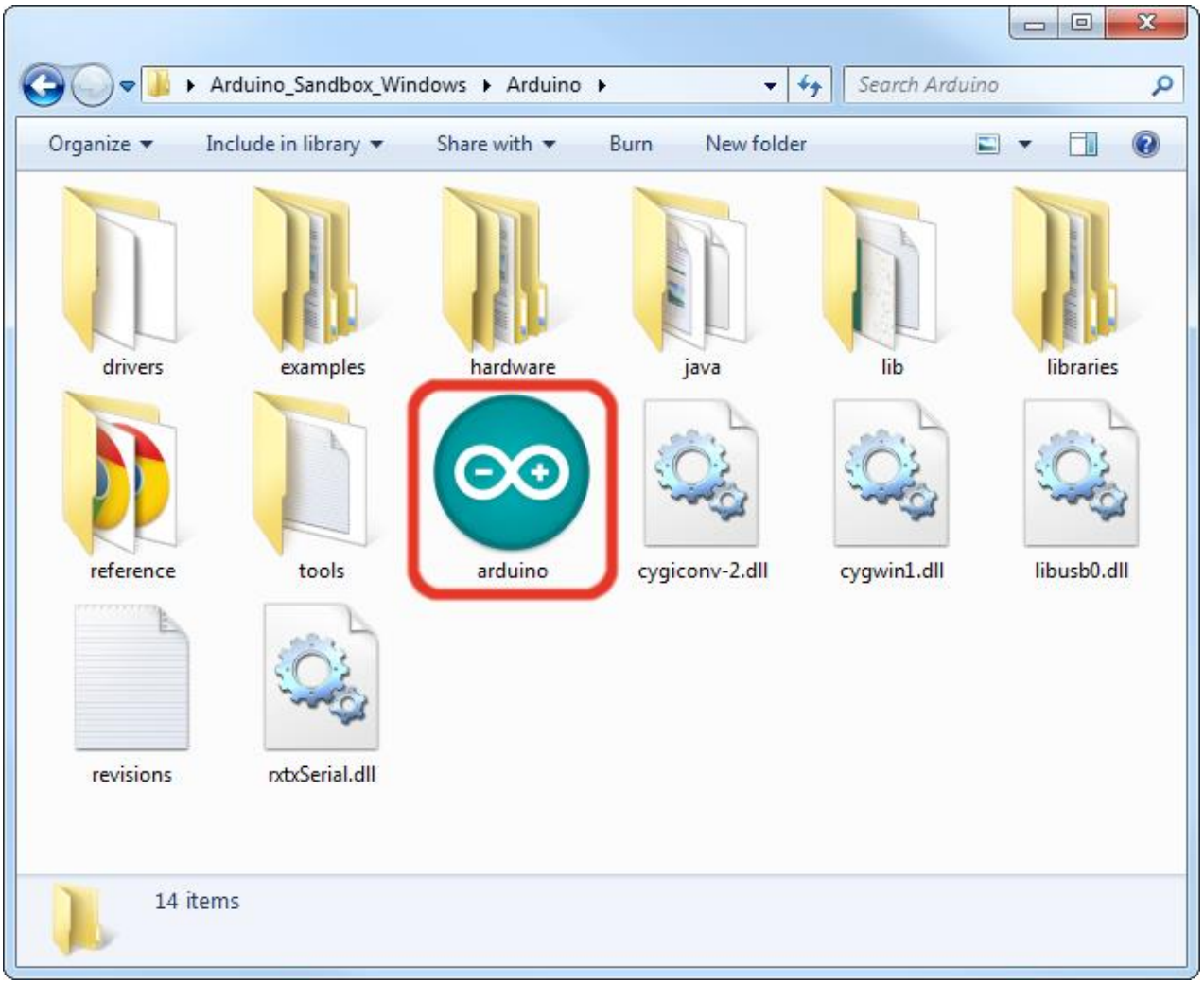
När kortet är anslutet måste du installera drivrutiner. Gå till www.sparkfun.com/ftdi för instruktioner som är specifika för ditt operativsystem.

*Översättarens anmärkning: I Windows 10 verkar det som om ftdi-drivrutinen installeras automatiskt direkt vid anslutning av DS-kortet.*

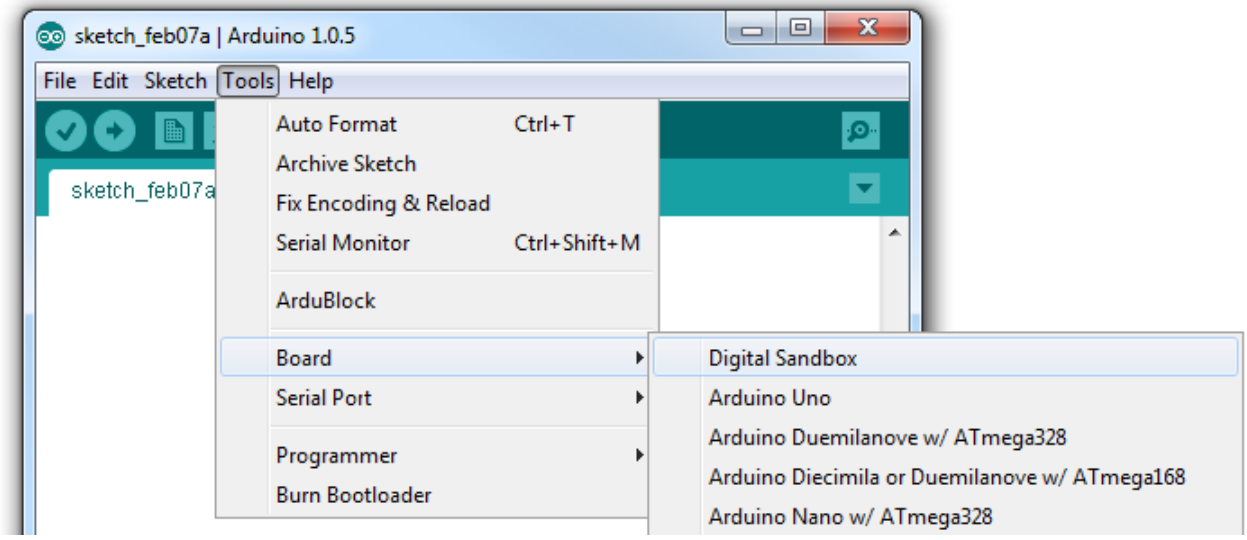
## Öppna Arduino och ArduBlock

ArduBlock är ett tillägg som nu ska finnas tillgängligt i Arduino IDE. För att köra den, öppna först Arduino-programmet. Windows-användare ska köra Arduino.exe; Mac-användare kan köra Arduino-programmet.

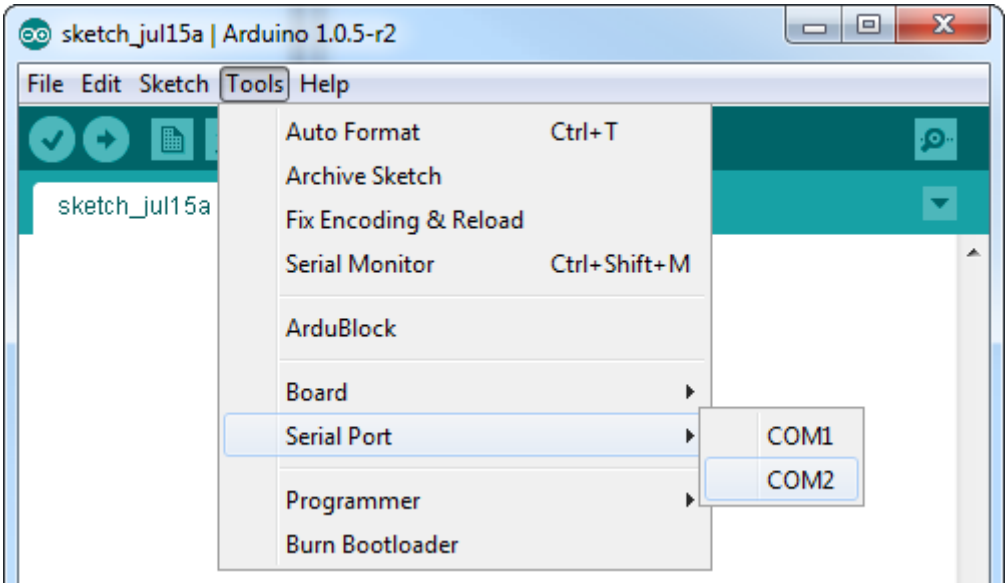
*Översättarens anmärkning: Du som installerat Alegas paket dubbelklickar på ikonen Arduino i din Arduinomapp.*




Låt oss göra några förberedelser innan du öppnar ArduBlock. Gå först till Verktyg (Tools) i menyn, gå till Kort (Board) och välj SparkFun Digital Sandbox.

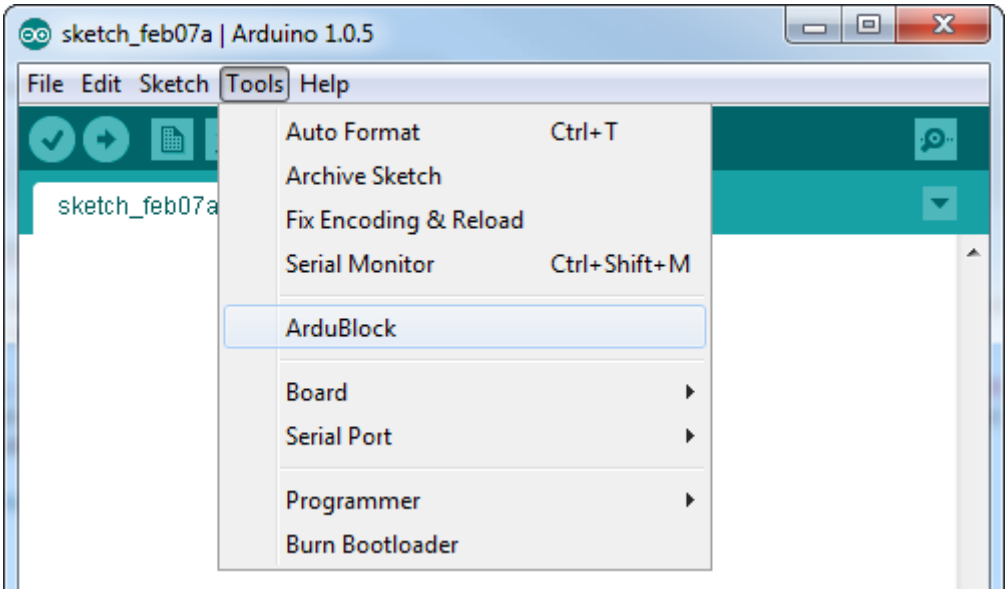


Gå sedan tillbaka till Verktyg i menyn, gå till Port (Serial Port) och välj porten som matchar ditt DS-kort.

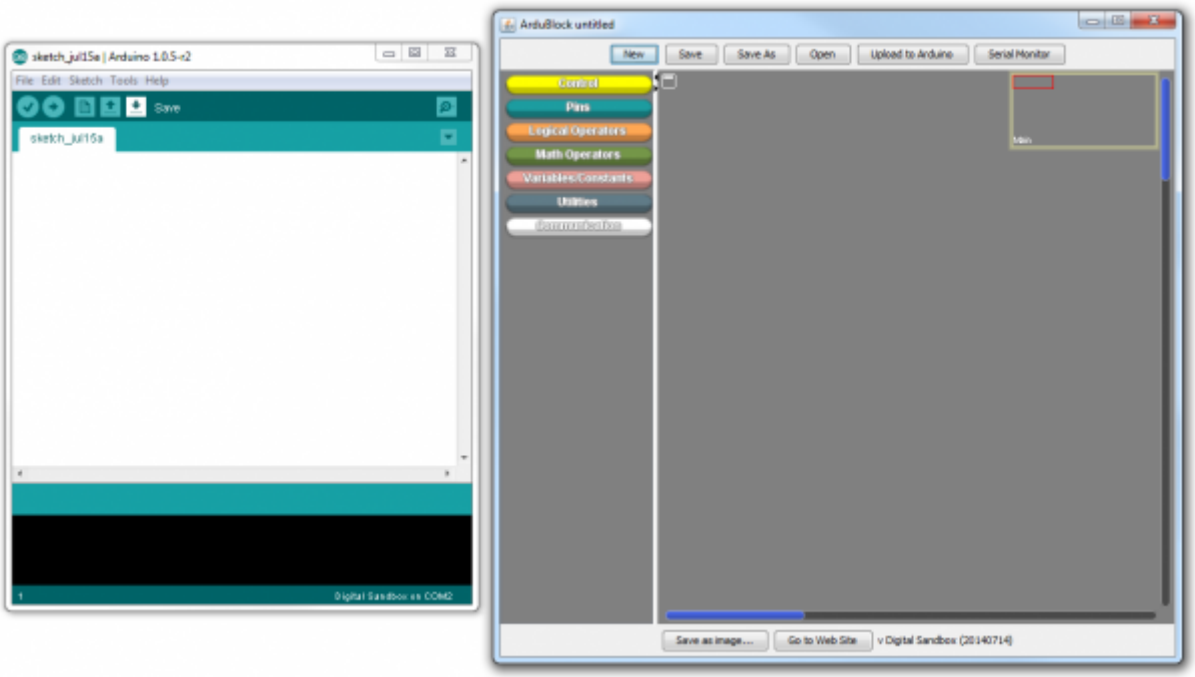



- Windowsanvändare: Detta kommer sannolikt att vara COM2 *eller högre* (COM1 är vanligtvis reserverat för seriella portar för hårdvara). För att ta reda på vilken, kan du koppla från din Sandbox och öppna menyn igen. Posten som försvinner bör vara Sandboxen. Anslut kortet och välj korrekt port.
- Macanvändare: På Mac bör det vara något med "/dev/tty.usbmodem" eller "/dev/tty.usbserial".

Slutligen, för att öppna ArduBlock, gå till Vektyg och välj ArduBlock.




Vad som öppnas då är gränssnittet till ArduBlock. Se till att Arduino-fönstret fortfarande körs i bakgrunden. Om du stänger det stängs ArduBlock också.




*Obs! Om du inte ser ArduBlock under Verktyg kan du behöva installera programmet manuellt. Gå till* this tutorial *för att få hjälp med att lägga till ArduBlock till en tidigare Arduino-installation*.

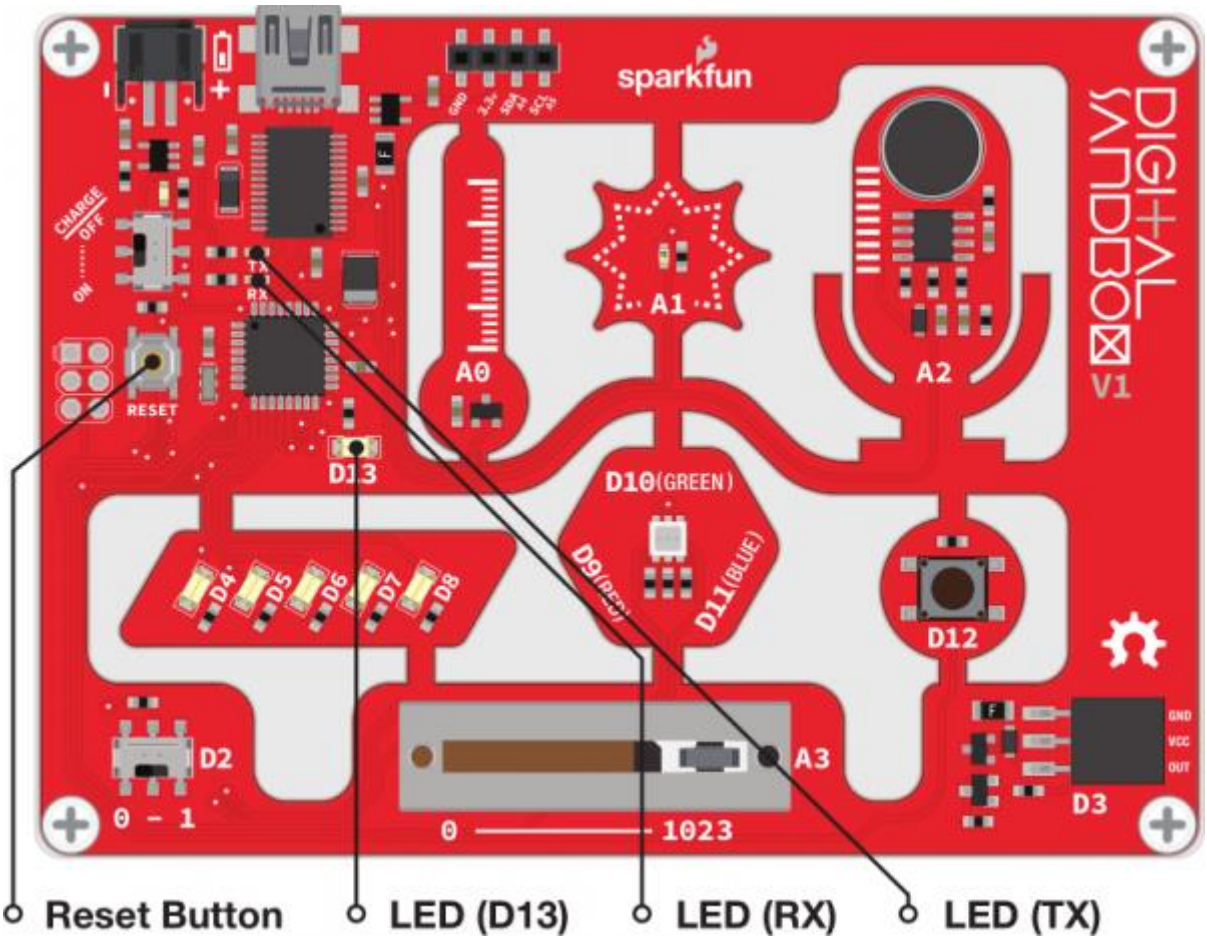
# 0: Setup, Loop, och Blink

När man möter en ny plattform är en programmerares första uppgift vanligen att skriva ett "Hello World" -program. Vanligtvis skrivs ett "Hello World" -program faktiskt ut på en skärm. Digital Sandbox ger oss inte en skärm för att skriva ut ord, men vi har LED-lampor! Underbara, blinkande, ljusa, glänsande lysdioder. Istället för att skriva ut ord, låt oss få en LED att blinka för att säga "Hello World."

## Bakgrund

Detta experiment introducerar det allmänna begreppet fysisk programmering. Ändringar som du gör i ditt program påverkar faktiskt vad som händer på Digital Sandbox-kortet.
Denna övning introducerar några av de mest grundläggande begreppen i Arduino-programmering (och nästan all annan programmering), nämligen: `setup` och `loop`.

## Aktiva delar




## Block till koden

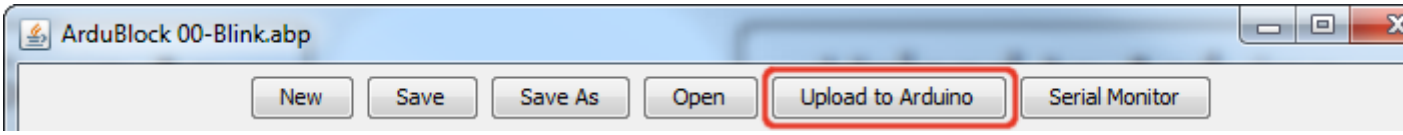
Denna programmering kräver två block (tja, egentligen tre):


- Program: Detta block är nödvändigt i alla program som görs i ArduBlock! Du kan bara ha en för varje skiss (sketch). *Program* har alltid två anslutningar för andra block - en som heter "setup" och en annan som heter "loop". Du hittar blocket under Control till vänster i ArduBlock-fönstret.
- Blink: Detta block hittar du under Pins. Det här blocket "blinkar" till ett stift (pin) på DS-kortet. Blink-blocket ger dig faktiskt *två* block till priset av en! Den innehåller också ett rosa block med nummer 13 inuti. Vi lämnar det blocket ensamma för nu, vi kommer att undersöka dess användning i senare experiment.

## Gör så här

Det finns bara två sätt att koppla ihop blocken. Du kan antingen dra *Blink*-blocket till *setup* eller till *loop*.


Dra *Blink*-blocket till *setup*, sen klickar du på ikonen *Upload to Arduino*.




Håll ögonen på din Digital Sandbox när koden laddas ner till den. Du ska se de röda och gröna RX- och TX-LED-lamporna blinka om allt fungerar. Var extra uppmärksam på vad som händer *efter* att de röda och gröna lysdioderna blinkat färdigt. Märker du något?

Flytta nu *Blink*-blocket från *setup* till *loop*, och gör om nedladdningen via *Upload to Arduino*. Märker du någon skillnad?

---

Varje Arduino-program kräver att två funktioner alltid är närvarande: *setup* och *loop*.

Setup körs en gång, i början av programmet. Dess syfte är oftast att ställa in olika funktioner som behövs under hela programmet (tills DS-kortet återställs eller man bryter strömmen). När vi fortsätter med dessa experiment får du en större förståelse av vad som måste ställas in i förväg.

Om nu *setup* ställer in din DS, så måste loop …göra en loop. Koden i det här blocket kommer att utföras i tur och ordning men ”för evigt”. När vi kommer till botten av loopen hoppar vi rakt upp till toppen och gör det hela om och om igen. Detta fortsätter tills du antingen återställer (Reset) eller tar bort strömmen (drar ur USB-kabeln).

## Upptäck mer

- Vad händer om du trycker på återställningsknappen (reset button)?
- Vad händer om det inte finns något block i antingen *setup* eller *loop* (dra bort *blink*-blocket)?
- Vad händer om du lägger till ett andra *Blink*-block till ditt program? Oavsett var du lägger det, kan du skilja på vilket av dina *Blink*-block som körs?
- Vad tror du att *13* kopplat till *Blink*-blocket är till för?

# 1: Lär dig mer om Blink

Nu fuskade vi inte direkt i experiment noll, men *Blink*-blocket var lite av en genväg. Vad händer om du vill få ett snabbare blinkande? Eller ändra hur länge det är på hur länge det är av? Eller till och med blinka med något annat än den lilla gula LED-lampan?

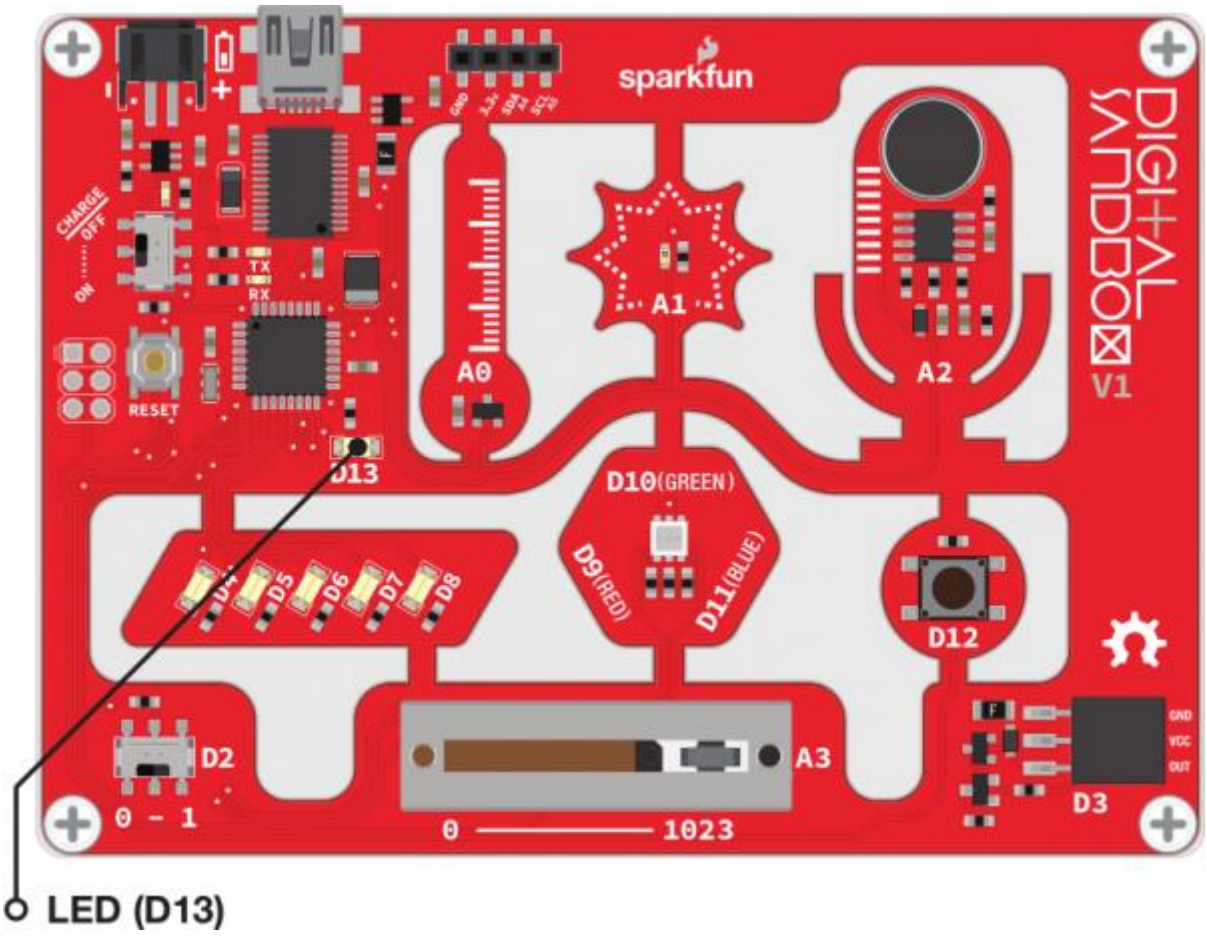
## Bakgrund

Detta experiment tränger lite djupare i styrningen med *Blink*-blocket. Vi kan skräddarsy en LED-blinkning med en kombination av två block – Set Digital Pin och Delay Milliseconds.

Här introduceras begreppet digital utgång. LED-lamporna på din DS är “utgångar” (outputs), eftersom det är något som DS-kortet *utför*.

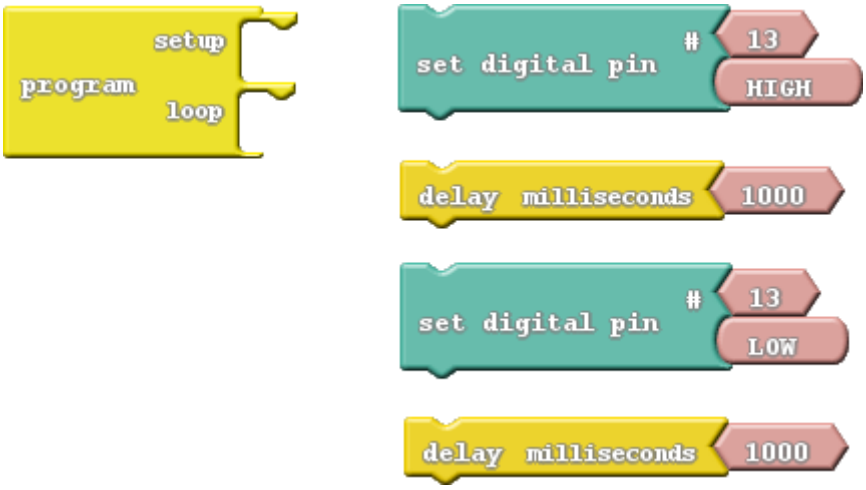
Termen "digital" betyder att utsignalen endast kan anta ett av två tillstånd: PÅ eller AV. Vi kan också se dessa tillstånd som motsatserna HÖG (HIGH) / LÅG (LOW) eller 1/0. När en utgång som är ansluten till en LED är HIGH, lyser lysdioden. När den är LOW, släcks lysdioden.

## Aktiva delar



## Block till koden

Här är den uppsättning block vi ska använda för att koda programmet:




Bortsett från *program*-blocket (som ska vara med i varje program) finns två nya block:

- Set Digital Pin: Detta block sätter en utgång till antingen HIGH eller LOW, så att det kan användas för att slå på eller av LED-lampan. Hitta det här blocket under Pins-ikonen. När du drar det här blocket, innehåller det ett par rosa block som innehåller "13" och "HIGH." Låt oss bara behandla det nedre rosa blocket just nu.
    - HIGH/LOW-block: Om du sätter muspekaren över det här blocket visas en nedrullningsbar pil. Klicka på pilen och du kan ändra värdet på blocket till antingen "HIGH" eller "LOW." Det här avgör vilken av de två tillstånden du ställer in den digitala utgången till.
- Delay Milliseconds: Digital Sandbox kör kod så snabbt att vi ibland måste sakta ner programmet genom att ställa in en viss fördröjning. Detta block kommer att stoppa Sandboxen från att göra någonting för ett visst antal millisekunder. Det går 1000 millisekunder (ms) på en sekund, så fördröjningen för *1000 ms* kommer att stoppa utförandet i *en sekund*. Hitta det här blocket under Control-ikonen.

## Gör så här

Dra och sätt samman blocken *Set Digital Pin* och *Delay Milliseconds* så att de växlar - blågrön, gul, blågrön, gul. Dra sen de fyra blocken till loop-delen av *Program*-blocket. Tryck sen på Upload the code.

Du borde känna igen vad du ser på ditt DS-kort, men den här gången har du kontroll över blinkhastighetn! Justera värdet i Delay Milliseconds-blocken. Vad händer om du förkortar ”förseningarna”? Vad händer om de två förseningarna inte är sätts till samma tid?

Digital Sandbox fungerar med hastigheten (frekvensen) 8MHz - det finns en slags klocka som markerar åtta miljoner gånger per sekund. Det betyder att det kan köra miljontals rader av kod varje sekund. Utan några förseningar i programmet skulle den digitala utgången sätta av och på så fort att du inte skulle kunna se vad som händer.

## Upptäck mer

- Hur korta förseningarna kan du ha i programmet och ändå se ett blink? Tio millisekunder? En millisekund?
- Vad händer om du tar bort blocket *Delay Milliseconds* från programmet?
- När du letade efter blocket *Delay Milliseconds*, kanske du också såg blocket Delay Microseconds. Vad händer om använder det istället?

# 2: Multi-Blink

Grupper av LED används ofta för att skapa stora utomhusskyltar och mycket annat eftersom de både lyser bra och är energieffektiva. Medan vi inte har miljoner LED-bildpunkter som en stor skärm kan ha så kan vi ändå skapa några roliga mönster med Digital Sandbox.

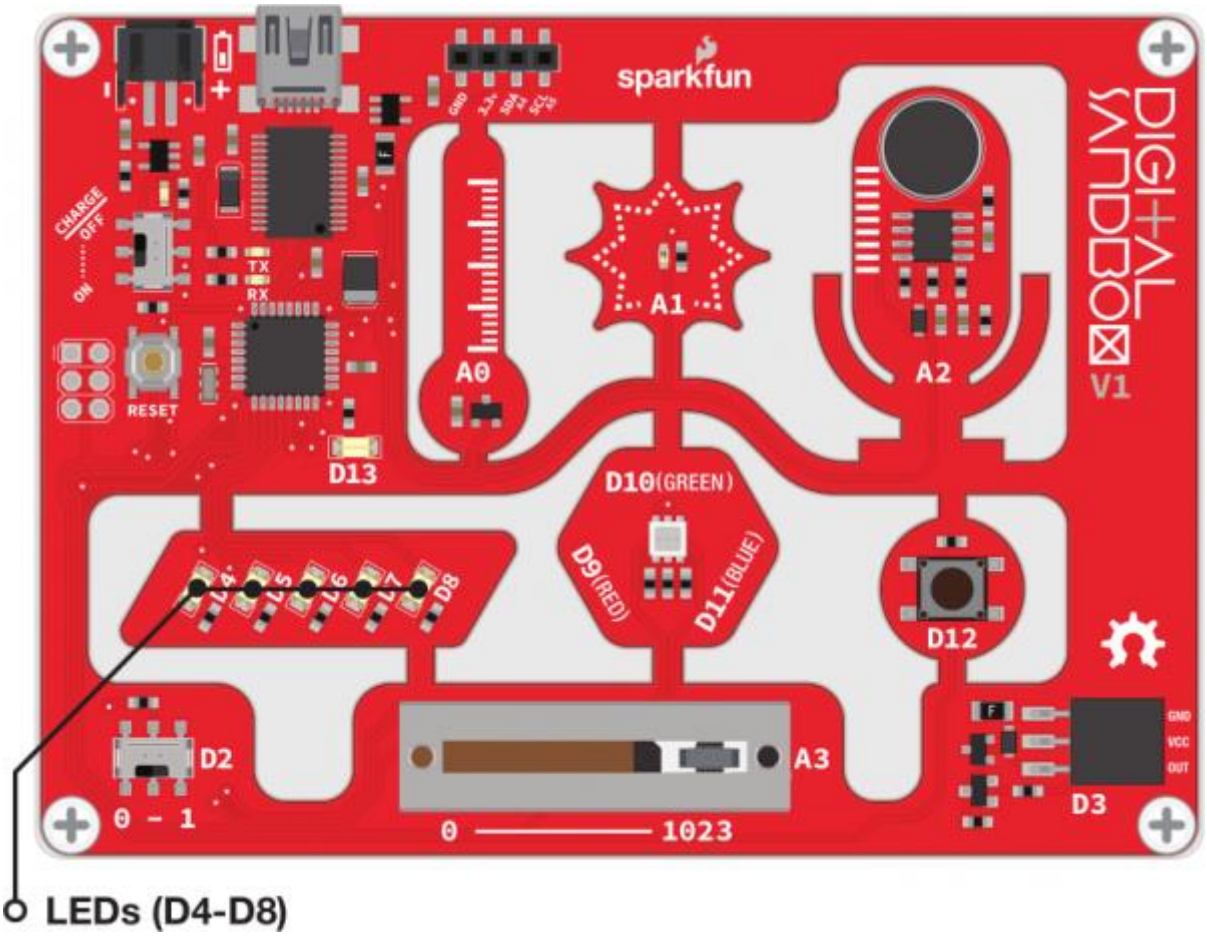
## Bakgrund

I det här experimentet utforskar vi betydelsen av pins. Varje LED (liksom de andra ingångarna och utgångarna på Digital Sandbox) är ansluten till ett specifikt stift (pin) på Sandboxens mikrokontroller.

Alla pins har egna unika nummer, och varje ingångs- eller utgångskomponent på DS-kortet är märkt med pin-numret det är anslutet till, det vill säga *D2, D4, D11, A1*, och så vidare står angivet bredvid varje LED, strömbrytare och sensor.

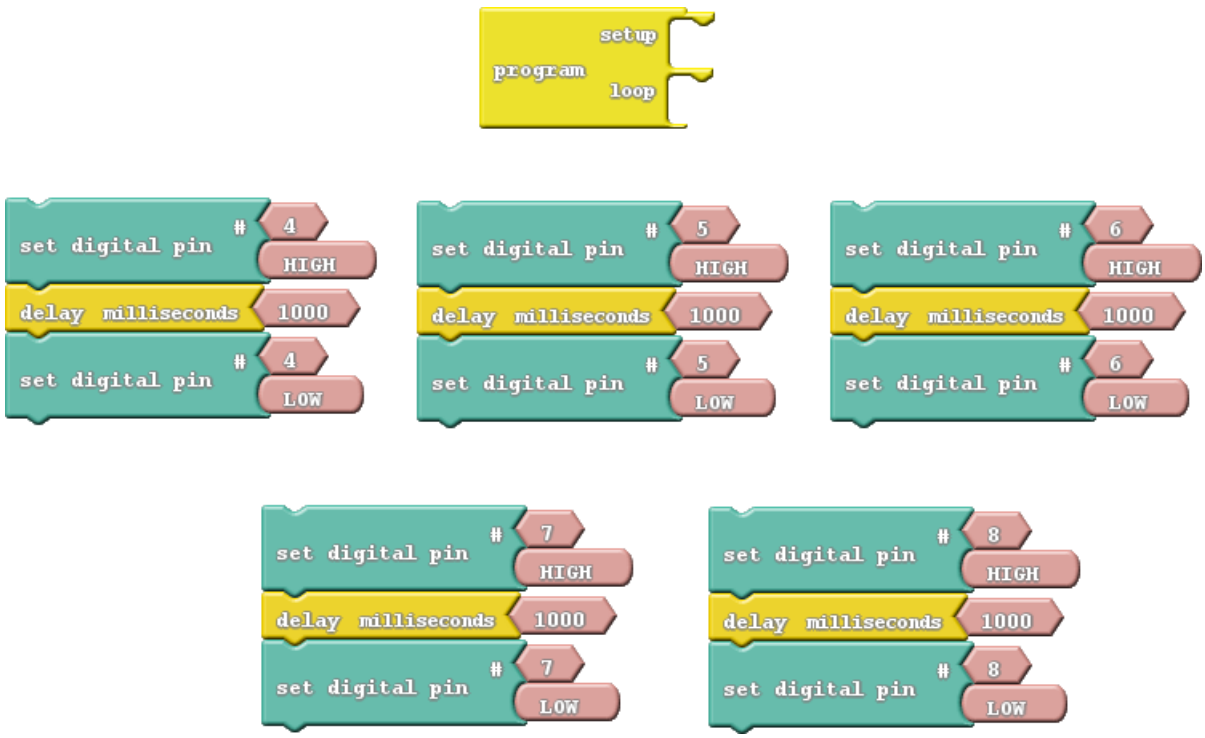
Varje pin kan styras separat; t ex pin 4 kan ställas på HIGH samtidigt som pin 5 ställs på LOW. Några pins (som vi ska se senare) har speciellt kraftfulla möjligheter, men varje pin kan åtminstone tjänstgöra som en digital ingång eller utgång.

## Aktiva delar



## Block till koden

Wow! En formidabel blockexplosion! I detta experiment krävs totalt sexton block:




Istället för att använda ett nytt block ska vi justera värdet på *Set Digital Pin* – numret på den pin som ska programmeras. Detta värde anger vilken av DS-kortets pin vi ska byta till.

## Gör detta

I vårt oavslutade exempel är blocken ordnade i grupper om tre. Varje grupp börjar med att ställa en pin på HIGH, sedan förseningar i en sekund och sedan tillbaka till LOW. Observera att varje grupp av tre växlar till en annan pin, allt från pin 4 till pin 8. Dra de fyra grupperna av block I tur och ordning till *loop*, sen ladda ner och se resultatet.

Om du tycker att dioderna blinkar för långsamt, försök att justera fördröjningarna för att göra blinkningarna snabbare. Du kanske vill ändra pin-ordningen för att justera i vilken ordning lysdioderna blinkar.

## Upptäck mer

- Prova att lägga till flera block för att skapa nya häftiga ljusmönster.
- Försök att slå på mer än en LED i taget. Slå på dem alla men akta så du inte bländas!

# 3: Dimbelysning (det svåra sättet)

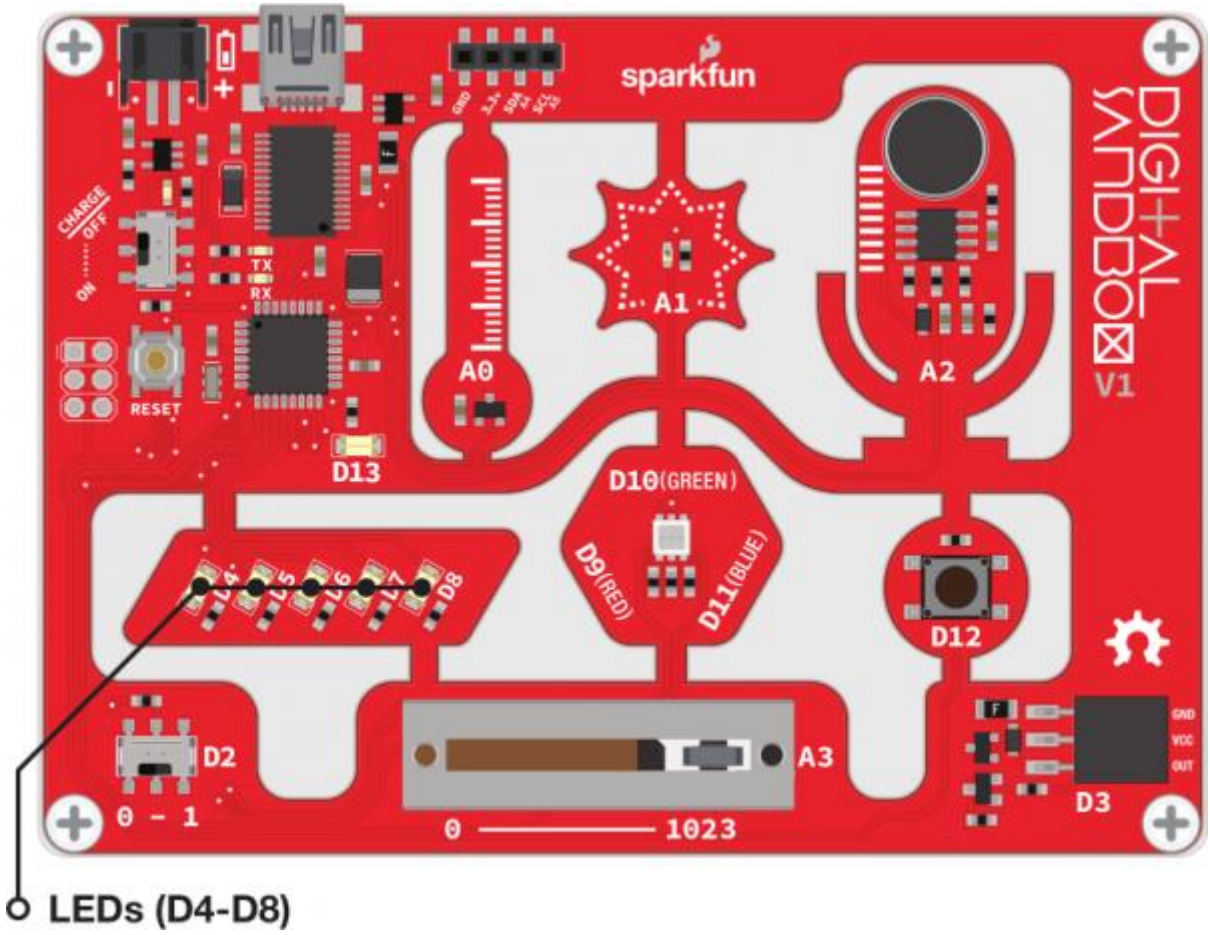
Usch! De vita lysdioderna är bländande ljusa! Finns det något sätt att dämpa dem? Vi rekommenderar att du lägger en bit papper över LED-lamporna i detta experiment ... eller bär solglasögon.

## Bakgrund

Kom ihåg att Digital Sandbox arbetar snabbt. Den kan slå på och av en LED miljoner gånger varje sekund. Vad händer om vi blinkar LED supersnabbt, men gör det så att tiden lysdioden är av är längre än tiden den är på? Detta kallas pulsbreddsmodulering (pulse-width modulation, PWM), ett verktyg med en mängd olika applikationer, inklusive att dimma lysdioder eller andra lampor.

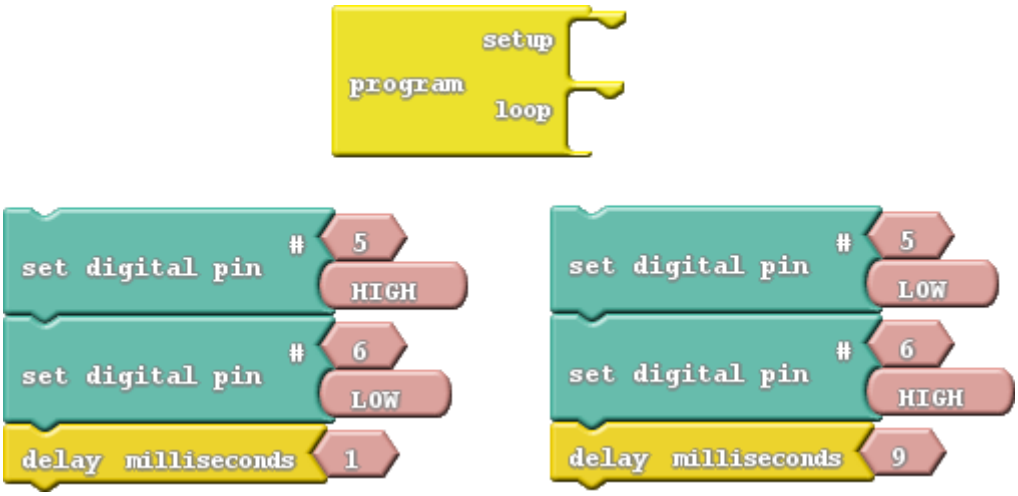
I det här experimentet utforskar vi PWM den svåra vägen genom att koda DS manuellt.

## Aktiva delar



## Block till koden

Vi använder en liknande uppsättning block:




Notera hur lång varje fördröjning är och vilka pins som är på respektive av i varje grupp.

## Gör detta

Dra de två grupperna i ordning till *loop*-sektionen och ladda sedan ner koden till DS-kortet.

Efter laddningen, ta en titt på LED-lamporna anslutna till pin 5 och 6. Kan du upptäcka skillnaden mellan de två? D6-LED-lampan ska se svagare ut i jämförelse med D5. Det beror på att D6 är inställd på att vara LOW 90% av tiden och HIGH endast 10% av tiden. Det blinkar så snabbt att du inte märker det men blinkandet skapar ändå en dimningseffekt.

Vad händer om du byter de två *Delay Millisecond*-blocken? Vad händer om du ändrar värdena i var och en av fördröjningsblocken (försök att hålla summan av fördröjningstiderna till cirka 10 ms)?

## Upptäck mer

- Hur lång tid kan du sätta förseningarna till innan du börjar se att lysdioderna blinkar?
- Försök att jämföra båda LED-lamporna med en vanlig lysdiod. Lägg till ett *Set Digital Pin*-block till inställningen och koda den att tända D4 LED HIGH. Kan du skilja mellan D4, D5 och D6?
- Vad händer om du lägger till något annat i loop-sektionen, som exempelvis din kod från experiment 2?

# 4: Dimbelysning (det lätta sättet)

Manuell PWM är svår, och det lämnar inte plats för något annat i programmet. Varför kan vi inte låta DS-kortets mikrokontroller sköta detta? Den är smart nog för det ... eller hur?

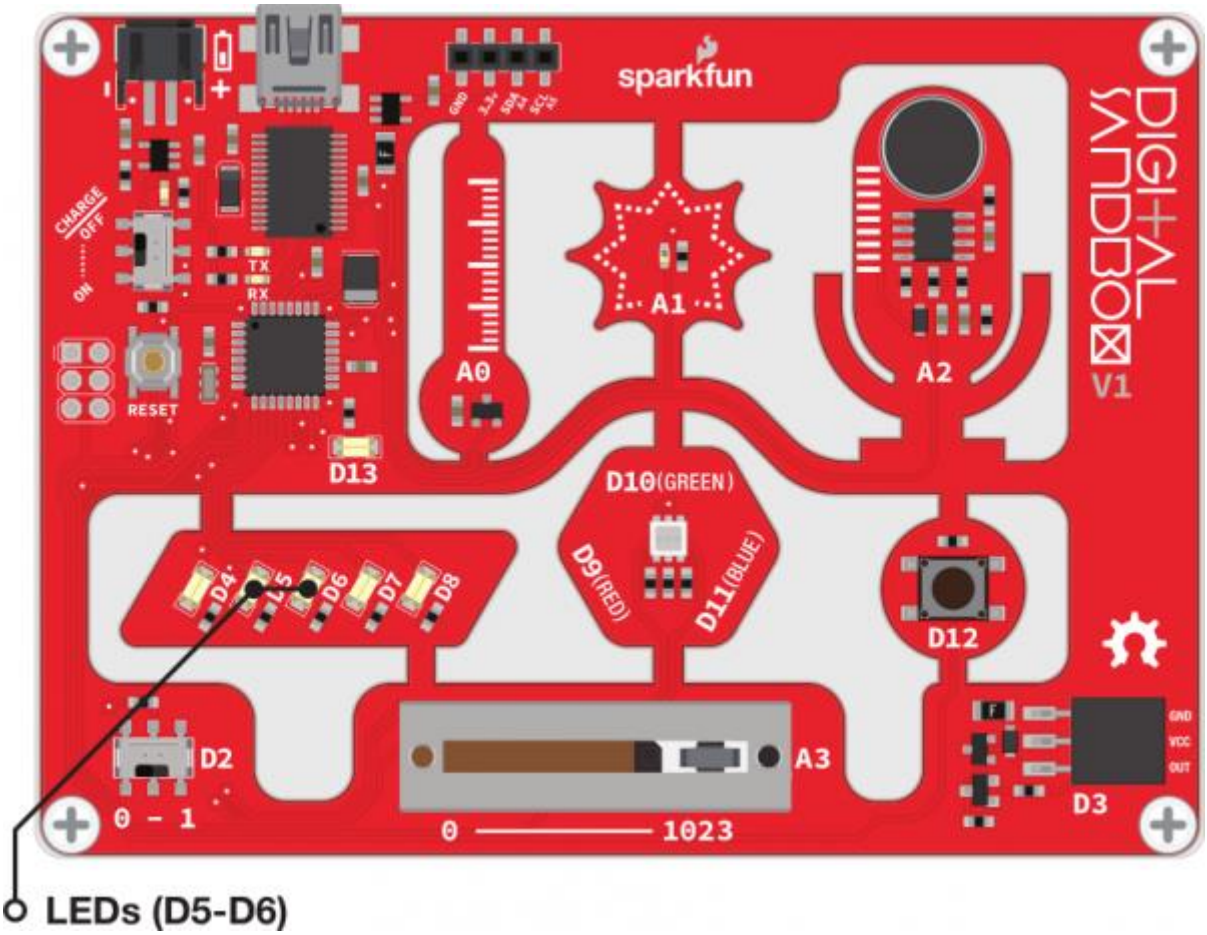
## Bakgrund

PWM är ett så populärt verktyg så att många mikrokontroller använder speciell hårdvara så att de utan problem kan växla värden på olika pins samtidigt som de gör något annat. Vi kallar den här PWM-baserade utgången för en *analog* utgång.

Till skillnad från *digitala* utgångar, som endast har två möjliga värden, har *analoga* utgångar ett stort antal möjliga värden. På Sandboxen kan vi analogt sätta 256 olika värden. Om vi ställer in en analog utgång till noll, är det som att ställa in en pin till LOW, och 255 är som att ställa in en pin till HIGH. Alla värden däremellan ger ett resultat som varken är HIGH eller LOW - det ligger någonstans däremellan.

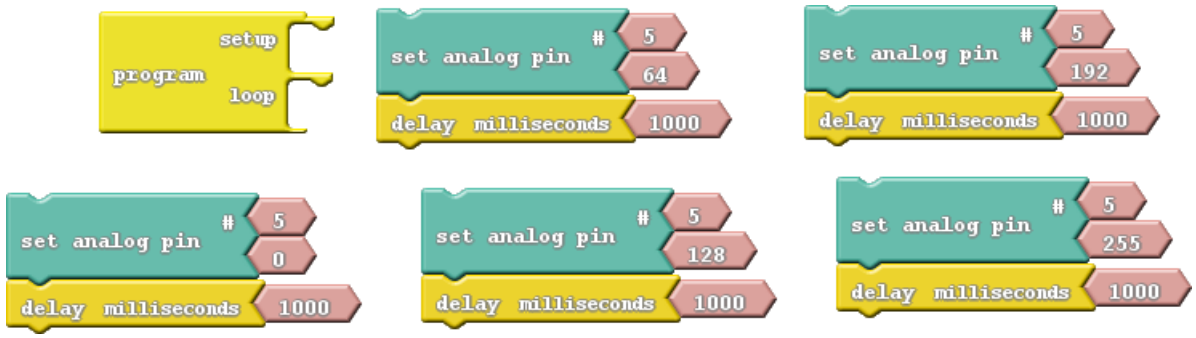
Analoga utgångar verkar ju bra - varför kan vi inte använda sådana hela tiden? Tyvärr har inte alla pins särskilda PWM-egenskaper. Endast *pin 3, 5, 6, 9, 10* och *11* kan användas som analoga utgångar.

## Aktiva delar



## Block till koden

Ny blockvarning! Fast det kan se likadant ut som förut, använder vi *Set Analog Pin* den här gången, istället för dess digitala motsvarighet:




- Set Analog Pin: Det här blocket liknar *Set Digital Pin*-blocket. Vi anger fortfarande vilken pin som ska kontrolleras, men istället för ett begränsat digitalt LOW eller HIGH, får vi välja ett tal mellan noll och 255 för utgången. Du hittar även det här blocket under Pins.

## Gör detta

Dra blocken i rätt ordning till loop-sektionen. Ordna dem så att de analoga värdena går från noll längst upp till 255 längst ner. Ladda sedan programmet till DS-kortet!

Lysdioden på pin 5 ska gå igenom fem olika nivåer av ljusstyrka (inklusive helt på och helt av). Kom ihåg att inställningen av analog utgång till noll släcker lysdioden, och 255 är som att ställa in den till HIGH.

Försök att lägga till analog kontroll av LED 6 till programmet. Du kan skapa samma effekt från det senaste experimentet, med bara två rader av kod (och du kan låta din DS köra annan kod medan LED-lamporna är kvar i sina dimmade tillstånd!).

## Upptäck mer

- Vad är det minsta analoga värdet du kan ställa in lysdioden på och fortfarande se att den lyser?
- Varför tror du att det finns just 256 möjliga analoga utgångsvärden? Varför inte 100, 250 eller 300? (Ledtråd: $2^8$).

# 5: Blandade färger

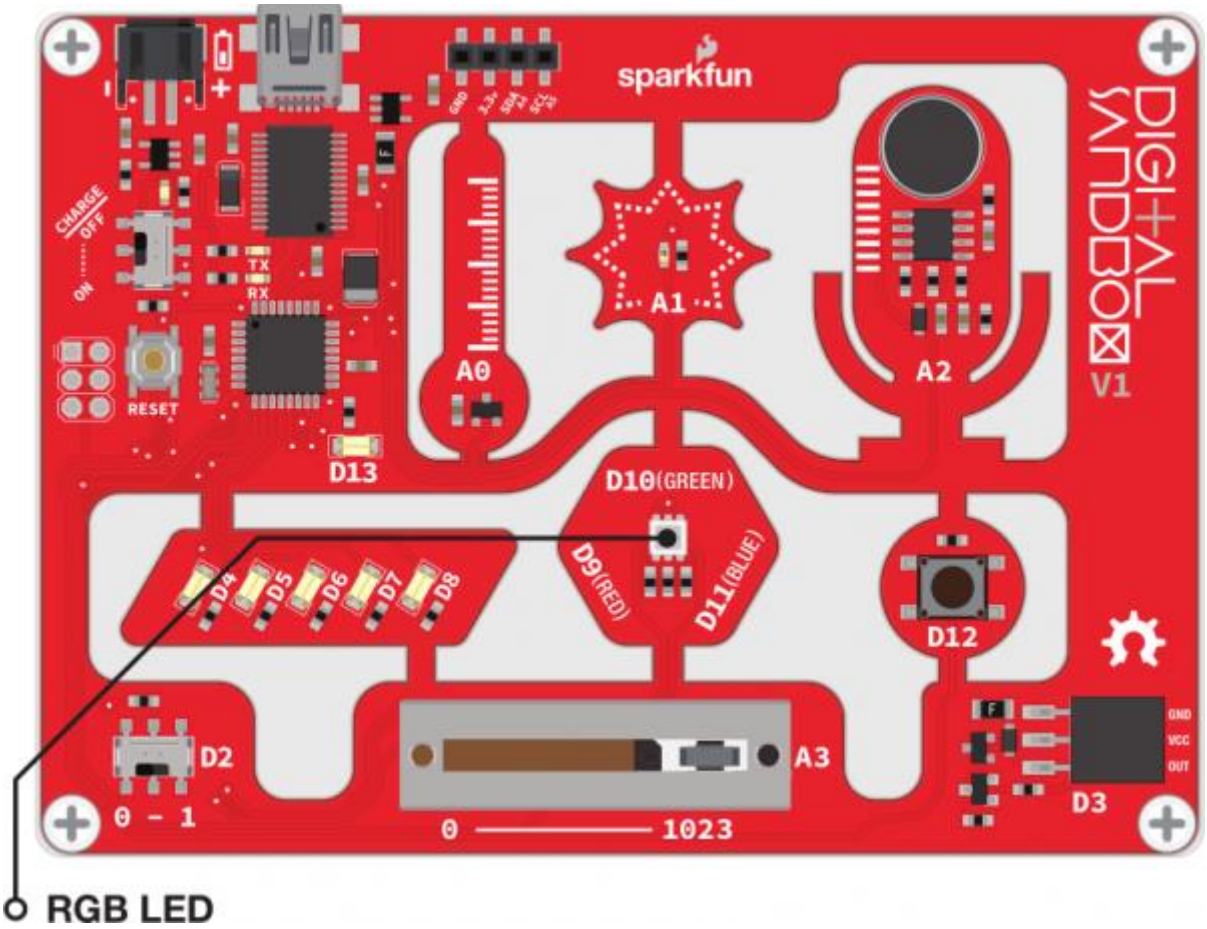
Blä ... vitt. Så tråkigt. Dags att lägga till lite färg på din Digital Sandbox. Genom att kombinera analog utgång med en RGB-LED, kan vi blanda olika mycket av rött, grönt och blått för att skapa en regnbåge av färger!

## Bakgrund

Under Bildlektionerna har du förmodligen lärt dig om grundfärger och hur du kan blanda dem för att få någon annan färg. Medan de grundfärger som du kanske känner till är rött, gult och blått, så är de inom elektronik (till exempel bildskärmar) rött, grönt och blått.

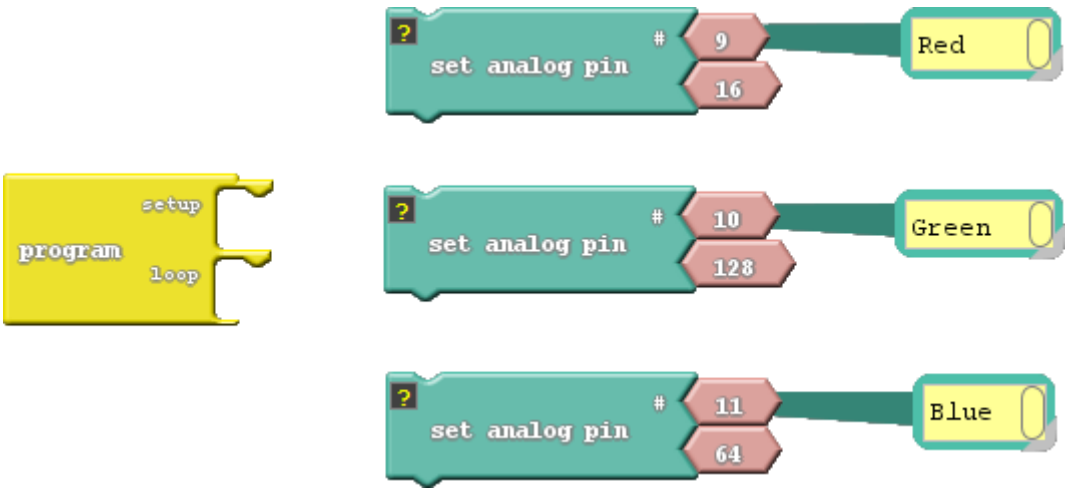
Genom att välja olika analoga nivåer för våra grundfärger kan vi blanda dem för att skapa vilken annan färg som helst som vi vill ha. Behöver du gul? Blanda grönt och rött. Lila? Rött och blått. I det här experimentet kombinerar vi det vi har lärt oss om analoga utgångar för att visa anpassade färger på DS-kortet.

## Aktiva delar



## Block till koden

För den mest grundläggande koden till blandning av RGB-färgerna är detta allt vi behöver:




I exemplet har vi lagt till *kommentarer* till vart och ett av *Set Analog Pin*-blocken. Kommentarer har ingen effekt på själva koden, men de hjälper till att göra koden mer läsbar för dig och för andra som läser den i efterhand. Med kommenterar ser vi enklare vilka pins som går till vilka färger utan att titta på DS-kortet.

Du kan lägga till kommentarer genom att högerklicka på ett block och välja " Add Comment". Visa eller dölj kommentarer genom att klicka på "?".

## Gör detta

Dra de tre *set analog pin* så de sätts ihop, antingen till *setup* eller *loop*. Detta ställer det röda värdet till 16, gröna till 255 och blåa till 128. Vilken färg tror du att det kommer att bli? Ladda upp för att ta reda på om du fick rätt! (Om det är svårt att se vilken färg det är, lägg en bit papper över RGB-lysdioden.)

Ändra de analoga värdena för att skapa egna färger. Vad sägs om lila, eller apelsin eller lax? Du kan ta det ännu längre genom att lägga till förseningar och blinka olika färger för att göra ljusspel.

## Upptäck mer

- Blanda färgerna för att göra din favoritfärg. Eller, om din favoritfärg är röd, grön eller blå, försök att göra den som minst av allt är din favoritfärg.
- Gör ett trafikljus som blinkar grönt, sen gult och sist rött och sedan börjar om.

# 6: Lagra värden med variabler

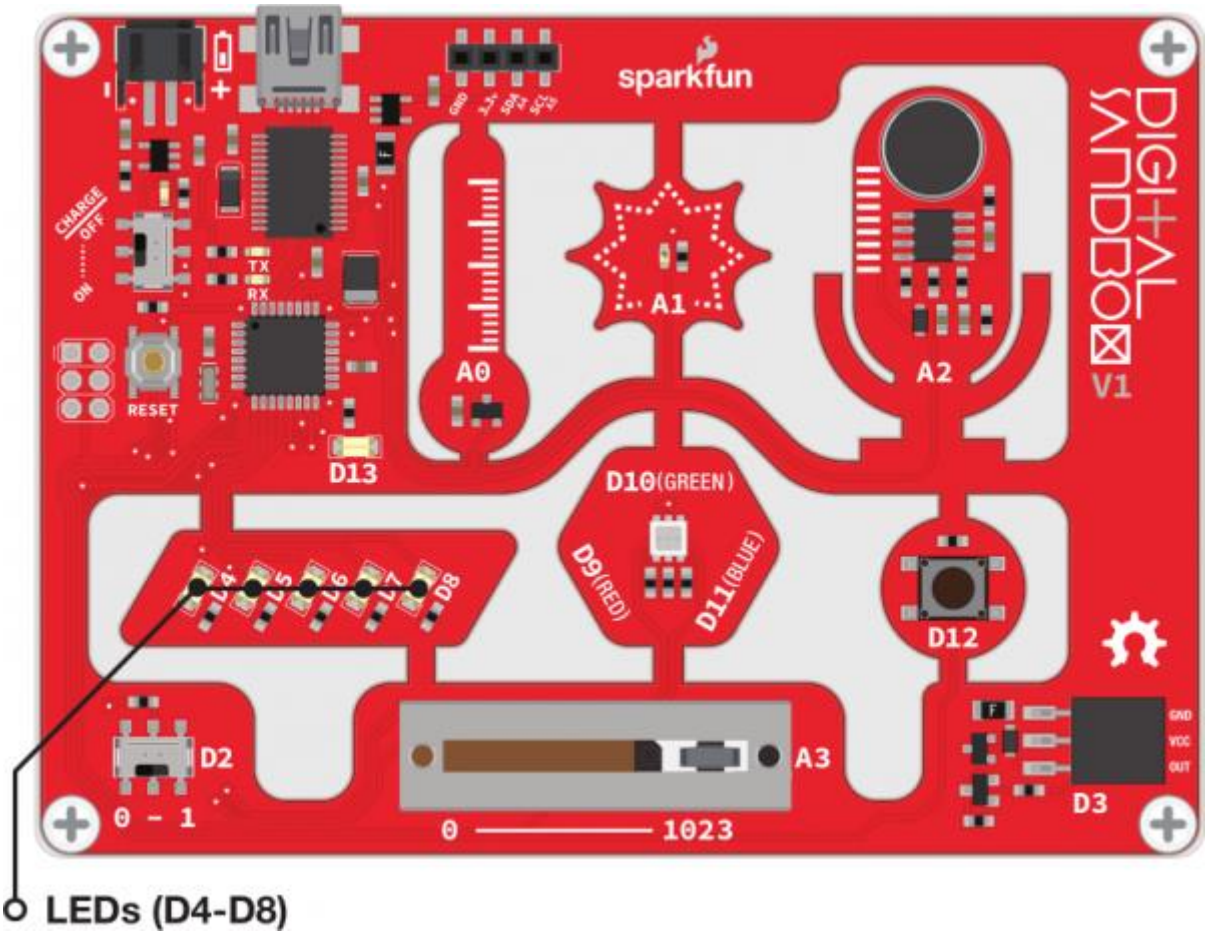
Den stegvisa dimningen i experiment 4 klarade uppgiften, men vi missar ju massor av mellanliggande värden på ljusstyrkan! Hur gör vi så att ljuset från LED-lampan justeras bättre? Man kan använda 256 olika värden på *Set analog pin*-blocken, eller du kan använda ett block, genom att utnyttja variabler.

## Bakgrund

Variabler är som förvaringsbehållare för värden. Vi kan ange ett tal i en variabel, och antingen återkalla den och använda den, eller manipulera den för att ändra det värde som lagras. Överallt där du använder värden (som "0" eller "255") kan du istället använda en variabel.

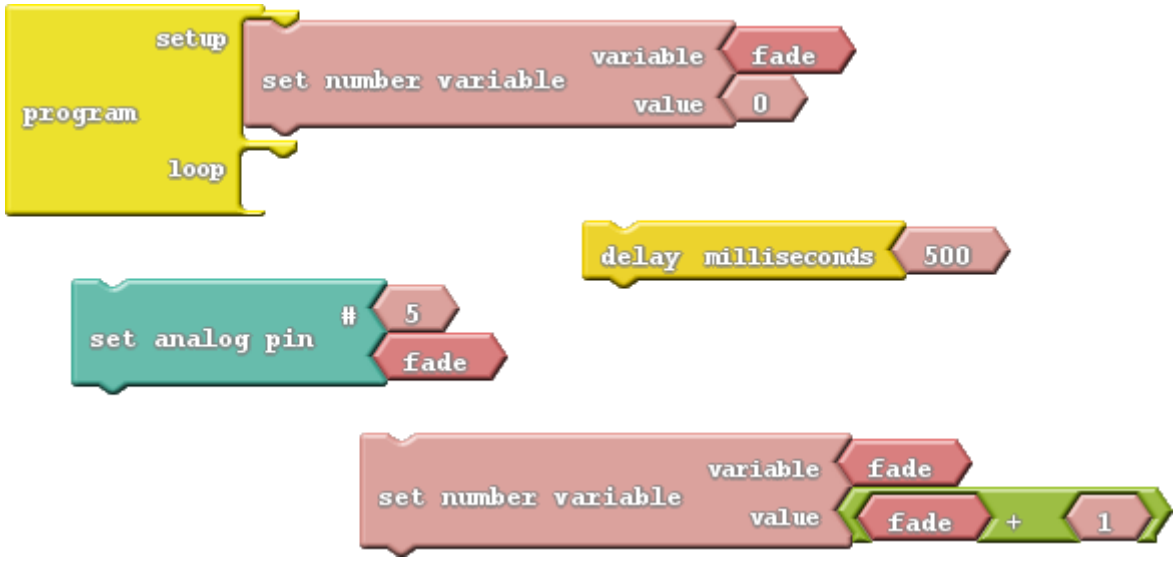
Det finns några regler när det gäller att skapa en variabel. Den kan heta nästan vad som helst, men den måste börja med en bokstav, och den kan inte ha mellanslag (använd "_" istället). De är skiftlägeskänsliga, så en variabel som heter "blekna" är inte samma variabel som "Blekna." Försök att göra variablerna korta, men använd beskrivande ord för att göra koden mer läsbar.

## Aktiva delar



## Block till koden

Tack vare variablerna, så är detta alla block vi behöver för att skapa en smidigare dimning:




Det finns några nya block att bekanta dig med den här gången:

- Namnge Number Variable: Dessa block har ungefär samma storlek och form som de nummerblock som vi har använt. Men i stället för att skriva ett nummer i dessa block skriver du in namnet på din variabel (se till att det stavas på samma sätt som du vill referera till). Du hittar dessa block under Variables/Constants ikonen till vänster.
- Set Number Variable: Detta block, som också finns under Variables/Constants, används till att ge en variabel ett specifikt värde. Två block fasts på detta – ett variable name överst, och värdet (value) du vill att variabeln ska ha. Värdet kan vara ett vanligt tal, en annan variabel eller resultatet av en matematisk uträkning.
- Math operator-blocket: Om du klickar på Math Operators-ikonen, och tittar på de fyra första inmatningarna, bör du se några mycket välkända symboler: +, -, × och ÷. Dessa *matematikoperatorer* kan användas för att göra uträkningar på ett par variabler eller siffror, eller olika kombinationer av dessa.

## Gör detta

Lägg först till ett Set Variable Number block, som kommer att innehålla en variabel utan namn och ett värde. Klicka på *number variable name* och skriv “fade”. Variabeln “fade” kommer att hålla ordning på ljusstyrkan hos vår LED. Blocket *Set Variable Number* som sitter vid *setup* ska ha värdet 0 (noll) vilket alltså ger variabeln ”fade” startvärdet 0.

Du bör vara väl bekant med *Set Analog Pin* och *Delay Milliseconds*; dra dessa block och fäst dem vid *loop* i valfri ordning.

Vi måste kasta bort det värdeblock som följer med Set Analog Pin (dra det över till vänster i fönstret) och ersätt det med en variabel. För att lägga till en variabel, dra blocket Number Variable Name och skriv variabelns namn på avsedd plats. Alternativt kan du, när du har gjort en variabel, högerklicka på den och kopiera den för att få fler "fade"-variabler.

Slutligen, lägg till ytterligare ett block *Set Number Variable*, och ersätt det *0-värde* som det innehåller med en + operator. Ändra den så att den lägger till 1 för att ändra ljusstyrkan och koppla den till värdedelen av *Set Number Variable*-blocket. Dra sedan den blockgruppen till slutet av *loop*.

Puh! Låt oss se vad allt arbete var bra för genom att ladda upp programmet. Lysdioden på pin 5 ska nu, till synes steglöst, gå från mörkt till fullt ljus.

## Upptäck mer

- Spelar det någon roll I vilken ordning blocket i *loop* ligger?
- Kan du dimma andra lysdioder på DS-kortet på samma sätt? Vad sägs om att dimma fler än en LED samtidigt?
- Kan du göra så att LED-lampan ändras från HIGH till LOW istället? (Tips: Du kan behöva ändra inställningsvärdet för "fade" och ändra + till -.)

# 7: ”If” *villkor* ”Then” *följdhändelse*

Dimningen från det senaste experimentet fungerade bra ända tills vi kom fram till den maximala ljusstyrkan på 255. Vad som händer är då ett mysterium som bara är känt för kompilatorn* (och dig, när du en gång lär dig lite om datatyper). Men vad händer om (*if)* vi tvingade `fade` -variabeln att återgå till startvärdet när den nått ett visst värde?

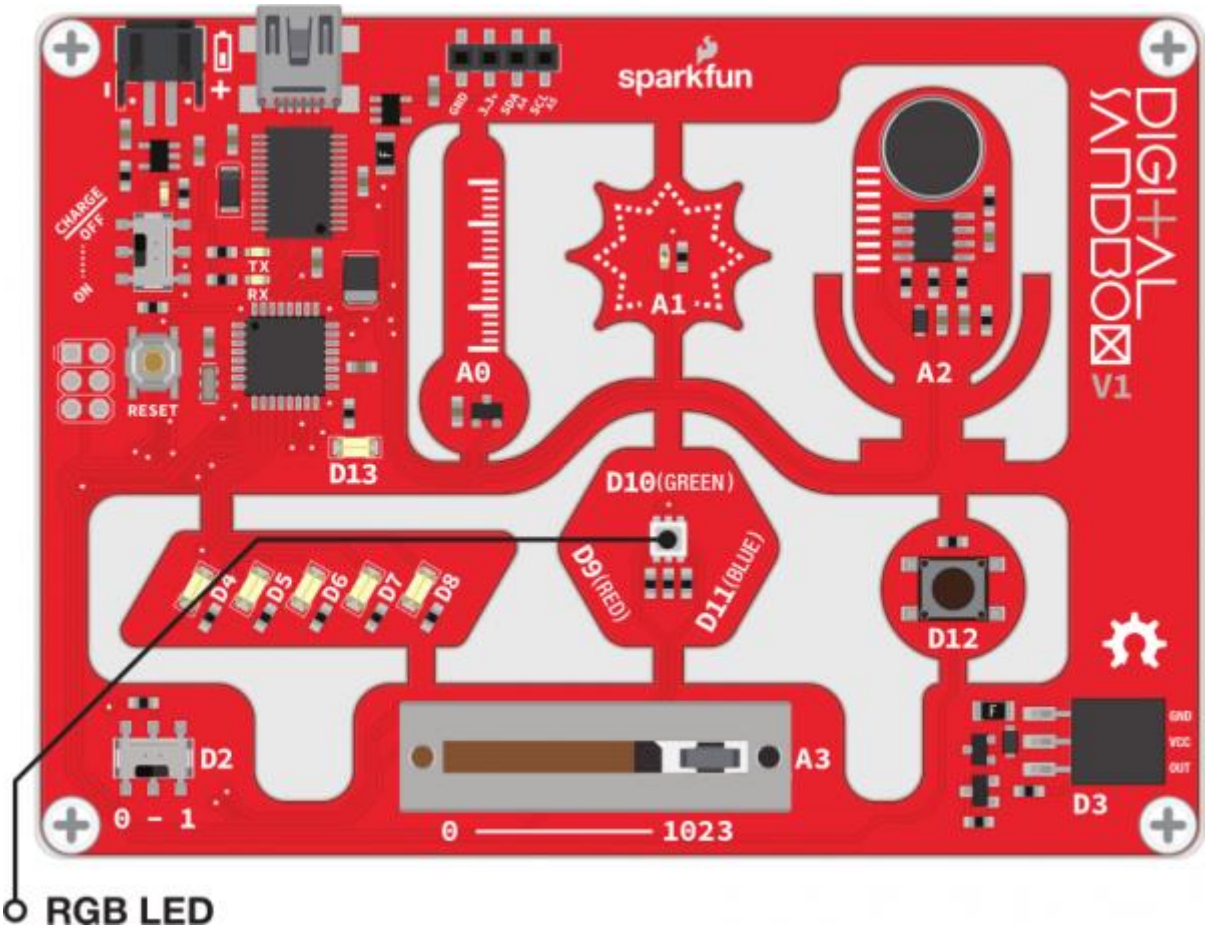
## Bakgrund

Detta experiment introducerar kommandot *if*, en av de mest grundläggande satserna inom programmering. Det är inte enbart för datorer som *if* -satser är viktiga, de styr också de flesta beslut som vi fattar i våra liv: *If det regnar ute*, *then ta med ett paraply*. *If du är hungrig*, *then fixa dig en macka*. Precis som för oss så använder datorer *if* -satser för att välja olika alternativ.

En *if* -sats behöver två delar för att bli komplett: ett villkor och en följdhändelse. Ett villkor är ett värde eller en matematisk uträkning som antingen uppfyller villkoret (true) eller inte (false). Om villkoret är sant (true), blir följdhändelsen utförd. Följdhändelsen kan bestå av ett enda block eller hundratals block.

Om villkoret för *if* -satsen är falskt, sker inte följdhändelsen, utan programmet fortsätter med den kod som kommer efter *if* -blocket.

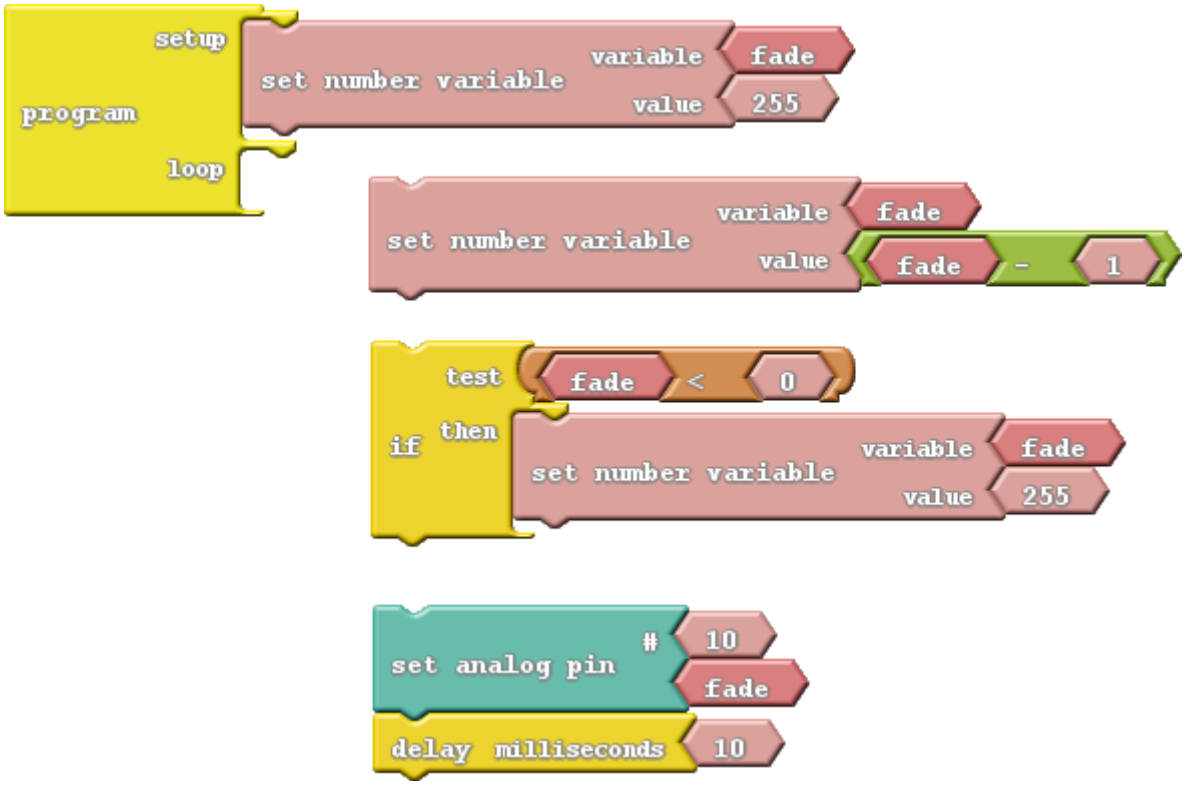
## Aktiva delar




*En kompilator är ett program som översätter textkod till maskinkod (ettor och nollor) som datorn kan ”läsa”

## Block till koden

Detta är de block du behöver till detta experiment.




Det finns två nya block som du behöver veta mer om:

- If: Stjärnan i detta experiment finns under Kontrollfältet. If-blocket kräver minst två andra block som ska fästas på det: villkoret och följdhändelsen. I det här fallet är följdhändelsen endast ett block - Set Number Variable. Den villkorade delen av if-blocket är ett block med en logisk operator.
- Logisk Operator: Logiska operatörer är symboler som fungerar på ett eller två värden och utvärderar till antingen sant eller falskt, vilket gör dem perfekt anpassade för att testa if-satsen! I det här fallet använder vi mindre än (<) operatören. Om värdet till vänster om (<) symbolen är mindre än värdet till höger, är operatören sann. Om vänster inte är mindre än (antingen större än eller lika med), kommer villkoret att utvärderas till falskt.

## Gör detta

I det här programmet vill vi att den blå lysdioden gradvis går från starkt ljus till avstängt och upprepa detta oändligt länge. Vi kommer att använda variabeln `fade` för att hålla ordning på det analoga utvärdet (output). I början av varje loop, kommer 1 att subtraheras (tas bort) från `fade`-variabeln.
Efter att detta gjorts och `fade`variabeln fått ett nytt värde måste *if* -satsen testas igen så att värdet ligger inom villkoren. *if* -satsen i detta program säger att om `fade` är mindre än 0 (det vill säga ett negativt tal), då ska `fade` ändras till 255.
Slutligen, när vi har genererat vårt värde för dimning, kan vi ställa in pin 10 (eller välj en annan LED om du vill) till det analoga utgångsvärdet.

Ladda upp och njut av en snygg, steglös dimning.

## Upptäck mer

- Kan du utföra dimningen på motsatt sätt? Börja vid 0, dimma upp till 255 och gå tillbaka till 0. (Tips: Du måste vända den logiska operatören om.)
- Gör det ännu jämnare! Kan du göra dimma ljuset upp och jämnt ner i samma program? Från 0 till 255, sedan 255 till 0, sedan 0 till 255, sedan tillbaka igen.

# 8: Testa din reaktionstid

Datorer är bra att räkna matte och automatisera tråkiga uppgifter, men alla vet att deras verkliga syfte är att spela spel. Låt oss skapa ett spel på Digital Sandbox! För att styra spelet måste vi lägga till input.

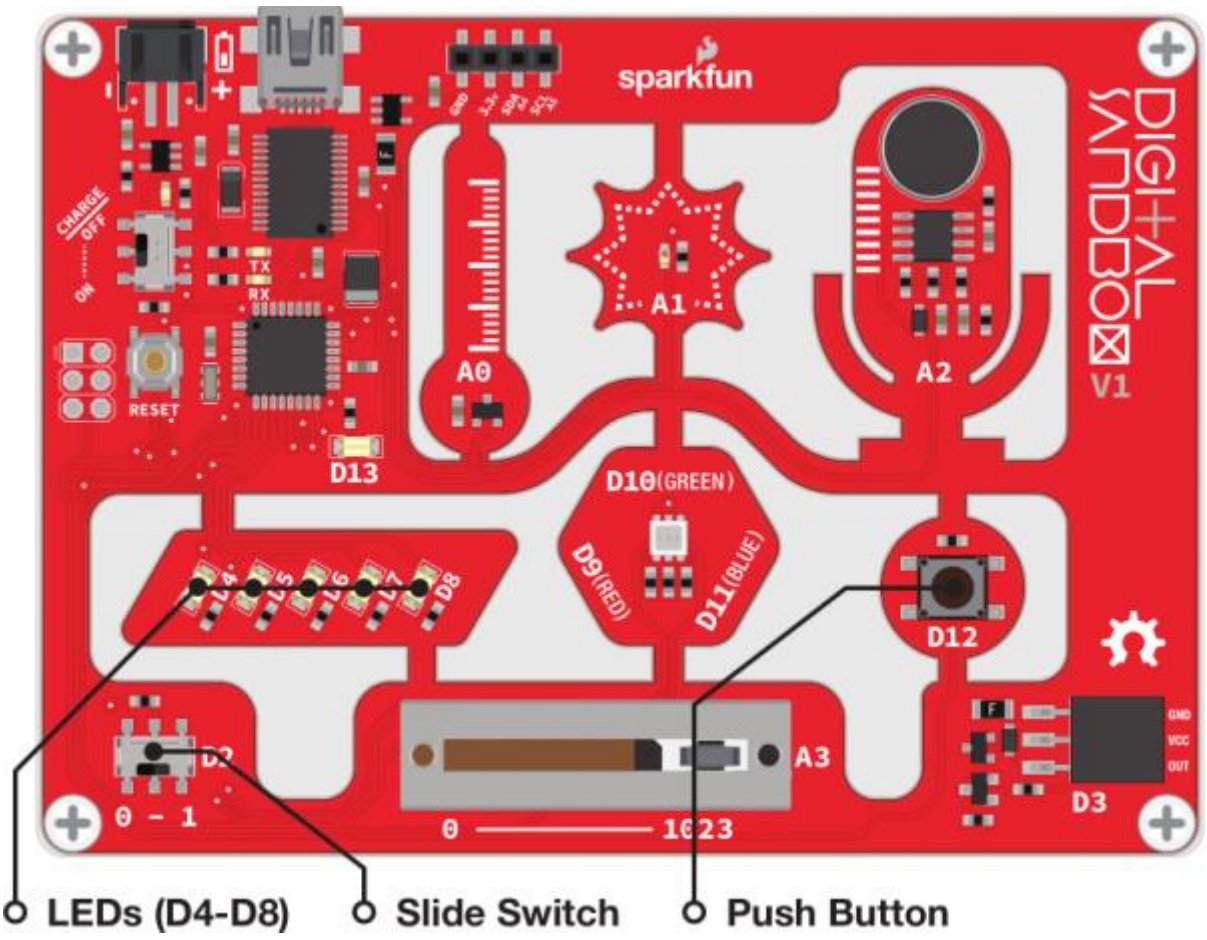
## Bakgrund

Hittills har vår Digital Sandbox-upplevelse varit mycket ensidig. Utgång till små gula lysdioder. Utgång till större vita lysdioder. Utgång till RGB-lysdioder. Ändra värdet för utmatningen (output). Utgång, utgång, utgång. Låt oss ändra på detta och göra lite inmatningar (input) till DS-kortet!

Ingångar (input) är signaler eller värden som skickas till ett system. Några av de vanligaste inmatningskomponenterna är knappar. Knapparna på ett tangentbord är en inmatning till din dator eftersom de skickar data till det systemet.

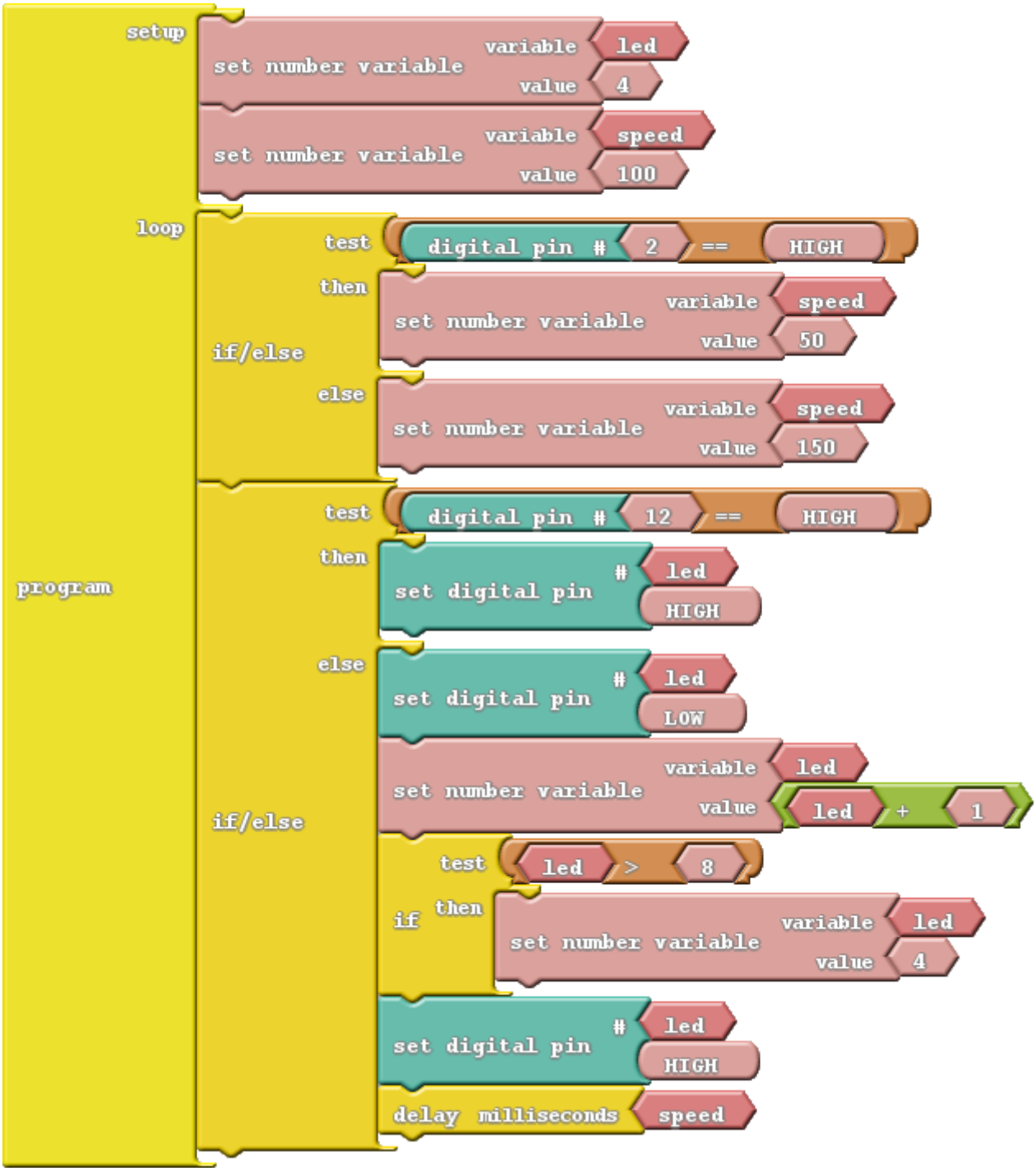
*If* -satser är kritiska när man bedömer statusen för en ingång och utför en åtgärd baserad på den. Exempelvis om knappen A trycks in skriver du ut ett "a." Vi kan ta *if*-satsen ett steg längre genom att lägga till ett annat villkor, vilket gör att vi att kontrollera vad som händer om *if*-satsen utvärderas till falskt. Så nu kan vi säga något som "om ägget flyter, släng det bort, annars (else) stek det och ät det!"

## Aktiva delar



## Block till koden

Vårt spel kommer att använda både omkopplaren (längst ner till vänster på Sandboxen) och den lilla knappen - komponenterna bundna till pin D2 respektive D12. Programmet är ganska stort, så vi sätter ihop det för dig. Så här ser det ut:




- If/Else: Det här blocket fungerar precis som *if*-blocket, men det låter dig bestämma vad som händer om villkoret utvärderas till falskt. Återigen behöver du ett block med villkors-sats (eller en uppsättning block) som utvärderar blocken vid Test till antingen sant eller falskt. Du måste också lägga till två separata block med kod för att fylla både then och else.

- Likhetstest (==): För att testa om två värden är lika, så används påståendet ==. Helt riktigt, här används två likhetstecken. Detta för att skilja sig från ett enda likhetstecken, som används för att tilldela ett visst värde. Dubbla likhetstecken är som att fråga "*är dessa två värden lika?"*

## Gör detta

Ordna dina block så att de ser ut som på bilden på förra sidan. Det finns två viktiga *if/else* -satser i detta program, som vart och ett testar ett invärde (input). Det första *if/else* testar pin 2, som är ansluten till strömbrytaren. Om brytaren står på ett (HIGH), då anges värdet på variabeln `speed` till 50. Om brytaren står på noll (LOW), då sätts värdet på `speed` till 150.
Det andra *if/else* testar pin 12, som är en liten tryckströmbrytare. När knappen trycks in, är ingången inställd på ett (HIGH), och den är noll (LOW) när den släpps. Detta innebär att när knappen trycks in körs koden efter *then*. När knappen inte trycks in, kommer *else* -blocket att köras.

Kan du gissa vad som händer i de två olika fallen med tryckknappen (pin12)? Ladda upp programmet till ditt DS-kort och se om dina antaganden stämmer!

Detta är ett mycket enkelt spel. Välj en siffra mellan fyra och åtta, och försök att få motsvarande LED att stanna på den siffran genom att trycka på knappen.

För att växla mellan lätt och svårt läge, flytta omkopplaren från 0 till 1. Klarar du att stoppa rätt LED när du kör i svåra läget?

## Upptäck mer

- Lura en kompis genom att byta vilken riktning som ställer den i det lätta läget - gör noll svårt och ett lätt.
- Byt funktionen på strömbrytaren och knappen, så du måste trycka på knappen för att ställa in svårigheten och slå på strömbrytaren för att stoppa lysdioderna.

# 9: Seriell Räknare

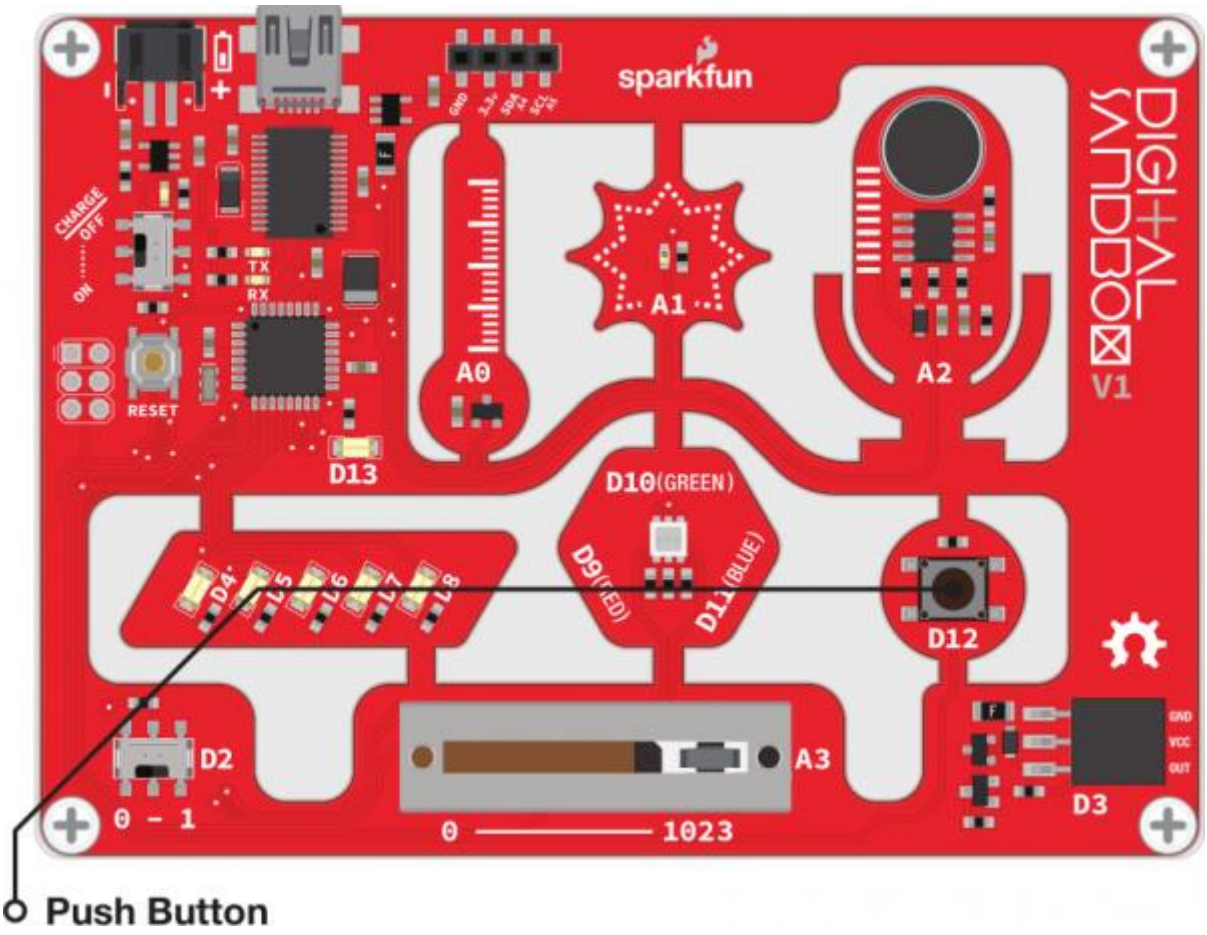
Medan du förmodligen inte kan ha en särskilt stimulerande konversation med Digital Sandbox, kan den ändå skicka dig intressanta uppgifter. Den är bra på matte, så låt oss få Sandboxen att göra lite beräkningar åt oss! Problemet är hur kan vi få den att skriva ut siffror (utan Morse-kod)? Jo, genom att ange seriell kommunikation (Serial communication)!

## Bakgrund

Serial communication är en form av dataöverföring där vi kan skicka en sträng av 1 och 0 mellan två enheter och faktiskt bilda olika tecken, som ord. Exempelvis så är 01101000 01100101 01101100 01101100 01101111 00101100 00100000 01110111 01101111 01110010 01101100 01100100 maskinkod för “Hello, world.”

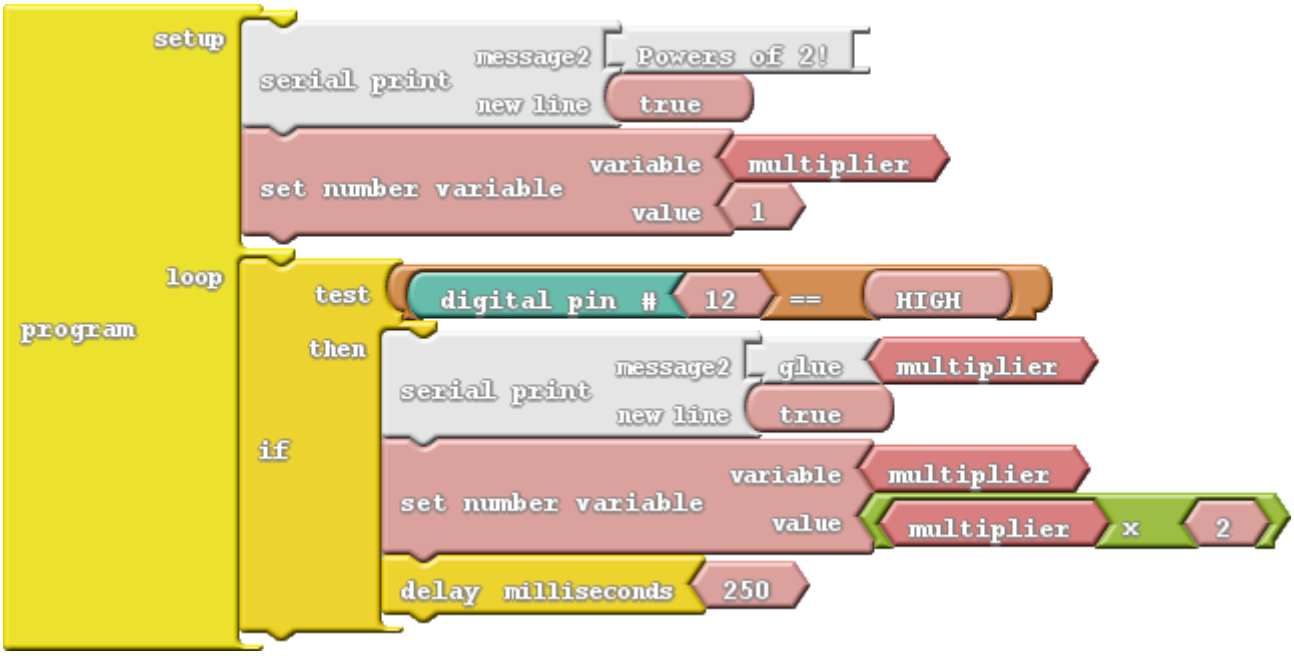
Med seriell överföring kan vi skicka text från DS-kortet och visa den på vår dator genom att använda Serial Monitor.

## Aktiva delar



## Block till koden

Här ser du hur blocken ska arrangeras för detta experiment:



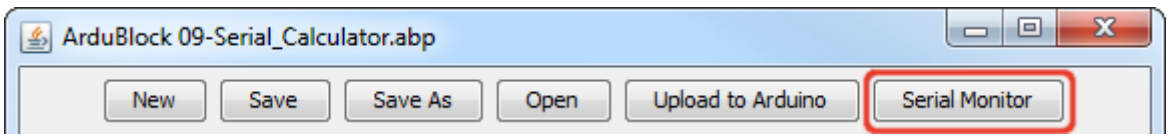

Det finns två nya block den här gången, båda ljust gråblå till färgen och du hittar dom under ikonen communication :

- Serial Print: Detta block använder två parametrar. Längst upp vid message2 skriver du det som du vill ska synas på datorskärmen. Du kan skriva nästan vad du vill här, även mellanslag. Om du vill lägga till en variabel eller ett nummer måste du använda lim (glue). Den nedersta urtaget i *serial print* bestämmer om en ny rad skrivs ut efter meddelandet. Vanligtvis vill du att detta ska ställas in på `true`.
- Glue: Glue-blocket gör att du kan skriva ut värden som variabler eller siffror - allt som inte är ett meddelande (text) du har skrivit in. Om du vill skriva ut en variabel måste du lägga till ett *glue*-block mellan variabeln och *Serial Print*-blocket. Det finns tre olika sorters *glue*-block, var och en med olika form på urtaget till höger. Den här gången använder vi blocket med ett trekantsuttag på *glue*-blocket.

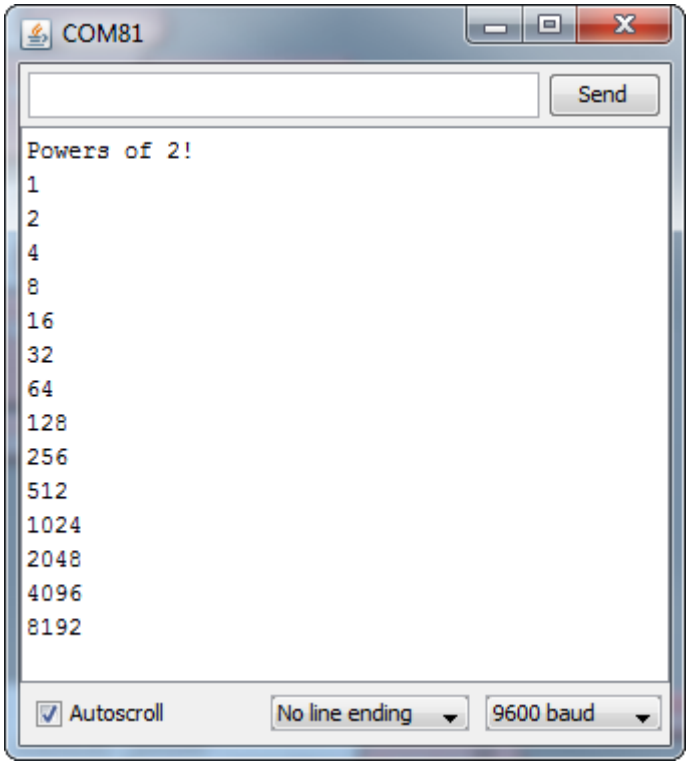
## Gör detta

I början av vårt program vill vi skriva ut ett meddelande. Vad sägs om en kort beskrivning av vad vår räknare ska göra: "Heltalspotenser med basen 2!" Sätt sedan in en variabel som vi kan använda att räkna med, ge den startvärdet 1.

I loopen vill vi bara göra beräkningar och skriva ut resultatet när knappen trycks ned. Så börja med att lägga till ett *if*-block för att kontrollera om pin 12 är HIGH (knappen trycks in). Om knappen trycks in, görs beräkningar och resultatet skrivs ut. För att skriva ut en variabel måste vi "limma" den till *Serial Print*-blocket med ett *glue*-block som har ett trekantsformat uttag. Gör ditt program så att det ser ut som bilden ovanför.
Ladda nu programmet till ditt DS-kort. För att se utskriften, glöm inte att klicka på Serial Monitor.



Nu ska du kunna se ditt meddelande utskrivet på skärmen. Tryck nu på D2-knappen för att börja själva beräkningen.




## Upptäck mer

- Något konstigt händer när svaret på beräkningarna kommer över 16 834 och sedan blir -32 768 för att slutligen bli noll. Detta beror på att vår variabel har nått sitt maximala värde och på sätt och vis blivit förvirrad. Kan du lägga till en *if*-sats för att få en `multiplier` -variabel som inte är lika begränsad utan återställer den på ett mer korrekt sätt?
- Prova några av de andra matematiska operatorerna. Du är förmodligen bekant med +, -, × och ÷, men vad gör operatorn %?

# 10: Använda ett analogt skjutreglage

Digitala ingångar, som knappen, tillåter bara två inmatningsvärden: HIGH (På) eller LOW (Av). Men hur är det med mittemellan-värden? När du vrider volymknappen på din stereo, behöver du inte bara välja mellan tyst och "vrålhögt." För volymkontroller och andra "finjusterade" inställningar behöver vi analoga ingångar.

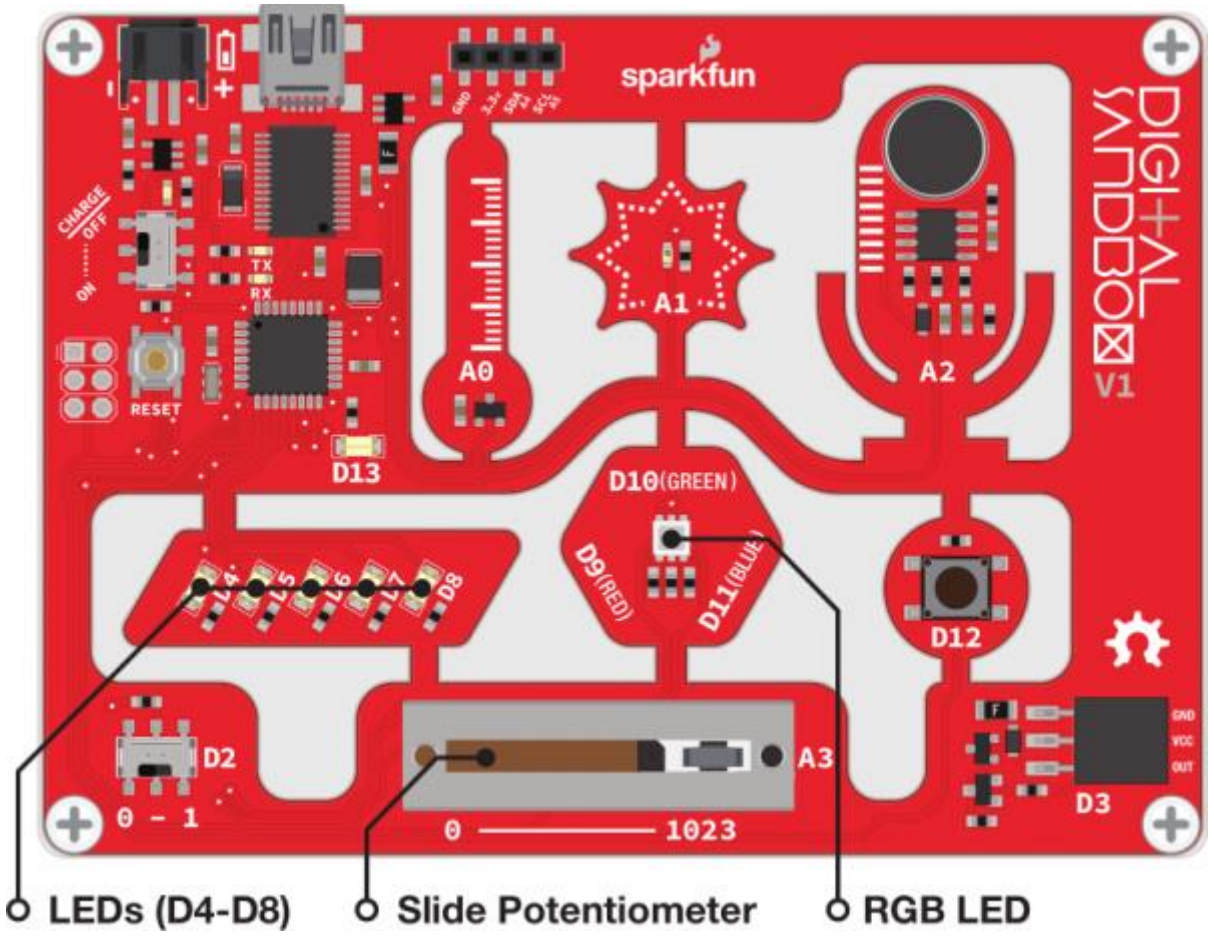
## Bakgrund

Analoga ingångar är komponenter som skriver data till ett system med ett intervall på mer än två värden. På DS-kortet kan analoga ingångar ge ett värde på vad som helst mellan noll och 1023 (totalt 1024 värden). Värdet som produceras av en analog ingång är proportionell mot den spänning som den producerar. Om en analog komponent läser ett värde på noll är spänningen 0V. Om utgångsvärdet är 1023, är spänningen 5V. En analog läsning av 512 är ca 2,5V och så vidare.

En speciell komponent I den mikrokontroller som finns på DS-kortet kallas en analog-till-digital-omvandlare, analog-to-digital converter (ADC). Den klarar av att omvandla intervallet av ingångsspänningar till ett diskret värde (heltalsvärden). Det här är en speciell krets som de flesta pins på Sandboxen inte har. Den är så speciell att alla ADC pins är märkta med ett inledande **'A'**. De analoga sensorerna på kortet är märkta som "A0", "A1", "A2" och "A3".

Många elektroniska komponenter har en analog utgång, vanligast är potentiometern. Dessa förekommer i olika former och storlekar. Vridpotentiometrar (rattar) används ofta för att justera volymen på en stereo. En skjutpotentiometer, som den längst ner på DS-kortet, används ofta för att justera ljudnivåerna på ett mixerbord.

## Aktiva delar



## Block till koden

Okej, efter det senaste experimentet är det dags för en ganska enkel blockprogrammering:


- Analog Pin #: Precis som för Digital Pin-blocket, läser detta block värdet av en ingång. Men i stället för att producera antingen sant eller falskt (1/0, HIGH/LOW), producerar detta block ett tal mellan noll och 1024. Det rosa blocket som är fastsatt till höger om *glue*-blocket anger vilken analoga ingång (pin) som ska läsas.

## Gör detta

Programmera enligt bilden ovan. Ladda ner och öppna sedan Serial Monitor.

Varje 100:de millisekund kommer ett analogt ingångsvärde att skrivas ut. Flytta det analoga reglaget (skjutpotentiometern) för att justera värdet. Kan du göra värdet till noll? Till 1023? Till 512? Notera vilken position som hör ihop med vilket värde.

## Upptäck mer

- Kan du få skjutreglaget att styra LED-lamporna? Du kan prova att justera ljuset från de vita lysdioderna, eller försök att styra ljusstyrkan hos RGB-lysdioderna med skjutreglaget.
- Varför finns det just 1024 värden? Varför inte i så fall 1000 som ju är mycket “jämnare” tal? (Tips: $2^{10}$.)

# 11: Automatisk nattlampa

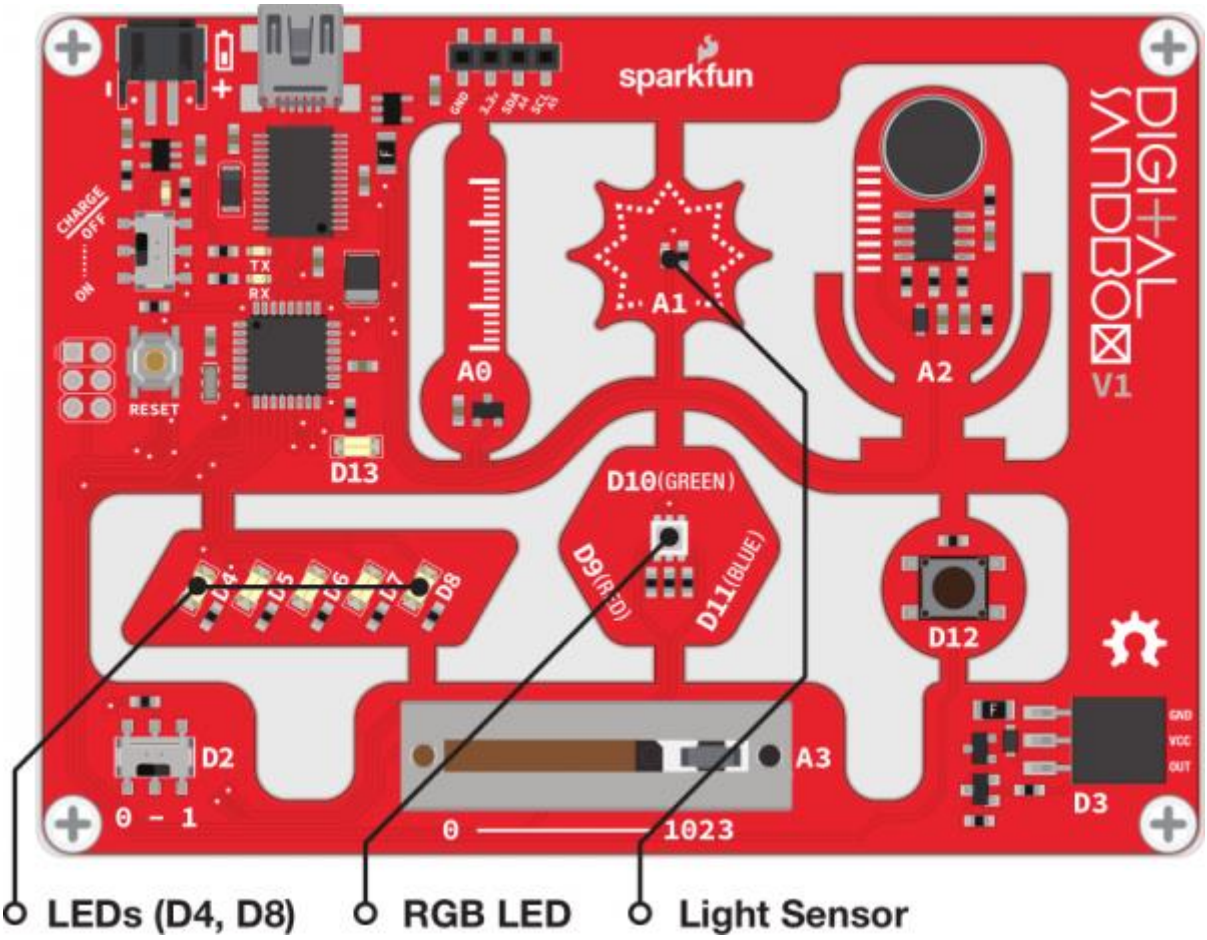
Nu har vi alla programmeringsverktyg som vi behöver för att göra några helt fantastiska interaktiva projekt. Låt oss infoga ljussensorn, en annan analog ingångskomponent, för att skapa en automatisk nattlampa som tänds när det är mörkt.

## Bakgrund

Du kanske inte ser dem, men ljussensorer är inbyggda i många typer av moderna elektroniska enheter. Det finns ljussensorer i smartphones som mäter hur ljus din miljö är och justerar skärmens ljusstyrka efter detta. Det finns ljussensorer i rökdetektorer som upptäcker partiklar i luften.

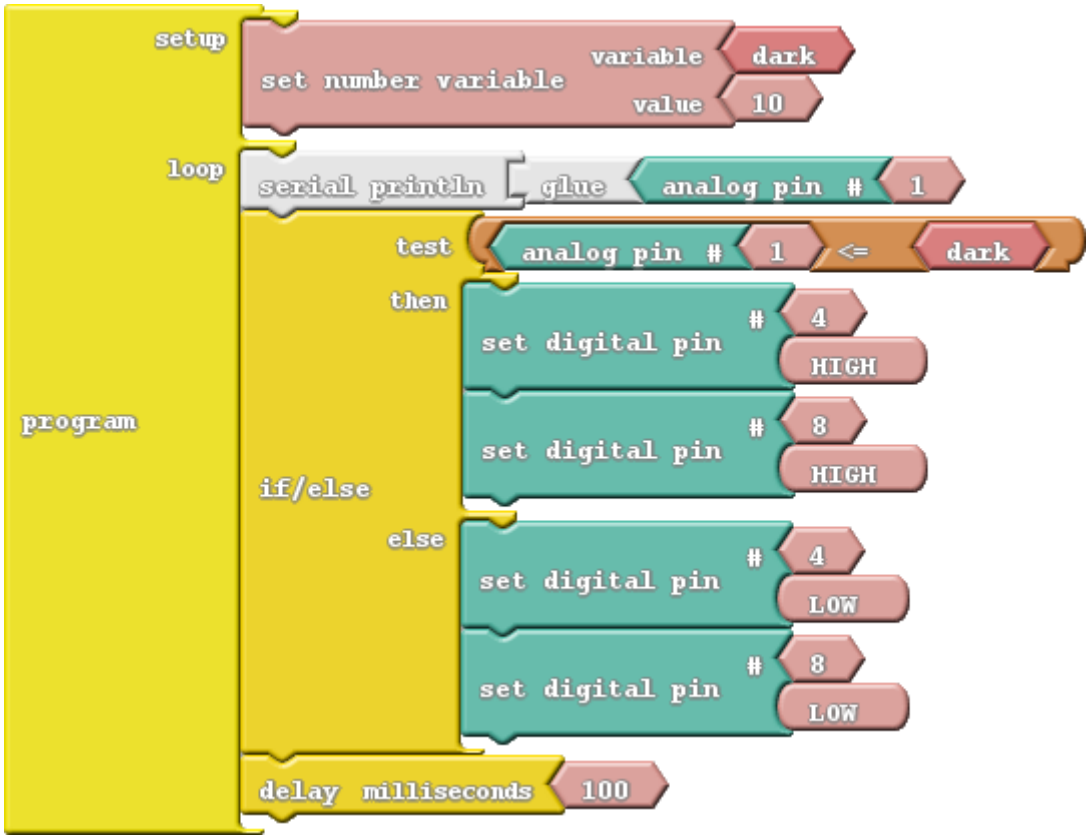
Den typ av ljussensor som finns på DS-kortet kallas för fototransistor. Den ger en analog spänning i förhållande till mängden ljus som registreras. Ju lägre analogvärdet desto mörkare är omgivningen. Om du täcker sensorn helt kan du få utvärdet hela vägen ner till noll. Lys med en ficklampa på den och du kanske kan få ett maximalt värde, det vill säga 1023.

## Aktiva delar



## Block till koden

Så här ska blocken arrangeras för detta experiment:




Det finns inga nya block, men vi kanske måste justera värdet på mörkervariabeln för att få nattlampan att fungera perfekt. Det är där seriell kommunikation kommer att vara till nytta!

## Gör detta

Sätt ihop dina block enligt bilden ovan och ladda till din DS.

Tricket för detta experiment är att hitta den perfekta inställningen för mörkervariabeln. Om ditt rum är ljust, ska de vita lysdioderna på pin 4 och 8 vara avstängda. När det är mörkt (eller om sensorn är täckt) ska lysdioderna lysa.
Det är kankse så att lamporna inte beter sig ordentligt från början! Då behöver du bara finjustera den mörkervariabeln. Öppna *Serial Monitor* för att se ljussensorns utgångsvärden. Notera värdet när lamporna lyser och stäng sedan av lamporna. Vad är värdena nu? Öka detta värde på mörkervariabeln lite grann och prova igen.

## Upptäck mer

- Om ljusstyrkan ligger precis vid gränsen för på/av, kan lamporna blinka på ett tråkigt sätt. Försök att lägga till en annan *if*-sats för att få ljussensorn att få en korrekt inställning, utifrån detta kan du dimma lysdioderna baserat på vad sensorn registrerar.
- Försök med att integrera RGB-LED i detta projekt. Till exempel, om det är superljust, tänd gul LED, blå om det är mörkt.

# 12: Temperaturvarning!

"Är det varmt här, eller är det bara jag?" Med hjälp av en temperatursensor, som kan mäta rumstemperaturen, kan vi svara på den där frågan en gång för alla!

## Bakgrund

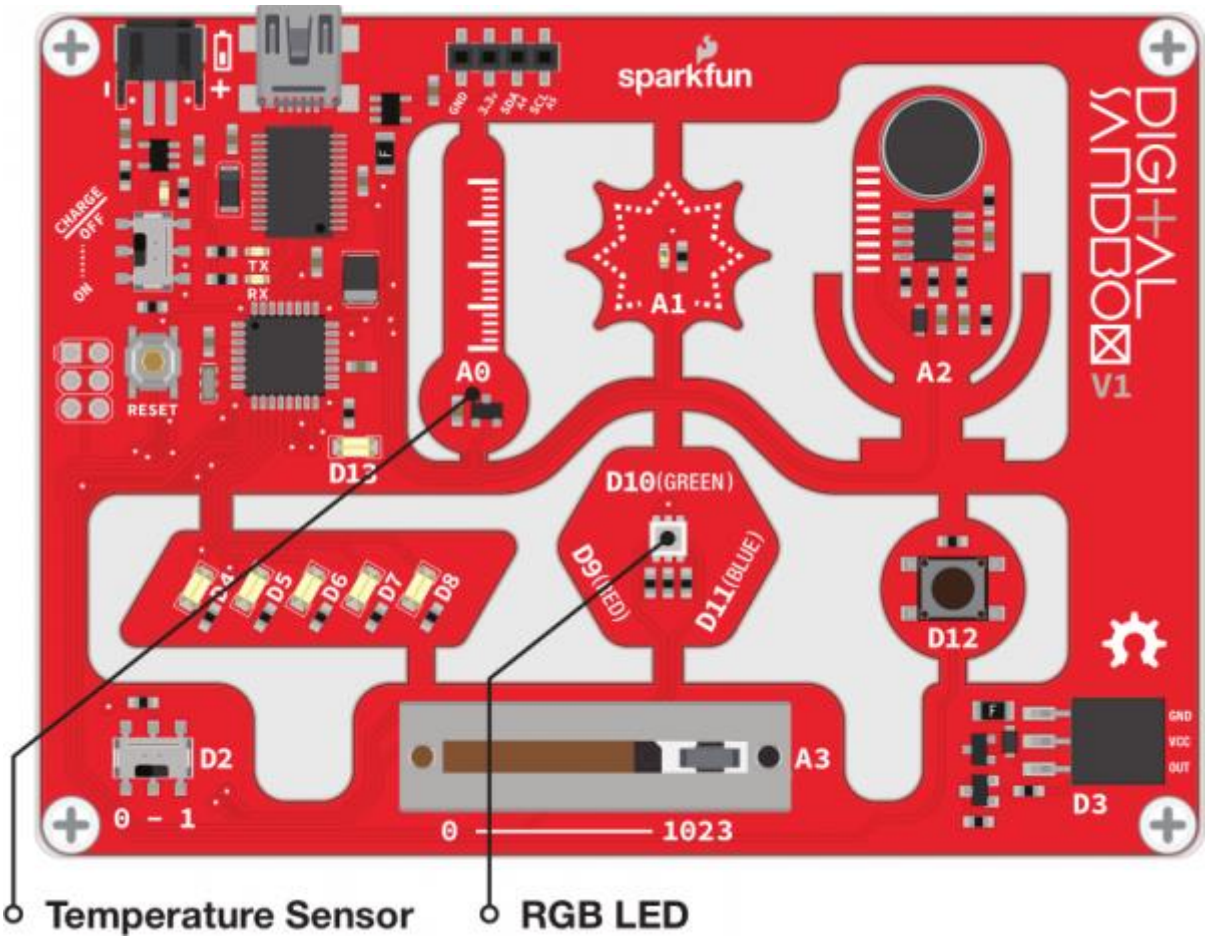
Temperatursensorer är en kritisk komponent i många kretsar, oavsett om du styr ett A/C-system eller skapar en säkerhetsmekanism för gasdrivna apparater. Elektroniska temperatursensorer finns i många olika varianter, från stora termoelement som kan mäta upp till 1000°C till den lilla svarta rektangeln på Digital Sandbox.

Temperaturgivaren på DS-kortet ger en analog spänning som representerar temperaturen nära sensorn. Spänningen är linjärt proportionell mot temperaturen i grader Celsius. Om du känner till sensorns utspänning kan du beräkna temperaturen med denna ekvation:

$$\text{Temperatur} = (\text{Volt} - 0{,}5) \times 100$$

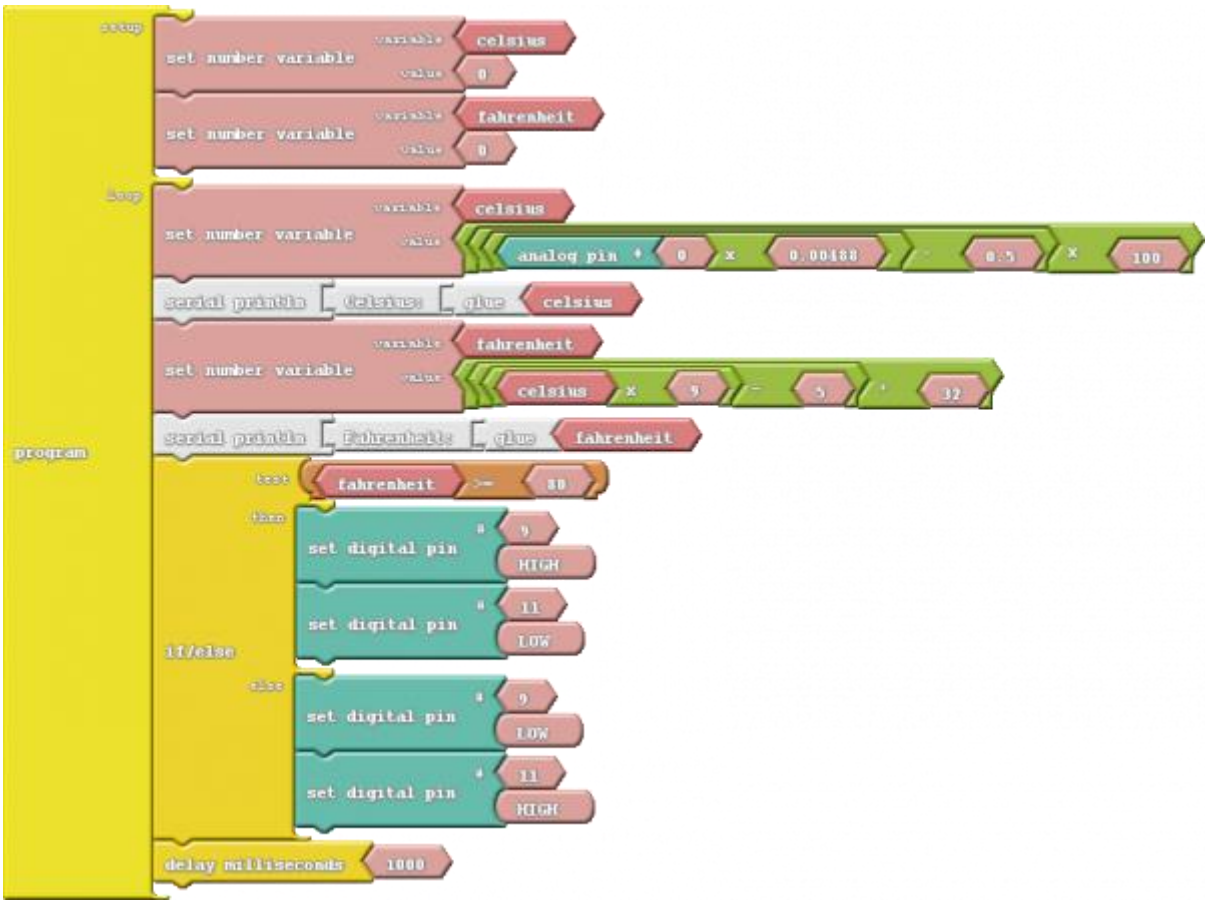
Vi kan få vår mikrokontroller göra beräkningen för oss så länge vi hittar rätt algoritm, en ekvation eller en uppsättning instruktioner som åstadkommer en specifik uppgift.

## Aktiva delar



## Block till koden

Vi varnade dig, det blir mycket matematik i den här programmeringen. För en större bild, gå till https://cdn.sparkfun.com/assets/learn_tutorials/1/7/5/12_snapped_update.png




Det finns inga nya block här men som vi kan se får vi använda en mängd olika matematiska operatorer. Var uppmärksam på ordningen på blocken. När du har en serie beräkningar utförs den innersta först.

## Gör detta

Konstruera programmet enligt ovan. Se till att de matematiska funktionerna är i rätt ordning! Decimaldelen (t ex ".0", ".5", ".01") är viktig för att få rätt precision vid beräkningen.

Ladda ner till DS-kortet och kolla in RGB-lysdioden. Är det rött eller grönt? Om det är rött, är du förmodligen varm, eftersom din rumstemperatur är över 78°F (25°C) . Om det är grönt, försök värma upp sensorn genom att blåsa på den. Kan du få den att bli röd?

För att få korrekt avläst temperatur, öppna Serial Monitor. Efter att ha sett vad som skrivs på skärmen kanske du vill ändra `78` i *if/else* -testet.

## Upptäck mer

- Kan du lägga till en tredje kontroll för att varna när den är för kall genom att slå på den blå lysdioden? Det verkliga tricket här är att lyckas kyla ner Sandboxen. Ett alternativ är att driva kortet med ett batteri och ha det i kylskåpet.
- Celsius och Fahrenheit är två av de vanligaste temperaturskalorna, men de är inte de enda. Kan du skriva ut temperaturen i enheterna Kelvin eller Rankine? Du måste hitta en algoritm för att konvertera från Celsius.

# 13: Mäta ljud

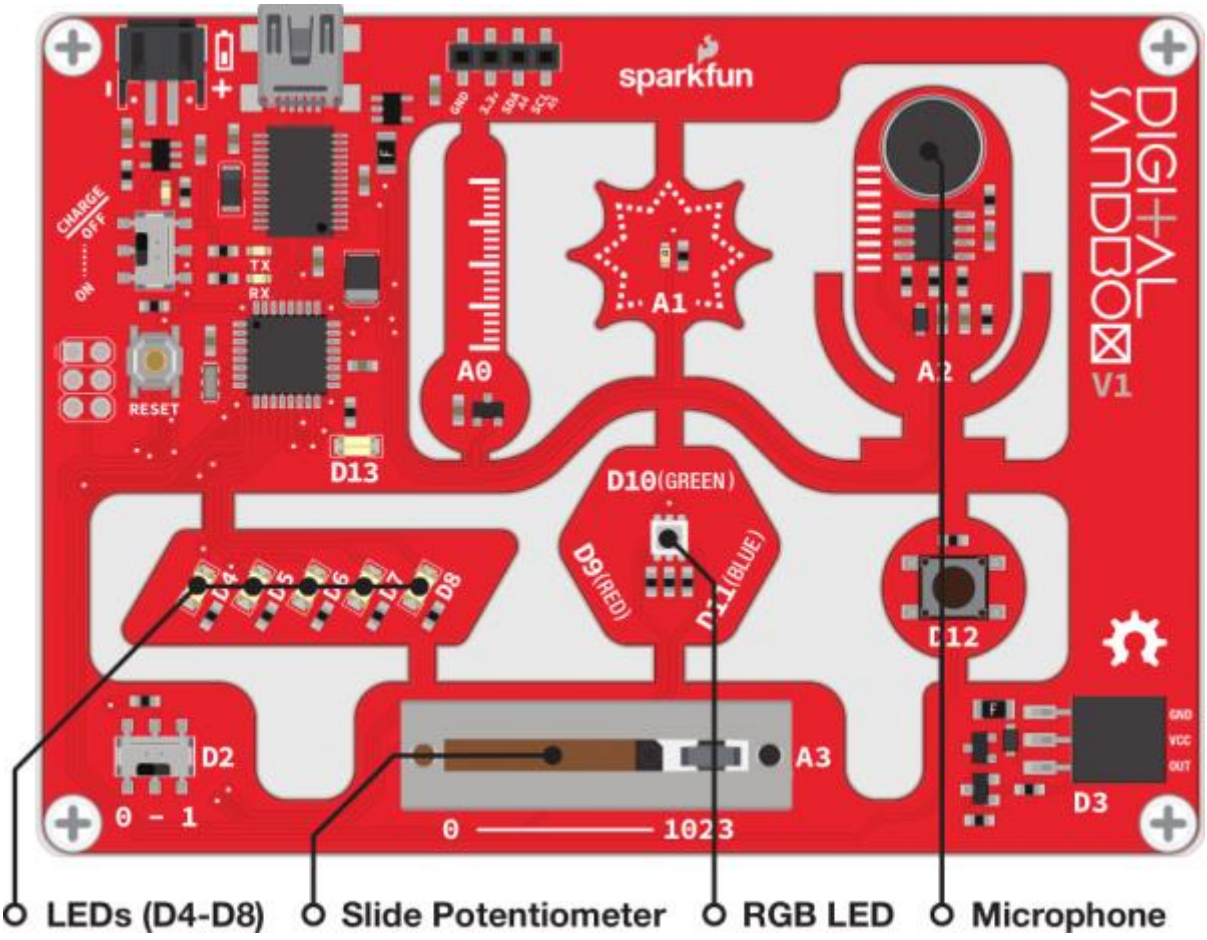
<*Pitchman voice*> Introducing the fabulously groundbreaking SOUND (Sandbox's Over/Under Nominal Decibels) System! Microphone check 1..2..1..2. With the SOUND you'll always have an adjustable sound level detector handy! </*Pitchman voice*>

## Bakgrund

I detta experiment använder vi Sandboxens inbyggda mikrofon för att mäta volymen och visa den med LED-lamporna. Mikrofonen producerar en ljudvåg, som omvandlas till en analog spänning som vi kan mäta. Ju högre ljud, desto högre är vågens amplitud och desto högre spänning.

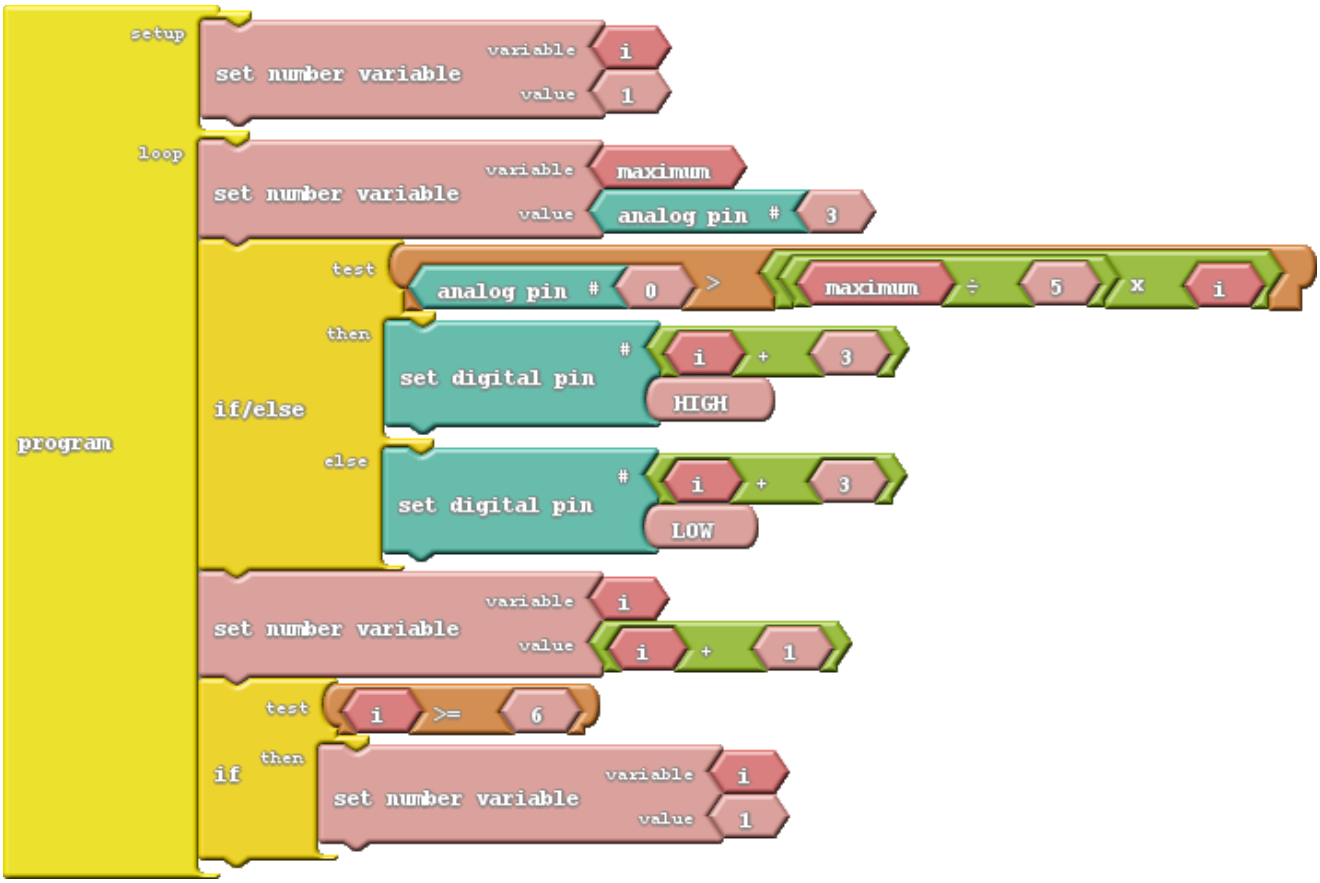
Utan komplicerade beräkningar och bra filter kan ljud vara en svår sak att mäta. Att använda Sandboxen för röstigenkänning är inte möjligt, men den kan programmeras att plocka ut höga volymer så länge som mikrofoningången klarar att reagera tillräckligt snabbt. Vi kan använda glidpotentiometern för att ställa in känsligheten.

## Aktiva delar



## Block till koden

Inga nya block att introducera den här gången. Vi tar den analoga ingången på pin A2 för att läsa av mikrofonnivån och tända LED-lampor baserat på detta värde.




## Gör detta

Efter att ha arrangerat blocken, ladda upp programmet och titta på lysdioderna. Reagerar de på din röst? Om inte, knacka lätt på mikrofonen för att se om det blir någon reaktion.

För att justera volymmätarens känslighet, flytta glidpotentiometern upp eller ner. Med skjut-reglaget inställt längst till höger är den känslig och då kanske samtliga LED lyser. Ställer du skjutreglaget för långt till vänster kanske inget ljud registreras och inga LED lyser.

## Upptäck mer

- Kan du koda om programmet för att använda RGB-lysdioden istället för de vita lysdioderna? Gör den röd när volymen är riktigt hög, och blå och/eller grön annars. GULDSTJÄRNA om du använder analoga utgångar!

# 14: Opto-theremin (kräver tillbehörssatsen)

I detta experiment ansluter vi en högtalare till DS-kortet och gör det till ett ljusstyrt musik-instrument! Genom att använda ljussensorn för att styra högtalarens tonhöjd kan vi skapa en ljusstyrd theremin - ett tidigt elektroniskt musikinstrument (läs mer, exempelvis på Wikipedia).

OBS! Detta experiment kräver tillbehörssatsen (Alega art.nr. PR6x) Digital Sandbox Add-On Kit.

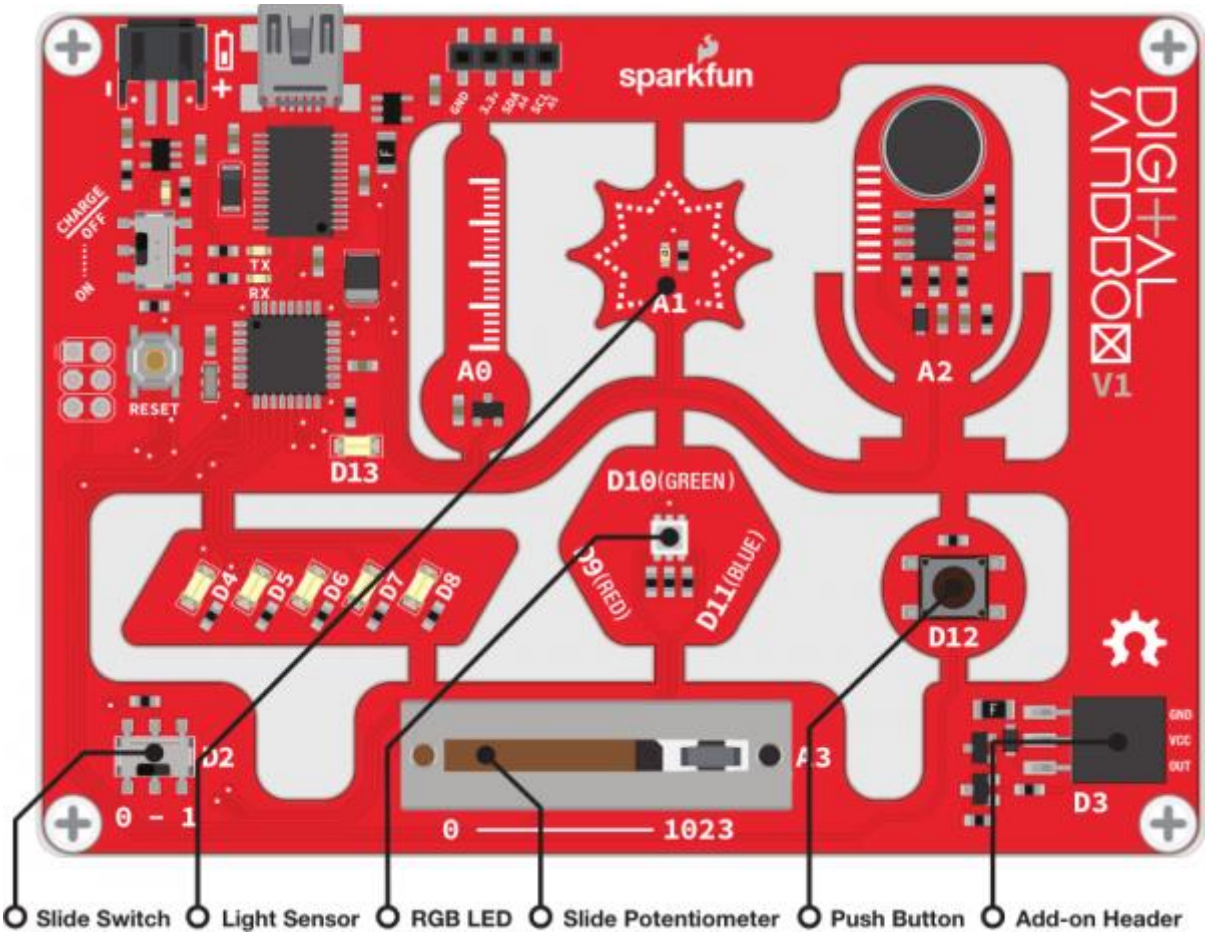
## Bakgrund

Genom att justera en pin exakt, kan DS-kortet skapa elektroniska vågor som, när de körs via en högtalare, kan producera en musikalisk ton. Vi kan programmera Sandboxen för att styra två egenskaper hos en musikalisk ton: tonhöjd och varaktighet.

Tonhöjd är vad vi uppfattar när vi tänker på en ton som mycket hög (skrik, bestick som skrapar mot en tallrik, dvs. hög diskant) mot mycket låga (dvs. mycket bas). Tonhöjden är mycket nära relaterad till frekvensen som hörs i en högtalare. Om vi exempelvis justerar en pin från HÖG till LÅG och sedan LÅG till HÖG 440 gånger per sekund, producerar den en frekvens på 440 Hz (Hertz). Människor kan normalt höra frekvenser som sträcker sig från ungefär 20 Hz (låg ton, bas) till 20 000 Hz (hög ton, "Aj, mina öron").

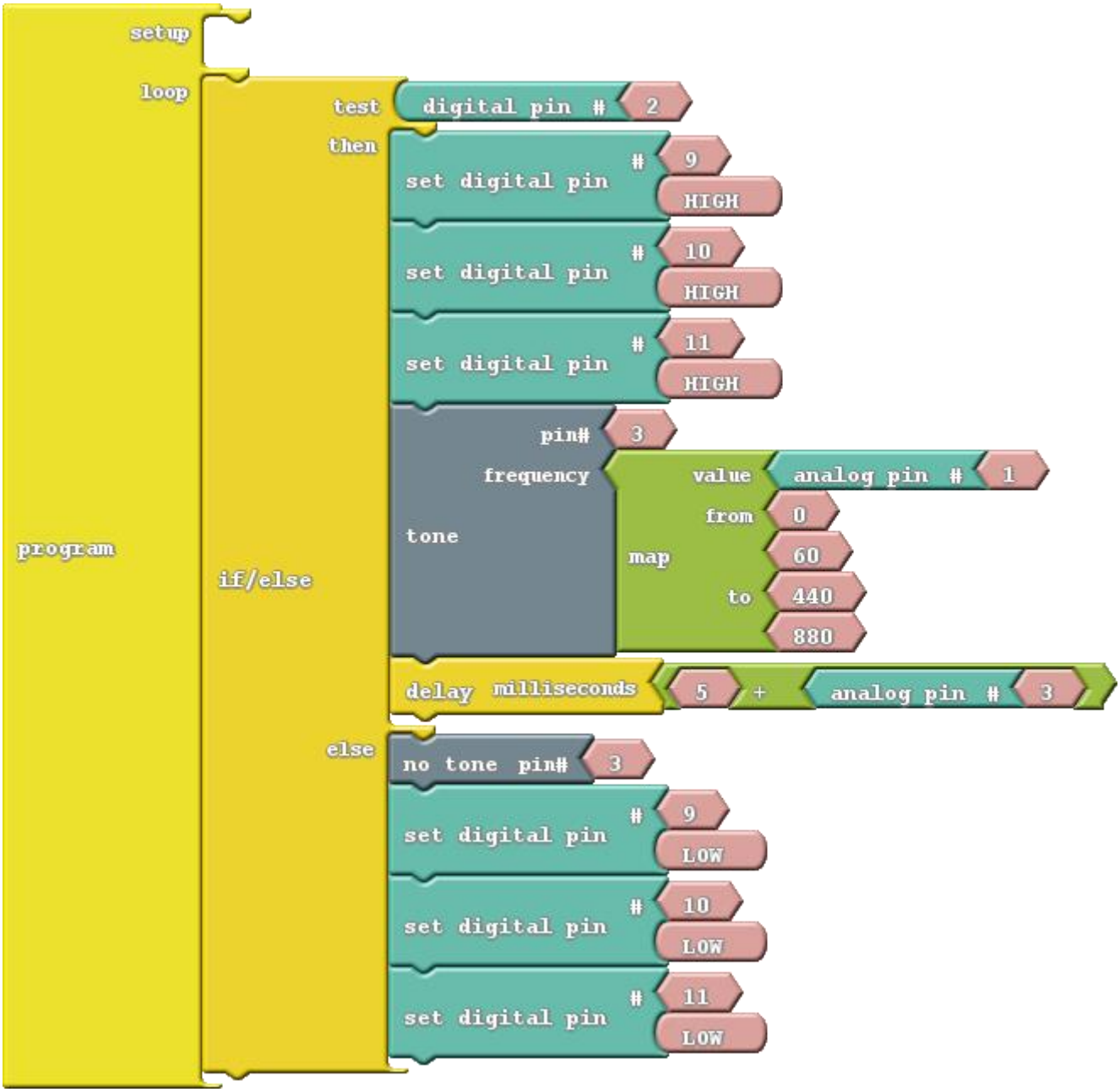
Vi kan också programmera tiden för en ton - hur lång tid en ton spelas. I vårt program använder vi fördröjningsfunktionen för att ställa in varaktigheten. Att spela en ton med DS-kortet är väldigt lätt. Ange bara en tonhöjd. Liksom en analog utgång, kan du ställa in tonhöjden och sedan glömma det, tonen slutar inte spela förrän du säger till DS-kortet att göra så.

## Aktiva delar



## Block till koden

I detta program ska du använda de nya blocken, Tone och No Tone. Här är det programmet:

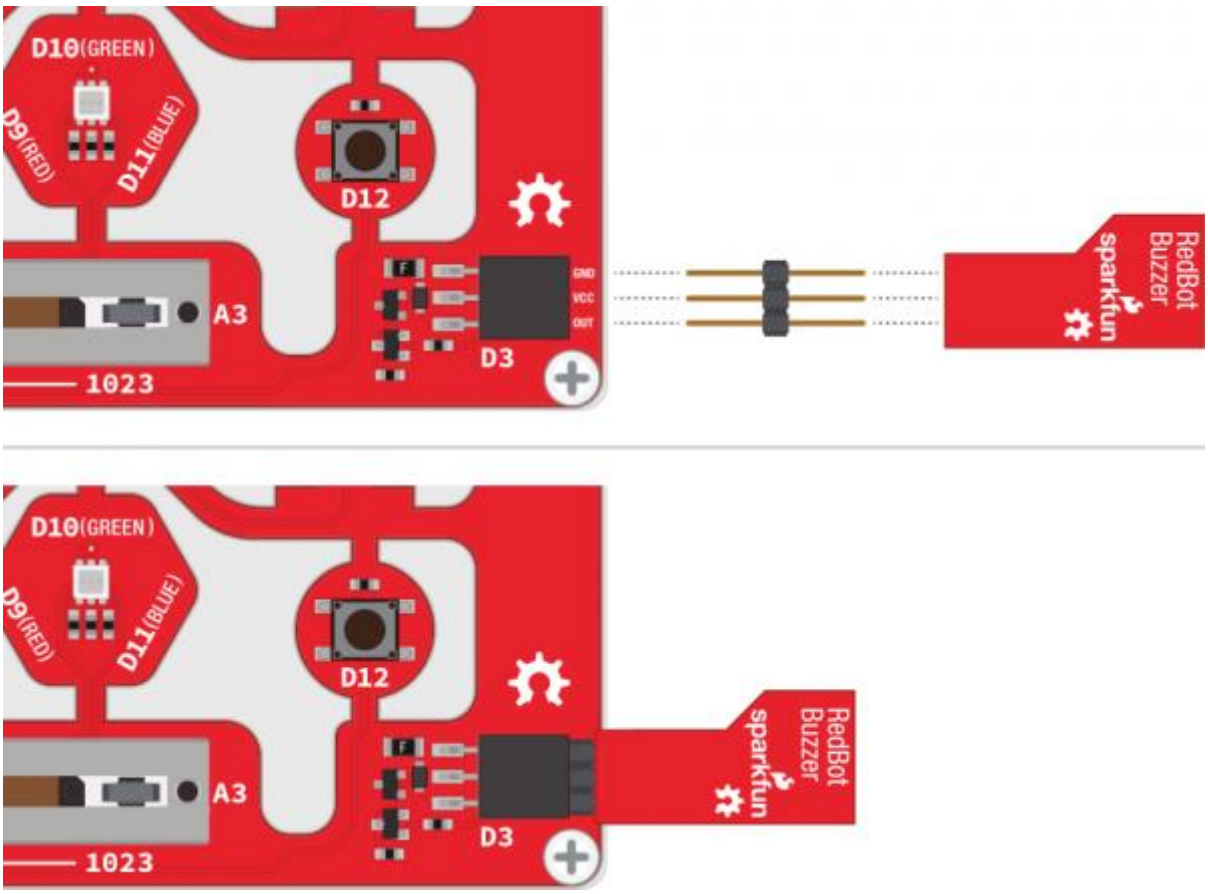



- Tone: Detta block använder två ingångsvärden: ett pin nummer och en frekvens (frequency). Den pin du använder kan vara vilken digital pin som helst, men i det här fallet använder vi extrakontakten, dvs. pin 3. Frekvensen kan vara allt från 31 Hz och uppåt till så hög frekvens som du hör.
  En ton som startas av tone-blocket kommer att fortsätta programmet anger ett No Tone-block. No tone stoppar helt enkelt en pin från att spela en ton. Båda dessa block finns under *Utilities* -ikonen.
- Map: Denna praktiska funktion tar ett värde från ett område och omvandlar till ett annat. I exemplet ovan tar vi ett värde (utgången från A3) mellan noll och 60 och omvandlar det till ett intervall från 440 till 880. Så en analog läsning av noll blir 440 och en avläsning av 60 blir 880, Allting mellan dem anges i förhållande till dessa två intervall. Map-funktionen kan vara särskilt användbar vid omvandling av en analog ingång (0–1023) till en analog utgång (0–255).

## Gör detta

Arrangera blocken enligt bilden på förra sidan och ladda ner till DS-kortet!

Du måste också ansluta högtalaren (en *summer* i detta fall) till extrautgången. Innan du kan göra det måste du bryta av en grupp med tre stift från stiftlisten. Nu kan du koppla summern, med den övre sidan nedåt, till extrautgången som visas på bilden:




Hur rolig den optiska thereminen än kan vara uppskattas inte ljudet (oljudet)? av alla, så i koden finns en enkel ON/OFF-funktion. Skjut omkopplaren till läget "1" för att aktivera thereminen.

När thereminen är på ska summern börja låta. Försök att täcka ljussensorn, ändras tonhöjden? Vi har kodat vår RGB-LED till vit, så du kan försöka spegla ljuset från den för att styra ljussensorn.

Du kan justera tonens varaktighet genom att skjuta potentiometern. Skjut reglaget hela vägen till noll för att få ett riktigt snabbt "zappande" ljud, eller skjut åt höger för att skapa ett lugnare ljud.

## Upptäck mer

- Försök lägga till en "vilo" -funktion för din opto-theremin. Använd knappen för att avbryta ljudutgången tillfälligt.
- I stället för att använda DS-kortet som ett musikinstrument kan du programmera det för att spela förbestämd musik? Använd olika toner och fördröjningar och försök att få DS-kortet att spela din favoritlåt!
- Gör ett hörselprov! Vilken är den högsta frekvensen du kan höra? Kan du höra toner som andra inte kan? Kan ditt husdjur höra toner som du inte kan?

# 15: Kör en elmotor (kräver tillbehörssatsen)

Motorer får världen att gå runt. Tja, inte bokstavligen, men de gör många saker som vi använder oss av varje dag. Det finns små vibrationsmotorer i mobiltelefoner, snabba motorer som driver CD och Blu-Ray-skivor, och naturligtvis stora motorer som hjälper till att driva våra bilar. I detta experiment utforskar vi en av de mest grundläggande motortyperna, en likströmsmotor (DC-motor DC=Direct Current) och vi kommer att programmera DS-kortet att driva motorn i olika hastigheter.

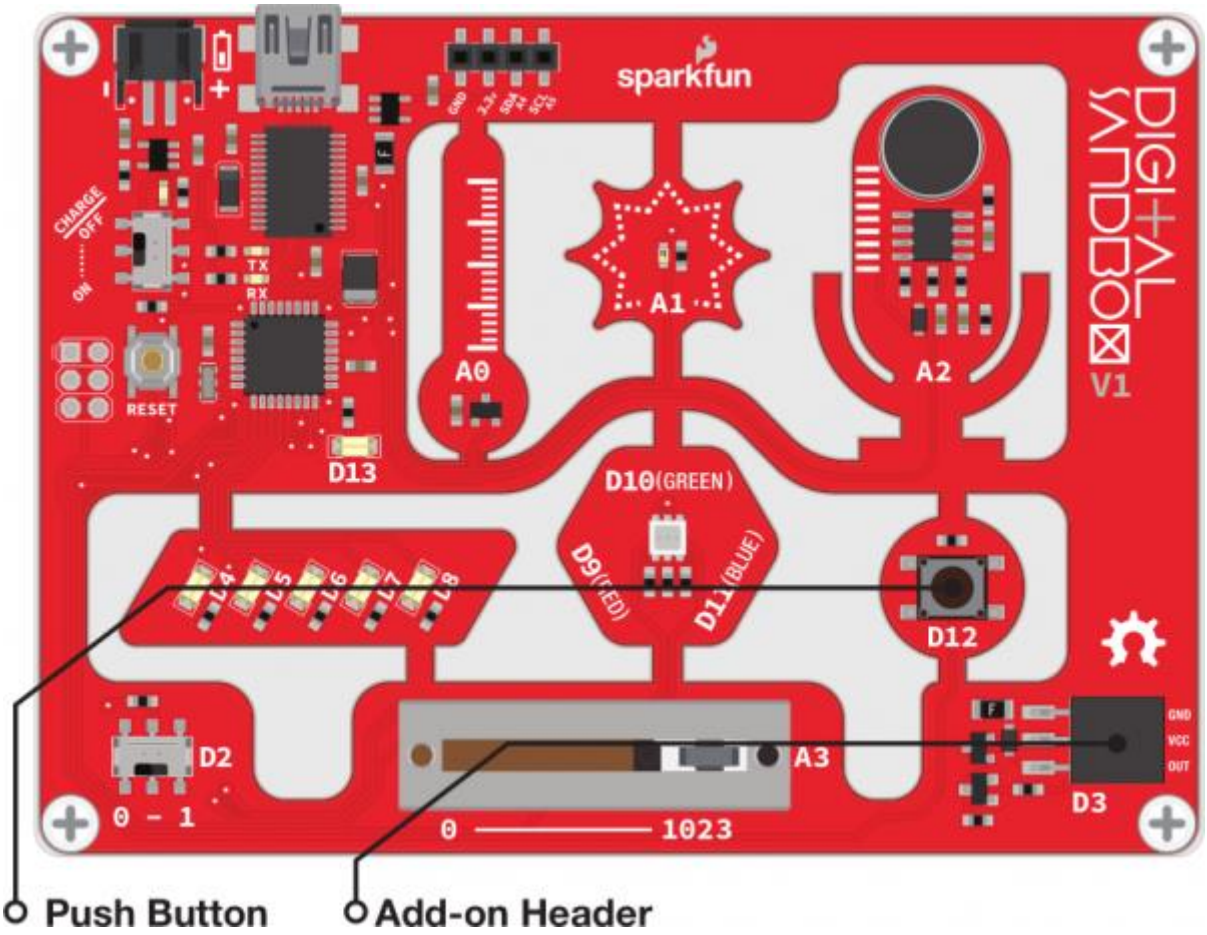
OBS! Detta experiment kräver tillbehörssatsen (Alega art.nr. PR6x) Digital Sandbox Add-On Kit.

## Bakgrund

En DC-motor omvandlar elektrisk energi till roterande, mekanisk energi. DC-motorer är populära eftersom de är mycket enkla att kontrollera: Ge dem lite spänning och de snurrar. Du kan styra motorns hastighet på samma sätt som du kan styra intensiteten hos en LED - med PWM - så i det här experimentet använder vi blocket analog output för att kontrollera motorns hastighet.

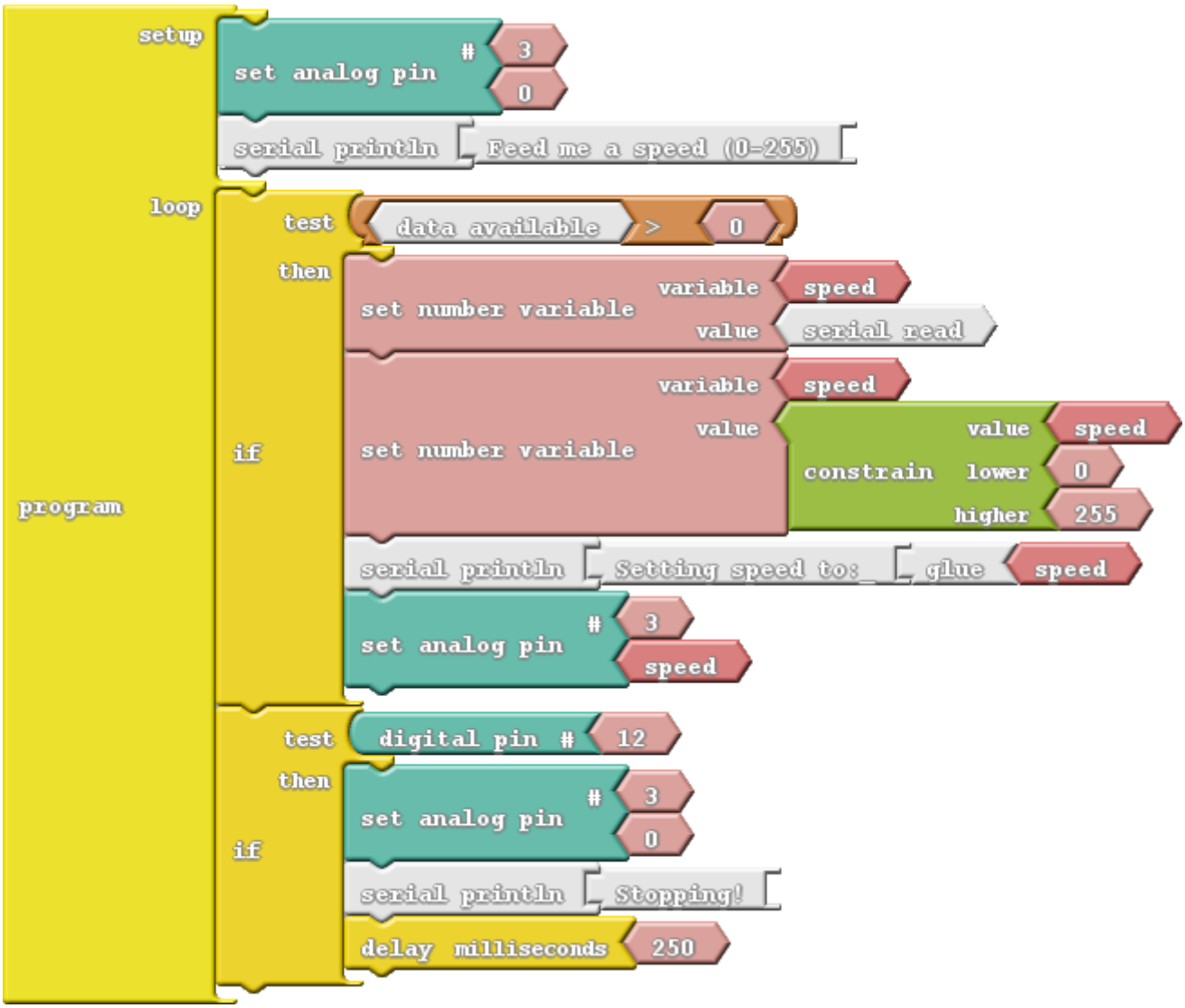
Detta experiment introducerar också serial input. Fram tills nu har kommunikationen med DS-kortet varit mycket ensidiga - det har skickat data till seriell bildskärm (Serial Monitor). Serial input gör att vi kan skicka data till DS-kortet via Serial Monitor.

## Aktiva delar



## Block till koden

Här är schemat för detta program. Set Analog Pin-blocken används för att styra motorn, och en `speed` -variabel används för att hålla koll på motorns hastighet. Några nya block, relaterade till seriell kommunikation, introduceras.

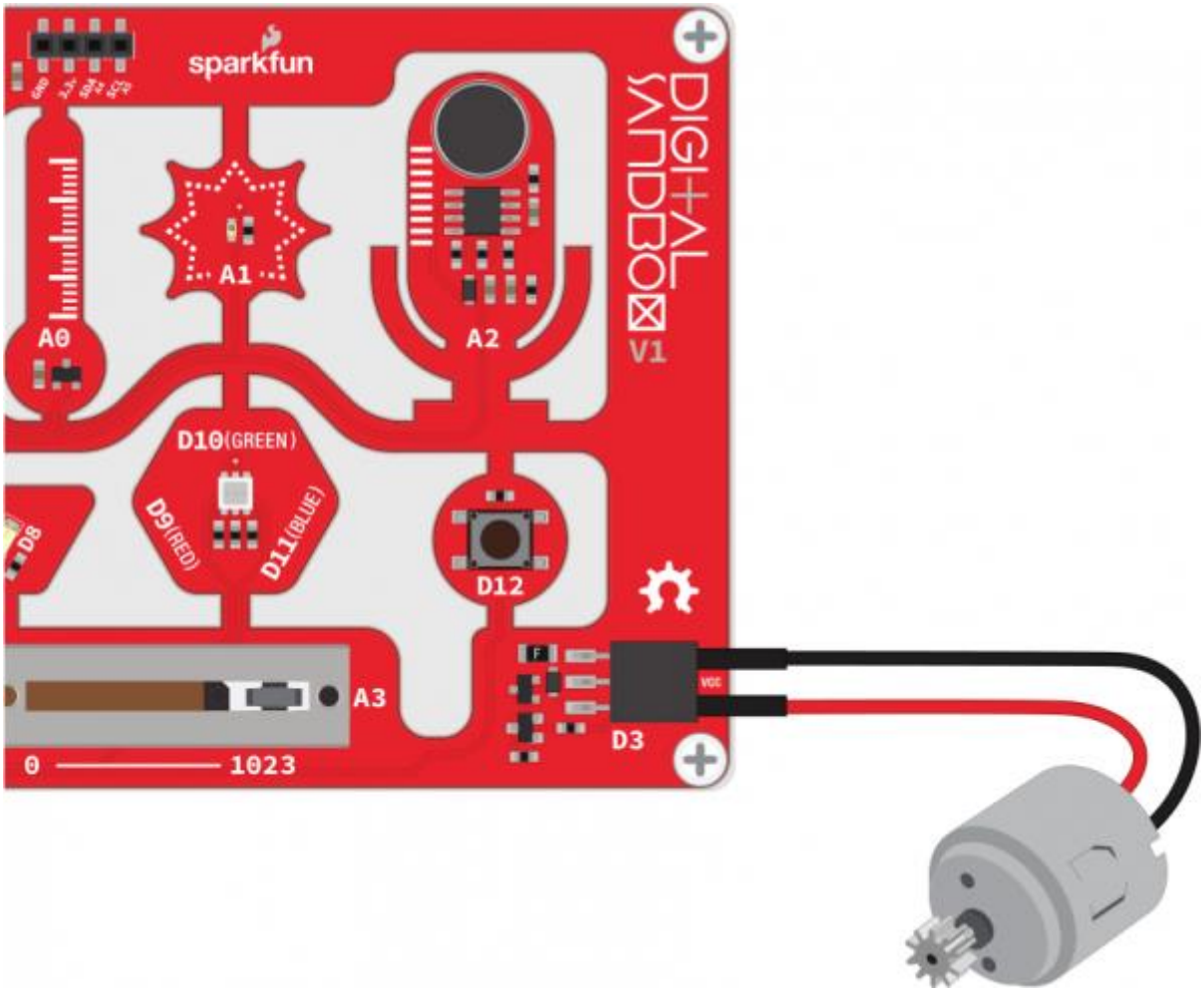



- Serial Read: Vi använder det här blocket för att ange seriella data till Sandboxen. Detta block är som en talvariabel men i stället för att använda ett *Set Number Variable*-block för att ställa in det använder vi *Serial Monitor*. Detta block kommer att lagra det heltal som senast skickats till DS-kortet från *Serial Monitor*.
- Data Available: Detta block håller reda på huruvida seriella data finns tillgänglig eller inte. Om inga seriella data har skickats till DS-kortet lagras värdet noll. Om data skickas till Sandboxen kommer detta block att returnera 1 (en etta). Värdet på detta block återgår till noll när seriella data väl har lästs av (i *Serial Read* -blocket).

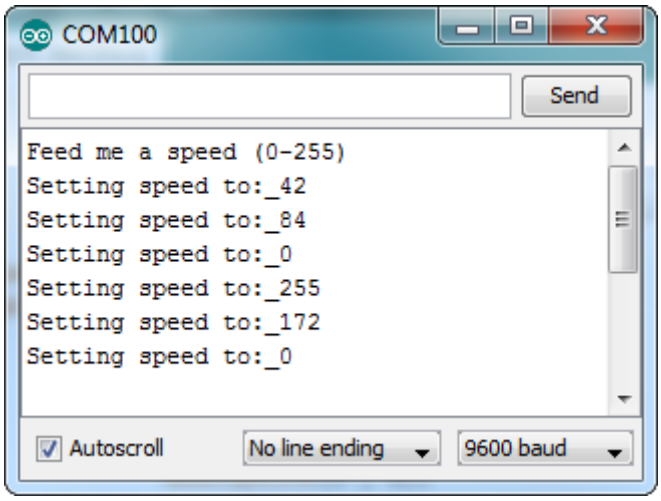
## Gör detta

Konstruera programmet enligt bilden på förra sidan och ladda upp koden.

Efter det att koden har laddats, anslut motorns svarta ledning (GND) till den pin som är märkt GND på kortets extrautgång. Anslut sedan motorns röda ledning till den pin som är märkt OUT på kortets extrautgång. Din motor ska nu vara ansluten till DS-kortet enligt bilden nedan:




Öppna nu Serial Monitor och skriv ett tal mellan noll och 255 i rutan bredvid "Send" och klicka sedan på den knappen. DS-kortet ska svara med ett meddelande, och motorn ska börja snurra.



Vad händer när du skriver 255? Vad sägs om noll? Vad händer när du skickar ett tal som är större än 255 eller mindre än noll (ett negativt)? Kan du lägga till ett block i koden som begränsar dessa värden?

Som en "säkerhetsmekanism", om du någonsin behöver stanna motorn, tryck på knappen för att stoppa den.

## Upptäck mer

- Försök ansluta något till motorn. Kanske tejpa fast ett smalt papper för att skapa en propeller. Eller flera pappersbitar för att skapa en fläkt. Vad mer kan du ansluta till axeln för att dra nytta av rotationsrörelsen?
- Som en extra utmaning kan du försöka koda programmet så att motorn ändrar hastigheten smidigare och snabbare när ett nytt seriellt värde tas emot?

# 16: Styra ett servo (kräver tillbehörssatsen)

DC-motorer är bra för att få ett föremål att rotera med hög hastighet utan hänsyn till i vilken position den startar eller stannar. För många applikationer är det dock viktigt att exakt styra motorns position. Vingklaffarna på ett flygplan, styrmekanismer i RC-bilar och industrirobotar är exempel där man ofta kräver en motoriserad positionsstyrning. För dessa applikationer ratar vi den vanliga DC-motorn och välkomnar servomotorn!

OBS! Detta experiment kräver tillbehörssatsen (Alega art.nr. PR6x) Digital Sandbox Add-On Kit.

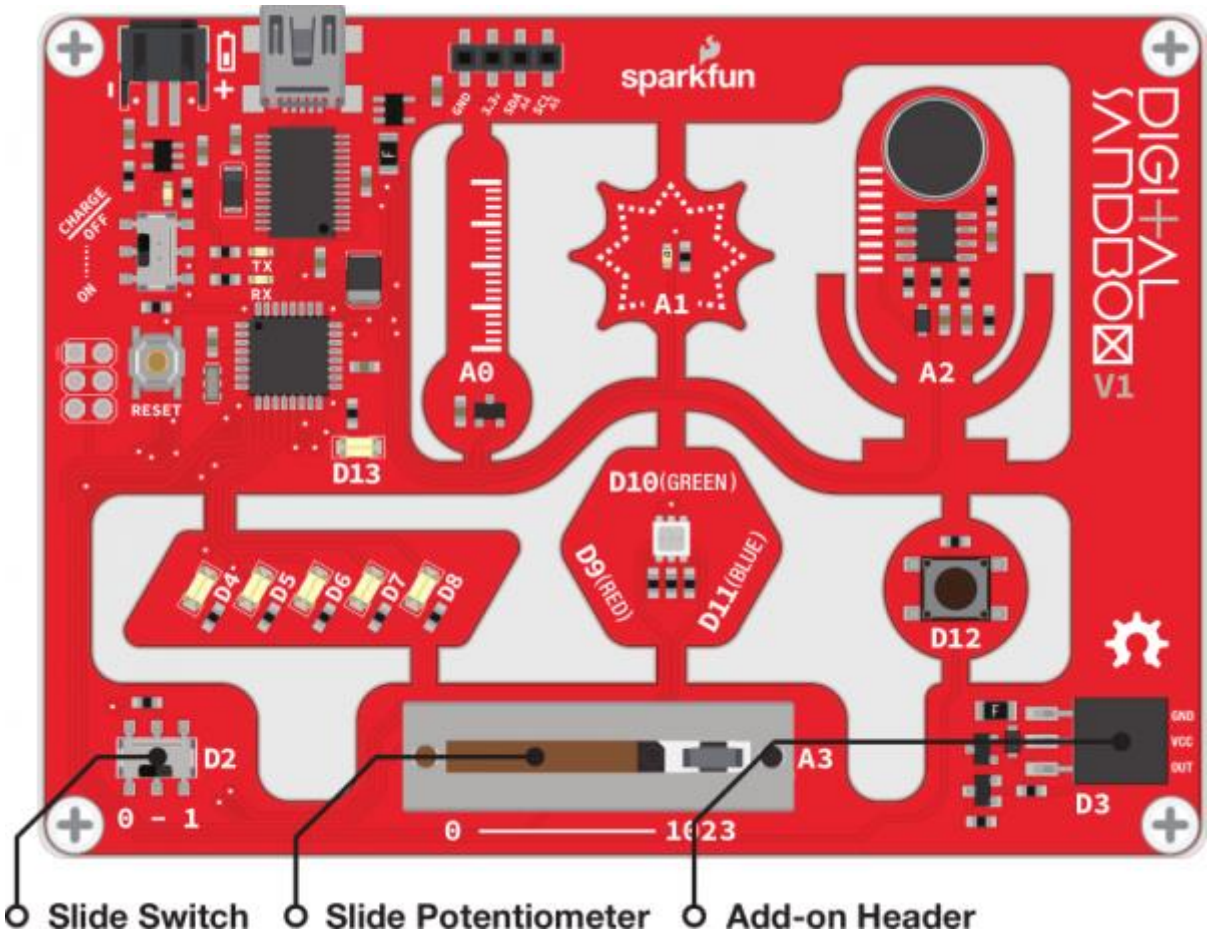
## Bakgrund

En servomotor är som en likströmsmotor med en intern styrenhet och inbyggda sensorer som hjälper till att hålla koll på axelpositionen. En servomotor känner till exempel om den pekar på 15° eller 115°.

Alla servo har tre ledningar som behöver anslutas till: matningsspänning, jord och en signal. Spännings- och jordanslutningarna ger motorn ström, och styrsignalen - en PWM-utgång (förvånad?) - anger servomotorns position. När servomotorn når önskad position, stannar den tills den beordras att flytta till en ny position.

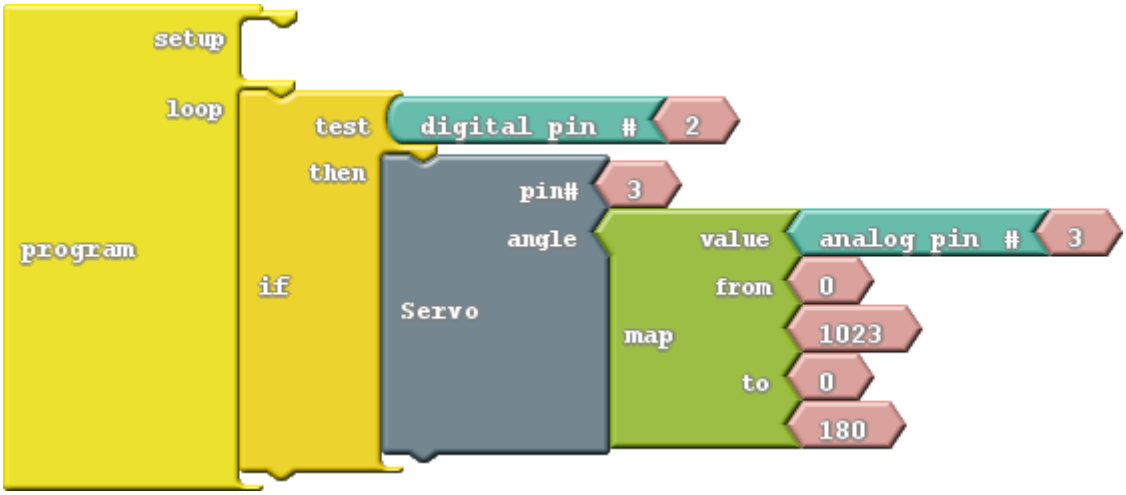
Servomotorer varierar I det intervall de kan positionera sig i, det vill säga den största och den minsta vinkel som de kan peka på. Endast specialiserade, kontinuerliga rotationsservo kan rotera ett helt varv, 360°. De flesta har ett angivet rörelseområde mellan 90° och 180°. Den servomotor som vi använder i detta experiment har ett rörelseområde på 180°.

## Aktiva delar



## Block till koden

I detta experiment introduceras blocket *Servo*. Så här ser koden ut:

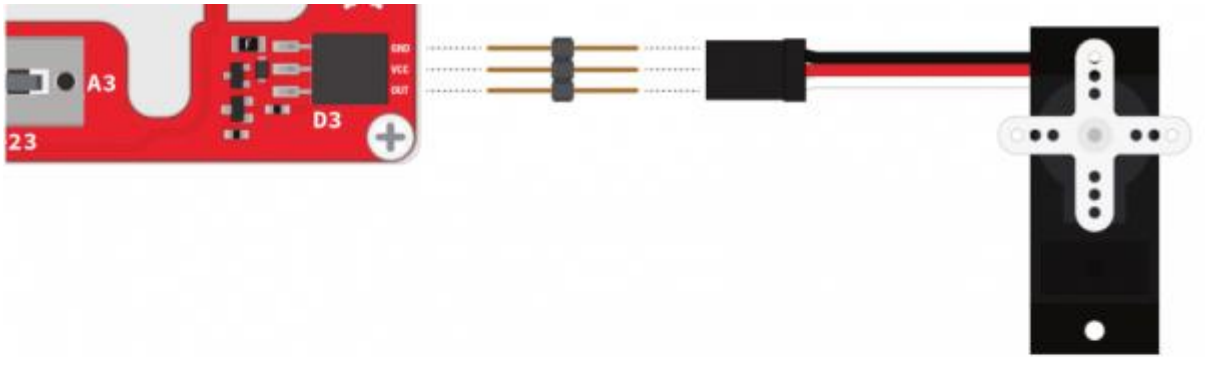


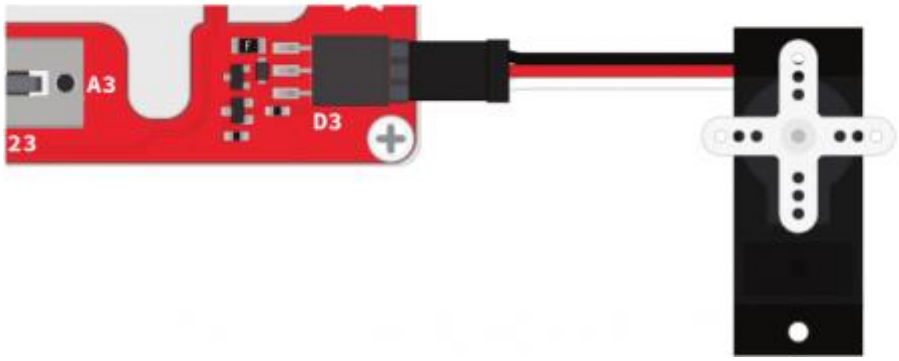

- Servo: Med detta block, som finns under ikonen *Utilities*, kan du flytta ett servo till en inställd position. Det finns två ingångar till detta block: pin-numret och en vinkel. Servos kan anslutas till valfri pin, vi använder extrautgången D3 i det här exemplet. Vinkeln ska vara ett tal mellan noll och det maximala värde ditt servo kan ha. I det här fallet begränsar vi intervallet mellan noll och 180 med blocket *Map*.

## Gör detta

Konstruera programmet enligt bilden ovan och ladda det till DS-kortet.

Efter att koden har laddats upp, anslut servot med hjälp av trestiftskontakten till kortet. Se till att servomotorns svarta ledning (GND) ansluts till GND på kortet enligt bilden:



Med detta program kan du styra servomotorns position med glidpotentiometern. Skjut hela vägen till höger för att ställa in servot på 180° och hela vägen till vänster för att ställa det på 0°.

Servo flyttas endast om strömbrytaren (ansluten till pin 2) är inställd på ON. Om du lämnar strömbrytaren i läge ON kan du se hur snabbt motorn svarar på *servo*-blocket. Om du flyttar strömbrytaren till OFF, ställ in glidpotentiometern läge och sätt strömbrytaren till ON, så kan du se hur snabbt servomotorn går och hur den styrs.

## Upptäck mer

- Vad händer om du försöker rotera servot över 180° (ändra det sista värdet i *Map*-funktionen)?
- Fundera på att koppla något till servot. Du kan säkert hitta på någon kul sak att vifta med!

# Göra mer

Grattis till slutförandet av DS-Guiden, experimenthandboken för Digital Sandbox! Om du känner dig bekväm med ArduBlock, och vill fortsätta utmana dig själv, testa Digital Sandbox Guide for Arduino https://learn.sparkfun.com/tutorials/digital-sandbox-arduino-companion


*Återupplev minnena av experiment 12! Den här gången med Arduino-kod.*

I den handledningen återskapar vi samma experiment som från den här guiden men med det faktiska programmeringsspråket Arduino. Säg adjö till block och hej till semikolon!