NYBÖRJARKURS · 5 TRÄFFAR

Elektronik & Programmering med Arduino

Fem träffar. Fem moduler. Ett fungerande larm.

FRO Ånge · 13 april 2026 · FRO-lokalen

ELEGOO UNO
Basic Starter Kit

KURSÖVERSIKT

Fem träffar. Fem moduler.

MODUL 01

LED & krets

Digital output, Ohms lag

MODUL 02

PWM & RGB

analogWrite, färgblandning

MODUL 03

Digital input

Knapp, buzzer, logik

MODUL 04

Analog input

Sensorer, Serial Monitor

MODUL 05

Integration

Hackathon: bygg larmet

Varje träff bygger vidare på den föregående.

TRÄFF 1 AV 5

LED & krets

Modul 1: **Digital output, Ohms lag**

DAGENS UPPLÄGG

Idag — två timmar.

00:00 – 00:15

Slutmålet & Arduinon

Vi tittar på det färdiga larmet och ramar in vad ett Arduino-kort egentligen är.

00:15 – 00:35

Krets, spänning, ström

Kort genomgång av vad en krets är och varför LED:en behöver en resistor.

00:35 – 01:05

Bygg kretsen & ladda upp första programmet

Praktiskt: LED + resistor på breadboarden, första uppladdningen från IDE:n.

01:05 – 01:10

Paus

Vatten, WC, sträck. Tillbaka fem minuter senare.

01:10 – 02:00

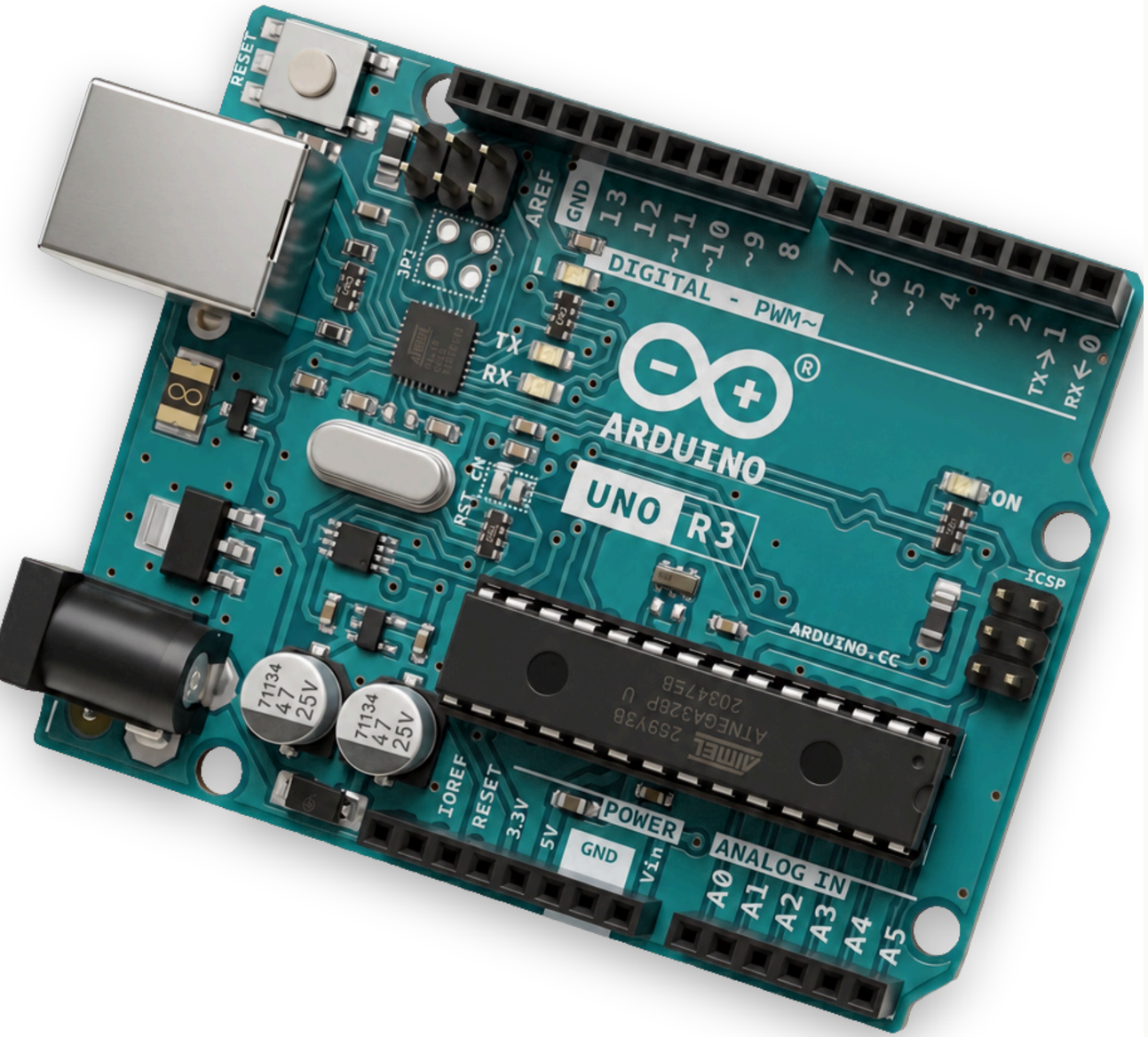
Din egen rytm & delresultat

Fri övning: ändra delay-värdena till ett eget mönster. Alla visar upp.

SLUTMÅLET

Det här ska vi bygga.

Ett larm som känner av mörker, lyser upp som stämningsljus och tjuter när någon rör det.



ARDUINO UNO R3

Mikrokontrollern.

En mikrokontroller — processor, minne och in-/utgångar på ett chip.
Kör ett enda program, helt förutsägbart.

Arduino UNO R3 · ATmega328P
16 MHz · 32 KB flash · 2 KB SRAM

TRE GRUNDBEGREPP

Spänning, ström, GND.

SPÄNNING

Volt · V

STRÖM

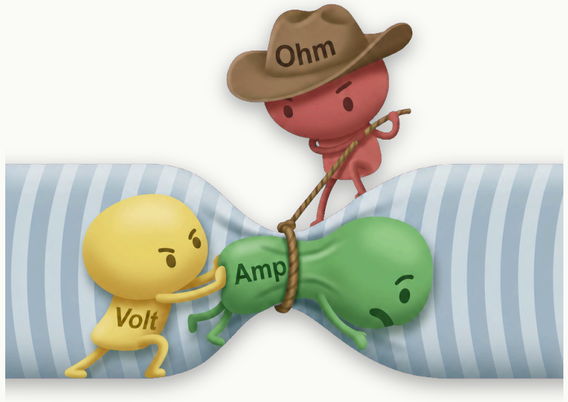
Ampere · A

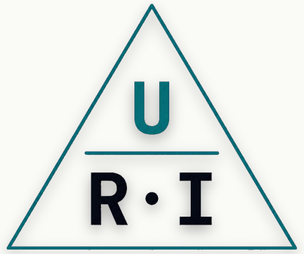
GND

0 V · referens

OHMS LAG

$U = R \cdot I$





täck variabeln ni söker

Volt trycker · Ohm stryper · Amp flödar

Räkneexempel och framspänningsfall: → Bilaga F i kompendiet

KRETSEN

Kretsen — som ett vattensystem.

Strömmen går ut från en utgångspinne (t.ex. pin 13 när den är HIGH), genom komponenterna, och tillbaka till GND.

Utan bromsning: **LED:en brinner upp**.

Lösningen: en **strypventil** — en resistor på 220 Ω i serie.

> MINNESREGEL
> Ingen krets utan väg tillbaka till GND. Strömmen måste kunna fullborda varvet — annars händer ingenting.

KOMPONENTERNA

LED & resistor.

LED:en

- **Har polaritet.** Kör du den baklänges lyser den inte.
- **Långt ben = anod (+)** → till signalpinnen (pin 13).
- **Kort ben + platt kant = katod (–)** → till GND.
- Tål ca **20 mA**. Mer → den brinner upp permanent.

Kör du den baklänges går den inte sönder — den lyser bara inte. Bara att vända.

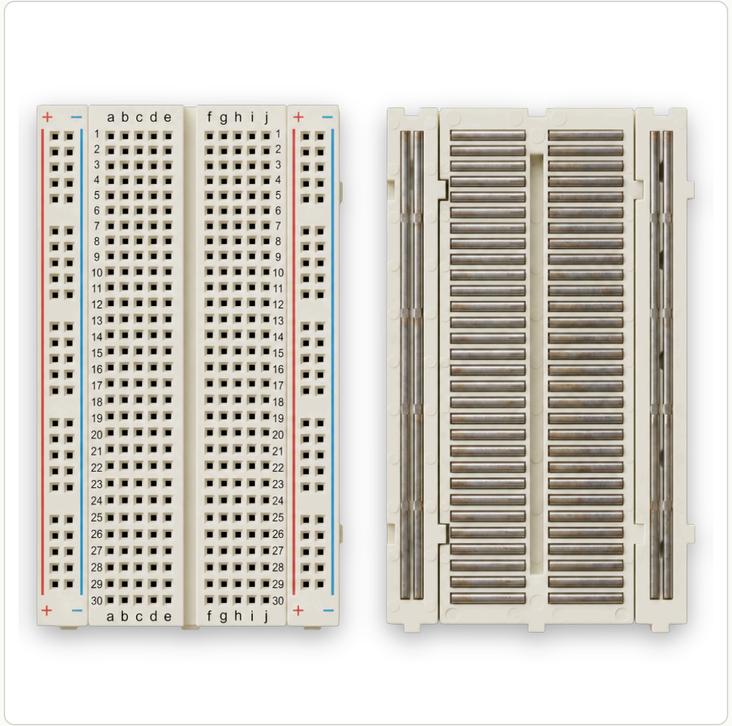
Resistorn

- **Passiv komponent.** Ingen polaritet — koppla hur du vill.
- Värdet läses på **färgringarna**.
- **220 Ω** (4-band) = `röd · röd · brun` + guld.
  220 Ω (5-band) = `röd · röd · svart · svart` + brun.

röd · röd · brun
2 · 2 · ×10¹
= 220 Ω

BREADBOARDEN, STEG 1

Fem hål = en nod.



- **Fem hål i rad** = en nod
- Raden bredvid = **egen** nod
- Gapet i mitten **bryter**
- +/−-skenorna går hela vägen

FELSÖKNING · REGEL #1

Kontrollera raden först.

90 % av alla nybörjarfel
är ett hål fel.

Kabeln och komponentbenet måste
sitta på **samma rad**.

PRAKTISKT MOMENT

Bygg kretsen.

01 Sätt LED:en på breadboarden. **Långt ben** i en ledig rad, **kort ben** i raden bredvid.

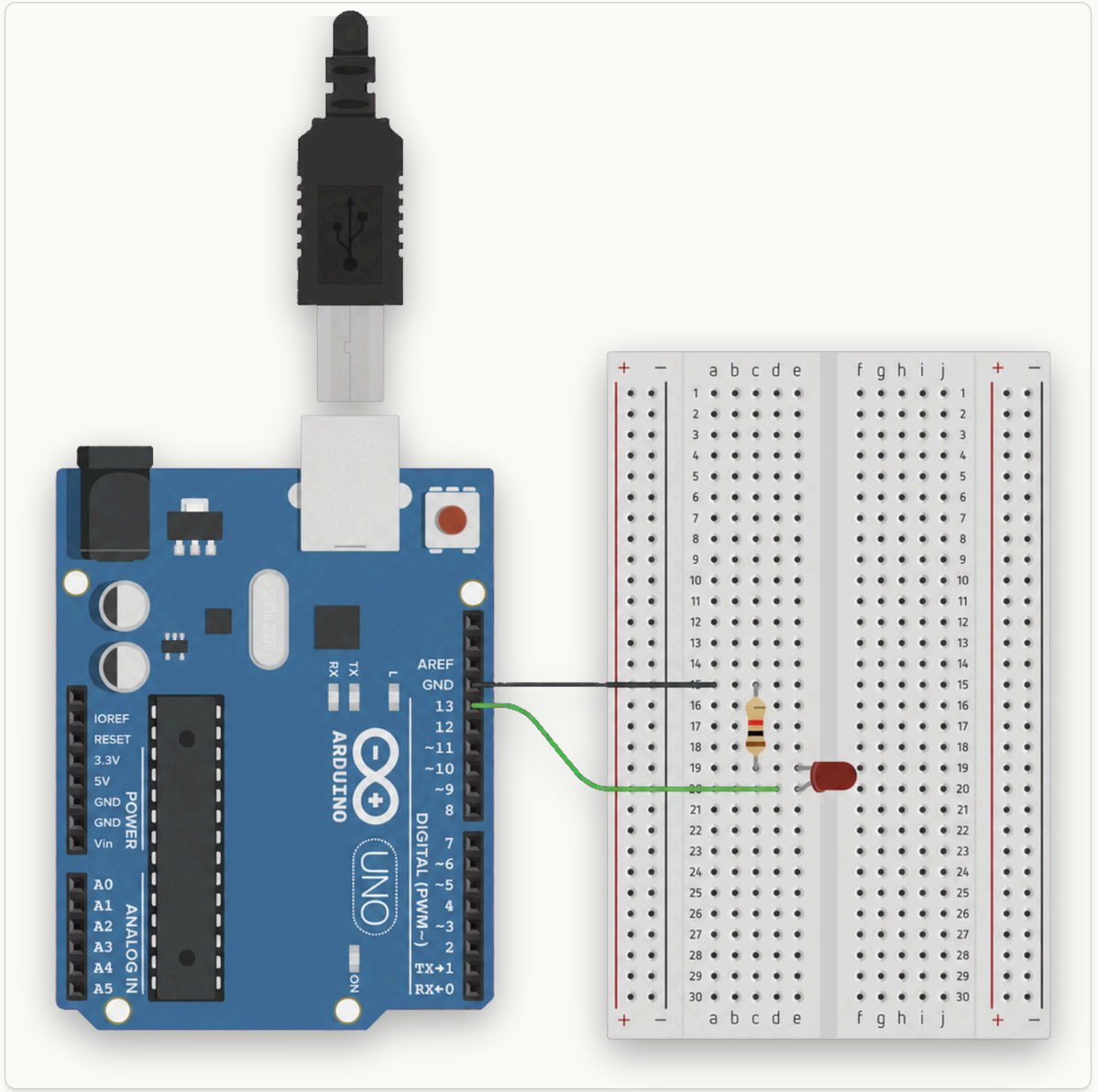
02 Kabel från **pin 13** på Arduinon → samma rad som långa benet.

03 **220 Ω-resistor** (4-band: röd-röd-brun; 5-band: röd-röd-svart-svart-brun). Ett ben i samma rad som korta LED-benet. Andra benet i valfri ledig rad.

04 Kabel från resistorns andra ben → **GND** på Arduinon.

05 Ladda upp `File → Examples → 01.Basics → Blink`. Lampan ska börja blinka.

Så här ska det se ut.



D13 → LED långt ben · LED kort ben → 220 Ω → GND

KODENS ANATOMI

`setup()` och `loop()`.

void setup()

Körs **en gång** när Arduinon startar eller när ni laddar upp. Används för konfiguration: vilka pinnar är utgångar, hastighet på Serial, etc.

void loop()

Körs **om och om igen**, så länge kortet har ström. Här bor logiken: läs, reagera, vänta, upprepa.

Regel: exakt en `setup()` och en `loop()` per Arduino-kod. Aldrig fler.

```
1  void setup() {
2    pinMode(LED_BUILTIN, OUTPUT);  // körs en gång vid uppstart
3  }
4
5  void loop() {
6    digitalWrite(LED_BUILTIN, HIGH);  // körs om och om igen
7    delay(1000);
8    digitalWrite(LED_BUILTIN, LOW);
9    delay(1000);
10 }
```

TRE KOMMANDON

Hela Blink — på tre rader.

```
pinMode(LED_BUILTIN, OUTPUT);
```

"`LED_BUILTIN` är en utgång." Arduinons alias för `pin 13` — där den inbyggda LED:en sitter. Körs i `setup()`.

```
digitalWrite(LED_BUILTIN, HIGH);
```

"Sätt pinnen till 5 V." `HIGH` = tänd (5 V), `LOW` = släckt (0 V). Körs i `loop()`.

```
delay(1000);
```

"Vänta 1000 millisekunder." 1000 ms = 1 sekund. Arduinon pausar allt annat under tiden.

Med bara dessa tre kan ni få en lysdiod att blinka i vilken rytm som helst.

UPPGIFT

Din egen rytm.

Ändra `Blink` till ett mönster du själv väljer.

- **SOS-rytm:** · · · — — — · · ·
- **Ditt eget tempo** — snabb eller långsam, ojämn
- **Hjärtslag** — två snabba, sen paus

TIPS FÖR BYGGET

Lägg tider i `const int`-variabler — då slipper du ändra många siffror när du justerar tempot. SOS växlar mellan kort blink (200 ms) och lång blink (600 ms), separerade av paus.

DELRESULTAT · TRÄFF 1

KLART

Ni har byggt en blinkare.

Er första rad kod som styr den fysiska världen.

Er första krets
LED + resistor + GND

Er första kod
setup · loop · tre rader

Er egen rytm
kod som styr världen

Ett varv runt rummet — visa upp ditt mönster för grannen.

NÄSTA TRÄFF

Nästa gång...

Ni kan tända en lampa i en egen rytm.
Nästa steg: blanda färg ur rött, grönt och blått.

TRÄFF 2 AV 5

PWM & RGB

Modul 2: **analogWrite & färgblandning**

TRE KANALER, ALLA FÄRGER

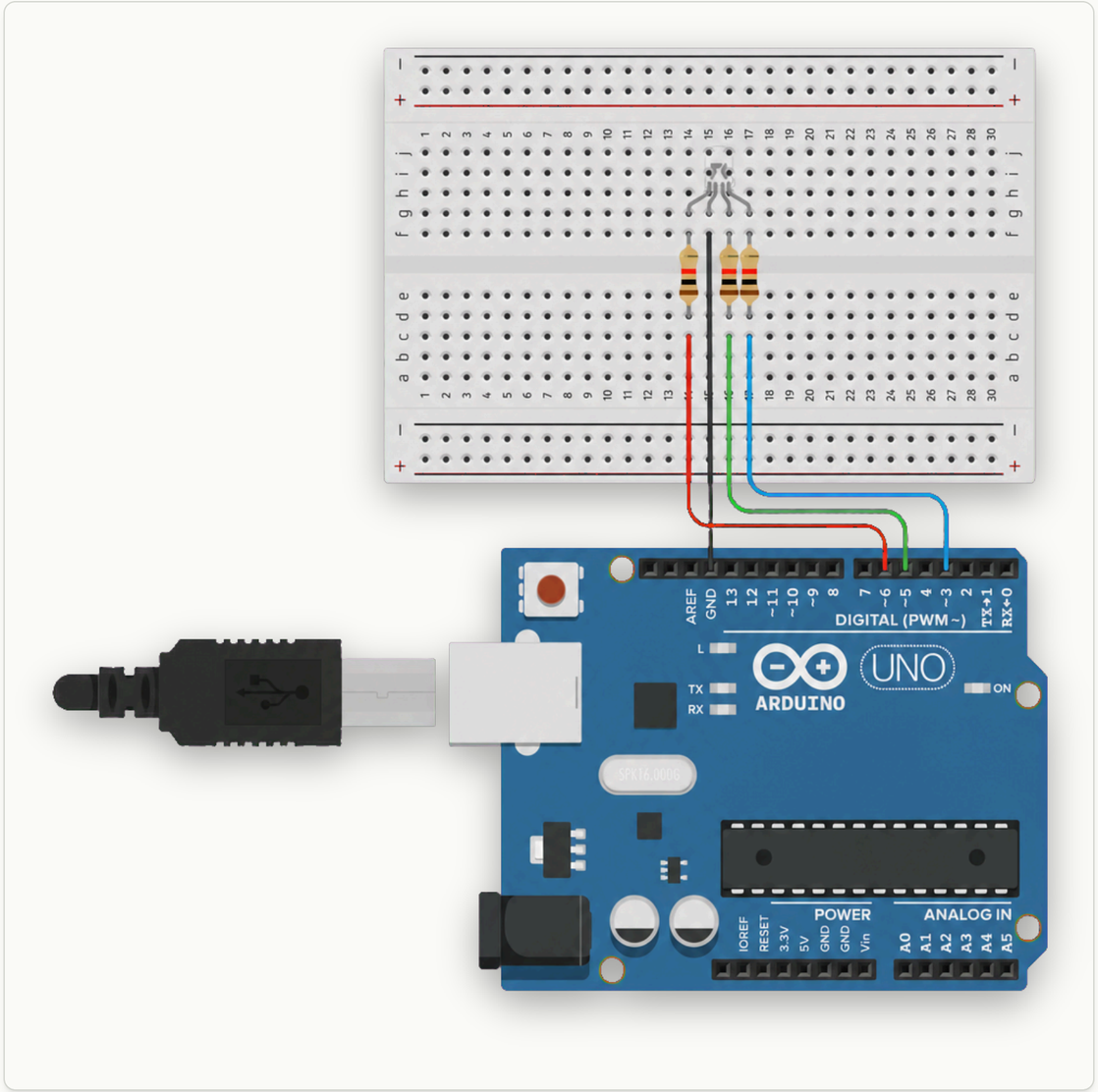
Alla färger, av bara tre.

Skärmen ni tittar på just nu har miljoner pixlar, och varje pixel är bara en röd, en grön och en blå punkt. Idag programmerar vi er egen pixel.

VIKTIGT — RGB-LED:EN

Common Cathode. 4 ben i rad.
Ordning från platta sidan: `Röd` · `Katod` · `Grön` · `Blå`
Katoden är **längst**, sitter **andra från platta sidan**, och går till GND.

Koppla RGB-LED:en.



R → D6 · G → D5 · B → D3 · katod → GND · varje färg via egen 220 Ω.

Gör detta först: dra en kabel från Arduinons GND till `minus-skenan` på breadboarden. Då har alla komponenter en minus-väg.

NYTT KOMMANDO · MELLAN AV OCH PÅ

`analogWrite`.

`digitalWrite` kunde bara två saker: `HIGH` eller `LOW`.

`analogWrite` tar ett tal från `0` till `255`. Allt däremellan = PWM.

```
const int ledR = 6, ledG = 5, ledB = 3;

analogWrite(ledR, 200);  // röd  hög
analogWrite(ledG,   0);  // grön av
analogWrite(ledB, 200);  // blå  hög  → lila
```

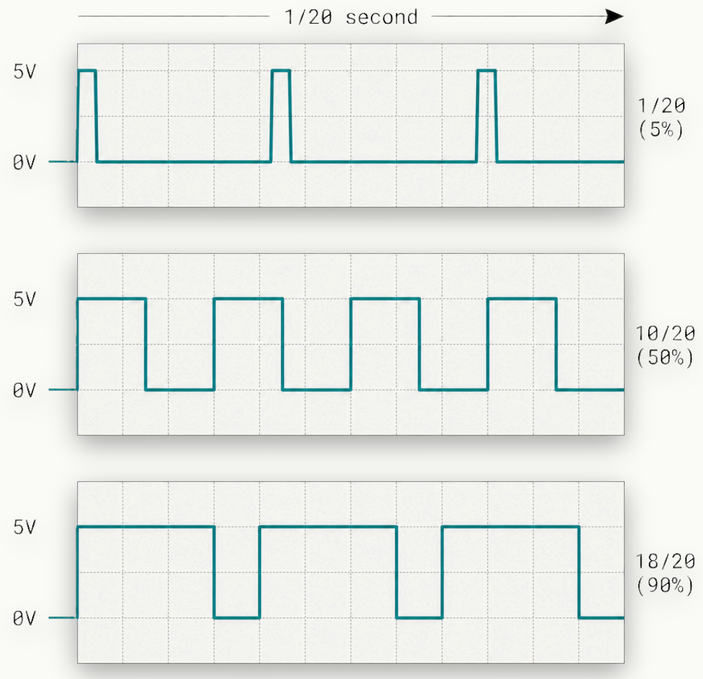
PWM

Duty cycle.

`0` → alltid LOW → släckt

`128` → 50 % PÅ-tid → halvstyrka

`255` → alltid HIGH → full styrka

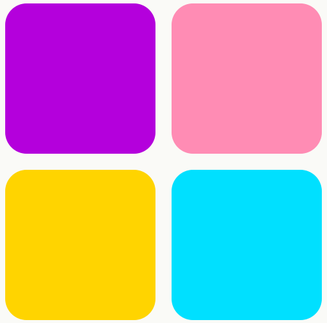


ÖVNING

Hitta färgen.

Experimentera med värden `0–255` på varje kanal. Ladda upp, titta, justera.

- **Lila** — röd + blå, ingen grön
- **Gammelrosa** — mycket röd, lagom blå, lite grön
- **Skolgul** — full röd, lagom grön, ingen blå
- **Cyan** — ingen röd, full grön + blå



> STRATEGI
>
> Prova flera kombinationer — färgerna är ofta överraskande.

EFTER TRÄFF 2

Ni har en pixel.

Ni kan blanda alla färger ur tre kanaler, och ni har lärt Arduinon att leverera något *mellan* helt av och helt på.

Nästa gång: vi läser av världen och svarar. Knapp + buzzer.

TRÄFF 3 AV 5

Digital input

Modul 3: **Knapp, buzzer, logik**

NAMNGIVNA VÄRDEN

`const int` och `int`.

Ni har redan använt `const int` — för pin-nummer.

Nu behöver ni också `int` (utan `const`) — för värden som **ska kunna ändras under tiden programmet kör**: räknare, tillstånd, sensormätningar.

> TUMREGEL
> `const int` = sätts en gång, aldrig mer.
> `int` = kan ändras.
> `bool` = sant eller falskt.

```
1 const int knappPin = 9;     // ändras aldrig
2 int lastState = HIGH;       // kan ändras
3 bool larmPaslaget = false; // kan togglas
```

Full förklaring av datatyper, scope och operatorer i kompendiets Bilaga A.

DIGITAL INPUT

Input ≠ Output.

Hittills har Arduinon bara skickat ström ut (OUTPUT).
Nu läser den av ett tillstånd in (INPUT).

```
pinMode(knappPin, INPUT_PULLUP);
// digitalRead(knappPin):
//   LOW = tryckt, HIGH = släppt
```

VARFÖR PULLUP?

En pinne som bara "lyssnar" utan koppling till plus eller minus *flyter* — läsningen blir slumpmässig. `INPUT_PULLUP` låter Arduinon själv dra pinnen till HIGH. När du trycker dras den till GND (LOW). Därför: **tryckt = LOW**, släppt = HIGH.

FATTA BESLUT

`if` / `else`.

Läs ett värde → jämför → agera olika beroende på resultatet.

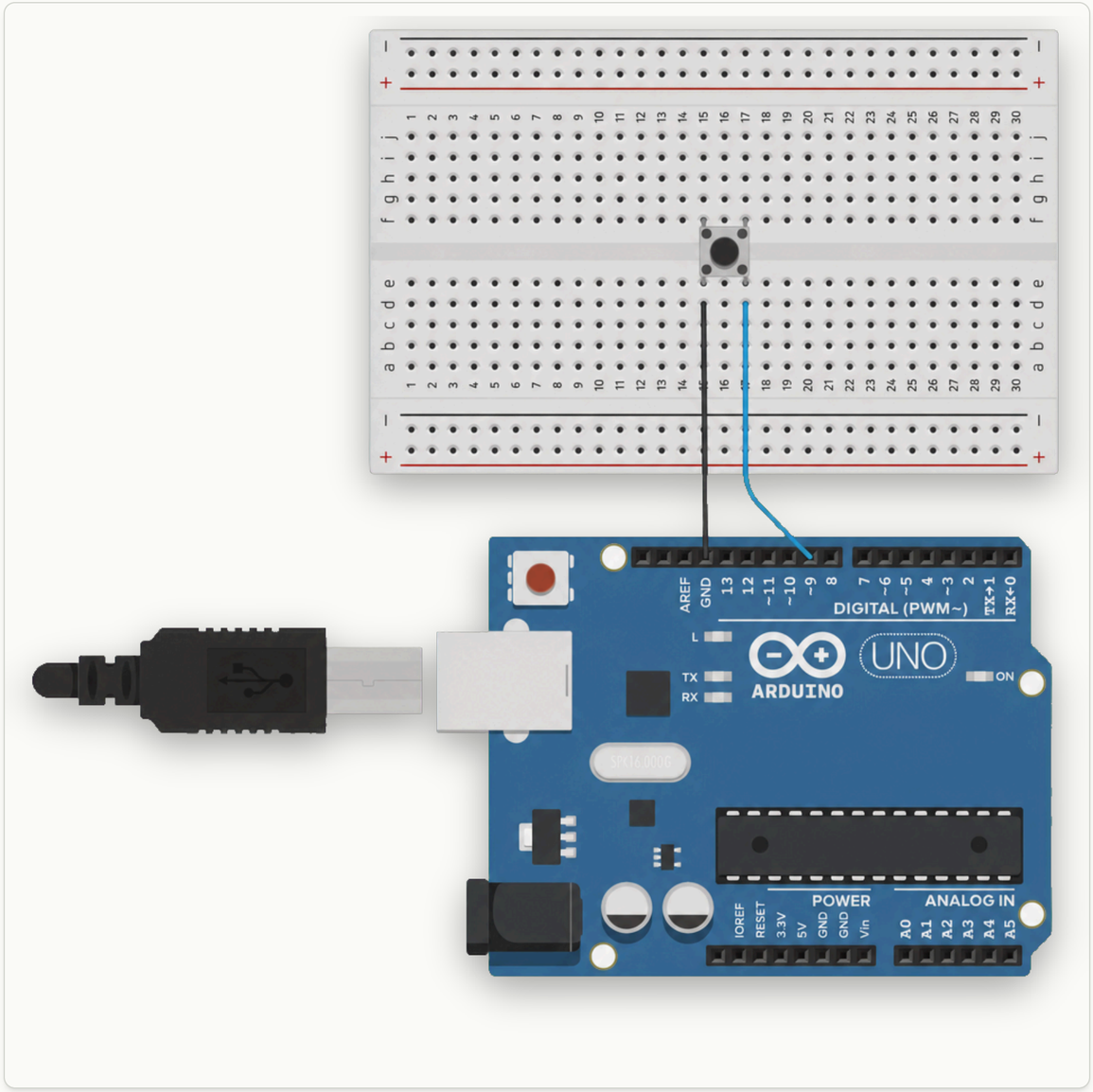
Operatorerna: `==` lika, `!=` olika, `<` mindre, `>` större.

> **VANLIGASTE NYBÖRJARFELET**
>
> = tilldelar värde.
> == jämför.
> Glöm inte andra likhetstecknet.

```
1  if (digitalRead(knappPin) == LOW) {
2    // knappen är tryckt
3    digitalWrite(LED_BUILTIN, HIGH);
4  } else {
5    // släppt
6    digitalWrite(LED_BUILTIN, LOW);
7  }
```

Fler operatorer (&&, ||, !)
och kedjade if/else i kompendiets Bilaga A.

Koppla knappen.



Vi använder bara `knapp A → D9` · andra benet till GND. Ingen pulldown-resistor — `INPUT_PULLUP` sköter det internt.

PEDAGOGISKT KRITISK

Reagera på flanken.

En knapp som **hålls nere** skulle toggla hundratals gånger per sekund. Resultat: larmet blinkar mellan av/på som ett stroboskop.

Lösningen: agera inte på att knappen *är* nere. Agera på att den **just nu gick från HIGH till LOW** — flanken.

> MÖNSTRET
> Spara förra värdet. Jämför med nuvarande. Om det ändrats åt rätt håll → det är en flank → agera en gång.

Inbyggda LED:en (pin 13) speglar `larmPaslaget` — så ni *ser* när larmet växlar.

```
1  bool larmPaslaget = false;
2  int lastState = HIGH;
3
4  void loop() {
5    int state = digitalRead(knappPin);
6    if (state == LOW && lastState == HIGH) {
7      larmPaslaget = !larmPaslaget;        // flank → toggla
8    }
9    digitalWrite(LED_BUILTIN, larmPaslaget ? HIGH : LOW);
10   lastState = state;
11   delay(10);  // mot studs
12 }
```

ACTIVE BUZZER

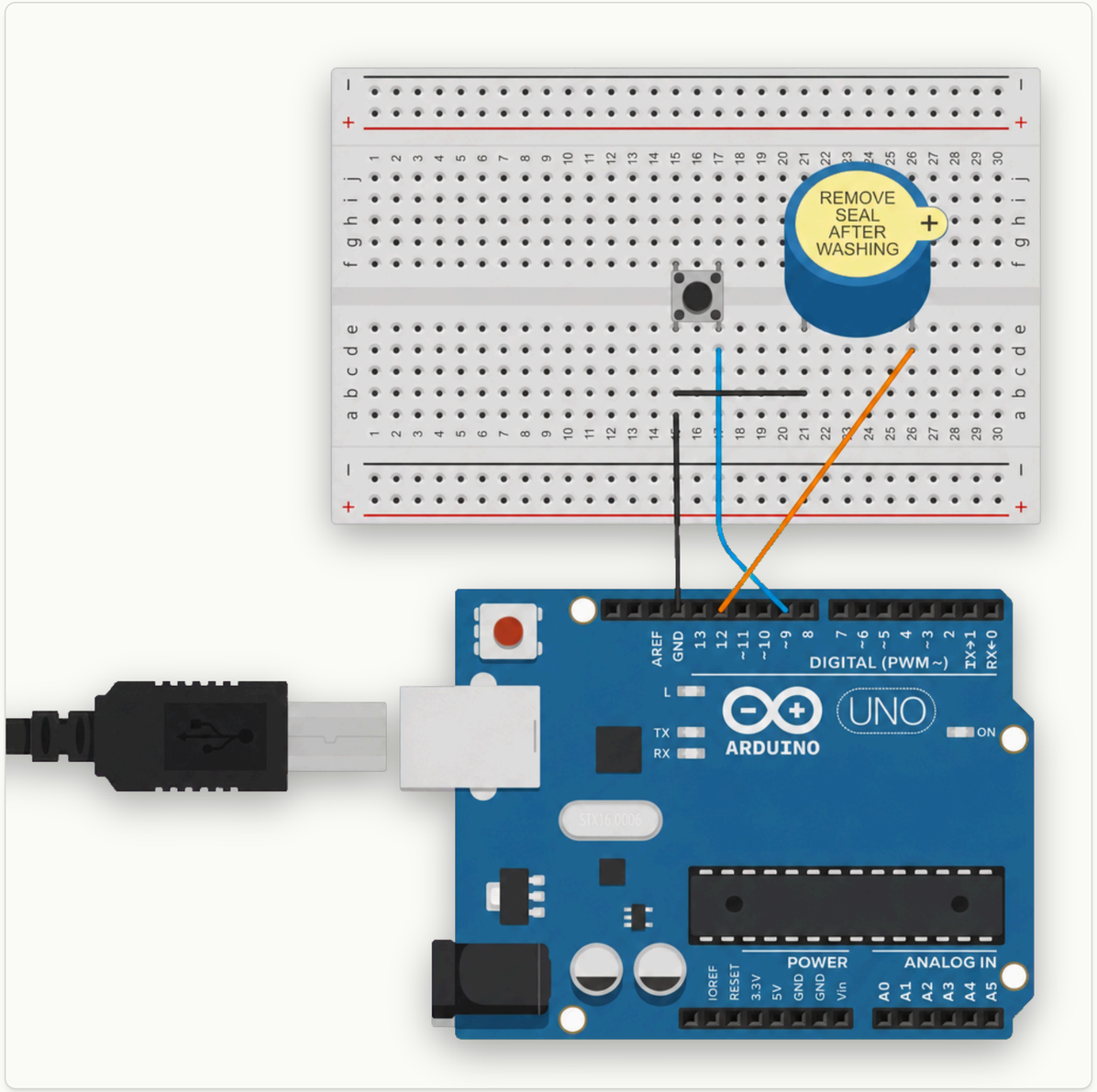
En varning först.

EXTREMT VIKTIGT

Dra absolut inte av den lilla klisterlappen `"REMOVE SEAL AFTER WASHING"` på er buzzer.

Utan lappen är buzzern obehagligt högljudd.
Lappen är vår ljuddämpare — den stannar på.

Koppla buzzern.



+-ben (höger) → D12 · −-ben → GND-skenan. Knappen står kvar — buzzern är tillägget. Ingen resistor.

SÄTT IHOP ALLT

Läs knapp. Styr buzzer.

Tryck → larm på (buzzern tjuter).
Tryck igen → tyst.

`INPUT → flank → OUTPUT`

SÄTT IHOP ALLT · LARM-MÖNSTRET

Kärnan i `loop()`.

```
int state = digitalRead(knappPin);
if (state == LOW && lastState == HIGH) {
  larmPaslaget = !larmPaslaget;           // flank → toggla
}
digitalWrite(buzzerPin, larmPaslaget ? HIGH : LOW);
lastState = state;
```

Samma mönster som "Reagera på flanken" — `LED_BUILTIN` bytt mot `buzzerPin`. Active buzzer: `HIGH` = pip, `LOW` = tyst.

EFTER TRÄFF 3

Arduinon lyssnar.

Ni har byggt Arduinons första sensor — en knapp — och gett den en röst — buzzern. Arduinon läser den ena — och styr den andra.

Nästa gång: analoga sensorer. Mörker, rörelse, mätvärden. Serial Monitor.

TRÄFF 4 AV 5

Analog input

Modul 4: **Sensorer & Serial Monitor**

ANALOG INPUT

Knappen var digital. Världen är analog.

Knappen: **PÅ eller AV.**

Ljuset: **halvmörkt, starkt, svagt.**

`analogRead(A0)` ger 0–1023

NYTT KOMMANDO

`analogRead()` — Arduinons linjal.

```
1  int ljus = analogRead(A0);
```

`0` → 0 V → helt mörkt

`1023` → 5 V → fullt ljus

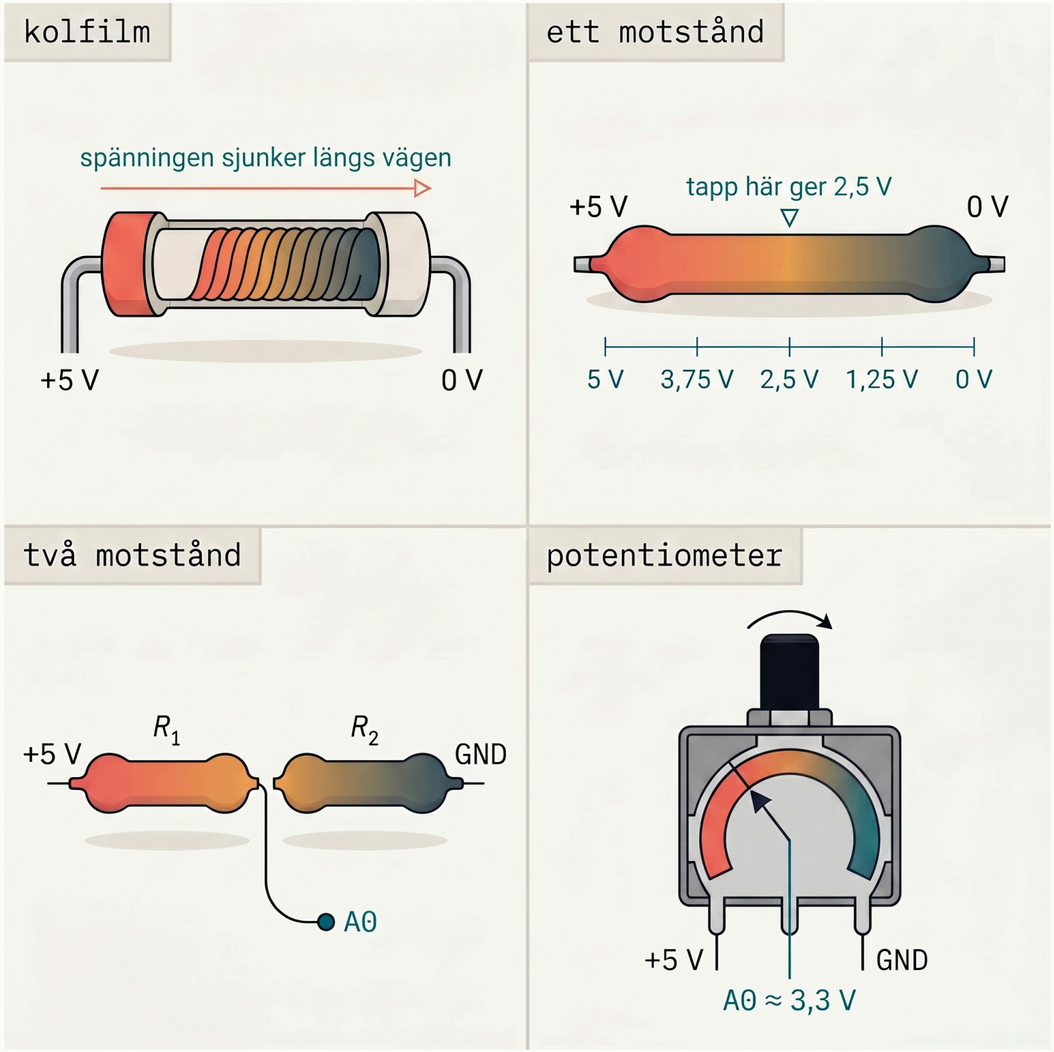
Allt däremellan = mellanting.

GÖR DETTA NU

Bygg **spänningsdelaren:**
fotocell + 1 kΩ → pin `A0`

Fotocell (LDR): 50 kΩ i mörker · 500 Ω i solljus.
Starkt ljus → A0 läser högt värde.

Tänk dig ett låångt motstånd.



Inuti motståndet sitter en **kolfilm** kring en keramisk kärna. Strömmen kämpar sig igenom — och spänningen sjunker **linjärt** längs vägen.

Två motstånd i serie = samma sak, bara delat i två. Skarven är mätpunkten. `A0` läser spänningen där.

En **vridpotentiometer** är detta i hårdvara: tre ben, och ratten flyttar wipern längs det inre motståndsspåret. Vrid → `A0` sveper jämnt 0–1023.

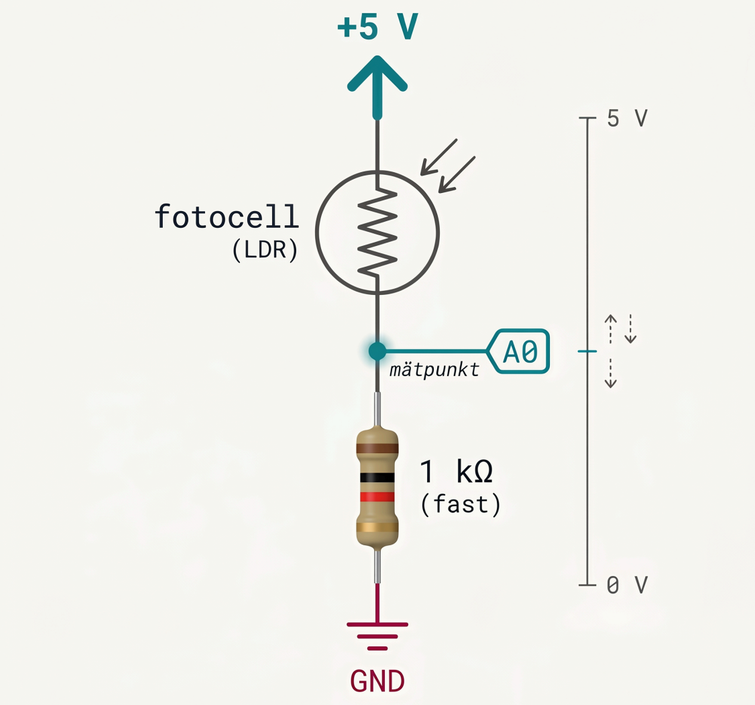
NYTT KONCEPT · FRÅN MOTSTÅND TILL MÄTVÄRDE

Spänningsdelaren.

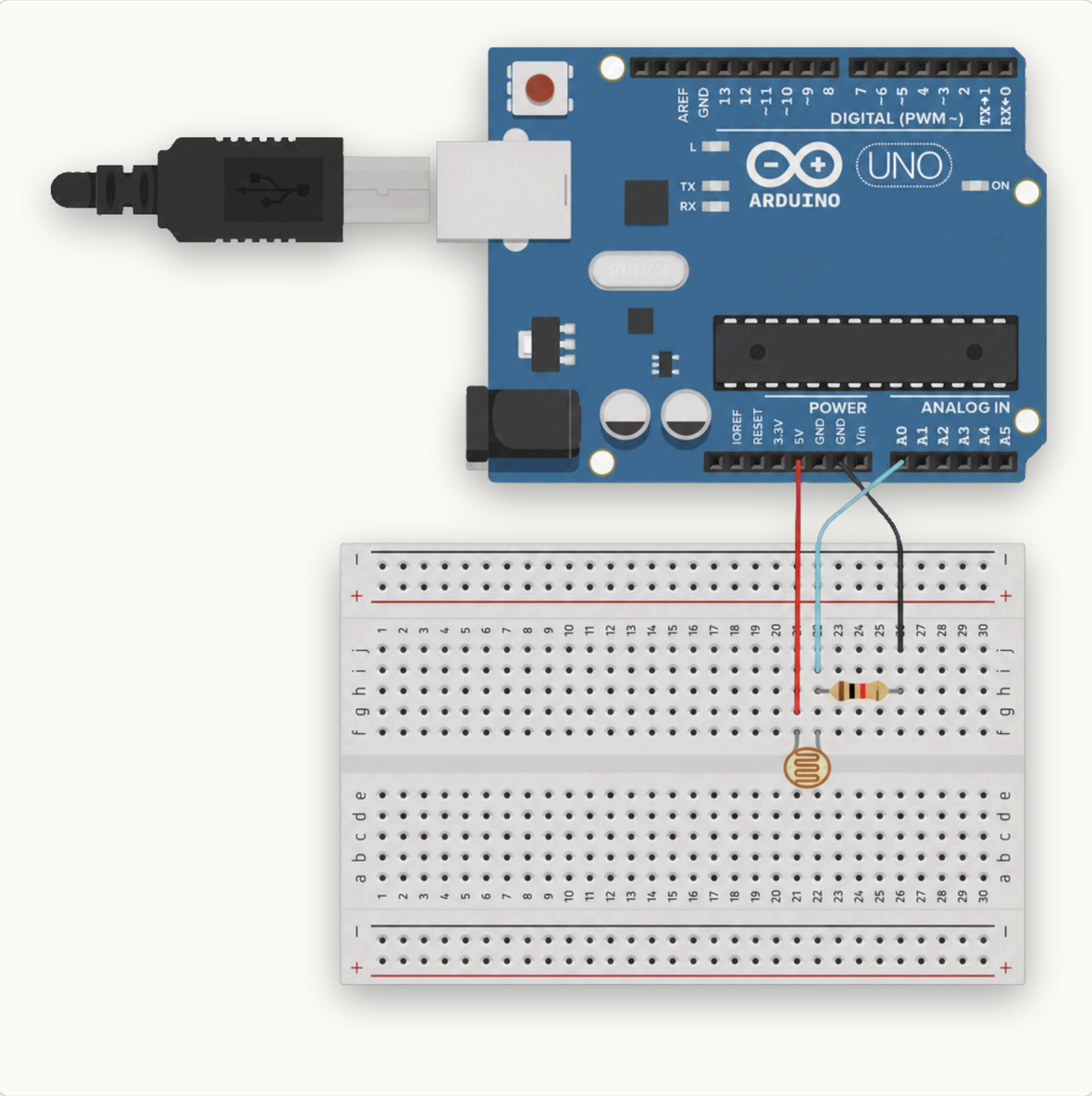
Tänk dig ett **vattenrör som höjdskala**: `+5 V` är uppe, `GND` är nere. Två motstånd i serie är två rörsegment; skarven mellan dem är en tapp — där mäter `A0`.

- Lika stora → skarven mitt på → `A0 ≈ 2,5 V`.
- Mer motstånd **ovanför** → skarven pressas ner → `A0` **sjunker**.
- Mer motstånd **nedanför** → skarven dras upp → `A0` **stiger**.

Fotocellen är det övre motståndet. Mörker → högt → `A0` sjunker. Ljus → lågt → `A0` stiger. `analogRead(A0)` ger **0–1023**.

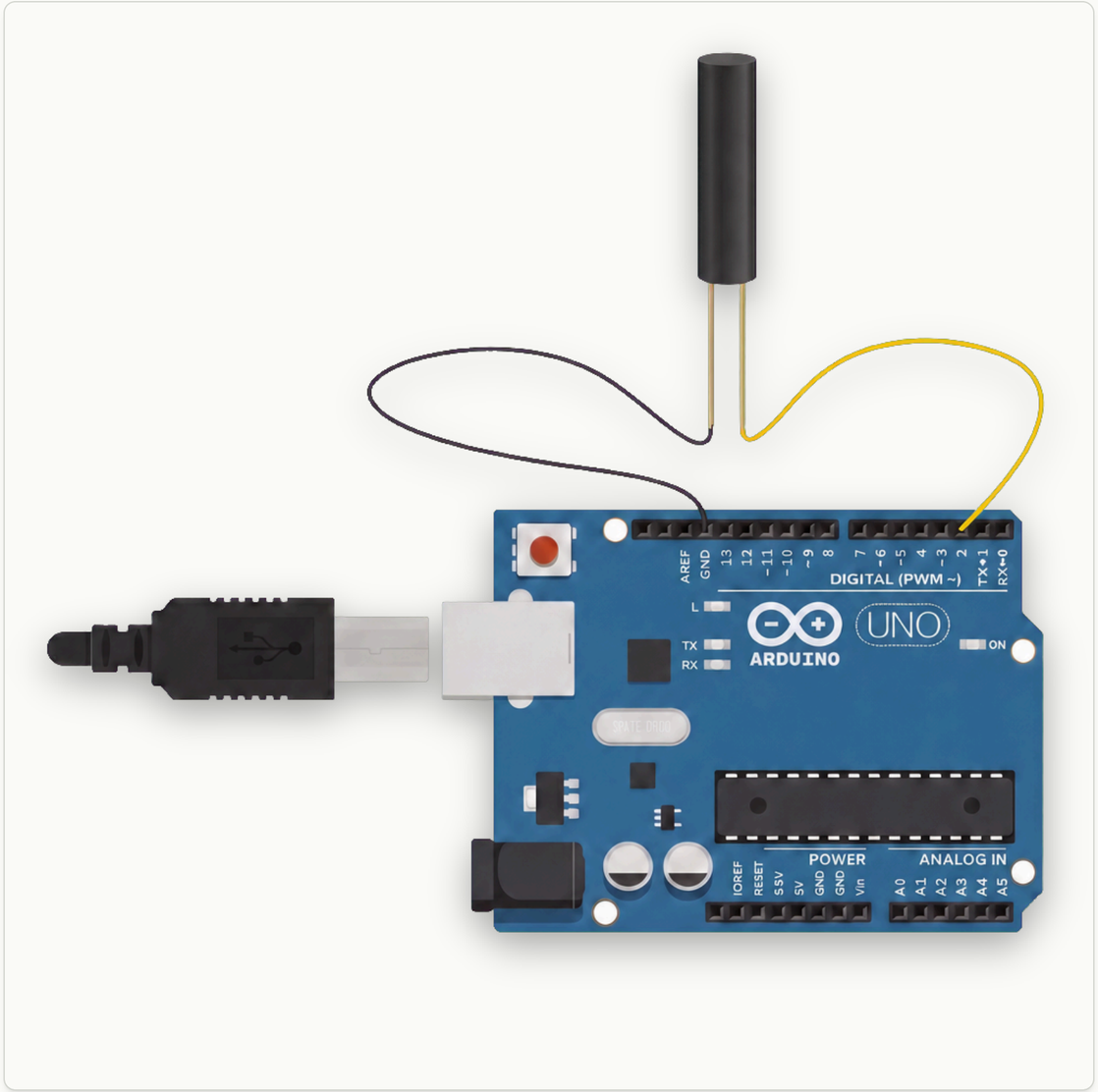


Koppla fotocellen.



Spänningsdelare: `5 V → fotocell → A0 → 1 kΩ → GND`. Mörker ger lågt värde, ljus ger högt värde.

Koppla tilt-sensorn.



Rakt in i Arduino-headerna med `långa kablar` så du kan vicka på sensorn: `ena benet → D2, andra benet → GND`. Kräver ingen breadboard. Digital sensor — öppen/sluten, precis som en knapp. Läses med `digitalRead`, inte `analogRead`.

FELSÖKNING

Serial Monitor.

Hur ser vi vilka värden Arduinon läser?

Vi låter den skriva ut dem till datorn över USB.

Öppna `förstoringsglaset` uppe till höger i Arduino IDE.

```
void setup() {
  Serial.begin(9600);
  // analoga pinnar är INPUT som default –
  // följande är onödigt men inte fel:
  // pinMode(A0, INPUT);
}

void loop() {
  int ljus = analogRead(A0);
  Serial.println(ljus);
  delay(100);
}
```

FELSÖKNING · VARIANT

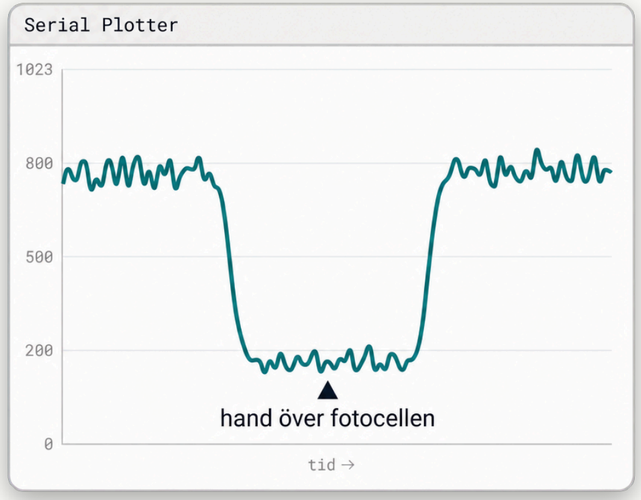
Serial Plotter.

Samma `Serial.println(…)`-data — men ritad som **rörlig kurva** istället för rader av siffror.

Håll handen över fotocellen → **linjen dippar**. Vrid potentiometern → **linjen sveper**. Signalen blir synlig.

> ÖPPNA
>
> `Verktyg → Serial Plotter`

Plotter = se signalen. **Monitor** = läsa exakta tal + statusmeddelanden (`"LARM!"`).



FRI ÖVNING

Hitta din tröskel.

Printa **både** ljus och tilt till Serial Monitor. Jämför siffrorna mot omgivningen.

- Täck fotocellen — vilket värde?
- Lys på den — vilket värde?
- Luta tilt-sensorn — ser du `HIGH → LOW`?

FRI ÖVNING · KODEN

Koden att köra.

```
const int ldrPin  = A0;
const int tiltPin = 2;

void setup() {
  Serial.begin(9600);
  pinMode(tiltPin, INPUT_PULLUP);   // tilten kopplas som knappen
}

void loop() {
  int ljus = analogRead(ldrPin);
  int tilt = digitalRead(tiltPin);
  Serial.print("ljus=");
  Serial.print(ljus);
  Serial.print(" tilt=");
  Serial.println(tilt);
  delay(200);
}
```

EFTER TRÄFF 4

Arduinon känner världen.

Ni kan läsa ljus med en fotocell, lutning med
en tilt-sensor, och titta in i Arduinons minne
via Serial Monitor.

Nästa vecka: hackathon. Ni sätter ihop allt till ett fungerande tjuvlarm. Kom hungriga.

TRÄFF 5 AV 5

Integration

Modul 5: **Hackathon — bygg larmet**

HACKATHON

Ingenjörsuppgiften.

Ingen manual att kopiera. Ni har alla delar. Sätt ihop dem.

INPUT

Knapp · LDR · Tilt

LOGIK

Arduino & kod

OUTPUT

RGB-LED · Buzzer

`Regler:` Knappen togglar larm-läge. Larm PÅ + tilt rörd = tjut + rött blink. Larm AV + mörkt = stämningsljus.

STARTMALL · PINS

Pin-tilldelning

Samma pinnar som ni använt i varje lektion.

VARIABEL	PIN	MODUL
knappPin	D9	Modul 3
tiltPin	D2	Modul 4
ldrPin	A0	Modul 4
buzzerPin	D12	Modul 3
ledR / G / B	6 / 5 / 3	Modul 2

```
bool larmPaslaget = false;
const int morkTroskel = 300;  // startvärde – kalibrera själv
```

→ Inkrementell uppbyggnad i tre steg: **Modul 5** · komplett referens-sketch (Bilaga D) delas ut efter hackathonen.

EFTER KURSEN

Vart härifrån?

Ni byggde ett fungerande tjuvlarm. Det var målet. Ni klarade det.

TinkerCAD Circuits `tinkercad.com/circuits`

Gratis online-simulator för Arduino. Bygg kretsen i browsern, skriv samma kod som i IDE:n, tryck play — **hela kretsen simuleras**, inklusive LED:ar, sensorer och Serial Monitor. Perfekt för att prova idéer utan att ha kittet framför sig.

Online-projekt

Hackster.io · Instructables · Make: — recept från väderstationer till MIDI-instrument. Kopiera något enkelt, förstå vad du kopierade.

Ert eget hem

Vad skulle ni vilja automatisera? Lampa i hallen. Brevlåde-notis. Kaffebryggar-timer. Ett litet verkligt problem slår varje tutorial.

Fullständig lista i kompendiets Modul 5-avslutning.

EFTER FEM TRÄFFAR

Tack.

Ni har byggt ett system som läser av omvärlden och reagerar på den.
Grunden i inbyggda system.

FRO Ånge · 2026