MODUL 4

Analog input

Sensorer, spänningsdelare, Serial Monitor — Arduinons fönster mot världen.

Vad du lärde dig idag

Fjärde träffen öppnade upp en ny dimension. Modul 3 lärde Arduinon säga ”tryckt / inte tryckt” — en av två möjliga svar. Modul 4 lär den säga ”500 av 1023” — **graderade** svar.

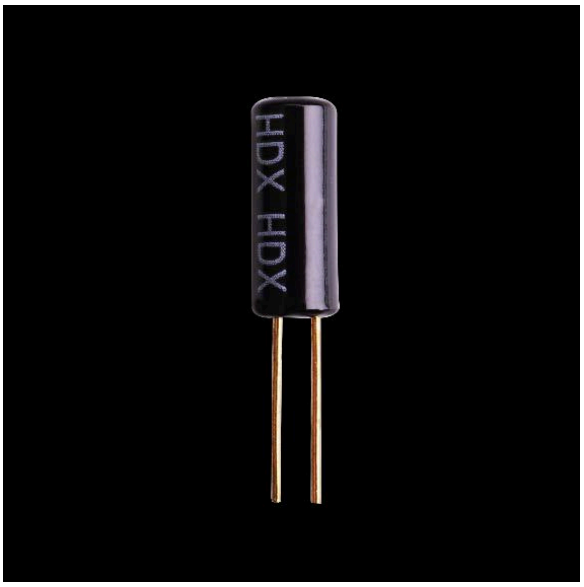
Kvällens nyheter:

- `analogRead(pin)` mäter spänningen på en analog pinne (A0–A5 på Uno) och returnerar ett heltal **0–1023**. 0 = pinnen på 0 V, 1023 = pinnen på 5 V, däremellan linjärt.
- `Serial.begin(9600)` och `Serial.println(värde)` låter Arduinon skicka text tillbaka till datorn över USB. Du ser det i **Serial Monitor** — förstoringsglas-ikonen uppe till höger i IDE:n.
- **Spänningsdelaren** är tekniken som gör det möjligt att läsa av ett motstånd med en pinne som bara kan mäta spänning. Två resistorer i serie, mellanpunkten går till A0.
- **Fotocellen** (LDR) är en resistor som ändrar värde med ljus: ca 50 kΩ i mörker, ca 500 Ω i starkt ljus. I serie med en fast 1 kΩ-resistor bildar den en spänningsdelare du kan läsa av.
- **Tilt-sensorn** (tekniskt en **ball-tilt switch** — en kula i en hylsa, inte en accelerometer) är **digital**, trots att den ofta kallas ”sensor”. Läses med `digitalRead`, inte `analogRead`. En knapp som gravitationen trycker.

Ni byggde spänningsdelaren med fotocellen, öppnade Serial Monitor tillsammans för första gången i klassrummet, och såg siffrorna ändra sig i realtid när ni höll handen över cellen. Det var Arduinons första riktiga mätinstrument.

Idag fullbordades er **sense-katalog**: en knapp (digital), en tilt-switch (digital), en fotocell (analog), plus Serial-kanalen tillbaka till datorn. Modul 5 är att sätta ihop **sense** och **act** till en riktig reaktiv maskin.

Fotocell (LDR) — en liten rund skiva med gult "ormögon-mönster" på ovansidan. Motståndet sjunker med ökande ljus.



Tilt-sensor — cylindrisk kapsel med en lös metallkula inuti. Lutad → kulan rullar mot två stift → kortslutning.

Repetition av de viktigaste begreppen

ANALOG VS DIGITAL

En **digital** ingång har två tillstånd: HIGH eller LOW. En knapp är digital. En tilt-sensor är också digital — även om vi i dagligt tal kallar den ”sensor”.

En **analog** ingång har en graderad skala. Spänningen på pinnen kan vara 0 V, 2,4 V, 4,7 V eller vad som helst däremellan. `analogRead` omvandlar den spänningen till ett tal 0–1023.

> **VARFÖR JUST 1024 STEG?**
> Arduinons analog-till-digital-omvandlare (ADC) har **10 bitars upplösning**. Det betyder $2^{10} = 1024$ möjliga utvärden, från 0 till 1023. Upplösningen blir $5\ \text{V}\ / 1024 \approx 4,88\ \text{mV}$ per steg — finkornigt nog för de flesta kursens uppgifter.

Bara pinnarna **A0–A5** på Uno har ADC-hårdvara. `analogRead` på en digital pinne returnerar skräpvärden.

SPÄNNINGSDELAREN

Bryggan: ett motstånd → två motstånd

Tänk dig först ett **enda låååångt motstånd**. Det inre materialet — i kursens resistorer en tunn kolfilm runt en keramisk kärna — är jämnt fördelat. Att skicka ström genom det är som att låta vattnet kämpa sig genom en lång trång kanal: trycket sjunker **linjärt** längs vägen. `+5 V` i ena änden, `0 V` i andra. Vid mitten är spänningen exakt `2,5 V`. Vid en fjärdedel: `3,75 V`. Vid tre fjärdedelar: `1,25 V`.

Två motstånd i serie är samma sak — bara delat på en specifik punkt. Skarven mellan dem är en mätpunkt, och det är där `A0` läser av. Värdet du får på `A0` är helt enkelt spänningen vid den brytpunkt du valde.

En **vridpotentiometer** är denna idé i hårdvara: tre ben (yttre två = ändarna av motståndsmaterialet, mittben = wipern), och ratten flyttar wipern längs det inre motståndsspåret. Kopplar du yttre benen till `+5 V` och `GND` och mittbenet till `A0`, får du ett `analogRead`-värde som sveper jämnt 0–1023 när du vrider — exakt det fenomen vi nu utnyttjar med fotocellen, fast där är det ljuset som flyttar mätpunkten.

Vatten-metaforen

Tänk dig ett genomskinligt **vattenrör som höjdskala**: `+5 V` är uppe vid kranen, `GND` är nere vid utloppet. Två motstånd i serie är två rörsegment i rad, och skarven mellan dem är en **tapp** på skalan — det är där `A0` sitter och mäter.

Regeln är enkel när man ser det så här:

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

Byt nu ut det **övre** motståndet mot en fotocell. Fotocellens motstånd ändras med ljus — högt i mörker, lågt i ljus. Det flyttar skarven upp och ner:

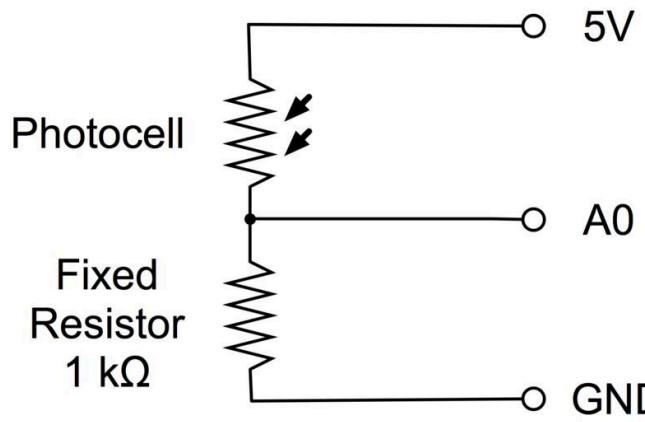
- **Mörker** → högt motstånd ovanför → skarven pressas ner → `A0` sjunker.
- **Ljus** → lågt motstånd ovanför → skarven dras upp → `A0` stiger.

`A0` läses med `analogRead(A0)` och ger ett tal **0–1023** — Arduinons 10-bits linjal över området 0–5 V.

Formellt: om R_{foto} är fotocellens motstånd och R_{under} är den fasta 1 kΩ-resistorn, så är mellanspänningen

$$V_{\text{mid}} = \frac{R_{\text{under}}}{R_{\text{foto}} + R_{\text{under}}} \cdot 5\ \text{V}$$

Fullständig härledning av formeln finns i Bilaga F.



Schematisk ritning av fotocellens spänningsdelare. +5 V → fotocell → A0 → 1 kΩ → GND. A0 mäter mellanpunktens spänning.

ORDNINGEN SPELAR ROLL

I kursens koppling sitter fotocellen **överst** (mot +5 V) och 1 kΩ **nederst** (mot GND). Detta ger: mörker = LÅG siffra, ljus = HÖG siffra. Byter du plats på dem **inverteras** skalan. Ingen fysik skadas, men logiken i koden måste följa med. Vi kör fotocellen överst genomgående.

TYPISKA A0-VÄRDEN MED 1 KΩ

Beror på ditt rum, ditt exemplar, och mängden ljus som faller in. Men som riktvärden:

Hand över cellen (nästan mörkt)	A0 ≈ **20–100**
Rumsljus, taklampa på	A0 ≈ **150–400**
Lampa nära, ljust arbetsbord	A0 ≈ **500–700**
Ficklampa direkt mot cellen	A0 ≈ **700–900**

Kalibrera själv: öppna Serial Monitor, lek med ljuset, och notera vilka värden ni ser. Den tröskel ni väljer för ”det är mörkt nu” ska ligga någonstans mellan era rumsljus- och mörker-värden.

VARFÖR JUST 1 KΩ?

Fotocellen går från 50 kΩ (mörker) till 500 Ω (ljust). Med **1 kΩ** som motvikt hamnar mätvärdet mitt i A0:s 0–1023-skala för **rumsljus** — det är där ni jobbar. 10 kΩ skulle pressa det mesta mot toppen av skalan, 220 Ω skulle pressa allt mot botten. 1 kΩ är storleksordningen som ger bäst läsbarhet för ljusnivåerna i ett klassrum.

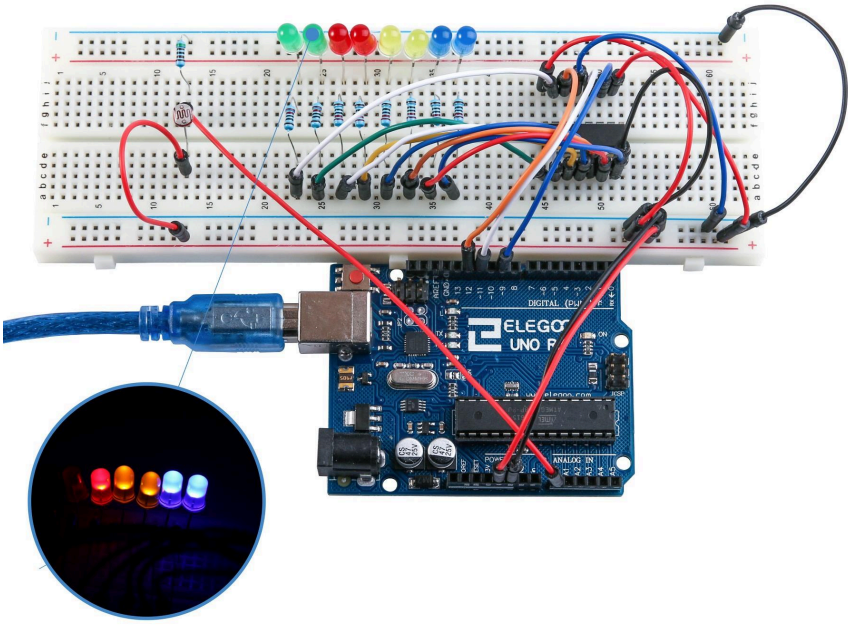
SKALA OM VÄRDEN MED `MAP()`

Ibland vill ni använda A0:s 0–1023 för att styra något som tar 0–255 (PWM) eller 0–100 (procent). Arduino har en inbyggd funktion för just detta — `map()`:

```
int pwmVarde = map(ljus, 0, 1023, 0, 255);
```

Läs det som: ”Ta `ljus` som ligger i intervallet 0–1023, skala om det linjärt till 0–255, och spara i `pwmVarde`.” Fungerar lika bra med inverterade intervall — `map(ljus, 0, 1023, 255, 0)` ger 0 vid 1023 och 255 vid 0 (användbart när mörkare ska ge starkare ljusstyrka).

Signaturen: `map(värde, frånMin, frånMax, tillMin, tillMax)`. Ni kommer se den igen i hemma-övning 2 och senare i hackathonen.



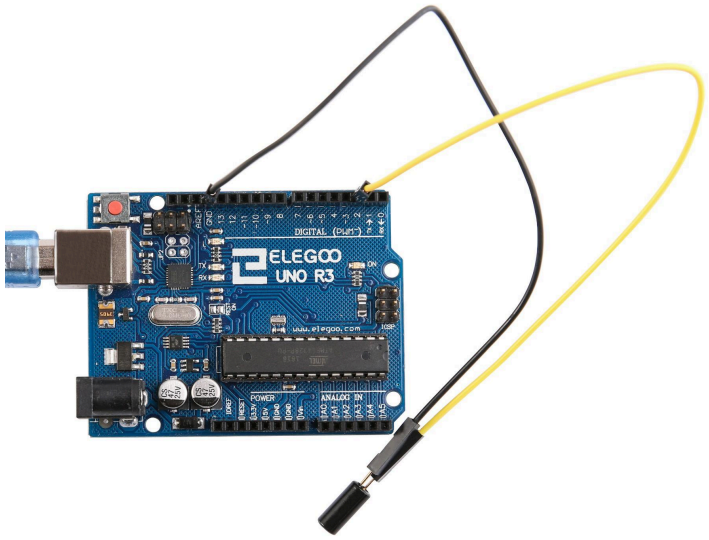
Fotocell-kretsen byggd på breadboard med RGB-LED:er som lyser i bakgrunden. Den här bilden är från en hackathon-variant där fotocellen styr LED:erna.

TILT-SENSORN

Tilt-sensorn är en liten metallcylinder med en lös kula inuti — tekniskt en **ball-tilt switch**, inte en accelerometer. I upprätt läge rör kulan bara ett av benen — kretsen är bruten. Lutas sensorn rullar kulan åt sidan och kortsluter båda benen — kretsen sluts.

Elektriskt är det alltså **identiskt med en knapp**. Kopplas som knappen: ena benet till `D2` (i kursen), andra benet till GND, och pinnen sätts upp som `INPUT_PULLUP`. Läses med `digitalRead`. (Du kan använda en annan digital pinne, men då måste du ändra pin-variabeln i koden.)

Tilten studsar fysiskt — kulan bouncar inuti sin hylsa i en tiondels sekund efter en rörelse. En `delay(50);` efter läsningen räcker som debounce i kursens sammanhang.



Tilt-sensorn kopplad direkt i Arduino-headers med F-M DuPont-kablar — ingen breadboard. Ena benet till D2, andra till GND.

SERIAL MONITOR

Första gången på kursen kan Arduinon kommunicera **tillbaka** till er. Fram till nu har ni varit envägs-kommunicerande: ni skriver kod, laddar upp, kollar om LED:en blinkar. Nu kan Arduinon skriva text till er.

Syntaxen är tre saker:

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

void loop() {
  int ljus = analogRead(A0);
  Serial.println(ljus);   // skriv värdet + ny rad
  delay(100);             // inte för snabbt, annars drunknar du
}
```

Öppna Serial Monitor genom att klicka på **förstoringsglaset** uppe till höger i Arduino IDE. En ny ruta dyker upp där siffror rullar förbi. Du kan ändra raden där det står ”9600 baud” till samma värde som du skrev i `Serial.begin()` — de måste matcha.

`Serial.print()` skriver utan radbrytning. `Serial.println()` skriver med radbrytning. För flera värden på samma rad:

```
Serial.print("ljus=");
Serial.print(ljus);
Serial.print(" tilt=");
Serial.println(tilt);
```

SERIAL MONITOR ÄR DIN FELSÖKNINGSVANA

Serial Monitor är den viktigaste felsökningsvanan i hela kursen. När ett program inte gör vad du väntar, **printa** det du tror är fel. Printa villkorsvariabler. Printa mätvärden. Printa vilken gren av en if/else som körs. Arduinon är inte svår att felsöka — den behöver bara skriva ut vad den ser.

Bygg från minnet

Skriv från scratch en sketch som printar både ljus- och tilt-värden i Serial Monitor varje hundratals millisekund. Du behöver:

1. Tre `const int` för pin-nummer: `ldrPin = A0`, `tiltPin = 2`, plus eventuellt `buzzerPin`.
2. `Serial.begin(9600)` och `pinMode(tiltPin, INPUT_PULLUP)` i setup.
3. En loop som läser båda värdena, printar dem på en rad, pausar.

Målet: se siffrorna i realtid och använda dem för att kalibrera tröskeln ”det är mörkt”.

Hemma-övningar

ÖVNING 1 — STÄMNINGSLAMPA

Skriv en sketch som tänder den inbyggda LED:en (pin 13) **endast** när fotocellen läser under en tröskel du själv väljer. När det är mörkt, tänds lampan. När det ljusnar, släcks den. Använd din egen kalibrering från lektionen.

STRUKTUR

```
if (ljus < troskel) {
  digitalWrite(LED_BUILTIN, HIGH);
} else {
  digitalWrite(LED_BUILTIN, LOW);
}
```

Där `troskel` är ett `const int` i toppen av filen. Ändra det tills beteendet känns lagom.

ÖVNING 2 — PROPORTIONAL RGB

Använd fotocellens värde som direktstyrning av RGB-LED:ens ljusstyrka. När det är mörkt → LED:en är full röd. När det är ljust → LED:en är svag röd. Formeln: `analogWrite(ledR, map(ljus, 0, 1023, 255, 0));`

`map(värde, frånMin, frånMax, tillMin, tillMax)` är en inbyggd funktion som skalar om ett tal från ett intervall till ett annat. Här: 0–1023 → 255–0 (inverterad, så att mörker ger starkt ljus).

ÖVNING 3 — TILT-ALARM

Skriv en sketch som bara piper buzzern när tilt-sensorn läser `LOW` (lutad). När den är upprätt, tystnad. Det är ett **bekräftelsetest** — visar att tilt-sensorn fungerar och styr buzzern. Hackathonens tjuvlarm bygger sedan vidare med larm-mönstret från Modul 3 (knappens toggle av `larmPaslaget`) och låter `larmPaslaget && tiltLutad` styra buzzern.

För extra credit: lägg till att buzzern piper i ett mönster istället för kontinuerligt — till exempel 200 ms på, 200 ms av, så länge tilten är lutad.

Vanliga fel och snabblösningar

analogRead ger alltid 0	Pinnen är inte kopplad (flyter, men drar mot noll). Kontrollera att fotocellen har ström från 5 V och att 1 kΩ-resistorn sitter på plats.
analogRead ger alltid 1023	Du har kopplat +5 V direkt till A0 utan resistor-par. Kolla att spänningsdelaren är komplett.
Siffrorna darrar	Normalt — sista biten på ADC:n är alltid lite brusig. Om du vill stabilisera: ta medel av flera avläsningar innan du agerar.
Serial Monitor är tom	Glömt `Serial.begin(9600)` i setup, eller Serial Monitor står på fel baud rate. Matcha samma siffra i båda.
Serial Monitor visar konstig text	Baud rate matchar inte. Båda ska vara 9600 i kursen.
Tilt-sensorn verkar inte reagera	Kontrollera `pinMode(tiltPin, INPUT_PULLUP)`. Utan pullup är det slumpmässigt.
Programmet printar 10 000 gånger per sekund	Saknar `delay(100)` i loopen. Lägg in en paus.

Snabbreferens

`analogRead(pin)`	Läser en analog pinne (A0–A5), returnerar 0–1023.
`Serial.begin(9600)`	Startar kommunikation med datorn, 9600 baud.
`Serial.print(x)`	Skriver utan radbrytning.
`Serial.println(x)`	Skriver med radbrytning.
`map(v, lo1, hi1, lo2, hi2)`	Skalar om ett värde från ett intervall till ett annat.
Fotocell-koppling	5 V → fotocell → A0 → 1 kΩ → GND. Mörker = låg siffra.
Tilt-sensor	D2 + GND, `INPUT_PULLUP`. Lutad = LOW.
Typisk debounce	`delay(50)` efter läsningen räcker.

Inför nästa träff

Nästa vecka är **hackathon**. Inga nya begrepp — allt ni behöver kan ni redan. I stället sätter vi ihop alla fyra moduler till ett fungerande tjuvlarm: knappen togglar larmläget, tilt-sensorn triggar tjut + rött blink, fotocellen styr stämningsljus i mörker.

Glöm inte dator eller kitet hemma! Hackathon brukar bli längre än man tror.