MODUL 2

PWM & RGB

analogWrite, färgblandning, och en pixel som ni programmerar själva.

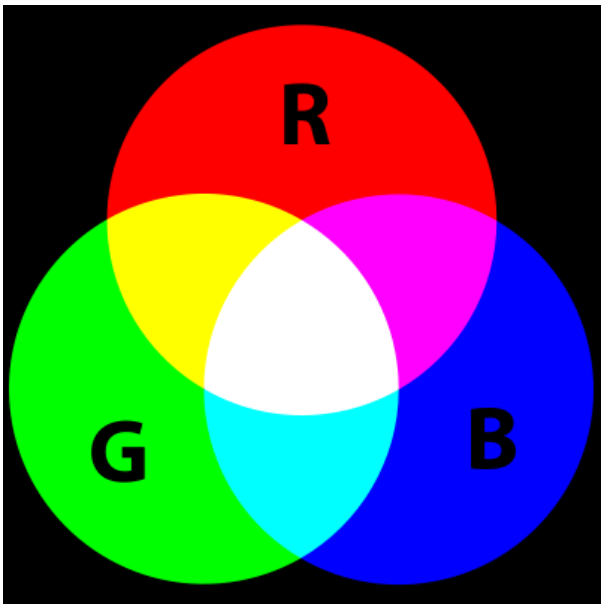
Vad du lärde dig idag

Den andra träffen handlade om att ge Arduinon ett nytt verb. Fram till nu kunde den bara säga ”PÅ” och ”AV” — `digitalWrite(pin, HIGH)` eller `digitalWrite(pin, LOW)`. I dag fick den säga ”någonstans emellan”.

Kvällens nyheter i punktform:

- `analogWrite(pin, värde)` tar ett tal från **0** till **255**. 0 = helt släckt. 255 = helt tänt. 128 = ungefär halvstyrka.
- **Arduinon kan inte göra halvstyrka på riktigt**. Den fuskar genom att blinka pinnen väldigt snabbt — flera hundra gånger i sekunden. Ögat hinner inte med och upplever blandningen som en mellannivå. Tekniken heter **PWM**, Pulse Width Modulation.
- **Bara vissa pinnar stödjer PWM**. På Arduino Uno är det pinnarna markerade med `~` på kretskortet: **3, 5, 6, 9, 10, 11**.
- **RGB-LED:en blandar färg ur tre kanaler**: röd, grön, blå. Precis som en pixel på skärmen ni sitter framför. Alla färger ni ser är kombinationer av dessa tre — det finns ingen egen gul lysdiod inblandad.
- **Kittets RGB-LED är en common cathode**. Det betyder att det längsta benet — katoden — går till GND, och de tre andra till PWM-pinnar via varsin 220 Ω-resistor.

Ni kopplade RGB-LED:en, upptäckte att pin-ordningen är rött/katod/grönt/blå (inte intuitivt — katoden är **andra** benet, inte i mitten), och experimenterade med att blanda fram lila, gammelrosa, cyan, skolgul.



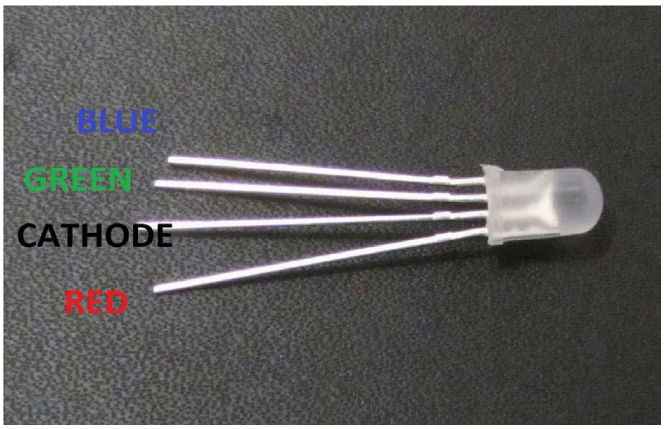
Additiv färgblandning — tre färgkanaler (rött, grönt, blått) som överlappar ger gult, cyan, magenta och (alla tre) vit. Samma princip som för skärmens pixlar.

> **TÄNK SKÄRM, INTE FÄRGBURK**
> RGB-LED:ens blandning är **additiv** — du lägger till ljus och närmar dig vitt. Målarens färglära är **subtraktiv** — du lägger till pigment och närmar dig svart. Därför: röd + grön = **gul** på er LED, men **brun-mudd** i akvarell. Den som bygger intuition åt fel håll här kommer att fastna senare i kursen.

Repetition av de viktigaste begreppen

RGB-LED:ENS PINOUT

Ditt första möte med en komponent som INTE tål att kopplas baklänges utan att krångla. Håll LED:en med den platta sidan mot dig:



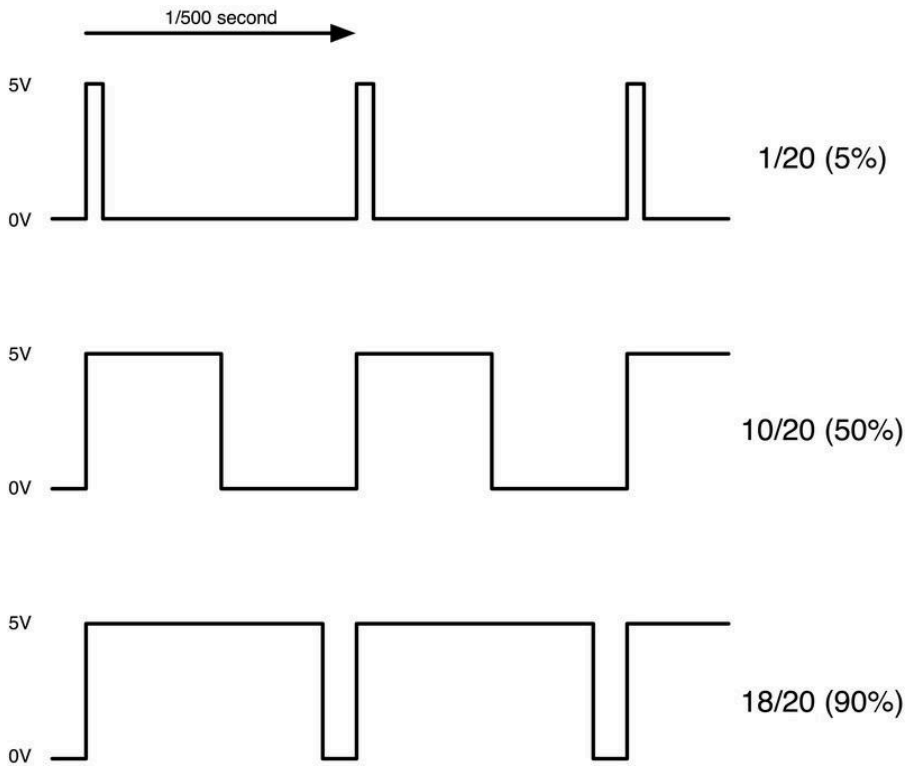
*RGB-LED:ens fyra ben, ordning från **platta sidan**: **röd · katod · grön · blå**. Katoden är det **andra** benet från platta sidan och längst av de fyra.*

Ben 1 (från platta sidan)	Röd anod — via 220 Ω → **D6**
Ben 2 — längst	**Gemensam katod** — direkt till **GND**
Ben 3	Grön anod — via 220 Ω → **D5**
Ben 4	Blå anod — via 220 Ω → **D3**

> **INTE COMMON ANODE**
> Det finns två sorters RGB-LED:er: **common cathode** (katoden är gemensam) och **common anode** (anoden är gemensam). Kittets variant är common cathode. Kopplar du som common anode lyser den inte. Värt att veta om du köper en LED lös i elektronikbutiken — fråga alltid vilken typ.

PWM OCH DUTY CYCLE

PWM fungerar så här: pinnen blinkar mellan HIGH och LOW flera hundra gånger i sekunden. Andelen tid den är HIGH kallas **duty cycle**.



PWM vid tre olika duty cycles: 5 %, 50 % och 90 %. Bilden visar en generaliserad 500 Hz. Verklig frekvens på Uno är 490 Hz (de flesta pinnar) eller 980 Hz (pin 5 och 6). Ögat hinner inte med och upplever medeleffekten.

- `analogWrite(pin, 0)` → pinnen är alltid LOW → duty cycle 0 % → släckt.
- `analogWrite(pin, 64)` → HIGH ungefär 25 % av tiden → svag lyster.
- `analogWrite(pin, 128)` → HIGH 50 % av tiden → halvstyrka.
- `analogWrite(pin, 255)` → pinnen är alltid HIGH → duty cycle 100 % → full.

På Arduino Uno växlar det mellan HIGH och LOW flera hundra gånger i sekunden. Pin 5 och pin 6 kör på ca **980 Hz**, de övriga PWM-pinnarna på ca **490 Hz**. För ögat är båda omöjliga att urskilja. För vissa känsliga sensorer eller kameror kan PWM-flimmer däremot synas — men det påverkar inte den här kursen.

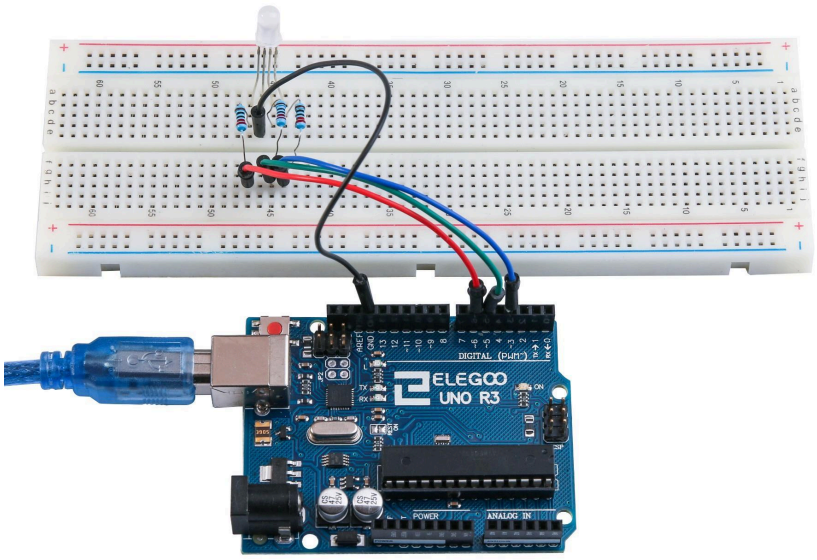
> **DÄRFÖR `~`**
> Pinnarna med tilde-tecken på Arduinon (3, 5, 6, 9, 10, 11) har hårdvarustöd för PWM. Om du av misstag använder `analogWrite` på pin 7 händer inget synligt fel, men LED:en kommer antingen vara helt av eller helt på — ingen mellannivå.

FÄRGBLANDNING I PRAKTIKEN

Tänk så här: varje kanal (röd, grön, blå) är en tonskala från 0 till 255. Du är en målare med tre tuber. Du kan klämma ut hur mycket du vill av varje, och resultatet blir en mix. Några startvärden att testa:

Röd	R 255, G 0, B 0
Grön	R 0, G 255, B 0
Blå	R 0, G 0, B 255
Vit	R 255, G 255, B 255
Gul	R 255, G 255, B 0
Cyan	R 0, G 255, B 255
Magenta	R 255, G 0, B 255
Lila	R 180, G 0, B 220
Gammelrosa	R 255, G 140, B 180
Skolgul	R 255, G 200, B 0
Mörk cyan	R 0, G 80, B 80

Du kommer märka att ögat INTE upplever 128 som ”hälften så ljust” som 255. Det beror på att ljusuppfattning är logaritmisk — vi är mer känsliga för skillnader i mörker än i ljus. Räkna inte med matematisk exakthet, experimentera fram tills det ser rätt ut.



En RGB-LED kopplad på breadboarden via tre 220 Ω-resistorer till PWM-pinnarna D3, D5, D6 och katoden till GND. Samma koppling som ni byggde.

Bygg från minnet

Skriv en sketch från scratch som cyklar mellan tre färger var fjärde sekund: röd → grön → blå → röd igen. Du behöver:

1. Tre `const int`-variabler för pin-numren (`ledR`, `ledG`, `ledB`).
2. En `setup()` som sätter alla tre som OUTPUT.
3. En `loop()` som anropar `analogWrite` på alla tre kanalerna med rätt värden, pausar, och byter färg.

SKELETT

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

void setup() {
  pinMode(ledR, OUTPUT);
  pinMode(ledG, OUTPUT);
  pinMode(ledB, OUTPUT);
}

void loop() {
```

```
  analogWrite(ledR, 255); analogWrite(ledG,   0); analogWrite(ledB,
0);
  delay(4000);
  // ... fortsätt med grön och blå
}
```

Hemma-övningar

ÖVNING 1 — DIN EGEN FÄRG

Hitta en kombination av R, G, B-värden som matchar en färg du tycker om i verkligheten. Åk ut och titta på en solnedgång, bilens instrumentbräda, tapeten hemma. Försök återskapa. Spara värdena i en sketch med kommentarer: `// solnedgång 19:42`.

ÖVNING 2 — FYRA-FÄRGS-ROTATION

Utöka ”Bygg från minnet”-övningen till fyra färger som roterar i loop: röd → gul → grön → blå → röd. Varje färg i två sekunder. Notera: gul är R 255 + G 255, inte en egen färg.

ÖVNING 3 — LÅNGSAM ÖVERGÅNG

Skriv en sketch som långsamt **tonar** över från röd till blå under 5 sekunder, istället för att hoppa. Detta kräver en `for`-loop — en ny konstruktion vi inte tagit upp på lektionen men som är lätt att läsa:

`FOR`-LOOP FÖR ÖVERGÅNGAR

```
for (int i = 0; i <= 255; i++) {
  analogWrite(ledR, 255 - i);   // röd sjunker
  analogWrite(ledB, i);         // blå stiger
  delay(20);                    // total tid: 255 × 20 ms ≈ 5 s
}
```

`for (int i = 0; i <= 255; i++)` — tre delar separerade med semikolon:

- `int i = 0` → **start:** skapa räknaren `i` och sätt till 0.
- `i <= 255` → **villkor:** kör så länge detta är sant.
- `i++` → **steg:** efter varje varv, öka `i` med 1.

Fullständig förklaring av `for` finns i Bilaga A.

Vanliga fel och snabblösningar

RGB-LED:en lyser bara i en färg	Två av tre ben är i fel hål, eller två resistorer saknas. Kontrollera att VARJE färg-ben har sin egen 220 Ω-resistor till sin PWM-pinne.

LED:en blinkar märkligt	Du har råkat koppla den som common anode — du har dragit katoden till +5 V istället för GND. Vänd på kopplingen.
`analogWrite(ledR, 200)` gör inget	Du har använt en pinne som inte stödjer PWM. Kolla att det finns ett `~` framför pin-numret på Arduinon. Giltiga: 3, 5, 6, 9, 10, 11.
Fel färg kommer ut	Du har blandat ihop vilket ben som är röd, grön, blå. Kom ihåg: från platta sidan är ordningen röd, katod, grön, blå.
Färgerna är inverterade (0 = full, 255 = av)	Du har en **common anode**-RGB istället för common cathode. Kittets LED är common cathode, men om du köpt en lös LED i elektronikbutik kan det vara motsatsen. Koppla längsta benet till `+5 V` istället för GND — då fungerar den, men du måste också invertera logiken i koden: `analogWrite(ledR, 255 - värde)`.
LED:en blinkar inte alls, men är ansluten	Glömt `pinMode(ledR, OUTPUT)` för en eller flera kanaler. Alla tre måste upp som OUTPUT i `setup()`.
Programmet fungerade, nu gör det inget	Du har glömt att ladda upp den nya versionen efter en ändring. Klicka på pilen uppe till vänster i IDE:n.

Snabbreferens

`analogWrite(pin, v)`	Skickar ut PWM med duty cycle $v/255$. Pinnen måste vara en PWM-pinne (`~`).
`pinMode(pin, OUTPUT)`	Samma som tidigare — fungerar för alla digitala pinnar, även PWM-pinnar.
PWM-pinnar på Uno	3, 5, 6, 9, 10, 11 — markerade med `~` framför pin-numret.
Kittets RGB-LED	Common cathode. R/K/G/B från platta sidan. Varje färg-ben via 220 Ω.
Kursens pin-ordning	R → D6, G → D5, B → D3.

Inför nästa träff

Modul 1 och 2 har varit rent **act** — Arduinon har bara skickat ström ut till LED-ar. Modul 3 lägger till den andra halvan: **sense**. Arduinon får sitt första öra.

Vi introducerar `digitalRead` och `INPUT_PULLUP`, bygger en första sketch som reagerar på knapptryck, lägger till en buzzer (den med den viktiga klisterlappen), och slutligen bakar in

edge-detection — den tekniska detalj som gör det möjligt att toggla ett tillstånd på och av utan att knappen skriver över sig själv 50 gånger per sekund.

Glöm inte dator eller kitet hemma!