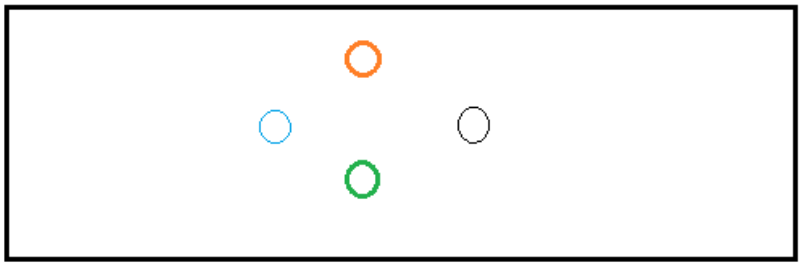
rd

Sessie 3: het gebruik van functies/procedures.

1. Functie: is een geheel van commando's die kunnen uitgevoerd worden, maar steeds afhankelijk zijn van het bovenliggend programma codes. De uitgang van een functie moet opgeslagen worden in een aparte variabelen. Een functie kan bepaalde input variabelen krijgen.
2. procedure is een geheel van commando's die onafhankelijk van boven liggende code kan gebruikt worden in het bestaande programma. De procedure kan een bepaalde uitgang (output) hebben welke dan in een bepaalde variabele kan opgeslagen worden. Een procedure is een op zichzelf bestaande programma structuur. Dit tegenstelling met een functie.
3. In de eerste sessie heb ik al gewag gemaakt van een functie (wiskunde). Daarin kon je zien dat een functie f steeds nood heeft aan bepaalde ingang variabelen. Herinner u de functie **SOM** waarbij we twee factoren (algemeen a,b) trachten bij elkaar op te tellen. De uitslag van die uitslag van de optelling stoppen we in een variabele C. in totaal krijgen we dus **SOM (a,b)→C**
4. Gans deze operatie kunnen we verkort stellen als C=A+B, waarbij + som betekend. A en B kunnen nu vervangen worden door getallen welke dan ook een getal als uitslag heeft. Vb c=12 + 6 of in dit geval 18, dan ook de uitslag van die functie.
5. Ik blijf maar het woord **variabele** uitbrengen. Voor ik verder doe met functies en procedures zullen we eerst een paar veel gebruikte *soorten* variabelen bespreken. Tijdens verdere sessies zal u er nog soorten tegen komen, laat ons nu beginnen met de meest voorkomende Integer, cijfer of samenstelling van cijfers. Deze Variable heeft een beperkte grootte. Gelegen tussen -32.768 en +32767 . Is het getal groter wordt dat opgeslagen in een long integer of kortweg een long.. Deze gaat van −2.147.483.647 tot +2.147.483.647. Later zal je wel de reden ontdekken.
6. In onze programmering van processoren zal je niet zo rap iets groter tegen komen dan een integer, dus laat ons het daar maar bij houden. Er bestaan nog andere types van variabelen maar dze zal je niet in de eerste sessie-reeks tegen komen en in geval van zal ik daar verder over uitweiden.
7. Om terug te komen op onze functie. Ik heb nog niet aan de voordelen van een functie toe gekomen. Deze zal ons programmeren van code een stuk makkelijker maken. Immers men kan een functie/procedure blijven herhalen in het zelfde programma. Welke ons dat veel nutteloos werk kan besparen.
8. Na deze brok theorie ga ik over baar de praktijk. Daarvoor kom ik terug op de gevraagde
9. oefening. Laat ons die code eens nader bekijken. De opstelling was als volgt;



10. De bijgaande code:

```
const time = 1000;
int rood = 3;
int wit = 4:
int blauw = 5;
int groen = 6;

void setup() {
  pinMode(rood,OUTPUT):
  pinMode(wit,OUTPUT):
  pinMode(blauw,OUTPUT):
  pinMode(groen,OUTPUT):
}

void loop() {
 digitalWrite(rood,HIGH);
 delay(time);
 digitalWrite(rood,LOW);
 digitalWrite(wit,HIGH);
 delay(time);
 digitalWrite(wit,LOW);
 digitalWrite(blauw,HIGH);
 delay(time);
 digitalWrite(blau,LOW);
 digitalWrite(groen,HIGH);
 delay(time);
 digitalWrite(groen,LOW);
}
```

Merk op dat ik een speciale variabele gebruik, namelijk ‘const’. Het nut van deze is dat bij het veranderen van de vertragingstijd naar bv 500 ms hoeft ik enkel de constante bovenaan te veranderen en automatisch wordt dat toegepast op de code, overal waar time staat.

Vergewis u dat deze natuurlijk bij de publieke variabelen staat! Dit wordt aangeduid met *const* .

Stel dat we van gans de code welke in loop staat een functie/procedure maken. Tevens maken we een tweed functie/procedure die de richting van oplichten veranderd. Dit gieten we dan ook in een functie/procedure. Daaruit volgt dat we dat op willekeurig tijdstip in het programma kunnen veranderen.

11. Onze neuwe code wordt dan:

```
const time = 1000;
int rood = 3;
int wit = 4:
int blauw = 5;
int groen = 6;
```

rd

```
void setup() {
  pinMode(rood,OUTPUT):
  pinMode(wit,OUTPUT):
  pinMode(blauw,OUTPUT):
  pinMode(groen,OUTPUT):
}

void loop() { rechts();
delay(time);
links();
delay(time);
}

void rechts(){
 digitalWrite(rood,HIGH);
 delay(time);
 digitalWrite(rood,LOW);
 digitalWrite(wit,HIGH);
 delay(time);
 digitalWrite(wit,LOW);
 digitalWrite(blauw,HIGH);
 delay(time);
 digitalWrite(blau,LOW);
 digitalWrite(groen,HIGH);
 delay(time);
 digitalWrite(groen,LOW);
}
void links(){
  digitalWrite(blauw,HIGH);
 delay(time);
 digitalWrite(blauw,LOW);
 digitalWrite(groen,HIGH);
 delay(time);
 digitalWrite(grorn,LOW);
 digitalWrite(wit,HIGH);
 delay(time);
 digitalWrite(wit,LOW);
 digitalWrite(rood,HIGH);
 delay(time);
 digitalWrite(rood,LOW);

}
```

12. merk op dat de code in het deel loop verkleint is. Immers we gebruiken nu de functies rechts en links (gegeven volgens de draai mogelijkheid. Prachtig hoe dat werkt. Met deze zie ook de sterkte van procedures/functies!,