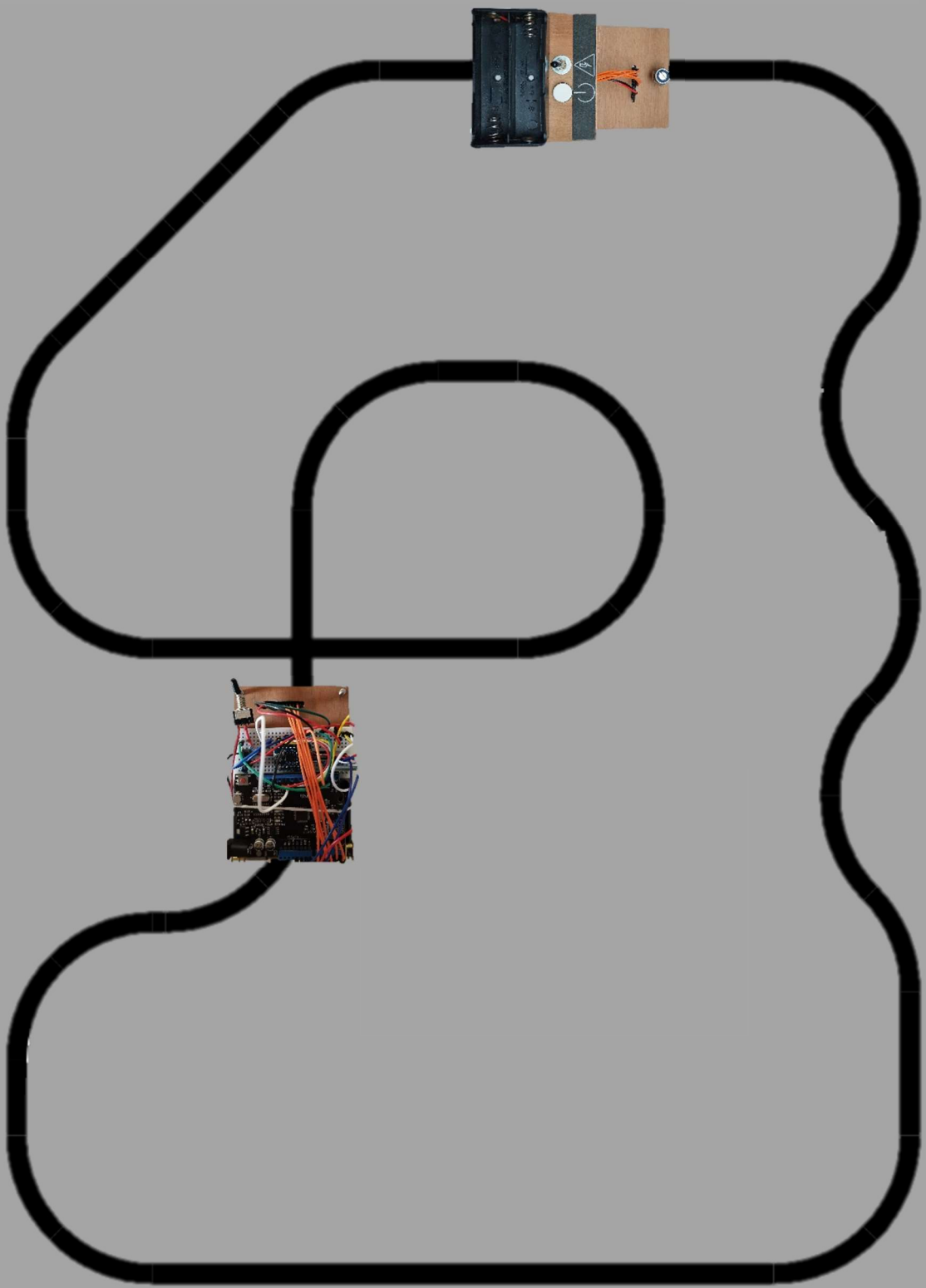
Instructable linefollower

Lancelot De Bruycker



Inhoud

Benodigdheden

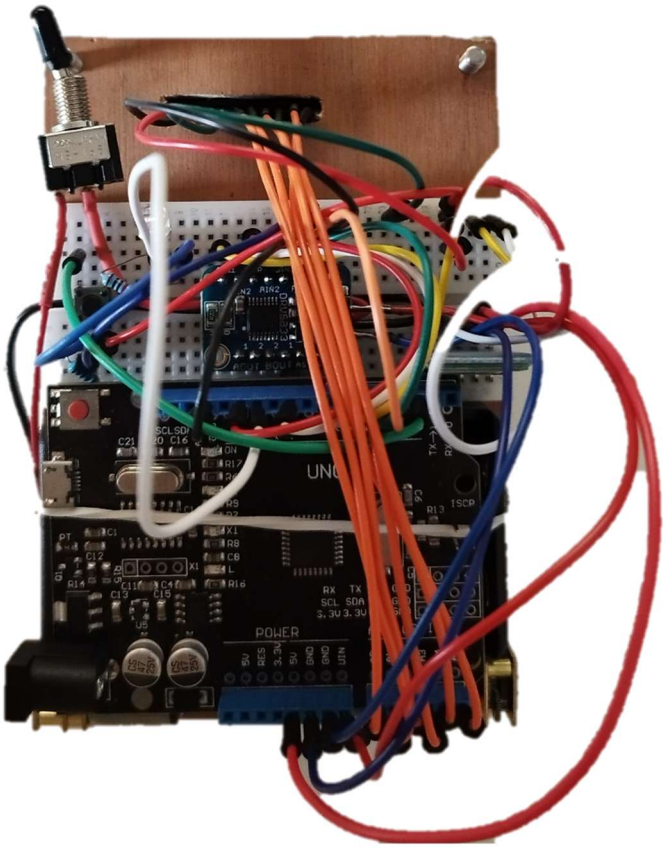
Onderdelen:

1/50 6 VDC gearbox motor
18650 Batterijhouder
Adafruit DRV8833
Arduino Leonardo
Bouten met moer
Breadboard
Drukknop
Dupont kabels
Efest 18650
Efest 18650 lader
Elastiek 60 x1,5
HC-05 bluetooth communicatie
Hotglue
Leds
Multiplex plaat
Pololu 32x7 wielen
QTR-8A sensor array
Soldeertin
Tuimelschakelaar
Weerstanden

Gereedschap:

Boormachine met boor 4mm
Breekmes
Figuurzaag
Schroevendraaiers, plat en kruis
Schuurpapier
Soldeerbout
Vijl
Hotglue pistool
Secondenlijm
Decoupeerzaag

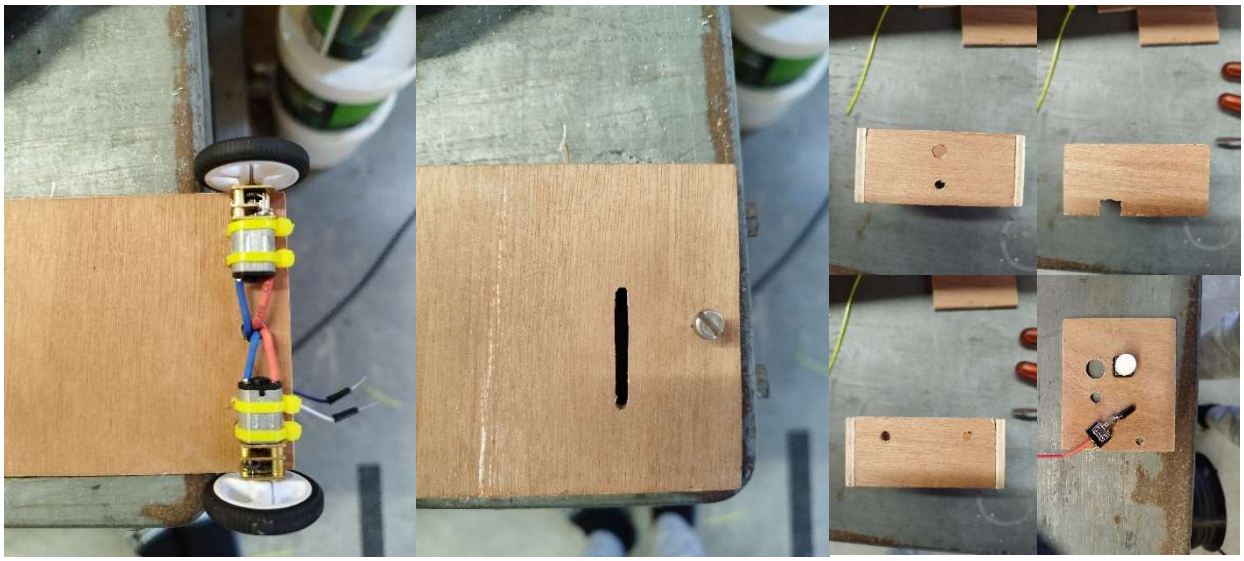
Plan B versie 1



In versie 1 is de linefollower volledig functioneel. Echter is dit niet het eindontwerp. De eerste versie is heel eenvoudig in elkaar te knutselen. We gebruiken een breadboard en dupont kabeltjes, aanpassingen maken kan dus heel makkelijk.

Dit is echter niet ideaal, tijdens het rijden kunnen er connecties loskomen waardoor er veel tijd verloren gaat aan probleemzoeken. Het oogt ook zeer slordig, daarom maken we ook een 2de versie waar we alle kabels gaan solderen en in een behuizing gaan verbergen. In deze instructable wordt niet dieper ingegaan op versie 1.

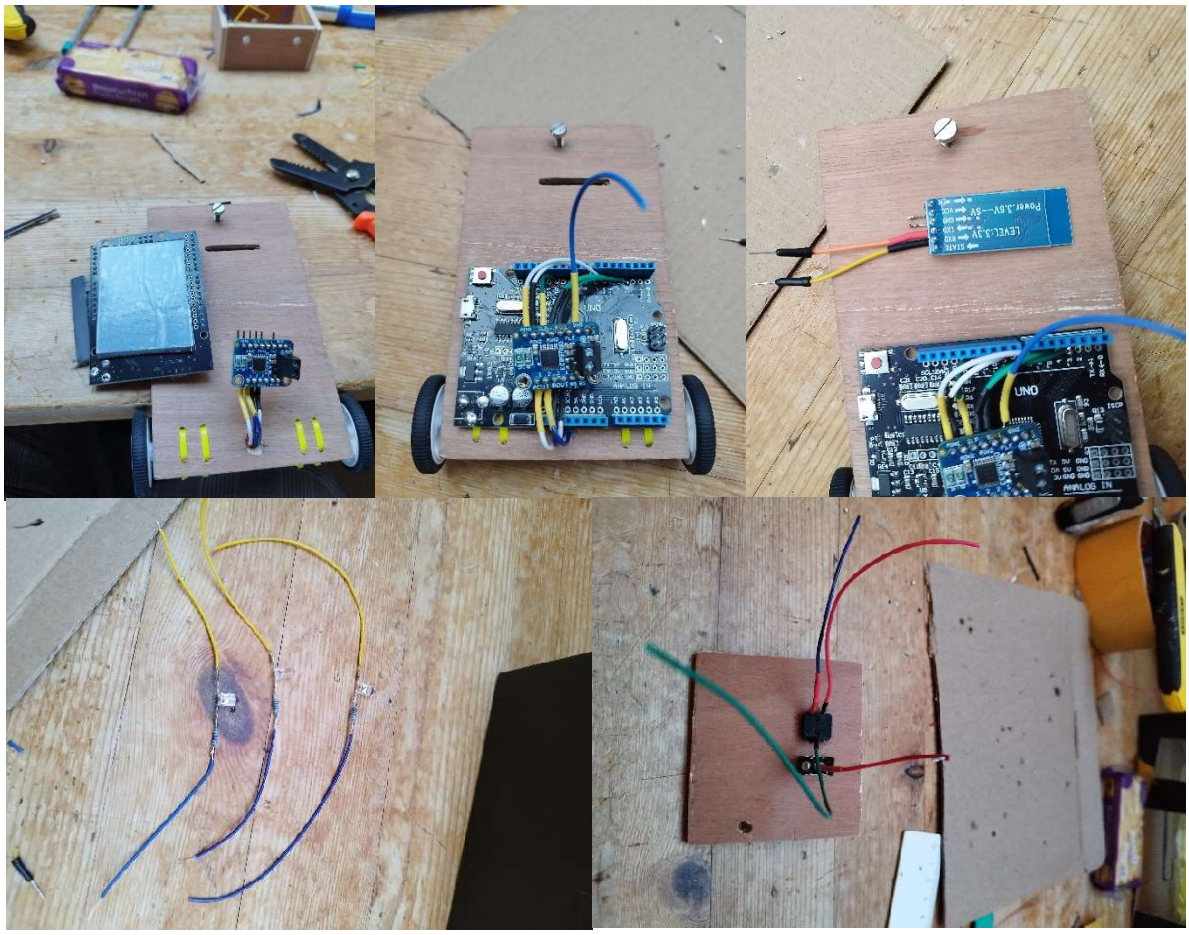
Stap 1: Chassis maken



Ga als volgt tewerk:

- Teken alle onderdelen van de linefollower af.
- Zaag de vormen uit met een decoupeerzaag en schuur alle zijkanten met schuurpapier tot zo mooi glad zijn.
- Boor diameter 6 waar de motoren moeten komen. Boor ook aan elke motor 4 gaatjes waar de spanbanden moeten komen.
- Monteer de motoren. Doe de spanbanden in de gaatjes en trek ze stevig aan.
- Boor een gat vooraan in het chassis voor de ondersteuning van de linefollower. Hier plaatsen we een bout in met een kogel aangelijmd.
- Maak met een breekmes de uitsparing voor de IR-sensor. Maak deze nauwkeurig om speling te voorkomen.
- Boor diameter 3 gaatjes waar de LED's moeten komen.
- Maak ook een uitsparing voor de seriële poort van de Arduino, zo kunnen we later nog sketches uploaden.
- Maak in de bovenkant van de linefollower 3 gaten. Eén voor de aan/uit schakelaar, één voor de drukknop en één voor de kabels van de batterijhouder.

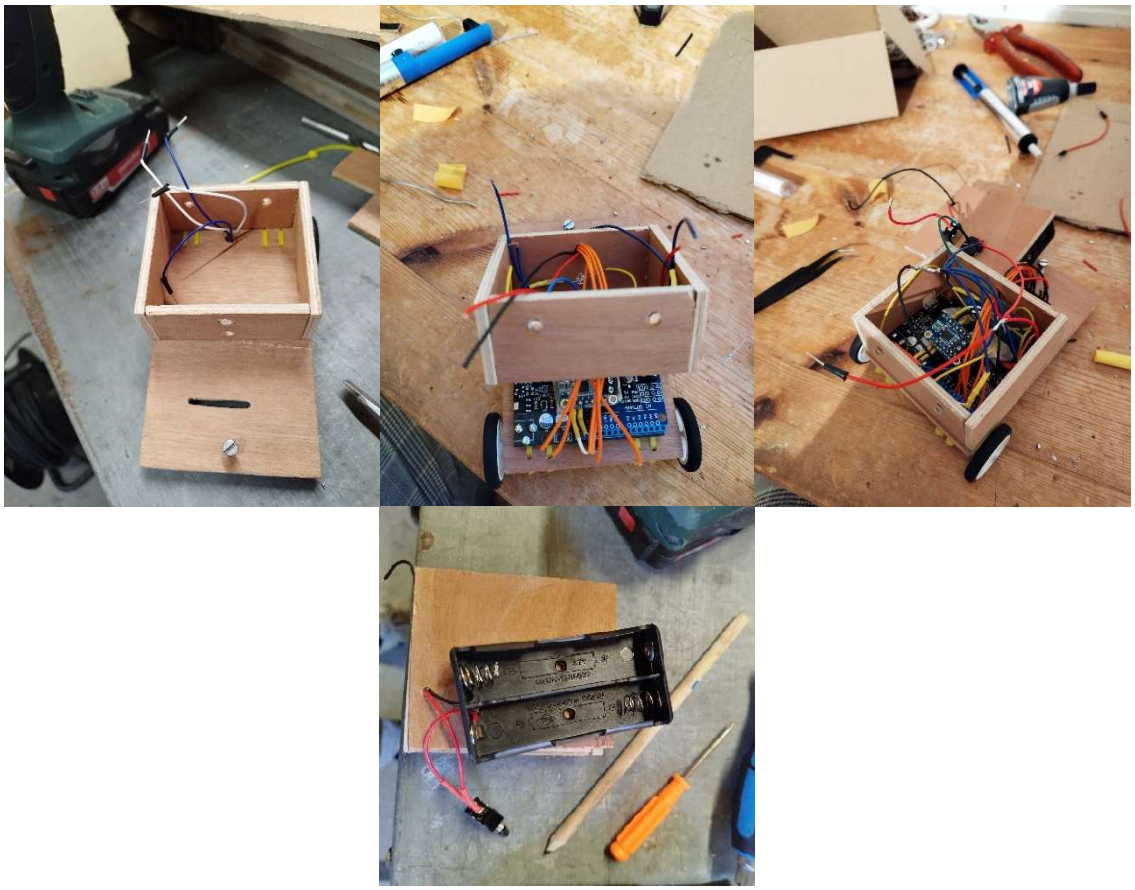
Stap 2: Soldeerwerk



Ga als volgt te werk:

- Soldeer de uitgangen van de motoren op de uitgangen van de DRV8833
- Plak de Arduino op de bodemplaat juist achter de kabels van de motoren.
- Soldeer de digitale PWM-uitgangen aan de ingangen van de DRV8833 (kijkend naar het schema). Soldeer eveneens een kabel aan de GND-aansluiting.
- Soldeer een kabel aan de TXD, RXD, GND en VCC op de HC-05 module. Gebruik voor de TXD en RXD een dupont kabel, deze dienen losgemaakt te kunnen worden bij het uploaden van een sketch.
- Maak nu alle LED's klaar. Soldeer aan de kathode een kabel en een weerstand van 220 Ohm. Aan de anode soldeert u enkel een kabel.
- Soldeer aan de drukknop 3 kabels. Tussen de kabel die naar GND gaat moet een weerstand van 1kOhm. De kabel aan de schakelaar komt van de positieve klem van de batterij.
- Soldeer als laatst de uitgangen van de IR-array aan de analoge ingangen van de microcontroller.

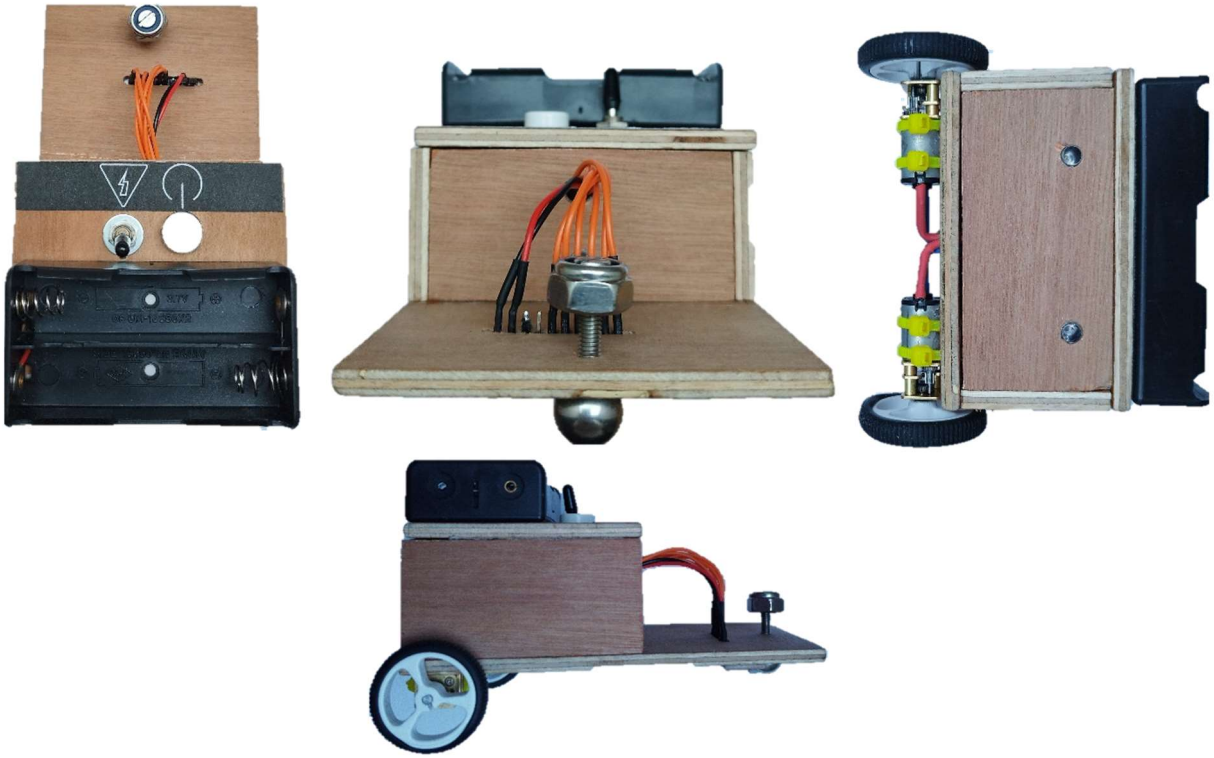
Stap 3: Componenten op het chassis lijmen



Ga als volgt te werk:

- Lijm alle zijkanten aan elkaar.
- Lijm het frame met secondelijm aan de bodemplaat.
- Plak met dubbelzijdige tape de batterijhouder aan de bovenkant.
- Als het chassis volledig is gelijmd kunnen we de laatste zaken solderen. We nemen alle GND-kabels bij elkaar en plaatsen er een dupont kabel bij zodanig dat deze makkelijk kan verbonden worden. Dan plaatsen we er een krimpkous over en solderen we ze samen. We doen exact hetzelfde voor alle VCC-kabels die naar +5V moeten.
- Als alles gelijmd is kunnen we de bovenplaat toeplakken met twee stukjes dubbelzijdige tape.

Stap 4: Alle kabels verbinden



Ga als volgt te werk:

- Bijna alles is gesoldeerd. Enkel de kabels van de batterijhouder, de GDN, de VCC, en de RXD en TXD. Dit zijn doupont kabels en zijn dus makkelijk te verbinden.

- Verbind RXD met TX op de Arduino.
- Verbind TXD met RX op de Arduino.
- Verbind de plus klem van de batterij met de plusklem van de DRV8833. Voeg een extra kabel toe in deze klem en verbind dit met de VIN van de microcontroller.
- Verbind de min klem van de batterij met de negatieve klem van de DVR8833. Voeg een extra kabel toe en verbind dit met de GND van de microcontroller.
- Verbind alle de samengenomen GND-kabels aan de GND van de Arduino.
- Verbind alle de samengenomen VCC-kabels aan de 5V van de Arduino.

Stap 5: Programmeren

We beginnen met benodigde bibliotheken toe te voegen. Hierna definiëren we de seriële poort waarover gecommuniceerd zal worden en de baudrate. De LED's en drukknop worden eveneens gedefinieerd.

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include "SerialCommand.h"
#include "EEPROMAnything.h"

#define SerialPort Serial
#define Baudrate 9600
#define BUTTON_PIN 2
#define LED_PINAL 13
#define LED_PINAR 10
#define LED_PINV 7
```

Nu voegen we de PWM-uitgangen voor de motoren toe.

```
int MotorLeftForward = 5;
int MotorLeftBackward = 3;
int MotorRightForward = 6;
int MotorRightBackward = 9;
```

We maken enkele globale variabelen aan en definiëren de analoge ingangen voor de sensoren. Als bij het aansluiten een andere volgorde van de IR-LED's gekozen is zal dit hier moeten aangepast worden.

```
volatile byte intr = LOW;
int sleep = 12;
bool start;
bool debug;
float iTerm;
float lastErr;
unsigned long previous, calculationTime;
const int sensor[] = {A5, A4, A3, A2, A1, A0};
int normalised[6];
float debugPosition;
```

Nu maken me een `struct` functie aan waar we de parameters insteken die we in het EEPROM-geheugen willen opslaan.

```
struct param_t
{
  unsigned long cycleTime;
  int black[6];
  int white[6];
  int power;
  float diff;
  float kp;
  float ki;
  float kd;
} params;
```

In de setup zetten we alles wat maar 1 maal moet uitgevoerd worden wanneer we het programma starten. Hier worden dus de commando's aangemaakt en pinnen gedefinieerd als in- of uitgang. Ook worden de parameters die voordien in het EEPROM-geheugen werden opgeslaan uitgelezen.

```
  SerialPort.begin(Baudrate);

  // Alle comando's hier:
  sCmd.addCommand("set", onSet);
  sCmd.addCommand("debug", onDebug);
  sCmd.addCommand("calibrate", onCalibrate);
  sCmd.addCommand("start", onStart);
  sCmd.addCommand("stop", onStop);
  sCmd.addCommand("lichtshow", onLights);
  sCmd.setDefaultHandler(onUnknownCommand);

  pinMode(sleep, OUTPUT);
  digitalWrite(sleep, true);
  pinMode(MotorLeftForward, OUTPUT);
  pinMode(MotorLeftBackward, OUTPUT);
  pinMode(MotorRightForward, OUTPUT);
  pinMode(MotorRightBackward, OUTPUT);
  pinMode(LED_PINAL, OUTPUT);
  pinMode(LED_PINAR, OUTPUT);
  pinMode(LED_PINV, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
  attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), interrupt, RISING);

  EEPROM_readAnything(0, params);
```

In de setup plaatsen we ook enkele seriële communicatie die zal getoond worden bij het opstarten van de robot.

```
  SerialPort.println("--- Ready for take off 🚀 ---");
  Serial.println();
  SerialPort.println("To start: ");
  SerialPort.println(" - Type start ");
  SerialPort.println(" - Press start ");
  SerialPort.println("To Stop: ");
  SerialPort.println(" - Type stop ");
  SerialPort.println(" - Press stop ");
```

In de loop programmeren we alles dat herhaaldelijk moet uitgevoerd worden. We beginnen met de sensoren uit te lezen en ze te normaliseren. Dan gaan we de waardes ook interpoleren.

```
 for (int i = 0; i < 6; i++)
  {
      normalised[i]= map(analogRead(sensor[i]), params.black[i], params.white
[i], 1, 1000);
  }
    float position = 0;
    int index = 0;
    for (int i = 0; i < 6; ++i) if (normalised [i] < normalised[index]) index
= i;
```

Heel belangrijk is volgende lijn code:

```
 if (normalised[index] > 750) start = false, intr = false;
```

Met deze lijn code kunnen we ervoor zorgen dat wanneer de robot de zwarte lijn kwijt is, ze stopt met rijden.

We kunnen nu beginnen met de regeling van de robot. Beginnend met proportioneel regelen:

```
float error = -position;
float output = error* params.kp;
```

Integrerend:

```
iTerm += params.ki * error;
iTerm = constrain(iTerm, -510, 510);
output += iTerm;
```

Differentiërend:

```
output += params.kd * (error - lastErr);
lastErr = error;
```

We begrenzen de output waarde tot waardes die fysisch mogelijk zijn.

```
output = constrain(output, -510, 510);
```

Hierna sturen we onze motoren aan. Dit hangt af van de gekozen H-brug en is volledig afhankelijk van de aansluitingen.

```
if (start || intr) if (output >= 0)
{
  powerLeft = constrain(params.power + params.diff * output, -255, 255);
  powerRight = constrain(powerLeft - output, -255, 255);
  powerLeft = powerRight + output;
}
else
{
  powerRight = constrain(params.power - params.diff * output, -255, 255);
  powerLeft = constrain(powerRight + output, -255, 255);
  powerRight = powerLeft - output;
}

analogWrite(MotorRightForward, powerLeft > 0 ? powerLeft : 0);
analogWrite(MotorRightBackward, powerLeft < 0 ? -powerLeft : 0);
analogWrite(MotorLeftForward, powerRight > 0 ? powerRight : 0);
analogWrite(MotorLeftBackward, powerRight < 0 ? -powerRight : 0);
```

Nu gaan we enkele voids aanmaken waar we commando's die we in de setup hebben aangemaakt kunnen uitlezen.

We beginnen met het commando onUnknownCommand, deze zal alle invoer die niet herkend wordt uitfilteren en antwoorden dat het commando niet is gekend.

```
void onUnknownCommand(char *command)
{
  SerialPort.print("Unknown command: \"");
  SerialPort.print(command);
  SerialPort.println("\"");
}
```

In de void OnSet wordt alles wat parameters betreft uitgelezen en eventueel aangepast. Alles wordt ook naar het EEPROM-geheugen weggeschreven.

```
  char* param = sCmd.next();
  char* value = sCmd.next();

  else if(strcmp(param, "power") == 0) params.power = atol(value);
  else if(strcmp(param, "diff") == 0) params.diff = atof(value);
  else if(strcmp(param, "kp") == 0) params.kp = atof(value);
  EEPROM_writeAnything(0, params);
```

De volgende void die we aanmaken is de onDebug. Hierin zullen alle waardes die in het EEPROM- geheugen zitten weergegeven worden in de seriële monitor wanneer het commando debug wordt ingegeven. Hier kunnen waardes naar hartenlust toegevoegd worden die mogen weergegeven worden in de seriële monitor.

```
SerialPort.print("----------Parameters----------");
SerialPort.println();

SerialPort.print("power: ");
SerialPort.println(params.power);

SerialPort.print("diff: ");
SerialPort.println(params.diff);

SerialPort.print("kp: ");
SerialPort.println(params.kp);
```

In de void onCalibrate kunnen we bij het ingeven van het commando “calibrate black/white” de sensoren kalibreren. Nadien schrijven we ook alles naar het EEPROM-geheugen.

```
if (strcmp(param, "black") == 0)
{
  SerialPort.print("Start calibrating black... ⚫ ");
  for (int i = 0; i < 6; i++) params.black[i] = analogRead(sensor[i]);
  delay(1500);
  SerialPort.println();
  SerialPort.print("Calibrated values: 📊 ");
  for (int i = 0; i < 6; i++)
  {
    SerialPort.print(params.black[i]);
    SerialPort.print(" ");
  }
  SerialPort.println();
}
```

Als laatst is er een void voorzien voor de start, stop, interrupt, en lichten commando. Deze zijn vanzelfsprekend.

```
void onStart()
{
  start = true;
  iTerm = 0;
  SerialPort.print("Started 🏳");
  SerialPort.println();
}

void onStop()
{
  start = false;
  intr = false;
  SerialPort.print("Stopped 🏁");
  SerialPort.println();
}

void interrupt()
{
  intr = !intr;
  if (intr)
  {
    iTerm = 0;
    SerialPort.print("Started 🏳");
    SerialPort.println();
  }
  else if (!intr)
  {
    start = false;
    SerialPort.print("Stopped 🏁");
    SerialPort.println();
  }
void onLights()
{

  SerialPort.print("Lichtshow 💡");
  SerialPort.println();
  digitalWrite(LED_PINV, HIGH);
  delay(600);
  digitalWrite(LED_PINV, LOW);
  digitalWrite(LED_PINAR, HIGH);
  delay(600);
. . .
```

Stap 5: Uitvoerig testen

De linefollower is klaar voor gebruik. Nu kunt u uitvoerig testen en met de parameters prutsen.

Enkele problemen die kunnen opduiken:

Probleem	Oplossing
Kan geen verbinding maken met de robot	COM-poort veranderen Andere seriële kabel Drivers geïnstalleerd Bluetooth ingeschakeld
Kan programma niet uploaden	RX en TX ontkoppelen op Arduino
Kan geen verbinding maken met bluetooth	Juiste baudrate instellen
Startknop werkt niet	Aansluitingen controleren Juiste pin definiëren
Motoren draaien traag of niet	Voedingspanning batterij te laag
Sensoren lezen verkeerde waardes uit	Sensorwaarde wit kalibreren Sensorwaarde zwart kalibreren Sensoren verkeerd aangesloten
Niets zichtbaar in seriële monitor	Juiste baudrate kiezen Both NL & CR kiezen
Ongekend probleem	Alle connecties controleren Programma opnieuw uploaden
Robot valt stil	Parameter power verhogen
Robot rijdt te traag	Parameter power verhogen
Robot stuurt niet snel genoeg bij	Parameter kp verhogen
Robot waggelt over de zwarte lijn	Parameter kp verlagen
Robot versnelt in de bochten	Parameter diff verlagen
Robot vertraagt in de bochten	Parameter diff verhogen
Robot valt stil in de bochten	Parameter diff verhogen