IoT Project

Line Following Robot

Graduaat in Internet of Things

Maxim Bozek

Academiejaar 2022-2023

Campus De Nayer

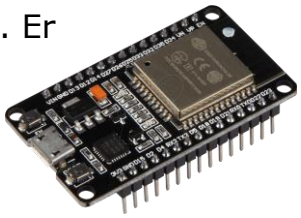
LID VAN ASSOCIATIE KU LEUVEN

THOMAS MORE

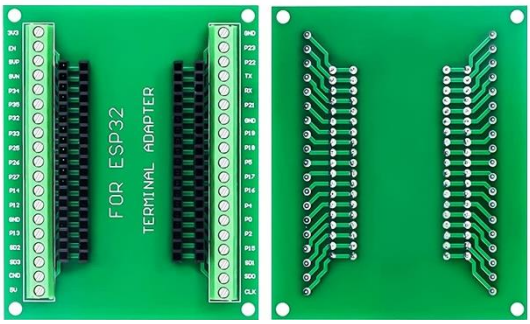
HARDWARE EN BEHUIZING

Een lijst van alle gebruikte componenten en waar ze voor dienen:

ESP32 (WROOM 32)



Microcontroller die alle andere hardware zal aansturen. Er wordt hierbij ook gebruik gemaakt van de geïntegreerde Wi-Fi om data te kunnen versturen naar een webinterface. Bij dit project wordt de ESP32 uitgebreid met een PCB om door middel van vastgesoldeerde rijgklemmen makkelijk goede contacten te maken met de draadbruggen.



Dual Motor Driver Module (L298N)

Hiermee zullen er in het wagentje 2 5V DC-motoren aangedreven en aangestuurd kunnen worden met een enkele spanningsbron. De Driver wordt aangestuurd door de ESP32 microcontroller (hierover meer uitleg bij het schema).

5V DC motor (x2)

Deze motoren, met aangekoppelt tandwielsysteem, zullen de auto laten rijden door middel van een aangekoppelt wiel.

5x infrarood sensor array

Hiermee kan de robot een lijn gaan volgen. Door te bepalen of er een hoge (witte) of lage (zwarte) waardes gemeten worden, kan het autootje gecorrigeerd worden.

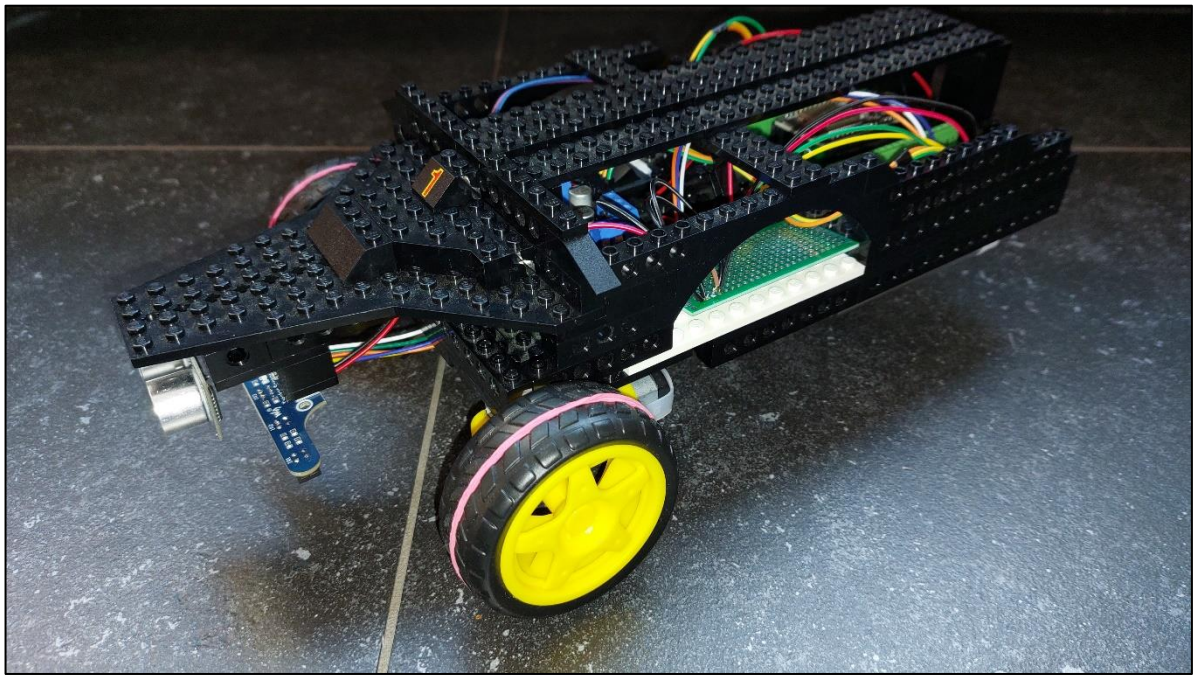
Afstandssensor (HC-SR04)

Met deze sensor zullen er vooral botsingen met andere autootjes (of andere) voorkomen kunnen worden aangezien deze geprogrammeerd zal zijn om bij een bepaalde gemeten afstand de motoren stop te zetten.

Verder worden er nog 3 LEDs, een drukknop, draadbruggen, een schakelaar voor de batterij en een printplaat gebruikt. Ook werd er een Odroid C4 gebruikt, hierover later meer uitleg.

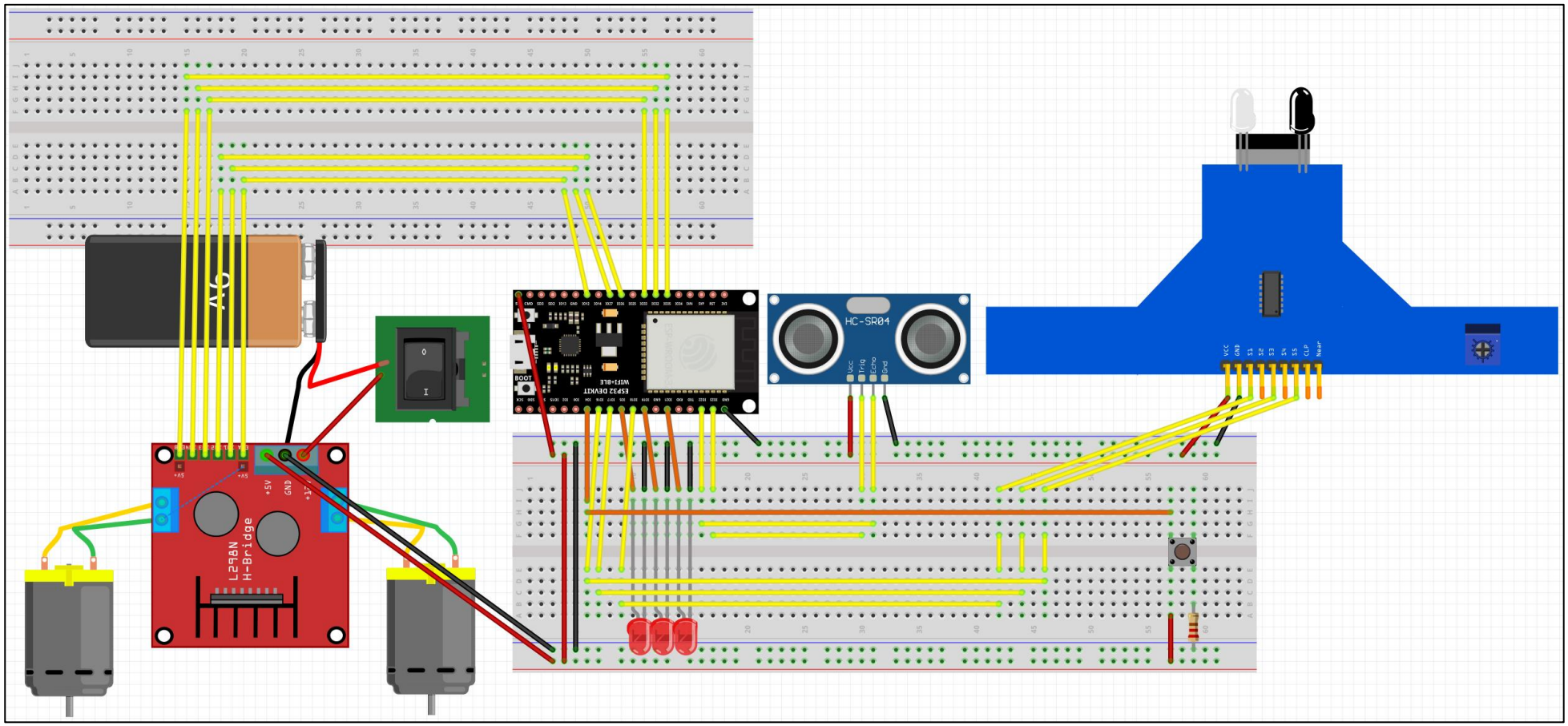
<u>Behuizing:</u>

Voor de behuizing heb ik gewerkt met diverse lego blokken om de robot vorm te geven. Dit leek in mijn geval het makkelijkste aangezien ik niet zo snel beschikking heb tot een 3D-printer om meerdere prints uit te voeren moest de vorige ge-update moeten worden.



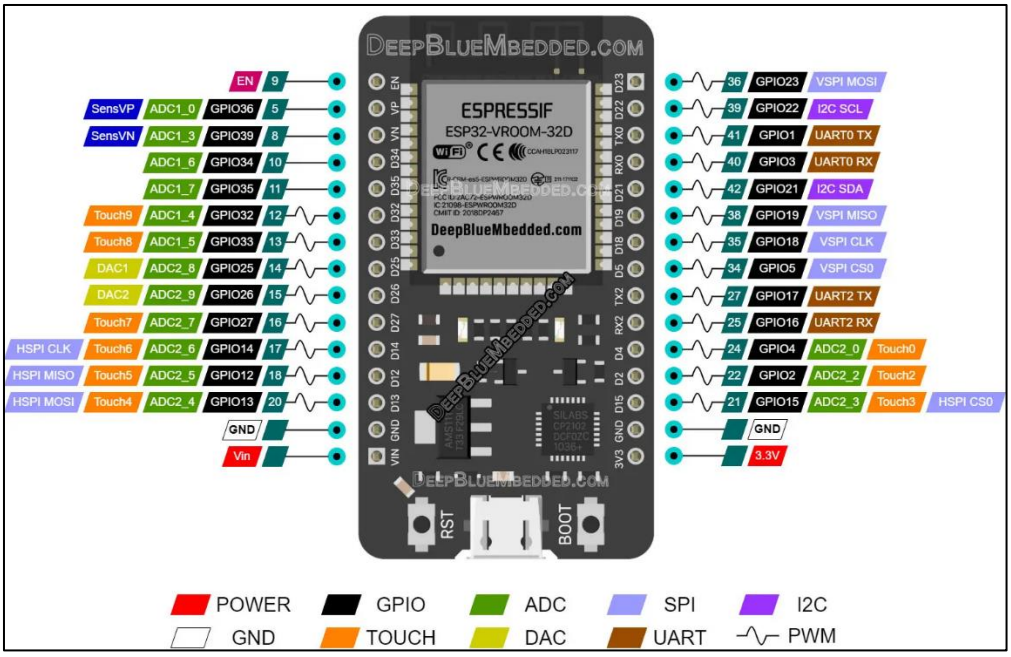
SCHEMA

De nauwkeurigheid van het schema (gebruikte GPIO) kan variëren met het daarwerkelijke project, maar is bij de uitleg niet van toepassing.



Uitleg per component:

De **afstandssensor** is aangesloten aan 5V en wordt aangestuurd met 2 pins (echo en trig), 1 voor de ontvanger, GPIO 5, en 1 voor de emitter, GPIO 4. In de code wordt echo ingesteld op INPUT en trig op OUTPUT. Hierbij moet er rekening gehouden worden met welke GPIO er exact gebruikt worden aangezien deze analoog signalen moeten kunnen meten en INPUT of OUTPUT moeten kunnen regelen.



De **infrarood sensor array** is ook aangesloten aan 5V en aan 3 pins. Van deze array worden er dus 2 sensoren niet gebruikt aangezien deze te dicht bij elkaar staan voor dikte van de lijn (route.docx). Alle pins (GPIO 25, 33 en 32) worden ingesteld op INPUT.

De **driver** wordt gevoed door een 9V batterij (met schakelaar) en biedt op zijn beurt ook 5V aan om in dit geval de ESP32 te kunnen voeden. De VIN poort op de ESP32 kan namelijk gebruikt worden zowel als input als output van 5V. De "enable" pins (snelheid van de motor) en de "in" pins om de motoren aan en uit te zetten worden als volgt bekabelt:

Motor 1:

IN1	IN2	ENA
26	27	12

Motor 2:

IN3	IN4	ENB
18	19	2

Er moet zeker gelet worden welke GPIO er gebruikt worden voor de “enable” pins aangezien deze pins gebruik maken van een PWM signaal om de snelheid van de motoren te regelen.

De **motoren** worden respectievelijk tegenover elkaar aangekoppelt aan hun toegewijde rijgklemmen op de driver.

De **LEDs** worden aangesloten aan GPIO 23, 13 en 14.

De **drukknop** wordt aangesloten aan GPIO 35.

Buiten het schema om wordt er ook een **Odroid** singleboardcomputer (model C4) gebruikt. Dit is eigenlijk simpelweg een kleine computer. Hierop draait een linux besturingssysteem (Ubuntu) waar op zijn beurt een MQTT server draait. Kort uitgelegd zal deze server data ontvangen die wordt gestuurd, over WiFi, van de ESP32 om zo via een “brug” script een database te vullen met informatie. In dit geval zal dit de afstand zijn die wordt gemeten door de ultrasone sensor. De database is op zijn beurt verbonden met een webinterface, grafana, die de data zal weergeven.

CODE (ARDUINO IDE – C++)

```
// ----------------- PINS -----------------
#define ena 12
#define in1 26
#define in2 27

#define enaB 2
#define in3 18
#define in4 19

#define ir3Pin 25 // orange wire (IR)
#define ir1Pin 33 // blue wire (IR)
#define ir5Pin 32 // white wire (IR)

#define echoPin 5 // purple wire (HC)
#define trigPin 4 // blue wire (HC)

#define redPin 23
#define orangePin 13
#define greenPin 14

#define button 35
```

De code begint met het definïeren van de pins om de driver, de sensoren, LEDs en de drukknop aan te sturen. Hierbij moet er steeds gelet worden op welke pins bruikbaar zijn voor de use case van dat component.

De “in” pins van de driver moeten aangesloten zijn op digitale GPIO die output ondersteunen. De GPIO 34 en 35 ondersteunen bijvoorbeeld enkel INPUT, daarom ook dat de drukknop op een van deze aangesloten is. Bij de “enable” pins van de driver moet gebruik gemaakt worden van een PWM signaal om een zelf instelbare snelheid in te stellen tussen 0-255.

Aangezien de infrarood sensor een analoge waarde uitleest, moeten de sensoren aangesloten worden op analoge GPIO. Voor de emitter en ontvanger van de ultrasone sensor geldt hetzelfde.

De LEDs worden aangesloten op digitale GPIO. Er wordt in dit project niet gewerkt met de helderheid van de LEDs, dus moet er geen rekening gehouden worden met PWM.

```
// ---------------- GLOBAL VARIABLES ----------------
int sped = 200;

float duration;
float distance;

int ir3;
int ir1;
int ir5;
int m = 15;

char distanceArray[20];
```

Bij de snelheid van de motoren en de sensoren worden er gobale variabelen gedeclareerd om duidelijke en verstaanbare code te schrijven. Deze worden globaal gedefinieerd zodat ze overal in de rest van de code gebruikt kunnen worden.

De marge (m) variabele wordt later gebruikt bij de beweging van het autootje en de afstand dit de ultrasone sensor meet gaat met behulp van een array weergegeven kunnen worden op een webinterface (Grafana).

```
// ---------------- WIFI & MQTT ----------------
#include <WiFi.h>
#include <PubSubClient.h>
WiFiClient espClient;
PubSubClient client(espClient);

const char* ssid = "embed";
const char* password = "weareincontrol";
const char* mqtt_server = "192.168.1.231";
const int mqtt_port = 1883;
const char* mqtt_username = "maximbozek";
const char* mqtt_password = "odroid";
```

Om met de webinterface te kunnen werken zal ESP32 moeten verbinden met een WiFi netwerk. Hiervoor zijn libraries nodig. De character variabelen daaronder zijn er ook weer om duidelijke en verstaanbare code te schrijven.

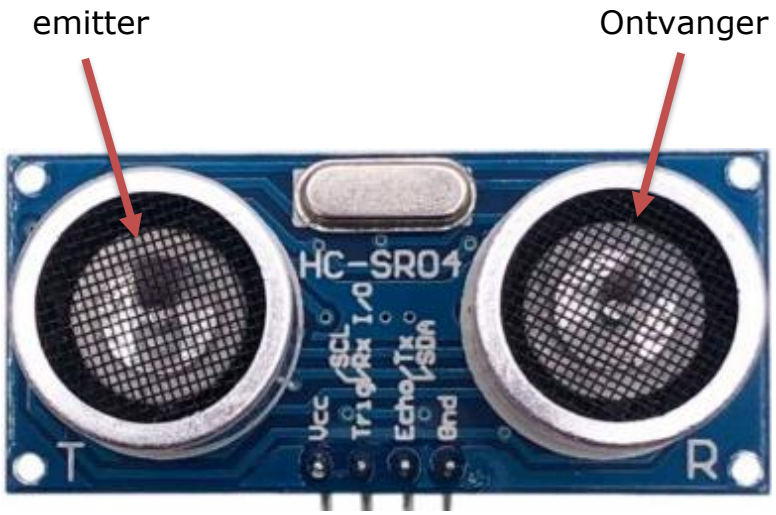
```
void setup() {
  Serial.begin(115200);

  // ---------------- PINMODES ----------------
  pinMode(ena, OUTPUT);
  digitalWrite(ena, HIGH);
  analogWrite(ena, sped);

  pinMode(enaB, OUTPUT);
  digitalWrite(enaB, HIGH);
  analogWrite(enaB, sped);

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  pinMode(ir3Pin, INPUT);
  pinMode(ir1Pin, INPUT);
  pinMode(ir5Pin, INPUT);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(redPin, OUTPUT);
  pinMode(orangePin, OUTPUT);
  pinMode(greenPin, OUTPUT);

  pinMode(button, INPUT);
```

De code bevat 2 hoofd functies. De setup -en de loop functie. In de setup functie worden alles geschreven wat maar 1 keer doorlopen moet worden. De loop functie wordt doorlopen totdat de ESP32 niet meer van energie voorzien wordt. Bijvoorbeeld voor de snelheid van de motoren in te stellen is het niet nodig om dit keer op keer te doen.

Voor de beweging van de robot daarintegen is het cruciaal om heel de tijd te controleren wat de infrarood sensor meet om zo de volgende bewegingsmogelijkheid te bepalen.

In de setup functie wordt als eerste connectie gestart met de seriële monitor. Deze werd gebruikt om de positie infrarood sensor te optimaliseren en zal later gebruikt kunnen worden bij het troubleshooten van de robot, moest een van de sensoren defect.

Daarna wordt elke pin ingesteld of deze een output moeten ontvanger of een input zullen doorgeven. De driver verwacht een output van de ESP32 om de motoren aan te sturen en de sensoren zullen data doorgeven (inputten) die wordt gemeten. De “trig” pin (emitter) van de ultrasone sensor staat ingesteld op output aangezien dit deel van de HC-SR04 module puur zal zorgen voor de ultrasone geluidgolf die op zijn beurt terug ontvangen wordt (echo).



```
  // ----------------- WIFI & MQTT -----------------
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting...");
  }
  Serial.println("Connected to wifi.");
  Serial.println("-----");

  client.setServer(mqtt_server, mqtt_port);
}
```

Om de setup functie af te sluiten wordt de ESP32 verbonden met een WiFi netwerk. Hierover zal ook een bericht gestuurd worden de seriële monitor moest dit gebeurt zijn. Er wordt ook elke halve seconde een “connecting…” bericht gestuurd.

Ook wordt de server connectie opgezet met welke server en poort de ESP32 connectie moet maken.

```
void loop() {
  // ---------------- MQTT CONNECTION ----------------
  client.loop();

  while (!client.connected())
  {
    if (client.connect("ESP32Client", mqtt_user, mqtt_passw))
    {
     Serial.println("connected to MQTT");
     Serial.println("-----");
    }
    else
    {
      Serial.print("failed with state ");
      Serial.println(client.state());
      Serial.println("-----");
      delay(2000);
    }
  }
```

Aan het begin van de loop functie wordt een “client.loop()” gedefinieerd. Deze is nodig om elke keer weer te controleren of er nieuwe data is die naar de MQTT server, die op de Odroid draait, moet gestuurd worden zodat er op de webinterface steeds geupdate data binnenkomt.

Ook wordt hieronder daadwerkelijk connectie gemaakt met de server. Dit wordt niet in de setup functie gedaan aangezien het handig is om zelfs na een correcte connectie, moest de connectie wegvallen, error codes als berichten op de seriële monitor kunnen weergegeven worden die makkelijker tot oplossingen kunnen leiden tijdens het troubleshooten.

```
// ---------------- SENSOR READS & PUBLISHES ----------------
digitalWrite(trigPin, LOW);
delayMicroseconds(5);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);
distance = (duration * 0.0343) / 2;

dtostrf(distance, 2, 2, distanceArray);
client.publish("home/linerobot/distance", distanceArray);
```

Om gebruik te kunnen maken van de afstandssensor moet het “trig” module geactiveerd worden. Dit werkt het beste deze, in dit 10 microseconden, een kleine periode per keer actief is (aangezien de loop fucntie oneindig keren herstart).

De uitgelezen data, door het “echo” module wordt opgeslagen in een eerder globaal gedeclareerde variabele, waar op zijn beurt een berekening op wordt uitgevoerd die deze data in centimeters zal weergeven. Deze centimeters worden opgeslagen in een nieuw variabele.

Daarna zal een “dtostrf()” functie helpen om deze data om te zetten van een float waarde in character array. Dit doen we omdat de “client.publish()” functie elkel characterized formaten (strings of arrays) kan verwerken. Een tag (home/linerobot/distance) zal de “client.publish()” functie helpen de data naar de correcte bestemming te sturen.

```
// ---------------- CAR MOVEMENT ----------------
ir3 = map(analogRead(ir3Pin), 0, 4096, 0, 20);
ir1 = map(analogRead(ir1Pin), 0, 4096, 0, 20);
ir5 = map(analogRead(ir5Pin), 0, 4096, 0, 20);

if(distance > 10)
{
  if ((ir1 <= m) && (ir3 > m) && (ir5 <= m))
  {
    carDrive();
    green();
  }
```

Voor de beweging van start de code met het inlezen van de sensoren. Deze lezen een waarde in van 0-4096 omdat deze analoog zijn. Om gemakkelijker met de sensoren te werken worden ze gemappt naar een nieuw minimum en maximum van 0-20. Dit maakt het ook makkerlijker om te zien wat de sensoren juist meten bij het troubleshooten.

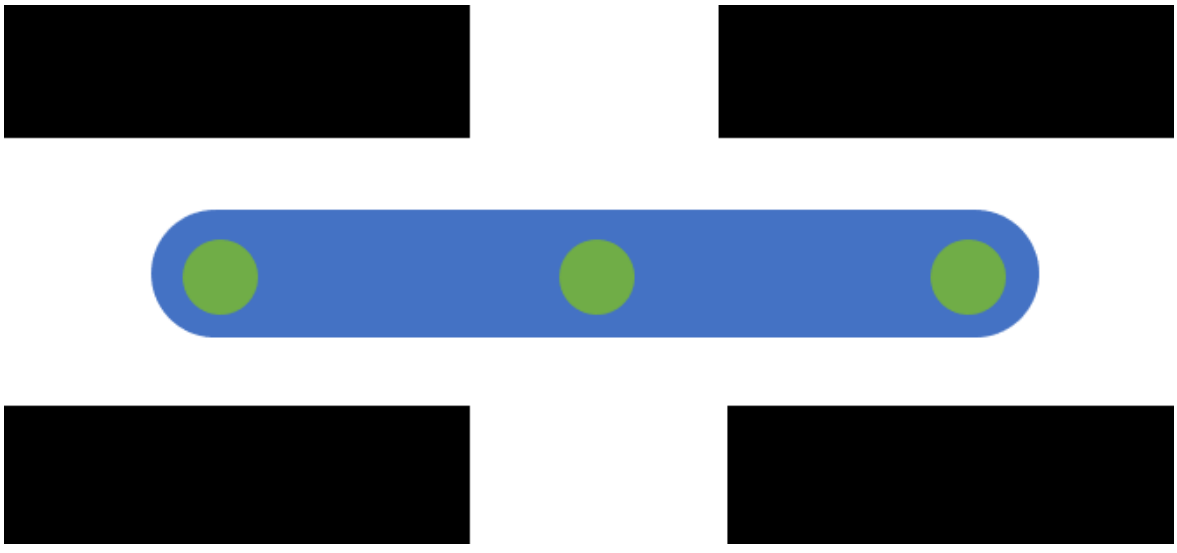
Daarna start er een serie aan if-statements. De beweging van de robot zit eigenlijk allemaal genest in een hoofd if statement. Als de afstandssensor een waarde van boven de 10 cm meet, gaat de robot kunnen bewegen.

De auto zal rijden met de “carDrive()” functie, later meer hierover. Ook de configuraties van de LEDs zijn verwerkt in aparte functies voor een duidelijkere en verstaanbaardere code. Enkel als de middenste sensor een witte kleur meer, en de 2 buitenste zwart meten, zal de robot recht vooruit rijden.

Hierbij wordt gebruik gemaakt van een marge (m) om de if statement te doen werken. De if statement werkt als volgt: Als de linkse sensor kleiner of gelijk is aan de marge, EN de middenste sensor groter is dan de marge, EN de rechtse sensor is kleiner of gelijk aan marge, dan zal het de “carDrive()” functie en de “green()” functie uitvoeren.

```
    else if ((ir1 > m) && (ir3 > m) && (ir5 > m))
    {
      carStop();
      orange();
      for(int i = 0;i <= 5000;)
      {
        i++;
        delay(1);
        if(digitalRead(button) == 1)
        {
          break;
        }
      }
      // delay(2000);
      carDrive();
      green();
      delay(200);
    }
```

Bij de specificaties van het autootje stond er dat bij elk stuk van het museum een witte lijn ervoor moet zorgen dat de robot 5 minuten stil zal komen te staan. Moest de bezoeker geen interesse hebben in dat stuk kan hij/zij een drukknop gebruiken om direct verder te rijden.

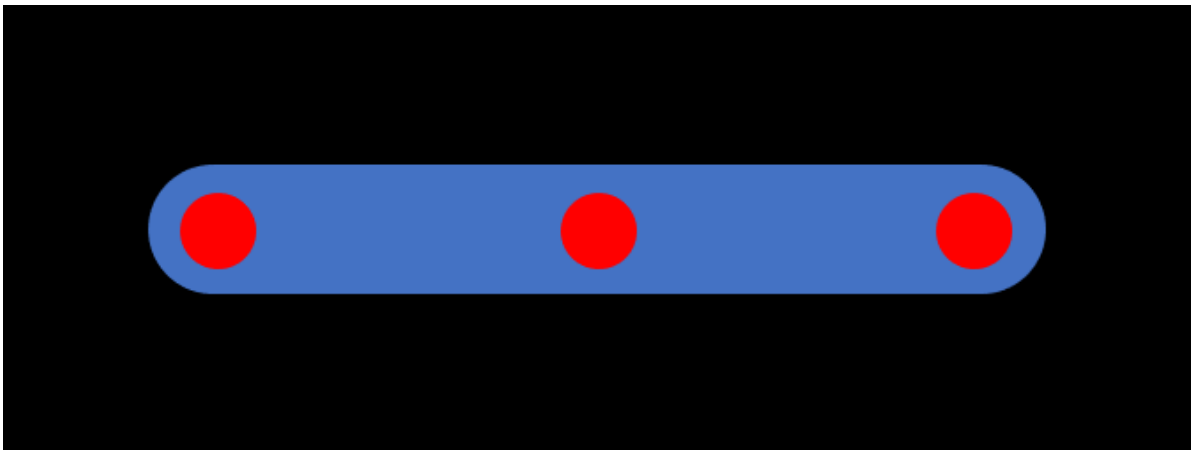


Bij deze if statement stopt de auto meteen nadat de infrarood sensor een witte lijn tegenkomt. De oranje LED gaat aan en we komen terecht in een for loop.

Deze for loop wordt uitgevoerd zoals als "i" kleiner of gelijk is aan 5000. "i" wordt elke milliseconde ("delay(1)"), 1 keer opgeteld ("i++"). Moest de drukknop een waarde doorgeven van 1, dus moest de bezoeker erop drukken, zal er een "break" worden uitgevoerd en zal de for loop direct verlaten worden, waarna de auto tijd heeft (200 milliseconden) om het volgende stuk van de baan terug op te pakken. De groene LED gaat terug aan.

```
else if ((ir1 <= m) && (ir3 <= m) && (ir5 <= m))
{
  carStop();
  red();
}
else if ((ir1 = 0) && (ir3 = 0) && (ir5 = 0))
{
  carStop();
  red();
}
else if ((ir1 = 0) || (ir3 = 0) || (ir5 = 0))
{
  carStop();
  red();
}
```

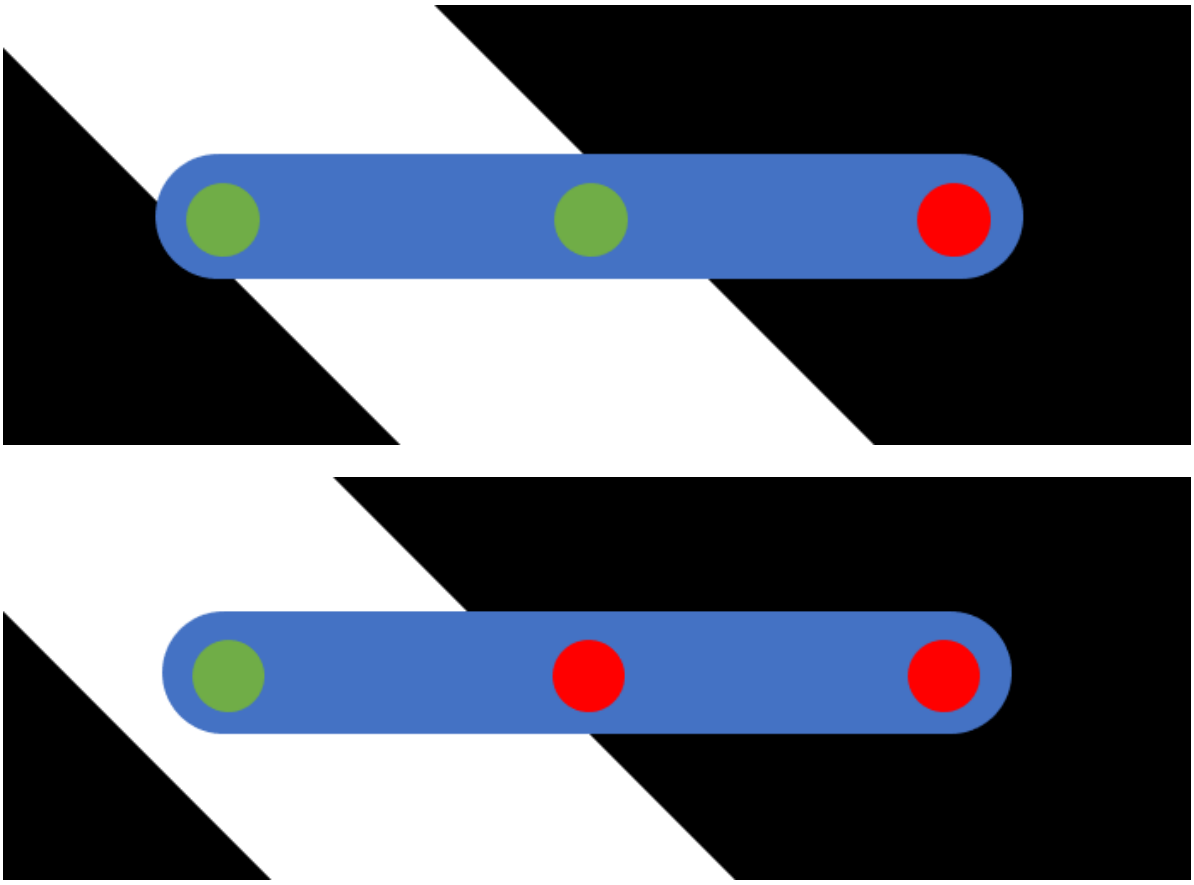
Moest de sensor de lijn kwijtraken, waardoor hij enkel zwart meet, komt hij stil te staan ("carStop()") en gaat de rode LED aan.



Moest de sensor defect zijn zal er een waarde van 0 doorgegeven worden, aangezien er niet meer gemeten wordt. Dit zal ook resulteren in een stilstaande auto en een rode LED.

```
    else if ((ir1 > m) && (ir3 > m) && (ir5 <= m))
    {
      carLeft();
      green();
    }
    else if ((ir1 > m) && (ir3 <= m) && (ir5 <= m))
    {
      carLeft();
      green();
    }
```

De auto zal een beweging maar links (“carLeft()”) maken als de linkse en de middenste sensoren wit meten of als enkel de linkse wit meet. Dit maakt het zodat moest de auto iets schuiner aankomen bij een bocht, kan het in beide gevallen zodoende naar links bewegen.



```
    else if ((ir1 <= m) && (ir3 > m) && (ir5 > m))
    {
      carRight();
      green();
    }
    else if ((ir1 <= m) && (ir3 <= m) && (ir5 > m))
    {
      carRight();
      green();
    }
  }
```

Hetzelfde geldt voor een beweging naar recht (“carRight()”), maar dan alles omgekeerd voor de code van een beweging naar links.

```
  else if(distance <= 10)
  {
    carStop();
    orange();
  }
```

Om al deze geneste bewegings statements af te sluiten stopt de auto als de afstandssensor een waarde van kleiner of gelijk aan 10 cm meet.

```
void carStop(){
  digitalWrite(in1,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,LOW);
}

void carDrive(){
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
}

void carLeft(){
  digitalWrite(in1,LOW);
  digitalWrite(in2,HIGH);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
}

void carRight(){
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
}
```

Nu al deze functies om de auto te later bewegen zijn vrij eenvoudig gemaakt. Er wordt een digitaal signaal aan beide “in” pins van de driver gestuurd om zo de motoren, of vooruit, of achteruit te later bewegen.

Om een beweging vooruit (“carDrive()”) te maken worden de eerste pins van elke motor op high gezet.

Om een beweging naar links te maken wordt bij de linker motor de polariteit verandert om het linker wiel achteruit te laten draaien. Zo maakt de auto een beweging zonder dat de infrarood sensor de baan verliest (later meer over de plaatsing).

Hetzelfde geldt voor een beweging naar rechts, maar dan alles omgekeerd van wat er gebeurt bij “carLeft()”.

Om de auto stop te zetten wordt er simpelweg een laag signaal naar beide “in” pins gestuurd.

```
void red(){
  digitalWrite(redPin, HIGH);
  digitalWrite(orangePin, LOW);
  digitalWrite(greenPin, LOW);
}

void orange(){
  digitalWrite(redPin, LOW);
  digitalWrite(orangePin, HIGH)
  digitalWrite(greenPin, LOW);
}

void green(){
  digitalWrite(redPin, LOW);
  digitalWrite(orangePin, LOW);
  digitalWrite(greenPin, HIGH);
}
```

Om de LEDs te configureren was het zeer makkelijk om functies hiervan te maken aangezien bij het aangaan van een LED elke andere LED uit moet gaan. Er wordt respectievelijk per functie een hoog signaal gestuurd naar de desbetreffende LED en een laag naar de andere.

PRODUCTIE

Bij de aanpak van de productie zijn er een paar belangrijke punten:

- De infraroodsensor MOET recht tussen of voor de wielen worden geplaatst, anders werkt de beweging helemaal niet zoals gewild.
- De behuizing man de afstandssensor NIET blokkeren.
- De behuizing mag niet te zwaar wegen waaronder de motoren anders bezwijken onder het gewicht.
- Er moet genoeg stroomsterkte vanuit de voeding naar de motoren worden gestuurd voor vlotte beweging.
- Om de behuizing van de robots te kunnen voorzien kan er verwezen worden naar de eerder afgenomen presentatie. Er zou een samenwerkingsakkoord kunnen ontstaan tussen Lego en Mercedes om de behuizing van de auto's te voorzien.

<u>Productieproces:</u>

De wagen ontworpen door mij kan gezien worden als een soort prototype (hierover later meer).

- De ESP32 wordt voorzien van de nodige software om de componenten goed aan te sturen (code).
- De componenten worden bevestigd aan de behuizing en correct aangesloten (uitleg schema), voorzien door Lego in verband met een mogelijke samenwerking.
- De robots worden getest op de daadwerkelijk route die ze zullen afleggen in het museum en kunnen op basis van de resultaten van zo'n test getweaked worden om keer op keer gebruik perfect te laten verlopen.

ZELFREFLECTIE

Na het maken van het project zijn er zeker een aantal aspecten die in, met de kennis die ik nu heb, beter zou doen in de toekomst.

- De gekozen infrarood sensor las analoge waardes uit wat niet echt paste bij de functionaliteit van de robot. Hierdoor kon de sensor niet mooi tegen de grond worden geplaatst was het volgen van de lijn veel gemakkelijker hads laten verlopen.
    - De beste oplossing hievoor zou simpelweg zijn om 3 aparte afstelbare IR sensoren te bevestigen aan de auto om ze zeer optimaal af te stellen aan de route.
- De voeding voor de motoren bracht niet genoeg stroomsterkte met zicht meer waardoor de motoren vaak niet genoeg power hadden om voort te bewegen.
    - Een goede oplossing hiervoor zijn kunnen zijn om 2 9V batterijen in parallel met elkaar te schakelen waardoor ze 9V blijven leveren, maar tevens 2x zoveel stroomsterkte voorzien.
- Het solderen van de ground draden van de LEDs en de sensoren en 5V draden van de sensoren had veel makkelijker gegaan door een reeks rijklemmen te solderen op de printplaat in de plaats van de draden rechtstreeks te solderen aan de printplaat.
- Alhoewel de lego behuizing mij niet slecht afging is het natuurlijk veel voordeliger om een behuizing te 3D printen. Dit had bij productie van de daadwerkelijke autootjes veel makkelijker gegaan en had er niet gerekend moeten worden op een samenwerking met Lego.