**Documentatie project**



**Inhoud**

[**Elektronica componenten** 3](#_Toc136373989)

[- **ESP32** 3](#_Toc136373990)

[- **ESP32 board** 4](#_Toc136373991)

[- **motor driver (L298N)** 5](#_Toc136373992)

[- **2x (DC)motor** 6](#_Toc136373993)

[- **IR-sensor** 8](#_Toc136373994)

[- **Ultrasone sensor** 9](#_Toc136373995)

[Odroid C4 10](#_Toc136373996)

[Software 10](#_Toc136373997)

[Arduino IDE 10](#_Toc136373998)

[Python 10](#_Toc136373999)

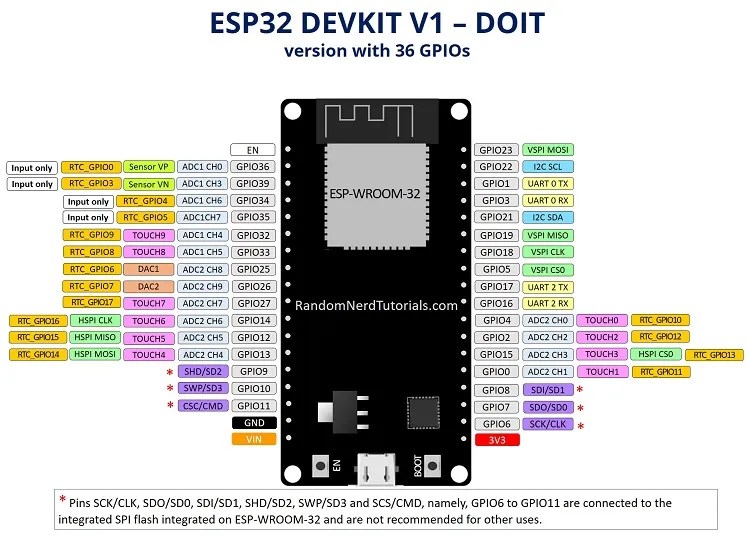
[TUTORIAL 11](#_Toc136374000)

[**Bronnen** 20](#_Toc136374001)

# **Elektronica componenten**

## **ESP32**

De ESP32 is een compacte en low-cost microcontroller. Dit model heeft Wi-Fi (2.4 GHz, tot 150Mbps) zodat deze makkelijk met een netwerk kan verbinden alsook een eigen Wi-Fi netwerk creëren (access point mode). Deze heeft Bluetooth classic en Bluetooth Low Energy (BLE).

* 34 programmeerbare GPIO’s
* 12-bit SAR ADC tot 18 kanalen
* 2x 8-bit DAC
* 10x touch sensors
* 4x SPI
* 2x I2S
* 2x I2C
* 3x UART
* 1 host (SD/eMMC/SDIO)
* 1 slave (SDIO/SPI)
* RMT (TX/RX)
* Motor PWM
* 16 kanalen LED PWM

Afbeelding met tekst, elektronica, Elektronische engineering, schermopname

Automatisch gegenereerde beschrijving

De reden dat ik deze gebruik is omdat deze compatibel is met de arduino “programmeer taal”.

## **ESP32 board**

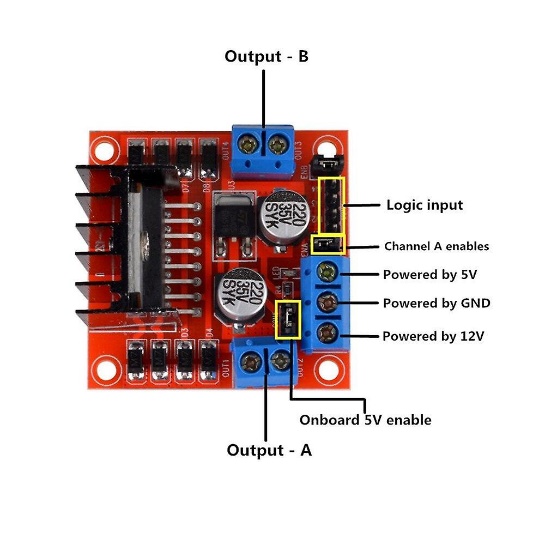
Dit heb ik gebruikt om makkelijk alle sensoren en drivers aan te sluiten zonder te moeten solderen op de ESP32. dit is niet noodzakelijk voor dit project maar wel handig (deze kan je makkelijk op Amazon kopen voor een paar eurootjes). Doormiddel van deze rijgklemmen blijft alles goed vastzitten en kunnen we makkelijk aanpassingen maken, terwijl als je alles soldeert en je wil iets veranderen dan is dit een heel gedoe.

Afbeelding met tekst, elektronica, schermopname, software

Automatisch gegenereerde beschrijving

## **motor driver (L298N)**

Deze stuurt de twee motoren aan. Om deze driver te voorzien van stroom te voorzien gebruik ik een 9V batterij. De 5V output word gebruikt om de ESP32 te voorzien van spanning. Deze driver kan motoren aansturen tussen 5V en 35V met een piekstroom tot 2A. Output A en Output B dienen om de motoren op aan te sluiten. Ook zijn er twee channels (A en B) deze worden gebruikt om de motoren aan te zetten en de snelheid van de motoren te regelen doormiddel van een PWM signaal. Input 1 en 2 word gebruikt voor motor A en input 3 en 4 voor motor B. Als input 1 laag is en input 2 hoog dan zal de motor vooruit draaien en als input 1 hoog is en input 2 laag dan zal de motor achteruit draaien. Als beide inputs laag zijn dan stopt de motor met draaien. Dit geld ook voor input 3 en 4.



Afbeelding met tekst, elektronica, schermopname, software

Automatisch gegenereerde beschrijving

## **2x (DC)motor**

De 2 motoren die gebruikt zijn om de auto te laten rijden werken op 5V. Dit zijn twee DC motoren. Deze zijn heel gemakkelijk aan te sluiten doormiddel van twee draden (+ en -). Omdat deze klein zijn kunnen we deze dus makkelijk voeden met een lage spanning.

Afbeelding met tekst, schermopname, software, Computerpictogram

Automatisch gegenereerde beschrijving

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | De werking van een gelijkstroom motor (DC) | | | [Afbeelding met ontwerp  Beschrijving automatisch gegenereerd met lage betrouwbaarheid](https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_1.png) | Wanneer er een gelijkstroom loopt door de draaiende wikkeling of ankerwikkeling, wordt er in het anker (ook rotor genoemd) een magneetveld opgewekt. De linkerzijde van de rotor wordt weggedrukt van de vaste linkermagneet en aangetrokken door de vaste rechtermagneet (de stator), waardoor de rotor begint te draaien. | | [Afbeelding met ontwerp  Beschrijving automatisch gegenereerd met lage betrouwbaarheid](https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_2.png) | De rotor draait 180°. | | [Afbeelding met ontwerp  Beschrijving automatisch gegenereerd met lage betrouwbaarheid](https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_3.png) | Wanneer de rotor horizontaal is aangekomen bij de juiste polen zal de beweging stoppen. De enige manier om de rotor verder door te doen draaien is het omkeren van de polen op het draaibaar anker. Dit gebeurt door de collector. Deze keert de richting van de stroom door de spoelen om waardoor het magneetveld in de rotor of anker genaamd omkeert. De linkerzijde van de rotor wordt weggedrukt van de vaste linkermagneet en aangetrokken door de vaste rechtermagneet, waardoor de rotor, ook anker genoemd, verder gaat draaien. | | [Afbeelding met plastic  Automatisch gegenereerde beschrijving](https://commons.wikimedia.org/wiki/File:Electric_motor.gif) | Dit proces herhaalt zich steeds opnieuw, waardoor de motor continue blijft ronddraaien | |

## **IR-sensor**

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijvingPower: 3.3V ~ 5V

Afmetingen: 78mm x 18mm

Detector spacing: 16mm

Mounting holes size: 3mm

Detection range: 1cm ~ 5cm

5x IR sensor

Deze IR sensor word gebruikt in dit project om de witte lijn te detecteren. Dit doormiddel van infrarood licht dat reflecteert. Aan de hand van hoeveel ligt er weerkaatst en terug word ontvangen weet deze sensor waar de witte lijn ligt. Dit omdat wit beter reflecteert dan zwart. Een nadeel dat ikzelf heb ondervonden is dat de sensoren te dichtbij elkaar staan om goed te kunnen functioneren. De sensor aansluiten kan heel gemakkelijk door een 5V en ground aansluiting en een data pin voor elke sensor (5 stuks).

## **Ultrasone sensor**

Met deze sensor kunnen we ervoor zorgen dat de auto zal stoppen wanneer deze te dicht tegen een object komt. Dit werkt doormiddel van ultrasone golven (40 kHz). 2cm tot 400cm met een nauwkeurigheid van 0.3cm. deze sensor is makkelijk aan te sluiten met een 5v en ground pin. de trigpin dient om de ultrasone golven uit te sturen en de echopin het woord zegt het zelf de echo terug op te vangen. Aan de hand van hoelang de geluidsgolven erover doen om terug tot de sensor te komen weten we hoe ver een object verwijderd is van de sensor. Een belangrijk puntje bij deze sensor is dat deze niet reageert onder de 2cm.

Power: +5V DC

<2mA

15mA

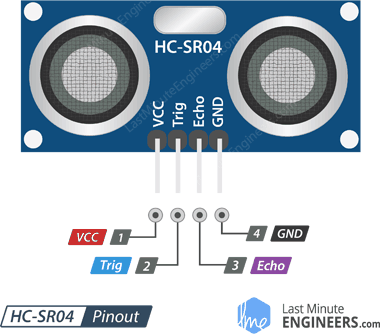
<15°

2cm – 400cm

0.3cm

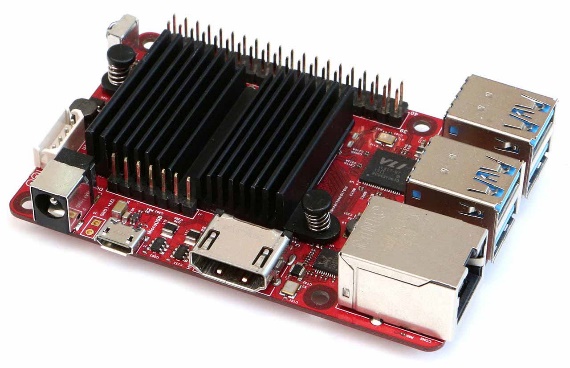
Angle 30°

Dimensions 45mm x 20mm x 15mm



## **Odroid C4**

Dit is ook een microcontroller. Deze gaan we gebruiken om onze MQTT te laten werken. voordat we deze odroid kunnen gebruiken moeten we er een besturingssysteem opzetten via een SD kaart. Het besturingssysteem dat wij gaan gebruiken is ubuntu mate. Deze heeft ook aansluitingen zoals de ESP32 maar deze gebruiken we niet in dit project.



# Software

## Arduino IDE



Dit is de software van arduino zelf waarmee we microcontrollers kunnen programmeren.

## Python

Dit is ook een programma waarmee we code kunnen schrijven en gaan we gebruiken om onze data via mqtt in een database en zo in grafana te kunnen weergeven.

# TUTORIAL

Om te beginnen met deze tutorial, heb je alle componenten nodig en de juiste software.

Om de arduino IDE software te installeren

<https://www.arduino.cc/en/software>

volg nu de stappen van de installatie

om ervoor te zorgen dat we onze ESP32 met deze software kunnen programmeren moeten we nog een bestand installeren.

<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

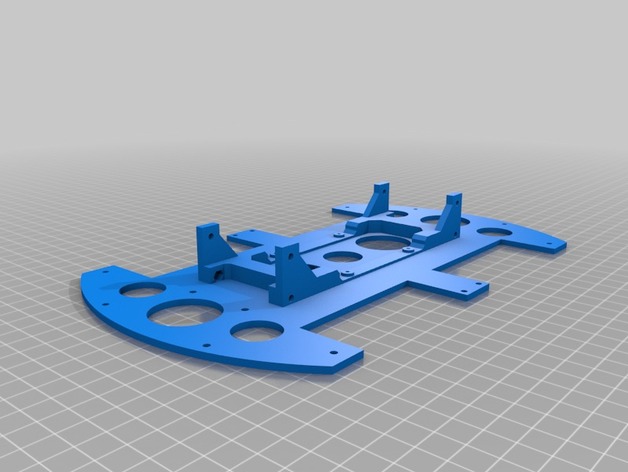
Als je de bovenstaande tutorial volgt dan kunnen we verder

Om te beginnen hebben we een aansluitingsschema nodig en een PCB schema. Deze heb ik getekend in fritzing, je kan er ook voor kiezen een ander programma te gebruiken zoals eagle. In deze programma’s zitten al heel veel standaard componenten, indien er een onderdeel is dat er niet tussenzit kan je deze makkelijk downloaden van internet en toevoegen aan de lijst.

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Als eerste zullen we een auto moeten bouwen. Dit kan je doen door een frame te 3D printen. Monteer op het frame je componenten.

Zorg ervoor dat de IR en ultrasone vooraan de auto worden bevestigd.

als eerste in de code wijzen we onze pinnen van de ESP32 toe aan onze componenten. Dit doen we door int eigen gekozen naam van de pin = pin.

deze pinnen zijn voor de motordriver.

// Motor A

int motor1Pin1 = 21;

int motor1Pin2 = 22;

int enable1Pin = 12;

// Motor B

int motor2Pin1 = 18;

int motor2Pin2 = 19;

int enable2Pin = 2;

de volgende pinnen zijn voor onze IR-sensor. Elke pin krijgt data binnen van 1 sensor.

// IR-sensor

int ir5 = 32;

int ir4 = 33;

int ir3 = 15;

int ir2 = 27;

int ir1 = 14;

om de led’s te kunnen aansturen sluiten we deze ook aan op een pin.

// LED'S

int LEDROOD = 23;

int LEDGROEN = 13;

drukknop

// drukknop

int button = 5;

int isgestopt;

unsigned long startMillis;  //some global variables available anywhere in the program

unsigned long currentMillis;

const unsigned long period = 1000;  //the value is a number of milliseconds

voor de ultrasone sensor hebben we een pin (trigpin) die ervoor zal zorgen dat de ultrasone sensor een ultrasone golf zal uitzenden. De echopin dient om de data te laten binnenkomen van de sensor.

// ultrasone sensor

int trigPin = 26;

int echoPin = 25;

omdat de data van de ultrasone sensor variabel is maken we een float. Ook kunnen we de data van onze ultrasone niet zomaar doorsturen via mqtt maar moeten we deze eerst in een array steken en deze zo doorsturen.

// ultrasone sensor

int trigPin = 26;

int echoPin = 25;

float usread;

float cm;

// ultrasone sensor array

char usarray[5];

void forward();

// Setting PWM properties

const int freq = 30000;

const int pwmChannel = 0;

const int resolution = 8;

int dutyCycle = 200;

Nu gaan we de MQTT opzetten. Als eerste moeten we de twee librarys toevoegen. De wifi.h zal ervoor zorgen dat we een wifi verbinding kunnen maken. De pubsubclient dient om te kunnen publishen naar de odroid.

Om een verbinding met wifi te kunnen maken hebben we de naam en het wachtwoord nodig. De mqtt server moet overeen komen met die van je odroid. De mqtt port is standaard 1883. Om de data te kunnen publishen naar een database in de odroid vul je je username en wachtwoord in van influxdb.

//MQTT

#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient espClient;

PubSubClient client(espClient);

const char\* ssid = "embed"; // je eigen SSID

const char\* password = "weareincontrol"; // je WiFi wachtwoord

const char\* mqtt\_server = "192.168.1.90"; // je IP address van je odroid

const int mqtt\_port = 1883;

const char\* mqtt\_username = "maxim"; // username van je influxDB

const char\* mqtt\_password = "maxim"; // wachtwoord van je influxDB

we hebben eerder in deze tutorial al onze pinnen toegewezen aan onze componenten. Nu moeten we het programma nog laten weten wat inputs zijn en wat outputs zijn. Dit doen we doormiddel van pinMode -> de naam van de pin -> input of output. Een input is bijvoorbeeld een sensor die data doorgeeft naar het programma en een output is bijvoorbeeld een led die aangestuurd word door het programma. De pinnen voor de motordriver zijn allemaal outputs, terwijl de led’s en de IR sensoren allemaal inputs zijn. Bij de ultrasone sensor is de trigpin een output en de echopin een input. Dit omdat de trigpin een ultrasonegolf uitstuurt en de echopin deze terug ontvangt na een bepaalde tijd.

void setup() {

  // motorpinnen als outputs:

  pinMode(motor1Pin1, OUTPUT); // draairichting regelen motor A

  pinMode(motor1Pin2, OUTPUT); // draairichting regelen motor A

  pinMode(enable1Pin, OUTPUT); // motor enabelen en de snelheid regelen

  pinMode(motor2Pin1, OUTPUT); // draairichting regelen motor B

  pinMode(motor2Pin2, OUTPUT); // draairichting regelen motor B

  pinMode(enable2Pin, OUTPUT); // motor enabelen en de snelheid regelen

  // ir sensor pinnen als inputs:

  pinMode(ir5, INPUT);

  pinMode(ir4, INPUT);

  pinMode(ir3, INPUT);

  pinMode(ir2, INPUT);

  pinMode(ir1, INPUT);

  //LED'S

  pinMode(LEDROOD, OUTPUT);

  pinMode(LEDGROEN, OUTPUT);

  // button

  pinMode(button, INPUT);

  // ultrasone sensor

  pinMode(trigPin, OUTPUT); // transmitter

  pinMode(echoPin, INPUT); // ontvanger

dit stukje is om de twee outputs van de motordriver te activeren en de snelheid aan te passen.

  // motor driver

  digitalWrite(enable1Pin, HIGH); // motor A enabelen

  digitalWrite(enable2Pin, HIGH); // motor B enabelen

  analogWrite(enable1Pin, 150); // snelheid regelen motor A

  analogWrite(enable2Pin, 150); // snelheid regelen motor B

om onze seriele monitor te starten gebruiken we Serial.begin(de gewenste baud rate).

  Serial.begin(115200);

Met dit stukje code geven we weer in onze seriële monitor wanneer er problemen zijn met de verbinding en wanneer we geconnecteerd zijn met wifi. Deze heeft een kleine delay zodat de esp tussen de berichten door tijd heeft om toch te connecteren.

 //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);

  delay(1000);

hier doen we exact hetzelfde en controleert ons programma of we verbonden zijn met de mqtt (connected to mqtt). Indien deze niet verbonden is krijg je een bericht in de seriele monitor failed with state. Vandaar de if en else. If verbonden = print connected . else print failed. Hier staat ook een kleine delay om wat tijd tussen de berichten te laten, zo blijft de seriele monitor wat overzichtelijker.

void loop() {

  client.loop();

  while (!client.connected())

  {

    if (client.connect("ESP32Client", mqtt\_username, mqtt\_password))

    {

     Serial.println("connected to MQTT");

     Serial.println("-----------------------");

    }

    else

    {

      Serial.print("failed with state ");

      Serial.println(client.state());

      Serial.println("-----------------------");

      delay(2000);

Onze ultrasone sensor zal een ultrasone geluidsgolf uitsturen en daarna terug ontvangen doormiddel van weerkaatsing op objecten. Omdat dit een paar milliseconden duurt om terug te ontvangen en we duidelijke/zuivere data willen, gebruiken we delaymilliseconds om ervoor te zorgen dat de ultrasone golf eerst ontvangen word voor er een nieuwe geluidsgolf word uitgestuurd. Omdat de sensor geen data in cm weergeeft moeten we deze nog omrekenen. Nu we onze data in cm hebben zullen we deze in een array zetten om te kunnen doorsturen naar onze odroid via mqtt en zo dan weergeven in grafana. Het publishen doen we door client.publish te gebruiken en daarna een pad te creëren zodat onze code in de odroid weet waar deze data zich bevind.

// ultrasone sensor

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  usread = pulseIn(echoPin, HIGH);

  cm = (usread \* 0.0343) / 2;

  dtostrf(cm, 2, 2, usarray);

  Serial.println(usarray);

  delay(500);

// publish

  client.publish("home/auto/ultrasone", usarray); // publishen naar de odroid

Om de auto op de lijn te laten rijden gebruiken we al deze if’s. elke if lijn is hetzelfde buiten de waardes die anders zijn ingesteld. In de if lezen we de ir sensor uit. Deze is analoog dus gebruiken we analogRead. Het komt erop neer dat als de waarde van de sensor of de ingestelde waarde hoger of lager is tenopzichte van elkaar we de auto vooruit, link, rechts of laten stoppen.

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) >= 3800) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    forward();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) >= 3800) && (analogRead(ir3) <= 2600) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) >= 3800) && (analogRead(ir2) <= 2600) && (analogRead(ir3) <= 2600) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) <= 2600) && (analogRead(ir4) >= 3800) && (analogRead(ir5) <= 2600))

  {

    turnLeft();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) <= 2600) && (analogRead(ir4) <= 2600) && (analogRead(ir5) >= 3800))

  {

    turnLeft();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) >= 3800) && (analogRead(ir4) >= 3800) && (analogRead(ir5) <= 2600))

  {

    turnLeft();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) >= 3800) && (analogRead(ir3) >= 3800) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) >= 3800) && (analogRead(ir2) >= 3800) && (analogRead(ir3) >= 3800) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) >= 3800) && (analogRead(ir4) >= 3800) && (analogRead(ir5) >= 3800))

  {

    turnLeft();

  }

  if((analogRead(ir1) >= 3800) && (analogRead(ir2) >= 3800) && (analogRead(ir3) >= 3800) && (analogRead(ir4) >= 3800) && (analogRead(ir5) >= 3800))

  {

    Stop();

Om ervoor te zorgen te zorgen dat de auto op de witte dwars lijn zal stoppen en na een bepaalde tijd zal verder rijden plaatsen we een delay van een tijd naar keuze en daarna de functie forward. Deze zal dan terug vooruit rijden en de lijn terug oppikken. Om de delay te skippen gebruiken we een for loop voor onze drukknop. Doormiddel van deze for loop zal het programma constant luisteren naar de status van de drukknop. Eens deze wordt ingedrukt tijdens dat de auto op een witte dwarslijn stilstaat zal deze direct vertrekken en zijn route verder zetten. Om te controleren of de knop is ingedrukt of niet laten we dit nog eens printen in onze seriele monitor.

  if((analogRead(ir1) >= 3800) && (analogRead(ir2) >= 3800) && (analogRead(ir3) >= 3800) && (analogRead(ir4) >= 3800) && (analogRead(ir5) >= 3800))

  {

    Stop();

    delay(2000);

      forward();

      delay(500);

    for(int i = 0;i <= 5000;){

      i++;

      delay(1);

      if(digitalRead(button) == HIGH){

        Serial.println("button pressed");

          break;

      }

    }

    forward();

    delay(100);

  }

Om recht vooruit te rijden maken we een functie aan die ik forward heb genoemd. Doormiddel van digitalWrite en de juiste motorpin laten we beide wielen in de zelfde richting draaien. Om de groene led te laten branden en de rode uit te schakelen zetten we onder in de functie nog een digitalWrite met high en low.

void forward(){  //forward

  digitalWrite(motor1Pin1, HIGH); //Right Motor forward Pin

  digitalWrite(motor1Pin2, LOW);  //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW);  //Left Motor forward Pin

  digitalWrite(motor2Pin2, HIGH); //Left Motor backward Pin

  digitalWrite(LEDROOD, LOW);

  digitalWrite(LEDGROEN, HIGH);

}

om de auto naar rechts te laten draaien maken we een functie met de naam turnRight. Deze bestaat ook uit digitalWrite’s en we zorgen ervoor dat 1 wiel stopt met draaien of achteruit draait zodat de auto draait. Ook hier zetten we de led’s erbij zodat deze correct blijven branden.

void turnRight(){ //turnRight

  digitalWrite(motor1Pin1, LOW);  //Right Motor forward Pin

  digitalWrite(motor1Pin2, HIGH); //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW);  //Left Motor forward Pin

  digitalWrite(motor2Pin2, HIGH); //Left Motor backward Pin

  digitalWrite(LEDROOD, LOW);

  digitalWrite(LEDGROEN, HIGH);

}

Om de auto naar links te laten draaien doen we exact hetzelfde als naar rechts maar dan andersom waar de pins high waren om naar rechts te draaien worden deze nu low en degene die low waren worden high. Ook hier weer de leds zodat de groene led blijft branden wanneer de auto rijd.

void turnLeft(){ //turnLeft

  digitalWrite(motor1Pin1, HIGH); //Right Motor forward Pin

  digitalWrite(motor1Pin2, LOW);  //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW); //Left Motor forward Pin

  digitalWrite(motor2Pin2, HIGH);  //Left Motor backward Pin

  digitalWrite(LEDROOD, LOW);

  digitalWrite(LEDGROEN, HIGH);

}

Om de auto te laten stoppen hoeven we in deze functie alleen maar alle pinnen van onze motordriver laag te zetten en de rode led laten aangaan en de groene uit.

void Stop(){ //stop

  digitalWrite(motor1Pin1, LOW); //Right Motor forward Pin

  digitalWrite(motor1Pin2, LOW); //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW); //Left Motor forward Pin

  digitalWrite(motor2Pin2, LOW); //Left Motor backward Pin

  digitalWrite(LEDROOD, HIGH);

  digitalWrite(LEDGROEN, LOW);

}

Om nog eens te dubbel checken staat hieronder nog eens de volledige code.

// MAXIM RATIAU

// Motor A

int motor1Pin1 = 21;

int motor1Pin2 = 22;

int enable1Pin = 12;

// Motor B

int motor2Pin1 = 18;

int motor2Pin2 = 19;

int enable2Pin = 2;

// IR-sensor

int ir5 = 32;

int ir4 = 33;

int ir3 = 15;

int ir2 = 27;

int ir1 = 14;

// LED'S

int LEDROOD = 23;

int LEDGROEN = 13;

// drukknop

int button = 5;

int isgestopt;

unsigned long startMillis;  //some global variables available anywhere in the program

unsigned long currentMillis;

const unsigned long period = 1000;  //the value is a number of milliseconds

// ultrasone sensor

int trigPin = 26;

int echoPin = 25;

float usread;

float cm;

// ultrasone sensor array

char usarray[5];

void forward();

// Setting PWM properties

const int freq = 30000;

const int pwmChannel = 0;

const int resolution = 8;

int dutyCycle = 200;

//MQTT

#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient espClient;

PubSubClient client(espClient);

const char\* ssid = "embed"; // je eigen SSID

const char\* password = "weareincontrol"; // je WiFi wachtwoord

const char\* mqtt\_server = "192.168.1.90"; // je IP address van je odroid

const int mqtt\_port = 1883;

const char\* mqtt\_username = "maxim"; // username van je influxDB

const char\* mqtt\_password = "maxim"; // wachtwoord van je influxDB

void setup() {

  // motorpinnen als outputs:

  pinMode(motor1Pin1, OUTPUT); // draairichting regelen motor A

  pinMode(motor1Pin2, OUTPUT); // draairichting regelen motor A

  pinMode(enable1Pin, OUTPUT); // motor enabelen en de snelheid regelen

  pinMode(motor2Pin1, OUTPUT); // draairichting regelen motor B

  pinMode(motor2Pin2, OUTPUT); // draairichting regelen motor B

  pinMode(enable2Pin, OUTPUT); // motor enabelen en de snelheid regelen

  // ir sensor pinnen als inputs:

  pinMode(ir5, INPUT);

  pinMode(ir4, INPUT);

  pinMode(ir3, INPUT);

  pinMode(ir2, INPUT);

  pinMode(ir1, INPUT);

  //LED'S

  pinMode(LEDROOD, OUTPUT);

  pinMode(LEDGROEN, OUTPUT);

  // button

  pinMode(button, INPUT);

  // ultrasone sensor

  pinMode(trigPin, OUTPUT); // transmitter

  pinMode(echoPin, INPUT); // ontvanger

  // motor driver

  digitalWrite(enable1Pin, HIGH); // motor A enabelen

  digitalWrite(enable2Pin, HIGH); // motor B enabelen

  analogWrite(enable1Pin, 150); // snelheid regelen motor A

  analogWrite(enable2Pin, 150); // snelheid regelen motor B

  Serial.begin(115200);

  //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);

  delay(1000);

}

void loop() {

  client.loop();

  while (!client.connected())

  {

    if (client.connect("ESP32Client", mqtt\_username, mqtt\_password))

    {

     Serial.println("connected to MQTT");

     Serial.println("-----------------------");

    }

    else

    {

      Serial.print("failed with state ");

      Serial.println(client.state());

      Serial.println("-----------------------");

      delay(2000);

    }

  }

// ultrasone sensor

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  usread = pulseIn(echoPin, HIGH);

  cm = (usread \* 0.0343) / 2;

  dtostrf(cm, 2, 2, usarray);

  Serial.println(usarray);

  delay(500);

// publish

  client.publish("home/auto/ultrasone", usarray); // publishen naar de odroid

// line following

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) >= 3800) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    forward();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) >= 3800) && (analogRead(ir3) <= 2600) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) >= 3800) && (analogRead(ir2) <= 2600) && (analogRead(ir3) <= 2600) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) <= 2600) && (analogRead(ir4) >= 3800) && (analogRead(ir5) <= 2600))

  {

    turnLeft();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) <= 2600) && (analogRead(ir4) <= 2600) && (analogRead(ir5) >= 3800))

  {

    turnLeft();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) >= 3800) && (analogRead(ir4) >= 3800) && (analogRead(ir5) <= 2600))

  {

    turnLeft();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) >= 3800) && (analogRead(ir3) >= 3800) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) >= 3800) && (analogRead(ir2) >= 3800) && (analogRead(ir3) >= 3800) && (analogRead(ir4) <= 2600) && (analogRead(ir5) <= 2600))

  {

    turnRight();

  }

  if((analogRead(ir1) <= 2600) && (analogRead(ir2) <= 2600) && (analogRead(ir3) >= 3800) && (analogRead(ir4) >= 3800) && (analogRead(ir5) >= 3800))

  {

    turnLeft();

  }

  if((analogRead(ir1) >= 3800) && (analogRead(ir2) >= 3800) && (analogRead(ir3) >= 3800) && (analogRead(ir4) >= 3800) && (analogRead(ir5) >= 3800))

  {

    Stop();

    delay(2000);

      forward();

      delay(500);

    for(int i = 0;i <= 5000;){

      i++;

      delay(1);

      if(digitalRead(button) == HIGH){

        Serial.println("button pressed");

          break;

      }

    }

    forward();

    delay(100);

  }

  if((analogRead(ir1) <= 3800) && (analogRead(ir2) <= 3800) && (analogRead(ir3) <= 3800) && (analogRead(ir4) <= 3800) && (analogRead(ir5) <= 3800))

  {

    Stop();

  }

  if(cm < 15)

  {

    Stop();

  }

}

void forward(){  //forward

  digitalWrite(motor1Pin1, HIGH); //Right Motor forward Pin

  digitalWrite(motor1Pin2, LOW);  //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW);  //Left Motor forward Pin

  digitalWrite(motor2Pin2, HIGH); //Left Motor backward Pin

  digitalWrite(LEDROOD, LOW);

  digitalWrite(LEDGROEN, HIGH);

}

void turnRight(){ //turnRight

  digitalWrite(motor1Pin1, LOW);  //Right Motor forward Pin

  digitalWrite(motor1Pin2, HIGH); //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW);  //Left Motor forward Pin

  digitalWrite(motor2Pin2, HIGH); //Left Motor backward Pin

  digitalWrite(LEDROOD, LOW);

  digitalWrite(LEDGROEN, HIGH);

}

void turnLeft(){ //turnLeft

  digitalWrite(motor1Pin1, HIGH); //Right Motor forward Pin

  digitalWrite(motor1Pin2, LOW);  //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW); //Left Motor forward Pin

  digitalWrite(motor2Pin2, HIGH);  //Left Motor backward Pin

  digitalWrite(LEDROOD, LOW);

  digitalWrite(LEDGROEN, HIGH);

}

void Stop(){ //stop

  digitalWrite(motor1Pin1, LOW); //Right Motor forward Pin

  digitalWrite(motor1Pin2, LOW); //Right Motor backward Pin

  digitalWrite(motor2Pin1, LOW); //Left Motor forward Pin

  digitalWrite(motor2Pin2, LOW); //Left Motor backward Pin

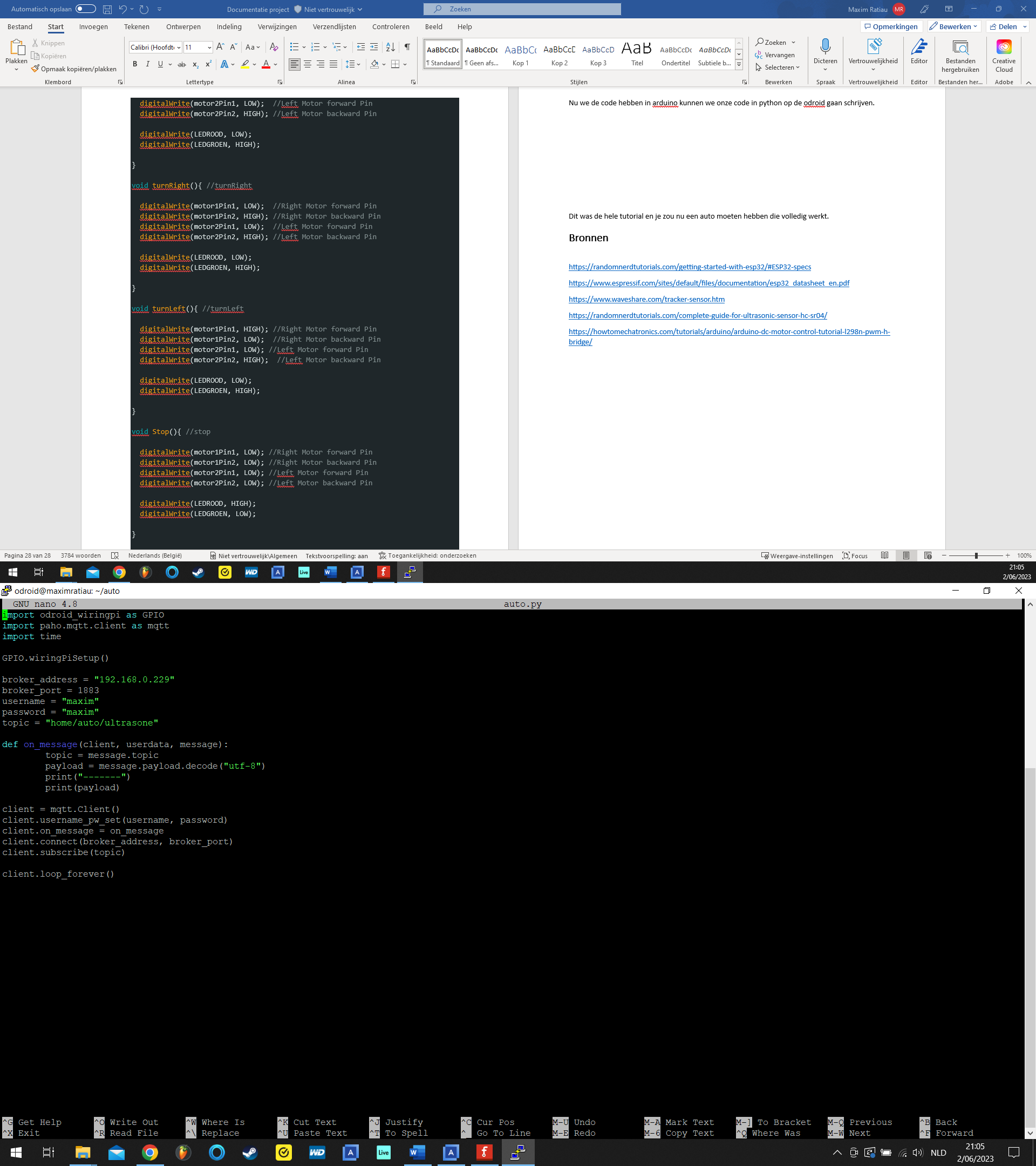
  digitalWrite(LEDROOD, HIGH);

  digitalWrite(LEDGROEN, LOW);

}

// MAXIM RATIAU

Nu we de code hebben in arduino kunnen we onze code in python op de odroid gaan schrijven. Als eerste importeer je paho.mgtt.client as mqtt om de mqtt op te zetten. Ook importeer je time voor eventuele delays. Het broker adres is je ip. Broker port standaard 1883. Daarna voeg je terug je username en wavhtwoord van influxDB in. En tenslotte je topic die je hebt gegeven in je arduino code. Na dit stukje gaan we onze payload ontcijferen zodat deze in de database geplaatst kan worden



Nu deze code is moet je nog een database aanmaken. Dit doe je door create database en use database

Run nu python bridge.py en typ in je adresbalk van je webbrowser het ip adres van je odroid dubbelpunt 3000. Log nu in op grafana en maak een dashboard aan.

Dit was de hele tutorial en je zou nu een auto moeten hebben die volledig werkt.

# **Bronnen**

<https://randomnerdtutorials.com/getting-started-with-esp32/#ESP32-specs>

<https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>

<https://www.waveshare.com/tracker-sensor.htm>

<https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>

<https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/>