Documentatie

De robot

Embedded devices

Rik Daems 2ITF
Cisse T'Syen 2ITF
Tobias Geyskens 2ITF
Seppe Vissers 2ITF

Academiejaar 2020-2021

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

1 VOORWOORD

Dit is een project voor het vak Embedded devices, binnen het tweede jaar IT-Factory in de keuzerichting IoT aan Thomas More Geel. Voor dit vak moesten wij een project maken gebaseerd op 1 van de voorgestelde scenario's. Aan dit project hebben we kunnen werken tijdens het tweede semester van het schooljaar 2020-2021.

Het is een onderwerp dat ons heeft uitgedaagd, wat ons dus extra motivatie gaf om dit project succesvol af te werken.

Graag zouden we Quinten Desmyter bedanken voor ons de kans te geven om dit project te kunnen verwezenlijken, tips te geven en te helpen vervaardigen van dit project. Ook willen Michiel Verboven bedanken voor het leveren van onze componenten.

2 INHOUDSTAFEL

3 INLEIDING

Deze bundel is de schriftelijke weergave van ons project. Het hoofddoel van ons project is om een robot te ontwikkelen die in zwaar getroffen gebieden stralingsniveaus kan meten, op afstand bestuurbaar is, obstakels zoals trappen op en neer kan rijden en videobeelden kan doorsturen naar een First Person View bril. We hebben deze opdracht verdeelt in aparte delen.

Eerst gaan we het hebben over de 2 versies van de robot. Vervolgens bespreken we het frame dat we hebben opgebouwd. Dan verdiepen we ons in de aansturing van de robot en over het brein van de robot genaamd de ESP32. Hierna lichten we de geigerteller toe die de straling meet en doorstuurt naar de ESP32. Vervolgens de FPV-bril en de camera die gebruikt wordt als de ogen van de robot. Dan bespreken we de extra componenten en de batterij die we gebruiken. Tenslotte gaan we het hebben over de PCB, de componentenlijst en maken we nog een besluit.

4 ROBOT VERSIE 1

Voor onze eerste versie van de robot dachten we echt groot. Het oorspronkelijke plan was om een robot te maken die op trappen kan rijden en elk obstakel zou kunnen overwinnen. We zouden deze robot dan ook met onze smartphone besturen. Maar we hadden misschien wat overdreven waardoor deze versie van onze robot niet is kunnen doorgaan.

Product	Aantal	Prijs/stuk	Totaal
Robot frame:			- €
Motor rechts (right angle) DC24V 80W 180 rpm	1	53,69 €	53,69 €
Motor links (left angle) DC24V 80W 180 rpm	1	57,99 €	57,99 €
Motor controller Sabertooth Dual 25A 6V-24V	1	92,40 €	92,40 €
V-snaar	2	25,00 €	50,00 €
V-polie (wielen)	8	19,72 €	157,76 €
Mtb band	2	9,50 €	19,00 €
Bearings (17mm)	8	4,88 €	39,04 €
Batterij	2	45,99 €	91,98 €
Fpv Bril (camera + monitor) PAL	1	107,69 €	107,69 €
Alumunium + metaal plaatwerk	1	300,00 €	300,00 €
diversen bouten	1	30,00 €	30,00 €
			- €
Sensoren/modules:			- €
Adafruit BME680	1	22,95 €	22,95 €
LoRa Radio Module - 868MHz	1	47,95 €	47,95 €
GP2Y0A710K0F Sharp, Reflective Sensor	1	21,04 €	21,04 €
			- €
Extra:			- €
Full range 4 ohm speaker	1	8,99 €	8,99 €
Lamp	1	13,31 €	13,31 €
			- €

Totaal incl. Btw
1.113,79 €

Btw
233,90 €

Totaal excl. Btw
879,89 €

Figuur 1: robot versie 1

5 ROBOT VERSIE 2

Voor de tweede versie van de robot zijn we wat kleinschaliger gaan denken en zoeken. We hadden online een mooie robot frame gevonden inclusief motors, en hebben hierop verder gebouwd. Om de motors aan te sturen hadden we een L298N aangekocht. In een zoektocht extra features om de robot interessanter te maken hebben we een FPV-camera en bril aangeschaft, alsook een geigerteller om radioactiviteit te meten. De stroomvoorziening voor de robot gingen we doen aan de hand van een LiPo-batterij die we nog hadden liggen.

Met de grote componenten bij de hand zijn we begonnen met het ontwerpen van een PCB om alles op een nette manier te verbinden. Terwijl een teamlid aan de PCB werkte, waren de andere ook druk bezig met het testen van de hardware alsook al het schrijven van de documentatie.

6 FRAME

Voor het frame hebben we met verschillende factoren rekening gehouden namelijk: grote, kwaliteit, mobiliteit en hoe praktisch het is. Het frame ‘Yuewalker Tracked Robot Chassis’ voldeed perfect aan onze eisen.

6.1 Yuewalker Tracked Robot Chassis

Het Yuewalker Tracked Robot Chassis is zeer een wendbaar frame. Het is volledig gemaakt van aluminium met een niet geleidende coating. Op de bovenplaat van de robot zijn veel mogelijkheden voorzien voor componenten te bevestigen als ook een compartiment aan de voorkant van het frame voor een batterij. De tracks zijn gemaakt van stevige plastic met een goed grip. Ook zijn er al twee motoren aanwezig bij het frame wat maakte dat we waar voor ons geld hebben kregen. Later in de documentatie komen we nog terug op de motoren.

Het frame hebben we zelf moeten bouwen omdat we hebben gekregen in een bouwpakket. De was zeer duidelijk en voorzelfsprekend. We hadden dus snel een gebruiksklaar frame.

Figuur 2: componenten frame

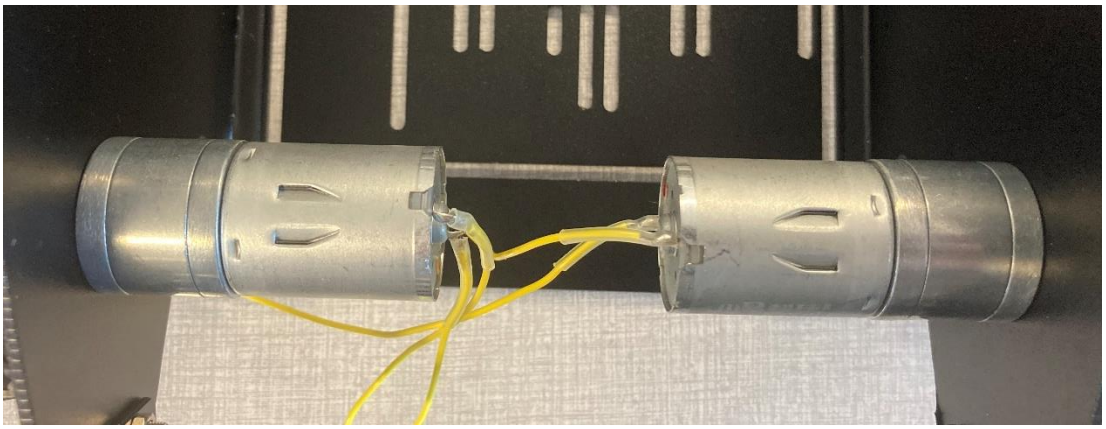
Figuur 3: Frame robot

7 AANSTURING

Voor het aansturen van de robot hebben we verschillende componenten nodig. We hebben twee motoren nodig, een H-brug voor de motoren aan te sturen en twee optocouplers voor het PWM-signaal te kunnen regelen.

7.1 Motoren

Bij het frame zijn twee 12V DC motoren met 160RPM aanwezig. Deze motoren hebben een hoog vermogen en een relatief grote snelheid. Dit in combinatie met de tracks zorgt ervoor dat de robot over verschillende wegdekken kan rijden.



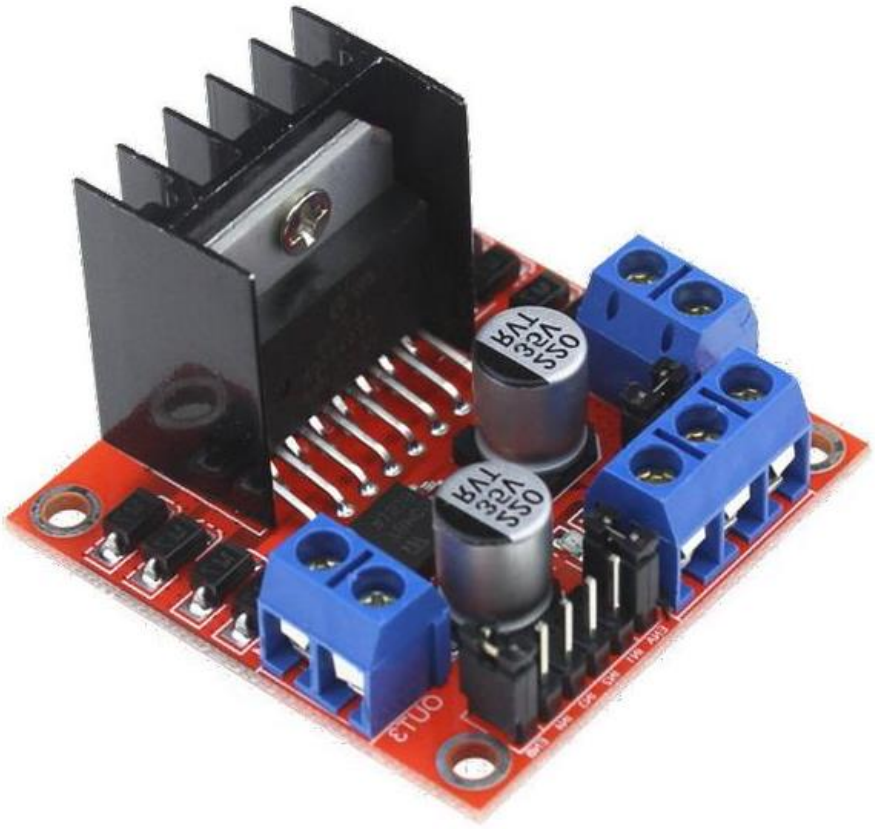
Figuur 4: motoren

7.2 H-brug L298N

De L298N is een motorcontroller met een dubbele H-brug. Door de grote aanwezige koelplaat heeft hij zeer weinig warmteproductie. De maximum stroom die hij zal kunnen leveren is 3A en de maximum spanning die hij zal kunnen verwerken is 46V. Dit zijn uiteraard de piekstroom en piekspanning. We gaan met deze motordriver dus onze twee DC motoren aansturen. Deze zullen dus kunnen worden aangestuurd met een stroom van 2A en een vermogen van 25W.

Specificaties:

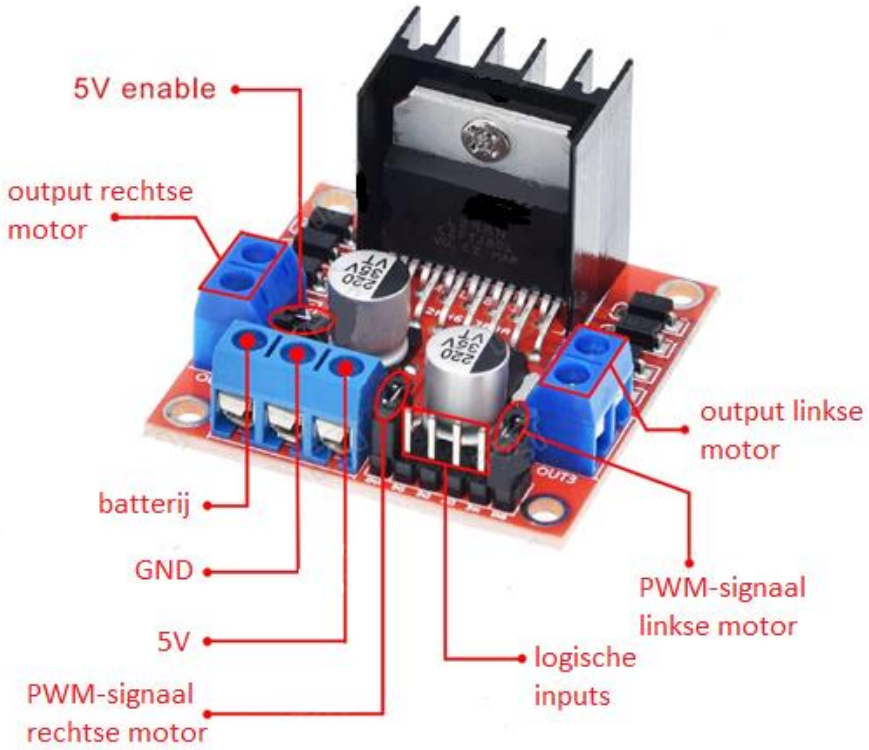
- Driver spanning: 5V-35V
- Digitale spanning: 5V
- Driver stroom: 2A per brug
- Maximum vermogen: 25W
- Piekspanning: 46V
- Piekstroom: 3A



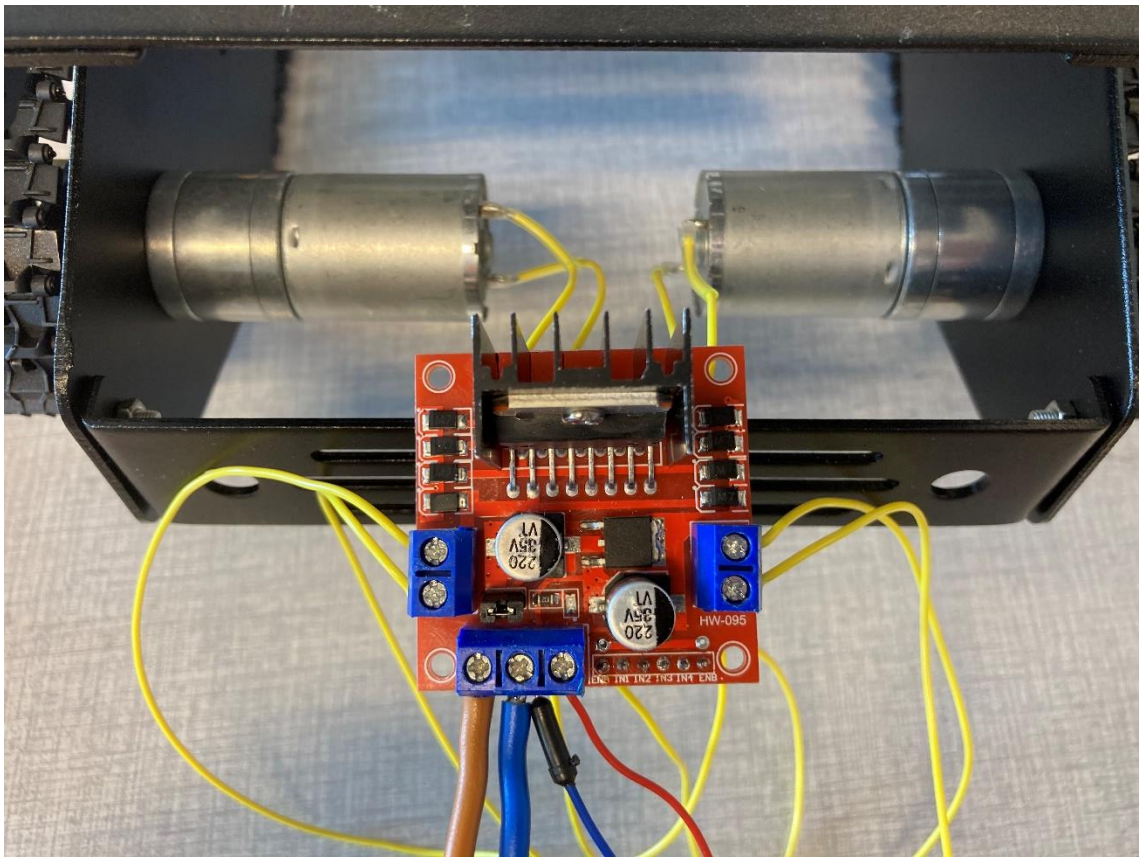
Figuur 5: L298N motordriver

7.2.1 Werking L298N

Op het onderstaande schema is te zien hoe wij de L298N hebben aangesloten. De 2 motoren zullen hierdoor worden aangestuurd. Dat is te zien bij de output van de rechtse en van de linkse motor. Ook zal onze batterij er op worden aangesloten. Er is ook een 5V output aanwezig. Ook is er de mogelijkheid om het PWM-signaal van elke motor apart te regelen. Hiermee kunnen we de snelheid van de motoren dus zelf bepalen. Als laatste hebben we nog de logische inputs. Hiermee kunnen we bepalen in welke richting een motor zal draaien. Elke motor gebruikt hiervoor 2 input pinnen.



Figuur 6: Benamingen aansluitingen L298N



Figuur 7: eigen aansluitingen L298N

7.3 Optocouplers

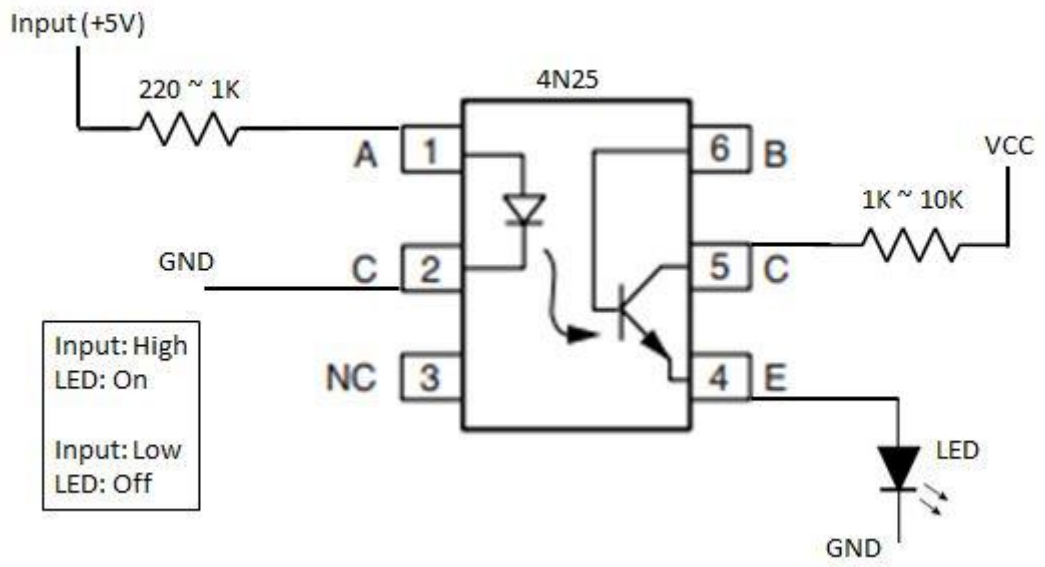
Een optocoupler is een component dat elektrische signalen overdraagt tussen twee elektrisch geïsoleerde kringen doormiddel van licht. Hierdoor kunnen we een circuit met een hogere spanning aansturen met een lagere spanning.

7.3.1 Werking optocouplers 4N25

We gebruiken de optocouplers om de H-brug aan te sturen met PWM-signalen. Dit doen we omdat de ESP32 een 3.3v PWM-signaal geeft, de H-brug kan een 5V PWM-signaal aan. Als we dit 3.3V PWM-signaal rechtstreeks aansluiten op de H-brug gaat deze de motoren nooit op volledige kracht kunnen laten draaien.

De optocoupler bestaat uit 2 diodes: één diode is een LED deze stuurt een licht signaal naar de 2de diode die fungeert als een fotodiode en het lichtsignaal opvangt zodat deze het 2de circuit kan aansturen.

Zoals u kan zien op onderstaande afbeelding heeft de 4N25 optocoupler 6 pinnen. Op pin 1 sluiten we het ingang signaal (3.3V PWM-signaal) aan, hier zetten we een weerstand tussen zodat we geen kortsluiting veroorzaken. Op pin 2 sluiten we de GND aan van het 3.3V circuit, pin 3 is not connected (NC) deze sluiten we nergens aan. Pin 4 sluiten we aan de input voor het PWM-signaal van de H-brug aan. Op pin 5 hangen we 5V met een weerstand zodat we ook hier geen kortsluiting maken, pin 6 gebruiken we niet in onze opstelling.



Figuur 8: optocoupler schema

Figuur 9: optocoupler 4N25

7.4 Programmatie besturing

De code op de ESP32 werkt in grote lijnen als volgt:

- We starten een Wi-Fi netwerk op en hosten een webserver.
- Ook initialiseren we de pinnen waarmee we de l298n aansturen.
- Als er een request naar de webserver wordt gestuurd voor een bestand, sturen we dit op
- Als er een request naar de webserver wordt gestuurd met data, sturen we de juiste motor aan met de ontvangen snelheid.

We beginnen met het invoegen van enkele libraries. Met name:

<Arduino.h> om toegang te krijgen tot de hardware van de ESP32
"WiFi.h" om eenvoudig een Wi-Fi netwerk te kunnen opstarten
"ESPAsyncWebServer.h" om een asynchrone webserver te kunnen hosten
"SPIFFS.h" voor toegang tot het filesysteem.

Vervolgens maken we een aantal #define statements om de gebruikte pinnen te identificeren. De gedefinieerde pinnen zijn:

- speedAPin en speedBPin: dit zijn de pinnen waar een PWM signaal op komt om de snelheid van elke motor te regelen
- in1Pin, in2Pin, in3Pin en in4Pin: deze pinnen worden gebruikt om de richting van de motor in te stellen

Hierna declareren we enkele variabelen voor het genereren van PWM signalen:

- const int freq: de frequentie van het PWM-signaal

- const int pwmChannelA/B: een A en een B kanaal waarop de PWM wordt gegenereerd
- const int resolution: de resolutie van het PWM-signaal
- int currentCycleA/B: de huidige duty cycle van elke PWM kanaal

Dan volgt nog een enkele variabele genaamd lastMessageMillies die bijhoud wanneer het laatst ontvangen bericht is binnen gekomen.

Ten slotte nog een paar laatste variabelen voor het Wi-Fi netwerk en de webserver:

const char *ssid: de naam van het Wi-Fi -netwerk
const char* password: het wachtwoord van het Wi-Fi netwerk
AsyncWebServer server(80): het object van de webserver, gestart op poort 80

We zijn aangekomen in de "setup()" functie. Deze functie loopt 1 keer als de ESP32 opstart of reset. Hierin beginnen we eerst met het initialiseren van Serial met een baud rate van 115200. De seriële verbinding is niet noodzakelijk, maar maakt het debuggen van de code een pak eenvoudiger aangezien we hiermee data kunnen sturen van de ESP32 naar een verbonden computer. Verder in de beschrijving van de code worden de "Serial.print" en "Serial.println" lijnen overgeslagen aangezien deze geen meerwaarde geven.

Na de seriele poort te hebben gestart, configureren we de 2 PWM-kanalen met behulp van "ledCSetup" en verbinden het PWM-kanaal aan een pin met "ledCAttachPin".

Na de PWM pinnen is het de beurt aan de richting pinnen, met namen in1Pin tot in4Pin. Deze worden allemaal via "pinMode" als output pinnen gezet.

Vervolgens starten we het SPIFFS file systeem op door middel van SPIFFS.begin(true). Deze functie returned 1 als het opstarten gelukt is en 0 als dit niet gelukt is. Als het opstarten van SPIFFS mislukt stoppen we met de rest van de setup door te returnen.

hierna starten we een Wi-Fi access point op met het gegeven SSID via "WiFi.softAP". We kunnen eventueel een wachtwoord meegeven met de functie, maar hebben wij niet gedaan om sneller te kunnen verbinden.

Ook vragen we het IP adres van de access point op via "WiFi.softAPIP" en printen dit naar de seriële lijn.

We declareren ook de callback functies voor de webserver, met andere woorden: wat moet er gebeuren als er een request naar deze url wordt gestuurd?

- Voor de root "/" sturen we een html pagina terug genaamd motorPage.html die we uit het SPIFFS file-systeem ophalen.
- Voor de requests naar "/bootstrap.min.css", "/bootstrap.min.js" en "/jquery.min.js" doen we hetzelfde als met de root pagina: Haal het juiste bestand op uit SPIFFS en stuur deze door. We moeten zelf bootstrap en Jquery hosten aangezien de ESP32 geen toegang heeft tot het internet en we does geen CDN kunnen bereiken.

- Ten slotte komen we in de callback voor /left en /right. Hierna wordt enkel de right beschreven aangezien de left identiek dezelfde werking heeft en enkel andere pinnen aanstuurt.

In de callback slagen we eerst de huidige millies op om bij te houden wanneer het laatste bericht is ontvangen.

Hierna slagen we het aantal parameters in de request op en loopen we over elke parameter. We halen de parameter uit de request uit en slagen dit op in "p". Uit "p" halen we dan de waarde en vormen dit om naar een integer genaamd speed. Afhankelijk van de waarde van speed kunnen er een paar dingen gebeuren:

- Speed kleiner dan 0, wat wilt zeggen dat deze motor achteruit moet draaien.
  We stellen eerst de duty cycle gelijk aan het -speed, aangezien de duty cycle niet negatief kan zijn. Vervolgens sturen we in3Pin laag en in4Pin hoog, om de tegen de [motor bestuur ding] te zeggen in welke richting de motor moet draaien. Vervolgens passen we duty cycle toe op het PWM-kanaal. Ten slotte sturen we een reply terug met response code 200 om de browser te laten weten dat we het pakket goed hebben ontvangen.
- Speed groter dan 0, wat wilt zeggen dat deze motor vooruit moet draaien.
  We stellen weer de duty cycle in, maar deze keer zonder te inverteren.
  We stellen in3Pin hoog en in4Pin laag om vooruit te rijden.
  We passen weer het PWM-signaal toe en antwoorden weer met een 200 code.
- in elk ander geval (dat enkel kan zijn dat speed gelijk aan 0) wat wilt zeggen dat deze motor moet stoppen.
  We stellen de duty cycle in, en schrijven zetten zowel inPin3 als inPin4 laag en passen de duty cycle toe en sturen een 200 code. Ditzelfde doen we dan ook voor de "/left" callback, maar dan met andere pinnen.

Dan ten slotte hebben we nog de loop-functie. Hierin kijken we of het laatste bericht langer geleden is dan een bepaalde threshold, stoppen we alle motoren door elke in*Pin op laag te zetten en beide PWM-kanalen op 0 te zetten.

```
#include <Arduino.h>
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
#include "SPIFFS.h"

#define speedAPin 14
#define speedBPin 15
#define in1Pin 22
#define in2Pin 23
#define in3Pin 19
#define in4Pin 5
```

```
const int freq = 5000;
const int pwmChannelA = 0;
const int pwmChannelB = 1;
const int resolution = 8;
int currentCycleA = 0;
int currentCycleB = 0;

int lastMessageMillies = 0;
// Set your access point network credentials
const char *ssid = "ESP32-WallE";
//const char* password = "wally";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

void setup()
{
  Serial.begin(115200);
  ledcSetup(pwmChannelA, freq, resolution);
  ledcAttachPin(speedAPin, pwmChannelA);
  ledcSetup(pwmChannelB, freq, resolution);
  ledcAttachPin(speedBPin, pwmChannelB);

  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(in3Pin, OUTPUT);
  pinMode(in4Pin, OUTPUT);

  // Initialize SPIFFS
  if (!SPIFFS.begin(true))
  {
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }

  // Setting the ESP as an access point
  Serial.print("Setting AP (Access Point)…");
  // Remove the password parameter, if you want the AP (Access Point) to be op
en
  WiFi.softAP(ssid);

  IPAddress IP = WiFi.softAPIP();

  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
            { request->send(SPIFFS, "/motorPage.html", String(), false); });
```

```
  server.on("/bootstrap.min.css", HTTP_GET, [](AsyncWebServerRequest *request)
            { request-
>send(SPIFFS, "/bootstrap.min.css", String(), false); });

  server.on("/bootstrap.min.js", HTTP_GET, [](AsyncWebServerRequest *request)
            { request->send(SPIFFS, "/bootstrap.min.js", String(), false); });

  server.on("/jquery.min.js", HTTP_GET, [](AsyncWebServerRequest *request)
            { request->send(SPIFFS, "/jquery.min.js", String(), false); });

  server.on("/right", HTTP_GET, [](AsyncWebServerRequest *request)
            {
              lastMessageMillies = millis();
              if (request == nullptr)
              {
                Serial.println("Right request was null");
                return;
              }
              size_t paramsNr = request->params();
              for (int i = 0; i < paramsNr; i++)
              {
                AsyncWebParameter *p = request->getParam(i);

                int speed = p->name().toInt();
                Serial.println(speed);
                if (speed < 0)
                {
                  currentCycleB = -speed;
                  digitalWrite(in3Pin, LOW);
                  digitalWrite(in4Pin, HIGH);
                  ledcWrite(pwmChannelB, currentCycleB);
                  request->send(200, "text/plain", "right");
                }
                else if (speed > 0)
                {
                  currentCycleB = speed;
                  digitalWrite(in3Pin, HIGH);
                  digitalWrite(in4Pin, LOW);
                  ledcWrite(pwmChannelB, currentCycleB);
                  request->send(200, "text/plain", "right");
                }
                else
                {
                  currentCycleB = speed;
                  digitalWrite(in3Pin, LOW);
                  digitalWrite(in4Pin, LOW);
                  ledcWrite(pwmChannelB, currentCycleB);
                  request->send(200, "text/plain", "right");
```

```
    }
  }

  request->send(200, "text/plain", "right");
});

server.on("/left", HTTP_GET, [](AsyncWebServerRequest *request)
{
  lastMessageMillies = millis();
  if (request == nullptr)
  {
    Serial.println("Left request was null");
    return;
  }
  int paramsNr = request->params();
  for (int i = 0; i < paramsNr; i++)
  {
    AsyncWebParameter *p = request->getParam(i);

    int speed = p->name().toInt();
    Serial.println(speed);
    if (speed < 0)
    {
      currentCycleA = -speed;
      digitalWrite(in1Pin, LOW);
      digitalWrite(in2Pin, HIGH);
      ledcWrite(pwmChannelA, currentCycleA);
      request->send(200, "text/plain", "left");
    }
    else if (speed > 0)
    {
      currentCycleA = speed;
      digitalWrite(in1Pin, HIGH);
      digitalWrite(in2Pin, LOW);
      ledcWrite(pwmChannelA, currentCycleA);
      request->send(200, "text/plain", "left");
    }
    else
    {
      currentCycleA = speed;
      digitalWrite(in1Pin, LOW);
      digitalWrite(in2Pin, LOW);
      ledcWrite(pwmChannelA, currentCycleA);
      request->send(200, "text/plain", "left");
    }
  }
  request->send(200, "text/plain", "left");
});
server.begin();
```

```
  Serial.println("HTTP Server Started");
}

void loop()
{
  return;
  if (millis() - lastMessageMillies >= 150)
  {
    Serial.println("resetting");
    lastMessageMillies = millis();
    currentCycleA = 0;
    currentCycleB = 0;
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, LOW);
    digitalWrite(in3Pin, LOW);
    digitalWrite(in4Pin, LOW);

    ledcWrite(pwmChannelA, currentCycleA);
    ledcWrite(pwmChannelB, currentCycleB);
  }
}
```

7.5 Interface besturing

Onze eerste aanpak was om via een mobiele smart-telefoon te verbinden met het Wi-Fi netwerk dat de ESP32 uitstuurt. Als we dan naar het IP-adres van de ESP32 surfte, kregen we een webpagina te zien met de besturing op.

De eerste versie bevatten 4 knoppen, twee voor de linker kant vooruit en achteruit te laten rijden en 2 knoppen voor de rechter kant. Dit was echter geen handige manier van besturen van de robot.

Vervolgens zijn we overgeschakeld naar een interface met 2 sliders, een aan elke zijde van de pagina. Elke slider zou dan 1 van de twee kanten van de robot besturen. Deze konden dan elk met één duim worden bestuurd.

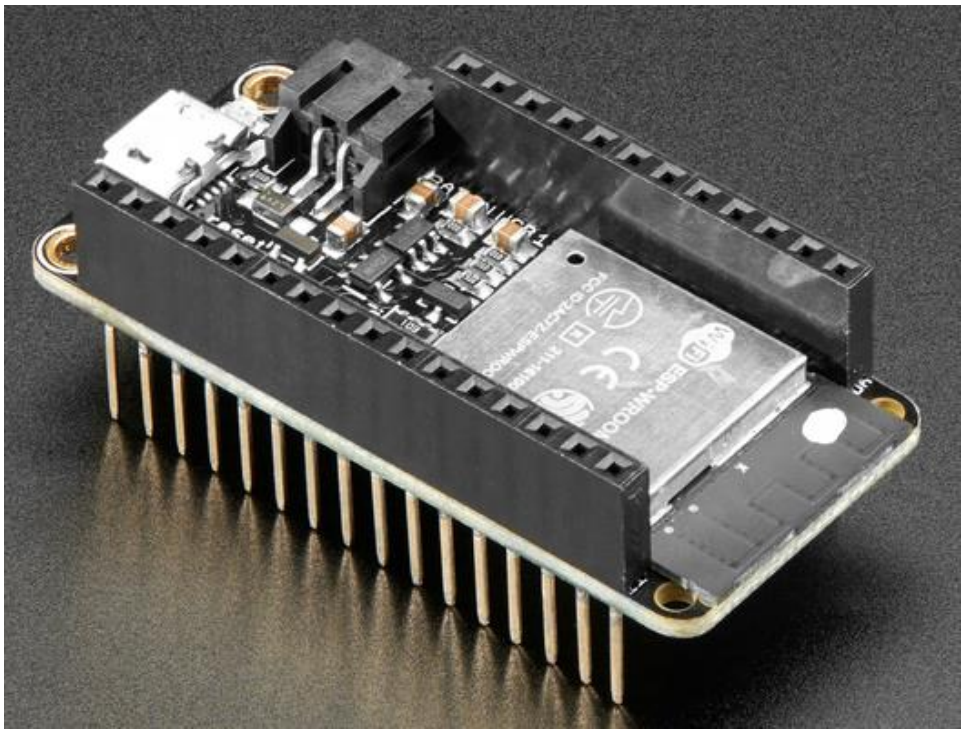
Dit werkte echter ook niet naar toebehoren. Ondanks dat moderne smartphones gemakkelijk meerdere inputs tegelijkertijd kunnen detecteren, is dit niet het geval voor websites. Hierdoor konden we steeds maar een van de 2 kanten van de robot aansturen met als gevolg dat deze in rondjes begon te draaien.

Ten slotte hebben we besloten om de besturing via een computer te laten gebeuren. Men moet nog steeds verbinden met het Wi-Fi-netwerk van de ESP32, maar nu werd de robot bestuurd via de pijltjes toetsen. Maar de optie met de sliders ligt nog altijd open.

Figuur 10: interface besturing

8 ESP32

Het hart van onze robot is natuurlijk een ESP32. Een ESP32 is een microcontroller met verscheidende functies geïntegreerd zoals Wi-Fi en dual-mode Bluetooth. Perfect voor de besturing van ons project dus.



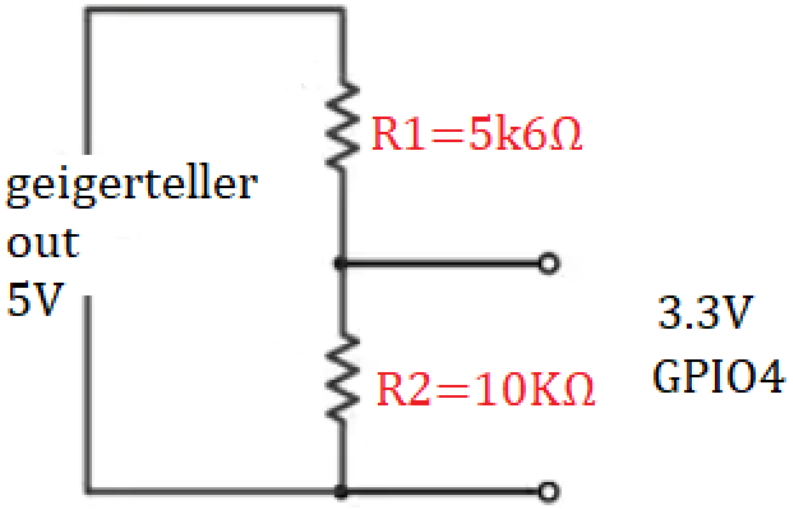
Figuur 11: ESP32 feather

8.1 De ESP32 in detail

De ESP32 is uitgerust met 28 GPIO (general-purpose input/output) pinnen. Deze pinnen kunnen worden gebruikt voor een groot aantal toepassingen. Wij gaan enkele GPIO-pinnen gebruiken. Om te beginnen voor de LEDS. Voor de LEDS aan te sturen gaan wij GPIO33 en GPIO27.

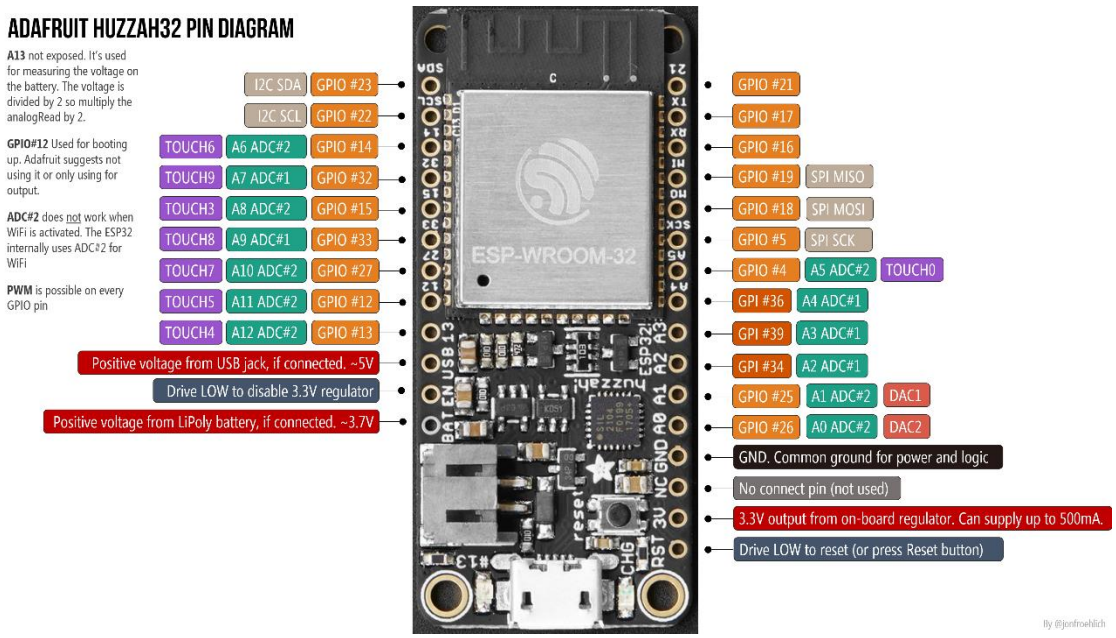
Voor de volgende toepassing gaan we de GPIO-pinnen gebruiken voor de motoren de besturen. Daarvoor is het handig dat er PWM is op elke GPIO-pin. Ook voor de signalen van de motoren zullen we GPIO-pinnen gebruiken. Hiervoor gebruiken we er 4, namelijk GPIO22, GPIO23, GPIO19 en GPIO5. Ook voor de optocouplers zullen op GPIO-pinnen worden aangesloten. Dit zal gebeuren op GPIO14 en GPIO15.

Een aantal GPIO-pinnen kunnen ook worden gebruikt als analoge pin. Dit zorgt ervoor dat we ook de geigerteller kunnen uitlezen. Wij doen dit op GPIO4. We moeten wel opletten met de geigerteller want er mag maar 3.3V op de GPIO-pinnen komen te staan. Dit wil zeggen da we een spanningsdeler gaan moeten bijplaatsen aan de uitgang van de geigerteller. Deze spanningsdeler bestaat uit een 5k6Ω weerstand en een 10kΩ weerstand.



$$V_{out} = V_1 \frac{IR_2}{I(R_1 + R_2)} = \frac{V_1 R_2}{(R_1 + R_2)}$$

Figuur 12: spanningsdeler geigerteller en ESP32

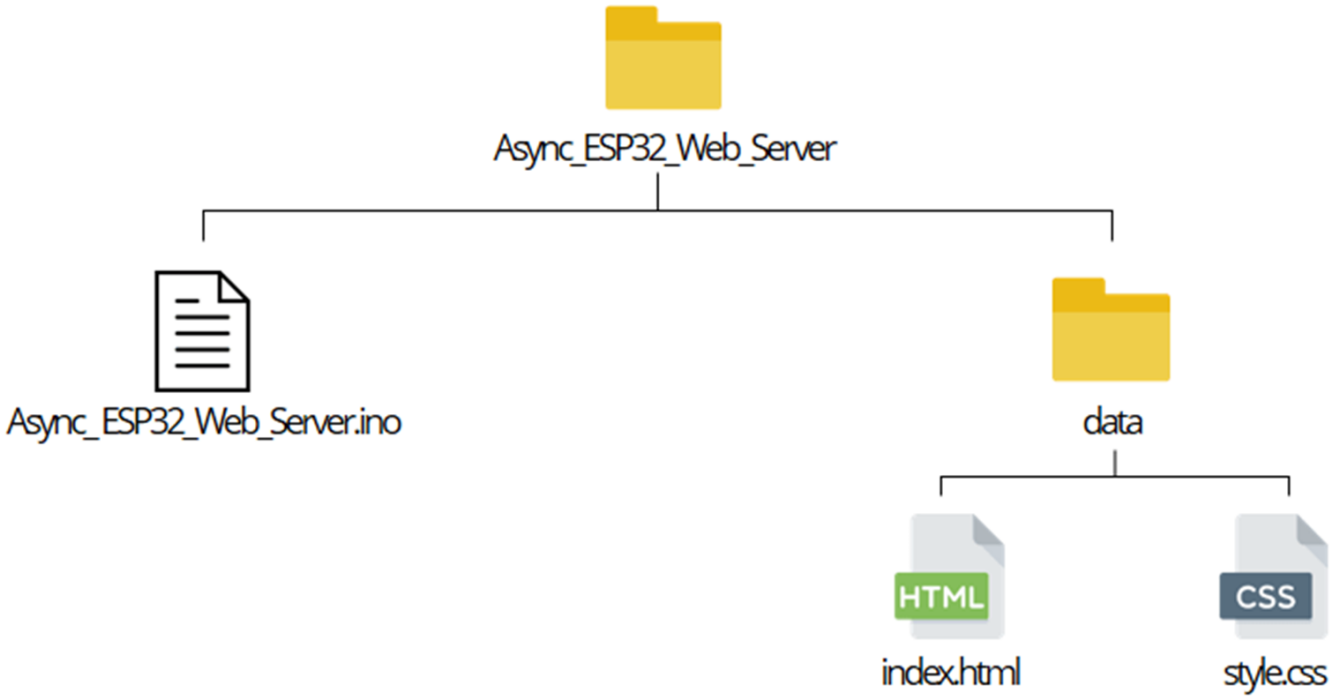


Figuur 13: esp32 pins

8.2 Filesystem

De ESP32 kan ook de inhoud van de bestanden uitlezen dankzij zijn SPIFFS bestandssysteem. Wij gaan dit bestandssysteem gebruiken voor onze webpagina in te lezen. Het SPIFFS

bestandsysteem zal alles opslagen in het flash-geheugen. We zullen dus ons bestand naar het bestandsysteem uploaden wat er dan voor zal zorgen dat we onze webserver kunnen gebruiken.



Figuur 14: SPIFFS-filesystem

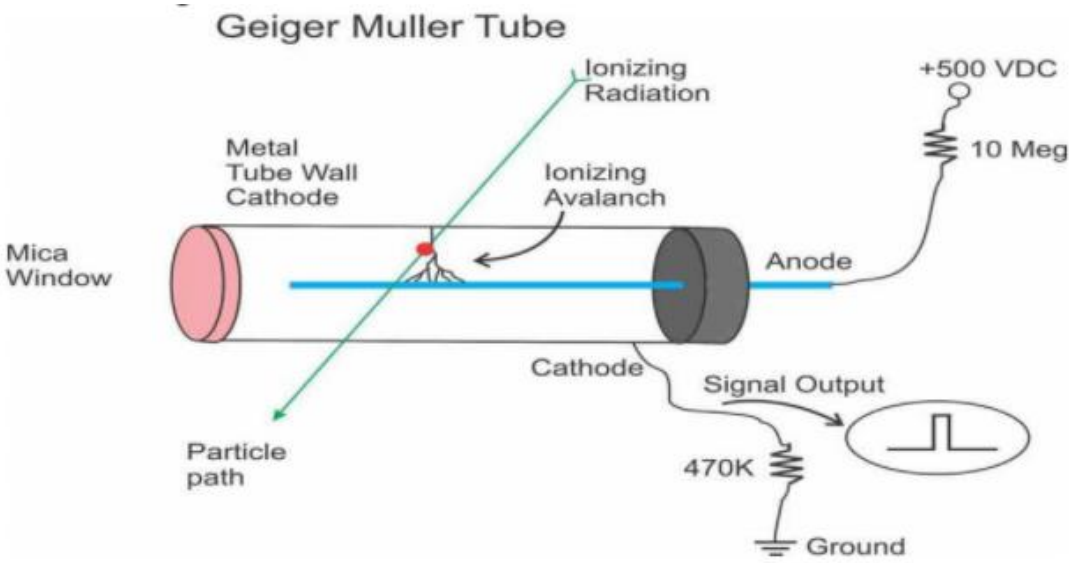
9 GEIGERTELLER

Omdat onze robot kan worden ingezet in gebieden waar er een hoge radiatie is, gebruiken we op onze robot een geigerteller om de radiatie rondom de robot te meten. De geigerteller die wij gekozen hebben zal de Bètastraling en de Gammastraling meten van een gebied en dit laten zien op het toestel waarmee de robot bestuurd word. Dit kan gebruikt worden om waar te nemen of een gebied veilig is om te betreden door reddingswerkers of andere mensen.

Figuur 15: geigerteller

9.1 Werking geigerteller

De Geigerteller bevat een buis (ionisatiekamer) met een draad erdoorheen. Er zal een hoge spanning over de buitenkant van de buis en de draad in het midden worden gezet. In de glazen buis zit een gas Deze spanning is hoog maar net niet hoog genoeg om spontaan elektrische doorslag op te laten treden. Dit spanningsgebied noemt men het Geiger-Müller-gebied. Als de Gamma of Bètastralen door de buis komen zal er ionisatie ontstaan in het gas. Dit wil zeggen dat de gasatomen gesplitst worden in elektronen en positieve ionen. Deze ionen bewegen zich naar de buitenkant van de buis en de elektronen bewegen naar de binnenkant van de buis. Omdat de elektronen naar de draad in de buis bewegen kunnen ze botsen met andere gasatomen. Als zo een botsing plaatsvindt zal het atoom splitsen en de ionen bewegen naar de buitenkant en de elektronen naar de draad in het midden. Dit vormt een lawine-effect en dit zal een puls veroorzaken aan de uitgang van de buis. Deze puls kunnen we dan meten en door de puls knippert de led op het bordje 1keer en hoor je een “tik” van de buzzer die op het bordje staat.



Figuur 16: geigerteller buis

9.2 Programmatie geigerteller

Door de pulsen te tellen die door de geigerteller gevormd zijn krijgen we een beeld van hoeveel straling er aanwezig is in de buurt van de teller. Door een programma te maken dat alle pulsen telt en dit omzet naar het aantal pulsen per minuut krijgen we een beeld van hoeveel radiatie er is.

10 FPV-BRIL MET CAMERA

We willen natuurlijk de mogelijkheid hebben om te zien wat de robot zien indien de robot zich niet meer bevindt in ons gezichtsveld. Hiervoor gaan we gebruik maken van een FPV-bril en een PAL-camera. De camera zal de beelden doorsturen naar de bril waardoor we zullen kunnen zien wat de robot ziet.

10.1 FPV-bril

Zoals eerder aangehaald gaan we kunnen zien wat de robot ziet. Dit doen we dus met een FPV-bril of zoals de afkorting helemaal zegt “first person view goggles”. Onze bril is van het merk Eachine. De bril is uitgerust met een RHCP-antenne en een dBi antenne. Het beeldscherm heeft een hoge resolutie van 800X480. Dit alles zorgt ervoor dat er geen vertraging is in de weergaven van het beeld en ook dat we geen frames gaan verliezen.

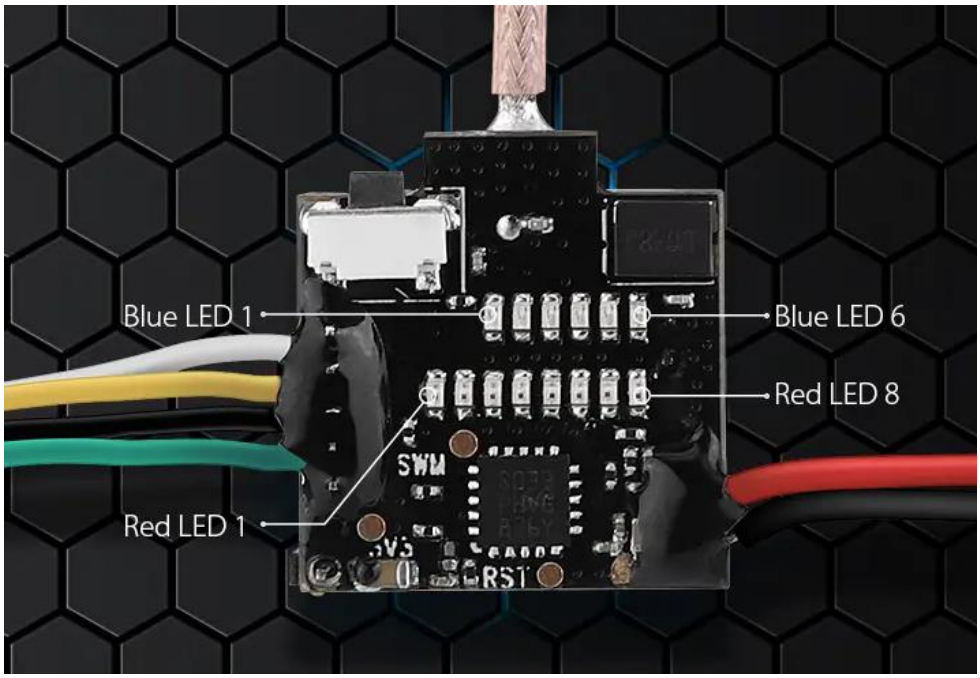
Figuur 17: FPV-bril

10.2 PAL-camera

We hebben natuurlijk ook een camera nodig op onze robot. Dit zal een TX06 PAL-camera zijn van Eachine. De camera is zeer makkelijk te monteren in ons geval. Er zijn meerdere opties mogelijk met de camera zoals smart audio en een video in/output. Wij daarentegen zullen de camera alleen moeten aansluiten aan de 5V.

Figuur 18: TX06 PAL-camera

Op de camera staat ook een drukknop. Met deze knop kunnen we de channels en bands veranderen zodat deze gelijk zijn dan degene waarop de bril is ingesteld. De blauwe led staat voor de bands en de rode led staat voor de channels.



Figuur 19: TX06 PAL-camera leds

10.3 FPV-bril en PAL-camera communicatie

De FPV-Bril en PAL-camera communiceren met elkaar doormiddel van radiofrequentie signalen. Er bestaat analoge- en digitale- FPV-transmissie. Onze Bril en camera gebruiken de analoge FPV-transmissie. Dit is de meest gebruikte variant voor consumenten en zelfs professionele FPV-systemen. Bij analoge FPV-transmissie is geen beeldcompressie of geavanceerde verwerking nodig. Een ander voordeel van analoge FPV-transmissie is dat er bijna geen vertraging zit op het waargenomen beeld door de camera en het beeld dat weergegeven wordt in de bril. Dit is niet noodzakelijk voor onze robot toepassing maar mooi meegenomen en zeer geschikt voor toepassingen zoals drones etc.

Een ander groot voordeel is de geleidelijk toenemende ruis. Dit houdt in dat als de camera bijna buiten bereik is van de signalen die de camera uitstuurt zal de beeldkwaliteit slechter worden en het beeld zal haperen. Dit is een groot voordeel bij onze toepassing omdat de bestuurder van de robot zal weten wanneer de robot bijna buiten bereik is. Dan kan de bestuurder de robot veilig terug binnen bereik brengen.

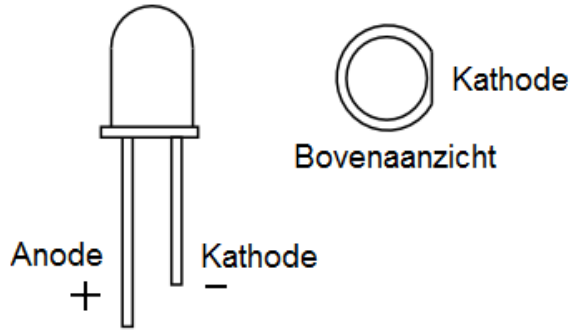
Omdat er obstakels kunnen staan tussen de bril en de camera kan het gebeuren dat er reflectie van het signaal optreedt. Dit zorgt voor ruis omdat de signalen die gereflecteerd worden trager aankomen bij de bril dan de signalen die rechtstreeks de bril bereiken. De antenne die we gebruiken is een paneelantenne van Eachine. Deze antenne zorgt ervoor dat de ruis beperkt wordt. Deze paneelantenne is gemaakt om de gereflecteerde signalen zo veel mogelijk te negeren en enkel de signalen op te vangen die rechtstreeks van de robot komen.

11 EXTRA COMPONENTEN

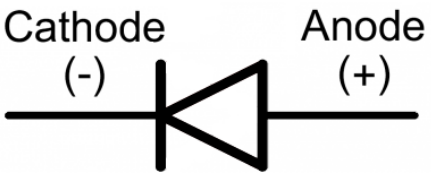
We wouden natuurlijk ook iets klein extra toevoegen aan ons project. We wisten niet goed wat maar we merkten op dat er op ons frame ruimte was voorzien aan de voor- en achterkant voor lampjes toe te voegen dus hebben we besloten om leds als voor- en achterlichten te gebruiken.

11.1 Led

Een led (light emitting diode of in het Nederlands licht uitstralende diode) is een halfgeleidercomponent die licht geeft als er stroom in de doorlaat richting wordt gestuurd. De halfgeleider in een led heeft een positieve en een negatieve kant. In de negatieve zijde zitten er meer negatieve deeltjes (elektronen) dan positieve (protonen) en in de positieve kant omgekeerd. Tussen de negatieve en positieve kant zit een verboden zone. Als we een positieve spanning aan de negatieve kant van de halfgeleider zetten en een negatieve aan de positieve zijde van de halfgeleider. Dan zullen de elektronen naar de positieve kant van de halfgeleider “springen” en dit zal zichtbaar zijn als licht. Door de dode zone kleiner of groter te maken kunnen we de kleur van de led aanpassen.



Figuur 20: anode/kathode led



Figuur 21: Symbool led

12 BATTERIJ

Voor het geheel aan te sturen hebben we natuurlijk een batterij nodig. Deze batterij natuurlijk krachtig zijn. Wij hebben gekozen voor een Lipo-batterij.

12.1 LiPo-batterij

Voor de batterij hebben we gebruikt gemaakt van een 4S LiPo batterij van Tattu. Meer specifiek de Tattu 2300 4s. We hebben voor deze batterij gekozen aangezien we deze nog hadden liggen van een vorig project. Dit soort LiPo-batterijen zijn uitermate geschikt voor ons doelbereik, wegens hun mogelijkheid om veel stroom te leveren. Deze Tattu 2300 4s is zelfs speciaal gemaakt voor het gebruik in rijdende robots, drones en andere op afstand bestuurbare apparaten.

De 4s in de naam betekend dat de batterij uit 4 cellen bestaat. Elke cel heeft een nominale spanning van 3.7V. De 's' in 4s wilt zeggen dat deze cellen in serie staan waardoor we hun spanning kunnen optellen. Dat geeft ons dan 4 x 3.7V = 14.8V.

Uit de batterij komen 2 connectoren, een grotere gele connector, genaamd een XT-60, met draden met een grotere doorsnede. Deze connector wordt gebruikt om de spanning van de batterij aan de robot te verbinden. De andere connector, met 5 dunnere draden, is bedoeld om elke cel van de LiPo gebalanceerd op te laden. Er is één grondkabel en 4 kabels die verbinden met elke individuele cel.

Op het omhulsel van de LiPo wordt de stroom die geleverd kan worden aangegeven met 45C. Dit wil echter niet zeggen dat deze 45 Ampère kan leveren, maar wel hoe we de maximale stroom kunnen berekenen. Om de effectieve stroom capaciteit te kunnen bepalen, moeten we als volgt te werk gaan: We nemen de capaciteit van de batterij (meestal aangeduid in mAh) en rekenen dit om naar Ah. Dit resultaat vermenigvuldigen we vervolgens met 45. Dat geeft ons de uitkomst, wat in ons geval overeenstemt met 2,3 * 45 = 103,5A

Tijdens de loop van het project hebben we helaas geen tijd genoeg om een batterij monitor systeem te implementeren. Dit systeem zou dan de spanning van de batterij meten en bijvoorbeeld de spanning onderbreken als deze te laag valt. Met deze meting zouden we ook de gebruiker of bestuurder kunnen verwittigen als de batterij bijna leeg is.

Een andere leuke en handige feature zou zijn als we een laad-circuit op de robot zelf zouden hebben kunnen plaatsen, waardoor we niet noodzakelijk de batterij van de robot moeten halen om deze op te laten, maar simpel weg een kabel aansluiten op de robot die dan de batterij zou bijladen.

Helaas is op onze laatste werkdag van het project de LiPo stuk gegaan, waardoor we gedwongen waren om over te schakelen naar een Lood-accu.

Figuur 22: LiPo batterij

12.2 Loodaccu

Een loodaccu is het type batterij dat ook in elke auto terug te vinden is. De voordelen van een loodaccu zijn dat ze een langere levensduur hebben en veiliger zijn om te hanteren. Het grootste nadeel van dit soort batterij is dat deze een hoger gewicht hebben in verhouding met hun capaciteit.

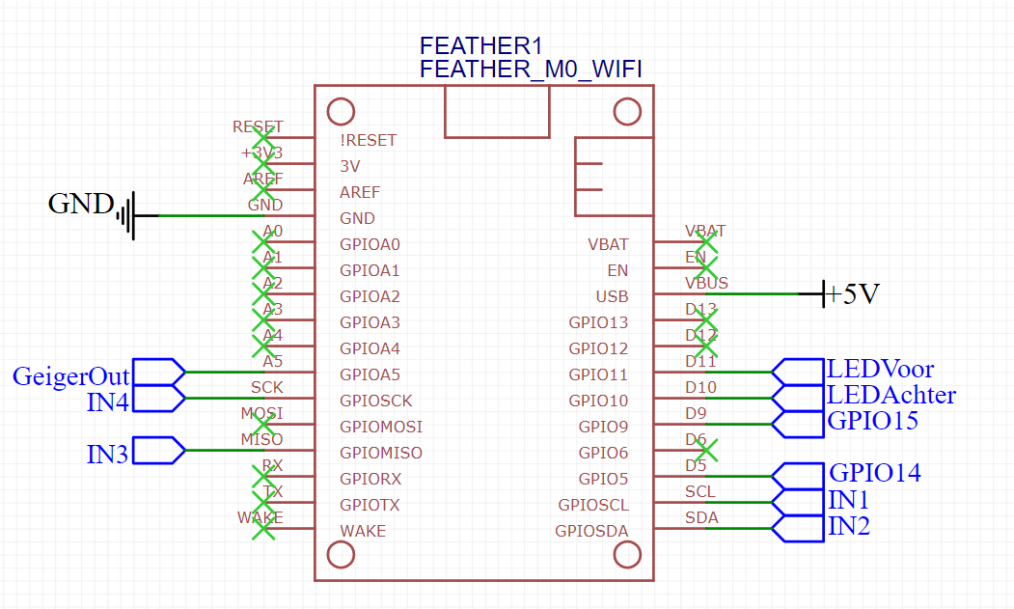
Figuur 23: loodaccu

13 PCB-ONTWERP

We hebben de PCB ontworpen via EasyEDA, dit is een simpel online pcb-ontwerp tool. Deze PCB hebben we laten maken door JLCPCB deze fabrikant is een partner van EasyEDA. Als eerst teken we een aansluitingsschema, hieruit kunnen we dan een PCB-ontwerp generen.

13.1 PCB esp32

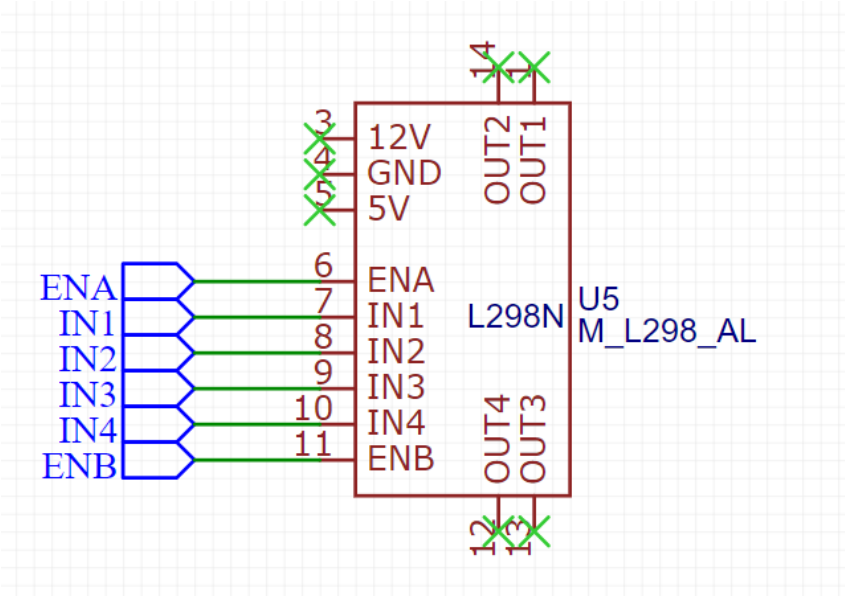
Hier ziet u de schematische voorstelling van de ESP32. De ESP32 is aangesloten op de gezamenlijke ground (GND) en de 5V om deze te voeden. Voor de rest van de connecties gebruiken we net ports, dit doen we zodat het schema overzichtelijk blijft. De overige pinnen die we niet gebruiken zetten we een no connection vlag op, dit doen we zodat er geen onnodige errors tevoorschijn komen.
Vanuit de ESP32 worden alle verdere connecties gelegd voor de verschillende deelsystemen.



Figuur 24: PCB ESP32

13.2 PCB H-brug

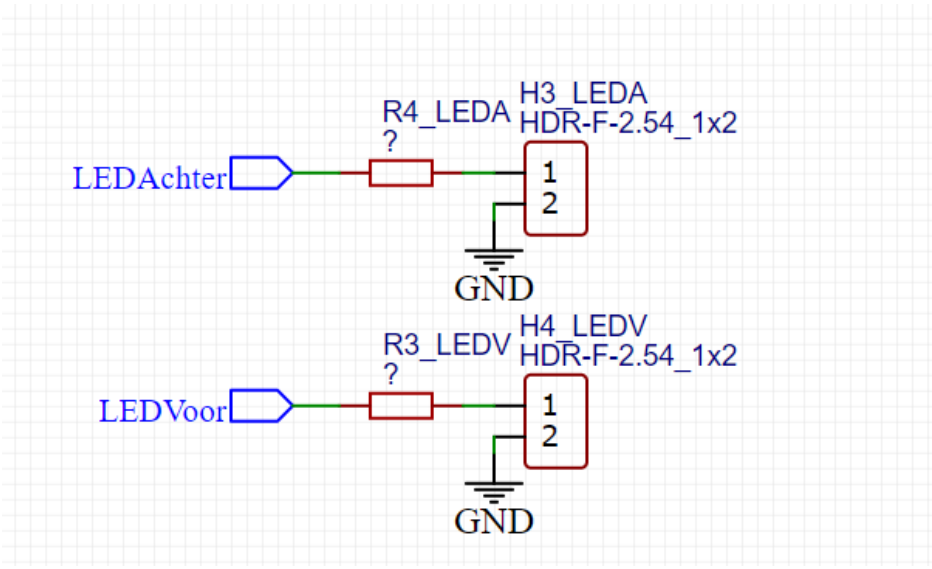
Op de H-brug wordt op pin ENA een PWM-signaal aangesloten om de draaisnelheid van kanaal A te bepalen, dit doen we ook voor kanaal B. De pinnen IN1 en IN2 dienen voor de draairichting te bepalen van kanaal A, dit doen we door op IN1 een 0 of 1 aan te sturen en op IN2 het tegenovergestelde. Pin IN3 en IN4 dienen voor de draairichting van kanaal B aan te sturen deze werken idem als hiervoor.



Figuur 25: PCB H-brug

13.3 PCB LEDS

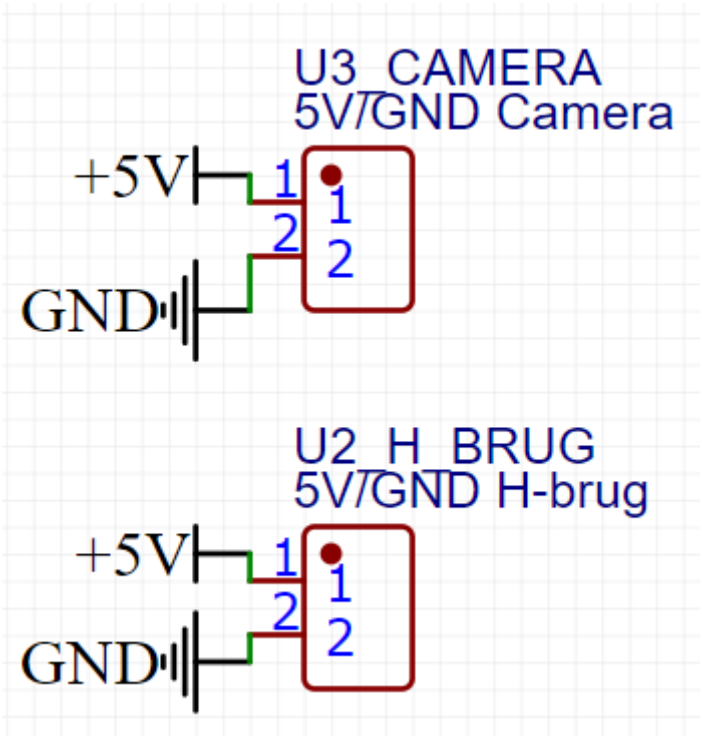
De extra LEDS worden aangesloten met een voorschakel weerstand op externe pinnen zodat we deze gemakkelijk kunnen aansluiten.



Figuur 26: PCB LEDS

13.4 PCB camera en 5V

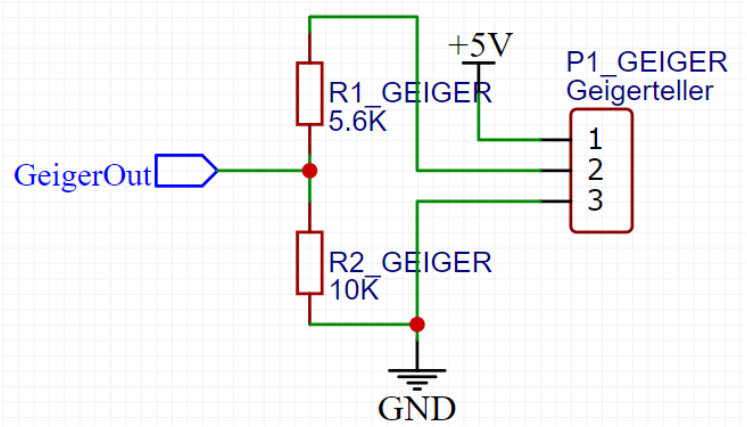
We sluiten de camera voor de FPV-bril aan op een klem zodat we deze kunnen voeden met 5V en GND. De ganse printplaat wordt gevoed met 5V via een klem die we aansluiten aan de spanningsdeler, hierdoor kunnen we van de 12V van de batterij 5V maken.



Figuur 27: PCB camera en 5V

13.5 PCB geigerteller

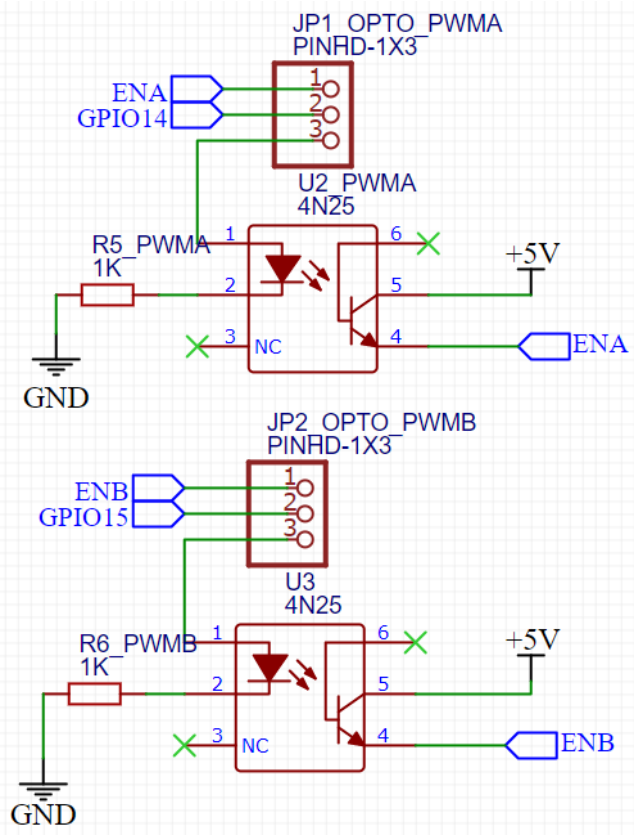
We sluiten de geigerteller op externe pinnen aan, pin 1 is de voeding 5V, pin 2 gaat naar de spanningsdeler en pin 3 gaat naar de GND. We gebruiken hier een spanningsdeler omdat het signaal dat uit de geigerteller komt 5V is en onze ESP32 kan maar 3.3V aan. Door de spanningsdeler wordt deze spanning verdeelt en krijgen we de gewilde spanning op onze ESP32.



Figuur 28: PCB geigerteller

13.6 PCB optocouplers

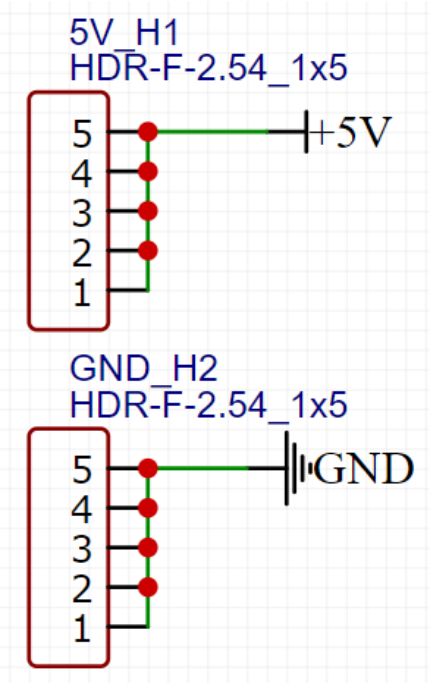
We hebben 3 externe pinnen bij de optocouplers gezet, hier kunnen we kiezen dat we ons signaal dat uit de ESP32 komt rechtstreeks op de H-burg zetten of via de optocouplers zodat deze het signaal kunnen versterken.



Figuur 29: PCB optocouplers

13.7 PCB Externe 5V en GND pinnen

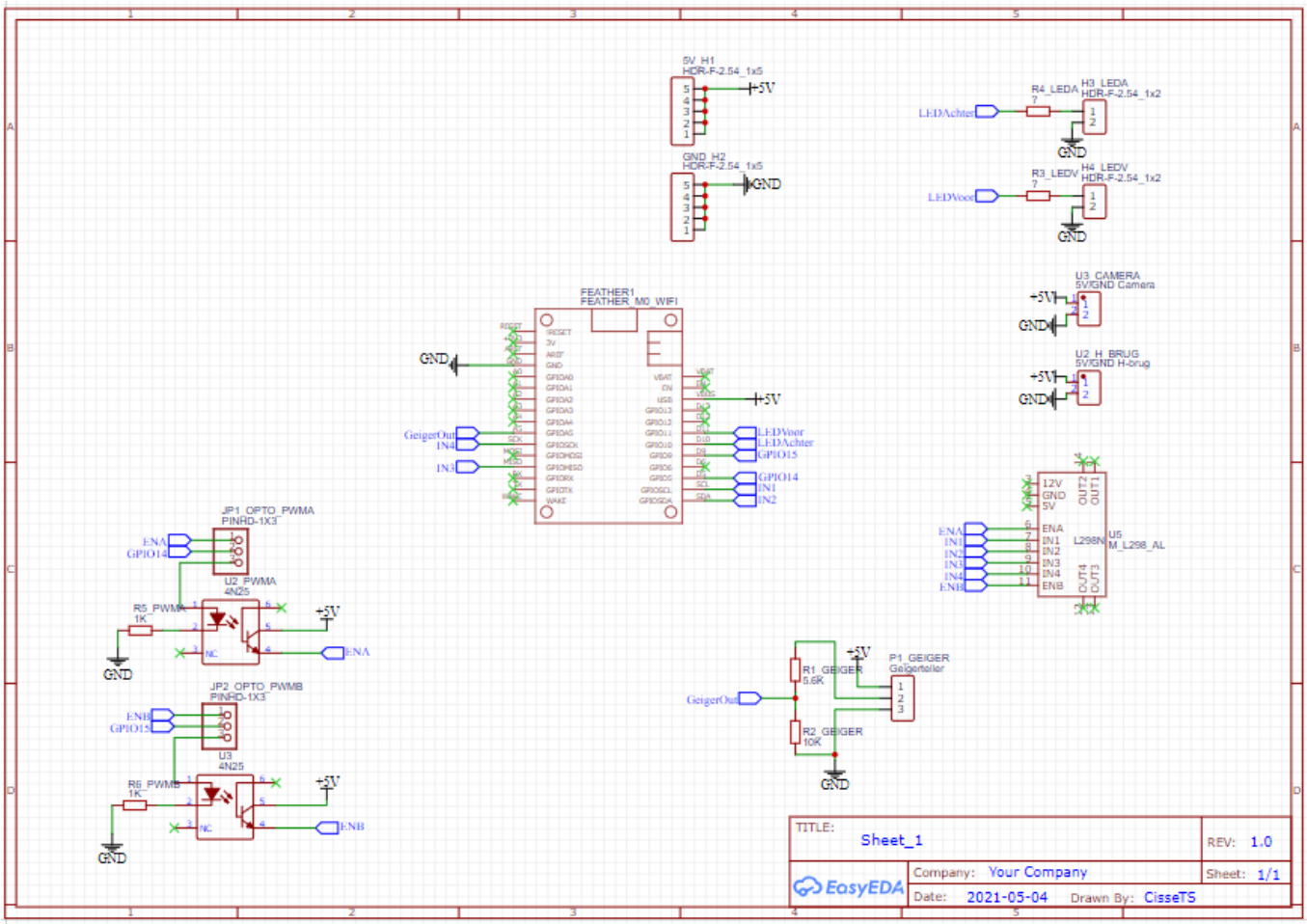
We hebben elk 5 externe pinnen voor de 5V en de GND op ons printplaatje gezet. Deze kunnen altijd van pas komen zouden we de robot nog willen uitbreiden met extra snufjes.



Figuur 30: PCB Externe 5V en GND pinnen

13.8 PCB schema

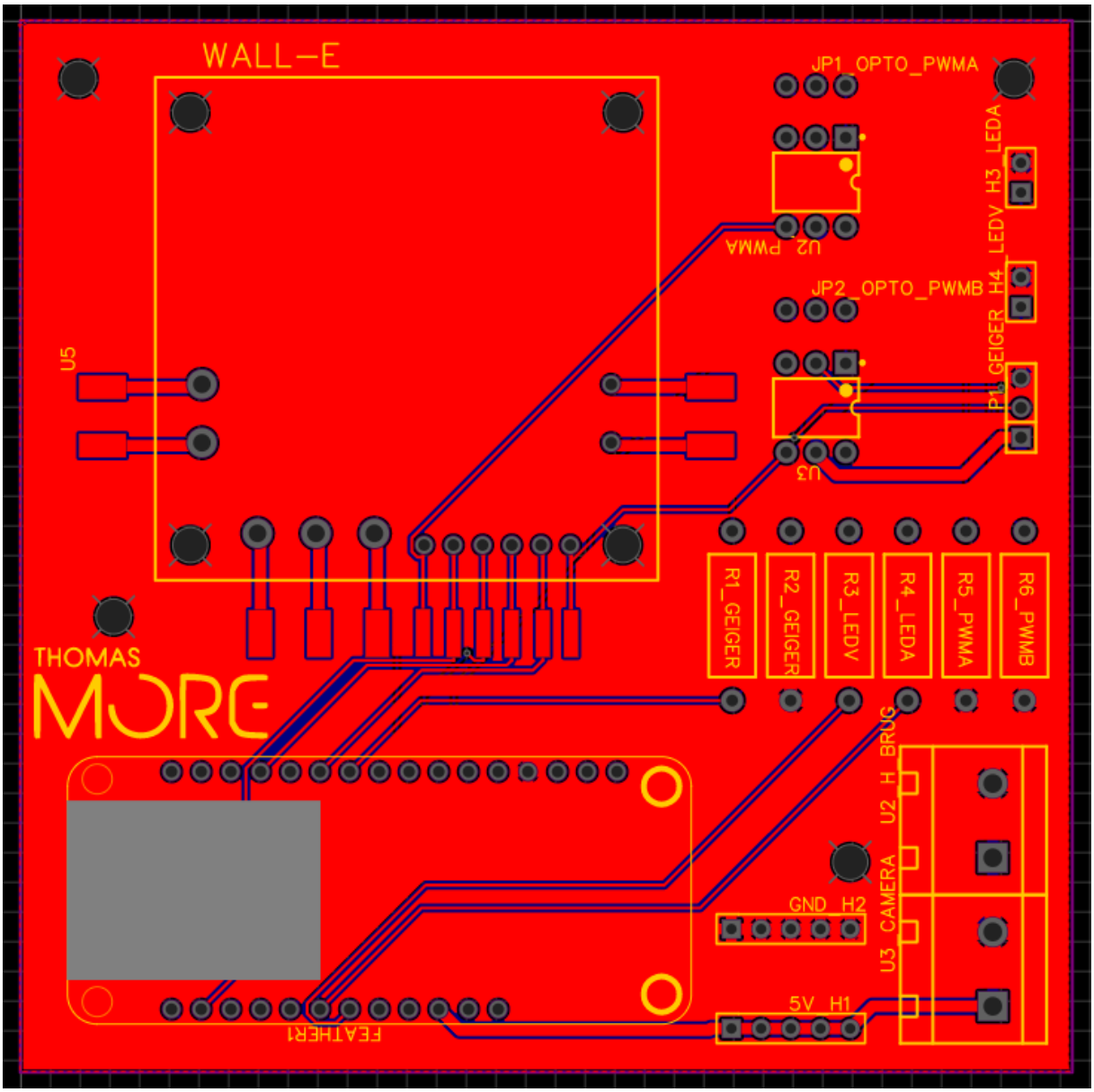
Op onderstaande afbeelding kan u het volledige schema van de printplaat bekijken.



Figuur 31: PCB schema

13.9 PCB ontwerp

We converteren het schema naar een PCB, deze PCB is nog leeg. We zetten alle componenten, boorgaten op hun plek en sluiten deze aan doormiddel van de autoroute. Dan kijken we of alle connecties juist gemaakt zijn en er geen fouten ontstaan zijn. Tenslotte zetten we over heel de PCB een GROUND plane deze zorgt ervoor dat er minder storingen zijn die de signalen van de ESP32 kunnen beïnvloeden, zoals Wi-FI.



Figuur 32: PCB ontwerp

14 COMPONENTENLIJST

Product	Aantal	Prijs/stuk	Totaal
Robot frame	1	89,95	89,95 €
H bridge	1	4,95 €	4,95 €
			- €
Camera:			- €
Fpv Bril (camera + monitor) PAL	1	107,69 €	107,69 €
Esp camera	1	18,24 €	18,24 €
Batterij lipo 14,8v			- €
			- €
Sensoren/modules:			- €
Geiger teller	1	28,06 €	28,06 €
			- €
PCB	5	5,62 €	28,10 €

Totaal incl. Btw
276,99 €

15 BESLUIT

Het is fijn de kans te krijgen om een robot te bouwen die kan ingezet worden in zwaar getroffen gebieden waar metingen moeten gebeuren. Dit project is zeer leerrijk en zeker een uitdaging omdat deze opdracht buiten onze comfortzone ligt, omwille van het mechanische gedeelte.

16 LIJST MET AFBEELDINGEN

17 BIBLIOGRAFIE

Bergersen, B. (sd). *Drones and wireless video*. Opgehaald van https://datarespons.com/: https://datarespons.com/drones-wireless-video/

ESP32 Web Server using SPIFFS (SPI Flash File System). (sd). Opgehaald van https://randomnerdtutorials.com/: https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/

ESP32 with DC Motor and L298N Motor Driver – Control Speed and Direction. (sd). Opgehaald van https://randomnerdtutorials.com/: https://randomnerdtutorials.com/esp32-dc-motor-l298n-motor-driver-control-speed-direction/

How-ToDo. (sd). *DIY Arduino Geiger Counter*. Opgehaald van https://www.instructables.com/: https://www.instructables.com/DIY-Arduino-Geiger-Counter/

witnessmenow. (sd). *Simple WiFi Controlled RC Car*. Opgehaald van https://www.instructables.com/: https://www.instructables.com/Simple-WiFi-Controlled-RC-Car/