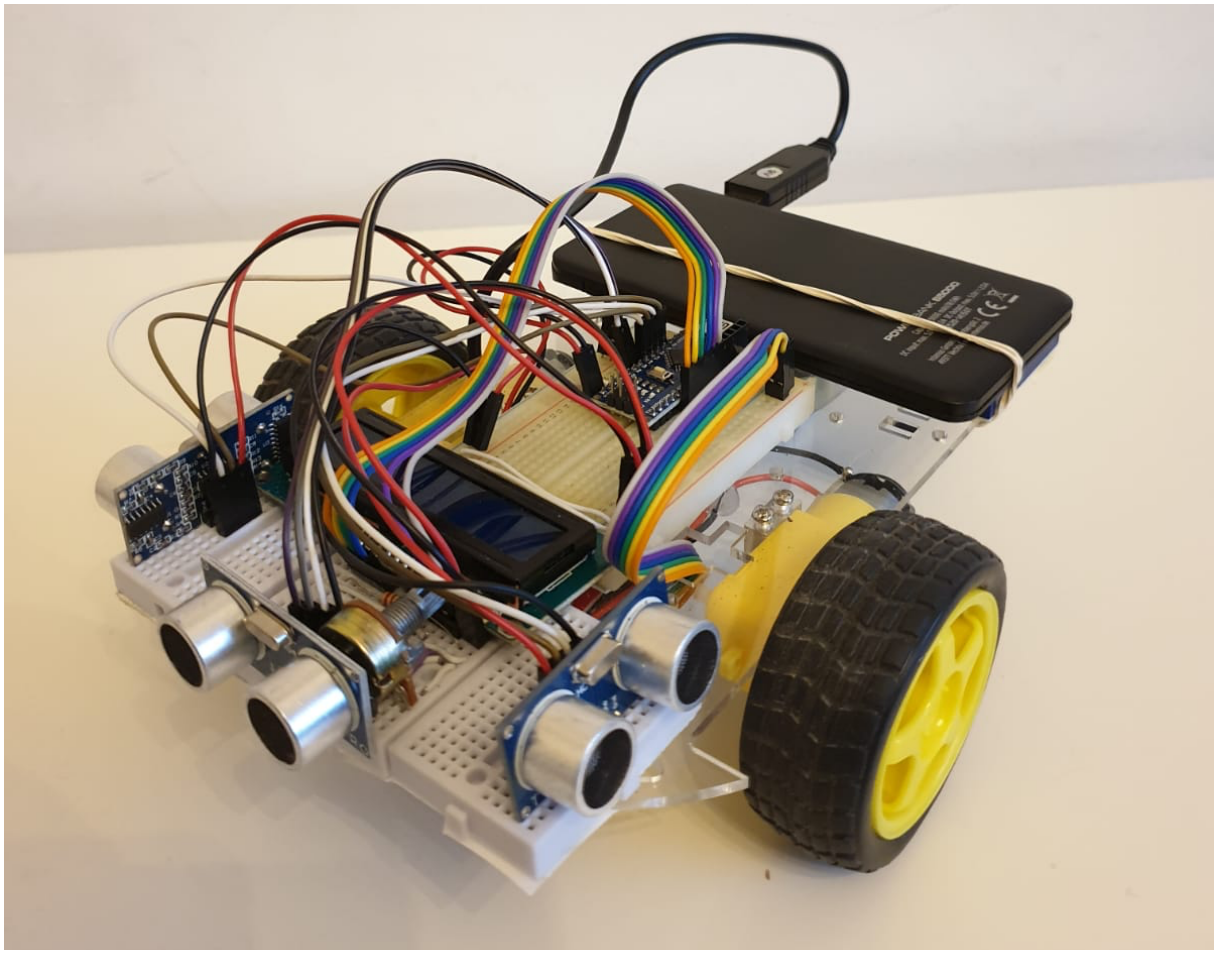
Obstakel ontwijkende robot

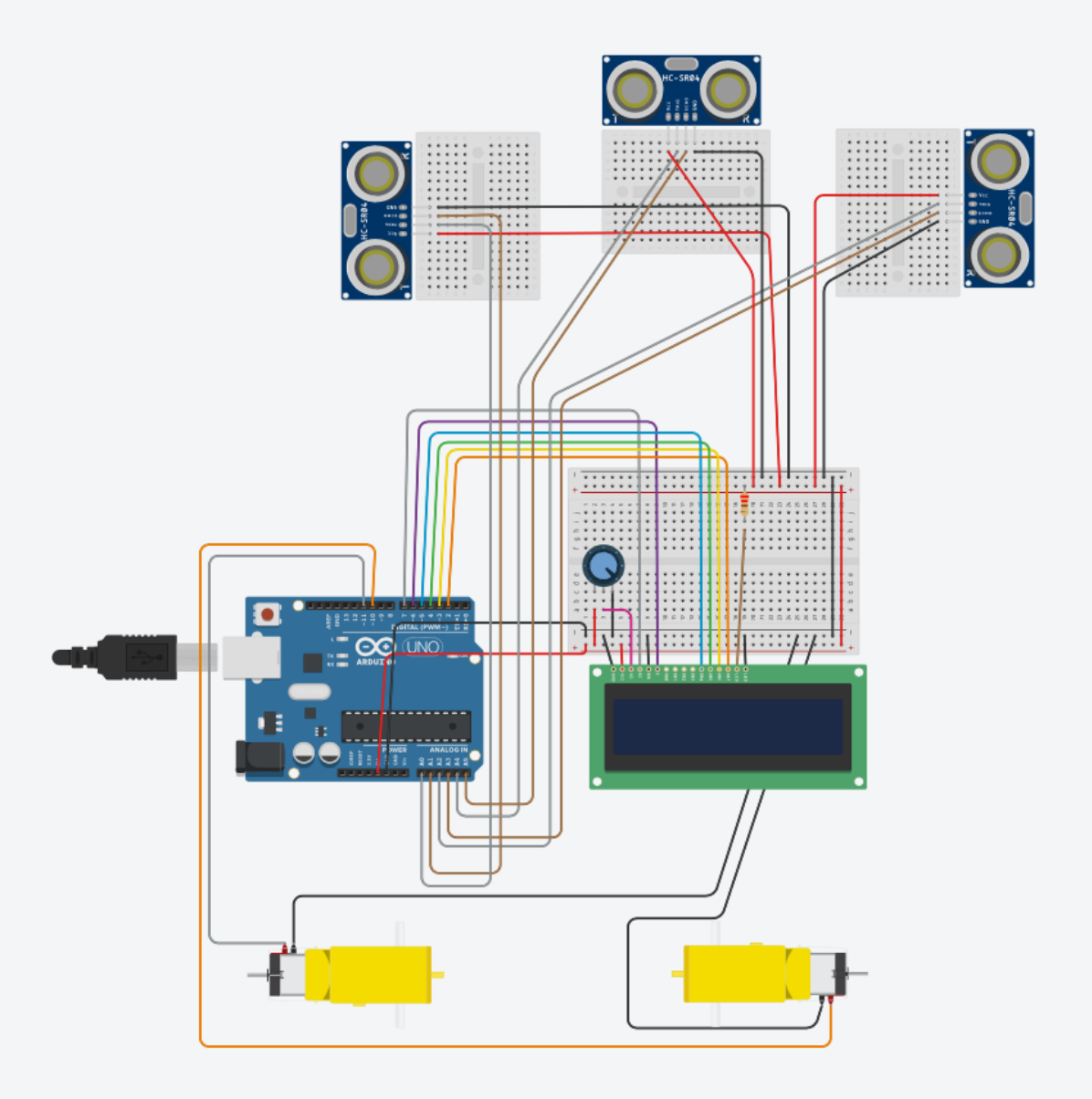
Beschrijving van de hardware v20201022 gee@emmauscollege.nl



Inleiding

Dit document beschrijft de hardware van de obstakel ontwijkende robot. Het is een bijlage van de Arduino-opdracht voor 5 havo en 5 vwo. Bij deze hardware is een voorbeeld-sketch beschikbaar. Alle documenten (opdracht, dit document en de sketch) staan in de studiewijzer op Magister.

Het schema



Onder op de auto zit een motor driver, deze is niet zichtbaar in het schema.

Het schema is gemaakt met TinkerCad. Als je wilt dan kun je ook Arduino-code maken en uitvoeren (simuleren) in TinkerCad. Daarvoor moet je een gratis account aanmaken.
De link naar het schema op TinkerCad is:
https://www.tinkercad.com/things/dIbOGA3vZfJ

Hoe werken de onderdelen?

Display

Op het display kunnen twee regels met elk 16 karakters worden weergegeven. Door aan de potmeter te draaien kun je het contrast tussen de voorgrond en achtergrond instellen. Draai aan de potmeter totdat de karakters goed leesbaar zijn. Als de potmeter in de verkeerde

stand staat dan kan het zijn dat je de karakters niet ziet. Het lijkt dan alsof het display het niet doet, terwijl hij goed is aangesloten.

Meer informatie:
https://www.arduino.cc/en/Tutorial/LibraryExamples/HelloWorld

Afstandssensor met 4 pinnen

De afstandssensoren sturen een geluid uit en meten hoelang het duurt voordat het geluid terugkaatst. Het geluid is zo hoog, dat mensen het niet horen. De sensoren werken tot een afstand van ongeveer 5 meter.

Meer informatie:
https://arduino-lessen.nl/les/afstand-meten-met-de-hcsr04-ultrasoon-sensor-op-arduino

Motor driver

De motordriver is nodig omdat de arduino-pinnen niet genoeg vermogen kunnen leveren om de motoren aan te sturen. De motordriver kan 2 motoren onafhankelijk van elkaar aansturen.

De snelheid van waarmee de motor draait kun je regelen met de duty-cycle op een arduino-pin met PWM.

In ons circuit is de motordriver zo aangesloten dat de wielen alleen vooruit kunnen draaien. Met andere bedrading en het gebruik van meer pinnen op de Arduino is het mogelijk om het circuit zo aan te passen dat de motoren zowel vooruit als achteruit kunnen draaien.

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

Arduino Nano

Een beschrijving van de mogelijkheden en de pinnen van de Arduino Nano vind je op:
https://store.arduino.cc/arduino-nano

Onderdelenlijst

Onderdeel	**Aantal**
LCD display 2x16, compatibel met Hitachi HD44780 driver (er zijn ook OLED displays of displays met I2C, daarvoor moet je het circuit en de sketch aanpassen)	1x
Weerstand 220 ohm	1x
Potmeter 10kOhm	1x
Ultrasonic Range Finder met 4 pinnen (er zijn ook versies met 3 pinnen, daarvoor moet je het circuit en de sketch aanpassen)	3x

170-punt breadboard	3x
L298N motor driver	1x
Hobby DC motor (0-9V, suggested 4.5V)	2x
Wiel	2x
Chassis	1x
Arduino Nano (+ bijbehorende usb-kabel)	1x
400-punt breadboard	1x
Powerbank 5V, minimaal 2A	1x
USB 5V to 9V step up converter	1x
Overige benodigdheden	
Dupont stekkers male/male 20cm en male/female 20 cm	
Dupont connectors 2, 4 en 6 pinnen	
Stukjes draad met harde kern voor korte verbindingen op het breadboard	
Dubbelzijdig tape	
Soldeerbout,	
Lijmpistool	
Schroevendraaiers, kniptang, striptang, schaar	

Voorbeeld-sketch

De voorbeeld-sketch bevat de code waarmee je kunt testen of alle onderdelen van je auto het doen. Deze voorbeeld-sketch is het begin van je Arduino-opdracht. Je moet de sketch zelf verder uitbreiden.

```
/****************************************
 * Auto Startcode                       *
 * met toestansdiagrammen               *
 * Emmauscollege                        *
 * v20201022GEE                         *
 ****************************************/

// libaries die je gebruikt
#include <LiquidCrystal.h>

/****************************************
 * variabelen die je gebruikt maken     *
 ****************************************/

// initialize het display
// de helderheid van het display regel je met de potmeter op de auto,
// daarvoor is geen code nodig
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

// gebruikte pinnen
const int pinAfstandTrigM = A4; // afstandssensor midden
const int pinAfstandEchoM = A5; // afstandssensor midden
const int pinAfstandTrigR = A2; // afstandssensor rechts
const int pinAfstandEchoR = A3; // afstandssensor rechts
const int pinAfstandTrigL = A0; // afstandssensor links
const int pinAfstandEchoL = A1; // afstandssensor links
const int pinMotorSnelheidR = 11; // motor rechts
```

```
const int pinMotorSnelheidL = 10; // motor links

// variabelen om waarden van sensoren en actuatoren te onthouden
long afstandR = 0;
long afstandL = 0;
long afstandM = 0;
int snelheidR = 0;
int  snelheidL = 0;
String regelBoven = "";
String regelOnder = "";

// variabelen voor de toestanden maken
// toestanden
const int TEST = 1;
const int STOP = 2;
int toestand = TEST;
unsigned long toestandStartTijd = 0;
// subtoestanden van TEST
const int RECHTSAF = 1;
const int LINKSAF = 2;
const int VOORUIT = 3;
const int WACHT = 4;
int testToestand = RECHTSAF;
unsigned long testToestandStartTijd = 0;

/***************************************
 * functies die je gebruikt maken        *
 ***************************************/

// functie om afstandssensor uit te lezen
long readDistance(int triggerPin, int echoPin)
{
  long echoTime = 0;
  pinMode(triggerPin, OUTPUT);  // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  // Reads the echo pin, and returns the sound wave travel time in microseconds
  // timeout after 30.000 microseconds (around 5 meters)
  echoTime = pulseIn(echoPin, HIGH, 30000);
  if (echoTime == 0) {
    echoTime = 30000;
  }
  return echoTime;
}

void testLoop() {
  // lees afstandssensoren uit
  // dit is nodig voor alle test toestanden
  // omrekenen naar centimers = milliseconde / 29 / 2
  afstandR = readDistance(pinAfstandTrigR, pinAfstandEchoR) / 29 / 2;
  afstandL = readDistance(pinAfstandTrigL, pinAfstandEchoL) / 29 / 2;
  afstandM = readDistance(pinAfstandTrigM, pinAfstandEchoM) / 29 / 2;

  // bepaal toestand
  if (testToestand == RECHTSAF) {
    if (millis() - testToestandStartTijd > 1000) {
      testToestandStartTijd = millis();
      testToestand = LINKSAF;
    }
  }
  if (testToestand == LINKSAF) {
    if (millis() - testToestandStartTijd > 1000) {
      testToestandStartTijd = millis();
      testToestand = VOORUIT;
    }
```

```
  }
  if (testToestand == VOORUIT) {
    if (millis() - testToestandStartTijd > 1000) {
      testToestandStartTijd = millis();
      testToestand = WACHT;
    }
  }

  // bepaal snelheid afhankelijk van toestand
  // snelheid kan 0 t/m 255 zijn
  // bij lage getallen (ongeveer onder 100) heeft de motor
  // te weinig kracht om te rijden
  if (testToestand == RECHTSAF) {
    snelheidR = 0;
    snelheidL = 128;
  }
  if (testToestand == LINKSAF) {
    snelheidR = 128;
    snelheidL = 0;
  }
  if (testToestand == VOORUIT) {
    snelheidR = 128;
    snelheidL = 128;
  }
  if (testToestand == WACHT) {
    snelheidR = 0;
    snelheidL = 0;
  }

  // zet waarden voor acturatoren, voor alle testToestanden
  // zet motorsnelheid
  analogWrite(pinMotorSnelheidR, snelheidR);
  analogWrite(pinMotorSnelheidL, snelheidL);
  // zet tekst op display
  regelBoven = String(afstandL) + "   " +
               String(afstandM) + "   " +
               String(afstandR) + "   ";
  regelOnder = String(snelheidL) +
               " TEST" + String(testToestand) + " " +
               String(snelheidR) + "      ";
  lcd.setCursor(0, 0); // zet cursor op het begin van de bovenste regel
  lcd.print(regelBoven);
  lcd.setCursor(0, 1); // zet cursor op het begin van de onderste regel
  lcd.print(regelOnder);
  // zet debug info op de seriële monitor
  Serial.print(regelBoven);
  Serial.print("--");
  Serial.println(regelOnder);
}

/*****************************************
 * setup() en loop()                     *
 *****************************************/

void setup() {
  // pinnen voor afstandssensor worden
  // voor elke meting in readDistance()
  // in de goede mode gezet

  // zet pinmode voor motor aansturing via PWM
  pinMode(pinMotorSnelheidL, OUTPUT);
  pinMode(pinMotorSnelheidR, OUTPUT);

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);

  // enable console
  Serial.begin(9600);

  // opstart bericht op console en seriële monitor
```

```
  lcd.setCursor(0, 0); // zet cursor op het begin van de bovenste regel
  lcd.print("Auto v20201021");
  lcd.setCursor(0, 1); // zet cursor op het begin van de onderste regel
  lcd.print("SETUP");// print demo bericht
  Serial.println("Auto start");
  delay(2000); // wachttijd om het display te lezen en de auto neer te zetten

  // zet toestanden en in beginstand
  toestand = TEST;
  toestandStartTijd = millis();
  testToestand = RECHTSAF;
  testToestandStartTijd = millis();
}

void loop()
{
  // toestand bepalen
  if (toestand == TEST) {
    if (millis() - toestandStartTijd > 10000) {
      toestandStartTijd = millis();
      toestand = STOP;
    }
  }
  if (toestand == STOP) {
    // de auto blijft in de toestand STOP
  }

  // de dingen doen die per toestand gedaan worden
  if (toestand == TEST) {
    testLoop();
  }
  if (toestand == STOP) {
    // zet motoren stil
    analogWrite(pinMotorSnelheidR, 0);
    analogWrite(pinMotorSnelheidL, 0);
    // zet tekst op display
    regelBoven = "                ";
    regelOnder = "      STOP      ";
    lcd.setCursor(0, 0); // zet cursor op het begin van de bovenste regel
    lcd.print(regelBoven);
    lcd.setCursor(0, 1); // zet cursor op het begin van de onderste regel
    lcd.print(regelOnder);
    Serial.println("STOP");
  }

  // vertraging om te zorgen dat de seriële monitor de berichten bijhoudt
  delay(100);
}
```