**ARDUINO**

**ADVANCED**

Jeroen van Hattem

**Voordat we kunnen beginnen met de workshop moet je de Arduino IDE gedownload hebben. Dit is de omgeving die we gaan gebruiken om de Arduino te programmeren.**

# Intro

Bij deze workshop wordt er van je verwacht dat je al enige achtergrond kennis hebt met programmeren.

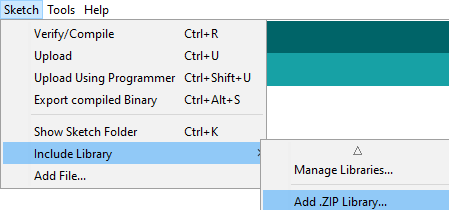
# 1. Wat gaan we maken?

Vandaag gaan we een bestuurbare auto maken met een bijbehorende controller. Dit alles gaan we zelf programmeren. De auto en de controller zullen samen communiceren via infrarood. Voor de auto maken we gebruik van de L9110 H-Bridge om de motoren aan te sturen, hier is op internet voorbeeld code voor te vinden, indien nodig. Ook gebruiken we een infrarood ontvanger, de VS1838B, in combinatie met het NEC protocol. Hier bestaan library’s voor, om dit nu helemaal vanaf scratch te maken hebben we niet genoeg tijd.

Voor de controller gebruiken we een eenvoudig infrarood ledje, in combinatie met het RC-5 protocol. Dit kan je eenvoudig programmeren als je de goede library gebruikt. In hoofdstuk 3 gaan we beginnen met hoe we onderdelen van de auto programmeren. In hoofdstuk 6 gaan we beginnen aan de controller.

# 2. Library importeren

Voordat we met infrarood kunnen werken, moeten we hier een aparte library voor importeren. We zouden dit zelf kunnen schrijven, maar dit eist veel moeite, tijd en kennis. Dus dit slaan we mooi over.

Ga naar “Sketch” -> “Include Library” -> “Add .ZIP Library…”. Ga naar de “Libraries map die je van GitHub hebt gehaald. Importeer dan “Arduino-IRremote-master.zip”.  


# 3. Aansturen van de motoren

Wat is een auto zonder zijn motoren? In dit hoofdstuk gaan we de motor aansturen, zodat we daadwerkelijk een rijdende auto hebben!

## 3.1 De opstelling

De bridge voor de motoren heeft 6 pinnen. 1 voor “5V” en 1 voor “GND”. Daarnaast zijn er 2 per motor, 1 om vooruit te gaan en 1 om achteruit te gaan. Op de bridge staat genoteerd welke pin waar thuis hoort.

## 3.2 De code

const int motorA1 = 2;

const int motorA2 = 3;

const int motorB1 = 4;

const int motorB2 = 5;

void setup() {

Serial.begin(9600);

pinMode(motorA1, OUTPUT);

pinMode(motorB1, OUTPUT);

pinMode(motorA2, OUTPUT);

pinMode(motorB2, OUTPUT);

}

void loop() {

// Vooruit

digitalWrite(motorA1, HIGH);

digitalWrite(motorB1, HIGH);

digitalWrite(motorA2, LOW);

digitalWrite(motorB2, LOW);

}

## 3.3 Code uitleg

In principe zetten we de pinnen alleen maar aan en uit. De snelheid kan je bepalen door snel te schakelen tussen aan en uit. Als je de motor 100ms aan zet en daarna 20ms uit, zal de auto sneller rijden dan wanneer je de motor 20ms aan zet en dan 100ms uit.

## 3.4 Testen

Wanneer we het programma runnen, zal de auto vooruit rijden, mits je alles goed hebt aangesloten natuurlijk.

# 4. Data lezen van de infrarood ontvanger

Nadat we de motoren aan kunnen sturen, is het ook leuk als we daar wat meer mee zouden kunnen doen. We kunnen bijvoorbeeld een infrarood signaal op kunnen vangen, en hiermee commando’s sturen om de wielen aan te sturen. Ook zullen we de RGB leds op de auto kunnen veranderen wanneer we dat willen. Zo kunnen we met onze zelfgemaakte controller de auto besturen.

## 4.1 De opstelling

De linker pin is voor het teruggeven van het ontvangen signaal, deze sluiten we aan op “D7”. De middelste pin sluiten we aan op de “GND”. De rechter pin sluiten we aan op de “5V”.

## 4.2 De code

#include <IRremote.h>

#include <IRremoteInt.h>

const int RECV\_PIN = 7;

IRrecv irrecv(RECV\_PIN);

decode\_results results;

void setup() {

Serial.begin(9600);

pinMode(RECV\_PIN, INPUT);

irrecv.enableIRIn(); // Start the receiver

}

void loop() {

if (irrecv.decode(&results)) {

Serial.println("VALUE");

Serial.println(results.value);

irrecv.resume(); // Receive the next value

}

delay(50);

}

## 4.3 Code uitleg

IRrecv irrecv(RECV\_PIN);

Met deze regel maken we een object van de klasse “IRrecv" aan. Als parameter geven we de pin mee waar we de receiver hebben aangesloten.

decode\_results results;

Hier maken we een object aan van een bepaald type, hier slaan we de ontcijferde infrarood signalen in op.

irrecv.enableIRIn();

Hier schakelen we de code in die er voor zorgt dat alles bij de receiver wordt afgehandeld.

if (irrecv.decode(&results)) {

Hier wordt gekeken of de ontvangen informatie kan worden ontcijferd. Als dit kan zijn we er zeker van dat het een correct en volledig bitpatroon is, waar we mee kunnen werken.

irrecv.resume();

Hier zeggen we dat de code die de ontvangen signalen afhandelt weer door kan met scannen.

## 4.4 Testen

Wanneer we het programma runnen zien we dat het ledje knippert om de 100ms.

# 5. Aansturen van de RGB leds

Een goede auto heeft natuurlijk neon verlichting. Dit gaan we nabootsen door middel van RGB leds. Deze RGB leds werken in principe hetzelfde als gewone LEDs. Er zitten 3 aparte ledjes in, die we individueel zullen aansturen. Door middel van deze 3 ledjes kunnen we 7 kleuren nabootsen.

## 5.1 De opstelling

De RGB leds hebben 4 pins. 1 voor de GND, 1 voor blauw, 1 voor rood en 1 voor groen. Tussen de laatste drie pinnen MOET een weerstand van 220Ω.

## 5.2 De code

const int rgb\_red = 11;

const int rgb\_green = 12;

const int rgb\_blue = 13;

int led\_toggle;

void setup() {

Serial.begin(9600);

pinMode(rgb\_red, OUTPUT);

pinMode(rgb\_green, OUTPUT);

pinMode(rgb\_blue, OUTPUT);

}

void loop() {

led\_toggle = (led\_toggle + 1) % 7;

switch (led\_toggle) {

case 0:

digitalWrite(rgb\_red, HIGH);

digitalWrite(rgb\_green, HIGH);

digitalWrite(rgb\_blue, HIGH);

break;

case 1:

digitalWrite(rgb\_red, LOW);

digitalWrite(rgb\_green, HIGH);

digitalWrite(rgb\_blue, HIGH);

break;

case 2:

digitalWrite(rgb\_red, HIGH);

digitalWrite(rgb\_green, LOW);

digitalWrite(rgb\_blue, HIGH);

break;

case 3:

digitalWrite(rgb\_red, HIGH);

digitalWrite(rgb\_green, HIGH);

digitalWrite(rgb\_blue, LOW);

break;

case 4:

digitalWrite(rgb\_red, LOW);

digitalWrite(rgb\_green, LOW);

digitalWrite(rgb\_blue, HIGH);

break;

case 5:

digitalWrite(rgb\_red, LOW);

digitalWrite(rgb\_green, HIGH);

digitalWrite(rgb\_blue, LOW);

break;

case 6:

digitalWrite(rgb\_red, HIGH);

digitalWrite(rgb\_green, LOW);

digitalWrite(rgb\_blue, LOW);

break;

}

}

## 5.3 Code uitleg

led\_toggle = (led\_toggle + 1) % 7;

Deze regel zorgt er voor dat de “led\_toggle” variabele telkens iets word verhoogd. De “% 7” zorgt ervoor dat deze variabele nooit boven de 6 zal komen.

irsend.sendNEC(0x03FCFF00, 32);

We roepen hier van de klasse “irsend” de “sendNEC()” functie aan. Deze functie vraagt om 2 parameters: de code die je wilt sturen, en het aantal bits die je wilt sturen. Bij de eerste parameter geven we dus de code mee die we bepalen: groepsnummers, inverted groepsnummer, command en inverted command. In totaal 4 bytes, omgerekend naar hexadecimale getallen. Bij de 32 parameter geven we 32 mee, omdat het NEC protocol met 32 bits data werkt.

delay(40);

We willen hier even wachten, zodat de signalen niet direct achter elkaar verzonden worden.

## 5.4 Testen

Wanneer je deze code uitvoert, moet je op de tweede Arduino de Serial Monitor openen (CTRL+SHIFT+M). Hier zal je dan de codes uit kunnen lezen.

# 6.Data lezen van de joystick

We gaan hier een stukje code schrijven waarmee we de data van de joystick kunnen uitlezen.

De joystick werkt analoog en stuurt een X en Y waarde door. We hebben hiervoor 2 pins nodig. Voor de knop hebben we een digitale pin nodig.

## 6.1 De opstelling

De joystick heeft 5 pins. 1 voor 5V, 1 voor GND, 1 voor de Y as, 1 voor de X as en 1 voor de knop. De pinnen voor de 2 assen sluiten we een op de **analoge** pins A0 en A1. De knop sluiten we aan op D2.

Aansluiting kan afwijken van het plaatje, afhankelijk van welke joysticks we hebben.

## 6.2 De code

const int joystick\_x = A0;

const int joystick\_y = A1;

const int joystick\_button = 2;

int x\_value = 0;

int y\_value = 0;

void setup() {

pinMode(joystick\_x, INPUT);

pinMode(joystick\_y, INPUT);

pinMode(joystick\_button, INPUT);

digitalWrite(joystick\_button, HIGH);

Serial.begin(9600);

}

int convert\_value(int data) {

return (data \* 12 / 1024);

}

void loop() {

x\_value = convert\_value(analogRead(joystick\_x));

y\_value = convert\_value(analogRead(joystick\_y));

Serial.print("Joystick button: ");

Serial.println(digitalRead(joystick\_button));

Serial.print("X-axis: ");

Serial.println(convert\_value(analogRead(joystick\_x)));

Serial.print("Y-axis: ");

Serial.println(convert\_value(analogRead(joystick\_y)));

Serial.print("\n\n");

delay(500);

}

## 6.3 Code uitleg

x\_value = convert\_value(analogRead(joystick\_x));

De “convert\_value()” functie neemt de waarde die de pin terug keert en zet deze om in een eenvoudige waarde waar we mee kunnen werken. In dit geval 0 tot 11, waar 5 neutraal is. De “analogRead()” functie geeft een waarde van 0 tot 1023 terug. Dit is niet erg handig om mee te werken.

## 6.4 Testen

Wanneer we de “Serial Monitor” openen, zullen we zien dat je kan zien wanneer de knop wordt ingedrukt. Ook worden er 2 waardes tussen 0 en 11 terug gegeven, respectievelijk voor de X en Y as.

# 7. Aansturen van de infrarood verzender

Voor het communiceren met de auto moeten we draadloos data kunnen versturen. Hiervoor gebruiken wij infrarood in combinatie met het NEC protocol.

Bij het NEC protocol moeten 32 bits worden verstuurd. Eerst 8 bits met het adres. Daarna dezelfde 8 bits, maar dan inverted. Eentjes en nullen flippen dus. Daarna komen er 8 bits met het command, hierna komen dezelfde 8 bits inverted.

Bij het adres pakken we je groepsnummer aangevuld met nullen. Zou je groep nummer 3 zijn zou dit in binair ‘00000011’ zijn. Het inverted deel zou dan ‘11111100’ zijn. Bij het command gedeelte geven we het de code voor de opdracht die de auto moet uitvoeren. Al wil je de auto vooruit laten rijden zou je bijvoorbeeld alle bitjes hoog kunnen zetten. Dan zou je uitkomen op ‘11111111’. Het inverted deel zou dan ‘00000000’ zijn.

Hiermee zou je eindigen op het volgende bit-patroon: ‘00000011111111001111111100000000’. Om dit overzichtelijker te kunnen maken kan je dit omzetten in hexadecimale getallen. Dit zou dan ‘03FC FF00’ zijn. Ook kan je het omzetten in decimale getallen, dit zou dan op ‘66912000’ uitkomen.

Als je de “sendNEC()” functie niet fijn vindt werken, staat er in de complete code in de “Code” map, een functie die eenvoudiger werkt.

## 7.1 De opstelling

Het ledje in de opstelling is de infrarood led. Het is essentieel om deze op pin 3 aan te sluiten! Anders gaat je code niet werken. Dit zit zo verwerkt in de library. De led moet worden aangesloten op een 220Ω weerstand.

## 7.2 De code

#include <IRremote.h>

IRsend irsend;

void setup() {

Serial.begin(9600);

}

void loop() {

for (int i = 0; i < 3; i++) {

irsend.sendNEC(0x03FCFF00, 32);

delay(40);

Serial.println("Sending signal");

}

delay(100);

}

## 7.3 Code uitleg

for (int i = 0; i < 3; i++) {

We gaan hier door een “for-loop” waarin we de infrarood signalen verzenden. We doen dit 3 keer, omdat de zender soms de signalen niet opvangt. Om er zeker van te zijn dat dit wel gebeurd doen we dit dus 3 keer.

irsend.sendNEC(0x03FCFF00, 32);

We roepen hier van de klasse “irsend” de “sendNEC()” functie aan. Deze functie vraagt om 2 parameters: de code die je wilt sturen, en het aantal bits die je wilt sturen. Bij de eerste parameter geven we dus de code mee die we bepalen: groepsnummers, inverted groepsnummer, command en inverted command. In totaal 4 bytes, omgerekend naar hexadecimale getallen. Bij de 32 parameter geven we 32 mee, omdat het NEC protocol met 32 bits data werkt. Omdat we hexadecimale getallen sturen, moet er “0x” voor het getal staan.

delay(40);

We willen hier even wachten, zodat de signalen niet direct achter elkaar verzonden worden.

## 7.4 Testen

Wanneer je deze code uitvoert, moet je op de tweede Arduino (met de code voor de ontvanger) de Serial Monitor openen (CTRL+SHIFT+M). Hier zal je dan de codes uit kunnen lezen.

# 8. Zelf aan de slag

Als je al deze dingen samenvoegt, kan je zelf een bestuurbare auto in elkaar zetten. Succes!

# Nawoord

## Studievereniging

Deze workshop is gegeven in samenwerking met de studievereniging Indicium**.** Deze studievereniging is speciaal voor studenten van de opleiding HBO-ICT. Wanneer je lid bent van deze studievereniging heb je toegang tot veel toffe activiteiten die georganiseerd zullen worden.

## Thuis verder aan de slag met een Arduino?

Als je het werken met een Arduino leuk hebt gevonden en je zou thuis verder willen gaan met het programmeren van een microcontroller, dan zou je kunnen overwegen om zelf een Arduino Uno aan te schaffen.

Officiële Arduino’s zijn te koop op veel plekken in Nederland, waaronder Conrad. Mocht je budget wat kleiner zijn, kan je overwegen om een Arduino “clone” te halen. Deze bordjes werken hetzelfde, maar komen niet van Arduino zelf.

Heb je toch geen zin om een Arduino te halen? Op [circuits.io](http://circuits.io) kan je online een Arduino simuleren. Als je problemen hebt met een code kan je hier ook kijken of het aan jouw Arduino ligt of aan je code.