DESIGN DOCUMENT

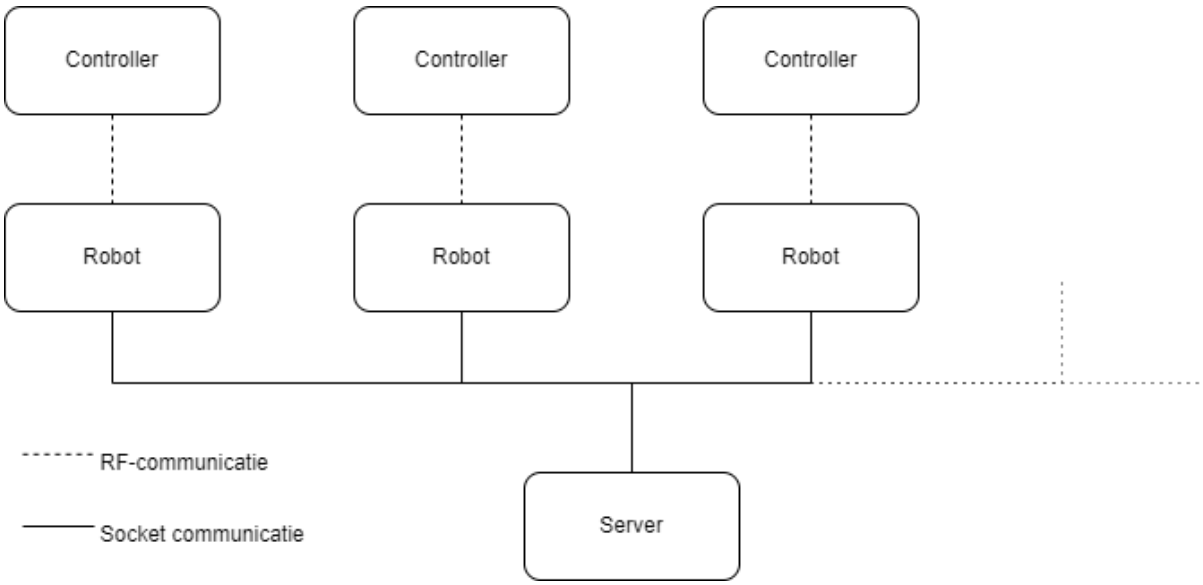
Battlebots project

Contents

1. Introductie

In dit document komen de verschillende designs te staan van het Battlebots project. Dit wil zeggen dat er voor elk component van het systeem een klasse diagram wordt gemaakt. Ook komen de afspraken over de communicatie in dit document te staan.

2. Het systeem



In de bovenstaande afbeelding is te zien dat er aan de server verschillende robots verbonden kunnen worden. De communicatie tussen deze componenten gaat via socket communicatie.

Elke robot heeft een eigen controller. De communicatie tussen de controller en de robot gaat via radio frequentie.

De server fungeert als game manager, dit wil zeggen dat de server bijhoudt welke robots er geconnect zijn en welke games er actief zijn.

Voor elke actieve game maakt de server een nieuw proces aan. Als de game is afgelopen houdt de server bij welke robot er heeft gewonnen.

3. Protocollen

Alle berichten tussen de robot en de server zullen beginnen met een '#' en eindigen met een '%'.

3.1. Robot – Server

Server -> Robot	*Betekenis:*
Ping	Check of de connectie nog aanwezig is.
Game:GAMEID:id,id,id,…	De robot zit in een game met de volgende andere robots. En het volgende gameID
GameStart	Start de game die van te voren is doorgegeven.
GameStop	Stop de game die bezig was.
ACK	Bericht correct ontvangen.
NACK	Bericht niet correct ontvangen.

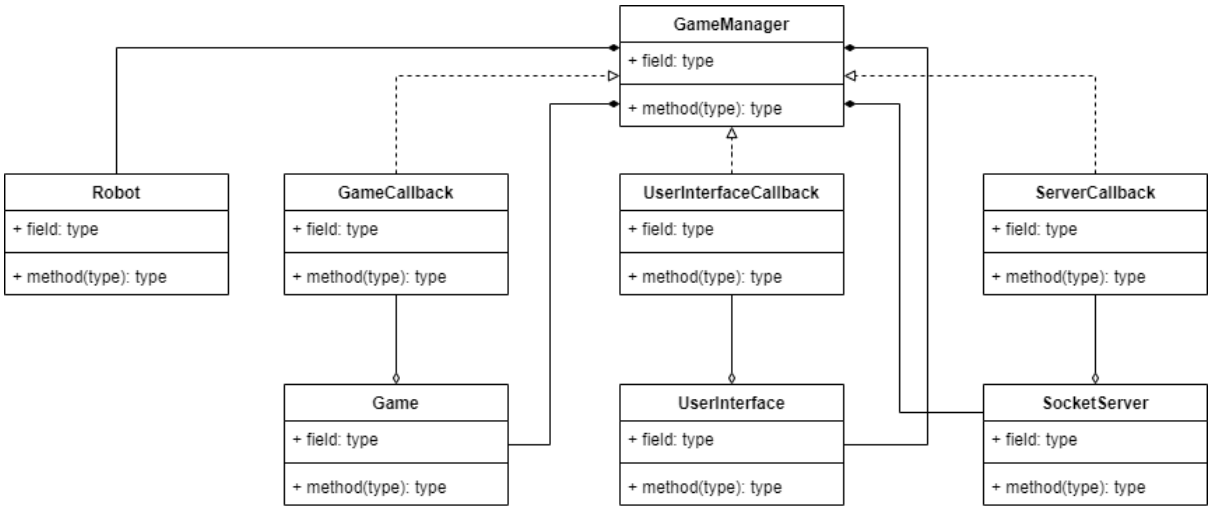
Robot -> Server	*Betekenis:*
*Hit:*id	Geraakt door robot met nummer: id.
ACK	Bericht correct ontvangen.
NACK	Bericht niet correct ontvangen.
PACK	Acknowledge op een ping bericht.

3.2. Robot – Controller

Controller -> Robot	*Betekenis:*
Data	Stuurt de 16-bit waardes van de X- en Y-as van de joystick naar de Robot.
	Eerst de onderste 8 bits van de X-as gevolgd door de bovenste 8 bits. Zelfde geldt voor de Y-as

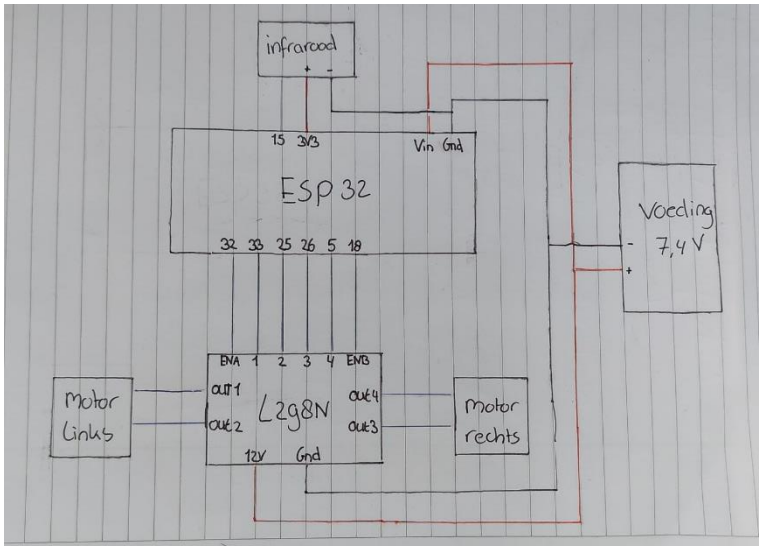
4. Klasse diagram

4.1. Server



5. Robot

De robot is het grootste hardware deel van dit project, hierbij was het belangrijk dat de microcontroller kon communiceren met de server via wifi. De ESP32 is hier een hele goede microcontroller voor omdat die zelf al op het bordje een ingebouwde wifi module heeft ter verglijking met de arduino waarbij je een wifi shield erop moet doen en het dan ook nog heel lastig is om die te coderen door de wifi library. De ESP32 is daar dan gelukkig heel makkelijk in en is dan ook de keuze waarvoor ik gegaan ben. Vervolgens hadden we motoren nodig om de robot te kunnen laten bewegen, eerder in dit semester heb ik een onderzoek gedaan naar welke motoren geschikt waren voor ons project daar kwamen de DC motoren van arduino uit. Maar toen kwam nog de vraag hoeveel motoren en welke wielen, daarvoor is gekozen voor 2 wielen aan de achter kant en dan een klein zwenkwiel aan de voorkant dit zorgt ervoor dat de robot ook over kleine obstakels kan rijden. Om de motoren aan te kunnen sluiten op de ESP32 is er een motor module nodig, ik heb voor de l298N gekozen waarbij je de 4 inputs voor de motoren (Vooruit/achteruit) en voor elk wiel de PWM waarde. Ook zit er nog een infrarood sensor op voor de shooter maar dat word later in dit document nog beschreven.



6. IR Systeem

6.1. TSOP34836

De TSOP34836 is een infraroodsensor die specifiek is ontworpen voor gebruik met een IR-afstandsbediening. Het heeft een werkende frequentie van 36 kHz en een gevoeligheid van -40 dBm. Het is verpakt in een kleine SMD-behuizing en heeft een brede invalshoek, waardoor het geschikt is voor toepassingen waar de afstandsbediening niet precies gericht hoeft te zijn. Het wordt vaak gebruikt in huishoudelijke apparaten, audio- en videoapparatuur, enz.

Wij kunnen deze sensor gebruiken voor het uitlezen van de IR signalen wat het schietsysteem afschiet. Hierdoor kunnen we zien wie er het signaal afgeschoten heeft.

6.2. Uitleg code

De IrReceiverLoop functie haalt gegevens op van de IR-ontvanger, decodeert deze gegevens en slaat de gedecodeerde gegevens op in de pointer ID als het geldige gegevens zijn, anders geeft het -1 terug. Als er geen bericht ontvangen wordt, geeft het -2 terug.

De eerste stap in de functie is om te controleren of de pointer ID niet null is. Als de pointer null is, geeft de functie 0 terug en worden er geen gegevens opgeslagen.

Vervolgens gebruikt de functie de IrReceiver.decode() functie van de IRremote library om te proberen gegevens te decoderen die zijn ontvangen door de IR-ontvanger. Als er gegevens zijn gedecodeerd, controleert de functie of de gedecodeerde gegevens geldig zijn door te controleren of IrReceiver.decodedIRData.decodedRawData niet 0 is. Als de gedecodeerde gegevens geldig zijn, slaat de functie de gedecodeerde gegevens op in de pointer ID met de regel: *ID = IrReceiver.decodedIRData.decodedRawData; en geeft de functie 1 terug om aan te geven dat de operatie succesvol is geweest.

Als de gedecodeerde gegevens ongeldig zijn, geeft de functie -1 terug.

Als er geen gegevens zijn gedecodeerd, geeft de functie -2 terug om aan te geven dat er geen bericht ontvangen is.

Tot slot, wordt IrReceiver.resume() aangeroepen om te voorkomen dat de ontvanger blijft hangen in de decode modus. Dit zorgt ervoor dat de ontvanger weer in staat is om nieuwe gegevens te ontvangen.