**Dia 1 (intro): Wout**

Goedemorgen allemaal, wij zijn Wout Vandevelde & Noah Vermaerke, en wij hebben voor ons eindproject een Robot met mecanum wielen gemaakt.

**Dia 2 (doel project) Noah**

Het doel van ons project was om een robot te maken die we kunnen besturen met een zelfgemaakte afstandsbediening, en een digitaal dashoard op de laptop, gsm of tablet. Dit moet communiceren met elkaar via het Wifi-netwerk.

En we hebben een website over ons project gemaakt die je kan bezoeken via deze link.

**Dia 3 (blokschema en globale uitleg) Wout**

Hier zie je het blokschema van het project, je ziet links het robotframe met de Raspberry Pi en de Powerbank.

Rechts zie je het blokschema van de afstandsbediening, het bevat een ESP32 Feather, waar alles op geprogrammeerd staat, je ziet ook 2 joysticks, een potentiometer voor de snelheid te regelen, een batterij en een schakelaar op de controller aan en uit te zetten.

Onderaan rechts zie je het digitaal dashboard die we gemaakt hebben om op bvb je gsm de robot te kunnen besturen.

En als laatst zie je de router die het draadloos netwerk voorziet, alle communicatie tussen de robot, controller en dashboard verloopt langst hier.

**Dia 4 (de robot) Noah**

De robot zelf bestaat uit het frame, 4 DC motoren, een Raspberry Pi, 2 H-bruggen en een powerbank.

DC-motoren zijn kleine elektrische motoren die we gebruiken om de mecanum wielen te laten draaien.

Een Raspberry Pi is een mini-computer, enhij is verbonden met het draadloos netwerk. De controller stuurt zijn berichten ernaartoe, en de raspberry stuurt dan via de H-bruggen de DC-motoren aan.

De H-bruggen gebruiken wij, omdat we hiermee de draairichting en de draaisnelheid van de DC motoren kunnen regelen, door de juiste pinnen aan te sturen op de Raspberry Pi.

**Dia 5 (mecanum wielen) Wout**

We gebruiken mecanumwielen omdat dit speciale wielen zijn waarmee de robot in alle richtingen kan rijden, zonder eerst te moeten draaien.

Dat komt doordat er schuine rolletjes op elk wiel zitten. Door de wielen op een slimme manier te laten draaien, kan de robot vooruit, achteruit, zijwaarts en zelfs schuin bewegen.

**Dia 6 (de controller) Noah**

De controller bestaat uit 5 onderdelen, de ESP32 Feather, 2 joysticks, een potentiometer, een batterij en een schakelaar om de controller aan en uit te zetten

De ESP32 Feather is een micro-controller, we kunnen deze makkelijk programmeren, zodat we de waardes van de joysticks en de potentiometer kunnen uitlezen en die doorsturen.

De linker joystick is om de robot te laten rijden in de richting waar je de joystick naartoe duwt, de rechter joystick is om de robot links of recht rond z’n as te laten draaien.

En de potentiometer gebruiken we om de snelheid van de robot te regelen.

**Dia 7 (MQTT) Wout**

MQTT gebruiken wij om te communicatie te voorzien tussen de controller en de robot, en tussen de robot en het dashboard op je gsm, laptop of tablet.

Het is een lichtgewicht communicatieprotocol, het wordt veel gebruikt in IoT-toepassingen, bijvoorbeeld een slimme deurbel of slimme koelkast, etc..

MQTT werkt met het publish en subscribe model.

Eerst hebben we de **broker**, dit is de centrale server die de berichten ontvangt en doorstuurt. Bij ons staat deze geïnstalleerd op de Raspberry Pi.

Dan, een **publisher** is bijvoorbeeld een sensor of apparaat die een bericht stuurt naar een topic, bij ons is dat bijvoorbeeld “Rij vooruit” dat verstuurd wordt naar het topic “Robot/Richting”.

En als laatst, een **subscriber**, dezeontvangt de berichten van de topics waarop hij is geabonneerd, deze info kan dan worden gebruikt om specifieke pinnen aan te sturen op de Raspberry Pi.

**Dia 8 (Node-RED) Noah**

Node-RED is een programma die we gebruiken om de logica in de Raspberry Pi makkelijk visueel op te bouwen, en het werkt perfect met MQTT-communicatie.

Door de MQTT-berichten te ontvangen op Node-RED, kan het de pinnen op de Raspberry Pi aansturen, zodat de DC-motoren in de juiste richting draaien met de juiste snelheid.

Ons dashboard en de logica erachter is ook gemaakt met Node-red

**Dia 9 (Dashboard) Wout**

Het dashboard is een soort website/app waarop je bepaalde aansturingen kan doen. Je kan op het dashboard de robot vooruit, achteruit, links en rechts laten rijden, en ook rond zijn as laten draaien. Met de controller kan je dus meer precieze richtingen op gaan met de robot.

Door op de pijltjes te klikken - of op het draai icoontje te klikken - en de slider van de maximum snelheid te verhogen kan je de robot laten rijden. Als je de robot wilt laten stoppen druk je op het vierkante knopje in het midden, of zet je de slider op 0.

Als je het dashboard wilt gebruiken in plaats van de controller, zet je de controller gewoon uit via de schakelaar op de achterkant.

**Dia 10.1 (Flow Node-RED RPi) Noah**

Op de foto zie je links 3 MQTT-in nodes (paars), die dienen om de MQTT-berichten, die binnen komen van de mosquitto broker op de Raspberry Pi, op te halen. De eerste ontvangt berichten met topic “Robot/Richting” op, de tweede berichten met topic “Robot/Draaien”, en de laatste berichten met topic “Robot/Snelheid”.

Deze worden verder gestuurd naar één function node (oranje), deze filtert alle berichten op topic (vb. Robot/Richting, …), daarna stuurt deze, afhankelijk van wat het bericht is in de topic, de juiste pinnen aan op de Raspberry Pi (via de rpi-gpio out nodes (blauw)).

De eerste 4 rpi-gpio out nodes zijn bedoelt om elke motor aan of uit te zetten (true = aan), de volgende 4 zijn om de richting van elke motor te bepalen (true = vooruit) en de laatste 4 zijn om de snelheid van elke motor te regelen (getal van 0 tot 100) (100 = snelste). Dit gebeurt met PWM, dat wil zeggen als een men getal van 60 meegeeft, de pin 60% van de tijd hoog zal zijn en 40% van de tijd laag. Dit gebeurt heel snel.

Op de foto kan je ook zien aan de titels van de nodes, welke pinnen ze aansturen.

**Dia 10.2 (Flow Node-RED Dashboard) Wout**

Dit is de flow van het dashboard, links zie je de 8 blokken die gekoppeld zijn aan de knoppen/slider op het dashboard. De eerste 4 zijn voor de richting van de robot, de 2 erna zijn voor het draaien van de robot. Ze geven hun staat (true of false) door aan de bovenste function node die ernaast staat (oneven function node), en die zorgt ervoor dat de MQTT-out node (paars) ernaast, het juiste bericht krijgt, dat bericht wordt dan verstuurd via MQTT naar de mosquitto broker op de Raspberry Pi. De andere function node zorgt ervoor dat als je een knop aanzet dat de andere knoppen uitgaan, zo kunnen er geen 2 richtingen tegelijk aan staan. De laatste (de slider) is voor het regelen van de snelheid van de robot, deze geeft direct een waarde van 0 tot 100 door aan de MQTT-out node en dit wordt dan verder verstuurd op dezelfde manier.

**Dia 11 (Energiebeheer) Noah**

Om onze Raspberry Pi op de robot te voorzien van stroom, maken we gebruik van een normale powerbank, omdat de Raspberry Pi werkt op een spanning van 5V. Dus dat is ideaal. Deze is makkelijk op te laden met een USB-C kabeltje.

En om de Controller te voorzien van stroom, hebben we een batterij aangesloten op de microcontroller in de controller. Deze werkt op een Spanning van 3,7 Volt, en dat is ideaal voor de microcontroller. De batterij heeft een capaciteit van 500 mili Ampère uur. De batterij kan je opladen door een USB kabeltje aan te sluiten op de USB poort van de microcontroller.

We hebben ook in de controller een aan en uitschakelaar zodat als we de controller niet gebruiken, er niks verloren gaat.

**Dia 12 (Laptop & programmering) Wout**

De programma’s die we gebruikten om ons project te maken zijn;

**Node-RED**, voor het dashboard te maken.

**MQTT-Explorer**, dit gebruikten we om de MQTT-communicatie tussen de controller en de Raspberry Pi te controlleren, we kunnen inloggen op de MQTT broker op de Raspberry Pi en zo kijken per topic of de berichten wel toekwamen, en zo konden we mogelijke problemen uitsluiten en verder aan het werk.

Om de controller te programmeren maakten we gebruik van **Arduino IDE**, de taal die we programmeerden op de microcontroller is C, het is relatief eenvoudig om hierop de controller te programmeren.

Vervolgens, om de website te maken, gebruikten we **Visual Studio Code**, daar programmeerden we de opmaak in HTML en de lay-out in CSS

Node-RED gebruikten we ook nog eens om de pinnen aan te sturen op de Raspberry Pi, De programmeertaal die we daar op hebben gebruikt is javascript.

En om alle behuizingen van ons project te maken, zoals de behuizing van de controller, gebruikten we Autodesk Inventor. Dat is een programma waar je objecten in 3D kunt tekenen. Die 3D-modellen hebben we dan afgeprint met de 3D printer.

**Declaraties Noah**

Dit is de code die geprogrammeerd staat op onze microcontroller in de afstandsbediening.

Eerst voert de controller de methode “declaraties” uit, deze gaat eerst aan de pinnen die we gebruiken op de Microcontroller, een duidelijke naam geven.

Hij zal ook een deadzone instellen voor de joysticks.

Daarna gaat de methode alle methodes die we later gebruiken bekend maken.

Dan worden de Wifi instellingen ingesteld, zoals de naam en het wachtwoord. En er worden ook instellingen gemaakt voor de MQTT communicatie.

Als laatst in de declaraties, zullen er enkele variabelen aangemaakt worden waar er later waardes in komen.

**Setup Wout**

In de methode van de setup, word als eerst een andere methode aangeroepen “**initWifi**”.

Deze methode zal verbinding maken met het draadloos netwerk, daarna controlleert hij als de verbinding is gemaakt, zoja, zal de naam van de wifi verbinding ingesteld worden op “ESP32 node wout”. Zoniet zal deze elke halve seconde controlleren of de verbinding al is gemaakt.

Daarna zal in de setup, de MQTT-verbinding gemaakt worden met de broker op de Raspberry Pi, de verbinding heeft de naam “mqttClient1”.

Als laatst zal de setup de methode “**initINPUTS**” aanroepen. Deze methode stelt de in en uitgangen in. Eerst stelt hij de X- en Y-assen van beide joysticks in als een INGANG, daarna de 2 drukknoppen op de joystick als een INGANG met pullup weerstand. Dat wil zeggen dat er intern een weerstand is voorgeschakelt, zo moet je er geen fysieke weerstand voor plaatsen. En als laatst wordt de potentiometer ook ingesteld als een ingang.

**Loop Noah**

Dan gaan we verder naar de loop, deze methode wordt in een oneindige lus herhaald.

In de loop wordt eerst gecontrolleerd als de MQTT-verbinding nog in stand is, zoniet, zal de methode “**mqttConnect**” aangeroepen worden. Deze methode gaat om de 5 seconden proberen verbinding te maken met de broker, als dit gelukt is gaan we verder naar de loop.

Dan gaan we verder door de MQTT-verbinding in stand te houden.

Als laatst zal de loop de methode “**afstandsbediening**” aanroepen.

**Afstandsbediening Wout**

Deze is de langste methode van de code.

De methode zal wat variabelen aanmaken met de mogelijke rijrichtingen van de robot, dan zal er een methode aangeroepen worden die de waardes van de joystick en de potentiometer uitleest. Die waardes plaatst hij dan in de gemaakte variabelen.

Dan conrtolleert de methode of dat de waardes van joystick voorbij de waardes van de deadzone komen.

Als dat zo is zal hij de juiste variabelen True maken voor de juiste rij- of draairichting.

Als alle mogelijkheden zijn gecontrolleerd, zal hij de juiste rij- of draairichting doorsturen naar de Rapberry Pi, door de richting in een bericht te publishen met het juiste topic naar de broker op Rasoberry Pi.

Maar als er geen enkele rij- of draairichting is geselecteerd, zal er een bericht met het woord “STOP” gepublished worden naar het topic Robot/Richting op de broker.

De berichten worden dan verwerkt in Node-RED op de Raspberry Pi en dan worden de juiste pinnen aangestuurd, en draaien de motoren in de juiste richting.

Deze hele loop wordt elke 100 milliseconden herhaald.

**Laatste Dia**

Dit was hoe wij ons project maakten, zijn er nog vragen?