15 april 2024

Jonas Van Hool

Thomas More Hogeschool

De Nayer

Test WEEK 7

Sensors and interfacing

Inhoud

[Hardware (tekst) 3](#_Toc164087728)

[Hardware (foto’s) 5](#_Toc164087729)

**De RC-Boot**

In deze bundel ga ik wat meer uitleg geven over hoe je een RC-boot kan aan- /besturen via BLYNK met behulp van Arduino, meer bepaald de Wemos D1 mini. Hierdoor zal alle terug te vinden code in C++ staan.

Vooraleer we deze code kunnen gebruiken, geef ik eerst wat meer uitleg over de gebruikte componenten, oftewel de hardware.

# Hardware (tekst)

1. Mini D1 (WeMos D1 Mini):

De Mini D1, ook bekend als de WeMos D1 Mini, is een compact ontwikkelingsbord dat draadloze communicatie mogelijk maakt en verbinding kan maken met het internet.

In dit programma fungeert de Mini D1 als de hoofdcontroller die verbinding maakt met het WiFi-netwerk en communiceert met de Blynk IoT-platformserver.

2. Servomotor:

Een servomotor is een gespecialiseerd type motor dat wordt gebruikt voor het positioneren van mechanische systemen. Het kan nauwkeurig worden aangestuurd om een specifieke hoek te bereiken en te behouden.

In dit programma wordt de servo motor gebruikt om de stuurvin van de boot te bedienen. Het ontvangt commando's van de Blynk-app via de Mini D1 en past zijn positie dienovereenkomstig aan.

3. Motor Driver en DC Motor:

Een motor driver is een elektronisch apparaat dat wordt gebruikt om de snelheid en richting van een DC-motor te regelen. Het ontvangt signalen van een microcontroller en levert de benodigde vermogenssignalen aan de motor.

In dit programma wordt de motor driver gebruikt in combinatie met een DC-motor om de voorwaartse en achterwaartse beweging van een boot te regelen. Het ontvangt commando's van de Blynk-app via de Mini D1 en stuurt de motor dienovereenkomstig aan.

4. QMC5883L Kompasmodule:

De QMC5883L is een digitaal kompas dat magnetische velden kan detecteren en de kompasrichting kan bepalen. Het wordt vaak gebruikt in navigatie- en oriëntatie-toepassingen.

In dit programma wordt de QMC5883L kompasmodule gebruikt om de richting van de boot te bepalen. Het leest magnetische gegevens uit en stuurt deze naar de Blynk-app via de Mini D1 voor visualisatie en controle.

5. Blynk IoT-platform:

Blynk is een IoT-platform waarmee gebruikers eenvoudig mobiele apps kunnen maken om hun hardwareapparaten te bedienen en te monitoren. Het biedt een reeks widgets en integratiemogelijkheden voor verschillende hardwareplatforms.

In dit programma wordt de Blynk-app gebruikt als een interface voor de gebruiker om de servo, motor en kompasrichting te bedienen en te monitoren.

Samengevat, de Mini D1 fungeert als de centrale controller die verbinding maakt met het WiFi-netwerk en communiceert met de Blynk-app om de servo, motor en kompasrichting van een boot te regelen en te monitoren. De andere hardwarecomponenten worden gebruikt om de fysieke bewegingen en sensorgegevens van de boot te verwerken.

# Hardware (foto’s)

## Algemeen

## Afbeelding met voertuig, Auto-onderdeel, auto, overdekt Automatisch gegenereerde beschrijvingHieronder zie je de miniD1 (1), de Motordriver (2) en de DC-motor(3)

3

2

1

## Afbeelding met plastic, Elektrisch blauw, blauw, overdekt Automatisch gegenereerde beschrijvingMini D1 + motordriver Dit zijn grotere foto’s van de mini D1 (1) en de motor-Driver (2) Afbeelding met Auto-onderdeel, rood, auto Automatisch gegenereerde beschrijving

2

1

1

1

## Servomotor Afbeelding met machine, Elektrisch blauw, engineering, Elektrische bedrading Automatisch gegenereerde beschrijving

# Aansluitschema

Afbeelding met elektronica, stroomkring, Stroomkringonderdeel, Elektronische engineering

Automatisch gegenereerde beschrijving

# Software

#define BLYNK\_TEMPLATE\_ID    "user3"  // Definieert de Blynk-template ID

#define BLYNK\_TEMPLATE\_NAME  "user3@server.wyns.it"  // Definieert de Blynk-template naam

#include <ESP8266WiFi.h>  // Inclusief de ESP8266WiFi-bibliotheek

#include <BlynkSimpleEsp8266.h>  // Inclusief de BlynkSimpleEsp8266-bibliotheek

#include <Servo.h>  // Inclusief de Servo-bibliotheek

#include <Wire.h> // Toegevoegd voor I2C-communicatie

#include <QMC5883LCompass.h> // Correcte bibliotheekinclusie voor de kompasmodule

char auth[] = "IiLevYfwU0GhCWnCEfNdq55uwuvdeyN";  // Blynk-authenticatietoken

char ssid[] = "Jonas van hool";  // SSID van het WiFi-netwerk

char pass[] = "jonasvanhool";  // Wachtwoord van het WiFi-netwerk

int ENA = 14; // D5 voor motor  // Definieert de pin voor ENA (snelheidsregeling) van de motor

int IN1 = 12; // D6 voor motor  // Definieert de pin voor IN1 (richting) van de motor

int IN2 = 13; // D7 voor motor  // Definieert de pin voor IN2 (richting) van de motor

#define SERVO\_PIN 15 // Pin voor de servo  // Definieert de pin voor de servo

Servo servo;  // Maakt een Servo-object aan

int servoPosition = 90;  // Startpositie voor de servo  // Initialiseert de startpositie voor de servo

QMC5883LCompass compass; // Kompas object  // Instantieert het kompasobject

double heading;   // Variabele om de richting op te slaan  // Variabele om de richting op te slaan

BLYNK\_WRITE(V1) {  // Schuifregelaar-widget in de Blynk-app voor de servo  // Functie voor het verwerken van de schuifregelaarwidget voor de servo

  servoPosition = param.asInt();  // Haalt de waarde van de schuifregelaar op

  servo.write(servoPosition);  // Stelt de positie van de servo in

}

BLYNK\_WRITE(V2) {  // Schuifregelaar-widget in de Blynk-app voor de motor (snelheid)  // Functie voor het verwerken van de schuifregelaarwidget voor de motorsnelheid

  int speed = param.asInt();  // Haalt de waarde van de schuifregelaar op

  int motorSpeed = map(speed, 0, 100, 0, 255); // Mapping van de snelheid naar het bereik [0, 255]

  analogWrite(ENA, motorSpeed);  // Stelt de motorsnelheid in

}

BLYNK\_WRITE(V3) {  // Knoppen-widget in de Blynk-app voor de motor (vooruit/achteruit)  // Functie voor het verwerken van de knoppenwidget voor de motorrichting

  int forwardButton = param.asInt();  // Haalt de waarde van de knoppenwidget op

  if (forwardButton == HIGH) {  // Controleert of de vooruitknop is ingedrukt

    digitalWrite(IN1, HIGH);  // Zet IN1 hoog

    digitalWrite(IN2, LOW);  // Zet IN2 laag

  } else {  // Anders achteruit

    digitalWrite(IN1, LOW);  // Zet IN1 laag

    digitalWrite(IN2, HIGH);  // Zet IN2 hoog

  }

}

void setup() {  // Setup-functie

  pinMode(ENA, OUTPUT);  // Initialiseert ENA als uitgang

  pinMode(IN1, OUTPUT);  // Initialiseert IN1 als uitgang

  pinMode(IN2, OUTPUT);  // Initialiseert IN2 als uitgang

  servo.attach(SERVO\_PIN);  // Koppelt de servo aan de juiste pin

  Blynk.begin(auth, ssid, pass, "server.wyns.it", 8081);  // Initialiseert Blynk

  // Initialiseer de kompasmodule

  Wire.begin(); // Start I2C-bus

  compass.init(); // Initialiseer kompasmodule

  Serial.begin(9600); // Initialiseer seriële communicatie voor debuguitvoer

}

void loop() {  // Loop-functie

  Blynk.run();  // Voert Blynk-taken uit

  // Lees kompasrichting

  compass.read();

  byte a = compass.getAzimuth(); // Leest azimuthhoek (het hoekverschil van een gefixeerd punt)

  char myArray[3];

  compass.getDirection(myArray, a); // Leest richting uit

  Serial.print(myArray[0]); // Druk richting af naar de seriële monitor

  Serial.print(myArray[1]);

  Serial.print(myArray[2]);

  Serial.println();

  Blynk.virtualWrite(V4, a); // Stuurt azimuthhoek naar Blynk-app

  delay(100); // Wacht een korte tijd voordat de volgende meting wordt uitgevoerd

}

# App Design

De smartphone laad niet meer op, dus heb ik maar 1 foto (gsm bleef uitvallen).

De **FW-**button = forward; Als je daarop klikt dan zet je de boot in achteruit modus.

De **Speed** meter geeft 0 (stilstaand) tot 255 (volle snelheid

De **Joystick (**Verticaal**) =** regelen sneheid; onderaan = stilstaand.

De **Joystick (**Horizontaal**) =** sturen; links = links, rechts = rechts, midden = rechtdoor.

De **Gyro-**knop geeft richting v.d. boot weer..

Afbeelding met schermopname, tekst, cirkel, ontwerp

Automatisch gegenereerde beschrijving