

**EDUGO Campus Glorieux**

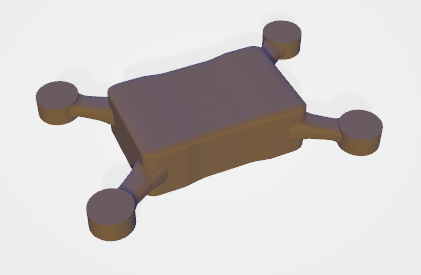
SCHOOL VOOR WETENSCHAP & TECHNIEK

Sint-Jozefstraat 7 9041 Gent-Oostakker

Tel. school : 09/255 91 15

**Verslag geïntegreerde proef**

**Drone**



Naam: *Rune Eerdekens*

Studierichting: *elektronica elektriciteit*

Klas: *6EE*

GIP-begeleiders: *dhr. Eggermond, dhr. Vallozie, dhr. Van Eetvelde*

Schooljaar: *2022-2023*

# Woord vooraf

Dit verslag is opgesteld als deel van onze geïntegreerde proef in het laatste jaar van het middelbaar waarin ik uitleg zal geven over mijn eindejaar project.

Mijn doel was om een GIP te maken waarmee ik mezelf zou kunnen uitdagen om een goed beeld te krijgen van wat ik allemaal kon op het einde van mijn 6de jaar. Daarom heb ik geprobeerd om zo weinig mogelijk het internet te gebruiken, dus veel wan wat u zal lezen zijn mijn eigen conclusies.

Ik wens volgende personen te bedanken voor alle hulp die zij me geboden hebben:

* Het directieteam;
* dhr. Eggermond, dhr. Van Eetvelde, dhr. Vallozie en dhr. Wulleman;
* Mijn vrienden en familie;
* Medeleerlingen;
* Mijn leerkrachten tot nu;

Tekstverwerkingsprogramma: Microsoft Office Word 2022

Printer: HP DeskJet 3762 All-in-One

# Inhoudsopgave

# Inleiding

Mijn naam is Rune Eerdekens ik zit in het 6de jaar Elektronica Elektriciteit en als laatstejaarsstudent kreeg ik samen met mijn klasgenoten de opdracht om een eindwerk te maken en verdedigen.

Het doel van deze proef is om een ontwerp naar jouw keuze te ontwerpen en realiseren dat binnen het vakgebied EE valt, voor mijn project koos ik voor het bouwen van een drone. Ik wist dat dit een grote opdracht ging zijn en heb veel plezier gehad met het bouwen van dit project.

In de loop van het uitdenken en bouwen/programmeren van mijn GIP ben ik verschillende ingewikkelde obstakels tegengekomen en overwonnen. Het was zeker een uitdaging wat het uiteindelijk leuk maakte voor mij.

In deze versie van mijn GIP kan nog veel verbeterd of toegevoegd worden maar daaraan zal ik een uniek hoofdstuk voor voorzien.

Uiteindelijk ben ik blij hoe het is geëindigd ook al had ik liever iets meer tijd gehad om aan uitbreidingen te werken zodat ik die ook zou kunnen presenteren in mijn verdediging.

Wie geïnteresseerd is in alles dat met mijn GIP te maken heeft kan al mijn bestanden zien op GitHub met deze link: https://github.com/RuneEerdekens/GIP

# Praktijk

## Algemeen

Zoals eerder vermeld is mijn GIP een drone in dit hoofdstuk zal ik toelichten hoe hij in elkaar zit in grote lijnen.

## Kern

Zoals eerder vermeld wordt deze drone gestuurd door een zelfontworpen breadboard met 3 componenten. Een Gyroscoop/acceleratiemeter, Arduino Nano, en een HC-05 bluetooth ontvanger

### HC-05 bluetooth antenne

\*foto antenne

Dit zijn de oren van de drone hier ziet u een foto van hoe die eruitziet.

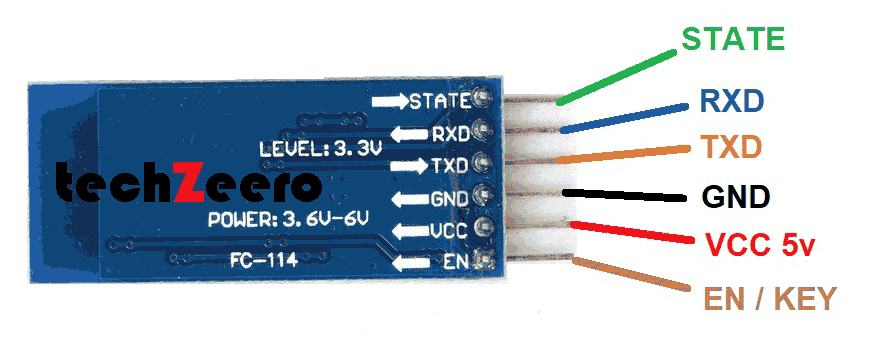


Foto 1: HC-05 Bluetooth module (www.theengineeringprojects.com)

Zoals u ziet op dit schema zijn vier van de zes pinnen verbonden deze zijn de enige die we nodig hebben voor dit project. De vier verbonden pinnen zijn de Vcc, de GND, de TX en de RX pinnen hun functie is als volgt:

* **Vcc**: +3.3 tot +6 Volt komt hiertoe, dit voed de component. In mijn project gebruik ik hiervoor de +5V van de Arduino.
* **GND**: dit is de ground pin deze gaat ook naar de ground van de Arduino
* **TX**: de TX pin is de 1ste van twee pinnen om data te communiceren naar de Arduino, deze word aan de RX pin van de Arduino gekoppeld.
* **RX**: net zoals de TX pin is deze nodig voor het praten met de Arduino deze word dus aan de TX pin gekoppeld.

De HC-05 module heeft nog twee andere pinnen die ik niet gebruik met volgende functies:

* **EN**: de enable pin van de module is actief laag dus als je een positieve spanning op deze pin zet zal de module dus de bluetooth afgezet worden.
* **State**: de state pin is heel handig aangezien deze hoog komt de staan als er een connectie is over bluetooth. Je kan deze gebruiken om in je Arduino code alleen maar dingen aan te zetten als bluetooth verbonden is.

In dit project gebruik ik bluetooth om te communiceren met de drone vanuit een app die op mijn GSM staat. Ook al heeft de HC-05 module de mogelijkheid om data te ontvangen en uit te sturen gebruik ik het alleen maar voor het ontvangen van data aangezien ik geen data terug stuur. Je zou de stuur functie bijvoorbeeld kunnen gebruiken om door te geven aan de gsm hoeveel procent de batterij nog heeft voordat hij plat is.

Er staat een voorbeeld hoe ik de module gebruik in mijn code op pagina X waar.

### MPU 6050 gyroscoop

\*foto gyroscoop

Dit is een foto van de gyroscoop deze component is verantwoordelijk voor het doorgeven van hoeks verandering en versnelling van de drone in alle drie de dimensies.

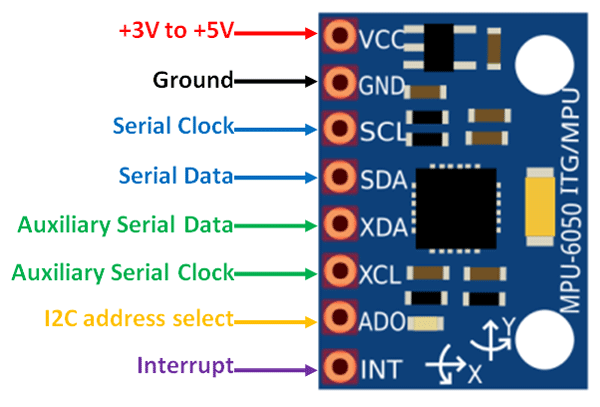


Foto 2: MPU-6050 gyroscoop (components101.com)

De MPU 6050 of gyroscoop heeft acht pinnen in totaal. Van deze acht gebruik ik er maar vier.

De vier pinnen die ik gebruik zijn de volgende:

* **Vcc**: zoals bij de HC-05 word deze gevoed door de Arduino en kan een spanning van +3V tot +5V
* **Ground**: ook zoals bij de HC-05 word deze pin aan de ground van de Arduino gekopeld.
* **SCL**: de seriële klok stuurt een kloksignaal door naar de Arduino zodat de Arduino de data kan inlezen bij de juiste frequentie en word aan de SCL van de Arduino gekoppeld
* **SDA**: de seriële data net zoals de klok is nodig voor het doorsturen van data van de MPU-6050 naar de Arduino en word aan de SDA pin van de Arduino gekoppeld.

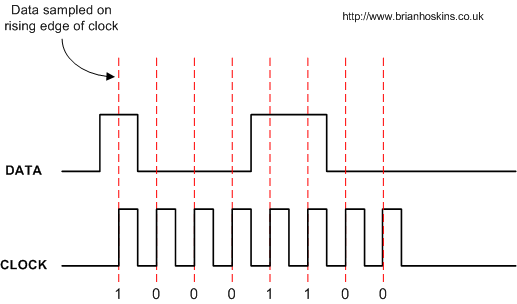


Foto 3: SDA en SCL voor data transfer (electronics.stackexchange.com)

In de foto ziet u dat de data alleen maar word ingelezen op de positieve flank van de klok. Zo wordt er verzekerd dat alles goed verloopt en dat je het juiste aantal bits leest. Als er bijvoorbeeld op de datalijn vier nullen na elkaar worden doorgestuurd kijk je bij elke positieve flank van de klok naar of de datalijn hoog (1) of laag (0) staat en zo kom je te weten hoeveel bits er in die lange lijn van nullen of één’ s zitten.

### Arduino Nano.

Dit is het brijn van de drone, alle wiskundige operaties worden op de Arduino uitgevoerd. Voor dit project gebruik ik een Arduino Nano Every, een variant niet veel verschillend van de normale Arduino Nano.

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving

Foto 4: Arduino Nano Every pin lay-out (wiki.rocrail.net)

Er zijn veel pinnen die niet worden gebruikt in dit project maar er zijn een paar belangrijk. Natuurlijk zijn de +5V en de GND (ground) belangrijk voor het voeden van alle componenten, en ook de TX / RX en SDA / SCL pinnen voor het doorsturen van data. Aangezien de motors van de drone BLDC (brushless DC) motoren zijn hebben we ook nog de PWM (puls width modulation) pinnen nodig om een signaal naar de motors te sturen die de motors begrijpen. Deze zijn aangeduid door een ~ teken naast het pin nummer. Degene die ik gebruik zijn D3, D5, D6 en D9. Er zijn vier PWM pinnen nodig omdat ik vier motors heb waarvan ik de snelheid individueel wil kunnen besturen. Het uploaden van code naar de Arduino wordt gedaan via een micro-USB. De TX en RX pinnen mogen wel niet verbonden zijn of de upload van de code faalt.

### Electronic Speed Controller

De volgende component is de Electronic Speed Controller of ESC in het kort. Hiervan heb ik er vier aangezien ik vier motoren heb.

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Foto 5: pin layout ESC

Zoals je ziet op het schema wordt de ESC gevoed met 7.4V vaan de 2S LiPo batterij. Dit is een soort batterij die vaak in drones wordt gebruikt voor hun energiedichtheid, ze zijn ook her oplaadbaar wat ze een perfecte kandidaat maken voor een Drone. Er zit op deze soort ESC een power regulator die handig is voor een regulated +5V te genereren om de Arduino met +5V te voeden. Dit is ook de component die het PWM signaal ontvangend van de Arduino en het omzet naar de 3 fase van de BLDC motor.

#### Werking PWM

Ik praat hier al een tijdje over PWM maar heb het nog niet duidelijk uitgelecht wat ik hier zal doen.

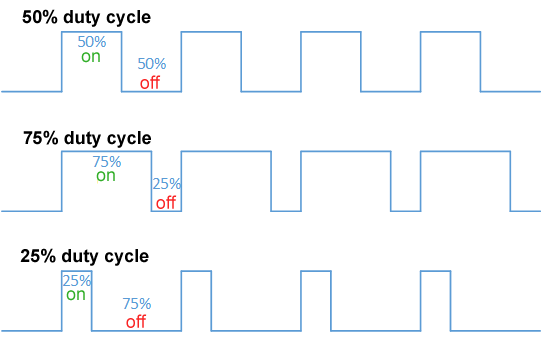


Foto 6: PWM signaal (en.wikipedia.org)

Een PWM signaal is opgebouwd uit pulsen. Deze pulsen zijn altijd een fractie van de duty cycle, als je in je PWM signaal doorstuurt waarbij de puls 100% is van de duty cycle zal de motor op volledige snelheid draaien, bij mijn ESC ligt de minimum puls grote op 700ms en da maximum op 2300ms dus als ik pulsen van 2300ms stuur zullen de motors op 100% snelheid draaien. Voor dit soort signaal te maken gebruik ik in mijn Arduino een bibliotheek die veel hiervoor wordt gebruikt genaamd <Servo.h>. Met deze bibliotheek kan ik de min en max puls grote instellen en dan later via een schaal van 0 tot 180 de grote van mijn pulsen aanpassen. Dus 0 zou een puls grote van 700ms zijn en 180 een grote van 2300ms.

### BLDC motor

Hiervan heb ik er ook vier. De specifieke motor die ik gebruik is de mt2204 2300KV. Ik heb twee die in wijzerzin draaien en twee tegenwijzerzin. 

Foto 7: motors ([www.aliexpress.com](http://www.aliexpress.com))

De reden dat we twee motors tegen wijzerzin en twee in wijzerzin laten draaien is voor 2 redenen

#### Draaien van de drone

Om de drone naar rechts te laten draaien verhogen we de kracht op de wijzenzin motoren

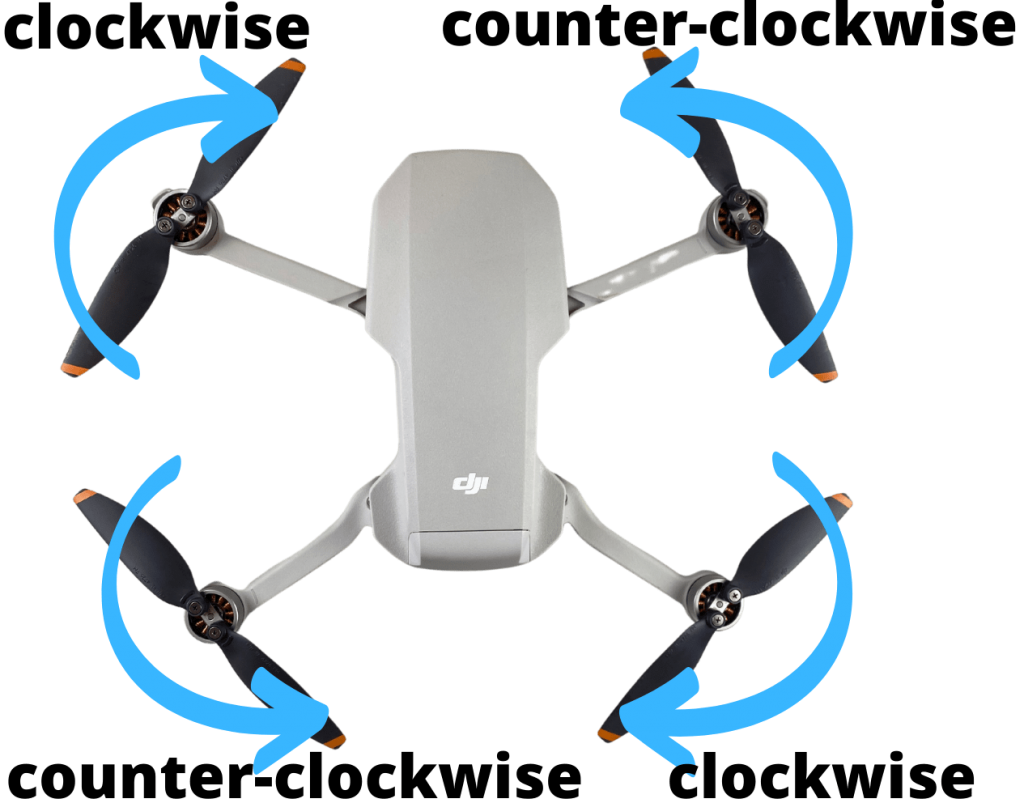


Foto 8: propeller richting (easydrone.com)

Op deze foto zie je het impulsmoment dat elke propeller maakt, het is dus ook belangrijk dat 2 van je propellers het spiegelbeeld zijn van de andere. Anders zou je naast een omgekeerd impulsmoment ook een omgekeerde stootkracht genereren waardoor de drone over kop zou gaan.

#### Stabiliteit

Zoals eerde vermeld is de 2de reden stabiliteit. Als alle vier de motors op gelijke snelheid draaien zullen de impulsmomenten even groot zijn en zal de drone niet draaien. Dit komt omdat alle impulsmomenten elkaar opheffen.

### Hoe vliegt een drone

De hooft reden waardoor een drone vliegt komt toe op een wet van fysica. Newton ’s 3de wet van beweging. Voor elke actie is er een gelijke en tegenovergestelde reactie. Dit is ook de reden waarom als je op een skateboard een zwaar object snel naar links gooit dat je naar rechts beweegt. Of de reden dat raketten de ruimte in kunne vliegen.

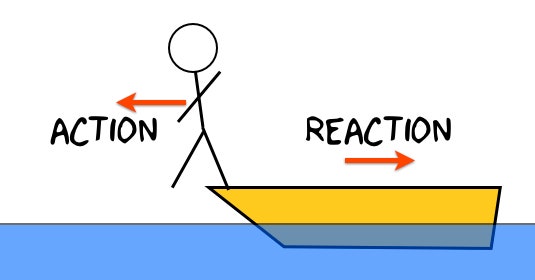


Foto 9: Newton ‘s 3de e wet ([www.wired.com](http://www.wired.com))

Dus dat is de reden waarom een drone vliegt, hij genereert een neerwaartse kracht waardoor hij opstijgt. Maar hoe doet een drone dat? Het antwoord is luchtdruk. De proppelors zuigen de lucht naar achter hun via het “coanda effect”.

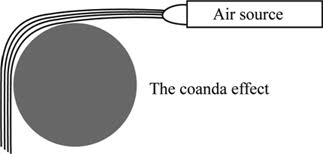


Foto 10: coanda effect ([www.chegg.com](http://www.chegg.com))

Het coanda effect is dus de reden waarom de lucht door onze propellers wordt weggezogen, alleen is het bij een drone de propellers die bergwegen en niet de lucht. Dit is hetzelfde effect dat je ziet als je een lepel verticaal onder de kraan houdt, het water vloeit langs de lepel.

Dus nu dat de lucht verplaatst is komt er een drukverschil boven en onder de propellers. Waardoor er nog meer lucht wordt verplaatst. Omdat dit aan zo een hoge snelheden gebeurt wordt er zoals eerder vermeld een kracht naar benden gegenereerd en vliegt de drone naar boven.

## Software

### Algemeen

Nu dat de hardware uit de weg is is het tijd voor de software, voor dit project heb ik mijn software hoofdzakelijk in twee verschillende talen geschreven. De 1ste is een variant van C++ de taal waarop Arduino zichzelf baseert, dit gebruik ik voor mijn drone zelf te programmeren. De 2de is Dart waarbij ik een framework genaamd flutter gebruik voor de app waarmee ik mijn drone kan besturen.

### Arduino code

Dit is het kortste deel van de twee.

@flowchart

|  |
| --- |
| #include <Servo.h> //importeer de servo biblioteek voor het gemakelijk maken van PWM singaalen  const char SIZE = 5;  unsigned char In[SIZE]; // zet groote voor de input buffer, dus ik verwatch om 5 bytes per keer te ontvangen  unsigned char MotorSpeed[4]; // houd de totaale motor snelheid bij voor elke motor  unsigned char Trothle[4]; // algemeene trothle van de drone dus hoe snel ze draaien  unsigned char Dir1[4]; //dir 1 is for forward back left and right, houd bij hoeveel elke motor bij krijgt als er in een bepaalde richting bewogen word  Servo ESC1;  Servo ESC2;  Servo ESC3;  Servo ESC4; //initializeer alle ESC's (Electronic Speed Controller's)  int i = 0; //iterator  void setup() {  Serial.begin(9600);  Serial1.begin(9600); // initializeer hardware en software Serieele comunicaties  ESC1.attach(3, 700, 2300);  ESC2.attach(5, 700, 2300);  ESC3.attach(6, 700, 2300);  ESC4.attach(9, 700, 2300); // assing pinnummer, min pulsgrote voor pwm, max pulsegroote voor pwm  }  void loop() {  if (Serial1.available() > 0) {  int len = Serial1.readBytes(In, SIZE); //lees "SIZE" (SIZE = 5) bytes per keer en sla ze op in de "In" lijst  if (In[0] == 0xF2) { // 1ste byte is signal byte die aantoont welke soort data het is  for (i = 1; i < 5; i++) {  Trothle[i - 1] = In[i]; // in dit geval is het de trothel data dus set de trothel lijst naar de nieuwe waarden  }  }  if (In[0] == 0xF0) { // zelfde als hierboven dit is True als de signal byte gelijk is aan 0xF0 of 240  for (i = 1; i < 5; i++) {  Dir1[i - 1] = In[i]; // zet de Dir1 lijst gelijk aan de nieuwe waarden  }  }  for (int j = 0; j < 4; j++) { //voor elke motor word de totaal snelheid gelijkgezet aan de Trothle + Dir  MotorSpeed[j] = Trothle[j] + Dir1[j];  Serial.print(j);//prints voor debugig  Serial.print(": ");  Serial.println(MotorSpeed[j], DEC);  }  ESC1.write(MotorSpeed[0]);  ESC2.write(MotorSpeed[1]);  ESC3.write(MotorSpeed[2]);  ESC4.write(MotorSpeed[3]); // schrijf alle motorsnelheeden naar de ESC's min waarde is 0 en max is 180  // dus de motors draaien op 100% snelheid als motorspeed voor die motor gelijk is aan 180  }  } |

### Flutter

In flutter is bijna alles een widget, je bouwt een app door deze te combineren. Het voordeel is het is gemakkelijk en alles legt zichzelf uit. Het nadeel is dat je in een Hell van genestelde widgets terecht komt.

|  |
| --- |
| import 'package:flutter/services.dart';  import 'package:flutter/material.dart';  import 'package:gip\_app/bt\_screen.dart';  import 'test\_functions\_screen.dart';  import 'package:gip\_app/stick.dart';  import 'bluetooth\_connection.dart';  void main() {    runApp(const MyApp());    SystemChrome.setPreferredOrientations([      DeviceOrientation.landscapeRight,      DeviceOrientation.landscapeLeft,    ]);  }  double sliderVal = 0;  class GlobalData{    void setVal(double val){      sliderVal = val;    }    double getVal(){      return sliderVal;    }  }  class MyApp extends StatelessWidget {    const MyApp({super.key});    @override    Widget build(BuildContext context) {      return MaterialApp(        debugShowCheckedModeBanner: false,        theme: ThemeData(primarySwatch: Colors.lime),        home: const RootePage(),      );    }  }  class RootePage extends StatefulWidget {    const RootePage({super.key});    @override    State<RootePage> createState() => \_RootePageState();  }  class \_RootePageState extends State<RootePage> {    BluetoothConnectionManager bluetoothManager = BluetoothConnectionManager();    int updateDelayMs = 100;    @override    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(          title: const Text("Main page"),        ),        body: Stack(          children: [            Stack(              children: [                Align(                  alignment: AlignmentDirectional.center,                  child: Container(                    margin: const EdgeInsets.only(left: 400, top: 40),                    child: Stick(                      bluetoothManager: bluetoothManager,                      sig: 0xF0,                      updateTimeMs: updateDelayMs,                    ),                  ),                ),                Align(                  alignment: AlignmentDirectional.center,                  child: Container(                    margin: const EdgeInsets.only(right: 400, top: 40),                    child: Stick(                      text: "Up/down, Left/Right",                      bluetoothManager: bluetoothManager,                      sig: 0xF1,                      updateTimeMs: updateDelayMs,                    ),                  ),                ),              ],            ),            Align(              alignment: AlignmentDirectional.bottomCenter,              child: Container(                margin: const EdgeInsets.only(top: 100),                child: Column(                  children: [                    Container(                      padding: const EdgeInsets.only(),                      child: FloatingActionButton(                        onPressed: () {                          Navigator.push(                            context,                            MaterialPageRoute(                              builder: (context) => TestFunctions(                                bluetoothManager: bluetoothManager,                                sig: 0xF2,                              ),                            ),                          );                        },                        child: const Icon(Icons.arrow\_upward\_rounded),                      ),                    ),                    Container(                      padding: const EdgeInsets.only(top: 20),                      child: FloatingActionButton(                        onPressed: () {                          Navigator.push(                            context,                            MaterialPageRoute(                              builder: (context) => BtScreen(                                bluetoothManager: bluetoothManager,                              ),                            ),                          );                        },                        child: const Icon(Icons.bluetooth),                      ),                    ),                  ],                ),              ),            ),          ],        ),      );    }  } |

Dit is de main File deze word geladen de app opstart

|  |
| --- |
| import 'dart:typed\_data';  import 'package:flutter/material.dart';  import 'bluetooth\_connection.dart';  import 'package:flutter\_bluetooth\_serial/flutter\_bluetooth\_serial.dart';  class BtScreen extends StatefulWidget {    final BluetoothConnectionManager bluetoothManager;    const BtScreen({super.key, required this.bluetoothManager});    @override    State<BtScreen> createState() => \_BtScreenState();  }  class \_BtScreenState extends State<BtScreen> {    List<BluetoothDevice> devices = [];    BluetoothDevice? selectedDevice;    @override    void initState() {      super.initState();      \_getBondedDevices();    }    Future<void> \_sendMessage(Uint8List val) async {      await widget.bluetoothManager.sendMessage(val);    }    Future<void> \_getBondedDevices() async {      try {        devices = await widget.bluetoothManager.getBondedDevices();      } catch (ex) {        print('Error getting bonded devices: $ex');      }      setState(() {});    }    Future<void> \_connectToDevice() async {      if (selectedDevice == null) {        print('No device selected');        return;      }      try {        await widget.bluetoothManager.connect(selectedDevice!);        if (widget.bluetoothManager.connection.isConnected) {          \_sendMessage(Uint8List.fromList([0xF2, 162, 162, 162, 162]));          await Future.delayed(const Duration(milliseconds: 2500));          for (var i = 162; i >= 1; i--) {              \_sendMessage(Uint8List.fromList([0xF2, i, i, i, i]));          }        }      } catch (ex) {        print('Error connecting to device: $ex');      }    }    Future<void> \_disconnectFromDevice() async {      \_sendMessage(Uint8List.fromList([0xF2, 0, 0, 0, 0]));      await Future.delayed(const Duration(milliseconds: 500));      await widget.bluetoothManager.disconnect();    }    @override    Widget build(BuildContext context) {      return Scaffold(        body: Center(          child: Column(            mainAxisAlignment: MainAxisAlignment.center,            children: <Widget>[              const Text('Paired devices:'),              DropdownButton<BluetoothDevice>(                value: selectedDevice,                items: devices.map((device) {                  return DropdownMenuItem<BluetoothDevice>(                    value: device,                    child: Text(device.name ?? 'unkown device'),                  );                }).toList(),                onChanged: (value) {                  setState(() {                    selectedDevice = value;                  });                },              ),              ElevatedButton(                  onPressed: () => \_connectToDevice(),                  child: const Text("Connect")),              ElevatedButton(                  onPressed: () => \_disconnectFromDevice(),                  child: const Text("Disconnect")),              ElevatedButton(                  onPressed: () {                    Navigator.pop(context);                  },                  child: const Icon(Icons.home))            ],          ),        ),      );    }  } |

Dit is het scherm met de Bluetooth knoppen op

|  |
| --- |
| import 'dart:async';  import 'dart:typed\_data';  import 'package:flutter/material.dart';  import 'package:flutter\_joystick/flutter\_joystick.dart';  import 'bluetooth\_connection.dart';  import 'package:gip\_app/stick\_base.dart';  import 'package:gip\_app/stick\_handle.dart';  import 'package:gip\_app/stick\_data.dart';  double x = 0, y = 0;  class Stick extends StatefulWidget {    final String? text;    final int sig;    final BluetoothConnectionManager bluetoothManager;    final int updateTimeMs;    const Stick(        {super.key,        this.text,        required this.bluetoothManager,        required this.sig,        required this.updateTimeMs});    @override    State<Stick> createState() => \_StickState();  }  class \_StickState extends State<Stick> {    Future<void> \_sendMessage(Uint8List val) async {      await widget.bluetoothManager.sendMessage(val);    }    @override    Widget build(BuildContext context) {      return Column(        children: [          Container(            margin: const EdgeInsets.only(bottom: 20),            child: Text(                widget.text ??                    "X: ${x.toStringAsFixed(2)} Y: ${y.toStringAsFixed(2)}",                style:                    const TextStyle(fontSize: 15, fontWeight: FontWeight.bold)),          ),          Joystick(            base: const StickBase(),            stick: const StickHandle(),            period: Duration(milliseconds: widget.updateTimeMs),            onStickDragEnd: () {              \_sendMessage(format255(0, 0, 30, widget.sig));            },            listener: (details) {              setState(() {                x = details.x;                y = details.y;                \_sendMessage(format255(x, y, 30, widget.sig));              });            },          )        ],      );    }  } |

Dit is een template voor de joysticks in de app

|  |
| --- |
| import 'package:flutter/material.dart';  class StickBase extends StatelessWidget {    const StickBase({super.key});    @override    Widget build(BuildContext context) {      return Container(        alignment: Alignment.center,        width: 160,        height: 160,        decoration:            const BoxDecoration(shape: BoxShape.circle, color: Colors.lightBlue),      );    }  } |

Dit is de grote cirkel rond de joystick

|  |
| --- |
| import 'package:flutter/material.dart';  class StickHandle extends StatelessWidget {    const StickHandle({super.key});    @override    Widget build(BuildContext context) {      return Container(          alignment: Alignment.center,          width: 40,          height: 40,          decoration:              const BoxDecoration(shape: BoxShape.circle, color: Colors.amber));    }  } |

En dit is de kleine

|  |
| --- |
| import 'dart:math';  import 'dart:typed\_data';  import 'package:flutter/foundation.dart';  import 'package:flutter/material.dart';  Uint8List format255(double x, double y, double scalar, int signal) {    Uint8List vals = Uint8List(5);    double dm1 = 0,        dm2 = 0,        dm3 = 0,        dm4 = 0; //distance to Motor (motor in corner)    double pDistX = 0,        nDistX = 0,        pDistY = 0,        nDistY = 0; //distance along X and Y acces of current point    pDistX = 1 - x;    nDistX = 2 - pDistX;    pDistY = 1 - y;    nDistY = 2 - pDistY;    pDistX.clamp(1, 2);    nDistX.clamp(1, 2);    pDistY.clamp(1, 2);    nDistY.clamp(1, 2);    dm1 = sqrt(pow(nDistX, 2) + pow(pDistY, 2)) - sqrt(2); //pytagoras    dm2 = sqrt(pow(pDistX, 2) + pow(pDistY, 2)) - sqrt(2);    dm3 = sqrt(pow(nDistX, 2) + pow(nDistY, 2)) - sqrt(2);    dm4 = sqrt(pow(pDistX, 2) + pow(nDistY, 2)) - sqrt(2);    vals[0] = signal;    vals[1] = (dm1 \* scalar).floor();    vals[2] = (dm2 \* scalar).floor();    vals[3] = (dm3 \* scalar).floor();    vals[4] = (dm4 \* scalar).floor();    return vals;  }  Uint8List ud255(double y, double scalar, {int signal = 0xF3}) {    Uint8List vals = Uint8List(5);    vals[0] = signal;    vals[1] = ((y + 1) \* scalar).floor();    vals[2] = vals[1];    vals[3] = vals[1];    vals[4] = vals[1];    return vals;  }  Uint8List lr255(double x, double scalar, {int signal = 0xF4}) {    return Uint8List(1);  } |

Dit is de code die de stick data formatteert naar bruikbare waarden

|  |
| --- |
| import 'dart:typed\_data';  import 'dart:async';  import 'package:gip\_app/main.dart';  import 'package:flutter/material.dart';  import 'bluetooth\_connection.dart';  class TestFunctions extends StatefulWidget {    final BluetoothConnectionManager bluetoothManager;    final int sig;    const TestFunctions(        {super.key, required this.bluetoothManager, required this.sig});    @override    State<TestFunctions> createState() => \_TestFunctionsState();  }  class \_TestFunctionsState extends State<TestFunctions> {    Uint8List \_formatData(double inputVal, int signalVal) {      int newInpVal = inputVal.floor();      return Uint8List.fromList(          [signalVal, newInpVal, newInpVal, newInpVal, newInpVal]);    }    Future<void> \_sendMessage(Uint8List val) async {      await widget.bluetoothManager.sendMessage(val);    }    double sliderVal = GlobalData().getVal();    bool canSend = true;    @override    Widget build(BuildContext context) {      return Scaffold(        appBar: AppBar(          leading: IconButton(            icon: const Icon(Icons.arrow\_upward),            onPressed: () {              Navigator.pop(context);              GlobalData().setVal(sliderVal);            },          ),          title: const Text("Test functions"),        ),        body: Align(          alignment: AlignmentDirectional.center,          child: Column(            children: [              Container(                padding: const EdgeInsets.only(top: 60),                child: Text(                  sliderVal.round().toString(),                  style:                      const TextStyle(fontSize: 40, fontWeight: FontWeight.bold),                ),              ),              Align(                alignment: AlignmentDirectional.center,                child: Container(                  margin: const EdgeInsets.only(top: 50),                  child: Slider(                    max: 120,                    value: sliderVal,                    onChanged: (double val) {                      if (canSend) {                        \_sendMessage(                          \_formatData(val, widget.sig),                        );                        canSend = false;                        Timer(const Duration(milliseconds: 50), () {                          canSend = true;                        });                      }                      setState(() {                        sliderVal = val;                      });                    },                  ),                ),              ),            ],          ),        ),      );    }  } |

Dit zijn de test functies voor het testen van de drone

|  |
| --- |
| import 'dart:typed\_data';  import 'package:flutter\_bluetooth\_serial/flutter\_bluetooth\_serial.dart';  class BluetoothConnectionManager {    late BluetoothConnection connection;    Future<List<BluetoothDevice>> getBondedDevices() async {      List<BluetoothDevice> devices = [];      try {        List<BluetoothDevice> bondedDevices =            await FlutterBluetoothSerial.instance.getBondedDevices();        devices = bondedDevices;      } catch (ex) {        print('Error getting bonded devices: $ex');      }      return devices;    }    Future<void> connect(BluetoothDevice device) async {      try {        BluetoothConnection newConnection =            await BluetoothConnection.toAddress(device.address);        connection = newConnection;        print('Connected to ${device.name}');      } catch (ex) {        print('Error connecting to device: $ex');      }    }    Future<void> sendMessage(Uint8List message) async {      try {        connection.output.add(message);        await connection.output.allSent;        print('Sent message: $message');      } catch (ex) {        print('Error sending message: $ex');      }    }    Future<void> disconnect() async {      try {        await connection.finish();        print('Disconnected from device');      } catch (ex) {        print('Error disconnecting from device: $ex');      }    }  } |

Dit is een template voor de bluetooth manager die eerder voorkwam. Ik gebruik hiervoor een class omdat ik dezelve instantie van de bluetooth manager wil gebruiken over verschillende schermen.

|  |
| --- |
| name: gip\_app  description: A new Flutter project.  # The following line prevents the package from being accidentally published to  # pub.dev using `flutter pub publish`. This is preferred for private packages.  publish\_to: 'none' # Remove this line if you wish to publish to pub.dev  # The following defines the version and build number for your application.  # A version number is three numbers separated by dots, like 1.2.43  # followed by an optional build number separated by a +.  # Both the version and the builder number may be overridden in flutter  # build by specifying --build-name and --build-number, respectively.  # In Android, build-name is used as versionName while build-number used as versionCode.  # Read more about Android versioning at https://developer.android.com/studio/publish/versioning  # In iOS, build-name is used as CFBundleShortVersionString while build-number is used as CFBundleVersion.  # Read more about iOS versioning at  # https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html  # In Windows, build-name is used as the major, minor, and patch parts  # of the product and file versions while build-number is used as the build suffix.  version: 1.0.0+1  environment:    sdk: '>=2.19.4 <3.0.0'  # Dependencies specify other packages that your package needs in order to work.  # To automatically upgrade your package dependencies to the latest versions  # consider running `flutter pub upgrade --major-versions`. Alternatively,  # dependencies can be manually updated by changing the version numbers below to  # the latest version available on pub.dev. To see which dependencies have newer  # versions available, run `flutter pub outdated`.  dependencies:    flutter:      sdk: flutter      # The following adds the Cupertino Icons font to your application.    # Use with the CupertinoIcons class for iOS style icons.    cupertino\_icons: ^1.0.2    flutter\_joystick: ^0.0.3    flutter\_bluetooth\_serial: ^0.4.0    tuple: ^2.0.1    shared\_preferences: ^2.1.1  dev\_dependencies:    flutter\_test:      sdk: flutter    # The "flutter\_lints" package below contains a set of recommended lints to    # encourage good coding practices. The lint set provided by the package is    # activated in the `analysis\_options.yaml` file located at the root of your    # package. See that file for information about deactivating specific lint    # rules and activating additional ones.    flutter\_lints: ^2.0.0  # For information on the generic Dart part of this file, see the  # following page: https://dart.dev/tools/pub/pubspec  # The following section is specific to Flutter packages.  flutter:    # The following line ensures that the Material Icons font is    # included with your application, so that you can use the icons in    # the material Icons class.    uses-material-design: true    # To add assets to your application, add an assets section, like this:    # assets:    #   - images/a\_dot\_burr.jpeg    #   - images/a\_dot\_ham.jpeg    # An image asset can refer to one or more resolution-specific "variants", see    # https://flutter.dev/assets-and-images/#resolution-aware    # For details regarding adding assets from package dependencies, see    # https://flutter.dev/assets-and-images/#from-packages    # To add custom fonts to your application, add a fonts section here,    # in this "flutter" section. Each entry in this list should have a    # "family" key with the font family name, and a "fonts" key with a    # list giving the asset and other descriptors for the font. For    # example:    # fonts:    #   - family: Schyler    #     fonts:    #       - asset: fonts/Schyler-Regular.ttf    #       - asset: fonts/Schyler-Italic.ttf    #         style: italic    #   - family: Trajan Pro    #     fonts:    #       - asset: fonts/TrajanPro.ttf    #       - asset: fonts/TrajanPro\_Bold.ttf    #         weight: 700    #    # For details regarding fonts from package dependencies,    # see https://flutter.dev/custom-fonts/#from-packages |

Dit zijn alle packeten die mijn app nodig heeft om te werken.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Dit is de project structuur, er zijn nog veel bestanden niet getoond maar buiten wat er in de code staat zijn de enige belangrijke dingen nog dat de min versie van de Android SDK v19 is



En dat de bluetooth permissies ingesteld worden.

Afbeelding met schermopname, tekst, cirkel, Kleurrijkheid

Automatisch gegenereerde beschrijving

Dit is een foto van het hooft scherm van de app, de linkse en rechtse blauwe cirkels met gele cirkels zijn twee joysticks die je kan gebruiken voor de drone te controleren. De linke is voor naar boven en onder te bewegen of naar links of rechts te draaien, en de rechtse is voor naar voor, achter, links of rechts te bewegen.

Afbeelding met tekst, schermopname, lijn

Automatisch gegenereerde beschrijving

Je beland op deze pagina als je op het pijltje naar boven duwt, dit is een pagina voor dingen te testen. Nu staat er een slider op die de motors van 0% kracht naar 66% laten gaan (120/180).

Afbeelding met tekst, schermopname, software, Besturingssysteem

Automatisch gegenereerde beschrijving

Dit is de laatste pagina en hier kom je door op het Bluetooth icon te duwen in de hooft pagina. Hier kan je verbinden met de drone via bluetooth.