

**EDUGO Campus Glorieux**

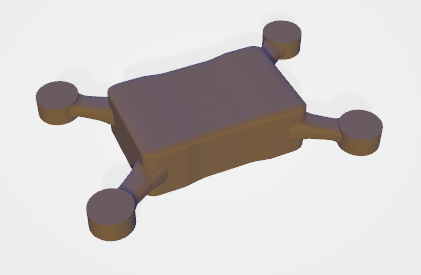
SCHOOL VOOR WETENSCHAP & TECHNIEK

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. Eggermont, 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. Eggermont, 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

Inhoud

[Woord vooraf 2](#_Toc136525964)

[Inhoudsopgave 3](#_Toc136525965)

[Inleiding 5](#_Toc136525966)

[1 Praktijk 6](#_Toc136525967)

[1.1 Algemeen 6](#_Toc136525968)

[1.2 Kern 6](#_Toc136525969)

[1.2.1 HC-05 bluetooth antenne 6](#_Toc136525970)

[1.2.2 MPU 6050 gyroscoop 7](#_Toc136525971)

[1.2.3 Arduino Nano. 9](#_Toc136525972)

[1.2.4 Electronic Speed Controller 10](#_Toc136525973)

[1.2.5 BLDC motor 11](#_Toc136525974)

[1.2.6 Hoe vliegt een drone 13](#_Toc136525975)

[1.2.7 veiligheid 14](#_Toc136525976)

[1.3 Software 14](#_Toc136525977)

[1.3.1 Algemeen 14](#_Toc136525978)

[1.3.2 Arduino code 14](#_Toc136525979)

[1.3.3 Flutter 17](#_Toc136525980)

[2 Uitbreiding 35](#_Toc136525981)

[2.1 Algemeen 35](#_Toc136525982)

[2.2 Gyroscoop / versnellingsmeter 35](#_Toc136525983)

[2.2.1 PID algoritme 35](#_Toc136525984)

[2.3 Camera 36](#_Toc136525985)

[2.4 FPV 37](#_Toc136525986)

[3 Reflectie 38](#_Toc136525987)

[4 Bibliografie 39](#_Toc136525988)

# 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 eigen keuze te ontwerpen en te 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 die ik heb 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

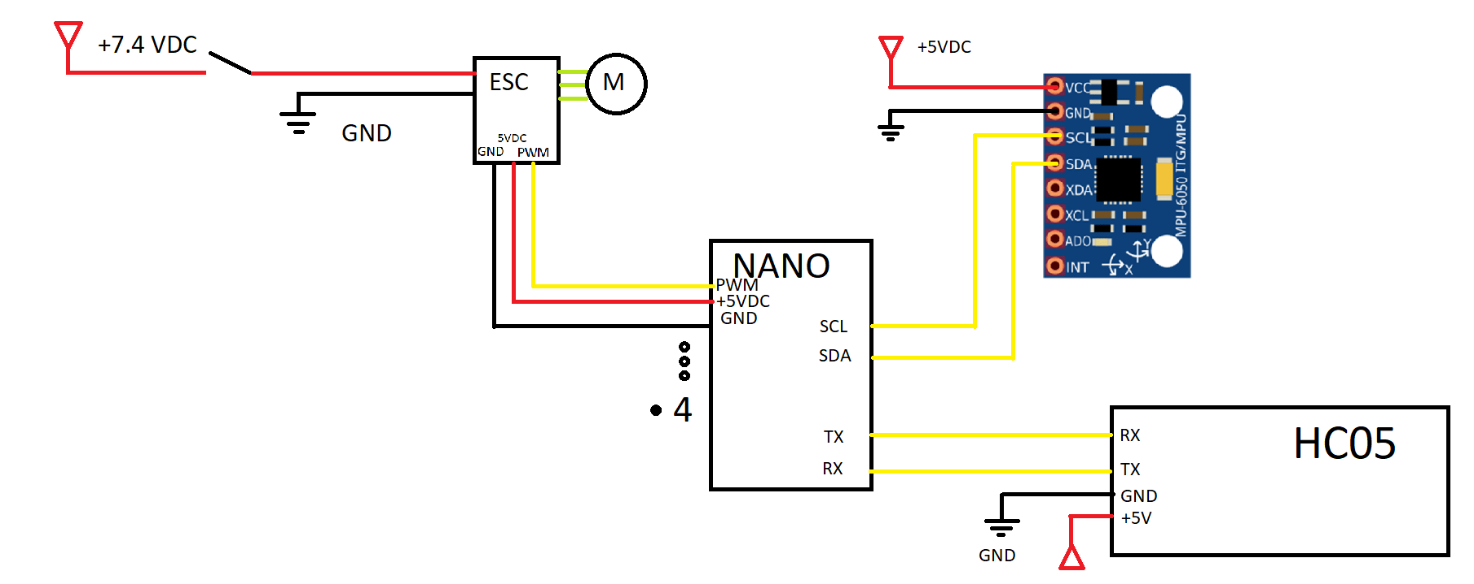
# Praktijk

## Algemeen

Zoals eerder vermeld is mijn GIP een drone bouwen. 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



Dit is een foto van het blockschema van de drone

### HC-05 bluetooth antenne

Afbeelding met stroomkring, Elektronisch onderdeel, Stroomkringonderdeel, Passief stroomkringonderdeel

Automatisch gegenereerde beschrijvingDit zijn de oren van de drone. Hier ziet u een foto van hoe die eruitziet.

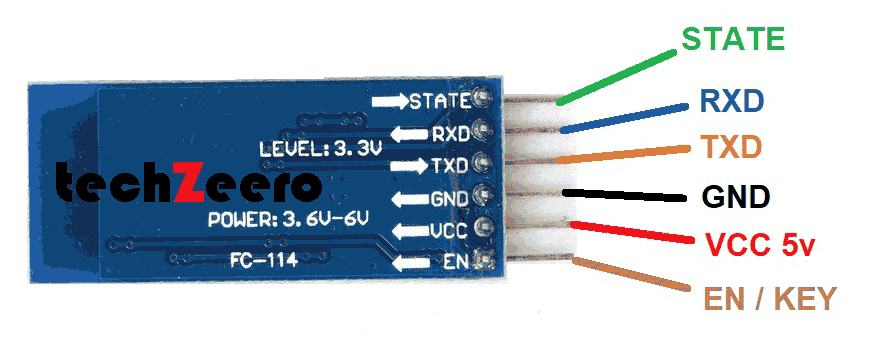


Foto 1: HC-05 Bluetooth module (www.theengineeringprojects.com)

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 voedt 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 wordt aan de RX pin van de Arduino gekoppeld.S
* **RX**: net zoals de TX pin is deze nodig voor het praten met de Arduino deze wordt dus aan de TX pin gekoppeld.

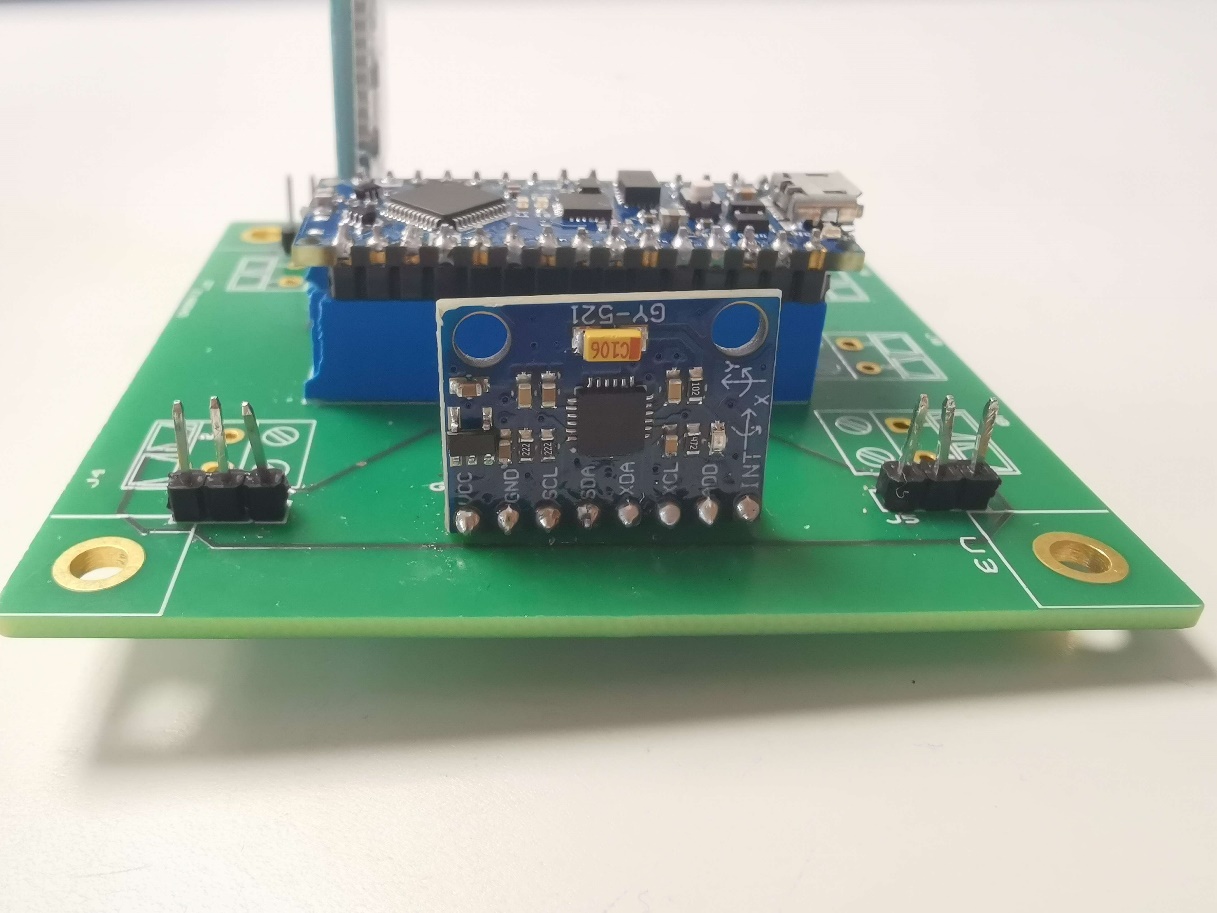
De HC-05 module heeft nog twee andere pinnen die ik niet gebruik met volgende functies:

* **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



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.

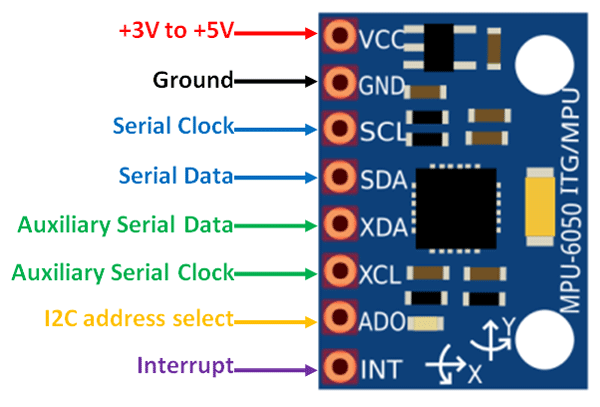


Foto 2: MPU-6050 gyroscoop (components101.com)

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 wordt deze gevoed door de Arduino en kan een spanning van +3V tot +5V
* **Ground**: ook zoals bij de HC-05 wordt deze pin aan de ground van de Arduino gekoppeld.
* **SCL**: de seriële klok stuurt een kloksignaal door naar de Arduino zodat de Arduino de data kan inlezen bij de juiste frequentie en wordt 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 wordt aan de SDA pin van de Arduino gekoppeld.

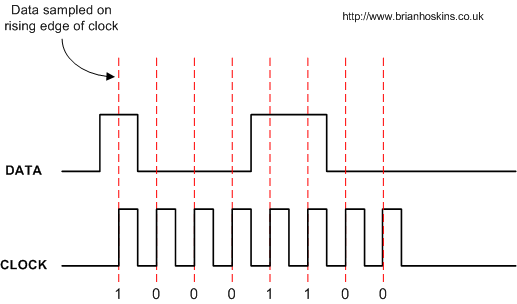


Foto 3: SDA en SCL voor data transfer (electronics.stackexchange.com)

In de foto ziet u dat de data alleen maar wordt 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 brein 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 belangrijke. 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 motoren van de drone BLDC (brushless DC) motoren zijn, hebben we ook nog de PWM (puls width modulation) pinnen nodig om een signaal naar de motoren te sturen die de motoren 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 zeker 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 van de 2S LiPo batterij. Dit is een soort batterij die vaak in drones wordt gebruikt voor hun energiedichtheid, ze zijn ook heroplaadbaar wat ze perfect geschikt 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 uitgelegd wat ik hier zal doen.

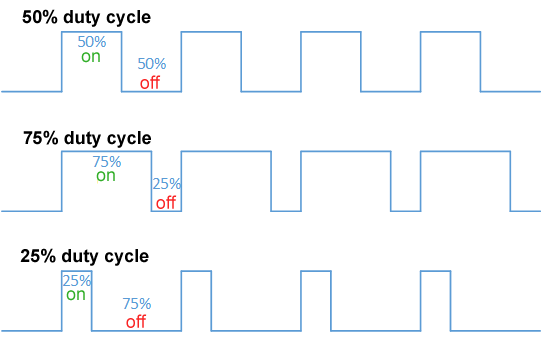


Foto 6: PWM signaal (en.wikipedia.org)

Een PWM signaal is opgebouwd uit pulsen. Deze pulsen zijn altijd een fractie van de duty cycle. Als je in je PWM een 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 grootte op 700ms en dat maximum op 2300ms dus als ik pulsen van 2300ms stuur, zullen de motoren op 100% snelheid draaien. Om 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 grootte instellen en dan later via een schaal van 0 tot 180 de grootte van mijn pulsen aanpassen. Dus 0 zou een puls grootte van 700ms zijn en 180 een grootte van 2300ms.

### BLDC motor

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

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

De reden dat we twee motoren tegenwijzerzin en twee in wijzerzin laten draaien, heeft twee redenen:

#### Draaien van de drone

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

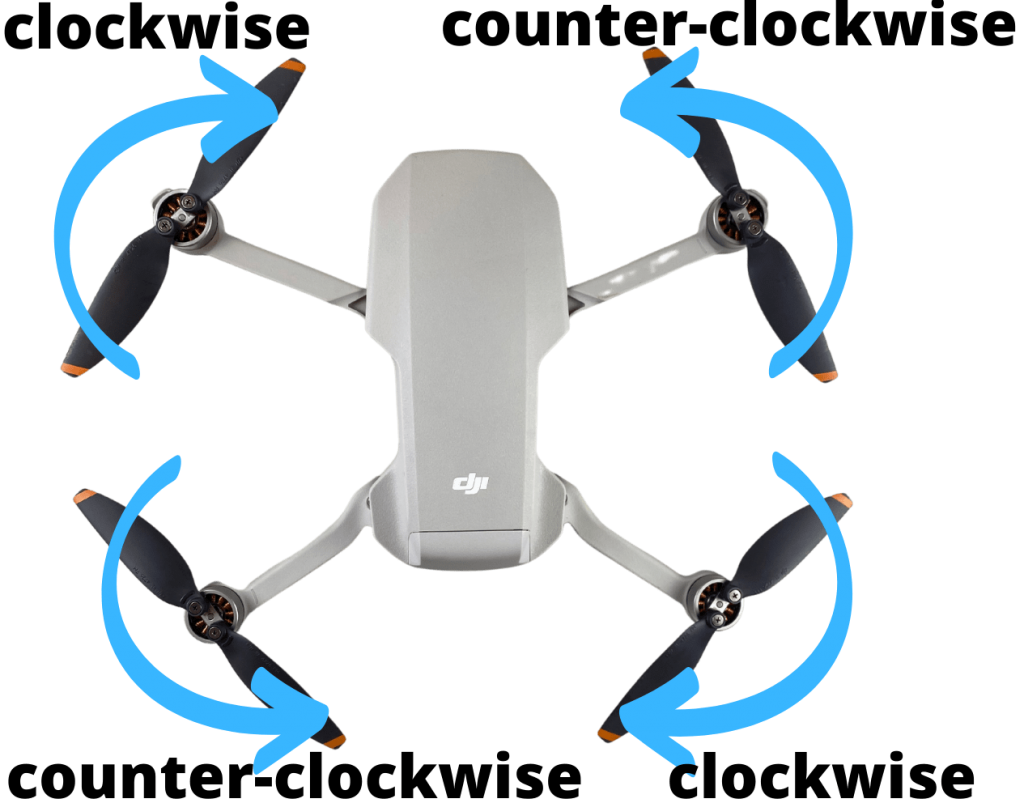


Foto 8: propeller richting (easydrone.com)

Op deze foto zie je het impulsmoment dat elke propeller maakt, het is dus ook belangrijk dat twee 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 eerder vermeld is de tweede reden stabiliteit. Als alle vier de motoren 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 hoofdreden waardoor een drone vliegt, komt uit een wet van fysica, namelijk 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 kunnen vliegen.

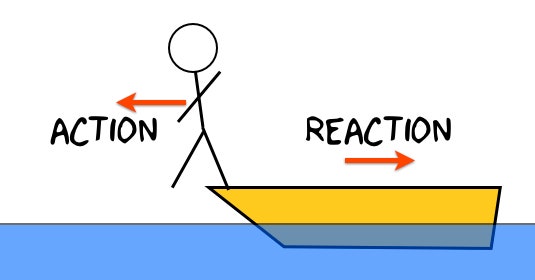


Foto 9: Newton ‘s 3de wet ([www.wired.com](http://www.wired.com))

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 “bernoulli effect”.

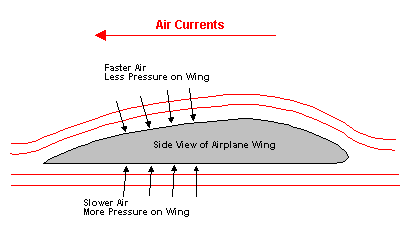


Foto 10: bernoulli effect (encyclopedia2.thefreedictionary.com)

Het bernoulli effect is dus de reden waarom de lucht door onze propellers wordt weggezogen, bij een normaal vliegtuig staat de vleugel stil en beweegt het vliegtuig. Daarentegen bij een drone bewegen de “vleugels” (propellers) en de drone beide.

Dit veroorzaakt een drukverschil boven en onder de drone en zoals u weet heeft lucht de neiging om van hoge naar lage druk te bewegen. Dus de lucht beweegt naar benden wat een kracht genereerd die de drone omhoog duwt.

### veiligheid

naar veiligheid toe is mijn GIP redelijk veilig maar er zijn een paar dingen die je niet mag doen:

* de propellers aanraken als ze draaien, dit is gezond verstand maar er is geen beveilig ingebouwd

als ik dit zou beveiligen zou ik een rand invoegen in de omhulling van mijn drone die de radius van de propellers duidelijk is aangeduid. Ik zou mogelijks ook een sensor kunnen toevoegen die de stroom op de motors meet, als iemand zijn hand er tussen zou zitten zou de motor grote weerstand ondervinden waardoor de stroom zou stijgen. Als de stoom te hoog word zou ik de motor uit zetten.

* De drone mag niet uitgezet worden als de drone nog vliegt. Ook al zou dit moeilijk zijn aangezien je de schakelaar niet zou kunnen bereiken als de drone in de lucht aan hey vliegen is. Maar als dit lukt gebeuren er slechte dingen de schaaklaar zou overslaan omdat je opeens een stroom van 20 A opeens uit schakelt (20 A is max). om dit op te lossen zou ik of een groter schakelaar kunnen zetten die de 20A aan kan of op elke motor een terugkoppel diode steken die deze piekstoom zou kunnen herleiden.
* de snelheid van de drone redelijk veilig de max snelheid is in software ingesteld op maar ongeveer 65%. En als je je signaal manueel ontkoppeld dan word de motorsnelheid waar naar 0 gezet. Maar als dit spontaan gebeurd door storing of als je verder dan ongeveer 20 meter komt en je het signaal verliest bestaat de kans dat de drone verder zou blijven opstijgen. Ik zou dit oplossen door de State pin van de HC-05 module te verbinden aan een digitale pin van de Arduino en te kijken of deze Low word. In dit geval zet ik de motorsnelheid vanuit de Arduino zelf weer naar 0 want dit betelen dat het signaal verloren is. Hopelijk verbind de drone dat terug binnen de tijd dat de drone naar benden valt. Dan zal de laatste motorsnelheid die ontvangen werd weer toegepast worden op de motors.

Dit zijn de 3 hoofdzakelijke dingen waarvoor je moet opletten en hoe ik ze zou oplossen. Ik zou dit ook hebben gedaan met meer tijd.

## Software

### Algemeen

Nadat de hardware besproken werd, is het tijd voor de software. Voor dit project heb ik mijn software hoofdzakelijk in twee verschillende talen geschreven. De eerste is een variant van C++ de taal waarop Arduino zichzelf baseert. Dit gebruik ik om mijn drone zelf te programmeren. De tweede 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 gemakkelijk maken van PWM signalen  const char SIZE = 5;  unsigned char In[SIZE]; // zet grootte voor de input buffer, dus ik verwacht om 5 bytes per keer te ontvangen  unsigned char MotorSpeed[4]; // houd de totale motor snelheid bij voor elke motor  unsigned char Trothle[4]; // algemene 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 Seriële communicatie  ESC1.attach(3, 700, 2300);  ESC2.attach(5, 700, 2300);  ESC3.attach(6, 700, 2300);  ESC4.attach(9, 700, 2300); // assing pinnummer, min puls grootte voor pwm, max puls grootte 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 trothle data dus set de trothle 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() { // main routine start de app en zet gsm in landchap modus    runApp(const MyApp());    SystemChrome.setPreferredOrientations([      DeviceOrientation.landscapeRight,      DeviceOrientation.landscapeLeft,    ]);  }  double sliderVal = 0;  class GlobalData{ //getter en setter voor Globaale variablen    void setVal(double val){      sliderVal = val;    }    double getVal(){      return sliderVal;    }  }  class MyApp extends StatelessWidget { // start app    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 { // functie om lijst aan data door te sturen over bluetooth      await widget.bluetoothManager.sendMessage(val);    }    Future<void> \_getBondedDevices() async { // functie om bleutooth gepairde devices te vinden      try {        devices = await widget.bluetoothManager.getBondedDevices();      } catch (ex) {        print('Error getting bonded devices: $ex');      }      setState(() {});    }    Future<void> \_connectToDevice() async { // functie om te verbinden met device      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 { // functie om verbinding te verbreeken      \_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 { // stuur bericht via bluetooth      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;   //bereken waarden voor wiskunde    nDistX = 2 - pDistX;    pDistY = 1 - y;    nDistY = 2 - pDistY;    pDistX.clamp(1, 2); // zet de warde tussen 1 en 2    nDistX.clamp(1, 2);    pDistY.clamp(1, 2);    nDistY.clamp(1, 2);    dm1 = sqrt(pow(nDistX, 2) + pow(pDistY, 2)) - sqrt(2); //pytagoras voor motorsnelheid    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; //stel lijst samen met signal en data bytes    vals[1] = (dm1 \* scalar).floor(); //doe snelheid maal constante en rond het af naar beneden    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}) { // WIP    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}) { // WIP    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> { // functie die een lijst met 1 singaal byte en 4 dezelvde data bytes maakt    Uint8List \_formatData(double inputVal, int signalVal) {      int newInpVal = inputVal.floor();      return Uint8List.fromList(          [signalVal, newInpVal, newInpVal, newInpVal, newInpVal]);    }    Future<void> \_sendMessage(Uint8List val) async { // stuut een lijst van bytes door      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 { //functie om verbonde devices te vinden      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 { // functie om te verbinden via bluetooth      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 { // testfunctie voor een bericht te sturen via bluetooth      try {        connection.output.add(message);        await connection.output.allSent;        print('Sent message: $message');      } catch (ex) {        print('Error sending message: $ex');      }    }    Future<void> disconnect() async {// functie voor los te koppelen van bluetooth      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.

Om de code te snappen heb je een kleine hoeveelheid voorkennis nodig maar dit is de essentie. Het flutter platform word gebruikt voor het ontwikkelen van apps. In flutter bijna alles wat je ziet dat geen Dart code is één van twee dingen. Of je bent naar een widget aan het kijken, of een attribuut. Als voorbeeld zie deze code

|  |
| --- |
| 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));    }  } |

Dit ziet er misschien ingewikkeld uit maar alles wat ons interesseert is dit deel.

|  |
| --- |
| Widget build(BuildContext context) {      return Container(          alignment: Alignment.center,          width: 40,          height: 40,          decoration:              const BoxDecoration(shape: BoxShape.circle, color: Colors.amber));    } |

In deze Build functie schrijven we wat we willen laten zien en hoe het zich moet gedragen. In dit geval starten we met een Container. Een container is een widget met een paar niet zo speciale attributen.

De attributen die we hier aanpassen zijn de “alingnment” de “width” de “height” en de “decoration”. We stelen deze gelijk aan een ander widget zoals bij de decoration of alingment of aan een nummer. Het const woord dat voor de BoxDecoration widget staat betekend gewoon dat het nooit zal veranderen. Ook kan je misschien verward zijn bij het alignement attribuut. Ook al lijkt Alingment.center niet op een widgit wat meestal een naam met haakjes erachter is [bv: BoxDecoration( )]. Maar het is wel een widget, gewoon 1 met vooringestelde waarde. Je ziet bij de BoxDecoration dat het een vorm (shape) en kluer heeft. Je zou ook een vooringestelde widget kunnen gebruiken bij het decoration attribuut wat misschien “AmberCricle” zou noemen.

Afbeelding met tekst, schermopname, nummer, software

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 hoofdscherm van de app, de linkse en rechtse blauwe cirkels met kleine gele cirkels in het midden, zijn twee joysticks die je kunt gebruiken om de drone te controleren. De linkse is om naar boven en onder te bewegen of naar links of rechts te draaien. De rechtse is om naar voor, achter, links of rechts te bewegen.

Afbeelding met tekst, schermopname, lijn

Automatisch gegenereerde beschrijving

Je belandt op deze pagina als je op het pijltje naar boven duwt. Dit is een pagina om 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 hoofdpagina. Hier kan je verbinden met de drone via bluetooth.

# Uitbreiding

## Algemeen

Nu wat gerealiseerde is uit de weg is gaan we over naar uitbreidingen en dingen die ik niet heb kunnen doen.

## Gyroscoop / versnellingsmeter

Ook al zit er een gyroscoop aanboort heb ik hem niet kunnen gebruiken in de eindversie van mijn GIP. Het was de bedoeling om deze te gebruiken om een zweef functie te implementeren. Dit zou ik gedaan hebben via de versnellingsmeter.

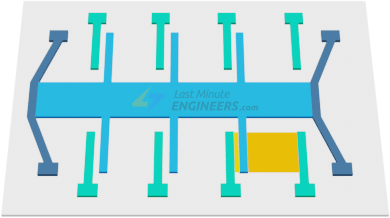


foto 11: versnellingsmeter (lastminuteengineers.com)

deze sensor meet de versnelling op elke as (x, y, z), als de y-as de gene is die versnelling naar boven en beneden meet dan, kan je de snelheid van je motors met een PID regelaar aanpassen tot dat je op de perfecte snelheid zweeft in de lucht. Alternatief zou je ook de motorsnelheid kunnen berekenen zodat de opwaartse kracht exact gelijk is aan de zwaartekracht. Waardoor je ook zou zweven.

Alternatief zou ik de gyroscoop gebruikt kunnen worden voor het stabiel houden van de drone bij hevige wind of ander omstandigheden. Aangezien deze de hoeks verandering meet zou je ook weer een PID regelaar kunnen gebruiken.

### PID algoritme

Een PID-regelaar is een techniek om een gewenste waarde te bereiken en te behouden in een regelsysteem. Het bestaat uit drie termen: proportioneel (P), integraal (I) en afgeleid (D). P reageert op de huidige fout, I corrigeert voor aanhoudende fouten en D anticipeert op toekomstige fouten. De regelactie wordt berekend door deze termen te combineren. De waarden van de termen worden afgestemd op het systeem voor optimale prestaties.

Een PID-regelaar is een techniek om een gewenste waarde te bereiken in een regelsysteem. Het gebruikt proportionele, integrale en afgeleide termen om de regelactie te bepalen op basis van de fout. Door deze termen te combineren, kan de regelaar de regelactie aanpassen en het systeem naar het gewenste punt sturen. De waarden van de termen worden aangepast voor optimale prestaties.

## Camera

als ik tijd had gegald zou ik een camera op mijn drone gezet hebben en zodat ik mee kon kijken uit het perspectief van de drone. Dit zou ook handig zijn voor foto’s of video te nemen van in de lucht. Een andere optie is om het te verbinden met een gezichtsherkenningen algoritme en de drone iemand te laten volgen zoals een cameraman. Dit zou ik doen met de ESP32, een Arduino met ingebouwd wifi en camera, dus eigenlijk perfect voor deze taak.



foto 12: ESP 32 Wi-Fi camera Arduino (store.fut-electronics.com)

## FPV

Als ik mij echt zou uitleven dan zou ik de camera iets verder nemen, ik zou deze normale drone ombouwen naar een FPV drone. FPV staat voor first person view, dus je zou kunnen kijven vanaf de drone zijn perspectief. Hiervoor zouden we geen normale camera kunnen gebruiken maar 2 voor diepte te zien. Zie deze 2 camera’s als 2 ogen van een mens. Ik zou mijn video samenzitten in 1 beeld en op een VR bril zetten. Of ik zou een camera gebruiken die voor VR is gemaakt zoals de doen in commercieel FPV drones.



foto 13: VR bril ([www.pcmag.com](http://www.pcmag.com))

dan zou je door de ogen van de drone hem kunnen besturen zoals de doen in pro drone racen. Ik zou waarschijnlijk Unity 3D gebruiken voor dit te realiseren aangezien ik vertrouwd ben met deze software en als dingen in VR heb gemaakt.



foto 14: FPV drone met VR bril (www.uploadvr.com)

# Reflectie

Op het einde van dit project heb ik veel bijgeleerd en ben blij dat ik mezelf heb uitgedaagd, er zijn een paar dingen die ik anders had gedaan. In het begin ben ik veel in de fout gegaan door te snel te willen zijn waardoor er later problemen waren. Het 1ste ding dat beter kon was het ontwerpen van de print ik had langer moeten denken over hoe het in de drone zou passen en wat er allemaal op moest komen en ook hoe de componenten georiënteerd moesten worden. Het 2de is meer aandacht vestigen aan de connecties, ik had wat problemen op een geven moment dat mijn connecties van de ESC’s die te groot waren voor mijn batterij. Als 3de had ik beter meer aandacht besteed aan de bestellingen ik, heb een paar keer de foute componenten besteld zoals te kleine propellers. Als 4de en laatste had ik meer moeten denken over mijn schakeling waaraan ik veel tijd heb verspild om met een gebroken arm transistors te soldereen die niet essentieel waren.

# Bibliografie

*Foto 1: HC-05 Bluetooth module*. (sd). Opgehaald van www.theengineeringprojects.com

*Foto 2: MPU-6050 gyroscoop*. (sd). Opgehaald van components101.com

*Foto 3: SDA en SCL voor data transfer*. (sd). Opgehaald van electronics.stackexchange.com

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

*Foto 6: PWM signaal*. (sd). Opgehaald van en.wikipedia.org

*Foto 7: motors*. (sd). Opgehaald van www.aliexpress.com

*Foto 8: propeller richting*. (sd). Opgehaald van easydrone.com

*Foto 9: Newton ‘s 3de wet*. (sd). Opgehaald van www.wired.com

*Foto 10: bernoulli effect*. (sd). Opgehaald van encyclopedia2.thefreedictionary.com

*foto 11: versnellingsmeter*. (sd). Opgehaald van lastminuteengineers.com

*foto 12: ESP 32 Wi-Fi camera Arduino*. (sd). Opgehaald van store.fut-electronics.com

*foto 13: VR bril*. (sd). Opgehaald van www.pcmag.com

*foto 14: FPV drone met VR*. (sd). Opgehaald van (www.uploadvr.com)