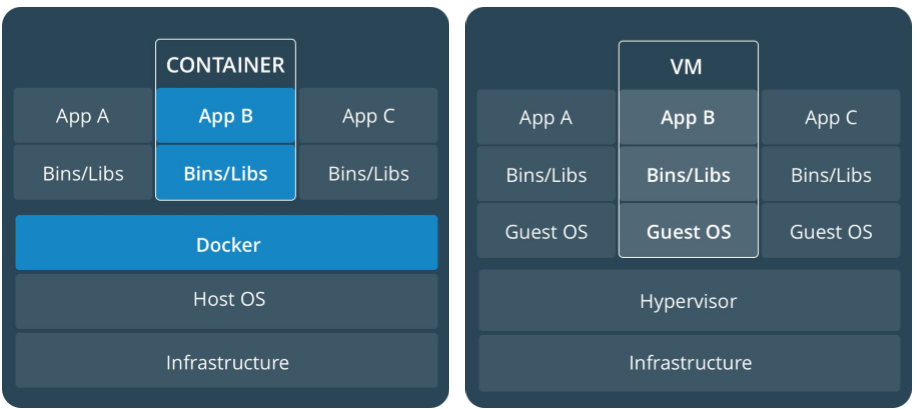
***Robotics advanced***

**Containerized ROS architectures:**

Goals:

* **Can describe the difference between containers and virtual machines.**



Het containersysteem vereist een onderliggend besturingssysteem dat basisservices biedt voor alle container applicaties met behulp van virtual memory support voor isolatie. Containers bundelen hun eigen software, libraries en config files. Meerde containers kunnen op hetzelfde machine draaien en delen de OS kernel. Een hypervisor daarentegen voert VM’s uit die een eigen besturingssysteem hebben met harware VM support. Container systemen hebben een lagere overhead dan VM’s.

* **Can explain the difference between containers and images.**

Het grootste verschil tussen een container en een image is de top writable layer. Alle writes naar de container die nieuwe gegevens toevoegen of bestaande gegevens wijzigen, worden opslagen in de writable layer. Als de container verwijderd wordt, wordt de writable layer ook verwijderd. Het onderliggende image blijft ongewijzigd

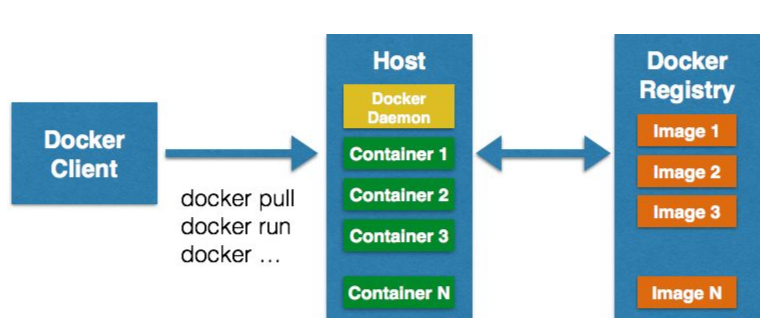
Images zijn frozen immutable snapshots van live containers. Containers draaien (of stoppen) exemplaren van een image.

* **Can explain what Docker is.**

Docker is de meest gebruikte manier om reproducibility, isolated containers te maken. Het is een handige en lightweight manier om een applicatie te encapsuleren voor het te (re)deployen met hetzelfde resultaat.

* **Can describe what registries are.**

Een Docker-register is een opslag- en distributiesysteem voor named Docker-images. Dezelfde image kan meerdere verschillende versies hebben, te herkennen aan hun tags. Een Docker-register is georganiseerd in Docker-repositories, waar een repository alle versies van een specifieke image bevat. Met het register kunnen Docker-gebruikers images lokaal ophalen en nieuwe images naar het register pushen.



* **Can explain docker volumes.**

Volumes zijn het voorkeursmechanisme voor het bewaren van gegevens die zijn gegenereerd door en worden gebruikt door Docker-containers. Volumes zijn externe opslaggebieden die worden gebruikt om gegevens op te slaan die zijn geproduceerd door een Docker-container. Volumes kunnen zich op de docker-host bevinden of zelfs op externe machines.

**Multi container ROS architectures:**

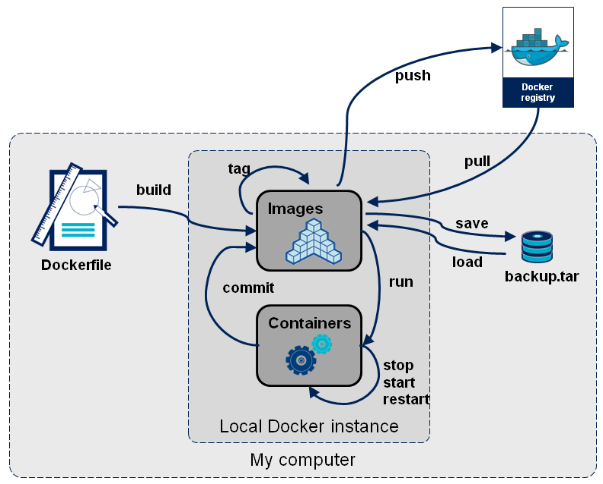
Goals:

* **Can explain what a container is.**

Een container is een standaardeenheid van software die code en al zijn dependencies verpakt, zodat de toepassing snel en betrouwbaar van de ene computeromgeving naar de andere wordt uitgevoerd.

Een afbeelding van een docker-container is een lichtgewicht, stand-alone, uitvoerbaar softwarepakket dat alles bevat wat nodig is om een applicatie uit te voeren: code, runtime, systeemtools, systeembibliotheken en instellingen.

* **Can explain the benefits of containers.**
  + Images are self-contained
  + Images can be run ad-hoc, no booting
  + Minimal overhead
  + Standardized environment
  + Host isolation
  + Cloud-ready
  + Repeatable & reproducible robotics
* **Can explain all docker stages. #TODO**



* **Understands why decouple applications are good.**
* **Can explain the link between decoupled applications and multiple containers.**

Elke container moet maar een zorg hebben. Applicaties decoupling in meerdere containers maakt het eenvoudiger om horizontaal te schalen en containers opnieuw te gebruiken. (vb een web applicatiestack kan bestaan uit drie afzonderlijke containers, elk met zijn eigen unieke image; om de web app, datase, in-memory cach te beheren op een decoupled manier)

**Rapid Drone Software Prototyping:**

Goals:

Introduction to drones:

* **Can explain what a drone is in context of robotics.**

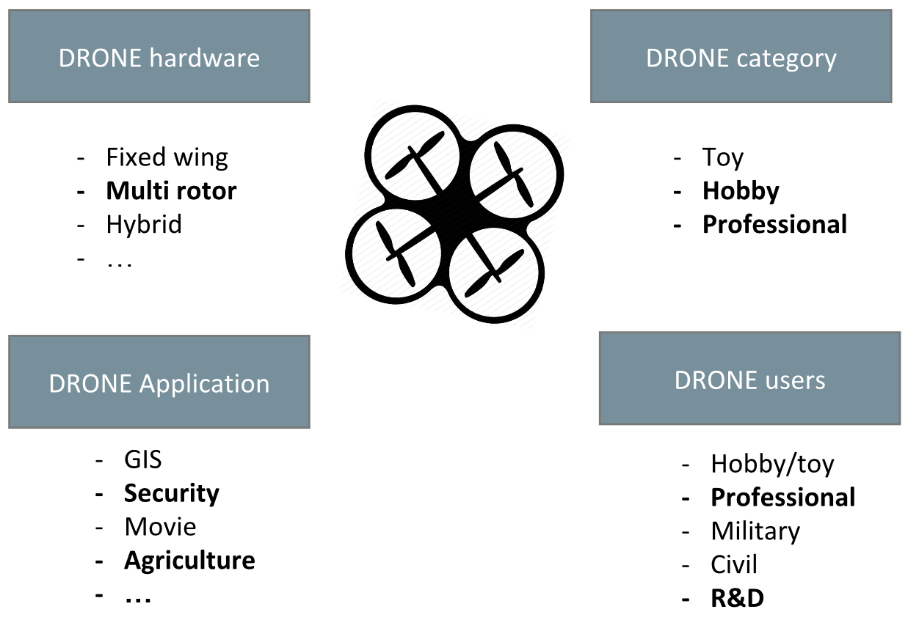
A remote-controlled pilotless aircraft.

* **Can explain the abbreviations UAV and RPAS.**

Uav = Unmannded Aerial Vehicle

RPAS = Remotely Piloted Aircraft Systems

* **Can layout a taxonomy of UAVs / drones.**



* **Can describe why drones became popular during the 2010s.**

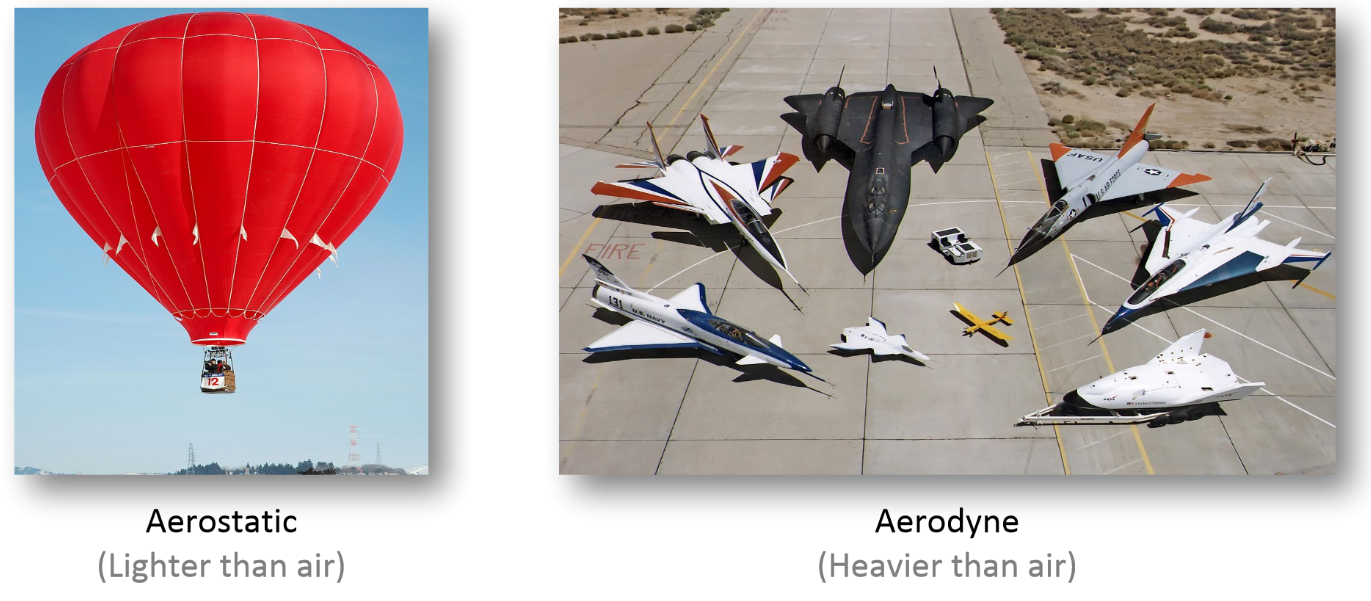
Dankzij de lage prijs & gewicht van electronische componenten: camera’s, data storage, sensors, GPS, radar,…

* **Can categorize UAVs in to the corresponding types: fixed wing, multirotor / multicopter, quadcopter, hexacopter, hybrid.**
  + Fixed wing: 
  + Multirotor/ multicopter: 
  + Hybrid: 
* **Can list the different drone usages and provide at least one example per usage.**
  + Military: RQ-4 Global Hawk, MQ-1 Predator, Black Hornet, Reaper
  + Foto & film
    - Lucht foto’s
    - Real estate
    - Onbereikbare plaatsen
    - Interessante camera perspectieven
  + Entertainment: lampjes op quadcopters
  + Infrastructuur onderzoeken: bruggen nakijken, deffecte zonnepanelen
  + Gevaarlijke plaatsen onderzoeken: kerncentrale, branden
  + Search & rescue: mensen in nood proberen te localiseren
  + Parcel delivery: Amazon, dhl packet levering
  + Inventory/ stock control: inventaris controleren
  + Argiculture
  + Fun, hobby en sport: drone racing
* **Can explain why drone safety and legislation is a necessity.**

Safety is belangerijk omdat een drone rotor veel schade kan aanrichten als het in contact komt met een mens, dier of object. Wetgeving (legislation) is ook belangerijk voor privacy, zodat niet iedereen overal een drone mag vliegen.

Drone physics:

* **Can describe the difference between aerostatic and aerodyne aeroplanes.**



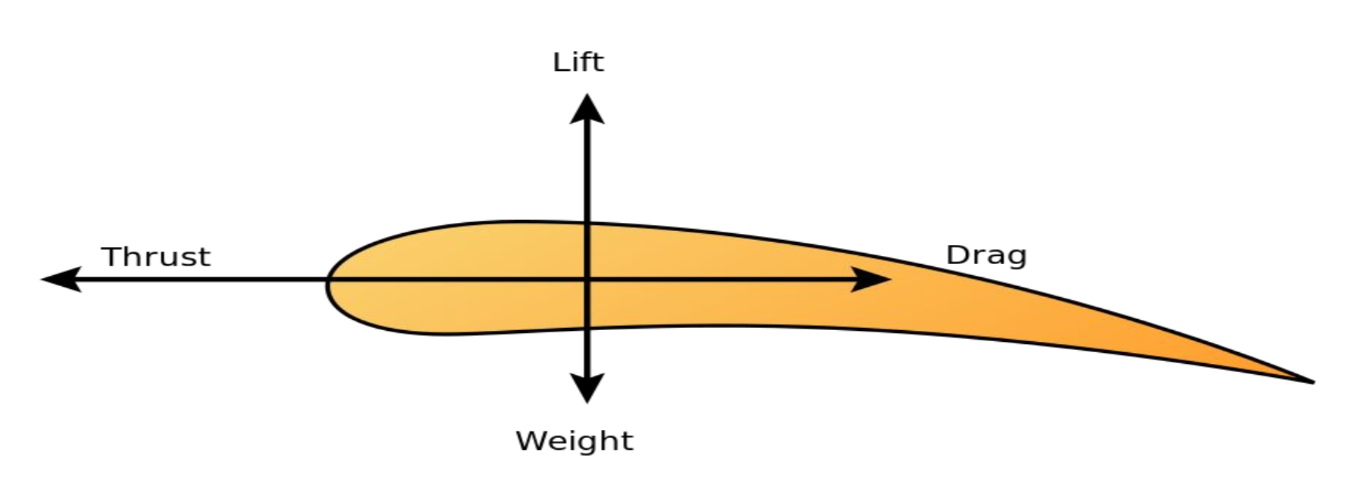
* **Can explain the four different forces that exist during flight.**

Lift: Opwaartse kracht gegenereerd door de luchtstroom rond het vleugelprofiel / vleugel.

Drag: Weerstandskracht die door het vliegtuig wordt ervaren door de lucht vanwege zijn voorwaartse beweging.

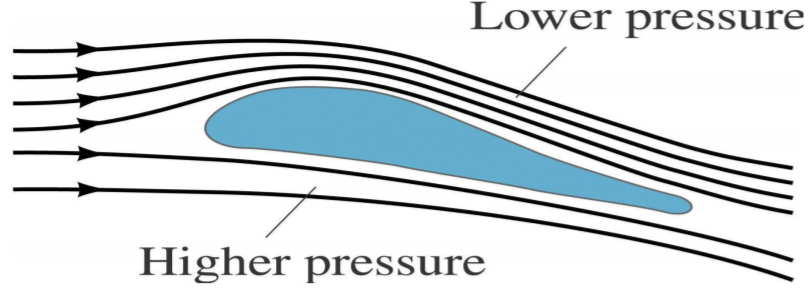
Thrust: De kracht dat het vliegtuig ervaart wat het vooruit helpt en helpt te vliegen.

De waarde van thrust en lift componenten moet hoger zijn dan het gewicht en sleep componenten voor een vliegtuig om te vliegen.

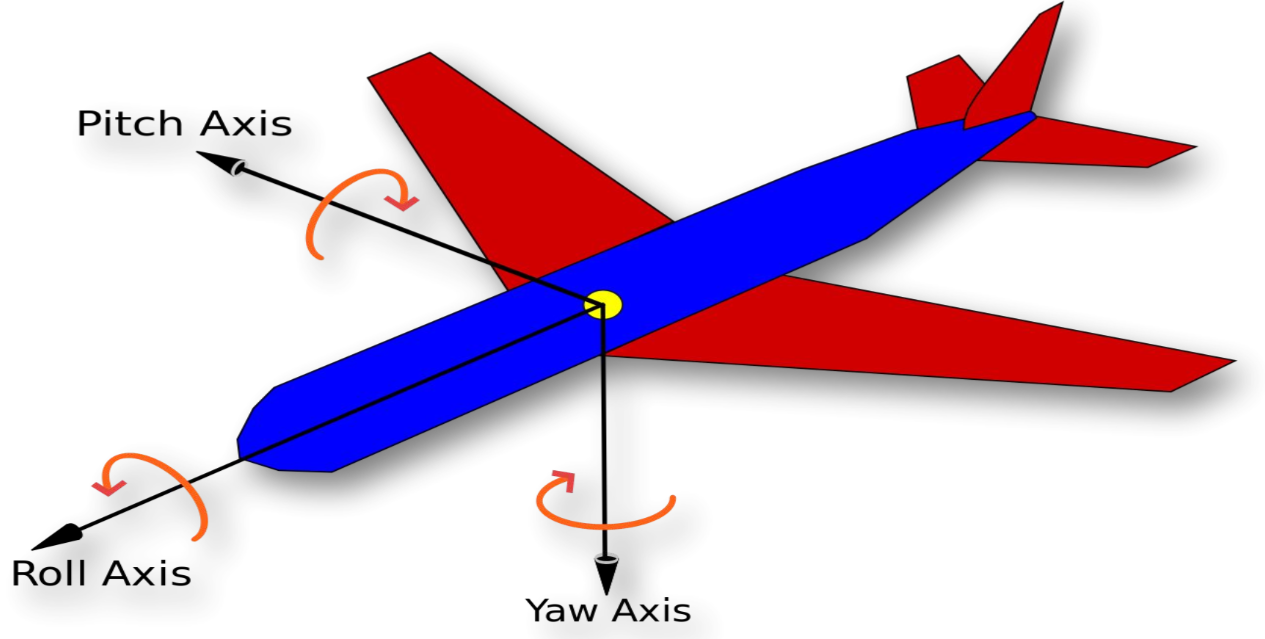


* **Can explain Bernoulli's principle.**

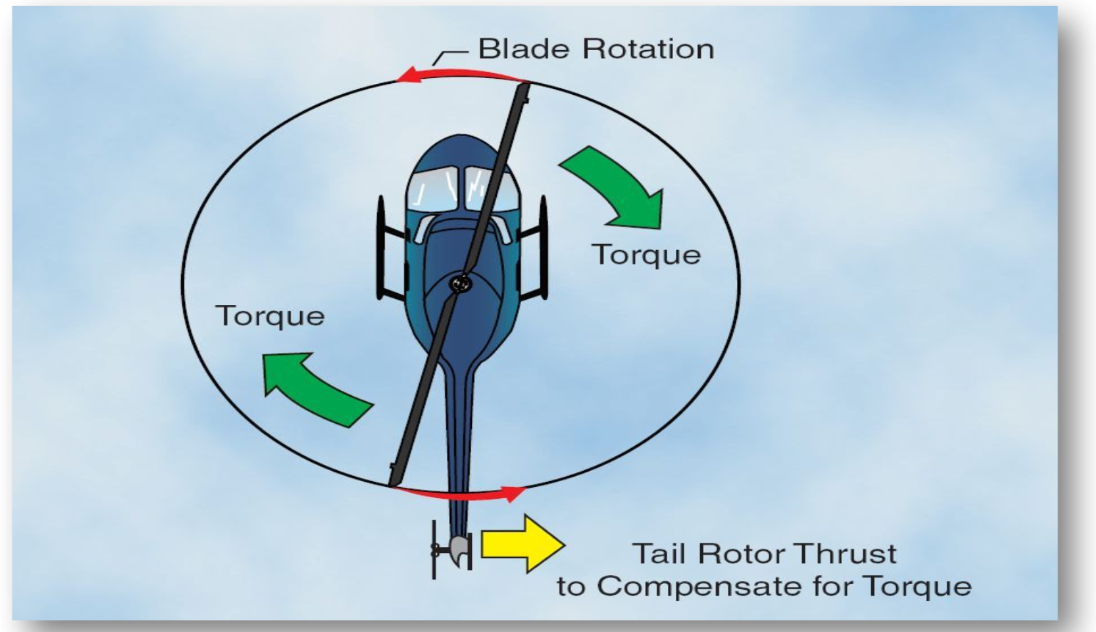
Het principe van Bernoulli helpt verklaren dat een vliegtuig lift kan bereiken vanwege de vorm van zijn vleugels. Ze zijn zo gevormd dat die lucht sneller over de bovenkant van de vleugel stroomt en langzamer eronder. Snel bewegende lucht is gelijk aan lage luchtdruk, terwijl langzaam bewegende lucht gelijk is aan hoge luchtdruk. De hoge luchtdruk onder de vleugels zal het vliegtuig daarom omhoog duwen door de lagere luchtdruk.



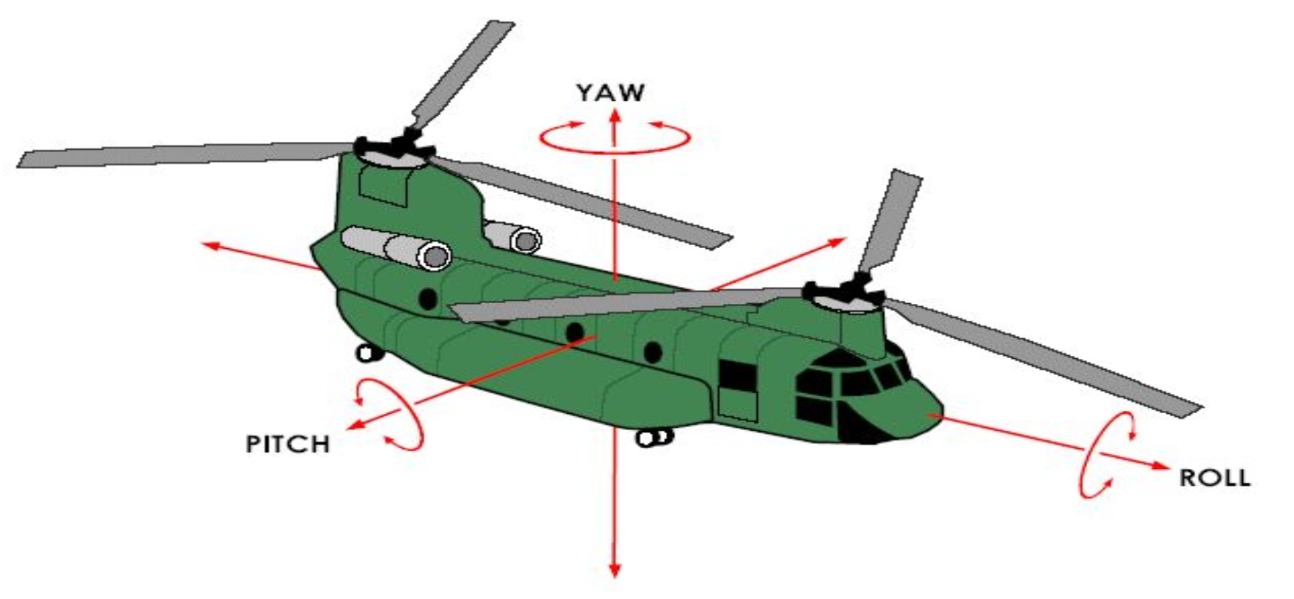
* **Can describe pitch, roll and yaw and draw a figure to depict the different axes.**



* **Understands blade rotation, torque and tail rotor thrust in context of helicopters.**



* **Understands pitch, roll and yaw in context of helicopters.**



* **Can explain lift, hoovering in context of helicopters.**

Lift: up/down, een helicopter's rotor blades zijn wings en creëren lift

Hoovering: Een helikopter kan, door zijn vleugel te draaien, de airflow over het lift oppervlak produceren zonder dat de hele helikopter vooruit hoeft te gaan.

* **Can explain collective pitch rotor helicopters and UAVs. (complex in een drone)**

Met CP verandert de pitch of invalshoek van de hoofdrotorbladen om de lift te regelen, terwijl de motor / motorsnelheid en rotorsnelheid min of meer constant blijven.

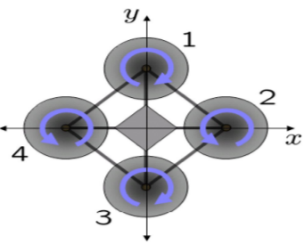
* **Can explain fixed pitch rotors in context of UAVs.**

De pitch van de hoofdrotorbladen wordt onder een vaste of constante invalshoek gehouden. U regelt de hoeveelheid lift naar uw helikopter door eenvoudig de snelheid van de motor / motor te variëren.

* **Can explain the different electronic components of an UAV: autopilot, motors, ground station, GPS antenna, radio, telemetry, remote control, battery, buzzer, ESC (electronic speed controllers). #TODO**

<https://dronebotworkshop.com/how-does-a-quadcopter-work/>

* **Can explain the modes of motion of a quadcopter and the direction and speed of its rotors.**

Direction of rotation 1 + 3 vs direction of rotation 2 + 4: cancels internal torque

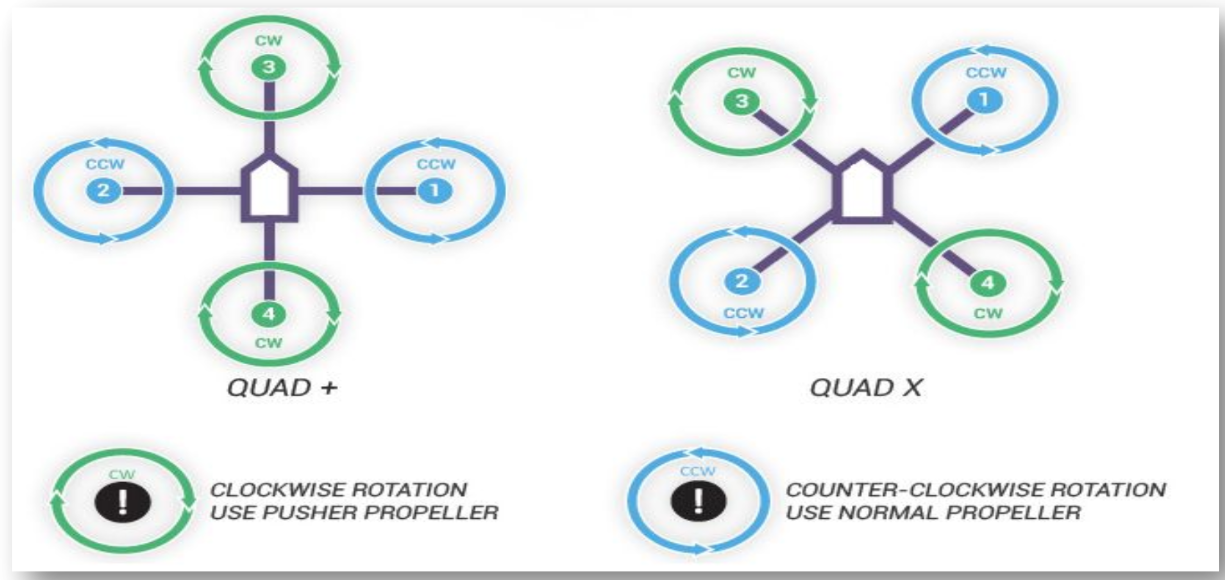
Modes of motion:

• Lift, hover: thrust aanpassen aan alle 4 motors

• Yaw: meer thrust aan 1 motorset (bv 1 en 3)

• Pitch: meer thrust aan 1 motor van de motorset (bv 1 meer dan 3)

• Roll: hetzelfde als pitch maar dan over de andere as



* **Can explain the magnitudes of lift.**

Stijgen, dalen en hover. Throttle 4 propellers aan de zelfde snelheid.

AR.Drone 2.0 Programming:

* Understands the different topics provided by the AR.Drone 2.0 ROS driver.

/ardrone/land  
/ardrone/navdata  
/ardrone/reset  
/ardrone/takeoff

/pxldrone/takeoff  
/pxldrone/land  
/pxldrone/pitch  
/pxldrone/yaw\_velocity  
/pxldrone/roll  
/pxldrone/z\_velocity

**Rapid Drone Software Prototyping PX4:**

Goals:

* **Can describe in own words the PX4 project and it’s software.**

PX4 is een open source software voor flight control voor drones en andere onbemande voertuigen. Het project biedt flexibele hulpmiddelen voor drone-developers om technologieën te delen om op maat gemaakte oplossingen voor drone-applicaties te creëren. PX4 biedt een standaard voor ondersteuning van drone-hardware en softwarestack, waardoor een ecosysteem hardware en software op een schaalbare manier kan bouwen en onderhouden.

* **Can describe in own words MAVLINK, QGroundControl, MAVSDK.**

MAVLINK (Micro Air Vehicle Link) is een protocol voor te communiceren met kleine unmanned vehicles. Ontworpen als header-only message library.

QgroundControl is een app om het configureren en vliegen van een PX4 based autopilot. Bied volledige flight control en mission planning voor alle MVLink enabled drone.

The MAVSDK is a MAVLink Library with APIs for C++, iOS, Python and Android. ... Developers can extend the core C++ SDK using plugins in order to add any other required MAVLink API (for example, to integrate a flight controller with custom cameras, gimbals, or other hardware over MAVLink).

* **Can describe in own words companion computer.**

Companion Computers gebruikt worden als interface voor te communiceren met een flight controller met behulp van het MAVLink protocol. Door dit doen ontvangt de companion computer alle MAVLink-gegevens die door de autopilot zijn geproduceerd en kunnen deze gebruiken om intelligente beslissingen te nemen tijdens de vlucht.

**Simultaneous localization and mapping:**

Goals:

* **Can describe in own words the SLAM problem including its components (mapping and localization).**

SLAM is het simultaan localiseren en mappen van de onbekende omgeving.

Mapping:

* + Wat is de wereld rondom me
    - Verstand van verschillende plekken en metingen toevoegen voor een map te maken
    - Veronderstelt perfecte kennis van positie

Localization:

* + Waar ben ik ergens in de wereld
    - Sensor waarden vertaald naar wereld model
    - Huidige locatie -> relatieve locatie van het model
* **Can explain the link between SLAM and the “chicken-egg” problem.**

Een map is nodig voor localisatie en een schatting van locatie is nodig voor mapping. 🡪 oplosingen: particle filter, extended kamlan filter, graphSLAM, …

* **Can describe and explain the accumulation of uncertainty in context of sensors.**

Het berekenen van de verwachte foutmarge (covariantie) tussen frames die de relatieve locatie voorstellen.

Als men een fout heeft bij de eerste frame, wordt de fout van de tweede frame hierbij opgeteld. Na een reeks frames, is de fout uit de hand aan het lopen. 🡪 oplossing: meerdere referentiepunten.

* **Can explain long-term position estimation without a priori information.**

Geen idee

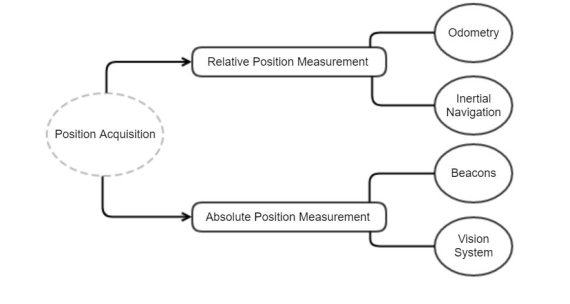
* **Can describe and explain the accumulation of uncertainty in context of SLAM.**

Indien de initële locatie van de robot verkeerd wordt geschat, wordt dit altijd bij opgeteld bij de volgende berekening. Hierdoor gaat de voorspelde locatie altijd meer en meer afwijken van de echte locatie.

* **Knows that there are multiple estimation methods, for example Kalman and Particle (How they work is out of scope!).**

Particle filter, extended kamlan filter, graphSLAM, …

* **Can divide localization methods starting from the position acquisition.**



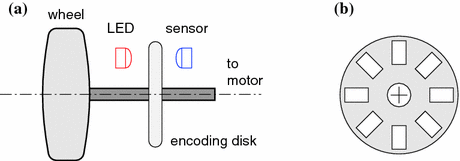
* **Can explain in own words relative and absolute positioning and the difference between them.**

Relative position measurement

* + Een van de simpelste methoden is gebruiken van wheel edemetry methoden die afhankelijk zijn van encoders om de hoeveelheid rotatie van de wielen te meten
    - De Rotatie metingen worden stapsgewijs gebruikt in combinatie met het bewegingsmodel om de huidige locatie van de robot te vinden
    - De belangrijste foutbron is wielslip op oneffen terreinen
  + Onafhankelijk van externe informatiebronnen
  + Nadeel: fouten worden groter en groter naar verlang van tijd

Absolute position measurement

* + Zijn wel afhankelijk van externe informatie
  + Deze systemen leveren de locatie van de robot dat is onafhankelijk van zijn vorige locaties
  + Locatie word gekend door slechts 1 meting
  + Voordeel: als er een foutmarge is wordt deze niet uitvergroot naargelang het aantal metingen stijgt
* **Can explain odometry and inertial navigation in context of absolute position measurement (-> moet relative zijn denk ik).**

Odometry staat in de goal hierboven uitgelegd 

Inertial navigation

* + Net als bij odometry worden positiebepalingen van inertial anvigation verkregen door de informatie van de sensoren te integreren; eenmaal voor de snelheid en dan voor de afgelegde afstand van de robot.
    - Dead recongnition
      * Berekenen van een positie aan de hand van vorige posities en bevorderen van die positie door rekening te houden met snelheid, afgelgegde tijd en course
  + Net zoals bij odometry worden deze bepaald door integratie wat maakt dat fouten uit vorige meting vergroot worden, deze metingen zijn onafhankelijk van externe informatiebronnen.
* Can explain beacons and visions systems in context of absolute position measurement.
* Can describe odometry and give an example.
* Can explain dead reckoning.
* Can explain inertial navigation including the term IMU.
* Can pinpoint the most significant error source of rotary encoders.
* Can explain drive and other issues with IMUs.
* Can explain the link between LIDAR, optical sensors, WiFi, GPS and absolute positioning measurement.
* Can explain why WiFi isn’t a good sensor in the context of SLAM.
* Can list at least two issues with GPS sensors.
* Can describe in own words the term Visual Odometry (VO).
* Can explain the difference between monocular and stereo cameras in the context of VO.
* Can explain the difference between feature based and direct method in the context of VO.
* Can describe visual inertial odometry in own words.
* Can describe (a 6 step) generic VO algorithm.
* Can explain egomotion.
* Can explain the benefit of multi sensor fusion and can draw an example to support the explanation.
* Can explain the basic idea of Markov localization and draw a simple figure to support the explanation.
* Can list at least two SLAM applications.
* Can describe the difference between interoceptive and exteroceptive sensors and there link with relative or absolute position measurements.
* Can explain how acoustic sensors measure distance; where they are applicable; and their shortcomings.
* Can explain how Laser rangefinders measure distance; where they are applicable; and their shortcomings.
* Can describe how monocular cameras measure distance.
* Can explain the unknown scale factor issue including dimensionless maps.
* Can describe a solution for the unknown scale factor issue.
* Can describe in own words how stereo cameras work.
* Can explain the term FPGA.
* Can describe in own words how RGB-D cameras work, including the difference between structured light and time-of-flight and their shortcomings.
* Can explain the difference between feature maps and occupancy grids.
* Can draw a simple architecture of a SLAM system divided in frond-end and back-end.
* Can explain the difference between sparse and dense Visual SLAM methods.
* Can explain the difference between feature-based and direct Visual SLAM methods.
* Can describe feature extraction and feature matching in context of Visual SLAM methods.
* Can explain loop closure in context of SLAM.
* Can explain why back-end optimization is needed and the difference between camera pose optimization and bundle adjustment.
* Can run a gmapping SLAM with the TurtleBot2.

**Multi Agent Systems, Swarms and ROS2:**

Goals:

* Can describe the term multi agent systems.
* Can explain Distributed artificial intelligence; Parallel AI Distributed problem solving and multi-agent-systems (MAS).
* Can describe an agent in the context of MAS.
* Can draw the typical building blocks of an autonomous agent and describe it.
* Can explain the link between agent technology and MAS.
* Can list at least two MAS benefits.
* Can list at least two MAS critical challenges.
* Can explain the difference between homogeneous structure and heterogeneous structure in contact of MAS.
* Can explain the difference between hierarchical organization, holonic agent organization, coalitions teams and in contact of MAS and can draw a figure per type.
* Can describe the difference between local communication and backboards in context of MAS using self drawn figures to support the explanation.
* Can describe blackboards in a multi computer setup using self drawn figure to support the explanation.
* Can describe hierarchical state machine.
* Can create a simple hierarchical state machine using the SMACH library.
* Can describe active learning, reactive learning and learning based on consequence in context of MAS.
* Can describe the term Swarms.
* Can draw a simple representation of the ROS1 and ROS2 architecture to depict the differences.
* Can explain the differences between the ROS1 and ROS2 architectures.
* Can draw a figure depicting the ROS2, DDS, UDP/IP and Ethernet stack and an OSI 7 layer model.
* Can explain the benefit of DDS using the OSI 7 layer model.
* Can create a simple ROS2 robot project using publishers and subscribers.
* Can explain the ros\_brige to connect ROS1 with ROS2.

**Reinforcement learning (Guest Lecture):**

Goals:

* Can explain why computer vision is hard.
* Can, by using an example, explain why having no prior knowledge could be a problem for Neural Networks and how to solve it.
* Can explain the need for edge cases in an image dataset.
* Can give an example which can occur when using a dataset build out of uniform samples.
* Can explain the benefit of using simulations in the context of RL.
* Can describe why simulations are hard in context of RL and Robotics.
* Can explain how Tesla is able to accumulate so much annotated (training) data.
* Can describe what reinforcement learning (RL) is, including: action, observation and reward.
* Can list and describe four RL examples.
* Can give an example of a reward shaping side effect (alignment problem or unintended behavior).