

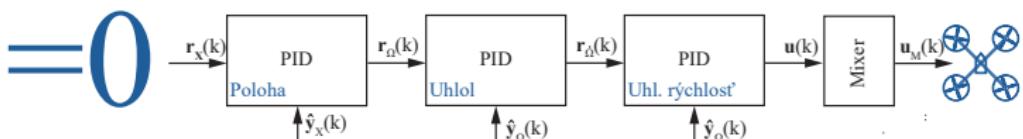
Working title drone control

Riadenie dronov

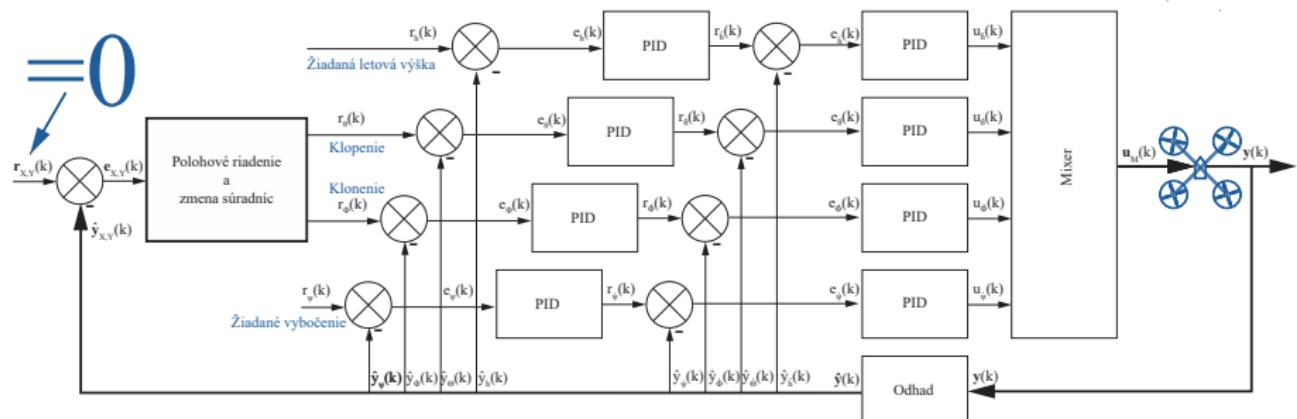
prof. Ing. Gergely Takács, PhD.



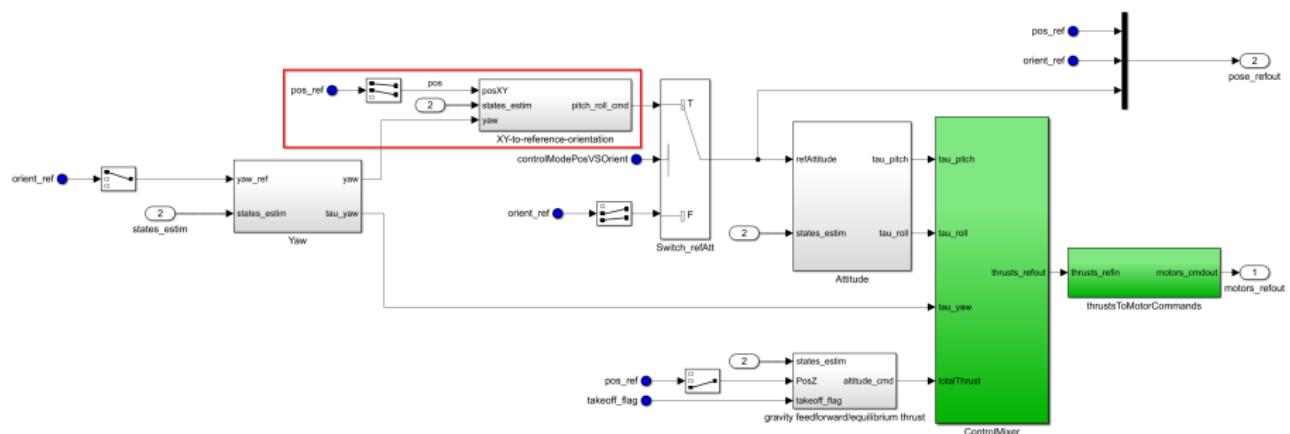
- Úlohou je držať polohu na základe globálnych súradníc (GS) napr. GPS. Pri minimalistickom prevedení (po zmene súradníc!) môžeme priamo premeniť polohu na uhol
- Ako by vyzeral náš kaskádovaný riadiaci systém?
- Môže to tak priamo fungovať?



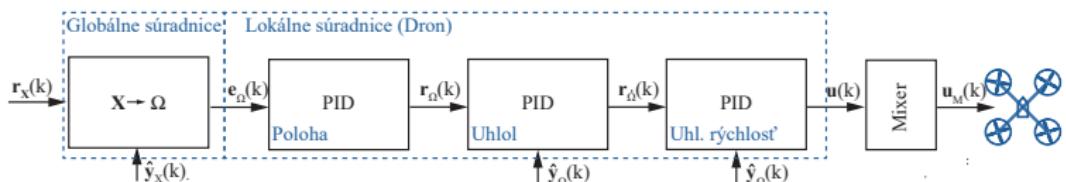
- Úlohou je držať polohu na základe globálnych súradníc (GS) napr. GPS. Pri minimalistickom prevedení (po zmene súradníc!) môžeme priamo premeniť polohu na uhol
- Ako by vyzeral náš kaskádovaný riadiaci systém?
- Môže to tak priamo fungovať?



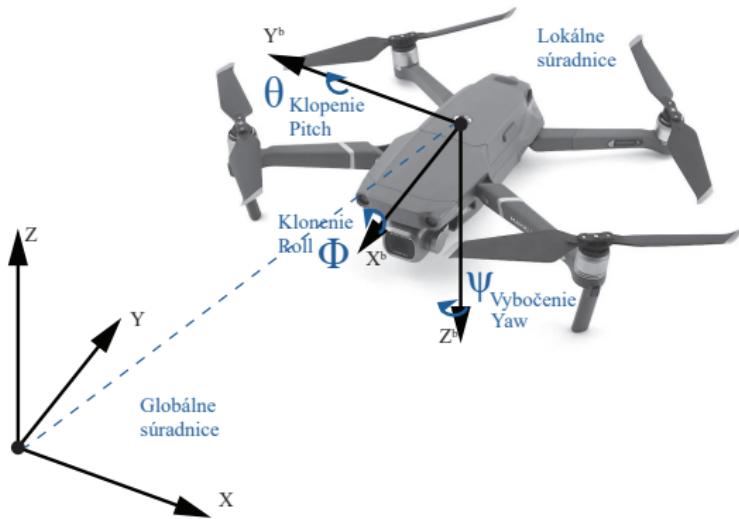
- Úlohou je držať polohu na základe globálnych súradníc (GS) napr. GPS. Pri minimalistickom prevedení (po zmene súradníc!) môžeme priamo premeniť polohu na uhol
- Ako by vyzeral náš kaskádovaný riadiaci systém?
- Môže to tak priamo fungovať?



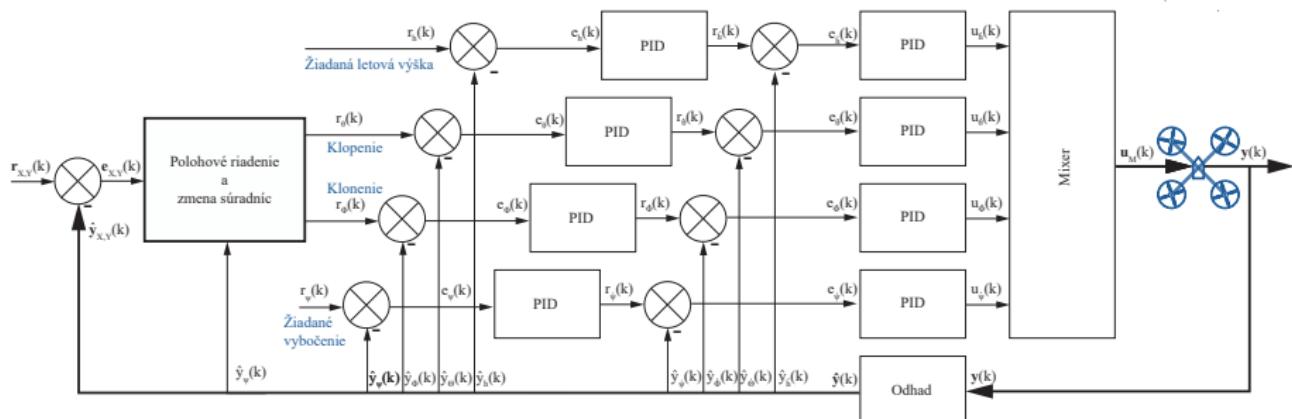
- Aj pri priamom riadení polohy musíme zmeniť súradnice
- Klopenie a klonenie určíme na základe GS súradníc, t.j. premeníme  $X, Y \rightarrow \Theta, \Phi\dots$
- ... ale potrebujeme na výpočet aj želané/aktuálne vybočenie, t.j.  $X, Y, \Psi \rightarrow \Theta, \Phi,$



- Aj pri priamom riadení polohy musíme zmeniť súradnice
- Klopenie a klonenie určíme na základe GS súradníc, t.j. premeníme  $X, Y \rightarrow \Theta, \Phi\dots$
- ... ale potrebujeme na výpočet aj želané/aktuálne vybočenie, t.j.  $X, Y, \Psi \rightarrow \Theta, \Phi,$



- Aj pri priamom riadení polohy musíme zmeniť súradnice
- Klopenie a klonenie určíme na základe GS súradníc, t.j. premeníme  $X, Y \rightarrow \Theta, \Phi\dots$
- ... ale potrebujeme na výpočet aj želané/aktuálne vybočenie, t.j.  $X, Y, \Psi \rightarrow \Theta, \Phi,$



- Môžeme napr. používať trigonometriu, teda rotačné matice (*angl.*: direction cosine matrix, DCM) na premenu z globálnych (G) do lokálnych (L) súradníc
- Najjednoduchší ale funkčný príklad, stabilizácia klonenia a klopenia. Majme odchýlku polohy  $e_{X,Y}^G = [X, Y]^T$  a aktuálne vybočenie  $\Psi$ , potom [Ben-Ari 2017]

$$e_{\Theta,\Phi}^L = \begin{bmatrix} \cos \Psi & -\sin \Psi \\ \sin \Psi & \cos \Psi \end{bmatrix} e_{X,Y}^G \quad (1)$$

- V príklade v MATLAB/Simulink je to prakticky identické

- Môžeme napr. používať trigonometriu, teda rotačné matice (*angl.*: direction cosine matrix, DCM) na premenu z globálnych (G) do lokálnych (L) súradníc
- Najjednoduchší ale funkčný príklad, stabilizácia klonenia a klopenia. Majme odchýlku polohy  $e_{X,Y}^G = [X, Y]^T$  a aktuálne vybočenie  $\Psi$ , potom [Ben-Ari 2017]

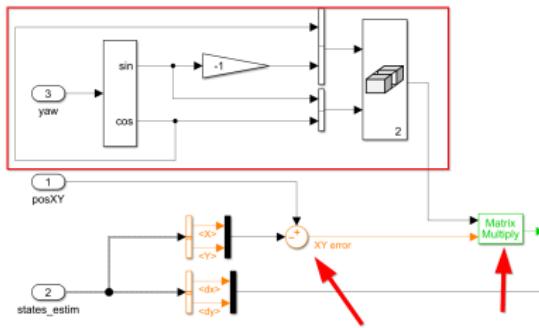
$$e_{\Theta, \Phi}^L = \begin{bmatrix} \cos \Psi & -\sin \Psi \\ \sin \Psi & \cos \Psi \end{bmatrix} e_{X,Y}^G \quad (1)$$

- V príklade v MATLAB/Simulink je to prakticky identické

- Môžeme napr. používať trigonometriu, teda rotačné matice (*angl.: direction cosine matrix, DCM*) na premenu z globálnych (G) do lokálnych (L) súradníc
- Najjednoduchší ale funkčný príklad, stabilizácia klonenia a klopenia. Majme odchýlku polohy  $e_{X,Y}^G = [X, Y]^T$  a aktuálne vybočenie  $\Psi$ , potom [\[Ben-Ari 2017\]](#)

$$e_{\Theta, \Phi}^L = \begin{bmatrix} \cos \Psi & -\sin \Psi \\ \sin \Psi & \cos \Psi \end{bmatrix} e_{X,Y}^G \quad (1)$$

- V príklade v MATLAB/Simulink je to prakticky identické

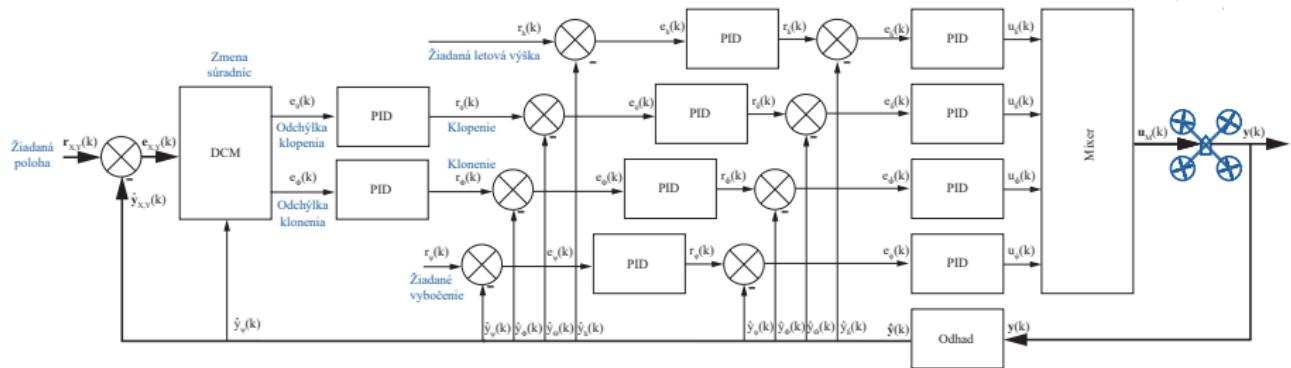


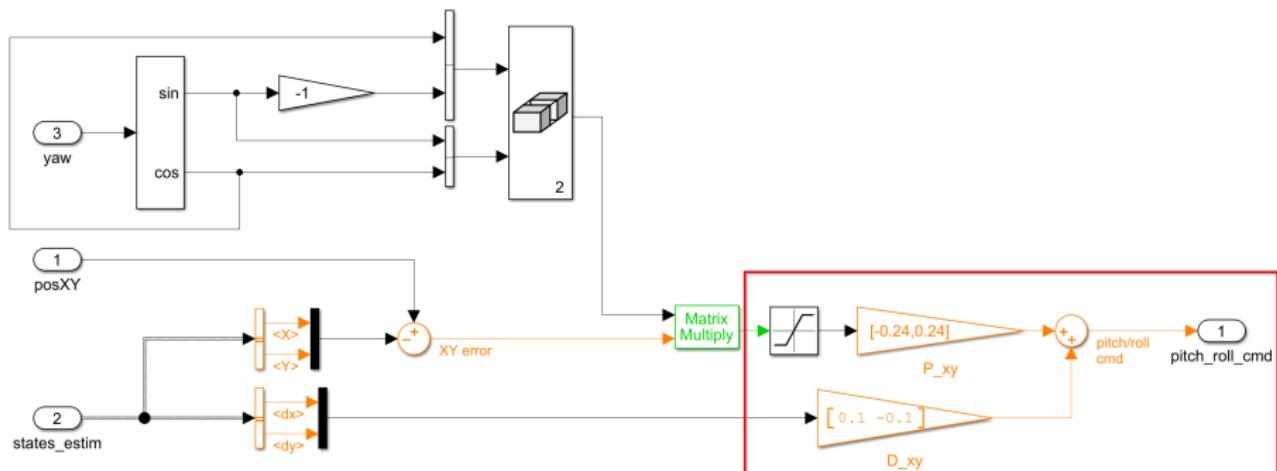
- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [Ben-Ari 2017]
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [Ben-Ari 2017]
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny

- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [Ben-Ari 2017]
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [Ben-Ari 2017]
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny

- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [Ben-Ari 2017]
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [Ben-Ari 2017]
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny

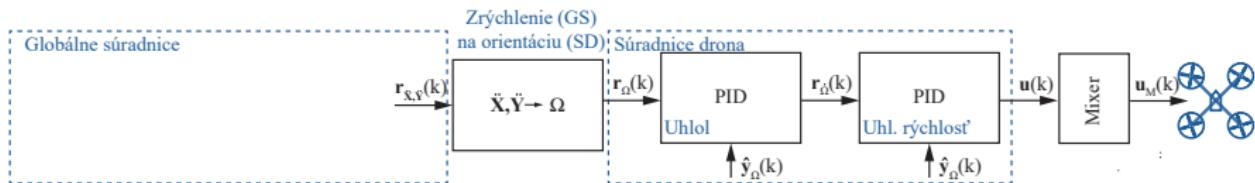
- ArduCopter využíval DCM až do v. 3.2.
- Pri 3D reprezentácii DCM sa začínajú komplikovať, a obsahujú singularity [\[Ben-Ari 2017\]](#)
- Riešenie: reprezentácia Eulerových uhlov cez kvaternióny (*angl.:* quaternion) ktoré odstránia problém so singularitou a sú výpočtovo efektívnejšie [\[Ben-Ari 2017\]](#)
- V súčasných verziách aj ArduCopter aj PX4 využíva kvaternióny



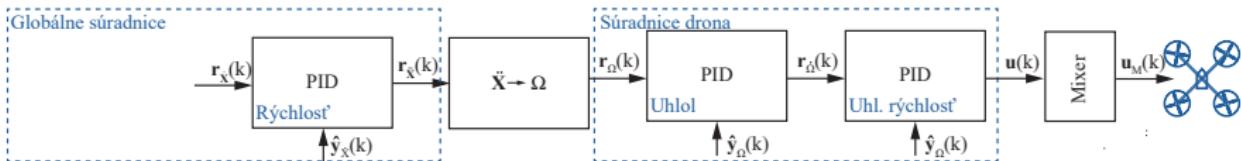


- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto  $X, Y \rightarrow \Theta, \Phi$  máme  $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).

- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto  $X, Y \rightarrow \Theta, \Phi$  máme  $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).

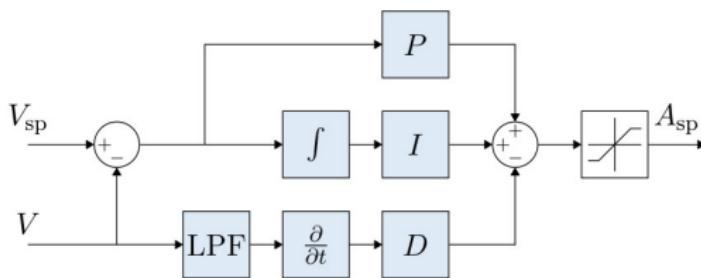


- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto  $X, Y \rightarrow \Theta, \Phi$  máme  $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).

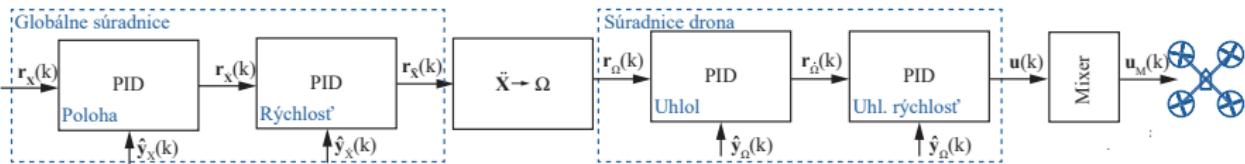


# Nepriame riadenie polohy: Riadenie rýchlosťi

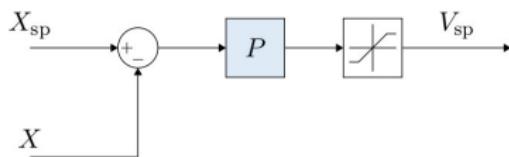
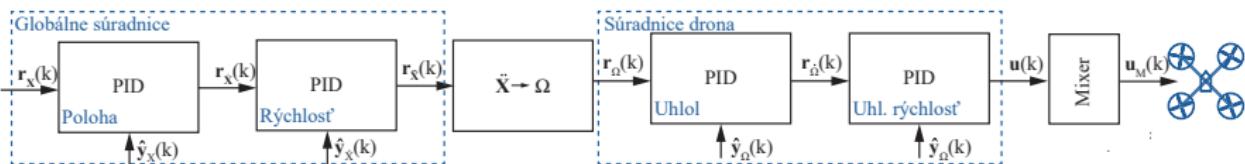
- V skutočnosti aby sme vedeli korigovať aj polohu aj rýchlosť na základe odhadov, aj tu máme ďalšie kaskádové prevedenie. Pridáme tzv. rýchlostnú slučku
- Transformácia súradníc je logicky na inom mieste
- Namiesto  $X, Y \rightarrow \Theta, \Phi$  máme  $X, Y \rightarrow \dot{X}, \dot{Y} \rightarrow \ddot{X}, \ddot{Y} \rightarrow \Theta, \Phi$
- Väčšinou je to plné PID riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. [Saha 2020](#)).



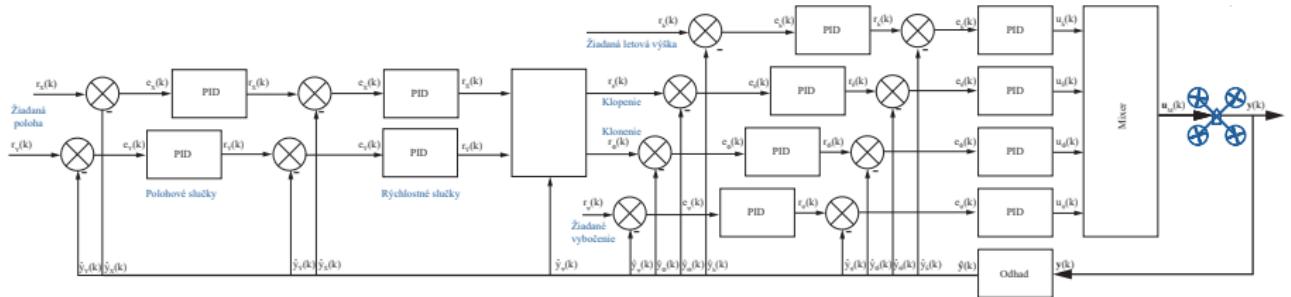
- Potrebujeme ešte jednu slučku — polohovú, ktorá premení polohu na rýchlosť
- Väčšinou je to P riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).

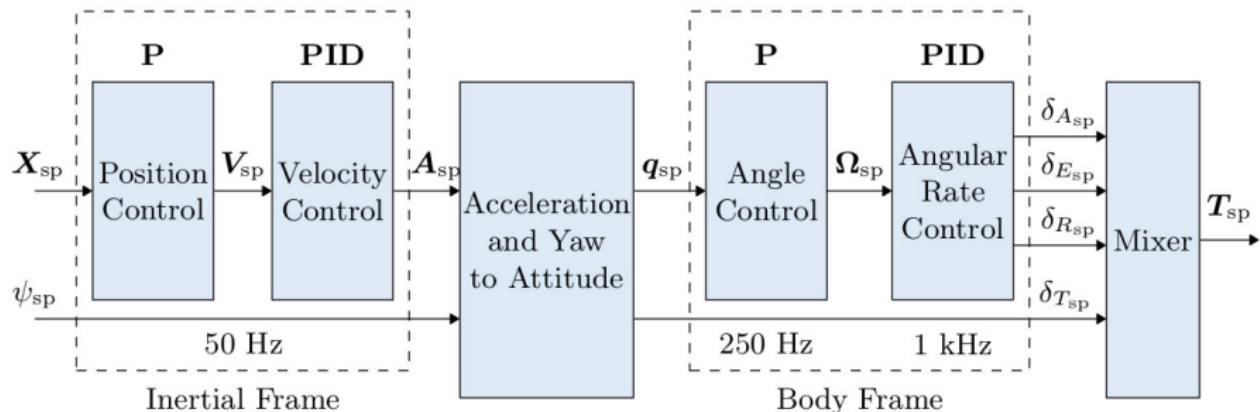


- Potrebujeme ešte jednu slučku — polohovú, ktorá premení polohu na rýchlosť
- Väčšinou je to P riadenie, ako aj pri ArduCopter, PX4 a inde (c.f. Saha 2020).



# Celková riadiaca architektúra

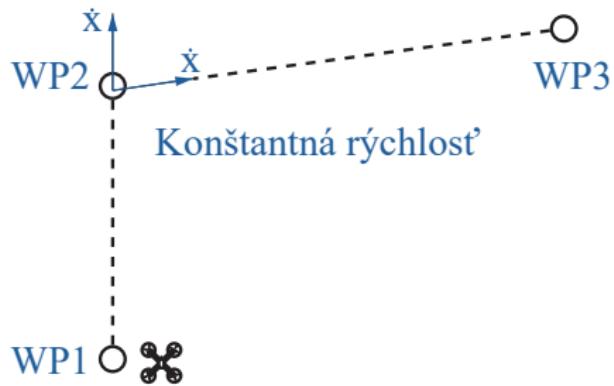




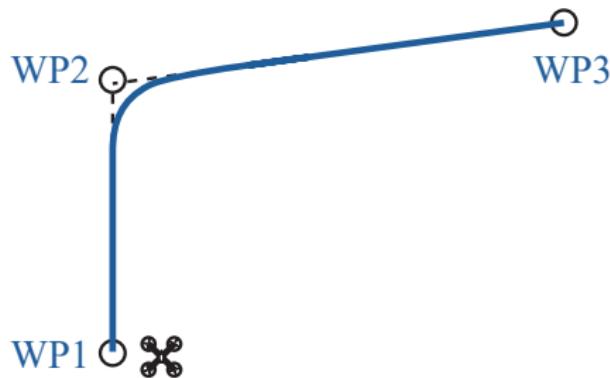
- Majme trasu WP1 do WP2, všetko je v poriadku.
- Pridajme WP3 a rozmýšľajme čo sa deje pri WP2. Je možné preletieť nad WP2?
- Bud' musíme úplne sa zastaviť alebo nemôžeme priamo preletieť — ináč by sme potrebovali nekonečne veľké zrýchlenia



- Majme trasu WP1 do WP2, všetko je v poriadku.
- Pridajme WP3 a rozmýšľajme čo sa deje pri WP2. Je možné preletieť nad WP2?
- Bud' musíme úplne sa zastaviť alebo nemôžeme priamo preletieť — ináč by sme potrebovali nekonečne veľké zrýchlenia



- Majme trasu WP1 do WP2, všetko je v poriadku.
- Pridajme WP3 a rozmýšľajme čo sa deje pri WP2. Je možné preletieť nad WP2?
- Bud' musíme úplne sa zastaviť alebo nemôžeme priamo preletieť — ináč by sme potrebovali nekonečne veľké zrýchlenia



Notes: [Hall 2018](#)

User (ROS) → Shaping → PID → Actuators Yaw je prioritizovanych nad Pitch Roll, lebo to drzi dron v lufte [\[Erasmus 2020\]](#) 50 Hz -> 400 Hz Min 24 tazsie uchopytelne koncepty pre prezentaciu, dava menej konkretnosti Velocity prioritizuje vertikalnu rychlosť [\[Erasmus 2020\]](#)

Ďakujem za Vašu pozornosť.

- [1] Moti Ben-Ari. *A Tutorial on Euler Angles and Quaternions*. Online. Version 2.0.1 [cited 5.12.2021]; Available from  
<https://www.weizmann.ac.il/sci-tea/benari/sites/sci-tea.benari/files/uploads/softwareAndLearningMaterials/quaternion-tutorial-2-0-1.pdf>. 2017.
- [2] Anton Erasmus. *An In-depth Look at the Multicopter Control System Architecture*. Online. PX4 Developer Summit Virtual 2020. [cited 1.12.2021]; Available from  
<https://www.youtube.com/watch?v=nEo4WG14Lgc>. 2020.
- [3] Leonard Hall. *Practical PID implementation and the new Position Controller*. [online]. ArduPilot UnConference 2018, uploaded Feb 22, 2018 [cited 24.11.2021]; Available from  
<https://www.youtube.com/watch?v=-PC69jcMizA>. 2018.

- [4] Dola Saha. *Precise Unmanned Aerial Vehicle (UAV) Flight Control*. Online. 2020 Fall ECE553 Resources [cited 3.12.2021]; Available from [https://www.albany.edu/faculty/dsaha/teach/2020Fall\\_ECE553/](https://www.albany.edu/faculty/dsaha/teach/2020Fall_ECE553/). 2020.