KeyboardControl.py doc.
How it works

1. Connecting

You can connect to drone or simulator by udp protocol.
For simulator IP address is **127.0.0.1** ,port is 14551 (in this script).For port working in simulator you should open it in Mavproxy by command:

mavproxy.py --master tcp:127.0.0.1:5760 --out udpout:127.0.0.1:14550 --out udpout:127.0.0.1:14551 --out udpout:127.0.0.1:14553 --out udpout:127.0.0.1:14552

For Solo drone, IP address is **0.0.0.0** port is 14550 (+1..+2..+3). Drone opens ports automatically you just should be connected to wifi of Solo.

2. Movements

We can control drone by Dronekit library and Mavlink protocol. Its pretty enough for controlling, but Solo has some limitations.

- we only can control speed and yaw.
- drone orient itself relative to NED coordinate system

For drone controlling we have 2 functions :

- send_ned_velocity(velocity_x, velocity_y, velocity_z, duration)
  velocity_x — North (go north velocity_x>0)
  velocity_y — East (go east velocity_y>0)
  velocity_z — Down
  duration — duration of speed keeping
- condition_yaw(heading, relative=False)
  (Comments in code enough for understand how its work)

We should recalculate speed vectors and project it on «normal» coordinate system.We show it on simple movements (go forward/backward/right/down):

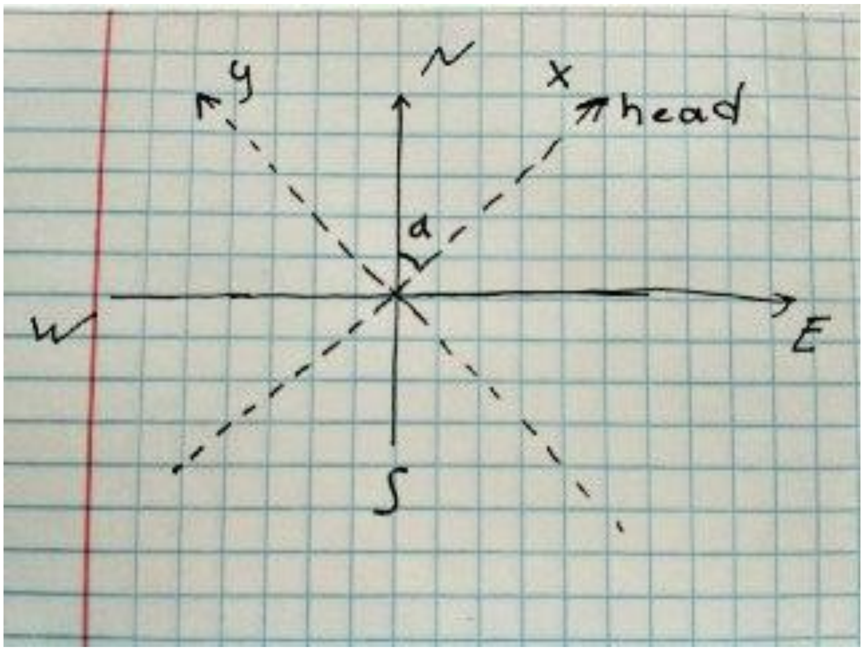
For recalculating we use *heading* function .It returns athimut (North 0 deg, East 90 deg).

calculateX(speed, heading):

Recalculate and project vector of speed frome NED coordinate system *to related to vehicle(normal) system. From N-S to X*

calculateY(speed, heading):

Recalculate and project vector of speed frome NED coordinate system to related to vehicle(normal) system. From E-W to Y



Go forward:

We get athimut(a) and know speed(S) which we want .
Then we just project vector of speed on NED coordinates:

a = heading = athimuth

N-S → S*cos a
W-E → S* sin a

and after that we get valid vectors

N-E (0→ 90) cos>0 sin >0 (N+E+)
S-E (90→ 180) cos<0 sin>0(N-E+)
S-W(180-270) cos<0sin<0(N- E-)
N-W(270-360) cos>0sin<0(N+E-)

Go backward:

Just inverse vectors from go forward

N-S → -1*S*cos a
W-E → -1*S* sin a

Go left:

New angle calculated , and projects calculates by this angle

New angle = heading — 90 (it left direction relative to drone)

Then works like go forward

Go right:

New angle calculated , and projects calculates by this angle

New angle = heading + 90 (it rightdirection relative to drone)

Then works like go forward

3. Following

For following we use 2 laptop system. Laptops must be connected via ssh.

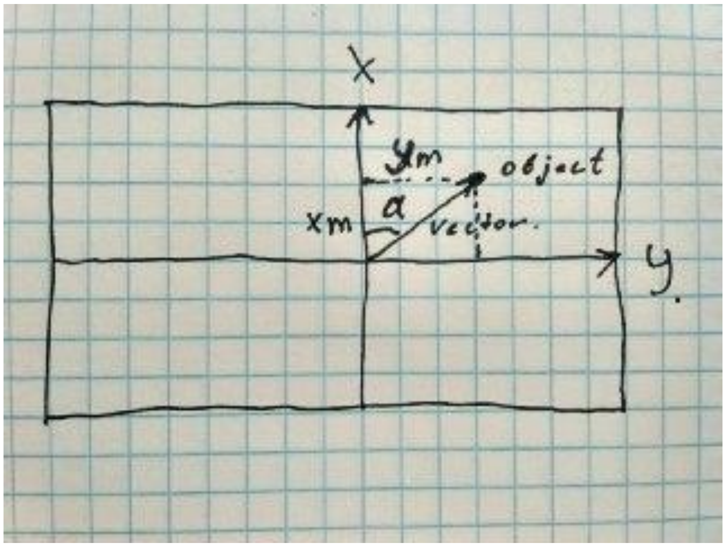
Laptop1 - «vision part» IP: **192.168.1.1**

Laptop1 connects to camera throught wifi and process videostream.
Then send data (coordiantes of bounding box that track object) throught socket (port 5000) to Laptop2.

Laptop2 - «drone control part » IP: **192.168.1.2**

Laptop2 connects to drone via wifi and has complete control over drone.

How following work:
image representation from camera



Firstly,we choose max speed(M) of our object (For example, human max speed 10 km/h).

Then base on this max speed we calculate speed for drone. From Laptop1 we get margins of object in image (xM,yM) . Then calculate proportion (max speed will equal to speed of drone , when object as much as possible far from drone: top (right/left) /bottom(right/left)):

proportion= $sqrt((xM^2+yM^2)/((vr/2)^2+(hr/2)^2))$ vr — vertical resolution of image , hr — horizontal resolution of image

So , speed will be: Speed = MAX_SPEED*proportion

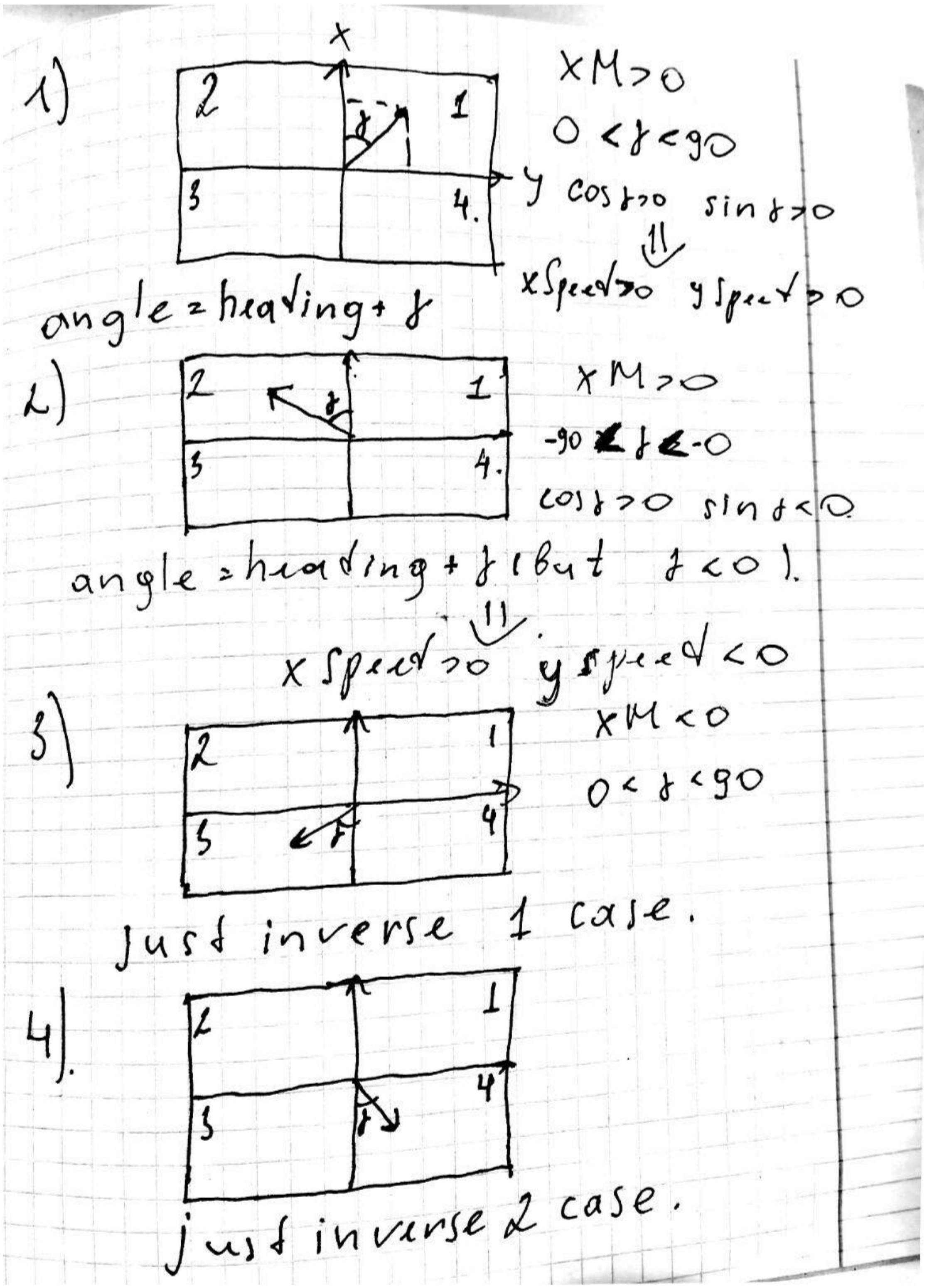
Calculate angle:

a = arctg(yM/xM)
We use numpy.arctan() function , that means that:

- Upper right/bittom left : 0→90
- Upper left/bottom right: -0→ -90

However, we should avoid cases when xM == 0 :

First, xM !=0:



1)

$xM > 0$

$0 < \gamma < 90$

$\cos\gamma > 0 \quad \sin\gamma > 0$

$\Downarrow$

$xSpeed > 0 \quad ySpeed > 0$

angle = heading + γ

2)

$xM > 0$

$-90 < \gamma < 0$

$\cos\gamma > 0 \quad \sin\gamma < 0$

angle = heading + γ (but $\gamma < 0$).

$\Downarrow$

$xSpeed > 0 \quad ySpeed < 0$

3)

$xM < 0$

$0 < \gamma < 90$

just inverse 1 case.

4).

just inverse 2 case.

Second, xM==0:

Speed by x axes will be equal to zero, by y we just send vector.

```
send_ned_velocity(0, vector,0, 0.25)
```