lstlisting[language = Java , frame = trBL, escapeinside=(*@@*)] def $_{c}ontrol_{l}oop(self)$: $whilenotself.kill_{t}hreads$:
$current_{p}ose = self.current_{p}osecurrent_{o}rientation = self.current_{o}rientation_{,}current_{y}aw = transformations.euler_{f}r$
$R.from_{q}uat(current_{o}rientation)current_{r}otation_{m}atrix = np.array(r.as_{m}atrix()).reshape((3,3))ifself.current_{s}tate! =$
$self.drone_{s}tates["GROUND"]$:

SET POINT SMOOTHING AND ERROR CALCULATION $set_{p}oint_{s}mooth = self.low_{p}ass_{f}ilter.filter(self.ma_{f}ilt$
$set_{p}oint_{s}mooth[: 3]des_{y}aw = set_{p}oint_{s}mooth[3]current_{e}rror_{y}aw = des_{y}aw - current_{y}awcurrent_{e}rror =$
$des_{p}ose - current_{p}ose$

REGISTRAZIONE OUTPUT (X,Y,Z) xp = np.append(self.pose$_x$, $[current_{p}ose[0]])self.pose_{x} = xp$

yp = np.append(self.pose$_y$, $[current_{p}ose[1]])self.pose_{y} = yp$

zp = np.append(self.pose$_z$, $[current_{p}ose[2]])self.pose_{z} = zp$

tapp = np.append(self.time$_d$$ata$, $[time.time()])self.time_{d}ata = tapp$

error$_x$, $error_{y}, error_{z} = current_{e}rror$

if not self.current$_s$$tate == self.drone_{s}tates["FAIL"]$:

AFTER TAKE OFF if not self.current$_s$$tate == self.drone_{s}tates["TAKING_{O}FF"]andself.control_{p}itch_{r}oll$:

CONTROLLO STABILIZZATO if self.angle$_m$$ode$: $rotation_{m}atrix = np.array([[np.cos(current_{y}aw), -np.sin(curre$

error$_m$$atrix = np.array([[error_{x}, error_{y}]]).transpose()$

rotated$_e$$rror_{x}, rotated_{e}rror_{y} = np.dot(rotation_{m}atrix, error_{m}atrix).flatten()$

pitch = self.controller.pitch(rotated$_e$$rror_{x})roll = self.controller.roll(rotated_{e}rror_{y})yaw = des_{y}awtrust =$
$self.controller.trust(error_{z}, mes = current_{p}ose[2])$

REGISTRAZIONE INPUT (PHI, THETA, PSI, T) r1 = np.append(self.controller$_r$$oll, [roll])self.controller_{r}oll =$
$r1$

p1 = np.append(self.controller$_p$$itch, [pitch])self.controller_{p}itch = p1$

y1 = np.append(self.controller$_y$$aw, [yaw])self.controller_{y}aw = y1$

t1 = np.append(self.controller$_t$$hrust, [trust])self.controller_{t}hrust = t1$

self.publish$_a$$ttitude_{a}ngle(pitch = pitch, roll = roll, yaw = yaw, trust = trust)$

CONTROLLO ACROBATICO else: lqr$_o$$bservation = "p" : current_{e}rror.reshape((1,3)), "R" : current_{r}otation_{m}at$r
$self.controller_{l}qr.perform_{a}ction(lqr_{o}bservation)self.publish_{a}ttitude_{a}cro(wtr[0], wtr[1], wtr[2], trust)$

TAKE OFF else: pitch = self.controller.default$_p$$itchroll = self.controller.default_{r}ollyaw = self.init_{y}aw[0]ifself.an$
$ifself.trust_{a}s_{t}rust : trust = self.controller.trust(error_{z}, mes = current_{p}ose[2])else : trust = self.controller.TRUST_{S}T$
$1.2else : lqr_{o}bservation = "p" : current_{e}rror.reshape((1,3)), "R" : current_{r}otation_{m}atrix, "yaw" : np.array([0.0])_{,}trust$
$self.controller_{l}qr.perform_{a}ction(lqr_{o}bservation)$

r2 = np.append(self.controller$_r$$oll, [roll])self.controller_{r}oll = r2$

p2 = np.append(self.controller$_p$$itch, [pitch])self.controller_{p}itch = p2$

y2 = np.append(self.controller$_y$$aw, [yaw])self.controller_{y}aw = y2$

t2 = np.append(self.controller$_t$$hrust, [trust])self.controller_{t}hrust = t2$

self.publish$_a$$ttitude_{a}ngle(pitch = pitch, roll = roll, yaw = yaw, trust = trust)$

else: self.publish$_g$$uided_{s}etpoint(des_{p}ose[0], des_{p}ose[1], des_{p}ose[2], des_{y}aw)$

DATA DICTIONARY input$_o$$utput_{d}ic = "controller_{r}oll" : self.controller_{r}oll, "controller_{p}itch" : self.controller_{p}itc$h

sio.savemat("input$_o$$utput_{d}ata.mat", input_{o}utput_{d}ic)$

self.rate$_p$$id.sleep()$