

```
1  //-----
2  //      Title:MASTER DEGREE THESIS by ANTONIO SCAZZI
3  //
4  //  Description:header file with all autopilots functions
5  //-----
6  #pragma once
7  #include "globalvar.h"
8
9  double Headingtakeoff(double comand_heading, double actual_heading);
10
11 double Pitchhold(double comand_pitch, double actual_pitch, double actual_pitch_vel);
12
13 double Bankhold(double max_bank, double actual_bank, double roll_rate);
14
15 double Headinghold(double comand_heading, double actual_heading, double max_bank, double actual_bank, double
    roll_rate);
16
17 double Altitudehold(double comand_altitude, double actual_altitude, double max_pitch, double actual_pitch, double
    actual_pitch_vel);
18
19 double Autothrottle(double comand_velocity, double C2_velocity, double acceleration);
20
21 double VerticalSeparation(std::vector<double> NED, double verticalseparation, double max_pitch, double
    actual_pitch, double pitch_rate);
22
23 double LateralSeparationApproach(std::vector<double> NED, double max_bank, double actual_bank, double
    actual_heading, double roll_rate);
24
25 double LateralSeparation(std::vector<double> NED, double comand_separation, double max_bank, double actual_bank,
    double roll_rate);
26
27 double ForwardSeparation(std::vector<double> NED, double comand_separation, double actual_velocity, double
    acceleration);
```

```
28
29
30
31 //autopilota dell'angolo di heading con controllo del solo timone
32 double Headingtakeoff(double comand_heading, double actual_heading)
33 {
34
35     static double err = 0;
36     static double err_integral = 0;
37     static double err_derivative = 0;
38     static double err_previous = 0;
39     if (actual_heading >= comand_heading)
40     {
41
42         if (actual_heading - comand_heading >= 180)
43         {
44             err = -(actual_heading - comand_heading);
45         }
46         else
47         {
48             err = (actual_heading - comand_heading);
49         }
50
51     }
52     else
53     {
54         if (comand_heading - actual_heading >= 180)
55         {
56             err = -(actual_heading - comand_heading);
57         }
58         else
59         {
60             err = (actual_heading - comand_heading);
```

```
61     }
62 }
63 err_integral += err;
64 err_derivative = err - err_previous;
65 err_previous = err;
66 double max_err_integral = 10000/50.0;
67 if (err_integral > max_err_integral && err_integral > 0)
68 {
69     err_integral = max_err_integral;
70 }
71 else if (err_integral < -max_err_integral && err_integral < 0)
72 {
73     err_integral = -max_err_integral;
74 }
75 double rudder = 1500 * err + 50* err_integral + 0 * err_derivative;
76 if (rudder > 16383)
77 {
78     rudder = 16382;
79 }
80 else if (rudder < -16383)
81 {
82     rudder = -16382;
83 }
84
85
86 return (rudder);
87 }
88
89 double Pitchhold(double comand_pitch, double actual_pitch, double pitch_rate)
90 {
91     static double time = 0;
92     static double previoustime = 0;
93     static double elevator = 0;
```

```
94     static double previouselevator = 0;
95
96     time = UserPlane.simtime;
97
98     static double err = 0;
99     static double err_previous = 0;
100    static double err2 = 0;
101    static double err_previous2 = 0;
102
103    //pitch negativo a cabrare
104    err = comand_pitch - actual_pitch;
105
106    //lead controller
107    double comand_pitch_rate = 9000 * (err + (0.3 * (err - err_previous) / (time - previoustime)));
108
109    err2 = comand_pitch_rate - 4500*pitch_rate;
110
111    //lag
112    double alpha = (time - previoustime) / ((time - previoustime) + 0.8);
113    elevator = alpha * err2 + ((1 - alpha) * previouselevator);
114
115    if (elevator > 16383)
116    {
117        elevator = 16383;
118    }
119    else if (elevator < -16383)
120    {
121        elevator = -16383;
122    }
123
124    err_previous = err;
125    err_previous2 = err2;
126    previoustime = time;
```

```
127     if (!std::isnan(elevator))
128     {
129         previouslevator = elevator;
130     }
131
132     return (elevator);
133 }
134
135 double Bankhold(double comanded_bank, double actual_bank, double roll_rate)
136 {
137     static double time = 0;
138     static double previoustime = 0;
139     static double aileron = 0;
140     static double previousaileron = 0;
141
142     time = UserPlane.simtime;
143
144     static double err = 0;
145     static double err_previous = 0;
146     static double err2 = 0;
147     static double err_previous2 = 0;
148
149
150     err = comanded_bank - actual_bank;
151
152     //lead controller
153     double comand_roll_rate = 400 * (err + (0.64* (err - err_previous) / (time - previoustime)));
154
155     err2 = comand_roll_rate - 200 * roll_rate;
156
157     //lag
158     double alpha = (time - previoustime) / ((time - previoustime) + 0.7);
159     aileron = alpha * err2 + ((1 - alpha) * previousaileron);
```

```
160
161     if (aileron > 16383)
162     {
163         aileron = 16383;
164     }
165     else if (aileron < -16383)
166     {
167         aileron = -16383;
168     }
169
170     err_previous = err;
171     err_previous2 = err2;
172     previoustime = time;
173     if (!std::isnan(aileron))
174     {
175         previousaileron = aileron;
176     }
177
178     return (aileron);
179 }
180
181 //autopilota di controllo della quota
182 double Altitudehold(double comand_altitude, double actual_altitude, double max_pitch, double actual_pitch, double ↗
    pitch_rate)
183 {
184     static double time = 0;
185     static double previoustime = 0;
186     time = UserPlane.simtime;
187
188     static double altitude_err = 0;
189     static double altitude_err_integral = 0;
190     static double altitude_err_derivative = 0;
191     static double altitude_err_previous = 0;
```

```
192     static double comand_pitch = 0;
193     static double saturated_pitch = 0;
194     static double windup = 0;
195
196     altitude_err = actual_altitude - comand_altitude;
197
198     //protezione dall'inizializzazione
199     if ((time - previoustime) > 1)
200     {
201         altitude_err_derivative = 0;
202         altitude_err_integral = 0;
203     }
204     else
205     {
206
207         if (!std::isnan(comand_pitch - saturated_pitch))
208         {
209             windup = (comand_pitch - saturated_pitch);
210
211         }
212         else
213         {
214             windup = 0.0;
215
216         }
217         altitude_err_derivative = (altitude_err - altitude_err_previous) / (time - previoustime);
218         altitude_err_integral += ((altitude_err + altitude_err_previous) * (time - previoustime) / 2) - 1 * windup;
219     }
220
221     altitude_err_previous = altitude_err;
222     comand_pitch = 0.08 * altitude_err + 0.01 * altitude_err_integral + 0.09 * altitude_err_derivative;
223
224
```

```
225
226     if (comand_pitch > max_pitch && comand_pitch > 0)
227     {
228         saturated_pitch = max_pitch;
229     }
230     else if (comand_pitch < -max_pitch && comand_pitch < 0)
231     {
232         saturated_pitch = -max_pitch;
233     }
234     else
235     {
236         saturated_pitch = comand_pitch;
237     }
238
239     static double elevator = 0;
240     static double previouslevator = 0;
241
242     static double err = 0;
243     static double err_previous = 0;
244     static double err2 = 0;
245     static double err_previous2 = 0;
246
247     //pitch negativo a cabrare
248     err = saturated_pitch - actual_pitch;
249
250     //lead controller
251     double comand_pitch_rate = 9000 * (err + (0.3 * (err - err_previous) / (time - previoustime)));
252
253     err2 = comand_pitch_rate - 4500 * pitch_rate;
254
255     //lag
256     double alpha = (time - previoustime) / ((time - previoustime) + 0.8);
257     elevator = alpha * err2 + ((1 - alpha) * previouslevator);
```



```
258
259     if (elevator > 16383)
260     {
261         elevator = 16383;
262     }
263     else if (elevator < -16383)
264     {
265         elevator = -16383;
266     }
267
268     err_previous = err;
269     err_previous2 = err2;
270     previoustime = time;
271     if (!std::isnan(elevator))
272     {
273         previouselevator = elevator;
274     }
275
276     return (elevator);
277 }
278
279 // autopilota dell'angolo di heading con controllo del solo alettone
280 double Headinghold(double comand_heading, double actual_heading, double max_bank, double actual_bank, double
    roll_rate)
281 {
282     static double time = 0;
283     static double previoustime = 0;
284     time = UserPlane.simtime;
285
286     static double heading_err = 0;
287     static double heading_err_derivative = 0;
288     static double heading_err_previous = 0;
289     static double heading_err_integral = 0;
```

```
290     static double comanded_bank = 0;
291     static double saturated_bank = 0;
292     static double windup = 0;
293
294     if (actual_heading >= comand_heading)
295     {
296
297         if (actual_heading - comand_heading >= 180)
298         {
299             heading_err = (comand_heading-actual_heading);
300         }
301         else
302         {
303             heading_err = -(comand_heading - actual_heading);
304         }
305     }
306     else
307     {
308
309         if (comand_heading - actual_heading >= 180)
310         {
311             heading_err = (comand_heading-actual_heading);
312         }
313         else
314         {
315             heading_err = -(comand_heading- actual_heading);
316         }
317     }
318
319     if ((time - previoustime) > 1)
320     {
321         heading_err_derivative = 0;
322         heading_err_integral = 0;
```

```
323     }
324     else
325     {
326
327         if (!std::isnan(comanded_bank - saturated_bank))
328         {
329             windup = (comanded_bank - saturated_bank);
330         }
331         else
332         {
333             windup = 0.0;
334         }
335         heading_err_derivative = (heading_err - heading_err_previous) / (time - previoustime);
336         heading_err_integral += ((heading_err + heading_err_previous) * (time - previoustime) / 2) - 0.001 *
337             windup;
338     }
339     heading_err_previous = heading_err;
340     comanded_bank = 10*heading_err + 0.00 * heading_err_integral + 0 * heading_err_derivative;
341
342     if (comanded_bank > max_bank && comanded_bank > 0)
343     {
344         saturated_bank = max_bank;
345     }
346     else if (comanded_bank < -max_bank && comanded_bank < 0)
347     {
348         saturated_bank = -max_bank;
349     }
350     else
351     {
352         saturated_bank = comanded_bank;
353     }
354
```

```
355
356     static double aileron = 0;
357     static double previousaileron = 0;
358
359     static double err = 0;
360     static double err_previous = 0;
361     static double err2 = 0;
362     static double err_previous2 = 0;
363
364
365     err = saturated_bank - actual_bank;
366
367     //lead controller
368     double comand_roll_rate = 400 * (err + (0.6 * (err - err_previous) / (time - previoustime)));
369
370     err2 = comand_roll_rate - 200 * roll_rate;
371
372     //lag
373     double alpha = (time - previoustime) / ((time - previoustime) + 0.7);
374     aileron = alpha * err2 + ((1 - alpha) * previousaileron);
375
376     if (aileron > 16383)
377     {
378         aileron = 16383;
379     }
380     else if (aileron < -16383)
381     {
382         aileron = -16383;
383     }
384
385     err_previous = err;
386     err_previous2 = err2;
387     previoustime = time;
```

```
388     if (!std::isnan(aileron))
389     {
390         previousaileron = aileron;
391     }
392
393     return (aileron);
394 }
395
396
397
398
399 double Autothrottle(double comand_velocity, double actual_velocity, double acceleration)
400 {
401     static double time = 0;
402     static double previoustime = 0;
403     static double throttle = 0;
404     static double previousthrottle = 0;
405
406     time = UserPlane.simtime;
407
408     static double err = 0;
409     static double err_previous = 0;
410     static double err2 = 0;
411     static double err_previous2 = 0;
412
413
414     err = comand_velocity - actual_velocity;
415
416     //lead controller
417     double comand_throttle = 3000 * (err + (0.27 * (err - err_previous) / (time - previoustime)));
418
419     err2 = comand_throttle - 1500 * acceleration;
420
```

```
421 //lag
422 double alpha = (time - previoustime) / ((time - previoustime) + 0);
423 throttle = alpha * err2 + ((1 - alpha) * previousthrottle);
424
425 if (throttle > 16383*0.9)
426 {
427     throttle = 16383*0.9;
428 }
429 else if (throttle < 16383*0.3)
430 {
431     throttle = 16383*0.3;
432 }
433 err_previous = err;
434 err_previous2 = err2;
435 previoustime = time;
436 if (!std::isnan(throttle))
437 {
438     previousthrottle = throttle;
439 }
440
441 return (throttle);
442 }
443
444 //autopilota di controllo della quota separazione verticale positiva = drone + in basso
445 double VerticalSeparation(std::vector<double> NED, double verticalseparation, double max_pitch, double
    actual_pitch, double pitch_rate)
446 {
447     static double time = 0;
448     static double previoustime = 0;
449     time = UserPlane.simtime;
450
451     static double altitude_err = 0;
452     static double altitude_err_integral = 0;
```

```
453     static double altitude_err_derivative = 0;
454     static double altitude_err_previous = 0;
455     static double comand_pitch = 0;
456     static double saturated_pitch = 0;
457     static double windup = 0;
458
459     altitude_err = -(NED[2] - verticalseparation)* 3.28084;
460
461     //protezione dall'inizializzazione
462     if ((time - previoustime) > 1)
463     {
464         altitude_err_derivative = 0;
465         altitude_err_integral = 0;
466     }
467     else
468     {
469
470         if (!std::isnan(comand_pitch - saturated_pitch))
471         {
472             windup = (comand_pitch - saturated_pitch);
473         }
474         else
475         {
476             windup = 0;
477         }
478
479         altitude_err_derivative = (altitude_err - altitude_err_previous) / (time - previoustime);
480         altitude_err_integral += ((altitude_err + altitude_err_previous) * (time - previoustime) / 2) - 1 * windup;
481     }
482
483     altitude_err_previous = altitude_err;
484     comand_pitch = 0.08 * altitude_err + 0.01 * altitude_err_integral + 0.09 * altitude_err_derivative;
485
```

```
486     if (comand_pitch > max_pitch && comand_pitch > 0)
487     {
488         saturated_pitch = max_pitch;
489     }
490     else if (comand_pitch < -max_pitch && comand_pitch < 0)
491     {
492         saturated_pitch = -max_pitch;
493     }
494     else
495     {
496         saturated_pitch = comand_pitch;
497     }
498
499     static double elevator = 0;
500     static double previouslevator = 0;
501
502     static double err = 0;
503     static double err_previous = 0;
504     static double err2 = 0;
505     static double err_previous2 = 0;
506
507     //pitch negativo a cabrare
508     err = saturated_pitch - actual_pitch;
509
510     //lead controller
511     double comand_pitch_rate = 9000 * (err + (0.2 * (err - err_previous) / (time - previoustime)));
512
513     err2 = comand_pitch_rate - 4500 * pitch_rate;
514
515     //lag
516     double alpha = (time - previoustime) / ((time - previoustime) + 0.8);
517     elevator = alpha * err2 + ((1 - alpha) * previouslevator);
518
```



```
519     if (elevator > 16383)
520     {
521         elevator = 16383;
522     }
523     else if (elevator < -16383)
524     {
525         elevator = -16383;
526     }
527
528     err_previous = err;
529     err_previous2 = err2;
530     previoustime = time;
531     if (!std::isnan(elevator))
532     {
533         previouselevator = elevator;
534     }
535
536     return (elevator);
537 }
538
539
540 double LateralSeparationApproach(std::vector<double> NED, double max_bank, double actual_bank, double
    actual_heading, double roll_rate)
541 {
542     double comand_heading;
543     double comanded_bank;
544     double distance;
545     distance = sqrt(NED[1] * NED[1] + NED[0] * NED[0] + NED[2] * NED[2]);
546     double forwardist = 0;
547     forwardist = abs(NED[0]);
548
549     static double heading_err = 0;
550     static double heading_err_derivative = 0;
```

```
551     static double heading_err_previous = 0;
552     static double heading_err_integral = 0;
553
554     comand_heading = -atan((NED[1]) / forwarddist) * (180.0 / PI);
555
556
557     if (comand_heading < 0)
558     {
559         comand_heading = comand_heading + 360;
560     }
561
562     if (actual_heading >= comand_heading)
563     {
564
565         if (actual_heading - comand_heading >= 180)
566         {
567             heading_err = -(actual_heading - comand_heading);
568         }
569         else
570         {
571             heading_err = (actual_heading - comand_heading);
572         }
573     }
574
575     else
576     {
577         if (comand_heading - actual_heading >= 180)
578         {
579             heading_err = -(actual_heading - comand_heading);
580         }
581         else
582         {
583             heading_err = (actual_heading - comand_heading);
```

```
584     }
585 }
586
587 static double time = 0;
588 static double previoustime = 0;
589 time = UserPlane.simtime;
590
591
592
593
594
595 heading_err_derivative = (heading_err - heading_err_previous) / (time - previoustime);
596 heading_err_integral += (heading_err + heading_err_previous) * (time - previoustime) / 2;
597 heading_err_previous = heading_err;
598
599 comanded_bank = 2 * heading_err + 0 * heading_err_integral + 0 * heading_err_derivative;
600
601
602 if (comanded_bank > max_bank && comanded_bank > 0)
603 {
604     comanded_bank = max_bank;
605 }
606 else if (comanded_bank < -max_bank && comanded_bank < 0)
607 {
608     comanded_bank = -max_bank;
609 }
610 static double aileron = 0;
611 static double previousaileron = 0;
612
613
614 static double err = 0;
615 static double err_previous = 0;
616 static double err2 = 0;
```

```
617     static double err_previous2 = 0;
618
619     //pitch negativo a cabrare
620     err = comanded_bank - actual_bank;
621
622
623     //lead controller
624     double comand_roll_rate = 250 * (err + (0.1 * (err - err_previous) / (time - previoustime)));
625
626
627
628     err2 = comand_roll_rate - 125 * roll_rate;
629
630
631     //lag
632     double alpha = (time - previoustime) / ((time - previoustime) + 1);
633
634     aileron = alpha * err2 + ((1 - alpha) * previousaileron);
635
636
637     if (aileron > 16383)
638     {
639         aileron = 16383;
640     }
641     else if (aileron < -16383)
642     {
643         aileron = -16383;
644     }
645
646     err_previous = err;
647     err_previous2 = err2;
648     previoustime = time;
649     if (!std::isnan(aileron))
```

```
650     {
651         previousaileron = aileron;
652     }
653
654     return (aileron);
655 }
656
657 double LateralSeparation(std::vector<double> NED, double comand_separation, double max_bank, double actual_bank,
658     double roll_rate)
659 {
660     static double previousned = 0;
661     static double comanded_bank=0;
662     static double saturated_bank = 0;
663     static double lateral_err = 0;
664     static double lateral_err_derivative = 0;
665     static double lateral_err_previous = 0;
666     static double lateral_err_integral = 0;
667     static double windup = 0;
668     static double aileron = 0;
669     static double previousaileron = 0;
670
671
672     static double time = 0;
673     static double previoustime = 0;
674     time = UserPlane.simtime;
675
676     lateral_err = -(comand_separation - NED[1]);
677     if ((time - previoustime) > 1)
678     {
679         lateral_err_derivative = 0;
680         lateral_err_integral = 0;
681     }
```

```
682     else
683     {
684
685         if (!std::isnan(comanded_bank - saturated_bank))
686         {
687             windup = (comanded_bank - saturated_bank);
688         }
689         else
690         {
691             windup = 0;
692         }
693
694         lateral_err_derivative = (lateral_err - lateral_err_previous) / (time - previoustime);
695         lateral_err_integral += ((lateral_err + lateral_err_previous) * (time - previoustime) / 2) - 0.001 * windup;
696     }
697
698     lateral_err_previous = lateral_err;
699     comanded_bank = 1 * lateral_err + 0.001 * lateral_err_integral + 3.6 * lateral_err_derivative;
700
701     if (comanded_bank > max_bank && comanded_bank > 0)
702     {
703         saturated_bank = max_bank;
704     }
705     else if (comanded_bank < -max_bank && comanded_bank < 0)
706     {
707         saturated_bank = -max_bank;
708     }
709     else
710     {
711         saturated_bank = comanded_bank;
712     }
713     static double err = 0;
```

```
714     static double err_previous = 0;
715     static double err2 = 0;
716     static double err_previous2 = 0;
717
718
719     err = saturated_bank - actual_bank;
720
721     //lead controller
722     double comand_roll_rate = 400 * (err + (0.6 * (err - err_previous) / (time - previoustime)));
723
724     err2 = comand_roll_rate - 200 * roll_rate;
725
726     //lag
727     double alpha = (time - previoustime) / ((time - previoustime) + 0.7);
728     aileron = alpha * err2 + ((1 - alpha) * previousaileron);
729
730     if (aileron > 16383)
731     {
732         aileron = 16383;
733     }
734     else if (aileron < -16383)
735     {
736         aileron = -16383;
737     }
738
739     err_previous = err;
740     err_previous2 = err2;
741     previoustime = time;
742     if (!std::isnan(aileron))
743     {
744         previousaileron = aileron;
745     }
746
```

```
747     return (aileron);
748 }
749
750 double ForwardSeparation(std::vector<double> NED, double comand_separation, double actual_velocity, double
    acceleration)
751 {
752
753     static double time = 0;
754     static double previoustime = 0;
755     time = UserPlane.simtime;
756
757     static double forward_err = 0;
758     static double forward_err_integral = 0;
759     static double forward_err_derivative = 0;
760     static double forward_err_previous = 0;
761     static double windup = 0;
762     static double throttle = 0;
763     static double comand_velocity = 0;
764     static double saturated_velocity = 0;
765     static double previousthrottle = 0;
766     forward_err = comand_separation - NED[0];
767
768     //protezione dall'inizializzazione
769     if ((time - previoustime) > 1)
770     {
771         forward_err_derivative = 0;
772         forward_err_integral = 0;
773     }
774     else
775     {
776
777         if (!std::isnan(comand_velocity - saturated_velocity))
778         {
```



```
779         windup = (comand_velocity - saturated_velocity);
780     }
781     else
782     {
783         windup = 0;
784     }
785     forward_err_derivative = (forward_err - forward_err_previous) / (time - previoustime);
786     forward_err_integral += ((forward_err + forward_err_previous) * (time - previoustime) / 2) - 0.5* windup;
787 }
788 forward_err_previous = forward_err;
789
790
791
792 comand_velocity = 1 * forward_err + 0.03 * forward_err_integral + 2 * forward_err_derivative;
793
794
795 if (comand_velocity > 1060)
796 {
797     saturated_velocity = 1060;
798 }
799 else if (comand_velocity < 600)
800 {
801     saturated_velocity = 600;
802 }
803 else
804 {
805     saturated_velocity = comand_velocity;
806 }
807
808 printf("time %.1f\t err%.5f\t int%.5f\t comand_velocity%.5f\n", UserPlane.simtime, forward_err,
809     forward_err_integral, comand_velocity);
810
```

```
811     static double err = 0;
812     static double err_previous = 0;
813     static double err2 = 0;
814     static double err_previous2 = 0;
815
816
817     err = saturated_velocity - actual_velocity;
818
819     //lead controller
820     double comand_throttle = 3000 * (err + (0.27 * (err - err_previous) / (time - previoustime)));
821
822     err2 = comand_throttle - 1500 * acceleration;
823
824     //lag
825     double alpha = (time - previoustime) / ((time - previoustime) + 0);
826     throttle = alpha * err2 + ((1 - alpha) * previousthrottle);
827
828     if (throttle > 16383 * 0.9)
829     {
830         throttle = 16383 * 0.9;
831     }
832     else if (throttle < 16383 * 0.3)
833     {
834         throttle = 16383 * 0.3;
835     }
836     printf("\nthrottle %f", throttle);
837     err_previous = err;
838     err_previous2 = err2;
839     previoustime = time;
840     if (!std::isnan(throttle))
841     {
842         previousthrottle = throttle;
843     }
```

```
844  
845     return (throttle);  
846 }  
847
```