

vedderb / rise\_sdvp

## Prepared for motor controller simulation

[Browse files](#)

master

vedderb committed 11 days ago

1 parent df3dd9e commit 993154bd58db75c883eca3d1cbc1f79f980493c5

Showing 16 changed files with 161 additions and 4 deletions.

Unified Split

## 3 Embedded/RC\_Controller/CHANGELOG

```
...    ...    @@ -1,3 +1,6 @@
      1    +=== FW 8.13 ===
      2    +* Motor simulation support.
      3    +
1    4    === FW 8.12 ===
2    5    * MPU9250 support.
3    6    * Larger stack for polled UART logging.
```

## 3 Embedded/RC\_Controller/Makefile

```
157    157    pwm_esc.c \
158    158    mr_control.c \
159    159    actuator.c \
160    -    radar_cont.c
      160    +    radar_cont.c \
      161    +    motor_sim.c
161    162
162    163    # C++ sources that can be compiled in ARM or THUMB mode depending on the global
163    164    # setting.
```

## 40 Embedded/RC\_Controller/bldc\_interface.c

```
70    70    static void(*rx_dec_chuk_func)(float val) = 0;
71    71    static void(*rx_mconf_received_func)(void) = 0;
72    72    static void(*rx_appconf_received_func)(void) = 0;
      73    +static void(*motor_control_set_func)(motor_control_mode mode, float value) = 0;
      74    +static void(*values_requested_func)(void) = 0;
73    75
74    76    void bldc_interface_init(void(*func)(unsigned char *data, unsigned int len)) {
75    77        send_func = func;
511    513        rx_appconf_received_func = func;
512    514    }
513    515
      516    +void bldc_interface_set_sim_control_function(void(*func)(motor_control_mode mode, float value)) {
      517    +    motor_control_set_func = func;
      518    +}
      519    +
      520    +void bldc_interface_set_sim_values_func(void(*func)(void)) {
      521    +    values_requested_func = func;
      522    +}
      523    +
514    524    // Setters
515    525    void bldc_interface_terminal_cmd(char* cmd) {
516    526        int len = strlen(cmd);
520    530    }
521    531
522    532    void bldc_interface_set_duty_cycle(float dutyCycle) {
      533    +    if (motor_control_set_func) {
      534    +        motor_control_set_func(MOTOR_CONTROL_DUTY, dutyCycle);
      535    +        return;
      536    +    }
```

```

523 537         int32_t send_index = 0;
524 538         send_buffer[send_index++] = COMM_SET_DUTY;
525 539         buffer_append_float32(send_buffer, dutyCycle, 100000.0, &send_index);
526 540         send_packet_no_fwd(send_buffer, send_index);
527 541     }
528 542
529 543     void bldc_interface_set_current(float current) {
544 +         if (motor_control_set_func) {
545 +             motor_control_set_func(MOTOR_CONTROL_CURRENT, current);
546 +             return;
547 +         }
530 548         int32_t send_index = 0;
531 549         send_buffer[send_index++] = COMM_SET_CURRENT;
532 550         buffer_append_float32(send_buffer, current, 1000.0, &send_index);
533 551         send_packet_no_fwd(send_buffer, send_index);
534 552     }
535 553
536 554     void bldc_interface_set_current_brake(float current) {
555 +         if (motor_control_set_func) {
556 +             motor_control_set_func(MOTOR_CONTROL_CURRENT_BRAKE, current);
557 +             return;
558 +         }
537 559         int32_t send_index = 0;
538 560         send_buffer[send_index++] = COMM_SET_CURRENT_BRAKE;
539 561         buffer_append_float32(send_buffer, current, 1000.0, &send_index);
540 562         send_packet_no_fwd(send_buffer, send_index);
541 563     }
542 564
543 565     void bldc_interface_set_rpm(int rpm) {
566 +         if (motor_control_set_func) {
567 +             motor_control_set_func(MOTOR_CONTROL_RPM, rpm);
568 +             return;
569 +         }
544 570         int32_t send_index = 0;
545 571         send_buffer[send_index++] = COMM_SET_RPM;
546 572         buffer_append_int32(send_buffer, rpm, &send_index);
547 573         send_packet_no_fwd(send_buffer, send_index);
548 574     }
549 575
550 576     void bldc_interface_set_pos(float pos) {
577 +         if (motor_control_set_func) {
578 +             motor_control_set_func(MOTOR_CONTROL_POS, pos);
579 +             return;
580 +         }
551 581         int32_t send_index = 0;
552 582         send_buffer[send_index++] = COMM_SET_POS;
553 583         buffer_append_float32(send_buffer, pos, 1000000.0, &send_index);
755 785     }
756 786
757 787     void bldc_interface_get_values(void) {
788 +         if (values_requested_func) {
789 +             values_requested_func();
790 +             return;
791 +         }
758 792         int32_t send_index = 0;
759 793         send_buffer[send_index++] = COMM_GET_VALUES;
760 794         send_packet_no_fwd(send_buffer, send_index);
812 846         send_packet_no_fwd(send_buffer, send_index);
813 847     }
814 848
849 +void send_values_to_receiver(mc_values *values) {
850 +    if (rx_value_func) {
851 +        rx_value_func(values);
852 +    }

```

```

853 +}
854 +
815 855 // Helpers
816 856 const char* bldc_interface_fault_to_string(mc_fault_code fault) {
817 857     switch (fault) {

```

#### 4 Embedded/RC\_Controller/bldc\_interface.h

```

41 41 void bldc_interface_set_rx_mcconf_received_func(void(*func)(void));
42 42 void bldc_interface_set_rx_appconf_received_func(void(*func)(void));
43 43
44 44 +void bldc_interface_set_sim_control_function(void(*func)(motor_control_mode mode, float value));
45 45 +void bldc_interface_set_sim_values_func(void(*func)(void));
46 46 +
44 47 // Setters
45 48 void bldc_interface_terminal_cmd(char* cmd);
46 49 void bldc_interface_set_duty_cycle(float dutyCycle);
66 69 void bldc_interface_detect_motor_param(float current, float min_rpm, float low_duty);
67 70 void bldc_interface_reboot(void);
68 71 void bldc_interface_send_alive(void);
72 72 +void send_values_to_receiver(mc_values *values);
69 73
70 74 // Helpers
71 75 const char* bldc_interface_fault_to_string(mc_fault_code fault);

```

#### 7 Embedded/RC\_Controller/commands.c

```

36 36 #include "comm_cc1120.h"
37 37 #include "mr_control.h"
38 38 #include "adconv.h"
39 39 ++include "motor_sim.h"
39 40
40 41 #include <math.h>
41 42 #include <string.h>
568 569 main_config.car.yaw_use_odometry = data[ind++];
569 570 main_config.car.yaw_imu_gain = buffer_get_float32_auto(data, &ind);
570 571 main_config.car.disable_motor = data[ind++];
572 572 + main_config.car.simulate_motor = data[ind++];
571 573
572 574 main_config.car.gear_ratio = buffer_get_float32_auto(data, &ind);
573 575 main_config.car.wheel_diam = buffer_get_float32_auto(data, &ind);
578 580 main_config.car.steering_ramp_time = buffer_get_float32_auto(data, &ind);
579 581 main_config.car.axis_distance = buffer_get_float32_auto(data, &ind);
580 582
583 583 +if MAIN_MODE == MAIN_MODE_CAR
584 584 +    motor_sim_set_running(main_config.car.simulate_motor);
585 585 +endif
586 586 +
581 587 // Multirotor settings
582 588 main_config.mr.vel_decay_e = buffer_get_float32_auto(data, &ind);
583 589 main_config.mr.vel_decay_l = buffer_get_float32_auto(data, &ind);
709 715 m_send_buffer[send_index++] = main_cfg_tmp.car.yaw_use_odometry;
710 716 buffer_append_float32_auto(m_send_buffer, main_cfg_tmp.car.yaw_imu_gain, &send_index);
711 717 m_send_buffer[send_index++] = main_cfg_tmp.car.disable_motor;
718 718 + m_send_buffer[send_index++] = main_cfg_tmp.car.simulate_motor;
712 719
713 720 buffer_append_float32_auto(m_send_buffer, main_cfg_tmp.car.gear_ratio, &send_index);
714 721 buffer_append_float32_auto(m_send_buffer, main_cfg_tmp.car.wheel_diam, &send_index);

```

#### 1 Embedded/RC\_Controller/conf\_general.c

```

117 117 conf->car.yaw_use_odometry = false;
118 118 conf->car.yaw_imu_gain = 0.5;
119 119 conf->car.disable_motor = false;
120 120 + conf->car.simulate_motor = false;

```

120	121	
121	122	conf->car.gear_ratio = (1.0 / 3.0) * (21.0 / 37.0);
122	123	conf->car.wheel_diam = 0.12;

#### 4 Embedded/RC\_Controller/conf\_general.h

```

37 37
38 38 // Firmware version
39 39 #define FW_VERSION_MAJOR 8
40 40 -#define FW_VERSION_MINOR 11
41 41 +#define FW_VERSION_MINOR 13
42 42
43 43 // Default car settings
44 44 // #define CAR_TERO // Benjamins tero car
45 45 // #define EBIKE_BENJAMIN // Benjamins ebike
46 46
47 47 // Defaults for different cars
48 47 #ifdef CAR_TERO
49 48 -#define BOARD_ROT_180 1
50 48 +#define BOARD_ROT_180 1
51 49 #endif
52 50
53 51 // Ublox settings

```

#### 9 Embedded/RC\_Controller/datatypes.h

```

208 208 bool yaw_use_odometry; // Use odometry data for yaw angle correction.
209 209 float yaw_imu_gain; // Gain for yaw angle from IMU (vs odometry)
210 210 bool disable_motor; // Disable motor drive commands to make sure that the motor does not move.
211 211 + bool simulate_motor; // Simulate motor movement without motor controller feedback
212 212
213 213 float gear_ratio;
214 214 float wheel_diam;
215 215
216 216 DRV8301_OC_DISABLED
217 217 } drv8301_oc_mode;
218 218
219 219 +typedef enum {
220 220 + MOTOR_CONTROL_DUTY = 0,
221 221 + MOTOR_CONTROL_CURRENT,
222 222 + MOTOR_CONTROL_CURRENT_BRAKE,
223 223 + MOTOR_CONTROL_RPM,
224 224 + MOTOR_CONTROL_POS
225 225 +} motor_control_mode;
226 226 +
227 227
228 228 typedef struct {
229 229 // Switching and drive
230 230 mc_pwm_mode pwm_mode;
231 231
232 232
233 233
234 234
235 235
236 236
237 237
238 238
239 239
240 240
241 241
242 242
243 243
244 244
245 245
246 246
247 247
248 248
249 249
250 250
251 251
252 252
253 253
254 254
255 255
256 256
257 257
258 258
259 259
260 260
261 261
262 262
263 263
264 264
265 265
266 266
267 267
268 268
269 269
270 270
271 271
272 272
273 273
274 274
275 275
276 276
277 277
278 278
279 279
280 280
281 281
282 282
283 283
284 284
285 285
286 286
287 287
288 288
289 289
290 290
291 291
292 292
293 293
294 294
295 295
296 296
297 297
298 298
299 299
300 300
301 301
302 302
303 303
304 304
305 305
306 306
307 307
308 308
309 309
310 310
311 311
312 312
313 313
314 314
315 315
316 316
317 317
318 318
319 319
320 320
321 321
322 322
323 323
324 324
325 325
326 326
327 327
328 328
329 329
330 330
331 331
332 332
333 333
334 334
335 335
336 336
337 337
338 338
339 339
340 340
341 341
342 342
343 343
344 344
345 345
346 346
347 347
348 348
349 349
350 350
351 351
352 352
353 353
354 354
355 355
356 356
357 357
358 358
359 359
360 360
361 361
362 362
363 363
364 364
365 365
366 366
367 367
368 368
369 369
370 370
371 371
372 372
373 373
374 374
375 375
376 376
377 377
378 378
379 379
380 380
381 381
382 382
383 383
384 384
385 385
386 386
387 387
388 388
389 389
390 390
391 391
392 392
393 393
394 394
395 395
396 396
397 397
398 398
399 399
400 400
401 401
402 402
403 403
404 404
405 405
406 406
407 407
408 408
409 409
410 410
411 411
412 412
413 413
414 414
415 415
416 416
417 417
418 418
419 419
420 420
421 421
422 422
423 423
424 424
425 425
426 426
427 427
428 428
429 429
430 430
431 431
432 432
433 433
434 434
435 435
436 436
437 437
438 438
439 439
440 440
441 441
442 442
443 443
444 444
445 445
446 446
447 447
448 448
449 449
450 450
451 451
452 452
453 453
454 454
455 455
456 456
457 457
458 458
459 459
460 460
461 461
462 462
463 463
464 464
465 465
466 466
467 467
468 468
469 469
470 470
471 471
472 472
473 473
474 474
475 475
476 476
477 477
478 478
479 479
480 480
481 481
482 482
483 483
484 484
485 485
486 486
487 487
488 488
489 489
490 490
491 491
492 492
493 493
494 494
495 495
496 496
497 497
498 498
499 499
500 500
501 501
502 502
503 503
504 504
505 505
506 506
507 507
508 508
509 509
510 510
511 511
512 512
513 513
514 514
515 515
516 516
517 517
518 518
519 519
520 520
521 521
522 522
523 523
524 524
525 525
526 526
527 527
528 528
529 529
530 530
531 531
532 532
533 533
534 534
535 535
536 536
537 537
538 538
539 539
540 540
541 541
542 542
543 543
544 544
545 545
546 546
547 547
548 548
549 549
550 550
551 551
552 552
553 553
554 554
555 555
556 556
557 557
558 558
559 559
560 560
561 561
562 562
563 563
564 564
565 565
566 566
567 567
568 568
569 569
570 570
571 571
572 572
573 573
574 574
575 575
576 576
577 577
578 578
579 579
580 580
581 581
582 582
583 583
584 584
585 585
586 586
587 587
588 588
589 589
590 590
591 591
592 592
593 593
594 594
595 595
596 596
597 597
598 598
599 599
600 600
601 601
602 602
603 603
604 604
605 605
606 606
607 607
608 608
609 609
610 610
611 611
612 612
613 613
614 614
615 615
616 616
617 617
618 618
619 619
620 620
621 621
622 622
623 623
624 624
625 625
626 626
627 627
628 628
629 629
630 630
631 631
632 632
633 633
634 634
635 635
636 636
637 637
638 638
639 639
640 640
641 641
642 642
643 643
644 644
645 645
646 646
647 647
648 648
649 649
650 650
651 651
652 652
653 653
654 654
655 655
656 656
657 657
658 658
659 659
660 660
661 661
662 662
663 663
664 664
665 665
666 666
667 667
668 668
669 669
670 670
671 671
672 672
673 673
674 674
675 675
676 676
677 677
678 678
679 679
680 680
681 681
682 682
683 683
684 684
685 685
686 686
687 687
688 688
689 689
690 690
691 691
692 692
693 693
694 694
695 695
696 696
697 697
698 698
699 699
700 700
701 701
702 702
703 703
704 704
705 705
706 706
707 707
708 708
709 709
710 710
711 711
712 712
713 713
714 714
715 715
716 716
717 717
718 718
719 719
720 720
721 721
722 722
723 723
724 724
725 725
726 726
727 727
728 728
729 729
730 730
731 731
732 732
733 733
734 734
735 735
736 736
737 737
738 738
739 739
740 740
741 741
742 742
743 743
744 744
745 745
746 746
747 747
748 748
749 749
750 750
751 751
752 752
753 753
754 754
755 755
756 756
757 757
758 758
759 759
760 760
761 761
762 762
763 763
764 764
765 765
766 766
767 767
768 768
769 769
770 770
771 771
772 772
773 773
774 774
775 775
776 776
777 777
778 778
779 779
780 780
781 781
782 782
783 783
784 784
785 785
786 786
787 787
788 788
789 789
790 790
791 791
792 792
793 793
794 794
795 795
796 796
797 797
798 798
799 799
800 800
801 801
802 802
803 803
804 804
805 805
806 806
807 807
808 808
809 809
810 810
811 811
812 812
813 813
814 814
815 815
816 816
817 817
818 818
819 819
820 820
821 821
822 822
823 823
824 824
825 825
826 826
827 827
828 828
829 829
830 830
831 831
832 832
833 833
834 834
835 835
836 836
837 837
838 838
839 839
840 840
841 841
842 842
843 843
844 844
845 845
846 846
847 847
848 848
849 849
850 850
851 851
852 852
853 853
854 854
855 855
856 856
857 857
858 858
859 859
860 860
861 861
862 862
863 863
864 864
865 865
866 866
867 867
868 868
869 869
870 870
871 871
872 872
873 873
874 874
875 875
876 876
877 877
878 878
879 879
880 880
881 881
882 882
883 883
884 884
885 885
886 886
887 887
888 888
889 889
890 890
891 891
892 892
893 893
894 894
895 895
896 896
897 897
898 898
899 899
900 900
901 901
902 902
903 903
904 904
905 905
906 906
907 907
908 908
909 909
910 910
911 911
912 912
913 913
914 914
915 915
916 916
917 917
918 918
919 919
920 920
921 921
922 922
923 923
924 924
925 925
926 926
927 927
928 928
929 929
930 930
931 931
932 932
933 933
934 934
935 935
936 936
937 937
938 938
939 939
940 940
941 941
942 942
943 943
944 944
945 945
946 946
947 947
948 948
949 949
950 950
951 951
952 952
953 953
954 954
955 955
956 956
957 957
958 958
959 959
960 960
961 961
962 962
963 963
964 964
965 965
966 966
967 967
968 968
969 969
970 970
971 971
972 972
973 973
974 974
975 975
976 976
977 977
978 978
979 979
980 980
981 981
982 982
983 983
984 984
985 985
986 986
987 987
988 988
989 989
990 990
991 991
992 992
993 993
994 994
995 995
996 996
997 997
998 998
999 999
1000 1000

```

#### 6 Embedded/RC\_Controller/main.c

```

48 48 #include "pwm_esc.h"
49 49 #include "mr_control.h"
50 50 #include "radar_cont.h"
51 51 +#include "motor_sim.h"
52 52
53 53 /*
54 54 * Timers used:
55 55
56 56
57 57
58 58
59 59
60 60
61 61
62 62
63 63
64 64
65 65
66 66
67 67
68 68
69 69
70 70
71 71
72 72
73 73
74 74
75 75
76 76
77 77
78 78
79 79
80 80
81 81
82 82
83 83
84 84
85 85
86 86
87 87
88 88
89 89
90 90 autopilot_init();
91 91 timeout_init();
92 92 log_init();
93 93
94 94 + motor_sim_init();
95 95
96 96 #if RADAR_EN
97 96 radar_init();
98 97 radar_setup_measurement_default();
99 98
100 99
101 100 ublox_init();

```

```
122 124 #endif
123 125
126 + #if MAIN_MODE == MAIN_MODE_CAR
127 +     motor_sim_set_running(main_config.car.simulate_motor);
128 + #endif
129 +
130     timeout_configure(2000, 20.0);
131     log_set_rate(main_config.log_rate_hz);
132     log_set_enabled(main_config.log_en);
```

64 Embedded/RC\_Controller/motor\_sim.c

```
... ... @@ -0,0 +1,64 @@
1 +
2 + #include <math.h>
3 +
4 + #include "motor_sim.h"
5 + #include "ch.h"
6 + #include "hal.h"
7 + #include "bldc_interface.h"
8 +
9 + // Settings
10 + #define SIMULATION_TIME_MS 10
11 +
12 + // Private variables
13 + static bool m_is_running;
14 +
15 + // Private functions
16 + static void motor_control_set(motor_control_mode mode, float value);
17 + static void motor_values_requested(void);
18 +
19 + // Threads
20 + static THD_WORKING_AREA(sim_thread_wa, 2048);
21 + static THD_FUNCTION(sim_thread, arg);
22 +
23 + void motor_sim_init(void) {
24 +     m_is_running = false;
25 +     chThdCreateStatic(sim_thread_wa, sizeof(sim_thread_wa), NORMALPRIO, sim_thread, NULL);
26 + }
27 +
28 + void motor_sim_set_running(bool running) {
29 +     m_is_running = running;
30 +
31 +     if (m_is_running) {
32 +         bldc_interface_set_sim_control_function(motor_control_set);
33 +         bldc_interface_set_sim_values_func(motor_values_requested);
34 +     } else {
35 +         bldc_interface_set_sim_control_function(0);
36 +         bldc_interface_set_sim_values_func(0);
37 +     }
38 + }
39 +
40 + static THD_FUNCTION(sim_thread, arg) {
41 +     (void) arg;
42 +
43 +     chRegSetThreadName("MotorSim");
44 +
45 +     systime_t iteration_timer = chVTGetSystemTime();
46 +
47 +     for(;;) {
48 +         if (m_is_running) {
49 +             // TODO!
50 +         }
51 +     }
```

```
52 +             iteration_timer = chThdSleepUntilWindowed(iteration_timer,
53 +                                                         iteration_timer + MS2ST(SIMULATION_TIME_MS));
54 +         }
55 +     }
56 +
57 +static void motor_control_set(motor_control_mode mode, float value) {
58 +    (void)mode;
59 +    (void)value;
60 +}
61 +
62 +static void motor_values_requested(void) {
63 +
64 +}
```

10 Embedded/RC\_Controller/motor\_sim.h

```
...    ...    @@ -0,0 +1,10 @@
1    +ifndef MOTOR_SIM_H_
2    +define MOTOR_SIM_H_
3    +
4    +include "datatypes.h"
5    +
6    +// Fucntions
7    +void motor_sim_init(void);
8    +void motor_sim_set_running(bool running);
9    +
10   +endif /* MOTOR_SIM_H_ */
```

2 Embedded/RC\_Controller/pos.c

```
15    15    along with this program. If not, see <http://www.gnu.org/licenses/>.
16    16    */
17    17
18    18    -include <ahrs.h>
19    18    include <math.h>
20    19    include <string.h>
21    20    include <stdio.h>
22    21    include <stdlib.h>
23    22    include "ch.h"
24    23    include "hal.h"
24    24    +include "ahrs.h"
25    25    include "stm32f4xx_conf.h"
26    26    include "led.h"
27    27    include "mpu9150.h"
```

2 Linux/RControlStation/carinterface.cpp

```
794    794    conf.car.yaw_use_odometry = ui->confOdometryYawBox->isChecked();
795    795    conf.car.yaw_imu_gain = ui->confYawImuGainBox->value();
796    796    conf.car.disable_motor = ui->confMiscDisableMotorBox->isChecked();
797    797    + conf.car.simulate_motor = ui->confMiscSimulateMotorBox->isChecked();
798    798
799    799    conf.car.gear_ratio = ui->confGearRatioBox->value();
800    800    conf.car.wheel_diam = ui->confWheelDiamBox->value();
813    814    ui->confOdometryYawBox->setChecked(conf.car.yaw_use_odometry);
814    815    ui->confYawImuGainBox->setValue(conf.car.yaw_imu_gain);
815    816    ui->confMiscDisableMotorBox->setChecked(conf.car.disable_motor);
817    817    + ui->confMiscSimulateMotorBox->setChecked(conf.car.simulate_motor);
818    818
819    819    ui->confGearRatioBox->setValue(conf.car.gear_ratio);
820    820    ui->confWheelDiamBox->setValue(conf.car.wheel_diam);
```

7 Linux/RControlStation/carinterface.ui

```
586    586    </property>
```

```
587 587         </widget>
588 588     </item>
589 +
590 +     <item row="4" column="1">
591 +         <widget class="QCheckBox" name="confMiscSimulateMotorBox">
592 +             <property name="text">
593 +                 <string>Simulate Motor</string>
594 +             </property>
595 +         </widget>
596 +     </item>
589 596 </layout>
590 597 </widget>
591 598 </item>
```

**1** Linux/RControlStation/datatypes.h

```
107 107     bool yaw_use_odometry; // Use odometry data for yaw angle correction.
108 108     float yaw_imu_gain; // Gain for yaw angle from IMU (vs odometry)
109 109     bool disable_motor; // Disable motor drive commands to make sure that the motor does not move.
110 + 110     bool simulate_motor; // Simulate motor movement without motor controller feedback
111 111
112 112     float gear_ratio;
112 113     float wheel_diam;
```

**2** Linux/RControlStation/packetinterface.cpp

```
442 442     conf.car.yaw_use_odometry = data[ind++];
443 443     conf.car.yaw_imu_gain = utility::buffer_get_double32_auto(data, &ind);
444 444     conf.car.disable_motor = data[ind++];
445 + 445     conf.car.simulate_motor = data[ind++];
446 446
447 447     conf.car.gear_ratio = utility::buffer_get_double32_auto(data, &ind);
448 448     conf.car.wheel_diam = utility::buffer_get_double32_auto(data, &ind);
873 874     mSendBuffer[send_index++] = conf.car.yaw_use_odometry;
874 875     utility::buffer_append_double32_auto(mSendBuffer, conf.car.yaw_imu_gain, &send_index);
875 876     mSendBuffer[send_index++] = conf.car.disable_motor;
877 + 877     mSendBuffer[send_index++] = conf.car.simulate_motor;
878 878
879 879     utility::buffer_append_double32_auto(mSendBuffer, conf.car.gear_ratio, &send_index);
880 880     utility::buffer_append_double32_auto(mSendBuffer, conf.car.wheel_diam, &send_index);
```

0 comments on commit 993154b