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_mcconf_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 +       }
```

```
+}
+
 // Helpers
 const char* bldc_interface_fault_to_string(mc_fault_code fault) {
        switch (fault) {
```

4 Embedded/RC_Controller/bldc_interface.h

```
 void bldc_interface_set_rx_mcconf_received_func(void(*func)(void));
 void bldc_interface_set_rx_appconf_received_func(void(*func)(void));

+void bldc_interface_set_sim_control_function(void(*func)(motor_control_mode mode, float value));
+void bldc_interface_set_sim_values_func(void(*func)(void));
+
 // Setters
 void bldc_interface_terminal_cmd(char* cmd);
 void bldc_interface_set_duty_cycle(float dutyCycle);
 void bldc_interface_detect_motor_param(float current, float min_rpm, float low_duty);
 void bldc_interface_reboot(void);
 void bldc_interface_send_alive(void);
+void send_values_to_receiver(mc_values *values);

 // Helpers
 const char* bldc_interface_fault_to_string(mc_fault_code fault);
```

7 Embedded/RC_Controller/commands.c

```
 #include "comm_cc1120.h"
 #include "mr_control.h"
 #include "adconv.h"
+#include "motor_sim.h"

 #include <math.h>
 #include <string.h>
                        main_config.car.yaw_use_odometry = data[ind++];
                        main_config.car.yaw_imu_gain = buffer_get_float32_auto(data, &ind);
                        main_config.car.disable_motor = data[ind++];
+                       main_config.car.simulate_motor = data[ind++];

                        main_config.car.gear_ratio = buffer_get_float32_auto(data, &ind);
                        main_config.car.wheel_diam = buffer_get_float32_auto(data, &ind);
                        main_config.car.steering_ramp_time = buffer_get_float32_auto(data, &ind);
                        main_config.car.axis_distance = buffer_get_float32_auto(data, &ind);

+#if MAIN_MODE == MAIN_MODE_CAR
+                       motor_sim_set_running(main_config.car.simulate_motor);
+#endif
+
                        // Multirotor settings
                        main_config.mr.vel_decay_e = buffer_get_float32_auto(data, &ind);
                        main_config.mr.vel_decay_l = buffer_get_float32_auto(data, &ind);
                        m_send_buffer[send_index++] = main_cfg_tmp.car.yaw_use_odometry;
                        buffer_append_float32_auto(m_send_buffer, main_cfg_tmp.car.yaw_imu_gain, &send_index);
                        m_send_buffer[send_index++] = main_cfg_tmp.car.disable_motor;
+                       m_send_buffer[send_index++] = main_cfg_tmp.car.simulate_motor;

                        buffer_append_float32_auto(m_send_buffer, main_cfg_tmp.car.gear_ratio, &send_index);
                        buffer_append_float32_auto(m_send_buffer, main_cfg_tmp.car.wheel_diam, &send_index);
```

1 Embedded/RC_Controller/conf_general.c

```
        conf->car.yaw_use_odometry = false;
        conf->car.yaw_imu_gain = 0.5;
        conf->car.disable_motor = false;
+       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      -#define FW_VERSION_MINOR                    11
    40  +#define FW_VERSION_MINOR                    13
41  41
42  42  // Default car settings
43  43  //#define CAR_TERO // Benjamins tero car
44  44  //#define EBIKE_BENJAMIN // Benjamins ebike
45  45
46  46  // Defaults for different cars
47  47  #ifdef CAR_TERO
48      -#define BOARD_ROT_180                       1
    48  +#define BOARD_ROT_180                       1
49  49  #endif
50  50
51  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 +       bool simulate_motor; // Simulate motor movement without motor controller feedback
211 212
212 213         float gear_ratio;
213 214         float wheel_diam;
781 782         DRV8301_OC_DISABLED
782 783  } drv8301_oc_mode;
783 784
    785 +typedef enum {
    786 +       MOTOR_CONTROL_DUTY = 0,
    787 +       MOTOR_CONTROL_CURRENT,
    788 +       MOTOR_CONTROL_CURRENT_BRAKE,
    789 +       MOTOR_CONTROL_RPM,
    790 +       MOTOR_CONTROL_POS
    791 +} motor_control_mode;
    792 +
784 793  typedef struct {
785 794         // Switching and drive
786 795         mc_pwm_mode pwm_mode;
```

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  +#include "motor_sim.h"
51  52
52  53  /*
53  54   * Timers used:
90  91         autopilot_init();
91  92         timeout_init();
92  93         log_init();
    94  +       motor_sim_init();
93  95  #if RADAR_EN
94  96         radar_init();
95  97         radar_setup_measurement_default();
121 123        ublox_init();
```

```
 #endif

+#if MAIN_MODE == MAIN_MODE_CAR
+	motor_sim_set_running(main_config.car.simulate_motor);
+#endif
+
	timeout_configure(2000, 20.0);
	log_set_rate(main_config.log_rate_hz);
	log_set_enabled(main_config.log_en);
```

64 Embedded/RC_Controller/motor_sim.c

```
@@ -0,0 +1,64 @@
+
+#include <math.h>
+
+#include "motor_sim.h"
+#include "ch.h"
+#include "hal.h"
+#include "bldc_interface.h"
+
+// Settings
+#define SIMULATION_TIME_MS 10
+
+// Private variables
+static bool m_is_running;
+
+// Private functions
+static void motor_control_set(motor_control_mode mode, float value);
+static void motor_values_requested(void);
+
+// Threads
+static THD_WORKING_AREA(sim_thread_wa, 2048);
+static THD_FUNCTION(sim_thread, arg);
+
+void motor_sim_init(void) {
+	m_is_running = false;
+	chThdCreateStatic(sim_thread_wa, sizeof(sim_thread_wa), NORMALPRIO, sim_thread, NULL);
+}
+
+void motor_sim_set_running(bool running) {
+	m_is_running = running;
+
+	if (m_is_running) {
+		bldc_interface_set_sim_control_function(motor_control_set);
+		bldc_interface_set_sim_values_func(motor_values_requested);
+	} else {
+		bldc_interface_set_sim_control_function(0);
+		bldc_interface_set_sim_values_func(0);
+	}
+}
+
+static THD_FUNCTION(sim_thread, arg) {
+	(void)arg;
+
+	chRegSetThreadName("MotorSim");
+
+	systime_t iteration_timer = chVTGetSystemTime();
+
+	for(;;) {
+		if (m_is_running) {
+			// TODO!
+		}
+
```

```
+              iteration_timer = chThdSleepUntilWindowed(iteration_timer,
+                                iteration_timer + MS2ST(SIMULATION_TIME_MS));
+       }
+}
+
+static void motor_control_set(motor_control_mode mode, float value) {
+       (void)mode;
+       (void)value;
+}
+
+static void motor_values_requested(void) {
+
+}
```

10 Embedded/RC_Controller/motor_sim.h

```
@@ -0,0 +1,10 @@
+#ifndef MOTOR_SIM_H_
+#define MOTOR_SIM_H_
+
+#include "datatypes.h"
+
+// Fucntions
+void motor_sim_init(void);
+void motor_sim_set_running(bool running);
+
+#endif /* MOTOR_SIM_H_ */
```

2 Embedded/RC_Controller/pos.c

```
     along with this program.  If not, see <http://www.gnu.org/licenses/>.
  */

-#include <ahrs.h>
 #include <math.h>
 #include <string.h>
 #include <stdio.h>
 #include <stdlib.h>
 #include "ch.h"
 #include "hal.h"
+#include "ahrs.h"
 #include "stm32f4xx_conf.h"
 #include "led.h"
 #include "mpu9150.h"
```

2 Linux/RControlStation/carinterface.cpp

```
    conf.car.yaw_use_odometry = ui->confOdometryYawBox->isChecked();
    conf.car.yaw_imu_gain = ui->confYawImuGainBox->value();
    conf.car.disable_motor = ui->confMiscDisableMotorBox->isChecked();
+   conf.car.simulate_motor = ui->confMiscSimulateMotorBox->isChecked();

    conf.car.gear_ratio = ui->confGearRatioBox->value();
    conf.car.wheel_diam = ui->confWheelDiamBox->value();
    ui->confOdometryYawBox->setChecked(conf.car.yaw_use_odometry);
    ui->confYawImuGainBox->setValue(conf.car.yaw_imu_gain);
    ui->confMiscDisableMotorBox->setChecked(conf.car.disable_motor);
+   ui->confMiscSimulateMotorBox->setChecked(conf.car.simulate_motor);

    ui->confGearRatioBox->setValue(conf.car.gear_ratio);
    ui->confWheelDiamBox->setValue(conf.car.wheel_diam);
```

7 Linux/RControlStation/carinterface.ui

```
              </property>
```

```
587 587              </widget>
588 588             </item>
    589 +           <item row="4" column="1">
    590 +            <widget class="QCheckBox" name="confMiscSimulateMotorBox">
    591 +             <property name="text">
    592 +              <string>Simulate Motor</string>
    593 +             </property>
    594 +            </widget>
    595 +           </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 +   bool simulate_motor; // Simulate motor movement without motor controller feedback
110 111
111 112     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 +       conf.car.simulate_motor = data[ind++];
445 446
446 447         conf.car.gear_ratio = utility::buffer_get_double32_auto(data, &ind);
447 448         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 +   mSendBuffer[send_index++] = conf.car.simulate_motor;
876 878
877 879     utility::buffer_append_double32_auto(mSendBuffer, conf.car.gear_ratio, &send_index);
878 880     utility::buffer_append_double32_auto(mSendBuffer, conf.car.wheel_diam, &send_index);
```

0 comments on commit `993154b`