Arduino {code}racer API

Generated by Doxygen 1.8.13

Contents

1 Module Index **1**

1.1 Modules 1

2 Module Documentation **3**

2.1 Higher level methods, setters and getters 3

2.1.1 Detailed Description 3

2.2 Methods 4

2.2.1 Detailed Description 4

2.2.2 Function Documentation 4

2.2.2.1 stop_driving() 5

2.2.2.2 drive_forward() [1/2] 5

2.2.2.3 drive_forward() [2/2] 5

2.2.2.4 drive_backward() [1/2] 6

2.2.2.5 drive_backward() [2/2] 6

2.2.2.6 turn_left() [1/3] 7

2.2.2.7 turn_left() [2/3] 7

2.2.2.8 turn_left() [3/3] 8

2.2.2.9 turn_right() [1/3] 9

2.2.2.10 turn_right() [2/3] 9

2.2.2.11 turn_right() [3/3] 10

2.2.2.12 start_stop_at_min_distance() [1/2] 10

2.2.2.13 start_stop_at_min_distance() [2/2] 11

2.2.2.14 stop_stop_at_min_distance() 11

2.2.2.15 start_stop() 12

2.3 Getters and setters . . . 13
2.3.1 Detailed Description . . . 13
2.3.2 Function Documentation . . . 13
2.3.2.1 stopped_at_min_distance() . . . 13
2.3.2.2 is_driving() . . . 14
2.3.2.3 turn_left_for_ms() . . . 14
2.3.2.4 turn_right_for_ms() . . . 14
2.3.2.5 set_inactive() . . . 15
2.3.2.6 set_active() . . . 15
2.3.2.7 is_active() . . . 15
2.4 Lower level fun stuff methods . . . 16
2.4.1 Detailed Description . . . 16
2.4.2 Function Documentation . . . 16
2.4.2.1 look_around() . . . 16
2.4.2.2 kitt() . . . 17
2.5 Lower level servo drive methods and getters . . . 18
2.5.1 Detailed Description . . . 18
2.6 Methods . . . 19
2.6.1 Detailed Description . . . 19
2.6.2 Function Documentation . . . 19
2.6.2.1 servo_settings() . . . 19
2.6.2.2 servo_sweep() . . . 20
2.6.2.3 servo_set_to_right() . . . 21
2.6.2.4 servo_set_to_left() . . . 21
2.6.2.5 servo_set_to_center() . . . 21
2.6.2.6 servo_set_position_wait() . . . 21
2.6.2.7 servo_set_position() . . . 22
2.7 Getters . . . 23
2.7.1 Detailed Description . . . 23
2.7.2 Function Documentation . . . 23

2.7.2.1 servo_position() . . . 23
2.7.2.2 servo_position_set_at_ms() . . . 23
2.7.2.3 servo_position_eta_in_ms() . . . 24
2.8 Lower level ultra sonic methods and getters . . . 25
2.8.1 Detailed Description . . . 25
2.9 Methods . . . 26
2.9.1 Detailed Description . . . 26
2.9.2 Function Documentation . . . 26
2.9.2.1 usonic_measure_cm() [1/2] . . . 26
2.9.2.2 usonic_measure_us() [1/2] . . . 27
2.9.2.3 usonic_measure_cm() [2/2] . . . 27
2.9.2.4 usonic_measure_us() [2/2] . . . 28
2.9.2.5 usonic_measure_single_shot_cm() [1/2] . . . 28
2.9.2.6 usonic_measure_single_shot_us() [1/2] . . . 29
2.9.2.7 usonic_measure_single_shot_cm() [2/2] . . . 29
2.9.2.8 usonic_measure_single_shot_us() [2/2] . . . 30
2.10 Setters and getters . . . 31
2.10.1 Detailed Description . . . 31
2.10.2 Function Documentation . . . 31
2.10.2.1 usonic_set_stop_distance_cm() . . . 31
2.10.2.2 usonic_set_stop_distance_us() . . . 32
2.10.2.3 usonic_distance_us() . . . 32
2.10.2.4 usonic_distance_cm() . . . 32
2.11 Lower level drives methods and getters . . . 33
2.11.1 Detailed Description . . . 33
2.12 Methods . . . 34
2.12.1 Detailed Description . . . 34
2.12.2 Function Documentation . . . 34
2.12.2.1 drives_settings() . . . 34
2.12.2.2 set_drives_stop_left_right() . . . 35

2.12.2.3 set_drives_states_left_right() . . . 35
2.12.2.4 set_drive_left_state() . . . 36
2.12.2.5 set_drive_right_state() . . . 36
2.12.2.6 set_drive_state() . . . 37
2.12.2.7 set_drives_speed_left_right() . . . 37
2.12.2.8 set_drive_left_speed() . . . 38
2.12.2.9 set_drive_right_speed() . . . 38
2.12.2.10 set_drive_speed() . . . 39
2.13 Getters . . . 40
2.13.1 Detailed Description . . . 40
2.13.2 Function Documentation . . . 40
2.13.2.1 drive_right_speed() . . . 40
2.13.2.2 drive_left_speed() . . . 40
2.14 Lower level LED methods . . . 41
2.14.1 Detailed Description . . . 41
2.15 Methods . . . 42
2.15.1 Detailed Description . . . 42
2.15.2 Function Documentation . . . 42
2.15.2.1 set_leds_left_stop_frwd_right() . . . 42
2.15.2.2 set_leds_all() . . . 43
2.15.2.3 set_leds_all_off() . . . 43
2.15.2.4 set_leds_all_on() . . . 43

Index **45**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Higher level methods, setters and getters . . . 3
Methods . . . 4
Getters and setters . . . 13
Lower level fun stuff methods . . . 16
Lower level servo drive methods and getters . . . 18
Methods . . . 19
Getters . . . 23
Lower level ultra sonic methods and getters . . . 25
Methods . . . 26
Setters and getters . . . 31
Lower level drives methods and getters . . . 33
Methods . . . 34
Getters . . . 40
Lower level LED methods . . . 41
Methods . . . 42

Chapter 2

Module Documentation

2.1 Higher level methods, setters and getters

Modules

- Methods
- Getters and setters

2.1.1 Detailed Description

2.2 Methods

Functions

- void CodeRacer::stop_driving ()

  Stops the racer and sets status LEDs.
- void CodeRacer::drive_forward ()

  Sets the speed and the directions of both drives so that it will move forward.
- void CodeRacer::drive_forward (uint8_t left_speed, uint8_t right_speed)

  Sets the speed as specified for both drives and the directions of both drives so that it will move forward.
- void CodeRacer::drive_backward ()

  Sets the speed and the directions of both drives so that it will move backward.
- void CodeRacer::drive_backward (uint8_t left_speed, uint8_t right_speed)

  Sets the speed as specified for both drives and the directions of both drives so that it will move backward.
- void CodeRacer::turn_left ()

  Will turn the racer to the left for the internally stroe time in ms and with the internally stored speed.
- void CodeRacer::turn_left (unsigned long turn_for_ms)

  Will turn the racer to the left for the specified time in ms and with the internally stored speed.
- void CodeRacer::turn_left (unsigned long turn_for_ms, uint8_t left_speed, uint8_t right_speed)

  Will turn the racer to the left for the specified time in ms and with the specified speed.
- void CodeRacer::turn_right ()

  Will turn the racer to the right for the internally stroe time in ms and with the internally stored speed.
- void CodeRacer::turn_right (unsigned long turn_for_ms)

  Will turn the racer to the right for the specified time in ms and with the internally stored speed.
- void CodeRacer::turn_right (unsigned long turn_for_ms, uint8_t left_speed, uint8_t right_speed)

  Will turn the racer to the right for the specified time in ms and with the specified speed.
- void CodeRacer::start_stop_at_min_distance ()

  Enables to stopp the racer if during a distance measurement the measured distance is smaller then the internally stored minimal distance.
- void CodeRacer::start_stop_at_min_distance (unsigned long min_distance_cm)

  Enables to stopp the racer if during a distance measurement the measured distance is smaller then the specified minimal distance.
- void CodeRacer::stop_stop_at_min_distance ()

  Disables to stopp the racer if during a distance measurement the measured distance is smaller then the specified minimal distance.
- bool CodeRacer::start_stop ()

  This will return if the codracer is in active mode or not.

2.2.1 Detailed Description

2.2.2 Function Documentation

2.2.2.1 stop_driving()

```
void CodeRacer::stop_driving ( )
```

Stops the racer and sets status LEDs.

Returns

nothing

Definition at line 161 of file CodeRacer.cpp.

```
161                             {
162     _servo_sweep = false;
163     _drive = false;
164     set_drives_stop_left_right();
165     set_leds_left_stop_frwd_right(LEDOFF, LEDON, LEDOFF, LEDOFF);
166 }
```

2.2.2.2 drive_forward() [1/2]

```
void CodeRacer::drive_forward ( )
```

Sets the speed and the directions of both drives so that it will move forward.

The speed is taken from the header file or set by one of the methods defined in the lower level drive methods section

Returns

nothing

Definition at line 173 of file CodeRacer.cpp.

```
174 {
175     drive_forward(_drive_left_speed, _drive_right_speed);
176 }
```

Here is the call graph for this function:

2.2.2.3 drive_forward() [2/2]

```
void CodeRacer::drive_forward (
            uint8_t left_speed,
            uint8_t right_speed )
```

Sets the speed as specified for both drives and the directions of both drives so that it will move forward.

The specified speed values for both drives will be stored internaly so next time if you use e.g. drive_forward() exactly the here specified values will be taken.

Parameters

left_speed	speed for the left side drive. 0<=speed<=255
right_speed	speed for the right side drive. 0<=speed<=255

Returns

nothing

Definition at line 185 of file CodeRacer.cpp.

```
186 {
187     set_drives_states_left_right(DRIVEFRWD, DRIVEFRWD);
188     set_drives_speed_left_right(left_speed, right_speed);
189     set_leds_left_stop_frwd_right(LEDOFF, LEDOFF, LEDON, LEDOFF);
190     _drive = true;
191     _drive_set_at_ms = millis();
192 }
```

2.2.2.4 drive_backward() [1/2]

```
void CodeRacer::drive_backward ( )
```

Sets the speed and the directions of both drives so that it will move backward.

The speed is taken from the header file or set by one of the methods defined in the lower level drive methods section

Returns

nothing

Definition at line 199 of file CodeRacer.cpp.

```
200 {
201     drive_backward(_drive_left_speed, _drive_right_speed);
202 }
```

2.2.2.5 drive_backward() [2/2]

```
void CodeRacer::drive_backward (
            uint8_t left_speed,
            uint8_t right_speed )
```

Sets the speed as specified for both drives and the directions of both drives so that it will move backward.

The specified speed values for both drives will be stored internaly so next time if you use e.g. drive_backward() exactly the here specified values will be taken.

Parameters

left_speed	speed for the left side drive. 0<=speed<=255
right_speed	speed for the right side drive. 0<=speed<=255

Returns

nothing

Definition at line 211 of file CodeRacer.cpp.

```
212 {
213     set_drives_states_left_right(DRIVEBACK, DRIVEBACK);
214     set_drives_speed_left_right(left_speed, right_speed);
215     set_leds_left_stop_frwd_right(LEDOFF, LEDON, LEDON, LEDOFF);
216     _drive = true;
217     _drive_set_at_ms = millis();
218 }
```

2.2.2.6 turn_left() [1/3]

```
void CodeRacer::turn_left ( )
```

Will turn the racer to the left for the internally stroe time in ms and with the internally stored speed.

The speed for both the left side and right side drive can be set by the methods described in the lower level drive section. The turn to left time can be set by thoose methods as well. The method is delayed until the racer has turned.

Returns

nothing

Definition at line 226 of file CodeRacer.cpp.

```
227 {
228     turn_left(_turn_left_for_ms, _drive_left_speed, _drive_right_speed);
229 }
```

2.2.2.7 turn_left() [2/3]

```
void CodeRacer::turn_left (
            unsigned long turn_for_ms )
```

Will turn the racer to the left for the specified time in ms and with the internally stored speed.

The specified duration of time is stored internally and will be used next time by e.g. turn_left()

Parameters

turn_for_ms	duration of time in ms to turn the racer

Returns

nothing

Definition at line 237 of file CodeRacer.cpp.

```
238 {
239     turn_left(turn_for_ms, _drive_left_speed, _drive_right_speed);
240 }
```

2.2.2.8 turn_left() [3/3]

```
void CodeRacer::turn_left (
            unsigned long turn_for_ms,
            uint8_t left_speed,
            uint8_t right_speed )
```

Will turn the racer to the left for the specified time in ms and with the specified speed.

The specified duration of time and the specified speeds are stored internally and will be used next time by e.g. turn_left()

Parameters

turn_for_ms	duration of time in ms to turn the racer
left_speed	speed for the left side drive
right_speed	speed for the right side drive

Returns

nothing

Definition at line 250 of file CodeRacer.cpp.

```
251 {
252     _drive = false;
253     set_leds_left_stop_frwd_right(LEDON, LEDOFF, LEDOFF, LEDOFF);   // LEDs setzen
254     set_drives_states_left_right(DRIVEBACK, DRIVEFRWD);
255     set_drives_speed_left_right(left_speed, right_speed);
256     // wait set delay for the timed turn
257     _turn_left_for_ms = turn_for_ms;
258     delay(_turn_left_for_ms);
259     // stop drives again
260     set_drives_stop_left_right();
261 }
```

2.2.2.9 turn_right() [1/3]

```
void CodeRacer::turn_right ( )
```

Will turn the racer to the right for the internally stroe time in ms and with the internally stored speed.

The speed for both the left side and right side drive can be set by the methods described in the lower level drive section. The turn to right time can be set by thoose methods as well. The method is delayed until the racer has turned.

Returns

nothing

Definition at line 269 of file CodeRacer.cpp.

```
270 {
271     turn_right(_turn_right_for_ms, _drive_left_speed, _drive_right_speed);
272 }
```

2.2.2.10 turn_right() [2/3]

```
void CodeRacer::turn_right (
            unsigned long turn_for_ms )
```

Will turn the racer to the right for the specified time in ms and with the internally stored speed.

The specified duration of time is stored internally and will be used next time by e.g. turn_right()

Parameters

turn_for_ms	duration of time in ms to turn the racer

Returns

nothing

Definition at line 280 of file CodeRacer.cpp.

```
281 {
282     turn_right(turn_for_ms, _drive_left_speed, _drive_right_speed);
283 }
```

2.2.2.11 turn_right() [3/3]

```
void CodeRacer::turn_right (
            unsigned long turn_for_ms,
            uint8_t left_speed,
            uint8_t right_speed )
```

Will turn the racer to the right for the specified time in ms and with the specified speed.

The specified duration of time and the specified speeds are stored internally and will be used next time by e.g. turn_right()

Parameters

turn_for_ms	duration of time in ms to turn the racer
left_speed	speed for the left side drive
right_speed	speed for the right side drive

Returns

nothing

Definition at line 293 of file CodeRacer.cpp.

```
294 {
295     _drive = false;
296     set_leds_left_stop_frwd_right(LEDOFF, LEDOFF, LEDOFF, LEDON);       // LEDs setzen
297     set_drives_states_left_right(DRIVEFRWD, DRIVEBACK);
298     set_drives_speed_left_right(left_speed, right_speed);
299     // wait set delay for the timed turn
300     _turn_right_for_ms = turn_for_ms;
301     delay(_turn_right_for_ms);
302     // stop drives again
303     set_drives_stop_left_right();
304 }
```

2.2.2.12 start_stop_at_min_distance() [1/2]

```
void CodeRacer::start_stop_at_min_distance ( )
```

Enables to stopp the racer if during a distance measurement the measured distance is smaller then the internally stored minimal distance.

This allow all of the none-single shot ultra sonic measurement methods to stopp the racer in the case the measured distance is smaller than minimal distance. This is just the enablement - you have to call one of the none-single-shot ultra sonic measurement methods continously while driving and maybe sweeping the servo. If the racer was stopped can be checked with stopped_at_min_distance() - it will return true in that case. The minimal distance can be set by the ultrasonic sensor setter methods. There is an example coming with the coderacer libary that you can load and get an idea how that works.

Returns

nothing

Definition at line 314 of file CodeRacer.cpp.

```
314                                         {
315     start_stop_at_min_distance(_usonic_stop_distance_cm);
316 }
```

2.2.2.13 start_stop_at_min_distance() [2/2]

```
void CodeRacer::start_stop_at_min_distance (
            unsigned long min_distance_cm )
```

Enables to stopp the racer if during a distance measurement the measured distance is smaller then the specified minimal distance.

This is almos the same as described for start_stop_at_min_distance(). You can specify the distance in cm here - this value will be stored internally and used by start_stop_at_min_distance() next time.

Parameters

min_distance_cm	the minimal disctance in cm

Returns

nothing

Definition at line 324 of file CodeRacer.cpp.

```
324                                                                    {
325     if (false == _coderacer_stop_at_distance_enabled) {
326         _coderacer_stopped_at_min_distance = false;
327         usonic_set_stop_distance_cm(min_distance_cm);
328         _coderacer_stop_at_distance_enabled = true;
329     }
330 }
```

2.2.2.14 stop_stop_at_min_distance()

```
void CodeRacer::stop_stop_at_min_distance ( )
```

Disables to stopp the racer if during a distance measurement the measured distance is smaller then the specified minimal distance.

Returns

nothing

Definition at line 335 of file CodeRacer.cpp.

```
335                                        {
336     _coderacer_stop_at_distance_enabled = false;
337 }
```

2.2.2.15 start_stop()

```
bool CodeRacer::start_stop ( )
```

This will return if the codracer is in active mode or not.

There is a button used to toogle between active and inactive each time it is pressed This may help you to start driving and scanning the distance by pressing the button - and as well to stop the racer by pressing the button. This method will set the LEDs depending on the mode and sets the servo back to center and stopps the racer when leaving the active mode. You can leave or enter the active mode as well by setting using set_active() and set_inactive(). The button itself triggers and internall interrupt event and sets the active state - but you have to continously call that method to switch between inactive and active mode depending on the active state. If in inactive mode and fun is enabeld by just setting the coderacer_fun_enabled = true ... some fun will happen :-)

Returns

True if the coderacer is in active mode. False if in inactive mode.

Definition at line 348 of file CodeRacer.cpp.

```
348                             {
349     if (_coderracer_activ != coderracer_activ) {
350         _coderracer_activ = coderracer_activ;
351         if (true == _coderracer_activ) {
352             set_leds_all_off();
353             delay(500);
354         }
355         else {
356             stop_driving();
357             set_leds_all_on();
358             delay(200);
359             servo_set_to_center();
360             _servo_look_around_at_ms = millis() + random(20000, 120000);
361         }
362     }
363 
364     if ((false == _coderracer_activ) && (true == coderacer_fun_enabled)) {
365         kitt();
366         look_around();
367     }
368 
369     return(_coderracer_activ);
370 }
```

2.3 Getters and setters

Functions

- bool CodeRacer::stopped_at_min_distance ()

  Returns true if the racer was stopped at minimum distance. This set to false each time start_stop_at_min_distance() is called.
- bool CodeRacer::is_driving ()

  Return true if the racer is driving - forward or backward.
- unsigned long CodeRacer::turn_left_for_ms ()

  Returns the duration of time that is internally stored and used for turning the racer left.
- unsigned long CodeRacer::turn_right_for_ms ()

  Returns the duration of time that is internally stored and used for turning the racer left.
- void CodeRacer::set_inactive ()

  Sets the coderracer_active state to inactive. If start_stop() is used - the inactive mode will be entered.
- void CodeRacer::set_active ()

  Sets the coderracer_active state to active. If start_stop() is used - the active mode will be entered.
- bool CodeRacer::is_active ()

  Checks if coderracer_activ is set.

2.3.1 Detailed Description

2.3.2 Function Documentation

2.3.2.1 stopped_at_min_distance()

```
bool CodeRacer::stopped_at_min_distance ( )
```

Returns true if the racer was stopped at minimum distance. This set to false each time start_stop_at_min_distance() is called.

Returns

True if stopped.

Definition at line 380 of file CodeRacer.cpp.

```
380                                         {
381     return(_coderacer_stopped_at_min_distance);
382 }
```

2.3.2.2 is_driving()

```
bool CodeRacer::is_driving ( )
```

Return true if the racer is driving - forward or backward.

Returns

True if driving forward or backward

Definition at line 387 of file CodeRacer.cpp.

```
387                             {
388     return(_drive);
389 }
```

2.3.2.3 turn_left_for_ms()

```
unsigned long CodeRacer::turn_left_for_ms ( )
```

Returns the duration of time that is internally stored and used for turning the racer left.

Returns

Time to turn left in ms

Definition at line 394 of file CodeRacer.cpp.

```
394                                         {
395     return(_turn_left_for_ms);
396 }
```

2.3.2.4 turn_right_for_ms()

```
unsigned long CodeRacer::turn_right_for_ms ( )
```

Returns the duration of time that is internally stored and used for turning the racer left.

Returns

Time to turn left in ms

Definition at line 401 of file CodeRacer.cpp.

```
401                                          {
402     return(_turn_right_for_ms);
403 }
```

2.3.2.5 set_inactive()

```
void CodeRacer::set_inactive ( )
```

Sets the coderracer_active state to inactive. If start_stop() is used - the inactive mode will be entered.

Returns

nothing

Definition at line 408 of file CodeRacer.cpp.

```
408                              {
409     coderracer_activ = false;
410 }
```

2.3.2.6 set_active()

```
void CodeRacer::set_active ( )
```

Sets the coderracer_active state to active. If start_stop() is used - the active mode will be entered.

Returns

nothing

Definition at line 415 of file CodeRacer.cpp.

```
415                             {
416     coderracer_activ = true;
417 }
```

2.3.2.7 is_active()

```
bool CodeRacer::is_active ( )
```

Checks if coderracer_activ is set.

coderracer_activ is toggled each time the button is pressed. After power on coderracer_activ is set to False.

Returns

True id coderracer_activ is set - False if not.

Definition at line 424 of file CodeRacer.cpp.

```
424                             {
425     return(coderracer_activ);
426 }
```

2.4 Lower level fun stuff methods

Functions

- void CodeRacer::look_around ()

  Fun stuff - will move the servo around after a random amount of time.

- void CodeRacer::kitt ()

  Fun stuff - you know Knightrider...

2.4.1 Detailed Description

2.4.2 Function Documentation

2.4.2.1 look_around()

```
void CodeRacer::look_around ( )
```

Fun stuff - will move the servo around after a random amount of time.

Returns

nothing

Definition at line 440 of file CodeRacer.cpp.

```
441 {
442     if (_servo_look_around_at_ms < millis()) {
443         _servo_look_around_at_ms = millis() + random(FUN_MIN_PAUSE_MS, FUN_MAX_PAUSE_MS);
444         servo_set_to_left();
445         delay(500);
446         servo_set_to_right();
447         delay(800);
448         servo_set_to_center();
449         delay(300);
450         servo_set_to_left();
451         delay(100);
452         servo_set_to_center();
453     }
454 }
```

2.4.2.2 kitt()

```
void CodeRacer::kitt ( )
```

Fun stuff - you know Knightrider...

Returns

nothing

Definition at line 459 of file CodeRacer.cpp.

```
460 {
461     if (millis() - _last_led_switched_at_ms > LED_SWITCH_MS) {
462         _last_led_switched_at_ms = millis();
463         if (_last_led_on >= 5) {
464             _led_count = -1;
465         }
466         else if (_last_led_on <= 0) {
467             _led_count = 1;
468         }
469         _last_led_on = _last_led_on + _led_count;
470         switch (_last_led_on) {
471         case 0:
472             set_leds_left_stop_frwd_right(LEDON, LEDOFF, LEDOFF, LEDOFF);
473             break;
474         case 1:
475             set_leds_left_stop_frwd_right(LEDON, LEDOFF, LEDOFF, LEDOFF);
476             break;
477         case 2:
478             set_leds_left_stop_frwd_right(LEDOFF, LEDON, LEDOFF, LEDOFF);
479             break;
480         case 3:
481             set_leds_left_stop_frwd_right(LEDOFF, LEDOFF, LEDON, LEDOFF);
482             break;
483         case 4:
484         case 5:
485             set_leds_left_stop_frwd_right(LEDOFF, LEDOFF, LEDOFF, LEDON);
486             break;
487         }
488     }
489 }
```

2.5 Lower level servo drive methods and getters

Modules

- Methods
- Getters

2.5.1 Detailed Description

2.6 Methods

Functions

- void CodeRacer::servo_settings (uint8_t pos_center, uint8_t pos_left, uint8_t pos_right, uint8_t sweep_left↩ _pos, uint8_t sweep_right_pos)

  Overwrites the default settings taken from header file by the parameters given to this method.
- void CodeRacer::servo_sweep ()

  Turns sweeping of the servo from left to right and back on.
- void CodeRacer::servo_set_to_right ()

  Drives the servo to the postion that is defined by #servo_right_pos.
- void CodeRacer::servo_set_to_left ()

  Drives the servo to the postion that is defined by #servo_left_pos.
- void CodeRacer::servo_set_to_center ()

  Drives the servo to the postion that is defined by #servo_center_pos.
- uint8_t CodeRacer::servo_set_position_wait (uint8_t position)

  Drive the servo to the postion given to this method.
- unsigned long CodeRacer::servo_set_position (uint8_t position)

  Drive the servo to the postion given to this method.

2.6.1 Detailed Description

2.6.2 Function Documentation

2.6.2.1 servo_settings()

```
void CodeRacer::servo_settings (
            uint8_t pos_center,
            uint8_t pos_left,
            uint8_t pos_right,
            uint8_t sweep_left_pos,
            uint8_t sweep_right_pos )
```

Overwrites the default settings taken from header file by the parameters given to this method.

Parameters

pos_center	The postion at which the servo moves to straight forward. Default is 90. Allowed 10 <= pos_center <= 170.
pos_left	The postion at which the servo moves to the left. Default is 170. Allowed 10 <= pos_center <= 170.
pos_right	The postion at which the servo moves to the right. Default is 10. Allowed 10 <= pos_center <= 170.
sweep_left_pos	If the servo is sweeping from left to the right - this defines the most left postion. Default is 140. Allowed 10 <= pos_center <= 170.
sweep_right_pos	If the servo is sweeping from left to the right - this defines the most right postion. Default is 40. Allowed 10 <= pos_center <= 170.

Returns

nothing

Definition at line 510 of file CodeRacer.cpp.

```
511 {
512     servo_center_pos = pos_center;
513     servo_left_pos = pos_left;
514     servo_right_pos = pos_right;
515     servo_sweep_left_pos = sweep_left_pos;
516     servo_sweep_right_pos = sweep_right_pos;
517 }
```

2.6.2.2 servo_sweep()

```
void CodeRacer::servo_sweep ( )
```

Turns sweeping of the servo from left to right and back on.

The sweeping range is defind by #servo_sweep_left_pos and #servo_sweep_right_pos attributes. Both can be set by either servo_settings() or as public members. Every time servo_sweep() is called the servo is driven by 5 steps until either #servo_sweep_left_pos or #servo_sweep_right_pos is reached. Then it will turn the direction and step to the other side every time this method is called.

Returns

nothing

Definition at line 526 of file CodeRacer.cpp.

```
527 {
528     uint8_t position;
529     _servo_sweep = true;
530     if (millis() - _servo_position_set_at_ms > SERVO_SWEEP_MS) {
531         position = _servo_position + _servo_sweep_step;
532         //sprintf(_debugmsg,"[%s] current position=%ld newpostion=%ld", __func__, _servo_position,
      position);
533         if ((position >= servo_sweep_left_pos) || (position >= SERVO_MAX_POSITION)) {
534             position = servo_sweep_left_pos;
535             _servo_sweep_step = SERVO_SWEEP_TO_RIGHT_STEP;
536         }
537         if ((position <= servo_sweep_right_pos) || (position <= SERVO_MIN_POSITION)) {
538             position = servo_sweep_right_pos;
539             _servo_sweep_step = SERVO_SWEEP_TO_LEFT_STEP;
540         }
541         _servo_set_position(position);
542     }
543 }
```

2.6.2.3 servo_set_to_right()

```
void CodeRacer::servo_set_to_right ( )
```

Drives the servo to the postion that is defined by #servo_right_pos.

Returns

nothing

Definition at line 548 of file CodeRacer.cpp.

```
549 {
550     servo_set_position_wait(servo_right_pos);
551 }
```

2.6.2.4 servo_set_to_left()

```
void CodeRacer::servo_set_to_left ( )
```

Drives the servo to the postion that is defined by #servo_left_pos.

Returns

nothing

Definition at line 556 of file CodeRacer.cpp.

```
557 {
558     servo_set_position_wait(servo_left_pos);
559 }
```

2.6.2.5 servo_set_to_center()

```
void CodeRacer::servo_set_to_center ( )
```

Drives the servo to the postion that is defined by #servo_center_pos.

Returns

nothing

Definition at line 564 of file CodeRacer.cpp.

```
565 {
566     servo_set_position_wait(servo_center_pos);
567 }
```

2.6.2.6 servo_set_position_wait()

```
uint8_t CodeRacer::servo_set_position_wait (
            uint8_t position )
```

Drive the servo to the postion given to this method.

The method will wait until the servo has reached its new position.

Parameters

position	Position the servo will be drived to. Allowed are values 10<=postion<=170. 10 is at the right hand side, 170 at the left hand side.

Returns

The new servo position

Definition at line 575 of file CodeRacer.cpp.

```
576 {
577     _servo_sweep = false;
578     unsigned long wait_time_ms = _servo_set_position(position);
579     delay(wait_time_ms);
580     return(_servo_position);
581 }
```

2.6.2.7 servo_set_position()

```
unsigned long CodeRacer::servo_set_position (
            uint8_t position )
```

Drive the servo to the postion given to this method.

The method will not wait until the servo has reached its new position.

Parameters

position	Position the servo will be drived to. Allowed are values 10<=postion<=170. 10 is at the right hand side, 170 at the left hand side.

Returns

The time in ms the servo will need to reach the new position

Definition at line 589 of file CodeRacer.cpp.

```
590 {
591     _servo_sweep = false;
592     unsigned long wait_time_ms = _servo_set_position(position);
593     return(wait_time_ms);
594 }
```

2.7 Getters

Functions

- uint8_t CodeRacer::servo_position ()

  Get the actual position of the servo.
- unsigned long CodeRacer::servo_position_set_at_ms ()

  Get the system time in ms the servo was set to the actual position.
- unsigned long CodeRacer::servo_position_eta_in_ms ()

  Get the system time in ms the servo will reach its position This is an estimated time. If this is a time in the future, the servo may still moving. If this is a time in the past , the servo should have reached its postion already.

2.7.1 Detailed Description

2.7.2 Function Documentation

2.7.2.1 servo_position()

```
uint8_t CodeRacer::servo_position ( )
```

Get the actual position of the servo.

Returns

Actual position of the servo

Definition at line 632 of file CodeRacer.cpp.

```
632                                 {
633     return(_servo_position);
634 }
```

2.7.2.2 servo_position_set_at_ms()

```
unsigned long CodeRacer::servo_position_set_at_ms ( )
```

Get the system time in ms the servo was set to the actual position.

Returns

System time in ms the servo was set

Definition at line 639 of file CodeRacer.cpp.

```
639                                                 {
640     return(_servo_position_set_at_ms);
641 }
```

2.7.2.3 servo_position_eta_in_ms()

```
unsigned long CodeRacer::servo_position_eta_in_ms ( )
```

Get the system time in ms the servo will reach its position This is an estimated time. If this is a time in the future, the servo may still moving. If this is a time in the past , the servo should have reached its postion already.

Returns

System time in ms the servo will reach its position

Definition at line 649 of file CodeRacer.cpp.

```
649                                                 {
650     return(_servo_position_eta_in_ms);
651 }
```

2.8 Lower level ultra sonic methods and getters

Modules

- Methods
- Setters and getters

2.8.1 Detailed Description

2.9 Methods

Functions

- unsigned long CodeRacer::usonic_measure_cm ()

  Measures the distance to the next object in front of the ultra sonic sensor in cm.
- unsigned long CodeRacer::usonic_measure_us ()

  Measures the distance to the next object in front of the ultra sonic sensor in microseconds.
- unsigned long CodeRacer::usonic_measure_cm (unsigned long max_echo_run_time_us)

  Measures the distance to the next object in front of the ultra sonic sensor in cm.
- unsigned long CodeRacer::usonic_measure_us (unsigned long max_echo_run_time_us)

  Measures the distance to the next object in front of the ultra sonic sensor in microseconds.
- unsigned long CodeRacer::usonic_measure_single_shot_cm ()

  Measures the distance to the next object in front of the ultra sonic sensor in cm.
- unsigned long CodeRacer::usonic_measure_single_shot_us ()

  Measures the distance to the next object in front of the ultra sonic sensor in microseconds.
- unsigned long CodeRacer::usonic_measure_single_shot_cm (unsigned long max_echo_run_time_us)

  Measures the distance to the next object in front of the ultra sonic sensor in cm.
- unsigned long CodeRacer::usonic_measure_single_shot_us (unsigned long max_echo_run_time_us)

  Measures the distance to the next object in front of the ultra sonic sensor in microseconds.

2.9.1 Detailed Description

2.9.2 Function Documentation

2.9.2.1 usonic_measure_cm() [1/2]

```
unsigned long CodeRacer::usonic_measure_cm ( )
```

Measures the distance to the next object in front of the ultra sonic sensor in cm.

This is the medium one out of 3 measurements. The maximum measured distance is about 100cm and defined by the US_MAX_ECHO_TIME_US setting in the header file.

Returns

The measured distance in cm.

Definition at line 672 of file CodeRacer.cpp.

```
673 {
674     return(usonic_measure_cm(US_MAX_ECHO_TIME_US));
675 }
```

2.9.2.2 usonic_measure_us() [1/2]

```
unsigned long CodeRacer::usonic_measure_us ( )
```

Measures the distance to the next object in front of the ultra sonic sensor in microseconds.

This is the medium one out of 3 measurements. The maximum measured distance is about 6000 microseconds and defined by the US_MAX_ECHO_TIME_US setting in the header file.

Returns

The measured distance in microseconds.

Definition at line 682 of file CodeRacer.cpp.

```
683     {
684         return(usonic_measure_us(US_MAX_ECHO_TIME_US));
685     }
```

2.9.2.3 usonic_measure_cm() [2/2]

```
unsigned long CodeRacer::usonic_measure_cm (
             unsigned long max_echo_run_time_us )
```

Measures the distance to the next object in front of the ultra sonic sensor in cm.

This is the medium one out of 3 measurements. Be careful with high values for max_echo_run_time_us - this may increase run time due to the fact that if there is nothing in range of the sensor it will wait until this specified run time of the echo is over. The maximum range the sensor is specified for is about 300cm.

Parameters

max_echo_run_time_us	Defines the maximum echo run time and by that the maximum of distance that can be measured.

Returns

The measured distance in cm.

Definition at line 694 of file CodeRacer.cpp.

```
695 {
696     unsigned long echo_runtime_us = usonic_measure_us(max_echo_run_time_us);
697     unsigned long distance_cm = echo_runtime_us * 0.0172;
698     //Serial.print("MEASURE_DISTANCE. Distance in cm is: ");
699     //Serial.println(distance_cm);
700     _usonic_distance_cm = distance_cm;
701     return(distance_cm);
702 }
```

2.9.2.4 usonic_measure_us() [2/2]

```
unsigned long CodeRacer::usonic_measure_us (
            unsigned long max_echo_run_time_us )
```

Measures the distance to the next object in front of the ultra sonic sensor in microseconds.

This is the medium one out of 3 measurements. Be careful with high values for max_echo_run_time_us - this may increase run time due to the fact that if there is nothing in range of the sensor it will wait until this specified run time of the echo is over. The maximum range the sensor is specified for is about 300cm.

Parameters

max_echo_run_time_us	Defines the maximum echo run time in microseconds and by that the maximum of distance that can be measured.

Returns

The measured distance in microseconds.

Definition at line 711 of file CodeRacer.cpp.

```
712 {
713     unsigned long echo_runtime_us[3] = { 0,0,0 };
714     uint8_t measnr = 0;
715 
716     do {
717         echo_runtime_us[measnr] = usonic_measure_single_shot_us(max_echo_run_time_us);
718         if (echo_runtime_us[measnr] > 200) {
719             measnr++;
720         }
721     } while (measnr < 3);
722 
723     // we will take the medium out of 3 values ...
724     if (echo_runtime_us[0] > echo_runtime_us[1]) { std::swap(echo_runtime_us[0], echo_runtime_us[1]); }
725     if (echo_runtime_us[1] > echo_runtime_us[2]) { std::swap(echo_runtime_us[1], echo_runtime_us[2]); }
726     if (echo_runtime_us[0] > echo_runtime_us[1]) { std::swap(echo_runtime_us[0], echo_runtime_us[1]); }
727 
728     //Serial.print("MEASURE_DISTANCE_US. Echo runtime in us is: ");
729     //Serial.println(echo_runtime_us[1]);
730 
731     // if the stop at minimal distance is enabeled - check for minimal distance is reached
732     if (true == _coderacer_stop_at_distance_enabled) {
733         if (echo_runtime_us[1] <= _usonic_stop_distance_us) {
734             _coderacer_stopped_at_min_distance = true;
735             stop_driving();
736             _coderacer_stop_at_distance_enabled = false;
737         }
738     }
739     _usonic_distance_us = echo_runtime_us[1];
740     return(echo_runtime_us[1]);
741 }
```

2.9.2.5 usonic_measure_single_shot_cm() [1/2]

```
unsigned long CodeRacer::usonic_measure_single_shot_cm ( )
```

Measures the distance to the next object in front of the ultra sonic sensor in cm.

This is a one shot measurement. The maximum measured distance is about 6000 microseconds and defined by the US_MAX_ECHO_TIME_US setting in the header file.

Returns

The measured distance in cm.

Definition at line 748 of file CodeRacer.cpp.

```
749 {
750     return(usonic_measure_single_shot_cm(US_MAX_ECHO_TIME_US));
751 }
```

2.9.2.6 usonic_measure_single_shot_us() [1/2]

```
unsigned long CodeRacer::usonic_measure_single_shot_us ( )
```

Measures the distance to the next object in front of the ultra sonic sensor in microseconds.

This is a one shot measurement. The maximum measured distance is about 6000 microseconds and defined by the US_MAX_ECHO_TIME_US setting in the header file.

Returns

The measured distance in microseconds.

Definition at line 758 of file CodeRacer.cpp.

```
759 {
760     return(usonic_measure_single_shot_us(US_MAX_ECHO_TIME_US));
761 }
```

2.9.2.7 usonic_measure_single_shot_cm() [2/2]

```
unsigned long CodeRacer::usonic_measure_single_shot_cm (
            unsigned long max_echo_run_time_us )
```

Measures the distance to the next object in front of the ultra sonic sensor in cm.

This is a one shot measurement. Be careful with high values for max_echo_run_time_us - this may increase run time due to the fact that if there is nothing in range of the sensor it will wait until this specified run time of the echo is over. The maximum range the sensor is specified for is about 300cm.

Parameters

max_echo_run_time_us	Defines the maximum echo run time in microseconds and by that the maximum of distance that can be measured.

Returns

The measured distance in cm.

Definition at line 770 of file CodeRacer.cpp.

```
771 {
772     // convert into cm ... 344m/sec is the speed of noise - thus 34400cm/sec ... or 34,400cm/milisec ... or
      0,0344cm/microsec
773     // the echo has to go the distance twice - forth and back - so the duration has to be the half of the
      measured one
774     // distance_cm = echo_duration/2 * 0,0344   or   distance_cm = echo_duration/2 / 29,1 or distance_cm =
      echo_duration * 0,0172
775     // distance_cm = (echo_duration/2) / 29.1;
776     unsigned long echo_runtime_us = usonic_measure_single_shot_us(max_echo_run_time_us);
777     unsigned long distance_cm = echo_runtime_us * 0.0172;
778     //Serial.print("MEASURE_DISTANCE. Distance in cm is: ");
779     //Serial.println(distance_cm);
780     _usonic_distance_cm = distance_cm;
781     return(distance_cm);
782 }
```

2.9.2.8 usonic_measure_single_shot_us() [2/2]

```
unsigned long CodeRacer::usonic_measure_single_shot_us (
            unsigned long max_echo_run_time_us )
```

Measures the distance to the next object in front of the ultra sonic sensor in microseconds.

This is a one shot measurement. Be careful with high values for max_echo_run_time_us - this may increase run time due to the fact that if there is nothing in range of the sensor it will wait until this specified run time of the echo is over. The maximum range the sensor is specified for is about 300cm.

Parameters

max_echo_run_time_us	Defines the maximum echo run time in microseconds and by that the maximum of distance that can be measured.

Returns

The measured distance in microseconds.

Definition at line 791 of file CodeRacer.cpp.

```
792     {
793         unsigned long echo_runtime_us;
794         // start measurement - send a short pulse "HIGH" to the TRIG pin of the ultrasonic sensor
795         pinMode(_us_echo_pin, OUTPUT);
796         pinMode(_us_echo_pin, INPUT);
797         digitalWrite(_us_trigger_pin, LOW);
798         delayMicroseconds(2);
799         digitalWrite(_us_trigger_pin, HIGH);
800         delayMicroseconds(10);
801         digitalWrite(_us_trigger_pin, LOW);
802         // measure runtime in microseconds until the ECHO pin aof the sensor goes HIGH
803         echo_runtime_us = pulseInLong(_us_echo_pin, HIGH, max_echo_run_time_us);
804         if (echo_runtime_us == 0) {
805             echo_runtime_us = max_echo_run_time_us; // US_MAX_ECHO_TIME_US;
806         }
807         //Serial.print("MEASURE_DISTANCE_US. Echo runtime in us is: ");
808         //Serial.println(echo_runtime_us);
809         _usonic_distance_us = echo_runtime_us;
810         return(echo_runtime_us);
811     }
```

2.10 Setters and getters

Functions

- void CodeRacer::usonic_set_stop_distance_cm (unsigned long stop_distance_cm)

  Sets the stop distance in cm.
- void CodeRacer::usonic_set_stop_distance_us (unsigned long stop_distance_us)

  Sets the stop distance in cm.
- unsigned long CodeRacer::usonic_distance_us ()

  Returns the last measured distance in microseconds.
- unsigned long CodeRacer::usonic_distance_cm ()

  Returns the last measured distance in cm.

2.10.1 Detailed Description

2.10.2 Function Documentation

2.10.2.1 usonic_set_stop_distance_cm()

```
void CodeRacer::usonic_set_stop_distance_cm (
            unsigned long stop_distance_cm )
```

Sets the stop distance in cm.

If start_stop_at_min_distance() is used and distance measured with one of the measurement methods - the racer will be stopped immediately. All except the singe shot methods of the ultra sonic measurements methods supports that. Internally the stop distance will be set as both - in cm and in microseconds.

Parameters

stop_distance_cm	Distance in cm the racer will be stopped if that features was enabled by start_stop_at_min_distance() before.

Returns

nothing

Definition at line 825 of file CodeRacer.cpp.

```
826 {
827     _usonic_stop_distance_us = stop_distance_cm * 58.14;
828 }
```

2.10.2.2 usonic_set_stop_distance_us()

```
void CodeRacer::usonic_set_stop_distance_us (
            unsigned long stop_distance_us )
```

Sets the stop distance in cm.

If start_stop_at_min_distance() is used and distance measured with one of the measurement methods - the racer will be stopped immediately. All except the singe shot methods of the ultra sonic measurements methods supports that. Internally the stop distance will be set as both - in cm and in microseconds.

Parameters

stop_distance_us	Distance in cm the racer will be stopped if that features was enabled by start_stop_at_min_distance() before.

Returns

nothing

Definition at line 837 of file CodeRacer.cpp.

```
838 {
839     _usonic_stop_distance_us = stop_distance_us;
840 }
```

2.10.2.3 usonic_distance_us()

```
unsigned long CodeRacer::usonic_distance_us ( )
```

Returns the last measured distance in microseconds.

Returns

Distance in microseconds

Definition at line 845 of file CodeRacer.cpp.

```
845                                         {
846     return(_usonic_distance_us);
847 }
```

2.10.2.4 usonic_distance_cm()

```
unsigned long CodeRacer::usonic_distance_cm ( )
```

Returns the last measured distance in cm.

Returns

Distance in cm

Definition at line 852 of file CodeRacer.cpp.

```
852                                         {
853     return(_usonic_distance_cm);
854 }
```

2.11 Lower level drives methods and getters

Modules

- Methods
- Getters

2.11.1 Detailed Description

2.12 Methods

Functions

- void CodeRacer::drives_settings (uint8_t drive_left_speed, uint8_t drive_right_speed, unsigned long turn_↩ left_ms, unsigned long turn_right_ms)

  Overwrites some drive settings. This will replace the defaults set by the values in the header file.
- void CodeRacer::set_drives_stop_left_right ()

  Stopps both drives.
- void CodeRacer::set_drives_states_left_right (drivestate stateleft, drivestate stateright)

  Sets both of the drives to a specified drivestate (DRIVESTOP, DRIVEFRWD, DRIVEBACK)
- void CodeRacer::set_drive_left_state (drivestate state)

  Sets the left side drive to the specified drivestate (DRIVESTOP, DRIVEFRWD, DRIVEBACK)
- void CodeRacer::set_drive_right_state (drivestate state)

  Sets the right side drive to the specified drivestate (DRIVESTOP, DRIVEFRWD, DRIVEBACK)
- void CodeRacer::set_drive_state (drivestate state, uint8_t frwdpin, uint8_t backpin)

  Sets the specified drivestate for the drive connected to the sepecified pins (DRIVESTOP, DRIVEFRWD, DRIVEBACK)
- void CodeRacer::set_drives_speed_left_right (uint8_t speedleft, uint8_t speedright)

  Sets the speed for both of the drives.
- void CodeRacer::set_drive_left_speed (uint8_t speed)

  Sets the speed for the left side drive.
- void CodeRacer::set_drive_right_speed (uint8_t speed)

  Sets the speed for the right side drive.
- void CodeRacer::set_drive_speed (uint8_t speed, uint8_t enablepin)

  Sets the speed for the drive of the enable pin connected to the specified pin.

2.12.1 Detailed Description

2.12.2 Function Documentation

2.12.2.1 drives_settings()

```
void CodeRacer::drives_settings (
            uint8_t drive_left_speed,
            uint8_t drive_right_speed,
            unsigned long turn_left_for_ms,
            unsigned long turn_right_for_ms )
```

Overwrites some drive settings. This will replace the defaults set by the values in the header file.

Parameters

drive_left_speed	Speed of the left side drive
drive_right_speed	Speed of the right side drive
turn_left_for_ms	Time in ms the racer will turn to the left around its center if turn_left() is called
turn_right_for_ms	Time in ms the racer will turn to the right around its center if turn_right() is called

Returns

nothing

Definition at line 878 of file CodeRacer.cpp.

```
879 {
880     _drive_left_speed = drive_left_speed;
881     _drive_right_speed = drive_right_speed;
882     _turn_left_for_ms = turn_left_for_ms;
883     _turn_right_for_ms = turn_right_for_ms;
884 }
```

2.12.2.2 set_drives_stop_left_right()

```
void CodeRacer::set_drives_stop_left_right ( )
```

Stopps both drives.

Returns

nothing

Definition at line 889 of file CodeRacer.cpp.

```
889                                         {
890     set_drive_left_state(DRIVESTOP);
891     set_drive_right_state(DRIVESTOP);
892 }
```

2.12.2.3 set_drives_states_left_right()

```
void CodeRacer::set_drives_states_left_right (
             drivestate stateleft,
             drivestate stateright )
```

Sets both of the drives to a specified drivestate (DRIVESTOP, DRIVEFRWD, DRIVEBACK)

Parameters

stateleft	drivestate to set for the left side drive
stateright	drivestate to set for the right side drive

Returns

nothing

Definition at line 899 of file CodeRacer.cpp.

```
899                                                                                 {
900     set_drive_left_state(stateleft);
901     set_drive_right_state(stateright);
902 }
```

2.12.2.4 set_drive_left_state()

```
void CodeRacer::set_drive_left_state (
            drivestate state )
```

Sets the left side drive to the specified drivestate (DRIVESTOP, DRIVEFRWD, DRIVEBACK)

Parameters

state	drivestate to set for the left side drive

Returns

nothing

Definition at line 908 of file CodeRacer.cpp.

```
908                                                 {
909     set_drive_state(state, _drive_left_frwd_pin, _drive_left_back_pin);
910 }
```

2.12.2.5 set_drive_right_state()

```
void CodeRacer::set_drive_right_state (
            drivestate state )
```

Sets the right side drive to the specified drivestate (DRIVESTOP, DRIVEFRWD, DRIVEBACK)

Parameters

state	drivestate to set for the right side drive

Returns

nothing

Definition at line 916 of file CodeRacer.cpp.

```
916                                                  {
917     set_drive_state(state, _drive_right_frwd_pin, _drive_right_back_pin);
918 }
```

2.12.2.6 set_drive_state()

```
void CodeRacer::set_drive_state (
            drivestate state,
            uint8_t frwdpin,
            uint8_t backpin )
```

Sets the specified drivestate for the drive connected to the sepecified pins (DRIVESTOP, DRIVEFRWD, DRIVEB↩ ACK)

Parameters

state	drivestate to set for the connected drive
frwdpin	Pin the forward signal of the drive device driver is connected at
backpin	Pin the backward signal of the drive device driver is connected at

Returns

nothing

Definition at line 926 of file CodeRacer.cpp.

```
926                                                                                 {
927     switch (state) {
928     case DRIVESTOP:
929         digitalWrite(frwdpin, LOW);
930         digitalWrite(backpin, LOW);
931         break;
932     case DRIVEFRWD:
933         digitalWrite(frwdpin, HIGH);
934         digitalWrite(backpin, LOW);
935         break;
936     case DRIVEBACK:
937         digitalWrite(frwdpin, LOW);
938         digitalWrite(backpin, HIGH);
939         break;
940     }
941 }
```

2.12.2.7 set_drives_speed_left_right()

```
void CodeRacer::set_drives_speed_left_right (
            uint8_t speedleft,
            uint8_t speedright )
```

Sets the speed for both of the drives.

The drive will run with that speed afterwards. The direction of the drive has to be specfied with one of the set_↩ drive_state methods before

Parameters

speedleft	speed of the left side drive. 0<=speed<=255
speedright	speed of the right side drive. 0<=speed<=255

Returns

nothing

Definition at line 950 of file CodeRacer.cpp.

```
950                                                                                 {
951     set_drive_left_speed(speedleft);
952     set_drive_right_speed(speedright);
953 }
```

2.12.2.8 set_drive_left_speed()

```
void CodeRacer::set_drive_left_speed (
            uint8_t speed )
```

Sets the speed for the left side drive.

The drive will run with that speed afterwards. The direction of the drive has to be specfied with one of the set_↩ drive_state methods before

Parameters

speed	speed of the left side drive. 0<=speed<=255

Returns

nothing

Definition at line 961 of file CodeRacer.cpp.

```
961                                                 {
962     set_drive_speed(speed, _drive_left_enable_pin);
963 }
```

2.12.2.9 set_drive_right_speed()

```
void CodeRacer::set_drive_right_speed (
            uint8_t speed )
```

Sets the speed for the right side drive.

The drive will run with that speed afterwards. The direction of the drive has to be specfied with one of the set_↩ drive_state methods before

Parameters

speed	speed of the right side drive. 0<=speed<=255

Returns

nothing

Definition at line 971 of file CodeRacer.cpp.

```
971                                                 {
972     set_drive_speed(speed, _drive_right_enable_pin);
973 }
```

2.12.2.10 set_drive_speed()

```
void CodeRacer::set_drive_speed (
            uint8_t speed,
            uint8_t enablepin )
```

Sets the speed for the drive of the enable pin connected to the specified pin.

The drive will run with that speed afterwards. The direction of the drive has to be specfied with one of the set_↩ drive_state methods before

Parameters

speed	speed of the drive. 0<=speed<=255
enablepin	Pin the drives device driver enable pin is connected at

Returns

nothing

Definition at line 982 of file CodeRacer.cpp.

```
982                                                                 {
983     _analog_write(enablepin, (int)speed);
984 }
```

2.13 Getters

Functions

- uint8_t CodeRacer::drive_right_speed ()

  Get the setting for the speed of the right side drive.
- uint8_t CodeRacer::drive_left_speed ()

  Get the setting for the speed of the left side drive.

2.13.1 Detailed Description

2.13.2 Function Documentation

2.13.2.1 drive_right_speed()

```
uint8_t CodeRacer::drive_right_speed ( )
```

Get the setting for the speed of the right side drive.

Returns

Speed of the right side drive

Definition at line 994 of file CodeRacer.cpp.

```
994                                     {
995     return _drive_right_speed;
996 }
```

2.13.2.2 drive_left_speed()

```
uint8_t CodeRacer::drive_left_speed ( )
```

Get the setting for the speed of the left side drive.

Returns

Speed of the left side drive

Definition at line 1001 of file CodeRacer.cpp.

```
1001                                    {
1002     return(_drive_left_speed);
1003 }
```

2.14 Lower level LED methods

Modules

- Methods

2.14.1 Detailed Description

2.15 Methods

Functions

- void CodeRacer::set_leds_left_stop_frwd_right (ledstate leftled, ledstate stopled, ledstate frwdled, ledstate rightled)

  Sets all of the 4 LEDs to a ledstate (LEDON, LEDOFF)
- void CodeRacer::set_leds_all (ledstate alleds)

  Sets all of the 4 LEDs to the same ledstate (LEDON, LEDOFF)
- void CodeRacer::set_leds_all_off ()

  Sets all of the 4 LEDs to the ledstate LEDOFF.
- void CodeRacer::set_leds_all_on ()

  Sets all of the 4 LEDs to the ledstate LEDON.

2.15.1 Detailed Description

2.15.2 Function Documentation

2.15.2.1 set_leds_left_stop_frwd_right()

```
void CodeRacer::set_leds_left_stop_frwd_right (
             ledstate leftled,
             ledstate stopled,
             ledstate frwdled,
             ledstate rightled )
```

Sets all of the 4 LEDs to a ledstate (LEDON, LEDOFF)

Parameters

leftled	set state of status left LED (most left yellow led)
stopled	set state of status stop LED (red led)
frwdled	set state of status forward LED (green or blue led)
rightled	set state of status right LED (most right yellow led)

Returns

nothing

Definition at line 1037 of file CodeRacer.cpp.

```
1037
                {
1038     digitalWrite(_led_left_pin, leftled);
1039     digitalWrite(_led_frwd_pin, frwdled);
1040     digitalWrite(_led_right_pin, rightled);
1041     digitalWrite(_led_stop_pin, stopled);
1042 }
```

2.15.2.2 set_leds_all()

```
void CodeRacer::set_leds_all (
            ledstate alleds )
```

Sets all of the 4 LEDs to the same ledstate (LEDON, LEDOFF)

Parameters

alleds	set state to all status LEDs

Returns

nothing

Definition at line 1048 of file CodeRacer.cpp.

```
1048                                            {
1049     digitalWrite(_led_left_pin, alleds);
1050     digitalWrite(_led_frwd_pin, alleds);
1051     digitalWrite(_led_right_pin, alleds);
1052     digitalWrite(_led_stop_pin, alleds);
1053 }
```

2.15.2.3 set_leds_all_off()

```
void CodeRacer::set_leds_all_off ( )
```

Sets all of the 4 LEDs to the ledstate LEDOFF.

Returns

nothing

Definition at line 1058 of file CodeRacer.cpp.

```
1058                                {
1059     set_leds_all(LEDOFF);
1060 }
```

2.15.2.4 set_leds_all_on()

```
void CodeRacer::set_leds_all_on ( )
```

Sets all of the 4 LEDs to the ledstate LEDON.

Returns

nothing

Definition at line 1065 of file CodeRacer.cpp.

```
1065                                {
1066     set_leds_all(LEDON);
1067 }
```

Index

drive_backward
    Methods, 6
drive_forward
    Methods, 5
drive_left_speed
    Getters, 40
drive_right_speed
    Getters, 40
drives_settings
    Methods, 34

Getters, 23, 40
    drive_left_speed, 40
    drive_right_speed, 40
    servo_position, 23
    servo_position_eta_in_ms, 23
    servo_position_set_at_ms, 23
Getters and setters, 13
    is_active, 15
    is_driving, 13
    set_active, 15
    set_inactive, 14
    stopped_at_min_distance, 13
    turn_left_for_ms, 14
    turn_right_for_ms, 14

Higher level methods, setters and getters, 3

is_active
    Getters and setters, 15
is_driving
    Getters and setters, 13

kitt
    Lower level fun stuff methods, 16

look_around
    Lower level fun stuff methods, 16
Lower level drives methods and getters, 33
Lower level fun stuff methods, 16
    kitt, 16
    look_around, 16
Lower level LED methods, 41
Lower level servo drive methods and getters, 18
Lower level ultra sonic methods and getters, 25

Methods, 4, 19, 26, 34, 42
    drive_backward, 6
    drive_forward, 5
    drives_settings, 34
    servo_set_position, 22
    servo_set_position_wait, 21
    servo_set_to_center, 21
    servo_set_to_left, 21
    servo_set_to_right, 20
    servo_settings, 19
    servo_sweep, 20
    set_drive_left_speed, 38
    set_drive_left_state, 36
    set_drive_right_speed, 38
    set_drive_right_state, 36
    set_drive_speed, 39
    set_drive_state, 36
    set_drives_speed_left_right, 37
    set_drives_states_left_right, 35
    set_drives_stop_left_right, 35
    set_leds_all, 42
    set_leds_all_off, 43
    set_leds_all_on, 43
    set_leds_left_stop_frwd_right, 42
    start_stop, 11
    start_stop_at_min_distance, 10
    stop_driving, 4
    stop_stop_at_min_distance, 11
    turn_left, 7, 8
    turn_right, 8, 9
    usonic_measure_cm, 26, 27
    usonic_measure_single_shot_cm, 28, 29
    usonic_measure_single_shot_us, 29, 30
    usonic_measure_us, 26, 27

servo_position
    Getters, 23
servo_position_eta_in_ms
    Getters, 23
servo_position_set_at_ms
    Getters, 23
servo_set_position
    Methods, 22
servo_set_position_wait
    Methods, 21
servo_set_to_center
    Methods, 21
servo_set_to_left
    Methods, 21
servo_set_to_right
    Methods, 20
servo_settings
    Methods, 19
servo_sweep
    Methods, 20

set_active
Getters and setters, 15
set_drive_left_speed
Methods, 38
set_drive_left_state
Methods, 36
set_drive_right_speed
Methods, 38
set_drive_right_state
Methods, 36
set_drive_speed
Methods, 39
set_drive_state
Methods, 36
set_drives_speed_left_right
Methods, 37
set_drives_states_left_right
Methods, 35
set_drives_stop_left_right
Methods, 35
set_inactive
Getters and setters, 14
set_leds_all
Methods, 42
set_leds_all_off
Methods, 43
set_leds_all_on
Methods, 43
set_leds_left_stop_frwd_right
Methods, 42
Setters and getters, 31
usonic_distance_cm, 32
usonic_distance_us, 32
usonic_set_stop_distance_cm, 31
usonic_set_stop_distance_us, 31
start_stop
Methods, 11
start_stop_at_min_distance
Methods, 10
stop_driving
Methods, 4
stop_stop_at_min_distance
Methods, 11
stopped_at_min_distance
Getters and setters, 13

turn_left
Methods, 7, 8
turn_left_for_ms
Getters and setters, 14
turn_right
Methods, 8, 9
turn_right_for_ms
Getters and setters, 14

usonic_distance_cm
Setters and getters, 32
usonic_distance_us
Setters and getters, 32
usonic_measure_cm
Methods, 26, 27
usonic_measure_single_shot_cm
Methods, 28, 29
usonic_measure_single_shot_us
Methods, 29, 30
usonic_measure_us
Methods, 26, 27
usonic_set_stop_distance_cm
Setters and getters, 31
usonic_set_stop_distance_us
Setters and getters, 31