# adc_driver.c

```c
/*
 * adc_driver.c
 *
 * Created: 18-Oct-17 9:51:05 AM
 *  Author: ScorpionIPX
 */

#include <avr/io.h>

void ADC_init(void)
{
	DDRA = 0x00;
	// AREF = AVcc
	ADMUX = (1<<REFS0);

	// ADC Enable and prescaler of 128
	// 16000000/128 = 125000
	ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}

// read ADC value
uint16_t ADC_get_value(uint8_t ch)
{
	// select the corresponding channel 0~7
	// ANDing with '7' will always keep the value
	// of 'ch' between 0 and 7
	ch &= 0b00000111;  // AND operation with 7
	ADMUX = (ADMUX & 0xF8)|ch;     // clears the bottom 3 bits before ORing

	// start single conversion
	// write '1' to ADSC
	ADCSRA |= (1<<ADSC);

	// wait for conversion to complete
	// ADSC becomes '0' again
	// till then, run loop continuously
	while(ADCSRA & (1<<ADSC));

	return (ADC);
}
```

## adc_driver.h

```
/*
 * adc_driver.h
 *
 * Created: 18-Oct-17 9:58:32 AM
 *  Author: uidq6025
 */


#ifndef ADC_DRIVER_H_
#define ADC_DRIVER_H_

#include <avr/io.h>

#define ADC_MAX 1023
#define ADC_HALF 512

void ADC_init(void);
uint16_t ADC_get_value(uint8_t ch);

#endif /* ADC_DRIVER_H_ */
```

## l293d.c

```c
/*
 * l293d.c
 *
 * Created: 22-Apr-18 17:49:49
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <util/delay.h>
#include "l293d.h"

void init_l293d_control(void)
{
	L293D_DDR |= ((1 << L293D_HB2_DIRECTION_LEFT) | (1 <<
L293D_HB2_DIRECTION_RIGHT) | (1 << L293D_HB2_ENABLE));
	L293D_PORT &= ~((1 << L293D_HB2_DIRECTION_LEFT) | (1 <<
L293D_HB2_DIRECTION_RIGHT) | (1 << L293D_HB2_ENABLE));
}

void l293d_hb2_rotate_left(void)
{
	L293D_CLEAR_HB2_DIRECTION_RIGHT;
	_delay_ms(L293D_DEAD_TIME_MS);
	L293D_SET_HB2_DIRECTION_LEFT;
	L293D_SET_HB2_ENABLE;
}

void l293d_hb2_rotate_right(void)
{
	L293D_CLEAR_HB2_DIRECTION_LEFT;
	_delay_ms(L293D_DEAD_TIME_MS);
	L293D_SET_HB2_DIRECTION_RIGHT;
	L293D_SET_HB2_ENABLE;
}

void l293d_hb2_stop(void)
{
	L293D_CLEAR_HB2_ENABLE;
	L293D_CLEAR_HB2_DIRECTION_LEFT;
	L293D_CLEAR_HB2_DIRECTION_RIGHT;
}
```

## l293d.h

```c
/*
 * l293d.h
 *
 * Created: 22-Apr-18 17:50:02
 *  Author: ScorpionIPX
 */


#ifndef L293D_H_
#define L293D_H_

#include <avr/io.h>

#define L293D_PORT PORTD
#define L293D_DDR DDRD

#define L293D_DEAD_TIME_MS 1  /* delay time to wait before changing H bridge current
direction */

#define L293D_HB2_DIRECTION_LEFT PORTD3
#define L293D_HB2_DIRECTION_RIGHT PORTD4
#define L293D_HB2_ENABLE PORTD5

#define L293D_SET_HB2_DIRECTION_LEFT (L293D_PORT |= (1 << L293D_HB2_DIRECTION_LEFT))
#define L293D_SET_HB2_DIRECTION_RIGHT (L293D_PORT |= (1 << L293D_HB2_DIRECTION_RIGHT))
#define L293D_SET_HB2_ENABLE (L293D_PORT |= (1 << L293D_HB2_ENABLE))

#define L293D_CLEAR_HB2_DIRECTION_LEFT (L293D_PORT &= ~(1 <<
L293D_HB2_DIRECTION_LEFT))
#define L293D_CLEAR_HB2_DIRECTION_RIGHT (L293D_PORT &= ~(1 <<
L293D_HB2_DIRECTION_RIGHT))
#define L293D_CLEAR_HB2_ENABLE (L293D_PORT &= ~(1 << L293D_HB2_ENABLE))


void init_l293d_control(void);
void l293d_hb2_rotate_left(void);
void l293d_hb2_rotate_right(void);
void l293d_hb2_stop(void);

#endif /* L293D_H_ */
```

## light.c

```c
/*
 * light.c
 *
 * Created: 28-Oct-17 7:02:05 PM
 *  Author: ScorpionIPX
 */

#include <avr/io.h>
#include "light.h"
#include "adc_driver.h"

#define FILTLER_RANK 15

int get_light_intensity(uint8_t sensor)
{
	uint16_t adc_value = ADC_get_value(sensor);
	adc_value = percentage_value(adc_value);
	return adc_value;
}

int get_filtered_light_intensity(uint8_t sensor)
{
	uint16_t adc_value = 0;
	for(char i = 0; i < FILTLER_RANK; i++)
	{
		adc_value += ADC_get_value(sensor);
	}
	adc_value /= FILTLER_RANK;
	adc_value = percentage_value(adc_value);
	return adc_value;
}

int percentage_value(int raw_value)
{
	raw_value = raw_value*((long)100)/1023;
	raw_value = 100 - raw_value;
	return raw_value;
}
```

## light.h

```
/*
 * light.h
 *
 * Created: 28-Oct-17 7:02:15 PM
 *  Author: ScorpionIPX
 */


#ifndef LIGHT_H_
#define LIGHT_H_

#define LS_UP_LEFT    0
#define LS_UP_RIGHT   1
#define LS_DOWN_LEFT  2
#define LS_DOWN_RIGHT 3

#define LS_UP_LEFT_RAW_OFFSET 0
#define LS_UP_RIGHT_RAW_OFFSET   0
#define LS_DOWN_LEFT_RAW_OFFSET  -5
#define LS_DOWN_RIGHT_RAW_OFFSET 0

static const int LS_RAW_OFFSETS[4] = {LS_UP_LEFT_RAW_OFFSET, LS_UP_RIGHT_RAW_OFFSET,
LS_DOWN_LEFT_RAW_OFFSET, LS_DOWN_RIGHT_RAW_OFFSET};

int get_light_intensity(uint8_t sensor);
int get_filtered_light_intensity(uint8_t sensor);
int percentage_value(int raw_value);

#endif /* LIGHT_H_ */
```

## main.c

```
#include "global.h"
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "adc_driver.h"
#include "hx1230.h"
#include "hx_8x6_characters.h"
#include "graphics.h"
#include "light.h"
#include "pwm_driver.h"
#include "sg90_driver.h"
#include "tracking.h"
#include "user_interface.h"
#include "state_handler.h"
#include "joystick_driver.h"
#include "monitoring.h"
#include "unipolar_driver.h"
#include "l293d.h"

void uC_init(void);

int main(void)
{
	STATE = STATE_INIT;
	OLD_STATE = STATE_INIT;
	uC_init();

	STATE = STATE_IDLE;

	while (1)
	{
		if(STATE_CHANGED)
		{
			OLD_STATE = STATE; // update state
			go_to_state(STATE);
			_delay_ms(250);
			sei(); // enable interrupts
		}

		switch(OLD_STATE)
		{
			case STATE_TRACKING:
			{
				track();
				break;
			}
			case STATE_MANUAL:
			{
				manual_control();
				break;
			}
			case STATE_MONITORING:
			{
				monitor();
				break;
			}
			default:
```

```
                        {
                                break;
                        }
                }
        }
}

void uC_init(void)
{
        // Wait for system to get fully powered up
        _delay_ms(100);

        // initialize required modules
        ADC_init();
        _delay_ms(50);

        init_user_interface();
        _delay_ms(50);

        init_unipolar_control();
        _delay_ms(50);

        init_l293d_control();
        _delay_ms(50);

        init_hx1230_control();
        _delay_ms(50);
        hx_fill_screen();
        _delay_ms(500);
        hx_clear_screen();
        _delay_ms(50);

        display_title();
        display_idle_state_message();

        sei(); // enable global interrupts
}
```

## stand_control.c

```c
/*
 * stand_control.c
 *
 * Created: 29-Apr-18 21:55:45
 *  Author: ScorpionIPX
 */

#include "global.h"
#include "l293d.h"
#include "unipolar_driver.h"

#define ROTATE_LEFT_ALLOWED 1
#define ROTATE_RIGHT_ALLOWED 1
#define INCLINE_UP_ALLOWED 1
#define INCLINE_DOWN_ALLOWED 1

void stand_rotate_left(void)
{
	if(ROTATE_LEFT_ALLOWED)
	{
		l293d_hb2_rotate_left();
	}
}

void stand_rotate_right(void)
{
	if(ROTATE_RIGHT_ALLOWED)
	{
		l293d_hb2_rotate_right();
	}
}

void stand_stop_rotation(void)
{
	l293d_hb2_stop();
}

void stand_incline_down(void)
{
	if(INCLINE_DOWN_ALLOWED)
	{
		unipolar_01_step_backward(UNIPOLLAR_01_CURRENT_STEP);
	}
}

void stand_incline_up(void)
{
	if(INCLINE_UP_ALLOWED)
	{
		unipolar_01_step_forward(UNIPOLLAR_01_CURRENT_STEP);
	}
}

void stand_stop_incline(void)
{
	unipolar_01_clear_steps();
}
```

## stand_control.h

```c
/*
 * stand_control.h
 *
 * Created: 29-Apr-18 21:56:02
 *  Author: ScorpionIPX
 */


#ifndef STAND_CONTROL_H_
#define STAND_CONTROL_H_

void stand_rotate_left(void);
void stand_rotate_right(void);
void stand_stop_rotation(void);
void stand_incline_down(void);
void stand_incline_up(void);
void stand_stop_incline(void);

#endif /* STAND_CONTROL_H_ */
```

## tracking.c

```
/*
 * tracking.c
 *
 * Created: 29-Oct-17 5:24:58 PM
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <avr/io.h>
#include <stdlib.h>
#include <util/delay.h>
#include "tracking.h"
#include "light.h"
#include "hx1230.h"
#include "graphics.h"
#include "stand_control.h"

int light_up_left;
int light_up_right;
int light_down_left;
int light_down_right;

int up_intensity_average;
int down_intensity_average;
int left_intensity_average;
int right_intensity_average;

int up_down_movement_gradient_request;
int left_right_movement_gradient_request;

void track(void)
{
	light_up_left = get_filtered_light_intensity(LS_UP_LEFT);
	light_up_right = get_filtered_light_intensity(LS_UP_RIGHT);
	light_down_left = get_filtered_light_intensity(LS_DOWN_LEFT);
	light_down_right = get_filtered_light_intensity(LS_DOWN_RIGHT);

	display_light_sensor_data(LS_UP_LEFT, light_up_left);
	display_light_sensor_data(LS_UP_RIGHT, light_up_right);
	display_light_sensor_data(LS_DOWN_LEFT, light_down_left);
	display_light_sensor_data(LS_DOWN_RIGHT, light_down_right);

	up_intensity_average = light_up_left + light_up_right;
	up_intensity_average >>= 1;

	down_intensity_average = light_down_left + light_down_right;
	down_intensity_average >>= 1;

	left_intensity_average = light_up_left + light_down_left;
	left_intensity_average >>= 1;

	right_intensity_average = light_up_right + light_down_right;
	right_intensity_average >>= 1;

	hx_set_coordinates(42, 2);
	hx_write_char('0' + (up_intensity_average / 10) % 10);
	hx_write_char('0' + up_intensity_average % 10);

	hx_set_coordinates(42, 6);
	hx_write_char('0' + (down_intensity_average / 10) % 10);
	hx_write_char('0' + down_intensity_average % 10);

	hx_set_coordinates(6, 4);
	hx_write_char('0' + (left_intensity_average / 10) % 10);
	hx_write_char('0' + left_intensity_average % 10);

	hx_set_coordinates(78, 4);
	hx_write_char('0' + (right_intensity_average / 10) % 10);
	hx_write_char('0' + right_intensity_average % 10);

	up_down_movement_gradient_request = up_intensity_average - 
down_intensity_average;
	left_right_movement_gradient_request = left_intensity_average - 
right_intensity_average;

	if(abs(up_down_movement_gradient_request) > INCLINE_TRACKING_TOLERANCE)
	{
		if(up_down_movement_gradient_request > 0)
		{
			stand_incline_up();
		}
		else
		{
			stand_incline_down();
		}
	}
	else
	{
		stand_stop_incline();
	}

	if(abs(left_right_movement_gradient_request) > ROTATE_TRACKING_TOLERANCE)
	{
		if(left_right_movement_gradient_request > 0)
		{
			stand_rotate_right();
		}
		else
		{
			stand_rotate_left();
		}
	}
	else
	{
		stand_stop_rotation();
	}
	_delay_ms(40);
}
```

## unipolar_driver.c

```c
/*
 * unipolar_driver.c
 *
 * Created: 17-Apr-18 18:55:52
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <avr/io.h>
#include <util/delay.h>
#include "unipolar_driver.h"

void init_unipolar_control(void)
{
	UNIPOLAR_01_DDR |= ((1 << UNIPOLAR_01_STEP_1) | (1 << UNIPOLAR_01_STEP_2) | (1
<< UNIPOLAR_01_STEP_3) | (1 << UNIPOLAR_01_STEP_4));
	UNIPOLAR_01_CLEAR_STEP_1;
	UNIPOLAR_01_CLEAR_STEP_2;
	UNIPOLAR_01_CLEAR_STEP_3;
	UNIPOLAR_01_CLEAR_STEP_4;
	UNIPOLLAR_01_CURRENT_STEP = 4;
}

void unipolar_01_step_forward(unsigned char current_step)
{
	switch(current_step)
	{
		case 1:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_SET_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 2;
			break;
		}
		case 2:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_SET_STEP_3;
			UNIPOLAR_01_CLEAR_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 3;
			break;
		}
		case 3:
		{
			UNIPOLAR_01_CLEAR_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
			UNIPOLAR_01_SET_STEP_4;
			UNIPOLLAR_01_CURRENT_STEP = 4;
			break;
		}
		case 4:
		{
			UNIPOLAR_01_SET_STEP_1;
			UNIPOLAR_01_CLEAR_STEP_2;
			UNIPOLAR_01_CLEAR_STEP_3;
```

```
                        UNIPOLAR_01_CLEAR_STEP_4;
                        UNIPOLLAR_01_CURRENT_STEP = 1;
                        break;
                }
        }
        _delay_ms(3);
}

void unipolar_01_step_backward(unsigned char current_step)
{
        switch(current_step)
        {
                case 1:
                {
                        UNIPOLAR_01_CLEAR_STEP_1;
                        UNIPOLAR_01_CLEAR_STEP_2;
                        UNIPOLAR_01_CLEAR_STEP_3;
                        UNIPOLAR_01_SET_STEP_4;
                        UNIPOLLAR_01_CURRENT_STEP = 4;
                        break;
                }
                case 2:
                {
                        UNIPOLAR_01_SET_STEP_1;
                        UNIPOLAR_01_CLEAR_STEP_2;
                        UNIPOLAR_01_CLEAR_STEP_3;
                        UNIPOLAR_01_CLEAR_STEP_4;
                        UNIPOLLAR_01_CURRENT_STEP = 1;
                        break;
                }
                case 3:
                {
                        UNIPOLAR_01_CLEAR_STEP_1;
                        UNIPOLAR_01_SET_STEP_2;
                        UNIPOLAR_01_CLEAR_STEP_3;
                        UNIPOLAR_01_CLEAR_STEP_4;
                        UNIPOLLAR_01_CURRENT_STEP = 2;
                        break;
                }
                case 4:
                {
                        UNIPOLAR_01_CLEAR_STEP_1;
                        UNIPOLAR_01_CLEAR_STEP_2;
                        UNIPOLAR_01_SET_STEP_3;
                        UNIPOLAR_01_CLEAR_STEP_4;
                        UNIPOLLAR_01_CURRENT_STEP = 3;
                        break;
                }
        }
        _delay_ms(3);
}

void unipolar_01_clear_steps(void)
{
        UNIPOLAR_01_CLEAR_STEP_1;
        UNIPOLAR_01_CLEAR_STEP_2;
        UNIPOLAR_01_CLEAR_STEP_3;
        UNIPOLAR_01_CLEAR_STEP_4;
}
```

## unipolar_driver.h

```c
/*
 * unipolar_driver.h
 *
 * Created: 17-Apr-18 18:56:06
 *  Author: ScorpionIPX
 */


#ifndef UNIPOLAR_DRIVER_H_
#define UNIPOLAR_DRIVER_H_

void init_unipolar_control(void);
void unipolar_01_step_forward(unsigned char current_step);
void unipolar_01_step_backward(unsigned char current_step);
void unipolar_01_clear_steps(void);

#define UNIPOLAR_01_DDR DDRB
#define UNIPOLAR_01_PORT PORTB

#define UNIPOLAR_01_STEP_1 PORTB0
#define UNIPOLAR_01_STEP_2 PORTB1
#define UNIPOLAR_01_STEP_3 PORTB2
#define UNIPOLAR_01_STEP_4 PORTB3

#define UNIPOLAR_01_SET_STEP_1 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_1)
#define UNIPOLAR_01_SET_STEP_2 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_2)
#define UNIPOLAR_01_SET_STEP_3 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_3)
#define UNIPOLAR_01_SET_STEP_4 (UNIPOLAR_01_PORT |= 1 << UNIPOLAR_01_STEP_4)

#define UNIPOLAR_01_CLEAR_STEP_1 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_1))
#define UNIPOLAR_01_CLEAR_STEP_2 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_2))
#define UNIPOLAR_01_CLEAR_STEP_3 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_3))
#define UNIPOLAR_01_CLEAR_STEP_4 (UNIPOLAR_01_PORT &= ~(1 << UNIPOLAR_01_STEP_4))

#endif /* UNIPOLAR_DRIVER_H_ */
```