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_ */
```

charge_driver.c

```
/*
 * charge_driver.c
 *
 * Created: 24-Mar-18 14:39:59
 *  Author: ScorpionIPX
 */

#include "charge_driver.h"
#include "adc_driver.h"

void init_charge_control(void)
{
	RELAY_DDR |= 1 << RELAY_PIN;  // Relay pin as output
	RELAY_PORT &= ~(1 << RELAY_PIN);  // Relay default state OFF
}

void turn_on_charging(void)
{
	TURN_ON_RELAY;
}

void turn_off_charging(void)
{
	TURN_OFF_RELAY;
}

unsigned int get_battery_voltage(void)
{
	unsigned int battery_voltage = ADC_get_value(BATTERY_ADC_CHANNEL);
	battery_voltage = battery_voltage*((long)BATTERY_MAX_MV) / ADC_MAX;
	battery_voltage = battery_voltage*((long) V_BAT_GAIN) / 1000;
	return battery_voltage;
}

unsigned int get_converter_voltage(void)
{
	unsigned int battery_voltage = ADC_get_value(CONVERTER_ADC_CHANNEL);
	battery_voltage = battery_voltage*((long)CONVERTER_MAX_MV)/ADC_MAX;
	battery_voltage = battery_voltage*((long) V_CHR_GAIL) / 1000;
	return battery_voltage;
}
```

charge_driver.h

```
/*
 * charge_driver.h
 *
 * Created: 24-Mar-18 14:40:12
 *  Author: ScorpionIPX
 */

#include <avr/io.h>

#ifndef CHARGE_DRIVER_H_
#define CHARGE_DRIVER_H_

#define BATTERY_ADC_CHANNEL 6
#define CONVERTER_ADC_CHANNEL 7

#define BATTERY_MAX_MV 20000  // ADC channel uses a 1/4 voltage divider
#define CONVERTER_MAX_MV 20000  // ADC channel uses a 1/4 voltage divider

#define V_BAT_GAIN 975  // as unit of 1000
#define V_CHR_GAIL 975  // as unit of 1000

#define RELAY_PIN PINB4
#define RELAY_PORT PORTB
#define RELAY_DDR DDRB
#define TURN_ON_RELAY (RELAY_PORT |= 1 << RELAY_PIN)
#define TURN_OFF_RELAY (RELAY_PORT &= ~(1 << RELAY_PIN))


unsigned int get_battery_voltage(void);
unsigned int get_converter_voltage(void);
void init_charge_control(void);
void turn_on_charging(void);
void turn_off_charging(void);



#endif /* CHARGE_DRIVER_H_ */
```

global.h

```
/*
 * global.h
 *
 * Created: 06-Oct-17 11:42:12 AM
 *  Author: ScorpionIPX
 */


#ifndef GLOBAL_H_
#define GLOBAL_H_

#define F_CPU 8000000UL


unsigned char STATE;
unsigned char OLD_STATE;

unsigned char UNIPOLLAR_01_CURRENT_STEP;

#define STATE_INIT 0
#define STATE_IDLE 1
#define STATE_MANUAL 2
#define STATE_TRACKING 3
#define STATE_MONITORING 4


#endif /* GLOBAL_H_ */
```

graphics.c

```
/*
 * graphics.c
 *
 * Created: 28-Oct-17 6:25:15 PM
 *  Author: ScorpionIPX
 */

#include "hx1230.h"
#include "joystick_driver.h"
#include "sg90_driver.h"
#include <stdlib.h>

void display_title(void)
{
	hx_set_coordinates(0, 0);
	hx_write_string("AutoTracking LDR");
	hx_set_coordinates(0, 1);
	hx_write_string("  ScorpionIPX");
}

void display_data_menu(void)
{
	hx_set_coordinates(0, 3);
	hx_write_string("       ||");
	hx_set_coordinates(0, 4);
	hx_write_string("================");
	hx_set_coordinates(0, 5);
	hx_write_string("       ||");
}

void display_light_sensor_data(uint8_t sensor, int data)
{
	//hx_set_coordinates(60, 3 + sensor);
	hx_set_coordinates(24 + 36 * (sensor & 1), 3 + 2 * (sensor >> 1));

	// hx_write_char('0' + ((data / 100) % 10));
	hx_write_char('0' + ((data / 10) % 10));
	hx_write_char('0' + (data % 10));

	hx_set_coordinates(18, 7);
	hx_write_char('0' + OCR1A / 100);
	hx_write_char('0' + (OCR1A / 10) % 10);
	hx_write_char('0' + OCR1A % 10);

	hx_set_coordinates(66, 7);
	hx_write_char('0' + OCR1B / 100);
	hx_write_char('0' + (OCR1B / 10) % 10);
	hx_write_char('0' + OCR1B % 10);
}

void display_idle_state_message(void)
{
	hx_set_coordinates(0, 3);
	hx_write_string("- system is in ");
	hx_set_coordinates(0, 4);
	hx_write_string("IDLE mode");
	hx_set_coordinates(0, 6);
	hx_write_string("going to sleep");
}
```

```
void display_manual_state_message(void)
{
	hx_set_coordinates(0, 3);
	hx_write_string("- system is in ");
	hx_set_coordinates(0, 4);
	hx_write_string("MANUAL mode");

	hx_set_coordinates(0, 6);
	hx_write_string("JX:");
	hx_set_coordinates(54, 6);
	hx_write_string("B:");

	hx_set_coordinates(0, 7);
	hx_write_string("JY:");
	hx_set_coordinates(54, 7);
	hx_write_string("A:");
}

void display_joystick_data(unsigned int x, unsigned int y)
{
	hx_set_coordinates(24, 6);

	hx_write_char('0' + x / 1000);
	hx_write_char('0' + (x / 100) % 10);
	hx_write_char('0' + (x / 10) % 10);
	hx_write_char('0' + x % 10);

	hx_set_coordinates(72, 6);

	hx_write_char('0' + SG90_ROTATE_DUTY_CYCLE_REGISTER / 1000);
	hx_write_char('0' + (SG90_ROTATE_DUTY_CYCLE_REGISTER / 100) % 10);
	hx_write_char('0' + (SG90_ROTATE_DUTY_CYCLE_REGISTER / 10) % 10);
	hx_write_char('0' + SG90_ROTATE_DUTY_CYCLE_REGISTER % 10);

	hx_set_coordinates(24, 7);

	hx_write_char('0' + y / 1000);
	hx_write_char('0' + (y / 100) % 10);
	hx_write_char('0' + (y / 10) % 10);
	hx_write_char('0' + y % 10);

	hx_set_coordinates(72, 7);

	hx_write_char('0' + SG90_INCLINE_DUTY_CYCLE_REGISTER / 1000);
	hx_write_char('0' + (SG90_INCLINE_DUTY_CYCLE_REGISTER / 100) % 10);
	hx_write_char('0' + (SG90_INCLINE_DUTY_CYCLE_REGISTER / 10) % 10);
	hx_write_char('0' + SG90_INCLINE_DUTY_CYCLE_REGISTER % 10);


}

void display_monitoring_message(void)
{
	hx_set_coordinates(0, 3);
	hx_write_string("- system is in ");
	hx_set_coordinates(0, 4);
	hx_write_string("MONITORING mode");

	hx_set_coordinates(0, 6);
	hx_write_string("Vbat:");
```

```
        hx_set_coordinates(80, 6);
        hx_write_string("V");

        hx_set_coordinates(0, 7);
        hx_write_string("Vchr:");
        hx_set_coordinates(80, 7);
        hx_write_string("V");
}

void display_monitoring_data(unsigned int v_bat, unsigned int v_chr)
{
        hx_set_coordinates(42, 6);

        hx_write_char('0' + v_bat / 10000);
        hx_write_char('0' + (v_bat / 1000) % 10);
        hx_write_char('.');
        hx_write_char('0' + (v_bat / 100) % 10);
        hx_write_char('0' + (v_bat / 10) % 10);
        hx_write_char('0' + v_bat % 10);

        hx_set_coordinates(42, 7);

        hx_write_char('0' + v_chr / 10000);
        hx_write_char('0' + (v_chr / 1000) % 10);
        hx_write_char('.');
        hx_write_char('0' + (v_chr / 100) % 10);
        hx_write_char('0' + (v_chr / 10) % 10);
        hx_write_char('0' + v_chr % 10);
}
```

graphics.h

```
/*
 * graphics.h
 *
 * Created: 28-Oct-17 6:25:27 PM
 *  Author: ScorpionIPX
 */


#ifndef GRAPHICS_H_
#define GRAPHICS_H_

void display_title(void);
void display_data_menu(void);
void display_light_sensor_data(unsigned char sensor, int data);
void display_idle_state_message(void);
void display_manual_state_message(void);
void display_joystick_data(unsigned int x, unsigned int y);
void display_monitoring_message(void);
void display_monitoring_data(unsigned int v_bat, unsigned int v_chr);
#endif /* GRAPHICS_H_ */
```

hx_8x6characters.h

```
/*
 * hx_8x6characters.h
 *
 * Created: 06-Oct-17 2:02:58 PM
 *  Author: ScorpionIPX
 */

#ifndef HX_8X6CHARACTERS_H_
#define HX_8X6CHARACTERS_H_

static const unsigned char HX_character[][6] = {
	{0x00,0x00,0x00,0x00,0x00,0x00},	//   0x 0 0
	{0x00,0x64,0x18,0x04,0x64,0x18},	// _ 0x 1 1
	{0x00,0x3c,0x40,0x40,0x20,0x7c},	// _ 0x 2 2
	{0x00,0x0c,0x30,0x40,0x30,0x0c},	// _ 0x 3 3
	{0x00,0x3c,0x40,0x30,0x40,0x3c},	// _ 0x 4 4
	{0x00,0x00,0x3e,0x1c,0x08,0x00},	// _ 0x 5 5
	{0x00,0x04,0x1e,0x1f,0x1e,0x04},	// _ 0x 6 6
	{0x00,0x10,0x3c,0x7c,0x3c,0x10},	// _ 0x 7 7
	{0x00,0x20,0x40,0x3e,0x01,0x02},	// _ 0x 8 8
	{0x00,0x22,0x14,0x08,0x14,0x22},	//   0x 9 9
	{0x00,0x00,0x38,0x28,0x38,0x00},	//   0x a 10
	{0x00,0x00,0x10,0x38,0x10,0x00},	// _ 0x b 11
	{0x00,0x00,0x00,0x10,0x00,0x00},	// _ 0x c 12
	{0x00,0x08,0x78,0x08,0x00,0x00},	//   0x d 13
	{0x00,0x00,0x15,0x15,0x0a,0x00},	// _ 0x e 14
	{0x00,0x7f,0x7f,0x09,0x09,0x01},	// _ 0x f 15
	{0x00,0x10,0x20,0x7f,0x01,0x01},	// _ 0x10 16
	{0x00,0x04,0x04,0x00,0x01,0x1f},	// _ 0x11 17
	{0x00,0x00,0x19,0x15,0x12,0x00},	// _ 0x12 18
	{0x00,0x40,0x60,0x50,0x48,0x44},	// _ 0x13 19
	{0x00,0x06,0x09,0x09,0x06,0x00},	// _ 0x14 20
	{0x00,0x0f,0x02,0x01,0x01,0x00},	// _ 0x15 21
	{0x00,0x00,0x01,0x1f,0x01,0x00},	// _ 0x16 22
	{0x00,0x44,0x44,0x4a,0x4a,0x51},	// _ 0x17 23
	{0x00,0x14,0x74,0x1c,0x17,0x14},	// _ 0x18 24
	{0x00,0x51,0x4a,0x4a,0x44,0x44},	// _ 0x19 25
	{0x00,0x00,0x00,0x04,0x04,0x04},	//   0x1a 26
	{0x00,0x00,0x7c,0x54,0x54,0x44},	// _ 0x1b 27
	{0x00,0x08,0x08,0x2a,0x1c,0x08},	//   _ 0x1c 28
	{0x00,0x7c,0x00,0x7c,0x44,0x7c},	// _   0x1d 29
	{0x00,0x04,0x02,0x7f,0x02,0x04},	// _   0x1e 30
	{0x00,0x10,0x20,0x7f,0x20,0x10},	// _   0x1f 31
	{0x00,0x00,0x00,0x00,0x00,0x00},	//   0x20 32
	{0x00,0x00,0x00,0x6f,0x00,0x00},	// ! 0x21 33
	{0x00,0x00,0x07,0x00,0x07,0x00},	// " 0x22 34
	{0x00,0x14,0x7f,0x14,0x7f,0x14},	// # 0x23 35
	{0x00,0x00,0x07,0x04,0x1e,0x00},	// $ 0x24 36
	{0x00,0x23,0x13,0x08,0x64,0x62},	// % 0x25 37
	{0x00,0x36,0x49,0x56,0x20,0x50},	// & 0x26 38
	{0x00,0x00,0x00,0x07,0x00,0x00},	// ' 0x27 39
	{0x00,0x00,0x1c,0x22,0x41,0x00},	// ( 0x28 40
	{0x00,0x00,0x41,0x22,0x1c,0x00},	// ) 0x29 41
	{0x00,0x14,0x08,0x3e,0x08,0x14},	// * 0x2a 42
	{0x00,0x08,0x08,0x3e,0x08,0x08},	// + 0x2b 43
	{0x00,0x00,0x50,0x30,0x00,0x00},	// , 0x2c 44
	{0x00,0x08,0x08,0x08,0x08,0x08},	// - 0x2d 45
	{0x00,0x00,0x60,0x60,0x00,0x00},	// . 0x2e 46
	{0x00,0x20,0x10,0x08,0x04,0x02},	// / 0x2f 47
```

```
{0x00,0x3e,0x51,0x49,0x45,0x3e},   // 0 0x30 48
{0x00,0x00,0x42,0x7f,0x40,0x00},   // 1 0x31 49
{0x00,0x42,0x61,0x51,0x49,0x46},   // 2 0x32 50
{0x00,0x21,0x41,0x45,0x4b,0x31},   // 3 0x33 51
{0x00,0x18,0x14,0x12,0x7f,0x10},   // 4 0x34 52
{0x00,0x27,0x45,0x45,0x45,0x39},   // 5 0x35 53
{0x00,0x3c,0x4a,0x49,0x49,0x30},   // 6 0x36 54
{0x00,0x01,0x71,0x09,0x05,0x03},   // 7 0x37 55
{0x00,0x36,0x49,0x49,0x49,0x36},   // 8 0x38 56
{0x00,0x06,0x49,0x49,0x29,0x1e},   // 9 0x39 57
{0x00,0x00,0x36,0x36,0x00,0x00},   // : 0x3a 58
{0x00,0x00,0x56,0x36,0x00,0x00},   // ; 0x3b 59
{0x00,0x08,0x14,0x22,0x41,0x00},   // < 0x3c 60
{0x00,0x14,0x14,0x14,0x14,0x14},   // = 0x3d 61
{0x00,0x00,0x41,0x22,0x14,0x08},   // > 0x3e 62
{0x00,0x02,0x01,0x51,0x09,0x06},   // ? 0x3f 63
{0x00,0x3e,0x41,0x5d,0x49,0x4e},   // @ 0x40 64
{0x00,0x7e,0x09,0x09,0x09,0x7e},   // A 0x41 65
{0x00,0x7f,0x49,0x49,0x49,0x36},   // B 0x42 66
{0x00,0x3e,0x41,0x41,0x41,0x22},   // C 0x43 67
{0x00,0x7f,0x41,0x41,0x41,0x3e},   // D 0x44 68
{0x00,0x7f,0x49,0x49,0x49,0x41},   // E 0x45 69
{0x00,0x7f,0x09,0x09,0x09,0x01},   // F 0x46 70
{0x00,0x3e,0x41,0x49,0x49,0x7a},   // G 0x47 71
{0x00,0x7f,0x08,0x08,0x08,0x7f},   // H 0x48 72
{0x00,0x00,0x41,0x7f,0x41,0x00},   // I 0x49 73
{0x00,0x20,0x40,0x41,0x3f,0x01},   // J 0x4a 74
{0x00,0x7f,0x08,0x14,0x22,0x41},   // K 0x4b 75
{0x00,0x7f,0x40,0x40,0x40,0x40},   // L 0x4c 76
{0x00,0x7f,0x02,0x0c,0x02,0x7f},   // M 0x4d 77
{0x00,0x7f,0x04,0x08,0x10,0x7f},   // N 0x4e 78
{0x00,0x3e,0x41,0x41,0x41,0x3e},   // O 0x4f 79
{0x00,0x7f,0x09,0x09,0x09,0x06},   // P 0x50 80
{0x00,0x3e,0x41,0x51,0x21,0x5e},   // Q 0x51 81
{0x00,0x7f,0x09,0x19,0x29,0x46},   // R 0x52 82
{0x00,0x46,0x49,0x49,0x49,0x31},   // S 0x53 83
{0x00,0x01,0x01,0x7f,0x01,0x01},   // T 0x54 84
{0x00,0x3f,0x40,0x40,0x40,0x3f},   // U 0x55 85
{0x00,0x0f,0x30,0x40,0x30,0x0f},   // V 0x56 86
{0x00,0x3f,0x40,0x30,0x40,0x3f},   // W 0x57 87
{0x00,0x63,0x14,0x08,0x14,0x63},   // X 0x58 88
{0x00,0x07,0x08,0x70,0x08,0x07},   // Y 0x59 89
{0x00,0x61,0x51,0x49,0x45,0x43},   // Z 0x5a 90
{0x00,0x3c,0x4a,0x49,0x29,0x1e},   // [ 0x5b 91
{0x00,0x02,0x04,0x08,0x10,0x20},   // \ 0x5c 92
{0x00,0x00,0x41,0x7f,0x00,0x00},   // ] 0x5d 93
{0x00,0x04,0x02,0x01,0x02,0x04},   // ^ 0x5e 94
{0x00,0x40,0x40,0x40,0x40,0x40},   // _ 0x5f 95
{0x00,0x00,0x00,0x03,0x04,0x00},   // ` 0x60 96
{0x00,0x20,0x54,0x54,0x54,0x78},   // a 0x61 97
{0x00,0x7f,0x48,0x44,0x44,0x38},   // b 0x62 98
{0x00,0x38,0x44,0x44,0x44,0x20},   // c 0x63 99
{0x00,0x38,0x44,0x44,0x48,0x7f},   // d 0x64 100
{0x00,0x38,0x54,0x54,0x54,0x18},   // e 0x65 101
{0x00,0x08,0x7e,0x09,0x01,0x02},   // f 0x66 102
{0x00,0x0c,0x52,0x52,0x52,0x3e},   // g 0x67 103
{0x00,0x7f,0x08,0x04,0x04,0x78},   // h 0x68 104
{0x00,0x00,0x44,0x7d,0x40,0x00},   // i 0x69 105
{0x00,0x20,0x40,0x44,0x3d,0x00},   // j 0x6a 106
{0x00,0x00,0x7f,0x10,0x28,0x44},   // k 0x6b 107
{0x00,0x00,0x41,0x7f,0x40,0x00},   // l 0x6c 108
{0x00,0x7c,0x04,0x18,0x04,0x78},   // m 0x6d 109
```

```
{0x00,0x7c,0x08,0x04,0x04,0x78},   // n 0x6e 110
{0x00,0x38,0x44,0x44,0x44,0x38},   // o 0x6f 111
{0x00,0x7c,0x14,0x14,0x14,0x08},   // p 0x70 112
{0x00,0x08,0x14,0x14,0x18,0x7c},   // q 0x71 113
{0x00,0x7c,0x08,0x04,0x04,0x08},   // r 0x72 114
{0x00,0x48,0x54,0x54,0x54,0x20},   // s 0x73 115
{0x00,0x04,0x3f,0x44,0x40,0x20},   // t 0x74 116
{0x00,0x3c,0x40,0x40,0x20,0x7c},   // u 0x75 117
{0x00,0x1c,0x20,0x40,0x20,0x1c},   // v 0x76 118
{0x00,0x3c,0x40,0x30,0x40,0x3c},   // w 0x77 119
{0x00,0x44,0x28,0x10,0x28,0x44},   // x 0x78 120
{0x00,0x0c,0x50,0x50,0x50,0x3c},   // y 0x79 121
{0x00,0x44,0x64,0x54,0x4c,0x44},   // z 0x7a 122
{0x00,0x00,0x08,0x36,0x41,0x41},   // { 0x7b 123
{0x00,0x00,0x00,0x7f,0x00,0x00},   // | 0x7c 124
{0x00,0x41,0x41,0x36,0x08,0x00},   // } 0x7d 125
{0x00,0x04,0x02,0x04,0x08,0x04},   // ~ 0x7e 126
{0x00,0x7f,0x6b,0x6b,0x6b,0x7f},   //  0x7f 127
{0x00,0x00,0x7c,0x44,0x7c,0x00},   // ¬ 0x80 128
{0x00,0x00,0x08,0x7c,0x00,0x00},   // • 0x81 129
{0x00,0x00,0x64,0x54,0x48,0x00},   // ‚ 0x82 130
{0x00,0x00,0x44,0x54,0x28,0x00},   // ƒ 0x83 131
{0x00,0x00,0x1c,0x10,0x78,0x00},   // „ 0x84 132
{0x00,0x00,0x5c,0x54,0x24,0x00},   // … 0x85 133
{0x00,0x00,0x78,0x54,0x74,0x00},   // † 0x86 134
{0x00,0x00,0x64,0x14,0x0c,0x00},   // ‡ 0x87 135
{0x00,0x00,0x7c,0x54,0x7c,0x00},   // ˆ 0x88 136
{0x00,0x00,0x5c,0x54,0x3c,0x00},   // ‰ 0x89 137
{0x00,0x78,0x24,0x26,0x25,0x78},   // Š 0x8a 138
{0x00,0x78,0x25,0x26,0x24,0x78},   // ‹ 0x8b 139
{0x00,0x70,0x2a,0x29,0x2a,0x70},   // Œ 0x8c 140
{0x00,0x78,0x25,0x24,0x25,0x78},   // • 0x8d 141
{0x00,0x20,0x54,0x56,0x55,0x78},   // } 0x8e 142
{0x00,0x20,0x55,0x56,0x54,0x78},   // • 0x8f 143
{0x00,0x20,0x56,0x55,0x56,0x78},   // • 0x90 144
{0x00,0x20,0x55,0x54,0x55,0x78},   // ‘ 0x91 145
{0x00,0x7c,0x54,0x56,0x55,0x44},   // ’ 0x92 146
{0x00,0x7c,0x55,0x56,0x54,0x44},   // “ 0x93 147
{0x00,0x7c,0x56,0x55,0x56,0x44},   // ” 0x94 148
{0x00,0x7c,0x55,0x54,0x55,0x44},   // • 0x95 149
{0x00,0x38,0x54,0x56,0x55,0x18},   // – 0x96 150
{0x00,0x38,0x55,0x56,0x54,0x18},   // — 0x97 151
{0x00,0x38,0x56,0x55,0x56,0x18},   // ˜ 0x98 152
{0x00,0x38,0x55,0x54,0x55,0x18},   // ™ 0x99 153
{0x00,0x00,0x44,0x7e,0x45,0x00},   // š 0x9a 154
{0x00,0x00,0x45,0x7e,0x44,0x00},   // › 0x9b 155
{0x00,0x00,0x46,0x7d,0x46,0x00},   // œ 0x9c 156
{0x00,0x00,0x45,0x7c,0x45,0x00},   // • 0x9d 157
{0x00,0x00,0x48,0x7a,0x41,0x00},   // ~ 0x9e 158
{0x00,0x00,0x49,0x7a,0x40,0x00},   // Ÿ 0x9f 159
{0x00,0x00,0x4a,0x79,0x42,0x00},   //   0xa0 160
{0x00,0x00,0x49,0x78,0x41,0x00},   // ¡ 0xa1 161
{0x00,0x38,0x44,0x46,0x45,0x38},   // ¢ 0xa2 162
{0x00,0x38,0x45,0x46,0x44,0x38},   // £ 0xa3 163
{0x00,0x38,0x46,0x45,0x46,0x38},   // ¤ 0xa4 164
{0x00,0x38,0x45,0x44,0x45,0x38},   // ¥ 0xa5 165
{0x00,0x30,0x48,0x4a,0x49,0x30},   // ¦ 0xa6 166
{0x00,0x30,0x49,0x4a,0x48,0x30},   // § 0xa7 167
{0x00,0x30,0x4a,0x49,0x4a,0x30},   // ¨ 0xa8 168
{0x00,0x30,0x49,0x48,0x49,0x30},   // © 0xa9 169
{0x00,0x3c,0x40,0x42,0x41,0x3c},   // ª 0xaa 170
{0x00,0x3c,0x41,0x42,0x40,0x3c},   // « 0xab 171
```

```
{0x00,0x3c,0x42,0x41,0x42,0x3c},   // ¬ 0xac 172
{0x00,0x3c,0x41,0x40,0x41,0x3c},   // - 0xad 173
{0x00,0x3c,0x40,0x42,0x21,0x7c},   // ® 0xae 174
{0x00,0x3c,0x41,0x42,0x20,0x7c},   // ¯ 0xaf 175
{0x00,0x38,0x42,0x41,0x22,0x78},   // ° 0xb0 176
{0x00,0x3c,0x41,0x40,0x21,0x7c},   // ± 0xb1 177
{0x00,0x4e,0x51,0x71,0x11,0x0a},   // ² 0xb2 178
{0x00,0x58,0x64,0x64,0x24,0x10},   // ³ 0xb3 179
{0x00,0x7c,0x0a,0x11,0x22,0x7d},   // ´ 0xb4 180
{0x00,0x78,0x12,0x09,0x0a,0x71},   // µ 0xb5 181
{0x00,0x00,0x00,0x04,0x02,0x01},   // ¶ 0xb6 182
{0x00,0x01,0x02,0x04,0x00,0x00},   // · 0xb7 183
{0x00,0x00,0x02,0x00,0x02,0x00},   // ¸ 0xb8 184
{0x00,0x30,0x48,0x45,0x40,0x20},   // ¹ 0xb9 185
{0x00,0x00,0x00,0x7b,0x00,0x00},   // º 0xba 186
{0x00,0x38,0x44,0x44,0x38,0x44},   // » 0xbb 187
{0x00,0x40,0x3e,0x49,0x49,0x36},   // ¼ 0xbc 188
{0x00,0x08,0x04,0x08,0x70,0x0c},   // ½ 0xbd 189
{0x00,0x60,0x50,0x48,0x50,0x60},   // ¾ 0xbe 190
{0x00,0x20,0x52,0x55,0x59,0x30},   // ¿ 0xbf 191
{0x00,0x38,0x54,0x54,0x54,0x00},   // À 0xc0 192
{0x00,0x00,0x00,0x7f,0x41,0x00},   // Á 0xc1 193
{0x00,0x40,0x22,0x14,0x18,0x60},   // Â 0xc2 194
{0x00,0x7c,0x20,0x20,0x1c,0x20},   // Ã 0xc3 195
{0x00,0x44,0x3c,0x04,0x7c,0x44},   // Ä 0xc4 196
{0x00,0x40,0x3c,0x12,0x12,0x0c},   // Å 0xc5 197
{0x00,0x41,0x63,0x55,0x49,0x41},   // Æ 0xc6 198
{0x00,0x38,0x44,0x44,0x3c,0x04},   // Ç 0xc7 199
{0x00,0x08,0x04,0x3c,0x44,0x24},   // È 0xc8 200
{0x00,0x08,0x14,0x7f,0x14,0x08},   // É 0xc9 201
{0x00,0x4e,0x71,0x01,0x71,0x4e},   // Ê 0xca 202
{0x00,0x45,0x29,0x11,0x29,0x45},   // Ë 0xcb 203
{0x00,0x0d,0x51,0x51,0x51,0x3d},   // Ì 0xcc 204
{0x00,0x00,0x00,0x05,0x02,0x05},   // Í 0xcd 205
{0x00,0x40,0x00,0x40,0x00,0x40},   // Î 0xce 206
{0x00,0x00,0x08,0x1c,0x3e,0x00},   // Ï 0xcf 207
{0x00,0x1c,0x1c,0x1c,0x00,0x00},   // Ð 0xd0 208
{0x00,0x00,0x70,0x08,0x07,0x00},   // Ñ 0xd1 209
{0x00,0x00,0x08,0x08,0x08,0x00},   // Ò 0xd2 210
{0x00,0x00,0x1d,0x15,0x17,0x00},   // Ó 0xd3 211
{0x00,0x00,0x07,0x05,0x07,0x00},   // Ô 0xd4 212
{0x00,0x00,0x11,0x15,0x0a,0x00},   // Õ 0xd5 213
{0x00,0x00,0x00,0x00,0x00,0x00},   // Ö 0xd6 214
{0x00,0x04,0x3c,0x41,0x20,0x00},   // × 0xd7 215
{0x00,0x7c,0x16,0x15,0x16,0x08},   // Ø 0xd8 216
{0x00,0x21,0x16,0x08,0x34,0x42},   // Ù 0xd9 217
{0x00,0x7f,0x09,0x1d,0x01,0x03},   // Ú 0xda 218
{0x00,0x38,0x54,0x54,0x14,0x08},   // Û 0xdb 219
{0x00,0x00,0x00,0x7c,0x40,0x40},   // Ü 0xdc 220
{0x00,0x7f,0x0e,0x1c,0x38,0x7f},   // Ý 0xdd 221
{0x00,0x41,0x22,0x5d,0x22,0x1c},   // Þ 0xde 222
{0x00,0x1c,0x3e,0x1c,0x08,0x00},   // ß 0xdf 223
{0x00,0x7f,0x7f,0x7f,0x7f,0x7f},   // à 0xe0 224
{0x00,0x77,0x7b,0x01,0x7b,0x77},   // á 0xe1 225
{0x00,0x7f,0x43,0x75,0x43,0x7f},   // â 0xe2 226
{0x00,0x7f,0x6f,0x55,0x43,0x7f},   // ã 0xe3 227
{0x00,0x40,0x40,0x40,0x40,0x40},   // ä 0xe4 228
{0x00,0x44,0x42,0x5f,0x42,0x44},   // å 0xe5 229
{0x00,0x40,0x5e,0x45,0x5e,0x40},   // æ 0xe6 230
{0x00,0x40,0x48,0x55,0x5e,0x40},   // ç 0xe7 231
{0x00,0x00,0x04,0x08,0x10,0x20},   // è 0xe8 232
{0x00,0x03,0x07,0x0e,0x1c,0x38},   // é 0xe9 233
```

```
        {0x00,0x01,0x03,0x07,0x0f,0x1f},   // ê 0xea 234
        {0x00,0x7c,0x78,0x70,0x60,0x40},   // ë 0xeb 235
        {0x00,0x08,0x08,0x1c,0x22,0x1c},   // ì 0xec 236
        {0x00,0x00,0x1c,0x22,0x1c,0x00},   // í 0xed 237
        {0x00,0x02,0x00,0x08,0x00,0x20},   // î 0xee 238
        {0x00,0x04,0x3e,0x3f,0x3e,0x04},   // ï 0xef 239
        {0x00,0x10,0x3e,0x7e,0x3e,0x10},   // ð 0xf0 240
        {0x00,0x55,0x2a,0x55,0x2a,0x55},   // ñ 0xf1 241
        {0x00,0x24,0x2a,0x7f,0x2a,0x12},   // ò 0xf2 242
        {0x00,0x04,0x1e,0x1f,0x1e,0x04},   // ó 0xf3 243
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ô 0xf4 244
        {0x00,0x00,0x00,0x00,0x00,0x00},   // õ 0xf5 245
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ö 0xf6 246
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ÷ 0xf7 247
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ø 0xf8 248
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ù 0xf9 249
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ú 0xfa 250
        {0x00,0x00,0x00,0x00,0x00,0x00},   // û 0xfb 251
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ü 0xfc 252
        {0x00,0x00,0x00,0x00,0x00,0x00},   // ý 0xfd 253
        {0x00,0x00,0x00,0x00,0x00,0x00},   // þ 0xfe 254
        {0x00,0x00,0x00,0x00,0x00,0x00}    // ÿ 0xff 255
};

#endif /* HX_8X6CHARACTERS_H_ */
```

hx1230.c

```
/*
 * hx1230.c
 *
 * Created: 06-Oct-17 12:35:49 AM
 *  Author: ScorpionIPX
 */

#include "global.h"
#include "hx1230.h"
#include <util/delay.h>
#include "hx_8x6_characters.h"
#include <string.h>


void init_hx1230_control(void)
{
	// set required pins as output
	HX1230_DDR |= ((1 << HX_RST) | (1 << HX_CE) | (1 << HX_DIN) | (1 << HX_CLK));

	// set idle state
	CLEAR_HX_CLK;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif
	CLEAR_HX_RST;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif
	_delay_ms(50);
	SET_HX_RST;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif
	CLEAR_HX_CE;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif
	_delay_ms(1);
	SET_HX_CE;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif
	_delay_ms(1);

	// commands needed to initialize hx1230 display
	// found within a chinese data sheet

	hx_send_command(0x2f);
	hx_send_command(0x90);
	hx_send_command(0xa6);
	hx_send_command(0xa4);
	hx_send_command(0xaf);

	hx_send_command(0x40);
	hx_send_command(0xb0);
	hx_send_command(0x10);
	hx_send_command(0x00);
}
```

```
void hx_send_data(unsigned char _data)
{
	// activate hx1230
	CLEAR_HX_CE;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif

	// configure communication for data transfer
	SET_HX_DIN;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif

	// toggle clock
	SET_HX_CLK;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif
	CLEAR_HX_CLK;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif

	// send the actual data, MSB fiHX_RST
	for(int bit_position = 7; bit_position >= 0; bit_position--)
	{
		// calculate bit to be send
		if(((_data >> bit_position) & 1) == 1)
		{
			SET_HX_DIN;
			#ifdef HX_DELAY_ENABLED
			_delay_us(HX_DELAY_US);
			#endif
		}
		else
		{
			CLEAR_HX_DIN;
			#ifdef HX_DELAY_ENABLED
			_delay_us(HX_DELAY_US);
			#endif
		}

		// toggle clock
		SET_HX_CLK;
		#ifdef HX_DELAY_ENABLED
		_delay_us(HX_DELAY_US);
		#endif
		CLEAR_HX_CLK;
		#ifdef HX_DELAY_ENABLED
		_delay_us(HX_DELAY_US);
		#endif
	}

	// deactivate hx1230
	SET_HX_CE;
}

void hx_send_command(unsigned char _command)
{
	// activate hx1230
	CLEAR_HX_CE;
```

```
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif

	// configure communication for command transfer
	CLEAR_HX_DIN;

	// toggle clock
	SET_HX_CLK;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif
	CLEAR_HX_CLK;
	#ifdef HX_DELAY_ENABLED
	_delay_us(HX_DELAY_US);
	#endif

	// send the actual command, MSB fiHX_RST
	for(int bit_position = 7; bit_position >= 0; bit_position--)
	{
		// calculate bit to be send
		if(((_command >> bit_position) & 1) == 1)
		{
			SET_HX_DIN;
			#ifdef HX_DELAY_ENABLED
			_delay_us(HX_DELAY_US);
			#endif
		}
		else
		{
			CLEAR_HX_DIN;
			#ifdef HX_DELAY_ENABLED
			_delay_us(HX_DELAY_US);
			#endif
		}

		// toggle clock
		SET_HX_CLK;
		#ifdef HX_DELAY_ENABLED
		_delay_us(HX_DELAY_US);
		#endif
		CLEAR_HX_CLK;
		#ifdef HX_DELAY_ENABLED
		_delay_us(HX_DELAY_US);
		#endif
	}

	// deactivate hx1230
	SET_HX_CE;
}

void hx_set_coordinates(unsigned char _x, unsigned char _y)
{
	// 0, 0 is the upper left corner

	hx_send_command(0xB0 + _y);
	hx_send_command(0x10 | ((_x & 0x7F) >> 4));
	hx_send_command(0x0F & _x);
}

void hx_clear_screen(void)
{
```

```
	unsigned char col, row;

	hx_set_coordinates(0, 0);

	for(row = 0; row <= HX_MAX_ROW_ROOT; row ++)
	{
		for(col = 0; col <= HX_MAX_COL; col ++)
		{
			hx_send_data(0x00);
		}
	}
}

void hx_fill_screen(void)
{
	unsigned char col, row;

	hx_set_coordinates(0, 0);

	for(row = 0; row < 9; row ++)
	{
		for(col = 0; col < 96; col ++)
		{
			hx_send_data(0xFF);
		}
	}
}

void hx_write_char(const unsigned char _character)
{

	for(int row_index = 0; row_index < 6; row_index ++)
	{
		hx_send_data(HX_character[_character][row_index]);
	}
}

void hx_write_string(const char *_characters_array)
{
	int string_length = strlen(_characters_array);
	for(int char_index = 0; char_index < string_length; char_index++)
	{
		hx_write_char((const unsigned char)(_characters_array[char_index]));
	}
}
```

hx1230.h

```
/*
 * hx1230.h
 *
 * Created: 06-Oct-17 12:36:42 AM
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <avr/io.h>
#include "hx_8x6_characters.h"

#ifndef HX1230_H_
#define HX1230_H_

//#define HX_DELAY_ENABLED // if uC is too fast, HX1230 won't be able to read commands
#ifdef HX_DELAY_ENABLED
	#define HX_DELAY_US 1
#endif


#define HX1230_PORT PORTC //port used to control hx1230
#define HX1230_DDR  DDRC //data direction register used for hx1230

#define HX_RST PC0 //external reset input
#define HX_CE  PC1 //chip enable
#define HX_DIN PC6 //serial data input
#define HX_CLK PC7 //serial clock input

#define SET_HX_RST (HX1230_PORT |= (1 << HX_RST))
#define SET_HX_CE  (HX1230_PORT |= (1 << HX_CE))
#define SET_HX_DIN (HX1230_PORT |= (1 << HX_DIN))
#define SET_HX_CLK (HX1230_PORT |= (1 << HX_CLK))

#define CLEAR_HX_RST (HX1230_PORT &= ~(1 << HX_RST))
#define CLEAR_HX_CE  (HX1230_PORT &= ~(1 << HX_CE))
#define CLEAR_HX_DIN (HX1230_PORT &= ~(1 << HX_DIN))
#define CLEAR_HX_CLK (HX1230_PORT &= ~(1 << HX_CLK))

#define HX_MAX_ROW 64
#define HX_MAX_ROW_ROOT 8
#define HX_MAX_COL 96

void init_hx1230_control(void);
void hx_send_data(unsigned char _data);
void hx_send_command(unsigned char _command);
void hx_set_coordinates(unsigned char _x, unsigned char _y);
void hx_clear_screen(void);
void hx_fill_screen(void);
void hx_write_char(const unsigned char _character);
void hx_write_string(const char *_characters_array);

#endif /* HX1230_H_ */
```

joystick_driver.c

```
/*
 * joystick_driver.c
 *
 * Created: 3/19/2018 11:34:43 PM
 *  Author: uidq6025
 */


#include "global.h"
#include <util/delay.h>
#include "user_interface.h"
#include "adc_driver.h"
#include "graphics.h"
#include "joystick_driver.h"
#include "sg90_driver.h"
#include "unipolar_driver.h"
#include "l293d.h"

void manual_control(void)
{
	unsigned int x = ADC_get_value(ADC_CHANNEL_X_AXIS);
	unsigned int y = ADC_get_value(ADC_CHANNEL_Y_AXIS);

	display_joystick_data(x, y);

	if(y > (JOYSTICK_IDLE_VALUE + JOYSTICK_DEAD_ZONE))
	{
		unipolar_01_step_backward(UNIPOLLAR_01_CURRENT_STEP);
	}
	else if(y < (JOYSTICK_IDLE_VALUE - JOYSTICK_DEAD_ZONE))
	{
		unipolar_01_step_forward(UNIPOLLAR_01_CURRENT_STEP);
	}
	else
	{
		unipolar_01_clear_steps();
	}

	if(x > (JOYSTICK_IDLE_VALUE + JOYSTICK_DEAD_ZONE))
	{
		l293d_hb2_rotate_right();
	}
	else if(x < (JOYSTICK_IDLE_VALUE - JOYSTICK_DEAD_ZONE))
	{
		l293d_hb2_rotate_left();
	}
	else
	{
		l293d_hb2_stop();
	}

	_delay_ms(25);
}

signed int format_axis(unsigned int axis)
{
	signed int formated_axis;
	if(axis >= ADC_HALF)
	{
```

```
			formated_axis = (axis - ADC_HALF) / JOYSTICK_RESOLUTION;
		}
		else
		{
			formated_axis = axis / JOYSTICK_RESOLUTION;
			formated_axis = formated_axis * (-1);
			{
			}
		}

		return formated_axis;
}
```

joystick_driver.h

```
/*
 * joystick_driver.h
 *
 * Created: 3/19/2018 11:34:57 PM
 *  Author: uidq6025
 */


#ifndef JOYSTICK_DRIVER_H_
#define JOYSTICK_DRIVER_H_

#define JOYSTICK_RESOLUTION 10
#define JOYSTICK_IDLE_VALUE 512
#define JOYSTICK_DEAD_ZONE 100

void manual_control(void);
signed int format_axis(unsigned int axis);



#endif /* JOYSTICK_DRIVER_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
}
```

monitoring.c

```
/*
 * monitoring.c
 *
 * Created: 24-Mar-18 15:06:41
 *  Author: ScorpionIPX
 */

#include "global.h"
#include <util/delay.h>
#include "charge_driver.h"
#include "graphics.h"

void monitor(void)
{
	unsigned int battery_voltage = get_battery_voltage();
	unsigned int converter_voltage = get_converter_voltage();

	display_monitoring_data(battery_voltage, converter_voltage);
	_delay_ms(100);
}
```

monitoring.h

```
/*
 * monitoring.h
 *
 * Created: 24-Mar-18 15:06:54
 *  Author: ScorpionIPX
 */


#ifndef MONITORING_H_
#define MONITORING_H_

void monitor(void);


#endif /* MONITORING_H_ */
```

state_handler.c

```
/*
 * state_handler.c
 *
 * Created: 3/19/2018 8:08:05 PM
 *  Author: uidq6025
 */

#include "global.h"
#include <util/delay.h>
#include "graphics.h"
#include "state_handler.h"
#include "hx1230.h"
#include "unipolar_driver.h"
#include "l293d.h"

void go_to_state(unsigned char state)
{
	l293d_hb2_stop();  /* make sure motor control is turned off when changing
states */
	unipolar_01_clear_steps();  /* make sure motor control is turned off when
changing states */

	hx_clear_screen();
	switch(state)
	{
		case STATE_IDLE:
		{
			STATE = STATE_IDLE; // update global state
			idle_state_setup();
			break;
		}
		case STATE_MANUAL:
		{
			STATE = STATE_MANUAL; // update global state
			manual_state_setup();
			break;
		}
		case STATE_TRACKING:
		{
			STATE = STATE_TRACKING; // update global state
			tracking_state_setup();
			break;
		}
		case STATE_MONITORING:
		{
			STATE = STATE_MONITORING; // update global state
			monitoring_state_setup();
			break;
		}
	}
}

void idle_state_setup(void)
{
	_delay_ms(200);
	hx_clear_screen();
	display_title();
	display_idle_state_message();
}
```

```
void manual_state_setup(void)
{
	_delay_ms(200);
	hx_clear_screen();
	display_title();
	display_manual_state_message();
}

void tracking_state_setup(void)
{
	_delay_ms(200);
	hx_clear_screen();
	display_title();
	display_data_menu();
	_delay_ms(500);
}

void monitoring_state_setup(void)
{
	_delay_ms(200);
	hx_clear_screen();
	display_title();
	display_monitoring_message();
}
```

state_handler.h

```
/*
 * state_handler.h
 *
 * Created: 3/19/2018 8:10:15 PM
 *  Author: uidq6025
 */

#include "global.h"

#ifndef STATE_HANDLER_H_
#define STATE_HANDLER_H_


#define STATE_CHANGED (!(OLD_STATE == STATE))


void go_to_state(unsigned char state);
void idle_state_setup(void);
void manual_state_setup(void);
void tracking_state_setup(void);
void monitoring_state_setup(void);


#endif /* STATE_HANDLER_H_ */
```

tracking.h

```
/*
 * tracking.h
 *
 * Created: 29-Oct-17 5:25:08 PM
 *  Author: ScorpionIPX
 */


#ifndef TRACKING_H_
#define TRACKING_H_

#define INCLINE_TRACKING_TOLERANCE 2
#define ROTATE_TRACKING_TOLERANCE 2

void track(void);

#endif /* TRACKING_H_ */
```

user_interface.c

```
/*
 * user_interface.c
 *
 * Created: 3/19/2018 8:38:39 PM
 *  Author: uidq6025
 */




#include "global.h"
#include <util/delay.h>
#include <avr/interrupt.h>
#include "user_interface.h"
#include "state_handler.h"
#include "adc_driver.h"



void init_user_interface(void)
{
	init_next_state_button();
}


void init_next_state_button(void)
{
	BUTTON_1_DRR &= ~(1 << BUTTON_1_PIN); // PD2 is input

	BUTTON_1_PORT |= (1 << BUTTON_1_PIN);	// turn on the pull-up resistor
	// PD2 is now an input with pull-up enabled

	MCUCR &= ~(1 << ISC00 | 1 << ISC01);	// low level of INT0 generates an
interrupt request: when BUTTON_1 is pressed
	GICR |= (1 << INT0);	// turns on INT0
}

ISR (INT0_vect)
{
	cli(); // temporarily disable interrupts
	switch(STATE)
	{
		case STATE_IDLE:
		{
			STATE = STATE_MANUAL;
			break;
		}
		case STATE_MANUAL:
		{
			STATE = STATE_TRACKING;
			break;
		}
		case STATE_TRACKING:
		{
			STATE = STATE_MONITORING;
			break;
		}
		case STATE_MONITORING:
```

```
                {
                        STATE = STATE_IDLE;
                        break;
                }
                default:
                {
                        STATE = STATE_IDLE;
                        break;
                }
        }
}
```

user_interface.h

```
/*
 * user_interface.h
 *
 * Created: 3/19/2018 8:38:53 PM
 *  Author: uidq6025
 */

#include <avr/io.h>


#ifndef USER_INTERFACE_H_
#define USER_INTERFACE_H_

#define BUTTON_1_DRR DDRD
#define BUTTON_1_PORT PORTD
#define BUTTON_1_PIN 2 // PD2

#define ADC_CHANNEL_X_AXIS 4
#define ADC_CHANNEL_Y_AXIS 5

void init_next_state_button(void);
void init_user_interface(void);


#endif /* USER_INTERFACE_H_ */
```