**adc\_driver.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**

/\*

\* 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**

/\*

\* 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

/\*

\* 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**

/\*

\* 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**

/\*

\* 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**

/\*

\* 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**

/\*

\* 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\_ \*/