```
/*
 * lux.h
 *
 *  Created on: Apr 9, 2019
 *      Author: Steve Antony
 */

#ifndef LUX_H_
#define LUX_H_


/*********************************************
 *        Includes
 *********************************************/
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>
#include <math.h>

#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"
#include "utils/uartstdio.h"
#include "uart.h"
#include "driverlib/gpio.h"
#include "driverlib/inc/hw_memmap.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "log.h"
#include "i2c.h"

/*********************************************
 *        MACRO
 *********************************************/
#define CONTROL_REGISTER (0X00)
#define TIMING_REGISTER (0X01)
#define THRESHLOWLOW (0x02)
#define THRESHLOWHIGH (0x03)
#define THRESHHIGHLOW (0x04)
#define THRESHHIGHHIGH (0x05)
#define INTERRUPT (0x06)
#define INDICATION_REGISTER (0x0A)
#define DATA0LOW_REGISTER (0X0C)
#define DATA0HIGH_REGISTER (0X0D)
#define DATA1LOW_REGISTER (0X0E)
#define DATA1HIGH_REGISTER (0X0F)

#define WRITE_COMMAND (0x80)


#define QUEUE_TIMEOUT_TICKS (10)
```

```
#define NOTIFY_TAKE_TIMEOUT (500)
#define TEMP_TIME_PERIOD_MS (1000)
#define TEMP_SENSOR_ADDR (0x48)
#define TEMP_REG_OFFSET_ADDR (0x00)
#define LIGHT_SENSOR (0x39)

#define BUFFER (50)

/*********************************************
 *        GLOBALS
 *********************************************/
struct log_struct_temp
{
    char time_stamp[30];
    char temp[40];

};

struct log_struct_led
{
    char time_stamp[30];
    long count;
    char name[10];

};


extern TaskHandle_t handle;

extern uint32_t output_clock_rate_hz;

extern QueueHandle_t myQueue_light;


/*********************************************
 *        Function Prototypes
 *********************************************/
/*******************************************
 * Func name :   TempTask
 * Parameters:   none
 * Description : Thread for temperature task
 *******************************************/
void LightTask(void *pvParameters);
/*******************************************
 * Func name :   vTimerCallback_Temp_handler
 * Parameters:   none
 * Description : handler for temperature timer
 *******************************************/
void vTimerCallback_Light_handler( TimerHandle_t  *);

/*******************************************
 * Func name :   i2c_setup
 * Parameters:   none
 * Description : Configuration of i2c bus
```

```
 ******************************************/
void i2c_setup();

/******************************************
 * Func name :   read_lux_CH0
 * Parameters:   none
 * Description : Reads CH0 value of the lux sensor
 ******************************************/
void read_lux_CH0();

/******************************************
 * Func name :   read_lux_CH1
 * Parameters:   none
 * Description : Reads CH1 value of the lux sensor
 ******************************************/
void read_lux_CH1();

/******************************************
 * Func name :   lux_sensor_setup
 * Parameters:   none
 * Description : Wrapper for configuring lux sensor registers
 ******************************************/
int8_t lux_sensor_setup();

/******************************************
 * Func name :   read_byte_i2c2
 * Parameters:   slave address,  register address, address of data
 * Description : Read a byte to any register
 ******************************************/
void read_byte_i2c2(uint8_t slave, uint8_t register_addr, uint8_t *data);

/******************************************
 * Func name :   write_byte_i2c2
 * Parameters:   slave address,  register address, data
 * Description : Write a byte to any register
 ******************************************/
void write_byte_i2c2(uint8_t slave, uint8_t register_addr, uint8_t data);

/******************************************
 * Func name :   lux_measurement
 * Parameters:   CH0 value, CH1 value
 * Description : Calculate lux value based on the channel values
 ******************************************/
float lux_measurement(float , float );

#endif /* LUX_H_ */
/*
 * main.h
 *
 *  Created on: Mar 28, 2015
 *      Author: steve
 */

#ifndef OBJECT_DETECTION_H_
```

```
#define OBJECT_DETECTION_H_

/*********************************************
 *        Includes
 *********************************************/
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>

#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"
#include "timer.h"

#include "driverlib/gpio.h"
#include "driverlib/inc/hw_memmap.h"
#include "driverlib/inc/hw_timer.h"
#include "driverlib/inc/hw_types.h"

#include "driverlib/sysctl.h"
#include "driverlib/rom.h"
#include "utils/uartstdio.h"
#include "driverlib/inc/hw_ints.h"
#include "driverlib/fpu.h"
#include "log.h"
#include "motor_driver.h"
#include "semphr.h"


/*********************************************
 *        Macros
 *********************************************/
#define DETECT_TIME_PERIOD_MS (1000)
#define PERIOD_ULTRASONIC     (530)


/*********************************************
 *        Function prototypes
 *********************************************/
/*********************************************
 * Func name :   init_ultrasonic_sensor
 * Parameters:   none
 * Description : Initiates the trigger and echo pins of ultrasonic
sensors
 *********************************************/
void init_ultrasonic_sensor();

/*********************************************
 * Func name :   PortFIntHandler
 * Parameters:   none
 * Description : Interrupt handler
```

```
 ******************************************/
void PortFIntHandler();

/*****************************************
 * Func name :   findobject
 * Parameters:   none
 * Description : makes trigger pin high for 10ms
 *****************************************/
void find_object();

/*****************************************
 * Func name :   vTimerCallback_Temp_handler
 * Parameters:   none
 * Description : handler for temperature timer
 *****************************************/
void vTimerCallback_Ultra_handler( TimerHandle_t  *);

/*Ultrasonic task*/
void UtrasonicTask(void *);

/*****************************************
 *        Global declaration
 *****************************************/
extern uint32_t output_clock_rate_hz;
extern QueueHandle_t myQueue_ultra, myQueue_light, myQueue_log;
extern TaskHandle_t handle_motor;



#endif
/*
 * log.h
 *
 *  Created on: Apr 8, 2019
 *      Author: Steve Antony
 */

#ifndef LOG_H_
#define LOG_H_

/*****************************************
 *        Includes
 *****************************************/
#include <lux.h>
#include "FreeRTOS.h"
#include "queue.h"
#include "portmacro.h"
#include "utils/uartstdio.h"
#include "portmacro.h"
#include "time.h"
#include "semphr.h"
/*****************************************
 *        MACRO
 *****************************************/
```

```
#define QueueLength (110)
#define TIMEOUT_TICKS (10)
#define BUFFER (100)


#define LOG_INFO(str) {\
xSemaphoreTake(xSemaphore, 0);\
memset(temp_buffer,'\0',100);\
sprintf(temp_buffer,"INFO RN: [%d] %s",xTaskGetTickCount(),str);\
xQueueSendToBack( myQueue_log,( void * ) temp_buffer, QUEUE_TIMEOUT_TICKS
) ;\
xSemaphoreGive(xSemaphore);\
}

#define LOG_ERROR(str) {\
xSemaphoreTake(xSemaphore, 0);\
memset(temp_buffer,'\0',100);\
sprintf(temp_buffer,"ERROR RN: [%d] %s",xTaskGetTickCount(),str);\
xQueueSendToBack( myQueue_log,( void * ) temp_buffer, QUEUE_TIMEOUT_TICKS
) ;\
xSemaphoreGive(xSemaphore);\
}

#define LOG_WARN(str) {\
xSemaphoreTake(xSemaphore, 0);\
memset(temp_buffer,'\0',100);\
sprintf(temp_buffer,"WARN RN: [%d] %s",xTaskGetTickCount(),str);\
xQueueSendToBack( myQueue_log,( void * ) temp_buffer, QUEUE_TIMEOUT_TICKS
) ;\
xSemaphoreGive(xSemaphore);\
}
/*********************************************
 *        Global declarations
 *********************************************/
extern QueueHandle_t myQueue_light, myQueue_ultra, myQueue_log,
myQueue_water, myQueue_heartbeat;
extern int CN_ACTIVE ;
extern int8_t mode;
extern uint32_t DEGRADED_MODE_MANUAL;
extern SemaphoreHandle_t xSemaphore;

/*********************************************
 *        Function Prototypes
 *********************************************/
/*******************************************
 * Func name : queue_init
 * Parameters: none
 * Description : initiates the queues for logger
 */
void queue_init();

/*******************************************
 * Func name :   LogTask
 * Parameters:   none
```

```
 * Description : Thread for logger task
 ******************************************/
void LogTask(void *pvParameters);


/******************************************
 * Func name :   UART_send
 * Parameters:   Address, length
 * Description : Uart function to send sensor values to the control node
 ******************************************/
void UART_send(char* ptr, int len);

/******************************************
 * Func name :   UART_send_log
 * Parameters:   Address, length
 * Description : Uart function to send log data to the control node
 ******************************************/
void UART_send_log(char* ptr, int len);
#endif /* LOG_H_ */
/*
 * heartbeat.h
 *
 *  Created on: Apr 23, 2019
 *      Author: Steve
 */

#ifndef INC_HEARTBEAT_H_
#define INC_HEARTBEAT_H_

/******************************************
 *               Includes
 ******************************************/
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>

#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"
#include "timer.h"

#include "driverlib/gpio.h"
#include "driverlib/inc/hw_memmap.h"
#include "driverlib/inc/hw_timer.h"
#include "driverlib/inc/hw_types.h"

#include "driverlib/sysctl.h"
#include "driverlib/rom.h"
#include "utils/uartstdio.h"
#include "driverlib/inc/hw_ints.h"
#include "driverlib/fpu.h"
```

```
#include "driverlib/gpio.h"
#include "drivers/pinout.h"

#include "motor_driver.h"

/*********************************************
 *               Function Prototypes
 *********************************************/
/*Heartbeat task*/
void Control_Node_heartbeat(void *pvParameters);

/*Heartbeat Timer handler*/
void vTimerCallback_HB_handler( TimerHandle_t  *pxTimer );

/*********************************************
 *               Global declarations
 *********************************************/
extern QueueHandle_t myQueue_heartbeat;
extern uint32_t DEGRADED_MODE_MANUAL;

#endif /* INC_HEARTBEAT_H_ */



#ifndef MOTOR_DRIVER_H_
#define MOTOR_DRIVER_H_


/*
 * motor_driver.h
 *
 *  Created on: Apr 14, 2019
 *      Author: Steve Antony
 */

/*********************************************
 *        Includes
 *********************************************/
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>

#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"
#include "timer.h"

#include "driverlib/gpio.h"
#include "driverlib/inc/hw_memmap.h"
#include "driverlib/inc/hw_timer.h"
#include "driverlib/inc/hw_types.h"
```

```
#include "driverlib/sysctl.h"
#include "driverlib/rom.h"
#include "utils/uartstdio.h"
#include "driverlib/inc/hw_ints.h"
#include "driverlib/fpu.h"


/*********************************************
 *        Function Prototypes
 *********************************************/
/*
-------------------------------------------------------------------------------
-------------------
init_motor
-------------------------------------------------------------------------------
-------------------
*   This functions is used to initiate the motor output pins
*
*   @\param         void
*
*   @\return        void
*
*/
void init_motor();

/*
-------------------------------------------------------------------------------
-------------------
stop
-------------------------------------------------------------------------------
-------------------
*   This functions stops the motion of robot
*
*   @\param         void
*
*   @\return        void
*
*/
void stop();

/*
-------------------------------------------------------------------------------
-------------------
forward
-------------------------------------------------------------------------------
-------------------
*   This functions moves the robot forward
*
*   @\param         void
*
*   @\return        void
*
*/
```

```
void forward();

/*
------------------------------------------------------------------------
-------------------
left
------------------------------------------------------------------------
-------------------
*   This functions turns the robot left
*
*   @\param         void
*
*   @\return        void
*
*/
void left();

/*
------------------------------------------------------------------------
-------------------
right
------------------------------------------------------------------------
-------------------
*   This functions turns the robot right
*
*   @\param         void
*
*   @\return        void
*
*/
void right();

/*
------------------------------------------------------------------------
-------------------
backward
------------------------------------------------------------------------
-------------------
*   This functions moves the robot backward
*
*   @\param         void
*
*   @\return        void
*
*/
void backward();




#endif
/*
 * main.h
 *
```

```
 *  Created on: Apr 20, 2019
 *      Author: Steve
 */

#ifndef MAIN_H_
#define MAIN_H_

/*********************************************
 *              MACROS
 *********************************************/
// System clock rate, 120 MHz
#define SYSTEM_CLOCK    (120000000U)

#define QUEUE_TIMEOUT_TICKS (10)

/*********************************************
 *              GLOBAL DECLARATION
 *********************************************/
extern uint32_t DEGRADED_MODE_MANUAL;

/*********************************************
 *              Function Prototypes
 *********************************************/
/*
-------------------------------------------------------------------------------
-------------------
ConfigureUART2
-------------------------------------------------------------------------------
-------------------
*   This configures UART2
*
*   @\param         none
*
*   @\return        none
*
*/
void ConfigureUART2();

/*
-------------------------------------------------------------------------------
-------------------
ConfigureUART1
-------------------------------------------------------------------------------
-------------------
*   This configures UART1
*
*   @\param         none
*
*   @\return        none
*
*/
void ConfigureUART1();

/*
```

```
------------------------------------------------------------------------
-------------------
ConfigureUART3
------------------------------------------------------------------------
-------------------
*   This configures UART3
*
*   @\param         none
*
*   @\return        none
*
*/
void ConfigureUART3();
#endif /* MAIN_H_ */
/*
 * waterlevel.h
 *
 *  Created on: Apr 24, 2019
 *      Author: Steve
 */

#ifndef INC_WATERLEVEL_H_
#define INC_WATERLEVEL_H_


/**********************************************
 *        Includes
 **********************************************/
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>

#include "FreeRTOSConfig.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"
#include "timer.h"

#include "driverlib/gpio.h"
#include "driverlib/inc/hw_memmap.h"
#include "driverlib/inc/hw_timer.h"
#include "driverlib/inc/hw_types.h"

#include "driverlib/sysctl.h"
#include "driverlib/rom.h"
#include "utils/uartstdio.h"
#include "driverlib/inc/hw_ints.h"
#include "driverlib/fpu.h"
#include "driverlib/gpio.h"
#include "drivers/pinout.h"
#include "driverlib/adc.h"
#include "log.h"
```

```
/*********************************************
 *         Global declaration
 *********************************************/
extern QueueHandle_t myQueue_water;


/*********************************************
 *         Function prototypes
 *********************************************/
/*********************************************
          Water level task
 *********************************************/
void Water_level(void *pvParameters);

/*********************************************
      Timer callback for water level task
 *********************************************/
void vTimerCallback_WaterLevel_handler( TimerHandle_t  *pxTimer );

/*********************************************
 * Func name :   init_valve
 * Parameters:   none
 * Description : initiates the valve control pin
 *********************************************/
void init_valve();

/*********************************************
 * Func name :   close_value
 * Parameters:   none
 * Description : close the water valve
 *********************************************/
void close_value();

/*********************************************
 * Func name :   open_value
 * Parameters:   none
 * Description : open the water valve
 *********************************************/
void open_value();


#endif /* INC_WATERLEVEL_H_ */
```