```
/* ========================================
 *
 * Copyright Team Lamp, 2019
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/
#include "project.h"
#include "stdio.h"
#include <math.h>

#define STOP 255
#define CW 0
#define CCW 1
#define FOUND 1
#define LOST 0
#define PI 3.14159265
#define g 9.81

/*                  *\
   Global Variables
\*                  */

char state = 'S';
char laststate = 'X';

/*              *\
     UART_COM
\*              */

uint8 receive = 1;
uint8 transmit = 0;
char datapacket[256];

/*              *\
       LIDAR
\*              */

uint8 cb = 0;
uint8 lindex = 0;
uint8 bytes[9];
uint16 cm = 0;
uint16 signal = 0;
uint16 cmlast = 0;
uint32 to = 0;

//Compare cm
uint8 compareCm = 0;
```

```
uint8 objectdet = FOUND;

uint16 cmsum = 0;
uint16 cmnum = 0;
uint16 cmavg = 0;
uint16 cmavglast = 0;

/*              *\
    | Trajectory
\*              */
//double z_hoop = 2.5;
double z_hoop = 0.86;
double z_robot = 0.48;
double z;
double phi;
double v;

/*              *\
    | Spinner
\*              */
uint16 PWM_Compare_Min = 670;
uint16 PWM_Compare_Max = 2330;
uint16 PWM_Compare_Neutral = 1496;
int32 rpm_des;
int32 rpm = 0;
int32 spin_counter = 0;
int32 spin_counter_last = 0;
uint8 spinner_index = 0;

/*              *\
    | Steppers
\*              */

uint8 stepper = 1;

//Stepper 1
uint8 dir = STOP;
uint8 dirlast = CW;
uint8 break_counter = 5;
int8 step_index = 0;
uint32 step_counter = 0;
uint32 sweep = 20;
uint32 sweep_min = 8;
uint32 sweep_max = 160;

//Stepper 2
uint32 step_counter_2 = 0;
uint32 step_des = 4844;
uint32 step_max = 10030;
uint8 step_index_2 = 0;
uint8 comcount = 20;
```

```
/*                                                                          *\
------------------------------Interrupt 1-----------------------------------

During Idle State (state = "S") the interrupt triggers once every 20ms.

    -Current value of cm is compared with the previous value, and used
     to determine whether object is LOST or FOUND.

During Arm/Disarm/Exit (state = "A","D","X") interrupt triggers once every 1ms.

    -Spinner ramps to desired rpm value. For states "D" and "X" rpm_des = 0.
     For "A" rpm_des is determined by an equation in main().
\*                                                                           */

CY_ISR(isr_1)
{
    if (stepper == 2)
    {
        //The spinner is only controlled once every 100 cycles (100ms) 
because otherwise the resolution for current rpm value is too low.
        if (spinner_index == 99)
        {
            spin_counter_last = spin_counter;
            spin_counter = QuadDec_GetCounter();
            rpm =  ((abs(spin_counter-spin_counter_last)*600)/20);
            if (rpm_des != 0)
            {
                uint16 PWM_Compare = PWM_ReadCompare();
                if (rpm < rpm_des && PWM_Compare > PWM_Compare_Min)
                {
                    if (rpm < 0.9 * rpm_des)
                    {
                        PWM_Compare -= 3;
                        PWM_WriteCompare(PWM_Compare);
                    }
                    else if (rpm > 0.9 * rpm_des)
                    {
                        PWM_Compare--;
                    }
                }
                else if (rpm > rpm_des && PWM_Compare < PWM_Compare_Max)
                {
                    if (rpm > 1.1 * rpm_des)
                    {
                        PWM_Compare += 2;
                        PWM_WriteCompare(PWM_Compare);
                    }
                    else if (rpm < 1.1 * rpm_des)
                    {
                        PWM_Compare++;
                        PWM_WriteCompare(PWM_Compare);
                    }
                }
            }
            else if (rpm_des == 0)
            {
                PWM_WriteCompare(PWM_Compare_Neutral);
            }
            spinner_index = 0;
        }
        spinner_index++;
    }
    else
    {
        //LIDAR's error value is 65533
        if (cm == 65533)
        {
            cm = cmlast;
        }
        //Object is LOST when distance jumps by 50cm or more.
        //Object is FOUND when distance falls by 50cm or more.
        //dirlast can't be set to STOP or else the yaw stepper won't turn ↩
when object is lost again.
        //Sweep value is restored to minimum.
        if (cm > cmlast + 50)
        {
            objectdet = LOST;
            dir = dirlast;
        }
        else if (cm < cmlast - 50)
        {
            objectdet = FOUND;
            if (dir != STOP)
            {
                dirlast = (dir+1)%2;
            }
            sweep = sweep_min;
            dir = STOP;
        }
        //When step_counter has reached sweep_max and the object is still LOST↩
, sweep_max is increased, and the direction is switched.
        if (objectdet == LOST && step_counter == sweep )
        {
            step_counter = 0;
            sweep += sweep;
            dir = (dir+1)%2;
        }
    }
    //Writing to the control register and reading the status register ↩
activates interrupt 2.
    Control_Reg_Write(1);
    Status_Reg_Read();
}
```

```
/*                                                                          *\
----------------------------Interrupt 2-------------------------------------

Interrupt is triggered immediately after Interrupt 1.

During Idle State (state = "S") the interrupt triggers once every 20ms.

    -dir value set during Interrupt 1 is used to determine what values to send
     to the yaw stepper. The stepper performs one step every cycle (20ms). The
     stepper_index counter determines what values to put to the stepper to
     make it continue turning a certain direction.

During Arm/Disarm/Exit (state = "A","D","X") interrupt triggers once every 1ms.

    -Yaw stepper is stopped.
    -Pitch stepper goes to desired step value. For states "D" and "X"
     step_des = 0. For "A" step_des is determined by an equation in main().
\*                                                                           */

CY_ISR(isr_2)
{
    if (stepper == 1)
    {
        if (dir == STOP)
        {
            A_1_Write(0);
            B_1_Write(0);
            C_1_Write(0);
            D_1_Write(0);
            step_index = 0;
            step_counter = 0;
        }
        else if (dir == CW)
        {
            if (step_index == 0)
            {
                A_1_Write(1);
                B_1_Write(0);
                C_1_Write(1);
                D_1_Write(0);
                step_index++;
            }
            else if (step_index == 1)
            {
                A_1_Write(1);
                B_1_Write(0);
                C_1_Write(0);
                D_1_Write(1);
                step_index++;
            }
            else if (step_index == 2)
            {
                A_1_Write(0);
                B_1_Write(1);
                C_1_Write(0);
                D_1_Write(1);
                step_index++;
            }
            else
            {
                A_1_Write(0);
                B_1_Write(1);
                C_1_Write(1);
                D_1_Write(0);
                step_index = 0;
            }
            step_counter++;
        }
        else if (dir == CCW)
        {
            if (step_index == 0)
            {
                A_1_Write(1);
                B_1_Write(0);
                C_1_Write(0);
                D_1_Write(1);
                step_index++;
            }
            else if (step_index == 1)
            {
                A_1_Write(1);
                B_1_Write(0);
                C_1_Write(1);
                D_1_Write(0);
                step_index++;
            }
            else if (step_index == 2)
            {
                A_1_Write(0);
                B_1_Write(1);
                C_1_Write(1);
                D_1_Write(0);
                step_index++;
            }
            else
            {
                A_1_Write(0);
                B_1_Write(1);
                C_1_Write(0);
                D_1_Write(1);
                step_index = 0;
            }
            step_counter++;
        }
```

```
}
else if (stepper == 2)
{
    A_1_Write(0);
    B_1_Write(0);
    C_1_Write(0);
    D_1_Write(0);
    if (step_counter_2 == step_des)
    {
        A_2_Write(0);
        B_2_Write(0);
        C_2_Write(0);
        D_2_Write(0);
        step_index_2 = 0;
        //Return to Idle state if the current state is Disarm
        //and the rpm and pitch angle are 0.
        if ((state == 'D' || state == 'd') && rpm == 0)
        {
            stepper = 1;
            comcount = 5;
            Timer_1_WritePeriod(160000);
            state = 'S';
        }
    }
    else if (step_counter_2 < step_des)
    {
        if (step_index_2 == 0)
        {
            A_2_Write(1);
            B_2_Write(0);
            C_2_Write(1);
            D_2_Write(0);
            step_index_2++;
        }
        else if (step_index_2 == 1)
        {
            A_2_Write(1);
            B_2_Write(0);
            C_2_Write(0);
            D_2_Write(1);
            step_index_2++;
        }
        else if (step_index_2 == 2)
        {
            A_2_Write(0);
            B_2_Write(1);
            C_2_Write(0);
            D_2_Write(1);
            step_index_2++;
        }
        else
        {
            A_2_Write(0);
            B_2_Write(1);
            C_2_Write(1);
            D_2_Write(0);
            step_index_2 = 0;
        }
        step_counter_2++;
    }
    else if (step_counter_2 > step_des)
    {
        if (step_index_2 == 0)
        {
            A_2_Write(1);
            B_2_Write(0);
            C_2_Write(0);
            D_2_Write(1);
            step_index_2++;
        }
        else if (step_index_2 == 1)
        {
            A_2_Write(1);
            B_2_Write(0);
            C_2_Write(1);
            D_2_Write(0);
            step_index_2++;
        }
        else if (step_index_2 == 2)
        {
            A_2_Write(0);
            B_2_Write(1);
            C_2_Write(1);
            D_2_Write(0);
            step_index_2++;
        }
        else
        {
            A_2_Write(0);
            B_2_Write(1);
            C_2_Write(0);
            D_2_Write(1);
            step_index_2 = 0;
        }
        step_counter_2--;
    }
}
comcount--;
if (comcount == 0)
{
    transmit = 1;
}
Control_Reg_Write(0);
Status_Reg_Read();
```

```
}

/*                                                                          *\
---------------------------Get LIDAR Data-----------------------------------

    Collect bytes, perform checksum, then set cm to Byte 3 + Byte 4

    Byte 1 | 0x59
    Byte 2 | 0x59
    Byte 3 | Low,  8-bit Distance
    Byte 4 | High, 8-bit Distance
    Byte 5 | Low,  8-bit Strength
    Byte 6 | High, 8-bit Strength
    Byte 7 | Reserved Byte
    Byte 8 | Original Signal Quality Degree
    Byte 9 | Checksum

\*                                                                           */

void GetLidarData(void)
{
    for(;;)
    {
        uint8 cb = UART_LIDAR_GetRxBufferSize();
        if (cb>0)
        {
            bytes[lindex] = UART_LIDAR_ReadRxData();
            if (lindex == 0 && bytes[0] == 0x59)
            {
                lindex++;
                continue;
            }
            if (lindex == 1 && bytes[1] != 0x59)
            {
                lindex = 0;
                continue;
            }
            if (lindex >= 1 && lindex < 9)
            {
                lindex++;
                continue;
            }
            if (lindex == 9)
            {
                uint8 checksum = 0;
                for(int i = 0; i < 8; i++)
                {
                    checksum += bytes[i];
                }
                if(checksum == bytes[8])
                {
                    lindex = 0;
                    cmlast = cm;
                    cm = (bytes[3] << 8) + bytes[2];
                    signal = (bytes[5] << 8) + bytes[4];
                    cmsum += cm;
                    cmnum++;
                    return;
                }
                else
                {
                    lindex = 0;
                    continue;
                }
                lindex = 0;
            }
        }
    }
}


int main(void)
{

    UART_COM_Start();
    UART_LIDAR_Start();

    isr_1_Start();
    isr_1_StartEx(isr_1);

    isr_2_Start();
    isr_2_StartEx(isr_2);

    Timer_1_Start();
    Status_Reg_InterruptEnable();

    PWM_Start();
    QuadDec_Start();

    CyGlobalIntEnable;

    for(;;)
    {
        if (receive == 1)
        {
            if (UART_COM_GetRxBufferSize()!=0)
            {
                laststate = state;
                state = UART_COM_GetChar();

                /*

                    Allowed:
                    X -> S
```

```
                        Set stepper to 1. Change timer period to 160000 (20ms
)
                    S -> A
                        If object is LOST, set state back to "S." Otherwise
                        calculate rpm_des & step_des based on current cm.
                    A -> D
                        Set rpm_des & step_des to 0.
                    Any State -> X
                        Set rpm_des & step_des to 0. Set stepper to 2. Set
                        timer period to 8000 (1ms)

                    All other state transitions not allowed.

                */
                if ((laststate == 'X' || laststate == 'x') && (state == 'S'
||state == 's'))
                {
                    if (step_counter_2 == 0)
                    {
                        comcount = 5;
                        stepper = 1;
                        Timer_1_WritePeriod(160000);
                        state = 'S';
                    }
                    else
                    {
                        state = 'X';
                    }
                }
                else if ((laststate == 'A' || laststate == 'a') && (state == '
D' ||state == 'd'))
                {
                    step_des = 0;
                    rpm_des = 0;
                }
                else if ((laststate == 'S' || laststate == 's') && (state == '
A' ||state == 'a'))
                {
                    if (objectdet == LOST)
                        {
                            state = 'S';
                            continue;
                    }
                        else
                        {
                            double m = (cm+79+30)/100;
                            z = z_hoop - z_robot;
                            phi = 0.5 * atan(z/m);
                            v = sqrt(g*(z+sqrt(z*z+m*m)));
                            step_des = round(46350*sin(phi/2));
                            if (v < 7)
                            {
                                rpm_des = round(5.2628*v*v + 197.34*v+2.7248)↩
+300;
                            }
                            else
                            {
                                rpm_des = round(265.87*v*v-3089.2*v+10241)+300;
                            }
                            if (step_des > step_max)
                            {
                                    step_des = step_max;
                            }
                            comcount = 100;
                            stepper = 2;
                            Timer_1_WritePeriod(8000);
                        }
                }
                else if (state == 'X' || state == 'x')
                {
                    comcount = 100;
                    sweep_max = 8;
                    stepper = 2;
                    Timer_1_WritePeriod(8000);
                    step_des = 0;
                    rpm_des = 0;
                }
                else
                {
                    state = laststate;
                }
            }
            receive = 0;
        }

        switch(state)
        {
            case 'X' :
            case 'x' :
                cm = cmlast;
                A_2_Write(0);
                B_2_Write(0);
                C_2_Write(0);
                D_2_Write(0);
                if(transmit == 1)
                {
                    sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, ↩
step_counter_2, rpm);
                    UART_COM_PutString(datapacket);
                    receive = 1;
                    transmit = 0;
                    comcount = 100;
                }
            break;
            case 'S' :
            case 's' :
                GetLidarData();
                if(transmit == 1)
                {
                    sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, ↩
step_counter_2, rpm);
                    UART_COM_PutString(datapacket);
                    receive = 1;
                    transmit = 0;
                    comcount = 5;

                }
            case 'A' :
            case 'a' :
                if(transmit == 1)
                {
                    sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, ↩
step_counter_2, rpm);
                    UART_COM_PutString(datapacket);
                    receive = 1;
                    transmit = 0;
                    comcount = 100;
                }
            break;
            case 'D' :
            case 'd' :
                if(transmit == 1)
                {
                    sprintf(datapacket, "%d,%d,%lu,%lu\r\n", cm, objectdet, ↩
step_counter_2, rpm);
                    UART_COM_PutString(datapacket);
                    receive = 1;
                    transmit = 0;
                    comcount = 100;
                }
            break;
        }
    }
}
```