```
#define F_CPU 16000000UL
#include <avr/io.h>
#include "serial.h"
#include <util/delay.h>
#include <avr/interrupt.h>

/*****************************************************************************
                                 GLOBAL VARIABLES
*****************************************************************************/
int g_period;           /* period of color that is sensed by color sensor */
int g_timer;                /* timer 1 time value (used by color sensor to
                               calculate the period) */
const int DEAD_TIME=10;     /* small deadtime between pwn output waves in order
                               to prevent shorting out the h-bridges */
int g_setPoint2;            /* set point for the speed of right motor using
                               timer 2 */
int g_setPoint0;            /* set point for the speed of left motor using
                               timer 0 */
int g_isblue=0;             /* int that acts like a boolean to determine if we
                               are starting on the blue side */
volatile int g_isLineCenter=1;  /* int that acts like a boolean to check if
                                   center qti sensor is triggered
                                   WIRED WITH PULLUP RESISTOR */
volatile int g_isLineLeft=1;    /* int that acts like a boolean to check if
                                   left qti sensor is triggered
                                   WIRED WITH PULLUP RESISTOR */
volatile int g_isLineRight=1;   /* int that acts like a boolean to check if
                                   right qti sensor is triggered
                                   WIRED WITH PULLUP RESISTOR */
int g_isLoopClockwise=-1;   /* int that acts like a boolean to check if we
                               started off driving clockwise */
int g_isLoopDetected=-1;    /* int that acts like a boolean to check if we
                               have identified which direction we started
                               driving in */
int g_isQtiTriggered=0;     /* int that acts like boolean to check if any qti
                               sensor has been triggered */
int g_onOtherSide=0;        /* int that acts like boolean to check if we are on
                               the opposite side */
int g_turnCounter=0         /* int that counts number of times we have turned */
int g_middleDriveCounter=0;     /* int that counts how long we drive in the
                                   middle */

/*****************************************************************************
                                 INTERRUPTS
*****************************************************************************/
/*****************************************************************************
* NAME :            ISR(PCINT2_vect)
*
* DESCRIPTION :     Pin change interrupt service routine for the color
*                   sensor (pin7/PD7), center qti sensor (pin2/PD2), and left
*                   qti sensor (pin4/PD4). Triggers on rising edge.
*/
```

```
ISR(PCINT2_vect) {
    if (PIND & 0b10000000) { //if color sensor (pin7/PD7) is rising edge
      TCNT1=0; //reset timer
    }
    else {
      g_timer=TCNT1; //save current period
    }
    if (PIND & 0b00000100) { //if center qti sensor (pin2/PD2) is triggered
      g_isLineCenter=0; //center qti is sensing a line
    }
    else {
      g_isLineCenter=1; //center qti is not sensing a line
    }
    if (PIND & 0b00010000) { //if left qti sensor (pin4/PD4) is triggered
      g_isLineLeft=0; //left qti sensor is sensing a line
    }
    else {
      g_isLineLeft=1; //left qti sensor is not sensing a line
    }
}
/*****************************************************************************
* NAME :            ISR(PCINT0_vect)
*
* DESCRIPTION :     Pin change interrupt service routine for the right qti
*                   sensor (pin8/PB0). Triggers on rising edge.
*/
ISR(PCINT0_vect) {
    if (PINB & 0b00000001) { //if right qti sensor (pin8/PB0) is triggered
      g_isLineRight=0; //right qti sensor is sensing a line
    }
    else {
      g_isLineRight=1; //right qti sensor is not sensing a line
    }
}

/*****************************************************************************
                              INITIALIZING FUNCTIONS
****************************************************************************/
/*****************************************************************************
* NAME :            void initSensors()
*
* DESCRIPTION :     Initializes registers that LED and color sensor are
*                   connected to and sets up timer 1
*/
void initSensors() {
    DDRD=0b00000000;    //set pin7 (color sensor, PD7, PCINT23) to input,
                        //pin 2 (qti center, PD2, PCINT18) to input
                        //and pin 4 (qti left, PD4, PCINT20) to input
    DDRB=0b00000000;    //set pin 8 (qti right, PB0, PCINT0) as input
    /*set up pin change interrupt registers for all sensors*/
    PCICR=0b00000101;   //enable PCMSK2 and PCMSK0
    PCMSK2=0b00010100;  //enable qti center (PCINT18) and qti left (PCINT20)
    PCMSK0=0b00000001;  //enable qti right (PCINT0)
    sei();              //enable all interrupts
    /*set up timer 1*/
    TCCR1A=0b00000000;  //set timer to normal mode
    TCCR1B=0b00000001;  //set prescaler to 1
}
/****************************************************************************
* NAME :            void initMotors()
*
* DESCRIPTION :     Initializes registers that motors are connected to and
*                   sets up timers 0 and 2
*/
void initMotors() {
    DDRB |= 0b00001000; // set Arduino pin 11 (PB3, OC2A) as output
    DDRD |= 0b01101000; //set Arduino pins 5 & 6 (OC0B and OC0A) as outputs
                        //and pin 3 (PD3, OC2B) as output
    TCCR0A=0b10110001;  //clear OC0A on up-counting, set on down-counting;
                        //clear OC0B on down-counting, set on up-counting
    TCCR0B=0b00000001;  //waveform generation mode 1 PWM, phase correct
    TCCR2A=0b10110001;  //clear OC2A on up-counting, set on down-counting;
                        //clear OC2B on down-counting, set on up-counting
    TCCR2B=0b00000001;  //waveform generation mode 1 PWM, phase correct
}

/*****************************************************************************
                                SENSOR FUNCTIONS
*****************************************************************************/
/*****************************************************************************
* NAME :            void getColor()
*
* DESCRIPTION :     Gets the period from the color sensor by triggering the
*                   interrupt for 5ms
*/
void getColor() {
    PCMSK2=0b10000000; //enable color sensor (PCINT23)
    _delay_ms(5); //delay 5 ms to give interrupt time to trigger
    PCMSK2=0b00000000; //disable color sensor (PCINT23)
    g_period=g_timer*.0625*2; //convert ticks to microseconds
  }

/*****************************************************************************
                                DRIVING FUNCTIONS
*****************************************************************************/
/*****************************************************************************
* NAME :            void drive(int sp2, int sp0)
*
* DESCRIPTION :     Drives robot using PWM setpoints and checks if qti
*                   sensors are being triggered during driving and adjusts
*                   accordingly.
*                   setPoint_=10 is max speed backwards
*                   setPoint_=245 is max speed forwards
*                   setPoint_=118 is stopped
* INPUTS :
*       PARAMETERS:
*           int     sp2               right motor setPoint
*           int     sp0               left motor setPoint
*/
void drive(int sp2, int sp0) {
    //IF WE ARE BACK ON OUR SIDE ON A LINE
    if (g_turnCounter==7) {
        //STOP
        g_setPoint2=118; //PWM set point for right motor
        OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers such
        OCR2B=g_setPoint2+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
        g_setPoint0=118; //PWM set point for left motor
        OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers such
        OCR0B=g_setPoint0+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
        _delay_ms(150); //delay in ms to drive
    }
    //IF ALL QTI SENSORS ARE SENSING BLACK
    else if(g_isLineCenter==0 && g_isLineLeft==0 && g_isLineRight==0) {
        g_isQtiTriggered=1; //a qti has been triggered
        //DRIVE BACKWARDS
        g_setPoint2=10; //PWM set point for right motor
        OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers such
        OCR2B=g_setPoint2+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
        g_setPoint0=10; //PWM set point for left motor
        OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers such
        OCR0B=g_setPoint0+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
        _delay_ms(200); //delay in ms to drive
        //IF LOOPING COUNTER CLOCKWISE
        if((g_isLoopClockwise==0 && g_turnCounter<3) ||
           (g_isLoopClockwise==1 && g_turnCounter>2)) {
            //TURN LEFT
            g_setPoint2=245; //PWM set point for right motor
            OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers
            OCR2B=g_setPoint2+DEAD_TIME; //such that OCR2B>OCR2A ensuring
                                         //dead time
            g_setPoint0=10; //PWM set point for left motor
            OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers
            OCR0B=g_setPoint0+DEAD_TIME; //such that OCR2B>OCR2A ensuring
                                         //dead time
            _delay_ms(150); //delay in ms to drive
        }
        //IF LOOPING CLOCKWISE
        else {
            //TURN RIGHT
            g_setPoint2=10; //PWM set point for right motor
            OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers
            OCR2B=g_setPoint2+DEAD_TIME; //such that OCR2B>OCR2A ensuring
                                         //dead time
            g_setPoint0=245; //PWM set point for left motor
            OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers
            OCR0B=g_setPoint0+DEAD_TIME; //such that OCR2B>OCR2A ensuring
```

```
                                //dead time
            _delay_ms(150); //delay in ms to drive
        }
    }
    //IF LEFT QTI IS SENSING BLACK
    else if (g_isLineLeft==0 && g_isLineRight==1) {
        g_isQtiTriggered=1; //a qti has been triggered
        //IF THIS IS THE PART OF THE LOOP WHERE WE HIT OUR BACK LINE FOR THE
        //FIRST TIME
        if (g_turnCounter==2) {
            //TURN LEFT (INTO THE LINE)
            g_setPoint2=245; //PWM set point for right motor
            OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers such
            OCR2B=g_setPoint2+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            g_setPoint0=10; //PWM set point for left motor
            OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers such
            OCR0B=g_setPoint0+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            _delay_ms(300); //delay in ms to drive
            g_turnCounter=3; //move on to next part of loop
        }
        //IF THIS IS ANY OTHER PART OF THE LOOP
        else {
            //TURN RIGHT (AWAY FROM THE LINE)
            g_setPoint2=10; //PWM set point for right motor
            OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers such
            OCR2B=g_setPoint2+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            g_setPoint0=245; //PWM set point for left motor
            OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers such
            OCR0B=g_setPoint0+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            _delay_ms(150); //delay in ms to drive
        }
        //IF THIS IS THE FIRST TIME ANY QTI IS SENSING A LINE
        if (g_isLoopDetected==-1) {
            //WE ARE LOOPING CLOCKWISE
            g_isLoopClockwise=1; //looping clockwise
            g_isLoopDetected=1; //loop direction has been determined
        }
    }
    //IF RIGHT QTI IS SENSING BLACK
    else if (g_isLineLeft==1 && g_isLineRight==0) {
        g_isQtiTriggered=1; //a qti has been triggered
        //IF THIS IS THE PART OF THE LOOP WHERE WE HIT OUR BACK LINE FOR THE
        //FIRST TIME
        if (g_turnCounter==2) {
            //TURN RIGHT (INTO THE LINE)
            g_setPoint2=10; //PWM set point for right motor
            OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers such
            OCR2B=g_setPoint2+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            g_setPoint0=245; //PWM set point for left motor
            OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers such
            OCR0B=g_setPoint0+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            _delay_ms(300); //delay in ms to drive
            g_turnCounter=3; //move on to next part of the loop
        }
        //IF THIS IS ANY OTHER PART OF THE LOOP
        else {
            //TURN LEFT (AWAY FROM THE LINE)
            g_setPoint2=245; //PWM set point for right motor
            OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers such
            OCR2B=g_setPoint2+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            g_setPoint0=10; //PWM set point for left motor
            OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers such
            OCR0B=g_setPoint0+DEAD_TIME; //that OCR2B>OCR2A ensuring dead time
            _delay_ms(150); //delay in ms to drive
        }
        //IF THIS IS THE FIRST TIME ANY QTI IS SENSING A LINE
        if (g_isLoopDetected==-1) {
            //WE ARE LOOPING COUNTERCLOCKWISE
            g_isLoopClockwise=0; //looping counterclockwise
            g_isLoopDetected=1; //loop direction has been determined
        }
    }
    //NO QTI SENSORS WERE TRIGGRED
    else {
        g_isQtiTriggered=0; //checkqti has not been triggered
        //DRIVE ACCORDING TO USER INPUTS
        g_setPoint2=sp2; //PWM set point for right motor
        OCR2A=g_setPoint2-DEAD_TIME; //set output compare registers such that
        OCR2B=g_setPoint2+DEAD_TIME; //OCR2B>OCR2A ensuring dead time
        g_setPoint0=sp0; //PWM set point for left motor
        OCR0A=g_setPoint0-DEAD_TIME; //set output compare registers such that
        OCR0B=g_setPoint0+DEAD_TIME; //OCR2B>OCR2A ensuring dead time
        _delay_ms(50); //delay in ms to drive
    }
}

/******************************************************************************
                                MAIN FUNCTION
******************************************************************************/
int main(void) {
    init_uart(); //allow printf to work
    initSensors(); //call initSensors() to initialize all sensor variables
    initMotors(); //call initMotors() to initialize all motor variables
    getColor(); //determine initial color we are on
    if (g_period>200) { //check what color we are starting on
        g_isblue=1;
    }
    while (1) {
        getColor(); //get the color from the color sensor
        //IF WE ARE ON OUR OWN SIDE
        if(((g_isblue==1 && g_period>200) ||(g_isblue==0 && g_period<200))) {
            //IF THIS IS THE PART OF THE LOOP WHERE WE HAVE JUST SENSED OUR
            //OWN BACK LINE FOR THE FIRST TIME
            if(g_turnCounter==3) {
```

```
        //IF WE ARE ON A BLACK LINE
        if(g_isQtiTriggered==1) {
            //DRIVE BACKWARDS TO GET OFF THE LINE
            for (int i=0; i<12; i++) {
                drive(10,10);
            }
            //TURN ACCORDING TO WHICH WAY WE ARE LOOPING
            for (int i=0; i<6; i++) {
                if(g_isLoopClockwise==0) {
                    drive(10,245); //turn right
                }
                else {
                    drive(245,10); //turn left
                }
            }
            g_turnCounter=4; //move on to next part of the loop
        }
    }
    //IF THIS IS THE PART OF THE LOOP WHERE WE ARE BACK ON OUR SIDE
    //FOR THE SECOND TIME
    if (g_turnCounter==6) {
        //IF WE ARE SENSING A BLACK LINE
        if (g_isQtiTriggered==1) {
            g_turnCounter=7; //move on to next part of the loop
        }
        drive(245,245); //drive forward
    }
    //IF THIS IS PART OF THE LOOP WHERE WE ARE ON A BLACK LINE ON
    //OUR SIDE
    if (g_turnCounter==7) {
        //STOP
        for (int i=0; i<6000; i++) {
            drive(118,118);
        }
    }
    drive(245,245); //drive forward
    g_onOtherSide=0; //back on our side
}
//IF WE ARE ON THE OPPOSITE SIDE
else if ((g_isblue==1 && g_period<200) ||
         (g_isblue==0 && g_period>200)) {
    //IF WE JUST CAME FROM OUR SIDE
    if(g_onOtherSide==0) { //checks if just came from own side
        //TURN ACCORDING TO WHICH DIRECTION WE ARE LOOPING
        for (int i=0; i<20; i++) {
            if((g_isLoopClockwise==0 && g_turnCounter==0)||
               (g_isLoopClockwise==1 && g_turnCounter==4)) {
                drive(245,118); //turn left
                _delay_ms(25);
            }
            else {
                drive(118,245); //turn right
```

```
                        _delay_ms(25);
                    }
                }
                if(g_turnCounter==0) { //move on to next part of loop
                    g_turnCounter=1;
                }
                if(g_turnCounter==4) { //move on to next part of loop
                    g_turnCounter=5;
                }
                g_onOtherSide=1; //on opposite side
                g_middleDriveCounter=0; //reset counter for driving to the
                                        //center
            }
            //IF WE HAVE JUST TURNED AND ARE HEADING TO THE CENTER
            else if (g_middleDriveCounter<45) {
                drive(245,245); //drive forward
                g_middleDriveCounter++;
            }
            //IF WE HAVE REACHED THE CENTER
            else {
                //SPIN IN THE CENTER, ENDING THE SPIN FACING OUR SIDE
                for (int i=0; i<14; i++) {
                    if((g_isLoopClockwise==0 && g_turnCounter==1)||
                       (g_isLoopClockwise==1 && g_turnCounter==5)) {
                        drive(245,10); //turn left
                        _delay_ms(245);
                    }
                    else {
                        drive(10,245); //turn right
                        _delay_ms(245);
                    }
                }
                g_middleDriveCounter=-40; //set counter to very low number to
                                          //drive forward
                if(g_turnCounter==1) { //move on to next part of loop
                    g_turnCounter=2;
                }
                if(g_turnCounter==5) { //move on to next part of loop
                    g_turnCounter=6;
                }
            }
        }
    }
}
```