```c
#include <msp430g2553.h>
#include "Motor_Control.h"
#include "sounds.h"
#include "buttons.h"
#include "ADC10.h"
#include <signal.h>

//One second
#define ONE_SEC 1000000

//GLOBAL VARIABLES
volatile unsigned char sawLine1, sawLine2;
volatile unsigned int sensitivity = 500;
volatile unsigned int sensitivity2 = 700;
volatile unsigned int fullSpeed = 0;
volatile unsigned int fullStop = 100;
volatile unsigned int count;
volatile unsigned char goRight,goLeft,goFull;
volatile signed int s1,s2,s3,s4,s4,s5,s6,s7,s8,s9,s10,s11;
volatile signed int sumRead;
volatile signed int totalSensors;
volatile signed int os1,os2,os3,os4,os5,os6,os7,os8,os9,os10;

//PID VARIABLES
volatile signed int err;
volatile signed int dErr;
volatile signed int iErr;
volatile signed int lastErr;
volatile signed int             sp = 0;
volatile float                  kp = 1.0;
volatile float                  ki = 0;
volatile float                  kd = 0.4;
volatile signed int os;
volatile signed int output;

//FUNCTIONS
void init();
void startUp();
void adcInit();
void initMotorControl();
inline void leftMotor(int x);
inline void rightMotor(int x);
void initSounds();
inline void tone(unsigned long z,unsigned long k);
void zap();
int buttonPress();

void main(void)
{
    //INITIALIZATIONS
    init();
    startUp();
    initSounds();
    initMotorControl();
    initAdc();

    while(buttonPress() == 0){      //Wait for button press to start program
    }
    zap();
    _bis_SR_register(GIE);          //enable global interrupts

for(;;){    //main loop
}
}
void init()
{
    WDTCTL = WDTPW + WDTHOLD;
    BCSCTL1 = CALBC1_1MHZ; //set range
    DCOCTL = CALDCO_1MHZ;//set dco step and modulation
    BCSCTL3 |= LFXT1S_2; // dco
    BCSCTL2 |= SELM_0; //sel DCO = mclk
    P1DIR |= BIT5;
    P2DIR |= BIT5;
    P1OUT &= ~BIT5;
    P2REN |= BIT3;
    P2OUT |= BIT3;
```

```
77 }
78 void startUp()  //flash led to ensure cpu running
79 {
80     unsigned int y=0;
81     for(y=0;y<5;y++){
82         P1OUT |= BIT5;
83         __delay_cycles(ONE_SEC/10);
84         P1OUT &= ~BIT5;
85         __delay_cycles(ONE_SEC/10);
86         }
87 }
88 #pragma vector = TIMER0_A0_VECTOR
89 __interrupt void Timer_A(void) {
90
91     ADC10CTL0 &= ~ENC; // disable conversion
92     while (ADC10CTL1 & BUSY)
93             // make sure ADC is done
94             ;
95         ADC10SA = (unsigned int) & adcValues[0]; // set data buffer start address
96         ADC10CTL0 |= ENC + ADC10SC; // start sampling and conversion
97         count++;
98 }
99 // ADC10 interrupt service routine
100 #pragma vector=ADC10_VECTOR
101 __interrupt void ADC10_ISR(void) {
102     //ADC is done, do something with adcValues
103     if (adcValues[2] < sensitivity){ // 00100
104         goFull = 1;
105         s5 = 1;
106             }else{s5 = 0;}
107
108     if (adcValues[4] < sensitivity){    // 10000
109             os1 = -80;
110             sawLine1 = 1;
111             sawLine2 = 0;
112             goFull =0;
113             s1 = 1;
114             }else{s1 = 0;os1 =0;}
115         if (adcValues[4] < sensitivity2 && adcValues[3] < sensitivity2){    // 11000
116             os2 = -60;
117             goFull =0;
118             s2 = 1;
119             }else{s2 = 0;os2 =0;}
120         if (adcValues[3] < sensitivity){    // 01000
121             os3 = -40;
122             goFull =0;
123             s3 = 1;
124             }else{s3 = 0;os3 =0;}
125         if (adcValues[3] < sensitivity2 && adcValues[2] < sensitivity2){    // 01100
126             os4 = -20;
127             goFull =0;
128             s4 = 1;
129             }else{s4 = 0;os4 =0;}
130
131         if (adcValues[1] < sensitivity2 && adcValues[2] < sensitivity){ // 00110
132             os5 = 20;
133             goFull =0;
134             s6 = 1;
135             }else{s6 = 0;os5 =0;}
136         if (adcValues[1] < sensitivity){    // 00010
137             os6 = 40;
138             goFull =0;
139             s7 = 1;
140             }else{s7 = 0;os6 =0;}
141         if (adcValues[0] < sensitivity2 && adcValues[1] < sensitivity2){    // 00011
142             os7 = 60;
143             goFull =0;
144             s8 = 1;
145             }else{s8 = 0;os7 =0;}
146         if (adcValues[0] < sensitivity){    // 00001
147             os8 = 80;
148             sawLine1 = 0;
149             sawLine2 = 1;
150             goFull =0;
151             s9 = 1;
152             }else{s9 = 0;os8 =0;}
```

```
153         if (adcValues[4] > sensitivity && adcValues[3] > sensitivity && adcValues[2] >
    sensitivity && adcValues[1] > sensitivity && adcValues[0] > sensitivity && sawLine1 == 1){
    //00000-1
154             //Line is all the way to the right
155             os9 = -100;
156             P1OUT |= BIT5;
157             goFull =0;
158             s10 = 1;
159             }else{P1OUT &= ~BIT5;s10 = 0;os9 =0;}
160         if (adcValues[4] > sensitivity && adcValues[3] > sensitivity && adcValues[2] >
    sensitivity && adcValues[1] > sensitivity && adcValues[0] > sensitivity && sawLine2 ==
    1){ //1-00000
161             //Line is all the way to the left
162             os10 = 100;
163             P1OUT |= BIT5;
164             goFull =0;
165             s11 = 1;
166             }else{P1OUT &= ~BIT5;s11 = 0;os10 =0;}
167         if (adcValues[2] > sensitivity){
168                 iErr = iErr + err;
169         }else{iErr = 0;}
170
171         //OUTPUT CALCULATIONS
172         totalSensors = s1 + s2 + s3 + s4 + s5 + s6 + s7 + s8 + s9 + s10 + s11;
173         sumRead = os1 + os2 + os3 + os4 + os5 + os6 + os7 + os8 + os9 + os10;
174         //AVERAGE OF READINGS
175         err = sp - (sumRead/totalSensors);
176         if(err < 0){
177             goRight =1;
178         }else if(err>0){
179             goRight =0;
180         }
181
182         dErr = err + lastErr;
183         lastErr = err;
184
185         output = kp * err + kd * lastErr + ki * iErr;
186
187         if(goFull == 0){
188         if(goRight == 1){
189             leftMotor(output*-1);
190             rightMotor(fullSpeed);
191         }else if(goRight == 0){
192             leftMotor(fullSpeed);
193             rightMotor(output);
194         }
195             }else if(goFull == 1){
196             leftMotor(fullSpeed);
197             rightMotor(fullSpeed);
198         }
199 }
200
201
```