Light-Following Robotic Car

Supervisor: Dr. Esam Qaralleh

Embedded Systems Final Design Project, Fall 2024/2025

King Abdullah || School of Engineering

Princess Sumaya University for Technology

R.Amer 20201088

M.Rowad 20200726

J.Tutanji 20200247

Table of Contents

Abstract

The light-following robotic car is an embedded systems project developed to implement navigation and real-time decision-making. This car will be able to follow a light source by detecting light with the help of LDRs, while an ultrasonic sensor enables safe operation by avoiding obstacles. The system is powered by the PIC16F877A microcontroller, integrating sensor data and driving the servo motors through an H-Bridge motor driver. During operation, the car will stop at the detection of an obstacle, compute the best path by using some pre-programmed maneuvers, and then continue light tracking smoothly. This project represents embedded systems as an integration of electrical, software, and mechanical design. With its steady performance, the light-following car serves as a low-cost prototype of advanced robotics applications and demonstrates real-world sensor fusion and automation.

Introduction and Background

The light-following robotic car is a system designed for autonomous navigation and obstacle avoidance. Utilizing LDR light sensors, the car tracks and moves toward a light source while ensuring safe operation through an ultrasonic sensor that detects obstacles in its path. The car halts upon detecting an obstacle and intelligently navigates around it by executing predefined right and left movements. By integrating a PIC16F877A microcontroller with servo motors, an H-Bridge motor driver, and a robust power supply system, this project highlights the practical applications of embedded systems in autonomous vehicles and automation.

Design (Mechanical, Electrical, Software)

The project aims to develop an autonomous car capable of detecting and following a light source while avoiding obstacles in its path. This is achieved by integrating light-dependent resistors (LDRs) to detect light intensity and ultrasonic sensors to measure distances from nearby obstacles. A microcontroller processes the input data and sends signals to the motors to adjust the car's movement accordingly.

The system is divided into three main designs: electrical, software, and mechanical. The electrical design focuses on connecting and configuring all components such as sensors, motors, and the microcontroller. The software design ensures the car responds correctly to environmental changes by using real-time data to make decisions. The mechanical design involves a robust chassis and strategic placement of components, allowing efficient functionality and maneuverability.

Software:

```
//Ultrasonic  Trig  RC6 ,, Echo RC7

// SERVO PIN RC2

// PWM  EN MOTORS RC1

// DiRECTION PINS RB0 , RB1 , RB2 , RB3

// LDR RD0 , RD1 , RD2

// Analog LDR RA0

//Start  Buttno RD3

//LEDS Pin RD6, RD7


int tick;

int tick1;

unsigned int sensor_voltage;

unsigned char H_L;

unsigned int angle;

int distance;

int a;

int b;




void initialize();

void interrupt();
```

```
void stop_moving();

void move_left();

void move_right();

void move_forward();

void move_backwards();

void adjust_position();

void ENDD();

int dist();



void check_start(){

if(PORTD & 0B00001000){

     b++;

   }

   if(b%2==0){

   b=0;

   }else{b=1;}



}
```

```
void CCPPWM_init(){              // Configure and CCP2 at 2ms period with 50%
duty cycle

      T2CON = 0x07;              // Enable Timer2 at Fosc/4 with 1:16 prescaler (8
uS percount 2000uS to count 250 counts)

      CCP2CON = 0x0C;             // Enable PWM for CCP2

      PR2 = 250;                 // 250 counts = 8uS *250 = 2ms period

      CCPR2L = 125;              // Buffer where we are specifying the pulse width
(duty cycle)

}



void Speed(int p){

     CCPR2L = p;             // PWM from RC1

}


void mymsDelay(int x);


void ATD_init_A0();

unsigned int ATD_read_A0();


void check_front_right();

void check_front_left();

void check_front();
```

```
void main() {

    initialize();
    ATD_init_A0();
    CCPPWM_init();
    b=0;
    a=0;
    while(1){
      PORTD=PORTD & 0B11011111;
      check_start();

          while(b){
          PORTD=PORTD | 0B00100000;
          check_front_right();
          check_start();
          check_front_left();
          check_start();
          check_front();
          check_start();
          adjust_position();
          check_start();
          ENDD();
```

```
        check_start();
}           }
}


void initialize(){
TRISA=0X01;
TRISB=0X00;
TRISC=0B10000000;
TRISD=0B00001111;

PORTA=0X00;
PORTB=0X00;
PORTC=0X00;
PORTD=0X00;


  OPTION_REG= 0x87;//Use internal clock Fosc/4 with a prescaler of 256
  TMR0=248;// will count 8 times before the overflow (8* 128uS = 1ms)
  INTCON = 0b11100000; //GIE and , T0IE, peripheral interrupt

  T1CON=0x01;
```

```
  TMR1H=0;

  TMR1L=0;


  CCP1CON=0x08;

  PIE1=PIE1|0x04;// Enable CCP1 interrupts

  CCPR1H=2000>>8;

  CCPR1L=2000;


  H_L = 1;

}





void interrupt(){

   if(INTCON & 0x04){// TMR0 Overflow interrupt, will get here every 1ms

     TMR0=248;

     tick++;

     tick1++;

     check_start();

     INTCON = INTCON & 0xFB;//Clear T0IF

     }
```

```
if(PIR1&0x04){//CCP1 interrupt
  if(a==1){                          // CCP1 interrupt
        if(H_L){                     // high
              CCPR1H = angle >> 8;
              CCPR1L = angle;
              H_L = 0;               // next time low
              CCP1CON = 0x09;        // compare mode, clear output on match
              TMR1H = 0;
              TMR1L = 0;
        }
        else{                        //low
              CCPR1H = (40000 - angle) >> 8;     // 40000 counts correspond to
20ms
              CCPR1L = (40000 - angle);
              CCP1CON = 0x08;        // compare mode, set output on match
              H_L = 1;               //next time High
              TMR1H = 0;
              TMR1L = 0;
        }

        PIR1 = PIR1&0xFB; }else{
```

```
            PIR1 = PIR1&0xFB;
          }
      }

}




void ATD_init_A0(){
ADCON0 = 0x41; // ATD ON, Dont go, channel 0, fosc/16
ADCON1 = 0xCE; // All channels are digital except A0 , 500 khz , right justified
}


unsigned int ATD_read_A0(){
ADCON0 = ADCON0 | 0x04; // GO
while(ADCON0 & 0x04);
return ((ADRESH<<8) | ADRESL);
}


void mymsDelay(int const x){
```

```
    tick=0;
    while(tick<x);
}

void check_front_right(){
// read port D0 : right sensor
if(!(PORTD & 0b00000001)){
  tick1 = 0;
  // read port D2 : front sensor
  while((PORTD & 0b00000100)){
  // if it turns more than the turning_th stop (turning_th is time)
  if (tick1 >= 4000) break;
  // move right
  move_right();


  }
  mymsDelay(100);
// stop moving
stop_moving();
}
}
```

```
void check_front_left(){
// read port D1 : left sensor
if (!(PORTD & 0b00000010)){
    tick1 = 0;
   // read port D2 : front sensor
   while((PORTD & 0b00000100)){
     // move left
     if (tick1 >= 4000) break;
     move_left();
     }
// stop moving
stop_moving();
}
}


void check_front(){
while(!(PORTD & 0b00000100)){
sensor_voltage = ATD_read_A0();


while(sensor_voltage <= 70 || sensor_voltage >= 100){
  sensor_voltage = ATD_read_A0();
move_forward();
distance=dist();
```

```
if(distance<20){

move_right();

mymsDelay(2000);

move_forward();

mymsDelay(2000);

move_left();

mymsDelay(2000);

move_forward();

}else{move_forward();}

 if(PORTD & 0b00000100){break;}

}

if(sensor_voltage <= 100) {break;}

if(PORTD & 0b00000100){break;} }


 mymsDelay(100);

 stop_moving();

}





void stop_moving(){

   PORTB = PORTB & 0b11110000;

}
```

```
void move_left(){

    Speed(90);
    PORTB =(PORTB & 0b11110000)| 0b00001001;
}

void move_right(){
   Speed(90);
   PORTB = (PORTB & 0b11110000) | 0b00000110;
}

void move_forward(){
   Speed(140);
   PORTB = (PORTB & 0b11110000)| 0b00000101;

}

void move_backwards(){
   Speed(90);
   PORTB = (PORTB & 0b11110000)| 0b00001010;
}
```

```
void adjust_position(){
while(!(PORTD & 0b00000100)){
    sensor_voltage = ATD_read_A0();
    while(sensor_voltage < 70){
    move_backwards();
    sensor_voltage = ATD_read_A0();
}

if( sensor_voltage>=70) {break;}
if(PORTD & 0b00000100){break;}
}
mymsDelay(100);
stop_moving();
}




void ENDD(){


sensor_voltage = ATD_read_A0();
```

```
while (!(PORTD & 0b00000100))
{
    if(sensor_voltage <70 || sensor_voltage > 100) break;
    stop_moving();
    a=1;
    angle = 1000;
    PORTD=PORTD | 0B11000000;
    mymsDelay(2000);
    angle = 3500;
    PORTD=PORTD & 0B00111111;
    mymsDelay(2000);
    angle = 1000;
    PORTD=PORTD | 0B11000000;
    mymsDelay(2000);
    angle = 3500;
    PORTD=PORTD & 0B00111111;
    mymsDelay(2000);
    angle = 1000;
    PORTD=PORTD | 0B11000000;
}
PORTD=PORTD & 0B00111111;
angle =  2250;
a=0;
```

```
}

int dist(){
  int d = 0;
  T1CON = 0x10; // Use internal clock, no prescaler
  mymsDelay(200);

  T1CON = 0x10;
  TMR1H = 0;            // Reset Timer1
  TMR1L = 0;

  PORTC = PORTC | 0b01000000; // Trigger HIGH
  delay_us(10);         // 10 µs delay
  PORTC = PORTC & 0b10111111; // Trigger LOW

  while (!(PORTC & 0b10000000));



  T1CON = T1CON | 0b00000001; // Start Timer



  while (PORTC & 0b10000000);
```

```
    T1CON = T1CON & 0b11111110; // Stop Timer

    d = (TMR1L | (TMR1H << 8)); // Read Timer1 value
    d = d / 58.82;         // Convert time to distance (cm)

    mymsDelay(10);
    T1CON = 0x01;
    return d;
}
```

Mechanical design

1. Power Supply System

The power supply system supplies power to the car's components. It provides the regulated voltage levels required by the PIC16F877A microcontroller, sensors, and motors. It consists of 3 batteries, 3.7v each.

2. Microcontroller: PIC16F877A

The PIC16F877A microcontroller is the central processing unit of the system. It processes data from sensors, executes control algorithms, and generates signals for motor control. The GPIO pins of the microcontroller are configured as follows:

Digital Inputs: Ultrasonic sensor's Echo signal and start button.

Analog Input: LDRs for measuring light intensity by using ADC pins.

PWM Outputs: Speed control of motors using the H-Bridge driver.

3. Light Sensors: LDRs

three LDRs attached with analog pins of the microcontroller are used for detecting the direction of the strongest light. Voltage output from each of the LDRs is proportional to the light intensity it gets. The microcontroller processes the resultants and selects a direction to move based on which LDR had a greater amount of light. The LDR that gets the highest amount of light, lights up.

4. Ultrasonic Sensor

Distance Calculation: It employs an ultrasonic sensor that detects the distance to the obstacle.

5. Motor Control

Four DC motors drive the wheels of the car, and are controlled by:
H-Bridge Motor Driver: The driver enables both the forward and backward motion through which the speed and direction of the motors are controlled.

There are 2 DC motors connected together on each side, therefore, each two work together accordingly.

6. Servo Motor

A servo motor, connected to RC2, changes the orientation of the car to make precise navigation. The position of the servo is determined by a PWM signal provided by the microcontroller.

7. LEDs and Indicators

Two LEDs, RD6 and RD7, are used for indication purposes, showing the system's status. They can give indications when the car is on track, to avoid hurdles, or at rest.

8. User Interface: Start Button

There is a start button, RD3, which starts the operation of the car. The microcontroller monitors the press of the button and switches the car between an active and idle state

9. System Interconnections

Sensor Integration: The inputs from LDRs and the ultrasonic sensor are fed to the microcontroller, which does real-time processing.

Motor Driver Control: Digital and PWM signals from the microcontroller drive the H-Bridge to actuate the motors.

Power Distribution: Proper wiring ensures that all components have sufficient and stable power.

Problems & Recommendations

- ### Inconsistent Ambient Lighting Detection

  Problem: The LDRs could give spurious readings in changing ambient light conditions, for example, outdoors where the sun changes position or shade, or indoors where shadows may be present. Recommendation: Employ higher end light sensors such as photodiodes or phototransistors that have filters that can block interfering ambient light. Alternatively, use an algorithm that dynamically adjusts the LDR threshold levels.

- ### Obstacle Detection Blind Spots

  Problem: The ultrasonic sensor is unable to detect obstacles at certain angles or when the surface material is not a good reflector of ultrasonic waves, such as soft or angled surfaces. Recommendation: Add more ultrasonic sensors or IR sensors to increase the angles for higher accuracy of obstacle detection.

Conclusion

The light-following car with obstacle avoidance demonstrates the practical application of embedded systems in autonomous navigation. By integrating LDR sensors, an ultrasonic sensor, and a PIC16F877A microcontroller, the project achieves a cost-effective yet efficient design capable of real-time decision-making. This project serves as a stepping stone for more advanced systems, showcasing how sensor fusion and programming can solve real-world challenges. It has potential applications in robotics, automation, and smart mobility solutions, providing a foundation for future exploration in autonomous vehicle technologies. The success of this project highlights the importance of interdisciplinary collaboration between hardware, software, and mechanical design, proving that embedded systems are a cornerstone for innovation in modern engineering