Light following Robot using LDR Sensor and EAB

Introduction

A Robot is always a fascinating object developed by engineers. A robot, in general, is an electro-mechanical device that can perform tasks automatically or semi-automatically. It is not necessary that it must be humanoid in appearance. A robot requires some degree of guidance, this can be provided either by using a remote control or with a computer interface.

A Light Following Robot is also an electro-mechanical device (robot) but with an added intelligence that it follows the route or decides the path to follow according to the light that falls on it. In this project we have designed this robot such that the amount of light falling on it, from every direction, will be calculated and then it will find out the direction from which the light with maximum intensity falls on it. Then it moves in that direction only.

LDR sensor

It is observed that the LDR's (Light Dependent Resistor) resistance is inversely proportional to the intensity of light.

When there is more amount of light the resistance of LDR is less and the resistance is more when intensity of light decreases.
Motor Driver Board

The motor driver board consists of ICL293D. It has 4 inputs and 4 outputs. Since, the current from the EAB is not sufficient enough to drive the motors, so we have to use this Driver to increase the current by which we can drive the DC Motors.

Components

The Components required for building the Light Follower application are:

- Embedded Application Board
- Motor Driver Board (containing L293D)
- LDR Sensor Card (4 nos)
- DC Motors (2 nos)
- Chassis
- Connectors (Jumpers)
- 9V battery
- Wheels
- Light Source
Application Notes

Block Diagram
Block level representation of the different components of the light follower application.

---

Schematic Diagram
The Schematic diagram illustrates the circuit connections for designing the application.
Connection Description

Here, the sensor board consists of 4-pairs of LDR and Resistor. An LDR is a light sensor. Whenever light falls on it, its resistance value decreases depending on the intensity of that light. So, in this circuit we have connected the LDRs with resistor in series. One end of LDR is connected to the Vcc while the free end of resistor is connected to GND. The output of this combination is taken from the junction of LDR and resistor. In this board each pair is performing just as a voltage divider network. The output is in analog form.

The above four analog signals are now given to four analog pins of the Microcontroller board. The microcontroller has an in-built ADC, where all these channels' analog signal will be converted to Digital signal (Digital value). Now the maximum value will be found out and according to this result the Digital output pins of this board (i.e., EAB) will be either in HIGH or LOW state.

Note: Any GPIO pin can be used for this project. Set the particular GPIO pins to output and provide a high /low signal to the pin.
Application Notes

**Code Flow Chart**

```
Start

1. Define System clock
2. Library Header File Declaration

Declare local variables
Set Oscillator Frequency

1. Configure UART2
2. Configure I/O pins
3. Configure ADC(4 channels)

While(1)

1. Read Sensor_0 and Store to D0
2. Read Sensor_1 and Store to D1
3. Read Sensor_2 and Store to D2
4. Read Sensor_3 and Store to D3

1. Check with Lower offset
2. Find the max value of (D0,D1,D2,D3)
3. Check with Higher offset
4. Decide Direction

x = Direction

Switch (x)

0 1 2 3 4
Move Forward Move Left Move Right Move Backward Pause

Copyright: Brigosha reserves all rights in the event of industrial property rights. We reserve all rights of disposal such as copying and passing on to third parties. Brigosha is the registered trademark of Brigosha Technologies Pvt. Ltd. Specifications within this document are preliminary and subject to change without notification. Document is not intended for publication.
Source Code

The Source code shown below is the firmware to be flashed in the microcontroller of the Embedded Application Board. The Source code is commented for better understanding of the user.

```
#define SYS_CLK 8000000  //Required for delay macro functions
//Default 1MHZ, else change as per configuration

/*** INCLUDE STANDARD HEADERS & LIBRARY ***/
#include <stdio.h>
#include <stdlib.h>

#include "EAB_Library.h"

/*** GLOBAL VARIABLES ****/

/***************************************************************************/

void main(void)
{
    /*-**************************************************************************/
    
    short DO=0,DI=0,DE=0,DS=0,DS5=0, x=0, MSB=0, LSB=0;

    /*-**************************************************************************/
    
    Oscillator.SetFreq_8MHZ(); // Select system clock at 8 MHz

    Serial2.Open(9600); // Open serial port with 9600 baud rate

    PinDigitalOut(RCO);    // RCO as digital output
    PinDigitalOut(RA5);    // RA5 as digital output
    PinDigitalOut(RB4);    // RB4 as digital output
    PinDigitalOut(SDO2);   // SDO2 as digital output
    PinDigitalOut(RCS);    // RCS as digital output

    /*-**************************************************************************/
    
    PinAnalogIn(AN0);      // AN0 as digital input
    PinAnalogIn(AN1);      // AN1 as digital input
    PinAnalogIn(AN2);      // AN2 as digital input
    PinAnalogIn(AN3);      // AN3 as digital input

    ADC.Open(); // Open ADC

    /***************************************************************************/
    
    /***************************************************************************/
```
```c
/*** PLACE THE REPETITIVE TASKS IN THIS LOOP ***/
while(1)
{
    PIR1bits.ADIF=0;           // Clear ADC Flag
    ADC.SetChannel(CHANNEL_AN0); // Select Channel 0
    while( PIR1bits.ADIF==0);
    D0 = ADC.ReadData();        // Read ADC Data
    MSB = D0>>8;
    LSB = D0;
    D0 = D0>>2;
    Serial2.SendByte('A');      // Send 'A' via UART
    Serial2.SendByte(MSB);      // Send higher byte of ADC value via UART
    Serial2.SendByte(LSB);      // Send lower byte of ADC value via UART

    PIR1bits.ADIF=0;           // Clear ADC Flag
    ADC.SetChannel(CHANNEL_AN1); // Select Channel 1
    while( PIR1bits.ADIF==0);
    D1 = ADC.ReadData();        // Read ADC Data
    MSB = D1>>8;
    LSB = D1;
    D1 = D1>>2;
    Serial2.SendByte('B');      // Send 'B' via UART
    Serial2.SendByte(MSB);      // Send higher byte of ADC value via UART
    Serial2.SendByte(LSB);      // Send lower byte of ADC value via UART

    PIR1bits.ADIF=0;           // Clear ADC Flag
    ADC.SetChannel(CHANNEL_AN2); // Select Channel 2
    while( PIR1bits.ADIF==0);
    D2 = ADC.ReadData();        // Read ADC Data
    MSB = D2>>8;
    LSB = D2;
    D2 = D2>>2;
    Serial2.SendByte('C');      // Send 'C' via UART
    Serial2.SendByte(MSB);      // Send higher byte of ADC value via UART
    Serial2.SendByte(LSB);      // Send lower byte of ADC value via UART
}```
PIR1bits.ADIF=0;       // Clear ADC Flag
ADC.SetChannel(CHANNEL_AN3);    // Select Channel 3
while( PIR1bits.ADIF==0);          
D3 = ADC.ReadData();            // Read ADC Data
MSB = D3>>8;
LSB = D3;
D3 = D3>>2;
Serial2.SendByte('D');          // Send 'D' via UART
Serial2.SendByte(MSB);          // Send higher byte of ADC value via UART
Serial2.SendByte(LSB);          // Send lower byte of ADC value via UART

D5 = 150;                        // offset
x = 4;                           // initialize direction index value as 4

/* To find the MAX value */

if(D0>D5)
{
    D5=D0;
    x = 0;
}
else
{
    D5 = D5;
}

if(D1>D5)
{
    D5=D1;
    x = 1;
}
else
{
    D5 = D5;
}
if(D2>D5)
{
    D5=D2;
    x = 2;
}
else
{
    D5 = D5;
}

if(D3>D5)
{
    D5=D3;
    x = 3;
}
else
{
    D5 = D5;
}

Serial2.WriteByte('M'); // Send 'M' via UART
Serial2.WriteByte(x>>8); // Send higher byte of direction index
Serial2.WriteByte(x); // Send lower byte of direction index

if(D5>230)
{
    x = 4;
}

switch(x)
{
    case 0:
    {
        PinWrite.RC0 = 1; // Set RC0 output High
        PinWrite.RA5 = 0; // Set RA5 output Low
        PinWrite.RB4 = 1; // Set RB4 output High
        PinWrite.RC5 = 0; // Set RC5 output Low
    }
break;
}
case 1:
{
PinWrite.RC0 = 0; //Set RC0 output Low
PinWrite.RA5 = 1; //Set RA5 output High
PinWrite.RB4 = 1; //Set RB4 output High
PinWrite.RC5 = 0; //Set RC5 output Low
break;
}
case 2:
{
PinWrite.RC0 = 1; //Set RC0 output High
PinWrite.RA5 = 0; //Set RA5 output Low
PinWrite.RB4 = 0; //Set RB4 output Low
PinWrite.RC5 = 1; //Set RC5 output High
break;
}
case 3:
{
PinWrite.RC0 = 0; //Set RC0 output Low
PinWrite.RA5 = 1; //Set RA5 output High
PinWrite.RB4 = 0; //Set RB4 output Low
PinWrite.RC5 = 1; //Set RC5 output High
break;
}
case 4:
{
PinWrite.RC0 = 0; //Set RC0 output Low
PinWrite.RA5 = 0; //Set RA5 output Low
PinWrite.RB4 = 0; //Set RB4 output Low
PinWrite.RC5 = 0; //Set RC5 output Low
break;
}
}
How to Operate

This is a simple plug-and-play type robot. Follow the steps mentioned below in order to operate the project...

- You have to power the EAB, Sensor Board and Motor Driver Circuit with 9V DC. Carefully check the polarities and then connect them.
- Place the robot in a suitable arena.
- Be ready with a light source e.g. Good quality torch.
- Switch on the EAB.

The Robot is in action now...!!!

The Robot will exactly follow the path in the direction from which maximum amount of light falls on it.

More Projects

Various other applications can be built using LDR Sensor and Embedded Application Board.

Some such applications are given below:

- Anti-collision system
- Alarm systems
- Animation
- Automatic Street Light Intensity Controller