Introduction to embedded systems (CSE211s)

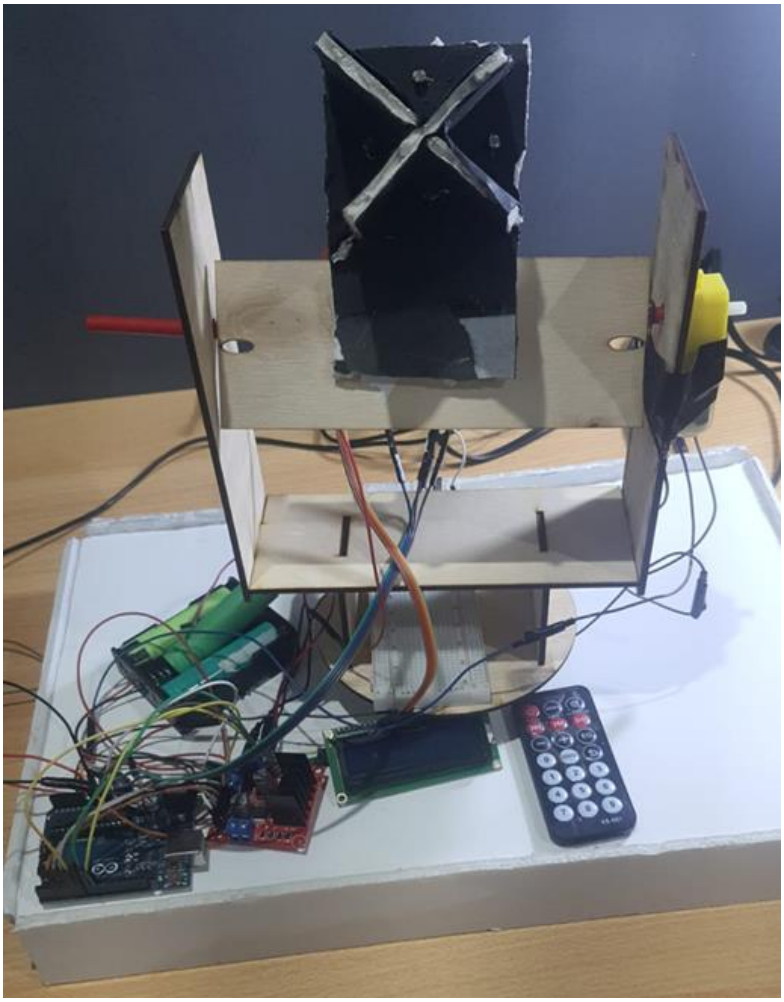
ARDUINO SOLAR TRACKER

TEAM MEMBERS

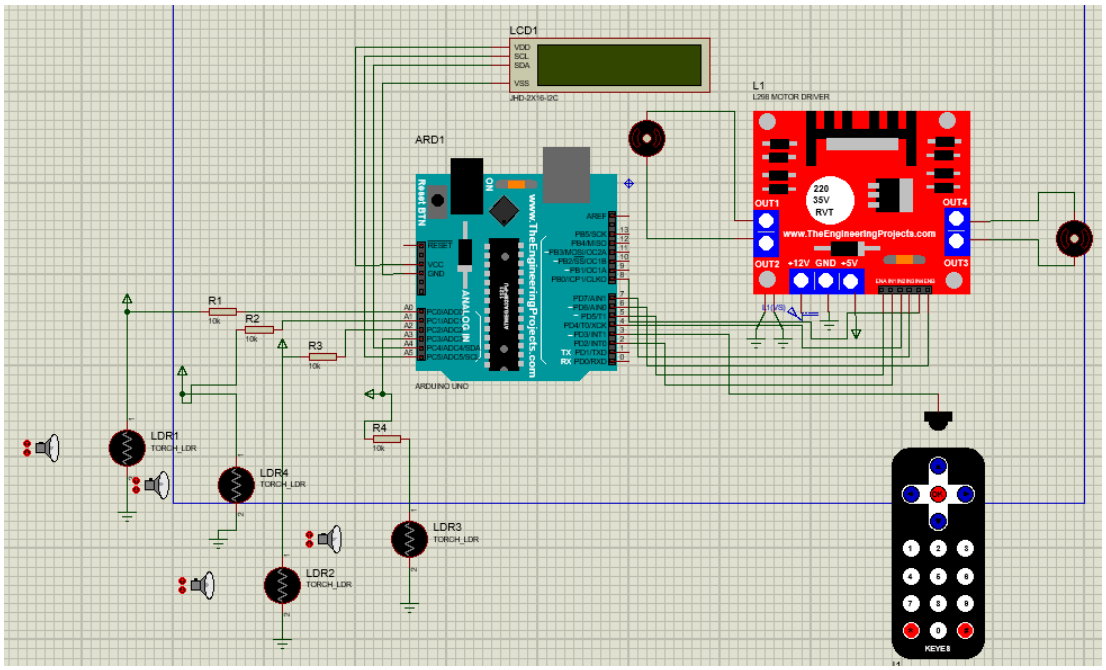
NAME	CODE
Abdullah Mohamed Ibrahim	2001826
Jana Maher Fouad	2000053
Nourhan ElSayed Fouad	2101639
Areej Ahmed Medhat	2000136
Nihal Osama Abo-ElWafa	2001314
Abdelrahman Ali Mostafa	2000312
Habiba Mohamed Magdi	2000015
Rawan Khalid Mohamed	2000248
Abdelrahman Ahmed Mohamed	2001142
Abdelrahman Mohamed Nour	2000412

PROJECT DESCRIPTION

This project presents an open hardware/software test bench for solar tracker. The proposed prototype is based on a dual-axis solar tracker controlled with Arduino Uno which is an open-source platform based on easy-to-use hardware and software. The solar tracker can be controlled automatically with the help of Light Dependent Resistor (LDR) sensors. The hardware used has been chosen to be inexpensive, compact and versatile, as shown, we've used wood frame made by laser cutting. The proposed test bench is designed to destiny to the position where the light is coming from, this is achieved by the use of 2 gear motors, Arduino uno, motor driver L298N, 3 rechargeable batteries each of 3.7V. the manual part is controlled by a remote, and the automatic part is adjusted by PID.



SCHEMATIC CIRCUIT



CODE

```
#include <Wire.h>
#include <IRremote.hpp>
#include <LiquidCrystal_I2C.h>

IRrecv IR(3);
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for 16 chars and 2 line display

#define LDR_U A0  //up ldr
#define LDR_D A1  //down ldr
#define LDR_R A2  //right ldr
#define LDR_L A3  //left ldr

#define hor_1 4 // in1 for horizontal motor
#define hor_2 2 //in2 for horizontal motor
#define e_hor 5 // horizontal motor "speed"

#define ver_1 7 //in1 for vertical motor
#define ver_2 8 //in2 for vertical motor
#define e_ver 6 //enable vertical, must be pwm "speed"

void setup()
{
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("SOLAR TRACKER");
  lcd.setCursor(0, 1);
```

```
28    lcd.print("PROJECT");
29    IR.enableIRIn();
30    Serial.begin(9600);
31    pinMode(e_hor, OUTPUT); pinMode(hor_2, OUTPUT); pinMode(hor_1, OUTPUT);//horizontl motor
32    pinMode(e_ver, OUTPUT); pinMode(ver_1, OUTPUT); pinMode(ver_2, OUTPUT);//vertical motor
33    pinMode(LDR_U, INPUT); pinMode(LDR_L, INPUT); pinMode(LDR_R, INPUT); pinMode(LDR_D, INPUT);
34  }
35  void loop()
36  {
37    if ( IR.decode())
38    {
39      Serial.println(IR.decodedIRData.decodedRawData, HEX);
40
////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////
41      if (IR.decodedIRData.decodedRawData == 0xB847FF00)
42      {
43        analogWrite(e_hor, 90); digitalWrite(hor_1, 1); digitalWrite(hor_2, 0);
44        analogWrite(e_ver, 0); digitalWrite(ver_1, 0); digitalWrite(ver_2, 0);
45        lcd.clear();
46        lcd.setCursor(0, 0);
47        lcd.print("MOVING RIGHT");
48      }
49
//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&//
50      else if (IR.decodedIRData.decodedRawData == 0xBC43FF00)
51      {
52        analogWrite(e_hor, 90); digitalWrite(hor_1, 0); digitalWrite(hor_2, 1);
53        analogWrite(e_ver, 0); digitalWrite(ver_1, 0); digitalWrite(ver_2, 0);
54        lcd.clear();
55        lcd.setCursor(0, 0);
56        lcd.print("MOVING LEFT");
57      }
58
//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&//
59      else if (IR.decodedIRData.decodedRawData == 0xF609FF00)
60      {
61        analogWrite(e_hor, 0); digitalWrite(hor_1, 0); digitalWrite(hor_2, 0);
62        analogWrite(e_ver, 80); digitalWrite(ver_1, 1); digitalWrite(ver_2, 0);
63        lcd.clear();
64        lcd.setCursor(0, 0);
65        lcd.print("MOVING DOQN");
66      }
67
//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&//
68      else if (IR.decodedIRData.decodedRawData == 0xF20DFF00)
69      {
70        analogWrite(e_hor, 0); digitalWrite(hor_1, 0); digitalWrite(hor_2, 0);
71        analogWrite(e_ver, 80); digitalWrite(ver_1, 0); digitalWrite(ver_2, 1);
72        lcd.clear();
73        lcd.setCursor(0, 0);
74        lcd.print("MOVING UP");
75      }
76
//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&//
77      else
78      {
79        analogWrite(e_hor, 0); digitalWrite(hor_1, 0); digitalWrite(hor_2, 0);
```

```
        analogWrite(e_ver, 0); digitalWrite(ver_1, 0); digitalWrite(ver_2, 0);
        lcd.setCursor(0, 1);
        lcd.print("'last motion'");
      }
      IR.resume();

////////////////////////////////////////////////////////////////////////////////////////////////
//////////////
    }
    //else if (IR.decodedIRData.decodedRawData == 0xBA45FF00)
    //{
    float up = analogRead(LDR_U);
    float down = analogRead(LDR_D);
    float left = analogRead(LDR_L);
    float right = analogRead(LDR_R);
    // PID parameters
    float Kph = 0.6; float Kih = 0; float Kdh = 0.009;
    float Kpv = 0.5; float Kiv = 0.01;  float Kdv = 0.006;
    float integral_V = 0; float integral_H = 0;
    float derivative_V = 0; float derivative_H = 0;
    float last_error_V = 0; float last_error_H = 0;
    int Error_V = up - down; int Error_H = left - right;
    integral_V += Error_V; derivative_V = Error_V - last_error_V;
    integral_H += Error_H; derivative_H = Error_H - last_error_H;
    double output_V = Kpv * Error_V + Kiv * integral_V + Kdv * derivative_V;
    last_error_V = Error_V;
    double output_H = Kph * Error_H + Kih * integral_H + Kdh * derivative_H;
    last_error_H = Error_H;
    Serial.print("H= ");
    //Serial.println(output_H);
    //Serial.print("V= ");
    //Serial.println(output_V);
    //Serial.println(left);
    //Serial.println(right);
    //Serial.println(down);
    //Vertical motion
    if (output_V > 0)
    {
      analogWrite(e_hor, 0); digitalWrite(hor_1, 0); digitalWrite(hor_2, 0);
      analogWrite(e_ver, abs(output_V)); digitalWrite(ver_1, 0); digitalWrite(ver_2, 1);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("MOVING UP");
    }

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&//
    else if (output_V < 0)
    {
      analogWrite(e_hor, 0); digitalWrite(hor_1, 0); digitalWrite(hor_2, 0);
      analogWrite(e_ver, abs(output_V)); digitalWrite(ver_1, 1); digitalWrite(ver_2, 0);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("MOVING DOWN");
    }

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&//
    //Horizontal motion
    if (output_H < 0)
    {
      analogWrite(e_hor, abs(output_H)); digitalWrite(hor_1, 1); digitalWrite(hor_2, 0);
```

```
      analogWrite(e_ver, 0); digitalWrite(ver_1, 0); digitalWrite(ver_2, 0);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("MOVING RIGHT");
    }

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&//
    else if (output_H > 0)
    {
      analogWrite(e_hor, abs(output_H)); digitalWrite(hor_1, 0); digitalWrite(hor_2, 1);
      analogWrite(e_ver, 0); digitalWrite(ver_1, 0); digitalWrite(ver_2, 0);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("MOVING LEFT");
    }

//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&//
    else

    {
      analogWrite(e_hor, 0); digitalWrite(hor_1, 0); digitalWrite(hor_2, 0);
      analogWrite(e_ver, 0); digitalWrite(ver_1, 0); digitalWrite(ver_2, 0);
      lcd.clear();
      //lcd.setCursor(0, 1);
      //lcd.print("'last motion'");
    }
  }
```

EMBEDDED SOFTWARE DESIGN

The Arduino “microcontroller” is the main brain for all the actions.

The motor driver which include 2 H-Bridges will control the speed of the motors as it contains 6 pins :

For the horizontal motor there are 3 pins:

1,2 are for controlling the direction of rotation and the other pin is ENABLE for controlling speed of rotation

For the vertical motor there are 3 pins:

3,4 are for controlling the direction of rotation and the other pin is ENABLE for controlling speed of rotation

In manual mode:

This mode depends on the serial sent to the receiver by the remote-control which is in hexadecimal form,

When the read is equal to 0xB847FF00:

The horizontal motor will move with speed 90, input 1 will turn on, input 2 will turn off. On the other side all the pins of the vertical motor are turned off. Therefore, the horizontal motor will move in the clockwise direction and the LCD will print "MOVING RIGHT".

When the read is equal to 0xBC43FF00:

The horizontal motor will move with speed 90, input 1 will turn off, input 2 will turn on. On the other side all the pins of the vertical motor are turned off. Therefore, the horizontal motor will move in the anti-clockwise direction and the LCD will print "MOVING LEFT".

When the read is equal to 0xF609FF00:

The vertical motor will move with speed 80, input 3 will turn on, input 4 will turn off. On the other side all the pins of the horizontal motor are turned off. Therefore, the horizontal motor will move in the clock wise direction and the LCD will print "MOVING UP".

When the read is equal to 0xF20DFF00:

The vertical motor will move with speed 80, input 3 will turn off, input 4 will turn on. On the other side all the pins of the horizontal motor are turned off. Therefore, the horizontal motor will move in the anti-clockwise direction and the LCD will print "MOVING DOWN".

If no serial sent at last:

The pins of the two motors are turned off and the LCD will print: its last step + "last motion"

In automatic mode: This mode depends on the LDRs as their resistance decrease when light intensity increase, and by **PID controller:** which is the ability to use the three control terms of proportional, integral and derivative influence on the controller output to apply accurate and optimal control.

The equations for the proportional, integral, and derivative terms for vertical motor are as follows:

Vertical error = up- down

Proportional Output = Kpv * (Vertical error)

Integral of Error = summation of vertical errors

Integral Output = Kiv * Integral of Error

Derivative of Error = vertical error – last error

Where last error is the error in the previous loop

Derivative Output = Kdv * Derivative of Error

The PID output is the sum of the proportional, integral, and derivative terms:

PID Output for vertical = Proportional Output + Integral Output + Derivative Output

And in some case the output may be in negative so we put it in absolute value.

The equations for the proportional, integral, and derivative terms for horizontal motor are as follows:

Horizontal error = left - right

Proportional Output = Kph * (Horizontal error)

Integral of Error = summation of horizontal errors

Integral Output = Kih * Integral of Error

Derivative of Error = horizontal error – last error

Where last error is the error in the previous loop

Derivative Output = Kdh* Derivative of Error

The PID output is the sum of the proportional, integral, and derivative terms:

PID Output for horizontal = Proportional Output + Integral Output + Derivative Output

And in some case the output may be in negative so we put it in absolute value.

When the light is focused on the upper LDR more than the lower one :

The vertical motor will move with speed Output for vertical , input 3 will turn on, input 4 will turn off. On the other side all the pins of the horizontal motor are turned off.

Therefore, the horizontal motor will move in the clock wise direction and the LCD will print "MOVING UP".

When the light is focused on the lower LDR more than the uper one:

The vertical motor will move with speed Output for vertical, input 3 will turn off, input 4 will turn on. On the other side all the pins of the horizontal motor are turned off. Therefore, the horizontal motor will move in the anti-clockwise direction and the LCD will print "MOVING DOWN".

When the light is focused on the right LDR more than the left one:

The horizontal motor will move with speed Output for horizontal, input 1 will turn on, input 2 will turn off. On the other side all the pins of the vertical motor are turned off. Therefore, the horizontal motor will move in the clockwise direction and the LCD will print "MOVING RIGHT".

When the light is focused on the left LDR more than the right one:

The horizontal motor will move with speed Output for horizontal, input 1 will turn off, input 2 will turn on. On the other side all the pins of the vertical motor are turned off. Therefore, the horizontal motor will move in the anti-clockwise direction and the LCD will print "MOVING LEFT".

If no more light is focused at last:

The pins of the two motors are turned off and the LCD will print: its last step + "last motion"

COMMENT:

There is no option for two LDRs to have the same reading as they are very sensitive for light, therefore it’s very hard for the motors speed to reach zero as the error in this case will be very small and in reality we can see that the motors make noise trying to move but actually they can’t.

LIST OF COMPONENTS

COMPONENTS	FUNCTION
Arduino Uno	Microcontroller board that can be integrated into a variety of electronic projects, it controls LEDs and motors as output.
L298N motor driver	Dual H-bridge motor driver that controls direction and speed of two DC motors at the same time.
4 LDRs	Detect light levels.
3 lithium batteries	Flows the electrical from the positive current collector through the device being powered to the negative current collector.
Battery holder	Holds the batteries in place safely and securely.
2 DC motors	Take electrical power through direct current and convert it into mechanical rotation.
LCD I²C	A controller chip handling I²C communications and an adjustable potentiometer for changing the intensity off the LED backlight.
Receiver	A circuit that accepts signals from a transmission medium and decodes or translates them into a form that can drive local circuits.
Resistances	Electrical component that limits or regulates the flow of electrical current in an electronic circuit.
Bread board	Used for building temporary circuits.
Remote control	Electronic device used to operate another device from a distance, usually wirelessly.
Jumper wires	Connect two points in a circuit without soldering.

APPS AND PLATFORMS

Arduino IDE, proteus.