Arduino Solar Tracker

Open hardware/software test bench for solar tracker with virtual instrumentation.

Components and supplies

4→Resistor 330 ohm
4 →LDR, 5 Mohm
1→ Arduino UNO
1→Mini Solar Panel
2→SG90 Micro-servo motor
1→Rotary potentiometer (generic)
2→Pushbutton Switch, Pushbutton

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 prototyping platform based on easy-to-use hardware and software. The solar tracker can be controlled automatically with the help of LightDependent Resistor (LDR) sensors or manually using a potentiometer. Moreover, this test bench provides virtual instrumentation based on Excel in which its solar tracker data can be recorded and presented. The hardware used has been chosen to be inexpensive, compact and versatile. The proposed test bench is designed to help students develop their understanding of control theory and their application.

The proposed test bench is presented in Fig. 1. It is based on a solar tracker that can rotate automatically to track the sun with the help of four LDR sensors and two servomotors (SM1 and SM2), or manually using a potentiometer. To switch between the two modes (automatic and manual), a push-button is used. Another push-button is used to link either the SM1(up-down servomotor) or SM2 (left-right servomotor) to the potentiometer to control their movement. Moreover, a computer is used as a virtual instrument to visualize the mode and current, voltage and power of the PV panel according to time in MS Excel. Arduino Uno board is utilized to implement all software requirements of the system.

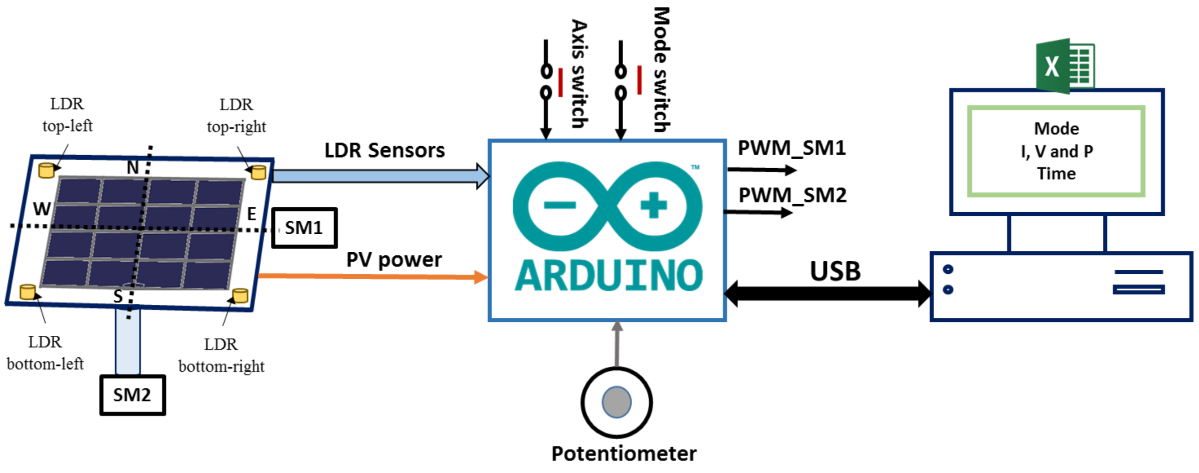


Fig. 1. Schematic of the proposed test bench

Mechanical design

As shown in Fig. 2, the computer-aided design (CAD) 3D model of the solar tracker is designed in CATIA. It is composed of the PV panel, the left-right and up-down servomotors, and four LDR sensors. For the horizontal axis, a bearing is fixed in parallel with the up-down servomotor for better flexibility. The solar tracker is designed to have two degrees of freedom, from east to west by the left-right servomotor and from south to north by the up-down servomotor. The LDR sensors are placed in the four corners of the PV panel and are put in dark tubes with a small hole on the top to detect the illumination of the sun. These dark tubes are also considered a concentrator of radiation and are used to increase the solar tracker robustness.

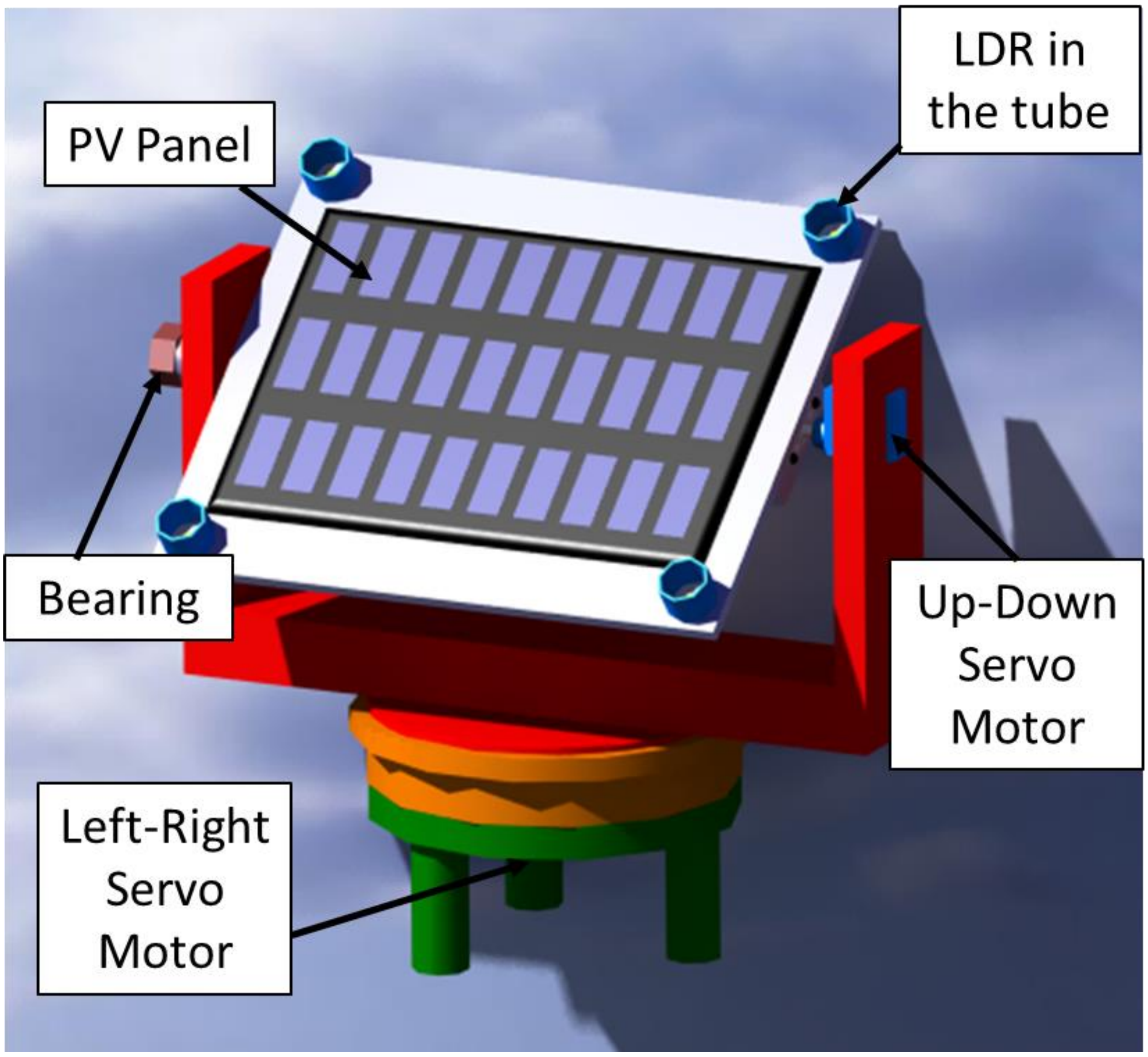


Fig. 2. CAD 3D model of the solar tracker in CATIA

Hardware system

Fig. 3 presents the electronic circuit of the proposed testbench. For automatic mode, the microcontroller converts the analogs values of LDRsensors (pins A0 to A3) into digitals. Then it controls two servomotors(up-down and left-right) using two Pulse-Width Modulation (PWM) signals (pins 5and 6) to track the sun. The rotation movements occur in two axes, in azimuth from east to west according to the daily sun's path and in elevation from south to north according to the seasonal sun's path. For manual mode, a potentiometer (pin A4) is used to control the movement of the two servo motors, a push-button (pin 11) is deployed to connect the potentiometer either to up-down servomotor or left-right servomotor. Besides, another pushbutton (pin 12) is used to switch between the two modes. Furthermore, the PV voltage is measured through the analog pin A5 of the Arduino, then the PV current is calculated since the resistor of the load is already known. Next, the PV current, voltage and power versus time and the actual mode are sent to the computer to present them in real-time on MS Excel.

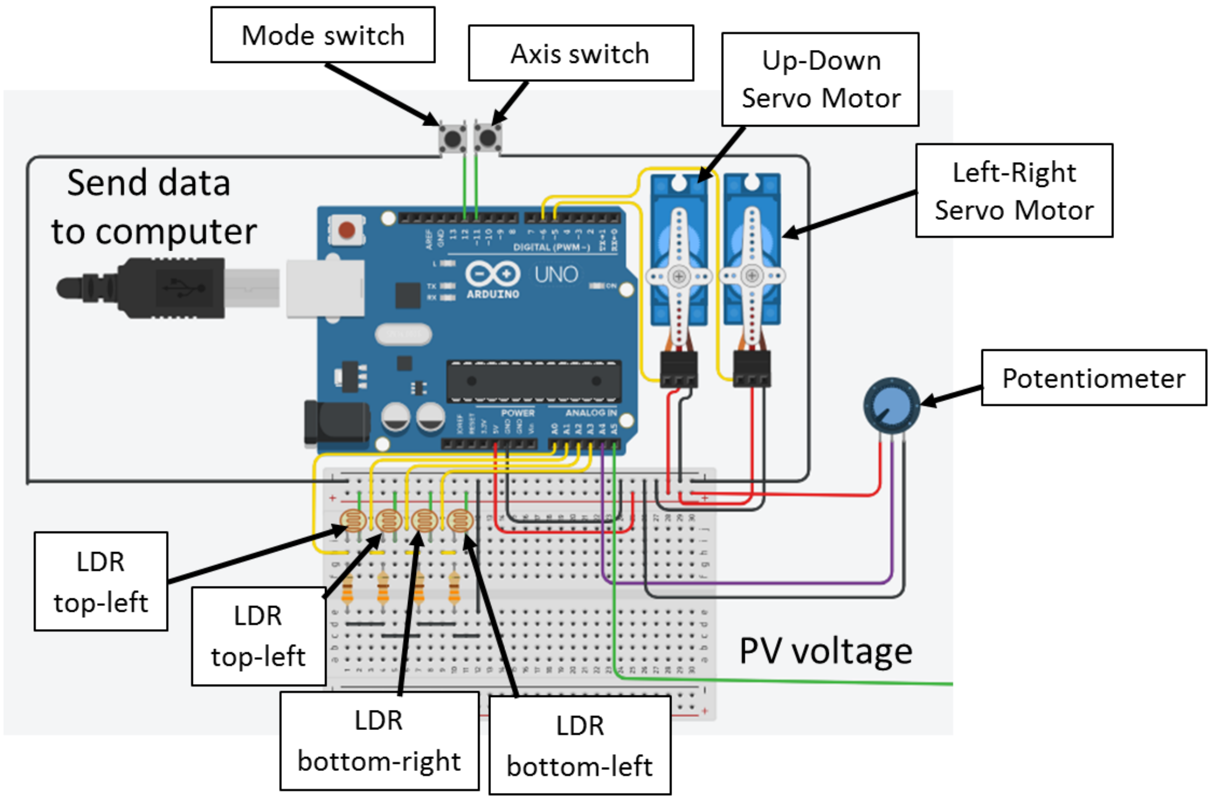


Fig. 3. Electronic circuit of the solar tracker with manual and automatic modes

The LDR sensor circuitry is designed as a voltage divider circuit. The variation in the light intensity is proportional to the variation of the divider output voltage. The top of the potential divider is 5 V, the ground is at 0 V, and the output of the voltage divider is connected to an analog input (A 0 for instance) of the microcontroller. Subsequently, the Analog to Digital Converter(ADC) of the microcontroller converts the analog value read by A 0 into a digital value between 0 and 1023 because the ADC is coded in 10 bits, and according to this value, it is possible to know the level of light. The value of resistors used in voltage dividers is 330 Ω.

Two 180 degrees servomotors are used. A servomotor (MG996R) to control the solar tracker according to the vertical axis, which is the left-right servomotor. And a micro servo motor (SG90) to control the solar tracker according to the horizontal axis, which is the up-down servomotor. The advantage of the servomotor is that we can control its stop, run, the direction of rotation and speed using a single low current wire connected directly to an output of the microcontroller without needing any drivers. The used servo motors are controlled by the Arduino UNO board via 3-wire electrical cable as shown in Fig. 3, two wires for supply and one wire for PWM to control its positions.

The embedded software design

The embedded software is the piece that will be embedded in the hardware (Arduino Uno) to control and monitor the solar tracker test bench. The embedded software is designed to cover the following requirements:

1. The test bench has two modes: manual and automatic. A pushbutton is connected to pin 12 to switch between the two modes.

2. If the manual mode is active, the potentiometer can control servomotors either from east to west for left-right motor or from south to north for the up-down motor. A push-button is connected to pin 11 to switch the potentiometer between the two motors, either it controls the left-right servomotor or up-down servo motor.

3. If the automatic mode is active, the algorithm presented in Fig. 4 will be executed. The latter uses the analog values returned by LDR sensors. For instance, considering azimuth or vertical axis, the average values from two right LDRs and two left LDRs are compared and if the left set of LDRs receives more light, the solar tracker will move in that direction through the left-right servomotor. The latter will continue to rotate until the difference result is in the range [−10, 10]. This range is used to stabilize the controller and once the solar tracker is perpendicular to the sun, no further control is made. On the other hand, if the right set of LDRs receive more light, the solar tracker moves in that direction through the left-right servomotor and will continue to rotate until the difference result is in the range [−10, 10]. The same way is used for the elevation axis. Moreover, we also determined the average radiation between the four LDR sensors and if this value is less than a little value (8: a value which has been adjusted and tested practically and is returned when the irradiation is null). That is to say, the night has come. In this case, the solar tracker must return to the sun’s rising position. For example, if the sun’s rising position can be reached by setting 0 degrees in the left-right servomotor, and 30 degrees in the up-down servomotor. This can easily be done through the C function “servox. write(angle)” provided by Arduino IDE.

4. The PV voltage acquired through the analog pin A5 must be treated and used to compute the PV current and power. Then all these data and the actual mode must be sent through a USB cable to the computer and then present them in MS Excel.

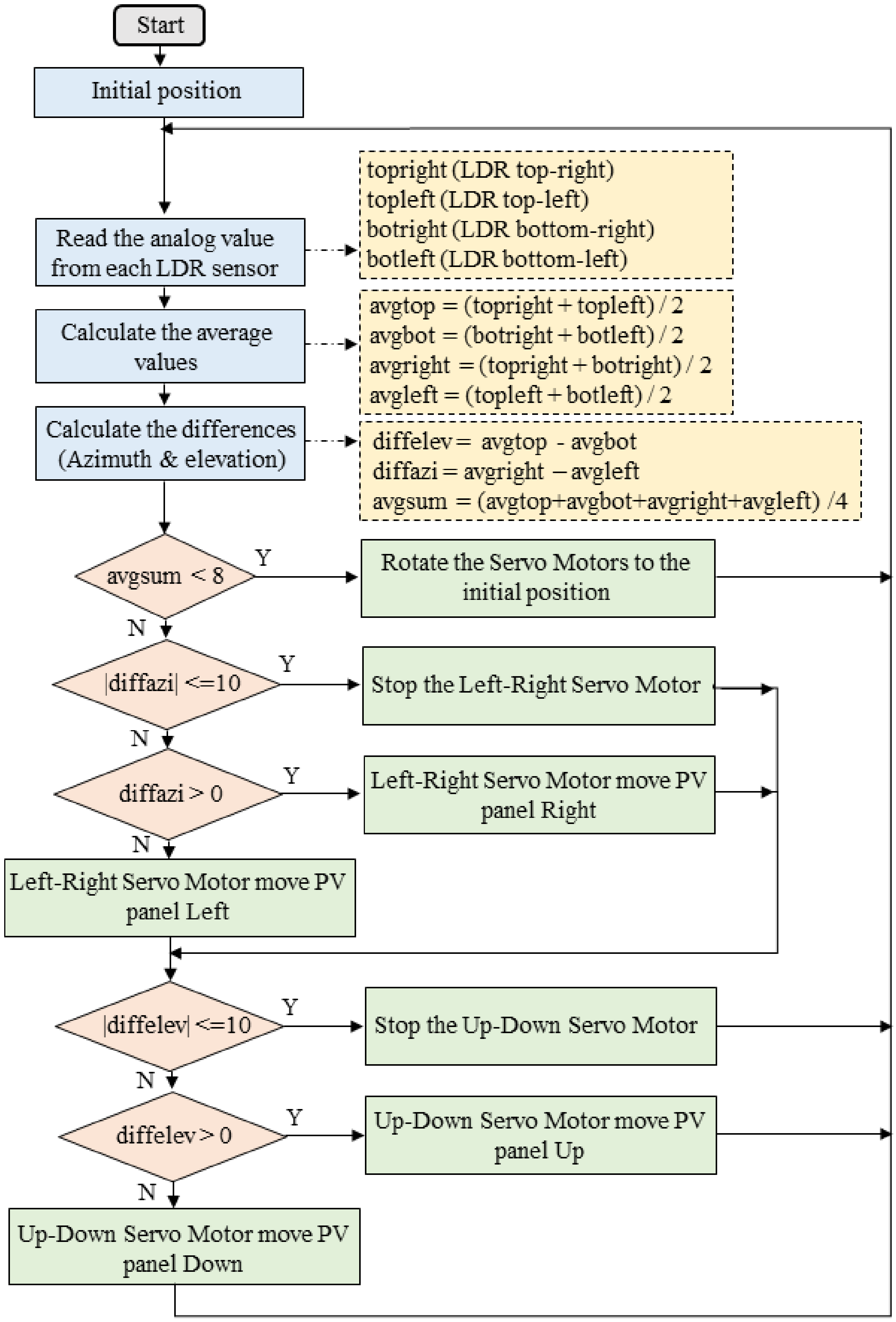
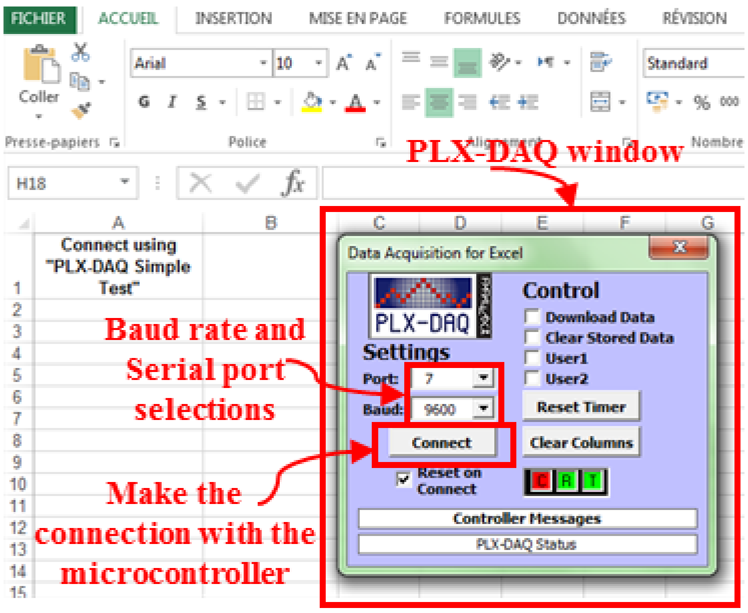

Start
Initial position
Read the analog value from each LDR sensor
topright (LDR top-right)
topleft (LDR top-left)
botright (LDR bottom-right)
botleft (LDR bottom-left)
Calculate the average values
avgtop = (topright + topleft) / 2
avgbot = (botright + botleft) / 2
avgright = (topright + botright) / 2
avgleft = (topleft + botleft) / 2
Calculate the differences (Azimuth & elevation)
diffelev = avgtop - avgbot
diffazi = avgright – avgleft
avgsum = (avgtop+avgbot+avgright+avgleft) /4
avgsum < 8
Y
Rotate the Servo Motors to the initial position
N
|diffazi| <=10
Y
Stop the Left-Right Servo Motor
N
diffazi > 0
Y
Left-Right Servo Motor move PV panel Right
N
Left-Right Servo Motor move PV panel Left
|diffelev| <=10
Y
Stop the Up-Down Servo Motor
N
diffelev > 0
Y
Up-Down Servo Motor move PV panel Up
N
Up-Down Servo Motor move PV panel Down

Fig. 4. The algorithm for automatic mode of the solar tracker

The **PLX-DAQ** Excel Macro is used for data acquisition from the Arduino microcontroller to an Excel Spreadsheet. We only need to download it. After installation, a folder named "PLX-DAQ" will automatically be created on the PC in which a shortcut named "PLX-DAQ Spreadsheet" is inside. Then, to establish the communication between the board and Excel, we just need to open the Spreadsheet and defining the connections settings (Baud rate and port) in the PLX-DAQ window (Fig. 5). Thereafter, after clicking on "connect" the output data will be collected and displayed in real-time on the Excel Spreadsheet


Connect using "PLX-DAQ Simple Test"

Data Acquisition for Excel

Settings

Port: 7

Baud: 9600

Reset on Connect: ✓

Fig. 5. PLX-DAQ Spreadsheet window

The prototype

Figure 6 shows the solar tracker in detached and assembled states. As presented, the entire structure has been manufactured using wooden plates, and it is clear that the all mentioned components have been used to build the solar tracker with manual and automatic modes (LDR sensors, Arduino Uno, Servo motors, potentiometer, pushbuttons, and the small PV panel).

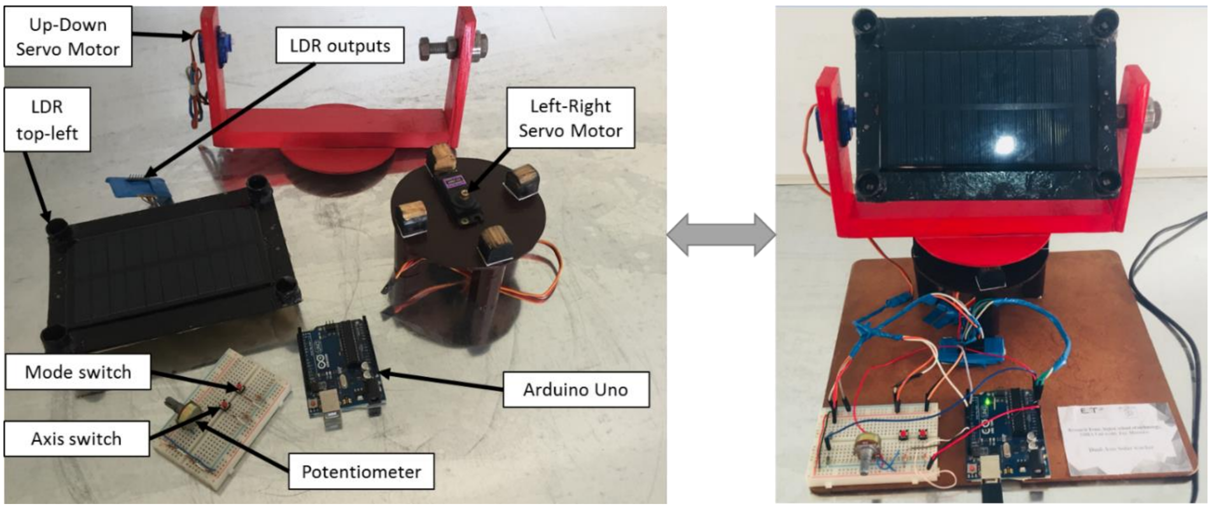


Fig. 6. Composition of the solar tracker

Figure 7 shows the entire test bench, solar tracker with virtual instrumentation, and an artificial lamp that can expose light to test the solar tracker. The solar tracker and the computer are connected through a USB cable. Once the PV voltage is acquired, the controller treats this information and uses it to compute the PV current and power. Then, all these data are sent to the computer to present them in MSExcel. From Figs. 5 and 6, it is clear that the proposed test bench is small, flexible and easy to use. It can enable students, researchers and engineers to apply their algorithms in an easy way before proceeding with the implementation of a large solar tracking device.

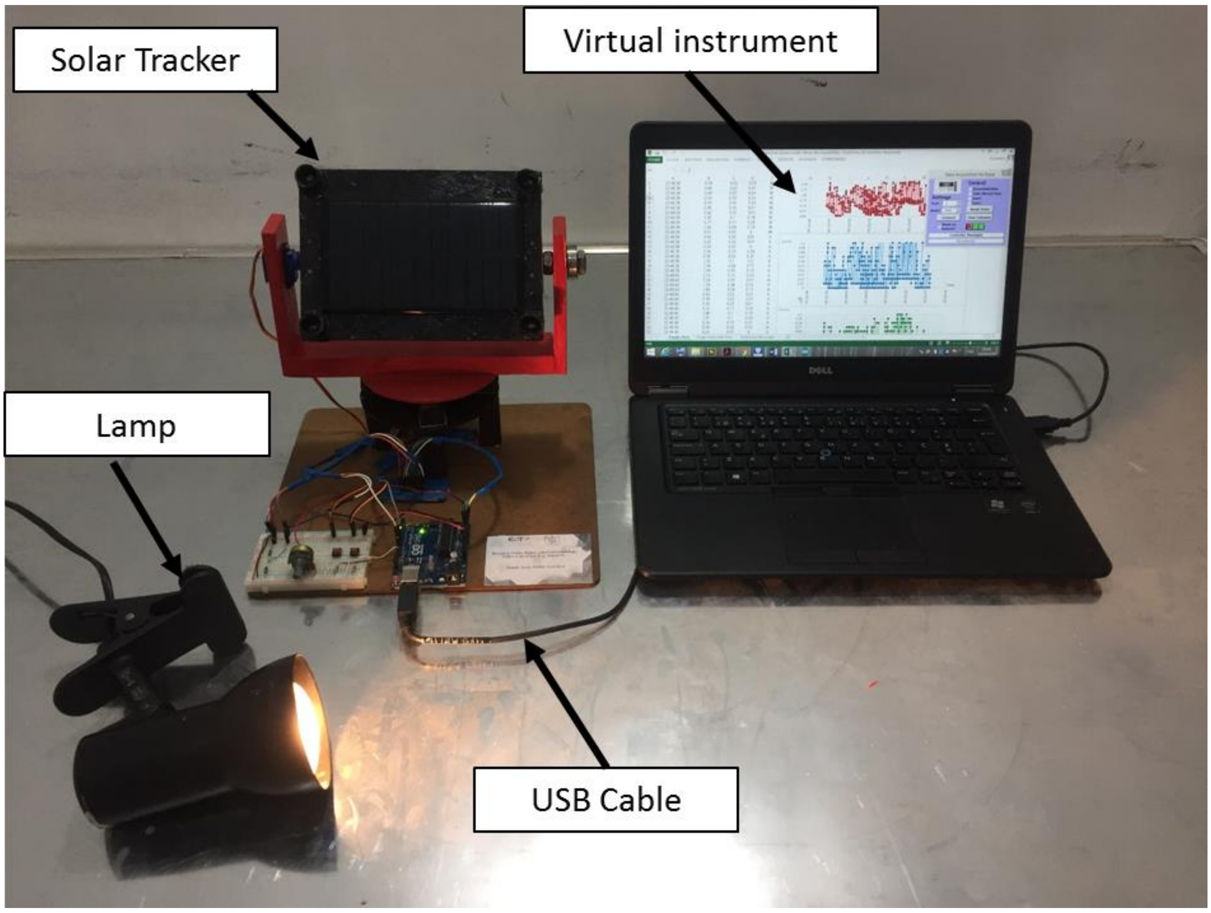


Fig. 7. The entire test bench with virtual instrumentation

Code

→Embedded Software as solar tracker test bench

```
//Servo motor library

#include <Servo.h>
```

```
//Initialize variables

int  mode = 0;

int axe = 0;

int buttonState1 = 0;

int buttonState2  = 0;

int prevButtonState1 = 0;

int prevButtonState2 = 0;


int ldrtopr=  0;                // top-right LDR

int ldrtopl = 1;               // top-left LDR

int ldrbotr = 2;               //  bottom-right LDR

int ldrbotl = 3;               // bottom-left  LDR

int topl = 0;

int topr = 0;

int botl = 0;

int  botr = 0;


//Declare two servos

Servo servo_updown;

Servo servo_rightleft;


int  threshold_value=10;           //measurement sensitivity


void setup()

{
  Serial.begin(9600);                          //serial connection setup  //opens  serial port, sets
data rate to 9600 bps

  Serial.println("CLEARDATA");                   //clear  all data that’s been place in already

  Serial.println("LABEL,t,voltage,current,power,Mode");   //define the column headings
(PLX-DAQ command)



  pinMode(12, INPUT);            //Mode  switch Button

  pinMode(11, INPUT);            //Axis switch

  pinMode(A4,  INPUT);             //Potentiometer for right-left movement and for up-down
movement



  servo_updown.attach(5);            //Servo motor up-down movement

  servo_rightleft.attach(6);         //Servo motor right-left movement

}



void loop()

{

//  pv_power();

char  Mode;

    float volt = analogRead(A5)*5.0/1023;

    float voltage = 2*volt;               //  Volt=(R1/R1+R2)*Voltage / R1=R2=10Ohms  =>
voltage=2*volt)

    float current = voltage/20;           //  I=voltage/(R1+R2)

    float power  = voltage*current;

    Serial.print("DATA,TIME,"); // PLX-DAQ command

    Serial.print(voltage);    //send the voltage to serial port

    Serial.print(",");
    Serial.print(current);    //send the current to serial port
    Serial.print(",");
    Serial.print(power);  //send the power to serial port
    Serial.print(",");

//    Serial.println(Mode);
  buttonState1 = digitalRead(12);
  if (buttonState1 != prevButtonState1)  {
    if (buttonState1 == HIGH) {
      //Change mode and ligh up the correct  indicator
      if (mode == 1) {
        mode = 0;
      } else {
        mode = 1;
      }
    }
  }
  prevButtonState1 = buttonState1;
  delay(50); // Wait for 50 millisecond(s)

  if (mode == 0) {
    Mode='M';
    Serial.println(Mode);   //send Mode "Manual" to serial port
    manualsolartracker();
  } else { // mode automatic
```

```
    Mode = 'A';

    Serial.println(Mode);

    automaticsolartracker(); //send Mode "Automatic" to serial port

    }

}


void  automaticsolartracker(){


     //capturing analog values of each LDR

     topr= analogRead(ldrtopr);         //capturing analog value of top right LDR

     topl= analogRead(ldrtopl);         //capturing analog value of top left LDR

     botr= analogRead(ldrbotr);         //capturing analog value of bot right LDR

     botl= analogRead(ldrbotl);         //capturing analog value of bot left LDR


    // calculating average

     int avgtop = (topr + topl) / 2;     //average  of top LDRs

     int avgbot = (botr + botl) / 2;     //average of bottom LDRs

     int avgleft = (topl + botl) / 2;    //average of left LDRs

     int avgright  = (topr + botr) / 2;   //average of right LDRs


    //Get the different

     int diffelev = avgtop - avgbot;      //Get the different average betwen  LDRs top and LDRs
bot

     int diffazi = avgright - avgleft;    //Get the different  average betwen LDRs right and LDRs
left
```

```
    //left-right movement of  solar tracker



      if (abs(diffazi) >= threshold_value){        //Change  position only if light difference is
bigger then the threshold_value

       if  (diffazi > 0) {

        if (servo_rightleft.read() < 180) {

          servo_rightleft.write((servo_updown.read()  + 2));

        }

      }

      if (diffazi <  0) {

        if (servo_rightleft.read()  > 0) {

          servo_rightleft.write((servo_updown.read() - 2));

        }

      }

    }



      //up-down movement of solar tracker



      if (abs(diffelev) >= threshold_value){    //Change position only if light  difference is
bigger then thethreshold_value

       if (diffelev > 0) {

        if  (servo_updown.read() < 180) {

          servo_updown.write((servo_rightleft.read()  - 2));

        }

      }

      if (diffelev <  0) {
        if (servo_updown.read()  > 0) {
          servo_updown.write((servo_rightleft.read() + 2));
        }
      }
    }
  }

void manualsolartracker(){
  buttonState2  = digitalRead(13);
  if (buttonState2 != prevButtonState2) {
    if (buttonState2  == HIGH) {
      //Change mode and ligh up the correct indicator
      if  (axe == 1) {
        axe = 0;
      } else {
        axe = 1;
      }
    }
  }
  prevButtonState2 = buttonState2;
  delay(50); // Wait for 50  millisecond(s)
  if (axe == 0) {     //control right-left movement
    servo_rightleft.write(map(analogRead(A4),  0, 1023, 0, 180));
  } else { // //control up-down movement
    servo_updown.write(map(analogRead(A4),  0, 1023, 0, 180));
  }

}
```

→Embedded Software os solar tracker test bench

```
//Servo motor library

#include <Servo.h>

//Initialize variables

int

  mode = 0;

int axe = 0;

int buttonState1 = 0;

int buttonState2

  = 0;

int prevButtonState1 = 0;

int prevButtonState2 = 0;


int ldrtopr=

  0;              // top-right LDR

int ldrtopl = 1;

              // top-left LDR

int ldrbotr = 2;             //

  bottom-right LDR

int ldrbotl = 3;             // bottom-left

  LDR

int topl = 0;

int topr = 0;

int botl = 0;

int

  botr = 0;
```

```
//Declare two servos

Servo servo_updown;

Servo servo_rightleft;


int

  threshold_value=10;          //measurement sensitivity


void setup()

{


  Serial.begin(9600);                              //serial connection setup  //opens

  serial port, sets data rate to 9600 bps

  Serial.println("CLEARDATA");                     //clear

  all data that’s been place in already

  Serial.println("LABEL,t,voltage,current,power,Mode");

   //define the column headings (PLX-DAQ command)


  pinMode(12, INPUT);            //Mode

  switch Button

  pinMode(11, INPUT);            //Axis switch

  pinMode(A4,

  INPUT);            //Potentiometer for right-left movement and for up-down movement



  servo_updown.attach(5);            //Servo motor up-down movement

  servo_rightleft.attach(6);

         //Servo motor right-left movement

}


void loop()

{
```

```
//  pv_power();
char
  Mode;
    float volt = analogRead(A5)*5.0/1023;
    float voltage = 2*volt;
                //  Volt=(R1/R1+R2)*Voltage / R1=R2=10Ohms  => voltage=2*volt)

    float current = voltage/20;           //  I=voltage/(R1+R2)
    float power
  = voltage*current;
    Serial.print("DATA,TIME,"); // PLX-DAQ command
    Serial.print(voltage);
    //send the voltage to serial port
    Serial.print(",");
    Serial.print(current);
    //send the current to serial port
    Serial.print(",");
    Serial.print(power);
  //send the power to serial port
    Serial.print(",");

//    Serial.println(Mode);

  buttonState1 = digitalRead(12);
  if (buttonState1 != prevButtonState1)
  {
    if (buttonState1 == HIGH) {
      //Change mode and ligh up the correct
  indicator
      if (mode == 1) {
        mode = 0;
      } else {

        mode = 1;
      }
    }
  }
  prevButtonState1 = buttonState1;

  delay(50); // Wait for 50 millisecond(s)

  if (mode == 0) {
    Mode='M';

    Serial.println(Mode);   //send Mode "Manual" to serial port
    manualsolartracker();

  } else { // mode automatic
    Mode = 'A';
    Serial.println(Mode);

    automaticsolartracker(); //send Mode "Automatic" to serial port
    }
}

void
  automaticsolartracker(){

     //capturing analog values of each LDR

     topr= analogRead(ldrtopr);         //capturing analog value of top right LDR
  topl= analogRead(ldrtopl);        //capturing analog value of top left LDR

  botr= analogRead(ldrbotr);        //capturing analog value of bot right LDR

  botl= analogRead(ldrbotl);        //capturing analog value of bot left LDR


 // calculating average
  int avgtop = (topr + topl) / 2;     //average
of top LDRs
  int avgbot = (botr + botl) / 2;     //average of bottom LDRs

  int avgleft = (topl + botl) / 2;    //average of left LDRs
  int avgright
= (topr + botr) / 2;   //average of right LDRs

 //Get the different

  int diffelev = avgtop - avgbot;      //Get the different average betwen
LDRs top and LDRs bot
  int diffazi = avgright - avgleft;    //Get the different
average betwen LDRs right and LDRs left

 //left-right movement of
solar tracker

   if (abs(diffazi) >= threshold_value){       //Change
position only if light difference is bigger then the threshold_value
    if
(diffazi > 0) {
     if (servo_rightleft.read() < 180) {
        servo_rightleft.write((servo_updown.read()
+ 2));
      }
    }
    if (diffazi <  0) {
      if (servo_rightleft.read()
> 0) {
        servo_rightleft.write((servo_updown.read() - 2));
      }


    }
  }


    //up-down movement of solar tracker



    if (abs(diffelev) >= threshold_value){    //Change position only if light
difference is bigger then thethreshold_value
     if (diffelev > 0) {
      if
(servo_updown.read() < 180) {
        servo_updown.write((servo_rightleft.read()
- 2));
      }
    }
    if (diffelev <  0) {
      if (servo_updown.read()
> 0) {
        servo_updown.write((servo_rightleft.read() + 2));
      }
      }
    }
 }

void manualsolartracker(){
  buttonState2
  = digitalRead(13);
  if (buttonState2 != prevButtonState2) {
    if (buttonState2
  == HIGH) {
      //Change mode and ligh up the correct indicator
      if
  (axe == 1) {
        axe = 0;
      } else {
        axe = 1;
      }


    }
  }
  prevButtonState2 = buttonState2;
  delay(50); // Wait for 50
  millisecond(s)
  if (axe == 0) {     //control right-left movement
    servo_rightleft.write(map(analogRead(A4),
  0, 1023, 0, 180));
  } else { // //control up-down movement
    servo_updown.write(map(analogRead(A4),
  0, 1023, 0, 180));
  }
}
```