# Microcontroller based Industrial Applications : Project Report

*By-Pranati Limaye(21EE0104)-VIT (Batch 9)*

## Problem Statement:

To develop a microcontroller based solar panel positioning system that offers maximum energy capture from sunlight. The main objective of this project is to attempt a solar panel positioning system which can change position(angle) dynamically to optimize it's energy generation according to the sun's position and intensity with respect to it and provide a live feedback to the user regarding it's position.

This system can further be extended to an IOT system using Arduino Nano or ESP8266 WIFI providing the consumer live feedback and allowing a certain degree of shut down or position control at the consumer-end for emergencies ( windy/stormy conditions) which may threaten the entire panel assembly.

## Scope of Solution:

The dynamic solar panel positioning system senses the sun's position and intensity based on the feedback from it's sensors. Based on the feedback it decides and calculates the angle of positioning of the solar panel and feeds it to the servo motors. The angles fed to the servo motors are complementary to with respect to one another since they are placed opposite to each other. The system is dynamic and changes position as soon as required since the looping period for the code is quick. The system provides a live feedback to the user with the help of an LCD.

Objectives of System:

1. Positioning the solar panel at the right angle.
2. Providing Live feedback to the consumer via LCD of the solar panel's position.

## Components Used

IDE : Arduino IDE

Software/Programming Language: C++

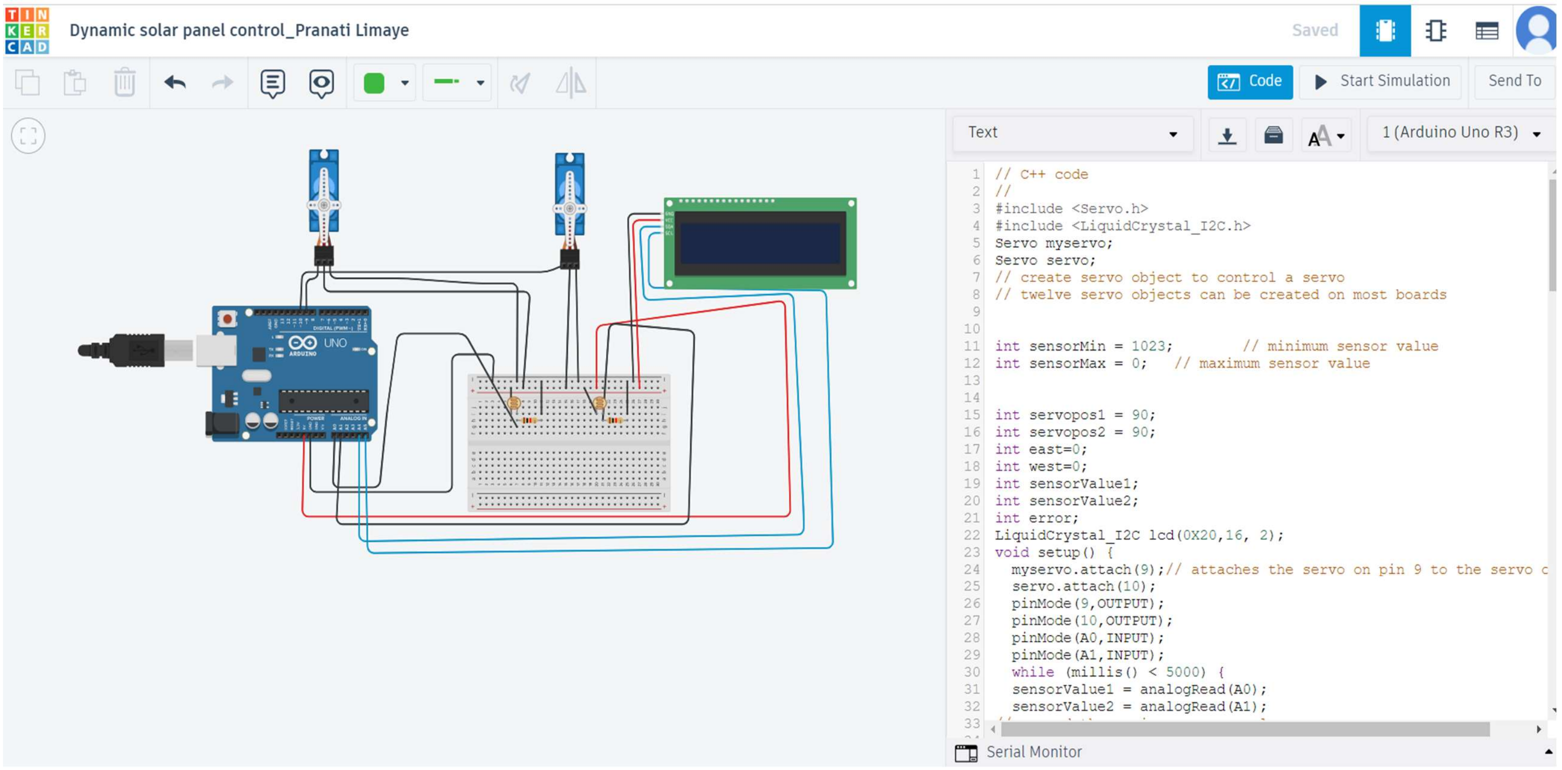
Libraries: < Servo.h> ; < LiquidCrystal_I2C.h>

Hardware Components: Arduino Uno R3 board, Servo motorSG90, Servo MotorSM-S2098, G15528 Photo resistors(2), 110kohm resistors(2),LCD (I2C),jumper wires

## Simulated circuit:

Platform(TinkerCAD):

Link:
https://www.tinkercad.com/things/gOoIUAFxK26?sharecode=KU9XDcS2r30Zj7uo7hvdr7G5jcTUb4drqBuWj_sMnes



```
// C++ code
//
#include <Servo.h>
#include <LiquidCrystal_I2C.h>
Servo myservo;
Servo servo;
// create servo object to control a servo
// twelve servo objects can be created on most boards


int sensorMin = 1023;         // minimum sensor value
int sensorMax = 0;   // maximum sensor value


int servopos1 = 90;
int servopos2 = 90;
int east=0;
int west=0;
int sensorValue1;
int sensorValue2;
int error;
LiquidCrystal_I2C lcd(0X20,16, 2);
void setup() {
  myservo.attach(9);// attaches the servo on pin 9 to the servo c
  servo.attach(10);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  while (millis() < 5000) {
  sensorValue1 = analogRead(A0);
  sensorValue2 = analogRead(A1);
```

## Video for Demo:

Google Drive Link for TinkerCAD Demo:

https://drive.google.com/file/d/1Oysha06TNMV0lxgHX7nI0CxPd5SRA7tU/view?usp=sharing

Google Drive Link for Hardware Demo:

https://drive.google.com/file/d/1isdfYfewdpjqerhOol3K43M7YooxZ8uo/view?usp=sharing

Code for solution:

```
#include <Servo.h>
#include <LiquidCrystal_I2C.h>
Servo myservo;
Servo servo;
// create servo object to control a servo
// twelve servo objects can be created on most boards


int sensorMin = 1023;        // minimum sensor value
int sensorMax = 0;   // maximum sensor value


int servopos1 = 90;
int servopos2 = 90;
int east=0;
int west=0;
int sensorValue1;
int sensorValue2;
int error;
LiquidCrystal_I2C lcd(0X20,16, 2);
void setup() {
  myservo.attach(9);// attaches the servo on pin 9 to the servo object
  servo.attach(10);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  while (millis() < 5000) {
  sensorValue1 = analogRead(A0);
  sensorValue2 = analogRead(A1);
// record the maximum sensor value
 if (sensorValue1 > sensorMax) {
  sensorMax = sensorValue1;
}
// record the minimum sensor value
 if (sensorValue1 < sensorMin) {
  sensorMin = sensorValue1;
}
}
 if (sensorValue2 > sensorMax) {
  sensorMax = sensorValue2;
}
// record the minimum sensor value
 if (sensorValue2 < sensorMin) {
  sensorMin = sensorValue2;
}
```

```
  lcd.init();
  lcd.backlight();
}
void loop() {
  sensorValue1 = analogRead(A0);

  sensorValue2 = analogRead(A1);

  if (sensorValue1 < 350 && sensorValue2 < 350)

  {

    while (servopos1 <= 150)

    {

      servopos1++;

      myservo.write(servopos1);

      delay(100);

    }
    while (servopos2 >= 30)

    {

      servopos2++;

      servo.write(servopos2);

      delay(100);

    }


  }

  error = sensorValue1 - sensorValue2;

  if (error > 15)

  {

    if (servopos1 <= 150)

    {
```

```
    servopos1++;

    myservo.write(servopos1);

  }
  if (servopos2 >= 30)

  {

    servopos2--;

    servo.write(servopos2);

  }

}

else if (error < -15)

{

  if (servopos1 > 20)

  {

    servopos1--;

    myservo.write(servopos1);

  }
  if (servopos2 < 160)

  {

    servopos2++;

    servo.write(servopos2);

  }

}
lcd.setCursor(3, 0);
lcd.print(myservo.read());
lcd.setCursor(7,0);
lcd.print("degrees");
lcd.setCursor(3,1);
lcd.print("from east");
```

```
  delay(100);


 // waits 15ms for the ser
}
```

Results:

The circuit was first simulated on TinkerCAD using the listed components and secured the desired output. The circuit was implemented in hardware using a breadboard and the listed components, an (I2C Lcd couldn't be sourced hence the output couldn't be visualised on an LCD.) The required output was obtained successfully on the servo motors.

Gerber File: Uploaded in GitHub Repository

**GitHub Repository Link:** **https://github.com/Pplimaye/PROJECT.git**