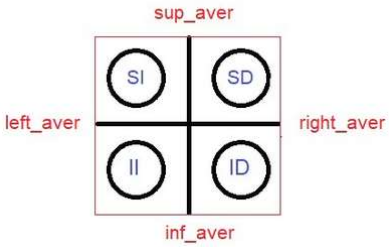
Beschreibung der Funktionsweise und des Quellcodes

Für die Bewegung des Solarpanels wurden zwei 28BYJ-Motoren und die dazugehörigen ULN2003-Controllermodule gewählt.

Der neue Code enthält Änderungen im Code Text und Treiber TB7600 und NEMA17

Wie auf dem folgenden Foto zu sehen ist, befinden sich sowohl das Solarpanel als auch die LDR-Widerstände in derselben Ebene. Dies ist notwendig, da die Lichtmenge, die das Solarpanel empfängt, dieselbe sein muss, die auch auf die vier LDRs trifft. Diese sind in einer 2 x 2-Matrix angeordnet.

Die folgende Zeichnung zeigt die Anordnung der LDR-Widerstände von vorne gesehen. Diese Referenzen werden bei den Berechnungen berücksichtigt, um die beste Lichtmessung zu erhalten.



Das Herzstück des Aufbaus ist der Mega 2560 R3, der die Messwerte der 4 LDR-Widerstände empfängt, die notwendigen Berechnungen durchführt, die beiden Schrittmotoren bewegt und die Spannungsdaten an das OLED-Display sendet.

Als erstes müssen die notwendigen Bibliotheken für die I2C-Kommunikation und das OLED-Display in den Sketch eingebunden werden.

```
#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>
```

Anschließend müssen wir die Anzahl der Pixel für Breite und Höhe des OLED-Bildschirms definieren und ein Bildschirm-Objekt implementieren. Als Konstruktor-Parameter übergeben wir die Variablen der Anzahl der Pixel für Breite und Höhe des Bildschirms, die Referenz auf das Wire-Objekt. Außerdem hier noch die -1 für den Anschluss des Reset-Pins, wenn er vorhanden ist.

```
#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

Wir deklarieren vier Variablen, um anzugeben, welcher LDR-Widerstand mit welchem Port des Mikrocontrollers verbunden ist. Außerdem deklarieren wir auch vier weitere Variablen für den Wert des Widerstandes, wenn das Licht auftrifft.

```
int LDR_SD = A0;
int LDR_SI = A1;
int LDR_II = A2;
int LDR_ID = A3;

int SD, SI, II, ID;
```

Um Durchschnittswerte in den jeweils gleichen Spalten und Zeilen der gemessenen vier Werte der LDR-Widerstände zu berechnen, deklarieren wir weitere vier Variablen. Dadurch erhalten wir eine Referenz, um die Ausrichtung des Solarpanels zu verändern. In der obigen Zeichnung sehen Sie, welche Widerstände auf jeder Seite beteiligt sind. Wir deklarieren auch eine Toleranzvariable, um ständige Bewegungen zu vermeiden.

```
int sup_aver, inf_aver, right_aver, left_aver;
int tolerance = 10;
```

Für die Spannungsmessung am Solarpanel deklarieren wir die Variable SPV. Sie enthält den Pin, an den wir das Panel anschließen (in unserem Projekt ist es Pin 4). Die Variable solar_panel_read für den Eingangswert der Messung der Spannung (Es handelt sich um analoge Werte) und die Variable solar_panel_voltage, um die Analog-Digital-Umwandlung durchzuführen. Mit dem Wert können wir arbeiten und ihn auf dem OLED-Bildschirm anzeigen.

```
int SPV = A4;
int solar_panel_read;
float solar_panel_voltaje;
```

Für die Geschwindigkeit der Schrittmotoren werden wir zwei Variablen deklarieren. Die Variable retardo (sp. delay) ist für die Bewegung der Motoren im normalen Betrieb gedacht. Die Variable retardo_inicializacion (sp. delay_initialisation) verwenden wir für die Bewegung des horizontalen Motors, wenn er sich in die Ausgangsposition bewegt.

```
int retardo = 5;
int retardo_inicializacion = 15;
```

Es wurden zwei Endschalter für Sonnenauf- und untergang in die Anlage eingebaut. Wenn der Sonnenaufgang stattfindet, befindet sich die Anlage in der Sonnenaufgangsposition. Der Sonnenaufgangsendschalter (switch_sunrise) ist normal closed (NC) und mit Pullup-Widerstand angeschlossen. Er ist immer geschlossen, wodurch dauerhaft 5V am Pin anliegen und sein Zustand somit HIGH ist. Er wird gedrückt und damit geöffnet, wenn sich das Panel in einer fast vertikalen Position befindet. Es zeigt in östlicher Richtung, da dort die Sonne aufgeht. Wenn das Panel während des Tages der Bewegung der Sonne am Himmel folgt, wird es bei Erreichen der Sonnenuntergangsposition wieder fast senkrecht stehen und Richtung Westen ausgerichtet sein.

Dann wird der normal open (NO) Sonnenuntergangs-Endschalter (switch_sunset) aktiviert. Er ist mit Masse und dem RESET-Pin des Mikrocontrollers verbunden. So wird er zurückgesetzt, wenn der Pin auf Masse gelegt wird. Dadurch kehrt das Solarpanel in die Sonnenaufgangsposition zurück. Dieser Code wird im setup()-Teil des Sketches ausgeführt. In den nächsten beiden Zeilen konfigurieren wir also den Anschluss des Endschalters an Port 10 des Mikrocontrollers und definieren ihn standardmäßig mit HIGH.

```
int sunrise_pin = 10;

int switch_sunrise = HIGH;
```

Nachdem wir die Bibliotheken eingebunden, die Variablen deklariert und die Komponentenobjekte instanziiert haben, müssen wir sie in der Methode setup() initialisieren. Als erstes initialisieren wir den seriellen Monitor und den OLED-Bildschirm an Adresse 0x3C, wir löschen den Bildschirm und stellen die Textfarbe Weiß ein.

```
Serial.begin(9600);


if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  for(;;);
}


display.clearDisplay();
display.setTextColor(WHITE);
```

Als Nächstes konfigurieren wir die Mikrocontroller-Pins, die von den LDR-Widerständen, dem Solarpanel, den Motoren und dem Endschalter für den Sonnenaufgang verwendet werden, als Eingang oder Ausgang.

```
pinMode (LDR_SD, INPUT);      // LDR resistors
pinMode (LDR_SI, INPUT);
pinMode (LDR_II, INPUT);
```

```
pinMode (LDR_ID, INPUT);

pinMode (SPV, INPUT);        // Solar panel

pinMode (5, OUTPUT);         // Horizontal motor
pinMode (4, OUTPUT);
pinMode (3, OUTPUT);
pinMode (2, OUTPUT);

pinMode (9, OUTPUT);         // Vertical motor
pinMode (8, OUTPUT);
pinMode (7, OUTPUT);
pinMode (6, OUTPUT);

pinMode (sunrise_pin, INPUT); // Sunrise limit switch
```

Die letzten Punkte, die wir konfigurieren werden, sind die Zustände des LDR-Widerstands und des Solarpanels, wenn der Mikrocontroller initialisiert wird. Als Erstes müssen wir den Zustand des Endschalters switch_sunrise kennen. Wenn er sich im HIGH-Zustand befindet, d.h. nicht gedrückt ist, zeigen wir eine Meldung über den seriellen Monitor an und führen die Methode inicializacion_motor_horizontal() (initialization_horizontal_motor) aus. Solange er nicht gedrückt ist, bewegt sich der Motor der horizontalen Achse, bis dieser Endschalter gedrückt wird und die Initialisierung abgeschlossen ist.

```
switch_sunrise = digitalRead(sunrise_pin);

while (switch_sunrise == HIGH) {
  Serial.println("Inicializando motor horizontal, espere ......");
  inicializacion_motor_horizontal();
  switch_sunrise = digitalRead(sunrise_pin);
}
```

Wir sind mit der setup()-Methode fertig, jetzt werden wir die loop()-Methode analysieren, die kontinuierlich ausgeführt wird. In dieser Methode werden das empfangene Licht gemessen und die Motoren bewegt.

Ein LDR (Light Dependent Resistor - lichtabhängiger Widerstand) ändert seinen Widerstand mit der Lichtmenge, die auf ihn fällt. Die beste Leistung wird erzielt, wenn das Licht senkrecht einfällt. Wie Sie im Schaltbild sehen können, wurde eine Spannungsteilerschaltung mit jedem LDR und einem 10K-Widerstand hergestellt. Wenn wir sie an eine Spannung von 5V anschließen und jeder LDR mit einem analogen Port des Mikrocontrollers verbunden ist, können wir die jeweils variierende Spannung messen. Wir werden die Spannungswerte jedes LDRs in den Variablen speichern, die wir zu diesem Zweck deklariert haben.

```
SD = analogRead(LDR_SD);
SI = analogRead(LDR_SI);
II = analogRead(LDR_II);
ID = analogRead(LDR_ID);
```

Die Berechnungen, die der Mikrocontroller für die Bewegung der Motoren durchführen muss, lauten wie folgt:

Es werden jeweils die Werte der horizontal gelegenen LDRs addiert und dann halbiert. Dann werden die jeweils vertikal liegenden LDRs ebenfalls addiert und halbiert. So erhalten wir die Mittelwerte. Die Ergebnisse speichern wir in den zuvor deklarierten Variablen. Die Anordnung der LDRs sehen Sie oben in der Abbildung.

```
sup_aver = (SD + SI)/2;
inf_aver = (ID + II)/2;
right_aver = (SD + ID)/2;
left_aver = (SI + II)/2;
```

Nun, da wir die Durchschnittswerte haben, müssen wir die beiden Motoren in Bewegung setzen. Dazu vergleichen wir die Werte der gegenüberliegenden Seiten mit der Toleranz. Immer wenn die berechnete Differenz der Werte der verglichenen Seiten größer ist, als die Toleranz (int tolerance = 10;), müssen wir mit Hilfe der Methoden paso_izq(), paso_der(), paso_up() y paso_down() (step left, right, up und down) dem entsprechenden Motor den Befehl geben. Er ändert die Richtung der Seite, deren Wert höher ist, bis der Wert der Differenz kleiner als die Toleranz ist.

```
/***************** Horizontal Motor *************************/
if (right_aver == left_aver) {
  apagado_hor();
}
```

```
if (right_aver > left_aver && (right_aver-left_aver) > tolerance) {
  paso_izq();
}

if (left_aver > right_aver && (left_aver-right_aver) > tolerance) {
  paso_der();
}
apagado_hor();
delay(500);

/*************** Vertical Motor ***************************/

if (sup_aver == inf_aver) {
   apagado_ver();
}

if (sup_aver > inf_aver && (sup_aver-inf_aver) > tolerance) {
    paso_up();
}

if (inf_aver > sup_aver && (inf_aver-sup_aver) > tolerance) {
    paso_down();
}
apagado_ver();
delay(500);
```

Für die Bewegung des Solarpanels und der LDR-Widerstände wurden zwei 28BYJ-Motoren und die dazugehörigen ULN2003-Controllermodule gewählt. Wobei ein Motor auf die Bewegung der X-Achse und der andere Motor auf die Y-Achse wirkt.

Hinweis: Es sollte eine externe Spannungsquelle für die Versorgung der Motoren angeschlossen werden.

In den folgenden Zeilen des Sketches sehen Sie, dass immer zwei Spulen, also zwei Pins für einen Schritt des Motos aktiviert werden. Die Geschwindigkeit der Motordrehung regulieren wir mit delay(retardo).

```
void paso_XXX() {
  digitalWrite(5, LOW);
  digitalWrite(4, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(2, HIGH);
     delay(retardo);
  digitalWrite(5, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(2, LOW);
     delay(retardo);
  digitalWrite(5, HIGH);
  digitalWrite(4, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(2, LOW);
     delay(retardo);
  digitalWrite(5, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(3, LOW);
  digitalWrite(2, HIGH);
     delay(retardo);
}
```

Um die Spannungswerte zu visualisieren, die das Solarmodul zu einem bestimmten Zeitpunkt erhält, haben wir einen 0,96-Zoll-OLED-Bildschirm installiert. Die Spannung des Solarpanels ermitteln wir an Pin 4. Vor der Ausgabe auf dem Display müssen wir den analogen in einen digitalen Wert umwandeln.

Das Solarpanel liefert eine maximale Spannung von 5V, die dem am analogen Eingang eingelesenen Wert 1023 entspricht. Mit einer einfachen Operation können wir diesen Wert umwandeln und auf dem Bildschirm ausgeben.

```
solar_panel_read = analogRead(SPV);
solar_panel_voltaje = (solar_panel_read*5.0) / 1023.0;
oled_message();
```

Die Methode zur Anzeige von Daten auf dem OLED-Bildschirm (oled_message()) ist sehr einfach. Sie führt folgende Schritte aus:

Löschen des Bildschirms, Einstellen der Textgröße auf 2, Platzieren des Cursors auf den X- und Y-Koordinaten in Pixeln und Vorbereiten des Textes, um ihn in die erste Zeile des Textes zu schreiben. Textgröße auf 3 einstellen, Cursor erneut auf die neuen X- und Y-Koordinaten bringen, den neuen anzuzeigenden Text vorbereiten (Spannungswert mit 2 Dezimalstellen des Solarpanels) und die letzte Zeile ist die Ausgabe des gesamten zuvor eingestellten Textes.

```
void oled_message() {
  display.clearDisplay();
  display.setTextSize(2);
  display.setCursor(7,0);
  display.print("Solar Vols");
  display.setTextSize(3);
  display.setCursor(20, 30);
  display.print(solar_panel_voltaje, 2);
  display.display();
}
```