מנוע SERVO

א. כללי

מנוע סרוו הוא מנוע שבו יש את המרכיבים הבאים :

- מנוע זרם ישר (DC Motor)
- מערכת תמסורת פנימית של גלגלי שיניים
- בקרה אלקטרונית על מיקום המנוע.

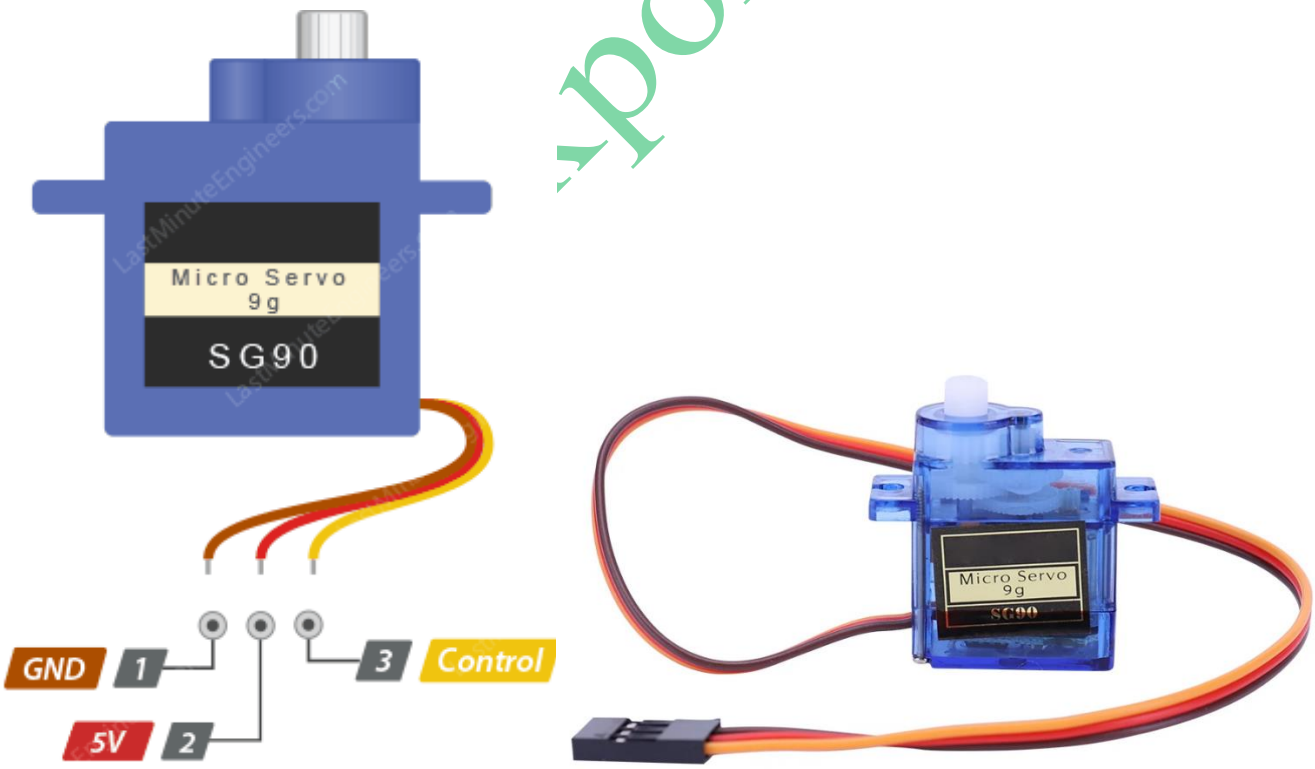
מנועי סרוו לא מסתובבים בצורה חופשית כמו מנועי DC , אלא נעים על פי זווית – לרוב בין 0 ל- 180 מעלות (אם כי יש מנועי סרוו שמסתובבים גם עד 360 מעלות) .

מנועי סרוו נפוצים מאוד ומשמשים באפליקציות רבות . הסיבות לכך הן :

קלות השליטה על מנועי הסרוו, דרישות האנרגיה הנמוכות (יעילות), הכוח החזק יחסית של המנוע , רמת המתח שמשתמשים להפעלתו, הגודל, המשקל והמחיר הנמוכים שלו.

מנועי סרוו פועלים בחוג סגור, כלומר יש להם מערכת בקרה על מיקום המנוע והם יכולים לתקן הפרשים מהמיקום הרצוי.

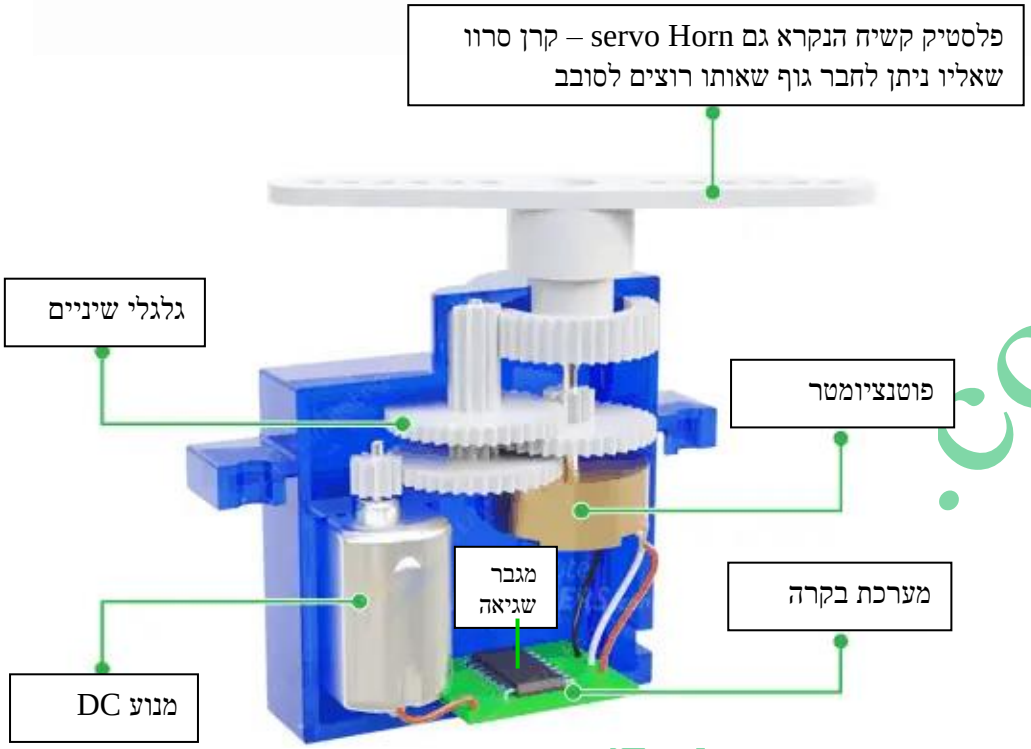
האיור הבא מתאר מנוע סרוו שניתן להשיג באינטרנט במחירים נמוכים יחסית ואת החיבורים שלו . המנוע נקרא SG90:



איור 1 : מנוע סרוו והחיבורים שלו .

מנוע סרוו מכיל מנוע DC קטן המחובר לציר היציאה דרך גלגלי שיניים. על הציר מחובר פוטנציומטר המחזיר משוב למערכת בקרה על המיקום הנוכחי.

האיור הבא מתאר את המבנה של מנוע סרוו :



איור 2 : מבנה עקרוני של מנוע סרוו

ב. עקרון פעולה של מנוע סרוו

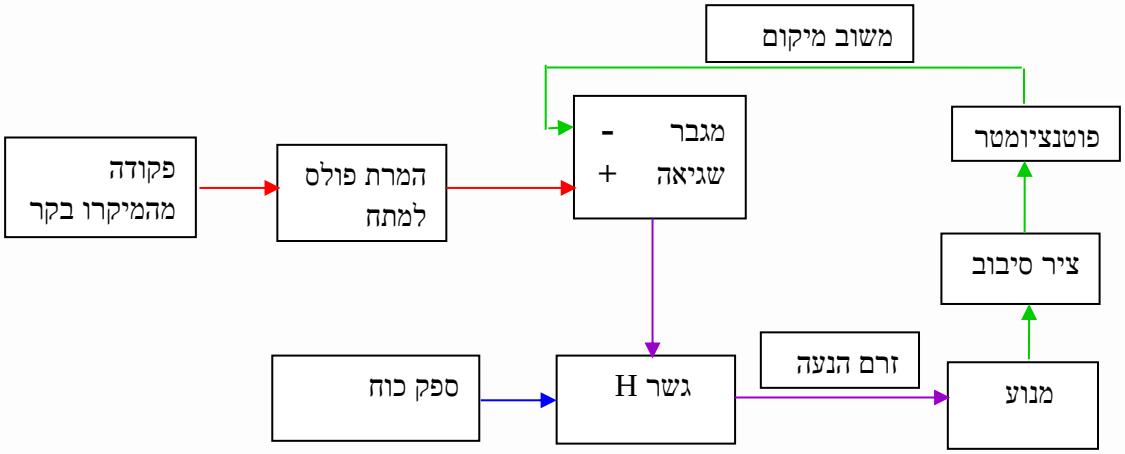
מנוע סרוו מופעל בדרך כלל על ידי מיקרו בקר הגורם לו להגיע לזווית רצויה. פקודת התוכנה מהמיקרו בקר מתורגמת לפולס מתח PWM שמועבר ליחידת הבקרה הנמצאת בתוך המנוע. ערך זה הוא **הערך הרצוי** בחוג הבקרה.

הערך המצוי של המנוע, נמדד בעזרת פוטנציומטר המחובר לציר המניע של המנוע (הציר שיוצא מחלקו העליון של המנוע ואליו מחברים את המנגנון שיש להניע). סיבוב הפוטנציומטר גורם לשינוי ההתנגדות שלו ובהתאם לשינוי מפל המתח על זחלן הפוטנציומטר. מתח זה הוא **הערך המצוי** שמגיע כמשוב לחוג הבקרה הסגור. כל עוד יש הפרש (הנקרא **מתח שגיאה**) בין המתח המצוי ולבין המתח הרצוי, מנוע ה-DC הנמצא בתוך הסרוו, ימשיך לנוע בכיוון המתאים.

סיבוב מנוע ה DC מתבצע יחד עם סיבוב גלגלי השיניים - שמהווים תמסורת הפחתה (מאיטה את מהירות הסיבוב ומגדילה את מומנט הכוח). גלגלי השיניים מניעים את ציר היציאה ואת הפוטנציומטר. ברגע שהפרש המתחים יהיה קטן מערך סף (קרוב ל 0) , יחידת הבקרה של המנוע תנתק את מקור המתח למנוע ה DC והוא יחדל לנוע. עצירת המנוע תהיה בדיוק בזווית שנשלחה מהתוכנית במיקרו בקר.

אם התוכנית שולחת למנוע פקודה לנוע לזווית מעבר ל-180 מעלות , המנוע יתחיל לרעוד מכיוון שהוא ינסה לסגור את השגיאה בין הערך הרצוי למצוי ולא יצליח. כמו כן יש על ציר היציאה מחסום מכני המונע מהמנוע לעבור את זווית הסיבוב של 180 מעלות .

האיור הבא מתאר בצורה מלבנית עיקרון עבודה של מערכת בקרה של סרוו.



איור 3 : מערכת בקרה סרוו

בצד שמאל מגיע אות הפקודה מהמיקרו בקר על הזווית הרצויה שנקרא **הערך הרצוי**. זהו פולס עם Duty Cycle כאשר הזמן בין ה ON ל OFF קובע את הזווית הרצויה . אות זה נקרא גם **מיקום מטרה**.

מעגל "המרת פולס למתח" מעביר למגבר השגיאה מתח שהוא ממוצע ה Duty Cycle של אות הפקודה.

מגבר השגיאה מקבל מהפוטנציומטר מתח היחסי למיקום ציר המנוע, מתח זה הוא של המיקום הנוכחי של המנוע ונקרא **ערך מצוי.**

מגבר השגיאה מגביר את הפרש המתח בין הרצוי למתח המצוי. יציאת המגבר מחוברת לגשר H הנקרא H-Bridge שמעביר זרם המסובב את המנוע.

כאשר המנוע מסתובב הוא מסובב את הפוטנציומטר שמעביר משוב מתח שלילי אל הכניסה המהפכת (המינוס של המגבר) של המגבר . ככל שהמנוע מסתובב וציר המנוע מתקרב אל הזווית הרצויה הפרש המתח הולך וקטן. כאשר מגיעים לזווית הרצויה , הפרש המתח למגבר הוא 0 והוא איננו מוציא פקודה לסיבוב המנוע והמנוע עוצר.

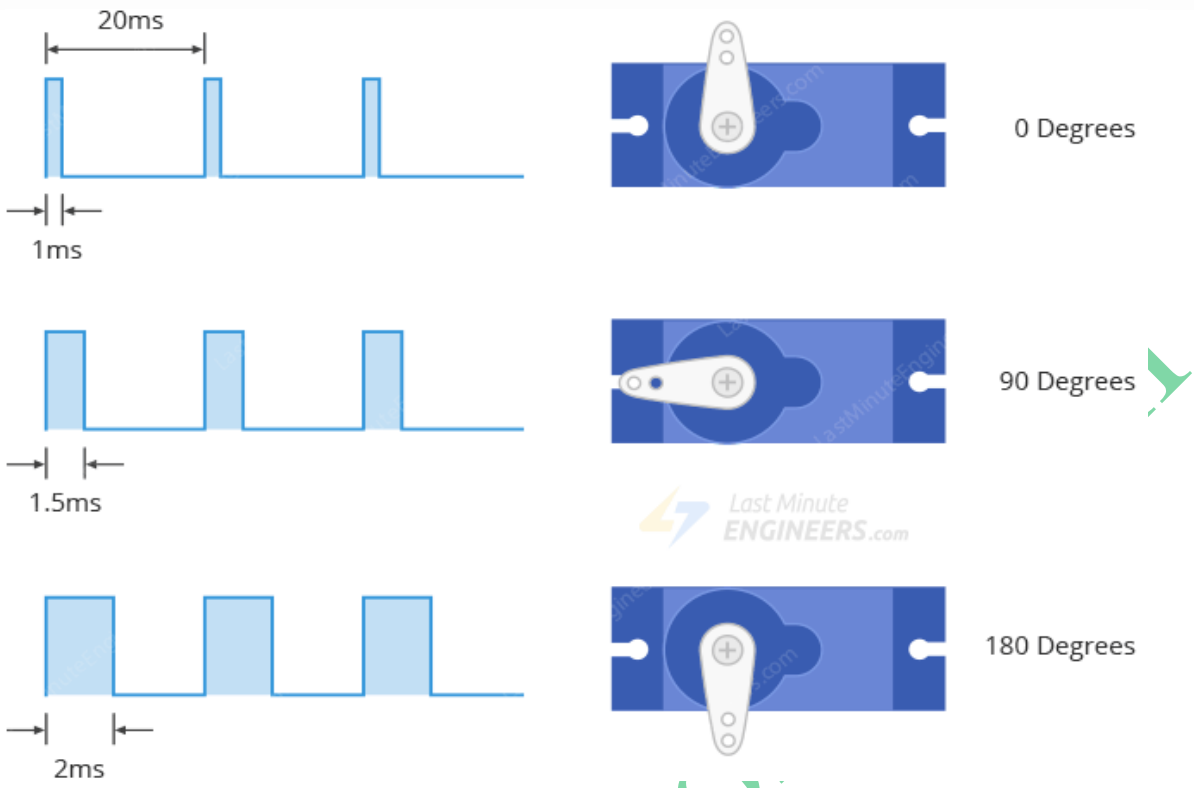
ג. כיצד עובד מנוע סרוו ?

ג.1 עבודה עם פולסים

אפשר לשלוט על מנוע סרוו על ידי שליחת סדרה של פולסים אליו. מנוע סרוו טיפוסי מחכה לפולס כל 20 אלפיות השנייה (כלומר, תדר האות שנשלח המיקרו בקר צריך להיות 50Hz).

רוחב הדופק קובע את מיקום מנוע הסרוו.

האיור הבא מראה את הזווית של המנוע עבור Duty Cycles עם רוחבי דופק שונים .



איור 4 : מיקום המנוע כתלות ברוחב הדופק של ה Duty Cycle .

מהאיור רואים :

- פולס קצר ברוחב של 1 מילישניות או פחות מסובב את מנוע הסרוו ל-0 מעלות .
- פולס ברוחב של 1.5 אלפיות השנייה יסובב את מנוע הסרוו ל 90 מעלות .
- פולס ברוחב של 2 אלפיות השנייה בערך יסובב את מנוע הסרוו ל 180 מעלות .

פולסים בטווח של 1ms עד 2ms יסובבו את הסרוו למיקום פרופורציונלי לרוחב הפעימה.

יש מנועי סרוו המסוגלים להסתובב 360 מעלות. רוחב הפולס במנועים אלו ייתן את הזוויות הבאות :

5.0 ms נותן זווית של 0 מעלות.

1 ms נותן זווית של 45 מעלות.

ms,1.5 נותן זווית של 90 מעלות.

2 ms נותן זווית של 135 מעלות.

ms2.5 נותן זווית של 180 מעלות.

הערה : אין תקן ליחס המדויק בין רוחב הפולסים למיקום ולכן ייתכן שיהיה עלינו לשנות את התכנון שלנו כדי להתאים לטווח של מנוע הסרוו שלנו. כמו כן, משך/רוחב הדופק יכול להשתנות בין יצרני מנועי סרוו שונים; לדוגמה, זה יכול להיות 2.5ms עבור 180 מעלות ו 0.5ms עבור 0 מעלות .

ג.2 מאפיינים טכניים

הטבלה הבאה מתארת את המאפיינים של המנוע שהשתמשתי בו.

Tech specs

Dimensions מידות	23.2 x 12.5 x 22 mm
Weight משקל	9 g
Operating Speed מהירות פעולה	0.12sec/60degrees (4.8V) 0.10sec/60degrees (6V)
Stall Torque מומנט מעכב	1.3kg.cm/18.09oz.in (4.8V) 1.5kg.cm/20.86oz.in(6V)
Operating Voltage מתח הפעלה	4.8V~6V
Control System מערכת בקרה	Analog
Direction כיוון	CCW
Operating Angle זווית פעולה	120degrees
Required Pulse הפולס הנדרש	900us-2100us
Bearing Type סוג מיסב	None
Gear Type סוג גלגל השיניים	Metal
Motor Type סוג המנוע	Plastic
Connector Wire Length אורך חוטי החיבור	20 cm

טבלה 1 : מאפיינים טכניים

למנוע הסרוו הנקרא - Servo Micro S90 יש צריכת זרם נמוכה . בסביבות של mA100-200 ללא עומס והוא יכול להגיע לזרם של mA500-700 בזמן עבודה עם עומס.

ד. תכנות מנוע סרוו עם מיקרו בקר בסביבת ארדואינו

כאשר עובדים עם מנוע סרוו בסביבת העבודה של ארדואינו – Arduino IDE – יש לדאוג שבספרייה יהיה הקובץ Servo.h . בקובץ הזה יש הגדרות של כל הפונקציות/מתודות המיוחדות שנכתבו עבור תפעול מנוע סרוו. כדי להשתמש בתוכן הספרייה עלינו לכתוב בראש התוכנית את השורה הבאה, שאומרת לבקר לצרף/לכלול אל התוכנית שאנחנו כותבים גם את כל הגדרות הפונקציות/מתודות המתאימות למנוע סרוו:

```
#include <Servo.h>
```

בתוך הספרייה ישנה הגדרה של מחלקה (class) שמאפשר גישה לכל הפונקציות המיוחדות. שמו של מבנה הנתונים Servo ועושים בו שימוש כמו שמגדירים משתנה חדש (למעשה יוצרים כאן **אובייקט**) : סוג המשתנה ושם המשתנה. נבחר שם משתנה שייצג את תפקיד המנוע: למשל myservo ואז שורת הגדרת המשתנה תהיה:

```
Servo myservo;
```

מרגע שיצרנו את המשתנה, כל הפונקציות יהיו זמינות על ידי כתיבת שם המשתנה, נקודה, ושם הפונקציה.

הפונקציה הראשונה שיש להפעיל נקראת attach. פונקציה זו מקבילה במשמעותה לפונקציה pinMode שקיימת ברכיבים אחרים והיא מודיעה למיקרו בקר שיש מנוע סרוו שמחובר לפין שמספרו רשום בין הסוגריים (בדוגמה כאן פין 9)

```
myservo.attach(9);
```

ללא קריאה לפונקציה זו המנוע לא יעבוד בכלל.

עכשיו ניתן להפעיל את המנוע על ידי כתיבת של ערכי זווית רצויה בעזרת פקודת write. הפקודה מקבלת פרמטר אחד של זווית בין הסוגריים, שערכו בין 0-180 מעלות. ישנם מנועים שמסוגלים להגיע ליותר מ-180 מעלות ואז ניתן לכתוב ערך זווית גדול יותר

```
myservo.write(47);
```

ניתן לבצע מספר פעולות כתיבה ברצף כדי ליצור תנועה מסויימת אבל יש לתת השהייה בין כל שתי פקודות מכוון שהתנועה המכנית של המנוע יותר איטית מקצב הריצה של בקר ארדואינו, וכדי להימנע ממצב בו שיגרנו הרבה פקודות תנועה למנוע אבל הוא לא עומד בקצב ולכן יש לתכנן את התיזמון בהתאמה.

ד.1 תוכניות דוגמה למיקרו בקר ארדואינו

החומרה שנשתמש בה נראית בטבלה הבאה המתארת את הדקי הארדואינו והדקי מנוע הסרוו :

Servo Motor	Arduino
5V	5V
GND	GND
Control	9

טבלה 2 : חיבור הדקי כרטיס הארדואינו אל מנוע הסרוו.

ד.2 התוכנה

שליטה במנוע סרוו אינה משימה פשוטה אבל הארדואינו IDE כבר כולל ספרייה בשם סרוו. הוא מכיל פקודות פשוטות שניתן להשתמש בהן כדי להורות במהירות למנוע הסרוו להסתובב לזווית מסוימת.

בסביבת הפיתוח של ארדואינו ניתן למצוא שתי תוכניות דוגמא בתפריט : **קובץ->דוגמאות->SERVO.**

דוגמא אחת נקראת sweep שגורמת למנוע סרוו לנוע במחזוריות מ-0 מעלות ל-180 וחזרה.

דוגמא שניה נקראת knob והיא משלבת מנוע סרוו ופוטנציומטר. התוכנית קוראת ערך מהפוטנציומטר, ממפה אותו לתחום 0 עד 180 מעלות ומניעה את המנוע לזווית בהתאם.

ד.2.א תוכנית ראשונה

בתוכנית ה sweep.ino חיברנו את הפקודה למנוע הסרוו להדק 9 של הארדואינו.

תחילה נראה את התוכנית ולאחר מכן נסביר אותה. התוכנית נראית כך :

```
1/* Sweep
2by BARRAGAN
3This example code is in the public domain.
5modified 8 Nov 2013
6by Scott Fitzgerald
7http://www.arduino.cc/en/Tutorial/Sweep
8*/
9
10 #include <Servo.h>
11
12 Servo myservo;   // create servo object to control a servo
13 // twelve servo objects can be created on most boards
14
15 int pos = 0;              // variable to store the servo position
16
17 void setup() {
18 myservo.attach(9);   // attaches the servo on pin 9 to the servo object
19 }
20
21 void loop() {
22 for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
23 // in steps of 1 degree
24 myservo.write(pos);                    // tell servo to go to position in variable 'pos'
25 delay(15);                            // waits 15ms for the servo to reach the position
26 }
27 for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
```

```
myservo.write(pos);          // tell servo to go to position in variable 'pos'
delay(15);                   // waits 15ms for the servo to reach the position
}
}
```

ד.2.ב הסבר התוכנית

בתחילת הסקיצה אנו כוללים את הספרייה Servo.h :

```
##include <Servo.h>
```

השורה הבאה יוצרת אובייקט של סרוו :

```
Servo myservo;
```

ניתן להגדיר מספר אובייקטים של סרוו (תלוי כמה הדקים PWM יש בכרטיס הארדואינו שעובדים איתו) :

לדוגמה, אם יש לנו שני מנועי סרוו נכתוב :

```
Servo myservo1;
Servo myservo2;
```

המשתנה pos מראה לנו את הזווית אליה רוצים להגיע . בהתחלה הזווית היא 0 .

```
int pos = 0;
```

בפונקציית ה setup ( ) עושים קישור לאובייקט ה myservo להדק הבקרה של מנוע הסרוו שלנו בעזרת הפקודה attach

```
myservo.attach(9);
```

בפונקציית ה loop( ) יש שתי לולאות for . הלולאה הראשונה תסובב את המנוע בכיוון אחד, ואילו השנייה תסובב אותו בכיוון ההפוך. הפונקציה/מתודה myservo.write(pos) אומרת לשלוח פקודת הזזה למנוע הסרוו לזווית שנמצאת במשתנה pos :

```
myservo.write(pos);
```

בין כל מעבר אל הזווית הבאה עושים השהייה שך 15 מילי שניות כדי שהמנוע יספיק לבצע את הזזת המנוע לזווית הרצויה :

```
delay(15);
```

הערה חשובה : לפעמים אם נפעיל את המנוע ישירות מכרטיס הארדואינו נקבל כבר בהתחלת הרצת התוכנית RESET . הסיבה לכך היא שמנוע הסרוו צורך כמות משמעותית של זרם במיוחד במהלך ההפעלה הראשונה שלו , דבר שיכול לגרום ל RESET – איפוס – של כרטיס הארדואינו כי צריכת הזרם הראשונית "מפילה" את מתח ה 5 וולט שמגיע להפעלת ג'וק המיקרו בקר בכרטיס הארדואינו .

כדי לפתור את הבעיה נשים קבל אלקטרוליטי גדול יחסית (בין 470µF ל 1000µF) בין הדקי הכניסה של ספק הכוח. יש לשים לב לחבר את הקבל בקוטביות הנכונה.

הקבל מאחסן מטען חשמלי, כך שכאשר המנוע מתניע הוא "שואב" כוח הן מאספקת המתח של הארדואינו והן מהקבל, ומבטיח זרימה חלקה של זרם.

ד.3 התוכנית - knob.ino

בתוכנית ה knob.ino חיברנו את הפקודה למנוע הסרוו להדק 13 של הארדואינו ואת הזחלן של הפוטנציומטר חיברו אל רגל A0 (הדק כניסה אנלוגית מספר 0) של הארדואינו . בדוגמא כאן התוכנית קוראת ערך מהפוטנציומטר, ממפה אותו לתחום שבין 0 עד 180 מעלות בעזרת הפונקציה ( )map המעבירה את הערך הנקרא מהפוטנציומטר שהוא בין 0 ל 1023 לערך בין 0 ל 180 ומניעה את המנוע לזווית בהתאם.

התוכנית נראית כך :

```
1   /*
2     Controlling a servo position using a potentiometer (variable resistor)
3     by Michal Rinott <http://people.interaction-ivrea.it/m.rinott> 4
5     modified on 8 Nov 2013
6     by Scott Fitzgerald
7     http://www.arduino.cc/en/Tutorial/Knob
8   */
9
10  #include <Servo.h>
11
12  Servo myservo;  // create servo object to control a servo
13
14  int potpin = A0;  // analog pin used to connect the potentiometer
15  int val;        // variable to read the value from the analog pin A0
17  void setup() {
18    myservo.attach(13);  // attaches the servo on pin 13 to the servo object
19  }
20
21  void loop() {
22    val = analogRead(potpin);  // reads the value of the potentiometer (value between 0 and 1023)
23    val = map(val, 0, 1023, 0, 180);// scale it to use it with the servo (value between 0 and 180)
24    myservo.write(val);    // sets the servo position according to the scaled value
25    delay(15);                                  // waits for the servo to get there
```

```
26      }
```

ד.4 עבודה עם מנוע סרוו ומיקרו בקר ESP32

העבודה עם מיקרו בקר ESP32 בסביבת AEDUINO IDE דומה מאוד לעבודה עם כרטיס מיקרו בקר ארדואינו (אונו , נאנו , מגה וכו') . התוכניות כמעט זהות לתוכניות עבור הארדואינו.

הערה חשובה : היות יש לחבר את קו המתח של מנוע הסרוו אל מתח שבין 4.8 עד 6 וולט ולא את מתח ה 3.3 וולט שבכרטיס.

ד.4.א התוכנית sweep.ino

הנקראת sweep.ino שגורמת למנוע סרוו לנוע במחזוריות אין סופית מ-0 מעלות ל-180 מעלות וחזרה ל 0 . אנחנו חיברנו את הפקודה למנוע לרגל 18 של ה ESP32 שהוא אחד מהדקי ה PWM בכרטיס. כמו כן החלפנו את הספרייה Servo.h בספרייה ESP32Servo.h .

התוכנית נראית כך :

```
/* Sweep
 by BARRAGAN <http://barraganstudio.com>
 This example code is in the public domain.

 modified 8 Nov 2013
 by Scott Fitzgerald

 modified for the ESP32 on March 2017
 by John Bennett

 see http://www.arduino.cc/en/Tutorial/Sweep for a description of the original code

 * Different servos require different pulse widths to vary servo angle, but the range is
 * an approximately 500-2500 microsecond pulse every 20ms (50Hz). In general, hobbyist
servos
 * sweep 180 degrees, so the lowest number in the published range for a particular servo
 * represents an angle of 0 degrees, the middle of the range represents 90 degrees, and the top
 * of the range represents 180 degrees. So for example, if the range is 1000us to 2000us,
```

```
 * 1000us would equal an angle of 0, 1500us would equal 90 degrees, and 2000us would equal
1800
 * degrees.
 *
 * Circuit: (using an ESP32 Thing from Sparkfun)
 * Servo motors have three wires: power, ground, and signal. The power wire is typically red,
 * the ground wire is typically black or brown, and the signal wire is typically yellow,
 * orange or white. Since the ESP32 can supply limited current at only 3.3V, and servos draw
 * considerable power, we will connect servo power to the VBat pin of the ESP32 (located
 * near the USB connector). THIS IS ONLY APPROPRIATE FOR SMALL SERVOS.
 *
 * We could also connect servo power to a separate external
 * power source (as long as we connect all of the grounds (ESP32, servo, and external power).
 * In this example, we just connect ESP32 ground to servo ground. The servo signal pins
 * connect to any available GPIO pins on the ESP32 (in this example, we use pin 18.
 *
 * In this example, we assume a Tower Pro MG995 large servo connected to an external power
source.
 * The published min and max for this servo is 1000 and 2000, respectively, so the defaults are
fine.
 * These values actually drive the servos a little past 0 and 180, so
 * if you are particular, adjust the min and max values to match your needs.
 */

#include <ESP32_Servo.h>

Servo myservo;  // create servo object to control a servo
                // 16 servo objects can be created on the ESP32

int pos = 0;    // variable to store the servo position
// Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-27,32-33
int servoPin = 18;

void setup() {
```

```
  myservo.attach(servoPin);   // attaches the servo on pin 18 to the servo object
                              // using default min/max of 1000us and 2000us
                              // different servos may require different min/max settings
                              // for an accurate 0 to 180 sweep
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

ד.4.ב התוכנית knob.ino

גם כאן התוכנית דומה מאוד לזו של כרטיס ארדואינו. אנחנו חיברנו את הפקודה למנוע לרגל 18 של ה ESP32 שהוא אחד מהדקי ה PWM בכרטיס. את זחלן הפוטנציומטר חיברנו לרגל 34 . כמו כן החלפנו את הספרייה Servo.h בספרייה ESP32Servo.h .

```
/*
 Controlling a servo position using a potentiometer (variable resistor)
 by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

 modified on 8 Nov 2013
 by Scott Fitzgerald

 modified for the ESP32 on March 2017
 by John Bennett

 see  http://www.arduino.cc/en/Tutorial/Knob for a description of the original code
```

```
 * Different servos require different pulse widths to vary servo angle, but the range is
 * an approximately 500-2500 microsecond pulse every 20ms (50Hz). In general, hobbyist servos
 * sweep 180 degrees, so the lowest number in the published range for a particular servo
 * represents an angle of 0 degrees, the middle of the range represents 90 degrees, and the top
 * of the range represents 180 degrees. So for example, if the range is 1000us to 2000us,
 * 1000us would equal an angle of 0, 1500us would equal 90 degrees, and 2000us would equal
1800
 * degrees.
 *
 * Circuit: (using an ESP32 Thing from Sparkfun)
 * Servo motors have three wires: power, ground, and signal. The power wire is typically red,
 * the ground wire is typically black or brown, and the signal wire is typically yellow,
 * orange or white. Since the ESP32 can supply limited current at only 3.3V, and servos draw
 * considerable power, we will connect servo power to the VBat pin of the ESP32 (located
 * near the USB connector). THIS IS ONLY APPROPRIATE FOR SMALL SERVOS.
 *
 * We could also connect servo power to a separate external
 * power source (as long as we connect all of the grounds (ESP32, servo, and external power).
 * In this example, we just connect ESP32 ground to servo ground. The servo signal pins
 * connect to any available GPIO pins on the ESP32 (in this example, we use pin 18.
 *
 * In this example, we assume a Tower Pro SG90 small servo connected to VBat.
 * The published min and max for this servo are 500 and 2400, respectively.
 * These values actually drive the servos a little past 0 and 180, so
 * if you are particular, adjust the min and max values to match your needs.
 */

// Include the ESP32 Arduino Servo Library instead of the original Arduino Servo Library
#include <ESP32_Servo.h>

Servo myservo;  // create servo object to control a servo
```

```
// Possible PWM GPIO pins on the ESP32: 0(used by on-board button),2,4,5(used by on-board
LED),12-19,21-23,25-27,32-33
int servoPin = 18;      // GPIO pin used to connect the servo control (digital out)
// Possible ADC pins on the ESP32: 0,2,4,12-15,32-39; 34-39 are recommended for analog input
int potPin = 34;        // GPIO pin used to connect the potentiometer (analog in)
int ADC_Max = 4096;     // This is the default ADC max value on the ESP32 (12 bit ADC
width);
                        // this width can be set (in low-level oode) from 9-12 bits, for a
                        // a range of max values of 512-4096

int val;    // variable to read the value from the analog pin

void setup()
{
  myservo.attach(servoPin, 500, 2400);   // attaches the servo on pin 18 to the servo object
                                         // using SG90 servo min/max of 500us and 2400us
                                         // for MG995 large servo, use 1000us and 2000us,
                                         // which are the defaults, so this line could be
                                         // "myservo.attach(servoPin);"
}

void loop() {
  val = analogRead(potPin);          // read the value of the potentiometer (value between 0 and
1023)
  val = map(val, 0, ADC_Max, 0, 180);     // scale it to use it with the servo (value between 0 and
180)
  myservo.write(val);                // set the servo position according to the scaled value
  delay(200);                        // wait for the servo to get there
}
```

ה. ביבליוגרפיה

1. https://lastminuteengineers.com/servo-motor-arduino-tutorial/ האתר
2. https://docs.arduino.cc/learn/electronics/servo-motors/
3. https://store.arduino.cc/products/feetech-mini-servo-motor-120-degrees-9g
4. https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/