

**מגישות:**

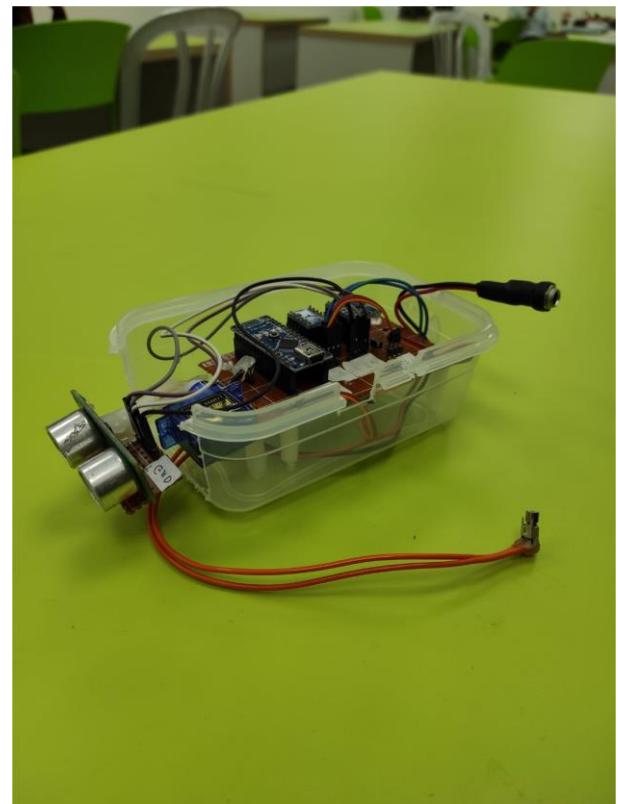
שרון גולדשטייד ת.ז : 212118731  
תמי פרידמן-יברוב ת.ז: 322366717

**מנחים:**

מוטי מאיר, גיאורגי שריבמן

**מועד לימודים:**

'אופק'



# מקל עווירים אלקטרוני

פרויקט גמר בмагמת אלקטרוניקה ומחשבים

## תוכן עניינים

2 .....	תודות.....
3 .....	תיאור הפרויקט .....
4 .....	תרשימים מלבנים:.....
5 .....	شرطות חשמלי.....
6 .....	פירוט רכיבים:.....
6 .....	ארדואינו UNO:.....
9 .....	מנוע סרו.....
9 .....	אפנון רוחב פולס (דופק) :PWM - Pulse With Modulation .....
10 .....	המנוע:.....
14 .....	ממיר מתח 5 וולט ל 3 וולט:.....
16 .....	היישן התאוצה :MPU6050 .....
16 .....	רגיסטרים:.....
21 .....	מסנן משלים – COMPLEMENTARY FILTER .....
22 .....	תקשורת I2C .....
25 .....	מנוע רטט .....
26 .....	התנסות עם הרכיבים:.....
26 .....	ניסוי 1: מנוע סרו.....
27 .....	ניסוי 2: היישן מרחק.....
28 .....	ניסוי 3: שימוש ברכיב PCF8574 עבור תקשורת i2c .....
31 .....	ניסוי 4: היישן MPU6050- מד מהירות זוויתית ותאוצה .....
33 .....	תיעוד בניית הפרויקט:.....
33 .....	כתיבת התכנית:.....
33 .....	תיעוד תהליך .....
37 .....	התוכנית:.....
40 .....	בנייה המנגנון:.....
41 .....	מסקנות וסיכום .....
42 .....	דף נתונים:.....
65 .....	נספחים .....
68 .....	מקורות .....

**תודות**

בזהzmanות זו רצינו להזכיר תודה לכל המסייעים והאחראים להצלחתנו בبنית הפרויקט, שננתנו מזמן ומכוון ופעלו כל שביכולתם ועמדנו לצידנו לאורך כל הדרך

ראשית, נבע את תודתנו העמוקה למנהל מוטי מאיר, שהיה לנו לשענת בכל שלב ושלב. סיע ויעץ מתווך נסיוונו הרב וחכמתו הרבה, מעולם לא הגביל אותנו ברעיונות, העמיד לרשותנו את כל המשאבים הדרושים לתוכנית ללא כל היסוס. ובמיוחד בשנה מתガרת זו דאג תוק שמירה מוחלטת על המגבילות לדרבן ולעוזד אותנו למקסם את הפרויקט ויחד עם זאת להנות מהדרך, הייתה לנו הזכות ללמידה ולהשכיל לאורו.

שניית, נודה למנהל המלווה גאורגי שריבמן, שליווה אותנו, ועזר בכל שאלה, בעיה או תהיה.

נרצה להודות גם כן להוריינו היקרים שהביאונו עד הלום, שתמכו ועוזדו אותנו תמיד לייצוריות חשיבה מחוץ לקופסה והשכלה גבוהה לצד ערכים ומסגרת תומכת.

לו שעומדת מאחורי המქם המדהים "סמינר אפק"-המנהל היקраה שדואגת לנו הבנות שתהייה אוירה נעימה ובמאור פנים מעודדת לשאוף ולהגיע להישגים גבוהים וכן מעניקה לנו סביבת עבודה מרשים עם מעבדות וציוד משוכללים.

אנו מודות לכל העומדים לצינו, תודה רבה על הסיע ועל העזרה שהענתקתם.

## תיאור הפרויקט

לצערנו לכולנו יצא לראות לא פעם ולא פעמיים אדם הלוקה בקוצר ראייה או עיוורון הנעדר במקל, אחד מן הקשיים הבולטים ביותר הוא הקושי של אף השימוש במקל – העיוור תלוי בגורמים נוספים והוא לא יכול להתניע בעצמאות מוחלטת

לכן רצינו ליעיל את המקל המוכר לכולנו באמצעות עזרים אלקטرونים ועוד. כך קם לו רעיון הפרויקט – בניית דגם אלקטרוני של מקל נחייה לעווירים.

הפרויקט מורכב מלווח ארדואינו – לב המערכת, מיקרו בקר עליון נऋבת התוכנה וממנו הפקודות לשיליטה במערכת כולה. למקל חישן מרחק אשר ימוקם מლפנים וימדוד את המרחק מהעצם מולו.

כדי שחיישן המרחק יעמוד תמיד בצורה אופקית לקרקע השתמשנו במד תואזה ובמד מהירות זוויתית (אקסלורומטר + ג'ירו) שתפקידם לחשב את הזווית המקל ביחס לקרקע. את חישן המרחק הרכבנו על גבי מנוע סרוו. התוכנית שכתבנו גורמת למנוע לכוון את החישן תמיד בזווית אופקית לקרקע על אף שהעיוור מטה את המקל בزواיות שונות. כך הבחחנו שהחישן יעמוד מול המכשולים ולא יפספס אותם.

המקל יתריע מפני מכשולים שנמצאים מLfנים. בנקודת האחיזה של המקל ימוקם מנוע רטט בעיר כדי שהעיוור יוכל אינדקציה של המרחקים מהמכשולים. ככל שהמרחק מהאובייקט יקטן כך קצב הריטוטים יגדל.

**תרשים מבנים:****הסבר:**

ארדוינו – לב המערכת, מיקרו בקר עלי נציבות התוכנה וממנו הפקודות לשיטה במערכת כולה

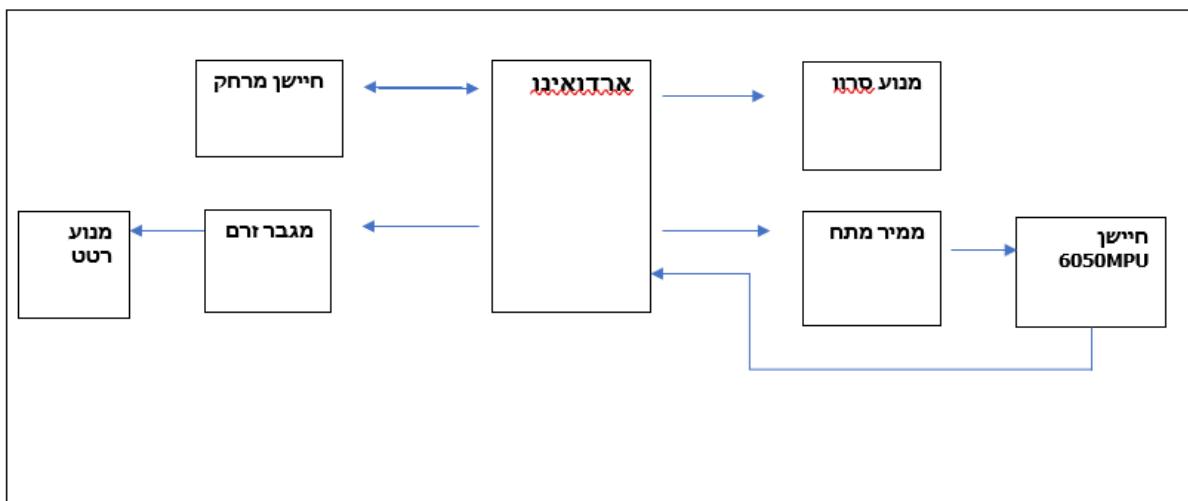
ממיר מתח – משמש להמרת המתח 5V המתתקבל מהבקר ל 3V עבור רכיב הגירוי.

חיישן 6050MPU – מודד מהירות זוויתית ותואוצה, מורכב מאוקיולומטר וגירוסקופ. תפקידיו לתת אינדיקציה על זוית המקל ביחס לקרקע

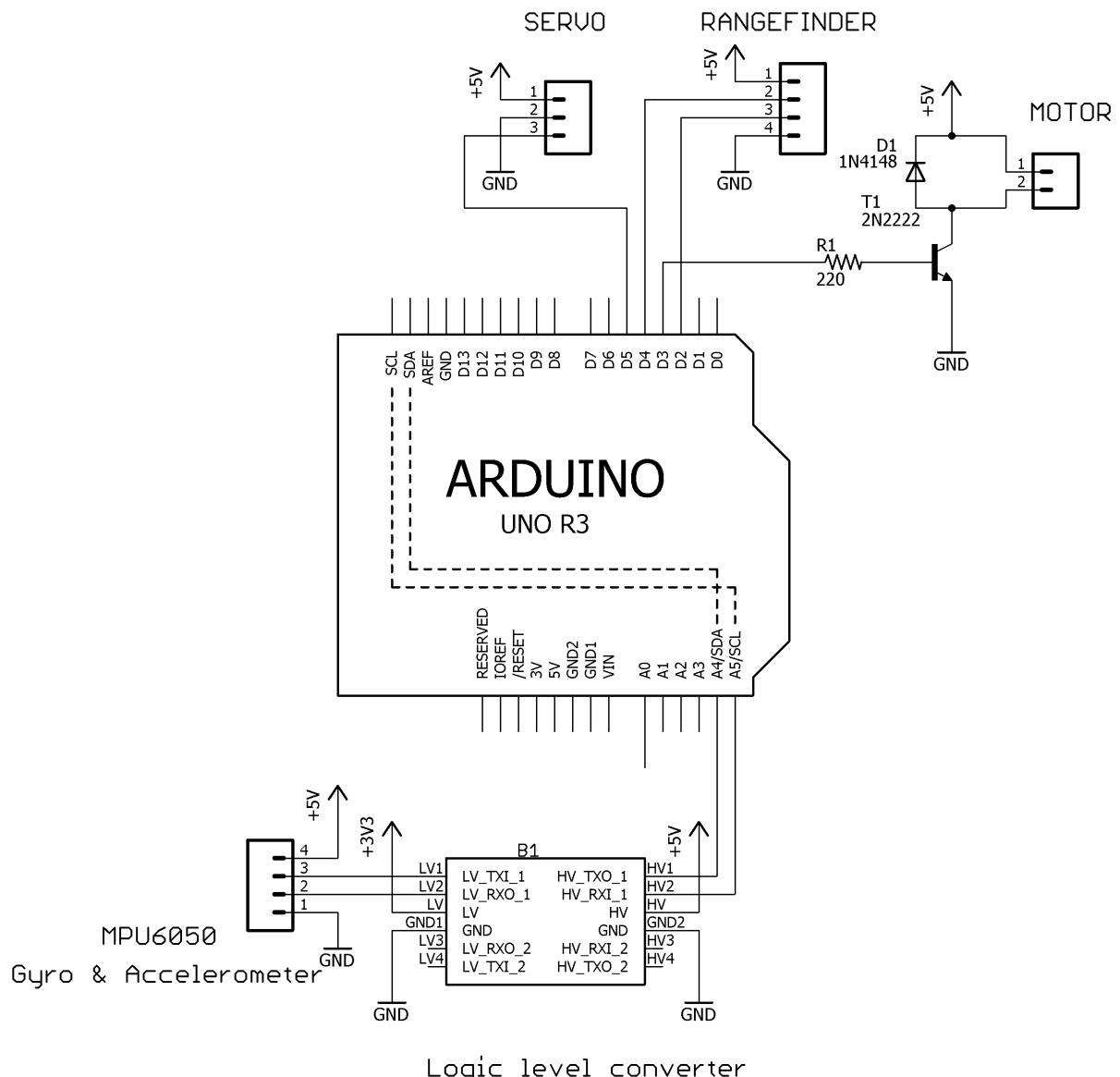
מנוע סרו – משמש לסייע חיישן המרחק לפי המיקום הזוויתי של המקל ביחס לקרקע  
חיישן מרחק – פועל על עיקרון התפשטות גלי הקול למרחב. תפקידו לזיהות עצמים במסלול ההליכה של העוור

מגבר זרם – טרנזיסטור שתפקידו לספק זרם למנוע הרטט

מנוע רטט – תפקידו לתת הטראה לעוור על המכשול שלפניו, בקצב התלי במרחב



شرطוט חשמלי



פירוט רכיבים:

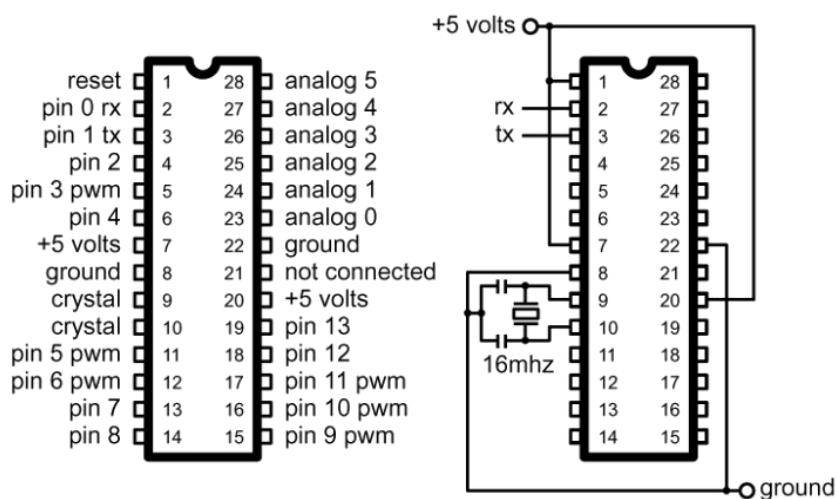
ארדואינו UNO:

ארדואינו (arduino) הוא כרטיס המכיל 'מיקרובייר' של חברת ATMAIL. Atmega328P הוא שבב שוכן בArduino, המבוסס על המיקרובייר. לארדואינו סביבת פיתוח נוחה ופешטה לפיתוח פרויקטים המשלבים תוכנה עם חומרה (אלקטרונית). לאחר ומודבר ב מוצר פיזי יש לכתוב את התוכנה בסביבת הפיתוח ואז לצרוב אותה מהמחשב לארדואינו. לוח הפיתוח של הארדואינו ניתן לחבר רכיבי קלט (mpsks), חיישנים, וכו') ורכיבי פלט (לדים, מנועים, וכו')

#### מאפייני הארדואינו:

- תדר שעון Z.16MHz.
- מתח עבודה 5V. תחום המתוח הוא 6-1.8 וולט
- אפשרות חיבור לאספקת מתח, מומלץ בין 9-7, כאשר התחום האפשרי הוא -6v 20v
- זרם בהדק I/O עד A.40mA.
- ذיכרון תוכנית (flash) בגודל 32k.
- זיכרון נתונים (ram) בגודל 2k.
- זיכרון נתונים נוספים EEPROM בגודל 1K בתים
- 14 כניסות ויציאות דיגיטליות, מתווך 6 יציאות PWM
- 6 כניסות אנלוגיות ברזולוציה של 10 סיביות.
- תקשורת טורית (rs232 , i2c, spi).
- 2 פסיקות חיצוניות.

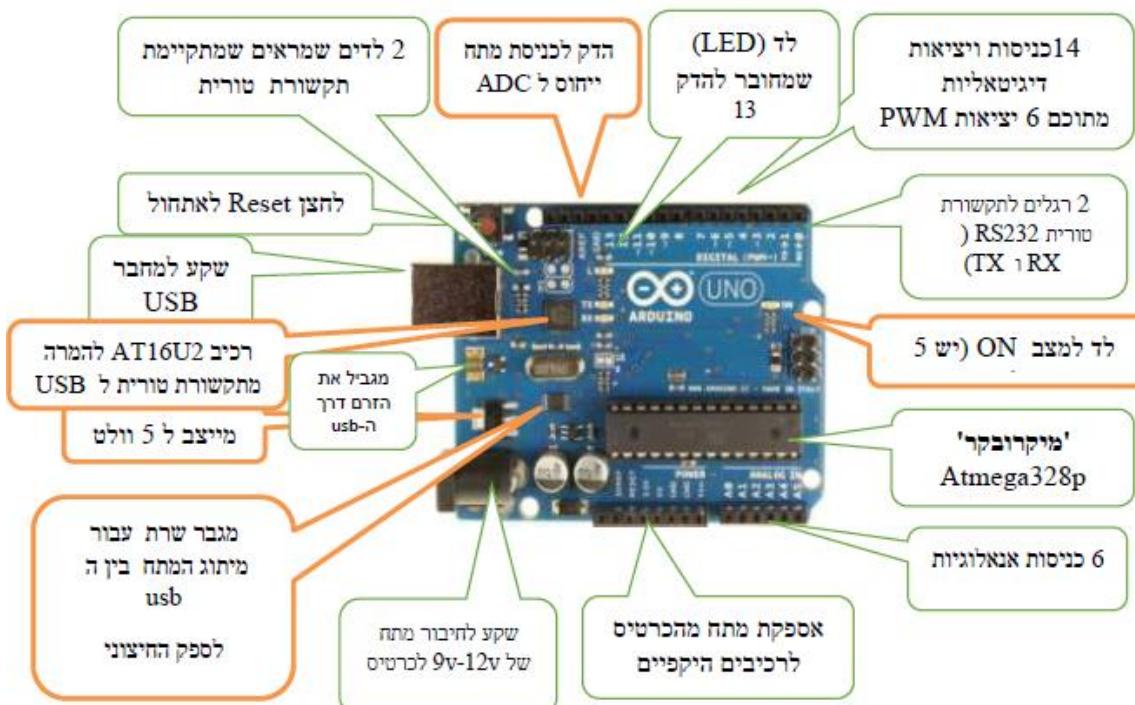
#### הדקmicro בקר:



### תפקיד הדקים בכרטיס ארדואינו המבוסס על הרכיב ATmega328 :

- TX, RX – שני הדקים טוריים המשמשים לקליטה ושידור טוריים.
- הדקים 2 ו 3 - הדקי פסיקות חיצונית. נוכל לתכנת הדקים אלו להפעלת פסיקה בrama נמוכה, עלייה, ירידה ושינוי רמת מתח. כאשר נרצה להשתמש בהדקים אלו נעזר בפונקציה attachInterrupt(interrupt, function, mode)
- הדקים 11, 10, 9, 8 – PWM (אפנון רוחב דופק). כאשר נרצה להשתמש בהדקים אלו בPWM ניעזר בפונקציה digitalWrite(pin, mode).
- הדק 13 – זהה הדק המחבר לד'
- לארדואינו אונו 6 כניסה אנלוגיות המסומנות A0 עד A5 כשמייל כניסה ניתן לקבל רזולוציה (כשר הבדיקה) של 10 ביט לעומת, הארדואינו יוכל לבדוק בין 1024 רמות מתח שונות.
- הדק 1 - זהה הדק האתחול (RESET), ב '0' ההדק יאטחל את הרגיסטרים הפנימיים, והמעגלים האלקטרוניים שבtower המיקרובקר במצב ידוע ומוגדר.
- הדק 21- לא מחובר.
- הדקים 28-23: הדקי כניסה אנלוגיים.
- הדקים 6-14: אלו הדקים המשמשים כקלט/פלט דיגיטליים.
- הדקים 9, 10: בין הדקים אלו מתחבר גביש של 16MHz. הגביש משמש כمعالג תへודה, ויחד עם המגבר הנמצא בתוך המיקרו בקר, נוצר מתנד שיזכר את דפקי השעון המפעלים את המיקרו בקר וקובעים את קצב העבודה שלו.

### **סיכום מבנה הכרטיס:**



אנחנו השתמשנו בארדואינו ננו מכיוון שלא היה צריך בכל קר הרבה הדקים לפרויקט שבחרנו.



#### פיתוח תוכנית:

כל תוכנית בארדואינו מכילה לפחות את 2 הפקציות הבאות-

1. **setup()** - בה כתבים הגדרות ו/או הוראות שהמיקרובוקר יבצע באופן חד פעמי.
2. **loop()** - בה כתבים הוראות שהמיקרובוקר יבצע בלולאה אינסופית

## מנוע סרו

:PWM - Pulse With Modulation (דופק)

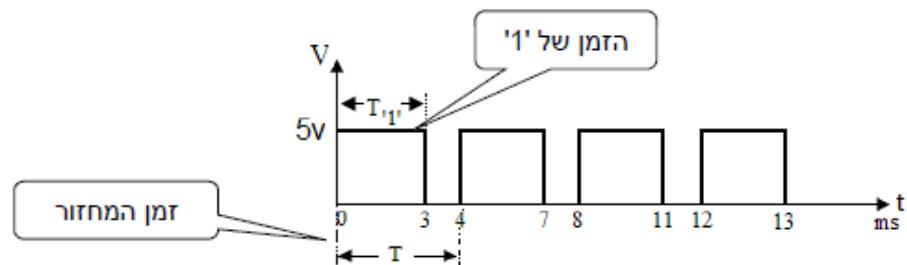
כידוע, הבקר 'ambil' רק אותות דיגיטליים של '0' או '1'. כלומר, שני מצבים מתח בלבד. 0v או 5v

לעתים אנו חייבים לספק מתח שהוא שונה מ-5v. אחת הדרכים לעשות זאת היא באמצעות שיטת ה PWM.

בשיטת זו הבקר אמם מתח קבוע (5v) אבל בתדר מסוים. כדי לשנות את המתח הממוצע משנים את **'גורם המחזור'** של התדר.

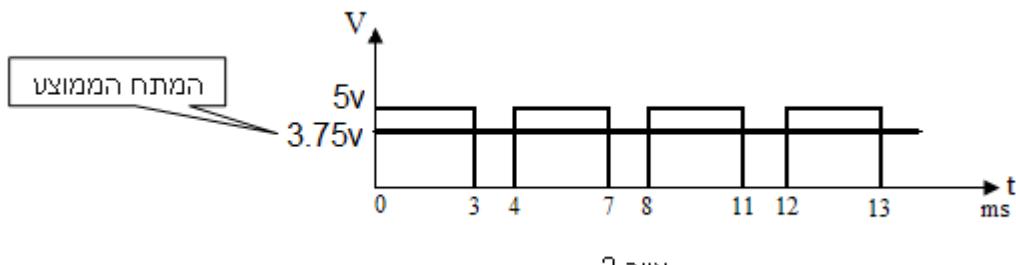
**'גורם המחזור'** (Duty Cycle) מוגדר כיחס שבין הזמן בו האות נמצאת ב'1' לוגי, לבין הזמן המחזור.

דוגמא:



זמן המחזור הוא  $T=4\text{ms}$ . הזמן שבו האות נמצאת ב'1' הוא  $3\text{ms}$ . לכן 'גורם המחזור' הוא:  $\frac{3\text{ms}}{4\text{ms}} = 0.75$ . המתח הממוצע שנקבל הוא:  $v_{\text{ ממוצע}} = 5\text{v} \cdot 0.75 = 3.75\text{v}$ .

אספקת תדר עם מתח של 5v וגורם מחזור של 0.75 היא שווה ערך לאספקת אנרגיה חשמלית של 3.75v

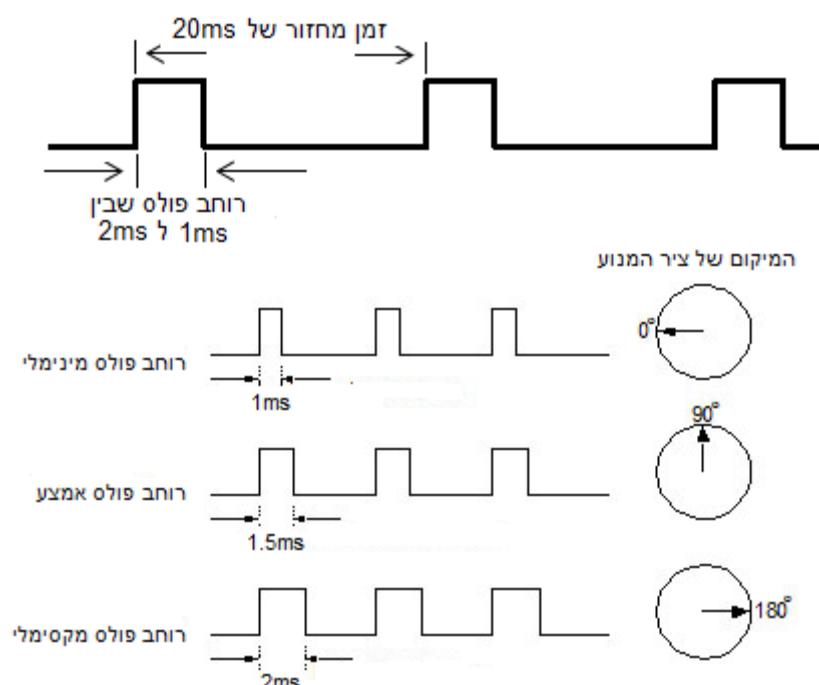


הfonkציית `analogWrite(pin, value)` מוציאה להדק הנבחר תדר של 490hz עם 'גורם המחזור' שהוא ערך שבין 0 ל 255 (מתוך 255).

המנוע:  
בפרויקט שלנו המנוע מסובב חישון המרחק

מנוע סרוו הוא מנוע זרם ישיר (DC motor) בעל מערכת תמסורת פנימית של גלגלי שיניים המהווים תמסורת הפעטה (ማיטה את מהירות הסיבוב ומגדילה את מומנט הכוח), ובקרה אלקטронית פנימית על מיקום המנוע. מנוע סרוו עובדים בטוחה שונה של מתחים, בדרך כלל בין 4.8 ל- 6 וולט.

למנוע יש 3 חוטים. שניים לאספקת מתח (+/-) וחוט נוסף לבקרה  
בקרת המנוע מנעה את ציר המנוע לזרoit מדויקת בהתאם לרוחב הפולס המתתקבל בקנו הבקרה. בנוסף, הוא בעל יכולת תיקון פערים מהמיקום הרצוי.  
בד"כ רוחב הפולס נע בין רוחב פולס מינימלי של 1ms (שמייצג זווית של 0 מעלות) לרוחב פולס מקסימלי של 2ms (שמייצג 180 מעלות):

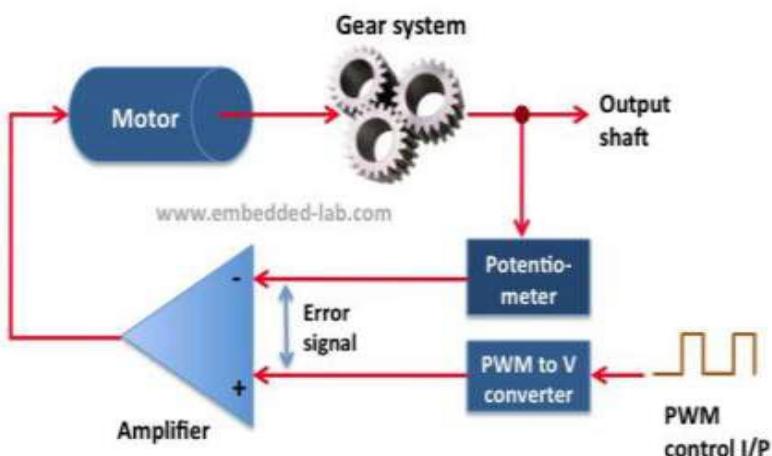


### עיקרונות הפעולה של מנוע הסרו:

המנגנון האלקטרוני מכיל משווה. להדק המבואה (-) מחובר פוטנציאומטר המחבר לציר המנוע. סיבוב הפוטנציאומטר גורם לשינוי ההתנגדות הנגדית ובהתאם לשינוי מפל המתח הנמדד על פניו, ככלומר המתח עליו תלוי בויתת של ציר המנוע.

להדק המבואה (+) מחובר הגל הריבועי של PWM שעובר המרה לDC.

כל עוד יש הבדל בין המתח המתkeletal מהגל גבוה והמתוך המתkeletal מהפוטנציאומטר – המנוע יסתובב בכיוון המתאים עד שהפרש המבאות ישתווה. עצירת המנוע תהיה בדיק בזווית שנדרשה בתוכנית הבקר.



### הספריה Servo.h

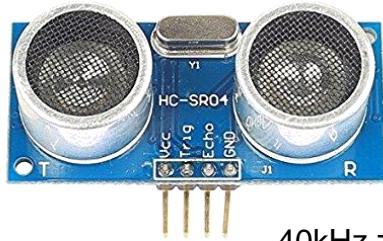
מהפונקציות שבספרייה:

- קביעת הדק בקרלה: `myservo.attach(pin)`
- יצירת PWM בマイקו שניות: `myservo.writeMicroseconds(number)`
- פונקציה המקבלת זווית בין 0-180 ושולחת אותה למנוע כPWM  
`myservo.write(number)`

### חישון מרחק אולטרא סוני Ultra-Sonic SRF05

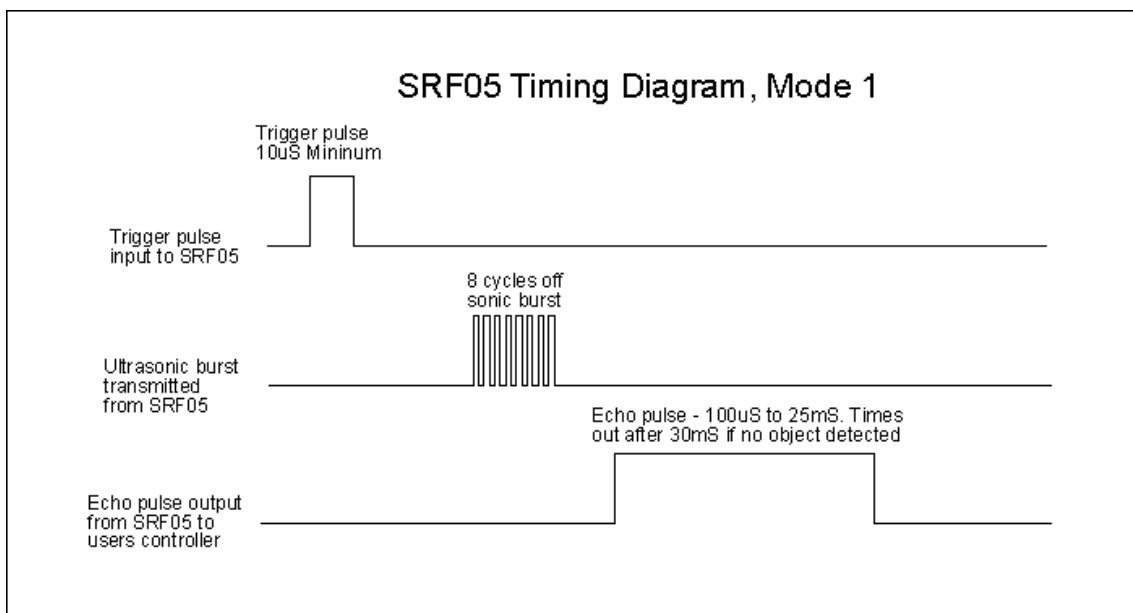
חישון ה Ultra-Sonic מძר ל送出 גלי קול בתדר (Ultra-Sonic) של  $40\text{kHz}$ . החישון נותן לנו את פרק הזמן שחלף מרגע שידור גלי הקול ועד לקבלת ההד בחרזה. ניתן לחשב את המרחק על פי הנוסחה:  $\text{מרחק} = \frac{\text{מהירות}}{2} \times \text{זמן}$ .

לחישון 4 הדקים. שני הדקים לאספקת מתח ( $5\text{V}$ , GND). הדק מבוא (Trigger) למטען 'דרבן' לחישון. והדק מוצא (Echo) לקבלת הזמן שחלף מרגע שידור גלי הקול ועד לקבלת ההד בחרזה.



#### התנהגות החישון:

- א. נתונים בהדק Trigger – פולס שרוחבו  $10\mu\text{s}$ .
- ב. בעקבות הפולס, החישון מבצע שני דברים:
  - i. מძר למרחב 8 מחזורים של גלי קול בתדר  $40\text{kHz}$ .
  - ii. מעלה את הדק Echo ל'1'.
- ג. כאשר ההד חוזר אל 'המקלט'. הרכיב מוריד את הדק ל'0'. עפ"י הגדרות היצرن אם גלי הקול לא יחרזו כעבור  $30\text{ms}$  הרכיב יוריד אותו בכל מקרה.



$$\text{כדי לחשב את המרחק, אנו משתמשים בנוסחת המהירות: } V = \frac{S}{t}$$

כאשר:  $V$  – מהירות.  $S$  – דרכ (ביחידות *meter*).  $t$  – זמן (ביחידות *second*).

המיקרובוקר מודד את הזמן במיילוניות השנייה (ב $\text{s}$ ). כדי לקבל את המרחק בס"מ נחלק את הזמן שהחישון מדד במספר 58.

**הסבר:**

דוע שמהירות גלי הקול (ב $24^{\circ}C$ ) היא:  $346 \text{ meter/second}$  קלומר זמן התפשטות (שיסומן ב $X$ ) עבר ס"מ אחד הוא:

$$X = \frac{2.89 \text{ mili second}}{\text{meter}} = \frac{28.9 \text{ micro second}}{\text{cm}}$$

$$S = \frac{t}{x} = \frac{t}{28.9} \quad \text{חישוב המרחק הוא לפ:}$$

$S$  - המרחק ב $\text{cm}$ .

$t$  - הזמן שנמדד (ע"י החישון) ב $\text{s}$ .

$X$  – זמן התפשטות של גלי הקול (ב $S$  לס"מ אחד).

אחר וגלי הקול עוברים דרך כפולה (מהחישון למשטח ובחזרה), יש לחלק את הזמן ב 2.

$$\text{קלומר: } S = \frac{t/2}{x} = \frac{t/2}{28.9}$$

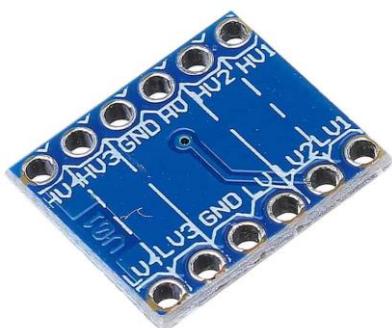
. ומקבלים  $28.9 \cdot 2 \cong 58$

ממיר מתח 5 וולט ל 3 וולט:

תקשורת C2I של הארדואינו פועלת בرمמת מתח של 5 וולט '1', ואילו הגיירו ומד התאוצה הזרותית פועלם בرمמת מתח של 3.3 וולט '1'.

הבעיות שועלות לקרות מכך הן שהארדואינו עלול לקרוא מתח 3.3 כ '1', והחישון עלול להינזק מהמתוך הגובה של 5.

לכן ע"מ ליצור תיאום בין הארדואינו לרכיב השתמשנו בממיר מתחים.

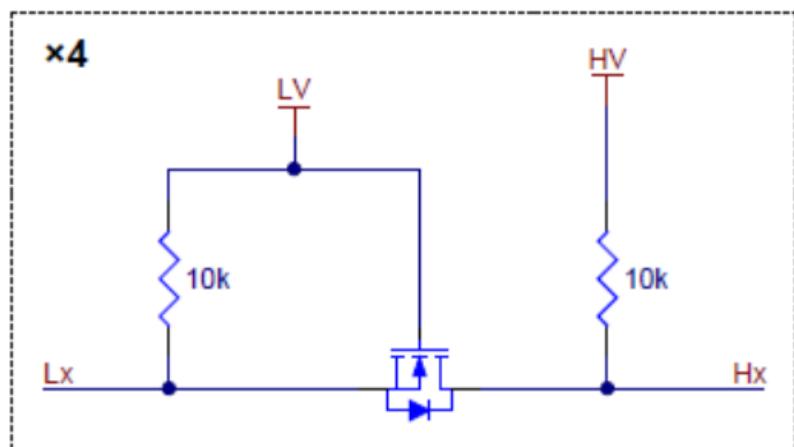


#### **מאפיינים חשמליים:**

- בין 1.5 ל 7 וולט עבור אספקת מתח נמוך
- עד 18 וולט עבור אספקת מתח גובה
- ארבעה ערוצים דו כיווניים

המיר מtabסס על טרנזיסטורי MOSFET ועל דיודות.

בטריזיסטור זה השדה החשמלי בסיס משפיע על התכונות שלו כמוליך, ובכך ניתן לשנות על הזרם הזורם דרכו. ניתן לראותו כמו מג.

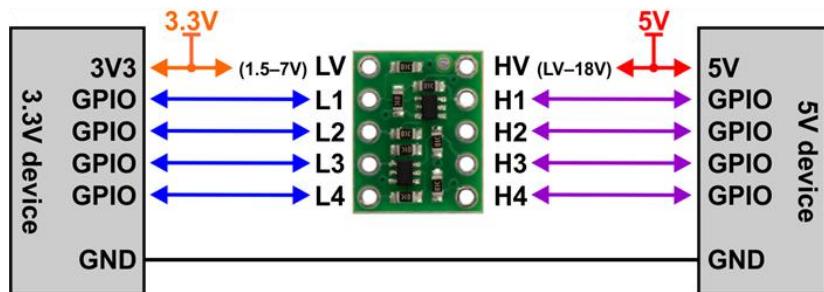


#### **קיימות 3 מצבים:**

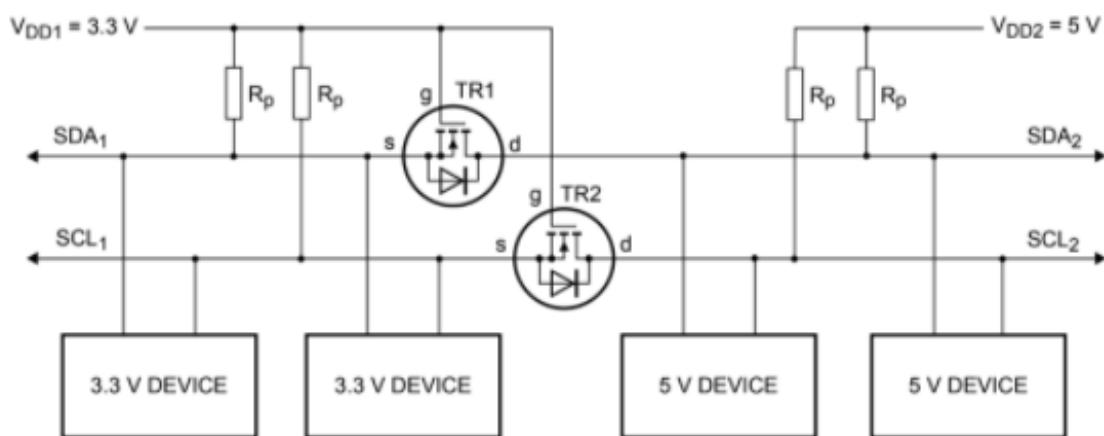
- הקו הנמוך במצב '0' – הטרנזיסטור מוליך אך שיש '0' גם בקו הגובה. הדiodה בנתך.
- הקו הגובה במצב '0' – הטרנזיסטור לא מוליך והדiodה מוליכה, ובגלל שmafל המתח שלה הוא כ 0.7 גם בקו הנמוך יהיה '0'. בשלב זה הטרנזיסטור יפעל.

- הקו הנמוך או הגבוה במצב '1' – הטרנסיזטור לא מוליך והדיודה בנתק, לכן לא יזרום זרם ובקו השני יהיה גם '1'.

### חיבורים:

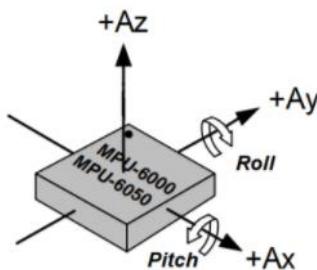


### חיבורים בפרויקט שלנו:



## חישון התאוצה:MPU6050

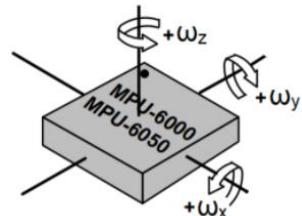
החישון MPU6050 מכיל מד תאוצה (accelerometer) ומד מהירות זוויתית (gyroscope). מד תאוצה: הרכיב כולל 3 חיישני תאוצה עבור 3 הצירים, ובעזרתם ניתן לחשב זווית ההטייה של הגוף, לפי הנוסחאות:



$$\tan \phi_{xyz} = \left( \frac{G_{py}}{G_{pz}} \right) - \text{עבור roll}$$

$$\tan \theta_{yxz} = \left( \frac{-G_{px}}{G_{pz}} \right) - \text{עבור pitch}$$

מד מהירות זוויתית: הרכיב כולל 3 חיישני גירוי עבור 3 הצירים.



כל חישון מדויק מאד, כי הוא מכיל 16 סיביות אנלוגיות של חומרה דיגיטלית בשבייל המרה עבור כל ערז. לשם כך הוא מקבל את הנתונים של X,Y,-Z בו זמן. החישון משתמש בתקורת I2C כדי להתmeshק עם הארדואינו.



בפרויקט שלנו השתמשנו ברכיב ע"מ לדיאוג לkr שחיישן המרחק יכוון בזווית מסוימת ביחס לקרע כל הזמן, למראות תזוזת המקל.

רגיסטרים:

**רגיסטרים 3B-40:**

ניתן לקרוא מהם את הנתונים הנמדדים. בתוכם נשמרות המדידות האחרונות של מדי התאוצה:

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59								ACCEL_XOUT[15:8]
3C	60								ACCEL_XOUT[7:0]
3D	61								ACCEL_YOUT[15:8]
3E	62								ACCEL_YOUT[7:0]
3F	63								ACCEL_ZOUT[15:8]
40	64								ACCEL_ZOUT[7:0]

**רגיסטרים 43-48:**

ניתן לקרוא מהם את הנתונים הנמדדים. בתוכם נשמרות המדידות האחרונות של מד' הזויה.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67								GYRO_XOUT[15:8]
44	68								GYRO_XOUT[7:0]
45	69								GYRO_YOUT[15:8]
46	70								GYRO_YOUT[7:0]
47	71								GYRO_ZOUT[15:8]
48	72								GYRO_ZOUT[7:0]

**שלבים לקריאת הנתונים:**

- הגדרת מצב צrichtת החשמל ואת מקור השעון. זו יבוצע ע"י פניה לרגיסטר B6 ואיפוא ערכו.
- כתיבת מצביע הזיכרון את כתובות רגיסטר הנתונים הראשון – B3 ממנוначילה לקרוא.
- קריאת תוכן הרגיסטרים, והמרת הנתונים לערכים שנוכל להשתמש בהם:
  - מד התאוצה- ע"י הנוסחאות הנ"ל
  - מד מהירות זוויתית- הנתון נמצא במספר דיגיטלי בן 16 סיביות, כלומר המקסימלי הוא  $2^{15}$ .

לא שינו את sel\_fs لكن לפי הטווח הערך המקסימלי שנקלט בגיירו יהיה 250.

נחלק  $2^{15}$  ב 250 ונקבל 131. לעומת זאת נרמלת את הנתון לזוית.

**רגיסטר 1B:**

סיביות 7-5 מאפשרות בדיקה עצמית של החלקים המכניים והחישומיים של חיישני הגyro. SEL\_FS : בוחר את טווח המידה המלא של רגישות יציאות הגיروسופ עבור מעקב מדויק אחר תנודות מהירות ואיטיות, עפ"י הטבלה:

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

כברירת מחדל הוא מוגדר כ 0.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]	-	-	-	-

**רגיסטר 6B:**

ברגייסטר זה נקבע את מצב הפעולה של ה-MPU6050.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]	-	-

כברירת מחדל כל הסיביות שוות '0'.

DEVICE\_RESET : כשרכו '1' ה-MPU מסofs את כל הנתונים לערכי ברירות המחדל.

SLEEP : כשרכו 1 ה-MPU נכנס למצב שינה בעל צריכה חשמלית נמוכה.

CYCLE : כאשר אנו שמים 1 או ה-SLEEP הוא 0 אז ה-MPU נכנס למצב מחזור. מעבר מחזור בין מצב שינה למצב פעיל.

TEMP\_DIS : מכבה את חיישני הtemp (לא שימושי בפרויקט שלנו)

CLK\_SEL : מכיל 3 ביטים ורcco קובע את מקור השעון. הוא יכול להיות שעון פנימי, ציר הגyro, או שעון חיצוני

CLKSEL	Clock Source
0	Internal 8MHz oscillator
1	PLL with X axis gyroscope reference
2	PLL with Y axis gyroscope reference
3	PLL with Z axis gyroscope reference
4	PLL with external 32.768kHz reference
5	PLL with external 19.2MHz reference
6	Reserved
7	Stops the clock and keeps the timing generator in reset

**רגיסטר C9:**

אפשר לכבות את קליטת צרי החישנים ע"מ לחסוך בצריכת החשמל של הרכיב ב 6 הסיביות הנמוכות- צרי X,Y,Z של חישן הזווית ב 2-0 ושל חישן התאוצה ב 5-3.

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6C	108	LP_WAKE_CTRL[1:0]	STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG	

השבנת צרי מזווית: אם אחד מהצירים של מזווית שימש כמקור של ה- CLK אז הוא חוזר לערכו כברירת מחדל (8MHZ).

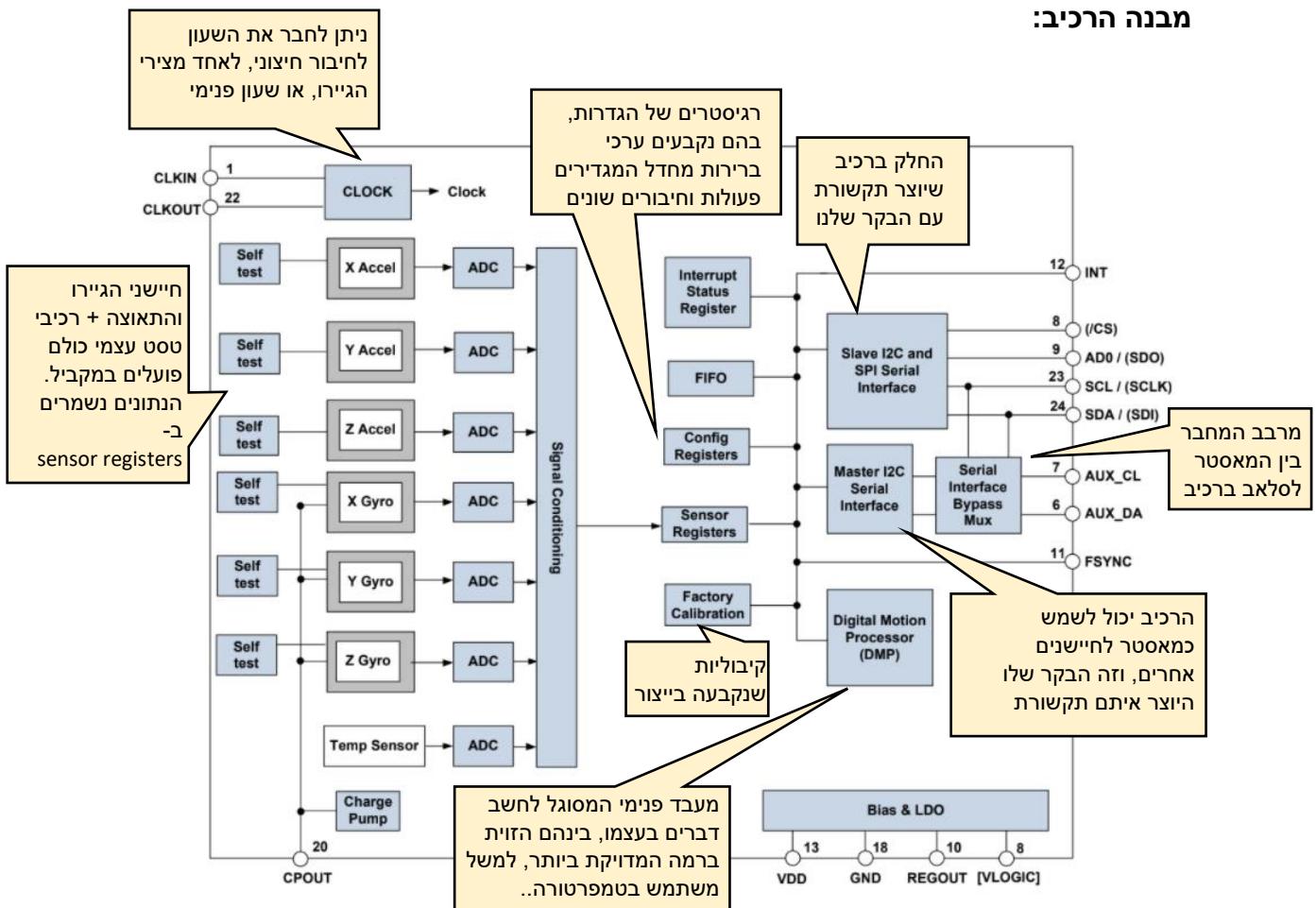
השבנת צרי מזווית: החישן יעבור למצב השכמה- כלומר בתדרות שתקבע ע"י 2 הסיביות השמאליות (6-7) הרכיב יפעיל את החישן במרווח זמן קבועים כדי לבצע מדידה אחת. תדרות השעון נקבעת עפ"ו הטבלה:

LP_WAKE_CTRL	Wake-up Frequency
0	1.25 Hz
1	5 Hz
2	20 Hz
3	40 Hz

בפרויקט שלנו לא השתמשנו באפשרות להשבנת צרי החישנים.

**פירוט חיבורים לרכיב:**

- .INT (12) – הודעת פסיקה כשייש נתונים זמינים חדשים. אנחנו לא משתמשים בה.
- .VCC (13), GRD (18) – מתח ודמה עבור אספקת החשמל של הרכיב
- .SDA (23), SCL (24) – קו נתונים וקו שעון עבור תקשורת טורית, כך שהרכיב הוא העבד.
- .XDA (6), XCL (7) – קו נתונים וקו שעון עבור תקשורת טורית, כך שהרכיב הוא מסטר המתחבר לחישנים חיצוניים. אנחנו לא משתמשים בה.
- .ADO (9) – כתובת העבד ברכיב עבור התקשרות הטורית הנ"ל



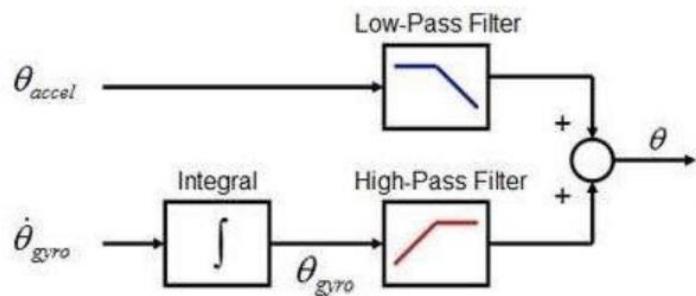
## מסנן משלימים – COMPLEMENTARY FILTER

## הבעיה עם מד תואוצה:

מד תואוצה מודד את כל הכוחות הפעילים על עצם כלשהו. כל כוח נוסף שיופיע על העצם ישפיע על הדגימות בצורה קיצונית. המידע ממד התואוצה אמין רק בטווח האור.

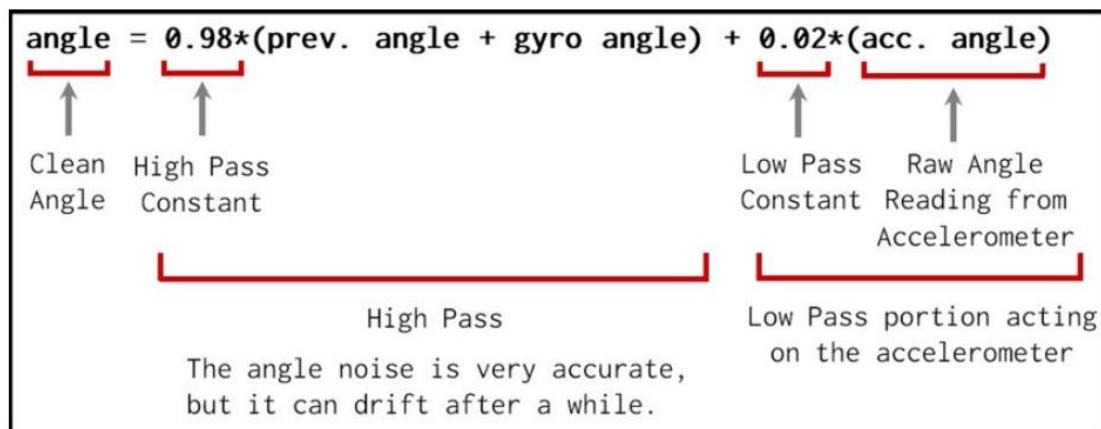
## הבעיה עם מד זווית:

מד זווית אינו מושפע מכוחות חיצוניים, אבל מכיוון שהזווית נמדדת לאורך זמן נוצרת סטייה. המידע ממד הזווית אמין רק בטווח הקצר.

מסנן משלימים

המסנן בניי מביר נמוכים עבור מד-התואוצה, (מנחית את השינויים הקיצוניים של מד התואוצה בטווח הקצר) ומסנן גבוהים עבור מד הזווית (מנחית את הסטייה של מד הזווית בטווח האור), כך נקבל את הדיקוק המרבי.

## שימוש בתוכנית:



## הסבר:

0.98: קבוע איסוף הנתונים ( $\hat{z}_p$ ) כפול מהירות הזוויתית (מניחים שהמהירות קבועה בזמן זהה) + הזווית הקודמת

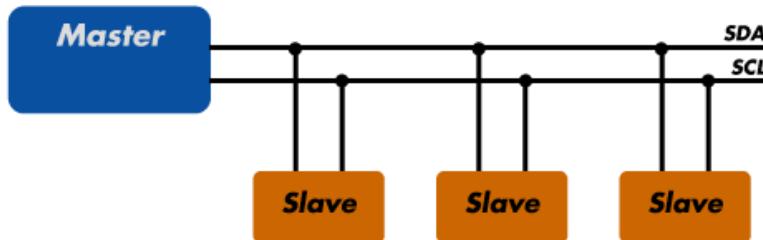
0.02: הזווית שחושבה ע"י מד התואוצה- ROLL.

## תקשורת I2C

בתקשורת I2C ישנו Master שהוא בדרך כלל המעבד, שמתקשר עם מגוון רחב של מודולים וחישנים (Slaves) באמצעות שני קווים בלבד:

1. קו הנטוני - SDA (Serial Data) קו דו כיווני עבור קריאה/כתיבה.
2. קו דופק השעון - SCL (Serial Clock) קו חד כיווני ומופעל ע"י המאסטר. קובע את קצת העברת הנתונים.

בארכיטריאנו אונו A4 משמש כקו SDA ו A5 משמש כקו SCL  
ניתן לחבר על זוג החוטים עד 128 התקנים (Slaves) שונים במקביל.



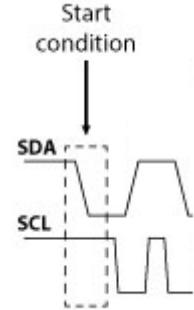
לכל רכיב/מודול/חישן יש כתובות ייחודית משלו. ה Master בשלב ראשון מוציא את הכתובת הייחודית של רכיב מסוים. רכיב שמצויה שזו הכתובת שלו, 'מאשר' את זהה הכתובת והוא זה שיתקשר עם Master . כל שאר הרכיבים נשאים במצב 'נתק' מהקווים.

דופק השעון שעל קו SCL מיוצר ע"י ה Master והוא זה שקובע את קצב העברת הנתונים. בקו SDA מעבירים בצורה טורית את הכתובות ואת הנתונים שבין ה Master וה Slaves .

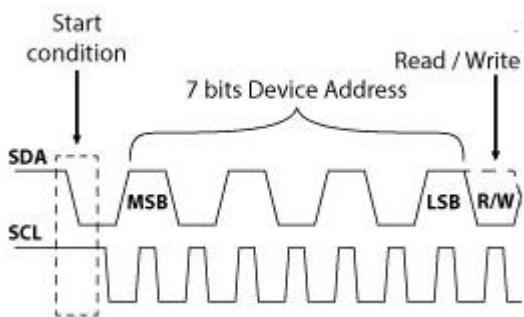
כאשר אין תקשורת, שני הקווים SDA ו SCL נמצאים במצב של '1' קבוע.

### פירוט הפעולות:

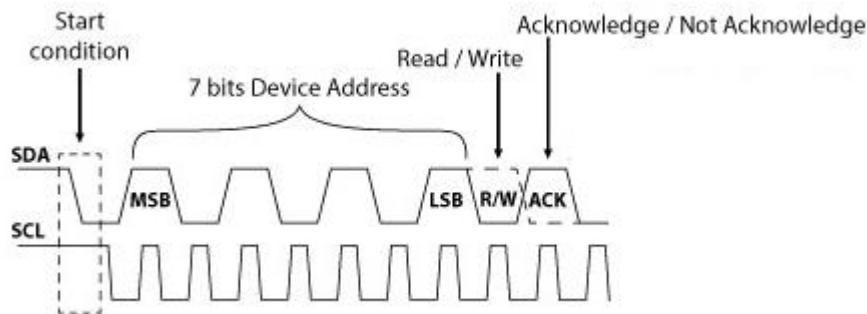
(1) - הורדת קו SDA ל '0' כאשר קו SCL עדין נמצא במצב '1' קבוע.



(2) ה Master מshedר על קו ה SDA 7 סיביות של כתובות (סיבית ה MSB נשלחת ראשונה) + סיבית שמינית המודיעה לרכיב האם המאסטר רוצה לכתוב מידע לרכיב ('0') או לקרוא מידע ממנו ('1')

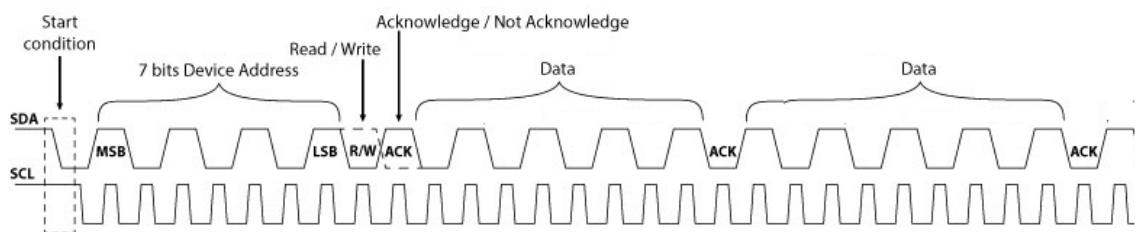


(3) הרכיב שזו כתובתו מגיב ב'אישור' (Acknowledge) , המתבטא בהורדת קו SDA ל '0' למשך דופק שעון אחד.

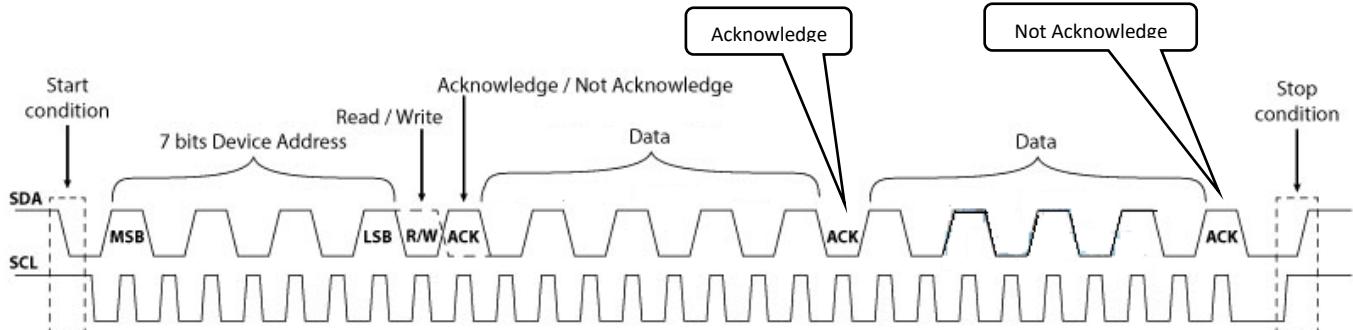


אם אין Acknowledge , ה Master יסימם את התקשרות בפעולה Stop (שתוגדר בהמשך). לרכיב יש כתובות זיהוי יחודית , אבל בד"כ יש בו אוגרים (Registers) המכילים מידע, וכל אוגר זה יש כתובת משלה, כך שבד"כ אחרי שידור כתובת הרכיב ציריך לשדר את כתובת האוגר ורק אז מגיעים אל קריית/כתיבת הנתונים.

(4) בעקבות Acknowledge , המעבד כותב (או קורא) נתונים בגודל Byte (8 סיביות) אחרי כל שידור בין ה Master ובין Slave יש לקבל Acknowledge .



(5) סיום התקשרות - פעולה Stop מתבצעת על ידי העילאת קו SDA ל '1' כאשר קו SCL במצב '1'



**הספריה wire.h**

מהפונקציות בספרייה:

- אחול ייחידת C2I של הארדואינו - Wire.Begin()
- התחלה תקשורת עם ה slave - Wire.beginTransmission(address)
- סיום תקשורת, stop - Wire.endTransmission()
- שידור נתונים - Wire.write(number)
- (במקרה 'ערך מספרי' אפשר לכתוב כ'פרמטר' גם מחוזצת או כתובות של מערך + גודל המערך)
- ועוד ...

**מנוע רטט****מנוע DC:**

מנוע צפוני הממיר אנרגיה חשמלית לאנרגיה מכנית.

מנוע DC (זרם ישיר) מtabסס על תופעת האלקטרומגנטיות – המאפשרת יצירת שדה מגנטי "ע"י העברת זרם צפוני דרך סליל. שדה זה מפעיל כח על הcyיר העובר דרכו, מה שגורם לו להסתובב. כיוון סיבוב המנוע תלוי בקצבוביות המתה, لكن אם נהפוך את החיבורים המנוע יסתובב בכיוון ההפוך.

מנוע הרטט שלנו הוא מנוע DC עם משקלות לא מאוזנת בקצה, מה שגורם לרעדות רטטיים.

המנוע צריך זרם גבוה שהארודאיינו לא יכול לספק. לכן חיבורנו את המנוע לטרנזיסטור (c) של טרנזיסטור שייהווה מוגבר זרם.

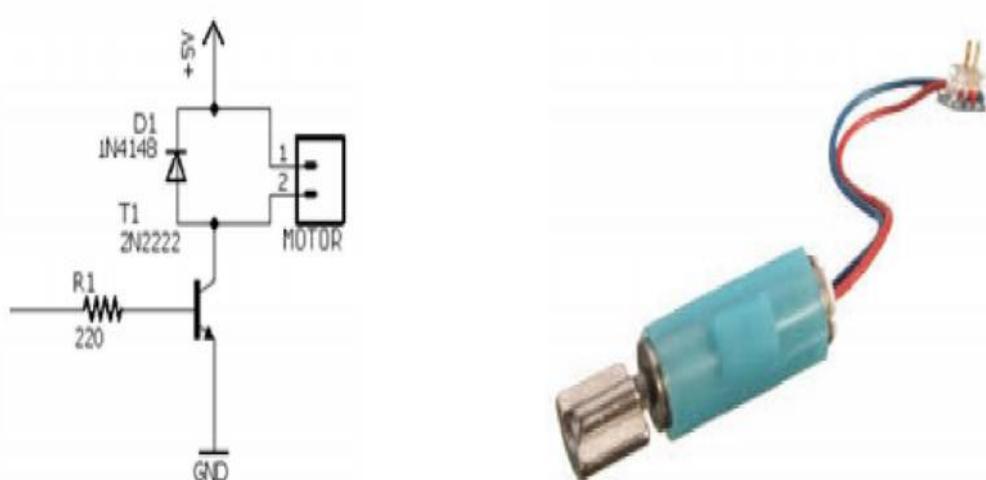
הארודאיינו מספק זרם קטן לבסיס (b) של הטרנזיסטור והמנוע מקבל זרם גדול יותר לפי הנוסחה:  $I_c = \beta I_b$

הדיודה מחוברת במקביל למנוע נועדה להגן על הטרנזיסטור שלא ישרפף.

מצב צזה עשוי לגרום כאשר במצב קיטוען הזרם על הסליל של המנוע יתפרק דרך

הטרנזיסטור לפי חוק לנץ שאומר:  $I_{0+} = I_0^-$

הדיודה מאפשרת לסיליל של המנוע להתפרק ולהזרים את הזרם דרכה.



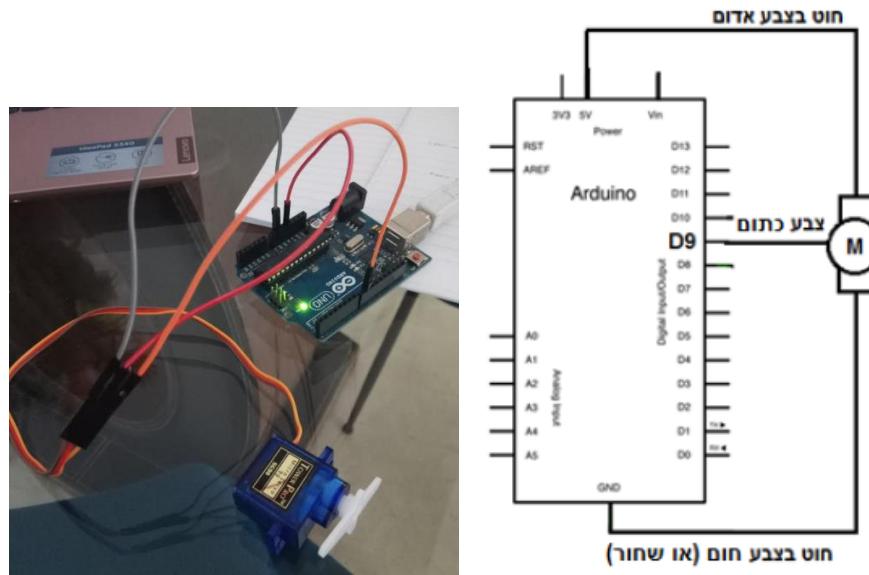
התנסות עם הרכיבים:

לפני בניית הפרויקט התנסינו בהפעלת הרכיבים השונים הקשורים לפרויקט.

### ניסוי 1: מנוע סרו

מנוע סרו הוא מנוע זרם ישיר (DC motor) בעל מערכת תמסורת פנימית של גלגלי שניינים ובקשה אלקטרונית פנימית על מנת המנוע.

בקרת המנוע מנעה את ציר המנוע לזרoit מדוייקת בהתאם לרוחב הפолос המתkeletal בכו הבקשה. בנוסף, הוא בעל יכולת תיקון פערים מהמיוקם הרצוי.



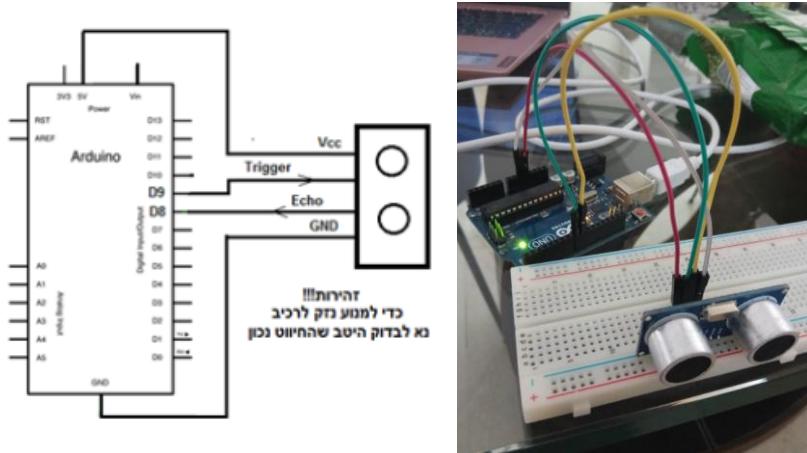
למנוע 3 חוטים: חום- GRD, אדום- מתח 5V, כתום- כנתום- אות לשיליטה במנוע.

הרכזו את התוכנית הבאה להתנסות:

```
//התוכנית מציבה את ציר המנוע בזווית 45 מעלות ולאחר מכן בזווית 135 מעלות וחזרה חלילה:  
#include <Servo.h>  
Servo myservo; //myservo בשם servo  
void setup()  
{  
    myservo.attach(9); // קבע הדק בקרה למנוע סרו בדיק 9 ;  
}  
void loop()  
{  
    myservo.write(45); // קבע ציר מנוע ל 45 מעלות ;  
    delay(1000);  
    myservo.write(135); // קבע ציר מנוע ל 135 מעלות ;  
    delay(1000);  
}
```

## ניסוי 2: חישון מרחק

ידוע שהתפשטות גלי הקול במרחב היא ב מהירות של  $c = 340m/s$ . חישון ה Ultra-Sonic משדר למרחב גלי קול בתדר (Ultra-Sonic) של  $40kHz$ . החישון נותן לנו את פרק הזמן שחלף מרגע שידור גלי הקול ועד לקבלת ההד בחרזה. ניתן לחשב את המרחק על פי הנוסחה:  $\text{מרחק} = \text{מהירות} \times \text{זמן}$ .



לחישון 4 הדקים: 2 לאספקת מתח (GRD,5V) (Trigger), הדק מבוא (Trigger) למתן דרבון, הדק מוצא (Echo) לקבלת הזמן שחלף מרגע שידור גלי הקול ועד לקבלת ההד מוצא.

בין התוכניות שהרכזנו להתנסות:

```

int trig=8, echo=9;
int time, distance;
void setup()
{
    // הדק 8 הוא מוצא כדי לחתת 'דרבון' למבוא החישון
    // הדק 9 הוא מבוא כדי לקבל מידע על ההד
    pinMode(trig,OUTPUT);
    pinMode(echo,INPUT);
    Serial.begin(9600);
}
void loop()
{
    digitalWrite(trig,HIGH); // מעלים את הדק 8 ל '1' כדי לחתת דרבון ;
    delayMicroseconds(10); // למשך 10 מיקרו שנייה ;
    digitalWrite(trig,LOW); // הפסיקת אותן הדרבון //
    time=pulseIn(echo,HIGH); // מדדים את המרחק בס"מ
    distance=time/58; // חישוב המרחק
    Serial.print ("distance = ");
    Serial.println(distance); // הדפס במוניטור את המרחק
    delay(100);
}

```

תקלה: החילפנו בין הדקים של הדרבן והמוצא. פתרון: בתוכנית נהפוך את ערכי `trig`, `echo` להדקים תהיה נכונה (`driven=9, מוצא=8`)

### ניסוי 3: שימוש ברכיב PCF8574 עבור תקשורת I2C

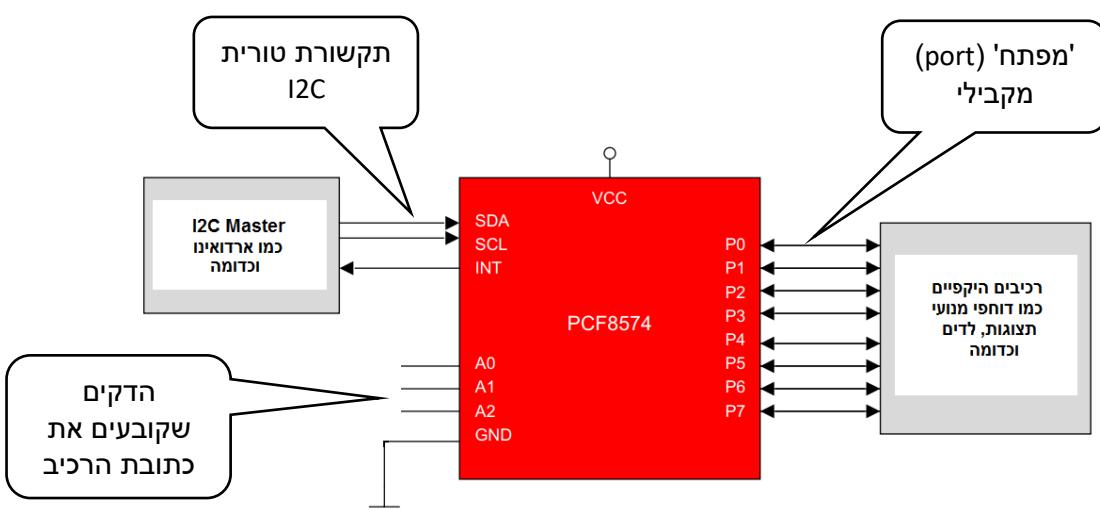
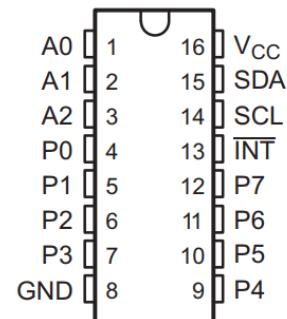
בתקשורת I2C ישנו Master שהוא בדרך כלל המעבד, שמתקשר עם מגוון רחב של מודולים וחישנים (Slaves) באמצעות שני קווים בלבד.

לכל רכיב/מודול/חישן יש כתובות ייחודית משלו. ה Master מוציא את הכתובת הייחודית של רכיב מסוים. רכיב שמצויה שזו הכתובת שלו, מאשר' את זהה הכתובת והוא זה שיתקשר עם Master. כל שאר הרכיבים נשאים במצב של 'נתק' מהקוים.

הרכיב שבחרנו לתרגול ה I2C משמש כ'מרחיב פורטים'.

הרכיב דו ציוני, כלומר אפשר לשדר לו מידע (בגודל byte) טורי שיופיע בפתח המקביל ואפשר גם לקרוא את המידע שנמצא בפתח המקביל.

#### מבנה הרכיב:



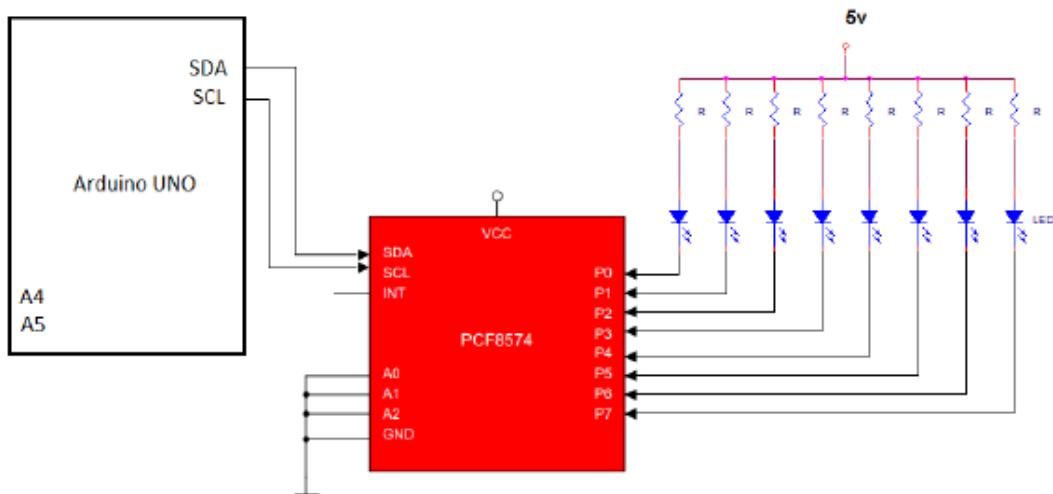
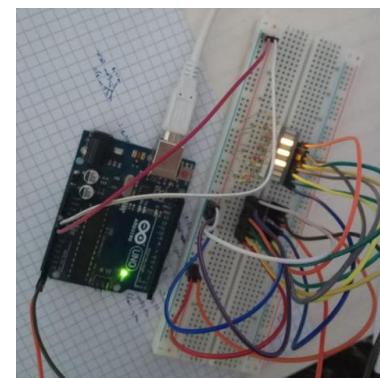
לרכיב 3 הדקינו כתובות, כלומר ניתן ליצור 8 כתובות שונות:

BYTE	BIT							
	7 (MSB)	6	5	4	3	2	1	0 (LSB)
I <sup>2</sup> C slave address	L	H	L	L	A2	A1	A0	R/W

חיבורנו את 3 ההדקינים לאדמה ולכן נוצרה הכתובת 0x20 – 0x20 – זה תהיה הכתובת הרכיב.

בארדואינו אונו A4 משמש כקו SDA ו A5 משמש כקו SCL

התנסות בתקשרות עם רכיב המשמש כקלטן:  
למפתח המקביל של הרכיב חיבורנו בר לדים בחיבור 'אנודה משותפת'. בתוכנית הבאה הארדואינו יכתוב לרכיב אילו לדים להדליק (ע"י '0') ואילו לכבות (ע"י '1').



```
#include <Wire.h> // I2C
#define SDA_A4
#define SCL_A5

void setup() {
    Wire.begin(); // Initialize the i2c bus
}

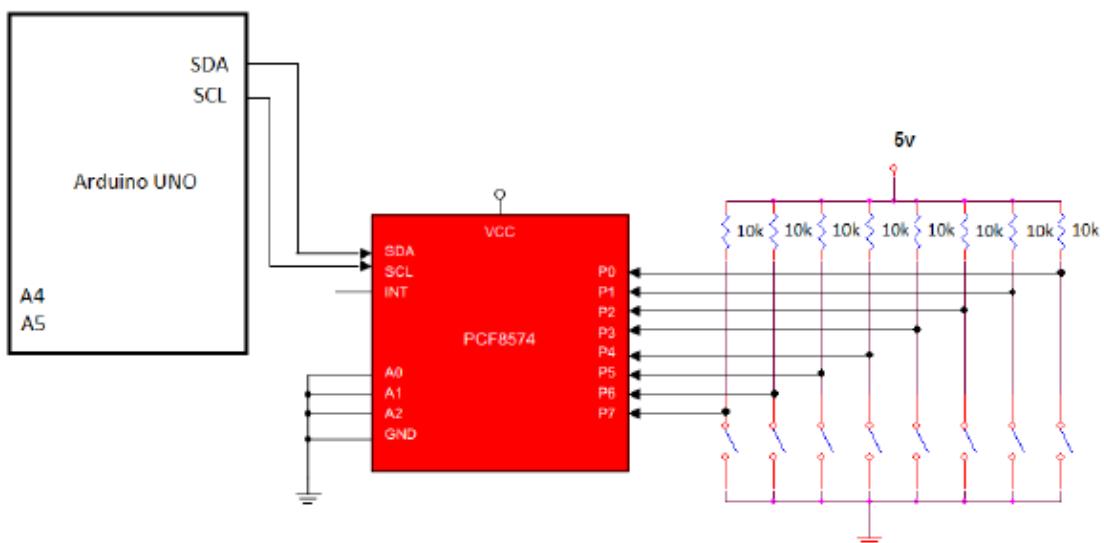
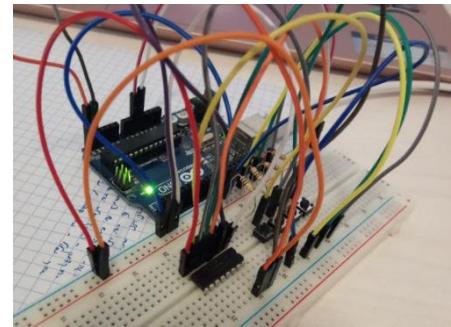
void loop()
{
    // The program sends 4 bytes to the slave device
    Wire.beginTransmission(0x20); // Address
    Wire.write(0x55); // 01010101
    Wire.endTransmission(); // End transmission

    delay(500);

    Wire.beginTransmission(0x20);
    Wire.write(0xaa); // 10101010
    Wire.endTransmission();
    delay(500);
}
```

התנסות בתקשרות עם רכיב המשמש כפלט:

למפתח המקבילי של הרכיב חיבורנו 8 מפסקים. בתוכנית הבאה הארדואינו יקלוט מהרכיב את המפתח המקבילי נתון בינאריו וידפיס אותו על מסך המחשב.  
מספק לחץ במצב הבא יתן '0' בהדק, אם לא לחוץ-'1'.



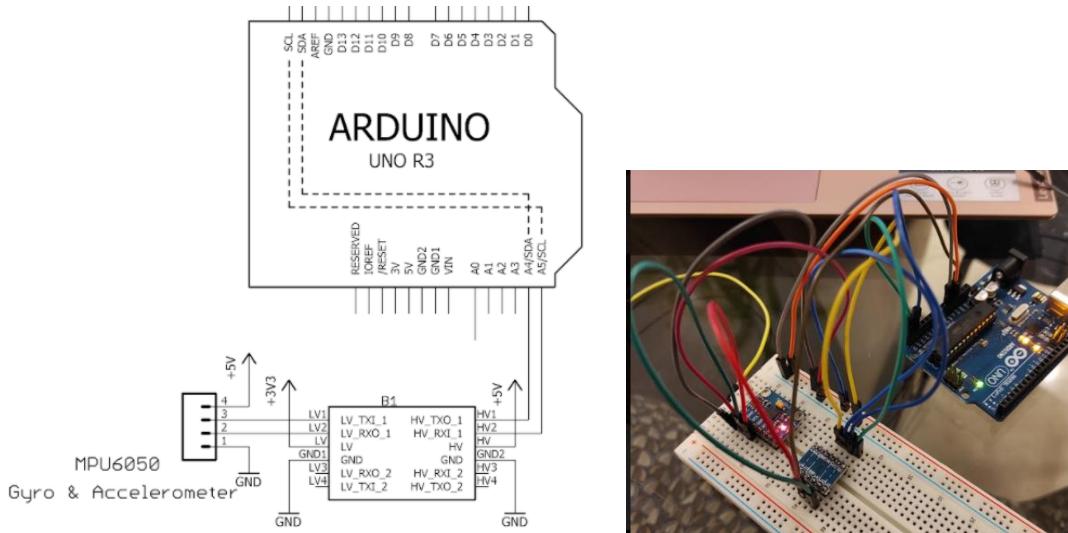
```
//התוכנית קולטת מהמפתח המקבילי של התוכנית נתון אחד ומציג את ערכו במוניטור של הארדואינו
#include <Wire.h>
void setup() {
    Wire.begin();
    Serial.begin(9600);
}
void loop()
{
    Wire.requestFrom(0x20,1); //דרוש רכיב slave שכתובתו 0x20, נתון (byte) byte
    byte c = Wire.read(); // slave הצלב במשתנה c את הנתון שקראת מרכיב ה
    Serial.println(c); //הדפס את תוכן המשתנה על המוניטור
    delay(500);
}
```

#### ניסוי 4: חיישן MPU6050- מד מהירות זוויתית ותאוצה

החיישן מודד מהירות זוויתית ותאוצה, מכיל אקסלורומטר (מד תאוצה) וגירוסקופ. הוא מדויק מאוד מכיוון שכל נתון נשמר כמספר דיגיטלי בן 16 ביטים.

תפקיד החישן בפרויקט הינו לתת אינדיקציה על זווית המקל ביחס לקruk.

הארדואינו קורא את הנתונים ע"י שימוש בתקשורת I2C



לחישן 4 הדקים בהם אנו עושים שימוש: 2 לאספקת מתח (5V), 2 עבור תקשורת (I2C) (SCL, SDA)

ישנו 4 הדקים נוספים שלא נשתמש בהם, והם מפורטים בהසבר על הרכיב.

את התוכנית הבאה להתקנות הורדנו מהקובץ [choveret\\_ezer\\_electronica.pdf](#) הנמצא בקישור, שם הופיעו גם הסברים על הרכיב והשימוש בו:

[https://sites.education.gov.il/cloud/home/cyber\\_israel/Documents/choveret\\_ezer\\_electronica.pdf](https://sites.education.gov.il/cloud/home/cyber_israel/Documents/choveret_ezer_electronica.pdf)

לא מופיע שם השימוש בממיר המתח, אבל בהמלצת המנהים הוסףנו זאת כדי שלא יהיה נזק לחישן בעקבות מתח גובה מד.

התוכנית מחשבת את ערכי roll, pitch ע"י קריית הערכים במד התאוצה ומדפסה אותם במניטור.

תקלות שהו:

המניטור הראה שערך roll, pitch מחושבים לערך מסוים בהתחלה ומיד הולכים וושאפים למינוס 1, והבנו שכנראה החיבורם לא תקין. החלפנו משטח מטריצה וחיברנו מחדש וaz התוכנית עבדה כראוי.

---

```

#include<Wire.h>
#define MPU_addr 0x68
int AccX, AccY, AccZ;
double pitch, roll;
unsigned long time=0;

void setup() {
    Wire.begin(); //Initialization of I2C port
    Wire.beginTransmission(MPU_addr); //Initial transmission with address
    Wire.write(0x6B); //Register 6B
    Wire.write(0); //Register 0
    Wire.endTransmission(true); //End transmission
    Serial.begin(115200);
}

void loop() {
    if(millis()-time>200) { //Every 200ms
        Wire.beginTransmission(MPU_addr);
        Wire.write(0x3B);
        Wire.endTransmission();
        Wire.requestFrom(MPU_addr, 6, true);
        AccX = Wire.read() << 8 | Wire.read(); //0x3B
        AccY = Wire.read() << 8 | Wire.read(); //0x3D
        AccZ = Wire.read() << 8 | Wire.read(); //0x3F
        // RAD_TO_DEG = 180 / PI(3.141592)
        // calculate pitch and roll angles in radians
        pitch = RAD_TO_DEG *atan(AccX / sqrt(square(AccY) + square(AccZ)));
        roll = RAD_TO_DEG *atan(AccY / sqrt(square(AccX) + square(AccZ)));
        Serial.print(" pitch = ");
        Serial.print(pitch);
        Serial.print(" roll = ");
        Serial.println(roll);
        Serial.println(" =====");
        time=millis();
    }
}

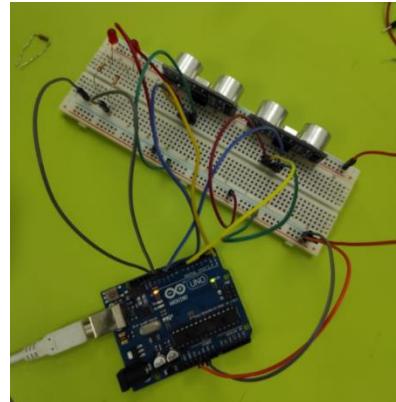
```

**תיעוד בנית הפרויקט:**

**כתיבת התוכנית:**

**תיעוד תחיל**

החלנו להתחל בחלק של חישני המרחק שיגרם להבاهוב של לד בקצב התלי במרקח הנמדד.



בשלב הראשון כתבנו תוכנית הגורמת לד לדלק **באופן רציף** כאשר החישון מזהה עצם קרוב מדי, אחרת הוא נכבת. (בהתחלתו עבדנו עם שני חישנים, אבל בסוף השארנו רק 1 בהמלצת המנחה

```
void loop()
{
    if(millis()-time>500) { // מדידת מרחק כל 500 שניות
        Serial.print(distance1); // הדפס את המרחק בס"מ
        Serial.print(" cm- distance1 "); // הדפס ס"מ ועובד שורה
        trigUs(); // אחורית
        time=millis(); // לכבות אותו
    }
    if (distance1<10 ) // אם המרחק קטן מ 10 ס"מ
        digitalWrite(led,HIGH); // הדלק את הלהד
    else // אחרית
        digitalWrite(led,LOW); // לכבות אותו
}
```

על מנת למנוע שימוש בדילי במהלך מדידת המרחק השתמשנו בתוכנית פסיקה חיצונית (בנספחים)

**שיטת ה'פסיקה' - (Interrupt).** בשיטה זו כל עוד מצב הקולט לא השתנה, המיקרובוקר משHIR בפעולות שגרתית של הריצת תוכנית ראשית. רק כאשר יש 'AIROU', כלומר רק כאשר מצב ה'קולט' משתנה (אנו קוראים למצב זה 'פסיקה'), המיקרובוקר **מפסיק** את פעילותו השגרתית ומבצע 'תוכנית פסיקה'. בסיום תוכנית הפסיקה המיקרובוקר חוזר לתוכנית הראשית, זאת עד לבקשת 'פסיקה' נוספת. בשיטת ה'פסיקה' המיקרובוקר לא מזבז זמן בהמתנה לאירוע'...

לארדואינו אנו יש שני מקורות פסיקה חיצוניים:

. interrupt 0 - בבדיקה 2.

. interrupt 1 - בבדיקה 3.

**פקודות עבר שימוש בפסיכה:**

- אפשר בקשה פסיקה - attachInterrupt( high/low/change/rising/falling ) שם תוכנית פסיקה, מס' פסיקה 0/1 ( detachInterrupt(interrupt 0/1) – חסימת האפשרות לפסיקה - )

הבהוב הלאן: הוספנו את הפונקציה דילאי על מנת שהלאן יבהיר במקום שידליך באופן רציף.  
זמן שהצבנו בפונקציה היה משתנה שקיבל את המרחק שנמדד כפול קבוע:

```
if (distance<10 )
    digitalWrite(led,HIGH);
    delay(distance*10);
    digitalWrite(led,LOW);
    delay(distance*10);
else
    digitalWrite(led,LOW);
```

הבנו שນוצר צורך לשנות את התוכנית כך שלא יהיה שימוש בדילאי כדי לא לפגוע בביטוי התוכנית.

בעזרתו האדיבת של גאורגי שינוינו את התוכנית כך שהשתמשנו בפונקציה מילוי () לשמרות זמן שעון והשימוש בהם הוא בפקודות תנאי.

כעת נרצה להחליש את עצמת הרטט מכיוון שהרטטים שלו כרגע חזקים מאוד ולא נעימים.  
לשם כך שינוינו את הפקודה digitalWrite(led,HIGH) ל analogWrite(led,100) (המספר נבחר שרירותית כדי לקבל בערך שליש מהעצמה המקורית) ושינוינו את מוצא הלאן להדק התומך בפקודה זו.

ע"מ שקצב ההבהובים ישנה ביחס למרחוק קבענו משתנה שקבע את הקצב באופן יחסית לмерחוק זה בעזרת הפונקציה map(value, fromLow, fromHigh, toLow, toHigh).

הפונקציהמחזירה את הנתון בטוחה הקודם באופן יחסית לטווח החדש, למשל מרחק 110 יחזיר 450

**הчисוב:**

$$(value - \text{fromLow}) * (\text{toHigh} - \text{toLow}) / (\text{fromHigh} - \text{fromLow}) = \text{value\_return}$$

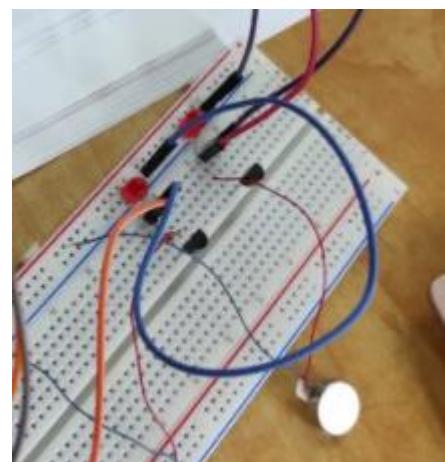
$$\text{distance} * 3.9 + 22 = \text{value\_return}$$

```
if ( millis() - timer_retet > 50)    50ms
{
    rate_retet = map(distance, 20, 200, 100, 800);
    timer_retet = millis();
}
```

לאחר מכן כתבנו את הפונקציה הבאה הייצרת הבהובים באופן לינארי ביחס למרחק. הפונקציה תפעיל את המנווע במשך 100ms ולאחר מכן תכבה אותו עד לסיום הזמן שנקבע במשתנה `.rate_retet`.

```
static unsigned long vibrationTimer = 0;
if (millis() - vibrationTimer < rate_retet) //בדיקה האם אנחנו באותו מועד
{
    if (millis() - vibrationTimer < 100) //בדיקה האם להdelay או לכבות
        analogWrite(led, 100);
    else
        analogWrite(led, 0);
}
else
    vibrationTimer = millis();
```

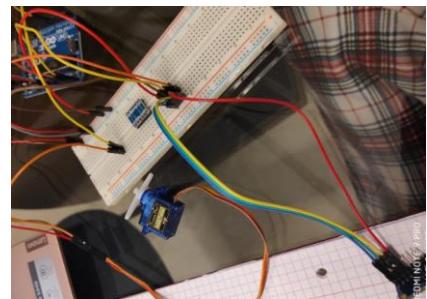
לאחר שחלק זה בתוכנית עבד כראוי הוציאנו את הלהד ושמנו במקום את מנווע הרטט כדי שבמקום הבהובי אור קיבל רטטים.



עברנו לחלק הבא – חישוב הזווית על מנת לייצב את חישון המרחק.

ראשית הרצינו תוכנית לדוגמה כדי להבין את השימוש במיד התאוצה והמהירות הזוויתית (נמצא בניםויים).

לאחר מכן הרצינו תוכנית שקיבלו מגורגי המחשבת את הזווית ביחס לקרקע ע"י הנתונים שמודד החישון, ואת הזווית שחישבה שולחת למנוע סרו ע"מ שיסתובב לזווית הרצואה.



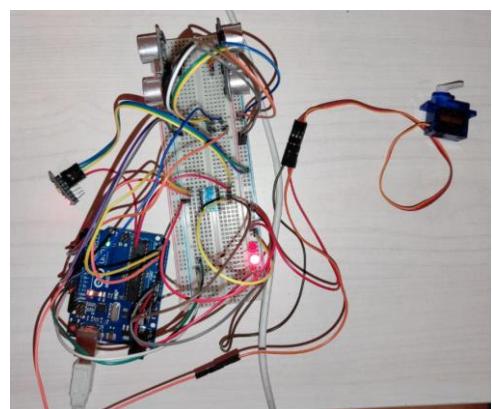
(התוכנית נמצאת בקובץ deg\_servo6050.h וعليה הוספנו תיעוד כדי שהתוכנית תהיה ברורה יותר – מופיע בנספחים).

בעיה שנוצרה: פתח הקופסא מגביל את מנוע הסרוו להסתובב בטוווח המלא של 0°>זווית<180°. לכן הוספנו תנאי המגביל את סיבוב המנוע לטוווח 20°>זווית<160°:

```
int servoAngle = compAngleX + 90;
//תנאי להגבלת הזווית כך שייתאים להגבלות הקופסא
myservo.write(160);
else if(servoAngle < 20)
    myservo.write(20);
myservo.write(servoAngle);
```

בנוסף הוספנו משתנה לשימרת הדק מנוע הסרוו.

לאחר שחלק זה בתוכנית עבד כראוי ביצענו מיזוג בין שני החלקים לקובץ שלישי ע"מ שבו נחHAMם יעבדו במקביל. בגלל שלא השתמשנו בפקודות דילאי אלא בפסקיות ובטיימרים שת' התוכניות לא מפריעות זו לזו והן יכולות לעבוד במקביל.



בעיה שנוצרה:

הטיימרים בשני הקבצים היו בעלי אותו השם. שינוו את שמות הטיימרים כך שלא תהיה חפיפה, ובנוסף יהיה קל לבדוק ביניהם.

## התוכנית:

```

#include <Servo.h> //ספרייה מנוע סרוו
#include <Wire.h> //ספרייה I2C

Servo myservo; //אובייקט של המאצלקה סרוו עבור המנוע
int servoPin = 5;

const int MPU_addr = 0x68; //כתובת הרכיב MPU6050
double AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; //משתנים עבור הנתונים שנקרא מהרכיב
unsigned long timer; //טיימר עבור מדידת הזמן בין חישובי הדזוזית
double compAngleX, compAngleY; //משתנים עבור הדזוזיות שנחשב
#define degconvert 57.2957786 //המרה מרדיאנים למלילות

#define trigPin 4
#define echoPin 2 //בבדיקה זה מתאפשרה תוכנית פסיקה
#define led 3 //הדק מנוע הרט
unsigned long timer_rete=0; //טיימר עבור יצירת פעימות רטט
int rate_rete=0; //משתנה לשמירה תדירות הרטט כתלות במרקז
unsigned long duration; //משתנה עבור רוחב הפולס
int distance=0; //משתנה עבור המרחק הנמדד

unsigned long timer_distance=millis(); //טיימר עבור מדידת המרחק

//הפונקציה נותנת דרבונו 10us בקבוא trigger של ההיישן
void trigUs() {
    digitalWrite(trigPin, HIGH); //יצירת דרבונו
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    attachInterrupt(0, startCount ,RISING); // startCount () // startCount ()()
}

//הפונקציה מאפשרת את מדידת הזמן
void startCount() {
    duration=micros(); //התחלת רוחב הפולס
    detachInterrupt(0); // 0 כסום פסיקה מס' 0 //
    attachInterrupt(0, measurement, FALLING); // measurement () // measurement ()()
}

//הפונקציה מודדת את הזמן ומחשבת את המרחק
void measurement()
{
    duration=micros()-duration; //חישוב רוחב הפולס
    distance=duration/58; // חישוב המרחק בס"מ
    detachInterrupt(0); // 0 כסום פסיקה מס' 0 //
    trigUs(); //דרבוון ואפסור פסיקה מחדש
}

```

```

void vibrate() { //פונקציה המפעילה את מנוע הרטט בקצב ביחס למרחק
    if ( millis() - timer_repetet > 50) //עדכן את הקצב כל 50ms
    {
        rate_repetet = map(distance, 20, 200, 100, 800);
        timer_repetet = millis();
    }

    static unsigned long vibrationTimer = 0;
    if (millis() - vibrationTimer < rate_repetet) //בדיקה האם אנחנו באותו מזמן
    {
        if (millis() - vibrationTimer < 100) //בדיקה האם להציג או לכבות
            analogWrite(led, 100);
        else
            analogWrite(led, 0);
    }
    else
        vibrationTimer = millis();
}

void servoLoop() { //איסוף הנתונים כל פעם מודש – תוכנית גירר
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 14, true);
    AcX = Wire.read() << 8 | Wire.read(); // 0x3B & 0x3C
    AcY = Wire.read() << 8 | Wire.read(); // 0x3D & 0x3E
    AcZ = Wire.read() << 8 | Wire.read(); // 0x3F & 0x40
    Tmp = Wire.read() << 8 | Wire.read(); // 0x41 & 0x42
    GyX = Wire.read() << 8 | Wire.read(); // 0x43 & 0x44
    GyY = Wire.read() << 8 | Wire.read(); // 0x45 & 0x46
    GyZ = Wire.read() << 8 | Wire.read(); // 0x47 & 0x48

    double dt = (double)(micros() - timer) / 1000000; //caculation of time. micros() - timer in seconds
    timer = micros(); //caculation of timer for new calculation

    double roll = atan2(AcY, AcZ) * degconvert;
    double pitch = atan2(-AcX, AcZ) * degconvert;

    double gyroXrate = GyX / 131.0; //calculation of gyroscope rates
    double gyroYrate = GyY / 131.0;

    compAngleX = 0.99 * (compAngleX + gyroXrate * dt) + 0.01 * roll; //calculation of compensated angles
    compAngleY = 0.99 * (compAngleY + gyroYrate * dt) + 0.01 * pitch;

    int servoAngle = compAngleX + 90;
    if(servoAngle > 160) //condition to limit the angle to 160 degrees
        myservo.write(160);
    else if(servoAngle < 20)
        myservo.write(20);
    myservo.write(servoAngle);
}

```

```

void setup() {
    myservo.attach(servoPin); //הגדרות ואתחול מנווע
    myservo.write(90);

    //הגדרות ואתחול – גיררו
    Wire.begin(); //אתחול יחידת I2C של הארדואינו
    Wire.setClock(400000UL); //קובע את תדריות התקשורת ל 400kHz
    Wire.beginTransmission(MPU_addr); //התחלת תקשורת עם החישון – מקבל את הכתובת לשידור
    Wire.write(0x6B); //פנינה לרגיסטר 6B .
    Wire.write(0); //אייפוס הרגיסטר
    Wire.endTransmission(true); stop //סיום התקשורת – שחרור הקו נ"י שליחת
    Serial.begin(115200); //קבע את קצב השידור ל115200 סיביות בשנית ;
    delay(100);

    Wire.beginTransmission(MPU_addr); //התחלת תקשורת עם החישון
    Wire.write(0x3B); //פנינה לרגיסטר 3B לצורך התחלת קריאת נתונים
    Wire.endTransmission(false); stop //הפסקת השידור, בלי להפסיק את התקשורת – אינן שליחת stop
    Wire.requestFrom(MPU_addr, 14, true); //גיישה לבטים בכתובת החישון, 14 קריאות, והפסיקת התקשורת בסיום
    AcX = Wire.read() << 8 | Wire.read(); // 0x3B & 0x3C
    AcY = Wire.read() << 8 | Wire.read(); // 0x3D & 0x3E
    AcZ = Wire.read() << 8 | Wire.read(); // 0x3F & 0x40
    Tmp = Wire.read() << 8 | Wire.read(); // 0x41 & 0x42
    GyX = Wire.read() << 8 | Wire.read(); // 0x43 & 0x44
    GyY = Wire.read() << 8 | Wire.read(); // 0x45 & 0x46
    GyZ = Wire.read() << 8 | Wire.read(); // 0x47 & 0x48

    double roll = atan2(AcY, AcZ) * degconvert;
    double pitch = atan2(-AcX, AcZ) * degconvert;

    compAngleX = roll;
    compAngleY = pitch;

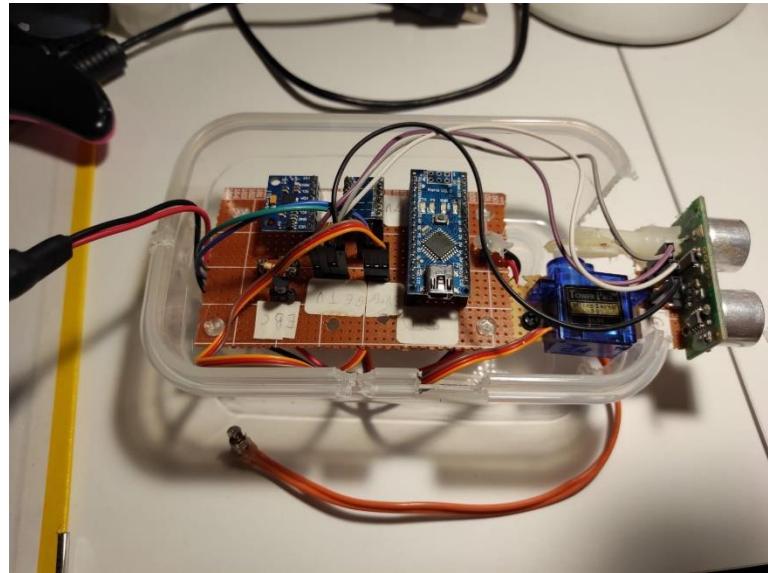
    timer = micros(); //הפעלת הטימר ליחסוב הזמן
}

pinMode(trigPin, OUTPUT); //הגדרות ואתחול דיזון מרחק
pinMode(echoPin, INPUT);
pinMode(led, OUTPUT);
trigUs(); //דרבול ריאשוני של 10us ואפשור פסיקה
timer_distance=millis();
}

void loop() {
    if ( millis() - timer_distance > 50) 50ms //מדידת מרחק כל 50ms
    {
        trigUs(); //דרבול ואפשור פסיקה מזדק
        timer_reteet = millis();
    }
    servoLoop(); //תוכנית גיררו
    vibrate(); //הפעלת רטטים
}

```

**בבנייה המנגנון:**  
בהתמלצת המנהים, השתמשנו בمعالג חשמלי מוכן על מנת לחסוך בזמן



הרכיבים מחוברים ע"י הלחמה ללוח עבודה קטן, והם מחוברים זה לזה ע"י חיווט חוטים. על לוחות נוספים הולחמו חישון מרחק ומנוע סרו  
מנוע הרטט חובר לגומייה שתתלבש על היד במקום ישירות למקל כדי שהרטטים לא יפגעו בחישובי החישונים  
המנגנון כולל חובר לקופסא, ואת הקופסה הצמדנו למקל שיישמש כמקל נחיה.  
לאחר מכן צרבנו את התוכנית על כרטיס הארדואינו.

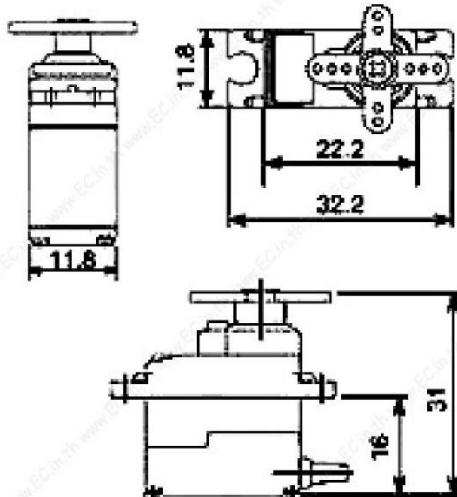
**מסקנות וסיכום**

במשך תקופה ארוכה תוך התמודדות יומיומית עם הקשיים שנוצרו בשל משבר הקורונה השקענו הרבה מרכיבים בפרויקט, הכרנו רכיבים חדשים, העשנו את הידע והביצוע בתכנות, בנוסף רכשנו את יכולות לעבוד בעבודת צוות, ורחבנו את עולם הידע הטכנולוגי שלנו.

נדרשנו לחשב מוחץ לקופסא ולהתמודד עם אתגרים ועת בשלב מרגש זה- בו אנו נהנות מפרוטינו וזכות להגיש את הפרויקט מרגישות אנו תחשות סיפוק אדרה ושמחה על מרכיבינו והשתדלותנו ברוך הוא אהבנו מכך את הפרויקט נהנו מהיצירה וחיבור רחב בא על שפטינו לקרה התוצאה היא רצון שנזכה תמיד לעשות חיל מתקר סיפוק ושמחה!

דף נתוני:

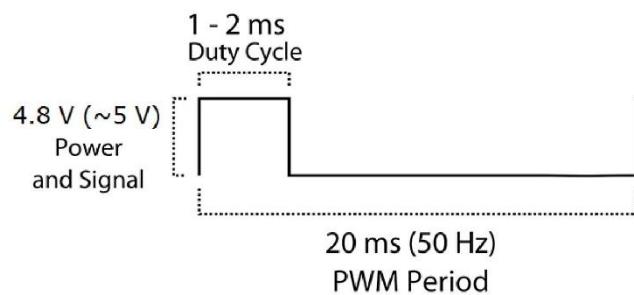
## SG90 9 g Micro Servo



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

### Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10  $\mu$ s
- Temperature range: 0 °C – 55 °C



Position "0" (1.5 ms pulse) is middle, "90" (~2 ms pulse) is all the way to the right, "-90" (~1 ms pulse) is all the way to the left.



## *Understanding the I<sup>2</sup>C Bus*

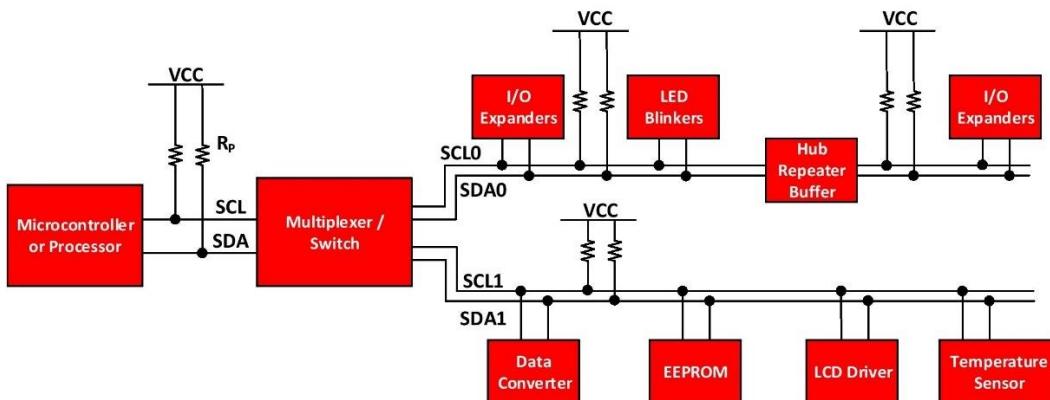
Jonathan Valdez, Jared Becker

### **ABSTRACT**

The I<sup>2</sup>C bus is a very popular and powerful bus used for communication between a master (or multiple masters) and a single or multiple slave devices. **Figure 1** illustrates how many different peripherals may share a bus which is connected to a processor through only 2 wires, which is one of the largest benefits that the I<sup>2</sup>C bus can give when compared to other interfaces.

This application note is aimed at helping users understand how the I<sup>2</sup>C bus works.

**Figure 1** shows a typical I<sup>2</sup>C bus for an embedded system, where multiple slave devices are used. The microcontroller represents the I<sup>2</sup>C master, and controls the IO expanders, various sensors, EEPROM, ADCs/DACs, and much more. All of which are controlled with only 2 pins from the master.



**Figure 1. Example I<sup>2</sup>C Bus**



## 1 Electrical Characteristics

$\text{I}^2\text{C}$  uses an open-drain/open-collector with an input buffer on the same line, which allows a single data line to be used for bidirectional data flow.

### 1.1 Open-Drain for Bidirectional Communication

Open-drain refers to a type of output which can either pull the bus down to a voltage (ground, in most cases), or "release" the bus and let it be pulled up by a pull-up resistor. In the event of the bus being released by the master or a slave, the pull-up resistor ( $R_{PU}$ ) on the line is responsible for pulling the bus voltage up to the power rail. Since no device may force a high on a line, this means that the bus will never run into a communication issue where one device may try to transmit a high, and another transmits a low, causing a short (power rail to ground).  $\text{I}^2\text{C}$  requires that if a master in a multi-master environment transmits a high, but sees that the line is low (another device is pulling it down), to halt communications because another device is using the bus. Push-pull interfaces do not allow for this type of freedom, which is a benefit of  $\text{I}^2\text{C}$ .

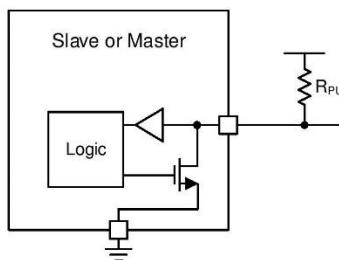


Figure 2. Basic Internal Structure of SDA/SCL Line

Figure 2 shows a simplified view of the internal structure of the slave or master device on the SDA/SCL lines, consisting of a buffer to read input data, and a pull-down FET to transmit data. A device is only able to pull the bus line low (provide short to ground) or release the bus line (high impedance to ground) and allow the pull-up resistor to raise the voltage. This is an important concept to realize when dealing with  $\text{I}^2\text{C}$  devices, since no device may hold the bus high. This property is what allows bidirectional communication to take place.

#### 1.1.1 Open-Drain Pulling Low

As described in the previous section, the Open-Drain setup may only pull a bus low, or "release" it and let a resistor pull it high. Figure 3 shows the flow of current to pull the bus low. The logic wanting to transmit a low will activate the pull-down FET, which will provide a short to ground, pulling the line low.

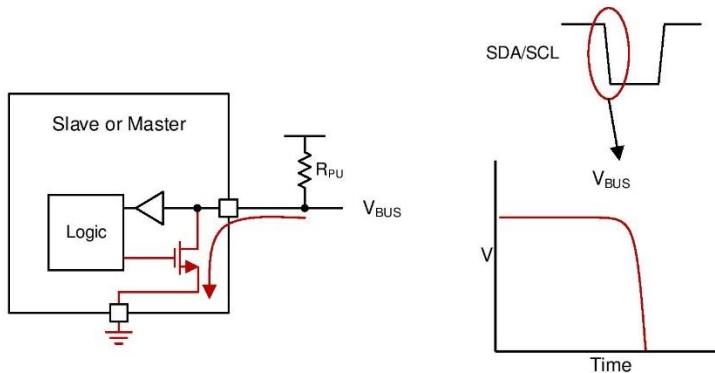


Figure 3. Pulling the Bus Low With An Open-Drain Interface



### 1.1.2 Open-Drain Releasing Bus

When the slave or master wishes to transmit a logic high, it may only release the bus by turning off the pull-down FET. This leaves the bus floating, and the pull-up resistor will pull the voltage up to the voltage rail, which will be interpreted as a high. Figure 4 shows the flow of current through the pull-up resistor, which pulls the bus high.

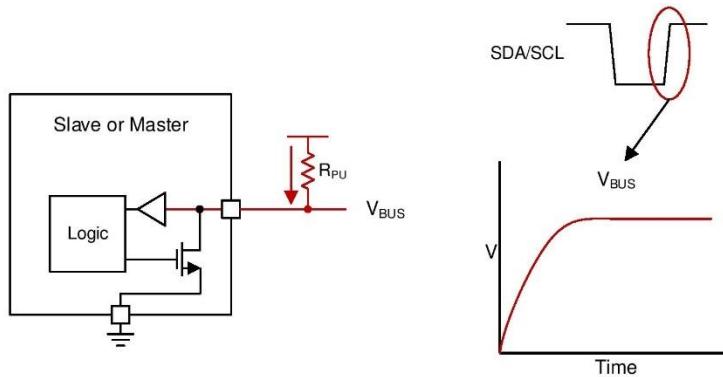


Figure 4. Releasing the Bus With An Open-Drain Interface

## 2 I<sup>2</sup>C Interface

### 2.1 General I<sup>2</sup>C Operation

The I<sup>2</sup>C bus is a standard bidirectional interface that uses a controller, known as the master, to communicate with slave devices. A slave may not transmit data unless it has been addressed by the master. Each device on the I<sup>2</sup>C bus has a specific device address to differentiate between other devices that are on the same I<sup>2</sup>C bus. Many slave devices will require configuration upon startup to set the behavior of the device. This is typically done when the master accesses the slave's internal register maps, which have unique register addresses. A device can have one or multiple registers where data is stored, written, or read.

The physical I<sup>2</sup>C interface consists of the serial clock (SCL) and serial data (SDA) lines. Both SDA and SCL lines must be connected to  $V_{CC}$  through a pull-up resistor. The size of the pull-up resistor is determined by the amount of capacitance on the I<sup>2</sup>C lines (for further details, refer to *I<sup>2</sup>C Pull-up Resistor Calculation (SLVA689)*). Data transfer may be initiated only when the bus is idle. A bus is considered idle if both SDA and SCL lines are high after a STOP condition.

The general procedure for a master to access a slave device is the following:

1. Suppose a master wants to send data to a slave:
  - Master-transmitter sends a START condition and addresses the slave-receiver
  - Master-transmitter sends data to slave-receiver
  - Master-transmitter terminates the transfer with a STOP condition
2. If a master wants to receive/read data from a slave:
  - Master-receiver sends a START condition and addresses the slave-transmitter
  - Master-receiver sends the requested register to read to slave-transmitter
  - Master-receiver receives data from the slave-transmitter
  - Master-receiver terminates the transfer with a STOP condition



### 2.1.1 START and STOP Conditions

$I^2C$  communication with this device is initiated by the master sending a START condition and terminated by the master sending a STOP condition. A high-to-low transition on the SDA line while the SCL is high defines a START condition. A low-to-high transition on the SDA line while the SCL is high defines a STOP condition.

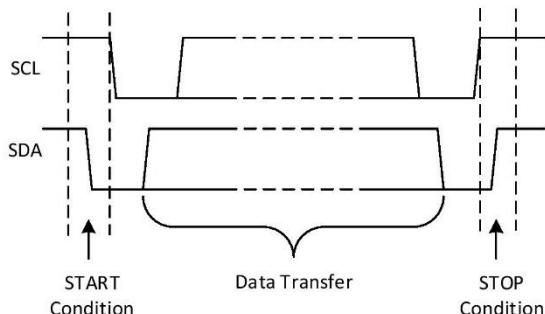


Figure 5. Example of START and STOP Condition

### 2.1.2 Repeated START Condition

A repeated START condition is similar to a START condition and is used in place of a back-to-back STOP then START condition. It looks identical to a START condition, but differs from a START condition because it happens before a STOP condition (when the bus is not idle). This is useful for when the master wishes to start a new communication, but does not wish to let the bus go idle with the STOP condition, which has the chance of the master losing control of the bus to another master (in multi-master environments).

### 2.2 Data Validity and Byte Format

One data bit is transferred during each clock pulse of the SCL. One byte is comprised of eight bits on the SDA line. A byte may either be a device address, register address, or data written to or read from a slave. Data is transferred Most Significant Bit (MSB) first. Any number of data bytes can be transferred from the master to slave between the START and STOP conditions. Data on the SDA line must remain stable during the high phase of the clock period, as changes in the data line when the SCL is high are interpreted as control commands (START or STOP).

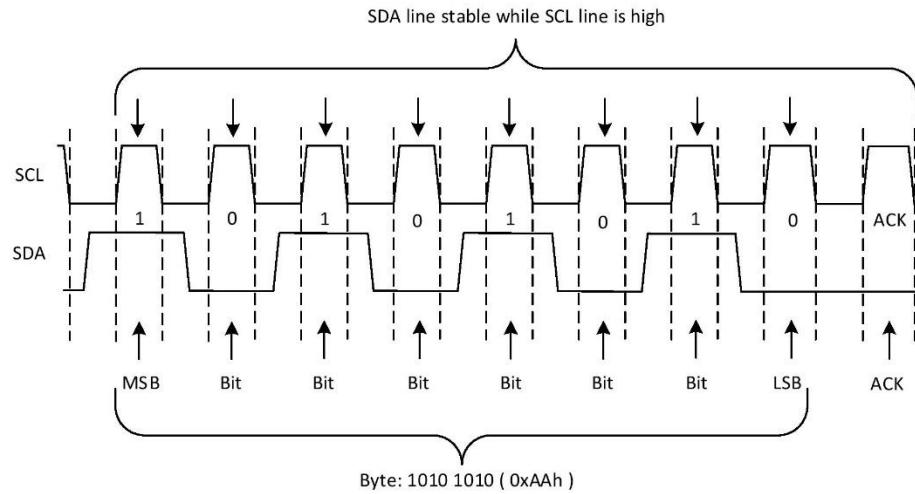


Figure 6. Example of Single Byte Data Transfer

### 2.3 Acknowledge (ACK) and Not Acknowledge (NACK)

Each byte of data (including the address byte) is followed by one ACK bit from the receiver. The ACK bit allows the receiver to communicate to the transmitter that the byte was successfully received and another byte may be sent.

Before the receiver can send an ACK, the transmitter must release the SDA line. To send an ACK bit, the receiver shall pull down the SDA line during the low phase of the ACK/NACK-related clock period (period 9), so that the SDA line is stable low during the high phase of the ACK/NACK-related clock period. Setup and hold times must be taken into account.

When the SDA line remains high during the ACK/NACK-related clock period, this is interpreted as a NACK. There are several conditions that lead to the generation of a NACK:

1. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master.
2. During the transfer, the receiver gets data or commands that it does not understand.
3. During the transfer, the receiver cannot receive any more data bytes.
4. A master-receiver is done reading data and indicates this to the slave through a NACK.

*I<sup>2</sup>C Data*

www.ti.com

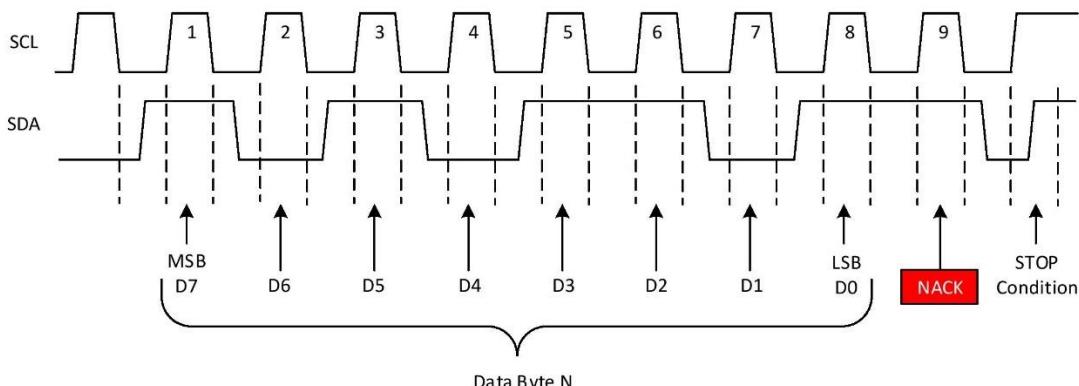


Figure 7. Example NACK Waveform

### 3 I<sup>2</sup>C Data

Data must be sent and received to or from the slave devices, but the way that this is accomplished is by reading or writing to or from registers in the slave device.

Registers are locations in the slave's memory which contain information, whether it be the configuration information, or some sampled data to send back to the master. The master must write information into these registers in order to instruct the slave device to perform a task.

While it is common to have registers in I<sup>2</sup>C slaves, please note that not all slave devices will have registers. Some devices are simple and contain only 1 register, which may be written directly to by sending the register data immediately after the slave address, instead of addressing a register. An example of a single-register device would be an 8-bit I<sup>2</sup>C switch, which is controlled via I<sup>2</sup>C commands. Since it has 1 bit to enable or disable a channel, there is only 1 register needed, and the master merely writes the register data after the slave address, skipping the register number.



### 3.1 Writing to a Slave On The I<sup>2</sup>C Bus

To write on the I<sup>2</sup>C bus, the master will send a start condition on the bus with the slave's address, as well as the last bit (the R/W bit) set to 0, which signifies a write. After the slave sends the acknowledge bit, the master will then send the register address of the register it wishes to write to. The slave will acknowledge again, letting the master know it is ready. After this, the master will start sending the register data to the slave, until the master has sent all the data it needs to (sometimes this is only a single byte), and the master will terminate the transmission with a STOP condition.

Figure 8 shows an example of writing a single byte to a slave register.

- Master Controls SDA Line
- Slave Controls SDA Line

#### Write to One Register in a Device

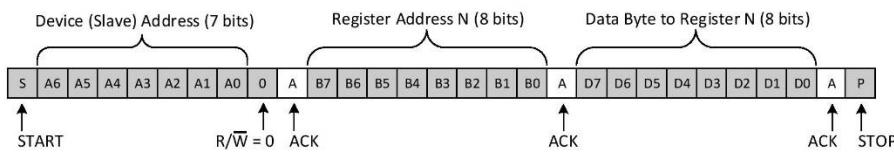


Figure 8. Example I<sup>2</sup>C Write to Slave Device's Register

### 3.2 Reading From a Slave On The I<sup>2</sup>C Bus

Reading from a slave is very similar to writing, but with some extra steps. In order to read from a slave, the master must first instruct the slave which register it wishes to read from. This is done by the master starting off the transmission in a similar fashion as the write, by sending the address with the R/W bit equal to 0 (signifying a write), followed by the register address it wishes to read from. Once the slave acknowledges this register address, the master will send a START condition again, followed by the slave address with the R/W bit set to 1 (signifying a read). This time, the slave will acknowledge the read request, and the master releases the SDA bus, but will continue supplying the clock to the slave. During this part of the transaction, the master will become the master-receiver, and the slave will become the slave-transmitter.

The master will continue sending out the clock pulses, but will release the SDA line, so that the slave can transmit data. At the end of every byte of data, the master will send an ACK to the slave, letting the slave know that it is ready for more data. Once the master has received the number of bytes it is expecting, it will send a NACK, signaling to the slave to halt communications and release the bus. The master will follow this up with a STOP condition.

Figure 9 shows an example of reading a single byte from a slave register.

- Master Controls SDA Line
- Slave Controls SDA Line

#### Read From One Register in a Device

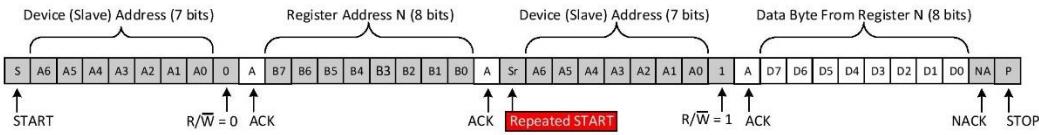


Figure 9. Example I<sup>2</sup>C Read from Slave Device's Register

### IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have *not* been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

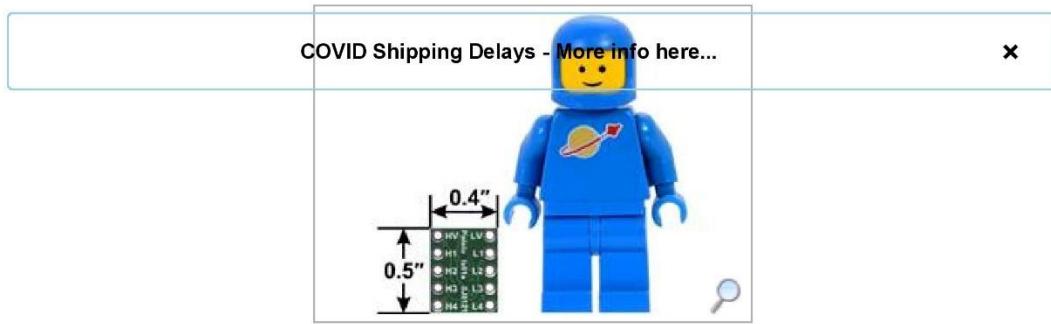
TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products	Applications
Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>
	<b>TI E2E Community</b>
	<a href="http://e2e.ti.com">e2e.ti.com</a>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
 Copyright © 2015, Texas Instruments Incorporated

19.4.2021

Logic Level Shifter, 4-Channel, Bidirectional



Logic level shifter, 4-channel, bidirectional, bottom view with dimensions next to a LEGO Minifigure for size reference

## Overview

As digital devices get smaller and faster, once ubiquitous 5 V logic has given way to ever lower-voltage standards like 3.3 V, 2.5 V, and even 1.8 V, leading to an ecosystem of components that need a little help talking to each other. For example, a 5 V part might fail to read a 3.3 V signal as high, and a 3.3 V part might be damaged by a 5 V signal. This level shifter solves these problems by offering bidirectional voltage translation of up to four independent signals, converting between logic levels as low as 1.5 V on the lower-voltage side and as high as 18 V on the higher-voltage side, and its compact size and breadboard-compatible pin spacing make it easy to integrate into projects.

The logic high levels on each side of the shifter are achieved by 10 kΩ pull-up resistors to their respective supplies; these provide quick enough rise times to allow decent conversion of fast mode (400 kHz) I<sup>2</sup>C signals or other similarly fast digital interfaces (e.g. SPI or asynchronous TTL serial). External pull-ups can be added to speed up the rise time further at the expense of higher current draw. See the schematic diagram below for more information.

## Features

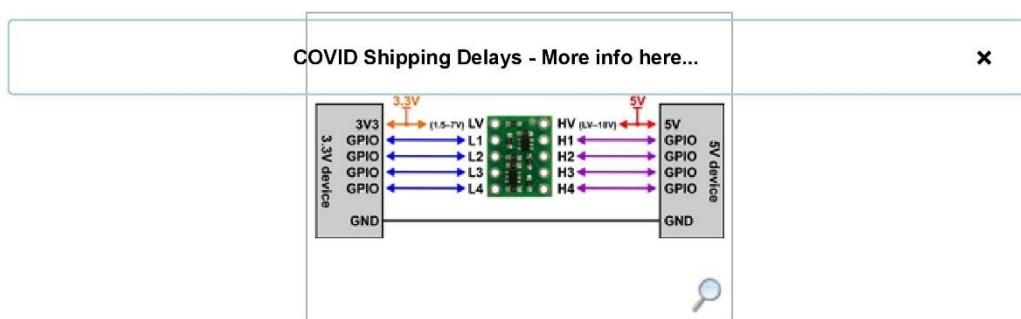
- Dual-supply bus translation:
  - Lower-voltage (LV) supply can be 1.5 V to 7 V
  - Higher-voltage (HV) supply can be LV to 18 V
- Four bidirectional channels
- Small size: 0.4" × 0.5" × 0.08" (13 mm × 10 mm × 2 mm)
- Breadboard-compatible pin spacing

## Connections

<https://www.robotgear.com.au/Product.aspx/Details/4407-Logic-Level-Shifter-4-Channel-Bidirectional> 2/6

19.4.2021

Logic Level Shifter, 4-Channel, Bidirectional

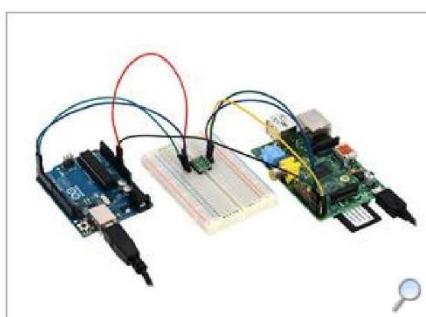


Example wiring diagram for connecting 5 V and 3.3 V devices through the 4-channel bidirectional logic level shifter

This logic level converter requires two supply voltages: the lower-voltage logic supply (1.5 V to 7 V) connects to the LV pin and the higher-voltage supply (LV to 18 V) connects to the HV pin. The HV supply must be higher than the LV supply for proper operation. Logic low voltages will pass directly from Hx to the corresponding Lx (and vice versa), while logic high voltages will be converted between the HV level to the LV level as the signal passes from Hx to Lx or Lx to Hx.

The level shifter circuit does not require a ground connection to either device, so there are no ground pins on the board. (Some competing level shifter modules provide ground connections that simply act as a pass-through; we have opted to leave these off and make the board smaller.) The two devices being connected through the level shifter must still share a common ground.

The picture below shows a level-shifted TTL serial connection (RX and TX) between a 5 V Arduino Uno and a 3.3 V Raspberry Pi .



Using the 4-channel bidirectional logic level shifter to create a serial connection between a 5 V Arduino Uno and a 3.3 V Raspberry Pi .

## Included hardware

A 0.1"-pitch male header strip is included for use with this board. The strip can be broken into smaller pieces and soldered in so the board can be used with perfboards, breadboards, or 0.1" female connectors. Alternatively, wires can be soldered directly to the board for more compact installations. The connections are labelled on the back side of the of the PCB, so you might find it more convenient to solder the pins in the way that allows the labelled side to be facing up.

19.4.2021

Logic Level Shifter, 4-Channel, Bidirectional

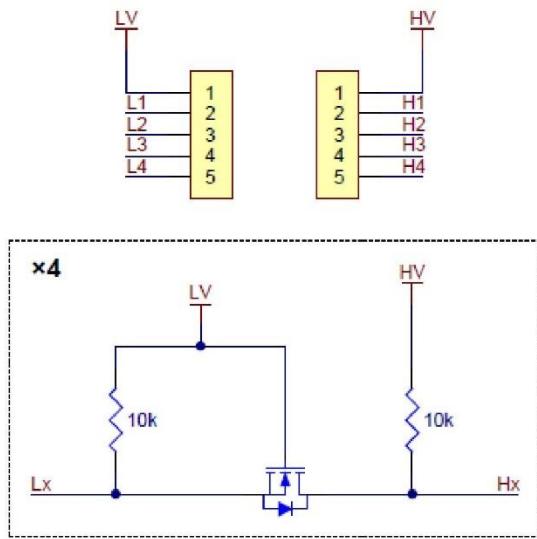


Logic level shifter, 4-channel, bidirectional, with included hardware

Logic level shifters in breadboard, one soldered label-side up and the other soldered component-side up

## Schematic diagram

The logic level conversion is accomplished with a simple circuit consisting of a single n-channel MOSFET and a pair of 10 k $\Omega$  pull-up resistors for each channel. When Lx is driven low, the MOSFET turns on and the zero passes through to Hx. When Hx is driven low, Lx is also driven low through the MOSFET's body diode, at which point the MOSFET turns on. In all other cases, both Lx and Hx are pulled high to their respective logic supply voltages. External pull-ups can be added to speed up the rise time. This same circuit is detailed in NXP's application note on I<sup>2</sup>C bus level-shifting techniques , and we have used it before on carrier boards for 3.3 V sensors with I<sup>2</sup>C interfaces – like the MinIMU-9 – to enable them to work directly with both 3.3 V and 5 V systems.



This schematic is also available as a downloadable PDF (135k pdf).

<https://www.robotgear.com.au/Product.aspx/Details/4407-Logic-Level-Shifter-4-Channel-Bidirectional> 4/6

	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013
---	---	---

#### 4.4 Register 27 – Gyroscope Configuration GYRO\_CONFIG

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1B	27	XG_ST	YG_ST	ZG_ST	FS_SEL[1:0]	-	-	-	-

Description:

This register is used to trigger gyroscope self-test and configure the gyroscopes' full scale range.

Gyroscope self-test permits users to test the mechanical and electrical portions of the gyroscope. The self-test for each gyroscope axis can be activated by controlling the XG\_ST, YG\_ST, and ZG\_ST bits of this register. Self-test for each axis may be performed independently or all at the same time.

When self-test is activated, the on-board electronics will actuate the appropriate sensor. This actuation will move the sensor's proof masses over a distance equivalent to a pre-defined Coriolis force. This proof mass displacement results in a change in the sensor output, which is reflected in the output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test limits for each gyroscope axis is provided in the electrical characteristics tables of the MPU-6000/MPU-6050 Product Specification document. When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values specified in the document, the part is deemed to have failed self-test.

FS\_SEL selects the full scale range of the gyroscope outputs according to the following table.

FS_SEL	Full Scale Range
0	$\pm 250^{\circ}/s$
1	$\pm 500^{\circ}/s$
2	$\pm 1000^{\circ}/s$
3	$\pm 2000^{\circ}/s$

Bits 2 through 0 are reserved.

Parameters:

- |        |  |
|--------|--|
| XG_ST  | Setting this bit causes the X axis gyroscope to perform self test. |
| YG_ST  | Setting this bit causes the Y axis gyroscope to perform self test. |
| ZG_ST  | Setting this bit causes the Z axis gyroscope to perform self test. |
| FS_SEL | 2-bit unsigned value. Selects the full scale range of gyroscopes.  |

	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013
---	---	---

#### 4.17 Registers 59 to 64 – Accelerometer Measurements

ACCEL\_XOUT\_H, ACCEL\_XOUT\_L, ACCEL\_YOUT\_H, ACCEL\_YOUT\_L, ACCEL\_ZOUT\_H, and ACCEL\_ZOUT\_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59								ACCEL_XOUT[15:8]
3C	60								ACCEL_XOUT[7:0]
3D	61								ACCEL_YOUT[15:8]
3E	62								ACCEL_YOUT[7:0]
3F	63								ACCEL_ZOUT[15:8]
40	64								ACCEL_ZOUT[7:0]

**Description:**

These registers store the most recent accelerometer measurements.

Accelerometer measurements are written to these registers at the Sample Rate as defined in Register 25.

The accelerometer measurement registers, along with the temperature measurement registers, gyroscope measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the accelerometer sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Each 16-bit accelerometer measurement has a full scale defined in ACCEL\_FS (Register 28). For each full scale setting, the accelerometers' sensitivity per LSB in ACCEL\_xOUT is shown in the table below.

AFS_SEL	Full Scale Range	LSB Sensitivity
0	$\pm 2g$	16384 LSB/g
1	$\pm 4g$	8192 LSB/g
2	$\pm 8g$	4096 LSB/g
3	$\pm 16g$	2048 LSB/g

**Parameters:**

- ACCEL\_XOUT      16-bit 2's complement value.  
Stores the most recent X axis accelerometer measurement.
- ACCEL\_YOUT      16-bit 2's complement value.  
Stores the most recent Y axis accelerometer measurement.
- ACCEL\_ZOUT      16-bit 2's complement value.  
Stores the most recent Z axis accelerometer measurement.

	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013
---	---	---

#### 4.19 Registers 67 to 72 – Gyroscope Measurements

GYRO\_XOUT\_H, GYRO\_XOUT\_L, GYRO\_YOUT\_H, GYRO\_YOUT\_L, GYRO\_ZOUT\_H, and GYRO\_ZOUT\_L

Type: Read Only

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
43	67								GYRO_XOUT[15:8]
44	68								GYRO_XOUT[7:0]
45	69								GYRO_YOUT[15:8]
46	70								GYRO_YOUT[7:0]
47	71								GYRO_ZOUT[15:8]
48	72								GYRO_ZOUT[7:0]

**Description:**

These registers store the most recent gyroscope measurements.

Gyroscope measurements are written to these registers at the Sample Rate as defined in Register 25.

These gyroscope measurement registers, along with the accelerometer measurement registers, temperature measurement registers, and external sensor data registers, are composed of two sets of registers: an internal register set and a user-facing read register set.

The data within the gyroscope sensors' internal register set is always updated at the Sample Rate. Meanwhile, the user-facing read register set duplicates the internal register set's data values whenever the serial interface is idle. This guarantees that a burst read of sensor registers will read measurements from the same sampling instant. Note that if burst reads are not used, the user is responsible for ensuring a set of single byte reads correspond to a single sampling instant by checking the Data Ready interrupt.

Each 16-bit gyroscope measurement has a full scale defined in FS\_SEL (Register 27). For each full scale setting, the gyroscopes' sensitivity per LSB in GYRO\_xOUT is shown in the table below:

FS_SEL	Full Scale Range	LSB Sensitivity
0	± 250 °/s	131 LSB/°/s
1	± 500 °/s	65.5 LSB/°/s
2	± 1000 °/s	32.8 LSB/°/s
3	± 2000 °/s	16.4 LSB/°/s

**Parameters:**

GYRO\_XOUT 16-bit 2's complement value.

Stores the most recent X axis gyroscope measurement.

GYRO\_YOUT 16-bit 2's complement value.

Stores the most recent Y axis gyroscope measurement.

GYRO\_ZOUT 16-bit 2's complement value.

Stores the most recent Z axis gyroscope measurement.

	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013
---	---	---

#### 4.28 Register 107 – Power Management 1 PWR\_MGMT\_1

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Description:

This register allows the user to configure the power mode and clock source. It also provides a bit for resetting the entire device, and a bit for disabling the temperature sensor.

By setting *SLEEP* to 1, the MPU-60X0 can be put into low power sleep mode. When *CYCLE* is set to 1 while *SLEEP* is disabled, the MPU-60X0 will be put into Cycle Mode. In Cycle Mode, the device cycles between sleep mode and waking up to take a single sample of data from accelerometer at a rate determined by *LP\_WAKE\_CTRL* (register 108). To configure the wake frequency, use *LP\_WAKE\_CTRL* within the Power Management 2 register (Register 108).

An internal 8MHz oscillator, gyroscope based clock, or external sources can be selected as the MPU-60X0 clock source. When the internal 8 MHz oscillator or an external source is chosen as the clock source, the MPU-60X0 can operate in low power modes with the gyroscopes disabled.

Upon power up, the MPU-60X0 clock source defaults to the internal oscillator. However, it is highly recommended that the device be configured to use one of the gyroscopes (or an external clock source) as the clock reference for improved stability. The clock source can be selected according to the following table.

CLKSEL	Clock Source
0	Internal 8MHz oscillator
1	PLL with X axis gyroscope reference
2	PLL with Y axis gyroscope reference
3	PLL with Z axis gyroscope reference
4	PLL with external 32.768kHz reference
5	PLL with external 19.2MHz reference
6	Reserved
7	Stops the clock and keeps the timing generator in reset

For further information regarding the MPU-60X0 clock source, please refer to the MPU-6000/MPU-6050 Product Specification document.

Bit 4 is reserved.

	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013
---	---	---

**Parameters:**

<i>DEVICE_RESET</i>	When set to 1, this bit resets all internal registers to their default values. The bit automatically clears to 0 once the reset is done. The default values for each register can be found in Section 3.
<i>SLEEP</i>	When set to 1, this bit puts the MPU-60X0 into sleep mode.
<i>CYCLE</i>	When this bit is set to 1 and <i>SLEEP</i> is disabled, the MPU-60X0 will cycle between sleep mode and waking up to take a single sample of data from active sensors at a rate determined by <i>LP_WAKE_CTRL</i> (register 108).
<i>TEMP_DIS</i>	When set to 1, this bit disables the temperature sensor.
<i>CLKSEL</i>	3-bit unsigned value. Specifies the clock source of the device.

**Note:**

When using SPI interface, user should use *DEVICE\_RESET* (register 107) as well as *SIGNAL\_PATH\_RESET* (register 104) to ensure the reset is performed properly. The sequence used should be:

1. Set *DEVICE\_RESET* = 1 (register *PWR\_MGMT\_1*)
2. Wait 100ms
3. Set *GYRO\_RESET* = *ACCEL\_RESET* = *TEMP\_RESET* = 1 (register *SIGNAL\_PATH\_RESET*)
4. Wait 100ms

	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013
---	---	---

#### 4.29 Register 108 – Power Management 2 PWR\_MGMT\_2

Type: Read/Write

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6C	108	LP_WAKE_CTRL[1:0]	STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG	

**Description:**

This register allows the user to configure the frequency of wake-ups in Accelerometer Only Low Power Mode. This register also allows the user to put individual axes of the accelerometer and gyroscope into standby mode.

The MPU-60X0 can be put into Accelerometer Only Low Power Mode using the following steps:

- (i) Set CYCLE bit to 1
- (ii) Set SLEEP bit to 0
- (iii) Set TEMP\_DIS bit to 1
- (iv) Set STBY\_XG, STBY\_YG, STBY\_ZG bits to 1

All of the above bits can be found in Power Management 1 register (Register 107).

In this mode, the device will power off all devices except for the primary I<sup>2</sup>C interface, waking only the accelerometer at fixed intervals to take a single measurement. The frequency of wake-ups can be configured with LP\_WAKE\_CTRL as shown below.

LP_WAKE_CTRL	Wake-up Frequency
0	1.25 Hz
1	5 Hz
2	20 Hz
3	40 Hz

For further information regarding the MPU-6050's power modes, please refer to Register 107.

The user can put individual accelerometer and gyroscopes axes into standby mode by using this register. If the device is using a gyroscope axis as the clock source and this axis is put into standby mode, the clock source will automatically be changed to the internal 8MHz oscillator.

**Parameters:**

LP_WAKE_CTRL	2-bit unsigned value. Specifies the frequency of wake-ups during Accelerometer Only Low Power Mode.
STBY_XA	When set to 1, this bit puts the X axis accelerometer into standby mode.
STBY_YA	When set to 1, this bit puts the Y axis accelerometer into standby mode.
STBY_ZA	When set to 1, this bit puts the Z axis accelerometer into standby mode.
STBY_XG	When set to 1, this bit puts the X axis gyroscope into standby mode.
STBY_YG	When set to 1, this bit puts the Y axis gyroscope into standby mode.
STBY_ZG	When set to 1, this bit puts the Z axis gyroscope into standby mode.

	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	---	---

### 3 Product Overview

#### 3.1 MPU-60X0 Overview

MotionInterface™ is becoming a “must-have” function being adopted by smartphone and tablet manufacturers due to the enormous value it adds to the end user experience. In smartphones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, MotionTracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for MotionInterface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point.

The MPU-60X0 is the world's first integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I<sup>2</sup>C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output. The MPU-60X0 MotionTracking device, with its 6-axis integration, on-board MotionFusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers. The MPU-60X0 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I<sup>2</sup>C port. The MPU-60X0 is footprint compatible with the MPU-30X0 family.

The MPU-60X0 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g.

An on-chip 1024 Byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-60X0 uniquely enables low-power MotionInterface applications in portable applications with reduced processing requirements for the system processor. By providing an integrated MotionFusion output, the DMP in the MPU-60X0 offloads the intensive MotionProcessing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output.

Communication with all registers of the device is performed using either I<sup>2</sup>C at 400kHz or SPI at 1MHz (MPU-6000 only). For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz (MPU-6000 only). Additional features include an embedded temperature sensor and an on-chip oscillator with ±1% variation over the operating temperature range.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the MPU-60X0 package size down to a revolutionary footprint of 4x4x0.9mm (QFN), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000g shock tolerance, and has programmable low-pass filters for the gyroscopes, accelerometers, and the on-chip temperature sensor.

For power supply flexibility, the MPU-60X0 operates from VDD power supply voltage range of 2.375V-3.46V. Additionally, the MPU-6050 provides a VLOGIC reference pin (in addition to its analog supply pin: VDD), which sets the logic levels of its I<sup>2</sup>C interface. The VLOGIC voltage may be 1.8V±5% or VDD.

The MPU-6000 and MPU-6050 are identical, except that the MPU-6050 supports the I<sup>2</sup>C serial interface only, and has a separate VLOGIC reference pin. The MPU-6000 supports both I<sup>2</sup>C and SPI interfaces and has a single supply pin, VDD, which is both the device's logic reference supply and the analog supply for the part. The table below outlines these differences:

	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	---	---

**Primary Differences between MPU-6000 and MPU-6050**

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I <sup>2</sup> C, SPI	I <sup>2</sup> C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA

	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	---	---

## 5 Features

### 5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$ , and  $\pm 2000^\circ/\text{sec}$
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5 $\mu\text{A}$
- Factory calibrated sensitivity scale factor
- User self-test

### 5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$  and  $\pm 16g$
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500 $\mu\text{A}$
- Low power accelerometer mode current: 10 $\mu\text{A}$  at 1.25Hz, 20 $\mu\text{A}$  at 5Hz, 60 $\mu\text{A}$  at 20Hz, 110 $\mu\text{A}$  at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

### 5.3 Additional Features

The MPU-60X0 includes the following additional features:

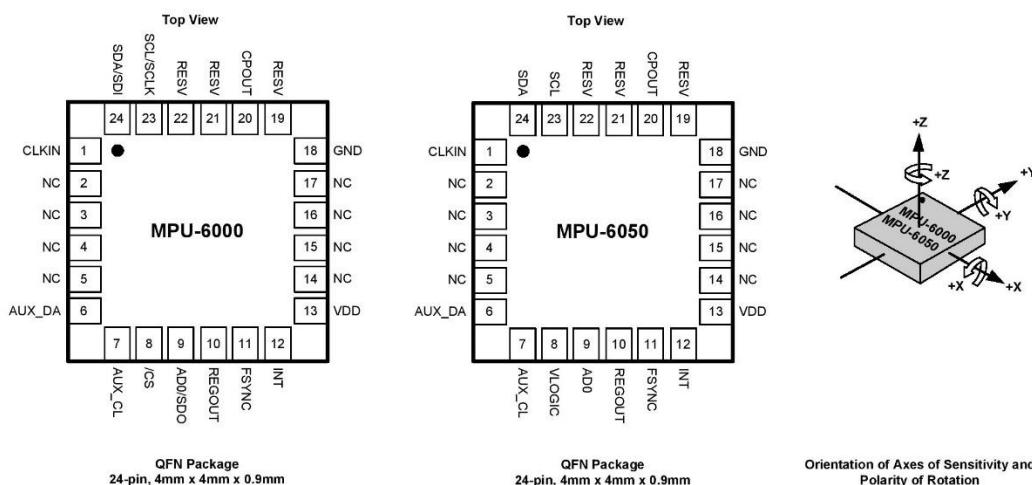
- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I<sup>2</sup>C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I<sup>2</sup>C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 g shock tolerant
- 400kHz Fast Mode I<sup>2</sup>C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)

	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013
---	---	---

## 7 Applications Information

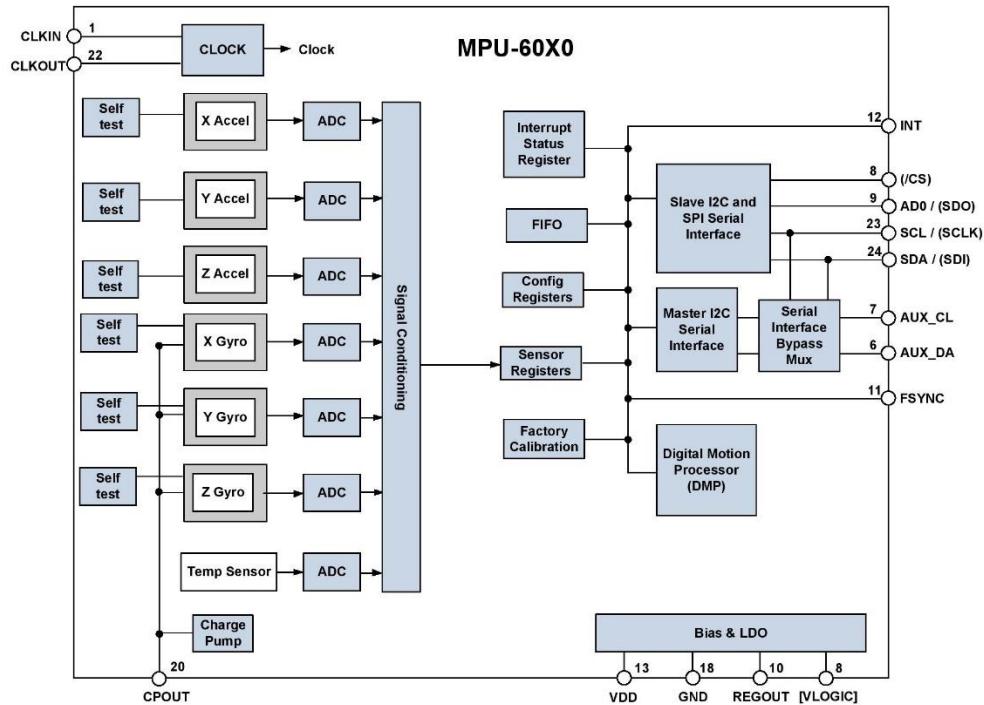
### 7.1 Pin Out and Signal Description

Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
1	Y	Y	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	Y	Y	AUX_DA	I <sup>2</sup> C master serial data, for connecting to external sensors
7	Y	Y	AUX_CL	I <sup>2</sup> C Master serial clock, for connecting to external sensors
8	Y		/CS	SPI chip select (0=SPI mode)
8		Y	VLOGIC	Digital I/O supply voltage
9	Y		AD0 / SDO	I <sup>2</sup> C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I <sup>2</sup> C Slave Address LSB (AD0)
10	Y	Y	REGOUT	Regulator filter capacitor connection
11	Y	Y	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	Y	Y	INT	Interrupt digital output (totem pole or open-drain)
13	Y	Y	VDD	Power supply voltage and Digital I/O supply voltage
18	Y	Y	GND	Power supply ground
19, 21	Y	Y	RESV	Reserved. Do not connect.
20	Y	Y	CPOUT	Charge pump capacitor connection
22	Y	Y	RESV	Reserved. Do not connect.
23	Y		SCL / SCLK	I <sup>2</sup> C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I <sup>2</sup> C serial clock (SCL)
24	Y		SDA / SDI	I <sup>2</sup> C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I <sup>2</sup> C serial data (SDA)
2, 3, 4, 5, 14, 15, 16, 17	Y	Y	NC	Not internally connected. May be used for PCB trace routing.





### 7.5 Block Diagram



Note: Pin names in round brackets ( ) apply only to MPU-6000  
Pin names in square brackets [ ] apply only to MPU-6050

### 7.6 Overview

The MPU-60X0 is comprised of the following key blocks and functions:

- Three-axis MEMS rate gyroscope sensor with 16-bit ADCs and signal conditioning
- Three-axis MEMS accelerometer sensor with 16-bit ADCs and signal conditioning
- Digital Motion Processor (DMP) engine
- Primary I<sup>2</sup>C and SPI (MPU-6000 only) serial communications interfaces
- Auxiliary I<sup>2</sup>C serial interface for 3<sup>rd</sup> party magnetometer & other sensors
- Clocking
- Sensor Data Registers
- FIFO
- Interrupts
- Digital-Output Temperature Sensor
- Gyroscope & Accelerometer Self-test
- Bias and LDO
- Charge Pump

## נופחים

**תוכנית פסיקה עבור חיישני המרחק:** ( מtabסס על החומרה "התנסות בארדואינו אופק 2020" )

```
#define trigPin 4
#define echoPin 2 //בבדיקה זה מתאפשרת תוכנית פסיקה
#define led 3
unsigned long duration; //משתנה עבור משך רוחב הפולס
int distance=0; //משתנה עבור המרחק הנמדד
unsigned long timer=0; //טיימר עבור מדידה כל זמן קבוע

//הפונקציה נזנת דרבון 10us בambil trigger
void trigUs() {
    digitalWrite(trigPin, HIGH); //יצירת דרבון
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    attachInterrupt(0, startCount, RISING); // startCount () בצע ()
}

כאשר יש עליה בבדיקה 2 בצע () בצע
{

//הפונקציה מפעילה את מדידת הזמן
void startCount() {
    duration=micros(); //התחלת רוחב הפולס
    detachInterrupt(0); // חסום פסיקה מס' 0
    attachInterrupt(0, measurement, FALLING); // measurement() בצע
}

כאשר יש ירידה בבדיקה 2 בצע () בצע
{
    duration=micros()-duration; //חישוב רוחב הפולס
    distance=duration/58; // חישוב המרחק בס"מ
    detachInterrupt(0); // חסום פסיקה מס' 0
}

//הפונקציה מודדת את הזמן וממחשבת את המרחק
void measurement()
{
    duration=micros(); //חישוב רוחב הפולס
    distance=duration/58; // חישוב המרחק בס"מ
    detachInterrupt(0); // חסום פסיקה מס' 0
}

void setup() {
    Serial.begin(9600); //קבע את קצב הシリינר ל 115200 סיביות בשניה
    pinMode(trigPin, OUTPUT); //הגדרות ואתחול חיישן מרחק
    pinMode(echoPin, INPUT);
    trigUs(); //דרבן ראשי של 10us ואפשר פסיקה
}

void loop() {
    if(millis()-timer > 50) { // 50mS מדידת מרחק כל 50ms
        Serial.print(distance); // הדפס את המרחק בס"מ
        Serial.println(" cm"); // הדפס ס"מ וunaroor שורה
        trigUs(); // 10us 'דרבן' של 10us
        timer=millis();
    }
    if (distance<15) { // אם המרחק קטן מ 15 ס"מ
        digitalWrite(led,HIGH); // הידק את הלד
    } else {
        digitalWrite(led,LOW); // כבה את הלד
    }
}
```

התוכנית בקובץ **deg6050.ino** עבור חישוב הזווית:

```
#include <Servo.h> // ספריית מנוע סרוו
#include <Wire.h> // ספרייה I2C

Servo myservo; // אובייקט של המצלקה סרוו עבור המנוע
int servoPin = 5;

const int MPU_addr = 0x68; // כתובת הרכיב MPU6050
double AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ; // משתנים עבור הנתונים שנקריא מהרכיב
unsigned long timer; // טיימר עבור מדידת הזמן בין חישובי הזווית
double compAngleX, compAngleY; // משתנים עבור הזוויות שנחשב
#define degconvert 57.2957786 // המרת מעלות למדיאנים

void setup() {
    myservo.attach(servoPin); // הגדרות ואתחולן מנוע
    myservo.write(90);

    //הגדירות ואתחולן – גירוי
    Wire.begin(); //אתחולן יחידת I2C של הארדואינו
    Wire.setClock(400000UL); // 400kHz
    Wire.beginTransmission(MPU_addr); //התחלת תקשורת עם החישון – מקבל את הכתובת לשידור
    Wire.write(0x6B); //פניה לרגיסטר .6B
    Wire.write(0); //אייפוס הרגייסטר
    Wire.endTransmission(true); //סיום התקשורת – שחרור הקו ע"י שיליחת stop
    Serial.begin(115200); //קבע את קצב השידור ל115200 סיביות לשניה
    delay(100);

    //התחלת תקשורת עם החישון //
    Wire.write(0x3B); //פניה לרגיסטר 3B לצורך התחלת קריאת נתונים
    Wire.endTransmission(false); //הפסיק את התקשורת – אין שליחת stop
    Wire.requestFrom(MPU_addr, 14, true); //גישה לביטים בכתובת החישון, 14 קריאות, והפסיקת התקשורת בסיום

    AcX = Wire.read() << 8 | Wire.read(); // 0x3B & 0x3C
    AcY = Wire.read() << 8 | Wire.read(); // 0x3D & 0x3E
    AcZ = Wire.read() << 8 | Wire.read(); // 0x3F & 0x40
    Tmp = Wire.read() << 8 | Wire.read(); // 0x41 & 0x42
    GyX = Wire.read() << 8 | Wire.read(); // 0x43 & 0x44
    GyY = Wire.read() << 8 | Wire.read(); // 0x45 & 0x46
    GyZ = Wire.read() << 8 | Wire.read(); // 0x47 & 0x48

    double roll = atan2(AcY, AcZ) * degconvert;
    double pitch = atan2(-AcX, AcZ) * degconvert;
    compAngleX = roll;
    compAngleY = pitch;

    timer = micros(); //הפעלת הטיימר לחישוב הדuration
}

void loop() { //איסוף הנתונים כל פעם מחודש – תוכנית גירוי
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 14, true);
    AcX = Wire.read() << 8 | Wire.read(); // 0x3B & 0x3C
    AcY = Wire.read() << 8 | Wire.read(); // 0x3D & 0x3E
    AcZ = Wire.read() << 8 | Wire.read(); // 0x3F & 0x40
    Tmp = Wire.read() << 8 | Wire.read(); // 0x41 & 0x42
    GyX = Wire.read() << 8 | Wire.read(); // 0x43 & 0x44
    GyY = Wire.read() << 8 | Wire.read(); // 0x45 & 0x46
    GyZ = Wire.read() << 8 | Wire.read(); // 0x47 & 0x48
}
```

```
// חישוב הזמן. המרת מיקרו שניות לשניות
double dt = (double) (micros() - timer) / 1000000; timer = micros(); // הפעלת הטימר מחדש(dt

double roll = atan2(AcY, AcZ) * degconvert;
double pitch = atan2(-AcX, AcZ) * degconvert;

double gyroXrate = GyX / 131.0; // המרת הנתון מספר דיגיטלי ליחידה של מעלה
double gyroYrate = GyY / 131.0;

compAngleX = 0.99 * (compAngleX + gyroXrate * dt) + 0.01 * roll; // חישוב הזווית
compAngleY = 0.99 * (compAngleY + gyroYrate * dt) + 0.01 * pitch;

myservo.write(compAngleX);
}
```

## מקורות

<https://www.youtube.com/watch?v=XCyRXMvVSCw>

[https://dronebotworkshop.com/mpu-6050-level/#MPU-6050\\_with\\_Arduino](https://dronebotworkshop.com/mpu-6050-level/#MPU-6050_with_Arduino)

[https://sites.education.gov.il/cloud/home/cyber\\_israel/Documents/choveret\\_ezer\\_elec\\_tronica.pdf](https://sites.education.gov.il/cloud/home/cyber_israel/Documents/choveret_ezer_elec_tronica.pdf)

<https://www.robotgear.com.au/Product.aspx/Details/4407-Logic-Level-Shifter-4-Channel-Bidirectional>

[https://www.youtube.com/watch?v=hAB\\_X5EBtyU](https://www.youtube.com/watch?v=hAB_X5EBtyU)

[/https://www.arduino.cc/reference/en](https://www.arduino.cc/reference/en)