**סמל מוסד - 261065**

**עבודת גמר פרויקט בהיקף של 5 יחידות**

**הנדסת חשמל ואלקטרוניקה**

**בהתמחות:** מחשוב ובקרה

**הנושא: מד מהירות**

**מנחה הפרויקט : חברבר אייל**

**המגישים:**

שם: עידן קורז'ק ת.ז. 0215675059

שם: אור-ים ביבי - רובי ת.ז.

תאריך ההגשה: תשע"ז

# דף הצהרת הסטודנט ואישור המנחה :

**הצהרת הסטודנטים**אני \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ת.ז. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
אני \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ת.ז. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

החתום מטה, מצהיר בזאת שכל עבודת הפרויקט המוגשת בחוברת זו הינה פרי עבודתי בלבד, על בסיס הנחייתו של המנחה ותוך הסתמכות על מקורות הידע והמידע האחרים המצוינים בביבליוגרפיה המובאת בסיום חוברת זו.  
  
אני מודע לאחריות שהנני מקבל על עצמי ע"י חתימתי על הצהרה זו שכל הנאמר בה הינה אמת ורק אמת.

\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_  
 (חתימת מגישי החוברת)

**אישור המנחה**הריני מאשר הגשת החוברת להערכה \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (חתימת המנחה)

# הצעת נושא

שם הנושא:

רובוט משדר תמונה ונתונים לאחור בטכנולוגית IOT.

תיאור הפרויקט ואופן פעולתו:

רכב נשלט מרחוק ע"י ממשק שאנו מפתחים באנדרואיד סטודיו בשפת JAVA בשימוש בתקשורת Firebase המבוססת על Wi-Fi.

הרכב מצויד עם מצלמה (ESP CAM 32) המשדרת מידע לאחור לאפליקציה לצורך סריקת השטח שהמשתמש בוחר, בנוסף על כך יש חיישן טמפרטורה כדי לבדוק את תנאי השטח.

יעוד \ צורך ובעיות שהוא פותר:

מטרות הרכב להחליף כוח אדם ולהיכנס למקומות מסורבלים ולא נוחים לבני אדם, לאסוף מידע מהשטח ולהחזירו למשתמש לצורך בקרה.

שפות תכנות:

VHDL לתכנות הרכיב בר תכנות של חברת אלטרה.

C++ לתכנות רכיבי ה esp המשמשים להעברת מידע למשתמש.

JAVA לתכנות הממשק של האפליקציה.

רשימת רכיבים:

|  |  |  |
| --- | --- | --- |
| מנועי DC | דוחף זרם למנועים l293d | ESPCAM מצלמה אלחוטית |
| חיישן טמפרטורה lm75 | ESP32 בקר להעברת מידע ברשת | FPGA MAX10 ALTERA |
| לייזר 5v | מנוע סרבו MG90S | סוללה נטענת v9 |

# תרשים מלבנים סופי

PWM

SERVO

Temp Rx

### הסבר

בתרשים זה ניתן את חיבורי הרכיבים במעגל הפרויקט שלנו. אל הבקר הראשי שלנו ה – lite-De10 Fpga (אלטרה) נחבר בהצלבה את חיבורי המיקרו בקר esp32 (rx ו-tx). דרך חיבור זה המידע שקיים באלטרה נשלח אל הesp32 ולאחר מכן אל שרת ה-firebase אל האפליקציה , וכך גם ההפך, כלומר המידע נשלח מהאפליקציה אל שרת ה-firebase לאחר מכן אל ה-esp32 ואל האלטרה. המצלמה מחוברת ל5 וולט של הesp32 ומתחברת לאותו WIFI כמוהו ושולחת את המידע שלה לאפליקציה דרך פרוטוקול WIFI ושרת UDP.

לרכיב uln2803 מחובר לייזר שמודבק למנוע סרבו להזזת לפי רוצנו של המשתמש.

חיישן הטמפרטורה LM75 מחובר לאלטרה ומתקשר איתה דרך פרוטוקול I2C ומה שהוא קולט מומר לטמפרטורה בפרנהייט בתוכנית Temp Rx.

# מבוא

הפרויקט שעשינו הוא רכב חכם המשדר נתונים לאחור בעזרת IOT עם "תותח" לייזר.

הרכב מאפשר למשתמש לאסוף מידע מהשטח כמו תמונה וטמפרטורה כך המשתמש יכול לדעת את מצב השטח ולהחליט לפי שיפוטו אם כדאי "לפלוש" אליו, כך לא צריך לשלוח כוח אדם לשטח בשביל לראות את מצבו וזה מונע מאיבוד לא נחוץ של כוח אדם.

לרכב יש מצלמה המשדרת בשידור חי את התמונה שלה וניתן גם להפעיל את זיהוי התמונה המאפשר לזהות חפצים שהמצלמה משדרת בתמונה. על המצלמה יש לד שאפשר להפעיל לצורך ראייה יותר טובה של המצלמה, ישנו גם לייזר שניתן להזיז אותו 180 מעלות ו"לירות" איתו על חפצים.

הרכב בנוי מיחידת עיבוד ראשית שהיא בקר האלטרה ומיקרו בקר מסוג 32Esp המאפשר לנו תקשורת עם רשת האינטרנט. אל בקר האלטרה חובר המיקרו בקר 32Esp על ידי כניסות Rx ו – Tx של שני הבקרים. יחד עם זאת חובר אל בקר האלטרה שני מנועי DC עם מסילות כמו של טנק להזזת הרכב, וגם מנוע סרבו להזזת הלייזר. חיישן הטמפרטורה גם מחובר אל בקר האלטרה להצגת הטמפרטורה באפליקציה.

האפליקציה שלנו שאותה תכנתנו בשפת ג'אבה בסביבת אנדרואיד סטודיו, שולחת packet המכיל את כל הנתונים של המערכת אל ענן ה – Firebase ומשם אל מיקרו בקר שלנו 32Esp שתוכנת בשפת ++c על ידי תוכנת הארדואינו, הפאקט עובר דרכו אל הבקר הראשי בקר האלטרה ובתוך האלטרה הנתונים מתפרקים ונשלים לתוכניות המתאימות שלהם בתכנון ההיררכי.

נשתמש נשתמש ב - esp32 – cam המתוכנתת בשפת ++c על ידי תוכנת הארדואינו, בשביל לשלוח תמונה אל המשתמש המוצגת באפליקציה ובשביל לשלוט בלד של המצלמה. נתונים אלה לא עוברים באלטרה אלא עוברים בUDP socket שהמצלמה יוצרת בעזרת ספריית socket. המשתמש יכול לבחור אם להפעיל זיהוי תמונה או לא.

[דף הצהרת הסטודנט ואישור המנחה : 1](#_Toc135184229)

[הצעת נושא 2](#_Toc135184230)

[תרשים מלבנים סופי 3](#_Toc135184231)

[מבוא 4](#_Toc135184232)

[פרק 1- חומרה 7](https://d.docs.live.net/e2e87f845015ff01/פרויקט%20גמר%20יב/פרויקט%20גמר%20יב.docx#_Toc135184233)

[ורכיבי פרויקט וחומרה 8](#_Toc135184234)

[דוחף זרםL293d – 8](#_Toc135184235)

[מנוע Direct Current Motor - DC- 10](#_Toc135184236)

[PWM 12](#_Toc135184237)

[חיישן טמפרטורה – 18b20 לא נכנס בדגם הסופי 14](#_Toc135184238)

[בקר – ESP32 16](#_Toc135184239)

[פרוטוקולים- ESP32 20](#_Toc135184240)

[מייצב מתח של הESP (Voltage Regulator) 21](#_Toc135184241)

[ESP32 - CAM 22](#_Toc135184242)

[Altera 23](#_Toc135184243)

[FPGA-Field-Programmable Gate Array 23](#_Toc135184244)

[תקן תקשורת RS232 24](#_Toc135184245)

[LM75 – Digital Temperature Sensor 27](#_Toc135184246)

[Firebase – Google 30](#_Toc135184247)

[דוחף זרם – uln2803 31](#_Toc135184248)

[נורת להט (הוחלף בלד של ESP32CAM) 33](#_Toc135184249)

[מנוע Servo – MG90S 34](#_Toc135184250)

[פרק 2- מעגלים 36](https://d.docs.live.net/e2e87f845015ff01/פרויקט%20גמר%20יב/פרויקט%20גמר%20יב.docx#_Toc135184251)

[מעגלים חשמליים 37](#_Toc135184252)

[DC Motor to L293D 37](#_Toc135184253)

[ALTERA to ESP – TX to RX 38](#_Toc135184254)

[ESP32 – CAM to Firebase to Java App 39](#_Toc135184255)

[LM75 TEMP SENSOR to ALTERA And ESP32 40](#_Toc135184256)

[Servo motor to Altera 41](#_Toc135184257)

[Uln to laser to altera 42](#_Toc135184258)

[מייצב מתח של הESP 43](#_Toc135184259)

[מעגל שלם של כל הפרויקט 44](#_Toc135184261)

[פרק 3 - מדידות 46](https://d.docs.live.net/e2e87f845015ff01/פרויקט%20גמר%20יב/פרויקט%20גמר%20יב.docx#_Toc135184262)

[מדידות 47](#_Toc135184263)

[מדידות מנועי DC 47](#_Toc135184264)

[מדידות מנוע Servo 47](#_Toc135184265)

[מדידות Rx 48](#_Toc135184266)

[פרק 4 - VHDL 49](https://d.docs.live.net/e2e87f845015ff01/פרויקט%20גמר%20יב/פרויקט%20גמר%20יב.docx#_Toc135184267)

[VHDL 50](#_Toc135184268)

[תוכניות VHDL 53](#_Toc135184269)

[משדרTx – Transmitter – 53](#_Toc135184270)

[קולט Rx – Receiver – 55](#_Toc135184271)

[מחלק תדר – BaudRate 57](#_Toc135184272)

[מחלק מפענח תצוגה – Splitter 59](#_Toc135184273)

[מפענח תצוגה – (bin2hex) 7 Segment Decoder 61](#_Toc135184274)

[LED - נשימה 63](#_Toc135184275)

[PWM\_Motor 64](#_Toc135184276)

[PWM\_Ctrl 65](#_Toc135184277)

[pmod\_temp\_sensor\_tcn75a 67](#_Toc135184278)

[Bit\_9\_to\_8 – ממיר 9 סיביות ל - 8 71](#_Toc135184279)

[data\_spliter – מחלק מידע 72](#_Toc135184280)

[Servo\_control 73](#_Toc135184281)

[Low\_freq\_reset 75](#_Toc135184282)

[תכנון היררכי אלטרה 76](#_Toc135184283)

[PIN PLANNER 77](#_Toc135184284)

[פרק 5 – C++ & Java 78](https://d.docs.live.net/e2e87f845015ff01/פרויקט%20גמר%20יב/פרויקט%20גמר%20יב.docx#_Toc135184286)

[שפת++ C 79](#_Toc135184287)

[שפת – JAVA 80](#_Toc135184288)

[JVM – Java Virtual Machine 80](#_Toc135184289)

[OpenJDK 81](#_Toc135184290)

[JDK 81](#_Toc135184291)

[קוד בקר ה – ESP32 81](#_Toc135184292)

[קוד ESP32 – CAM 84](#_Toc135184293)

[שפת תוויות XML 91](#_Toc135184294)

[האפליקציה צילומי מסך 92](#_Toc135184295)

[קוד - Esp32CameraFragment 93](#_Toc135184296)

[קוד – MainActivity 115](#_Toc135184297)

[קוד - fragment\_camera xml 117](#_Toc135184298)

[קוד - activity\_main xml 120](#_Toc135184299)

[קוד - Settings gradle 120](#_Toc135184300)

[קוד – colors xml 120](#_Toc135184301)

[קוד - strings xml 120](#_Toc135184302)

[קוד - styles xml 121](#_Toc135184303)

[קוד - arrays xml 121](#_Toc135184304)

[פרק 6 – תקשורת מצלמה ואפליקציה 122](https://d.docs.live.net/e2e87f845015ff01/פרויקט%20גמר%20יב/פרויקט%20גמר%20יב.docx#_Toc135184305)

[פירוט הקלאסים באפליקציה 123](#_Toc135184306)

[מושגים 123](#_Toc135184307)

[איך התמונה נשלחת לאפליקציה? 124](#_Toc135184308)

[פרק 7 – תיעוד עבודה ורפלקציה 126](https://d.docs.live.net/e2e87f845015ff01/פרויקט%20גמר%20יב/פרויקט%20גמר%20יב.docx#_Toc135184309)

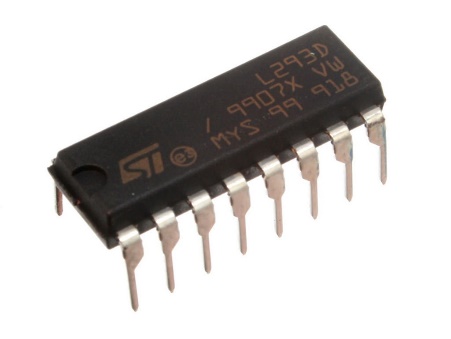
[תיעוד תקלות ותיקון 127](#_Toc135184310)

[תיעוד תהליך עבודה 128](#_Toc135184311)

[רפלקציה 131](#_Toc135184312)

[ביבליוגרפיה 131](#_Toc135184313)

# פרק 1- חומרה



# ורכיבי פרויקט וחומרה

## דוחף זרםL293d –

דוחף זרם שמספק הנעה של זרמים דו – כיוונים עד ל – mA600 במתחים מ – 4.5 ל – 36 וולט, רכיב זה נועד להניע עומסים אינדוקטיביים כגון ממסרים, סולונואידים, מנוע סטפר.

אנו צריכים רכיב זה על מנת לספק זרם למנועי גלגלי הרכב.

לרכיב יש 16 רגלים:

רגל 1 – רגל זו מאפשרת החלפה על כניסת המידע הרצויה ברגלים 2,7

רגל 2 – כניסת מספר 1

רגל 3 – יציאה מספר 1 (מחוברת ישירות ליציאה אחת של מנוע 1)

רגל 4 – אדמה (GND)

רגל 5 – אדמה (GND)

רגל 6 – יציאה מספר 2 (מחוברת ליציאה שניה של מנוע 1)

רגל 7 – כניסה מספר 2, שולטת ישירות על יציאה מספר 2 (נשלט על ידי מעגליים דיגיטליים)

רגל 8 – Vcc2(Vs) מתח להנעת המנוע (4.5 – 36 וולט)

רגל 9 - רגל זו מאפשרת החלפה על כניסת המידע הרצויה ברגלים 10,15

רגל 10 – כניסה מספר 3 שולטת ישירות על יציאה מספר 3 (נשלט על ידי מעגליים דיגיטליים)

רגל 11 – יציאה מספר 3 (מחוברת ישירות ליציאה אחת של מנוע 2)

רגל 12 – אדמה (GND)

רגל 13 – אדמה (GND)

רגל 14 – יציאה מספר 4 (מחוברת ישירות ליציאה שניה של מנוע 2)

רגל 15 – כניסה מספר 4 שולטת ישירות על יציאה מספר 4 (נשלט על ידי מעגליים דיגיטליים)

רגל 16 - Vcc2(Vss) מחוברת ל – 5+ וולט כדי לאפשר פונקציית IC (מיקרוצ'יפ)

### עיקרון הפעולה

ה-IC L293D מקבל אותות מהמיקרו-מעבד ומשדר את האות היחסי למנועים. יש לו שני פיני מתח (8,16), שאחד מהם משמש למשיכת זרם לעבודה של ה-L293D והשני משמש להפעלת מתח על המנועים.

### מבנה פנימי של הרכיב

L293D בנוי משני גשרי H המאפשרים לסובב שני מנועי DC בנפרד.

גשר H - גשר H הוא מעגל אלקטרוני שהופך את הקוטביות של המתח המופעל על העומס. מעגלים אלה נמצאים בשימוש נפוץ ברובוטיקה וביישומים אחרים להנעת מנועי DC בכיוון קדימה או אחורה. השם בא מהעיצוב הנפוץ בו האלמנטים הגמישים מסודרים כמו ענפי האות 'H' והסורגים מחוברים כמו צלב.

זרמים ומתחים – הרכיב יכול לספק עד ל – 600mA במתחים בטווח 4.5 – 36 וולט, ההספק המירבי הוא 21.6 וואט.

תדר – 5KHz שמיוצר על ידי מיקרוצ'יפ IC אם עוברים את מגבלת הזרם של ה – 600mA.

חוק לנץ - חוק לנץ הוא חוק פיזיקלי בתחום המגנטיות, לקביעת כיוון הכא"מ (כמות אנרגיה ליחידת מטען, נקרא גם "כוח אלקטרו-מניע") והזרם החשמלי המושרים על ידי שינוי בשטף המגנטי. השדה המגנטי המושרה על ידי זרם "שואף" לבטל את השינוי בשטף המגנטי היוצר אותו. למעשה, חוק לנץ הוא הסימן השלילי בחוק פאראדיי, אשר קובע את כיוון הכא"מ המושרה:

חוק זה נובע מחוק שימור האנרגיה: אילו הכא"מ המושרה היה גורם להגדלת השינוי בשטף המגנטי, אז השטף המגנטי היה גדל עוד יותר (בגלל הגידול בזרם), ויוצר בכך גידול נוסף בזרם, וכן הלאה עד אינסוף, תוך ביצוע עבודה אינסופית. כאשר השטף המגנטי בכריכה גדל נוצר בכריכה שדה מגנטי ההפוך בכיוונו לשדה בו נמצאת הכריכה. כאשר השטף המגנטי בכריכה קטן נוצר בכריכה שדה מגנטי השווה בכיוונו לשדה בו נמצאת הכריכה.

## מנוע Direct Current Motor - DC-

מנועי DC מופעלים על ידי זרם ומתח ישרים, שליטתם מתבצעת על ידי שינוי זרם ומתח האספקה אליהם. בדרך כלל משתמשים במנועים מהסוג הזה במערכות בעלות אורך חיים קצר או עם כמות הפעלה קטנה הוא עובד בכך שהוא מנוע הממיר את הזרם הישר לעבודה המכנית. זה עובד על העיקרון של חוק לורנץ, הקובע כי "המוליך נושא הזרם המוצב בשדה מגנטי וחשמלי חווה כוח". והכוח הזה הוא כוח לורנץ.

למנוע DC ישנם שתי חיבורים :

חיבור 1 - VCC כניסת המתח

חיבור 2 – GND אדמה

מנוע זרם ישר (DC) הוא מכונה חשמלית הממירה אנרגיה חשמלית לאנרגיה מכנית. מנוע זה מבוסס על עקרון האלקטרומגנטיות שמאפשר יצירת שדה במגנטי על ידי העברת זרם חשמלי דרך סליל.

מנוע DC מניע רוטור קבוע בתוך פיר הפלט באמצעות שדה מגנטי שיוצר את הזרם המתקבל.

סטטור – סדרה של סלילים סביב ליבה פרו-מגנטית מוצקה. הסטטור יכול להיות מורכב גם משני מגנטים חזקים. המגנטים ממוקמים מול הרוטור כך שהקטבים שלהם (צפוני ודרום) מנוגדים.

רוטור - ציר שעובר בתוך הסטטור ויש עליו מגנטים, או סלילים. ציר זה מסתובב בחופשיות. כאשר זרם זורם דרך סלילי הרוטור (או הסטטור), הוא יוצר שדה מגנטי סביב עצמו (דרך הליבה). שדה מגנטי זה מפעיל כוח על המוט כשהוא עובר דרכו, וגורם לו להסתובב בהשפעת מומנט סיבובי (כוח סיבוב). בקרת קטע נוכחית משלבת תנועות זוויתיות קטנות עם סיבוב מלא.



בתמונה הזאת הסליל הכחול ממוגנט כקוטב צפוני במקרה הזה, לכן הוא נדחה על ידי המגנט הצפוני. והסליל האדום ממוגנט כקוטב דרומי ונדחה על ידי הקוטב הדרומי וכך הרוטור מסתובב בכיוון השעון.



בתמונה הזאת הסלילים נמשכים על ידי כוח מגנטי לצבע הנוגד להם, כלומר הסליל הכחול נדחה מהמגנט הכחול ונמשך למגנט האדום וכך גם לסליל האדום. אפשר גם לראות את חיבור הזרם למנוע.



בתמונה זו הסלילים קרובים לקטבים ההפוכים להם במגנטיות והכוח נחלש, אבל במקום לעצור מתחלפים כיווני אספקת החשמל, והסלילים מחליפים את המגנטיות שלהם.

אופן הפעולה של מנוע DC הוא שבעזרת הזרם בעובר בסלילי הרוטור מכוון כך שהסלילים שמסתובבים מייצרים שדה אלקטרומגנטי עם קוטביות משתנה וכך אותו הקוטב מכוון למגנטים בסטטור. במצב הזה אחד מהמגנטים דוחה את הסליל שקרוב אליו והשני מושך את הסליל הקרוב אליו.

עקרון הפעולה של המנוע הינו אסינכרוני כלומר המנוע מסתובב במהירות שלא מסונכרנת לתדר שהוא מקבל. כאשר הוא נשמר בשדה מגנטי, מוליך נושא זרם צובר מומנט ומפתח נטייה לזוז. בקיצור, כאשר שדות חשמליים ושדות מגנטיים מתקשרים, נוצר כוח מכני. זהו העיקרון שעל פיו פועלים מנועי ה-DC

מנוע DC שאנחנו נעבוד איתו עובד עם 5V, צורך 1.42A ומספק 18000 סל”ד. על ככל שיש יותר עומס על המנוע כך ירד הסל”ד כדי להתנהל עם העומס הנוצר.

ציר אנכי – RPM, זה הוא סיבובים לדקה.

מהירות (N) - כחול

נמדד בסיבובים לדקה (סל"ד), קו ישר זה בשיפוע כלפי מטה מראה את הקשר בין מומנט ומהירות על פני כל פס הכוח (ראה קו כחול בדוגמה למעלה). מכיוון שהמהירות והמומנט הם פרופורציונליים בעקיפין זה לזה, קו זה יקטן באופן ליניארי ככל שהמומנט יגדל עד לנקודת העצירה, שבה המהירות תהיה 0 סל"ד. למהירות ולמומנט יש קשר הפוך. המהירות היא הגבוהה ביותר כאשר המנוע מייצר את המומנט הנמוך ביותר, וכאשר המומנט הוא הגבוה ביותר המנוע בקושי מסתובב.

יעילות (η) - אדום

יעילות היא קשר בין הספק המבוא להספק המוצא, נתון באחוזים (%). קו זה הוא פחות או יותר פרבולי עם הקודקוד יותר לכיוון ערכי המומנט הנמוכים יותר (ראה קו ורוד בדוגמה למעלה). זה בדרך כלל צריך מוקדם בטווח המומנט ואז יורד באיטיות ככל שהמנוע מתקרב למומנט העצירה שלו.

שימוש במנוע קרוב לשיא היעילות שלו מבטיח חיי מנוע אופטימליים וצריכת חשמל. עדיף להשתמש במנוע בשיא היעילות שלו או בסמוך לו. ככל שמנוע מתרחק מיעילות מרבית, הביצועים שלו הופכים לפחות אמינים.

מומנט (Torque) - ציר אופקי

בין אם נמדד בק"ג-ס"מ, lb-in או Nm, מומנט הוא כמות העומס שציר המוצא של המנוע או מנוע ההילוכים יכול להתגבר עליו. בעקומת ביצועים של מנוע DC, המומנט מיוצג בדרך כלל על ידי ציר ה-X.

המפגש בין קו המהירות (N) וציר ה-X הוא נקודת מומנט העצירה (T). זה המקום שבו המנוע מייצר את המומנט המרבי שלו ואינו יכול יותר להסתובב. וודאו שהכוח הדרוש מהמנוע קטן בהרבה מיכולת המומנט הכוללת (מומנט עצור), אחרת המנוע לא יעבוד כמתוכנן ויהיה לו סיכון גבוה להינזק.

זרם (I) - שחור

מיוצג על ידי הקו הישר העולה (ראה קו צהבהב בדוגמה לעיל), זה משקף את יציאת הזרם ממצבי חוסר עומס לתנאי עצור. קו זה מראה את הקשר הישיר בין זרם ומומנט. אם המערכת מודעת לכוח זה יהיה השיטה הטובה ביותר להפעיל את המנוע ביעילות שיא. זה מפיק את הביצועים המאוזנים ביותר מהמנוע תוך דרישה לכמות סבירה של זרם.

כוח פלט (P) - ירוק

אולי המספר הפשוט ביותר בגרף כולו, Power Output מראה כמה כוח (בוואט) יכול מנוע לספק (ראה קו ירוק בדוגמה למעלה).

אלקטרומגנטיות - סוג של אינטראקציה פיזיקלית המתרחשת בין חלקיקים טעונים במטען חשמלי. הכוח האלקטרומגנטי המיוצר על ידי שדות אלקטרומגנטיים מורכב משדות חשמליים ושדות מגנטיים, ואחראי לקרינה אלקטרומגנטית כמו אור. זהו אחד מארבעת הכוחות הבסיסיים (המכונים לעתים קרובות כוחות) בטבע, יחד עם ההשפעה החזקה, ההשפעה החלשה וכוח המשיכה. באנרגיות גבוהות יותר, הכוח החלש והכוח האלקטרומגנטי משולבים לכוח אלקטרומגנטי אחד.

## PWM

PWM או אפנון רוחב דופק היא טכניקה המאפשרת לנו להתאים את הערך הממוצע של המתח שעובר למכשיר האלקטרוני על ידי הפעלה וכיבוי של המתח בקצב מהיר. המתח הממוצע תלוי במחזור העבודה, או משך הזמן שהאות מופעל לעומת משך הזמן שהאות כבוי בפרק זמן בודד.



### H- Bridge

גשר H הוא מעגל אלקטרוני המשמש לשליטה בכיוון ובמהירות של מנוע DC. השם "גשר H" מגיע מצורת דיאגרמת המעגל, שנראית כמו האות "H".

מעגל H-bridge הבסיסי מורכב מארבעה מתגים (בדרך כלל טרנזיסטורים) המסודרים בתצורת "H". המתגים מחוברים למנוע בצורה כזו שהם יכולים להפוך את הקוטביות של המתח המופעל על המנוע. זה מאפשר להניע את המנוע בכל כיוון, קדימה או אחורה.

המנוע מחובר בין המתגים, והמתגים מסודרים בשני זוגות, זוג אחד מלמעלה וזוג אחד מלמטה. ניתן להפעיל ולכבות את המתגים בשילובים שונים כדי לשלוט בכיוון ומהירות המנוע.

כדי לסובב את המנוע לכיוון אחד, נוכל להפעיל את המתגים 1 ו-4, ולכבות את המתגים 2 ו-3. זה מחבר את המסוף החיובי של ספק הכוח לצד אחד של המנוע, ואת המסוף השלילי לצד השני. זה מייצר זרימת זרם בכיוון אחד דרך המנוע, מה שגורם לו להסתובב בכיוון אחד.

כדי לסובב את המנוע לכיוון השני, נוכל להפעיל את המתגים 2 ו-3, ולכבות את המתגים 1 ו-4. זה הופך את הקוטביות של המתח המופעל על המנוע, וגורם לזרימת הזרם להתהפך ולמנוע להסתובב פנימה. הכיוון ההפוך.

על ידי שליטה בתזמון ומשך המעבר של גשר H, נוכל לשלוט גם על מהירות המנוע. לדוגמה, אם נפעיל מתג אחד לתקופה ארוכה יותר מהמתג השני, המתח המופעל על המנוע יהיה גדול יותר, והמנוע יסתובב מהר יותר.

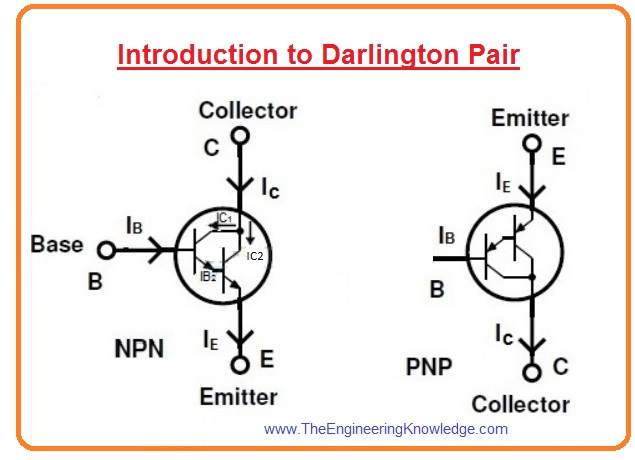
גשרי H נמצאים בשימוש נפוץ ברובוטיקה, כלי רכב חשמליים ויישומים אחרים שבהם נדרשת שליטה מדויקת של מהירות המנוע והכיוון.

### חוק לורנץ

כוח לורנץ, הכוח המופעל על חלקיק טעון q שנע במהירות v דרך שדה חשמלי E ושדה מגנטי B. כל הכוח האלקטרומגנטי F על החלקיק הטעון נקרא כוח לורנץ.

משוואה: ***F*** = q***E*** + q***v*** × ***B***

זה אומר שהכוח האלקטרומגנטי על מטען q הוא שילוב של כוח בכיוון השדה החשמלי E פרופורציונלי לגודל השדה וכמות המטען, וכוח בזווית ישרה לשדה המגנטי B וה- מהירות v של המטען, פרופורציונלית לגודל השדה, המטען והמהירות.

טרנזיסטור דארלינגטון - דארלינגטון הוא מעגל המורכב מטרנזיסטורים דו-קוטביים, הפלט של טרנזיסטור אחד מחובר לבסיס הטרנזיסטור השני כך שהזרם המוגבר על ידי הטרנזיסטור הראשון מוגבר על ידי הטרנזיסטור השני, הם מחוברים. לתצורה זו יש רווח זרם גבוה יותר מכל טרנזיסטור הוא פועל כמו טרנזיסטור ולעתים קרובות הוא ארוז כיחידה אחת. ניתן להסביר את הפעולה הבסיסית של טרנזיסטור דרלינגטון באופן הבא:

כאשר זרם קטן מופעל על בסיס הטרנזיסטור הראשון (Q1), הוא גורם לזרם גדול יותר לזרום מהקולט של Q1 לפולט של Q2. זרם גדול יותר זה גורם לזרם גדול עוד יותר לזרום מהקולט של Q2 לנגד העומס או לאלמנט מעגל אחר.

ניתן לחשב את רווח הזרם הכולל של טרנזיסטור דרלינגטון על ידי הכפלת הרווח הנוכחי של Q1 בהגבר הנוכחי של Q2. לדוגמה, אם ל-Q1 יש רווח זרם של 100 ול-Q2 יש גבר זרם של 200, רווח הזרם הכולל של הטרנזיסטור של דרלינגטון יהיה 100 x 200 = 20,000.

רווח הזרם הגבוה של טרנזיסטור דרלינגטון הופך אותו לשימושי ביישומים שבהם צריך לשלוט או להגביר כמות גדולה של זרם. עם זאת, ישנם כמה חסרונות לשימוש בטרנזיסטורים של דרלינגטון, כגון מתח רוויה גבוה יותר ומהירות מיתוג איטית יותר עקב ההגברה הדו-שלבית.

## DS18B20-One Wire Digital Temperature Sensorחיישן טמפרטורה – 18b20 לא נכנס בדגם הסופי

מד החום הדיגיטלי 18b20 מספק קריאות טמפרטורה של 9 עד 12 סיביות המציינות את הטמפרטורה של המכשיר. המידע נשלח מהחיישן באמצעות ממשק 1-Wire, כך שצריך לחבר רק חוט אחד (ואדמה) ממעבד מרכזי ל-18b20. ניתן להפיק כוח לקריאה, כתיבה וביצוע המרות טמפרטורה מקו הנתונים עצמו ללא צורך במקור מתח חיצוני.

לחיישן יש 3 רגלים:

* רגל 1 – מקבלת מתח בין 3 – 5.5 וולט.
* רגל 2 – העברת מידע בתקשורת Wire – 1.
* רגל 3 – אדמה (GND).

התמונה מציגה דיאגרמת בלוקים של החיישן, ה – ROM של 64 סיביות מאחסן את הקוד הסידורי הייחודי של המכשיר. הזיכרון מכיל את 2 אוגרי הטמפרטורה שמאחסנים בתוכם את הפלט הדיגיטלי מחיישן הטמפרטורה.



לוח ה – scratchpad מספק גישה לאוגרי אזעקה עליונים ותחתונים של בית אחד (TH ו -TL) ואוגר התצורה של בית אחד. אוגר התצורה מאפשר למשתמש להגדיר את הרזולוציה של המרת טמפרטורה לדיגיטל בין 9 ל – 12 סיביות.

האוגרים (TH,TL,והקונפיג) אינם נדיפים בעלי שבב EEPROM, כך שהם שומרים נתונים גם כשהחיישן מושבת.

### פרוטוקול – Wire -1

הבסיס של טכנולוגיית 1-Wire הוא תקשורת טורית המשתמש בקו נתונים בודד בתוספת הפניה לקרקע לתקשורת.

מאסטר 1-Wire יוזם ושולט בתקשורת עם התקני ה-SLAVE 1-Wire אחד או יותר באפיק (BUS) 1-Wire.

לכל התקן עבד 1-Wire יש מספר זיהוי ייחודי, בלתי ניתן לשינוי, מתוכנת במפעל, 64 סיביות (ID), המשמש ככתובת התקן באפיק (BUS) 1-Wire.

הקוד המשפחתי של 8 סיביות, קבוצת משנה של מזהה 64 סיביות, מזהה את סוג ההתקן ואת הפונקציונליות. בדרך כלל, התקני עבד 1-Wire פועלים על פני ארבעת טווחי המתח הבאים:

* 1.71V (מינימום) עד 1.89V (מקסימום)
* 1.71V (מינימום) עד 3.63V (מקסימום)
* 2.97V (מינימום) עד 3.63V (מקסימום)
* 2.8V (מינימום) עד 5.25V (מקסימום)

לרוב התקני 1-Wire אין פינים עבור ספק כוח. הם לוקחים את האנרגיה שלהם מהאפיק (BUS) 1-Wire (ספק כוח טפילי).

### אפיק - BUS

דרכו אפשר גם להוציא נתונים וגם להכניס נתונים. יש לו שימוש בתכנות עם זיכרון וכו'.

### UART- Universal Asynchronous Receiver-Transmitter

UART הוא אחד מפרוטוקולי התקשורת בין התקן למכשיר הנפוצים ביותר.

כאשר מוגדר כהלכה, UART יכול לעבוד עם סוגים רבים ושונים של פרוטוקולים טוריים הכוללים שידור וקבלה של נתונים טוריים.

בתקשורת טורית, נתונים מועברים סיבית אחר סיבית באמצעות קו בודד או חוט. בתקשורת דו-כיוונית, אנו משתמשים בשני חוטים להעברת נתונים טורית מוצלחת.

בהתאם ליישום ולדרישות המערכת, תקשורת טורית זקוקה לפחות מעגלים וחוטים, מה שמפחית את עלות ההטמעה.

## JOY-iT NodeMCU ESP32 Development Board | Elektorבקר – ESP32

ESP32 הוא שם השבב שפותח על ידי Espressif Systems. שבב זה משתמש ב- Wi-Fi ובדגמים מסוימים dual mode Bluetooth למכשירים משובצים.

לשבב ה-ESP32 יש מעבד Tensilica Xtensa LX6 בווריאציות כפולות ליבה ויחידות, עם קצב שעון של מעל 240 מגה-הרץ. ישנם כעת מספר דגמי שבבים שונים זמינים, כולל:

* ESP32-D0WDQ6 (ו-ESP32D0WD)
* ESP32-D2WD
* ESP32-S0WD

והמערכת בחבילה (SiP) – ESP32-PICO-D4

דגמים זמינים עם Wi-Fi ו-Bluetooth משולבת, או רק Wi-Fi.

ה-ESP32 מתוכנן לרוב עבור מכשירים ניידים, טכנולוגיה לבישה ויישומי IoT - כגון Nabto. יתרה מכך, עם הצגת מערכת ההפעלה Mongoose ESP32 IoT Starter Kit, ה-ESP32 צבר מוניטין של השבב או המודול האולטימטיבי עבור חובבים ומפתחי IoT.

אמנם המוניטין הזה אינו מוצדק, אבל המכשיר הזול יכול לשמש גם במספר מערכות ייצור שונות, והיכולות והמשאבים שלו גדלו בצורה מרשימה בארבע השנים האחרונות.

### תפקידי הדקים

^ - בטוח לשימוש

@ - ההתנהגות שלהם יכולה להיות בלתי צפויה, בעיקר במהלך האתחול. אל תשתמש בהם אלא אם כן אתה בהחלט צריך.

X – לא מומלץ

|  |  |  |  |
| --- | --- | --- | --- |
| שם הרגל | מספר הרגל – GPIO | בטוח לשימוש | תפקיד |
| EN | - | - | Enable- pin מופעל ב - HIGH |
| VIN | - | - | ניתן להשתמש בפין VIN כדי לספק בחשמל ישירות את ה-ESP32 |
| 3.3V | - | - | פין 3.3V הוא הפלט של וסת מתח על הלוח.  יכול לשמש לאספקת חשמל לרכיבים חיצוניים |
| D0 | 0 | @ | חייב להיות HIGH במהלך האתחול ו- LOW לתכנות |
| TX0 | 1 | X | משמש להבהוב וניפוי באגים |
| D2 | 2 | @ | חייב להיות LOW במהלך האתחול וגם מחובר לנורת ה-LED המובנית |
| RX0 | 3 | X | משמש להבהוב וניפוי באגים |
| D4 | 4 | ^ | פין לשימוש חופשי |
| D5 | 5 | @ | חייב להיות HIGH במהלך האתחול |
| D6 | 6 | X | מחובר לזיכרון פלאש |
| D7 | 7 | X | מחובר לזיכרון פלאש |
| D8 | 8 | X | מחובר לזיכרון פלאש |
| D9 | 9 | X | מחובר לזיכרון פלאש |
| D10 | 10 | X | מחובר לזיכרון פלאש |
| D11 | 11 | X | מחובר לזיכרון פלאש |
| D12 | 12 | @ | חייב להיות LOW במהלך האתחול |
| D13 | 13 | ^ | פין לשימוש חופשי |
| D14 | 14 | ^ | פין לשימוש חופשי |
| D15 | 15 | @ | חייב להיות HIGH במהלך האתחול, מונע startup log אם נמשך LOW |
| RX2 | 16 | ^ | פין לשימוש חופשי |
| TX2 | 17 | ^ | פין לשימוש חופשי |
| D18 | 18 | ^ | פין לשימוש חופשי |
| D19 | 19 | ^ | פין לשימוש חופשי |
| D20 | 20 | ^ | פין לשימוש חופשי |
| D21 | 21 | ^ | פין לשימוש חופשי |
| D22 | 22 | ^ | פין לשימוש חופשי |
| D23 | 23 | ^ | פין לשימוש חופשי |
| D24 | 24 | ^ | פין לשימוש חופשי |
| D25 | 25 | ^ | פין לשימוש חופשי |
| D26 | 26 | ^ | פין לשימוש חופשי |
| D27 | 27 | ^ | פין לשימוש חופשי |
| D28 | 28 | ^ | פין לשימוש חופשי |
| D29 | 29 | ^ | פין לשימוש חופשי |
| D30 | 30 | ^ | פין לשימוש חופשי |
| D31 | 31 | ^ | פין לשימוש חופשי |
| D32 | 32 | ^ | פין לשימוש חופשי |
| D33 | 33 | ^ | פין לשימוש חופשי |
| D34 | 34 | @ | קלט בלבד, לא ניתן להגדיר כפלט |
| D35 | 35 | @ | קלט בלבד, לא ניתן להגדיר כפלט |
| Sensor\_VP | 36 | @ | קלט בלבד, לא ניתן להגדיר כפלט |
| GND | 38 |  | אדמה |
| Sensor\_VN | 39 | @ | קלט בלבד, לא ניתן להגדיר כפלט |

### עקרון פעולה

עקרון הפעולה של ה – ESP32 הוא שאפשר להשתמש בו למערכות בעלות מספר רב של פינים, רכיב בר תכנות (חומר ביד היוצר), ניתן להתחבר לאינטרנט. לבקר יש מעבד חזק בעל תקשורת טורית.

ה – ESP32 צורך כמות נמוכה של חשמל ויש לו זיכרון פנימי, הוא פועל בצורה מבוקרת ומוגדרת בשביל לספק מינימום אנרגיה שצריך.

### מבנה פנימי

### רדיו

לשבב ESP32 SoC יש חיבור WiFi, תואם ל-802.11 b/g/n בפס 2.4GHz, ומגיע למהירויות של עד 150 Mbit/s. הוא כולל גם תקשורת Bluetooth התואמת ל-Bluetooth v4.2 ו-Bluetooth Low Energy (BLE).

בלוק הרדיו קשור קשר הדוק למודול התקשורת האלחוטית. למעשה, זהו זה שבעצם משדר ומקבל את המידע.

כלומר, הוא לוקח את הנתונים הדיגיטליים ממודולי ה-WiFi וה-Bluetooth וממיר אותם לאותות אלקטרומגנטיים שעוברים באוויר כדי לתקשר עם הטלפון הנייד או הנתב שלך.

הוא גם מבצע את הפעולה ההפוכה: מתרגם את הגלים האלקטרומגנטיים שנוצרים על ידי מכשירים אחרים לנתונים דיגיטליים שמודול ה-WiFi וה-Bluetooth מסוגלים לפרש.

### ליבה

ל-ESP32 יש מעבדי Tensilica Xtensa 32-bit LX6 בעלי עוצמה נמוכה כפולה.

כפי שניתן לראות מתמונת בלוק הליבה, יש לו מעבד משותף בעל הספק נמוך במיוחד המשמש לביצוע המרות אנלוגיות-דיגיטליות ופעולות אחרות בזמן שהמכשיר פועל במצב שינה עמוקה עם צריכת חשמל נמוכה. בדרך זו מושגת צריכה נמוכה מאוד על ידי ה-SoC.

חשוב לציין שמעבדים אלה מציעים יתרונות אופייניים גדולים של מעבד אותות דיגיטלי:

* תדר הפעלה: 240 מגה-הרץ (מבצע הוראות פי 15 מהר יותר מלוח Arduino UNO)
* הוא מאפשר לבצע פעולות עם מספרים ממשיים (מספרים עם פסיקים) ביעילות רבה.
* מאפשר לך להכפיל מספרים גדולים באופן מיידי.

### זיכרון

ברוב המיקרו-בקרים המבוססים על Arduino, ישנם שלושה סוגים של זיכרונות:

זיכרון תוכנית: לאחסון הסקיצה.

זיכרון SRAM: לאחסון המשתנים המשמשים בקוד.

זיכרון EEPROM: לאחסון משתנים שאינם מאבדים את ערכם גם כשהמכשיר כבוי.

ב-ESP32 זה לא קורה, למעשה ישנם סוגים נוספים של זיכרונות שבדרך כלל מסווגים לתוך פנימי וחיצוני.

הזיכרונות הפנימיים הם אלה שכבר כלולים ב-SoC, והחיצוניים הם אלה שניתן להוסיף כדי להרחיב את הקיבולת של המערכת.

לוחות פיתוח רבים מבוססי ESP32 מוסיפים זיכרון חיצוני למערכת בעלת ביצועים טובים יותר.

ESP32 זיכרונות פנימיים ותפקידיהם:

זיכרון ROM (448 KiB): זיכרון זה הוא לכתיבה בלבד, כלומר, לא ניתן לתכנת אותו מחדש. זה המקום שבו מאוחסנים הקודים המטפלים בערימת ה-Bluetooth, בקרת השכבה הפיזית של ה-Wi-Fi, כמה שגרות למטרות כלליות ומטען האתחול להפעלת הקוד מזיכרון חיצוני.

זיכרון SRAM פנימי (520 KiB): זיכרון זה משמש את המעבד לאחסון נתונים והוראות. היתרון שלו הוא שהרבה יותר קל לגישה למעבד מאשר ל-SRAM החיצוני.

RTC SRAM (16 KiB): זיכרון זה משמש את המעבד המשותף כאשר ההתקן פועל במצב שינה עמוקה.

Efuse (1 קילוביט): 256 סיביות מזיכרון זה משמשות את המערכת עצמה ו-768 הסיביות הנותרות שמורות ליישומים אחרים.

Flash Embedded ( Embedded flash ): הזיכרון הזה הוא המקום שבו קוד היישום שלנו מאוחסן. כמות הזיכרון משתנה בהתאם לשבב המשמש:

0 MB (שבבים ESP32-D0WDQ6, ESP32-D0WD, ESP32-S0WD)

2 MB (שבב ESP32-D2WD)

4 MB (שבב ESP32-PICO-D4)

עבור ESP32s שאין להם זיכרון מוטבע או פשוט כאשר הזיכרון אינו מספיק עבור היישום שלך, אפשר להוסיף יותר זיכרון חיצוני:

ניתן להוסיף עד 16 MB של זיכרון פלאש חיצוני. כך תוכלו לפתח אפליקציות מורכבות יותר.

הוא תומך גם בעד 8MB של זיכרון SRAM חיצוני.

### מאיצי חומרת הצפנה

אחד הגורמים החשובים ביותר בכל מערכת הוא אבטחה. לכן ל-ESP32 יש מאיצי אלגוריתמים המכוונים להצפנה:

* AES (FIPS PUB 197)
* SHA (FIPS PUB 180-4)
* RSA
* ETC

מאיצים אלו מאפשרים להגביר את מהירות הפעולה ולהפחית את מורכבות התוכנה המאפשרים הצפנה ופענוח דינמי. באופן זה המערכת מוגנת מפני התקפות פריצה אפשריות המבקשות להשיג את הקוד המאוחסן.

### ערכים חשמליים

* ה – ESP32 יכול לקבל בין 2.2 - 5 וולט
* זרם של 500mA
* תדר מתנד – 40MHz

## פרוטוקולים- ESP32

### WI-FI

ESP32 מיישם TCP/IP,full 802.11 b/g/n/e/i WLAN MAC ו- Wi-Fi Direct. המשמעות היא שה-ESP 32 יכול לדבר עם רוב נתבי ה-WiFi בחוץ כאשר משתמשים במצב תחנה. כמו כן, הוא מסוגל ליצור נקודת גישה עם 802.11 b/g/n/e/i מלא.

ESP32 תומך גם ב-Wi-Fi Direct. Wifi-Direct היא אפשרות טובה לחיבור peer to peer (p2p) ללא צורך בנקודת גישה. ה-Wifi-Direct קל יותר להגדרה ומהירויות העברת הנתונים טובות בהרבה מ-Bluetooth. זה יכול לשמש פוטנציאל להגדרת פרויקטים מבוססי ESP32 מטלפון/טאבלט התומך ב-WiFi ישיר. למימוש ESP-IDF WiFi יש את התכונות הבאות בפיתוח:

* תשתית מצב BSS Station / מצב P2P / תמיכה במצב softAP
* P2P Discovery, P2P Group בעלים, P2P Group Client ו-P2P Power Management
* מנהל התקן WPA/WPA2-Enterprise ו-WPS
* תכונות אבטחה נוספות של 802.11i כגון אימות מוקדם ו-TSN
* אלגוריתם החזרה של קצב הסתגלות מגדיר את קצב השידור וההספק האופטימלי על סמך יחס רעשי אותות בפועל (SNR) ומידע על אובדן מנות מידע (packet loss)
* שידור חוזר ותגובה אוטומטיים ב-MAC כדי למנוע השלכת מנות בסביבת מארח איטית

### Bluetooth Low Energy(BLE)

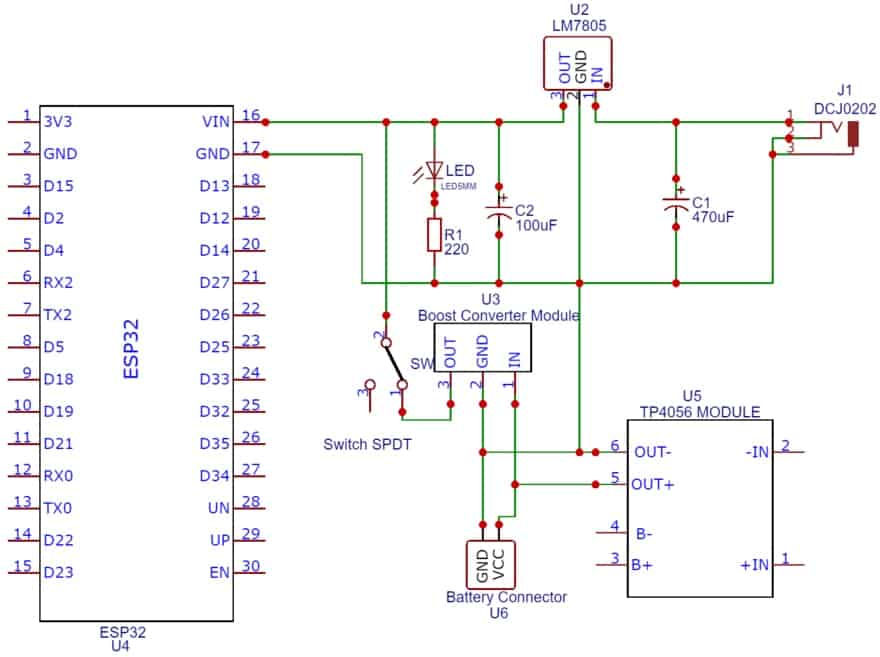
ESP32 לא רק תומך ב-BLE Bluetooth 4.2 העדכני ביותר, הוא תומך גם ב-Bluetooth קלאסי. זה בעצם אומר שהוא יכול לדבר עם טלפונים/טבלאות Bluetooth ישנים וחדשים. זו יכולה להיות אחת התכונות הטובות ביותר, במיוחד אם אתה מעצב מכשיר שצריך לעבוד עם טלפונים/טאבלטים קיימים כמו חדשים בשוק. רדיו ו-Baseband ESP32 Bluetooth תומכים בתכונות הבאות:

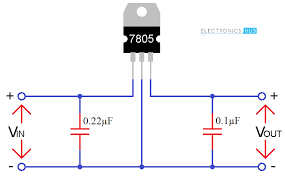
* Class-1, Class-2 ו- Class-3 משדרות הספקי מוצא ומעל טווח בקרה דינמי של 30dB
* אפנון π/4 DQPSK ו-8 DPSK
* ביצועים גבוהים ברגישות למקלט NZIF עם טווח דינמי של מעל 98 dB
* פעולה Class-1 ללא PA חיצוני
* SRAM פנימי מאפשר העברת נתונים במהירות מלאה, קול ונתונים מעורבים ופעולת פיקונט מלאה
* לוגיקה לתיקון שגיאות קדימה, בקרת שגיאות כותרת, מתאם קוד גישה, CRC, מודולציה, יצירת זרם סיביות בהצפנה, הלבנה ועיצוב דופק שידור
* ACL, SCO, eSCO ו-AFH
* A-law, µ-law ו-CODEC אודיו דיגיטלי CVSD בממשק PCM
* SBC אודיו CODEC
* ניהול צריכת חשמל עבור יישומים בהספק נמוך
* SMP עם AE של 128 סיביות

### ESP – NOW

ESP-NOW הוא מעין פרוטוקול תקשורת Wi-Fi ללא חיבור המוגדר על ידי Espressif. ב-ESP-NOW, נתוני האפליקציה מובלעים במסגרת פעולה ספציפית לספק ולאחר מכן מועברים ממכשיר Wi-Fi אחד לאחר ללא חיבור. CTR עם פרוטוקול CBC-MAC (CCMP) משמש להגנה על מסגרת הפעולה לצורך אבטחה. ESP-NOW נמצא בשימוש נרחב בתאורה חכמה, שליטה מרחוק, חיישן וכו'.

## מייצב מתח של הESP (Voltage Regulator)





מייצבי מתח הם מווסתים שמקבלים מתחים גבוהים מהרצוי ומוציאים את המתח הרצוי. סוג המתח שמייצב המתח מוציא נקבע לפי שתי הספרות האחרונות של הרכיב.

אנו משתמשים במייצב מתח 7805 לכן הוא מוציא 5v. בפרויקט הרכיב כבר בנוי בתוך ה-cap.

לפי המעגל בתמונה יש רגל כניסה של המתח המסופק (Vin),רגל של יציאת המתח הרצוי (Vout) המתח שיוצא ורגל המחוברת לאדמה.

בנוסף יש שני קבלים מנתקים ("decoupling capacitor"), אחד בין יציאת המתח ואחד בין מתח הכניסה בגדלים של 0.22pF ו – 0.1pF. ללא שני הקבלים הללו מתח היציאה עלול להיות לא מדויק ולהוציא את ערך גדול או קטן מידי.

קבל הניתוק השמאלי באמצע בין המייצב לבין המתח המסופק ומפריד ביניהם. קבל הניתוק הימני משמש כמוצא קטן לעומס.

. מייצב המתח LM7805 יכול לקחת את מתח הכניסה מ-5V ל-35V. אבל מומלץ להשתמש במתח הכניסה עד 15V בלבד. עם עלייה במתח, יש יותר פיזור חום שדורש גוף קירור גדול יותר. הפלט מווסת המתח מחובר לפין Vin של ESP32 ו-GND מחובר ל-GND. לפיכך אתה יכול להפעיל את המודול באמצעות מתאם 9V/12V DC או באמצעות סוללה של 9V.

## ESP32 - CAM

מצלמה זו מחוברת אל הבקר ESP32 היא יכולה לשדר בזמן אמת תמונות וסרטונים. אפשר לשים את הבקר בכל מקום ובכל פרויקט שיש בו צורך למצלמה כל יסופק אינטרנט.

|  |  |
| --- | --- |
| שם רגל | תפקיד |
| GPIO1 | משמש כ - משדר TX |
| GPIO3 | משמש כ- מקלט RX |
| GND | אדמה |
| GND | אדמה |
| Vcc | יציאת מתח |
| 5v | מספקת 5 וולט |
| 3.3v | מספקת 3.3v |
| GPIO2 | פין מידע 0 |
| GPIO4 | פין מידע 1 |
| GPIO12 | פין מידע 2 |
| GPIO13 | פין מידע 3 |
| GPIO14 | כניסת שעון (CLK) |
| GPIO15 | רגל בקרה |

### תפקידי הדקים



### עקרון פעולה

מצלמת אינטרנט המשדרת לאחור מידע (תמונות וסרטונים) בתקשורת WIFI בזמן אמת, הבקר שולח מידע דרך שרת LAN שפותח באמצעות WIFI וניתן לגשת למידע הנשלח על ידי כתובת IP שהבקר מספק לאחר הרצת קוד פשוט.

### ערכים חשמליים

* מתח מקסימלי – 5 וולט
* זרם מקסימלי – 310mA
* תדר – 80-240MHz

## FPGA Dev Kits for Altera Cyclone® IV - Terasic Technologies | MouserAltera

Altera Corporation הייתה יצרנית של התקני לוגיקה ניתנים לתכנות (PLD), קווי המוצרים העיקריים מבית Altera היו סדרת הדגל Stratix, סדרת Arria בטווח הביניים, ומערכת סדרת Cyclone בעלות נמוכה יותר על מערכי שערים הניתנים לתכנות בשדה שבב (FPGAs).

התקן לוגי מורכב מתכנות מסדרת MAX ו-FPGAs לא נדיפים תוכנת תכנון Quartus ופתרונות מתח של Enpirion PowerSoC DC-DC.

בתוך האלטרה יש רכיב שנקרא FPGA.

## FPGA-Field-Programmable Gate Array

FPGA או מערכי שער הניתנים לתכנות שדה הם התקני מוליכים למחצה המבוססים על מטריצה ​​של בלוקים לוגיים ניתנים להגדרה (CLBs) המחוברים באמצעות חיבורים ניתנים לתכנות.

ניתן לתכנת מחדש רכיבי FPGA לדרישות היישום או הפונקציונליות הרצויות לאחר הייצור. תכונה זו מבדילה בין רכיבי FPGA לבין מעגלים משולבים ליישום ספציפי (ASIC), המיוצרים בהתאמה אישית עבור משימות עיצוב ספציפיות.

### DE10 Lite_User_Manual Lite User Manualתפקיד הדקים



אלה הם extension pins בשביל חיבור רכיבים לבחירה.

### עקרון פעולה

הבקר פועל לפי תכנות שפת תיאור חומרה (VHDL) או על ידי הזנת תרשים לוגי, הוא מורכב מיחידות לוגיות הניתנות לתכנות ומרשת של אמצעים לחיבור וניתוק בין היחידות.\

הרכיב מבוסס על זיכרון מסוג flex בבקר האלטרה כלומר רכיב זה אינו זוכר את התוכן לאחר כיבוי/ניתוק של המתח ולכן יש לצרוב את התוכנית עליו לאחר הכיבוי.

### מבנה פנימי

ברכיב בנוי כדי לספק גמישות מרבית למשתמש, כל החיבורים נעשים באמצעות התקן MAX 10 FPGA. לפיכך, המשתמש יכול להגדיר את ה-FPGA ליישם כל עיצוב מערכת.

### ערכים חשמליים

הרכיב מקבל 5 או 0 וולט.

## תקן תקשורת RS232

UART – Universal Asynchronous Receiver Transmitter

משדר מקלט אסינכרוני אוניברסלי.

תקן תקשורת זה מתקשר תקשורת טורית ולא מקבילית, מכיוון שהעברת מידע מקבילי צורכת חוט לכל סיבית וזהו חיסרון ותקשורת טורית מספקת לנו מהירות מהירה מספיק. משתמשים בדרך כלל במידע מקבילי רק בין רכיבים ומערכות שצמודים פיזית כמו המחשב.

בהעברת מידע טורי יש חוט אחד לכל הסיביות ואפשר לחתוך את החוט ולשים משדר בכל צד ולהעביר מידע בין מערכות שפיזית רחוקות אחת מהשנייה, כלומר תקשורת אלחוטית.

**שימוש תקן RS232** – להעביר מידע ממקור ליעד בצורה טורית בכל מרחק (בכבל).

**סינכרוני** – כניסת clk אחת שמחוברת במקביל לשתי מערכות ושתיהן מסונכרנות ביחד כלומר התזמון הוא ביחד.

**אסינכרוני** – מערכות שאין להן את אותה כניסת clk פיזית, יכול להיות שבמערכות אסינכרוניות התזמון לא היה מתוזמן טוב.

בגלל שתקן זה עובד באסינכרוניות צריך פרוטוקול, שהוא מוודא שהמערכת ששולחת יהיו כללים בדיוק כמו המערכת שקולטת ומטפל בכל הבעיות האפשריות במערכות אסינכרוניות רחוקות.

* שליחת מידע שאומר שהמערכת ששולחת מתחילה לשלוח מידע.
* שליחת מידע בסדר מסוים שהמערכת שקולטת יודעת מהו הסדר.
* שליחת מידע שאומר שהיא מסיימת לשלוח את המידע והמערכת שקולטת צריכה "לארוז" את המידע ולהתחיל לעשות איתו דברים.
* שליחת מידע נוסף בשביל בדיקת המידע שנקלט אם הוא תקין או לא.

**קצב עבודה** – מספר סיביות המידע שנשלחות בשנייה.

ישנם הרבה קצבים, וקצב העבודה הנמוך ביותר הוא 1200bps וזה נקרא גם1200 baud rate.

**Baud Rate** - מספר סיביות המידע שנשלחות בשנייה.

עוד קצבי עבודה הם: 2400bps, 4800bps,**9600bps**,19200bps, 38400bpsוכו'. הם מוכרים בכל העולם, ואנו נשתמש בקצב עבודה 9600bps שהוא הנפוץ ביותר בעולם, והוא אינו גבוה מידי ואינו נמוך מידי לכן הוא הנפוץ ביותר.

**הסבר פעולת תקן RS232**

Data = 8bit (0 - 255) in decimal

(0 - FF) in hex

מידע

**Start bit**

**LSB**

**0**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**MSB**

**PB**

**Stop bit**

אופציונלי

שליחת המידע תמיד תתחיל מה – LSB אל ה – MSB.

**סיביות בקרה**

בתקן RS232 יש 8 סיביות מידע שתמיד עוטפות אותן סיביות הבקרה האלו:

Start bit = ‘0’

Stop bit = ‘1’

**זמן מחזור של כל סיבית בתקן RS232 הוא 16 זמני מחזור שעון**

סיביות אלו אינן חלק מהמידע, הן רק משמשות כאותות להתחלה וסיום. אם נשלח מידע שאינו עטוף בסיביות אלו (כשבהתחלה יש '0' ובסוף יש '1') אז ישנה תקלת תקשורת, והמערכת יודעת להתעלם מהמידע שנשלח עם הסיביות בקרה האלו.

כל עוד לא התחלנו לשלוח מידע יהיה מצב שנקרא **idle** שזה אומר שיש '1' לוגי על החוט של התקשורת כי הוא stop bit. מערכת שקולטת יודעת לזהות stop bit, כלומר היא יכולה לזהות סיבית של '1' לוגי שיש לה זמן מחזור ארוך יותר מ – 16 זמני מחזור שעון ולדעת שה – '0' לוגי הבא הוא start bit ואז יש מידע. או שהיא יכולה לספור שהיא קלטה 8 סיביות ואז הסיבית הבא חייבת להיות stop bit = ‘1’.

**סיבית זוגיות – Parity bit –** משלימה את מספר האחדות שבמידע לזוגי.

ישנה בעיית תקשורת שהיא דגימת המידע בזמן לא נכון, כלומר המערכת הקולטת הייתה מוכנה לקלוט מידע אבל היא לא הצליחה לקלוט אותו כמו שצריך. לכן נרצה לתפוס את הסיבית באמצע זמן המחזור של הסיבית כלומר חצי זמן מחזור של סיבית. במקרה שלנו זה 8 זמני מחזור שעון (ראה סרטוט).

לא ננסה לדגום את המידע כאן

ננסה לדגום אותה כשהיא כבר יציבה כמו כאן.

**דוגמה לתהליך זה בתקן RS232:**

אמצע סיבית ומתקיימת דגימה



**Stop bit**

**Start bit**

בירידת Rx סופרת 16 עליות וב – 8 דוגמת (באמצע). ומכאן סופרת 16 כל פעם בשביל להיות באמצע של כל סיבית (בעצם נקודה זו נהפכה לנקודת ההתחלה).

16x

16x

16x

8

16x

אמצע סיבית ומתקיימת דגימה

0

15

כל סיבית נשלחת למשך 16 זמני מחזור שעון כדי להידגם אחרי 8 (לבדוק באמצע).

# LM75 – Digital Temperature Sensor

LM75 הוא חיישן טמפרטורה דיגיטלי הכולל ממיר דלתא-סיגמה אנלוגי לדיגיטלי, וגלאי טמפרטורת יתר דיגיטלי. המארח יכול לקבוע את ה-LM75 דרך ממשק ה -I2C שלו לקריאת טמפרטורה בכל עת. פלט Open - drain של טמפרטורת יתר (OS) שוקע את הזרם כאשר חריגה ממגבלת הטמפרטורה הניתנת לתכנות. הכתובת של LM75 מוגדרת עם שלושה פינים כדי לאפשר מספר מכשירים לעבוד על אותו BUS.

Open drain - כאשר התקן הפלט כבוי, הפין נותר צף (פתוח, או hi-z). דוגמה נפוצה היא טרנזיסטור n-channel אשר מושך את האות לאדמה כאשר הטרנזיסטור דולק או משאיר אותו פתוח כאשר הטרנזיסטור כבוי.

שימושים:

● ניהול מערכת תרמית

● הגנה תרמית

● בדיקת ציוד

● מחשבים ואלקטרוניקה משרדית

תכונות:

● חבילות ISO (SOP) ו-μMAX (μSOP).

● ממשק אפיק I2C

● הניקוז פתוח פלט של טמפרטורת יתר שפועל ככניסת פסיקה או השוואת/תרמוסטט

● יכולת קריאה חוזרת של אוגר

● ברירות מחדל להפעלה מאפשרות פעולה עצמאית בתור תרמוסטט

● מתח אספקה ​​של 3.0V עד 5.5V

● זרם אספקת הפעלה נמוך μA 250, 1mA (מקסימום)

● μA 4 מצב כיבוי ממזער את הספק הצריכה

● ניתן לחבר עד שמונה LM75 לאוטובוס בודד

### תפקידי הדקים

|  |  |  |
| --- | --- | --- |
| **הדק** | **שם** | **תפקיד** |
| 1 | SDA | קו קלט/פלט טורי-נתונים. Open drain. חבר את SDA לנגד משיכה |
| 2 | SCL | קלט שעון טורי. Open drain. חבר את SCL לנגד משיכה. |
| 3 | OS | פלט כיבוי בטמפרטורת יתר. Open drain. חבר את מערכת ההפעלה לנגד משיכה. |
| 4 | GND | אדמה |
| 5 | A2 | קלט כתובת ממשק דו-חוטי. חבר את A2 ל-GND או +VS כדי להגדיר את כתובת הBUS - הרצויה של I2C.  אל תשאיר לא מחובר |
| 6 | A1 | קלט כתובת ממשק דו-חוטי. חבר את A1 ל-GND או +VS כדי להגדיר את כתובת ה BUS - הרצויה של I2C.  אל תשאיר לא מחובר |
| 7 | A0 | קלט כתובת ממשק דו-חוטי. חבר את A0 ל-GND או +VS כדי להגדיר את כתובת ה BUS - הרצויה של I2C.  אל תשאיר לא מחובר |
| 8 | +Vs | כניסת מתח אספקה ​​חיובית. עוקף ל-GND עם קבל עוקף 0.1µF. |

### עקרון פעולה

חיישן הטמפרטורה LM75 מודד טמפרטורה וממיר את הנתונים לצורה דיגיטלית באמצעות סוג חיישן טמפרטורה bandgap וממיר דלתא-סיגמה (אנלוגי לדיגיטלי) של 9 סיביות.

### קמבנה פנימי

אוגר המצביע של LM75 בוחר בין ארבעה נתונים רגיסטרים. בהפעלה, המצביע הוא מוגדר לקרוא את אוגר הטמפרטורה. אוגר המצביע (pointer) תופס את המיקום האחרון שאליו זה נקבע. כל הרשמים נקראים וכותבים, מלבד מאוגר טמפרטורה, הנקרא בלבד. כתיבה לאוגר ה- Configuration על ידי כתיבת כתובת בייט (a data pointer byte) ובייט מידע. אם 2 הבייטים מידע נכתבים, בייט הנתונים השני מחליף את הראשון.

אוגרי TOS ו-THYST דורשים בייט כתובת אחד, בייט מצביע (pointer) אחד ו-2 בייט מידע. אם נכתב רק בייט מידע אחד, זה נשמר בסיביות D15-D8 של האוגר המתאים. אם יותר מ-2 בייטים של מידע נכתבים, רק 2 הבייטים הראשונים מזוהים בעוד שהבייטים הנותרים מתעלמים.

קריאה מה-LM75 באחת משתי דרכים. אם המיקום נעול ב-Pointer register מוגדר מהקריאה הקודמת, הקריאה החדשה מורכבת מבייט כתובת, ואחריו על ידי שליפת המספר התואם של בתים של נתונים. אם יש להגדיר את אוגר המצביע לכתובת חדשה, בצע פעולת קריאה על ידי כתיבת בייט כתובת, בייט מצביע (pointer תחזור על ההתחלה ובייט כתובת נוסף. קריאה בשוגג של 8 סיביות מאוגר של 16 סיביות, עם ה-D7 bit נמוך, יכול לגרום להתקן לעצור במצב שבו קו SDA נשמר נמוך. בדרך כלל, זה ימנע כל תקשורת אוטובוס נוספת עד שהמאסטר ישלח תשעה מחזורי שעון נוספים או SDA עולה גבוה. בזה זמן, מצב עצירה מאפס את המכשיר. אם הנוסף מחזורי שעון אינם נוצרים על ידי המאסטר, ה- BUS של ה – LM75 מתאפס ונפתח לאחר תקופת הזמן הקצוב של ה - BUS חלף.

### גרפים ואופניים

### מושגים נלווים

נגד משיכה - במעגלים לוגיים אלקטרוניים, נגד משיכה (Pullup) או נגד משיכה Pulldown) הוא נגד המשמש להבטחת מצב ידוע לאות. הוא משמש בדרך כלל בשילוב עם רכיבים כגון מתגים וטרנזיסטורים, אשר קוטעים פיזית את החיבור של רכיבים עוקבים לאדמה או ל-VCC. סגירת המתג יוצרת חיבור ישיר לאדמה או ל-VCC, אך כאשר המתג פתוח, שאר המעגל יישאר צף (כלומר, יהיה לו מתח בלתי מוגדר).

קבל מעקף (bypass capacitor) - קבלים מעקפים משמשים לשמירה על עכבת אספקת חשמל נמוכה בנקודת העומס. התנגדות טפילית והשראות בקווי אספקה ​​אומרות שעכבת אספקת החשמל יכולה להיות גבוהה למדי. ככל שהתדר עולה, הטפיל האינדוקטיבי הופך לבעייתי במיוחד.

Bandgap temp sensor - חיישן הטמפרטורה לסיליקון הוא צורה נפוצה ביותר של חיישן טמפרטורה (מדחום) המשמש בציוד אלקטרוני. היתרון העיקרי שלו הוא שניתן לכלול אותו במעגל משולב סיליקון בעלות נמוכה מאוד. העיקרון של החיישן הוא שהמתח קדימה של דיודת סיליקון, שעשוי להיות צומת הבסיס-פולט של טרנזיסטור צומת דו-קוטבי (BJT), תלוי בטמפרטורה.

### פרוטוקולים

### I2C BUS

I2C, ראשי תיבות של Inter-Integrated Circuit, הוא פרוטוקול תקשורת בשימוש נרחב המיועד לתקשורת לטווח קצר בין מעגלים משולבים (ICs) או רכיבים אלקטרוניים בלוח מעגלים. הוא פותח על ידי Philips Semiconductor (כיום NXP Semiconductors) בתחילת שנות ה-80 ומאז הפך לסטנדרט דה פקטו לתקשורת בין ICs ביישומים שונים.

I2C הוא פרוטוקול סינכרוני, מאסטר, עבד, כלומר מספר ICs יכולים לתקשר זה עם זה באמצעות BUS משותף. הפרוטוקול משתמש בשני חוטים: קו נתונים טורי (SDA) וקו שעון טורי (SCL). קווים אלו מחוברים לכל ההתקנים בBUS I2C.

להלן סקירה בסיסית של אופן הפעולה של תקשורת I2C:

* תנאי התחלה: התקן הראשי יוזם תקשורת על ידי משיכת קו SDA נמוך בזמן שקו SCL גבוה. זה מסמן את תחילתה של עסקת תקשורת.
* כתובת: התקן הראשי שולח כתובת של 7 סיביות של התקן העבד איתו הוא רוצה לתקשר, ואחריו סיביות קריאה/כתיבה. הכתובת נשלחת ביט אחר ביט בקו SDA בעוד שקו SCL שולט בתזמון.
* אישור עבד: לאחר קבלת הכתובת, התקן העבד הממוען מושך את קו ה-SDA נמוך במהלך סיבית האישור. אם התקן העבד קיים באוטובוס ומוכן לתקשר, הוא יאשר את הכתובת.
* העברת נתונים: ברגע שהתקן העבד מאשר, המכשיר הראשי והתקן העבד יכולים להחליף נתונים. הנתונים מועברים בבייטים של 8 סיביות כשהשולח (מאסטר או עבד) מעביר את הנתונים לקו SDA בזמן שהמכשיר השני קורא את הנתונים. קו SCL שולט בתזמון העברת הנתונים.
* תנאי עצירה: כדי לסיים עסקה, ההתקן הראשי יוצר תנאי עצירה. זה משחרר את קו SDA בעוד קו SCL גבוה. זה מסמן את סיום התקשורת.

חשוב לציין ש-I2C תומך במספר התקני מאסטר באותו BUS. אם מספר מאסטרים מנסים לגשת לאוטובוס בו זמנית, נמנעת התנגשות על ידי מנגנון בוררות המבוסס על הייחודיות של הכתובות המועברות.

I2C משמש בדרך כלל ביישומים שונים, כגון חיישני ממשק, EEPROMs, שעוני זמן אמת, מודולי תצוגה והתקנים היקפיים אחרים עם מיקרו-בקרים או ICs אחרים. הפשטות שלו, ספירת הפינים הנמוכה והתמיכה בתקשורת מרובת התקנים הופכים אותו לבחירה פופולרית לתקשורת בין IC במערכות אלקטרוניות רבות.

## Firebase – Google

Firebase הוא קבוצה של שירותי אירוח (hosting services) עבור כל סוג של יישום (אנדרואיד, iOS, Javascript, Node.js, Java, Unity, PHP, C++ ...). הוא מציע NoSQL ואירוח בזמן אמת של מסדי נתונים, תוכן, אימות חברתי (גוגל, פייסבוק, טוויטר ו-Github), והודעות, או שירותים, כגון שרת תקשורת בזמן אמת.

NoSQL - מסד נתונים של NoSQL (הכוונה במקור ל"לא-SQL" או "לא-רלציונלי") מספק מנגנון לאחסון ושליפה של נתונים המעוצבים באמצעים אחרים מלבד היחסים הטבלאיים המשמשים במסדי נתונים יחסיים.

## ULN2803A ULN2803 Darlington Transistor Array NPN DIP-18 — Juried Engineeringדוחף זרם – uln2803

שמונת הטרנזיסטורים המחוברים NPN Darlington במשפחת מערכים זו מתאימים באופן אידיאלי לממשק בין מעגלים דיגיטליים ברמה לוגית נמוכה (כגון TTL, CMOS או PMOS/NMOS) לבין דרישות הזרם/מתח הגבוהות יותר של מנורות, ממסרים, פטישי מדפסת או עומסים דומים אחרים. מגוון יישומי מחשב, תעשייה וצרכנים. כל המכשירים כוללים יציאות אספן פתוחות ודיודות מהדקים מגלגלים חופשיים לדיכוי ארעיות. ה-ULN2803 תוכנן להיות תואם למשפחות TTL סטנדרטיות בעוד שה-ULN2804 מותאם ל-6 עד 15 וולט CMOS או PMOS ברמה גבוהה.

### הדקים

* פין 1 הוא הבסיס של הטרנזיסטור הראשון.
* פין 2 הוא הבסיס של הטרנזיסטור השני.
* פין 3 הוא הבסיס של הטרנזיסטור השלישי.
* פין 4 הוא הבסיס של הטרנזיסטור הרביעי.
* פין 5 הוא הבסיס של הטרנזיסטור החמישי.
* פין 6 הוא הבסיס של הטרנזיסטור השישי.
* פין 7 הוא הבסיס של הטרנזיסטור השביעי.
* פין 8 הוא הבסיס של הטרנזיסטור השמיני.
* פין 9 הוא הפולט כמו גם הארקה של כל הטרנזיסטורים.
* פין 10 ידוע בתור פין משותף כלומר צומת של קתודה משותפת המשמשת לדיודות ה-Fly Back.
* פין 11 הוא האספן של הטרנזיסטור השמיני.
* פין 12 הוא האספן של הטרנזיסטור השביעי.
* פין 13 הוא האספן של הטרנזיסטור השישי.
* פין 14 הוא האספן של הטרנזיסטור החמישי.
* פין 15 הוא האספן של הטרנזיסטור הרביעי.
* פין 16 הוא האספן של הטרנזיסטור השלישי.
* פין 17 הוא האספן של הטרנזיסטור השני.
* פין 18 הוא האספן של הטרנזיסטור הראשון.

דיודת fly – back - דיודת פליבק היא כל דיודה המחוברת על פני משרן המשמשת לביטול פליבק, שהוא זינוק המתח הפתאומי הנראה על פני עומס אינדוקטיבי כאשר זרם האספקה ​​שלה מצטמצם לפתע או מופרע. הוא משמש במעגלים שבהם עומסים אינדוקטיביים נשלטים על ידי מתגים, ובמיתוג ספקי כוח וממירים.

המפרטים, כמו גם התכונות של ULN2803, הם:

* המתח המרבי המותר בין פולט לאספן עבור כל טרנזיסטור דרלינגטון הוא 50 וולט.
* הזרם המרבי המותר לזרום דרך האספן של כל זוג דרלינגטון הוא 500 mA.
* בין הפולט לבסיס של צמד דארלינגטון, המתח המרבי המותר הוא 30 וולט.
* בכל זוג דרלינגטון, קיימת דיודת זבוב והזרם המרבי המותר לעבור דרכו הוא 500 mA.
* זמן העלייה הוא 130 ns.
* בדרך כלל, זמן הנפילה הוא 20 מיקרו-שניות.
* טמפרטורת הפעולה של ULN2803 נעה בטווח של - -65 – 150 מעלות צלזיוס
* על מנת להפעיל את השבב הזה, אין צורך בכוח נוסף.

### מבנה פנימי

השבב ULN2803 מורכב מ-8 זוגות דרלינגטון הפועלים כשמונה מתגים בודדים. מתוך שמונה זוג טרנזיסטורים של דרלינגטון, אם נסתכל על זוג בודד, נוכל לקבל משהו כזה:

ניתן להציג את צמד דרלינגטון הפשוט להלן:



מהנתונים שלעיל, ניתן לראות בבירור שכל זוג של דרלינגטון יכול לשמש כטרנזיסטור כוח יחיד. לפיכך, אנו יכולים לומר שב-ULN2803, שמונה טרנזיסטורי כוח מוטמעים. להבנה טובה יותר, בואו נסתכל לעבר דיאגרמת מעגלים אחרת כדי לייצג דברים בצורה מקיפה יותר.

טרנזיסטור דארלינגטון - דארלינגטון הוא מעגל המורכב מטרנזיסטורים דו-קוטביים, הפלט של טרנזיסטור אחד מחובר לבסיס הטרנזיסטור השני כך שהזרם המוגבר על ידי הטרנזיסטור הראשון מוגבר על ידי הטרנזיסטור השני, הם מחוברים. לתצורה זו יש רווח זרם גבוה יותר מכל טרנזיסטור הוא פועל כמו טרנזיסטור ולעתים קרובות הוא ארוז כיחידה אחת.

## נורת להט (הוחלף בלד של ESP32CAM)

נורת ליבון, מנורת ליבון או גלובוס אור ליבון היא אור חשמלי עם חוט מחומם עד שהוא זוהר. החוט מוקף בנורת זכוכית עם ואקום או גז אינרטי כדי להגן על החוט מפני חמצון. זרם מסופק לחוט הלהט על ידי מסופים או חוטים המוטבעים בזכוכית. שקע נורה מספק תמיכה מכנית וחיבורים חשמליים.

לנורה יש שני פינים של אדמה ו – 5 וולט.

## Laser 5v

ה-LASER 5V 5mW 650nm Red Dot Laser Diode Pointer הוא מכשיר קומפקטי הפולט קרן לייזר אדומה. בואו נפרט איך זה עובד:

מקור מתח: המודול דורש אספקת חשמל של 5 וולט (5V) כדי לפעול. ניתן להשיג זאת על ידי חיבורו למקור מתח מתאים כגון סוללה או מתאם מתח.

דיודה לייזר: בלב המודול נמצאת דיודת לייזר. דיודת לייזר היא מכשיר מוליכים למחצה הפולט אור קוהרנטי כאשר מופעל עליה זרם חשמלי. במקרה זה, דיודת הלייזר פולטת קרן לייזר אדומה באורך גל של 650 ננומטר (ננומטר). אורך הגל של 650 ננומטר מתאים לחלק האדום של ספקטרום האור הנראה.

מעגל דרייבר: דיודת הלייזר דורשת ויסות זרם מדויק כדי לפעול בצורה מיטבית ולמנוע נזק. המודל כולל בדרך כלל מעגל דרייבר השולט בזרם המסופק לדיודת הלייזר. זה מבטיח שהדיודה תקבל זרם יציב בגבולות ההפעלה הבטוחה.

אופטיקה: המודול משלב אופטיקה לעיצוב ומיקוד קרן הלייזר. זה עשוי לכלול עדשת קולימציה או שילוב של עדשות כדי להבטיח שקרן הלייזר תופיע כקרן מקבילה צרה. האלומה המאוחדת מסייעת בהשגת נקודה אדומה גלויה ומוגדרת היטב.

מנגנון הפעלה: המודל מצויד בדרך כלל במתג או כפתור להפעלה ולכיבוי של קרן הלייזר. כאשר המתג מופעל, או הלחצן נלחץ, הזרם החשמלי זורם דרך מעגל הנהג ומפעיל את דיודת הלייזר.

אמצעי בטיחות: מצביעי לייזר כפופים לתקנות הבטיחות, ויצרנים משלבים בדרך כלל תכונות בטיחות כדי למנוע חשיפה מקרית לקרן הלייזר. כמה אמצעי בטיחות נפוצים כוללים מנעול מפתח או מערכת נעילת בטיחות הדורשת פעולה מכוונת להפעלת הלייזר.

יישומים: מצביע LASER 5V 5mW 650nm Red Dot Laser Diode Pointer נמצא בשימוש נפוץ ביישומים שונים. זה יכול לשמש כעזר חזותי במצגות, אסטרונומיה ומשימות יישור. בנוסף, הוא משמש לעתים בפרויקטים תחביבים, רובוטיקה וכמכשיר ראייה לכלי נשק.



## מנוע Servo – MG90S

מנוע סרוו הוא סוג של מנוע חשמלי המשמש לשליטה מדויקת של מיקום זוויתי, מהירות ותאוצה. הוא מורכב ממנוע, מפחית, מעגל בקרה ומערכת משוב.

המנוע הוא בדרך כלל מנוע חשמלי DC, אך הוא יכול להיות גם מנוע צעד או מנוע סינכרוני AC. המפחית הוא רכבת הילוכים המפחיתה את מהירות המנוע ומגבירה את המומנט שלו, ומאפשרת למנוע הסרוו לייצר תנועות מדויקות וחזקות.

מעגל הבקרה מקבל אותות כניסה מבקר, כגון מחשב או מיקרו-בקר, ושולח אותות חשמליים למנוע כדי לשלוט בסיבובו. מערכת המשוב, המורכבת בדרך כלל מקודד או פוטנציומטר, מודדת את המיקום האמיתי של המנוע ושולחת מידע זה בחזרה למעגל הבקרה כדי להבטיח שהמנוע זז כמתוכנן.

מנועי סרוו נמצאים בשימוש נפוץ ברובוטיקה, ייצור ויישומים אחרים שבהם נדרשת שליטה מדויקת בתנועה. הם משמשים לעתים קרובות גם בתחביבים RC (מבוקרי רדיו), כמו מזל"טים, מטוסים ומכוניות.

כמה יתרונות של מנועי סרוו כוללים דיוק גבוה, זמן תגובה מהיר ויכולת לפעול במגוון רחב של טמפרטורות וסביבות. הם גם נוטים להיות יעילים יותר באנרגיה מאשר סוגים אחרים של מנועים, מכיוון שהם צורכים חשמל רק כשהם נעים באופן פעיל. סרוו יכול להסתובב בערך 180 מעלות (90 לכל כיוון), ועובד בדיוק כמו סוגים סטנדרטיים אך קטנים יותר.

אתה יכול להשתמש בכל קוד סרוו, חומרה או ספריה כדי לשלוט בסרוו. הסרוו מגיע עם 3 צופרים (זרועות) וחומרה.

יישומים:

* משמש כמפעילים ברובוטים רבים כמו רובוט דו-פעמי, Hexapod, זרוע רובוטית וכו'.
* משמש נפוץ למערכת היגוי בצעצועים RC
* רובוטים שבהם נדרשת בקרת מיקום ללא משוב
* פחות משקל ומכאן בשימוש ברובוטים מרובי DOF כמו רובוטים דמויי אדם

תכונות:

* משקל: 13.4 גרם
* מומנט עצירה: 1.8 kgf·cm (4.8V )
* סיבוב: 0°-180°
* מהירות פעולה: 0.1 שניות/60 מעלות
* מתח הפעלה: 4.8 וולט
* רוחב פס מת: 5 µs
* זרם (סרק) 10mA (אופייני)
* זרם (אופייני בזמן תנועה) 120 -250mA

נוכחי (דוכן)

kgf·cm –סנטימטר של כוח קילוגרם (kgf·cm) הוא יחידה מטרית של מומנט (נקראת גם "מומנט" או "רגע הכוח"). סנטימטר אחד של כוח קילוגרם שווה למומנט הנובע מכוח של כוח של קילוגרם אחד המופעל בניצב על זרוע מומנט באורך סנטימטר אחד.

### הדקים

המנוע מגיע עם שלושה הדקים המחוברים למנוע בצבעים קבועים:

* כתום – PWM
* אדום – 5v
* חום – GND

### עקרון פעולה

שליטה של מיקום זוויתי ודיוק מירבי ומהיר לפי קבלת ה – PWM הרצוי.

### מבנה פנימי



מנועי סרוו מורכבים לרוב ממנועי DC המשתמשים במנגנוני משוב כדי לנוע בדיוק רב ממיקום אחד לאחר כפי שניתן לראות בתמונה. מנועי סרוו מכילים סדרה של גלגלי שיניים שמאיצים או מאטים ומחליקים את התנועה של מנוע ה-DC. לבסוף, מנועי סרוו משתמשים במעגל כדי לשלוט ולשלוח מידע משוב לבקר נתון.

### גרפים ואופיינים

מהתמונה אנו יכולים להבין כי לאות ה-PWM המופק צריך להיות תדר של 50Hz כלומר, תקופת ה-PWM צריכה להיות 20ms. מתוכם זמן ההפעלה יכול לנוע בין 1ms ל-2ms. אז כאשר זמן ההפעלה הוא 1ms המנוע יהיה ב-0° וכאשר 1.5ms המנוע יהיה 90°, באופן דומה כאשר הוא 2ms הוא יהיה 180°. לכן, על ידי שינוי זמן ההפעלה מ-1ms ל-2ms ניתן לשלוט במנוע מ-0° ל-180°.

# 

# פרק 2- מעגלים

# מעגלים חשמליים

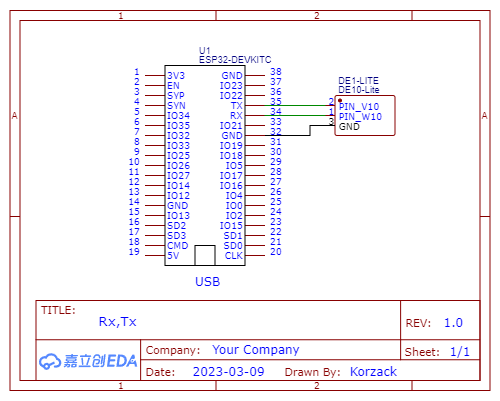
DC Motor to L293D

חיבור כניסות מתח ל - 5 וולט (1,16), חיבורי המנועים אדמה בESP (4,5,12,13) וכניסת מתח (3,6,11,14) בדוחף זרם. חיבור כניסת מידע לde10 (2,7,10,15). חיבור enable2 ו – vs ל – 5 וולט ו – 12 וולט.

המנועים מחוברים לדחוף זרם ברגלים 3,6,11,14 ומקבלים מתח של 9 וולט ישיר מהסוללה (הלחמנו חוט ישיר במקום שהמתח יועבר במייצב מתח ויקטן), לדוחף זרם יש 4 כניסות מידע שמחוברות לאלטרה ואומרות לדוחף זרם מתי להזרים זרם למנועים לפי הסיביות מידע שמקבלים מהמשתמש ששולח אותן דרך האפליקציה בפרוטוקול WIFI לפיירבייס ומשם לבקר ESP32 דרך פרוטוקול WIFI ואז לFPGA דרך TX,RX ששם הסיביות מידע מתחלקות לתוכניות המתאימות בתכנון ההיררכי שבתוך ה- FPGA.

למנועים יש תוכנית של PWM\_CTRL שנמצאת בתוך ה - FPGA שהיא אומרת לדוחף זרם לאן להזרים זרם (לאיזה חוט מידע של המנועים להזרים) לפי תדר של 10KHz ב – D.C של 80% . הזרם הוא 800mA. המטרה שלו לדחוף זרם מתאים למנועים בתדר של 10KHz עם D.C של 80% ברגע שהוא מקבל את הסיביות המתאימות מהאלטרה.

## ALTERA to ESP – TX to RX



חיבור Tx & Rx בין DE10 ל – ESP32.

המטרה של הבקר ESP32 הינו לשמש כמטווח בין האלטרה לאפליקציה. מהאפליקציה נשלחות 8 סיביות מידע דרך פרוטוקול WIFI לפיירבייס ומהפיירבייס לבקר גם דרך פרוטוקול WIFI ומהבקר לאלטרה דרך TX,RX. לאחר שהאלטרה קולטת את הסיביות, הן מתחלקות לתוכניות המתאימות שלהן בתכנון ההיררכי של האלטרה לפי I/O ב – PIN PLANNER.

## ESP32 – CAM to Firebase to Java App

המצלמה מתחברת לWIFI וכך מתקשרת עם השרת רשת, כאשר רוצים להדליק את הפנס שלה נשלחת פקודה דרך שרת הרשת להדלקתה אל המצלמה דרך WIFI. המצלמה משדרת את התמונה דרך פרוטוקול WIFI והIP שלה. באפליקציה התמונה מעובדת ואם רוצים אפשר להפעיל עיבוד/זיהוי תמונה.

## LM75 TEMP SENSOR to ALTERA And ESP32

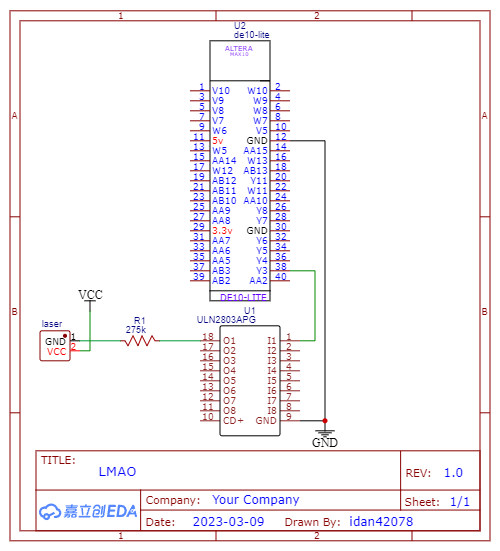
חיבור Vcc ואדמה ל – ESP32 כניסות מידע לאלטרה (1,2)

המטרה של רכיב זה היא להעביר את הטמפרטורה שהוא קולט בשטח דרך האלטרה בעזרת פרוטוקול I2C אל ה – ESP32 בעזרת RX,TX יש לו שני כניסות מידע שמחוברות לאלטרה אליה הוא שולח את הטמפרטורה דרך תוכנית VHDL בתשע סיביות ובאלטרה גם מומר ל8 סיביות. וממנו אל הפיירבייס דרך פרוטוקול WIFI ואל האפליקציה דרך WIFI שם יש המרה מפרנהייט לצלסיוס. יש לו כניסת אדמה וכניסת מתח בין 2.7 -5.5 וולט. כניסת ה- O.S אינה משומשת ומראה מתי הרכיב הגיע למקסימום טמפרטורה.

## Servo motor to Altera

מנוע הסרבו מחובר לאלטרה ומקבל ממנה 5v ואת הPWM שלו, כאשר רוצים להזיזו מהאפליקציה בעזרת פרוטוקול WIFI נשלחות 8 סיביות כאשר 3 מהן משומשות לסרבו לשמונה מצבים מ-"000" עד "111" כלומר כל הזזה הינה 22.5 מעלות. שלושת הסיביות נכנסות לתוכנית servo\_control בתכנון ההיררכי של האלטרה ומשם לPWM של הסרבו בתדר של 50MHz. בתדר PWM יש זמן בגבוה בין 1ms – 2ms לדוגמה כאשר זמן התדר הוא 1.5ms הסרבו יהיה ב - 90°, שמשפיע על זווית הסרבו שמשפיע על כיוון הלייזר.

## Uln to laser to altera



Ω270

הלייזר מחובר לדוחף זרם ULN שמספק לו את כניסת המידע אם להדליק אותו או לא (0 או 1 לוגי) ובנוסף מחובר לנגד של 270 אום לוסת את הזרם שהלייזר לא ישרף. כאשר האלטרה מקבלת סיבית מידע מהאפליקציה בעזרת פרוטוקול WIFI והESP היא עוברת בתכנון ההיררכי של האלטרה ומותאמת למוצא המתאים ב – PIN PLANNER.

## מייצב מתח של הESP

## 

ווסת המתח LM7805 IC יכול לקחת את מתח הכניסה מ-7V ל-35V. אבל מומלץ להשתמש במתח הכניסה עד 15V בלבד. עם עלייה במתח, יש יותר פיזור חום שדורש גוף קירור גדול יותר. הפלט מווסת המתח מחובר לפין Vin של ESP32 ו-GND מחובר ל-GND. לפיכך אתה יכול להפעיל את המודול באמצעות מתאם 9V/12V DC או באמצעות סוללה של 9V.

## מעגל שלם של כל הפרויקט

תהליך הפעלת הפרויקט מתחיל באפליקציה בכל פעולה, דרך האפליקציה המשתמש מקבל נתונים מהשטח ושולח פקודות אל השטח.

בשביל להניע את המנועים אם המשתמש לוחץ על אחד המצבים (קדימה, אחורה, ימין שמאלה) המידע עובר בפרוטוקול WIFI אל שרת הרשת ומשם אל הבקר ESP32, הבקר מקבל את הנתונים שמשמעותם מספר בטווח 0-255. הבקר שולח אותם אל האלטרה בתקן תקשורת RS232, האלטרה מחלקת את 8 הסיביות שהתקבלו לחלקים של מידע שמחולקים להזזת המנועים, קביעת מהירות ודברים אחרים.

לאחר פירוק המידע היא שולחת את הסיביות לתוכניות המתאימות לכן בתכנון ההיררכי המורכב בתוך ה – FPGA, ומשם לצרכנים שאותם החלטנו להפעיל במקרה הזה המנועים.

אם החלטנו על נסיעה קדימה התוכנית המתאימה באלטרה תקבל "0101", ודוחף הזרם- L293D מקבל PWM שתפקידו לקבוע את מהירות המנועים על ידי D.C משתנה בתדר קבוע של 10KHz. ה – L293D מכיל שני גשרי H שתפקידם לספק הספק גבוה למנועים שהבקר אינו יכול לספק, וגם אם כן זה אינו מקובל שהבקר שתפקידו לחשוב יעשה זאת, ללא דוחף הזרם לא היינו יכולים להניע את המנועים. הזרם המתאים עובר דרך גשרי ה – H לכל מנוע יש גשר H ועובר דרך המנועים בכיוון המתאים לכיוון הנסיעה שבחרנו.

במידה ונרצה להפעיל את הלייזר, המידע ששלחנו מהאפליקציה בגודל 8 סיביות שמתפרק של חבילות של סיביות עובר לתוכניות המתאימות בתכנון ההיררכי של ה – FPGA וממנו אל ה – uln2803 שתפקידו למשוך זרם ללייזר. האלטרה תספק ללייזר 5v בזרם מאוד נמוך הטרנזיסטור דארלינגטון שבתוכו יכנס לרוויה וזרם יגיע מה – Vcc, יעבור את האנודה אל הקתודה של הלד דרך נגד בגודל Ω275K שתפקידו לווסת את הזרם בצורה נכונה כך שהלד לא יישרף, ויזרום אל האדמה של המעגל. אם האלטרה תיתן 0 וולט הטרנזיסטור יהיה בקטעון ויהיה נתק.

מנוע הסרבו מקבל תדר של 50MHz שבתוכו זמן בגבוה בין 1ms – 2ms לדוגמה כאשר זמן התדר הוא 1.5ms הסרבו יהיה ב - 90°, שמשפיע על זווית הסרבו שמשפיע על כיוון הלייזר.

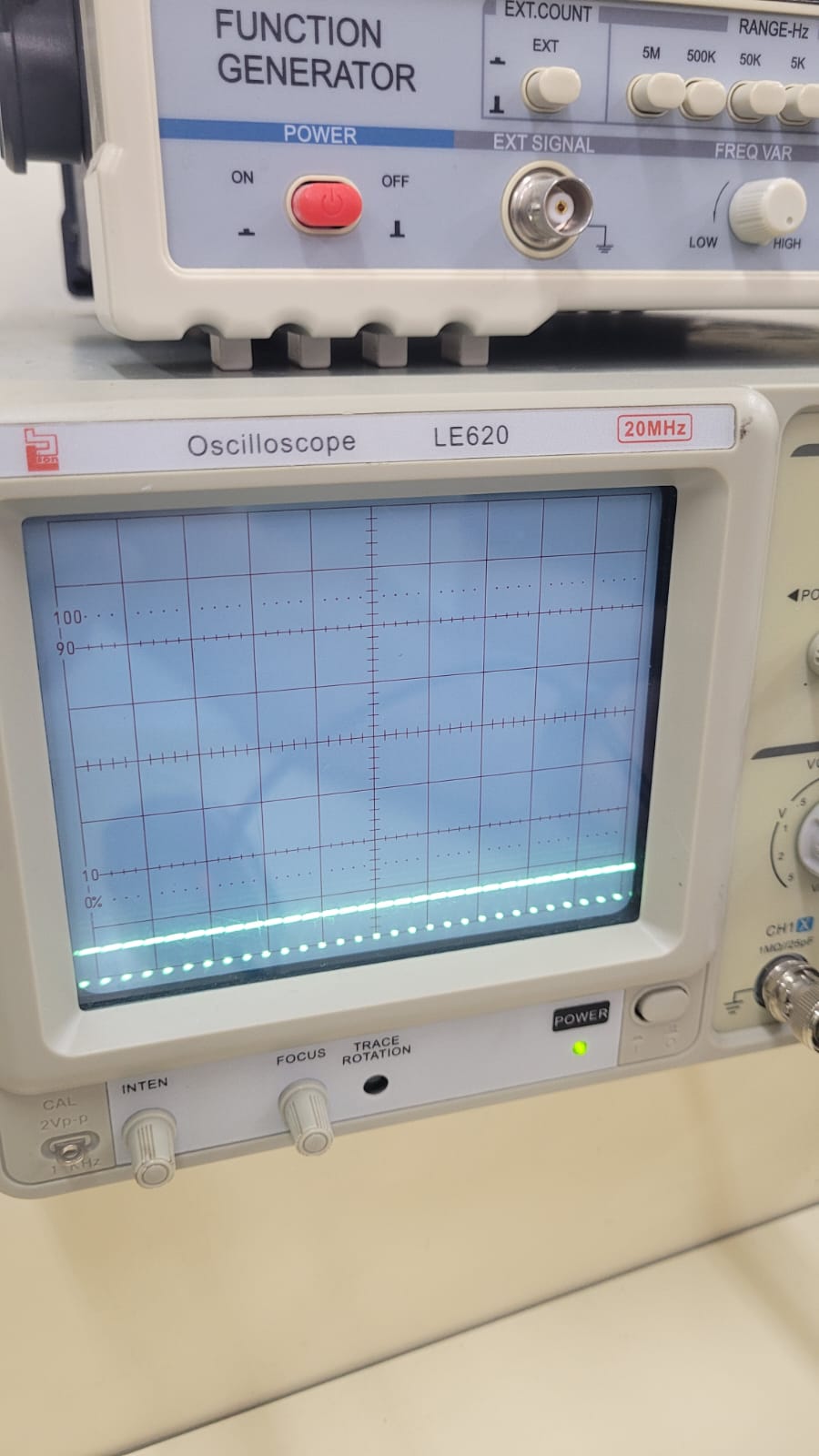
המצלמה נותנת לנו אפשרות לראות כל מקום שיש בוא אינטרנט את מה שיש בשטח שהיא נמצאת בו. אנו מבצעים גם פעולת עיבוד וזיהוי תמונה שתפקידה לאסוף נתונים בשטח ולשלוח אותם למשתמש ובנוסף נותנת אפשרות לרכב להגות אוטומטית כלפי חפץ שנבחר.

75LM הינו חיישן טמפרטורה שמשדר את טמפרטורת הסביבה של הרכב בעזרת פרוטוקול I2C אל ה – FPGA, התוכנית המתאימה ב – FPGA קולטת את הטמפרטורה שנוגע בפרוטוקול ב – 9 סיביות אחר ב - FPGA המידע מומר ל – 8 סיביות. לאחר מכן הטמפרטורה נשלחת בפרנהייט דרך המשדר של האלטרה אל הקולט של ה – ESP32 ומשם שולח את הטמפרטורה אל שרת הרשת בעזרת פרוטוקול WIFI, ומשם אל האפליקציה שם הטמפרטורה מומרת לצלסיוס. תהליך זה בכיוון ההפוך מכיוון שהמידע מגיע מהשטח אל השרת רשת ומשם לאפליקציה ולא ההפך.

# פרק 3 - מדידות

# מדידות

## מדידות מנועי DC

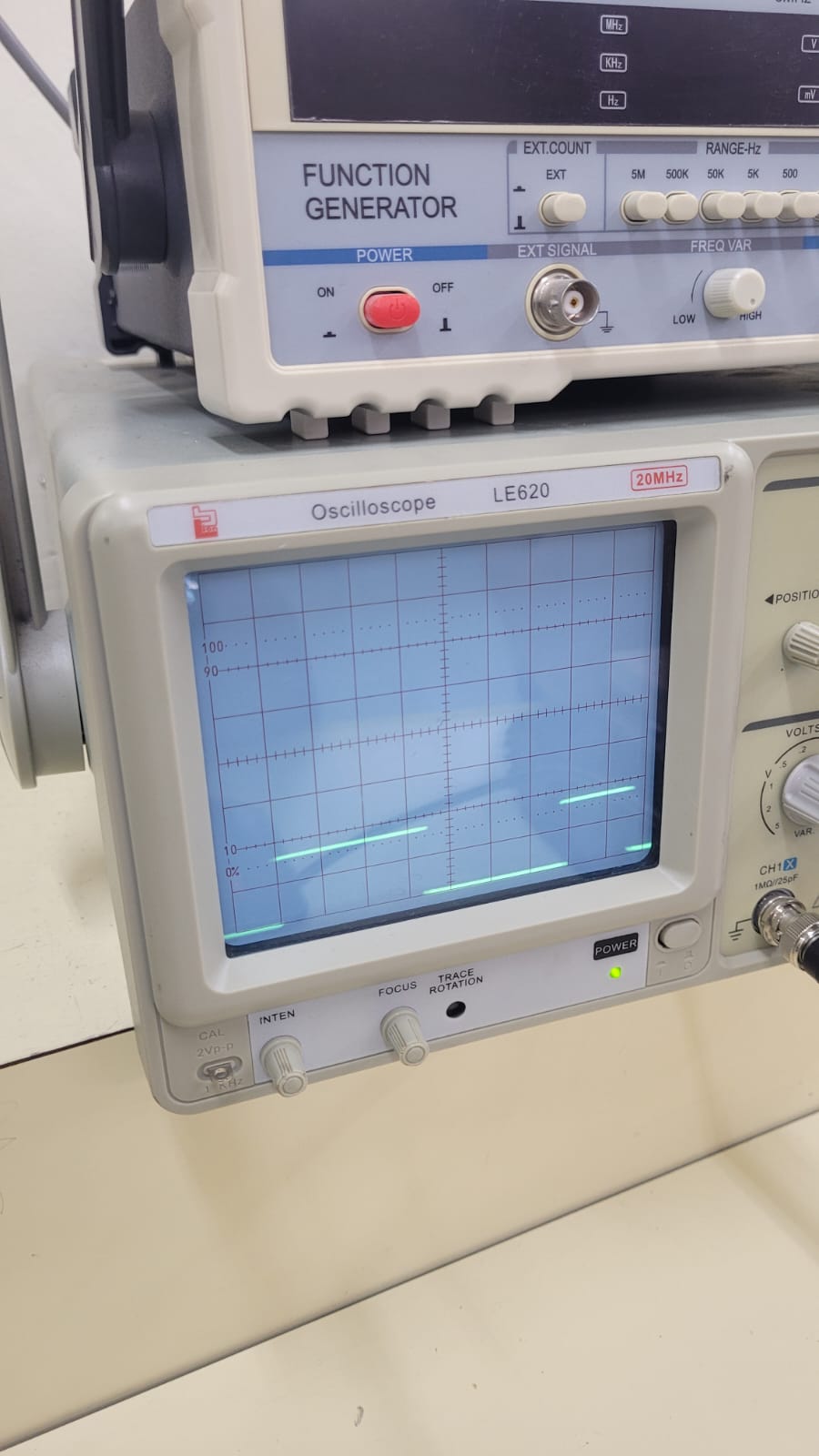


בתמונה זו ניתן לראות תדר PWM ב D.C -של 80% בגודל של 10KHz כל משבצת היא 5v.

## מדידות מנוע Servo

בתמונה זו ניתן לראות תדר PWM בגודל של 50MHz הזמן שבו הוא פועל הוא ms1.5.

## מדידות Rx



בתמונה זו ניתן לראות תדר של Hz153.600 המכיל את כל המידע הנחוץ למערכת שמגיע אל האלטרה.

# פרק 4 - VHDL

# VHDL

חיסרון משמעותי במעבדים של אינטל הוא שהמעבד יכול לבצע רק פעולה אחת.

אפשר לעשות אינסוף פעולות VHDL היום בזכות רכיבים המתוכנתים ב -

המתרחשות במקביל.

עוד יתרון זה שהכל נמצא על צ'יפ אחד לעומת מעבד רגיל של אינטל שצריך עזרה מלוח האם, בנוסף לכך ספיקת הכוח של הצ'יפ גם הרבה יותר נמוכה ויעילה בהשוואה לספיקת הכוח של מעבד רגיל.

**VHDL**

**V** - stands for VHSIC which stands for – Very High Speed Integrated Circuit רכיבים בעלי תכנות מהירים מאוד –

**HDL** – stands for – Hardware Description Language - שפת תיאור חומרה

תמצית רקע יתרונות ושימושים של השפה:

רקע

תחילת שנות ה – 80

דרישה של גופי פיתוח בעולם וה – DOD

יכולות גבוהות וחדשות של תכנות חומרה

יתרונות

תכנות התנהגותי של תוכנה – הפיכת קוד ב – VHDL למערכת שערים לוגים.

אוניברסליות – אפשר להעביר דרך האינטרנט קבצים למקומות שונים בעולם ולפתוח אותם בכל שפת פיתוח.

יכולות תיקון ושדרוג מהירים – אם רוצים לשנות רכיב כלשהו או משהו בתוך הרכיב כמו מספר כניסות או יציאות במקום לבנות ולממש אותו מחדש, הולכים לקוד של VHDL ומשנים שם.

יכולת עבודת צוות בתכנון היררכי – לקחת מערכת עם תפקיד כלשהו, לפרק אותה לחלקים קטנים, כל חלק קטן במערכת נפרק לעוד חלקים קטנים ובסוף נרכיב את כל החלקים ונקבל את המוח שעושה את הפעולה השלמה.

שימוש במיתולוגיות**:**

TOP DOWN – לקחת את המערכת השלמה והתפקיד, ולפרק אותו לחלקים ולבנות אותם אחד אחד.

דוגמה ל- TOP DOWN:

מחלקים מוח של תוכנית לתוכניות קטנות ואז , תחום האחריות של כל תוכנית נורא קטן אז יוצא שכל מתכנת צריך לעשות תוכנית מאוד קטנה ופשוטה וזה יתרון מאוד גדול כי קשה לטעות בה אפשר לבדוק מהר אם היא טובה וגם אפשר שכל מי שעובד יהיה בכל מקום בעולם (Outsourcing)

BOTTOM UP – לקחת את החלקים ולחבר אותם ולהגיע למערכת שלמה.

פעם בשביל ליצור מוח כלשהו לתפקיד מסוים היו יוצרים קובץ **אחד** ארוך מאוד ועליו לתכנת את המוח, וזה חיסרון גדול מאוד מכיוון שקשה לעבוד עם קובץ אחד ארוך, קשה למצוא טעויות, ומקסימום אנשים שיכולים לעבוד עליו הוא אולי 2 והם צריכים להיות מסונכרנים לשינויים שהם עושים שזה עוד יותר מסובך. לכן מחלקים

אותו לחלקים.

סימולציית חומרה על מסך מחשב - בשביל להגיש תוכניות שבוודאות עובדות ישנן 2 דרכים:

בדיקה בדיאגרמת זמן **(Waveform)**

**Test bench**

Test bench– לכתוב תוכנית ב – VHDL שבודקת את התוכנית שבנינו ב – VHDL

DUT -Design under test - שם לתוכנית ב – VHDL (מושג בתכנות של VHDL)

UUT - Unit under test - שם לתוכנית ב – VHDL (מושג בתכנות של VHDL)

התוכנית שבנינו היא תוכנית לסינטזה כלומר לצריבה על מכשיר, המטרה של Test bench היא להתקיל את תוכנית הסינטזה ובמקום לראות תוצאות ב – Waveform נקבל התראות. לדוגמה - "תגיד לי אם B השתנה כש – A השתנה זמן כזה או כזה אחרי ns20" (ns = nano seconds). כלומר כותב ב – Test bench מה שאני לא רוצה שיקרה כמו Compiler.

חסרונות ב - VHDL

ישנו חיסרון אחד ב – VHDL והוא שאי אפשר לתכנת רכיבי כוח (נגד, דיודה, מגבר). היום יש טכנולוגיה שעדיין בפיתוח שנקראת ASIC שמאפשרת לתכנת קצת פעולות אנלוגיות (כוח) ולא רק דיגיטליות.

שימושים של VHDL

ניצול יכולת פעולה מקבילית של אותות להתניות – תכנות רכיבים שיכולים לבצע אינסוף פעולות במקביל.

החלפת כרטיסים גדולים ומסורבלים ברכיב אחד – במקום לזרוק את הרכיב ולהחליף הכל ברכיב אחד, נשתמש ברכיב שמתוכנת ב – VHDL.

שימוש מהיר וקל במיקור חוץ לדרכי פיתוח – אפשר להשתמש במיקור חוץ (Outsourcing) בשביל לבנות רכיבים שונים גם אם האנשים לא חלק המחברה.

Outsourcing \מיקור חוץ– לא כל האנשים חייבים להיות בחברה ובאותו מקום בשביל לתכנת רכיב, כלומר אפשר לחפש כותבי VHDL מנוסים ולפי ניסיונם לשכור אותם ולתת להם משימה לכתוב רכיב כלשהו, וכך אפשר גם לחסוך בכוח אדם.

# 

# תוכניות VHDL

## משדרTx – Transmitter –

תפקיד התוכנית - לקבל מידע מקבילי בגודל 8 סיביות ולהוציאו למידע טורי בתקן RS232.

10 סיביות x16

מחזורי מחזור של כל סיבית

### עקרון כתיבת – Tx

1. נגדיר משתנה פנימי cnt שיספור עד 160 מחזורי.
2. נגדיר משתנה פנימי clr לאיפוס בסוף שליחת מידע.

ישנם 10 סיביות בגלל שזה בתקן RS232 ויש 8 סיביות מידע ו – 2 סיביות בקרה.

סמל הרכיב:



קוד התוכנית:



library ieee;

use ieee.std\_logic\_1164.all;

entity my\_tx is

port (clk: in bit;

di : in bit\_vector(7 downto 0);

so : out bit);

end;

architecture behave of my\_tx is

signal cnt : integer range 0 to 160;

signal clr : bit;

begin

process (clk)

begin

if clr='1' then cnt<=0; so<='1';

elsif clk'event and clk='1' then

if cnt<160 then cnt<=cnt+1; else cnt<=0; end if;

if cnt<16 then so<='0'; -- start bit

elsif cnt<32 then so<=di(0); -- LSB

elsif cnt<48 then so<=di(1);

elsif cnt<64 then so<=di(2);

elsif cnt<80 then so<=di(3);

elsif cnt<96 then so<=di(4);

elsif cnt<112 then so<=di(5);

elsif cnt<128 then so<=di(6);

elsif cnt<144 then so<=di(7); -- MSB

elsif cnt<160 then so<='1'; -- stop bit

end if;

end if;

end process;

clr <='1' when cnt=160 else '0';

end behave;

**דיאגרמות:**



1 1 1 1

0 0 0 0

Start bit = ‘0’

Stop bit = ‘1’

## קולט Rx – Receiver –

תפקיד התוכנית – לקבל מידע טורי בתקן RS232 ולהוציא מידע מקבילי בתנאי שתקין.

### עקרון כתיבת התוכנית Rx

1. נגדיר process ראשון לזיהוי ירידת '1' ל – '0' בכניסת si שמשמעותו ממעבר מ – stop bit ל – start bit. ונעלה דגל (משתנה מסוג סיגנל) START ל – '1'.
2. ב – process שני נספור בכל עליית שעון עד 160 מחזורי ונדגום באמצע זמן שליחת כל סיבית את המידע

בתנאי שהדגל START = ‘1’

1. לקראת סוף קבלת המידע לתוך משתנה פנימי Temp, נבדוק תקינות ורק אם תקין נעביר את Temp למוצא Do.

זוהי תוכנית ראשונה עם שני process, מכיוון שצריך לבדוק si’ event וירידה לאפס ואי אפשר לבצע שני events ב - process אחד.

מידע תקין – מידע שעוטף אותו start bit = ‘0’ ו – stop bit = ‘0’.



סמל הרכיב:

קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity rx is

port ( clk ,si : in bit;

data : out bit\_vector(7 downto 0));

end ;

architecture behave of rx is

signal start : bit; -- flag fpr starting recieving --

signal cnt: integer range 0 to 160; -- 16X10

signal clr,start\_bit,stop\_bit : bit;

signal tmp : bit\_vector(7 downto 0);

begin

-----------------------

process ( clr, si )

begin

if clr='1' then start<='0';

elsif si'event and si='0' then start<='1'; --- checking start bit

end if;

end process;

----------------------------

process ( clr,clk)

begin

if clr='1' then cnt<=0;

elsif clk'event and clk='1' then

if start='1' then

if cnt<160 then cnt<=cnt+1; else cnt<=0; end if;

if cnt=8 then start\_bit<=si;

elsif cnt=24 then tmp(0)<=si; -- LSB

elsif cnt=40 then tmp(1)<=si;

elsif cnt=56 then tmp(2)<=si;

elsif cnt=72 then tmp(3)<=si;

elsif cnt=88 then tmp(4)<=si;

elsif cnt=104 then tmp(5)<=si;

elsif cnt=120 then tmp(6)<=si;

elsif cnt=136 then tmp(7)<=si;

elsif cnt=152 then stop\_bit<=si;

elsif cnt=155 and start\_bit='0' and stop\_bit='1' then data<=tmp;

end if;

end if;

end if;

end process;

clr<='1' when cnt=160 else '0';

end behave;



**דיאגרמות:**



ספרנו עד 160 ויש במוצא את המידע שנקלט ממשתנה Temp, כלומר כל הסיביות הועברו ונקלטו והמידע הנו תקין.

**בזום אין:**



מתווסף '1' לוגי ב- cnt = 24 במשתנה Temp

כאן אפשר לראות שדגמנו ב – cnt = 24 כמו בקוד התוכנית כשכניסת si = ‘1’ אפשר לראות שהתווסף ‘1’ לוגי במשתנה Temp ואכן יש עליית שעון.

קוד התוכנית שמבצע את הפעולה: elsif cnt=24 then temp(0)<=si; -- LSB

## מחלק תדר – BaudRate

תפקיד התוכנית – לקבל תדר של MHz50 ולהוציא תדר של Hz153,600.

עיקרון התוכנית:

כלומר

= 40000

רוצים חצי זמן מחזור מכיוון שרוצים D.C של 80%

סמל רכיב: - BaudRate



BPS = Bits Per Second

אנו צריכים תדר גבוה פי 16 מקצב קבלה/שליחת מידע למחשב שהוא BPS9600.

9600 x 16 = 153,600

בחרנו בקצב זה מכיוון שזהו הקצב הנפוץ ביותר לעבודה עם תקן RS232.

קוד התוכנית:



library ieee;

use ieee.std\_logic\_1164.all;

entity BOUD\_RATE is

port (clk : in bit;

fout : buffer bit);

end;

architecture behave of BOUD\_RATE is

signal cnt : integer range 0 to 162; --

begin

process (clk)

begin

if clk'event and clk='1' then

if cnt<162 then --

cnt<=cnt+1;

else

cnt<=0;

fout<=not fout;

end if;

end if;

end process;

end behave;

בזום אין:

דיאגרמות:



אפשר לראות שברגע שמשתנה cnt ספר עד 82 במוצא יש אחד וככה כל פעם הוא מחליף מצב קודם, ויוצא תדר של 153,600.

## מחלק מפענח תצוגה – Splitter

תפקיד התוכנית -לחלק את מפענח התצוגה לחצי, ככה שחצי אחד מראה מספרים מאפס עד 3 והחצי השני מראה מספרים מארבע עד 7 (high, low). בערכה שאנו עובדים איתה יש 2 מפענחי תצוגות, לכן יש את המחלק שמפענח אחד יציג מספר בהקס נמוך ואחד יציג מספר בהקס גבוה.

סמל הרכיב:



קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity spliter is

port ( din : in bit\_vector(7 downto 0);

high,low : out bit\_vector(3 downto 0));

end;

architecture behave of spliter is

begin

high<=din(7 downto 4);

low<=din(3 downto 0);

end behave;

דיאגרמה:

אפשר לראות שכשיש F4 ב – din כלומר 11110100 בבינארי, זה מחולק לשני חצאים, high מקבל את המספר הגבוה כלומר 1111 שזה F בהקס (hex) ו – low מקבל את המספר הנמוך כלומר 0100 שהוא 4.



בבינארי:

## מפענח תצוגה – (bin2hex) 7 Segment Decoder

תפקיד התוכנית **–** לקבל מספר בבינארי בכל ירידת שעון (כלומר כשיש '0' לוגי) ולפי זה להפעיל פסי תצוגה בשביל להציג את התצוגה. לכל פס יש שם – abcdefg. אם המספר שהתקבל הוא 4 אז רק הפסים - f,g,b,c יודלקו וככה יוצג המספר 4.



פס תצוגה ששמו הוא b.



סמל הרכיב:



קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity bin2hex is

port ( number : in integer range 0 to 15;

seg : out bit\_vector ( 6 downto 0 ) );

end;

architecture behave of bin2hex is

signal temp : bit\_vector ( 6 downto 0 );

begin

with number select

temp <= "1111110" when 0,

"0110000" when 1,

"1101101" when 2,

"1111001" when 3,

"0110011" when 4,

"1011011" when 5,

"1011111" when 6,

"1110000" when 7,

"1111111" when 8,

"1111011" when 9,

"1110111" when 10,

"0011111" when 11,

"1001110" when 12,

"0111101" when 13,

"1001111" when 14,

"1000111" when 15;

seg<= not temp;

end behave;



דיאגרמה:



אפשר לראות בדיאגרמה שלפי הקוד כל פעם שרוצים להציג מספר אז במוצא seg אפשר לראות אילו פסי תצוגה עובדים בשביל להציג את המספר (הפסים עובדים ב – '0' לוגי). לדוגמה ב – 0 אפשר לראות במוצא – 0000001. כלומר כל פסי התצוגה עובדים חוץ מ – g, וזה יוצר 0.

בגלל שאי אפשר להציג מספרים דו – ספרתיים במפענח תצוגה יחיד נשתמש בסיס hex בשביל להציג אותם.

A = 10, B = 11, C = 12, D = 13, E = 13, F = 15

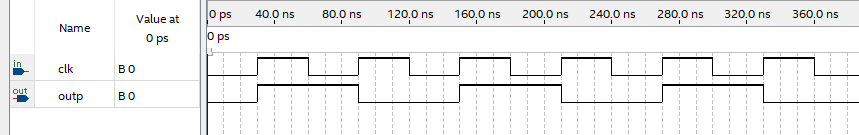
כלומר נציג את האותיות האלו על המפענח תצוגה.

## LED - נשימה

תפקיד התוכנית - להראות שהמערכת עובדת לפי הבהוב של LED על האלטרה כל שניה על ידי הוצאת תדר של 1Hz. מקבל שעון של 50MHz.

סמל הרכיב:





קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity led is

port(clk : in bit;

ledOUT : out STD\_LOGIC);

end;

architecture behave of led is

signal pulse : STD\_LOGIC := '0';

signal count : integer range 0 to 50000000 := 0;

begin

process (clk)

begin

if clk'event and clk = '1' then

if(count = 49999999) then

count <= 0;

pulse <= not pulse;

else

count <= count + 1;

end if;

end if;

end process;

ledOUT <= pulse;

end behave;



## PWM\_Motor

תפקיד התוכנית – להוציא תדר של 1KHz ב – D.C של 80% בשביל מנוע ה- DC.

סמל הרכיב:



דיאגרמה:



ניתן לראות שהמחלק תדר אכן מוציא תדר של 10KHz ב – D.C של 80% שזהו אפנון רוחב פס של המנועים הרצוי.

קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity PWM\_Motor is

port ( clk: in bit;

pOut : out STD\_LOGIC);

end;

architecture RTL of PWM\_Motor is

signal pulse : STD\_LOGIC := '1';

signal count : integer range 1 to 50000 := 1;

begin

process (clk)

begin

if clk'event and clk = '1' then

count <= count + 1;

if(count < 40000) then

pulse <= '1';

end if;

if(count > 40000) then

pulse <= '0';

end if;

if(count = 50000) then

count <= 1;

end if;

end if;

pOut <= pulse;

end process;

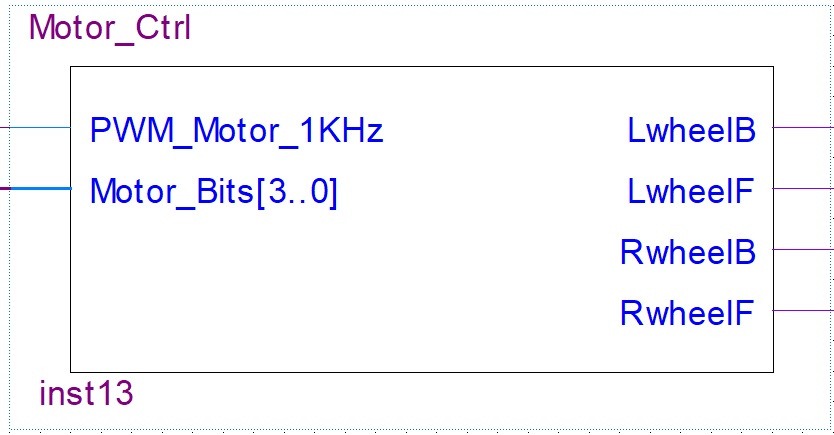
end RTL;



## PWM\_Ctrl

תפקיד התוכנית – להכניס את התדר של PWM\_Motor לגלגלים לפי פקודה מהאפליקציה ולהזיז את הרכב.

סמל הרכיב:



קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity Motor\_Ctrl is

port(PWM\_Motor\_1KHz: in bit;

Motor\_Bits: in bit\_vector (3 downto 0);

LwheelB: buffer bit;

LwheelF: buffer bit;

RwheelB: buffer bit;

RwheelF: buffer bit);

end;

architecture behave of Motor\_Ctrl is

begin

----------------------------------------------------------------------------------- Both wheels

process(PWM\_Motor\_1KHz,Motor\_Bits)

begin

-------------------------------------------------------------------- no movement

if Motor\_Bits = "0000" then LwheelB <= '0'; LwheelF <= '0'; RwheelB <= '0'; RwheelF <= '0'; end if;

-------------------------------------------------------------------- forward

if Motor\_Bits = "0101" then LwheelB <= '0'; LwheelF <= PWM\_Motor\_1KHz; RwheelB <= '0'; RwheelF <= PWM\_Motor\_1KHz; end if;

-------------------------------------------------------------------- backward

if Motor\_Bits = "1010" then LwheelB <= PWM\_Motor\_1KHz; LwheelF <= '0'; RwheelB <= PWM\_Motor\_1KHz; RwheelF <= '0'; end if;

-------------------------------------------------------------------- right

if Motor\_Bits = "0110" then LwheelB <= '0'; LwheelF <= PWM\_Motor\_1KHz; RwheelB <= PWM\_Motor\_1KHz; RwheelF <= '0'; end if;

-------------------------------------------------------------------- left

if Motor\_Bits = "1001" then LwheelB <= PWM\_Motor\_1KHz; LwheelF <= '0'; RwheelB <= '0'; RwheelF <= PWM\_Motor\_1KHz; end if;

-------------------------------------------------------------------- no movement

if Motor\_Bits = "1111" then LwheelB <= '0'; LwheelF <= '0'; RwheelB <= '0'; RwheelF <= '0'; end if;

end process;

end behave;

דיאגרמה:



ניתן לראות שכל המצבים בקוד אכן מיושמים בוייבפורם

## pmod\_temp\_sensor\_tcn75a

תפקיד התוכנית - לשלוח נתונים בני 9 סיביות בשביל להציג טמפרטורה הנקלטת על ידי חיישן הטמפרטורה LM75.

סמל הרכיב:



קוד התוכנית:

--הקוד לקוח מאתר בשם DIGIKEY

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY pmod\_temp\_sensor\_tcn75a IS --מוצג בבלוק ליד הsymbol של החיישון בדיאגרמת בלוקים

GENERIC(

sys\_clk\_freq : INTEGER := 50\_000\_000; --כניסת שעון מערכת

resolution : INTEGER := 9; --רזולוציה רצויה של הטמפרטורה בביט

temp\_sensor\_addr : STD\_LOGIC\_VECTOR(6 DOWNTO 0) := "1001000"); --כתובת הI2C של החיישן

PORT(

clk : IN STD\_LOGIC; --שעון מערכת

reset\_n : IN STD\_LOGIC; --מאפס את החיישן שיהיה לו הפסקה מידי פעם, בגבוה החיישן ימדוד בנמוך החיישן לא פעיל

scl : INOUT STD\_LOGIC; --I2C serial clock

sda : INOUT STD\_LOGIC; --I2C serial data

i2c\_ack\_err : OUT STD\_LOGIC; --מודיע כשיש שגיאה בחיישן משמש כדגל

temperature : OUT STD\_LOGIC\_VECTOR(resolution-1 DOWNTO 0)); --מידע הטמפרטורה בסיבית

END pmod\_temp\_sensor\_tcn75a;

ARCHITECTURE behavior OF pmod\_temp\_sensor\_tcn75a IS

TYPE machine IS(start, set\_resolution, set\_reg\_pointer, read\_data, output\_result); --מצבים של החיישן

SIGNAL state : machine; --מכונת מצבים

SIGNAL config : STD\_LOGIC\_VECTOR(7 DOWNTO 0); --ערך כדי להגדיר את רישום תצורת החיישן

SIGNAL i2c\_ena : STD\_LOGIC; --i2c משתמש עזר להפעלה

SIGNAL i2c\_addr : STD\_LOGIC\_VECTOR(6 DOWNTO 0); --i2c משתמש עזר לכתובת החיישן

SIGNAL i2c\_rw : STD\_LOGIC; --i2c משתמש עזר לפקודה של קריאה וכתיבה

SIGNAL i2c\_data\_wr : STD\_LOGIC\_VECTOR(7 DOWNTO 0); --i2c כותב מידע

SIGNAL i2c\_data\_rd : STD\_LOGIC\_VECTOR(7 DOWNTO 0); --i2c קורא מידע

SIGNAL i2c\_busy : STD\_LOGIC; --i2c משתמש עזר כשהקו תפוס

SIGNAL busy\_prev : STD\_LOGIC; --ערך קודם כאשר הקו תפוס בתקשורת I2C

SIGNAL temp\_data : STD\_LOGIC\_VECTOR(15 DOWNTO 0); --חוצץ מידע הטמפרטורה

COMPONENT i2c\_master IS

GENERIC(

input\_clk : INTEGER; --מידע כניסת השעון

bus\_clk : INTEGER); --מהירות בה רגל SCL תרוץ

PORT(

clk : IN STD\_LOGIC; --שעון מערכת

reset\_n : IN STD\_LOGIC; --ריסט בקצב נמוך כדי לא להעמיס על החיישן

ena : IN STD\_LOGIC; --מאפשר פעולת החיישן

addr : IN STD\_LOGIC\_VECTOR(6 DOWNTO 0); --כתובת העבד של החיישן

rw : IN STD\_LOGIC; --'0' זה כותב '1' זה קורא

data\_wr : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); --מידע לרשום לעבד I2C

busy : OUT STD\_LOGIC; --מודיע על העברת מידע בפעולתה

data\_rd : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0); --קריאת מידע מהעבד I2C

ack\_error : BUFFER STD\_LOGIC; --דגל מידע שגוי מהחיישן\עבד

sda : INOUT STD\_LOGIC; --יציאת מידע סדרתי מהחיישן בתקשורת I2C

scl : INOUT STD\_LOGIC); --יציאת שעון סדרתי מהחיישן בתקשורת I2C

END COMPONENT;

BEGIN

--מאתחל את התוכנית I2C\_MASTER

i2c\_master\_0: i2c\_master

GENERIC MAP(input\_clk => sys\_clk\_freq, bus\_clk => 400\_000)

PORT MAP(clk => clk, reset\_n => reset\_n, ena => i2c\_ena, addr => i2c\_addr,

rw => i2c\_rw, data\_wr => i2c\_data\_wr, busy => i2c\_busy,

data\_rd => i2c\_data\_rd, ack\_error => i2c\_ack\_err, sda => sda,

scl => scl);

--מגדיר את סיביות הרזולוציה עבור ערך רישום תצורת החיישן

WITH resolution SELECT

config <= "00100000" WHEN 10, --10 סיביות

"01000000" WHEN 11, --11 סיביות

"01100000" WHEN 12, --12 סיביות

"00000000" WHEN OTHERS; --9 סיביות(ברירת מחדל)

PROCESS(clk, reset\_n)

VARIABLE busy\_cnt : INTEGER RANGE 0 TO 2 := 0; --סופר את מעברי האות התפוסים במהלך העברה אחת

VARIABLE counter : INTEGER RANGE 0 TO sys\_clk\_freq/10 := 0; --סופר 100ms לחכות לפני תקשורת

BEGIN

IF(reset\_n = '0') THEN --כאשר reset\_n מופעל

counter := 0; --מאפס את המונה של העצירה

i2c\_ena <= '0'; --מאפס i2c enable

busy\_cnt := 0; --מאפס busy counter

temperature <= (OTHERS => '0'); --מאפס את המידע של הטמפרטורה

state <= start; --מאפס את מכונת המצבים

ELSIF(clk'EVENT AND clk = '1') THEN --בודק אם השעון בגבוה

CASE state IS --מכונת מצבים

--נותן לחיישן טמפ' 100ms להפעלה לפני התקשורת

WHEN start =>

IF(counter < sys\_clk\_freq/10) THEN --100ms עוד לא ספר

counter := counter + 1; --מוסיף למונה

ELSE --100ms הגיע ל

counter := 0; --מאפס מונה

state <= set\_resolution; --ממשיך לקביעת הרזולוציה של החיישן

END IF;

--קביעת הרזולוציה של החיישן

WHEN set\_resolution =>

busy\_prev <= i2c\_busy; --לוכד את הערך של האות התפוס הקודם של i2c

IF(busy\_prev = '0' AND i2c\_busy = '1') THEN --i2c busy עבר לגבוה

busy\_cnt := busy\_cnt + 1; --סופר את הפעמים busy הפך מנמוך לגבוה במהלך ההעברה

END IF;

CASE busy\_cnt IS --busy\_cnt עוקב אחר איזו פקודה אנחנו נמצאים

WHEN 0 => --אין שום פקודה נועחלת כרגע

i2c\_ena <= '1'; --התחל את העברה

i2c\_addr <= temp\_sensor\_addr; --קביעת הכתובת של החיישן טפמ'

i2c\_rw <= '0'; --הפקודה '1' זה כתיבה

i2c\_data\_wr <= "00000001"; --הגדר את מצביע הרשמה ל-Register Configuration

WHEN 1 => --קודם busy גבוה אחר כך פקודה 1 ננעלת ואפשר להמשיך לפקודה 2

i2c\_data\_wr <= config; --כתוב את ערך התצורה החדש ל-Configuration Register

WHEN 2 => --עכשיו busy גבוה ולכן פקודה 2 ננעלת

i2c\_ena <= '0'; --כאשר deassert מופעל ניתן להפסיק את העברה לאחר פקודה 2

IF(i2c\_busy = '0') THEN --העברה הושלמה

busy\_cnt := 0; --מאפס את busy\_cnt להעברה הבאה

state <= set\_reg\_pointer; --ממשיך להגדרת מצביע הרשמה לקריאות נתונים

END IF;

WHEN OTHERS => NULL;

END CASE;

--הגדר את מצביע האוגר למאגר טמפרטורת הסביבה

WHEN set\_reg\_pointer =>

busy\_prev <= i2c\_busy; --ללכוד את הערך של האות busy הקודם של i2c

IF(busy\_prev = '0' AND i2c\_busy = '1') THEN --i2c busy עלה לגבוה

busy\_cnt := busy\_cnt + 1; --סופר את הפעמים שbusy הפך מנמוך לגבוה במהלך ההעברה

END IF;

CASE busy\_cnt IS --busy\_cnt עוקב אחר איזו פקודה אנחנו נמצאים

WHEN 0 => --אין פקודה נעולה כרגע

i2c\_ena <= '1'; --התחלת העברת המידע

i2c\_addr <= temp\_sensor\_addr; --הגדרת הכתובת של החיישן

i2c\_rw <= '0'; --פקודה '1' זה כתיבה

i2c\_data\_wr <= "00000000"; --הגדר את מצביע ה-Register ל-Ambient Temperature Register

WHEN 1 => --1st busy גבוה: פקודה אחת נעולה

i2c\_ena <= '0'; --deassert אפשר לעצור את ההעברה לאחר פקודה 1

IF(i2c\_busy = '0') THEN --העברה הושלמה

busy\_cnt := 0; --מאפס את busy\_cnt להעברה הבאה

state <= read\_data; --ממשיך לקריאת מידע

END IF;

WHEN OTHERS => NULL;

END CASE;

--קורא את הטמפ' של הסביבה

WHEN read\_data =>

busy\_prev <= i2c\_busy; --ללכוד את הערך של האות busy הקודם של i2c

IF(busy\_prev = '0' AND i2c\_busy = '1') THEN --i2c busy עלה לגבוה

busy\_cnt := busy\_cnt + 1; --סופר את הפעמים שbusy הפך מנמוך לגבוה במהלך ההעברה

END IF;

CASE busy\_cnt IS --busy\_cnt עוקב אחר איזו פקודה אנחנו נמצאים

WHEN 0 => -- --אין פקודה נעולה כרגע

i2c\_ena <= '1'; ----התחלת העברת המידע

i2c\_addr <= temp\_sensor\_addr; ----הגדרת הכתובת של החיישן

i2c\_rw <= '1'; --פקודה '1' זה קריאה

WHEN 1 => --קודם busy : אחר כך פקודה 1 ננעלת, ניתן להוציא את פקודה 2

IF(i2c\_busy = '0') THEN --מודיע שמידע הנקרא בפקודה 1 זמין

temp\_data(15 DOWNTO 8) <= i2c\_data\_rd; --מחזיר את MSB מפקודה 1

END IF;

WHEN 2 => --קודם busy 2 גבוה, פקודה 2 ננעלת

i2c\_ena <= '0'; --deassert אפשר לעצור אץ ההעברה לאחר פקודה 2

IF(i2c\_busy = '0') THEN --מודיע שהמידע בפקודה 2 זמין

temp\_data(7 DOWNTO 0) <= i2c\_data\_rd; --מחזיר את LSB מפקודה 2

busy\_cnt := 0; --מאפס את busy\_cnt לפקודה הבאה

state <= output\_result; --ממשיך להוצאת הטמפ, לאחר כל התהליך

END IF;

WHEN OTHERS => NULL;

END CASE;

--הוצאת מידע הטמפ,

WHEN output\_result =>

temperature <= temp\_data(15 DOWNTO 16-resolution); --רושם את הטמפ, למוצא

state <= read\_data; --מחזיר את המידע הבא מהשלב הקודם של קריאת המידע

--חוזר להתחלה ומאפס הכל

WHEN OTHERS =>

state <= start;

END CASE;

END IF;

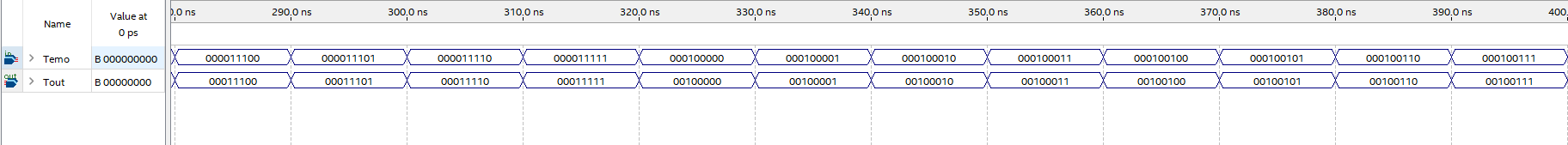
END PROCESS;

END behavior;

## Bit\_9\_to\_8 – ממיר 9 סיביות ל - 8

תפקיד התוכנית – לקבל ולהמיר 9 סיביות ל8 סיביות בלי לאבד מידע להצגת טמפרטורה.

סמל הרכיב:



דיאגרמת זמן:

ניתן לראות שהכניסה בגודל של 9 סיביות לעומת היציאה שהיא בגודל של 8 סיביות מה שמתאים לשליחת הטמפרטורה בתקן.

קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity bit\_9\_to\_8 is

port(temperature : in STD\_LOGIC\_VECTOR(8 DOWNTO 0);

tout: buffer STD\_LOGIC\_VECTOR( 7 downto 0));

end;

architecture behave of bit\_9\_to\_8 is

begin

process(temperature)

begin

tout(0) <= temperature(0);

tout(1) <= temperature(1);

tout(2) <= temperature(2);

tout(3) <= temperature(3);

tout(4) <= temperature(4);

tout(5) <= temperature(5);

tout(6) <= temperature(6);

tout(7) <= temperature(7);

end process;

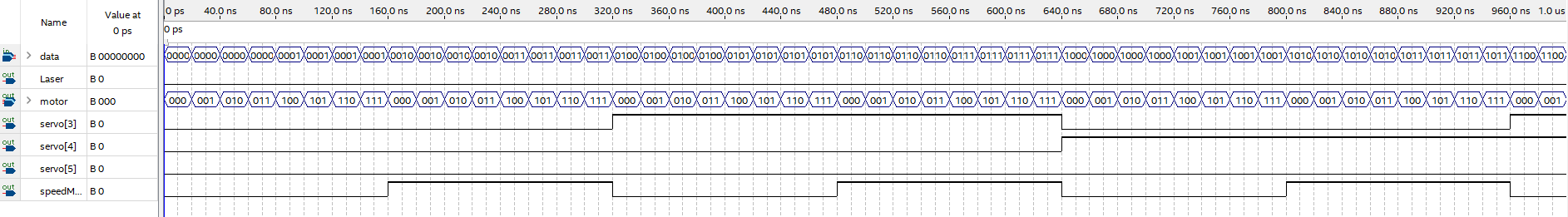
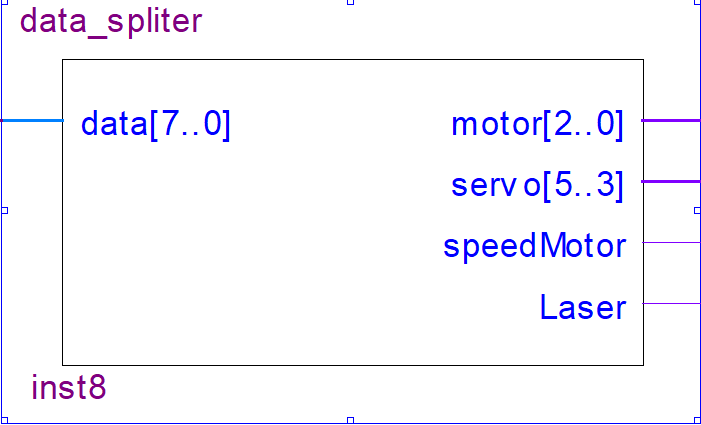
end behave;



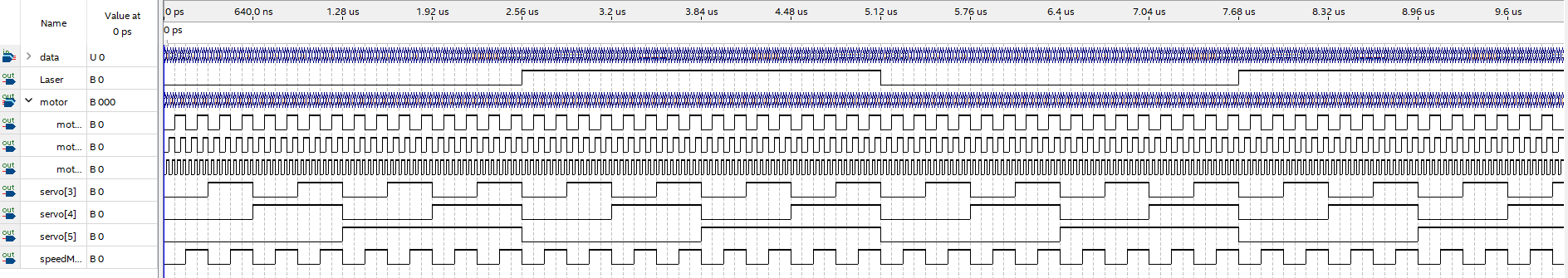
## data\_spliter – מחלק מידע

תפקיד תוכנית – לפרק ולחלק את המידע שהESP32 שולח לתוכניות המתאימות בהתכנון ההיררכי.

סמל הרכיב:



דיאגרמת זמן בזום אין:



דיאגרמת זמן:

ניתן לראות שכל המידע בכניסת data מתחלק כמו שצריך בשאר המשתנים שיוצאים לתוכניות המתאימות, לא ניתן לראות את סיבית הלייזר בזום אין כי קבענו בפאקט (כלומר המשתנה המכיל את כל המידע של המערכת שהבקר esp32 שולח לאלטרה מהאפליקציה.) שהיא (סיבית הלייזר) נשלחת אחרונה (packet = laser \* 128 + servo \* 16 + speed \* 8 + motors;)

קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity data\_spliter is

port (data : in bit\_vector(7 downto 0);

motor : out bit\_vector(2 downto 0);

servo : out bit\_vector(5 downto 3);

speedMotor : out bit;

Laser : out bit);

end;

architecture behave of data\_spliter is

begin

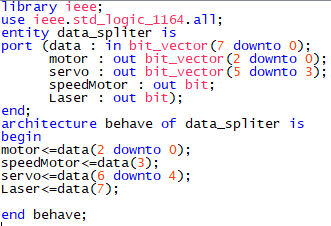
motor<=data(2 downto 0);

speedMotor<=data(3);

servo<=data(6 downto 4);

Laser<=data(7);

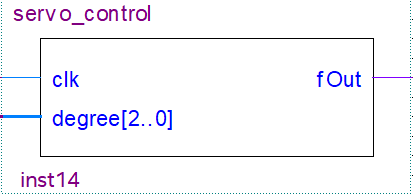
end behave;



## Servo\_control

תפקיד תוכנית – יצירת PWM מתאים למנוע סרבו, ולתת זווית לשמונה מצבים.

סמל הרכיב:



קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity servo\_control is

port (clk:in bit;

degree: in bit\_vector(2 downto 0);

fOut: out bit);

end;

architecture b of servo\_control is

signal cnt , cntMax : integer range 0 to 1000000 ; --50hz to servo

begin

cntMax<=20000 when degree="000" else

33125 when degree="001" else

46250 when degree="010" else

59375 when degree="011" else

72500 when degree="100" else

85625 when degree="101" else

98750 when degree="110" else

111875 when degree="111";

process (clk)

begin

if clk'event and clk='1' then

if cnt<1000000 then cnt<=cnt+1; else cnt<=0;

end if;

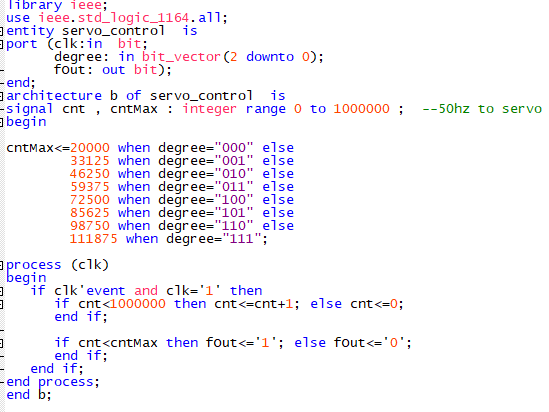
if cnt<cntMax then fOut<='1'; else fOut<='0';

end if;

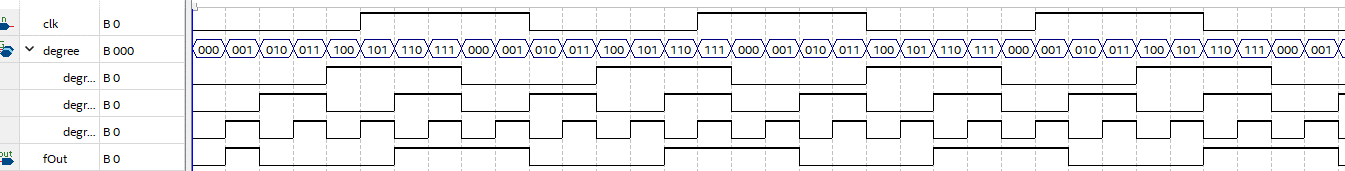
end if;

end process;

end b;



דיאגרמת זמן:

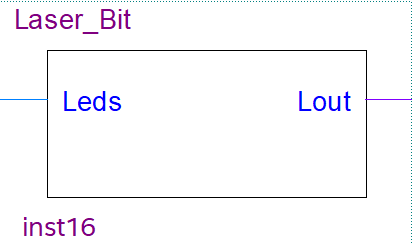


בדיאגרמות אלה ניתן שמשתנה היציאה מקבל רוחב פס (PWM) שונה לפי המספרים המוזנים לו במשתנה הכניסה - degree .

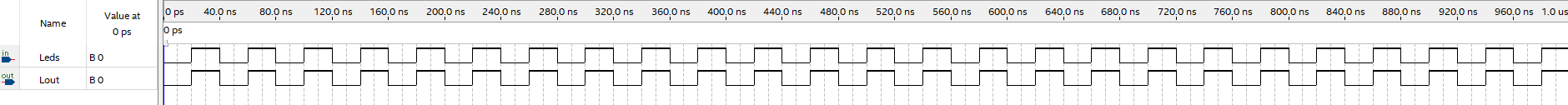
### Laser\_bit

תפקיד תוכנית – קליטת סיבית הפעלה (אם להפעיל את הלייזר או לא) ולשלוח ללייזר '1' או '0' בהתאם.

סמל הרכיב:



דיאגרמת זמן:



ניתן לראות שהכניסה והיציאה זהים כי כאשר יש עלייה בכניסה צריך עלייה ביציאה.

קוד התוכנית:

Library ieee;

use ieee.std\_logic\_1164.all;

entity Laser\_Bit is

port( Leds: in bit;

Lout: out bit);

end;

architecture behave of Laser\_Bit is

begin

process(Leds)

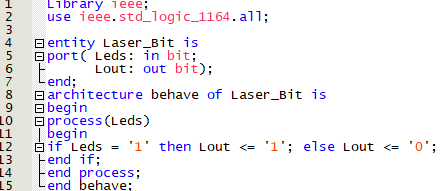
begin

if Leds = '1' then Lout <= '1'; else Lout <= '0';

end if;

end process;

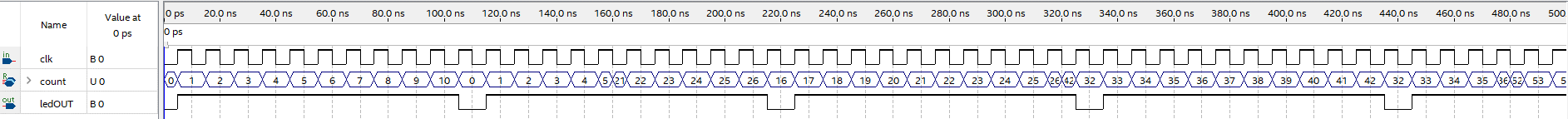
end behave;



## Low\_freq\_reset

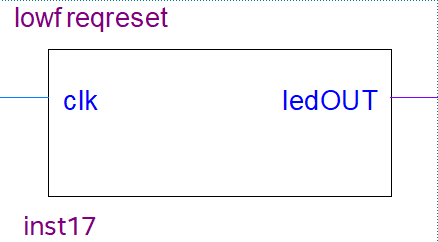
תפקיד תוכנית – יצירת תדר שמחליף בין אחד לוגי ואפס לוכי כל 10 שניות עם 90% D.C. (סופר עד 9 ומחליף לאפס לוגי לשנייה אחת).

דיאגרמת זמן:



ניתן לראות בתמונה שכל פעם שסיגנל count עובר את תשע הוא מאופס וחוזר לאפס והמוצא ledOUT עושה ריסט ויורד לאפס לוגי לשניה אחת וחוזר לאחד לוגי שניה אחרי. כמובן שיש דילאי אז זה לא נראה מדויק.

סמל הרכיב:



קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity lowfreqreset is

port(clk : in bit;

ledOUT : out STD\_LOGIC);

end;

architecture behave of lowfreqreset is

signal pulse : STD\_LOGIC := '0';

signal count : integer range 0 to 500000000 := 0;

begin

process (clk)

begin

if clk'event and clk = '1' then

if(count > 450000000) then

count <= 0;

pulse <= '0';

else

count <= count + 1;

pulse <= '1';

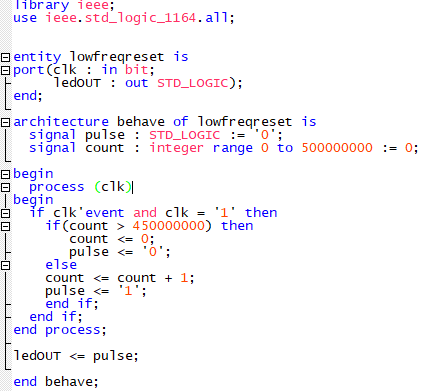
end if;

end if;

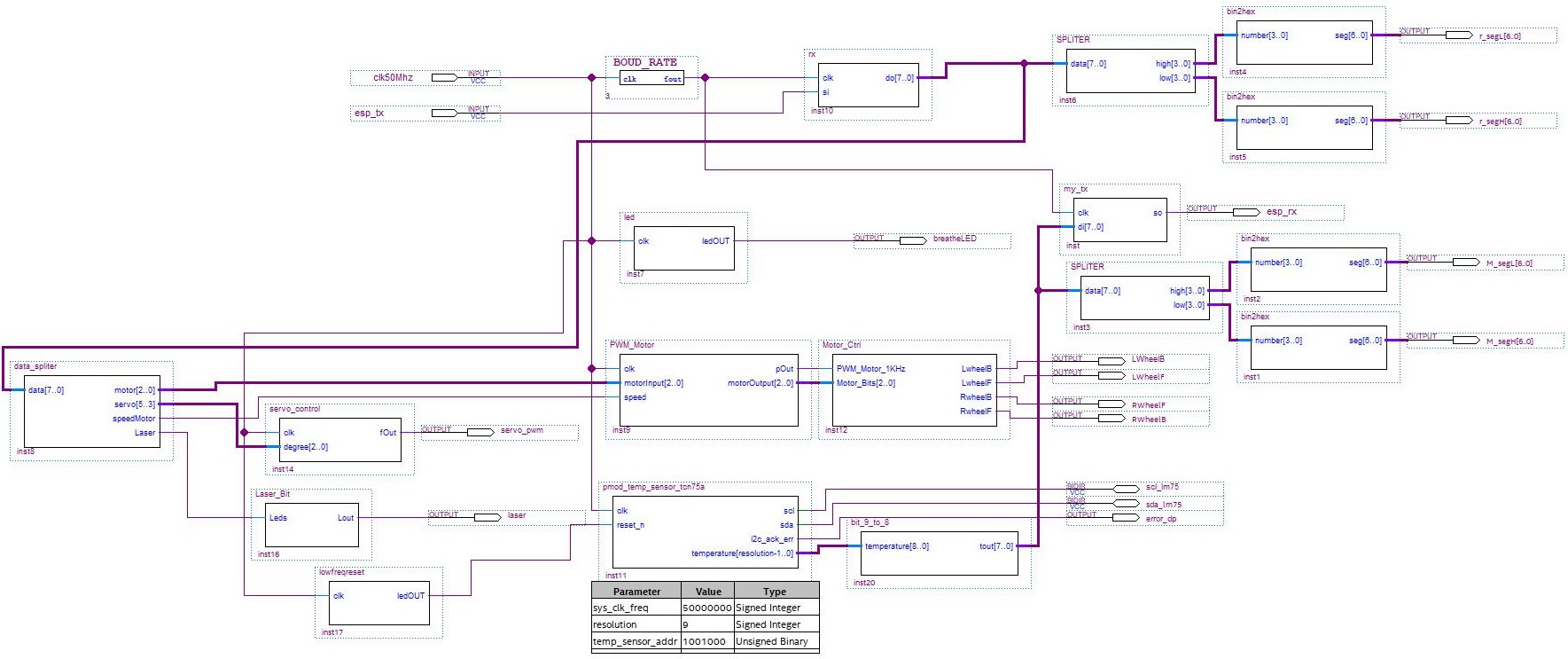
end process;

ledOUT <= pulse;

end behave;



## תכנון היררכי אלטרה



## PIN PLANNER

Pin Planner הוא כלי מפתח בחבילת התוכנה של Quartus המאפשרת למעצבים לנהל את הקצאות הפינים של העיצוב שלהם. הוא מספק ממשק גרפי המאפשר למשתמשים להקצות אותות לפינים של מכשיר היעד, להציג ולשנות את הקצאות הפינים ולפתור כל התנגשות שעלולה להתעורר במהלך התהליך.

מתכנן הפינים שימושי במיוחד עבור עיצובי FPGA ו-ASIC, שבהם הקצאות פינים הן קריטיות לתפקוד תקין של המכשיר. בעזרת Pin Planner, מעצבים יכולים להגדיר בקלות את הקישוריות בין הכניסות והיציאות של העיצוב שלהם לבין העולם החיצוני, שהוא חיוני להתממשקות נכונה ושילוב של המכשיר במערכת הגדולה יותר.

בסך הכל, Pin Planner הוא כלי חיוני המפשט את תהליך הקצאת הסיכות ומסייע למעצבים לייעל את הביצועים והאמינות של העיצובים שלהם.

# 

# 

# פרק 5 – C++ & Java

# שפת++ C

C++ היא שפת תכנות ברמה גבוהה הנמצאת בשימוש נרחב לפיתוח מגוון יישומים, כולל תוכנות מערכת, פיתוח משחקים ופיתוח אתרים. היא פותחה כהרחבה של שפת התכנות C, והוסיפה תכונות נוספות ופונקציונליות.

להלן כמה היבטים מרכזיים של C++:

* תחביר:
* תחביר C++דומה לשפת התכנות C. הוא משתמש בשילוב של הצהרות, ביטויים, פונקציות ומשתנים כדי להגדיר את ההיגיון וההתנהגות של תוכנית. הקוד מאורגן בדרך כלל בפונקציות, מחלקות ומרחבי שמות.
* תכנות מונחה עצמים (OOP):
* C++ תומך בתכנות מונחה עצמים, המאפשר לך להגדיר מחלקות ואובייקטים. מחלקות עוטמות נתונים והתנהגות לתוך אובייקטים, ומספקות דרך ליצור מודל של ישויות בעולם האמיתי. תכונות ה-OOP של C++ כוללות אנקפסולציה, תורשה ופולימורפיזם.
* ספריית תבניות רגילה (STL):
* הספרייה הסטנדרטית של C++ מספקת אוסף של מחלקות ואלגוריתמים הניתנים לשימוש חוזר הנקרא ספריית התבניות הסטנדרטית (STL). הוא כולל קונטיינרים (למשל וקטורים, רשימות, מפות) ואלגוריתמים (למשל מיון, חיפוש) שעוזרים בכתיבת קוד יעיל וניתן לשימוש חוזר.
* ניהול זיכרון:
* C++ מספק ניהול זיכרון ידני באמצעות תכונות כגון מצביעים והקצאת זיכרון דינמית באמצעות האופרטורים החדשים והמחקים. זה מאפשר שליטה מדויקת על השימוש בזיכרון, אבל זה גם דורש ניהול זהיר כדי למנוע דליפות זיכרון ובעיות אחרות.
* תבניות:
* C++ מציג את הרעיון של תבניות, המאפשרות תכנות גנרי. תבניות מאפשרות יצירת סוגים ופונקציות גנריות שיכולות לעבוד עם סוגי נתונים שונים, ומספקות גמישות וניתנות לשימוש חוזר בקוד.
* טיפול חריג:
* C++ כולל מנגנוני טיפול בחריגים להתמודדות עם שגיאות וחריגים בזמן ריצה. עם בלוקים של תפוס ניסיון, אתה יכול לתפוס ולטפל בחריגים בחן, למנוע קריסות של תוכנית ולאפשר שחזור שגיאות.
* תאימות עם C:
* C++ תואם לאחור עם C, כלומר, רוב תוכניות C ניתנות להידור ולביצוע כתוכניותC++.

C++מרחיב את C על ידי הוספת תכונות נוספות, כגון מחלקות ותבניות, תוך שמירה על היכולת להתקשר ולהשתמש בפונקציות וספריות C.

* תקינות:
* ל-C++ יש תקן ISO/IEC, כאשר הגרסה העדכנית ביותר היא C++17 (שוחררה ב-2017). תקן C++ מבטיח עקביות וניידות בשפה על פני מהדרים ופלטפורמות שונות.

ל- C++ מגוון רחב של יישומים, לרבות פיתוח תוכנה, פיתוח משחקים, מערכות משובצות, מחשוב מדעי ועוד. השילוב שלו של שליטה ברמה נמוכה, הפשטות ברמה גבוהה וביצועים הופכים אותו לבחירה פופולרית עבור תחומים שונים.

## 

# שפת – JAVA

Java היא שפת תכנות ברמה גבוהה, מבוססת class, object oriented, שתוכננה לכמה שפחות תלות ביישום. זוהי שפת תכנות למטרות כלליות שנועדה לאפשר למתכנתים לכתוב פעם אחת, לרוץ בכל מקום, כלומר, קוד ג'אווה מהידור יכול לרוץ בכל הפלטפורמות התומכות ב-Java ללא צורך בהידור מחדש .

יישומי Java מורכבים בדרך כלל לקוד ביטים שיכול לפעול בכל מכונה וירטואלית של Java (JVM) ללא קשר לארכיטקטורת המחשב הבסיסית. התחביר של Java דומה ל-C ו-C++, אך יש לו פחות מתקנים ברמה נמוכה מאשר כל אחד מהם. זמן הריצה של Java מספק יכולות דינמיות (כגון השתקפות ושינוי קוד זמן ריצה) שבדרך כלל אינן זמינות בשפות הידור מסורתיות.

נכון לשנת 2019, Java הייתה אחת משפות התכנות הפופולריות ביותר בשימוש לפי GitHub במיוחד עבור יישומי אינטרנט של שרת-לקוח, עם דיווח של 9 מיליון מפתחים.

Java פותחה במקור על ידי ג'יימס גוסלינג ב-Sun Microsystems. הוא שוחרר במאי 1995 כמרכיב ליבה של פלטפורמת Java של Sun Microsystems. מהדרים, מכונות וירטואליות וספריות מחלקות המקוריות והיישומיות של Java שוחררו במקור על ידי Sun תחת רישיונות קנייניים. החל ממאי 2007,

בהתאם למפרטים של תהליך הקהילה של Java, Sun העניקה רישיון מחדש לרוב טכנולוגיות ה-Java שלה תחת הרישיון GPL-2.0 בלבד. אורקל מציעה HotSpot Java Virtual Machine משלה, אולם יישום ההתייחסות הרשמי הוא OpenJDK JVM שהיא תוכנת קוד פתוח חינמית המשמשת את רוב המפתחים והיא ברירת המחדל של JVM עבור כמעט כל ההפצות של לינוקס.

עקרונות

היו חמש מטרות עיקריות ביצירת שפת Java:

* זה חייב להיות פשוט, מונחה עצמים ומוכר.
* זה חייב להיות חזק ומאובטח.
* זה חייב להיות ניטרלי אדריכלי ונייד.
* זה חייב לפעול בביצועים גבוהים.
* יש לפרש אותו, לשרשר ולדינמי.

## JVM – Java Virtual Machine

לכל שפת התכנות יש מהדרים משלהם כדי להדר את קוד המקור בזמן הביצוע. גם מהדר Java עושה את אותו הדבר. מהדר Java מתרגם את קוד המקור לקוד Byte עבור מכונה כלשהי שאינה קיימת ומכונה זו נקראת Java Virtual Machine.

מכונה וירטואלית של Java (JVM) היא מכונה וירטואלית המאפשרת למחשב להריץ תוכניות ג'אווה וכן תוכניות הכתובות בשפות אחרות, אשר מורכבות גם הן לקוד בייט של ג'אווה. ה-JVM מפורט על ידי מפרט המתאר באופן רשמי את הנדרש ביישום JVM.

קיום מפרט מבטיח יכולת פעולה הדדית של תוכניות Java על פני יישומים שונים, כך שמחברי תוכניות המשתמשים בערכת הפיתוח של Java (JDK) לא צריכים לדאוג לגבי אידיוסינקרטיות של פלטפורמת החומרה הבסיסית.

הטמעת התייחסות JVM פותחה על ידי פרויקט OpenJDK כקוד קוד פתוח וכוללת מהדר JIT בשם HotSpot. מהדורות Java הנתמכות מסחרית הזמינות מ-Oracle מבוססות על זמן הריצה של OpenJDK . Eclipse OpenJ9 הוא JVM נוסף בקוד פתוח עבור OpenJDK.

## OpenJDK

OpenJDK (ערכת פיתוח ג'אווה פתוחה) היא מימוש חינמי וקוד פתוח של פלטפורמת Java, Standard Edition (Java SE). היא תוצאה של מאמץ שהחלה Sun Microsystems בשנת 2006.

היישום מורשה תחת GPL-2.0 בלבד עם חריג קישור. אלמלא חריג הקישור של GPL, רכיבים המקושרים לספריית מחלקות Java יהיו כפופים לתנאי רישיון GPL. OpenJDK הוא יישום ההתייחסות הרשמי של Java SE מאז גרסה 7.

## JDK

ערכת הפיתוח של Java (JDK) היא הפצה של טכנולוגיית Java על ידי Oracle Corporation. הוא מיישם את Java Language Specification (JLS) ואת Java Virtual Machine Specification (JVMS) ומספק את המהדורה הסטנדרטית (SE) של Java Application Programming Interface (API).

זה נגזרת של OpenJDK המונע על ידי הקהילה שאורקל מנהלת. הוא מספק תוכנה לעבודה עם יישומי Java. דוגמאות לתוכנות כלולות הן המכונה הוירטואלית, מהדר, כלים לניטור ביצועים, מאתר באגים וכלי עזר אחרים ש-Oracle מחשיבה כשימושיים עבור מתכנת Java.

## קוד בקר ה – ESP32

#include <Arduino.h>

#include <WiFi.h>

#include <FirebaseESP32.h>

#include <WiFiMulti.h>

// Change to your Firebase RTDB project ID e.g. Your\_Project\_ID.firebaseio.com

#define FIREBASE\_HOST "alteratutorial-default-rtdb.europe-west1.firebasedatabase.app"

// Change to your Firebase RTDB secret password

#define FIREBASE\_AUTH "9U5cGGnuLfcHWckDeVCA5hvMSf1Ki55yUNlVU6BQ"

// legs of ESP that we receive and transmit

#define RXD2 16

#define TXD2 17

#define ledpin 2

// Define WifiMulti Variable

WiFiMulti wifiMulti;

// Define Firebase Data objects

FirebaseData fbdo;

int rx0;

// integers of the firebase

int motors, temperature, leds, packet, ServoMotors;

//Boolean which allows the LED to "breath"

bool a = false;

// function that shows the system working by flickering the LED of ESP

void blink();

void setup() {

//Serial - used for printing data, speed of reading from terminal

  Serial.begin(115200);

//Serial2 - sending and recieving data variable (bps, type of data, RX, TX)

  Serial2.begin(9600, SERIAL\_8N1, RXD2, TXD2);

//define the leg of output on the LED

  pinMode(ledpin, OUTPUT);

//define the wifi networks to search for and use

  wifiMulti.addAP("Ams\_2.4GHz", "0523993253A");

  wifiMulti.addAP("ORYAM", "12345678");

  wifiMulti.addAP("upstairs", "10203040");

  // WiFi.scanNetworks will return the number of networks found

  int n = WiFi.scanNetworks();

  Serial.println("scan done");

  if (n == 0) {

      Serial.println("no networks found");

  }

  else {

    Serial.print(n);

    Serial.println(" networks found");

    for (int i = 0; i < n; ++i) {

      // Print SSID and RSSI for each network found

      Serial.print(i + 1);

      Serial.print(": ");

      Serial.print(WiFi.SSID(i));

      Serial.print(" (");

      Serial.print(WiFi.RSSI(i));

      Serial.print(")");

      Serial.println((WiFi.encryptionType(i) == WIFI\_AUTH\_OPEN)?" ":"\*");

      delay(10);

    }

  }

  if(wifiMulti.run() != WL\_CONNECTED) {

  Serial.println("WiFi not connected!");

  delay(1000);

}

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.print("Connecting To: " );

  Serial.print(WiFi.SSID());

  Serial.println("");

  Serial.print("Connected! IP address: ");

  String ipAddress = WiFi.localIP().toString();;

  Serial.println(ipAddress);

//connects with Firebase, basically stores Firebase's legacy authentication credentials

  Firebase.begin(FIREBASE\_HOST, FIREBASE\_AUTH);

  Serial.println("Firebase Connected");

//reconnects WIFI if the connection is lost

  Firebase.reconnectWiFi(true);

}

// Always runs when the system is on

void loop() {

  while (Serial2.available()) {

    temperature = Serial2.read();

  }

// When firebase is ready to receive data

  if (Firebase.ready()) {

//Firebase reads the integer in the firebase database(the realtime console) and then fbdo = firebase data object

    Firebase.setInt(fbdo, "kar98Info/temperature", temperature);

    if (Firebase.getInt(fbdo, "kar98Info/carControl")) {

      packet = fbdo.intData();

    }

  }

//delay for the ESP so its not getting seizure (overload)

  delay(10);

// gets the stored bytes from the serial port that are available for reading

  if (Serial2.available()) {

//Serial.Write() Prints data to the serial port as human-readable ASCII text and this converts FromFB to binary

    Serial2.write(packet);

    delay(10);

  }

  fbdo.clear();

//LED blink to check if the loop worked

  blink();

}

// if a is false then a is changed to true and them runs a loop once and turns back to false and that causes the LED to "breath" or turn on and off a lot

void blink() {

  a = not a;

  if (a)

    digitalWrite(ledpin, HIGH);

  else

    digitalWrite(ledpin, LOW);

}

## קוד ESP32 – CAM

#include <WebSocketsServer.h>

#include <WiFi.h>

#include <WiFiUdp.h>

#include "camera\_wrap.h"

#include <vector>

#include <WiFiMulti.h>

  WiFiMulti wifiMulti;

// #define DEBUG

// #define SAVE\_IMG

enum TRACK{

  TRACK\_NONE = 0,

  TRACK\_FW,

  TRACK\_LEFT,

  TRACK\_RIGHT,

  TRACK\_STOP

};

// const char\* ssid = "Ams\_2.4GHz";    // <<< change this as yours

// const char\* password = "0523993253A"; // <<< change this as yours

//holds the current upload

int cameraInitState = -1;

uint8\_t\* jpgBuff = new uint8\_t[68123];

size\_t   jpgLength = 0;

uint8\_t camNo=0;

bool clientConnected = false;

//Creating UDP Listener Object.

WiFiUDP UDPServer;

IPAddress addrRemote;

unsigned int portRemote;

unsigned int UDPPort = 6868;

const int RECVLENGTH = 16;

byte packetBuffer[RECVLENGTH];

WebSocketsServer webSocket = WebSocketsServer(86);

String html\_home;

const int LED\_BUILT\_IN        = 4;

const uint8\_t TRACK\_DUTY      = 100;

const int PIN\_SERVO\_PITCH     = 12;

// const int PIN\_SERVO\_YAW       = 2;

const int PINDC\_LEFT\_BACK     = 13;

const int PINDC\_LEFT\_FORWARD  = 15;

const int PINDC\_RIGHT\_BACK    = 14;

const int PINDC\_RIGHT\_FORWARD = 2;

const int LEFT\_CHANNEL        = 2;

const int RIGHT\_CHANNEL       = 3;

const int SERVO\_PITCH\_CHANNEL = 4;

const int SERVO\_YAW\_CHANNEL   = 5;

const int SERVO\_RESOLUTION    = 16;

unsigned long previousMillisServo = 0;

const unsigned long intervalServo = 10;

bool servoUp = false;

bool servoDown = false;

bool servoRotateLeft = false;

bool servoRotateRight = false;

int posServo = 75;

int PWMTrackHIGH = 138;

int PWMTrackLOW = 138;

void servoWrite(uint8\_t channel, uint8\_t angle) {

  // regarding the datasheet of sg90 servo, pwm period is 20 ms and duty is 1->2ms

  uint32\_t maxDuty = (pow(2,SERVO\_RESOLUTION)-1)/10;

  uint32\_t minDuty = (pow(2,SERVO\_RESOLUTION)-1)/20;

  uint32\_t duty = (maxDuty-minDuty)\*angle/180 + minDuty;

  ledcWrite(channel, duty);

}

void controlServo(){

  if(servoUp){

    if(posServo>2){

      posServo -= 2;

    }

  }

  if(servoDown){

    if(posServo<180){

      posServo += 2;

    }

  }

  servoWrite(SERVO\_PITCH\_CHANNEL,posServo);

}

void controlDC(int left0, int left1, int right0, int right1){

  digitalWrite(PINDC\_LEFT\_BACK, left0);

  if(left1 == HIGH){

    ledcWrite(LEFT\_CHANNEL, 255);

  }else{

    ledcWrite(LEFT\_CHANNEL, 0);

  }

  digitalWrite(PINDC\_RIGHT\_BACK, right0);

  if(right1 == HIGH){

    ledcWrite(RIGHT\_CHANNEL, 255);

  }else{

    ledcWrite(RIGHT\_CHANNEL, 0);

  }

}

void controlDCTrack(int left, int right){

  digitalWrite(PINDC\_LEFT\_BACK, 0);

  ledcWrite(LEFT\_CHANNEL, left);

  digitalWrite(PINDC\_RIGHT\_BACK, 0);

  ledcWrite(RIGHT\_CHANNEL, right);

}

void webSocketEvent(uint8\_t num, WStype\_t type, uint8\_t \* payload, size\_t length) {

  switch(type) {

      case WStype\_DISCONNECTED:

          Serial.printf("[%u] Disconnected!\n", num);

          camNo = num;

          clientConnected = false;

          break;

      case WStype\_CONNECTED:

          Serial.printf("[%u] Connected!\n", num);

          clientConnected = true;

          break;

      case WStype\_TEXT:

      case WStype\_BIN:

      case WStype\_ERROR:

      case WStype\_FRAGMENT\_TEXT\_START:

      case WStype\_FRAGMENT\_BIN\_START:

      case WStype\_FRAGMENT:

      case WStype\_FRAGMENT\_FIN:

          Serial.println(type);

          break;

  }

}

std::vector<String> splitString(String data, String delimiter){

    std::vector<String> ret;

    // initialize first part (string, delimiter)

    char\* ptr = strtok((char\*)data.c\_str(), delimiter.c\_str());

    while(ptr != NULL) {

        ret.push\_back(String(ptr));

        // create next part

        ptr = strtok(NULL, delimiter.c\_str());

    }

    return ret;

}

void processUDPData(){

  int cb = UDPServer.parsePacket();

  if (cb) {

      UDPServer.read(packetBuffer, RECVLENGTH);

      addrRemote = UDPServer.remoteIP();

      portRemote = UDPServer.remotePort();

      String strPackage = String((const char\*)packetBuffer);

  #ifdef DEBUG

      Serial.print("receive: ");

      // for (int y = 0; y < RECVLENGTH; y++){

      //   Serial.print(packetBuffer[y]);

      //   Serial.print("\n");

      // }

      Serial.print(strPackage);

      Serial.print(" from: ");

      Serial.print(addrRemote);

      Serial.print(":");

      Serial.println(portRemote);

  #endif

      if(strPackage.equals("whoami")){

          UDPServer.beginPacket(addrRemote, portRemote-1);

          String res = "ESP32-CAM";

          UDPServer.write((const uint8\_t\*)res.c\_str(),res.length());

          UDPServer.endPacket();

          Serial.println("response");

      }else if(strPackage.equals("forward")){

        controlDC(LOW,HIGH,LOW,HIGH);

      }else if(strPackage.equals("backward")){

        controlDC(HIGH,LOW,HIGH,LOW);

      }else if(strPackage.equals("left")){

        controlDC(LOW,LOW,LOW,HIGH);

      }else if(strPackage.equals("right")){

        controlDC(LOW,HIGH,LOW,LOW);

      }else if(strPackage.equals("stop")){

        controlDC(LOW,LOW,LOW,LOW);

      }else if(strPackage.equals("camup")){

        servoUp = true;

      }else if(strPackage.equals("camdown")){

        servoDown = true;

      }else if(strPackage.equals("camstill")){

        servoUp = false;

        servoDown = false;

      }else if(strPackage.equals("ledon")){

        digitalWrite(LED\_BUILT\_IN, HIGH);

      }else if(strPackage.equals("ledoff")){

        digitalWrite(LED\_BUILT\_IN, LOW);

      }else if(strPackage.equals("lefttrack")){

        controlDCTrack(0, PWMTrackHIGH);

      }else if(strPackage.equals("righttrack")){

        controlDCTrack(PWMTrackHIGH, 0);

      }else if(strPackage.equals("fwtrack")){

        controlDCTrack(PWMTrackLOW, PWMTrackLOW);

      }

      memset(packetBuffer, 0, RECVLENGTH);

  }

}

void setup(void) {

  Serial.begin(115200);

  Serial.print("\n");

  WiFi.mode(WIFI\_STA);

  #ifdef DEBUG

  Serial.setDebugOutput(true);

  #endif

  wifiMulti.addAP("Ams\_2.4GHz", "0523993253A");

  wifiMulti.addAP("ORYAM", "12345678");

  wifiMulti.addAP("upstairs", "10203040");

  pinMode(LED\_BUILT\_IN, OUTPUT);

  digitalWrite(LED\_BUILT\_IN, LOW);

  pinMode(PINDC\_LEFT\_BACK, OUTPUT);

  ledcSetup(LEFT\_CHANNEL, 100, 8);//channel, freq, resolution

  ledcAttachPin(PINDC\_LEFT\_FORWARD, LEFT\_CHANNEL);

  pinMode(PINDC\_RIGHT\_BACK, OUTPUT);

  ledcSetup(RIGHT\_CHANNEL, 100, 8);//channel, freq, resolution

  ledcAttachPin(PINDC\_RIGHT\_FORWARD, RIGHT\_CHANNEL);

  controlDC(LOW,LOW,LOW,LOW);

  // 1. 50hz ==> period = 20ms (sg90 servo require 20ms pulse, duty cycle is 1->2ms: -90=>90degree)

  // 2. resolution = 16, maximum value is 2^16-1=65535

  // From 1 and 2 => -90=>90 degree or 0=>180degree ~ 3276=>6553

  ledcSetup(SERVO\_PITCH\_CHANNEL, 50, 16);//channel, freq, resolution

  ledcAttachPin(PIN\_SERVO\_PITCH, SERVO\_PITCH\_CHANNEL);// pin, channel

  servoWrite(SERVO\_PITCH\_CHANNEL, posServo);

  // ledcSetup(SERVO\_YAW\_CHANNEL, 50, 16);//channel, freq, resolution

  // ledcAttachPin(PIN\_SERVO\_YAW, SERVO\_YAW\_CHANNEL);// pin, channel

  // servoWrite(SERVO\_YAW\_CHANNEL, posServo);

  cameraInitState = initCamera();

  Serial.printf("camera init state %d\n", cameraInitState);

  if(cameraInitState != 0){

    return;

  }

  // WiFi.scanNetworks will return the number of networks found

  int n = WiFi.scanNetworks();

  Serial.println("scan done");

  if (n == 0) {

      Serial.println("no networks found");

  }

  else {

    Serial.print(n);

    Serial.println(" networks found");

    for (int i = 0; i < n; ++i) {

      // Print SSID and RSSI for each network found

      Serial.print(i + 1);

      Serial.print(": ");

      Serial.print(WiFi.SSID(i));

      Serial.print(" (");

      Serial.print(WiFi.RSSI(i));

      Serial.print(")");

      Serial.println((WiFi.encryptionType(i) == WIFI\_AUTH\_OPEN)?" ":"\*");

      delay(10);

    }

  }

  if(wifiMulti.run() != WL\_CONNECTED) {

  Serial.println("WiFi not connected!");

  delay(1000);

}

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("");

  Serial.print("Connecting To: " );

  Serial.print(WiFi.SSID());

  Serial.println("");

  Serial.print("Connected! IP address: ");

  String ipAddress = WiFi.localIP().toString();;

  Serial.println(ipAddress);

  webSocket.begin();

  webSocket.onEvent(webSocketEvent);

  UDPServer.begin(UDPPort);

}

void loop(void) {

  webSocket.loop();

  if(clientConnected == true){

    grabImage(jpgLength, jpgBuff);

    webSocket.sendBIN(camNo, jpgBuff, jpgLength);

    // Serial.print("send img: ");

    // Serial.println(jpgLength);

  }

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillisServo >= intervalServo) {

    previousMillisServo = currentMillis;

    processUDPData();

    controlServo();

  }

  #ifdef DEBUG

  if (Serial.available()) {

    String data = Serial.readString();

    Serial.println(data);

    std::vector<String> vposVals = splitString(data, ",");

    if(vposVals.size() != 4){

      return;

    }

    int left0 = vposVals[0].toInt();

    int left1 = vposVals[1].toInt();

    int left2 = vposVals[2].toInt();

    int left3 = vposVals[3].toInt();

    controlDC(left0, left1, left2, left3);

  }

  #endif

}

# שפת תוויות XML

XML (extensible Markup Language) היא שפת סימון המשמשת לקידוד מסמכים בפורמט הניתן לקריאה אנושית וגם למכונה. הוא משמש למבנה, אחסון והובלה של נתונים, ולעתים קרובות משמש להחלפת נתונים דרך האינטרנט.

XML מבוסס על שפת סימון כללית (SGML) וחולקת רבות מהתכונות שלה. הבדל מרכזי אחד בין שתי השפות הוא ש-XML גמיש וניתן להרחבה יותר מ-SGML, מה שמאפשר למשתמשים להגדיר תגים ותכונות משלהם.

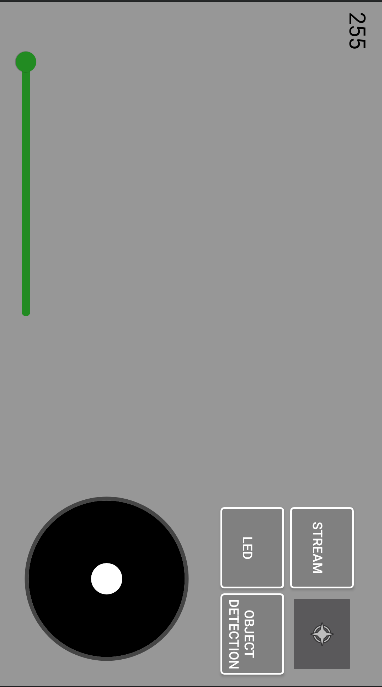
ל-XML יש מערכת כללים להגדרת אלמנטים ותכונות שניתן להשתמש בהם כדי לייצג נתונים בצורה מובנית. יש לו גם מערכת כללים לציון כיצד יש לארגן את האלמנטים והתכונות בתוך מסמך. ניתן לקנן אותם בתוך תגים אחרים כדי ליצור היררכיה של מידע.

XML נמצא בשימוש נרחב במגוון יישומים, כולל פיתוח אתרים, אחסון ושידור נתונים וניהול מסמכים. זהו כלי חשוב לארגון והחלפת נתונים בפורמט סטנדרטי. XML גם נתמך באופן נרחב על ידי שפות תכנות ויישומי תוכנה רבים ושונים, מה שהופך אותו לבחירה פופולרית לאחסון והחלפת נתונים.

אחד היתרונות המרכזיים של XML הוא שהוא מתאר את עצמו, מה שאומר שמבנה הנתונים מוגדר במפורש בתוך המסמך עצמו. זה מקל על מערכות שונות לפרש ולעבד את הנתונים, שכן המבנה מוגדר בבירור.

## האפליקציה צילומי מסך





## קוד - Esp32CameraFragment

package com.p4f.esp32camai;  
  
import android.app.AlertDialog;  
import android.app.Dialog;  
import android.content.DialogInterface;  
import android.content.res.AssetManager;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Matrix;  
import android.graphics.Paint;  
import android.graphics.Rect;  
import android.os.Bundle;  
import android.os.Handler;  
import android.text.InputType;  
import android.util.Log;  
import android.util.Pair;  
import android.util.Size;  
import android.view.Gravity;  
import android.view.LayoutInflater;  
import android.view.MotionEvent;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.FrameLayout;  
import android.widget.ImageButton;  
import android.widget.ImageView;  
import android.widget.LinearLayout;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
  
import com.google.android.material.slider.Slider;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
import com.p4f.esp32camai.tflite.Classifier;  
import com.p4f.esp32camai.tflite.TFLiteObjectDetectionSSDAPIModel;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.net.InetAddress;  
import java.net.SocketAddress;  
import java.net.URI;  
import java.net.URISyntaxException;  
import java.nio.ByteBuffer;  
import java.util.ArrayList;  
import java.util.List;  
import android.graphics.Point;  
  
import org.java\_websocket.client.WebSocketClient;  
import org.java\_websocket.handshake.ServerHandshake;  
import org.opencv.android.OpenCVLoader;  
import org.opencv.android.Utils;  
import org.opencv.core.CvException;  
import org.opencv.core.Mat;  
import org.opencv.core.Scalar;  
import org.opencv.imgproc.Imgproc;  
import org.opencv.tracking.Tracker;  
import org.opencv.tracking.TrackerCSRT;  
import org.opencv.tracking.TrackerKCF;  
import org.opencv.tracking.TrackerMIL;  
import org.opencv.tracking.TrackerMOSSE;  
import org.opencv.tracking.TrackerMedianFlow;  
import org.opencv.tracking.TrackerTLD;  
  
import io.github.controlwear.virtual.joystick.android.JoystickView;  
  
public class Esp32CameraFragment extends Fragment{  
  
 public enum STATE {  
 *STOP*,  
 *RUN*,  
 *PAUSE* };  
  
 enum Drawing{  
 *DRAWING*,  
 *TRACKING*,  
 *CLEAR*,  
 }  
  
 final String TAG = "ExCameraFragment";  
  
 private UDPSocket mUdpClient;  
 private String mServerAddressBroadCast = "255.255.255.255";  
 InetAddress mServerAddr;  
 int mServerPort = 6868;  
 final byte[] mRequestConnect = new byte[]{'w','h','o','a','m','i'};  
 final byte[] mRequestForward = new byte[]{'f','o','r','w','a','r','d'};  
 final byte[] mRequestForwardTrack = new byte[]{'f','w','t','r','a','c','k'};  
 final byte[] mRequestBackward = new byte[]{'b','a','c','k','w','a','r','d'};  
 final byte[] mRequestLeft = new byte[]{'l','e','f','t'};  
 final byte[] mRequestLeftTrack = new byte[]{'l','e','f','t','t','r','a','c','k'};  
 final byte[] mRequestRight = new byte[]{'r','i','g','h','t'};  
 final byte[] mRequestRightTrack = new byte[]{'r','i','g','h','t','t','r','a','c','k'};  
 final byte[] mRequestStop = new byte[]{'s','t','o','p'};  
 final byte[] mRequestCamUp = new byte[]{'c','a','m','u','p'};  
 final byte[] mRequestCamDown = new byte[]{'c','a','m','d','o','w','n'};  
 final byte[] mRequestCamLeft = new byte[]{'c','a','m','l','e','f','t'};  
 final byte[] mRequestCamRight = new byte[]{'c','a','m','r','i','g','h','t'};  
 final byte[] mRequestCamStill = new byte[]{'c','a','m','s','t','i','l','l'};  
 final byte[] mLedOn = new byte[]{'l','e','d','o','n'};  
 final byte[] mLaserOn = new byte[]{'s','p','e','e','d','o','n'};  
 final byte[] mLaserOff = new byte[]{'s','p','e','e','d','o','f','f'};  
 final byte[] mLedOff = new byte[]{'l','e','d','o','f','f'};  
 private Handler handler = new Handler();  
 private Bitmap mBitmap;  
  
  
 ImageView mServerImageView;  
 Handler mHandler = new Handler();  
  
 private WebSocketClient mWebSocketClient;  
 private String mServerExactAddress;  
 private boolean mInitStream = false;  
 private boolean mInitTrackObj = false;  
 private boolean mStream = false;  
 private boolean mObjDet = false;  
 private boolean mLed = false;  
 private boolean mLaser = false;  
  
 private Classifier detectorSSD;  
 private List<TFLiteObjectDetectionSSDAPIModel.Recognition> detectorSSDResult = new ArrayList<>();  
  
 private final Size CamResolution = new Size(640, 480);  
  
 private OverlayView mTrackingOverlay;  
 private Bitmap mBitmapDebug;  
 private boolean mProcessing = false;  
 private Point[] mPoints = new Point[4];  
 private Point mPointCircle = new Point();  
 private int mRadiusCircle = 0;  
 private Drawing mDrawing = Drawing.*CLEAR*;  
 private boolean mTargetLocked = false;  
 private Bitmap mBitmapGrab = null;  
  
 private String mSelectedTracker = "None";  
 private String mSelectedTrackerPre = "None";  
 private Tracker mTracker;  
 private Mat mMatGrabInit;  
 private Mat mMatGrab;  
 private org.opencv.core.Rect2d mInitRectangle = null;  
 private int mBinaryThreshold = 80;  
 private int mRadioIndex = 0;  
 int packet,motors,servo,speed,laser;  
 TextView tempText;  
 Slider servoSlider;  
  
  
  
 private Bitmap mBitmapLaneTracking = null;  
 // final String[] mRadioBtnNames = {  
// "None",  
// "TrackerMedianFlow",  
// "TrackerCSRT",  
// "TrackerKCF",  
// "TrackerMOSSE",  
// "TrackerTLD",  
// "TrackerMIL",  
// "LaneTracking"  
// };  
 final String[] mRadioBtnNames = {  
 "None",  
 "ObjectTracking",  
 "LaneTracking"  
 };  
  
 public void onWindowFocusChanged(){  
 int testW = mTrackingOverlay.getWidth();  
 int testH = mTrackingOverlay.getHeight();  
// mTrackingOverlay.setLayoutParams(new FrameLayout.LayoutParams(testW, CamResolution.getWidth()/CamResolution.getHeight()\*testW));  
// mServerImageView.setLayoutParams(new FrameLayout.LayoutParams(testW, CamResolution.getWidth()/CamResolution.getHeight()\*testW));  
 }  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 mUdpClient = new UDPSocket(12345);  
 mUdpClient.runUdpServer();  
  
 try {  
 mServerAddr = InetAddress.*getByName*(mServerAddressBroadCast);  
 }catch (Exception e){  
  
 }  
  
 AssetManager assetManager = getActivity().getAssets();  
 if (MyConstants.*DEBUG*) {  
 try {  
 InputStream istr = assetManager.open("image1.jpg");  
 Bitmap tmpBitmap = BitmapFactory.*decodeStream*(istr);  
 mBitmapDebug = Bitmap.*createScaledBitmap*(tmpBitmap, CamResolution.getWidth(), CamResolution.getHeight(), false);  
 } catch (IOException e) {  
 // handle exception  
 }  
 }  
  
 if (!OpenCVLoader.*initDebug*()) {  
 Log.*d*(TAG, "Internal OpenCV library not found. Using OpenCV Manager for initialization");  
 OpenCVLoader.*initAsync*(OpenCVLoader.*OPENCV\_VERSION\_3\_4\_0*, getActivity(), null);  
 }  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup parent, Bundle savedInstanceState) {  
 View rootView = inflater.inflate(R.layout.*fragment\_camera*, parent, false);  
  
  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 DatabaseReference dbRef = database.getReference("kar98Info");  
  
 tempText = (TextView) rootView.findViewById(R.id.*tempView*);  
 servoSlider = (Slider) rootView.findViewById(R.id.*servoSlider*);  
 JoystickView joystick = (JoystickView) rootView.findViewById(R.id.*joystick*);  
 joystick.setAutoReCenterButton(true);  
  
 joystick.setOnMoveListener(new JoystickView.OnMoveListener() {  
 @Override  
 public void onMove(int angle, int strength) {  
 CompilePacket(dbRef);  
 if(strength < 30){ //stop car when no input  
 motors = 0;  
 speed = 0;  
 return;  
 }  
 CompilePacket(dbRef);  
 if (strength > 90){  
 speed = 1;  
 CompilePacket(dbRef);  
 }  
 if(angle < 135 && angle > 45) //forward  
 motors =2;  
 CompilePacket(dbRef);  
  
 if(angle < 315 && angle > 225)//backward  
 motors =1;  
 CompilePacket(dbRef);  
  
 if(angle < 45 && angle > 0 || angle > 315 && angle < 360)  
 {  
 //right  
 motors =4;  
 }  
 CompilePacket(dbRef);  
  
 if(angle < 180 && angle > 135 || angle > 180 && angle < 225)  
 {  
 //left  
 motors =3;  
 }  
 CompilePacket(dbRef);  
  
 }  
 });  
  
 mServerImageView = (ImageView)rootView.findViewById(R.id.*imageView*);  
 Button streamBtn = (Button) rootView.findViewById(R.id.*streamBtn*);  
 streamBtn.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v){  
 if (!mStream) {  
 try {  
 mServerAddr = InetAddress.*getByName*(mServerAddressBroadCast);  
 }catch (Exception e){  
  
 }  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestConnect);  
 Pair<SocketAddress, String> res = mUdpClient.getResponse();  
 int cnt = 3;  
 while (res.first == null && cnt > 0) {  
 res = mUdpClient.getResponse();  
 cnt--;  
 }  
 if (res.first != null) {  
 Log.*d*(TAG, res.first.toString() + ":" + res.second);  
 mServerExactAddress = res.first.toString().split(":")[0].replace("/","");  
 mStream = true;  
 connectWebSocket();  
 ((Button) getActivity().findViewById(R.id.*streamBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg\_2*);  
 ((Button) getActivity().findViewById(R.id.*streamBtn*)).setTextColor(Color.*rgb*(0,0,255));  
 try {  
 mServerAddr = InetAddress.*getByName*(mServerExactAddress);  
 }catch (Exception e){  
  
 }  
 }else{  
 Toast toast =  
 Toast.*makeText*(  
 getActivity(), "Cannot connect to ESP32 Camera", Toast.*LENGTH\_LONG*);  
 toast.setGravity(Gravity.*CENTER*, 0, 0);  
 toast.show();  
 }  
 } else {  
 mStream = false;  
 mWebSocketClient.close();  
 ((Button) getActivity().findViewById(R.id.*streamBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg*);  
 ((Button) getActivity().findViewById(R.id.*streamBtn*)).setTextColor(Color.*rgb*(255,255,255));  
 }  
 }  
 });  
  
 Button ledBtn = (Button) rootView.findViewById(R.id.*ledBtn*);  
 ledBtn.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v){  
 if (!mLed) {  
 mLed = true;  
 ((Button) getActivity().findViewById(R.id.*ledBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg\_2*);  
 ((Button) getActivity().findViewById(R.id.*ledBtn*)).setTextColor(Color.*rgb*(0,0,255));  
 dbRef.child("led").setValue(1);  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mLedOn);  
 }else{  
 mLed = false;  
 ((Button) getActivity().findViewById(R.id.*ledBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg*);  
 ((Button) getActivity().findViewById(R.id.*ledBtn*)).setTextColor(Color.*rgb*(255,255,255));  
 dbRef.child("led").setValue(0);  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mLedOff);  
 }  
 }  
 });  
  
 ImageButton ShootBtn = (ImageButton) rootView.findViewById(R.id.*shootBtn*);  
 ImageButton.OnTouchListener listener = new ImageButton.OnTouchListener() {  
 @Override  
 public boolean onTouch(View arg0, MotionEvent event){  
 if(event.getAction() == MotionEvent.*ACTION\_DOWN*) {  
 if (((ImageButton) arg0).getId() == R.id.*shootBtn*);  
 {  
 mLaser = true;  
 ((ImageButton) getActivity().findViewById(R.id.*shootBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg\_2*);  
 laser = 1;  
 CompilePacket(dbRef);  
 }  
 return true;  
  
 }  
 else if(event.getAction() == MotionEvent.*ACTION\_UP*) {  
 if (((ImageButton) arg0).getId() == R.id.*shootBtn*) {  
 mLaser = false;  
 ((ImageButton) getActivity().findViewById(R.id.*shootBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg*);  
 laser =0;  
 CompilePacket(dbRef);  
 }  
 return true;  
 }  
 return true;  
 }  
 };  
 ShootBtn.setOnTouchListener(listener);  
  
  
 servoSlider.addOnChangeListener(new Slider.OnChangeListener() {  
 @Override  
 public void onValueChange(@NonNull Slider slider, float value, boolean fromUser) {  
 switch((int)value) {  
 case 0:  
 servo=7;  
 CompilePacket(dbRef);  
 break;  
 case 10:  
 servo=6;  
 CompilePacket(dbRef);  
 break;  
 case 20:  
 servo=5;  
 CompilePacket(dbRef);  
 break;  
 case 30:  
 servo=4;  
 CompilePacket(dbRef);  
 break;  
 case 40:  
 servo=3;  
 CompilePacket(dbRef);  
 break;  
 case 50:  
 servo=2;  
 CompilePacket(dbRef);  
 break;  
 case 60:  
 servo=1;  
 CompilePacket(dbRef);  
 break;  
 case 70:  
 servo=0;  
 CompilePacket(dbRef);  
 break;  
 default:  
 servo=0;  
 CompilePacket(dbRef);  
 break;  
 }  
 }  
 });  
  
 dbRef.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 String temp = dataSnapshot.child("temperature").getValue().toString();  
 tempText.setText(CalculateCel(Integer.*parseInt*(temp)));  
 }  
 @Override  
 public void onCancelled(DatabaseError error) {  
 // Failed to read value  
 Log.*w*("Firebase", "Failed to read value.", error.toException());  
 }  
 });  
  
  
  
  
  
  
 Button objDetBtn = (Button) rootView.findViewById(R.id.*objDetBtn*);  
 objDetBtn.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v){  
 if (!mObjDet) {  
 ((Button) getActivity().findViewById(R.id.*objDetBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg\_2*);  
 ((Button) getActivity().findViewById(R.id.*objDetBtn*)).setTextColor(Color.*rgb*(0,0,255));  
 } else {  
 ((Button) getActivity().findViewById(R.id.*objDetBtn*)).setBackgroundResource(R.drawable.*my\_button\_bg*);  
 ((Button) getActivity().findViewById(R.id.*objDetBtn*)).setTextColor(Color.*rgb*(255,255,255));  
 }  
 mObjDet = !mObjDet;  
 }  
 });  
  
  
  
 try {  
 detectorSSD =  
 TFLiteObjectDetectionSSDAPIModel.*create*(  
 getActivity().getAssets(),  
 "ssdlite\_mobilenet\_v2\_quantized.tflite",  
 "",  
 300,  
 Classifier.Device.*CPU*,  
 MyConstants.MODEL\_TYPE.*UINT8*,  
 0.5f,  
 1,  
 CamResolution.getWidth(),  
 CamResolution.getHeight()  
 );  
 detectorSSD.startThread();  
 } catch (final IOException e) {  
 Log.*e*(TAG, "Exception initializing classifier!");  
 Toast toast =  
 Toast.*makeText*(  
 getActivity(), "Classifier could not be initialized", Toast.*LENGTH\_SHORT*);  
 toast.show();  
 }  
  
 for (int i=0; i<mPoints.length;i++){  
 mPoints[i] = new Point(0,0);  
 }  
  
 mTrackingOverlay = (OverlayView) rootView.findViewById(R.id.*tracking\_overlay*);  
 assert (mTrackingOverlay != null);  
  
 mTrackingOverlay.addCallback(  
 new OverlayView.DrawCallback() {  
 @Override  
 public void drawCallback(Canvas canvas) {  
 if (MyConstants.*DEBUG*) {  
 Rect dstRectForRender = new Rect(0, 0, mTrackingOverlay.getWidth(), mTrackingOverlay.getHeight());  
 Matrix matrix = new Matrix();  
 matrix.postRotate(90);  
 Bitmap scaleBitmap = Bitmap.*createScaledBitmap*(mBitmapDebug, mTrackingOverlay.getWidth(), mTrackingOverlay.getHeight(), false);  
 Bitmap rotatedBitmap = Bitmap.*createBitmap*(scaleBitmap, 0, 0, mTrackingOverlay.getWidth(), mTrackingOverlay.getHeight(), matrix, true);  
 canvas.drawBitmap(rotatedBitmap, null, dstRectForRender, null);  
 }  
 if (detectorSSD != null && mObjDet) {  
 int overlayWidth = mTrackingOverlay.getWidth();  
 int overlayHeight = mTrackingOverlay.getHeight();  
 int imgWidth = mBitmapGrab.getWidth();  
 int imgHeight = mBitmapGrab.getHeight();  
 ((TFLiteObjectDetectionSSDAPIModel) detectorSSD).getResult(detectorSSDResult);  
 Paint paint = new Paint();  
 Paint paintText = new Paint();  
 paint.setColor(Color.*rgb*(0, 255, 0));  
 Log.*d*(TAG, "Obj cnt: " + detectorSSDResult.size());  
 for (TFLiteObjectDetectionSSDAPIModel.Recognition det : detectorSSDResult) {  
 Log.*d*(TAG, "processing: " + det);  
 paint.setStrokeWidth(10);  
 paint.setStyle(Paint.Style.*STROKE*);  
 float left = det.getLocation().left \* mTrackingOverlay.getWidth();  
 if (left < 0) {  
 left = 0;  
 } else if (left > mTrackingOverlay.getWidth()) {  
 left = mTrackingOverlay.getWidth();  
 }  
  
 float top = det.getLocation().top \* mTrackingOverlay.getHeight();  
 if (top < 0) {  
 top = 0;  
 } else if (top > mTrackingOverlay.getHeight()) {  
 top = mTrackingOverlay.getHeight();  
 }  
  
 float right = det.getLocation().right \* mTrackingOverlay.getWidth();  
 ;  
 if (right < 0) {  
 right = 0;  
 } else if (right > mTrackingOverlay.getWidth()) {  
 right = mTrackingOverlay.getWidth();  
 }  
  
 float bottom = det.getLocation().bottom \* mTrackingOverlay.getHeight();  
 if (bottom < 0) {  
 bottom = 0;  
 } else if (bottom > mTrackingOverlay.getHeight()) {  
 bottom = mTrackingOverlay.getHeight();  
 }  
 paintText.setColor(Color.*BLUE*);  
 paintText.setStrokeWidth(2);  
 paintText.setStyle(Paint.Style.*FILL*);  
 paintText.setTextSize(50);  
 canvas.drawRect(left, top, right, bottom, paint);  
 paint.setStyle(Paint.Style.*FILL*);  
 String txt = det.getTitle();// + "(" + String.format("%.2f", det.getConfidence()) + ")";  
 canvas.drawRect(left, top, left-60, top+txt.length()\*30+50, paint);  
 canvas.save();  
 canvas.rotate(90, left-50, top + 50);  
 canvas.drawText(txt, left - 50, top + 50, paintText);  
 canvas.restore();  
 }  
 }  
  
 if(mSelectedTracker.equals("ObjectTracking") && mStream){  
 if(!mInitTrackObj){  
 String msg1 = "Object is selected by 1 touch and drag following by a";  
 String msg2 = "rectangle, make double touch with another finger to lock";  
 String msg3 = "the object, double touch again to release the tracking object";  
 Paint paintText = new Paint();  
 paintText.setColor(Color.*YELLOW*);  
 paintText.setStrokeWidth(2);  
 paintText.setStyle(Paint.Style.*FILL*);  
 paintText.setTextSize(mTrackingOverlay.getWidth()/23);  
 canvas.save();  
 canvas.rotate(90, mTrackingOverlay.getWidth()\*10/12, mTrackingOverlay.getHeight()/8);  
 canvas.drawText(msg1, mTrackingOverlay.getWidth()\*5/6, mTrackingOverlay.getHeight()/8, paintText);  
 canvas.restore();  
 canvas.save();  
 canvas.rotate(90, mTrackingOverlay.getWidth()\*9/12, mTrackingOverlay.getHeight()/8);  
 canvas.drawText(msg2, mTrackingOverlay.getWidth()\*9/12, mTrackingOverlay.getHeight()/8, paintText);  
 canvas.restore();  
 canvas.save();  
 canvas.rotate(90, mTrackingOverlay.getWidth()\*8/12, mTrackingOverlay.getHeight()/8);  
 canvas.drawText(msg3, mTrackingOverlay.getWidth()\*8/12, mTrackingOverlay.getHeight()/8, paintText);  
 canvas.restore();  
 canvas.save();  
 }  
 if(mDrawing != Drawing.*CLEAR*) {  
 Paint paint = new Paint();  
 paint.setColor(Color.*rgb*(0, 0, 255));  
 paint.setStrokeWidth(10);  
 paint.setStyle(Paint.Style.*STROKE*);  
 canvas.drawRect(mPoints[0].x, mPoints[0].y, mPoints[1].x, mPoints[1].y, paint);  
 if (mDrawing == Drawing.*TRACKING*) {  
 paint.setColor(Color.*rgb*(0, 255, 0));  
 canvas.drawLine((mPoints[0].x + mPoints[1].x) / 2,  
 0,  
 (mPoints[0].x + mPoints[1].x) / 2,  
 mTrackingOverlay.getHeight(),  
 paint);  
 canvas.drawLine(0,  
 (mPoints[0].y + mPoints[1].y) / 2,  
 mTrackingOverlay.getWidth(),  
 (mPoints[0].y + mPoints[1].y) / 2,  
 paint);  
 paint.setColor(Color.*YELLOW*);  
 paint.setStrokeWidth(2);  
 paint.setStyle(Paint.Style.*FILL*);  
 paint.setTextSize(30);  
 String strX = Integer.*toString*((mPoints[0].x + mPoints[1].x) / 2) + "/" + Integer.*toString*(mTrackingOverlay.getWidth());  
 String strY = Integer.*toString*((mPoints[0].y + mPoints[1].y) / 2) + "/" + Integer.*toString*(mTrackingOverlay.getHeight());  
 canvas.drawText(strX, (mPoints[0].x + mPoints[1].x) / 4, (mPoints[0].y + mPoints[1].y) / 2 - 10, paint);  
 canvas.save();  
 canvas.rotate(90, (mPoints[0].x + mPoints[1].x) / 2 + 10, (mPoints[0].y + mPoints[1].y) / 4);  
 canvas.drawText(strY, (mPoints[0].x + mPoints[1].x) / 2 + 10, (mPoints[0].y + mPoints[1].y) / 4, paint);  
 canvas.restore();  
 }  
 }  
 }else if(mSelectedTracker.equals("LaneTracking") && mStream){  
 Rect dstRectForRender = new Rect(0, 0, mTrackingOverlay.getWidth(), mTrackingOverlay.getHeight());  
 Matrix matrix = new Matrix();  
 matrix.postRotate(90);  
 Bitmap scaleBitmap = Bitmap.*createScaledBitmap*(mBitmapLaneTracking, mTrackingOverlay.getWidth(), mTrackingOverlay.getHeight(), false);  
 Bitmap rotatedBitmap = Bitmap.*createBitmap*(scaleBitmap, 0, 0, mTrackingOverlay.getWidth(), mTrackingOverlay.getHeight(), matrix, true);  
 Paint alphaPaint = new Paint();  
 alphaPaint.setAlpha(42);  
 canvas.drawBitmap(rotatedBitmap, null, dstRectForRender, alphaPaint);  
 }else if(mSelectedTracker.equals("ColorTracking") && mStream){  
 Paint paint = new Paint();  
 paint.setColor(Color.*argb*(50,0, 0, 255));  
 paint.setStrokeWidth(10);  
 paint.setStyle(Paint.Style.*FILL*);  
 canvas.drawCircle(mPointCircle.x, mPointCircle.y, mRadiusCircle, paint);  
 }else if(mSelectedTracker.equals("None") && mStream){  
 mInitTrackObj = false;  
 //*TODO* if(!mInitStream){  
 canvas.save();  
 canvas.rotate(90, mTrackingOverlay.getWidth()\*5/6, mTrackingOverlay.getHeight()/8);  
 canvas.restore();  
 canvas.save();  
 canvas.rotate(90, mTrackingOverlay.getWidth()/6, mTrackingOverlay.getHeight()/8);  
 canvas.restore();  
 }  
 }  
 }  
 }  
 );  
  
 mTrackingOverlay.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View view, MotionEvent event) {  
 final int X = (int) event.getX();  
 final int Y = (int) event.getY();  
 Log.*d*(TAG, ": " + Integer.*toString*(X) + " " + Integer.*toString*(Y) );  
 mInitStream = true;  
 mInitTrackObj = true;  
 switch (event.getAction() & MotionEvent.*ACTION\_MASK*) {  
 case MotionEvent.*ACTION\_UP*:  
// Log.d(TAG, ": " + "MotionEvent.ACTION\_UP" );  
 if (mSelectedTracker.equals("None")) {  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestCamStill);  
 break;  
 }  
 if(!mTargetLocked) {  
 mDrawing = Drawing.*CLEAR*;  
 mTrackingOverlay.postInvalidate();  
 }  
 break;  
 case MotionEvent.*ACTION\_POINTER\_DOWN*:  
// Log.d(TAG, ": " + "MotionEvent.ACTION\_POINTER\_DOWN" );  
 if(mSelectedTracker.equals("ObjectTracking")==false){  
 break;  
 }  
 if (mTargetLocked == false) {  
 if((mPoints[0].x-mPoints[1].x != 0) && (mPoints[0].y-mPoints[1].y != 0)) {  
 mTargetLocked = true;  
 mMatGrab = new Mat();  
 Toast toast = Toast.*makeText*(getActivity(), "Target is LOCKED !", Toast.*LENGTH\_LONG*);  
 toast.setGravity(Gravity.*TOP* | Gravity.*CENTER*, 0, 0);  
 toast.show();  
 }else{  
 mTargetLocked = false;  
 }  
 }else{  
 mTargetLocked = false;  
 Toast toast = Toast.*makeText*(getActivity(), "Target is UNLOCKED !", Toast.*LENGTH\_LONG*);  
 toast.setGravity(Gravity.*TOP* | Gravity.*CENTER*, 0, 0);  
 toast.show();  
 }  
 mDrawing = Drawing.*DRAWING*;  
 mTrackingOverlay.postInvalidate();  
 break;  
 case MotionEvent.*ACTION\_POINTER\_UP*:  
// Log.d(TAG, ": " + "MotionEvent.ACTION\_POINTER\_UP" );  
 break;  
 case MotionEvent.*ACTION\_DOWN*:  
// Log.d(TAG, ": " + "MotionEvent.ACTION\_DOWN" );  
 if (mSelectedTracker.equals("None")) {  
 if (X < mTrackingOverlay.getWidth() / 2) {  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestCamDown);  
 } else {  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestCamUp);  
 }  
 break;  
 }  
 if(!mTargetLocked && mSelectedTracker.equals("ObjectTracking")) {  
 mDrawing = Drawing.*DRAWING*;  
 mPoints[0].x = X;  
 mPoints[0].y = Y;  
 mPoints[1].x = X;  
 mPoints[1].y = Y;  
 mTrackingOverlay.postInvalidate();  
 }  
 break;  
 case MotionEvent.*ACTION\_MOVE*:  
// Log.d(TAG, ": " + "MotionEvent.ACTION\_MOVE" );  
 if(!mTargetLocked && mSelectedTracker.equals("ObjectTracking")) {  
 mPoints[1].x = X;  
 mPoints[1].y = Y;  
 mTrackingOverlay.postInvalidate();  
 }  
 break;  
 }  
// if(mTargetLocked==true){  
// getView().findViewById(R.id.objTrackBtn).setEnabled(false);  
// }else{  
// getView().findViewById(R.id.objTrackBtn).setEnabled(true);  
// }  
 return true;  
 }  
 });  
  
 return rootView;  
 }  
  
 private void connectWebSocket() {  
 URI uri;  
 try {  
 uri = new URI("ws://"+mServerExactAddress+":86/");  
 } catch (URISyntaxException e) {  
 e.printStackTrace();  
 return;  
 }  
  
 mWebSocketClient = new WebSocketClient(uri) {  
 @Override  
 public void onOpen(ServerHandshake serverHandshake) {  
 Log.*d*("Websocket", "Open");  
 }  
  
 @Override  
 public void onClose(int i, String s, boolean b) {  
 Log.*d*("Websocket", "Closed " + s);  
 }  
  
 @Override  
 public void onMessage(String message){  
 Log.*d*("Websocket", "Receive");  
 }  
  
 @Override  
 public void onMessage(ByteBuffer message){  
// Log.d("Websocket", "Receive");  
 getActivity().runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 byte[] imageBytes= new byte[message.remaining()];  
 message.get(imageBytes);  
 final Bitmap bmp=BitmapFactory.*decodeByteArray*(imageBytes,0,imageBytes.length);  
 if (bmp == null)  
 {  
 return;  
 }  
 int viewWidth = mServerImageView.getWidth();  
 Matrix matrix = new Matrix();  
 matrix.postRotate(90);  
 final Bitmap bmp\_traspose = Bitmap.*createBitmap*(bmp, 0, 0, bmp.getWidth(), bmp.getHeight(), matrix, true );  
 float imagRatio = (float)bmp\_traspose.getHeight()/(float)bmp\_traspose.getWidth();  
 int dispViewH = (int)(viewWidth\*imagRatio);  
 mServerImageView.setImageBitmap(Bitmap.*createScaledBitmap*(bmp\_traspose, viewWidth, dispViewH, false));  
  
 mBitmapGrab = bmp;  
 mProcessing = detectorSSD.IsProcessing;  
 if (!mProcessing) {  
 processing();  
 }  
 }  
 });  
 }  
  
 @Override  
 public void onError(Exception e) {  
 Log.*d*("Websocket", "Error " + e.getMessage());  
 }  
 };  
 mWebSocketClient.connect();  
 }  
  
 private void trackingDlg(){  
 AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
 builder.setTitle("Tracker Selection");  
  
 final RadioButton[] rb = new RadioButton[mRadioBtnNames.length];  
 RadioGroup rg = new RadioGroup(getActivity()); //create the RadioGroup  
 rg.setOrientation(RadioGroup.*VERTICAL*);  
  
 for(int i=0; i < mRadioBtnNames.length; i++){  
 rb[i] = new RadioButton(getActivity());  
 rb[i].setText(" " + mRadioBtnNames[i]);  
 rb[i].setId(i + 100);  
 rg.addView(rb[i]);  
 if(mRadioBtnNames[i].equals(mSelectedTracker)){  
 rb[i].setChecked(true);  
 }  
 }  
  
 // This overrides the radiogroup onCheckListener  
 rg.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener()  
 {  
 public void onCheckedChanged(RadioGroup group, int checkedId){  
 // This will get the radiobutton that has changed in its check state  
 RadioButton checkedRadioButton = (RadioButton)group.findViewById(checkedId);  
 // This puts the value (true/false) into the variable  
 boolean isChecked = checkedRadioButton.isChecked();  
 if (isChecked)  
 {  
 // Changes the textview's text to "Checked: example radiobutton text"  
 int i = 0;  
 for( i = 0; i < mRadioBtnNames.length; i++) {  
 if(checkedRadioButton.getText().toString().replace(" ", "").equals(mRadioBtnNames[i])){  
 break;  
 }  
 }  
 mRadioIndex = i;  
 }  
 }  
 });  
  
 LinearLayout lay = new LinearLayout(getActivity());  
 lay.setOrientation(LinearLayout.*VERTICAL*);  
 lay.setPadding(0,30,0,0);  
 lay.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
 lay.addView(rg);  
  
 final TextView labelThresh = new TextView(getActivity());  
 labelThresh.setText("Binary Threshold:");  
 // Set up the input  
 final EditText binThresh = new EditText(getActivity());  
 binThresh.setBackground(null);  
 // Specify the type of input expected; this, for example, sets the input as a password, and will mask the text  
 binThresh.setInputType(InputType.*TYPE\_CLASS\_NUMBER*);  
 binThresh.setText(Integer.*toString*(mBinaryThreshold));  
 lay.addView(labelThresh);  
 lay.addView(binThresh);  
  
 builder.setView(lay);  
  
 // Set up the buttons  
// builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
// @Override  
// public void onClick(DialogInterface dialog, int which) {  
// mSelectedTrackerPre = mSelectedTracker;  
// mSelectedTracker = mRadioBtnNames[mRadioIndex];  
// if (!mSelectedTracker.equals("None")) {  
// ((Button) getActivity().findViewById(R.id.shootBtn)).setBackgroundResource(R.drawable.my\_button\_bg\_2);  
// ((Button) getActivity().findViewById(R.id.shootBtn)).setTextColor(Color.rgb(0,0,255));  
// } else {  
// ((Button) getActivity().findViewById(R.id.shootBtn)).setBackgroundResource(R.drawable.my\_button\_bg);  
// ((Button) getActivity().findViewById(R.id.shootBtn)).setTextColor(Color.rgb(255,255,255));  
// }  
// mBinaryThreshold = Integer.parseInt(binThresh.getText().toString());  
// }  
// });  
  
  
 builder.setCancelable(false);  
 Dialog dialog = builder.show();  
 }  
  
 private void processing() {  
  
 int overlayWidth = mTrackingOverlay.getWidth();  
 int overlayHeight = mTrackingOverlay.getHeight();  
 mRadiusCircle = 0;  
  
 if(mObjDet) {  
 if (MyConstants.*DEBUG*) {  
 detectorSSD.setBitmap(mBitmapDebug);  
 } else {  
 detectorSSD.setBitmap(mBitmapGrab);  
 }  
 }  
  
 if(mSelectedTracker.equals("LaneTracking")){  
 if(mMatGrab==null){  
 mMatGrab = new Mat();  
 }  
 Utils.*bitmapToMat*(mBitmapGrab, mMatGrab);  
 org.opencv.imgproc.Imgproc.*resize*(mMatGrab, mMatGrab, new org.opencv.core.Size(320,240));  
 Mat gray = new Mat();  
 Mat binary = new Mat();  
 org.opencv.imgproc.Imgproc.*cvtColor*(mMatGrab, gray, Imgproc.*COLOR\_RGBA2GRAY*);  
 org.opencv.imgproc.Imgproc.*threshold*( gray, binary, mBinaryThreshold, 255, 0 );  
  
 int y = binary.rows()\*2/3;  
 int x0 = -1;  
 for(int x = 0; x < binary.cols(); ++x){  
 if(binary.get(y,x)[0] < 125){  
 x0 = x;  
 break;  
 }  
 }  
  
 if(x0 < 0){  
 mUdpClient.sendBytes(mServerAddr,mServerPort, mRequestForwardTrack);  
 }else if(x0 < binary.width()/2){  
 mUdpClient.sendBytes(mServerAddr,mServerPort, mRequestRightTrack);  
 }else if(x0 > binary.width()/2) {  
 mUdpClient.sendBytes(mServerAddr,mServerPort, mRequestLeftTrack);  
 }  
  
 //*TODO: DEBUG* Mat tmp = new Mat();  
 try {  
 Imgproc.*cvtColor*(binary, tmp, Imgproc.*COLOR\_GRAY2BGRA*);  
 mBitmapLaneTracking = Bitmap.*createBitmap*(tmp.cols(), tmp.rows(), Bitmap.Config.*ARGB\_8888*);  
 Utils.*matToBitmap*(tmp, mBitmapLaneTracking);  
 }  
 catch (CvException e){  
 Log.*d*("Exception",e.getMessage());  
 }  
 }else if(mSelectedTracker.equals("ColorTracking")){  
 if(mMatGrab==null){  
 mMatGrab = new Mat();  
 }  
 Utils.*bitmapToMat*(mBitmapGrab, mMatGrab);  
 org.opencv.imgproc.Imgproc.*resize*(mMatGrab, mMatGrab, new org.opencv.core.Size(320,240));  
 Mat gray = new Mat();  
 org.opencv.imgproc.Imgproc.*cvtColor*(mMatGrab, gray, Imgproc.*COLOR\_RGBA2GRAY*);  
 Mat circles = new Mat();  
 org.opencv.imgproc.Imgproc.*HoughCircles*(gray, circles, Imgproc.*CV\_HOUGH\_GRADIENT*, 2, gray.rows()/4, 200, 120, 10, 80);  
 for (int i = 0; i < circles.cols(); i++) {  
 double[] vCircle = circles.get(0, i);  
  
 org.opencv.core.Point pt = new org.opencv.core.Point((int)Math.*round*(vCircle[0]), (int)Math.*round*(vCircle[1]));  
 int radius = (int)Math.*round*(vCircle[2]);  
  
 org.opencv.imgproc.Imgproc.*circle*(gray, pt, radius, new Scalar(255, 0, 0), 2);  
 mPointCircle.x = (int)(overlayWidth-pt.y\*overlayWidth/gray.rows());  
 mPointCircle.y = (int)(pt.x\*overlayHeight/gray.cols());  
 mRadiusCircle = radius\*overlayWidth/gray.rows();  
 }  
 //*TODO: DEBUG* Bitmap bmp = null;  
 Mat tmp = new Mat();  
 try {  
 Imgproc.*cvtColor*(gray, tmp, Imgproc.*COLOR\_GRAY2BGRA*);  
 bmp = Bitmap.*createBitmap*(tmp.cols(), tmp.rows(), Bitmap.Config.*ARGB\_8888*);  
 Utils.*matToBitmap*(tmp, bmp);  
 }  
 catch (CvException e){  
 Log.*d*("Exception",e.getMessage());  
 }  
 }else if(mTargetLocked && mSelectedTracker.equals("ObjectTracking")) {  
 Utils.*bitmapToMat*(mBitmapGrab, mMatGrab);  
 org.opencv.imgproc.Imgproc.*resize*(mMatGrab, mMatGrab, new org.opencv.core.Size(320,240));  
 org.opencv.imgproc.Imgproc.*cvtColor*(mMatGrab, mMatGrab, Imgproc.*COLOR\_RGBA2BGR*);  
  
 if(mDrawing==Drawing.*DRAWING*) {  
  
 int imgWidth = mMatGrab.cols();  
 int imgHeight = mMatGrab.rows();  
  
 int x0 = mPoints[0].y;  
 int y0 = overlayWidth - mPoints[0].x;  
 int x1 = mPoints[1].y;  
 int y1 = overlayWidth - mPoints[1].x;  
  
 int minX = (int)((float)Math.*min*(x0, x1)/overlayHeight\*mMatGrab.cols());  
 int minY = (int)((float)Math.*min*(y0, y1)/overlayWidth\*mMatGrab.rows());  
 int maxX = (int)((float)Math.*max*(x0, x1)/overlayHeight\*mMatGrab.cols());  
 int maxY = (int)((float)Math.*max*(y0, y1)/overlayWidth\*mMatGrab.rows());  
  
 mInitRectangle = new org.opencv.core.Rect2d(minX, minY, maxX-minX, maxY-minY);  
 mMatGrabInit = new Mat();  
 mMatGrab.copyTo(mMatGrabInit);  
  
 if(mSelectedTracker.equals("TrackerMedianFlow")) {  
 mTracker = TrackerMedianFlow.*create*();  
 }else if(mSelectedTracker.equals("TrackerCSRT")||mSelectedTracker.equals("ObjectTracking")) {  
 mTracker = TrackerCSRT.*create*();  
 }else if(mSelectedTracker.equals("TrackerKCF")) {  
 mTracker = TrackerKCF.*create*();  
 }else if(mSelectedTracker.equals("TrackerMOSSE")) {  
 mTracker = TrackerMOSSE.*create*();  
 }else if(mSelectedTracker.equals("TrackerTLD")) {  
 mTracker = TrackerTLD.*create*();  
 }else if(mSelectedTracker.equals("TrackerMIL")) {  
 mTracker = TrackerMIL.*create*();  
 }  
  
 mTracker.init(mMatGrabInit, mInitRectangle);  
 mDrawing = Drawing.*TRACKING*;  
  
 //*TODO: DEBUG*// org.opencv.core.Rect testRect = new org.opencv.core.Rect(minX, minY, maxX-minX, maxY-minY);  
// Mat roi = new Mat(mMatGrab, testRect);  
// Bitmap bmp = null;  
// Mat tmp = new Mat (roi.rows(), roi.cols(), CvType.CV\_8U, new Scalar(4));  
// try {  
// Imgproc.cvtColor(roi, tmp, Imgproc.COLOR\_RGB2BGRA);  
// bmp = Bitmap.createBitmap(tmp.cols(), tmp.rows(), Bitmap.Config.ARGB\_8888);  
// Utils.matToBitmap(tmp, bmp);  
// }  
// catch (CvException e){  
// Log.d("Exception",e.getMessage());  
// }  
  
 }else{  
 org.opencv.core.Rect2d trackingRectangle = new org.opencv.core.Rect2d(0, 0, 1,1);  
 mTracker.update(mMatGrab, trackingRectangle);  
  
// //*TODO: DEBUG*// org.opencv.core.Rect testRect = new org.opencv.core.Rect((int)trackingRectangle.x,  
// (int)trackingRectangle.y,  
// (int)trackingRectangle.width,  
// (int)trackingRectangle.height);  
// Mat roi = new Mat(mMatGrab, testRect);  
// Bitmap bmp = null;  
// Mat tmp = new Mat (roi.rows(), roi.cols(), CvType.CV\_8U, new Scalar(4));  
// try {  
// Imgproc.cvtColor(roi, tmp, Imgproc.COLOR\_RGB2BGRA);  
// bmp = Bitmap.createBitmap(tmp.cols(), tmp.rows(), Bitmap.Config.ARGB\_8888);  
// Utils.matToBitmap(tmp, bmp);  
// }  
// catch (CvException e){  
// Log.d("Exception",e.getMessage());  
// mTargetLocked = false;  
// mDrawing = Drawing.DRAWING;  
// }  
  
 mPoints[1].x = overlayWidth - (int)(trackingRectangle.y\*(float)overlayWidth/(float)mMatGrab.rows());  
 mPoints[0].y = (int)(trackingRectangle.x\*(float)mTrackingOverlay.getHeight()/(float)mMatGrab.cols());  
 mPoints[0].x = mPoints[1].x - (int)(trackingRectangle.height\*(float)mTrackingOverlay.getWidth()/(float)mMatGrab.rows());  
 mPoints[1].y = mPoints[0].y +(int)(trackingRectangle.width\*(float)mTrackingOverlay.getHeight()/(float)mMatGrab.cols());  
  
 int centerX = (mPoints[0].x+mPoints[1].x)/2;  
 int centerY = (mPoints[0].y+mPoints[1].y)/2;  
 if(centerX-mTrackingOverlay.getWidth()/2 > 150){  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestCamUp);  
 }else if(centerX-mTrackingOverlay.getWidth()/2 < -150){  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestCamDown);  
 }else{  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestCamStill);  
 }  
  
// if(centerY-mTrackingOverlay.getHeight()/2 > 200){  
// Log.d(TAG, ": " + (centerY-mTrackingOverlay.getHeight()/2) );  
// mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestRightTrack);  
// }else if(centerY-mTrackingOverlay.getHeight()/2 < -200){  
// Log.d(TAG, ": " + (centerY-mTrackingOverlay.getHeight()/2) );  
// mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestLeftTrack);  
// }  
  
 mTrackingOverlay.postInvalidate();  
 }  
 }else{  
 if(mSelectedTrackerPre != "None"){  
 mUdpClient.sendBytes(mServerAddr, mServerPort, mRequestStop);  
 }  
 if (mTracker != null) {  
 mTracker.clear();  
 mTracker = null;  
 }  
 }  
 mSelectedTrackerPre = mSelectedTracker;  
 mTrackingOverlay.invalidate();  
 }  
  
 public void onDestroy() {  
 Log.*e*(TAG, "onDestroy");  
 detectorSSD.requestStop();  
 detectorSSD.waitForExit();  
 mWebSocketClient.close();  
 super.onDestroy();  
 }  
 public String CalculateCel(int tempInC){  
 tempInC -= 32;  
 tempInC /= 1.8000;  
  
 return String.*valueOf*(Math.*abs*(tempInC));  
 }  
 public void CompilePacket(DatabaseReference dbRef){  
 packet = laser \* 128 + servo \* 16 + speed \* 8 + motors;  
 dbRef.child("carControl").setValue(packet);  
 }  
  
}

## קוד – MainActivity

package com.p4f.esp32camai;  
  
import android.os.Bundle;  
import android.view.Menu;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import androidx.fragment.app.FragmentManager;  
  
  
public class MainActivity extends AppCompatActivity implements FragmentManager.OnBackStackChangedListener{  
 private static final String *TAG* = "MainActivity";  
 private String mSelectedCam = "";  
 private String mSelectedCamPre = "";  
 Bundle mSavedInstanceState;  
 boolean mConnected = false;  
 private Esp32CameraFragment mFragmentCam = null;  
 private Menu mMenu = null;  
 Esp32CameraFragment cameraFragment;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 mSavedInstanceState = savedInstanceState;  
 getSupportFragmentManager().addOnBackStackChangedListener(this);  
 cameraFragment = new Esp32CameraFragment();  
 if (savedInstanceState == null)  
 getSupportFragmentManager().beginTransaction().add(R.id.*fragment*, cameraFragment, "camera").commit();  
 else  
 onBackStackChanged();  
 }  
  
 @Override  
 public void onWindowFocusChanged(boolean hasFocus) {  
 super.onWindowFocusChanged(hasFocus);  
 cameraFragment.onWindowFocusChanged();  
 }  
  
 @Override  
 public void onBackStackChanged() {  
 getSupportActionBar().setDisplayHomeAsUpEnabled(getSupportFragmentManager().getBackStackEntryCount()>0);  
 }  
  
 @Override  
 public boolean onSupportNavigateUp() {  
 onBackPressed();  
 return true;  
 }  
  
// @Override  
// public boolean onCreateOptionsMenu(Menu menu) {  
// // Inflate the menu; this adds items to the action bar if it is present.  
// getMenuInflater().inflate(R.layout.menu\_main, menu);  
// mMenu = menu;  
// return true;  
// }  
}

## קוד - fragment\_camera xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/linearLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#a6dfdfdf"  
 android:orientation="vertical"  
 android:weightSum="10">  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="256dp"  
 android:gravity="center|center\_vertical"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@+id/tracking\_overlay"  
 app:layout\_constraintHorizontal\_bias="0.138"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <com.p4f.esp32camai.OverlayView  
 android:id="@+id/tracking\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center\_vertical"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.686"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.109" />  
  
 <io.github.controlwear.virtual.joystick.android.JoystickView  
 android:id="@+id/joystick"  
 android:layout\_width="234dp"  
 android:layout\_height="229dp"  
 app:JV\_backgroundColor="#000000"  
 app:JV\_borderColor="#454545"  
 app:JV\_borderWidth="4dp"  
 app:JV\_buttonColor="#FFFFFF"  
 app:JV\_buttonSizeRatio="15%"  
 app:JV\_fixedCenter="false"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
  
 <com.p4f.esp32camai.VerticalButton  
 android:id="@+id/objDetBtn"  
 android:layout\_width="68dp"  
 android:layout\_height="87dp"  
 android:background="@drawable/my\_button\_bg"  
 android:minHeight="80dip"  
 android:text="Object Detection"  
 android:textColor="#ffffff"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@+id/tracking\_overlay"  
 app:layout\_constraintHorizontal\_bias="0.688"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.981" />  
  
 <com.p4f.esp32camai.VerticalButton  
 android:id="@+id/streamBtn"  
 android:layout\_width="68dp"  
 android:layout\_height="87dp"  
 android:background="@drawable/my\_button\_bg"  
 android:minHeight="80dip"  
 android:text="Stream"  
 android:textColor="#ffffff"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.906"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.838" />  
  
 <com.p4f.esp32camai.VerticalButton  
 android:id="@+id/ledBtn"  
 android:layout\_width="68dp"  
 android:layout\_height="87dp"  
 android:background="@drawable/my\_button\_bg"  
 android:minHeight="80dip"  
 android:text="LED"  
 android:textColor="#ffffff"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.688"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.838" />  
  
<!-- <com.p4f.esp32camai.VerticalButton-->  
<!-- android:id="@+id/shootBtn"-->  
<!-- android:layout\_width="68dp"-->  
<!-- android:layout\_height="87dp"-->  
<!-- android:background="@drawable/my\_button\_bg"-->  
<!-- android:minHeight="80dip"-->  
<!-- android:text="Shoot"-->  
<!-- android:textColor="#ffffff"-->  
<!-- app:layout\_constraintBottom\_toBottomOf="parent"-->  
<!-- app:layout\_constraintEnd\_toEndOf="parent"-->  
<!-- app:layout\_constraintHorizontal\_bias="0.906"-->  
<!-- app:layout\_constraintStart\_toStartOf="parent"-->  
<!-- app:layout\_constraintTop\_toTopOf="parent"-->  
<!-- app:layout\_constraintVertical\_bias="0.981" />-->  
  
 <TextView  
 android:id="@+id/tempView"  
 android:layout\_width="73dp"  
 android:layout\_height="66dp"  
 android:rotation="90"  
 android:text="@string/\_255"  
 android:textColor="@android:color/black"  
 android:textSize="24sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.976"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.021" />  
  
 <ImageButton  
 android:id="@+id/shootBtn"  
 android:layout\_width="68dp"  
 android:layout\_height="87dp"  
 android:contentDescription="Shoot button"  
 android:tooltipText="Shoot"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.906"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.981"  
 app:srcCompat="@android:drawable/ic\_menu\_compass" />  
  
 <com.google.android.material.slider.Slider  
 android:id="@+id/servoSlider"  
 android:layout\_width="304dp"  
 android:layout\_height="48dp"  
 android:layout\_marginEnd="232dp"  
 android:contentDescription="@string/sliderservo"  
 android:rotation="90"  
 android:stepSize="10"  
 android:value="0"  
 android:valueFrom="0"  
 android:valueTo="70"  
 app:haloColor="@android:color/white"  
 app:labelBehavior="floating"  
 app:layout\_constraintBottom\_toBottomOf="@+id/tracking\_overlay"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/tracking\_overlay"  
 app:layout\_constraintVertical\_bias="0.254"  
 app:thumbColor="@color/colorAccent"  
 app:thumbRadius="11dp"  
 app:tickColor="@android:color/white"  
 app:tickVisible="false"  
 app:trackColor="@color/colorAccent"  
 app:trackHeight="9dp" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

## קוד - activity\_main xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <RelativeLayout  
 android:id="@+id/fragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">  
  
 </RelativeLayout>  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

## קוד - Settings gradle

include ':ESP32CamAI'  
rootProject.name='ESP32CamAI'  
include ':openCVLibrary348'

## קוד – colors xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <color name="colorPrimary">#000080</color>  
 <color name="colorPrimaryDark">#228b22</color>  
 <color name="colorAccent">#228b22</color>  
  
 <color name="colorRecieveText">#00FF00</color>  
 <color name="colorSendText">#82CAFF</color>  
 <color name="colorStatusText">#FFDB58</color>  
</resources>

## קוד - strings xml

<resources>  
 <string name="app\_name">ESP32 AI Camera</string>  
 <string name="camera\_permission\_title">Camera permission required</string>  
 <string name="camera\_permission\_message">This app uses camera only for object detection and segmentation, the information from your camera is not sent or collected anywhere else</string>  
 <string name="sliderservo">sliderservo</string>  
 <string name="SliderServo">SliderServo</string>  
 <string name="\_255">255</string>  
  
</resources>

## קוד - styles xml

<resources>  
  
 <!-- Base application theme. -->  
 <style name="AppTheme" parent="Theme.MaterialComponents.NoActionBar">  
 <!-- Customize your theme here. -->  
 <item name="colorPrimary">@color/colorPrimary</item>  
 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>  
 <item name="colorAccent">@color/colorAccent</item>  
 </style>  
  
</resources>

## קוד - arrays xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <string-array name="newline\_names">  
 <item>LF</item>  
 <item>CR+LF</item>  
 </string-array>  
</resources>

# פרק 6 – תקשורת מצלמה ואפליקציה

# פירוט הקלאסים באפליקציה

MainActivity.java: זוהי הפעילות העיקרית של האפליקציה. הוא מכיל את הקוד להגדרת התצוגה של המצלמה, צילום תמונות וטיפול באירועי לחיצה על כפתורים.

שיטת onPictureTaken נקראת בכל פעם שתצלום תמונה חדשה. שיטה זו טוענת את התמונה לאובייקט Bitmap ומעבירה אותה למחלקה ObjectDetector לזיהוי אובייקט.

ObjectDetector.java: מחלקה זו מכילה את הקוד לזיהוי אובייקטים באמצעות TensorFlow Lite. הוא טוען מודל TensorFlow Lite מאומן מראש ומשתמש בו כדי לבצע זיהוי אובייקטים בתמונת הקלט. שיטת detectObjects לוקחת אובייקט Bitmap ומחזירה רשימה של אובייקטי Detection (המייצגים את האובייקטים שזוהו ואת מיקומם בתמונה).

Detection.java: מחלקה זו מייצגת אובייקט שזוהה ומיקומו בתמונה.

## מושגים

TensorFlow Lite

היא גרסה קלת משקל של מסגרת TensorFlow שתוכננה במיוחד עבור מכשירים ניידים ומשובצים. זה מאפשר פריסה יעילה של מודלים של למידת מכונה בפלטפורמות מוגבלות במשאבים כמו סמארטפונים, התקני IoT ומיקרו-בקרים.

להלן כמה היבטים ומאפיינים מרכזיים של TensorFlow Lite:

* אופטימיזציה של מודלים: TensorFlow Lite משתמש בטכניקות שונות כדי לייעל מודלים של למידת מכונה לפריסה במכשירים ניידים ומשובצים. אופטימיזציות אלו כוללות קוונטיזציה, אשר מפחיתה את הדיוק של המשקולות וההפעלה של הדגם, וכתוצאה מכך גדלים קטנים יותר של דגם וזמני מסקנות מהירים יותר.
* המרת מודל: TensorFlow Lite מספק כלים וממשקי API להמרת מודלים של TensorFlow לפורמט המתאים לפריסה בפלטפורמות ניידות ומשובצות. ממיר TensorFlow Lite ממיר דגמים מהפורמט המקורי של TensorFlow (.pb) או ממסגרות אחרות כמו Keras לפורמט TensorFlow Lite (.tflite).
* מנוע הסקה: TensorFlow Lite משלב מנוע הסקה המאפשר ביצוע יעיל של מודלים של למידת מכונה במכשירים ניידים ומשובצים. הוא ממנף האצת חומרה, אם אפשר, כדי להאיץ מסקנות של מודלים ולהשיג ביצוע אחזור נמוך וביצועים גבוהים.
* תמיכה בפלטפורמה: TensorFlow Lite תומך במגוון רחב של פלטפורמות, כולל אנדרואיד, iOS, Linux, מיקרו-בקרים (כגון מכשיר ה-ESP32-CAM) ועוד. זה מאפשר שילוב חלק של מודלים של למידת מכונה לתוך יישומים שונים על פני מכשירים שונים.
* פריסת מודלים: עם TensorFlow Lite, ניתן לפרוס מודלים של למידת מכונה באופן מקומי במכשיר, מה שמאפשר הסקה לא מקוונת ובזמן אמת. זה מאפשר יישומים רגישים לפרטיות, מפחית את ההסתמכות על קישוריות רשת ומשפר את חוויות המשתמש.
* ערכת כלי אופטימיזציה של מודלים: TensorFlow Lite מספק ערכת כלי אופטימיזציה של מודלים המציעה חבילה של כלים וטכניקות לאופטימיזציה ודחיסת מודלים של למידת מכונה. כלים אלו מאפשרים למפתחים לצמצם עוד יותר את גדלי המודל, לשפר ביצועים ולהתאים מודלים לדרישות פריסה ספציפיות.

על ידי שימוש ב-TensorFlow Lite, מפתחים יכולים למנף את הכוח של למידת מכונה במכשירים מוגבלי משאבים, מה שמאפשר מגוון רחב של יישומים, כולל סיווג תמונות, זיהוי אובייקטים, עיבוד שפה טבעית ועוד.

### BitMap

באנדרואיד, Bitmap הוא אובייקט המייצג תמונת Bitmap, שהיא סוג של פורמט תמונה המורכב מרשת מלבנית של פיקסלים. כל פיקסל ברשת מייצג ערך צבע שקובע את הצבע של אותו חלק מסוים בתמונה.

מחלקת Bitmap באנדרואיד מספקת שיטות ליצירה, מניפולציה והצגת תמונות Bitmap. בהקשר של אפליקציית ESP32-CAM, התמונה שנלכדה ממכשיר ה-ESP32-CAM מתקבלת על ידי אפליקציית אנדרואיד דרך Wi-Fi בצורה של נתוני תמונה גולמיים. נתוני תמונה גולמיים אלה מומרים לאובייקט Bitmap באמצעות המחלקה BitmapFactory. המחלקה BitmapFactory מספקת שיטות לפענוח נתוני תמונה לתוך אובייקטי Bitmap.

ברגע שנתוני התמונה הגולמיים מומרים לאובייקט Bitmap, אפליקציית אנדרואיד יכולה לתפעל ולהציג את התמונה על המסך. מחלקת Bitmap מספקת שיטות לתפעול הפיקסלים בתמונה, כגון שינוי הצבע של פיקסלים בודדים, שינוי גודל התמונה והחלת מסננים על התמונה.

בהקשר של אפליקציית ESP32-CAM, אובייקט Bitmap משמש להצגת התמונה שנלכדה על מסך מכשיר האנדרואיד ולביצוע זיהוי אובייקטים באמצעות דגם TensorFlow Lite שהוכשר מראש. המחלקה ObjectDetector לוקחת אובייקט Bitmap כקלט ומבצעת זיהוי אובייקט על נתוני התמונה באמצעות מודל TensorFlow Lite. לאחר שהאובייקטים זוהו, אובייקט Bitmap משמש להצגת התמונה על המסך כשהאובייקטים שזוהו מודגשים.

# איך התמונה נשלחת לאפליקציה?

התמונה מועברת דרך Wi-Fi באמצעות חיבור socket TCP בין מכשיר ה-ESP32-CAM לאפליקציית אנדרואיד. להלן פירוט של איך זה עובד:

* אפליקציית אנדרואיד שולחת בקשה למכשיר ה-ESP32-CAM לצלם תמונה.
* מכשיר ה-ESP32-CAM מצלם תמונה באמצעות המצלמה שלו ושומר אותה במאגר.
* מכשיר ה-ESP32-CAM פותח socket TCP ומאזין לחיבורים נכנסים מאפליקציית אנדרואיד.
* אפליקציית אנדרואיד פותחת socket TCP ומתחברת לכתובת ה-IP ומספר היציאה של מכשיר ה-ESP32-CAM.
* ברגע שחיבור socket TCP נוצר, אפליקציית אנדרואיד שולחת בקשה למכשיר ה-ESP32-CAM לשלוח את התמונה שנלכדה.
* התקן ESP32-CAM שולח את נתוני התמונה דרך חיבור socket ה-TCP בחתיכות.
* אפליקציית אנדרואיד מקבלת את נתוני התמונה דרך חיבור socket TCP ומאחסנת אותם במאגר.
* לאחר שהתקבלו כל נתוני התמונה, אפליקציית אנדרואיד ממירה את נתוני התמונה הגולמיים לאובייקט Bitmap ומציגה אותם על המסך.

בסך הכל, העברת התמונה באמצעות Wi-Fi מיושמת באמצעות חיבורי socket TCP, המספקים דרך אמינה ויעילה להעברת נתונים ברשת. נתוני התמונה נשלחים בחתיכות על חיבור socket ה-TCP כדי להבטיח שניתן להעביר כמויות גדולות של נתונים ללא בעיות.

שקע ה-TCP נפתח על ידי מכשיר ה-ESP32-CAM ואפליקציית אנדרואיד באמצעות מחלקה Socket ב-Java. מחלקת Socket מספקת דרך ליצור חיבור שקע TCP בין שני התקנים ברשת.

בקוד של התקן ESP32-CAM, שקע TCP נפתח על ידי יצירת אובייקט ServerSocket חדש וקריאה לשיטת accept() שלו. האובייקט ServerSocket מאזין לחיבורים נכנסים במספר יציאה שצוין. ברגע שמתקבלת בקשת חיבור, השיטה accept() מחזירה אובייקט Socket חדש המייצג את החיבור. לאחר מכן ניתן להשתמש באובייקט Socket כדי לשלוח ולקבל נתונים דרך החיבור.

# פרק 7 – תיעוד עבודה ורפלקציה

# 

# תיעוד תקלות ותיקון

תקלת ESP32 – בזמן שניסינו לצרוב על הבקר קיבלנו הודעת שגיאה לאחר מכן הבנו שצריך למחוק את הזיכרון פלאש של הבקר וניסינו בעזרת VSCODE אבל זה לא עבד, לבסוף החלפנו לבקר ESP32 חדש.

תקלת מנועי DC וההדקים – בתוכנית PWM\_ctrl יש יציאות של קדימה ואחורה של כל מנוע, פיזית לא ידענו איזה יציאות זה אחורה או קדימה, זה כל פעם חיברנו בדרך אחת וניסינו לראות עם כל הסיביות עוברות כמו שצריך והרכב זז כפי שרצינו, ואם לא פשוט חיברנו את ההדקים בדרך שונה עד שחיברנו להדקים הנכונים.

תקלת חוטים – החוטים שהשתמשנו בהם לפעמים היו רופפים או ממש גרועים עד כדי שקיצרו את המערכת והיינו צריכים להחליף אותם, גם בשביל לחבר את הכניסות של המנועים למסרקים של האלטרה וה – ESP32 היינו צריכים להלחים אותם.

תקלת מתג הריגה – המתג שקיבלנו בשביל להפעיל ולכבות את המערכת היה נורא קטן ,צפוף ושביר לכן כשהלחמנו עליו את הכבלים הנחוצים למערכת הם התנתקו יותר מידי פעמים וזה הגיע למצב שאחד מאיתנו החזיק את הכבלים שיהיו במגע והשני מפעיל את המערכת.

# 

# תיעוד תהליך עבודה

**תאריך 8.9.2022:** יצירת תופס פרויקט למשרד החינוך.

**תאריך 11.9.2022:** סיום סיכום רכיבים ופרוטוקולים.

**תאריך 18.9.2022:** יצירת מעגלים חשמליים בEASYEDA.

**תאריך 20.9.2022:** בניית תכנון היררכי לתקשורת בין אלטרה ל- ESP32 והוספת קוד העברת מידע והתחברות ל- WIFI בesp32 -.

**תאריך 29.9.2022:** יצירת FIREBASE של גוגל בכללי וגם בתוך ה-code Studio Visual- וה Android Studio. כך שנוכל לקשר בין דגם הרובוט ל-FIREBASE ונוכל לשלוט עליו כבר דרך הענן של גוגל והאפליקציה.

**תאריך 8.10.2022:** קבלת אלטרה + Esp32 וחיבור תקשורת ביניהם.

**תאריך 5.12.2022:** קבלת דגם הרכב עם מנועי DC והרכבתו

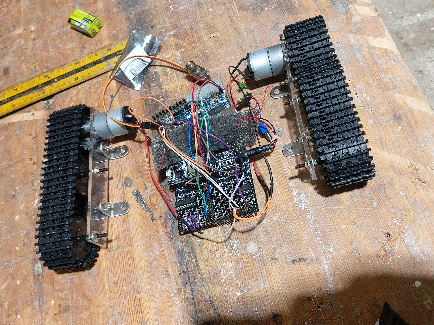
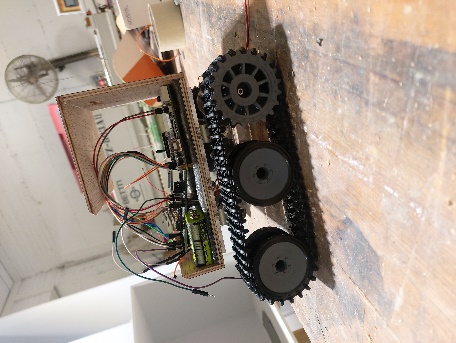
**תאריכים 8.10.2022 – 8.1.2022** בניית תוכניות VHDL בתכנון ההיררכי בשביל הוספת טמפרטורה, מנועים, וסוללה, ובנוסף התחלת בניית האפליקציה ומבנה הרכב.

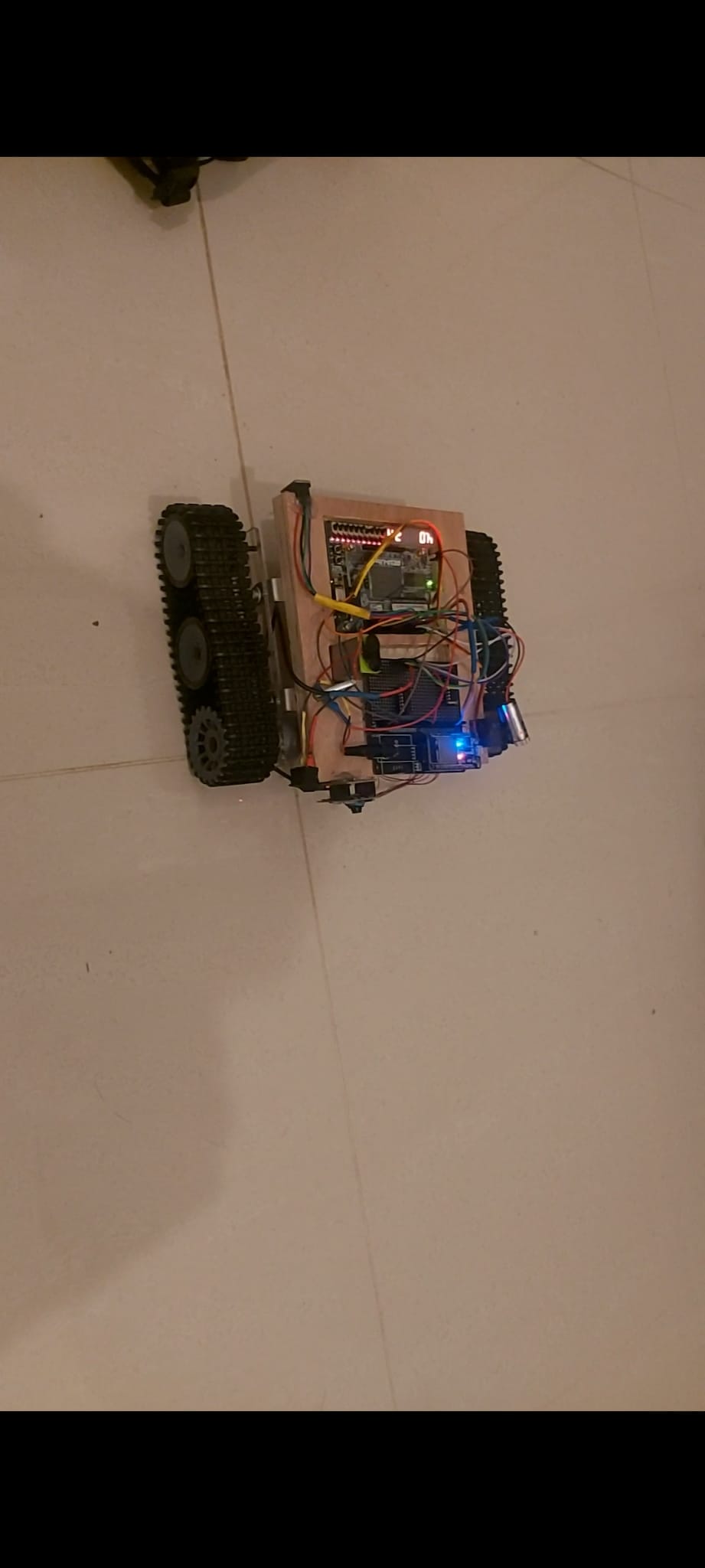
**תאריך 23.1.2022:** הוספת מנוע סרבו עם לייזר מודבק עליו ובניית תוכניות VHDL מתאימות.

**תאריך 15.3.2022:** שינוי קוד הesp32 לקוד שעובד עם הפיירבייס ללא דיליי.

תאריך 15.4 בניית דגם חדש לרעב כי הקודם היה לא יעיל

תאריך 5.1 הרכבת הרכב מאפס רכיב אחר רכיב ובדיקתו כך שלא יקצר את המערכת.

****



****

### בניית הדגם

יצירת פלטפורמה מעץ שמודבק עם דבק וסיכות בשביל הדגם עם ריפוד מבפנים.

חיבור הפלטפורמה מעץ למנועים ובדיקת חלל בשביל אלטרה ובקר ESP.



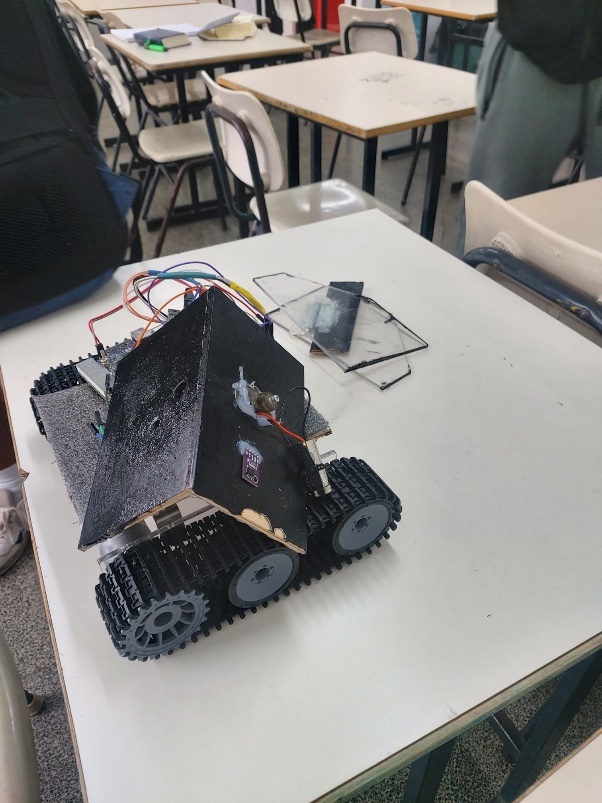


הדבקת קירות מחומר אקרילי שקוף בשביל לראות עיצוב מבפנים וקדיחת חורים בשביל חוטים של המנועים.



סידור כבלים עם שרינקים לפני ואחרי:

צביעת הדגם בשחור והדגם עם האפליקציה:



הדגם לא היה יעיל ונשבר



בנינו אותו שוב עם תאים מובנים לכל רכיב כולל הסוללה. בצד הדבקנו את הסרבו ומהצד השני הדבקנו את המצלמה, שניהם עם דבק חם

# רפלקציה

### עידן

אחד ההיבטים המאתגרים בפרויקט זה הוא בניית הרכיבים המכניים של המכונית. תכנון והרכבת רכב שיכול לנהוג בצורה חלקה וזו משימה מורכבת הדורשת תשומת לב לפרטים ודיוק. בנוסף, חשוב לוודא שהרכיבים החשמליים והאלקטרוניים של המכונית משולבים כראוי ומוגנים מפני נזק.

היבט מאתגר נוסף הוא תכנות המכונית לשליחת נתונים לאפליקציה. זה דורש הבנה טובה של שפות קוד ויכולת לעבוד עם חיישנים, מנועים ורכיבים אלקטרוניים אחרים. הנתונים המועברים מהמכונית לאפליקציה עשויים לכלול מידע על טמפרטורת הסביבה ומצלמה שניתן לראות איתה. לוודא שהנתונים האלה מדויקים ומועברים בצורה מהימנה יכולה להיות משימה לא פשוטה.

בסך הכל, עבודה על פרויקט אלקטרוניקה כמו בניית מכונית שיכולה לנהוג ולשלוח נתונים לאפליקציה יכולה להיות מאתגרת ומתגמלת כאחד. הפרויקט דורש שילוב של מיומנויות מכניות, אלקטרוניות ותכנות, והשלמה מוצלחת יכולה לספק תחושת סיפוק עמוקה. עם מסירות, התמדה ותשומת לב לפרטים, כל אחד יכול לקחת על עצמו פרויקט כזה ולהצליח.

### אור-ים

תהליך יצור הפרויקט היה מהנה ברובו, נהניתי לעשות עבודת צוות עם השותף, החלק שהכי זרם לנו היה העבודה הפיזית ולא התיעוד, אם הייתי משנה משהו בפרויקט הזה זה רק התכנון שלו חוץ מזה הכל היה כיף, מהיר וחוויתי.

# ביבליוגרפיה

### דוחף זרם – :L293D

<https://pdf1.alldatasheet.com/datasheet-pdf/view/89353/TI/L293D.html>

### חיישן טמפרטורה – :18B20

<https://pdf1.alldatasheet.com/datasheet-pdf/view/58559/DALLAS/18B20.html>

### מנוע DC – :HC385MG-301

<https://disti-assets.s3.amazonaws.com/testco-inc/files/datasheets/25281.pdf>

### בקר – :ESP32

<https://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>

### :ESP32 – CAM

<https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf>

https://nugroho.xyz/mendeteksi-sentuhan-dengan-esp32-cam/

### L293D הדקים ומידע:

<https://components101.com/ics/l293d-pinout-features-datasheet>

### מידע על הדקים ופעולות :ESP32

<https://www.circuitschools.com/what-is-esp32-how-it-works-and-what-you-can-do-with-esp32/#Peripheral_Features>

### חיישן טמפרטורה – :LM75

[https://datashee ts.m aximintegrated.com/en/ds/LM75.pdf](https://datasheets.maximintegrated.com/en/ds/LM75.pdf)

### דוחף זרם – :ULN2803

<https://www.alldatasheet.com/datasheet-pdf/pdf/12687/ONSEMI/ULN2803.html>

### תקלת עיבוד תמונה:

<https://stackoverflow.com/questions/67448987/indexerror-list-index-out-of-range-object-detection>

### קוד עיבוד תמונה

<https://github.com/longpth/ESP32CamAI>

### מנוע סרבו

<https://components101.com/motors/mg90s-metal-gear-servo-motor>

<https://www.electronicoscaldas.com/datasheet/MG90S_Tower-Pro.pdf>