פרויקט גמר י"ב

רובוט משדר תמונה ונתונים לאחור בטכנולוגית IOT

דרכא בית ירח

אור-ים ביבי רובי, עידן קורז'ק

[הצעת נושא 4](#_Toc122130495)

[תרשים מלבנים סופי 5](#_Toc122130496)

[מערכת כללית 8](#_Toc122130497)

[ורכיבי פרויקט וחומרה 9](#_Toc122130498)

[דוחף זרםL293d – 9](#_Toc122130499)

[עיקרון הפעולה 10](#_Toc122130500)

[מבנה פנימי של הרכיב 10](#_Toc122130501)

[מנוע Direct Current Motor - DC- 11](#_Toc122130502)

[PWM 13](#_Toc122130503)

[חיישן טמפרטורה – 18b20 15](#_Toc122130504)

[בקר – ESP32 17](#_Toc122130505)

[עקרון פעולה 19](#_Toc122130506)

[מבנה פנימי 19](#_Toc122130507)

[פרוטוקולים- ESP32 21](#_Toc122130508)

[ESP32 - CAM 23](#_Toc122130509)

[עקרון פעולה 23](#_Toc122130510)

[Altera 24](#_Toc122130511)

[FPGA-Field-Programmable Gate Array 24](#_Toc122130512)

[עקרון פעולה 25](#_Toc122130513)

[מבנה פנימי 25](#_Toc122130514)

[תקן תקשורת RS232 25](#_Toc122130515)

[LM75 – Digital Temperature Sensor 28](#_Toc122130516)

[Firebase – Google 31](#_Toc122130517)

[דוחף זרם – uln2803 31](#_Toc122130518)

[מבנה פנימי 32](#_Toc122130519)

[נורת להט 33](#_Toc122130520)

[מנוע Servo – MG90S 34](#_Toc122130521)

[מעגלים חשמליים 37](#_Toc122130522)

[DC Motor to L293D 37](#_Toc122130523)

[ALTERA to ESP – TX to RX 38](#_Toc122130524)

[ESP32 – CAM to Firebase to Java App 39](#_Toc122130525)

[LM75 TEMP SENSOR to ALTERA And ESP32 40](#_Toc122130526)

[תוכניות VHDL 42](#_Toc122130527)

[PIN PLANNER 68](#_Toc122130528)

[שפת C 70](#_Toc122130529)

[קוד בקר ה – ESP32 72](#_Toc122130530)

[קוד ESP32 – CAM 75](#_Toc122130531)

[שפת תוויות XML 80](#_Toc122130532)

[Python שפת 81](#_Toc122130533)

[ESP32 – CAM קוד עיבוד תמונה של 81](#_Toc122130534)

[האפליקציה 84](#_Toc122130535)

[קוד - main activity 85](#_Toc122130536)

[קוד – Servo Controller 87](#_Toc122130537)

[קוד - activity\_main xml 88](#_Toc122130538)

[קוד - activity\_servo\_controller xml 89](#_Toc122130539)

[קוד - Settings gradle 90](#_Toc122130540)

[קוד – colors xml 90](#_Toc122130541)

[קוד - strings xml 91](#_Toc122130542)

[קוד - themes xml 91](#_Toc122130543)

[קוד - themes xml (night) 91](#_Toc122130544)

[תיעוד תקלות ותיקון 96](#_Toc122130545)

[אבולוציית הפרויקט 97](#_Toc122130546)

[ביבליוגרפיה 100](#_Toc122130547)

**סמל מוסד - 261065**

**עבודת גמר**

**פרוייקט בהיקף של 5 יחידות**

**הנדסת חשמל ואלקטרוניקה**

**בהתמחות:** מחשוב ובקרה.

**הנושא:** רובוט משדר תמונה ונתונים לאחור בטכנולוגית IOT.

**מנחה הפרויקט:** חברבר אייל.

**מגישים:**

|  |  |
| --- | --- |
| **שם** | **תעודת זהות** |
| אור-ים ביבי רובי | 328412261 |
| עידן קורזיק | 21567059 |

**תאריך הגשה:** תשפ"ג

# 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 18b20 | ESP32 בקר להעברת מידע ברשת | FPGA MAX10 ALTERA |

# תרשים מלבנים סופי

**Main\_clk**

**50MHz**

My\_GND

**Spliter**

**Tx**

**מידע טורי בתקן**

**RS232**

clk

**Pmod\_temp\_sensor\_tcn75a**

scl

Scl\_lm75

sda

Scl\_lm75

i2c\_ack\_err

Erorr\_dp

9

**מידע מקבילי**

**Bit\_9\_to\_8**

7

8

**מידע מקבילי**

**Rx**

אפליקציה בשפת - Java

מידע טורי שמשמעותו פעולה ברכב (אדום)

**Rx**

clk

si



כניסה 1 מנוע A

כניסה 2 מנוע A

כניסה 1 מנוע B

כניסה 2 מנוע B

**BOUD\_RATE**

מחלק תדר

153,600MHz

clk

**Led**

clk

Ledout

**PWM\_Motor**

clk

pOut

8

מידע מקבילי מהאפליקציה

**Data\_Spliter**

Motors [3..0]

Led [4..3]

TBA [7..4]

**Fb2bin**

clk

Number[3..0]

Ma0

Ma1

Mb0

Mb1

**Motor\_Ctrl**

Ma0

Ma1

Mb0

Mb1

PWM\_Motor

LwheelB

LwheelF

RwheelB

RwheelF

LwheelB

LwheelF

RwheelB

RwheelF

BreatheLED

מחובר ל – 7 seg DP

7

7

**Low**

**high**

**חיווי לתהליך פיתוח**

data

**C.A**

data

Led [4..3]

LED\_CAR

מחובר ל – ESP32

Led\_car

Lout

מחובר פיזית ל – DE10 אדמה ו – Vcc מחובר ל – ESP32

Din[7..0]

reset\_n

clk

Temperature[resolution-1..0]

Tout[7..0]

Temperature[8..0]

BOUD\_RATE – מחלק תדר ל –153,600MHz.

Led – מוודא שהמערכת עובדת בכך שנדלק כל שניה.

PWM\_Motor – יוצר תדר של 10KHz ב D.C של 80% בשביל מנועי DC.

Data\_Spliter – מחלק את הסיביות של הקולט (RX) לפעולות שונות (מנועים, לד).

Fb2bin – קולט את המידע מהFIREBASE ומעביר אותו לבינארי אינטג'ר.

Motor\_Ctrl – מקבל את הסיביות מה - Fb2bin ומחלק אותם לכניסות של המנועים (קדימה אחורה) ועובד לפי תדר ה - PWM\_Motor.

Rx – מקבל שעון מה - BOUD\_RATE ומידע בגודל של 8 סיביות מהמשדר (TX) של ה – ESP32.

Tx – מקבל מידע מקבילי בגודל 8 סיביות ומשדר אותו בתקן RS232 בטורי, מקבל את אותו השעון כמו הקולט לסנכרון.

Spliter – מחלק את הסיביות מידע מקבילי ל0..3 נמוך ו 4..7 גבוה בשביל מפענח תצוגה.

Pmod\_temp\_sensor\_tcn75a – תוכנית שעובדת על חיישן טמפרטורה LM75 ומוציאה את המידע הנחוץ לחיישן (scl,sda).

Bit\_9\_to\_8 – מעביר את המידע של החיישן טמפרטורה מ - 9 סיביות ל – 8.

# מערכת כללית

הפרויקט שלנו הינו רכב חכם המשדר נתונים לאחור בעזרת IOT לטובת איסוף מידע בשטח ומקומות שכוח אדם אינו יכול לתפעל.

# 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.

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.

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

מצד שני, כדי לשלוט בכיוון הסיבוב, אנחנו רק צריכים להפוך את כיוון זרימת הזרם דרך המנוע, והשיטה הנפוצה ביותר לעשות זאת היא באמצעות H-Bridge.

מעגל H-Bridge מכיל ארבעה רכיבי מיתוג, טרנזיסטורים או MOSFET, כאשר המנוע במרכז יוצר תצורה דמוית H. על ידי הפעלת שני מתגים מסוימים בו זמנית נוכל לשנות את כיוון זרימת הזרם, ובכך לשנות את כיוון הסיבוב של המנוע.

אז אם נשלב את שתי השיטות הללו, ה-PWM וה-H-Bridge, נוכל לקבל שליטה מלאה על מנוע ה-DC.

### 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.

## DS18B20-One Wire Digital Temperature Sensorחיישן טמפרטורה – 18b20

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'.

### I2C BUS

I2C הוא פרוטוקול תקשורת טורי, סינכרוני, חצי דופלקס המאפשר קיום משותף של מספר מאסטרים ועבדים על אותו BUS. BUS I2C מורכב משני קווים: קו נתונים טורי (SDA) ושעון טורי (SCL). שני הקווים דורשים נגדי משיכה.

עם יתרונות כמו פשטות ועלות ייצור נמוכה, I2C משמש בעיקר לתקשורת של התקנים היקפיים במהירות נמוכה למרחקים קצרים (בתוך רגל אחת).

ל-ESP32 יש 2 בקרי I2C (המכונה גם יציאה), האחראי לטיפול בתקשורת בBUS- I2C. בקר I2C יחיד יכול לפעול כמאסטר או כעבד.

## 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 הוא פרוטוקול תקשורת טורי, סינכרוני, חצי דופלקס המאפשר קיום משותף של מספר מאסטרים ועבדים על אותו BUS. BUS I2C מורכב משני קווים: קו נתונים טורי (SDA) ושעון טורי (SCL). שני הקווים דורשים נגדי משיכה.

עם יתרונות כמו פשטות ועלות ייצור נמוכה, I2C משמש בעיקר לתקשורת של התקנים היקפיים במהירות נמוכה למרחקים קצרים (בתוך רגל אחת).

## 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

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.

## 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°.

# מעגלים חשמליים

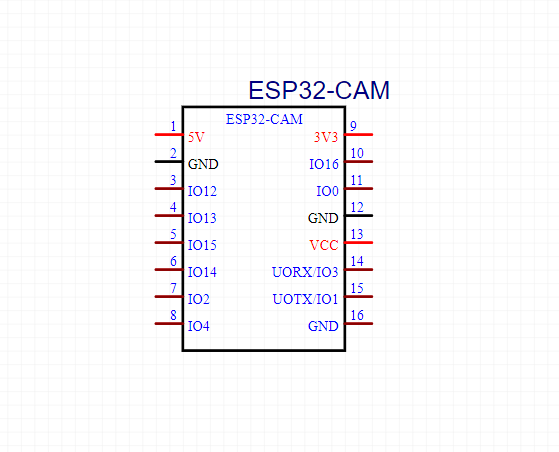
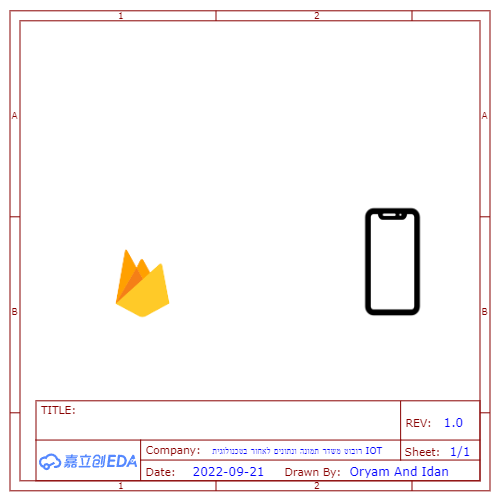
DC Motor to L293D

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.

## ALTERA to ESP – TX to RX

חיבור Tx & Rx בין DE10 ל – ESP32

## ESP32 – CAM to Firebase to Java App



עוד לא סוכם.

## LM75 TEMP SENSOR to ALTERA And ESP32

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

# תוכניות 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.

### משדרTx – Transmitter –

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

### עקרון כתיבת – Tx

10 סיביות x16

מחזורי מחזור של כל סיבית

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.

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:



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;

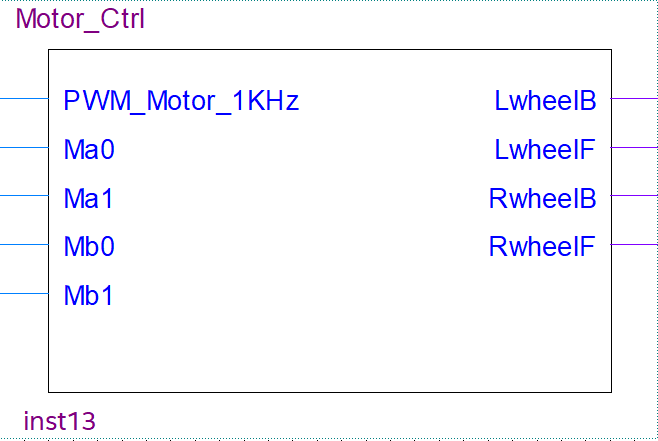
end RTL;



### PWM\_Ctrl

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

סמל הרכיב:



דיאגרמה:



קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity Motor\_Ctrl is

port(PWM\_Motor\_1KHz: in bit;

Ma0: in bit;

Ma1: in bit;

Mb0: in bit;

Mb1: in bit;

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,Ma0,Ma1,Mb0,Mb1)

begin

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

if Ma0 = '0' then if Ma1 = '0' then if Mb0 = '0' then if Mb1 = '0' then LwheelF <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '0' then if Mb0 = '0' then if Mb1 = '0' then LwheelB <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '0' then if Mb0 = '0' then if Mb1 = '0' then RwheelF <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '0' then if Mb0 = '0' then if Mb1 = '0' then RwheelB <= '0';

end if;

end if;

end if;

end if;

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

if Ma0 = '0' then if Ma1 = '1' then if Mb0 = '0' then if Mb1 = '1' then LwheelF <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '1' then if Mb0 = '0' then if Mb1 = '1' then RwheelF <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '1' then if Mb0 = '0' then if Mb1 = '1' then LwheelB <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '1' then if Mb0 = '0' then if Mb1 = '1' then RwheelB <= '0';

end if;

end if;

end if;

end if;

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

if Ma0 = '1' then if Ma1 = '0' then if Mb0 = '1' then if Mb1 = '0' then RwheelB <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '0' then if Mb0 = '1' then if Mb1 = '0' then LwheelB <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '0' then if Mb0 = '1' then if Mb1 = '0' then RwheelF <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '0' then if Mb0 = '1' then if Mb1 = '0' then LwheelF <= '0';

end if;

end if;

end if;

end if;

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

if Ma0 = '0' then if Ma1 = '1' then if Mb0 = '1' then if Mb1 = '0' then LwheelF <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '1' then if Mb0 = '1' then if Mb1 = '0' then RwheelF <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '0' then if Ma1 = '1' then if Mb0 = '1' then if Mb1 = '0' then RwheelB <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

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

if Ma0 = '1' then if Ma1 = '0' then if Mb0 = '0' then if Mb1 = '1' then RwheelF <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '0' then if Mb0 = '0' then if Mb1 = '1' then LwheelF <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '0' then if Mb0 = '0' then if Mb1 = '1' then LwheelB <= PWM\_Motor\_1KHz;

end if;

end if;

end if;

end if;

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

if Ma0 = '1' then if Ma1 = '1' then if Mb0 = '1' then if Mb1 = '1' then LwheelF <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '1' then if Mb0 = '1' then if Mb1 = '1' then LwheelB <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '1' then if Mb0 = '1' then if Mb1 = '1' then RwheelF <= '0';

end if;

end if;

end if;

end if;

if Ma0 = '1' then if Ma1 = '1' then if Mb0 = '1' then if Mb1 = '1' then RwheelB <= '0';

end if;

end if;

end if;

end if;

end process;

end behave;

### פיירבייס לבינארי FB2Bin –

תפקיד התוכנית - לחבר בין המידע המקבילי של הפיירבייס למידע שנכנס לתוכנית PWM\_Ctrl

סמל הרכיב:





קוד התוכנית:

library ieee;

use ieee.std\_logic\_1164.all;

entity fb2bin is

port ( CLK: IN BIT;

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

Ma0: buffer bit;

Ma1: buffer bit;

Mb0: buffer bit;

Mb1: buffer bit);

end;

architecture behave of fb2bin is ד

SIGNAL DATA : bit\_vector (7 downto 0);

begin

PROCESS (CLK)

BEGIN

IF CLK'EVENT AND CLK='1' THEN

DATA<=NUMBER;

END IF;

END PROCESS;

process (CLK)

begin

IF CLK'EVENT AND CLK='1' THEN

IF DATA="00000000" THEN

MA0<='0'; MA1<='0'; MB0<='0'; MB1<='0';

ELSIF DATA="00000001" THEN

MA0<='0'; MA1<='1'; MB0<='0'; MB1<='1';

ELSIF DATA="00000010" THEN

MA0<='1'; MA1<='0'; MB0<='1'; MB1<='0';

ELSIF DATA="00000011" THEN

MA0<='1'; MA1<='0'; MB0<='0'; MB1<='1';

ELSIF DATA="00000100" THEN

MA0<='0'; MA1<='1'; MB0<='1'; MB1<='0';

END IF;

END IF;

end process;

end behave;

### 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:



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;



## PIN PLANNER

Pin Planner מספק תצוגת חבילה גרפית, המאפשרת לך לאמת הקצאות קלט/פלט על ידי ביצוע בדיקות חוקיות על פיני הקלט/פלט של עיצוב והלוגיקה שמסביב.

עם מתכנן ה-Pin, אתה יכול לזהות בנקי קלט/פלט, קבוצות VREF וצמדי סיכות דיפרנציאליים כדי לעזור לך בתהליך תכנון הקלט/פלט.

# שפת C

היא שפת תכנות מחשב לשימוש כללי. הוא נוצר בשנות ה-70 על ידי דניס ריצ'י, ונשאר בשימוש נרחב ובעל השפעה. לפי התכנון, התכונות של C משקפות בצורה נקייה את היכולות של המעבדים הממוקדים. זה מצא שימוש מתמשך במערכות הפעלה, מנהלי התקנים, ערימות פרוטוקולים, אם כי הולך ופוחת עבור תוכנות יישומים. C משמש בדרך כלל בארכיטקטורות מחשבים הנעות ממחשבי העל הגדולים ביותר ועד למיקרו-בקרים הקטנים ביותר ולמערכות משובצות.

יורש של שפת התכנות B, C פותחה במקור ב-Bell Labs על ידי ריצ'י בין 1972 ל-1973 כדי לבנות כלי עזר הפועלים על יוניקס. הוא הוחל ליישום מחדש של הליבה של מערכת ההפעלה יוניקס. במהלך שנות ה-80, C צברה פופולריות בהדרגה. היא הפכה לאחת משפות התכנות הנפוצות ביותר, כאשר מהדרים C זמינים כמעט לכל ארכיטקטורות המחשב ומערכות ההפעלה המודרניות. C תוקן על ידי ANSI מאז 1989 (ANSI C) ועל ידי ארגון התקינה הבינלאומי (ISO).

C היא שפה פרוצדורלית הכרחית התומכת בתכנות מובנה, היקף משתנה לקסיקלי ורקורסיה, עם מערכת סטטית. הוא תוכנן להידור כדי לספק גישה ברמה נמוכה לזיכרון ולמבנה שפה הממפה ביעילות להוראות מכונה, והכל עם תמיכה מינימלית בזמן ריצה. למרות היכולות הנמוכות שלה, השפה תוכננה כדי לעודד תכנות בין פלטפורמות. ניתן להרכיב תוכנית C תואמת תקנים שנכתבה תוך מחשבה על ניידות עבור מגוון רחב של פלטפורמות מחשב ומערכות הפעלה עם מעט שינויים בקוד המקור שלה.

שפת C מציגה גם את המאפיינים הבאים:

* לשפה יש מספר קטן וקבוע של מילות מפתח, כולל קבוצה מלאה של פרימיטיביות של זרימת בקרה: if/else, for, do/while, while ו- switch. שמות המוגדרים על ידי משתמש אינם מובחנים ממילות מפתח בשום סוג של סימן.
* יש לו מספר רב של אופרטורים אריתמטיים, סיביים ולוגיים: +,+=,++,&,|| וכו'.
* ניתן לבצע יותר ממטלה אחת בהצהרה אחת.
* פונקציות:
* ניתן להתעלם מערכי החזרת פונקציות, כאשר אין צורך.
* מצביעי פונקציות ונתונים מאפשרים פולימורפיזם בזמן ריצה אד-הוק.
* לא ניתן להגדיר פונקציות בטווח המילוני של פונקציות אחרות.
* ניתן להגדיר משתנים בתוך פונקציה, עם היקף.
* פונקציה עשויה לקרוא לעצמה, ולכן רקורסיה נתמכת.
* הקלדת הנתונים היא סטטית, אך נאכפת בצורה חלשה; לכל הנתונים יש סוג, אבל המרות מרומזות אפשריות.
* ניתן להגדיר על ידי משתמש (typedef) וסוגים מורכבים.
* סוגי נתונים מצטברים הטרוגניים (struct) מאפשרים לגשת לרכיבי נתונים קשורים ולהקצות אותם כיחידה.
* איחוד הוא מבנה עם חברים חופפים; רק החבר האחרון המאוחסן חוקי.
* אינדקס מערך הוא סימון משני, המוגדר במונחים של אריתמטיקה מצביע. בניגוד למבנים, מערכים אינם אובייקטים ממדרגה ראשונה: לא ניתן להקצות אותם או להשוות אותם באמצעות אופרטורים מובנים בודדים. אין מילת מפתח "מערך" בשימוש או בהגדרה; במקום זאת, סוגריים מרובעים מציינים מערכים באופן תחבירי, למשל חודש.
* סוגים מסופרים אפשריים עם מילת המפתח enum. הם ניתנים להמרה הדדית באופן חופשי עם מספרים שלמים.
* מחרוזות אינן סוג נתונים מובחן, אלא מיושמות באופן קונבנציונלי כמערכי תווים עם סיומת אפס.
* גישה ברמה נמוכה לזיכרון המחשב אפשרית על ידי המרת כתובות מכונה למצביעים.
* נהלים (תתי שגרות שאינן מחזירות ערכים) הם מקרה מיוחד של פונקציה, עם ריק מסוג החזרה לא מודפס.
* ניתן להקצות זיכרון לתוכנית עם שיחות לשגרה לספרייה.
* מעבד קדם מבצע הגדרת מאקרו, הכללת קובץ קוד מקור והידור מותנה.
* ישנה צורה בסיסית של מודולריות: ניתן להרכיב קבצים בנפרד ולקשר אותם יחד, עם שליטה באילו פונקציות ואובייקטי נתונים גלויים לקבצים אחרים באמצעות תכונות סטטיות וחיצוניות.
* פונקציונליות מורכבת כגון קלט/פלט, מניפולציה של מחרוזות ופונקציות מתמטיות מואצלות באופן עקבי לשגרות של ספרייה.
* לקוד שנוצר לאחר הקומפילציה יש צרכים פשוטים יחסית בפלטפורמה הבסיסית, מה שהופך אותו למתאים ליצירת מערכות הפעלה ולשימוש במערכות משובצות.

שפות רבות מאוחרות יותר שאלו ישירות או בעקיפין מ-C, כולל C++, C#, מעטפת C של Unix, D, Go, Java, JavaScript (כולל טרנספילרים), Julia, Limbo, LPC, Objective-C, Perl, PHP, Python, Ruby, Rust, Swift, Verilog ו-SystemVerilog (שפות תיאור חומרה). שפות אלו שאבו רבים ממבני השליטה שלהן ותכונות בסיסיות אחרות מ-C. רובן (פייתון הוא חריג דרמטי) מבטאות גם תחביר דומה מאוד ל-C, והן נוטות לשלב את הביטוי וההצהרה המוכרים של C עם הסוג הבסיסי. מערכות, מודלים של נתונים וסמנטיקה שיכולים להיות שונים בתכלית.

## קוד בקר ה – ESP32

#include <Arduino.h>

#include <WiFi.h>

#include <FirebaseESP32.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"

// What WIFI we are using to send and receive.

#define WIFI\_SSID "ORYAM"

#define WIFI\_PASSWORD "10203040"

// legs of ESP that we receive and transmit

#define RXD2 16

#define TXD2 17

#define ledpin 2

// 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;

/\*  ------------- functions declaration---------------------\*/

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);

  //defines frequency of reading and writing from ALTERA

  WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

    // Connects to wifi

  Serial.print("Connecting to Wi-Fi");

  // while WIFI is not connected print "." every 300ms

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

  {

    Serial.print(".");

    delay(300);

  }

  Serial.println("Connected with IP: ");

  Serial.print(WiFi.localIP());

  //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.setInt(fbdo, "messages/temperature", temperature);

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

      if (Firebase.getInt(fbdo, "messages/motors"))

      {

      // motors in here is the motors on the Firebase database

       motors = fbdo.intData();

      }

      if (Firebase.getInt(fbdo, "messages/leds"))

      {

        leds = fbdo.intData();

      }

      if (Firebase.getInt(fbdo, "messages/ServoMotors"))

      {

        ServoMotors = 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())

  {

    //packet is the 8 bit\_vector when sending data, the first 4 bits are for the motors, the fifth is for the leds, and the last 3 are for the Servo motors

    packet = leds \* 16 + motors + ServoMotors;

    //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 "esp\_camera.h"

#include <WiFi.h>

#define CAMERA\_MODEL\_AI\_THINKER // Has PSRAM

#include "camera\_pins.h"

// ===========================

// Enter your WiFi credentials

// ===========================

const char\* ssid = ORYAM

const char\* password = 12345678

void startCameraServer();

void setup() {

  Serial.begin(115200);

  Serial.setDebugOutput(true);

  Serial.println();

  camera\_config\_t config;

  config.ledc\_channel = LEDC\_CHANNEL\_0;

  config.ledc\_timer = LEDC\_TIMER\_0;

  config.pin\_d0 = Y2\_GPIO\_NUM;

  config.pin\_d1 = Y3\_GPIO\_NUM;

  config.pin\_d2 = Y4\_GPIO\_NUM;

  config.pin\_d3 = Y5\_GPIO\_NUM;

  config.pin\_d4 = Y6\_GPIO\_NUM;

  config.pin\_d5 = Y7\_GPIO\_NUM;

  config.pin\_d6 = Y8\_GPIO\_NUM;

  config.pin\_d7 = Y9\_GPIO\_NUM;

  config.pin\_xclk = XCLK\_GPIO\_NUM;

  config.pin\_pclk = PCLK\_GPIO\_NUM;

  config.pin\_vsync = VSYNC\_GPIO\_NUM;

  config.pin\_href = HREF\_GPIO\_NUM;

  config.pin\_sscb\_sda = SIOD\_GPIO\_NUM;

  config.pin\_sscb\_scl = SIOC\_GPIO\_NUM;

  config.pin\_pwdn = PWDN\_GPIO\_NUM;

  config.pin\_reset = RESET\_GPIO\_NUM;

  config.xclk\_freq\_hz = 20000000;

  config.frame\_size = FRAMESIZE\_UXGA;

  config.pixel\_format = PIXFORMAT\_JPEG; // for streaming

  config.grab\_mode = CAMERA\_GRAB\_WHEN\_EMPTY;

  config.fb\_location = CAMERA\_FB\_IN\_PSRAM;

  config.jpeg\_quality = 12;

  config.fb\_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality

  //                      for larger pre-allocated frame buffer.

  if(config.pixel\_format == PIXFORMAT\_JPEG){

    if(psramFound()){

      config.jpeg\_quality = 10;

      config.fb\_count = 2;

      config.grab\_mode = CAMERA\_GRAB\_LATEST;

    } else {

      // Limit the frame size when PSRAM is not available

      config.frame\_size = FRAMESIZE\_SVGA;

      config.fb\_location = CAMERA\_FB\_IN\_DRAM;

    }

  } else {

    // Best option for face detection/recognition

    config.frame\_size = FRAMESIZE\_240X240;

#if CONFIG\_IDF\_TARGET\_ESP32S3

    config.fb\_count = 2;

#endif

  }

#if defined(CAMERA\_MODEL\_ESP\_EYE)

  pinMode(13, INPUT\_PULLUP);

  pinMode(14, INPUT\_PULLUP);

#endif

  // camera init

  esp\_err\_t err = esp\_camera\_init(&config);

  if (err != ESP\_OK) {

    Serial.printf("Camera init failed with error 0x%x", err);

    return;

  }

  sensor\_t \* s = esp\_camera\_sensor\_get();

  // initial sensors are flipped vertically and colors are a bit saturated

  if (s->id.PID == OV3660\_PID) {

    s->set\_vflip(s, 1); // flip it back

    s->set\_brightness(s, 1); // up the brightness just a bit

    s->set\_saturation(s, -2); // lower the saturation

  }

  // drop down frame size for higher initial frame rate

  if(config.pixel\_format == PIXFORMAT\_JPEG){

    s->set\_framesize(s, FRAMESIZE\_QVGA);

  }

#if defined(CAMERA\_MODEL\_M5STACK\_WIDE) || defined(CAMERA\_MODEL\_M5STACK\_ESP32CAM)

  s->set\_vflip(s, 1);

  s->set\_hmirror(s, 1);

#endif

#if defined(CAMERA\_MODEL\_ESP32S3\_EYE)

  s->set\_vflip(s, 1);

#endif

  WiFi.begin(ssid, password);

  WiFi.setSleep(false);

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

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");

  Serial.print(WiFi.localIP());

  Serial.println("' to connect");

}

void loop() {

  // Do nothing. Everything is done in another task by the web server

  delay(10000);

}

### 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.

# 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.

# Python שפת

Python היא שפת תכנות פופולרית ברמה גבוהה הידועה בפשטות, בקריאות ובגמישות שלה. זוהי שפה מפורשת, כלומר היא מבוצעת על ידי מתורגמן במקום להיות קומפילציה לקוד מכונה שיכול לפעול על מחשב. זה הופך אותה לשפה אידיאלית למתחילים ללמוד, כמו גם למתכנתים מנוסים שרוצים ליצור אבטיפוס במהירות ולפתח יישומים.

לפייתון יש קהילה גדולה ופעילה של משתמשים ומפתחים, והיא נמצאת בשימוש במגוון רחב של תחומים, לרבות מחשוב מדעי, ניתוח נתונים, בינה מלאכותית ופיתוח אתרים. יש לו ספרייה סטנדרטית גדולה המספקת תמיכה למשימות תכנות נפוצות רבות, ויש גם ספריות רבות של צד שלישי זמינות המרחיבות את יכולותיה.

אחת התכונות המרכזיות של Python היא התמיכה שלה בתכנות מונחה עצמים, שהיא פרדיגמת תכנות המאפשרת יצירת קוד לשימוש חוזר באמצעות שימוש במחלקות ואובייקטים. לפייתון יש גם תמיכה חזקה בתכנות פונקציונלי, שהיא פרדיגמת תכנות המתמקדת בשימוש בפונקציות כדי לפעול על נתונים.

בנוסף לשימוש בתכנות למטרות כלליות, Python נמצא בשימוש נרחב גם במחשוב מדעי וניתוח נתונים. יש לו מספר ספריות, כגון NumPy ו-Pandas, שתוכננו במיוחד עבור משימות אלו ומקלות על העבודה עם מערכי נתונים גדולים ולבצע חישובים מורכבים.

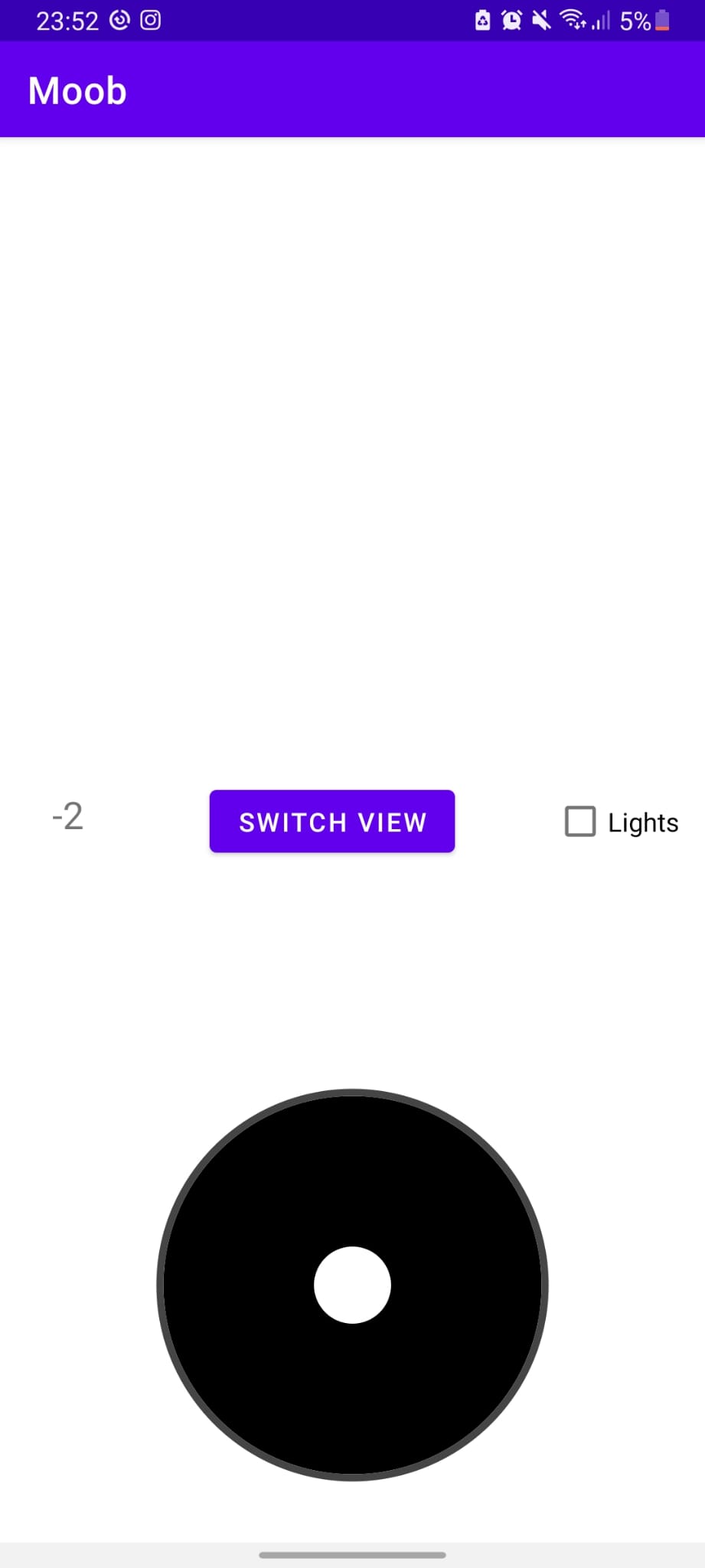
Python היא שפה רב-תכליתית וניתן להשתמש בה למגוון רחב של משימות, כולל פיתוח אתרים, יישומי שולחן עבודה ואפילו פיתוח אפליקציות לנייד. יש לו תחביר פשוט וספרייה סטנדרטית גדולה, מה שמקל על הלמידה והשימוש, והתמיכה החזקה שלו בתכנות מונחה עצמים ופונקציונלי הופכת אותו לכלי רב עוצמה לבניית קוד מורכב וניתן לשימוש חוזר.

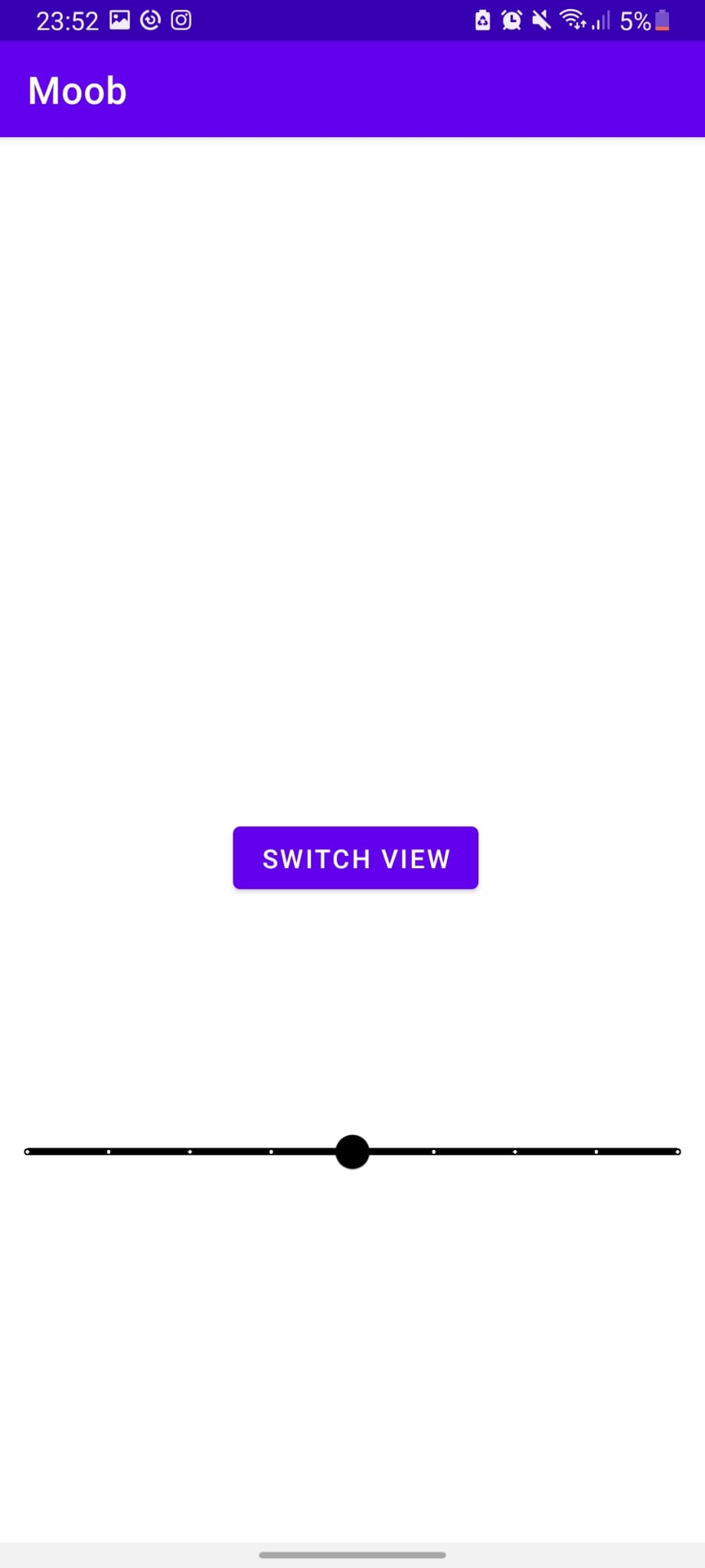
## ESP32 – CAM קוד עיבוד תמונה של

import cv2  
thres = 0.6# Threshold to detect object  
  
classNames = []  
classFile = "coco.names"  
with open(classFile,'rt') as f:  
 classNames=[line.rstrip() for line in f]  
  
configPath = "ssd\_mobilenet\_v3\_large\_coco\_2020\_01\_14.pbtxt"  
weightsPath = "frozen\_inference\_graph.pb"  
  
net = cv2.dnn\_DetectionModel(weightsPath, configPath)  
net.setInputSize(320, 240)  
net.setInputScale(1.0 / 127.5)  
net.setInputMean((127.5, 127.5, 127.5))  
net.setInputSwapRB(True)  
  
  
  
def getObjects(img, draw=True,objects=[]):  
  
 classIds, confs, bbox = net.detect(img, confThreshold=thres, nmsThreshold=0.3)  
 #print(classIds, bbox)  
  
 objectInfo=[]  
 if(len(objects) == 0): objects = classNames  
 if len(classIds) != 0:  
 for classId, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):  
 className = classNames[classId - 1]  
 if className in objects:  
 objectInfo.append([box, className])  
 if (draw):  
 cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)  
 cv2.putText(img, className.upper(), (box[0] + 10, box[1] + 30),  
 cv2.FONT\_HERSHEY\_COMPLEX,1,(0,255,0),2)  
 cv2.putText(img, str(round(confidence \* 100, 2)), (box[0] + 200, box[1] + 30),  
 cv2.FONT\_HERSHEY\_COMPLEX, 1, (0, 255, 0), 2)  
 return img, objectInfo  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 cap = cv2.VideoCapture("http://192.168.1.34:81/stream")  
 #cap.set(3, 1280)  
 #cap.set(4, 720)  
 #cap.set(10, 70)  
  
 while True:  
 success, img = cap.read()  
 result,objectInfo = getObjects(img, True)  
  
 cv2.imshow("Detection Window", img)  
 cv2.waitKey(1)

## האפליקציה

### צילומי מסך





## קוד - main activity

package com.moob.app;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.webkit.WebView;  
import android.widget.Button;  
import android.widget.CheckBox;  
import android.widget.TextView;  
  
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 io.github.controlwear.virtual.joystick.android.JoystickView;  
  
public class MainActivity extends AppCompatActivity {  
  
 Button switchView;  
 CheckBox lightBox;  
 TextView tempText;  
 WebView camView;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 switchView = findViewById(R.id.*switchButton*);  
 lightBox = findViewById(R.id.*lightBox*);  
 tempText = findViewById(R.id.*tempText*);  
 camView = findViewById(R.id.*camView*);  
  
  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 DatabaseReference myRef = database.getReference("messages");  
  
 JoystickView joystick = (JoystickView) findViewById(R.id.*joystick*);  
 joystick.setOnMoveListener(new JoystickView.OnMoveListener() {  
 @Override  
 public void onMove(int angle, int strength) {  
  
 if(strength < 60){ //stop car when no input  
 myRef.child("motors").setValue(0);  
 return;  
 }  
  
 if(angle < 135 && angle > 45) //forward  
 myRef.child("motors").setValue(2);  
  
 if(angle < 315 && angle > 225)//backward  
 myRef.child("motors").setValue(1);  
  
 if(angle < 45 && angle > 0 || angle > 315 && angle < 360)//right  
 myRef.child("motors").setValue(3);  
  
 if(angle < 180 && angle > 135 || angle > 180 && angle < 225)//left  
 myRef.child("motors").setValue(4);  
 }  
 });  
  
 lightBox.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if(lightBox.isChecked()){  
 myRef.child("leds").setValue(1);  
 }else {  
 myRef.child("leds").setValue(0);  
 }  
 }  
 });  
  
 myRef.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 String temp = dataSnapshot.child("temperature").getValue().toString();  
 tempText.setText(CalculateCel(Integer.*parseInt*(temp)));  
  
 String ip = dataSnapshot.child("ip").getValue().toString();  
 if (ip != "")  
 {  
 camView.loadUrl(ip);  
 }  
 }  
 @Override  
 public void onCancelled(DatabaseError error) {  
 // Failed to read value  
 Log.*w*("Firebase", "Failed to read value.", error.toException());  
 }  
 });  
  
 switchView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intet = new Intent(MainActivity.this, ServoController.class);  
 startActivity(intet);  
 finish();  
 }  
 });  
 }  
 public String CalculateCel(int tempInC){  
 tempInC -= 32;  
 tempInC /= 1.8000;  
  
 return String.*valueOf*(tempInC - 25);  
 }  
}

## קוד – Servo Controller

package com.moob.app;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
  
  
import com.google.android.material.slider.Slider;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
  
import io.github.controlwear.virtual.joystick.android.JoystickView;  
  
public class ServoController extends AppCompatActivity {  
  
 Button switchButton;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_servo\_controller*);  
  
 FirebaseDatabase database = FirebaseDatabase.*getInstance*();  
 DatabaseReference myRef = database.getReference("messages");  
  
 switchButton = findViewById(R.id.*switchViewToMain*);  
  
 Slider servoSlider = (Slider) findViewById(R.id.*ServoSlider*);  
 servoSlider.addOnSliderTouchListener(new Slider.OnSliderTouchListener() {  
 @Override  
 public void onStartTrackingTouch(@NonNull Slider slider) {  
 switch((int) slider.getValue()) {  
 case 0:  
 myRef.child("ServoMotors").setValue(0);  
 break;  
 case 10:  
 myRef.child("ServoMotors").setValue(1);  
 break;  
 case 20:  
 myRef.child("ServoMotors").setValue(2);  
 break;  
 case 30:  
 myRef.child("ServoMotors").setValue(3);  
 break;  
 case 40:  
 myRef.child("ServoMotors").setValue(4);  
 break;  
 case 50:  
 myRef.child("ServoMotors").setValue(5);  
 break;  
 case 60:  
 myRef.child("ServoMotors").setValue(6);  
 break;  
 case 70:  
 myRef.child("ServoMotors").setValue(7);  
 break;  
 default:  
 myRef.child("ServoMotors").setValue(0);  
 }  
  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(@NonNull Slider slider) {  
  
 }  
 });  
  
  
 switchButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intet = new Intent(ServoController.this, MainActivity.class);  
 startActivity(intet);  
 finish();  
 }  
 });  
 }  
}

## קוד - activity\_main 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:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <WebView  
 android:id="@+id/camView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="360dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/tempText"  
 android:layout\_width="50dp"  
 android:layout\_height="40dp"  
 android:layout\_marginStart="15dp"  
 android:layout\_marginTop="15dp"  
 android:gravity="center"  
 android:text="@string/\_255"  
 android:textSize="20sp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/camView" />  
  
 <Button  
 android:id="@+id/switchButton"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="60dp"  
 android:layout\_marginTop="15dp"  
 android:layout\_marginEnd="60dp"  
 android:text="@string/switch\_view"  
 app:layout\_constraintEnd\_toStartOf="@+id/lightBox"  
 app:layout\_constraintStart\_toEndOf="@+id/tempText"  
 app:layout\_constraintTop\_toBottomOf="@+id/camView" />  
  
 <CheckBox  
 android:id="@+id/lightBox"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="15dp"  
 android:layout\_marginEnd="15dp"  
 android:text="@string/lights"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/camView" />  
  
 <io.github.controlwear.virtual.joystick.android.JoystickView  
 android:id="@+id/joystick"  
 android:layout\_width="300dp"  
 android:layout\_height="300dp"  
 android:layout\_marginStart="130dp"  
 android:layout\_marginEnd="130dp"  
 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\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

## קוד - activity\_servo\_controller 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:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".ServoController">  
  
 <WebView  
 android:id="@+id/camView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="360dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <Button  
 android:id="@+id/switchViewToMain"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="136dp"  
 android:layout\_marginTop="396dp"  
 android:text="@string/switch\_view"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <com.google.android.material.slider.Slider  
 android:id="@+id/ServoSlider"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:contentDescription="SliderServo"  
 android:stepSize="10"  
 android:value="40"  
 android:valueFrom="0"  
 android:valueTo="80"  
 app:haloColor="@color/white"  
 app:labelBehavior="floating"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.736"  
 app:thumbColor="@color/purple\_500"  
 app:tickColor="@color/white"  
 app:trackColor="@color/purple\_500" />  
  
  
</androidx.constraintlayout.widget.ConstraintLayout>

## קוד - Settings gradle

dependencyResolutionManagement **{** repositoriesMode.set(RepositoriesMode.FAIL\_ON\_PROJECT\_REPOS)  
 repositories **{** google()  
 mavenCentral()  
 jcenter() // Warning: this repository is going to shut down soon  
 **}  
}**rootProject.name = "Moob"  
include ':app'

## קוד – colors xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <color name="purple\_200">#FFBB86FC</color>  
 <color name="purple\_500">#FF6200EE</color>  
 <color name="purple\_700">#FF3700B3</color>  
 <color name="teal\_200">#FF03DAC5</color>  
 <color name="teal\_700">#FF018786</color>  
 <color name="black">#FF000000</color>  
 <color name="white">#FFFFFFFF</color>  
</resources>

## קוד - strings xml

<resources>  
 <string name="app\_name">Moob</string>  
 <string name="lights">Lights</string>  
 <string name="switch\_view">Switch View</string>  
 <string name="\_255">255</string>  
 <!-- *TODO: Remove or change this placeholder text* -->  
 <string name="hello\_blank\_fragment">Hello blank fragment</string>  
</resources>

## קוד - themes xml

<resources xmlns:tools="http://schemas.android.com/tools">  
 <!-- Base application theme. -->  
 <style name="Theme.Moob" parent="Theme.MaterialComponents.DayNight.DarkActionBar">  
 <!-- Primary brand color. -->  
 <item name="colorPrimary">@color/purple\_500</item>  
 <item name="colorPrimaryVariant">@color/purple\_700</item>  
 <item name="colorOnPrimary">@color/white</item>  
 <!-- Secondary brand color. -->  
 <item name="colorSecondary">@color/teal\_200</item>  
 <item name="colorSecondaryVariant">@color/teal\_700</item>  
 <item name="colorOnSecondary">@color/black</item>  
 <!-- Status bar color. -->  
 <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>  
 <!-- Customize your theme here. -->  
 </style>  
</resources>

## קוד - themes xml (night)

<resources xmlns:tools="http://schemas.android.com/tools">  
 <!-- Base application theme. -->  
 <style name="Theme.Moob" parent="Theme.MaterialComponents.DayNight.DarkActionBar">  
 <!-- Primary brand color. -->  
 <item name="colorPrimary">@color/purple\_200</item>  
 <item name="colorPrimaryVariant">@color/purple\_700</item>  
 <item name="colorOnPrimary">@color/black</item>  
 <!-- Secondary brand color. -->  
 <item name="colorSecondary">@color/teal\_200</item>  
 <item name="colorSecondaryVariant">@color/teal\_200</item>  
 <item name="colorOnSecondary">@color/black</item>  
 <!-- Status bar color. -->  
 <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>  
 <!-- Customize your theme here. -->  
 </style>  
</resources>

# 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 והיינו צריכים להחליף אותם, גם בשביל לחבר את הכניסות של המנועים למסרקים של האלטרה וה – ESP32 היינו צריכים להלחים אותם.

# אבולוציית הפרויקט

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:

# 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>

### 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://www.computervision.zone/topic/nms-code/>

### מנוע סרוו

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

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