פרויקט גמר י"ב

רובוט משדר תמונה ונתונים לאחור בטכנולוגית IOT

דרכא בית ירח

אור-ים ביבי רובי, עידן קורז'ק

[סיכום רכיבים 2](#_Toc118467400)

[דוחף זרםL293d – 2](#_Toc118467401)

[עיקרון הפעולה 3](#_Toc118467402)

[מבנה פנימי של הרכיב 3](#_Toc118467403)

[מנוע Direct Current Motor - DC- 4](#_Toc118467404)

[PWM 6](#_Toc118467405)

[חיישן טמפרטורה – 18b20 8](#_Toc118467406)

[בקר – ESP32 10](#_Toc118467407)

[עקרון פעולה 12](#_Toc118467408)

[מבנה פנימי 12](#_Toc118467409)

[פרוטוקולים- ESP32 14](#_Toc118467410)

[ESP32 - CAM 16](#_Toc118467411)

[עקרון פעולה 16](#_Toc118467412)

[Altera 17](#_Toc118467413)

[FPGA-Field-Programmable Gate Array 17](#_Toc118467414)

[עקרון פעולה 18](#_Toc118467415)

[מבנה פנימי 18](#_Toc118467416)

[תקן תקשורת RS232 18](#_Toc118467417)

[LM75 – Digital Temperature Sensor 21](#_Toc118467418)

[תוכניות VHDL 25](#_Toc118467419)

# סיכום רכיבים

## 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.

### מבנה 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).

# תוכניות VHDL

### 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;

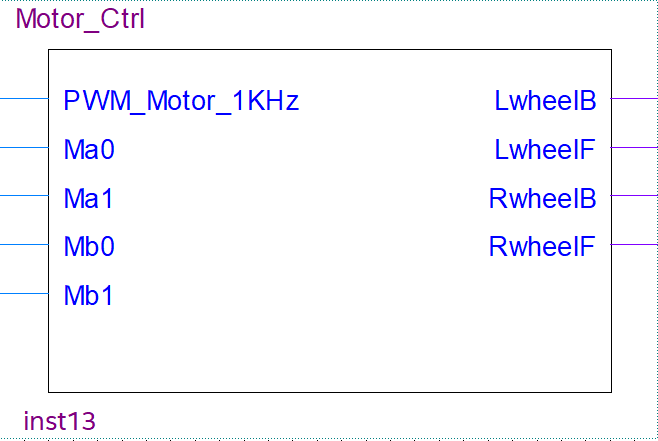
end RTL;



### 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 <= '0';

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 <= '0';

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;



ביבליוגרפיה

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

<https://pdf1.alldatasheet.com/datasheet-pdf/view/58559/DALLAS/18B20.html> <https://disti-assets.s3.amazonaws.com/testco-inc/files/datasheets/25281.pdf>

<https://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf> <https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf>

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

<https://www.engineersgarage.com/l293d-pin-description-and-working/>

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

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