ספר פרויקט

מערכות אוטונומיות

שם המכללה: מכללת מקיף ה' אשקלון דרכא

סמל מוסד: 644450

שם הסטודנט: לידור דוקרקר

ת.ז של הסטודנט: 324948058

שם המנחה: משה זזק

שם הפרויקט: מערכת המתריעה בזמן אמת שרכב זר חונה בחניה הפרטית של אדם אחר

מקיף ה'
דרכא
אשקלון

תוכן עניינים

תקציר:

הפרויקט שלי עוסק בבעיה שאני ומשפחתי עוסקים בה יום ויום זה בעצם מערכת שמתריעה בזמן אמת שרכב זר חונה בחניה הפרטית של אדם שהוא בעל החניה. והיא עושה שימוש בזיהוי לוחית רישוי של רכב הזר ושיגורו לבעל החניה.

בעל החניה יוכל להפעיל מערכת אזעקה ברגע שיקבל את ההתרעה על רכב זר שחונה בחניה שלו ובכך לגרום לאדם שחנה לו בחניה להזיז את רכבו או לפחות להזכיר לו שהוא חונה בחניה שלא שלו למשל שכן שהתבלבל בחניה. האזעקה לא תפסיק עד שהאדם הזר יזיז את רכבו.

על מנת לממש את הפרויקט עשיתי שימוש בעיבוד תמונה עם raspberry phi 4 מעבד בעל מעבד חזק ועוצמתי. מכיוון שעיבוד תמונה דורש זמן עיבוד עשיתי גם שימוש בחיישן מרחק ultrasonic על מנת לייעל את המערכת בכך שחיישן זה יזהה האם רכב מתחיל לחנות בחניה ובכך אנחנו צמצמים משמעותית את זמן העיבוד של המערכת. מערכת האזעקה מורכבת מ- arduino nano , LCD קטן על מנת להציג טקסט ו – buzzer שיצפצף עד שהרכב יזוז.

התוצאות היו מוצלחות המערכת הצליחה לזהות את רוב הרכבים שצילמתי ונתתי לה בהתאם לדרישות כלומר הרכב החונה הוא חונה ישר כלומר לא על שני חניות או משהו כזה ותנאיי הסביבה סבירים ולא קיצוניים המקשים על עיבוד התמונה ופעונחו.

מבוא:

סקירה כללית של הפרויקט:

עיבוד תמונה הוא תחום במדעי המחשב המתמקד במעבדה, מעבדה ושיפור תמונות דיגיטליות. עיבוד תמונה כולל מגוון רחב של טכניקות וכלים, החל מתהליכים פשוטים כמו חיתוך והגדלה ועד לטכניקות מתקדמות יותר כמו השימוש בלמידה עמוקה ובאלגוריתמים של רשתות נוירונים.

בין הטכניקות הנפוצות בעיבוד תמונה ניתן למנות:

- תהליכי פילוט תמונה: כוללים חיתוך תמונה, הסרת רעש וחיסור רקע.
- תהליכי שיפור תמונה: כוללים טיפול באיכות התמונה כמו תקן צבעים ובהירות, פילטרים למיזוג וזרימת תמונות, ושיפור תמונות רפלקטיביות.
- תהליכי זיהוי והבנת תמונה: כוללים זיהוי עצמים ופנים בתמונות, זיהוי מרקמים וצורות, ותהליכי הזדהות.
- תהליכי תפעול על תמונות: כוללים החלפת צבעים וטקסטורות, גרפיקה ממוחשבת, ותהליכי הרחבה כמו זום וקירוב.

תחום זה חשוב ביישומים מגוונים כגון רפואה, תעשייה, גרפיקה ממוחשבת, אבטחת מידע ועוד. בין היתר זיהוי לוחית רישוי עוסקת בעיבוד תמונה.

זיהוי לוחית רישוי הוא תחום חשוב של עיבוד תמונה. כדי לזהות לוחית רישוי בתמונה, יש לבצע רצף של צעדים.

השלב הראשון הוא הגעת התמונה לעיבוד דיגיטלי, כלומר הפיכתה לפורמט דיגיטלי. לאחר מכן, יש לבצע עיבוד תמונה כדי לזהות את האזור בתמונה שבו יש לוחית רישוי. כך ניתן להשתמש בטכניקות של זיהוי קווים ומרחקים בן הקווים באמצעות פילטרים מתאימים לזיהוי קווים.

השלב הבא הוא זיהוי סימנים ייחודיים בלוחית הרישוי, כמו אותיות ומספרים, באמצעות טכניקות לזיהוי אובייקטים ולמיפוי עצמי של האובייקטים הללו. ניתן להשתמש בטכניקות כמו רשתות נוירונים עמוקות לזיהוי תבניות וסימנים ייחודיים.

סוף סוף, יש להחזיר את המידע המתאים מהתמונה, כמו מספר הרישוי וכל נתוני המידע הקשורים אליו, באמצעות תהליך ניתוח הנתונים של הלוחית הרישוי והפעולות המתאימות לחישוב וזיהוי המידע המבוקש.

כולל יש לציין שזיהוי לוחית רישוי הוא תחום מורכב ושונה, ולכן תהליך העיבוד תמונה עשוי לכלול מספר צעדים נוספים, כגון:

- קירוב גודל הלוחית הרישוי - לפעמים נדרש להתאים את גודל הלוחית הרישוי שנזהה לגודל המצופה על ידי המערכת המבצעת את זיהוי הרישוי.
- זיהוי לוחיות רישוי מרובות - כאשר יש כמה לוחיות רישוי בתמונה, יש להפריד ביניהן ולזהות את המספר המתאים לכל לוחית.
- התאמת זוויות - לפעמים נדרש לסדר את התמונה כך שהלוחית הרישוי יהיה בזווית מתאימה כדי להפחית את הסיכוי לשגיאות זיהוי.

- מתן פידבק - כאשר נזהה לוחית רישוי, נדרש להחזיר פידבק למשתמש אם המספר שזוהה אותו הוא תקין או לא.

בסיכום, עיבוד תמונה משמש כלי חשוב לזיהוי לוחות רישוי ופענוחה של מספרים בתמונה. התהליך כולל מספר צעדים שונים וכלי טכנולוגיים מתקדמים לזיהוי ופענוחה של לוחות רישוי באופן יעיל ומדויק.

בעיה האלגוריתמית של זיהוי לוחית רישוי על ידי עיבוד תמונה היא בעיה שנמצאת בתחום מדעי המחשב ובעיקר בתחום הראייה הממוחשבת. המטרה של הבעיה היא לזהות את מיקום ותוכן הלוחית הרישוי בתמונה דיגיטלית של מכונית.

הבעיה הזו קשה כי הלוחית הרישוי יכולה להיות בגדלים וצורות שונות, כמו גם להיות מוטמעת בתוך רקע מגוון של צבעים ופסים. כדי לפתור את הבעיה הזו, משתמשים במגוון של טכניקות עיבוד תמונה ולמידה עמוקה.

אחת הטכניקות הנפוצות לזיהוי לוחות רישוי היא תהליך המכונה "זיהוי ישיר". בתהליך זה, משתמשים בפעולות חשיבה מתמטית ובעיות הגיוניות כדי לזהות את הלוחית הרישוי. ככל שהתמונה יותר מסובכת ויותר שונות הלוחות הרישוי בה, כך יהיה קשה יותר לבצע זיהוי ישיר.

טכניקה נוספת המשמשת לזיהוי לוחות רישוי היא תהליך המכונה "למידה עמוקה". במסגרת התהליך הזה, משתמשים במודלי למידה עמוקה ללמידה ולאימון על תמונות של לוחות רישוי. המודל מכיר את מאפייני הלוחית הרישוי באופן אוטומטי, וכך יכול לזהות לוחות רישוי על בסיס זיהוי המאפיינים הללו בתמונה. התהליך הזה עשוי להיות יעיל יותר מתהליך הזיהוי הישיר כיוון שהוא מאפשר למודל ללמוד ולהתאים את עצמו למגוון רחב של צורות וגדלי לוחות רישוי וכן לרקעים ותנאי תאורה שונים.

כדי להשיג זיהוי יעיל של לוחות רישוי, עשוי להיות צורך בשימוש במספר טכניקות עיבוד תמונה נוספות כמו סינון רעשים, תיקון אור, חיתוך ופילוח תמונה, ועוד.

בעיית זיהוי לוחות רישוי נחשבת לפופולרית ועוסקים בה חוקרים ומפתחים בתחום הראייה הממוחשבת. הבעיה מצוינת כדוגמה לאפליקציות של תמונות רפלקטיביות בכביש, ואף נעשה שימוש בה ביישומי אבטחה ומעקב באוטומציה של תנועת כלי רכב.

עיבוד תמונה מצריך מעבד חזק במיוחד כדי לבצע את החישובים המורכבים שדרושים לעיבוד ולזיהוי אובייקטים בתמונה. העיבוד של תמונות מצריך חישוב מקבילי ומספר רב של פעולות מתמטיות, ולכן מעבדים עם יכולת חישוב גבוהה נדרשים כדי לבצע עיבוד תמונה בצורה מהירה ויעילה. ולכן על מנת לייעל את עיבוד התמונה אנחנו נשתמש ב – raspberry phi 4 רכיב שיעזור לנו בבעיית מעבד זמן CPU.

מטרות והיקף הפרויקט:

מטרת הפרויקט היא לאפשר למחשב לזהות אובייקטים מסוימים בתמונה או בווידאו ולזהות אפיונים ומאפיינים מסוימים על מנת לזהות אותם בצורה יעילה מדויקת יותר.

בפרויקט זה, המטרה היא לזהות לוחות רישוי ולפענח את המספר המופיע עליהם. ההיקף של הפרויקט כולל מספר צעדים טכניים להשגת המטרה.

בכל היקף הפרויקט, חשוב להבטיח שהזיהוי מתבצע בצורה מדויקת ויעילה כדי להפחית את הסיכון לטעויות זיהוי או לחוסר יעילות בתהליך. כמו כן, חשוב לבצע עבודת מחקר ופיתוח מתמשכת כדי לשפר ולפתח את הטכנולוגיות המתקדמות שמאפשרות זיהוי מדויק של חפצים ואובייקטים בתמונות ובווידאו.

דרישות המערכת ורכיבי החומרה:

למערכת שאני אפתח יש כמה דרישות בין היתר הם:

- יעילות
- מהירות
- חסכוני מבחינה כספית

יעילות צריך על מנת שהמערכת לא תעבוד לשווא כלומר לא לגרום למערכת לעבוד מתי שהיא לא צריכה לעבוד למשל עם המערכת תשב על הקיר מול החניה ועובר אובייקט זר שהוא לא רכב אז צריך מנות שהיא תעבוד סתם כלומר שלא ישר תחפש לוחית רישוי על מנת להגבר על קושי זה נחפש רכב בתמונה ואז נמשיך בתהליך של המערכת.

מהירות צריך על מנת שהמערכת תעבד תמונה בצורה הכי מהירה על מנת לעדכן את המשתמש בזמן אמת שרכב זר חונה לו בחניה הפרטית.

חסכוני מבחינה כספית אנחנו צריכים להנגיש את המוצר לכל המשתמשים ומוצר זול יכול להוות גורם משמעותי לרכישה בקרב אנשים.

Ultrasonic HC-SR04 חיישן מרחק – הוא מודול חיישן קולי שנמצא בשימוש נרחב ביישומי מדידת מרחק. הוא פועל על ידי שליחת דופק קולי ומדידת הזמן שלוקח לדופק לחזור אחורה לאחר פגיעה בחפץ. המודול מורכב מזוג משדרים, ויכול למדוד במדויק מרחקים של עד 4 מטר מפה המספר 04 בדיוק של 3 מילימטר. ה – HC-SR04 קל לשימוש וניתן לשלב אותו עם כל מיני מיקרו בקרים בין היתר Arduino nano ו – raspberry pi 4 שנשתמש בפרויקט.

המודול מופעל על ידי ספק 5V ובעל צריכת חשמל נמוכה מה שהופך אותו ליישומים המופעלים על ידי סוללה. ה – HC-SR04 נמצא בשימוש נרחב ברחבי העולם ועולם האלקטרוניקה בשל העובדה שהוא בעל עלות נמוכה וקל לשימוש.

לחיישן זה יש 4 רגליים או פינים והם:

- GND - פין זה מחובר למסוף השלילי של ספק הכוח.
- VCC - פין זה מחובר למסוף החיובי של ספק הכוח.
- TRIG - באמצעות פין זה אנו שולחים את גל האולטרסאונד מהמשדר.
- ECHO - באמצעות פין זה אנו מקשיבים לאות המשתקף.

אני משתמש בחיישן מרחק על מנת לא לגרום ל – raspberry pi 4 לעבוד מתי שהוא לא צריך כלומר ברגע שהחיישן יזהה שאובייקט מתקרב הוא יתחיל את תהליך עיבוד התמונה.

Arduino nano מיקרו-בקר – הוא לוח קומפקטי ורב – תכליתי המבוסס על המיקרו-בקר ATmega328P , שהוא אותו שבב המשתמש ב – Arduino uno. ההנו מתוכנן להיות קטן ובעלות נמוכה, מה שהופך אותו לבחירה אידיאלית עבור פרויקטים הדורשים גורם צורה קומפקטי.

ללוח יש בסך הכל 22 פיני קלט \ פלט דיגיטליים, 6 מהם יכולים לשמש כיציאות PWM ו – 8 כניסות אנלוגיות. הוא כולל גם ממשק USB לתכנות ומתח. ניתן לתכנת את ה – nano באמצעות סביבת הפיתוח המשולבת Arduino ide וניתן להשתמש בו כדי לשלוט במגוון רחב

של רכיבים אלקטרוניים כגון חיישנים, מנועים וצגים. אני אעשה שימוש ברכיב זה על לשלוט בצג ובחיישן קולי כלומר buzzer.

Arduino nano הוא בחירה מצוינת עבור סטודנטים ואנשי מקצוע בשל גודלו הקטן, העלות הנמוכה והמגוון שיש בו. הוא נמצא בשימוש נפוץ בפרויקטים כמו רובוטיקה ותכנות.

רכיב זה שולט במערכת האזעקה על מנת להתריע לאדם החונה שהינו חונה בחניה פרטית שאינו שלו.

ws28112b rgb led 24-bit ring מעגל של 24 LED – הוא סוג פופולרי של LED RGB הכולל שבב דרייבר משולב וניתן לשלוט בו באמצעות סיכת נתונים אחת. זה מקל על שליטה במספר גדול של נוריות WS2812B באמצעות רק כמה פינים של מיקרו-בקר.

טבעת WS2812B RGB LED היא מערך מעגלי של 24 נוריות WS2812B הניתנות לשליטה בנפרד כדי להציג מגוון רחב של צבעים ואנימציות. לכל WS2812B LED בטבעת עומק צבע של 24 סיביות, המאפשר למעלה מ- 16 מיליון שילובי צבעים שונים מה שהופך אותה ל-RGB. טבעת WS2812B LED מופעלת בדרך כלל על ידי אספקת 5V וניתנת לשליטה באמצעות פלטפורמות כגון Arduino ו – Raspberry pi. טבעת ה- WS2812B LED נמצאת בשימוש נפוץ בפרויקטים של אלקטרוניקה מכיוון שעלותו נמוכה וקל לשימוש והצבעים שהוא מייצג יפים ונעימים.

אני אעשה שימוש ברכיב זה על מנת להקל על החונה בכך שהצבע ברכיב זה יתמלא בהתאם למרחק של הרכב מהקיר שהרכב מתחיל להתקרב אליו.

לרכיב זה 4 רגליים או פינים בשפה המקצועית והם:

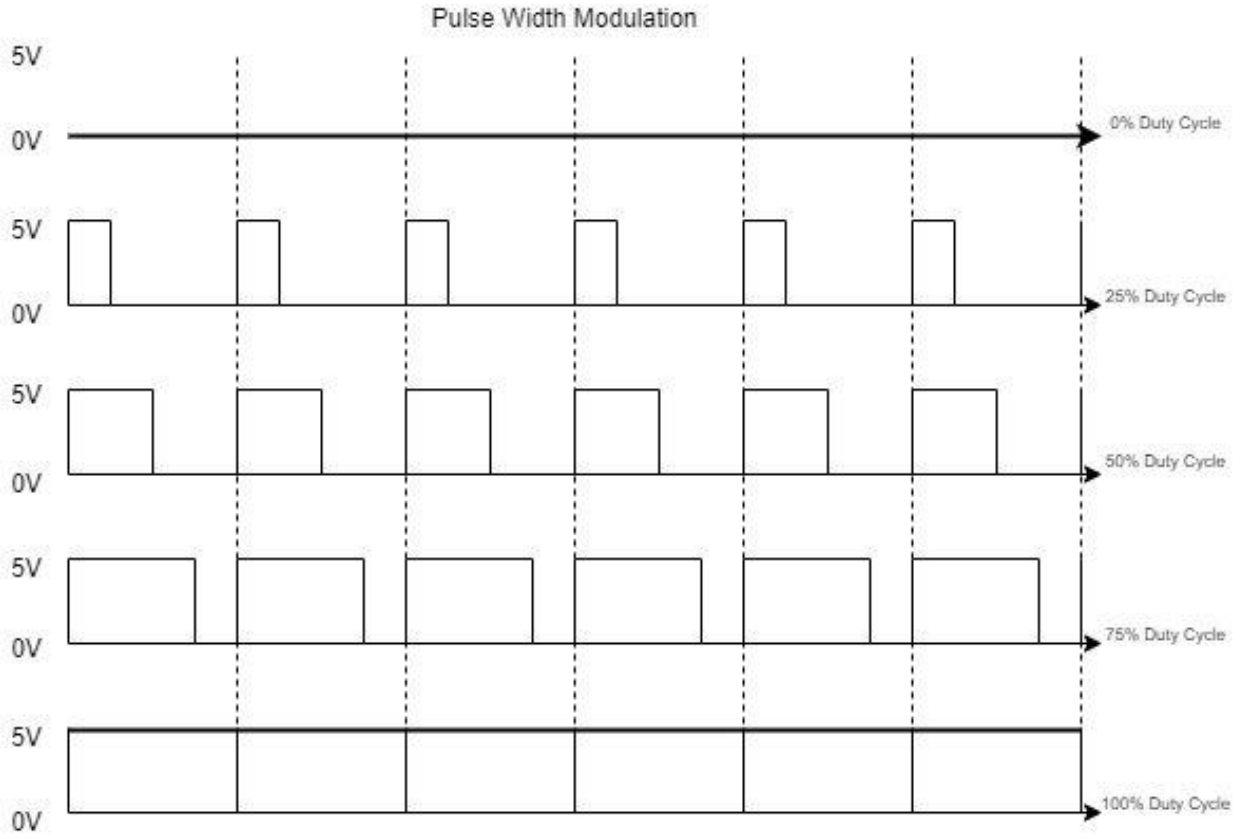
- VCC - פין זה מחובר למסוף החיובי של ספק הכוח. ל-WS2812B מתח הפעלה מומלץ של V5, אך ניתן להפעיל אותו גם ב-V3.3.
- GND - פין זה מחובר למסוף השלילי של ספק הכוח.
- DI – קיצור של DATA IN סיכה זו משמשת להזנת אות הנתונים לתוך ה-LED הראשון בשרשרת. אות הנתונים הוא זרם נתונים סדרתי המורכב מ-24 סיביות של נתונים עבור כל LED בשרשרת.
- DO – קיצור של DATA OUT סיכה זו משמשת לפלט את אות הנתונים מה-LED הראשון בשרשרת אל LED הבא בשרשרת. זה מאפשר לשרשרת מספר נוריות WS2812B יחד ולשלוט על ידי אות נתונים בודד.

הפינים של ארדואינו מוציאים מתח של 5V או 0V כמו במספריים בינאריים אחד ואפס כאשר אפס מייצג את 0V ואחד מייצג 5V , אך לעיתים אנחנו רוצים להוציא מתח ששונה מהמתח שארדואינו יכול להציג לדוגמא במקרה שאנחנו רוצים לעבוד עם LED RGB ורוצים לייצר צבע המורכב ממספר צבעי יסוד.

אחד הפתרונות לבעיה זאת היא PWM שינוי של המתח במוצא הפין בין גבוה לנמוך בתדר מהיר יחסית, ברוחב פולס משתמה, כך שבאופן אפקטיבי יש מתח ביניים.

PWM הוא קיצור של Pulse Width Modulation אפנון רוחב פולס. במצב זה המתח יהיה 5V או 0V אבל משך השהייה על 5V ישתנה לפי המתח האפקטיבי שנרצה לקבל.

להלן דיאגרמה המתארת את PWM:



<u>Arduino buzzer זמזם –</u> הוא התקן פלט שמע קטן שניתן לשלוט בו על ידי מיקרו-בקר. הוא נמצא בשימוש נפוץ בפרויקטים אלקטרוניים העוסקים במגוון רחב של צלילים, אזעקות, צפצופים ומנגינות. הזמזם מורכב בדרך כלל ממרכיב אלקטרומכני הרוטט במהירות כאשר מופעל אות חשמלי, ויוצר גל קול שניתן לשמוע על ידי בני אדם. זמזמי Arduino מונעים בדרך כלל על ידי פיני פלט דיגיטליים, כאשר הטון ומשך הצליל נשלטים על ידי התוכנה הפועלת על ה- Arduino ide כולל ספריית צלילים המפשטת את תהליך הפקת צלילים שונים עם הזמזם. זמזם Arduino הוא רכיב פופולרי ורב-תכליתי שקל לשימוש ומספק דרך פשוטה להוסיף פלט אודיו לכל פרויקט Arduino.

אעשה שימוש ברכיב זה על מנת להפעיל את מערכת האזעקה המתריעה שרכב זר חונה בחניה פרטית של העל החניה.

לרכיב זה 2 רגליים או פינים, אחד מהפינים הוא פין מתח של 5V והשני הוא פין של אדמה GND.

<u>0.91in OLED display צג –</u> הוא מודול תצוגה קומפקטי, בעל הספק נמוך שניתן להשתמש בו להצגת מידע בפרויקטים אלקטרוניים שונים. טכנולוגיית OLED משמשת בחומרים אורגניים ליצירת שכבה פולטת אור דקה וגמישה, המאפשרת ליצור צגים דקים, קלים וחסכוניים באנרגיה. תצוגת OLED בגודל 0.91 אינץ' היא בדרך כלל בעלת רזולוציה של 128x32 פיקסלים, שיכולה להציג גרפיקה וטקסט פשוטים. הוא כולל בדרך כלל ממשק I2C

או למחשב עם לוח יחיד, כגון Arduino או raspberry pi. תצוגת ה- OLED משמשת בדרך כלל במכשירים שבהם צריכת החשמל היא גורם קריטי.

OLED בגודל 0.91 אינץ' הוא בחירה מצוינת מכיוון מחירו ועלותו הנמוכה ובנוסף הוא קל לשימוש וקומפקטי.

לרכיב זה 4 פינים והם:

- VCC - פין זה מחובר למסוף החיובי של ספק הכוח. מתח ההפעלה המומלץ עבור תצוגת ה-OLED הוא בדרך כלל בין V3.3 ל-V5, אך הוא יכול להשתנות בהתאם למודול התצוגה הספציפי.
- GND - פין זה מחובר למסוף השלילי של ספק הכוח.
- SCL - סיכה זו משמשת לסנכרון שעון בין מודול התצוגה לבין המיקרו-בקר. הוא חלק מפרוטוקול I2C (מעגל משולב) המשמש לתקשורת בין תצוגת ה-OLED למיקרו-בקר.
- SDA - סיכה זו משמשת להעברת נתונים בין מודול התצוגה לבין המיקרו-בקר. הוא גם חלק מפרוטוקול I2C ומשמש לשליחת פקודות והצגת נתונים לתצוגת ה-OLED.

עשיתי שימוש ב – OLED ולא ב – LCD מהסיבות הבאות:

- מבחינה טכנולוגית מסכי OLED פולטות אור כאשר זרם חשמלי עובר דרכם. כל פיקסל ב – OLED פולט את עצמו וניתן להפעילו או לכבות אותו בנפרד בניגוד למסכי LCD לעומת זאת משתמשים במערך של דיודות פולטות אור, דיודות אלו מאירות את הפיקסלים על המסך, והפיקסלים עצמם לא פולטים אור.
- למסכי OLED יש יחס ניגודיות יוצאי דופן מכיוון שכל פיקסל יכול לפלוט אור משלו או להיות כבוי לחלוטין. זה מאפשר לצבע השחור להיות עמוק יותר ואמיתי ומפיק צבעים מרהיבים ומלאים. במסכי LCD , יחס הניגודיות מוגבל על ידי התאורה האחורית, שעלולה לפגוע באזורים כהים יותר ועלול לגרום לדליפה של האור וכתוצאה מכך לגרום לבצע השחור להיות פחות מדויק.
- צגי OLED מאפשרים זוויות צפייה מצוינות ולא רגילות כלומר איכות התמונה נשארת קבועה ועקבית ללא קשר לזווית הצפייה. מסכי LCD יכולים לגרום לשינוי בצבע כאשר אנחנו צופים בו מזווית צפייה קיצונית מה שעלול לגרום לבעיה אצלי בפרויקט.
- מסכי OLED חסכוניים בהרבה ממסכי LCD בהצגת תוכן כהה יותר או תמונות שחורות בעיקר. הסיבה לכך היא כי ניתן לכבות בנפרד פיקסלים של OLED , ולמעשה לא צורכים חשמל כדי לייצר את הצבע השחור. לעומת זאת, מסכי LCD דורשים שהתאורה האחורית תהיה דולקת תמידית וכתוצאה מכך צריכת החשמל עולה וגבוהה יותר ללא קשר לתוכן המיוצג במסך ה – LCD.
- במסך OLED זמן התגובה מהיר בהרבה מזמן התגובה שנמצא במסך LCD מאותה סיבה שכל פיקסל במסך OLED יכול להידלק ולכבות במהירות גורם זה מפחית את טשטוש תנועה ומעברים חלקים יותר. מסך LCD יכולים להפגין זמני תגובה איטיים בעיקר דגמים ישנים יותר ולגרום לתמונה להיות מטושטשת.
- מסכי LCD נוטים להיות עבים יותר ופחות גמישים מאשר מסכי OLED שאותם ניתן להפוך אותם לגמישים ודקים יותר.

ברכיב זה אעשה שימוש על מנת להציג לחונה שאינו מורשה לחנות בחניה שלי ושהוא חונה לי בחניה פרטית ואליו להזיז את הרכב.

בקר Raspberry pi 4 – הוא האיטרציה האחרונה של המחשב הפופולרי בעל לוח יחיד שפותח על ידי קרן raspberry pi. זהו מכשיר רב עוצמה ורב-תכליתי שיכול להריץ מגוון מערכות הפעלה, כולל לינוקס ו- windows ה- raspberry pi4 כולל מעבד עם ארבע ליבות במהירות 1.5HZ עם אפשרות בחירה זיכרון RAM של 2GB, 4GB, 8GB אני אעשה שימוש ב- 4GB של RAM הוא כולל WI-FI וגם Ethernet עם 2 יציאות USB 3.0 וגם 2 יציאות USB 2.0. שתי יציאות של מיקרו HDMI המסוגלות לתמוך בשני צגי 4K. המכשיר נמצא בשימוש רחב של פרויקטים, כגון אוטומציה ביתית, רובוטיקה ועיבודי תמונה. בקר זה הוא בחירה פופולרית מכיוון עלותו הנמוכה וגורם הצורה הקטן הגודל שלו הוא כמו כרטיס אשראי.

אני אעשה שימוש בבקר זה מכיוון שרכיב זה מספיק חזק לעיבוד התמונה שצולמה על ידו ופיעונחו במהירות בשל עוצמתו של המעבד וה- RAM שיש לו.

בקר ESP32 - הוא לוח מיקרו-בקר רב עוצמה ורב-תכליתי שנמצא בשימוש נרחב ביישומי האינטרנט. הוא מבוסס על שבב ESP32, הכולל מעבד כפול ליבה, Wi-Fi, Bluetooth.

לוח ה-ESP32 כולל בדרך כלל מגוון חיישנים, כגון חיישני טמפרטורה ולחות, וניתן לחבר אותו בקלות לחיישנים ומפעילים אחרים באמצעות מגוון פרוטוקולי תקשורת, כולל I2C, SPI ו- UART. ניתן לתכנת את ה-ESP32 באמצעות Arduino IDE.

ה-ESP32 הוא בחירה פופולרית בגלל שני גורמים עיקריים בשל עלותו הנמוכה וגודלו הקומפקטי.

אני משתמש בבקר זה על מנת לשלוט ב- WS2812B OLED וב- חיישן מרחק על מנת לקבל מידע על הסביבה.

Breadboard מטריצה – לוח בצורת מטריצה המשמשת ליצירה ובדיקה של מעגלים ופרויקטים אלקטרוניים. זה בדרך כלל מורכב מלוח עם רשת של חורים המאפשרים להכניס רכיבים אלקטרוניים ולחברם יחד באמצעות חוטים.

Arduino היא מיקרו-בקר פופולרית שניתן לשלב בקלות במעגלי לוח מטריצה, מה שהופך אותה לבחירה אידיאלית ליצירה ובדיקה של פרויקטים.

עם breadboard של Arduino, משתמשים יכולים במהירות לבדוק את הרעיונות שלהם לפני תכנון לוח מעגל הקבוע. לוחות breadboard של ארדואינו מגיעים בגדלים ובתצורות שונות.

לוח ה- breadboard של Arduino הוא כלי פופולרי בשל קלות השימוש שלו ומחירו הזול.

חיבורים ותקשורת:

תקשורת טורית serial communication

במערכת שלנו יש את הרכיב raspberry pi 4 שהוא אמור לתקשר עם שאר הרכיבים שלנו כגון Arduino nano ו – ESP32 את התקשורת ביניהם נממש באמצעות serial communication או בעברית תקשורת טורית היא שיטת תקשורת המשתמשת בקו שידור אחד או שניים כדי לשלוח ולקבל נתונים, ונתונים אלה נשלחים ומתקבלים באופן רציף ביט אחד בכל פעם, בניגוד לתקשורת מקבילה, שבה נשלחים ביטים רבים בו-זמנית.

Raspberry Pi ו- Arduino יכולים לתקשר זה עם זה באמצעות תקשורת טורית. ניתן להשיג תקשורת זו באמצעות יציאת הטורי אוניברסלי (USB) בשני המכשירים.

ליתר דיוק, כשאנחנו משתמש ב-Serial עם Arduino ו-Raspberry Pi, אנחנו משתמש בפרוטוקול UART. UART פירושו "קליטה ושידור אסינכרוני אוניברסלי". בעיקרון מדובר בפרוטוקול רב מאסטר אסינכרוני המבוסס על תקשורת טורית, שיאפשר לך לתקשר בין 2 הרכיבים, יש ספריות שיטפלו בכל השכבות הנמוכות.

התקשורת הטורית בין הרכיבים מאוד פשוטה וקלה להבנה.

פרוטוקול TCP/IP

עוד תקשורת נוספת שאעשה שימוש בה היא פרוטוקול TCP הוא פרוטוקול תקשורת המספק מסירה אמינה, מסודרת ונבדקת שגיאות של נתונים בין יישומים הפועלים על התקנים ברשת. TCP הוא חלק מחבילת פרוטוקול האינטרנט IP והוא משמש יישומים רבים הדורשים תקשורת אמינה, כגון דואר אלקטרוני, העברת קבצים וגלישה באינטרנט.

הנתונים הנשלחים באמצעות TCP, הם מחולקים למנות, אשר נשלחות לאחר מכן דרך הרשת ליעד. TCP מבטיח שהמנות מתקבלות בסדר הנכון ושאף אחת מהן לא תאבד או פגומה במהלך השידור.

TCP משתמש בלחיצת יד תלת כיוונית כדי ליצור חיבור בין שתי נקודות קצה. השלב הראשון הוא חבילת ה-SYN (סנכרון), הנשלחת על ידי נקודת הקצה היוזמת לנקודת הקצה המקבלת. השלב השני הוא חבילת SYN-ACK (סינכרון-אישור), הנשלחת על ידי נקודת הקצה המקבלת לנקודת הקצה היזומה כדי לאשר את חבילת ה-SYN ולאשר שהיא מוכנה ליצור חיבור. השלב השלישי הוא חבילת ה-ACK (אישור), הנשלחת על ידי נקודת הקצה היוזמת לנקודת הקצה המקבלת כדי לאשר שהיא קיבלה את חבילת SYN-ACK.

אני אעשה שימוש בפרוטוקול TCP על מנת לקשר בין רכיב ה – raspberry pi 4 לתוכנה שמתריעה לי בזמן אמת שרכב שמספרו כך וכך חונה בחניה הפרטית שלי ואני יאשר את האזעקה או יגרום לאזעקה לא לעבוד עד שהרכב יזוז.

I2c

I2C הוא פרוטוקול תקשורת טורית בשימוש נרחב המאפשר למספר מעגלים משולבים לתקשר זה עם זה.

פרוטוקול I2C משתמש בארכיטקטורת מאסטר-slave, כאשר המכשיר הראשי יוזם ושולט בתקשורת, והתקני האחרים המגיבים לבקשות של המאסטר. שני קווי התקשורת המשמשים ב-I2C הם:

SDA – serial data line קו דו-כיווני זה משמש לשידור וקבלה של נתונים בין ה-Master וה-slave. הוא נושא הן את הכתובת של מכשיר היעד והן את הנתונים המועברים בפועל.

SCL - קו זה נושא אות שעון שנוצר על ידי המכשיר הראשי, המסנכרן את העברת הנתונים בין מכשירים. אות השעון קובע את התזמון לשידור וקליטה של נתונים.

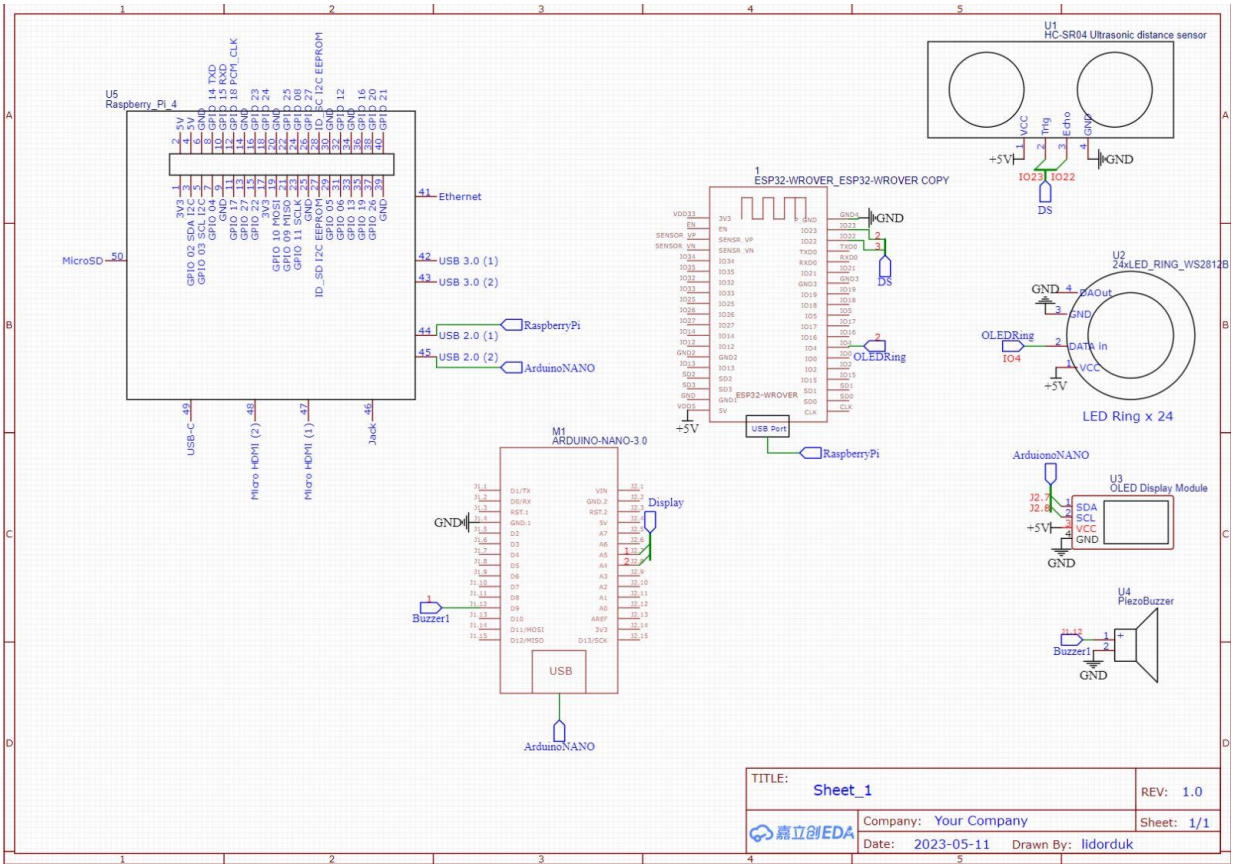
התכונות המאפיינים את פרוטוקול זה הם:

- לכל מכשיר המחובר לאפיק I2C מוקצית כתובת ייחודית המזהה אותו. זה מאפשר למכשיר הראשי לתקשר עם התקנים אחרים ספציפיים על ידי פנייה אליהם בנפרד.
- I2C תומך במספר התקנים באותו bus. לכל מכשיר יש כתובת ייחודית, והמאסטר יכול לבחור את מכשיר היעד לתקשורת על ידי פנייה אליו.
- I2C משתמשת במערכת רב מאסטר, מה שאומר שניתן לחבר מספר התקני מאסטר ל - bus. כדי למנוע התנגשויות כאשר מספר מאסטרים מנסים לתקשר בו זמנית, I2C משתמש במנגנון בוררות אפיק כדי לקבוע לאיזה מאסטר יש שליטה על ה - bus בזמן נתון.
- פרוטוקול I2C תומך במהירויות שעון שונות, המאפשר גמישות בקצב העברת הנתונים. מהירות השעון מוגדרת בדרך כלל על ידי מכשיר המאסטר וניתנת להתאמה בהתאם ליכולות של המכשירים המחוברים.

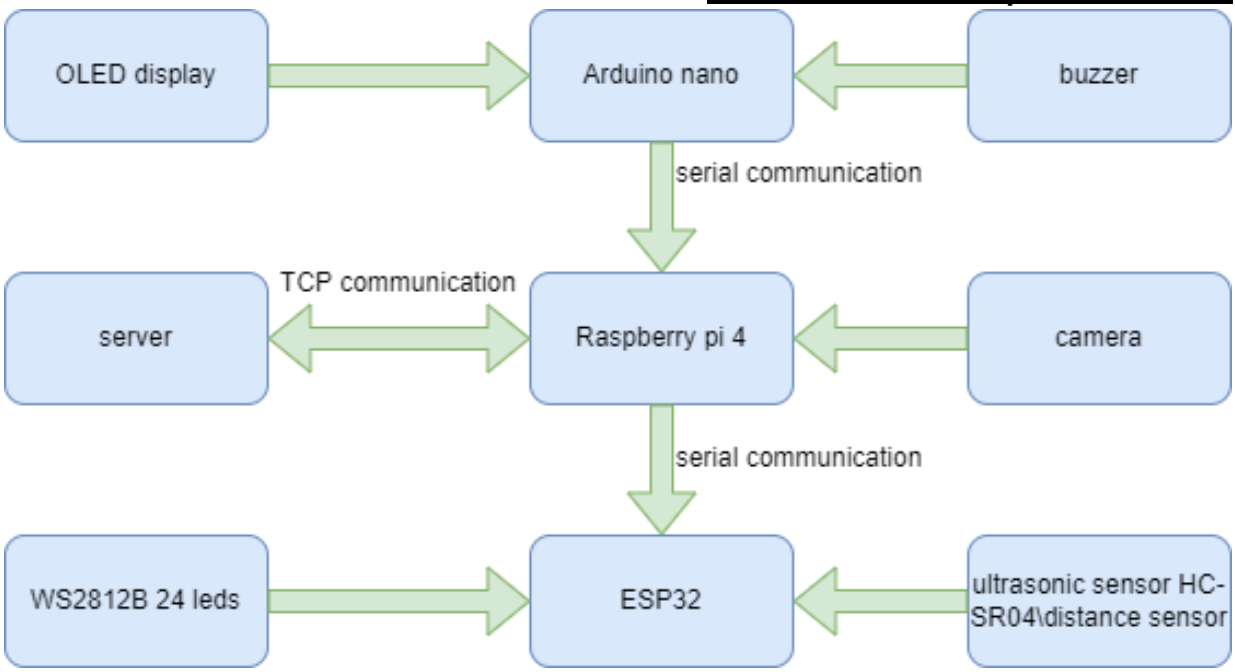
I2c נמצא בשימוש בכל העולם ביישומים כגוון חיישנים, צגים שהם נמצאים בשימוש אצלי במערכת והוא מספק דרך פשוטה ויעילה לחיבור התקנים במערכת ומאפשרת שילוב קל של רכיבים מיצרנים שונים.

שרטוטי חיבורים ומערכת:

שרטוט רכיבי האלקטרוניקה והחיבורים:



תרשים בלוקים של המערכת:



בעיות ופתרונות בפרויקט:

אחד הבעיות שנוצרו לי בתחילה הפרויקט הוא זיהוי לוחית הרישוי ופעונחה למספר כפי שציינתי עיבוד תמונה הוא בעיה מאוד מורכבת בתחום מדעי המחשב גרפיקה ממוחשבת ואליי להתגבר על הבעיה אחד הפתרונות לבעיה הזאת הם:

- לוחית הרישוי צריכה להיות גלויה ולא מוסתרת.
- לוחית הרישוי צריכה להיות תקינה הכוונה היא שלא יהיה מספר מחוק או לא ברור.
- תנאי מזג האויר יכולה להוות גורם לשיבוש לא נכון ופענוח לא תקין ולכן תנאי מזג האויר צריכה להיות נורמלית.
- הצילום של לוחית הרישוי צריכה להיות ישרה כלומר הרכב לא נמצא בזווית.

למשל התמונה הבאה עונה על כל התנאים:

על מנת לעבד את התמונה השתמשתי ב – opencv היא ספריית ראייה ממוחשבת ולמידת מכונה פופולרית בקוד פתוח המשמשת לעיבוד תמונה ווידאו בזמן אמת. OpenCV מספק מגוון רחב של פונקציות עיבוד תמונה ווידאו, כגון סינון תמונות, זיהוי תכונות, זיהוי אובייקטים, למידת מכונה. הוא כתוב ב-C++ וניתן להשתמש בו עם Python ,מה שהופך אותו לכלי רב-תכליתי עבור חוקרי ראייה ממוחשבת. OpenCV נמצא בשימוש נרחב בתעשיות כמו רובוטיקה, רכב ומדעי המחשב. עוד גורם שיעזור לנו להוציא את המספר שנמצאת בלוחית רישוי היא ספריית pytesseract היא מנוע קוד פתוח לזיהוי תווים אופטי (OCR) שפותח על ידי גוגל. זה מאפשר למפתחים להשתמש ב-Tesseract-OCR כדי לזהות טקסט בתמונות ולבצע פעולות OCR ביישומי Python שלהם.

מספקת ממשק קל לשימוש ל-Tesseract-OCR, הידוע בדיוק שלו בזיהוי טקסט ממקורות שונים, כולל מסמכים סרוקים, תמונות וסרטונים. עם Pytesseract, מפתחים יכולים לעבד תמונות מראש, להתאים פרמטרים של תמונה ולחלץ נתוני טקסט מתמונות בפורמטים שונים, כולל JPEG, PNG, GIF ו-BMP.

OCR פועל בכמה שלבים ובין היתר השלבים הם:

- ראשית אלינו לפני שמעבדים את התמונה אנחנו צריכים לשפר את איכות התמונה ולהפוך את הטקסט לקריא יותר. הוא כלול בין היתר בפעולות של שינוי גודל התמונה, הפחתת רעש של התמונה ועוד.
- לאחר שלב הראשון מערכת ה – OCR מזהה את האזורים בתמונה שבהם קיים טקסט. זה נעשה לעתים קרובות באמצעות טכניקות זיהוי קצוות, ניתוח קווים הנמצאים בתמונה או אלגוריתמים של למידת מכונה לאתר ולבודד אזורי טקסט.
- כאשר עיבוד התמונה עוסקת בעיבוד תמונה של כתב יד OCR יודע לחלק את האזורים בטקסט לתווים בודדים מה שיכול להועיל לפענוח הטקסט הנמצא בתמונה.
- כל תו מנותחת והתכונות הייחודיות שלה, כגון קווים, עקומות וצמתים, נשלפות. תכונות אלו משמשות ליצירת ייצוג של כל תו שניתן להשוות למסד נתונים של תווים ידועים.
- התכונות שחולצו מושוות למסד הנתונים של התווים כדי לקבוע את ההתאמה הסבירה ביותר עבור כל תו. תהליך זה כולל שימוש במודלים סטטיסטיים, אלגוריתמים של למידת מכונה או טכניקות התאמת תבניות לזיהוי התו.

חשוב לציין שהדיוק והביצועים של מערכות OCR, כולל pytesseract, יכולים להשתנות בהתאם לאיכות תמונת הקלט, שפת הטקסט וגורמים נוספים.

אלגוריתם שהעוסק בבעיית זיהוי לוחית רישוי ופעונחה:

לקיחת התמונה:

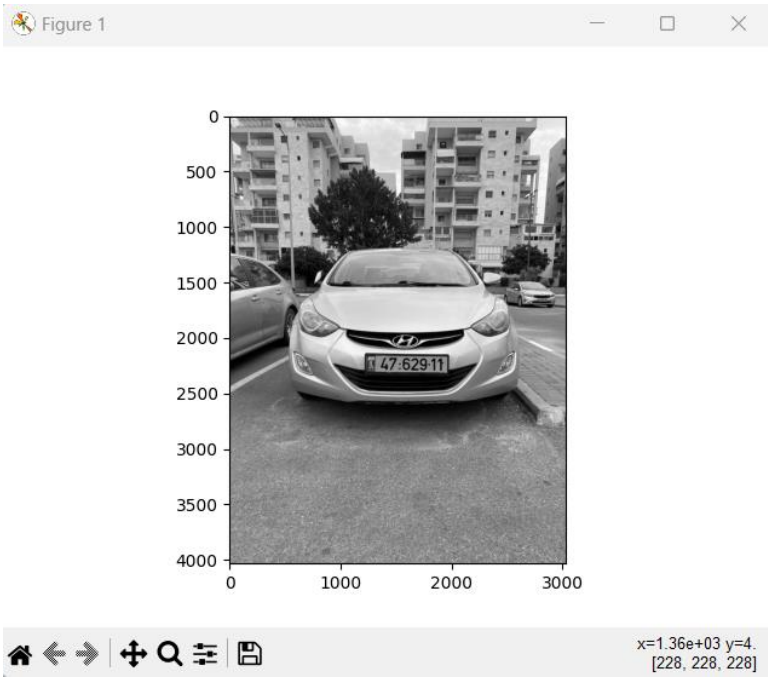
ראשית אלינו לקרוא את התמונה באמצעות cv2.imread הפרמטר שהוא יקבל זה את התמונה של הרכב שמתחיל לחנות

```
# Load the image
img = cv2.imread(path_of_car)
```

עיבוד מוקדם של התמונה:

לאחר כן על מנת להקל על עיבוד התמונה ושהיא תיקח פחות זמן מהצפוי נמיר את התמונה שקראנו לצבעי אפור כלומר נמיר את התמונה מ – RGB לגווני אפור.

```
# Convert the image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```



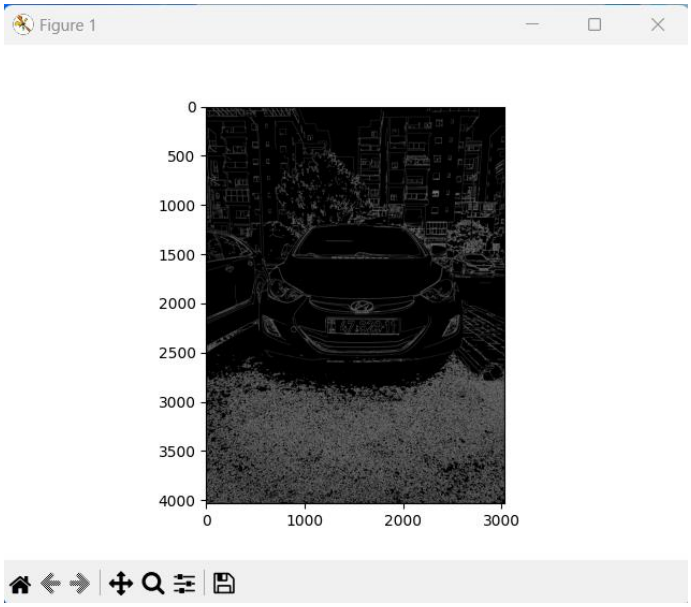
ואז נסיר את הרעש בתמונה על ידי פונקציה bilateralFilter.

```
# Apply bilateral filter to remove noise
blur = cv2.bilateralFilter(gray, 11, 17, 17)
```

זיהוי לוחית הרישוי על התמונה:

זיהוי לוחית הרישוי הוא תהליך קביעת החלק ברכב בעל תווי לוחית הרישוי. ביצוע זיהוי קצה נקרא לפונקציה cv2.Canny אשר תזהה אוטומטית את הקצוות בתמונה המעובדת מראש.

```
# Detect edges using Canny edge detection
edged = cv2.Canny(blur, 30, 200)
```

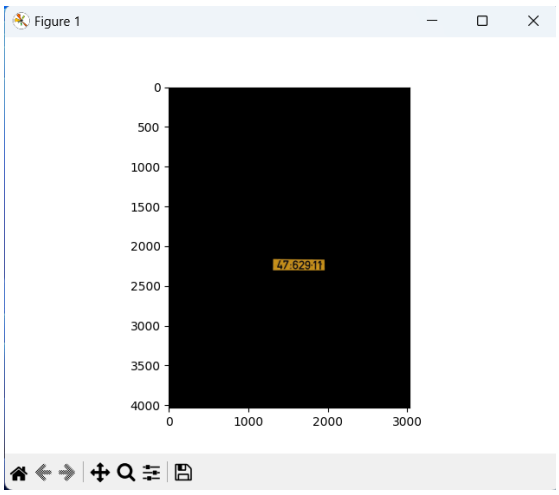


לאחר מכן אנחנו נקרא לפונקציה cv2.findcontours אחד מהפרמטרים שנביא לו הוא את הפלט שקיבלנו בצעד הקודם. שורת הקוד משרטטת את כל קווי התמונה שהיא מוצאת בתמונת המכונית באופן מובהק.

```
# Find contours in the image
contours, hierarchy = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

לאחר מציאת קווי התמונה אלינו לזהות את המועמדים הטובים ביותר ועל מנת לסנן את הקווים הלא קשורים ללוחית הרישוי נצטרך לסנן אותם על ידי מיון. נמיין את קווי התמונה על שטח מינימלי של 10. התעלם מהקווי התמונה שלמטה כי יש סיכוי נמוך יותר שהם יהיו קו המתאר של לוחית הרישוי.

```
# Sort the contours by area in descending order
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
```



יש עכשיו פחות קווי המתארות את לוחית הרישוי ממה שהיו בהתחלה. קווי שנשארו היחידים הם אלו המשוערים להכיל את לוחית הרישוי.

לבסוף, אלינו לעבור בלולאה על קווי המשוערים הממוינים ולקבוע מי מהם הוא לוחית המספר.

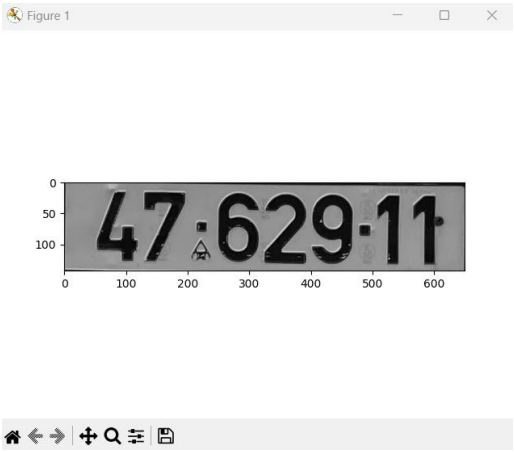
צור לולאה ללולאה על פני קווי המתארים את לוחית הרישוי. חפש את קו המתאר עם ארבע פינות, וקבע את היקפו ואת הקואורדינטות שלו. אחסן את התמונה של קו המתאר המכילה את לוחית הרישוי. לבסוף פענח את מה שרשום בלוחית רישוי באמצעות Pytessercat עם ה – config המתאים.

```
# Loop over the contours
for contour in contours:
    # Approximate the contour to a polygon
    perimeter = cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, 0.02 * perimeter, True)

    # If the polygon has four vertices, then it could be a license plate
    if len(approx) == 4:
        # Calculate the bounding box of the polygon
        x, y, w, h = cv2.boundingRect(approx)

        # Check if the bounding box has the aspect ratio of a license plate
        aspect_ratio = w / float(h)
        if aspect_ratio >= 2.5 and aspect_ratio <= 5.0:
            license_plate = gray[y:y + h, x:x + w]
            cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
            image_rgb = cv2.cvtColor(license_plate, cv2.COLOR_BGR2RGB)
            cv2.imshow("License Plate", image_rgb)
            custom_config = r'--oem 3 --psm 10 -c tessedit_char_whitelist=0123456789 --user-words israeli_number_plate_words.txt'
            text = pytesseract.image_to_string(license_plate, config=custom_config)
            text = text[:len(text) - 1]
            cv2.putText(img, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 2)
            img = cv2.resize(img, (1920, 1080), interpolation=cv2.INTER_AREA)
            cv2.imshow("License Plate Detection", img)
            # print("License Plate :", text)
            if text == my_license_plate:
                return (True, "")
            else:
```

```
            else:
                foreign_car_license_plate = text
                cv2.imwrite("license_plate.jpg", image_rgb)
                return (False, foreign_car_license_plate)
return (False, foreign_car_license_plate)
```



נספחים:

Raspberry pi 4 code:

```
import threading
import serial
import pytesseract
import cv2
import numpy as np
import os
import socket
from  matplotlib import  pyplot as plt
import imutils

pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files
(x86)\Tesseract-OCR\tesseract'

my_license_plate = "4762911"#My license plate
path_of_car = r'Images\10.jpg'#path of car to detection car and
license plate recognition
path_license_plate = "license_plate.jpg"

esp32_port = 'COM3' #The serial port of esp32
arduino_nano_port = 'COM4'#The serial port of arduino nano

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 12345  # The port used by the server

MODE = b"no active\n"

def Get_distance_from_ultrasonic_sensor(mode):
    ser = serial.Serial(port=esp32_port, baudrate=115200, timeout= 1)
    ser.reset_input_buffer()
    while True:
        if ser.in_waiting > 0:
            line1 = ser.readline().decode('utf-8').rstrip()
            converted_num = int(line1)
            if mode == 0:
                if converted_num < 100:
                    break
            if mode == 1:
                if converted_num > 100:
                    break
            print(converted_num)
    ser.close()

def Detect_car():
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
    classes = []
    with open("coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i - 1] for i in
net.getUnconnectedOutLayers()]
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    f.close()

    img = cv2.imread(path_of_car)
    height, width, channels = img.shape
```

```
    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0),
True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5 and class_id == 2:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
                return True
    return False
    # indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
    # font = cv2.FONT_HERSHEY_SIMPLEX
    # for i in range(len(boxes)):
    #     if i in indexes:
    #         x, y, w, h = boxes[i]
    #         label = str(classes[class_ids[i]])
    #         color = colors[class_ids[i]]
    #         cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
    #         cv2.putText(img, label, (x, y - 5), font, 1, color, 2)
    # cv2.imshow("Image", img)

def License_plate_recognition():
    foreign_car_license_plate = ""
    # Load the image
    img = cv2.imread(path_of_car)

    # Convert the image to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #gray = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Apply bilateral filter to remove noise
    blur = cv2.bilateralFilter(gray, 11, 17, 17)

    # Detect edges using Canny edge detection
    edged = cv2.Canny(blur, 30, 200)

    # Find contours in the image
    contours, hierarchy = cv2.findContours(edged.copy(),
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # Sort the contours by area in descending order
    contours = sorted(contours, key=cv2.contourArea,
reverse=True)[:10]

    # Initialize the license plate contour
    license_plate = None
```

```
    # Loop over the contours
    for contour in contours:
        # Approximate the contour to a polygon
        perimeter = cv2.arcLength(contour, True)
        approx = cv2.approxPolyDP(contour, 0.02 * perimeter, True)

        # If the polygon has four vertices, then it could be a
license plate
        if len(approx) == 4:
            # Calculate the bounding box of the polygon
            x, y, w, h = cv2.boundingRect(approx)

            # Check if the bounding box has the aspect ratio of a
license plate
            aspect_ratio = w / float(h)
            if aspect_ratio >= 2.5 and aspect_ratio <= 5.0:
                license_plate = gray[y:y + h, x:x + w]
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255,
0), 2)
                image_rgb = cv2.cvtColor(license_plate,
cv2.COLOR_BGR2RGB)
                cv2.imshow("License Plate", image_rgb)
                custom_config = r'--oem 3 --psm 10 -c
tessedit_char_whitelist=0123456789 --user-words
israeli_number_plate_words.txt'
                text = pytesseract.image_to_string(license_plate,
config=custom_config)
                text = text[:len(text) - 1]
                cv2.putText(img, text, (x, y - 5),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 2)
                img = cv2.resize(img, (1920, 1080),
interpolation=cv2.INTER_AREA)
                cv2.imshow("License Plate Detection", img)
                # print("License Plate :", text)
                if text == my_license_plate:
                    return (True, "")
                else:
                    foreign_car_license_plate = text
                    cv2.imwrite("license_plate.jpg", image_rgb)
                    return (False, foreign_car_license_plate)
    return (False, foreign_car_license_plate)

def Send_Image():
    client_socket = socket.socket()  # instantiate
    client_socket.connect((HOST, PORT))  # connect to the server
    message_to_server = "!image"
    client_socket.send(message_to_server.encode())

    file = open(path_license_plate, 'rb')

    image_data = file.read(2048)
    while image_data:
        client_socket.send(image_data)
        image_data = file.read(2048)
    file.close()
    client_socket.close()

def Dialogue(number_of_foreign_car):
    client_socket = socket.socket()  # instantiate
    client_socket.connect((HOST, PORT))  # connect to the server
```

```
    message_to_server1 = "!message" + number_of_foreign_car
    client_socket.send(message_to_server1.encode())

    message_from_server = client_socket.recv(2048).decode()
    print(message_from_server)

    client_socket.close()

    return message_from_server

def Alarm_system():
    ser = serial.Serial(port=arduino_nano_port, baudrate=9600,
timeout= 0.2)
    ser.reset_input_buffer()
    while True:
        ser.write(MODE)
        line = ser.readline().decode('utf-8').rstrip()

def main():
    global MODE
    while True:
        Get_distance_from_ultrasonic_sensor(0)
        flag_detect_car = Detect_car()
        if flag_detect_car == True:
            (flag_my_license_plate, foreign_car_license_plate) =
License_plate_recognition()
            if flag_my_license_plate == False:
                print("A foreign car is parked in your private
parking lot with its number:", foreign_car_license_plate)
                Send_Image()
                response_from_server =
Dialogue(foreign_car_license_plate)
                if response_from_server == "activate":
                    MODE = b"active\n"
                else:
                    MODE = b"no active\n"
            Get_distance_from_ultrasonic_sensor(1)
        else:
            print("No detect car")

if __name__ == '__main__':
    # t1 = threading.Thread(target= Alarm_system)
    # t1.start()
    # main()
    (flag_my_license_plate, foreign_car_license_plate) =
License_plate_recognition()
    if flag_my_license_plate == False:
        print("A foreign car is parked in your private parking lot
with its number:", foreign_car_license_plate)
    cv2.waitKey(0)
```

Server:

```
import threading
import tkinter
from tkinter import *
from PIL import Image, ImageTk
import os
import socket

BG_GRAY = "#ABB2B9"
BG_COLOR = "#17202A"
TEXT_COLOR = "#EAECEE"

FONT = "Helvetica 14"
FONT_BOLD = "Helvetica 13 bold"

path = "Image_from_client.jpg"
host = "127.0.0.1"  # as both code is running on same pc
port = 12345  # socket server port number

server_socket = socket.socket()  # instantiate

server_socket.bind((host, port))

server_socket.listen()

print("listening...")

class ChatApplication:

    def __init__(self, user):
        self.window = Tk()
        self.user = user
        self._setup_main_window()
        self.flag_activate_button = False
        self.flag_dectivate_button = False

    def run(self):
        self.window.mainloop()


    def close(self):
        self.window.destroy()

    def _setup_main_window(self):
        self.window.title("MY_GUI")
        self.window.resizable(width=False, height=False)
        self.window.configure(width=470, height=550, bg=BG_COLOR)

        # head label
        head_label = Label(self.window, bg=BG_COLOR, fg=TEXT_COLOR,
                           text=f'Welcome {self.user}',
font=FONT_BOLD, pady=10)

        head_label.place(width=465, height=50)

        # tiny divider
        line1 = Label(self.window, width=450, bg=BG_GRAY)
        line1.place(relwidth=1, rely=0.07, relheight=0.012)
        line2 = Label(self.window, width=450, bg=BG_GRAY)
        line2.place(relwidth=4, rely=0.139, relheight=0.012)
```

```
        # text widget
        self.text_widget = Text(self.window, width=20, height=2,
bg=BG_COLOR, fg=TEXT_COLOR,
                                font=FONT, padx=5, pady=5)
        self.text_widget.place(relheight=0.68, relwidth=1, rely=0.15)
        self.text_widget.configure(cursor="arrow", state=DISABLED)

        self.text_widget.configure(state=NORMAL)
        self.text_widget.configure(state=DISABLED)
        self.text_widget.see(END)

        # scroll bar
        scrollbar = Scrollbar(self.text_widget)
        scrollbar.place(relheight=1, relx=0.974)
        scrollbar.configure(command=self.text_widget.yview)

        # bottom label
        bottom_label = Label(self.window, bg=BG_GRAY, height=80)
        bottom_label.place(relwidth=1, rely=0.825)

        #Text
        text = Text(bottom_label, width=20, height=2, bg=BG_COLOR,
fg=TEXT_COLOR,font=FONT, padx=5, pady=5)
        text.place(relwidth=0.74, relheight=0.06, rely=0.008,
relx=0.011)
        text.configure(cursor="arrow", state=DISABLED)

        text.configure(state=NORMAL)
        text.configure(state=DISABLED)
        text.see(END)

        text.configure(state=NORMAL)
        text.insert(END, 'If you want to activate the alarm click
on')
        text.insert(END, '\n')
        text.insert(END , 'Activate if not click on Deactivate:')
        text.configure(state=DISABLED)
        text.see(END)

        # Activate button
        activate_button = Button(bottom_label, text="Activate",
font=FONT_BOLD, width=20, bg=BG_GRAY,
command=self.Activate_button_pressed)
        activate_button.place(relx=0.77, rely=0.008, relheight=0.025,
relwidth=0.22)

        # Deactivate button
        deactivate_button = Button(bottom_label, width=20,
bg=BG_GRAY,text="Deactivate",font=FONT_BOLD,
command=self.Deactivate_button_pressed)
        deactivate_button.place(relx=0.77, rely=0.047,
relheight=0.025, relwidth=0.22)

    def Activate_button_pressed(self):
        if os.path.exists(path):
            self.flag_activate_button = True
            os.remove(path)

    def Deactivate_button_pressed(self):
        if os.path.exists(path):
```

```
            self.flag_dectivate_button = True
            os.remove(path)

def Server():
    while True:
        conn, address = server_socket.accept()
        print("Connection from: " + str(address))
        message_from_client = conn.recv(2048).decode()
        if message_from_client == "!image":
            print(message_from_client)
            Get_image(conn)
        elif message_from_client[:8] == "!message":
            print(message_from_client[:8])
            msg = message_from_client[8:]
            print(msg)
            app_login.text_widget.configure(state=NORMAL)
            app_login.text_widget.insert(END, "A foreign vehicle has
been detected in your private")
            app_login.text_widget.insert(END, '\n')
            app_login.text_widget.insert(END, "parking lot with its
number " + msg)
            app_login.text_widget.insert(END, '\n')
            app_login.text_widget.configure(state=DISABLED)
            app_login.text_widget.see(END)
            while ((app_login.flag_dectivate_button == False) and
(app_login.flag_activate_button == False)):
                continue
            if app_login.flag_activate_button == True:
                message_to_client = "activate"
                conn.send(message_to_client.encode())
            if app_login.flag_dectivate_button == True:
                message_to_client = "deactivate"
                conn.send(message_to_client.encode())
            app_login.flag_activate_button = False
            app_login.flag_dectivate_button = False
        conn.close()

def Get_image(connection):
    file = open(path, "wb")
    image_chunk = connection.recv(2048)
    while image_chunk:
        file.write(image_chunk)
        image_chunk = connection.recv(2048)
    file.close()


# step 5 - start reviving msg from client in different thread!!
t1 = threading.Thread(target=Server)
t1.start()
app_login = ChatApplication("Server")
app_login.run()
```

ESP32 code:

```
#include <Adafruit_NeoPixel.h>
#ifdef _AVR_
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN        4 // On Trinket or Gemma, suggest changing this to 1

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 24 // Popular NeoPixel ring size

// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible values.
#define TRIG_PIN 23 // ESP32 pin GIOP23 connected to Ultrasonic
Sensor's TRIG pin
#define ECHO_PIN 22 // ESP32 pin GIOP22 connected to Ultrasonic
Sensor's ECHO pin
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define TRIG_PIN 23 // ESP32 pin GIOP23 connected to Ultrasonic
Sensor's TRIG pin
#define ECHO_PIN 22 // ESP32 pin GIOP22 connected to Ultrasonic
Sensor's ECHO pin
float duration_us, distance_cm;
int   stopdistance=40; //parking position from sensor (CENTIMETERS)
int startdistance=280;   //distance from sensor to begin scan as car
pulls in(CENTIMETERS)
int increment=((startdistance-stopdistance)/NUMPIXELS);

void setup() {
  // begin serial port
  Serial.begin (115200);

  // configure the trigger pin to output mode
  pinMode(TRIG_PIN, OUTPUT);
  // configure the echo pin to input mode
  pinMode(ECHO_PIN, INPUT);

  pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
}
int timer_reset = 0;
void loop() {
  // generate 10-microsecond pulse to TRIG pin
  pixels.setBrightness(32);
  digitalWrite(TRIG_PIN, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // measure duration of pulse from ECHO pin
  duration_us = pulseIn(ECHO_PIN, HIGH);

  // calculate the distance
  distance_cm = (float)duration_us * 0.034 / 2;
  int currentdistance = (int)distance_cm;
  //Serial.println(currentdistance);

  // print the value to Serial Monitor
  Serial.println(currentdistance);

 if(currentdistance > 240)
  {
    timer_reset = timer_reset  + 1;
    if(timer_reset > 25)
    {
      timer_reset = 0;
      colorWipe(pixels.Color(0, 0, 0) , NUMPIXELS);
    }
  }
  if(currentdistance <= stopdistance)
  {
    colorWipe(pixels.Color(255, 0, 0) , NUMPIXELS);
  }
  else if(currentdistance < 240)
  {
    for(int i = 1 ; i <= NUMPIXELS ; i++)
    {

      if(currentdistance < stopdistance + increment * i && i >= 7 && i
<= NUMPIXELS)
      {
        swap_to_green(i);
        break;
      }
      if(currentdistance<stopdistance+increment*i && i>=1 && i<=6)
      {
        swap_to_yellow(i);
        break;
      }

    }
  }
```

```
}

void colorWipe(uint32_t c , int j) {
  for(uint16_t i=0; i < j; i++) {
    pixels.setPixelColor(i, c);
    pixels.show();
  }
}

void swap_to_green(int i)
{
  colorWipe(pixels.Color(0, 255, 0) , NUMPIXELS - i + 1);
  for(int j = NUMPIXELS - i + 1; j < NUMPIXELS ; j++)
  {
    pixels.setPixelColor(j, pixels.Color(0, 0, 0));
    pixels.show();
  }
}

void swap_to_yellow(int i)
{
  for (int   j = 0; j < NUMPIXELS - 6; j++)
  {
    pixels.setPixelColor(j, pixels.Color(0, 255, 0));
    pixels.show();
  }
  for (int   c = NUMPIXELS - 6; c < NUMPIXELS - i + 1; c++)
  {
    pixels.setPixelColor(c, pixels.Color(255, 255, 0));
    pixels.show();
  }
  for (int   a = NUMPIXELS - i + 1; a < NUMPIXELS; a++)
  {
    pixels.setPixelColor(a, pixels.Color(0, 0, 0));
    pixels.show();
  }
}
```

Arduino nano code:

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 display(4);

char message[]="You are parking in private parking, please move your
car";
int x, minX;
const int buzzer = 9;
unsigned long prevTime_activate = millis();
unsigned long prevTime_no_activate = millis();

void setup()
{
  Serial.begin(9600);
  pinMode(buzzer , OUTPUT);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setTextWrap(false);
  display.clearDisplay();
  display.display();
  x = display.width();
  minX = -12 * strlen(message);  // 12 = 6 pixels/character * text size
2
}

void loop()
{
  if (Serial.available() > 0)
  {
    String data = Serial.readStringUntil('\n');
    if (data == "active")
    {
      Lcd_Show_Text();
      Buzzer_alarm_active();
      Buzzer_alarm_no_active();
    }
    else
    {
      noTone(buzzer);
      display.clearDisplay();
      display.display();
    }
  }
  // Lcd_Show_Text();
  // Buzzer_alarm_active();
```

```
  // Buzzer_alarm_no_active();
}

void Buzzer_alarm_active()
{
  unsigned long currentTime = millis();
  if (currentTime - prevTime_activate > 1000)
  {
    tone(buzzer , 10000);
    prevTime_activate = currentTime;
  }
}

void Buzzer_alarm_no_active()
{
  unsigned long currentTime = millis();
  if (currentTime - prevTime_no_activate > 2000)
  {
    noTone(buzzer);
    prevTime_no_activate = currentTime;
  }
}

void Lcd_Show_Text()
{
  display.clearDisplay();
  display.setCursor(0,0);
  display.setTextSize(2);
  display.setCursor(x,10);
  display.print(message);
  display.setCursor(x,28);
  display.setCursor(x,38);
  display.setCursor(x,48);
  display.display();
  x=x-8; // scroll speed, make more positive to slow down the scroll
  if(x < minX) x= display.width();
}
```