בית ספר אורט ע"ש י. רבין" גן-יבנה

הלסט - alset

שם בית ספר: אורט רבין גן יבנה

שם המנחה: משה זזק

שם החלופה: הגנת סייבר

מגישים: נתנאל חכמון-324199504 אייל גלם-325754919

תאריך הגשה:8/6/2021

בית ספר אורט ע"ש י. רבין" גן-יבנה

תוכן עניינים

בית ספר אורט ע"ש י. רבין" גן-יבנה

תקציר

הפרויקט שלנו הוא דגם של מכונית אוטונומית המשלבת אלקטרוניקה,פיזיקה,מכניקה ותוכנה. מטרת הפרויקט היא לזהות את השוליים של הכביש ולתת פקודה לרכב לנוע בכיוון של הכביש בלי סטיה מהמסלול.

בספר זה אנו נסביר על הפרויקט , על אך הוא עובד , על האלקטרוניקה שלו, על המחשבה של איך ליבנות פיזית את הרובוט, על כל החישובים הפיזיקליים, על האלגוריתם של המציית השוליים של הכביש, על התקשורת בין החלקים השונים ועל הסיבה שבגללה בחרנו בפרויקט זה.

בית ספר אורט ע"ש י. רבין" גן-יבנה

מבוא

ספר זה מורכב מכמה חלקים חלק ראשון תקציר אשר מטרתו היא להסביר בקצרה על הספר ועל הפרויקט.

חלק שני הוא המבוא שבחלק זה אנו מרחיבים את אמירותיו בתקציר בנוסף לכך אנו מוסיפים את תהליך המחקר של הפרויקט,האתגרים שהיו לנו בפרויקט ופתרונות לבעית הללו.

חלק שלישי הוא הארכיטקטורה ששם אנו מסבירים על הפרויקט עצמו לעומק, מציגים את הפתרון הסופי שבו השתמשו למציאת הבעיה שמוזכרת במבוא, את הארכיטקטורה של הפתרון, בחלק זה אנו גם מסבירים גם על כל חלק אלקטרוני ברובוט ומה תפקידו, שרטוטים של האלקטרוניקה , נסביר גם על הקוד שכתבנו למציאת השוליים, נסביר על הספריות שהשתמשו בהם, נסביר על הקוד של הרובוט ועל הקוד השולט ומתרגם את המידע מהקוד של המציאת שולים, נסביר על התקשורת בין חלקים שונים של הרובוט כגון(התקשורת של רוזברי פי עם הארדרואינו) וכולי..

חלק רביעי הוא מדריך למשתמש ששם נדבר למשתבש איך להשתמש בפרויקט ואיך להפעיל ולהדליק את הפרויקט, נדבר על אך המשתמש יודע אם הפרויקט עובד כרעוי, נדבר על האפליקציה שמשתמש יצטרך להפעיל, נדבר על איך להתחבר לרובוט ולהזיז אותו ידנית ונדבר על האינדיקציות שהמשתמש צריך לשים לב אליהם.

חלק חמישי הוא מדריך למפתח שבו אנו נדבר הקבצים של הפרויקט על מיקומם בפרויקט, הסברים של פונקציות ומשתנים הנמצאים בקוד של הפרויקט, הסבר על איך לשנות או להוסיף דברים למעגל החשמלי של הרובוט, על המגבלות של הפרויקט והסבר על איך לערוך קבצים של הפרויקט.

מטרת הפרויקט היא לחקור ולהבין מקרוב את רעיון המכונית האוטונומית, ולאחר מכן ליצור דגם מוקטן של מכונית או רובוט אשר יוכל לפעול בצורה אוטונומית(חלקית) בהתאם לסביבה שנתנו לרובוט ללא עזרה מהמפעיל. חשבנו על פרויקט זה בגלל שהשימוש של מכוניות אוטונומיות הולך וגדל ורצינו לדעת איך הם עובדות ומה האתגרים שהם צריכים לעבור מבחינה של תוכנה ומבחינה של חשמל ואלקטרוניקה.

הרעיון של מכוניות אוטונומיות נמצא בשוק כבר מספר שנים כאשר מטרתם למזער את השתתפות האדם בנהיגה ובכך למזער את כמות התאונות וליצור כלי נוח ומדויק יותר לשימוש. מכוניות אוטונומיות כיום קיימות 6 רמות מרמה 0 לרמה 5 אך בשוק קיימות רק מכוניות עד רמה 4 כאשר המשמעות שלה היא שהנהג יכול אף לישון במהלך הנהיגה והמכונית תוכל לפעול לבד בשעת סכנה אך מגבלה חוקית לא מאפשרת למכוניות אלו לפעול בכל מקום ומחייבות אפשרות התערבות אנושית בנהיגה.

רמה 5 היא הרמה הגבוהה ביותר של אוטונומיות שניתן להעניק לרכב ומשמעותה שהמכונית לא צריכה שום התערבות אנושית אף אל פי השוק הרחב רכבים אלו עדיין לא נמצאים חברות רבות מנסות לעבור אתגר זה ואחת מהן היא טסלה אשר היום מייצגת אחת מהחברות המובילות ביותר בתחום טכנולוגיית המכוניות האוטונומיות.

הפרויקט שאנו בנינו אמור לדמות מכונית אוטונומית שמזה את השוליים של הכביש ובצורה אוטונומית שולטת ברכב ומתקנת אותו ומחזירה אותו למסלול. חלק מהפרויקט הוא התוכנה וחלקו השני הוא החומרה והאלקטרוניקה והפתרון שהוא אמור לתת זה חקירה של מכוניות אוטונומיות ועיבוד תמונה.

האתגרים הניצבים בפנינו הם כך:

1) איך לזהות את השוליים של הכביש מוידאו חיי.
2) איך להעביר את המידה הזה לרובוט ולתרגם אותו לפעולה שהרובוט יעשה ויגרום לו לזוז.
3) איך להשאיר את הרובוט במהירות קבוע.
4) איך ליבנות את הרובוט.

הפתרונות לבעיות הללו:

1] שימוש בעיבוד תמונה והספריות אשר נקראות opencv שמטרתם היא להקל על המתכנתים שהם לא יתחילו מאפס אלה בספריה בנויות פונקציות אשר נותנות לנו לדאוג רק ליצירת האלגוריתם והפונקציות של עיבוד התמונה בלי להתעסק במתמטיקה והתרגום וידאו לשפה שהמחשב יבין. ספריה זו ניתנת לשימוש בסוגי שפות תכנות שונות אנחנו בחרנו להשתמש בפייתון בגלל ההיכרות שלנו עם השפה. המחשב אשר עושה את החישובים הללו ברובוט הוא ה Raspberry PI או בקיצור

RP אנחנו נשתמש בקיצור זה בספר זה במקום ב שם Raspberry PI.

2] בשביל להעביר את מידה זה אנו השתמשנו בRP וב arduino שהוא מיקרו-בקר בעל מעגל מודפס יחיד אשר מטרתה ליצור סביבה נוחה וזולה לפיתוח פרויקטים המשלבים תוכנה עם רכיבי אלקטרוניקה. והעברנו מידה בכך שחיברנו כבל בין ה RP וה arduino והעברנו את המידה עם הפורטים והפרוטוקולים שלהם. מי שתרגם את זה הוא ה arduino והוא הפך את הקוד למידע אשר המנועים והבקרים יכולים להבין.

3] כדי שהרובוט ישאר במהירות קבוע אנחנו משתמשים בפונקציות אשר משתמשות בשיטת הpid אשר לוקחת מהירות ושגיעות של הרובוט ומתקנת אותה כדי שהרובוט לא ינוע במהירות אשר אנו לא רוצים.

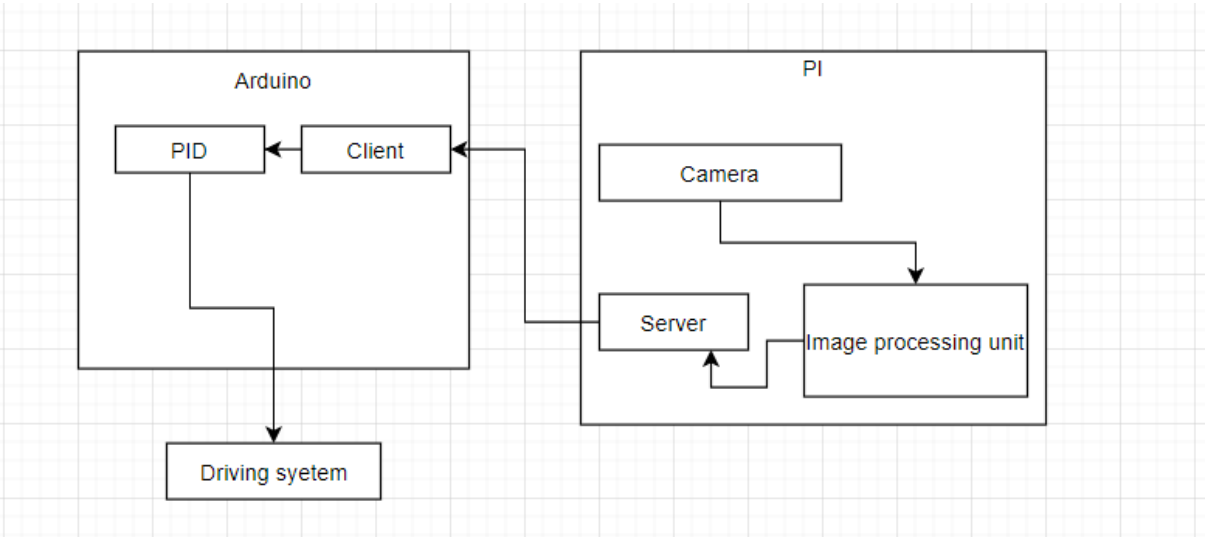
4] בשביל לבנות את הרובוט אנחנו שרטטנו סקיצות של הלוחות חשמל והשתמשנו בסימולציות כדי לבדוק אם האלקטרוניקה עובדת לפני בנייתה. אחר כך אנחנו הרכבנו את האלקטרוניקה ובדקנו שזה עובד ואז חיברנו את האלקטרוניקה לשלדה של הרובוט.

בית ספר אורט ע"ש י. רבין" גן-יבנה

ארכיטקטורה

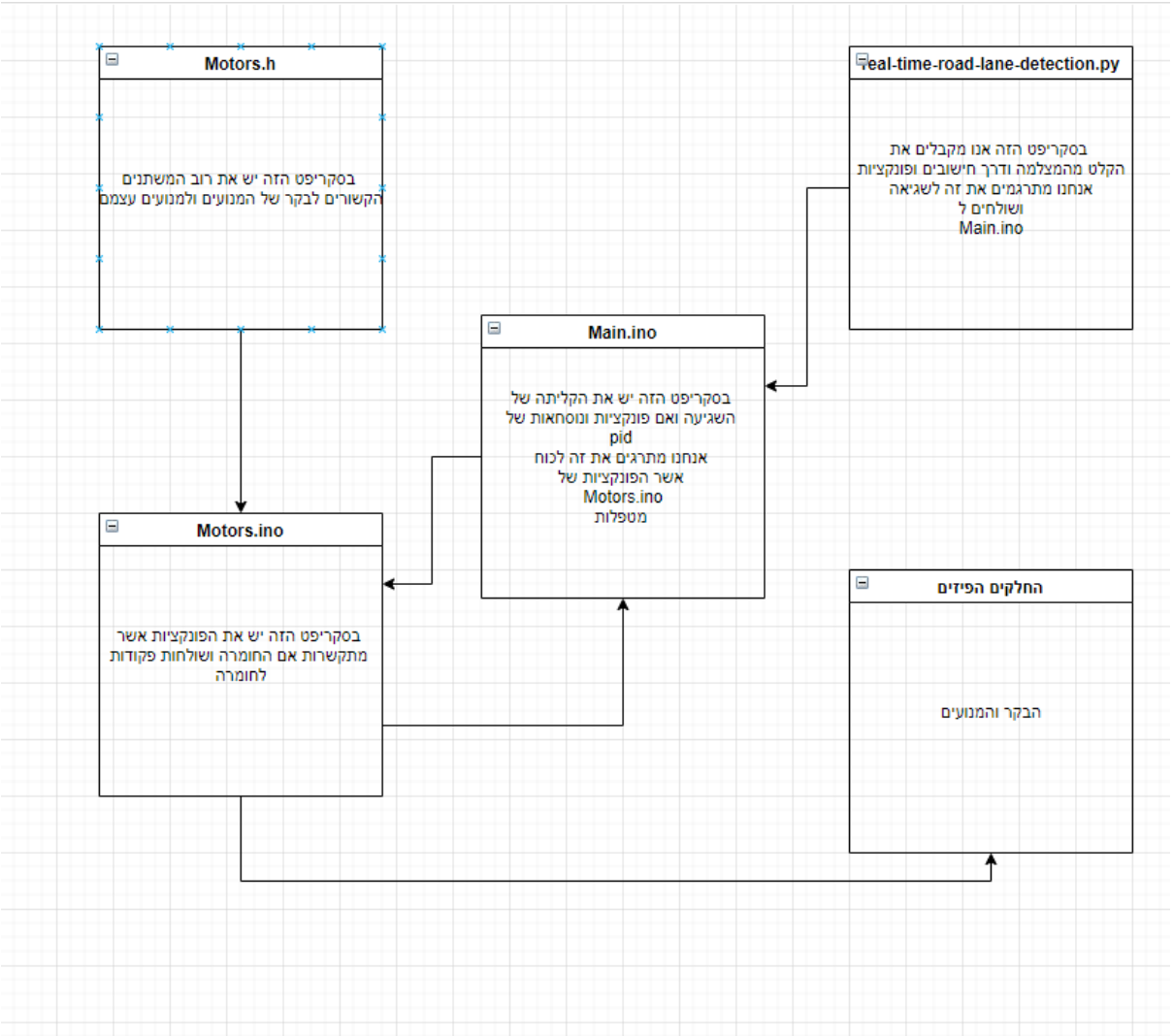
הפרויקט שלנו מביא את הפתרון שמוזכרת במבוא לידי ביטוי, בכך מפחית מצבי נהיגה מסוכנים על ידי שמירת נתיבים. הרובוט מקבל תמונה בעזרת המצלמה והיא עוברת עיבוד בעזרת הקוד שכתבנו בתוכנה . לאחר מכן הוא מקבל החלטה האם הרובוט נוסע במצב הנכון ובתוך הנתיבים , אם לא הרובוט אמור לסטות ימינה או שמאלה בהתאם למה שהוא רואה . בכל רגע נתון מתקבלת תמונה חדשה ואם הוא רואה איזה שינוי הוא יודע מה לעשות. לדוגמא, הרובוט יודע לזהות פניות חדות ולהתאים את המהירות לפי זה.

תהליך העבודה על הפרויקט התחיל ממספר שרטוטים בסיסיים אשר עזרנו לנו להבין כיצד לעצב את הקוד. מכיוון שהפרויקט שלנו הוא פיזי ומכיל בתוכו רכיבים אלקטרוניים התחלנו עם שרטוט אשר מכיל את כל הרכיבים אשר יהיו חלק מהפרויקט שלנו. חלק זה כולל את הרכיבים האלקטרוניים שרטוט מופשט של הRP שמפעיל את הרובוט ושרטוט של מבט על של הפרויקט.



(השרטוט הראשוני של מבט על של פרויקט)

לאחר מכן התחלנו לפרק כל חלק מהרכיבים המופשטים למספר סקריפטים שיבצעו חלקים שונים מהעבודה הכוללת. תרשים זה היה יותר מפורט, הוא כלל הסבר על כל חלק ויכולנו להשתמש בו כדי לתכנן מוחשית כל אחד מהסקריפטים בהם תכננו להשתמש בפרויקט זה.

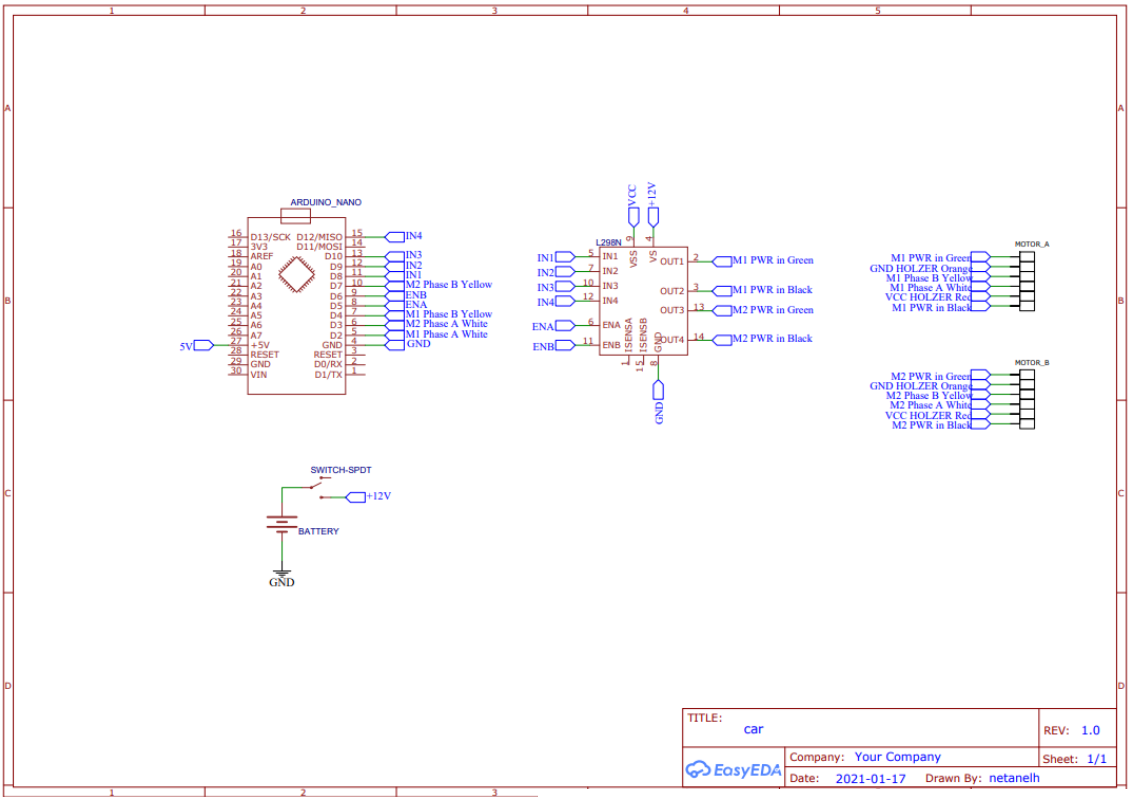


בית ספר אורט ע"ש י. רבין" גן-יבנה

האלקטרוניקה של הפרויקט:

הרכיבים האלקטרונים שהשתמשנו בהם בפרויקט הם:

1. רזברי פי 3- מחשב - לוח יחיד בגודל כרטיס אשראי שבעזרתו בפרויקט אנו מעבדים את התמונה .
2. בקר -לוח חשמלי שמטרתו היא לקבל נתונים ולהפוך אותם לזרמים חשמליים ששולטים במנוע
3. מצלמה - בעזרת המצלמה מחוברת אל הרזברי פי אנחנו מזהים את הנתיבים.
4. סוללות נטענות - בעזרת הסוללות אנחנו יכולים להפעיל את המנועים והמאוורר.
5. סוללה נטענת נוספת- הסוללה מספקת חשמל על הרזברי פי
6. מטריצה- מחברת בין כל האלקטרוניקה בפרויקט.
7. ארדואינו- הוא מיקרו-בקר בעל מעגל מודפס יחיד, עם סביבת פיתוח משולבת ברישיון קוד פתוח, אשר מטרתה ליצור סביבה נוחה וזולה לפיתוח פרויקטים המשלבים תוכנה עם רכיבי אלקטרוניקה.



(שרטוט של מבט על של האלקטרוניקה פרויקט)

בית ספר אורט ע"ש י. רבין" גן-יבנה

היחידות המרכיבות את הפרויקט

Road_line_detection:

יחידה זו מקבלת תמונה סטטית של הכביש ומוצאת את גבולות הכביש.
גבולות הכביש נמצאים בעזרת מספר פילטרים שהתמונה עוברת דרכם.
ראשית אנחנו מסננים את כל האיברים בתמונה שהם אינם אפורים(הסימון שלנו לגבולות הכביש) ,לאחר מכן אנחנו משתמשים בטכנולוגיית detection Edge של ספריית OPENCV של פייטון כדי למצוא את כל האיברים האפורים בתמונה המייצגים קו ישר או חלק מקו ישר.
אחר כך אנו קובעים לתמונה reign of interest ומסננים את כל מה שלא נמצא בתוך זה.

לאחר שמצאנו את גבולות הכביש אנו נבדוק את המיקום היחסי של המכונית ביחס לכביש על ידי החישוב נקודה שהיא אמצע הכביש על ידי נוסחה הנקראת distance ונחסר ממנו את הנקודה שהיא האמצע של המכונית.
כאשר מצאנו את המיקום היחסי אנחנו משתמשים בו לשני פונקציות שונות אחת אשר הופכת את המיקום היחסי למשתנה שמשמעותו היא התיקון שהמכונית צריך לעשות במהירות הסיבובית והמשתנה הוא כל מספר (גם מספר ממשי) בין 1- ל 1.
והשניה היא פונקציה אשר קובעת את המהירות הקווית של המכונית והיא עובדת כך שאם אנו נמצאים באמצע של הכביש או שאנו קרובים לאמצאה אז המשתנה של המהירות הקווית היה 10 ואם המכונית לא רואה לפחות קו אחד אז המשתנה של המהירות הקווית היא 0.

<u>תקשורת בין ארדואינו ל rp:</u>

בחלק זה יש את הצד של הארדרואינו ואת הצד של ה rp והמטרה היא ליצור תקשורת פרוטוקול רציפים ומהירים.

צד של הארדואינו:

קלט-מהירות קווית ומהירות סיבובית(בתור משתנה אחד שהארדואינו מתרגם על ידי פונקציה שבנינו).

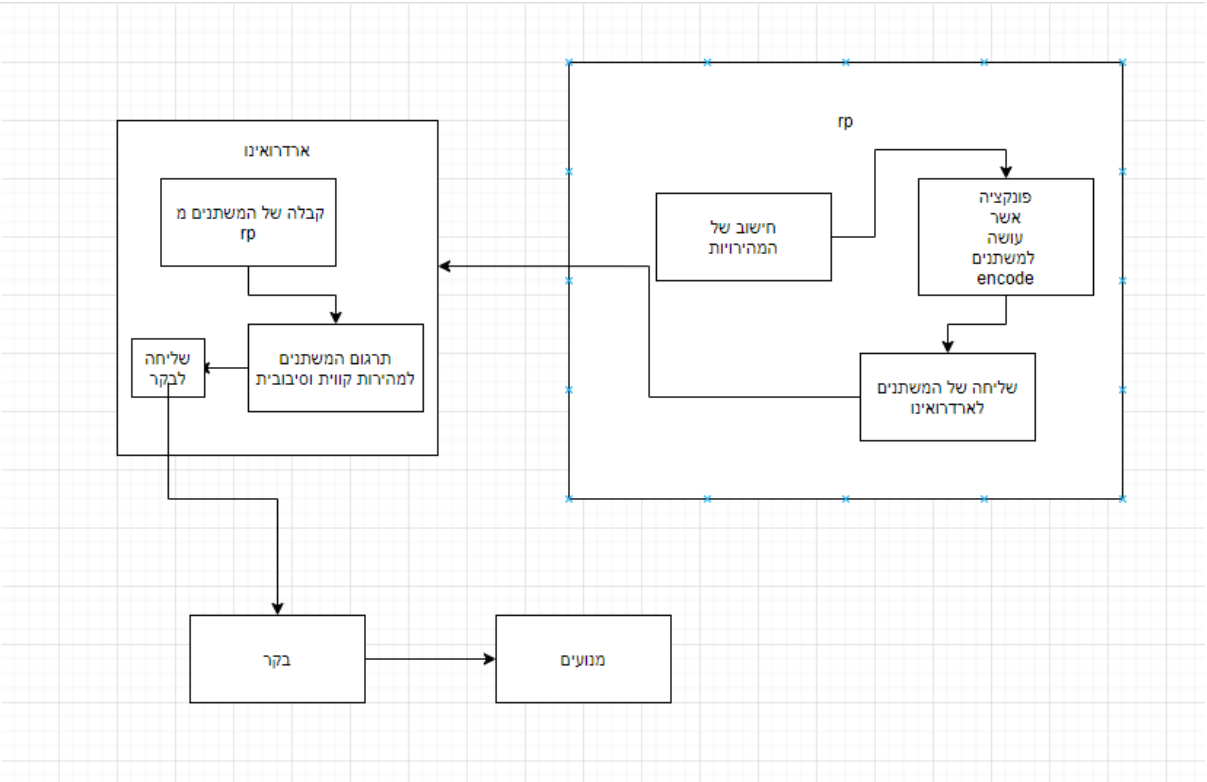
פלט-אין.

צד של הrp:

קלט-אין.

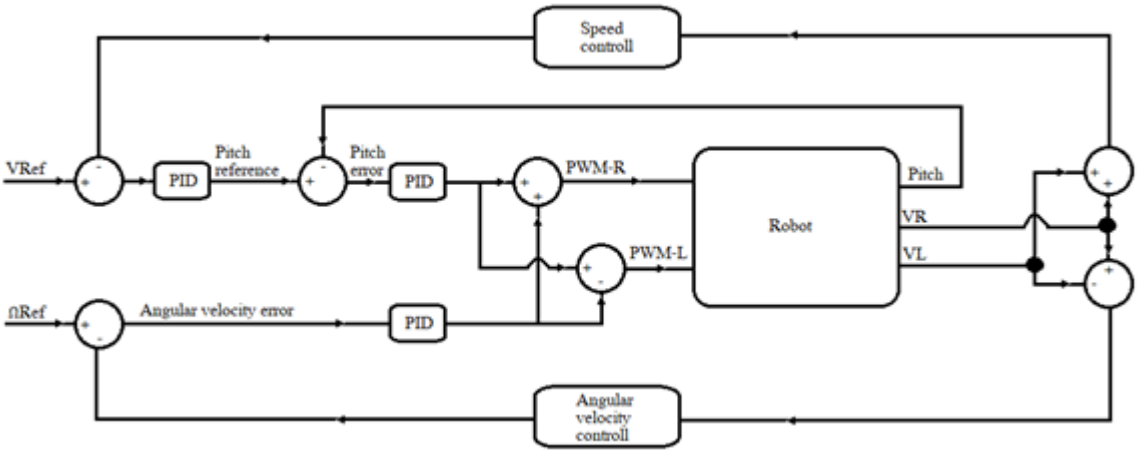
פלט- מהירות קווית ומהירות סיבובית(בתור משתנה אחד שהrp עושה לו encode).

לאחר כך אנו יצרתו תרשים המתאר את התקשורת ביניהם:



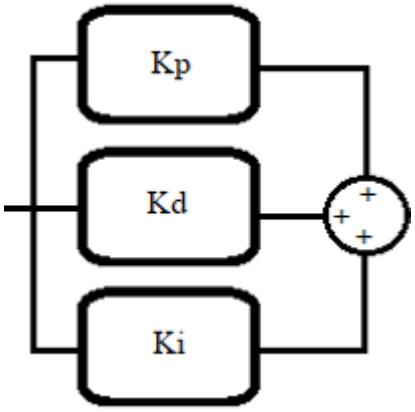
בית ספר אורט ע"ש י. רבין" גן-יבנה

PID-Proportional-Integral-Derivative



PID זה בעצם דרך לשלוט על המהירות של המנועים והמטרה שלה היא מניעת שגיאות וסטיות שהרובוט יכול לבצעה. בPID אנו לוקחים את השגיאה מחשבים אותה ומחברים אותה אם הנגזרת שלה והאינטגרל שלה.

אנו משתמשים במערכת כדי שהרובוט לא יסטה מהמסלול בכך שהמהירויות שאנו אומרים לו הם לא נכונות בגלל שיש שגיאה בגלל כל מיני פרמטרים כמו זמן מרחק וחיכוך. את השגיאה אנו מקבלים מ החלק של ה**Road_line_detection** והארנרואינו מחשב מתקן את השגיאה ושולח את המהירות הסופית לרובוט. בפרויקט שלנו יש שני PID אחד אשר שייך למהירות הקווית והשני אשר שייך למהירות סיבובית.



בית ספר אורט ע"ש י. רבין" גן-יבנה

מדריך למשתמש

ראשית לפני ההפעלה יש לוודא שה RP מחובר למטען נייד ושהארדרואינו מחובר ל RP ויש לוודא שהמצלמה מחוברת והרובוט נמצא בתוך המסלול.

כדי להפעיל את הרובוט צריך להדליק את המטען הנייד ולחבר את הסוללות האדומות.

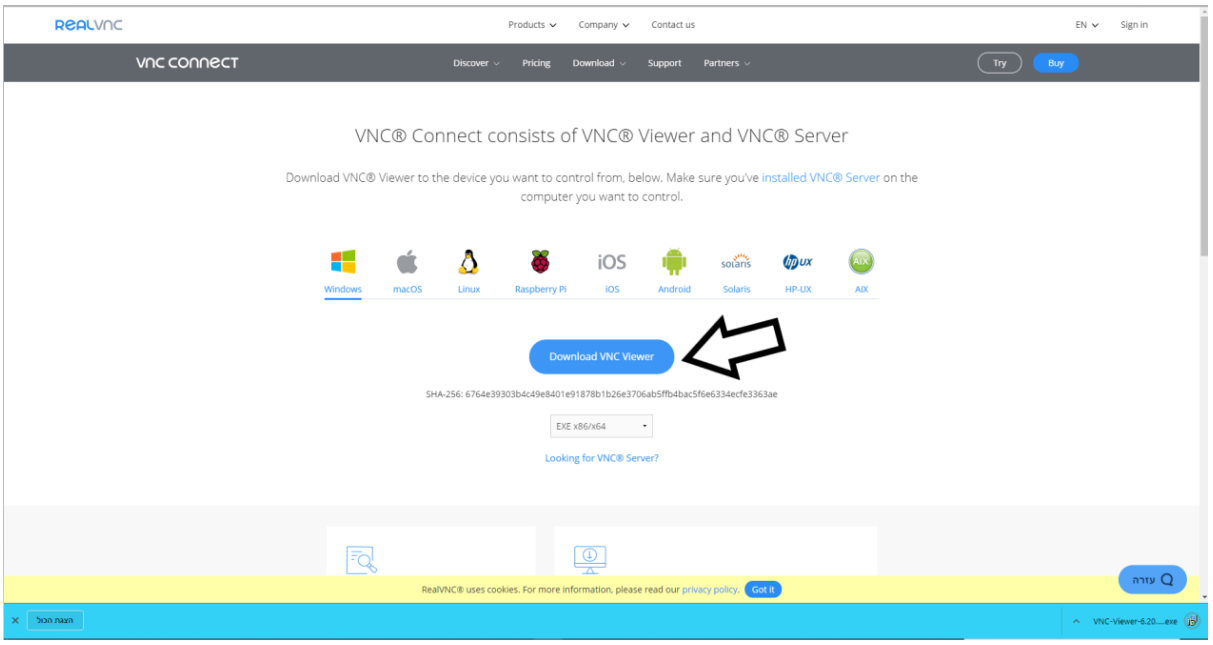
פעולה זו תדליק את ה RP הארדרואינו ואת המנועים.

בשביל להריץ את הקוד של הפרויקט נצטרך לשלוט על ה RP מרחוק נוכל לעשות זאת נצטרך את האפליקציה VNC Viewer.

מדריך להורדת ותפעול VNC Viewer

ראשית נכנס לאתר שלהם מהקישור הבא:

https://www.realvnc.com/en/connect/download/viewer/

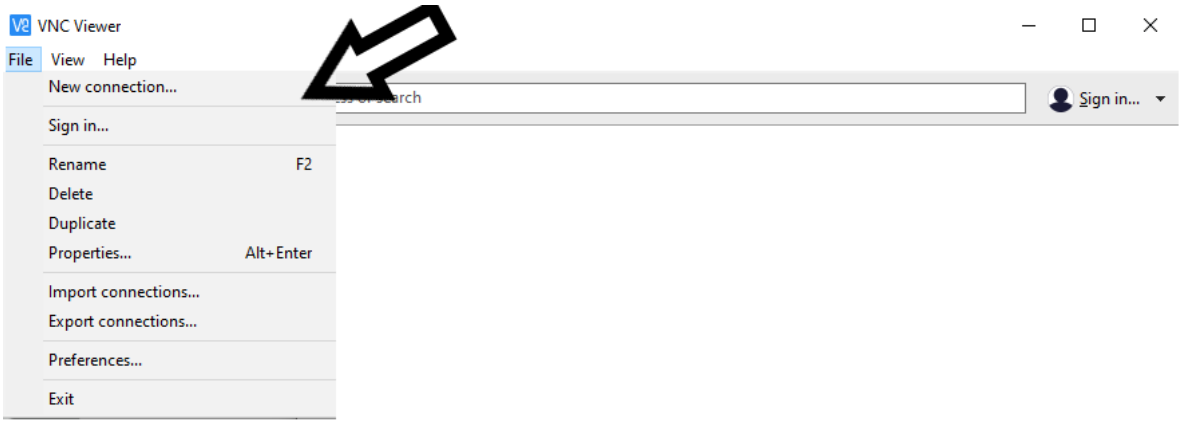


ונוריד את האפליקציה כך:

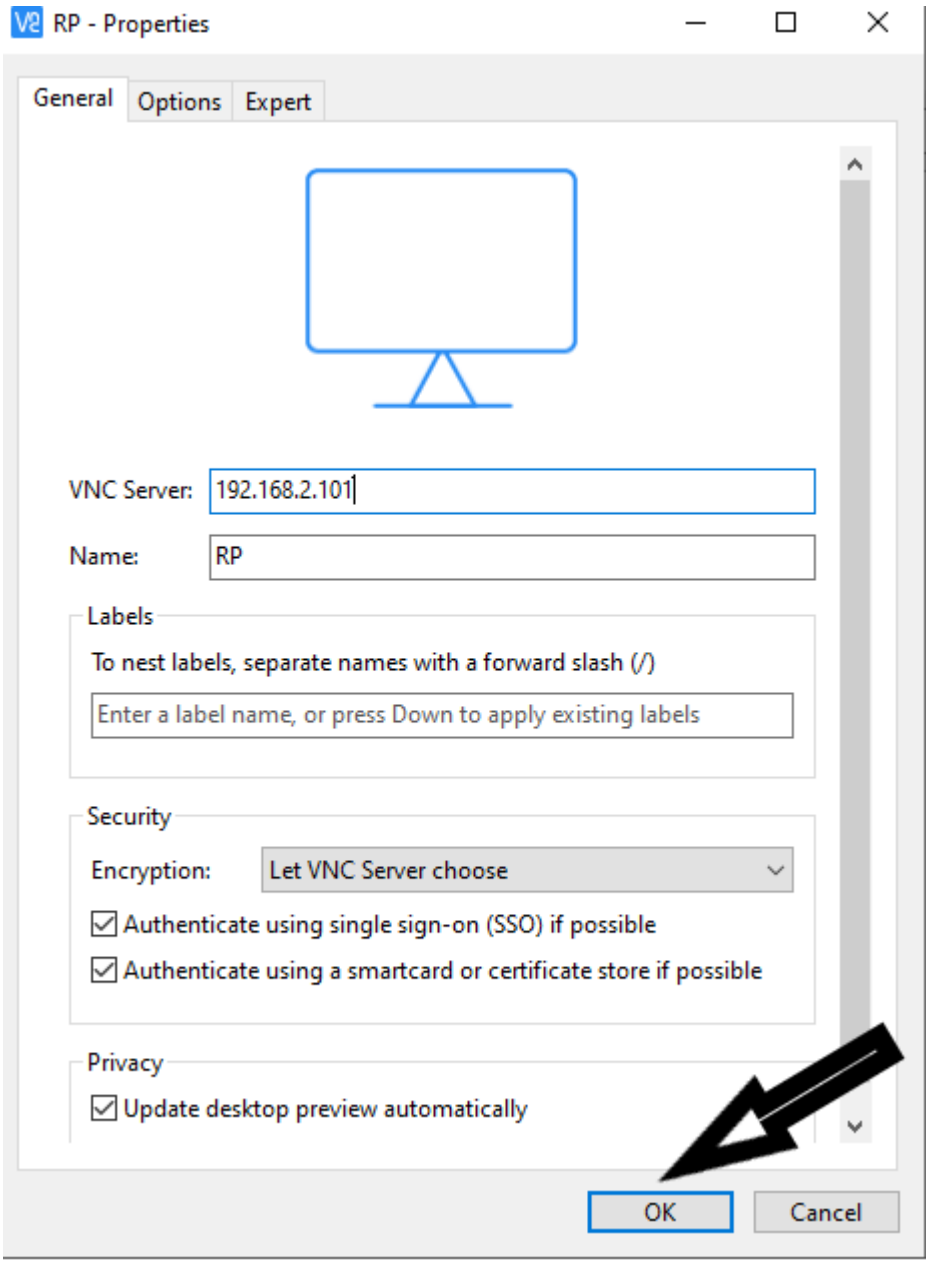
אחרי זה מתקינים את התוכנה עוקבים אחרי ההנחיות הבאות:

בית ספר אורט ע"ש י. רבין" גן-יבנה

פותחים תקשורת חדשה:



מכניסים את הנתונים הללו ולוחצים ok.



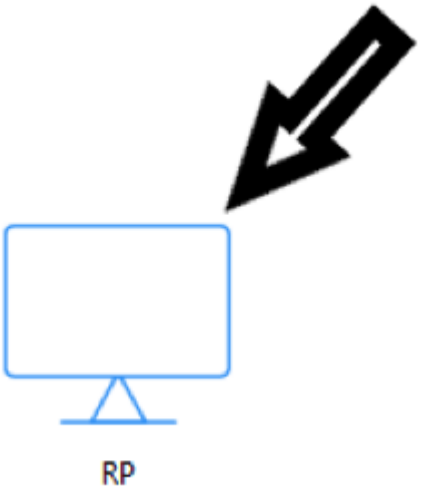
בית ספר אורט ע"ש י. רבין" גן-יבנה

לוחצים לחיצה שמאלית

כפולה על זה:

ומכניסים בשם משתמש:rp

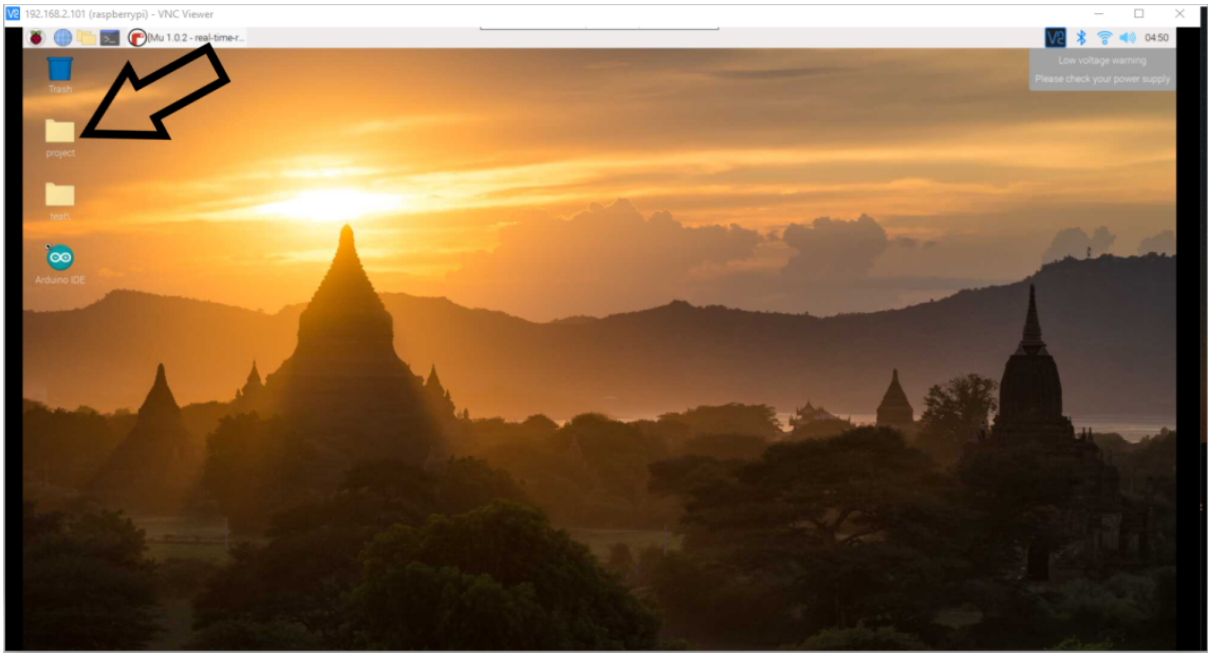
ובסיסמה:moshe 1234



מדריך להפעלת

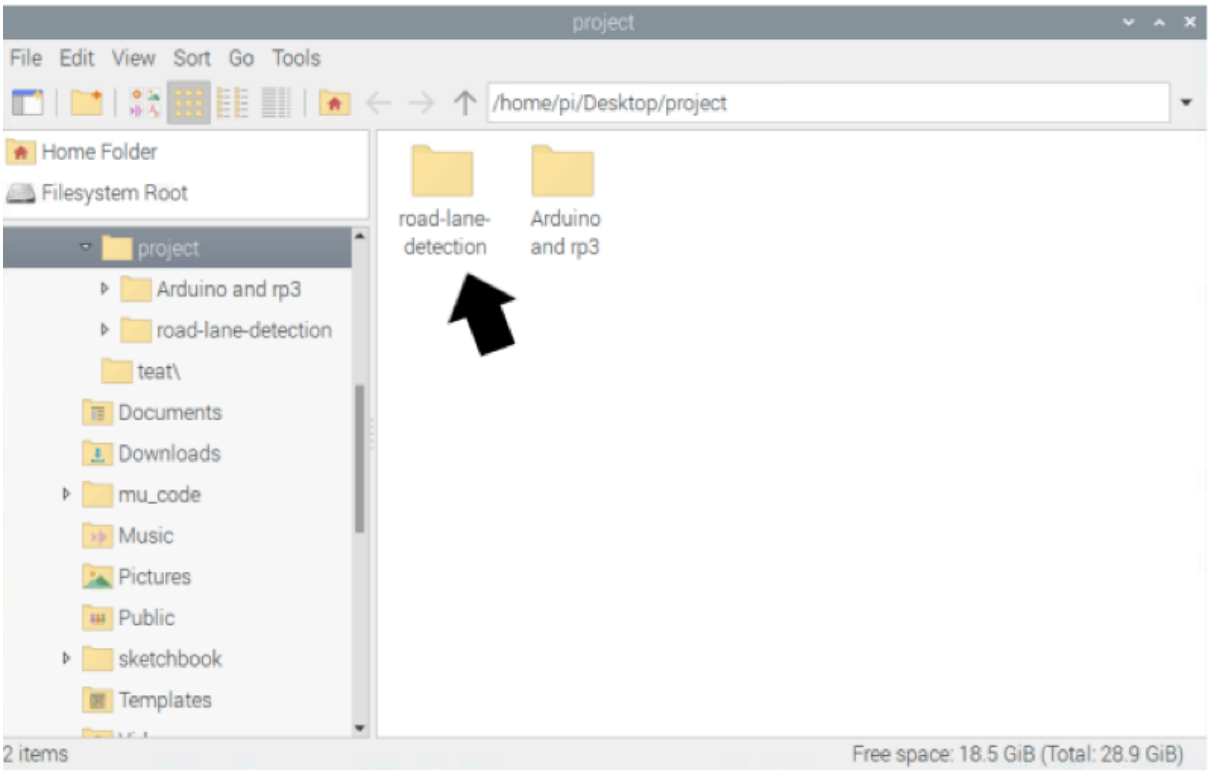
הקוד

כאשר אנחנו מחוברים אנחנו פותחים את תיקייה פרויקט:

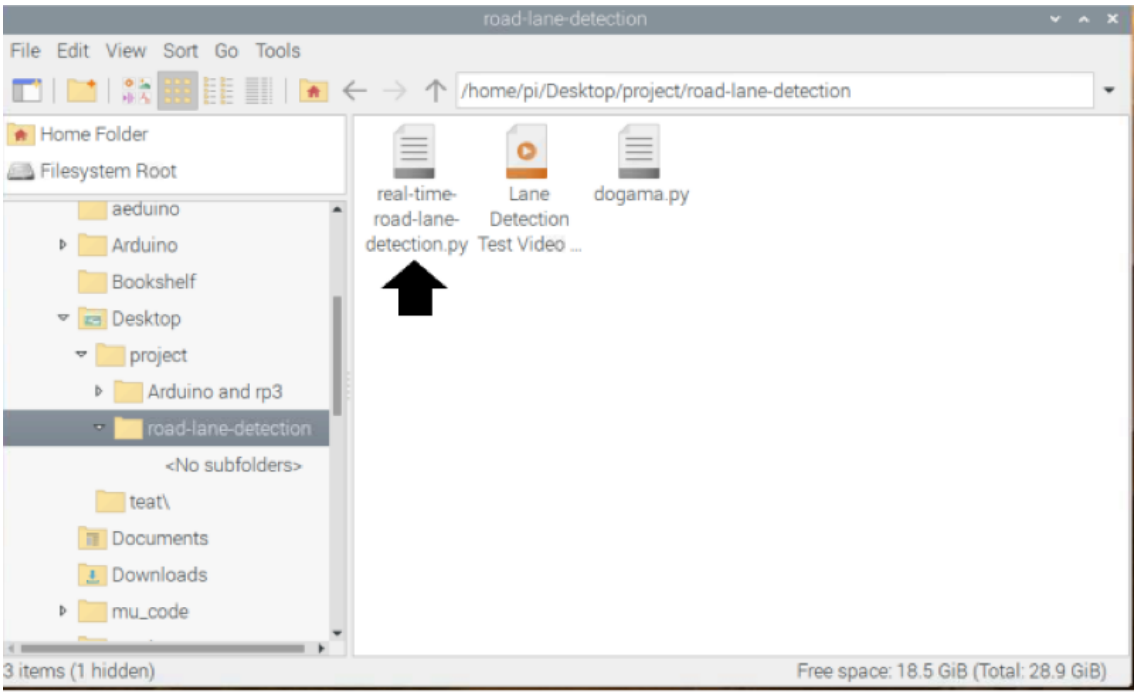


בית ספר אורט ע"ש י. רבין" גן-יבנה

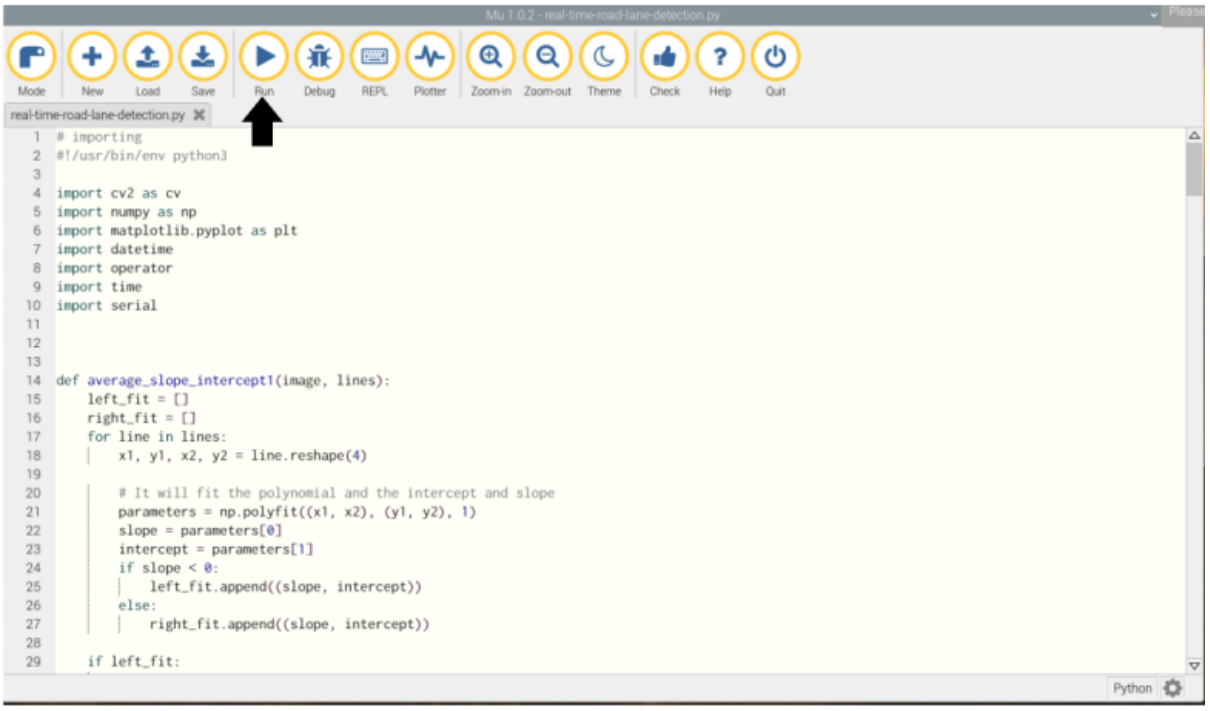
ובתוך התיקיה פותחים את התיקיה של road line:



ובתיקיה פותחים את הקוד של road line:



כאשר הוא פתוח אנחנו לוחצים על run:



וכאשר אנחנו רוצים לעצור את הקוד אנחנו לוחצים על stop

כאשר אנו רוצים לכבות את הרובוט אנחנו מכבים את הRP בכך שנכנסים ל logout ולוחצים על shutdown:

בית ספר אורט ע"ש י. רבין" גן-יבנה

מדריך למפתח

כל הקבצים שלנו נמצאים בתיקיות:

1) /home/pi/Desktop/project/road-line-detection
2) /home/pi/Desktop/project/Arduino and rp3/Main2

הפרויקט מכיל 4 קבצים שהם:

real-time-road-lane-detection.py

Main.ino

Motors.h

Motors.ino

ארדואינו והפעלת מנועים

על הארדואינו יש את מחלקת motors האחראית על לשלוח את הערכים המתאימים למנועים של הרובוט ובנוסף לכך בארדואינו נמצא הקובץ main נמצאת הלולאה הראשית של הארדואינו אשר מקבלת מספר מהRP המייצג את הפקודה ומבצע אותה על ידי שליחת מידע מתאים למנועים. בקובץ h.motors נמצאת הגדרת קבועים המייצגים את נקודות החיבור של החיישנים והמנועים עם הארדואינו אשר ישמשו אותנו מאוחר יותר כדי לשוח ערכים לרכיבים המתאימים. בנוסף לכך הוספנו קבוע המייצג את קוטר הגלגל ובמחלקה נמצאות פעולות השולחות מתח לגלגלים (RightMotor ו LeftMotor)ופעולה המאתחלת את המנועים בתחילת התוכנה(motorInit)

```
#ifndef MOTOR_H
#define MOTOR_H
//=========================================
//motor encoder
#define encoder_Ticks_Rate 824.424

//=========================================
//motor power
#define Max_Motors_Power 191

//=========================================
// L298N defines
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 12
#define ENA 5
#define ENB 6
#define M1_PHASE_A 2
#define M2_PHASE_A 3
#define M1_PHASE_B 4
#define M2_PHASE_B 7

//=========================================
//Whells
#define Weel_Diameter 6.45

// distance between wheels
#define DISTANCE 18.5

void MotorsInit();
void RightMotor(int power);
void LeftMotor(int power);
double GetRightWheelSpeed();
double GetLeftWheelSpeed();


#endif
```

במחלקה motors ישנם מספר מישתנים ופונקציות אחת מהם היא GetRightWheelSpeed אשר מישתמשת בפונקציות הבאות שנדבר אילהם ומהנתונים של הפונקציות הללו היא מחשבת את המהירות של הגלגלים.

```
#include "Motors.h"


volatile long LeftMotorCounter;
volatile long RightMotorCounter;
volatile bool LeftWheelIsMovingForward;
volatile bool RightWheeltIsMovingForward;

long OldLeftMotorCounter, OlddRightMotorCounter;
int OldLeftCurrentEncoderTicks, OldRightCurrentEncoderTicks;

long LeftMotorCounterCopy;
long RightMotorCounterCopy;
bool RightWheeltIsMovingForwardCopy;
bool LeftWheelIsMovingForwardCopy;

//=====================================
double GetRightWheelSpeed()
{
    // 1. CopyISRCounters() must be called right before calling this function
    // 2. this function must be called every 50 ms but caculats average
    // speed for the last 100 ms

    int currentEncoderTics = (RightMotorCounterCopy - OlddRightMotorCounter);
    OlddRightMotorCounter = RightMotorCounterCopy;

    // V = (2*PI*mu*R)/E
    // mu - ticks per second, E - encoder tick per round
    // (currentEncoderTics + OldLeftCurrentEncoderTicks) is mu per 100 ms, should be per second
    // that's why it is multilied by 10
    double Vcurr = (PI * (double)(currentEncoderTics + OldRightCurrentEncoderTicks) * 10 * Weel_Diameter) / encoder_Ticks_Rate;

    OldRightCurrentEncoderTicks =  currentEncoderTics;

    return Vcurr;
}// end RightWheelPID

//=====================================
double GetLeftWheelSpeed()
{
    // 1. CopyISRCounters() must be called right before calling this function
    // 2. this function must be called every 50 ms but caculats average
    // speed for the last 100 ms

    int currentEncoderTics = LeftMotorCounterCopy - OldLeftMotorCounter;
    OldLeftMotorCounter = LeftMotorCounterCopy;

    // V = (2*PI*mu*R)/E
    // mu - ticks per second, E - encoder tick per round
    // (currentEncoderTics + OldLeftCurrentEncoderTicks) is mu per 100 ms, should be per second
    // that's why it is multilied by 10
    double Vcurr = (PI * (double)(currentEncoderTics + OldLeftCurrentEncoderTicks)* 10 * Weel_Diameter) / encoder_Ticks_Rate ;

    OldLeftCurrentEncoderTicks =  currentEncoderTics;

    return Vcurr;
}

//-------------------------------------
```

הפעולה CopyISRCounters מעתיקה את הערך מהאינקודרים לעותקים שלהם, פעולה זו תרוץ בכל פעם שנרצה לבדוק את ערך האינקודרים. יש לשים לב כי פעולה זו עוצרת את כל האינטראפטים כלומר

```
//=====================================

void CopyISRCounters()
{
    noInterrupts();
    LeftMotorCounterCopy = LeftMotorCounter;
    RightMotorCounterCopy = RightMotorCounter;
    LeftWheelIsMovingForwardCopy = LeftWheelIsMovingForward;
    RightWheeltIsMovingForwardCopy = RightWheeltIsMovingForward;

    interrupts();
}

//=====================================
```

אינטראפט זו פעולה הדואגת למענה מהיר להתרחשויות שונות לדוגמא שינוי ערך האינקודר , כלומר עצירת האינטראפטים גורמת לכך שחיישן האינקודר לא ישנה את ערך המשתנים שלו כלומר בהעתק של המשתנים יהיה הערך המעודכן ביותר לאחר מכן האינטראפטים חוזרים לפעולה בסיום פעולה זו.

הפעולות LeftEncoderISR ו RightEncoderISR אחריות לעדכן את משתני האינקודרים בהתאם.
אם הרובוט נא קדימה המשתנים גדלים בהתאם אם הרובוט נע בכיוון ההפוך המשתנים יקטנו.
פעולות אלו יחוברו לאינקודרים בתור אינטראפטים אשר יתרחשו בעת שינוי ערך האינקודרים.

```
//=====================================

void LeftEncoderISR()
{
    if(digitalRead(M1_PHASE_B))
    {
        LeftWheelIsMovingForward = true;
        LeftMotorCounter ++;
    }
    else
    {
        LeftWheelIsMovingForward = false;
        LeftMotorCounter --;
    }

}
//=====================================
```

```
//=====================================
void RightEncoderISR()
{
    if(digitalRead(M2_PHASE_B))
    {
        RightWheeltIsMovingForward = true;
        RightMotorCounter --;
    }
    else
    {
        RightWheeltIsMovingForward = false;
        RightMotorCounter ++;
    }

}// end RightEncoderISR
```

הפעולות LeftMotor ו RightMotor הן פעולות לשליחת המתח למנוע שמאל ומנוע ימין בהתאמה. פעולות אלה מקבלות ערך של המתח למנוע הימני ושמאלי בהתאם . הפעולות קודם כל בודקות האם הערך של המתח שלילי או חיובי כלומר אם הפעולה מקבלת ערך שלילי של מתח היא מחליפה את כיוון המנוע . לאחר מכן בהתאם לכיוון אנו שולחים ערכים לבקר כדי לקבוע את כיוון תנועת המנועים. ולאחר מכן אנו שולחים את המתח שקיבלנו למנוע דרך הפורט האנלוגי המתאים.

```
void LeftMotor(int power)
{
    // if power >= 0 then move forward, direction = true
    // otherwise move backward, direction = false and make power positive
    //
    bool direction = false;

    if (power >= 0)
        direction = true;
    else
        power *= -1;

    if(power >  Max_Motors_Power)
            power =  Max_Motors_Power;

    if(direction)
    {
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH);
    }
    else
    {
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
    }

    analogWrite(ENB, power);
}
```

```
//-------------------------------------------------------------

void RightMotor(int power)
{
    // if power >= 0 then move forward, direction = true
    // otherwise move backward, direction = false and make power positive
    //
    bool direction = false;

    if (power >= 0)
        direction = true;
    else
        power *= -1;

    if(power > Max_Motors_Power)
            power = Max_Motors_Power;

    if(direction)
    {
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
    }
    else
    {
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
    }

    analogWrite(ENA, power);
```

מטרת הפעולה MotorsInit היא לאתחל את המנועים בתחילת ריצת התוכנה, לאפס את האינקודרים ולאתחל את הפינים המתאימים של הארדואינו חלק לקלט וחלק לפלט בהתאם למטרה שלהם.

```
void MotorsInit()
{
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);

    pinMode(M1_PHASE_B, INPUT);
    pinMode(M2_PHASE_B, INPUT);

    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);

    LeftMotorCounter = 0L;
    RightMotorCounter = 0L;
}
```

כעת נעבור למחלקת main אשר שם נימצה הלולאה של הארדרואינו:

אנו מגדירים את המישתנים של המחלקה של main ומוסיפים את המחלקות הקודמות שלנו.

```
#include "Motors.h"


#define VKp 10
#define VKd 0// vk/dt -0.5/50
#define VKi 0.0  //0.001

#define OKp 10
#define OKd 0.0
#define OKi 0.0

double OldError;

double ErrorSum;
unsigned long OldTimer, OldSpeedTimer ;

int ComSpeed;
double OldSpeedError;
double SpeedErrorSum;

double TurnReference;
double OldTurnError;
double TurnErrorSum;

double ReferenceAngleY;

double RightWheelSpeed, LeftWheelSpeed;

double correction;

int TurnCorrection_R;
int TurnCorrection_L;

//==============================================================
```


```
int incomingByte = 0;
char letter = '\0';
String data = String('$');
String ComplitData = String();

void CommProcess()
{
  if(Serial.available() <= 0)
    {
      ComplitData = String('$');
      return;
    }
 if(Serial.available()>0)
  {
    int incomingByte = Serial.read();
    if(incomingByte != 95)
    {
      letter = incomingByte;
      if(data == "$")
      {
        data = String(letter);
      }
      else
      {
        data +=letter;
      }
    }
    if(incomingByte == 95)
    {
      ComplitData = String(data);
      data = String('$');

    }
  }
}
```

הפעולה commProcess() היא פונקציה שמטרתה היא לתקשר עם ה RP ואם היא הצליחה לתקשר אז היא לוקחת את המידע שהיא קיבלה מ RP ומכניסה אותו(את המידע שהוא char) למחרוזת data וכאשר היא מקבלת את המידע האסקי 95 שזה("_") אז אנו שומרים את המחרוזת data במחרוזת אחרת בשם ComplitData ומאפסים את מחרוזת של data לסימן $. סימן ה $ זה הסימן הקבוע שבחרנו בו כדי לייצג את המחרוזת בתור מחרוזת ריקה.

הפונקציה Translation מתרגמת את המחרוזת ComplitData לפי הפרוטוקול הבא:

יש שני משתנים אחד למהירות סיבובית ואחד למהירות קווית לפי הפרוטוקול כאשר אנו מקבלים את המחרוזת (x-מספר בין 1 ל -1) y,0-20-מספר בין VxTy אז הV מסמן שהמספר x הוא מהירות קווית וה T מסמל שזה מהירות סיבובית. כאשר תירגמנו אז המהירות הקווית נכנסת למשתנה (מסוג int) בשם ComSpeed המהירות הסיבובית נכנסת למשתנה (מסוג double) בשם TurnReference.

```
//==============================================================
int ComplitDataLength;
String ComSpeedSTR = String();
String TurnReferenceSTR = String();
int IndexOfT;

void Translation()
{
  if(ComplitData != "$")
  {
    ComplitDataLength = ComplitData.length();
    IndexOfT = ComplitData.indexOf("T");

    ComSpeedSTR = String(ComplitData.substring(1,IndexOfT));
    TurnReferenceSTR = String(ComplitData.substring(IndexOfT+1,ComplitDataLength));

    ComSpeed = ComSpeedSTR.toInt();
    TurnReference = TurnReferenceSTR.toDouble();


  }

    //Serial.print(ComSpeed);
    //Serial.print('\n');
}
```

הפונקציה speedControl היא אחד מהפונקציות שהמטרה שלהם זה לעשות pid לשגיאה
והפונקציה שאנו מדברים אליה היא פונקציה שעושה pid למהירות הקווית של הרובוט.
בהתחלה היא מחשבת את השגיאה של המהירות על ידי הנוסחה הבאה.

error=(RightWheelSpeed - LeftWheelSpeed) / (RightWheelSpeed /ComSpeed)

ואחרי זה אנו עושים
לשגיאה pid ושומרים
משתנה ReferenceAngleY.

```
void SpeedControl()
{
  Er = (RightWheelSpeed - LeftWheelSpeed) / (RightWheelSpeed /ComSpeed);



  double vCurr = (RightWheelSpeed + LeftWheelSpeed) / 2.0;
  double error = (ComSpeed - vCurr);

  double Kcorrection = VKp * error;
   //
   if(ComSpeed >=0 && Kcorrection < 0)
    Kcorrection = 0;
   else if(ComSpeed <= 0 && Kcorrection > 0)
    Kcorrection = 0;


  double Dcorrection = VKd * (error - OldSpeedError);

  OldSpeedError = error ;

  correction = Kcorrection + Dcorrection;

  ReferenceAngleY = correction;


}
```

פונקציה StabilizeRobot אנו לוקחים את התוצאות של הpid של שתי מהירויות(קווית, סיבובית)
מחברים ביניהם ואז משתמשים בפונקציה RightMotor,LeftMotor (על פונקציות הללו הסברתי
מוקדם יותר
בספר).

```
void StabilizeRobot()
{

    double correction = ReferenceAngleY;
    double FinalPowerR=((int)correction + (int)TurnCorrection_R );
    double FinalPowerL((int)correction + (int)TurnCorrection_L);

    RightMotor(FinalPowerR);
    LeftMotor(FinalPowerL);

}
```

פונקציה setup היא הפונקציה הראשונה שתפעל כאשר אנו מדליקים את הארדרואינו והיא תופעל רק פעם אחת והמטרה שלה היא איפוס משתנים וקביעת משתנים ושימוש בפונקציות שמטרתם דומה. גם אנו קובעים שם משתנים שהם ברירת מחדל.

```
void setup()
{
  MotorsInit();
  Serial.begin(9600);

  ErrorSum = SpeedErrorSum = TurnErrorSum = 0.0;
  ReferenceAngleY = 0;
  ComSpeed = 0.0;
  TurnReference = 0.0;
  TurnCorrection_L = TurnCorrection_R = 0;
  RightWheelSpeed = LeftWheelSpeed = 0.0;
  Er = 0;
  attachInterrupt(digitalPinToInterrupt(M1_PHASE_A), LeftEncoderISR, RISING);
  attachInterrupt(digitalPinToInterrupt(M2_PHASE_A), RightEncoderISR, RISING);
  OldTimer = OldSpeedTimer = millis();
}

//
```

פונקציה loop היא הפונקציה החשובה ביותר בגלל שהיא הפונקציה פועלת אחרי setup והיא פועלת עד שהארדואינו מפסיק לקבל חשמל.

בפונקציה הזות אנו משתמשים בכל הפונקציות האחרות.

```
void loop()
{
  CommProcess();
  Translation();


    unsigned long currMillis = millis();
    if(currMillis>10000){
    if(currMillis - OldTimer >= 10L)
    {
        OldTimer = currMillis;
         StabilizeRobot();
    }
   if (currMillis - OldSpeedTimer >= 50L)
    {
      OldSpeedTimer = currMillis;
      CopyISRCounters();

      RightWheelSpeed = GetRightWheelSpeed();
      LeftWheelSpeed = GetLeftWheelSpeed();

      SpeedControl();

       double omegaCurr = (RightWheelSpeed - LeftWheelSpeed) / DISTANCE;
       double error = TurnReference*10-omegaCurr;
       double Kcorrection = OKp * error;
       double Dcorrection = OKd * (error - OldTurnError);
       OldTurnError = error;
       TurnErrorSum += OKi * error;
       double correction = Kcorrection + Dcorrection + TurnErrorSum;

       TurnCorrection_R = correction;
       TurnCorrection_L = -correction;



    }
    }
```

תחילה אנו משתמשים בתקשורת והתרגום כדי לקבל ולתרגם את המידע.

```
void loop()
{
  CommProcess();
  Translation();
```

אחר כך אנו בודקים האם עברו 10000 מילישניות אם כן אז אנו מפעילים את הפונקציה StabilizeRobot.

```
if(currMillis>10000){
if(currMillis - OldTimer >= 10L)
{
    OldTimer = currMillis;
     StabilizeRobot();
}
```

אנו גם בודקים עם עברו 50 מילישניות ואם כן אז הרובוט מפעיל את הפונקציות שמטרתם היא לחשב את המהירות והוא גם מפעיל את הפונקציה SpeedControl. הוא גם עושה pid למהירות סיבובית ושומר את התוצאות במשתנים: TurnCorrection_R, TurnCorrection_L.

```
if (currMillis - OldSpeedTimer >= 50L)
 {
   OldSpeedTimer = currMillis;
   CopyISRCounters();

   RightWheelSpeed = GetRightWheelSpeed();
   LeftWheelSpeed = GetLeftWheelSpeed();

   SpeedControl();

    double omegaCurr = (RightWheelSpeed - LeftWheelSpeed) / DISTANCE;
    double error = TurnReference*10-omegaCurr;
    double Kcorrection = OKp * error;
    double Dcorrection = OKd * (error - OldTurnError);
    OldTurnError = error;
    TurnErrorSum += OKi * error;
    double correction = Kcorrection + Dcorrection + TurnErrorSum;

    TurnCorrection_R = correction;
    TurnCorrection_L = -correction;

 }
```

הפונקציה loop פועלת כך בגלל שכול 50 מילישניות הוא מחשב ועושה pid למהירויות וכל 10000 מילישניות הוא משתמש במהירויות ומזיז את הרובוט(בשימוש בפונקציה StabilizeRobot) לפי זה.

בית ספר אורט ע"ש י. רבין" גן-יבנה

עיבוד תמונה זיהוי שוליים של הכביש

על הRP יש קובץ שנקרא real-time-road-lane-detection.py

והמתרה שלו היא עובד תמונה זיהוי שולים ותיקשורת עם

הארדרואינו.

תחילה אנו מוסיפים את הספריות שבהם אנו נשתמש.

cv2 או opencv זה ספריה לעיבוד תמונה

numpy זה ספריה לפעולות מתמטיות

שאר הספריות הם לשימושים כמו זמן תקשורת והצגת תמונה.

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import datetime
import operator
import time
import serial
```

פעולה infroCam היא בשביל שנקבל ונישמור נתונים של המצלמה כמו רזולוציה, כמות תמונות בשנייה. וגם אנו מוסיפים את הנתונים הללו כטקסט על המצלמה.

```
# Place the information on the camera screen
def infroCam(frame, cap):
    font = cv.FONT_HERSHEY_COMPLEX_SMALL
    datet = str(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
    text = 'width: ' + str(cap.get(3)) + ' Height: ' + str(cap.get(4)) + ' Frame: ' + str(cap.get(5))
    frame = cv.putText(frame, datet, (0, 20), font, 0.7, (0, 0, 255), 1, cv.LINE_AA)
    frame = cv.putText(frame, text, (0, 40), font, 0.7, (0, 0, 255), 1, cv.LINE_AA)
```

פעולה infrocamPrint היא בשביל המפתח והיא מדפיסה את האינפורמציה של המצלמה.

```
# Prints camera information
def infrocamPrint(cap):
    print('infro from camara:')
    print('the HEIGHT ', cap.get(cv.CAP_PROP_FRAME_HEIGHT))
    print('the WIDTH ', cap.get(cv.CAP_PROP_FRAME_WIDTH))
    print("fps ", cap.get(cv.CAP_PROP_FPS))
```

בפעולה makeMask אנו מיצרים את הפילטר שיסנן את הצבעים וישאיר רק את הצבע של הקויים. אנו גם קובעים את הצבע שהוא יחפש שבמיקרה זה הוא הקויים.

```
# make the mask image
def makeMask(frame):
    l_h = 82
    l_s = 48
    l_v = 43

    u_h = 121
    u_s = 134
    u_v = 146

    l_b = np.array([l_h, l_s, l_v])
    u_b = np.array([u_h, u_s, u_v])

    hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

    mask = cv.inRange(hsv, l_b, u_b)
    mask = cv.GaussianBlur(mask, (5, 5), 0)
    mask = cv.medianBlur(mask, 5)

    return mask
```

בפעולה showImages אנו מציגים במסך את הוידאו של המצלמה בכמה מצבים שחלק מהם היא אם הפילטר בלי הפילטר ואחרי זיהוי הקווים.

```
# show the images on the screen
def showImages(frame, mask, edges, roi, res, line_image):
    #cv.imshow("video", frame)
    #cv.imshow("mask", mask)
    cv.imshow('Edges', edges)
    cv.imshow("roi", roi)
    cv.imshow("res", res)
    # cv.imshow("line_image", line_image)
```

פעולה breakLoop אחראית על סיום לולאה בפונקציה הראשית.

```
def breakLoop():
    key = cv.waitKey(1000)
    if key == 27:
        return False
    return True
```

בפעולה region_of_interest אנו קובעים את אזור העניין של הזיהוי קווים כדי להקל על הRP וכדי שהפילטרים ועיבוד תמונה לא יזהו דברים ועצמים שלא שייכים לכביש.

```
def region_of_interest(edges):
    height = edges.shape[0]
    #polygons1 = np.array([[(0, height), (640, height), (320, 240)]])
    polygons1 = np.array([[(0, height), (640, height), (0, 250)]])
    polygons2 = np.array([[(0, 250), (640, height), (640, 250)]])

    mask = np.zeros_like(edges)
    cv.fillPoly(mask, polygons1, 255)
    cv.fillPoly(mask, polygons2, 255)

    masked_image = cv.bitwise_and(edges, mask)
    return masked_image
```

הפעולה create_coordinates מייצרת את הנקודות ציון של הקווים ושומרת אותם בתוך מערך של np.

```
def create_coordinates(image, line_parameters):
    slope, intercept = line_parameters
    y1 = image.shape[0]
    y2 = int(y1 * (3 / 5))
    x1 = int((y1 - intercept) / slope)
    x2 = int((y2 - intercept) / slope)
    return np.array([x1, y1, x2, y2])
```

הפונקציה midelOFaLine היא פונקציה שמטרתה היא למצוא את האמצע של הקו שהיא מקבלת היא עושה את זה על ידי השימוש בנוסחה :

midx=(x1-x2)/2 midy=(y1-y2)/2

ואז היא מחזירה את התוצאה בתור מחרוזת.

```
def midelOFaLine(lines_coordinates):
    xMID = (lines_coordinates[0] + lines_coordinates[2]) / 2
    yMID = (lines_coordinates[1] + lines_coordinates[3]) / 2
    return np.array([xMID, yMID])
```

הפונקציה midelOfRode עושה את אותו הדבר אך במקום לקחת קו אחד היא לוקחת את שניהם ומחשבת את הנקודה שהיא האמצע של שני הקווים. ומחזירה את התוצאה בתור מחרוזת.

```
def midelOfRode(lineL, lineR):
    lineMidL = midelOFaLine(lineL)
    lineMidR = midelOFaLine(lineR)
    midX = (lineL[0] + lineR[0]) / 2
    midY = (lineL[1] + lineR[1]) / 2
    return np.array([midX, midY])
```

פונקציה errorTurn אנו מחשבים את השגיאה ואת המרחק בין נקודות ציון של midelOfRode ושל אמצע המסך וזה קדי להגיד לרובוט מתי הוא באמצע ומתי הוא קצת ימינה או שמאלה מאמצע הכביש. הוא עושה את זה בכך שהוא בודק אם הוא מוצא את קו שמאל וקו ימין אם כן אז הוא מחשב את אמצע הכביש לוקח את נקודת ה x שלה ומחסר אותה בנקודת ה x של אמצע המסך וכדי שזה היה בין 1 ל 1-

אנו מחלקים בנקודה x של אמצע המסך. אם אנו לא רואים את קו ימין או שמואל אז הטעות היא מקסימלית ואנו צרכים להחזיר 1 או 1- תלוי לי איזה קו הוא לא רואה. פונקציה זה היא בשביל המהירות הסיבובית של הרובוט.

```
def errorTurn(left_fit, right_fit, image):
    global turnError

    if left_fit:
        left_fit_average = np.average(left_fit, axis=0)
        # print(left_fit_average, 'left')
        left = True
        left_line = create_coordinates(image, left_fit_average)
        # print(left_line, "left_line")
    else:
        left = False
    if right_fit:
        right_fit_average = np.average(right_fit, axis=0)
        # print(right_fit_average, 'right')
        right = True
        right_line = create_coordinates(image, right_fit_average)
        # print(right_line, "right_line")
    else:
        right = False

    if(left and right):
        turnError=((midelOfRode(left_line, right_line)[0])-320)/320
    elif(not(right) and left):
        turnError=1
    elif(not(left) and right):
        turnError=-1
    else:
        turnError=0
```

פונקציה checkAndPrint עושה את אותו עבודה כמו errorTurn אך היא עושה את זה למהירות הקווית בכך שאם היא לא רואה אף אחד מהקווים אז המהירות היא 0 והם היא רואה את שניהם אז היא בודקת אם הנקודת ציון בציר ה x של midelOfRode ובודקת אם היא בטווח של pixel 30 מאמצע המסך אם כן אז היא מחזירה למישתנה msg מהירות 10 ומוסיפה את זה כדי שזה יראה כמו בפרוטוקול תקשורת שעליו דיברנו לפני בארכיטקטורה אם הוא כן רואה את שני הקווים אך זה לא באמצע אז הוא עדין יחזיר למשתנה מהירות 10 ויוסף את זה לפרוטוקול. אם הוא לא רואה את אחד הקווים המהירות שיחזיר תהיה 0.

```
def checkAndPrint(left_fit, right_fit, image):
    global msg
    if left_fit:
        left_fit_average = np.average(left_fit, axis=0)
        # print(left_fit_average, 'left')
        left = True
        left_line = create_coordinates(image, left_fit_average)
        # print(left_line, "left_line")
    else:
        left = False
    if right_fit:
        right_fit_average = np.average(right_fit, axis=0)
        # print(right_fit_average, 'right')
        right = True
        right_line = create_coordinates(image, right_fit_average)
        # print(right_line, "right_line")
    else:
        right = False

    if (left and right and (310 < midelOfRode(left_line, right_line)[0] < 330)) and (
            left and right and midelOFaLine(left_line)[1] == midelOFaLine(right_line)[1]):
        msg = 'V10T' #MID
    elif left is False and right is False:
        msg = 'V0T' #STOP
    elif right and (left is False):
        msg = 'V0T' #LEFT
    elif (right is False) and left:
        msg = 'V0T' #RIGHT
    elif midelOfRode(left_line, right_line)[0] > 330:
        msg = 'V10T' #right + drive
    elif 310 > midelOfRode(left_line, right_line)[0]:
        msg = 'V10T' #left + drive
    else:
        msg = 'error'

    d[msg] = d[msg] + 1
    # print(msg)
```

הפונקציה average_slope_intercept מחשבת את המדרון ואת הקווים ומפעילה את הפונקציות errorTurn ו checkAndPrint.

```
def average_slope_intercept(image, lines):
    left = False
    right = False
    left_fit = []
    right_fit = []
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line.reshape(4)
            parameters = np.polyfit((x1, x2), (y1, y2), 1)
            slope = parameters[0]
            intercept = parameters[1]
            if slope < 0:
                left_fit.append((slope, intercept))
            else:
                right_fit.append((slope, intercept))
    errorTurn(left_fit, right_fit, image)
    checkAndPrint(left_fit, right_fit, image)
```

הפונקציה main היא הפונקציה הראשית של המחלקה ושם בהתחלה אנו מגדירים את המשתנים ופותחים את התקשורת שלנו.

```
def main():
    #communication
    ser = serial.Serial('/dev/ttyACM0', 9600)

    counter = 0
    flag = True
    cap = cv.VideoCapture(0)
    cap.set(3, 640)
    cap.set(4, 480)
    cap.set(5, 10)
```

אחר כך אנו פותחים לולאה שתהיה אינסופית אלא אם המצלמה תכבה או כאשר נילחץ esc בהמשך אנו משתמשים בפונקציות כדי ליצור פילטרים ולהוסיף אינפורמציה.

```
infroCam(frame, cap)  # Activates the function infrocam
mask = makeMask(frame)  # Activates the function makemask
edges = cv.Canny(mask, 0, 400)  # Prepares an image only at the edges of the mask
roi = region_of_interest(edges)
debug = region_of_interest(frame)
#cv.imshow('debug', debug)
```

כפי שניתן לראות בהמשך הפונקציה מחפשת אחר הקויים על ידי שימוש בפונקציה מה opencv הנקרא HoughLinesP. ואז היא מחשבת את ה averaged_lines אחרי חישובים הללו אנו מדפיסים ושולחים לארדרואינו את המהירויות כפי שצריך לשלוח (על פי הפרוטוקול) לבסוף אנו משתמשים בפונקציה showImages כדי להראות את התמונות.

```
    msg = 'V0T0'+"_"
    lines = cv.HoughLinesP(roi, 2, np.pi / 180, 100, np.array([]), minLineLength=40, maxLineGap=5)
    averaged_lines = average_slope_intercept(edges, lines)
    counter += 1
    if counter > 2:
        msg = (max(d.items(), key=operator.itemgetter(1))[0])
        counter = 0
        for i in d.keys():
            d[i] = 0
        print((f'{msg}{turnError}')+'_')
        #communication
        buff = ((f'{msg}{turnError}')+'_').encode()
        ser.write(buff)

    line_image = display_lines(frame, averaged_lines)
    res = cv.addWeighted(frame, 0.8, line_image, 1, 1)
    showImages(frame, mask, edges, roi, res, line_image)  # Activates the function showImages


    flag = breakLoop()  # Activates the function showImages breakLoop
cap.release()
cv.destroyAllWindows()
```

בית ספר אורט ע"ש י. רבין" גן-יבנה

רפלקציה-נתנאל

העבודה בפרויקט והמחקר שלו היה מיוחד,מעניין, קשה, ומעייף. אני בהתחלה העלתי רעיונות אחרים לפרויקט והתחלתי לחקור עליהם אך גיליתי שהם לא מתאימים לפרויקט בסדר גודל כזה אז חשבנו על פרויקט ואני רציתי לעבוד עם חומרה בגלל שאני התעסקתי עם חומרה בעבר וזה חומר שמעניין אותי ובסוף הגענו להחלטה לעשות מכונית שמזה את השוליים. כשהתחלנו הרגשתי עניין בגלל עיבוד תמונה זה חומר חדש בשבילי ובאמצע הפרויקט הרגשתי בלי מוטיבציה אך לעומת כל העומס סיימתי לעבוד על העיבוד תמונה והתחלנו לעבוד על הרובוט ולבסוף הרגשתי לחץ כי הפרויקט הרגיש לא מוכן (החלקים של הפרויקט עבדו איך כשילוב זה השתבש) אבל כאשר הוא עבד הרגשתי שמחה.

הכלים שקיבלתי הם: עמידה בזמנים, עבודה בלחץ, עבודה עם חשמל ואלקטרוניקה, עבודה עם מנועים, עבודה עם עיבוד תמונה ודרך לחקור אלגוריתמים ונוסחאות מסובכות. כלים הללו אני ישתמש בשאר החיים שלי.

אם הייתי מתחיל היום את הפרויקט הייתי מפצל אותו לשני פרויקטים שונים הראשון הוא חיקוי של מוביליט על ידי עיבוד תמונה ותמרורים ולמידת מכונה. והשני שהוא רובוט אשר נישלת בצורה אלחוטית מהמחשב על ידי בלוטוס או wifi ולמחשב היה שלט ששולח את הנתונים של התזוזה למחשב. כדי שהעבודה הייתה יעילה יותר הייתי צריך להגביל את הזמן בין המחקר לעבודה ולקבוע מטרות ברורות ומפורטות לגבי הפרויקט.

אחד הדברים שסיבכו את הפרויקט היה הקורונה בגלל שהפרויקט שלנו היה פיזי לא היינו יכולים להיפגש והמורה לא היה יכול להסביר אחד לאחד על הפרויקט ואני חושב שאם הינו במצב אחר הדרך לפרויקט היה יותר ברור ופשוט.

המסקנות שלי מהפרויקט הם שכול פרויקט יהיה מפחיד ומסובך בהתחלה אך אם תשקיעו את הזמן ותיבחר פרויקט שמעניין אותך אך לבסוף הפרויקט היה קל ולא מסובך.

בית ספר אורט ע"ש י. רבין" גן-יבנה

רפלקציה-אייל

בתחילת העבודה ממש התלהבתי מהרעיון שאנחנו עומדים לעשות רובוט שנוסע לבד אך ככל שעבר הזמן הבנתי שזה פרויקט לא קל שדורש המון המון ידע ,למזלי שותפי לפרוייקט מכיר את העולם האלקטרוני לפני וממש הסביר לי כמו שצריך.
לבסוף בשלבי הסיום הרגשתי הנאה מכיוון שסיימנו בהצלחה ושהצלחנו להתגבר על הקשיים.

במהלך העבודה על הפרוייקט רכשתי ידע בעולם האלקטרוניקה ולמדתי עוד תוכנות עיבוד שלא הכרתי כלל.
האתגרים שעמדו בפנינו הם בניית הרובוט עצמה ולמידת המכונה דרך המחשב ,בנוסף עשינו את הפרויקט בתקופת הקורונה כך שחלק גדול מאוד מהזמן עבדנו עם המורה דרך הזום וגם כך לא יכולתי להיפגש עם נתנאל חברי לקבוצה לעבוד איתו אחד ליד השני.

המסקנות שלי מהפרוייקט הם שהייתי צריך לקחת משחק מחשב במקור מידע ותוכנות שאני מכיר לפני ולא ללמוד מהתחלה את כל התוכנות והאלקטרוניקה.
אני חושב שעבודה הייתה יותר יעילה אם היינו מסדרים את הזמן שלנו אחרת טיפה (יותר מסודר)

בית ספר אורט ע"ש י. רבין" גן-יבנה

ביבליוגרפיה

סירטון הסבר ללימוד ב opencv
https://www.youtube.com/watch?v=kdLM6AOd2vc&list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K&index=1&ab_channel=ProgrammingKnowledge

סירטון הסבר ללימוד ב ארדרואינו
https://www.youtube.com/watch?v=1R3fqSFCAjM

opencv documentation
https://docs.opencv.org/master/

Arduino Reference
https://www.arduino.cc/reference/en/

סירטון לימוד pid
https://www.youtube.com/watch?v=wkfEZmsQqiA&list=PLn8PRpmsu08pQBgjxYFXSsODEF3Jqmm-y&ab_channel=MATLABMATLAB%D7%9E%D7%90%D7%95%D7%9E%D7%AA

בית ספר אורט ע"ש י. רביץ' גן-יבנה

נספחים

הקוד של העיבוד תמונה:

```python
# importing
#!/usr/bin/env python3

import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import datetime
import operator
import time
import serial


# פונקציות שאני לא מישתמש
def average_slope_intercept1(image, lines):
    left_fit = []
    right_fit = []
    for line in lines:
        x1, y1, x2, y2 = line.reshape(4)

        # It will fit the polynomial and the intercept and slope
        parameters = np.polyfit((x1, x2), (y1, y2), 1)
        slope = parameters[0]
        intercept = parameters[1]
        if slope < 0:
            left_fit.append((slope, intercept))
        else:
            right_fit.append((slope, intercept))

    if left_fit:
        left_fit_average = np.average(left_fit, axis=0)
        print(left_fit_average, 'left')
        left_line = create_coordinates(image, left_fit_average)
    if right_fit:
        right_fit_average = np.average(right_fit, axis=0)
        print(right_fit_average, 'right')
        right_line = create_coordinates(image, right_fit_average)


# Prints camera information
def infrocamPrint(cap):
    print('infro from camara:')
    print('the HEIGHT ', cap.get(cv.CAP_PROP_FRAME_HEIGHT))
    print('the WIDTH ', cap.get(cv.CAP_PROP_FRAME_WIDTH))
    print("fps ", cap.get(cv.CAP_PROP_FPS))


# Place the information on the camera screen
def infroCam(frame, cap):
    font = cv.FONT_HERSHEY_COMPLEX_SMALL
    datet = str(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
    text = 'width: ' + str(cap.get(3)) + ' Height: ' + str(cap.get(4)) + ' Frame: ' + str(cap.get(5))
    frame = cv.putText(frame, datet, (0, 20), font, 0.7, (0, 0, 255), 1, cv.LINE_AA)
    frame = cv.putText(frame, text, (0, 40), font, 0.7, (0, 0, 255), 1, cv.LINE_AA)
```

```python

# make the mask image
def makeMask(frame):
    l_h = 82
    l_s = 48
    l_v = 43

    u_h = 121
    u_s = 134
    u_v = 146

    l_b = np.array([l_h, l_s, l_v])
    u_b = np.array([u_h, u_s, u_v])

    hsv = cv.cvtColor(frame, cv.COLOR_BGR2HSV)

    mask = cv.inRange(hsv, l_b, u_b)
    mask = cv.GaussianBlur(mask, (5, 5), 0)
    mask = cv.medianBlur(mask, 5)

    return mask


# show the images on the screen
def showImages(frame, mask, edges, roi, res, line_image):
    #cv.imshow("video", frame)
    #cv.imshow("mask", mask)
    cv.imshow('Edges', edges)
    cv.imshow("roi", roi)
    cv.imshow("res", res)
    # cv.imshow("line_image", line_image)


# Checks if esc is pressed and if esc is pressed returns False otherwise returns True
# The function exists to exit the loop
def breakLoop():
    key = cv.waitKey(1000)
    if key == 27:
        return False
    return True


def region_of_interest(edges):
    height = edges.shape[0]
    #polygons1 = np.array([[(0, height), (640, height), (320, 240)]])
    polygons1 = np.array([[(0, height), (640, height), (0, 250)]])
    polygons2 = np.array([[(0, 250), (640, height), (640, 250)]])

    mask = np.zeros_like(edges)
    cv.fillPoly(mask, polygons1, 255)
    cv.fillPoly(mask, polygons2, 255)

    masked_image = cv.bitwise_and(edges, mask)
    return masked_image
```

בית ספר אורט ע"ש י. רביץ' גן-יבנה

```python
def create_coordinates(image, line_parameters):
    slope, intercept = line_parameters
    y1 = image.shape[0]
    y2 = int(y1 * (3 / 5))
    x1 = int((y1 - intercept) / slope)
    x2 = int((y2 - intercept) / slope)
    return np.array([x1, y1, x2, y2])


def midelOFaLine(lines_coordinates):
    xMID = (lines_coordinates[0] + lines_coordinates[2]) / 2
    yMID = (lines_coordinates[1] + lines_coordinates[3]) / 2
    return np.array([xMID, yMID])


def midelOfRode(lineL, lineR):
    lineMidL = midelOFaLine(lineL)
    lineMidR = midelOFaLine(lineR)
    midX = (lineL[0] + lineR[0]) / 2
    midY = (lineL[1] + lineR[1]) / 2
    return np.array([midX, midY])


d = {'V10T': 0, 'V0T': 0, 'V0T': 0, 'V0T': 0, 'V10T': 0, 'V10T': 0, 'error': 0}

turnError = 8
def errorTurn(left_fit, right_fit, image):
    global turnError

    if left_fit:
        left_fit_average = np.average(left_fit, axis=0)
        # print(left_fit_average, 'left')
        left = True
        left_line = create_coordinates(image, left_fit_average)
        # print(left_line, "left_line")
    else:
        left = False
    if right_fit:
        right_fit_average = np.average(right_fit, axis=0)
        # print(right_fit_average, 'right')
        right = True
        right_line = create_coordinates(image, right_fit_average)
        # print(right_line, "right_line")
    else:
        right = False

    if(left and right):
        turnError=((midelOfRode(left_line, right_line)[0])-320)/320
    elif(not(right) and left):
        turnError=1
    elif(not(left) and right):
        turnError=-1
    else:
        turnError=0
```

```python
msg = ''
def checkAndPrint(left_fit, right_fit, image):
    global msg
    if left_fit:
        left_fit_average = np.average(left_fit, axis=0)
        # print(left_fit_average, 'left')
        left = True
        left_line = create_coordinates(image, left_fit_average)
        # print(left_line, "left_line")
    else:
        left = False
    if right_fit:
        right_fit_average = np.average(right_fit, axis=0)
        # print(right_fit_average, 'right')
        right = True
        right_line = create_coordinates(image, right_fit_average)
        # print(right_line, "right_line")
    else:
        right = False

    if (left and right and (310 < midelOfRode(left_line, right_line)[0] < 330)) and (
            left and right and midelOFaLine(left_line)[1] == midelOFaLine(right_line)[1]):
        msg = 'V10T' #MID
    elif left is False and right is False:
        msg = 'V0T' #STOP
    elif right and (left is False):
        msg = 'V0T' #LEFT
    elif (right is False) and left:
        msg = 'V0T' #RIGHT
    elif midelOfRode(left_line, right_line)[0] > 330:
        msg = 'V10T' #right + drive
    elif 310 > midelOfRode(left_line, right_line)[0]:
        msg = 'V10T' #left + drive
    else:
        msg = 'error'

    d[msg] = d[msg] + 1
    # print(msg)


def average_slope_intercept(image, lines):
    left = False
    right = False
    left_fit = []
    right_fit = []
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line.reshape(4)
            parameters = np.polyfit((x1, x2), (y1, y2), 1)
            slope = parameters[0]
            intercept = parameters[1]
            if slope < 0:
                left_fit.append((slope, intercept))
            else:
                right_fit.append((slope, intercept))
    errorTurn(left_fit, right_fit, image)
    checkAndPrint(left_fit, right_fit, image)
```

בית ספר אורט ע"ש י. רביץ' גן-יבנה

```python
def display_lines(image, lines):
    line_image = np.zeros_like(image)
    if lines is not None:
        for x1, y1, x2, y2 in lines:
            cv.line(line_image, (x1, y1), (x2, y2), (255, 0, 0), 10)
    return line_image


def main():
    #communication
    ser = serial.Serial('/dev/ttyACM0', 9600)

    counter = 0
    flag = True
    cap = cv.VideoCapture(0)
    cap.set(3, 640)
    cap.set(4, 480)
    cap.set(5, 10)



    while flag & cap.isOpened():
        _, frame = cap.read()

        infroCam(frame, cap)  # Activates the function infrocam
        mask = makeMask(frame)  # Activates the function makemask
        edges = cv.Canny(mask, 0, 400)  # Prepares an image only at the edges of the mask
        roi = region_of_interest(edges)
        debug = region_of_interest(frame)
        #cv.imshow('debug', debug)



        msg = 'V0T0'+"_"
        lines = cv.HoughLinesP(roi, 2, np.pi / 180, 100, np.array([]), minLineLength=40, maxLineGap=5)
        averaged_lines = average_slope_intercept(edges, lines)
        counter += 1
        if counter > 2:
            msg = (max(d.items(), key=operator.itemgetter(1))[0])
            counter = 0
            for i in d.keys():
                d[i] = 0
            print((f'{msg}{turnError}')+'_')
            #communication
            buff = ((f'{msg}{turnError}')+'_').encode()
            ser.write(buff)

        line_image = display_lines(frame, averaged_lines)
        res = cv.addWeighted(frame, 0.8, line_image, 1, 1)
        showImages(frame, mask, edges, roi, res, line_image)  # Activates the function showImages



        flag = breakLoop()  # Activates the function showImages breakLoop
    cap.release()
    cv.destroyAllWindows()


if __name__ == '__main__':
    main()
```

הקוד של הארדרואינו

Motors.h

```
#ifndef MOTOR_H
#define MOTOR_H
//=================================================
//motor encoder
#define encoder_Ticks_Rate 824.424

//=================================================
//motor power
#define Max_Motors_Power 191

//=================================================
// L298N defines
#define IN1 8
#define IN2 9
#define IN3 10
#define IN4 12
#define ENA 5
#define ENB 6
#define M1_PHASE_A 2
#define M2_PHASE_A 3
#define M1_PHASE_B 4
#define M2_PHASE_B 7

//=================================================
//Whells
#define Weel_Diameter 6.45

// distance between wheels
#define DISTANCE 20

void MotorsInit();
void RightMotor(int power);
void LeftMotor(int power);
double GetRightWheelSpeed();
double GetLeftWheelSpeed();


#endif
```

Motors.ino

```
#include "Motors.h"


volatile long LeftMotorCounter;
volatile long RightMotorCounter;
volatile bool LeftWheelIsMovingForward;
volatile bool RightWheeltIsMovingForward;

long OldLeftMotorCounter, OlddRightMotorCounter;
int OldLeftCurrentEncoderTicks, OldRightCurrentEncoderTicks;

long LeftMotorCounterCopy;
long RightMotorCounterCopy;
bool RightWheeltIsMovingForwardCopy;
bool LeftWheelIsMovingForwardCopy;

//=====================================
double GetRightWheelSpeed()
{
    // 1. CopyISRCounters() must be called right before calling this function
    // 2. this function must be called every 50 ms but caculats average
    // speed for the last 100 ms

    int currentEncoderTics = (RightMotorCounterCopy - OlddRightMotorCounter);
    OlddRightMotorCounter = RightMotorCounterCopy;

    // V = (2*PI*mu*R)/E
    // mu - ticks per second, E - encoder tick per round
    // (currentEncoderTics + OldLeftCurrentEncoderTicks) is mu per 100 ms, should be per second
    // that's why it is multilied by 10
    double Vcurr = (PI * (double)(currentEncoderTics + OldRightCurrentEncoderTicks) * 10 * Weel_Diameter) / encoder_Ticks_Rate;

    OldRightCurrentEncoderTicks =  currentEncoderTics;

    return Vcurr;
}// end RightWheelPID

//=====================================
double GetLeftWheelSpeed()
{
    // 1. CopyISRCounters() must be called right before calling this function
    // 2. this function must be called every 50 ms but caculats average
    // speed for the last 100 ms

    int currentEncoderTics = LeftMotorCounterCopy - OldLeftMotorCounter;
    OldLeftMotorCounter = LeftMotorCounterCopy;

    // V = (2*PI*mu*R)/E
    // mu - ticks per second, E - encoder tick per round
    // (currentEncoderTics + OldLeftCurrentEncoderTicks) is mu per 100 ms, should be per second
    // that's why it is multilied by 10
    double Vcurr = (PI * (double)(currentEncoderTics + OldLeftCurrentEncoderTicks)* 10 * Weel_Diameter) / encoder_Ticks_Rate ;

    OldLeftCurrentEncoderTicks =  currentEncoderTics;

    return Vcurr;
}
```


```
61  void CopyISRCounters()
62  {
63      noInterrupts();
64      LeftMotorCounterCopy = LeftMotorCounter;
65      RightMotorCounterCopy = RightMotorCounter;
66      LeftWheelIsMovingForwardCopy = LeftWheelIsMovingForward;
67      RightWheeltIsMovingForwardCopy = RightWheeltIsMovingForward;
68
69      interrupts();
70  }
71
72  //=====================================
73
74  void LeftEncoderISR()
75  {
76      if(digitalRead(M1_PHASE_B))
77      {
78          LeftWheelIsMovingForward = true;
79          LeftMotorCounter ++;
80      }
81      else
82      {
83          LeftWheelIsMovingForward = false;
84          LeftMotorCounter --;
85      }
86
87  }
88  //=====================================
89  void RightEncoderISR()
90  {
91       if(digitalRead(M2_PHASE_B))
92      {
93          RightWheeltIsMovingForward = true;
94          RightMotorCounter --;
95      }
96      else
97      {
98          RightWheeltIsMovingForward = false;
99          RightMotorCounter ++;
100     }
101
102 }// end RightEncoderISR
103
```

```
void LeftMotor(int power)
{
    // if power >= 0 then move forward, direction = true
    // otherwise move backward, direction = false and make power positive
    //
    bool direction = false;

    if (power >= 0)
        direction = true;
    else
        power *= -1;

    if(power >  Max_Motors_Power)
            power =  Max_Motors_Power;

    if(direction)
    {
        digitalWrite(IN3, LOW);
        digitalWrite(IN4, HIGH);
    }
    else
    {
        digitalWrite(IN3, HIGH);
        digitalWrite(IN4, LOW);
    }

    analogWrite(ENB, power);
}
//==============================================================

void RightMotor(int power)
{
    // if power >= 0 then move forward, direction = true
    // otherwise move backward, direction = false and make power positive
    //
    bool direction = false;

    if (power >= 0)
        direction = true;
    else
        power *= -1;

    if(power > Max_Motors_Power)
            power = Max_Motors_Power;

    if(direction)
    {
        digitalWrite(IN1, LOW);
        digitalWrite(IN2, HIGH);
    }
    else
    {
        digitalWrite(IN1, HIGH);
        digitalWrite(IN2, LOW);
    }

    analogWrite(ENA, power);
}// end LeftMotor
```


```
//==========================================================
void MotorsInit()
{
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);

    pinMode(M1_PHASE_B, INPUT);
    pinMode(M2_PHASE_B, INPUT);

    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);
    pinMode(IN3, OUTPUT);
    pinMode(IN4, OUTPUT);

    LeftMotorCounter = 0L;
    RightMotorCounter = 0L;
}
```

Main.ino

```
#include "Motors.h"


#define VKp 10
#define VKd 0// vk/dt -0.5/50
#define VKi 0.0  //0.001

#define OKp 10
#define OKd 0.0
#define OKi 0.0

double OldError;

double ErrorSum;
unsigned long OldTimer, OldSpeedTimer ;

int ComSpeed;
double OldSpeedError;
double SpeedErrorSum;

double TurnReference;
double OldTurnError;
double TurnErrorSum;

double ReferenceAngleY;

double RightWheelSpeed, LeftWheelSpeed;

double correction;

int TurnCorrection_R;
int TurnCorrection_L;

//==============================================================
```


```
//=============================================================
int incomingByte = 0;
char letter = '\0';
String data = String('$');
String ComplitData = String();

void CommProcess()
{
  if(Serial.available() <= 0)
    {
      ComplitData = String('$');
      return;
    }
 if(Serial.available()>0)
  {
    int incomingByte = Serial.read();
    if(incomingByte != 95)
    {
      letter = incomingByte;
      if(data == "$")
      {
        data = String(letter);
      }
      else
      {
        data +=letter;
      }
    }
    if(incomingByte == 95)
    {
      ComplitData = String(data);
      data = String('$');

    }
  }
}
//=============================================================
```

```
74   int ComplitDataLength;
75   String ComSpeedSTR = String();
76   String TurnReferenceSTR = String();
77   int IndexOfT;
78
79   void Translation()
80   {
81     if(ComplitData != "$")
82     {
83       ComplitDataLength = ComplitData.length();
84       IndexOfT = ComplitData.indexOf("T");
85
86       ComSpeedSTR = String(ComplitData.substring(1,IndexOfT));
87       TurnReferenceSTR = String(ComplitData.substring(IndexOfT+1,ComplitDataLength));
88
89       ComSpeed = ComSpeedSTR.toInt();
90       TurnReference = TurnReferenceSTR.toDouble();
91
92
93     }
94
95       //Serial.print(ComSpeed);
96       //Serial.print('\n');
97   }
98   //==============================================================
99   int Er;
100
101  void StabilizeRobot()
102  {
103
104      double correction = ReferenceAngleY;
105      double FinalPowerR=((int)correction + (int)TurnCorrection_R );
106      double FinalPowerL((int)correction + (int)TurnCorrection_L);
107
108      RightMotor(FinalPowerR);
109      LeftMotor(FinalPowerL);
110
111  }
112
113  //==============================================================
114
115  void Print(double correction, double FinalspeedR, double FinalspeedL)
116  {
117    Serial.print(" FinalspeedR: ");
118    Serial.print(FinalspeedR);
119    Serial.print(" FinalspeedL: ");
120    Serial.print(FinalspeedL);
121    Serial.print(" correction: ");
122    Serial.print(correction);
123    Serial.print('\n');
124  }
```

```
124 }
125 void PrintSpeetControl(double RightWheelSpeed ,double LeftWheelSpeed,
126  double vCurr, double error, double Kcorrection, double Dcorrection, double OldSpeedError,
127  double SpeedErrorSum, double correction , int Er )
128 {
129   //Serial.print(" RightWheelSpeed: ");
130   //Serial.print(RightWheelSpeed);
131   //Serial.print(" LeftWheelSpeed: ");
132   //Serial.print(LeftWheelSpeed);
133   Serial.print(" vCurr: ");
134   Serial.print(vCurr);
135   Serial.print(" error: ");
136   Serial.print(error);
137   //Serial.print(" Kcorrection: ");
138   //Serial.print(Kcorrection);
139   //Serial.print(" Dcorrection: ");
140   //Serial.print(Dcorrection);
141   //Serial.print(" OldSpeedError: ");
142   //Serial.print(OldSpeedError);
143   //Serial.print(" SpeedErrorSum: ");
144   //Serial.print(SpeedErrorSum);
145   Serial.print(" correction: ");
146   Serial.print(correction);
147   Serial.print(" Er: ");
148   Serial.print(Er);
149   Serial.print('\n');
150
151 }
152
153
154 //==============================================================
155 void SpeedControl()
156 {
157   Er = (RightWheelSpeed - LeftWheelSpeed) / (RightWheelSpeed /ComSpeed);
158
159
160
161   double vCurr = (RightWheelSpeed + LeftWheelSpeed) / 2.0;
162   double error = (ComSpeed - vCurr);
163
164   double Kcorrection = VKp * error;
165    //
166    if(ComSpeed >=0 && Kcorrection < 0)
167     Kcorrection = 0;
168    else if(ComSpeed <= 0 && Kcorrection > 0)
169     Kcorrection = 0;
170
171
172   double Dcorrection = VKd * (error - OldSpeedError);
173
174   OldSpeedError = error ;
175
176   correction = Kcorrection + Dcorrection;
177
178   ReferenceAngleY = correction;
179
180
181 }
```

```
//---------------------------------------
void setup()
{
  MotorsInit();
  Serial.begin(9600);

  ErrorSum = SpeedErrorSum = TurnErrorSum = 0.0;
  ReferenceAngleY = 0;
  ComSpeed = 0.0;
  TurnReference = 0.0;
  TurnCorrection_L = TurnCorrection_R = 0;
  RightWheelSpeed = LeftWheelSpeed = 0.0;
  Er = 0;
  attachInterrupt(digitalPinToInterrupt(M1_PHASE_A), LeftEncoderISR, RISING);
  attachInterrupt(digitalPinToInterrupt(M2_PHASE_A), RightEncoderISR, RISING);
  OldTimer = OldSpeedTimer = millis();
}

//---------------------------------------
void loop()
{
  CommProcess();
  Translation();



    unsigned long currMillis = millis();
    if(currMillis>10000){
    if(currMillis - OldTimer >= 10L)
    {
        OldTimer = currMillis;
         StabilizeRobot();
    }
   if (currMillis - OldSpeedTimer >= 50L)
    {
      OldSpeedTimer = currMillis;
      CopyISRCounters();

      RightWheelSpeed = GetRightWheelSpeed();
      LeftWheelSpeed = GetLeftWheelSpeed();

      SpeedControl();

       double omegaCurr = (RightWheelSpeed - LeftWheelSpeed) / DISTANCE;
       double error = TurnReference*10-omegaCurr;
       double Kcorrection = OKp * error;
       double Dcorrection = OKd * (error - OldTurnError);
       OldTurnError = error;
       TurnErrorSum += OKi * error;
       double correction = Kcorrection + Dcorrection + TurnErrorSum;

       TurnCorrection_R = correction;
       TurnCorrection_L = -correction;



    }
    }

}
```