**מערכת לניהול מרוץ מכוניות קו ישר**

פרויקט מסכם בקורס "אפיון ותכנון מערכות משובצות מחשב"

**תיאור הפרויקט**

הפרויקט שלנו הוא מימוש מערכת לניהול מרוץ מכוניות צעצוע במסלול קו ישר, המערכת יודעת להזניק מכוניות צעצוע ע"י אפשור תנועתם בעזרת שער ממונע על מדרון משופע המתוזמן ע"י קלט מן המשתתפים במרוץ. המשתתפים ישלטו על הזנק הרכבים ע"י חייוי ויזואלי מרמזור לדים צבעוניים המדמה רמזור אמיתי של מסלול דראג, בעת סיום חייוי האור המשתתפים יזניקו את רכביהם והמערכת תמדוד את ביצועי זמן התגובה של המשתתפים מול חיווי האור.

לאחר אפשור תנועת הרכבים הם יגיעו באופן טבעי לסוף המסלול ו"חציית קו הסיום" תורגש ע"י נגד תלוי אור (LDR) אשר יידגם בקצב גבוה ע"י המערכת על מנת לזהות צל אשר יוטל על הנגד ע"י הרכב החוצה. ברגע חציית קו הסיום המערכת תשמור את זמן סיום המסלול של כל משתתף. במידה ושני המשתתפים חצו את קו הסיום בהצלחה בזמן המוגדר למרוץ המערכת תחליט מי מן המשתתפים ניצח במקצה זה לפי זמן ההגעה לקו הסיום הנמוך מבין ה 2 ותשמור את הניקוד המתעדכן לשני המשתתפים.

הקלט והפלט מן המערכת ייעשה ע"י אפליקציית Android אותה מימשנו בעצמינו בעזרת פלטפורמת “MIT App Inventor” . האפליקציה תתחבר אל המערכת דרך ממשק Bluetooth אל כרטיס מתאם Bluetooth – UART אשר מתקשר עם המיקרובקר של המערכת באופן רציף.

האפליקציה מנהלת את הקלט מן המשתתפים להזנקת הרכבים וגם מנהלת את ניקוד המשתתפים המתעדכן.

**פירוט טכנולוגי**

**מכונת המצבים**

את הקוד הרץ על המיקרו בקר מימשנו ב 2 צירים מקביליים –

1. פסיקה בעקבות תקשורת עם האפליקציה.
2. מכונת מצבים ראשית הרצה ב Main.

תחילה נסביר על מכונת המצבים הראשית

Diagram, text

Description automatically generated

מכונת המצבים הראשית מנהלת את הממשק הפיזי של המערכת הכולל מנועים, לדים, דגימות מתח וטיימר.

**מצב READY**

מצב זה מאפס את המערכת לקראת תחילת המקצה, האיפוס מתבטא בהבאת השערים (מנועים) למצב של חסימת האוטו, כיבוי הרמזור ,איפוס הרגיסטרים ששומרים את הזמנים וכיול ה LDR'ים. היציאה ממצב זה מתבצעת ע"י קלט מהמשתתף בלבד (דרך ה BT).

case READY:

LIGHTS\_OFF();

dark\_off1 = finish\_line\_LDR(0);

dark\_off2 = finish\_line\_LDR(1);

PWM\_Servo(0, CLOSED);

PWM\_Servo(1, 120);

response\_car1 = 0;

response\_car2 = 0;

total\_car1 = 0;

total\_car2 = 0;

**מצב LIGHTS**

במצב זה מתחילים את רצף הדלקת הלדים ברמזור – צהוב1 – צהוב2 – ירוק. לאחר הדלקת הלד הירוק אנו רוצים להתחיל למדוד את זמן התגובה של המשתתפים, לשם כך מאפסים ומאפשרים את Timer0 של המיקרובקר אשר מקדם מונה זמן כל 1ms. לאחר אפשור המונה עוברים ישירות למצב RACING

case LIGHTS:

LIGHTS\_START();

if(state == FAULT) *//PRE LAUNCH*

break;

TIMER0->TCSR.CRST = 1; *//Reset up counter*

Timing\_Counter\_0 = 0;

TIMER0->TCSR.CEN = 1; *//Enable Timer0*

state = RACING;

break;

**מצב RACING**

מצב זה מתאר את מהלך המרוץ עצמו. במצב זה אנו דוגמים את 2 ה LDR'ים דרך שני ערוצי A2D של המיקרובקר וחשים האם קיימת עליה במתח הנדגם עליהם (התנגדותם עלתה בעקבות צל). במידה וכן, שומרים את זמן המקצה של המשתתף ושולחים אותו לאפליקציה בעזרת פקודה מתאימה. במידה ושני המשתתפים סיימו עוברים למצב DONE, במידה ועבר הזמן המותר למקצה – שולחים איתות כי המקצה תקול ועוברים למצב FAULT, חישוב המנצח יעשה באפליקציה.

case RACING:

if(total\_car1 == 0 && response\_car1 > 0){

light\_sns = finish\_line\_LDR(0);

if(light\_sns > (dark\_off1 + DARK\_SNS)){ *//Car1 Done*

total\_car1 = Timing\_Counter\_0;

send\_UART("T1", total\_car1);

}

}

if(total\_car2 ==0 && response\_car2 > 0){

light\_sns = finish\_line\_LDR(1);

if(light\_sns > (dark\_off2 + DARK\_SNS)){ *//Car2 Done*

total\_car2 = Timing\_Counter\_0;

send\_UART("T2", total\_car2);

}

}

if(total\_car2 != 0 && total\_car1 != 0){ *//both done*

state = DONE;

response\_car1 = 0;

response\_car2 = 0;

total\_car1 = 0;

total\_car2 = 0;

}

else if(Timing\_Counter\_0 > 4000){ *//Time over*

state = FAULT;

if(total\_car1 == 0)

send\_UART("F1", 0000);

if(total\_car2 == 0)

send\_UART("F2", 0000);

}

**מצב DONE ומצב FAULT**

במצבים אלו לא קורה הרבה, אלו הם מצבי "המתנה". במצב DONE מכבים את הרמזור וממתינים שקלט מן המשתמש יחזיר את מכונת המצבים להתחלה, במצב FAULT מהבהבים את הלד האדום ברמזור ומחכים שקלט מן המשתתף יחזיר את מכונת המצבים להתחלה.

case DONE:

LIGHTS\_OFF();

break;

case FAULT:

LIGHTS\_FAULT();

break;

**פסיקת UART**

כפי שצוין קודם, החלק השני שמנהל את המערכת הוא פונקציית הפסיקה של תקשורת ה UART עם הרחבת ה Bluetooth, הכניסה לפונקציה זו היא ברגע שיש בתים מוכנים לקריאה דרך ה UART. כאשר יש בתים מוכנים נרצה להגיב אליהם במהירות בגלל שהקלט על זמן תגובת המשתתפים להזנקה מתבצע בממשק זה. מבנה התקשורת בין המערכת לבין האפליקציה נראה כך

Graphical user interface, text, application, chat or text message

Description automatically generated

החלק העליון מתאר את פונקציית הפסיקה. פונקציה זו מבצעת פעולות שונות בהתאם למצב הנוכחי במכונת המצבים הראשית. החלק התחתון כולל את התקשורת המגיעה מפונקציית ה Main.

קריאת בתים מן הרגיסטרים של ממשק התקשורת מתבצע ע"י קריאת הבתים בצורה טורית עד אשר מתקבל תו 'X', בצורה זו אנו יכולים לקרוא כמות משתנה של בתים.

while(UART0->ISR.RDA\_IF==1) {

comRbuf[comRbytes]=UART0->DATA;

comRbytes++;

if(comRbuf[comRbytes-1] == 'X'){ *//Data must end with 'X'*

comRbuf[comRbytes-1] = '\0';

sprintf(rCMD,"%s",comRbuf);

comRbytes = 0;

**קלט READY**

כאשר משתתף באפליקציה לחץ על כפתור "READY" המערכת מקבלת את הבתים בהתאם, בקבלת בתים אלו הפונקציה תעביר את מכונת המצבים הראשית למצב READY.

if(!strcmp(rCMD, "READY")){ *//READY FOR RACE*

state = READY;

}

**קלט START**

קלט זה מתקבל לאחר שמשתתף לחץ באפליקציה על כפתור "START", כפתור זה מאותת על מוכנות המשתתפים להזנקה. הפקודה מעבירה את מכונת המצבים הראשית למצב LIGHTS.

else if(!strcmp(rCMD, "START")){ *//START RACE*

if(state == READY )

state = LIGHTS;

}

**קלט הזנקה**

ברגע שמשתתף לוחץ על כפתור ההזנקה שלו נקבל את הבתים "L1" או “L2” בהתאם למשתתף. בקבלת פקודה זו נרצה קודם כל לבדוק האם המשתתף לחץ על הפקודה לפני הזמן (לפי הדלקת האור הירוק ברמזור), במידה וכן המשתתף שפקודתו התקבלה נפסל, נשדר את המשתתף הנפסל לאפליקציה ע"י שליחת “F1” או “F2” ונעביר את מכונת המצבים הראשית למצב FAULT.

במידה והפקודה שהתקבלה הגיעה בתזמון נכון נרצה לשמור את זמן התגובה של המשתתף ע"י שמירת זמן הטיימר המאותחל ,נרים את שער המסלול הרלוונטי ונשלח את זמן התגובה של המשתתף ע"י “R1” או “R2” בצירוף עם זמן התגובה במילישניות (עד 4 ספרות).

else if(!strcmp(rCMD, "L1")){ *//CAR1 LAUNCHED*

if(state != RACING){

state = FAULT;

send\_UART("F1", 0000);

break;

}

if(response\_car1 != 0)

break;

response\_car1 = Timing\_Counter\_0;

send\_UART("R1", Timing\_Counter\_0);

PWM\_Servo(0, OPEN);

}

else if(!strcmp(rCMD, "L2")){ *//CAR2 LAUNCHED*

if(state != RACING){

state = FAULT;

send\_UART("F2", 0000);

break;

}

if(response\_car2 != 0)

break;

response\_car2 = Timing\_Counter\_0;

send\_UART("R2", Timing\_Counter\_0);

PWM\_Servo(1, 80);

}

**דגימת A2D**

את דגימת ה LDR'ים אנו מבצעים ע"י דגימת מפל המתח עליהם בחיבור מחלק נגדים מול נגד 10K המחובר למתח ה 5V של המעגל.

כדי לשקף את זמן המרוץ המדויק ביותר אנו צריכים שהבדיקות יתבצעו במהירות, לכן אנו עובדים עם ה A2D במצב Continuous ומאפשרים אך ורק את שני הערוצים הנדרשים למערכת.

DrvADC\_Open(ADC\_SINGLE\_END, ADC\_CONTINUOUS\_OP, 0x03, INTERNAL\_HCLK, 1); *// ADC1= LDR1 & ADC0 = LDR0*

double finish\_line\_LDR(uint8\_t car)

{

int16\_t adc\_value;

DrvADC\_StartConvert();

while(DrvADC\_IsConversionDone()==FALSE);

*// wait till conversion flag = 1, conversion is done*

adc\_value = ADC->ADDR[car].RSLT & 0xFFF;

*// input 12-bit ADC value*

ADC->ADSR.ADF=1; *// write 1 to clear the flag*

return adc\_value;

}

**האפליקציה**

כפי שצוין קודם, את האפליקציה ביצענו בעזרת כלי נוח המאפשר יצירת אפליקציות בקלות בלי ידע רב – MIT App inventor

**התחברות BT**

את ההתחברות מבצעים ע"י רשימת המכשירים שהמכשיר מזהה-

Timeline

Description automatically generated

**אתחול והתחלת המירוץ**

את המעבר למצבי READY ו- LIGHTS במכונת המצבים מבצעים ע"י לחיצה על כפתורים באפליקציה

Graphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

**הזנקת מכוניות**

הזנקת המכוניות שולחת את הבתים שהוסברו בעבר עם צירוף התו 'X':

Timeline

Description automatically generated

**זיהוי בתים נקלטים**

בדיקת הבתים המתקבלים מתבצע בתהליך מתוזמן כל חצי שניה הבודק האם יש בתים לקריאה ואז בודק איזו פקודה התקבלה

Timeline

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

**חישוב ניקוד**

בעקבות זמן המקצה הנשמר לכל משתתף האפליקציה יכול להסיק מי המשתתף הזוכה, האפליקציה מונה ניקוד לכל משתתף אותו היא מעלה באחד עבור כל ניצחון, פסילה נותנת נקודה למשתתף השני, בדיקה זו מתבצעת גם כן על אותו שעון של כל חצי שניה

Graphical user interface, application

Description automatically generatedGraphical user interface, application, table

Description automatically generated

לבסוף האפליקציה נראית כך-

Chart, waterfall chart

Description automatically generated

כאשר משתתף מנצח או נפסל יש תמונה של דגל מרוץ או דגל אדום בהתאמה אך לא רואים זאת בתמונה זו.

**סכמת המערכת**

Diagram, schematic

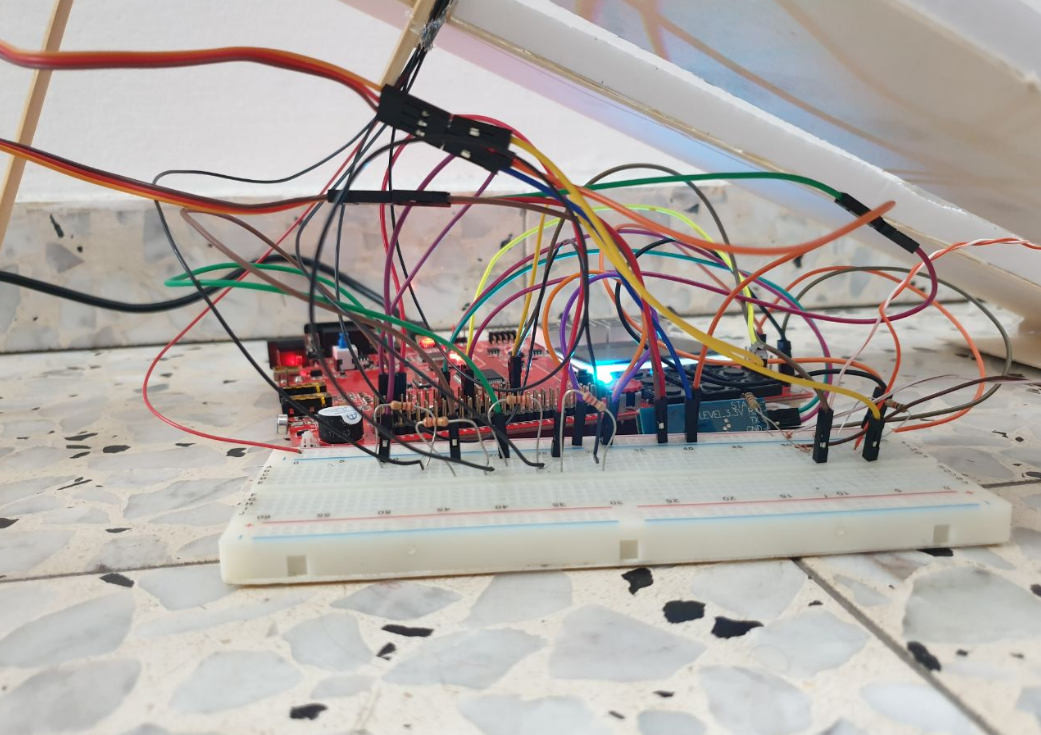
Description automatically generated

**תמונות המערכת**

A picture containing indoor

Description automatically generated A picture containing indoor

Description automatically generated



**מקצה תקין**

במקצה תקין בו שני המשתתפים הגיעו לקו הסיום לפני סוף זמן המרוץ האפליקציה תוסיף נקודה למשתתף עם זמן המקצה הקצר יותר ותאותת עם תמונה של דגל מרוץ:

Timeline

Description automatically generated with medium confidence

**מקצה פסול**

במידה ואחד המשתתפים לחץ לפני הזמן (לפי האור הירוק ברמזור) או שאחד המשתתפים לא סיים את המקצה עד סוף זמן המרוץ, המשתתף שלא ביצע את העבירה יקבל נקודה והאפליקציה תאותת עם דגל אדום, במקרה זה גם הרמזור יאותת באור אדום:

Timeline

Description automatically generated A picture containing indoor

Description automatically generated