



מס' פרויקט: 19-1-1-1829  
מבצעים: הראל חי | הראל חכם  
מנחה: ד"ר אנטולי חינה  
מקום ביצוע הפרויקט: אוניברסיטת ת"א

# סקירה כללית



- רובוט מלהטט הוא רובוט המתוכנן לכדרר או ללהטט בהצלחה כדורים או חפצים אחרים.
- במסגרת הפרויקט עבדנו עם רובוט קולבורטיבי דגם UR3 של חברת Universal Robots
- דגם זה בעל יתרונות רבים:
  1. עלות הזרוע היא נמוכה
  2. אורכה הוא 500 מ"מ
  3. בעלת 6 צירים
  4. מסוגלת להרים משקל כולל של עד 3 ק"ג
  5. מאפשרת יכולות מגוונות כגון: צביעה, משטוח, אריזה, הרכבה, שיוף פעולות הדבקה ועוד.

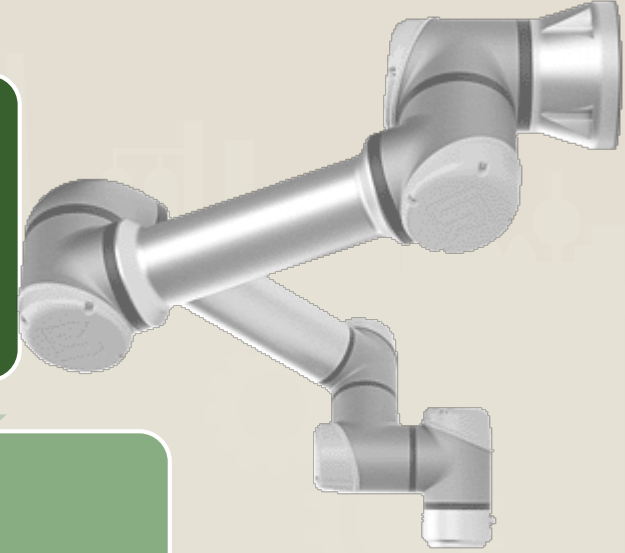
מסיבות אלו הזרוע הרובוטית נפוצה כמעט בכל מפעל.

# מוטיבציה

הזרוע מסוגלת לתקשר במהירות של 500Hz אך בפועל תקשרה במהירות של 10Hz בלבד - מיקסום מהירות התגובה של המערכת יוביל לתיקון שגיאות ותגובה בזמן אמת של מערכת בקרה בחוג סגור.

הפתרונות הנפוצים כיום מבוססים על תוכנות קנייניות המייקרות את הפעלת הזרוע הרובוטית ולעיתים אף דורשות דמי מנוי ותחזוקה.

שיפור ניכר במהירות התגובה של הזרוע ופיתוח יכולת בקרה בחוג סגור יעילה וזולה (שליחה וקבלת מידע אודות מיקום הזרוע בזמן אמת) יכולה לתרום רבות לתעשייה.





השוואת פתרונות קיימים

---

כנגד פתרון הפרויקט



RoboDK

מערכת הפעלה חינמית מבוססת open source.

החברה מוכרת פתרון של בקר בזמן אמת (חומרתי) המתממשק עם רובוטים של מספר חברות פופלריות (בניהן UR) הבקר והתוכנה שלו נותנים פתרון לבקרה בחוג סגור בזמן אמת

תוכנה בתשלום אשר נותנת אפשרות שליטה במספר רב של רובוטים דרך המחשב באמצעות כלים גרפיים.

תיאור כללי

עלות זמן פיתוח

עלות המוצר (לא מפורסם)

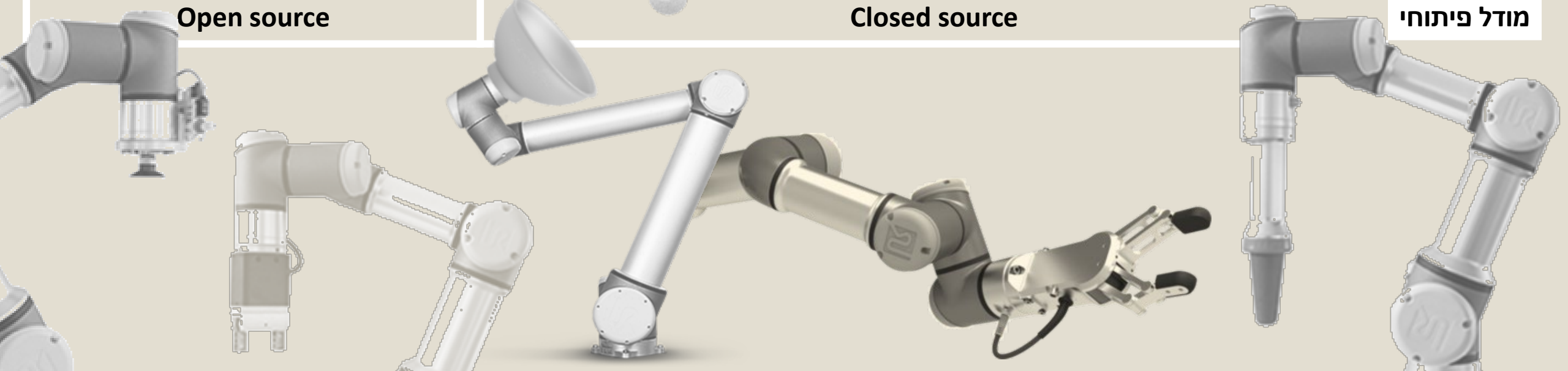
עלות מוצר - 3000 יורו (כולל מנוי לשנה) עלות מנוי של 1250 יורו לשנה.

עלות

Open source

Closed source

מודל פיתוחי





**חינמי**

**שליטה במס' רובוטים במקביל**

**קהילת המפתחים גדולה וגדלה**

**שירות תמיכה טכנית**

**מגוון כלים וחבילות בפלטפורמה**

**שירות התאמה אישית לצרכי הלקוח**

**ניתן לראות, לבנות וללמוד מקוד קיים ולהעזר בו כדי לשפר את הקוד הדרוש (open source)**

**פתרון קיים וישים עבור הבעיה הקיימת שלנו**

**תמיכה במגוון גדול של תוכנות (כגון: CAD, SolidWorks...)**

**ממשק משתמש מלוטש ונוח.**

**אופטימזצית זמן מחזור**

**מעבד חזק ומהיר ובקרה בזמן אמת**

**ניתן להגיע לדיוק תנועה גבוה (כלי קליברציה)**

**יתרונות**

**עלות זמן הפיתוח**

**תוספת חומרית (מייתר את הבקר נוכחי)**

**לא קיימת שליטה בזמן אמת (שליטה ב- URP and SCRIPT files)**

**מוגבלות בתחום הפיתוח (Closed source) יישום פתרונות בהתאמה אישית תלויה בשירות החברה**

**חסרונות**

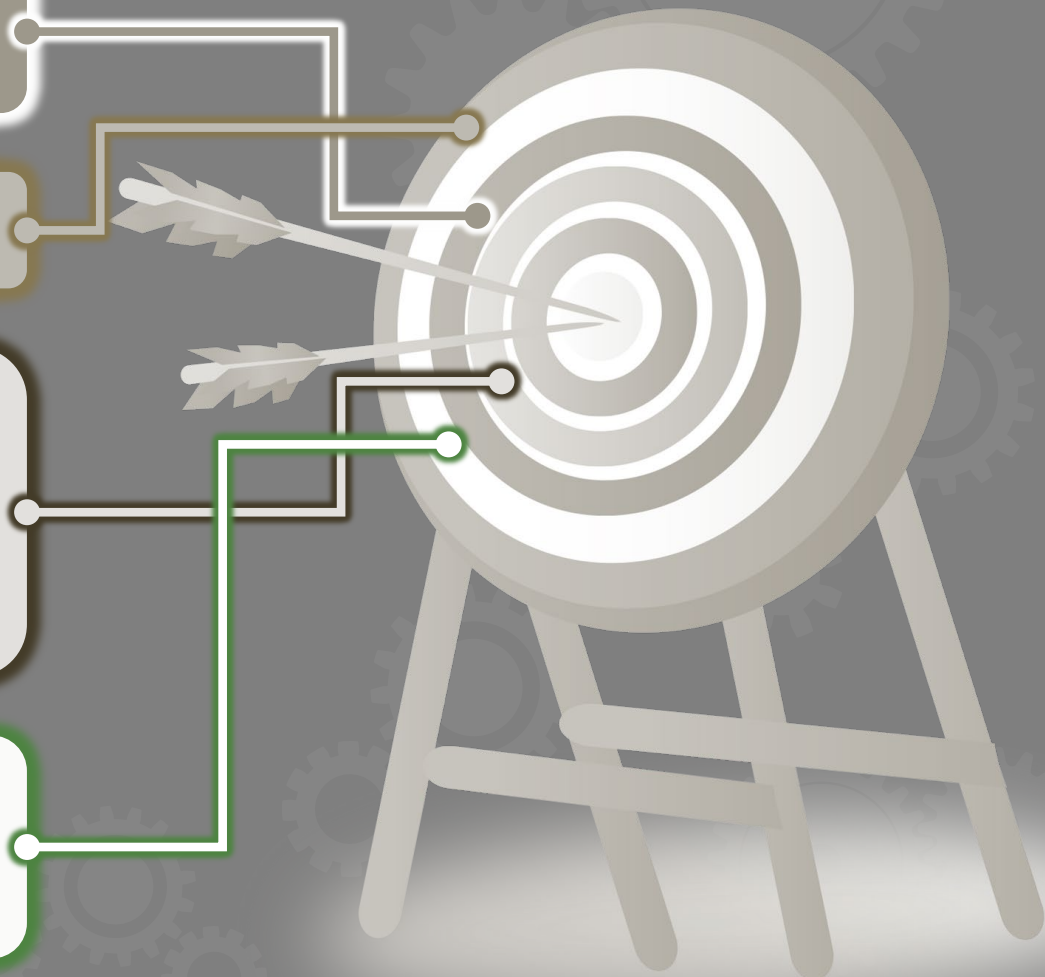
# מטרות

מיקסום יכולת השליטה בזרוע הרובוטית: קבלת ושליחת פקודות בזמן אמת - בקצב של 500Hz.

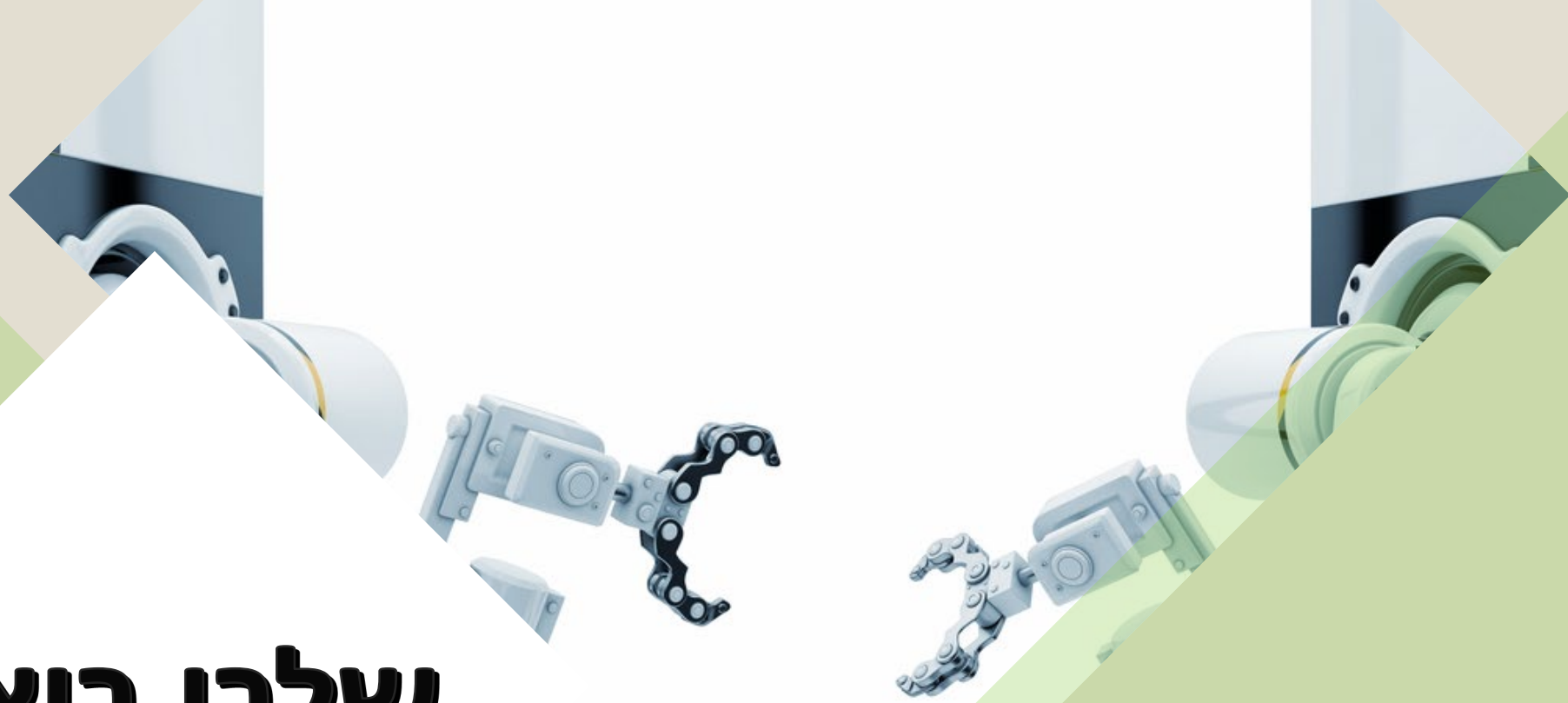
שיפור יכולות הזרוע להתגבר על בעיות תזמון וסנכרון.

הקמת סביבת עבודה ב- ROS ויצירת בסיס לבקרה בחוג סגור בזמן אמת תוך יצירת סביבת עבודה נוחה למשתמש הקצה.

ייצור התשתית הדרושה לצורך המשך הפרויקט (על בסיס פרויקט זה).



# שילבי ביצוע





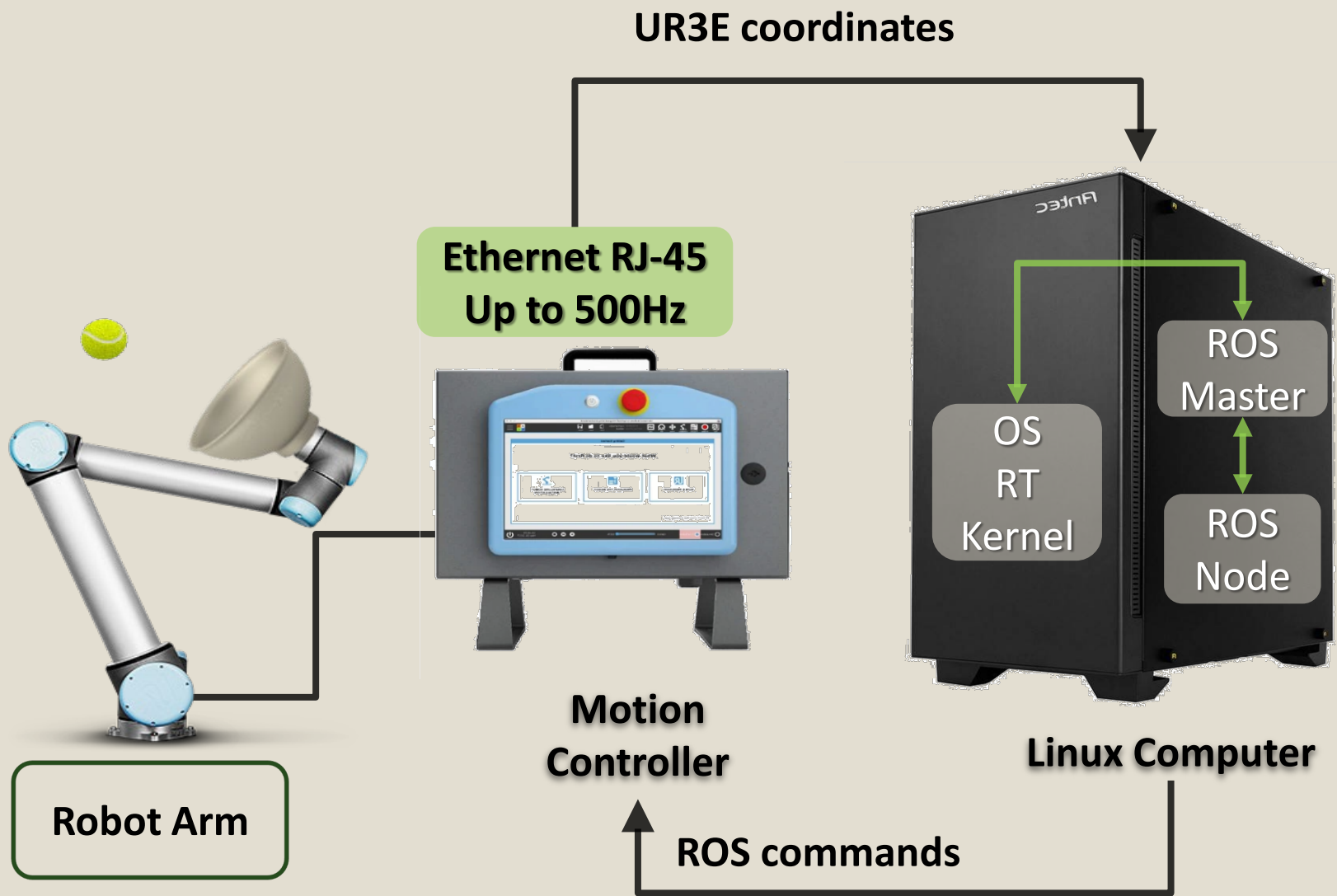
- לאחר שלמדנו את הרקע התיאורטי, מצאנו כי ניתן לבצע שליטה על הזרוע באמצעות מספר דרכים ושתי סביבות עבודה עיקריות:
  1. ממשק ישיר (server client).
  2. ממשק ROS.
- החלטנו לנסות שני ממשקים לשליטה בזרוע.
- ביצענו השוואה, לצורך בחירה בסביבה אשר תהווה בסיס נוח לפרויקטים הבאים.



לטבלת ההשוואה

- החלטנו להמשיך ולפתח את סביבת העבודה ROS שכן שמנו דגש גדול יותר על נוחות שימוש ואפשרויות פיתוח מורכבות בעתיד.

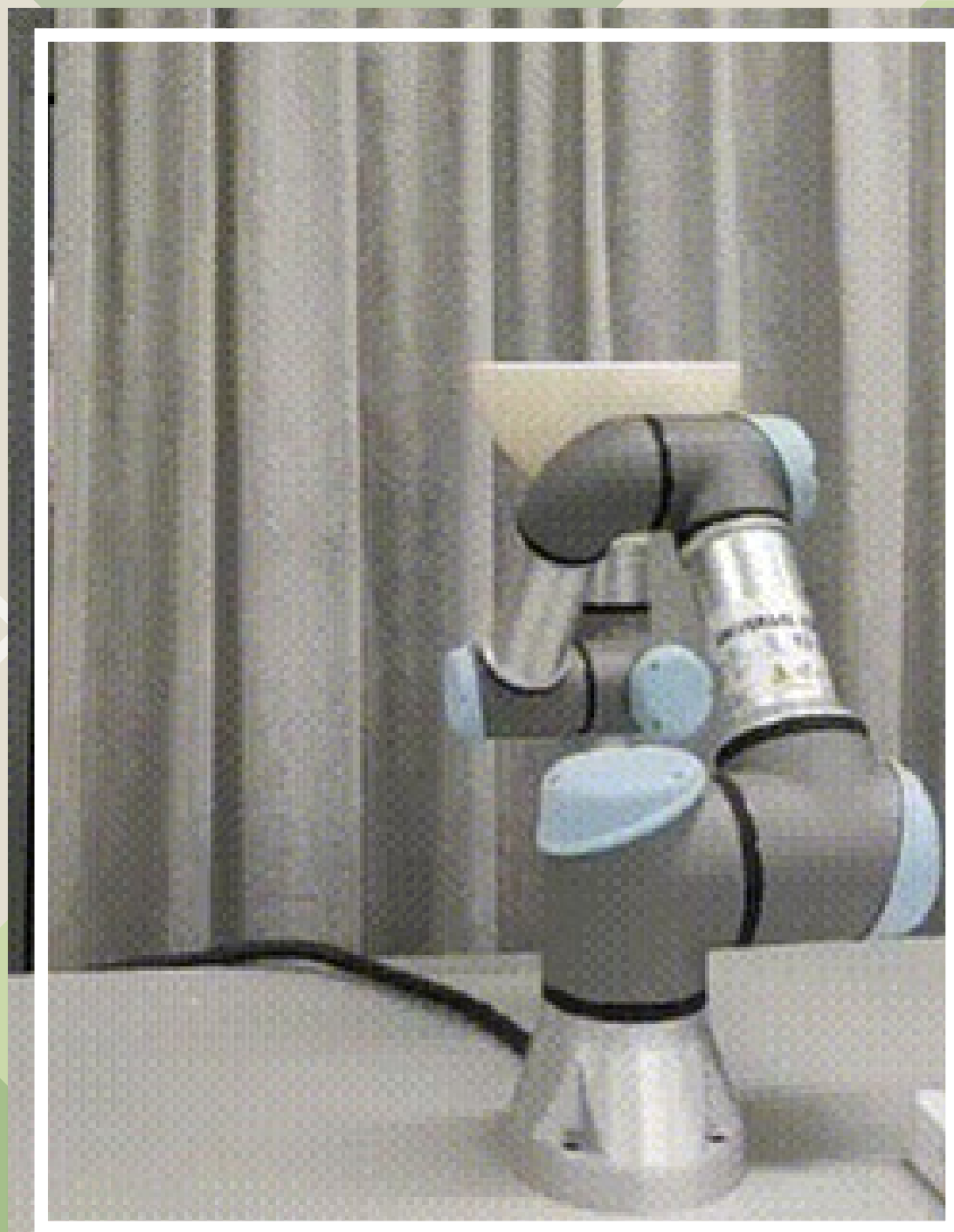
- התקנת Real Time kernel של Ubuntu 18.4 – לצורך שליטה במהירות מקס' של עד 500Hz
- ביצוע התאמות בסביבת עבודה ROS – התקנת מערכת ROS גרסת MELODIC
- הקמת סביבת עבודה – הקמת catkin workspace ועדכון משתני הסביבה ב-ROS.
- חבילות ROS – הכוללות דרייברים של הבקר, כלים גרפיים לשליטה בזרוע, כלי תכנון מתקדמים ועוד.
- בדיקת יכולות זמן אמת - באמצעות שליחת פקודה וקבלת מידע אודות מיקום הזרוע
- ביצוע הרצות ראשוניות – התגברנו על מספר בעיות וביניהן אי התאמות של קבצי הבקרים, ובחבילות לא מעודכנות שהתמיכה בהן ירדה.



# הקמת סביבת ROS - 2

- בניית אלגוריתם המאפשר לחקות את תנועות הזריקה שקודמינו תכננו בעזרת החבילות הקיימות.
- החלטנו להשתמש בבקרת מיקום המקבלת מערך של מיקומים וזמן לביצוע התנועה.
- הכנו אלגוריתם המקבל מהבקר את מיקום הזרוע ולפיו קובע אם הסתיימה התנועה.
- כתבנו תוכנית לתזמון התנועות בעזרת נתוני מיקום הזרוע, בעזרתו שיפרנו את יכולת הסקריפט לזרוק ולתפוס.

# תוצאות ומסקנות



# 1 Real Time

הטמעת יכולות זמן האמת של ה UR3E הייתה הצלחה. ראינו אישורים ליכולות ה RTDE גם מתוך הדרייברים וגם תוך כדי תקשורת עם הזרוע המתבטאים בזרם הודעות אל ומהזרוע בקצב 500Hz. וכן ההבדל ניכר בתגובה המיידית בין שליחת פקודה לביצועה, מבקר ה ROS אל בקר הזרוע.

# 2 ROS

הטמעת סביבת העבודה של ROS הושלמה ודרכה הצלחנו לממש מגוון פתרונות שליטה על הזרוע:

שליטה באמצעות בקר ROS – פרסום ישיר של ROS messages דרך topics של בקרי ה-ROS אל בקר הזרוע. אופציה זו נראית מבטיחה להמשך הפרויקט, שכן אפשרויות הבקרה בחוג הסגור מוכרות בקהילת ROS.

שליטה באמצעות מעטפת MoveIt – מעטפת קוד נחה לשימוש, עם פוטנציאל פיתוחי טוב להמשך בממשק קוד ה C++

שליטה גרפית – באמצעות תוכנת RVIZ, מאפשרת GUI ידידותי למשתמש. התוכנה תהיה שימושית יותר למידול סביבת הזרוע מאשר לבקרה עליה.

# שליטה ובקרה

# 3

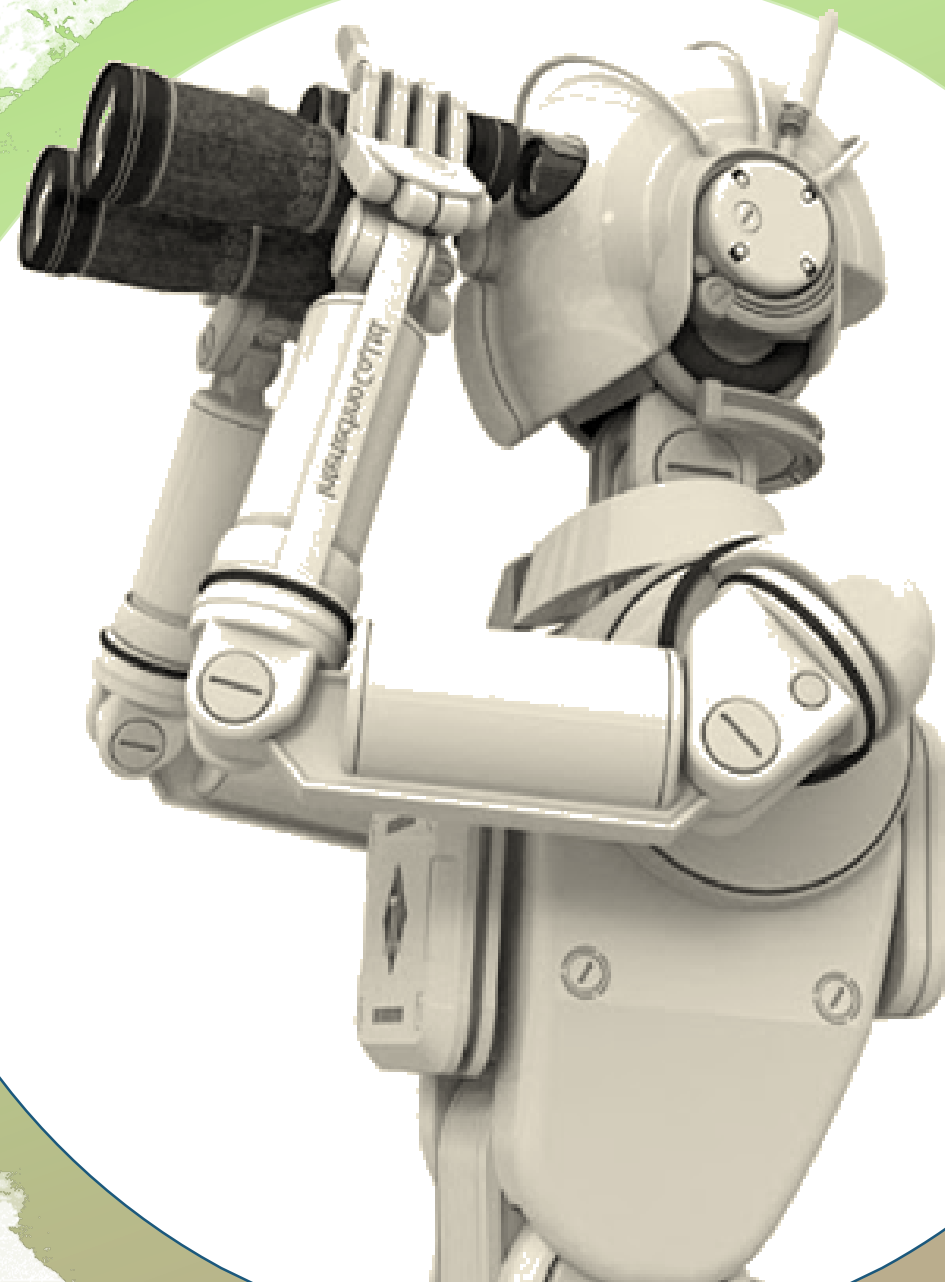
בקר ה ROS ואלגוריתם המיקום שבנינו מאפשר לחקות את התנועות של הזריקה שתוכננה על בקר UR3E בצורה יעילה אך לא מושלמת.  
אנו סבורים שכאשר הקוד רץ מקומית על הבקר של UR3E לבקר יש מידע על התנועה הבאה מראש הוא מסוגל לחשב כיצד לשלב בין סופי התנועות כך שמקבלים תנועה רציפה יותר.



# אפשרויות

## לעתיד...

- שילוב מערך מצלמות לסגירת חוג הבקרה.
- נמליץ על המשך פיתוח משולב הבוחן את האפשרויות ש ROS וחבילותיו מציעים
- נרצה לעבור בקרת מהירות המתאימה יותר לבקרה הנדרשת בחוג סגור.





**תודה על  
ההקשבה**