Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

הפקולטה להנדסה
ע"ש איבי ואלדר פליישמן
אוניברסיטת תל-אביב

TEL AVIV UNIVERSITY

Juggling Robot

Project Number: 18-1-1-1623

By: Omer Ben-Nun , Ehud Hayat Advisor: Dr. Anatoly Khina

Project Carried Out at Tel Aviv University

A robotic arm system that receives commands using a custom Python interface via TCP/IP protocol, and can juggle a ball.

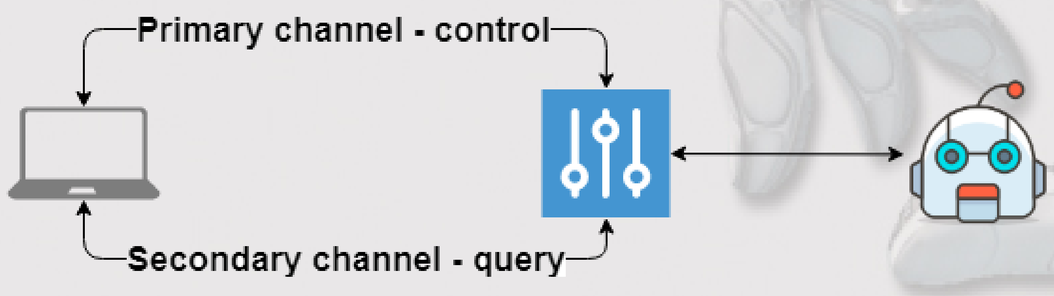
Overview

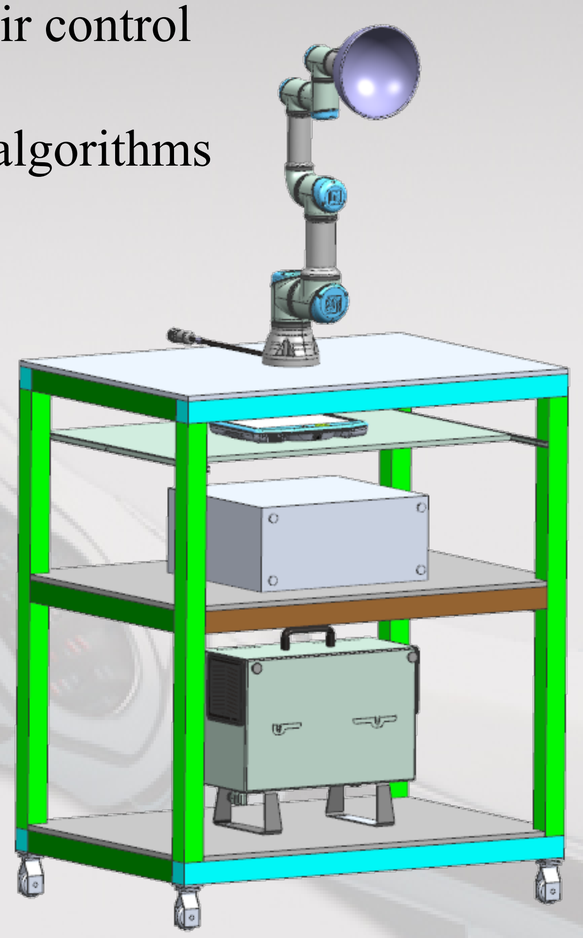
- A juggling robot is a robot which can successfully throw and catch balls or other objects
- These robots usually depend on optical sensors in a closed loop to function properly
- The first juggling robot was built by Claude Shannon (1916-2001) and was able to juggle 3 balls.
- Robotic arms today are paired with a controller, which gives instructions to the arm itself.
- We have implemented an open loop system, with possibility to close a control loop in the future.

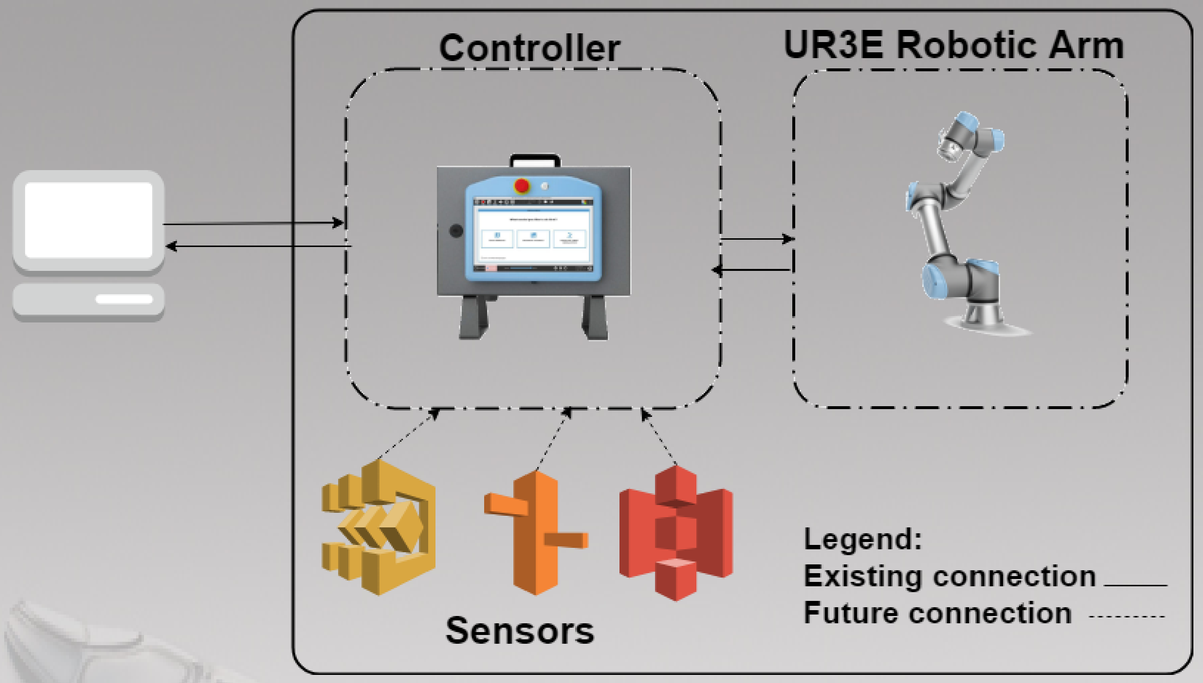
Claude Shannon, juggling

Motivation

- Studying human motion.
- Understanding advantages of closed loop control
- Gaining understanding of sensorimotor control
- Studying theory of robotics and overcoming engineering challenges in their control
- Creating a modular system for future implementation of complex control algorithms



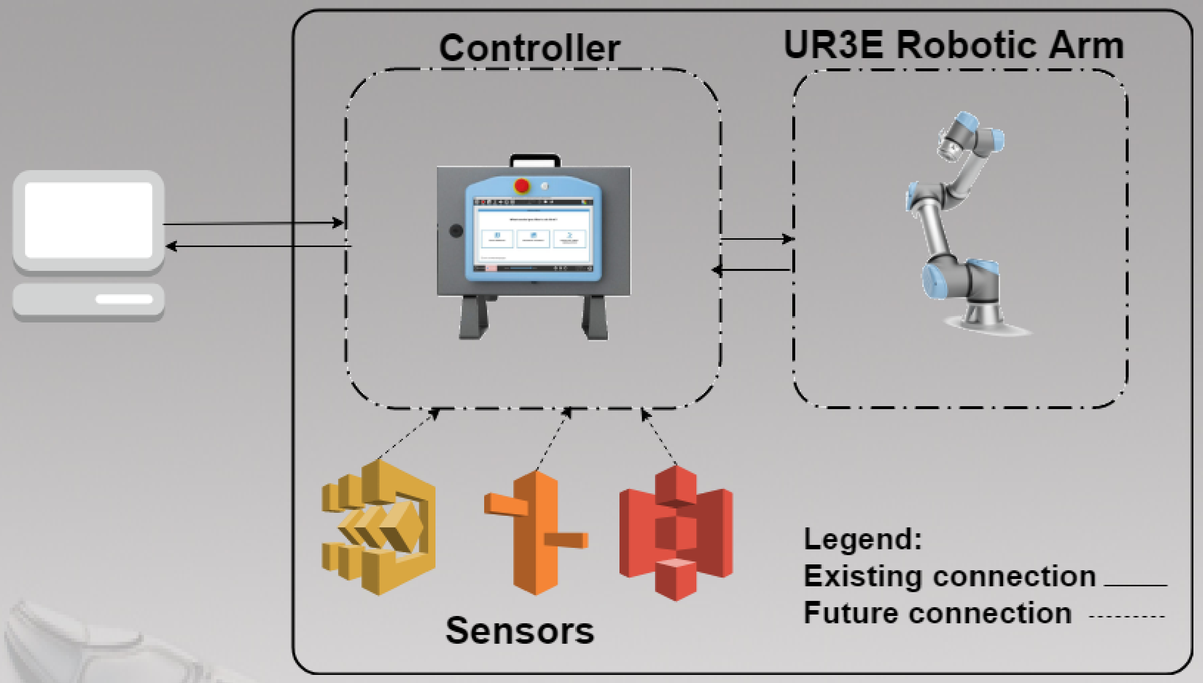




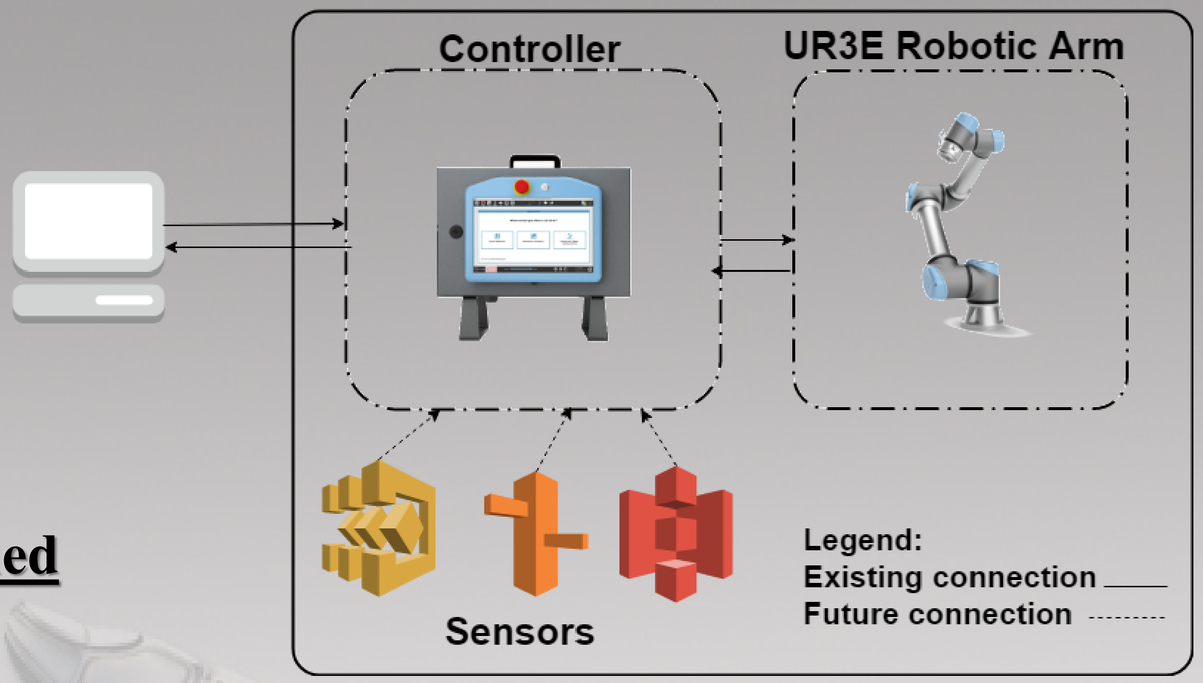
Implementation

- Computer communicates with robotic-arm controller
- Communication is via TCP/IP protocol, over Ethernet
- High communication frequency of up to 500 [Hz]
- The UR3E robot is a Cobot; a robotic arm that is intended to physically interact with humans in a shared workspace

Implementation - hardware



- UR3E robotic arm
  - A collaborative robotic arm
  - Can be controlled directly from a touchpad, or by computer over TCP/IP protocol
- UR3E controller
  - All instructions to the arm pass through the controller
  - Has many inputs and outputs, to which sensors can be connected
- Control touch pad
  - Easy and intuitive control of the robot directly
  - Very efficient for simple programs and tasks

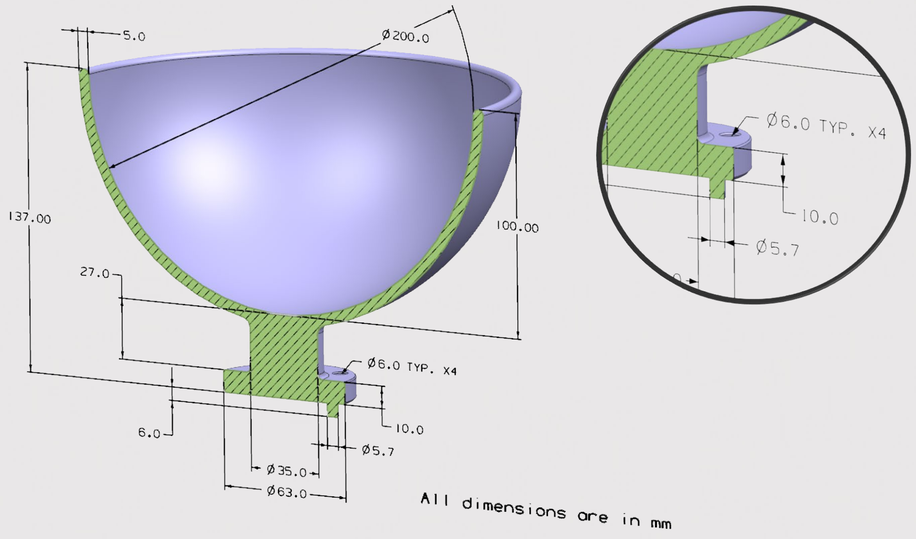


Implementation – hardware, continued

- Computer
  - Strong HW, able to run complex algorithms quickly
  - Communicates with controller over TCP/IP protocol
- Sensors
  - May be optical, or any other kind
  - Control with sensors is not implemented in the first phase of the project

What have we done:

- Establish system requirements for desired performance
- Survey a variety of robots and purchase the best one
- Create and print a 3D grip model
- 3D modeling of a table for robotics' arm lab
- Develop user friendly control environment in Python language
- Conduct system ‘bring up’
- Juggle using a computer via network connection



Robotic arm survey

- The first stage of our project was understanding our needs and conducting a survey of our possibilities
- UR3E was our best option, in most aspects
  - Control is done by sending scripts over TCP/IP, which is compatible with any programming language
  - It is collaborative, therefore much safer
  - Cheaper than other alternatives

	UNIVERSAL ROBOTICS - UR3e (UR, n.d.)	KUKA - KR3 (KUKA, n.d.)	Denso - Vs050 (Denso, n.d.)
מחיר משוער	₪70,000	€20,000	€20,000
משקל מקסימלי להרמה	3 ק"ג	3 ק"ג	4 ק"ג
מהירות תקשורת	TCP/IP 100 Mbit	12 ms latency	12 ms latency
עבודה בסביבת אנשים	אינו זקוק לכלוב לעבודה	זקוק לכלוב	אינו זקוק לכלוב לעבודה
דיוק	0.03mm	0.02mm	0.02mm
יציאות וכניסות לבקר	כניסות: • 16 דיגיטליות • 2 אנלוגיות יציאות • 16 דיגיטליות • 2 אנלוגיות	כניסות: • 16 דיגיטליות יציאות: • 16 דיגיטליות	כניסות • 16 דיגיטליות יציאות: • 16 דיגיטליות
שפת תכנות	URscript Api	C#	C#

Communication

- Communication with UR3E robotic arm is possible over multiple channels
- In the first phase (our project), we chose to implement communication on both primary and secondary ports with a slow frequency
- To effectively control the arm in a closed loop, future phases shall implement communication on the high frequency ports

e-Series							
	Primary		Secondary		Real-time		Real-time Data Exchange (RTDE)
Port no.	30001	30011	30002	30012	30003	30013	30004
Frequency [Hz]	10	10	10	10	500	500	500
Receive	URScript commands	-	URScript commands	-	URScript commands	-	Various data
Transmit	See attachment from the bottom		See attachment from the bottom		See attachment from the bottom		See RTDE Guide

Python Implementation

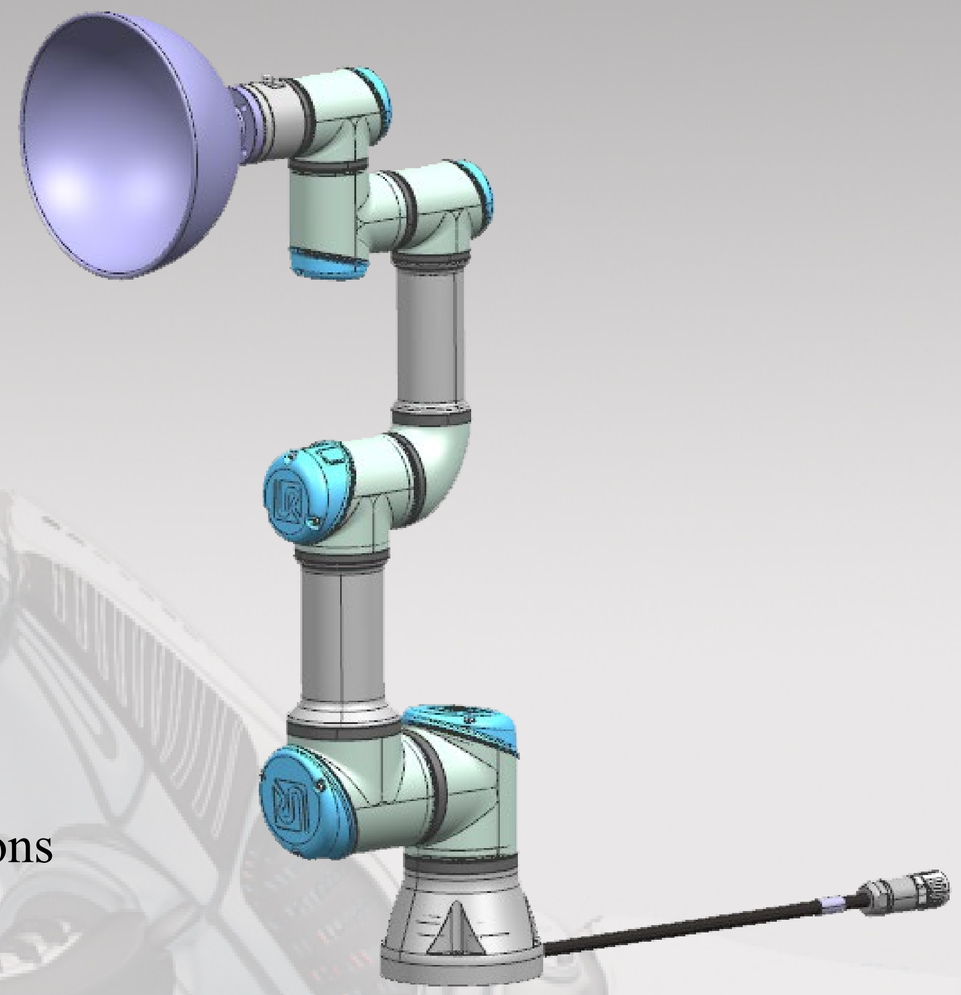
- Implementation is via OOP in Python language
- Classes simplify programming of the robot
- Point Class
  - Defines 6 DOF in Joint space
  - Enables user to easily create a new Point, modify/increase/decrease its values
  - Automatically checks validity of values before sending to the robotic arm
  - Enables user to work in both radians and degrees

Python Implementation

- Robot Class
  - Wraps the communication with the robotic arm in an easy-to-use package
  - User can change the payload of the arm and open/close connection manually with a single line of code.
  - The method get_point_from_freedrive enables the user to move the arm to a position he desires and create a Point object with the exact coordinates
  - Movej method moves the arm to a Point at a desired time or velocity

Results

- Up and running robotic arm system
- Throw and catch a ball - controlled by computer
- Receive feedback of joint position on secondary port
- Open-loop methodology ➔ motions took trial & error



Future

- Sensors will give feedback on ball location
- Closed loop control will allow for noise and non-ideal conditions

Demonstration and Questions

Thank you