RobotRaconteur®

PyRI

Robot Raconteur: an Interoperable Middleware for Robotics and PyRI Open Source Teach Pendant

John Wason, Ph.D.
Wason Technology, LLC, Tuxedo, NY

June 10, 2022

https://www.robotraconteur.com/
https://github.com/robotraconteur/robotraconteur
http://pyri.tech

Webinars

Robot Raconteur: an Interoperable Middleware
June 22nd 1 pm – 3 pm EDT

PyRI (Python Restricted Industrial) Open Source Teach Pendant
June 29th 1 pm – 3 pm EDT

Webinars will consist of one hour of presentations and questions followed by demonstrations

Contributors

Wason Technology, LLC
John Wason

Rensselaer Polytechnic Institute (RPI)
John Wen

Glenn Saunders

William Lawler

Honglu He

Burak Aksoy

Southwest Research (SwRI)
Levi Armstrong

Matt Robinson

General Electric
Pinghai Yang

Raytheon Technologies

Brigid Blakeslee

ARM Institute

Christopher Adams

Wason Technology

Rensselaer

SwRI

GE Global Research

Raytheon Technologies

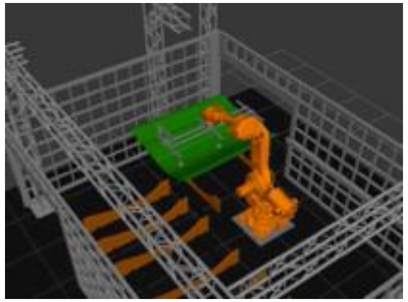
ARM

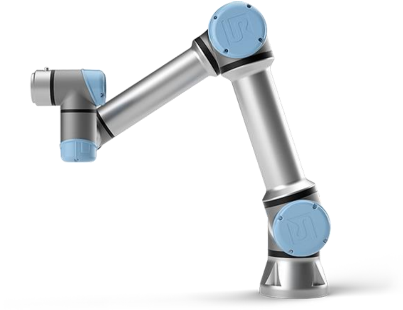
NEW YORK STATE OF OPPORTUNITY

Robot Raconteur Motivation

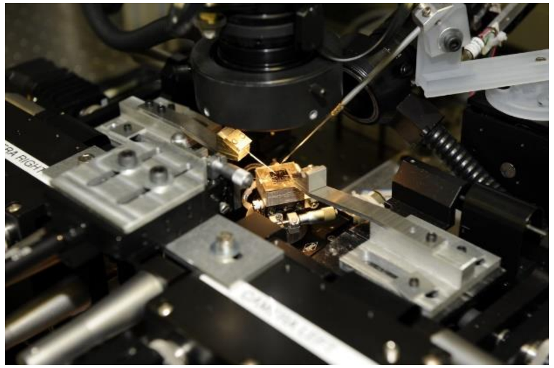
Rapid integration of robots, sensors, simulation packages, under various OS

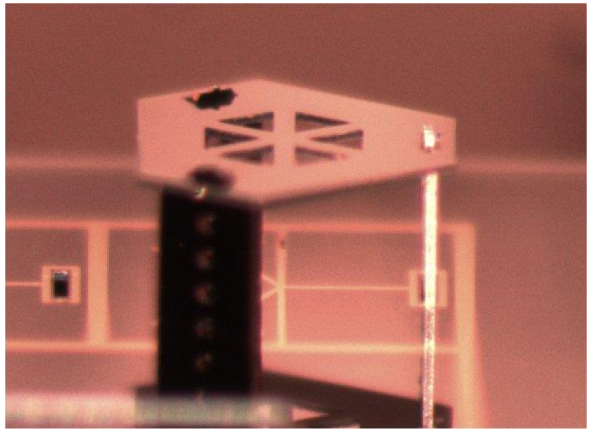
- ABB External Guided Motion (UDP/IP)
- ABB RobotStudio (Windows)
- Motoman High Speed Controller (PCI)
- Baxter and Sawyer (ROS)
- Universal Robotics (TCP/IP)
- Kinect Azure (Windows/Linux)
- ATI force/torque sensor (TCP/IP)
- Soft Robotics gripper (Digital I/O)
- Cognex Machine Vision (TCP/IP)

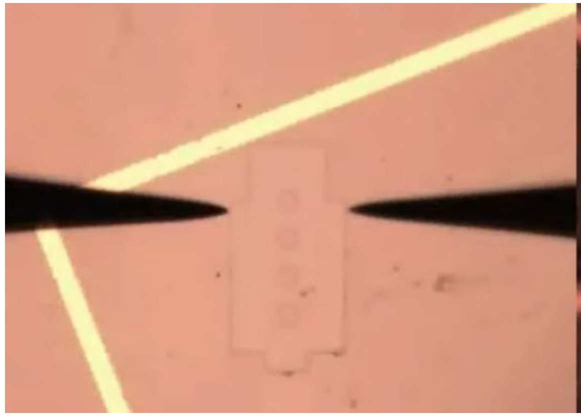


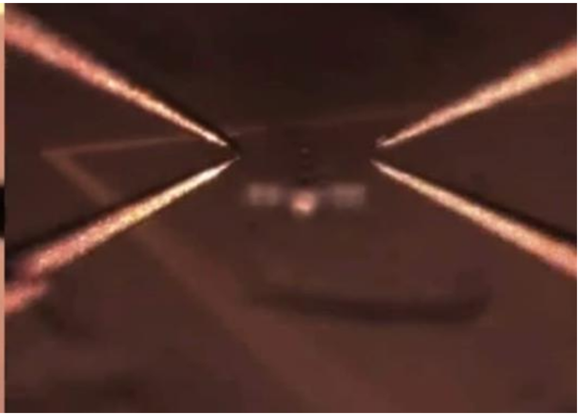


Motivation: Microassembly









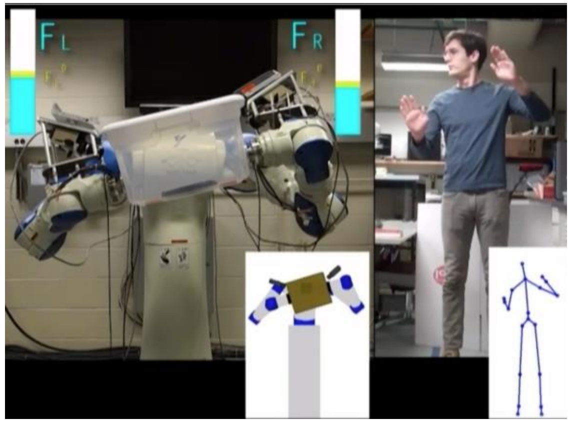
- System to manipulate microscale parts
  - 50 µm – 1000 µm
  - 25 µm thin
- 24 actuators
- 4 cameras
- 3 auxiliary actuators
- Force feedback joysticks
- 4 computers
- No existing middleware met requirements
- Primary motivation and first application of Robot Raconteur
- Project completed in 2011

Robot Raconteur: Overview

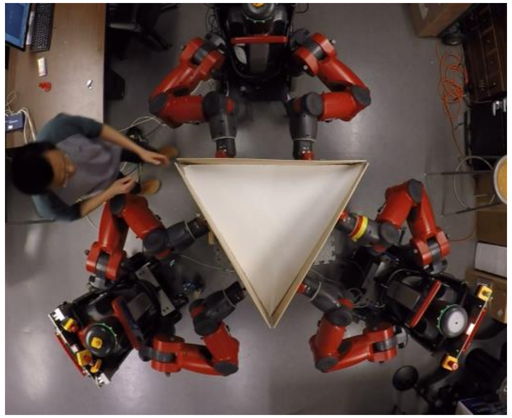
Language, platform, transport independent communication framework

- Compatibility: 22 platforms/architectures, 7 languages, 6 transport technologies
- Client-service model
- “Augmented Object-Oriented” model
  - Forward and backwards compatibility using polymorphism
- Plug and play capability
- Request-Response, streaming, and “most recent”
- TLS, certificates, and password security
  - Two central certificate authority chains, by Digicert and private HSM
  - Certificates available at nominal cost
- Compatible with Web and Cloud
- Node and service discovery
- Long-Term compatibility
- Open Source, Apache 2.0 License, first open source release Fall 2018
- Open Standards: https://github.com/robotraconteur/robotraconteur_standards
- Core library package “robotraconteur” available in ROS Noetic and ROS Humble
- Robot Raconteur ↔ ROS 2 Bridge: https://github.com/robotraconteur-contrib/robotraconteur_ros2_bridge

Examples

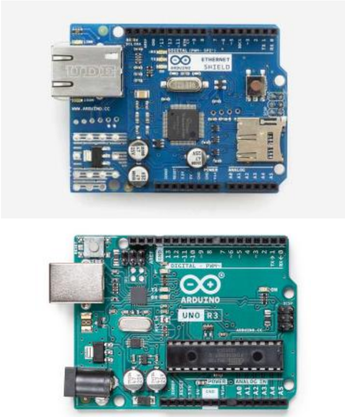


Human Guided Dual-Arm Manipulation



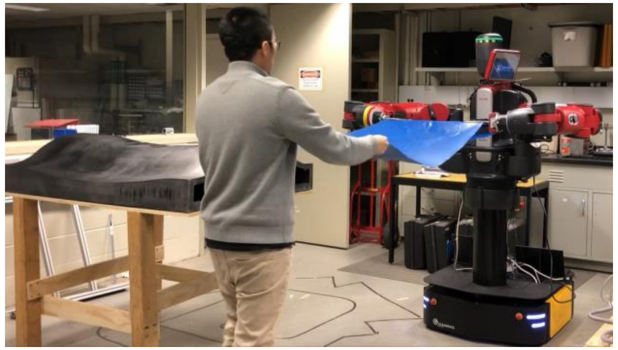
Cooperative Robotics

Smart Conference Room



Arduino Uno

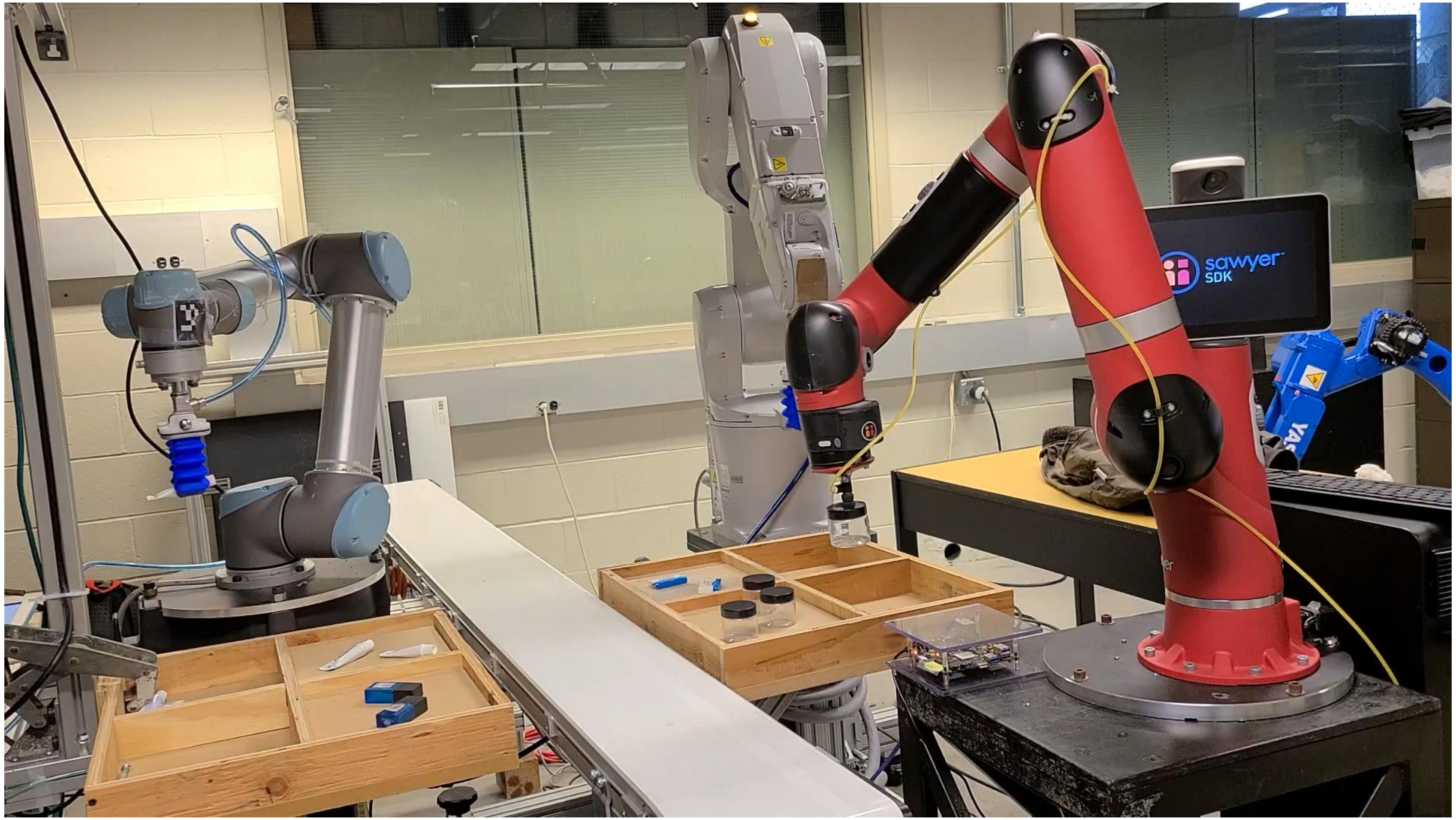
Assistive Robotics



Composite Layup

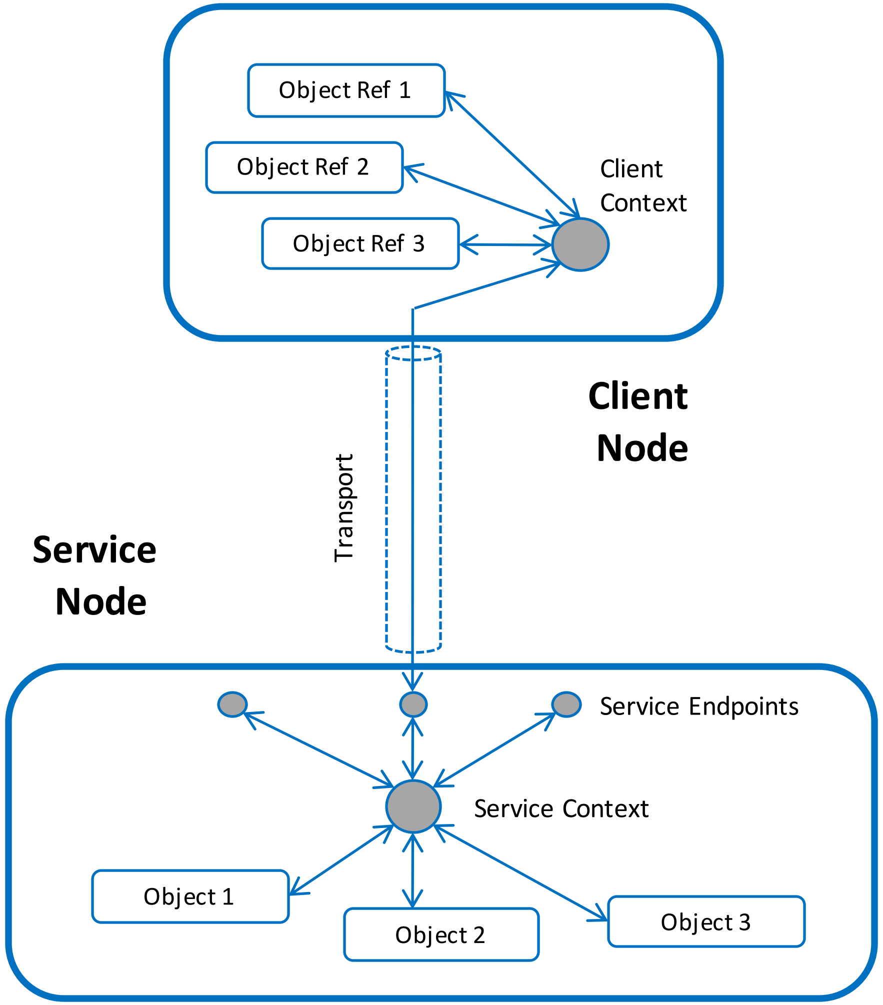
Stretch Cooperative Robots

Example: Multi-Robot Testbed



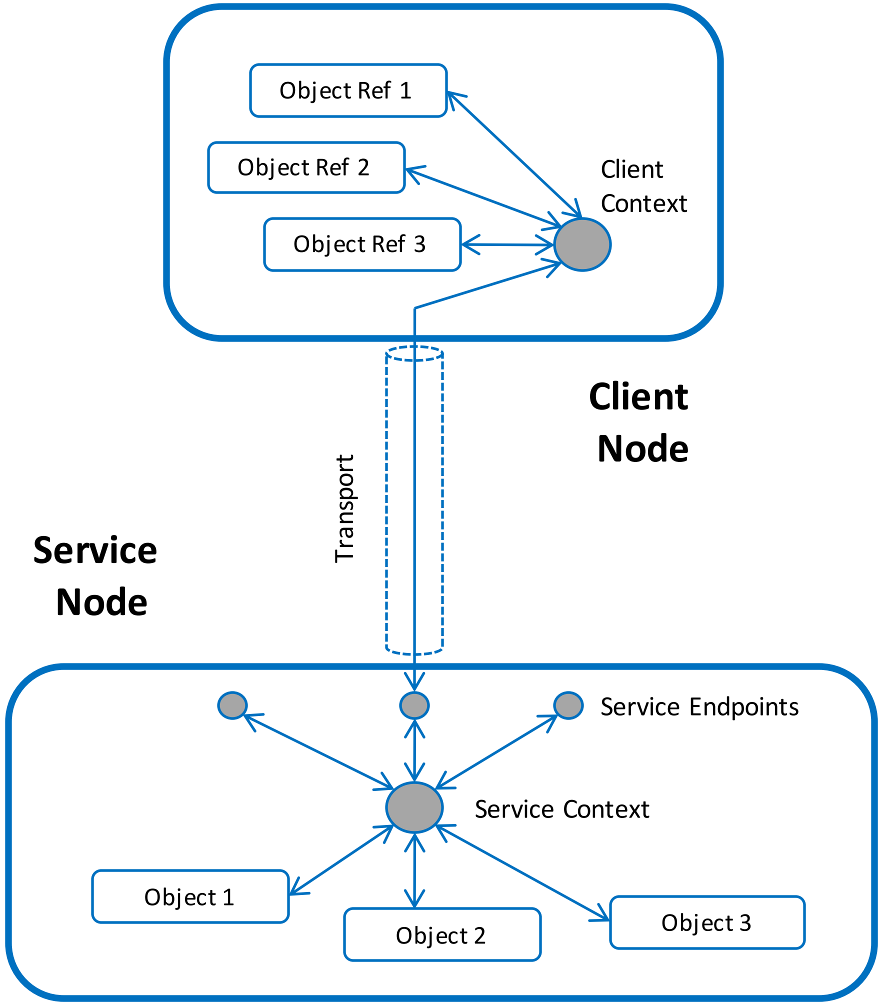
Full Video: https://youtu.be/3jhDXIRUiQY

Client-Service Model



- **Service**: Base object reference with members, and references to other objects.
- **Service Definition File**: Definition of object and structure members
- Support of **try/catch error** transmission across boundary (error in service transmitted to client, reversed for callback)

Client-Service Operation



Service Side

- Starts service node
- Service node
  - reads service definition file
  - listens on specified end point (port, usb, etc.)
  - provides the member request by client

Client Side

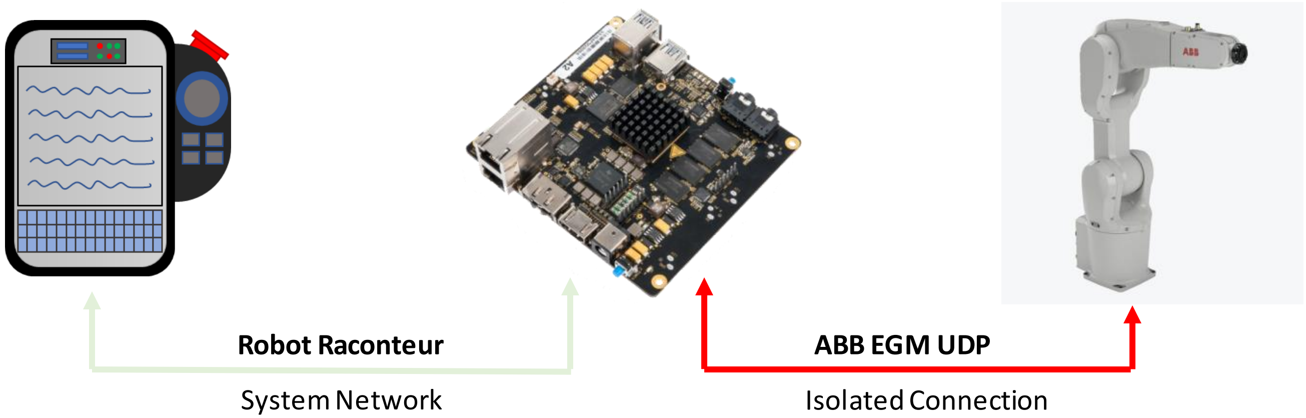
- Discovers service through node discovery or known URL of Service
- Connects to Service through the URL (with authentication: password or certificate)
- Reads Service Definition File from connected Service and sees the exposed functionality and data structure
- Requests the exposed members from Service.

Plug and play and Interoperability

- Dynamic type and proxy handling
  - Clients connecting to service receive “Service Definition”, and can dynamically handle objects and value types
  - Used for scripting languages like MATLAB and Python
- Interoperability through common or standardized “Service Definition”
  - Clients are designed to connect to specific defined types. If the service implements the expected types, the client can interact with the service
  - Standardization effort underway to develop reusable types
- Deployed systems will require standardized types for interoperability
  - Dynamic typing for scripting intended for laboratory and prototyping use

Standard Robot Type

- Standard robot type used for articulated robots
- Four command modes: Jog, Trajectory, Position, Velocity
- Robot Raconteur driver situated between robot and network
- Demo system: BeagleBoard x15 devices with dual ethernet ports and TI Sitara industrial processor running Robot Raconteur drivers



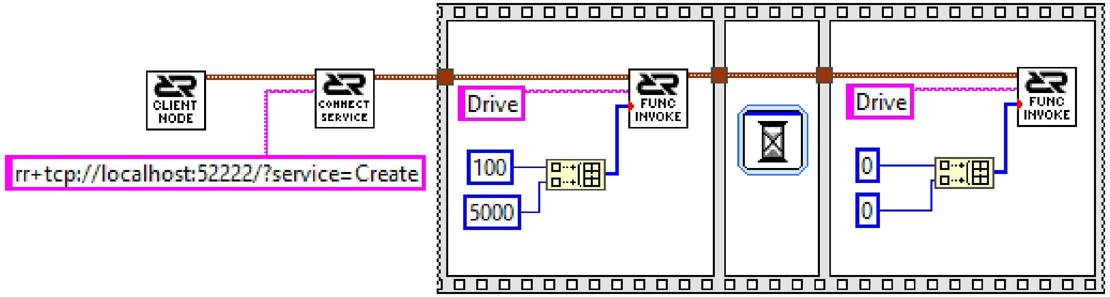
Example Clients

```
from RobotRaconteur.Client import *
import time

obj=RRN.ConnectService('rr+tcp://localhost:52222/?service=Create')

obj.Drive(100,5000)
time.sleep(1)
obj.Drive(0,0)
```

Python

```
o=RobotRaconteur.Connect('rr+tcp://localhost:52222/?service=Create');
o.Drive(int16(100),int16(5000));
pause(1);
o.Drive(int16(0),int16(0));
```

MATLAB



LabVIEW

Robot Raconteur Libraries

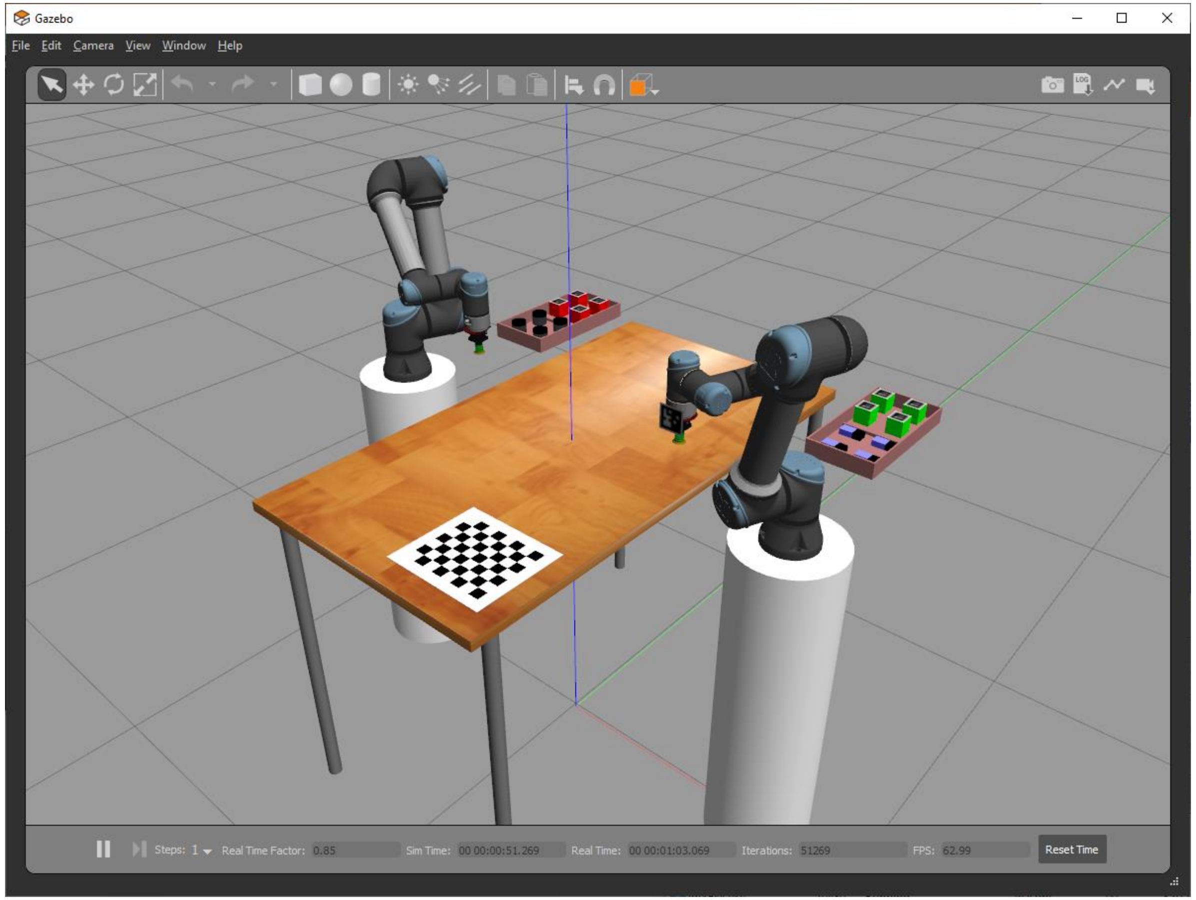
- Robot Raconteur libraries:
  - RobotRaconteur Core – standard library written in C++ using Boost ASIO, wrapped with multiple languages using SWIG
    - RobotRaconteur_Pyodide fork for running Python in WebAssembly
    - RobotRaconteur_WinXP fork for use on Windows XP
  - RobotRaconteurWeb – Pure C# implementation intended for use with Web Browsers, ASP.NET servers and on Xamarin mobile framework
    - JavaScript available using Bridge.NET C# to JavaScript compiler
  - RobotRaconteurLite – ANSI C99 minimalist implementation intended for real-time and embedded systems
    - Under development, currently supports message serialization

Standardized Service Definitions

- Standard Service Definitions are used to allow interoperability between devices
- Currently mostly using the “com.robotraconteur.robotics.robot.Robot” type
  - Provides for feedback on state of the robot
  - Allows for four command modes:
    - Jog
    - Trajectory
    - Velocity
    - Position
  - Note: Not all robots will support all command modes
- “Abstract Robot” base driver can be used with any robot that supports external command mode (ie ABB EGM, Sawyer ROS SDK, UR RTDE, etc)
  - Adding additional robot interfaces is relatively easy, typically a few hundred lines of code
- Total of 45 standard service definition files have been defined
- Group 1 frozen on April 5th, 2021
- https://github.com/robotraconteur/robotraconteur_standard_robdef

Robot Raconteur Training Simulator

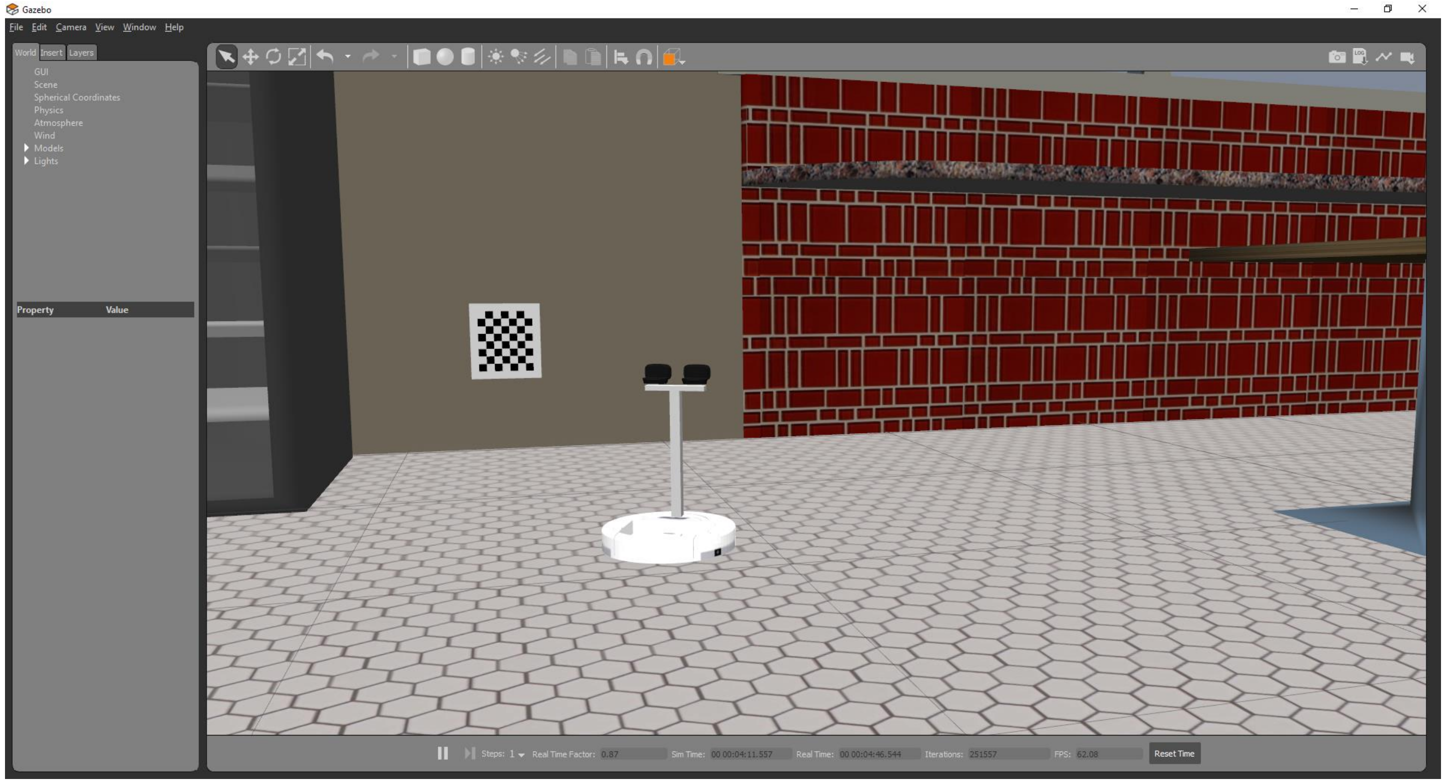
- Training simulator is available for Robot Raconteur based on Gazebo
- Installs easily using conda, works on Windows, Linux, Mac OS
- Includes:
  - Two UR 5e robots
  - Two simulated vacuum grippers
  - Simulated camera
  - Payloads
  - Calibration Target
  - Example Python Scripts



Steps: 1 Real Time Factor: 0.85 Sim Time: 00 00:00:51.269 Real Time: 00 00:01:03.069 Iterations: 51269 FPS: 62.99

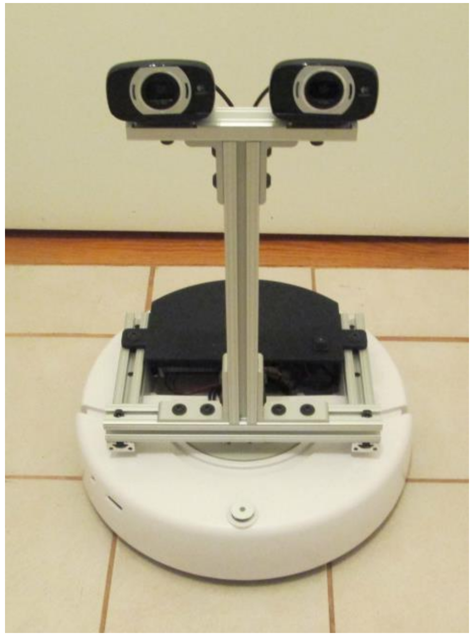
https://github.com/robotraconteur-contrib/robotraconteur_training_sim

iRobot Create Training Simulator Scene



Steps: 1 Real Time Factor: 0.87 Sim Time: 00 00:04:11.557 Real Time: 00 00:04:46.544 Iterations: 251557 FPS: 62.08

- iRobot Create with camera mast
- Designed to match example robot interface
- Used with Python examples



https://github.com/robotraconteur-contrib/robotraconteur_training_sim

https://github.com/robotraconteur/RobotRaconteur_Python_Examples

Open Source Teach Pendant Motivation

PyRI

- Open-source ecosystems including ROS and Orocos have advanced capabilities, but are difficult to program
  - Require extensive programming expertise, often with Linux and C++
  - Manufacturing organizations typically lack expertise to take advantage of these capabilities
- Robot vendors offer easier programming environment, but with limited capabilities
  - Typically use simplified text-based language or visual programming in some cases.
  - Programming environment and language is typically proprietary and non-interoperable

Project Objective:

Design a modern, easy-to-use open-source programming environment and teach pendant that can be used with equipment from any vendor

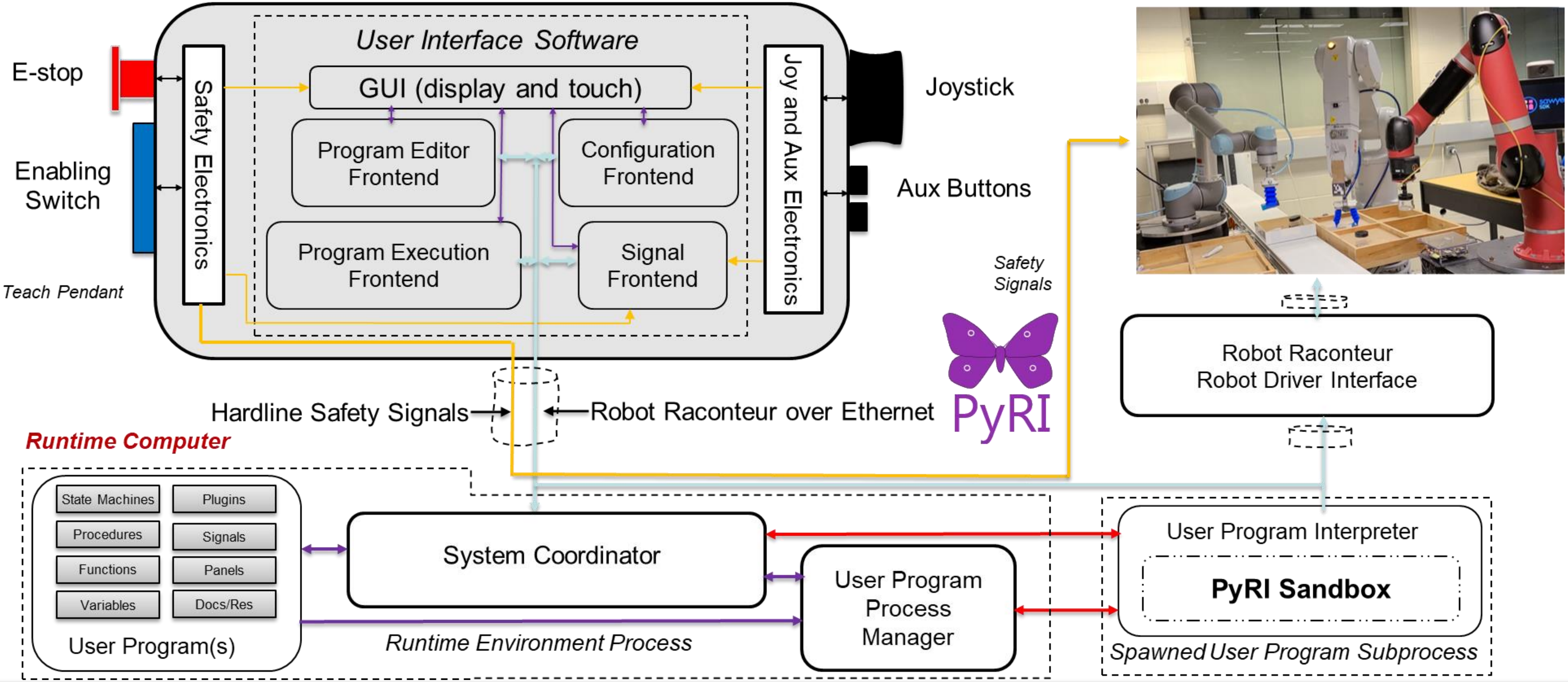
Open Source Teach Pendant Approach

- Robot Raconteur (RR) as communication middleware (open-source)
  - Build on ARM project F-18-01-F-19
  - Built-in interoperability, auto-discovery, multi-OS, encryption and authentication
  - Standard RR interface to ROS devices
- Vendor-Agnostic Robot Interface
  - Outer-loop robot motion command
  - Independent of vendor-specific robot programming language
- Simplified robot programming without extensive programming experience
  - Restricted Python dialect
  - Blockly visual programming

RobotRaconteur®

- Industrial run-time environment
  - Runtime environments for Python, Blockly
  - Manage hardware through plugins
- Touch screen user interface
  - Web browser-based implementation (Internet connection not needed)
  - Customizable GUI for equipment management and programming
- Prototype teach pendants hardware
  - Microsoft Surface Pro
  - Raspberry Pi

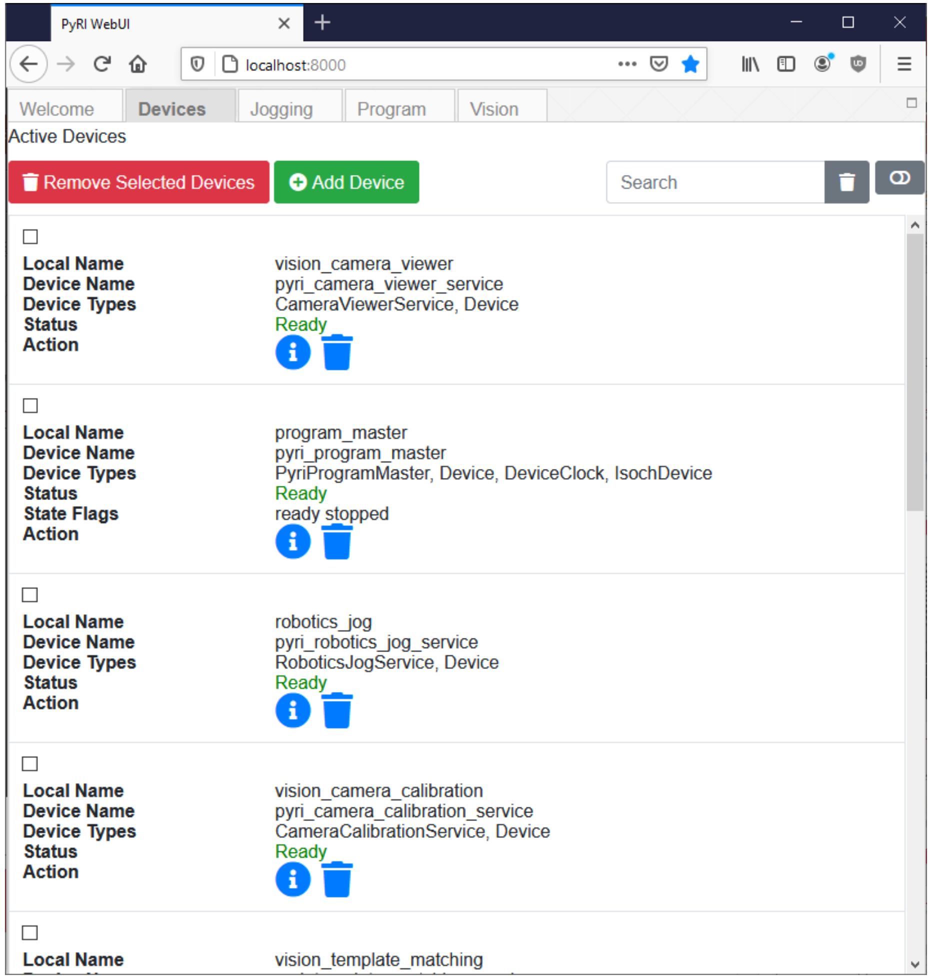
Teach Pendant Architecture



Pyodide

- Pyodide is a Python runtime environment for WebAssembly (https://github.com/pyodide/pyodide)
  - Allows a full Python Scientific stack to run within a web browser
  - Uses Emscripten to compile to Web Assembly
  - Originally started by Mozilla
  - Modified to be used with Robot Raconteur to allow for connection to Robot Raconteur services using Web Sockets
- Allows for the WebUI to be developed mostly in Python
  - Re-use of almost all Python code is possible
  - Unified code base between runtime and WebUI
- Vue.js is used along with Pyodide for developing interactive HTML elements (http://vuejs.org)

WebUI Panels (Devices)



Active Devices

☐

Local Name	vision_camera_viewer
Device Name	pyri_camera_viewer_service
Device Types	CameraViewerService, Device
Status	Ready
Action	

☐

Local Name	program_master
Device Name	pyri_program_master
Device Types	PyriProgramMaster, Device, DeviceClock, IsochDevice
Status	Ready
State Flags	ready stopped
Action	

☐

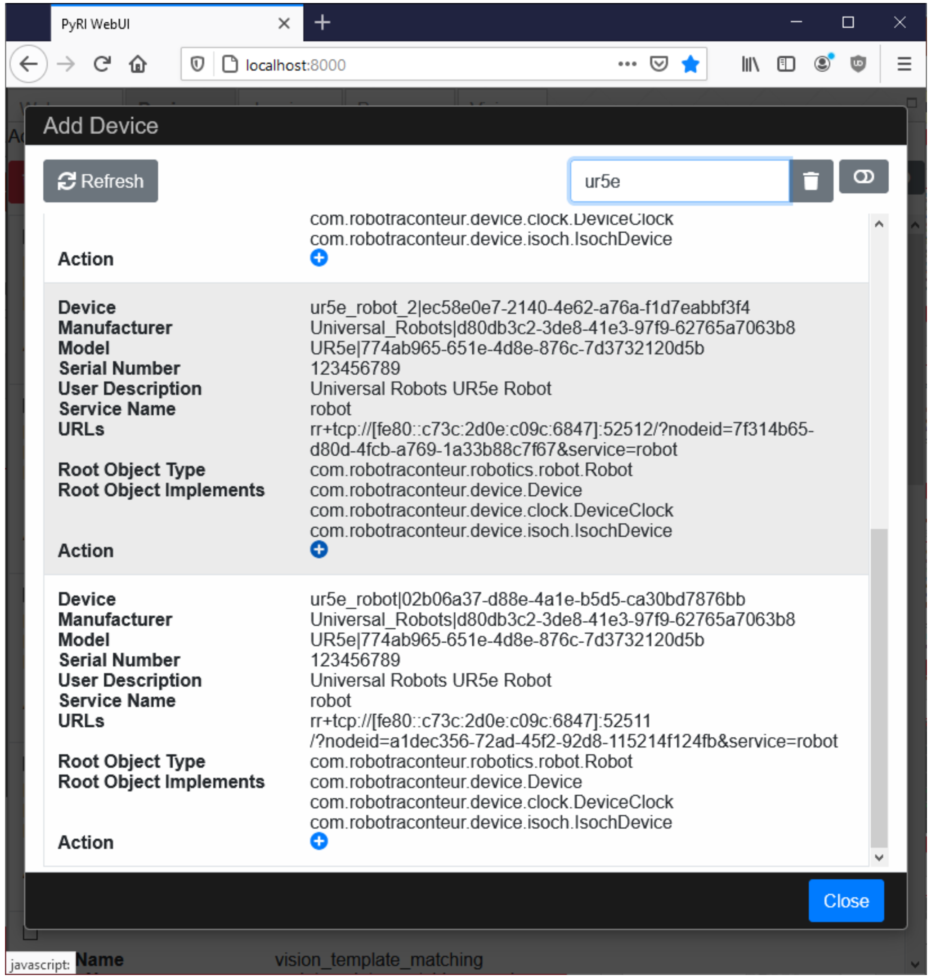
Local Name	robotics_jog
Device Name	pyri_robotics_jog_service
Device Types	RoboticsJogService, Device
Status	Ready
Action	

☐

Local Name	vision_camera_calibration
Device Name	pyri_camera_calibration_service
Device Types	CameraCalibrationService, Device
Status	Ready
Action	

☐

Local Name	vision_template_matching



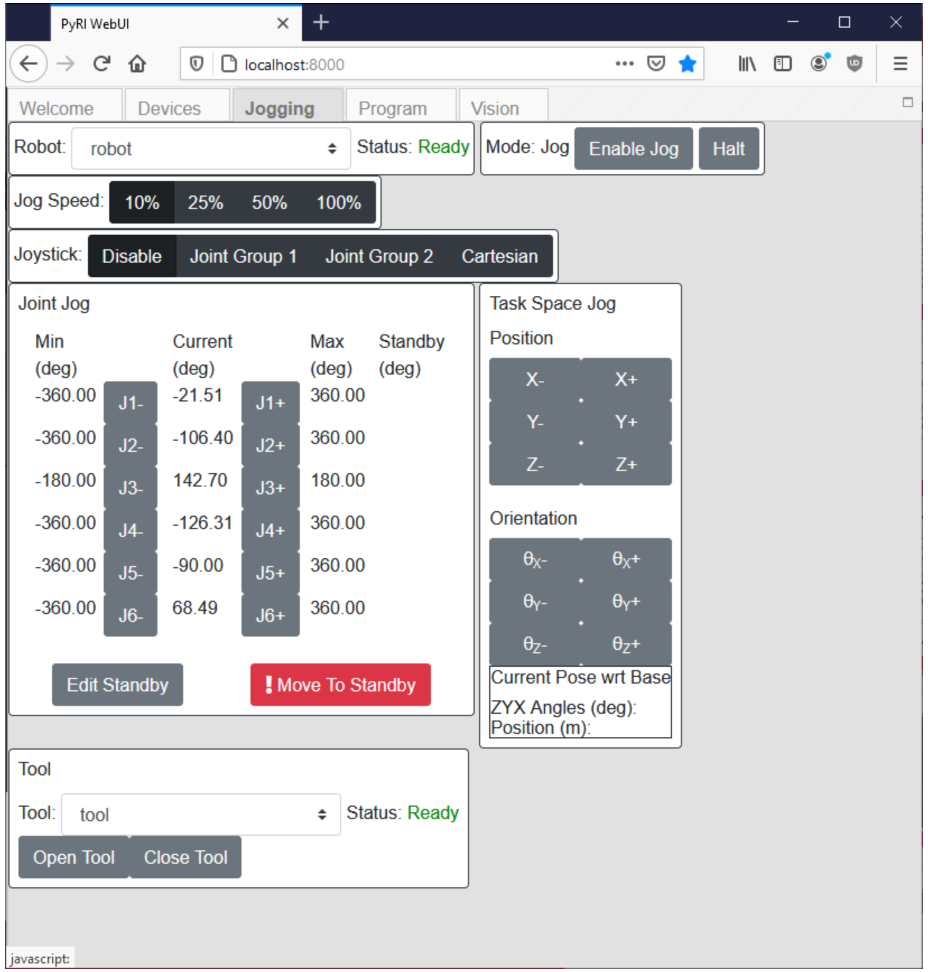
Add Device

	com.robotraconteur.device.clock.DeviceClock com.robotraconteur.device.isoch.IsochDevice
Action	

Device	ur5e_robot_2|ec58e0e7-2140-4e62-a76a-f1d7eabbf3f4
Manufacturer	Universal_Robots|d80db3c2-3de8-41e3-97f9-62765a7063b8
Model	UR5e|774ab965-651e-4d8e-876c-7d3732120d5b
Serial Number	123456789
User Description	Universal Robots UR5e Robot
Service Name	robot
URLs	rr+tcp://[fe80::c73c:2d0e:c09c:6847]:52512/?nodeid=7f314b65-d80d-4fcb-a769-1a33b88c7f67&service=robot
Root Object Type	com.robotraconteur.robotics.robot.Robot
Root Object Implements	com.robotraconteur.device.Device com.robotraconteur.device.clock.DeviceClock com.robotraconteur.device.isoch.IsochDevice
Action	

Device	ur5e_robot|02b06a37-d88e-4a1e-b5d5-ca30bd7876bb
Manufacturer	Universal_Robots|d80db3c2-3de8-41e3-97f9-62765a7063b8
Model	UR5e|774ab965-651e-4d8e-876c-7d3732120d5b
Serial Number	123456789
User Description	Universal Robots UR5e Robot
Service Name	robot
URLs	rr+tcp://[fe80::c73c:2d0e:c09c:6847]:52511 /?nodeid=a1dec356-72ad-45f2-92d8-115214f124fb&service=robot
Root Object Type	com.robotraconteur.robotics.robot.Robot
Root Object Implements	com.robotraconteur.device.Device com.robotraconteur.device.clock.DeviceClock com.robotraconteur.device.isoch.IsochDevice
Action	

WebUI Panels (Jog)



Robot: robot Status: Ready Mode: Jog

Jog Speed: 10% 25% 50% 100%

Joystick: Disable Joint Group 1 Joint Group 2 Cartesian

Joint Jog

Min (deg)	Current (deg)	Max (deg)	Standby (deg)
-360.00	-21.51	360.00	
-360.00	-106.40	360.00	
-180.00	142.70	180.00	
-360.00	-126.31	360.00	
-360.00	-90.00	360.00	
-360.00	68.49	360.00	

Task Space Jog

Position

Orientation

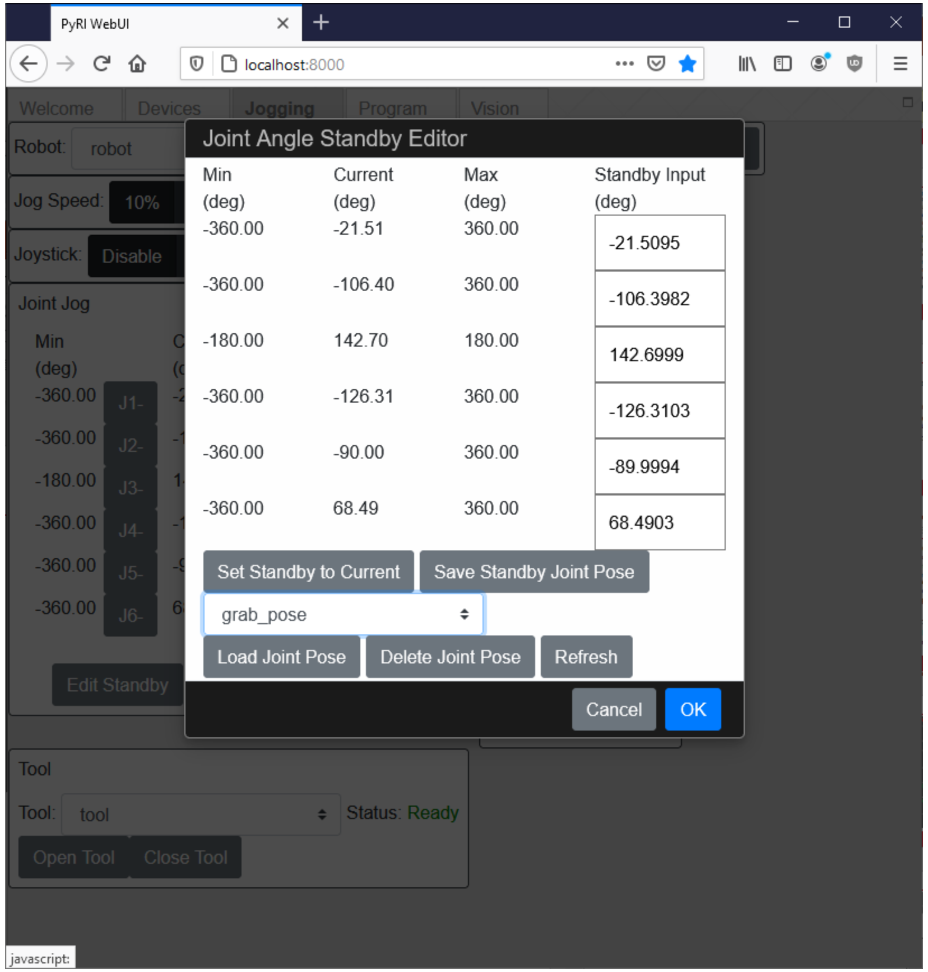
Current Pose wrt Base

ZYX Angles (deg):

Position (m):

Tool

Tool: tool Status: Ready

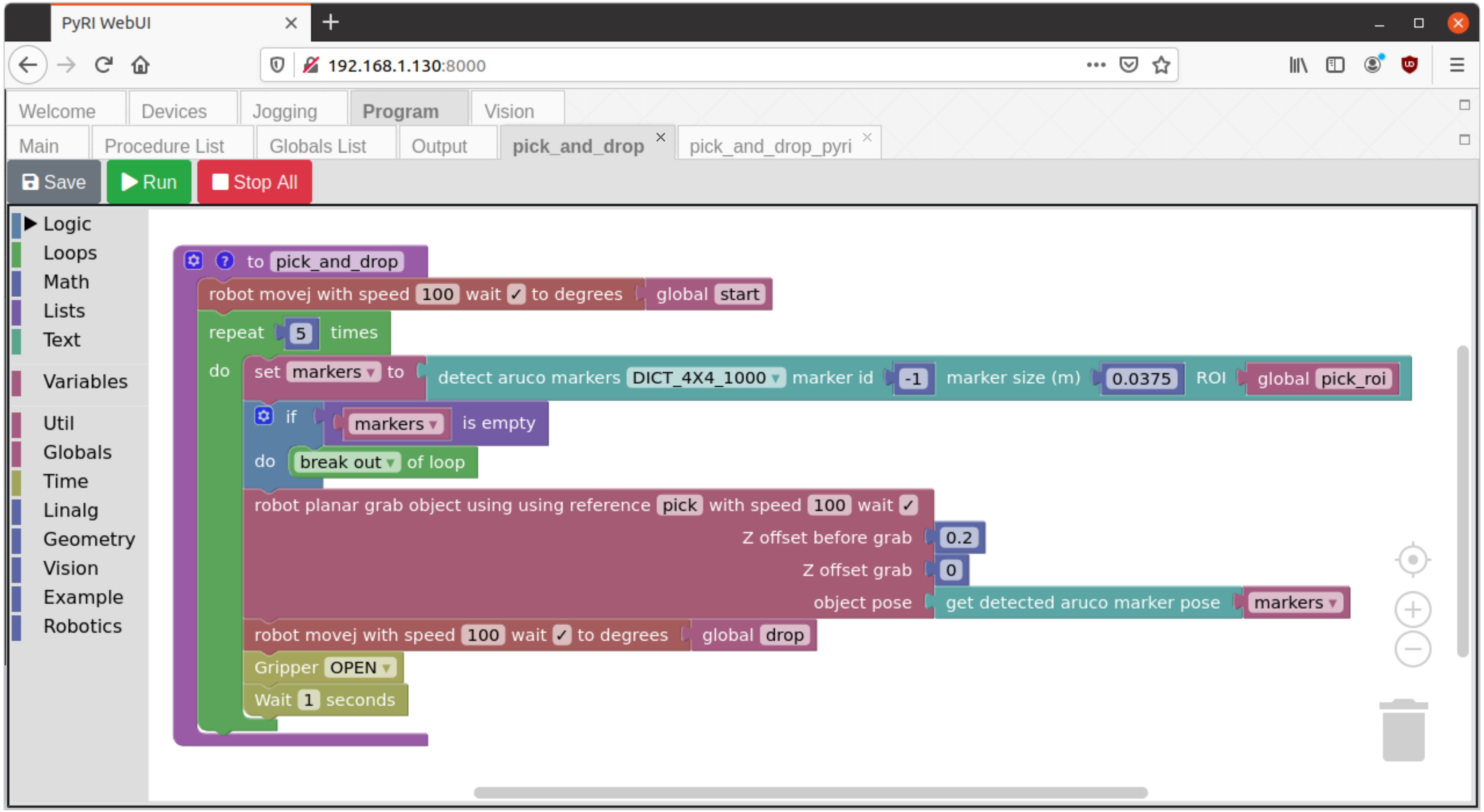


Joint Angle Standby Editor

Min (deg)	Current (deg)	Max (deg)	Standby Input (deg)
-360.00	-21.51	360.00	-21.5095
-360.00	-106.40	360.00	-106.3982
-180.00	142.70	180.00	142.6999
-360.00	-126.31	360.00	-126.3103
-360.00	-90.00	360.00	-89.9994
-360.00	68.49	360.00	68.4903

grab_pose

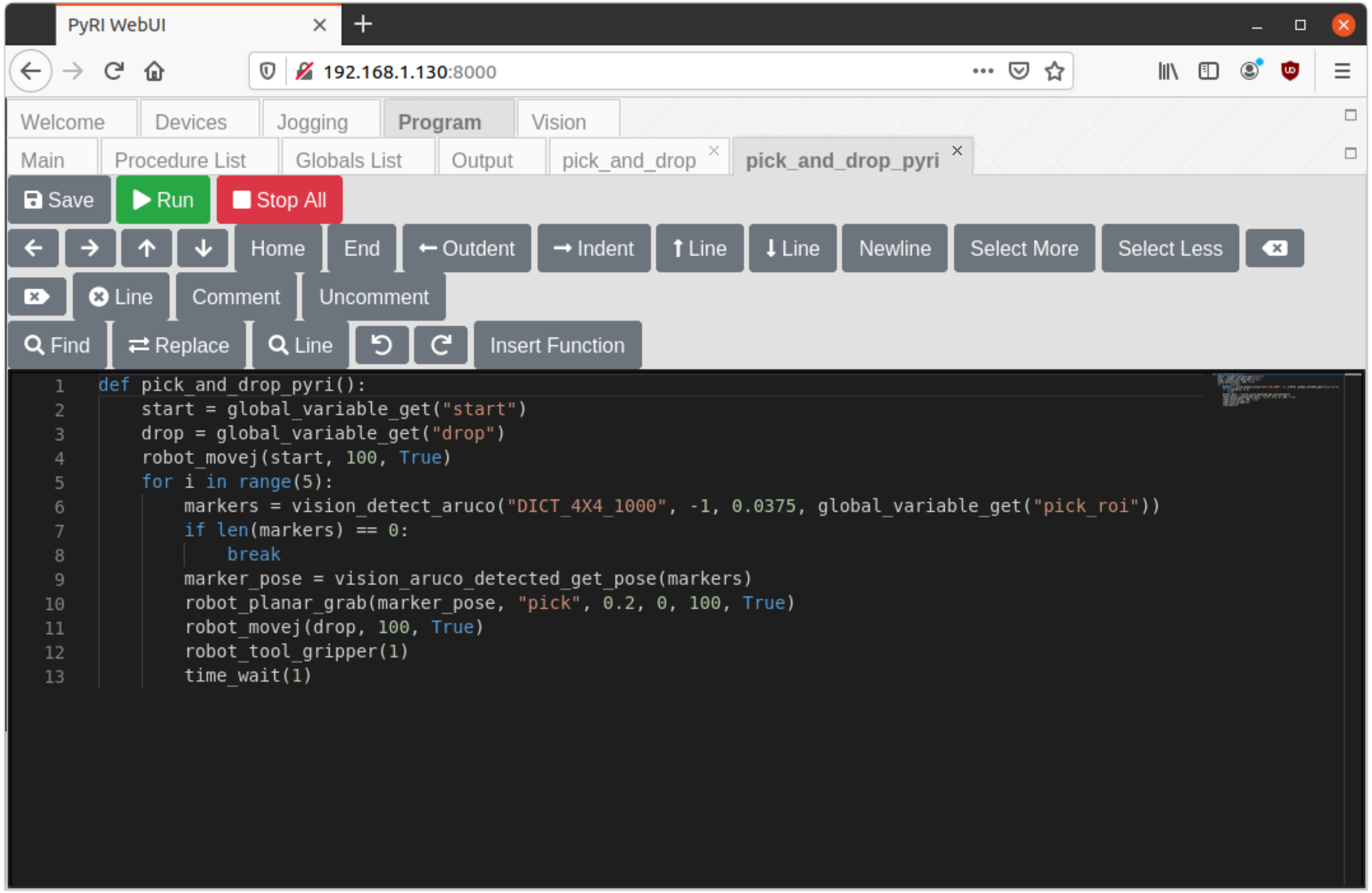
WebUI Panels (Program, Blockly)



```
to pick_and_drop
  robot movej with speed 100 wait ✓ to degrees global start
  repeat 5 times
  do set markers to detect aruco markers DICT_4X4_1000 marker id -1 marker size (m) 0.0375 ROI global pick_roi
     if markers is empty
     do break out of loop
     robot planar grab object using using reference pick with speed 100 wait ✓
                                                   Z offset before grab 0.2
                                                           Z offset grab 0
                                 object pose get detected aruco marker pose markers
     robot movej with speed 100 wait ✓ to degrees global drop
     Gripper OPEN
     Wait 1 seconds
```

- Google Blockly visual programming
- Simple example program to pick up marked cubes using vision, and drop into bin
- Executed in sandbox after compilation to Python
- Calls “sandbox functions” to interact with system

WebUI Panels (Program, PyRI)



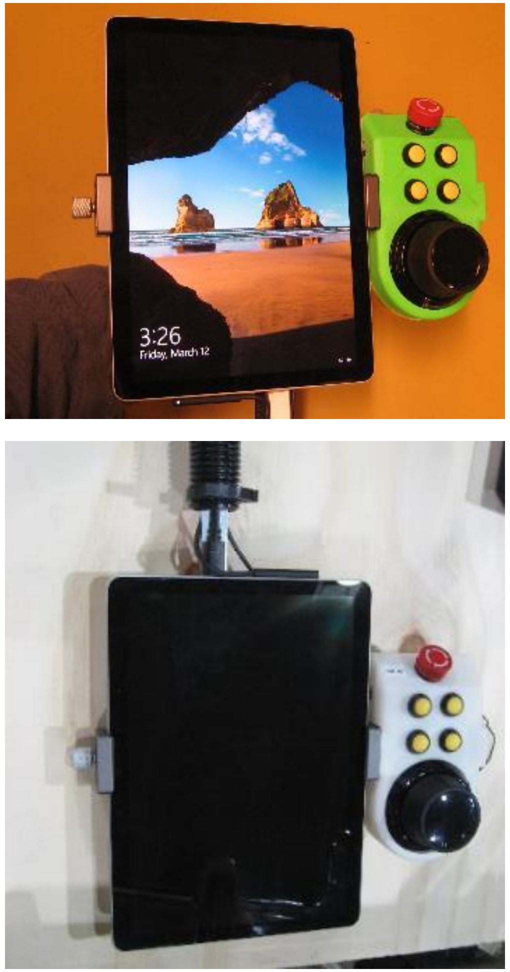
```python
def pick_and_drop_pyri():
    start = global_variable_get("start")
    drop = global_variable_get("drop")
    robot_movej(start, 100, True)
    for i in range(5):
        markers = vision_detect_aruco("DICT_4X4_1000", -1, 0.0375, global_variable_get("pick_roi"))
        if len(markers) == 0:
            break
        marker_pose = vision_aruco_detected_get_pose(markers)
        robot_planar_grab(marker_pose, "pick", 0.2, 0, 100, True)
        robot_movej(drop, 100, True)
        robot_tool_gripper(1)
        time_wait(1)
```

- Python Restricted Dialect (PyRI) directly written as code
- Simple example program to pick up marked cubes using vision, and drop into bin
- Executed in sandbox
  - Essentially the same as Blockly, but skips visual layer
- Calls “sandbox functions” to interact with system
- WebUI uses “Monaco Editor”
  - Same editor as Visual Studio Code IDE
- Extra softkeys to help when used on touchscreen

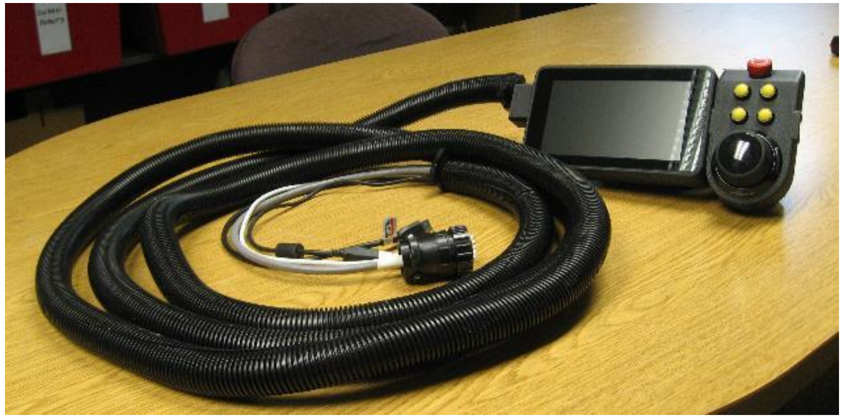
Teach Pendant and Runtime Computer Hardware

Completed:

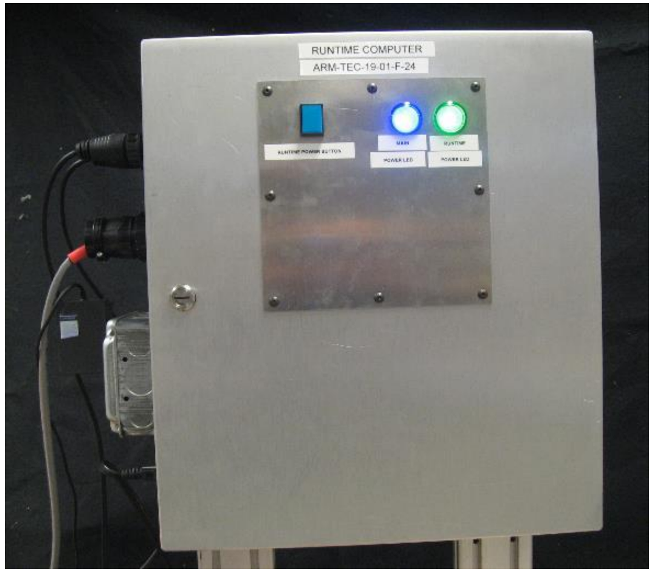
- Two Microsoft Surface version teach pendant prototypes
- One Raspberry Pi version teach pendant prototype
- Weight: ~2lb, R-Pi version is ~10% lighter
- One Runtime Computer prototype



Microsoft Surface Versions

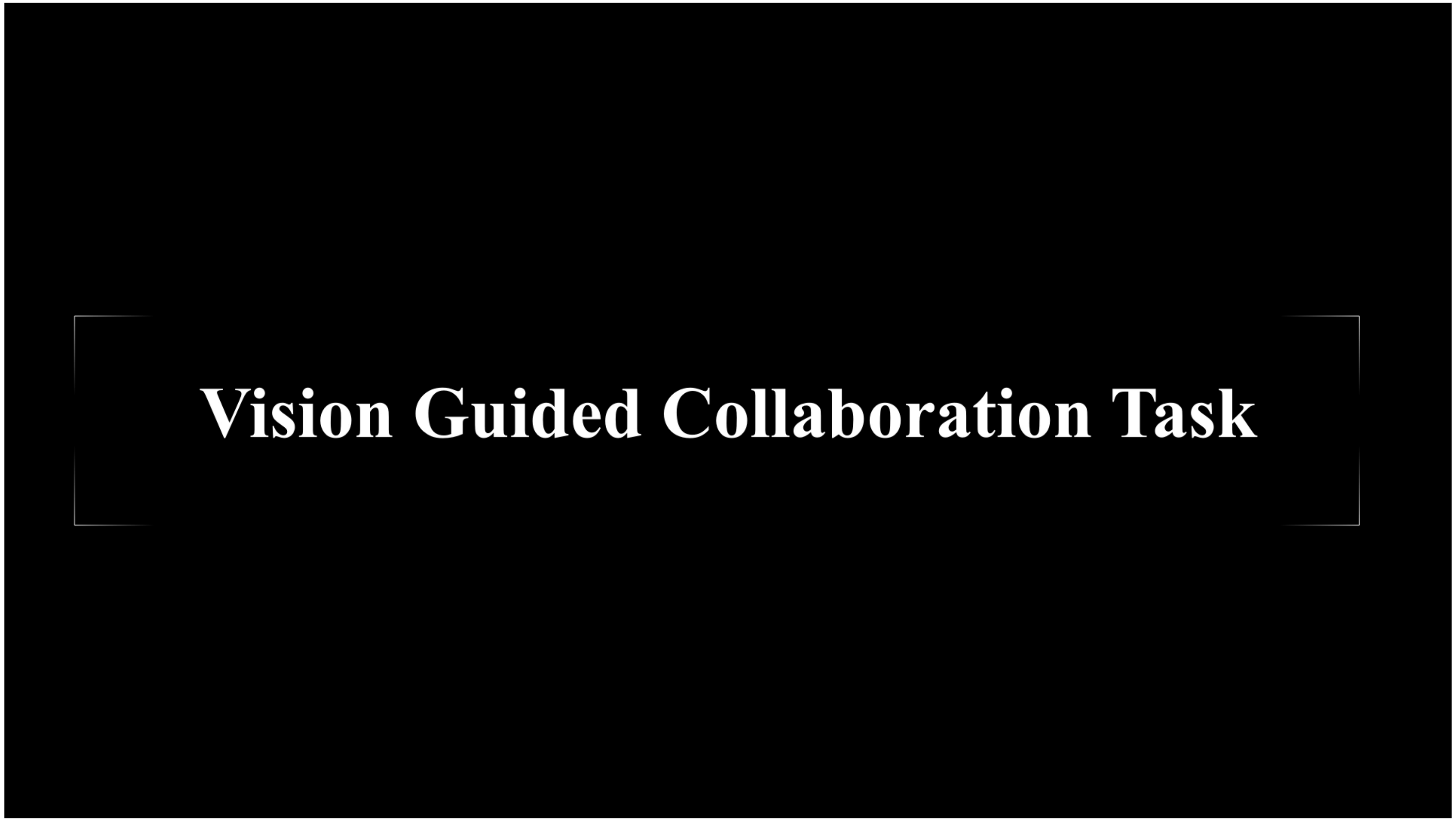


Raspberry Pi Version



Runtime Computer

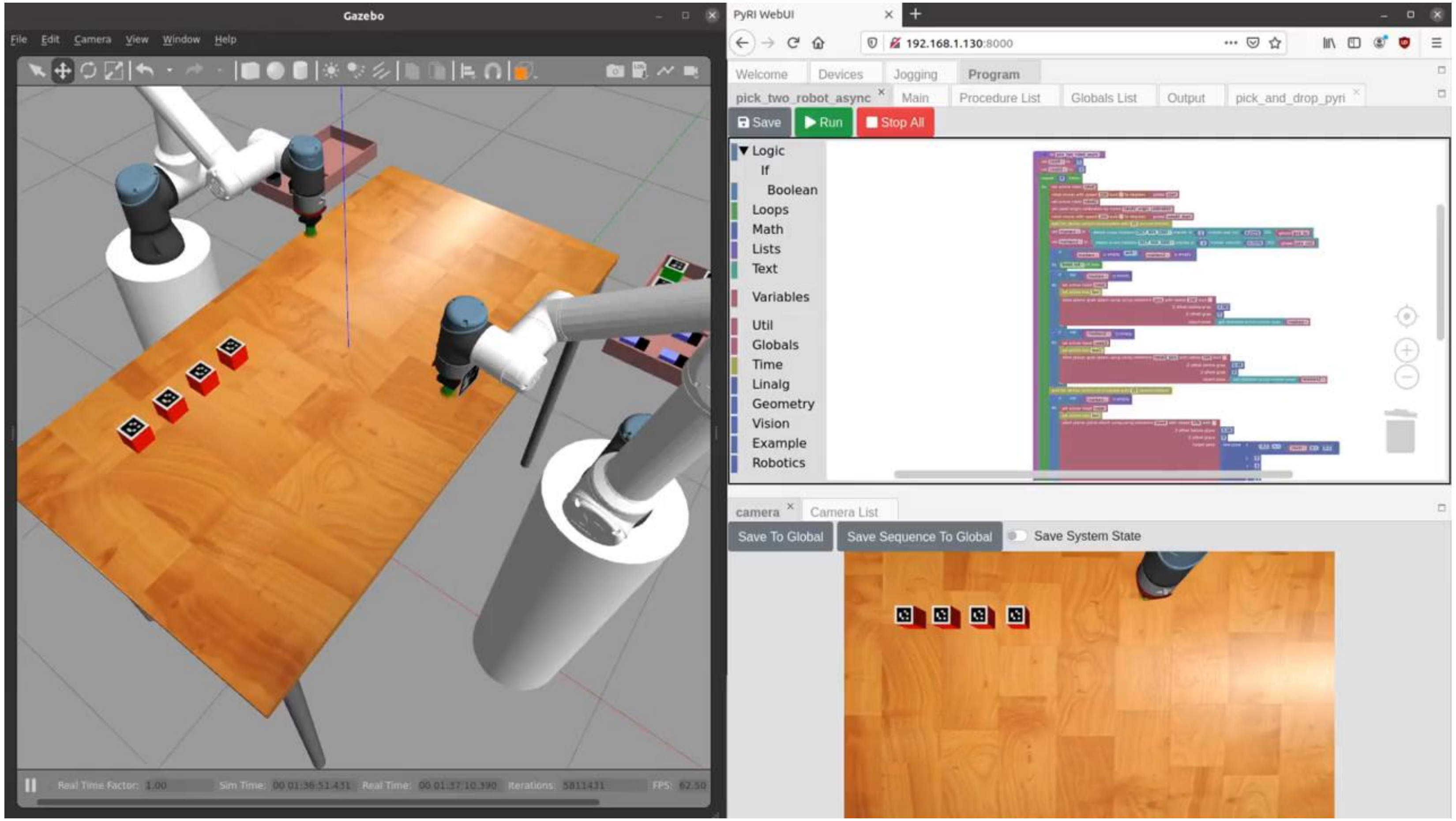
Vision Guided Collaboration Tasks



Vision Guided Collaboration Task

Full Videos: https://youtu.be/9KSYgGpG8mk https://youtu.be/jF_BGaFI7Qc

Simulated Pick and Place



Tesseract Planning

- Tesseract used for kinematics, path planning, and visualization
- Currently implemented in separate PyRI module
  - Will merge into primary Robotics modules
- Tesseract Python wrappers utilized
  - https://github.com/tesseract-robotics/tesseract_python
  - Developed by Wason Technology
- Available on PyPi and Conda for easy installation (Windows and Linux)
  - PyPi: https://pypi.org/project/tesseract-robotics/
    - Self contained wheel
  - Conda: https://anaconda.org/Tesseract-Robotics/tesseract-robotics-superpack

TESSERACT

Try it now!

Robot Raconteur Training Sim and PyRI Open Source Teach pendant are both available as conda packages (Windows and Linux)

Install using following command in Anaconda or Miniconda (one line):

```
mamba create -n pyri -c conda-forge -c robotraconteur -c pyri-project robotraconteur_training_sim pyri-robotics-superpack
```

Run using:

```
conda activate pyri

run_2ur5e_sim

pyri-core --db-file=my_project.db
```

Open Firefox and go to http://localhost:8000

Thank You

John Wason
Wason Technology LLC, Tuxedo, NY

wason@wasontech.com

(518) 279-6234

https://www.robotraconteur.com/
https://github.com/robotraconteur/robotraconteur
http://pyri.tech
https://github.com/pyri-project
https://github.com/robotraconteur/robotraconteur-directory
http://wasontech.com

This work was supported in part by Subaward No. ARM-17-QS-F-01, ARM-TEC-18-01-F-19, ARM-TEC-19-01-F-24, and ARM-TEC-21-02-F-19 from the Advanced Robotics for Manufacturing ("ARM") Institute under Agreement Number W911NF-17-3-0004 sponsored by the Office of the Secretary of Defense. ARM Project Management was provided by Christopher Adams. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of either ARM or the Office of the Secretary of Defense of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

This work was supported in part by the New York State Empire State Development Division of Science, Technology and Innovation (NYSTAR) under contract C160142.

ARM

NEW YORK
STATE OF
OPPORTUNITY