

DeepRacer Demo Procedure

LAST UPDATED May 11, 2025

Jump to Section:

1. [Quick Reference](#)
2. [Links](#)
3. [DeepRacer Demo - Procedure](#)
 - a. [Pre-Procedure Notes](#)
 - b. [Main Procedure](#)
 - c. [Other Procedures](#)
4. [DeepRacer Demo - Troubleshoot](#)
 - a. [Basic Issues](#)
 - b. [Ventuz - Multi-Projector Setup](#)
 - c. [Controller Graphics Card - HP Z8 G4 Workstation](#)
5. [Closedloop_rt.py - Troubleshoot Summary & Next Steps](#)
 - a. [Issues with the original closedloop_rt.py](#)
 - b. [Improved Controllers](#)
 - c. [Latest Development: Server Connection Issue](#)
 - d. [Tests designed to diagnose OptiTrack server connection timeout](#)
 - e. [Next Steps](#)

Quick Reference:

Media Server:

192.168.1.194:12345/OptiTrackRestServer

pFaces Server:

192.168.1.144:12345/pFaces/REST/dictionary/DeepRacer1

pFaces:

D:/Workspace/pFaces-SymbolicControl/ex_gb_fp/deepracer_rt/run_d_1_2_fast.bat

Links:

Google doc for this file:

[DeepRacer Demo Procedure](#)

Other Github Resources

<https://github.com/HyConSys/CUBLab>

<https://github.com/HyConSys/deepracer-utils>

DeepRacer Demo - Procedure

Pre-Procedure Notes

1. DeepRacer Power-Up

· Charge both robot batteries: one for the drivetrain and one for the compute board. If battery is low, the robot may ignore compute server commands.

2. Compute Server IP Configuration

On the compute server PC, run the command below to find the IPv4 address:

`ipconfig`

Do the same for the media server PC.

Open the following Python files on the deepracer:

- deepracer-utils/examples/sym_control/closedloop_rt.py
- deepracer-utils/examples/sym_control/closedloop_online.py

Make sure the IP addresses are current (example):

`LOCALIZATION_SERVER_IPPORT = "192.168.1.194:12345"`

`COMPUTE_SERVER_IPPORT = "192.168.1.147:12345"`

3. SSH into Robot

Use this to connect:

`ssh deepracer@192.168.1.70`

Password: `deepracer1234`

If SSH times out, retry multiple times until it connects.

4. Before Running the Controller `closedloop_rt`

Place one obstacle and one target on the grid. The system won't respond if these are missing.

5. Log Files

Each time `closedloop_rt.py` runs, a log file is saved in:

`deepracer-utils/examples/sym_control/`

Format: `%d%m%Y%H%M` (e.g., `110520251810` = May 11, 2025 18:10)

6. Troubleshooting

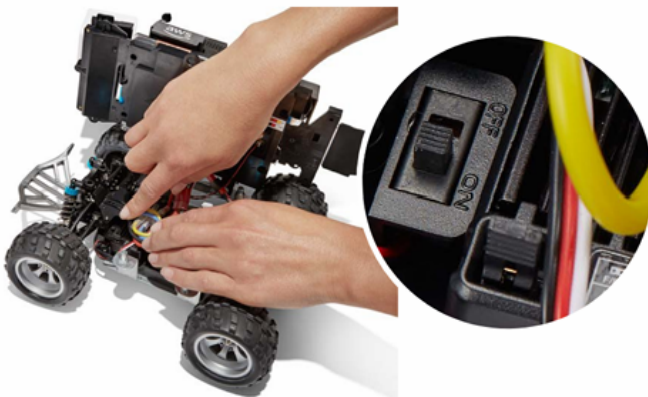
For more help, see: <https://github.com/HyConSys>

Main Procedure

1. Projectors and DeepRacer:

(Make sure not to hit the reset button (the one next to the power button) on the robot)

- Connect the drivetrain and compute board batteries to the deepracer. The drivetrain battery connects and charges via the three red wires, while the compute board batteries connect through the USB.
- Turn the deepracer on (look for a small switch on the chassis, see image below) and place the robot on the grid.



- Turn on all four projectors and the deepracer. Use the remote and point at projectors and hit on
 - The deepracer power button may take pressing 3 times to work: it blinks white then red the first two times, then lights a stable white on the third time.

The cameras will be on after initializing the environment

2. Media Server PC: GUI

a. In a terminal app:

```
cd D:/Workspace/ArenaManager  
python AutoDeploy.py
```

This script deploys the GUI to initialize the environment (Ventuz, Motive, and localization server) and place obstacles and targets on the grid. Motive tracks marker motion via OptiTrack cameras. Ventuz blends the projected image on the mat.

b. Click "Initialize Environment" and wait for the grid to appear on the mat.

3. Controller

On compute server, run the file (shortcut on desktop):

```
D:/Workspace/pFaces-SymbolicControl/ex_gb_fp/deepracer_rt/run_d  
_1_2_fast.bat
```

This sets up compute server to receive and process data from the robot. The

server synthesizes commands and sends {steer, throttle} between -1 and 1. Negative values are for steering left, positive values are for steering right. Negative throttle puts it in reverse.

DeepRacer 1 (make sure target is on mat)

1. In a new terminal:

`ssh deepracer@192.168.1.70`

Password: `deepracer1234`

2. **`source /opt/aws/deep racer/setup.sh`**

This sets environmental variables and sources environmental hooks such as python and library paths

3. **`python ~/deep racer-utils/put_best_cal.py`**

This calibrates for steering and speed via rospy and returns calibration status.

4. **`cd deep racer-utils/examples/sym_control`**

Navigates to controller folder.

5. **`python closedloop_rt.py`**

This is the controller code. It uses RemoteSymbolicControl to send data to compute server and synthesizes a control loop using DeepRacerController.

Other Procedures

DeepRacer Obstacle:

14. Run new terminal

15. ssh root@192.168.1.110

16. password: deepracer1234

17. source /opt/ros/foxy/setup.sh

- Setting up environment paths and customizing runtime environment

18.

source/root/deepracer_ws/aws-deepracer-servo-pkg/install/setup.sh

- This script extends the environment with the environment of other prefix paths which were sourced when this file was generated as well as all packages contained in the prefix path

19. cd deepracer-utils/tools

20. python3 ManualControlServer.py Initiates deepracer obstacle into manual drive mode by utilizing the MotionControls class. By using the python library HTTPServer this script becomes a simple server where manual drive controls are sent to the car.

21. Run new terminal

22. python C:\Users\CUBLab\Desktop\KeyboardControl.py

- Allows the keyboard to control the car. Matches each keyboard action with the necessary action in the car and runs it through ManualControlServer which tells the car what to do.

DeepRacer1 Procedure with Auto Button:

1. Open new terminal
2. ssh `deepracer@192.168.1.70` (retry if it fails)
3. password: `deepracer1234`
4. Once connected, return to Arena Manager and place targets/obstacles
5. Enable logging if needed (check log), disable before closing apps
6. Hit "connect and go to target" to start
7. After run, hit "transfer and delete files" to save to:
C:\Users\CUBLab\Desktop\deepracer-logs

DeepRacer Demo - Troubleshoot

Basic Issues

- If you connect the batteries but there is no beeping when turning the drivetrain on, the battery may be locked out. Take the jumper cable and connect the output and the input connectors of the battery as shown in the figure below.

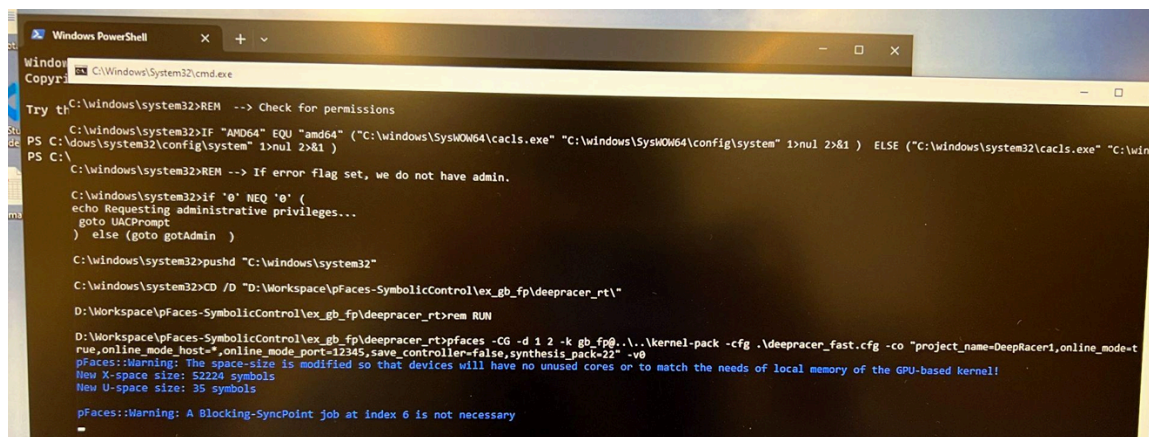


- If (STEP 2b) clicking "Initialize Environment" does not display the grid on the mat, some programs may have failed to boot.

Exit everything and repeat the setup. If it still fails after several attempts, reboot the computer.

- If (STEP 3) running the compute server throws a licensing issue, email Mahmoud for a new license and replace it in every folder.

A blue warning message is normal and can be ignored.



```

Windows PowerShell
C:\Windows\System32\cmd.exe

Try to: C:\Windows\system32>REM --> Check for permissions

PS C:\Windows\system32>IF "AMD64" EQU "amd64" ("C:\Windows\System64\cacls.exe" "C:\Windows\System64\config\system" 1>nul 2>&1 ) ELSE ("C:\Windows\system32\cacls.exe" "C:\win
PS C:\
C:\Windows\system32>REM --> If error flag set, we do not have admin.

C:\Windows\system32>if '0' NEQ '0' (
echo Requesting administrative privileges...
goto UACPrompt
) else (goto gotAdmin )

C:\Windows\system32>pushd "C:\Windows\system32"

C:\Windows\system32>CD /D "D:\Workspace\pFaces-SymbolicControl\ex_gb_fp\deepracer_rt\"

D:\Workspace\pFaces-SymbolicControl\ex_gb_fp\deepracer_rt>rem RUN

D:\Workspace\pFaces-SymbolicControl\ex_gb_fp\deepracer_rt>pfaces -CG -d 1 2 -k gb_fp@.\..\kernel-pack -cfg .\deepracer_fast.cfg -co "project_name=DeepRacer1,online_mode=t
rue,online_mode_host=",online_mode_port=12345,save_controller=false,synthesis_pack=22" -v0
pFaces::Warning: The space-size is modified so that devices will have no unused cores or to match the needs of local memory of the GPU-based kernel!
New X-space size: 52224 symbols
New U-space size: 35 symbols

pFaces::Warning: A Blocking-SyncPoint job at index 6 is not necessary
  
```

- If (DeepRacer STEP 1) SSH to the robot times out repeatedly, try again. If you've tried over 15 times, reboot the computer and confirm the robot batteries are charged.

- If (DeepRacer STEP 5) the robot doesn't move after running closedloop_rt.py, check:

- Both batteries are fully charged
- Correct host URL is used in the symbolic control file

- To check robot packets, use:

`sudo tcpdump -w FileName.pcap`

Then inspect with Wireshark

- Add print statements in all control functions to trace execution.
If stuck, it's likely trapped in a while loop. Wrap suspicious function calls in try-except blocks.

Example:

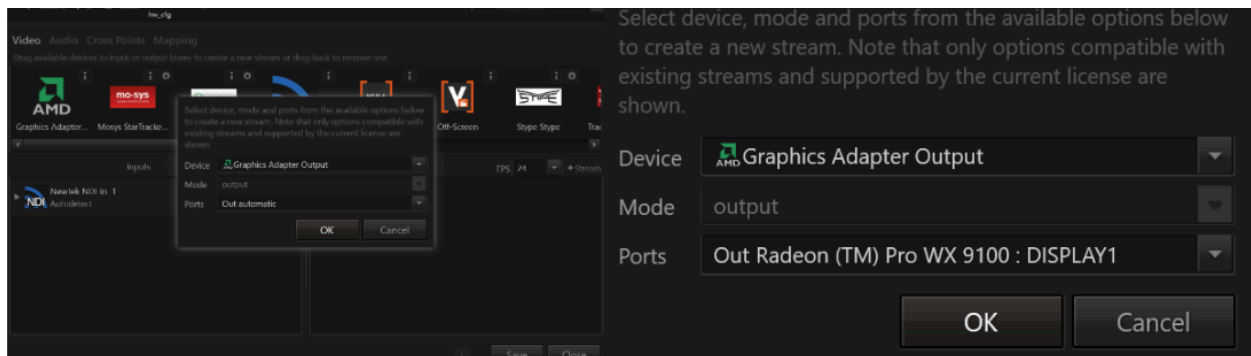
```
# Get the mode of the server
def getMode(self):
    return self.rest_client.restGETjson()["mode"]
```

- If DeepRacer Obstacle moves on its own without script input:
Shut down everything and reboot until it behaves correctly.
- Flushing DeepRacer Logs (if needed):
 1. Connect keyboard and monitor to the robot
 2. Boot the system
 3. If screen is black, press CTRL+ALT and cycle through Fn keys until terminal appears
 4. Run:

```
rm -r /home/deepracer/.ros/log/*
sudo rm -r /home/deepracer/.ros/log/*
[Optional] sudo rm -r /var/log/*
```
 5. Reboot the system

Ventuz - Multi-Projector Setup (for other issues I recommend reaching out to Ventuz on their Discord channel)

Problem: When trying to configure Ventuz to span across multiple projectors as a single large display, you may encounter an issue where only one display (DISPLAY1) is visible in the Ventuz configuration app, even though all projectors show signal outside of Ventuz.



Solution: (for Ventuz 6.9)

1. Clean Graphics Drivers:
 - Use AMD Cleanup Utility to remove existing graphics drivers
 - Reinstall the latest drivers for your AMD graphics card
2. Configure AMD Eyefinity:
 - Open AMD Radeon Software
 - Navigate to the Display settings
 - Set up Eyefinity to merge multiple displays into a single large display

- This creates a "spanned" display that Ventuz can recognize as a single output

3. Ventuz Configuration:

- After setting up Eyefinity, launch Ventuz
- The merged display should now appear as a single output option
- Configure this output in Ventuz for your multi-projector setup

For Ventuz 7+ (in case of future upgrades):

1. Windows 10:

- Ventuz 7+ can handle multiple outputs without requiring display spanning
- Add the AMD device to the Output panel 4 times (once for each output)

2. Windows 11:

- Due to Windows changes to Full-Screen Optimization, you may still need to use AMD Eyefinity/Mosaic mode
- This helps avoid sync issues between displays

Controller Graphics Card - HP Z8 G4 Workstation

System Information

- Model: HP Z8 G4 RCTO Base Model Workstation
- Property Code: UCB 215914 UCPC

- Serial Number: MXL12446V8

Problem 1: No Display Signal

Symptoms

- No display output
- May receive hardware error 3.3 (no graphic initialization)

Potential Causes

- Interference from Dual Port Thunderbolt 3 PCIe AiC card
- NVIDIA graphics card connection issues

Troubleshooting Steps (following official guide [HP Thunderbolt 3 PCIe Card Installation](#))

1. Disconnect the Dual Port Thunderbolt 3 PCIe AiC card
2. Ensure at least one NVIDIA card is properly connected
3. Power on the system
4. Once display is working, update BIOS to v.2.94 Rev A (for Win 10 x64) as recommended in the Dual Port Thunderbolt 3 PCIe AiC installation guide

Problem 2: RAM Memory Recognition (Error 3.2)

Symptoms

- Error 3.2 (no memory initialization)
- System fails to recognize installed RAM

Resolution: CMOS Reset Procedure

1. Power down the system completely:
 - Shut down the system
 - Disconnect the AC power cord
 - Remove all external devices
2. Reset the CMOS:
 - Locate the yellow CMOS reset switch on the system board
 - Press and hold for 5-8 seconds
3. Establish minimal configuration:
 - Keep only essential components:
 - System board
 - Power supply
 - Processor
 - One known working memory module
4. Restore connections:
 - Reattach the access panel
 - Reconnect the power cord
5. Test the system:
 - Power on the workstation
 - Verify the system completes POST (Power On Self Test)

Closeedloop_rt.py - Troubleshoot Summary & Next Steps

Issues with the original closedloop_rt.py

1. Missing Theta Normalization:

The original controller did not normalize theta values to the required [-1.7, 1.7] range. pFaces server rejects theta values outside this range, causing controller failures that were not identifiable due to lack of error handling.

2. Incomplete Hyperrectangle Formatting:

The `get_hyper_rec_str` function in `LocalizationServerInterface` only creates the first part of the required format and misses the fixed reference frame that pFaces uses as standardized computational domain.

The format returned:

`{x_min,x_max},{y_min,y_max},{-3.2,3.2},{0.0,0.8}/{-2.1,2.1}`

(object x and y coordinate bounds, calculated from OptiTrack server data, theta and velocity ranges (for targets/obstacles))

Required format:

`{x_min,x_max},{y_min,y_max},{-3.2,3.2},{0.0,0.8}|{2.0,3.0},{2.0,3.0},{-3.2,3.2},{0.0,0.8}`

(fixed x, y coordinate bounds, same theta and velocity ranges as in first part)

3. Poor Error Handling:

Bare except blocks without specific error information, diagnostic messages, or fallback mechanisms when server connections fail. Resulted in empty responses.

4. Connection Management Issues:

No timeout handling for server connections or caching of data to reduce server requests. Connection reuse causing state management problems.

Improved Controllers

1. closedloop_rt_robust.py

Added Theta Normalization:

- normalize_theta function clamps values to [-1.7, 1.7] range
- Sends warning message when theta values are being normalized

Improved Error Handling:

- Specific exception handling with detailed error messages
- Explicit mention of theta range issues in error messages
- Improved logging to help diagnose failures

2. robust_controller_py27.py

Fixed Hyperrectangle Format

Improved Connection Management:

- Better handling of HTTP connections
- Explicit error handling for connection timeouts
- *Fallback to hardcoded values when connections fail (should be removed)*

3. robust_controller_recalibrated.py

Theta Value Recalibration (instead of clamping): Mapped the full range [-3.2, 3.2] to [-1.7, 1.7] to reflect accurate information from the original theta values.

4. hybrid_controller.py

Adaptive Connection Strategy:

- Uses a 3-minute timeout for the initial connection to OptiTrack

- Uses a shorter 3-second timeout for subsequent requests
- Implements data caching to avoid unnecessary requests

Better Error Recovery:

- Graceful fallback to hardcoded values
- Clear error messages for different failure scenarios
- Proper cleanup of connections to avoid resource leaks

Latest Development: Server Connection Issue

1. Simple Test vs. Full Controller:

- The simple test version (based on `synth_and_test`) returned action controls from pFaces instantly
- The full controller (`robust_controller_py27`) needed 2-3 minutes to establish the initial connection

2. Potential Causes:

- Connection Reuse: The full controller may be reusing connections improperly
- Network Stack Differences: Different methods of making HTTP requests
- Error Handling Overhead: More complex error handling adding delays
- Python 2.7 Limitations: Older networking libraries with less efficient implementations. Python 2.7 vs. Python 3.8:
 - Python 2.7's networking stack is outdated and less efficient
 - Python 3.8 offers significant improvements in HTTP connection handling:
 1. Modern SSL/TLS implementation with better performance
 2. More efficient socket management
 3. Better timeout handling and connection pooling

4. Access to newer libraries like `requests` instead of `httplib2`

Tests designed to diagnose OptiTrack server connection timeout:

Each test builds on previous one to isolate specific issue causing the 3-minute delay:

1. Basic Connection Test: Tests direct HTTP connection to OptiTrack
2. RESTApiClient Test: Tests if the abstraction layer adds overhead
3. LocalizationServerInterface Test: Tests if the higher-level interface causes delays
4. pFaces Connection Test: Tests if the delay is with pFaces specifically
5. Synthesize Controller Test: Tests the controller synthesis step
6. Connection Reuse Test: Tests if reusing connections causes increasing delays
7. Full Controller Sequence Test: Breaks down the entire process with timing for each step

What to Look For

- If `test6_connection_reuse.py` shows increasing delays with each reused connection, that suggests connection reuse is the issue
- If `test2_rest_client.py` is much slower than the basic connection test, that points to the `RESTApiClient` implementation
- If specific steps in `test7_full_controller_sequence.py` add significant overhead, that indicates where error handling might be causing delays
- If all tests show consistent delays that don't appear in Python 3.8, that confirms Python 2.7 limitations

Next Steps:

1. test out the seven diagnostic tests above
2. if diagnostic unsuccessful, install Python 3.8