Program documentation

# Dashboard

Used to collect data from the UR controller. You are also able to start, stop, reset safety, unlock protective, close connection, power on, power off, and release the break

Uses socket to read data from controller. Access controller by IP address on port 29999. Sends the data over websocket to ‘ws://localhost:4007’. Host IP address and server IP address need to be sent for this program to work correctly.

Under normal conditions. Status will be sent to the user interface. On startup of the rover the command “power on” will be sent. After mode changes to idle “brake release” should be sent. After this arm is controllable. When not using the battery to save power a power off command can be sent. This reduces the power consumption from 100W to around 10W. (need to verify usage)

## Data Produced

Produced data is sent once per second. Will send to the server with the message “dashboardData”

Most data is pulled at that rate. A few items are pulled only when a command is sent.

Produced message will be in the following format

data={

"msg": "dashboardData",

"data":

{"Connected": true,

"robotmode": "Robotmode: POWER\_OFF",

"Polyscope Version": "URSoftware 5.12.4.1101661 (Aug 31 2022)",

"SafetyStatus": "Safetystatus: NORMAL",

"In Remote": "true",

"Serial": "20195399999",

"model": "UR3",

"Unlocking Protective": "Unknown",

"Quit": "unknown"

}}

Data pulled once a second is:

* Connected state – Returns true or false depending on if UR controller is connected
  + Possibilities are True or False
* Robot mode – sends the current mode of the robot
  + Possibilities are no\_controller, disconnected, confirm\_safety, booting, Power\_off, Power\_on, Idle, Backdrive, running
* Polyscope Version – Returns the software version installed
* Safety Status – Returns the current status of the safety system
  + Possibilities are – Normal, reduced, protective\_stop, recovery, safetgaurd\_stop, system\_emergency\_stop, Robot\_emergency\_stop, Violation, Fault, Automatic\_mode\_safegaurd\_stop, System\_three\_position\_enablin\_stop
* In Remote – Returns if the remote is in remote control local control mode.
  + For commands to be send robot must be in remote mode. This change must be done from the teach pendant
  + Returns True or False
* Serial – Returns the serial number of the robot
* Model – Returns the model of the robot arm connected
* Unlocking Protective – Returns the state of unlocking the protective stop (updated on command)
  + Will be unknown until the unlock protective command is sent to the controller.
  + On success message will be “Protective stop releasing”
  + On Failure message will be “Cannot unlock protective stop until 5s after occurrence. Always inspect cause of protective stop before unlocking”
* Quit – Used to disconnect the robot will show disconnected if robot is disconnected (updated on command)

## Data Consumed

Is pull as needed. Looks for a message called “dashboardCMD”.

Accepts the following messages:

* “shutdown”
  + Used to turn off the power to the controller.
  + Note: this may fault the program reading from the controller. Only do this under controlled circumstances
* “power off”
  + Turns the controller to the off state. Polyscope is still running but no power is applied to the arm
  + Can be run at any time
* “power on”
  + Turns the controller to the idle state. Arm will not have power after powering on
  + Can be run when mode is power off
* “brake release”
  + Can be run from power off or idle state will apply power to robot arm.
  + State will be running after running this command
* “unlock protective stop”
  + If a protective stop triggered this will reset the fault.
  + Note: must wait at least 5 seconds after failure to start robot again
  + Must power on and release brake after running
* “restart safety”
  + If safety is triggered you can run this to reset the safety
  + You will need to run power on and brake release after running this command
* “quit”
  + Used to close the connection to the rover.
  + Note: the way the program is written after running this the program will stop and start again resetting the connection.

# Connecting to Casual

Because Casual is a web based application there are two ways to connect. If you are running casual from the same PC as the Node server you can use ws://localhost:4007 to connect to the nodejs server. If you are trying to connect to a remote server currently you have to use a tunnel. Using localtunnel.me you will generate a randome HTTPS:// address. You will change the HTTPS to wss with the randomly generated URL to access the data.

The simplest form of connected and reading the data from the socket in causal is the following.

//let socket = new WebSocket("ws://localhost:4007");

let socket = new WebSocket("wss://sad-mice-read-162-191-47-209.loca.lt");

socket.onopen = function(e) {

  os.toast("[open] Connection established");

  os.toast("Sending to server");

  socket.send('{"msg": "dashboardCMD", "data": "shutdoswn"}');

  socket.onmessage

};

socket.onmessage = function(event) {

  console.log(`[message] Data received from server: ${event.data}`)

  return event.data

}