VIM-303

User Interface Manual

VISUAL ROBOTICS

For Firmware Release 2.6.1

Scope and other documentation

This manual covers the web User Interface of VIM-303. Other relevant manuals include:

- Unboxing and Hardware Assembly Manual
- Settings Manual
- Blockly Programming Manual
- First Picks with VIM-303 Manual

Connecting to VIM-303 User Interface

The VIM-303 vision guidance system (the camera) hosts a webpage at its host address on standard http port 80. The host address of the camera can be accessed in two ways:

- IPv4 address, found via the router or static address, such as 192.168.0.22
- mDNS address, which is found on the label on the back of the camera, including the suffix ".local", for example vrobotics88ec.local, as shown on the camera below (Figure 1)

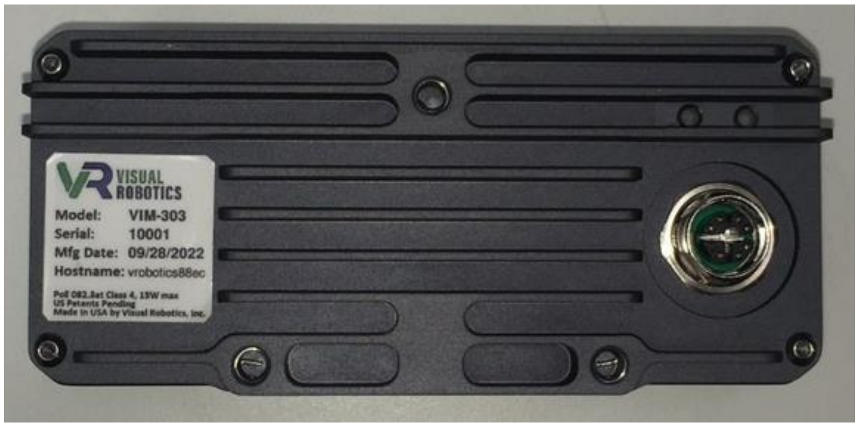
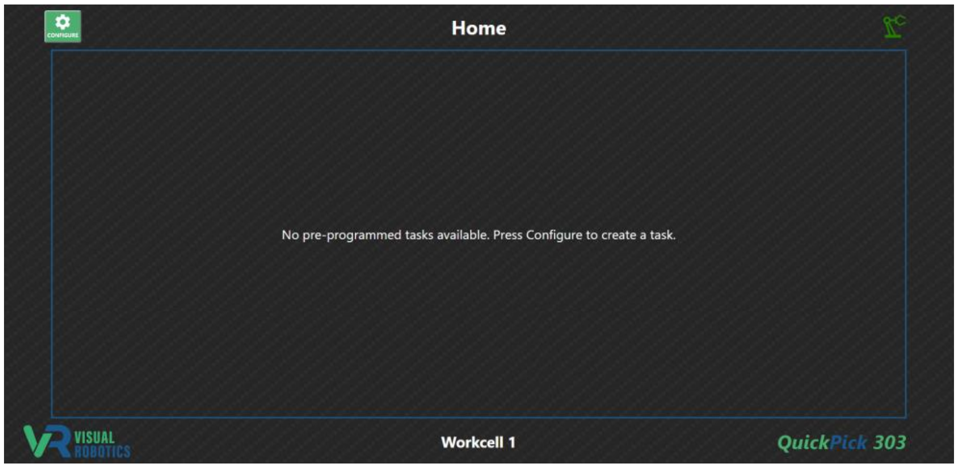


Figure 1 - Back side of VIM-303 camera, showing hostname

For example, the camera in Figure 1 could be accessed in a browser (such as Google Chrome) by navigating to either the address **192.168.0.22** or **vrobotics88ec.local**.

Task Menu

The first menu that the Operator will see is the **Task Menu**. This is used by the Operator of a VIM-303 workcell to select tasks (programs) to run. The first time the camera is used, there will be no programmed tasks and the page will indicate that the **Configure** gear must be pressed by the Programmer to create new tasks (Figure 2). If tasks (Blockly programs) have been created, they will show up on the task menu as green buttons labeled with the filename of the associated Blockly program (Figure 3).



No pre-programmed tasks available. Press Configure to create a task.

Figure 2 - Task Menu with no tasks

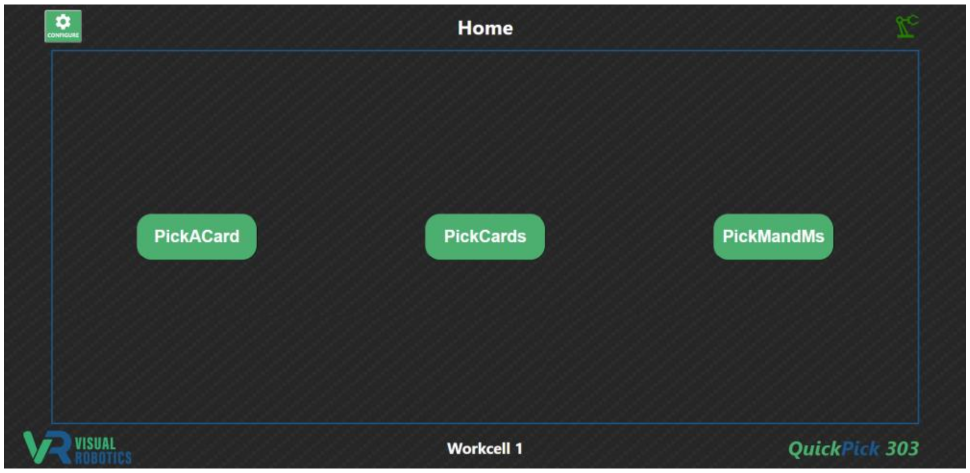


Figure 3 - Task Menu with programmed tasks (Blockly programs)

The Programmer is able to organize the appearance of the Task Menu for the Operator by creating **Folders**, which may contain tasks and/or other folders. Tasks and folders can be hidden, for example to hide obsolete versions of tasks. For example, Figure 4 shows the menu tree from the Task Menu Editor (described in the programming section), which is used by the Programmer to organize the appearance of the Task Menu with a folder called **Cards** containing the tasks **PickACard** and **PickCards**.

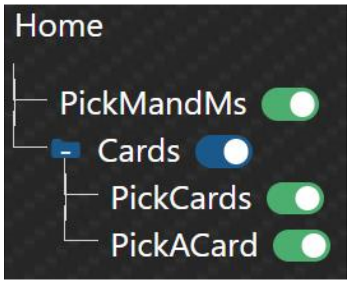


Figure 4 - Task Menu Editor menu tree

Figure 5 shows the Task Menu when configured by the Programmer as shown in Figure 4. The task **PickMandMs** and folder **Cards** appear at the home screen. If the Operator presses the **Cards** folder, the two tasks **PickCards** and **PickACard** are shown. The Operator can press the blue up-arrow button to leave the **Cards** folder to the **Home** level.

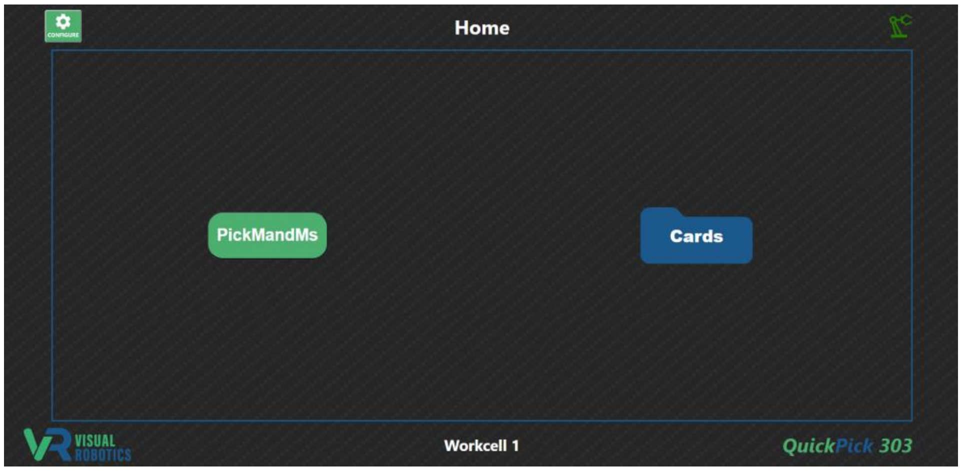


Figure 5 - Task Menu with a subfolder

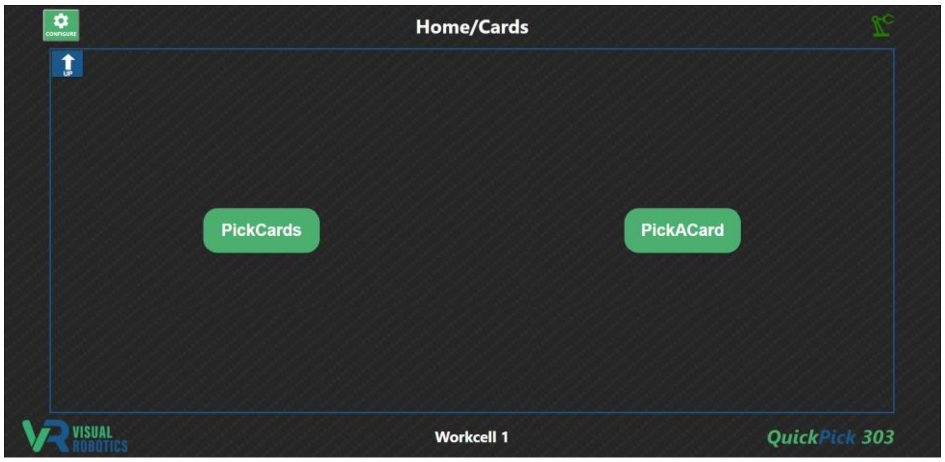


Figure 6 - Clicking into a subfolder

When an Operator presses one of the green task buttons, such as the PickCards button, the task is loaded and is prepared to be executed. A green **Start** button appears (Figure 7). When the Operator presses the **Start** button, the task executes and a red **Stop** button appears while the task is executing (Figure 8). When the task completes, the button will change to a green **Start** button. If the Operator wishes to stop the program before completion, they may press the **Stop** button and the task will execute whatever steps it may have been programmed to execute when the stop button is pressed and then the task completes and the button changes to a green **Start** button.

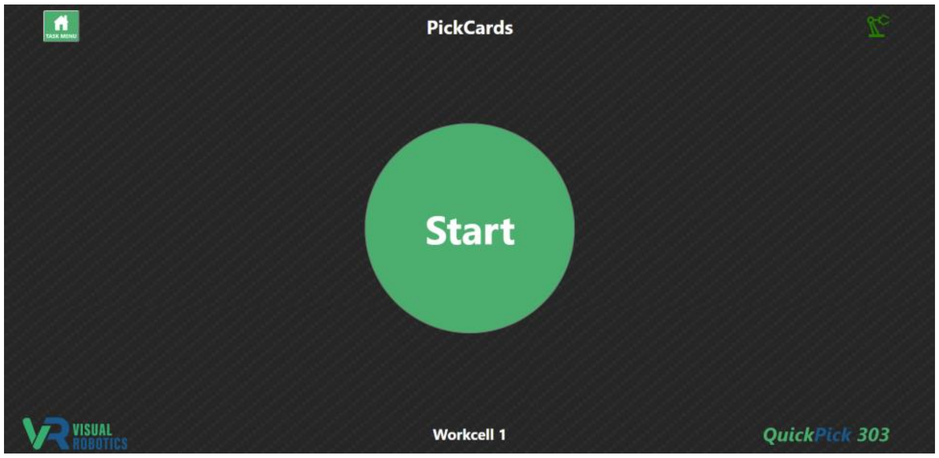


Figure 7 - task Start button

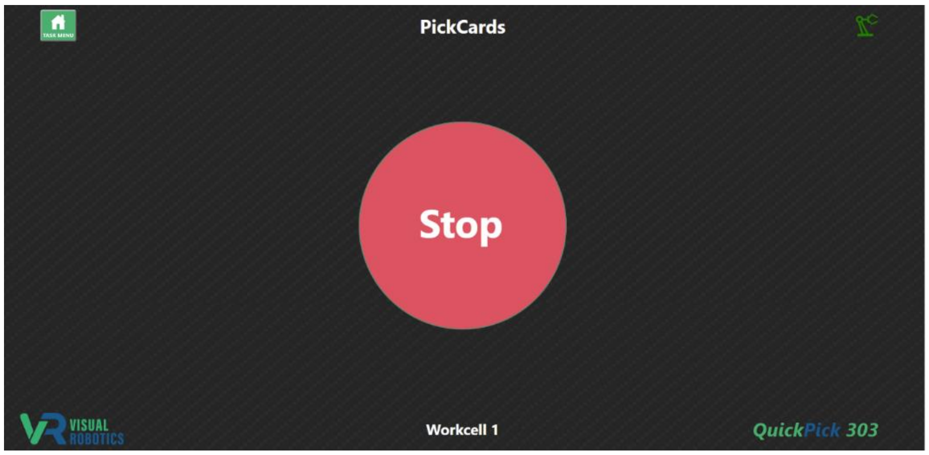


Figure 8 - task Stop button

Configuring (Programming) Tasks

To create a new task or modify the behavior of an existing task, the Programmer presses the **Configure** gear from the task menu (Figure 9). The Programmer is presented with several tabs (Figure 10) including Workspaces, Objects, Waypoints, Blockly, Camera, Settings, and Tasks, which will each be described further in this manual.

Returning to the Task Menu when the Programmer has completed the creation of new tasks, is accomplished by pressing the **Task Menu** button (home icon), allowing the Operator to execute tasks with the workcell.

Administrative tasks, such as viewing logs of the system behavior, network administration, and updating firmware, are accessed by pressing the **Admin** button (padlock icon).

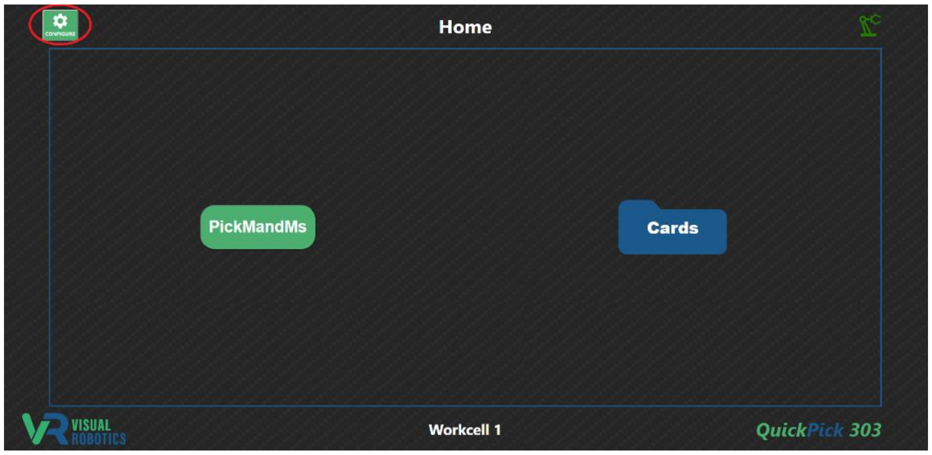
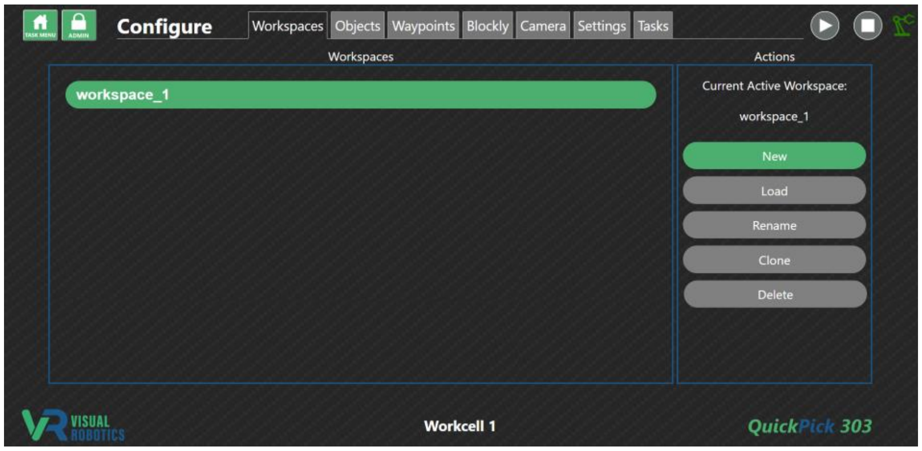


Figure 9 - Configuring (Programming) tasks



Current Active Workspace:
workspace_1

Figure 10 - Configure (Programming) Tabs

Workspaces Tab

Workspaces are a collection of Objects, Waypoints, Blockly tasks (programs), and Settings. It is often convenient for multiple Blockly tasks to share common Objects, Waypoints, and Settings. These tasks should reside in the same Workspace. However, it may be beneficial to create a new Workspace, in order to have a different set of Objects, redefine the location of Waypoints with the same name, or use different settings (for the camera, gripper, etc).

The **Workspaces Tab** (Figure 11) allows the Programmer to manage Workspaces. A camera fresh from the factory will have one Workspace defined, **workspace_1**. New Workspaces can be created by pressing the **New** button (Figure 12), and will create a new Workspace with no Objects, no Waypoints, no Blockly tasks, and factory default settings.

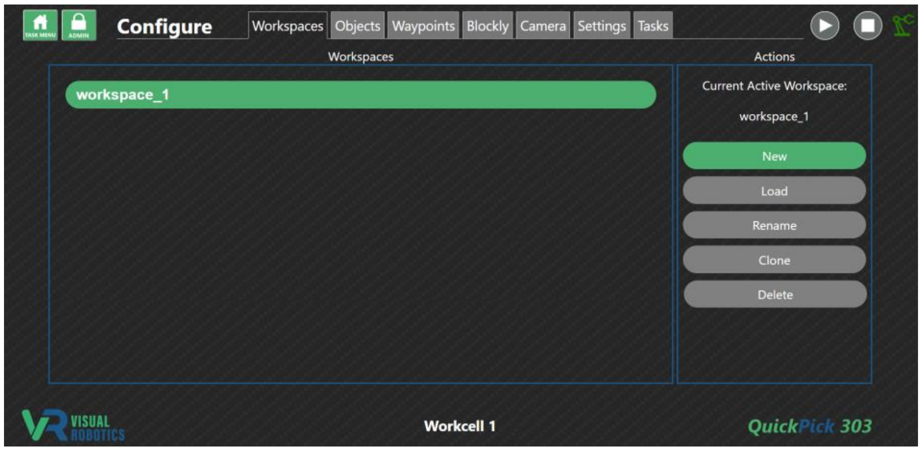


Figure 11 - Workspaces Tab

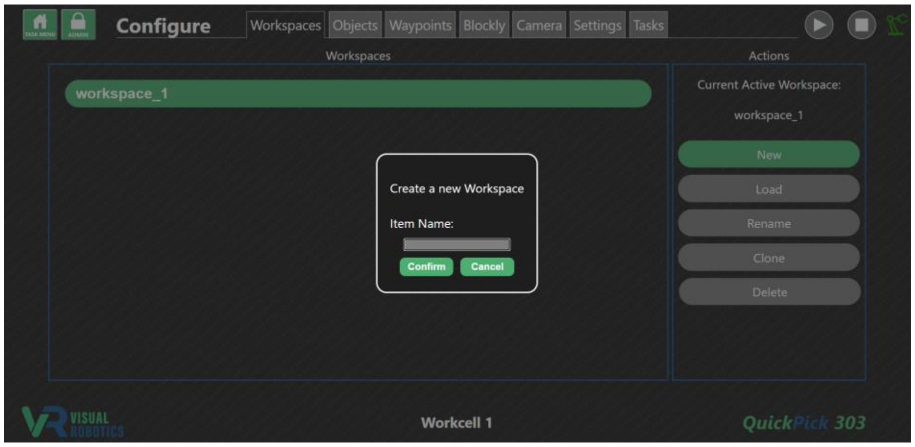


Figure 12 - Creating a New Workspace

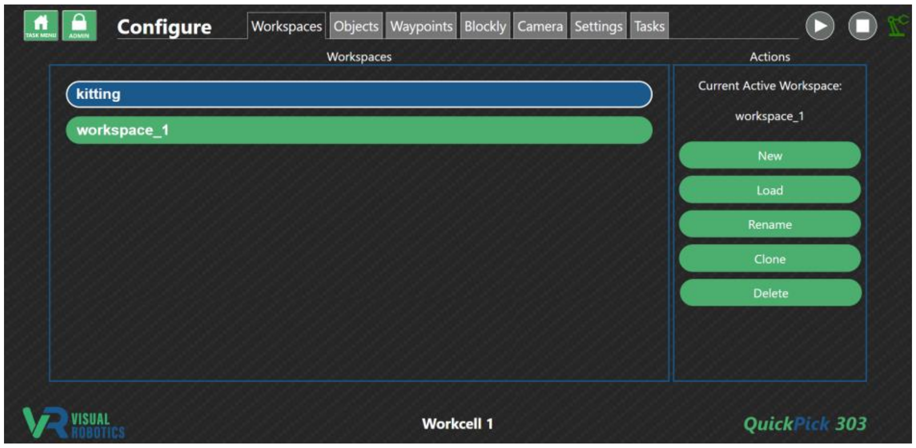
Figure 13 shows two Workspaces. The active Workspace (for example workspace_1) is indicated in green and other Workspaces are indicated in blue. Single clicking on a Workspace selects it (for example kitting) and is shown as outlined in white.

Double click on a Workspace to make it active or select it and press the **Load** button. All of the Objects, Waypoints, Blockly tasks, and Settings will be loaded.

Rename a Workspace by selecting it and pressing the **Rename** button.

A copy (clone) of a Workspace can be created by selecting the Workspace and pressing the **Clone** button. This may be useful to create a copy of another Workspace's Objects, Waypoints, Blockly tasks, and Settings as a starting point and then change them for a specific purpose.

A Workspace that is not active can be deleted by selecting it and pressing the **Delete** button. The active Workspace cannot be deleted. There must be at least one Workspace.



Current Active Workspace:

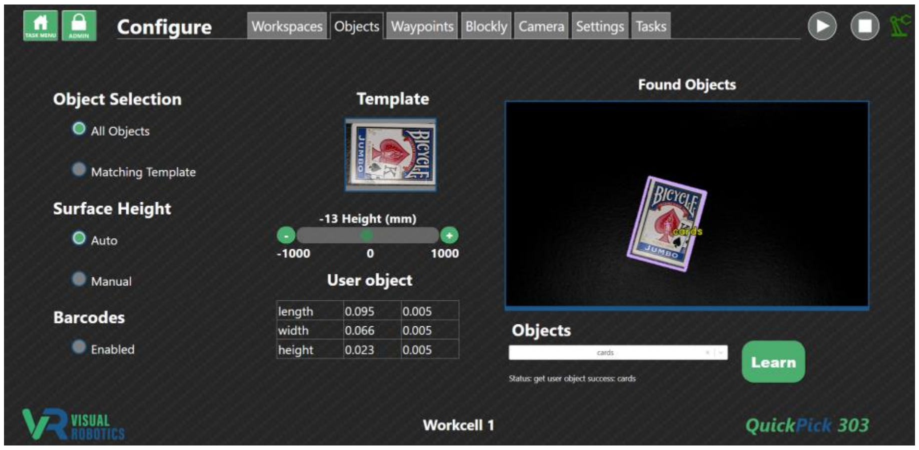
workspace_1

Workcell 1

Figure 13 - Selecting a Workspace

Objects Tab

The **Objects Tab** is used to define objects for the VIM-303 camera to see (Figure 14). Objects are detected using the Depth Finder, which measures the dimensions and 3D location of prismatic objects (objects that have a flat top surface) on a flat surface. In subsequent software releases, more complex objects may be detected. Software version 2.6.0 requires that the camera be pointing straight down for object detection. See the **Waypoints Tab** for modifying the camera's location on the robot to aim it straight down if needed. If the object is sitting on a surface that covers most of the camera's field of view, the **Surface Height = Auto** button can be selected. The surface and any other background below it will show up dimmed and gray in the **Found Objects** image. The object of interest should be shown in color and outlined in purple (Figure 14). If the object is sitting on a relatively narrow surface, the object of interest may not become outlined in purple, or perhaps the entire surface will be outlined in purple. If this is the case, then the surface height may need to be manually defined. Select **Surface Height = Manual** and adjust the **Height** slider (by sliding the green dot or clicking the + or - buttons) until the surface height (in mm) matches the height of the surface. This can be determined by using the **Waypoints Tab** and jogging the robot to have the end effector touch the surface and remembering the Z coordinate where it touches.



length	0.095	0.005
width	0.066	0.005
height	0.023	0.005

Objects: cards

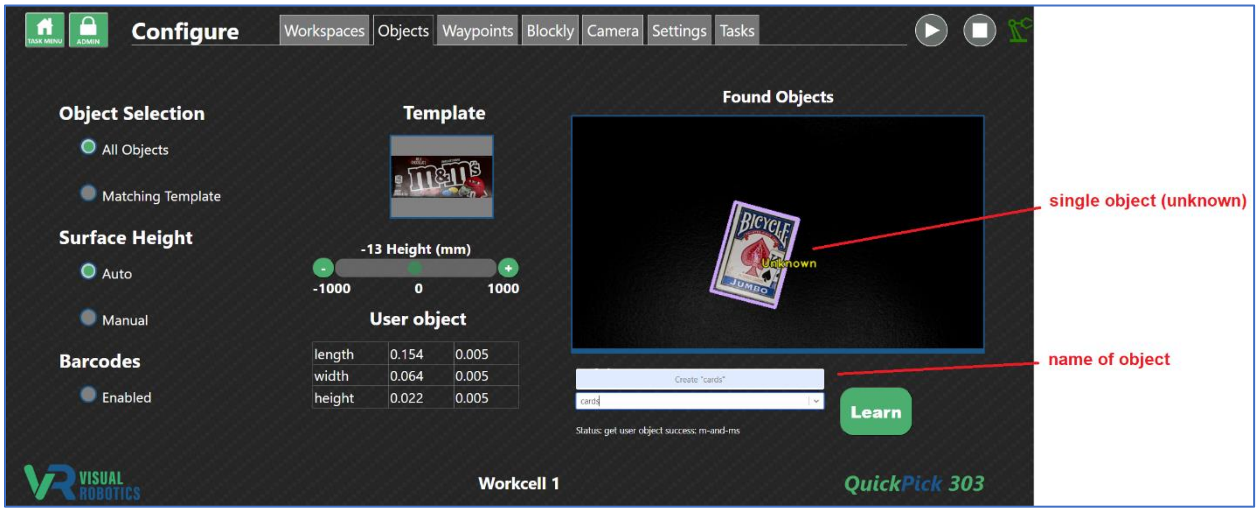
Status: get user object success: cards

Figure 14 - Objects tab

To teach a new object, ensure that only one object is in the field of view of the camera. The object should be outlined in purple and may be labeled as **Unknown** if the camera has never been taught this object before (Figure 15). Type the name of the object in the **Objects** pulldown menu. Press **Create** when prompted. The object's label should change to match the name that was defined (Figure 16). The dimensions should match that of the object (the first column in the table are the dimensions in meters, and the second column specifies the matching tolerance in meters). A template picture is captured at the time of teaching, for use in future versions of software that include template matching.

An existing object can be retrained by placing a single copy of the object in the field of view, selecting the object name from the menu and pressing **Learn**. The template image and dimensions will be updated accordingly.

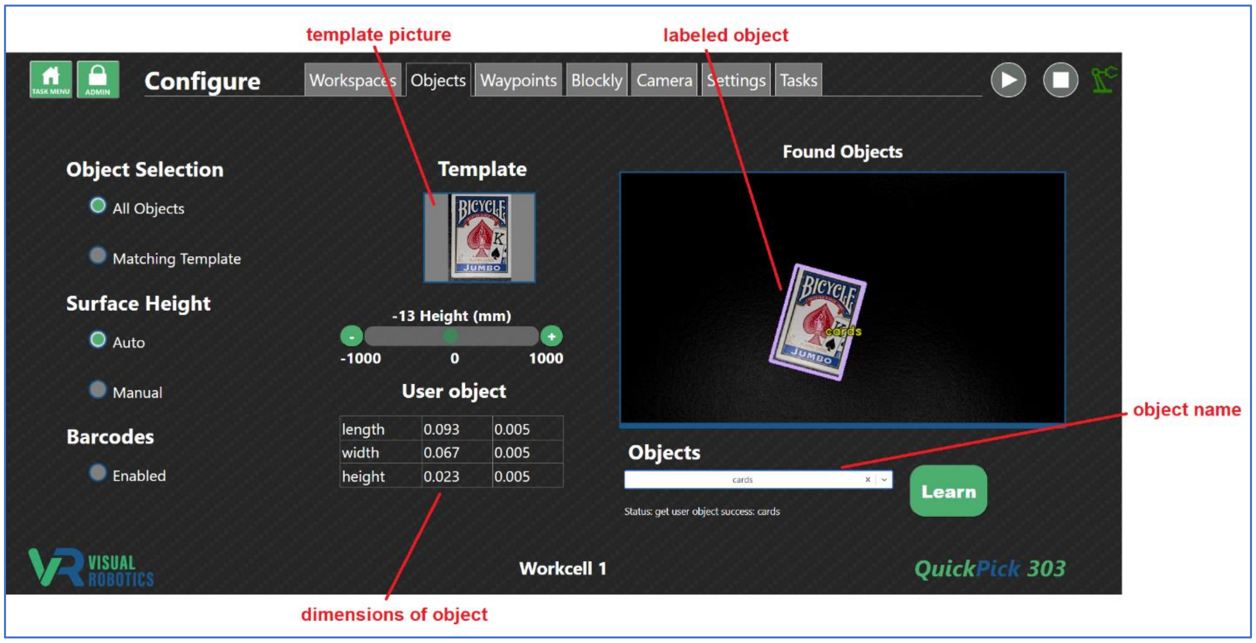
Barcode reading is not supported in v2.6.0. Checking **Enabled** has no effect.



length	0.154	0.005
width	0.064	0.005
height	0.022	0.005

Status: get user object success: m-and-ms

Figure 15 - Create a new object



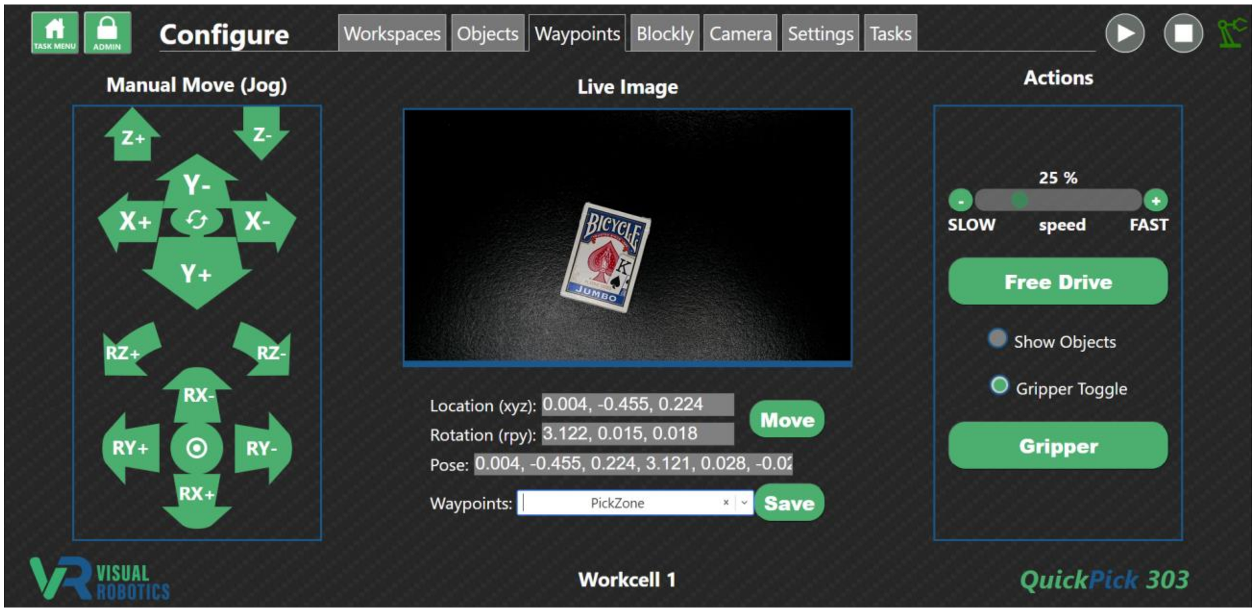
length	0.093	0.005
width	0.067	0.005
height	0.023	0.005

Status: get user object success: cards

Figure 16 - Object definition

__Waypoints Tab__

The **Waypoints Tab** allows the Programmer to define Waypoints, which are locations where the robot is programmed to move (Figure 17). Waypoints are created by jogging the robot to a particular location, with the **Manual Move (Jog)** buttons, typing in a robot pose (by using the entry box below the Live Image), or by pressing the **Free Drive** button, which causes the robot to be moved by hand to the desired location.



Location (xyz): 0.004, -0.455, 0.224

Rotation (rpy): 3.122, 0.015, 0.018

Pose: 0.004, -0.455, 0.224, 3.121, 0.028, -0.0

Waypoints: PickZone

Figure 17 - Waypoints tab

Press the **Free Drive** button (it will become outlined in white) to cause the robot to be easily moved by hand to a new location. Press the **Free Drive** button again to disable this mode when you have arrived at the desired location. The **Manual Move (Jog)** buttons can be used to fine-tune the position.

The **Gripper** button can either be momentary or toggle, depending on the state of the **Gripper Toggle** button. The **Gripper** button is used to manually enable the end effector for testing.

The **Jog Speed** slider is used to set the speed of manual movement (jogging) as well as the speed of moving to waypoints.

The **Manual Move** buttons - Jog Panel (Figure 18) allow the Programmer to move the robot in different directions and aim the camera in different orientations. The **Rotate Orientation** button is not implemented in version 2.6.0 but is intended to allow the user to orient the jog buttons to match the direction that the operator is facing the robot. The bump at the bottom of the oval represents the orientation of the cable sticking out of the robot base. The bullseye shaped **Aim Straight Down** button aims the camera directly downward, which is required for 4 degrees-of-freedom (4DOF) visual picking.

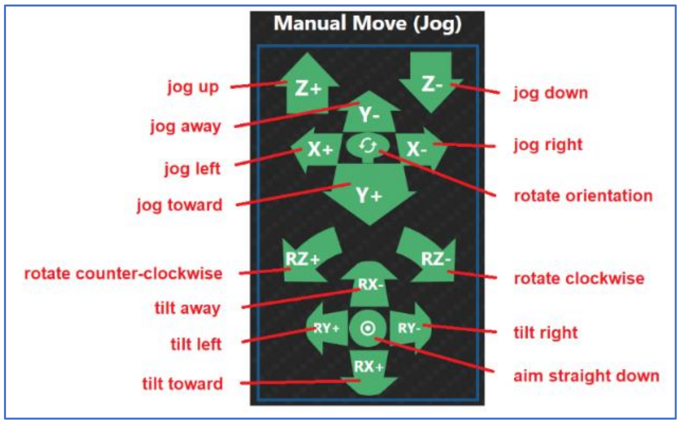
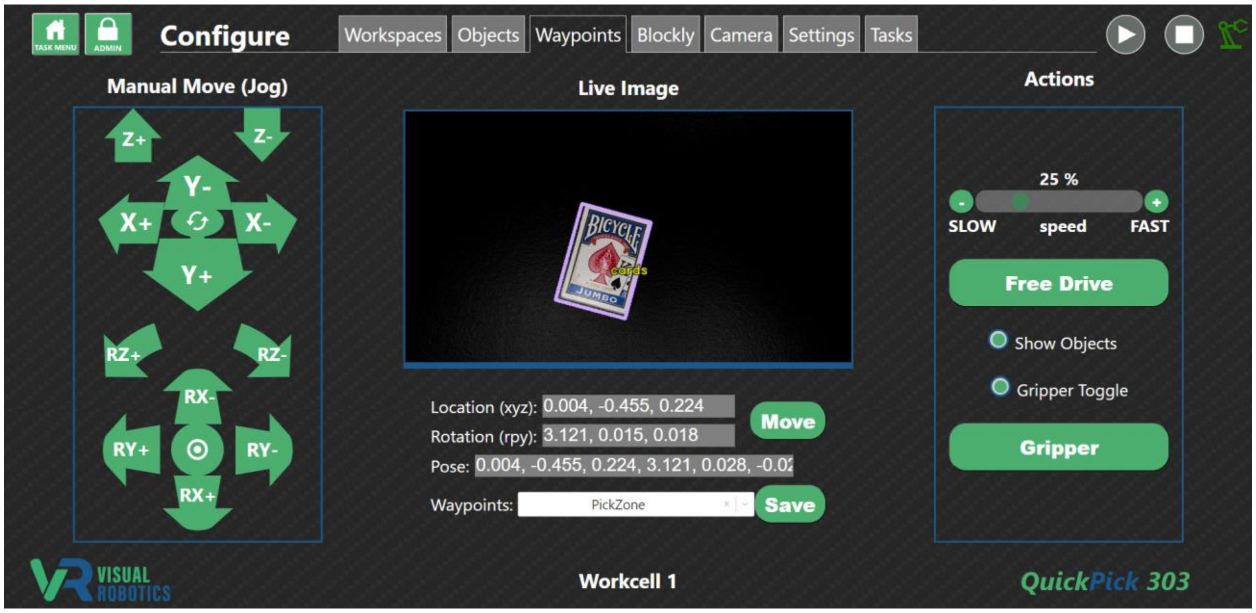


Figure 18 - Jog Panel

The center panel on the Waypoints tab displays the live image. Depending on the state of the **Show Objects** button, the displayed image is either the raw image from the color camera or the annotated image from the image processing algorithm (the **Finder**), as shown in Figure 19. The coordinate readout shows the current 6DOF pose (in meters and rotation vector, per the Universal Robot convention). The (X,Y,Z) location and rotation (in roll, pitch, yaw format) is also shown. In addition to being a readout display, the values can be changed by the Programmer by clicking into the boxes. The current location of the robot is shown in gray. If a waypoint is selected that is at a different location than the current location, the readout will show in yellow.



Location (xyz): 0.004, -0.455, 0.224

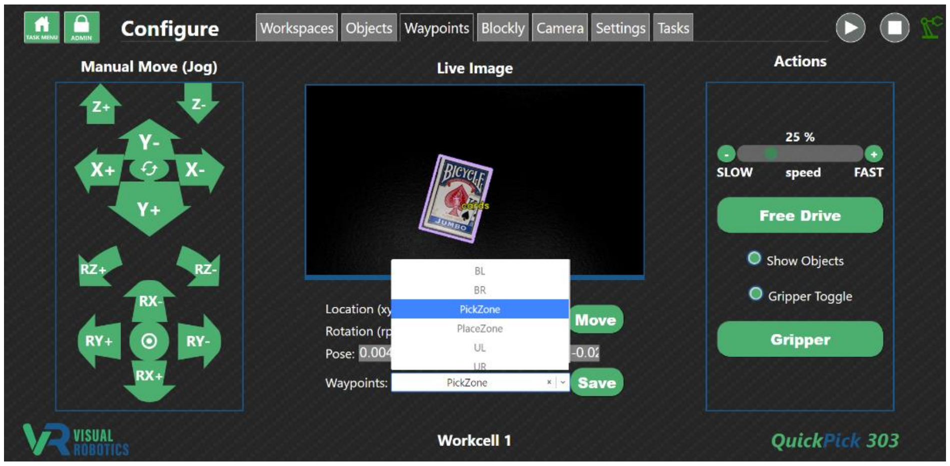
Rotation (rpy): 3.121, 0.015, 0.018

Pose: 0.004, -0.455, 0.224, 3.121, 0.028, -0.0

Waypoints: PickZone

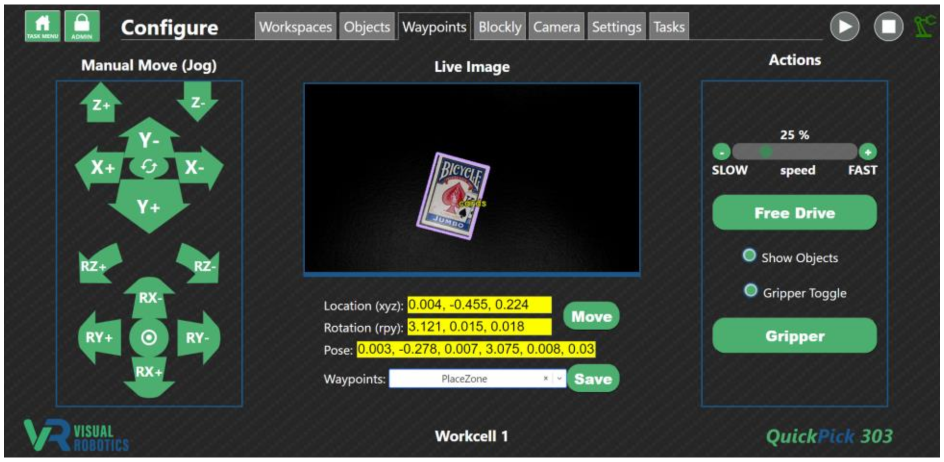
Figure 19 - Live Image with Show Objects enabled

The Programmer can select an existing waypoint from the pulldown menu (Figure 20). Its coordinates will show up, either in gray if this waypoint is at the robot's current location, or in yellow if is different from the current location (Figure 21). When the Programmer presses the **Move** button, the robot will move to that location. Then the display will turn gray to indicate the current location is being displayed (Figure 22).



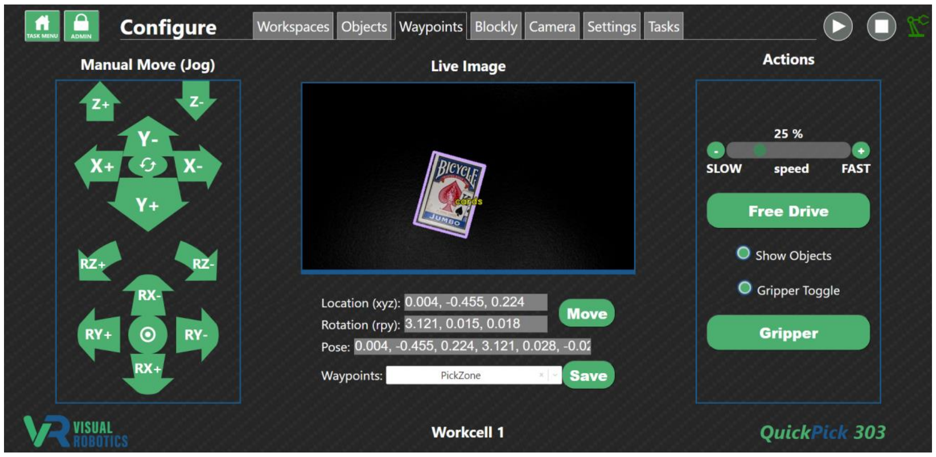
Location (xy
Rotation (rp
Pose: 0.004 … -0.02
Waypoints: PickZone

Figure 20 - Selecting a Waypoint



Location (xyz): 0.004, -0.455, 0.224
Rotation (rpy): 3.121, 0.015, 0.018
Pose: 0.003, -0.278, 0.007, 3.075, 0.008, 0.03
Waypoints: PlaceZone

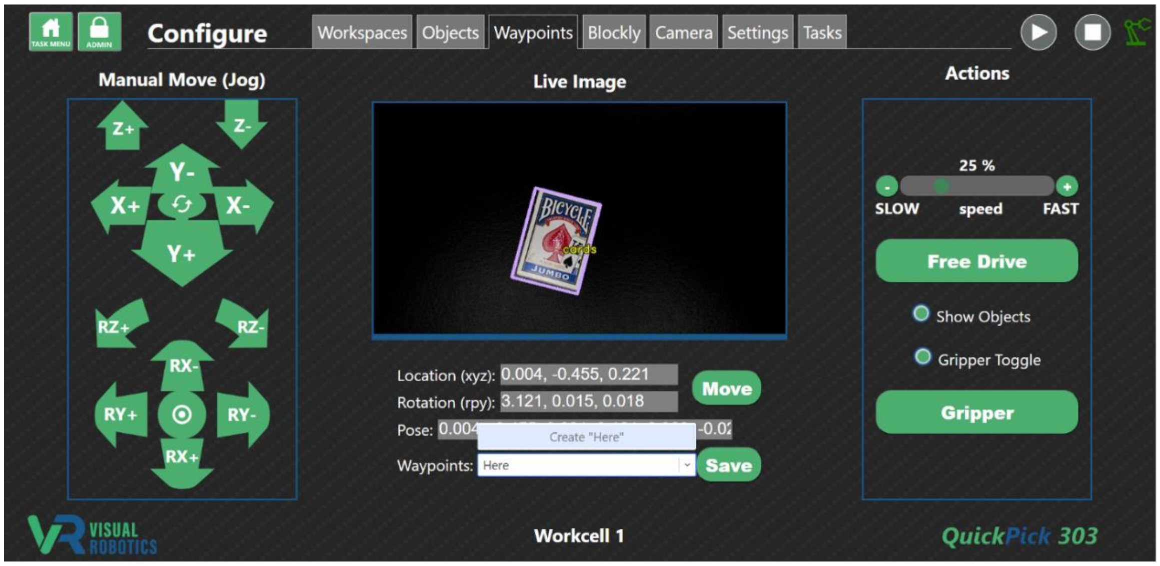
Figure 21 - Waypoint different from the current location



Location (xyz): 0.004, -0.455, 0.224
Rotation (rpy): 3.121, 0.015, 0.018
Pose: 0.004, -0.455, 0.224, 3.121, 0.028, -0.02
Waypoints: PickZone

Figure 22 - Move to Waypoint

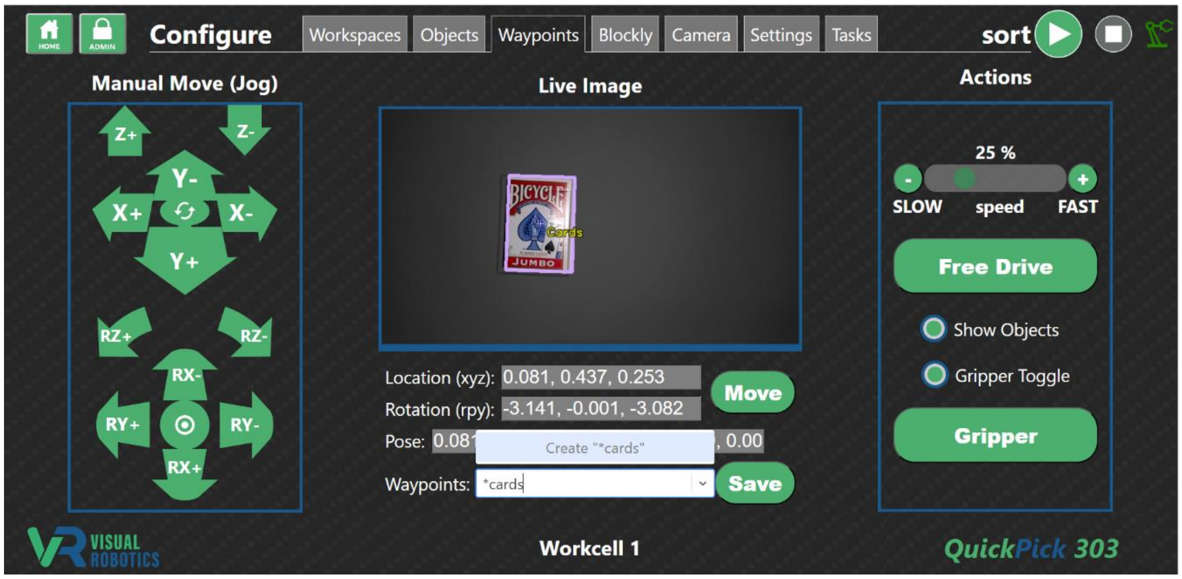
New waypoints are created when the Programmer types the name into the waypoint box. A popup saying Create "waypoint" will appear (Figure 23). The Programmer presses the popup to save the current location as the new waypoint. To save the current location and override the previous location stored in a waypoint, the Programmer would press the **Save** button. Waypoints are deleted by pressing the **X** button next to the waypoint.



Location (xyz): 0.004, -0.455, 0.221
Rotation (rpy): 3.121, 0.015, 0.018
Pose: 0.004 -0.02
Create "Here"
Waypoints: Here

Figure 23 - Create a new Waypoint

Visual Waypoints are waypoints that are defined based on the top-center location of an object within the field of view (Figure 23B). With a single object in the field of view (known or unknown), create a waypoint that begins with an asterisk (*). This signifies that the waypoint should be created at the top-center of the visible object, rather than at the current robot's position. For example, in Figure 23B, the waypoint ***cards** is created. The waypoint's location becomes the top-center of the deck of cards visible to the camera. This is a powerful method for creating waypoints by viewing objects in the field of view. For example, to set the four waypoints needed to specify a pallet, a box could be selectively moved to the four corners and four visual waypoints created while the camera is viewing the scene.



Location (xyz): 0.081, 0.437, 0.253
Rotation (rpy): -3.141, -0.001, -3.082
Pose: 0.081 , 0.00
Create "*cards"
Waypoints: *cards

Figure 23B – Create a Visual Waypoint

Blockly Tab

The **Blockly Tab** is where the Programmer creates Blockly **Tasks** (programs) for the Operator run (Figure 24). Blockly is a graphical programming language developed by Google. The Programmer clicks on the **Toolbox** panel to choose a programming block and places it on the programming **Canvas** to create the program (Figure 25). Every program starts with the **start block**, which is where the program starts execution when the **Start** button is pressed.

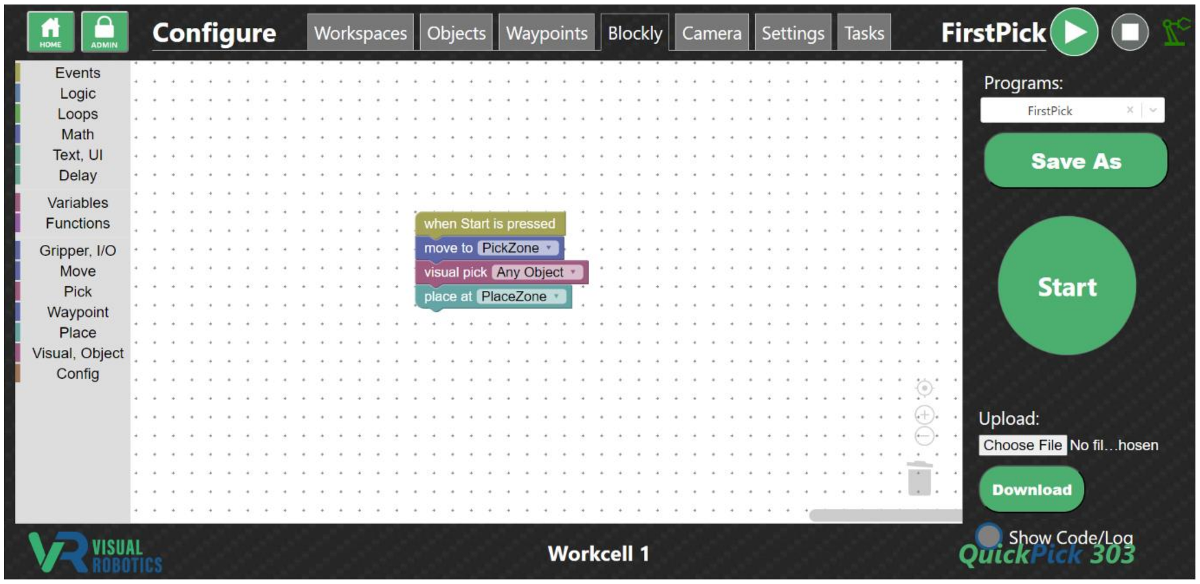


Figure 24 - Blockly Editor

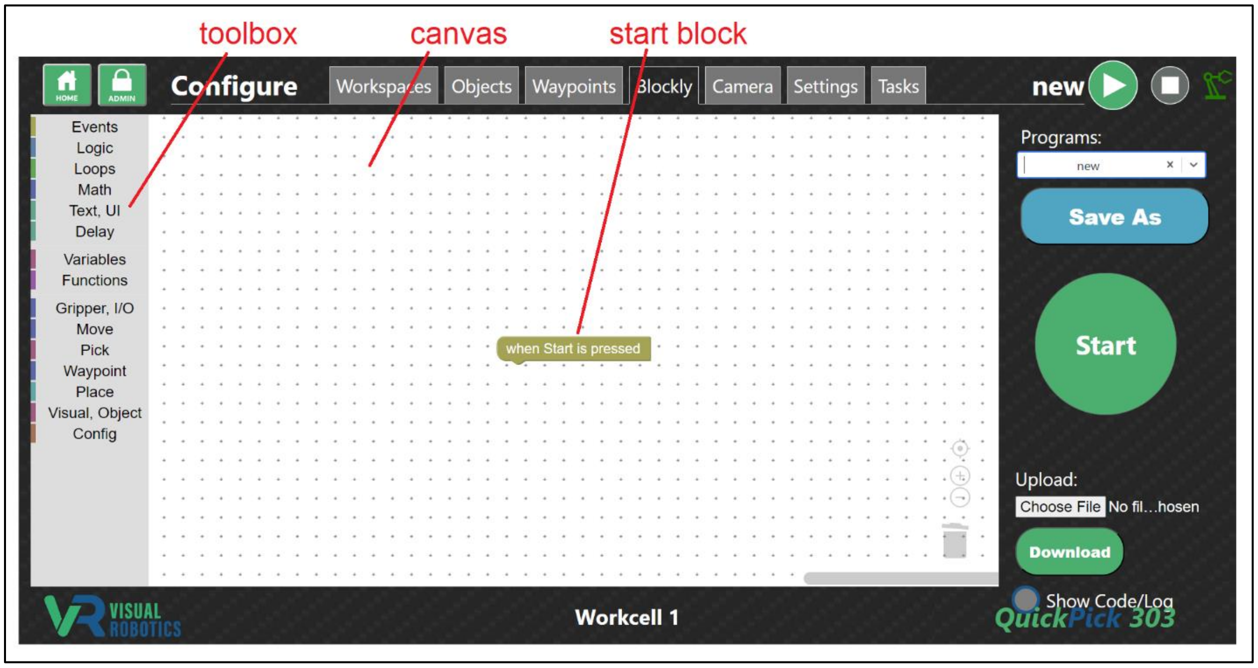


Figure 25 - Blockly Editor Toolbox and Canvas

Select the **Task** (program to load) by selecting with the pull-down menu (Figure 26). The Programmer can save the program by pressing the **Save As** button and choosing the filename (Figure 27). A new program is created when the Programmer types the **Task** name in the program box.

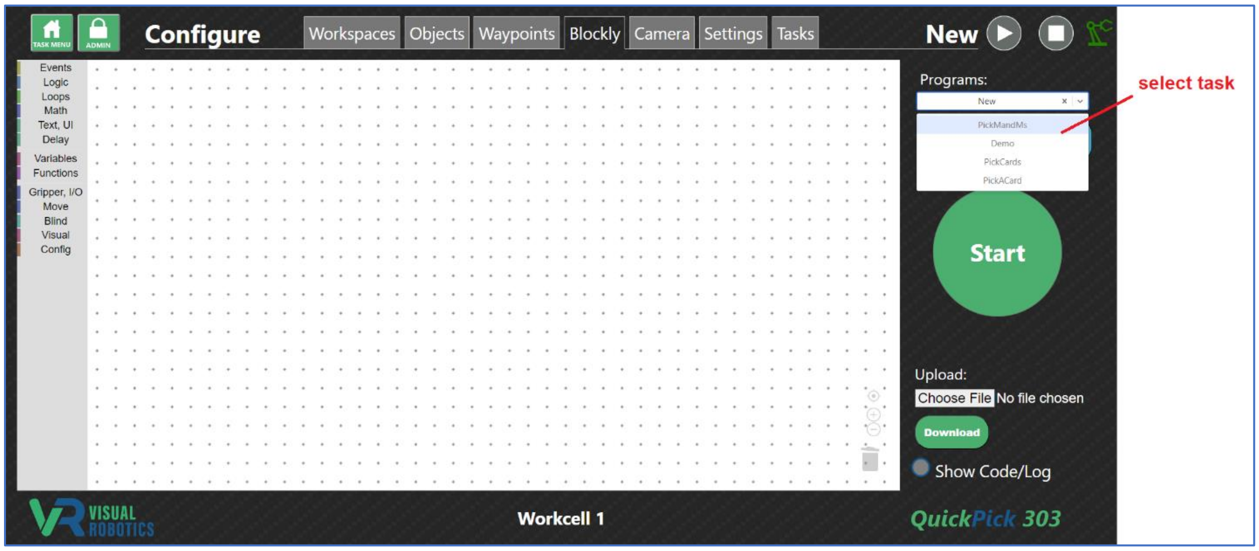


Figure 26 - Select program (task)

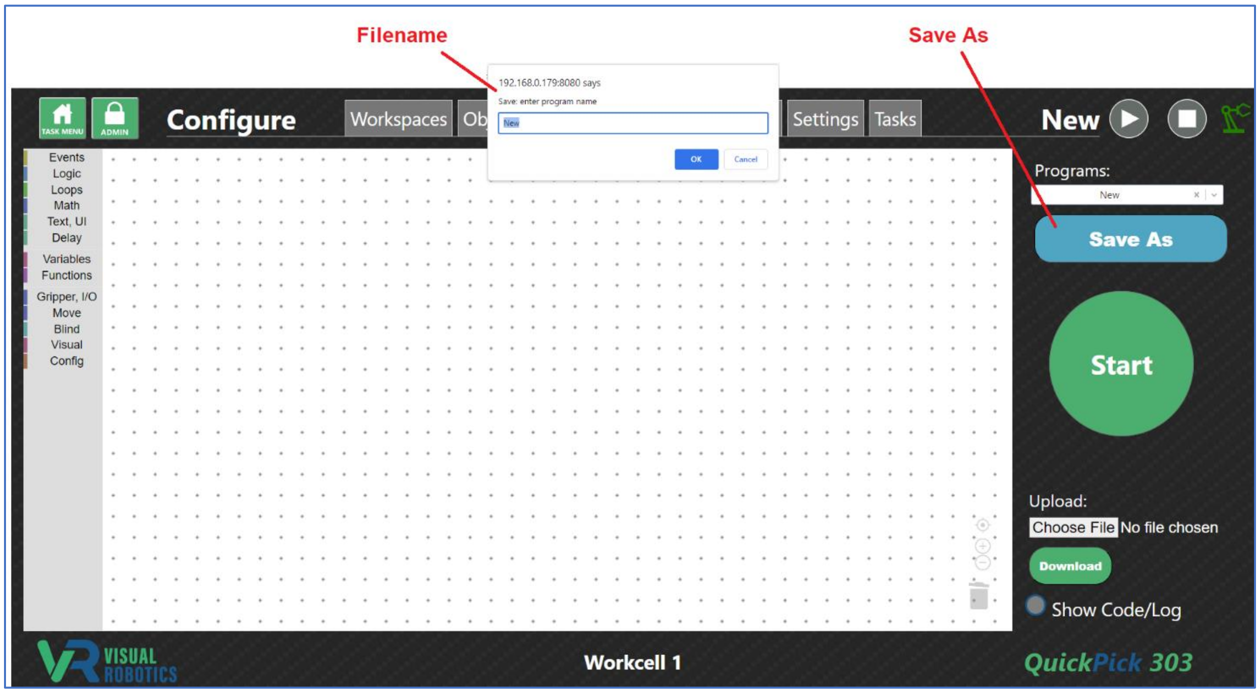


Figure 27 - Save program (task)

Clicking on a category in the **Toolbox** shows the list of blocks in that category, for example, Loops (Figure 28). Clicking on a block in the category places it on the **Canvas**. Other blocks can be chosen and slid into place. They click when they attach to a compatible block (Figure 29). A full list of blocks and their behavior can be found in the Blockly Programming Manual.

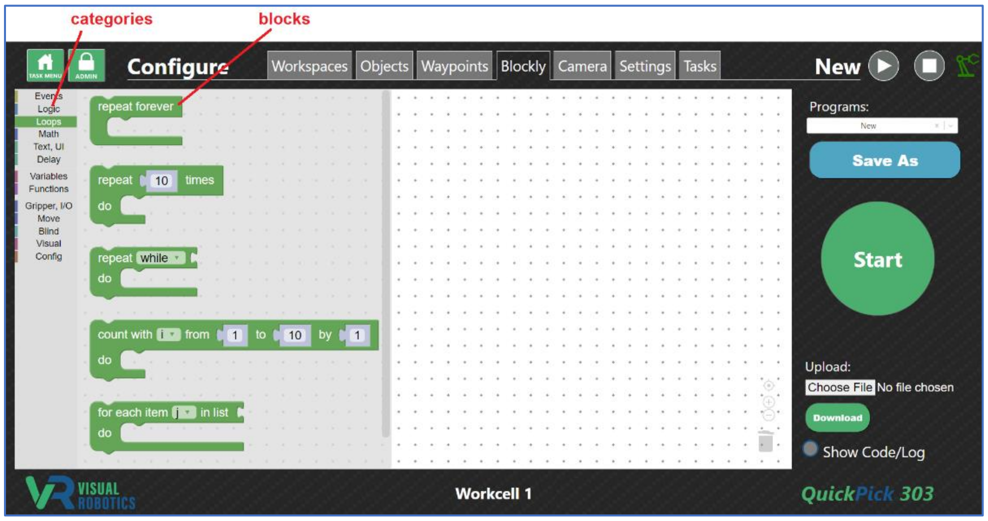


Figure 28 - Selecting the Loops category

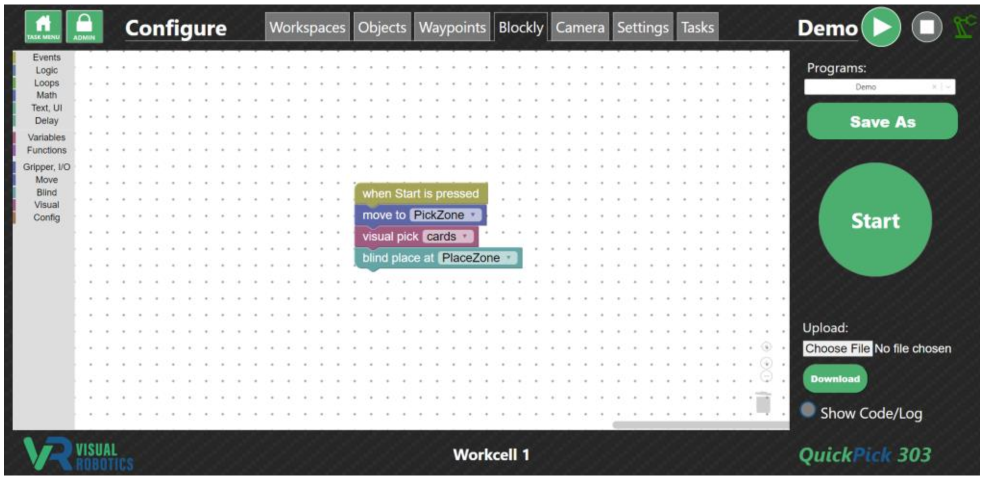


Figure 29 - Creating a program

Figure 30 shows step-by-step how to create a trivial program that repeatedly moves somewhere, to show how the Blockly editor is used. First select the **Start** block from the **Events** menu (1). Select the **Repeat** block from the **Loops** menu (2). It defaults to repeating 10 times. Choose the **Number** block from the **Math** menu and change the value to loop 5 times (3). Add a **Move** block from the **Move** menu (4). It defaults to **nowhere**. Change the waypoint in the pulldown menu to **PickZone** (5).

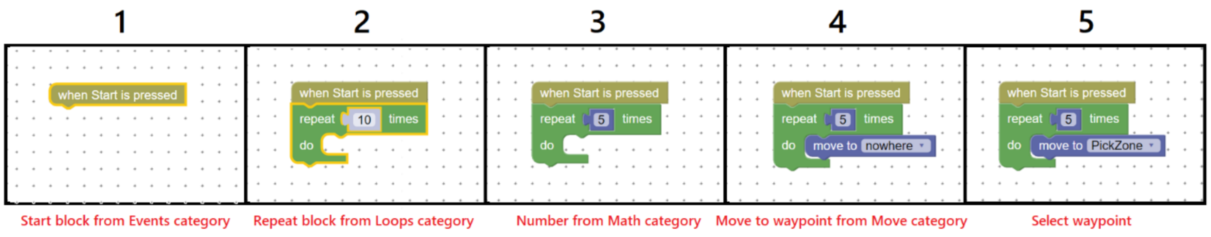
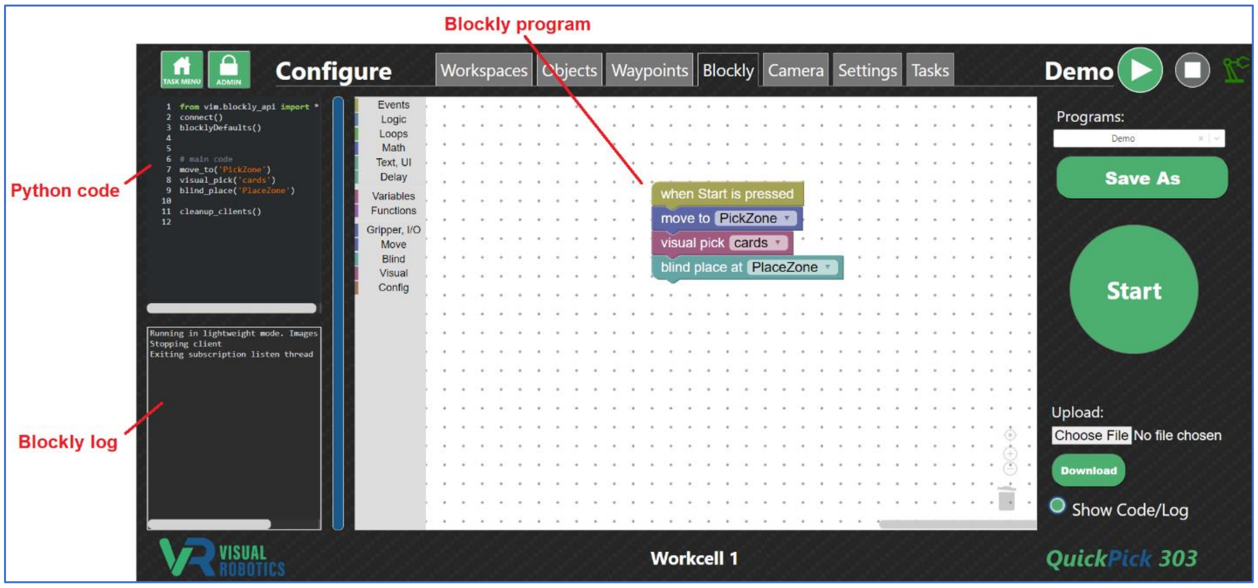

1
2
3
4
5
when Start is pressed
repeat 10 times
do
repeat 5 times
move to nowhere
move to PickZone
Start block from Events category
Repeat block from Loops category
Number from Math category
Move to waypoint from Move category
Select waypoint

Figure 30 - Creating a program - step by step

Press the **Start** button to run the program. It will change to a **Stop** button which stops the program when pressed. If pressing **Stop** doesn't stop the program, press **Stop** again, if it is executing a command that takes a while to respond.

Press the **Show Code/Log** button to see the Python code that corresponds to the Blockly program and log of print statements and code execution (Figure 31). The program consists of three commands at the beginning that import the VIM library, connect the program to the camera's execution engine, and set the default parameters for Blockly programs. What follows is the translation of the Blockly blocks into Python code. Typically each block translates into a single line of Python. The last line disconnects the program from the camera's execution engine.



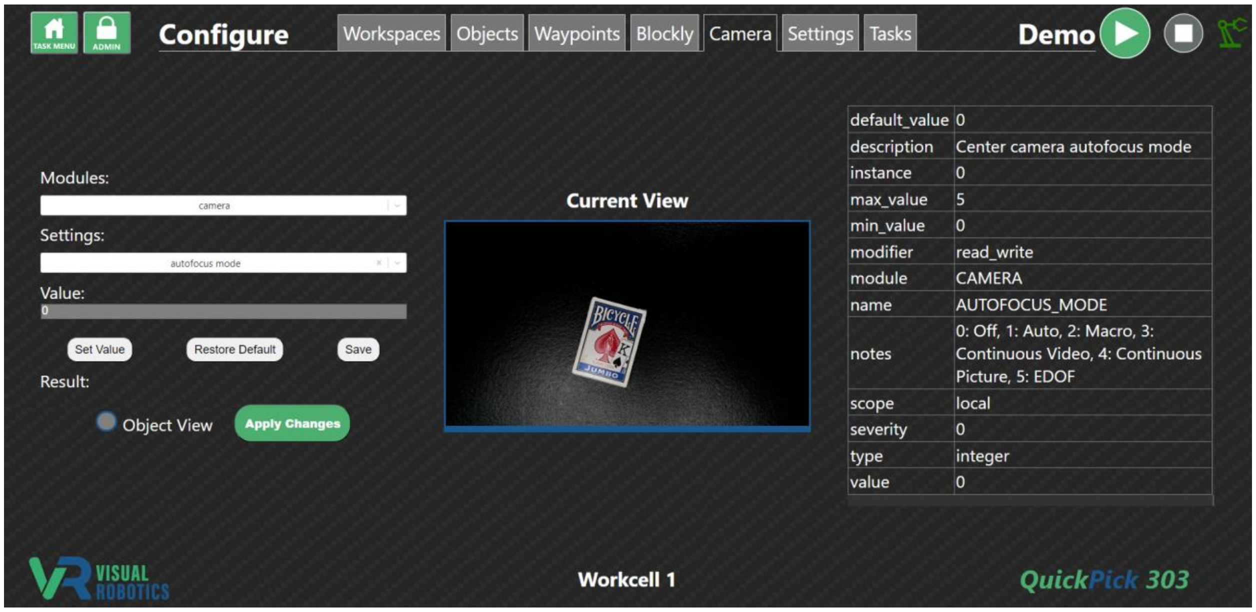
```python
from vim.blockly_api import *
connect()
blocklyDefaults()


# main code
move_to('PickZone')
visual_pick('cards')
blind_place('PlaceZone')

cleanup_clients()

```

```
Running in lightweight mode. Images
Stopping client
Exiting subscription listen thread
```

Figure 31 - Show Code to see Blockly program translated to Python code

Camera Tab

The **Camera Tab** (Figure 32) allows camera parameters, illumination settings, and debug image preferences to be adjusted. This provides a subset of the full parameter set available in the **Settings** menu but shows a live image so that the change in the settings can be immediately seen. Choose a **Module** (either **camera** or **debug image**) from the pulldown menu. Choose one of the **Settings** from the pulldown menu. Change the value and press **Set Value**. Press **Save** to save the setting so it is retained upon power cycling. The table on the right side describes the setting, its allowable values, default value, and current value.



Modules: camera

Settings: autofocus mode

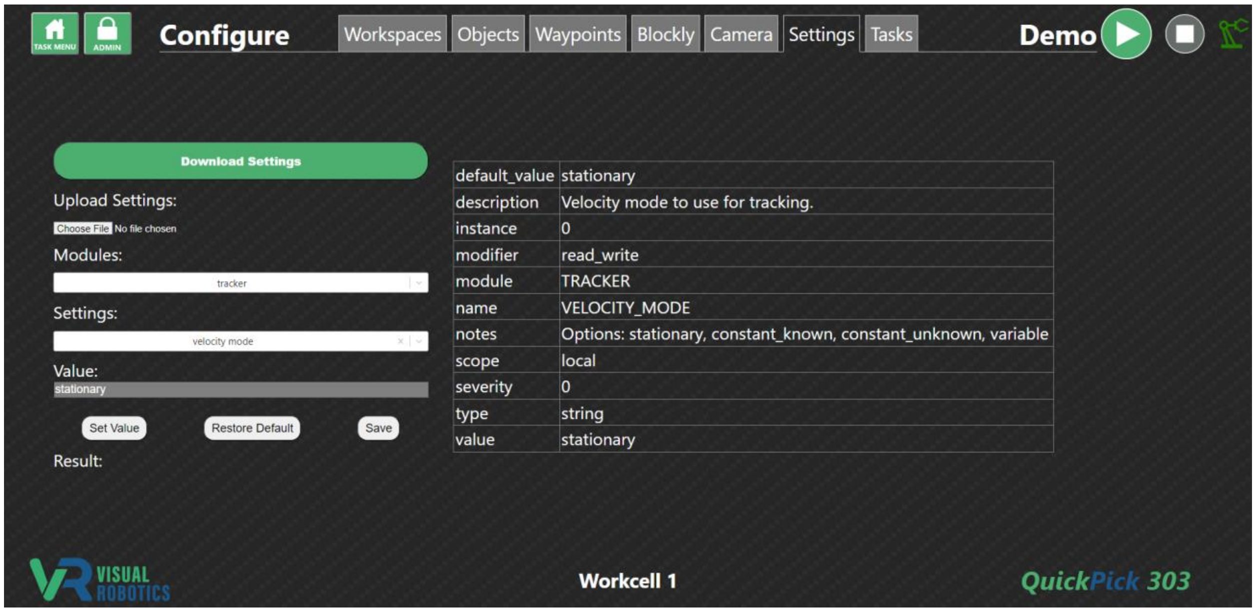
Value: 0

default_value	0
description	Center camera autofocus mode
instance	0
max_value	5
min_value	0
modifier	read_write
module	CAMERA
name	AUTOFOCUS_MODE
notes	0: Off, 1: Auto, 2: Macro, 3: Continuous Video, 4: Continuous Picture, 5: EDOF
scope	local
severity	0
type	integer
value	0

Figure 32 - Camera tab

Settings Tab

The **Settings Tab** allows the Programmer to modify any of the settings for the VIM-303 vision guidance system (Figure 33). Settings are divided into categories called **Modules**. Selecting one or more **Modules** in the pulldown menu limits the displayed settings to those categories (Figure 34). An individual **Setting** within the **module** is chosen from the pulldown menu (Figure 35). The current value is displayed in the **Value** box. The table on the right side provides details about the setting, including its current and default value. The Programmer can change the value by typing in the **Value** box. The value is not changed until the **Set Value** button is pressed. The **Default** value can be restored to the setting by pressing the **Restore Default** button. The settings are not saved after powering off the camera unless the Programmer presses the **Save** button. Description of the operation of all of the **Settings** are described in the **Settings Manual**.



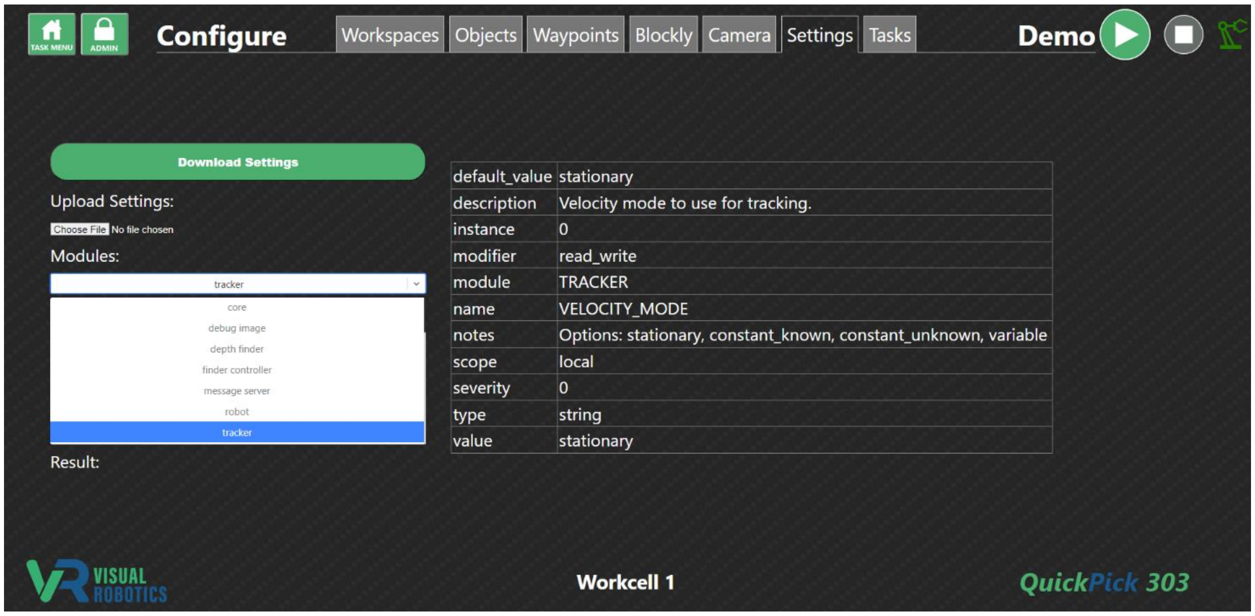
Modules: tracker

Settings: velocity mode

Value: stationary

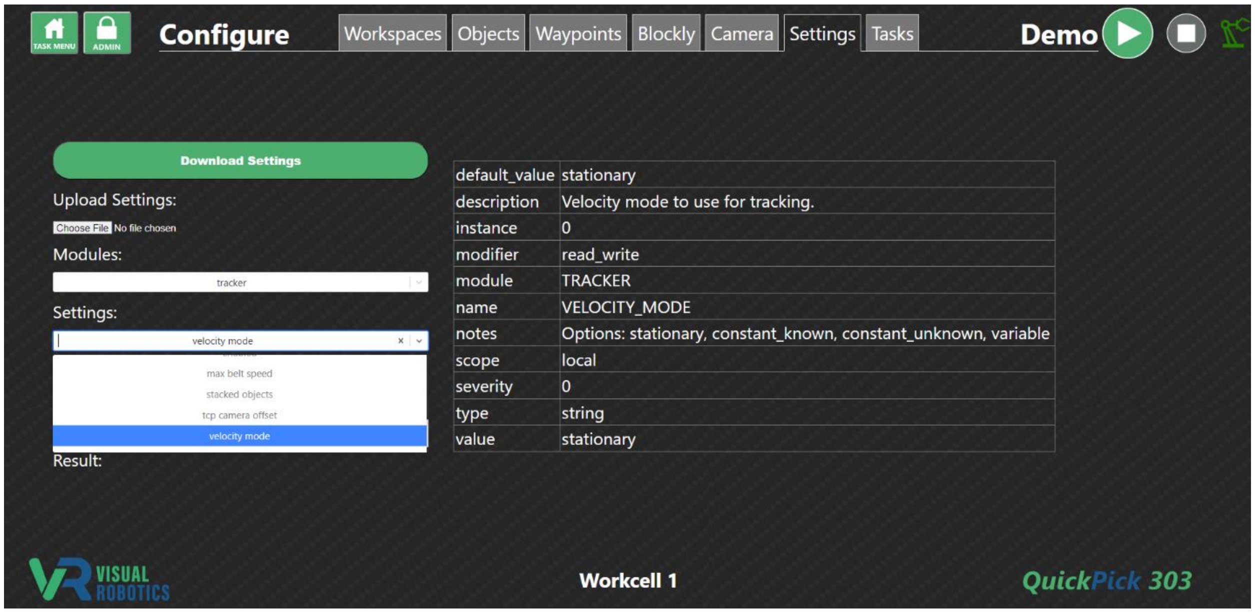
default_value	stationary
description	Velocity mode to use for tracking.
instance	0
modifier	read_write
module	TRACKER
name	VELOCITY_MODE
notes	Options: stationary, constant_known, constant_unknown, variable
scope	local
severity	0
type	string
value	stationary

Figure 33 - Settings Tab



default_value	stationary
description	Velocity mode to use for tracking.
instance	0
modifier	read_write
module	TRACKER
name	VELOCITY_MODE
notes	Options: stationary, constant_known, constant_unknown, variable
scope	local
severity	0
type	string
value	stationary

Figure 34 - Select a module



default_value	stationary
description	Velocity mode to use for tracking.
instance	0
modifier	read_write
module	TRACKER
name	VELOCITY_MODE
notes	Options: stationary, constant_known, constant_unknown, variable
scope	local
severity	0
type	string
value	stationary

Figure 35 - Selecting a setting within a module

Tasks Tab

The **Tasks Tab** allows the Programmer to organize the layout of the **Task Menu** for the Operator (Figure 36). Every program from every workspace shows up in the menu tree in the **Task Menu Layout** section of the Tasks Tab.

Folders can be created by pressing the **New Folder** button. Select and item (either a folder or task) by clicking on it and it will be outlined in white (Figure 37). Drag an item onto a folder to place it inside the folder or drag it somewhere else in the menu tree. When an item is selected, the available actions (buttons) will show in green. Folders can be renamed and deleted. Tasks cannot be renamed or deleted from this tab but are changed in the Blockly tab.

The show/hide toggle allows the Programmer to hide folders and tasks (Figure 37). Hidden tasks and folders are shown in gray (Figure 38). Hiding folders and tasks can be useful to allow the Operator to only execute selected programs. It can also be useful for version control, to show only the latest version of a task.

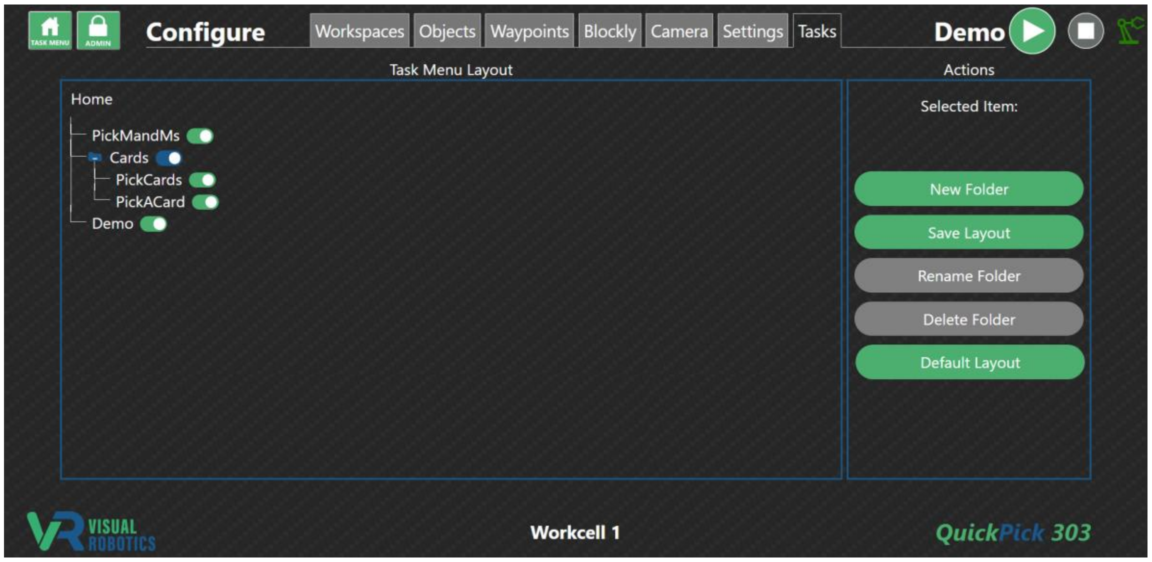


Figure 36 - Tasks tab

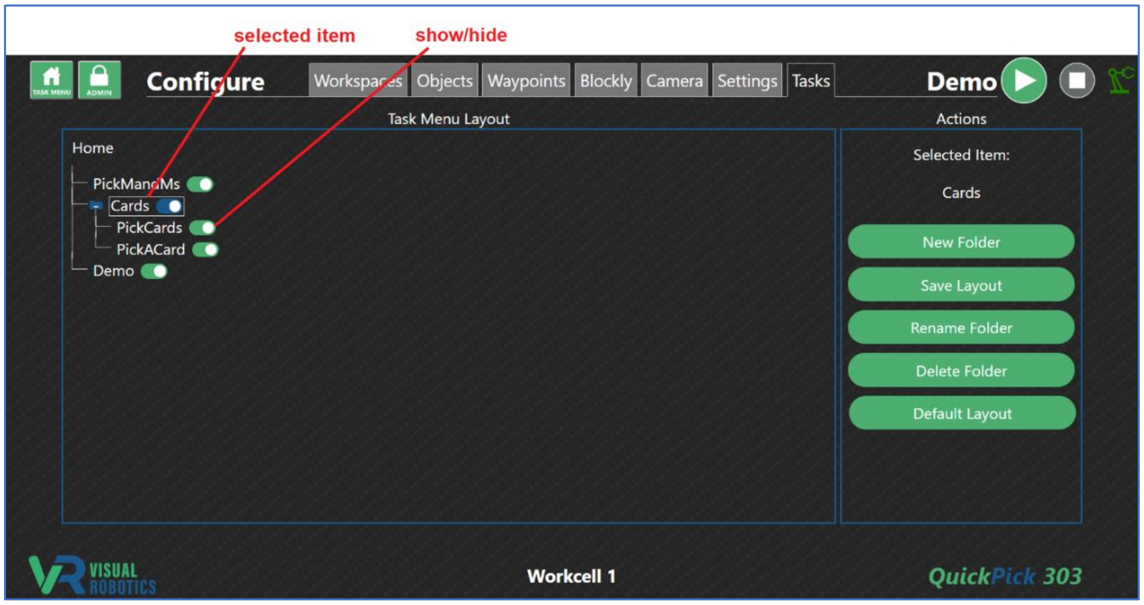


Figure 37 - Selected Item

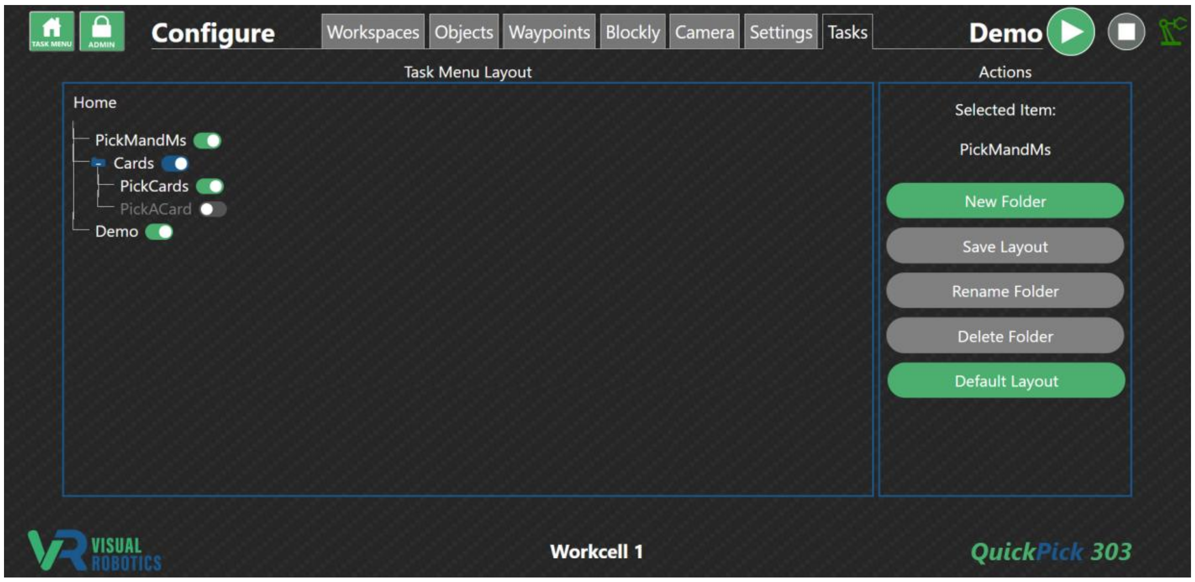


Figure 38 - Hidden task

Figure 39 shows a sample Task Menu Layout tree and the Operator's view of the Task Menu. PickMandMs is a task at the Home level of the Task Menu and a Cards folder is the other item at the Home level. If the Operator clicks into the Cards folder, two tasks are shown, PickCards and PickACard.

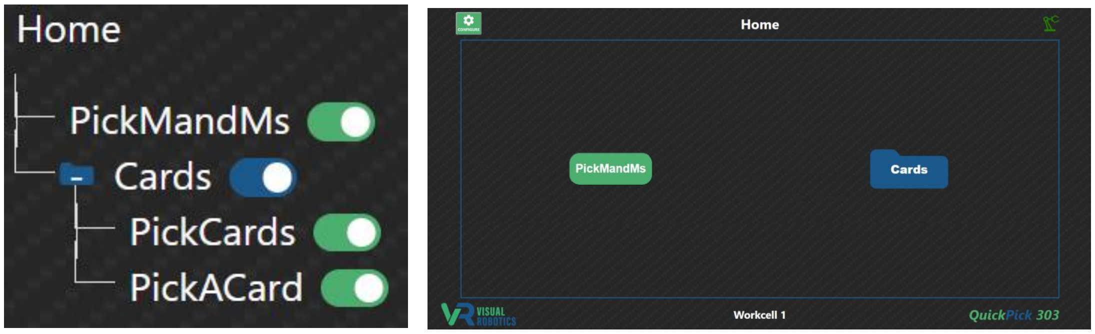
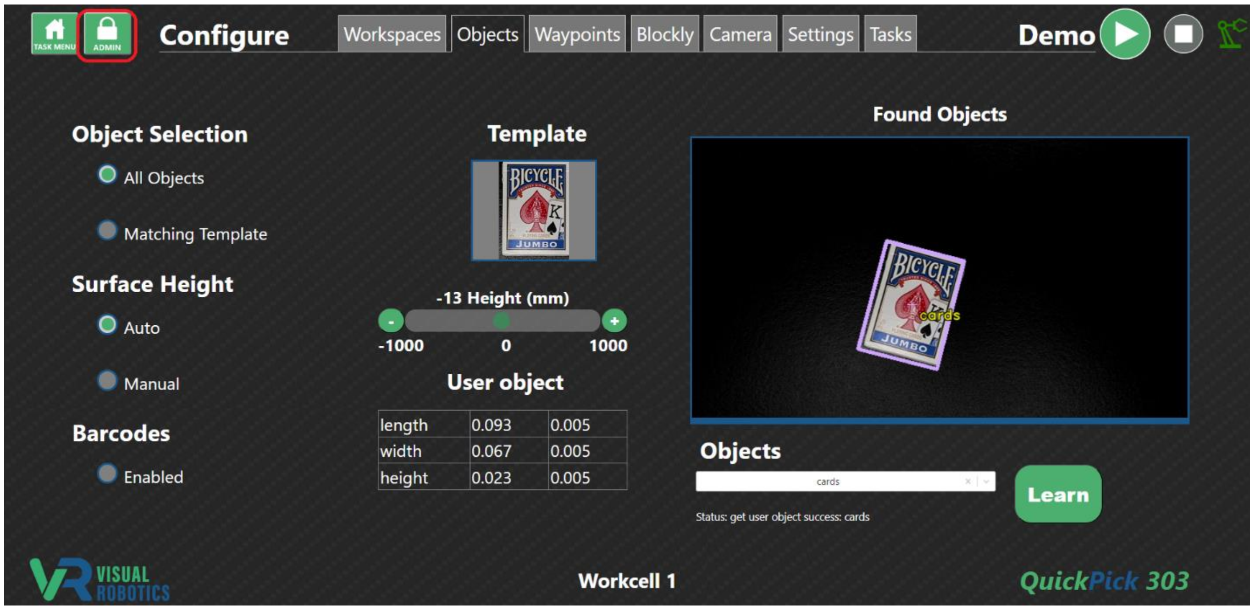


Figure 39 - Task Menu Layout tree and Operator's view of Task Menu

<u>Administration</u>

The **Administration** panel allows an Administrator to view the status of the VIM-303 camera, set up the camera's connection to a robot using PolyScope, update camera firmware and reboot the camera. The Programmer arrives at the **Administration** panel by pressing the **Admin** button (Figure 40) from any tab in the **Configure** menu.



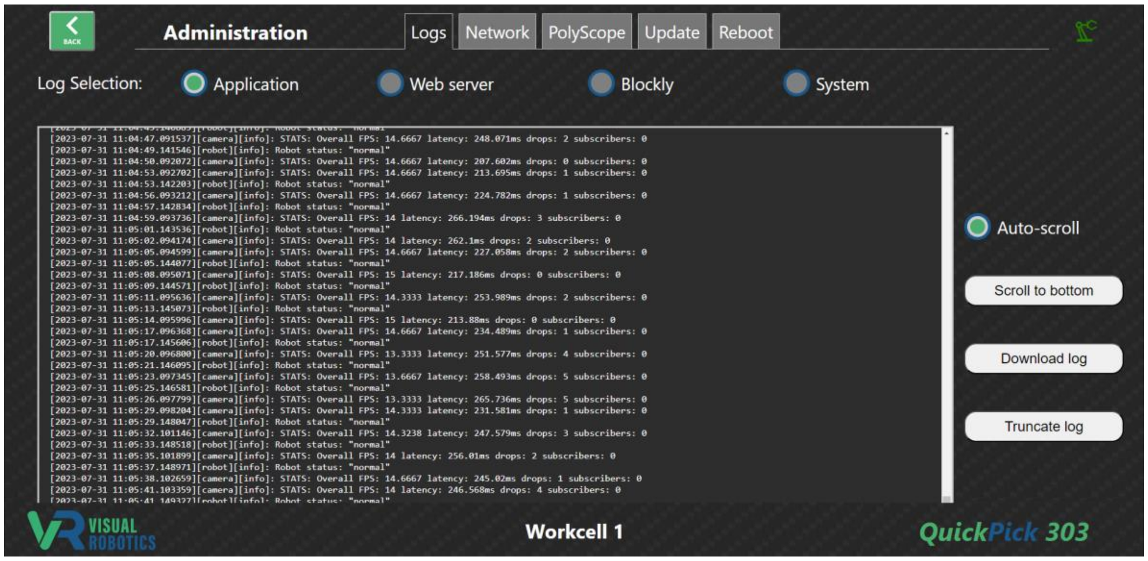
length	0.093	0.005
width	0.067	0.005
height	0.023	0.005

Status: get user object success: cards

Figure 40 - Going to the Administration panel

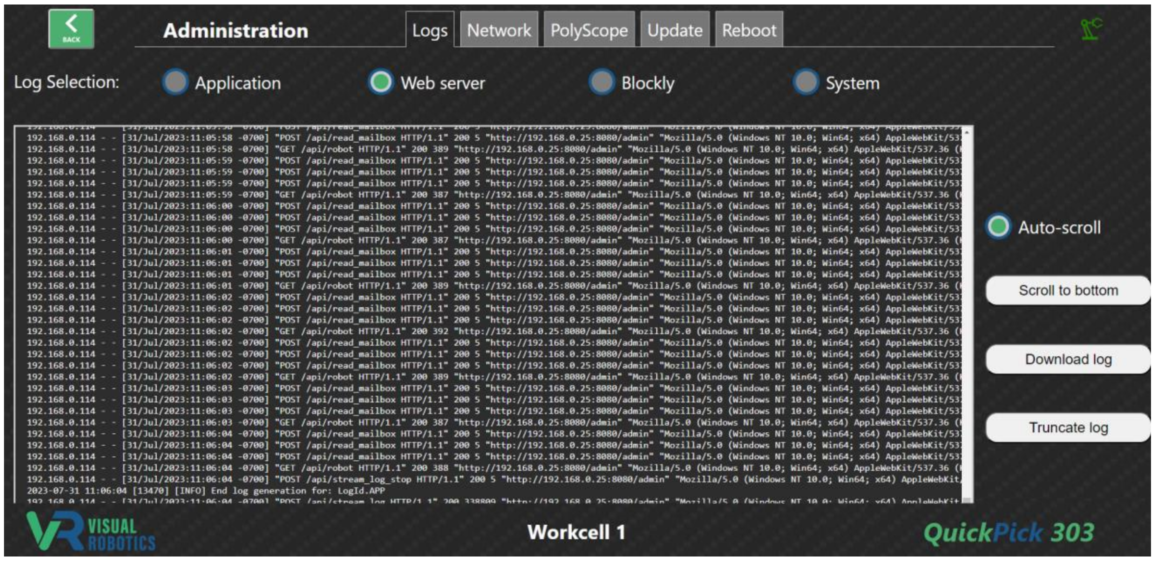
Log Tab

The **Log** tab shows various real-time logs of different parts of the system for troubleshooting purposes. The **Application Log** (Figure 41) provides debug information from the camera, finders, tracker, and robot controller components of the VIM-303 vision guidance system. The verbosity of the log level can be adjusted in the **Settings** tab in the **Core** module. The **Web Server Log** (Figure 42) shows the requests and responses to the on-board web server. The **Blockly Log** (Figure 43) shows the program state and output of **Print** blocks. It can also be viewed on the **Blockly** Tab by pressing the **Show Code/**Log button. The **System Log** (Figure 44) provides logging from the Linux operating system.



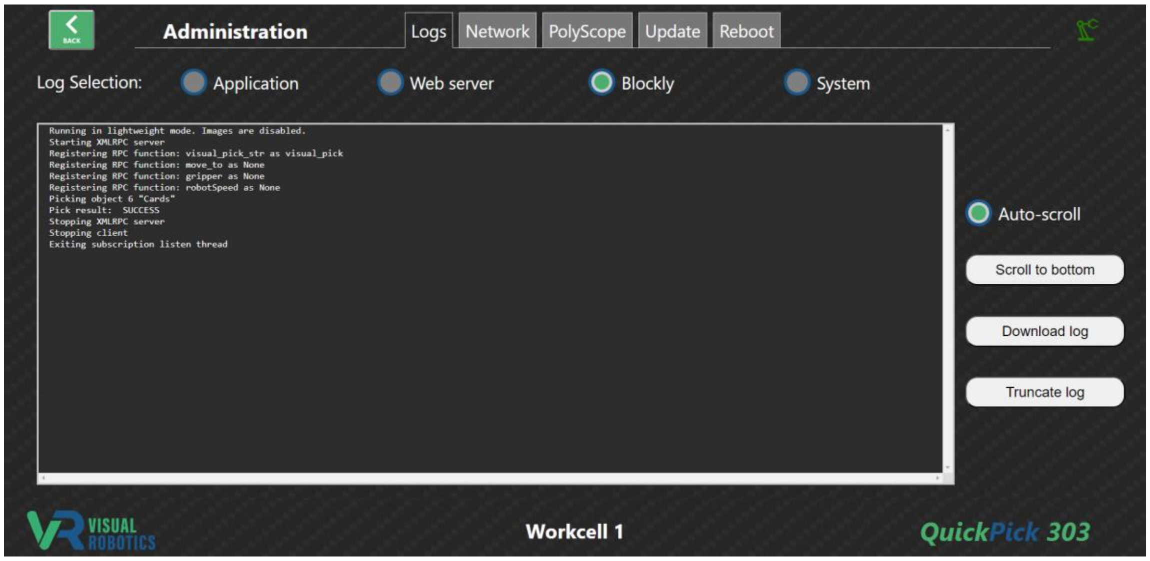
```
[2023-07-31 11:04:47.091537][camera][info]: STATS: Overall FPS: 14.6667 latency: 248.071ms drops: 2 subscribers: 0
[2023-07-31 11:04:49.141546][robot][info]: Robot status: "normal"
[2023-07-31 11:04:50.092072][camera][info]: STATS: Overall FPS: 14.6667 latency: 207.602ms drops: 0 subscribers: 0
[2023-07-31 11:04:53.092702][camera][info]: STATS: Overall FPS: 14.6667 latency: 213.695ms drops: 1 subscribers: 0
[2023-07-31 11:04:53.142203][robot][info]: Robot status: "normal"
[2023-07-31 11:04:56.093212][camera][info]: STATS: Overall FPS: 14.6667 latency: 224.782ms drops: 1 subscribers: 0
[2023-07-31 11:04:57.142834][robot][info]: Robot status: "normal"
[2023-07-31 11:04:59.093736][camera][info]: STATS: Overall FPS: 14 latency: 266.194ms drops: 3 subscribers: 0
[2023-07-31 11:05:01.143536][robot][info]: Robot status: "normal"
[2023-07-31 11:05:02.094174][camera][info]: STATS: Overall FPS: 14 latency: 262.1ms drops: 2 subscribers: 0
[2023-07-31 11:05:05.094599][camera][info]: STATS: Overall FPS: 14.6667 latency: 227.058ms drops: 2 subscribers: 0
[2023-07-31 11:05:05.144077][robot][info]: Robot status: "normal"
[2023-07-31 11:05:08.095071][camera][info]: STATS: Overall FPS: 15 latency: 217.186ms drops: 0 subscribers: 0
[2023-07-31 11:05:09.144571][robot][info]: Robot status: "normal"
[2023-07-31 11:05:11.095636][camera][info]: STATS: Overall FPS: 14.3333 latency: 253.989ms drops: 2 subscribers: 0
[2023-07-31 11:05:13.145073][robot][info]: Robot status: "normal"
[2023-07-31 11:05:14.095996][camera][info]: STATS: Overall FPS: 15 latency: 213.88ms drops: 0 subscribers: 0
[2023-07-31 11:05:17.096368][camera][info]: STATS: Overall FPS: 14.6667 latency: 234.489ms drops: 1 subscribers: 0
[2023-07-31 11:05:17.145606][robot][info]: Robot status: "normal"
[2023-07-31 11:05:20.096800][camera][info]: STATS: Overall FPS: 13.3333 latency: 251.577ms drops: 4 subscribers: 0
[2023-07-31 11:05:21.146095][robot][info]: Robot status: "normal"
[2023-07-31 11:05:23.097345][camera][info]: STATS: Overall FPS: 13.6667 latency: 258.493ms drops: 5 subscribers: 0
[2023-07-31 11:05:25.146581][robot][info]: Robot status: "normal"
[2023-07-31 11:05:26.097799][camera][info]: STATS: Overall FPS: 13.3333 latency: 265.736ms drops: 5 subscribers: 0
[2023-07-31 11:05:29.098204][camera][info]: STATS: Overall FPS: 14.3333 latency: 231.581ms drops: 1 subscribers: 0
[2023-07-31 11:05:29.148047][robot][info]: Robot status: "normal"
[2023-07-31 11:05:32.101146][camera][info]: STATS: Overall FPS: 14.3238 latency: 247.579ms drops: 3 subscribers: 0
[2023-07-31 11:05:33.148518][robot][info]: Robot status: "normal"
[2023-07-31 11:05:35.101899][camera][info]: STATS: Overall FPS: 14 latency: 256.01ms drops: 2 subscribers: 0
[2023-07-31 11:05:37.148971][robot][info]: Robot status: "normal"
[2023-07-31 11:05:38.102659][camera][info]: STATS: Overall FPS: 14.6667 latency: 245.02ms drops: 1 subscribers: 0
[2023-07-31 11:05:41.103359][camera][info]: STATS: Overall FPS: 14 latency: 246.568ms drops: 4 subscribers: 0
```

Figure 41 - Administration Log tab (showing Application log tab)



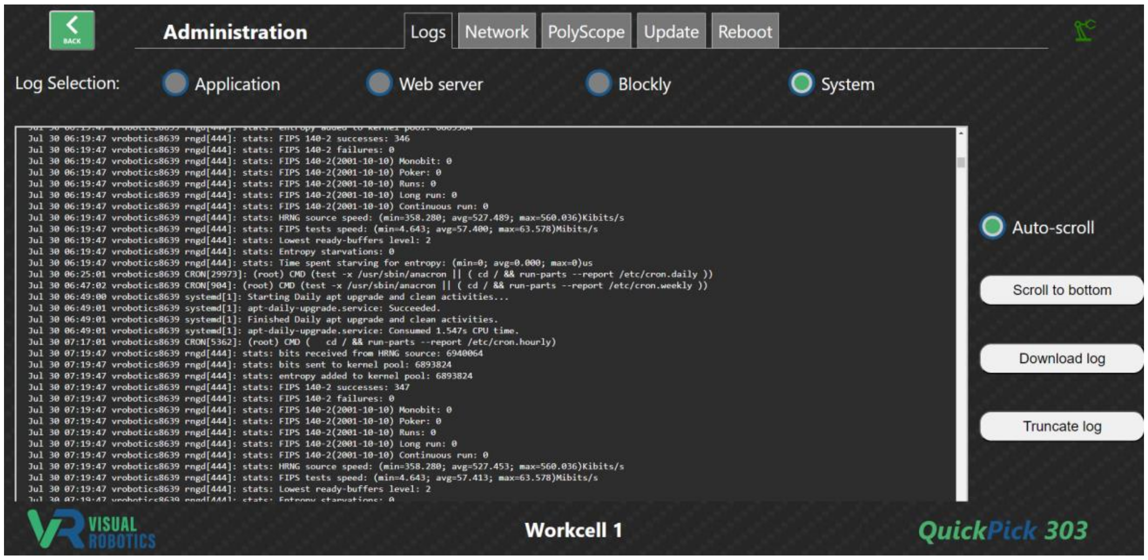
```
192.168.0.114 - - [31/Jul/2023:11:05:58 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:05:58 -0700] "GET /api/robot HTTP/1.1" 200 389 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:05:59 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:05:59 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:05:59 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:05:59 -0700] "GET /api/robot HTTP/1.1" 200 387 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:06:00 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:00 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:00 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:00 -0700] "GET /api/robot HTTP/1.1" 200 387 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:06:01 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:01 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:01 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:01 -0700] "GET /api/robot HTTP/1.1" 200 389 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "GET /api/robot HTTP/1.1" 200 392 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:02 -0700] "GET /api/robot HTTP/1.1" 200 389 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:06:03 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:03 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:03 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:03 -0700] "GET /api/robot HTTP/1.1" 200 387 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:06:04 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:04 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:04 -0700] "POST /api/read_mailbox HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
192.168.0.114 - - [31/Jul/2023:11:06:04 -0700] "GET /api/robot HTTP/1.1" 200 388 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
192.168.0.114 - - [31/Jul/2023:11:06:04 -0700] "POST /api/stream_log_stop HTTP/1.1" 200 5 "http://192.168.0.25:8080/admin" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
2023-07-31 11:06:04 [13470] [INFO] End log generation for: LogId.APP
```

Figure 42 - Web server log tab



```
Running in lightweight mode. Images are disabled.
Starting XMLRPC server
Registering RPC function: visual_pick_str as visual_pick
Registering RPC function: move_to as None
Registering RPC function: gripper as None
Registering RPC function: robotSpeed as None
Picking object 6 "Cards"
Pick result:  SUCCESS
Stopping XMLRPC server
Stopping client
Exiting subscription listen thread
```

Figure 43 - Blockly log tab

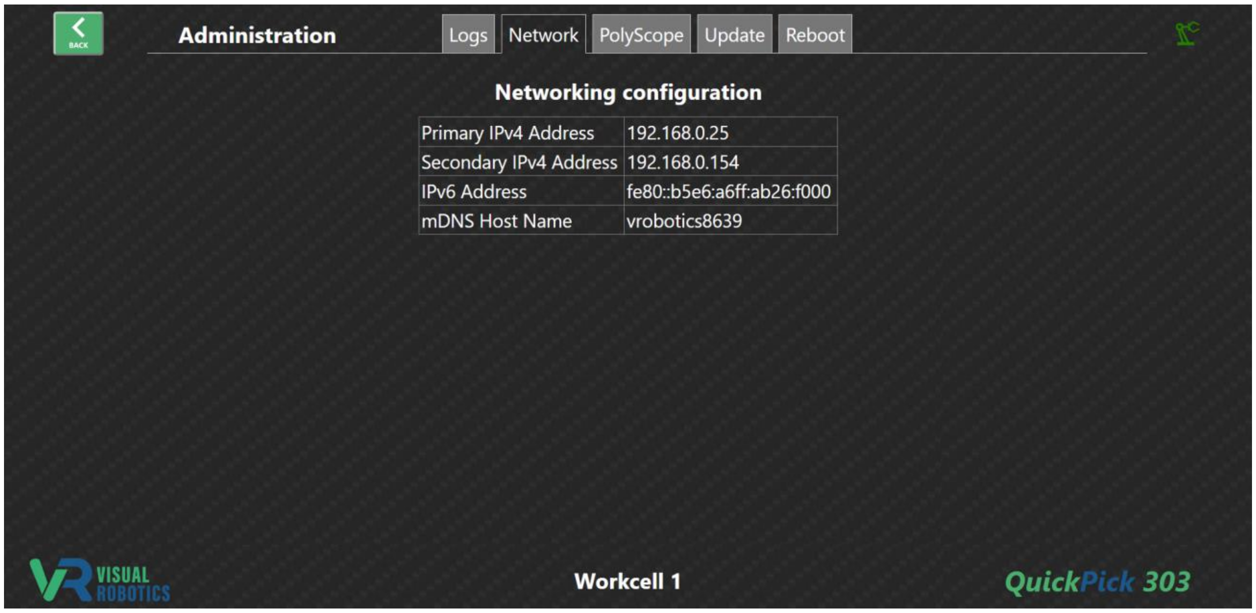


```
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2 successes: 346
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2 failures: 0
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Monobit: 0
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Poker: 0
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Runs: 0
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Long run: 0
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Continuous run: 0
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: HRNG source speed: (min=358.280; avg=527.489; max=560.036)Kibits/s
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: FIPS tests speed: (min=4.643; avg=57.400; max=63.578)Mibits/s
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: Lowest ready-buffers level: 2
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: Entropy starvations: 0
Jul 30 06:19:47 vrobotics8639 rngd[444]: stats: Time spent starving for entropy: (min=0; avg=0.000; max=0)us
Jul 30 06:25:01 vrobotics8639 CRON[29973]: (root) CMD (test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily ))
Jul 30 06:47:02 vrobotics8639 CRON[904]: (root) CMD (test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly ))
Jul 30 06:49:00 vrobotics8639 systemd[1]: Starting Daily apt upgrade and clean activities...
Jul 30 06:49:01 vrobotics8639 systemd[1]: apt-daily-upgrade.service: Succeeded.
Jul 30 06:49:01 vrobotics8639 systemd[1]: Finished Daily apt upgrade and clean activities.
Jul 30 06:49:01 vrobotics8639 systemd[1]: apt-daily-upgrade.service: Consumed 1.547s CPU time.
Jul 30 07:17:01 vrobotics8639 CRON[5362]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: bits received from HRNG source: 6940064
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: bits sent to kernel pool: 6893824
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: entropy added to kernel pool: 6893824
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2 successes: 347
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2 failures: 0
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Monobit: 0
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Poker: 0
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Runs: 0
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Long run: 0
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS 140-2(2001-10-10) Continuous run: 0
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: HRNG source speed: (min=358.280; avg=527.453; max=560.036)Kibits/s
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: FIPS tests speed: (min=4.643; avg=57.413; max=63.578)Mibits/s
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: Lowest ready-buffers level: 2
Jul 30 07:19:47 vrobotics8639 rngd[444]: stats: Entropy starvations: 0
```

Figure 44 - System log tab

Network Tab

The **Network Tab** (Figure 45) shows the IP addresses and network name of the VIM-303 camera. Figure 46 shows what network addresses the camera will use, depending on the existence of a DHCP server and the value of the IP Address Setting (in the Robot module in the Settings tab). Under some circumstances, there will only be a single IP address, so only a Primary IPv4 address will be shown on the Network tab. In other circumstances, the camera will have two IP addresses, the static address and the DHCP address. The mDNS Host Name matches what is printed on the back of the camera. The camera can be located using mDNS using the suffix **.local** as in **vrobotics8639.local** in the example of Figure 45.



Networking configuration

Primary IPv4 Address	192.168.0.25
Secondary IPv4 Address	192.168.0.154
IPv6 Address	fe80::b5e6:a6ff:ab26:f000
mDNS Host Name	vrobotics8639

Figure 45 - Network tab

IP Address Setting	DHCP Server?	DHCP Subnet	Static Address	DHCP Address
auto	Yes	192.168.1.xx	disabled	Yes
auto	Yes	not 192.168.1.xx	disabled	Yes
auto	No	N/A	192.168.1.47	No
static address	Yes	same as address	static address	Yes
static address	Yes	different from address	disabled	Yes
static address	No	N/A	statis address	No

Figure 46 - Network address table

PolyScope Tab

The **PolyScope Tab** has a single button **Save Command Proecssor** (Figure 47). When pressed, this button copies the VIM-303 **command processor** to the Universal Robot at the IP address specified in **Robot IP** in the **Robot** module of the settings tab. The **command processor** controls the robot to implement visual picking and placement. The button uses **ssh** to copy the command processor using the robot's default username and password. This method is used by the VIM-303 camera in contrast to requiring a URCap to be loaded onto the robot. In addition to the **command processor**, a sample PolyScope program is copied to the robot that shows how the VIM-303 can be commanded from PolyScope to implement visual picking.

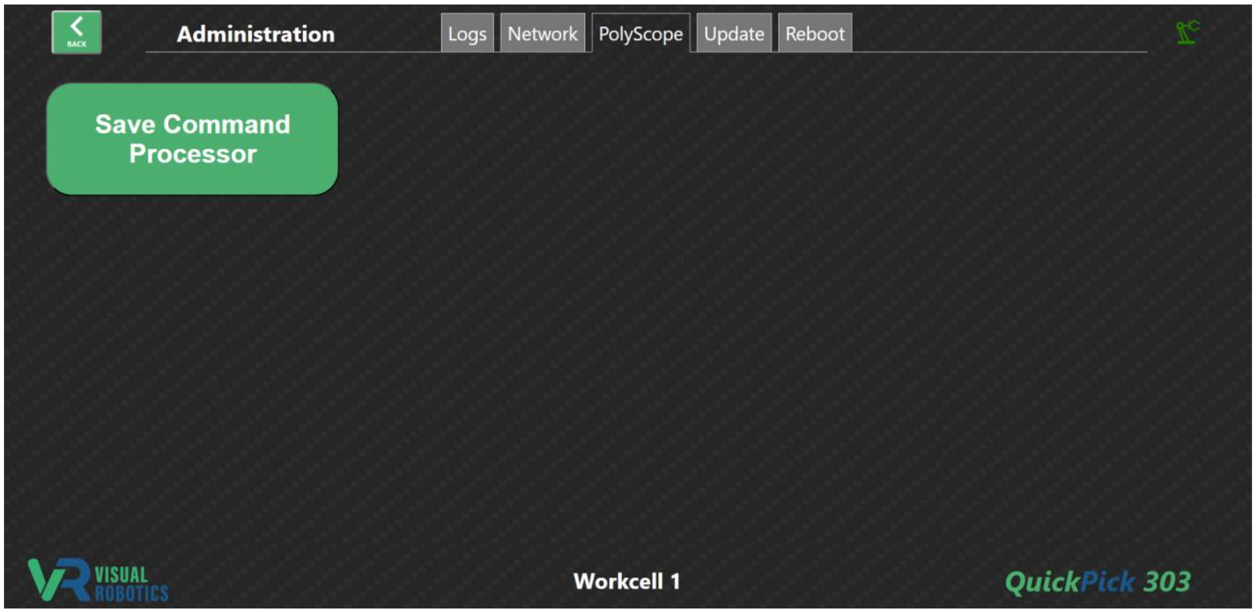
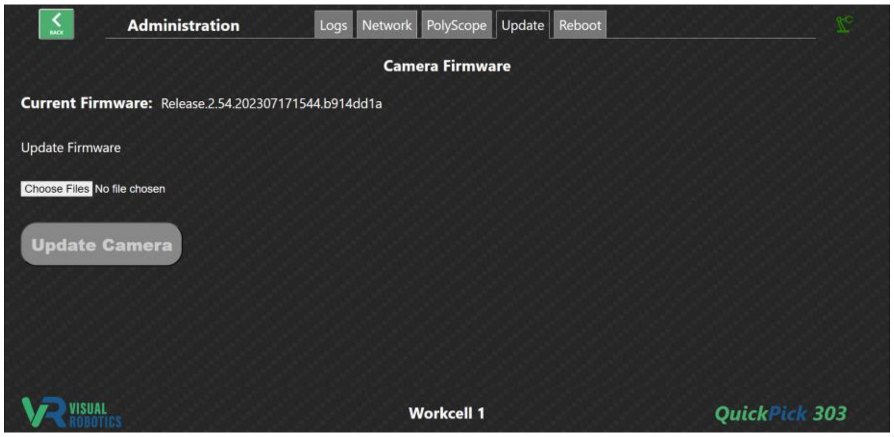


Figure 47 - PolyScope tab

Update Tab

The **Update Tab** is used to update firmware on the VIM-303 camera (Figure 48). The current firmware version is listed at the top of the screen. Press the **Choose Files** button to select a file from your PC to send to the camera (Figure 49). The **Update Camera** button will turn green (Figure 50). Press it to begin the firmware update. The camera will restart (you may need to refresh the page) and the Current Firmware will indicate the recently updated firmware version.



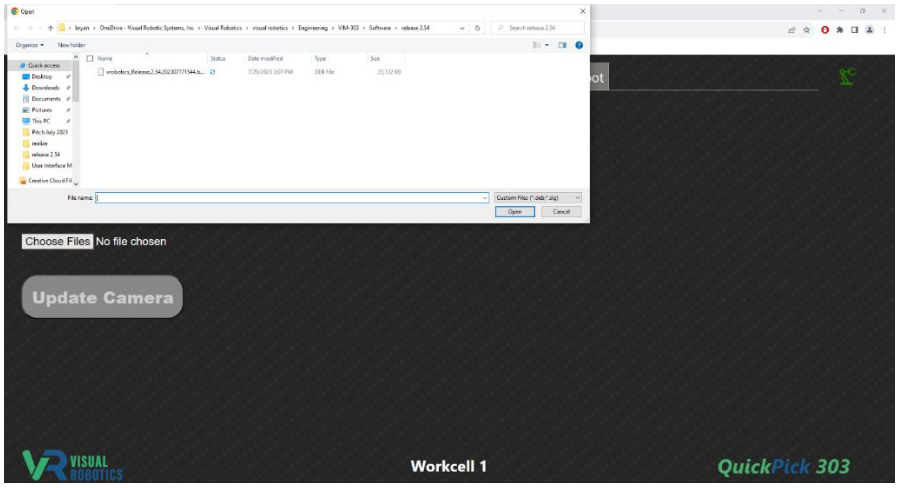
Camera Firmware

Current Firmware: Release.2.54.202307171544.b914dd1a

Update Firmware

Choose Files No file chosen

Figure 48 - Update firmware tab



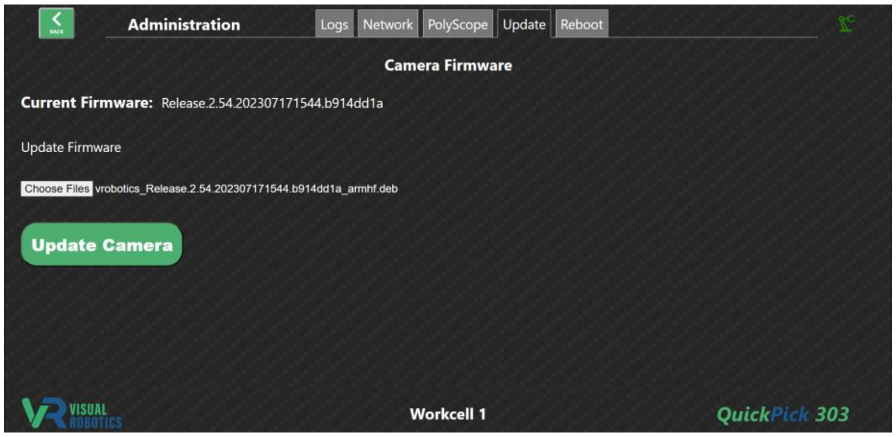
Name	Status	Date modified	Type	Size
vrobotics_Release.2.54.202307171544.b...		7/29/2023 2:07 PM	DEB File	23,132 KB

File name:

Custom Files (*.deb;*.sig)

Choose Files No file chosen

Figure 49 - Select firmware file from PC



Camera Firmware

Current Firmware: Release.2.54.202307171544.b914dd1a

Update Firmware

Choose Files vrobotics_Release.2.54.202307171544.b914dd1a_armhf.deb

Figure 50 - Update Camera button to upgrade firmware

Reboot Tab

The **Reboot Tab** is used to reboot the camera (Figure 51). If the camera is unresponsive or under the direction of a Visual Robotics technician, rebooting the camera may be required to restore proper performance. Press the **Reboot Camera** button and after a while (perhaps a minute) the camera will be restarted. You may need to refresh the webpage.

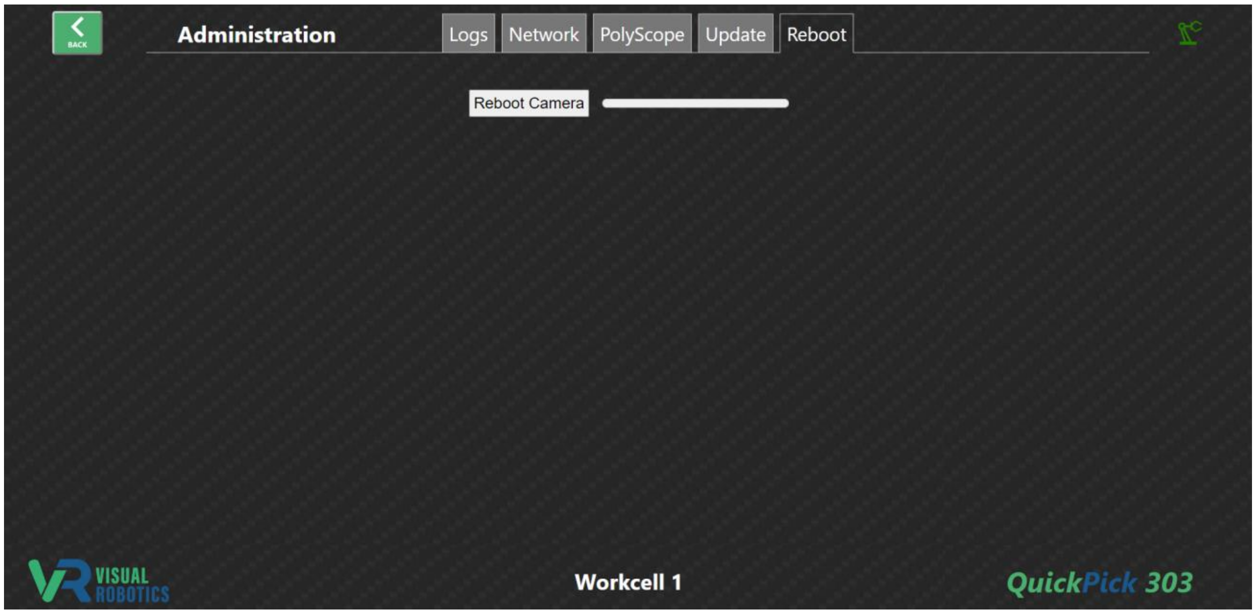


Figure 51 - Reboot tab