Using Teledyne Sherlock

with LEonard





Software Version 22.11.1.0

**LEonard Software by Lecky Engineering, LLC**

|  |  |  |
| --- | --- | --- |
| **Document Version** | **Date** | **Major Additions** |
| 21.11.4.0 | 11/04/2021 | Initial user interface and device management system, Java interpreter |
| 22.04.1.0 | 04/01/2022 | Universal Robot interface and grinding system, LEScript support |
| 22.08.1.0 | 08/15/2022 | LMI Gocator interface and demonstration |
| 22.11.1.0 | 11/14/2022 | Python support, screen sizing and display management |

**CONTENTS**

[Overview 4](#_Toc119225497)

[Connection 4](#_Toc119225498)

[The LEonard Interface 5](#_Toc119225499)

[Using the Accelerator 6](#_Toc119225500)

[Your Gocator Job 7](#_Toc119225501)

[Using the Results in LEonard 10](#_Toc119225502)

# 

# Overview

LEonard provides a custom interface for the LMI Gocator product line.

The interface is currently in use with and heavily tested with the Gocator 3200 Series Snapshot cameras.

For more information on these powerful systems direct from [LMI Technologies](https://lmi3d.com/), see [lmi3d.com/series/gocator-3210/](https://lmi3d.com/series/gocator-3210/).

# Basic Ethernet Connection

The Gocator and the computer running LEonard must have an Ethernet interface capable of communication. At Lecky Engineering, our test machine is on 192.168.0.252/24 and our Gocator is on 192.168.0.3/24.

You should be able to browse directly to the Gocator to verify communication.

Graphical user interface, text, application

Description automatically generated

# The LEonard Interface

To communicate with the Gocator, the **Devices** list in LEonard needs an entry for the Gocator. The default devices include two properly setup entries for a Gocator- you will just need to verify and adjust your IP address appropriately.

Graphical user interface, application

Description automatically generated

Figure Device Entries for LMI Gocator

It is important to use the gocator callback as well as the displayed TxSuffix and RxTerminator.

The Gocator job that you wish to be loaded can be included in the Jobfile field of the device entry.

Connection is initiated by selecting the desired row and pressing **Connect**. In addition, if you have selected **Auto Connect On Load** for your device file, the connection will be started automatically when LEonard starts.

LEonard always starts a Gocator-style connection with a set of commands:

stop

clearalignment

loadjob <jobFile>

start

Upon successful connection, the Connected field should check itself and the Gocator Status annunciators should appear on the Run tab.

Diagram

Description automatically generated with low confidence

Figure Gocator Status Annunciators

## Using the Accelerator

It is common to use PC-based acceleration with the Gocator to accelerate processing speed.

The Accelerator software is downloaded from the LMI website. Ours is at:

"C:\Users\nedlecky\Desktop\Gocator 3210\14405-6.1.32.12\_SOFTWARE\_GO\_Utilities\Emulator and Accelerator\bin\win64\GoAccelerator.exe"

When this software is running, and started, you will see the dialog below. Sometimes it takes a few attempts to get the connection initiated.

Graphical user interface, application

Description automatically generated

Figure LMI Gocator Accelerator in Operation

For native connection to the Gocator, we would use 192.168.0.3:8190 to connect straight to the Gocator. To use the accelerators, this becomes localIP:8190. In our case, that would be 192.168.0.252:8190.

To browse to the Gocator for monitoring and programming while using the accelerator, just browse to localhost:8080

# Your Gocator Job

The LEonard interface to Gocator is simple. The gocator\_trigger(int pre\_delay\_ms) command sends the “trigger” command to the Gocator which runs whatever job you have loaded.

When the trigger is sent, LEonard clears a variable called gocator\_ready.

Your Gocator job must send gocator\_ready=True as its final output. This is how LEonard knows that processing is complete!

The example Gocator job LEonardRoot/3rdParty/Gocator is called LeonardHolefinder.job.

It measures hole sizes, angles, and locations for both countersunk and thru holes.

It does this using the Surface Countersunk Hole, Surface Hole, and Surface Plane tools as shown below.

Your job could use any Gocator tools!

Graphical user interface

Description automatically generated

Figure LEonardHolefinder.job Tools

After running your inspection, you need to send results back to LEonard, followed by that all-important gocator\_ready=True.

Here’s how LEonardHolefinder.job does it:

A screenshot of a computer

Description automatically generated

Figure Output from the Gocator job to LEonard

The detail on the data format field is as follows:

gc\_decision=%decision[100]

gh\_decision=%decision[200]

gc\_offset\_x=%value[100]

gc\_offset\_y=%value[101]

gc\_offset\_z=%value[102]

gc\_outer\_radius=%value[103]

gc\_depth=%value[104]

gc\_bevel\_radius=%value[105]

gc\_bevel\_angle=%value[106]

gc\_xangle=%value[107]

gc\_yangle=%value[108]

gc\_cb\_depth=%value[109]

gc\_axis\_tilt=%value[110]

gc\_axis\_orient=%value[111]

gh\_offset\_x=%value[200]

gh\_offset\_y=%value[201]

gh\_offset\_z=%value[202]

gh\_radius=%value[203]

gp\_xangle=%value[300]

gp\_yangle=%value[301]

gp\_z\_offset=%value[302]

gp\_std\_dev=%value[303]

gocator\_ready=True

DON’T FORGET THAT LAST LINE!

LEonard will assume the Gocator is still crunching data until it receives that. That will also trigger the Gocator Ready annunciator to go back to green.

That’s it! You can send as many results back as you like, and they will be received and remembered by LEonard.

These variables are stored in the LEonard LEScript variable set as well as the variable lists in Java and Python.

The variable assignment statements are passed to Java and Python exactly as received, so the automatic typing rules implemented in those languages will automatically happen. The received values will be integers or floats as assumed in Python, for example.

# Using the Results in LEonard

LEonard includes a hard-coded gocator\_adjust function that automatically moves a UR robot to try to drive offsets and angles to 0. This code makes assumptions about orientation, scale, and offset that might not be what you need, however.

Look at the example Python alignment, which is entirely user-modifiable, in LEonardRoot/Code/Lib/leGocatorSupport.py to see how custom functions can be used to feed robot motion commands back as a result of measurements.

Look at the examples in LEonardRoot/Code/Examples/Gocator to see how these functions are used.

## Example Walk-Through

For example, LEonardRoot/Code/Examples/Gocator/00p can be used

TBD