

LabRecon - Code Link with other programming environments

LabRecon software (Windows or Mac OS X) and hardware provides a stimulating educational environment for science and engineering by allowing students and educators to easily create rich graphical interfaces for measurements, robotic control, or simulation.

This document presents multiple examples using the “Code Link” feature, wherein LabRecon can serve as an interface for other programming languages.

Interfaces with the following languages:

- Java
- Python
- C/C++
- Javascript
- Visual Basic

Car turning dynamics with Python code

Below is an example, written in Python, for a physics simulation of a car turning radius accounting for friction between the tires and the road.

The LabRecon Panel allows Python variables to be controlled with buttons, a knob, and scrolls and output variables to be displayed and plotted on an XY Chart.

Variables are shared with LabRecon.

```

IDLE File Edit Format Run Options Window Help
CodeLink.py - /Users/Recon/Desktop/LabRecon/Projects/CodeL

X = 0 #initializing variables to be used in while loop
Xval = 0.0
Y1 = 1
Y1val = 0.0
Y2 = 2
Y2val = 0.0
rad = 0.0
v = 0.0
mu = 0.0
direction = 1.0
angle = 0.0
notif = 0.0
delay = .01

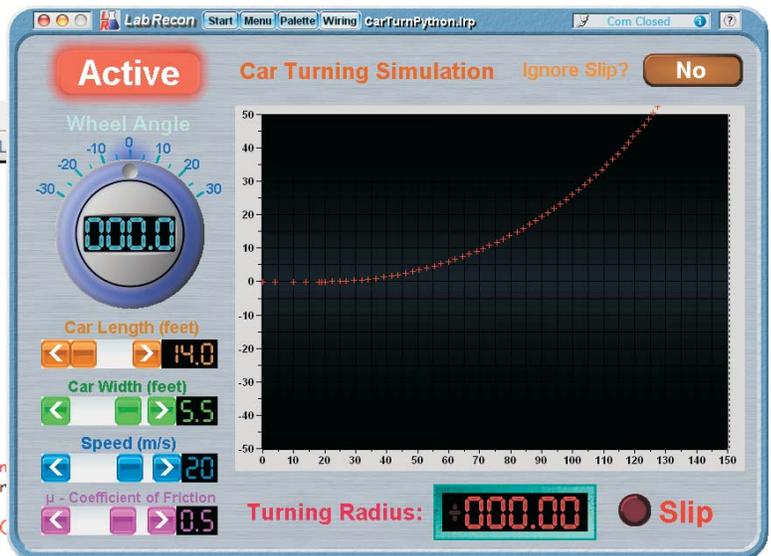
def radius(angle, wheelbase, track): #returns turning radius based on
    return (wheelbase/math.sin(math.radians(angle)))+(track/2)

def maxRadius(v,mu): #returns maximum turning radius based on speed
    return (v**2)/(mu*32)

LR = LRVars() #create instance of LabRecon Code Link class
LR.Connect(40000) # Port LabRecon is listening on
print ('Connected')
while True:
    while(LR.GetVar(9)==1): #checks whether continue button is activated
        LR.SetVar(5, 0) #deactivate reset object
        direction = LR.GetVar(0) #setting values in the code that are received from LabRecon
        angle = abs(direction)
        wheelbase = LR.GetVar(1)
        track = LR.GetVar(2)
        v = LR.GetVar(3)
        mu = LR.GetVar(4)
        allowSlip = LR.GetVar(5)

        Y1val = 0 #initializing values to be used later
        notif = 0

        if(angle>0):
            rad = radius(angle, wheelbase, track) #setting turning radius
            if(rad<maxRadius(v, mu) and allowSlip==0): #checking to see if max radius should
                rad = maxRadius(v, mu)
            notif = 1
    
```

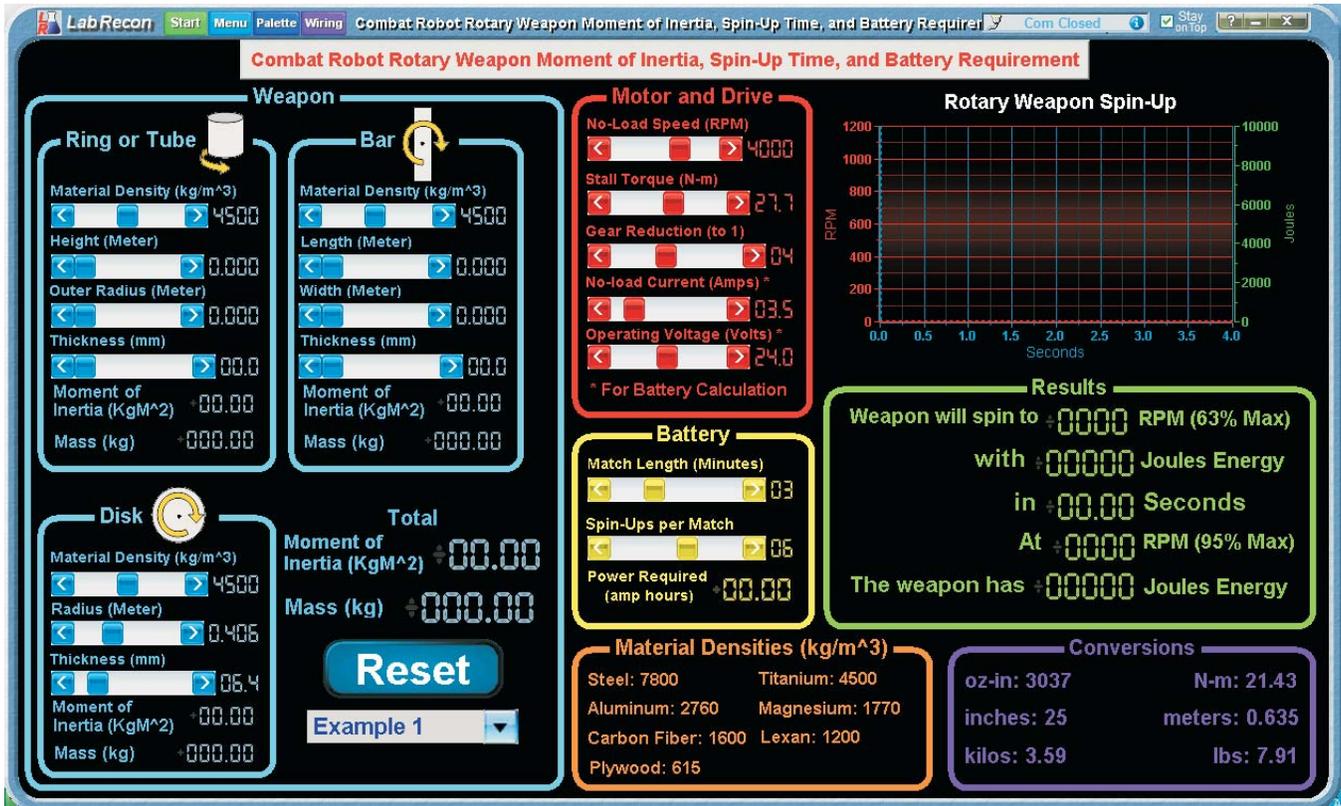


Battle-Bot weapon dynamics with Java code

Up to 40 variables can be shared between LabRecon and the programming environment.

Java code has been written to model the dynamics of a rotating weapon of a BattleBot.

A panel was created to allow many input parameters to be controlled by scrolls with performance results displayed and charted.

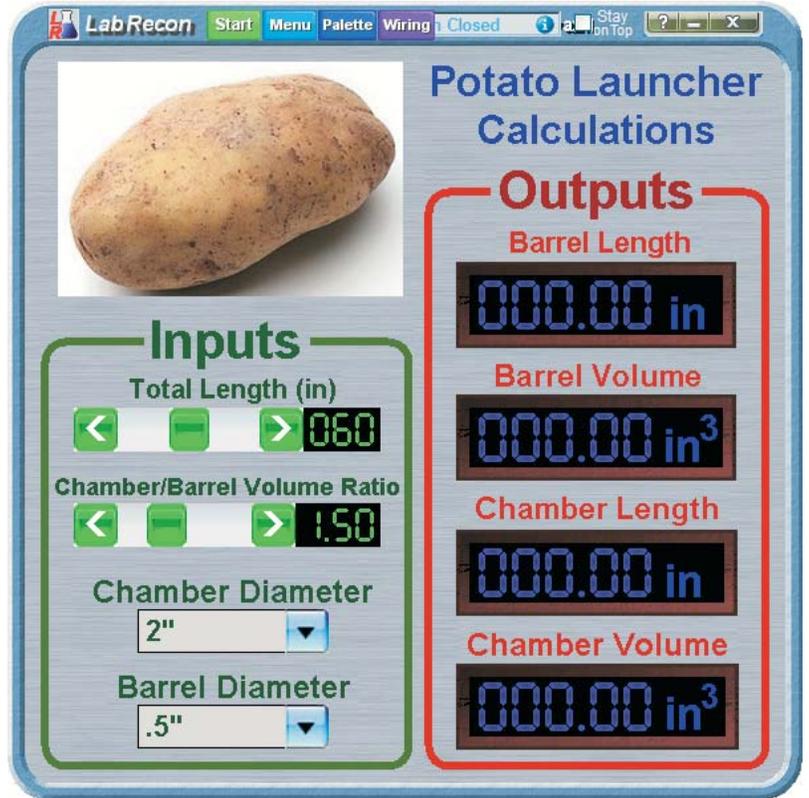


```

Java - LabRecon/src/CombatRobotCalc.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Expl...
DiningComputerScientists
LabRecon
Scientist.java Dining.java Chopstick.java CombatRobotCalc.java
256 BarMass = BarLength * BarWidth * BarThickness * (float)0.001 * BarDensity;
257 BarMOI = (BarLength * BarLength * BarWidth * BarWidth * BarMass / ((float)12);
258 DiskMass = ((float)3.14/1000) * ((float)Math.pow(DiskRadius, 2)) * DiskThickness * DiskDensity;
259 DiskMOI = (float)0.5 * DiskMass * (float)Math.pow(DiskRadius, 2);
260 RTMass = ((float)3.14) * RTDensity * RTHeight * ((float)Math.pow(RTRadius, 2) - (float)Math.pow((RTRadius - (RTThick
261 RTMOI = ((float)0.5) * RTMass * ((float)Math.pow(RTRadius, 2) + (float)Math.pow((RTRadius - (RTThick
262 MOI = BarMOI + DiskMOI + RTMOI;
263 MassTot = BarMass + DiskMass + RTMass;
264
265 RPM63 = RPM(NoLoadSpeed, GearRed, 0.63);
266 Joules63 = Joules(MOI, RPM63);
267 Seconds = (((float)0.105) * NoLoadSpeed / GearRed) / (StallTorque * GearRed * MOI;
268 RPM95 = RPM(NoLoadSpeed, GearRed, 0.95);
269 Joules95 = Joules(MOI, RPM95);
270 PowerReq = SpinUps * (Joules95 / (1800 * OpVoltage)) + (NoLoadCurrent * MatchLength) / 60;
271 MaxRPM = RPM(NoLoadSpeed, GearRed, 1);
272
273
274
275
276 LR.SetVar(1, 9);
277
278 for(float i = 0; i < 1; i += .05) {
279     x = -Seconds * ((float)Math.log(1-i));
280     y1 = MaxRPM * i;
281     y2 = ((float)0.5) * MOI * ((float)Math.pow((y1 * (float).105), 2));
282     LR.SetVar(x, 6);
283     LR.SetVar(y1, 7);
284     LR.SetVar(y2, 8);
285     System.out.println("X:" + String.valueOf(x));
286     System.out.println("Y1:" + String.valueOf(y1));
287 }
288
Task List
Quick Access
Connect Mylyn
Outline
RTMass : float
RTMOI : float
MaxRPM : float
x : float
y1 : float
y2 : float
LR : LRCodeLink
RPM(float, float, double) : float
Joules(float, float) : float
main(String[]) : void
Problems Javadoc Declaration Console
CombatRobotCalc [Java Application] C:\Program Files\Java\jre1.8.0_71\bin\javaw.exe (Mar 12, 2016, 11:23:38 AM)
Y1:950.0002
Y2:6112.1245
Writable Smart Insert 263 : 51
    
```

Potato launcher calculations with Java code

This example shows Java code with a panel created to perform calculations for a potato launcher fabricated from PVC piping.



```

class PotatoLauncher {
    //variables from LabRecon "To Code (IP)" Wiring object
    static float tLength = 0; // Total length
    static float cbRatio = 0; // Chamber:Barrel volume ratio
    static float knobDiameter = 0; // Nominal barrel diameter
    static float knobDiameter = 0; // Nominal chamber diameter
    static float bRadius = 0; // Actual barrel radius - found using nominal
    static float cRadius = 0; // Actual chamber radius

    //variables sent to LabRecon "From Code (IP)" Wiring object
    static float bLength = 0; // Barrel length
    static float bVolume = 0; // Barrel volume
    static float cLength = 0; // Chamber length
    static float cVolume = 0; // Chamber volume

    //Corresponding inner diameters of common PVC nominal sizes used for volume calculations
    static double[] barrelDiameters = { .546, .742, .957, 1.278, 1.5, 1.939, 2.323, 2.9 };
    static double[] chamberDiameters = { 1.939, 2.323, 2.9, 3.826, 5.761, 7.625 };

    static LRCodeLink LR = new LRCodeLink(); //LabRecon CodeLink class

    //program entry
    // "throws InterruptedException" needed for Thread.sleep(1000);
    public static void main(String[] args) {

        LR.Connect(40000); //connect port LabRecon is listening on
        LR.SetWaitMs(100); //set wait time for LR.GetVar()

        while(true) {

            // receive values from LabRecon program
            tLength = LR.GetVar(0);
            cbRatio = LR.GetVar(1);
            knobDiameter = LR.GetVar(2);
            knobDiameter = LR.GetVar(3);

            // find actual inner radii from nominal diameters
            bRadius = (float)(barrelDiameters[(int) knobDiameter])/2;
            cRadius = (float)(chamberDiameters[(int) knobDiameter])/2;

            // solve system of 4 equations
            bLength = (float)((tLength*Math.pow(cRadius,2))/(Math.pow(bRadius, 2)*cbRatio))/((Math.pow(cRadius, 2))/(Ma
            cLength = (float)(tLength-bLength);
    }
}
    
```

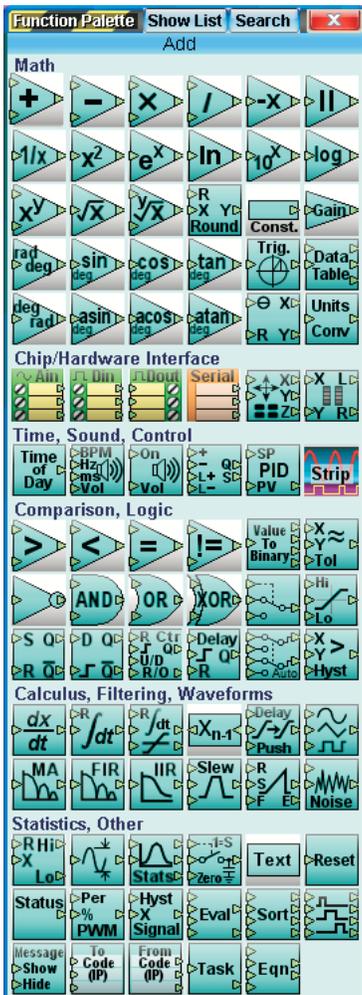
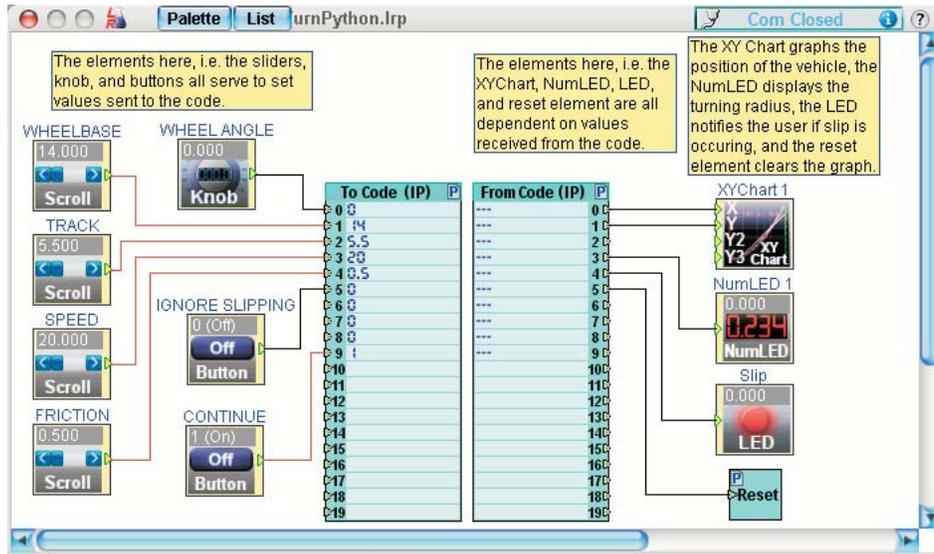
LabRecon Wiring view

LabRecon's Wiring view allows one to create connections between display and control objects on the Panel and LabRecon Wiring functions.

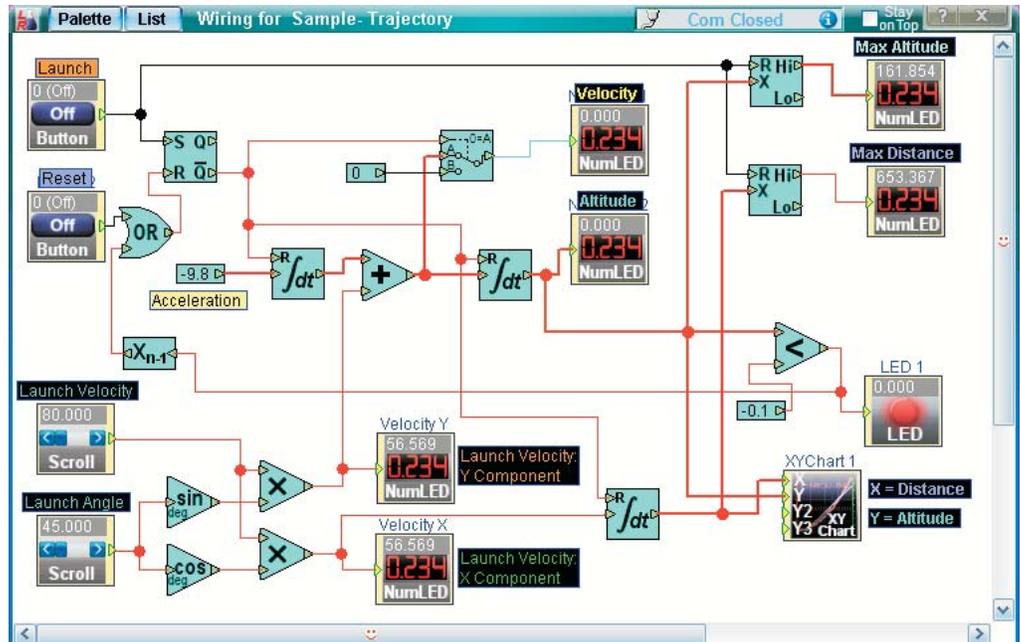
Below is the Wiring for the Car turning dynamics project, wherein the LabRecon "To Code"

and "From Code" functions provide the link to the Python code.

The wires provide a "graphical extension" of the variables in the user's code.



LabRecon has many additional Wiring functions, as its palette shows, to allow LabRecon to serve as a powerful graphical programming language on its own or in combination with user text based code.



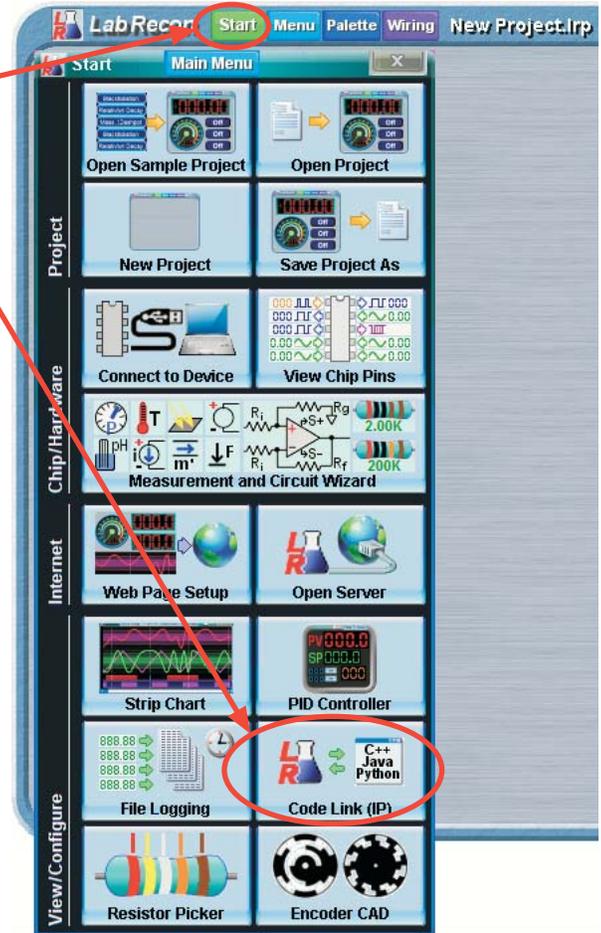
Opening Code Link Status

Clicking on the LabRecon “Start” and then on the “Code Link (IP)” tile opens the status window shown below, provides example code and diagnostics.

LabRecon acts as a **TCP/IP server** and accepts connections on the port number shown. By default the port is **40000**, but a different port number can be entered if desired.

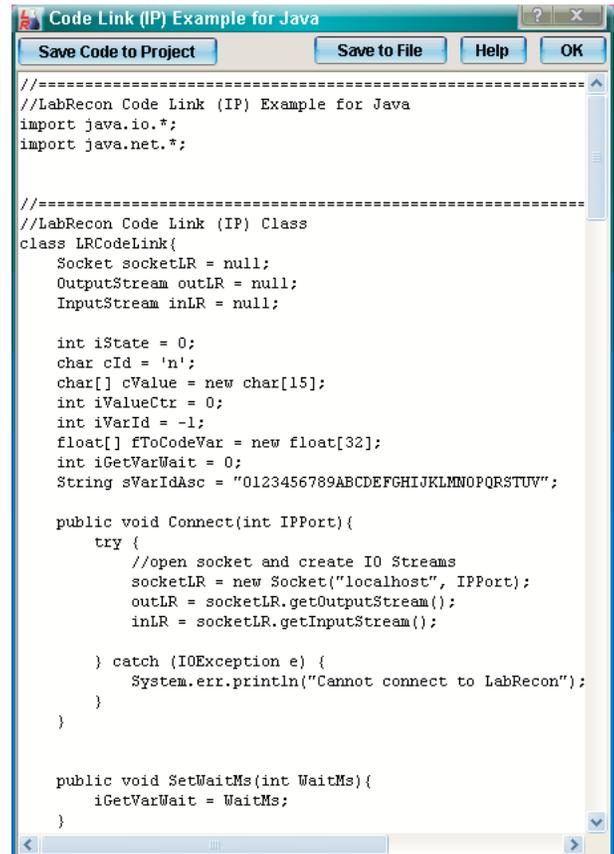
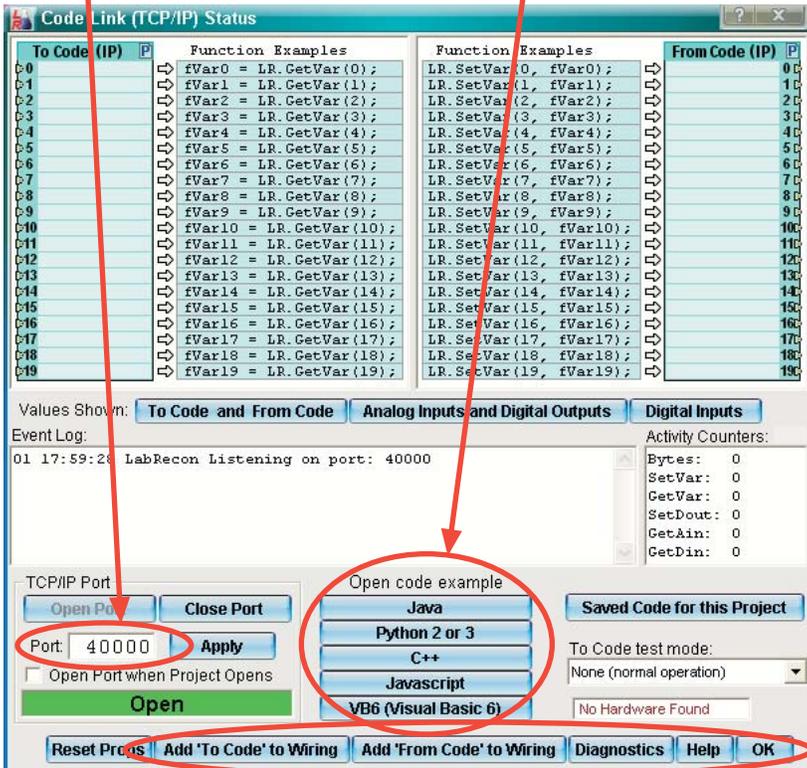
As demonstrated in the code examples, the user’s code connects to the LabRecon server port as a **client** and sends simple ASCII messages to set or request variables. The user’s code typically uses an **IP address of 127.0.0.1** (local host) when running on the same computer as LabRecon.

The bottom buttons can add the Code Link objects onto the **Wiring** and can also open a **Diagnostics** window to view TCP/IP events.



Clicking on a “Code example” button will open a code window, as shown on right, wherein the code can be copied and pasted into the code editor of the user’s choice.

TCP/IP Port



Additional Documents at www.LabRecon.com/Documents

LabRecon - Getting Started with the IoT (Internet of Things).pdf
LabRecon - Getting Started with the Measurement Wizard.pdf
LabRecon - Getting Started with Simulations.pdf
LabRecon - Equation Object.pdf
LabRecon - Getting Started with Robotics.pdf
LabRecon - Chip Datasheet (rev 2.0).pdf
LabRecon - MiniDAQ Datasheet (rev1.0).pdf
LabRecon - Chip Quick Start Sheet.pdf
LabRecon - Breadboard Experimenter (rev0).pdf
LabRecon - Photovoltaics.pdf
LabRecon - Reflow Oven PID Control.pdf
LabRecon - Measurement Configuration.pdf

Instructional Videos

www.LabRecon.com/videos

Revisions to this Document

Rev 0 Initial release

Support

www.LabRecon.com/help
support@LabRecon.com

Contact

info@LabRecon.com
Recon Industrial Controls Corp.
9 East Sheffield Ave.
Englewood, NJ 07631
201-894-0800

Copyrights and Trademarks

This documentation is Copyright 2016 by Recon Industrial Controls Corp.
LabRecon is a registered trademark of Recon Industrial Controls Corp.

Disclaimer of Liability

Recon Industrial Controls Corp does not assume any liability arising from the use of this product and related software described herein. Recon is not responsible for any equipment or property damage or personal injury resulting from the use or failure of this product and related software.

This product and related documentation are supplied as-is and no warranty is made or implied as to their use for any particular application.