A table-top smart manufacturing test bed with Fischertechnik and Raspberry Pi

This report documents the development and implementation of a table-top testbed for studies in smart manufacturing. The testbed was developed mostly for educational purposes but will also support our studies in Software-Defined Control of Smart Manufacturing Systems. As of now, the testbed supports the configuration of logic controllers on a network, remote monitoring, and reconfiguration of logic controllers. The test-bed is expected to be extended to support the export of data to a High Performance Computing (HPC) system for data processing, similarly to the work currently performed on SMART, the smart manufacturing test bed of the University of Michigan. This report describes the table-top test bed and includes instructions on how to operate it. For simplicity, it is organized in four sections: a general description of the test bed, a description of the physical model, the logic controllers, and the network.

# General description

Smart Manufacturing (SM) is a new trend that seeks to improve production by connecting the different stages of the production lifecycle, gathering large volumes of data from each stage and using it to dynamically adapt the system to variations in production demand, operating conditions, and the detection of sub-optimal settings. This approach is expected to revolutionize supply chain management, safety, sustainability, energy efficiency, and productivity in manufacturing. To achieve these goals, a larger degree of connectivity is implemented in the factory floor, to support communication between devices, control systems, and information systems, as shown in the multilayer structure of Figure 1.

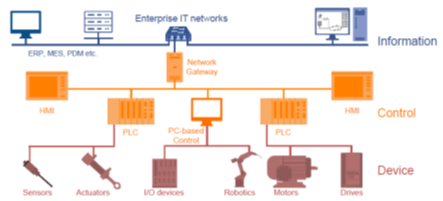


Figure 1. Multilayer structure of a smart manufacturing system showing the connection between devices, control systems, and information systems.

The purpose of our table-top testbed it to incorporate all the components of a SM system, but in way that is easier to operate, less expensive, and more portable. Currently, our test-bed incorporates: physical devices (e.g., conveyors, machine tools, sensors), controllers, and an HMI (in development). In the future, the testbed will be extended to support connection to information systems.

The table-top system is designed to mimic the behavior of a manufacturing facility divided in four cells, each one of them controlled by a separate logic controller. Figure 2 shows the top view of the testbed, which contains:

* Cell 1: Conveyor 1 and Ram 1
* Cell 2: Conveyor 2 and Machine 1 (boring machine)
* Cell 3: Conveyor 3 and Machine 2 (drill)
* Cell 4: Conveyor 4 and Ram 2.

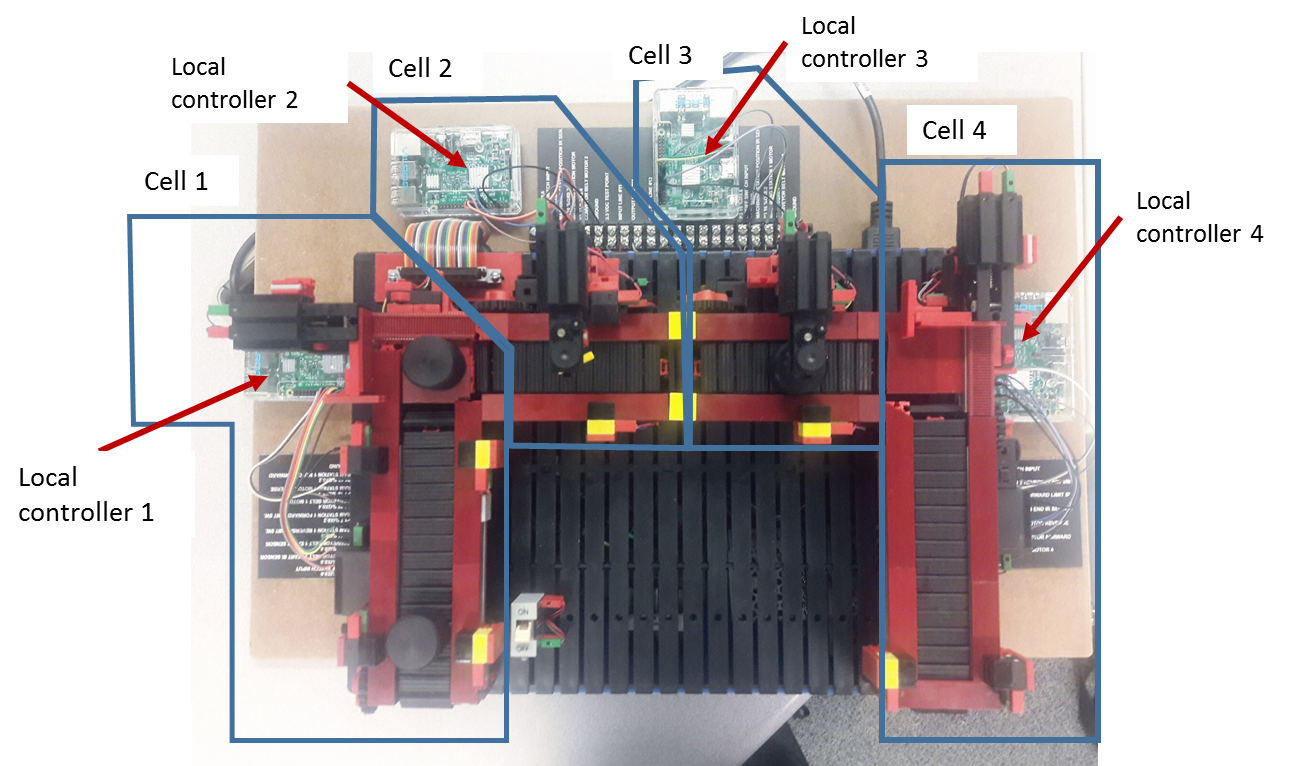


Figure 2: Top view of table-top testbed

# Physical model

The testbed is based on the indexed line with two machine tools model of Fischertechnik[[1]](#footnote-1). The model was purchased from Tim King Electronics, based in Allen Park, MI. As an alternative, the model can be purchased from other Fischertechnik distributors in the US and the world.

The indexed line is made up of four 30 mm wide conveyor belt lines. Each conveyor belt is powered by an S motor and a gear box. There are two push rams powered by an S motor each and a gear rack, which move parts sideways. Each push ram has two limit switches for position detection. The two machining stations use IR sensors to detect the arrival of a part at the station. The model has a central ON/OFF switch, which also acts as an emergency stop. There are several IR sensors distributed along the conveyor lines for detecting the position of parts entering and leaving conveyor 1, in front of both machines, and one at the end of conveyor 4. Each one of the machining stations are made up of an S motor and a gear box that simulate a boring machine and a vertical drill. Both machines only simulate machining by rotating and do not move up or down.

A 12 VDC switching power supply provides all the power needed for both the RoBo Pro interface and the I/O extension unit, which can be found inside the blue box. Additionally, the power supplies are coupled to a customized interface board that allows the system to operate with 3.3 VDC inputs and outputs (requirement to work with Raspberry Pi, as explained in the following section). The interface board is divided in two sections: one for inputs and one for outputs. The addressing sheet for the model inputs and outputs is given in Table I, for a 34 ribbon standardized cable:

Table I: Addressing sheet[[2]](#footnote-2)

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin No.** | **Wire color** | **Raspberry Pi terminal** | **Description** |
| 1 | Brown | P1-P2 19 (%IX0.6) | ON/OFF switch |
| 2 | Red | P1 15 (%IX0.5) | Conveyor belt 1: Start IR sensor |
| 3 | Orange | P1 13 (%IX0.4) P2 11 (%IX0.3) | Conveyor belt 1: End IR sensor |
| 4 | Yellow | P1 11 (%IX0.3) | Ram station 1: Reverse limit switch |
| 5 | Green | P1 7 (%IX0.2) | Ram station 1: Forward limit switch |
| 6 | Blue | P2 13 (%IX0.4) P3 11 (%IX0.3) | Machine 1: position IR sensor |
| 7 | Violet | P3 13 (%IX0.4) P4 11 (%IX0.3) | Machine 2: position IR sensor |
| 8 | Gray | P4 11 (%IX0.3) | Ram station 2: Reverse limit switch |
| 9 | White | P4 7 (%IX0.2) | Ram station 2: Forward limit switch |
| 10 | Black | P4 13 (%IX0.4) | Conveyor belt 4: End IR sensor |
| 11 | Brown | P3-P4 19 (%IX0.6) | ON/OFF switch |
| 12 | Red |  | Extra input |
| 13 | Orange |  | Extra input |
| 14 | Yellow |  | Extra input |
| 15 | Green |  | Extra input |
| 16 | Blue |  | Extra input |
| 17 | Violet | +3.3 VDC |  |
| 18 | Gray | P1-4 6 (Ground) | Ground: shared by all R Pi |
| 19 | White | P1 22 (%QX0.4) | Conveyor belt 1: motor |
| 20 | Black | P1 18 (%QX0.3) | Ram station 1: motor reverse |
| 21 | Brown | P1 16 (%QX0.2) | Ram station 1: motor forward |
| 22 | Red | P2 22 (%QX0.4) | Conveyor belt 2: motor |
| 23 | Orange | P2 16 (%QX0.2) | Machine 1: motor |
| 24 | Yellow | P3 22 (%QX0.4) | Conveyor belt 3: motor |
| 25 | Green | P3 16 (%QX0.2) | Machine 2: motor |
| 26 | Blue | P4 18 (%QX0.3) | Ram station 2: motor reverse |
| 27 | Violet | P4 16 (%QX0.2) | Ram station 2: motor forward |
| 28 | Gray | P4 22 (%QX0.4) | Conveyor belt 4: motor |
| 29 | White |  | Extra output |
| 30 | Black |  | Extra output |
| 31 | Brown |  | Extra output |
| 32 | Red |  | Extra output |
| 33 | Orange |  | Extra output |
| 34 | Yellow |  | Extra output |

Some features of the model are:

* Standard 30 mm wide conveyor belt systems, which makes the system compatible with similar Fischertechnik models.
* IR start sensor, to detect the arrival of new parts to the model.
* IR conveyor belt sensor to detect when part is about to enter the first transfer ram station.
* IR conveyor belt sensor to detect when a part is in position to be machined at each station.
* A second transfer ram station to move the part onto the last conveyor belt.
* Conveyor belt system that can be connected with other Fischertechnik factory models.

Inputs and outputs

* Inputs: 4 limit switches, 5 IR sensors, and 1 ON/OFF switch
* Outputs: 8 DC motors

# Logic controllers

The testbed is designed to replicate the behavior of actual industrial controller. We adopt the standardized languages from IEC-61131-3: ladder logic, structured text, instruction lists, function block diagrams, and sequential function charts. We adopt the tools developed by the Open PLC project to use Raspberry Pi as emulators of industrial PLCs.

To operate, it is first necessary to connect four Raspberry Pi 3 Model B[[3]](#footnote-3) following the instructions given in Table I. Then, we need to follow these steps on *every* Raspberry Pi:

* Install wiringPi to be able to interface with the GPIO pins. The program can be downloaded from its [website](http://wiringpi.com/download-and-install/) or from its [Github repository](https://github.com/WiringPi/WiringPi).
* Install a few packages from the Raspberry Pi terminal:  
  *sudo apt-get update  
  sudo apt-get install build-essential pkg-config bison flex  
  sudo apt-get install autoconf automake libtool make nodejs git*
* Install OpenPLC on each Raspberry Pi from its [Github repository](https://github.com/thiagoralves/OpenPLC_v2.git). Alternatively, the repository can be cloned directly from terminal:  
  *git clone* [*https://github.com/thiagoralves/OpenPLC\_v2.git*](https://github.com/thiagoralves/OpenPLC_v2.git) *cd OpenPLC\_v2  
  ./build.sh*At the end of the installation, select *4) Raspberry Pi* as the physical device to control.
* A separate computer will be used to program the Raspberry Pi. In the programming computer, install PLCopen Editor v.1.1 from its [website](http://www.openplcproject.com/plcopen-editor).

After the installation, open the PLCOpenEditor, which should look similar to Figure 3.

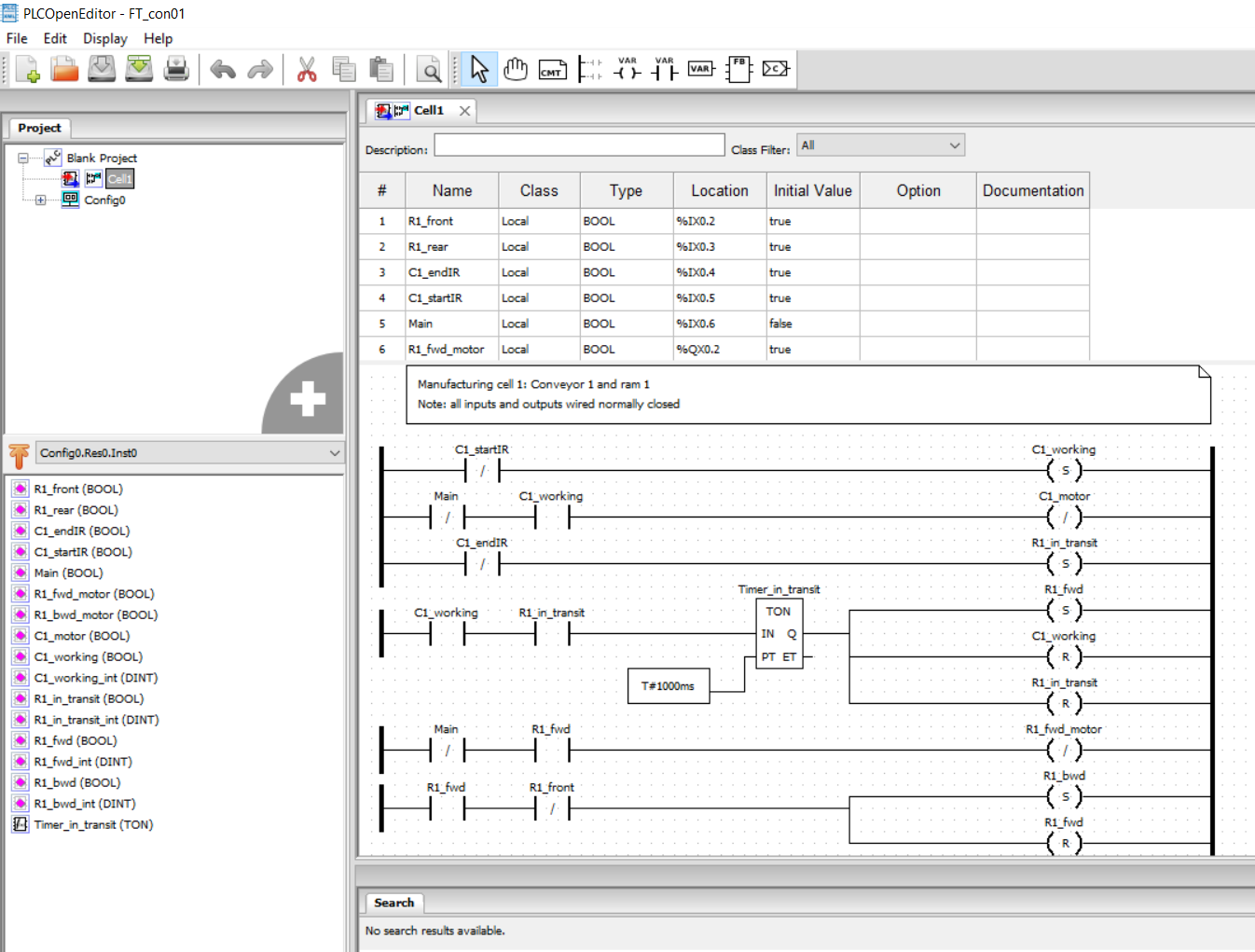


Figure 3. PLCOpenEditor environment and a ladder diagram.

From this environment, it is possible to develop programs that can be uploaded to any of the configured Raspberry Pi if the programming computer and the R Pi are part of the same network. The GPIO pins must be addressed following Figure 4.

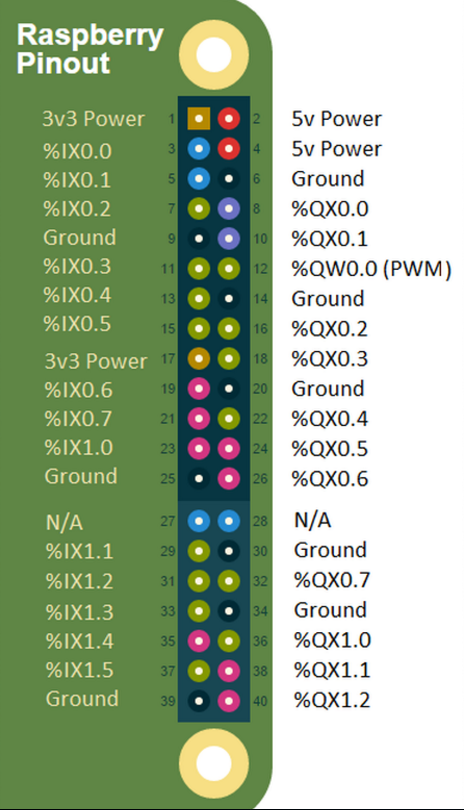


Figure 4. Addressing of GPIO from PLCOpenEditor.

Any standardized programming language can be used to developed the program, which is stored as an XML file. The XML file is then compiled into an ST file (structured text) by PLCOpenEditor (File -> Generate program), which is then uploaded to each Raspberry Pi. The process is illustrated in Figure 5.

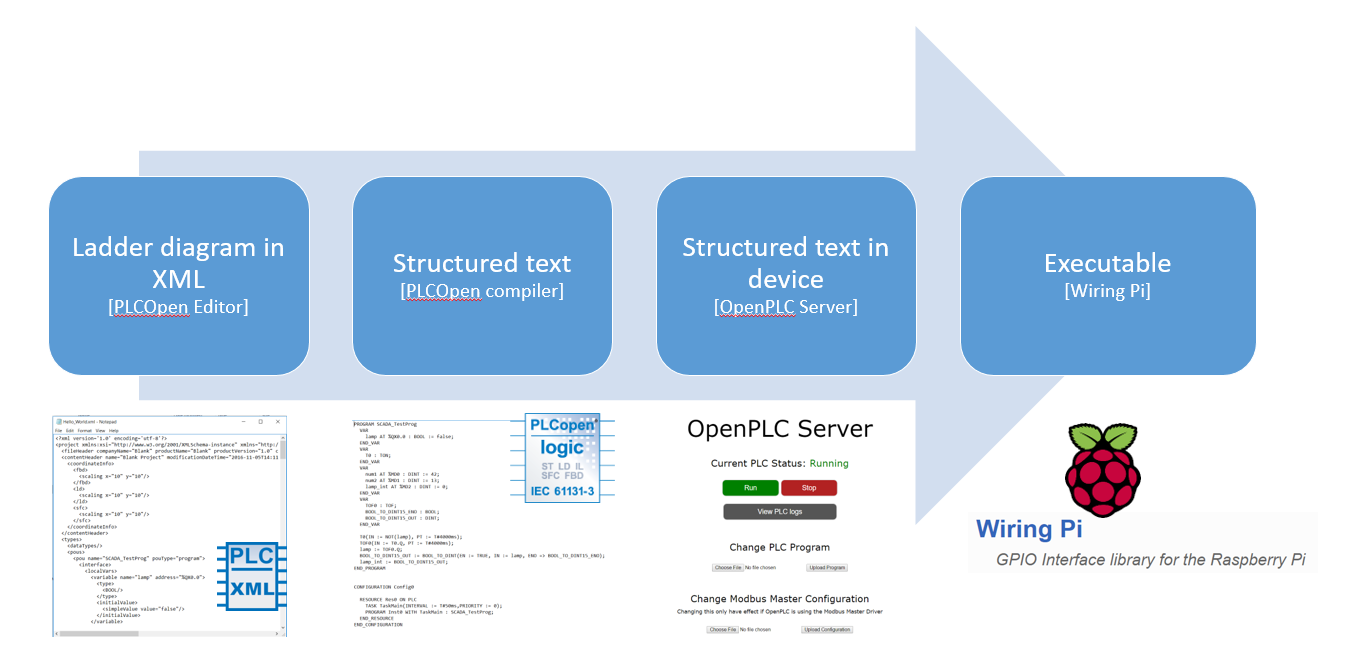


Figure 5. Implementation of ladder diagrams on Raspberry Pi.

The upload of the structured text file to the Raspberry Pi is performed by loading a server from each individual Raspberry Pi. To do so, make sure that the programming computer and the Raspberry Pi are connected to the same network, and then access the Raspberry Pi terminal. Once in the terminal, go to the OpenPLC\_v2 directory, and type in:

*sudo nodejs server.js*

That should start the OpenPLC server. Then, from another terminal window, use the *ifconfig* command to find out the IP address of the Raspberry Pi.

From the programming computer, open any browser and go to https://IPAddress:8080, e.g., http://192.168.1.1:8080. That should open the OpenPLC server. Upload the ST file, and run the code.

Alternatively, the programming computer can be used to SSH into the individual Raspberry Pi and launch the server without having to connect a monitor, and a keyboard to each Raspberry Pi. They can be accessed with PuTTy, for example, by using the username **pi** and the password **raspberry**, which are the default passwords for these devices.

# Network

The entire system has been connected to a local network, which is (temporarily) isolated from the Internet. Two routers are used for this purpose[[4]](#footnote-4). A schematic showing the current configuration is given in Figure 6. The purpose of the network is to support:

* Configuration and reconfiguration of logic controllers from the programming computer.
* The remote visualization of IO data and holding registers on the programming computer.
* The development of a centralized Human Machine Interface (HMI) on the programming computer (to be developed).

*Router Configuration*

Since the routers provide both wireless and wired access, we can connect to them through either WiFi or a cable. We just need to login into the admin page to change their default configurations.

The default internal (LAN) IP addresses of these two routers are 192.168.2.1. So we need to navigate the browser to 192.168.2.1 for accessing the admin page. For Router One we change its LAN address to 192.168.1.1. Keep Router Two’s LAN IP address as it is. Subnet masks should both be 255.255.255.0. See Figure 6 and Figure 7.

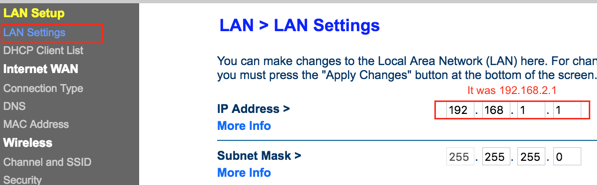


Figure 6. ***Router One*** LAN IP Address

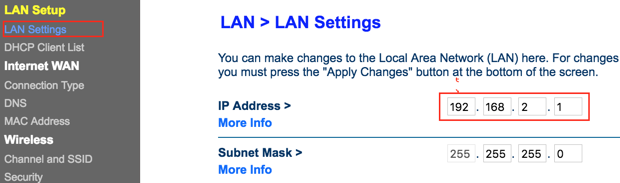


Figure 7. ***Router Two*** LAN IP Address

Router Two’s WAN interface is connected to one of Router One’s LAN interfaces therefore Router Two needs an address from Router One’s network, 192.168.1.\*. Figure 9 shows how to wire the two routers. We assign a dedicated address 192.168.1.101 to Router Two, as shown in Figure 6.



Figure 8. ***Router Two*** Static WAN IP Address

*Static IP Configuration*

It’s easy to configure static IP addresses for Raspberry Pis. Open /etc/dhcpcd.conf as root and append the following lines to the end of the file. We use Raspberry Pi Cell 1 as an example.

|  |
| --- |
| interface eth0  static ip\_address=192.168.2.10  static routers=192.168.2.1 |

The file will finally look like:

|  |
| --- |
| ...  ...  # A hook script is provided to lookup the hostname if not set by the DHCP  # server, but it should not be run by default.  nohook lookup-hostname  interface eth0  static ip\_address=192.168.2.10  static routers=192.168.2.1 |

Simply reboot the Raspberry Pi and the configuration will take effect. For other Raspberry Pis, see Figure 9 for their assigned IP addresses. Note that in /etc/dhcpcd.conf the first three numbers in ip\_address and routers should always be the same.

*Controller Configuration*

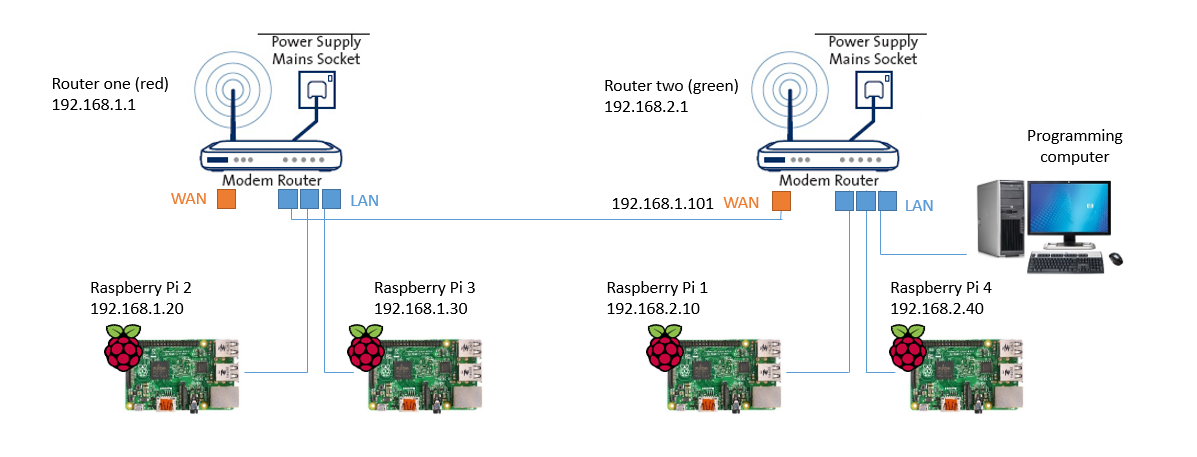


Figure 9. Schematic of the network. The IP addresses are static.

For the configuration of controllers, each Raspberry Pi is made accessible from the programming computer. Static IP addresses are assigned to each Raspberry Pi, as shown in Fig 6. The IP addresses can be used to: (1) access the terminal through an SSH service (e.g., PuTTy), (2) launch the OpenPLC Server from each R Pi, and (3) access each OpenPLC Server application and upload the logic program.

Additionally, the same IP addresses can be used to monitor IO and internal variables from each Raspberry Pi. OpenPLC is already built on top of a Modbus network. A quick Modbus simulator (e.g., Radzio) can be used to access each variable just by entering the IP address. We expect to improve this process with the design of an HMI that will allow the visualization of all IO and the internal state of the system without having to use a Modbus simulator. We are evaluating the use of ScadaBR for that purpose.

# ScadaBR

ScadaBR is an open-source, Java-based application that performs Supervisory Control And Data Acquisition (SCADA). The application can be accessed from any web browser, but it has been reported to work best on Firefox or Chrome. ScadaBR was developed at the University of Sao Paulo, Brazil (hence, the BR). The main interface of ScadaBR offers visualization of variables, graphs, statistics, protocol configuration, alarms, construction of HMI screens, and a number of configuration options. More information is available at <http://www.scadabr.com.br/>

These are the steps to install ScadaBR on Windows 8 or 10:

* Install Java jre1.8.0\_77 – the installation has been tested only for this distribution.
* Install Apache Tomcat 7 32-bit/64-bit Windows Service Installer from https://tomcat.apache.org/download-70.cgi  
  Do not install any more recent versions as ScadaBR 1.0 was compiled for Tomcat 7.0

When installing Apache Tomcat, follow default options, except for the configuration. Once in the configuration window, choose 8005 for the server shutdown port, 8085 for the HTTP1.1 connector port, and 8009 for the AJP/1.3 connector port. Also, make sure to select the path to the current version of Java JRE

* Download ScadaBR from its Dropbox repository:  
  https://www.dropbox.com/s/3m8uf4kvvfgezfm/ScadaBR.rar?dl=0  
  Paste the ScadaBR in the C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0\webapps\ folder.
* Once ScadaBR has been copied, initiate Apache Tomcat (C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0\bin\Tomcat7w.exe)
* In the General window, set startup type as automatic, and start Apache Tomcat.
* Go to the Java window to ensure that the path to Java JDK is correct.
* To verify that Apache Tomcat has been started, try to start again and the option will be inhibited.
* Once started, go to C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0\bin and verify that the scadaBR folder has been created.
* Create a shortcut on the Desktop and point it to <http://localhost:8085\ScadaBR>  
  Default credentials are user: admin and password: admin

Once in ScadaBR, we can use the same IP addresses to access the input, output (coil), and internal (holding register) of each Raspberry Pi from the programming computer. In ScadaBR, go to the *Data sources* tab and *Add* a new Modbus IP data source. The first Raspberry Pi, for example, is configured as shown in Figure 10. Note that the update (sampling) period has been defined as 0.2 s, but the user can change it easily. Make sure to choose *TCP with keep-alive* as the transport type or ScadaBR will finish the execution of the OpenPLC server after one minute. Repeat the process for the remaining Raspberry Pi changing only the IP address of the source.

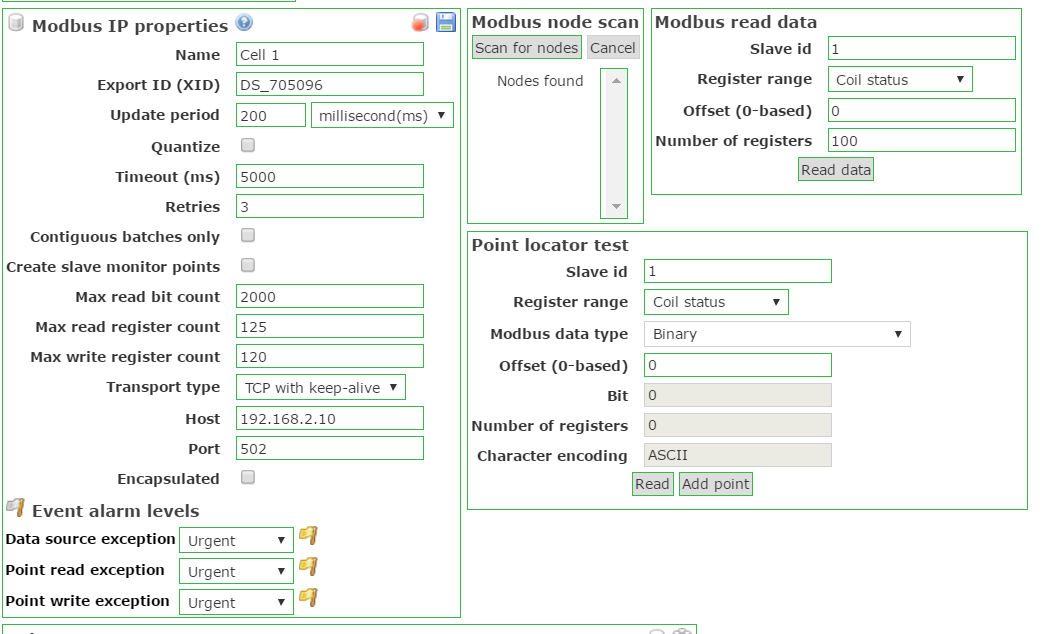


Figure 10. Configuration of the first Raspberry Pi (192.168.2.10) as a Modbus IP data source.

The variables to monitor are defined as *data points*, as shown in Figure 11. For convenience, we are using the same names used in the XML files containing ladder logic. The address to access is defined as an *offset*. In the case of internal variables (holding registers), counting starts from 2049, and goes every 2 bits (i.e., MD%1 is 2051 and MD%3 is 2055).

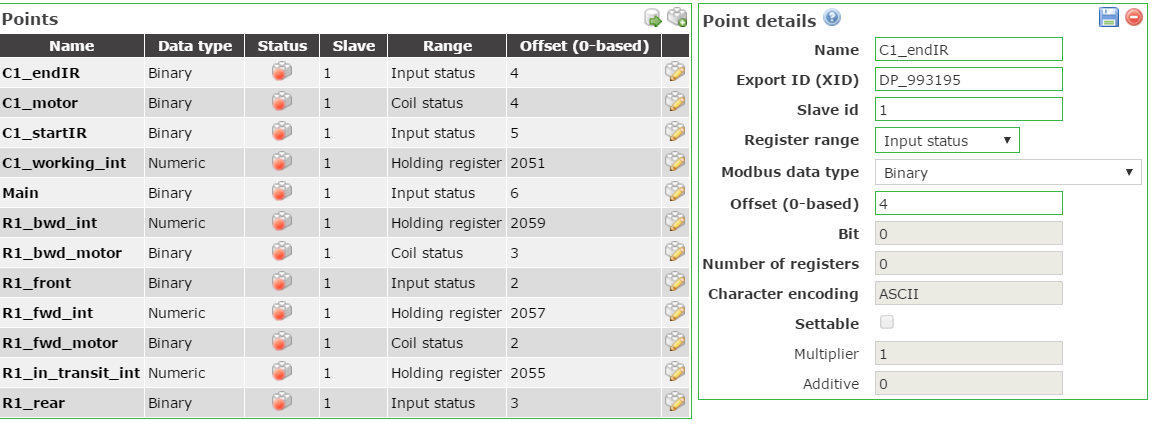


Figure 11. Data points for first data source.

Future work will include the design of an HMI using the tab *Graphical view* and the curation of acquired data using the import/export and SQL options.

1. Available on: http://www.fischertechnik.de/en/desktopdefault.aspx/tabid-24/41\_read-63/usetemplate-2\_column\_pano/Fischertechnik is a vendor of educational material from K-12 education to technical education (i.e., robotics and PLC programming). [↑](#footnote-ref-1)
2. P1 refers to Raspberry Pi number 1 and the following number is the corresponding GPIO pin. The code in parenthesis is the label of the GPIO from OpenPLC Editor, as shown in Figure 3. [↑](#footnote-ref-2)
3. It is recommended to buy the Raspberry Pi 3 Complete Starter Kit from VILROS because it comes with all of the accessories required for its operation: micro SD card, charger, and micro SD card adapter. Additionally, the micro SD card comes already loaded with NOOBS software. [↑](#footnote-ref-3)
4. We use 2 Belkin wireless G 4-port routers. There have wireless capabilities, but we’re currently only using wired connections. The information for each router is: Router 1 (SSID: Belkin\_5BC452, Password: 1E34963E, Type: WPA/WPA2) and Router 2 (SSID: Belkin\_6561CE, Password: 2E5D5693, Type: WPA/WPA2). [↑](#footnote-ref-4)