Defence College
of Technical Training

SCADA VULNERABILITY AWARENESS AND SECURITY TRAINING

Bascule Bridge Build Guide

SIMR 21/001

Submitted By: Syndicate 1

Version 1.0

17/03/2023

Contents

Introduction

A Supervisory Control and Data Acquisition (SCADA) system controls and monitors industrial processes and infrastructure such as power plants, water treatment facilities, and oil refineries. The system is made up of hardware and software components that work together to gather data, analyse it, and present it to operators. The hardware components of a SCADA system include sensors and actuators that measure and control physical processes. These devices are connected to a network of Programmable Logic Controllers (PLCs) that gather data and perform automated control functions. The data collected by the sensors is sent to the SCADA software, which stores it and displays it to operators in a user-friendly format on a Human Machine Interface (HMI) display.

SCADA system help ensure the smooth and efficient functioning of critical infrastructure, and any disruptions or failures can lead to significant consequences, including economic losses and risks to public safety. However, SCADA systems are also vulnerable to cyberattacks, which can have severe consequences. Some of the vulnerabilities include weak authentication mechanisms, insecure communication channels, and outdated software and hardware.

The aim of the Bascule bridge project is to design and develop a SCADA system that can be used for cyber training purposes. The system is intended to educate individuals who have limited or no knowledge about SCADA systems on the cybersecurity concerns related to these systems.

Design

The design emulates the functionality of a bascule bridge, with the capability of opening and closing through either manual switch control or an HMI controller operated by the bridge operator. Automated bridge operation is enabled by the integration of IR sensors, which detect incoming ships and trigger the bridge to open and close accordingly. A warning alarm is also activated before the bridge opens, after which barriers are closed and traffic lights turn red to stop traffic and pedestrians from accessing the bridge.

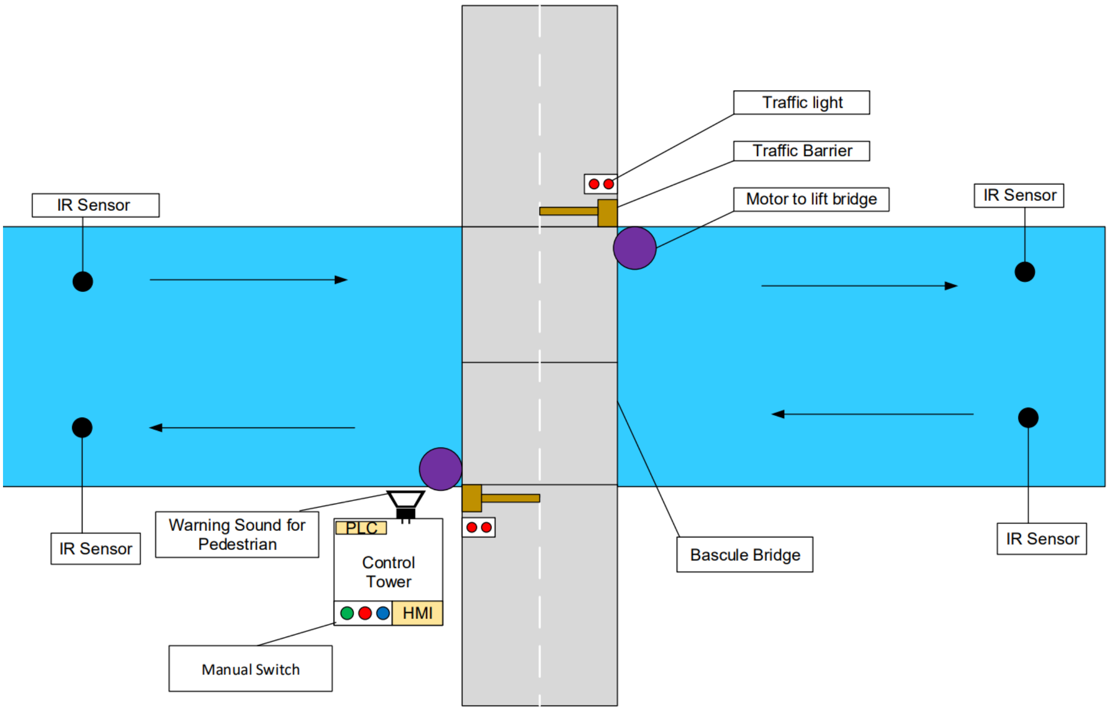


Fig 1: High level design concept

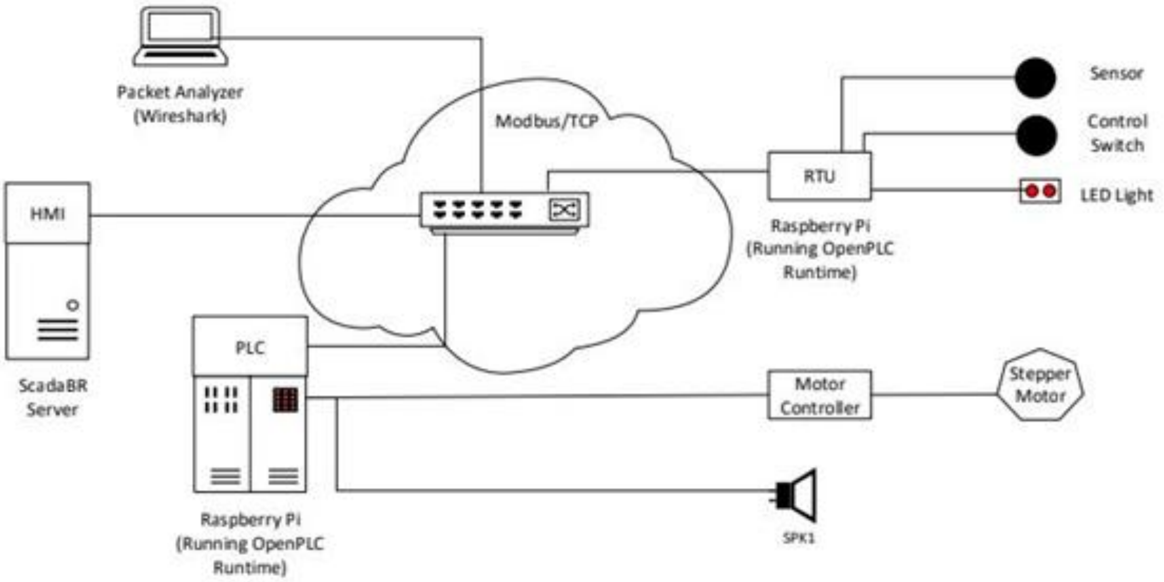

Packet Analyzer (Wireshark)
Modbus/TCP
HMI
ScadaBR Server
PLC
Raspberry Pi (Running OpenPLC Runtime)
RTU
Raspberry Pi (Running OpenPLC Runtime)
Sensor
Control Switch
LED Light
Motor Controller
Stepper Motor
SPK1

Fig 2: L1 Network Diagram

The network diagram in Figure 2 shows how the PLC, HMI, and RTU communicate with each other through the Modbus/TCP communication protocol. The RTU functions as a remote device that provides data from connected sensors and devices to the PLC, which then uses this information to monitor and control the connected devices. The HMI provides users with a graphical interface to monitor and control different aspects of the ICS. A packet analyser could be utilised to inspect the Modbus/TCP traffic traversing through the system.

Hardware Required for the Build

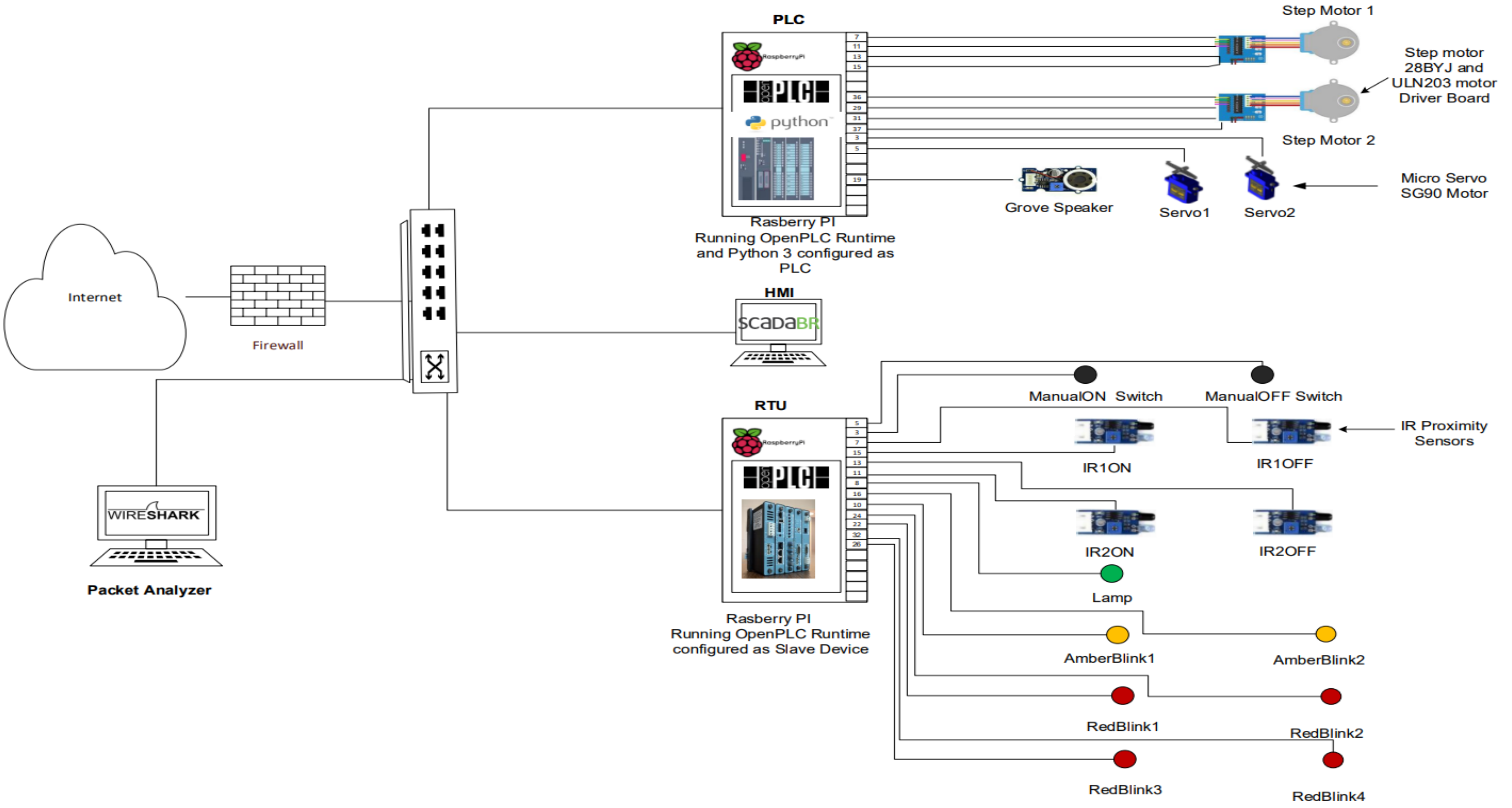
To build the representative SCADA system, the following hardware was required:

Ser	Item	Quantity	Requirement
1	Laptop		For project research and development on the project.
2	Raspberry Pi	3	Used as a PLC, HMI and RTU.
3	6-Port Switch	1	For all devices to be interconnected.
4	Step Motor 28bYJ-48 with ULN2003 Driver Board	2	Used for opening and closing the bascule bridge.
5	Micro Servo SG90	2	Used for opening and closing the traffic barrier.
6	Grove Speaker	1	Used for the warning alarm.
7	IR Proximity Sensors	4	To automate the bascule bridge operation.
8	Electronic Accessory Kit	3	Provide important components like a breadboard, jumper wire, power supply module, LED, resistors, etc. to create the circuit.
9	Pi 7-inch touchscreen display	1	Will be used as an HMI display.
10	3D Printer	1	To build the 3D model of the bascule bridge.

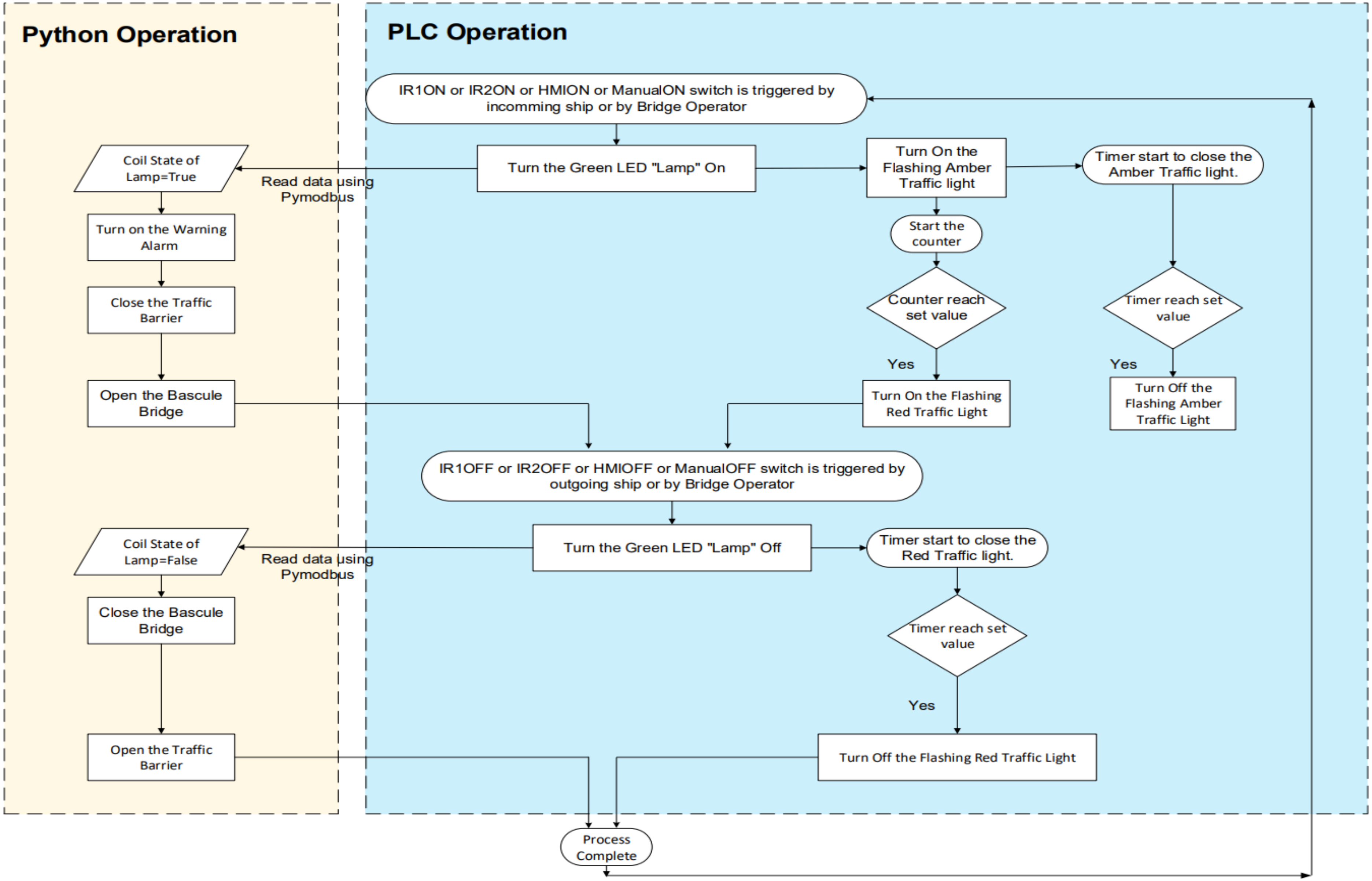
Software Required for the Build

Ser	Item	Requirement
1	OpenPLC Editor	Used to create PLC programmes.
2	OpenPLC Runtime	Used to execute the PLC programme and emulate the Raspberry Pi as a PLC and RTU.
3	ScadaBR	Used as HMI.
4	Python 3 with RPi.GPIO and PyModbus library	Control the operation of step motors, servo motors, and Grove speakers.
5	FreeCAD 0.20	Used to design a 3D model of the bascule bridge.
6	Raspberry Pi OS	The operating system for the Raspberry Pi.
7	Wireshark	Packet analyser.

Network Diagram



Flow Diagram



Python Operation
PLC Operation
IR1ON or IR2ON or HMION or ManualON switch is triggered by incomming ship or by Bridge Operator
Coil State of Lamp=True
Read data using Pymodbus
Turn the Green LED "Lamp" On
Turn On the Flashing Amber Traffic light
Timer start to close the Amber Traffic light.
Turn on the Warning Alarm
Start the counter
Close the Traffic Barrier
Counter reach set value
Timer reach set value
Yes
Yes
Open the Bascule Bridge
Turn On the Flashing Red Traffic Light
Turn Off the Flashing Amber Traffic Light
IR1OFF or IR2OFF or HMIOFF or ManualOFF switch is triggered by outgoing ship or by Bridge Operator
Coil State of Lamp=False
Read data using Pymodbus
Turn the Green LED "Lamp" Off
Timer start to close the Red Traffic light.
Close the Bascule Bridge
Timer reach set value
Yes
Open the Traffic Barrier
Turn Off the Flashing Red Traffic Light
Process Complete

Build Process

Build process is divided into following phases:

1) OpenPLC Editor Install
2) PLC Build
3) RTU Build
4) Ladder Logic
5) Python Code
6) HMI Build
7) 3D Printing
8) Assembly and Testing

1. OpenPLC Editor Install

OpenPLC is a user-friendly and easy-to-use open-source PLC that is designed to provide a flexible and cost-effective solution for industrial control and automation. It is the first fully functional open-source PLC, compliant with both the software and hardware standards of the IEC 61131-3 standard, which defines the software architecture and programming languages for PLCs. With its support for various programming languages, customizable and modular architecture, OpenPLC can be used for a broad range of applications, from small-scale projects to large industrial systems. Its open-source nature ensures that users can modify and enhance it as needed.

The OpenPLC editor is a graphical programming tool used for developing and editing PLC programs on the OpenPLC platform. It offers a user-friendly interface with drag-and-drop function blocks, support for various programming languages, and real-time simulation for testing and debugging PLC programs. It can be used to upload PLC code directly to any board or system running OpenPLC Runtime.

To install OpenPLC Editor:

1. OpenPLC Editor can run on any platform that has support for Python. Currently, there are official installers for Windows, Linux (Debian, Ubuntu, Fedora, and variants), and macOS.
2. To install OpenPLC Editor for the officially supported platforms, download the provided installer for your operating system from https://openplcproject.com/download/ and follow the instructions on the screen to have it installed in your system.
3. For other operating systems, you can still install and run OpenPLC Editor, provided you meet the dependencies. Follow the instructions on the official website documentation, https://openplcproject.com/docs/installing-openplc-editor/, to install OpenPLC Editor on any other OS.

2. PLC Build

The process of building a PLC involves two phases. The first phase is to install the Raspberry Pi operating system and build the Pi itself. The second phase is to install the OpenPLC Runtime on the Pi.

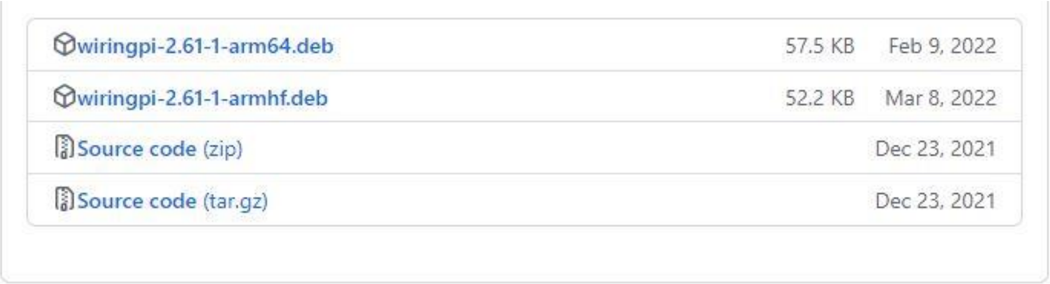
Note: For this project, this Raspberry Pi is configured as ‘PiMaster1’ with Ip address ‘192.168.1.15’

Phase 1: This phase will provide step-by-step instructions on how to install the operating system on the Raspberry Pi.

1. Gather all the necessary materials, including a Raspberry Pi board, an SD card with a capacity of at least 8 GB, a power supply, and a computer with an SD card reader.
2. Go to the official Raspberry Pi website (https://www.raspberrypi.com/software/) and download the Raspberry Pi Imager for your operating system.
3. Once the download is complete, open the installer and follow the on-screen instructions to install the Raspberry Pi Imager on your computer.
4. Insert an SD card into your computer. Make sure the SD card is blank, or backup any important data, as the next steps will format the card.
5. Open the Raspberry Pi Imager on your computer.
6. Select the operating system image you want to download from the "Operating System" section.
7. Choose the model of your Raspberry Pi from the "Board" section.
8. Select the SD card as the storage location for the Raspberry Pi image.
9. Click on the "Write" button to begin the installation process.
10. Wait for the Raspberry Pi imager to download the selected OS image and write it to the SD card. This may take a few minutes, depending on the speed of your SD card and internet connection.
11. Once the installation process is complete, remove the SD card from your computer and insert it into your Raspberry Pi.
12. Power on your Raspberry Pi and wait for it to boot up. The first boot may take a few minutes as the Raspberry Pi configures the OS and installs updates.

Phase 2: The OpenPLC Runtime enables the execution of PLC programs produced in the OpenPLC Editor. This phase will walk you through the process of installing the OpenPLC Runtime on the Raspberry Pi.

1. Connect your Raspberry Pi to the internet and open the Terminal.
2. Update your Raspberry Pi by entering the following command “sudo apt-get update”.
3. Install Git by running the following command in the terminal:
   ‘sudo apt-get install git’
4. Clone the OpenPLC Runtime repository using Git by running the following command in the terminal:
   ‘git clone https://github.com/thiagoralves/OpenPLC_v3.git’
5. Navigate to the OpenPLC Runtime directory by running the following command in the terminal:
   ‘cd OpenPLC_v3’
6. Install the dependencies required for OpenPLC Runtime by running the following command in the terminal:
   ‘sudo ./install.sh rpi’
7. To run OpenPLC on a Raspberry Pi, you need to have the WiringPi library installed, which is responsible for controlling the GPIO pins on the board. You can download the latest .deb version of WiringPi to install it on your Raspberry Pi from:
   https://github.com/WiringPi/WiringPi/releases/



wiringpi-2.61-1-arm64.deb	57.5 KB	Feb 9, 2022
wiringpi-2.61-1-armhf.deb	52.2 KB	Mar 8, 2022
Source code (zip)		Dec 23, 2021
Source code (tar.gz)		Dec 23, 2021

The -armhf.deb file should be used on 32-bit OS (Raspberry Pi 3 and under), and the -arm64.deb is meant for 64-bit OS (Raspberry Pi 4 and up).

8. Download the appropriate file for your architecture on your Raspberry Pi and install it with the dpkg command:
   ‘dpkg -i wiringpi-[version]-armhf.deb’
   Or
   ‘dpkg -i wiringpi-[version]-arm64.deb’
9. Test that the WiringPi installation finished successfully with the command:
   ‘gpio -v’
10. Use the following command on the terminal to check the status of the OpenPLC service:
    ‘sudo systemctl status openplc’
    If the service is running, you should see a message that says "Active: active (running)". If the service is not running, you should see a message that says "Active: inactive (dead)".
11. To start the OpenPLC Runtime service, type the following command in the terminal:
    ‘sudo systemctl start openplc’
    (To stop the OpenPLC Runtime service, type the following command in the terminal:
    ‘sudo systemctl stop openplc’)
12. By default, the OpenPLC Runtime web server listens on port 8080. Open a web browser on your Raspberry Pi and enter the following URL in the address bar:
    ‘http://localhost:8080’ or ‘http://192.168.1.15:8080’ (where 192.168.1.15 is Raspberry pi Ip address).
13. You should now see the OpenPLC Runtime web interface, which allows you to configure various parameters of the runtime.
14. On the OpenPLC Runtime web interface, go to Hardware, and ensure that under OpenPLC Hardware Layer, ‘Raspberry Pi’ is selected.

For further information on how to install the OpenPLC Runtime visit the OpenPLC Project Documentation section.

3. RTU Build

Repeat the Phase 1 and Phase 2 of the PLC build with the separate Raspberry Pi.

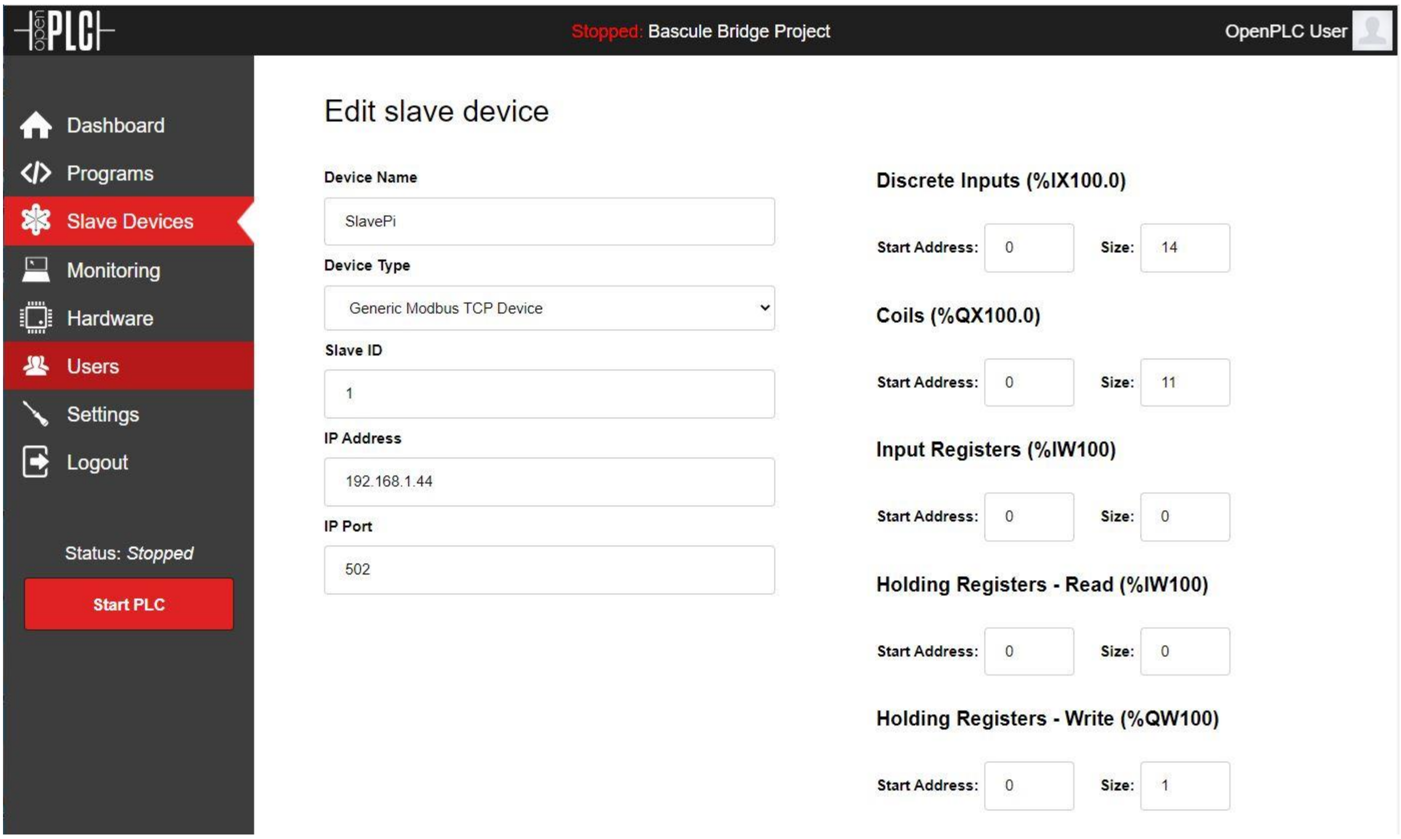
Note: For this project, this Raspberry Pi is configured as ‘PiSlave1’ with Ip address ‘192.168.1.44’

To run this PiSlave1 as RTU, on the OpenPLC Runtime it will run the ‘Blank Program’ and the slave configuration will be carried out in PiMaster1 running as PLC.

Slave configuration:

1. Access the OpenPLC Runtime web interface for PiMaster1 by opening a web browser and entering http://192.168.1.15:8080 in the address bar.

2. To add a new slave device, navigate to the left-hand panel in the OpenPLC Editor and click on 'Slave Device'. From there, click on the 'Add New Device' button to create the new device.
3. Configure the parameters as follow:



Edit slave device

Device Name: SlavePi
Device Type: Generic Modbus TCP Device
Slave ID: 1
IP Address: 192.168.1.44
IP Port: 502

Discrete Inputs (%IX100.0)
Start Address: 0 Size: 14

Coils (%QX100.0)
Start Address: 0 Size: 11

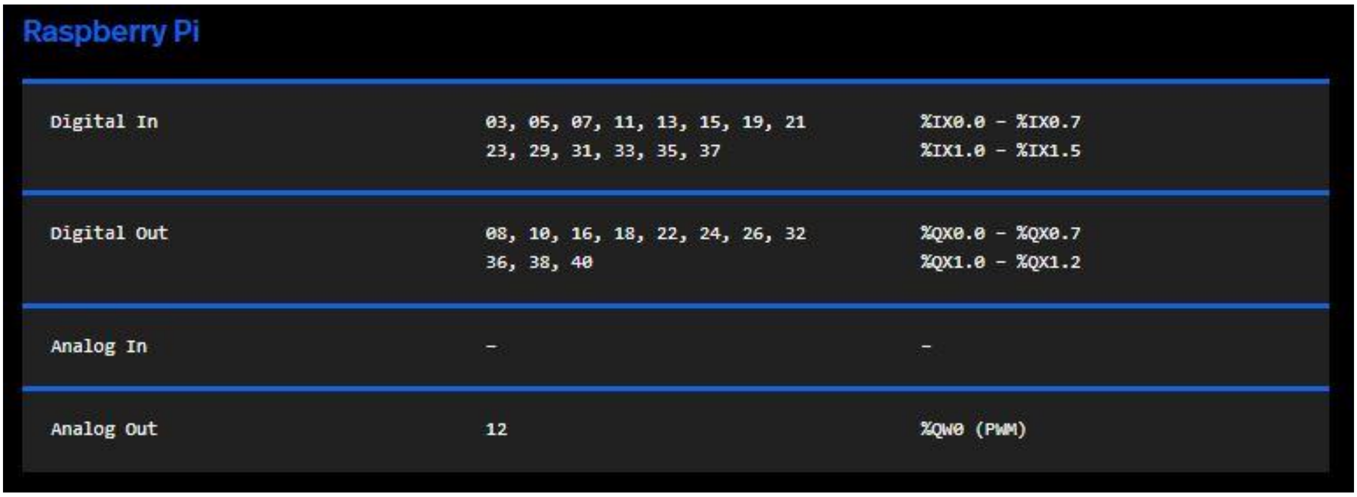
Input Registers (%IW100)
Start Address: 0 Size: 0

Holding Registers - Read (%IW100)
Start Address: 0 Size: 0

Holding Registers - Write (%QW100)
Start Address: 0 Size: 1

Device Name: Give a Suitable Name
Device Type: Generic Modbus TCP Device
Slave ID: 1
IP Address: 192.168.1.44 (Ip address of PiSlave1)
IP Port: 502 (default)
Discrete Inputs: Start=0 & size=14 (As Raspberry Pi has physically 14 ‘Digital In’ GPIO pins)
Coils: Start = 0 and Size=11 (As Raspberry Pi has physically 11 ‘Digital Out’ GPIO pins)

Note: Refer to OpenPLC Project documentation about Physical Addressing, Modbus Addressing, Input, Output And Memory Addressing and Slave Devices for further detail.



Raspberry Pi

Digital In	03, 05, 07, 11, 13, 15, 19, 21 23, 29, 31, 33, 35, 37	%IX0.0 - %IX0.7 %IX1.0 - %IX1.5
Digital Out	08, 10, 16, 18, 22, 24, 26, 32 36, 38, 40	%QX0.0 - %QX0.7 %QX1.0 - %QX1.2
Analog In	-	-
Analog Out	12	%QW0 (PWM)

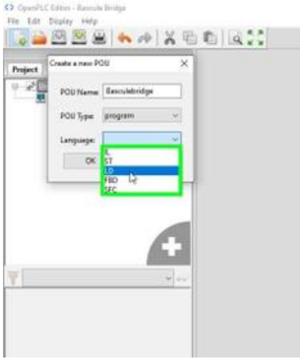
Fig 3: Raspberry Pi Physical Addressing for OpenPLC

4. Ladder Logic

The OpenPLC Editor is designed to support all five of the standard programming languages commonly used for PLCs. However, we chose to create our PLC program using ladder logic due to its user-friendly and straightforward nature, making it intuitive and easy to learn.

To start creating a PLC program for the OpenPLC Runtime, you can follow these steps:

1. Open the OpenPLC Editor software.
2. Click on the "File" tab located at the top of the window.
3. Select "New" from the dropdown menu that appears.
4. Create a new folder by selecting the "Create New Folder" option.
5. Once the new folder is created, select it, and click on "Select Folder" to confirm your selection.
6. On Create a new POU window, select Language as LD and click Ok.



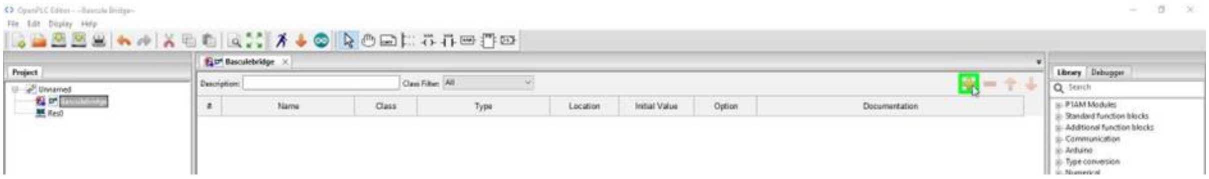
Create a new POU

POU Name: Basculebridge

POU Type: program

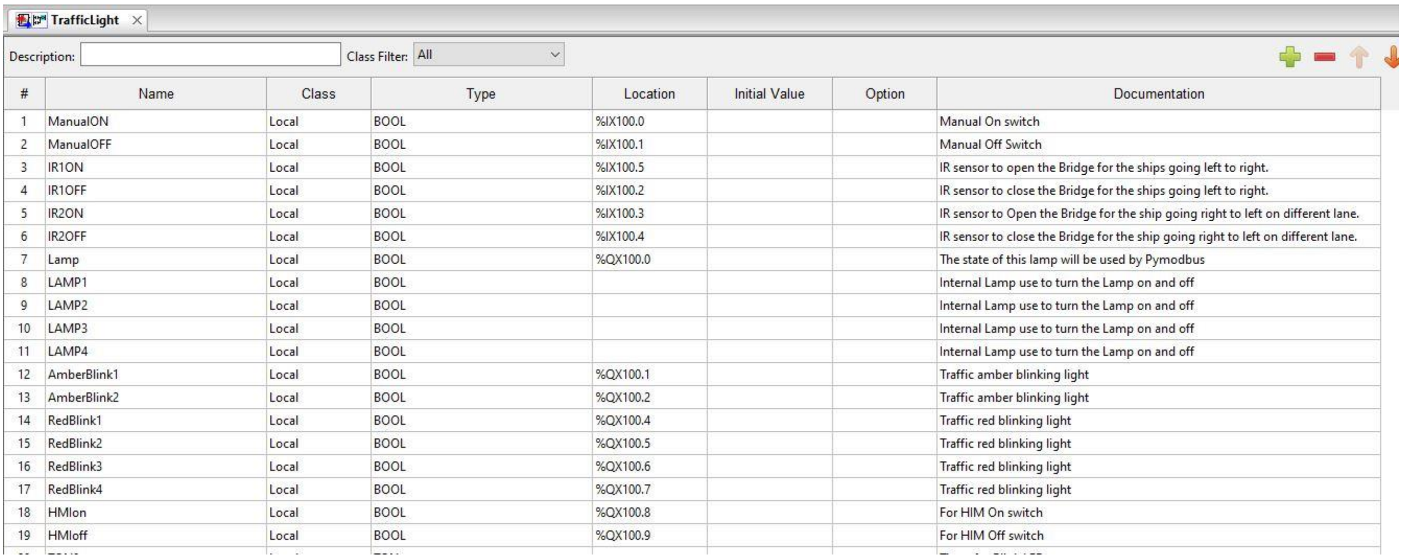
Language: LD

7. Now add the variables by clicking ‘Add variable’ button.



#	Name	Class	Type	Location	Initial Value	Option	Documentation

8. Add following variables:



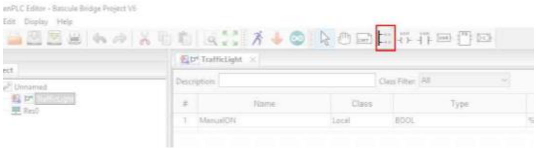
TrafficLight

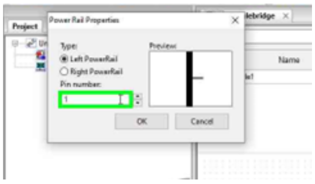
Description: Class Filter: All

#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	ManualON	Local	BOOL	%IX100.0			Manual On switch
2	ManualOFF	Local	BOOL	%IX100.1			Manual Off Switch
3	IR1ON	Local	BOOL	%IX100.5			IR sensor to open the Bridge for the ships going left to right.
4	IR1OFF	Local	BOOL	%IX100.2			IR sensor to close the Bridge for the ships going left to right.
5	IR2ON	Local	BOOL	%IX100.3			IR sensor to Open the Bridge for the ship going right to left on different lane.
6	IR2OFF	Local	BOOL	%IX100.4			IR sensor to close the Bridge for the ship going right to left on different lane.
7	Lamp	Local	BOOL	%QX100.0			The state of this lamp will be used by Pymodbus
8	LAMP1	Local	BOOL				Internal Lamp use to turn the Lamp on and off
9	LAMP2	Local	BOOL				Internal Lamp use to turn the Lamp on and off
10	LAMP3	Local	BOOL				Internal Lamp use to turn the Lamp on and off
11	LAMP4	Local	BOOL				Internal Lamp use to turn the Lamp on and off
12	AmberBlink1	Local	BOOL	%QX100.1			Traffic amber blinking light
13	AmberBlink2	Local	BOOL	%QX100.2			Traffic amber blinking light
14	RedBlink1	Local	BOOL	%QX100.4			Traffic red blinking light
15	RedBlink2	Local	BOOL	%QX100.5			Traffic red blinking light
16	RedBlink3	Local	BOOL	%QX100.6			Traffic red blinking light
17	RedBlink4	Local	BOOL	%QX100.7			Traffic red blinking light
18	HMIon	Local	BOOL	%QX100.8			For HIM On switch
19	HMIoff	Local	BOOL	%QX100.9			For HIM Off switch

(%IX---.- and %QX---.- address location correspond to the GPIO pins where these devices will be connected. Refer to OpenPLC Physical Addressing table, Modbus Addressing and Slave configuration for further information.)

9. Click ‘Create a new power rail’ radio button to create a power rail. You need to create left and right power rails.





Power Rail Properties

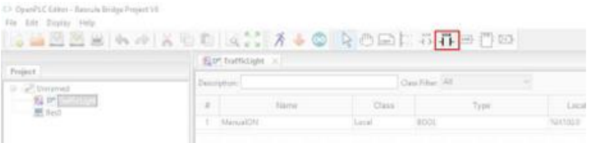
Type:

● Left PowerRail

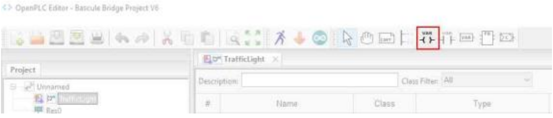
○ Right PowerRail

Pin number: 1

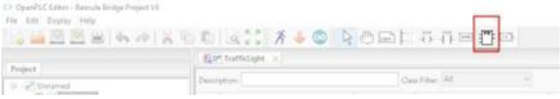
10. To create new contact, click ‘create new contact’ radio button.



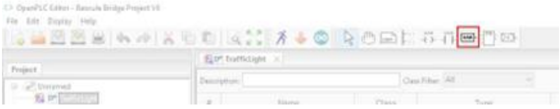
11. To create new coil, click ‘create new coil’ radio button.



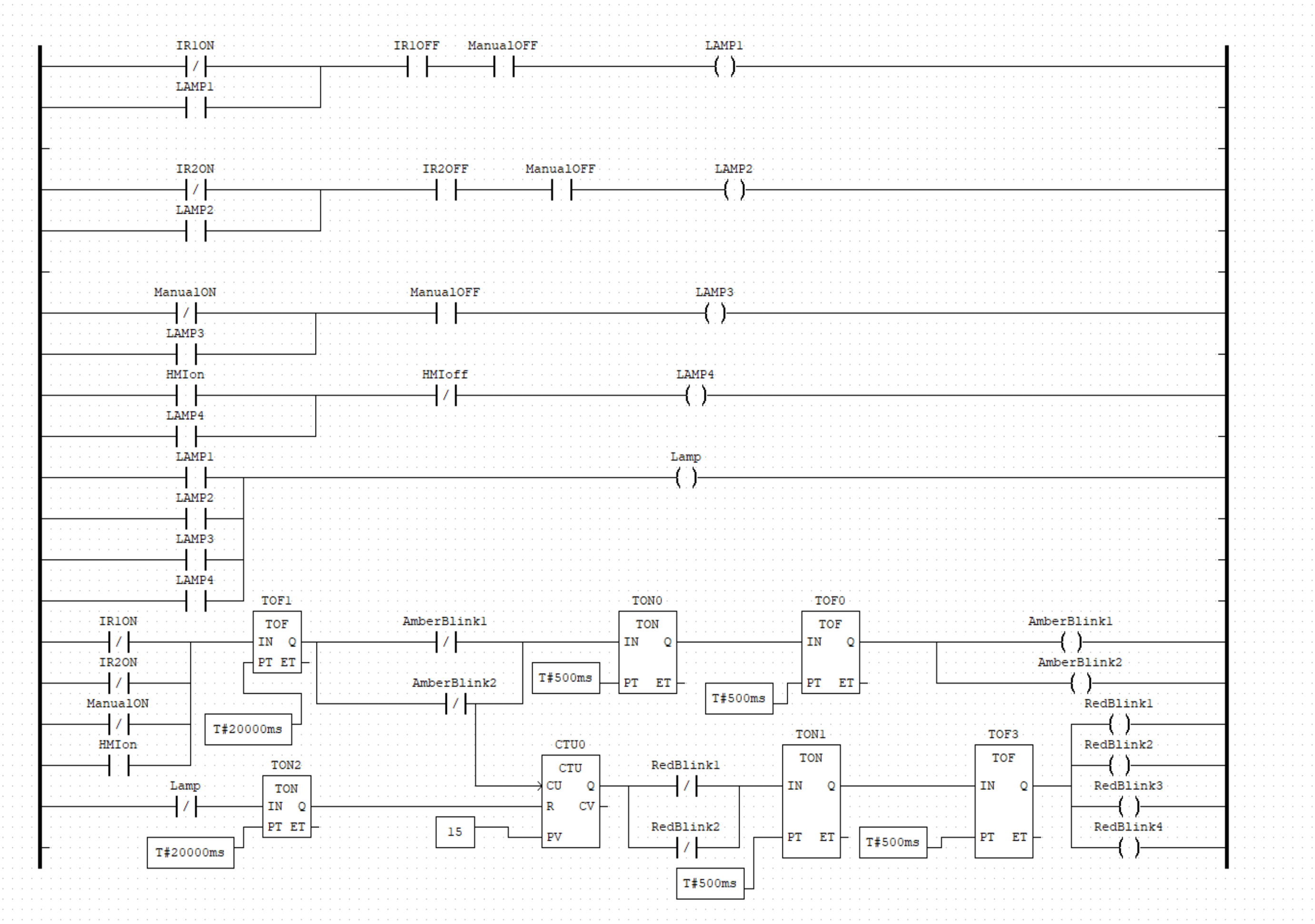
12. To create function block, click ‘create a new block’ radio button.



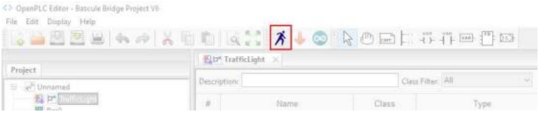
13. To create a new variable, click ‘create a new variable’ radio button.



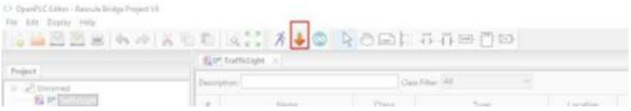
14. Using these tools create the ladder diagram as shown in the picture below.



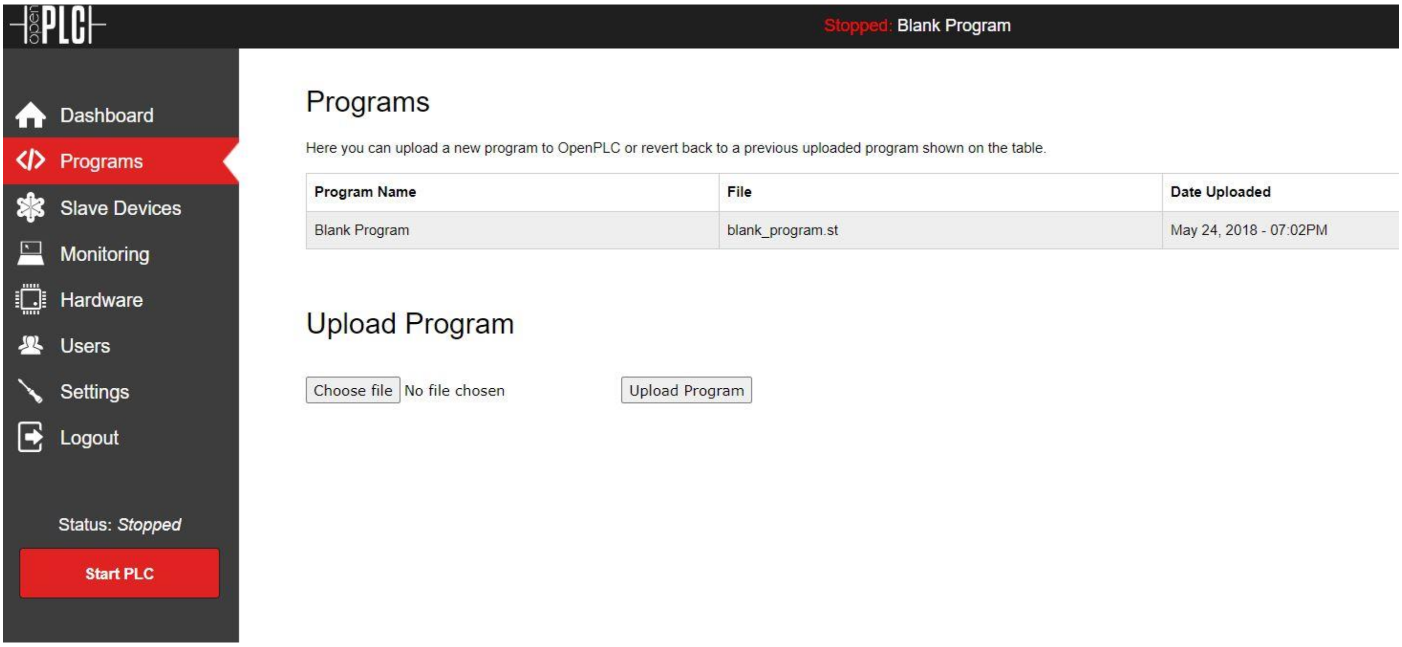
15. Once completed you can check the ladder logic by clicking ‘start PLC simulation’ radio button.



16. Finally download the PLC program for the OpenPLC Runtime by clicking ‘Generate program for OpenPLC Runtime’ radio button and save the file in desired location.



17. Access the OpenPLC Runtime web interface for PiMaster1 by opening a web browser and entering http://192.168.1.15:8080 in the address bar.
18. To add a new PLC program, navigate to the left-hand panel in the OpenPLC Editor and click on 'Programs'. From there, click on the 'Choose File' and select the PLC program file for OpenPLC Runtime you have recently created and click ‘Upload Program’.



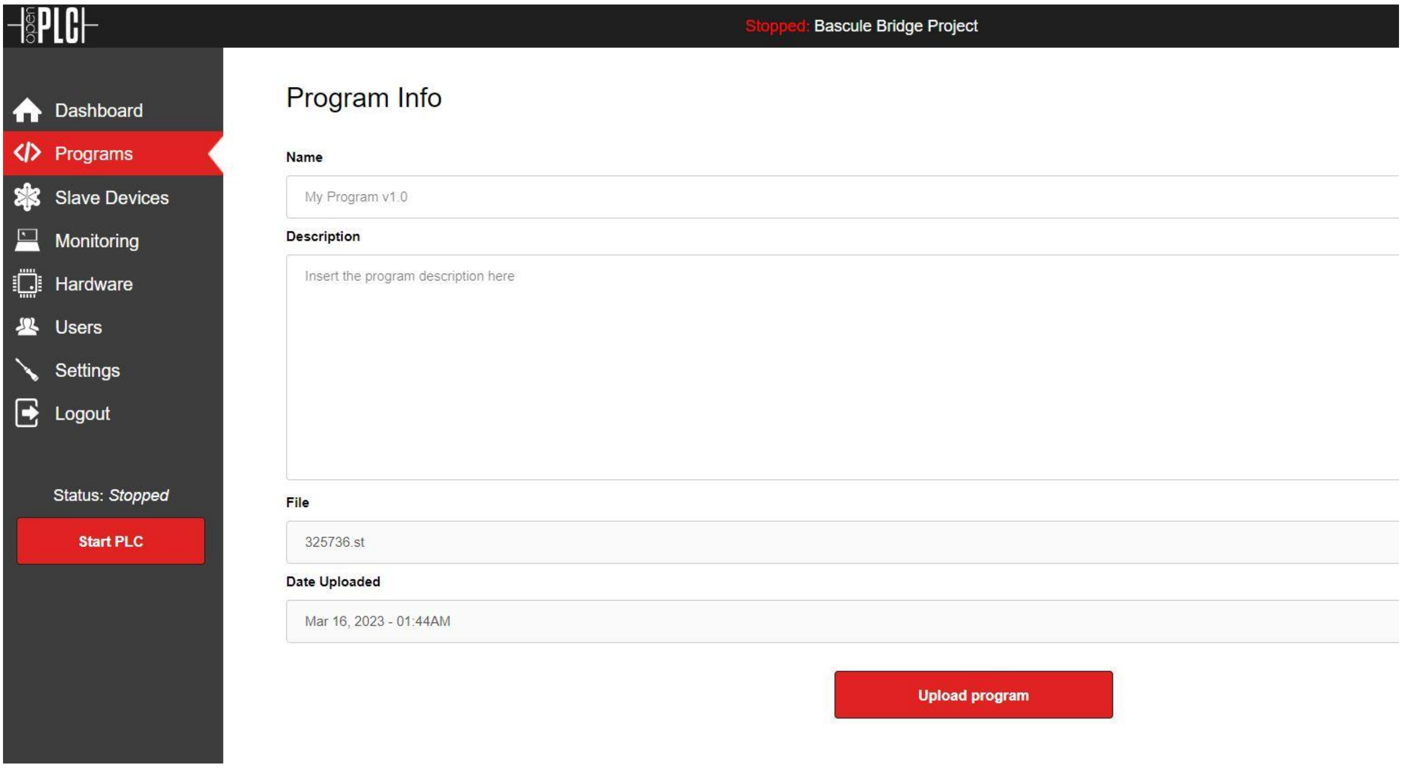
Programs

Here you can upload a new program to OpenPLC or revert back to a previous uploaded program shown on the table.

Program Name	File	Date Uploaded
Blank Program	blank_program.st	May 24, 2018 - 07:02PM

Upload Program

19. Next add relevant Name and Description and click ‘Upload Program’.



Program Info

Name: My Program v1.0

Description: Insert the program description here

File: 325736.st

Date Uploaded: Mar 16, 2023 - 01:44AM

Note: This ladder logic is designed to receive input signals from IR sensors, HMI, and manual switches and activate multiple LED lamps in response. Upon detection of a trigger event by either the IR sensor, manual switch, or switch from the HMI interface, the "Lamp" is turned on, and various timer and counter functions are employed to control the activation and deactivation of traffic amber and red lights. The state of the "lamp" is subsequently utilised by a Python code to initiate further actions.

5. Python Code

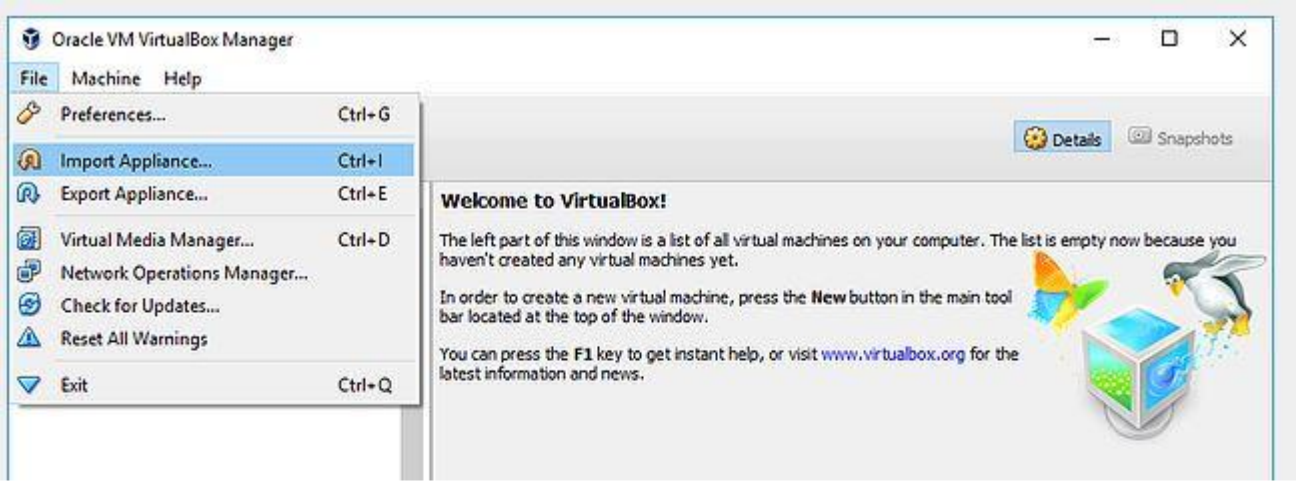
In this project python code is used to control the Step Motor 28BYJ-48, Micro Servo SG90, and Grove speaker. Copy the code ‘Bascule Bridge Python Code.py’ from the project GitHub page and upload it to PiMaster1. This code uses the Pymodbus library to read the coil status of the "Lamp" connected to RTU and RPi.GPIO library to interface with the Step Motor 28BYJ48, Micro Servo SG90, and Grove speaker. In this Python code, when the incoming ship triggers the IR sensors or the operator triggers the switch to change the coil state of the "Lamp" true, initially it will sound the alarm tone using the Grove Speaker, then the traffic barrier is closed using the Micro Servo SG90, and finally the bascule bridge is opened using Step Motor 28BYJ-48. Once the ship has crossed the bridge and triggered the second IR sensor on the other side or the operator has triggered the close switch, the coil state of the "Lamp" will change to False, the bridge will start to close, and the traffic barrier will rise.

6. HMI Build

This project involves using ScadaBR as the HMI. The HMI building process comprises two phases, namely, installing ScadaBR through a virtual machine and configuring the HMI.

Phase 1: Installing ScadaBR

1. Download VirtualBox from the official website.
2. Download ScadaBR virtual image from OpenPLC Gitlab.
3. Install VirtualBox by running the installer downloaded on step 1.
4. Run the VirtualBox and import ScadaBR image by clicking on ‘File’ and then ‘Import Appliance’.



Oracle VM VirtualBox Manager
File
Machine
Help
Preferences... Ctrl+G
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Virtual Media Manager... Ctrl+D
Network Operations Manager...
Check for Updates...
Reset All Warnings
Exit Ctrl+Q
Details
Snapshots

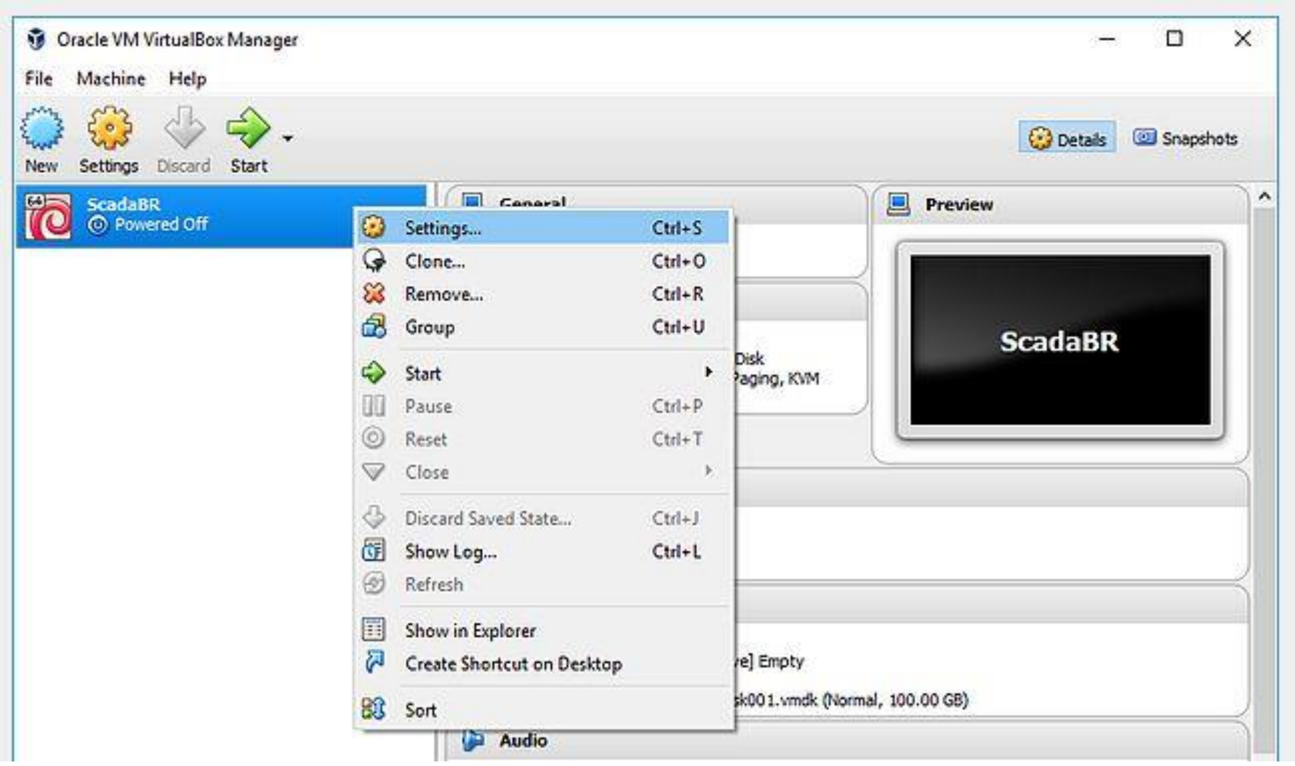
Welcome to VirtualBox!

The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

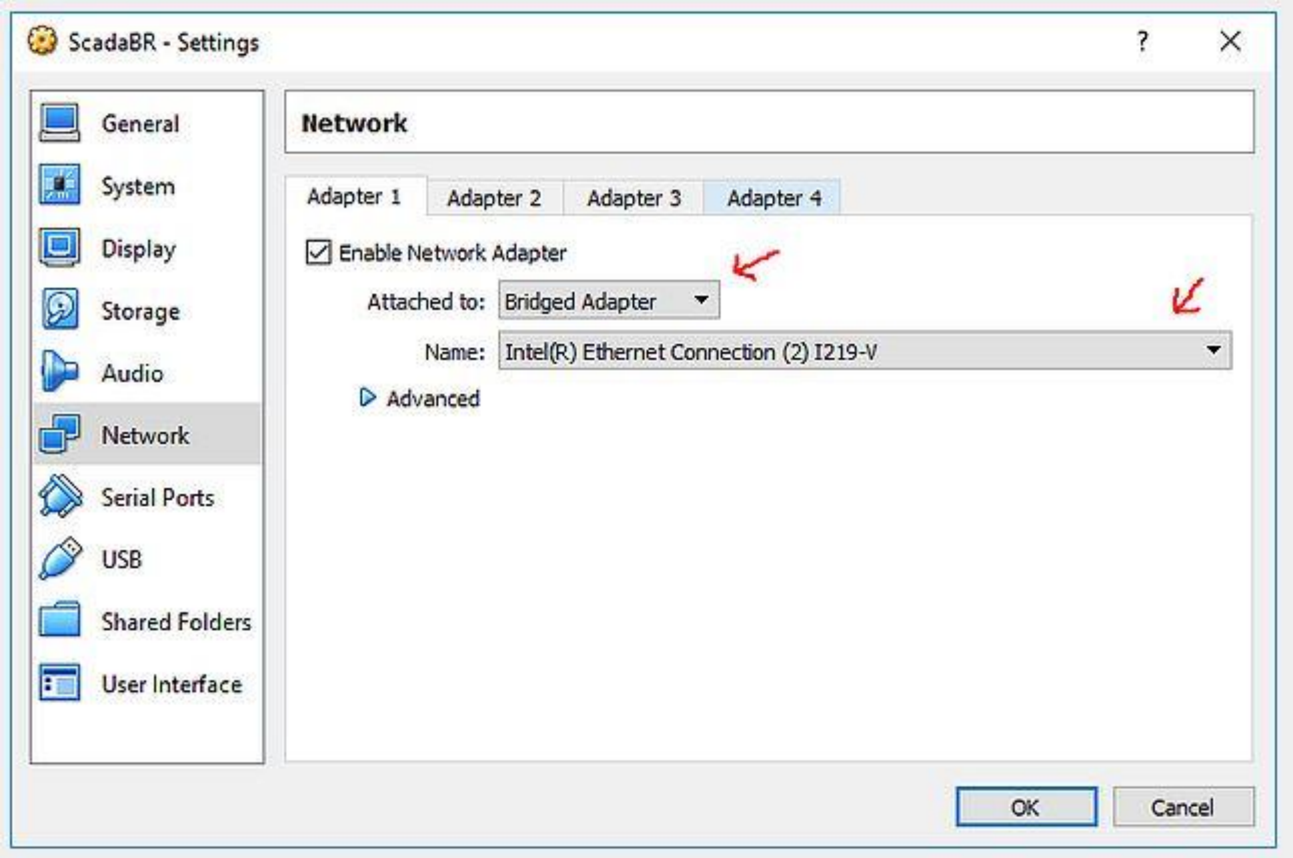
In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.

5. Click on the folder icon and select the ScadaBR.ova file downloaded on step 2.
6. Click on Next and then Import to load ScadaBR image into VirtualBox.
7. Before starting ScadaBR, you need to configure the network of your virtual machine so that ScadaBR can see the PLCs connected to your network. On VirtualBox main window, right-click on ScadaBR and select settings.



8. Go to Network and at the option “Attached to:” make sure that Bridged Adapter is selected. Then on “Name:” choose your network adapter. VirtualBox will replicate your network adapter for the virtual machine. Therefore, make sure that the adapter you select is the network adapter that is connected to your network. Once you’re done, click on OK to close this window.

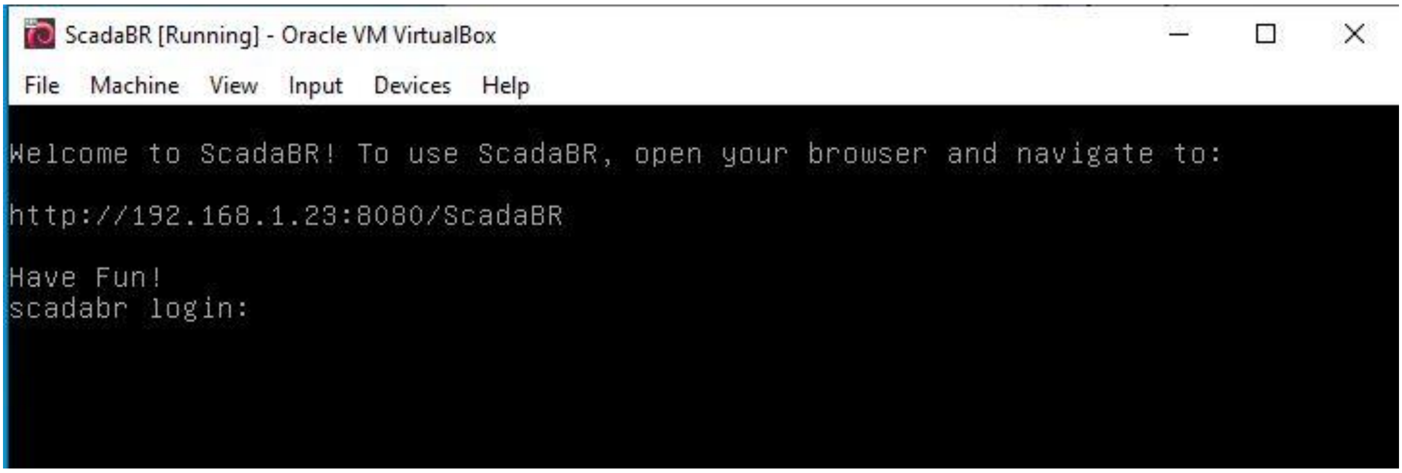


☑ Enable Network Adapter

Attached to: Bridged Adapter

Name: Intel(R) Ethernet Connection (2) I219-V

9. On the main VirtualBox screen click on Start to launch ScadaBR. It might take a few seconds to load. Once it is finally loaded, you will see this message on the screen:

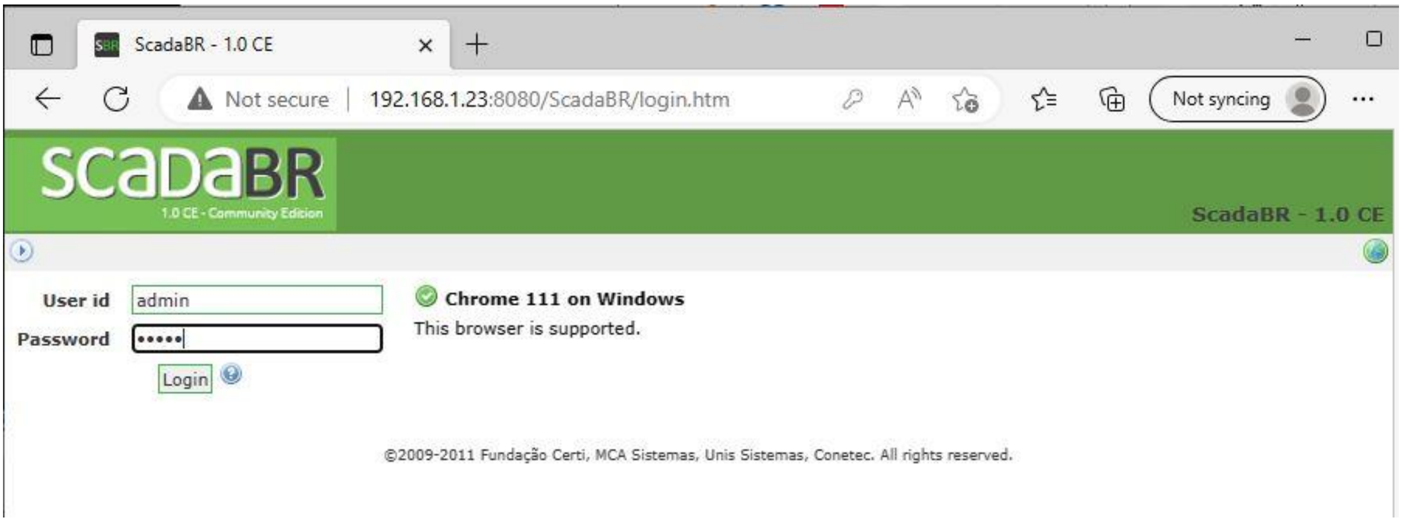


```
Welcome to ScadaBR! To use ScadaBR, open your browser and navigate to:

http://192.168.1.23:8080/ScadaBR

Have Fun!
scadabr login:
```

10. ScadaBR is a web application. To use it, you will need to open the browser on your computer and navigate to the address shown on the message. You will then be presented with a login page. Use the following credentials to login for the first time:

User id: admin

Password: admin



User id: admin

Password: •••••

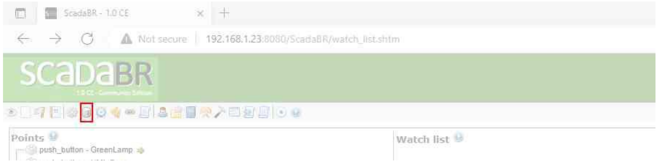
Chrome 111 on Windows

This browser is supported.

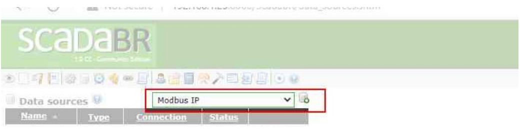
©2009-2011 Fundação Certi, MCA Sistemas, Unis Sistemas, Conetec. All rights reserved.

Phase 2: To create a functional HMI using ScadaBR and a Modbus TCP data source follow following steps:

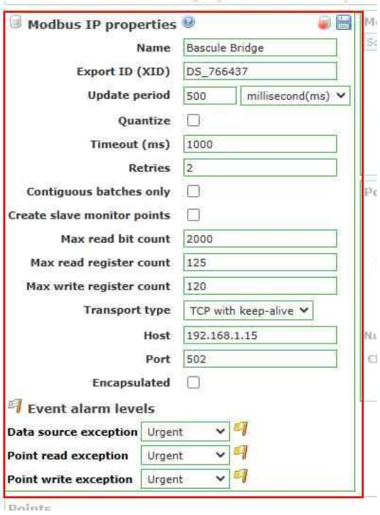
1. Log in to the ScadaBR web application to configure the HMI.
2. Click on the "Data Source" radio button to create a new data source.



3. Select 'Modbus IP' from the drop-down menu and click the 'Add' radio button to add a new Modbus TCP data source.



4. Define the Modbus IP properties, including updating the name, update period, timeout, transport type, host (IP address of the PLC, e.g., PiMaster1), port, and save the changes.



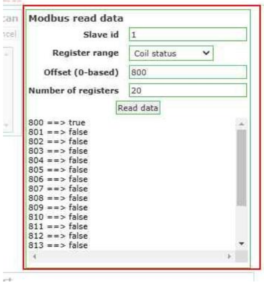
Modbus IP properties

Name	Bascule Bridge
Export ID (XID)	DS_766437
Update period	500 millisecond(ms)
Quantize	□
Timeout (ms)	1000
Retries	2
Contiguous batches only	□
Create slave monitor points	□
Max read bit count	2000
Max read register count	125
Max write register count	120
Transport type	TCP with keep-alive
Host	192.168.1.15
Port	502
Encapsulated	□

Event alarm levels

Data source exception	Urgent
Point read exception	Urgent
Point write exception	Urgent

5. Click on the 'Modbus Read Data' and set the offset to 800 and the number of registers to 20. Then click 'Read Data'.

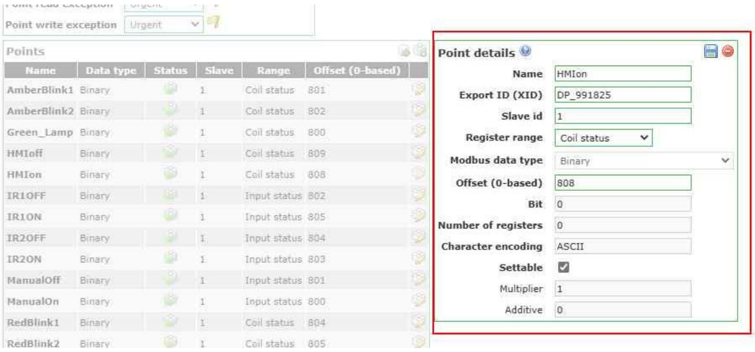


Modbus read data

Slave id	1
Register range	Coil status
Offset (0-based)	800
Number of registers	20

```
800 ==> true
801 ==> false
802 ==> false
803 ==> false
804 ==> false
805 ==> false
806 ==> false
807 ==> false
808 ==> false
809 ==> false
810 ==> false
811 ==> false
812 ==> false
813 ==> false
```

6. Add points to monitor the coil and input status. Click 'Add' and complete the point details.



Point write exception Urgent

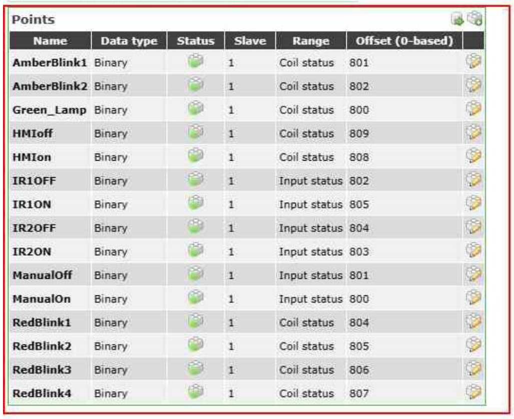
Points

Name	Data type	Status	Slave	Range	Offset (0-based)
AmberBlink1	Binary		1	Coil status	801
AmberBlink2	Binary		1	Coil status	802
Green_Lamp	Binary		1	Coil status	800
HMIoff	Binary		1	Coil status	809
HMIon	Binary		1	Coil status	808
IR1OFF	Binary		1	Input status	802
IR1ON	Binary		1	Input status	805
IR2OFF	Binary		1	Input status	804
IR2ON	Binary		1	Input status	803
ManualOff	Binary		1	Input status	801
ManualOn	Binary		1	Input status	800
RedBlink1	Binary		1	Coil status	804
RedBlink2	Binary		1	Coil status	805

Point details

Name	HMIon
Export ID (XID)	DP_991825
Slave id	1
Register range	Coil status
Modbus data type	Binary
Offset (0-based)	808
Bit	0
Number of registers	0
Character encoding	ASCII
Settable	☑
Multiplier	1
Additive	0

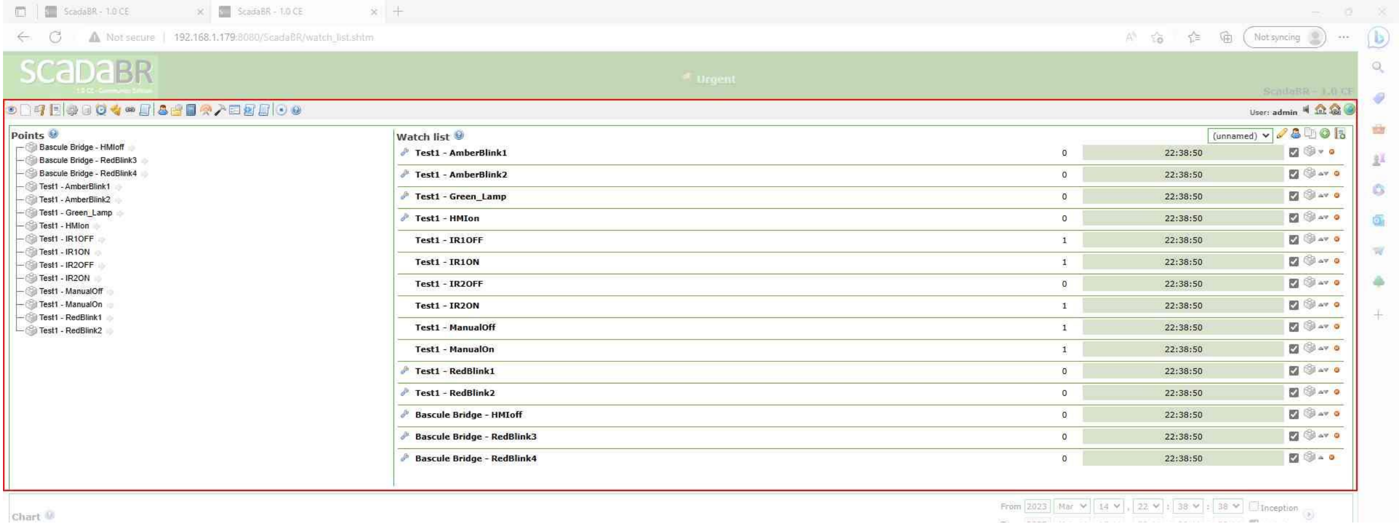
7. Add following Points:



Points

Name	Data type	Status	Slave	Range	Offset (0-based)
AmberBlink1	Binary		1	Coil status	801
AmberBlink2	Binary		1	Coil status	802
Green_Lamp	Binary		1	Coil status	800
HMIoff	Binary		1	Coil status	809
HMIon	Binary		1	Coil status	808
IR1OFF	Binary		1	Input status	802
IR1ON	Binary		1	Input status	805
IR2OFF	Binary		1	Input status	804
IR2ON	Binary		1	Input status	803
ManualOff	Binary		1	Input status	801
ManualOn	Binary		1	Input status	800
RedBlink1	Binary		1	Coil status	804
RedBlink2	Binary		1	Coil status	805
RedBlink3	Binary		1	Coil status	806
RedBlink4	Binary		1	Coil status	807

8. Now you can view the status of coil and inputs on ‘Watch list’.



ScadaBR - 1.0 CE

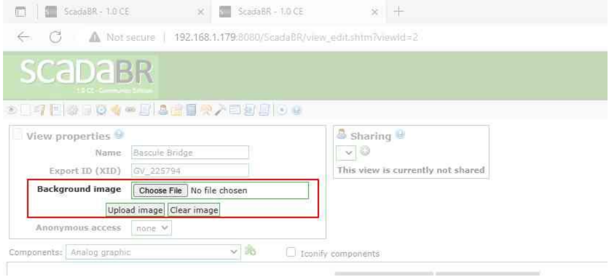
192.168.1.179:8080/ScadaBR/watch_list.shtm

Points
- Bascule Bridge - HMIoff
- Bascule Bridge - RedBlink3
- Bascule Bridge - RedBlink4
- Test1 - AmberBlink1
- Test1 - AmberBlink2
- Test1 - Green_Lamp
- Test1 - HMIon
- Test1 - IR1OFF
- Test1 - IR1ON
- Test1 - IR2OFF
- Test1 - IR2ON
- Test1 - ManualOff
- Test1 - ManualOn
- Test1 - RedBlink1
- Test1 - RedBlink2

Watch list

Test1 - AmberBlink1	0	22:38:50
Test1 - AmberBlink2	0	22:38:50
Test1 - Green_Lamp	0	22:38:50
Test1 - HMIon	0	22:38:50
Test1 - IR1OFF	1	22:38:50
Test1 - IR1ON	1	22:38:50
Test1 - IR2OFF	0	22:38:50
Test1 - IR2ON	1	22:38:50
Test1 - ManualOff	1	22:38:50
Test1 - ManualOn	1	22:38:50
Test1 - RedBlink1	0	22:38:50
Test1 - RedBlink2	0	22:38:50
Bascule Bridge - HMIoff	0	22:38:50
Bascule Bridge - RedBlink3	0	22:38:50
Bascule Bridge - RedBlink4	0	22:38:50

9. To create a graphical view, click on the "Graphic Views" radio button and select "New View." In the View Properties, give the view a name, and import any background image and upload it.



192.168.1.179:8080/ScadaBR/view_edit.shtm?viewId=2

View properties

Name: Bascule Bridge

Export ID (XID): GV_225794

Background image: Choose File No file chosen

Upload image | Clear image

Anonymous access: none

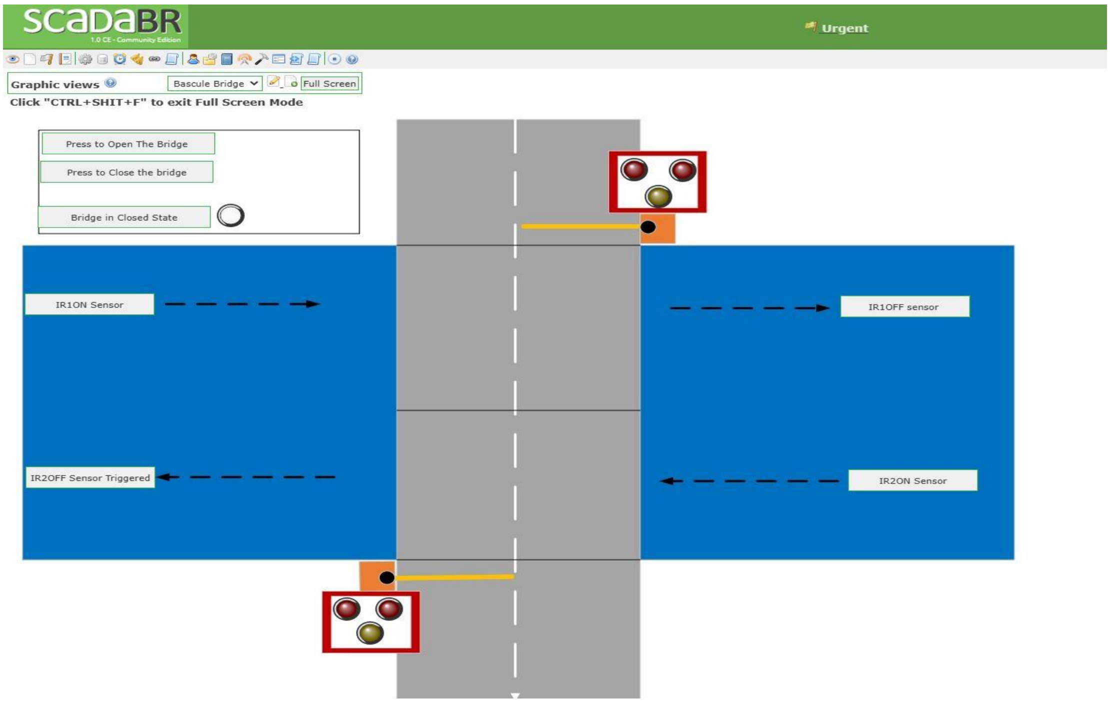
Sharing

This view is currently not shared

Components: Analog graphic

□ Iconify components

10. Select different components from the components dropdown menu, such as analogue graphics, binary graphics, buttons (write), etc. Add the selected components to the graphical view, and finally save the changes.



Graphic views: Bascule Bridge

Click "CTRL+SHIT+F" to exit Full Screen Mode

Fig 4. HMI graphical view.

7. 3D Printing

The 3D printer was used to print the model displayed below. If you're interested in creating your own, you can find the design for this model on the project's GitHub page. Alternatively, you can use FreeCAD, a free and open-source 3D parametric modelling software that supports a variety of modelling techniques, including sketch-based modelling, solid modelling, surface modelling, and mesh modelling, to design and print your own unique models. The bascule bridge 3D model presented here was created using FreeCAD.

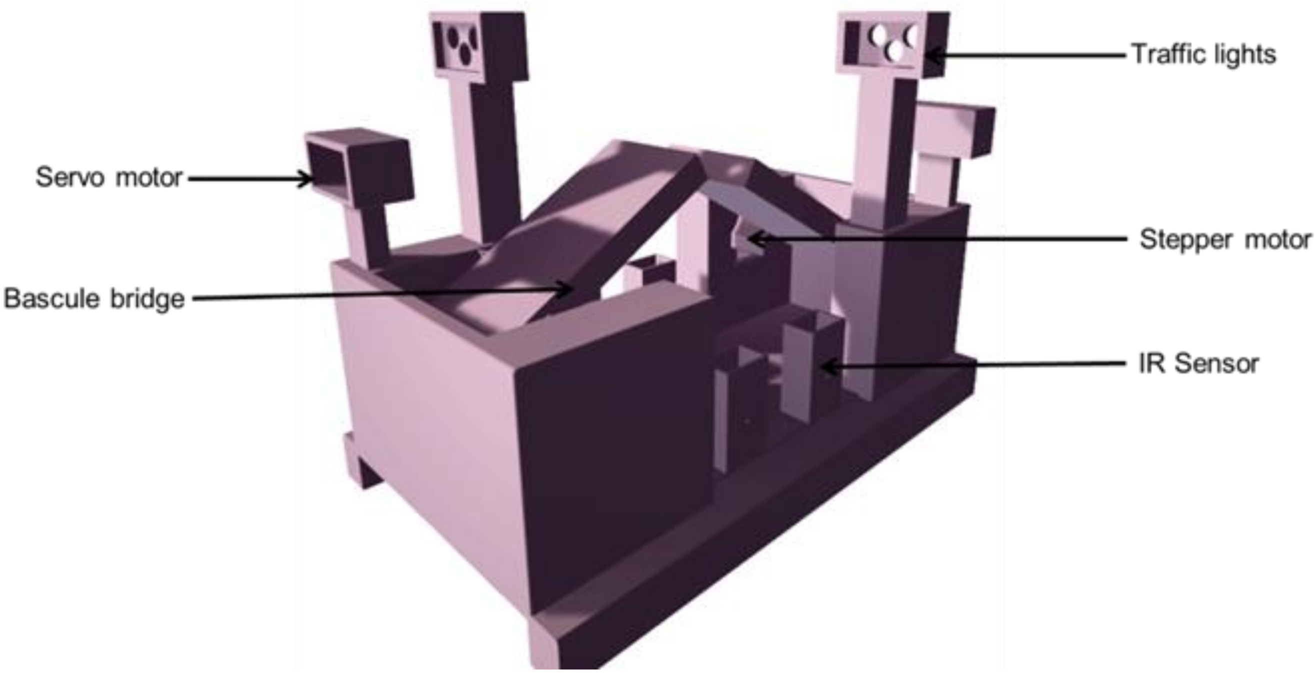


Fig 5. Bascule bridge system 3D design.

8. Assembly and Testing

To build the bridge control system, gather all the necessary components, such as the breadboard, power supply module, resistor, LED, jumper wires, push button, and other electronic accessory kit components. Follow the network diagram to connect all the components appropriately. After powering on the system, log in to the PiMaster1 OpenPLC Runtime web application, select the uploaded PLC program, and click on 'Start PLC.' Then, log in to the PiSlave1 OpenPLC Runtime web application and start the blank program by clicking on 'Start PLC.' Wait for the amber and red traffic lights to complete one cycle.

Once both traffic lights stop flashing, run the python code on PiMaster1. Log in to the ScadaBR web application and ensure that the data source is active and connected. Click on "Graphic Views" to observe real-time updates on the HMI. Test the bridge's operation as per the 'Flow Diagram' and confirm that all the components are working correctly. Check whether you can open and close the bridge using the IR sensors, manual push buttons, and switch from the HMI. If everything is working correctly, bridge is fully operation and ready for cyber training.

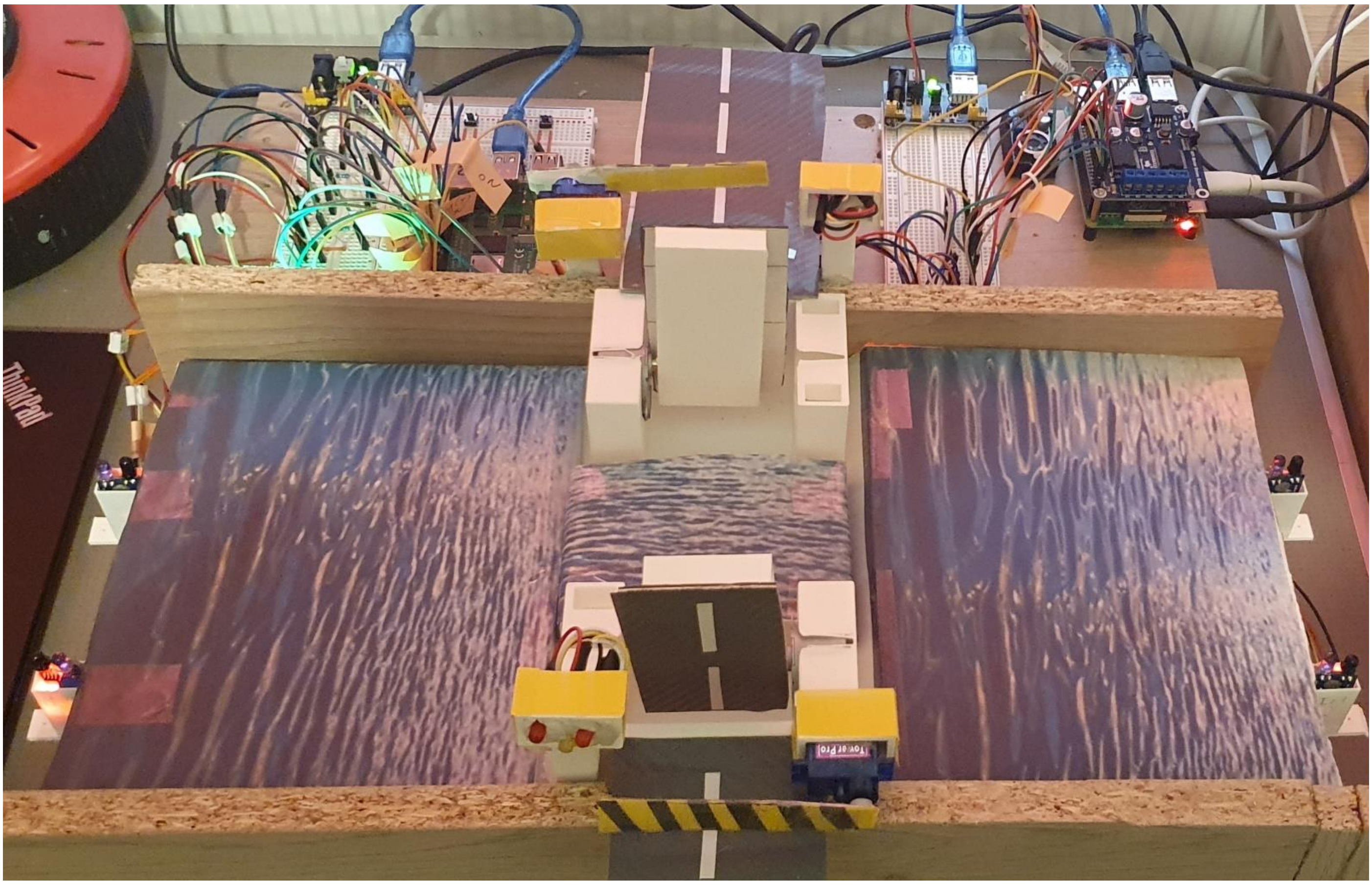


Fig 6. Final Model of Bascule Bridge.