Numato Lab

2024

Controlling Numato Lab Ethernet GPIO Modules using Python

Table of Contents

Contents

1. Introduction

In the age of automation and interconnectivity, the ability to control devices programmatically opens a world of possibilities. Python, with its simplicity and versatility, has become the language of choice for many developers seeking to interact with hardware effortlessly. Whether you're a hobbyist tinkering with electronics or a professional integrating devices into complex systems, Python provides a robust framework for device control.

This article aims to clarify the process of controlling Numato Lab Ethernet GPIO modules using Python, catering to beginners and seasoned developers alike.

2. Prerequisites

1. Python (less than version 3.13) (Download and Install Python)
2. pip version 6.x (Install pip)
3. telnetlib – open powershell, send command “pip install telnetlib”

3. Fundamentals of communicating to the device through Python

Once all the above prerequisites are installed, we can start writing the Python code.

The very first step is to import all the required libraries like telnetlib.

```
import sys
import telnetlib
```

Now, it’s time to initialize the variables required. In our case, the IP address of the module, username, and the password.

```
DeviceIP = "192.168.10.55"          #Device IP address
user = "admin"                      #Device Telnet user name
password = "admin"                  #Device Telnet password
```

Create a Telnet object using the ‘Telnet’ class from the ‘telnetlib’ module. This initializes a Telnet connection to the device specified by the IP address stored in the ‘DeviceIP’ variable. This will create a connection using default port 23.

```
telnet_obj = telnetlib.Telnet(DeviceIP)
```

Once the telnet connection is successful, read from the Telnet session to check for the prompt for credentials. Here, we will try to read for a specific delimiter “login” using the read_until() method.

Encode the username to ASCII and write when reading from the device is finished using the

write() method. Repeat the same for sending the password to the device. "\r\n" stands for CR – Carriage Return and LF – Line Feed(new line). This should be sent along with any data to the device.

```
telnet_obj.read_until(b"login")
telnet_obj.write(user.encode('ascii') + b"\r\n")

telnet_obj.read_until(b"Password: ")
telnet_obj.write(password.encode('ascii') + b"\r\n")
```

Now we will check if the credentials sent to the device matched and the login attempt was a success.

```
if b"successfully" in log_result:
    print("\nLogged in successfully... Connected to device", DeviceIP, "\n")
```

If it is a success, it will print the "Logged in successfully" message with the module's IP address.
Otherwise, if the device denies the login attempt, it will show the "Login failed" message, close the Telnet session using the close() method, and exit from the program.

```
elif "denied" in log_result:
    print("Login failed!!!! Please check login credentials or Device IP Address\n\n")
telnet_obj.close()
exit(0)
```

Let's see how to send commands and read responses from the device once the login attempt is successful.

```
telnet_obj.write(b"gpio read 0\r\n")
```

The above line sends the command 'gpio read 0' to the device using the write() method.

Now we will read using the read_until() method (delimiter is ">") and store the result in a variable named 'response' for future operations.

```
response = telnet_obj.read_until(b">")
```

It is time to print and view the response.

```
print("GPIO #0 Read : ", response.decode()[0])
```

The GPIO status will be printed with the help of the above statement. Decoding the response makes it easy to read.

Finally, the opened Telnet connection can be closed using the close() method.

```
telnet_obj.close()
```

Now, you are ready to write the sample code for Numato Lab Ethernet GPIO modules

4. Python Sample Code and Result

Create a new Python file and add the following complete Python code.
Before executing the program, change the IP address and credentials as required.
This will open a Telnet connection, login to the module, send commands and receive responses. Any other command described in the product user manual can be sent to the device the same way.

```
#Import required libraries
import sys
import telnetlib
import time

#Initialize variables
DeviceIP = "192.168.10.129"        #Device IP address
user = "admin"                     #Device Telnet username
password = "admin"                 #Device Telnet password

#Create a new TELNET object
telnet_obj = telnetlib.Telnet(DeviceIP)

#Connect to the device using the credentials provided
#Wait for the login prompt from the device and enter username when prompted
telnet_obj.read_until(b"login")
telnet_obj.write(user.encode('ascii') + b"\r\n")

telnet_obj.read_until(b"Password: ")
telnet_obj.write(password.encode('ascii') + b"\r\n")

log_result = telnet_obj.read_until(b"successfully\r\n")
telnet_obj.read_until(b">")

#Check if login attempt was successful
if b"successfully" in log_result:
    print("\nLogged in successfully... Connected to device", DeviceIP, "\n")

elif "denied" in log_result:
    print("Login failed!!!! Please check login credentials or Device IP Address\n\n")
    telnet_obj.close()
    exit(0)
```

```
#Read GPIO #0 status
telnet_obj.write(b"gpio read 0\r\n")
response = telnet_obj.read_until(b">")
print("GPIO #0 Read : ", response.decode()[0])
time.sleep(1)

#Set GPIO 0 to High
telnet_obj.write(b"gpio set 0\r\n")
telnet_obj.read_until(b">")
print("\nSet GPIO 0")
time.sleep(1)

#Set GPIO 0 to Low
telnet_obj.write(b"gpio clear 0\r\n")
telnet_obj.read_until(b">")
print("\nClear GPIO 0\n")
time.sleep(1)

#Read GPIO 0 status
telnet_obj.write(b"gpio read 0\r\n")
response = telnet_obj.read_until(b">")
print("GPIO #0 Read : ", response.decode()[0])

#Finally close the TELNET session before exiting
telnet_obj.close()
```

The result is as shown below: