

Journées AlpesVIEW 2021

Faire du transfert de fichiers en SFTP avec LabVIEW

Sabine DOUILLET





Transférer des données en utilisant le protocole SSH, supporté en natif sur MacOS, Windows Raspberry Pi OS ...mais pas LabVIEW!

Programme LabVIEW sur PC ou Mac



Raspberry



SSH

SFTP

Capteur

Capteur

Capteur

Capteur

Actionneur



LabVIEW Tools for SSH and SFTP

Updated May 28, 2021



Issue Details

- I want to transfer files in LabVIEW using SFTP, but I cannot find any libraries to implement this.
- I want to communicate with a remote client via SSH using LabVIEW, but I cannot find any tools to implement this.

Solution

Secure File Transfer Protocol (SFTP) employs Secure Shell (SSH) protocol to provide file access, file transfer and file management over any reliable data stream.

NI doesn't currently have any SFTP or SSH software of its own, but [Labvolution \(NI Alliance partner\)](#) and [LABWERX](#) do. Labvolution provides [LabSFTP](#) and [LabSSH](#) library for free. LABWERX provides [LabSSH](#), which is available for purchase.

Disclaimer: Third-Party Add-Ons for LabVIEW are offered by independent third-party providers who are solely responsible for such products. NI shall have no responsibility whatsoever for the performance of the Third-Party Add-Ons.



SFTP File Transfer for LabVIEW

📅 April 11, 2016 👤 GregPayne 📁 LabVIEW, Linux, SFTP

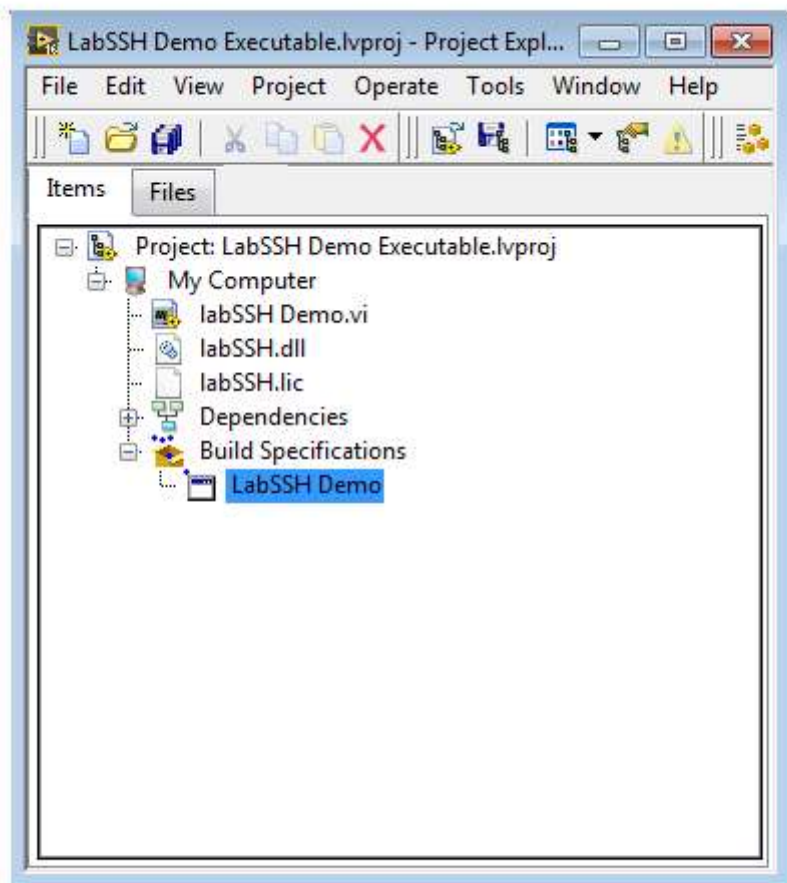
Please note that this project is no longer being supported and does not support some of the newer connection methods.

Leading on from the [LabSSH Library](#) that I shared a while back, I have also created a LabSFTP LabVIEW library.

From [Wikipedia](#), SFTP can be described as follows:

In computing, the SSH File Transfer Protocol (also Secure File Transfer Protocol, or SFTP) is a network protocol that provides file access, file transfer, and file management over any reliable data stream.





LABWERX LabSSH

Basic

\$249

Unlimited Email Support

Choose a License:

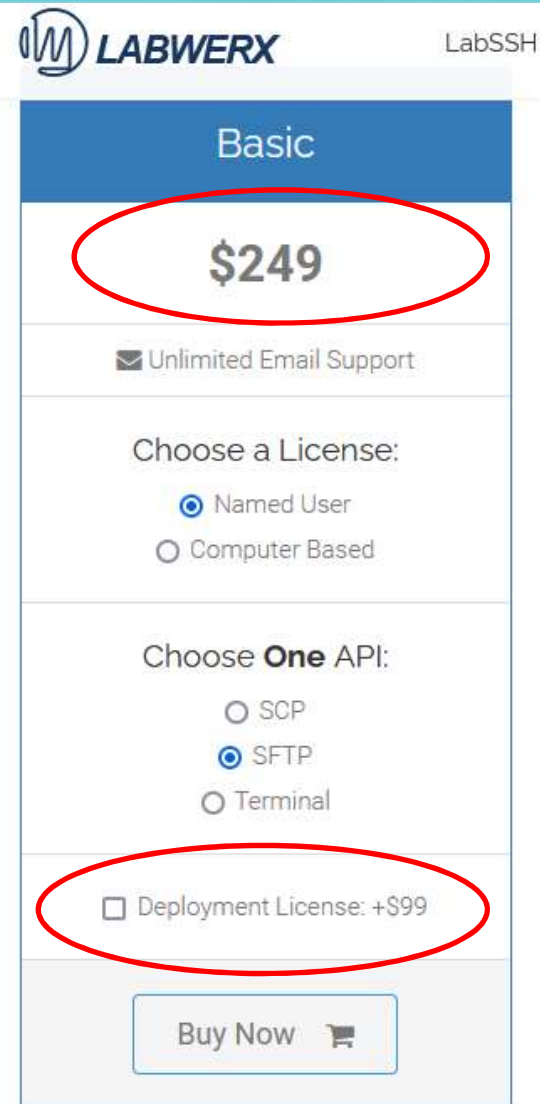
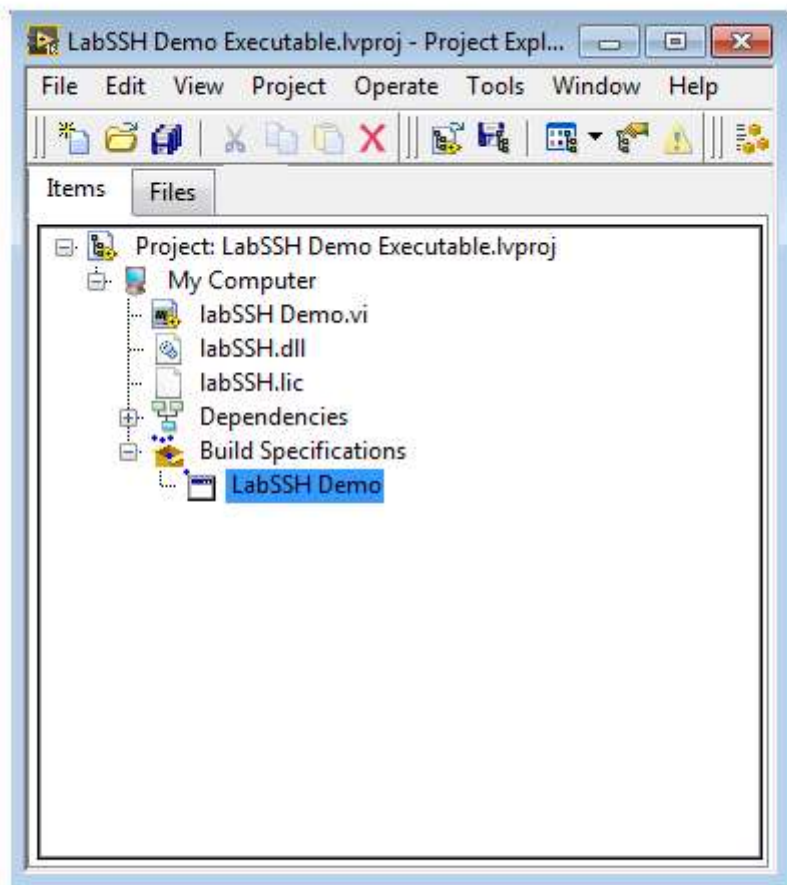
Named User
 Computer Based

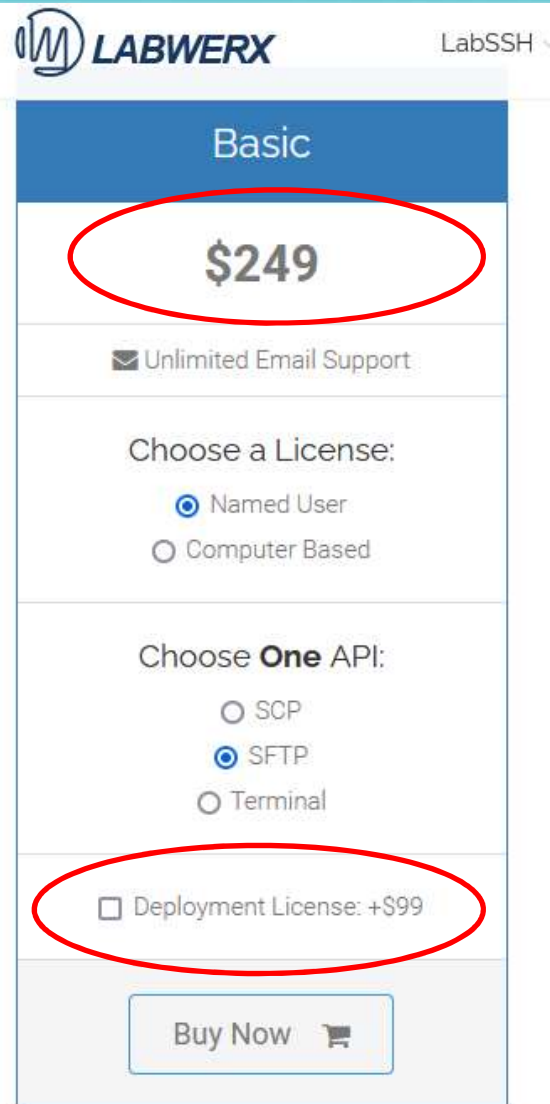
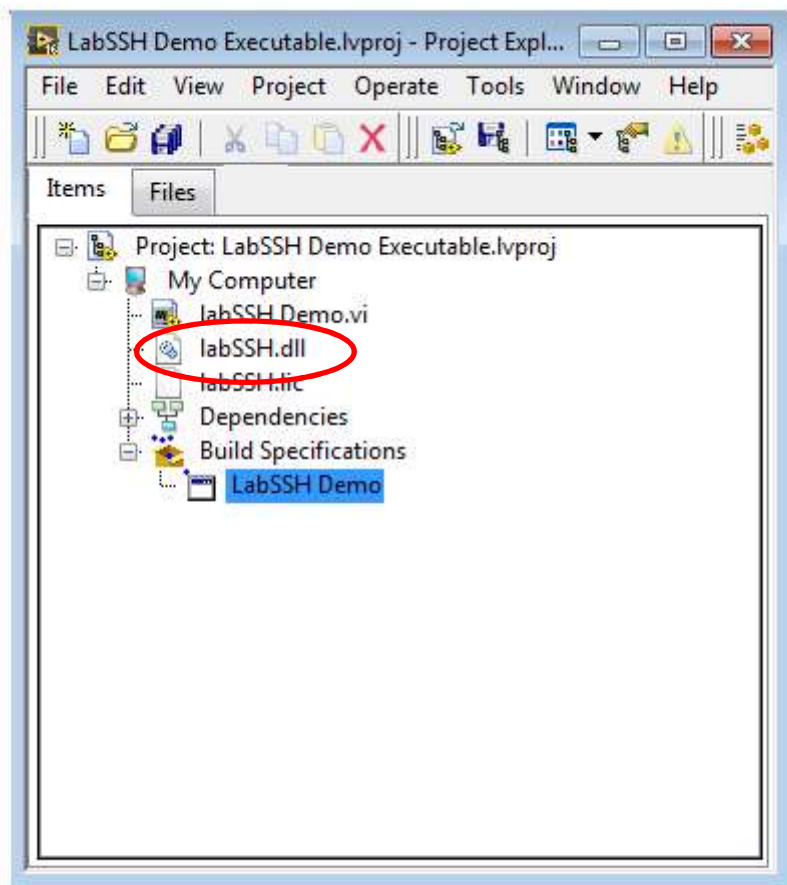
Choose **One** API:

SCP
 SFTP
 Terminal

Deployment License: +\$99

[Buy Now](#)

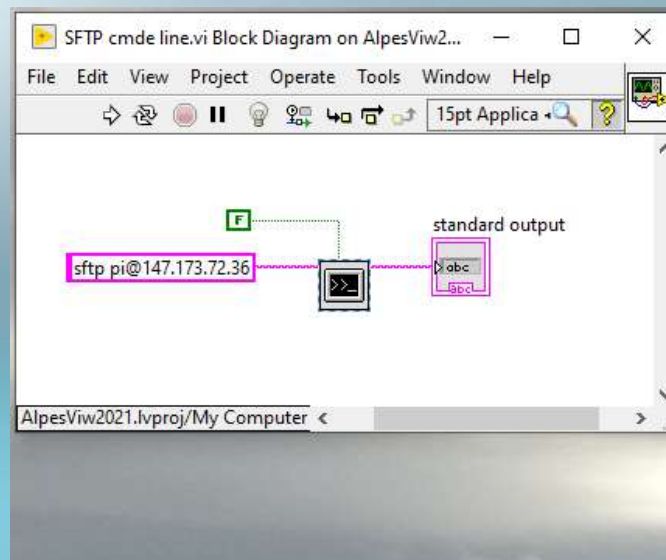






```

C:\Users\sabine.douillet>sftp pi@147.173.72.36
pi@147.173.72.36's password:
Connected to 147.173.72.36.
sftp> cd Desktop/Enregistrement
sftp> get f_temperatures.txt
Fetching /home/pi/Desktop/Enregistrement/f_temperatures.txt to f_temperatures.txt
/home/pi/Desktop/Enregistrement/f_temperatures.txt
 100% 157KB  5.0MB/s  00:00
sftp>
  
```



Context Help

System Exec.vi

wait until completion? (T)

command line

standard input

working directory

error in (no error)

run minimized? (F)

standard output

standard error

return code




error out

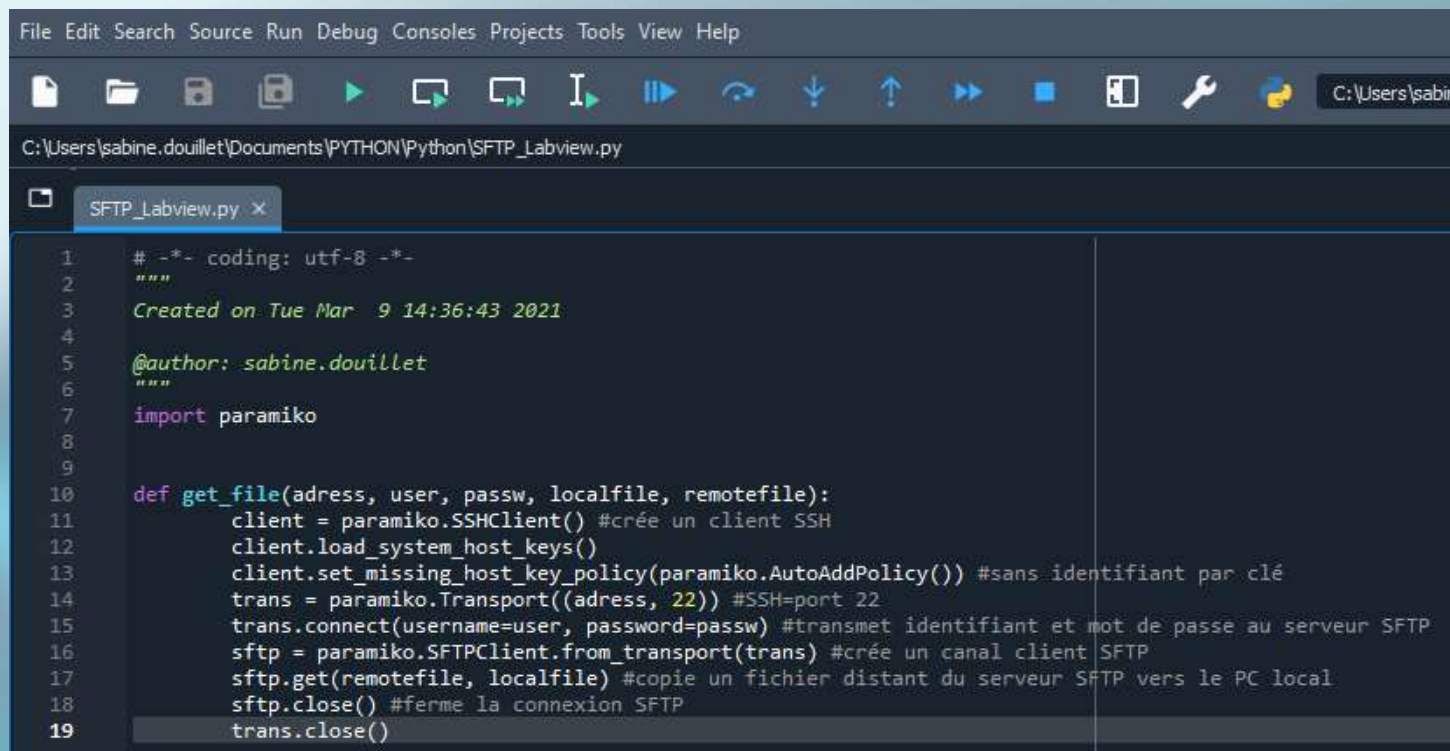
Executes a system command. Use the System Exec VI to execute or launch other Windows-based applications, command-line applications, (Windows) batch files, or (macOS and Linux) script files from within VIs. With the System Exec VI, you can include any parameters within your command string that the executing command supports.

Terminal Data Type
 return code (long [32-bit integer (-2147483648 to 2147483647)])

[Detailed help](#)



-  Il existe une bibliothèque Python (gratuite) qui gère le protocole SSH (et SFTP), elle s'appelle **Paramiko** (elle est incluse dans le package Anaconda et dernières versions de Spyder sinon sur Paramiko.org)
-  On peut intégrer facilement du code Python dans LabVIEW,
-  Alors allons-y ...



```

File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\sabine.douillet\Documents\PYTHON\Python\SFTP_Labview.py
SFTP_Labview.py x
1  # -*- coding: utf-8 -*-
2  """
3  Created on Tue Mar  9 14:36:43 2021
4
5  @author: sabine.douillet
6  """
7  import paramiko
8
9
10 def get_file(adress, user, passw, localfile, remotefile):
11     client = paramiko.SSHClient() #crée un client SSH
12     client.load_system_host_keys()
13     client.set_missing_host_key_policy(paramiko.AutoAddPolicy()) #sans identifiant par clé
14     trans = paramiko.Transport((adress, 22)) #SSH=port 22
15     trans.connect(username=user, password=passw) #transmet identifiant et mot de passe au serveur SFTP
16     sftp = paramiko.SFTPClient.from_transport(trans) #crée un canal client SFTP
17     sftp.get(remotefile, localfile) #copie un fichier distant du serveur SFTP vers le PC local
18     sftp.close() #ferme la connexion SFTP
19     trans.close()
    
```



The screenshot displays the LabVIEW interface with a Python Node configuration window and a block diagram. The Python Node configuration window shows the following settings:

- Python version:** 3.7
- module path:** Parh
- function name:** get_file
- return type:** abc
- input parameter:** pi
- input parameter:** *****

The block diagram includes the following components and connections:

- Open Python Session:** Configured with Python version 3.7.
- Python Node:** Receives inputs from the Open Python Session and the configuration window. It outputs a session ID (abc) and a return value (a 2x2 grid of abc).
- Close Python Session:** Receives the session ID from the Python Node.
- File Path:** A text box containing "C:\Users\sabine.douillet\Desktop\data_Raspberry.txt".
- File Download:** A text box containing "Desktop/Enregistrement/f_temperatures.txt".
- File Transfer:** A text box containing "fichier téléchargé" with a sub-text box containing "abc".
- Control Elements:** Two "Enabled" dropdown menus and a "pi" constant.

Two callout boxes provide additional information:

- Left Callout:** Explains the "Python version" block, showing "session out" and "error out" outputs. It includes a search bar with "Add to the block diagram" and "Find on the palette" buttons. Below it, a text box states: "Python version specifies the version of Python in which the Python session runs. This function supports Python of version 2.7 and 3.6 only. Although unsupported versions might work with..."
- Right Callout:** Explains the "Python Node" block, showing "session in", "module path", "function name", "error in (no error)", "return type", and "input parameter" inputs. It includes a "Detailed help" link and a note: "utilise la library pour tout ce q".

The status bar at the bottom left shows "AlpesViv2021.lvproj/My Computer".

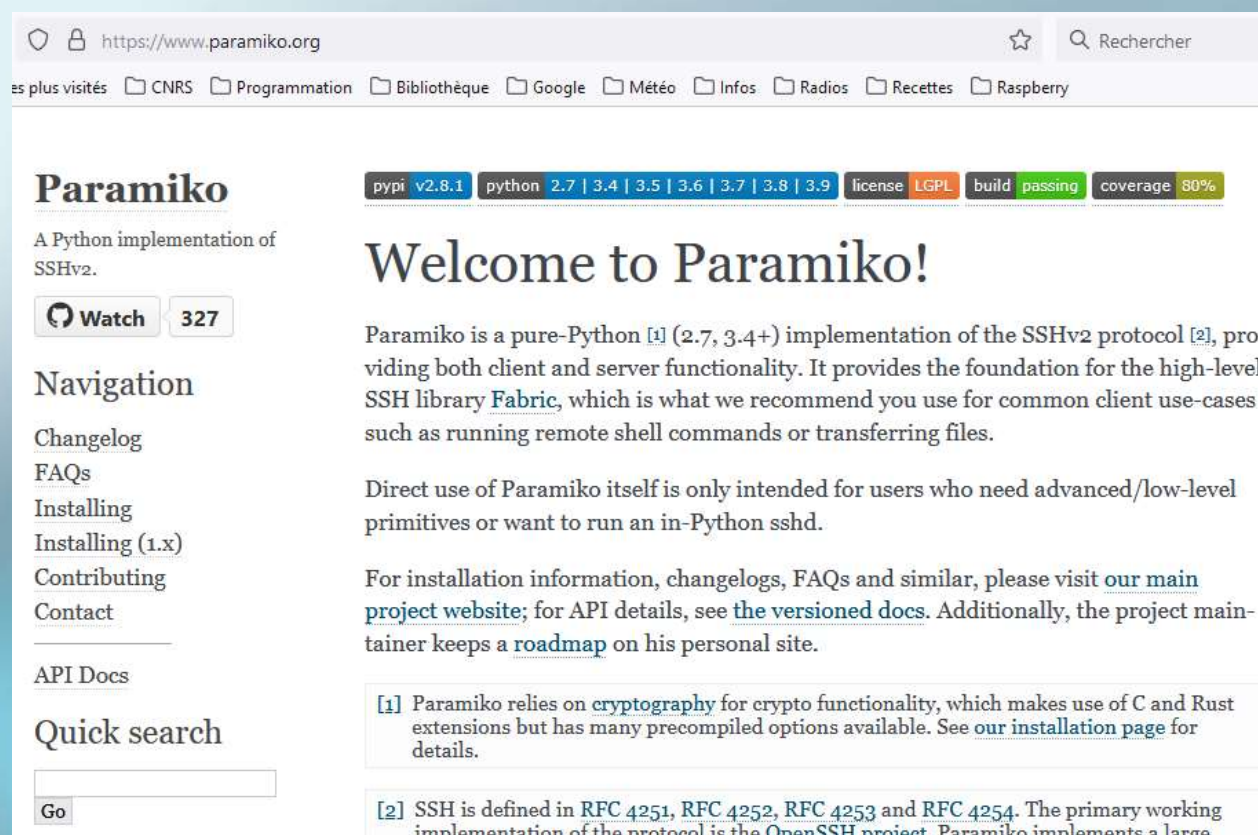


Modifier le Path

The screenshot shows a Windows desktop with several windows open. In the foreground, the 'Variables d'environnement' (Environment Variables) dialog box is open, displaying a list of system variables. The 'Path' variable is selected, and a secondary dialog box 'Modifier la variable système' (Modify system variable) is open over it. This secondary dialog has 'Path' in the 'Nom de la variable' field and a long file path in the 'Valeur de la variable' field. The path is: `ation\visa\WinNT\Bin;C:\Users\sabine.douillet\AppData\Local\Programs\Spyder\Python;`. The 'Variables d'environnement' dialog also shows other variables like OS, Path, and PROCESSOR_ARCHITECTURE. In the background, a terminal window shows a command prompt with the user 'sabine.douillet' and a system properties window is partially visible.



Installer, utiliser Paramiko



The screenshot shows the Paramiko website homepage. At the top, there's a navigation bar with links for 'es plus visités', 'CNRS', 'Programmation', 'Bibliothèque', 'Google', 'Météo', 'Infos', 'Radios', 'Recettes', and 'Raspberry'. The main content area features the Paramiko logo, a description 'A Python implementation of SSHv2.', and a 'Watch' button with '327' notifications. A 'Navigation' sidebar lists links for 'Changelog', 'FAQs', 'Installing', 'Installing (1.x)', 'Contributing', and 'Contact'. Below this is an 'API Docs' section and a 'Quick search' box. The main text area has a large heading 'Welcome to Paramiko!' followed by a paragraph describing the library as a pure-Python implementation of the SSHv2 protocol. It also includes a direct use warning and installation information. At the bottom, there are two footnotes: [1] explaining the use of cryptography and [2] defining SSH according to RFCs.



Plus

- Solution gratuite, multiplateforme et pérenne
- Facile à mettre en œuvre même avec des connaissances réduites de Python

Moins

- Si on distribue notre application, en plus de l'application installée grâce au Builder de LabVIEW, il faudra aussi installer Python, la Librairie Paramiko et paramétrer le Path du système hôte



Ne pas se priver d'intégrer des morceaux de code Python (bcp plus de codes Python que de VI partagés sur Internet)



Vous arrive-t-il d'acheter des modules complémentaires à LabVIEW (hors driver pour matériel spécifique) à des partenaires NI?



Votre expérience? Support, pérennité...



Merci

