ET0735 - DEVOPS FOR AIOT
SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING, SINGAPORE POLYTECHNIC

**LABORATORY 4: INTRODUCTION TO RASPBERRY PI**

## Objectives

By the end of the laboratory, students will be able to
- Introduction to the Raspberry Pi
- Configure the Raspberry Pi in "Headless Mode"
- Adapt Raspberry Pi OS configuration to auto-start a Python script

## Activities
- Introduction to Raspberry Pi and Development board hardware
- Setting up remote connection to Raspberry Pi
- Configure PyCharm remote debug session with Raspberry Pi

## Review
- Raspberry Pi is configured in "Headless Mode"
- PyCharm is configured to remotely debug and run Python code on the Raspberry Pi
- Successfully run a Python demo application on the Raspberry Pi Development board to test the LED, LCD, Buzzer, Servo Motor and DC Motor

## Equipment:
- Windows OS laptop
- Raspberry Pi + Development Board
- USB-C 5 volt power adapter
- 9 volt power adapter
- USB to Ethernet adapter

# 1 Setting up of Remote Connection to Raspberry Pi

1.1 To develop applications on the Raspberry Pi, the first step is to connect remotely from your laptop to the Raspberry Pi.

1.2 Power up your Raspberry Pi and the Development Board and connect the Ethernet Network cable by connecting the following

1. DC 5 volt supply using USB-C connector to the Raspberry Pi
2. Ethernet Network cable
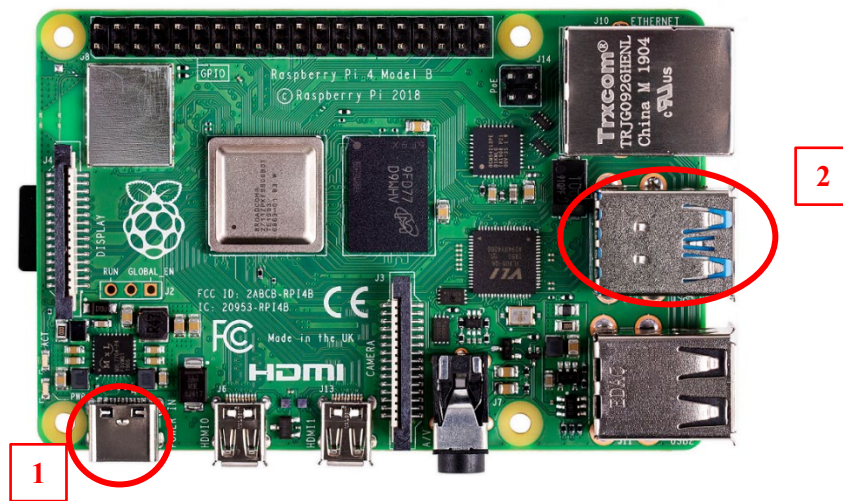3. DC 9volt supply to the Development as shown in the diagram below.
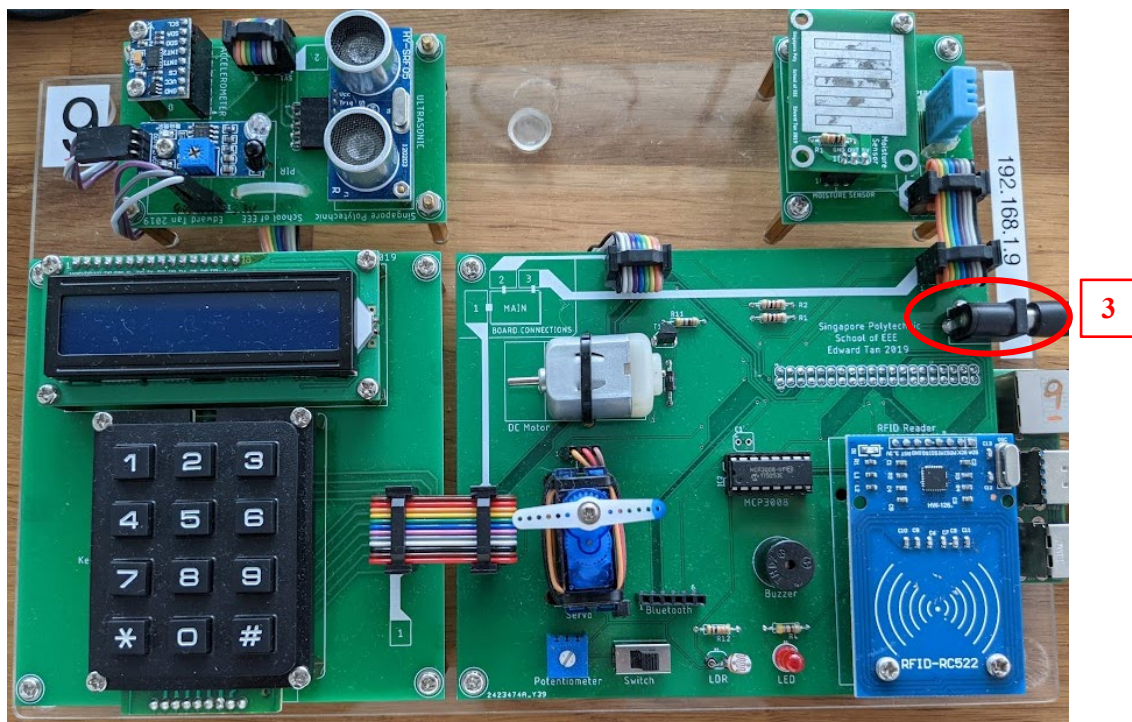


**Figure 1 – Raspberry Pi 4**



**Figure 2 – Raspberry Pi Development Board**

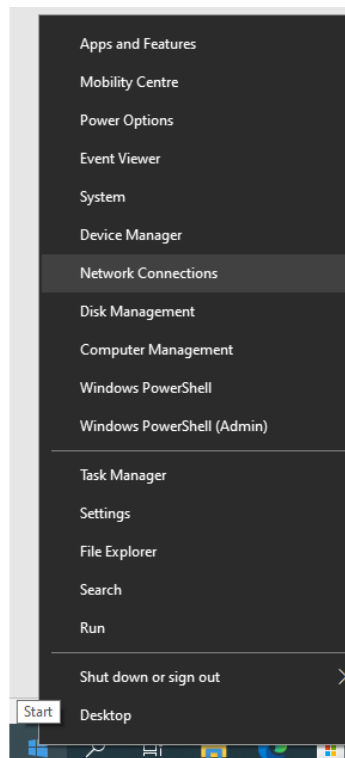1.3    Right click on the Windows Start menu below and select "Network Connections",



**Figure 3 – View the Python Package window in PyCharm.**

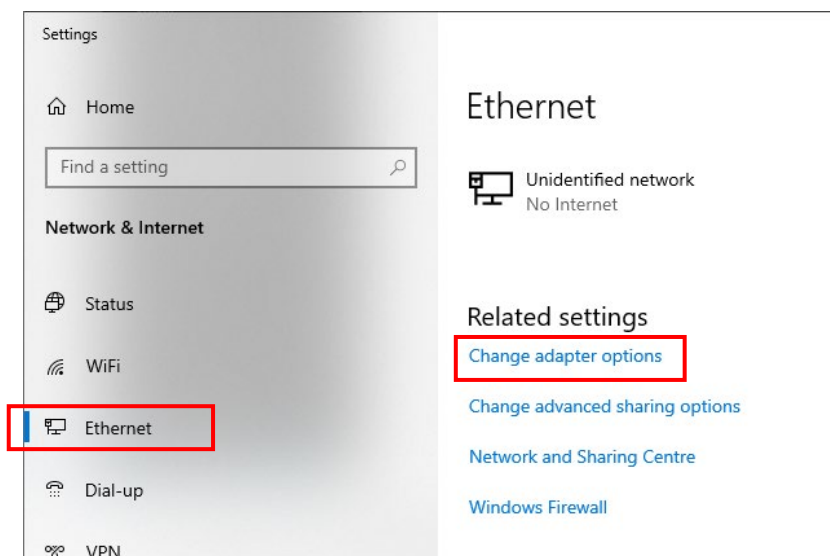1.4    Select "Ethernet" below and then select "Change adapter options"



**Figure 4 – Windows "Ethernet" settings**

1.5 Select the Ethernet adapter that is physically connected to the Ethernet port that is connected to the Raspberry Pi below then right click and select "Properties"
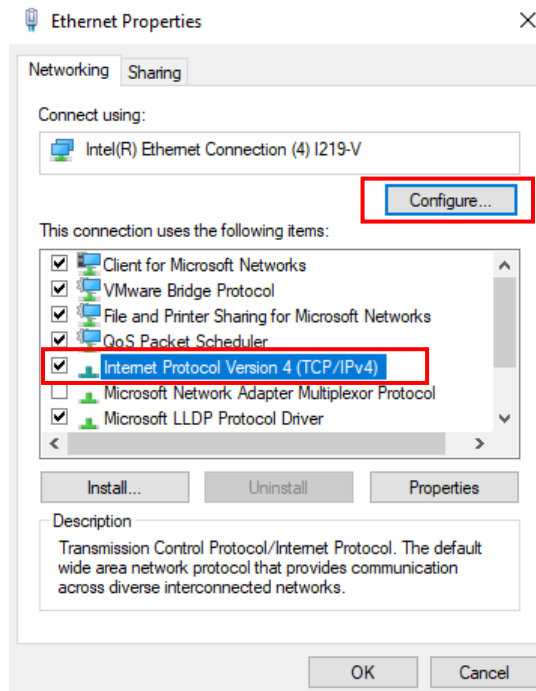


**Figure 5 – Ethernet Configuration**

1.6 Select "Internet Protocol Version 4 (TCP/IPv4) then click "Properties" and configure the following settings below,
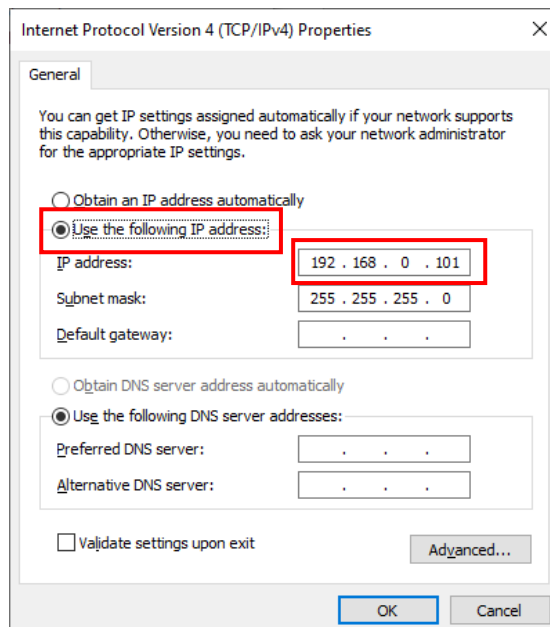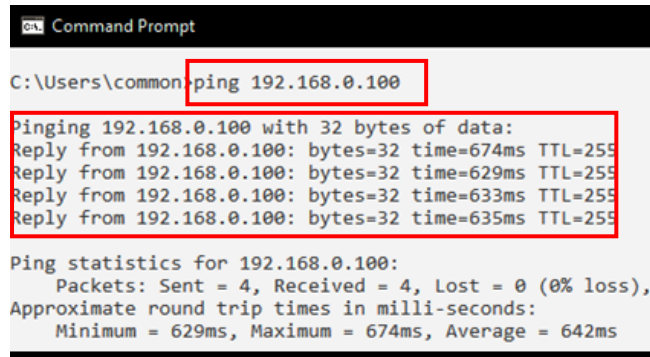


**Figure 6 – IP configuration Settings**

1.7 Check that the connection between your laptop and the Raspberry Pi is working by running a "ping" command below at the Windows Command Prompt below to ping the Raspberry Pi which has an IP Address = 192.168.0.100

If the Network connection was successfully setup on your laptop and the Raspberry Pi then you should see a similar result above when the "ping" has 0% loss packets.



**Figure 7 – Ping communication check between PC and Raspberry Pi**

## 2 Installation of Putty

We will use the PuTTY tool to allow us to connect to the Raspberry Pi and remotely run Linux command from our laptop via the SSH protocol

2.1. Download PuTTY for Windows from https://www.putty.org/ and install it in your laptop.

2.2. Connect the Ethernet cable from your laptop to the Raspberry Pi. If your laptop does not have a physical Ethernet port, please request to loan a USB to Ethernet Adapter from your lecturer.

2.3. Run Putty and Connect to the Raspberry Pi using the IP address 192.168.0.100

2.4. If this is the first time you are connecting to the Raspberry Pi via Putty, then you should see the prompt window "PuTTY Security Alert" to accept new Security keys.
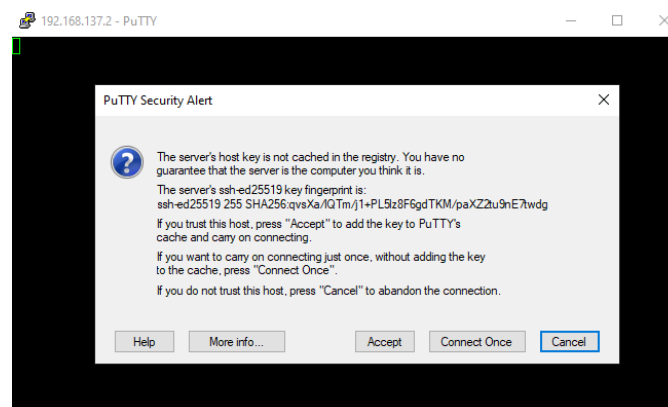


**Figure 8 – Ping communication check between PC and Raspberry Pi**

2.5. After accepting the security key, enter the username as "pi". When prompted for password, enther "password". Note that unlike most of other software, when you enter the password, no '*' will be shown.



**Figure 9 – PuTTY login prompt**

2.6. After entering the username and password, the SSH remote terminal session will start and show the console output below.

2.7. You are now ready to start running Linux commands remotely via the PuTTY SSH Terminal.



**Figure 10 – PuTTY Remote Terminal**

## 3    Installation of VNC Viewer

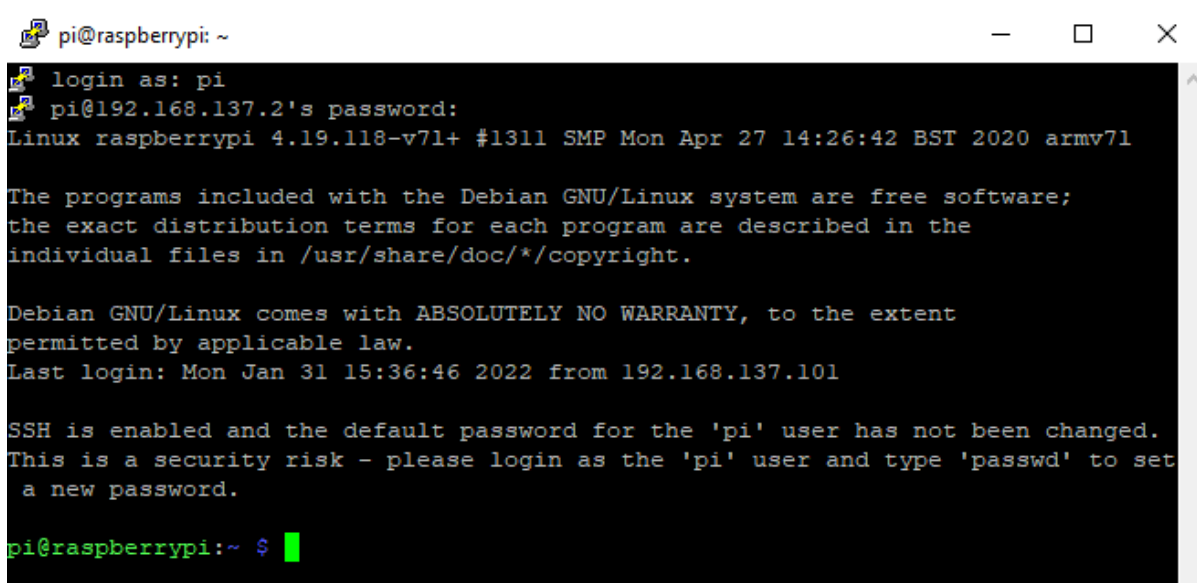We will now install VNC Viewer which is a Remote Desktop tool that allows us to connect to the Raspberry Pi remotely and view the Raspberry Pi OS Desktop interface.

3.1.    Download VNC Viewer for Windows from the link below

https://downloads.realvnc.com/download/file/viewer.files/VNC-Viewer-6.22.315-Windows.exe?_ga=2.227493657.524886960.1651897950-1801372009.1651897950

3.2.    After installing VNC Viewer, create a new connection instance to the Raspberry Pi from "File $\rightarrow$ "New Connection"

3.3.    In the "New Connection", set the field "VNC Server" to the static IP Address of the Raspberry Pi = **192.168.0.100:5900**
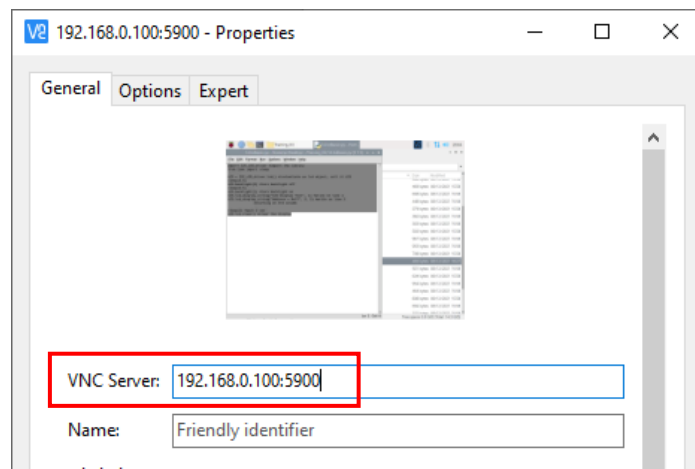


**Figure 11 – VNC Viewer to Raspberry Pi connection**

3.4.    Using the same "user name = pi" and password = "password", start a VNC Remote Desktop connection to the Raspberry Pi and you should see the Raspberry Pi desktop below.
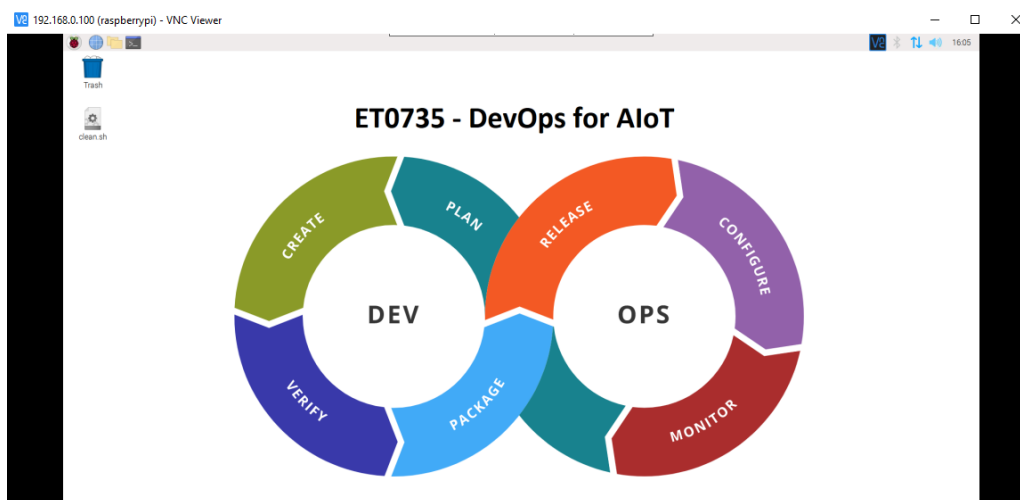


**Figure 12 – VNC Viewer Raspberry Pi Remote Desktop**

## 4    Introduction to Linux Commands

The Raspbian OS running on the Raspberry Pi is a Linux based OS, therefore you will need to be familiar with several Linux commands to develop applications on the Raspberry Pi.

Since we are developing on the Raspberry Pi in "headless" mode, ie: without the connection of a physical keyboard, mouse and display, all the Linux commands will be executed remotely via a SSH session using Putty in Windows or Terminal in Mac OS.

The following are some common Linux commands that are useful,

| Linux Command | Description | Example |
|---|---|---|
| pwd | Displays the current directory path | |
| ls | Lists all the files and folders in the current directory | |
| mkdir | Makes a new directory at the current or specified path | |
| cd | Changes the current path to the specified directory path | |
| sudo | Changes the user permissions level to the root user which may be required to run some Linux commands | "sudo shutdown now" |

**Table 1**

**Exercise 1**

In the next section of this lab we will clone and run a sample Python application on the Raspberry Pi.

To prepare for this, we first need to create a new directory in the Raspberry Pi where a Github repository will cloned in the next part of this lab,

In the next section of this lab, we will need to upload the local PyCharm project to the Raspberry Pi. Therefore, we first need to create a new folder on the Raspberry Pi.

1. Before working on the exercises on the Raspberry Pi, please double click on the bash script file "clean.sh" on the Desktop of the Raspberry Pi using the VNC Viewer shown below

2. When the pop-up box appears, select "Execute" and check that the folder at /home/pi/ET0735 is removed
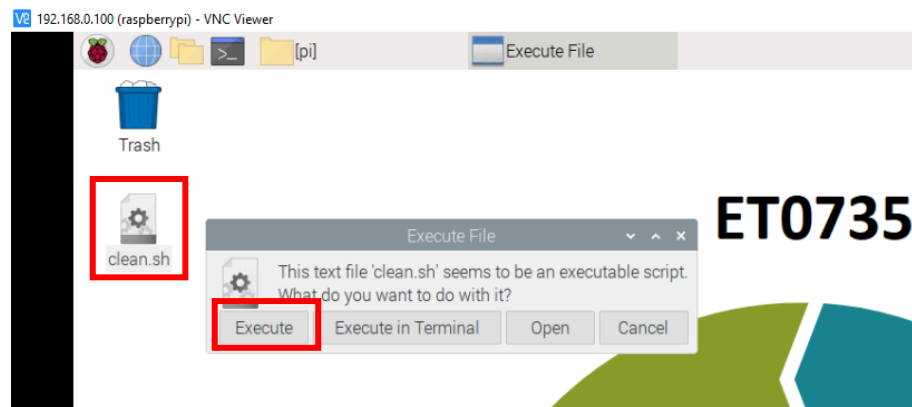


**Figure 13**

3. Using PuTTY connect to the Raspberry Pi terminal and enter a Linux command to create the directory **/home/pi/ET0735/Lab4**

Write down the Linux command/s you used below

```
cd /home/pi
mkdir ET0735
cd /home/pi/ET0735
mkdir Lab4
```

2. Change to the directory **/home/pi/ET0735/Lab4** and check that the directory change is successful.

Write down the Linux command/s you used below

```
cd Lab4          cd /home/pi/ET0735/Lab4
```

## 5    Cloning Github repository containing Git submodules

The Lab 4 Github repository contains a git submodule 'hal' which contains the Hardware Abstraction Layer (HAL) code and provides Application Programming Interfaces (APIs) that allow us to access the sensors and actuators connected to the Raspberry Pi on the development board.

5.1.    To clone all the files in a Git repository including files directories with Git submodules, we need to add the additional parameter "--recurse-submodules" to the Git "clone" command that we have been using in the previous labs.

5.2.    On your laptop, make sure that you are in 'C:\Local_Git_Repository'(use cd command to navigate). Clone the Lab4 repository for Lab 4 from the URL https://github.com/ET0735-DevOps-AIoT/Lab4 using the slightly modified Git clone command below,

```
git clone --recurse-submodules https://github.com/ET0735-
DevOps-AIoT/Lab4.git
```

5.3.    After cloning the Lab 4 repository, check all the files have been cloned into your local Git repository including the Git submodule folder at **src/hal**

## 6      Setup of PyCharm Remote Debugger

To develop and test Python code on the Raspberry Pi, we need to configure PyCharm to connect remotely to the Raspberry Pi and allow for remote debugging and testing.

In the previous labs, we configured PyCharm to use the Python interpreter installed on our local machine. However we are now developing Python applications on the Raspberry Pi, therefore we will need to use the Python interpreter on the Raspberry Pi and not the interpreter on our laptops.

We now need to configure the PyCharm project interpreter to now remotely connect to the Raspberry Pi to allow us to debug and test Python code running on the Raspberry Pi.

6.1.    Launch PyCharm software on your laptop.

6.2.    When prompted to choose project, click "Open" and navigate to c:\Local_Git_Repository\Lab4

6.3.    In the "Lab4" PyCharm project, Go to "File → Settings" then under "Project: Lab4" select "Python Interpreter" as shown below and click the gear icon.
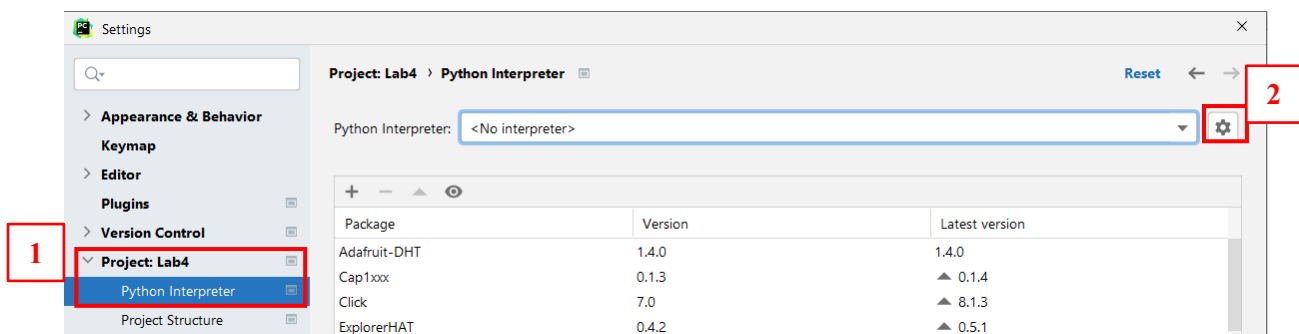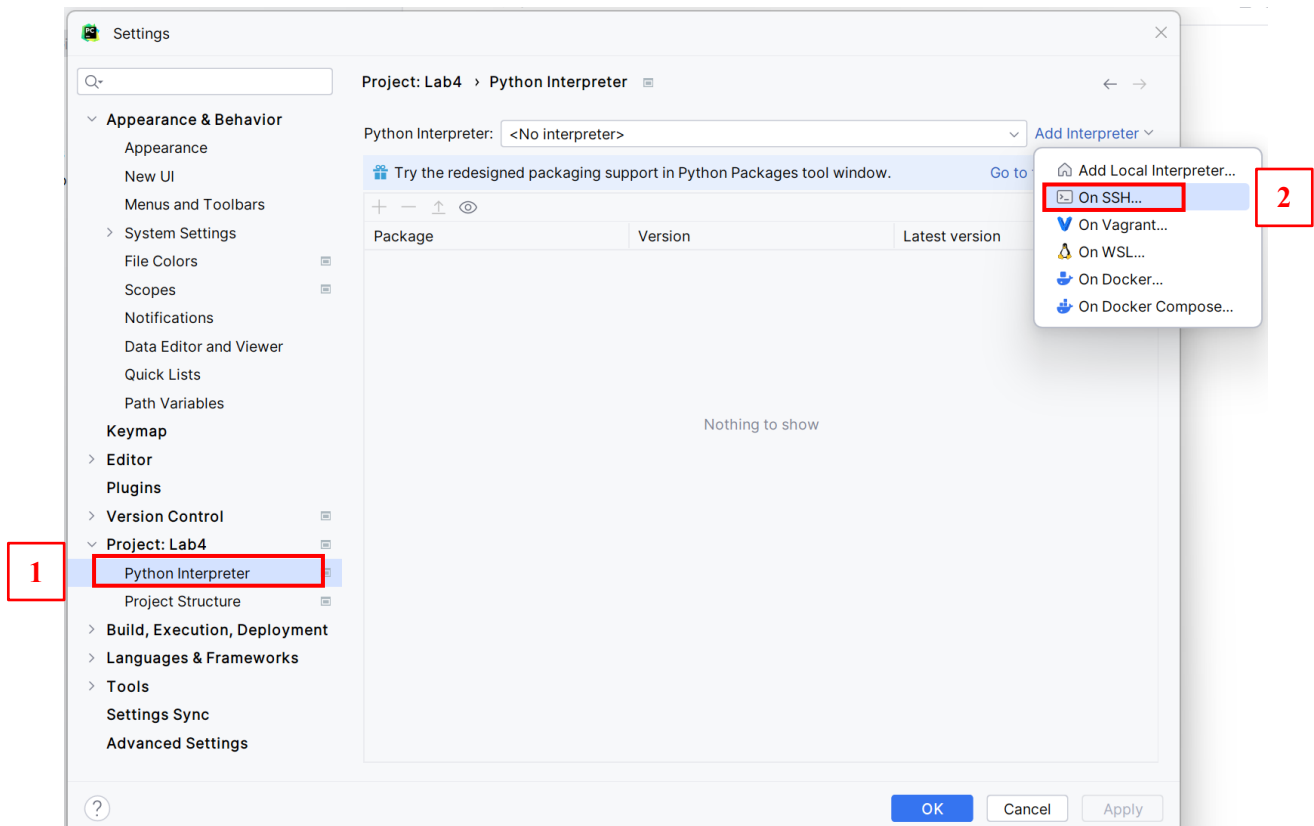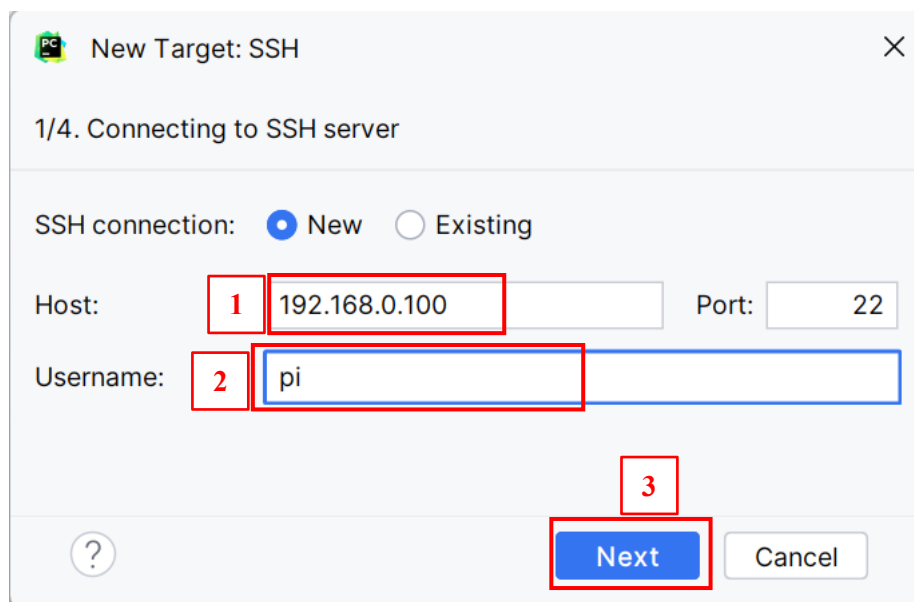


**Figure 14 – PyCharm Interpreter Configuration**

6.4.    To add a new Python Interpreter, click "Add"

**Figure 15 – Add new Python Interpreter**

6.5. Select SSH interpreter and fill in the "New Server configuration" below and set the "Host" = 192.168.0.100 and the "username" as "pi" then click "Next"



**Figure 16 – PyCharm SSH Interpreter Configuration**

6.6. Enter the password = "password" to log in to the Raspberry Pi remotely, select "Save Password" and then click "Next"
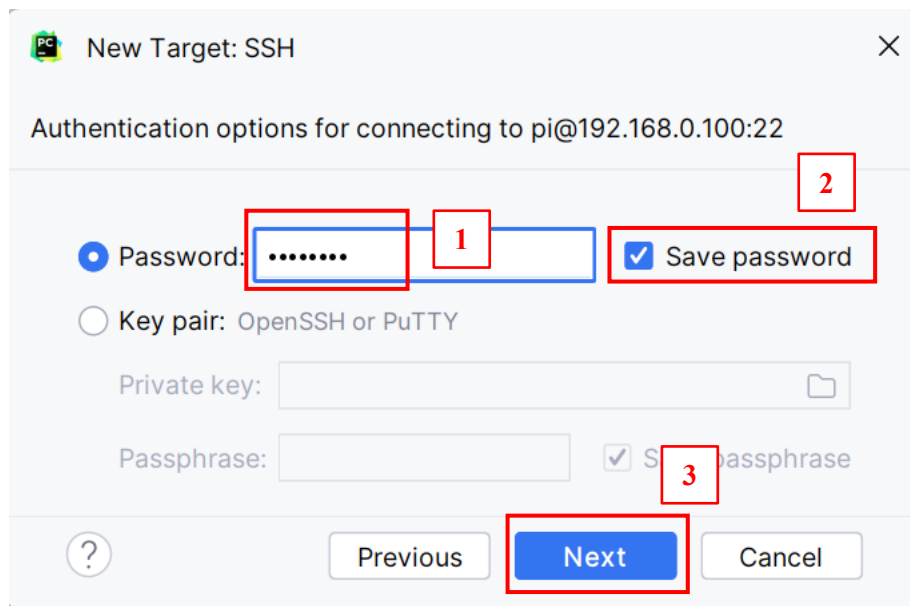


**Figure 17 – PyCharm SSH Interpreter Configuration**

6.7. When a window pops up and shows "Introspection completed", click "Next".

6.8. When a new window pops up for you to set the Project directory and Python runtime configuration, click "System Interpreter".

6.9. Complete the configuration to add the Raspberry Pi Python Interpreter

6.9.1. "Interpreter" = **/usr/bin/python3**

6.9.2. Select "Execute code using this interpreter with root privileges via sudo"

6.9.3. For Sync folders, click the "Folders" icon at the far right. When the "Edit Sync Folders" window pops up, make the window wider and under the "Remote Path" column, enter /home/pi/ET0735/Lab4

You should see the "Sync Folders" setting is changed to <Project root>-> /home/pi/ET0735/Lab4, as shown in Figure 18.

6.9.4. Select "Automatically upload project files to the server"

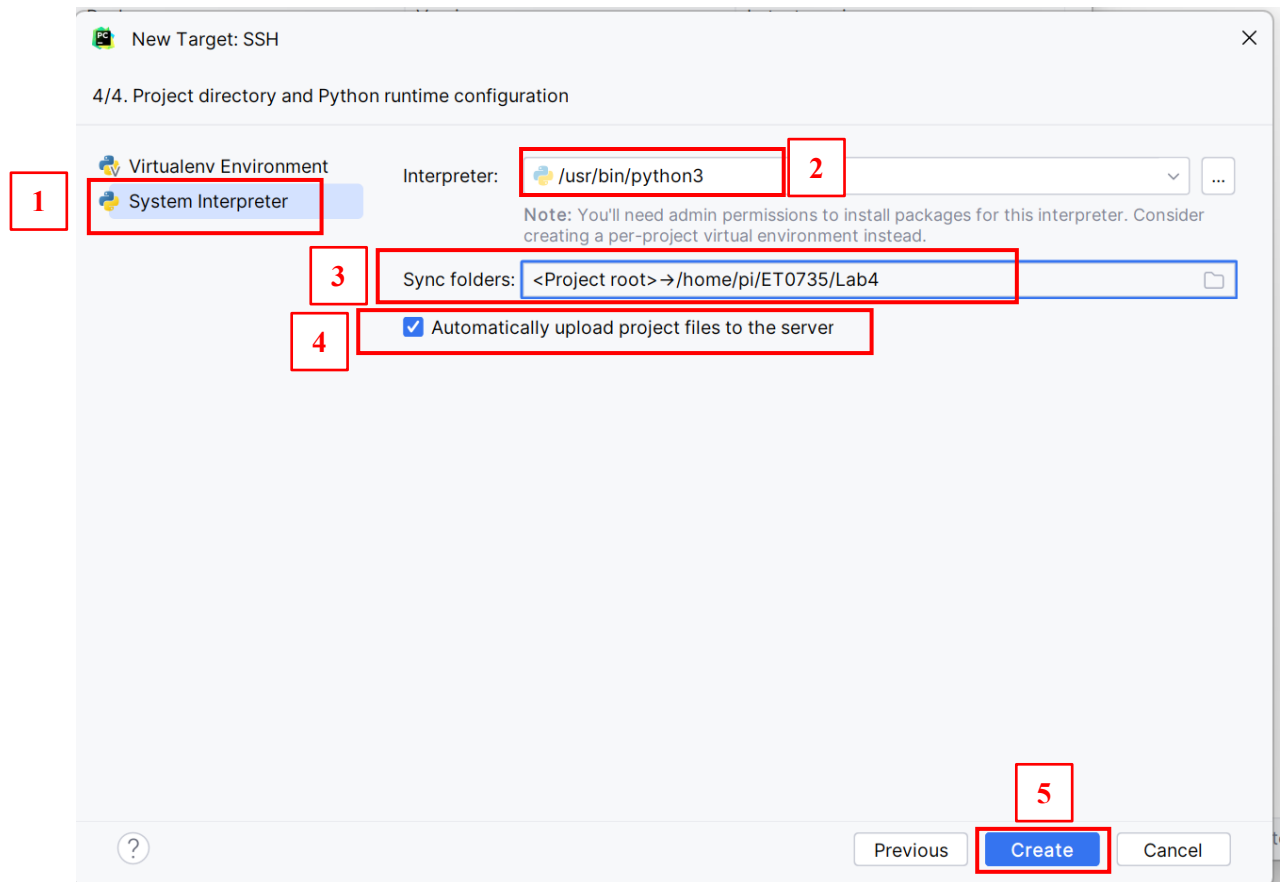6.9.5. Click "Create", "Apply", then "OK".

**Figure 18 – PyCharm SSH Interpreter Configuration**

# 7    Python Programming on Raspberry Pi

We will start to develop Python applications that run on the Raspberry Pi hardware and using PyCharm to develop and test our Python code

7.1.   We already configured PyCharm to remotely connect to the Raspberry Pi in section 4 so we need to first check the connection is working

7.2.   To test the PyCharm connection to the Raspberry Pi, under "Tools → Deployment → Configuration", click the "Test Connection" button
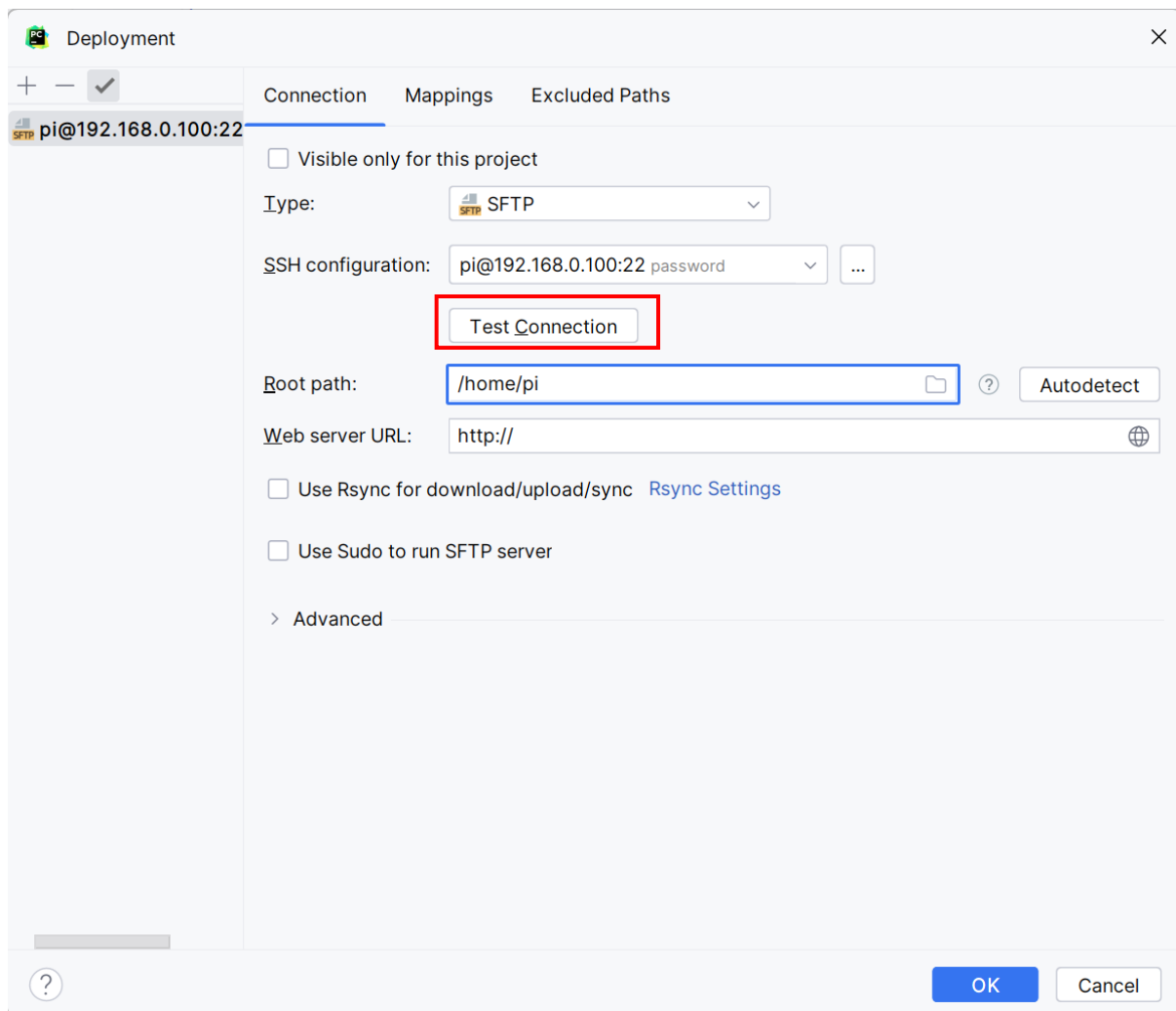
ET0735 - DevOps for AIoT



**Figure 19 – PyCharm Deployment Configuration**

7.3. If PyCharm successfully connects to the Raspberry Pi, then you should see the message box below



**Figure 20 – Successful connection from PyCharm to Raspberry Pi**

7.4. We will now run the Lab4 sample code in PyCharm, which will then Deploy and Run the Python code on the Raspberry Pi Development Board.

7.5.  Set the deployment base directory on the Raspberry Pi in the field "Root path" to the Raspberry Pi home directory at "/home/pi"

7.6.  Under "Tools → Deployment → Configuration", in mappings tab, set the field for the "Deployment path" = /ET0735/Lab4

**Note: The "Deployment path" is based on the base folder set in the field "Root path" in the "Connection" tab.**

**Usually after clicking on "Autodetect", the "Root path" field will be set to /home/pi.**

7.7.  Therefore the full Deployment path will be internally configured by PyCharm to "/home/pi/ET0735/Lab4". You should confirm this by clicking on the "Mapping" tab and check that the "Deployment Path" is set to /home/pi/ET0735/Lab4. Click "OK".
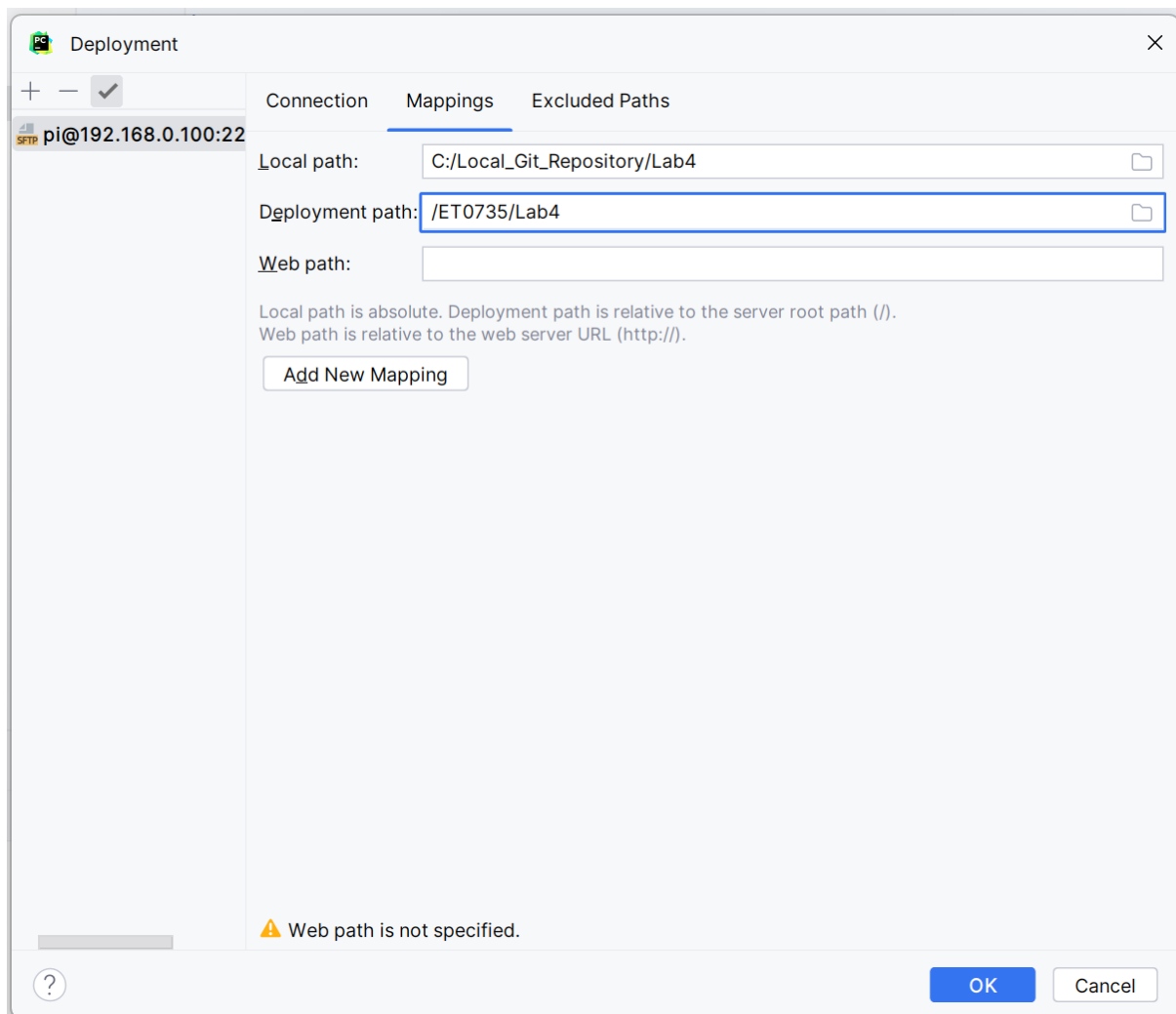


**Figure 21 – Successful connection from PyCharm to Raspberry Pi**

7.8.  The final step before we can run a Python project in PyCharm on the Raspberry Pi is to configure a "Run Configuration" in PyCharm

7.9.  In PyCharm under "Run $\rightarrow$ Run", check if there is at least 1 "Run Configuration" that has already been setup as shown below
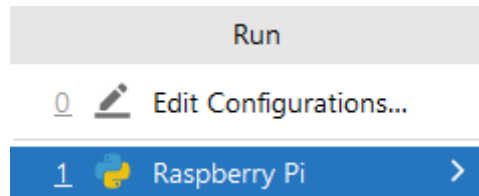


**Figure 22 – PyCharm "Run Configuration"**

7.10. If there are no valid "Run Configurations", then you need to create a new Python "Run Configuration"

7.11. To create a new "Run Configuration", select "Edit Configurations" in the "Run" pop-up shown in Figure 22

7.12. Click the "+" button to create a new :Run Configuration". Choose "Python" from the long list

7.13. For the "Name" field, enter "Raspberry Pi" or other name you prefer. Set Script path" to point to the full path of PiDemo.py" (i.e. C:\ET0735\Lab4\src\PiDemo.py). For the "Python Interpreter" field, select the Remote Python SSH interpreter that you had setup in Steps 6.4 to 6.9.
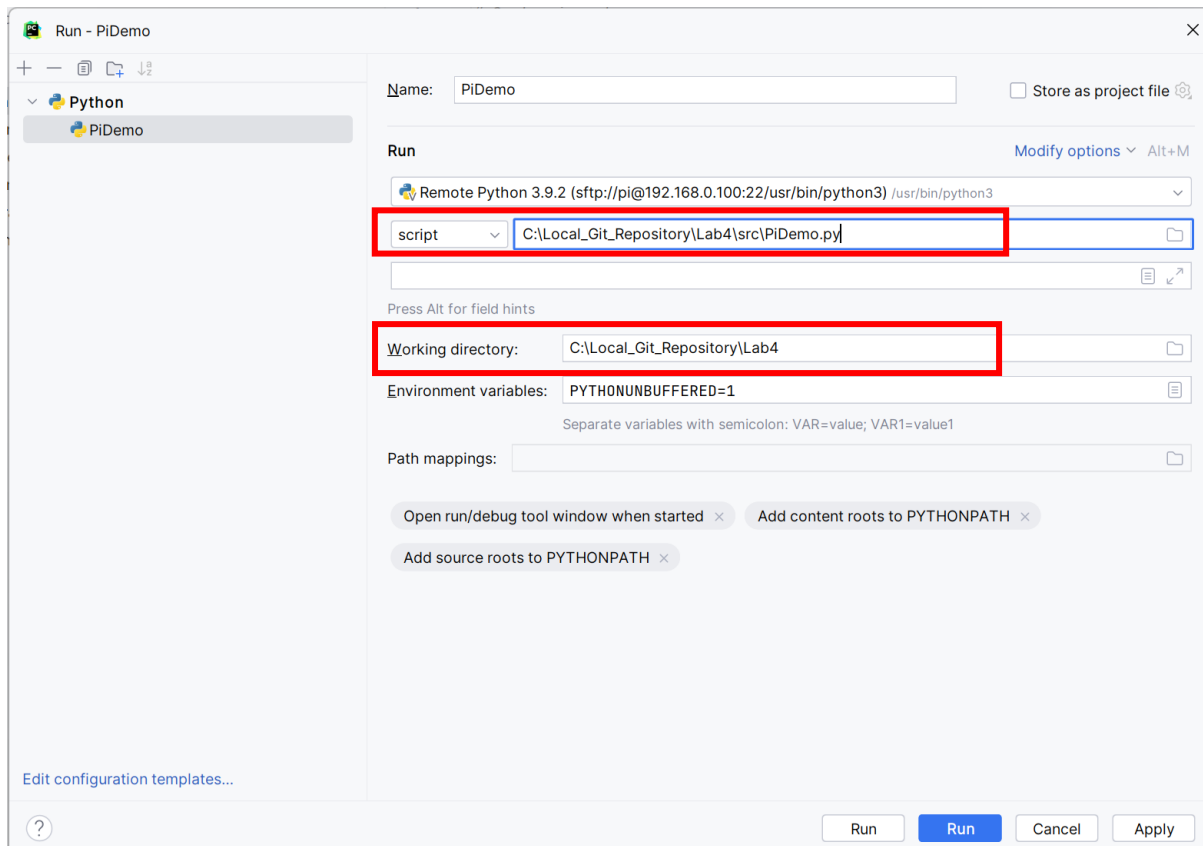
7.14. Click "Apply" then "Close".

ET0735 - DevOps for AIoT



**Figure 23 – PyCharm "Run Configuration"**

**Note: The fields "Script path" and "Working directory" are based on the location of the Python file that you are loading from your own laptop. Therefore please select the correct path on your laptop and not to directly use the same paths in Figure 23.**

## 8.   Deployment - Uploading files to Raspberry Pi via SSH in PyCharm

PyCharm conveniently supports the transfers of file from your local machine (laptop) to a remote machine (Raspberry Pi).

Before we start debugging any Python code on the Raspberry Pi, we first need to transfer ("deploy") the entire PyCharm project and all related files to the Raspberry Pi.

8.1. To copy the files to the Raspberry Pi, in PyCharm right click on the top level project folder "Lab 4" and select "Deployment → Upload to pi@192.168.0.100:22" as shown below in the Figure 24.
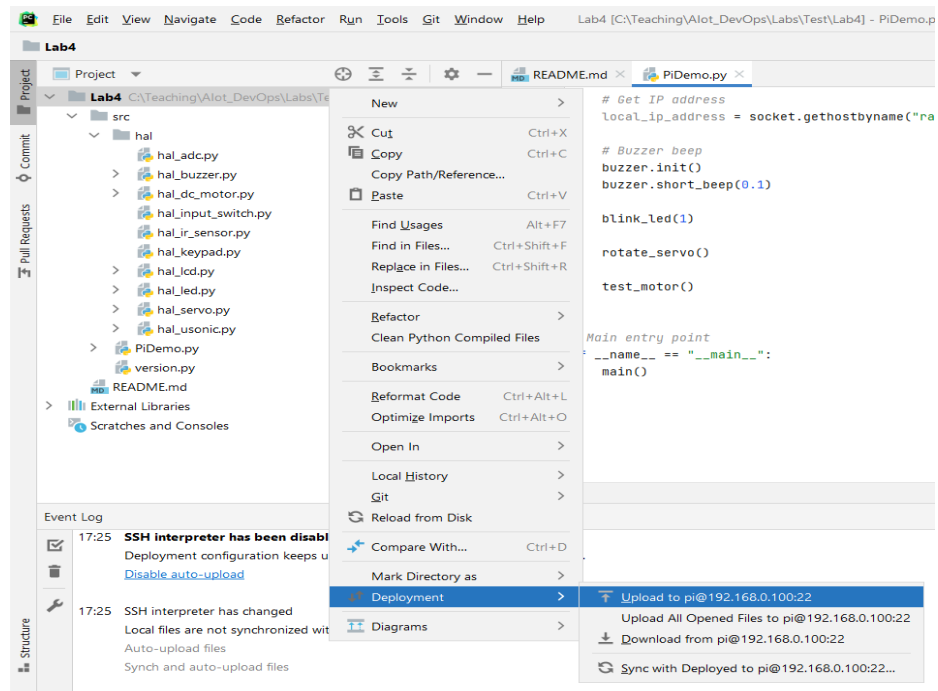
**Figure 24**

8.2. When all the Lab 4 files have been transferred to the Raspberry Pi, use VNC viewer to verify that all the files have been copied to the Raspberry Pi directory **/home/pi/ET0735/Lab4**, including the files under /home/pi/ET0735/Lab4/src directory (see figure 25).
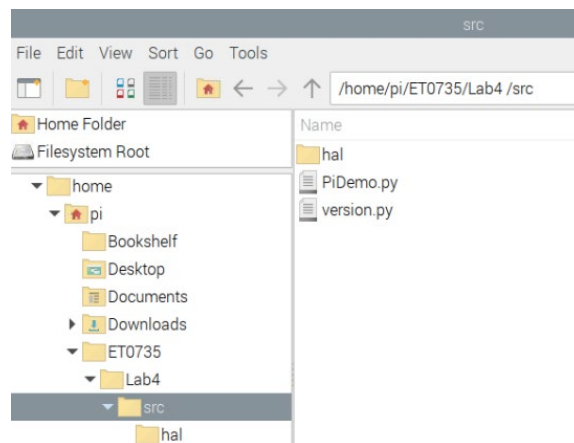


**Figure 25**

8.3. Finally, after setting up a "Run Configuration", run the main Python file "PiDemo.py by clicking "Run" and select "Run Raspberry pi". Check that the following Raspberry Pi peripherals are activated:

- LCD shows the message "DevOps for AIoT" and the Software version number "v1.00"
- Buzzer beeps
- LED blinks
- Servo Motor rotates
- DC motor is switched on for 1 second

**Exercise 2**

Now that we have setup PyCharm to deploy and execute Python application remotely on the Raspberry Pi, we can now start to implement some basic Python applications on the Raspberry Pi.

In this exercise, we will implement a simple Python application to control the blinking rate of the LED on the Raspberry Pi development board.

For additional documentation on the Raspberry Pi Development board hardware pin assignments, HAL sample code, etc, please refer to the documentation at the Github page below,
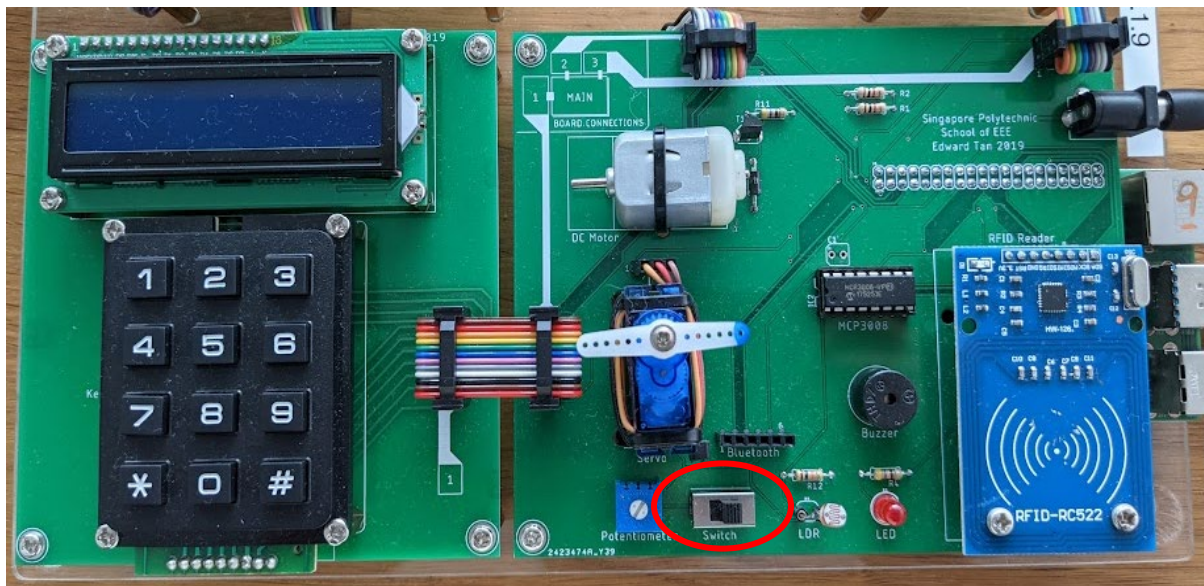
https://github.com/ET0735-DevOps-AIoT/raspberry-pi-dev-board/wiki



**Figure 26**

(a) In the existing PyCharm project for Lab 4, create a new Python file "PiLedTest.py". Under the src directory by right-click the src folder in the PyCharm project directory, and select "python file", enter the file name "PiLedTest.py" and press <ENTER>

(b) Using the HAL Python file "hal_input_switch.py", implement Python code that will blink the LED at a frequency of 5 Hz if the switch highlighted in Figure 26 above is moved to the **left position**.

If the switch is changed to the **right position**, then the LED should be turned OFF.

(c) Update your implementation to increase the blinking rate of the LED to 10Hz if the switch is changed to the **right position**. The LED should continue to blink at 5 Hz if the switch is in the **left position**.

(d) Extend your implementation such that when the LED is changed to the **right position** the LED for will blink at 10 Hz for 5 seconds, after which the LED should be turned OFF.

**Exercise 3**

To complete the lab exercises, create a new Github repository for Lab 4.

(a) Add commit and push all changes to Github

(b) Create a new Github release "v1.1" in your Lab 4 Github repository with the changes pushed to Github for Exercise 2