|  |  |
| --- | --- |
|  | [Warszawa, dnia 31 sierpnia 2020 r.] |
| [AB/XYZ/000/2020] |  |

Pan

Sebastian Buczek

Praktykant

[Wpisz tekst…]

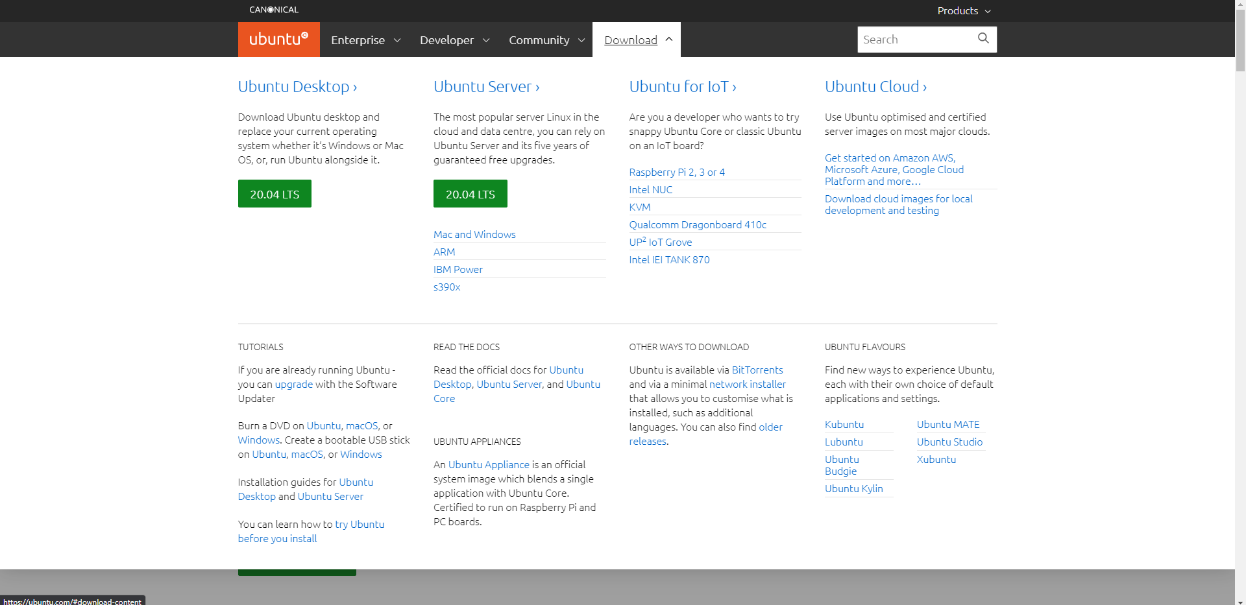
[Wpisz tekst...]

[Z wyrazami szacunku]

[dr Imię Nazwisko]

[Kierownik]

**Instalacja systemu UNIX na komputerze instytutu**

1. Pobieram plik iso Ubuntu Desktop w najnowszą wersję: 
2. A także PowerISOx32 link: https://www.poweriso.com/download.php
3. Instaluję ubuntu z zaznaczeniem funkcji „Erase disk and install Ubuntu”. Zainstalowana wersja to 20.04.

Nazwa komputera oraz użytkownika: iel-zne-serwer

Hasło: ielnauka2020

1. Po zakończeniu instalacji oraz zresetowaniu komputera przechodzę do instalacji PostgreSQL oraz pgAdmin. Wykorzystuję do tego następujące komendy:
2. wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
3. sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb\_release -sc)-pgdg main" > /etc/apt/sources.list.d/PostgreSQL.list'
4. sudo apt update
5. sudo apt-get install postgresql-11
6. sudo passwd postgres
7. sudo su -l postgres
8. su - postgres
9. psql
10. postgres=# \password postgres
11. \q
12. Exit

(wszystkie hasła to: ielnauka2020)

1. sudo apt-get install pgadmin4 pgadmin4-apache2

(w wyskakującym oknie ustawiam hasło ielnauka2020)

1. Uruchamiam pgAdmin i tworzę strukturę bazy tak jak w przypadku systemu windows.
2. Zgrywam dane starej bazy z komputera który dotychczas pełnił rolę serwera na zewnętrzny nośnik.
3. Aby dodać adresy mogące łączyć się z bazą wyszukuję pliku pg\_hba.conf – początkowo znaleznienie go było problematyczne. Oto rozwiązanie:
4. cd /etc/postgresql/11/main

(zamiast 11 może pojawic sie inna wersja)

1. sudo gedit pg\_hba.conf

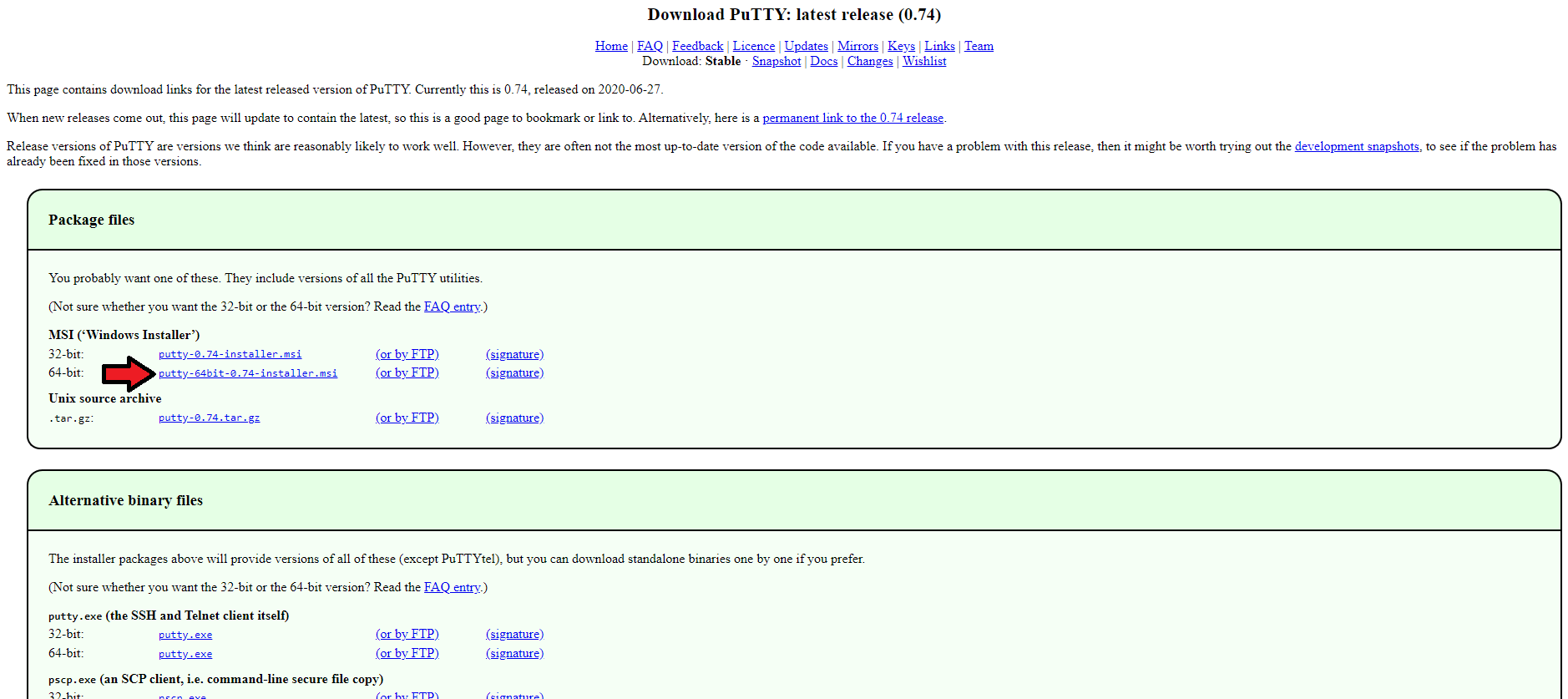
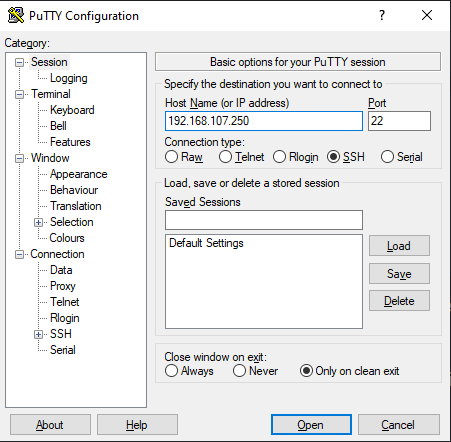
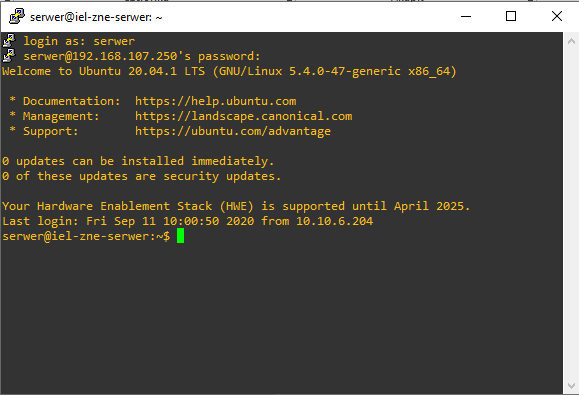
Może zdarzyć się tak, że nie będziemy mieli uprawnień do wykonania wyzej wykonanych poleceń. Należy wtedy posłużyć się kontem postgres jako użytkownik bazy (w terminalu wpisać - su – postgres) i następnie spróbować przejść do folderu main.

1. **W pliku postgresql.conf zmieniam atrybut listen address na ‘\*’ co umożliwia łączenie się do bazy po sieci lan. Należy zwrócić uwagę, czy linia nie jest zakomentowana ( znak # na początku linii)**

---------------------------------------------------------------------------------------------------------------------------------

**Połączenie SSH z komputerem serwerowym**

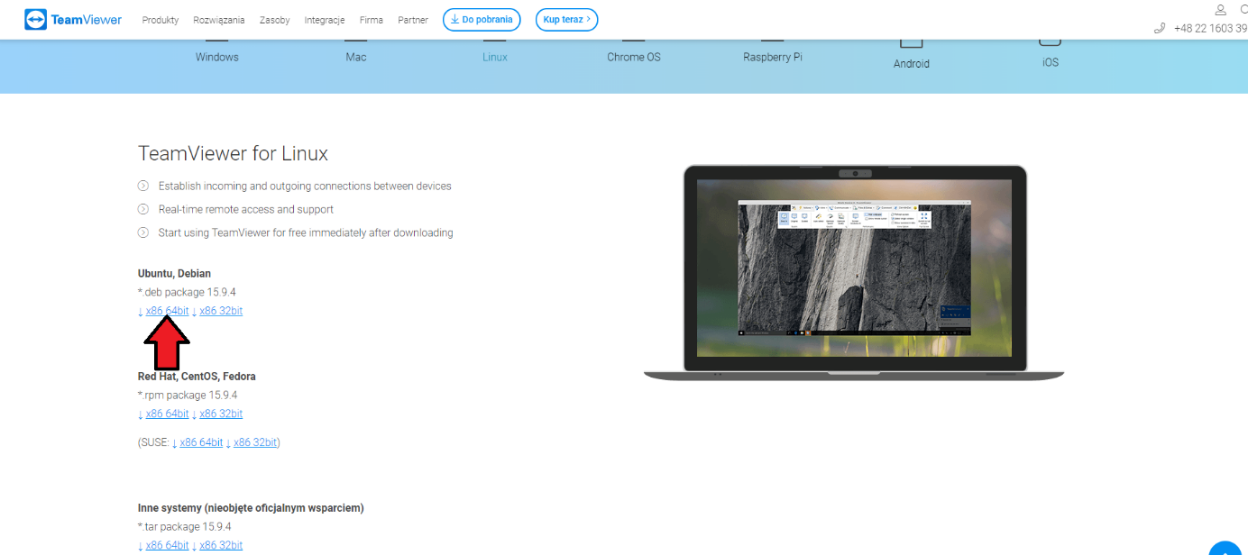
Chcę umożliwić zarządzanie komputerem serwerowym za pomocą innego komputera. Chcę do tego wykorzystać połączenie SSH. W przyszłości możliwe że wykorzystam również program TeamViewer do zdalnej kontroli komputera.

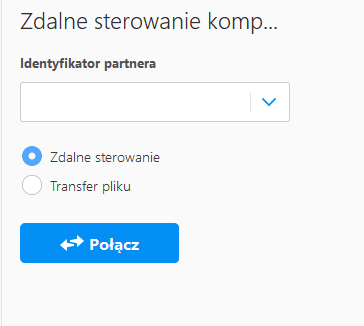
1. Instalacja SSH
2. Sudo apt install ssh
3. Sudo systemctl enable --now ssh
4. Sudo systemctl status ssh
5. Pobieram program Putty
6. <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
7. 
8. Instaluję z domyślnymi ustawieniami
9. Wpisuję adres serwera i ustawiam parametry jak poniżej:
10. Jako login wpisuję**:** **iel-zne-serwer**
11. Jako hasło: ielnauka2020
12. Jest to terminal taki jak ten którego używamy podczas pracy na systemie linux, obsługuje wszystkie jego komendy.

---------------------------------------------------------------------------------------------------------------------------------

**Zdalne sterowanie serwerem**

1. Instaluję program TeamViewer na ubuntu poprzez pobranie pliku .deb z oficjalnej strony

link: [https://www.teamviewer.com/pl/do-pobrania/linux/](https://www.teamviewer.com/pl/do-pobrania/linux/)

1. Instaluję poprzez dwuklik na pobrany plik i dalej domyślnie zgodnie sugestiami instalatora.
2. Po uruchomieniu programu (na Ubuntu) widzimy okno pokazujące ID logowania wraz z hasłem.
3. Wpisujemy powyższe dane na komputerze klienta 
4. Po wyłączeniu komputera z Ubuntu kod ulegnie zmianie. Można temu zapobiec poprzez utworzenie konta w TeamViewer oraz kupno wersji premium. W dalszej części raportu zaprezentowałem alternatywny sposób uzyskania ID do połączenia z komputerem.

-------------------------------------------------------------------------------

**Instalacja Pythona**

1. Aktualizacja pythona do najnowszej wersji:

-sudo add-apt-repository ppa:deadsnakes/ppa

-sudo apt-get update

-sudo apt-get install python3.7

-sudo apt install python3-pip

-sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.7 2

2. Instalacja modułów

3. Po aktualizacji wersji Pythona wystąpił błąd z terminalem. Wymagane było przeinstalowanie wszystkich plików pythona

4. Poprawka: python jest domyślnie już zainstalowany w systemie ubuntu w wersji 2 oraz 3. Aby użyć wersji 3 wystarczy przy uruchamianiu skryptu użyć: ***python3 [nazwa\_skryptu]***

5. Aby uruchomić skrypt py\_arduino\_udp należy otworzyć konsolę w folderze z projektem, a następnie wpisać: **python3 main.py.**

**---------------------------------------------------------------------------------**

**Auto-restart po zaniku zasilania**

1. Wchodzę w ustawienia BIOSu poprzez przycisk F2 w trakcie uruchamiania systemu.

2. W ustawieniach BIOSu przechodzę do: **ADVANCED -> ACPI Settings -> Restore on AC/Power Loss** zmieniam parametr na **Power On**.

3. Wychodzę poprzez **Exit -> Save Changes and Exit**

4. Po sprawdzeniu wszystko działa zgodnie z oczekiwaniami.

**Auto-start skryptu py\_arduino\_udp**

1. W folderze /etc/systemd/system/ tworzę plik service:

cd /etc/systemd/system/

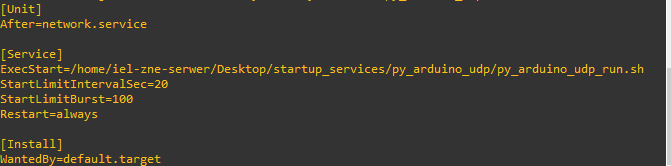
sudo touch py\_arduino\_udp.service

2. Nadaję uprawnienia do pliku :

sudo chmod 777 py\_arduino\_udp.service

3. Otwieram plik w edytorze tekstowym : sudo gedit py\_arduino\_udp.service\

4. Zapisuję w tym pliku:



5. Powyższa ścieżka zapisana w ExecStart jeszcze nie istnieje zatem tworzę ją. Do katalogu na pulpicie kopiuję plik main.py z programu oraz tworzę skrypt uruchomieniowy poprzez **touch py\_arduino\_udp\_run.sh**, nadaję uprawnienia przez **sudo chmod 777 py\_arduino\_udp.sh**.

6. Otwieram powyższy skrypt poprzez **gedit** oraz zapisuje w nim:

Polega to na tym, że w pliku .service uruchamiamy ten skrypt, który to uruchamia nasz program do odczytu i zapisu danych do bazy.

7. Aktualizuję konfigurację serwisów i ustawiam mój serwis na autostart:

**sudo systemctl daemon-reload**

**sudo systemctl enable py\_arduino\_udp.service**

**Uwagi:** Wszystko działa zgodnie z założeniami jednakże nie można śledzić działania programu w terminalu. Dane są poprawnie wpisywane do bazy. W przyszłości będę próbował to zmienić.

Aby sprawdzić czy serwis aktualnie działa używamy komendy:

**sudo systemctl status py\_arduino\_udp.service**

Aby zatrzymać działanie serwisu:

**sudo systemctl stop py\_arduino\_udp.service**

Aby manualnie uruchomić serwis:

**sudo systemctl start py\_arduino\_udp.service**

**---------------------------------------------------------------------------------**

**Po podłączeniu komputera-serwera do zasilacza UPS.**

Komputer działa po podłączeniu do zasilania. Do działania serwerowego wymaga jedynie połączenia z internetem.

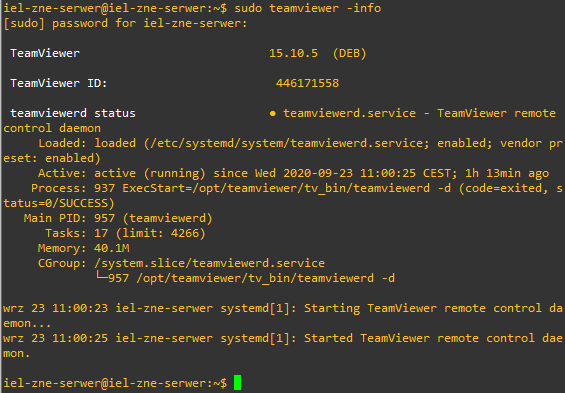
1. Obsługa SSH: Za pomocą programu Putty łączymy się za pomocą protokołu SSH do serwera.

Adres IP serwera: **192.168.107.250**

Nazwa użytkownika: **iel-zne-serwer**

Hasło: **ielnauka2020**

2. Obsługa **TeamViewer:**

Aby uzyskać od serwera poprzez TeamViewer należy najpierw przez konsolę ssh odpytać urządzenie o ID połączenia. Realizujemy to poprzez wpisanie komendy **sudo teamviewer -info**. Otrzymujemy następującą informację zwrotną

W tym przypadku ID to 446171558 (może ulec zmianie, należy sprawdzać przed połączeniem).

Hasło TeamViewer ustawiłem jako **ielnauka2020** używając polecenia **sudo teamviewer passwd ielnauka2020**

**---------**

**UWAGA:**

Może zaistnieć sytuacja, w której po ręcznym wylogowaniu użytkownika nie będzie można użyć programów zdalnej kontroli. W tej sytuacji należy połączyć się przez SSH i zrestartować system komendą:

**sudo reboot**

**---------------------------------------------------------------------------------**

**Tworzenie repozytorium git na serwerze**

Zaczynam od pobrania Gita: **sudo apt install git**

1. W katalogu użytkownika tworzę folder gitrepo.git z użyciem komendy: **mkdir gitrepo.git** można dodać **chmod 777 gitrepo.git** (nadające uprawnienia wszystkim grupom użytkowników).

2. Przechodzę do folderu poprzez **cd gitrepo.git**

3. Inicjalizuję repozytorium: **git --bare init**

4. Tworzę na komputerze klienta folder w którym będę lokalnie pracował nad repozytorium, następnie w tej lokacji otwieram konsolę Git i wpisuję:

$**git init**

Tworzę plik tekstowy test.txt i wpisuję w nim losowe znaki, a następnie:

$**git status** (sprawdzam czy plik jest widoczny)

$**git add .**

$**git commit -m ”.”**

$**git remote add origin iel-zne-serwer@192.168.107.250:/home/iel-zne-serwer/gitrepo.git**

$**git push origin master**

Następnie tworzę folder testowy na komputerze klienta i sprawdzam, czy mogę pobrać pliki z repozytorium.

$**git init**

$ **git remote add origin iel-zne-serwer@192.168.107.250:/home/iel-zne-serwer/gitrepo.git**

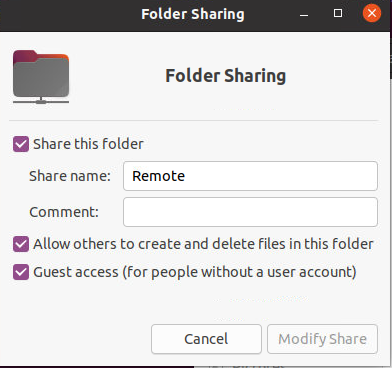
$**git pull origin master**

Wszystko działa poprawnie.

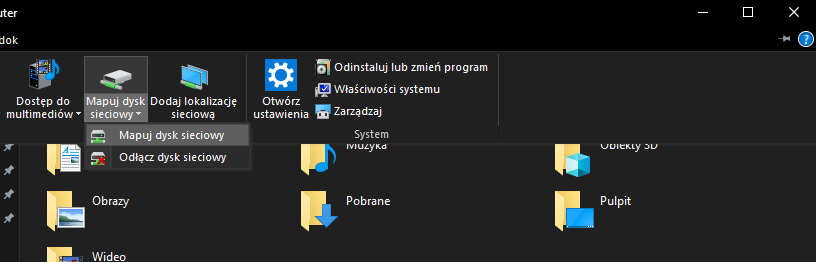
--------------------------------------------------------------------------------------

**Udostępniony katalog**

1. Tworzę katalog o nazwie Remote na pulpicie serwera.

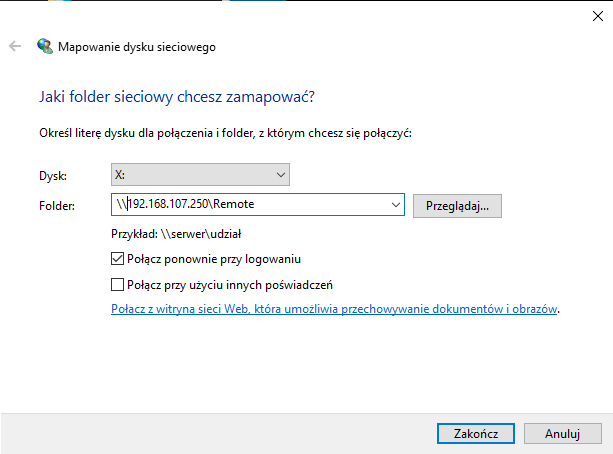
2. W menu podręcznym folderu wybieram **Local Network Share**, a następnie zaznaczam jak poniżej:

3. Aby połączyć się z folderem należy wpisać adres maszyny w eksplorator plików (Windows) poprzedzony „\\” czyli [**\\192.168.107.250**](file:///\\192.168.107.250).

****4. Aby mapować dysk sieciowy z systemu windows należy w Eksploratorze plików w zakładce **Ten komputer** wybrać opcję „**Mapuj dysk sieciowy**”.

5. Następnie w oknie wybrać nazwę dysku oraz wpisać następujący adres:

[**\\192.168.107.250**](file:///\\192.168.107.250)**\Remote**

 I opcjonalnie zaznaczyc opcje poniżej adresu.

6. Zatwierdzić przyciskiem **Zakończ**

Niestety napotkałem problemy z odczytem plików. Wytłumaczenie wymaga zagłębienia się w temat więc znajdzie się w dalszej części raportu.

--------------------------------------------------------------------------------------

**Skrypt python-CAN**

1. Instaluję bibliotekę CAN:(cmd) **python -m pip install python-can**

Napotkałem spore problemy związane z błędem, przez który kompilator źle czytał bibliotekę can. Próbowałem wielu sposobów więc ostatecznie ciężko mi określić który doprowadził do oczekiwanego wyniku. Być może to połączenie kilku. Wypisuję poniżej te kroki:

1.1 Pobieram sterowniki PCAN:

<https://www.peak-system.com/Support.55.0.html?&L=1>

(instalacja domyślna)

1.2 Pobieram sterowniki Kvaser:

<https://www.kvaser.com/download/>

(instalacja domyślna)

1.3 Zmieniam wersję biblioteki poprzez cmd:

**python -m pip install python-can==4.0.0.dev0**

1.4 W środowisku python w przypadku błędu z importem biblioteki, w wyskakującym oknie pomocniczym wybieram **install python-can package**

2.