Kilobot GUI Setup for Linux

Authors: Jack Mirenzi & Joseph Katakam

Disclaimer:

It is important to note that, based on the current knowledge at the time of writing this documentation, the Kilobot GUI is only guaranteed to work without any issues on Ubuntu 18.04. If you are using a different version of Linux, there may be errors or missing dependencies that arise during the setup process.

Despite this, it is possible that the Kilobot GUI may still work on other Linux versions, but it is not guaranteed and the setup process may not be as smooth as it is on Ubuntu 18.04. If you encounter any difficulties while setting up the Kilobot GUI on a different Linux version, it is recommended to refer to the Kilobot GUI documentation or seek support from the Kilobot community.

Using Linux (for Windows users)

There are three options for running Linux on your computer:

1. Using WSL (Windows Subsystem for Linux) on Windows OS.
    - Uses compatibility layer on windows to run native Linux command-line tools and applications directly on windows.
2. Using Virtual Machine software to run Linux.
    - Provides an isolated environment in Windows that runs a computer system simulation on top of the existing operating system using more system resources.
3. Using a dual-booted system for Linux.
    - Underlying bootup system has to be changed by changing the BIOS configuration and consumes a significant amount of memory compared to the above steps.

Note: Option 1, using WSL on Windows, is the recommended option as it is the simplest procedure and does not involve installing another operating system or making any changes to the BIOS setup. This article will guide you through the setup process using WSL on Windows.

KiloGUI

- Kilobot GUI, also known as KiloGUI, is a graphical interface used to program and control Kilobots through the overhead controller. More info *here*.

KiloGUI SOP

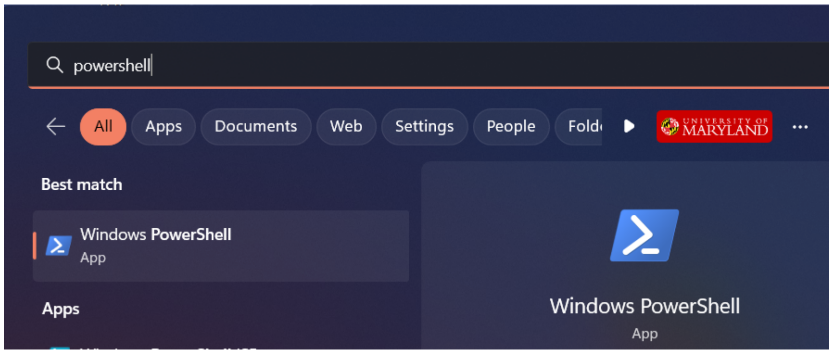
The Standard Operating Procedure (SOP) for setting up and using KiloGUI on a Windows computer running WSL with Ubuntu 18.04 is as follows:

I. Linux Setup with WSL2

This article provides a clear guide for setting up Windows Subsystem for Linux (WSL) on a Windows computer to run Linux. This process is based on the guidelines provided by Microsoft, which can be referred to *here*.

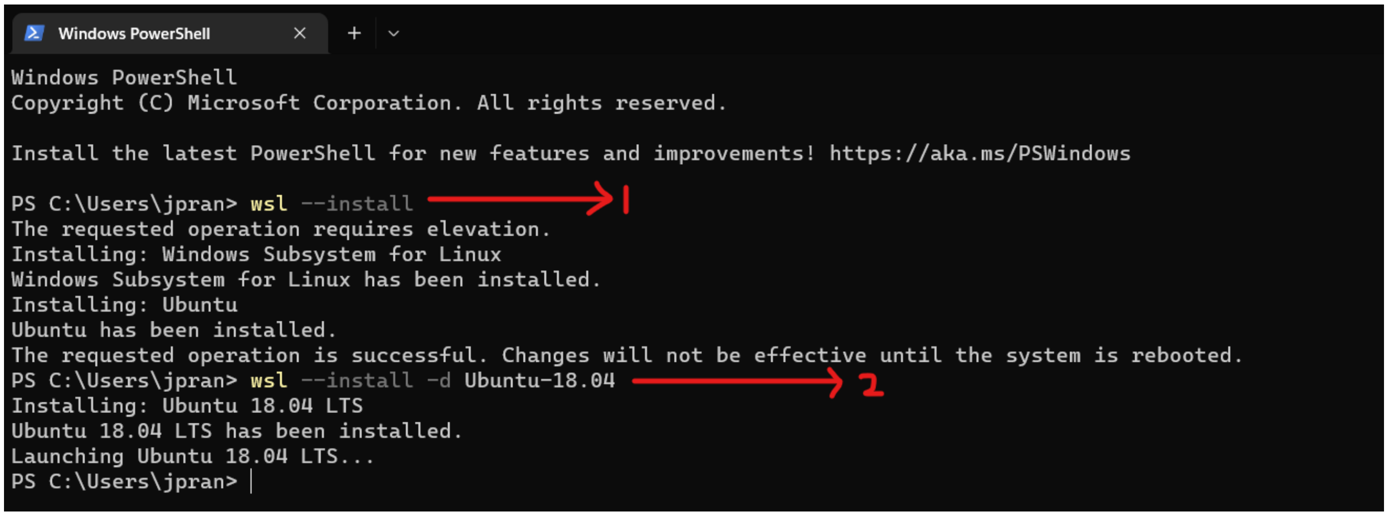
Step 1: Installing WSL

- Open PowerShell and run the command
  - `wsl --install`



Step 2: Installing the Correct Version of Ubuntu

- Run the following command in PowerShell
  - `wsl --install -d Ubuntu-18.04`



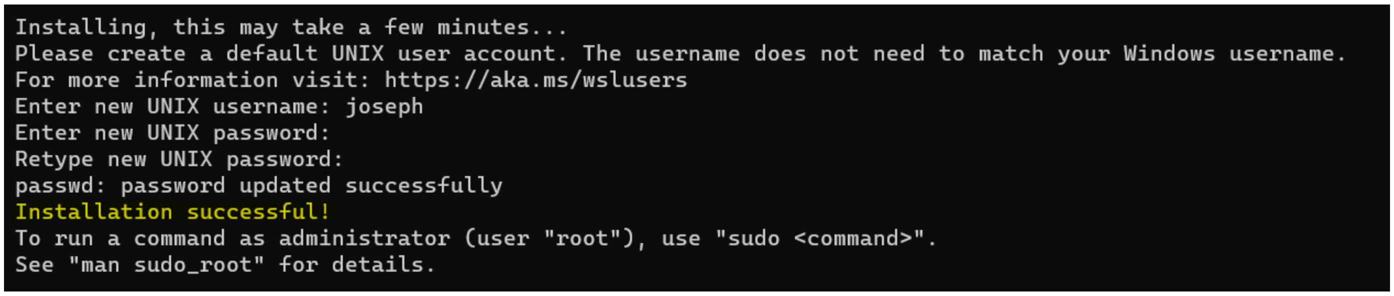
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\jpran> wsl --install
The requested operation requires elevation.
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Installing: Ubuntu
Ubuntu has been installed.
The requested operation is successful. Changes will not be effective until the system is rebooted.
PS C:\Users\jpran> wsl --install -d Ubuntu-18.04
Installing: Ubuntu 18.04 LTS
Ubuntu 18.04 LTS has been installed.
Launching Ubuntu 18.04 LTS...
PS C:\Users\jpran>
```

Step 3: Setting Up the Ubuntu User

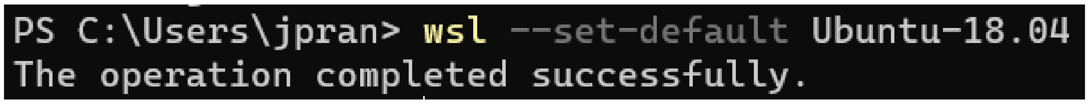
- The newly installed Ubuntu will pop up, and you will need to set up your username and password. This will be your WSL terminal/window.



```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: joseph
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Step 4: Setting the Default Distro

- In the WSL terminal, run the following command to set the default distro
  - `wsl --set-default Ubuntu-18.04`



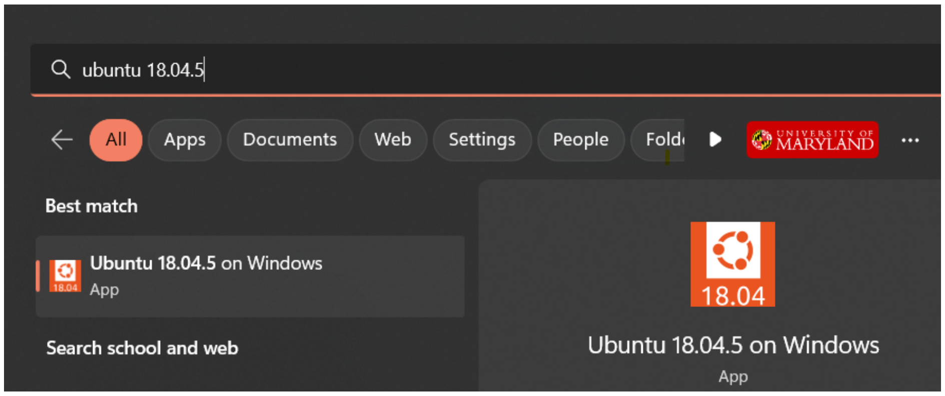
```
PS C:\Users\jpran> wsl --set-default Ubuntu-18.04
The operation completed successfully.
```

Step 5: Ending the installation process

- Close all the windows.
- After following these steps, you should now have WSL set up on your Windows computer and be able to run Linux. It is important to follow each step carefully to avoid any errors.

Step 6: Launching Ubuntu 18.04

- Launch Ubuntu 18.04 from your Windows start menu.



Step 7: Updating Ubuntu

- In the WSL terminal, run the following commands to update Ubuntu:

```
sudo apt update
sudo apt full-upgrade
```

II. Setting up Kilogui on WSL

In this article, we will guide you through the process of setting up Kilogui on WSL (Windows Subsystem for Linux).

Step 1: Install Required Packages

- The first step is to install all the necessary packages. Open the terminal and enter the following commands:

```
sudo apt-get install avr-libc gcc-avr
sudo apt-get install avrdude
sudo apt-get install libftdi-dev
sudo apt-get install build-essential cmake qtbase5-dev
sudo apt-get install qt5-default
sudo apt-get install qt4-default
```

Step 2: Download the Linux Source Code

- Use the following command in the terminal to download the zip file from GitHub and extract it:

```
curl -L
https://github.com/acornejo/kilogui/archive/refs/tags/v0.1.tar.gz |
tar -xz
```

Step 3: Build Executable

- Once the source code is downloaded, navigate to the ‘kilogui-0.1’ directory and run the following command to build the executable:

```
cd kilogui-0.1
make all
```

Step 4: Environment setup

- Note: To allow for the gui to run as sudo, use nano to edit.

```
# go into root folder
cd ../..
# open environment using nano
sudo nano /etc/environment
```

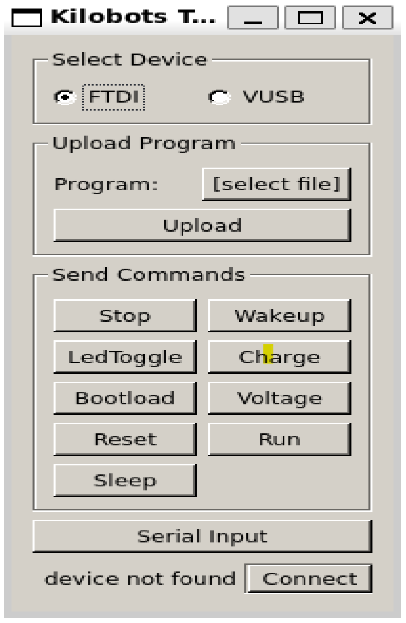
- Add `QT_X11_NO_MITSHM=1` to the file (Paste this line into the code) and save (hit ‘Ctrl+z’ to exit and press ‘Y’ to save)

Step 5: Run Kilogui

- Finally, navigate to the ‘kilogui-0.1’ directory and run Kilogui using the following command:

```
cd
cd kilogui-0.1/kilogui
sudo ./kilogui
```

- This will open the KiloGUI interface, as shown below:

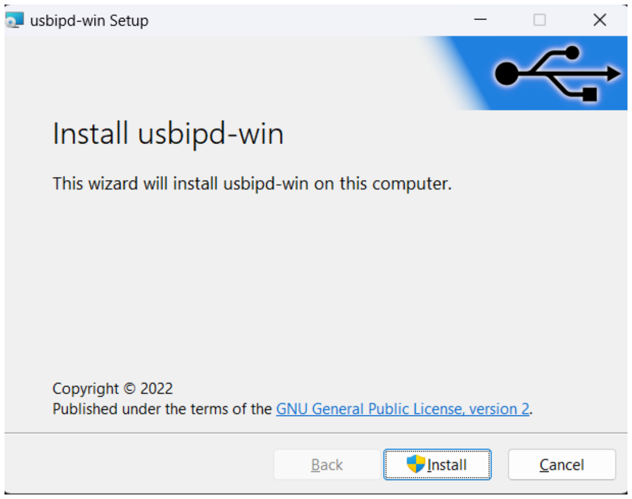


III. Setting up WSL to Recognize the USB Device (Arduino Controller)

In order to use the Overhead Controller with WSL, it is necessary to allow WSL to recognize the device. This process is based on the guide found *here*.

Step 1: Download the software:

- Download and run the .msi file from this *link*.


usbipd-win Setup

Install usbipd-win

This wizard will install usbipd-win on this computer.

Copyright © 2022
Published under the terms of the GNU General Public License, version 2.

Step 2: Restart the system.

- This is advised because the terminal may not recognize the newly installed ‘usbip’ without a restart.

Step 3: Driver Setup:

- In the WSL terminal (newly installed ubuntu terminal), run the following commands:

```
# command 1
sudo apt install linux-tools-5.4.0-77-generic hwdata
# command 2
sudo apt install linux-tools-virtual
# command 3
sudo update-alternatives --install /usr/local/bin/usbip usbip
/usr/lib/linux-tools/5.4.0-77-generic/usbip 20
```

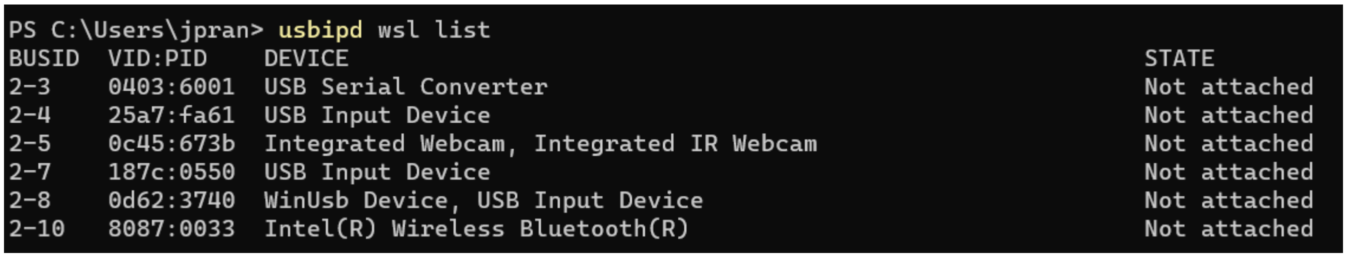
Step 4: Plug in the transmitter.

- Insert the ‘Overhead Controller’ into one of the USB slots in the Laptop.

Step 5: List USB devices:

- In the PowerShell terminal, run the following command to get the list of USB devices

```
usbipd wsl list
```

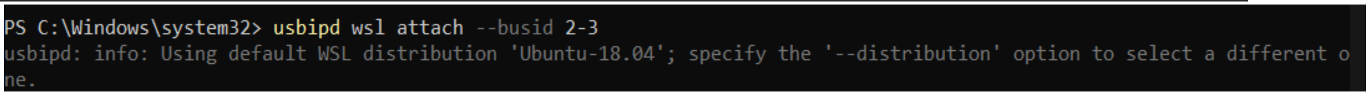


```
PS C:\Users\jpran> usbipd wsl list
BUSID  VID:PID    DEVICE                                                        STATE
2-3    0403:6001  USB Serial Converter                                          Not attached
2-4    25a7:fa61  USB Input Device                                              Not attached
2-5    0c45:673b  Integrated Webcam, Integrated IR Webcam                       Not attached
2-7    187c:0550  USB Input Device                                              Not attached
2-8    0d62:3740  WinUsb Device, USB Input Device                               Not attached
2-10   8087:0033  Intel(R) Wireless Bluetooth(R)                                Not attached
```

Step 6: Connect device

- Using the 'busid' found in the above command, run the following in PowerShell to attach the device (make sure the desired Ubuntu version is your default version):
- When running this command, if there is an administrator access issues, close the PowerShell and when opening the application, right-click on the app and choose 'Run as administrator' option. Then, run the same below code.

```
usbipd wsl attach --busid <busid>
```

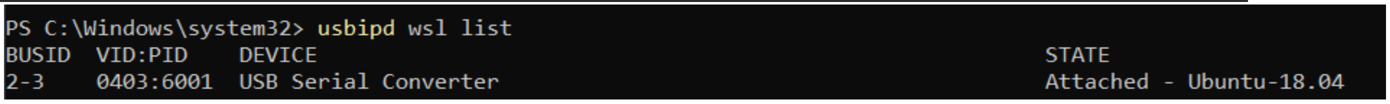


```
PS C:\Windows\system32> usbipd wsl attach --busid 2-3
usbipd: info: Using default WSL distribution 'Ubuntu-18.04'; specify the '--distribution' option to select a different o
ne.
```

Step 7: Cross-check the connection

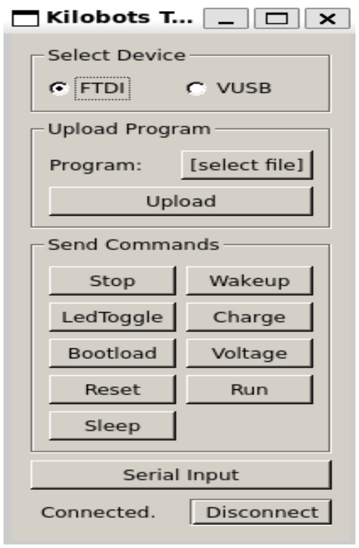
- Run 'usbipd' on WSL to ensure the device is connected.

```
usbipd wsl list
```



```
PS C:\Windows\system32> usbipd wsl list
BUSID  VID:PID    DEVICE                                                   STATE
2-3    0403:6001  USB Serial Converter                                     Attached - Ubuntu-18.04
```

Finally, this shows that the link has been established. When the KiloGUI is run, it will show you 'Connected' at the bottom.



Kilobots T...

Select Device: [✓ FTDI] VUSB

Upload Program

Program: [select file]

Connected.