

Rust on Raspberry PI Zero

User Guide

Software Requirements:

- 1) Windows 10 with last updates (2004)



- 2) PiDebug V0.0.1 Beta



- 3) VirtualBox 6.12



- 4) Ubuntu / Ubuntu Mate 18.0.4.5 LTS



- 5) Rust (Last version)



- 6) Raspberry PI Image Flasher



First Step (Prepare SSH)

- Flash SD-card via RPI Writer
- Open PiDebug and Press Edit Kernel Button.
- Select a boot volume and press OK
- Wait while text Patched ? : will changed and become olive green.



PiDebug Beta



Edit Kernal for run SSH

UserName

Patched?

Password

IP-Adress on SSH

Display Linux Log

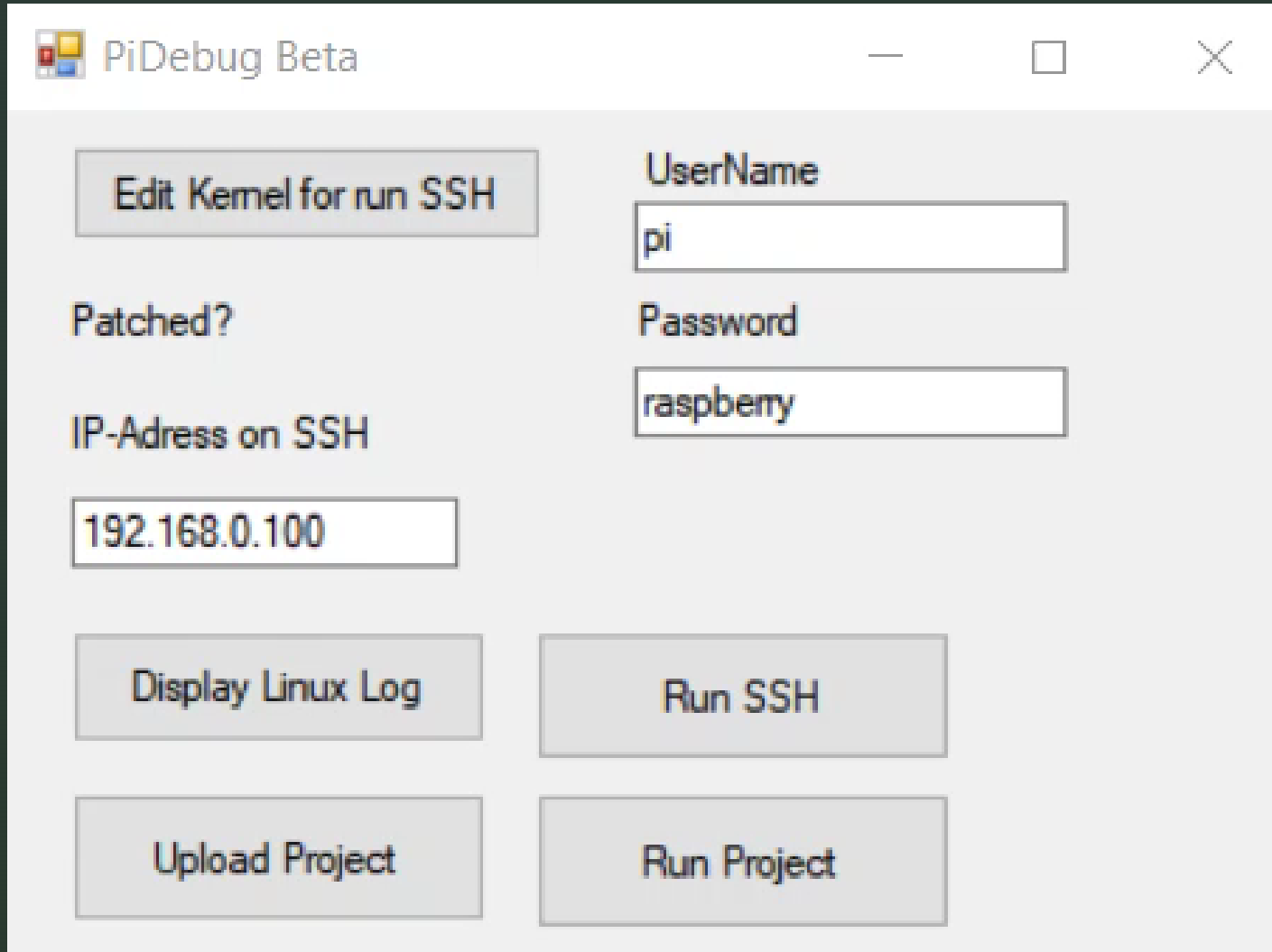
Run SSH

Upload Project

Run Project

Next Install SD card to RPI and connect in to Router.

After it enter all credits to PiDebug and check connection via getting Linux Log. After it connect via SSH.

The image shows a screenshot of the PiDebug Beta application window. The window has a title bar with the text "PiDebug Beta" and standard window controls (minimize, maximize, close). The main interface is light gray and contains several interactive elements. On the left side, there is a button labeled "Edit Kernel for run SSH". Below this, the text "Patched?" is displayed. Further down, the label "IP-Adress on SSH" is positioned above a text input field containing the IP address "192.168.0.100". At the bottom left, there are two buttons: "Display Linux Log" and "Upload Project". On the right side, there are two text input fields. The top one is labeled "UserName" and contains the text "pi". The bottom one is labeled "Password" and contains the text "raspberry". Below these input fields, there are two buttons: "Run SSH" and "Run Project".

PiDebug Beta

Edit Kernel for run SSH

UserName

pi

Password

raspberry

Patched?

IP-Adress on SSH

192.168.0.100

Display Linux Log

Run SSH

Upload Project

Run Project

Install&Configure Samba

- Install Samba by : `sudo apt-get install samba samba-common-bin`
- Create folder by : `sudo mkdir -m 1777 /share`
- Open Samba config by : `sudo nano /etc/samba/smb.conf`
- Add to bottom of file this config :

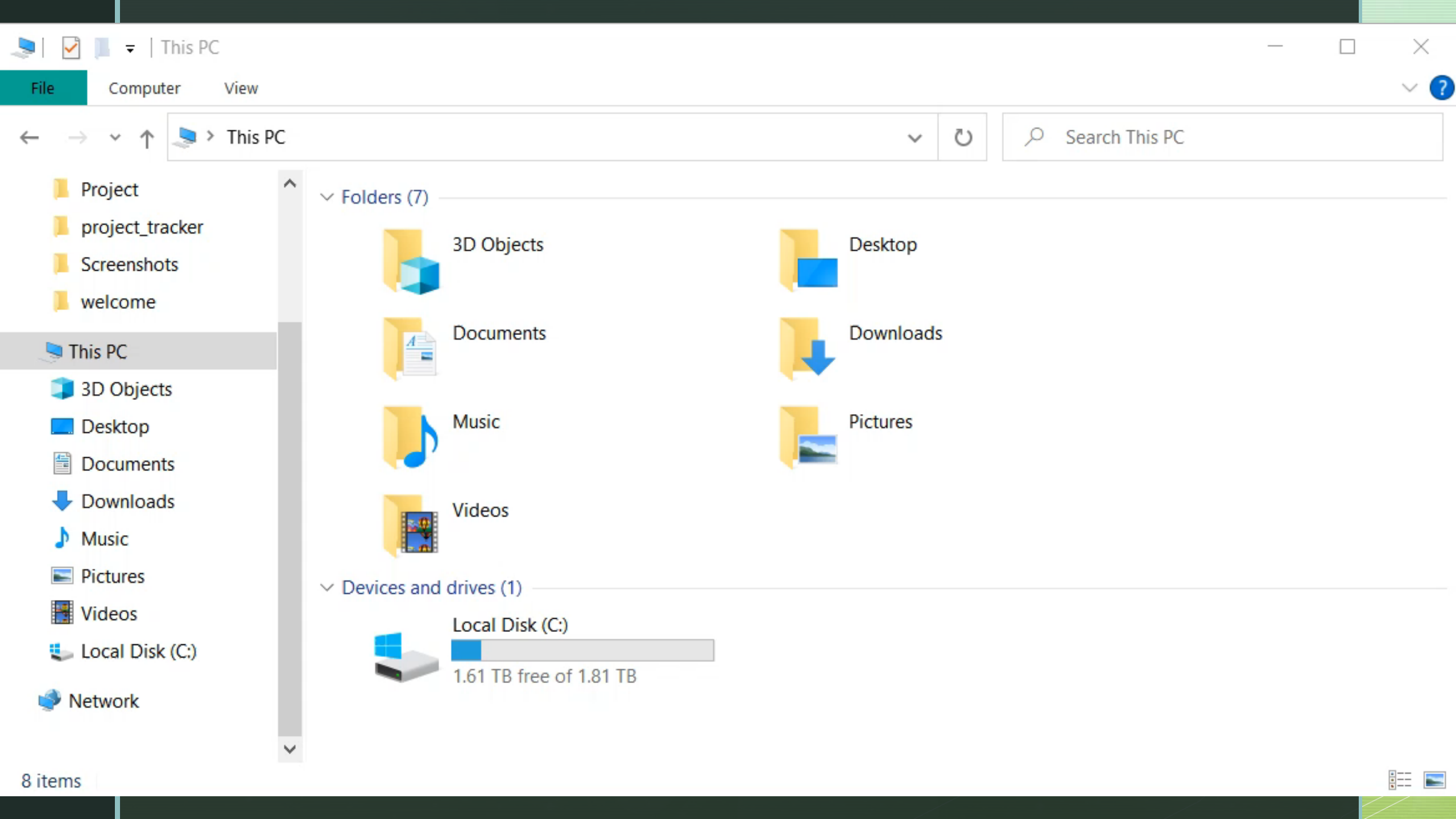
```
[share]
Comment = Pi shared folder
Path = /share
Browseable = yes
Writeable = Yes
only guest = no
create mask = 0777
directory mask = 0777
```

Reboot RPI after it

```
root@raspberrypi:/home#
```

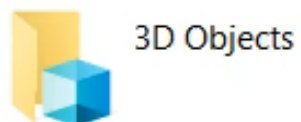


- Open Network Option in Explorer and try access to folder that Raspberry is sharing.

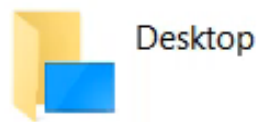


Project
project_tracker
Screenshots
welcome

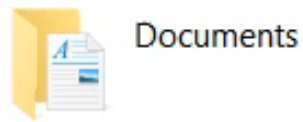
Folders (7)



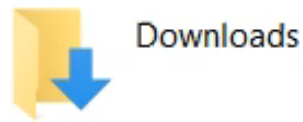
3D Objects



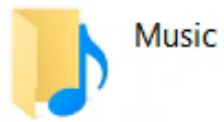
Desktop



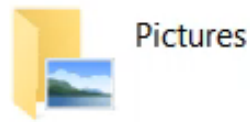
Documents



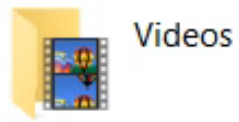
Downloads



Music



Pictures



Videos

Devices and drives (1)



Local Disk (C:)

1.61 TB free of 1.81 TB

- Create Virtual Machine and install Ubuntu.

- 1) Make : `sudo apt update`

- 2) Install VirtualBox Additions

- 3) Install rustup using : `curl https://sh.rustup.rs -sSf | sh`

- 4) Set default toolchain by : `rustup default stable`

- 5) Install all dependencies by : `sudo apt-get install gcc-arm-linux-gnueabi lib64-armhf-cross lib64-dev-armhf-cross`

- 6) Set another toolchain by : `rustup target add arm-unknown-linux-gnueabi`

- 7) In user folder create new folder .cargo

- 8) Step into it

- 9) Create file config.toml

- 10) Open it with nano by : `sudo nano config.toml`

- File content :

- `[target.arm-unknown-linux-gnueabi]`

- `linker = "$HOME/rpi_tools/arm-bcm2708/arm-linux-gnueabi/bin/arm-linux-gnueabi-gcc"`

- 1) Set one more toolchain by : `rustup target add arm-unknown-linux-gnueabi`

- 2) Get c++ linkers by : `git clone https://github.com/raspberrypi/tools $HOME/rpi_tools`

- 3) set it by : `RUSTFLAGS="-C linker=$HOME/rpi_tools/arm-bcm2708/arm-rpi-4.9.3-linux-gnueabi/bin/arm-linux-gnueabi-gcc" cargo build --target arm-unknown-linux-gnueabi --tests`

- 4) Finally build project using: `cargo build --target=arm-unknown-linux-gnueabi`



Trash



VBox_GAs_6.1.12

