nzk-scripts

- nzk-app-add
- install log fail
- py scripts from other repo's that i use (need to add to my repo)
- new ubuntu script
- recharjme
  - ubuntu kiosk with phidgethub using touch screen
  - restart chromium demon
  - Installer Demon Template

nzk-app-add

```
#!/bin/bash
#nzk-app-add
#i use kali linux
# Check for at least one argument
if [ "$#" -lt 1 ]; then
    echo "Usage: nzk-app-add <executable> [--icon <iconpath>]"
    exit 1
fi

EXECUTABLE=$(realpath "$1")
APP_NAME=$(basename "$EXECUTABLE")
ICON_PATH=""

# Check for optional icon argument
if [ "$2" == "--icon" ] && [ -n "$3" ]; then
    ICON_PATH=$(realpath "$3")
fi

# Create .desktop file
DESKTOP_ENTRY="$HOME/.local/share/applications/$APP_NAME.desktop"
echo "[Desktop Entry]" > "$DESKTOP_ENTRY"
echo "Type=Application" >> "$DESKTOP_ENTRY"
echo "Name=$APP_NAME" >> "$DESKTOP_ENTRY"
echo "Exec=$EXECUTABLE" >> "$DESKTOP_ENTRY"
echo "Icon=$ICON_PATH" >> "$DESKTOP_ENTRY"
echo "Terminal=false" >> "$DESKTOP_ENTRY"

# Make the .desktop file executable
chmod +x "$DESKTOP_ENTRY"

echo "$APP_NAME added to applications menu."
```

install log fail

```
┌──(kali㉿ kali)-[~]
└─$ cd nzk-apps

┌──(kali㉿ kali)-[~/nzk-apps]
└─$ ls
bash  install-NZK-Scripts.sh  LICENSE  nzk-code  README.md

┌──(kali㉿ kali)-[~/nzk-apps]
└─$ bash ./install-NZK-Scripts.sh
Are you NaruZKurai? (y/n)
If you aren't and you say yes, things will break.
So if you are you NaruZKurai, v1 will be installed and uses /home/kali as default user path
Please enter your answer: n
Cool, let's get started.
nzk-code3 copied successfully.
cp: cannot create regular file '/usr/local/bin/nzk-nature': Permission denied
Failed to copy nzk-nature. Ensure the file exists and try again.
Installation complete.

┌──(kali㉿ kali)-[~/nzk-apps]
└─$ cat ./bash/ nzk-nature
cat: ./bash/: Is a directory
cat: nzk-nature: No such file or directory

┌──(kali㉿ kali)-[~/nzk-apps]
└─$ cat ./bash/nzk-nature
#!/bin/bash
#nzk-nature
# Define associative arrays for natures and their properties
CONFIG_FILE="~/.config/nzk-apps/settings.conf"
declare -A nature_stats=(
    [0]="— —" [1]="Attack Defense" [2]="Attack Speed" [3]="Attack Sp. Attack" [4]="Attack Sp. Defense"
    [5]="Defense Attack" [6]="— —" [7]="Defense Speed" [8]="Defense Sp. Attack" [9]="Defense Sp. Defense"
    [10]="Speed Attack" [11]="Speed Defense" [12]="— —" [13]="Speed Sp. Attack" [14]="Speed Sp. Defense"
    [15]="Sp. Attack Attack" [16]="Sp. Attack Defense" [17]="Sp. Attack Speed" [18]="— —" [19]="Sp. Attack Sp. Defense"
    [20]="Sp. Defense Attack" [21]="Sp. Defense Defense" [22]="Sp. Defense Speed" [23]="Sp. Defense Sp. Attack" [24]="— —"
```

```
)
declare -A nature_flavors_english=(
    [0]="— —" [1]="Spicy Sour" [2]="Spicy Sweet" [3]="Spicy Dry" [4]="Spicy Bitter"
    [5]="Sour Spicy" [6]="— —" [7]="Sour Sweet" [8]="Sour Dry" [9]="Sour Bitter"
    [10]="Sweet Spicy" [11]="Sweet Sour" [12]="— —" [13]="Sweet Dry" [14]="Sweet Bitter"
    [15]="Dry Spicy" [16]="Dry Sour" [17]="Dry Sweet" [18]="— —" [19]="Dry Bitter"
    [20]="Bitter Spicy" [21]="Bitter Sour" [22]="Bitter Sweet" [23]="Bitter Dry" [24]="— —"
)
declare -A nature_flavors_japanese=(
    [0]="— —" [1]="からい　すっぱい　" [2]="からい　あまい　" [3]="からい　のま　" [4]="からい　にがい　"
    [5]="すっぱい　からい　" [6]="— —" [7]="すっぱい　あまい　" [8]="すっぱい　のま　" [9]="すっぱい　にがい　"
    [10]="あまい　からい　" [11]="あまい　すっぱい　" [12]="— —" [13]="あまい　のま　" [14]="あまい　にがい　"
    [15]="のま　からい　" [16]="のま　すっぱい　" [17]="のま　あまい　" [18]="— —" [19]="のま　にがい　"
    [20]="にがい　からい　" [21]="にがい　すっぱい　" [22]="にがい　あまい　" [23]="にがい　のま　" [24]="— —"
)
declare -A nature_flavors_romanji=(
    [0]="— —" [1]="Karaī Suppai" [2]="Karaī Amai" [3]="Karaī Noma" [4]="Karaī Nigai"
    [5]="Suppai Karaī" [6]="— —" [7]="Suppai Amai" [8]="Suppai Noma" [9]="Suppai Nigai"
    [10]="Amai Karaī" [11]="Amai Suppai" [12]="— —" [13]="Amai Noma" [14]="Amai Nigai"
    [15]="Noma Karaī" [16]="Noma Suppai" [17]="Noma Amai" [18]="— —" [19]="Noma Nigai"
    [20]="Nigai Karaī" [21]="Nigai Suppai" [22]="Nigai Amai" [23]="Nigai Noma" [24]="— —"
)
# Define associative arrays for English, Japanese, and Romanji nature names
declare -A english_natures=(
    [0]="Hardy" [1]="Lonely" [2]="Brave" [3]="Adamant" [4]="Naughty"
    [5]="Bold" [6]="Docile" [7]="Relaxed" [8]="Impish" [9]="Lax"
    [10]="Timid" [11]="Hasty" [12]="Serious" [13]="Jolly" [14]="Naive"
    [15]="Modest" [16]="Mild" [17]="Quiet" [18]="Bashful" [19]="Rash"
    [20]="Calm" [21]="Gentle" [22]="Sassy" [23]="Careful" [24]="Quirky"
)
declare -A japanese_natures=(
    [0]="がんばりや　" [1]="さみしがり　" [2]="ゆうかん　" [3]="いじっぱり　" [4]="やんちゃ　"
    [5]="ずぶとい　" [6]="すなお　" [7]="のんき　" [8]="わんぱく　" [9]="のうてんき　"
    [10]="おくびょう　" [11]="せっかち　" [12]="まじめ　" [13]="ようき　" [14]="むじゃき　"
    [15]="ひかえめ　" [16]="おっとり　" [17]="れいせい　" [18]="てれや　" [19]="うっかりや　"
    [20]="おだやか　" [21]="おとなしい　" [22]="なまいき　" [23]="しんちょう　" [24]="きまぐれ　"
)
declare -A romanji_natures=(
    [0]="Gannbariya" [1]="Samishigari" [2]="Yuukan" [3]="Ijippari" [4]="Yancha"
    [5]="Zubutoi" [6]="Sunao" [7]="Nonnki" [8]="Wanpaku" [9]="Noutennki"
    [10]="Okubiyou" [11]="Sekkachi" [12]="Majime" [13]="Youki" [14]="Mujyaki"
    [15]="Hikaeme" [16]="Ottori" [17]="Reisei" [18]="Tereya" [19]="Ukkariya"
    [20]="Odayaka" [21]="Otonashii" [22]="Namaiki" [23]="Shinnchyou" [24]="Kimagure"
)
# Function to find the index of a nature name, case-insensitive
find_nature_index() {
```

```
    local nature_name_lower=$(echo "$1" | tr '[:upper:]' '[:lower:]')
    local -n arr=$2
    for i in "${!arr[@]}"; do
        if [[ "$(echo "${arr[$i]}" | tr '[:upper:]' '[:lower:]')" == "$nature_name_lower" ]]; then
            echo $i
            return
        fi
    done
    echo "-1"
}

# Check for correct number of arguments
if [ $# -ne 2 ]; then
    echo "Usage: $0 <nature_name> [-e|-j|-r]"
    exit 1
fi

# Get the nature name and flag
nature_name="$1"
flag="$2"

# Main logic to get nature information based on flag and nature name
case $flag in
    -e)
        index=$(find_nature_index "$nature_name" english_natures)
        if [[ $index -ne -1 ]]; then
            echo "${japanese_natures[$index]}"
        else
            echo "Error: Nature '$nature_name' not found in English natures."
        fi
        ;;
    -j)
        index=$(find_nature_index "$nature_name" japanese_natures)
        if [[ $index -ne -1 ]]; then
            echo "${english_natures[$index]}"
        else
            echo "Error: Nature '$nature_name' not found in Japanese natures."
        fi
        ;;
    -r)
        index=$(find_nature_index "$nature_name" romanji_natures)
        if [[ $index -ne -1 ]]; then
            echo "${english_natures[$index]}"
        else
            echo "Error: Nature '$nature_name' not found in Romanji natures."
        fi
```

```
        ;;
    *)
        echo "Invalid flag: $flag. Use -e for English to Japanese, -j for Japanese to English, -r for
Romanji to English."
        echo "correct usage would be something like:"
        echo "nzk-nature "jolly" -e"
        exit 1
        ;;
esac

┌──(kali kali)-[~/nzk-apps]
└─$
```

py scripts from other repo's that i use (need to add to my repo)

https://github.com/Crypto-Cat/CTF/blob/main/pentesting/gen_nmap.py

```
#!/bin/python
#
# can be found on crypto cat's repo,
# https://github.com/Crypto-Cat/CTF/blob/main/pentesting/gen_nmap.py
#
# sudo apt-get install python3 masscan nmap
#
# This script will take in Masscan results and produce an output which can be fed into NMap

# Save your Masscan result as mscan.txt then run this script to produce nmap.txt then you can run:
# while read item; do sudo nmap -sV -sC -sS -sU $item; done < nmap.txt; rm mscan.txt nmap.txt

# If you want to export to .xml file you can use the following command and then later use this script to merge files: https://github.com/sidaf/scripts/blob/master/nmap_merge.py
# while read item; do filename=$(echo $item | grep -o "^\S*"); sudo nmap -O -sV -sC -sS -sU $item -oX $filename.xml; done < nmap.txt

import re
from socket import inet_aton
from os import path

regex = re.compile(r"Discovered open port (\d+)\/(udp|tcp) on (\d+\.\d+\.\d+\.\d+)", re.I)

ip_list = {}

with open(path.abspath('mscan.txt')) as f:
    lines = f.readlines()
    for line in lines:
        port = regex.match(line).group(1)
        protocol = regex.match(line).group(2)
```

```
        ip = regex.match(line).group(3)

        # Add the IP to dictionary if it's not already
        try:
            ip_list[ip]
        except KeyError:
            ip_list[ip] = {}

        # Add protocol to dictionary if it's not already
        try:
            ip_list[ip][protocol]
        except KeyError:
            ip_list[ip][protocol] = []

        # Append the port to the list
        ip_list[ip][protocol].append(port)

with open(path.abspath('nmap.txt'), 'a') as f:
    sorted_ips = sorted(ip_list.items(), key=lambda item: inet_aton(item[0]))
    for ip, protocols in sorted_ips:
        udp_ports = ""
        tcp_ports = ""

        # Check to see if any UDP ports were found
        try:
            for port in protocols['udp']:
                udp_ports += port + ','
        except KeyError:
            pass

        # Check to see if any TCP ports were found
        try:
            for port in protocols['tcp']:
                tcp_ports += port + ','
        except KeyError:
            pass

        # Print IP and ports to file ready for NMap scan
        if udp_ports and tcp_ports:
            line = ip + ' -p U:' + udp_ports + 'T:' + tcp_ports
```

```
        elif udp_ports:
            line = ip + ' -p U:' + udp_ports
        elif tcp_ports:
            line = ip + ' -p T:' + tcp_ports
        f.write(line + '\n')
```

new ubuntu script

```
sudo apt update && sudo apt install -y firefox ubuntu-restricted-extras build-essential curl wget software-properties-common kde-plasma-desktop inkscape gimp openscad meshlab librecad python3 python3-tk python3-pip flatpak gnome-software-plugin-flatpak && \
[ -f /tmp/chrome.deb ] || wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb -O /tmp/chrome.deb && sudo apt install -y /tmp/chrome.deb || sudo apt --fix-broken install -y && \
sudo ubuntu-drivers install && \
sudo dpkg --add-architecture i386 && sudo apt update && sudo apt install -y wine64 wine32 winetricks lutris && \
[ -f /tmp/obsidian.deb ] || wget "$(curl -s https://api.github.com/repos/obsidianmd/obsidian-releases/releases/latest | grep browser_download_url | grep amd64.deb | cut -d '"' -f 4)" -O /tmp/obsidian.deb && sudo apt install -y /tmp/obsidian.deb && \
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor | sudo tee /usr/share/keyrings/vscode.gpg > /dev/null && \
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/vscode.gpg] https://packages.microsoft.com/repos/vscode stable main" | sudo tee /etc/apt/sources.list.d/vscode.list > /dev/null && \
sudo apt update && sudo apt install -y code && \
flatpak remote-add -y --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo && \
flatpak install -y flathub com.ultimaker.cura org.freecadweb.FreeCAD org.kicad.KiCad org.blender.Blender com.github.tchx84.Flatseal com.prusa3d.PrusaSlicer && \
sudo pip3 install bCNC && \
[ -f /usr/local/bin/OpenBuildsCONTROL.AppImage ] || wget https://github.com/OpenBuilds/OpenBuilds-Control/releases/latest/download/OpenBuildsCONTROL.AppImage -O /usr/local/bin/OpenBuildsCONTROL.AppImage && chmod +x /usr/local/bin/OpenBuildsCONTROL.AppImage && \
[ -f /tmp/ugs.tar.gz ] || wget https://github.com/winder/Universal-G-Code-Sender/releases/latest/download/ugsplatform-linux.tar.gz -O /tmp/ugs.tar.gz && mkdir -p /opt/UGS && tar -xzf /tmp/ugs.tar.gz -C /opt/UGS && \
WINEPREFIX=~/.wine-sparkmax winecfg && \
[ -f ~/sparkmax-client-latest.msi ] || wget https://www.revrobotics.com/content/sw/max/sparkmax-client-latest.msi -O ~/sparkmax-client-latest.msi && \
```

```
WINEPREFIX=~/.wine-sparkmax wine msiexec /i ~/sparkmax-client-latest.msi && \
sudo mkdir -p /etc/skel/Desktop && cd /etc/skel/Desktop && \
for app in firefox code obsidian prusa-slicer cura freecad kicad blender gimp inkscape bCNC
wine OpenBuildsCONTROL.AppImage; do \
  echo "[Desktop Entry]\nName=$app\nExec=$(command -v $app || echo $app)\nIcon=utilities-
terminal\nType=Application\nTerminal=false" | sudo tee "$app.desktop" > /dev/null; \
  chmod +x "$app.desktop"; \
done && \
[ -x /usr/local/bin/OpenBuildsCONTROL.AppImage ] || /usr/local/bin/OpenBuildsCONTROL.AppImage
&& \
[ -d /opt/UGS ] || xdg-open https://github.com/winder/Universal-G-Code-Sender/releases
```

orignial@Robotronix-cad-orignial:~$ sudo apt update && sudo apt install -y firefox ubuntu-restricted-extras build-essential curl wget software-properties-common kde-plasma-desktop inkscape gimp openscad meshlab librecad python3 python3-tk python3-pip flatpak gnome-software-plugin-flatpak && \
[-f /tmp/chrome.deb] || wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb -O /tmp/chrome.deb && sudo apt install -y /tmp/chrome.deb || sudo apt --fix-broken install -y && \
sudo ubuntu-drivers install && \
sudo dpkg --add-architecture i386 && sudo apt update && sudo apt install -y wine64 wine32 winetricks lutris && \
[-f /tmp/obsidian.deb] || wget "$(curl -s https://api.github.com/repos/obsidianmd/obsidian-releases/releases/latest | grep browser_download_url | grep amd64.deb | cut -d '"' -f 4)" -O /tmp/obsidian.deb && sudo apt install -y /tmp/obsidian.deb && \
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor | sudo tee /usr/share/keyrings/vscode.gpg > /dev/null && \
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/vscode.gpg] https://packages.microsoft.com/repos/vscode stable main" | sudo tee /etc/apt/sources.list.d/vscode.list > /dev/null && \
sudo apt update && sudo apt install -y code && \
flatpak remote-add -y --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo && \
flatpak install -y flathub com.ultimaker.cura org.freecadweb.FreeCAD org.kicad.KiCad org.blender.Blender com.github.tchx84.Flatseal com.prusa3d.PrusaSlicer && \leasesopt/UGS] || xdg-open https://github.com/winder/Universal-G-Code-Sender/re
Hit:1 http://ca.archive.ubuntu.com/ubuntu plucky InRelease
Hit:2 http://security.ubuntu.com/ubuntu plucky-security InRelease
Hit:3 http://ca.archive.ubuntu.com/ubuntu plucky-updates InRelease
Hit:4 http://ca.archive.ubuntu.com/ubuntu plucky-backports InRelease
Hit:5 https://dl.google.com/linux/chrome/deb stable InRelease

```
All packages are up to date.
Notice: Some sources can be modernized. Run 'apt modernize-sources' to do so.
firefox is already the newest version (1:1snap1-0ubuntu7).
ubuntu-restricted-extras is already the newest version (67).
build-essential is already the newest version (12.10ubuntu1).
curl is already the newest version (8.12.1-3ubuntu1).
wget is already the newest version (1.24.5-2ubuntu1).
software-properties-common is already the newest version (0.109).
kde-plasma-desktop is already the newest version (5:159ubuntu1).
inkscape is already the newest version (1.2.2-8build1).
gimp is already the newest version (3.0.0-2).
openscad is already the newest version (2021.01-8build1).
meshlab is already the newest version (2022.02+dfsg1-1).
librecad is already the newest version (2.2.0.2-1build3).
python3 is already the newest version (3.13.2-2).
python3-tk is already the newest version (3.13.2-1).
python3-pip is already the newest version (25.0+dfsg-1).
flatpak is already the newest version (1.16.0-2).
gnome-software-plugin-flatpak is already the newest version (48.0-1).
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Note, selecting 'google-chrome-stable' instead of '/tmp/chrome.deb'
google-chrome-stable is already the newest version (134.0.6998.165-1).
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0

All the available drivers are already installed.
Get:1 http://ca.archive.ubuntu.com/ubuntu plucky InRelease [265 kB]
Hit:2 http://security.ubuntu.com/ubuntu plucky-security InRelease
Hit:3 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:4 http://ca.archive.ubuntu.com/ubuntu plucky-updates InRelease
Hit:5 http://ca.archive.ubuntu.com/ubuntu plucky-backports InRelease
Get:6 http://ca.archive.ubuntu.com/ubuntu plucky/main amd64 Packages [1,437 kB]
Get:7 http://ca.archive.ubuntu.com/ubuntu plucky/main i386 Packages [1,072 kB]
Get:8 http://ca.archive.ubuntu.com/ubuntu plucky/main Translation-en [521 kB]
Get:9 http://ca.archive.ubuntu.com/ubuntu plucky/main amd64 c-n-f Metadata [31.7 kB]
Get:10 http://ca.archive.ubuntu.com/ubuntu plucky/universe i386 Packages [8,852 kB]
Get:11 http://ca.archive.ubuntu.com/ubuntu plucky/universe amd64 Packages [15.9 MB]
```

```
Get:12 http://ca.archive.ubuntu.com/ubuntu plucky/universe Translation-en [6,266 kB]
Get:13 http://ca.archive.ubuntu.com/ubuntu plucky/universe amd64 c-n-f Metadata [309 kB]
Fetched 34.6 MB in 12s (2,909 kB/s)
All packages are up to date.
Notice: Some sources can be modernized. Run 'apt modernize-sources' to do so.
wine64 is already the newest version (9.0~repack-4build3).
wine32:i386 is already the newest version (9.0~repack-4build3).
winetricks is already the newest version (20250102-1).
lutris is already the newest version (0.5.17-2).
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'sudo apt autoremove' to remove it.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
Reading package lists... Error!
Error: read, still have 8 to read but none left
Error: Internal error, could not locate member control.tar{.zst,.lz4,.gz,.xz,.bz2,.lzma,}
Error: Could not read meta data from /tmp/obsidian.deb
Error: The package lists or status file could not be parsed or opened.
/usr/local/bin/OpenBuildsCONTROL.AppImage: Permission denied
--2025-03-22 04:22:47--  https://github.com/winder/Universal-G-Code-
Sender/releases/latest/download/ugsplatform-linux.tar.gz
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github.com/winder/Universal-G-Code-
Sender/releases/download/v2.1.12/ugsplatform-linux.tar.gz [following]
--2025-03-22 04:22:47--  https://github.com/winder/Universal-G-Code-
Sender/releases/download/v2.1.12/ugsplatform-linux.tar.gz
Reusing existing connection to github.com:443.
HTTP request sent, awaiting response... 404 Not Found
2025-03-22 04:22:47 ERROR 404: Not Found.

--2025-03-22 04:22:47--  https://www.revrobotics.com/content/sw/max/sparkmax-client-latest.msi
Resolving www.revrobotics.com (www.revrobotics.com)... 192.200.160.248
Connecting to www.revrobotics.com (www.revrobotics.com)|192.200.160.248|:443... connected.
HTTP request sent, awaiting response... 404 Not Found
2025-03-22 04:22:48 ERROR 404: Not Found.

bash: /usr/local/bin/OpenBuildsCONTROL.AppImage: No such file or directory
orignial@Robotronix-cad-orignial:~$ Gtk-Message: 04:22:48.965: Not loading module "atk-bridge":
The functionality is provided by GTK natively. Please try to not load it.
```

```
install_if_missing() { pkgs=""; for p in "$@"; do dpkg -s "$p" &>/dev/null || pkgs="$pkgs $p"; done; [ -n "$pkgs" ] && sudo apt install -y $pkgs; } && \
sudo apt update && \
install_if_missing firefox ubuntu-restricted-extras build-essential curl wget software-properties-common kde-plasma-desktop inkscape gimp openscad meshlab librecad python3 python3-tk python3-pip flatpak gnome-software-plugin-flatpak && \
([ -f /tmp/chrome.deb ] || wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb -O /tmp/chrome.deb) && sudo apt install -y /tmp/chrome.deb || sudo apt --fix-broken install -y && \
sudo ubuntu-drivers install && \
sudo dpkg --add-architecture i386 && sudo apt update && \
install_if_missing wine64 wine32 winetricks lutris && \
[ -f /tmp/obsidian.deb ] || wget "$(curl -s https://api.github.com/repos/obsidianmd/obsidian-releases/releases/latest | grep browser_download_url | grep amd64.deb | cut -d '"' -f 4)" -O /tmp/obsidian.deb && sudo apt install -y /tmp/obsidian.deb && \
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | gpg --dearmor | sudo tee /usr/share/keyrings/vscode.gpg > /dev/null && \
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/vscode.gpg] https://packages.microsoft.com/repos/vscode stable main" | sudo tee /etc/apt/sources.list.d/vscode.list > /dev/null && \
sudo apt update && install_if_missing code && \
flatpak remote-add -y --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo && \
flatpak install -y flathub com.ultimaker.cura org.freecadweb.FreeCAD org.kicad.KiCad org.blender.Blender com.github.tchx84.Flatseal com.prusa3d.PrusaSlicer && \
sudo pip3 install bCNC && \
[ -f /usr/local/bin/OpenBuildsCONTROL.AppImage ] || { sudo wget https://github.com/OpenBuilds/OpenBuilds-Control/releases/latest/download/OpenBuildsCONTROL.AppImage -O /usr/local/bin/OpenBuildsCONTROL.AppImage && sudo chmod +x /usr/local/bin/OpenBuildsCONTROL.AppImage; } && \
UGS_URL="$(curl -s https://api.github.com/repos/winder/Universal-G-Code-Sender/releases/latest | grep browser_download_url | grep 'ugsplatform-linux.tar.gz' | cut -d '\"' -f 4)" && \
[ -f /tmp/ugs.tar.gz ] || wget "$UGS_URL" -O /tmp/ugs.tar.gz && mkdir -p /opt/UGS && tar -xzf /tmp/ugs.tar.gz -C /opt/UGS && \
WINEPREFIX=~/.wine-sparkmax winecfg && \
[ -f ~/REV-Hardware-Client.exe ] || wget "$(curl -s https://api.github.com/repos/REVrobotics/REV-Software-Binaries/releases/latest | grep
```

```
browser_download_url | grep 'REV-Hardware-Client-Setup' | grep exe | cut -d '"' -f 4)" -O
~/REV-Hardware-Client.exe && \
WINEPREFIX=~/.wine-sparkmax wine ~/REV-Hardware-Client.exe && \
sudo mkdir -p /etc/skel/Desktop && cd /etc/skel/Desktop && \
for app in firefox code obsidian prusa-slicer cura freecad kicad blender gimp inkscape bCNC
wine OpenBuildsCONTROL.AppImage; do \
  echo "[Desktop Entry]\nName=$app\nExec=$(command -v $app || echo $app)\nIcon=utilities-
terminal\nType=Application\nTerminal=false" | sudo tee "$app.desktop" > /dev/null; \
  chmod +x "$app.desktop"; \
done && \
[ -x /usr/local/bin/OpenBuildsCONTROL.AppImage ] || /usr/local/bin/OpenBuildsCONTROL.AppImage
&& \
[ -d /opt/UGS ] || xdg-open https://github.com/winder/Universal-G-Code-Sender/releases
```

recharjme

recharjme

ubuntu kiosk with phidgethub using touch screen

```
#!/bin/bash
set -eu
# Enable pipefail if supported
if set -o | grep -q pipefail; then
    set -o pipefail
fi

# Function definitions for each section
section_1() {
    #read "Press [Enter] key to continue... stopping screen blanking (best effort for GNOME;
harmless on Plasma)"
    ##############################################
    # SECTION 1: POWER MANAGEMENT AND SCREEN BLANKING for current user
    ##############################################
    echo "=== SECTION 1: POWER MANAGEMENT AND SCREEN BLANKING ==="
    # Stop screen blanking/suspend for kiosk session (no-op if schema absent)
    current_user=$(logname 2>/dev/null || echo "$SUDO_USER" || echo "$USER")
    sudo -u "$current_user" dbus-run-session gsettings set org.gnome.desktop.session idle-
delay 0 || true
    sudo -u "$current_user" dbus-run-session gsettings set org.gnome.settings-
daemon.plugins.power sleep-inactive-ac-type 'nothing' || true
    sudo -u "$current_user" dbus-run-session gsettings set org.gnome.settings-
daemon.plugins.power sleep-inactive-battery-type 'nothing' || true
}

section_2() {
    ##############################################
    # SECTION 2: INITIAL SYSTEM SETUP
    ##############################################
    echo "=== SECTION 2: INITIAL SYSTEM SETUP ==="
```

```
    apt-get update -yq
    apt-get upgrade -yq
    apt-get update -yq
    apt install -yq kde-plasma-desktop chromium-browser xdotool unclutter sed curl libinput-
tools net-tools ssh

    #wait for user input
    #read "Press [Enter] key to continue... try gdm3, ssdm has issues with my app"
}

section_3() {
    ########################################################
    # SECTION 3: DESKTOP ENVIRONMENT INSTALLATION
    ########################################################
    echo "=== SECTION 3: DESKTOP ENVIRONMENT INSTALLATION ==="
    systemctl enable ssh

    #wait for user input
    #read "Press [Enter] key to continue... plasma installed. installing other stuff for
phidget and enabling ssh  "
}

section_4() {
    ##############################################
    # SECTION 4: PHIDGET SETUP AND DEPENDENCIES
    ##############################################
    echo "=== SECTION 4: PHIDGET SETUP AND DEPENDENCIES ==="
    apt install -y chromium-browser sed xdotool unclutter
    curl -fsSL https://www.phidgets.com/downloads/setup_linux | bash -
    apt install -y libphidget22 phidget22networkserver
    apt install -y libphidget22* phidget22*
    apt install -y python3 npm nodejs
    ifconfig
}

section_5() {
    ##############################################
    # SECTION 5: TOUCHSCREEN CONFIGURATION
    ##############################################
    echo "=== SECTION 5: TOUCHSCREEN CONFIGURATION ==="
```

```
    #setup touchscreen
    echo -e "usbtouchscreen\nusbhid" |  tee /etc/modules-load.d/touchscreen.conf
    modprobe usbtouchscreen
    modprobe usbhid
    lsmod | grep usb
    lsmod | grep hid
    lsmod | grep usb
    lsmod | grep hid
    dmesg | grep -i touch
}

section_6() {
    ##############################################
    # SECTION 6: PHIDGET NETWORK SERVER SETUP
    ##############################################
    echo "=== SECTION 6: PHIDGET NETWORK SERVER SETUP ==="
    #setup phidget22networkserver
    DEFAULT_PHIDGET_PORT_VAR=""
    DEFAULT_WEB_PORT_VAR=""
    SERVICE_NAME="phidget22networkserver"
    INIT_PATH="/etc/init.d/$SERVICE_NAME"
    #v1 (works???)
    if [[ "$(id -u)" -ne 0 ]]; then
      echo "Run this as root."
      exit 1
    fi

    cat > "$INIT_PATH" <<'EOF'
#!/bin/sh
### BEGIN INIT INFO
# Provides:          phidget22networkserver
# Required-Start:    $network $remote_fs
# Required-Stop:     $network $remote_fs
# Should-Start:      avahi avahi-daemon
# Should-Stop:       avahi avahi-daemon
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Phidget Network Server
# Description:       Phidget network server for accessing Phidgets over the network.
### END INIT INFO
```

```
sudo phidget22networkserver -c /etc/phidgets/phidget22networkserver.pc -D

EOF

    #add ipv6 to phidget config??? { port: 8080 }

    chmod +x "$INIT_PATH"
    sudo sed -i "s#docroot: '/var/phidgets/www'#docroot: '/home/recharjme/Desktop/APP'#"
/etc/phidgets/phidget22networkserver.pc
    update-rc.d "$SERVICE_NAME" defaults
    service "$SERVICE_NAME" start
    systemctl daemon-reload
    systemctl enable "$SERVICE_NAME"
    systemctl restart "$SERVICE_NAME"
    #systemctl status "$SERVICE_NAME"

    #wait for user input
   # read "Press [Enter] key to continue... this is usuially where shit breaks"
}

section_7() {
    ##############################################
    # SECTION 7: KIOSK USER SETUP
    ##############################################
    echo "=== SECTION 7: KIOSK USER SETUP ==="
    # 1) Ensure kiosk exists
    if ! id kiosk &>/dev/null; then adduser --disabled-password --gecos "" kiosk; fi
    passwd -d kiosk || true
    usermod -aG sudo kiosk

    install -d -m 0755 /home/kiosk/.config/autostart
    chown -R kiosk:kiosk /home/kiosk

    #read "Press [Enter] key to continue... touch works now, configuring GDM for Plasma
Wayland"

    # Force Wayland in GDM for admin acct
    sed -i 's/^#\?WaylandEnable=.*/WaylandEnable=true/' /etc/gdm3/custom.conf
    grep WaylandEnable /etc/gdm3/custom.conf || true
```

```
    #read "Press [Enter] key to continue... setting autologin for GDM3 (Plasma Wayland)"

    sudo -u kiosk kwriteconfig6 --file startkderc --group General --key desktopSession empty
    sudo -u kiosk kwriteconfig6 --file ksmserverrc --group General --key loginMode
emptySession

}

section_8() {
    ##############################################
    # SECTION 8: GDM3 AUTOLOGIN CONFIGURATION
    ##############################################
    echo "=== SECTION 8: GDM3 AUTOLOGIN CONFIGURATION ==="
    # GDM3 Autologin to Plasma Wayland
    if [ -f /etc/gdm3/custom.conf ]; then
      awk '
        BEGIN{inDaemon=0; hasALE=0; hasAL=0; hasDS=0}
        /^\[daemon\]/{inDaemon=1; print; next}
        /^\[/{ if(inDaemon && !(hasALE&&hasAL&&hasDS)){print
"AutomaticLoginEnable=true\nAutomaticLogin=kiosk\nDefaultSession=plasma"}; inDaemon=0; print;
next}
        {
          if(inDaemon){
            if($0 ~ /^AutomaticLoginEnable=/){$0="AutomaticLoginEnable=true"; hasALE=1}
            if($0 ~ /^AutomaticLogin=/){$0="AutomaticLogin=kiosk"; hasAL=1}
            if($0 ~ /^DefaultSession=/){$0="DefaultSession=plasma"; hasDS=1}
          }
          print
        }
        END{
          if(inDaemon && !(hasALE&&hasAL&&hasDS)){
            print "AutomaticLoginEnable=true"
            print "AutomaticLogin=kiosk"
            print "DefaultSession=plasma"
          }
        }
      ' /etc/gdm3/custom.conf > /tmp/custom.conf && mv /tmp/custom.conf /etc/gdm3/custom.conf

      # Make sure AccountsService maps kiosk -> plasma
```

```
      install -d -m 0755 /var/lib/AccountsService/users
      cat >/var/lib/AccountsService/users/kiosk <<'EOF'
[User]
XSession=plasma
SystemAccount=false
EOF
      chmod 0644 /var/lib/AccountsService/users/kiosk
      #systemctl restart gdm3
    fi
}

section_9() {
    ##############################################
    # SECTION 9: SDDM CONFIGURATION (ALTERNATIVE)
    ##############################################
    echo "=== SECTION 9: SDDM CONFIGURATION (ALTERNATIVE) ==="

    # Make sure /etc/sddm.conf exists
    [ -f /etc/sddm.conf ] || touch /etc/sddm.conf

    # Ensure autologin block exists and points to kiosk + plasma
    if grep -q '^\[Autologin\]' /etc/sddm.conf; then
        sed -i '/^\[Autologin\]/,/^\[/{s/^User.*/User=kiosk/;s/^Session.*/Session=plasma/}'
/etc/sddm.conf
    else
        tee -a /etc/sddm.conf >/dev/null <<'EOF'
[Autologin]
User=kiosk
Session=plasma
EOF
    fi

    # Switch DM to SDDM
    echo "sddm shared/default-x-display-manager select sddm" | sudo debconf-set-selections
    sudo dpkg-reconfigure -fnoninteractive sddm
    systemctl disable gdm3 || true
    systemctl enable sddm
    systemctl set-default graphical.target
    #systemctl restart sddm
```

```
    # Show resulting configs
    [ -f /etc/gdm3/custom.conf ] && cat /etc/gdm3/custom.conf || true
    [ -f /etc/sddm.conf ] && cat /etc/sddm.conf || true

    #read "Press [Enter] key to continue... creating Chromium autostart"
}

section_10() {
    ##############################################
    # SECTION 10: CHROMIUM KIOSK AUTOSTART SETUP
    ##############################################
    echo "=== SECTION 10: CHROMIUM KIOSK AUTOSTART SETUP ==="
    # Create Chromium kiosk autostart for Plasma session
    APP_URL="http://localhost:8080"
    cat >/home/kiosk/.config/autostart/chromium-kiosk.desktop <<EOF
[Desktop Entry]
Type=Application
Name=Chromium Kiosk
Exec=/usr/bin/chromium-browser --ozone-platform=wayland --enable-
features=UseOzonePlatform,WaylandWindowDecorations --no-first-run --no-default-browser-check
--disable-session-crashed-bubble --disable-infobars --start-maximized --noerrdialogs --
incognito --kiosk "$APP_URL"
X-GNOME-Autostart-enabled=true
X-KDE-AutostartScript=true
EOF
    chown kiosk:kiosk /home/kiosk/.config/autostart/chromium-kiosk.desktop
    chmod 0644 /home/kiosk/.config/autostart/chromium-kiosk.desktop
}

section_11() {
    #read "Press [Enter] key to continue... verifying Wayland toggles and kiosk autostart
path"
    ##############################################
    # SECTION 11: WAYLAND AND KIOSK VERIFICATION
    ##############################################
    echo "=== SECTION 11: WAYLAND AND KIOSK VERIFICATION ==="
    # Ensure WaylandEnable=false is commented if present
    if grep -q '^WaylandEnable=false' /etc/gdm3/custom.conf; then
      sed -i 's/^WaylandEnable=false/#WaylandEnable=false/' /etc/gdm3/custom.conf
    fi
```

```
    # Make sure APP_URL uses http (Phidget web piece typically)
    sed -i 's#https://localhost:8080#http://localhost:8080#g'
/home/kiosk/.config/autostart/chromium-kiosk.desktop
    chown kiosk:kiosk /home/kiosk/.config/autostart/chromium-kiosk.desktop

}

section_12() {
    #read "Press [Enter] key to continue... disabling touch gestures (GNOME keys; ignored on
Plasma)"
    ##############################################
    # SECTION 12: TOUCH GESTURE CONFIGURATION
    ##############################################
    echo "=== SECTION 12: TOUCH GESTURE CONFIGURATION ==="
    # Remove Wayland touch shortcuts (GNOME dconf; harmless if schema missing)
    gsettings set org.gnome.desktop.peripherals.touchpad three-finger-swipe-enabled false ||
true
    gsettings set org.gnome.desktop.peripherals.touchpad pinch-to-zoom false || true
}

section_13() {
    #read "Press [Enter] key to continue... applying KWin tweaks"
    echo "=== SECTION 13: KWIN TWEAKS AND FINAL SETUP ==="

    kwriteconfig6 --file kwinrc --group "Global Shortcuts" --key _k_friendly_name "" || true

    mkdir -p ~/.config/autostart-scripts
    cat > ~/.config/autostart-scripts/reload-kwin.sh <<'EOF'
#!/bin/sh
(qdbus6 org.kde.KWin /KWin reconfigure || qdbus6 org.kde.KWin.Wayland /KWin reconfigure) ||
true
EOF
    chmod +x ~/.config/autostart-scripts/reload-kwin.sh
}

section_14() {
    #read "Press [Enter] key to continue... stopping screen blanking for kiosk user (best
effort for GNOME; harmless on Plasma)"
    ##############################################
```

```
    # SECTION 14: POWER MANAGEMENT AND SCREEN BLANKING for current user
    ###############################################
    echo "=== SECTION 1: POWER MANAGEMENT AND SCREEN BLANKING ==="
    # Stop screen blanking/suspend for kiosk session (no-op if schema absent)
    sudo -u kiosk dbus-run-session gsettings set org.gnome.desktop.session idle-delay 0 ||
true
    sudo -u kiosk dbus-run-session gsettings set org.gnome.settings-daemon.plugins.power
sleep-inactive-ac-type 'nothing' || true
    sudo -u kiosk dbus-run-session gsettings set org.gnome.settings-daemon.plugins.power
sleep-inactive-battery-type 'nothing' || true

}
section_15() {
    ###############################################
    # SECTION 15: FINAL SYSTEM RESTART
    ###############################################
    echo "=== SECTION 15: FINAL SYSTEM RESTART ==="
    systemctl enable chromium-kiosk || true
    systemctl disable gdm3 || true
    systemctl enable sddm
    systemctl set-default graphical.target
    im-config -n none
    sudo tee /etc/environment >/dev/null <<'EOF'
GTK_IM_MODULE=none
QT_IM_MODULE=none
XMODIFIERS=@im=none
EOF

}

section_16() {
    echo "=== SECTION 16: remove notifs and enable auto updater  ==="
    # Placeholder for future sections
    sudo snap remove snapd-desktop-integration
    sudo snap remove snapd-desktop-integration-test
    sudo -u kiosk im-config -n none || true



}
```

```
section_17() {
    ################################################
    # SECTION 17: DISABLE ERROR POPUPS (Apport, notifications)
    ################################################
    echo "=== SECTION 17: Disable error popups for kiosk user ==="

    # Disable Apport (Ubuntu crash reporter)
    sed -i 's/^enabled=1/enabled=0/' /etc/default/apport || true
    systemctl disable apport.service || true
    systemctl mask apport.service || true
    systemctl stop apport.service || true

    # Disable update-notifier crash reports
    rm -f /var/crash/* || true

    # For kiosk user: suppress all desktop notifications
    sudo -u kiosk dbus-launch gsettings set org.gnome.desktop.notifications show-banners false
|| true
    sudo -u kiosk dbus-launch gsettings set org.gnome.desktop.notifications show-in-lock-
screen false || true

    # Divert stderr of systemd services to logs only (no journal popups)
    mkdir -p /etc/systemd/system.conf.d
    cat >/etc/systemd/system.conf.d/no-notify.conf <<'EOF'
[Manager]
DefaultStandardError=journal
DefaultStandardOutput=journal
EOF

    systemctl daemon-reexec || true
}


# Automatically detect the number of sections
get_max_section() {
    declare -F | grep -o 'section_[0-9]\+' | grep -o '[0-9]\+' | sort -n | tail -1
}

MAX_SECTIONS=$(get_max_section)
```

```
# Main script execution logic
show_usage() {
    echo "Usage: $0 [-s SECTION_NUMBER]"
    echo ""
    echo "Options:"
    echo "  -s SECTION_NUMBER  Run only the specified section (1-$MAX_SECTIONS)"
    echo "  -h, --help         Show this help message"
    echo ""
    echo "If no options are provided, all sections will be run in order."
    echo ""
    echo "Available sections:"
    echo "  1  - Power Management and Screen Blanking for current user"
    echo "  2  - Initial System Setup"
    echo "  3  - Desktop Environment Installation"
    echo "  4  - Phidget Setup and Dependencies"
    echo "  5  - Touchscreen Configuration"
    echo "  6  - Phidget Network Server Setup"
    echo "  7  - Kiosk User Setup"
    echo "  8  - GDM3 Autologin Configuration"
    echo "  9  - SDDM Configuration (Alternative)"
    echo "  10 - Chromium Kiosk Autostart Setup"
    echo "  11 - Wayland and Kiosk Verification"
    echo "  12 - Touch Gesture Configuration"
    echo "  13 - KWin Tweaks and Final Setup"
    echo "  14 - Power Management and Screen Blanking for kiosk user"
    echo "  15 - Final System Restart"
    echo "  16 - Remove notifications and enable auto updater"
    echo "  17 - Disable error popups for kiosk user"
    #echo "  18 - USB Auto-Update Service and eventual reboot"
}

run_all_sections() {
    echo "Running all sections in order..."
    for i in $(seq 1 $MAX_SECTIONS); do
        echo ""
        echo "========================================"
        echo "Starting Section $i"
        echo "========================================"
        section_$i
        echo "Section $i completed."
```

```
    done
    echo ""
    echo "All sections completed successfully!"
}

run_specific_section() {
    local section_num=$1
    if [[ $section_num -lt 1 || $section_num -gt $MAX_SECTIONS ]]; then
        echo "Error: Section number must be between 1 and $MAX_SECTIONS"
        exit 1
    fi

    echo "Running section $section_num only..."
    echo "========================================="
    echo "Starting Section $section_num"
    echo "========================================="
    section_$section_num
    echo "Section $section_num completed successfully!"
}
SECTION_NUM=""
# Parse command line arguments
while [[ $# -gt 0 ]]; do
    case $1 in
        -s|--section)
            if [[ -z "$2" ]] || [[ "$2" =~ ^-.*$ ]]; then
                echo "Error: Option -s requires a section number"
                show_usage
                exit 1
            fi
            SECTION_NUM="$2"
            shift 2
            ;;
        -h|--help)
            show_usage
            exit 0
            ;;
        -a|--all)
            run_all_sections
            reboot
            exit 0
```

```
            ;;

        *)
            echo "Error: Unknown option: $1"
            show_usage
            exit 1
            ;;
    esac
done

# Execute based on arguments
if [[ -n "$SECTION_NUM" ]]; then
    # Validate that section number is numeric
    if ! [[ "$SECTION_NUM" =~ ^[0-9]+$ ]]; then
        echo "Error: Section number must be a valid integer"
        exit 1
    fi
    run_specific_section "$SECTION_NUM"
else
    run_all_sections
fi
```

```
rm -rf /home/kiosk/.config/chromium/Default/Preferences
rm -rf /home/kiosk/.config/chromium/Singleton*


kwriteconfig6 --file startkderc --group General --key desktopSession empty
kwriteconfig6 --file ksmserverrc --group General --key loginMode emptySession
```

recharjme

restart chromium demon

```
#!/bin/bash
set -euo pipefail

# CONFIGURATION
APP_URL="http://localhost:8080"
INACTIVITY_MINUTES=10
CHROMIUM_BIN="/usr/bin/chromium-browser"
WATCHDOG_SCRIPT="/usr/bin/chromium-idle-restart.sh"
SERVICE_FILE="/etc/systemd/system/chromium-idle-restart.service"
USER_NAME="kiosk"

echo "=== Installing Wayland-Compatible Chromium Idle Watchdog ==="

# 1. Write the Chromium Watchdog Script
cat >"$WATCHDOG_SCRIPT" <<EOF
#!/bin/bash
set -euo pipefail

APP_URL="$APP_URL"
INACTIVITY_MINUTES=$INACTIVITY_MINUTES
CHROMIUM_BIN="$CHROMIUM_BIN"
USER_NAME="$USER_NAME"
PARAMS="--ozone-platform=wayland --enable-features=UseOzonePlatform,WaylandWindowDecorations \
 --no-first-run --no-default-browser-check --disable-session-crashed-bubble --disable-infobars
\
 --start-maximized --noerrdialogs --incognito --kiosk \$APP_URL"

while true; do
    idle_ns=\$(loginctl show-user "\$USER_NAME" -p IdleSinceHint | cut -d= -f2)
    now_ns=\$(date +%s%N)

    if [[ "\$idle_ns" =~ ^[0-9]+$ ]]; then
        idle_ms=\$(( (now_ns - idle_ns) / 1000000 ))
```

```
        if [ "\$idle_ms" -ge \$((INACTIVITY_MINUTES*60*1000)) ]; then
            echo "\$(date) - Wayland Idle Detected. Restarting Chromium..."
            pkill -9 -f "\$CHROMIUM_BIN" || true
            sleep 2
            eval "\$CHROMIUM_BIN \$PARAMS &"
            sleep \$((INACTIVITY_MINUTES*60))
        fi
    fi

    sleep 30
done
EOF

chmod +x "$WATCHDOG_SCRIPT"

# 2. Create the systemd service
cat >"$SERVICE_FILE" <<EOF
[Unit]
Description=Wayland Chromium Idle Restart Watchdog
After=graphical.target

[Service]
ExecStart=$WATCHDOG_SCRIPT
Restart=always
User=$USER_NAME

[Install]
WantedBy=graphical.target
EOF

# 3. Activate the service
systemctl daemon-reload
systemctl enable --now chromium-idle-restart.service

echo "=== Chromium Idle Watchdog Installed and Running ==="
```

recharjme

Installer Demon Template

```
#!/bin/bash
set -euo pipefail

# CONFIG
MEDIA_BASE="/media"
USB_LABEL="recharjme"
MONITOR_BIN="/usr/bin/NZK_System_Updater.sh"
SERVICE_FILE="/etc/systemd/system/NZK_System_Updater.service"

echo "=== Installing USB Update Monitor daemon ==="

# 1) Write daemon script
rm -f "$MONITOR_BIN"
cat >"$MONITOR_BIN" <<'EOF'
#!/bin/bash
set -euo pipefail

#!/bin/bash
set -euo pipefail

MEDIA_BASE="/media"
USB_LABEL="recharjme"
UPDATE_FLAG="/tmp/_updating"
TEMP_UPDATE_SCRIPT="/tmp/update.sh"
USB_MOUNT_PATH=""  # Global variable to track current USB path

log(){ printf '%s - %s\n' "$(date '+%F %T')" "$*"; }

find_recharjme_usb(){
  # Find any USB mount with label matching "recharjme" (case-insensitive)
  for user_dir in "$MEDIA_BASE"/*; do
    [ -d "$user_dir" ] || continue
    for mount_dir in "$user_dir"/*; do
      [ -d "$mount_dir" ] || continue
```

```
      mount_name=$(basename "$mount_dir")
      # Case-insensitive comparison
      if [[ "${mount_name,,}" == "${USB_LABEL,,}" ]] && mountpoint -q "$mount_dir"; then
        echo "$mount_dir"
        return 0
      fi
    done
  done
  return 1
}

check_usb_connected(){
  USB_MOUNT_PATH=$(find_recharjme_usb)
  if [ -n "$USB_MOUNT_PATH" ]; then
    return 0
  else
    USB_MOUNT_PATH=""
    return 1
  fi
}

restart_display_managers(){
  log "Restarting display managers..."
  systemctl restart gdm3 2>/dev/null || true
  systemctl restart sddm 2>/dev/null || true
}

process_update(){
  local update_file="$1"
  log "Found update: $update_file"
  : > "$UPDATE_FLAG"
  log "Created update flag: $UPDATE_FLAG"
  cp -f "$update_file" "$TEMP_UPDATE_SCRIPT"
  chmod +x "$TEMP_UPDATE_SCRIPT"
  log "Executing: $TEMP_UPDATE_SCRIPT"
  bash "$TEMP_UPDATE_SCRIPT"
  log "Update finished; cleaning"
  rm -f "$TEMP_UPDATE_SCRIPT"
  rm -f "$UPDATE_FLAG"
  restart_display_managers
```

```
}

log "=== USB Update Monitor started ==="
shopt -s nullglob

while true; do
  if check_usb_connected; then
    log "Found RECHARJME USB at: $USB_MOUNT_PATH"

    # Check if update is already in progress
    if [ -f "$UPDATE_FLAG" ]; then
      log "Update already in progress, waiting..."
      sleep 3
      continue
    fi

    # Look for update scripts
    candidates=( "$USB_MOUNT_PATH"/update*.sh )
    if [ ${#candidates[@]} -gt 0 ]; then
      process_update "${candidates[0]}"
      # Wait for USB disconnect with proper state checking
      log "Waiting for USB disconnect..."
      while check_usb_connected; do
        sleep 2
      done
      log "USB disconnected; rebooting in 3 seconds..."
      sleep 3
      systemctl reboot
    else
      sleep 3
    fi
  else
    # No USB connected, remove updating flag if it exists
    if [ -f "$UPDATE_FLAG" ]; then
      log "No USB connected, removing stale update flag"
      rm -f "$UPDATE_FLAG"
    fi
    sleep 3
  fi
done
```

```
EOF
chmod +x "$MONITOR_BIN"

# 2) Write systemd unit
rm -f "$SERVICE_FILE"
cat >"$SERVICE_FILE" <<EOF
[Unit]
Description=USB Update Monitor
After=local-fs.target
Wants=local-fs.target

[Service]
Type=simple
ExecStart=$MONITOR_BIN
Restart=always
RestartSec=2

[Install]
WantedBy=multi-user.target
EOF

# 3) Activate
systemctl daemon-reload
systemctl enable --now usb-update-monitor.service
systemctl restart usb-update-monitor.service || true
echo "=== Installed and running: usb-update-monitor.service ==="
systemctl --no-pager status usb-update-monitor.service || true
```