

# 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, sddm 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 usually 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
            ;;
    esac
done

```

```
        *)
            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
```

---

Revision #49

Created 2025-08-27 19:46:03 UTC by naruzkurai

Updated 2025-08-29 18:24:45 UTC by naruzkurai