FOXBUNTU SDK

For Femtofox Board on Luckfox Pico Mini A Hardware

TABLE OF CONTENTS

---

0. TLDR;

Foxbuntu is an **Ubuntu 22.04-based** OS for the **Femtofox LoRa board** on **Luckfox Pico Mini A hardware**, leveraging the **Luckfox-SDK**. This comprehensive manual consolidates every detail about:

- **Prerequisites** (Ubuntu 22.04, disk space, root privileges, internet connection)
- **Script Structure** (placing `foxbuntu-builder.sh` in `~/` and how `luckfox-pico` + `femtofox` repos are laid out)
- **Interactive Menu Usage** (typical workflow of **SDK Install**, then **Modify Kernel** or **Modify Chroot** for everyday changes)
- **Advanced Commands** (e.g., `full_rebuild`, `rebuild_chroot`, inject chroot scripts)
- **Flashing** (writing the final `foxbuntu.img` to an SD card)
- **Detailed Functions** (how each internal function manages U-Boot, kernel config, rootfs, firmware, synchronization, etc.)

Tips and Summary

1. **Initial Setup**
    - Use **SDK Install** once to establish everything. This may take a long time, so plan accordingly.
    - The script will create or overwrite `~/femtofox` and `~/luckfox-pico`.
    - Enter 1,0,1 at the board selection screen choices if using Femtofox hardware.
2. **Post-Install Modifications**
    - **Modify Kernel** if you require special modules or debug flags.
    - **Enter and Modify Chroot** for package installs, custom configurations, or service setups.
    - These operations automatically regenerate the final `.img`.
3. **Maintenance**
    - **Get Image Updates** periodically if you need the latest Foxbuntu changes from upstream. Ensure your local modifications in `~/femtofox/foxbuntu` are saved or committed first.
    - If the root filesystem becomes problematic or cluttered, **Rebuild Chroot** gives you a fresh start.
4. **Advanced**
    - **Full Image Rebuild** (rarely needed) if you suspect mismatched components.
    - **Inject Chroot Script** for scripted batch updates without interactive chroot sessions.
    - **Manual Build** items for partial or specialized scenarios (e.g., only re-building U-Boot).
5. **Deployment**
    - After each change, flash `foxbuntu.img` to your micro SD card with `dd` (or another tool), insert it into the Femtofox board, and enjoy your updated system.

By following these steps, you can hopefully maintain a stable, customized Foxbuntu operating system on your **Femtofox** hardware.

---

1. INTRODUCTION

Foxbuntu is an operating system derived from the **Luckfox-SDK**, itself based on **Ubuntu 22.04** (Jammy Jellyfish) for `armhf`. The scripts generate a **raw `.img`** file (commonly **3–4 GB**), which is flashed onto a micro SD card for deployment on the **Femtofox LoRa board** (a custom board built around the **Luckfox Pico Mini A** hardware).

These scripts do the following:

- Clone and integrate the **Luckfox Pico SDK** (`luckfox-pico`) and **Femtofox** repositories (`femtofox`), merging in **Foxbuntu** customizations.
- Build the **kernel**, **U-Boot**, **root filesystem**, and **firmware** modules.
- Provide **menu-driven** options to **install**, **modify**, **rebuild**, or **upgrade** your Foxbuntu environment.

Primary Goals:

1. **Ease of Use**: A menu-based system for typical users.
2. **Advanced Options**: Expert users can call script functions manually for partial or custom rebuilds.
3. **One-Time SDK Install**: This sets up everything from scratch and builds an image as a proof of life of your development environment.
4. **Incremental Changes**: Modify kernel or chroot as needed and rebuild.

---

2. REQUIREMENTS

1. **Host OS**: Ubuntu **22.04**. The script checks `/etc/os-release` and warns if you're on something else but will allow you to continue (/glhf).
2. **Disk Space**: At least **20 GB** free (the final `.img` is ~3–4 GB, but intermediate steps need extra).
3. **Run as Root**: Must execute the script with `sudo`. For instance:
   sudo ~/foxbuntu-builder.sh
4. **Script in Home Directory**: The build script, named `foxbuntu-builder.sh`, **must** reside in your home directory (`/home/username/`).
5. **Internet Connection**: Required for installing packages and cloning Git repositories.
6. **Tool Installation**: The script installs most dependencies automatically (`git`, compilers, `dialog`, `qemu-user-static`, etc.), but your package manager must be functional (i.e., `apt` should work with no errors).

---

3. STRUCTURE AND SCRIPT PLACEMENT

Upon using the script, you will end up with:

- **`~/foxbuntu-builder.sh`**: The main image build script.

- **`~/luckfox-pico`**: The Luckfox SDK environment for the Pico Mini A. It holds toolchains, build scripts (`build.sh`), kernel sources, etc.
- **`~/femtofox`**: The Femtofox repository. Within it is `foxbuntu/`, which merges with `luckfox-pico` to form your final environment.

> **Important**: During **SDK Install**, if `~/femtofox` or `~/luckfox-pico` exist, they **will be removed** to ensure a clean setup. Make backups if you have custom changes.

Script Usage:

- If you run `sudo ./foxbuntu-builder.sh` with **no** arguments, you get the main menu (described below).
- If you pass an argument (e.g., `sdk_install`, `modify_chroot`), it directly calls that function.

---

4. MENU OVERVIEW

When you run `sudo ./foxbuntu-builder.sh` without any arguments, you'll see a text-based menu offering a series of numbered choices. Below is a detailed explanation of each menu item. Keep in mind that for **typical** users, the most common tasks after an initial SDK install are **Modify Kernel** and **Modify Chroot**.

1. **Full Image Rebuild**
   - Rebuilds nearly everything: the environment, kernel, U-Boot, root filesystem, firmware, and finally produces a fresh `foxbuntu.img`.
   - **Important**: It does **not** wipe or fully recreate the chroot from scratch. It reuses the existing environment.
   - **Not commonly used** by most users unless you suspect multiple components are out of sync or you've made broad changes to different areas.
2. **Get Image Updates**
   - Pulls the latest commits from your `femtofox/foxbuntu` repository and synchronizes them into `luckfox-pico`.
   - After merging changes, it rebuilds the kernel, root filesystem, firmware, and creates a new `.img`.
   - **Warning**: If you have local modifications to `~/femtofox/foxbuntu` that are not committed, this can overwrite them. Always back up or commit changes before updating.

3. **Modify Kernel Menu**
    - Opens the kernel's configuration interface (similar to `make menuconfig`), allowing you to enable or disable kernel features and modules.
    - After you exit and save your changes (`.config`), the script rebuilds the kernel, updates the root filesystem, rebuilds firmware, and generates a new `.img`.
    - **Typical** if you need to enable a specific driver or debug option.
4. **Enter and Modify Chroot**
    - Binds `/proc`, `/sys`, `/dev`, and `/dev/pts` into your ARM root filesystem and then drops you into a **chroot** environment.
    - Once in the chroot, you can run commands (e.g., `apt update && apt install <package>`) or edit config files (`/etc/...`).
    - When you type `exit`, the script automatically unmounts these directories, cleans up, **then** rebuilds the root filesystem, firmware, and the final image.
    - **Highly used** for standard package installs or user-level changes.
5. **Rebuild Chroot**
    - **Wipes** the existing root filesystem environment and recreates it from scratch, reapplying Foxbuntu modifications afterward.
    - Great for a **clean slate** if the chroot becomes corrupt or if you want to ensure no leftover packages are present.
    - This also can be used to make modifications to femtofox.chroot script in the `~/femtofox/foxbuntu/environment-setup` directory (**caution** will be overwritten by using the **Get Image Updates** option)
    - Ends by building a fresh `.img`.
6. **Inject Chroot Script (CAUTION)**
    - Copies a user-defined script (or defaults to `~/femtofox/environment-setup/femtofox.chroot`) into the chroot and executes it automatically.
    - Ideal for bulk or automated updates (e.g., installing multiple packages without manually typing each command).
    - After execution, the script unmounts the chroot, cleans up, and rebuilds the final `.img`.
7. **Manual Build Environment**
    - Calls the Luckfox SDK's environment setup routine, typically used to confirm or adjust board storage and OS base.

    - Usually invoked automatically elsewhere, so it's rarely used alone unless you need to reset or debug environment variables.

8. **Manual Build U-Boot**

    - Builds only the U-Boot bootloader.
    - Typically for advanced debugging or testing a custom bootloader change.

9. **Manual Build RootFS**

    - Builds only the root filesystem.
    - This bypasses kernel, U-Boot, and other components.
    - Useful if you only changed some rootfs-related aspects and want a quick rebuild.

10. **Manual Build Firmware**

    - Builds only firmware binaries/modules and installs them into the root filesystem.
    - Typically done automatically in normal use, so it's mainly for advanced debugging.

11. **Manual Create Final Image**

    - Takes existing built components and packages them into a `.img`.
    - **Not** the standard approach for typical users—who generally let the script handle image creation after other steps.
    - Handy for advanced users who did partial builds and only need to finalize the `.img`.

12. **SDK Install (Run this first.)**

    - **One-time** setup for a new machine. Installs prerequisites, clones the repositories, configures the environment, and produces your initial `foxbuntu.img`.
    - **Destructive** if `~/femtofox` or `~/luckfox-pico` exist—these directories are removed to ensure a clean start.
    - After finishing, you can do incremental changes with other menu items.

13. **Exit**

    - Leaves the menu interface.

---

5. TYPICAL WORKFLOW

The following sequence highlights **common** user flows. While the script offers many advanced options, these steps suffice for most users who want to install and maintain **Foxbuntu** on the **Femtofox LoRa board** (Luckfox Pico Mini A hardware).

A. Initial SDK Install

Objective: Set up Foxbuntu **from scratch** on a clean machine or environment.

1. **Run the Script**:
   cd ~
2. wget https://raw.githubusercontent.com/femtofox/femtofox/refs/heads/main/environment-setup/foxbuntu-builder.sh -O ./foxbuntu-builder.sh
   chmod +x ./foxbuntu-builder.sh
   sudo ./foxbuntu-builder.sh

3. **Choose "SDK Install (Run this first.)"**:
   - This removes any existing `~/femtofox` and `~/luckfox-pico` directories.
   - Installs all prerequisites (compilers, QEMU, etc.).
   - Clones the necessary Git repositories.
   - Prompts you (in the Luckfox SDK build) to select the board (Luckfox Pico Mini A), storage (SDCard), and OS base (Ubuntu).
   - Builds U-Boot, the kernel, the root filesystem, the firmware, and **creates an initial** `foxbuntu.img`.

4. **Completion**:
   - Once finished, you have a bootable image in `~/luckfox-pico` (often `~/luckfox-pico/foxbuntu.img` or `~/luckfox-pico/output/image/foxbuntu.img`).
   - Flash it (see Flashing the Image) to get a baseline Foxbuntu system on your target board.

Note: The SDK Install step is **not** repeated unless you want a total reset. Subsequent changes are done via kernel or chroot modifications.

B. Modifying the Kernel

Objective: Customize the kernel with specific drivers, features, or debugging options.

1. **Select "Modify Kernel Menu"** from the script's main menu.
2. The script loads the kernel configuration (similar to `make menuconfig`), prompting you to enable/disable features.
3. **Save** the configuration as `.config` before exiting the menu.
4. **Automatic Rebuild**:
    - The kernel is rebuilt with your changes.
    - The script then rebuilds the root filesystem, firmware, and regenerates `foxbuntu.img`.
5. **Result**:
    - A new image with your updated kernel.

C. Modifying the Chroot

Objective: Adjust userland aspects, install packages, or change system configurations inside the target OS.

1. **Select "Enter and Modify Chroot"** in the main menu.
2. The script mounts `/dev`, `/proc`, `/sys`, `/dev/pts` into the root filesystem, then chroots you into it.
3. **Inside the Chroot**:

Perform typical Linux operations:
 apt update
apt install <some_package>

-
- Edit config files (`/etc/fstab`, `/etc/hosts`, etc.).
- Create or remove users, modify services, etc.

4. **Exit**:
    - Type `exit` to leave the chroot.
    - The script unmounts everything, cleans up, and **automatically** rebuilds the root filesystem, firmware, and final `.img`.

This step is **common** after the initial install whenever you need to add or remove packages.

D. Get Image Updates Warning

Objective: Sync the latest Foxbuntu changes from the Femtofox repository.

1. **Select "Get Image Updates"** from the menu.
2. The script performs a `git pull` on `~/femtofox/foxbuntu`, then merges these changes into `~/luckfox-pico`.
3. **Potential Overwrites**:
   - If you edited files in `~/femtofox/foxbuntu` without committing or saving them, **they can be lost**.
   - Always commit or back up your local changes beforehand.
4. **Automatic Rebuild**:
   - Recompiles the kernel, root filesystem, firmware, and updates the `.img`.
5. **Usage**:
   - Typically used to stay current with upstream fixes or features but must be done cautiously.

E. Full Rebuild vs. Rebuild Chroot

- **Full Image Rebuild**:

  - Recompiles U-Boot, kernel, root filesystem, firmware, and creates the `.img`.
  - It **does not** wipe the chroot entirely; it uses the existing environment but rebuilds each component.
  - **Less commonly used** if you only need a kernel tweak or a chroot package change.

- **Rebuild Chroot**:

  - **Wipes** the existing root filesystem (chroot) completely and rebuilds it from scratch.
  - Useful if you want a **clean start** or suspect your chroot environment is broken/tainted.
  - This will remove packages or custom changes not captured in your Foxbuntu merges.

---

6. FLASHING THE IMAGE

Once you've built or updated your `foxbuntu.img`, you can **deploy** Foxbuntu by writing this `.img` to a micro SD card. The typical steps are:

1. **Use dd, Balena Etcher, Pi Imager or other raw disk imager tool.**
2. **If you encounter an error that the image has no partition table, it's ok. This is normal.**
3. **Write the `foxbuntu.img` to a micro SD card that is at least 8gb up to 128gb.**
4. **Insert and Boot**
    - Eject the SD card, insert it into the Femtofox LoRa board (Luckfox Pico Mini A).
    - Power on the board; Foxbuntu should boot automatically.

This is the **standard** deployment workflow after every new or updated `.img` build.

---

7. DETAILED FUNCTION EXPLANATIONS

In addition to the menu system, `foxbuntu-builder.sh` defines various functions that **orchestrate** the build process. Most users rely on the menu, but advanced users or automated scripts may call these functions directly with arguments (e.g., `sdk_install`, `modify_chroot`). Below is a thorough explanation of each function. Some are used internally; others are mapped to menu entries.

---

1. `install_prerequisites()`

- **What It Does**
  Installs all required packages (compilers, linkers, libraries, QEMU user-mode emulation, binfmt-support, `dialog`, etc.).
- **Why It Matters**
  Ensures the host system can compile and run cross-architecture tasks for ARM (`armhf`).
- **Typical Trigger**
  Automatically called during **`sdk_install()`** or when the script first detects missing dependencies.

---

2. `clone_repos()`

- **Purpose**
  Clones both **Luckfox Pico SDK** (`luckfox-pico`) and **Femtofox** (`femtofox`) repositories into your home directory.
- **Retries**
  Attempts cloning each repository up to three times if a network glitch or server error occurs.

- **Menu Involvement**
  Primarily invoked during **SDK Install**. If it fails, the SDK install process stops.

---

3. `build_env()`

- **Goal**
  Runs the Luckfox Pico SDK's environment configuration, letting you choose:
  - The board (`Luckfox Pico Mini A`)
  - The storage device (`SDCard`)
  - The base OS (`Ubuntu`)
- **Usage**
  Usually automated but can be called manually from the menu (item **7. Manual Build Environment**) to reconfigure if something changes.
- **Process**
  Invokes an interactive sub-menu from the Luckfox build system. After confirmation, it sets up environment variables.

---

4. `build_uboot()`

- **Objective**
  Compiles the **U-Boot** bootloader for the Pico hardware.
- **When Called**
  - During the full or SDK build routines.
  - Can be triggered manually (menu item **8**) for specialized testing.
- **Result**
  Produces `u-boot.bin` or related artifacts in the `luckfox-pico` output folders, eventually integrated into the final `.img`.

---

5. `build_rootfs()`

- **Function**
  Builds or updates the **root filesystem** for the ARM target. Often includes:
  - Creating an initial Ubuntu 22.04 userland (via debootstrap or a similar mechanism inside Luckfox).
  - Installing minimal packages.
- **Called By**
  Many processes (e.g., SDK install, modify chroot, kernel changes) rely on a fresh or updated rootfs.

- **Significance**
  The rootfs is the “user space” environment that runs on the Femtofox LoRa board.

---

6. `build_firmware()`

- **Purpose**
  Builds any additional firmware components—drivers, kernel modules, or binary blobs—that reside outside the main kernel build.
- **Triggered**
  Typically after `build_kernelconfig()` or `build_rootfs()`, ensuring newly compiled drivers are integrated.
- **Menu**
  Exposed as **10. Manual Build Firmware** for advanced usage, but generally invoked automatically.

---

7. `sync_foxbuntu_changes()`

- **Role**
  Synchronizes your **Foxbuntu** modifications (under `~/femtofox/foxbuntu`) with the **Luckfox Pico** SDK folder (`~/luckfox-pico`).
- **Mechanism**
  Uses `rsync` to copy or remove files so `luckfox-pico` aligns precisely with what’s in `femtofox/foxbuntu`.
- **Why It’s Important**
  Ensures the custom Foxbuntu patches, scripts, and configs overlay the default Luckfox environment.

---

8. `build_kernelconfig()`

- **Action**
  Calls the kernel’s **menuconfig** (or a similar interface) so you can configure features, modules, etc.
- **Post-Config**
  You must save `.config` to preserve your changes before exiting.
- **Menu Usage**
  Usually part of **Modify Kernel Menu**, but can be invoked in other build steps.

---

9. `modify_kernel()`

- **High-Level**
  This is a wrapper that:
  1. Invokes `build_kernelconfig()` for you to make changes.
  2. Builds the kernel with those changes.
  3. Updates the root filesystem, firmware, and regenerates the final `.img`.
- **Practical Benefit**
  Combines multiple steps into a single function, simplifying the kernel customization workflow.

---

10. `modify_chroot()`

- **Purpose**
  - Binds `/proc`, `/sys`, `/dev`, `/dev/pts` into your **root filesystem** directory.
  - Chroots you into that environment, allowing you to run commands as though you're on the target hardware.
  - After you exit, it automatically **rebuilds** the root filesystem, firmware, and the final `.img`.
- **When to Use**
  - Whenever you want to install or remove packages (e.g., `apt install something`) or edit configuration files (e.g., `/etc/network/interfaces`) **inside** the target OS.
- **Menu Item**
  - Corresponds to **4. Enter and Modify Chroot**.

---

11. `rebuild_chroot()`

- **Function**
  - **Wipes** the existing chroot (root filesystem) folder in `~/luckfox-pico/sysdrv/out/rootfs_uclibc_rv1106/` (or similar path).
  - Rebuilds it from the Luckfox base plus the Foxbuntu modifications.
  - Finishes by reinstalling and regenerating the `.img`.

- **Usage**
  - If the existing chroot environment is **corrupted** or you want a truly clean environment without leftover packages.
  - More destructive than just modifying the chroot.
- **Menu Entry**
  - **5. Rebuild Chroot**.

---

12. `inject_chroot()`

- **Mechanics**
  - Copies a script (`CHROOT_SCRIPT`, defaults to `~/femtofox/environment-setup/femtofox.chroot`) into the chroot's `/tmp/`.
  - Mounts system paths, **executes** that script inside the chroot, then unmounts and cleans up.
  - Rebuilds the root filesystem, firmware, and `.img`.
- **Benefit**
  - Ideal for **automating** changes that would otherwise require you to manually enter the chroot.
  - Example: a script that installs multiple packages, adds users, and configures services.
- **Menu Item**
  - **6. Inject Chroot Script (CAUTION)**.

---

13. `update_image()`

- **Purpose**
  - Performs a `git pull` on `~/femtofox/foxbuntu`, merges changes to `~/luckfox-pico`, rebuilds the kernel, rootfs, firmware, and the final `.img`.
- **Menu Usage**
  - **2. Get Image Updates**.
- **Caution**
  - Can overwrite local changes in `femtofox/foxbuntu` if they're not committed or saved elsewhere.

---

14. `full_rebuild()`

- **What It Does**
  - Re-invokes environment steps, builds U-Boot, syncs Foxbuntu changes, rebuilds kernel config, rootfs, firmware, and the final `.img`.
  - Does **not** remove the existing chroot—just recompiles most components.
- **When to Use**
  - Not commonly necessary unless you've made widespread changes or suspect some mismatched parts.
  - Corresponds to **1. Full Image Rebuild** in the menu.

---

15. `install_rootfs()`

- **Role**
  - Copies kernel modules into the root filesystem.
  - Uses `qemu-arm-static` for emulation inside the chroot, then runs any specified setup script.
  - Cleans up the chroot afterwards.
- **Primarily**
  - Called automatically during major build steps (e.g., `sdk_install()`). Rarely invoked by itself.

---

16. `create_image()`

- **Process**
  - Uses the Luckfox tools (like `mkenvimage`, `blkenvflash`) to package the kernel, U-Boot, and root filesystem into a single `.img` named **`foxbuntu.img`**.
  - Modifies `.env.txt` to enlarge the rootfs size from **6G** to **100G** if it sees the default "6G(rootfs)" line.
- **Manual Option**
  - Mapped to **11. Manual Create Final Image**, but typically called automatically by other steps (e.g., modify kernel, chroot, etc.).

---

17. `sdk_install()`

- **Primary**
  - The **one-time** function that does it all:
    1. **install_prerequisites()**
    2. **clone_repos()**
    3. **build_env()**
    4. **build_uboot()**
    5. **sync_foxbuntu_changes()**
    6. **build_kernelconfig()**, **build_rootfs()**, **build_firmware()**, and **create_image()**
  - Produces a ready-to-use **`foxbuntu.img`**.
- **Destructive**
  - If `~/femtofox` or `~/luckfox-pico` exist, it warns you before deleting them.
- **Menu Item**
  - **12. SDK Install (Run this first.)**.
- **Usage**
  - Ideal for a brand-new setup or complete environment reset.

---

18. `usage()`

- **Help Text**

Prints usage instructions for the script if you run:
sudo ./foxbuntu-builder.sh --help

  -
  - Lists the possible function arguments, including `sdk_install`, `modify_chroot`, etc.

---

8. NON-INTERACTIVE USAGE

While the script is primarily menu-driven, you can directly call any function for automation:

sudo ./foxbuntu-builder.sh sdk_install
sudo ./foxbuntu-builder.sh modify_chroot
sudo ./foxbuntu-builder.sh full_rebuild
sudo ./foxbuntu-builder.sh --chroot-script /home/user/my_script.sh inject_chroot

- **Automation**: This is helpful in CI/CD pipelines or if you prefer scripting each step without interactive prompts.
- **Caution**: Some functions rely on environment setup or previous steps, so ensure you call them in a logical order (for example, do not call `build_rootfs()` before `build_env()` if your environment isn't configured).