Electric Blocks II Handoff Document

Greyson Biggs

Samuel Frederickson

Ryan Buckel

If you're reading this, you're likely a University of Idaho Senior who chose Electric Blocks as your Capstone Design project for 2022-2023 (or later)! If not, that's okay too. This informal document gives a quick overview of the Electric Blocks codebases, in the hopes that it'll allow you to get programming quickly, without falling into some trappings. It is **not** extensive nor documentation of every file across all the codebases.

To push changes to the repos and/or do anything within the Electric Blocks GitHub organization, you'll need to contact Dr. Conte de Leon (dcontedeleon@uidaho.edu), as he holds the keys to the keep, so to speak. Until then, you may fork any repo and submit pull requests for merging.

Words of Wisdom and Java Noob Protips

1.15.2: At the time of writing, Electric Blocks builds on Minecraft 1.15.2. Unless it's trivially easy (frankly, we don't know how much the Forge mappings have changed, or if HWYLA/WAILA have updated to 1.18+ by now), we recommend not spending time updating the mod to later versions of Minecraft. Electric Blocks is likely not going to benefit from any other mods or Minecraft updates, so unless there's a specific problem that would most efficiently be solved by updating to 1.18+ (i.e., you specifically need EB to work with another mod that only exists on a different version of Minecraft), you probably shouldn't bother.

IntelliJ IDEA: The Electric Blocks mod repository is confirmed working using IntelliJ IDEA. We had problems getting Gradle to cooperate with Eclipse (meaning we couldn't get EB to build, and we tried a lot of things). IntelliJ is sleek, and it generally Just Works™.

Doks website: The Doks website repository is **not** able to be cloned to a Windows machine, as several filenames include colons (:). Therefore, you will probably want a Linux machine to modify this repo, or just edit the markdown files in GitHub itself.

The Mod Repository

Jargon:

There are several terms that are helpful to know before jumping into this, or any Minecraft mod, for the first time. Here's a few:

Items: The entities that show up in your inventory. They may represent blocks or other entities.

Blocks: The basic entities that are placed around the world of Minecraft, which occupy 1 m^3 of space with respect to other blocks.

Tile Entities: Extra data associated with blocks.

Simulation Tile Entities: Tile entity data specifically associated with Electric Blocks.

HWYLA: The mod “Here’s What You’re Looking At” which displays information about blocks and entities in a tooltip when you hover over them. HWYLA is used to display information about some EB blocks (via plugin classes in the EB mod), for ease-of-use.

Gradle: The open-source Java build automation tool.

EBPP: Electric Blocks PandaPower – the simulation server that the EB mod talks to.

Repo overview:

The following are simple explanations of important directories and files in the electricblocks repo that you may be modifying. Intended for those who haven’t had much experience with a large Java/Minecraft project before:

.gitignore: Defines files in your local electricblocks repo folder that will not be pushed to the main repository by git. (e.g. local build folders like /run)

.build.gradle: Defines build behavior. Mostly important if you’re adding other mods as dependencies.

gradle.properties: Version info goes here.

update.json: Link to the website, and more version info, for the sake of update notifications.

/assets: Contains EB logo images.

/lib: External library .jar files go here. At time of writing, this just contains the HWYLA sources .jar (allowing plug-in functionality for HWYLA).

/run: When you build and run the mod from within your IDE, this folder will be created and houses all the usual .minecraft folder data. It is separate and isolated from your main .minecraft installation folder.

/src/main: Where the magic happens. The following entries are all contained in this folder.

/java/edu.uidaho.electricblocks: All .java source files go here.

- **/blocks**: Classes for each new EB block, defining certain block attributes (hardness, sound when placed, etc.) and functions.
- **/containers**: Defines attributes about blocks with their own inventories (i.e. the Electric Furnace).
- **/eventhandlers**: Defines behavior for left/right-clicks (with the multimeter) and block-breaking with respect to EB blocks.
- **/guis**: Contains the mod configuration menu class (currently optionless), and the Simulation Tile Entity Screen class (the screen that pops up when you right-click an EB block with a multimeter).
- **/init/RecipeSerializerInit.java**: Defines stuff for the gathering of recipes during initialization.
- **/interfaces**: Defines the functions that are called when a player left or right-clicks with a Multimeter.
- **/items**: Constructor classes for the EB items (specifically, the entities exist in your inventory – blocks are entities that exist in the 3D world).

- **/lib**: External library classes. Currently only contains the Feature class to enable the HWYLA plugin classes.
- **/network**: Defines packets that are sent to EBPP.
- **/plugins**: HWYLA plugin feature classes.
- **/recipes**: More recipe behavior stuff. Folks probably aren't using EB in anything but creative mode, so you're unlikely to mess with this.
- **/simulation**: Client-side processing of PandaPower simulation information. The stuff in here pieces together what circuits are made in Minecraft and reformats it in a form that PandaPower can process, and vice versa to an extent.
- **/tileentities**: Defines attributes and functions related to power simulation (e.g. inputs/outputs, JSON storage for each block's simulation properties).
- **/utils**: Miscellaneous classes.
- **EBHWYLAPlugin**: Declares the EBHWYLA plugin to HWYLA and defines how each feature is registered in ElectricBlocksMod and /plugins.
- **ElectricBlocksConfig**: Helps set up the connection to EBPP.
- **ElectricBlocksMod**: The main mod declaration class, which also defines some behavior during initialization.
- **RegistryHandler**: Un-instantiable class that registers all blocks, items, and tile entities.

/resources: A lot of .json files, textures, and miscellaneous text files go here.

- **assets.electricblocks/blockstates**: Defines how blocks orient themselves when placed.
- **assets.electricblocks/lang/en_us.json**: Contains key-value string pairs (typically used for localization, but also good for making consistent string changes quickly). You will likely use this.
- **assets.electricblocks/models/block**: Defines what textures display on which faces of a block.
- **assets.electricblocks/models/item**: Defines what items are connected to which blocks (the wire item is connected to the wire block) or entities.
- **assets.electricblocks/textures**: Sprites, textures, and GUI image files go here.
- **META-INF/mods.toml**: Mod metadata displayed by the Forge loader in the Mods menu. Version, credits, and authors, mostly.
- **logo.png**: The EB logo! Again!

The Doks Repository

As stated above, you will probably want to clone the Doks repository to a Linux machine – the existing repo contains files with names incompatible with Windows, meaning you can't clone it there.
If/when you decide to modify the Doks website, you should begin by giving the Doks tutorial a whirl: https://getdoks.org/tutorial/introduction/

Your interaction with Doks should primarily involve markdown and image files, plus a bit of command line to add new pages. Overall, Doks is reasonably simple and you shouldn't run into too much trouble beyond the Windows incompatibility.

The EBPP Repository

EBPP is the power flow simulation server software that takes JSON input from the Electric Blocks mod and translates it to something that PandaPower can modify, and vice-versa. It is required for the EB mod to do anything.

Follow the instructions located at https://electricblocks.github.io/docs/ebpp/installation/ to get your EBPP image up and running.

The important files associated with EBPP are as listed:

- **ebpp.py:** Contains error handling, the API entry point, and the responses for Electric blocks. This controls the creation of all electric block elements created in Panda Power and returns them to the client.
- **errors.py:** Contains all the errors associated with Electric Blocks.
- **utils.py:** Has all required arguments for each block type created in EBPP. Think of this as the essential variables which are required to create any of the Electric Block items.

For a more in-depth explanation of EBPP. Refer to the README.md file inside of the EBPP repository.

Recommendations

Electric Blocks still has a way to go. Here are some suggestions for future teams to work on:

- 3-phase power support. The 2020-21 team has a 3-phase branch of EB, but we haven't tested it and there's no indication that it works right now. You may be able to build on whatever they got done, though.
- Updates to some clanky blocks. (e.g., the Electric Furnace could smelt over time like the vanilla Minecraft furnace does, wire block could be a wire model much like how fences or glass panes function.)
- More time-based elements and realistic in-world reactions to power systems. (e.g., Batteries could run out over time, Lamps' brightness and Electric Furnace's smelting time could scale with power, powered rails could exist, unrealistically high values could cause circuit failures.)
- Expand the orientation map with a full-scale tutorial course and more complex circuits.
- In-game tooltips/commands explaining blocks.
- Further integration and features supported with PandaPower.
- If absolutely necessary, an upgrade from Minecraft 1.15.2 to the latest version may be needed. This is a huge undertaking, however, and we don't recommend it unless there are some very specific and important reasons to do it.

We also recommend that an EE major be present on the 2022-23 team, or at least that one weigh-in and validate functionality from time to time.