Eigene Firmware

Introduction

Our "own firmware" (called spielzeug) is not really a custom firmware. This is just a custom runtime built on top of the original LEGO SPIKE v2 firmware, which needs to be installed to use it and will stay installed.

Installation

Installing the CLI

To manage the hub, you need to install the CLI tool "tigerente". To do this, use your python package manager of choice and install "tigerente".

PIP

```
pip install tigerente
```

PIPX

```
pipx install tigerente
```

UV/UVX

```
uv tool add tigerente
```

Installing the firmware

To install the current version of the firmware, ensure the newest LEGO firmware is installed and plug the hub into your computer using USB.

Then, run the following command:

```
tigerente fw-flash
```

This will flash spielzeug on your device and connect to it.

Updating your hub

You can always update the firmware using the command above, but for convenience, you can use the following one in order to update the firmware over bluetooth:

```
tigerente fw-update
```

Renaming your hub

When you install the firmware for the first time, a random name will be assigned to the hub. To assign your own name, run:

```
tigerente rename <NEW_NAME>
```

Uploading a program

First create a directory where your program will live in. In this tutorial we will call it `src`. Inside this directory, create a file called `__init__.py`.
The `__init__.py` needs to define an async function called `loop`. This is the main entrypoint of your program.

```
# src/__init__.py

async def loop():
    ...
```

For this tutorial, we will write a program that will make the power LED light up in green for five seconds and in red for another five. That would look something like this:

```
# src/__init__.py

import asyncio
import hub
import color



async def loop():
    hub.light.color(hub.light.POWER, color.GREEN)
    await asyncio.sleep(5)
    hub.light.color(hub.light.POWER, color.RED)
    await asyncio.sleep(5)
```

Great!
Now, we can upload this program by running the following command:

```
tigerente sync src
```

You should now see the program running on the hub. If it was too fast and you want to start the program again, run:

```
tigerente start
```

This command will start the program that was uploaded most recently.

You have successfully written your first program using spielzeug, great!

Uninstalling

To uninstall the firmware and go back to the official one from lego, use the following command:

```
tigerente fw-update -v restore-original
```

In case this does not work, you can always use the `fw-flash` command in the same way, just like with updates as explained above.

Host2Hub File Sync Protocol

Packet Format

Each packet starts with a one byte long identifier followed by the arguments. The end of the packet is signalized by a **SUB** (0x1A) byte. (This is subject to change and will likely be replaced by a **NUL** (0x00) byte in the future. The different packets are listed below.

Generally, the host sends a packet to the hub and the hub responds. The only cases in which the hub actively sends a packet are if it reports an error or delivers a print message.

Packet Types

The Host can send these Packets, the Hub must handle these packets:

Identifier	Type	Meaning	Argmuent
Y (0x59)	Actively Sent	Start *SYNC Mode*	
D (0x44)	Actively Sent	Ensure Directory exists	/path/to/dir
F (0x46)	Actively Sent	Ensure File has same Hash	/path/to/file SHA256
C (0x43)	Response	CHUNK	192 Bytes of the file data (base64-encoded compressed file)
E (0x45)	Response	End Of File	
N (0x4E)	Actively Sent	End *SYNC Mode*	
R (0x52)	Actively Sent	Remove File or Directory	/path/to/file/or/diretory
P (0x50)	Actively Sent	Start Program (loop() from /__init__.py)	
X (0x58)	Actively Sent	Stop Program	
& (0x26)	Actively Sent	Reboot	
= (0x3D)	Actively Sent	Sync Communication	timestamp
> (0x3E)	Actively Sent	Enter REPL	

$ (0x24)	Actively Sent	Reset State	

The Hub can send these Packets, the Host must handle these packets:

Identifier	Type	Meaning	Argmuent
K (0x4B)	Response	OK	
= (0x3D)	Response	Sync Communication	given timestamp
U (0x55)	Response	Request File Contents	
! (0x21)	Actively Sent	*Internal Error*	traceback
E (0x45)	Actively Sent	*User Error*	traceback
P (0x50)	Actively Sent	*Print*	json-encoded args & kwargs

Packets written in italics are not yet documented below.

Examples

Sync all files

If a full tree needs to be synced to the hub, this is the right choice.

1. Host sends `Y` to start *SYNC Mode.*
   Hub responds with `K`.
2. Here, Files and Directories are updated.

   Refer to Update an individual file and Update an individual directory.
3. Host sends `N` to stop *SYNC Mode*.
   Hub responds with `K`.

Effect

Upon receiving `N`, the Hub deletes all files and directories that were not updated since `Y` was sent.

Update an individual file

1. Host sends `F`. The argument is the file's path and its SHA256 Hash, seperated by a space.
   The file path needs to start with `/`.
   Hub responds with `K`, ***or*** with `U`.

2. If `U` was sent, Host repeatedly sends `C` with 192 bytes of the file as argmuent, ***or*** `E` if EOF is reached.
   Hub responds to each of these packets with `K`. Hub can respond with `U` to `E` if the hash is still not matching.

Effect

The Hub checks whether the file exists and has the correct hash. If it is correct, it responds with **K**. Otherwise, it sends **U** and stores the file received in the following **C** packets under the given file path.

Update an individual directory

1. Host sends `D` with the file path as argument. The file path needs to start with `/`.
   Hub responds with `K`.

Effect

The Hub ensures the directory exists.

Delete an individual file or directory

1. Host sends `R` with the file path as argument. The file path needs to start with `/`.
   Hub responds with `K`.

Effect

The Hub deletes the file or directory found under the given file path if it exists. Non-empty directories are cleared.

Start the program

1. Host sends `P`.
   Hub responds with `K`.

Effect

The Hub stops the task it created when it received `P` the last time, if it exists.
The Hub starts the `loop()` function it found in the `/__init__.py` file by creating a new task using `asyncio`.

Stop the program

1. Host sends `X`.
   Hub responds with `K`.

Effect

The Hub stops the task it created when it received `P` the last time, if it exists.

Sync connection

1. Host sends `=` with arbitrary data as argument.
   Hub echos the packet.

Effect

Echo. Ping-Pong. Verify connection.

Open REPL

1. Host sends `>`.

Effect

The Hub stops all running tasks and returns control to the Micropython REPL.

[OUTDATED: This is to be preferred in comparison to sending Ctrl-C (which also works), as it makes sure the `print()` function works as expected afterwards.] -> [NOTE: Ctrl-C should not work]

Reboot Hub

1. Host sends `&`.

Effect

The Hub reboots.

Cancel sync

1. Host sends `$`.
   Hub responds with `K`.

Effect

The Hub resets its sync state. It is recommended to send this when a new connection is made.

tigerente