OpenFocuser by Marco Cipriani

OpenFocuser is a MoonLite-compatible motorized focuser controller with absolute and relative positioning and half and full step switching. It is Unix/Linux (INDI) and Windows (MoonLite and ASCOM) compatible. Two editions are available: standard and Plus, which has a non-MoonLite extra function to control all the Arduino's digital and PWM pins, allowing the final user to turn on and off (or regulate in current) other devices like dew heaters, mirror coolers, or even a Raspberry Pi, directly from your computer, or remotely with Telescope-Pi. OpenFocuser also comes with a firmware update utility (via avrdude), a command line server for remote control (using INDI) and a control panel to manage the pins (name and value at startup). 3D mounting brackets and circuit/PCB projects are also provided.

Usage

Focuser

OpenFocuser is fully compatible with MoonLite softwares, so please refer to the official user guides for MoonLite focusers, or to the documentations for INDI (indi_moonlite_focus), KStars, ASCOM or whatever software you use for astrophotography. Recommended stad-alone software for Windows: MoonLite Single Focuser V1.4

OpenFocuser-Manager

OpenFocuser-Manager is a Java 8 desktop application that allows the end user to update the Arduino firmware easily (eliminates the need of installing the whole Arduino IDE, compiling and uploading the sketch) and to control the digital pins of the board if using the Plus edition. It comes bundled with `avrdude` and the latest `.hex` firmwares for Arduino Nano, serial libraries, INDI4Java server and self-updating utility.

Installation

- Debian and Ubuntu:
  - Install the Debian package: `dpkg` will automatically install `socat` and `avrdude` for you, a launcher icon will be created and you'll be ready to use it.
  - Packages `openssh-client` and `openssh-server` are needed if you want to upload and download the configuration files, see below for information.
- Other distros:
  - Download the `jar` archive and launch it. `socat` and `avrdude` **must** be installed depending on you package manager and in path.
  - `ssh` and its server are needed if you want to upload and download the configuration files, see below for information.
- Windows:
  - Download the `jar` archive and launch it. `avrdude.exe` and `libusb0.dll` are bundled in it, eliminating the need of installing them. The INDI server and driver and the settings sending won't be available, only firmware update.
- MacOS: I don't have a Mac, so I can't provide a package for this OS. Feel free to help!

Updating the firmware

Firmware update via `avrdude` is supported in Windows (`avrdude.exe` is bundled in the executable `jar` and will be unzipped at runtime in the system-default temp folder) and in Linux, no matter the distribution, if the `avrdude` executable is in path (in Debian, install it with `sudo apt-get install avrdude`). `avrdude`'s config is selected automatically by the Manager.
In the firmware update tab you can select the serial port of the target board, the board type (at the moment, the firmware is built only for Arduino Nano, new and old bootloaders), and the firmware edition: standard (focuser only) or Plus (controllable pins and polar finder illuminator). A label below the firmware selection shows the version of the selected software. Press Update to flash the board.

Plus edition pin management

Note: standard MoonLite focusers do NOT support pin management! They aren't compatible with OpenFocuser
If you are new to the INDI protocol please read more in the INDI website and in Wikipedia.
In the "Plus edition configuration" tab you'll be able to select the port for the INDI server (default 7625 to allow other INDI clients like KStars to use 7624) and define the list of digital and PWM pins. You can click "Add" to add a pin definition: adding a digital pin means adding an INDI *switch element* (a checkbox in the INDI control panel of the client) that allows the user to switch the state of the pin ON and OFF; instead, after creating a PWM pin, OpenFocuser will add an INDI *number element* to its driver that allows the end user to write the pin value (0→100%) directly from the client INDI control panel. You'll be asked about the pin port (for example, pin 13). **Note: you can add only the pins you have selected in the Arduino configuration before!** Then you can click "Edit" to modify the pin's properties: a custom name (e.g. "Dew heater") and a default value, applied when the driver starts. After defining all the pins, save the configuration and start the server directly from the control panel or close it to use the driver or the server from the command line (see advanced usage). In order to use the pin manager driver you'll need an INDI client. In KStars, open Ekos from the toolbar and create a new profile containing your telescope mount, CCD camera or reflex and a MoonLite focuser, and in the "Remote:" driver field write `INDI Arduino pin driver@localhost:7625`. Be careful to replace `localhost:7625` with the right host and port. **Uncheck** the auto-connect box and give the profile a name. Now start the OpenFocuser server (from the control panel or from the command line). Start the Ekos' INDI server and open the INDI control panel. Connect your devices, go to the "INDI Arduino pin driver" tab and connect the driver. In the "Serial connection" tab select a serial port and hit connect. A new tab called "Manage pins" will show up, in which you can mange the PWM and digital pins you selected in the control panel. Copy the MoonLite port to the clipboard and paste it in the "Port" field in the MoonLite driver tab. Select baud speed to 9600 and connect the MoonLite device. If everything is OK you'll get the full MoonLite control panel. Otherwise check if the OpenFocuser server is running, if the virtual port exists and if the speed is 9600. Enjoy!

Advanced usage and troubleshooting

Sending configuration to another computer

You can send the pin configuration and all the settings to another computer. Ensure OpenFocuser-Manager is installed on both computer alongside with the required dependencies. The other computer must have a SSH server installed, while the sender a SSH client. From the sender computer, open the control panel and click on "Send configuration". You'll asked about the remote host, username and password. If the process

fails due a missing remote folder, open and close one time the control panel in the remote computer and retry (this will create the required config folder in the remote user directory, which must be present in order to send the settings file).

Stand-alone CLI server

Use `openfocuser -p=xxxx`, replacing `xxxx` with whatever port you want (or `0` to use the saved port), to start the server without GUI in the terminal. The configuration will be the same saved before in the control panel.

INDI driver in another INDI server

The INDI driver can be run inside another INDI server executing `openfocuser -d`. No GUI will be loaded, nor the server will be set up: the driver will communicate with your external server with stdin/out, just like any other INDI driver.

Starting from the command line

- `bash`: `openfocuser <options>`
- Windows: `java -jar OpenFocuser-Manager.jar <options>`

Short option	Long option	Param	Description
`-a`	`--serial-port`	e.g. `/dev/ttyUSB0`	Specifies a serial port and connects to it if possible. Otherwise it will be stored to settings only.
`-c`	`--control-panel`		Shows the control panel.
`-d`	`--driver`		Driver-only mode (no server, stdin/stdout)
`-p`	`--indi-port`	e.g. `7625`	Stand-alone server mode, CLI. If port=0, fetch the last used port from the settings.
`-v`	`--verbose`		Verbose logging mode.

Common errors

Exit code	Error	Solution

Exit code	Error	Solution
0	Java not found	Install Java >8 and ensure it's in path
3	`socat` not found	Install `socat` (Linux)
4	INDI not found	Install `indiserver` (Linux)
5	`avrdude` not found	Install `avrdude` (Linux)
6	`jar` not found	Where did you put `OpenFocuser-Manager.jar`? Did you build the project and artifacts?
8	Config folder could not be initialized	OpenFocuser-Manager wasn't able to create the folder where it'll put the configuration. Using your file manager, go to you home directory and check if the `.config/OpenFocuser-Manager` folders exist and are directories
9	Unable to parse parameters	Check the command line arguments
10	Invalid options	Invalid combination of command line arguments (for example, you can't run the driver and the control panel or the stand-alone server at the same time)
11	`socat` error	`socat` could not be started. Check if it's installed, up-to-date and in path. No solution for operating systems other that Linux, but shouldn't happen: report an issue if so.

Hardware

Autodesk Eagle circuit

In the "Eagle" directory you can find the full circuit project, both schematics and PCB for the standard and Plus editions. Feel free to modify it to accomplish your necessities: for example, you could add another dew heater controller, or remove the Newton mirror cooler MOSFET.
Made with Eagle 9.2.2 Premium

Motor holders

I included the mounting brackets I made for the common SkyWatcher Dual-Speed Crayford focuser (I have a 200mm f/5 Newton OTA).
AutoCAD 2019 project, STL and IGS exported files ready for 3D printing.

Focuser motor and drivers

I use a Nema 11 motor that moves the focuser knob using a belt. If you don't have a heavy focuser, also consider using a Nema 8. A 20 teeth pulley and a 6mm wide 160mm long belt are enough. Supported motor drivers via the StepperDriver library:

- Generic 2-pin drivers
- DRV8825
- A4988
- DRV8834

Developer's guide

Arduino configuration

Compiling from sources

The Arduino sketch manages the communication with the computer and sends the appropriate commands to the motor driver. To configure the pins and enable/disable features refer to the "Using the Config file" section below.
Required: the Arduino IDE for editing (or compiling manually) and a Linux machine.
`Builder/build-fw.sh` is a `bash` script that automates the process of configuring, compiling, renaming and updating the `hex` files in the Manager sources. To run it, execute `Builder/build-fw.sh`, no arguments required. The script will create and place in the Manager sources the firmwares for both the standard and Plus editions of OpenFocuser, compiled for Arduino Nano with the new and old bootloaders. Arduino CLI is used to compile the sketch. It's part of the Arduino project. The Arduino trademark and this executable belong to its owner and I'm not affiliated with it. License of Arduino CLI.

Libraries

- AccelStepper by Mike McCauley, license
- StepperDriver by Laurentiu Badea, license
- Button Debounce by maykon, license

Contributors

This firmware uses parts of the sketches by Orly Andico and Daniel Franzén:

- Inspired by arduino-focuser-moonlite by Orly Andico, blog post
- Modified for INDI, easydriver support (*removed*) by Cees Lensink
- Added sleep function by Daniel Franzén, [GitHub repo]((https://github.com/FranzenD/arduinofocus)

Using the Config file

In the firmware folder, the `Config.h` defines:

- the pins and type of motor driver
- the status LED pin
- serial speed
- hand control buttons and potentiometer
- whether or not the polar finder illuminator is enabled
- the customizable pins

Motor drivers

- `STEPPER_TYPE`
    0. BasicStepperDriver
    1. DRV8825
    2. A4988
    3. DRV8834
- `DRIVER_DIR`: stepper driver DIR pin
- `DRIVER_STEP`: stepper driver STEP pin
- `M0`, `M1`, `M2`: pins for setting the microstepping mode (not used by `BasicStepperDriver`)

Hand controller

- `ENABLE_HC` to enable it
- `HC_SPEED_POT` the potentiometer that controls the number of steps the focuser moves every time you press a button
- `BUTTON_UP` and `BUTTON_DOWN`: pins of the buttons

Pin management

To change the default list of customizable pins, turn on this function with `ENABLE_PIN_CONTROL` and then define the customizable pins in the `CUSTOMIZABLE_PINS` array. You'll be able to control them using OpenFocuser-Manager, see below for further information.

Polar finder illuminator

You can include a LED output to illuminate the polar finder of you mount enabling `ENABLE_PFI` and setting

- `PFI_POT`: the analog input of the potentiometer that dims the LED
- `PFI_LED`: the pin of the LED (I suggest a red one)

Compiling the Manager from sources

Required: IntelliJ 2018.2 with Bash Support plugin, Java >8 - To build the Debian installer, run configuration "Generate Debian package" - The utput Debian package will be generated in `OpenFocuser-Manager/deb-builder/OpenFocuser-Manager.deb` - To compile the universal `jar` executable, start

configuration *Run Windows-compatible Jar* - The output archive will be generated in `OpenFocuser-Manager/out/artifacts/OpenFocuser_Manager_Windows_jar/OpenFocuser-Manager.jar` - To create Javadocs, use the dedicated tool in IntelliJ

OpenFocuser-Manager used libraries and resources

OpenFocuser-Manager uses some third party libraries. Their `jar` Packages can be found in the `OpenFocuser-Manager/lib` folder.

- Gson by Google, license
- Commons CLI by Apache Commons, license
- INDI for Java by Zerjillo, in module `OpenFocuser-Manager/I4J` with the following dependencies:
  - Ostermiller Java Utilities by Stephen Ostermiller, license
- jSSC by scream3r, license
- JSch by JCraft, license
- Radiance, Neon and Substance L&F by kirill-grouchnikov, license
- Materia Design icons by Google (icons are in the `OpenFocuser-Manager/src/marcocipriani/openfocuser/manager/res` folder), license
- GitHub logo, logo and Octocat terms of use:
  - *GITHUB®, the GITHUB® logo design, OCTOCAT® and the OCTOCAT® logo design are exclusive trademarks registered in the United States by GitHub, Inc.*

Behind the scenes of the Plus edition

To work, the OpenFocuser Plus INDI driver uses `socat` and creates two virtual devices (sockets). Let's say, for example, that `/dev/port1` and `/dev/port2` are created: the first virtual port is used to read whatever is sent to the second one and the end user will be asked to connect the MoonLite driver to /dev/port2. OpenFocuser will forward **every** byte sent to port2 to the real Arduino, plus command `:AVxxxx#`, where `xxxx` is an hex number that represent the target pin and its new value, to change the state of a pin, and command `:RS#` to reset the board.

License

Forking and issues

Feel free to submit pull requests, report an issue or suggest new features! Also, new mounting brackets are welcome!