AdvPiStepper Documentation

Release 0.9.0.alpha

Thomas Holland

Oct 28, 2020

Contents:

AdvPiStepper is a driver for all kinds of stepper motors, written in Python for the Raspberry Pi, using the pigpio library.

CHAPTER 1

Features

“Here comes the Hotstepper” – Ini Kamoze

- Uses acceleration and deceleration ramps.
- Fairly tight timing up to approx. 1500 steps per second (on Raspberry Pi 4)[1].
- Complete API for relative and absolute moves, rotations and continuous running.
- Runs in the background. Motor movements can be blocking or non-blocking.
- Support for microstepping (depending on the driver).
- **Support for any unipolar stepper motors, like:**
  - 28BYJ-48 (very cheap geared stepper)
- **{TODO} Support for Bipolar stepper drivers / dual H-Bridges like the**
  - L293(D)
  - DRV8833
- **{TODO} Support for Step/Direction controllers like**
  - A4988
  - DRV8825
  - STSPIN220 / 820
- Other drivers should be easy to implement
- Licensed under the very permissive MIT license.
- 100% Python, no dependencies except pigpio.

[1] At high step rates occasional stutters may occur when some Python / Linux background tasks run.

1.1 Uses

AdvPiStepper is suitable for

- Python projects that need to accuratly control a single stepper motor at reasonable speeds.
- Stepper motor experiments and prototyping.

It is not suitable for

- multi-axis stepper motor projects
- high speeds (> 1500 steps per second)

1.2 Caveats

- Currently no support for multiple motors. Single motor only.
- 100% Python, therefore no realtime performance - jitter and stutters may occur.

CHAPTER 2

Requirements

“One small step for [a] man” – Neil Armstrong

AdvPiStepper uses the pigpio library to access the Raspberry Pi GPIO pins. It requires at least V76 of the library, which at the time of writing has not yet been uploaded to PyPI.org and therefore has to be installed manually.

A multicore Raspberry Pi (Model 2/3/4) is recommended so that the stepper engine with its critical timings can run on a seperate core. Single Core Pi Models (or heavy load on more than one core) will have timing jitter - neither Linux nor Python is really suited for these realtime uses.

CHAPTER 3

Usage

“A journey of a thousand miles begins with a single step” – Laozi

3.1 Installation

3.1.1 pigpio

AdvPiStepper requires the pigpio library to work. If the Remote GPIO has been enabled in the Raspberry Pi configuration tool, then the pigpio daemon should already be installed and running. Run the following to check if pigpio daemon is installed and its version number:

```
$ pigpiod -v
76
```

If either pigpio is not installed or has a version smaller than 76 (the minimum version required by AdvPiStepper), then refer to the pigpio download & install page on how to install pigpio.

3.1.2 AdvPiStepper

AdvPiStepper can be simply installed with

```
$ pip install advpistepper
```

3.2 Usage

AdvPiStepper is very simple to use. Here is a small example using the `28BYJ-48` driver:

```python
import advpistepper

driver = advpistepper.Driver28BYJ48(pink=23, orange=25, yellow=24, blue=22)
stepper = advpistepper.AdvPiStepper(driver)
stepper.move(100, block = True)
stepper.move(-100)
while stepper.
```

This example will move the stepper motor 100 steps forward, waiting for it to finish, then move it 100 steps backward without waiting. Besides the obvious import of advpistepper, using it requires to instantiate a driver.

AdvPiStepper comes with multiple generic and specific drivers, refer to the *Drivers* Section of the documentation for more details. In this example the 28BYJ-48 Driver is used which needs four arguments, the gpio numbers that the motor is connected to.

When using a motor with a step & direction interface the driver can instantiated like this, with the step signal on pin 22 and the direction signal on pin 23

```python
driver = advpistepper.DriverStepDirGeneric(step=22, direction=23)
```

The next line of the example initializes the stepper engine. It needs the driver as an argument (without it defaults to a no-GPIO driver). It can take an optional argument with a Dict containing parameters to overwrite the build in default parameters.

The last two lines of the example first move the stepper 100 steps forward, waiting for the move to finish, then 100 steps backwards without waiting, that is with the move running in the background.

For all commands of AdvPiStepper refer to `the API`

3.3 Tuning

To get the best performance from AdvPiStepper there should be as few background processes running as possible. For expample, on the AdvPiStepper development system (Raspi 4) the Desktop process does interfere with the AdvPiStepper process about every 500ms causing step delays of a few milliseconds, enough to cause late step pulses at high speeds (>500 steps per second)

If AdvPiStepper is called with root privileges (sudo) it will decrease the niceness of the backend process to -10. This improves the timing at high speeds somewhat due to less interference by normal user processes.

CHAPTER 4

API

`advpistepper.`**`stepper`**

alias of `advpistepper.stepper`

Warning: This program is not finished. It was uploaded to GitHub as a backup. Feel free to look at the source code and give feedback, but do not expect it to work in any shape or form.

CHAPTER 5

Features

"Here comes the Hotstepper" – Ini Kamoze

- Uses acceleration and deceleration ramps.
- Fairly tight timing up to approx. 1500 steps per second (on Raspberry Pi 4)[1].
- Complete API for relative and absolute moves, rotations and continuous running.
- Runs in the background. Motor movements can be blocking or non-blocking.
- Support for microstepping (depending on the driver).
- **Support for any unipolar stepper motors, like:**
  - 28BYJ-48 (very cheap geared stepper)
- **{TODO} Support for Bipolar stepper drivers / dual H-Bridges like the**
  - L293(D)
  - DRV8833
- **{TODO} Support for Step/Direction controllers like**
  - A4988
  - DRV8825
  - STSPIN220 / 820
- Other drivers should be easy to implement
- Licensed under the very permissive MIT license.
- 100% Python, no dependencies except pigpio.

[1] At high step rates occasional stutters may occur when some Python / Linux background tasks run.

5.1 Uses

AdvPiStepper is suitable for

- Python projects that need to accuratly control a single stepper motor at reasonable speeds.
- Stepper motor experiments and prototyping.

It is not suitable for

- multi-axis stepper motor projects
- high speeds (> 1500 steps per second)

5.2 Caveats

- Currently no support for multiple motors. Single motor only.
- 100% Python, therefore no realtime performance - jitter and stutters may occur.

CHAPTER 6

Requirements

“One small step for [a] man” – Neil Armstrong

AdvPiStepper uses the pigpio library to access the Raspberry Pi GPIO pins. It requires at least V76 of the library, which at the time of writing has not yet been uploaded to PyPI.org and therefore has to be installed manually.

A multicore Raspberry Pi (Model 2/3/4) is recommended so that the stepper engine with its critical timings can run on a seperate core. Single Core Pi Models (or heavy load on more than one core) will have timing jitter - neither Linux nor Python is really suited for these realtime uses.

CHAPTER 7

Usage

"A journey of a thousand miles begins with a single step" – Laozi

7.1 Installation

7.1.1 pigpio

AdvPiStepper requires the pigpio library to work. If the Remote GPIO has been enabled in the Raspberry Pi configuration tool, then the pigpio daemon should already be installed and running. Run the following to check if pigpio daemon is installed and its version number:

```
$ pigpiod -v
76
```

If either pigpio is not installed or has a version smaller than 76 (the minimum version required by AdvPiStepper), then refer to the pigpio download & install page on how to install pigpio.

7.1.2 AdvPiStepper

AdvPiStepper can be simply installed with

```
$ pip install advpistepper
```

7.2 Usage

AdvPiStepper is very simple to use. Here is a small example using the `28BYJ-48` driver:

```
import advpistepper

driver = advpistepper.Driver28BYJ48(pink=23, orange=25, yellow=24, blue=22)
stepper = advpistepper.AdvPiStepper(driver)
stepper.move(100, block = True)
stepper.move(-100)
while stepper.
```

This example will move the stepper motor 100 steps forward, waiting for it to finish, then move it 100 steps backward without waiting. Besides the obvious import of advpistepper, using it requires to instantiate a driver.

AdvPiStepper comes with multiple generic and specific drivers, refer to the *Drivers* Section of the documentation for more details. In this example the 28BYJ-48 Driver is used which needs four arguments, the gpio numbers that the motor is connected to.

When using a motor with a step & direction interface the driver can instantiated like this, with the step signal on pin 22 and the direction signal on pin 23

```
driver = advpistepper.DriverStepDirGeneric(step=22, direction=23)
```

The next line of the example initializes the stepper engine. It needs the driver as an argument (without it defaults to a no-GPIO driver). It can take an optional argument with a Dict containing parameters to overwrite the build in default parameters.

The last two lines of the example first move the stepper 100 steps forward, waiting for the move to finish, then 100 steps backwards without waiting, that is with the move running in the background.

For all commands of AdvPiStepper refer to `the API`

7.3 Tuning

To get the best performance from AdvPiStepper there should be as few background processes running as possible. For expample, on the AdvPiStepper development system (Raspi 4) the Desktop process does interfere with the AdvPiStepper process about every 500ms causing step delays of a few milliseconds, enough to cause late step pulses at high speeds (>500 steps per second)

If AdvPiStepper is called with root privileges (sudo) it will decrease the niceness of the backend process to -10. This improves the timing at high speeds somewhat due to less interference by normal user processes.

CHAPTER 8

Theory of Operation

“Step by step, Heart to heart, Left, right, left” – Martika

CHAPTER 9

History

AdvPiStepper was started for a Raspberry Pi project where I needed to move a stepper motor for 400 +/- a few steps. I wanted acceleration and deceleration ramps because an early Arduino based prototype had them. As I could not find any suitable RPi library I started this programm, which quickly spiraled out of control and became this multipurpose stepper motor controller.

V0.9 Work in Progress - Not officially released

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Index