

Forum:[Hack Forums](#)

Thread: [Controlling bipolar stepper with RPI](#)

SIM-1517404 - July 31, 2017, 10:23 am

Hi,

I'm working on controlling a bipolar stepper motor with RPI using Python. The motor has 200 steps. I want to rotate the motor in intervals of anywhere between 12-18. I do believe if I use half stepping I can increase the amount of steps to 400 on the motor, which gives me multiples of 16.

I'm struggling with the sequence for halfstepping. The code I'm currently using:

```
Code import RPi.GPIO as GPIO
import time
```

```
# Variables
```

```
delay = 0.0055
steps = 50
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

```
# Enable pins for IN1-4 to control step sequence
```

```
coil_A_1_pin = 18
coil_A_2_pin = 23
coil_B_1_pin = 24
coil_B_2_pin = 25
```

```
# Set pin states
```

```
GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)
```

```
# Function for step sequence
```

```
def setStep(w1, w2, w3, w4):
    GPIO.output(coil_A_1_pin, w1)
    GPIO.output(coil_A_2_pin, w2)
    GPIO.output(coil_B_1_pin, w3)
    GPIO.output(coil_B_2_pin, w4)
```

```
# loop through step sequence based on number of steps
```

```
for i in range(0, steps):
```

```
setStep(1,0,1,0)
time.sleep(delay)
setStep(0,1,1,0)
time.sleep(delay)
setStep(0,1,0,1)
time.sleep(delay)
setStep(1,0,0,1)
time.sleep(delay)
```

The code rotates the motor one full revolution.

datguy.dev - July 31, 2017, 3:40 pm

So the motor has 200 steps, you send it only 50, but it moves a full rotation. What stepper motor and driver are you using exactly? If you want to post a pic of your wiring.

SIM-1517404 - August 2, 2017, 5:22 am

Well, the sequence steps the motor 4 times, and I loop through the sequence 50 times, adding up to a total of 200 steps. Using a 5V motor, bought it at a local store, it's from Luxorparts. And using a TB6612fng driver.

koolkid - August 2, 2017, 6:08 am

(Aug 2, 2017 05:22 AM)Swatchi Wrote: [\[CITE\]](#) Well, the sequence steps the motor 4 times, and I loop through the sequence 50 times, adding up to a total of 200 steps. Using a 5V motor, bought it at a local store, it's from Luxorparts. And using a TB6612fng driver.
?

post a pic or 2 of your setup

Jomaki - August 2, 2017, 6:42 am

Given in the for loop which actually controls the stepping there are four separate setStep commands, and you are doing four times less loops than there are steps, I am willing to bet that each of these setSteps does a singular step.

The code to explain what I am thinking:

```
Code import RPi.GPIO as GPIO
import time
```

```
# Variables
```

```
delay = 0.0055
steps = 200
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

```
# Enable pins for IN1-4 to control step sequence
```

```
coil_A_1_pin = 18
coil_A_2_pin = 23
coil_B_1_pin = 24
coil_B_2_pin = 25
```

```
# Set pin states
```

```
GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)
```

Function for step sequence

```
def set_step(w1, w2, w3, w4):
    GPIO.output(coil_A_1_pin, w1)
    GPIO.output(coil_A_2_pin, w2)
    GPIO.output(coil_B_1_pin, w3)
    GPIO.output(coil_B_2_pin, w4)
```

loop through step sequence based on number of steps

gives us a list so we can essentially iterate through each of the different step values
meaning we can iterate through each "part" of the original stepping sequence

```
step_values = [{"w1":1, "w2":0, "w3":1, "w4":0},
                {"w1":0, "w2":1, "w3":1, "w4":0},
                {"w1":0, "w2":1, "w3":0, "w4":1},
                {"w1":1, "w2":0, "w3":0, "w4":1}]
```

current_step_count = 0

```
def step_motor():
    needed_step_values = step_values[current_step_count % 4]
    set_step(needed_step_values["w1"], needed_step_values["w2"], needed_step_values["w3"],
needed_step_values["w4"])
    time.sleep(delay)
    current_step_count += 1
```

this should turn everything off to stop them moving -- not sure if this is necessary
trying commenting it out if it doesn't work with it in
set_step(0, 0, 0, 0)

```
for i in range(0, steps):
    step_motor()
```

As I said in the comments, I am not certain if line 54 would be necessary. (note: I also changed setStep to set_step to be coherent with the rest of the code).

Try this out and let me know if it works.

EDIT: Just realised I added to the step count too early and fixed it.

SIM-1517404 - August 15, 2017, 12:01 pm

(Aug 2, 2017 06:42 AM)Jomaki Wrote: [\[CITE\]](#) Given in the for loop which actually controls the stepping there are four separate setStep commands, and you are doing four times less loops than there are steps, I am willing to bet that each of these setSteps does a singular step.

The code to explain what I am thinking:

```
Code import RPi.GPIO as GPIO
import time
```

```
# Variables
```

```
delay = 0.0055
steps = 200
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

```
# Enable pins for IN1-4 to control step sequence
```

```
coil_A_1_pin = 18
coil_A_2_pin = 23
coil_B_1_pin = 24
coil_B_2_pin = 25
```

```
# Set pin states
```

```
GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)
```

```
# Function for step sequence
```

```
def set_step(w1, w2, w3, w4):
    GPIO.output(coil_A_1_pin, w1)
    GPIO.output(coil_A_2_pin, w2)
    GPIO.output(coil_B_1_pin, w3)
    GPIO.output(coil_B_2_pin, w4)
```

```
# loop through step sequence based on number of steps
```

```
# gives us a list so we can essentially iterate through each of the different step values
# meaning we can iterate through each "part" of the original stepping sequence
```

```
step_values = [{"w1":1, "w2":0, "w3":1, "w4":0},
               {"w1":0, "w2":1, "w3":1, "w4":0},
               {"w1":0, "w2":1, "w3":0, "w4":1},
               {"w1":1, "w2":0, "w3":0, "w4":1}]
```

```
current_step_count = 0
```

```
def step_motor():
    needed_step_values = step_values[current_step_count % 4]
    set_step(needed_step_values["w1"], needed_step_values["w2"], needed_step_values["w3"],
needed_step_values["w4"])
    time.sleep(delay)
    current_step_count += 1
```

```
# this should turn everything off to stop them moving -- not sure if this is necessary
```

```
# trying commenting it out if it doesn't work with it in
set_step(0, 0, 0, 0)
```

```
for i in range(0, steps):
    step_motor()
```

As I said in the comments, I am not certain if line 54 would be necessary. (note: I also changed setStep to set_step to be coherent with the rest of the code).

Try this out and let me know if it works.

EDIT: Just realised I added to the step count too early and fixed it.
Sorry for the late response, I've been away for a couple of weeks.

The result I get from running the code is:

Code Traceback (most recent call last):

```
File "StepperTest.py", line 68, in <module>
    step_motor()
File "StepperTest.py", line 58, in step_motor
    needed_step_values = step_values[current_step_count % 4]
UnboundLocalError: local variable 'current_step_count' referenced before assignment
```

datguy.dev - August 16, 2017, 12:30 pm

(Aug 15, 2017 12:01 PM)Swatchi Wrote: [\[CITE\]](#) Sorry for the late response, I've been away for a couple of weeks.

The result I get from running the code is:

Code Traceback (most recent call last):

```
File "StepperTest.py", line 68, in <module>
    step_motor()
File "StepperTest.py", line 58, in step_motor
    needed_step_values = step_values[current_step_count % 4]
UnboundLocalError: local variable 'current_step_count' referenced before assignment
```

PHP Code

```
<?php
current_step_count = 0
def step_motor():
    global current_step_count #ADD ME
    needed_step_values = step_values[current_step_count % 4]
    set_step(needed_step_values["w1"], needed_step_values["w2"], needed_step_values["w3"],
needed_step_values["w4"])
    time.sleep(delay)
    current_step_count += 1
```

See #add me. If you do this (current_step_count += 1) you need to add the global statement.

SIM-1517404 - August 17, 2017, 8:24 am

(Aug 16, 2017 12:30 PM)datguy.dev Wrote: [\[CITE\]](#) PHP Code

```
<?php
current_step_count = 0
def step_motor():
    global current_step_count #ADD ME
```

```
needed_step_values = step_values[current_step_count % 4]
set_step(needed_step_values["w1"], needed_step_values["w2"], needed_step_values["w3"],
needed_step_values["w4"])
time.sleep(delay)
current_step_count += 1
```

See #add me. If you do this (current_step_count += 1) you need to add the global statement.
It does one full rotation now. How can I change the sequence to allow half-stepping? I would like 400 steps per revolution so that I can have intervals of 16. 25 steps, delay for 5 seconds and repeat 16 times.