

# Half step Rotary Encoder

1.0.1

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>3 speed Rotary Encoder Half Step library for Arduino</b>	<b>1</b>
<b>2</b>	<b>Class Index</b>	<b>7</b>
2.1	Class List . . . . .	7
<b>3</b>	<b>File Index</b>	<b>9</b>
3.1	File List . . . . .	9
<b>4</b>	<b>Class Documentation</b>	<b>11</b>
4.1	RotaryHalfStep Class Reference . . . . .	11
4.1.1	Detailed Description . . . . .	11
4.1.2	Constructor & Destructor Documentation . . . . .	11
4.1.2.1	RotaryHalfStep() . . . . .	11
4.1.3	Member Function Documentation . . . . .	12
4.1.3.1	getSensitivity() . . . . .	12
4.1.3.2	read() . . . . .	12
4.1.3.3	setSensitivity() . . . . .	12
<b>5</b>	<b>File Documentation</b>	<b>15</b>
5.1	src/RotaryHalfStep.cpp File Reference . . . . .	15
5.1.1	Detailed Description . . . . .	15
5.2	src/RotaryHalfStep.h File Reference . . . . .	15
5.2.1	Detailed Description . . . . .	16
	<b>Index</b>	<b>17</b>



## Chapter 1

# 3 speed Rotary Encoder Half Step library for Arduino

This is an optimized three speed Rotary Encoder library for Arduino which supports:

- Half step Rotary Encoder types.
- Detect three rotation speeds.
- Configurable rotation speed sensitivity.
- Polling and interrupt based.
- Single or multiple Rotary Encoders.
- Optional Rotary button.
- Pin state table in flash.

### Half step / half step Rotary Encoders

The difference between a half step or half step Rotary Encoder type is how the data signals of the two pins are generated. It depends on the mechanical construction of the notches and contacts inside the Rotary Encoder.

Please refer to the [ErriezRotaryEncoderFullStep](#) library for full step Rotary Encoders. Experiment with the full step and half step libraries which works optimal for your Rotary Encoder.

### Hardware

Connect the two rotary pins to the DIGITAL pins of an Arduino board.

A third rotary button pin is not used in the Rotary library, but can be used in the sketch.

Tested with Arduino IDE v1.8.5 on hardware:

- Arduino UNO
- Arduino Nano
- Arduino Micro
- Arduino Pro or Pro Mini
- Arduino Mega or Mega2560
- Arduino Leonardo
- WeMos D1 R2 & mini (ESP8266)

## Interrupts

Both rotary pins must be connected to a DIGITAL pin with interrupt support, such as `INT0` or `INT1`. This is chip specific. Please refer to the documentation of your board or `attachInterrupt()`.

## Arduino UNO hardware

The connection below can be used for polled and interrupts. An optional button pin can be connected to DIGITAL pin 4.

Rotary pin	Arduino UNO/NANO/Mega2560/Leonardo board
1	D2 (INT0)
2	D3 (INT1)
Button (optional)	D4
GND	GND

## Arduino WeMos D1 R2 & mini (ESP8266) hardware

Note that some ESP8266 pins mixes ESP8622 GPIO pins with Arduino digital pins. Connect a Rotary Encoder to the following pins which can be used with polled and interrupt examples:

Rotary pin	ESP8622 pin	Text on board WeMos D1 R2
1	GPIO13	D7 MOSI
2	GPIO12	D6 MISO
Button (optional)	GPIO14	D5 SCK
LED (Not used)	GPIO2	D4
GND	GND	GND

**Note:** An external pull-up resistor is required when a pin does not have an internal pull-up.

```
{c++}
// Connect the rotary pins to the WeMos D1 R2 board:
#define ROTARY_PIN1      12
#define ROTARY_PIN2      13
#define ROTARY_BUTTON_PIN 14
```

## Examples

The following examples are available:

- Rotary | Interrupt | `InterruptHalfStepBasic`
- Rotary | Interrupt | `InterruptHalfStepButton`
- Rotary | Interrupt | `InterruptHalfStepCounter`
- Rotary | Polled | `PolledHalfStepBasic`
- Rotary | Polled | `PolledHalfStepButton`
- Rotary | Polled | `PolledHalfStepCounter`
- Rotary | Polled | `PolledHalfStepMultiple`

## Usage

### Read rotary with polling

```
{c++}
#include <RotaryHalfStep.h>

// Connect rotary pins to the DIGITAL pins of the Arduino board
#define ROTARY_PIN1 2
#define ROTARY_PIN2 3

// Enable ONE of the three constructors below with different number of arguments:

// Initialize half step rotary encoder, default pull-up enabled, default
// sensitive=100
RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2);

// Or initialize half step rotary encoder, pull-up disabled, default sensitive=100
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, false);

// Or initialize half step rotary encoder, pull-up enabled, sensitive 1..255
// A higher value is more sensitive
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, true, 150);

void loop()
{
    int rotaryState = rotary.read();

    // rotaryState = -3: Counter clockwise turn, multiple notches fast
    // rotaryState = -2: Counter clockwise turn, multiple notches
    // rotaryState = -1: Counter clockwise turn, single notch
    // rotaryState = 0: No change
    // rotaryState = 1: Clockwise turn, single notch
    // rotaryState = 2: Clockwise turn, multiple notches
    // rotaryState = 3: Clockwise turn, multiple notches fast
}
```

### Read rotary with interrupts

```
{c++}
#include <RotaryHalfStep.h>

// Connect rotary pins to Arduino DIGITAL pins with interrupt support:
//
// +-----+-----+-----+
// |          Board          | DIGITAL interrupt pins |
// +-----+-----+-----+
// | Uno, Nano, Mini, other 328-based | 2, 3 |
// | Mega, Mega2560, MegaADK         | 2, 3, 18, 19, 20, 21 |
// | Micro, Leonardo, other 32u4-based | 0, 1, 2, 3, 7 |
// +-----+-----+-----+
//
#define ROTARY_PIN1 2
#define ROTARY_PIN2 3

// Enable ONE of the three constructors below with different number of arguments:

// Initialize half step rotary encoder, default pull-up enabled, default
// sensitive=100
RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2);

// Or initialize half step rotary encoder, pull-up disabled, default sensitive=100
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, false);

// Or initialize half step rotary encoder, pull-up enabled, sensitive 1..255
// A higher value is more sensitive
// RotaryHalfStep rotary(ROTARY_PIN1, ROTARY_PIN2, true, 150);

void setup()
{
    // Initialize pin change interrupt on both rotary encoder pins
    attachInterrupt(digitalPinToInterrupt(ROTARY_PIN1), rotaryInterrupt, CHANGE);
    attachInterrupt(digitalPinToInterrupt(ROTARY_PIN2), rotaryInterrupt, CHANGE);
}

void rotaryInterrupt()
{
    int rotaryState = rotary.read();

    // rotaryState = -3: Counter clockwise turn, multiple notches fast
```

```
// rotaryState = -2: Counter clockwise turn, multiple notches
// rotaryState = -1: Counter clockwise turn, single notch
// rotaryState = 0: No change
// rotaryState = 1: Clockwise turn, single notch
// rotaryState = 2: Clockwise turn, multiple notches
// rotaryState = 3: Clockwise turn, multiple notches fast
}
```

## Library installation using Git

Git is the preferred way to keep this library up to date, because the Arduino Library manager does not update as long as this library is not added to the official Arduino Library database.

### Install Git client for Windows

Install a [Git client for Windows](#).

### Install Git client for Linux

Open a command prompt and install a Git client for Linux, such as Debian Ubuntu:

```
sudo apt-get install git
```

### Get Arduino libraries directory

This library must be installed in the Arduino Sketchbook library subdirectory.

To retrieve the Arduino Sketchbook directory, open the Arduino IDE Preferences dialog box via: **File | Preferences | Settings** tab and copy the Sketchbook location.

For example on:

- **Windows** : C:\Users\User\Documents\Arduino
- **Linux**: /home/user/Arduino

### Clone this library

Clone this library by opening a command prompt:

- **Windows**: (Windows key + R, Type cmd + [ENTER])
- **Linux**: Depends on your version.

Then type:

```
# Change directory to the sketchbook directory as configured in the Arduino IDE:
# Windows:
cd C:\Users\User\Documents\Arduino
# Linux:
cd ~/Arduino

# Go to the libraries subdirectory
cd libraries

# Run the git clone library once:
git clone https://github.com/Erriez/ErriezRotaryEncoderHalfStep.git
```

**IMPORTANT:** Restart the Arduino IDE.



### Update this library

Open a command prompt and type:

```
# Change directory to the sketchbook directory as configured in the Arduino IDE:
# Windows:
cd C:\Users\User\Documents\Arduino
# Linux:
cd ~/Arduino

# Go to the libraries subdirectory
cd libraries

# Update the library:
git pull
```

**IMPORTANT:** Restart the Arduino IDE.

### Library installation with a ZIP

This method is not preferred, because updates should be manually installed.

1. Download [the latest version here](#).
2. Open the Arduino IDE.
3. Sketch | Include Library | Add .ZIP library...
4. Browse to the downloaded ZIP.
5. Restart the IDE.



## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">RotaryHalfStep</a>	
Half step Rotary Encoder class . . . . .	11



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

src/ <a href="#">RotaryHalfStep.cpp</a>	
Three speed half step Rotary Encoder library for Arduino . . . . .	15
src/ <a href="#">RotaryHalfStep.h</a>	
Three speed half step Rotary Encoder library for Arduino . . . . .	15



## Chapter 4

# Class Documentation

### 4.1 RotaryHalfStep Class Reference

Half step Rotary Encoder class.

```
#include <RotaryHalfStep.h>
```

#### Public Member Functions

- [RotaryHalfStep](#) (uint8\_t pin1, uint8\_t pin2, bool pullUp=true, uint8\_t sensitivity=100)  
*Constructor half step Rotary Encoder.*
- int [read](#) ()  
*Read Rotary Encoder state.*
- void [setSensitivity](#) (uint8\_t sensitivity)  
*Set sensitivity value.*
- uint8\_t [getSensitivity](#) ()  
*Get sensitivity value.*

#### 4.1.1 Detailed Description

Half step Rotary Encoder class.

Definition at line 42 of file RotaryHalfStep.h.

#### 4.1.2 Constructor & Destructor Documentation

##### 4.1.2.1 RotaryHalfStep()

```
RotaryHalfStep::RotaryHalfStep (  
    uint8_t pin1,  
    uint8_t pin2,  
    bool pullUp = true,  
    uint8_t sensitivity = 100 )
```

Constructor half step Rotary Encoder.

**Parameters**

<i>pin1</i>	Rotary Encoder pin 1
<i>pin2</i>	Rotary Encoder pin 2
<i>pullUp</i>	true: Enable internal pull-up on Rotary Encoder pins [default argument]. false: Disable internal pull-up on Rotary Encoder pins.
<i>sensitivity</i>	Set sensitivity rotation speed value 0..255. A higher is more sensitive for rotation speed, a smaller value is less sensitive or will disable speed detection. Default is 100.

Definition at line 88 of file RotaryHalfStep.cpp.

### 4.1.3 Member Function Documentation

#### 4.1.3.1 `getSensitivity()`

```
uint8_t RotaryHalfStep::getSensitivity ( )
```

Get sensitivity value.

**Returns**

Sensitivity value 0..255.

Definition at line 180 of file RotaryHalfStep.cpp.

#### 4.1.3.2 `read()`

```
int RotaryHalfStep::read ( )
```

Read Rotary Encoder state.

Call this function as fast as possible to prevent missing pin changes. This seems to work for most rotary encoders when calling this function within 10ms in an endless loop.

The sensitivity value is used to calculate three rotation speeds by measuring the speed of the Rotary Encoder pin changes. The rotation speed depends on the number of Rotary notches and knob size. The value should be experimentally determined.

**Returns**

Rotary speed and direction -3: Counter clockwise turn, multiple notches fast -2: Counter clockwise turn, multiple notches -1: Counter clockwise turn, single notch 0: No change 1: Clockwise turn, single notch 2: Clockwise turn, multiple notches 3: Clockwise turn, multiple notches fast

Definition at line 120 of file RotaryHalfStep.cpp.

#### 4.1.3.3 `setSensitivity()`

```
void RotaryHalfStep::setSensitivity (
    uint8_t sensitivity )
```

Set sensitivity value.



## Parameters

<i>sensitivity</i>	Sensitivity value 0..255
--------------------	--------------------------

Definition at line 169 of file RotaryHalfStep.cpp.

The documentation for this class was generated from the following files:

- src/[RotaryHalfStep.h](#)
- src/[RotaryHalfStep.cpp](#)



## Chapter 5

# File Documentation

### 5.1 src/RotaryHalfStep.cpp File Reference

Three speed half step Rotary Encoder library for Arduino.

```
#include <pgmspace.h>
#include "RotaryHalfStep.h"
```

#### Macros

- `#define DIR_NONE 0x00`
- `#define DIR_CW 0x10`
- `#define DIR_CCW 0x20`
- `#define RHS_START 0x00`
- `#define RHS_CCW_BEGIN 0x01`
- `#define RHS_CW_BEGIN 0x02`
- `#define RHS_START_M 0x03`
- `#define RHS_CW_BEGIN_M 0x04`
- `#define RHS_CCW_BEGIN_M 0x05`

#### 5.1.1 Detailed Description

Three speed half step Rotary Encoder library for Arduino.

RotaryHalfStep.cpp

Source: <https://github.com/Erriez/ErriezRotaryEncoderHalfStep>

### 5.2 src/RotaryHalfStep.h File Reference

Three speed half step Rotary Encoder library for Arduino.

```
#include <Arduino.h>
```

## Classes

- class [RotaryHalfStep](#)  
*Half step Rotary Encoder class.*

### 5.2.1 Detailed Description

Three speed half step Rotary Encoder library for Arduino.

[RotaryHalfStep.h](#)

Source: <https://github.com/Erriez/ErriezRotaryEncoderHalfStep>

# Index

getSensitivity  
    RotaryHalfStep, [12](#)

read  
    RotaryHalfStep, [12](#)  
RotaryHalfStep, [11](#)  
    getSensitivity, [12](#)  
    read, [12](#)  
    RotaryHalfStep, [11](#)  
    setSensitivity, [12](#)

setSensitivity  
    RotaryHalfStep, [12](#)  
src/RotaryHalfStep.h, [15](#)  
src/RotaryHalfStep.cpp, [15](#)