

---

# **sparkfun\_de2120\_py**

***Release 0.0.01***

**SparkFun Electronics**

**May 26, 2021**



**CONTENTS:**

- 1 Contents 3**
- 2 Supported Platforms 5**
- 3 Dependencies 7**
- 4 Documentation 9**
- 5 Installation 11**
  - 5.1 PyPi Installation . . . . . 11
- 6 Example Use 13**
- 7 Table of Contents 15**
  - 7.1 API Reference . . . . . 15
    - 7.1.1 de2120\_barcode\_scanner . . . . . 15
  - 7.2 Example One - Serial Scan . . . . . 19
  - 7.3 Example 2 - Serial Settings . . . . . 20
  - 7.4 Example 3 - Send Command . . . . . 26
- 8 Indices and tables 29**
- Python Module Index 31**
- Index 33**



Python module for the [SparkFun 2D Barcode Scanner Breakout - DE2120](#).

This python package is a port of the existing [SparkFun DE2120 Arduino Library](#)



## CONTENTS

- *Supported Platforms*
- *Dependencies*
- *Installation*
- *Documentation*
- *Example Use*





## SUPPORTED PLATFORMS

The Qwiic Button Python package current supports the following platforms:

- [Raspberry Pi](#)



## DEPENDENCIES

This driver package depends on the pyserial package.



## DOCUMENTATION

The SparkFun 2D Barcode Scanner Breakout module documentation is hosted at [ReadTheDocs](#)



## INSTALLATION

### 5.1 PyPi Installation

This repository is hosted on PyPi as the [de2120-barcode-scanner](#) package. On systems that support PyPi installation via pip, this library is installed using the following commands

For all users (note: the user must have sudo privileges):

```
sudo pip install de2120-barcode-scanner
```

For the current user:

```
pip install de2120-barcode-scanner
```

To install, make sure the setuptools package is installed on the system.

Direct installation at the command line:

```
python setup.py install
```

To build a package for use with pip:

```
python setup.py sdist
```

A package file is built and placed in a subdirectory called dist. This package file can be installed using pip.

```
cd dist  
pip install de2120-barcode-scanner-<version>.tar.gz
```





## EXAMPLE USE

See the examples directory for more detailed use examples.

```
from __future__ import print_function
import de2120_barcode_scanner
import time
import sys

def run_example():

    print("\nSparkFun DE2120 Barcode Scanner Breakout Example 1")
    my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()

    if my_scanner.begin() == False:
        print("\nThe Barcode Scanner module isn't connected correctly to the system.␣
↪Please check wiring", \
            file=sys.stderr)
        return
    print("\nScanner ready!")

    scan_buffer = ""

    while True:
        scan_buffer = my_scanner.read_barcode()
        if scan_buffer:
            print("\nCode found: " + str(scan_buffer))
            scan_buffer = ""

            time.sleep(0.02)

if __name__ == '__main__':
    try:
        run_example()
    except (KeyboardInterrupt, SystemExit) as exErr:
        print("\nEnding Example 1")
        sys.exit(0)
```



## TABLE OF CONTENTS

### 7.1 API Reference

#### 7.1.1 de2120\_barcode\_scanner

Python module for the 2D Barcode Scanner.

This python package is a port of the existing [SparkFun DE2120 Arduino Library]([https://github.com/sparkfun/SparkFun\\_DE2120\\_Arduino\\_Library](https://github.com/sparkfun/SparkFun_DE2120_Arduino_Library))

**class** `de2120_barcode_scanner.DE2120BarcodeScanner`(*hard\_port=None*)

Initialize the library with the given port.

**Parameters** **hard\_port** – The port to use to communicate with the module, this is a serial port at 9600 baud rate.

**Returns** The DE2120BarcodeScanner object.

**Return type** Object

**USB\_mode**(*mode*)

Enable USB communication and set the mode. THIS WILL MAKE THE MODULE UNRESPONSIVE ON COM PORT

**Parameters** **mode** – string defining what USB mode to set the module in. Valid arguments are “KBD”, “HID”, “232”.

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**available**()

**Returns** the number of bytes in the serial receive buffer

**Return type** int

**begin**()

Initializes the device with basic settings. Calls the `is_connected()` function

**Returns** Returns true if initialization was successful

**Return type** bool

**change\_baud\_rate**(*baud*)

Change the serial baud rate for the barcode module. Default 115200

**Parameters** **baud** – baud rate to change to

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

**change\_buzzer\_tone(*tone*)**

Change the buzzer frequency between low, med, and high

**Parameters** **tone** – int that's 1 = low, 2 = med, 3 = high frequency

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

**change\_reading\_area(*percent*)**

Change the percentage of the frame to scan for barcodes

**Parameters** **percent** – Percentage of frame to scan. Valid values are 100, 80, 60, 40, 20 as stated in the DE2120 Scan Setting Manual

**Returns** true if command successfully sent, false otherwise

**Return type** bool

**disable\_all\_1D()**

Disable decoding of all 1D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**disable\_all\_2D()**

Disable decoding of all 2D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**disable\_boot\_beep()**

Disable beep on module startup

**Returns** true if command successfully sent, false otherwise

**Return type** bool

**disable\_decode\_beep()**

Disable beep on successful read

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

**disable\_image\_flipping()**

Disable mirror image reading

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**enable\_all\_1D()**

Enable decoding of all 1D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**enable\_all\_2D()**

Enable decoding of all 2D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**enable\_boot\_beep()**

Enable beep on module startup

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

**enable\_continuous\_read(*repeat\_interval=2*)**

Enable continuous reading of barcodes and set the time interval for same-code reads

**Parameters** **repeat\_interval** – int parameter. 0: same code output 1 times 1: continuous output with same code without interval 2: continuous output with same code, 0.5 second interval (default) 3: continuous output with same code, 1 second interval

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**enable\_decode\_beep()**

Enable beep on successful read

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

**enable\_image\_flipping()**

Enable mirror image reading

**Returns** true if the command successfully sent, false otherwise

**Return type** bool

**enable\_manual\_trigger()**

Disable the motion sensitive and continuous read mode. Return to the default trigger mode.

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**enable\_motion\_sense(*sensitivity=20*)**

Enable the motion sensitive read mode and set sensitivity level

**Parameters** **sensitivity** – int value. The smaller the sensitivity, the more sensitive. Values are taken from the DE2120 settings manual. Valid arguments are: 15 (very high), 20 (high/default), 30 (little high), 50 (general), 100 (low sensitivity)

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

**factory\_default()**

Send command to put the module back into factory default settings. This will disconnect the module from the serial port.

**Returns** True if command successfully received, false

otherwise. :rtype: bool

**is\_connected()**

Ask the DE2120 for the firmware version.

**Returns** Returns true if the DE2120 responds with an ACK.

Retruns false otherwise. :rtype: bool

**light\_off()**

Turn white illumination LED off

**Returns** true if command successfully sent, false otherwise

**Return type** bool

**light\_on()**

Turn white illumination LED on

**Returns** true if command successfully sent, false otherwise

**Return type** bool

**read()**

**Returns** the first byte on the serial port

**Return type** int

**read\_barcode()**

Read from the serial buffer until we hit a new line character

**Returns** the string in the serial buffer

**Return type** bool

**reticle\_off()**

Turn red scan line off

:return true if command successfully sent, false otherwise :rtype: bool

**reticle\_on()**

Turn red scan line on

**Returns** true if command successfully sent, false otherwise

**Return type** bool

**send\_command(cmd, arg="")**

Create command string and send to DE2120 over serial port. Check serial buffer for a response

**Parameters**

- **cmd** – The command name
- **arg** – The command variation, if there is one

**Returns** True if the response from DE2120 contains the

ACK character, false otherwise. :rtype: bool

**start\_scan()**

Start reading when in trigger mode (default)

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**stop\_scan()**

Stop reading when in trigger mode. Module will automatically stop reading after a few seconds

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

## 7.2 Example One - Serial Scan

Listing 1: examples/de2120\_ex1\_serial\_scan.py

```

1  #!/usr/bin/env python
2  #-----
3  # qde2120_ex1_serial_scan.py
4  #-----
5  #
6  # Written by Priyanka Makin @ SparkFun Electronics, April 2021
7  #
8  # This example demonstrates how to get the scanner connected and will output
9  # and barcode it sees.
10 #
11 # NOTE: you must put the module into COM mode by scanning the PORVIC barcode
12 # in the datasheet. This will put the module in the correct mode to receive
13 # and transmit serial.
14 #
15 # This package has been developed on a Raspberry Pi 4. Connect the DE2120 Barcode
16 # Scanner Breakout directly to your Pi using a USB-C cable
17 #
18 # Do you like this library? Help support SparkFun. Buy a board!
19 #
20 #=====
21 # Copyright (c) 2021 SparkFun Electronics
22 #
23 # Permission is hereby granted, free of charge, to any person obtaining a copy
24 # of this software and associated documentation files (the "Software"), to deal
25 # in the Software without restriction, including without limitation the rights
26 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
27 # copies of the Software, and to permit persons to whom the Software is
28 # furnished to do so, subject to the following conditions:
29 #
30 # The above copyright notice and this permission notice shall be included in all
31 # copies or substantial portions of the Software.
32 #
33 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
34 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
35 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
36 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
37 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
38 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
39 # SOFTWARE.
40 #=====
41 # Example 1
42
43 from __future__ import print_function
44 import de2120_barcode_scanner
45 import time
46 import sys
47
48 def run_example():

```

(continues on next page)

(continued from previous page)

```

49
50 print("\nSparkFun DE2120 Barcode Scanner Breakout Example 1")
51 my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()
52
53 if my_scanner.begin() == False:
54     print("\nThe Barcode Scanner module isn't connected correctly to the system.␣
↳Please check wiring", \
55         file=sys.stderr)
56     return
57 print("\nScanner ready!")
58
59 scan_buffer = ""
60
61 while True:
62     scan_buffer = my_scanner.read_barcode()
63     if scan_buffer:
64         print("\nCode found: " + str(scan_buffer))
65         scan_buffer = ""
66
67     time.sleep(0.02)
68
69 if __name__ == '__main__':
70     try:
71         run_example()
72     except KeyboardInterrupt, SystemExit as exErr:
73         print("\nEnding Example 1")
74         sys.exit(0)

```

## 7.3 Example 2 - Serial Settings

Listing 2: examples/de2120\_ex2\_serial\_settings.py

```

1  #!/usr/bin/env python
2  #-----
3  # de2120_ex2_serial_settings.py
4  #-----
5  #
6  # Written by Priyanka Makin @ SparkFun Electronics, April 2021
7  #
8  # This example demonstrates how to configure the settings of the DE2120 Breakout
9  #
10 # NOTE: you must put the module into COM mode by scanning the PORVIC barcode
11 # in the datasheet. This will put the module in the correct mode to receive
12 # and transmit serial.
13 #
14 # This package has been developed on a Raspberry Pi 4. Connect the DE2120 Barcode
15 # Scanner Breakout directly to your Pi using a USB-C cable
16 #
17 # Do you like this library? Help support SparkFun. Buy a board!
18 #

```

(continues on next page)



(continued from previous page)

```

19 #=====
20 # Copyright (c) 2021 SparkFun Electronics
21 #
22 # Permission is hereby granted, free of charge, to any person obtaining a copy
23 # of this software and associated documentation files (the "Software"), to deal
24 # in the Software without restriction, including without limitation the rights
25 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
26 # copies of the Software, and to permit persons to whom the Software is
27 # furnished to do so, subject to the following conditions:
28 #
29 # The above copyright notice and this permission notice shall be included in all
30 # copies or substantial portions of the Software.
31 #
32 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
33 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
34 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
35 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
36 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
37 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
38 # SOFTWARE.
39 #=====
40 # Example 2
41
42 from __future__ import print_function
43 import de2120_barcode_scanner
44 import time
45 import sys
46 import serial
47
48 def flash_light(bar_scanner):
49     print("\n")
50     print("\n-----")
51     print("\n1) Enable flash light")
52     print("\n2) Disable flash light")
53     print("\n-----")
54
55     val = input("\nSelect an option number: ")
56
57     if val == '1':
58         print("\nWhite scan light on")
59         bar_scanner.light_on()
60     elif val == '2':
61         print("\nWhite scan light off")
62         bar_scanner.light_off()
63     else:
64         print("\nCommand not recognized")
65
66 def reticle(bar_scanner):
67     print("\n")
68     print("\n-----")
69     print("\n1) Enable reticle")
70     print("\n2) Disable reticle")

```

(continues on next page)

(continued from previous page)

```

71     print("\n-----")
72
73     val = input("\nSelect an option number: ")
74
75     if val == '1':
76         print("\nRed scan reticle on")
77         bar_scanner.reticle_on()
78     elif val == '2':
79         print("\nRed scan reticle off")
80         bar_scanner.reticle_off()
81     else:
82         print("\nCommand not recognized")
83
84 def decode_beep(bar_scanner):
85     print("\n")
86     print("\n-----")
87     print("\n1) Enable decode beep")
88     print("\n2) Disable decode beep")
89     print("\n-----")
90
91     val = input("\nSelect an option number: ")
92
93     if val == '1':
94         print("\nDecode beep turned on")
95         bar_scanner.enable_decode_beep()
96     elif val == '2':
97         print("\nDecode beep turned off")
98         bar_scanner.disable_decode_beep()
99     else:
100         print("\nCommand not recognized")
101
102 def boot_beep(bar_scanner):
103     print("\n")
104     print("\n-----")
105     print("\n1) Enable beep on module power on")
106     print("\n2) Disable beep on module power off")
107     print("\n-----")
108
109     val = input("\nSelect an option number: ")
110
111     if val == '1':
112         print("\nBeep on power on enabled")
113         bar_scanner.enable_boot_beep()
114     elif val == '2':
115         print("\nBeep on power on disabled")
116         bar_scanner.disable_boot_beep()
117     else:
118         print("\nCommand not recognized")
119
120 def change_buzz_freq(bar_scanner):
121     print("\n")
122     print("\n-----")

```

(continues on next page)

(continued from previous page)

```

123 print("\n1) Passive low frequency")
124 print("\n2) Passive medium frequency")
125 print("\n3) Passive high frequency")
126
127 val = input("\nSelect an option number: ")
128
129 if val == '1':
130     bar_scanner.change_buzzer_tone(int(val))
131 elif val == '2':
132     bar_scanner.change_buzzer_tone(int(val))
133 elif val == '3':
134     bar_scanner.change_buzzer_tone(int(val))
135 else:
136     print("\nCommand not recognized")
137
138 def image_flip(bar_scanner):
139     print("\n")
140     print("\n-----")
141     print("\n1) Turn on image flipping")
142     print("\n2) Turn off image flipping (default)")
143     print("\n-----")
144
145     val = input("\nSelect an option number: ")
146
147     if val == '1':
148         bar_scanner.enable_image_flipping()
149     elif val == '2':
150         bar_scanner.disable_image_flipping()
151     else:
152         print("\nCommand not recognized")
153
154 def reading_area(bar_scanner):
155     print("\n")
156     print("\n-----")
157     print("\n1) Full width (default)")
158     print("\n2) Center 80%")
159     print("\n3) Center 60%")
160     print("\n4) Center 40%")
161     print("\n5) Center 20%")
162     print("\n-----")
163
164     val = input("\nSelect an option number: ")
165
166     if val == '1':
167         print("\nScanning 100% of frame")
168         bar_scanner.change_reading_area(100)
169     elif val == '2':
170         print("\nScanning center 80% of frame")
171         bar_scanner.change_reading_area(80)
172     elif val == '3':
173         print("\nScanning center 60% of frame")
174         bar_scanner.change_reading_area(60)
175     elif val == '4':

```

(continues on next page)

(continued from previous page)

```

175     print("\nScanning center 40% of frame")
176     bar_scanner.change_reading_area(40)
177     elif val == '5':
178         print("\nScanning center 20% of frame")
179         bar_scanner.change_reading_area(20)
180     else:
181         print("\nCommand not recognized")
182
183 def reading_mode(bar_scanner):
184     print("\n")
185     print("\n-----")
186     print("\n1) Manual read mode (default)")
187     print("\n2) Continuous read mode")
188     print("\n3) Motion sensor mode")
189     print("\n-----")
190
191     val = input("\nSelect an option number: ")
192
193     if val == '1':
194         print("\nManual trigger mode enabled")
195         bar_scanner.enable_manual_trigger()
196     elif val == '2':
197         print("\nContinuous read mode enabled")
198         bar_scanner.enable_continuous_read(1)
199     elif val == '3':
200         print("\nMotion trigger mode enabled")
201         bar_scanner.enable_motion_sense()
202     else:
203         print("\nCommand not recognized")
204
205 def symbologies(bar_scanner):
206     print("\n")
207     print("\n-----")
208     print("\n1) Enable all 1D symbologies")
209     print("\n2) Disable all 1D symbologies")
210     print("\n3) Enable all 2D symbologies")
211     print("\n4) Disable all 2D symbologies")
212     print("\n-----")
213
214     val = input("\nSelect an option number: ")
215
216     if val == '1':
217         print("\n1D symbologies enabled")
218         bar_scanner.enable_all_1D()
219     elif val == '2':
220         print("\n1D symbologies disabled")
221         bar_scanner.disable_all_1D()
222     elif val == '3':
223         print("\n2D symbologies enabled")
224         bar_scanner.enable_all_2D()
225     elif val == '4':
226         print("\n2D symbologies disabled")

```

(continues on next page)

(continued from previous page)

```

227     bar_scanner.disable_all_2D()
228 else:
229     print("\nCommand not recognized")
230
231 def run_example():
232
233     print("\nSparkFun DE2120 Barcode Scanner Breakout Example 2")
234     my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()
235
236     if my_scanner.begin() == False:
237         print("\nThe Barcode Scanner module isn't connected correctly to the system.↵
↵Please check wiring", \
238             file=sys.stderr)
239         return
240     print("\nScanner ready!")
241
242     while True:
243
244         print("\n")
245         print("\nSparkFun DE2120 Barcode Scanner Python Package")
246         print("\n-----")
247         print("\n1) Start Scan")
248         print("\n2) Stop Scan")
249         print("\n3) Enable/Disable Flashlight")
250         print("\n4) Enable/Disable Aiming Reticle")
251         print("\n5) Enable/Disable Decode Beep")
252         print("\n6) Enable/Disable Start Up Beep")
253         print("\n7) Change Buzzer Frequency")
254         print("\n8) Enable/Disable Image Flipping")
255         print("\n9) Set Reading Area")
256         print("\n10) Set Reading Mode")
257         print("\n11) Enable/Disable Symbolologies")
258         print("\n-----")
259
260         val = input("\nSelect an option number: ")
261
262         if val == '1':
263             my_scanner.start_scan()
264         elif val == '2':
265             my_scanner.stop_scan()
266         elif val == '3':
267             flash_light(my_scanner)
268         elif val == '4':
269             reticle(my_scanner)
270         elif val == '5':
271             decode_beep(my_scanner)
272         elif val == '6':
273             boot_beep(my_scanner)
274         elif val == '7':
275             change_buzz_freq(my_scanner)
276         elif val == '8':
277             image_flip(my_scanner)

```

(continues on next page)

(continued from previous page)

```

278     elif val == '9':
279         reading_area(my_scanner)
280     elif val == '10':
281         reading_mode(my_scanner)
282     elif val == '11':
283         symbologies(my_scanner)
284     else:
285         print("\nCommand not recognized")
286
287 if __name__ == '__main__':
288     try:
289         run_example()
290     except (KeyboardInterrupt, SystemExit) as exErr:
291         print("\nEnding Example 2")
292         sys.exit(0)

```

## 7.4 Example 3 - Send Command

Listing 3: examples/de2120\_ex3\_send\_command.py

```

1  #!/usr/bin/env python
2  #-----
3  # de2120_ex3_send_command.py
4  #-----
5  #
6  # Written by Priyanka Makin @ SparkFun Electronics, April 2021
7  #
8  # This example demonstrates how to use the "send_command()" method to send
9  # arbitrary serial commands to the barcode reader. It also demonstrates the "CIDENA"
10 # or "Code ID Enable" function, which includes the barcode type when transmitting the
11 # decoded string.
12 #
13 # send_command() takes two strings as arguments, concatenate them, adds the command
14 # prefix "^ ^" and the command suffix "." and then transmits the command to the module.
15 # For example, to enable matrix 2 of 5 scanning, which is done using the command
16 # "^ ^M25ENA1." you would make the call "my_scanner.send_command("M25ENA", 1)"
17 #
18 # While it is valid to call "my_scanner.send_command("M25ENA1")", the former method
19 # is preferred in many cases.
20 #
21 # NOTE: you must put the module into COM mode by scanning the PORVIC barcode
22 # in the datasheet. This will put the module in the correct mode to receive
23 # and transmit serial.
24 #
25 # This package has been developed on a Raspberry Pi 4. Connect the DE2120 Barcode
26 # Scanner Breakout directly to your Pi using a USB-C cable
27 #
28 # Do you like this library? Help support SparkFun. Buy a board!
29 #
30 #=====

```

(continues on next page)

(continued from previous page)

```

31 # Copyright (c) 2021 SparkFun Electronics
32 #
33 # Permission is hereby granted, free of charge, to any person obtaining a copy
34 # of this software and associated documentation files (the "Software"), to deal
35 # in the Software without restriction, including without limitation the rights
36 # to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
37 # copies of the Software, and to permit persons to whom the Software is
38 # furnished to do so, subject to the following conditions:
39 #
40 # The above copyright notice and this permission notice shall be included in all
41 # copies or substantial portions of the Software.
42 #
43 # THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
44 # IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
45 # FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
46 # AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
47 # LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
48 # OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
49 # SOFTWARE.
50 #=====
51 # Example 3
52
53 from __future__ import print_function
54 import de2120_barcode_scanner
55 import time
56 import sys
57
58 def run_example():
59
60     print("\nSparkFun DE2120 Barcode Scanner Breakout Example 3")
61     my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()
62
63     if my_scanner.begin() == False:
64         print("\nThe Barcode Scanner module isn't connected correctly to the system.↵
↵Please check wiring", \
65             file=sys.stderr)
66         return
67     print("\nScanner ready!")
68
69     print("\n")
70     print("\nTransmit Code ID with Barcode? (y/n)")
71     print("\n-----")
72
73     val = input("\nType 'y' or 'n' or scan a barcode: ")
74
75     if val == 'y':
76         print("\nCode ID will be displayed on scan")
77         my_scanner.send_command("CIDENA", "1")
78     elif val == 'n':
79         print("\nCode ID will NOT be displayed on scan")
80         my_scanner.send_command("CIDENA", "0")
81     else:

```

(continues on next page)

(continued from previous page)

```
82     print("\nCommand not recognized")
83
84     scan_buffer = ""
85
86     while True:
87         scan_buffer = my_scanner.read_barcode()
88         if scan_buffer:
89             print("\nCode found: ")
90             print("\n" + str(scan_buffer))
91             scan_buffer = ""
92
93             time.sleep(0.02)
94
95
96 if __name__ == '__main__':
97     try:
98         run_example()
99     except (KeyboardInterrupt, SystemExit) as exErr:
100         print("\nEnding Example 3")
101         sys.exit(0)
```



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### d

de2120\_barcode\_scanner, [15](#)



## INDEX

### A

`available()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 15

### B

`begin()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 15

### C

`change_baud_rate()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 15

`change_buzzer_tone()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`change_reading_area()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`enable_boot_beep()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

`enable_continuous_read()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

`enable_decode_beep()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

`enable_image_flipping()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

`enable_manual_trigger()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

`enable_motion_sense()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

### D

`de2120_barcode_scanner` module, 15

`DE2120BarcodeScanner` (class in *de2120\_barcode\_scanner*), 15

`disable_all_1D()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`disable_all_2D()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`disable_boot_beep()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`disable_decode_beep()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`disable_image_flipping()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

### F

`factory_default()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

`is_connected()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 17

### L

`light_off()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 18

`light_on()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 18

### M

module `de2120_barcode_scanner`, 15

### R

`read()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 18

`read_barcode()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 18

### E

`enable_all_1D()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`enable_all_2D()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner* method), 16

`reticle_off()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner*  
*method*), [18](#)  
`reticle_on()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner*  
*method*), [18](#)

## S

`send_command()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner*  
*method*), [18](#)  
`start_scan()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner*  
*method*), [18](#)  
`stop_scan()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner*  
*method*), [18](#)

## U

`USB_mode()` (*de2120\_barcode\_scanner.DE2120BarcodeScanner*  
*method*), [15](#)