AdafruitminiQR Library Documentation

Release 1.0

ladyada

Jul 09, 2020

Contents

A non-hardware dependant miniature QR generator library. All native Python!

CHAPTER 1

Dependencies

This driver depends on:

- Adafruit CircuitPython

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle.

CHAPTER 2

Installing from PyPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally from PyPI. To install for current user:

```
pip3 install adafruit-circuitpython-miniqr
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-circuitpython-miniqr
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name
python3 -m venv .env
source .env/bin/activate
pip3 install adafruit-circuitpython-miniqr
```

CHAPTER 3

Usage Example

```
import adafruit_miniqr

qr = adafruit_miniqr.QRCode()
qr.add_data(b'https://www.adafruit.com')
qr.make()
print(qr.matrix)
```

CHAPTER 4

Contributing

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER 5

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 6

Table of Contents

6.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/miniqr_simpletest.py

```
import sys
import adafruit_miniqr

# For drawing filled rectangles to the console:
out = sys.stdout
WHITE = "\x1b[1;47m  \x1b[40m"
BLACK = "  "


def prettyprint_QR(matrix):
    # white 4-pixel border at top
    for _ in range(4):
        for _ in range(matrix.width + 8):
            out.write(WHITE)
        print()
    for y in range(matrix.height):
        out.write(WHITE * 4)  # 4-pixel border to left
        for x in range(matrix.width):
            if matrix[x, y]:
                out.write(BLACK)
            else:
                out.write(WHITE)
        out.write(WHITE * 4)  # 4-pixel bporder to right
        print()
    # white 4-pixel border at bottom
    for _ in range(4):
        for _ in range(matrix.width + 8):
```

(continues on next page)

(continued from previous page)

```
            out.write(WHITE)
        print()


qr = adafruit_miniqr.QRCode(qr_type=3, error_correct=adafruit_miniqr.L)
qr.add_data(b"https://www.adafruit.com")
qr.make()
print(qr.matrix)
prettyprint_QR(qr.matrix)
```

6.2 adafruit_miniqr

A non-hardware dependant miniature QR generator library. All native Python!

- Author(s): ladyada

6.2.1 Implementation Notes

Hardware:

- Any!

Software and Dependencies:

- Python 3

class adafruit_miniqr.**QRBitBuffer**
Storage class for a length of individual bits

get(*index*)
The bit value at a location

get_length_bits()
Size of bit buffer

put(*num*, *length*)
Add a number of bits from a single integer value

put_bit(*bit*)
Insert one bit at the end of the bit buffer

class adafruit_miniqr.**QRBitMatrix**(*width*, *height*)
A bit-packed storage class for matrices

class adafruit_miniqr.**QRCode**(*, *qr_type=None*, *error_correct=1*)
The generator class for QR code matrices

add_data(*data*)
Add more data to the QR code, must be bytestring stype

make(*, *test=False*, *mask_pattern=0*)
Perform the actual generation of the QR matrix. To keep things small and speedy we don't generate all 8 mask patterns and pick the best. Instead, please pass in a desired mask_pattern, the default mask is 0.

class adafruit_miniqr.**QRPolynomial**(*num*, *shift*)
Structure for creating and manipulating error code polynomials

get(*index*)
The exponent at the index location

get_length()
Length of the poly

multiply(*e*)
Multiply two polynomials, returns a new one

class adafruit_miniqr.**QRUtil**
A selection of bit manipulation tools for QR generation and BCH encoding

static get_BCH_digit(*data*)
Count digits in data

static get_BCH_type_info(*data*)
Encode with G15 BCH mask

static get_BCH_type_number(*data*)
Encode with G18 BCH mask

static get_error_correct_polynomial(*ecc_length*)
Generate a ecc polynomial

static get_mask(*mask*, *i*, *j*)
Perform matching calculation on two vals for given pattern mask

static get_pattern_position(*qr_type*)
The mask pattern position array for this QR type

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index