sparkfun$_{q}wiic_{d}ual_{e}ncoder_{r}eader$

Release 0.0.1

Apr 09, 2020

Contents:

Python module for the qwiic dual encoder reader (ATTINY84), which is included on the SparkFun Auto pHAT for Raspberry Pi

This python package enables the user to take count readings from the on-board ATTINY84 that handles reading the dual motor encoders. The firmware that is used on the ATTiny84 is located in a separate repository here: SparkFun Dual Encoder Reader Firmware Repository

This package can be used in conjunction with the overall SparkFun qwiic Python Package

New to qwiic? Take a look at the entire SparkFun qwiic ecosystem.

CHAPTER 1

Contents

CHAPTER 2

Supported Platforms

The qwiic ICM20948 Python package current supports the following platforms:

- Raspberry Pi
- NVidia Jetson Nano
- Google Coral Development Board

CHAPTER 3

Dependencies

This driver package depends on the qwiic I2C driver: Qwiic_I2C_Py

CHAPTER 4

Documentation

The SparkFun qwiic Dual Encoder Reader documentation is hosted at ReadTheDocs

CHAPTER 5

Installation

5.1 PyPi Installation

This repository is hosted on PyPi as the sparkfun-qwiic-dual-encoder-reader 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 sparkfun-qwiic-dual-encoder-reader
```

For the current user:

```
pip install sparkfun-qwiic-dual-encoder-reader
```

5.2 Local Installation

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 sparkfun_qwiic_dual_encoder_reader-<version>.tar.gz
```

CHAPTER 6

Example Use

See the examples directory for more detailed use examples.

```
from __future__ import print_function
import qwiic_dual_encoder_reader
import time
import sys

def runExample():

    print("\nSparkFun Qwiic Dual Encoder Reader   Example 1\n")
    myEncoders = qwiic_dual_encoder_reader.QwiicDualEncoderReader()

    if myEncoders.connected == False:
        print("The Qwiic Dual Encoder Reader device isn't connected to the system.␣
↪Please check your connection", \
            file=sys.stderr)
        return

    myEncoders.begin()

    while True:

        print("Count1: %d, Count2: %s" % (myEncoders.count1, \
            myEncoders.count2, \
            ))

        time.sleep(.3)

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

CHAPTER 7

Table of Contents

7.1 API Reference

7.1.1 qwiic_dual_encoder_reader

Python module for the[SparkFun Auto pHAT for Raspberry Pi](https://www.sparkfun.com/products/16328)

This python package enables the user to take count readings from the on-board ATTINY84 that handles reading the dual motor encoders.

The firmware that is used on the ATTiny84 is located in a separate repository here: [SparkFun Dual Encoder Reader Firmware Repository](https://github.com/sparkfun/Qwiic_Dual_Encoder_Reader)

This package can be used in conjunction with the overall [SparkFun qwiic Python Package](https://github.com/sparkfun/Qwiic_Py)

New to qwiic? Take a look at the entire [SparkFun qwiic ecosystem](https://www.sparkfun.com/qwiic).

class qwiic_dual_encoder_reader.**QwiicDualEncoderReader**(*address=None*, *i2c_driver=None*)

Parameters

- **address** – The I2C address to use for the device. If not provided, the default address is used.
- **i2c_driver** – An existing i2c driver object. If not provided a driver object is created.

Returns The QwiicDualEncoderReader device object.

Return type Object

begin()

Initialize the operation of the Dual Encoder Reader module

Returns Returns true of the initializtion was successful, otherwise False.

Return type bool

clear_interrupts()
Clears the moved bit

Returns No return Value

connected
Determine if a device is conntected to the system..

Returns True if the device is connected, otherwise False.

Return type bool

count1
Returns the number of "ticks" the encoder1 has turned

Returns number of encoder pulses

Return type word as integer

count2
Returns the number of "ticks" the encoder2 has turned

Returns number of encoder pulses

Return type word as integer

get_count1()
Returns the number of "ticks" the encoder1 has turned

Returns number of encoder pulses

Return type word as integer

get_count2()
Returns the number of "ticks" the encoder2 has turned

Returns number of encoder pulses

Return type word as integer

get_diff(*clear_value=False*)
Returns the number of ticks since last check

Parameters **clearValue** – Set to True to clear the current value. Default is False

Returns the difference

Return type integer

get_int_timeout()
Get number of milliseconds that elapse between end of encoder turning and interrupt firing

Returns the timeout value

Return type integer

get_limit()
Returns the limit of allowed counts before wrapping. 0 is disabled

Returns The limit

Return type integer

get_version()
Returns a integer of the firmware version number

Returns The firmware version

Return type integer

has_moved()
Returns true if encoder has moved

Returns Moved state

Return type Boolean

int_timeout
Get number of milliseconds that elapse between end of encoder turning and interrupt firing

Returns the timeout value

Return type integer

is_connected()
Determine if a device is conntected to the system..

Returns True if the device is connected, otherwise False.

Return type bool

limit
Returns the limit of allowed counts before wrapping. 0 is disabled

Returns The limit

Return type integer

moved
Returns true if encoder has moved

Returns Moved state

Return type Boolean

set_count1(*amount*)
Set the encoder count1 to a specific amount

Parameters **amount** – the value to set the counter to

Returns no return value

set_count2(*amount*)
Set the encoder count2 to a specific amount

Parameters **amount** – the value to set the counter to

Returns no return value

set_int_timeout(*timeout*)
Set number of milliseconds that elapse between end of encoder turning and interrupt firing

Parameters **timeout** – the timeout value in milliseconds

Returns No return value

set_limit(*amount*)
Set the encoder count limit to a specific amount

Parameters **amount** – the value to set the limit to

Returns no return value

since_last_movement(*clear_value=True*)
Returns the number of milliseconds since the last encoder movement By default, clear the current value

Parameters **clearValue** – Clear out the value? True by default

Returns time since last encoder movement

Return type integer

version

Returns a integer of the firmware version number

Returns The firmware version

Return type integer

7.2 Basic Operation

Listing 1: examples/ex1_qwiic_dual_encoder_reader.py

```
#!/usr/bin/env python
#-------------------------------------------------------------------------
# ex1_qwiic_dual_encoder_reader.py
#
# Simple Example demonstrating how to read encoder counts for the Qwiic Dual Encoder␣
↪Reader (as part of the SparkFun Auto pHAT)
#-------------------------------------------------------------------------
#
# Written by  SparkFun Electronics, May 2019
#
# This python library supports the SparkFun Electroncis qwiic
# qwiic sensor/board ecosystem on a Raspberry Pi (and compatable) single
# board computers.
#
# More information on qwiic is at https://www.sparkfun.com/qwiic
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2019 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 1
```

(continues on next page)

(continued from previous page)

```
#

from __future__ import print_function
import qwiic_dual_encoder_reader
import time
import sys

def runExample():

        print("\nSparkFun Qwiic Dual Encoder Reader   Example 1\n")
        myEncoders = qwiic_dual_encoder_reader.QwiicDualEncoderReader()

        if myEncoders.connected == False:
                print("The Qwiic Dual Encoder Reader device isn't connected to the system. Please check your connection", \
                        file=sys.stderr)
                return

        myEncoders.begin()

        while True:

                print("Count1: %d, Count2: %s" % (myEncoders.count1, \
                        myEncoders.count2, \
                        ))

                time.sleep(.3)

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


```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

q

Index