sparkfun$_{q}wiic_{s}cmd$

Release 0.9.0

Apr 20, 2020

Contents:

Python module for the qwiic serial control motor driver

This python package is a port of the existing SparkFun Serial Controlled Motor Driver Arduino Library

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 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 SCMD module documentation is hosted at ReadTheDocs

CHAPTER 5

Installation

5.1 PyPi Installation

This repository is hosted on PyPi as the sparkfun-qwiic-scmd 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-scmd
```

For the current user:

```
pip install sparkfun-qwiic-scmd
```

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_scmd-<version>.tar.gz

Example Use
```

```
TBD
```

CHAPTER 6

Table of Contents

6.1 API Reference

6.1.1 qwiic_scmd

Python module for the serial control motor driver.

This python package is a port of the existing [SparkFun Serial Controlled Motor Driver Arduino Library](https://github.com/sparkfun/SparkFun_Serial_Controlled_Motor_Driver_Arduino_Library)

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_scmd.**QwiicScmd**(*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 Serial Control Motor Driver device object.

Return type Object

begin()

Initialize the operation of the SCMD module

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

Return type bool

bridging_mode(*driverNum*, *bridged*)

Configure a driver's bridging state

Parameters

- **driverNum** – Number of driver. Master is 0, slave 1 is 1, etc. 0 to 16
- **bridged** – 0 or 1 for forward and backward

Returns No return value

busy()
Returns if the driver is busy

Returns busy status

Return type boolean

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

Returns True if the device is connected, otherwise False.

Return type bool

disable()
Disable driver functions

Returns No return value

enable()
Enable driver functions

Returns No return value

get_diagnostics()
Get diagnostic information from the masterd

Returns Object returned with properties that are the diagnostic info

Return type Object - SCMDDiagnostics()

get_remote_diagnostics(*address*)
Get diagnostic information from a slave

Parameters **address** – Address of slave to read. Can be 0x50 to 0x5F for slave 1 to 16.

Returns Object returned with properties that are the diagnostic info

Return type Object - SCMDDiagnostics()

inversion_mode(*motorNum*, *polarity*)
Configure a motor's direction inversion

Parameters

- **motoNum** – Motor number from 0 to 33
- **polarity** – 0 or 1 for default or inverted

Returns No return value

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

Returns True if the device is connected, otherwise False.

Return type bool

read_remote_register(*address*, *offset*)
Read data from a slave. Note that this waits 5ms for slave data to be aquired before making the final read.

Parameters

- **address** – Address of slave to read. Can be 0x50 to 0x5F for slave 1 to 16.
- **offset** – Address of data to read. Can be 0x00 to 0x7F

Returns Register Value

Return type integer

ready()
Returns if the driver is ready

Returns Ready status

Return type boolean

reset()
This is a hack in the Arduino lib - the placeholder is for compatablity

reset_diagnostic_counts()
Reset the master's diagnostic counters

Returns No return value

reset_remote_diagnostic_counts(*address*)
Reset a slave's diagnostic counters

Parameters **address** – Address of slave to read. Can be 0x50 to 0x5F for slave 1 to 16.

Returns No return value

set_drive(*motorNum*, *direction*, *level*)
Drive a motor at a level

Parameters

- **motoNum** – Motor number from 0 to 33
- **direction** – 0 or 1 for forward and backward
- **level** – (-255) to 255 for drive strength

Returns No return value

write_remote_register(*address*, *offset*, *dataToWrite*)
Write data from a slave

Parameters

- **address** – Address of slave to read. Can be 0x50 to 0x5F for slave 1 to 16.
- **offset** – Address of data to read. Can be 0x00 to 0x7F
- **dataToWrite** – The data to write

Returns No return value

class qwiic_scmd.**SCMDDiagnostics**
Object used for diagnostic reporting.

Variables

- **numberOfSlaves** –
- **U_I2C_RD_ERR** –
- **U_I2C_WR_ERR** –
- **U_BUF_DUMPED** –

- **E_I2C_RD_ERR** –
- **E_I2C_WR_ERR** –
- **LOOP_TIME** –
- **SLV_POLL_CNT** –
- **MST_E_ERR** –
- **MST_E_STATUS** –
- **FSAFE_FAULTS** –
- **REG_OOR_CNT** –
- **REG_RO_WRITE_CNT** –

6.2 Example 1: Single Motor Test

Listing 1: examples/ex1_qwiic_scmd.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# A simple test to speed up and slow down 1 motor.
#-----------------------------------------------------------------------
#
# Written by Mark Lindemer
# SparkFun Electronics, April 2020
#
# 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.
```

(continues on next page)

(continued from previous page)

```
#===============================================================================
# Example 1
#

from __future__ import print_function
import time
import sys
import math
import qwiic_scmd

myMotor = qwiic_scmd.QwiicScmd()

def runExample():
        print("Motor Test.")
        R_MTR = 0
        L_MTR = 1
        FWD = 0
        BWD = 1

        if myMotor.connected == False:
                print("Motor Driver not connected. Check connections.", \
                        file=sys.stderr)
                return
        myMotor.begin()
        print("Motor initialized.")
        time.sleep(.250)

        # Zero Motor Speeds
        myMotor.set_drive(0,0,0)
        myMotor.set_drive(1,0,0)

        myMotor.enable()
        print("Motor enabled")
        time.sleep(.250)


        while True:
                speed = 20
                for speed in range(20,255):
                        print(speed)
                        myMotor.set_drive(R_MTR,FWD,speed)
                        time.sleep(.05)
                for speed in range(254,20, -1):
                        print(speed)
                        myMotor.set_drive(R_MTR,FWD,speed)
                        time.sleep(.05)

if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("Ending example.")
                myMotor.disable()
                sys.exit(0)
```

6.3 Example 2: Dual Motor Test

Listing 2: examples/ex2_qwiic_scmd.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# A simple test to speed up and slow down both motors in opposite directions.
#-----------------------------------------------------------------------
#
# Written by Mark Lindemer
# SparkFun Electronics, April 2020
#
# 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
#

from __future__ import print_function
import time
import sys
import math
import qwiic_scmd

myMotor = qwiic_scmd.QwiicScmd()

def runExample():
        print("Motor Test.")
        R_MTR = 0
        L_MTR = 1
        FWD = 0
```

(continues on next page)

(continued from previous page)

```
        BWD = 1

        if myMotor.connected == False:
                print("Motor Driver not connected. Check connections.", \
                        file=sys.stderr)
                return
        myMotor.begin()
        print("Motor initialized.")
        time.sleep(.250)

        # Zero Motor Speeds
        myMotor.set_drive(0,0,0)
        myMotor.set_drive(1,0,0)

        myMotor.enable()
        print("Motor enabled")
        time.sleep(.250)


        while True:
                speed = 20
                for speed in range(20,255):
                        print(speed)
                        myMotor.set_drive(R_MTR,FWD,speed)
                        myMotor.set_drive(L_MTR,BWD,speed)
                        time.sleep(.05)
                for speed in range(254,20, -1):
                        print(speed)
                        myMotor.set_drive(R_MTR,FWD,speed)
                        myMotor.set_drive(L_MTR,BWD,speed)
                        time.sleep(.05)

if __name__ == '__main__':
        try:
                runExample()
        except (KeyboardInterrupt, SystemExit) as exErr:
                print("Ending example.")
                myMotor.disable()
                sys.exit(0)
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

q

Index