Fusion Python library Documentation

Fusion - RaspberryPi python module for use with Fusion system

Copyright (c) 2016 Modern Robotics Inc.

Author: Justin Mathews

Version: 1.0

Fusion Web Page: http://www.modernroboticsinc.com

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.

# Fusion.driver()

#### Definition:

The following class is used to provide base functionality of the Fusion control board and provide drivers to interface with the various array of sensors and output for servo and motor control.

#### Parameters:

**None**

#### Returns:

**Driver object**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

## analogRead(*port)*

#### Definition:

Used to read analog devices connected to ports A0-A7

#### Parameters:

***port*** : driver.A0-A7

#### Returns:

**int (0-1023)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

**print** f.analogRead**(**f.A0**)** # Print A0 output to console

## digitalRead(*port)*

#### Definition:

Used to read digital devices connected to ports D0-D7

#### Parameters:

***port*** : driver.D0-D7

#### Returns:

**int (0-1)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

**print** f.digitalRead**(**f.D0**)** # Print D0 output to console

## digitalState(*port, state*)

#### Definition:

Sets the state of the selected digital port D0-D7 as input or output

#### Parameters:

***port*** : driver.D0-D7  
***state***: driver.INPUT / driver.OUTPUT

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

f.digitalState**(**f.D0**,** f.OUTPUT**)** # Set D0 to output

## digitalWrite(*port, state*)

#### Definition:

Sets the value of the selected digital port D0-D7 when in output mode

#### Parameters:

***port*** : driver.D0-D7  
***state***: int(1 or 0)

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

f.digitalState**(**f.D0**,** f.OUTPUT**)** # Set D0 to output

f.digitalWrite**(**f.D0**,** 1**)** # Set D0 to high

## servoEnable(*servo, state*)

#### Definition:

Enable or disable pwm output of the selected servo port

#### Parameters:

***servo*** : driver.S0-S3  
***state***: int(1 or 0)

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

f.servoEnable**(**f.S0**,** 1**)** # Enable PWM on S0

## servoTarget(*servo, target*)

#### Definition:

Sets the target position of the servo on the selected port

#### Parameters:

***servo*** : driver.S0-S3  
***target***: int(0-255)

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

f.servoEnable**(**f.S0**,** 1**)** # Enable PWM on S0

f.servoTarget**(**f.S0**,** 128**)** # Set servo position to center value 128

## motorMode(*motor, mode*)

#### Definition:

Set the stop mode of the motor on the selected port. Default value is FLOAT.

#### Parameters:

***motor*** : driver.M0-M1  
***mode***: driver.FLOAT/BRAKE

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

f.motorMode**(**f.M0**,** f.BRAKE**)** # Set M0 to brake when stopped

## motorSpeed(*motor, speed*)

#### Definition:

Set the speed of the motor on the selected port

#### Parameters:

***motor*** : driver.M0-M1  
***speed***: int(±100)

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

f.motorSpeed**(**f.M0**,** 100**)** # Set the speed of M0 to 100

f.motorSpeed**(**f.M1**,** **-**100**)** # Set the speed of M1 to -100

## i2cRead(*addr, reg, len*)

#### Definition:

Read up to 32 bytes from a device on the I2C buf.

#### Parameters:

***addr*** : Device I2C address  
***reg***: First register to read from  
**len**: Number of registers to read

#### Returns:

**buf [reg1, reg2, .....]  
 global: i2c\_error** Global value can be read or cleared at any time, displays errors detected during last I2C Transaction

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

**print** f.i2cRead**(**0x20**,** 0x00**,** 3**)** # Read 3 bytes starting at register 0x00 from device 0x20

## i2cWrite(*addr, reg, buf[]*)

#### Definition:

Write a buffer of up 32 bytes to a device on the I2C buf. The length is determined based on the length of the buffer being written.

#### Parameters:

***addr*** : Device I2C address  
***reg***: First register to read from  
**buf[]**: Buffer of up to 32 bytes

#### Returns:

**None   
 global: i2c\_error** Global value can be read or cleared at any time, displays errors detected during last I2C Transaction

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

buf **=** **[**1**,** 2**]** # Build a buffer with two bytes

f.i2cWrite**(**0x20**,** 0x00**,** buf**)** # Write 2 byte buffer starting at 0x00 to 0x20

f.i2cWrite**(**0x20**,** 0x02**,** **[**3**,** 4**])** # Buffer values can also be directly inserted

## readBattRaw()

#### Definition:

Read the raw value of the battery voltage from the onboard ADC converter

#### Parameters:

**None**

#### Returns:

**int(0-1023)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

**print** f.readBattRaw**()** # Return the raw ADC value of the battery voltage

## readBatt()

#### Definition:

Read the scaled value of the battery voltage from the onboard ADC converter

#### Parameters:

**None**

#### Returns:

**float(voltage)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

**print** f.readBatt**()** # Return the scaled ADC value of the battery voltage

## setLED(*led, value*)

#### Definition:

Set the selected onboard LED to ON or OFF

#### Parameters:

**led**: driver.BLUE / YELLOW  
 **value:** int(1 or 0)

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

f.setLED**(**f.YELLOW**,** 1**)** # Turn the onboard yellow LED on

## Data and Attributes:

#### Servo ports:

**S0** = 0x01

**S1** = 0x02

**S2** = 0x04

**S3** = 0x08

#### Servo attributes:

**ENABLE** = 0x01

**DISABLE** = 0x00

#### Motor ports:

**M0** = 0x01

**M1** = 0x02

#### Motor attributes:

**FLOAT** = 0x00

**BRAKE** = 0x01

#### Digital ports:

**D0** = 0x0001

**D1** = 0x0002

**D2** = 0x0004

**D3** = 0x0008

**D4** = 0x0010

**D5** = 0x0020

**D6** = 0x0040

**D7** = 0x0080

#### Digital attributes:

**INPUT** = 0x00

**OUTPUT** = 0x01

#### Analog ports:

**A0** = 0x0101

**A1** = 0x0102

**A2** = 0x0104

**A3** = 0x0108

**A4** = 0x0110

**A5** = 0x0120

**A6** = 0x0140

**A7** = 0x0180

#### LED attributes:

**YELLOW** = 0x00

**BLUE** = 0x01

#### I2C attributes:

**i2cError** = None

# Fusion.analog(*driver, port*)

#### Definition:

The following class provides a wrapper for the analog function to tie sensor names directly to the port and read all in one simple motion.

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **port:** Analog port to attach object

#### Returns:

**Analog object**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

ods1 **=** Fusion**.**analog**(**f**,** f.A0**)** # Create an instance of analog called ods1 on port A0

## read()

#### Definition:

Returns the current analog value of the currently configured port.

#### Parameters:

**None**

#### Returns:

**int(0-1023)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

ods1 **=** Fusion**.**analog**(**f**,** f.A0**)** # Create an instance of analog called ods1 on port A0

**print** ods1**.**read**()** # Print the current analog value of the sensor

## Data and Attributes:

**None**

# Fusion.color(*driver, addr=0x3C*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics Color sensor. Please refer to the color sensor documentation for further information on the use and calibration of the sensor .

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of color sensor (Default value = 0x3C)

#### Returns:

**Color sensor object**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

color1 **=** Fusion**.**color**(**f**)** # Color class instance with the default address 0x3C

color2 **=** Fusion**.**color**(**f**,** 0x40**)** # Color class instance with the custom address 0x40

## colorSetup(*mode, rate=SIXTY\_HZ*)

#### Definition:

Function to set the mode and sampling rate of the color sensor.

#### Parameters:

**mode:** color.ACTIVE/PASSIVE  
 **rate:** color.SIXTY\_HZ/FIFTY\_HZ

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

c **=** Fusion**.**color**(**f**)** # Instance of the color class

c**.**colorSetup**(**c**.**ACTIVE**)** # Color mode is Active with default 60Hz

c**.**colorSetup**(**c**.**PASSIVE**,** c**.**FIFTY\_HZ**)** # Color mode is Passive sampling at 50Hz

## whiteBalance()

#### Definition:

Calibration function to set the white balance of the sensor. Refer to sensor documentation for proper calibration proceduref.

#### Parameters:

**None**

#### Returns:

**None**

## blackBalance()

#### Definition:

Calibration function to set the black balance of the sensor. Refer to sensor documentation for proper calibration proceduref.

#### Parameters:

**None**

#### Returns:

**None**

## getColorNumber()

#### Definition:

Function to read the current color number detected by the color sensor. Refer to color sensor documentation for corresponding colors and color numberf.

#### Parameters:

**None**

#### Returns:

**int(color number)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

c **=** Fusion**.**color**(**f**)** # Instance of the color class

**print** c**.**getColorNumber**()** # Prints the current color number detected

## getColorValue()

#### Definition:

This function returns the red, green, blue, and white color values as a 4 byte array.

#### Parameters:

**None**

#### Returns:

**int[red, green, blue, white]**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

c **=** Fusion**.**color**(**f**)** # Instance of the color class

# The following returns the array and puts each individual byte in a

# variable to be called at a later time then prints the red value.

**(**red**,** green**,** blue**,** white**)** **=** c**.**getColorValue**()**

**print** red

# The following uses indexing to return one byte directly the print

# function. Example: index [0] refers to the red value

**print** c**.**getColorValue**()[**0**]**

## getColorIndex()

#### Definition:

Function to read the current color index detected by the color sensor. Refer to color sensor documentation for corresponding colors and color index.

#### Parameters:

**None**

#### Returns:

**int(color index)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

c **=** Fusion**.**color**(**f**)** # Instance of the color class

**print** c**.**getColorIndex**()** # Prints the current color index detected

## getRGBIndex()

#### Definition:

Function to read the current RGB index detected by the color sensor. Refer to color sensor documentation for corresponding colors and RGB Indexef.

#### Parameters:

**None**

#### Returns:

**int(RGB index)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

c **=** Fusion**.**color**(**f**)** # Instance of the color class

**print** c**.**getRGBIndex**()** # Prints the current RGB index detected

## getColorReading()

#### Definition:

The following function returns the current 16-bit red, green, blue, and white color readings detected from the sensor.

#### Parameters:

**None**

#### Returns:

**int[red, green, blue, white]**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

c **=** Fusion**.**color**(**f**)** # Instance of the color class

# The following returns the array and puts each individual byte in a

# variable to be called at a later time then prints the red value.

**(**red**,** green**,** blue**,** white**)** **=** c**.**getColorReading**()**

**print** red

# The following uses indexing to return one byte directly the print

# function. Example: index [0] refers to the red value

**print** c**.**getColorReading**()[**0**]**

## getColorNormalized()

#### Definition:

The following function returns the current normalized 16-bit red, green, blue, and white color readings detected from the sensor.

#### Parameters:

**None**

#### Returns:

**int[red, green, blue, white]**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

c **=** Fusion**.**color**(**f**)** # Instance of the color class

# The following returns the array and puts each individual byte in a

# variable to be called at a later time then prints the red value.

**(**red**,** green**,** blue**,** white**)** **=** c**.**getColorNormalized**()**

**print** red

# The following uses indexing to return one byte directly the print

# function. Example: index [0] refers to the red value

**print** c**.**getColorNormalized**()[**0**]**

## Data and Attributes:

#### Color modes:

**ACTIVE** = 0x00

**PASSIVE** = 0x01

#### Sampling Frequencies:

**FIFTY\_HZ** = 0x35

**SIXTY\_HZ** = 0x36

# Fusion.colorBeacon(*driver, addr=0x84*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics Color Beacon tricolor led indicator device

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of color sensor (Default value = 0x84)

#### Returns:

**Color Beacon object**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

cb1 **=** Fusion**.**colorBeacon**(**f**)** # Color class instance with the default address 0x84

cb2 **=** Fusion**.**colorBeacon**(**f**,** 0x40**)** # Color class instance with the custom address 0x40

## setColor(*color*)

#### Definition:

Function for setting one of the primary colors for the color beacon. Colors are oriented in a 3-bit fashion and hence can be a value between 0 and 7. The bit format is as follows (blue, green, red) so a value of 3 or 011 would yield a combination of green and red giving yellow.

#### Parameters:

**color:** 0-7

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

cb1 **=** Fusion**.**colorBeacon**(**f**)** # Color class instance

cb1**.**setColor**(**3**)** # Set the color to Yellow

cb1**.**setColor**(**4**)** # Set the color to Blue

cb1**.**setColor**(**7**)** # Set the color to White

## setCustomColor(*red, green, blue*)

#### Definition:

Function for setting a custom 8-bit RGB color value.

#### Parameters:

**red:** int (0-255)  
 **green:** int (0-255)  
 **blue:** int (0-255)

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

cb1 **=** Fusion**.**colorBeacon**(**f**)** # Color class instance

## Data and Attributes:

**None**

# Fusion.compass(*driver, addr=0x24*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics Compass sensor.

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of color sensor (Default value = 0x24)

#### Returns:

**Compass object**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

cmp1 **=** Fusion**.**compass**(**f**)** # Instance of compass with default address

cmp2 **=** Fusion**.**compass**(**f**,** 0x40**)** # Instance of compass with custom address

## hardIronCalibration()

#### Definition:

Calibration function to perform the hard iron calibration of the sensor. Refer to sensor documentation for proper calibration proceduref.

#### Parameters:

**None**

#### Returns:

**None**

## nullAccelerometer(*axis*)

#### Definition:

Nulls the current accelerometer value to 0 at the current location.

#### Parameters:

**axis:** char(X, Y, Z)

#### Returns:

**None**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

cmp1 **=** Fusion**.**compass**(**f**)** # Instance of compass with default address

cmp1**.**nullAccelerometer**(**'X'**)** # Nulls the X axis to 0 at current value

## getAccelerometer()

#### Definition:

Gets the current accelerometer value for the selected axis as a value of 0-255

#### Parameters:

**None**

#### Returns:

**int[x, y, z]**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

cmp1 **=** Fusion**.**compass**(**f**)** # Instance of compass with default address

# The following returns the array and puts each individual byte in a

# variable to be called at a later time then prints the X value.

**(**X**,** Y**,** Z**)** **=** cmp1**.**getAccelerometer**()**

**print** X

# The following uses indexing to return one byte directly the print

# function. Example: index [0] refers to the X value

**print** cmp1**.**getAccelerometer**()[**0**]**

## tiltUp()

#### Definition:

Calibration function to measure the tilt up value. Refer to sensor documentation for the proper calibration procedures and how to use these functionf.

#### Parameters:

**None**

#### Returns:

**None**

## tiltDown()

#### Definition:

Calibration function to measure the tilt down value. Refer to sensor documentation for the proper calibration procedures and how to use these functionf.

#### Parameters:

**None**

#### Returns:

**None**

## getMagnetometer()

#### Definition:

Gets the current magnetometer value for the selected axis as a value of 0-1023

#### Parameters:

**None**

#### Returns:

**int[x, y, z]**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver class

cmp1 **=** Fusion**.**compass**(**f**)** # Instance of compass with default address

# The following returns the array and puts each individual byte in a

# variable to be called at a later time then prints the X value.

**(**X**,** Y**,** Z**)** **=** cmp1**.**getMagnetometer**()**

**print** X

# The following uses indexing to return one byte directly the print

# function. Example: index [0] refers to the X value

**print** cmp1**.**getMagnetometer**()[**0**]**

## scaleAccelerometer()

#### Definition:

Calibration function to scale the accelerometer for greater accuracy. Refer to sensor documentation for the proper calibration procedures and how to use these functionf.

#### Parameters:

**None**

#### Returns:

**None**

## Data and Attributes:

**None**

## Fusion.digital(*driver, port*)

#### Definition:

The following class provides a wrapper for the digital function to tie sensor names directly to the port and read all in one simple motion.

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **port:** Digital port to attach object

#### Returns:

**Digital object**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

touch1 **=** Fusion**.**digital**(**f**,** f.D0**)** # Create instance named touch1 on port D0

## read()

#### Definition:

Returns the current analog value of the currently configured port.

#### Parameters:

**None**

#### Returns:

**int(0-1023)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

touch1 **=** Fusion**.**digital**(**f**,** f.D0**)** # Create instance named touch1 on port D0

**print** touch1**.**read**()** # Print the current digital value of the sensor

## Data and Attributes:

**None**

# Fusion.joystick(*joy\_num=0*)

#### Definition:

The following class provides a wrapper for the built in python joystick functionality and allows the user to connect and configure a joystick, wired, or wireless gamepad connected to the USB port.

#### Parameters:

**joy\_num:**  Joystick number assigned by system (defaults to 0, always 0 if only one is connected)

#### Returns:

**Joystick Object**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

joy **=** Fusion**.**joystick**()** # Create instance with only one joystick connected

## readAxis(*axis*):

#### Definition:

Read the selected axis as a value of -100 to 100

#### Parameters:

**axis:** Axis number, can be found in system manager or by example program

#### Returns:

**int(± 100)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

joy **=** Fusion**.**joystick**()** # Create instance with only one joystick connected

**print** joy**.**readAxis**(**1**)** # Read axis 1 and print to console

## readAxisFloat(*axis*):

#### Definition:

Read the selected axis as a value of -1 to 1

#### Parameters:

**axis:** Axis number, can be found in system manager or by example program

#### Returns:

**float (± 1)**

#### Example:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

joy **=** Fusion**.**joystick**()** # Create instance with only one joystick connected

**print** joy**.**readAxisFloat**(**1**)** # Read axis 1 and print to console

## mixer (*x\_axis, y\_axis, x\_inv=False, y\_inv=False*):

#### Definition:

Function to mix two axis providing one stick tank style steering and control. Returns a buffer with left and right values -100 to 100 that can be directly placed in the motor speed function for control.

#### Parameters:

**x\_axis:** X-Axis number, can be found in system manager or by example program  
 **y\_axis:** Y-Axis number, can be found in system manager or by example program  
 **x\_inv:** X-Axis invert (default=False)  
 **y\_inv:** Y-Axis invert (default=False)

#### Returns:

**int[left, right]**

#### Example 1:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

joy **=** Fusion**.**joystick**()** # Create instance with only one joystick connected

# Mix axis 1 and 2 for motor control

**(**left**,** right**)** **=** joy**.**mixer**(**1**,** 2**)**

f.motorSpeed**(**f.M0**,** left**)** # Set M0 with left motor speed

f.motorSpeed**(**f.M1**,** right**)** # Set M1 with right motor speed

#### Example 2:

**import** Fusion

f = Fusion**.**driver**()** # Create instance of driver object

joy **=** Fusion**.**joystick**()** # Create instance with only one joystick connected

# Mix axis 1 and 2 and invert X axis

**(**left**,** right**)** **=** joy**.**mixer**(**1**,** 2**,** **False,** **True)**

f.motorSpeed**(**f.M0**,** left**)** # Set M0 with left motor speed

f.motorSpeed**(**f.M1**,** right**)** # Set M1 with right motor speed

## readButton(*button*):

#### Definition:

Read the selected button and return a value on 1 or 0

#### Parameters:

**button:** Button number, can be found in system manager or by example program

#### Returns:

**int(1 or 0)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver object

joy **=** Fusion**.**joystick**()** # Create instance with only one joystick connected

**print** joy**.**readButton**(**1**)** # Read button 1 and print to console

## readHat(*hat*):

#### Definition:

Read the selected hat and return a array of 2 bytes corresponding to X and Y values as -1, 0, or 1

#### Parameters:

**hat:** Hat number, can be found in system manager or by example program

#### Returns:

**int[x ,y]**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver object

joy **=** Fusion**.**joystick**()** # Create instance with only one joystick connected

**print** joy**.**readHat**(**1**)** # Read hat 1 and print to console

## Data and Attributes:

**None**

# Fusion.lineScan(*driver, addr=0x40*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics Line Scan sensor. Please refer to the sensor documentation for further information on the use and calibration of the sensor .

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of color sensor (Default value = 0x3C)

#### Returns:

**Line scan sensor object**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

line1 **=** Fusion**.**lineScan**(**f**)** # LineScan class instance with the default address 0x40

line2 **=** Fusion**.**lineScan**(**f**,** 0x40**)** # LineScan class instance with the custom address 0x42

## setMode(*mode*)

#### Definition:

Function to set the mode of the line scan sensor. Default is black line on white background with auto exposure enabled.

#### Parameters:

**mode:** Mode setting, combine mode settings with & operator

#### Returns:

**None**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

l**.**setMode**(**l**.**BLACK\_LINE **&** l**.**DISABLE\_AUTO**)** # Set mode to black line, no auto exposure

## secondLeftReading()

#### Definition:

Read the center position of the second left reading of the sensor when multiple detections occur. Reads 0 if no detection has occured.

#### Parameters:

**None**

#### Returns:

**int(0-127)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**secondLeftReading**()** # Print the center of the second left detection

## firstLeftReading()

#### Definition:

Read the center position of the first left reading of the sensor when multiple detections occur. Reads 0 if no detection has occured.

#### Parameters:

**None**

#### Returns:

**int(0-127)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**firstLeftReading**()** # Print the center of the first left detection

## centerReading()

#### Definition:

Read the center position of the center most detection of the sensor. Reads 0 if no detection has occured.

#### Parameters:

**None**

#### Returns:

**int(0-127)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**centerReading**()** # Print the center most detection

## firstRightReading()

#### Definition:

Read the center position of the first right reading of the sensor when multiple detections occur. Reads 0 if no detection has occured.

#### Parameters:

**None**

#### Returns:

**int(0-127)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**firstRightReading**()** # Print the center of the first right detection

## secondRightReading()

#### Definition:

Read the center position of the second right reading of the sensor when multiple detections occur. Reads 0 if no detection has occured.

#### Parameters:

**None**

#### Returns:

**int(0-127)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**secondRightReading**()** # Print the center of the second right detection

## currentMaxBrightness()

#### Definition:

Return the current maximum brightness level detected by the sensor

#### Parameters:

**None**

#### Returns:

**int(0-127)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**currentMaxBrightness**()** # Print the max brightness detected by the sensor

## currentFrameTimer()

#### Definition:

Return the current frame timer in units of 128uS.

#### Parameters:

**None**

#### Returns:

**int(0-127)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**currentFrameTimer**()** # Print the current frame timer

## rawPixels()

#### Definition:

Return the current value of each of the 128 different pixels within the sensor as a 128 byte array

#### Parameters:

**None**

#### Returns:

**int [px1, px2, ......px128]**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**lineScan**(**f**)** # LineScan class instance

**print** l**.**rawPixels**()** # Print the raw reading of each pixel

## Data and Attributes:

#### Line modes:

**BLACK\_LINE** = 0x00

**WHITE\_LINE** = 0x80

#### Auto Exposure Setting:

**ENABLE\_AUTO** = 0x00

**DISABLE\_AUTO** = 0x01

# Fusion.locator360(*driver, addr=0x1C*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics Locator 360 sensor. Please refer to the sensor documentation for further information on the use and calibration of the sensor .

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of color sensor (Default value = 0x1C)

#### Returns:

**Locator360 sensor object**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

loc1 **=** Fusion**.**locator360**(**f**)** # LineScan class instance with the default address 0x1C

loc2 **=** Fusion**.**locator360**(**f**,** 0x20**)** # LineScan class instance with the custom address 0x20

## getHeading(*freq*)

#### Definition:

Returns the current heading of the selected frequency detection in 5 degree increments.

#### Parameters:

**freq:**int(1200 or 600)

#### Returns:

**int(0-255)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**locator360**(**f**)** # LineScan class instance

**print** l**.**getHeading**(**1200**)** # Print the current 1200hz detection heading

## getIntensity(*freq*)

#### Definition:

Returns the current intensity value of the selected frequency detection in 5 degree increments.

#### Parameters:

**freq:**int(1200 or 600)

#### Returns:

**int(0-255)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

l **=** Fusion**.**locator360**(**f**)** # LineScan class instance

**print** l**.**getIntensity**(**1200**)** # Print the current 1200hz detection intensity

## Data and Attributes:

**None**

# Fusion.range(*driver, addr=0x28*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics Locator 360 sensor. Please refer to the sensor documentation for further information on the use and calibration of the sensor .

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of color sensor (Default value = 0x28)

#### Returns:

**Range sensor object**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

r1 **=** Fusion**.**range**(**f**)** # Range class instance with the default address 0x28

r2 **=** Fusion**.**range**(**f**,** 0x20**)** # Range class instance with the custom address 0x20

## ultrasonic()

#### Definition:

Returns the current ultrasonic reading of the sensor

#### Parameters:

**None**

#### Returns:

**int(0-255)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

r **=** Fusion**.**range**(**f**)** # Range class instance

**print** r**.**ultrasonic**()** # Print the current ultrasonic reading

## optical()

#### Definition:

Returns the current optical distance reading of the sensor

#### Parameters:

**None**

#### Returns:

**int(0-255)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

r **=** Fusion**.**range**(**f**)** # Range class instance

**print** r**.**optical**()** # Print the current optical reading

## Data and Attributes:

**None**

# Fusion.seekerV3(*driver, addr=0x38*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics IR Seeker V3 sensor. Please refer to the sensor documentation for further information on the use and calibration of the sensor .

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of color sensor (Default value = 0x38)

#### Returns:

**Seeker V3 sensor object**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s1 **=** Fusion**.**seekerV3**(**f**)** # SeekerV3 class instance with the default address 0x38

s2 **=** Fusion**.**seekerV3**(**f**,** 0x20**)** # SeekerV3 class instance with the custom address 0x20

## getHeading(*freq*)

#### Definition:

Returns the current heading of the selected frequency.

#### Parameters:

**freq:**int(1200 or 600)

#### Returns:

**int(0-255)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**seekerV3**(**f**)** # SeekerV3 class instance

**print** s**.**getHeading**(**1200**)** # Print the current 1200hz detection heading

## getIntensity(*freq*)

#### Definition:

Returns the current intensity value of the selected frequency.

#### Parameters:

**freq:**int(1200 or 600)

#### Returns:

**int(0-255)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**seekerV3**(**f**)** # SeekerV3 class instance

**print** s**.**getIntensity**(**1200**)** # Print the current 1200hz detection intensity

## getLeftRaw(*freq*)

#### Definition:

Returns the current raw intensity value of the left sensor at the selected frequency.

#### Parameters:

**freq:**int(1200 or 600)

#### Returns:

**int(0-1023)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**seekerV3**(**f**)** # SeekerV3 class instance

**print** s**.**getLeftRaw**(**1200**)** # Print the raw left value of 1200Hz reading

## getRightRaw(*freq*)

#### Definition:

Returns the current raw intensity value of the right sensor at the selected frequency.

#### Parameters:

**freq:**int(1200 or 600)

#### Returns:

**int(0-1023)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**seekerV3**(**f**)** # SeekerV3 class instance

**print** s**.**getRightRaw**(**1200**)** # Print the raw right value of 1200Hz reading

## Data and Attributes:

**None**

# Fusion.sound(*driver, addr=0x34*)

#### Definition:

The following class provides the necessary drivers for the Modern Robotics Sound generator.

#### Parameters:

**driver:** Main driver object so class can call driver functions  
 **addr:** Optional address of sound generator (Default value = 0x34)

#### Returns:

**Sound gen object**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s1 **=** Fusion**.**sound**(**f**)** # Sound class instance with the default address 0x34

s2 **=** Fusion**.**sound**(**f**,** 0x20**)** # Sound class instance with the custom address 0x20

## setSound(*level, freq, length*)

#### Definition:

Sets the current sound profile and emits a tone for a certain duration. The function will immediately exit and a internal countdown timer will begin for the length specified so code execution can continue.

#### Parameters:

**level:** Sound level of output signal int (0, 1, 2, or 3)  
 **freq:** Frequency in increments of 1hZ int (0-65534)  
 **length:** Length of sound in 10mS increments int(0-255)

#### Returns:

**None**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**sound**(**f**)** # Sound class instance

s**.**setSound**(**s.MAX**,** 1000**,** 10**)** # Emit a 1000Hz signal at 3 intensity for 1 second

## setSoundBlocking(*level, freq, length, post\_pause*)

#### Definition:

Sets the current sound profile and emits a tone for a certain duration. The code will stop at this point until the duration specified has expired.

#### Parameters:

**level:** Sound level of output signal int (0, 1, 2, or 3)  
 **freq:** Frequency in increments of 1hZ int (0-65534)  
 **length:** Length of sound in 10mS increments int(0-255)  
 **post\_pause:** Pause in seconds after duration has expired, used for making a melody

#### Returns:

**None**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**sound**(**f**)** # Sound class instance

s**.**setSoundBlocking**(**s.MAX**,** 1000**,** 10, 0.1**)**# 1000Hz, 3 intensity, 1 second duration, 0.1s pause

## setVolume(*level*)

#### Definition:

Sets the current volume level of the next sound emission.

#### Parameters:

**level:** Sound level of output signal int (0, 1, 2, or 3)

#### Returns:

**None**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**sound**(**f**)** # Sound class instance

s**.**setVolume**(**s.LOW**)** # Set the volume level to 1

## setFreq(*freq*)

#### Definition:

Sets the current frequency of the next sound emission.

#### Parameters:

**freq:** Frequency in increments of 1hZ int (0-65534)

#### Returns:

**None**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**sound**(**f**)** # Sound class instance

s**.**setFreq**(**1000**)** # Set the frequency to 1kHz

## setDuration(*length*)

#### Definition:

Sets the duration and begins the next sound emission for the duration specified.

#### Parameters:

**length:** Length of sound in 10mS increments int(0-255)

#### Returns:

**None**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**sound**(**f**)** # Sound class instance

s**.**setVolume**(**s.LOW**)** # Set the volume level to 1

s**.**setFreq**(**1000**)** # Set the frequency to 1kHz

s**.**setDuration**(**100**)** # 100x10mS = 1 second duration

## getDuration()

#### Definition:

Returns the current duration timer and returns 0 when the timer has expired. Used to note when the current sound emission has completed and another can be submitted.

#### Parameters:

**None**

#### Returns:

**int (duration)**

#### Example:

**import** Fusion

f **=** Fusion**.**driver**()** # Create instance of driver class

s **=** Fusion**.**sound**(**f**)** # Sound class instance

s**.**setSound**(**s.MAX**,** 1000**,** 10**)** # Emit a 1000Hz signal at 3 intensity for 1 second

**while** s**.**getDuration**():** **pass** # Wait for sound timer to expire

s**.**setSound**(**s.MAX**,** 1000**,** 10**)** # Emit next sound

## Data and Attributes:

#### Volume Levels:

**LOW** = 0x00

**MED**  = 0x01

**HIGH** = 0x02

**MAX** = 0x03

# Fusion.usbCamera(*camera=0, resolution=(320,240), hsv\_value=((0,0,0),(255,255,255), iterations=2, min\_radius=10, cam\_output=False* )

#### Definition:

The following class provides provides the necessary driver for using a web camera to track an object and return the X/Y coordinates within the visual window of the camera. Calling this constructor begins a seperate thread process of python that updates global variables with the processed values so as to not slow down user code execution. See data and attributes for more information on returned values.

#### Parameters:

**camera:** Cam number assigned by the system, if one cam number is 0 default = 0  
 **resolution:** Resolution setting of the camera, higher res = less refresh speed default = (320,240)  
 **hsv\_value:** Sets the (upper) , (lower) hsv threshold values for object detection default = ((0,0,0),(255,255,255))  
 **iterations:** Number of dilate and erosions that take place to filter stray pixels default = 2  
 **min\_radius:** Minimum radius of pixels detected before returning the centroid default = 10  
 **cam\_output:** Camera output toggle when connected to VNC or HDMI desktop default = False

#### Returns:

**USB Camera object**

#### Example:

**import** Fusion

**import** time

camera **=** 0 # Selects first camera detected

resolution **=** **(**320**,**240**)** # Set camera resolution (width,height)

hsv\_value **=** **((**50**,**80**,**30**),(**80**,**255**,**255**))** # Set lower/upper HSV boundaries

iterations **=** 2 # Cleanup Iterations (optional)

radius **=** 2 # Set the minimum pixel radius for detection

cam\_output **=** **False** # See the camera output when in VNC or HDMI

# Constructor for the camera class

cam **=** Fusion**.**usbCamera**(**camera**,** resolution**,** hsv\_value**,** iterations**,** radius**,** cam\_output**)**

**while** **True:**

# Print the (x,y) coordinates of the centroid of the object

# based on the resolution.

**print** str**(**cam**.**COORDINATES**)** **+** " " **+** str**(**cam**.**RADIUS**)**

time**.**sleep**(**0.01**)**

## read()

#### Definition:

Wrapper for the COORDINATES and RADIUS global values making it easier to return one or all values as an array while also putting the proper delay in for allowing the sub process to update values.

#### Parameters:

**None**

#### Returns:

**int (X, Y, Radius)**

#### Example:

**import** Fusion

hsv\_value **=** **((**50**,**80**,**30**),(**80**,**255**,**255**))** # Set lower/upper HSV boundaries

# Constructor for the camera class using default cam, res, iterations, radius, and output

cam **=** Fusion**.**usbCamera**(**camera**,** resolution**,** hsv\_value**,** iterations**,** radius**,** cam\_output**)**

**while** **True:**

**print** cam**.**read**()** # Print the (X,Y,Radius) of the object

X **=** cam**.**read**[**0**]** # Put the X coordinate in variable "X"

Y **=** cam**.**read**[**1**]** # Put the Y coordinate in variable "Y"

radius **=** cam**.**read**()[**2**]** # Put the radius value into variable "radius"

## Data and Attributes:

#### Camera Global Variables:

**FRAME\_TIME** = Time it took in seconds to grab a frame from the camera **COMP\_TIME** = Time it took to process the previously grabbed frame **FPS** = Current number of Frames per Second grabbed from the camera  **KEY\_PRESS** = When using a keyboard and cam\_output = True, returns the number of the last key pressed **RADIUS** = Current radius of the detected object, used for distance tracking **COORDINATES** = Current X,Y Coordinates of the tracked object, used for direction tracking