REV Color V3

Release 2020.1.2.0

Feb 24, 2020

CHAPTER 1

REV Color V3 API

This is a python wrapper around the REV Robotics Color Sensor V3. The RobotPy project is not associated with or endorsed by REV Robotics.

1.1 rev.color Package

rev.color.CIEColor(self, X, Y, Z)	
rev.color.ColorMatch(self)	REV Robotics Color Sensor V3.
rev.color.ColorSensorV3(self, port)	REV Robotics Color Sensor V3.

1.1.1 CIEColor

class rev.color.**CIEColor**(*X: float*, *Y: float*, *Z: float*) → None

Bases: `pybind11_builtins.pybind11_object`

getX() → float

Get the X component of the color

Returns CIE X

getY() → float

Get the Y component of the color

Returns CIE Y

getYx() → float

Get the x calculated coordinate of the CIE 19313 color space

https://en.wikipedia.org/wiki/CIE_1931_color_space

Returns CIE Yx

getYy() → float

Get the y calculated coordinate of the CIE 19313 color space https://en.wikipedia.org/wiki/CIE_1931_color_space

Returns CIE Yy

`getZ()` → float

Get the Z component of the color

Returns CIE Z

1.1.2 ColorMatch

`class rev.color.ColorMatch()` → None

Bases: `pybind11_builtins.pybind11_object`

REV Robotics Color Sensor V3.

This class allows access to a REV Robotics color sensor V3 on an I2C bus.

`addColorMatch(color: wpilib._wpilib.Color)` → None

Add color to match object

Parameters `color` – color to add to matching

`matchClosestColor(colorToMatch: wpilib._wpilib.Color, confidence: float)` → wpilib._wpilib.Color

MatchColor uses euclidean distance to compare a given normalized RGB vector against stored values

Parameters

- `colorToMatch` – color to compare against stored colors
- `confidence` – The confidence value for this match, this is simply 1 - euclidean distance of the two color vectors

Returns Closest matching color

`matchColor(*args, **kwargs)`

Overloaded function.

1. matchColor(self: rev.color._rev_color.ColorMatch, colorToMatch: wpilib._wpilib.Color) -> Optional[wpilib._wpilib.Color]

MatchColor uses euclidean distance to compare a given normalized RGB vector against stored values

Parameters `colorToMatch` – color to compare against stored colors

Returns Matched color if detected

2. matchColor(self: rev.color._rev_color.ColorMatch, colorToMatch: wpilib._wpilib.Color, confidence: float) -> Optional[wpilib._wpilib.Color]

MatchColor uses euclidean distance to compare a given normalized RGB vector against stored values

Parameters

- `colorToMatch` – color to compare against stored colors
- `confidence` – The confidence value for this match, this is simply 1 - euclidean distance of the two color vectors

Returns Matched color if detected

`setConfidenceThreshold(confidence: float)` → None

Set the confidence interval for determining color. Defaults to 0.95

Parameters **`confidence`** – A value between 0 and 1

1.1.3 ColorSensorV3

class `rev.color.`**`ColorSensorV3`**(*port: wpilib._wpilib.I2C.Port*) → None

Bases: `pybind11_builtins.pybind11_object`

REV Robotics Color Sensor V3.

This class allows access to a REV Robotics color sensor V3 on an I2C bus.

Constructs a ColorSensorV3.

Note that the REV Color Sensor is really two devices in one package: a color sensor providing red, green, blue and IR values, and a proximity sensor.

Parameters **`port`** – The I2C port the color sensor is attached to

class **`ColorMeasurementRate`**(*arg0: int*) → None

Bases: `pybind11_builtins.pybind11_object`

Members:

k25ms

k50ms

k100ms

k200ms

k500ms

k1000ms

k2000ms

`k1000ms = ColorMeasurementRate.k1000ms`

`k100ms = ColorMeasurementRate.k100ms`

`k2000ms = ColorMeasurementRate.k2000ms`

`k200ms = ColorMeasurementRate.k200ms`

`k25ms = ColorMeasurementRate.k25ms`

`k500ms = ColorMeasurementRate.k500ms`

`k50ms = ColorMeasurementRate.k50ms`

`name`

(self: handle) -> str

class **`ColorResolution`**(*arg0: int*) → None

Bases: `pybind11_builtins.pybind11_object`

Members:

k20bit

k19bit

k18bit

k17bit

k16bit

k13bit

k13bit = ColorResolution.k13bit

k16bit = ColorResolution.k16bit

k17bit = ColorResolution.k17bit

k18bit = ColorResolution.k18bit

k19bit = ColorResolution.k19bit

k20bit = ColorResolution.k20bit

name
(self: handle) -> str

class GainFactor(*arg0: int*) → None
Bases: `pybind11_builtins.pybind11_object`

Members:

k1x

k3x

k6x

k9x

k18x

k18x = GainFactor.k18x

k1x = GainFactor.k1x

k3x = GainFactor.k3x

k6x = GainFactor.k6x

k9x = GainFactor.k9x

name
(self: handle) -> str

class LEDCurrent(*arg0: int*) → None
Bases: `pybind11_builtins.pybind11_object`

Members:

kPulse2mA

kPulse5mA

kPulse10mA

kPulse25mA

kPulse50mA

kPulse75mA

kPulse100mA

kPulse125mA

kPulse100mA = LEDCurrent.kPulse100mA

kPulse10mA = LEDCurrent.kPulse10mA

kPulse125mA = LEDCurrent.kPulse125mA

kPulse25mA = LEDCurrent.kPulse25mA

kPulse2mA = LEDCurrent.kPulse2mA

kPulse50mA = LEDCurrent.kPulse50mA

kPulse5mA = LEDCurrent.kPulse5mA

kPulse75mA = LEDCurrent.kPulse75mA

name
(self: handle) -> str

class LEDPulseFrequency(*arg0: int*) → None
Bases: `pybind11_builtins.pybind11_object`

Members:

k60kHz

k70kHz

k80kHz

k90kHz

k100kHz

k100kHz = LEDPulseFrequency.k100kHz

k60kHz = LEDPulseFrequency.k60kHz

k70kHz = LEDPulseFrequency.k70kHz

k80kHz = LEDPulseFrequency.k80kHz

k90kHz = LEDPulseFrequency.k90kHz

name
(self: handle) -> str

class ProximityMeasurementRate(*arg0: int*) → None
Bases: `pybind11_builtins.pybind11_object`

Members:

k6ms

k12ms

k25ms

k50ms

k100ms

k200ms

k400ms

k100ms = ProximityMeasurementRate.k100ms

k12ms = ProximityMeasurementRate.k12ms

k200ms = ProximityMeasurementRate.k200ms

k25ms = ProximityMeasurementRate.k25ms

k400ms = ProximityMeasurementRate.k400ms

k50ms = ProximityMeasurementRate.k50ms

k6ms = ProximityMeasurementRate.k6ms

name

(self: handle) -> str

class ProximityResolution(*arg0: int*) → None

Bases: `pybind11_builtins.pybind11_object`

Members:

k8bit

k9bit

k10bit

k11bit

k10bit = ProximityResolution.k10bit

k11bit = ProximityResolution.k11bit

k8bit = ProximityResolution.k8bit

k9bit = ProximityResolution.k9bit

name

(self: handle) -> str

class RawColor(*r: int*, *g: int*, *b: int*, *_ir: int*) → None

Bases: `pybind11_builtins.pybind11_object`

blue

green

ir

red

configureColorSensor(*res: rev.color._rev_color.ColorSensorV3.ColorResolution*, *rate: rev.color._rev_color.ColorSensorV3.ColorMeasurementRate*) → None

Configure the color sensor.

These settings are only needed for advanced users, the defaults will work fine for most teams. Consult the APDS-9151 for more information on these configuration settings and how they will affect color sensor measurements.

Parameters

- **res** – Bit resolution output by the respective light sensor ADCs
- **rate** – Measurement rate of the light sensor

configureProximitySensor(*res: rev.color._rev_color.ColorSensorV3.ProximityResolution*, *rate: rev.color._rev_color.ColorSensorV3.ProximityMeasurementRate*) → None

Configure the proximity sensor.

These settings are only needed for advanced users, the defaults will work fine for most teams. Consult the APDS-9151 for more information on these configuration settings and how they will affect proximity sensor measurements.

Parameters

- **`res`** – Bit resolution output by the proximity sensor ADC.
- **`rate`** – Measurement rate of the proximity sensor

`configureProximitySensorLED`(*freq: rev.color._rev_color.ColorSensorV3.LEDPulseFrequency*, *current: rev.color._rev_color.ColorSensorV3.LEDCurrent*, *pulses: int*) → None

Configure the the IR LED used by the proximity sensor.

These settings are only needed for advanced users, the defaults will work fine for most teams. Consult the APDS-9151 for more information on these configuration settings and how they will affect proximity sensor measurements.

Parameters

- **`freq`** – The pulse modulation frequency for the proximity sensor LED
- **`curr`** – The pulse current for the proximity sensor LED
- **`pulses`** – The number of pulses per measurement of the proximity sensor LED

`getCIEColor`() → rev.color._rev_color.CIEColor

Get the color converted to CIE XYZ color space using factory calibrated constants.

https://en.wikipedia.org/wiki/CIE_1931_color_space

Returns CIEColor value from sensor

`getColor`() → wpilib._wpilib.Color

Get the normalized RGB color from the sensor (normalized based on total R + G + B)

Returns frc::Color class with normalized sRGB values

`getIR`() → float

Get the normalzied IR value from the sensor. Works best when within 2 inches and perpendicular to surface of interest.

Returns Color class with normalized values

`getProximity`() → int

Get the raw proximity value from the sensor ADC. This value is largest when an object is close to the sensor and smallest when far away.

Returns Proximity measurement value, ranging from 0 to 2047 in default configuration

`getRawColor`() → rev.color._rev_color.ColorSensorV3.RawColor

Get the raw color value from the sensor.

Returns Raw color values from sensopr

`hasReset`() → bool

Indicates if the device reset. Based on the power on status flag in the status register. Per the datasheet:

Part went through a power-up event, either because the part was turned on or because there was power supply voltage disturbance (default at first register read).

This flag is self clearing

Returns bool indicating if the device was reset

`setGain`(*gain: rev.color._rev_color.ColorSensorV3.GainFactor*) → None

Set the gain factor applied to color ADC measurements.

By default, the gain is set to 3x.

Parameters **`gain`** – Gain factor applied to color ADC measurements measurements

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index