witmotion

Scott Torborg

Sep 06, 2022

CONTENTS:

This library provides a Python interface to the WitMotion line of serial-interface IMU sensors. It is tested against the HWT905, but is also expected to work for other products, including the HWT901B.

Both polling and callback-oriented APIs are available. At this time, the `witmotion` package is only designed to work in a threaded environment, and does not offer `async` APIs.

CHAPTER

ONE

QUICK START

Install and connect to a device:

```
$ pip install witmotion
$ witmotion-debug --path /dev/ttyUSB0
```

Get data via a callback:

```
from witmotion import IMU

def callback(msg):
    print(msg)

imu = IMU()
imu.subscribe(callback)
```

Get data via polling interface:

```
imu.get_quaternion()
```

CHAPTER

TWO

COMMAND-LINE DEBUG TOOL

The package includes a command-line debug and configuration tool. This tool can be used to update all parameters of the running or saved configuration on a WitMotion IMU device, or just display data to verify functionality.

Typical usage:

```
witmotion-debug --path /dev/ttyUSB0
```

Additional argumnets:

```
optional arguments:
  -h, --help            show this help message and exit
  --verbose             Verbose debugging
  --path PATH           Path to serial device
  --baudrate BAUDRATE   Serial baud rate
  --reset               Reset to default configuration
  --set-calibration-mode {none,gyro_accel,magnetic}
                        Set calibration mode
  --set-algorithm-dof {6,9}
                        Set algorithm degrees of freedom
  --set-gyro-automatic-calibration SET_GYRO_AUTOMATIC_CALIBRATION
                        Set gyro automatic calibration on or off
  --set-installation-direction {horizontal,vertical}
                        Set installation direction
  --set-baudrate SET_BAUDRATE
                        Set new baud rate
  --set-update-rate SET_UPDATE_RATE
                        Set new update rate (Hz)
  --set-messages SET_MESSAGES
                        Comma-separated list of message classes to send
  --set-acceleration-bias SET_ACCELERATION_BIAS
                        Set accelaration bias tuple (x,y,z)
  --set-angular-velocity-bias SET_ANGULAR_VELOCITY_BIAS
                        Set angular velocity bias tuple (x,y,z)
  --set-magnetic-bias SET_MAGNETIC_BIAS
                        Set magnetic bias tuple (x,y,z)
  --toggle-sleep        Toggle sleep mode
  --save                Save running configuration to NVM
```

CHAPTER

THREE

API REFERENCE

class witmotion.**IMU**(*path: str = '/dev/ttyUSB0'*, *baudrate: int = 9600*)

Main IMU interface. Instantiate to connect to a device.

Can be used via a polling interface or a streaming callback-based interface.

close() → None

Close IMU connection and stop background monitoring thread..

get_acceleration() → Optional[Tuple[float, float, float]]

Get the last acceleration vector received from the device. If no acceleration messages have been received, will return *None*.

get_angle() → Optional[Tuple[float, float, float]]

Get the last angle state received from the device. If no angle messages have been received, will return *None*.

get_angular_velocity() → Optional[Tuple[float, float, float]]

Get the last angular velocity state received from the device. If no angular velocity messages have been received, will return *None*.

get_magnetic_vector() → Optional[Tuple[float, float, float]]

Get the last magnetic vector received from the device. If no magnetic messages have been received, will return *None*.

get_quaternion() → Optional[Tuple[float, float, float, float]]

Get the last quaternion received from the device. If no quaternion messages have been received, will return *None*.

get_timestamp() → Optional[float]

Get the last timestamp received from the device. If no timestamp messages have been received, will return *None*.

save_configuration() → None

Save the currently running configuration to the device's nonvolatile memory.

send_command(*cmd: ConfigCommand*) → None

Send a command instance to the device. This should generally not be used directly: instead, use higher-level configuration methods.

send_config_command(*cmd: ConfigCommand*)

Send a configuration command instance to the device, proceeded by a special configuration sequence. This should generally not be used directly: instead, use higher-level configuration methods.

set_acceleration_bias(*values: Tuple[int, int, int]*) → None

Set the internal acceleration bias values.

set_algorithm_dof(*n: int*) → None

Set the currently active sensing algorithm in use on the device: either 6-DoF or 9-DoF.

set_angular_velocity_bias(*values: Tuple[int, int, int]*) → None

Set the internal angular velocity bias values.

set_baudrate(*rate: int*) → None

Set the serial baud rate used by the device.

set_calibration_mode(*mode:* CalibrationMode) → None

Set the current calibration mode.

set_default_configuration() → None

Restore the device to factory default configuration.

set_gyro_automatic_calibration(*enabled: bool = True*) → None

Set the current gyro automatic calibration mode: either enabled or disabled.

set_installation_direction(*direction:* InstallationDirection) → None

Set the current installation direction.

set_magnetic_bias(*values: Tuple[int, int, int]*) → None

Set the internal magnetic bias values.

set_messages_enabled(*classes: Set[Type[ReceiveMessage]]*) → None

Set the output message types enabled on the device. Pass in a set of ReceiveMessage subclasses.

set_update_rate(*rate: Optional[Union[str, float]]*) → None

Set the update rate emitted by the device.

subscribe(*callback: Callable[[ReceiveMessage], None], cls: Optional[Type[ReceiveMessage]] = None*) → None

Subscribe to update messages from the IMU.

toggle_sleep() → None

Toggle device sleep mode. If the device is currently active, it will go to sleep. If the device is current asleep, it will become active.

class witmotion.**CalibrationMode**(*value*)

Available sensor calibration modes.

gyro_accel = 1

Enable gyroscope and accelerometer calibration.

magnetic = 2

Enable magnetic calibration.

none = 0

No calibration mode enabled.

class witmotion.**InstallationDirection**(*value*)

Available installation directions.

horizontal = 0

Device installed horizontally (default).

vertical = 1

Device installed vertically.

CHAPTER

FOUR

DEVICE SUPPORT

At the time of this running, hardware support is only verified on the HWT905-232 device. Other variants of the HWT905, and other WitMotion devices, are expected to work. If you are successful or unsuccessful in using the package with any other devices, please submit a GitHub issue with your results.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX