Library.py Documentation

MyDelegate Class

“ MyDelegate class is overridden to create delegate object with custom settings that will receive Bluetooth messages asynchronously - such as notifications, indications, and advertising data - pass this information to the user by calling the class methods on the object. ”

Class Methods:

- **__init__(self):** as constructor of the class, initializes the object using DefaultDelegate and creates 6 class fields for each object that will help to calculate and store the important values such as revolutions per second and revolutions covered.

- **parse_rev (self, data):** takes the output received in bytes format from bike sensor characteristic UUID as a parameter and translates it to number of revolutions of bike wheel in int format as return value.

- **stopwatch (self, mode)**: takes the mode of the watch (“start” or “stop”) as a parameter, where if the mode is “start” the watch starts running until the “stop” mode occurs, which will stop the watch and return time recorded by the stopwatch.

- **stats (self, revAchieved, time)**: takes the revolutions achieved and time duration for revolution achieved values as parameters and uses them in a fashion like pretty print to display information in a useful organized manner and sets the values for revAchieved and the revPerSec class fields.

- **send_stats(self):** sends revolutions achieved and revolutions per sec values over to cumulocity using the imported MQTT.py module methods.

- **handleNotification (self, chandle, data):** takes the data received in bytes format and the characteristic handle as parameters, where the data is translated to revolutions achieved and revolutions per sec using (function)parse_rev and (function)stopwatch. In addition, it sends data to cumulocity using (function)send_stats.

Bluetooth Class

“ Bluetooth class manages the Bluetooth devices and retrieves information from devices using the (class)MyDelegate. ”

Class Methods:

- **__init__ (self, addr, chr):** as constructor of the class, initializes the object using the two parameters given addr and chr, which corresponds to the MAC address of the device and the characteristic UUID desired that contains the information to be retrieved.

- **connect (self, mode):** takes the mode as a parameter to decide the kind of connection desired with Bluetooth device given the mac address.

- **subscribe(self)**: subscribes to a certain characteristic UUID of Bluetooth device.

- **disconnect(self)**: closes connection of the Bluetooth device.

- **wait(self):** awaits the notification’s data to be received.