Roomba Python Manual

MESH: 2016

rread@nmsu.edu

Installation:

Download the program using github. In the terminal type the command:
git clone https://github.com/ichbinryan/ChipotleBurrito.git
then open a terminal in the directory that was downloaded. To run the program, type
python Roomba.py

Connecting to Server:

Roomba.py uses SocketIO_client library to talk to a server application. In order to make this work, you need to change the initialization of the socket so that it connects to the socket and IP address of the machine running the server. Find the line that says:

```
304     try:
305         socketIO = SocketIO("192.168.1.137", 3141)
306     except:
307         print 'issue connecting to server'
308
```

and change the IP and socket to your needs. Before starting Roomba.py, be sure the server application is running.

Class Roomba:

Class variables:

- port: name of usb port from host controller. E.g. if you are using a pi3 you might use /dev/tty/USB0
- baudrate: communication rate from serial to usb. Default is 115200 unless you are using an older Roomba. If you need to communicate at a slower rate, you can use 19200. To force the Roomba to communicate at 19200 you must do the following:

  "While powering off Roomba, continue to hold down the Clean/Power button after the light has turned off. After about 10 seconds, Roomba plays a tune of descending pitches. Roomba will communicate at 19200 baud until the processor loses battery power or the baud rate is explicitly changed by way of the OI."

Class functions:

Initializer:

```
def __init__(self, port, baud):
```

Example:

```
roomba = Roomba("/dev/ttyUSB0", 115200)
```

Initiallizes a roomba.

Parameters:
port: path to serial port.
baud: rate at which the information is sent through the serial port (115200 or 19200 unless otherwise specified by manufacturer).

Note: You must choose what mode you want to initialize your Roomba to. By default it is initialized to passive mode. This is the safest mode as it will use cliff, proximity, and other sensors to keep the robot from falling off a cliff, running into stuff, and the battery will be protected (in other modes, the battery will discharge continuously without going to sleep, this could cause issues with battery longevity or possibly other safety issues). However, passive mode will not let you take full control of the robot. In certain other functions, the robot will be taken out of safety mode in order to take control of the driving mechanisms or other, but will revert back to passive always. You can change the Roomba mode here by commenting/uncommenting full or safe (leave passive uncommented as it also tells the Roomba to begin listening):

```
self.passive()
time.sleep(0.5)
#self.safe()  #initialize roomba to safe mode
#self.full()
```

If you do change the mode, you will need to make changes later in the code so it doesn't revert back to passive mode (for example, see Seek Dock below).

Write Command:

```
def write_command(self, command):
```

Example (tell roomba to beep):

```
self.write_command("140 3 1 64 16 141 3")
```

Writes a command to the serial port converted to a byte string. This will be used to send all opcodes to control the Roomba. Used mostly by other functions to write their particular commands, but can be used otherwise to send a command of your choosing.

Parameters:
command: a **string** of numbers that represent the Roomba Open Interface opcode commands. Refer to the iCreate manual to get the opcodes for your series of robot.

Charge:

```
def charge(self):
    if MODE != 'P':
        self.passive()
    self.write_command("143")
```

Tells Roomba to return to charging dock to charge. Will return Roomba to passive mode if not already there. If you need to work outside of passive mode you will need to get rid of the entire if statement then reset the robot to your mode after reaching the charging dock.

Sing Song:

```
def sing_song(self):
    #beep
    self.write_command("140 3 1 64 16 141 3")

    #Mary had a little lamb
    #self.write_command("140 3 7 54 16 52 16 50 16 52 16 54 16 54 16 54 16 141
3")
```

Tells Roomba to beep. This is used as a signal to show that the serial port is connected and that commands are being received. If you want the Roomba to play "Mary had a little lamb" comment the line under #beep and uncomment the line under #Mary had a little lamb."

Modes (Passive, Safe, Full, Stop):

```
def passive(self):
    MODE = 'P'
    self.write_command('128')

def safe(self):
    MODE = 'S'
    self.write_command('131')

def full(self):
    MODE = 'F'
    self.write_command('132')
```

Sets the Roomba mode.

Passive: Sets Roomba to passive mode. In command mode, you cannot control motors, sensors, or other actuators, but you can receive signal data from them. When you command the Roomba to enter any cleaning mode it will revert back to passive.

Safe: Allows you to take full control of the Roomba except for cliff sensors (Roomba will stop if it senses a cliff), wheel drop sensor (Roomba will stop if it senses a wheel drop), or charging (Roomba senses it has returned to its dock and is charging). If any of these three events take place the Roomba will revert to passive mode. Roomba will not enter sleep mode and will continue to discharge battery indefinitely.

Full: Allows you to take full control of the Roomba. Roomba will wait for commands

and not respond to any of the buttons pressed (Roomba will only respond to Open Interface commands). Roomba will not enter sleep mode and will continue to discharge the battery. Further, Roomba will not charge on the dock when in safe mode. Be sure to reset or return to passive mode after you are done working in Full mode.

Stop: Roomba stops accepting OI commands and changes mode to off. Use this command when you are finished with the Roomba.

Police:

Flashes the onboard Roomba LED like police lights. Can only be used in safe or passive modes.

Clean:

Tells Roomba to enter cleaning mode. Roomba will return to passive mode once the clean command is given.

Max:

Tells Roomba to enter max cleaning mode. Roomba will clean until the battery is dead.

Spot:

Tells Roomba to enter spot cleaning mode. Roomba will clean in a spiral pattern out from its current location and clean the area near to its starting position.

Seek Dock:

```
def seek_dock(self):
    self.write_command("143")

    #stream charge
    x = -1
    while x < 0:
        x = self.get_current()
        print 'waiting'
        time.sleep(.5)

    time.sleep(3)

    #back roomba off dock
    if MODE == 'P':
        self.safe()
        time.sleep(.5)

    self.write_command('145 255 56 255 56')
    time.sleep(.5)
    self.write_command('145 0 0 0 0')
```

```
    if MODE == 'S':
        self.passive()
```

Will tell Roomba to return to the dock. When it senses it has begun charging, it will back up a few inches and wait to charge until commanded. In this command, the mode is changed twice. First, it is changed to Safe mode so that the function can take control of the wheel motors. Afterwards, it returns to passive. If you are working in another mode, you will need to return the Roomba to your desired (by calling one of the mode functions) mode at the end of the function.

Get Charge:

```
def get_charge(self):
```

Instructs Roomba to begin sending data packets containing the charge of the Roomba's battery. As it is implemented now, the Roomba will return an unsigned integer value from 0 to 65,000 (a value near 6,500 indicates a full charge). At the end of the function, get_charge calls the convertUnsigned function to convert the two bytes returned to an integer value, then returns that integer value.

Get Current:

```
def get_current(self):
```

Returns a 2 byte signed integer indicating the current flowing through the Roomba at the given moment in milliAmperes. A negative value indicates current is flowing out of the battery while a positive value indicates current flowing into the battery (charging).

Get Voltage:

```
def get_voltage(self):
```

Returns a 2 byte unsigned value indicating the voltage supplied by the Roomba battery.

Get Capacity:

```
def get_capacity(self):
```

Returns the estimated charge capacity of the Roomba's battery in milliamp-hours.

Convert Unsigned:

```
def convertUnsigned(self, high, low):
```

Converts 2 bytes received from the Roomba via one of the above get operations and converts them into an unsigned integer, and returns this unsigned integer. High is the high byte of the integer and low is the low byte. Different return values may be signed or unsigned, so the proper convert method must be used.

Convert Signed:

```
def convertSigned(self, high, low):
```

Converts 2 bytes received from the Roomba via one of the above get operations and converts them into an signed integer, and returns this signed integer. High is the high byte of the integer and low is the low byte.

Get Decoded Bytes:

```
getDecodedBytes(self, n, fmt):
```

Returns unpacked C struct data. Needed to unpack the serial communication sent by the Roomba. N is the byte of data and fmt is the format of the data type. Most of the get methods mentioned above use signed and unsigned characters rather than integers.

```
Formats needed
b = signed char
B = unsigned char
I = unsigned int
i = signed int
```