Robot Expansion Unit
Class Library Reference

Published 2019/04/23
Revised 2019/09/19

Revision History

Date Revised	Revised Contents
2019/4/23	First release
2019/9/19	Color Sensors and Ultrasonic Sensors added

Index

1. Getting Started

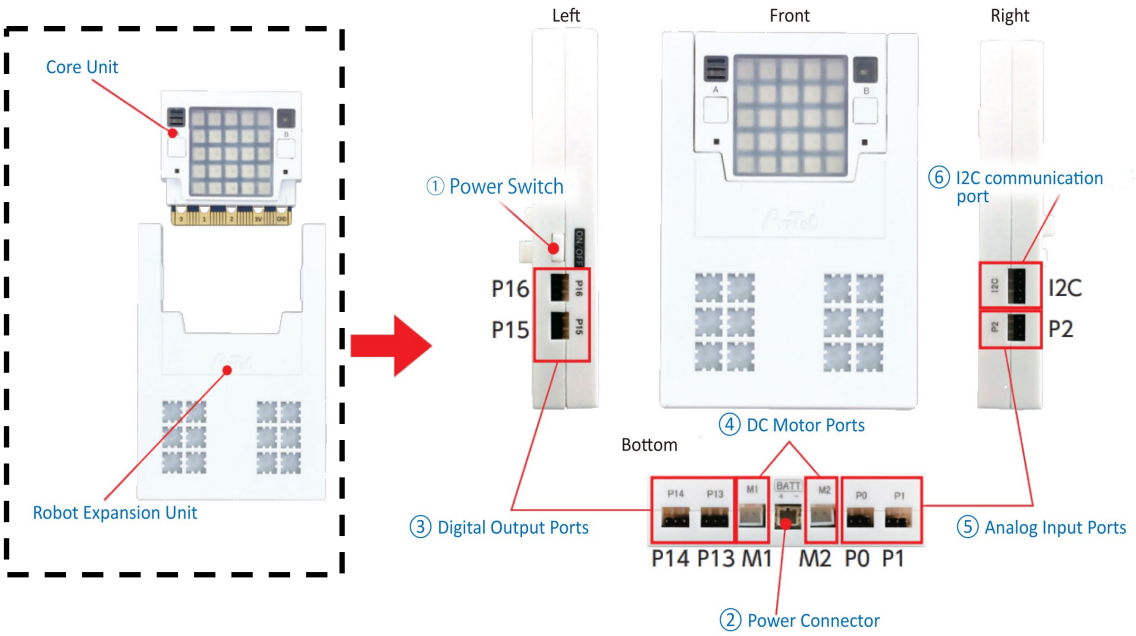
This manual is a reference guide for the ArtecRobo 2.0 class library. Using this manual requires some prior understanding of the basics of Python.

2. ArtecRobo 2.0

ArtecRobo 2.0 consists of the Core Unit (Studuino:bit) and the Robot Expansion Unit. MicroPython is one of the programming environments compatible with ArtecRobo 2.0.

2.1. Robot Expansion Unit Layout

The layout of the Robot Expansion Unit is shown below.



The Robot Expansion Unit provides additional ports you can use to attach sensors and motors to your robot. It contains digital output ports, DC Motor ports, analog input ports, and an I2C communications port. Consult the table below to see which parts are usable with which ports.

Port	Parts
Digital Output Ports	LEDs, Buzzers, Servomotors
DC Motor Ports	DC Motors
Analog Input Ports	Light Sensors, Sound Sensors, Touch Sensors, IR Photoreflectors
I2C Communication Port	Accelerometers

2.2. MicroPython

MicroPython is version of Python developed specifically for use with microcontrollers. The programming syntax is identical to Python 3.0, but some Python libraries are not usable in MicroPython because they're not designed to work with the limited memory and CPU of a microcontrolller. However, some MicroPython-specific libraries (such as the library for GPIO microcontrollers) are included as standard.

MicroPython has been widely ported to different microcontrollers, and the ArtecRobo 2.0 library utilizes a customized version.

3. Development Environment

uPyCraft is one development environment used for MicroPython. See section 6.2 in the Core Unit Class Library Reference manual for instructions on how to use uPyCraft.

4. ArtecRobo 2.0 Class Library

The ArtecRobo 2.0 class library is structured as follows.

Package	Module	Class	Parts
pyatcrobo2	parts	DCMotor	DC Motors
		Servomotor	Servomotors
		Buzzer	Buzzers
		LEDs	LEDs
		LightSensor	Light Sensors
		SoundSensor	Sound Sensors
		TouchSensor	Touch Sensors
		Temperature	Temperature Sensors
		UltrasonicSensor	Ultrasonic Sensors
		ColorSensor	Color Sensors
		Accelerometer	Accelerometers

4.1. The DCMotor Class

This class is used to control DC Motors.

4.1.1. Constructors

Use this to make objects that control DC Motors plugged into a specified DC Motor port (M1 or M2) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import DCMotor
m1 = DCMotor('M1')
```

4.1.2. Controlling a DC Motor

Use the cw() method to make a DC Motor spin clockwise and the ccw() method to make it spin counter-clockwise. The stop() method will stop making the DC Motor spin, while the brake() method will bring the DC Motor's movement to a full stop. The power(power) method can be used to adjust a DC Motor's speed. Set the power parameter to a number between 0 and 255 to pick a speed.

```
from pyatcrobo2.parts import DCMotor
import time
m1 = DCMotor('M1')
m2 = DCMotor('M2')
m1.power(100)
m2.power(100)
m1.cw()
m2.cw()
time.sleep_ms(1000)
m1.stop()
m2.stop()
```

This will make the DC Motors connected to ports M1 and M2 spin clockwise at speed 100 for 1 second before stopping.

4.2. The Servomotor Class

This class is used to control Servomotors.

4.2.1. Constructors

Use this to make objects that control Servomotors plugged into a specified port (P13, P14, P15 or P16) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import Servomotor
sv13 = Servomotor('P13')
```

4.2.2. Controlling Servomotors

Use the set_angle(degree) method to set a Servomotor's angle.

```
from pyatcrobo2.parts import Servomotor
import time
sv13 = Servomotor('P13')
sv13.set_angle(0)
time.sleep_ms(1000)
sv13.set_angle(180)
```

This will make the Servomotor connected to port P13 turn to 0°, then turn to 180° 1 second later.

4.2.3. Releasing PWMs

ArtecRobo 2.0 uses PWM to set Servomotor angles. It can use up to four PWMs simultaneously. If you're using 5 or more parts that utilize PWMs, you will be unable to use any Servomotor objects you make. You can resolve this by using the release() method to release a PWM assigned to a different Servomotor object.

4.3. The Buzzer Class

This class is used to control the Buzzers.

4.3.1. Constructors

Use this to make objects that control Buzzers plugged into a specified port (P13, P14, P15 or P16) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import Buzzer
bz13 = Buzzer('P13')
```

4.3.2. Controlling Buzzers

You can use the method on(sound, *, volume=None, duration=None) to play sound from Buzzers. Use the sound parameter to set the frequency of the sound using a MIDI note number or note name. For a list of usable note numbers and names, see section 5.1. The volume parameters lets you set the volume from 0 to 100. The duration parameter is used to set the length of the sound output in milliseconds. If the duration parameter is left unspecified, the off() method can be used to stop the buzzer.

```
from pyatcrobo2.parts import Buzzer
bz13 = Buzzer('P13')
bz13.on('C4', duration=1000)
```

This will make the Buzzer plugged into port P13 on the Robot Expansion Unit play the note C4 for 1 second.

4.3.3. Releasing PWMs

ArtecRobo 2.0 uses PWM for its sound output. It can use up to four PWMs simultaneously. If you're using 5 or more parts that utilize PWMs, you will be unable to use any Buzzer objects you make. You can resolve this by using the release() method to release a PWM assigned to a different Buzzer object.

4.4. The LED Class

This class is used to control LEDs.

4.4.1. Constructors

Use this to make objects that control LEDs plugged into a specified port (P13, P14, P15 or P16) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import LED
led13 = LED('P13')
```

4.4.2. Controlling LEDs

Use the on() method to switch an LED on and and the off() method to switch it off.

```
from pyatcrobo2.parts import LED
import time
led13 = LED('P13')
while True:
    led13.on()
    time.sleep_ms(500)
    led13.off()
    time.sleep_ms(500)
```

This will make the LED connected to port P13 on the Robot Expansion Unit turn on and off in 500 millisecond intervals.

4.5. The IRPhotoReflector Class

This class is used to control IR Photoreflectors.

4.5.1. Constructors

Use this to make objects that control IR Photoreflectors plugged into a specified port (P0, P1, or P2) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import IRPhotoReflector
sensor0 = IRPhotoReflector('P0')
```

4.5.2. Finding IR Photoreflector Values

Use the method get_value() to find the analog value of an IR Photoreflector. It will be formatted as an integer from 0 to 4095.

```
from pyatcrobo2.parts import IRPhotoReflector
import time
sensor0 = IRPhotoReflector('P0')
while True:
    print(sensor0.get_value())
    time.sleep_ms(500)
```

This will display the value of the IR Photoreflector plugged into port P0 on the Robot Expansion Unit through the terminal every 500 milliseconds.

4.6. The LightSensor Class

This class is used to control Light Sensors.

4.6.1. Constructors

Use this to make objects that control Light Sensors plugged into a specified port (P0, P1, or P2) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import LightSensor
sensor0 = LightSensor( 'P0' )
```

4.6.2. Finding Light Sensor Values

Use the method get_value() to find the analog value of a Light Sensor. It will be formatted as an integer from 0 to 4095.

```
from pyatcrobo2.parts import LightSensor
import time
sensor0 = LightSensor( 'P0' )
while True:
    print(sensor0.get_value())
    time.sleep_ms(500)
```

This will display the value of the Light Sensor plugged into port P0 on the Robot Expansion Unit through the terminal every 500 milliseconds.

4.7. The SoundSensor Class

This class is used to control Sound Sensors.

4.7.1. Constructors

Use this to make objects that control Sound Sensors plugged into a specified port (P0, P1, or P2) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import SoundSensor
sensor0 = SoundSensor('P0')
```

4.7.2. Finding Sound Sensor Values

Use the method get_value() to find the analog value of a Sound Sensor. It will be formatted as an integer from 0 to 4095.

```
from pyatcrobo2.parts import SoundSensor
import time
sensor0 = SoundSensor('P0')
while True:
    print(sensor0.get_value())
    time.sleep_ms(500)
```

This will display the value of the Sound Sensor plugged into port P0 on the Robot Expansion Unit through the terminal every 500 milliseconds.

4.8. The TouchSensor Class

This class is used to control Touch Sensors.

4.8.1. Constructors

Use this to make objects that control Touch Sensors plugged into a specified port (P0, P1, or P2) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import TouchSensor
sensor0 = TouchSensor('P0')
```

4.8.2. Finding Touch Sensor Values

Use the method get_value() to find the digital value of a Touch Sensor. The value will be either 0 or 1. Using the is_pressed() method will return a True/False to tell you whether the Touch Sensor is being pressed.

```
from pyatcrobo2.parts import TouchSensor
import time
sensor0 = TouchSensor('P0')
while True:
    print(sensor0.get_value())
    print(sensor0.is_pressed())
    time.sleep_ms(500)
```

This will state whether the Touch Sensor plugged into port P0 on the Robot Expansion Unit is being pressed or not on through the terminal every 500 milliseconds. If the Touch Sensor is being pressed, it will display 0 and True. If not, it will display 1 and False.

4.9. The Temperature Class

This class is used to control Temperature Sensors.

4.9.1. Constructors

Use this to make objects that control Temperature Sensors plugged into a specified port (P0, P1, or P2) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import Temperature
sensor0 = Temperature('P0')
```

4.9.2. Finding Temperature Sensor Values

Use the method get_value() to find the analog value of a Temperature Sensor. It will be formatted as an integer from 0 to 4095. The get_celsius() method can be used to find a Temperature Sensor's value in degrees Celsius.

```
from pyatcrobo2.parts import Temperature
import time
sensor0 = Temperature('P0')
while True:
    print(sensor0.get_value())
    print(sensor0.get_celsius())
    time.sleep_ms(500)
```

This will display the analog value and the temperature in Celsius from the Temperature Sensor plugged into port P0 through the terminal every 500 milliseconds.

4.10. The Accelerometer Class

This class is used to operate Accelerometers.

4.10.1. Constructors

Use this to make objects that control Accelerometers plugged into the I2C port on the Robot Expansion Unit.

```
from pyatcrobo2.parts import Accelerometer
sensor0 = Accelerometer( 'I2C' )
```

The Accelerometer has a full scale (maximum measurable acceleration) of ± 2G, shown in units of G.

4.10.2. Finding Accelerometer Values

Use the get_values() method to find your Accelerometer's values. The values will be returned in a tuple (x, y, z) format. The get_x(), get_y() and get_z() methods can each be used find the acceleration along the x-axis, y-axis and z-axis separately.

```
from pyatcrobo2.parts import Accelerometer
import time
sensor0 = Accelerometer( 'I2C' )
while True:
    print(sensor0.get_values())
    time.sleep_ms(500)
```

This will display the values of the Accelerometer plugged into the I2C port on the Robot Expansion Unit through the terminal every 500 milliseconds.

4.11. The UltrasonicSensor Class

This class is used to control Ultrasonic Sensors.

4.11.1. Constructors

Use this to make objects that control Ultrasonic Sensors plugged into a specified port (P0 or P1) on the Robot Expansion Unit.

```
from pyatcrobo2.parts import UltrasonicSensor
sensor0 = UltrasonicSensor ( 'P0' )
```

4.11.2. Finding Ultrasonic Sensor Values

Use the method get_distance() to find how far a target object is from the sensor. The distance will be measured in cm and returned as a float-type integer. You can also use the method get_pulse_time() to find the length of time (in microseconds) it takes for an ultrasonic wave to bounce back to the sensor after it is emitted. If the distance can't be measured, a Runtime Error exception will occur.

```
from pyatcrobo2.parts import UltrasonicSensor
import time
sensor0 = UltrasonicSensor('P0')
while True:
    try:
        print(sensor0.get_distance())
    except RuntimeError as e:
        print(e)

    time.sleep_ms(100)
```

This will display the value of the Ultrasonic Sensor plugged into port P0 on the Robot Expansion Unit through the terminal every 100 milliseconds.

4.12. The ColorSensor Class

This class is used to control Color Sensors.

4.12.1. Constructors

Use this to make objects that control Color Sensors plugged into the I2C port on the Robot Expansion Unit.

```
from pyatcrobo2.parts import ColorSensor
sensor0 = ColorSensor ( 'I2C' )
```

4.12.2. Finding Color Sensor Values

Use the get_values() method to find the color value of a target object. The color value will be returned in a tuple format showing the saturation of each component in the color (red, green, blue, brightness). The get_colorcode() method finds the color of the target object.

```
from pyatcrobo2.parts import ColorSensor
import time
color = ColorSensor('I2C')
while True:
    try:
        print(color.get_values())
        print(color.get_colorcode())
    except RuntimeError as e:
        print(e)

    time.sleep_ms(100)
```

This will display the values of the Color Sensor plugged into the I2C port on the Robot Expansion Unit through the terminal every 100 milliseconds.

5. Appendices

5.1. Sound Output

The MIDI note numbers and note names used in the Buzzer class are as follows.

MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note	MIDI	Note
48	C3	60	C4	72	C5	84	C6	96	C7	108	C8	120	C9
49	CS3	61	CS4	73	CS5	85	CS6	97	CS7	109	CS8	121	CS9
50	D3	62	D4	74	D5	86	D6	98	D7	110	D8	122	D9
51	DS3	63	DS4	75	DS5	87	DS6	99	DS7	111	DS8	123	DS9
52	E3	64	E4	76	E5	88	E6	100	E7	112	E8	124	E9
53	F3	65	F4	77	F5	89	F6	101	F7	113	F8	125	F9
54	FS3	66	FS4	78	FS5	90	FS6	102	FS7	114	FS8	126	FS9
55	G3	67	G4	79	G5	91	G6	103	G7	115	G8	127	G9
56	GS3	68	GS4	80	GS5	92	GS6	104	GS7	116	GS8		
57	A3	69	A4	81	A5	93	A6	105	A7	117	A8		
58	AS3	70	AS4	82	AS5	94	AS6	106	AS7	118	AS8		
59	B3	71	B4	83	B5	95	B6	107	B7	119	B8		

5.2. Color Codes

Constant	Color	Value
COLOR_RED	Red	1
COLOR_GREEN	Green	2
COLOR_BLUE	Blue	3
COLOR_WHITE	White	4
COLOR_YELLOW	Yellow	5
COLOR_ORANGE	Orange	6
COLOR_PURPLE	Purple	7
COLOR_UNDEF	Unknown	0

5.3. Class Table

The full ArtecRobo2 class library is as follows.

Function	Class/Module	Method/Attribute	Instructions
DC Motors	DCMotor	__init__(pin)	Use the pin parameter to specify pin M1 or M2 and create a DC Motor instance. dcm = DCMotor('M1')
		cw()	Spin a DC Motor clockwise.
		ccw()	Spin a DC Motor counter-clockwise.
		stop()	Cease spinning a DC Motor.
		brake()	Completely stop the rotation of a DC Motor.
		power(power)	Set the power parameter to a number 0-255 to change a DC Motor's speed.
Servomotors	Servomotor	__init__(pin)	Use the pin parameter to specify pin P13, P14, P15 or P16 and create a Servomotor instance. sv = ServoMotor('P13')
		set_angle(degree)	Set the degree parameter to an angle 0-180 to change a Servomotor's angle. sv.set_angle(90)
		release()	Release a PWM assigned to a Servomotor object.

Buzzers	Buzzer	__init__(pin)	Use the pin parameter to specify pin P13, P14, P15 or P16 and create a Buzzer instance. bzr = Buzzer('P13')

		on(sound, *,
		volume=-1,
		duration=-1)
			Set the Buzzer's sound in the sound, duration, and volume parameters. sound can be set using note names (C3-G9), MIDI note numbers (48-127), or frequency (as an integer). volume can be set 1-100, and duration can be set in ms. If the duration parameter is left unspecified, the Buzzer will keep playing until you call the off method. If the volume parameter is left unspecified the volume will be adjusted according to the pitch of the sound.
			bzr.on('50', duration=1000) # The buzzer will play the note corresponding to MIDI note number 50 for 1 second.
			bzr.on('C4', volume=1, duration=1000) # The buzzer will plat note C4 at volume 1 for 1 second.
			bzr.on(440)

		off()	Make the Buzzer stop playing sound. bzr.off()
		release()	Release a PWM assigned to a Buzzer object.
LEDs	LED	__init__(pin)	Use the pin parameter to specify pin P13, P14, P15 or P16 and create an LED instance. led = LED('P13')
		on()	Switch an LED on. led.on()
		off()	Switch an LED off. led.on()
IR Photoreflectors	IRPhotoReflector	__init__(pin)	Use the pin parameter to specify pin P0, P1 or P2 and create an IR Photoreflector instance. irp = IRPhotoReflector('P0')
		get_value()	Retrieve the sensor's value (0-4095). irp.get_value()

Light Sensors	LightSensor	__init__(pin)	Use the pin parameter to specify pin P0, P1 or P2 and create a Light Sensor instance. ls = LightSensor('P0')
		get_value()	Retrieve the sensor's value (0-4095). ls.get_value()
Temperature Sensors	Temperature	__init__(pin)	Use the pin parameter to specify pin P0, P1 or P2 and create a Temperature Sensor instance. ts = Temperature('P0')
		get_value()	Retrieve the sensor's value (0-4095). ts.get_value()
		get_celsius()	Retrieve the value of the Temperature Sensor in degrees Celsius. ts.get_celsius()
Sound Sensors	SoundSensor	__init__(pin)	Use the pin parameter to specify pin P0, P1 or P2 and create a Sound Sensor instance. ss = SoundSensor('P0')
		get_value()	Retrieve the sensor's value (0-4095). ss.get_value()
Touch Sensors	TouchSensor	__init__(pin)	Use the pin parameter to specify pin P0, P1 or P2 and create a Touch Sensor instance. ss = TouchSensor('P0')
		get_value()	Retrieve the sensor's value (0 or 1). ss.get_value()
		is_pressed()	Find the current state of the sensor. Returns a True if the Touch Sensor is being pressed.
Ultrasonic Sensors	UltrasonicSensor	get_pulse_time ()	Retrieve the time it takes for an ultrasonic wave to bounce back from the target object in microseconds.
		get_distance()	Retrieve the distance from the target object as a float-type integer. The units of distance will be cm.

Color Sensors	ColorSensor	get_values()	Retrieve the sensor's color value in a tuple format (red, green, blue, brightness). Each value in the tuple is the saturation of that component color.
		get_colorcode	Retrieve the color of the target object as a color code. (See section 5.2.)
Accelerometers	Accelerometer	__init__(pin)	Use the pin parameter to specify pin I2C and create an Accelerometer instance. acc = Accelerometer('I2C')
		get_values()	Find the acceleration along the x, y, and z axes. The maximum measurement range is ±2G and the units are in G. acc.get_values()
		get_x	= get_values()[0]
		get_y	= get_values()[1]
		get_z	= get_values()[2]