Documentation version 2.1: 2009 June 16

Pycreate Library API

This document explains the pycreate library used to control the iRobot Creates in Rose-Hulman Institute of Technology's CSSE-120R course. It contains classes, variables, and methods used to develop software for the course and to access additional functionality available on the Create.

CREATE

A Create object is an abstraction for the robot. Its methods are used to communicate with the robot via a serial port.

Initializing:

The code `robot = Create(port)` where `port` is the number of the Bluetooth com port to which the robot is connected, will make a Create object called `robot`. Once a Create object is instantiated, the program will control access to that serial port. This means that programs using a Create object should properly disconnect from the port at completion. Should code using a Create object crash, the port may still be controlled by the program which may cause problems with running other code attempting to use the same port.

Methods

This table describes the methods that allow you to interact with the Create robot.

Name	Description	
reconnect(comPort)	Closes and reopens the connection to the Create. Helpful if the sensors don't initialize properly.	
	Arguments:	comPort –port number to which the Create is connected
	Return:	n/a
shutdown()	Stops the Create, places it into PASSIVE_MODE, and closes the serial connection	
	Arguments:	n/a
	Return:	n/a
go(cmPerSec, degPerSec)	Moves the Create forward/reverse at the requested speed while rotating CW/CCW at the requested speed.	
	Arguments:	cmPerSec – velocity with which to translate the Create. Range is -50 to 50. Negative moves it backward.
		degPerSec – velocity with which to rotate the Create. Positive is CCW, negative is CW.
	Return:	n/a
driveDirect(leftCmSec, rightCmSec)	Sets the robot's wheel velocities to the given values	
	Arguments:	leftCm Sec – velocity for left wheel
		rightCmSec – velocity for right wheel
	Return:	n/a
drive(roombaMmSec, roombaRadiusMm, turnDir = 'CCW')	Instructs the robot to drive in an arc of the requested radius at the requested speed. Use a positive radius to drive a CW arc (and negative to drive CCW). Optionally, if radius = 0, specify whether the spin should be CCW or CW using turnDir. Note that the parameters for this function are in millimeters.	
	Arguments:	roombaMmSec – speed at which to move
		roombaRadiusMm – Radius of arc to move in (+ CW, - CCW)
		turnDir – If radius = 0, use turnDir to determine direction the robot should turn (optional argument, defaults to CCW)
	Return:	n/a
stop()	Stops the Create's wheel motors	
	Arguments:	n/a
	Return:	n/a

All wait commands do not allow new commands to run while the robot waits; but queue them for later execution.

waitDistance(centimeters)	Waits until the Create has traveled the requested distance before executing the next command	
	Arguments:	centimeters – Distance to travel
	Return:	n/a
waitAngle(degrees)	Waits until the Create has rotated the requested number of degrees before executing the next command	
	Arguments:	degrees – angle to rotate
	Return:	n/a
setLEDs(powerColor, powerIntensity, play, advance)	Turns the play and advance LEDs on or off (1 or 0, respectively). Sets the power LED to the requested brightness (0 to 255) and the requested color, ranging from 0 (Green) to 255 (Red).	
	Arguments:	powerColor – Color for power LED
		powerIntensity – Brightness for power LED
		play – play LED on/off
		advance – advance LED on/off
	Return:	n/a
playSong(noteList)	Plays the notes from noteList sequentially. The list consists of (noteNumber, duration) tuples.	
	Arguments:	noteList – list of notes to play
	Return:	n/a
playNote(noteNumber, duration, songNumber)	Plays a single note for the requested duration.	
	Arguments:	noteNumber – index of the note to play (0-38)
		duration – how long to play the note(0 to 255, measured in 64ths of a second)
		songNumber – number of song to play the note as
	Return:	n/a
setSong(songNumber, noteList)	Stores the list of notes from noteList (up to 16 notes per song) into the robot's memory as the song number given by songNumber (0 to 15).	
	Arguments:	songNumber – song number to store song as
		noteList – list of notes in the song
	Return:	n/a
playSongNumber(songNumber)	Plays the song stored in the robot's memory with the requested song number.	
	Arguments:	songNumber – number of song in memory to play
	Return:	n/a
getSensor(sensorToRead)	Queries the requested sensor and returns its interpreted value. The argument should be a String containing the name of one of the sensor mappings from the table in the "Sensor Access" section below.	
	Arguments:	sensorToRead – String name of the requested sensor
	Return:	The interpreted data for the requested sensor mapping: an integer or an array if successful or None if the sensor cannot be successfully queried.
startIR(byte_value)	Starts broadcasting the given IR signal.	
	Arguments:	byte – integer (0-254) to broadcast. 255 is "no signal".
	Return:	n/a
stopIR()	Stop broadcasting the IR signal being sent as a result of startIR().	
	Arguments:	n/a
	Return:	n/a

SENSOR ACCESS

This section contains important information on the available sensors, and how to properly access them. Examples of using getSensor():

```
	angle = robot.getSensor(“ANGLE”) # create’s current angle relative to its start

	leftBumper = robot.getSensor(“BUMPS_AND_WHEEL_DROPS”)[3] # or use BUMP_LEFT for 3
	if (leftBumper == 1):
		print “Hit something on the left!”
```

34 different sensors are available:

Name	Description
BUMPS_AND_WHEEL_DROPS	The values of the bumper and wheel drop sensors (0 = no bump, 1 = bump or 0 = wheel raised, 1 = wheel dropped). Interpreted as an array of 5 values: [WHEELDROP_CASTER, WHEELDROP_LEFT, WHEELDROP_RIGHT, BUMP_LEFT, BUMP_RIGHT]
CLIFF_LEFT_SIGNAL	The strength of the left cliff sensor’s signal (0-4095). Interpreted as an int.
CLIFF_FRONT_LEFT_SIGNAL	The strength of the front left cliff sensor’s signal (0-4095). Interpreted as an int.
CLIFF_FRONT_RIGHT_SIGNAL	The strength of the front right cliff sensor’s signal (0-4095). Interpreted as an int.
CLIFF_RIGHT_SIGNAL	The strength of the right cliff sensor’s signal (0-4095). Interpreted as an int.
WALL_SIGNAL	The strength of the wall sensor’s signal (0-4095). Interpreted as an int.
BUTTONS	The state of the Play and Advance buttons on the Create (0 = button not pressed, 1 = button pressed). Interpreted as an array of 2 values: [BUTTON_ADVANCE, BUTTON_PLAY]
DISTANCE	The distance in millimeters the Create has traveled since the last distance request (-32768 to 32767). Positive values indicate forward travel and negative values indicate reverse travel. Interpreted as an int.
ANGLE	The angle in degrees the Create has turned since the last angle request (-32768 to 32767). Positive values are CCW and negative are CW. Interpreted as an int.
IR_BYTE	The byte received by the IR sensor (255=no signal detected, 0-244 = specific IR value received) Interpreted as an int.
VOLTAGE	Voltage in millivolts of the Create’s battery (0-65535). Interpreted as an int.
OI_MODE	The current mode the Create is in (0, 1, 2, 3). Interpreted as an int.
SONG_PLAYING	State of whether or not a song is playing (0 = no song playing, 1 = song playing). Interpreted as an int.
SONG_NUMBER	The current song being played (0 to 15). Interpreted as an int.
VIRTUAL_WALL	State of the virtual wall sensor (0 = no virtual wall detected, 1 = virtual wall detected). Interpreted as an int.
OVERCURRENTS	State of the two wheel and three Low Side Driver overcurrent sensors (0 = no overcurrent, 1 = overcurrent). Interpreted as an array of 5 values: [LEFT_WHEEL, RIGHT_WHEEL, LD_2, LD_0, LD_1]
CHARGING_STATE	Indicated the current charging state of the Create. Interpreted as an int. Possible values: 0 Not charging 1 Reconditioning Charging 2 Full Charging 3 Trickle Charging 4 Waiting 5 Charging Fault Condition
CURRENT	The current in milliamps flowing into or out of the Create’s battery (-32768 to 32767). Negative currents are discharging and positive currents are charging. Interpreted as an int.
BATTERY_TEMPERATURE	Temperature in Celsius of the Create’s battery (-128 to 127). Interpreted as int.

BATTERY_CHARGE	The current charge of the Create's battery in milliamp-hours (0-65535). Interpreted as an int.
BATTERY_CAPACITY	The estimated charge capacity of the Create's battery in milliamp-hours (0-65535). Interpreted as an int.
NUMBER_OF_STREAM_PACKETS	Number of data stream packets (0 to 43). Interpreted as an int.
USER_DIGITAL_INPUTS	The values of the digital input pins from the cargo bay connector (each is 0 or 1). Interpreted as an array of 5 values: [BAUD_RATE_CHANGE, DIGITAL_INPUT_3, DIGITAL_INPUT_2, DIGITAL_INPUT_1, DIGITAL_INPUT_0]
USER_ANALOG_INPUT	The value of the analog input pin from the cargo bay connector (0 to 1023). Interpreted as an int.
CHARGING_SOURCES_AVAILABLE	The state of the Create's connection to a charging course (each is 0 or 1). Interpreted as an array of 2 values: [HOME_BASE, INTERNAL_CHARGER]
WALL	State of the wall sensor (0 = no wall, 1 = wall seen). Interpreted as an int .
CLIFF_LEFT	State of the left cliff sensor (0 = no cliff, 1 = cliff). Interpreted as an int.
CLIFF_FRONT_LEFT	State of the front left cliff sensor (0 = no cliff, 1 = cliff). Interpreted as an int.
CLIFF_FRONT_RIGHT	State of the front right cliff sensor (0 = no cliff, 1 = cliff). Interpreted as an int.
CLIFF_RIGHT	State of the right cliff sensor (0 = no cliff, 1 = cliff). Interpreted as an int.
VELOCITY	The velocity most recently requested in mm/s (-500 to 500). Interpreted as int.
RADIUS	The radius most recently requested in mm (-32768 to 32767). Interpreted as int.
RIGHT_VELOCITY	The velocity of the right wheel in mm/s (-500 to 500). Interpreted as an int.
LEFT_VELOCITY	The velocity of the left wheel in mm/s (-500 to 500). Interpreted as an int.

For more information on the USER_DIGITAL_INPUTS and USER_ANALOG_INPUT sensors, see the "User IO" section

ADVANCED FEATURES

You should be able to complete all the assignments for this course without anything else in this document. They are included for the sake of completeness and the adventuresome.

OTHER METHODS OF THE CREATE CLASS

Name	**Description**	
setDigitalOutputs(digOut2, digOut1, digOut0)	Sets the digital output pins to the requested values (each 0 or 1).	
	Arguments:	digOut2 – Value for digital output pin 2
		digOut1 – Value for digital output pin 1
		digOut0 – Value for digital output pin 0
	Return:	n/a
setLowSideDrivers(driver2, driver1, driver0)	Sets the low side driver pins to fully on (1) or fully off (0).	
	Arguments:	driver2 – Value for low side driver pin 2
		driver1 – Value for low side driver pin 1
		driver0 – Value for low side driver pin 0
	Return:	n/a
setPWMLowSideDrivers(dutyCycle2, dutyCycle1, dutyCycle0)	Sets the low side driver pins to the requested duty cycle (0 to 255).	
	Arguments:	dutyCycle2 – Duty cycle for low side driver pin 2
		dutyCycle1 – Duty cycle for low side driver pin 1
		dutyCycle0 – Duty cycle for low side driver pin 0
	Return:	n/a
seekDock()	Instructs the robot to begin looking for the home base and dock with it.	
	Arguments:	n/a
	Return:	n/a
demo(demoNumber)	Instructs the robot to run the requested demo. The possible values are as follows: -1 stop current demo 0 wander the surrounding area 1 wander and dock, when the docking station is seen 2 wander a more local area 3 wander to a wall and then follow along it 4 figure 8 5 "wimp" demo: when pushed, move forward when bumped, move back and away 6 home: will home in on a virtual wall, as long as the back and sides of the IR receiver are covered with tape 7 tag: homes in on sequential virtual walls 8 pachelbel: plays the first few notes of the *Canon in D* 9 banjo: plays chord notes according to its cliff sensors chord key is selected via the bumper	
	Arguments:	demoNumber – index of the demo to run
	Return:	n/a
toSafeMode()	Puts the robot into Safe Mode	
	Arguments:	n/a

	Return:	n/a
toFullMode()	Puts the robot into Full Mode	
	Arguments:	n/a
	Return:	n/a
getMode()	Returns the numerical value of the last known mode of the robot.	
	Arguments:	n/a
	Return:	Mode number
All wait commands do not allow new commands to run while the robot waits; but queue them for later execution.		
waitTime(seconds)	Instructs the robot to wait the requested number of seconds before executing the next command.	
	Arguments:	seconds – Number of seconds to wait
	Return:	n/a
waitEvent(eventNumber)	Waits for the specified event to happen before executing the next command	
	Arguments:	eventNumber – ID number of the event to wait for
	Return:	n/a

ADDITIONAL CLASSES

These are classes are used by the main Create class for exception and data handling.

Name	**Description**	
CommunicationError	Used as an exception for failures in serial communication with the Create robot	
SensorModule	Used to define the properties for a sensor, for querying the robot and storing the returned value. See the following "Sensor Access" section for more information.	
	Fields:	ID – the packet ID used for querying the robot interpret – How the raw sensor data will be interpreted before being returned size – The number of bytes the returned sensor data should be Data – the interpreted value of the sensor

PERTINENT MODULE LEVEL VARIABLES

This table describes the pertinent variables used in the library. Variables used internally are omitted.

Name	**Value**	**Description**
RADIUS	115	Radius of the Create robot in millimeters
OFF_MODE	0	Mapping for value returned by getMode()
PASSIVE_MODE	1	Mapping for value returned by getMode()
SAFE_MODE	2	Mapping for value returned by getMode()
FULL_MODE	3	Mapping for value returned by getMode()
SENSORS	-	List of SensorModule objects used to access the sensors on the Create (See the "Sensor Access" section for more information)

MODULE LEVEL FUNCTIONS

This table describes the create library's module-level functions.

Name	Description	
bytesOfR(r)	Prints the bytes of data in a sensor reply.	
	Arguments:	r – raw (binary) data from a sensor reply
	Return:	n/a
bitOfByte(bit, byte)	Returns the specified bit of a byte	
	Arguments:	bit – The index of the bit to return (0 to 7)
		byte – The byte from which to read the specified bit
	Return:	The specified bit (0 or 1), 0 if the request is out of range
toBinary(val,numBits)	Prints the specified number of bits of the given value in binary, starting with the least significant bit.	
	Arguments:	val – The value to print in binary
		numBits – The number of bits of the binary value to print
	Return:	n/a
fromBinary(s)	Creates a value from the given string of 0's and 1's	
	Arguments:	s – String consisting of 0's and 1's
	Return:	The value represented by the "binary" string passed in
twosComplement1byte(byte)	Returns the two's complement value of byte	
	Arguments:	byte – 1 byte value to use (0 to 255)
	Return:	byte value in two's complement (-128 to 127)
twosComplementInt2bytes(highByte, lowByte)	Returns the two's complement of the 2-byte value given	
	Arguments:	highByte – The upper byte of the two-byte value
		lowByte – The lower byte of the two-byte value
	Return:	Returns two's complement of given value (-32768 to 32767)
toTwosComplement2Bytes(value)	Returns the upper and lower bytes of the given value (interpreted in two's complement)	
	Arguments:	value – The value to read the 2 bytes of
	Return:	(upperByte, lowerByte), a tuple consisting of the bytes formed from the given value

USER IO

This section is intended for further information on the user I/O pins on the cargo bay connector of the Create. Documentation is not yet created for this, so please refer to the Create manual and the SCI manual for the time being. For more information on these documents and the use of the I/O, ask your professor. Improper connection to any of the IO pins can cause damage to the connected circuitry or to the robot itself ***(think sparks, smoke, and $$$ out of your pocket: don't use this feature yet!)***