

Matlab Toolbox for the iRobot Create (MTIC) Version 2.0 (8.25.2011)

Download at:

www.usna.edu/Users/weapsys/esposito/roomba.matlab/

Developed by:

Joel Esposito
Associate Professor
Systems Engineering
US Naval Academy
Annapolis, MD USA

- and -

Owen Barton
Joshua Koehler
David Lim
SEAP Interns

www.usna.edu/Users/weapsys/esposito/
esposito@usna.edu

Acknowledgements:

Joel Esposito thanks Patrick Brown for testing the commands on the Roomba platform. Owen Barton, Josh Koehler and David Lim acknowledge the financial support of the ONR SEAP program.

Citation Information:

The toolbox is free. If you publish research or pedagogical papers containing results gathered using the toolbox, please cite this document as follows.

Esposito, Joel M., Barton, Owen, Koehler, Joshua, Lim, David, "Matlab Toolbox for the Create Robot", www.usna.edu/Users/weapsys/esposito/roomba.matlab/, Copyright 2008-2011

Or in BibTex:

```
@MISC{MatlabCreate,  
author = {Joel M. Esposito and Owen Barton and Josh Koehler and David Lim},  
title = { Matlab Toolbox for the Create Robot },  
howpublished = { www.usna.edu/Users/weapsys/esposito/roomba.matlab/},  
year = {2011} }
```

MATLAB TOOLBOX FOR THE IROBOT CREATE (MTIC) VERSION 2.0 (8.25.2011)	1
MTIC PROJECT OVERVIEW	6
Background	6
Toolbox Features	6
New In Version 2.0 (8.5.2011) vs. Version 1	6
New In Version 1.0 (8.5.2009) vs. Beta Version	6
Requirements	7
Create:	7
Matlab:	7
Important Note About Matlab2010a:	7
Connection:	7
Command Module:	7
Downloading and Installation	8
Getting Started	8
Hardware Connection	8
Software Connection	8
Programming Tips	9
Help	9
When Things Don't Work Right	9
Roomba vs Create's Open interface	9
Streaming:	10
Timing:	10
AllSensorsReadRoomba vs. individual sensor commands:	10
Example:	10
FUNCTION LIST	11
Initialization	11
Reading Sensors	11
Sending Commands	12
DETAILED FUNCTION DESCRIPTIONS	12
RoombaInit	12
Syntax	12
Description	13
Note	13

Examples	13
AllSensorsReadRoomba.....	13
Syntax	14
Description.....	14
Note	14
.....	15
AngleSensorRoomba	15
Syntax	15
Description.....	15
Note	15
BatteryChargeReaderRoomba.....	15
Syntax	15
Description.....	15
BatteryVoltageRoomba.....	15
Syntax	16
Description.....	16
BumpWheelDropsSensorsRoomba	16
Syntax	16
Description.....	16
Note	16
ButtonSensorRoomba	16
Syntax	16
Description.....	17
CliffFrontLeftSensorRoomba.....	17
Syntax	17
Description.....	17
CliffFrontLeftSignalStrengthRoomba.....	17
Syntax	17
Description.....	17
Note	17
CliffFrontRightSensorRoomba	17
Syntax	18
Description.....	18
CliffFrontRightSignalStrengthRoomba	18
Syntax	18
Description.....	18
Note	18
CliffLeftSensorRoomba	18
Syntax	18
Description.....	18
CliffLeftSignalStrengthRoomba	19
Syntax	19

Description.....	19
Note	19
CliffRightSensorRoomba.....	19
Syntax	19
Description.....	19
CliffRightSignalStrengthRoomba.....	19
Syntax	20
Description.....	20
Note	20
CurrentTesterRoomba.....	20
Syntax	20
Description.....	20
Note	20
DemoCmdsCreate.....	20
Syntax	20
Description.....	21
DistanceSensorRoomba	21
Syntax	21
Description.....	21
Note	21
BeepRoomba	22
Syntax	22
Note	22
SetLEDsRoomba	22
Syntax	22
Description.....	22
SetFwdVelRadiusRoomba	23
Syntax	23
Description.....	23
Note	23
SetDriveWheelsCreate	23
Syntax	23
Description.....	23
Note	23
SetFwdAngVelCreate.....	24
Syntax	24
Description.....	24
Note	24
travelDist.....	24
Syntax	24
Description.....	24
Note	24
turnAngle	25

Syntax	25
Description.....	25
Note	25
RobotHardKeyboard	26
Syntax	26
Description.....	26
Notes.....	26
RobotGuiControl.....	26
Syntax	27
Description.....	27
Notes.....	27

MTIC Project Overview

Background

The iRobot Create is a reprogrammable version of the Roomba robot vacuum cleaner for robotics hobbyists, educators and researchers. Out of the package, users can use their PC to control the robot by sending numerical sentences over a serial connection.

Two obstacles to use, especially for educational purposes, are: (1) the cryptic nature of the command interface; and (2) the difficulty in establishing a software serial port connection to the robot. This Matlab toolbox seeks to overcome these difficulties.

Toolbox Features

The toolbox replaces the native low-level numerical commands, with a set of high level, intuitive, Matlab functions (aka "wrappers") that:

- Create a link between your PC and Create using your PC's serial port or Bluetooth connections.
- Provide a variety of user-friendly drive commands, calibrated in SI units.
- Read the bump and cliff sensors, determine distance driven, and battery life in SI units.
- Use the Matlab command line or script files to control the robot. Note that all code is developed, stored and executed on the PC base station -- not the Create.

New In Version 2.0 (8.5.2011) vs. Version 1

- Fixed bug in *CliffLeftSensorRoomba* and *CliffRightSensorRoomba* (previously checked front left cliff sensor)
- Enabled *SetLEDsRoomba* to set Play and Advance LEDs in all four possible combinations. (Previously could only have Play on and Advance off or vice versa. Now can also have both on or both off)
- Added *RobotHardKeyboard*. Roomba can be controlled via user inputs by computer keyboard.
- Added *RobotGuiControl*. Roomba can be controlled via user inputs by arrow display on computer screen. This is useful when laptop is attached to robot. One can remote desktop into laptop with smartphone and steer robot.

New In Version 1.0 (8.5.2009) vs. Beta Version

- Two new drive functions that make the robot travel a certain distance or through a certain angle then stop.
- Increased robustness of serial communications using try-catch statements
- All functions now return NaN (Matlab's not a number) if robot does not properly respond to sensor query

- Fixed bug in the wheel drop sensor function (did not previously read caster wheel drop correctly)
- Fixed bug in *SetFwdVelRadius* (did not go straight previously)
- Fixed typos in manual (especially syntax for *AllSensorsReadRoomba*)
- Initializes in FULL mode (not SAFE mode)
- Tested in Matlab 2008a

Requirements

Create: You need an iRobot Create. The version of the physical robot does not matter much, but the toolbox was developed for use with the [Create Open Interface](#) version 1.0. It has not been fully tested with the [Roomba Serial Interface](#) though it is quite similar.

Matlab: The toolbox was developed and tested with MATLAB Release 2008a, 2008b, 2009a, and 2010b though releases dating back to 2005 will likely work.

Important Note About Matlab2010a: There have been reports of toolbox errors using Matlab 2010a in conjunction with Bluetooth. The error messages say that `fwrite` or `fopen` fails. We were unable to replicate these errors as they may be specific to the Bluetooth driver. If you experience such problem, first test using a wired connection (no Bluetooth). Be sure your Bluetooth driver is up to date. Please forward us your bug reports and consider updating to Matlab 2010b+.

Connection: The toolbox assumes that you can establish a serial port connection between the Create and your PC, using either the Mini-DIN Connector cable that comes with Create, or the Bluetooth Adapter Module and Bluetooth USB Radio available from the iRobot website. Note that the Mini-Din is a wired connection, but it can be easily converted to a wireless one using a radio modem (ex. Xbee).

Command Module: If you are interested in writing programs that are stored and execute on the Create, rather than a base station PC, this toolbox is not applicable. Consider buying the Command Module, which lets you write C-programs to execute on board the Create.

Downloading and Installation

1. Download and extract the zip file from <http://www.usna.edu/Users/weapsys/esposito/roomba.matlab/>
2. Place all the files in a folder called CreateToolbox
3. In Matlab add this folder to the path. The path is the list of directories Matlab looks in for function definitions. For example, if you placed that folder in C:\Matlab:

```
>> addpath 'c:\matlab\CreateToolbox';
```

You will need to do Step 3 each time you restart Matlab. If you want the change to be permanent add:

```
>> savepath;
```

Getting Started

Hardware Connection

1. Connect your robot to your computer. Power it up.
2. Determine which communication port you are using. They are referred to as COM1, COM2, etc. For Windows XP try Control Panel / System / Hardware / Device Manager / Ports (COM & LPT) to see a list of available ports.
 - a. If you are using a desktop PC and a serial port, it is most likely COM1.
 - b. If you are using a laptop, you probably don't have a physical serial port. You can buy a USB to serial converter cable from any consumer electronics store. Your computer will assign the port number. Use the Control Panel as described above to determine the number.
 - c. If you are using a Bluetooth connection your computer will assign the port number. Use the Control Panel to determine the number.

Software Connection

Regardless of your physical connection (mini-DIN, USB-serial adapter, or Bluetooth), Matlab treats your connection as a serial port. Matlab manages serial port communications by creating something called "serial port objects". They are variables that contain all the port number and settings you need to communicate with the Roomba.

Create a serial port object: For example if you are using COM1:

```
>> [serialObject] = RoombaInit(1) ;
```

Important Notes:

- The function's input is an integer specifying the port name. Ex. `RoombaInit(1)` uses COM1.
- If the function correctly initialized the robot, you will see the two outboard light come on and hear a beep. If it failed, first check the com port number; then try power cycling the robot.
- Be sure the power cable is not plugged in. The function starts the robot in the FULL mode. That means you can drive it down the stairs on accident, drive it while its still plugged into the power outlet etc. Be careful. It is possible to change this to SAFE mode. See comments inside `RoombaInit`.
- The output is a serial port object that contains all the settings to talk to the robot. This variable must be passed to every command in the toolbox, each time they are called. If you type `clear all` you may inadvertently delete this variable. You will need to reinitialize with `RoombaInit` before you can send more commands to the robot.
- If you have many Roombas you would like to operate simultaneously, each must be connected to a different Com Port. Bluetooth makes that easy, but you can also use a USB hub or a serial hub. You will need a separate serial port object for each robot/port. Example:


```
>> [serialObjectRoomba1] = RoombaInit(1) ;
>> [serialObjectRoomba2] = RoombaInit(2) ;
```

Programming Tips

Help Once you initialize the connection, you can use the functions in this toolbox, either at the command line or in a script file, to send commands to the robot or read the robot's sensors. Finally you can use the *help* function to get more information about a particular function.

When Things Don't Work Right

In Version 1.0 and beyond, try-catch statements and buffer flushing are used to increase the robustness of the serial communications. That means that if you request a sensor reading or send a motor command to the robot and the robot does not reply appropriately:

- you will see a warning message but not get an error
- any values the function should have returned (eg. sensor readings) will be NaN (Matlab's data type for Not a Number)

The advantage of this is that a single bad sensor reading, for example if the robot drives out of range of the wireless communication device, will not crash your code. On the other hand the onus is on the programmer to check for bad sensor responses. The Matlab *isnan* function is useful to test for this.

Roomba vs Create's Open interface: The Roomba Open Interface is a subset of the Create Open Interface. That means that all the commands for Roomba work on Create; but not all the commands for Create work on the Roomba. **Any function in this toolbox with the word "Roomba" in the name works on the Roomba *and* on the**

Create. Any function with the word "Create" in the name works *only* with the Create.

Further, there are many commands in the Create Open Interface that were not included in the toolbox. It would seem some of the so-called Create Scripting commands are unnecessary if you are using Matlab. Also if you are interested in controlling the voltage of the pins in the cargo bay (because you want to add sensors or motors to the Create) you will need to consult the manual.

Streaming: Note that the Create has a mode where it streams sensor data over the serial cable continuously. This toolbox does not support that mode. You must send one of the sensor request commands each time you want to read data.

Timing: The PC and Matlab communicate at 57600 baud. That means there is a limit as to how quickly the robot can send, receive and respond to commands.

Tip: *If you find your programs crash quite frequently, you can add a small time delay after each command to give the robot time to process the instructions. Inside RoombaInit set td as small value (seconds): ex. td = 0.05.*

AllSensorsReadRoomba vs. individual sensor commands:

There are two ways you can read the sensors.

- **Method 1** is to use the AllSensorsReadRoomba function. This returns the values of all the sensors in one call.
- **Method 2** is to use the individual sensor commands (AngleSensorRoomba, BatteryVoltageRoomba, etc).

Method 1 AllSensorsReadRoomba (Pro) is faster than multiple calls to individual sensors. (Con) Each time you read the distance and angle sensor they return the change since the last time they were read, and then Create resets the encoder values. Therefore if you only want to read the bumper values, calling AllSensorsReadRoomba will inadvertently reset the distance and angular measurements. You need to track those values in your code if you are trying to navigate via dead reckoning.

Method 2 individual calls (Pro) is less cumbersome to call. And there is no risk of resetting the distance and angle measurements. (Con) It is slow if you need to read many sensors. Also, with this approach you are not reading the sensors *simultaneously*. For example if you call DistanceSensor, then AngleSensor while the robot is driving, you need to account for the fact that the two measurements were not taken at the same time and the robot moved in the time between the two calls.

Example: Here is a Matlab Program that turns on the robot, uses COM1, drives forward in an arc for 1 second, then reads the encoders to determine how far it has gone:

```
% Initialize communication
[serialObject] = RoombaInit(1)
```

```

% Read distance sensor (provides baseline)
InitDistance = DistanceSensorRoomba(serialObject);

%sets forward velocity 1 m/s and turning radius 2 m
SetFwdVelRadiusRoomba(serialObject, 1, 2);

%wait 1 second
pause(1)

% stop the robot
SetFwdVelRadiusRoomba(serialObject, 0, 1);

% read the distance sensor.
% returns dist since last reading in meters
Distance = DistanceSensorRoomba(serialObject)

```

Function List

Initialization

[RoombaInit](#)

Initializes serial port for use with Roomba

Reading Sensors

[AllSensorsReadRoomba](#)

Read all the sensors on the robot at once

[AngleSensorRoomba](#)

Angle that Roomba has turned

[BatteryChargeReaderRoomba](#)

Battery charge remaining

[BatteryVoltageRoomba](#)

Indicates battery voltage

[BumpsWheelDropsSensorsRoomba](#)

State of bump and wheel drop sensors

[ButtonSensorRoomba](#)

State of buttons onboard Roomba

[CliffFrontLeftSensorRoomba](#)

State of cliff front left sensor

[CliffFrontLeftSignalStrengthRoomba](#)

Strength of front left cliff sensor signal

[CliffFrontRightSensorRoomba](#)

State of cliff front right sensor

[CliffFrontRightSignalStrengthRoomba](#)

Strength of front right cliff sensor signal

[CliffLeftSensorRoomba](#)

State of cliff left sensor

[CliffLeftSignalStrengthRoomba](#)

Strength of left cliff sensor signal

<u>CliffRightSensorRoomba</u>	State of cliff right sensor
<u>CliffRightSignalStrengthRoomba</u>	Strength of right cliff sensor signal
<u>CurrentTesterRoomba</u>	Current flowing in/out of battery
<u>DistanceSensorRoomba</u>	Distance traveled

Sending Commands

<u>DemoCmdsCreate</u>	Plays built-in demos
<u>LEDsRoomba</u>	Manipulates LEDs
<u>SetFwdVelRadiusRoomba</u>	Drives by setting forward velocity and turning radius
<u>SetDriveWheelsCreate</u>	Drives by setting left right wheel linear velocity
<u>SetFwdVelAngVelCreate</u>	Drives by setting forward and angular velocity
<u>BeepRoomba</u>	Makes Roomba Beep
<u>travelDist</u>	Travels a specified distance
<u>turnAngle</u>	Turns a specified angle

User Control

<u>RobotHardKeyBoard</u>	Moves the Roomba using the keyboard with preset distances and turning angles
<u>RobotGuiControl</u>	Moves the Roomba using an arrow display on the computer screen with preset distances and turning angles

Detailed Function Descriptions

RoombaInit

Initializes serial port for use with Create. May need to modify for use with Roomba

Syntax

```
[serPort]= RoombaInit(portNum)
```

Description

Returns an initialized serial port object in Matlab, with baud rate 57600, terminator is a line feed, tagged 'Roomba'. The input is an integer specifying which com port you are using. The serPort object is then passed to other commands in the toolbox as an input.

Note

- This now initialized the robot in FULL mode which means you can drive robot while plugged in, off a cliff, etc. See comments in code if you want to start up in SAFE mode.
- If the function correctly initialized the robot you will see the two outboard light come on and a message displayed indicating that the initialization was successful. If it failed, first check the com port number; then try power cycling the robot.
- The function's input must be an integer.
- The output is a serial port object that contains all the settings to talk to the robot. This variable must be passed to every command in the toolbox, each time they are called. If you type `clear all` you may inadvertently delete this variable. You can recover this by using `serPort = instrfind()`.
- If you have many robots you would like to operate simultaneously, each must be connected to a different Com Port. You will need a separate serial port object for each robot/port.
- Baudrate is set to a default of 57600. Other baud rates can be forced (see Create SCI).
- Inside the code you can add a small time delay to be executed after every command in the tool box, by setting the variable `td` to something other than zero. This is useful if your code crashes frequently. Ex. `td = 0.05; %pause 1/20 of a second after every command in tool box`
- If you are having trouble getting this command to work for older Roombas (not the Create), try commenting out the code to set the LEDs and play the song.

Examples

```
[serPort]= RoombaInit(1) Initializes serPort to the COM1 port. For example, if  
[serPort]= RoombaInit(2) serPort would be initialized to COM2.
```

AllSensorsReadRoomba

Returns readings from all sensors

Syntax

```
[BumpRight, BumpLeft, BumpFront, Wall, virtWall, CliffLft, ...  
CliffRgt, CliffFrntLft, CliffFrntRgt, LeftCurrOver, RightCurrOver, ...  
DirtL, DirtR, ButtonPlay, ButtonAdv, Dist, Angle, ...  
Volts, Current, Temp, Charge, Capacity, pCharge] =  
AllSensorsReadRoomba(serPort);
```

Description

Reads all of Roomba Sensors at once. Pass an initialized serial port object. Returned values are (0 is false 1 is true):

- BumpRight (0/1),
- BumpLeft(0/1),
- BumpFront(0/1),
- Wall(0/1),
- virtWall(0/1),
- CliffLft(0/1),
- CliffRgt(0/1),
- CliffFrntLft(0/1),
- CliffFrntRgt(0/1),
- LeftCurrOver (0/1),
- RightCurrOver(0/1), ...
- DirtL(0/1),
- DirtR(0/1),
- ButtonPlay(0/1),
- ButtonAdv(0/1),
- Dist (meters since last call),
- Angle (rad since last call),
- Volts (V),
- Current (Amps, positive is charging),
- Temp (celcius),
- Charge (milliamphours),
- Capacity (milliamphours),
- pCharge (percent Charge)

Can add others if you like, see code

Note

If you need to read multiple sensors at once, this has a faster runtime than multiple individual calls to single sensor functions. Beware however that each time you call this function it resets the count for distance and angle sensors. They return distance traveled since last called. So if you are dead reckoning you need to track the accumulated values of those variables.

AngleSensorRoomba

Angle that Roomba has turned since the last reading

Syntax

```
[AngleR]=AngleSensorRoomba(serPort)
```

Description

Reads Displays the angle in radians that Create has turned since the angle was last requested. The variable serPort is the serial port which connects the Roomba to the pc.

Note

Counter-clockwise angles are positive and clockwise angles are negative. Can overflow if not polled frequently enough. Capped at + / - 571 radians.

BatteryChargeReaderRoomba

Battery charge remaining in Create's battery

Syntax

```
[Charge, Capacity, Percent] =BatteryChargeReaderRoomba(serPort)
```

Description

Gives the charge remaining (milliAmpHours), total battery capacity (mAH) and percent remaining in Create's Battery

BatteryVoltageRoomba

Indicates voltage in Create's battery

Syntax

```
[Voltage]=BatteryVoltageRoomba(serPort)
```

Description

The voltage of Create's battery is displayed in Volts

BumpWheelDropsSensorsRoomba

State of the bump and wheel drop sensors

Syntax

```
[BumpRight,BumpLeft,WheDropRight,WheDropLeft,WheDropCaster,BumpFront] =  
BumpsWheelDropsSensorsRoomba(serPort)
```

Description

Specifies the state of the bump and wheel drop sensors, either triggered (1) or not triggered (0).

Note

BumpFront is triggered when both the BumpRight and BumpLeft sensors are triggered. If so, Left and Right return false (0), and front returns true (1).

ButtonSensorRoomba

State of the buttons onboard Roomba

Syntax

```
[ButtonAdv,ButtonPlay] = ButtonsSensorRoomba(serPort)
```

Description

Displays the state of Create's Play and Advance buttons, either pressed (1) or not pressed (0).

CliffFrontLeftSensorRoomba

State of the cliff front left sensor

Syntax

```
[state] = CliffFrontLeftSensorRoomba(serPort)
```

Description

Displays the state of the cliff front left sensor, either triggered or not triggered.

CliffFrontLeftSignalStrengthRoomba

Strength of the front left cliff sensor signal

Syntax

```
[strg] = CliffFrontLeftSignalStrengthRoomba(serPort)
```

Description

Displays the percent strength of the front left cliff sensor signal.

Note

Ranges between 0-100 % signal.

CliffFrontRightSensorRoomba

State of the cliff front right sensor

Syntax

```
[state] = CliffFrontRightSensorRoomba(serPort)
```

Description

Displays the state of the cliff front right sensor, either triggered(1) or not triggered(0).

CliffFrontRightSignalStrengthRoomba

Strength of the front right cliff sensor signal

Syntax

```
[strg] = CliffFrontRightSignalStrengthRoomba(serPort)
```

Description

Displays the percent strength of the front right cliff sensor signal.

Note

Ranges between 0-100 % signal.

CliffLeftSensorRoomba

State of the cliff front left sensor

Syntax

```
[state] = CliffLeftSensorRoomba(serPort)
```

Description

Displays the state of the cliff left sensor, either triggered or not triggered.

CliffLeftSignalStrengthRoomba

Strength of the front left cliff sensor signal

Syntax

```
[strg] = CliffLeftSignalStrengthRoomba(serPort)
```

Description

Displays the percent strength of the left cliff sensor signal.

Note

Ranges between 0-100 % signal.

CliffRightSensorRoomba

State of the cliff front left sensor

Syntax

```
[state] = CliffRightSensorRoomba(serPort)
```

Description

Displays the state of the cliff right sensor, either triggered or not triggered.

CliffRightSignalStrengthRoomba

Strength of the front left cliff sensor signal

Syntax

```
[strg] = CliffRightSignalStrengthRoomba(serPort)
```

Description

Displays the percent strength of the right cliff sensor signal.

Note

Ranges between 0-100 % signal.

CurrentTesterRoomba

Current flowing in/out of battery

Syntax

```
[Current] = CurrentTesterRoomba(serPort)
```

Description

Displays the current (in amps) flowing into or out of Create's battery.

Note

Negative currents indicate that current is flowing out of the battery. Positive currents indicate that current is flowing into the battery (charging).

DemoCmdsCreate

Plays built-in demos

Syntax

```
[] = DemoCmdsCreate(serPort, DemoNum)
```

Description

The Create performs demos according to the `DemoNum` entered in the syntax. For example, `DemoCmdsCreate(serPort, 0)` performs the demo which corresponds to `DemoNum 0`. In this demo, the Create attempts to cover the room with a combination of different behaviors. There are 10 other demos built into the Create which correspond to different `DemoNum` values.

- 1: Aborts the current demo
- 1: Attempts to cover entire room unless Create sees a signal from home base, it uses that signal to dock with the home base and recharge.
- 2: Covers an area around its starting position by spiraling outward then inward
- 3: Searches for a wall, once a wall is found, create drives along the wall, around the circumference of the room
- 4: Drives in a figure 8 pattern
- 5: Drives forward when pushed from behind. If Create hits an obstacle while driving, it drives away from the obstacle
- 6: Drives towards an iRobot Virtual Wall
- 7: Drives into multiple virtual walls by bumping into one, turning around and bumping into the next virtual wall.
- 8: Plays the notes of Pachelbel's Cannon in sequence when cliff sensors are activated.
- 9: Plays a note of a chord for each of its cliff sensors, and a chord for each bumper You can play the robot like banjo.

DistanceSensorRoomba

Distance traveled since function was last called.

Syntax

```
[Distance] = DistanceSensorRoomba(serPort)
```

Description

Gives the distance traveled in meters since last requested.

Note

Positive values indicate travel in the forward direction. Negative values indicate travel in the reverse direction. If not polled frequently enough, it is capped at its minimum or maximum at 32.768 or -32.768.

BeepRoomba

Makes a beeping sound. The note and duration is specified in RoombaInit

Syntax

```
[ ] = BeepRoomba(serPort)
```

Note

The default beep for the toolbox is $20/64^{\text{th}}$ of a second long. A sequence of BeepRoomba commands, executed more frequently than this is likely to result in missed beeps. For example, the following will only produce a single audible beep.

```
>> BeepRoomba(serRoomba); BeepRoomba(serRoomba);
```

A proper double beep can be made using a pause command as follows.

```
>> BeepRoomba(serRoomba); pause(.75); BeepRoomba(serRoomba);
```

The pause should be at least equal to the duration of the note ($20/64^{\text{th}}$ sec) plus the latency of the wireless network (likely non-deterministic) .

Note this problem is specific to the with beeps function, due to the way the low-level speaker drivers work. For example, this sequence will execute properly with no pauses.

```
>> BeepRoomba(serRoomba); SetFwdVelRadiusRoomba(serRoomba, FwdVel, Radius);
```

SetLEDsRoomba

Manipulates LEDs on top of robot

Syntax

```
[ ] = SetLEDsRoomba(serPort, LED, Color, Intensity)
```

Description

Play and Advance LEDs are only on/off and are controlled by LED argument: LED = 0 (off), 1 to turn Play green or 2 to turn Advance green, and 3 to turn both green.

Power LED controlled by last two arguments:

Color determines which color(Red/Green) that the Power LED will illuminate as, from 0-100% 0 is pure green, 100 is pure red. Intensity determines how bright the Power LED appears from 1-100%

SetFwdVelRadiusRoomba

Drives by specifying forward velocity and turning radius

Syntax

```
[ ] = SetFwdVelRadiusRoomba(serPort, FwdVel, Radius)
```

Description

Moves Roomba by setting forward velocity and turn radius. serPort is a serial port object created by Roombainit. FwdVel is forward vel in m/sec [-0.5, 0.5]. Radius in meters, positive turns CCW, negative turns CW. Valid range is [-2,2]. Code saturates overflows.

Note

Special cases: Straight = inf (matlab's infinity)

Turn in place clockwise = -eps (eps is matlabs predefined smallest real number)

Turn in place counter-clockwise = eps (eps is matlabs predefined smallest real number)

SetDriveWheelsCreate

Specify linear velocity of right wheel and left wheel.

Syntax

```
[ ] = SetDriveWheelsCreate(serPort, rightWheel, leftWheel)
```

Description

Specify linear velocity of right wheel and left wheel in meters/ sec [-0.5, 0.5]. Values outside that range are capped. Negative velocity is backward.

Note

Note that if you prefer to specify ANGULAR velocity of wheels (omega in rad/sec) you must determine radius of wheel (r in meters). Ex: rightWheelVel =omegaRight*r

SetFwdAngVelCreate

Specify forward and angular velocity.

Syntax

```
[ ] = SetFwdVelAngVelCreate(serPort, FwdVel, AngVel )
```

Description

Specify forward velocity in meters/ sec [-0.5, 0.5]. Specify Angular Velocity in rad/sec. Negative velocity is backward/Clockwise. Caps overflow.

Note

Wheel speeds are capped at .5 meters per second. So it is possible to combinations of forward speeds and angular velocities that cannot be achieved simultaneously. In that case a warning is displayed. Results are a function of wheel base (width between left and right wheel assumed to be 0.258 Meters)

travelDist

Specify a distance to travel and a speed. Stop when done.

Syntax

```
travelDist(serPort, speed, distance)
```

Description

Moves the Create the distance entered in meters. Positive distances move the Create forward, negative distances move the Create backwards. Speed should be between 0.025 and 0.5 m/s.

Note

In general travelDist and turn Angle are the only commands in the toolbox that use scripting from the Create Open Interface. This means that the Create waits (it will not accept any new commands) until it thinks it has moved the requested distance. When done it displays a termination message. If for some reason the Create is unable to move it may wait forever causing the robot to become unresponsive. If this happens the robot must be power cycled (off-on).

To that end, Speeds below a certain threshold cause the Create to fail to move. Therefore for requested speeds below 0.025 m/s the function rounds them up to 0.025 m/s. Speeds are capped at 0.5

Note, the direction is derived from distance. If a negative speed is entered, function uses absolute value.

turnAngle

Specify an angle to turn though (in degrees) and a speed (in rad/sec). Robot turns in place and stops when done.

Syntax

```
turnAngle(serPort, speed, turnAngle)
```

Description

Turns the Create by *turnAngle* (degrees [-360, 360]) and stops when done. Positive is counter clockwise. Negative is clockwise. Speed is in rad/sec (between 0 and 0.2).

Note

In general *travelDist* and *turnAngle* are the only commands in the toolbox that use scripting from the Create Open Interface. This means that the Create waits (it will not accept any new commands) until it thinks it has moved the requested distance. When done it displays a termination message. If for some reason the Create is unable to move it may wait forever causing the robot to become unresponsive. If this happens the robot must be power cycled (off-on).

To that end, speeds below a certain threshold cause the Create to fail to move. Therefore for requested speeds below 0.025 m/s the function rounds them up to 0.025 m/s. Speeds are capped at 0.2.

Note that the direction is derived solely from the angle. If a negative speed is entered, the absolute value is used.

The command attempts to turn the shortest way around the circle. For example a turn angle of 350 degrees will result in the robot actually turning -10 degrees. So, positive *turnAngle* up to 180 degrees and negative *turnAngle* from -180 to -360 degrees turns the Create counterclockwise. Negative *turnAngle* up to -180 degrees and positive *turnAngle* from 180 to 360 degrees turns the Create clockwise.

Similarly an angle of 370 will actually only turn 10 degrees.

RobotHardKeyBoard

Allows user control of robot via computer keyboard.

Syntax

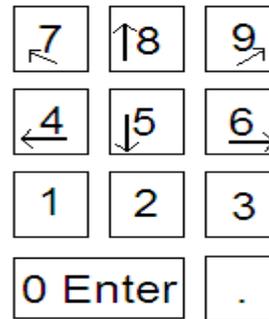
```
[keyPresses , elapsedTime] = RobotHardKeyBoard(CommPort)
```

Description

Roomba can be controlled through either the numerical keypad or alphabetical keys (more practical for laptop users). The given presets allow incremental movements that can be used to turn or move the robot. Pressing q exits the keyboard control mode.

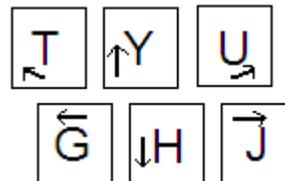
Numerical keypad presets:

- 8-moves Roomba forward 10cm
- 5-moves Roomba backward 5cm
- 4-turns Roomba left 15 degrees
- 7-slightly turns Roomba left 5 degrees
- 6-turns Roomba right 15 degrees
- 9-slightly turns Roomba right 5 degrees



Alphabetical key presets:

- y-moves Roomba forward 10cm
- h-moves Roomba backward 5cm
- g-turns Roomba left 15 degrees
- t-slightly turns Roomba left 5 degrees
- j-turns Roomba right 15 degrees
- u-slightly turns Roomba right 5 degrees



Notes

1. Preset keys and variables such as moving distance, turning angles, and Roomba speed can be changed inside this function to suit the user's preference.
2. Attention must be on the figure window to properly catch the keystrokes.

RobotGuiControl

Allows user control of robot via arrow display on the computer screen.

Syntax

```
varargout = RobotGuiControl(varargin)
```

Description

Roomba can be controlled by the arrows on the computer screen. Identical to *RobotHardKeyBoard*, the given presets allow incremental movements that can be used to turn or move the robot.

Arrow controls:

Large Forward Arrow-Moves Roomba forward 10cm

Small Forward Arrow-Moves Roomba forward 5cm

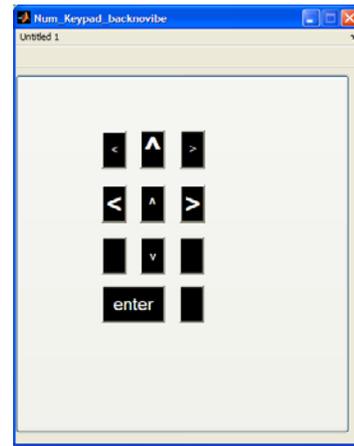
Back Arrow-Moves Roomba backward 5cm

Large Left Arrow-Turns Roomba 15 degrees left

Small Left Arrow-Turns Roomba 5 degrees left

Large Right Arrow-Turns Roomba 15 degrees right

Small Right Arrow-Turns Roomba 5 degrees right



Notes

1. Preset keys and variables such as moving distance, turning angles, and Roomba speed can be changed to the user's preference.
2. GUI can be used from a touch screen mobile device by making a remote desktop connection from mobile device to PC while running MTIC.

