CSCE 274 – Robotics Design and Applications – Fall 2019

Notes on the iRobot Create 2

This document contains some details on how to use the iRobot Create 2 robots.

1. Important Documents

These notes are intended to help you get started, but are not even close to being a replacement for the real documentation of the robot. The most important reference to consult is called:

iRobot® Create® 2 Open Interface (OI)

http://www.irobotweb.com/~/media/MainSite/PDFs/About/STEM/Create/iRobot_Roomba_600_Open_Interface_Spec.pdf?la=en

2. Lab facilities

The robots are stored in Swearingen 1D49. Access to this room is controlled using a combination lock, for which the combination will be announced through https://dropbox.cse.sc.edu.

As said during the class, the best way to work with the robots is to use your own laptops to write the program and then upload it to the Raspberry Pi mounted on the robots. The desktop computers in the lab, which can be accessed with your usual CSE department credentials, can also be used to write the program (contact Ryan Austin (Systems Administrator) if you have trouble logging in). However, to load the program to the Raspberry Pi an external laptop should be used.

Please ensure that the robots batteries are charged, by returning each robot to the charging station when you're done working with it.

3. Controller?

As shown in the class, the Create 2 robot itself has no user-accessible computer. Instead, it has a Raspberry Pi B+ mounted on top of it, which is connected to the robot by a serial cable. As such, your programs need to be uploaded and run on the Raspberry Pi through your computer, which will be connected with the provided LAN cable.

4. Writing programs to control the Robot

Here are the basic steps to write and execute a program that is going to be running on the Raspberry Pi:

- Use your favorite text editor to write or modify a Python program.
- Connect the micro USB to the related port on the Raspberry Pi to power it.
- Connect your computer to the Raspberry Pi with the provided LAN cable.
- Copy the program to the Raspberry Pi by using scp (IP address of the Raspberry Pi: `192.168.1.1`; username: `pi`; password: `raspberry`).
  e.g., using a Windows-based machine, you can use https://winscp.net/eng/download.php application to transfer the data from your laptop to your computer
  OR
  using a Unix-based machine, to copy a folder called `project_01` in the home directory of the Raspberry Pi

```
scp -r project_01 pi@192.168.1.1:~/
```

- Connect to the Raspberry Pi by using ssh.
  e.g., using a Windows-based machine, you can use PuTTY to access the Raspberry Pi
  http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  OR
  using a Unix-based machine,

  ```
  ssh pi@192.168.1.1
  ```
  and input the password.
- Use `screen` to start a shell window within the ssh session so that the shell is active even through network disruptions.
- Run your program using `python` interpreter.
  e.g.,

  ```
  python my_program.py
  ```
- Unplug the LAN cable from your computer.
- Press the power button to start the robot behavior (this should be checked inside your program).

To stop the robot:

- Press the power button (this should be checked inside your program).
- Connect your computer to the Raspberry Pi with the provided LAN cable.
- Connect to the Raspberry Pi by using `ssh`, as above.
- Type `screen -r` to reconnect to the shell that is running your program.
- Terminate your program by pressing Ctrl+c.
- Turn off the Raspberry Pi: `sudo halt`.
- Wait 1 minute and disconnect the micro USB from the powering of the Raspberry Pi.
- Place the robot back to the recharging station, by paying attention that the pads on the robot and on the dock are touching and thus the robot is actually charging; this can be observed if the Power button LED is blinking.

The program should be written by you and the code should be following an adequate and consistent coding style. In particular, the following coding style from Google is adopted https://google.github.io/styleguide/pyguide.html. This general guidelines should drive your code writing:

- **Don't repeat yourself.** Source code in which the same information appears in multiple places can be difficult to understand and, therefore, to debug.
  a) If you find yourself repeatedly copying and pasting a block of code, you should probably write a function to abstract that block.
  b) If your code contains explicit "magic numbers" that are repeated or copied from the documentation, you should probably replace those explicit numbers with named constants.
- **Write for your audience.** The "audience" for your source code includes, as a minimum (a) you and your group members, (b) your instructor, and (c) your interpreter. Therefore, your goal should be to write code that is easy for humans to read and understand. Important steps toward achieving this goal include:
  a) Using names that accurately describe the data or code that they name.

b) Writing comments for each function and section of code explaining its purpose and clarifying anything that's not obvious about how it works.
c) Using a consistent style for indentation and spacing, including indentation of nested blocks.

Lastly, please don't fall into the trap of writing bad code with the intention of "fixing" it to meet these general guidelines after it works. By doing that, you'll experience the pain that goes with these constraints but miss out on the benefits.

The use of external software is allowed only if:

- The use of it does not trivialize the assignment.
- The source should be clearly reported.

N.B.: Please remember to always **backup** your code, the best way being using a version control system (e.g., git, svn, or hg)!

N.B.2: As a suggestion, ensure that you modify the code in one place, e.g., your laptop, and synchronize it with the Raspberry Pi on the robot. If you modify the code on the Raspberry Pi, you might have inconsistent copies of your code across the computers and possibly it could lead to overwriting the files and thus losing important parts of code.

5. Communicating with the robot

As the robot provides a serial communication and Python language is going to be used in the class, the module pyserial (http://pythonhosted.org/pyserial/) will be used for communicating with the robot.

As shown in the class, once the module is imported

```
import serial
```

the main steps for communicating with the robot are:

1. Establish a serial connection, e.g., assuming that the serial device is at /dev/ttyUSB0

```
connection = serial.Serial('/dev/ttyUSB0', baudrate=115200)
```

2. Sending commands, e.g., START command 128 (remember that it should be encoded as ASCII character)

```
connection.write(chr(128))
```

3. Reading data, e.g., 1 byte

```
connection.read(1)
```

Remember when reading data, to look at the documentation of the robot to ensure that the information inside the packets are decoded properly, by, for example:

6. Interpreting strings as packed binary data, using the module `struct` (https://docs.python.org/2/library/struct.html)
    - e.g., reading and interpreting as byte the data from the bumper and the wheel drop sensor:

```
# Sending a request for Bumps and Wheel Drops sensor packet.
connection.write(chr(142)+chr(7))
# Reading 1 byte.
data = connection.read(1)
# Interpreting the data as byte.
import struct
byte = struct.unpack('B', data)[0]
```

7. Using bitwise operations to read each bit.
    - e.g., to find drop left state of the packet with ID 7, i.e., bumps and wheel drops:

```
bool(byte & 0x08)
```

Ensure also that each time you initialize the robot, you set it in Passive mode (opcode 128) and then to Safe mode (opcode 131).

The use of threads and timers can be useful to "parallelize" the tasks that handle, e.g., the sensors and the actuators. In particular, the module `threading` (https://docs.python.org/2/library/threading.html) and its following classes can be used (see the documentation for an example):

8. `Thread` (https://docs.python.org/2/library/threading.html#thread-objects), that can be used to run instructions in parallel.
   An example can be found here https://pymotw.com/2/threading/#thread-objects
9. `Timer` (https://docs.python.org/2/library/threading.html#timer-objects), that can be used to run instructions after the timer expires.
   An example can be found here https://pymotw.com/2/threading/#timer-threads
10. `Lock` (https://docs.python.org/2/library/threading.html#lock-objects), that can be used to control the access to shared resources.
    An example can be found here https://pymotw.com/2/threading/#controlling-access-to-resources

Note that, as the Raspberry Pi B+ has a single core computing unit, multiple tasks cannot be run in parallel. However, multi-tasking can be achieved by setting different timings.

For more information, please refer to the documentation of the robot and of the Python modules.

6. Common issues

Issue	Possible problem	Possible solution
The robot is not responding to commands.	1. The robot is off. 2. The robot is not started. 3. The command is not properly formatted.	1. Turn on the robot. 2. Set the robot in passive and safe mode. 3. Read the documentation to properly format the command. N.B.: Ensure that you wait enough time (0.5 seconds seem to work) between commands.
The sensor data is scrambled.	1. The serial connection is not properly set.	1. Close the serial connection with the function close of the Serial class and open it again.

	2. The robot is not initialized. 3. The sensor data is not properly interpreted.	2. Set the robot in passive and safe mode. 3. Read the documentation to properly interpret the data.