TDDD63 Project: NXT Lego Mindstorms

Jonathan Doherty

September 30, 2013

Contents

1 Introduction

In this project you will learn how to program your own real robot that can move around, react and interact with the world, the basic fundamentals of every robot in the world.

You will learn how to use touch sensors to detect objects in the world. You will detect colors and light using the color sensor, making it possible to follow lines, and you will beam out different colors to show the world what mode your robot is in. You will measure distances between objects with the ultrasonic sensor. Lastly using the three servo motors you will be able to navigate the world, pickup objects and shoot things at intruders.

By combining all of the above you will at the end of the project have the knowledge to build your own robot to do whatever pleases you, be it world domination or a new cuddly friend to your pet. You will also understand the strengths and weaknesses of robots and why robots and Skynet will not take over the world by themselves (at least not anytime soon).

To make this possible you will be using a new generation robotics kit from Lego Mindstorms called NXT 2.0, but instead of using the default limited program that ships with it, you will be using Python to program it directly giving you full control over everything.

2 Background

What is actually a robot? The answer depends who you ask and there is actually no one definition of robot. Joseph Engelberger, a pioneer in industrial robotics, once remarked: "I can't define a robot, but I know one when I see one."

But you can generally split up robots in three categories: autonomous robots that do not need human interaction to work, semi-autonomous robots that need some interaction and remotely controlled robots that need interaction. In this course we will be working with semi-autonomous robots because we have to change batteries to keep the robot going.

When you think about robots you often think about physical robots like industrial robots and NASA Mars robots, but robots can also be virtual software agents. Google for example uses virtual software agents that move around on the web collecting information about web pages. But in this course we will focus more on the physical Wall-E kind of robots that move around and perform tasks.

The robot you are building and programming here is the same type of robot used in the world right now. The main difference is the real world robots have more advanced parts making it possible to achieve even greater tasks but the concept and idea is still the same.

There is a lot of research about robots and in the future we will see a lot more robots entering areas beside the standard industrial area. At home we already see vacuum cleaner and lawn mower robots. In hospitals new robots are being developed that will perform challenging operations and help out with simple tasks. The list goes on and on and we will see more robots than ever within the nearest future.

One of the main advantages that robots have is that they can function in places where humans cannot, this makes them excellent in dangerous areas and workplaces. Linköping university, for example, has bleeding edge technology and research when it comes to unmanned aerial vehicles (UAV). They have created UAV helicopters that have the ability to fly and perform missions all by them themselves, which can be used in accident areas where dangerous chemicals make it impossible for humans to come near and similar scenarios.

3 Technical Information

This part contains basic information about the Lego Mindstorms NXT 2.0 and the files that come with the project. It also contains installation notes if you want to install and work with the project on your own computer.

The Lego Mindstorms NXT 2.0 package main parts:

- 1 NXT micro-computer - that acts as the brain of the robot
- 2 Touch Sensors - that makes the robot feel
- 1 Ultrasonic Sensor - that makes the robot "see" - and detect motion
- 1 Color Sensor - that can detect different colors, light settings and acts as a lamp
- 3 Interactive servo motors with built-in rotation sensors
- 7 connector cables for linking motors and sensors to the NXT

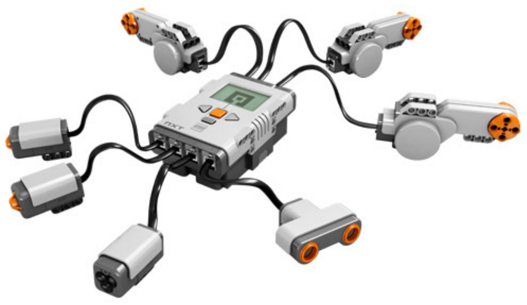

Figure 1: Main parts.

3.1 Package description

Installation folder:

- Contains all the files needed to install nxt-python on your computer.

Code folder:

- Contains all python files that are needed for the exercises.

Documentation folder:

- Contains all building instructions and documentation needed for the exercises.

3.2 Installation

This section contains installation notes for installing the software on the university computer. We also provide instructions for installing on your personal computer if you prefer, though this is optional. Apart from the PC-PUL installation, the instructions assume that you are root (administrator) and that you are using the command line/terminal.

All the files and folders that are needed below can be downloaded from `http://www.ida.liu.se/~TDDD63/projects/mindstorms/files`.

NOTE: If you do not understand or get problems with this, please don't hesitate to contact an assistant to help you out.

3.2.1 University Computer

At the university you will be working in IDA's Windows PC PULs, as this is necessary in order for you to be able to connect to the Lego Mindstorms kits using Bluetooth.

Step 1 - Settings:

1. Unpack the downloaded file Legomindstorm
2. Open Legomindstorm\Installation\folder.
3. Double-click "python-2.6.msi" and install it in Z:\python26
4. Open CMD and type:
5. python
6. If you look at the version it will say Python 2.7.3 this is the default version that will be started every time you use/login on the university computer, to change this temporary and start python 2.6 when typing python in the CMD we will have to change the PATH (press CTRL-Z to exit python).
7. Type:
8. set PATH=Z:\python26\;%PATH%
9. If you type python again it will now say Python 2.6 which we want (press CTRL-Z to exit python again).
10. Then type:
11. cd /D Z:
12. This will change from the C: drive to the Z: which is your home directory, the same directory that is known as ˜ on the Sun computers. We do this so we can use the CMD to access the Legomindstorm file you downloaded.
13. Continue with the steps below with the same CMD, because if you close the CMD or open a new one up you will have to retype the above instructions

because they are only temporary to this CMD instance. This is important to remember because the next time you log in and if you choose to use the CMD to run python programs(There are other ways you can do it like using the python 2.6 IDLE or eclipse with python 2.6 as the interpreter etc) you will need to retype the above steps in the CMD to get access to the right python version.

Step 2 - NXT-Python:

1. cd to Legomindstorm\Installation\University\nxt-python-2.2.2\then type:.
2. python setup.py install

Step 3 - Bluetooth

1. Open the Legomindstorm\Installation\University\folder and then copy the bluetooth folder to Z:\python26\Lib\site-packages\

3.2.2 Mac

This guide assumes your running Mountain Lion 10.8.4 and use the default python installation(2.7.3) that ships with the os.

Step 1 - NXT-Python:

1. Open terminal and cd to Legomindstorm/Installation/Mac/nxt-python-2.2.2/ then write the command below.
2. sudo python setup.py install

Step 2 - Bluetooth

1. Open terminal and cd to Legomindstorm/Installation/Mac/lightblue-0.4-master/ then write the command below.
2. sudo python setup.py install

Both of these steps will copy/install the modules to /Library/Python/2.7/site-packages/ so you can access them when using the default python installation 2.7.3.

3.2.3 Windows

Installing on your own Windows computer is easier than installing in the PC-PULs, as we assume you have full administrative rights. Please download and install Python 2.6 or use Legomindstorm/Installation/python-2.6.msi, and then:

Step 1 - NXT-Python:

1. Open Legomindstorm/Installation/Windows/nxt-python-2.2.2/ folder.
2. Double-click on install.bat

Step 2 - Bluetooth

1. Open Legomindstorm/Installation/Windows/bluetooth/ folder.
2. Double-click on PyBluez-0.18.win32-py2.6.exe and follow installation.

3.2.4 Linux

Depending on what distribution you are using go to `http://code.google.com/p/nxt-python/wiki/Installation` to see how you install correctly.

3.3 NXT

"The NXT is the intelligent micro-computer LEGO brick that can be programmed to take inputs from sensors and activate the servo motors"

This is the brain of the robot, also called the brick. It communicates with all the sensors, motors and your computer to execute commands. Below you can see how you navigate in the brick menu and how to connect all the different sensor and motors to the correct ports.

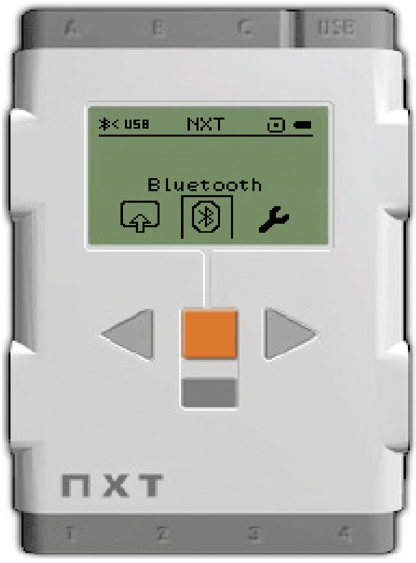

Figure 2: NXT brick

Output ports – Interactive servo motors

A – Motor used for an extra function

B – Motor for movement

C – Motor for movement

Input ports – Sensors

1 – Touch Sensor

2 – Touch Sensor

3 – Color Sensor

4 – Ultrasonic Sensor

Buttons

Left arrow – Left

Right arrow – Right

Orange square – Select / ON

Gray rectangle – Back / OFF

Output port – USB

USB – USB cable connection

3.4 Activating Bluetooth

These steps show you how to activate and connect your brick with your computer. Note that if you are using your own personal computer the instructions may differ a little. Also remember that if you are working at the university computers you will be using a USB dongle that you have to connect to the computer.

1. Press the orange square to start the brick.
2. Navigate to Bluetooth and enter.
3. Turn Bluetooth ON.
4. Connect the USB dongle to the computer and wait for it to install drivers.
5. Navigate to Control Panel –> Hardware and Sound –> Devices and Printers
6. Click button "Add a device"
7. Choose the NXT that shows up on the scan.
8. Make sure the robot is on.
9. A window will pop up on the robot that says enter passkey. Just press OK (default value is 1234).
10. The same thing will happen on the computer. Enter the value 1234 and press OK.
11. Now your robot and computer can send data to each other over Bluetooth.

NOTE: If you do not understand or get problems with this, please don't hesitate to contact an assistant to help you out.

3.5 Errors

Before you start programming your robot there are some things you will need to learn to avoid errors appearing constantly. Always start your program with connect() and end your program with the close() function. A lot of the time your program will crash due to programming errors. When this happens you have to press the orange button on the brick once to manually terminate the program. Lastly always make sure you have imported the API module otherwise you cannot use the functions it contains.

```
from API import *
connect()

#Your code here

close()
```

3.6 API

To make the robot move and to be able to read values from its sensors we need to talk with it somehow, tell it to do stuff we want. This is where an API (Application programming interface) comes in handy. The API shows us what functions we can call to control the robot. In your folder you can find the API.pdf document containing all the functions and examples how to use them.

Example of an API could be the manual to a car. You can read that if you turn the wheel the car will change direction, pushing down pedal 1 will make the car move forward and pushing down pedal 2 will make the car brake. The manual does not contain any information about how or why pressing pedal 1 makes the car move forward, that information is hidden/encapsulated and nothing you will need to worry about.

3.7 Your first program

When you connect to the brick it will usually take around 20 seconds to make the connection. This is because it looks if it can connect with USB first, if not it starts to scan the room for Bluetooth devices. We can skip all these steps and shorten the time to around 3 seconds, by finding our robot's unique id address and telling it to connect to this address directly.

These steps show you how to find your bricks unique id address.

1. Turn the brick on by pressing the orange square
2. Navigate to Settings and enter
3. Navigate to NXT version and enter

4. Write the numbers after ID on the last line down but with colons after each two numbers.

   Example write down 0017551A7A8F as 00:17:55:1A:7A:8F

Create a new py file and by using the API.pdf documentation make a program that connects to the robot by using the ID address then prints "connected" and closes the connection. From now on remember to always use your ID address in every program when connecting to greatly speed things up.

NOTE: If you do not understand or get a problem with this, please don't hesitate to contact an assistant to help you out.

4 Project: Introductory Phase

In the introductory phase you will do all the exercises for Motors, Sound, Touch sensor, Color sensor and Ultrasonic sensor to learn all the basic things your robot can do. You will also notice the limitations and strengths that it has.

After the introductory phase you should be done with all the exercises and gained enough knowledge to begin working on the project.

The structure of the exercises is the following.

Requirements: Displays the parts needed and how they should be connected before you start the exercise.

Overview: Gives you an overview of what should be done in the exercise.

(Optional) Extra info: Explains additional information that may be needed to complete the exercise.

Task: These are the questions that you should understand and answer before moving on to the next exercises.

You should write the answer or copy paste the code to the answer part in the file that was in the requirements.

Example: if motor_example.py was the file to be used in requirements you write the answers there.

4.1 Motors

Figure 3: "The Interactive Servo Motor has a built-in rotation sensor that gives you precise control of the movement of the motor (+/- 1 degree)"

4.1.1 Exercise 1

Requirements :

- API.pdf

- motor_example.py

- Interactive servo motor attached to port A

Overview:

Observe the interactive servo motor while running motor_example.py and use the code and API.pdf documentation as help to answer the tasks.

Task:

1. What does the numbers 100 and 360 stand for?
2. How do you make the motor rotate the other way?
3. What is the minimum and maximum values for power and tacho_units?

4.1.2 Exercise 2

Requirements :

- API.pdf

- motor_example.py

- Interactive servo motor attached to port A

- 1x lego connector peg (blue or black or grey these are the part that you connect lego parts together with)

Overview:

Attach the lego connector peg to any of the four orange holes (this is so you can observe how the motor rotates better). Modify the code in motor_example.py to help you figure out the tasks.

Task:

1. How do you make the interactive servo motor rotate exactly one and a quarter rotation with a power of 60?
2. Change the values and make sure you understand how you can control the interactive servo motor (written answer is not required).

4.1.3 Exercise 3

Requirements :

- API.pdf

- motor_example.py

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

Overview:

With help of the API.pdf modify the motor_example.py to run two interactive servo motors at port B and C instead of one at port A only (modify the previous code).

Task:

1. How do you make two interactive servo motors run at port B and C?
2. Do they run simultaneously or wait for each other to finish?

4.1.4 Exercise 4

Requirements :

- TrikeBase.pdf

Overview:

Now it is time to build a quick robot with wheels to test out the interactive servo motors in action. Open the TrikeBase.pdf file and follow the instructions. When you are done it should look somehow like the image above. Read the extra information before continuing.

Task:

1. Build the Trike Base robot.

Extra info:

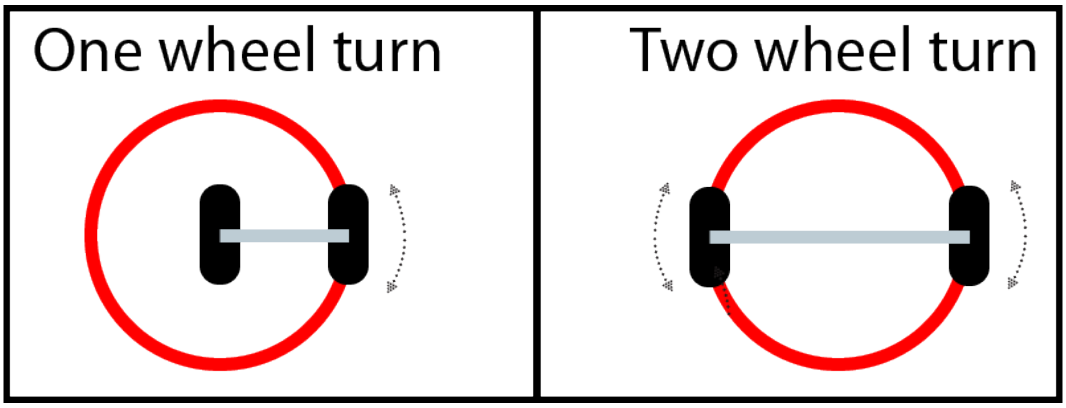

Figure 4: The robot you built can turn in two ways. The first is a "one wheel turn" which is done by moving one wheel forward or backwards while the other one stays in place. The other is a "Two wheel turn" which is done by moving one wheel forward and the other backwards simultaneously.

4.1.5 Exercise 5

Requirements :

- API.pdf

- motor_example.py

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

- Test pad

Overview:

Now that you understand how to control an interactive servo motor, let us try to turn the robot. Modify the motor_example.py to complete the task below.

Task A:

1. Put your robots left wheel in the absolute center of the test pad with the right wheel aligned on the 0/360 degree mark. Now make it turn 90 degrees so the left wheel ends up at the 90 degree line using the "one wheel turn". What was the tacho_unit value you had to use to make it turn all the way?

Explanation:

If you coded run_motor_C(100, 90) at first you will have noticed it did not turn 90 degrees and end up at the 90 mark, instead it only turned a few degrees. It is important to understand that when you code run_motor_C(100, 90) it rotates the motor at port C through 90 motor degrees – a quarter of a turn – which does not correspond to a 90 degree turn for the vehicle.

So how do we figure out how many motor degrees is needed to turn 90 degrees? This can all be done in three steps by using basic mathematics. Note that unit means cm/mm/inches the unit you decide to measure with, also remember to always use the same unit do not mix cm and mm while measuring or as input to functions.

- Calculate how many units equals a real degree
- Calculate how many motor degrees equals a unit.
- Calculate degrees to motor degrees

To make these calculations you will have to measure the "track" that is the distance between the wheels of your robot. Then you need to measure the "wheel diameter" of the wheel you are using. The pictures below will explain more and why you need to do this.

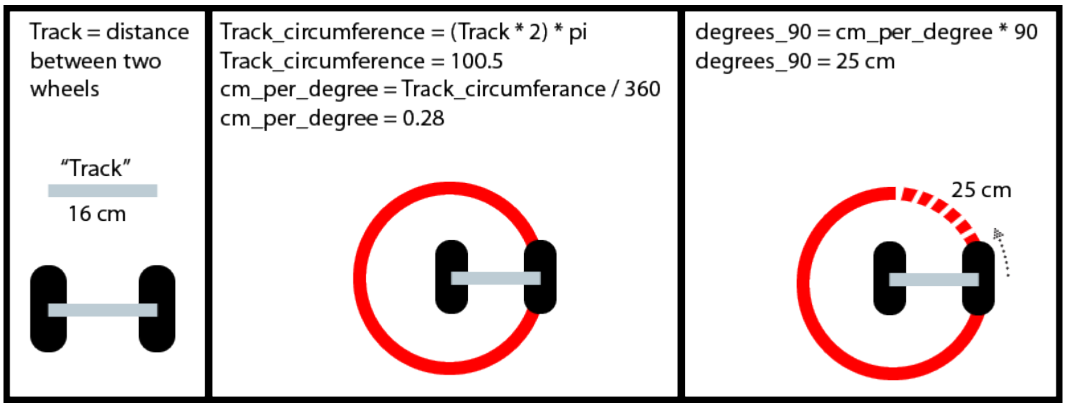
Track = distance between two wheels
"Track"
16 cm
Track_circumference = (Track * 2) * pi
Track_circumference = 100.5
cm_per_degree = Track_circumferance / 360
cm_per_degree = 0.28
degrees_90 = cm_per_degree * 90
degrees_90 = 25 cm
25 cm

Figure 5: Calculate how many units equals a degree

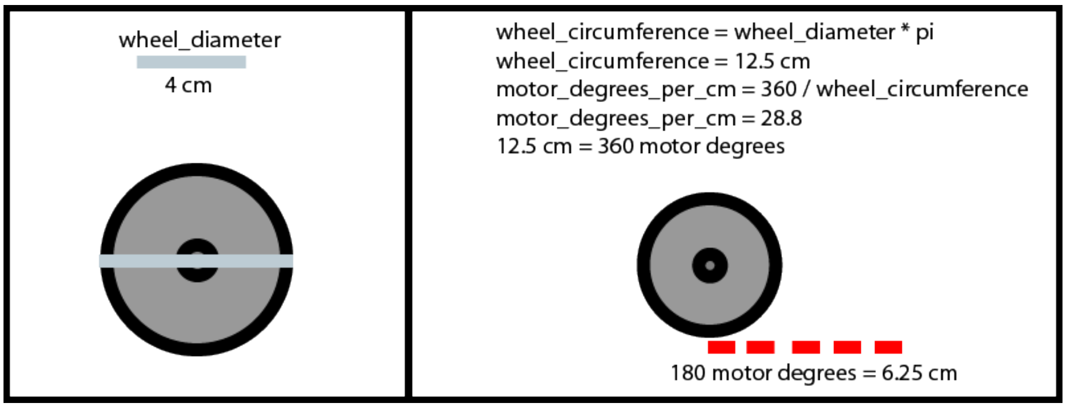
wheel_diameter
4 cm
wheel_circumference = wheel_diameter * pi
wheel_circumference = 12.5 cm
motor_degrees_per_cm = 360 / wheel_circumference
motor_degrees_per_cm = 28.8
12.5 cm = 360 motor degrees
180 motor degrees = 6.25 cm

Figure 6: Calculate how many motor degrees equals a unit.

Overview:

Modify the motor_example.py to complete the tasks below, note that you may have to experiment with the values a little to get precise turns.

Task B:

1. Make a function that takes in track as parameter and returns how many units equals one degree.
2. Make a second function that takes in wheel_diameter as parameter and returns how many motor degrees equals one unit.
3. Finally make a third function that takes in degree, track and wheel_diameter as parameters and by using the other two functions return how many motor degrees is needed to turn the given degree.

4.1.6 Exercise 6

Requirements :

- API.pdf

- motor_example.py

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

API:

- move()

- turn_left()

- turn_right()

Overview:

We will now quickly look at some functions in the API that uses synchronization to run both wheels simultaneously, making it possible to make a "two wheeled turn" or simply move forward in a straight line.

Modify the motor_example.py program with the API functions to make the robot move in a square once then cut of into the middle. But first we need to modify one of the earlier functions to make the robot calculate a "two wheel turn" instead of a "one wheel turn".

Task:

1. Figure out how to modify one of the functions you made earlier making it possible to calculate a "two wheel turn" instead of a "one wheel turn" when using them.

2. Using the API functions and the new functions you made in the previous exercise. Code an program that makes the robot move forward 20 cm, turn left 90 degrees, move forward 20 cm, turn left 90 degrees, move forward 20 cm, turn left 90 degrees, move forward 20 cm, turn left 90 degrees then finally turn left 45 degrees and move forward 10 cm.

4.1.7 Exercise 7

Requirements :

- API.pdf

- motor_example.py

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

API:

- move_forever()

- stop_move_forever()

Overview:

Until now all the functions that control movement have required a value telling them how far to rotate the motors. The function move_forever() tells the motor to rotate forever until you tell it otherwise. This makes it better in situations where you do not know how far you need to move the robot exactly.

It is important to remember (as you can read in the API description) that you always have to follow up move_forever() with the stop_move_forever() function and that you need to call the functions from within a while loop or with the built in sleep() function.

Modify the motor_example.py program with the new API functions to complete the tasks below.

Task:

1. Make the robot move forward 2 seconds using sleep then turn it left 360 degrees.
2. What happens if you comment out the stop_move_forever() function and run it again?

4.2 Sounds

The NXT brick has built in speakers that will let your robot play sounds.

4.2.1 Exercise 8

Requirements :

- API.pdf

- sound_example.py

API:

- play_file()

Overview:

The NXT brick has some built-in sound files that you can play by calling play_file(). To see what sound files you can play pick up the brick and navigate to My files then Sound files. Modify sound_example.py or the previous motor_example.py to make the robot play sounds.

Task:

1. Play around a little and make the program play some of the sounds (No answer required but make sure you understand because you will use this function later on)

4.3 Touch sensor

Figure 7: "The Touch Sensor makes your robot "feel" so that it can react to its environment"

4.3.1 Exercise 9

Requirements :

- API.pdf

- touch_example.py

- 1x Interactive servo motor attached to port B

- 1x Interactive servo motor attached to port C

- 1x Touch Sensor attached to port 1

- 1x Touch Sensor attached to port 2

API:

- get_touch1_sample()

- get_touch2_sample()

Python build-in module

- time.sleep()

Overview:

Modify the robot with touch sensors one in the front and one in the back. Open touch_example.py and use sleep() and the API functions to make a program that waits two seconds when any of the touch sensors are pressed, then moves in the opposite direction.

Task:

1. Create a program that moves forward until it hits something then waits 2 seconds before moving in the opposite direction.

4.4 Color sensor

Figure 8: "The Colour Sensor can distinguish between colours and also works as a Light Sensor, detecting light settings and ambient light, and works like a lamp, shining red, green or blue"

4.4.1 Exercise 10

Requirements :

- API.pdf

- color_example.py

- 1x Interactive servo motor attached to port B

- 1x Interactive servo motor attached to port C

- 1x Color Sensor attached to port 3

- Test pad

API:

- activate_color_sensor()

- get_color_sample()

- set_color(color)

- get_color()

Python build-in module

- time.sleep()

Extra info:

Color sensor mode is used by default every time you activate the Color sensor or by calling set_color("all"). Color sensor mode can only distinguish between black, blue, green, yellow, red and white colors.

Light sensor mode uses one of the red, blue or green lights to measure brightness. The lower the value the darker it is and vice versa.

To get correct readings when using any of the two modes you have to place the Color Sensor around 1 cm or closer to the ground/object. If it is further away the readings may become inaccurate and return wrong information. It may also provide wrong information if it is to bright in the room.

To use the Color Sensor you always have to activate it using activate_color_sensor() after that you can change color mode and fetch values.

Overview:

Modify the robot so the color sensor is placed approximately 1 cm from the ground. Then open and edit color_example.py to solve the tasks below by using the API functions and sleep().

Task:

1. Create a program that uses Color sensor mode to print the color it currently sees. (No moving is needed – you can just hold any object in front of it manually)
2. Create a function that changes the color of the color sensor every 2 seconds.
3. Create a program that uses Light sensor mode(red, green or blue) that makes the robot seek and move towards bright light. The robot should try to move in a new direction until it finds a brighter value, then move towards it as long as it get brighter. (If you find it hard to test, use a white paper that you put before the sensor and take away to make sure it behaves correctly.)

4.5 Ultrasonic sensor

Figure 9: "The Ultrasonic Sensor acts like a radar, making the robot see, measure distance and react to movement"

4.5.1 Exercise 11

Requirements :

- API.pdf
- ultrasonic_example.py
- 1x Interactive servo motor attached to port B
- 1x Interactive servo motor attached to port C
- 1x Ultrasonic Sensor attached to port 4

API:

- get_ultrasonic_sample()

Python build-in module

- random.randint()
- time.sleep()

Overview:

In this exercise we will be using one of Python's built-in modules called random. It can do a lot of things but we will be using the function randint() to return a random value between two numbers we decide. This way we can make the robot behave randomly.

Open and edit ultrasonic_example.py and use the randint() and sleep() functions to solve the task below.

Task:

1. Create a program that makes the robot move around randomly in the room playing a random sound file and turning around 180 degrees every time it comes close to an object.

4.6 Compass sensor

Figure 10: "The NXT Compass Sensor is a digital compass that measures the earth's magnetic field and outputs a value representing the current heading."

4.6.1 Exercise 12

Requirements :

- API.pdf
- compass_example.py
- 1x Interactive servo motor attached to port B
- 1x Interactive servo motor attached to port C
- 1x Compass Sensor attached to port 2
- 1x Touch Sensor attached to port 1

API:

- get_compass_sample()
- get_relative_compass()
- is_in_range_compass()

Python build-in module

- time.sleep()
- random.randint()

Overview:

In this exercise we will be using the compass sensor functions.

Open and edit compass_example.py and use the get_compass_sample(), get_relative_compass() and is_in_range_compass() functions to solve the tasks below.

Task:

1. Create a program that makes the robot move N, 2 second then S, 2 seconds then W, 2 seconds then E, 2 second.
2. Create a program that makes the robot turn in the direction of 300 every time you press the Touch Sensor.
3. Create a program that makes the robot move around randomly in the room playing a sound file, while it is in between the values 0 and 90.

5 Project: Main Phase Guard dog and GUI

In this project you will be creating a robot that drives around peacefully but when commanded acts like a guard dog. When commanded the robot will patrol the area you tell it to in a friendly manner but if an intruder enters the area they will get warned and if they do not remove themselves the robot will start shooting colorful happy balls at them. After this you will create a GUI (Graphical User Interface) to control the robot. In the end you will have a GUI that anybody can use to control the robot and perform different tasks.

The GUI should have two states "live state" and "building state". In "live state" anytime a button is pressed the robot will perform the action connected to the button directly. It will be like using a Xbox, PS controller to control the robot but in this case using the GUI on the computer instead.

In "building state" you will click a button which will add an instruction to a list or other data structure(list, dictionary etc) that you find best. When you are finished you will press a button that will run all the instructions in the list. This is the basic idea of programming, you build a program using instructions(buttons) and when you are done you press run and the program will be executed and perform all the tasks you told it to.

The GUI will also contain an area that shows how the robot has moved/move in the environment.

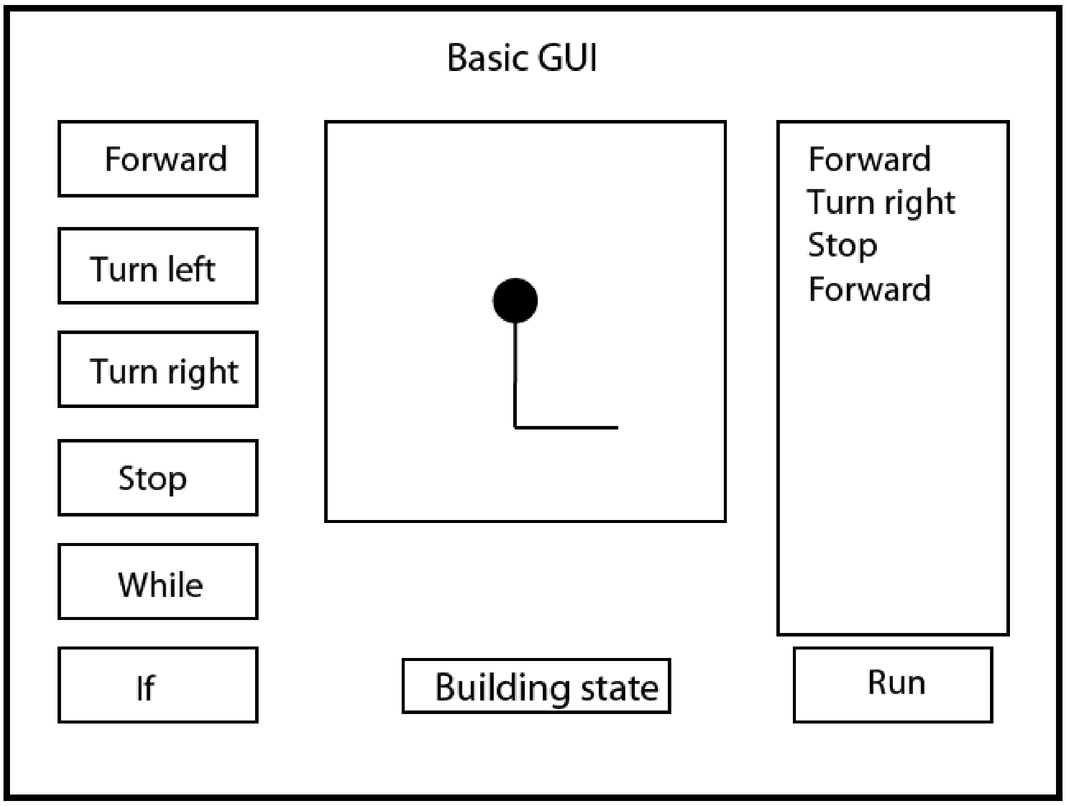

Figure 11: Picture showing you a very basic GUI.

5.1 Milestone 1

Use the shooterbot.pdf to build the shooterbot. IMPORTANT!!! skip page 10 and 11 when building.

5.2 Milestone 2

When you turn on the robot it should always start out in a playful/peaceful mode. Make the robot drive around randomly in the room and if it encounters an object it should make a sound and keep going another way.

5.3 Milestone 3

When you want your robot to start patrolling (from peaceful mode to guard mode) you will need to interact with the sensors somehow to let him know.

Find a way to make it possible for you to physically interact with the robot and fiddle with the sensor so it goes into guard mode. When it goes into guard mode it should do some kind of signature dance so you clearly can see that it is entering guard mode.

5.4 Milestone 4

After you built your robot you will notice it has more then 2 wheels or use tracks instead of wheels. Rewrite/remake the earlier functions you made in the exercises or find another way to make the new robot calculate turns correctly to be able to patrol later on.

5.5 Milestone 5

Know that the robot is in guard mode it will need to patrol and guard an area. Make it patrol a square of at least 1 * 1 meters repeatedly.

Explanation:

When entering guard mode and after it has done the signature dance it should start to patrol the given area(square) of at least 1 * 1 meter in front of it, you can assume it starts at the corner of the square. It is not enough that it runs around in a square repeatedly and only sees the edges of the square. It should patrol every space of the square at some point.

5.6 Milestone 6

If the robot spots an intruder it needs to be able to fire colorful happy balls at them. Create a function that uses the third servo motor to shoot away balls.

5.7 Milestone 7

When an intruder is spotted it should stop and give a warning sound. After the sound it should start an countdown and change the color of the light every two seconds until 6 seconds have passed, then initiate fire by using the function you created in milestone 5. If the intruder moves away so the robot cannot detect it anymore, the countdown should be stopped and the robot resume his patrolling from where it was before it detected the intruder.

5.8 Milestone 8

If the robot runs out of balls to shoot it can no longer scare away intruders. Keep an counter over how many balls you have fired and if no balls are left to shoot away, abort the current patrolling(exit guard mode) and make the robot make sounds until it finds a corner in the room to hide in then exit program.

Explanation:

The robot should be able to find his way to a corner somehow and be able to detect that it is a valid corner then move as close as possible before exiting the program.

5.9 Milestone 9

When you want your robot to stop patrolling and go back to peaceful mode, you will need to interact with the sensors somehow.

Find a way to make it possible for you to physically interact with the robot and fiddle with the sensor so it goes into peaceful mode. When it goes into peaceful mode it should do some kind of signature dance so you clearly can see that it is entering peaceful mode.

5.10 Milestone 10

Look at the tkinter wiki and do An Introduction to Tkinter to learn how to use it or try and find other examples. Tutorial - `http://effbot.org/tkinterbook`
Wiki - `http://wiki.python.org/moin/TkInter`

5.11 Milestone 11

This is were you start creating your own custom GUI. Create the "live state", connect all the functions from the API.pdf to your buttons. Solve how to enter additional parameter information when clicking the buttons. Your goal is to be able to click a button on the GUI that in turns calls the API function with the given information, which makes the robot perform the task.

5.12 Milestone 12

Combine some functions to create a button that does something the API.pdf functions cannot do themselves. An example would be to make a button that makes the robot make a soft turn by using one wheel only.

5.13 Milestone 13

Add functionality so you can see what mode the robot is in (guard mode or peaceful mode).

5.14 Milestone 14

Create an area in the GUI showing how the robot has moved so far look at Figure 11 to get an idea. If you detect and intruder/object it should show up as an object on the screen to.

5.15 Milestone 15

Create the "building state" using the previous buttons plus a run button. You will now need to figure out how to add the buttons instructions to some data structure(list, dictionary), and when you press the run button all the instructions in the data structure you have chosen should be run/executed.

5.16 Milestone 16

The user wants to get an overview over what building instructions has been added to the list or chosen data structure. Add an area to the GUI showing what buttons/instructions have been added and in what order they were added. This is so the user can get an overview over how the program will run, look at Figure 11 to get an idea.

5.17 Milestone 17

Replace the previous area from "live state" that shows how the robot is currently moving and instead create an area in the GUI showing what path the robot will move when the list with all the instructions is run.

5.18 Milestone 18

Add an if statement button/buttons and solve how to add conditions to an if statement and how to add buttons/instructions to be executed inside the if statement as long as it is true/false. Figure out a good way to display the if statement in the milestone 14 task, so the user can see how he built the if statement(the condition) and what instructions it contains.

Explanation:

The user adds a "forward" instruction to the list and then he wants to add an "if" statement with some condition and if the condition is true/false he wants the "if" statement to run an "Turn left" and a "forward" instruction then exit the "if" statement and continue with the rest of the instructions. And as said above figure out a good way to display the if statement/statements and conditions in the milestone 14 task.

5.19 Milestone 19

Add an while loop statement button/buttons and solve how to add conditions to a while loop block and how to add buttons/instructions to be executed inside the while loop as long as it is true. Figure out a good way to display the while loop in the milestone 14 task, so the user can see how he built the while loop and what instructions it contains.

5.20 Milestone 20

Add a feature so you can save the list with instructions(program) you have put together with a name, so you can restore everything as it was.

Explanation:

User adds some instructions to the list then saves it with the name example_A, the user then creates a new list with instructions and saves it with the name example_B. After a while the user decides to load example_A which shows up and loads exactly as it looked when you saved it.

5.21 Milestone 21

Add functionality so you can see what speed the robot is traveling in (try to find information about the acceleration sensor online and look at the API which has some functions to use it, or find another way to solve it)

5.22 Milestone 22

Add functionality so you can see how far the robot has traveled since the program started. (try to find information about the acceleration sensor online and look at the API which has some functions to use it, or find another way to solve it)

5.23 Milestone 23

Remake the guard mode so that instead of patrolling 1 * 1 meters it finds a way to find identify all the corners of the room then proceeds to patrol the whole room. To make it easier you can assume there is no objects like chairs or tables in the room.

Explanation:

You turn on guard mode in which the robot finds a way to scan the room somehow so it knows how to patrol every inch of it.

5.24 Milestone 24

After each day at work guarding stuff like a champ the robot enjoys his hobby of chasing black lines on the ground.

Modify the robot so the light sensor is pointing towards the ground instead(remember to place it around 1 cm from the ground). You can ignore all the previous milestone you have done and start these features in a new file.

5.25 Milestone 25

The first thing you need to know is that when you follow a line, you are not following the middle of the line, but the edge of it. If you try to follow the middle of the line, and if the robots light sensor encounters white, you will not know which way it should turn. By following either the left or right edge, you will always know what the robot should do.

To determine what is the line and what is the background, you will need to determine what a good brightness value to use as the "cut-off" between the two states(light and dark). You can do this in two steps, first measure the center of the line (darkest) then measure the floor next to it (brightest). By saving the values you can then average them to find the cut-off.

Create a function that calculates the cut-off value and saves it for further use. You can assume that you always start with the robot on the line. You can always read more about this on the web maybe there is a better way of doing it.

5.26 Milestone 26

Solve how to make the robot follow the black line on the testpad.

5.27 Milestone 27

Create a function that makes the robot turn around and keep going the other way on the testpad.

5.28 Milestone 28

Make it run as fast as it can without failing and then research online if there is a better way of doing it.

5.29 Milestone 29

Make sure you robot can handle very sharp turn. Put out duct tape or cut out a piece of paper and make a 90 degrees line turn, make sure the robot can pass this turn.

5.30 Milestone 30

Solve how to make the robot go around a object and continue if it should encounter something.