TDDD63 Project: EV3 Lego Mindstorms Exercises

Jonathan Doherty
Martin Söderén

September 25, 2014

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. Lastly using the four 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 EV3, 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 EV3 and the files that come with the project.

The Lego Mindstorms EV3 package main parts:

- 1 EV3 micro computer - it is actually a fullsize computer comparable with a Raspberry Pi. It uses a Debian distribution as operating system.
- 1 Touch Sensors - that makes the robot feel
- 1 Color Sensor - that can detect different colors, light settings, the reflection and acts as a lamp
- 1 IR sensor that can measure distance, track a beacon and receive commands from the remote control
- 1 remote control that sends commands to the IR sensor
- 4 Interactive servo motors with built-in rotation sensors
- 8 connector cables for linking motors and sensors to the EV3

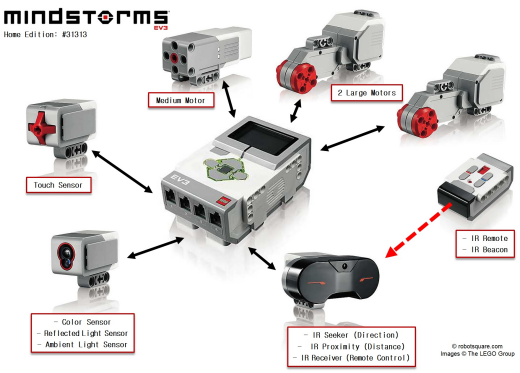

Figure 1: Main parts

3.1 EV3

This is the brain of the robot. It communicates with all the sensors, motors and your computer to execute commands.

Figure 2: EV3 brick

Output ports – Interactive servo motors

A – Motor for movement

B – Motor for movement

C – Motor for movement

D – Motor for movement

Input ports – Sensors

1 – Sensor

2 – Sensor

3 – Sensor

4 – Sensor

Output port – USB mini B

USB – USB for connection to computer

Output port – USB type B

USB – USB for connection to external peripherals

3.2 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 the folder intro 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.

4 Project: Introductory Phase

In the introductory phase you will do all the exercises for Motors, Speak, Touch sensor, Color sensor and IR 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. Some questions require regular answer in form of text. These will be answered in the document called answers which is located answers folder. Some questions will be answered with your program code. In that case you will copy your main_program.py file in the answers files.

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.

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

- Interactive servo motor attached to port A

API:

- motor()

- run_position_limited()

Overview:

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

```
#ev3 libraries
from system.sensors import touch,motor,speak,IR,color

def main(brick):
    motor_A=motor(brick,"A")
    motor_A.run_position_limited(100, 360)
```

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 speed and position?

4.1.2 Exercise 2

Requirements :

- API.pdf

- 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)

API:

- motor()

- run_position_limited()

Overview:

Attach the lego connector peg to any of the four orange holes (this is so you can observe how the motor rotates better).

Task:

1. How do you make the interactive servo motor rotate exactly one and a quarter rotation±1 degree with a speed of 100?
2. How exact is the movement?
3. Can you make it more exact?
4. 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

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

API:

- motor()

run_position_limited()

Overview:

Modify the main_program.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. Observe that this instruction are for the older NXT 2.0 so some parts might have different colors and differ in other ways but you are on your way to become engineers so you should be able to figure it out.

Task:

1. Build the Trike Base robot.

Extra info:

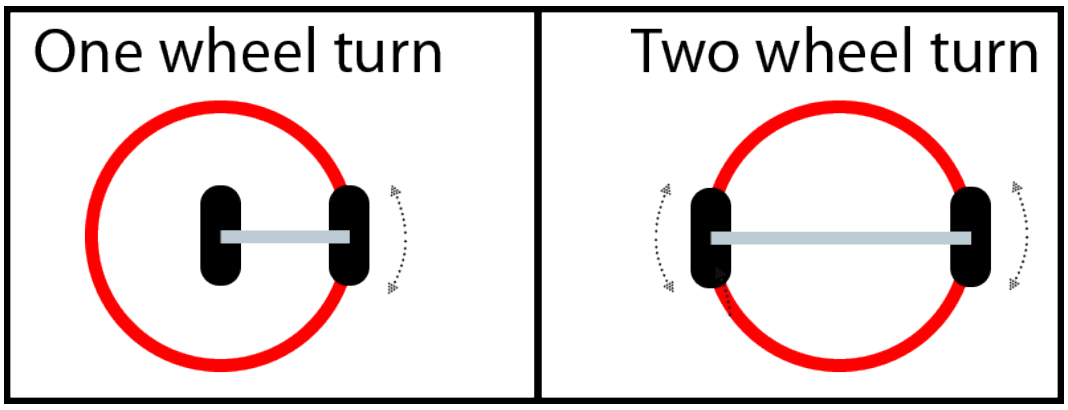

Figure 4: The robot you built can turn in two ways. The first is a "one wheel turn" which is done by moving one wheel forward or backwards while the other one stays in place. The other is a "Two wheel turn" which is done by moving one wheel forward and the other backwards simultaneously.

4.1.5 Exercise 5

Requirements :

- API.pdf

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

API:

- motor()

- run_position_limited()

Overview:

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

Task A:

1. Make it turn 90 degrees with a "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_position_limited(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_position_limited(100, 90) it rotates the motor 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.

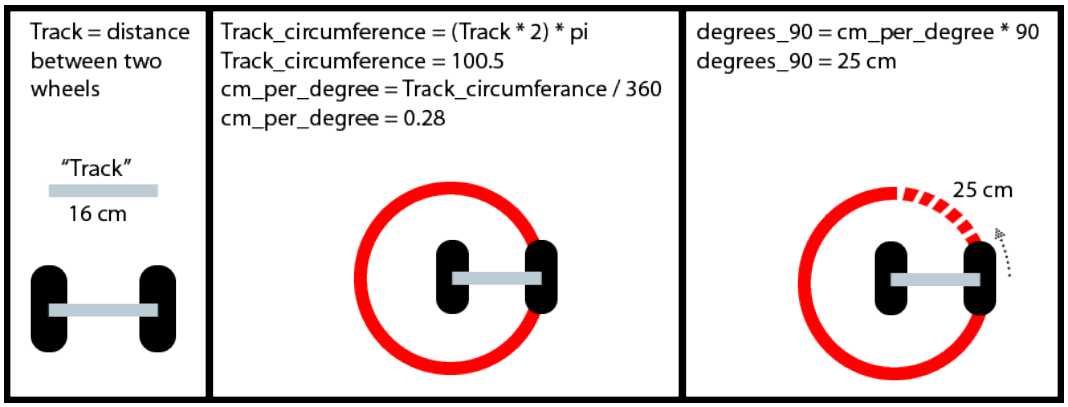

Figure 5: Calculate how many units equals a degree

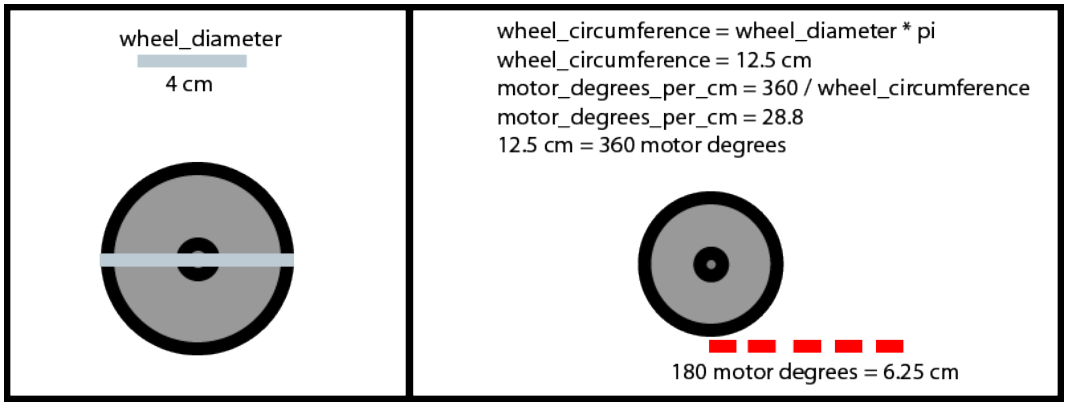

Figure 6: Calculate how many motor degrees equals a unit.

Overview:

Modify the main_program.py to complete the tasks below, note that you may have to experiment with the values a little to get precise turns. Remember that slower speed gets you more precision.

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. Copypaste the three functions you have made into the exercise_5.py file in the answers folder and save it.

4.1.6 Exercise 6

Requirements :

- API.pdf

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

API:

- motor()

- run_forever()

- run_position_limited()

- stop()

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.

Modify the main_program.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() function and run it again?
3. copypaste the main_program.py into the exercise_6.py file in the answers folder and save.

4.1.7 Exercise 7

Requirements :

- API.pdf

- Interactive servo motor attached to port B

- Interactive servo motor attached to port C

API:

-motor()

- run_time_limited()

- run_position_limited()

Overview:

The effect of last exercise can be achieved with the run_time_limited function instead. See the API.

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

Task:

1. Make the robot move forward 2 seconds then turn it left 360 degrees.
2. copypaste the main_program.py into the exercise_7.py file in the answers folder and save.

4.2 Sounds

The EV3 has built in speakers that can be used for making the robot talk.

4.2.1 Exercise 8

Requirements :

- API.pdf

API:

- speak()

- talk()

Overview:

The EV3 can talk by using the linux programs espeak and aplay. Espeak creates a wav file from your text input and aplay playes it using the EV3s tone generator. You give it different accents with the parameter lang. All available languages can be found here: http://espeak.sourceforge.net/languages.html

Task:

1. Play around a little and make the EV3 speak. If you want you can try out some different accents.No answers are required for the exercise.

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

- 1x Interactive servo motor attached to port B

- 1x Interactive servo motor attached to port C

- 1x Touch Sensor attached to port 1

API:

- touch()

- is_pressed()

Python build-in module

- time.sleep()

- random.randint()

Overview:

Modify the robot so it has one touch sensor in the front. Make use of the API and your earlier experience with the EV3 to finish the task

Task:

1. Create a program that makes the robot move forward until it hits something then waits 2 seconds before moving in the opposite direction by turning 180 degrees and then keeps moving forward until it hits something again. Remember to import the time module.
2. instead of making it turn 180, make it turn somewhere between 90 and 270 degrees by using the randint function from the random module.
3. copypaste the main_program.py into the exercise_9.py file in the answers folder and save.

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

- 1x Interactive servo motor attached to port B

- 1x Interactive servo motor attached to port C

- 1x Color Sensor attached to port 2

API:

- color()

- set_ambient_mode()

- set_color_mode()

- get_value()

- get_color_string()

Python build-in module

- time.sleep()

Overview:

Modify the robot so the color sensor is placed in front of the robot. Then solve the tasks below by using the API functions and sleep(). Remember to import the time module.

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 Ambient sensor mode 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. copypaste the main_program.py into the exercise_10.py file in the answers folder and save.

4.5 IR sensor

Figure 9: "The digital EV3 IR Seeker Sensor detects/measures proximity to the robot and reads signals emitted by the EV3 IR Beacon."

4.5.1 Exercise 11

Requirements :

- API.pdf

- 1x Interactive servo motor attached to port B

- 1x Interactive servo motor attached to port C

- 1x IR Sensor attached to port 4

API:

- IR()

- set_proximity_mode()

- set_seek_mode()

- set_remote_control_mode()

Overview:

Modify the robot so the IR sensor is placed in front of the robot. Then solve the tasks below by using the API functions and sleep(). Remember to import the time module.

Task:

1. Create a program that uses the Remote Control Mode and make the robot remote controlled. For example can the red up and red down button controll

the left motor and the blue up and the blue down can control the right motor. If you switch channels during operation you have a combination of 20 different buttons. You could for example make the robot say something when you press a certain button.

2. copypaste the main_program.py into the exercise_11a.py file in the answers folder and save.
3. Create a program that uses the Seek Mode and follows the IR beacon. It does not need to follow it regarding the distance but it should try to look straight onto the IR beacon by turning left and right.
4. copypaste the main_program.py into the exercise_11b.py file in the answers folder and save.
5. Create a program that uses Proximity Mode and makes the robot keep a distance of 30cm from the nearest object. If the object moves closer to the robot the robot should move backwards and if the object moves further away the robot should follow.
6. copypaste the main_program.py into the exercise_11c.py file in the answers folder and save.