

EV3 Quick Guide

SDP Team

January 2018

1 Introduction

The aim of this guide is to allow you to quickly setup your EV3 brick and how to execute programs. The brick has a custom OS installed called `ev3dev`, with more details found here <http://www.ev3dev.org/>.

The EV3 has a variety of sensors that are connected via one of the four ports on the bottom of the brick, listed as 1, 2, 3, and 4. The top ports, listed as A, B, C, and D are used for motors. This tutorial will go through setting up motors and a few of the sensors, specifically the touch and ultrasonic sensors.

2 Connecting to the brick

2.1 USB connection

To turn on the brick make sure that the EV3 brick is not connected to any motors or sensors, as this will significantly increase the time for the brick to boot. This guide will focus on using the brick via the DICE machines, and will only be concerned with connecting using a USB cable. For details on using non-DICE machines please see here [1].

The instructions for DICE are as follows:

1. Turn on the EV3 brick
2. Plug the USB cable into the PC port on the top of brick, and the cable into the computer
3. ssh into the brick through the terminal: `ssh robot@ev3dev`
4. The password is maker

If you are having difficulty connecting, the brick may not have its IP address setup correctly. To fix this:

1. Navigate on the brick to *Wireless and Networks*
2. Then go to *All Network Connections*

3. Click on *Wired*
4. Scroll to *IPv4*
5. Press *Change*
6. If you ping `ev3dev` this will say what the IP address should be, and this is what you need to change the IP address to

2.2 Bluetooth Connection

2.2.1 Connecting to Computer

As of writing this document, the EV3 currently cannot connect to the DICE OS via Bluetooth. If you wish to connect via Bluetooth you will need to get a self managed machine. Email Garry Ellard with a small justification why you need the self managed machine and one will be setup for you in your group's assigned demo room. You will also be given a Bluetooth dongle so that your machine can use Bluetooth. You may use this machine however your group wishes, installing Windows, Ubuntu, etc.

The instructions are taken from this tutorial [2], and the instructions for connecting with Ubuntu are summarised below.

1. You will need to have Bluetooth Manager installed: `sudo apt-get install blueman`
2. Open Bluetooth Manager
3. In the *View* menu select *Local Services*
4. Go to *Network*, and tick *Network Access Point (NAP)* and *dnsmasq*
5. Change the IP address to 10.42.0.1
6. Click *Apply*, and then *Close*
7. On the EV3, scroll to *Wireless and Networks*, then *Bluetooth*, and make sure that *Powered*, and *Visible* are both checked
8. On your computer, in Bluetooth Manager, click *Search*, and select *ev3dev*
9. Next you have to pair the devices, which is done by pressing the key icon. You must make sure that the EV3 is not paired with any other devices at this stage, as it will not recognise the new changes made on your computer via steps 3-5. After this you will be able to pair with multiple devices in different orders (e.g. a second EV3 can connect to your computer at a later date, even if the EV3 has also paired with your phone beforehand)
10. When you press the Key icon, both devices will be sent a confirmation code displayed on both the computer and the EV3, click accept on both devices

11. From here you need to have the computer trust the device by clicking the Star icon
12. With the devices successfully paired, there should now be a new button to press on on the EV3 screen, *Network Connection*. You can find this at a later date by going to *Wireless and Networks*→*All Network Connections*→*YOUR MACHINE*
13. In the new list of options, click *Connect*, an IP address should now be present on the top left of the EV3 screen (you can also check *Connect automatically* which will, if possible, connect to this machine when it boots up)

With the brick now connected you can access the brick through your computer in a similar manner to USB connection: `ssh robot@IP_ADDRESS_ON_EV3_SCREEN`.

2.2.2 Connecting to a Phone (Android)

You can also connect the EV3 to your phone via Bluetooth, but specifically Android. There does not seem to be official methods on connecting to iPhones. The instructions below are summarised from [3].

1. On the EV3, scroll to *Wireless and Networks*, then *Bluetooth*, and make sure that *Powered*, and *Visible* are both checked
2. On your Android phone turn on Bluetooth
3. Navigate to *Tethering & Mobile Hotspot* and turn on *Bluetooth Tethering*
4. On your phone pair with the EV3
5. A confirmation code will appear on both devices, click accept on both
6. With the devices successfully paired, there should now be a new button to press on on the EV3 screen, *Network Connection*. You can find this at a later date by going to *Wireless and Networks*→*All Network Connections*→*YOUR PHONE*
7. In the new list of options, click *Connect*, an IP address should now be present on the top left of the EV3 screen (you can also check *Connect automatically* which will, if possible, connect to this phone when it boots up)
8. You can ssh into the brick from the phone using the **ConnectBot** app, and in a similar manner to that connecting via desktop

2.2.3 Connecting to another EV3 brick

This is very similar to connecting to other devices (computer or phone), but on one of the bricks you will need to navigate to *Wireless and Networks*→*Tethering* and checking *Bluetooth Tethering*. From here you can pair the devices and on the second brick connect in a similar way as described earlier: *Wireless and Networks*→*All Network Connections*→*THE OTHER EV3 BRICK*→*Connect*.

3 First Commands

The EV3 brick can run python programs, and so can also run commands via the python terminal. Type `python3` to open the terminal. If you enter the code below

```
import ev3dev.ev3 as ev3
m=ev3.LargeMotor('outA')
if not (m.connected):
    print('Plug a motor into port A')
else:
    m.run_timed(speed_sp=300, time_sp=1000)
```

the motor will run for one second if connected properly. What the code is doing is firstly importing the EV3 modules, then assigning a motor, and lastly we perform a check to make sure the motor is connected.

This allows small snippets of code to be tested on the brick quickly in real time (as will be seen is useful in the next section). It is important to note that the commands the EV3 is currently running may continue even if you exit the python terminal, i.e. likely a motor is continuing to operate. If this happens open up the terminal again, assign the correct motor and use the code

```
import ev3dev.ev3 as ev3
m=ev3.LargeMotor('outA')
m.stop()
```

If you are running a program through the UI on the brick you can stop the program by pressing the top left button for 3 seconds.

4 Running Programs

4.1 Software Cycle for Programming the Brick

The general development cycle for software development with python on the EV3 is

1. Write your code in a text editor on your computer
2. Send your files to the EV3: `scp file1 file2 etc robot@ev3dev:/home/robot`

3. Test your code on the brick by using the UI on the brick, or connect to the brick and through the terminal type: `./program.py`
4. If the program returns an error stating that it is not an executable file use the command: `chmod +x file.py`
5. If you ran the program via the UI and the program is in a loop, you can stop the program by pressing the top left button for 3 seconds

To be able to run the python code via the UI on the brick it needs to be made as an executable file. The bricks should be set up already to do this, but at the start of any python file that you wish to run as an executable needs to have this line of code

```
#!/usr/bin/env python3
```

There is a text editor on the brick, nano, that you can use to edit files already on the brick. This will obviously not change the code you have on your computer, meaning that any uploads of your code from the computer will overwrite the files on the brick. It's advised that any changes on the code should be done on the files on your computer and then re-uploaded to the brick to ensure consistent work copies.

4.2 Example Files

On the SDP wiki [4] there is a folder containing several example files that you can download (*EV3 Test Files*). The file you will want to run is `main.py` which will call one of the functions in `tutorial.py`. To try a different function uncomment the function to be called from `tutorial.py` and make sure that the other call to functions are commented out.

Briefly going over the functions in `tutorial.py`

- Step A: This gives an example of motors running, and demonstrates that these commands are non-blocking
- Step B: By using a touch sensor, the user can turn on and off an LED light on the EV3 brick
- Step C: By using 2 touch sensors, the user can either make the motor(s) go forward or backward
- Step D: The ultrasound sensor records readings until the user presses the back button (top left on the brick), and it then outputs a text file with these readings

Lastly there is an example MATLAB file called `graph_maker.m` which takes in a text file, produces and saves a graph. This can be used with the output file from Step D. The function takes 4 arguments:

1. filename: A string of the path to the text file to be used

2. `tle`: A string which shall be the title on the graph
3. `x_axis`: A string stating what the x-axis should be named as
4. `y_axis`: A string stating what the y-axis should be named as

The lines of code

```
fileID = fopen(filename , 'r ');
formatSpec = '%f ';
results = fscanf(fileID , formatSpec );
```

is what reads in the values in the text file. The second line dictates how the data should be read. In this example it shall assume the data is one column of float numbers. To change the number of columns to be considered you just have to add another type to the string. For example if you wanted to read in a file with two columns of float numbers then it will be

```
formatSpec='%f %f ';
```

4.3 Communicating between Devices

Below is a summarised tutorial of using `Mosquitto` as a communication protocol for `ev3dev` from [5].

`Mosquitto`, `MQTT`, is a communication protocol between different devices. The general idea is that there are publishers and subscribers for topics. A very simple example is that there are two agents A and B and there is topic C, and let's also say A publishes to C, while B subscribes to C. What this means is that whenever A publishes to C, i.e. sends data to topic C, B will be sent this data since it is subscribing to it.

The last thing to consider is which device is going to be the Broker, which handles the messaging and determines who should be sent what data given their publishing/subscription to topics. A Broker can also publish and subscribe to topics. A good overview with more information on `MQTT` can be found here [6].

To setup `MQTT`

1. On the device that is going to be the Broker: `sudo apt-get install mosquitto`
2. You can check if this has installed properly by running the command: `systemctl status mosquitto`
3. You will need `pip3` to install the `MQTT` module on all devices that are going to use `MQTT`: `sudo apt-get install python3-pip`
4. Finally: `sudo pip3 install paho-mqtt`

It must be noted that on the EV3 bricks this can take a long time to install the necessary files.

There is example code of a publisher and subscriber in the test code found on the SDP wiki [4]. These files are `publisher.py` and `subscriber.py`. The machine running `subscriber.py` will be stuck in a loop until it hears a message saying “Hello World!” from the topic “test”. Once it has the machine will print to the terminal “Yes!”, disconnect and the program will end. Any machine that executes the `publisher.py` file will connect, send a message saying “Hello World!” to the topic “test”, and then disconnect.

It should also be noted that the first argument passed in the line of code `client.connect(“THE_IP_ADDRESS_OF_OUR_BROKER”,1883,60)`

in both files will need to be replaced with the IP address of the Broker, still in quotation marks. If it’s an EV3 brick this can be found on the top left of the screen, if it’s a computer then you can find this by running the command: `hostname -I`. If the machine publishing/subscribing is the Broker then the first argument can be set to “localhost”.

4.4 C++

ev3dev can also support programs written in `c++`. These programs have to be compiled first though, and requires to be done through the ev3dev OS. So while programs can be compiled on the brick itself this is very slow and it is advised instead to use docker to compile the programs first on your computer. Afterwards transfer the compiled programs across. This document will not go into detail on how to set this up, but you can find full details here [7].

References

- [1] <http://www.ev3dev.org/docs/tutorials/connecting-to-the-internet-via-usb/>.
- [2] <http://www.ev3dev.org/docs/tutorials/connecting-to-the-internet-via-bluetooth/>.
- [3] <https://github.com/ev3dev/ev3dev.github.io/blob/master/docs/tutorials/connecting-to-the-internet-via-bluetooth.md/>.
- [4] <https://wiki.inf.ed.ac.uk/SDP/WebHome>.
- [5] <http://www.ev3dev.org/docs/tutorials/sending-and-receiving-messages-with-mqtt/>.
- [6] <http://www.steves-internet-guide.com/mqtt/>.
- [7] <http://www.ev3dev.org/docs/tutorials/using-docker-to-cross-compile/>.