Getting started with Dash

What is Dash?

Dash is a small, programmable robot designed for educational purposes, particularly for teaching children coding and robotics concepts. However, it can also be programmed using python for more advanced users. Dash Robot is produced by Wonder Workshop, a company that focuses on creating interactive and engaging educational tools for children. **We recommend you first follow this guide to acquaint yourself with the robot. The official documentation for programming in Python is quite outdated, as such read this guide before delving deeper into the official documentation and the other third-party libraries that we will get to know in this document.**

What comes with the robot?

If you choose to use Dash, you will be given access to a box containing:

- 1 Dash
- 1 Bluetooth USB Adapter
- 1 charging cable

All equipment underwent thorough testing and verification prior to its availability to students. Kindly exercise caution and ensure the return of each item enclosed in the box in the same condition as it was borrowed.

What can Dash do?

Dash Robot from Wonder Workshop is a versatile educational robot with various features and capabilities. Here are some of its main functionalities:

1. **User programmable LED's and Buttons.**
2. **IR Receivers and Transmitters:**
    - Enables Dash to find and interact with other robots.
3. **Potentiometers and Dual Motors:**
    - Supports head pan and tilt
4. **3 Proximity Sensors:**
    - Detects objects left, right and back
5. **Wireless Connectivity:**
    - Dash can connect to devices wirelessly (via bluetooth), allowing for remote control and programming.

6. **3 Microphones and Speakers:**
    - Real-time voice triangulation and personalized recording and playback
7. **3 Processors and Sensor Fusion:**
    - Manages complex interactions among actuators and sensors
        i. Accelerometer encoder
        ii. Gyroscope encoder
        iii. Wheel encoder
8. **2 Powered Wheels (and one small non-powered one at the back):**
    - For quick navigation and distance tracking

You can find out more about Dash here.

How can I program Dash?

Dash Robot is produced by Wonder Workshop, a company that focuses on creating interactive and engaging educational tools for children. As such, all official applications available to control and program Dash only support either Draw or Drag-and-Drop coding languages. This is quite limiting for the university setting.

However, there exists an official python library that you can find here. This is the most complete library available for Dash but has two major drawbacks: it only supports python2.7 and **can only be runned on MacOS**. For this reason we recommend using one of the following two third party libraries to control Dash.

The recommended library is called MorseAPI and was created by Illya Sukhanov. To run this API you need to use python2.7 and a linux environment (macOS and windows are not supported). We tested using Debian 12 and were able to successfully connect and control the robot using this API. Although the documentation is, frankly, not there, the code is thoroughly commented and each function is easy to understand. This API implements all core functionalities needed for a project so it is the best option.

Another (but less complete) alternative is to use the python-dash-robot library. The advantage is this library supports python3, **however there are no functions to get the data gathered but the sensors, which might prevent you from using this library depending on your project.** But if only head movement, general movement, the eye leds and sounds are necessary then it is an option. We tested it with Ubuntu 18.04 using python3.6 and we successfully connected to a robot

This guide will explain how to set-up and run a script to control Dash using the MorseAPI code with python2.7 in Debian12.

Getting started: Turn on and run your first program in Python

As previously said, we tested the MorseAPI within a Debian environment. If you don't have a linux machine, you can create a virtual machine that runs Debian.

Creating your Debian environment for Dash from scratch:

Installing Debian 12 on a Virtual Machine:

If importing the virtual machine is not an option or you want to create it yourself you can follow these steps.

1. Install VirtualBox
2. Download a Debian 12 .iso image (here)
3. Follow this video tutorial and create a Debian 12 virtual machine
    a. Installing Guest Additions tutorial

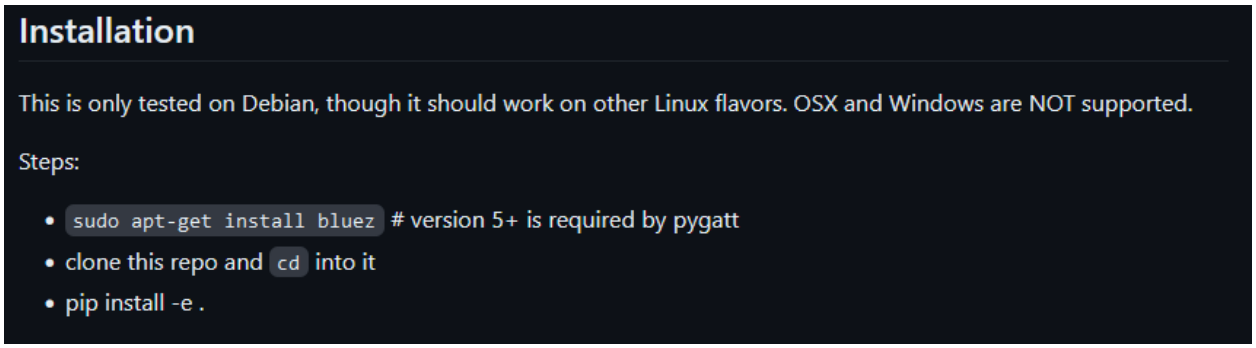
Installing Anaconda:

We recommend using anaconda to create a virtual environment that runs with python2.7 (which is a requirement for the MorseAPI) because it is easier to manager. However you can download python2.7 yourself with debian and create virtual environments using virtualenv. To use anaconda follow these steps:

1. Install Anaconda: Tutorial for installing anaconda in Debian 12
2. Create a python2.7 conda environment: See how here

Set-up the MorseAPI library:

1. Activate your python2.7 environment (with conda activate - see here)
2. Clone the MorseAPI repository
    a. Install Git - tutorial here
    b. Run *git clone https://github.com/IlyaSukhanov/morseapi.git*
    c. Follow the instruction available on the MorseAPI Github, namely:



Control Dash:

1. Find the bluetooth address of your robot
    a. Turn on the robot
    b. Use your smartphone or connect the bluetooth adapter provided and see if you can connect to Dash
    c. See the bluetooth address and copy it
    d. Disconnect Dash from your smartphone or computer, this is important, the code fails to connect via bluetooth if the robot is already connected to anything
2. Run the code with the command *python dash_test.py BLUETOOTH_ADDRESS_HERE*
    a. Note: If you get any errors try to run the python script with root
        i. Write su on the console and click enter
        ii. Input the password (1234) and click enter
        iii. Activate the conda environment again
        iv. Run the code with the python command above
            1. Code available here

Let's go over what the code does:

All relevant lines of code for the robot have comments, so open the code and follow along with the explanation of its functions. The primary logic is within the 'run()' function. Here, we attempt (a few times) to connect to the robot, then reset it to ensure any anomalies are cleared before the behavior begins.

Initially, the robot greets you by saying "hi" using 'robot-say("hi")', generating a predefined "Hi!" sound. Subsequently, we enter a while loop where the robot checks the direction of the sound. If you talk to the robot, it detects the direction from which the sound originates—an angle ranging from 0 to 360 degrees. To make the robot turn, we instruct it to turn towards the sound direction using 'robot.turn()' and passing the direction as an argument.

The robot can also sense when it's being picked up. We utilize this information to trigger a "aiaiai" sound and initiate movement. Upon detecting that it's picked up, the sound is produced, and we exit the loop, causing the robot to enter a second while loop.

In the second loop, we command the robot to move forward quickly with 'robot.move(1000)'. To detect collisions, we monitor the values of the front sensors (prox_left and prox_right), which range from 0 to 255 units. A value of 255 indicates contact with the sensor. To avoid obstacles, we check if something is detected at 150 units or more. If so, we stop the robot and rotate it 180 degrees.

To end the program, either send a keyboard interruption (like control + c) or close the terminal.

Further documentation and guides:

Official Python library - Look at this one to know what the sensors do!
MorseAPI Library - The code is commented, but no documentation
Python-dash-robot library
Official Dash Apps - Only for drag and drop languages

FAQs:

Want to know what the sensors are doing?

Run the following:
python examples/sensor_readout.py BLUETOOTH_ADDRESS_HERE

Note: If the robot fails to connect, attempt modifying the code to make five connection attempts, as demonstrated in the code provided with this guide.