Modern Robotics practical
ROBOTIC ARM MANUAL

Geert Folkertsma, Ivor Wanders, UT-RaM

2016

This document details how to work with the Robotic Arm during the Modern Robotics Pratical.

1 Hardware & Interface

The robot has three Dynamixel AX-12A servo's, these servos are connected to an Arduino Mega which is built into the base of the arm. The Arduino uses a serial port to communicate to the host PC. This allows it to receive commands and send information back to the PC.

The robot itself requires 12V DC to operate correctly, a power supply of atleast 1.5A must be used. It is a standard 2.1mm center-positive DC plug, so any external harddisk power supply should work.

Connecting the Arduino to the host PC requires a standard USB A-B cable. The drivers for the Arduino serial port should automatically be installed.

1.1 Windows

In Windows the serial ports are named `COM#`, where `#` is the number assigned by Windows.

1.2 Linux

In Linux (Ubuntu and other Debian derivatives), the Serial port should be visible in `/dev/` as `/dev/ttyACM#`, where the `#` is numbered incrementally. By default only `root` and members of the `dialout` group can read and write to this port. You can either add your own user to the `dialout` group, or use `chmod` each time to allow all users to access the port. When using Matlab with a serial port, it is a [known issue] that Matlab requires the Serial port to be available as `/dev/ttyS###` instead of `/dev/ttyACM0`. One can solve this problem by creating a symlink from `ttyS200` to `ttyACM0` (or create an udev rule...). See/run `readme_linux_users.sh` to do this.

1.3 OS X

In OS X the serial port should be available as `/dev/tty.usbmodem####`, where the last numbers correspond to the device number of the USB device. The access rights should be correct by default.

2 Use with Matlab

To use the robot arm with Matlab, several files have been provided to you. These files provide a convenient wrapper for use in Matlab and ensure that the serial port is configured and closed correctly.

The files `ArmInterface.m`, `cStruct.m`, `MCUInterface.m` should be available in your Matlab [`path`], you can add folders to this search path with [`addpath`].

In Matlab you can use ArmInterface.get_serial_ports() to list the serial ports Matlab can see and has the permissions to use. Use this command before and after connecting the robot arm to identify which serial port corresponds to the robot arm.

Then, we can create our ArmInterface object and connect to a serial port with:

```
3  % Create the arm and connect to it
4  arm = ArmInterface();
5  res = arm.open('/dev/ttyS200');
6  pause(1.5) % Give matlab the time to open the serial port.
```

If no errors occur, we can now use our `arm` object to interact with the robot arm. To verify we have connected to the Robot Arm and not some other serial port, we can get the version of the firmware with:

```
8  % Retrieve the arm firmware version and print it.
9  v = arm.get_version();
10 disp(['Arm firmware version: ' num2str(v(1)) '.' num2str(v(2))])
```

If that does not cause errors and reports a version number it means 'the board is green' to start moving the robot arm:

```
15 startpos = 400;
16
17 disp(['Moving to ' num2str(startpos) ', ' num2str(startpos) ', '
       num2str(startpos) ])
18 % Move the arm with default speed to the start positions.
19 arm.set_position([startpos, startpos, startpos]);
```

Or move the arm at a certain speed:

```
26     % Move the joints to the setpoints with the provided speeds
27     arm.set_position_speed([setpoint, setpoint, setpoint], [
           servo_speed, servo_speed, servo_speed]);
```

The current position and speeds can also be retrieved with:

```
29     [actual_pos speed] = arm.get_position_speed();
```

The complete example with all instructions shown here can be found in arm_example.m.