Programming a Robot Arm

Introduction

This idea was provided by Bodge N Hackitt at The MagPi magazine and leans heavily on that work.

Amongst other retailers, Maplin sell a yellow robot arm. At the time of writing it costs £35 (inc. VAT) and that includes the USB interface rather than the manual remote control.

The arm comes with its own software, but I threw that out in favour of programming the robot arm using Python.

The arm has been successfully tested using:

- Windows 7
- Mac OS X 10.8 (Mountain Lion)
- Raspian Wheezy

In other words, it works on Windows, Mac and Raspberry Pis.

The software setup is a bit different for each, and requires a bit of confidence as well as access to the command line. Once the machine is set up you can use the arm WITHOUT the command.

Linux / Raspberry Pi Instructions

You need 2 programs in order to get the arm working

Install libusb

If you are using a Rasbperry Pi then libusb is probably already installed.

If you think you need to install libusb then try the following:

```
sudo apt-cache search libusb
```

This will help you find the correct libusb package to install. It will be something like libusb-1.0.18.

```
sudo apt-get install libusb-1.0.18
```

This will install libusb.

Installing pyusb

At the moment there isn't a pyusb package you can install using apt.

In a webbrowser, go to the PyUSB website (http://sourceforge.net/projects/pyusb/) and download the tar file (big green button). Double click this file and it should unpack into a folder.

At the terminal, go to the folder where you unpacked the pyusb files.

```
cd ~/pyusb-1.0.0b1
```

Configure and install pyusb

```
python setup.py install
```

Programming the arm

Phew! Well done for making it this far!

Paste this into IDLE or in any text editor. Save the file as **army.py**.

```
# ROBOT ARM CONTROL PROGRAM
# import the USB and Time libraries into Python
import usb.core, usb.util, time

# Allocate the name 'RoboArm' to the USB device
RoboArm = usb.core.find(idVendor=0x1267,idProduct =0x0000)

# Check if the arm is detected and warn if not
if RoboArm is None:
    raise ValueError("Arm not found")

# Create a variable for duration
Duration=1

# Define a procedure to execute each movement
def MoveArm(Duration, ArmCmd):
    # Start the movement
    RoboArm.ctrl_transfer(0x40,6,0x100,0,ArmCmd, 1000)
    # Stop movement after waiting specified time
    time.sleep(Duration)
    ArmCmd=[0,0,0]
    RoboArm.ctrl_transfer(0x40,6,0x100,0,ArmCmd, 1000)

# Give the arm some commands
MoveArm(1,[0,1,0]) # Rotate Base Anti-clockwise
MoveArm(1,[0,2,0]) # Rotate Base Clockwise
MoveArm(1,[64,0,0]) # Shoulder Up
MoveArm(1,[128,0,0]) # Shoulder Down
MoveArm(1,[16,0,0]) # Elbow Up
MoveArm(1,[32,0,0]) # Elbow Down
MoveArm(1,[4,0,0]) # Wrist Up
MoveArm(1,[8,0,0]) # Wrist Down
MoveArm(1,[2,0,0]) # Grip Open
MoveArm(1,[1,0,0]) # Grip Close
MoveArm(1,[0,0,1]) # Light On
MoveArm(1,[0,0,0]) # Light Off
```

Either run the program as normal (Save and press F5) or at the Terminal type in:

```
python arm.py
```

You should see the robot are moving around.

You only need to edit the code at the bottom of the program, the lines that look like this:

```
MoveArm(1,[0,1,0])
```

The first number is the duration of the move, in seconds.

The codes for moving the arm work as follows:

Code	Result
`[1,0,0]` or `[2,0,0]`	Open or close the jaws
`[4,0,0]` or `[8,0,0]`	Raise or lower from the wrist
`[16,0,0]` or `[32,0,0]`	Raise or lower from the elbow
`[64,0,0]` or `[128,0,0]`	Raise or lower from the shoulder
`[0,1,0]` or `[0,2,0]`	Rotate clockwise or anti-clockwise
`[0,0,1]`	Turn the light on

You can combine these, so

`[1,1,1]` will open the jaws, rotate clockwise and turn the light on at the same time.
`[5,0,0]` will open the jaws and raise from the wrist at the same time.

You may find that some of the codes are the wrong way round - this usually means you have wired up or fitted the motors back to front!