1. Requirements

To perform tests with this first release, you will need:

- A recent Ubuntu workstation (denoted “PC” below). The latest Ubuntu LTS is recommended (currently LTS 14.04)
- A Bluetooth 3.1 adapter (tests were done at Parrot using an ASUS-BT400 USB dongle)
- Ubuntu Bluetooth stack. Usually installed using the following command:

```
sudo apt-get install bluez-compat
```

- A USB cable to connect the RollingSpider to the Ubuntu station
- A GCC toolchain with glibC. Provided with the firmware.
- A Parrot Rolling Spider ! (denoted RS below)

2. Installing the customized firmware

The firmware will be installed using the same procedure as normal firmware updates done by customers.

- Connect the RS to any Windows or Ubuntu station with the USB cable
- The RS should show as a USB mass-storage device (ie. like a pendrive)
  Note: on a Ubuntu station, the RS also shows a network interface, as seen in section 3.1.
- Copy/paste the **.plf** file in the root folder of the shown mass-storage device.
- Disconnect the mass-storage device using the safe removal procedure specific to your OS.
- Insert a battery in the RS if not already done
- Unplug the USB cable
- Wait for the update procedure to complete (blinking LEDs indicate the procedure progress; please refer t the user manual)
- Installation success can be checked by connecting the system shell (see section 3.1) and checking that the shell prompt says `[RS.edu]`

3. Getting access to the embedded system

3.1 Using the USB cable

The RS provides a RNDIS network interface, which allows to setup up a TCP/IP link with the Ubuntu station.

- Connect the RollingSpider to the PC with the USB cable.
- The RS should power on. If not, press the ON/OFF button
- The Ubuntu Network manager should show that a new network link is available.
  The *ifconfig* command should show a new interface usb0:

```
usb0      Link encap:Ethernet  HWaddr d2:ed:10:d2:fe:f1
          inet addr:192.168.2.2  Bcast:192.168.2.255  Mask:255.255.255.0
```

- To access the RS embedded shell, connect to the RS telnet server at IP address 192.168.**2.1** (default port). You will be welcomed by:

```
BusyBox v1.20.2 (2015-01-16 14:28:29 CET) built-in shell (ash)
Enter 'help' for a list of built-in commands.

-----------------------------------
HW Status :
-----------------------------------
 > Acc/Gyros MPU6050       :OK
 > Temp/Press MS5607       :OK

[RS.edu] $
```

3.2 Using the BT link

The RS provides a BNEP Bluetooth interface, which allows to setup up a TCP/IP link with the Ubuntu station.

- Find out the RS bluetooth MAC address
  - Either by connecting to the shell via USB (see above) and reading the /data/btconfig file. Example:

```
$ cat /data/btconfig
BDADDR=A0:14:3D:0D:5A:C3
BTNAME=RS_W000120
```

- or by scanning BT devices from the PC, for example using hcitool:

```
sudo hcitool scan
Scanning ...
      00:12:1C:00:00:F4 RS_W000120
```

- Connect the PC to the RS. With the above mac address as an example

```
sudo pand --connect 00:12:1C:00:00:F4 -dGN -n
sudo ifconfig bnep0 192.168.1.2 up
```

- To access the RS embedded shell, connect to the RS telnet server at IP address 192.168.**1.1.**
  You will be welcomed by the same messages as shown in section 2.1.

4. Insert customized control code

4.1 Build and execute the code

- Uncompress educode.tar.bz2
- Type 'make' to build the librsedu.so file
- Connect to RS using USB or BT
- Connect to RS with an FTP client – the root folder should contain a README.txt file telling you this is where to put the librsedu.so file.
- Send the librsedu.so file
- When the transfer is complete, the RS should start executing your custom control code

4.2 Check code execution

- Connect to the RS shell using USB or BT
- Stop the execution of the main process by executing the command “kk”
- Restart the main process by typing “dragon-prog”
- You should now see the debug messages from the main process, including prints from your customized code.