Activities

Part 1 – Your Kit

1. Check that all of the correct parts are in the box, the contents list in the rules is wrong. Refer to the website for a correct list.
2. Then show an HR-RoboCon official who will give you the password for the next phase.

Part 2 – Using the kit

3. Plug in the tablet to power and connect the battery to the brain box. Then connect the black connector to the on/off.
4. Connect the tablet to the brain box using the password that was given to you after step 2.
5. Open chrome and go to http://robot.sr
6. Open IDLE. Find the file called 'Hello World' on the desktop. You should change the code to print out something of your choice. Then upload and run the code by doing the following:

   Save the program on the desktop then run the program by uploading onto the website (http://robot.sr) via the "upload" link and then press the "Choose File" button, then press "upload".

   Return to the home page of the website, click the "run" link and then press the "start" button.

7. Show this to an HR-RoboCon official who will give you a screwdriver for the next part.

Part 3 – GPIOs as outputs

8. Create a new python file in your editor.
9. At the beginning of the script write the following import statements:

```
import time

from sr.robot import *
```

10. Initialise the Robot and assign the LED to a GPIO pin using the following code:

```
R = Robot() #initialises the Robot

LED_PIN = 2 #Sets the LED to GPIO pin 2
```

11. Set the GPIO pin to output.

```
R.gpio.pin_mode(<PIN>, <MODE>)
```

where <PIN> is the GPIO Pin you are using and <MODE> is either input or output.

```
R.gpio.digital_write(<PIN>, True)
```

12. Then, wait for 1 second using the line:

```
time.sleep(<SECONDS>)
```

13. Use a while loop to make this repeat.
14. You can then use these lines to change the LED however you like. Try making it flash on and off for a bit.
15. Show this to an HR-RoboCon official, who will not need to give you anything for the next part.

Part 4a – GPIOs as digital inputs

16. Connect one of your power switches to GPIO Pin 4 and the ground connection.
17. Create a new python file and initialise your robot (as described in step 9 and 10).
18. Set up pin 1 as a digital input.

```
R.gpio.pin_mode(4, INPUT_PULLUP)
```

19. To read the pin you need to call the `digital_read(<PIN>)` function which will return the value of the pin. Assign it to a variable as below:

```
reading = R.gpio.digital_read(4)
```

20. This will assign the value to "reading". If you want to see the value of reading you can print it like so:

```
print reading
```

21. Place the read function in a 'while True' loop so that we are constantly sampling and printing our sample.

    It is probably a good idea to put a sleep at the end of the loop as the program doesn't need to sample that much.

```
while True:
    reading = R.gpio.digital_read(4)
    print reading
    time.sleep(0.1)
```

22. Run the code using the robot.sr website.
23. Then show this to a HR-RoboCon official, who will not need to give you anything for the next part.

Part 4b – GPIOs as analog inputs

24. Connect the potentiometer to the GPIO connecter (see information sheet to find where they are):

- Black goes to ground
- Green goes to GPIO 1
- Red goes to 5V

25. The wires will be connected by inserting them into the empty ports on the GPIO plug. Tighten the screw above to secure them.
26. Plug the connecter into the Pi. (Screwdrivers can be collected from an HR-RoboCon official).
27. Open either IDLE and create a new file Analog.py and import time and everything in the 'sr.robot' library.
28. Call the robot library and initialise it as before.
29. Set up pin 1 as an analogue input. This configures pin 1 to take be able to take readings.

```
R.gpio.pin_mode(1, INPUT_ANALOG)
```

30. The syntax is as follows `R.gpio.pin_mode(<PIN>,<MODE>)`.
31. To read the pin you need to call the `analog_read(<PIN>)` function which will return the value of the pin. Assign it to a variable as below:

```
reading = R.gpio.analog_read(1)
```

32. This will assign the value to “reading”. If you want to see the value of reading you can print it like so.

```
print reading
```

33. Place the read function in a while True loop so that we are constantly sampling and printing our sample.
    - It is probably a good idea to put a sleep at the end of the loop as the program doesn’t need to sample that much.

```
while True:
    reading = R.gpio.analog_read(1)
    print reading
    time.sleep(0.1)
```

34. Save the program on the desktop then run the program by uploading onto the website via the “upload” link and then press the “Choose File” button, then press “upload”. Return to the home page of the website, click the “run” link and then press the “start” button. Turn the potentiometer and you should see the number of the screen change.

35. (a) Now let’s use this value to make something happen, for example turn on an LED after a certain voltage.
36. (b) Connect an LED’s positive leg (the long one) to GPIO 2 and the shorter one in the same port as the black wire on the potentiometer (leave the potentiometer wire in the port).

Set it as an output and set it low, before the while loop.

```
    R.gpio.pin_mode(2, OUTPUT)

    R.gpio.digital_write(2, 0)
```

Now in the while loop, but after you take the reading, create an if statement that will check if the value of pin 1 is more than 800. If it is, set pin 2 to high, else it sets it low:

```
if (Reading > 800):

    R.gpio.digital_write(2, 1)

else:

    R.gpio.digital_write(2, 0)
```

Now run the code and turn your potentiometer. The LED should turn on and off depending on the position of the potentiometer.

37. Show this to a RoboCon official who will give you a DC motor.

Part 5 – Motor control

38. Controlling a motor is fairly simple. The syntax for setting motor power is as follows, where `<CONNECTION>` is the index of the I^2C connection you want to use (unless you've added your own power board, that's '`0`'), and `<INDEX>` is the index of the motor (`0` for physical connection `M1` on the Brain Box, `1` for physical connection `M2`):

    ```
    R.motors[<CONNECTION>].m<INDEX>.power = <0-70>
    ```

    For example, to set the motor at physical connection `M1` to full throttle, the command would be:

    ```
    R.motors[0].m0.power = 70
    ```

39. Try and get a motor to spin at full speed for 10 seconds before coming to a stop.
40. Now see if you can get one to spin forwards at full speed for five seconds, then backwards at half speed for ten.
41. Show this to a RoboCon official who will give you a servo.

Part 5 – Servo control

42. Controlling a servo is very straightforward. The syntax for setting servo position is as follows, where <INDEX> is the index of the servo you want to use:

    ```
    R.servos[<INDEX>] = <-100 to 100>
    ```

For example, to set the servo at index 0 to the centre, the command would be:

```
R.servos[0] = 0
```

43. Try setting a servo to position 100, then back to 0 again.
44. Now connect up another servo, and set one to -75 and the other to 75.
45. Try writing a program that will set a servo back and forth from 100 to -100 every two seconds in an infinite loop.

Extension!

If you're feeling happy with the above, try the following extension questions (Don't worry if you can't do them, they're pretty tough!):

43. Turn an LED on/off with the black button in your kit.
44. Change the brightness of an LED to 50%.
45. Get the LED's brightness to be affected by the input voltage from a potentiometer.