Exercises 5:

First Hardware Exercise: The LED

Goal:

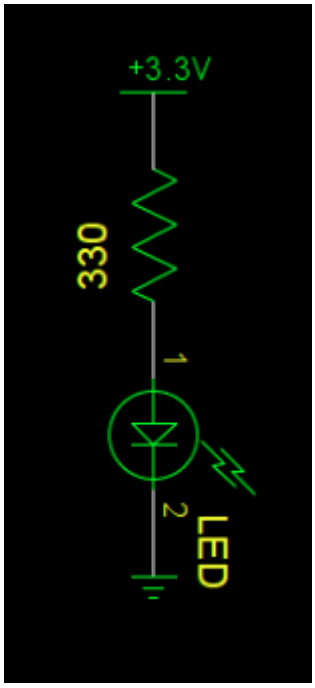
Now that we know how to program the Raspberry Pi we will start to setup our first electronic circuits, albeit extremely simple, and control them through the Raspberry Pi.

All circuits will be built on a bread board which is connected to the Rpi through a “cobbler” which supplies it with power and which gives access to the Rpis I/O pins. The significance of the pins is printed on the cobbler.

A word of warning: The Rpi can provide a maximum of 60 mA of current. Drawing more than that may damage the processor chip and thus kill our poor little Raspberry. **Therefore only wire up your circuit with the Rpi switched off! and triple control your wiring, before switching it on. When unsure, please ask a supervisor! If the Raspberry Pi was on, then shut it down orderly before switching off.**

Exercise 1: Testing the LED

As said above, we have to limit the current flowing through the LED, which is done with a 330Ω resistor as shown in the circuit diagram:



The long lead on the LED goes to Vcc (via the resistor) the short one to ground.

Under the assumption that the resistance of the LED can be neglected compared to the 330 Ω current limiting resistor and Vcc = 3.3V, what is the current flowing through the LED?

Exercise 2: Connect the LED to the GPIO pin

Now that we know the LED works fine, we disconnect (with Raspberry Pi switched off!) the LED from Vcc and connect it to GPIO pin 0 instead. Please check the pin correspondence table between GPIO numbering and physical numbering on the CPU chip.

We have installed the wiringpi package for you, which contains libraries for access to the GPIO pins but also to the SPI and I2C serial interfaces. This package contains in addition to the library and include files a new command *gpio* allowing us to talk to a gpio pin. We will have to define this pin to be output and then write 0 or 1 to it .

Have to look at the gpio man page to figure out how to do this.

Exercise 3: Shell script to make the LED blink

Once you are able to switch a LED on and off, write a shell script to make it permanently blink.

Exercise 4: Write a C program to make the LED blink

In order to translate your blinking program into C you will have to include <wiringPi.h>. Then you can call *wiringPiSetup ()* to initialize the library. Finally use *pinMode and digitalWrite* to tel the library you want to use the pin as output pin and write to it. Check the wiringpi API documentation for the parameters of these calls. When building the program you have to link the wiringPi library to your program (-lwiringPi). The wiringPi include files and the library are stored in /usr/include and /usr/lib respectively.

Write a Makefile to build your project. Test it to make sure it works as expected.

Exercise 5: Make the blink program exit gracefully

Include a signal handler in your code which handles the SIGINT signal. When called it will switch off the LED before finally exiting the blink program.

Exercise 6: Make your LED blink SOS

Write a C program that makes the LED blink an SOS. Write a subroutine pulse (int pulseLength) that switches the LED on for pulsLength ms and then off again for pulseLength ms. Use this routine to implement the SOS. Wait for 1 s between each SOS sequence.

Exercise 7: Traffic Light Simulation

Implement the traffic light simulator on the Raspberry Pi using 6 LEDs: 2 red, 2 yellow, 2 green to simulate the 2 traffic lights. Look up the problem description in exercise 3.