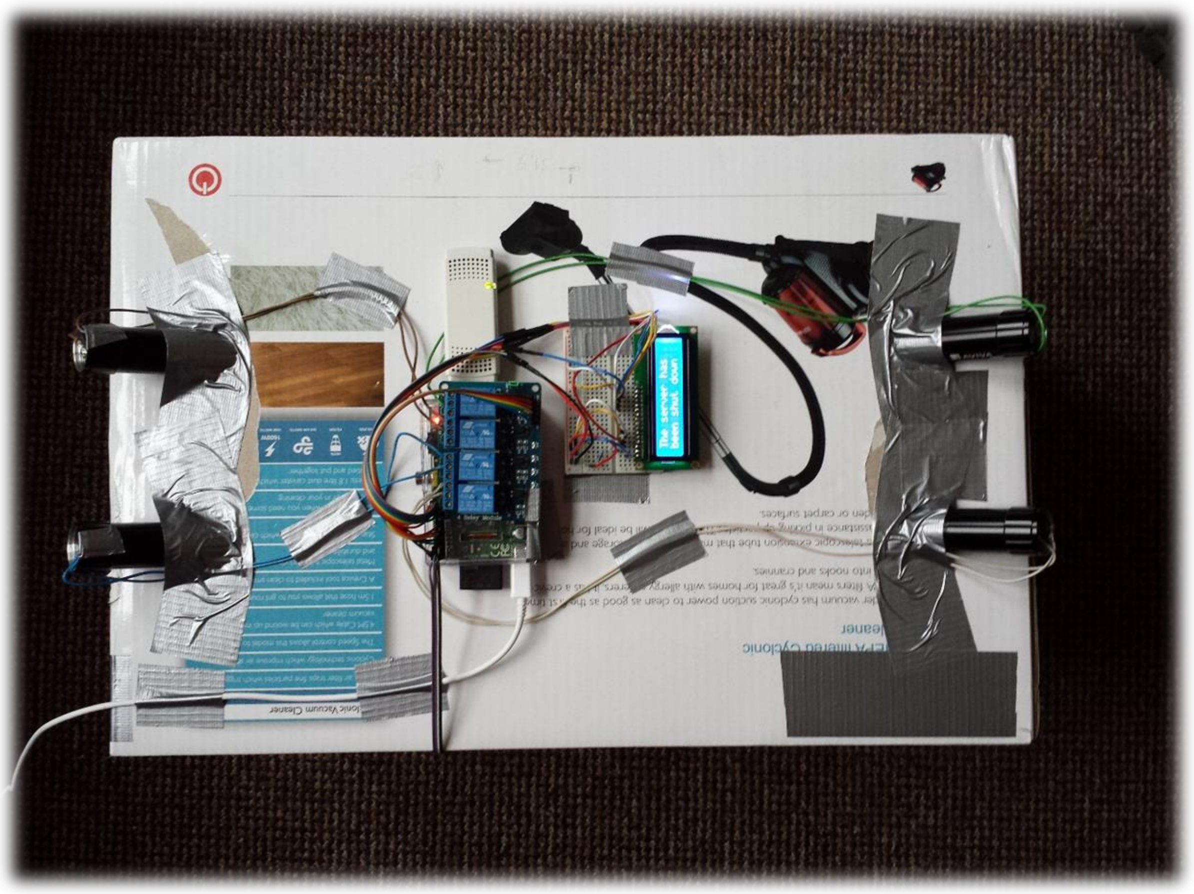
Raspberry Pi Multi-purpose Automated Light Control System

By Aleksander Antoniewicz and Plamen Kolev

School of Computing Science, Newcastle University



Summary

The main aims of this environmentally friendly project is to both save the electricity consumption on lights and automate the control of the illumination of a premises.
This particular combination of hardware and software used makes the system extremely versatile and easy to personalise as well as customise. In our project we simulate room conditions in an average house, but it is possible to install this system in office spaces, small shops and other premises having large window area.

It is quite important to save electricity, as this may significantly reduce monthly bills and help the environment, especially in countries with little energy coming from renewable resources. In order to minimise electrical energy loss, we decided to choose a relay board acting as a control unit over a variable resistor. This solution features much lower energy waste, but does not provide a large scale of luminosity differentiation. This can be however overcome by using multi-channel relay boards. Our project uses a four-channel board, being a reasonable balance between price and capabilities for everyday household use.
We used a cardboard box to simulate average home room environment. There are two coverable holes at opposite sides of the box, which act as windows.

The relay board is controlled by General Purpose Input Output pins of Raspberry Pi. In our model, we used evenly spaced LED torches to simulate ceiling lamps. Their switches have been removed and cables connecting torches with the relay board were installed instead. When Raspberry Pi sends a signal to the relay board, the board acts as a switch, enabling each of the torches.

There are two operation modes, automatic and manual. The first one controls the torches based on the luminosity detected by a sensor placed at the bottom of the box.
The second one enables user to turn on and off each of the lamps through a website. We also used a 16 character, 2 line LCD for debugging and informative purposes. Thanks to a Wi-Fi dongle, it is possible to mount the whole circuit far away from Local Area Network access point.

List of hardware components used and their prices:

Raspberry Pi Model B Rev.2	GBP 33.00
4 LED torches	GBP 6.00
5V 2A power adaptor adaptor	GBP 10.00
5V 1A power	GBP 7.00
TSL2561 light sensor	GBP 13.00
16x2 LCD	GBP 3.00
4 channel relay board	GBP 5.50
breadboard	GBP 5.00
cables	GBP 8.00
Atheros chipset-based Wi-Fi dongle	GBP 8.50
TOTAL EXCLUDING RASPBERRY PI	**GBP 66.00**
TOTAL EXCLUDING ABOVE AND LIGHTING FOR TEST PURPOSES	**GBP 53.00**

The automation script is written in Python and launched through WebIOPi HTTP server. The script is accessed using macros through JavaScript directly from the webpage. Thanks to using a website over dedicated application, it is possible to control lighting using Personal Computers, smartphones, SmartTVs, tablets and many other devices.

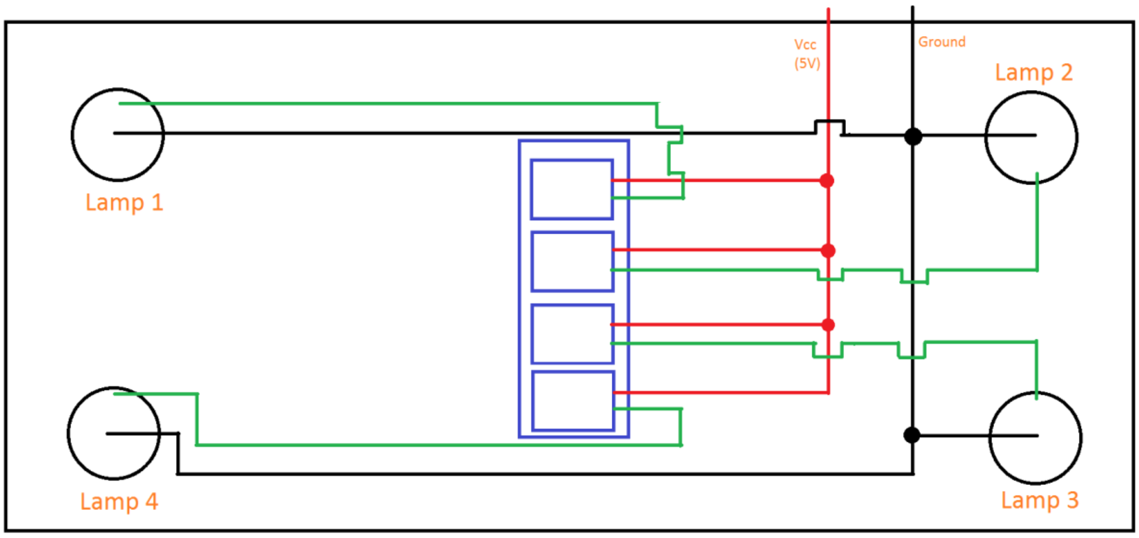
Hardware

We decided to use a cardboard box to be able to use low power lamps, in this case 1W LED lamps. They were powered by 5V 1A standard USB charger, while the Raspberry Pi was connected to another 5V 2A charger. Thanks to replacing batteries with the first one, the luminosity of each of the torches is now stable and not subject to discharging electricity source. To ensure maximum precision we used the TSL2561 electronic luminosity sensor. Thanks to this, Raspberry Pi receives processed information, i.e. light intensity measured in lux, directly from the sensor, without the need for complex processing. The debug device used is a generic 16-pin LCD which displays welcome information, IP address of the WLAN card, and, after server start, the luminosity read by the sensor and the current luminosity range used by automatic control software. The LCD also informs the user about server shutdown.

A 4 channel relay board is operated through GPIOs and is powered using a 5V pin from the Raspberry Pi. Every channel has a corresponding torch connected to it, but in a more complex design it is possible to connect multiple lamps to one relay. Each relay is capable of withstanding 250V/10A AC, which makes it extremely versatile not only in case of lighting systems. Each relay has a corresponding LED shining when the throughput is enabled.

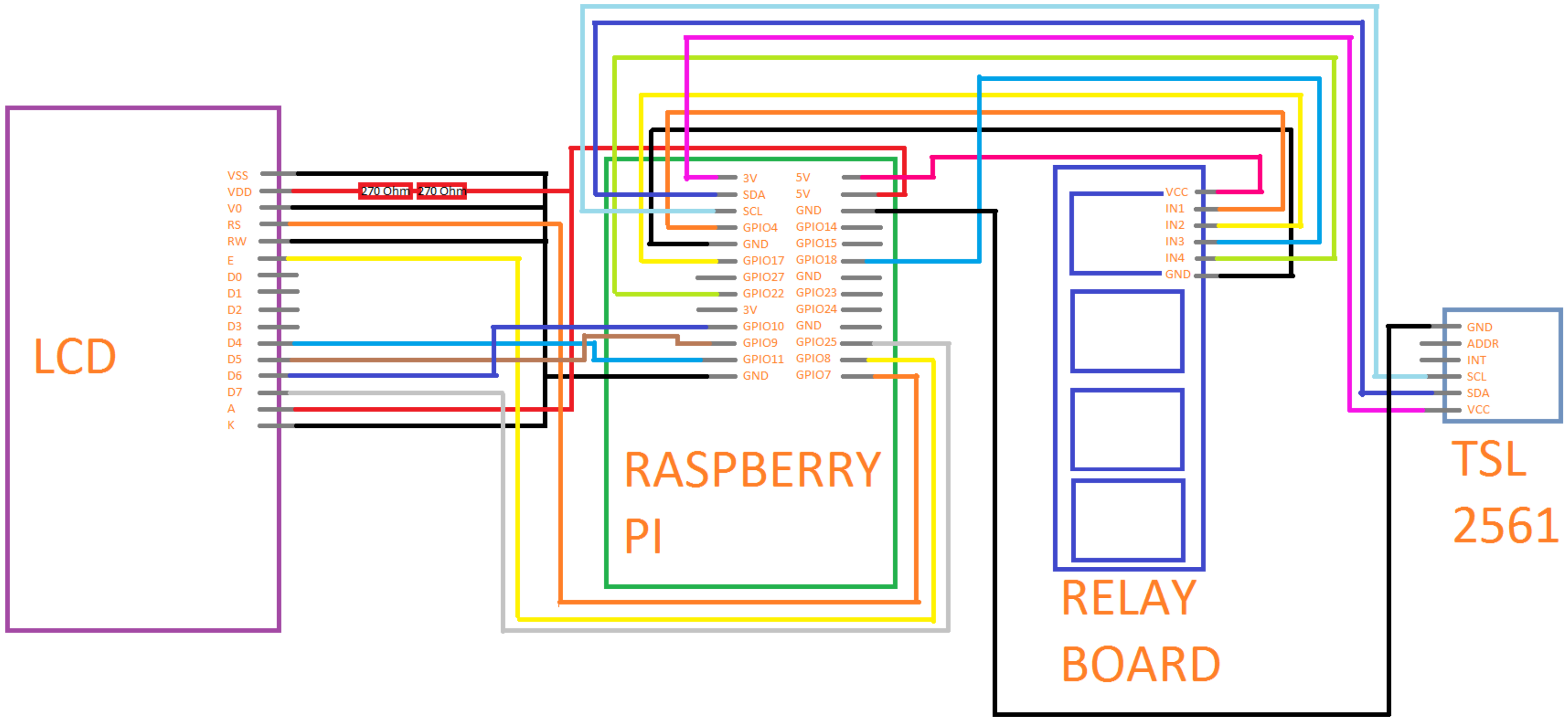
The Wi-Fi dongle makes our design quite portable. This means that our design, when implemented in a real room, would only require mains socket and to be within the range of an access point.

The image below shows how the lamps have been connected to the power supply and the relay board.



The next image shows connections between Raspberry Pi, relay board, LCD and luminosity sensor.

`


LCD
VSS
VDD
V0
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
A
K
270 Ohm
270 Ohm
RASPBERRY PI
3V
SDA
SCL
GPIO4
GND
GPIO17
GPIO27
GPIO22
3V
GPIO10
GPIO9
GPIO11
GND
5V
5V
GND
GPIO14
GPIO15
GPIO18
GND
GPIO23
GPIO24
GND
GPIO25
GPIO8
GPIO7
RELAY BOARD
VCC
IN1
IN2
IN3
IN4
GND
TSL 2561
GND
ADDR
INT
SCL
SDA
VCC

Software

Script2.py is the most essential script to the project, it is being called automatically when the WebIOPi starts. It does the basic math and automation of the lights. In the beginning we have initialized the global variable `threshold` that is used in the loop function later on for setting the light range.

Then we initialize the LCD variables, setting the pi to automatic controls.

The Setup function is being automatically called on startup, it sets the pins of the Pi to output, sets up the LCD addresses and prints to the display that the server has been started and then shows loading. After that the IP is shown for a short duration.

The most important function in the project is the loop function. After setting up everything this loop executes continuously. Here we use the variable `threshold` to represent the range of light currently received. We use a variable called `lights_On` to keep track of how many lights are shining inside the environment, dynamically adjusting the range, we compensate for the extra luminosity.

The `luminosity` variable is set by the luminosity sensor, then the if statement checks `autoEnable` if the server is set to operate the lights automatically. Next we check how many lights are on by using the `GPIO.digitalRead` (the relay board turns on a light when the signal of the PI is LOW and vice versa).

After counting the lights, the `threshold` global variable is set to a fixed value times the `lights_On` for dynamically adjusting the algorithm.

Then if-else statements compare the relationship between the `threshold` and `luminosity`, depending on their values one of these statements is executed and sets the GPIO pints to high or low, effectively turning on or off lights. Then this algorithm is executed over and over again adjusting for the range of the changing environment. We also display the luminosity and the range according to it on the LCD

List of software used:

- WebIOPi framework[1]
- LCD script[2]
- Python libraries

[1] https://code.google.com/p/webiopi/
[2] http://www.youtube.com/user/RaspberryPiBeginners

Links

YouTube video: http://www.youtube.com/watch?v=U18SBJ2zLDo

Python script: http://pastebin.com/UffJQvyN

HTML/Javascript document: http://pastebin.com/1pgsegKv