RaspBerry Pi 2 miniproject

Purpose: Full example on how to create a RaspBerry Pi project application and explaining on how-to control a servo motor and LED.

What is used in this project:

- Laptop
- RaspBerry Pi 2 model B
- Network connection between Laptop <-> RaspBerry (wireless)
- Servo motor (open/close dispenser)
- LED (signaling)
- wiringPi library (http://wiringpi.com/)

Steps:

1. As first start to play around with the RaspBerry Pi by creating small scripts to fully understand how it works. Found in:
`/Milestone_1/py_scripts && c_scripts`

2. The miniproject intention is a small implementation in order to implement a further project in order to control a food dispenser. Path of miniproject:
`/c_scripts/servo_motor.c`

3. I found a library that deals better with servo motors and installed it. After that I implemented it in the project:

```
#include <wiringPi.h>
#include <softPwm.h>
```

4. I chosen and defined the gpio pins (also seen in /Milestone_1/Raspberry-Pi2-GPIO-Layout.png) that will be used for the servo motor and LED as follows:

```
#define MOTOR 18
#define LED 17
```

5. Setup the needed pins as outputs:

```
pinMode(MOTOR, OUTPUT);
pinMode(LED, OUTPUT);
```

6. The servo motor needs to be given a certain impulse in order to move to a certain position. Thus, Creating a function that simulates the open-close dispenser and signaling:

```
digitalWrite(LED, HIGH);          //LED on (signal opening)
softPwmCreate(MOTOR, 0, 100);     //create a software pwm
softPwmWrite(MOTOR, 10);          //motor moves left (open)
delay(1000);                      //for 1 second

softPwmWrite(MOTOR, 0);           //motor stops
delay(5000);                      //for 5 seconds
```

```
softPwmWrite(MOTOR, 30);          //motor moves right (close)
delay(1000);                      //for 1 second

softPwmWrite(MOTOR, 0);           //motor stops again
digitalWrite(LED, LOW);           //LED off
```

The full program can be seen in /Milestone_1/c_scrypts/motor.c

7. Next it is needed to build the c program that we have created. To compile we make use of gcc, thus entering in the terminal:
`#gcc -o motor motor.c -lwiringPi`

8. Time to execute the program with superuser and see actual physical output:
`#sudo ./motor`

9. Final setup can be seen in figure bellow or in: /Milestone_1/motor.jpg , note that the LED isn't connected, because I lacked some wires, but will update on that and also make a video demo.

10. Further work:

- connect through nabto platform to remote control the motor
- implementation in c++, using classes
- nabto device application
- nabto client application
- further physical implementation, actual opening a small door