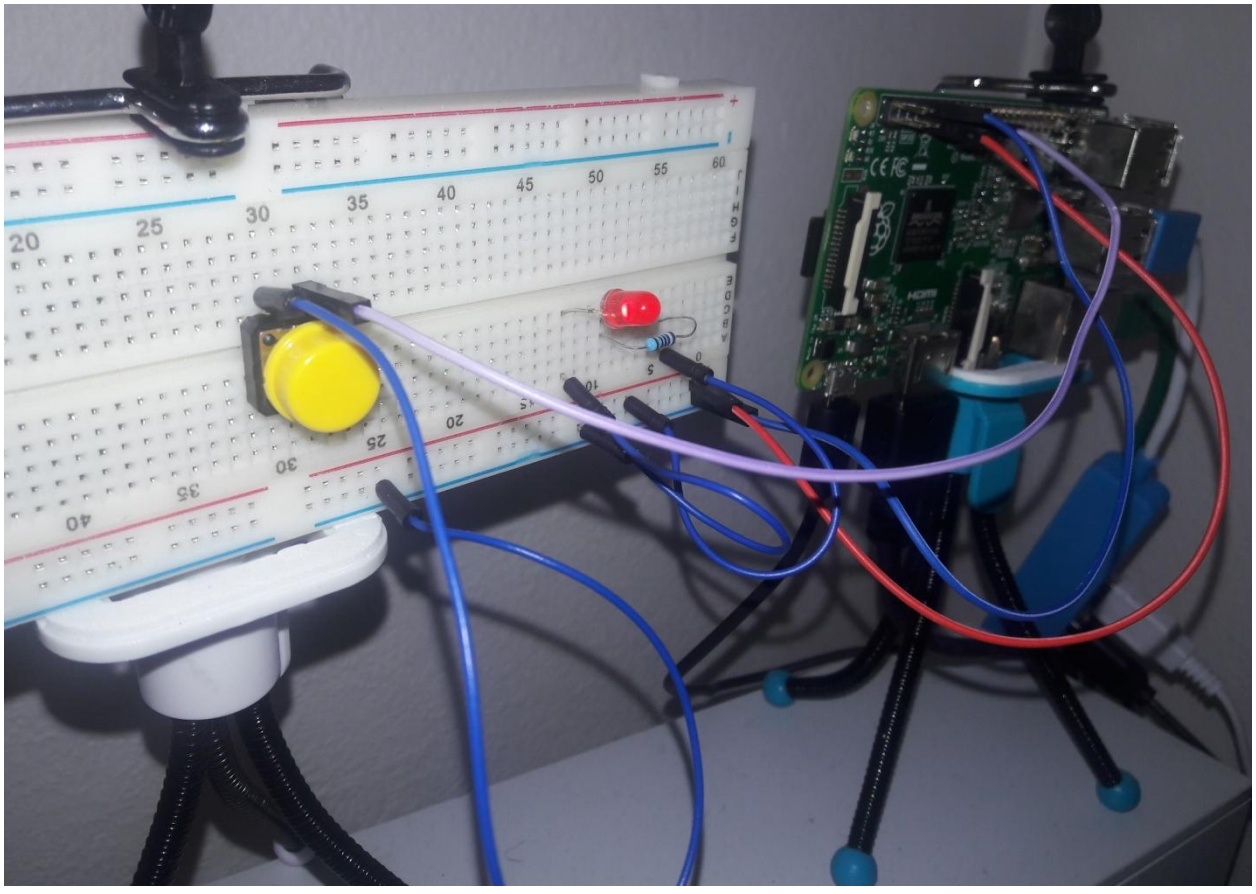
Switching an LED and Getting Feedback over the internet (Basic IoT) using a blend of Python, PHP, HTML, CSS, and Text Files

Baziramwabo Gabriel, S16147

1. Introduction

Encouraged by Professor Markon Sandor in the course of Computer Programming Exercises [2209] where he emphasizes on Python as a very resourceful programming language, and still sticking on PHP as my programming language for half a decade, I decided to use that opportunity to give a try Internet of Things using Raspberry Pi. This project has been a breakthrough to me. I had always looked in vain for a simple way of controlling things over the internet. It now seems to me like what I lacked was only the urge!

Figure 1: Project picture



2. Tools used for this project

2.1. Hardware:

- Raspberry Pi 2 with memory card preloaded with Linux OS
- Red LED
- 220Ω Resistor
- 4 jumper cables (2 of male-male type, and other 2 of male-female type)
- breadboard
- Ethernet cable for connecting the raspberry pi to the internet
- USB Cable for power-supplying the Raspberry Pi
- Mouse for the Raspberry Pi
- Keyboard for the Raspberry Pi
- Screen with HDMI Port
- HDMI Cable for connecting the Raspberry Pi to the screen

2.2. Software:

- Linux/Rasbian as Operating System on the Raspberry Pi
- Python as a Programming Language on the Raspberry Pi
- PHP as a Programming Language on the Web Server
- HTML/CSS for supporting a nice user Graphical Interface

3. What does this system simply do?

It is a simple project to test basics of IoT. It does what it is supposed to do. It is not yet a big deal though.

3.1. Switching an LED over the internet:

On the breadboard is a red LED connected to the GPIO Pin of the Raspberry Pi for testing the switching of things over the internet. Yes, beyond the Local Area Network**!** Using a computer or a mobile phone, any person (no authentication so far) opens a browser and types in ***yellowpagesrwanda.com/iot/*** (that is my personal website I hire for testing my web based projects). The user gets simply a button Turn On/Turn Off. The button is displayed as "Turn Off" when the LED is already ON and it becomes "Turn On" when the LED has been turned off.

3.2. Getting Feedback over the internet:

On the same breadboard is a yellow Push Button to test the feedback over the internet. In fact, in the real world projects we need to be informed by sensors on the status of our things located on the other side of the cloud.

Pushing that yellow button will trigger the sending data from the Raspberry Pi to the web server where they will be received by server-receiver. In this project I preferred to take advantage of the current status of an LED to toggle its status. It means, when the LED is ON and then the button is pushed, the LED should be go OFF and vice-versa. That is something that can be easily done without internet. However, here we're testing the monitoring over the internet. Otherwise it would be nonsense.

4. How does it work technically?

This system is made up of two parts:

Part One: Web Client which is the Raspberry Pi where Python is running in two separate files: **raspi-receiver.py** and **raspi-transmitter.py**. There is also a text file **ledStatus.txt** where the **raspi-receiver.py** writes and then the **raspi-transmitter.py** will later on read.

Part Two: Web server which hosts PHP, HTML, CSS and Text File. There are files **index.html**, **style.css**, **server-reciever.php**, **server-transmitter.php**, and **commands.txt**

4.1. Switching ON/OFF

The Python codes in **raspi-receiver.py** on the Raspberry Pi do the following:

a. Sending a request to the server targeting a text file **commands.txt** which stores the commands sent by the user via **server-transmitter.php**

b. Listening to the server response

c. If the message read from the server commands to turn on the LED a HIGH Signal will be sent to the GPIO pin on which the anode (+) of the LED is connected

d. If the message read from the server commands to turn off the LED a LOW Signal will be sent to the GPIO pin on which the anode (+) of the LED is connected

e. Whenever a signal is sent to the GPIO Pin to switch ON or OFF the LED, the *actual* LED status is saved stored into the file **ledStatus.txt**

4.2. Feedback Test

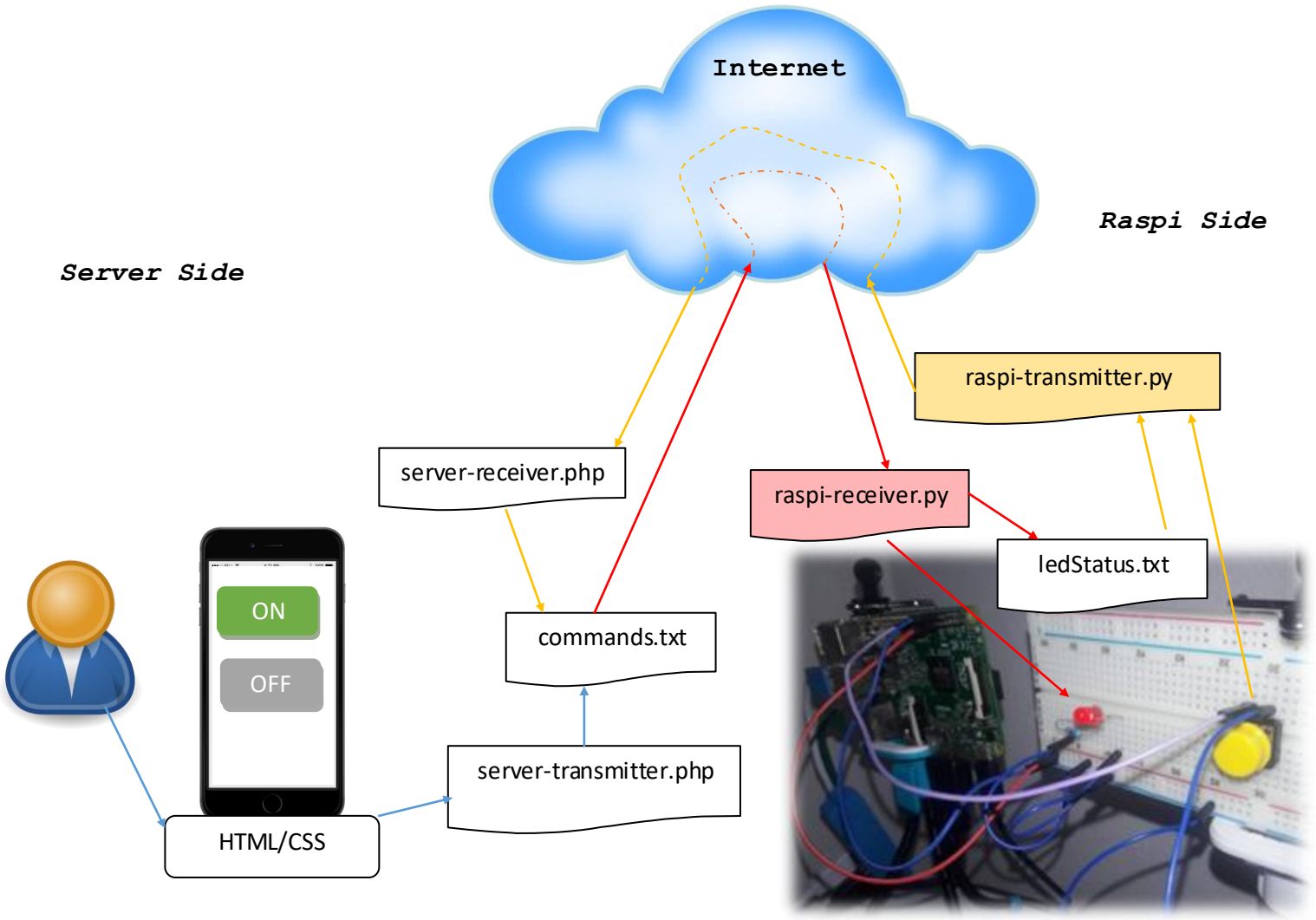
The Python codes in **raspi-transmitter.py** on the Raspberry Pi do the following:

a. Waiting for the button to be pushed

b. When the button is pushed the code gets the current time, the local IP address, and the current status of the LED as read from the file **ledStatus.txt.**

c. Collected data will be taken to the server where they will be received by **server-receiver.php** which will update the file **commands.txt** in a way to toggle the current LED status.

Figure 2: Flow Diagram



5. Where to go from here?

I have grabbed the practical working principal of IoT. It is the begenning and not the end. In the Lab assigned to me I will dive more into IoT Innovations.

I'm thankful to Markon Sensei who introduced Python to us and who urged us to do something practical no matter how simple it would look.

6. Source Code

Server-side: index.html, style.css, server-transmitter.php, server-receiver.php, commands.txt

Raspi-side: raspi-receiver.py, raspi-transmitter.py, ledStatus.txt