IMPLEMENTATION OF RASPBERRY PI SERVER FOR TYPING TOURNEYS

Atul Motilal Tiwari, Priya

1UG Student, 2Assistant Professor
Department of Electrical and Electronics Engineering, Saveetha School of Engineering, Saveetha University, Chennai, India.
Email: atul.tiwari310@gmail.com

Received on: 15.08.2016
Accepted on: 20.09.2016

Abstract:

The Raspberry Pi is a low cost single-board computer which has recently became very popular. In this paper we showcase our attempt in building a low cost server which is capable of conducting typing tourneys between connected users using raspberry pi. The server works by setting up tourney every 5 minutes and advertises it, which can be viewed by users using a client side application. The users can also create custom tourney with a authentication key, only users with authentication key can only access that tourney. After required users are connected, the tourney starts with one-minute duration. At the end of tourney, the user with highest typed words is considered as winner. Raspberry Pi runs a modified version of Linux for ARM Architecture and uses socket programming for setting up server and client side application with the help of python programming language. The client side application is written using kivy library.

Fig 1. Raspberry Pi Model B Plus.

A wireless local area network (WLAN) is a wireless
I. Introduction

The Raspberry Pi as shown in figure 1 is a low cost credit-size single-board computer with ARM architecture. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. It runs Linux a free operating system from a SD card and it is powered by a USB phone charger. It is currently having 5 different models; Model A, A+, B, B+, Generation 2 Model B. Model A, A+, B, B+ all are powered by 700 MHZ single-core ARM1176JZF-S, whereas the new 2nd generation Model B is powered by 900 MHZ quad-core ARM Cortex-A7. Model A and A+ have 256 of RAM, Model B and B+ have 512MB of Ram and 2nd Generation Model B has 1GB of RAM. These Pi have USB port within them and have ability to do networking. All of these Models are powered by Broadcom Video Core IV @ 250 MHz and power source is a 5 V via Micro-USB or a GPIO header. In this project we are using model B+. computer network that links two or more devices using a wireless distribution method (often spread-spectrum or OFDM radio) within a limited area. This gives users the ability to move around within a local coverage area and still be connected to the network, and can provide a connection to the wider Internet. Most modern WLANs are based on IEEE 802.11 standards, marketed under the Wi-Fi brand name.

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.

Kivy is an open source Python library for developing mobile apps and other multi touch application software with a natural user interface. It can be run on Android, iOS, Linux, OSX and Windows.

II. Interfacing Wlan Adapter With Pi

The Wireless adapter is connected to pi via a 2.0 USB port. We used a high gain TP-LINK (TL-WN722N) model with 150 Mbps rate speed. The adapter by default supports the Pi without any further configuration. The adapter does not use too much power thus keeping the Pi alive.

III. Setting Up Server

After interfacing of wireless adapter with raspberry pi, it is instructed to setup a wireless access point(hotspot). This access point is open and easily viewable by other devices. We have used Transmission Control Protocol because of its
reliable data stream for setting up server.

The features of server are

1. To conduct tourney every 5 Minutes
2. To conduct custom tourney created by user
3. Chat system for communication between users Tourney can be automatically created using the following way.

```python
def automatic_tourney(): stop_time = time.time() + 300 users = 0
    while time.time() <= stop_time: # accept_users
        users += 1
        if users >= 2: conduct_competition()
    return
```

The score. The score is calculated by comparing the original text and user typed text. The number of correct matching words gives the score. The score calculation can be understood by following way.

```python
def calculate_score(original_text, user_text):
    score = 0
    end = min(len(original_text), len(user_text))
    for word in range(end):
        if original_text[word] == user_text[word]: score += 1
    return score
```

The server broadcasts messages to all the connected users, like for example if a new tourney is created, the server sends this message to every user connected. This process can be implemented in following way.

```python
def broadcast(clients, message):
    for client in clients:
        client.send(message)
```

In above function, argument clients is a list containing socket object of all the clients connected to server. Then we iterate over clients list and use “send” method to send the required message. We can use exception handling to deal with any error while sending message.

**IV Setting Up Client Side Application**

There will be a loop running for 5 minutes, which will accept any registrations from user. After 5 minutes the loop breaks and checks if minimum required amount of users are registered which is two. If less than two are connected the competition doesn't start.

After the starting of competition, each user is given a text to type. This text can be viewed in client side application.

After completion of competition, the text typed by user is send back to the server, to calculate
As in any app, the user starts with home screen. From that point, the user can connect to server and join or create tourney. While connecting the application prompts from ip and port of server application. The user can likewise utilize automatic connect which doesn't request ip and port. In automatic connect the application itself searches for server in wireless network which takes much time as compared to manual connection. The client side application comprises of two screens, one the home screen and other the play screen. The center of screen comprises of notification panel, which consists of broadcasted messages from server.

![FIG 2. Home Screen.](image)

The play screen as shown in figure 3, consists of a question widget, which consists the question from server. A typing widget, where user can type the text. Button named “LEADERBOARD” can be clicked to view performance of other registered user. At top of screen time and score widget is also present.

**V. Working**

As soon as tourney is created, user can see the competition and its id in notification panel and utilize that id to join. User can likewise make a custom competition. This is trailed by setting up a confirmation key. This key ought to be utilized by different clients to enlist. After tourney advertisement, the enlisted clients are given with text content to type. Following 1 minute span the wrote the typed content is send back to server to validate. After validation the user can see his score at top of screen. The play screen at time of competition is shown in figure 4 and leaderboard of competition is shown in figure 5.

![Fig 3. Play Screen.](image)
VI. Result and Conclusion

With the help of raspberry pi, we were able to setup a server, capable of conducting and managing typing competition efficiently and effectively. We were even able to handle errors while connection and able to prevent common cheating methods like copying-pasting between users.

VI. Reference

