An Open Source *Raspberry Pi* based *Score Display System*

Gabriele Salvato
IPCF CNR Messina (Italy)

A brief introduction

All my three daughters play *Volley* in small teams of my city (Messina – Italy)  .

The economic support for all the teams' activities is modest and has to be carefully managed in order to maintain the team alive. The same condition apply to many other small sport teams in our and neighborhood cities.
In all the gymnasiums they play there are different systems for displaying score and (sometimes) other related information. Often the teams have only cardboard displays to show just the score

.

Since I had a powerful and inexpensive *Raspberry Pi*3  laying around my desk together with an old, but still working, 32" *Philips LCD TV*  , I asked myself if it could be possible to use them as a display system to show score information, as well as *slide-shows* and *videos* so as to encourage possible sponsors to give financial support to the teams.

I have then envisaged a (low cost) system that could be adapted to small gymnasiums (that will use a single *LCD* screen) as well as to bigger sport facilities in which many *LCD's* would show, in a coordinated way, the different information.

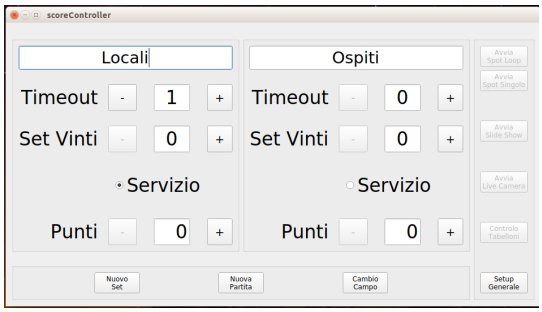
The system have to be simple to manage because it will be operated by people that are not necessarily computer *gurus* and, since I hate having messy cables laying around, I have wanted to create a system that could work wireless so that it can be easily moved without too much trouble.

I ended up in realizing a system in which an *Android* tablet (the "*Game Director*") controls, via *WiFi*, one or more "*Score Panels*", composed by a *Raspberry Pi* connected to a HDMI capable monitor (or TV). Each *Score Panel* is configurable in order to show the score information only or instead show score, slides and videos as dictated by the *Game Director*.
Each *Score Panels* can be equipped with a ***Raspberry Pi Camera module V2*** so as to show (under control of the *Game Director*) in real time what is happening on the game field, and the camera can be mounted on a ***Pan-Tilt module***  that, in turn, is controlled by the *Score Panels* via the tablet.

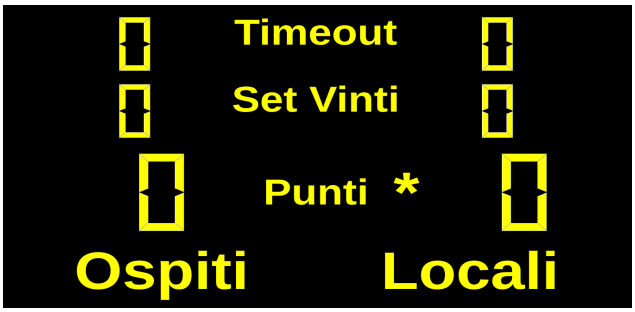
Hardware needed

From a ***hardware*** point of view, at minimum, you need a ***computer*** (in my case an ***Android tablet*** equipped with a touch screen and endowed of *WiFi* capabilities but you can use a laptop if you wish) that will stay on the scorekeeper table and one or more *Score Panels* composed of a ***Raspberry Pi*** connected by a short *HDMI* cable to a ***LCD TV*** (or ***monitor***).

One (and only one) *Raspberry Pi* has to be configured in such a way to realize a *Wireless Local Area Network* (*W-LAN*) acting as a sort of *"access point"*. The tablet plays the role of the *Game Director* and, together with all other Raspberry Pi, will connect to the *W-LAN* built up by the *Access Point* in order to send commands and exchange information with all the *Score Panels*.



The Game Director GUI (italian version)



The Volley Score Panel (italian version)

If *Volley* is the only sport you are interested in that's all the needed hardware. In the following we will see how to add timing capabilities to the system so to cover other sports.

The Software

In each *Raspberry Pi* there must be installed a (fresh) copy of the ***Raspbian*** operating system that can be downloaded from:

`https://www.raspberrypi.org/downloads/`

There are many tutorials that details very well how to download and install the required operating system into a micro *SD* and I will not go further on this argument apart that I strongly suggest to use a micro *SD* with a **storage capacity not less than *16GB*** and connect your *Raspberry* to the Internet, trough your router, **via an Ethernet cable**.

Once you have installed the operating system on your *Raspberry Pi*, let's be sure to have an updated version of the software by opening a terminal and issuing the following command:

```
$ sudo apt update && sudo apt upgrade -y
```

After the update has completed you need to modify some of the default setups of your *Raspberry*.

First of all, since all the information will be exchanged via *WiFi*, ***we need to change, for security reasons, the default password*** of the user ***`pi`***.

Click on the *Raspberry* icon  on your desktop and select:

Preferences → Raspberry Pi Configuration.

- From the *"System"* tab **change the password** and assign a different *"Hostname"* to each *Raspberry Pi score panel* (I use *panel-0*, *panel*-1 and so on...).
- Check the *"To Desktop"* radio button (to boot directly into the Desktop environment) and the *"Login as user 'pi'"* check-box (to be logged automatically without having to enter the password every time).
- Depending on your display (or TV), you may need to change also the *"Resolution"* and *"Overscan"* options.
- Under the tab *"Interfaces"* enable *"SSH"* and, if you intend to use the *Raspberry Camera Module* and the *Pan-Tilt* system, enable the *"Camera"* and *"Remote GPIO"* capabilities.
- Since we will use the *GPU* for showing slides and videos, under the *"Performance"* tab increase the *"GPU Memory"* to 128K or greater.

Close the configuration dialog. You will be asked to reboot: please do it now.

Then you need to clone the repository in which are stored the programs that build-up the *Score Panel System*. I suggest to clone the repository by using **`git`** so you can easily update the software as a new release became available. If you don't have it already installed, open a terminal window and issue the command:

```
$ sudo apt install git
```

Then tell **`git`** who you are by writing (this is not strictly needed):

```
$ git config -global user.email "your.email@address"
$ git config -global user.name "Your Name"
```

Obviously replace **"your.email@address"** with your actual email address and **"Your Name"** with your real name.

Now you are ready to clone the software repository. Open a terminal window and write:

```
$ git clone  https://github.com/salvato/ScorePanel_executables.git
```

A new folder, **ScorePanel_executables**, will be created that, at present, contains three more folders:

- **Android**
  containing **ScoreController.apk**, the executable for the "*Game Director*" part of the system.

- **Raspberry**
  which contains two executables: the **panelChooser** program, responsible of realize the *Score Panels* for a few sports, and the **SlideShow** program, the one that take care of showing, on request, the slides. The system make also use of the commonly installed program **omxplayer** to show short movies. It should be already existing in your *Raspbian* installation.

- **Ubuntu**
  which contains the version of the programs that you can use on *Ubuntu* (but we will not cover the installation on *Ubuntu* in this document).

Copy the two programs contained in the **ScorePanel_executables/Raspberry** folder into the ***pi*** home directory.

The **panelChooser** program depends on some ***Qt5*** libraries that can be already installed and on some others that can be missing. To check which libraries are missing you may issue the following command in a terminal window:

```
$ ldd ~/panelChooser
```

In my *Raspbian* version (*stretch*) two of the needed libraries are missing:

```
libQt5WebSockets.so.5 => not found
libQt5SerialPort.so.5 => not found
```

You can install such libraries by issuing the following command:

```
$ sudo apt install libqt5websockets5 libqt5serialport5
```

You can check if everything is working by launching the **panelChooser** program. Open a terminal and issue:

```
$ ~/panelChooser
```

A full screen black window should appears and a message should tell you that the program is "*Waiting for Server Connection*". If your *Raspberry* is not connected to a network the message will instead tell you that the program is "*Waiting for a Network Connection*". In any case click *Esc* on your keyboard to close the black window.

Now we are sure that the program is correctly installed and all the needed libraries are present.

Disable screen blanking

We have to **disable Screen Blanking** to prevent the disappearing of the score during the match. You need to edit a configuration file:

```
$ sudo nano /etc/lightdm/lightdm.conf
```

Search the line looking:

```
#xserver-command=X
```

and change it in:

```
# To disable screen blanking
xserver-command=X -s 0 -dpms
```

Enable servo control of the Camera (optional)

If you want to control the *pan* and the *tilt* of the *Raspberry* camera with two servos, you need to start the *gpio* daemon at boot time. You have to edit an other configuration file:

```
$ sudo nano /etc/rc.local
```

Insert:

```
/usr/bin/pigpiod &
```

just before the line containing:

```
exit(0)
```

If you are using the *WiFi* for the network connections you have to be sure that the power management does not place the *WiFi* interface in power save mode. According to a post in the *Raspberry* forum, for the built-in *WiFi* of the *Raspberry Pi3* the power management is always off even if the command:

```
$ iw dev wlan0 get power_save
```

will report that it is on, so we don't need to worry about.
https://www.raspberrypi.org/forums/viewtopic.php?t=194619#p1239465

Automatic start of the program at boot

Now it is time to tell your *Raspberry* to automatically start the program at boot so you don't need to start it any time you switch on the system.

Issue the following command on a terminal window:

```
$ nano ~/.config/lxsession/LXDE-pi/autostart
```

and, since we don’t want that a screen-saver can obscure the panel, comment the line

```
@xscreensaver -no-splash
```

changing it in:

```
#@xscreensaver -no-splash
```

Then, at the end of the file, insert a line to start the *score panel* program at every reboot:

```
/home/pi/panelChooser &
```

Save and reboot to check that everything has gone well.

To exit the program remember to hit *Esc* on your keyboard.

If you have a *WiFi* network up and running and don’t want to use a dedicated *WiFi* network for your system, you are almost done with your *Raspberry*, but I strongly suggest that you create your own *WiFi* network to separate the S*core Panel System* from all the other network traffic.

Configuring your Raspberry to act as an "Access Point"

To configure your *Raspberry* as an access point we will follow the instructions at:

https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

stopping at the paragraph "***ADD ROUTING AND MASQUERADE***" since we don't want to route any message out on Internet: all the network traffic should remain inside the local network.

To facilitate the configuration I transcript here the instructions reported on the previous link.

Let's start with installing the needed programs:

```
$ sudo apt install hostapd dnsmasq
```

We have to turn the new software off until correctly configured:

```
$ sudo systemctl stop dnsmasq
$ sudo systemctl stop hostapd
```

Then we have to configure a static IP for the *Access Point* editing the *dhcpcd* configuration file:

```
$ sudo nano /etc/dhcpcd.conf
```

At the end of the file insert the following lines (you can freely change the address 192.168.42.1 with any other in the private IP address space, see **https://en.wikipedia.org/wiki/Private_network**):

```
interface wlan0
static ip_address=192.168.42.1/24
static routers=192.168.42.1
```

Now we can restart the *dhcpcd* daemon to set up the new **wlan0** configuration:

```
$ sudo service dhcpcd restart
```

Configuring the DHCP server (dnsmasq)

The *DHCP* service is provided by *dnsmasq*. Rename its existing configuration file, and edit a new, empty, one:

```
$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
$ sudo nano /etc/dnsmasq.conf
```

Copy the following information into the */etc/dnsmasq.conf* file:

```
# Use the require wireless interface
# we are going to provide IP addresses between
# 192.168.42.2 and 192.168.42.20,
# with a lease time of 24 hours.
interface=wlan0
  dhcp-range=192.168.42.2,192.168.42.20,255.255.255.0,24h
```

Configuring the access point host software (hostapd):

Edit the configuration file (*/etc/hostapd/hostapd.conf*):

```
$ sudo nano /etc/hostapd/hostapd.conf
```

and insert the following text into the empty file:

```
interface=wlan0
driver=nl80211
ssid=myOwnScorePanelsSSID
country_code=IT
hw_mode=g
channel=6
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=myVerySecretPassword
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Please change the **country_code** with the one that apply to you and choose a **channel** that is free and possibly far from other occupied channels. To check which are the channels already occupied you can issue the following command:

```
$ sudo iwlist scan | grep Channel
```

We are ready to tell the *hostapd* program where to read its configuration by editing the **/etc/default/hostapd** file:

```
$ sudo nano /etc/default/hostapd
```

Find the line with **#DAEMON_CONF**, and replace it with this:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

We are now ready to start all the services:

```
$ sudo systemctl start hostapd
$ sudo systemctl start dnsmasq
```

If you plan to have more than one *score panel* in your system, only one have to be configured to act as an access point, all the other must just be connected to the network created by that one.

Install a Raspberry Pi Camera Module v2 (optional)

As said before, you can install on each *Score Panel*, a *Pi Camera*  ,

connected with its ribbon cable to the *CSI* port of your *Raspberry*  , so as to

show, on request, live images of the game field. Each *Camera*, in turn, can be mounted on top of a *pan-tilt* module like the one previously shown, equipped with two *SG90 micro servos*

 .

The servos must be connected to the *gpio* pins as follows (the pin numbers refer to the pin number of the 40 pin *gpio* connector **`https://pinout.xyz`**):

Pan Servo	GPIO pin
GND (Brown wire)	9
VCC (Red wire)	4
PWM (orange wire)	8

Tilt Servo	GPIO pin
GND (Brown wire)	34
VCC (Red wire)	2
PWM (orange wire)	37

An important consideration regards the way on how to switch off your *score panels*. When your panels will be connected to the *game director*, switching off the *game director* will offer the choice to switch off also the *score panels*. ***This is the correct procedure you have to follow to minimize the possibility of damaging the SD card inside your Raspberry.*** I have implemented this procedure because it is not convenient to switch off separately each *score panel* possibly placed in difficult to reach places. All that will remain to do is to switch off the power line after a couple of minutes.

We have done with the *score panel* part of the system. You can disconnect the mouse, keyboard and Ethernet cable (since you don't need it anymore) and leave your *Raspberry* connected to the *HDMI* display and power supply.

Well … **it is time now to setup the G*ame Director* part.**

Installing the Android App

Connect your *Android* tablet (only ARMv7 based ones are currently supported) to your *Raspberry* and copy the file:

`~/ScorePanel_executables/Android/ScoreController.apk`

into the *SD card* of your tablet. Create also two new folders into the same *SD card* named respectively: *slides* and *spots*.

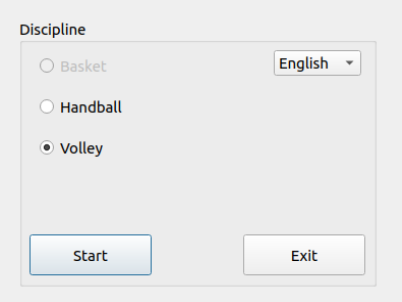
Copy all the slides that you want to show into the *slides* directory (only *jpeg* and *png* formats are currently recognized) and all the movies you want to play into the *spots* directory (only *mp4* files are recognized at present).

Disconnect your tablet from the *Raspberry* and open the file manager of your tablet. Navigate to where the **`ScoreController.apk`** file has been copied and click on its icon to install it.

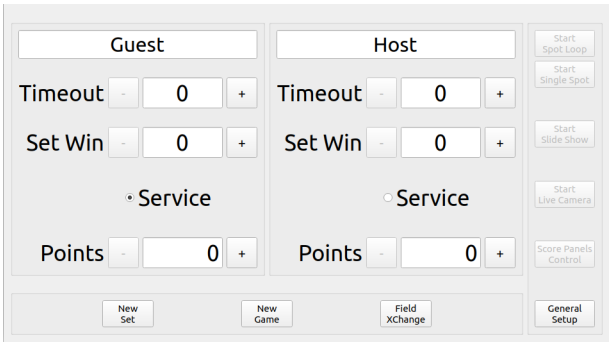
It is possible that the security options on your tablet will prevent to install the program since it comes from an unknown source. You should allow to install also from unknown sources just for the time to install the program and block again once the program has been installed.

A new applet icon should appear into your tablet. Be sure to connect the tablet to the same network of the *Score Panels* and then click on the program icon: **your, shiny new, S*core Panel System* should start !**

A dialog like the one depicted should appears allowing to choose the language (at present only Italiano and - a poor translation of – English) and the sport discipline you are interested in. At present only *Volley* can be run without further hardware. In a follow-up we will consider disciplines that requires timing information.



Select the discipline and click *Start: th*e dialog will close and a new window will be shown:



The very first time the *App* will run it will ask to locate the two directories containing the *slides* and the *spots*. Please select the right directories (the ones you have created before) and go on.

All the controls in the window should be really intuitive so I'll not explain their meaning. Please feel free to experiment and enjoy your new system.

The *Game Director* as well as all the *Score Panels* doesn't require further configuration. The *Game Director* should automatically connect to all the *Score Panels* associated to the same network and should send all the needed commands to all the *Score Panels*. Please allow some 10-15 seconds to the G*ame Director* to be discovered by the various *Score Panels* connected to the network.

As soon as each *Score Panel* has been connected, the *Game Director* will take care of updating (only if needed) the *Score Panels* local content of the *slides* and *spots* directories. In fact each *Score Panels* will maintain a local copy of the slides and movies in order to speed-up their appearing on the screen and reduce the network traffic after the startup phase. In this way you will have also a single source of such files to update when needed: the two directories on the SD card of your tablet. **All this housekeeping is done in the background without any user interaction.**

The Software Sources

As said in the title all the sources are freely available and I hope to receive contributions from the many brave developers around for improving the system.

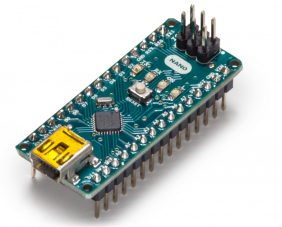
The programs are written in ***C++*** using the impressive framework ***Qt5*** (*https://www.qt.io/*). You can clone the following repositories:

```
https://github.com/salvato/scoreController.git
https://github.com/salvato/ScorePanel.git
https://github.com/salvato/SlideShow.git
```

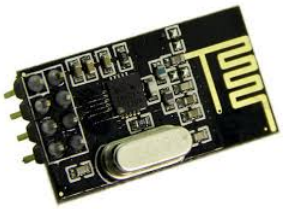
containing the sources, respectively, for the *Game Director* part, for the *Score Panels* and for the *SlideShow* program.

Add timing capabilities to the system (To be completed !!!!)

Many sports require timing capabilities that need much more prompt response than a multi task operating system (like the *Raspbian*) can offer. To overcome this problem I have realized a subsystem composed of two *Arduino* (I have used two *Arduino Nano*



because of their small sizes but other models can be used too) interconnected via two radio modules, *nRF24l01+*



or *nRF24l01+ PA +LNA*

.

The first *Arduino*, that will stay on the timekeepers table, is equipped with a number of push-buttons and act as a remote for the second *Arduino* that will be connected via an *USB* cable to the *Raspberry* score panel.

On the first *Arduino* must be loaded the **`StopWatchSender.ino`** sketch and on the second one you must load the **`stopWatch.ino`** sketch. Both sketches can be downloaded from ***github***:

`https://github.com/salvato/StopWatchSender.git`
`https://github.com/salvato/stopWatch.git`

The Stop Watch Sender (the remote)

Let's start to describe the *StopWatchSender* hardware connections. The communication between the Arduino and the *nRF24l01+* module is done using the Serial Peripheral Interface (*SPI*) presents on both the devices. The Arduino plays the role of *SPI Master* and the *nRF24l01+* the one of *SPI Slave*. The following table specify the connections that need to be made between the two devices:

***NRF24L01+* Pin**	***Arduino* Pin**
1 (GND)	GND
2 (3.3V)	3.3V
3 (CE)	D9
4 (CSN)	D10
5 (SCK)	D13
6 (MOSI)	D11 (MOSI)
7 (MISO)	D12 (MISO)
8 (IRQ)	D2

Connect five ***normally open*** push-buttons from the *Arduino* pins from *D3* to *D7* an *GND* and a buzzer from pin *A0* and *GND.*

With systemd there is a way to automatically restart services, such as hostapd.service is. In particular, this is an advantage of systemd over the traditional init V stuff, or some rc.local sleep-n-restart stuff. I successfully tested my approach on a set of Pi 3Bs with an additional Edimax WLAN dongle for AP.

So, edit `/etc/systemd/system/hostapd.service` and add to the section `[service]` these lines:

```
RestartSec=5
Restart=on-failure
```

Fortunately, systemd has a built-in check that will do only a limited number of restart attempts in case of failures. Normally, these settings should suffice.

More details can be found in systemd's documentation here: service unit configuration

Installing Qt5 on your Raspberry

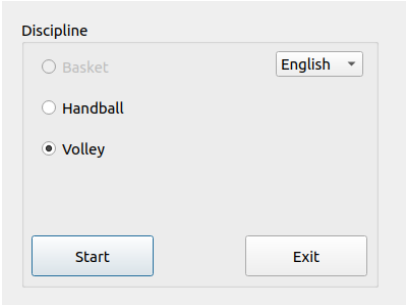
You can **install *Qt5*** and the required libraries in the Raspberry Pi by opening a terminal and writing:

```
$ sudo apt install qt5-default qtcreator
$ sudo apt install libqt5serialport5-dev libqt5websockets5-dev
$ sudo apt install libqt5multimedia5
```

If everything has gone well you should be able to see the freshly installed *Qt Creator* developing environment  by clicking on the raspberry icon  of your desktop under the *"Programming"* menu.

Score Controller walkthrough

At the program start a dialog allowing to select the sport discipline and the desired language is shown:



The selected *Score Controller* software start creating, for each network interface available on the device, an *UDP* socket listening for messages arriving at the *multicast* address 224.0.0.1 on port 45453.

I call this procedure a *discovery service* in that each client can *discover* the presence of a running *Score Controller,* independently from its network address. The clients send a short message to this address and listen for an answer within a given time. If there is a *Score* Controller active it reply with a message containing its network address and the *sport discipline* selected. This procedure allows to minimize the configuration of the computer in which the *Score Controller* software runs: in practice it only need to configure the connection with the *WiFi access point* common to all the computers in the system.

Then the *Score Controller* software create a *QWebSocketServer* for handling the incoming connections from the various clients. All the connections from the *Score Panels* will be served by this *QWebSocketServer*.

In order to maintain a centralized control over the *spots* and *slides* to be used by the *Score Panels*, the corresponding files are placed in two folders (usually named *slides* and *spots*) into a *SD* card (or *HD*) of the computer running the *Score Controller* software. To minimize the network traffic the *slides* and *spots* files are also stored locally on each *Score Panel*. Only the files that need to be changed or are not already stored locally are sent to each *Score Panel.* To allow the local storage update of each client the *Score Controller* computer creates two more *QWebSocketServer*, one for

the *slides* and the other for the *spots*. Each update process will be executed in a different thread so as to handle all the updates it in parallel.

Notes

The voltage regulator on Arduino Pro Mini 3.3V 8MHz is capable of supply 150mA Max. The ATmega328P data sheet say that @8MHz @5V requires 9.0mA (with Minimizing Power Consumption enabled).

The nRF24L01+ PA-LNA (RFX2401C) requires 90.0mA @3.3V, Pout=+20dBm and maximum data rate (2MHz).