AUES
Since 1975

**Non-commercial joint-stock company**

**ALMATY UNIVERSITY POWER AND COMMUNICATIONS**

Department of "Electronics and Robotics"

# COMPUTER TECHNOLOGIES IN INSTRUMENT MAKING

Guidelines for the implementation of laboratory works
for students majoring 5B071600 – Instrument making

Almaty 2019

COMPONENTS: Baikenov B.S., Ayazbay A.E. Computer technology in instrument making. Guidelines for the implementation of laboratory work for students of the specialty 5B071600 - "Instrument Engineering". - Almaty: AUPET, 2019. - 38 p.

The guidelines are devoted to the study of the Raspberry PI-3 microcomputer, which is the main element for the creation of digital remote control and monitoring systems over IP networks.

In detail, step by step, we consider the selection and installation of the Raspbian operating system, packages and applications, an SSH network protocol, a web server with PHP, HTML, a MySQL server, setting up a remote VNC control, creating a web server with temperature and humidity data.

In addition, there are two technologies for creating a remote control system for IP networks of a variable current drive and temperature control using a web page and a touch screen smartphone.

Methodical instructions are drawn up in order to consolidate the lecture material and are intended to students of the specialty 5B071600 - “Instrument Engineering”.

Il - 23, bibliogr -7

Reviewer: Senior Lecturer  Kim E.S.

Reprinted according to the plan of publishing a non-commercial joint-stock company “Almaty University of Power and Communications” for 2019.

# Introduction

The Raspberry Pi is a miniature credit card-sized computer with a clock frequency of 900 MHz and a 1 GB RAM module. Given the satisfactory power, low power consumption and low cost, Raspberry Pi can be used to create a personal mini-server.

One of the main features of the Raspberry Pi is the availability of GPIO hardware input / output ports (General Purpose Input / Output, a general-purpose input / output interface), which opens up prospects for its use in remote control and monitoring systems, robotic projects and smart home devices ".

The laboratory workshop on the subject “Computer Technologies in Instrument Engineering” consists of 10 jobs and is conditionally divided into two parts - installation of the operating system, applications, servers and the microcomputer programming itself to perform object management tasks over IP networks.

The implementation of the first part of the guidelines takes time and is very laborious, but the process itself follows the standard algorithm.

The second part is devoted directly to microcomputer programming in languages such as PHP, Python, HTML, MySQL, and requires certain skills and competencies.

# 1 Laboratory work №1. Choice and installation of the Raspbian operating system

Purpose: acquaintance with the technical characteristics of the Raspberry PI microcomputer, the acquisition of practical skills in installing the OS and setting parameters.

## 1.1 Summary of the theory

In order to run the Raspberry Pi, you need to install an operating system on it. Three official Linux distributions are available:

1) Pidora - based on Fedora.
2) Archlinux - the installation of this distribution is almost manually.
3) Raspbian - based on Debian.

In addition to these three operating systems, a lot of others are ported to the Raspberry Pi. Since the operating system is installed on the SD card, in order to start another system, it is enough to insert a card with this system into the device.

The easiest way to install the distribution on the Raspberry Pi is the NOOBS tool from the creators of the Raspberry Pi, which allows you to select one operating system from the following for the first boot:

1) ArchLinux.
2) OpenElec.
3) Pidora.
4) RaspBMC.

5) Raspbian.
6) RiscOs.
To install the distribution on Raspberry Pi using NOOBS, you need a microSD card with a capacity of 4 GB or more.

## 1.2 Order of performance

1.2.1 Open the SD Card Formatter application and format the memory card to prepare for the OS installation.

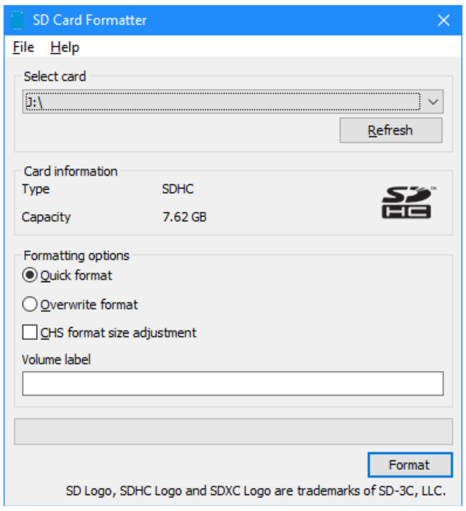

SD Card Formatter
File Help
Select card
J:\
Refresh
Card information
Type SDHC
Capacity 7.62 GB
Formatting options
(●) Quick format
( ) Overwrite format
[ ] CHS format size adjustment
Volume label
Format
SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC.

Figure 1.1 - SD Card Formatter window

1.2.2 It is required to download the OS image from the official site (there are many of them, but the most relevant and functional is Raspbian, which is taken on the basis of Linux).

1.2.3 After successfully downloading the OS, you need to load the OS image into a clean NOOBS memory card, for this we use adapters for the card reader and insert it into the slot, then we drop the image into the memory card.

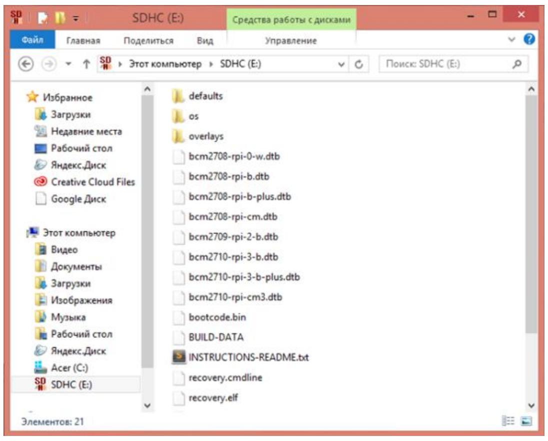

SDHC (E:)
Этот компьютер › SDHC (E:)
defaults
os
overlays
bcm2708-rpi-0-w.dtb
bcm2708-rpi-b.dtb
bcm2708-rpi-b-plus.dtb
bcm2708-rpi-cm.dtb
bcm2709-rpi-2-b.dtb
bcm2710-rpi-3-b.dtb
bcm2710-rpi-3-b-plus.dtb
bcm2710-rpi-cm3.dtb
bootcode.bin
BUILD-DATA
INSTRUCTIONS-README.txt
recovery.cmdline
recovery.elf
Элементов: 21

Figure 1.2 - The contents of the memory card, ready for installation in the board

1.2.4 Next, install the memory card on the Raspberry PI platform, then prepare the device for displaying information and suitable wires.

1.2.5 On the monitor we see the available image of the Raspberry PI logo with multi-colored flowers - this proves the successful loading of the OS. Next, slowly, click on the desired parameters, choosing the appropriate settings for the microcomputer.

1.2.6 We are waiting for the installation. Acquaintance with a desktop and surrounding elements.

### 1.3 Report content

1.3.1 Goal of the work.

1.3.2 Description of the OS installation with the bringing of screenshots and photos of the microcomputer.

1.3.3 Conclusions.

### 1.4 Test questions

1. What is the Raspberry PI?
2. What are the operating systems and describe their differences between the Raspbian?
3. How can I improve the screen output device?
4. Tell us about your model Raspberry PI.
5. Is it possible to manage the desktop from the command line, having access to all commands?

# 2 Laboratory work №2. Installing packages and applications

Purpose: familiarization with the command line and application management, installation of packages, work with screenshots.

## 2.1 Summary of the theory

The installed Raspbian OS has a main console - LXTerminal. It is more convenient to use the pre-installed program LXTerminal, which emulates the main console as a window in the graphical interface. When we open this program, we will see the line:

pi @ raspberrypi ~ $

where pi is the user name in the OS;

raspberrypi - hostname;

~ - current directory, ~ replaces / home / pi.

When we see such a line, we can write our team. In particular, here you can do actions with files, folders, archives (but these actions can be done in graphical mode, which is much more convenient). Some actions require administrative rights, they can be used simply by adding sudo at the beginning of the command.

The easiest way to install software is to use PiStore. This is a convenient and understandable way with a graphical interface, but the PiStore library does not contain as many programs as we would like. An alternative way to search, download and install programs, packages on the Raspberry Pi is apt-get console utility. Apt-get stores data on various console and graphics programs and utilities on the Raspberry Pi.

## 2.2 Order of performance

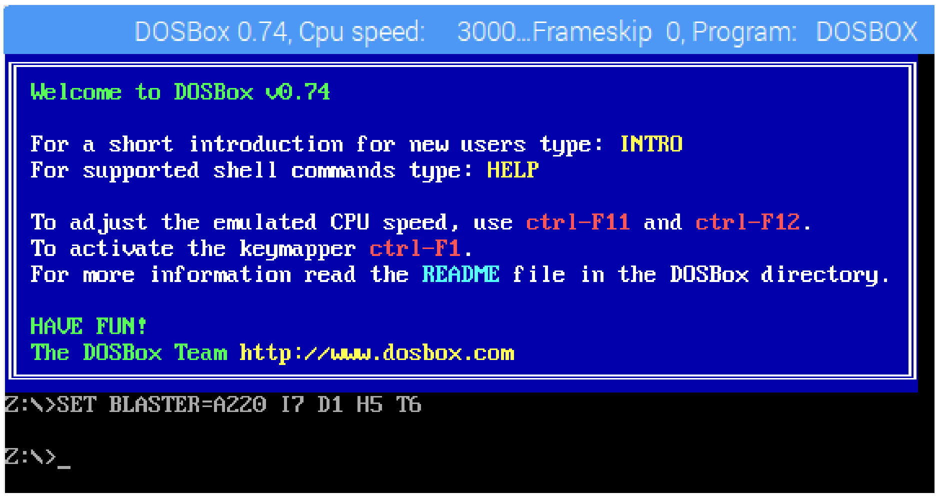
2.2.1 Before working with this utility, you need to update its program database with the command:

sudo apt-get update

2.2.2 After the upgrade, you can download the program. To search the utility database, you need to do the following query:

apt-cache search <request>

For example: sudo apt-get install dosbox. Installation will need to be confirmed with Enter.



```
DOSBox 0.74, Cpu speed:    3000...Frameskip  0, Program:  DOSBOX

Welcome to DOSBox v0.74

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>_
```

Figure 2.1 - DosBOX Interface

2.2.3 Periodically with this utility it is desirable to perform the following actions:

sudo apt-get update (update program database)
sudo apt-get upgrade (upgrade installed programs)
sudo apt-get autoremove (remove remaining library programs)

To uninstall the program, use the command (the deletion also needs to be confirmed):

sudo apt-get remove <name>

2.2.4 Unfortunately, Raspbian OS does not have any pre-installed tools for creating screenshots - screenshots. You can search for tools for creating screenshots in apt-get using the query:

```
apt-cache search screen capture
```

A more convenient application is the scrot data package, installation:

```
sudo apt-get install scrot
```

To create a screenshot, you must enter the scrot command. You can specify a delay in the execution of a command — this is necessary so that you can minimize or close the console — otherwise it will fall into the screenshot. To delay 5 seconds, enter:

```
scrot –d5
```

## 2.3 Report content

2.3.1 Goal of the work.
2.3.2 Description of the installed programs (optional), screenshots of the work done in the utilities.
2.3.3 Conclusions.

## 2.4 Test questions

1. What applications can I install besides dosbox? Tell us about current programs.
2. What will the command for installing Synaptic utility packages look like?
3. What command should I use to find the right package in the archives?
4. What team need to be added before the request to confirm admin rights?

# 3 Laboratory work №3. Installing the SSH network protocol

Objective: To familiarize, configure, and enable the SSH protocol.

## 3.1 Summary of the theory

First of all, you should consider what the Secure Shell network protocol is (it is called SSH for short).

SSH is a protocol that provides a connection over a network (local or Internet) between two remote devices. On it you can transfer almost any information:

- files;
- video and audio in the stream;
- teams.

Advantages: first, by it can compress data on the fly. Secondly, it creates an encrypted tunnel.

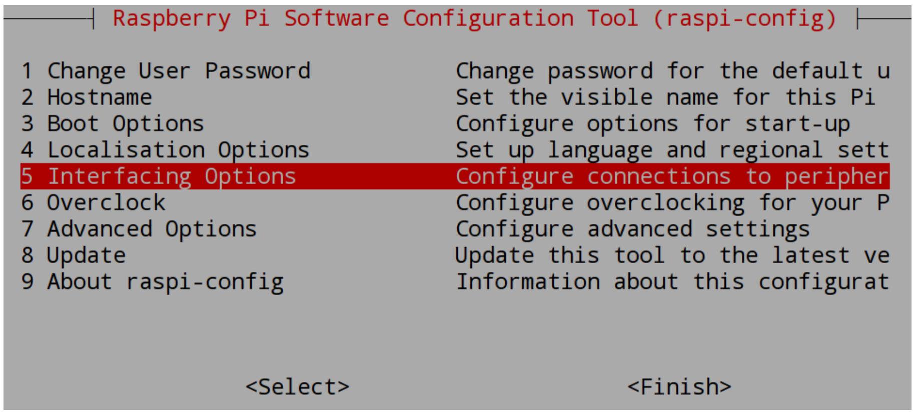
Figure 3.1 - Understanding SSH

When connecting to the Raspberry Pi via SSH on a single-board device, a terminal is launched, which is operated on another computer. It is possible to connect via Secure Shell, being just 1 meter from the "Raspberry", as well as on the other side of the globe.

## 3.2 Order of performance

3.2.1 Open the console and prescribe the command “raspi-config” and of course, for those who do not have rights, add the command “sudo”.

3.2.2 A moment after clicking Enter, the interface of the standard Raspberry configuration utility will appear. It is necessary to find the point Interfacing Option (it is the fifth) in it.



```
────┤ Raspberry Pi Software Configuration Tool (raspi-config) ├────

 1 Change User Password           Change password for the default u
 2 Hostname                       Set the visible name for this Pi
 3 Boot Options                   Configure options for start-up
 4 Localisation Options           Set up language and regional sett
 5 Interfacing Options            Configure connections to peripher
 6 Overclock                      Configure overclocking for your P
 7 Advanced Options               Configure advanced settings
 8 Update                         Update this tool to the latest ve
 9 About raspi-config             Information about this configurat



                  <Select>                      <Finish>
```

Figure 3.2 - Configuration Interfaces

3.2.3 Now you need to enable SSH on Raspberry Pi. To do this, select the SSH item (it goes second), press Enter and select Enable, and then press "Enter" again. Then it remains to return to the main screen and select Finish.

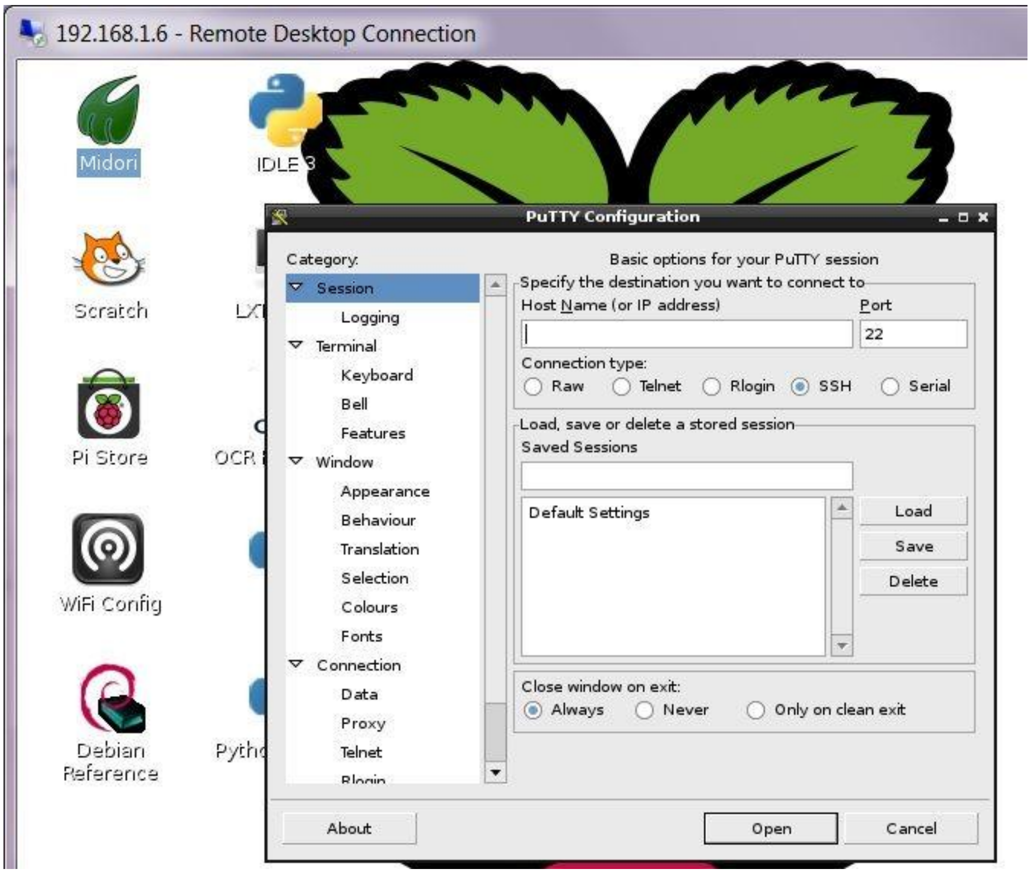
Note: After this, all changes will be applied. Now you can restart Malin, although even without this, SSH should already work.

3.2.4 Next, we check the operation of the protocol and try to create a connection. To do this, we enter the installed program “PuTTy” from another main PC.

3.2.5 In the "PuTTy Settings" window on the "Session" tab (it is active after opening the window) you need to do the following: enter the address "Raspberry" in the "Host Name" field, enter the port number (for SSH most often the 22nd is used, and it will be exactly like this if the user himself did not change anything), SSH should be active in the list of switches "Connection type".

3.2.6 You will need to click the "Connect" button. If the address, port and type of connection were specified correctly, and the network on two devices is working properly, then after a moment, the Windows command line will be displayed, and through it you can interact with Raspberry and all commands that are not supported by CMD will be available.

Note: When the console appears, you will need to log in to the system. To do this, firstly, specify the login (username on the "Malin"), and, secondly, the password that is assigned to the corresponding user. If they are correct, then full access to the computer will be given.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: ( ) Raw ( ) Telnet ( ) Rlogin [✓ SSH] ( ) Serial

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: [✓ Always] ( ) Never ( ) Only on clean exit

Figure 3.3 - Configuring PuTTy

**3.3 Report content**

3.3.1 Goal of the work.

3.3.2 Report on the done commands from the PC with screenshots.
3.3.3 Conclusions.

### 3.4 Test questions

1. What is a Seruce Shell?
2. Is it possible to work with graphic applications via SSH?
3. If not, how can you work remotely with graphics?
4. What are the disadvantages when connecting to SSH?

## 4 Laboratory work №4. Configure VNC Remote Control

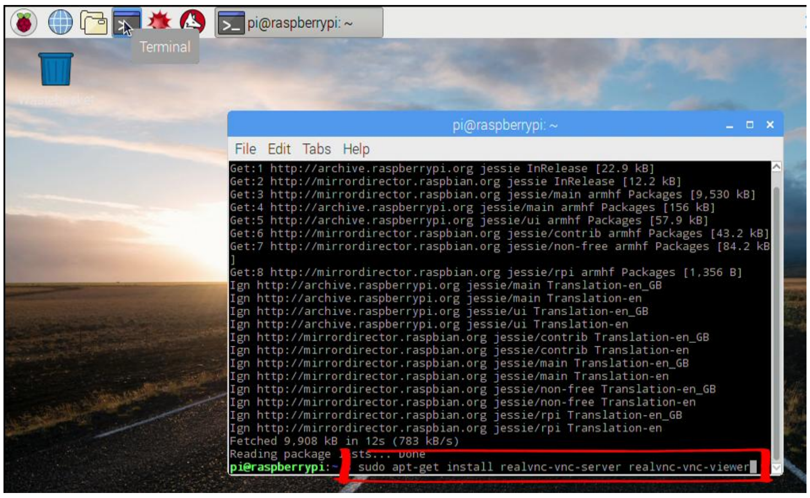
Purpose: to make the connection not only via the network protocol, but also control using remote access VNC.

### 4.1 Summary of the theory

One of the most common ways to remotely control the Raspberry Pi is to manage the board using the SSH protocol using Putty. There are other methods of control, not only using the console, but also graphically, controlling from any computer. It can be controlled remotely in full-fledged graphical mode by connecting to the raspberry using VNC (Virtual Network Computing).

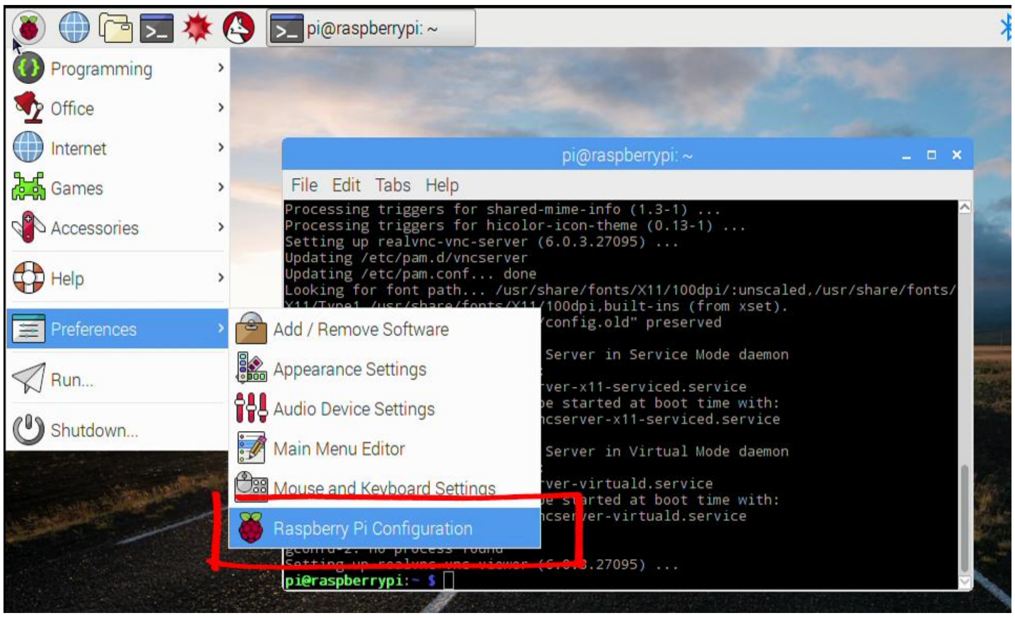
### 4.2 Order of performance

4.2.1 Open the terminal and update the list of packages in the system:
sudo apt-get update



```
Get:1 http://archive.raspberrypi.org jessie InRelease [22.9 kB]
Get:2 http://mirrordirector.raspbian.org jessie InRelease [12.2 kB]
Get:3 http://mirrordirector.raspbian.org jessie/main armhf Packages [9,530 kB]
Get:4 http://archive.raspberrypi.org jessie/main armhf Packages [156 kB]
Get:5 http://archive.raspberrypi.org jessie/ui armhf Packages [57.9 kB]
Get:6 http://mirrordirector.raspbian.org jessie/contrib armhf Packages [43.2 kB]
Get:7 http://mirrordirector.raspbian.org jessie/non-free armhf Packages [84.2 kB
]
Get:8 http://mirrordirector.raspbian.org jessie/rpi armhf Packages [1,356 B]
Ign http://archive.raspberrypi.org jessie/main Translation-en_GB
Ign http://archive.raspberrypi.org jessie/main Translation-en
Ign http://archive.raspberrypi.org jessie/ui Translation-en_GB
Ign http://archive.raspberrypi.org jessie/ui Translation-en
Ign http://mirrordirector.raspbian.org jessie/contrib Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/contrib Translation-en
Ign http://mirrordirector.raspbian.org jessie/main Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/main Translation-en
Ign http://mirrordirector.raspbian.org jessie/non-free Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/non-free Translation-en
Ign http://mirrordirector.raspbian.org jessie/rpi Translation-en_GB
Ign http://mirrordirector.raspbian.org jessie/rpi Translation-en
Fetched 9,908 kB in 12s (783 kB/s)
Reading package lists... Done
pi@raspberrypi:~ sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer
```

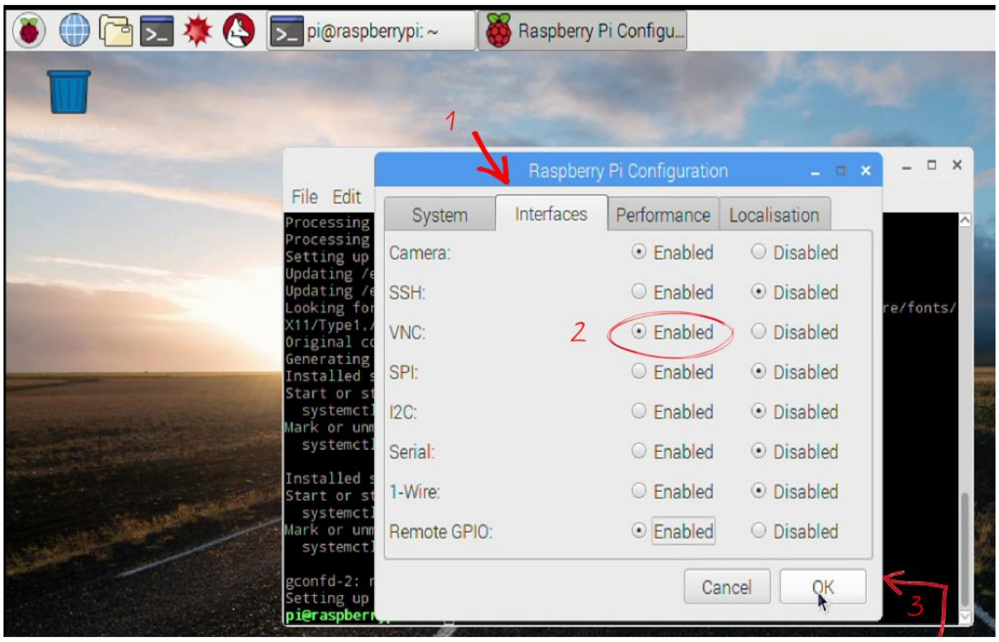
Figure 4.1 - VNC Installation
4.2.2 Install the VNC server on the Raspberry PI:
sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer
4.2.3 Go to the menu item: Menu> Preferences> Raspberry PI configuration.



```
Processing triggers for shared-mime-info (1.3-1) ...
Processing triggers for hicolor-icon-theme (0.13-1) ...
Setting up realvnc-vnc-server (6.0.3.27095) ...
Updating /etc/pam.d/vncserver
Updating /etc/pam.conf... done
Looking for font path... /usr/share/fonts/X11/100dpi/:unscaled,/usr/share/fonts/
                                100dpi,built-ins (from xset).
                                config.old" preserved

                                 Server in Service Mode daemon

                                ver-x11-serviced.service
                                e started at boot time with:
                                cserver-x11-serviced.service

                                 Server in Virtual Mode daemon

                                ver-virtuald.service
                                e started at boot time with:
                                cserver-virtuald.service

                                .27095) ...
pi@raspberrypi:~ $
```

Figure 4.2 - Menu item

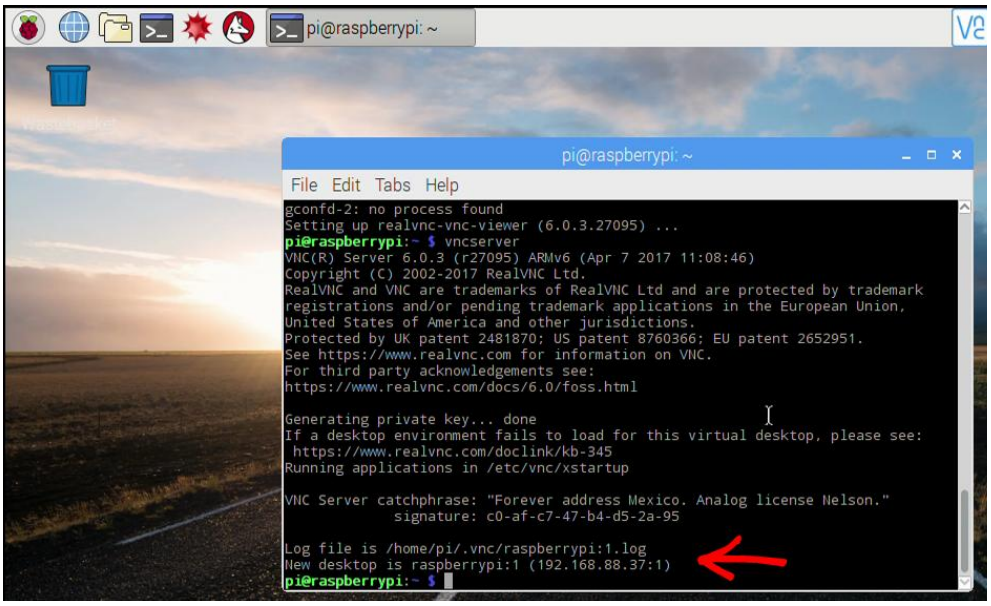
4.2.4 The Raspberry Pi microcomputer settings window will open. Go to the Interfaces tab, find the VNC menu item, select the Enabled state and click the OK button.



| | | |
|---|---|---|
| Camera: | [✓ Enabled] | Disabled |
| SSH: | Enabled | [✓ Disabled] |
| VNC: | [✓ Enabled] | Disabled |
| SPI: | Enabled | [✓ Disabled] |
| I2C: | Enabled | [✓ Disabled] |
| Serial: | Enabled | [✓ Disabled] |
| 1-Wire: | Enabled | [✓ Disabled] |
| Remote GPIO: | [✓ Enabled] | Disabled |

Figure 4.3 - Enable VNC

4.2.5 Return to the “terminal” window and start the VNC server:
vncserver
4.2.6 We learn about the successful launch of the server with the IP address and port number.

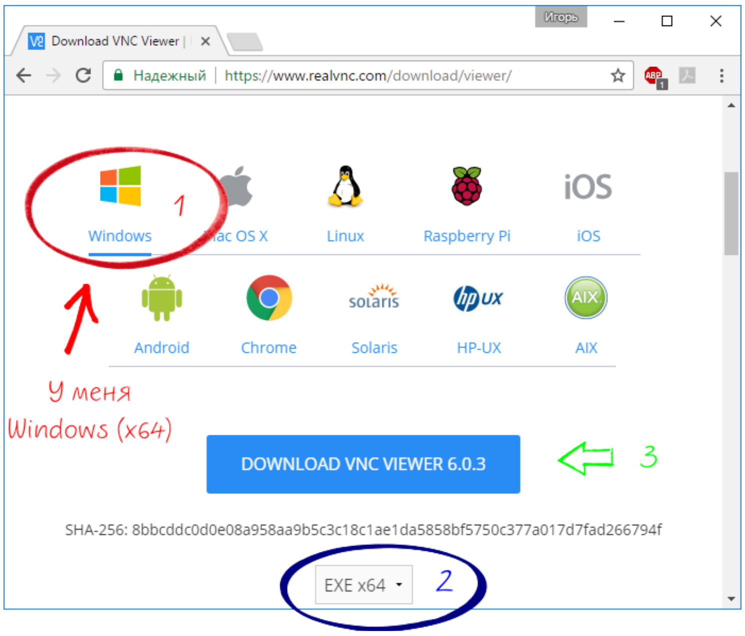


```
gconfd-2: no process found
Setting up realvnc-vnc-viewer (6.0.3.27095) ...
pi@raspberrypi:~ $ vncserver
VNC(R) Server 6.0.3 (r27095) ARMv6 (Apr 7 2017 11:08:46)
Copyright (C) 2002-2017 RealVNC Ltd.
RealVNC and VNC are trademarks of RealVNC Ltd and are protected by trademark
registrations and/or pending trademark applications in the European Union,
United States of America and other jurisdictions.
Protected by UK patent 2481870; US patent 8760366; EU patent 2652951.
See https://www.realvnc.com for information on VNC.
For third party acknowledgements see:
https://www.realvnc.com/docs/6.0/foss.html

Generating private key... done
If a desktop environment fails to load for this virtual desktop, please see:
 https://www.realvnc.com/doclink/kb-345
Running applications in /etc/vnc/xstartup

VNC Server catchphrase: "Forever address Mexico. Analog license Nelson."
             signature: c0-af-c7-47-b4-d5-2a-95

Log file is /home/pi/.vnc/raspberrypi:1.log
New desktop is raspberrypi:1 (192.168.88.37:1)
pi@raspberrypi:~ $
```

Figure 4.4 - IP Address and Port Number

Note: In our case, the IP address is 192.168.88.37, the port number is 1. This means everything turned out and you can proceed to the client settings.

4.2.7 To connect to the Raspberry Pi remote desktop, you need to install and configure the VNC client on the PC. To do this, go to the official page "REALVNC". Select the version of the environment depending on the operating system.



Download VNC Viewer |
Надежный | https://www.realvnc.com/download/viewer/
Windows
Mac OS X
Linux
Raspberry Pi
iOS
Android
Chrome
Solaris
HP-UX
AIX
1
2
3
У меня
Windows (x64)
DOWNLOAD VNC VIEWER 6.0.3
SHA-256: 8bbcddc0d0e08a958aa9b5c3c18c1ae1da5858bf5750c377a017d7fad266794f
EXE x64

Figure 4.5 - Site Display

4.2.8 Type in the address bar the IP address that was issued when creating the VNC server and press Enter.

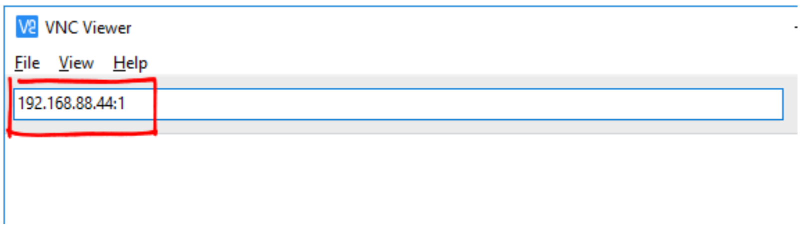


Figure 4.6 - Entering an Address

4.2.9 In the pop-up window, enter the login and password of the Raspberry Pi. The default username is pi, the password is raspberry.

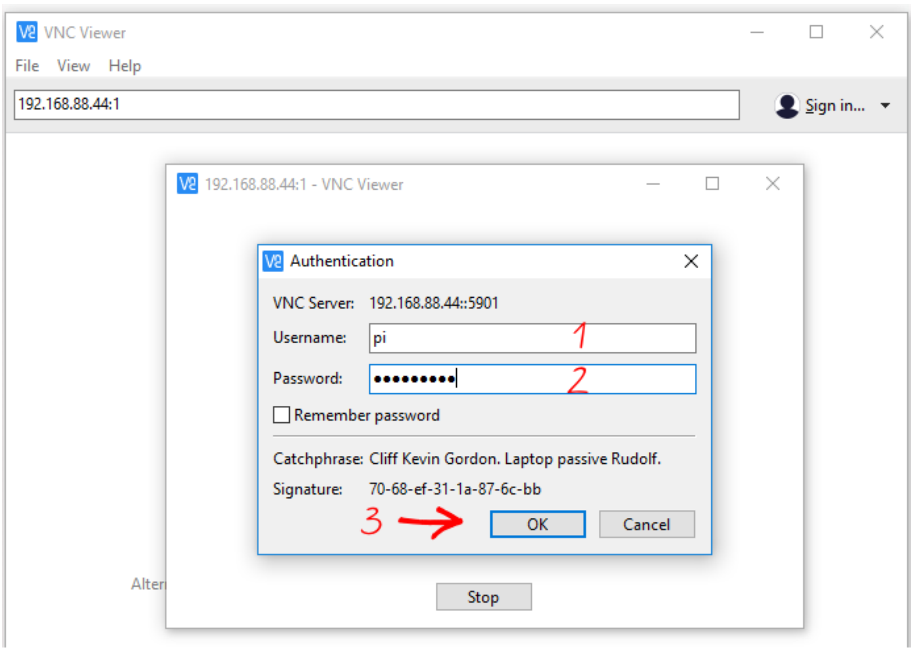

VNC Viewer
File View Help
192.168.88.44:1
Sign in...
192.168.88.44:1 - VNC Viewer
Authentication
VNC Server: 192.168.88.44::5901
Username: pi
1
Password: ●●●●●●●●●
2
Remember password
Catchphrase: Cliff Kevin Gordon. Laptop passive Rudolf.
Signature: 70-68-ef-31-1a-87-6c-bb
3
OK
Cancel
Alter
Stop

Figure 4.7 - Data Entry

Next, click "OK" and before you open the remote desktop Raspberry PI. Now you can work graphically from any PC, rejecting control through the terminal.

## 4.3 Report content

4.3.1 Goal of the work.
4.3.2 Report on the work done with remote access with screenshots.
4.3.3 Conclusions.

## 4.4 Test Questions

1. What is VNC and how is it different from managing SSH?
2. Describe the installation of the VNC server in console mode over SSH.
3. Write a script to start the VNC server automatically by writing it to the init.d directory.

## 5 Laboratory work №5. Installing a web server with PHP, HTML

The goal: to install and configure a web server with the appropriate services for proper operation and further editing.

### 5.1 Summary of the theory

Having your own web server is actually very convenient. Test scripts, display information about the state of the equipment, or simply post a small project - all this is possible with such a server.

Figure 5.1 - Services for the web server

### 5.2 Order of performance of work

5.2.1 To install Apache and PHP, run the following commands:

```
sudo apt-get install apache2 php5 libapache2-mod-php5
```

5.2.2 Now restart the service:

```
sudo service apache2 restart
```

or

```
sudo / ect / init.d / apache2 restart
```

5.2.3 Enter the IP address of your Raspberry Pi in a web browser. You should see a simple page that says "It Works!". To find out the IP address - write the command:

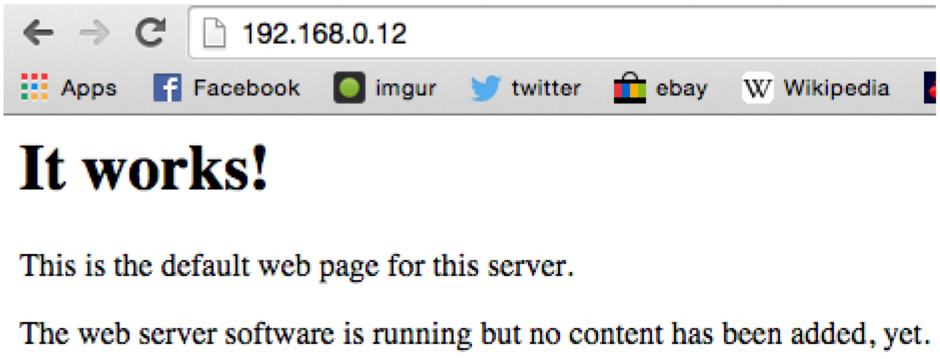
```
inconfig
```

Further, we see the following parameters:

```
Link encap:Ethernet  HWaddr fe:fd:45:xx:xx:xx
inet addr:69.164.xxx.xxx  Bcast:69.164.xxx.xxx  Mask:255.255.255.0
inet6 addr: fe80::fcfd:xxx:xxx:xxx/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:35463998 errors:0 dropped:0 overruns:0 frame:0
TX packets:30563995 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:11300336376 (11.3 GB)  TX bytes:33179556297 (33.1 GB)
Interrupt:76
```

Note: Make sure you are connected to the Internet. If the ethernet cable is connected, the eth0 label will appear. A wireless network may be designated wlan0, but another designation is possible.

The second line "inet addr: 69.164.xxx.xxx" shows the IP address of your Raspberry Pi. Now enter this address into the web browser.



It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

Figure 5.2 - Display as you type

Note: If an error message appears with the following text: "wget: command not found", run "sudo apt-get install wget".

### 5.3 Report content

5.3.1 Goal of the work.
5.3.2 A report on the work done by the web browser with appropriate screenshots.
5.3.4 Conclusions.

### 5.4 Test questions

1. Which of the following packages and services can add to the web page and their purpose?
2. What is the root of the web server files?
3. Using the command “nano test.php” check the operation of the PHP service and set the date and time when you go to the web page with the appropriate commands.

# 6 Laboratory work №6. MySQL server installation

Purpose: installation and configuration of the MySQL server, access, change and storage of information in the database.

## 6.1 Brief information from the theory

MySQL is a relational database management system. MySQL server allows you to work efficiently with data and provides quick access to information simultaneously to several users.

A relational database is a combination of two-dimensional tables that are related to each other. Each table contains a collection of records. In turn, a record is a set of fields containing related information. Any field in the database has a name and a specific type. The table name must be unique within the database. In turn, the field name must be unique within the table.

Specialized SQL language allows you to create databases and tables, add, change and delete data, retrieve data on request.

## 6.2 Order of performance

6.2.1 To install MySQL, install several packages using the following command:

```
sudo apt-get install mysql–server mysql–client php5-mysql
```

6.2.2 We will now set up FTP to transfer files to / from Raspberry Pi. Change root web folder permissions:

```
sudo chown –R pi/var/www
```

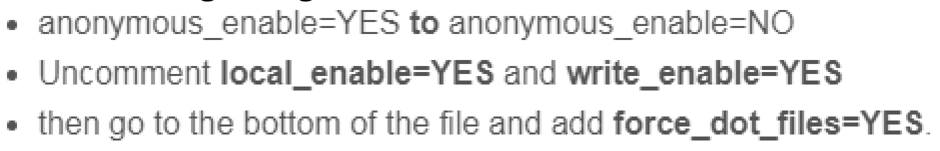
6.2.3 Next, install vsftpd:

```
sudo apt-get install vsftp
```

6.2.4 Edit the vsftpd.conf file:

```
sudo name /etc/vsftp.conf
```

Made following changes:



- anonymous_enable=YES **to** anonymous_enable=NO
- Uncomment **local_enable=YES** and **write_enable=YES**
- then go to the bottom of the file and add **force_dot_files=YES**.

Save the file and exit by pressing CTRL-O, CTRL-X.

6.2.5 Now restart vsftpd:

```
sudo service vsftpd restart
```

6.2.6 Add a quick link from the home folder of the user Pi to / var / www:

```
ln  –s/var/www/~/www
```

Now you can connect to the / var / www folder through a quick link that should appear when logging in via FTP and using the username Pi.

Note: If an error message appears with the following text: "wget: command not found", run "sudo apt-get install wget".

### 6.3 Report content

6.3.1 Goal of the work.

6.3.2 A report on the work done by the web browser with appropriate screenshots.

6.3.3 Conclusions.

### 6.4 Test questions

1. What is the MySQL server for?
2. How is the input and output of information from the database?
3. What functions are assigned to a specialized SQL language?

## 7 Laboratory work №7. Broadcast video from camera to webpage

Objective: the acquisition of practical skills for implementing broadcast images from a USB or CSI camera on a web page.

### 7.1 Summary of the theory

To implement image streaming, you need to install mjpg-streamer on Raspberry Pi.

### 7.2 Order of performance

7.2.1 Open a terminal, and install the libjpeg8-dev library by typing

```
sudo apt-get install libjpeg8-dev
```

7.2.2 Install the cmake application by typing

```
sudo apt-get install cmake
```

7.2.3 Download the mjpg-streamer installation package

```
wget https://github.com/jacksonliam/mjpg-streamer/archive/master.zip
```

7.2.4 Unzip the downloaded archive

```
unzip ./master
```

7.2.5 Go to folder

```
cd / home / pi / mjpg-streamer-master / mjpg-streamer-experimental
```

7.2.5 Build a program

```
make clean all
```

7.2.6 Run the program with the necessary attributes

```
./mjpg-streamer –i "./input_uvc.so -y" -o "./output_http.so -w ./www"
```

7.2.7 Open the page in the Raspberry Pi browser (or any other device on the same network) http://127.0.0.1:8080/?action=stream (for access from another device instead of 127.0.0.1, enter the IP address of the Raspberry Pi).

### 7.3 Report content

7.3.1 Goal of the work.
7.3.2 Description of actions taken.
7.3.3 Screenshots of the results of command execution.
7.3.4 Conclusions.

### 7.4 Test questions

1. What is the cmake package for?
2. What is the function of the wget command?
3. How to set the resolution and frame rate of the broadcast?
4. How to extract the contents of the archive?

# 8 Laboratory Work №8. Connecting a web server using Flask

Objective: to provide access to the web server and create a way to control the LEDs.

## 8.1 Theory brief

Raspberry Pi allows you to create a stand-alone web server that can enable / disable two LEDs, but they can be changed to other data output devices. For example, on the relay or thyristors. To create a web server, you need a Flask microframe using Python.

To complete the project, you will need the following additional equipment: a mock-up board, 2 LEDs, 2 resistors with a nominal value of 470 Ohms, jumper wires.

## 8.2 Order of performance

8.2.1 In order to turn the Raspberry Pi into a web server, you will need a Flask microform. To install Flask, you must first install pip. To update the Raspberry Pi and install pip, run the following commands:

8.2.2 Now with the help of pip install Flask and its dependencies:

8.2.3 The layout for this project is very simple. Just connect 2 LEDs to

GPIO-pins 23 and 24, as shown in the picture below.

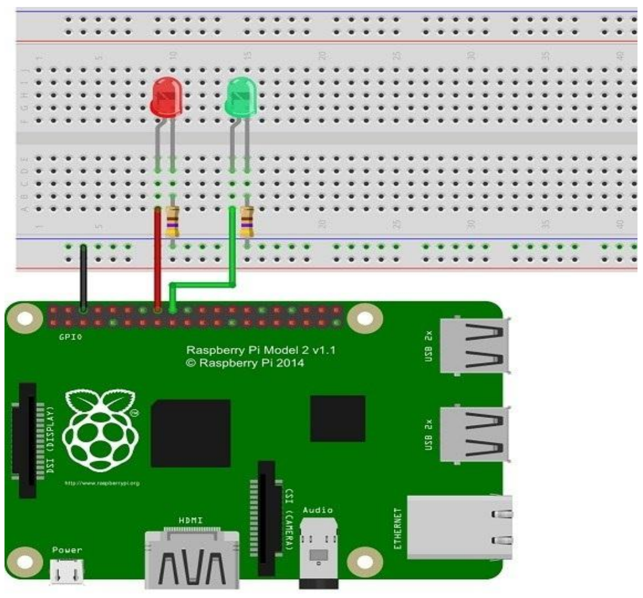


Рисунок 8.1 – Схема подключения

8.2.4 Create a python script. This script is a key component of the project. It configures the web server and interacts with the Raspberry Pi GPIO contacts. To make everything well organized, we start by creating a new folder:

```
pi@raspberrypi ~ $ mkdir web-server
pi@raspberrypi ~ $ cd web-server
pi@raspberrypi:~/web-server $
```

8.2.5 Create a new file called “app.py”:

```
pi@raspberrypi:~/web-server $ nano app.py
```

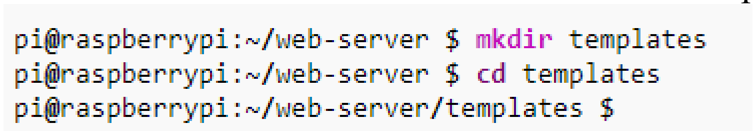
8.2.6 Copy and paste the script shown below into the Raspberry Pi:

```
import RPi.GPIO as GPIO
    from flask import Flask, render_template, request
    app = Flask (__ name__)
    GPIO.setmode (GPIO.BCM)
    # Create a dictionary called “pins” to store data about the numbers,
    # names and states of contacts:
    pins = {
      23: {'name': 'GPIO 23', 'state': GPIO.LOW},
      24: {'name': 'GPIO 24', 'state': GPIO.LOW}
      }
    # We set each contact to the output mode,
    # and also give them the value LOW:
    for pin in pins:
      GPIO.setup (pin, GPIO.OUT)
      GPIO.output (pin, GPIO.LOW)
    @ app.route ("/")
```

```
def main ():
  # We read the status of all contacts and save in the dictionary "pins":
  for pin in pins:
    pins [pin] ['state'] = GPIO.input (pin)
  # Put the “pins” dictionary into the “templateData” dictionary:
  templateData = {
    'pins': pins
    }
  # Pass the "templateData" dictionary to the HTML template,
  # and then return it to the user:
  return render_template ('main.html', ** templateData)
# This function is executed when someone requests a URL with information.
# about the contact number and the action to be performed:
@ app.route ("/ <changePin> / <action>")
def action (changePin, action):
  # Convert a contact from the URL to an integer:
  changePin = int (changePin)
  # Retrieve the device name:
  deviceName = pins [changePin] ['name']
  # If the action in the URL is “on” (i.e. “enable”),
  # then execute this code:
  if action == "on":
    # set the contact value HIGH:
    GPIO.output (changePin, GPIO.HIGH)
    # Make a status message that will be passed to the HTML template:
    message = "Turned" + deviceName + "on."
  # If the action in the URL is “off” (i.e. “off”),
  # then execute this code:
  if action == "off":
    # set the contact value LOW:
    GPIO.output (changePin, GPIO.LOW)
    # Make a status message that will be passed to the HTML template:
    message = "Turned" + deviceName + "off."
  # We read the status of all contacts and save in the dictionary "pins":
  for pin in pins:
    pins [pin] ['state'] = GPIO.input (pin)
  # Put the “pins” dictionary into the “templateData” dictionary:
  templateData = {
    'pins': pins
  }
  return render_template ('main.html', ** templateData)
if __name__ == "__main__":
  app.run (host = '0.0.0.0', port = 80, debug = True)
```

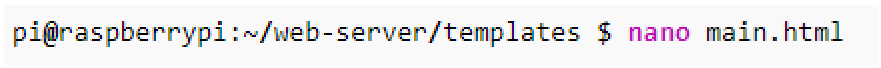
8.2.7 Create an HTML file. In order for the project to be well organized, HTML tags must be kept separate from the Python script. Flask uses Jinja2, an

HTML template engine that can be used to transfer dynamic data from a Python script to an HTML file. Create a new folder called “templates”:



```
pi@raspberrypi:~/web-server $ mkdir templates
pi@raspberrypi:~/web-server $ cd templates
pi@raspberrypi:~/web-server/templates $
```

Create a new file called “main.html”:



```
pi@raspberrypi:~/web-server/templates $ nano main.html
```

8.2.8 Copy and paste the template shown below into the Raspberry Pi:

```
<! DOCTYPE html>
<head>
  <title> RPi Web Server </ title>
  <! —- Last compressed and compiled CSS ->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGm
kzs7" crossorigin="anonymous">
  <!— Optional Theme-->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css" integrity="sha384-
fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En
5r" crossorigin="anonymous">
  <!-- Last compressed and compiled javascript-->
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-
0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ
7xS" crossorigin="anonymous"></script>
</head>
<body>
  <h1>RPi Web Server</h1>
  {% for pin in pins %}
  <h2>{{ pins[pin].name }}
  {% if pins[pin].state == true %}
    is currently <strong>on</strong></h2><div class="row"><div
class="col-md-2">
    <a href="/{{pin}}/off" class="btn btn-block btn-lg btn-default"
role="button">Turn off</a></div></div>
  {% else %}
    is currently <strong>off</strong></h2><div class="row"><div
class="col-md-2">
    <a href="/{{pin}}/on" class="btn btn-block btn-lg btn-primary"
role="button">Turn on</a></div></div>
```

```
   {% endif %}
   {% endfor %}
</body>
</html>
```

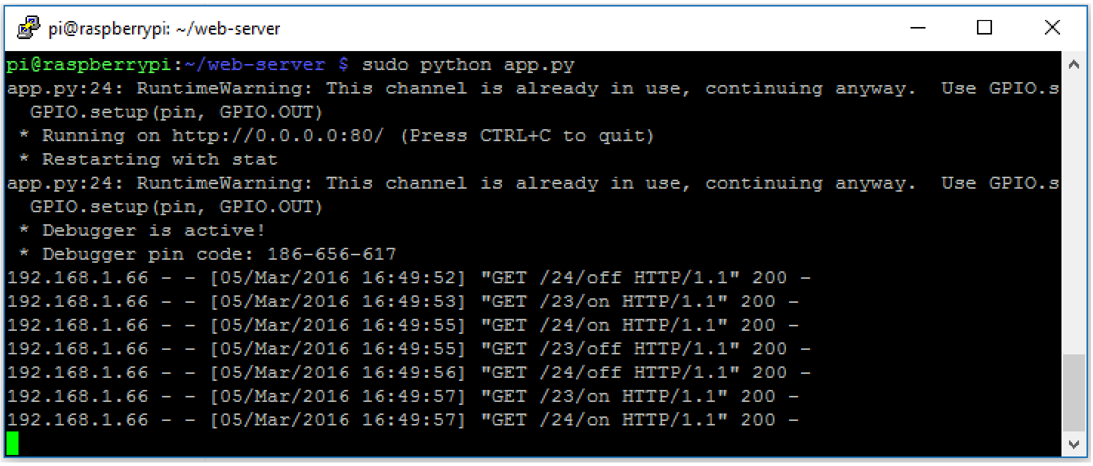
8.2.9 We start the web server. To start the Raspberry Pi web server, navigate to the folder that contains the app.py file::

```
pi@raspberrypi:~/web-server/templates $ cd ..
```

Then run the following command:

```
pi@raspberrypi:~/web-server $ sudo python app.py
```

Immediately the web server should start, clearly shown in the figure.



```
pi@raspberrypi: ~/web-server
pi@raspberrypi:~/web-server $ sudo python app.py
app.py:24: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.s
  GPIO.setup(pin, GPIO.OUT)
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
 * Restarting with stat
app.py:24: RuntimeWarning: This channel is already in use, continuing anyway.  Use GPIO.s
  GPIO.setup(pin, GPIO.OUT)
 * Debugger is active!
 * Debugger pin code: 186-656-617
192.168.1.66 - - [05/Mar/2016 16:49:52] "GET /24/off HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:53] "GET /23/on HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:55] "GET /24/on HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:55] "GET /23/off HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:56] "GET /24/off HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:57] "GET /23/on HTTP/1.1" 200 -
192.168.1.66 - - [05/Mar/2016 16:49:57] "GET /24/on HTTP/1.1" 200 -
```

Figure 8.2 - Starting the web server

Next, open a browser and enter the IP address of the Raspberry Pi.

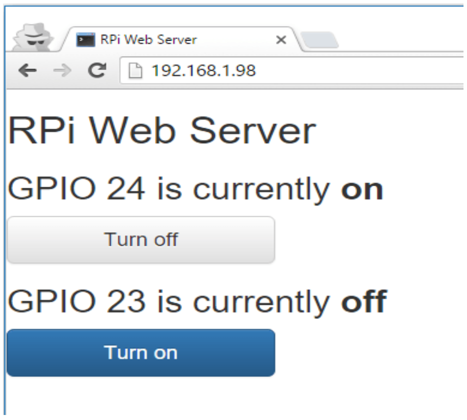


Figure 8.3 - Web Server Operation

Demonstrating, try pressing and testing the LED on / off parameters.

## 8.3 Report Contents

8.3.1 Goal of the work.
8.3.2 Report on the work done and screenshots about the performance of the

website.

8.3.3 Changes in the selection of each student unique GPIO port.

8.3.4 Conclusions.

### 8.4 Test questions

1. Using the available elements, connect other devices instead of diodes.
2. Having the skills, change the design of the site and simplify in their own way.
3. Is it possible to connect the Arduino platform to this lab? Explain what changes would apply to this scheme.

## 9 Laboratory work №9. Create a web server with temperature and humidity data

Objective: Create a stand-alone web server that displays temperature and humidity data read from a DHT22 sensor.

### 9.1 Summary of the theory

With ESP8266 and the MQTT protocol, you can control two data output devices (for example, LEDs).

To create a web server, we use the Flask microframe using the Python programming language. The scheme of the designed system is shown below.

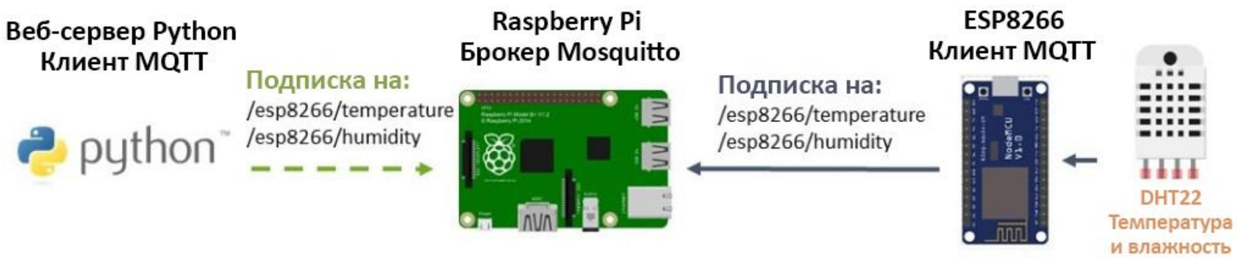


Figure 9.1 - Telemetry system diagram

### 9.2 Order of performance

9.2.1 The Raspberry Pi will communicate with the ESP8266 using the MQTT protocol. Therefore, we first install the Mosquitto broker, and then run it in the background:

```
pi@raspberry:~ $ mosquitto -d
```

9.2.2 Further, we need pip packages from previous work in order for Flask to function, we will enter the commands:

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
pi@raspberrypi ~ $ sudo apt-get install python-pip python-flask git-core
```

9.2.3 Now with the help of pip install Flask and Paho MQTT:

```
pi@raspberrypi ~ $ sudo pip install flask
pi@raspberrypi ~ $ sudo pip install paho-mqtt
```

Note: This project uses SocketIO, which allows you to create a Python Flask web page that can be updated asynchronously using a Python Flask program. This means that you will not need to update the web page to see the latest data, because they are constantly updated.

9.2.4 To install the Flask SocketIO package, enter the following:

```
pi@raspberrypi ~ $ sudo pip install flask-socketio
```

9.2.5 Creating a python script. This script is a key component of the project. It configures the web server, and when you click a button, it publishes an MQTT message on the ESP8266. In addition, he is subscribed to MQTT topics with information about humidity and temperature, which allows him to constantly read this data. To make everything well organized, we start by creating a new folder:

```
pi@raspberrypi ~ $ mkdir web-server
pi@raspberrypi ~ $ cd web-server
pi@raspberrypi:~/web-server $
```

Create a new file called “app.py”:

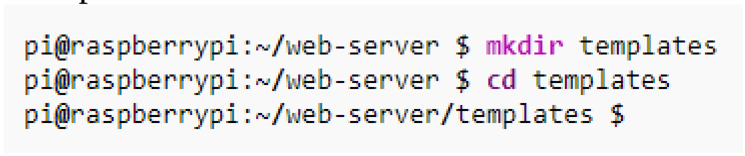
```
pi@raspberrypi:~/web-server $ nano app.py
```

Next, copy and paste into Raspberry Pi the script shown below:

```
import paho.mqtt.client as mqtt
from flask import Flask, render_template, request
from flask_socketio import SocketIO, emit
app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'
socketio = SocketIO(app)
# The callback function to be called
# when the client receives a CONNACK response from the server:
def on_connect(client, userdata, flags, rc):
   print("Connected with result code "+str(rc))
   # If you subscribe to on_connect (), then if you lose the connection
    # you can simply reconnect and restore subscriptions
   client.subscribe("/esp8266/temperature")
   client.subscribe("/esp8266/humidity")
# Callback function in case
# if a message comes from ESP8266 PUBLISH:
def on_message(client, userdata, message):
   #socketio.emit('my variable')
   print("Received message '" + str(message.payload) + "' on topic '"
       + message.topic + "' with QoS " + str(message.qos))
   if message.topic == "/esp8266/temperature":
       print("temperature update")
       socketio.emit('dht_temperature', {'data': message.payload})
   if message.topic == "/esp8266/humidity":
       print("humidity update")
       socketio.emit('dht_humidity', {'data': message.payload})
mqttc=mqtt.Client()
mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.connect("localhost",1883,60)
mqttc.loop_start()
# Create a dictionary called "pins" to store data about the numbers,
# names and states of contacts:
pins = {
  4 : {'name' : 'GPIO 4', 'board' : 'esp8266', 'topic' : 'esp8266/4', 'state' :
'False'},
  5 : {'name' : 'GPIO 5', 'board' : 'esp8266', 'topic' : 'esp8266/5', 'state' :
'False'}
  }
# Put the "pins" dictionary into the "templateData" dictionary:
templateData = {
  'pins' : pins
  }
@app.route("/")
def main():
  # Pass the "templateData" dictionary to the HTML template,
  # and then return it to the user:
  return   render_template('main.html',   async_mode=socketio.async_mode,
**templateData)
# This function is executed when someone requests a URL with information.
# about the contact number and the action to be performed:
@app.route("/<board>/<changePin>/<action>")
def action(board, changePin, action):
# Convert a contact from the URL to an integer:
  changePin = int(changePin)
  # Retrieve the device name:
  devicePin = pins[changePin]['name']
  # If the action in the URL is "1", then execute this code:
  if action == "1" and board == 'esp8266':
   mqttc.publish(pins[changePin]['topic'],"1")
   pins[changePin]['state'] = 'True'
  if action == "0" and board == 'esp8266':
   mqttc.publish(pins[changePin]['topic'],"0")
   pins[changePin]['state'] = 'False'
```

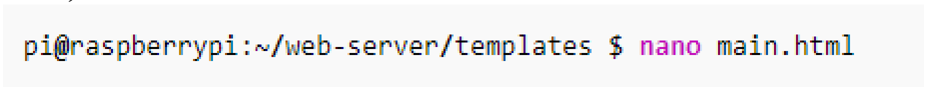
```
    # We place in the “templateData” dictionary the “pins” dictionary and the message:
    templateData = {
      'pins' : pins
    }
    return render_template('main.html', **templateData)
@socketio.on('my event')
def handle_my_custom_event(json):
    print('received json data here: ' + str(json))
if __name__ == "__main__":
    socketio.run(app, host='0.0.0.0', port=8181, debug=True)
```

9.2.6 As in previous work, we create an HTML file. For a project to be well organized, HTML tags must be kept separate from the Python script. Flask uses Jinja2, an HTML template engine that allows you to transfer dynamic data from a Python script to an HTML file. Create a new folder called “templates”:



```
pi@raspberrypi:~/web-server $ mkdir templates
pi@raspberrypi:~/web-server $ cd templates
pi@raspberrypi:~/web-server/templates $
```

Next, create a new file called “main.html”:



```
pi@raspberrypi:~/web-server/templates $ nano main.html
```

Copy and paste into the Raspberry Pi the template shown below:

```
<!DOCTYPE html>
<head>
  <title>RPi Web Server</title>
  <!-- Последний сжатый и скомпилированный CSS -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGm
kzs7" crossorigin="anonymous">
  <!-- Опциональная тема -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css" integrity="sha384-
fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En
5r" crossorigin="anonymous">
  <!-- Последний сжатый и скомпилированный JavaScript -->
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-
0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ
7xS" crossorigin="anonymous"></script>
```

```
  <script src="https://code.jquery.com/jquery-3.1.1.min.js" integrity="sha256-
hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
crossorigin="anonymous"></script>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/socket.io/1.3.6/socket.io.min.js"></script>
  <script type="text/javascript" charset="utf-8">
    $(document).ready(function() {
      var socket = io.connect('http://' + document.domain + ':' + location.port);
      socket.on('connect', function() {
        socket.emit('my event', {data: 'I\'m connected!'});
      });
      socket.on('dht_temperature', function(msg) {
        var nDate = new Date();
        $('#readingsUpdated').text(nDate.getHours() + 'h:' + nDate.getMinutes() +
           'm:' + nDate.getSeconds() + 's').html();
        $('#temperature').text(msg.data).html();
      });
      socket.on('dht_humidity', function(msg) {
        $('#humidity').text(msg.data).html();
      });
    });
  </script>
</head>
<body>
  <h1>RPi Web Server - ESP8266 MQTT</h1>
  {% for pin in pins %}
  <h2>{{ pins[pin].name }}
  {% if pins[pin].state == 'True' %}
    is currently <strong>on</strong></h2><div class="row"><div class="col-md-
2">
    <a href="/esp8266/{{pin}}/0" class="btn btn-block btn-lg btn-default"
role="button">Turn off</a></div></div>
  {% else %}
    is currently <strong>off</strong></h2><div class="row"><div class="col-md-
2">
    <a href="/esp8266/{{pin}}/1" class="btn btn-block btn-lg btn-primary"
role="button">Turn on</a></div></div>
  {% endif %}
  {% endfor %}
  <h3>DHT Readings (updated <span id="readingsUpdated"></span>)</h3>
  <h3>Temperature: <span id="temperature"></span>ºC</h3>
  <h3>Humidity: <span id="humidity"></span>%</h3>
</body>
</html>
```

### 9.3 Report Contents

9.3.1 Goal of the work.
9.3.2 A screenshot of the measured temperature and humidity of your room.
9.3.3 Conclusions.

### 9.4 Test questions

1. How does a DHT11 sensor differ from a DHT22?
2. List the advantages and disadvantages of this sensor.
3. If the DHT11 sensor is connected instead of the specified one, will the measurement result change?
4. If a DHT11 sensor is connected instead of the second LED, will the first LED light up?

# 10 Laboratory Work №10. Development of a telemetry system using IOT technology

Objective: programming of the ESP8266 chip and creation of a telemetry system using IOT technology.

## 10.1 Order of performance

10.1.1 Turn on the personal computer, or Raspberry Pi (in case it has an Arduino IDE installed). In order to the ESP8266 chip to communicate with the Raspberry Pi web server, you will need to install the PubSubClient library. With its help, you can create a client capable of sending messages as a subscription / publication to a server that supports MQTT (in fact, this allows the ESP8266 to "communicate" with a web server in Python).

10.1.2 You need to download the library PubSubClient, or take from the teacher. Unzip the downloaded ZIP archive. As a result, you should have a folder called "pubsubclient-master". Rename this folder to "pubsubclient". Move the pubsubclient folder to the Arduino IDE's libraries folder.

Note: Included with the library are a few example sketches. To see them, click the Arduino IDE at File> Examples> PubSubClient (File> Examples> PubSubClient).

10.1.3 Installing a library for a DHT sensor. We take the ZIP-archive from the teacher. This library makes it easy to use any DHT sensor to read temperature and humidity data using the ESP8266 or Arduino board.

10.1.4 Unzip the downloaded ZIP archive. As a result, you should have a folder called “DHT-sensor-library-master”. Rename it to "DHT". Move the DHT folder to the Arduino IDE libraries folder. Reopen the Arduino IDE again.

10.1.5 Finally, upload the sketch to ESP8266 (replace the SSID, password and IP address of the Raspberry Pi):

```
// load the ESP8266WiFi, PubSubClient and DHT libraries:
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
/ remove comment marks from one of the lines below,
// corresponding to a DHT sensor type:
// # define DHTTYPE DHT11 // DHT 11
// # define DHTTYPE DHT21 // DHT 21 (AM2301)
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
// change these constants so that ESP8266 can connect to the router:
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
/ change the variable below to the Raspberry Pi IP address,
// so that she can connect to the broker MQTT:
const char * mqtt_server = "YOUR_RPi_IP_Address";
// initialize espClient:
WiFiClient espClient;
PubSubClient client (espClient);
// connect the LEDs to the corresponding GPIO-pins ESP8266:
const int ledGPIO5 = 5;
const int ledGPIO4 = 4;
// DHT sensor:
const int DHTPin = 14;
// initialize the DHT sensor:
DHT dht (DHTPin, DHTTYPE);
// auxiliary variables for the timer:
long now = millis ();
long lastMeasure = 0;
// do not change the function below; it connects the ESP8266 to the router:
void setup_wifi () {
   delay (10);
   // Start by connecting to a WiFi network:
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("WiFi connected - ESP IP address: ");
// "WiFi connection done - IP address ESP8266:"
   Serial.println (WiFi.localIP ());
```

```
}
// this function is executed when a device publishes a message
// to the topic that ESP8266 is subscribed to; change the function below
// in accordance with the logic of your program - so that when the device
// will post the message to the topic that ESP8266 is subscribed to,
// your program worked as needed:
void callback (String topic, byte * message, unsigned int length) {
   Serial.print ("Message arrived on topic:");
   // "The message arrived at the topic:"
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;
  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();
  // if your project does not have 2 LEDs, but more
   // add more code below:
   // if the topic home / office / esp1 / gpio2 received a message,
   // see what this message is - “0” or “1”;
   // switch GPIO-contact in accordance with the value sent:
   if (topic == "esp8266 / 4") {
       Serial.print ("Changing GPIO 4 to");
       // "Changing the state of GPIO-contact 4 to"
      if(messageTemp == "1"){
        digitalWrite(ledGPIO4, HIGH);
        Serial.print("On");
      }
      else if(messageTemp == "0"){
        digitalWrite(ledGPIO4, LOW);
        Serial.print("Off");
      }
  }
  if(topic=="esp8266/5"){
      Serial.print("Changing GPIO 5 to ");
// "Changing the state of GPIO pin 5 to"
      if(messageTemp == "1"){
        digitalWrite(ledGPIO5, HIGH);
        Serial.print("On");
      }
      else if(messageTemp == "0"){
        digitalWrite(ledGPIO5, LOW);
        Serial.print("Off");
      }
   }
   Serial.println();
  }
  // this function reconnects ESP8266 to the MQTT broker;
  // change this function if you want ESP8266 to subscribe
  // for more topics:
  void reconnect () {
    // rerun the cycle until we connect:
    while (! client.connected ()) {
      Serial.print ("Attempting MQTT connection ...");
     // "Attempting to connect to an MQTT broker ..."
     // Trying to connect:
         / * if you have problems with connecting several devices to mqtt
broker, change the link below.
       * To change the chip ID of ESP8266, it needs to be given  unique name.
So it looks now:
       if (client.connect ("ESP8266Client")) {
       * If you want to connect to the MQTT broker additional devices, it can
be called:
       if (client.connect ("ESPOffice")) {
       * For other ESPs:
       if (client.connect ("ESPGarage")) {
       This should fix the connectivity issue.
       multiple devices to an MQTT broker:
       Also note that this name must
       match what will be shown below
       in the loop () block.
         * /
     if (client.connect ("ESP8266Client")) {
       Serial.println ("connected"); // "connected"
       // subscribe or re-subscribe to the topic;
       // you can subscribe not only to one, but to several topics
       // (as for this particular example, this will allow
       // manage a large number of LEDs):
       client.subscribe("esp8266/4");
       client.subscribe("esp8266/5");
     } else {
       Serial.print("failed, rc=");
       Serial.print(client.state());
       Serial.println(" try again in 5 seconds");
        // "5 seconds before the next attempt"
       // wait 5 seconds before trying again:
       delay (5000);
      }
    }
```

```
}
// this function configures GPIO-contacts ESP8266
// to output mode, starts serial communication
// at a speed of 112500 baud, sets up the MQTT broker
// and sets the callback function;
// callback function is used to receive messages and,
// Actually, LED control:
void setup() {
  dht.begin();
  pinMode(ledGPIO4, OUTPUT);
  pinMode(ledGPIO5, OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}
/ for this project in the loop () function, nothing needs to be changed;
// essentially this function establishes the connection
// between ESP8266 and MQTT broker:
void loop () {
   if (! client.connected ()) {
     reconnect ();
   }
   if (! client.loop ())
      / * if you have problems with connecting several devices
      to mqtt broker, change the link below.
        * To change the chip ID of ESP8266, it needs to be given
          unique name. So it looks now:
       if (client.connect("ESP8266Client")) {
```
* If you want to connect to the MQTT broker several devices, it can be called:
```
        if (client.connect ("ESPOffice")) {
        * For other ESPs:
        if (client.connect ("ESPGarage")) {
        This should fix the connectivity issue.
        multiple devices to an MQTT broker:
        Also note that this name must
        match what will be shown below
        in the loop () block.
     */
    client.connect("ESP8266Client");
  now = millis();
  // publish new data on temperature and humidity
   // every 10 seconds:
   if (now - lastMeasure> 10000) {
     lastMeasure = now;
     // data from the sensor may be 2 seconds behind
     // (this is a very slow sensor)
     float h = dht.readHumidity ();
     // read the temperature in Celsius (by default):
     float t = dht.readTemperature ();
     // read the temperature in Fahrenheit (isFahrenheit = true):
     float f = dht.readTemperature (true);
     // check if the data was obtained from the sensor
     // and if not, end the operation and try again:
    if (isnan(h) || isnan(t) || isnan(f)) {
      Serial.println("Failed to read from DHT sensor!");
      // "Could not read data from the sensor!"
       return;
     }
     // calculate the temperature in Celsius:
     float hic = dht.computeHeatIndex (t, h, false);
     static char temperatureTemp [7];
     dtostrf (hic, 6, 2, temperatureTemp);
     // if you want to calculate the temperature in Fahrenheit,
     // remove commenting marks from the lines below:
    // float hif = dht.computeHeatIndex(f, h);
    // static char temperatureTemp[7];
    // dtostrf(hic, 6, 2, temperatureTemp);
    static char humidityTemp[7];
    dtostrf(h, 6, 2, humidityTemp);
// publish data on temperature and humidity:
    client.publish("/esp8266/temperature", temperatureTemp);
    client.publish("/esp8266/humidity", humidityTemp);
    Serial.print("Humidity: ");
    Serial.print(h);
    Serial.print(" %\t Temperature: ");
    Serial.print(t);
    Serial.print(" *C ");
    Serial.print(f);
    Serial.print(" *F\t Heat index: ");
    Serial.print(hic);
    Serial.println(" *C ");
    // Serial.print(hif);
    // Serial.println(" *F");
  }
}
```

10.6 To complete the project, you will need the following components: one ESP8266 12E module, one DHT22 sensor, one 4700 Ohm resistor, two 470 Ohm resistors, two LEDs.

Note: Other models of the DHT sensor will work for this project, but this will require some minor changes to the code.

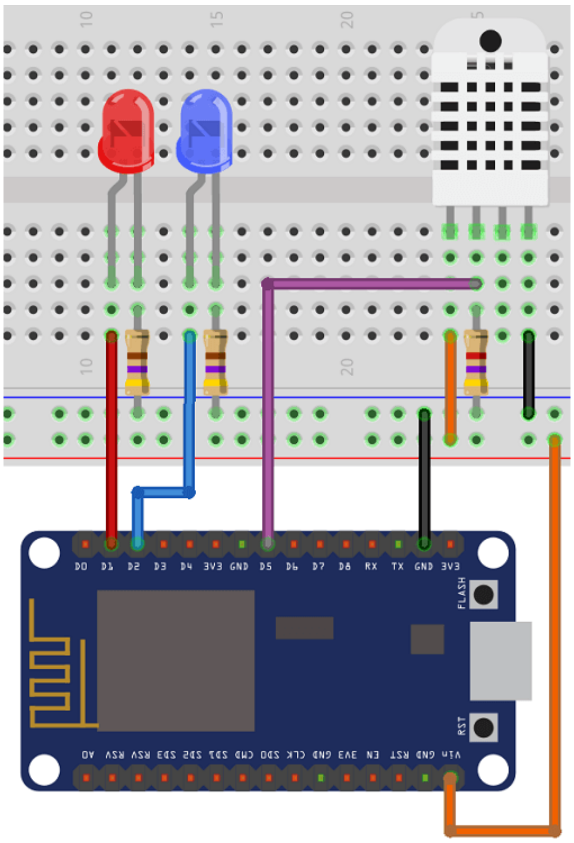

10
15
20
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
RST
A0 RSV RSV SD3 SD2 SD1 CMD SD0 CLK GND 3V3 EN RST GND Vin

Figure 10.2 - Connection Diagram

Note: The DHT sensor will need 5 volts for proper operation, so be sure to connect it to the Vin pin on the ESP8266, which outputs 5 volts.

10.1.7 To start the Raspberry Pi web server, go to the folder containing the app.py file:

```
pi@raspberrypi:~/web-server/templates $ cd ..
```

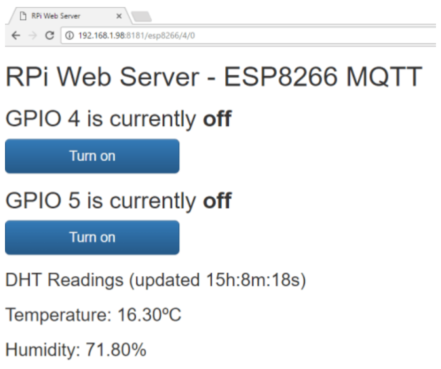
Then run the following command:

```
pi@raspberrypi:~/web-server $ sudo python app.py
```

Immediately after this, the web server should start, and this should happen on the port ": 8181".

Enter the IP address of the Raspberry Pi in your browser. In my case, this is http://192.168.1.98:8181.

Note: After the IP-address must be a postscript ": 8181".



RPi Web Server - ESP8266 MQTT

GPIO 4 is currently **off**

GPIO 5 is currently **off**

DHT Readings (updated 15h:8m:18s)

Temperature: 16.30ºC

Humidity: 71.80%

Figure 10.3 Demonstrating a webpage

As you can see, the web server is in working condition, temperature and humidity data are updated asynchronously in the SocketIO system, namely, the latest data will be updated constantly. Show the teacher the data on temperature and humidity and the performance of the LED buttons connected to the corresponding ports.

## 10.2 Report Contents

10.2.1 Goal of the work.
10.2.2 Make changes in the code taking into account other ports for LEDs.
10.2.3 Measure the temperature and humidity of your room.
10.2.4 Conclusions.

## 10.3 Test questions

1. What library do I need to install the library so that the ESP8266 chip can communicate with the Raspberry Pi web server?
2. What does the callback function do?
3. What is the function of the EQ8266-12E MQTT broker?
4. What channel is the ESP8266-12E chip connected to and the Raspberry Pi microcomputer?

# Bibliography

1 Vasiliev A.N. Python examples. Practical course on programming. - SPb.: Science and technology, 2017. - 432 p.
2 N.Voitov Basics of working with Linux. Training course. - M .: DMK

Press, 2010 - 216 p.

3 Igo T. Arduino, sensors and networks for communication devices: Trans. from English - SPb .: BHV-Petersburg, 2016. - 544 p.

4 Petin V.A. Arduino and Raspberry Pi in Internet of Things projects. - SPb.: BHV-Petersburg, 2016. - 464 p.

5 Petin V.A. - Microcomputers Raspberry Pi. A practical guide. SPb .: BHV-Petersburg, 2015. - 240 p.

6 Petin V.A. Projects using the Arduino controller - SPb .: BHV-Petersburg, 2016. - 464 p.

7 Prokhorenok N.A. HTML, JavaScript, PHP and MySQL. Dzhenlmensky set of Web-masters. - SPb .: BHV-Petersburg, 2019. - 912 p.

## Content

Master plan 2019 г., poz.32

Baikenov Bakhytzhan Sergeevich
Ayazbay Abu-Alim Erikuly

# COMPUTER TECHNOLOGIES IN INSTRUMENT MAKING

Guidelines for the implementation of laboratory works
for students majoring 5B071600 – Instrument making

Editor M.D. Kurmanbekova

Standardization Specialist G. Mukhametsarieva

| | |
|---|---|
| Signed to print 28.06.19 | Format 60x84 1/16 |
| Circulation of 20 copies | Typographical paper № 1 |
| Volume 2.0 publishing sheets | Order _____Price 1000 tg. |

Copying office non-commercial joint stock company
"Almaty University of Power and Communications"
050013, Almaty, Baitursynov, 126/1