PiRobo User Manual

1. General Usage

1.1 Settings Menu

The following information are shown on the display:

```
SSID: <ssid of connected wifi>
IP: <IP address of robot>
CPU Usage: <cpu usage of all 4 cores>
V <sw version>  ID <hw identifier>
```

As soon as the joystick next to the display is operated, a menu is shown. Just push the joystick left to go back to the main screen.

The items *Shutdown* and *Reboot* allow to shutdown and reboot the robot. **It is not recommended to just power off the robot without first initiating a shutdown** (as it could potentially lead to data loss or system failure).

The item *Change Wifi* allows to connect to a different wifi. Upon selection, the robot scans for nearby networks and shows a list of available SSIDs. Again push the joystick left to exit. Otherwise, if a SSID is chosen, the password prompt appears. Enter the password of the wifi and confirm by selecting *OK*. *DEL* deletes the last character, *EXIT* leaves the password prompt. If confirmed, the robot reboots to apply the new setting. If the password was correct, the SSID of the new wifi will be shown on the display.

Hint: The display update and joystick read-out are quite slow, thus the menu navigation and entering of a new password might require some patience ;-).

Hint: You can also just connect the robot to a wired network

1.2 Main Screen

Open a web-browser on a computer that is connected to the same network as your robot. Enter the ip-address of the robot in the address bar to get redirected to the main screen. The main-screen...

- shows the SW version and HW identifier
- contains a link to the code editor
- allows to compile, run and stop the compiled code
- shows the output of the running program (see chapter 2)
- sends keyboard and touch inputs to the program (see chapter 2)
- shows the camera preview
- allows to shutdown and reboot the robot

1.3 Code Editor

A link on the main screen opens ICECoder[1], check out the documentation on their website if needed. In general, only the file *main.c* needs to be edited. However, you can create as many additional files and include them from *main.c* as you need. Files in subfolders are ignored. Remember to first save your code before compiling it on the main screen.

2. SW Interface Documentation

2.1 Initialization

Before calling any driver function, *init_drivers()* should be called once. It does not return anything.

2.2 Wait

The functions *sleep(unsigned int seconds)* and *usleep(useconds_t micro_seconds)* allow to delay the execution of the program.

Important: You are programming on a CPU with an operating system, not on a microcontroller. Regularly call wait-functions to indicate to the operating system that you don't need to be running right now. A CPU usage of about 25% might indicate that you are fully occupying one of the four cores, which uses a lot of power and slows down the complete system.

Example:
Don't do this:

```
while(!key_w_state);
```

Instead do something like this:

```
while(!key_w_state) usleep(50000);
```

2.3 Print

The function *printf()* is available to print any output to the web interface, see the documentation[2].

Important: The output is only sent to the website after a new line (\n) is printed.

Examples:

```
printf("Hello World!\n");   // 'Hello World!'
printf("Var x is %d\n", x); // 'Var x is 5', x is integer
printf("Var a is %f\n", a); // 'Var a is 3.14', a is float/double
```

[1] https://icecoder.net/
[2] http://man7.org/linux/man-pages/man3/printf.3.html

2.4 Keyboard

There are different global variables indicating the keyboard state of a client using the web interface. For example, the variable *key_w_state* can be read to find out whether someone is pressing the button 'W' on their computer, while having the web interface opened (and focused). All variables are integers, but contain binary values. The following variables can be used:

```
key_w_state, key_a_state, key_s_state, key_d_state, key_i_state,
key_j_state, key_k_state, key_l_state, key_0_state, key_1_state,
key_2_state, key_3_state, key_4_state, key_5_state, key_6_state,
key_7_state, key_8_state, key_9_state
```

If you use a touchscreen-device, you can also use the keys in the 'Keyboard Input' panel. As long as your finger touches the button, the corresponding flag is set.

2.5 Joystick

On touchscreen-enabled devices, you can move the joystick on the main-screen with your finger. The variables `touch_x_state` and `touch_y_state` contain the x- and y-coordinate of the joystick. They are between 0 and 100, with (50,50) indicating the neutral position. The origin of the coordinate system is the top left corner.

2.6 Motors

The function *set_motors(float left, float right);* controls the motors. The arguments should be floats between -1 and +1, where -1 means fully backwards and +1 fully forwards. A value of 0 turns the motor off (and thus saves a lot of power).

2.7 Servos

The function *set_servos(float pan, float tilt);* controls the camera servos. The arguments should be between 0 and 1, where 0.5 indicates a neutral position (however, this depends on how you assembled the camera module of course).

2.8 Line Sensor

A call to the function *get_linevalues(unsigned int *data);* writes 5 integers to the buffer pointed by *data*. The values correspond to the brightness at the 5 sensors. Finding out the range / interpretation of these values is part of your task.

Example:

```
unsigned int data[5];
get_linevalues(data);
printf("Data0: %d, Data4: %d", data[0], data[4]);
```

3. Admin Control

If you click on the HW-identifier (the long number) five times and confirm the question, you can enable admin mode (disabling works the same way). Now, you have a new link in the title bar which allows you to drive the robot without user code. The arrow keys allow you to control the robot, the WASD-keys control the camera servos. If you visit the website on a smartphone, you can drag the virtual joystick below the camera preview to control the robot and drag the camera preview itself to control the servos.

Additionally, a software called *RPi Cam Web Interface*[3] is installed, just click on the camera preview in admin mode and you'll see it. It allows you to take pictures and videos, download them, control the camera settings and much more.

4. SW Concepts

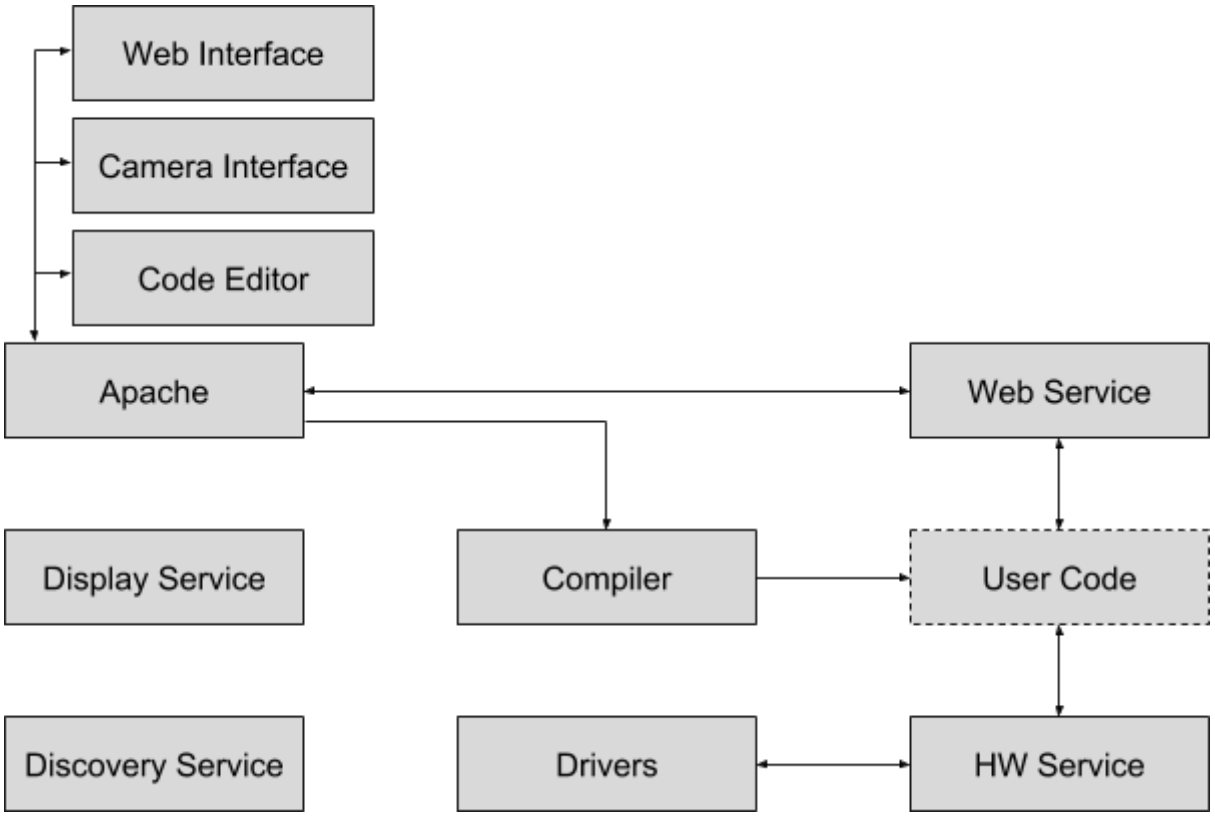
This section contains an overview of the software installed on PiRobo. It is meant as a help to understand the code and adapt it to your own needs. If you are new to linux/raspberry pi/ssh/…, first consulting a tutorial might help :-).

To access the raspberry pi, login via ssh. The user-name and password are the default ones on each raspberry pi (user: *pi*, pwd: *raspberry*). To transfer and modify files, sftp might be helpful.

You do not need to dismount the raspberry pi from the robot to e.g. connect a monitor and access the robot locally, ssh allows for full control over the network.

[3] https://github.com/silvanmelchior/RPi_Cam_Web_Interface

4.1 Abstract



An Apache web server is running on the raspberry pi. It hosts the camera interface, the code editor and the web interface.

If the user requests to compile the code, the apache web server initiates the gcc compiler to create an executable and returns its output. If the user now requests to run the code, apache starts the executable and redirects its output (*printf()*-calls) to the web service. The web server now interacts with the web service to get the program output and send keyboard inputs from the web interface. The user code itself spans a second thread (created by calling *init_drivers()*) which interacts with the web service to get e.g. keyboard inputs.

If the user code wants to control the hardware (e.g. set motor speed), it interacts with the HW service. This service controls all the AlphaBot Hardware via the drivers.

A display service is running, which controls the additional display and joystick. It is responsible for the settings menu.

A discovery service is running, which answers to broadcasts. It can be used to discover and control multiple PiRobots in the same network and was used during development mainly.

The communication between the individual parts is mostly realized with sockets.

4.2 Files

Most of the code is placed in */home/pi/PiRobo*, which is a clone of the official Git repo[4]. The web-based parts of the code are written in PHP, JS, HTML and CSS for the web-based parts, utilizing jQuery and Bootstrap, whereas the remaining part is mostly written in Python. This section explains the structure of the repo.

`copysrc`	files used during installation
`inc`	include-files for the user code, e.g. the the driver functions
`remote_ctrl`	if run on a separate web server, allows to discover and control multiple pirobos in the same network
`scoreboard`	if run on a separate web server, shows simple scoreboard as used during the competition
`services`	all services and drivers
`www`	web interface
`|--admin_ctrl`	admin web interface
`|--icecoder`	code editor

The remaining part of the repo are files used for installation, start-up and documentation.

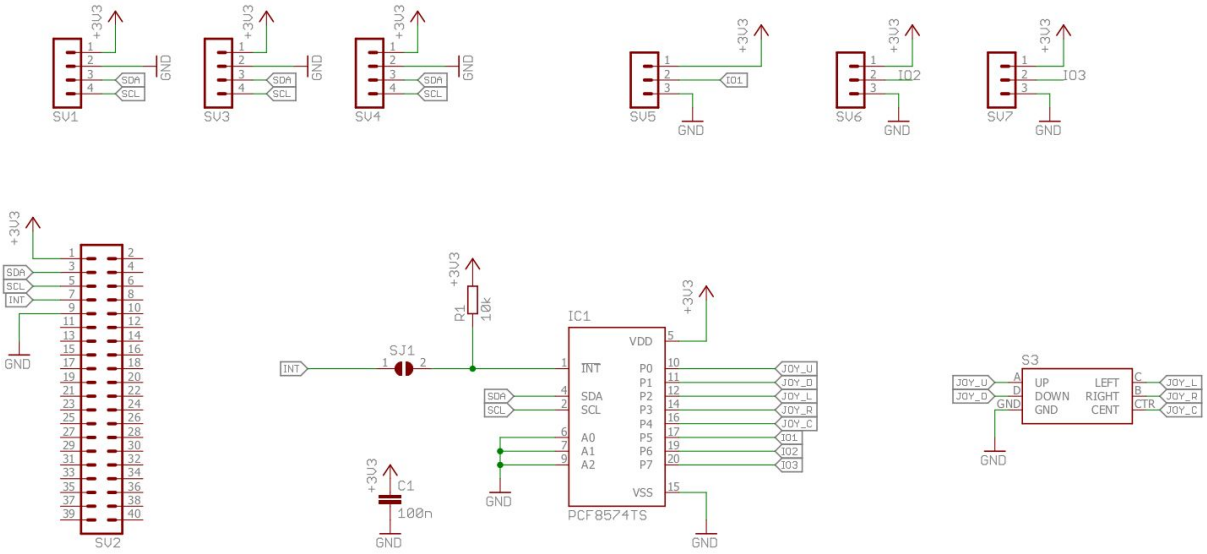
4.3 Update

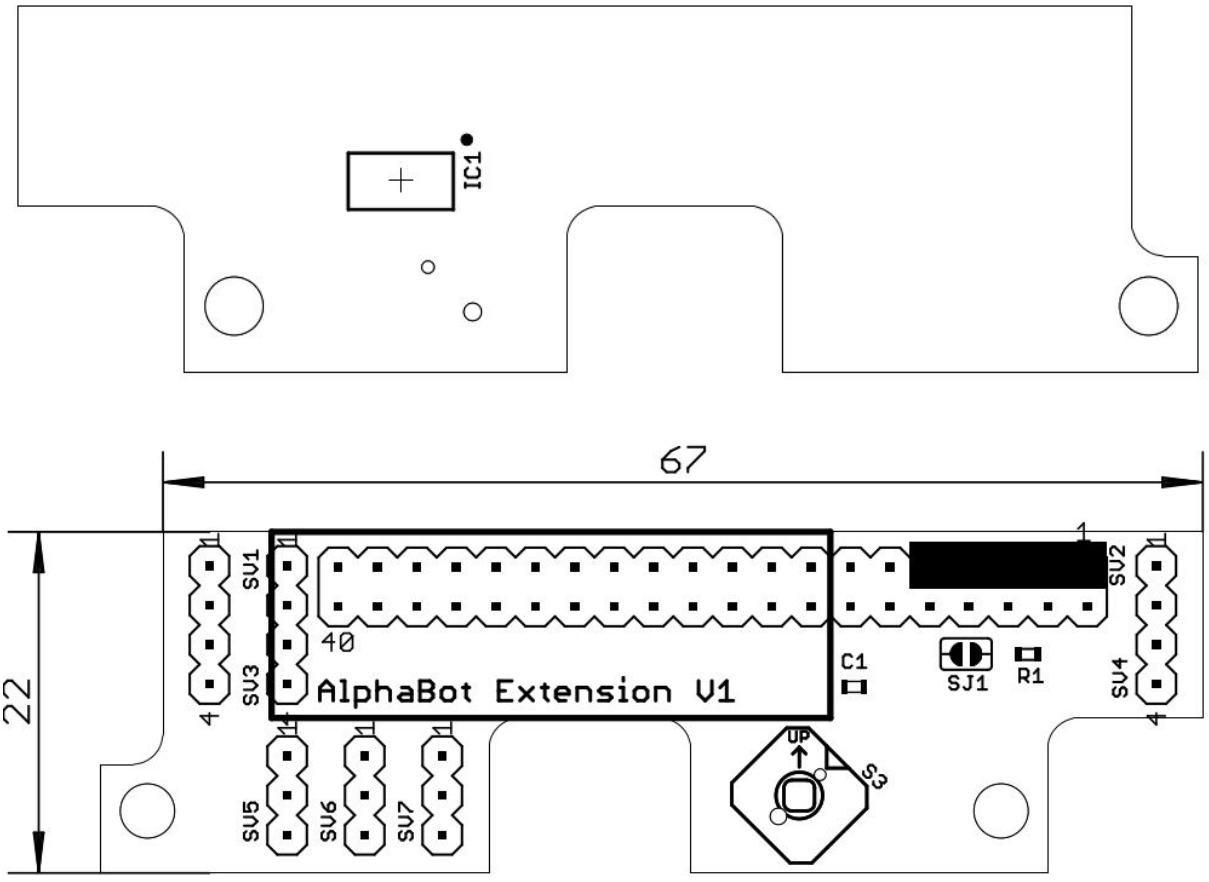
The robot updates itself automatically during start-up by pulling from the git-repository. If you want to disable this behaviour, remove the `git pull` command from *startup_l.sh*. If you make your own changes in the repository but still want to receive updated, commit them locally. If you are new to git, again first consulting a tutorial might be very helpful.

5. HW Documentation

PiRobo is based on AlphaBot[5], please see the website for documentation and schematics. The display documentation can be found here[6]. The PCB mounted on the AlphaBot containing the display and joystick is self-made:

[4] https://github.com/silvanmelchior/PiRobo
[5] https://www.waveshare.com/product/robotics/alphabot/kits/alphabot-pi3-b-plus.htm
[6] https://www.waveshare.com/product/modules/oleds-lcds/oled/0.91inch-oled-module.htm





6. Further

Never use PiRobo in a wifi where you don't fully trust all people having access to it. PiRobo is not designed safely, so it's very easy to access it if you are in the same wifi.