Remote_control project

The Remote_control project is a very simple example of remotely controlling motors with Raspberry PI using GPIO pins.

The remote control project tests if the direction and steering wheels motors of a Rastar remote control car, controlled by a Raspberry Pi Zero using GPIO pins, are working properly.

Getting Started

The Remote_control is a client-server application, coded in Pyhton, used to remotely control the movement of a **SmartMob car**. The movement is controlled by the user, through the keyboard arrows, using the client application installed on the PC. The server runs on the Raspberry Pi Zero of a SmartMob car and controls the direction and traction motors of the car according to the movement commands.

For this project, you should be familiar with the SmartMob platform, as well as with basic use of Raspberry Pi.

Before running the project you will need a that, the Raspberry Pi Zero of the SmartMob car is ready to use, with ssh enabled. You also need to have the PC and the Raspberry Pi Zero on the same wireless network.

Requirements

1. Hardware requirements

- **SmartMob car**: “Raystar remote control car”, at a scale 1:24, equipped with the SmartMob platform.
- **Regular AA batteries**: Three AA batteries of 1.5 V, used to to power up the motors.
- **Power bank**: a power bank used to power up the Raspberry Pi Zero.
- **Power-bank support**: an acrylic support to add on the top of the car to place a power bank.

2. Software requirements

- **Operating System**: last version available at Raspberry Pi website. The version used is “2018-11-13-raspbian-stretch.img” (updated and ready to use)
- **Programming language**: Python3.
- **Other tools**:

3. Raspberry Pi Configuration requirements

- **Remote access**: ssh enabled

4. Networking options

- **Raspberry PI Zero**: Raspberry PI and PC in the same wireless network.

Installation guide

I. SmartMob car assembly

Start by assembly the SmartMob car to work in integrated mode. Fo that:

1. Place three regular AA batteries into the batteries compartment.
2. Open the car and remove the coverage.
3. Place a power bank acrylic support on the top of the car and fixed the power bank to it.

II. Access the Raspberry PI

Now that you have assembled the car, it’s time to boot the system and remotely access the Raspberry Pi. For this, proceed as follows:

1. On the car, turn on the Switch S1, near the batteries compartment.
2. Connect the Raspberry Pi Zero to the power bank.
3. Check if there is connectivity between the Raspberry PI and the PC used for remote access. For that, open a terminal window at the PC and proceed as follows:

```
$ ping raspberrypi.local
```

4. Remotely access the Raspberry Pi by ssh . After digitising the command, upon request, digit “Yes” and enter the Raspberry Pi password.

```
$ ssh pi@raspberrypi.local
```

```
....upon request, digit "Yes"
.... upon request, enter the Raspberry Pi password.
```

III. Installing the software

1. At your PC, clone the git repository **remote-control**.

```
$ cd code-examples/sensors
$ git clone https://github.com/TeresaMVazao/Vehicula
rNetworks/sensors/remote-control remote-control
```

2. Check if the PC contains a complete version of software:

```
-> code-examples/sensors/remote-control/client-side
            +------------client_comm_services.py
            +------------remote_control.py
-> code-examples/sensors/remote-control/client-side
            +------------car.py
            +------------control_services.py
            +------------gpio_pins.txt
            +------------   server_comm_services.py
```

3. At your Raspberry Pi, clone the server-side component of the repository **remote-control/server-side** to the directory **/home/pi/code-example/sensors**.

```
$ cd /home/pi/code-examples/sensors
$ git clone https://github.com/TeresaMVazao/Vehicula
rNetworks/sensors/remote-control/server-side remote-
```

```
control
```

4. Check if the Raspberry Pi contains the server-side version of software:

```
-> /home/pi/code-examples/sensors/remote-control/cli
ent-side
             +------------car.py
             +------------control_services.py
             +------------gpio_pins.txt
             +------------   server_comm_services.py
```

Local test guide

I. System status check

Before running and testing the system, start by an initial check of the main components. For this, check if the:

1. PC contains a complete version of software: client.-side and server-side.
2. Code is prepared for local test. At the server-side directory, open the file control_services.py and change the flag to deny access to GPIO pins.

```
    RASPBERRY = False
```

II. Local execution and test

Now, the system is ready for test. You need to start by running the

server and only after it’s running you can start the client. For that:

1. Open a terminal window and run the server. As you do not enter any argument, the loopback address (127.0.0.1) will be used and the test must run in the local machine. The default port is 5003.

```
$ cd /code-examples/sensors/remote-control/server-si
de/
$ sudo python3 car.py <WLAN0_INET>
..Upon request, enter the password of your PC>
```

2. Check the appearance of a set of debug messages, describing the execution result of the initial phase. Note that, the GPIO information is related with the pins used and defined in the **gpio_pins.txt** file. The output is structured as follows:

```
127.0.0.1
Server started successfully
GPIO.setmode(GPIO.BOARD)
GPIO.setup(13, OUTPUT)
GPIO.output(13, GPIO.LOW)
...
gpio started successfully
```

3. Open a new terminal window and run the client. To use the same address of the server, do not enter any argument.

```
$ cd /code-examples/sensors/remote-control/client-si
de/
$ sudo python3 remote-control.py
```

```
    Upon request, enterthe password of your PC>
```

4. Check the instruction screen that appears in the client terminal.

```
To operate the system, please press:
    Arrow keys to select the car direction
    <TAB> to stop control
    <ESC> to exit keyboard operation
```

5. At the server terminal, check the new message:.

```
Connection from client
```

6. Start testing the system by digitising the keys and follow the output at the server side. Press <ESC> to terminate the program.
7. In case you detect any error, correct it in your local repository. Commit to a branch in github and issue a PullRequest to merge in the master.

Deployment

I. System status check

Now, the system is ready for deployment. Before, doing it, perform a preliminary check of the operational conditions.

1. Check if the SmartMob car is correctly assembled by detecting if:
    - There is no coverage

    - It has three AA 1.5V batteries and the switch S1 is turned on.
    - There is a power bank attached to an acrylic support and connected to the Raspberry Pi.
2. Check if the Raspberry PI is accessible by detecting if:
    - The boot is completed
    - There is an active remote access session between the Raspberry PI and your PC.
3. Check if the Raspberry Pi contains the server-side version of software ready for deployment. For this, open the file control_services.py and change the flag that grants access to GPIO pins.

```
RASPBERRY = True
```

II. Deployment and validation

Now, the system is ready for deployment. You need to start by running the server and only after it’s running you can start the client. For that:

1. Start by finding the IP address of the the Raspberry Pi, as this address will be used to launch the server. For this, at the Raspberry Pi proceed as follows:

```
$ if config
... IPv4 address of the interface (WLAN0_INET)
```

2. At the Raspberry Pi, run the server, using the IPv4 address previously identified. If you do not enter the address, the loopback address (127.0.0.1) will be used and the test must run

in the local machine.

```
$ cd /code-examples/sensors/remote-control/server-si
de/
$ sudo python3 car.py <WLAN0_INET>
```

3. At the PC, run the client, using the IPv4 address of the server previously identified. If you do not enter the address, the loopback address (127.0.0.1) will be used and the test must run in the local machine.

```
$ cd /code-examples/sensors/remote-control/cient-sid
e/
$ sudo python3 remote-control.py <WLAN0_INET>
```

4. Check the instruction screen that appears in the client terminal. At the PC, start controlling the SmartMob car movement and look at the outcome. Press <TAB> to stop the car and <ESC> to terminate.

Authors

- **Teresa Vazao** - *Initial work* - SmartMob@Tecnico

See also the list of contributors who participated in this project.