

Mobile Surveillance System



Students:

Ifrim Rareș

Ungureanu Andreea

Contents

- Introduction
- The conceptual Scheme
- Block Diagram
- Circuit Connecting
- Introduction to Sockets, Server, Client
- Streamer setup
- I2c bus interface setup

Introduction

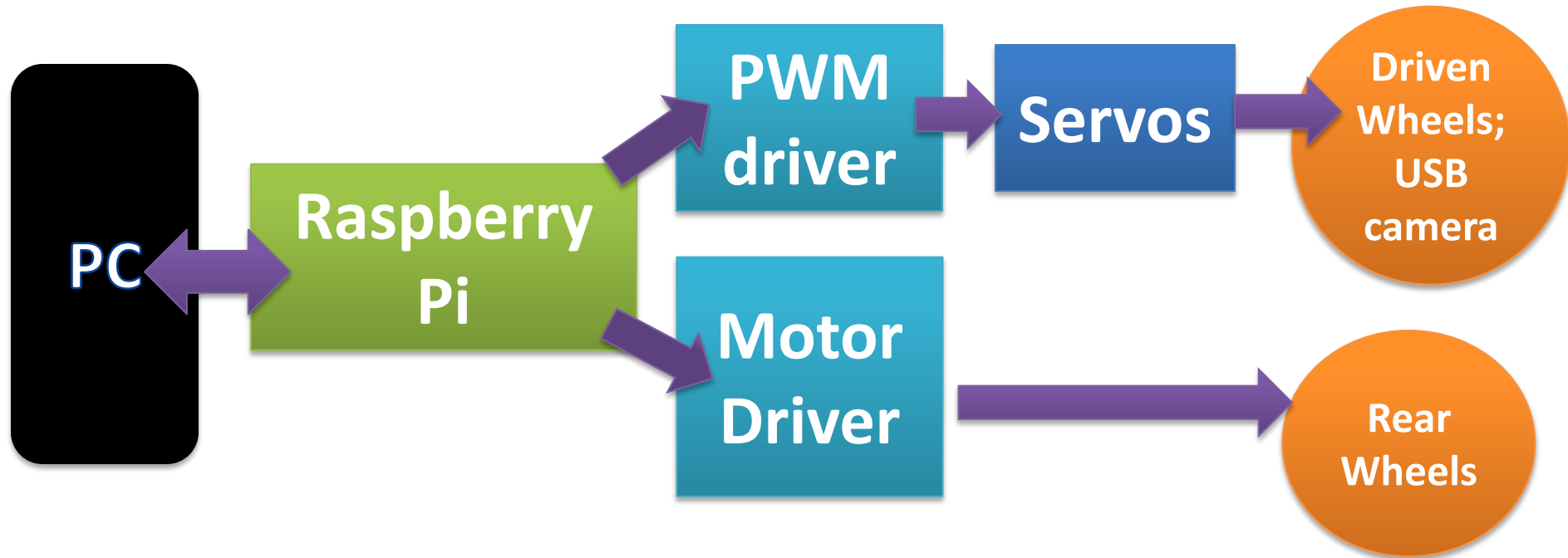
The SunFounder Smart Video Car Kit is composed of:

- Raspberry Pi 3b+, step-down DC-DC converter module, USB camera (alternative: Camera Module for Rpi), DC motor driver, PWM driver (alternative: another microcontroller) , Micro Servos, DC motors, rechargeable Li-ion battery.

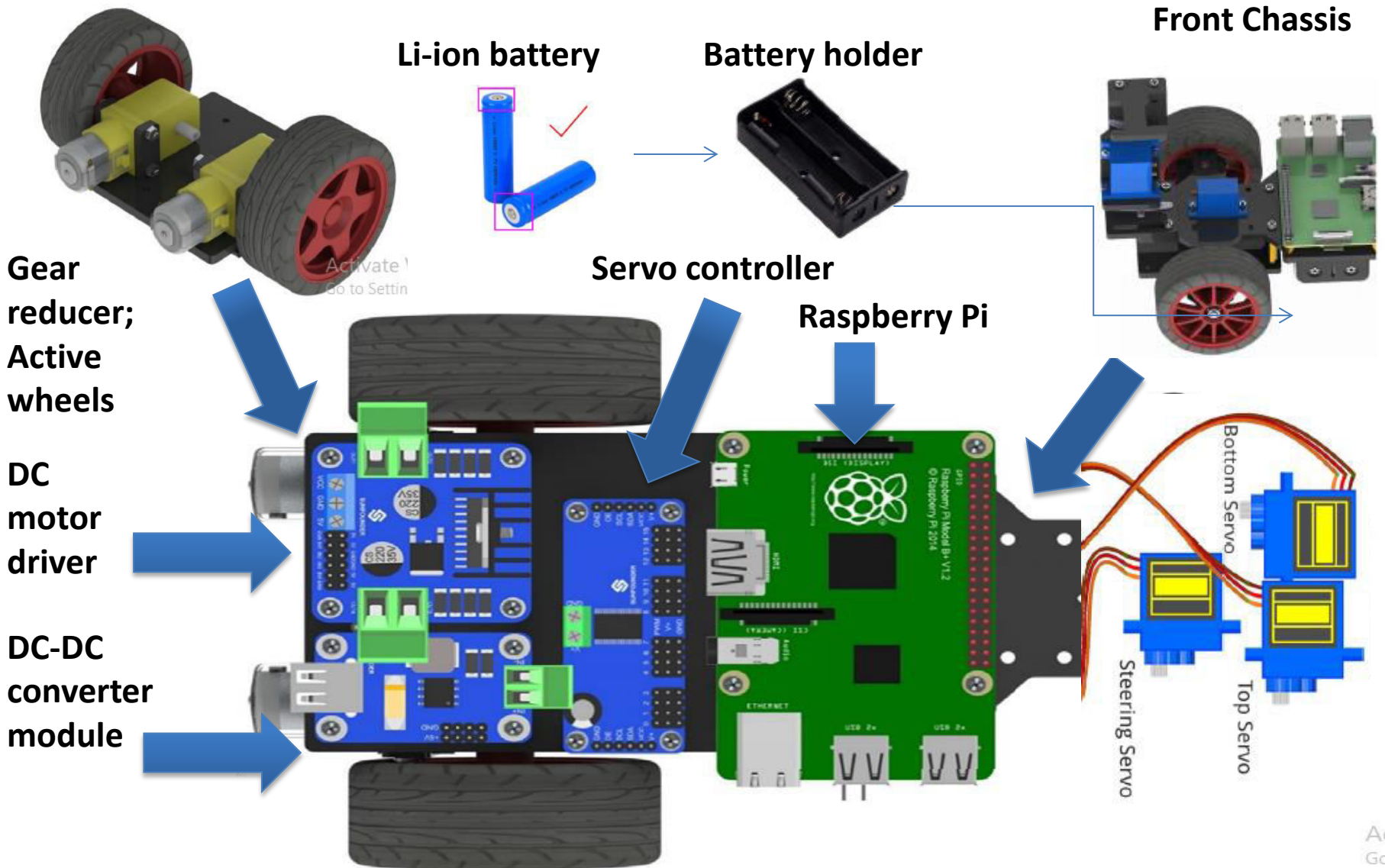
From the perspective of software, the smart car is of client/server (C/S) structure.

1. The TCP server program is run on Raspberry Pi for direct control of the car.
2. The video data are acquired and delivered via the open source software MJPG-streamer in a real-time manner.(You may view the video by web browser on any device).
3. The TCP client program is run on PC to send the control command.
(Alternative: the car can also be controlled by a phone application: PiCar Control)
4. Both the client and server programs are developed in Python.

The conceptual scheme

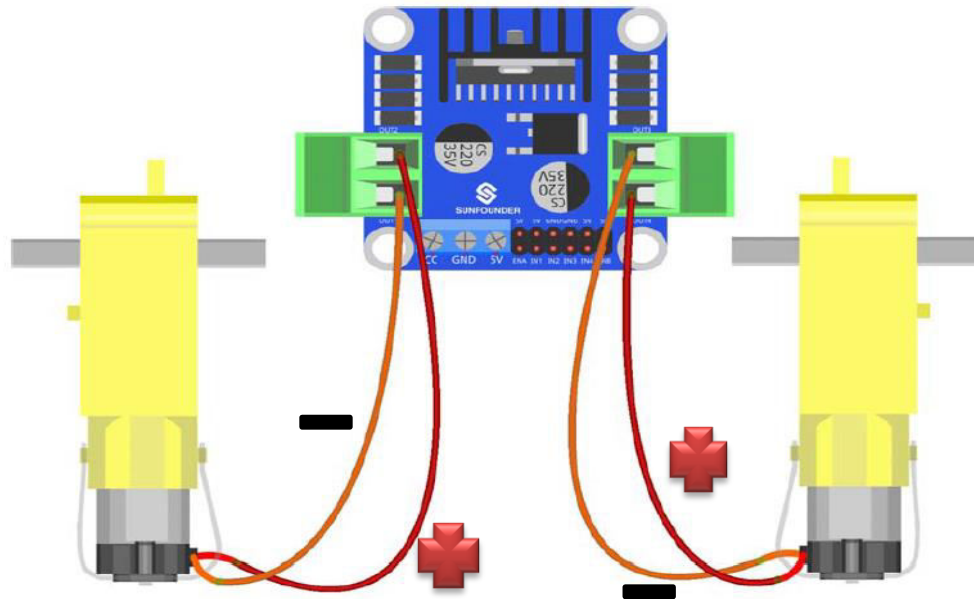


Block Diagram



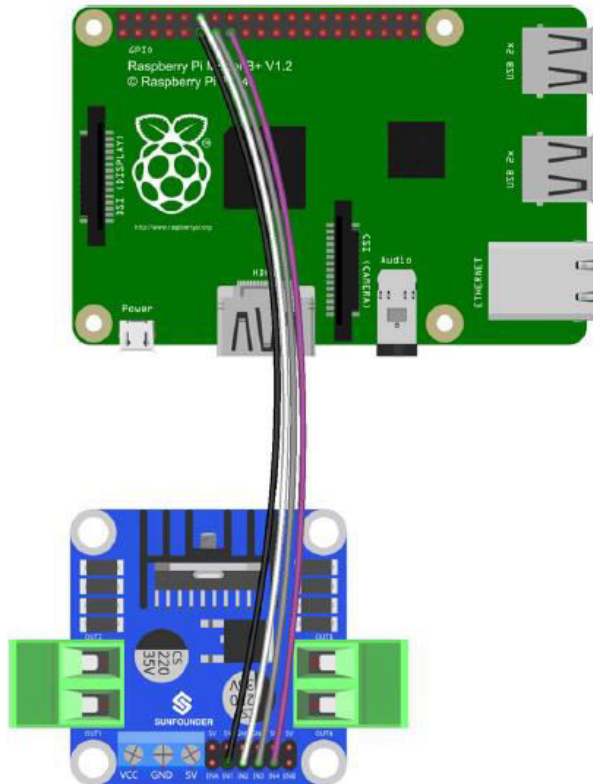
Circuit Connecting, steps:

- The two DC motors connected with the motor driver.



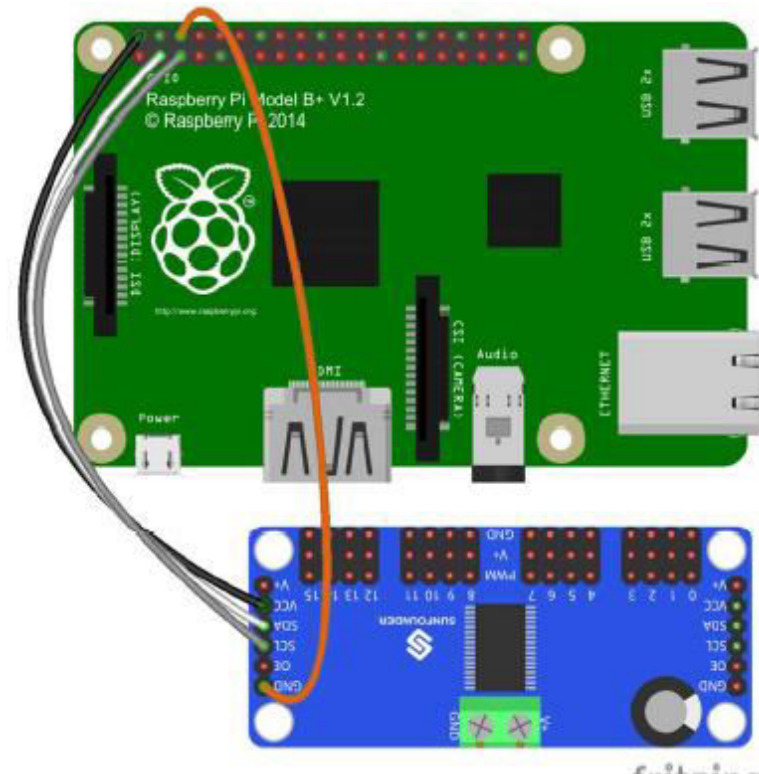
- The motor driver connected with the Raspberry Pi GPIO port based on the following table.

Raspberry Pi GPIO Port	DC Motor Driver
Pin11	IN1
Pin12	IN2
Pin13	IN3
Pin15	IN4 </td

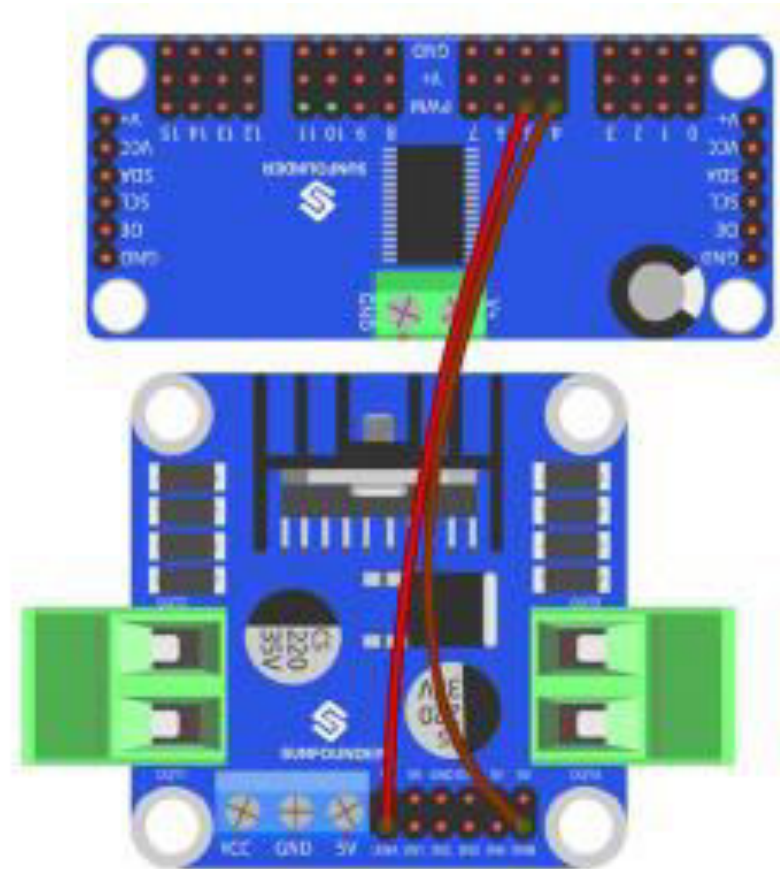
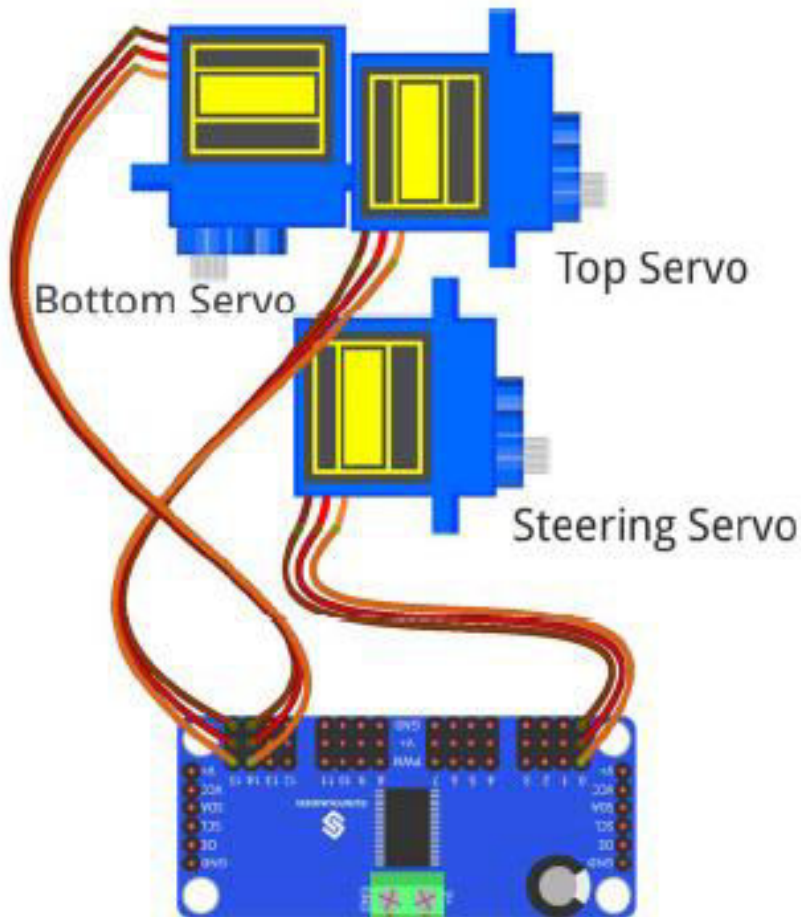


- The servo controller connected with the Raspberry Pi GPIO port as follows:

Raspberry Pi GPIO Port	Servo Controller
Pin 2 (5V)	VCC
Pin 3 (SDA)	SDA
Pin 5 (SCL)	SCL
GND	GND

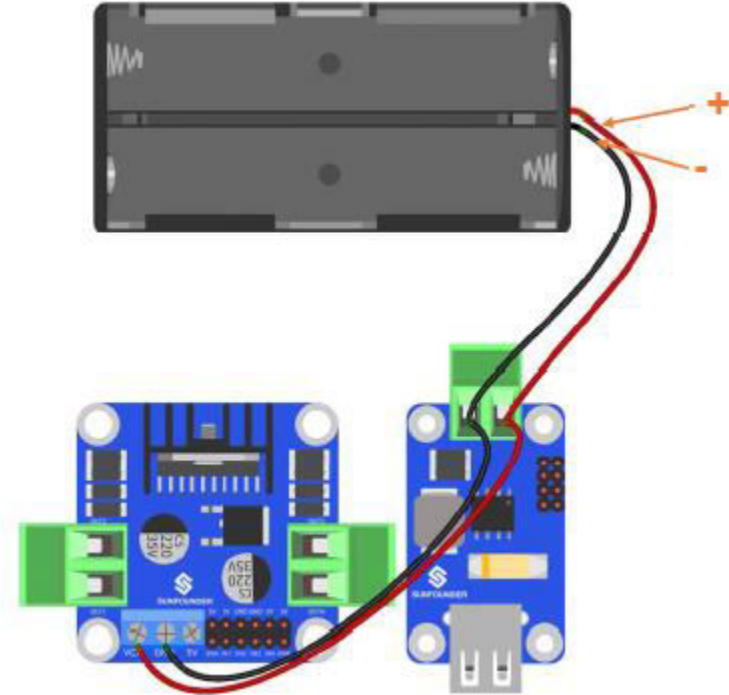
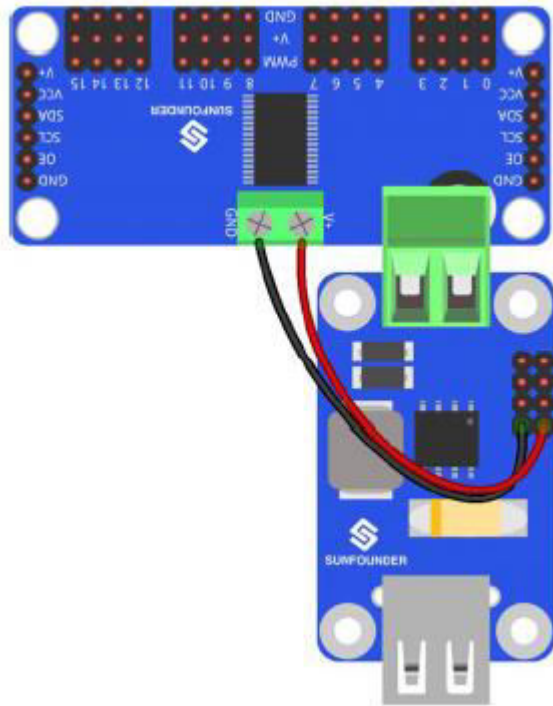


- The servo that controls the car's direction hooked up to CH0 of the servo controller, and the two servos that control the view of the camera to CH14 and CH15 respectively, as shown below:

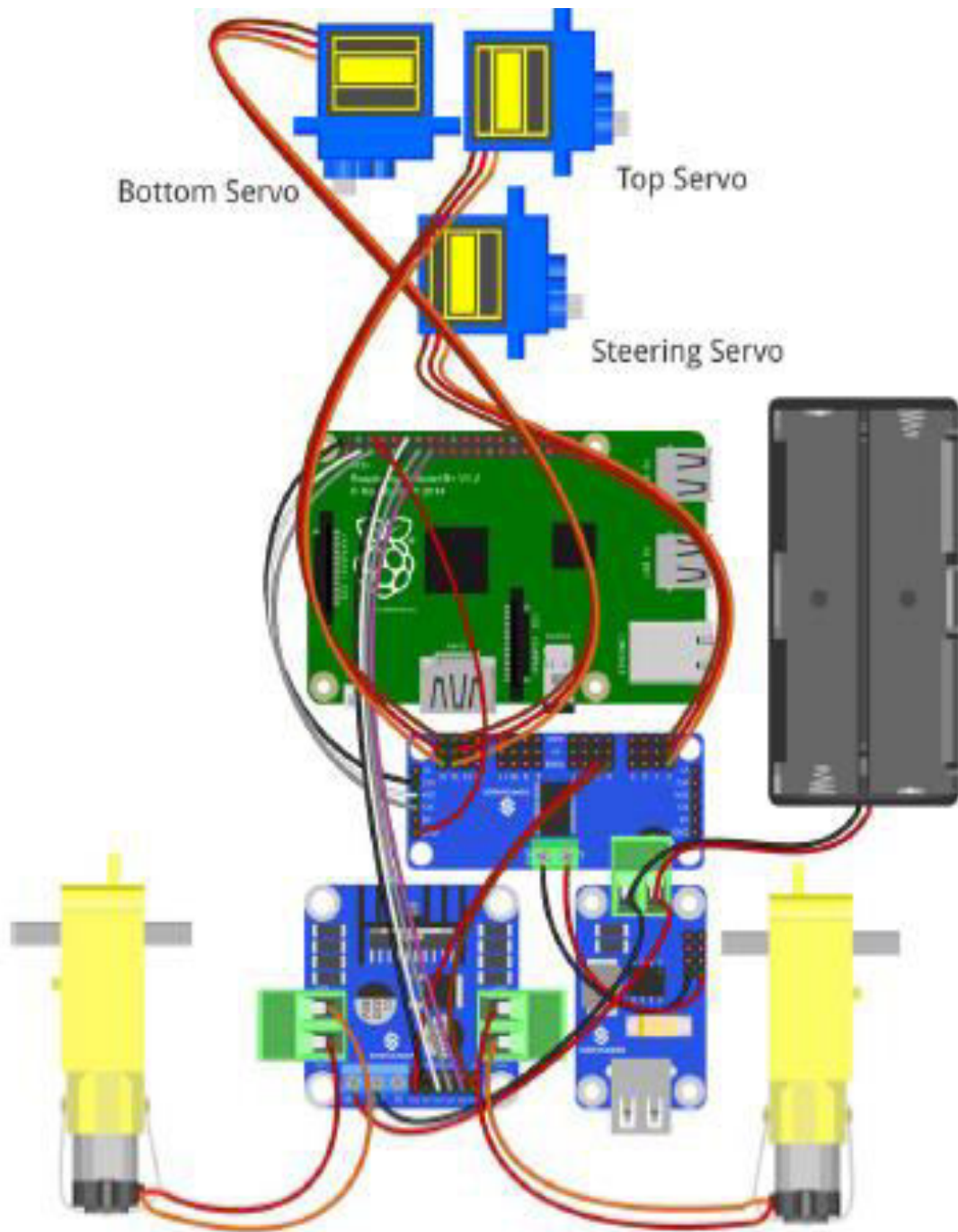


- The motor driver connected with the servo controller (CH5 -> ENA; CH4->ENB).

- The servo controller connected with the step-down DC-DC converter module.



- The battery holder connected with the step-down DC-DC converter module and the DC motor driver.



The whole picture of wiring should be like this:

Introduction of Socket

The C/S-structure program of the SunFounder Raspberry Pi-based Smart Car is written based on the socket module of the Python language. Socket wraps and applies the TCP/IP and is used to describe IP address and port. Also it is a network data structure for computer. The socket module should be created before the communication of network applications. If the socket can be said to be the plug of a telephone, which is the lowest layer of communication, then the combination of IP address and ports can be said to be that of area code and phone numbers. Only having the hardware for a phone call making is not enough. You still need to know whom and where to call. An Internet address is composed of the essential IP address and port for network communication.

1. Server

Here we provide a pseudocode which creates a universal TCP server for explanation. Note that this is just one of the methods for server design. After you have a good knowledge about it, you can alter the pseudocode as you want:

```
s = socket( )           # Create a socket for the server.
s.bind( )              # Bind the address to the socket.
s.listen( )           # Listen to the connection.
inf_loop:             # Indefinite Loop of the server.
    c = s.accept( )    # Accept the connection from the client.
comm_loop:           # Communication Loop.
    c.recv( )/c.send( ) # Dialog (receiving or sending data)
c.close( )           # Close the socket of the client.
s.close( )           # Close the socket of the server (optional).
```

All kinds of socket can be created via the function `socket.socket()` and then bound with IP address and port by the function `bind()`. Since TCP is a connection-oriented communication system, some settings need to be completed before the TCP server starts operation. The TCP server must "listen" to connections from the client.

After the settings are done, the server will enter an indefinite loop. A simple, like single-thread, server will call the function `accept()` to wait for the coming connection. By default, the function `accept()` is a blocking one, which means it is suspended before the connection comes. Once a connection is received, the function `accept()` returns a separate client socket for the subsequent communication. After the temporary socket is created, communication begins. Both the server and client use the new socket for data sending and receiving. The communication does not end until either end closes the connection or sends a null character string.

2. Client

It is easier to create a TCP client than to do a server. Take the following pseudocode:

```
c = socket( )           # Create a client socket.
c.connect( )           # Try to connect a server.
comm_loop:             # Communication Loop.
    c.send( )/c.recv( ) # Dialog (sending out and receiving data)
c.close( )              # Close the client socket.
```

As mentioned above, all sockets are created via the function `socket.socket()`. Then, the function `connect()` can be called to connect the server. After the connection is built, the dialog between the client and the server is enabled. When the dialog ends, the client can close the socket and the connection.

A. Run TCP server (Operation on Raspberry Pi)

Now a terminal's already open for remote login to the Raspberry Pi and run the `mjpg-streamer`, and you need to keep it RUNNING. Open one more to log into the Raspberry Pi to run `tcp_server`.

Go to the directory with `cd`.

```
cd ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server
```

Then run `tcp_server.py`:

```
sudo python tcp_server.py
```

The server program on the Raspberry Pi will be running and waiting for the client to connect to the Raspberry Pi.

```
pi@raspberrypi ~/Sunfounder_Smart_Video_Car_Kit_for_Raspberr  
yPi/server $ sudo python tcp_server.py  
Waiting for connection...  
█
```

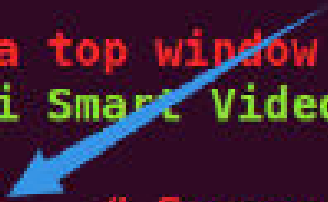
Open another terminal in Linux (not via ssh on your Pi). Find the sketch downloaded and edit `client/cali_client.py`:

```
cd client/  
sudo nano cali_client.py
```

```
pi@cavon:~$ cd Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/client/  
pi@cavon:~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/client$ sudo nano cali_client.py
```

Find the variable of `HOST`:

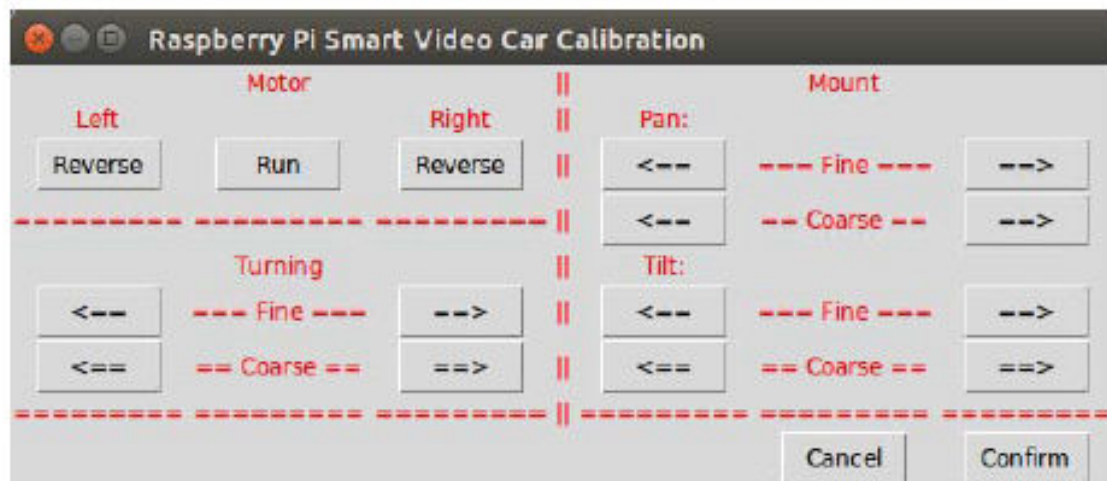
```
GNU nano 2.2.6      File: cali_client.py  
  
#!/usr/bin/env python  
# -*- coding: utf-8 -*-  
from Tkinter import *  
from socket import *      # Import necessary modules  
import os  
  
top = Tk()      # Create a top window  
top.title('Raspberry Pi Smart Video Car Calibration')  
  
HOST = '192.168.0.133'      # Server(Raspberry Pi) IP address  
PORT = 21567  
BUFSIZ = 1024      # buffer size  
ADDR = (HOST, PORT)
```



Run `cali_client.py`:

```
sudo python cali_client.py
```

No matter what system your computer is running on, Linux or Windows, when you run `cali_client.py`, a window Raspberry Pi Smart Video Car Calibration will pop up:



In the terminal remotely connected with the Raspberry Pi, the IP address of the PC will be printed.

```
pi@raspberrypi ~/Sunfounder_Smart_Video_Car_Kit_for_Raspber  
yPi/server $ sudo python cali_server.py  
offset_x = 0  
offset_y = 0  
offset = 0  
turning0 = True  
turning1 = True  
Waiting for connection...  
...connected from : ('192.168.0.124', 54220)
```

Then you can start calibrating. Before that, **take out your package box and place it vertically with the side face to the table**. Put the car inside the box and keep it balanced. The purpose is to keep the wheels of the car off the table.

By default, the front wheels should be directly pointed towards the front; the camera on the tilt servo should be face up no matter what directions the pan servo is pointed at.

```
pi@cavon:~$ cd Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi
/client/
pi@cavon:~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/cli
ent$ sudo nano client_App.py
```

Similar to changes in calibration, change the IP address in client program in your PC:

```
GNU nano 2.2.6 File: client App.py

#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import *
from socket import * # Import necessary modules

ctrl_cmd = ['forward', 'backward', 'left', 'right', 'stop', '$

top = Tk() # Create a top window
top.title('Sunfounder Raspberry Pi Smart Video Car')

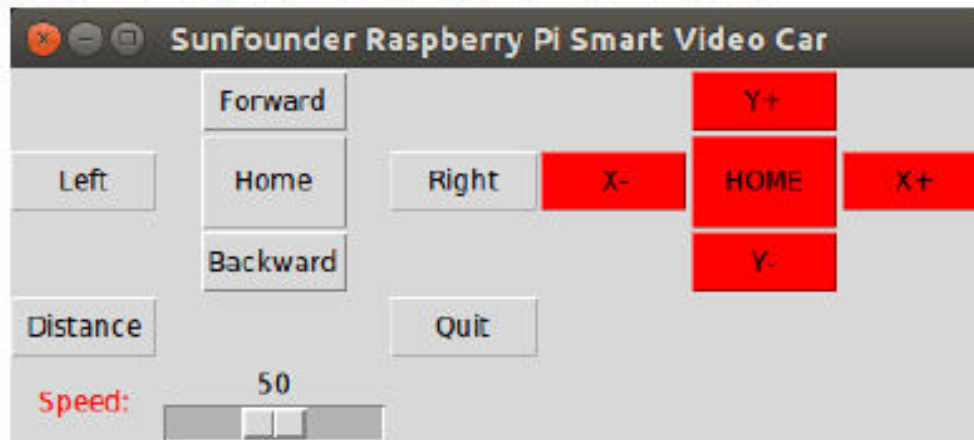
HOST = '192.168.0.133' # Server(Raspberry Pi) IP address
PORT = 21567
```

After the alteration, press Ctrl + O and save and Ctrl + X to exit.

Then run the client program:

```
sudo python client_App.py
```

No matter what system your computer is running on, Linux or Windows, when you run the client_App.py, the following window will appear on your screen:



Operation on Raspberry Pi

Keep Run the mjpg-streamer

Run the program:

```
sudo sh start.sh
```

Then the video data acquisition will start, like this:

```
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan Reset
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Tilt Reset
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan/tilt Reset
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Focus (absolute)
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Mode
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Frequency
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Disable video processing
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Raw bits per pixel
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
o: www-folder-path...: ./www/
o: HTTP TCP port.....: 8080
o: username:password.: disabled
o: commands.....: enabled
```

Type in the following address (replace `192.168.0.126` with your Raspberry Pi IP address) at the address bar of your browser (Firefox is recommended): (operation on PC)

<http://192.168.0.126:8080/stream.html>

Press **Enter** and you will see the view captured by the camera displayed on the screen in a real-time manner.

MJPEG-Streamer Demo Pages

a resource friendly streaming application

Home

Static

Stream

Java

Javascript

VideoLAN

Control

Version info:

v0.1 (Oct 22, 2007)

Display the stream

Hints

This example shows a stream. It works with a few browsers like Firefox for example. To see a simple example click [here](#). You may have to reload this page by pressing F5 one or more times.

Source snippet

```

```



F. Setup I2C Port

Run the command to open **Raspberry Pi Software Configuration Tool (raspi-config)**

```
sudo raspi-config
```

Enable I2C:

Select Interfacing Options => I2C => <Yes> => <Ok> => <Yes>

Select <Finish>. Close the window.

If a message of rebooting appears, click <No>. Before reboot, we still need to complete some configurations.

Bibliography

- ->Hardware components:
<https://www.sunfounder.com/rpi-car.html>
- ->Source code:
https://github.com/sunfounder/SunFounder_PiCar-V
- ->I2C bus interface:
[file:///C:/Users/Andreew/Desktop/Doc/Sunfounder Smart Video Car Kit for RaspberryPi-master/Sunfounder Smart Video Car Kit for RaspberryPi-master/datasheet/Basics%20of%20the%20I2C%20Communication%20Protocol.html](file:///C:/Users/Andreew/Desktop/Doc/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi-master/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi-master/datasheet/Basics%20of%20the%20I2C%20Communication%20Protocol.html)
- -> Streamer : https://www.acmesystems.it/ariag25_wirings