**Contents**

Components List ........................................................................................................................................1

Acrylic Plates..........................................................................................................................................1

Servo Accessories .................................................................................................................................2

Mechanical Fasteners.........................................................................................................................2

Wires .........................................................................................................................................................3

PCB ...........................................................................................................................................................4

Other Components..............................................................................................................................5

Tools ..........................................................................................................................................................6

Introduction .................................................................................................................................................7

Building the Car..........................................................................................................................................7

Fixing Rear Wheels................................................................................................................................7

Upper Plate ......................................................................................................................................... 10

Battery Holder..................................................................................................................................... 11

Rear Wheels (Driving) ....................................................................................................................... 13

TF Card Guard.................................................................................................................................... 15

Front Half Chassis ............................................................................................................................... 16

Front Wheels........................................................................................................................................ 17

Steering Part........................................................................................................................................ 18

PCB Assembly ..................................................................................................................................... 20

Circuits Building .................................................................................................................................. 21

Software Installation.......................................................................................................................... 23

1. Log into Raspberry Pi .............................................................................................................. 23

2. Get Source Code .................................................................................................................... 23

3. Go to the Code Directory..................................................................................................... 24

4. Install the Environment ........................................................................................................... 24

Adjust the Servo to 90 Degrees ..................................................................................................... 25

Build the Rest of the Car .................................................................................................................. 26

Configuration...................................................................................................................................... 28

Appendix .................................................................................................................................................. 33

Appendix 1: Modules ....................................................................................................................... 33

Robot HATS ..................................................................................................................................... 33

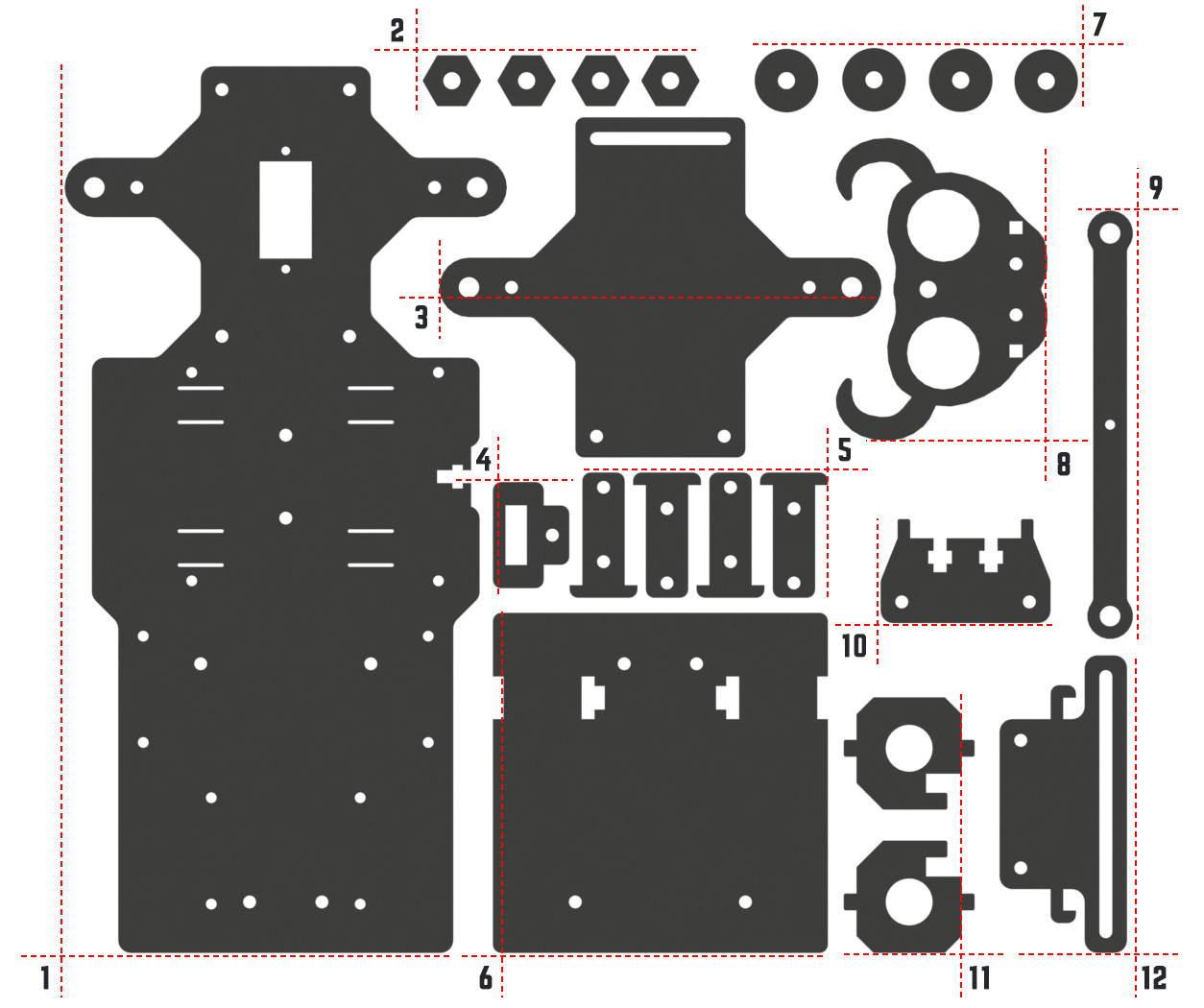
PCA9865.......................................................................................................................................... 34

TB6612 .............................................................................................................................................. 35

SunFounder SF0180 Servo ..............................................................................................................35

**Components List**

**Acrylic Plates**



1. Upper Plate x 1

2. Hex Front Wheel Fixing Plate x 4

3. Front Half Chassis x 1

4. TF Card Guard x 1

5. Motor Support x 4

6. Back Half Chassis x 1

7. Bearing Shield x 4

8. Ultrasonic Connector x 1

9. Steering Linkage x 1

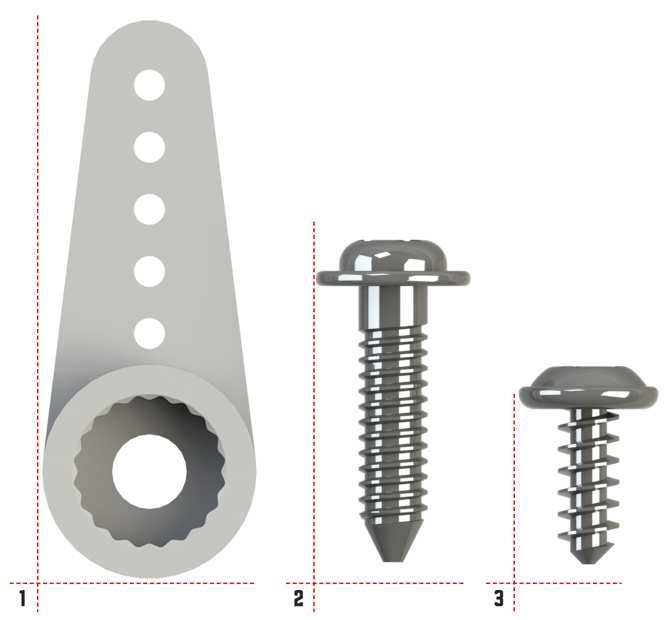
10. Ultrasonic Support x 1

11. Steering Connector x 2

12. Sensor Connector x 1

**Servo Accessories**

The following three parts will be used in the servo package:



1. Rocker Arm

2. Rocker Arm Screw

3. Rocker Arm Fixing Screw

**Mechanical Fasteners**

|  |  |  |
| --- | --- | --- |
| **Name** | **Component** | **Qty.** |
| M2x8 Screw |  | 2 |
| M2.5x6 Screw |  | 12 |
| M3x8 Screw |  | 8 |
| M3x8  Countersunk  Screw |  | 2 |
| M3x10 Screw |  | 9 |
| M3x30 Screw |  | 4 |

|  |  |  |
| --- | --- | --- |
| M4x25 Screw |  | 2 |
| M2 Nut |  | 2 |
| M2.5 Nut |  | 12 |
| M3 Nut |  | 23 |
| M4 Self-locking  Nut |  | 2 |
| M2.5x8 Copper  Standoff |  | 16 |
| M3x25 Copper  Standoff |  | 8 |
| 4x11x4 F694ZZ Flange Bearing |  | 2 |

**Wires**

|  |  |  |
| --- | --- | --- |
| 100mm HX2.54  5-Pin Jumper  Wire |  | 1 |
| 50mm HX-2.54 4- Pin Jumper Wire |  | 1 |
| 50mm HX-2.54 2- Pin Jumper Wire |  | 1 |
| 100mm HX-2.54  2-Pin Jumper  Wire |  | 1 |
| 200mm HX2.54  5-Pin Jumper  Wire |  | 1 |
| 200mm HX-2.54  4-Pin Jumper  Wire |  | 1 |

3

200mm HX2.54



3-Pin Jumper 1

Wire

**PCB**

|  |  |  |
| --- | --- | --- |
| Robot HATS |  | 1 |
| PCA9685 PWM Driver |  | 1 |
| TB6612 Motor  Driver |  | 1 |
| 5-CH Line  Follower Module |  | 1 |

4

**Other Components**

|  |  |  |
| --- | --- | --- |
| 2x18650 Battery  Holder |  | 1 |
| DC Gear Motor |  | 2 |
| SunFounder  SF0180 Servo |  | 1 |
| Rear Wheel |  | 2 |

5

|  |  |  |
| --- | --- | --- |
| Front Wheel |  | 2 |
| USB Wi-Fi Adapter |  | 1 |
| Ribbon (30cm) |  | 1 |

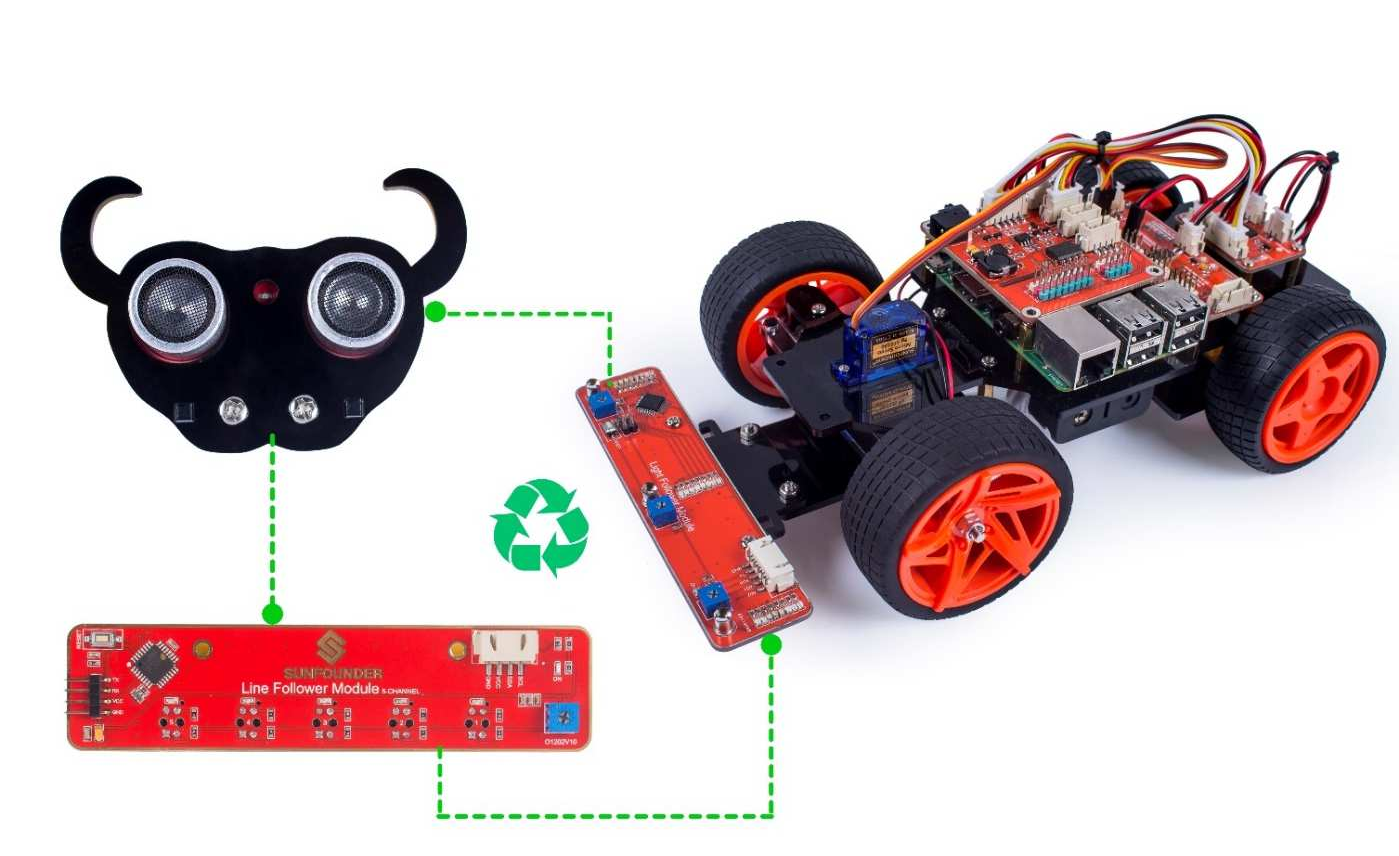
**Tools**

|  |  |  |
| --- | --- | --- |
| Cross Screwdriver |  | 1 |
| Cross Socket  Wrench |  | 1 |
| M2.5/M4 Small  Wrench |  | 1 |
| M2/M3 Small  Wrench |  | 1 |

6

**Introduction**

The is a **SMART SENSOR** car robot based on Raspberry Pi. With these modules, this smart car is capable of some simple automatic actions. Thus, you can learn some basics of programming in Python to control the car with these sensors. Let’s start with building this smart car!



**Building the Car**

Extremely excited when opening the box and checking so many components? Keep your patience and take it easy. Please note that some details in the following steps need **CAREFUL** observation. You should double-check your work based on the figures in the manual after finishing each step. Don’t worry! Kindly reminders will be given in some particular steps. Just follow the tutorial step by step. Okay, with no further ado, now let’s start!

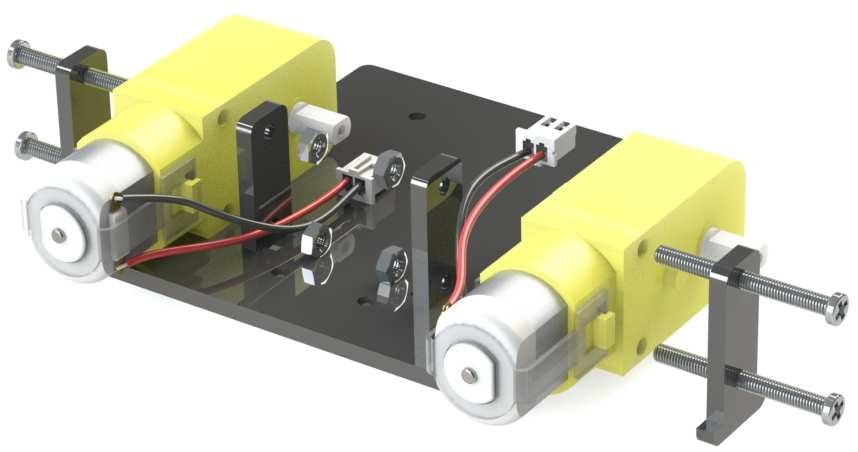
**Fixing Rear Wheels**

Assemble the **Motor Support Plate** into the **Back Half Chassis** as shown below.

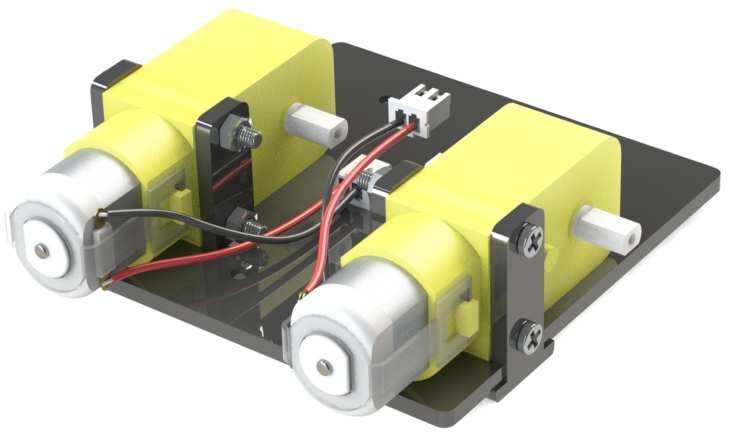


Assemble the two motors with four **M3x30 screws** and **M3 nuts**. Pay attention to place the

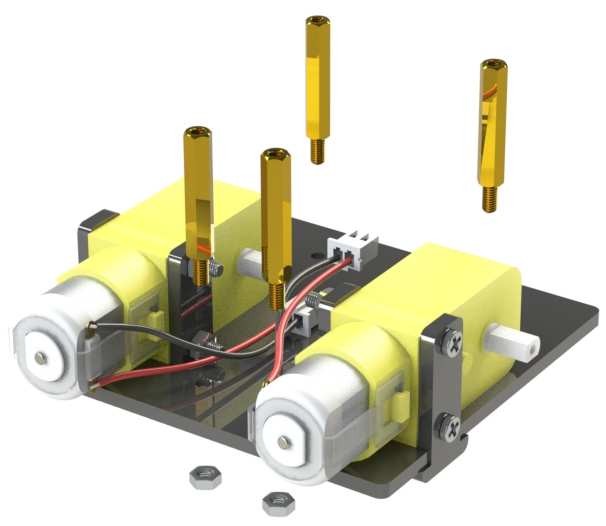
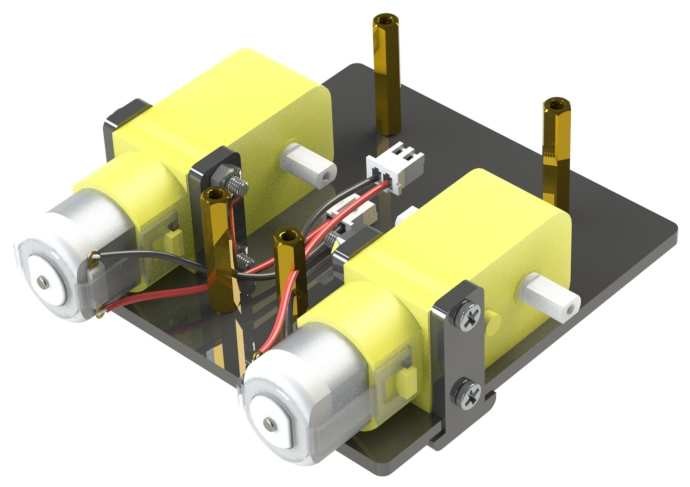
motors with wires inward, providing convenience for connecting the circuit.



Place wires inwards



Insert four **M3x25 copper standoffs** through the acrylic plate into four **M3 nuts** as shown below:



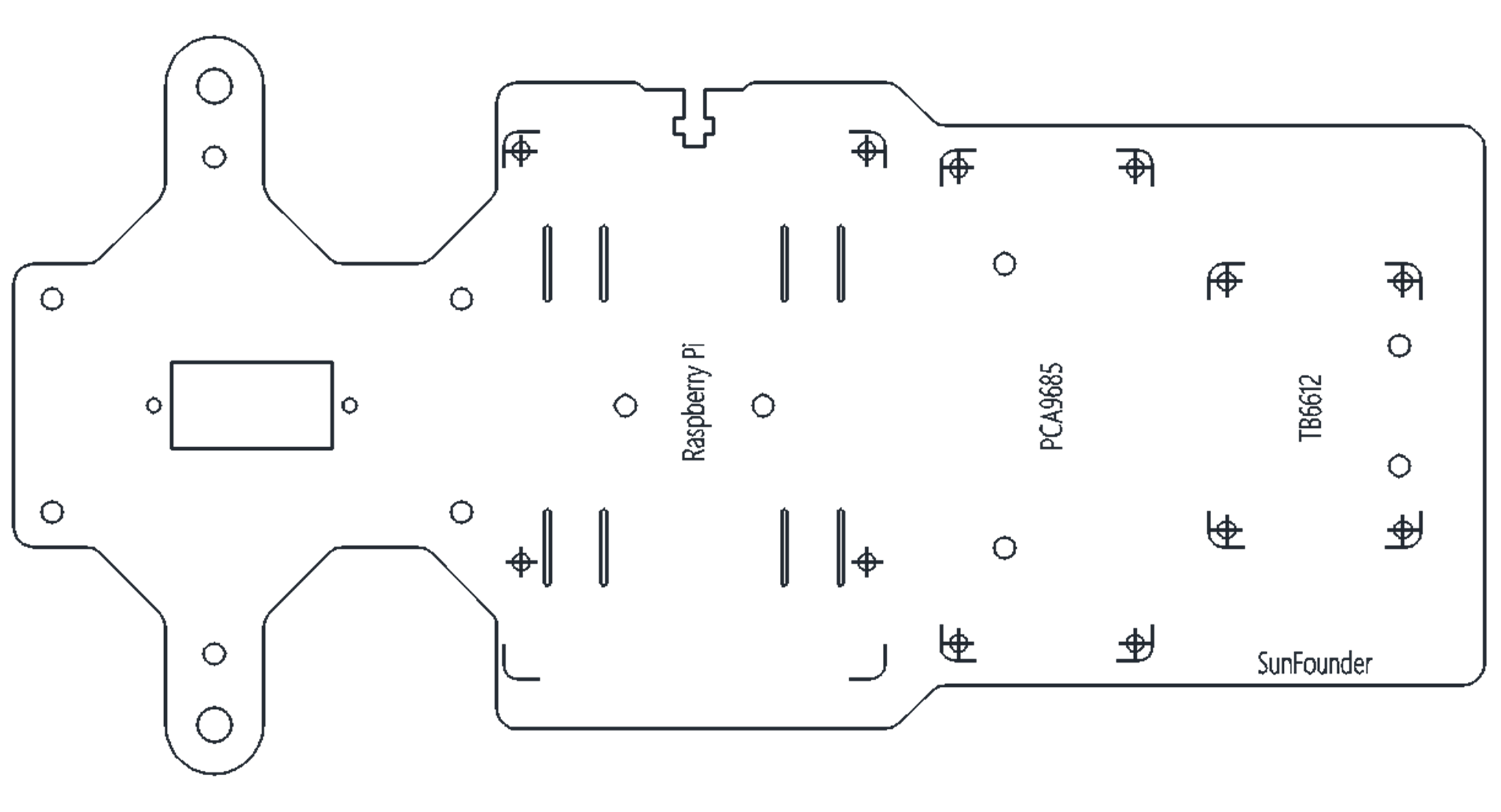
So now this part is completed. You can put it aside for now.

**Upper Plate**

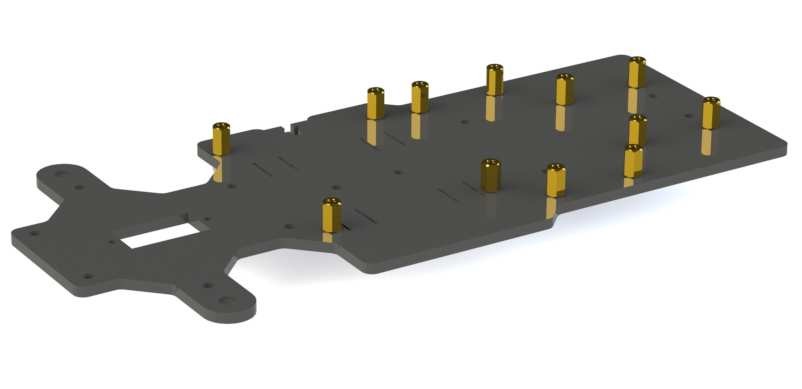
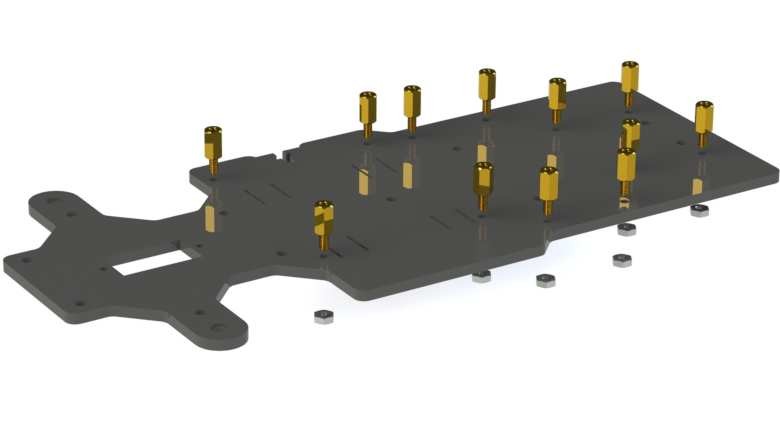
Mount the **M2.5x8 copper standoffs** and **M2.5 nuts** into the **upper plate** first. There are three

PCBs to be installed onto the plate and four copper standoffs are needed for each. So here

12 holes should be used, marked with cross as shown below:

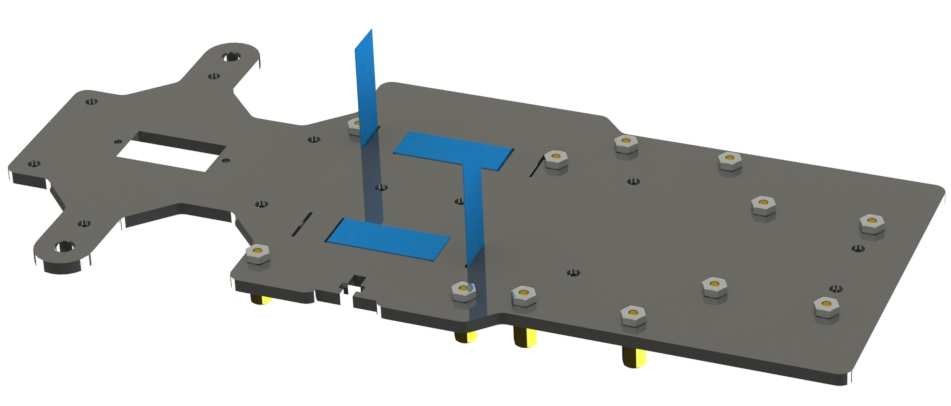


Assemble the **M2.5x8 copper standoffs** and **M2.5 nuts** as shown below. Pay attention that the side the logo carved should face up.

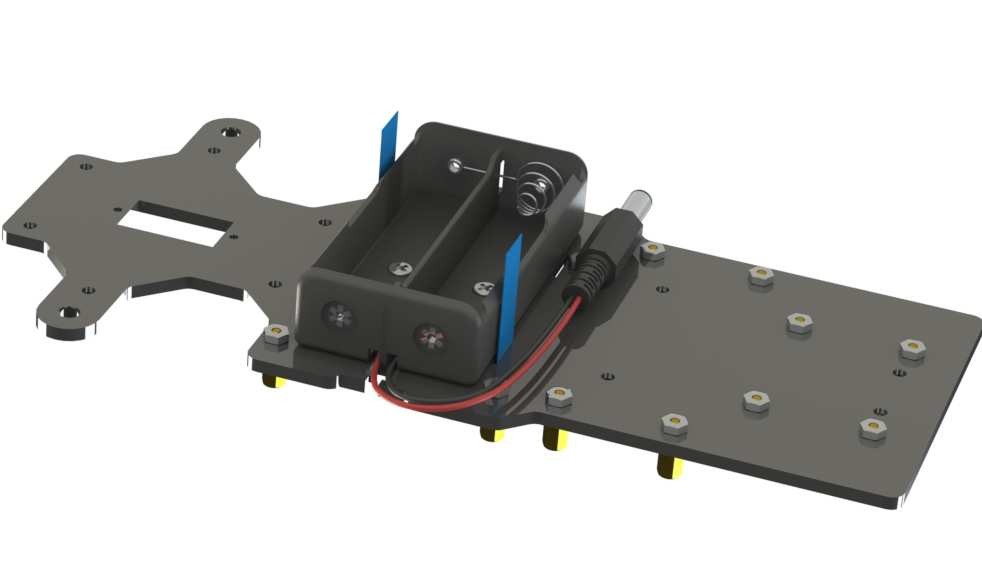
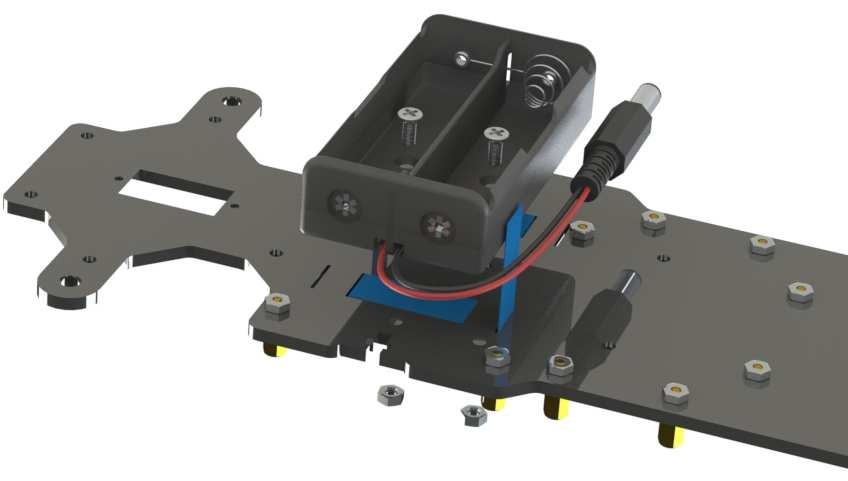


**Battery Holder**

Turn the Upper Plate upside down. Cut the **ribbon** into two halves. Thread them through the holes on the plate. Pay attention to the direction and leave one end longer out of the plate for each to remove the battery easily later.

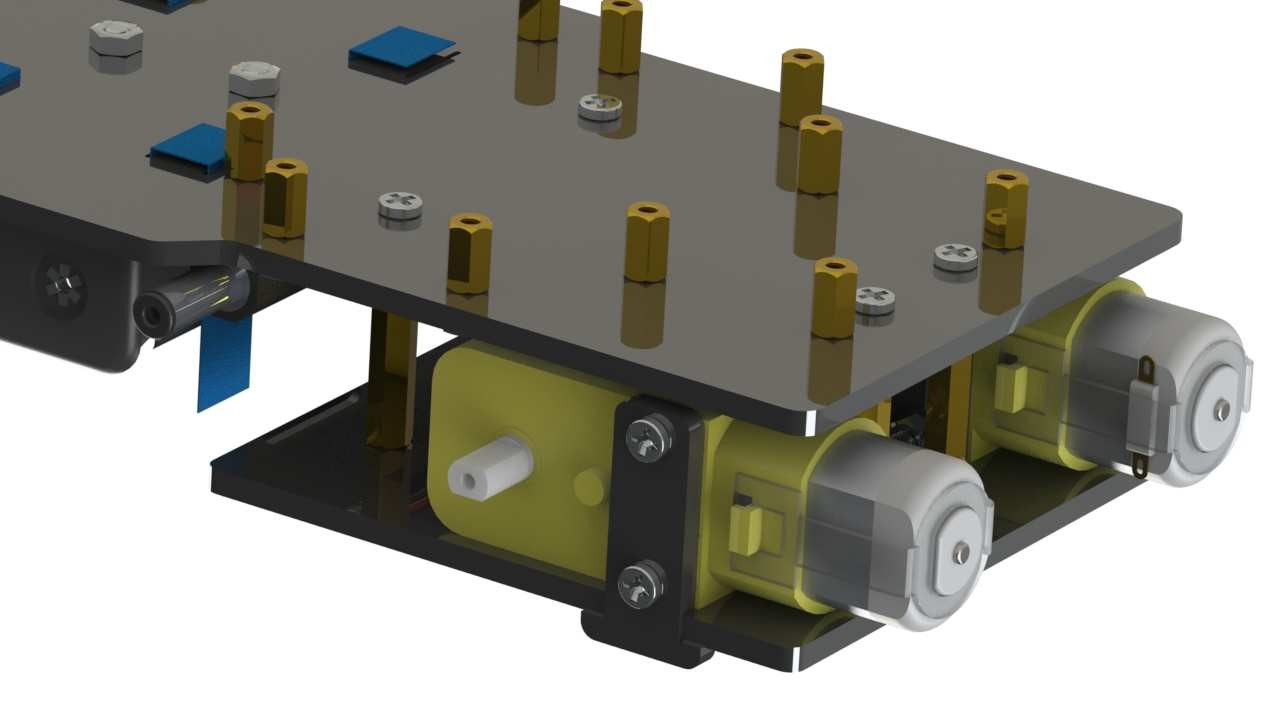
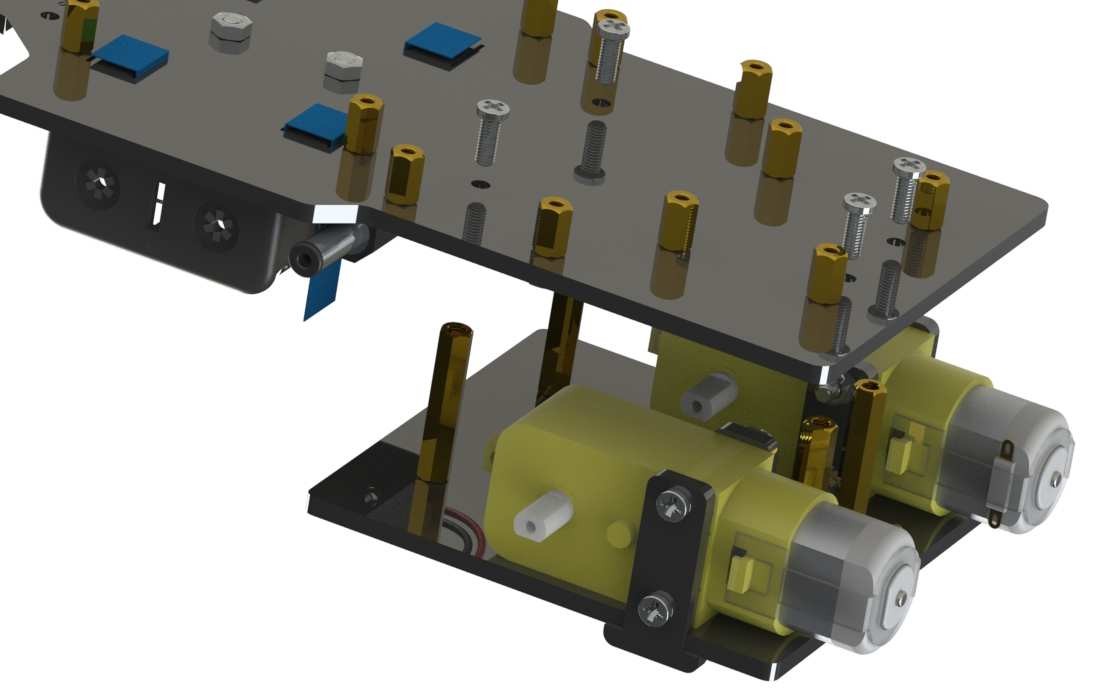


Fasten the battery holder with two **M3x8 countersunk screws** and two **M3 nuts**: pay attention to the direction of battery holder’s wire.



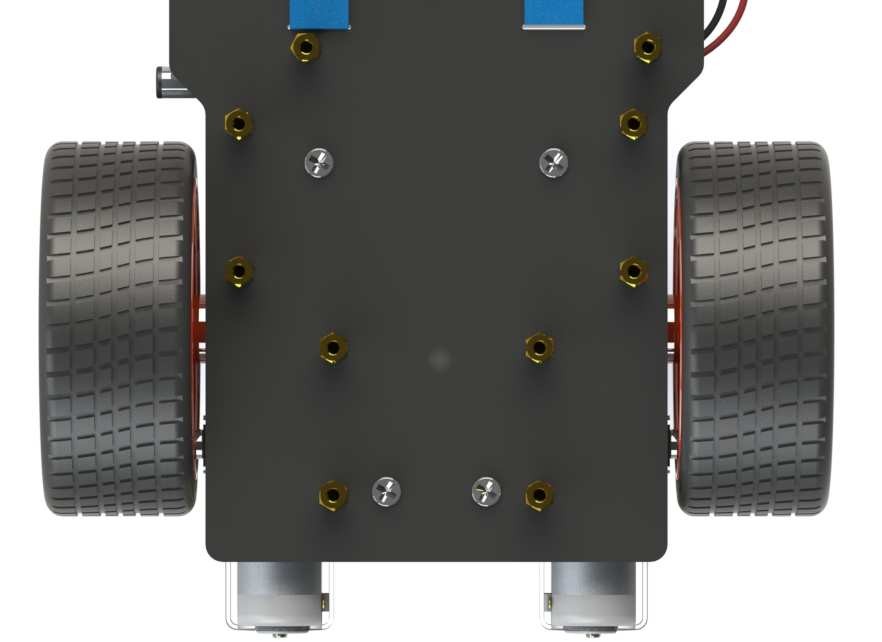
**Rear Wheels (Driving)**

Mount the assembled rear wheel driving part onto the Upper Plate with four **M3x8 screws**:



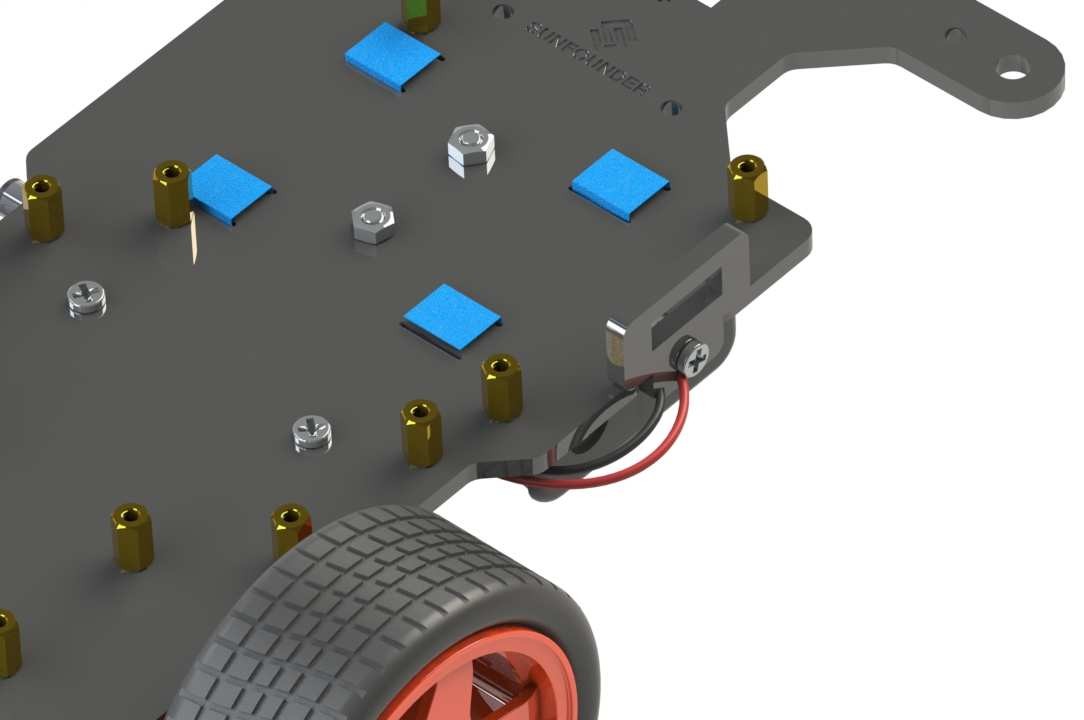
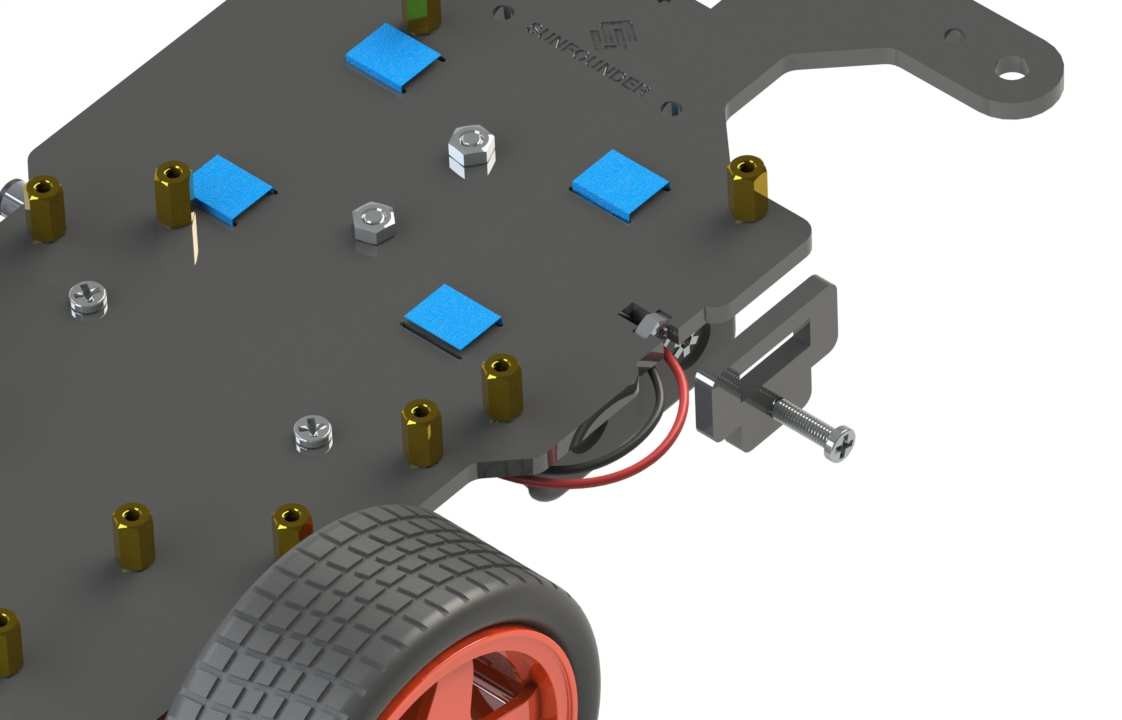
Assemble the **rear wheels**:

Align the **rear wheels** with the motor shaft, and rotate to insert them gently.



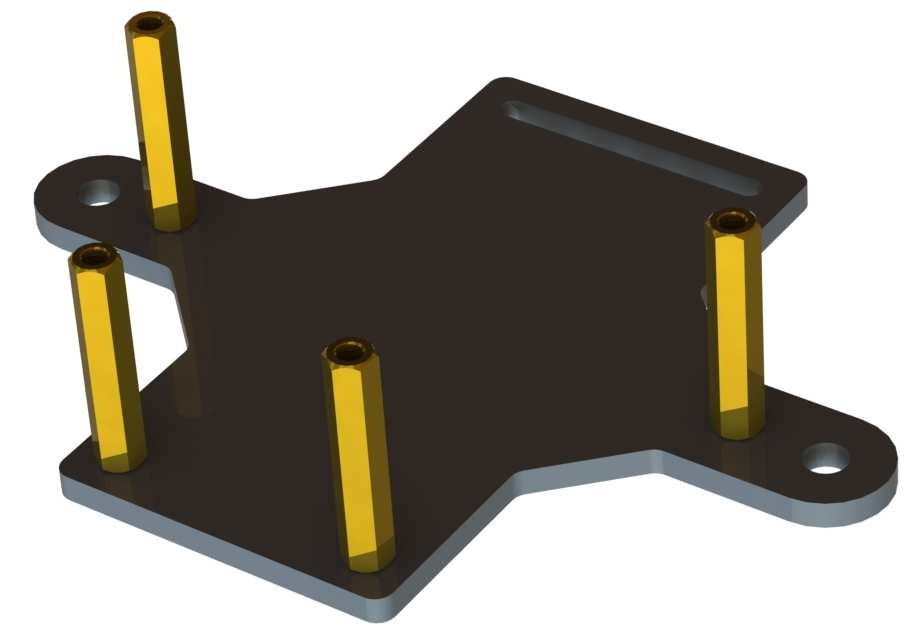
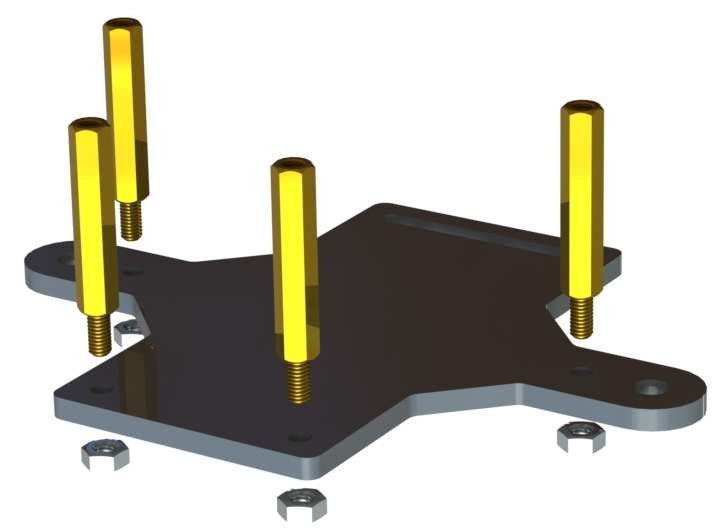
**TF Card Guard**

Mount the **TF Card Guard** plate onto the side near the front wheel with an **M3x10 screw** and an **M3 nut**. First place the nut into the slot from the underneath, and then insert the screw into the nut through the plate.



**Front Half Chassis**

Assemble the **Front Half Chassis** with four **M3x25 copper standoffs** and four **M3 nuts** as shown below:



**Front Wheels**

Insert an **M4x25 screw** through a **Flange Bearing** (pay attention to the direction – the flange near the cap of the screw), a **Steering Connector**, 2 **Bearing Shields**, 2 **Hex Front Wheel Fixing Plates**, and a **front wheel**, into an **M4 Self-locking Nut** (note the direction) as shown below:

Flange



The Self-locking Nut should be screwed tight enough. It would be better to tighten the screw until the wheel and Steering Connector cannot move first, then loosen the screw a little, so that the Steering Plate can just move. Thus, the wheel can turn flexibly when the connection would not be too loose.

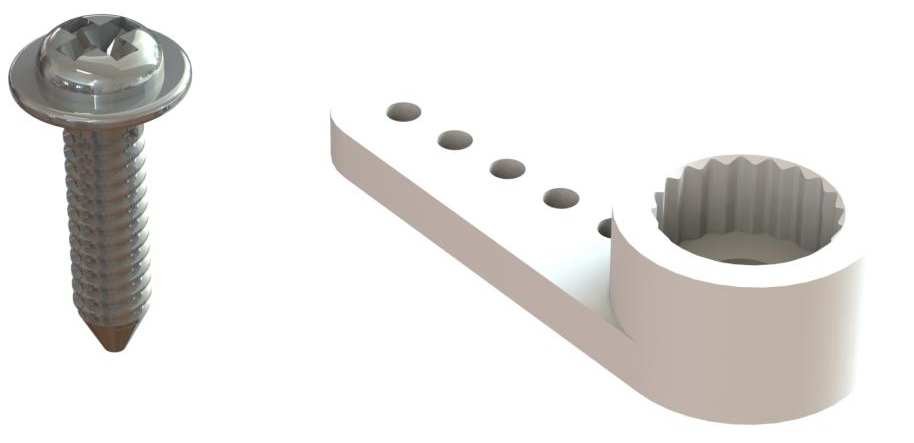
Assemble the other front wheel in the same way, but bear in mind the Steering Connector on the wheel should be symmetric with the previous one:



Now two front wheels have finished assembly.

**Steering Part**

Take out the **Rocker Arm** and the **Rocker Arm Screw** (the longer one):



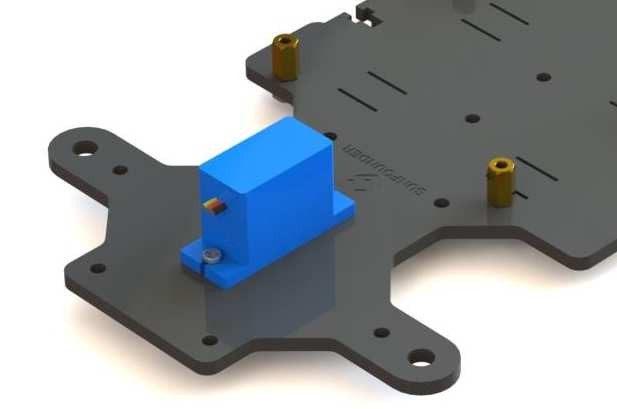
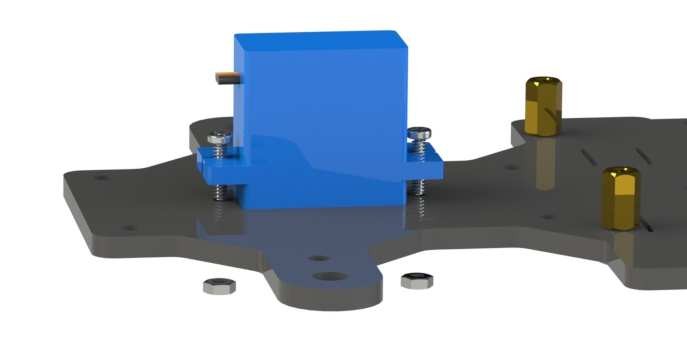
Connect the **Steering Linkage** and the **rocker arm** with the screw. **Note:** Insert it into the first hole of the arm (as indicated by the arrow below) which is the farthest from the gears. Since the screw is larger than the hole, you should try to screw it hardly so as tight to the arm. Don't worry of the arm which is soft.



And also fasten them as tightly as possible, and then loosen the screw a little so the Steering

Linkage can move flexibly.

Mount the steering servo to the Upper Plate with two **M2x8 Screws** and two **M2 nuts** (pay attention to the direction of the servo wires):

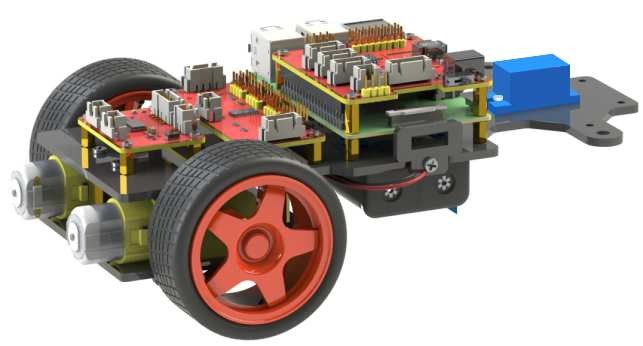


Till here the car assembly is mostly done. But **don’t hurry** to assemble the front wheels, because the servo has not been adjusted yet. An unadjusted servo may get stuck and further get hot and burnt when it is connected to power to use as it is. To adjust, you can use your own way, like uploading an Arduino program to the servo which makes it rotate to the 90 degrees and stop, or using the servo adjustment code provided in this kit.

For the latter method, since we need to connect the related driver board and Raspberry Pi to adjust the servo, now let's first assemble the **PCBs** onto the car, which is much easier than previous assembly.

**PCB Assembly**

Assemble the **Raspberry Pi** (TF Card inserted) with four **M2.5x8 copper standoffs**, then plug the **Robot HATS** onto it, and fix the **PCA9685 PWM Driver**, the **TB6612 Motor Driver** and the **Robot HATS** with twelve **M2.5x6 screws**.



TB6612

Motor Driver

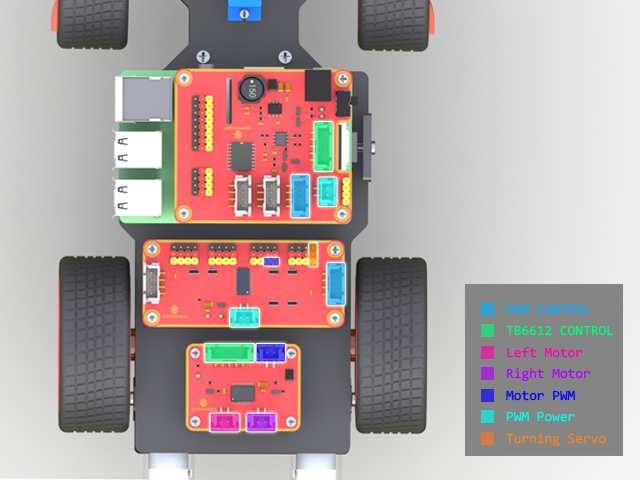
PCA9685

PWM Driver

Robot HATS

Raspberry Pi

**Circuits Building**



Connect the **PWM CONTROL** of the Robot HATS with the PCA9685 PWM Driver, and the **TB6612**

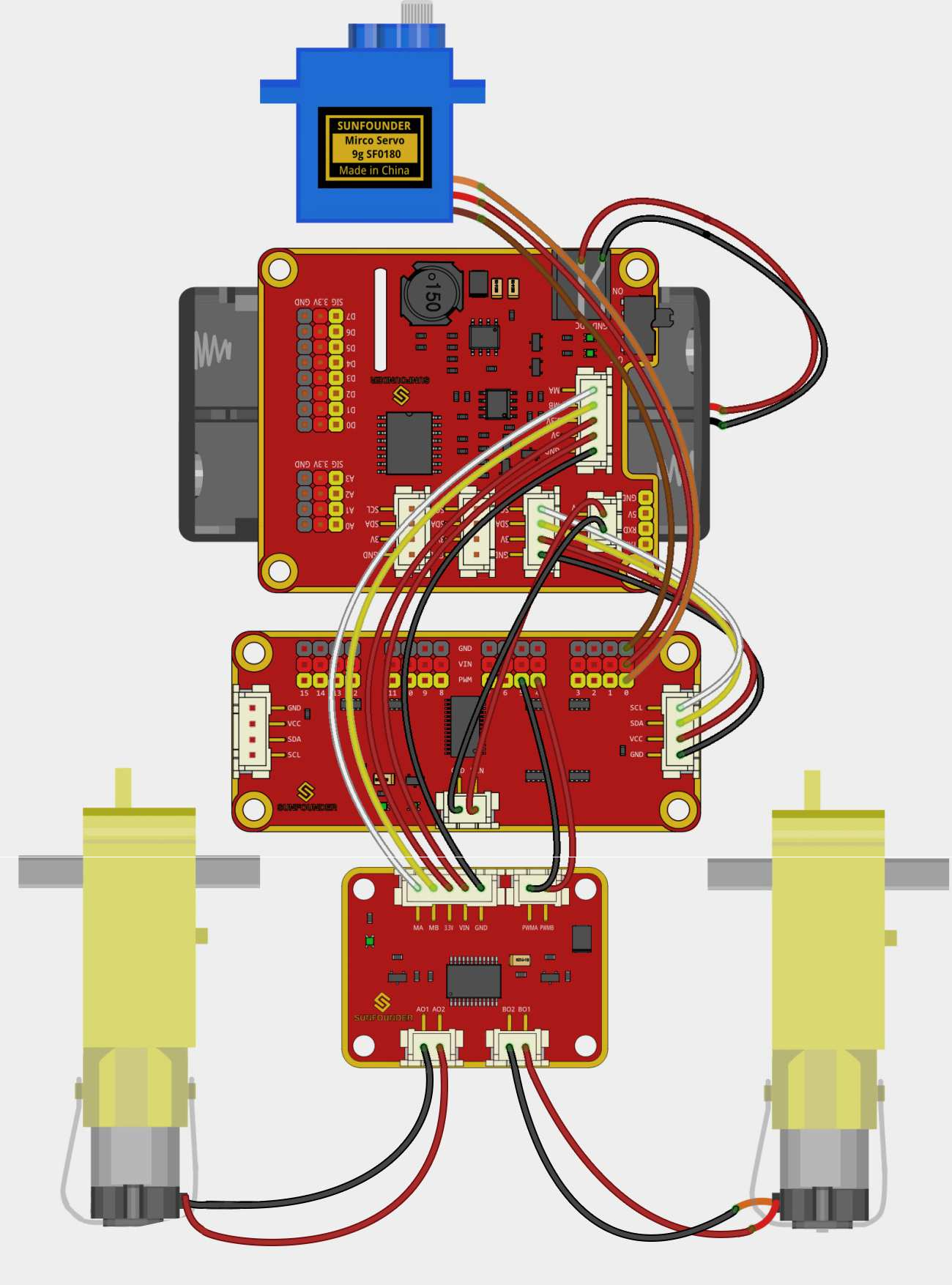
**CONTROL** with the TB6612 Motor Driver.

Then the **Left Motor** and the **Right Motor** of the TB6612 to two motors, and the **Motor PWM** with

the No. 4 and 5 PWM channel of the PCA9685.

Connect the **PWM Power** of the PCA9685 with the Robot HATS, and its channel 0 of with the

**Turning Servo.**



So now the circuit boards are all installed onto the car and the wiring is done. But still you're not ready to adjust the servo yet. First you need to complete some software installation.

**Software Installation**

**1. Log into Raspberry Pi**

The TF card onto which the Raspbian has been burnt is inserted into the Raspberry Pi before. Now power the Raspberry Pi.

*The installation may take a long time, so you're recommended to so you're recommended to supply the Raspberry Pi via a USB cable and then power the Robot HAT with batteries in case of power cut-off thus causing a sudden shutdown and file damage of the Raspberry Pi. You can directly power it via the Micro USB port because the Robot HATS won't get damaged by it due to the built-pin protective circuit.*

Plug in the USB Wi-Fi dongle (skip this if you use a Raspberry Pi 3 with the WiFi) and complete the setting in the way you're comfortable with.

Log into the Raspberry Pi via ssh or ssh tools like PuTTY.

**2. Get Source Code**

You can find the source code in our Github repositories. Download the source code by *git clone*:

git clone --recursive https://github.com/sunfounder/SunFounder\_PiCar-S.git

[pi@raspberrypi:~](mailto:pi@raspberrypi:~) $ git clone --recursive https://github.com/sunfounder/SunFounder\_PiC

ar-S.git

Cloning into 'SunFounder\_PiCar-S'... remote: Counting objects: 162, done.

remote: Total 162 (delta 0), reused 0 (delta 0), pack-reused 162

Receiving objects: 100% (162/162), 61.98 KiB | 36.00 KiB/s, done. Resolving deltas: 100% (80/80), done.

Checking connectivity... done.

Submodule 'example/SunFounder\_Light\_Follower' (https://github.com/sunfounder/SunFound er\_Light\_Follower.git) registered for path 'example/SunFounder\_Light\_Follower' Submodule 'example/SunFounder\_Line\_Follower' (https://github.com/sunfounder/SunFounde r\_Line\_Follower.git) registered for path 'example/SunFounder\_Line\_Follower'

Submodule 'example/SunFounder\_Ultrasonic\_Avoidance' (https://github.com/sunfounder/Su nFounder\_Ultrasonic\_Avoidance.git) registered for path 'example/SunFounder\_Ultrasonic

\_Avoidance'

Cloning into 'example/SunFounder\_Light\_Follower'... remote: Counting objects: 15, done.

remote: Total 15 (delta 0), reused 0 (delta 0), pack-reused 15

Unpacking objects: 100% (15/15), done. Checking connectivity... done.

Submodule path 'example/SunFounder\_Light\_Follower': checked out '10b3ccea709f34221c6d c137aca6cb00abfca417'

Cloning into 'example/SunFounder\_Line\_Follower'... remote: Counting objects: 8, done.

remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 8

Unpacking objects: 100% (8/8), done. Checking connectivity... done.

Submodule path 'example/SunFounder\_Line\_Follower': checked out '9560e7adbb52a883d438b

78de90d185fc168fed5'

Cloning into 'example/SunFounder\_Ultrasonic\_Avoidance'... remote: Counting objects: 24, done.

remote: Total 24 (delta 0), reused 0 (delta 0), pack-reused 24

Unpacking objects: 100% (24/24), done. Checking connectivity... done.

Submodule path 'example/SunFounder\_Ultrasonic\_Avoidance': checked out

'9e0349b2155fa7585021bfb979643e5f98aa51c5'

Check by the ls command, then you can see the code directory:

*SunFounder\_PiCar-S/*

[pi@raspberrypi:~](mailto:pi@raspberrypi:~) $ ls

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Desktop | Downloads | Pictures | python\_games | Templates |
| Documents | Music | Public | SunFounder\_PiCar-S | Videos |

**3. Go to the Code Directory**

cd ~/SunFounder\_PiCar-S/

Enter the code directory and you can see the installation script:

[pi@raspberrypi**:**~](mailto:pi@raspberrypi:~) $ cd SunFounder\_PiCar-S/

[pi@raspberrypi:~](mailto:pi@raspberrypi:~)/SunFounder\_PiCar-S $ ls

example install\_dependencies LICENSE README.md show

**4. Install the Environment**

**Installing Automatically by Script**

You can get all the required software and configuration done with the installation script. If you want to do step by step instead, refer to the operations in **Installing Manually** below.

sudo ./install\_dependencies

*Notes:*

*1. The installation script will install the required components and configure for the running environment. Make sure your Raspberry Pi is connected to the Internet during the installation, or it would fail.*

*2. The Raspberry Pi will reboot after the installation, so you need to log in again.*

**Installing Manually**

1. Update the apt list

|  |
| --- |
| sudo apt-get update |
| 2. Install python-smbus |
| sudo apt-get install python-smbus -y |
| 3. Install the PiCar module |
| cd ~  git clone --recursive https://github.com/sunfounder/SunFounder\_PiCar.git  cd SunFounder\_PiCar python setup.py install |
|
| 4. Enable I2C  Edit the file /boot/config.txt |
|
| sudo nano /boot/config.txt |
| The **"#"** in front of each line is to comment the following contents which does not take effect in a sketch. The I2C configuration part is commented by default too. Add the following code at the end of the file, or delete the pound mark "#" at the beginning of related line; either way will do. |
|
|
|
| dtparam=i2c\_arm=on |
| 5. Reboot |
| sudo reboot |
| **Adjust the Servo to 90 Degrees**  After reboot, type in the command: |
|
| picar |

[pi@raspberrypi**:**~](mailto:pi@raspberrypi:~) $ picar

Usage: picar [Command] [value] Commands:

servo-install Set 16 channel servos to 90 degree for installation front-wheel-test [chn] Test the steering servo connect to chn, chn default 0

rear-wheel-test Test the rear wheel

You can see three commands here.

The first one **servo-install** is for **servo adjustment**, which is used after the front wheels are assembled. The servo will rotate to 90 degrees after this command is run, so we will use this command here.

[pi@raspberrypi**:**~](mailto:pi@raspberrypi:~) $ picar servo-install

[pi@raspberrypi**:**~](mailto:pi@raspberrypi:~) $

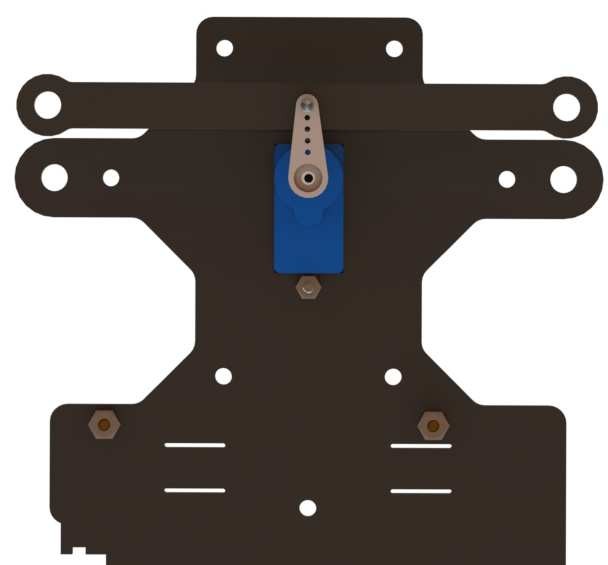
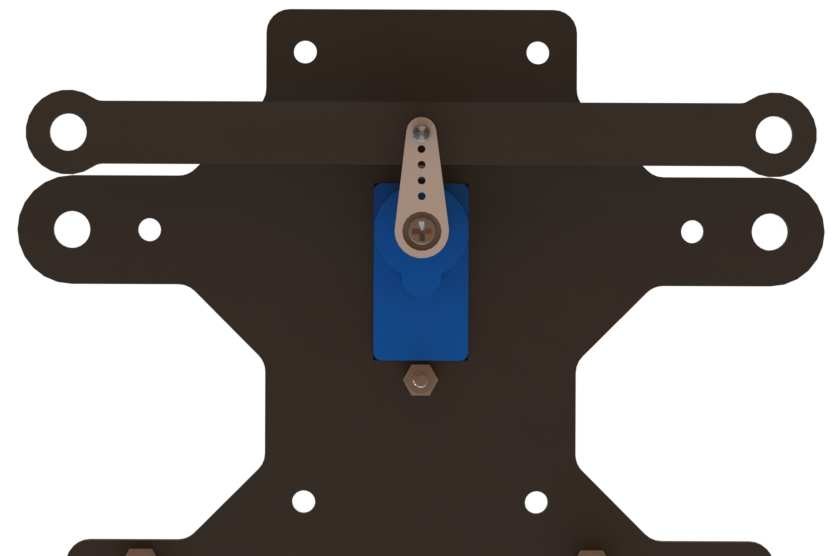
**Now the servo may or may not make noises to rotates to 90 degrees. Script will end almost**

**immediately, but all servos will keep at 90 degrees. You can continue the rest assembly.**

**Warning**: After power is on, a short circuit may happen because the exposed contacts of the PCB may be touched by conductive objects like screw or screwdriver. If it occurs, please remove the conductor immediately or unplug the power as quickly as you can.

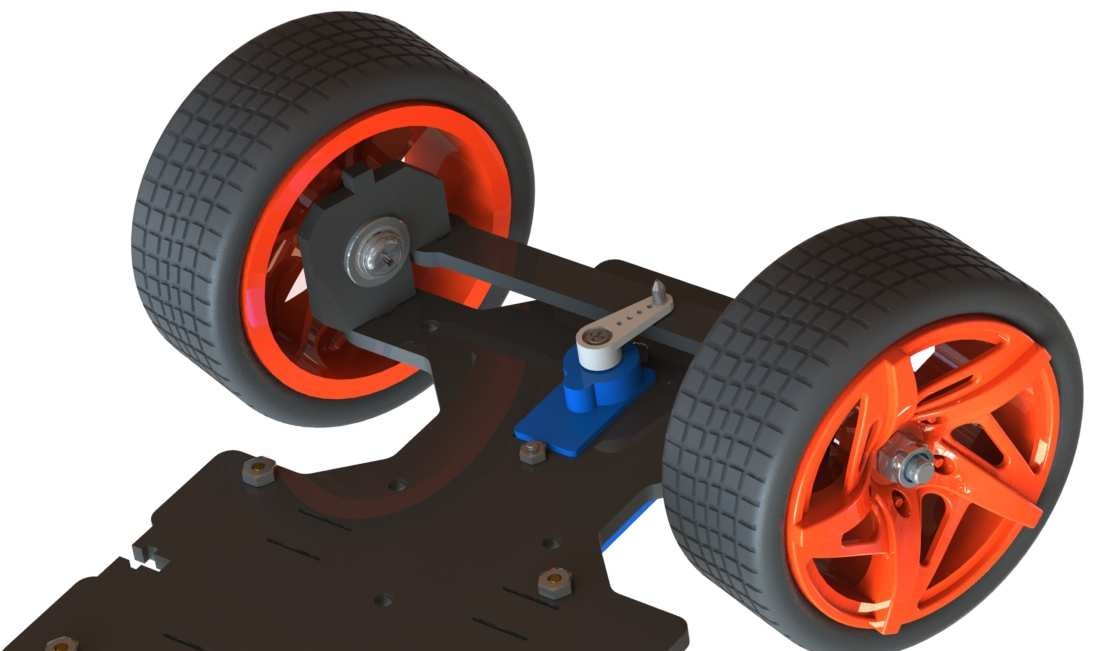
**Build the Rest of the Car**

Connect the Steering Linkage and the rocker arm. **Make sure the rocker arm is in an angle as shown below**. If there is just a littile deviation from the right position of the rocker arm, it can be fine tuned by software. Try to spin the arm on the shaft. If it's unmovable, it means the servo is adjusted to 90 degrees. But if it spins, check whether the wiring is correct. Besides, **the screw is quite sharp at the end, so be careful to assemble in case of getting hurt**.

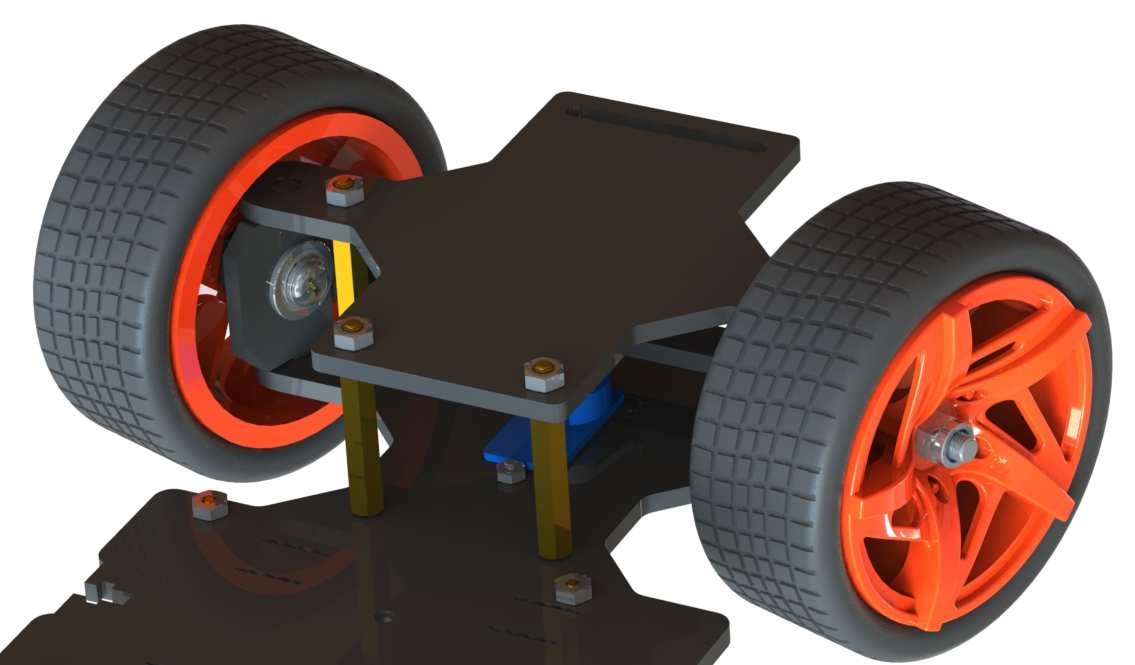


If everything is OK, take out the **Rocker Arm Fixing Screw** to connect them as shown below.

Take out the assembled front wheels and the Upper Plate, and mount the wheels onto the Upper Plate carefully: insert one bulge of the Steering Connector at either wheel into the hole on the Steering Linkage plate, and another bulge into the Upper Plate, and similar with the other wheel.

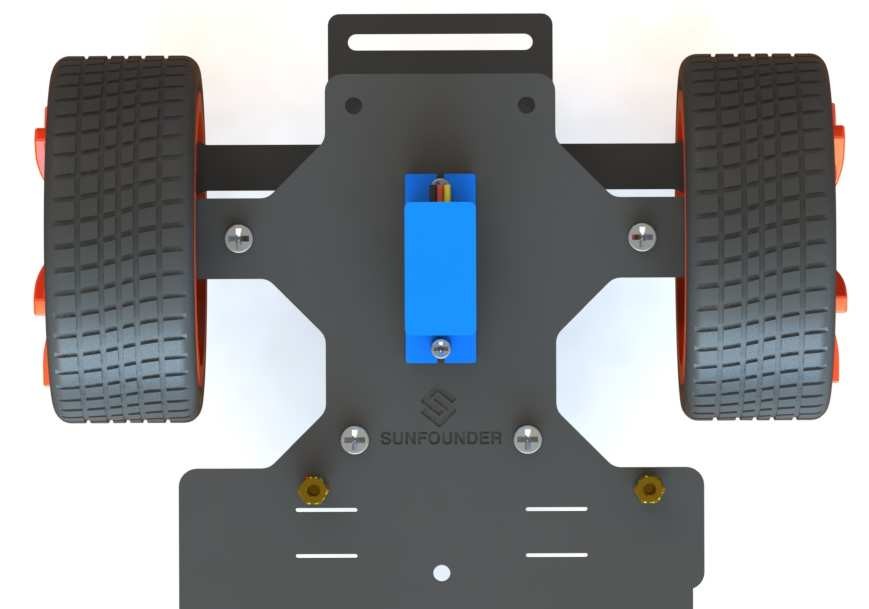


Then put the assembled Front Half Chassis onto the Upper Plate with standoffs aligned with the holes.



Hold them carefully, turn upside down, and fasten the standoffs and the Uppe Plate with four

**M3x8 screws**:



Now, the whole assembly, the wiring and code are all **DONE**! Congrats! Enjoy the satisfaction and thrill.

**Configuration**

Remember the commands to adjust the servo to 90 degrees previously? Now, let’s talk about the other two commands.

The second command **front-wheel-test** is used to test whether the front wheels can turn flexibly after assembly. When you run this command, it will drive them to turn left and right. Besides, there is an optional parameter **[chn]**. It will be used only when you change the PWM channel of the servo. If you use the default channel 0 according to the wiring diagram, there is no need to enter **[chn]**.

[pi@raspberrypi:~](mailto:pi@raspberrypi:~) $ picar front-wheel-test

DEBUG "front\_wheels.py": Set debug off

DEBUG "front\_wheels.py": Set wheel debug off

DEBUG "Servo.py": Set debug off turn\_left

turn\_straight turn\_right turn\_straight turn\_left turn\_straight turn\_right

^Cturn\_straight

You may find the direction of the front wheels is not facing exatly front when they are in the straight status (they will return to the status anytime the front wheels program stops running). If there is an obvious deviation from the middle line of the front chassis, remove the servo and run **servo-install** again; if it is just a little deviation (like about 0~15 degrees), it can be adjusted by software.

Get into the folder ***SunFounder\_PiCar/picar:***

cd /home/pi/SunFounder\_PiCar/picar ls

[pi@raspberrypi:~](mailto:pi@raspberrypi:~) $ cd /home/pi/SunFounder\_PiCar/picar

[pi@raspberrypi:~](mailto:pi@raspberrypi:~)/SunFounder\_PiCar/picar $ ls

back\_wheels.py filedb.py

init .py SunFounder\_PCA9685

config front\_wheels.py PCF8591.py SunFounder\_TB6612 [pi@raspberrypi:~](mailto:pi@raspberrypi:~)/SunFounder\_PiCar/picar $

Open the **config** file under the folder with an editor. You can see a few parameters. Modify the value of **turning\_offset** (the first parameter) to adjust the front wheels. Its value is **0** by default, and you can turn the wheels left by decreasing the value, or turn right by increasing it. For example, if you want to make the front wheels turn right a bit, just modify it to a larger number; to make it more towards the left, you can set it smaller (it can even be a negative number).

But **DO NOT** over-configure the wheels (recommended a value between -30 and 30), or the servo may be stuck and broken.

[pi@raspberrypi:~](mailto:pi@raspberrypi:~)/SunFounder\_PiCar/picar $ nano config

GNU nano 2.2.6 File: config

# File based database

turning\_offset = 10

forward\_A = 1

forward\_B = 1

[ Read 9 lines ]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ^G | Get Help | ^O | WriteOut | ^R | Read File | ^Y | Prev Page ^K | Cut Text | ^C | Cur Pos |
| ^X | Exit | ^J | Justify | ^W | Where Is | ^V | Next Page ^U | UnCut Text | ^T | To Spell |

After changing the value of turning\_offset, press **Ctrl** + **O** to save the changes, and press **Ctrl**

+ **X** to exit. Run the command **picar servo-install** to check the front wheel’s status.

[pi@raspberrypi**:**~](mailto:pi@raspberrypi:~)/SunFounder\_PiCar/picar $ picar servo-install

[pi@raspberrypi**:**~](mailto:pi@raspberrypi:~)/SunFounder\_PiCar/picar $

If the front wheels is still not facing the exact front, you may need to edit the file **config** for a couple of times. The front wheels may need to be adjusted about 3 to 5 times usually. We can move on to calibration of the rear wheels when the front wheels are done.

Since the wiring of the two DC motors is random, the VCC and GND of a motor may be connected to the wheel reversely, causing the wheel to spin forward when it should do backward as configured in the code. Thus we can use the third command which will drive the rear wheels to simultaneously speed up and slow down alternately.

[pi@raspberrypi**:**~](mailto:pi@raspberrypi:~) $ picar rear-wheel-test DEBUG "back\_wheels.py": Set debug off DEBUG "TB6612.py": Set debug off

DEBUG "TB6612.py": Set debug off DEBUG "PCA9685.py": Set debug off Forward, speed = 0

Forward, speed = 1

Forward, speed = 2

Forward, speed = 3

Forward, speed = 4

Forward, speed = 5

Forward, speed = 6

Forward, speed = 7

Forward, speed = 8

Forward, speed = 9

Forward, speed = 10

.

.

.

Backward, speed = 10

Backward, speed = 9

Backward, speed = 8

Backward, speed = 7

Backward, speed = 6

Backward, speed = 5

Backward, speed = 4

Backward, speed = 3

Backward, speed = 2

Backward, speed = 1

Finished, motor stop

Check whether both the two rear wheels rotate direction is the same as the screen promt.

Note that the two wheels are driven by the two motors separately. It may happen that one rotates forward, while the other does backwards. If so, we need to adjust one or both two wheels which rotate reversely under that command.

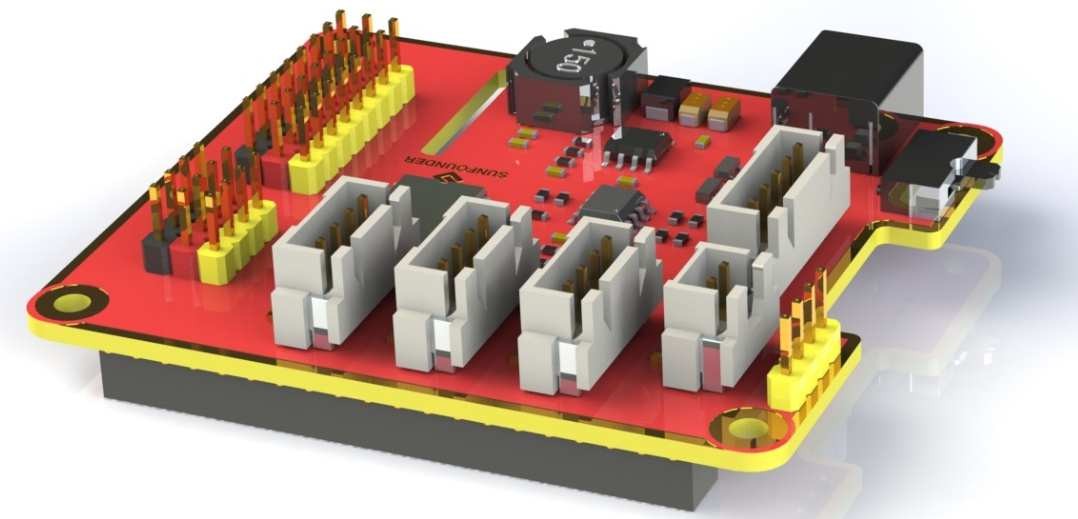
The two other parameters in the file **config** are to set the rear wheels: forward\_A and forward\_B. They can change the default spinning direction of the two motors. The value can only be **0** or **1**, which represents clockwise and counterclockwise rotation. By default, it's **1** for both parameters. Thus if a wheel spins reversely, you only need to change the corresponding parameter for the wheel to **0**.

Press **Ctrl** + **O** to save the changes, and press **Ctrl** + **X** to exit. Run the command **picar rear- wheel-test** again to check whether the rear wheels are rotating in accordance with the command.

Copy *config* to the directory *example* under *PiCar-S*.

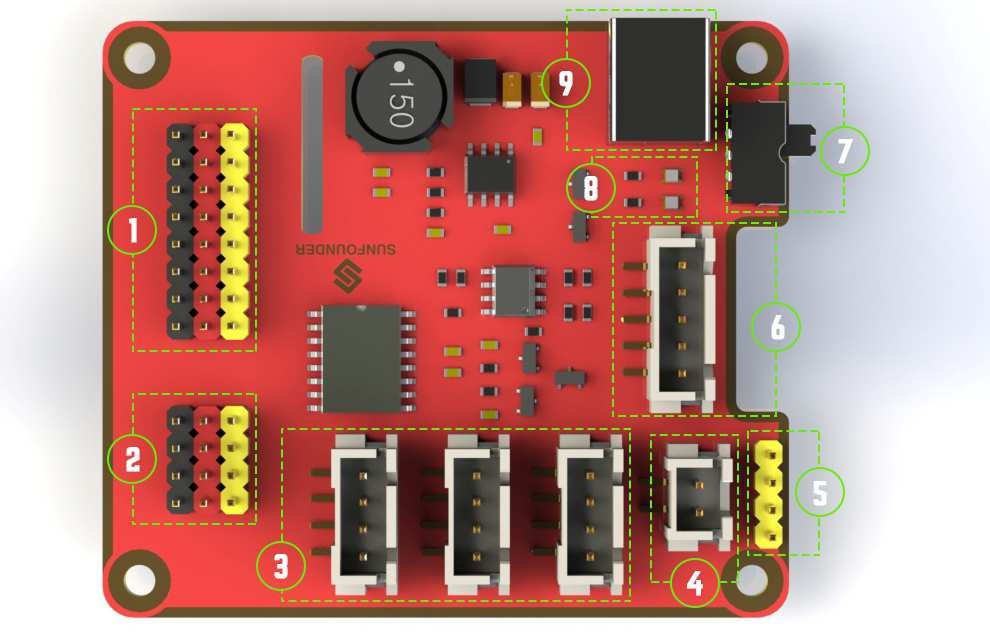
[pi@raspberrypi:~](mailto:pi@raspberrypi:~)/SunFounder\_PiCar/picar $ cp config ~/SunFounder\_PiCar-S/example

**Appendix Appendix 1: Modules Robot HATS**



**Robot HATS** is a specially-designed HAT for a 40-pin Raspberry Pi and can work with Raspberry Pi model B+, 2 model B, and 3 model B. It supplies power to the Raspberry Pi from the GPIO ports. Thanks to the design of the ideal diode based on the rules of HATS, it can supply the Raspberry Pi via both the USB cable and the DC port thus protecting it from damaging the TF card caused by batteries running out of power. The PCF8591 is used as the ADC chip, with I2C communication, and the address 0x48.

○**1 . Digital ports**: 3-wire digital sensor ports, signal voltage: 3.3V, VCC voltage: 3.3V.



○**2** . **Analog ports**: 3-wire 4-channel 8-bit ADC sensor port, reference voltage: 3.3V, VCC

voltage: 3.3V.

○**3 . I2C ports**: 3.3V I2C bus ports

○**4** . **5V power output**: 5V power output to PWM driver.

○**5** . **UART port**: 4-wire UART port, 5V VCC, perfectly working with SunFounder FTDI Serial to USB.

○**6** . **TB6612 motor control ports**: includes 3.3V for the TB6612 chip, 5V for motors, and direction

control of motors MA and MB; working with SunFounder TB6612 Motor driver.

○**7** . **Switch**: power switch

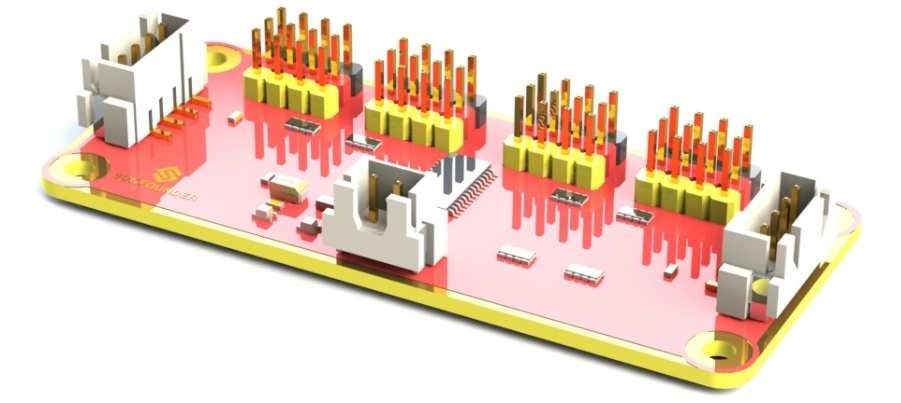
○**8** . **Power indicators**: indicating the voltage – 2 indicators on: >7.9V; 1 indicator on: 7.9V~7.4V;

no indicator on: <7.4V. To protect the batteries, you're recommended to take them out for charge when there is no indicator on. The power indicators depend on the voltage measured by the simple comparator circuit; the detected voltage may be lower than normal depending on loads, so it is just for reference.

○**9** . **Power port**: 5.5/2.1mm standard DC port, input voltage: 8.4~7.4V (limited operating

voltage: 12V~6V).

**PCA9865**



PCA9685 16-channel 12-bit I2C Bus PWM driver. It supports independent PWM output power and is easy to use 4-wire I2C port for connection in parallel, distinguished 3-color ports for PWM output.

○1 . **PWM output ports**: 3-color ports, independent power PWM output port, connect to the servo directly.



○2 & ○3 . **I2C port**: 4-wire I2C port, can be used in parallel. Compatible with 3.3V/5.5V

○4 . **PWM power input**: 12V max.

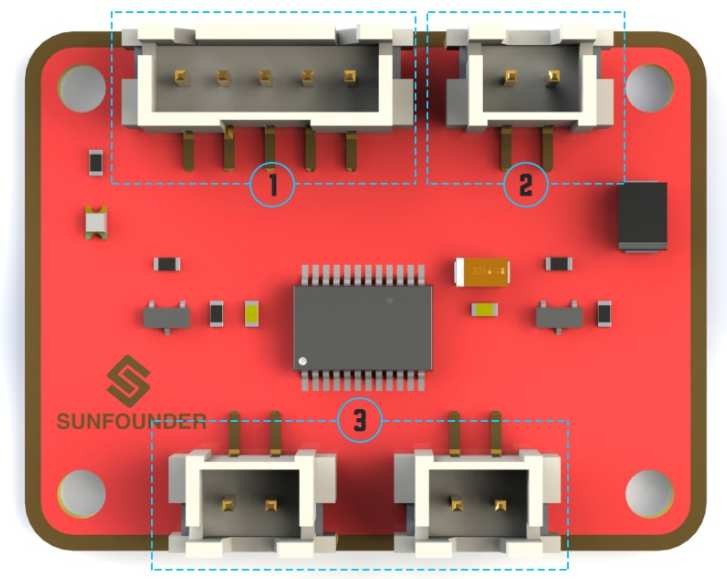
○5 . **LED**: power indicator for the chip and for the PWM power input.

**TB6612**



The TB6612 Motor Driver module is a low heat generation one and small packaged motor drive.

○**1 Power and motor control port**: includes pins for supplying the chip and the motors and controlling the motors' direction



○**2 PWM input for the motors**: PWM signal input for adjusting the speed of the two motors

○**3 Motor output port**: output port for two motors

**SunFounder SF0180 Servo**



The SunFounder SF0180 Servo is a 180-degree three-wire digital servo. It utilizes PWM signal of

60Hz and has no physical limit – only control by internal software to 180 degrees at most.