Configuration of the PiCar-V

This document serves as a guide on what steps are necessary to assemble, configure, and use the SunFounder PiCar-V kit that is used in our project.

The first step after getting the picar kit is to take the motors, and all the plastic pieces and assemble them. We attach the raspberry pi to the car as well as a Motor Driver and PWM Driver. The robot HATS that goes on the top of the raspberry pi then connects to these boards. The two motors connect to the motor driver and this connects to both the HATS and the PWM driver.

In order to configure the raspberry pi, we must first download and install the NOOBs operating system onto the micro-SD card This is done on a separate computer and then once the process is completed we start up the raspberry pi and install the OS. Installing the OS is very straight forward, but then there are a few more steps we need to do before we finish setting up the car.

The next step after getting the raspberry pi raspian OS installed is to download the source code from the SunFounder PiCar repository on github. This repo has code that we can execute in order to install the dependencies needed to control the picar.

Once we have used the git clone command and downloaded the repository, we navigate to the directory within the terminal and run the command ‘sudo ./install-dependencies’. This script checks whether or not the required dependencies are installed and if not then they will download them using the default package manager for raspian. After this, the raspberry pi will ask to be restarted, and needs to be before moving on to the next steps.

Once the pi has restarted, we can now begin to configure the servos. The open-source code that comes with the car has a command that is installed called ‘picar’ and we use the command ‘picar servo-install’ to begin this process. It should be noted that in order for this to work, we will first need to have batteries installed and the robot HATS should be turned on. After we execute the command, the servos will start to spin and configure themselves to 90-degrees. According to the information in the picar setup manual this means that the servos are configured to be pointing directly ahead (straight).

After the servos are configured to be 90 degrees, we can attach a load on them and finish the rest of the car assembly. It should be noted that once the car has been completed we cannot access the HDMI port on the raspberry pi because a servo is in the way. In order to get past this we have enabled SSH using the command ‘sudo raspi-config’. Further we also need to use this configuration tool to set the pi’s hostname on the network. This makes the task of connecting to the picar through the network much easier.

After we have completed the front chasis of the car, we assemble the camera attachment mount which has two servos on it for pan & tilt of the camera. Once this is completed and the camera is connected via usb, we are done assembling the car kit and its components.

One extra step I took with the raspberry pi is configuring it to have a vnc server on it, so that we can have some sort of graphical user interface if necessary. This was done by using the built-in package manager to download and install using the command ‘sudo apt-get install tightvncserver’ followed by the command ‘tightvncserver’. After we run the last command we will be prompted to set up a password so that we can connect via vnc. Once that is done, we can run the command ‘vncserve :1 –geometry 1920x1080 –depth 24’. This creates a vnc server that will give the resolution of 1920x1080 to anyone who connects. This should be plenty enough of a large screen for most operations on the pi.

Following the assembly of the car and the installation of the vnc server we then can install the client on our own computer. This allows us to do a few things such as control the car remotely, but more importantly change the default servo locations. By this point in the assembly you may have noticed that the front wheel servos don’t line up directly straight, and the servos for the pan and tilt of the camera are off as well. To correct we must first download the client and run it using python3. The issues most people face here is that they don’t use the correct python package manager to install the dependencies. In order to run the client, we need to have PyQt5 installed through the pip3 package manager. Once this is done it is as simple as navigating to the correct folder (client) and running the script using python3.

The client application allows us to do a few basic things, but we are interested the most in changing the default servo directions. This is not the same as configuring the servos to 90 degrees like we did earlier, although it is very similar. When we attached loads to the servo initially, the grooves on the servo force whatever its connecting to slightly off from straight. The client application allows us to use keyboard keys to turn the servos slightly enough that they will now be straight. What is important about this step is that following the manual configuration, the client application will write the values chosen to the PiCar so that these values bcome the default from now on. This was a very important step because it allowed us to manual move the pan & tilt of the camera, as well as the front wheels. One issue that might occur is that the wiring on the motors are done at random so each back wheel might be going in opposite directions or both of them go backwards when the car is supposed to be going forwards and vice-versa. Using the client application allows us to configure this very easily and not worry about changing the code within the repository or making it so that we will have to program it opposite.

Once these steps have been taken, the car is now ready to be programmed, or controlled remotely by the client application.

More information can be found the manual <https://www.sunfounder.com/learn/download/U21hcnRfVmlkZW9fQ2FyX1YyLjBfZm9yX1Jhc3BiZXJyeV9QaV9QaUNhci1WXy5wZGY=/dispi>