September 14, 2020 Yutong

- Date, name should be included in the title, for example, September 14, 2020 Yutong.
- Describe the progress in the main paragraph (may be use bullets to highlight the main progress).
- Include anything which you think will be valuable to other people.
- Describe the unsolved problems.
- Attach necessary codes, demonstration videos or other files in the /RC-car/Phase1/Attachment/, create a new folder with the name of the date when tests are conducted, so the progress log can be matched with the attached files.

Progress: In “get your Raspberry pi Working”, steps 1-4 are done

1) Rpi image is installed on the micros card

2) To login to the wifi on the first boot a file named wpa_supplicant.conf has been created in the root of the boot partition of microsd card, but the password for lab wifi is yet to be entered on that file.

3) (skipped) linux host pc command to setup Pi’s Hostname(by default it is : raspberrypi.local)

4) By creating a file named ssh in the root of the boot partition, SSH has been enabled.

ToDo: 1) The microsd card on which the image is installed has a boot partition. In the root of the boot partition, edit the file named wpa_supplicant.conf, by entering the password for lab wifi in psk , so it can connect to the wifi(This is from step 2)

2) After this start from step 5

--

September 16, 2020 Xiao

--

PROGRESS:

1. Set up Ubuntu 18.04 in Z230 workstation PC
2. Wireless module is installed
3. Git repo. is cloned into local folder /home/rc-car/RC-car

NOTE: password for PC is rccar

TODO:

1. Anal might need to spend some time move her work into Lab PC from her mac and set up her own git ssh

September 17,18 2020 Anal

Progress:

1) *Install Software on Host PC :* It involved git cloning of donkeycar in the local folder /home/rc-car/RC-car/rc_project and creating a python anaconda environment. **Note** : To activate that environment cd to /home/rc-car/RC-car/rc_project/donkeycar and use command *conda activate donkey*

2) *Get Your Raspberry Pi Working* : Steps 1-12 complete. It involved setting up Rpi (flashing a micros card with a raspian os —> setting up wifi network details in microsd folder so that pi gets connected to wifi automatically once it is switched on —> enabling ssh —>connecting to rpi via ssh using its ip address —> using command ***sudo raspi-config*** to configure the rpi such as enabling interfacing options I2C and camera —> then setting up a python3 virtual environment in the rpi and install Donkeycar Python Code and other python libraries in that environment.)

Notes:

1) pi's hostname: raspberrypi.local (default)

2) pi's username : pi (default)

3) pi's password : raspberry (default)

4) command to ssh into rc car's rpi is *pi@raspberrypi.local*

this command will work only when both the lab pc and raspberry pi are connected to same wifi network. Once rpi is switched on, it gets automatically connected to wifi with ssid NETGEAR as in wpa_supplicant.conf file (in the root of the boot partition of microsd card in the rpi), I have put the network details of only one wifi - the NETGEAR wifi. So make sure lab pc is also connected to NETGEAR . (password for wifi is _NETGEAR)

5) After ssh into rc car (rpi) and logging in, a python3 virtual environment will get automatically enabled and this is where donkeycar python code is installed. Opencv and tensorflow are also installed in this virtual env. (To return from python3 virtual environment to system python use command *deactivate).*

6) to change rpi configuration (such as changing password, enabling interfacing options, changing wifi network details etc) use command *sudo raspi-config* .after changing reboot the pi *(*same as step 7 in Get Your Raspberry Pi Working*)* .

*i*nterfacing options I2C and camera are already enabled.

ToDo: Create donkeycar app, steering/throttle calibration, drive with remote control

----
--

September 19,20,21 2020 Xiao

--

PROGRESS:

1. Installed and Enabled GUI Desktop for Boot Raspi

2. Connected I2C PCA9685 (Drive Board) and step "Create Donkeycar App" is finished (actually most of the steps can be skipped in that part). The car app is in folder *`/home/pi/mycar`*

3. Coarse calibration is done and fine calibration is in progress

4. Steering fine calibration is done and PWM signal to steering angle map built

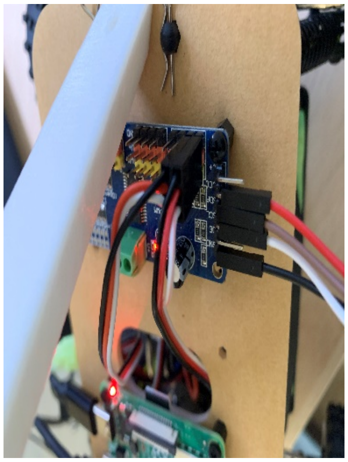
5. Remote drive functionality works well (using keyboard)

ISSUE:

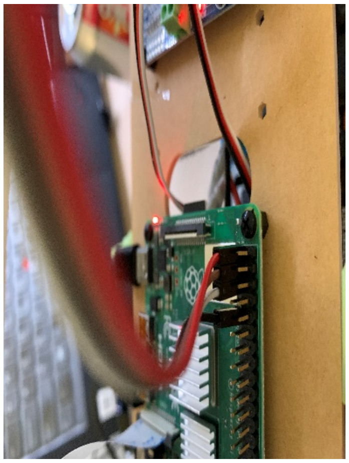
1. Battery power supply for motor, drive board and Raspi simultaneously can be unstable. This will constantly cause Raspi to reboot. Charge the battery before use.

Note:

1. If you want change back to Comand Line Interface (CLI):
   - Open terminal on the top navigation bar (ctrl+alt+t)
   - *sudo raspi-config*
   - Choose "3 Boot Options > B1 Desktop / CLI > B1 Console
   - Type *"Reboot"* in CLI to reset the configuration
2. SSH to your raspi
   - In your host PC CLI, type ssh pi@raspberrypi.local
   - If above command is not working:
     - *ping raspberrypi.local* (Id shown in the CLI e.g. 192.168.1.115)
     - *ssh pi@192.168.1.115*
3. Hardware wiring:
   - 20:00 min in https://www.youtube.com/watch?v=OaVqWiR2rS0&t=48s
   - In case the wires fall out, here are wire connection detail pictures



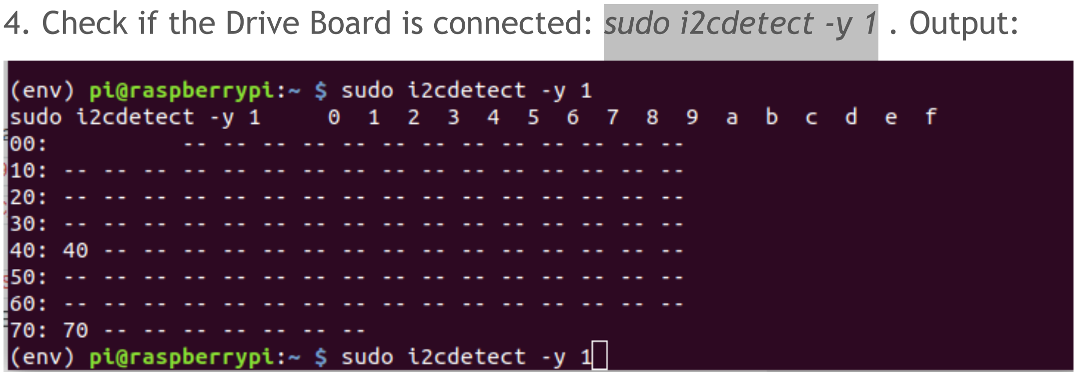
Wire to Drive Board



Wire to Rpi

-
-
-
-
-
-

4. Check if the Drive Board is connected: *sudo i2cdetect -y 1* . Output:

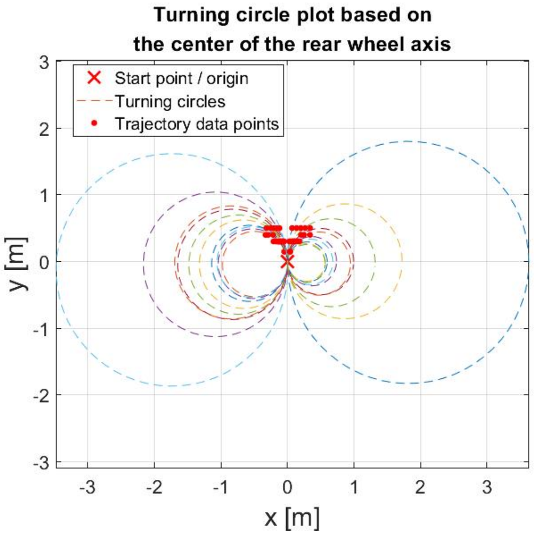


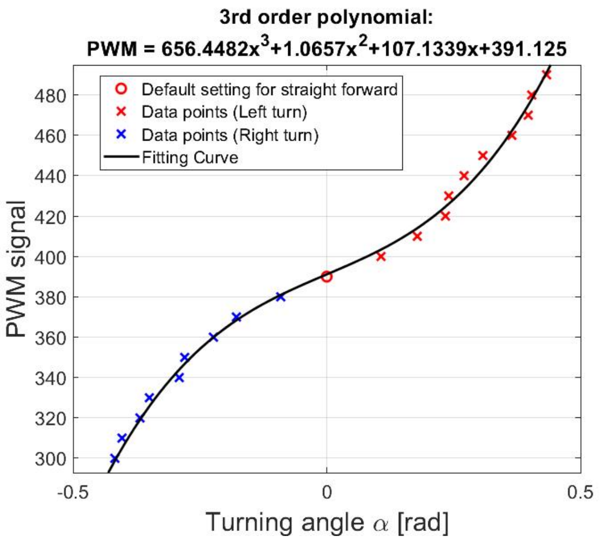
```
(env) pi@raspberrypi:~ $ sudo i2cdetect -y 1
sudo i2cdetect -y 1     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
(env) pi@raspberrypi:~ $ sudo i2cdetect -y 1
```

5. Steering and throttle calibration channel:

- <steering_channel> = 1
- <throttle_channel> = 0
- STEERING_LEFT_PWM = 490
- STEERING_RIGHT_PWM = 290
- THROTTLE_FORWARD_PWM = 470
- THROTTLE_STOPPED_PWM = 380
- THROTTLE_REVERSE_PWM = 290

6. Calibration data is in “Phase1/Car parameter and Calibration.xlsx”





7. Follow the step in “Drive your car”, you can maneuver the RC car using your keyboard.

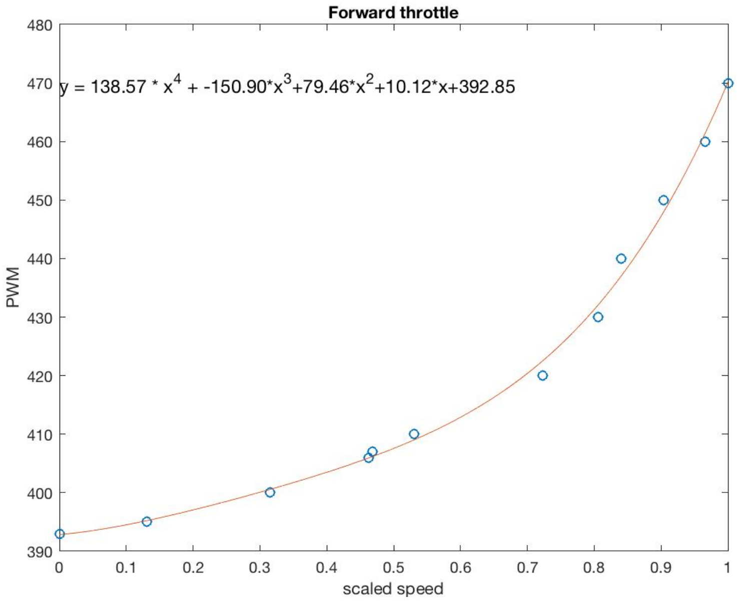
TODO:

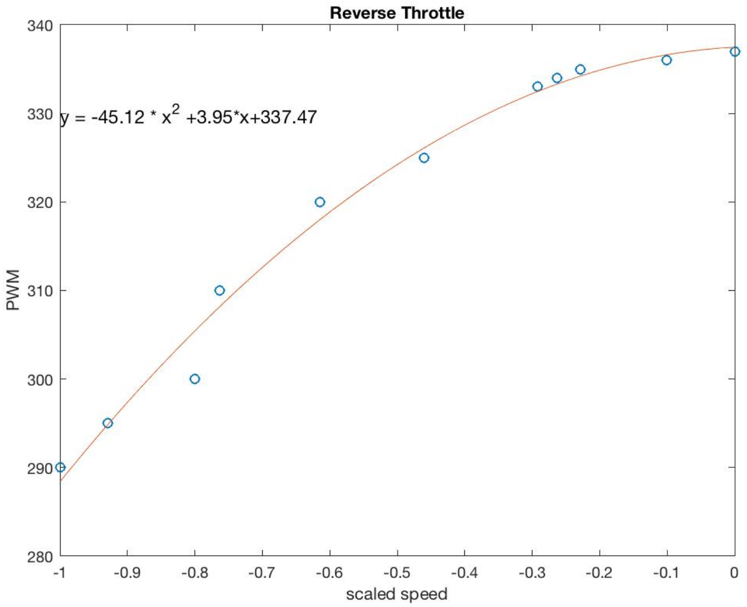
1. throttle calibration: Map car speed [m/s] to PWM signal input.

Sept 22,24,25, 2020 Anal

Progress:

1) Throttle calibration done





2)In home/pi/project/donkeycar/donkeycar/parts/actuator.py :

In class PWMSteering, inside functions __init__ and run_threaded, self.pulse changed from dk.utils.map_range(angle, self.LEFT_ANGLE, self.RIGHT_ANGLE, self.left_pulse, self.right_pulse) to dk.utils.map_range_steering(angle), since utils.map_range function is a linear mapping.

In class PWMThrottle, inside function run_threaded, for reverse throttle, self.pulse changed from dk.utils.map_range(throttle,self.MIN_THROTTLE,0,self.min_pulse,self-.zero_pulse) to dk.utils.map_range_reverse_throttle(throttle).

In home/pi/project/donkeycar/donkeycar/utils.py, map_range_steering(angle)(scaled for xrange=[-1,1]) and map_range_reverse_throttle(throttle)(scaled for xrange=[-1,1]) are added.

Todo: add function map_range_forward_throttle(throttle) in utils.py and change dk.utils.map_range to map_range_forward_throttle in the class PWMThrottle for throttle>0.

Issues:

1) Forward throttle calibration is done in the lab but reverse throttle calibration is done in the apartment. for (approx)337<=PWM<=(approx)345 the car was not moving, but in the lab it was moving for those PWM values.

2) In the lab, for PWM 380 the trajectory of car was closer to a straight line as compared to PWM=391.

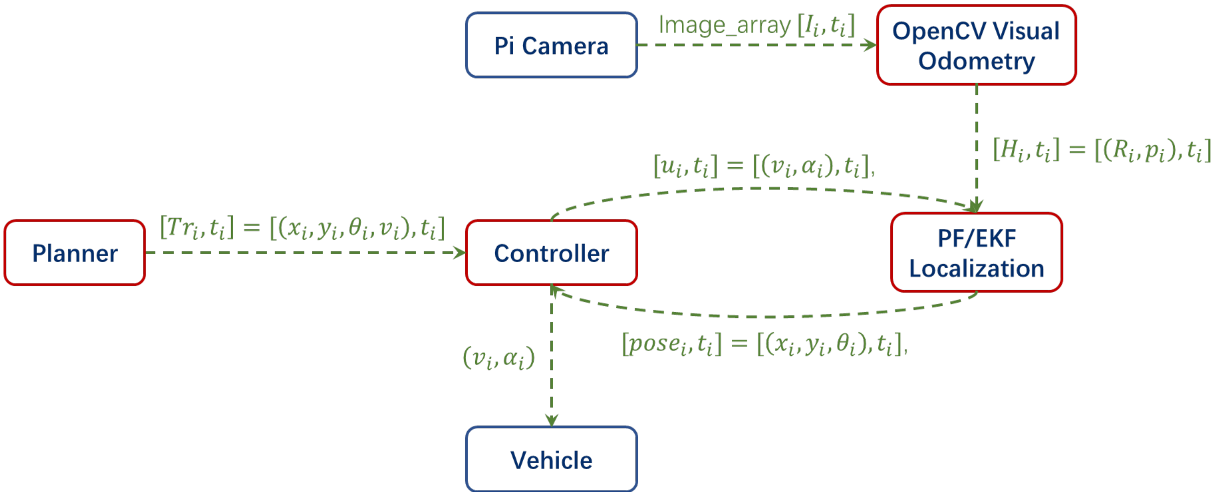
--

September 26，27，28 2020 Xiao

--

PROGRESS:

1. Resemble of ROS “Node” example and Extended Kalman Filter is written in Folder /home/pi/projects/donkeycar/donkeycar/parts/customized
2. Test example for node “TestController ” is written in /home/pi/mycar
3. Path following nodes structure: using landmarks (with QR code)


Pi Camera
Image_array $[I_i, t_i]$
OpenCV Visual Odometry
$[H_i, t_i] = [(R_i, p_i), t_i]$
$[u_i, t_i] = [(v_i, \alpha_i), t_i],$
Planner
$[Tr_i, t_i] = [(x_i, y_i, \theta_i, v_i), t_i]$
Controller
PF/EKF Localization
$[pose_i, t_i] = [(x_i, y_i, \theta_i), t_i],$
(v_i, α_i)
Vehicle

TODO:

1. Open CV get landmarks relative position p = [range, bearing] to RC car

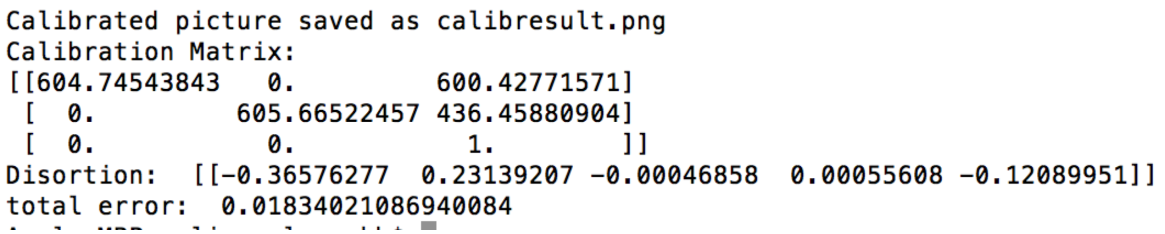
----
--

October 1,2 2020 Anal

--

Progress:

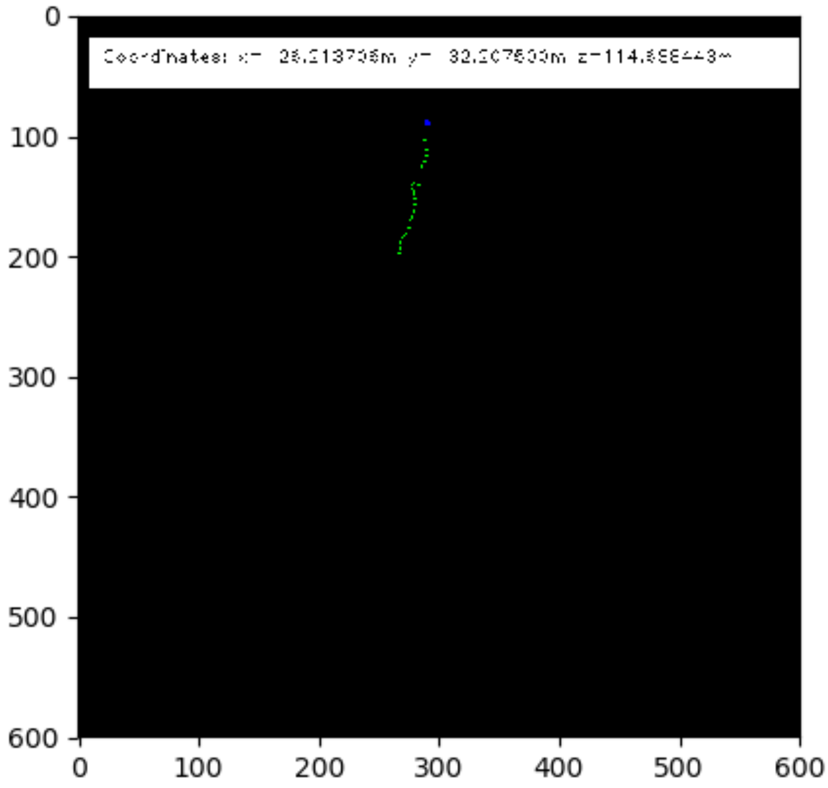
1) Rpi camera calibration: calibration matrix and Distortion



```
Calibrated picture saved as calibresult.png
Calibration Matrix:
[[604.74543843   0.         600.42771571]
 [  0.         605.66522457 436.45880904]
 [  0.           0.           1.        ]]
Disortion:  [[-0.36576277  0.23139207 -0.00046858  0.00055608 -0.12089951]]
total error:  0.01834021086940084
```

2) odometry file in /home/pi/mycar/

i don’t know how to find the scaling factor, which requires ground truth position, so i took it randomly to be 5 for testing. this plot is for images in data/tub_1_20-10_02. (x and y axis are pixel values).



x,y,z pixel values are in odom_data

x axis =x pixel values and y axis =z pixel values