# PROGRAMMING YOUR ROBOT

## 1. Software and Tools

|  |  |
| --- | --- |
| **PC** (Desktop or Laptop) | **Robot** (Raspberry Pi 3) |
| + Microsoft Window 10  + Python 3.7  + Libraries and Packages:  - OpenCV: for image processing  - PyGame: for Joystick  - Pocketsphinx: for local speech recognition engine  - PyAudio and Speech Recognition: for speech recognition interface  + IDLE (Python 3.7) or Pycharm Community: for Python Editor  + PuTTY: for remote Terminal  + VNC Viewer: for remote desktop) | + Raspbian  + Python 3.5 (3.7)  + Robot Software and Libraries  + IDLE (Python 3)  + VNC Server: for remote desktop |

References:

+ Manual-PiCar-B-V1.0-Compressed.pdf

+ <https://www.adeept.com/picar-b_p0117.html> / <https://www.adeept.com/learn/>

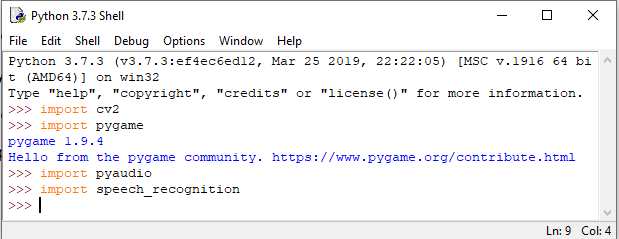
+ <https://github.com/adeept/Adeept_PiCar-B> (for updated python package from Adeept)

√ 1. *Check on your PC if the following software or libraries are installed?*

Python 3.7 IDLE (python 3.7) PuTTY VNC

√ 2. *Check on your PC if the following Python packages are installed? (Open IDLE and Test import commands)*

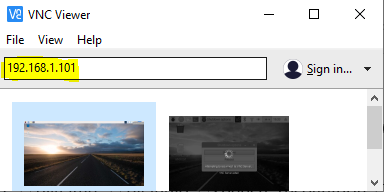
Opencv (cv2) pygame pyaudio speech\_recognition



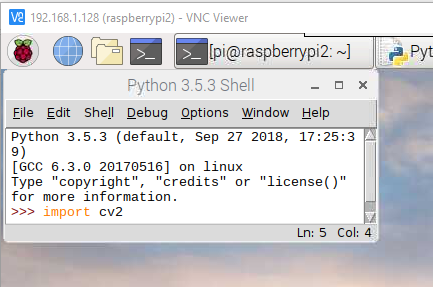
√ 3. *Check on your Robot (Raspberry Pi - RPi) if the following Python packages are installed?*

Opencv (cv2) pygame pyaudio speech\_recognition RPi.GPIO Adafruit\_PCA9685

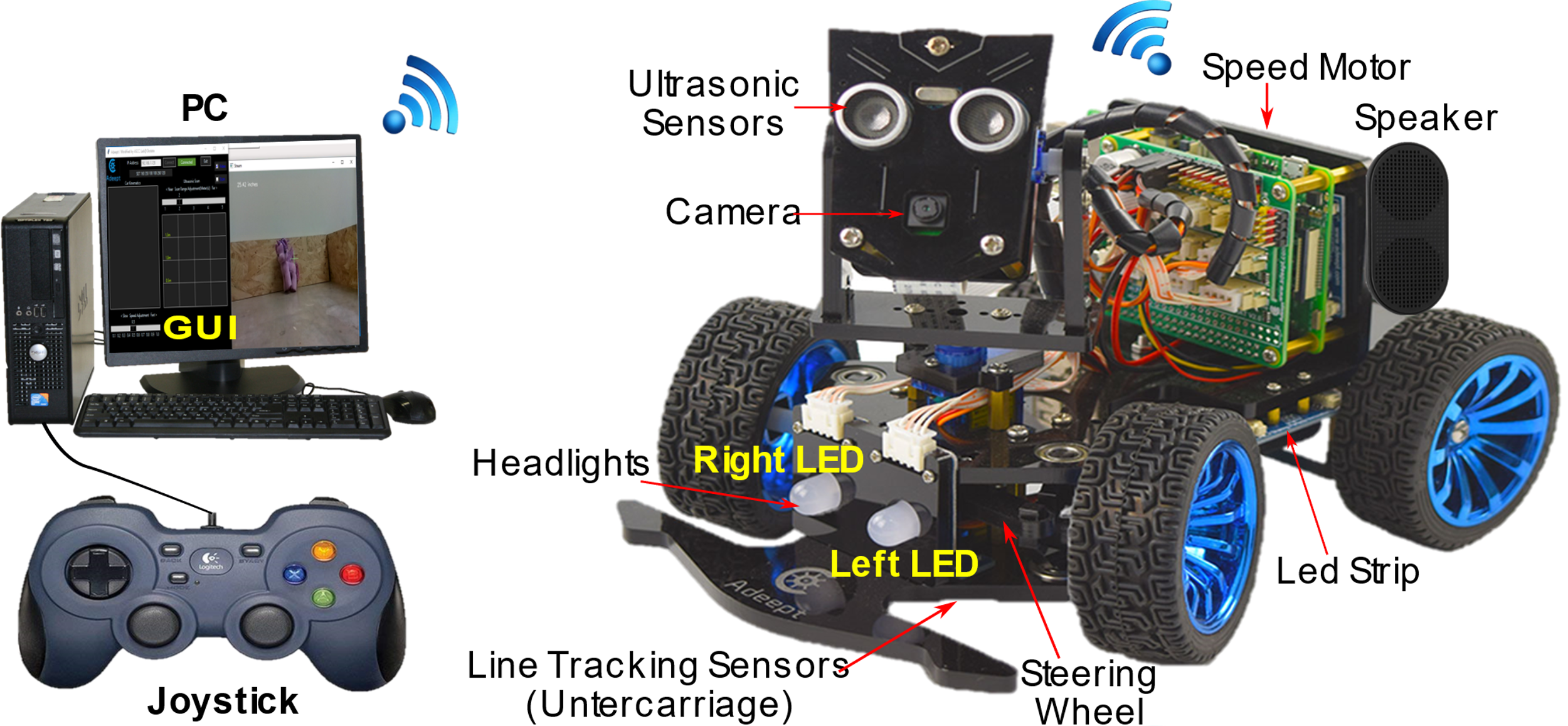
- Open VNC Viewer and Enter the IP address of your RPi



- Open IDLE3 and Test import commands



## 2. Hardware



## 3. Robot Software Architecture



## 4. Testing and Getting started

## 4.1 Calibration

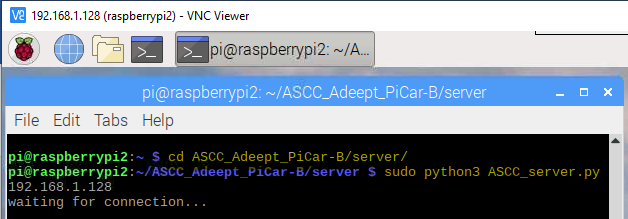
After assembling the robot, the servo motors of steering and pan-tilt unit need to be calibrated for correct positions. The steps of calibration are as follows:

1. Turn on the Robot

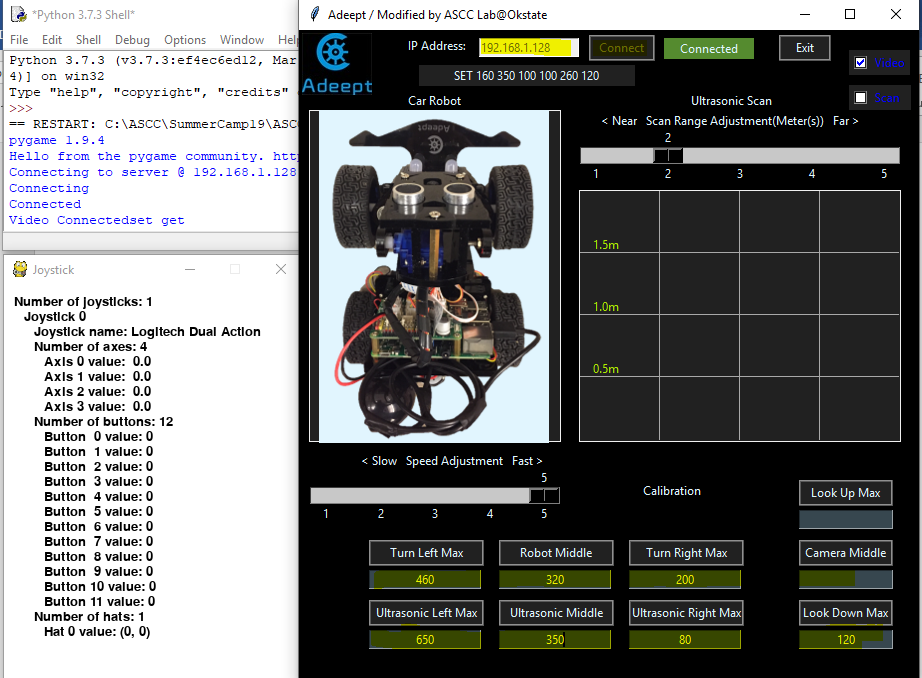
2. Run VNC Viewer on your PC to access the remote desktop of RPi

3. Start robot software (Server and Client) and Calibration

- Run ASCC\_server.py on RPi in Terminal



- Start IDLE or PyCharm on your PC and Run ASCC\_client.py



- Click Buttons in Calibration, if servo motors are not in right positions then input new value, click and check again. New values will be updated and saved to Set.txt file on RPi.

## 4.2 Remote Control

- Use the Joystick to slowly run the robot around and control the camera to look up, down, right and left.



|  |  |  |
| --- | --- | --- |
| **Pre-programed Buttons**  *(Do not change them)* | | |
| 1 | Right Ball | Moving Control |
| 2 | Left Ball | Steering Control |
| 3 | Hats | Head Control |
| 4 | LT | Reduce speed |
| 5 | RL | Increase speed |
| 6 | LB+Y | Take picture (Saved to C:/Photo/ |

## 5. Robot Programming with Python

## SECTION I – REMOTE CONTROL

### 5.1 Customize your Remote Control using a Joystick

|  |  |
| --- | --- |
| √ *Re-start ASCC\_client.py without connection to ASCC\_server.py. You can see the Joystick window (behind the GUI). Use joystick to see how the values of buttons are changed? You can learn the ID of each buttons, hat, or Axes*. |  |

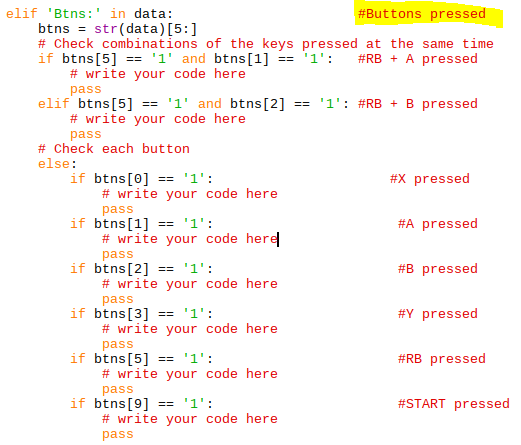
You can program or customize your remote control with the follow buttons and their combinations:

(START, X, Y, A, B, RB)

The Client will send a message to the Server if any button is pressed. The format of this messages is “Btns:X0X­1X2X3X4X5X6X7X8X9X10X11” where X0X­1X2X3X4X5X6X7X8X9X10X11 is value of buttons from 0 to 11. If only Button 2 is pressed, then X2=1, the other values are 0, and the message is “Btns:001000000000”

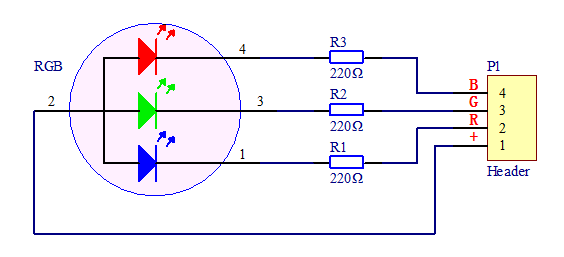
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value | X0 | X­1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 | X11 |
| Button | X | A | B | Y | LB | RB | LT | RT | BACK | START | None | None |

You can customize your control by calling a function you expect for above buttons and combinations in ASCC\_server.py. Find the Joystick control code in ASCC\_server.py as follows:



## 5.2 LED headlights (led.py)

This section you practice how to control the LED headlights. Each RGB LED headlight consists of three LEDs (a red, a green and a blue light) that can be controlled independently. The three-color LEDs can produce any color. (Ref: <https://wiki.adeept.com/pub:rgb_led>)

[](https://wiki.adeept.com/_media/pub:rgb_led:rgb-ledz.jpg?cache=)[](https://wiki.adeept.com/_media/pub:rgb_led:3z.png?cache=)

|  |  |
| --- | --- |
| 1. Open led.py and see how to write functions to turn on the LED headlights in different colors.  2. Open tutorial1.py, and run module (F5), and run function test\_led(). Watch the light and results printed out in Shell.  √ *Why the* ***red*** *and* ***pink*** *light were not turn on? Try to add your code to fixed that*.  3. Open ASCC\_sever.py and add your code to the Joystick control code to turn and mix the light of both LEDS using Buttons X, A, and B for blue, green, and red light, and Y for turn-off.  4. Run and connect ASCC\_server.py and ASCC\_client.py.  *√ Check your implementation with the joystick?* |  |

## 5.3 Play sound and Speech Synthesis

|  |  |
| --- | --- |
| 1. Open sound.py and see what functions use can use to play sound (wav or mp3).  2. Open tutorial1.py, and run module (F5), and run function test\_sound(). Listen carefully.  √ *Figure out why robot cannot play full voice synthesized by the third command? Try to add your code to fixed that.* |  |

## 5.4 LED strip (led\_strip.py)

|  |  |
| --- | --- |
| 1. Open led\_strip.py and see what functions you can use to display the led\_strip.  2. Run module (F5), and run function test() in Shell and see how the light is displayed. (**Note:** This function runs infinitely. You can press **Ctrl-C** to stop any time) |  |

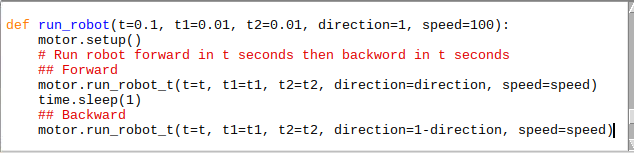
3. Write rainbow\_sound function to draw rainbow and play music (./music/rainbow1.mp3)

## 

## SECTION II – NAVIGATION CONTROL

## 1. Speed and Turn Control

Open motor.py and turn.py and see what functions you can use to control the robot speed and turn.



­a) Open tutorial1.py, and run module (F5), PLACE the robot on the ground with big enough space for running. Run function run\_robot() and see how far the robot can go.

b) Change speed = 50, 30, 10 and run the robot again.

c) Change t1 and t2, run the robot and compare how far the robot can go

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | How far the robot can go | |  |
| **t** | **t1** | **t2** | **Speed** | **Forward** | **Forward** | **Note** |
| 0.1 | 0.01 | 0.01 | 100 |  |  |  |
| 0.1 | 0.01 | 0.02 | 100 |  |  |  |
| 0.1 | 0.02 | 0.01 | 100 |  |  |  |
| 1 | 0.01 | 0.01 | 50 |  |  |  |
| 1 | 0.01 | 0.02 | 50 |  |  |  |
| 1 | 0.02 | 0.01 | 50 |  |  |  |
| 1 | 0.01 | 0.01 | 30 |  |  |  |
| 1 | 0.01 | 0.02 | 30 |  |  |  |
| 1 | 0.02 | 0.01 | 30 |  |  |  |
| 1 | 1 | 0 | 30 |  |  |  |
| 1 | 1 | 0 | 30 |  |  |  |
| 1 | 1 | 0 | 20 |  |  |  |

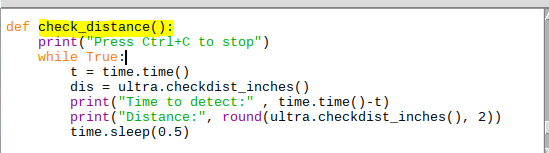
## 2. Obstacle Detection using ultrasonic sensors

- The robot can detect obstacles with ultrasonic sensors. The closer obstacle the faster can the robot detect.

- Open ultra.py and see what functions you can use.

def checkdist\_inches(): return the distance between the obstacle and the ultrasonic sensors

1. Run function check\_distance() in tutorial1.py. Slowly move robot closer an object or wall. Check the distance and time to detect.



1. Write your code for function avoid\_collision() in tutorial.py to slowly run the robot closer an object or wall and stop the robot about 30 inches before the object or wall. Then turn right or left.
2. Test the function*. Make sure you ready to secure the robot in case it hits the object.*

## 3 Line Detection with IR sensors

## 4. Object recognition using Camera

# SECTION III – GUIDELINES FOR THE COMPETION