

Unit code: CIS044-3

ADVANCED ROBOTICS

Assignment 2: Robot navigation and path planning

Unit Coordinator: Prof. Dayou Li and Phillip Hushani

Student Name: Raimonds Buls

Student ID: 1912944

Contents

**List of figures**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2

**Introduction**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

**Task Description** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

**Advanced Machine Learning and OpenCV**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5

**Linux and Virtual Machine workstation** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6

**Robotic Operating System (ROS)** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7

**Arduino**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9

**HC-05 Bluetooth module** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 10

**Artefact test results** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 11

**Conclusion**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 13

**References** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 14

**Appendix**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_15

**List of Figures**

1: ANYmal C quadruped robot with inspection head …………………………………………………………………3

2: Parameter’s values for training ……………………………………………………………………………………………..5

3: CNN model …………………………………………………………………………………………………………………..……….5

4: ROS package operating scheme…….…………………………………………………………………………………….7

5: Neural network training results …………………………………………………………………………………………11

6: OpenCV image processing script, integrated in ROS and publishing value of recognized traffic sign class………………………………………………………………………………………………………………..…….11

7: ROS running system with OpenCV publisher and Arduino subscriber……………….………………12

8: Paired Bluetooth module with ROS run test script with different errors...…….…………………12

**Introduction**

In the past few decades, autonomous robotic systems performed major steps in development. While, back in the time, the main focus was leaning towards military and space fields of research and use, nowadays huge aspect is going to industry and civilian needs. In contrast, revolutionary progress has been made from Tesla with their Tesla Autopilot technology, released in 2015, to provide self-driving capabilities for their cars (Tesla, 2022). Amazon company launched its Prime Air delivery with autonomous aerial vehicles (Drones) in 2016. Technology is still currently in development part, as various drone designs are being tested and Amazon claims to reach its high delivery standards to distribute packages to customers in 30 minutes or less (Amazon,2022). Another example can be seen from ANYbotics company, which designed ANYmal robot design for various industrial inspection tasks. Currently, robots successfully perform their duties in more than 50 service locations across Europe, whether it be train maintenance or certain areas in a chemical plant (Anybotics, 2022).

On the other hand, the most common product of robot design is the vacuum cleaner “Roomba”. Irobot company provides different models for sale and states that their robots can perform autonomous cleaning tasks by making a map of surrounded environment before. Also, the robot will move towards the charging station once the battery is low. Observation of the process and configuration can be done with a mobile phone application (Irobot,2022).

The examples listed above of the complex designs show the result of the hard work of a dedicated team of robotic engineers. Who was able to overcome challenges and push the fairly new concept of robotics technology to the next level. However, it requires an understanding of the fundamentals of the connection between software applications and hardware components and their integration and communication in the robot.



Figure 1: ANYmal C quadruped robot with inspection head. (Source: ANYbotics.com, 2022)

**Task Description**

In assignment two for this unit, I had to integrate advanced machine learning (ML) algorithm for road sign detection and recognition in a previously developed CIS117-2 Mechatronics unit mobile robot. To successfully achieve task requirements, the ML algorithm has to be processed with the use of the camera, which will serve as the input sensor.

Furthermore, the mobile robot has to navigate accordingly based on road sign image, to be precise, the required road signs were: Turn left and Turn right. Communication between processed images on camera and robot navigation will require the development of services in the robotic operating system (ROS).

The details below will provide an in-depth analysis of the solution designed, its challenges, extra features implemented, critical evaluation, and topics for discussion in software and hardware aspects.

**Advanced Machine Learning and OpenCV**

For image processing, I was considering using an already developed script from assignment one for this unit. However, that script was considered too basic and would not satisfy the needs for the given task. Taking this into consideration, I found an alternative solution by taking a traffic sign classification course from CVzone. The used script was based on Python programming language and for training and classification of traffic signs used convolution neural networks. Additionally, for training purposes were used 35000 images were divided into 43 different classes with the use of TensorFlow and Keras (CVzone, 2022). Parameters and activation function are fully adjustable, images below provide details for model used.

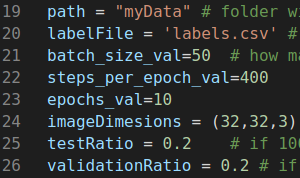
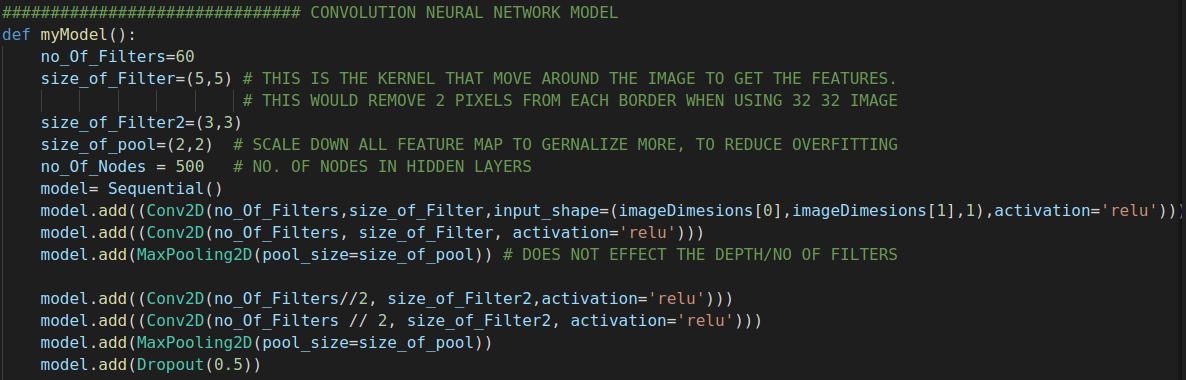
 

Figure 2: Parameter’s values for training Figure 3: CNN model

Interesting to note, that database with 35000 images is the from German Traffic Sign Recognition Benchmark. Which is widely used in traffic sign recognition topics and I had experience with it in assignment one. Also, is continuously being expanded and now contains over 50000 images (Benchmark.ini,2021).

The next step was to implement traffic sign recognition with the use of a camera. For this task, the OpenCV library was used as a modification of the script. That allows in real-time to perform road sign recognition and classify an image with displayed result and accuracy performance in the video frame. To provide accurate readings from image input, threshold parameter is to 0.90. That means, if probability value of image input will be higher than threshold, than road sign will be classified and results displayed.

Essential to mention, using all these features in one script would make the system flawed. As a result, we would have to run the script every time we would need to train, and that would take a significant amount of time if the training settings were very high, even on a powerful desktop computer. The implemented solution was achieved by separating scripts into two parts. The first part, which will be used for model training, and the second, which is most important, will be for real-time processing with a camera. When the training script section is finished, the model is saved and loaded at the start of the second part.

Initially, I tried using the Python pickle module to achieve this goal. That allows to implementation of binary protocols for serializing and de-serializing a Python object structure. However, there were issues with the loading model in the image processing script. The potential problem was in protocol versions for the Python version. The solution was found by using the TensorFlow Keras module. While provided, model. save("model.h5"), function saves results for a trained model in binary to be used further, there are other points to mention. According to Tensorflow developers, the model can be saved during the training period and continued later. This can be achieved by saving techniques used for the weight values, the model’s architecture, the model’s training configuration, and the optimizer and its state (Tensorflow, 2022). That feature can make the system robust when time and hardware resources are limited to finish the training process in one session.

As the database contains 43 classes of different traffic signs for image processing script 4 classes were selected. That includes required turn left and turn right and furthermore stop and ahead only. By doing so, mobile robots may reflect all four directions they can navigate, a stop sign can be considered as a return route.

**Linux and Virtual Machine workstation**

The further vital software component, in particular, the robotic operating system required Linux operating system rather than Windows. For this purpose, Ubuntu 18.04 version was installed, which provides a graphical user interface as Windows OS and is based on Linux. Worth mentioning, there are higher versions of Ubuntu available 20.04 and 21.10 according to developers’ information (Ubuntu,2022). But there were issues encountered during practical laboratory sessions in university, between Ubuntu 20.04 and ROS melodic. I decided to use recommended 18.04 stable version, rather than troubleshooting issues with an advanced one.

To efficiently use resources for artifact development at home and on campus, Ubuntu was emulated in the Virtual Machine 16 workstation. That allowed to replicate software environment on modern desktop station and laptop. VM workstation provides flexible options for system configuration, distributing computer resources. That being said, additionally RAM or either disc space can be expanded. However, during my experience with VM few issues came across. At first, my desktop station has 32GB of DDR4 RAM and VM operating there flawlessly, but the laptop model is old and with the use of 6GB of DDR3 RAM for Windows and VM, there were performance issues. Potentially, this can be solved by upgrading hardware or either installing Ubuntu directly on Windows OS. Lastly, Ubuntu was not able to detect the attached camera, however by the system it was recognized, and moreover to add on Windows OS camera Python script was working perfectly. The solution was found by changing USB port settings in VM configuration from 2.0 to 3.1. Troubleshooting this problem took a considerable amount of time, since it was so trivial and rarely encountered, that external internet resources were diverted to totally different areas.

**Robotic Operating System (ROS)**

The key aspect of software and hardware integration relies on ROS, however, what exactly is ROS and what it does do? Although ROS stands to be an operating system, rather it should be considered as a collection of software frameworks for robot development. Building a robot might be a complex task as there are many points to consider – motors, sensors, software, and batteries. All these things must work together seamlessly to accomplish the necessary task. ROS is one of the most popular, free, and open-source software that allows simplifying this process. It defines tools, components, and interfaces for building advanced robots. In general, the majority of robots are made up of actuators (things that move), sensors (things that read the world), and control systems (robot’s brain). With the ROS tools topics and messages, developers can build these components faster and easily connect them. These messages can be sent to different kinds of visualization and teleoperation tools. That allows working with simulated robots, making the testing and training part more effective. Also, ROS supports many common hardware components such as cameras, LIDARs, and motor controllers. Rather than reinventing the wheel, developers can focus on addressing the problem. ROS packages had over half-billion downloads in 2020 alone, that’s fascinating results. And make ROS an ideal platform not only for organizations but also for a startup (ROS,2022).

Important to understand how the system will perform overall. We want to Python script to perform image recognition with the use of a camera and then send the result to Arduino mobile robot to execute movement commands accordingly. For this purpose, there is a need to create Publisher and Subscriber nodes with communication between of same Ros topic and all this will be operating with the use of Ros Master.

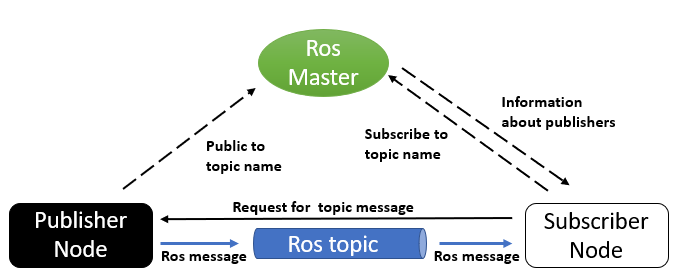


Figure 4: ROS package operating scheme (Source: <https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system(ros)/,2022>)

Firstly, in catkin created workspace I created a package, which will contain all necessary files. Worth noting, by creating a package with folders for further development, the system also creates a CMakeList.txt file. Mentioned file, is an essential configuration setup for required packages used and libraries. It allows created package to find catkin required components such as message\_generation, message\_runtime, roscpp, rospy, and std\_msgs. Which ones are crucial for the development of this artifact.

Afterward, the image processing script will be integrated into created Ros package, and with modifications implemented it becomes the publisher. Since the image recognition script will be running until canceled, there is no need to call the publisher himself. Each successfully recognized traffic sign image belongs to the class. This class number we will pass to the topic created in the Arduino subscriber. The created topic can contain different data types, either to be int8, float64, string, or even time. When it comes to choosing the right sensors, it is important to specify the type. Laser cutting is an example of a procedure in which precise sensor readings are extremely important for robotic arm movements. Also, the publisher contains the rate of how often data will be sent to subscribers and queue size. In order to avoid message drops, the setting should be based on the published rate (ROS.org, 2022).

Additionally, I made research in other ROS packages and was able to successfully launch TurtleBot3 in the Gazebo simulation environment. Wherewith the use of the Rviz packages for laser sensor and Gmapping, the robot was able to perform autonomous SLAM in simulation. On the other hand, the teleop package provides access for robot controls manually by using a keyboard. That provides a perfect example, of the complex robotic system created in ROS. Moreover, instead of launching nodes individually, all required nodes can be added to the ROS launch file and executed in a single command.

**Arduino**

The mobile robot developed for the previous unit, already had core components built-in and working and required modification part in Arduino coding. Important to mention of configuration part between the Linux system and Arduino. The robot is connected to the Linux with a USB cable and to avoid numerous issues by uploading any code, keen attention to certain details has to be made. In particular, in Arduino “Tools” section correct Arduino board has to be selected along with the processor model and most importantly port used. This can be tricky and certain machines use the ttyUSB0 port and others same as in my case, ttyACM0. More details about Linux ports can be reviewed by using the ls -l /dev command, where the camera port can be seen too as video0 or other, depending on devices connected. Also, there are limitations of the maximum number of Publishers/Subscribers in Arduino and depends on processing chip specifications. In the case of Atmega328p used in Nano, the limit is 25 (ROS,2022).

This can be important when building complex ROS structures with integration in the mobile robot, where Raspberry Pi or Jetson Nano would have much more computing processing power.

In order to make Arduino subscribers, ROS key components have to be included in Arduino code, in this case, ros.h and srt\_msgs with the same type of message as publisher was created. Code has to include NodeHandle and callback function for successful communication between publisher and subscriber.

Afterward, Arduino will read data from the subscribed message and that value is part of programmed movement commands. For instance, once the turn left road sign was successfully recognized on the camera, the publisher is sending the message of value 34, which reflects traffic sign class. Then, the Arduino subscriber receives that message and uses that value to perform the Turn left function. The function activates the right motor servo to move forward and the robot turns left, also the left blue LED is indicating the turning direction. For clarity, Python scripts code and Arduino are attached in appendix section.

**HC-05 Bluetooth module**

Lastly, I wanted to implement the HC-05 Bluetooth module in Arduino instead of using a USB cable. That would be a noticeable extra feature to match the state of the Mobile robot. For this purpose, besides the HC-05 module, TP-link Bluetooth 4.0 nano USB adapter was purchased too. For Bluetooth module integration I was following automaticaddison guide, where necessary steps were explained (Automaticaddison, 2022). Important to note on the hardware level, HC-05 operates with 5v however. Due to the fact that RXD operates at 3.3V, extra resistors have to be added to split input voltage in order to avoid any damage to components. Ideally, to match 3.3V by using Ohm’s law, there is a need 1k and 2k resistor. However, by using two 1k resistors left from the previous unit, the module seems to operate as supposed to be.

Then HC-05 was paired with Linux with the use of Blueman – Bluetooth manager. However, by trying to test Python script with the use of python-bluez in ROS, I was getting an error of 112 Host is busy. The encountered issue requires more time for troubleshooting and potentially will be resolved in the future, once the artifact will be demonstrated.

**Artefact test results**

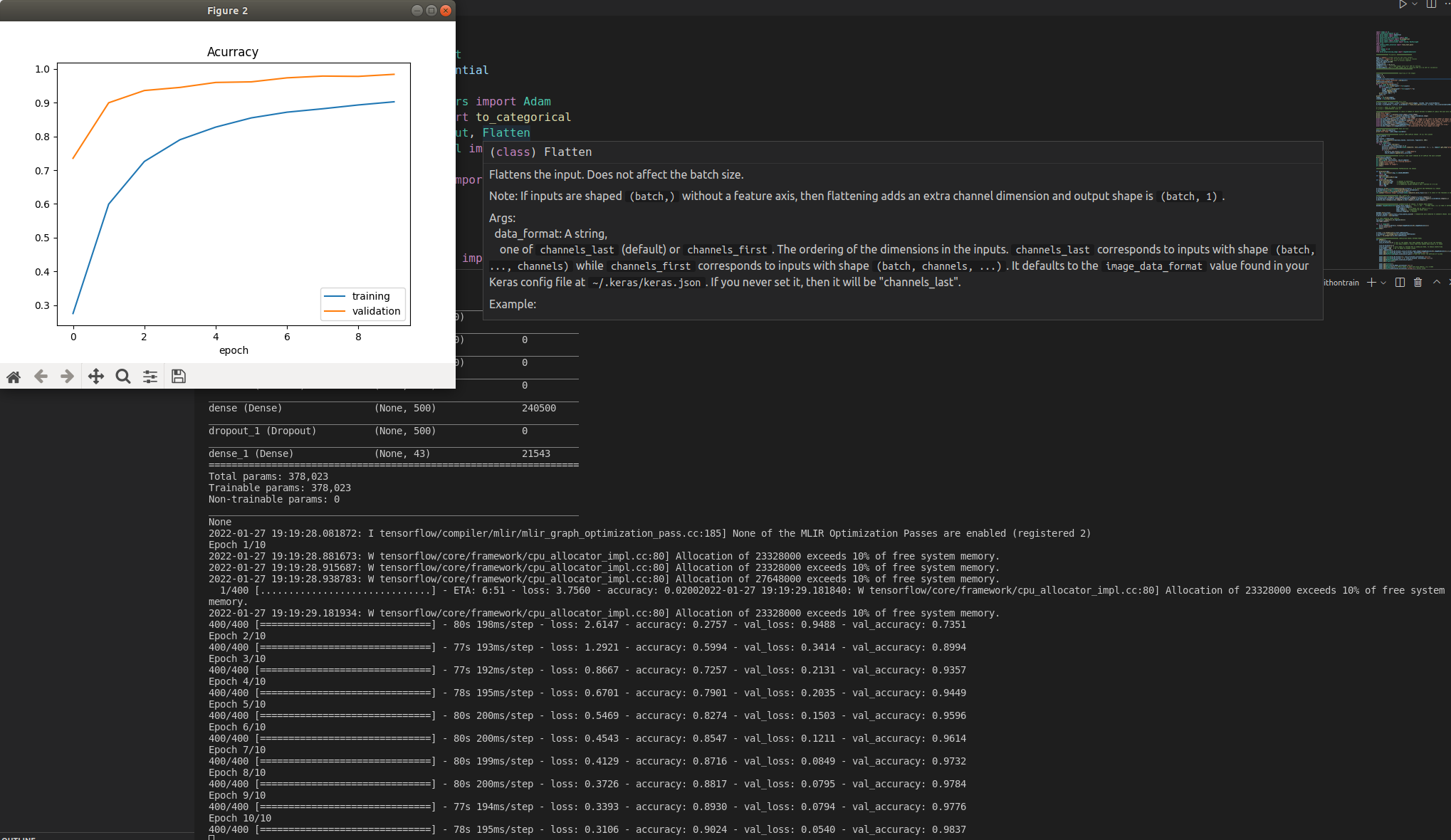


Figure 5: Neural network training results

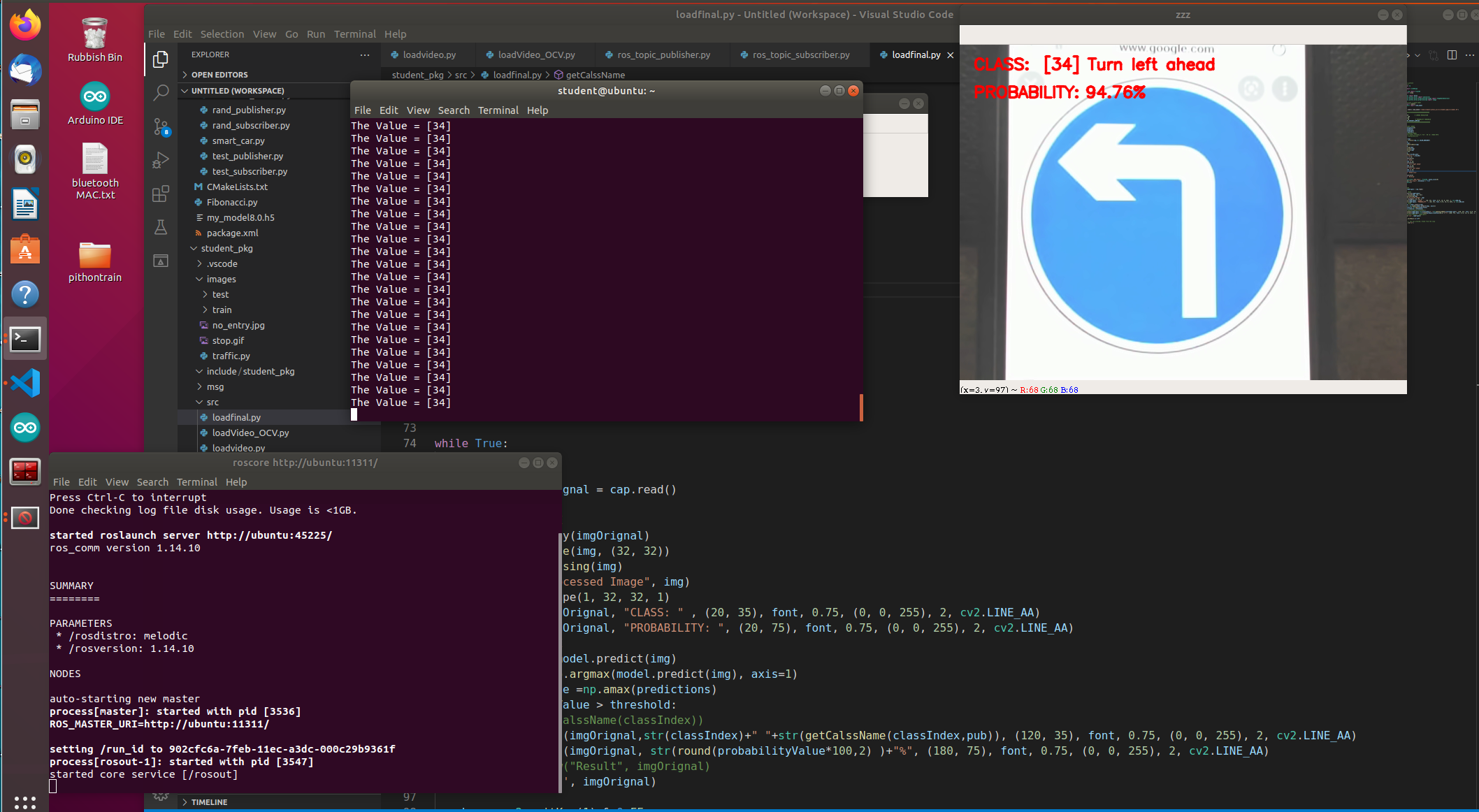


Figure 6: OpenCV image processing script, integrated in ROS and publishing value of recognized traffic sign class

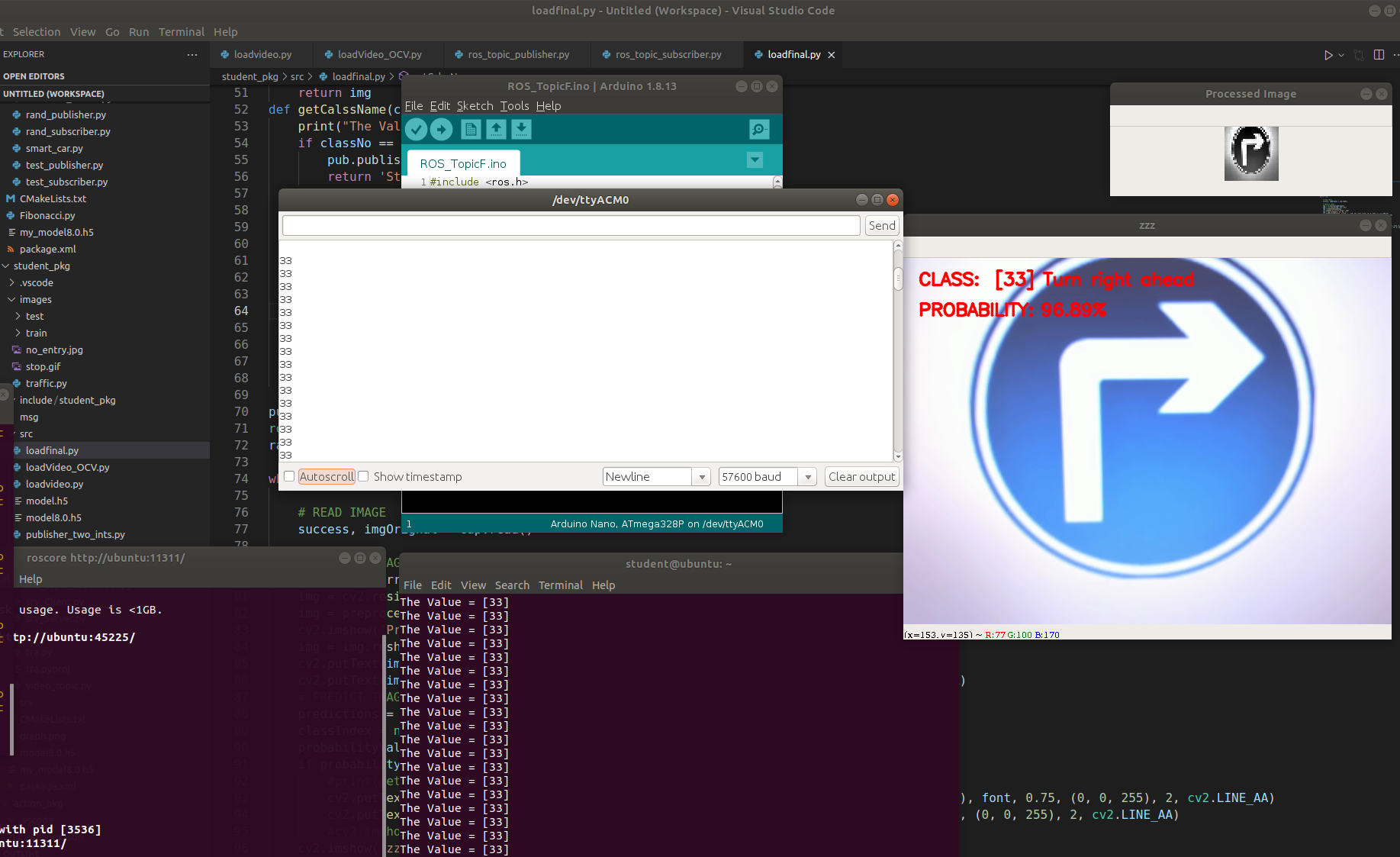


Figure 7: ROS running system with OpenCV publisher and Arduino subscriber.

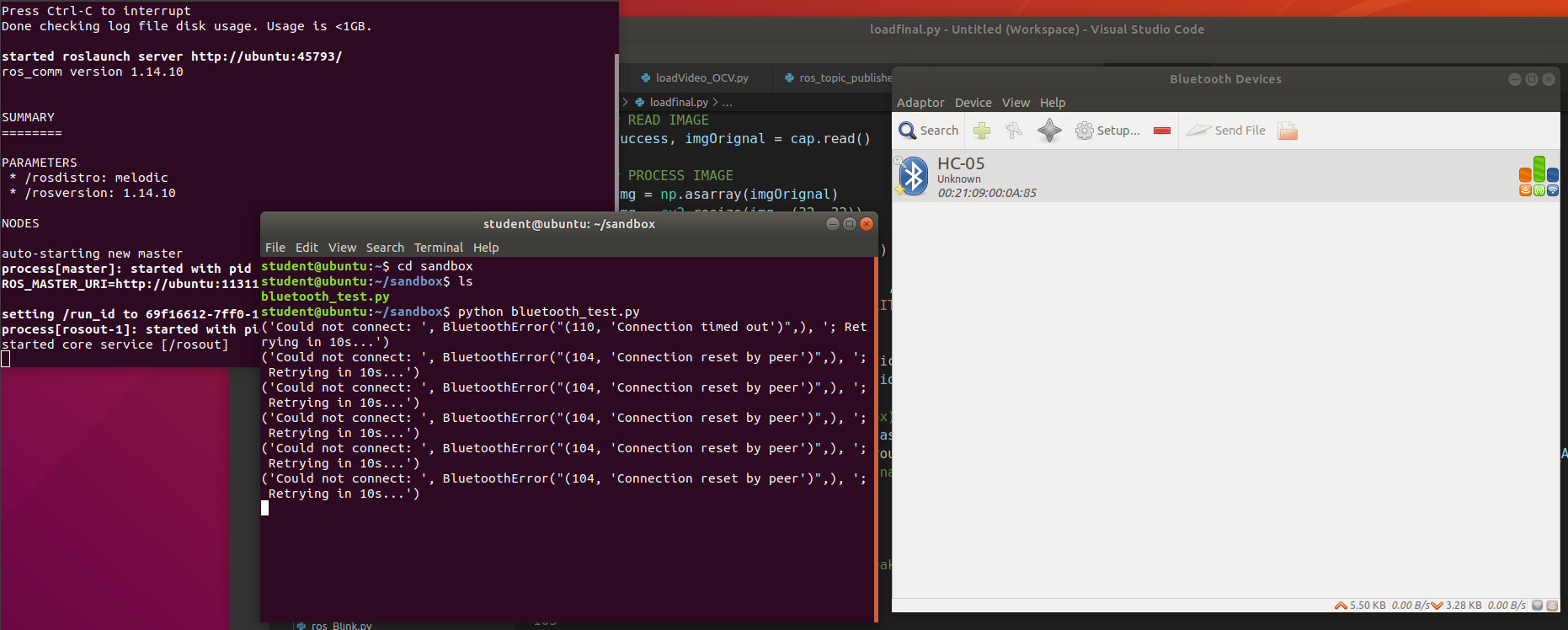


Figure 8: Paired Bluetooth module with ROS run test script with different errors.

**Conclusion**

Given task for this assignment was challenging and required knowledge acquired from previous units and most importantly in this one. Furthermore, to successfully achieve necessary goals additional research was done. Test results with training model provide 98% accuracy on average, and potentially this can go higher by tweaking more training parameters for image processing. On the other hand, OpenCV script validation works flawlessly once traffic sign image is position close to camera and is fully visible. There is accuracy drop with increased distance or either some parts of road sign being not fully visible. But this can be considered as analogy towards real life situation. In driver’s perspective eye sight matters to be fully confident on traffic sign recognition or either when it is covered by external obstacles, assumption would be made by its shape, color and visible borders. I consider ROS to be most essential part of the assignment. Experience gained with ROS tools and additional research made, allowed me to make further step in understanding and moving forward with my final year project. Where, TurtleBot3 robot will perform autonomous SLAM and visualize surrounded environment via LiDAR laser sensor. Additionally, ROS is considered leading platform in robotics development and vital knowledge of fundamentals gained benefits towards future job market potential in robotics field. In general, artefact development was going on steady phase, however troubleshooting on certain steps was taking massive amount of time. Searching solution for issues encounter from external resources was not all the time helpful – either provided information was leading to other topic or did not work in my particular case. This can be observed with HC-05 Bluetooth module implementation. Which requires me to dive deep and possibly sort out this issue in future. Besides that, key features were developed and further will be demonstrated on presentation.

**References**

1. Tesla 2022, *Tesla model 3*, Tesla corporation, viewed 02 January 2022,

< <https://www.tesla.com/en_gb/model3>>

2. Amazon 2022, *Prime air*, Amazon, viewed 05 January 2022,

< <https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011>>

3. Anybotics 2022, *Anymal robot,* Anybotics, viewed 06 January 2022,

< <https://www.anybotics.com/anymal-autonomous-legged-robot/>>

4. Irobot 2022, *robot cleaner “Roomba”,* Irobot LTD, viewed 16 January 2022,

< <https://www.irobot.co.uk> >

5. OpenCV, Traffic Sign Classification using OPENCV in real time, Vizdx LLC, viewed on 10 December 2021,

<<https://www.computervision.zone/courses/traffic-sign-classification/> >

6. Institut für Neuroinformatik, *The German Traffic Sign Recognition Benchmark*, Real-Time Computer Vision, viewed 12 December 2021,

<<https://benchmark.ini.rub.de/gtsrb_news.html>>

7. Tensorflow, *Save and load models*, Tensorflow.org, viewed on 15 December 2021,

<<https://www.tensorflow.org/tutorials/keras/save_and_load>>

8. Ubuntu, *Download Ubuntu Desktop*, Canonical Ltd., viewed on 18 November 2021,

<<https://ubuntu.com/download/desktop>>

9. ROS 2021, *What is ROS,* Open robotics, viewed 23 November 2021,

<<https://www.ros.org>>

10. ROS, *Publishers and Subscribers*, Open Robotics, viewed on 21 December 2021,

<<http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers>>

11. ROS, *Rosserial*, Open Robotics, viewed on 22 December 2021,

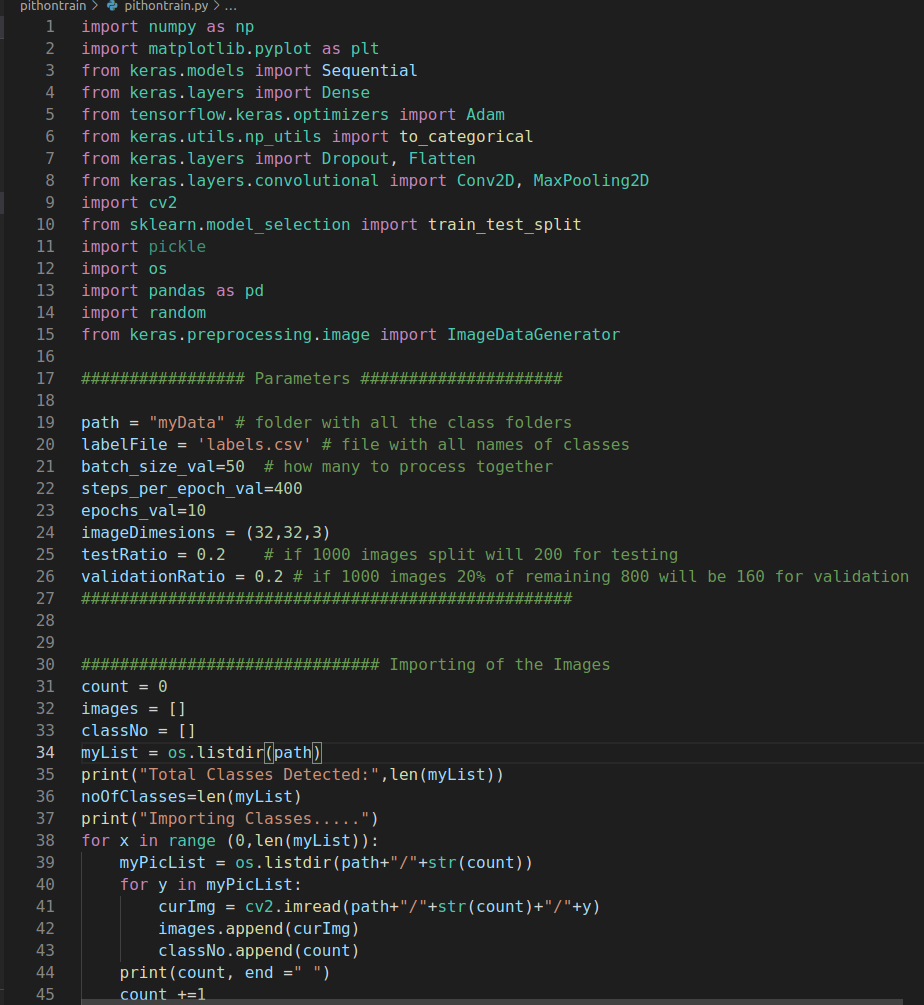
<<http://wiki.ros.org/rosserial>>

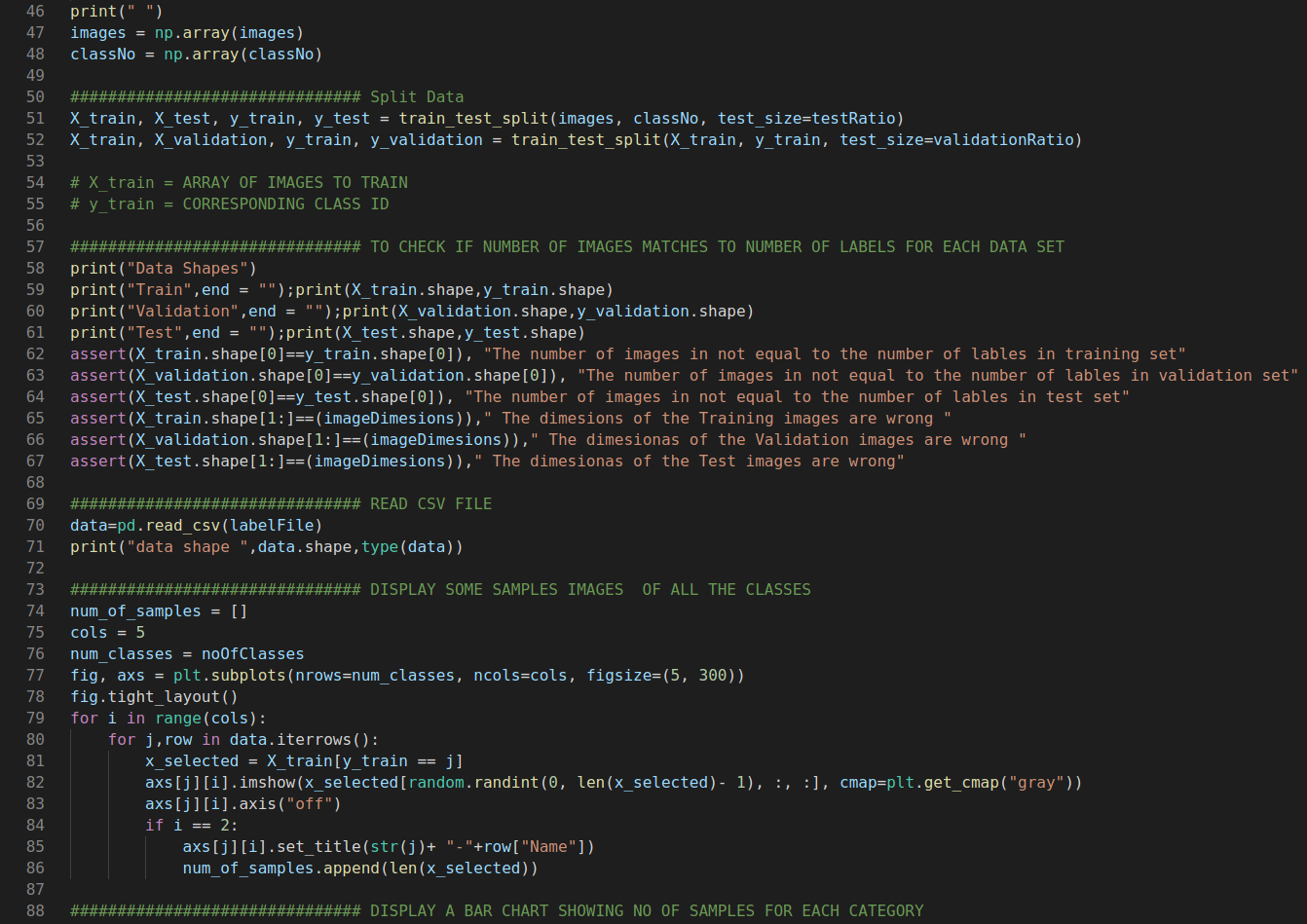
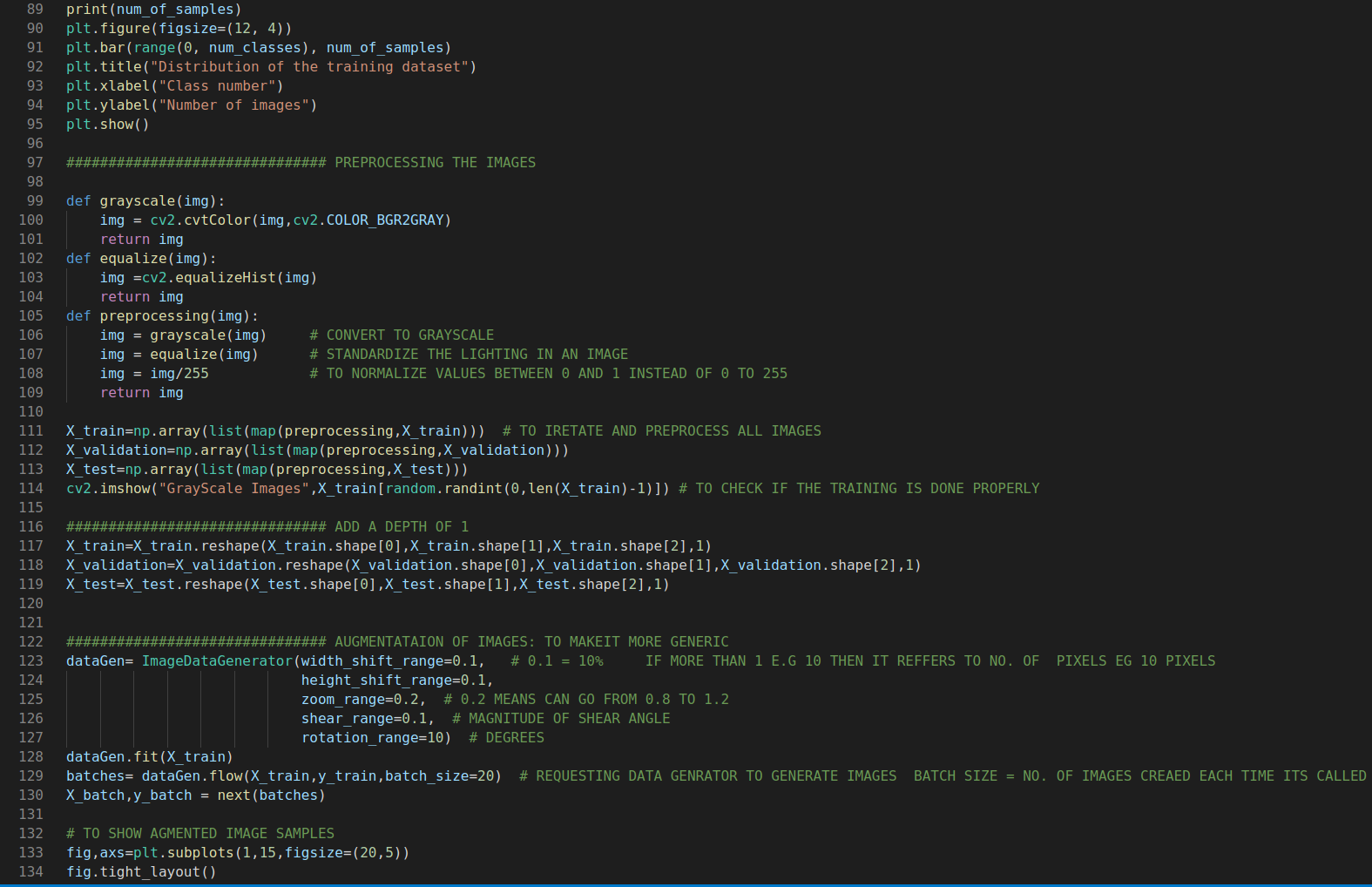
12. Automatic Addison, *How to make an autonomous wheeled robot using ros,* WordPress, viewed on 18 January 2022,

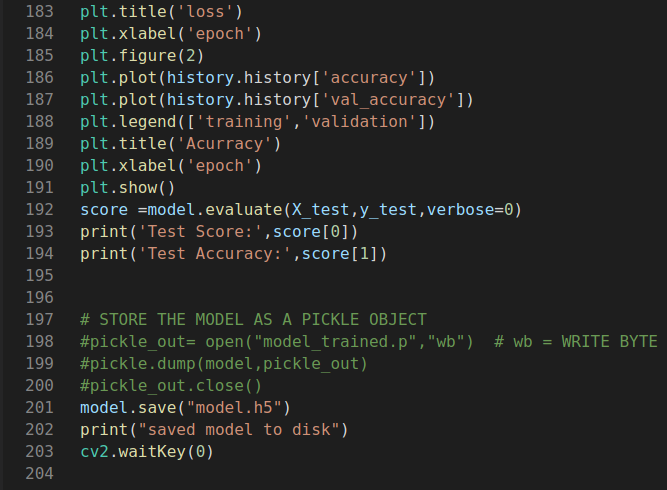
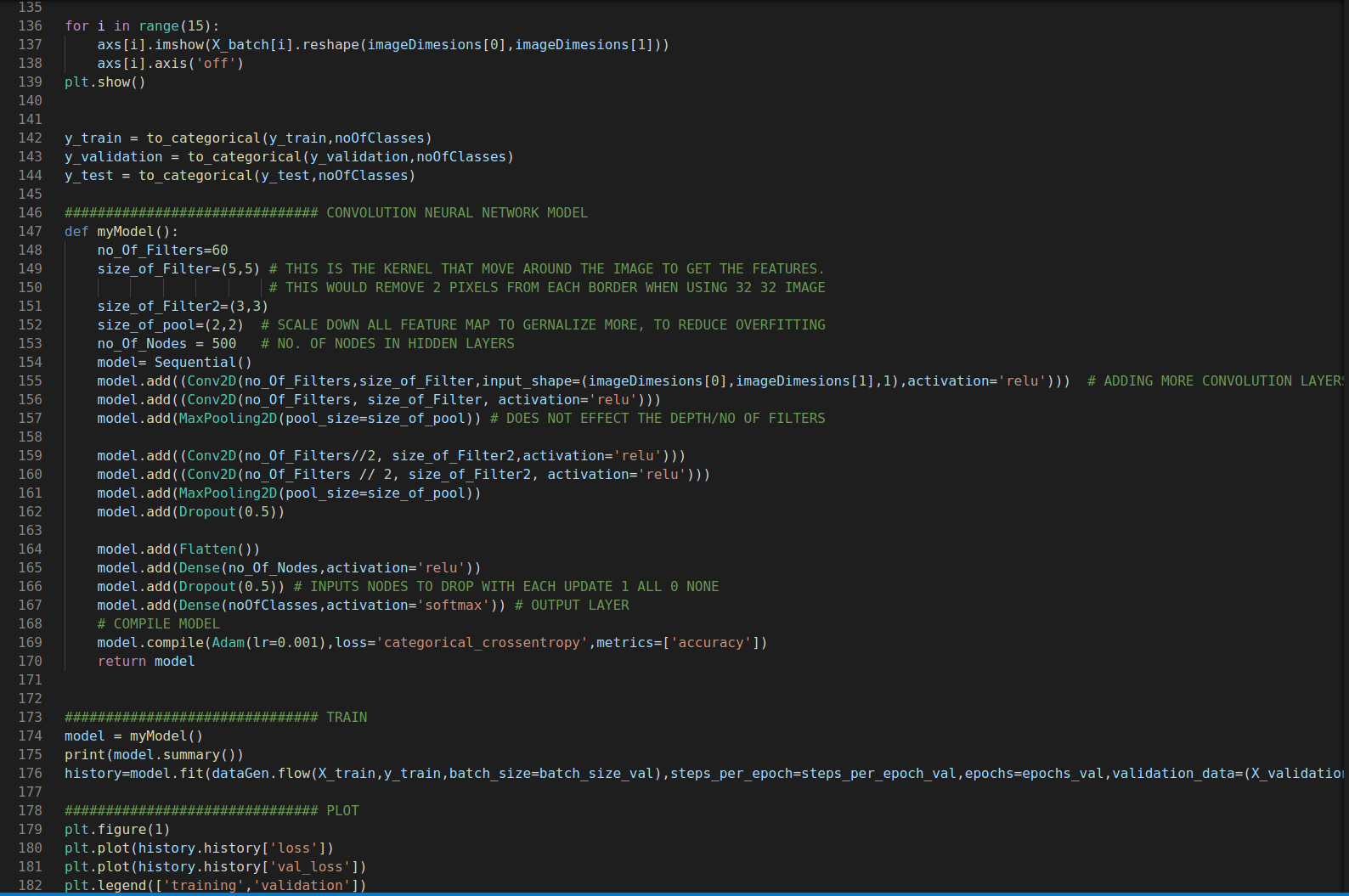
<<https://automaticaddison.com/how-to-make-an-autonomous-wheeled-robot-using-ros/#ultrasonic>>

**Appendix**

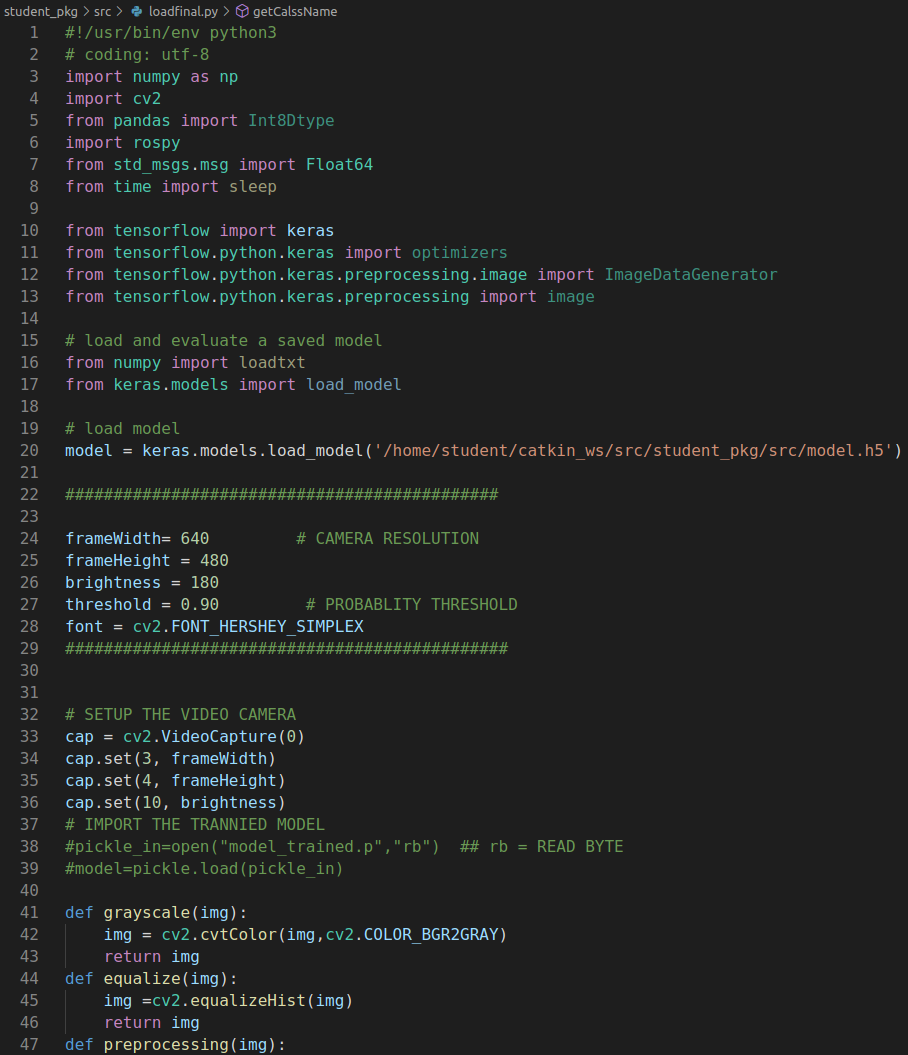
Referenced OpenCV script used for traffic sign training model (Modified)

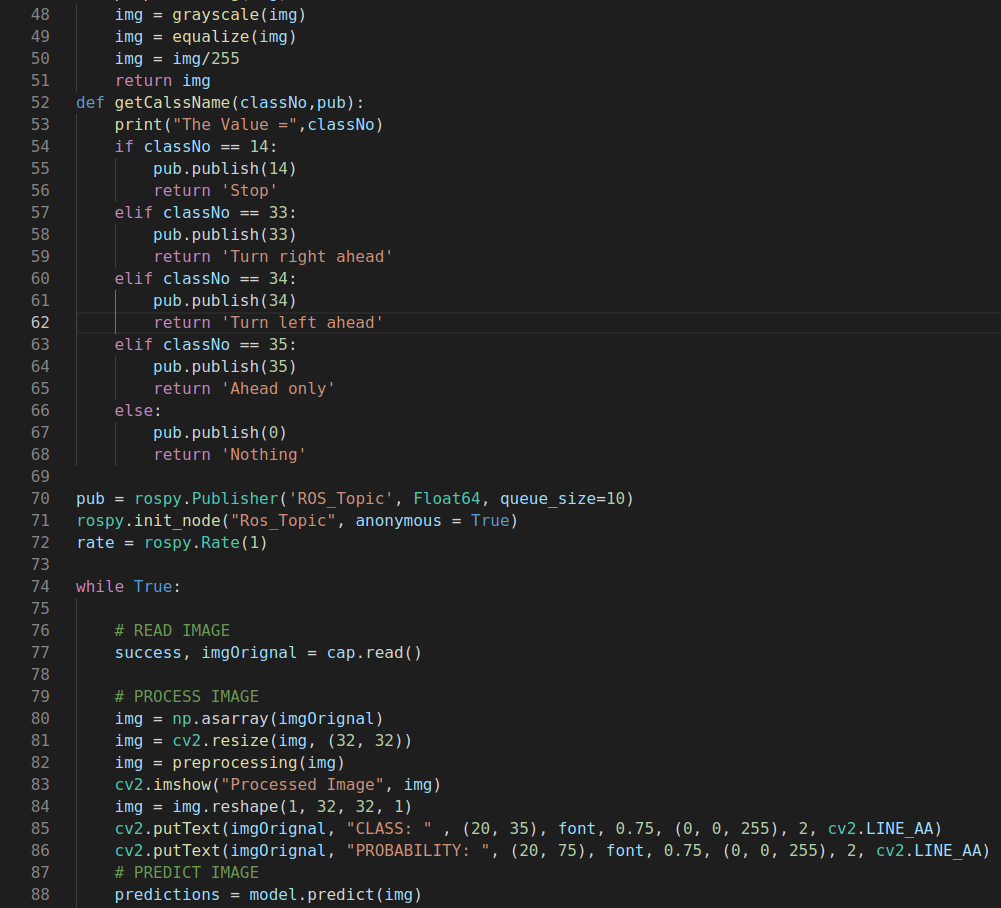


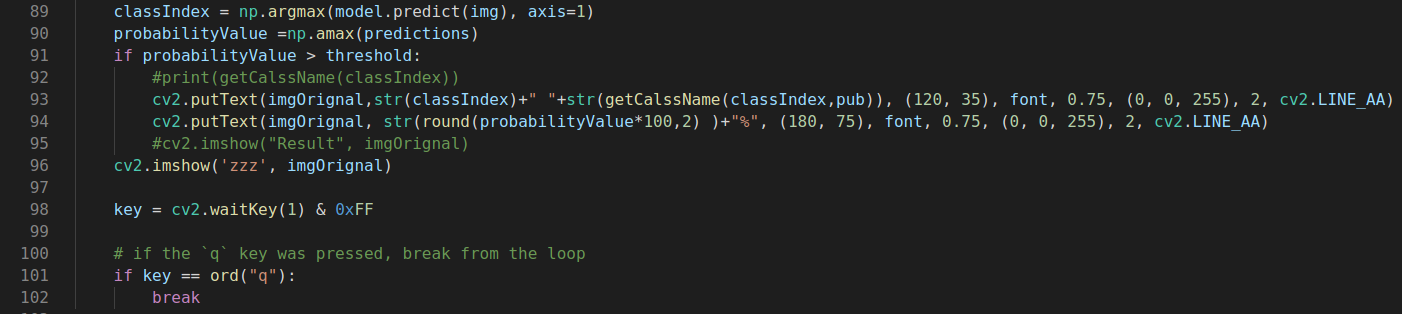




Final version of image processing script (ROS publisher):







Final version of Arduino script (ROS subscriber):

