PROJECT REPORT

ON

"AUTONOMOUS LANE FOLLOWING & TRACKING OF RC CAR"

Submitted in partial fulfilment of the requirements for the partial completion of

PROJECT FOR COMMUNITY SERVICE [16EC7DCPW1]

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

SUBMITTED BY:

Krishna Sai Tarun P	**1BM14EC046**
Kushal N	**1BM14EC047**
Mohammed Salman	**1BM14EC049**
Nikhilesh M	**1BM14EC062**

Under the Guidance of

Bhavana H T

(Academic Intern, ECE, BMSCE)

August - December 2017

Department of Electronics and Communication Engineering

B.M.S COLLEGE OF ENGINEERING

(Autonomous College Affiliated to Visvesvaraya Technological University, Belgaum)

Bull Temple Road, Basavanagudi, Bangalore-560019

DECLARATION

We undersigned students of final semester B.E in Electronics and Communication Engineering, BMS College of Engineering, Bangalore, hereby declare that the dissertation entitled "AUTONOMOUS LANE FOLLOWING & TRACKING OF RC CAR", embodies the report of my project work carried out independently by us under the guidance of Ms. Bhavana H T, Academic Intern, E&C Department, BMSCE, Bangalore in partial fulfilment for the award of Bachelor of Engineering in Electronics and Communication from Visvesvaraya Technological University, Belgaum during the academic year 2017-2018.

We also declare that to the best of our knowledge and belief, this project has not been submitted for the award of any other degree on earlier occasion by any student.

Place: Bangalore
Date:

Krishna Sai Tarun P	1BM14EC046
Kushal N	1BM14EC047
Mohammed Salman	1BM14EC049
Nikhilesh M	1BM14EC062

B.M.S COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the project entitled **"AUTONOMOUS LANE FOLLOWING & TRACKING OF RC CAR"** is a bonafide work carried out by **Krishna Sai Tarun P** (USN:1BM14EC046), **Kushal N** (USN:1BM14EC047), **Mohammed Salman** (USN:1BM14EC049) and **Nikhilesh M** (USN:1BM14EC062) in partial fulfillment for the partial completion of PROJECT FOR COMMUNITY SERVICE [16EC7DCPW1] during the academic year 2017-2018.

Ms. Bhavana H T	**Dr. G Poornima**	**Dr. K.Mallikharjuna Babu**
Academic Intern, ECE, BMSCE	HOD, ECE, BMSCE	Principal, BMSCE

External Examination: **Signature with date:**

1.

2.

Dedication

This work is dedicated to all the general people who are suffering due to traffic jams.

ABSTRACT

Self-driving vehicles represent a technological leap forward that can offer solutions to current transportation problems and dramatically change how people approach mobility. The development of self-driving cars or autonomous vehicles has progressed at an unanticipated pace. Fuelled by advances in artificial intelligence, cars are getting smart enough to begin to drive themselves. But autonomous vehicles will do more than change how we get around. They have the potential to dramatically reduce the number of car crashes, shrink carbon emissions, and provide mobility to people who can’t drive.

Driverless cars weren't possible until researchers adopted an AI technique known as deep learning, which relies on powerful GPUs, access to vast troves of data, and sophisticated algorithms for deep neural networks to solve complex problems. Deep learning is essential for autonomous vehicles because no one can write software that anticipates every possible scenario a self-driving car might encounter. With deep learning, the car’s computer can learn, adapt, and improve. This technology could transform the very interaction between society and its transportation system in ways that are scarcely imagined. Or, this technology could simply become a better cruise control for private autos or support current auto-oriented transportation and land use trends. Self-driving technology will likely be brought to market in the near future. Some of the world’s biggest companies, including GM, Google, BMW, and Volvo, are investing heavily in technologies designed to get humans out from behind the wheel. Building a Self-driving car is time consuming to set up everything, not very safe, less convenient and definitely not budget friendly. So the main idea of the project is to mimic the working of self-driving car using a remote controlled car and some of the open source software’s available for deep learning.

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We express profound gratitude to respected principal **Dr. K. Mallikharjuna Babu,** BMS College of Engineering for providing a congenial environment to work in. Our sincere gratitude to **Dr. G Poornima,** Head of the Department, Electronics and Communication Engineering for encouraging and providing this opportunity to carry out the project in the department.

We would like to thank our guide **Ms. Bhavana H T,** Academic Intern, Department of ECE who helped us in all the ways to carry out the project work. She stood beside and guided us in every step.

We would like to share the joy of completing the project to all the team members of NXP lab**.** We thank all our professors for providing the basic knowledge without which this project wouldn't have been possible. Last but not the least we thank our family and friends, who made their valuable support compelled us to maintain a standard throughout our endeavour.

-

Krishna Sai Tarun P

Kushal N

Mohammed Salman

Nikhilesh M

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction:

Autonomous or Self-driving cars are cars that ‘learn’ the way we humans interact with acceleration, brakes, turning, stopping for pedestrians and cars in a signal. By the help of machine learning these cars will be capable to take decisions based on how a human would. The difference of introduction of such cars will be a major reason for smooth flow of traffic because of perfected examples that will be trained over and over and then will be able to adapt by themselves (the cars) to find their way. Decision making will be much superior compared to human decision making due to the quick computational power that we have currently.

These automatic cars can be used for various purposes like transport and other applications. A machine cannot be completely trusted and it’s better to know about the vehicle’s whereabouts during an application. Tracking the vehicle using a GPS and a GSM system enables us to monitor the vehicle’s performance in an appropriate manner. In cases when the vehicle loses it way, the tracking system can be used to track the vehicle and make the required corrections for proper functioning.

1.2 Area of Application

1.2.1 Traffic and Safety

Traffic collisions (and resulting deaths and injuries and costs), caused by human errors, such as delayed reaction time, tailgating, rubbernecking, and other forms of distracted or aggressive driving should be substantially reduced. Additional advantages could include higher speed limits, smoother rides and increased roadway capacity and minimized traffic congestion, due to decreased need for safety gaps and higher speeds.

1.2.2 Welfare

Autonomous cars could reduce labour costs, relieve travellers from driving and navigation chores, thereby replacing behind-the-wheel commuting hours with more time for leisure or work and would lift constraints on occupant ability to drive, distracted and texting while driving, intoxicated, prone to seizures, or otherwise impaired. For the young, the elderly, people with disabilities, and low-income citizens, autonomous cars could provide enhanced mobility. The removal of the steering wheel along with the remaining driver interface and the requirement for any occupant to assume a forward-facing position would give the interior of the cabin greater ergonomic flexibility. Large vehicles, such as motor homes, would attain appreciably enhanced ease of use.

1.2.3 Related Effects on Community

By reducing the (labour and other) cost of mobility as a service, autonomous cars could reduce the number of cars that are individually owned, replaced by taxi/pooling and other car sharing services. This could dramatically reduce the need for parking space, freeing scarce land for other uses. This would also dramatically reduce the size of the automotive production industry, with corresponding environmental and economic effects. Assuming the increased efficiency is not fully offset by increases in demand, more efficient traffic flow could free roadway space for other uses such as better support for pedestrians and cyclists.

1.3 Problem Definition

Autonomous cars, itself has become a vast subject in recent times. Hence to keep up with the latest technologies and innovations in real world, this project has been selected. Yes, making any system autonomous itself is a difficult task, Autonomous car is even highly difficult task, time consuming and of high cost. So the main idea of the project is to mimic the working of self-driving car using a remote controlled car and some of the open source software's available for deep learning.

1.4 Objectives of the Project

The main objectives of project are as follows:

- Streaming continuous data of lane using Pi camera and Raspberry Pi.
- Use python and open CV libraries to create an autonomous module by collected data.
- Modify a RC car to self-drive on a given lane.
- Track the location of the RC car.

CHAPTER 2: LITERATURE SURVEY

Autonomous Driving has been said to be the next big disruptive innovation in the years to come. Considered as being predominantly technology driven, it is supposed to have massive societal impact in all kinds of fields. According to Marlon G. Boarnet (Ross, 2014, p. 90), a specialist in transportation and urban growth at the University of Southern California “Approximately every two generations, we rebuild the transportation infrastructure in our cities in ways that shape the vitality of neighbourhoods; the settlement patterns in our cities and countryside; and our economy, society and culture” and as many believe, autonomous driving cars are this new big change everyone is talking about. Leading not only to high impact environmental benefits such as the improvement of fuel economy [1] , through the optimization of highways the reduction of required cars to only 15% of the current amount needed (Ross, 2014), and platoon driving that would save to 20-30% fuel consumed, but also leading to societal aspects such as immense productivity gains while commuting, decline on the accident and death tolls considered as the eight highest death cause worldwide in 2013 (World Health Organization, 2013), stress reduction (Rudin-Brown & Parker, 2004a; Stanton & Young, 2005), and the decline of parking space to up to ¼ of the current capacity[2]. It would also, according to a study by Morgan Stanley (2013) lead to an average 38 hours reduction of commuting time per individual per year as well as saving the US economy alone 1.3 trillion dollars per year, creating a shift on the possibilities and different applications, developing completely new markets, partnerships and possible business models [3]. That will change the society as we know it.

All the benefits of course, and not only to mention the Technological difficulties do not come without a certain amount of challenges, complications and necessary changes in current systems to work. By now several States in the US have passed laws permitting autonomous cars testing on their roads (Walker S., 2014). The National Highway Traffic Safety Administration in the United States (2013) provides an official self-driving car classification dividing into No-Automation (Level 0), Function-specific Automation (Level 1), Combined Function Automation (Level 2), Limited Self-Driving Automation (Level 3) and Full Self-Driving Automation (Level 4). Europeans have also started modifying the Vienna Convention on Road Traffic and the Geneva Convention on Road Traffic to be able to adapt this new technology, but legal issues and doubt still arise as one of the main concerns of discussion. Some of the main issues surrounding the autonomous driving field found throughout the literature and the web are; test and standard set for critical event control, how to deal with the requirement for a ‘driver’, ownership and maintenance, civil and criminal liability, corporate manslaughter, insurance, data protection and privacy issues [4]

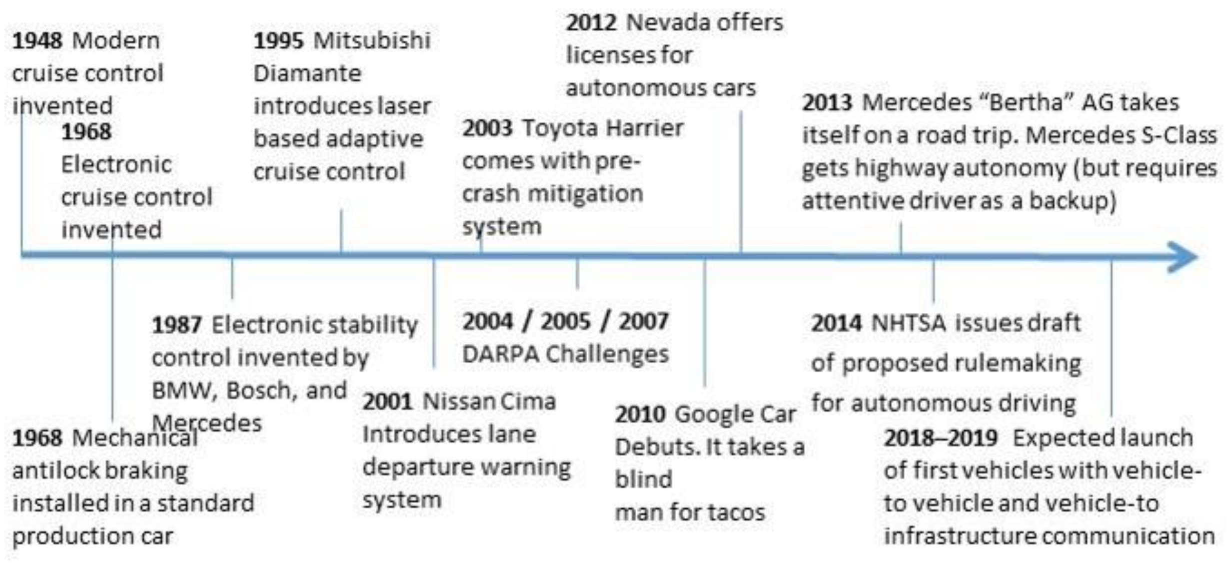

1948 Modern cruise control invented
1968 Electronic cruise control invented
1995 Mitsubishi Diamante introduces laser based adaptive cruise control
2003 Toyota Harrier comes with pre-crash mitigation system
2012 Nevada offers licenses for autonomous cars
2013 Mercedes "Bertha" AG takes itself on a road trip. Mercedes S-Class gets highway autonomy (but requires attentive driver as a backup)
1968 Mechanical antilock braking installed in a standard production car
1987 Electronic stability control invented by BMW, Bosch, and Mercedes
2001 Nissan Cima Introduces lane departure warning system
2004 / 2005 / 2007 DARPA Challenges
2010 Google Car Debuts. It takes a blind man for tacos
2014 NHTSA issues draft of proposed rulemaking for autonomous driving
2018–2019 Expected launch of first vehicles with vehicle-to vehicle and vehicle-to infrastructure communication

Figure 1: Sixty-five years of automotive baby steps.

Having a closer look at the history of Autonomous Driving, as explained in the IEEE Spectrum in Figure 1 it can be observed that the technological development and main milestones of the autonomous driving field started already a few decades ago. Leading to a vast analysis of some semi-autonomous features, development of present technologies and understanding on the future problematic while focusing soon in the connected car. [5]

Relevance of customer perspective

Autonomous cars are not that far away, for example Audi and Mercedes have announced almost being ready from production in highly automated features. As a reflection of the daily news, we can steadily see how this technology manages to get closer to be in our everyday life, with examples of cars driving a blind man for tacos already on 2012 (Google, 2012) [6], coast to coast trips, an Italy to China trip and 700,000 miles already travelled by Google (Urmson , 2014). But main automakers in the race such as Audi, BMW, Cadillac, Ford, General Motors, Jaguar, Land Rover, Lincon, Mercedes-Benz, Nissan, Tesla and Volvo, are trying to integrate it slowly to their models despite the fairly readiness of the technology. This can be interpreted as a futile attempt to keep this totally disruptive technology under control and to have overall slower customer integration, but this old model will prove to be not good enough due to the magnitude and impact of this technology. [8]

According to a survey with more than 200 experts on autonomous vehicles by the IEEE (2014), the world's largest professional association for the advancement of technology, the three biggest obstacles to reach the mass adoption of driverless cars are: legal liability, policymakers and customer acceptance, while the following three; cost, infrastructure and technology are less of a problem. [9]

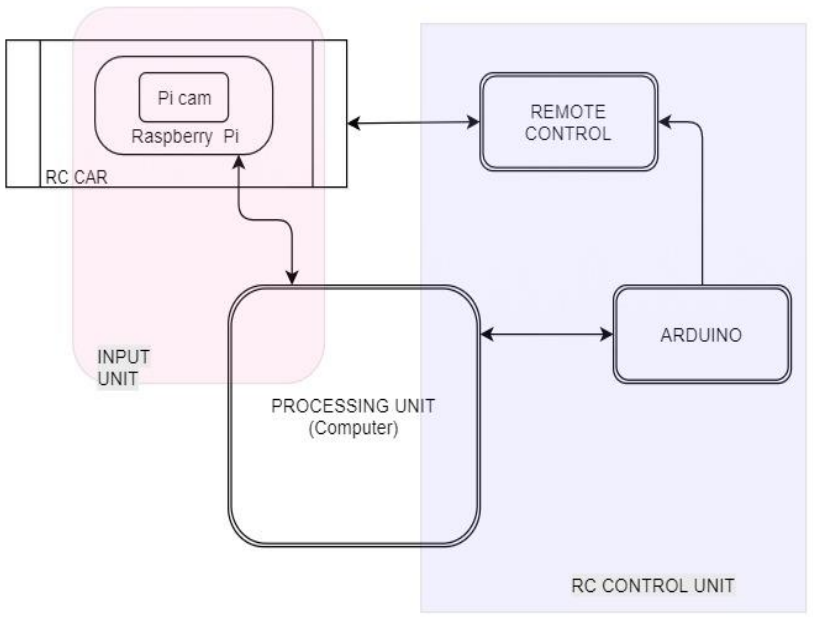
In the modern world of today, where technology prevails and scientific researches present a new era of discoveries and provide numerous luxuries to the common man; one also has to suffer the worse consequences of this advancement in technology. Just as the first gun was invented for the security of mankind, the criminal minded people used it for harming their own brethren. Similarly new & alarming technological methods have been adopted by the same people to harm peoples' comfort and have become a nuisance for the mankind. So the need of more security and safety has increased more than ever before. One of the major areas where security is needed round the clock is Automobiles. This is a major industry of the world and cars need to be secured otherwise they are packed gifts for the criminals. In a widely populated city, take Karachi for example, where cars are robbed or hijacked in huge numbers daily; the need of car security systems has become inevitable.[7] The need for security is one of the most important considerations in today's innovative world. Tracking of an autonomous car is one of the many ways to ensure that it is on the right path towards its destination. This tracking mechanism for security has been employed and has been evolving from a long time. The evolvement first started from the stages of development of the Global Positioning System (GPS). Today the GPS has developed high accuracies, but still has its problems for indoor positioning. A GSM network is the architecture used in the 2-G systems and a very widely used network though there have been advancements in UMTS and LTE.

Tracking till date (in most of the cases) has been all about interfacing a GSM and GPS module separately to a Micro-Controller. The cost of these required modules has been high and the advancements in technologies have been trying to make these modules available at cheaper rates. The need for security and tracking mechanism has pushed designers to come up with chips that are solely meant for this particular application. This has helped manufacturers to produce cost effective products using these chips as they are restricted to a specific application. SIMCom is one of the leading companies in designing GSM chips. SIM808, SIM908, SIM868 are a few chips that were designed solely for tracking and other related applications. SIM908 was a widely used chip a few years ago but the company stopped the production of the 900 series which led to designing of more advanced and better chips. SIM808 is a chip where the GSM/GPRS technology has been integrated with the GPS on the same IC. SIM808 is a new chip compared to all the chips that have been in market and its cost effectiveness makes it a worthy component for a cost effective tracking system.

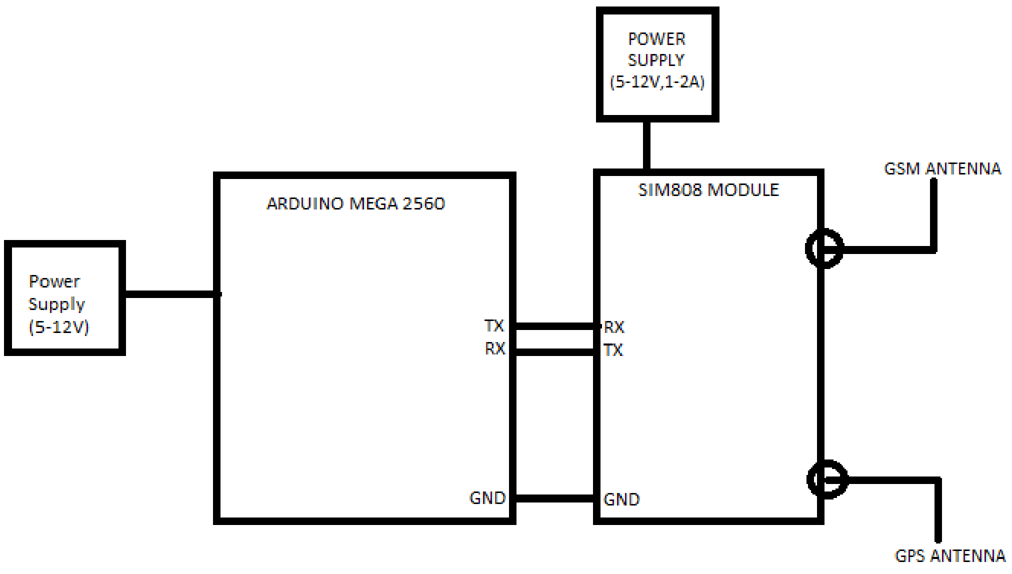
CHAPTER 3: METHODOLOGY AND IMPLEMENTATION

3.1 Block Diagram

a) Lane Following



b) Tracking



The system consists of four subsystems: input unit (Pi camera), processing unit (Raspberry PI & computer) RC car control unit and tracking unit.

Input Unit

A Raspberry Pi board (model B+), attached with a pi camera module is used to collect input data. Programs run on Raspberry Pi for streaming colour video data to the computer via local Wi-Fi connection. In order to achieve low latency video streaming, video is scaled down to QVGA (320×240) resolution.

Processing Unit

The processing unit (computer) handles multiple tasks: receiving data from Raspberry Pi, neural network training and prediction (steering) and sending instructions to Arduino through USB connection.

TCP Server

A multithread TCP server program runs on the computer to receive streamed image frames from the Raspberry Pi. Image frames are converted to gray scale and are decoded into numpy arrays.

Neural Network

One advantage of using neural network is that once the network is trained, it only needs to load trained parameters afterwards, thus prediction can be very fast. Only lower half of the input image is used for training and prediction purposes.

RC Car Control Unit

The RC car used in this project has an on/off switch type controller. When a button is pressed, the resistance between the relevant chip pin and ground is zero. Thus, an Arduino board is used to simulate button-press actions. Four Arduino pins are chosen to connect four chip pins on the controller, corresponding to forward, reverse, left and right actions respectively

Tracking Unit

This project uses only one GPS device and two-way communication is achieved using a GSM modem. GSM modem with a SIM card used here implements the same communication technique as in a regular cell phone.

3.2 Project Flow

A scaled down version of self-driving system is implemented using a RC car, Raspberry Pi, Arduino and open source software. The system uses a Raspberry Pi with a camera as inputs, a processing computer that handles steering, and an Arduino board for RC car control.

Dependencies

- Raspberry Pi:
    - Picamera
- Computer:
    - Numpy
    - OpenCV
    - Pygame
    - PySerial

About

- raspberry_pi
- arduino/
- computer/
    - test/
        - ***stream_client.py***: stream video frames in jpeg format to the host computer
        - ***rc_keyboard_control.ino***: acts as a interface between rc controller and computer and allows user to send command via USB serial interface
    - training_data/
        - training data for neural network in npz format
    - training_images/
        - saved video frames during image training data collection stage
    - mlp_xml/

        - trained neural network parameters in a xml file

    - ***collect_training_data.py***: receive streamed video frames and label frames for later training
    - ***mlp_training.py***: neural network training
    - ***rc_driver.py***: a multithread server program receives video frames and allows RC car driven by itself along the lane

How to drive

1. **Flash Arduino:** Flash *"rc_keyboard_control.ino"* to Arduino and run *"rc_control_test.py"* to drive the rc car with keyboard (testing purpose).
2. **Collect training data and testing data:** First run *"collect_training_data.py"*. User presses keyboard to drive the RC car, frames are saved only when there is a key press action. When finished driving, press "q" to exit, data is saved as an npz file.
3. **Neural network training:** Run *"mlp_training.py"*, depend on the parameters chosen, it will take some time to train. After training, model will be saved in "mlp_xml" folder
4. **Self-driving in action**: First run *"rc_driver.py"* to start the server on the computer and then run *"stream_client.py"* and *"ultrasonic_client.py"* on raspberry pi.

3.3 Hardware Architecture

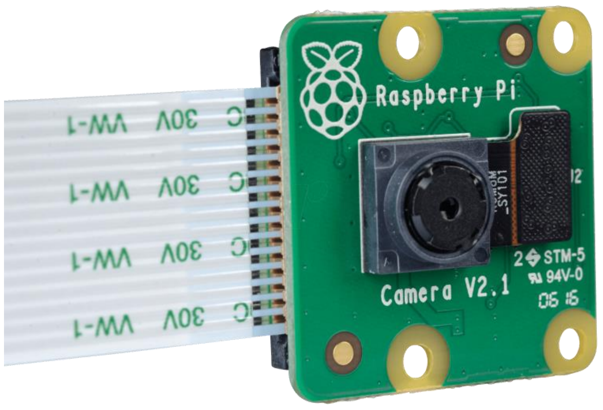
3.3.1 Components Description

Raspberry Pi

Raspberry Pi is an ARM based credit card sized SBC(Single Board Computer) created by Raspberry Pi Foundation. Raspberry Pi runs Debian based GNU/Linux operating system Raspbian and ports of many other OSes exist for this SBC. With on-board WiFi / Bluetooth support and a 64bit improved Processor.

Raspberry Pi Camera

The Raspberry Pi Camera Module v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSI interface, designed especially for interfacing to cameras.



Remote Control car

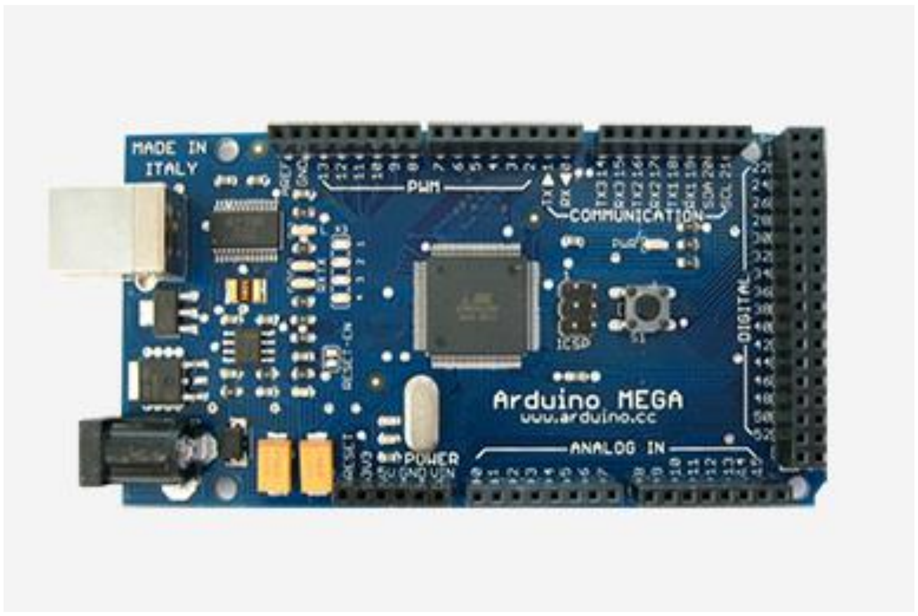
Radio-controlled cars use a common set of components for their control and operation. All cars require a transmitter, which has the joysticks for control, or in pistol grip form, a trigger for throttle and a wheel for turning, and a receiver which sits inside the car. The receiver changes the radio signal broadcast from the transmitter into suitable electrical control signals for the other components of the control system. Most radio systems utilize amplitude modulation for the radio signal and encode the control positions with pulse width modulation. Upgraded radio systems are available that use the more robust frequency modulation and pulse code modulation. Recently however, 2.4 GHz frequency radios have become the standard for hobby-grade R/C cars. The radio is wired up to either electronic speed controls or servomechanisms which perform actions such as throttle control, braking, steering.

ARDUINO MEGA 2560

The Arduino Mega is a microcontroller board based on the ATmega1280. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The ATmega1280 has 128 KB of flash memory for storing code (of which 4 KB is used for the boot loader), 8 KB of SRAM and 4 KB of EEPROM. A Software Serial library allows for serial communication on any of the Mega's digital pins.



SIM808

General features

• Quad-band 850/900/1800/1900MHz

• GPRS multi-slot class 12/10

• GPRS mobile station class B

• Dimensions: 24*24*2.6mm

• Weight: 3.3g

• Control via AT commands (3GPP TS 27.007, 27.005 and SIMCOM enhanced AT Commands)

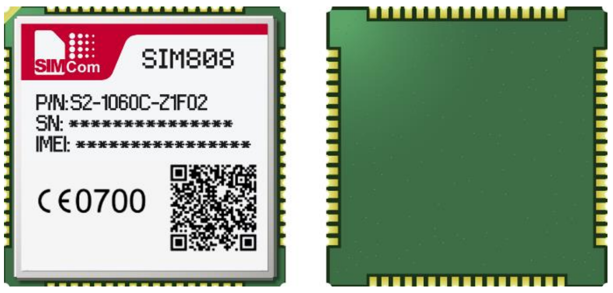
• Supply voltage range 3.4 ~ 4.4V

Specifications for SMS via GSM/GPRS

• Point to point MO and MT

• SMS cell broadcast

• Text and PDU mode



Specification for GPS

• Receiver type

- 22 tracking /66 acquisition

- channel-GPS L1 C/A code

• Sensitivity

- Tracking: -165 dBm

- Cold starts: -148 dBm

• Time-To-First-Fix

- Cold starts: 32s (typ.)

- Hot starts: <1s

- Warm starts: 3s

• Accuracy-Horizontal position : <2.5m CEP

Software features

• 0710 MUX protocol

• Embedded TCP/UDP protocol

• FTP/HTTP

• MMS

• POP3/SMTP

• DTMF

• Jamming Detection

• Audio Record

• SSL

• Bluetooth 3.0 (optional)

GPS ANTENNA

- Center Frequency 1575.42MHz±3 MHz
- V.S.W.R 1.5:1
- Band Width ±5 MHz
- Impendence 50 ohm
- Peak Gain ＞3dBic Based on 7×7cm ground plane
- Gain Coverage ＞-4dBic at –90°＜0＜+90°(over 75% Volume)
- Polarization RHCP
- DC Voltage 2.7V/3.0V/3.3V/5.0V/3.0V to 5.0V/other
- DC Current 5mA /11mA/15mA Max

3.4 Software Architecture

Open CV

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Pygame

Pygame is built over the Simple DirectMedia Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games (mainly the graphics part) can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

PySerial

PySerial encapsulates the access for the serial port. It provides backends for Python running on Windows, Linux, BSD (possibly any POSIX compliant system), Jython and IronPython (.NET and Mono). The module named serial automatically selects the appropriate backend.

Numpy

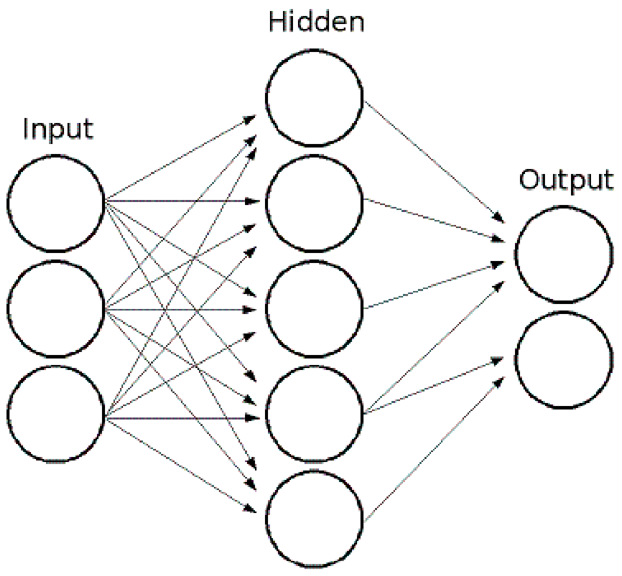
NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Scikit-learn

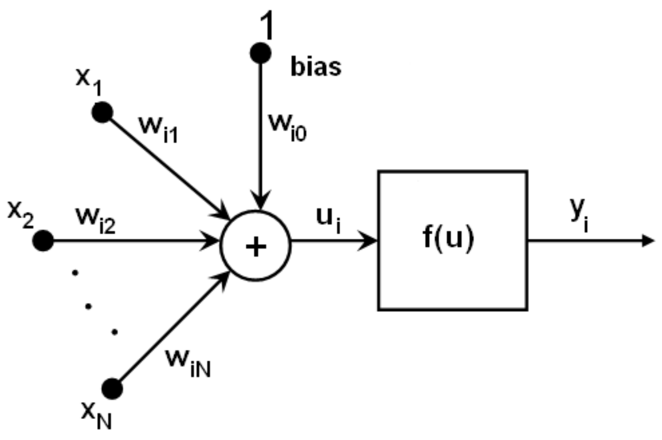
Scikit-learn is a free software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Machine Learning (ML) Implementation

Machine Learning (ML) implements feed-forward artificial neural networks or, more particularly, multi-layer perceptrons (MLP), the most commonly used type of neural networks. In this project we have implemented multi-layer perceptrons (MLP). MLP consists of the input layer, output layer, and one or more hidden layers. Each layer of MLP includes one or more neurons directionally linked with the neurons from the previous and the next layer. The example below represents a 3-layer perceptron with three inputs, two outputs, and the hidden layer including five neurons:



All the neurons in MLP are similar. Each of them has several input links (it takes the output values from several neurons in the previous layer as input) and several output links (it passes the response to several neurons in the next layer). The values retrieved from the previous layer are summed up with certain weights, individual for each neuron, plus the bias term. The sum is transformed using the activation function f that may be also different for different neurons.



In other words, given the outputs x_j of the layer n, the outputs y_i of the layer $n+1$ are computed as:

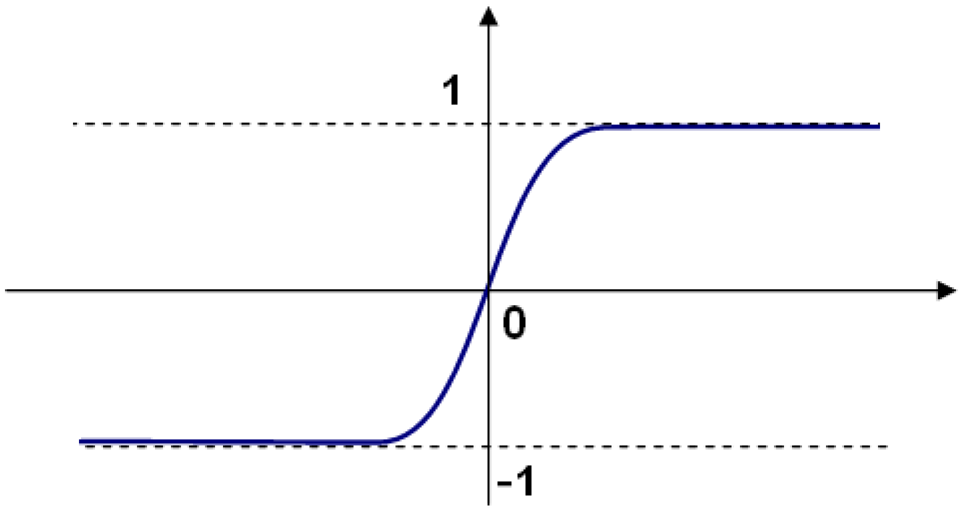
$$u_i = \sum_j (w^{n+1}_{i,j} * x_j) + w^{n+1}_{i,bias}$$

$$y_i = f(u_i)$$

Different activation functions may be used. ML implements three standard functions:

- Identity function (**CvANN_MLP::IDENTITY**): $f(x) = x$
- Symmetrical sigmoid

(**CvANN_MLP::SIGMOID_SYM**): $f(x) = \beta * (1 - e^{-\alpha x})/(1 + e^{-\alpha x})$), which is the default choice for MLP. The standard sigmoid with $\beta = 1, \alpha = 1$ is shown below:



- Gaussian function (CvANN_MLP::GAUSSIAN): $f(x) = \beta e^{-\alpha x * x}$, which is not completely supported at the moment.

In ML, all the neurons have the same activation functions, with the same free parameters (α, β) that are specified by user and are not altered by the training algorithms.

So, the whole trained network works as follows:

1. Take the feature vector as input. The vector size is equal to the size of the input layer.
2. Pass values as input to the first hidden layer.
3. Compute outputs of the hidden layer using the weights and the activation functions.
4. Pass outputs further downstream until you compute the output layer.

So, to compute the network, you need to know all the weights $w_{i,j}^{n+1)}$. The weights are computed by the training algorithm. The algorithm takes a training set, multiple input vectors with the corresponding output vectors, and iteratively adjusts the weights to enable the network to give the desired response to the provided input vectors.

The larger the network size (the number of hidden layers and their sizes) is, the more the potential network flexibility is. The error on the training set could be made arbitrarily small. But at the same time the learned network also "learns" the noise present in the training set, so the error on the test set usually starts increasing after the network size reaches a limit. Besides, the larger networks are trained much longer than the smaller ones, so it is reasonable to pre-process the data, using PCA::operator() or similar technique, and train a smaller network on only essential features.

Another MLP feature is an inability to handle categorical data as is. However, there is a workaround. If a certain feature in the input or output (in case of n -class classifier for $n > 2$) layer is categorical and can take $M > 2$ different values, it makes sense to represent it as a binary tuple of M elements, where the i -th element is 1 if and only if the feature is equal to the i -th value out of M possible. It increases the size of the input/output layer but speeds up the training algorithm convergence and at the same time enables "fuzzy" values of such variables, that is, a tuple of probabilities instead of a fixed value.

ML implements two algorithms for training MLP's. The first algorithm is a classical random sequential back-propagation algorithm. The second (default) one is a batch RPROP algorithm. Classical random sequential back-propagation algorithm is used to train MLP's in this project.

CvANN_MLP_TrainParams

struct **CvANN_MLP_TrainParams**

Parameters of the MLP training algorithm. You can initialize the structure by a constructor or the individual parameters can be adjusted after the structure is created.

The back-propagation algorithm parameters:

double **bp_dw_scale**

Strength of the weight gradient term. The recommended value is about 0.1.

double **bp_moment_scale**

Strength of the momentum term (the difference between weights on the 2 previous iterations). This parameter provides some inertia to smooth the random fluctuations of the weights. It can vary from 0 (the feature is disabled) to 1 and beyond. The value 0.1 or so is good enough

CvANN_MLP_TrainParams::CvANN_MLP_TrainParams

The constructors

C++: CvANN_MLP_TrainParams::**CvANN_MLP_TrainParams**()

C++: CvANN_MLP_TrainParams::**CvANN_MLP_TrainParams**(CvTermCriteria **term_crit**, int **train_method**, double **param1**, double **param2**=0)

Parameter

- **term_crit** – Termination criteria of the training algorithm. You can specify the maximum number of iterations (max_iter) and/or how much the error could change between the iterations to make the algorithm continue (epsilon).
- **train_method** –

  Training method of the MLP. Possible values are:

  - **CvANN_MLP_TrainParams::BACKPROP** The back-propagation algorithm.
- **param1** –Parameter of the training method. It is bp_dw_scale for BACKDROP.
- **param2** – Parameter of the training method. It is bp_moment_scale for BACKDROP.

CvANN_MLP

class **CvANN_MLP** : *public* **CvStatModel**

Unlike many other models in ML that are constructed and trained at once, in the MLP model these steps are separated. First, a network with the specified topology is created using the non-default constructor or the method CvANN_MLP::create(). All the weights are set to zeros. Then, the network is trained using a set of input and output vectors. The training procedure can be repeated more than once, that is, the weights can be adjusted based on the new training data.

CvANN_MLP::CvANN_MLP

The constructors

Python: cv2.**ANN_MLP**([layerSizes[, activateFunc[, fparam1[, fparam2]]]]) →

<ANN_MLP object>

The advanced constructor allows creating MLP with the specified topology.

CvANN_MLP::create

Constructs MLP with the specified topology

Python: cv2.ANN_MLP.**create**(layerSizes[, activateFunc[, fparam1[, fparam2]]]) → None

Parameters

- **layerSizes** – Integer vector specifying the number of neurons in each layer including the input and output layers.
- **activateFunc** – Parameter specifying the activation function for each neuron: one of **CvANN_MLP::IDENTITY, CvANN_MLP::SIGMOID_SYM**, and **CvANN_MLP::GAUSSIAN**.
- **fparam1** – Free parameter of the activation function, α. See the formulas in the introduction section.
- **fparam2** – Free parameter of the activation function, β. See the formulas in the introduction section.

The method creates an MLP network with the specified topology and assigns the same activation function to all the neurons.

CvANN_MLP::train

Trains/updates MLP

Python: cv2.ANN_MLP.**train**(inputs, outputs, sampleWeights[, sampleIdx[, params[, flags]]]) → retval

Parameters:

- **Inputs** – Floating-point matrix of input vectors, one vector per row.
- **Outputs** – Floating-point matrix of the corresponding output vectors, one vector per row.
- **sampleIdx** – Optional integer vector indicating the samples (rows of inputs and outputs) that are taken into account.
- **params** – Training parameters. See the **CvANN_MLP_TrainParams** description.
- **Flags** – Various parameters to control the training algorithm. A combination of the following parameters is possible:
    - **UPDATE_WEIGHTS** Algorithm updates the network weights, rather than computes them from scratch. In the latter case the weights are initialized using the Nguyen-Widrow algorithm.
    - **NO_INPUT_SCALE** Algorithm does not normalize the input vectors. If this flag is not set, the training algorithm normalizes each input feature independently, shifting its mean value to 0 and making the standard deviation equal to 1. If the network is assumed to be updated frequently, the new training data could be much different from original one. In this case, you should take care of proper normalization.
    - **NO_OUTPUT_SCALE** Algorithm does not normalize the output vectors. If the flag is not set, the training algorithm normalizes each output feature independently, by transforming it to the certain range depending on the used activation function.

This method applies the specified training algorithm to computing/adjusting the network weights. It returns the number of done iterations..

If you are using the default cvANN_MLP::SIGMOID_SYM activation function then the output should be in the range [-1,1], instead of [0,1], for optimal results.

CvANN_MLP::predict

Predicts responses for input samples

Python: cv2.ANN_MLP.predict(inputs[, outputs]) → retval, outputs

Parameters:

- **Inputs** – Input samples.
- **Outputs** – Predicted responses for corresponding samples.

The method returns a dummy value which should be ignored.

If you are using the default **cvANN_MLP::SIGMOID_SYM** activation function with the default parameter values fparam1=0 and fparam2=0 then the function used is y = 1.7159*tanh(2/3 * x), so the output will range from [-1.7159, 1.7159], instead of [0,1].

Tracking Implementation

Libraries Used:

SoftwareSerial Library

The Arduino hardware has built-in support for serial communication on pins 0 and 1 (which also goes to the computer via the USB connection). The native serial support happens via a piece of hardware (built into the chip) called a UART. This hardware allows the Atmega chip to receive serial communication even while working on other tasks, as long as there room in the 64 byte serial buffer.

The SoftwareSerial library has been developed to allow serial communication on other digital pins of the Arduino, using software to replicate the functionality (hence the name "SoftwareSerial"). It is possible to have multiple software serial ports with speeds up to 115200 bps. A parameter enables inverted signaling for devices which require that protocol.

```
SoftwareSerial mySerial(10,11);    // Pin 10: RX and PIN 11:TX
```

DFRobot_SIM808 Library

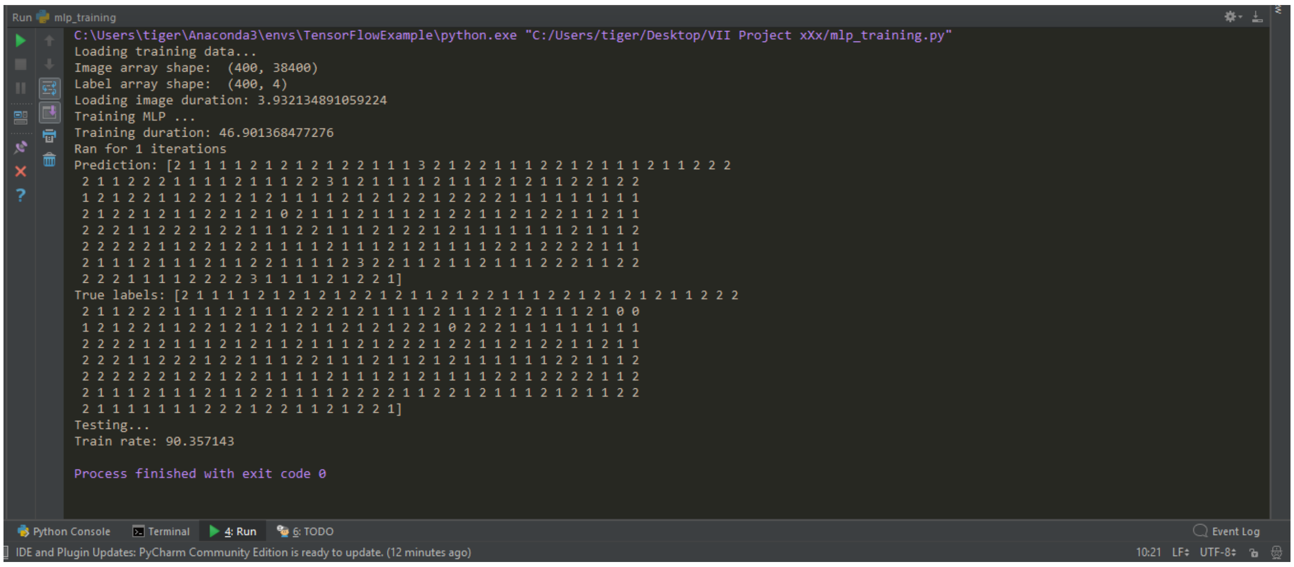
- Send and receive GPRS data (TCP/IP, HTTP, etc.)
- Receive GPS data and A-GPS data
- Send and receive SMS messages
- Make and receive phone calls

DFRobot_SIM808 sim808(&mySerial);   // Communicate with SIM808 using mySerial pins 10 and 11 (Software serial pins)//

CHAPTER 4: RESULTS AND DISCUSSION

Prediction on the testing samples returns an accuracy of greater than 80% compared to the accuracy of 96% that the training samples returns. In actual driving situation, predictions are generated about 10 times a second (streaming rate roughly 10 frames/second). Overall, the RC car could successfully navigate on the track based on the predictions made by the MLP program and tracking mechanism gives accurate position of the location.

Training Output

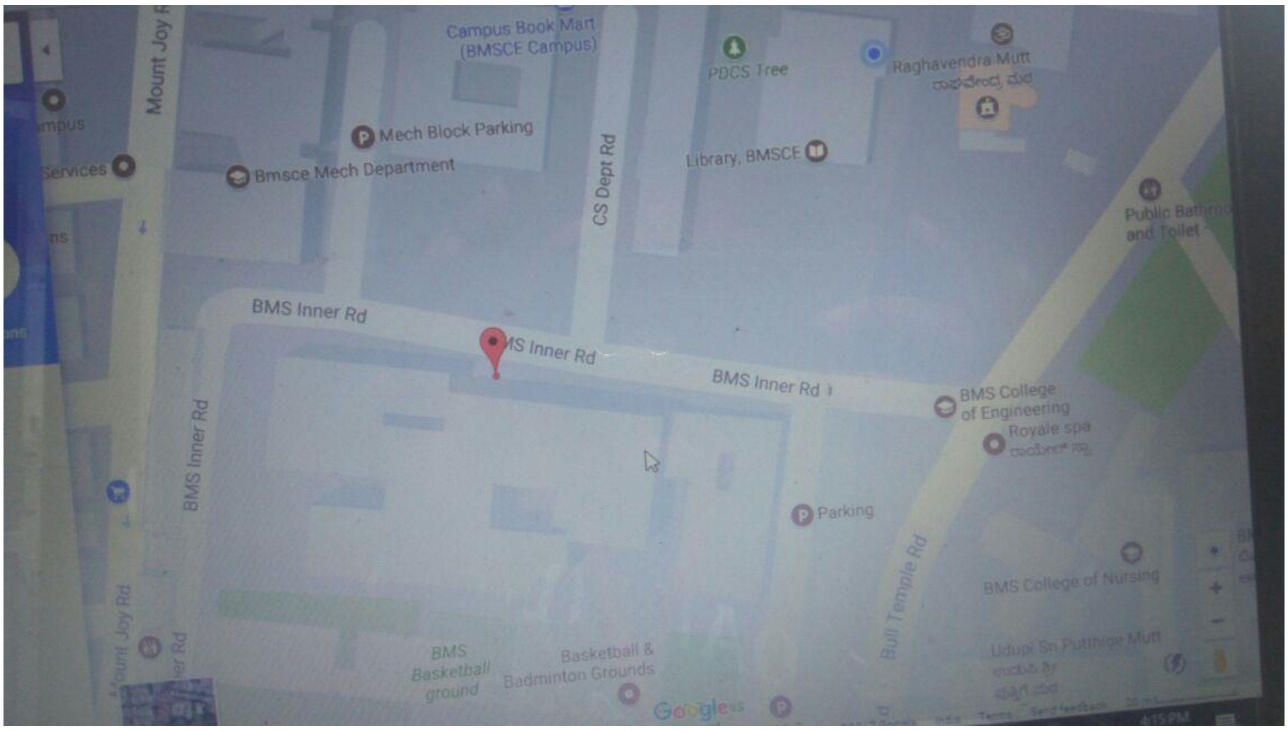


```
C:\Users\tiger\Anaconda3\envs\TensorFlowExample\python.exe "C:/Users/tiger/Desktop/VII Project xXx/mlp_training.py"
Loading training data...
Image array shape:  (400, 38400)
Label array shape:  (400, 4)
Loading image duration: 3.932134891059224
Training MLP ...
Training duration: 46.901368477276
Ran for 1 iterations
Prediction: [2 1 1 1 1 2 1 2 1 2 1 2 2 1 1 1 3 2 1 2 2 1 1 1 2 2 1 2 1 1 1 2 1 1 2 2 2
 2 1 1 2 2 2 1 1 1 1 2 1 1 1 2 2 3 1 2 1 1 1 1 2 1 1 1 2 1 2 1 1 2 2 1 2 2
 1 2 1 2 2 1 1 2 2 1 2 1 2 1 1 1 1 2 1 2 1 2 2 1 2 2 2 2 1 1 1 1 1 1 1 1 1
 2 1 2 2 1 2 1 1 2 2 1 2 1 0 2 1 1 1 2 1 1 1 2 1 2 2 1 1 2 1 2 2 1 1 2 1 1
 2 2 2 1 1 2 2 2 1 2 2 1 1 1 2 2 1 1 1 2 1 2 2 1 2 1 1 1 1 1 1 1 2 1 1 1 2
 2 2 2 2 2 1 1 2 2 1 2 2 1 1 1 1 2 1 1 1 2 1 2 1 1 1 1 2 2 1 2 2 2 2 1 1 1
 2 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 2 3 2 2 1 1 2 1 1 2 1 1 1 2 2 2 1 1 2 2
 2 2 2 1 1 1 1 2 2 2 2 3 1 1 1 1 2 1 2 2 1]
True labels: [2 1 1 1 1 2 1 2 1 2 1 2 2 1 2 1 1 2 1 2 2 1 1 1 2 2 1 2 1 2 1 2 1 1 2 2 2
 2 1 1 2 2 2 1 1 1 1 2 1 1 1 2 2 2 1 2 1 1 1 1 2 1 1 1 2 1 2 1 1 1 2 1 0 0
 1 2 1 2 2 1 1 2 2 1 2 1 2 1 2 1 1 2 1 2 1 2 2 1 0 2 2 2 1 1 1 1 1 1 1 1 1
 2 2 2 2 1 2 1 1 1 2 1 2 1 1 2 1 1 1 2 1 2 2 2 1 2 2 1 1 2 1 2 2 1 1 2 1 1
 2 2 2 1 1 2 2 2 1 2 2 1 1 1 2 2 1 1 1 2 1 1 2 1 2 1 1 1 1 1 1 2 2 1 1 1 2
 2 2 2 2 2 2 1 2 2 1 2 2 1 1 1 1 2 1 1 1 2 1 2 1 1 1 1 2 2 1 2 2 2 2 1 1 2
 2 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 2 2 2 2 1 1 2 2 1 2 1 1 1 2 1 2 1 1 2 2
 2 1 1 1 1 1 1 1 2 2 2 1 2 2 1 1 2 1 2 2 1]
Testing...
Train rate: 90.357143

Process finished with exit code 0
```

Run
mlp_training
Python Console
Terminal
4: Run
6: TODO
Event Log
IDE and Plugin Updates: PyCharm Community Edition is ready to update. (12 minutes ago)
10:21
LF÷
UTF-8÷

Tracking Output



CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

Mobility has been the buzzword of the 21st century. Autonomous travel falls in line with our push toward total mobility, enabling us to work, eat, sleep, and more on our daily commute in the privacy of our own vehicles. A car that safely drives itself and that can react to the environment and real-time roadway hazards could be society's answer to solving accidents caused by distractions and human error. An autonomous vehicle is the ideal solution. It's not only predicted to save 300,000 lives per decade, but it will also enable us to, literally, work on the go.

5.2 Future Work

Modify a RC car to handle tasks such as stop sign and traffic light detection(using Haar-Cascade Classifiers), and front collision avoidance(using Ultra-sonic sensor). Haar feature-based cascade classifiers for object detection, since each object requires its own classifier and follows the same process in training and detection as in this project

BIBLIOGRAPHY

1. Alessandrini, A., Campagna, A., Delle Site, A. & Filippi, F., 2015. Automated Vehicles and theRethinking of Mobility and Cities. Transportation Research Procedia, Band 5, pp. 145-160.
2. Bainbridge, L., 1983. Ironies of Automation. Automatica, 19(6), p. 775–779. Bartl, M., 2013.
3. Bartl, M., 2015. The Future of Autonomous Driving - Introducing the Foresight Matrix to Support
4. Bartl, M., Füller , J., Mühlbacher, H. & Ernst, H., 2012. A manager's perspective on virtual customer.
5. Bertozzi, A. M., Broggib, A. & Fascioli, A., 2000. Vision-based intelligent vehicles: State of the art and perspectives. Robotics and Autonomous Systems, 31 July, 32(1), pp. 1-16.
6. Broggi, A. et al., 2013. Extensive Tests of Autonomous Driving Technologies. IEEE TRANSACTIONSON INTELLIGENT TRANSPORTATION SYSTEMS, 14(3).
7. Sabooh Ajaz, Muhammad Asim, Muhammad Ozair, Muhammad Ahmed, Mansoor Siddiqui, Zahid Mushtaq, "Autonomous Vehicle Monitoring & Tracking System"

Books / Reports

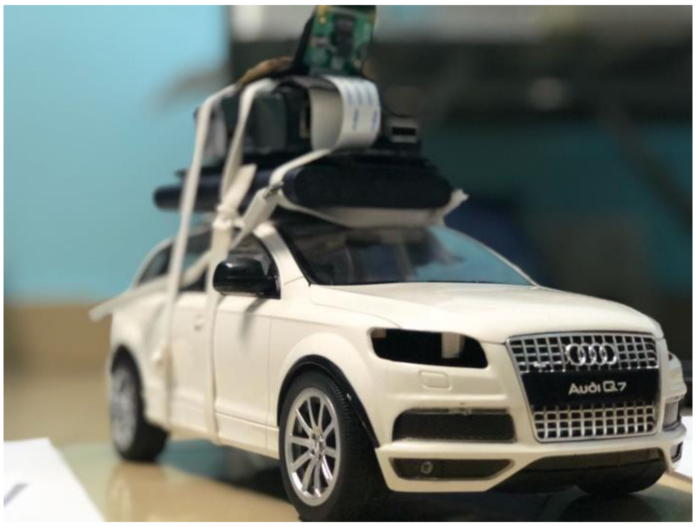
8. Chubin, D. E. & Hackett, E. J., 1990. Peerless science: peer review and US science policy, New York: Albany, State University of New York..
9. CNN, 2015. Money.CNN. [Online] [Zugriff am 6 July 2015].
10. Browning, J. G., 2014. Emerging Technology and Its Impact on Automotive Litigation. Defense Counsel Journal, 81(1), pp. 83-90.
11. Dahlander, L. & Gann, D. M., 2010. How open is innovation?. Research Policy, 39(6), pp. 699-709

GitHub and Webpages

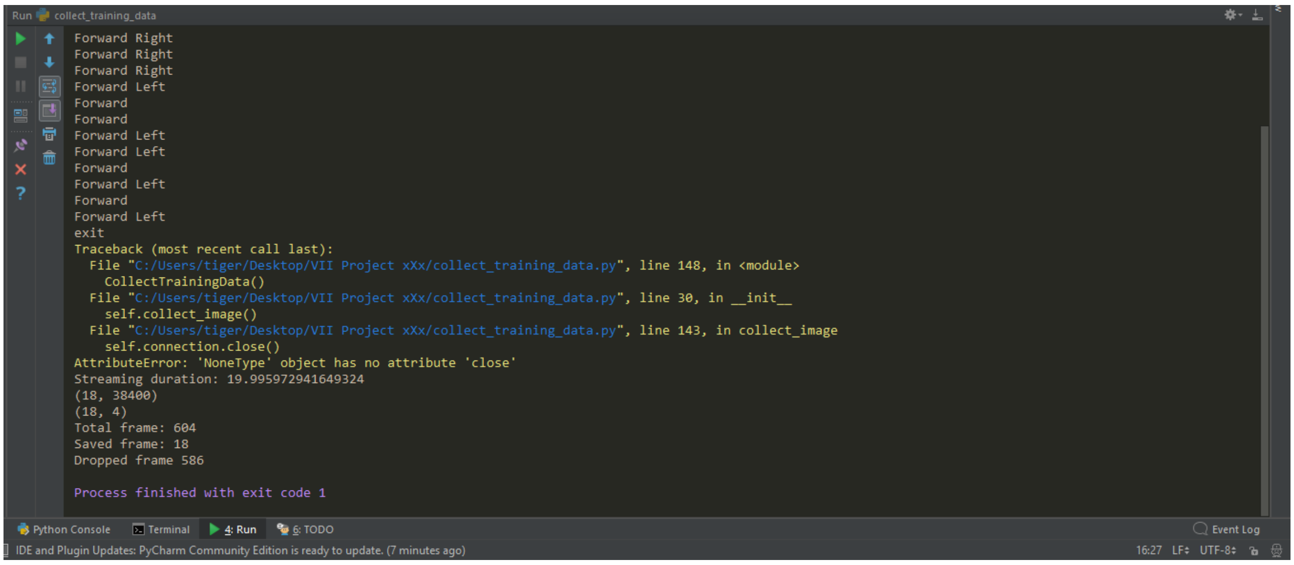
12. https://github.com/DFRobot/DFRobot_SIM808
13. http://electronicsforu.com/electronics-projects/hardware-diy/gsm-gps-based-vehicle-tracking-system
14. http://www.edgefxkits.com/vehicle-tracking-by-gps-gsm
15. https://github.com/mewanindula/GPS-Tracking-System/blob/master/README.md
16. https://github.com/hamuchiwa/AutoRCCar/blob/master/README.md

ANNEXURE

Project Pictures



Collecting Data Output

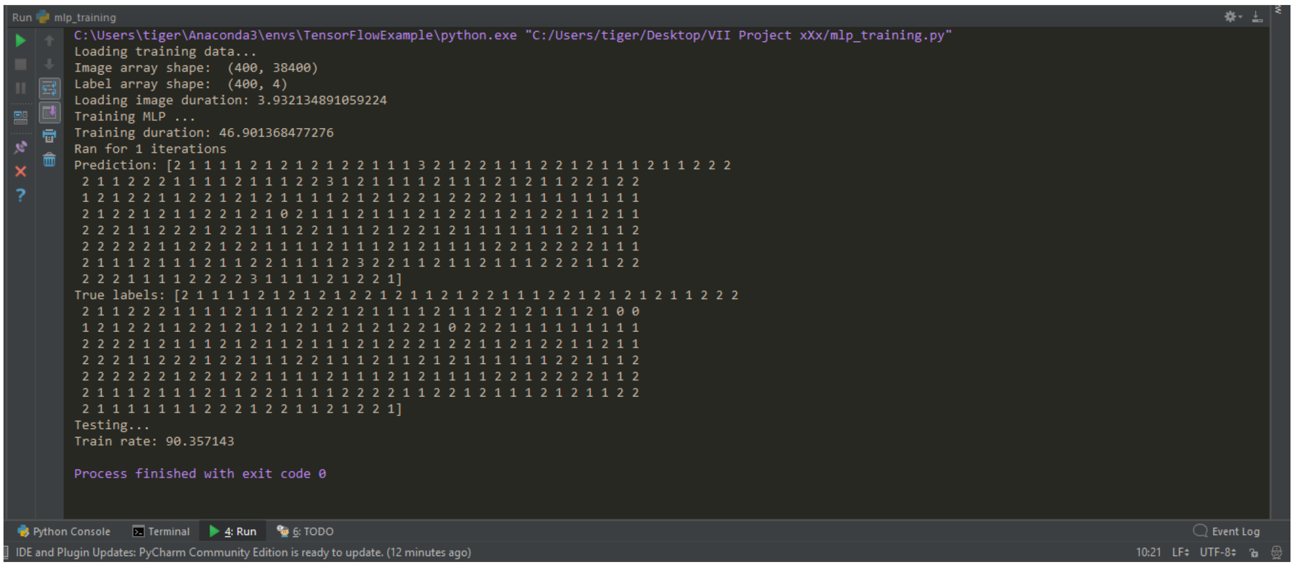


```
Forward Right
Forward Right
Forward Right
Forward Left
Forward
Forward
Forward Left
Forward Left
Forward
Forward Left
Forward
Forward Left
exit
Traceback (most recent call last):
  File "C:/Users/tiger/Desktop/VII Project xXx/collect_training_data.py", line 148, in <module>
    CollectTrainingData()
  File "C:/Users/tiger/Desktop/VII Project xXx/collect_training_data.py", line 30, in __init__
    self.collect_image()
  File "C:/Users/tiger/Desktop/VII Project xXx/collect_training_data.py", line 143, in collect_image
    self.connection.close()
AttributeError: 'NoneType' object has no attribute 'close'
Streaming duration: 19.995972941649324
(18, 38400)
(18, 4)
Total frame: 604
Saved frame: 18
Dropped frame 586

Process finished with exit code 1
```

IDE and Plugin Updates: PyCharm Community Edition is ready to update. (7 minutes ago)

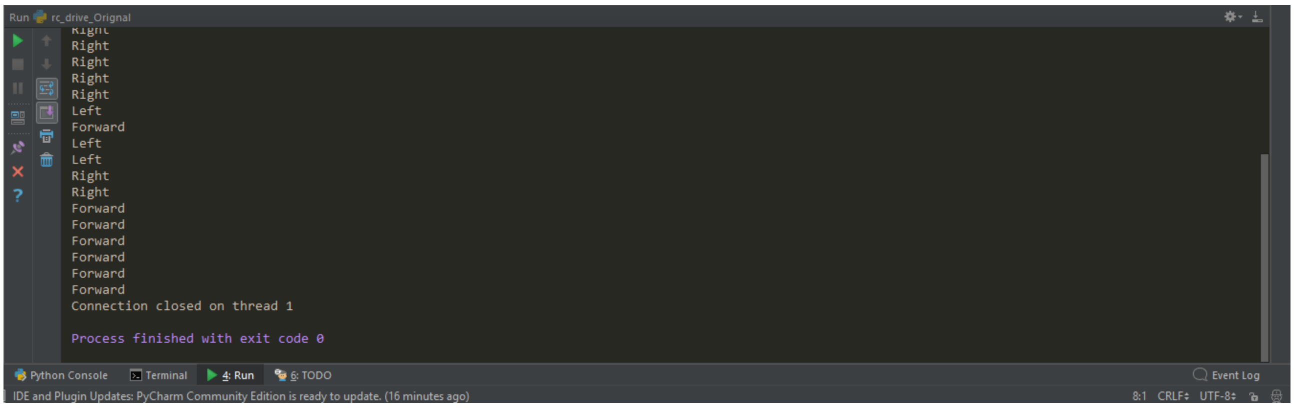
Training Output



```
C:\Users\tiger\Anaconda3\envs\TensorFlowExample\python.exe "C:/Users/tiger/Desktop/VII Project xXx/mlp_training.py"
Loading training data...
Image array shape:  (400, 38400)
Label array shape:  (400, 4)
Loading image duration: 3.932134891059224
Training MLP ...
Training duration: 46.901368477276
Ran for 1 iterations
Prediction: [2 1 1 1 1 2 1 2 1 2 1 2 2 1 1 1 3 2 1 2 2 1 1 1 2 2 1 2 1 1 1 2 1 1 2 2 2
 2 1 1 2 2 2 1 1 1 1 2 1 1 1 2 2 3 1 2 1 1 1 1 2 1 1 1 2 1 2 1 1 2 2 1 2 2
 1 2 1 2 2 1 1 2 2 1 2 1 2 1 1 1 1 2 1 2 1 2 2 1 2 2 2 2 1 1 1 1 1 1 1 1 1
 2 1 2 2 1 2 1 1 2 2 1 2 1 0 2 1 1 1 2 1 1 1 2 1 2 2 1 1 2 1 2 2 1 1 2 1 1
 2 2 2 1 1 2 2 2 1 2 2 1 1 1 2 2 1 1 1 2 1 2 2 1 2 1 1 1 1 1 1 1 2 1 1 1 2
 2 2 2 2 2 1 1 2 2 1 2 2 1 1 1 1 2 1 1 1 2 1 2 1 1 1 1 2 2 1 2 2 2 2 1 1 1
 2 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 2 3 2 2 1 1 2 1 1 2 1 1 1 2 2 2 1 1 2 2
 2 2 2 1 1 1 1 2 2 2 2 3 1 1 1 1 2 1 2 2 1]
True labels: [2 1 1 1 1 2 1 2 1 2 1 2 2 1 2 1 1 2 1 2 2 1 1 1 2 2 1 2 1 2 1 2 1 1 2 2 2
 2 1 1 2 2 2 1 1 1 1 2 1 1 1 2 2 2 1 2 1 1 1 1 2 1 1 1 2 1 2 1 1 1 2 1 0 0
 1 2 1 2 2 1 1 2 2 1 2 1 2 1 2 1 1 2 1 2 1 2 2 1 0 2 2 2 1 1 1 1 1 1 1 1 1
 2 2 2 2 1 2 1 1 1 2 1 2 1 1 2 1 1 1 2 1 2 2 2 1 2 2 1 1 2 1 2 2 1 1 2 1 1
 2 2 2 1 1 2 2 2 1 2 2 1 1 1 2 2 1 1 1 2 1 1 2 1 2 1 1 1 1 1 1 2 2 1 1 1 2
 2 2 2 2 2 2 1 2 2 1 2 2 1 1 1 1 2 1 1 1 2 1 2 1 1 1 1 2 2 1 2 2 2 2 1 1 2
 2 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 2 2 2 2 1 1 2 2 1 2 1 1 1 2 1 2 1 1 2 2
 2 1 1 1 1 1 1 1 2 2 2 1 2 2 1 1 2 1 2 2 1]
Testing...
Train rate: 90.357143

Process finished with exit code 0
```

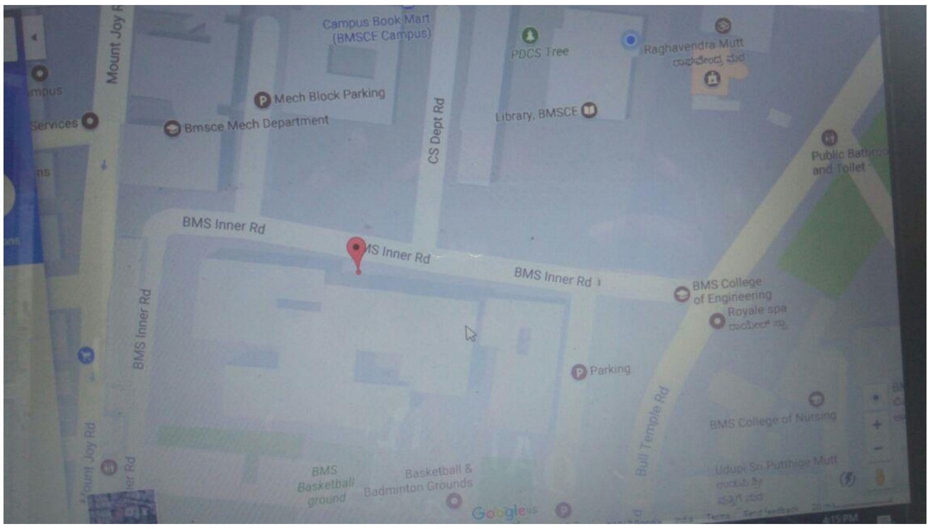
RC Self-Driving Output



```
Right
Right
Right
Right
Right
Left
Forward
Left
Left
Right
Right
Forward
Forward
Forward
Forward
Forward
Forward
Connection closed on thread 1

Process finished with exit code 0
```

Tracking Output



PROJECT CODES

stream_client.py

```
import numpy as np
import cv2
import socket

class VideoStreamingTest(object):
    def __init__(self):

        self.socket = socket.socket()
        self.connection = self.socket.connect(('192.168.0.6', 8081))
        self.streaming()

    def streaming(self):

        try:
            #print ("Connection from: ", self.client_address)
            print ("Streaming...")
            print ("Press 'q' to exit")
            frame = 0
            stream_bytes = b' '
            while True:

                stream_bytes += self.socket.recv(1024)
                first = stream_bytes.find(b'\xff\xd8')
                last = stream_bytes.find(b'\xff\xd9')
                if first != -1 and last != -1:
                    jpg = stream_bytes[first:last + 2]
                    stream_bytes = stream_bytes[last + 2:]

                    #image = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8), cv2.IMREAD_GRAYSCALE)

                    image = cv2.imdecode(np.fromstring(jpg, dtype=np.uint8),
cv2.IMREAD_COLOR)


#cv2.imwrite('stream_images/frame{:>05}.jpg'.format(frame), image)
                    cv2.imshow('image', image)

                    frame += 1

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break
        finally:
            #self.connection.close()
            self.socket.close()

if __name__ == '__main__':
    VideoStreamingTest()
```

rc_keyboard_control.ino

```
// set pin numbers:
const int UpPin =  10;      // The pin used to control the forward move
const int DwnPin =  11;      // The pin used to control the backward move
const int RghtPin =  13;      // The pin used to control the right move
const int LftPin =  12;      // The pin used to control the left move

int CommandByte = 0,i;   // Command received via serial port


void setup() {
// sets the digital pins as output
pinMode(UpPin, OUTPUT);
pinMode(DwnPin, OUTPUT);
pinMode(RghtPin, OUTPUT);
pinMode(LftPin, OUTPUT);
Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
}

void stop()
{
  digitalWrite(UpPin, LOW); // Stop forward
  digitalWrite(DwnPin, LOW); // Stop backward
  digitalWrite(RghtPin, LOW); // Stop turning right
  digitalWrite(LftPin, LOW);
}

void stopturn()
{
  digitalWrite(UpPin, LOW); // Stop forward
  digitalWrite(DwnPin, LOW); // Stop backward
}


void loop() {
// Whenever a byte is received via serial port
if (Serial.available() > 0)
    {
      // Read the incoming byte:
      CommandByte = Serial.read();
      // Perform the commands

      // Forward command is received
      if (CommandByte == '8') //W
        {
          // First be sure that Down button (go back) is unpressed, then
send forward command
            Serial.println("Forwards");
          //digitalWrite(DwnPin, LOW);  // Stop backward
          digitalWrite(UpPin, HIGH); // Go forward
          delay(135);
          stop();
        }

          // Backward command is received
            if (CommandByte == '2') //S
        {
          // First be sure that Up button (go forward) is unpressed, then
send backward command
            Serial.println("Backwards");
          //digitalWrite(UpPin, LOW); // Stop forward
          digitalWrite(DwnPin, HIGH); // Go backward
          delay(135);
          stop();
        }

         // Go right command is received
            if (CommandByte == '6') //D
        {
          // First be sure that Left button is unpressed, then send right
command
            Serial.println("Right");
          digitalWrite(LftPin, LOW); // Stop turning left
          digitalWrite(UpPin, HIGH);
          digitalWrite(RghtPin, HIGH);  // Turn right
          delay(135);
          stopturn();
        }

        // Go left command is received
            if (CommandByte == '4') //A
        {
            Serial.println("Left");
          // First be sure that right button is unpressed, then send left
command
          digitalWrite(RghtPin, LOW); // Stop turning right
          digitalWrite(UpPin, HIGH);
          digitalWrite(LftPin, HIGH); // Turn left
          delay(135);
          stopturn();
        }

        // Stop command is received
            if (CommandByte == '0') //B
        {
          // Stop everything
          Serial.println("Stop everything");
          digitalWrite(UpPin, LOW); // Stop forward
          digitalWrite(DwnPin, LOW); // Stop backward
          digitalWrite(RghtPin, LOW); // Stop turning right
          digitalWrite(LftPin, LOW); // Stop turning left
        }
    }
}
```

collect_training_data.py

```
import numpy as np
import cv2
import serial
import pygame
from pygame.locals import *
import socket
import time
import os


class CollectTrainingData(object):
    def __init__(self):

        self.socket = socket.socket()
        self.connection = self.socket.connect(('192.168.0.6', 8081))
        # self.streaming()


        # connect to a seral port
        self.ser = serial.Serial('COM3', 9600, timeout = 1)
        self.send_inst = True

        # create labels
        self.k = np.zeros((4, 4), 'float')
        for i in range(4):
            self.k[i, i] = 1
        self.temp_label = np.zeros((1, 4), 'float')

        pygame.init()
        screen = pygame.display.set_mode((400, 300))
        self.collect_image()

    def collect_image(self):

        saved_frame = 0
        total_frame = 0

        # collect images for training
        print('Start collecting images...')
        e1 = cv2.getTickCount()
        image_array = np.zeros((1, 38400))
        label_array = np.zeros((1, 4), 'float')

        # stream video frames one by one
        try:
            stream_bytes = b' '
            frame = 1
            while self.send_inst:
                stream_bytes += self.socket.recv(1024)
                first = stream_bytes.find(b'\xff\xd8')
                last = stream_bytes.find(b'\xff\xd9')
                if first != -1 and last != -1:
                    jpg = stream_bytes[first:last + 2]
                    stream_bytes = stream_bytes[last + 2:]
                    image = cv2.imdecode(np.fromstring(jpg, dtype =
np.uint8), cv2.IMREAD_GRAYSCALE)
```

```
                    # select lower half of the image
                    roi = image[120:240, :]

                    # save streamed images

cv2.imwrite('training_images/frame{:>05}.jpg'.format(frame), image)

                    # cv2.imshow('roi_image', roi)
                    cv2.imshow('image', image)

                    # reshape the roi image into one row array
                    temp_array = roi.reshape(1, 38400).astype(np.float32)

                    frame += 1
                    total_frame += 1

                    # get input from human driver
                    for event in pygame.event.get():
                        if event.type == KEYDOWN:
                            key_input = pygame.key.get_pressed()

                            # complex orders
                            if key_input[pygame.K_RIGHT]:
                                print("Forward Right")
                                image_array = np.vstack((image_array,
temp_array))
                                label_array = np.vstack((label_array,
self.k[1]))
                                saved_frame += 1
                                self.ser.write("6".encode())

                            elif key_input[pygame.K_LEFT]:
                                print("Forward Left")
                                image_array = np.vstack((image_array,
temp_array))
                                label_array = np.vstack((label_array,
self.k[0]))
                                saved_frame += 1
                                self.ser.write("4".encode())


                            # simple orders
                            elif key_input[pygame.K_UP]:
                                print("Forward")
                                saved_frame += 1
                                image_array = np.vstack((image_array,
temp_array))
                                label_array = np.vstack((label_array,
self.k[2]))
                                self.ser.write("8".encode())

                            elif key_input[pygame.K_DOWN]:
                                print("Reverse")
                                saved_frame += 1
                                image_array = np.vstack((image_array,
temp_array))
                                label_array = np.vstack((label_array,
self.k[3]))
                                self.ser.write("2".encode())
                            elif key_input[pygame.K_x] or 
key_input[pygame.K_q]:
                                print('exit')
                                self.send_inst = False
                                self.ser.write("0".encode())
                                break

                        elif event.type == pygame.KEYUP:
                            self.ser.write("0".encode())

            # save training images and labels
            train = image_array[1:, :]
            train_labels = label_array[1:, :]

            # save training data as a numpy file
            file_name = str(int(time.time()))
            directory = "training_data"
            if not os.path.exists(directory):
                os.makedirs(directory)
            try:
                np.savez(directory + '/' + file_name + '.npz', train = 
train, train_labels = train_labels)
            except IOError as e:
                print(e)

            e2 = cv2.getTickCount()
            # calculate streaming duration
            time0 = (e2 - e1) / cv2.getTickFrequency()
            print('Streaming duration:', time0)

            print(train.shape)
            print(train_labels.shape)
            print('Total frame:', total_frame)
            print('Saved frame:', saved_frame)
            print('Dropped frame', total_frame - saved_frame)

        finally:
            self.connection.close()
            self.socket.close()


if __name__ == '__main__':
    CollectTrainingData()
```

mlp_training.py

```
import cv2
import numpy as np
import glob
import sys
from sklearn.model_selection import train_test_split

print('Loading training data...')
e0 = cv2.getTickCount()

# load training data
image_array = np.zeros((1, 38400))
label_array = np.zeros((1, 4), 'float')
training_data = glob.glob('training_data/*.npz')

# if no data, exit
if not training_data:
    print("No training data in directory, exit")
    sys.exit()

for single_npz in training_data:
    with np.load(single_npz) as data:
        train_temp = data['train']
        train_labels_temp = data['train_labels']
    image_array = np.vstack((image_array, train_temp))
    label_array = np.vstack((label_array, train_labels_temp))

X = image_array[1:, :]
y = label_array[1:, :]
print('Image array shape: ', X.shape)
print('Label array shape: ', y.shape)

e00 = cv2.getTickCount()
time0 = (e00 - e0)/ cv2.getTickFrequency()
print('Loading image duration:', time0)

# train test split, 7:3
train, test, train_labels, test_labels = train_test_split(X, y,
test_size=0.3)

# set start time
e1 = cv2.getTickCount()

# create MLP
layer_sizes = np.int32([38400, 32, 4])
model = cv2.ml.ANN_MLP_create()
model.setLayerSizes(layer_sizes)
model.setTrainMethod(cv2.ml.ANN_MLP_BACKPROP)
model.setBackpropMomentumScale(0.0)
model.setBackpropWeightScale(0.001)
model.setTermCriteria((cv2.TERM_CRITERIA_COUNT, 20, 0.01))
model.setActivationFunction(cv2.ml.ANN_MLP_SIGMOID_SYM, 2, 1)

criteria = (cv2.TERM_CRITERIA_COUNT | cv2.TERM_CRITERIA_EPS, 500, 0.0001)
#criteria2 = (cv2.TERM_CRITERIA_COUNT, 100, 0.001)
params = dict(term_crit = criteria,
               train_method = cv2.ml.ANN_MLP_BACKPROP,
               bp_dw_scale = 0.001,
               bp_moment_scale = 0.0 )
```

```
print ('Training MLP ...')
num_iter = model.train(np.float32(train), cv2.ml.ROW_SAMPLE,
np.float32(train_labels))

# set end time
e2 = cv2.getTickCount()
time = (e2 - e1)/cv2.getTickFrequency()
print ('Training duration:', time)
#print 'Ran for %d iterations' % num_iter


# save model (THIS IS THE NEW GUY BTW)

model.save('mlp_xml/mlp.xml')

print('Ran for %d iterations' % num_iter)

ret, resp = model.predict(train)
prediction = resp.argmax(-1)
print('Prediction:', prediction)
true_labels = train_labels.argmax(-1)
print('True labels:', true_labels)

print('Testing...')
train_rate = np.mean(prediction == true_labels)
print('Train rate: %f' % (train_rate*100))
```

rc_driver.py

```
import socket
import serial
import cv2
import time
import numpy as np
import math


class NeuralNetwork(object):
    def __init__(self):
        self.model = cv2.ml.ANN_MLP_create()

    def create(self):
        layer_size = np.int32([38400, 32, 4])
        self.model.setLayerSizes(layer_size)
        self.model = cv2.ml.ANN_MLP_load('mlp_xml/mlp.xml')

    def predict(self, samples):
        ret, resp = self.model.predict(samples)
        return resp.argmax(-1)


class RCControl(object):
    def __init__(self):
        self.serial_port = serial.Serial('COM3', 9600, timeout = 1)

    def steer(self, prediction):
        if prediction == 2:
            self.serial_port.write("8".encode())
            print("Forward")
        elif prediction == 0:
            self.serial_port.write("4".encode())
            print("Left")
        elif prediction == 1:
            self.serial_port.write("6".encode())
            print("Right")
        else:
            self.stop()

    def stop(self):
        self.serial_port.write("0".encode())

class Execute(object):

    def __init__(self):
        self.socket = socket.socket()
        self.connection = self.socket.connect(('192.168.0.6', 8081))
        self.MainProgram()

    def MainProgram(self):
        NeuralModel = NeuralNetwork()
        NeuralModel.create()
        rc_car = RCControl()
        delay = 0
        stream_bytes = b' '
        # stream video frames one by one
        try:
            while True:
                stream_bytes += self.socket.recv(1024)
                first = stream_bytes.find(b'\xff\xd8')
                last = stream_bytes.find(b'\xff\xd9')
                if first != -1 and last != -1:
                    jpg = stream_bytes[first:last + 2]
                    stream_bytes = stream_bytes[last + 2:]
                    gray = cv2.imdecode(np.fromstring(jpg, dtype =
np.uint8), cv2.IMREAD_GRAYSCALE)
                    image = cv2.imdecode(np.fromstring(jpg, dtype =
np.uint8), cv2.IMREAD_UNCHANGED)
                    # lower half of the image
                    roi = gray[120:240, :]
                    cv2.imshow('image', image)
                    # cv2.imshow('mlp_image', half_gray)
                    # reshape image
                    image_array = roi.reshape(1, 38400).astype(np.float32)
                    # neural network makes prediction
                    # prediction = self.model.predict(image_array)
                    prediction = NeuralModel.predict(image_array)
                    if(delay == 0):
                        rc_car.steer(prediction)
                        delay = 50
                    delay = delay - 1
                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break
                    #time.sleep(1)

            cv2.destroyAllWindows()
        finally:
            print("Connection closed on thread 1")


if __name__ == '__main__':
    Execute()
```

Initializing the SIM808 module

```
mySerial.begin(9600);         // Communicate with module at 9600 baud rate
Serial.begin(9600);           // Serial monitor communication at 9600 baud rate

//******** Initialize sim808 module *************

while(!sim808.init())                         //When the module is not initialized

{

 Serial.print("Sim808 init error\r\n");        // Print initialization error on the serial monitor
delay(1000);
}

delay(3000);

Serial.println("SIM Init success");           //If initialized then print the success message

  // The module is successfully initialized and the operation starts once the SIM in the module
receives a message from another SIM to which the location needs to be sent//

Serial.println("Init Success\n please send SMS message to me!");
```

Functions written (main operational code)

1. **void readSMS()**

```
{

Serial.print("messageIndex: ");     // The message index is updated when a message is received
Serial.println(messageIndex);       // The message index is printed on the serial screen.

// The format of the inbuilt library function is given as"//

sim808.readSMS(messageIndex, message, MESSAGE_LENGTH, phone, datetime);

  //***********In order not to full SIM Memory, is better to delete it**********

  sim808.deleteSMS(messageIndex);

// Access the information read through sim808.readSMS() function and display on serial monitor

Serial.print("From number: ");          // Phone number from which the message was received
Serial.println(phone);

Serial.print("Datetime: ");             // Date and time when the message was received
Serial.println(datetime);

Serial.print("Received Message: ");     // The content of the message
Serial.println(message);
}

    2.  void getGPS()

{
while(!sim808.attachGPS())                    // If the GPS is not powered
{
Serial.println("GPS power failure");
delay(1000);
}
delay(3000);
Serial.println("GPS power success");     // The GPS is powered  successfully

    while(!sim808.getGPS())
{
// Wait for GPS to get its data once its powered
}

// Store the latitude and the longitude values in two variables //

float la = sim808.GPSdata.lat;        //LATITUDE//
float lo = sim808.GPSdata.lon         //LONGITUDE//

// Covert the float value into an array so that it can be sent as serial data in a message//

dtostrf(la, 4, 6, lat);       //4 = no. of digits before decimal. 6 = no. of digits after the decimal
dtostrf(lo, 4, 6, lon);       // high accuracy due to decimal extension

// Print the latitude (lat) and longitude (lon) values on the screen//

Serial.print("latitude :");
Serial.println(lat);
Serial.print("longitude :");
Serial.println(lon);

// Create the message that needs to be sent which contains the GPS data and the link on Google
Maps//

sprintf(MESSAGE, "Latitude : %s\n Longitude : %s \n
http://maps.google.com/maps?q=%s,%s \n", lat, lon, lat, lon);

}
```

3. **void sendSMS()**

```
{

Serial.println("Start to send message ...");

Serial.println(MESSAGE);      //Print the message created on the screen
Serial.println(phone);        //Print the phone number to which the message needs to be sent

// Use the library function to send the SMS from the SIM in the GSM module (SIM808) to the
phone number from which the initial message was received (interpreted from the ‘phone’ variable
from readSMS() function)//

sim808.sendSMS(phone,MESSAGE);

}
```