Project Report

Automotive Sensor for Object Recognition using Red Pitaya and Raspberry Pi

IT Master of Engineering (M. Eng.) Sehrish Fahim Mohd. Imranul Hoque Limon  
[limon@stud.fra-uas.de](mailto:limon@stud.fra-uas.de)(1276493) sfahim@stud.fra-uas.de(1277599)

*Abstract*— **Objective is to design software for receiving, displaying and saving the acquired ultrasonic signals from Red Pitaya with Raspberry Pi. The objective is to make the system stable. First is to acquire the data from the ultrasonic sensor with the range of 1 meter. taking the data with least noise ratio by capturing the time when object data is available using external trigger condition.**

**When we have desired object before working on feature part we have cut down the input part to make it more specific to get desired and accurate data for each object. For feature extraction we used fft,psd and variance for finding features at last we applied machine learning that is Naïve Bayes in extracted features and trained our system.**

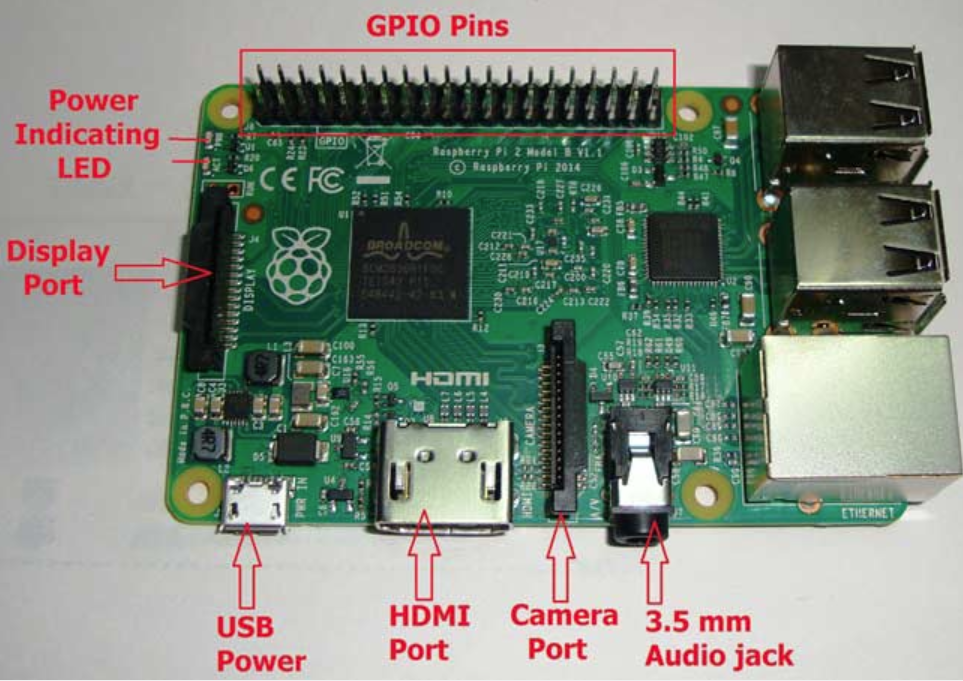
Keywords—Scpi , ssh, psd, fft

# Introduction (*Rasberry pi*)

The Raspberry Pi is a low cost computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. We can use it to design own software and we can use program in languages like Scratch and Python. It’s capable of doing everything you’d expect a desktop computer to do, from browsing the internet and coding for software.

The reason to use this is to able to connect it with Red pitaya to show how one device which is connected to the sensor can send data and we can use raspberry as a receiver for that information and code to make it intelligent to recognize different objects precisely.

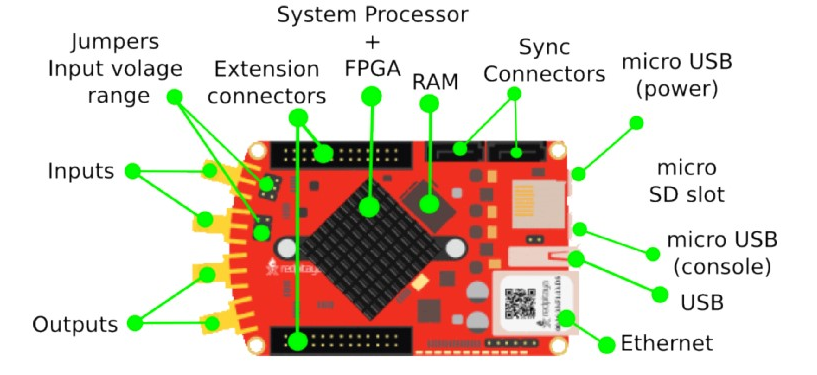
Like a normal computer raspberry pi contains processor, storage, connectivity, video audio support, multimedia



# Introduction (*red pitaya*)

Red Pitaya is an open-source hardware made to alternate for many expensive laboratory measurement and control instruments.

Red Pitaya offers open-source-software measurement and control tools that consists of easy-to-use visual programming software and free of charge, ready-to-use open-source, web-based test and measurement instruments running on powerful, credit card-sized boards. With a single click, the board can transform into a web-based oscilloscope, spectrum analyzer, signal generator, LCR meter, Bode analyzer, or one of many other applications. Red Pitaya can be controlled by using Matlab, LabVIEW, Python & Scilab. We use python code for our software development.



*Key Features*

* High performance hardware
* Replaces most essential instruments like Oscilloscope, Spectrum analyzer, Signal generator, LCR meter
* LAN or wireless access from any WEB browser via tablet or a PC regardless of the OS (MAC, Linux, Windows, Android, iOS) LabVIEW and MATLAB® interface.
* Possibility to make your own application and share it with others
* Open source software

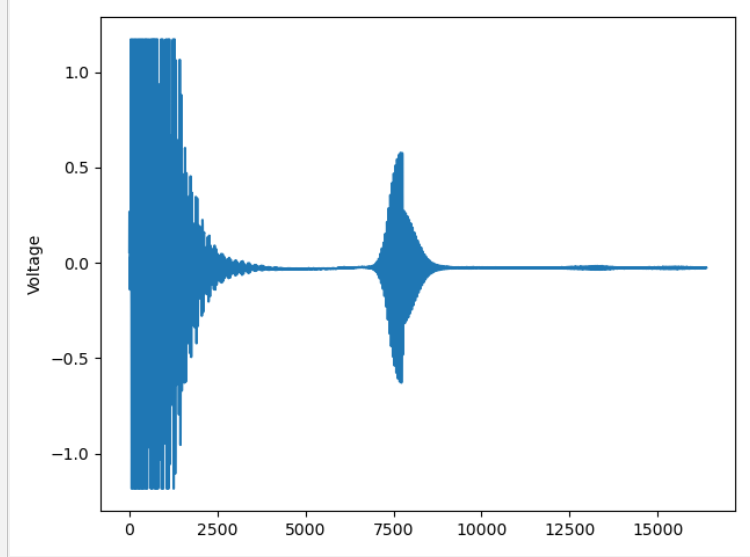
*Software*

Red Pitaya is based on GNU/Linux operating system and can be customized at different programming levels. Available software interfaces include: HDL, C/C++, scripting languages, MATLAB and HTML based web interfaces.Red Pitaya software is open source and can be downloaded from GitHub. All development tools are free.

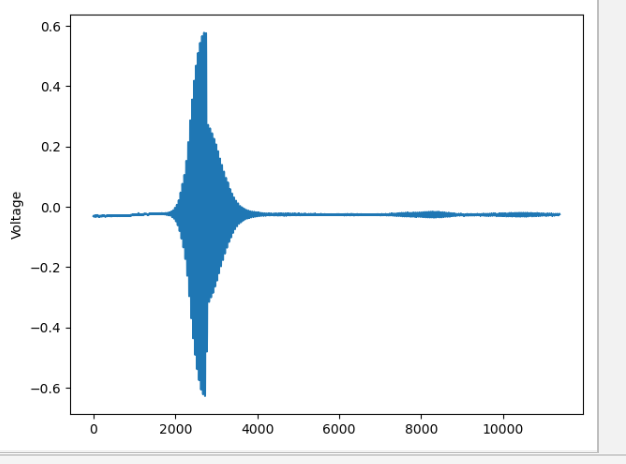
# Overview

## Data acquizition

When we have desired object before working on feature part we have cut down the input part to make it more stable and accurate data for each object. Then after analyzing the data, develop feature extraction and classification. We have chosen Bayes Classifier for Machine learning part and trained our model. We have used frequency domain for this purpose.



Signal input with object reflection: Fig-1



After elimination of input signal: Fig 2

## Feature Extracion

In Fig-2 which give us more precise look to object to extract features. We have applied fft (Fast Fourier transform) in order to have frequency domain for captured signal, after that we have done power spectrum.

A Power Spectral Density (PSD) is the measure of signal's power content versus frequency. A PSD is typically used to characterize broadband random signals. It can be looked upon as a frequency-domain plot of power per unit Hz vs. frequency.

Therefore, while the power spectrum calculates the area under the signal plot using the discrete Fourier Transform, the power spectrum density assigns unit of frequency and thus, enhances periodicities. By doing this it will give us more detailed signal to extract features. By this data we have taken peaks, shape, mean and variance as features to proceed with chosen Bayes Classification.

# Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience.

## Selection of Naive Bayes as Machine Learning

Naive Bayes comes under the class of generative models for classification. It models the posterior probability from the class conditional densities. So the output is a probability of belonging to a class.

SVM on the other hand is based on a discriminant function given by y = w.x+b. Here the weights w and bias parameter b are estimated from the training data. It tries to find a hyperplane that maximizes the margin and there is optimization function in this regard. Performance wise SVMs using the radial basis function kernel are more likely to perform better as they can handle non-linearity’s in the data. Naive Bayes performs best when the features are independent of each other which often does not happen in real. Having said that it still performs good even when the features are not independent. We have chosen Naive Bayes as our features are independent of each other and we have three objects to identify (class) Class 1 Wall, Class 2 Human, Class 3 Car. By using Naive Bayes, we will train and create these three type of classes and train our model.

## How Naïve Bayes Work(Implementation)

We can describe this in 5 part:

Step 1: Separate by Class.

Step 2: Summarize Dataset.

Step 3: Summarize Data by Class.

Step 4: Gaussian Probability Density Function.

Step 5: Class Probabilities

## Step 1:Separate By Class

We will need to calculate the probability of data by the class they belong to, the so-called base rate. This means that we will first need to separate our training data by class. A relatively straightforward operation. We can create a dictionary object where each key is the class value and then add a list of all the records as the value in the dictionary. function named is separate\_by\_class()

## Step 2: Summrize Dataset

We’ll describe how these statistics are used in the calculation of probabilities in a few steps. The two statistics we require from a given dataset are the mean and the standard deviation (average deviation from the mean). The mean is the average value and can be calculated as:

• mean = sum(x)/n \* count(x)

Where x is the list of values or a column we are looking.

We use small function named mean () that calculates the mean of a list of numbers. We require the mean and standard deviation statistics to be calculated for each input attribute or each column of our data. We can do that by gathering all of the values for each column into a list and calculating the mean and standard deviation on that list. Once calculated, we can gather the statistics together into a list or tuple of statistics. Then, repeat this operation for each column in the dataset and return a list of tuples of statistics.

## Step 3:Summarize Data By Class

We require statistics from our training dataset organized by class. we have developed the separate\_by\_class () function to separate a dataset into rows by class. And we have developed summarize\_dataset() function to calculate summary statistics for each column. We can put all of this together and summarize the columns in the dataset organized by class values. We create a function named summarize\_by\_class() that implements this operation. The dataset is first split by class, then statistics are calculated on each subset. The results in the form of a list of tuples of statistics are then stored in a dictionary by their class value.

## Step 4:Gaussian Probability Density Function

Calculating the probability or likelihood of observing a given real-value is difficult. One way we can do this is to assume that values are drawn from a distribution, such as a bell curve or Gaussian distribution. A Gaussian distribution can be summarized using only two numbers: the mean and the standard deviation. Therefore, with a little math, we can estimate the probability of a given value. This piece of math is called a Gaussian Probability Distribution Function (or Gaussian PDF) and can be calculated as:

• f(x) = (1 / sqrt(2 \* PI) \* sigma) \* exp(-((x-mean)^2 / (2 \* sigma^2)))

Where sigma is the standard deviation for x, mean is the mean for x and PI is the value of pi.

## Step 5:Class Probablities

Now it is time to use the statistics calculated from our training data to calculate probabilities for new data.

Probabilities are calculated separately for each class. This means that we first calculate the probability that a new piece of data belongs to the first class, then calculate probabilities that it belongs to the second class, and so on for all the classes. The probability that a piece of data belongs to a class is calculated as follows: P(class|data) = P(X|class) \* P(class)

This means that the result is no longer strictly a probability of the data belonging to a class. The value is still maximized, meaning that the calculation for the class that results in the largest value is taken as the prediction. This is a common implementation simplification as we are often more interested in the class prediction rather than the probability.

The input variables are treated separately, giving the technique its name “naive“. For example we have 2 input variables X1 and X2, the calculation of the probability that a row belongs to the first class 0 can be calculated as:

• P(class=0|X1,X2) = P(X1|class=0) \* P(X2|class=0) \* P(class=0)

Now you can see why we need to separate the data by class value. The Gaussian Probability Density function in the previous step is how we calculate the probability of a real value like X1 and the statistics we prepared are used in this calculation. We created a function named calculate\_class\_probabilities() that ties all of this together.

It takes a set of prepared summaries and a new row as input arguments. First the total number of training records is calculated from the counts stored in the summary statistics. This is used in the calculation of the probability of a given class or P(class) as the ratio of rows with a given class of all rows in the training data.

Next, probabilities are calculated for each input value in the row using the Gaussian probability density function and the statistics for that column and of that class. Probabilities are multiplied together as they accumulated. This process is repeated for each class in the dataset. Finally a dictionary of probabilities is returned with one entry for each class.

## Code Explanation: (Data Acquisation Part)

Data acquire from red pitaya: For supervise learning we first need to collect data from red pitaya to raspberry pi. Our main target is to recognize, human, wall and car. To collect data from red pitaya which will feed in machine learning part we have to run our “data\_get\_common\_code.py” file. This file will continuously receive signal data from red pitaya and store into a csv format file for future use. Let’s see how the data acquisition works.

In our script we have imported all the necessary library which will use while the script run. Figure 3 shows all the necessary library

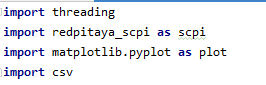


Fig-3

Now we will initialize scpi Object “rp\_s = scpi.scpi('192.168.128.1')” for sending scpi command to red pitaya. 192.168.128.1 is our red pitaya ip address.

The Standard Commands for Programmable Instruments (SCPI) defines a standard for syntax and commands to use in controlling programmable test and measurement devices, such as automatic test equipment and electronic test equipment

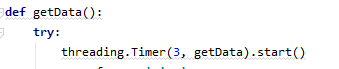


Fig-4

In this script (Fig-4) function getData() will trigger in every 3 second to get or acquire the signals from red pitaya. We used thread timer so that the method automatically run in very

given seconds.

In this function we have used different scpi commands to send to the red pitaya. Let’s see what are the commands we run sequentially to acquire data. The total function is given in below figure.

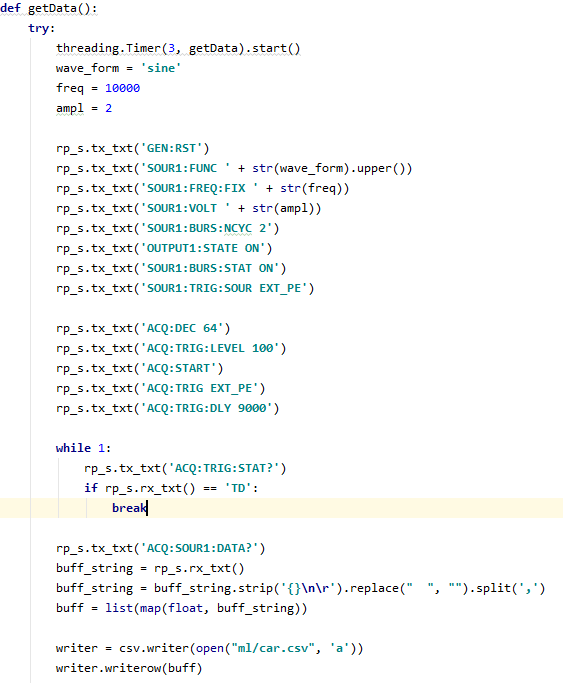
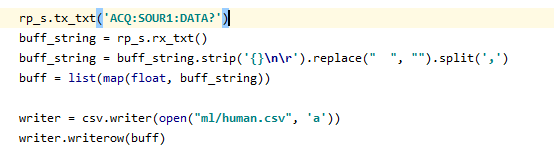


Fig-5

|  |  |
| --- | --- |
| Commands | What it does |
| GEN:RST | Reset generator to default settings. |
| SOUR1:FUNC | Set waveform of fast analog outputs. |
| SOUR1:FREQ:FIX | Set frequency |
| SOUR1:VOLT | Set amplitude voltage of fast analog outputs.Amplitude + offset value must be less than maximum output range ± 1V |
| SOUR1:BURS:NCYC 2 | Set N number of periods in one burst. |
| OUTPUT1:STATE ON | Enable fast analog outputs. |
| SOUR1:BURS:STAT ON | Enable burst (pulse) mode. Red Pitaya will generate R number of N periods of signal and then stop. Time between bursts is P. |
| SOUR1:TRIG:SOUR EXT\_PE | Set trigger source for selected signal. |
| ACQ:DEC 64 | Set decimation factor |
| ACQ:TRIG:LEVEL 100 | Set trigger level |
| ACQ:START | Starts acquisition |
| ACQ:TRIG EXT\_PE | Disable triggering, trigger immediately or set trigger source & edge. |
| ACQ:TRIG:DLY 9000 | Set trigger delay in samples |
| ACQ:TRIG:STAT? | Get trigger status |
| ACQ:SOUR1:DATA? | Read full buf |

Running these scpi command we continuously get buffered string data and store these in car.csv

We kept run the process until we received at least 100 data for car and then we continued the process to collect data for human and wall respectively. Code will be same except for human data file will be human.csv and for wall file will be wall.csv



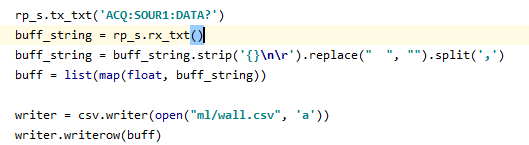


Fig-6

## Code Explanation: (All Used Libraries)

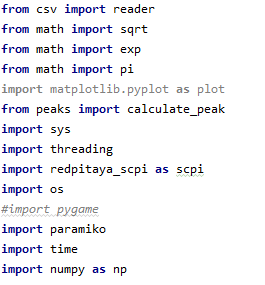


Fig-7

1. In our project we have to use different library to get advantage of some complex mechanism.
2. Csv, library to read data and write from csv files.
3. Math for mathematical squar root, exponential, pi and other operations in machine learning.
4. Matplotlib for plot the signal for more visualization.
5. Peaks is our own written python script.
6. Threading for running asynchronous process continuously. As we need to acquire signal from red pitaya in every n seconds.
7. redpitaya\_scpi to run all the scpi commands into red pitaya
8. pygame to play sound on the background
9. Finally, numpy for doing more complex stochastic operation for example Fourier Transform of a signal, mean, variance.

## Required Technologies

1. Python (Language) is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991. This scripting language is amazing to use for machine learning purpose as it has many machine learning library / api support. To write code in python we choose python version 3.7.3.

To make our solution running in python version 3.7.3 we have made our raspberry pi default python version from 2.7 to 3.7.3.

1. Paramiko install for ssh connection. As red pitaya starts with ssh connection and we can do it either manually connecting ssh connection or can make it automated in programming. To make it automated we installed paramiko library for establishing ssh connection between red pitaya and raspberry pi.
2. Matplotlib install to plot the signal coming from red pitaya. While doing the research with the signal or data we needed to plot to visualize the process.
3. Pygame install to play sound. Pip install pygame
4. Numpy is the fundamental package for scientific computing with Python. For example, we did fft for the signal and also got the peaks of the fft. These are the functionality we have achieved by numpy library.

## Code Explanation: (Feature Extraction Part)

Below Fig-8 code explain how to get feature for each file which we save after data acquisition. Read the data from “Car csv file” then for each data we calculated peaks, mean and variance We have set the label for each object here for Car label is 3.



Fig-8

Similarly, we have done for data file human and wall as shown in Fig 9 and Fig 10 respectively.



Fig 9



Fig 10

We have save data in csv file, here is separate file which calculate peaks shown in Fig-11



Fig-11

Below Fig-12 are the features extracted file Following are the features, Min peak, max peak,shape,mean,variance and label

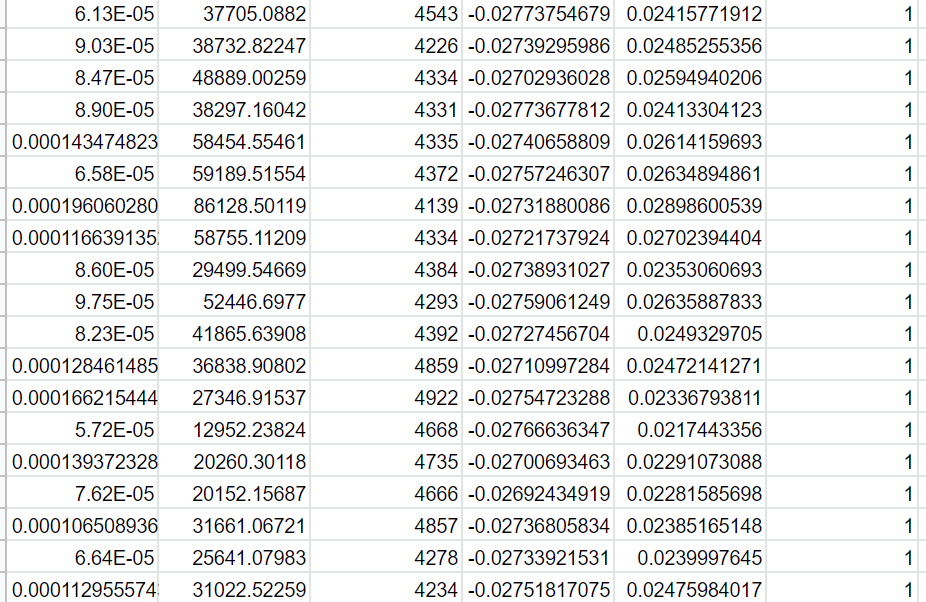


Fig-12

## Code Explanation: (Machine Learning Part)

Load data**:**First we load data from feature file in which we save all features.

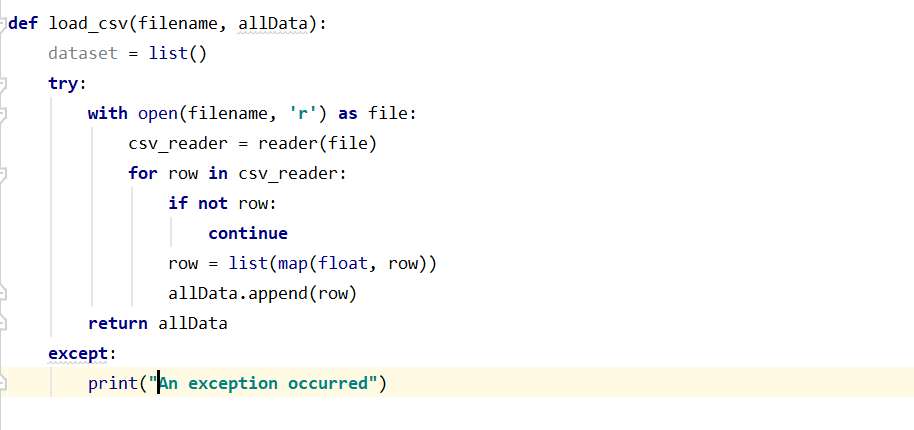


Fig-13

Then we separate data by class and save in dictionary shown in fig-14 and fig15:

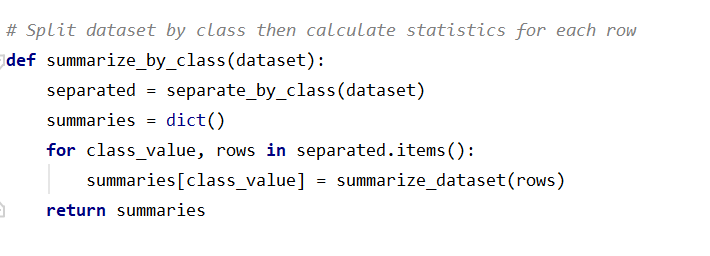


Fig-14

##### 

Fig-15

Then for each column we calculated mean, variance and standard deviation shown in fig 16:

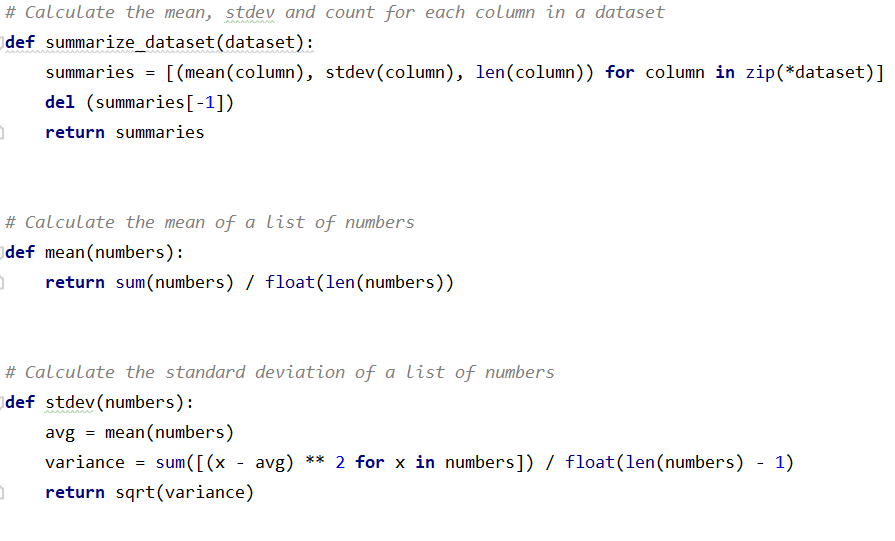


Fig-16

Till here we have trained our system, now we will call predict function to predict single data as shown in fig-17.

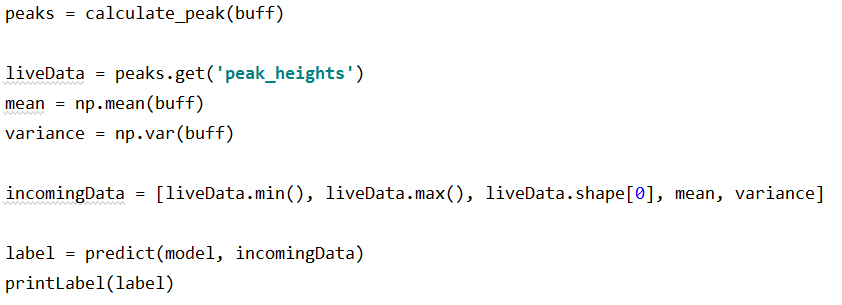


Fig-17

In predict function we will calculate Gaussian probability of this input data that how likely it is belongs to anyone of the trained class.

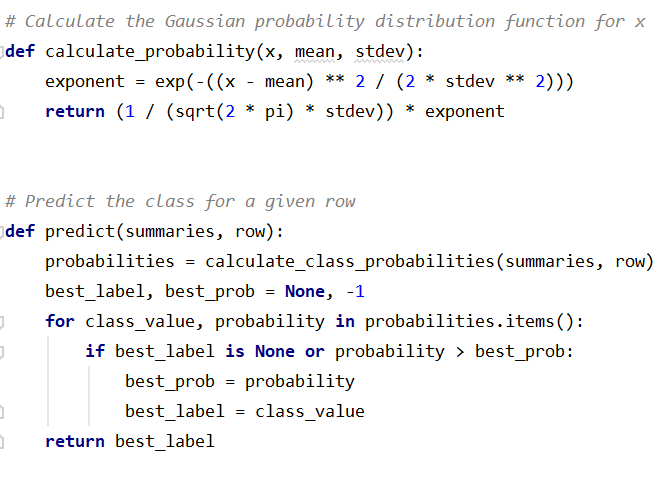


Fig-18

To show the result We have use two approaches.1 to print label on screen. 2 to play audio for object name

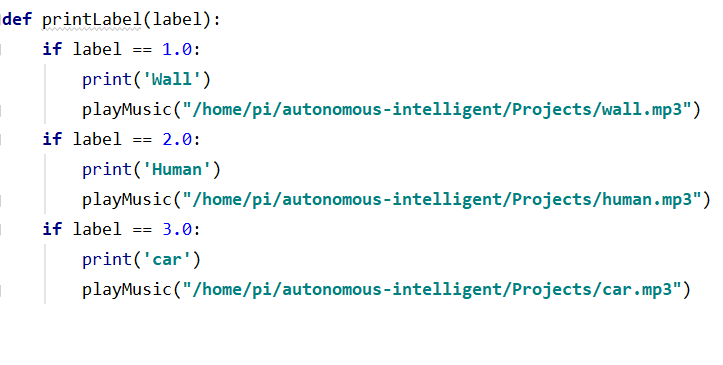


Fig-19

## Testing: (Plug & Play)

To test our project just power raspberry pie and red pitaya and wait for 40 seconds. Waiting for 40 second is important because raspberry pie files, folder, services, Wi-Fi service all important services should be ready. We forced our script to wait for raspberry pie to get ready. Point the red pitaya to a wall or human or a car. A monitor can be connected with raspberry pie but it is not mandatory. We have 2 kind of output one is sound and other is printed name of the object. If we have any monitor connected with raspberry pie just open the terminal that’s it, otherwise just connect a headphone or a speaker with raspberry pie to listen which object is detecting.

Our object detection python code is completely automated and enough stable to predict Human, Wall and Car. To make the object detection code automated we added the object\_detection.py path in “/home/pi/.bashrc”

In bashrc last line we added sudo python /home/pi/object\_detection.py

Now whenever the raspberry pie will boot or the terminal open the object\_detection python script will run. As we continuously acquire data from red pitaya we must have a stable ssh connection between raspberry pie and red pitaya. Instead of manually doing it paramiko library helps us to make an automated stable ssh connection



Fig-20

We initially try to make ssh connection using ssh\_connection() function but if it fails due to any circumstances for instance Wi-Fi connection, it will wait and continue to try in every 5 seconds. Finally, if it will success to make ssh connection it will never try again. This is our major goal to make our project stable, as stability is major concern in this software.

## Results Accuracy And Precision

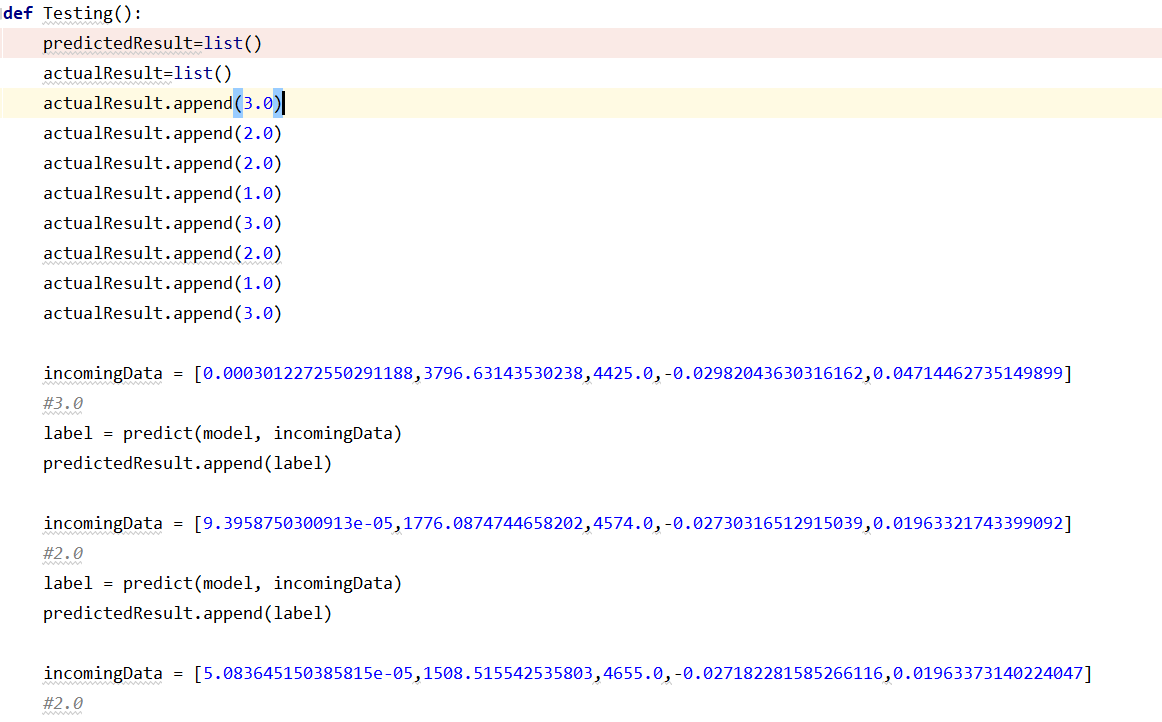


Fig-20

To test how much accurate our software is we ran test in few samples before training our modle we pick few data.as we already know the expected result, as have compare the actual result with predicted result and we concule with 87% of accurance as shown in fig-21

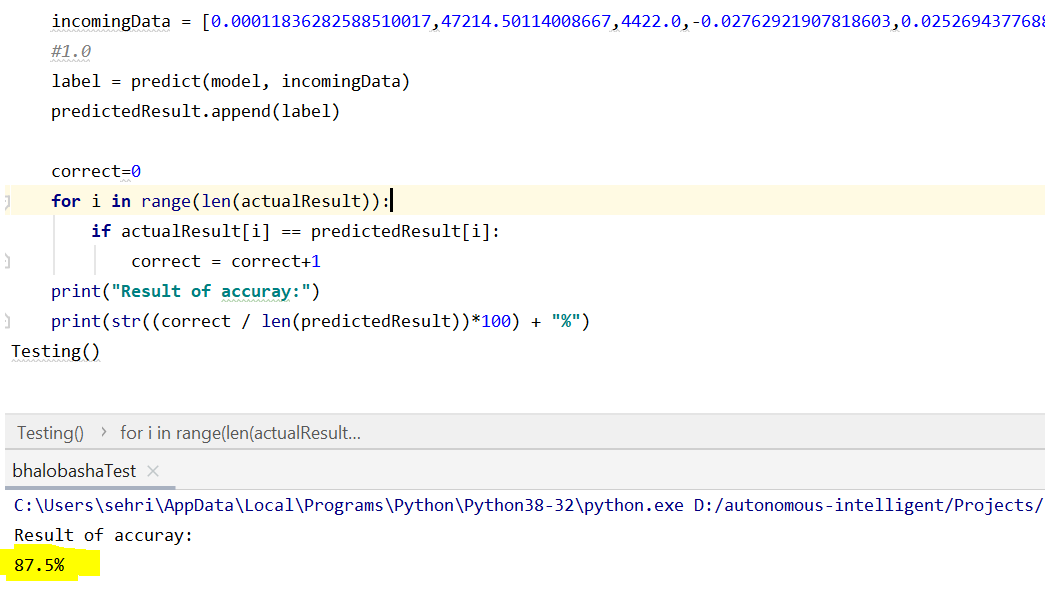


Fig-21

##### Acknowledgment

To summarize, this project is for getting the data from red pitaya sensor and transfer data to raspberry and it create stable connection between raspberry and red pitaya. while raspberry is already containing the script for feature extraction and training the data for three objects that is Car, Wall and Human as soon as it gets the data from red pitaya it runs the machine learning and check the probability how likely it is related to which class that we trained already.

Main objective was to achieve a stable solution between these two devices and create optimize solution for object detection.

##### References

1. Bernadette Johnson “How the Raspberry Pi Works” 30 August 2012 *(*[*references*](https://computer.howstuffworks.com/raspberry-pi2.htm)*)*
2. Jason Brownlee on October 18, 2019 in Code Algorithms From Scratch ([*references*](https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/)).
3. Trenz Electronic , “Red Pitaya Product” ([*references*](https://shop.trenz-electronic.de/en/Products/Red-Pitaya/))
4. Dexter Industries., “Five Ways To Run a Program On Your Raspberry Pi At Startup” ([*references*](https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/)).
5. Red Pitaya d.d. Revision a25e54a3, “STEMlab boards comparison.” 2017 ([*references*](https://redpitaya.readthedocs.io/en/latest/developerGuide/125-10/vs.html)).