NAME	ID
AHMED SAMY GOMMA	32
ANDROW MORKOS FEHMY	56
HASSAN ABD EL KAREEM KASSEM MOSTAFA	82
REHAB SAMY MOHAMED	96
ABDELRAHMAN IBRAHIM HOSNEY	132
ABDEL SALMA MAHMOULD AB DEL SALAM	144
MOHAMED SALAH MOHAMED	224
MOSTAFA ALI EL SAIED	264
MOHAB AHMED RAMADAN	274
HAGER IBRAHIM SAAD	299
HEBA SAFWAT ABDEL GHANY	305

Acknowledgement

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. A special gratitude we give to our project head and supervising doctor, Prof. Dr. Mona Lotfi.

Whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this book.

Furthermore, we would also like to acknowledge with much appreciation the crucial role of academic staff who contributed in teaching us the fundamentals of electronic engineering.

Moreover, they build our mentality to achieve our current result, in addition to appreciate the guidance given by other assistant lecturers who provided us with valuable information and explanation for the theory behind our project.

Last but not least, many thanks go to the head of the project, Prof .Dr. Mona Lotfi, who have invested her full effort in guiding the team in achieving the goal.

Table of Contents

Table of figures

1

Project Initiating

1.1. Introduction

Driver assistance systems are an indispensable part of modern cars and in the foreseeable future will further develop into complex systems leaving the task of driving and liability to the vehicle. Depending on the system and stage of development, active driver participation when desired will be required less and less, and eventually not at all. We will introduce to you the most popular and innovative driver assistance systems and explain which systems are suitable in which circumstances.

What is a driver assistance system? Driver assistance systems relieve the driver of the task of driving, offer more comfort and increase safety. In an emergency, a driver assistance system can even take control of the car. Liability for the task of driving always remains with the driver.

there for, Cars are becoming ever more connected to other cars, to transport infrastructure, to pedestrians, and to datacenters. But which standards will underpin this entire vehicle to everything (V2X) communication.

Unless you're a hardcore vintage car enthusiast, you'll have noticed that vehicles are becoming increasingly connected, both to each other and to the outside world.
With car operating systems running everything from infotainment to autonomous driving, vehicles are becoming ever more intelligent and less reliant on human operation. Vehicle users stand to benefit from safer, greener, and more efficient journeys thanks to copious sensors and onboard connectivity, while car manufacturers, tech companies, and communications providers have a whole new market to compete in.
V2X, which stands for 'vehicle to everything', is the umbrella term for the car's communication system, where information from sensors and other sources travels via high-bandwidth, low-latency, high-reliability links, paving the way to fully autonomous driving.
There are several components of V2X, including vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-pedestrian (V2P), and vehicle-to-network (V2N) communications. In this multifaceted ecosystem, cars will talk to other cars, to infrastructure such as traffic lights or parking spaces, to smartphone-toting pedestrians, and to datacenters via

cellular networks. Different use cases will have different sets of requirements, which the communications system must handle efficiently and cost-effectively.

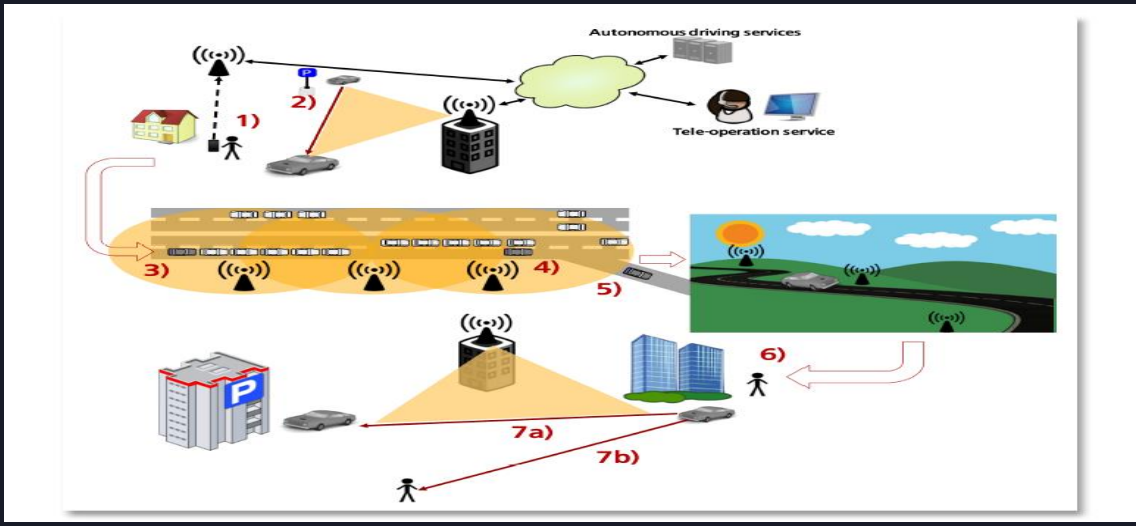


Figure 1: V2X Communication

1.2. Problem Definition

roads. Moreover, travelling between distant cities take hours and sometimes drivers lose their concentration. According to statistics: 78.9% of the road accidents in 2017 were caused by human error. So, in order to achieve safe and secure life, we concentrate on solving some of these problems like the detection of speed bumps which the driver may not see or take attention about. The second problem is the driver who doesn't keep his lane, and as a result, his vehicle may collide with other vehicles, in addition to the upcoming traffic light problem.

1.3. Literature Review

New cars can be equipped with many advanced safety solutions. Airbags, seatbelts, and all of the essential passive safety parts are standard equipment.

Now cars are often equipped with new advanced active safety systems that can prevent accidents. The functions of the Advanced Driver Assistance Systems are still growing.

Many of these accidents could be avoided if the automatic systems were used to help humans when braking.

Advanced Drive Assistance Systems (ADAS) cannot completely prevent accidents, but it could better protect us from some of the human factors and human error is the cause of most traffic accidents.

To counteract traffic accidents, we could change human behavior, adopt vehicle-related measures and a physical road infrastructure related measures.

Another approach is to transition from a passive safety measures to the active safety measures.

- Passive safety measures include airbags, the structure of car body, seatbelts and head restraints.
- Active safety measures include electronic stability control (ESC), anti-lock braking systems (ABS) and other Advanced Driver Assistance Systems like intersection collision avoidance (ICA) and line keeping assistant (LKA).

Now, in order to avoid accidents, we can use ICT (Information and Communication Technology) based ADAS applications, which continuously assist drivers with their driving tasks.

Also, In order to prevent traffic accidents, a roadmap for fully autonomous driving has been created. The development of driver assistance systems began with Anti-lock

Braking System (ABS) introduced into a serial production in the late 70s of the twentieth century. A roadmap concept consists of the following sensors:

- proprioceptive sensors - able to detect and respond to danger situation by analyzing the behavior of the vehicle;
- exteroceptive sensors – (e.g. ultrasonic, radar, lidar, infrared and vision sensors) able to respond on an earlier stage and to predict possible dangers;
- sensor networks - application of multisensory platforms and traffic sensor networks.

ADAS provides additional information from the car surrounding environment to support a driver and assist in implementing critical actions.

The synchronization of a driver's actions and the information from the environment is essential for the efficient performance of the various
applications of ADAS.

In the Adaptive Cruise Control system (ACC), three radar sensors are usually needed because two short range radars are used to detect objects in the adjacent lane and one long range radar is used to detect objects in-path.

As Driver Assistant Systems (DAS) and Active Safety Vehicles (ASV) with various functions become popular, it is not uncommon for multiple systems to be installed on a vehicle.

If each function uses its own sensors and processing unit, it will make installation difficult and raise the cost of the vehicle.

As a countermeasure, research integrating multiple functions into a single system has been pursued and is expected to make installation easier, decrease power consumption and vehicle pricing.
Road boundaries can give useful information for evaluating safe vehicle paths in intelligent vehicles.

Much previous research has studied road boundary detection, using different types of sensors such as vision, radar, and lidar. Lidar sensors, in particular, show advantages for road boundary extraction including high resolution and wide field of view. However, none of the previous studies examined the problem of detecting road boundaries when roads could be either structured or unstructured.

Traffic Sign Recognition is a display on the instrument panel that reminds drivers of the current speed limit.

This is achieved through the use of the same camera system that can also recognize speed limit signs.

Navigation systems also support this solution by storing information about the speed limit on an unsigned road.

The curve warning and Adaptive Cruise Control are examples of ADAS application that use this information.

Review Conclusion

The ADAS from different manufacturers that are described above are slowly becoming standard equipment and are now produced for more expensive vehicles or luxury vehicles. These systems can perform a greater number of functions.

Although the use of the ADAS greatly increases road safety, it is important for drivers to learn how it works and is used. In time users will trust these systems.

It is more important that ADAS manufacturers anticipate how these systems are used in a number of situations on the road.

In addition to testing individual systems, it is increasingly important to prepare tests of all of the systems that operate in parallel.
The use of an increasing number of sensors and ADAS additionally enables the implementation of advanced inference systems.

It also allows the construction of these systems based on artificial intelligence, which is a step towards automated driving.

The progress can be observed through the publications, contests and mass media. However, the details of these solutions are usually kept secret and little information of this type is available.

1.4. V2V Market

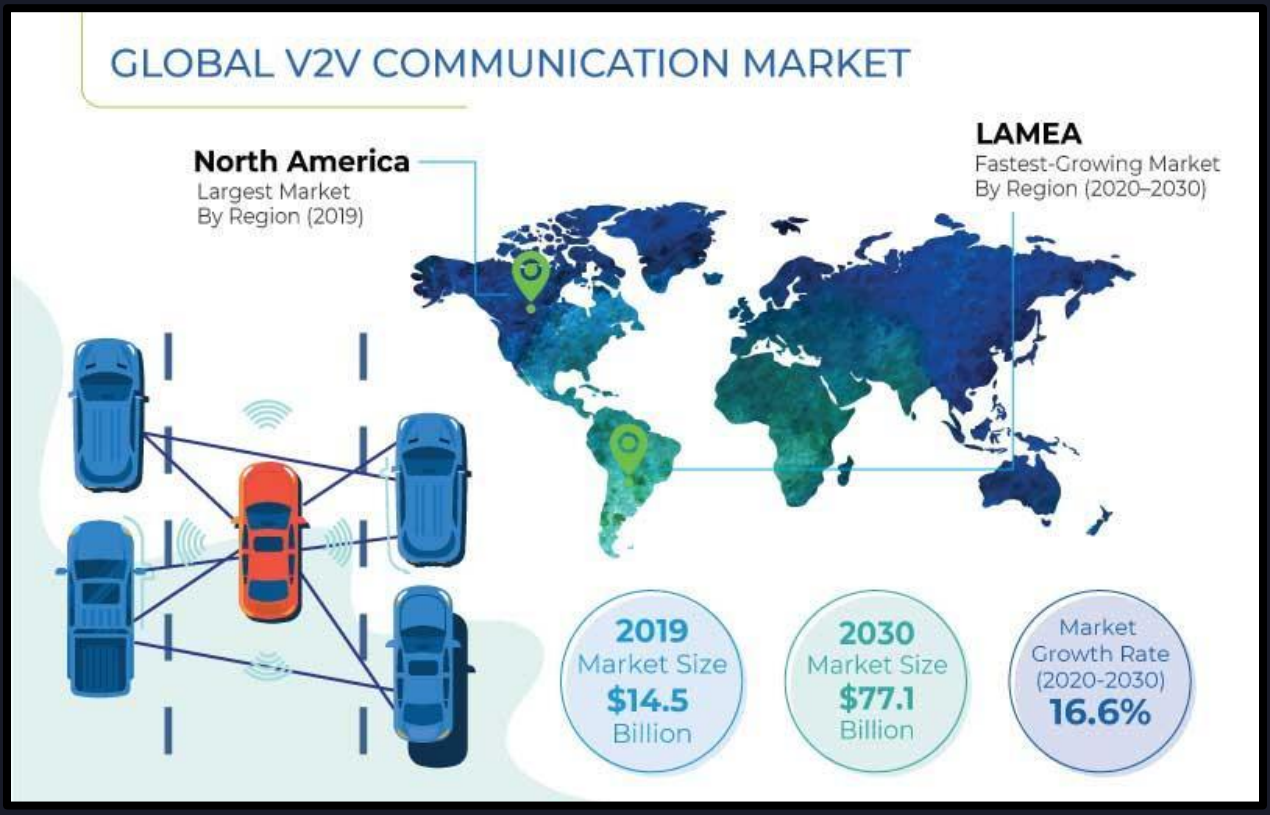


Figure 2: V2V Market

Vehicle to Vehicle (V2V) Communication Market size is set to account for phenomenal gains in through 2027 owing to increasing mortality rates as a result of road accidents, traffic congestion, and the technology's ability to reduce human errors. A WHO report suggests that approximately 1.3 million people die every year due to road traffic crashes.

V2V communication enables cars and other vehicles to communicate seamlessly with each other and other infrastructures like traffic lights. It also warns drivers about the blond spots, vehicles coming to a sudden stop, and other such scenarios that could lead to accidents. However, this technology is only likely to work between two vehicles that have the V2V technology installed.

Importantly, V2V communication technology operates up to 1,000 feet by providing drivers sufficient time to respond. Moreover, it makes use of dedicated short-range communication, a wireless network similar to Wi-Fi.

The escalating trend of advanced driver assistance systems (ADAS) is expected to have a positive influence on V2V communication market growth over 2021 to 2027. The market has already been observing an elevating adoption of ADAS, which has quite transformed the face of driving experience. It would be essential to note that the ADAS industry, in 2020, generated around $17.6 billion globally. This has urged various automakers to invest in V2V communication

solutions. For instance, Auto talks Ltd. offers V2V solutions for mobility and safety applications. Through this offering, the vehicle shares drive information like location, speed, path prediction, and other crucial data to other vehicles on road.

Based on the end-user segmentation, V2V communication market is characterized by passenger vehicle segment. The growth can potentially be ascribed to burgeoning sales of passenger vehicles worldwide as a result of gradual spike in disposable incomes of global population. According to OICA statistics, the sales of PVs reached to 63 million in 2019, only declining to 53 million in 2020 due to the coronavirus outbreak.

V2V communications market from commercial vehicles is also set to record massive impetus in the coming years, driven by pivotal factors including stringent government regulations and rise in adoption of cloud-based fleet management telematics solutions in various developed economies.

The aftermarket segment is expected to play a vital role in enhancing the growth prospects of vehicle-to-vehicle communications market. This can be reasoned to the inclination of older model vehicle owners to aftermarket solutions to upgrade and update their offerings in the cars. It has been reported that already several of top vehicle models in the UAE feature some sort of vehicle communication technology on existing models or have laid out plans on doing so.

Geographically, Asia Pacific V2V communication market is slated to garner massive momentum through 2027, driven by the growing trend of ADAS adoption, increasing production and sales of passenger and commercial vehicles, and rising awareness towards traffic congestion and accidents. Likewise, the Middle and East, and LATAM countries are also likely to account for major growth in the industry, fueled by the escalating deployment of vehicle communication technologies.

Some of the prominent leaders partaking in vehicle to vehicle (V2V) communication market are Daimler AG, BMW Group, Volkswagen group (Porsche SE), Toyota (Toyota Group), General Motors, Qualcomm, and others. These vehicle-to-vehicle communication market vendors are introducing novel innovations in the space, in an attempt to reap huge returns to the overall industry. Besides, they have also been indulging in various marketing strategies like acquisition, collaborations, and partnerships, to gain a competitive edge.

For instance, in 2020, General Motors unveiled V2X technology in China. The technology includes vehicle to vehicle and vehicle to infrastructure communication systems and bears the potential to alleviate traffic congestion while simultaneously augmenting the safety standards.

In yet another instance, Volkswagen, in 2019, announced building up a new tech system for Golf countries that would be able to communicate to other cars and infrastructure.

Analyzing the impact of Covid-19 on V2V communication market growth, the coronavirus pandemic has had a negative influence on growth statistics of various crucial industries. The auto industry is no exception. The automotive industry observed a drastic decline in terms of production, sales, and supply chain demand. Owing to these measures, the V2V communication market is also predicted to observe inconsistent growth over the coming years.

1.5 . Project Objectives

Over the past few years, we have witnessed an increasing research interest and efforts in the development of connected and intelligent vehicles. There have been significant advancements in electronics aiding car drivers, along with vehicle-to-everything (V2X) communications to enable mobile Internet and entertainment. However, automotive and transportation engineers have a very limited understanding of the way in which security breaches may occur, and what the best practices are to enforce security requirements in future vehicular networks. The novelty of this project is to integrate pervasive vehicular connectivity, i.e., V2X, and new features of communications infrastructure and cryptography technologies, into the development of a cross-layer security for vehicular networks and, in particular, ensuring compliance of mission-critical requirements of vehicular applications.

We wanted our project to fulfill the user's needs and expectations, thus we had some vision of these needs and planned its scope and our approach to satisfy it.

- Regulation of cars motion so that no car can't detect the other cars motion information in its prespecified range.
- Guaranteeing safety to people on the road.

1.6 . Stakeholders

Advanced Driver Assistant Systems (ADAS) are generally considered as most promising for reducing road transport externalities (e.g. congestion, traffic accidents, and environmental stress(. So in general, main groups of stakeholders that benefits from this system: The automobile industry, owners of Passenger Cars& Commercial Vehicles, the drivers, the passenger, each person walking or driving along the road.

1.7. Scope

Our project works as a driving assisting system using image processing and various embedded sensors to process the interfacing objects of the car and take various actions and alerts to it.

The project is introduced by a car controlled remotely to move in a model simulating the real life environment.

Our Driver Assistant Systen helps the driver to:
• Detect the upcoming un-marked and marked speed hump/bump.
• Inform the driver of the traffic lights.
• Notify the driver with an LCD and take action in case of emergency break
• upload the data to a server for the upcoming cars

So, we have trained a tensor flow model that detects seven objects (Speed pump, Traffic light , Person, Lane, Obstacles, Lampposts, Bins)

We will concentrate in our project on speed pump feature where as soon as the camera detects the speed pump it will use the ultrasonic sensor to calculate the distance to the pump.

With this information our robot could take the proper action by decreasing the speed or using the breaks. Moreover, the detected objects' name will be displayed on the lcd to notify the driver.
In order to ensure the flow of data between cars the data is uploaded by an esp-wifi module to a server that is going to prosses the data and send it to the upcoming vehicles that would need that information.
There is a future improvement, that the data can be transferred from a vehicle to another to perform a chained action to prevent accidents (V2V).

1.8. Constraints

- As vehicles in V2X are connected to internet, they are prone to hacking. Hackers can access and control the vehicle.
- Privacy of the owners and users of the vehicles are major concern. This is due to leakage of information such as vehicle location, daily routine, frequently used apps etc. This hacked information can be used for unauthorized purposes. These can also be used by organizations and government agencies.
- Autonomous driver system can cause fatal consequences during failure of the system.
- Malfunctioning of cars or sensors or networks lead to incorrect data. This leads to faulty communications.

2

Planning and Requirement

2. System Components

2.1. Hardware Component

2.1.1. TivaC (TM4C123GH6PM)

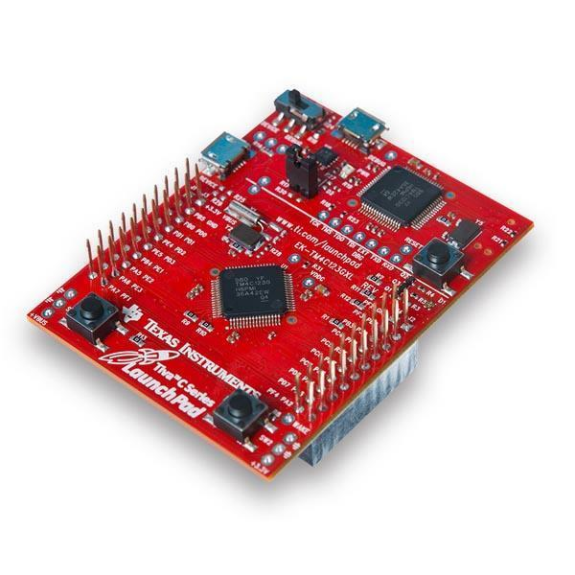


Figure 3: TivaC

- 32-bit ARM® Cortex™-M4 80-MHz processor core with System Timer (SysTick), integrated Nested Vectored Interrupt Controller (NVIC), Wake-Up Interrupt Controller (WIC) with clock gating, Memory Protection Unit (MPU), IEEE754-compliant single-precision Floating-Point Unit (FPU), Embedded Trace Macro and Trace Port, System Control Block (SCB) and Thumb-2 instruction set
- On-chip memory, featuring 256 KB single-cycle Flash up to 40 MHz (a prefetch buffer improves performance above 40 MHz), 32 KB single-cycle SRAM; internal ROM loaded with TivaWare™ for C Series software; 2KB EEPROM
- Two Controller Area Network (CAN) modules, using CAN protocol version 2.0 part A/B and with bit rates up to 1 Mbps
- Universal Serial Bus (USB) controller with USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation, 32 endpoints, and USB OTG/Host/Device mode
- Advanced serial integration, featuring: eight UARTs with IrDA, 9-bit, and ISO 7816 support (one UART with modem status and modem flow control); four Synchronous Serial Interface (SSI) modules, supporting operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces; four Inter-Integrated Circuit (I2C) modules, providing Standard (100 Kbps) and Fast (400 Kbps) transmission and support for sending and receiving data as either a master or a slave
- ARM PrimeCell® 32-channel configurable µDMA controller, providing a way to offload data transfer tasks from the Cortex™-M4 processor, allowing for more efficient use of the processor and the available bus bandwidth
- Analog support, featuring: two 12-bit Analog-to-Digital Converters (ADC) with 12 analog input channels and a sample rate of one million samples/second; two analog comparators; 16 digital comparators; on-chip voltage regulator
- Advanced motion control, featuring: eight Pulse Width Modulation (PWM) generator blocks, each with one 16-bit counter, two PWM comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector; two PWM fault inputs to

promote low-latency shutdown; two Quadrature Encoder Interface (QEI) modules, with position integrator to track encoder position and velocity capture using built-in timer

- Two ARM FiRM-compliant watchdog timers; six 32-bit general-purpose timers (up to twelve 16-bit); six wide 64-bit general-purpose timers (up to twelve 32-bit); 12 16/32-bit and 12 32/64-bit Capture Compare PWM (CCP) pins
- Up to 43 GPIOs (depending on configuration), with programmable control for GPIO interrupts and pad configuration, and highly flexible pin muxing
- Lower-power battery-backed Hibernation module with Real-Time Clock
- Multiple clock sources for microcontroller system clock: Precision Oscillator (PIOSC), Main Oscillator (MOSC), 32.768-kHz external oscillator for the Hibernation Module, and Internal 30-kHz Oscillator
- Full-featured debug solution with debug access via JTAG and Serial Wire interfaces, and IEEE 1149.1-1990 compliant Test Access Port (TAP) controller
- Industrial-range (-40°C to 85°C) RoHS-compliant 64-pin LQFP

2.1.2. Rasberry Pi 4 model B

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.1, Vulkan 1.0

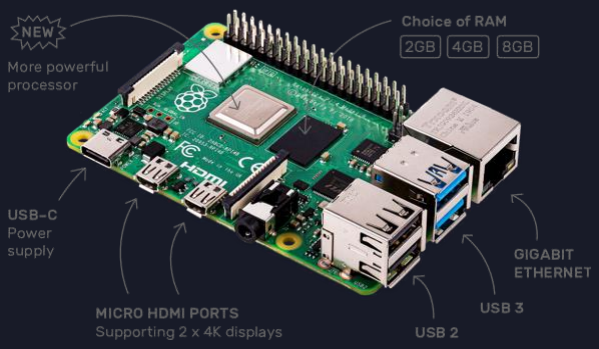


Figure 4: Raspberry Pi

- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient
- * A good quality 2.5A power supply can be used if downstream USB peripherals consume less than 500mA in total.

2.1.3. Wifi Module (Node MCU esp8266)

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

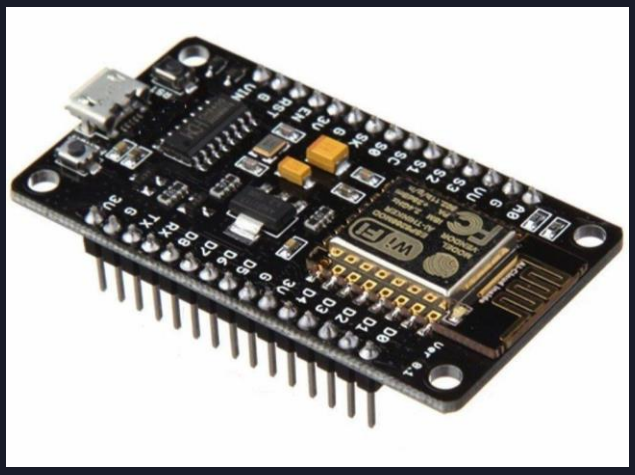


Figure 5: WiFi Module

2.1.4. GPS module (ublox Neo-6m)

Figure 6: GPS Module

- Cold start time of 38 s and Hot start time of 1 s
- Supply voltage: 3.3 V t
- Configurable from 4800 Baud to 115200 Baud rates. (default 9600)
- SuperSense ® Indoor GPS: -162 dBm tracking sensitivity
- 5Hz position update rate
- Operating temperature range: -40 TO 85°C
- UART TTL socket
- EEprom to save configuration settings
- Rechargeable battery for Backup
- Separated 18X18mm GPS antenna
- Dimension: 22X30X13 mm
- Weight: 12g

2.1.5. Bluetooth Module

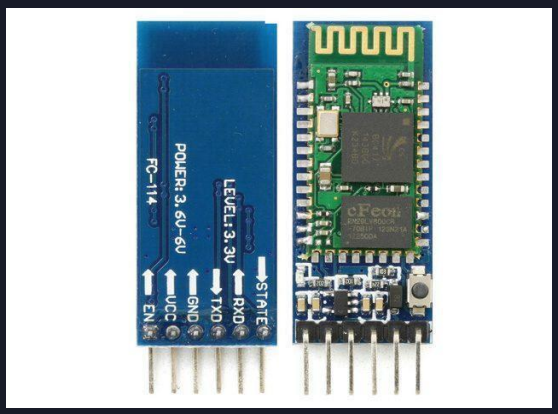


Figure 7: Bluetooth Module

- Serial Bluetooth module for Arduino and other microcontrollers
- Operating Voltage: 4V to 6V (Typically +5V)
- Operating Current: 30mA
- Range: <100m
- Works with Serial communication (USART) and TTL compatible
- Follows IEEE 802.15.1 standardized protocol
- Uses Frequency-Hopping Spread spectrum (FHSS)
- Can operate in Master, Slave or Master/Slave mode
- Can be easily interfaced with Laptop or Mobile phones with Bluetooth
- Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.

2.1.6. Ultrasonic (HCSR04)

- Input Voltage: 5V
- Current Draw: 20mA (Max)
- Digital Output: 5V
- Digital Output: 0V (Low)
- Working Temperature: -15°C to 70°C
- Sensing Angle: 30° Cone
- Angle of Effect: 15° Cone
- Ultrasonic Frequency: 40kHz
- Range: 2cm - 400cm
- Dimensions
- Length: 43mm
- Width: 20mm
- Height (with transmitters): 15mm
- Centre screw hole distance: 40mm x 15mm
- Screw hole diameter: 1mm (M1)
- Transmitter diameter: 8mm

Figure 8: Ultrasonic Sensor

2.1.7. Motor and Motor driver (L298N)

5v DC motor

Figure 9: DC Motor

L298N Basic properties

- Module Name: dual H-bridge motor driver module
- Work mode: H-bridge driver (Dual)
- Master chip: L298N
- Logic voltage: 5V
- Drive voltage: 5V-35V
- Logic Current: 0mA-36mA

- Drive Current: 2A (MAX single bridge)
- Storage Temperature:-20C to +135 C
- Maximum power: 25W
- Weight: 30g
- External dimensions: 43 * 43 * 27mm
- Product Features:
- This module uses ST's L298N chip as the main driver, with driving ability, low heat, strong anti-interference ability characteristics.

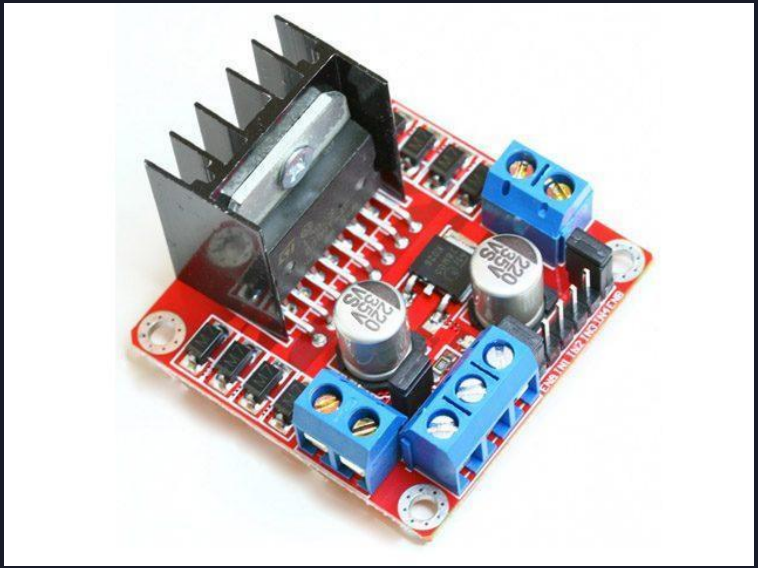


Figure 10: Motor Driver

- This module can use the built-78M05 take power by driving the power part of the work, but in order to avoid damage to the regulator chip, when the drive voltage is greater than 12V, please use an external 5V logic supply.
- This module uses high-capacity filter capacitors, wheeling protection diodes can improve reliability.

2.1.8. Speed Sensor (LM393 module)

- Main technical characteristics:
- Dimensions: 32 x 14 x 7mm.
- The sensor reading slot has a width of 5mm.
- Two outputs, one Digital and one Analog.
- LED power indicator.
- LED indicator of the output pulses of pin D0.
- Features
- Using imported trough type optical coupling sensor, groove width 5 mm.

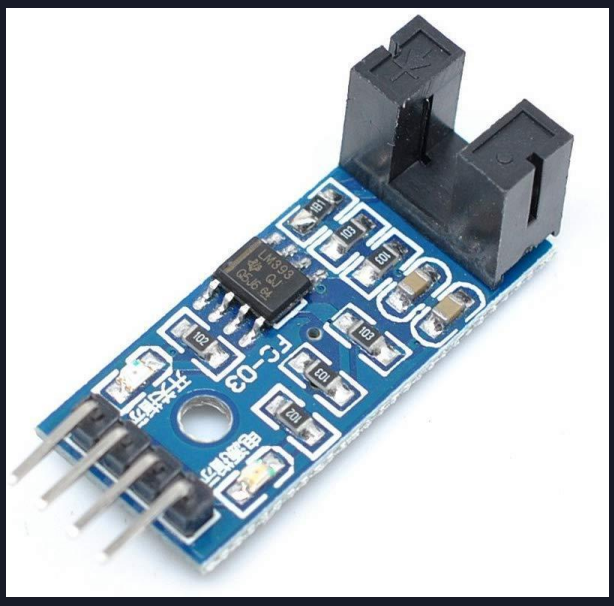


Figure 11: Speed Sensor

- The output state light, lamp output level, the output low level light.
- Covered : output high level; Without sunscreen : the output low level.
- The comparator output, signal clean, good waveform, driving ability is strong, for more than 15 ma.

- The working voltage of 3.3 V to 5 V
- Output form: digital switch output (0 and 1)
- A fixed bolt hole, convenient installation
- Small board PCB size: 3.2 cm x 1.4 cm
- Use the LM393 wide voltage comparator Module

2.1.9. Display (16x4 LCD)

Figure 12: LCD

- Display Format 16×4 Character
- Sunlight Readable No
- Touch Panel Optional No
- Outline Dimension 87.0(W)x60.0(H)x12.5(T)mm
- Visual Area 61.70×25.20mm
- Active Area 56.20×20.80mm
- Character Size 2.95×4.75mm
- Diagonal Size No
- Dot (Pixel) Size 0.55×0.55mm
- Dot Pitch 0.60×0.60mm
- IC Package COB
- IC or Equivalent AIP31066 , HD44780, KS0066 , SPLC780 , ST7066
- Interface 6800 4-bit Parallel , 6800 8-bit Parallel
- Display Type STN-LCD Yellow
- Response Time(Typ) No
- Contrast Ratio(Typ) No
- Colors No
- Viewing Direction 6:00
- Viewing Angle Range No
- Backlight Color Blue Color

2.1.10. Camera (Microsoft Webcap HD3000)

- 720p HD Video, 30 fps
- 16:9 Widescreen
- Integrated Omnidirectional Microphone
- TrueColor Technology, 24-bit Color Depth
- Movie Maker & Photo Gallery Software
- Skype Certified
- Compatible with Windows 7, 8, 8.1 & 10

Figure 13: Microsoft Webcam

2.1.11. Car Body & PCB

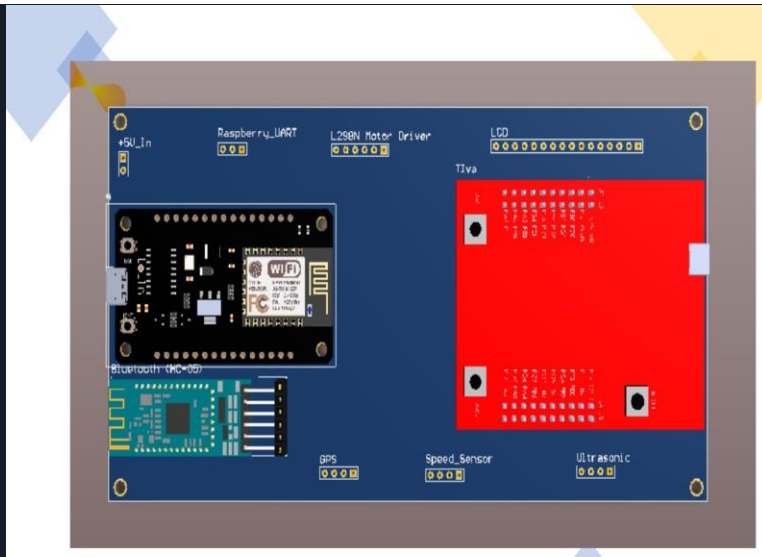


Figure 14: Car Circuit PCB

2.1.12. Objects

Figure 15: Training Images

- Traffic light
- Person
- Speed Pump
- Lane
- Lamp post
- Rubbish Bin
- Obstacle

2.2. Software components

2.2.1. Embedded part

● C language for TivaC programming

Although not originally designed for embedded software development, the C language allows a range of programming styles from high-level application code down to direct low-level manipulation of hardware registers. As a result, C has become the most popular programming language for embedded systems today.
This webinar is intended for anyone who is new to embedded systems or who wishes to start using the C language for an embedded application.

You can expect to learn about:

- Alternative approaches and languages used for embedded software development.
- The features that make C the first choice programming language for both large and small embedded systems.
- Details about key C language constructs used to write code for essential embedded system tasks such as:
    - accessing data in memory and registers;
    - working with interrupts;
    - driving peripheral devices.

- **Mobile App for Bluetooth based pilot**

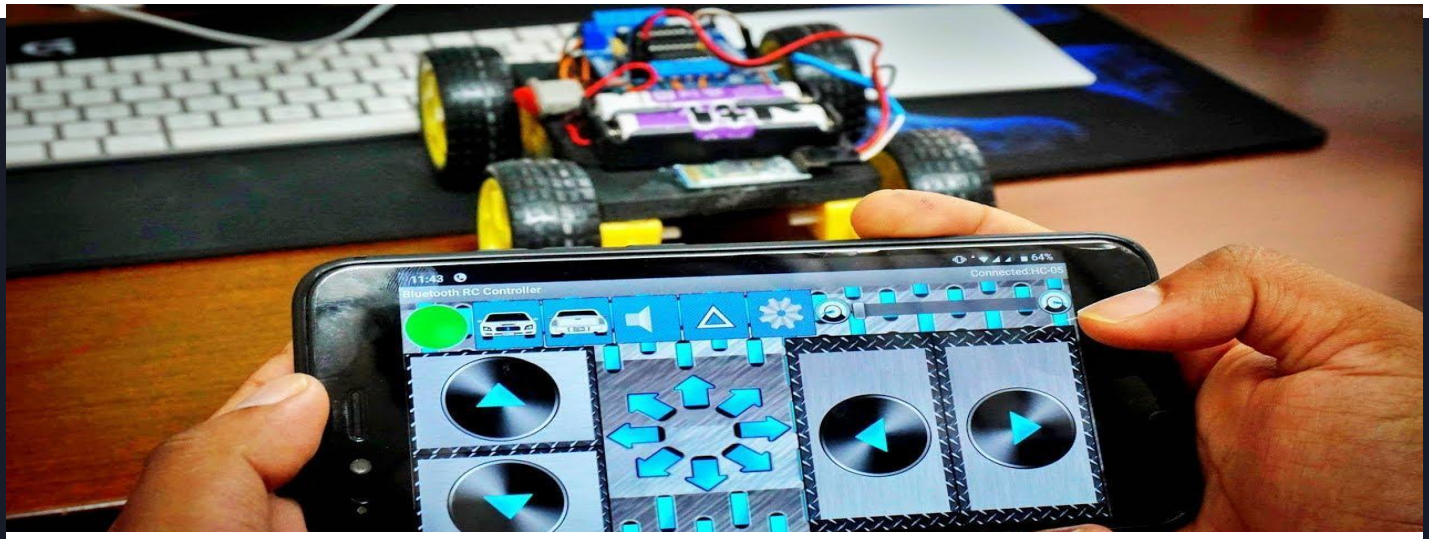


Figure 16: RC bluetooth App

- **Web Server for the V2X communication**
- **PCB Design Tool**
- **Altium Designer**

Figure 17: Altium Designer

2.2.2. Computer Vision part

- Anaconda

Figure 18: Anaconda Environment

- Python 3.7.0

Figure 19: Python

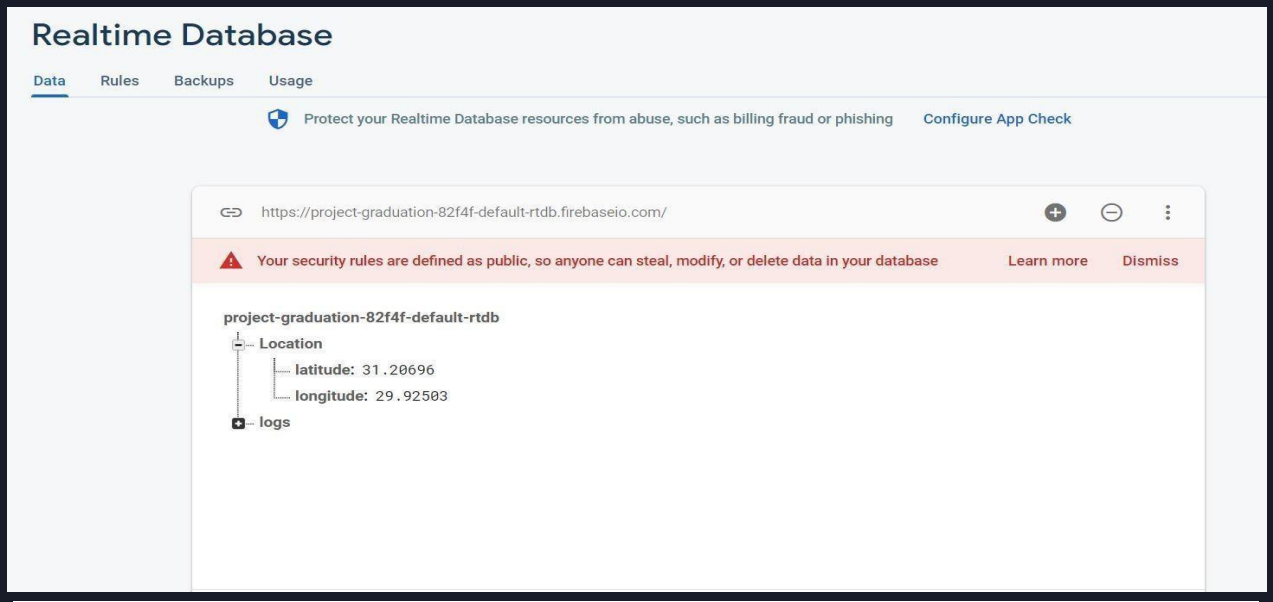


Figure 20: Real-time Database

- TensorFlow 1.15.0

Figure 21: Tensorflow

- Raspbian OS

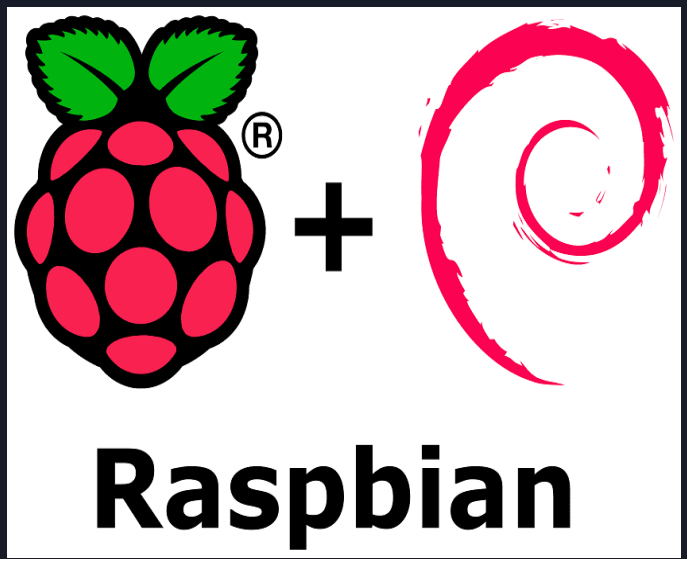


Figure 22: Rasbian OS

2.3. Risk List

2.3.1. Risks on the system

- **Hardware component failure**

Failure in any critical hardware component would result in malfunction in the system and could result to unexpected output that could put the vehicle at risk.

- **Communication interruption between the web server and the car**

Miscommunication between the vehicle and the server would leave some of the pump detection locations off the server data base.

- **Environmental obstacle blocking the ultrasonic vision**

Random obstacles block the vision of the ultrasonic sensor would disable the system from reading the distance between the vehicle and the pump. Which might result in damages to the vehicle if the driver is not aware to it.

2.3.2. Risks on the user

- **Failure from the camera to detect the pump**
- **System malfunction might result in delayed or no action towards the pump**

2.4. System Requirements

2.4.1. System block diagram

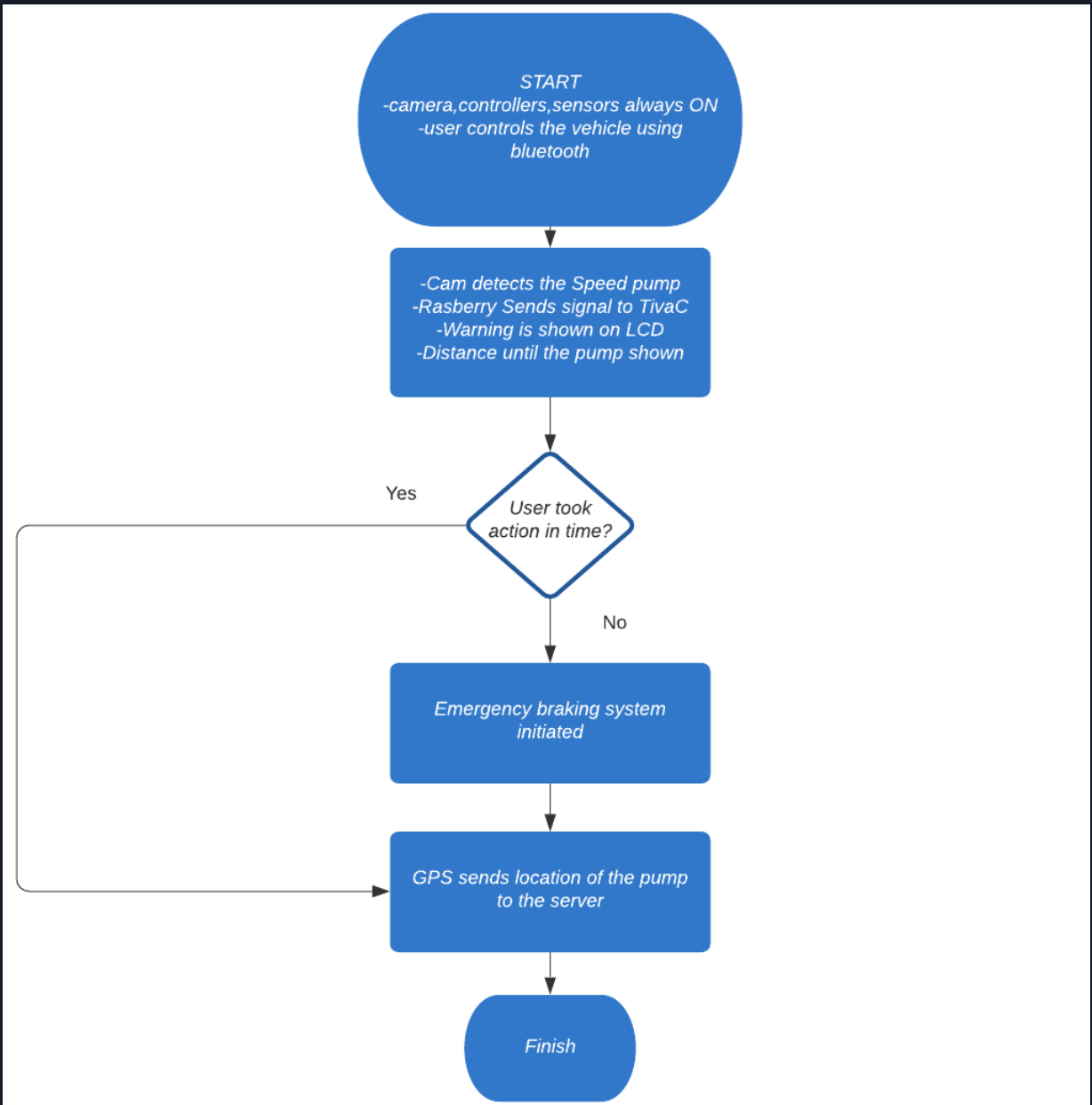


Figure 23: System Block Diagram

2.4.2. Functional Requirements

- Microcontrollers
- Motors
- LCD
- Ultrasonic sensor
- Camera & image processing

2.4.3. Non-functional Requirements

- GPS localization
- Web server communication & wifi module
- Speed sensor

- **TivaC:**

Tiva C is an embedded kit programming that is used to create an embedded system that does a specific task depends on the behavior of the connected components to it or the kit itself. As we said before, it has a lot of features that makes it capable of doing multi-tasks and create a full-functional embedded system. So, we can't just dispense it. However, we could use another embedded kits. One of these alternative kits and is likely to be popular is the Arduino, which is a simple kit with enough features to do most of the TivaC functions, which means that we can replace the TivaC with the Arduino board, which makes you able to create a fully-functional embedded system in just some minutes unlike the TivaC and that's an advantage of it on the TivaC, but in the opposite, you will sacrifice other advantages of the TivaC on the Arduino.

- **Raspberry Pi (Board & Camera):**

The camera is used to capture the objects in front of the car. Then, using the Raspberry Pi board, a process is done to these objects in order to detect specific objects with its coordinates and send these data to the TivaC for, where the car will take actions depends on these data. So, for an image processing project with some machine learning, we need to use the Raspberry Pi board with camera. However, an alternative device can be used for this detection process, which is the mobile phone (Android). Android phones can capture the objects with their camera. Then these objects can get processed using an application on the device (instead of the Raspberry Pi board).

- **Wi-Fi & GPS module:**

In our project, the GPS module is used to get the location of a specific object in the road, which is the speed bump. Then, this location is sent to other cars with a warning notification to warn other drivers of a speed bump found ahead. But, this process requires other cars to be smart (have some devices, which qualifies these cars to communicate with our car to receiver these data). So, if we concerned only on our car and make our project (V2X) communication, which makes the car communicate with the surrounding objects without the (V2V) communication, which makes the car communicate with the surrounding cars, we can then dispense these two modules.

- **Bluetooth module:**

The Bluetooth in our project is used to control the normal car movement, but in real life, the car will be driven by the driver himself with no need of this Bluetooth.

- **Ultrasonic sensor:**

Beside the camera detection, ultrasonic sensors are used to detect the objects too, but any objects, not only specific objects like what the camera does, which means that these sensors are used to detect any surrounding objects (without the need to identify what these objects are). This helps to protect the car from hitting anything in its way like obstacles, humans, baskets, other cars, and so on… . Our car must have at least one ultrasonic sensor as it helps protect the car and it has high reliability compared to the camera detection.

- **Speed sensor:**

Speed sensor is used to measure the speed of the car, this can be used in order to warn the driver of the car speed so it doesn't exceed a specific speed. This is used to protect the car and the driver, but here, in our project, this speed sensor is specially used to protect the car from getting damaged by the speed bump. When the car sees the speed bump, the driver will be notified of a speed bump ahead. Approaching this speed bump, the ultrasonic sensor will sense and detect the distance of this speed bump from it and the car will gradually reduce its speed (using the PWM) to a specific speed (using the speed sensor), so the car can safely cross the speed bump without getting damaged, which means that the speed sensor is a must for our project.

- **LCD display:**

The LCD display is used to display the type of warning to the drive. For example, if the car approached a speed bump and detected it, then it should notify the driver of it writing a “Speed bump” word in the LCD display. This notification might not be necessary to the driver as the car will take actions if the driver didn’t notice the notification on the LCD display. However, we see that this is a must in our project as if a trouble, lag, or a fault happened to our action system, the driver should at least see the LCD display to take actions manually and reduce the speed of the car, stop the car, or move the car away.

3

Features

3. Features :

3.1. Speed Bump and traffic light detection

Most of the bumps in Egypt are not being constructed and maintained according to the public safety, which causes damage to vehicles, severe discomfort to the driver and can even cause a loss of direction control which can lead to fatalities.
Speed bumps are randomly placed without engineering studies in Egypt. Figure 1-1 demonstrates the speed fluctuations and the excessive decelerations and accelerations, before and after speed humps, over small distances.
Such instability in travel speed of course will increase the travel time, possible damages to vehicles, discomfort to passengers, and increase in fuel consumption and pollution as well as deterioration in pavement condition.

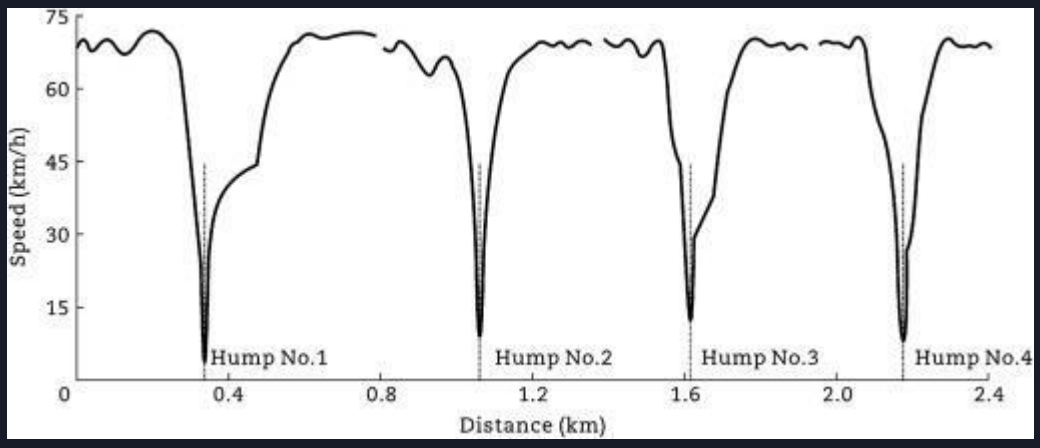


Figure 24: Speed vs Distance Chart

Road anomalies have always been a bad sign of road quality. Rough roads, speed bumps, and potholes endanger the safety and comfort of car drivers, so this project focuses on the problem of bumps in Egypt and provide a solution for it.

3.1.1. Technical Description of the solution

Figure 25: Technical Description of solution

3.1.1.1. Hump data set-

The dataset for fine-tuning the pre-trained model will be prepared using traffic light and speed hump/bump images taken from different views in our model

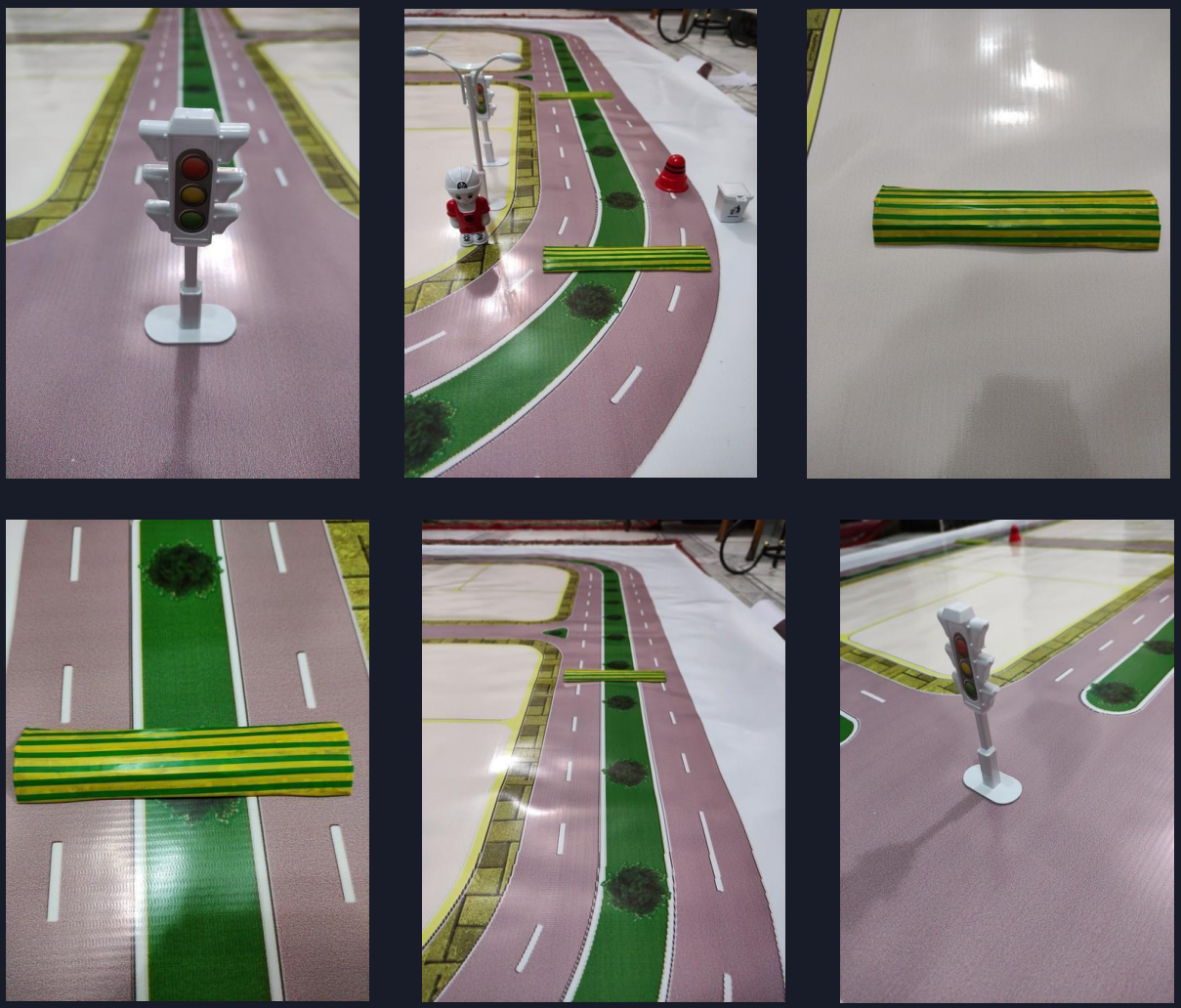


Figure 26: Hump Data Set

3.1.1.2. TensorFlow records

Tensorflow accepts inputs in a standard format called TFRecord file which is a simple record oriented binary format
Eighty percent of the input data is used for training and Twenty percent is used for testing
The split dataset of images and ground truth boxes are converted to train and test TFRecords

3.1.1.3. Parameter configuration

Parameters that can be configured to improve the system performance. As
we will try to reach the highest success percentage for the provided data

3.1.1.4. Training

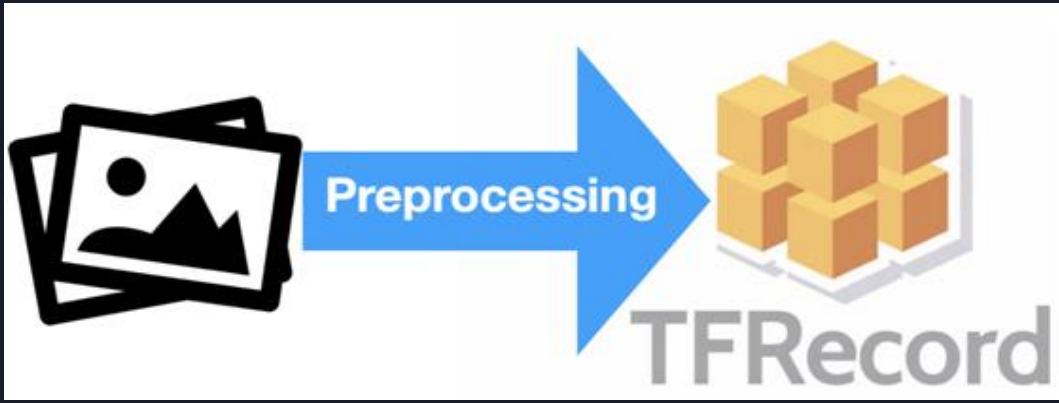
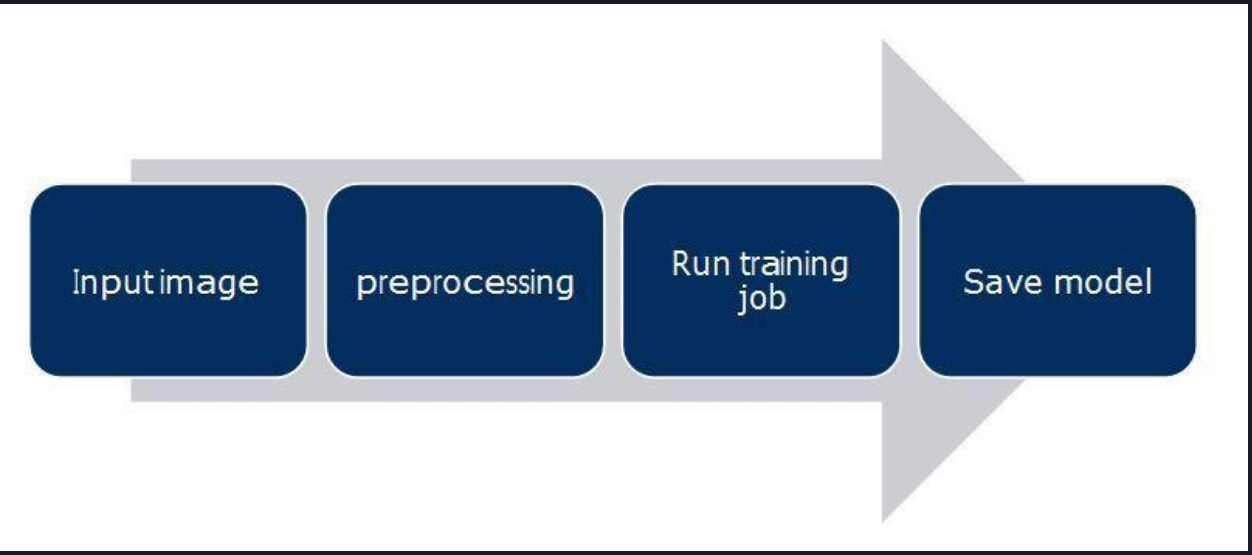


Figure 27: TensorFlow Records

Once the optimization parameters are set, the training file is executed.By default, the training job will continue to run until the user terminates it explicitly.The models will be saved at various checkpoint.



3.1.1.5. Distance Estimation

Information on distance of surrounding objects is useful for localization and navigation of the vehicle. In the driving experience for speed breaker detection,humans roughly estimate the distance towards it and controls the speed of vehicle by acceleration change and braking, we could use ultrasonic and speed sensor to accurately track the distance towards speed bump in real-time.

Figure 28: Ultrasonic Sensor in the car

3.1.1.6. Inference

The inference video will be first converted into frames.
These sets of frames are gaven to our model trained using transfer learning.
After the frames pass through the object detection pipeline,the bounding boxes will be drawn on the detected frames
These frames are finally merged to form the inferred video.

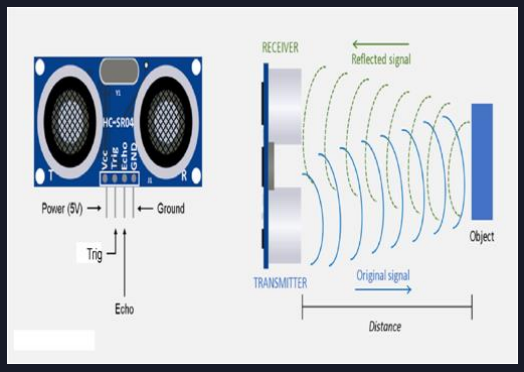


Figure 29: Ultrasonic Working Theory 1

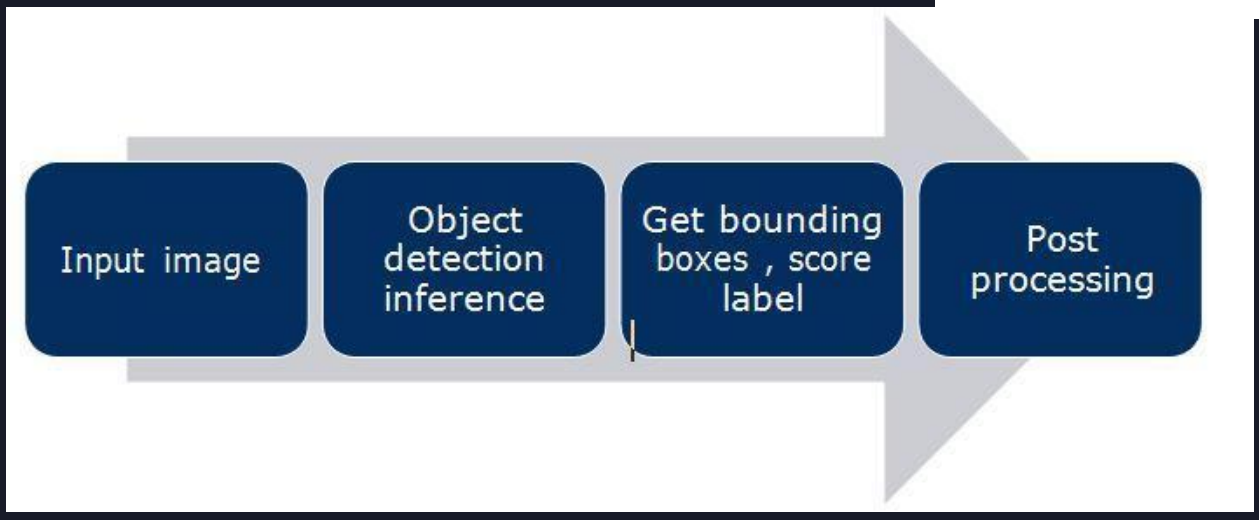


Figure 30: Ultrasonic Working Theory 2

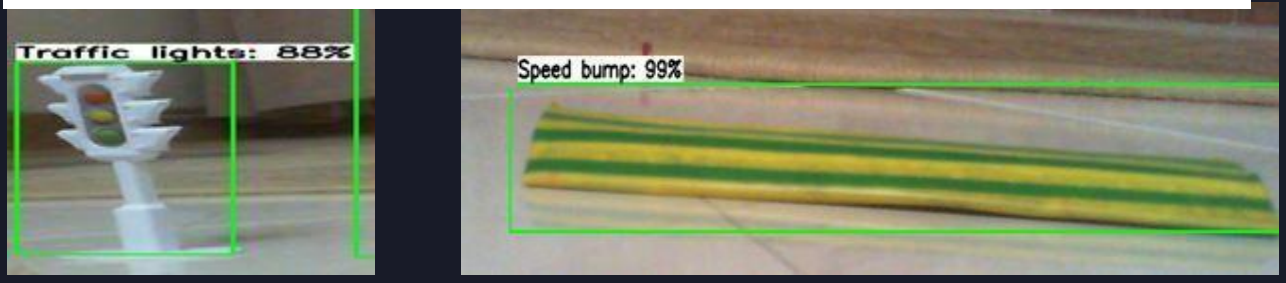


Figure 31: Detected Samples

3.2. Emergency Breaking:

3.2.1. Introduction:

According to WHO (World Health Organization), lives of approximately 1.27 million people are cut short every year as a result of a road traffic crash. Half of the 1.27 million people die in road traffic crashes are pedestrians, motorcyclists, and bicyclists.
Various advancements have been achieved in the field of active Safety systems to make the road safer. V2V communication, Adaptive Cruise Control, and Automatic Emergency Braking are some of the advances in the active safety systems. In the current V2V (vehicle to vehicle communication) system, vehicles communicate with each other over a wireless network. Whereas AEB (Autonomous Emergency Braking) system uses vehicle's on-board sensors such as radar, LIDAR, camera, infrared, etc. to detect and alert the driver for any potential collision The object information detected by vehicle and the information received through the V2V network is processed by the AEB system of the vehicle. If there is an imminent crash, the AEB system alerts the driver and applies brake automatically in critical condition
Moreover, rapid we can integrate these technologies with V2V-AEB system to improve safety, as When a vehicle gets the emergency information, it broadcasts the message on V2V network with higher priority and tries to apply brake using AEB system to stop safely. Moreover, it informs emergency services so that the driver can get a quick attention.
For a practical implementation of the V2V-AEB system, it is necessary to resolve certain issues. One of the key issues is a time delay in the system. As in real time V2V-AEB system, delays as shown in the following table:

Time Delay type	Definition
Sender delay	It occurs while detecting pedestrian through AEB system on the sender side and merging all the information of detected object in a message format.
Communication delay	It occurs during transmitting data from one object to another. This may occur due to packet collision or excess of information in the network.
Mechanical delay	It happens in the braking system. It represents the time difference between good and bad mechanical operation due to change in friction between road and tire.
Signal processing delay	Occurs, while processing and merging the information, received from other vehicles and its sensors, for making AEB decision.

Figure 32: Types of Delay

may be caused during the period of detection, communication, processing and while implementing the decision at the sender as well as at the receiver side. This delay will result in a late response by the AEB system which may lead to a collision. For the system to function correctly, the delay in the system should not exceed an absolute limit. If the delay is too long,

the receiving vehicle may not have enough time to respond to the received information. Hence the shared AEB information may become useless.
To make a system robust such that delay in the system does not reduce its effectiveness while making a decision, it is necessary to study the effect of delay thoroughly and to handle them properly.
without extrapolating trajectory of the pedestrian. However, when the delay is above certain time, the system fails to overcome that delay. So, it's necessary to keep the delay under a particular value. To reduce delay in V2V-AEB, various methods have been proposed and implemented. The proposed method helps to reduce communication delay, delay due to packet collision and signal processing delay by decreasing the number of messages in the V2V-AEB system. By preventing from sending messages related to the pedestrian who likely will not cause a collision (like the pedestrian who are walking on the sidewalk and who are standing off the road).
The other method developed, is grouping the pedestrians with similar features in one message, thus reducing the number of messages in the system and decreasing the communication and signal processing delay.

There has been a huge advancement in active safety features. AEB has been one of the vital parts of active safety systems development. AEB system is designed to alert the driver in emergency conditions and take the braking decisions if the driver does not make the desired decision to prevent a potential collision. A study performed by IIHS (Insurance Institute for Highway Safety) shows that AEB system can reduce rear-end collision around forty percent by 2025. NHTSA has announced automakers to make AEB system as a standard feature by 2022 claiming that it can prevent 28,000 crashes and 12,000 injuries.

3.2.2. Autonomous Emergency Break:

An emergency braking system is a representative active safety system that has functions to avoid crashes between the subject vehicle and frontal object, and to mitigate impact damage in emergency situations. As shown in Fig.1, this system is comprised of a radar sensor, camera sensor, AEB controller, braking actuator and a Human Machine Interface (HMI).
The radar sensor detects the frontal object and the camera sensor classifies the frontal object. The AEB controller recognizes the vehicle and pedestrian by using radar and camera sensors, and monitors the collision hazard between the subject vehicle and frontal object. Based on a situation analysis, the AEB controller generates a collision warning and braking command

depending on various criticality stages. Finally, the braking actuator generates a hydraulic braking force and the subject vehicle decelerates and reduces the vehicle speed

Fig. 1. Block diagram for an Autonomous Emergency Braking (AEB) system.

Pedestrian Autonomous Emergency Braking (PAEB) is an important part of the AEB system. The PAEB system calculates the probability of collision with a pedestrian by processing the

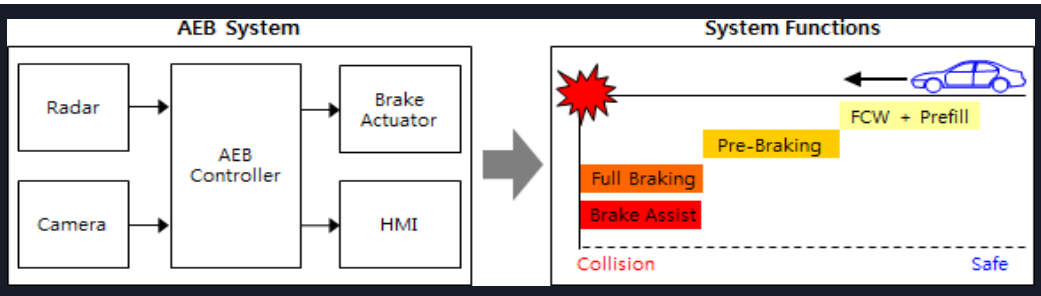


Figure 33: AEB System

position and relative speed of pedestrian with respect to vehicles current position. AEB system uses different types of sensors to detect the potential collision, such as radar, LIDAR, and camera [3]. The system processes the information provided by the onboard sensors of the vehicle and its current state (vehicle position, speed, direction, etc.), to recognize all potential collisions. To apply automatic brake some system uses full braking pressure, while other use elevated braking pressure proportional to the emergency level to prevent a possible collision. Furthermore, to improve the efficiency of AEB system some companies use automatic steering along with automatic braking to avoid collision [10]. For example, if there is a case when there is not enough room to avoid a collision by just applying the brake, the system can use automatic steering along with the brake to prevent the Collision

3.2.3. Control Algorithms:

Since AEB systems always work with a human driver, they can be useful and acceptable to the driver. The goal of this system is to allow the driver enough time to avoid the crash and yet avoid annoying the driver with satisfying performance of collision avoidance and mitigation. Fig. 2 shows a scheme of a proposed control algorithm. As shown in the figure, several indexes were used to determine an appropriate warning / braking intervention timing and calculate the braking command: time-to-collision (TTC), required deceleration, in-path probability (Position & Overlap). When indexes are satisfied with the specified condition based on a threshold value, the collision warning and braking intervention would be generated.

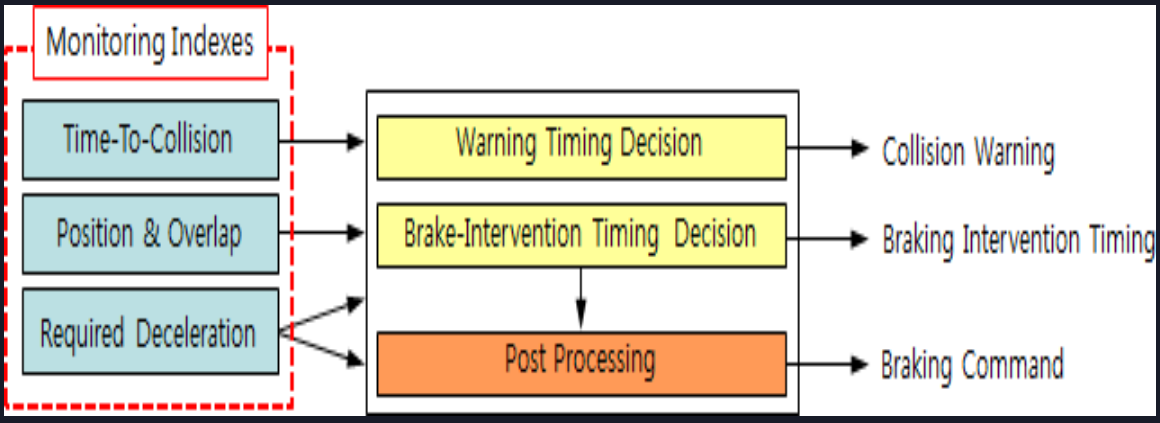


Figure 34: Emergency Brake Working Theory

3.2.4. V2V-AEB System:

To overcome the limitations of the V2V and the AEB system as mentioned above, we have integrated the complementary capabilities of the V2V and the AEB system by sharing the information detected by the vehicles onboard sensor over the V2V network, The other vehicles can use this information to make a safety decision if necessary. Thus, it compensates the limitations and disadvantages of both the V2V system and the AEB system.
For an AEB system to detect a pedestrian, it is necessary that the pedestrian is in its line of sight. These limitations of the AEB system might be nullified by the proposed V2V-AEB method, as vehicles can use the information sent by other vehicles. Another limitation of AEB system is that its sensors have short detection range, which provides little time for the system to react. The maximum detection range of AEB sensor is 80-100 meters. Beyond this, it is difficult for the vehicle to detect objects.

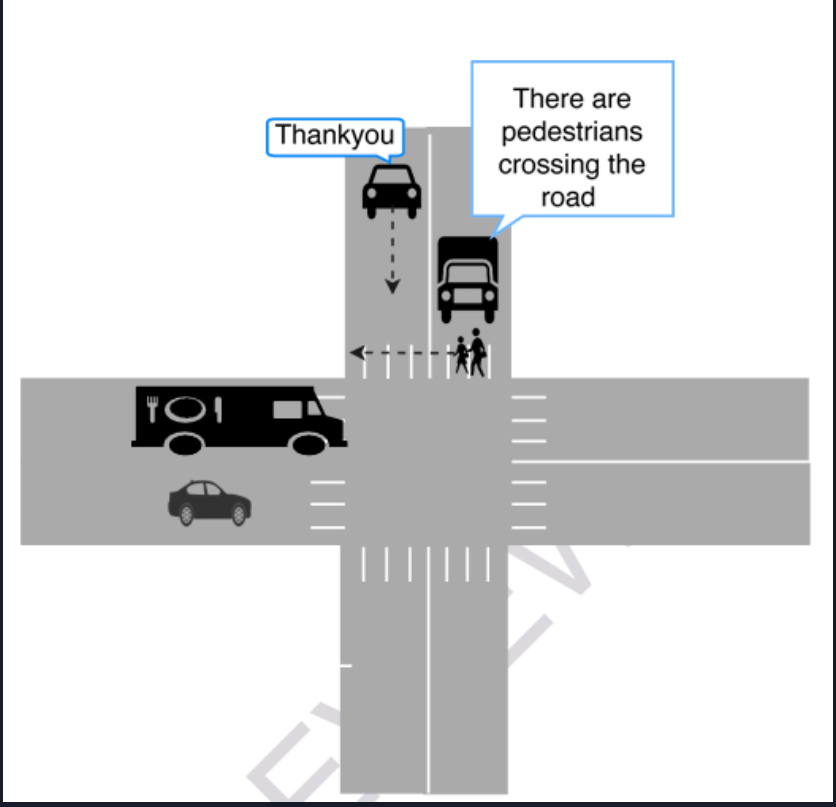


Figure 35: blind spot detection

This figure shows advantages of a V2V-AEB system. Suppose all the vehicles are equipped with the V2V-PAEB system, and a pedestrian is crossing the road. The pedestrian is not in the line of sight of a car, but the AEB system of the truck can detect a pedestrian, the truck broadcasts the pedestrian information on the V2V network. All vehicles receiving this information can make a safety decision. If the V2V-AEB system were not there, the car would not know that the pedestrians are crossing the road and there might be a collision.
We have seen the working of the V2V-AEB system and its advantages. Let us look at the basic architecture of the V2V-AEB enabled vehicle as shown in Fig. 1.2. The overview of an V2V-PAEB system is shown below in the Fig. 1.2. For a better understanding, we can divide the system in two-parts i.e. sender side and receiver side.

Sender side: The PAEB system processes the data from the camera sensor and the radar sensor. If there are any pedestrians in the scenario, the PAEB system senses the pedestrians and collects its information. Then the information is passed to the following blocks: here the data is converted into global coordinates from local coordinates of the respective vehicle, and it is given to the AEB system to make the desired decision to prevent a potential collision. Then the information is formulated in the message format along with the vehicle state. The vehicle broadcasts this message over the V2V network.
Receiver side: The message (V2V-PAEB) received by the vehicle is processed and merged, as different vehicles may send the information of the same pedestrian. Then the pedestrian

information is converted into local coordinate of the respective vehicle. Message merge block combines the information of the pedestrians from V2V with the Information of the pedestrians provided by its PAEB system. Then the pedestrian is tracked and time to collision of the pedestrian with respect to the vehicle is calculated to make the desired decision.

Figure 36: Lane capturing

3.3. Computer Vision for Lane Finding

Advanced computer vision techniques to identify lane lines from a video camera feed mounted on a car.

Result: Projection of our lane line onto the original video.

For developing this software, we need to do the following stages:

1. Camera calibration to remove lens distortion effects.
2. Image pre-processing to detect lane lines.
3. Perspective transform on road to aid in detection.
4. Lane Line Detection on transformed road.
5. Vehicle position and lane radius of curvature calculation.

3.3.1. Camera Calibration

The output from the camera is a video, which in essence is a time-series of images. Due to the nature of photographic lenses, images captured using pinhole camera models are prone to radial distortions which result in inconsistent magnification depending on the object's distance from the optical axis.

The following is an example image from OpenCV showcasing the two main types of radial distortions:

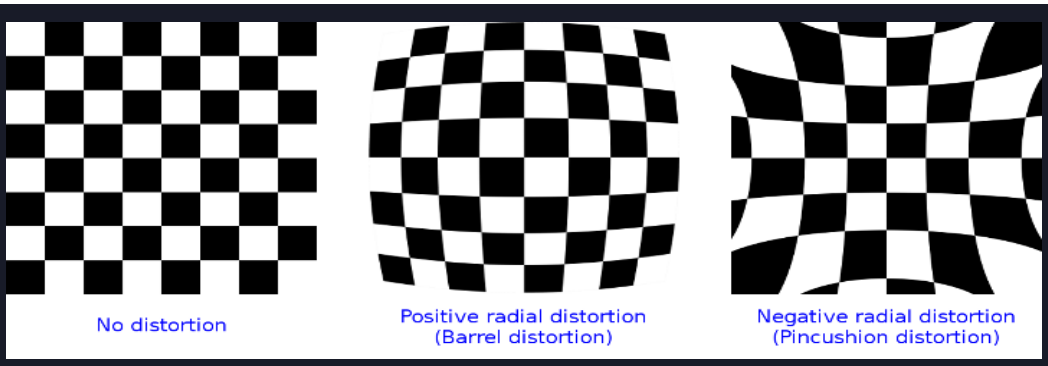


Figure 37: Radial Destortion

In order to correctly detect the lane lines in the image, we first need to correct for radial distortion.
Computer vision researchers have come up with a way to correct this radial distortion.
The camera to be calibrated is used to capture images of checkerboard patterns, where all the white and black boxes in the pattern are of the same size. If the camera suffers from distortions, the captured images will incorrectly show the measurements of the checkerboard.
To correct the effects of distortion, the corners of the checkerboard are identified and deviations from the expected checkerboard measurements are used to calculate the distortion coefficients.
These coefficients are then used to remove radial distortion from any image captured using that camera.

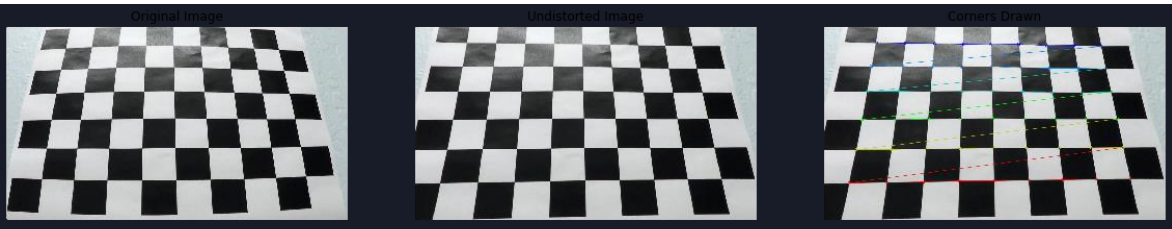


Figure 38: Distortion Fixing

In the diagram above, the leftmost image shows the original distorted image, the rightmost image shows the corners drawn on top of the distorted image, and the middle image shows the resultant undistorted image after camera calibration.
The OpenCV functions (findChessboardCorners) and (calibrateCamera) were used to achieve the above camera calibration process.

3.3.2. Image Pre-processing

With the undistorted images at hand, we now return to the main objective of detecting the lane lines on the road.

One way to separate and detect objects in an image is to use colour transforms and gradients to generate a filtered-down thresholded binary image.
For colour transforms, by experiment with three colour spaces in order to find out which one is best at filtering the pixels representing the lane line on the road. Three colour spaces were tested:

HSL: represents colour as three channels — hue, saturation, and lightness.

LAB: represents colour as three channels — lightness, component for green-red, and component b for blue-yellow.

LUV: a transformation of the XYZ colorspace that attempts *perceptual uniformity.*

After some experimentation, the conclusion was that the b channel of the LAB colorspace and the L channel of the LUV colour space are the best combination for detecting the lane lines on the road.
The Sobel gradient filter was also considered. An image gradient measures the directional intensity of the colour change. Sobel is a type of gradient filter that uses Gaussian smoothing and differentiation operations to reduce the effects of noise.

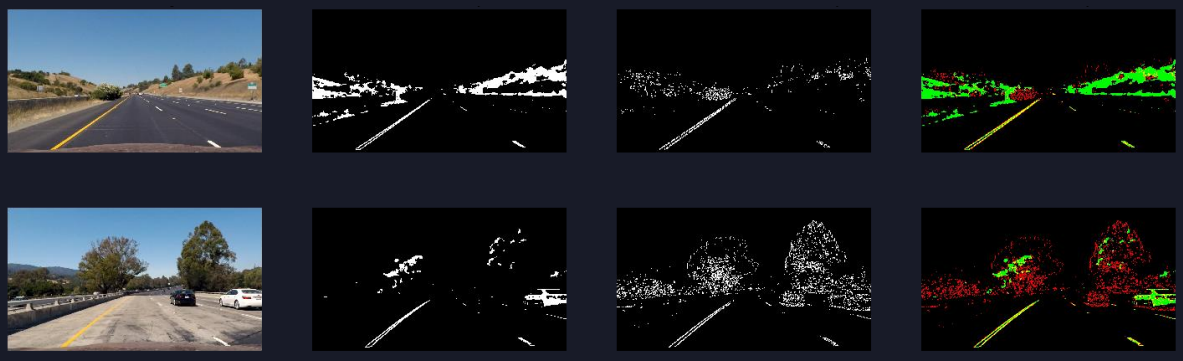


Figure 39: Fixing destortion using two filters

Figure 4. Original undistorted images in the first column, the b/L channel thresholding in the second column, the Sobel gradient filter in the third column, and the two filters combined in the last column.

3.3.3. Perspective Transform

We can now distinguish lane lines in the image, but the task of figuring out the exact angle/direction of the lane is difficult using the default camera view. In the default camera perspective, objects further away from the camera appear smaller and the lane lines appear to converge the further they are from the car, which is not a true representation of the real world.
One way to fix this perspective distortion is to transform the perspective of the image such that we are looking at the image from above, also known as birds-eye view.

OpenCV provides: functions)getPerspectiveTransform(and)warpPerspective(, which can be used to apply a perspective transformation to a segment in the image. Firstly, we pick the area in the image we would like to apply the transformation to.

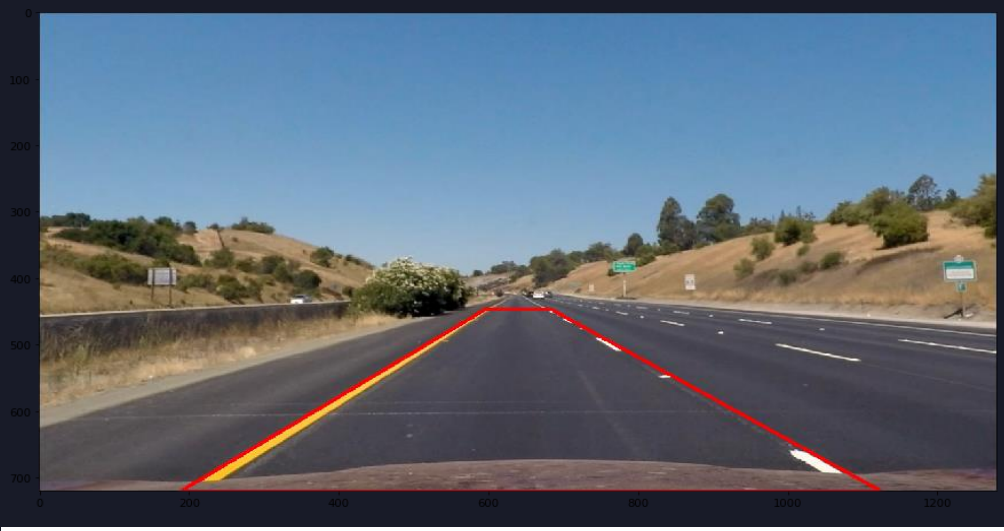


Figure 40: Lane Source Points detection

Then, we choose the points representing the destination space we would like to transform the segment to, in our case any rectangle would suffice. The function will then return a 3x3 transformation matrix which can be used to warp any segment into our chosen perspective using the warpPerspective function.
The following image shows lanes lines from two different segments of the road with the perspective transformation successfully applied:

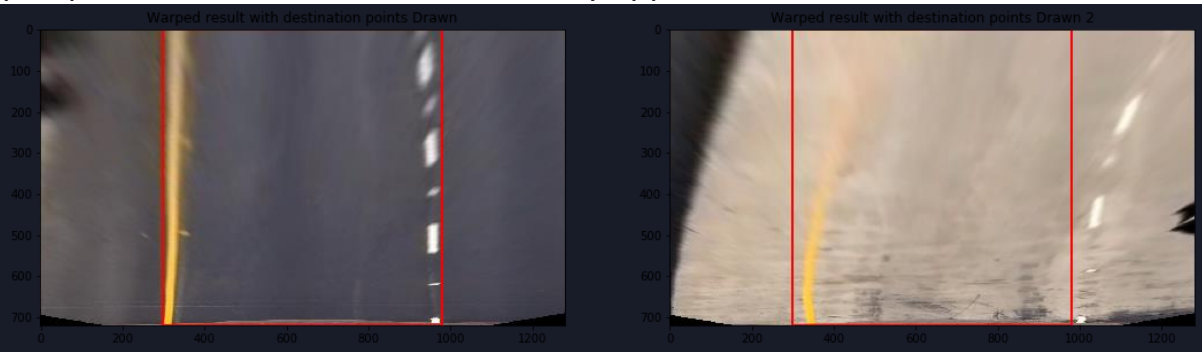


Figure 41: Road formation for detection

Notice how it is now much easier to determine the curvature of the lane line!

3.3.4. Lane Line Detection

We are now finally ready to fully detect the lane lines! As a start, we apply the binary thresholding discussed in the Image Pre-processing section to the perspective transformed

lane line segment. We now have an image where the white pixels represent parts of the lane line we are trying to detect.
Next, we need to find a good starting point to look for pixels belonging to the left lane line and pixels belonging to the right lane line. One approach is to generate a histogram of the lane line pixels in the image. The histogram should have two peaks each representing one of the lane lines, where the left peak is for the left lane line and the right peak is for the right lane line.

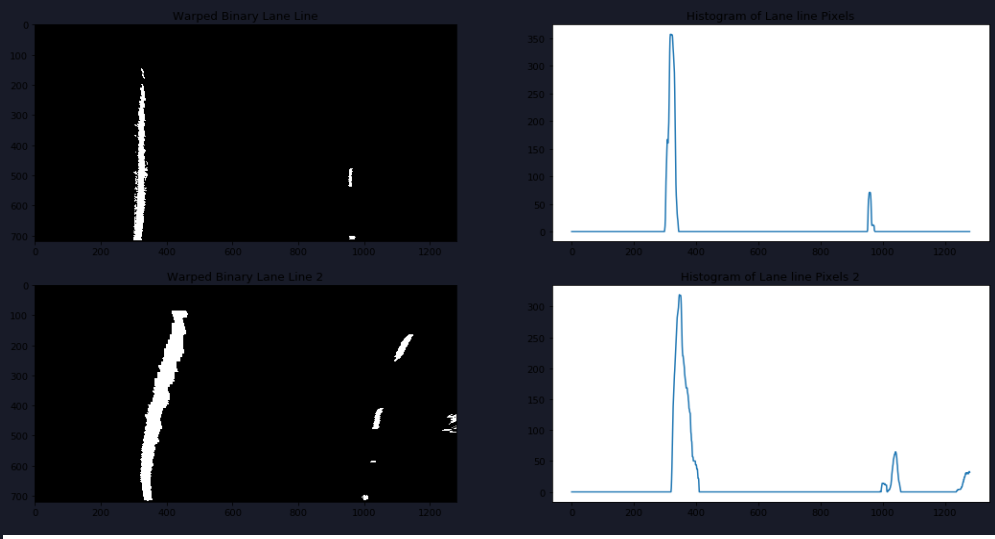


Figure 42: Image Histogram

The image below shows two example histograms generated from two binary images:
The locations of the two peaks are then used as a starting point to search for pixels belonging to each lane line. We employ a sliding window search technique that starts from the bottom and iteratively scans all the way to the top of the image, adding detected pixels to a list. If a sufficient number of pixels is detected in a window, the next window will be centred around their mean position, that way we are following the path of the pixels throughout the image.

After we've detected the pixels belonging to each lane line, we then fit a polynomial through the points, generating a smooth line which acts as our best approximation of where the lane line is.

The image below shows the sliding window technique in action, with the polynomial fit through the detected lane pixels (red for left lane pixels and blue for right lane pixels):

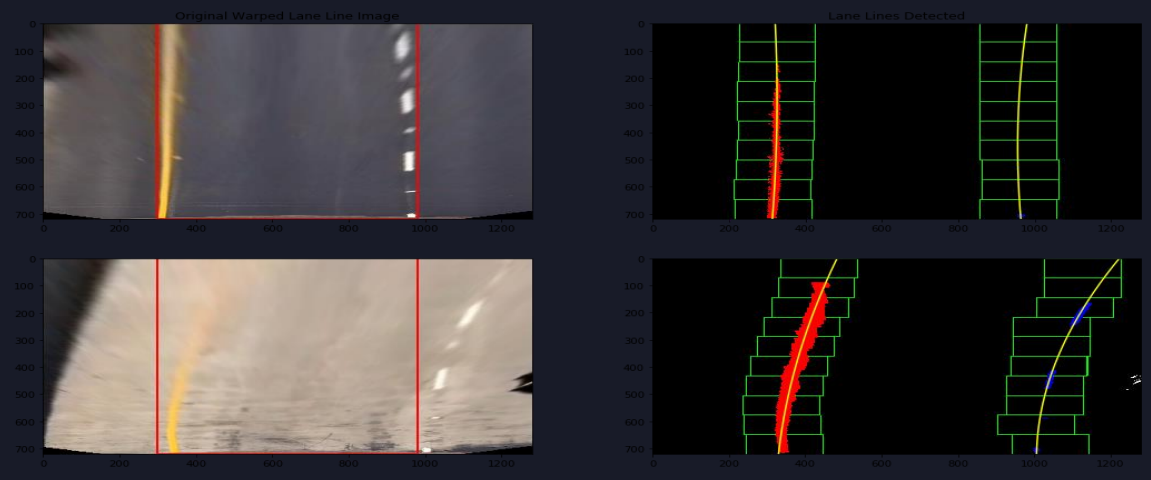


Figure 43: Wraped detected Images

Below is another view of the sliding window search technique, with the search area highlighted and filled:

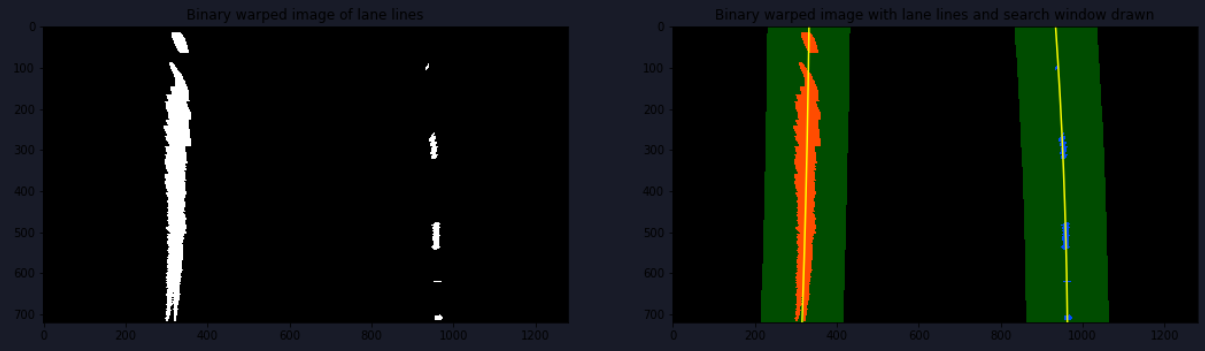


Figure 44: Sliding Window Technique

3.3.5. Vehicle/Lane Position

Finally, using the location of the two detected lane lines and the assumption that the camera is located at the centre of the image, we then calculate the position of the car relative to the lane. Scale measurements to convert from pixels to meters have been calculated using the resolution of the image.
Furthermore, using the scale measurements, we can also calculate the curvature of the lane by fitting a new polynomial to the world space, then calculating the radii of curvature. The radius

of curvature of the lane is then just the average of the two radii. Below image shows curve radius and centre offset for the two lane lines (detection not visible in image):

4

Computer Vision

4. Raspberry Pi

The Raspberry Pi is a low cost, **credit-card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to 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 playing high-definition video, to making spreadsheets, word-processing, and playing games.
Runs on several operating systems (Raspbian -Windows 10 - IoT Core &and more).
What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

Figure 45: RAspberry Pi

4.1. Setting up Raspberry Pi

4.1.1. Connect the keyboard, mouse, and monitor cables

Raspberry Pi 4 has plenty of connections, making it easy to set up. We'll interact with the Raspberry Pi using a keyboard and mouse.

4.1.2. Set up Raspberry Pi OS: Raspbian

Now we've got all the pieces together, it's time to install an operating system on your Raspberry Pi, so we can start using it. Raspberry Pi uses a custom operating system called Raspbian (based upon a variant of Linux called 'Debian').
We're going to use a set of software called NOOBS (New Out Of Box Software) to install Raspbian OS on our microSD card and start our Raspberry Pi

Raspbian is the official OS for Raspberry Pi, and the easiest way to set up Raspbian on your Raspberry Pi is to use NOOBS (New Out Of Box Software).

4.1.3. Download SD Card Formatter tool

Start by downloading SD Card Formatter Tool from the SD Card Association website,attach the micro SD card to our PC ,Launch the program , choose quick format and format the card.

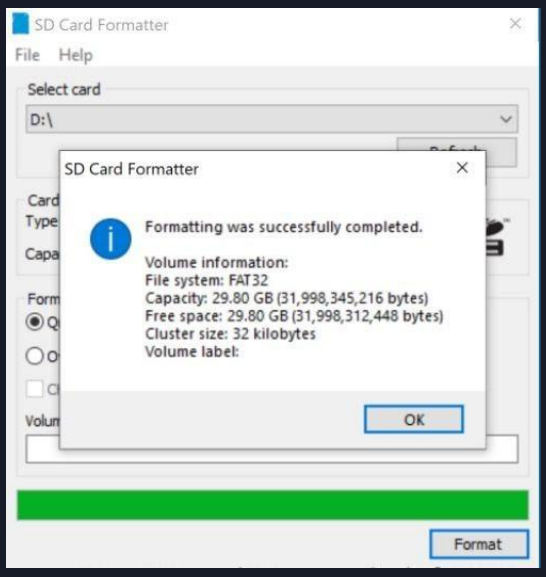


Figure 46: SD card formatter tool

4.1.4. Download NOOBS

Download NOOBS files and then copy all the files from the NOOBS folder to SD card.

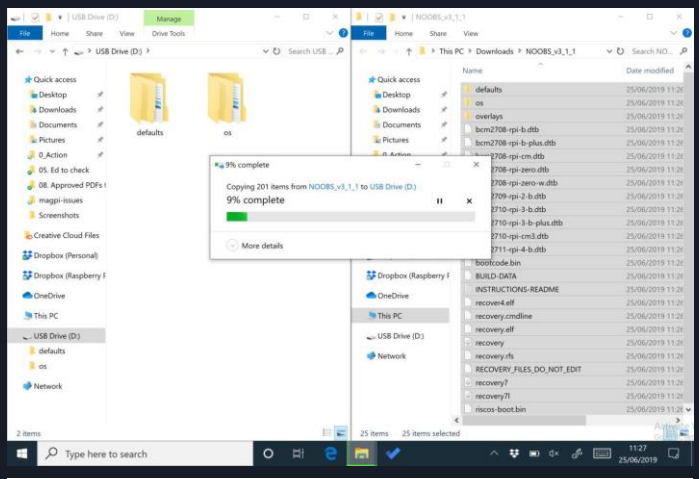


Figure 47: NOOBS

4.1.5. Insert the microSD card to Raspberry Pi 4

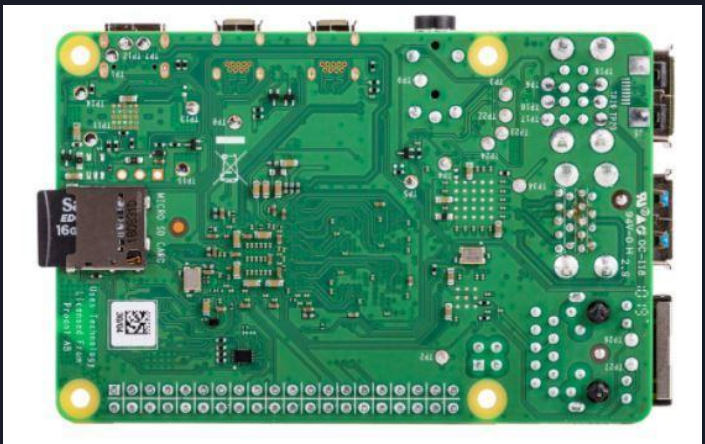


Figure 48: SD card insertion in the Raspberry pi

4.1.6. Power up

Plug in Raspberry Pi power supply and, after a few seconds, the screen should come on. When the NOOBS installer appears, we'll see a choice of operating systems. We're going to install Raspbian, the first and most popular one.

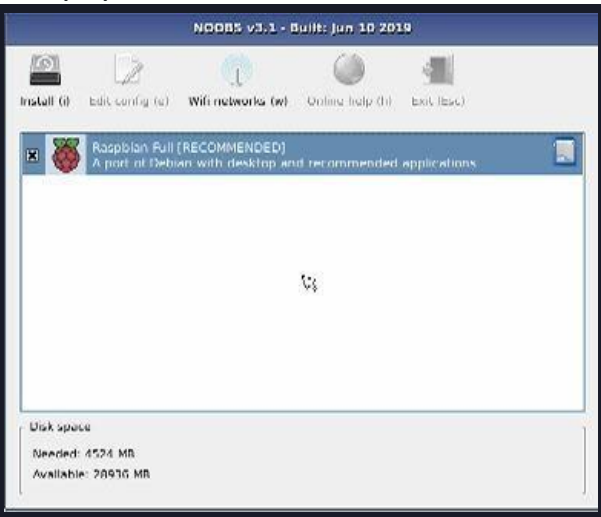


Figure 49: Powering on the raspberry pi

4.1.7. Welcome to Raspberry Pi

4.1.8. How to Set Up and Run TensorFlow Lite Object Detection Models on the Raspberry Pi

4.1.8.1. Update the Raspberry Pi

First, the Raspberry Pi needs to be fully updated. Open a terminal and issue:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Figure 50: Updating the Raspberry pi

Enable the camera

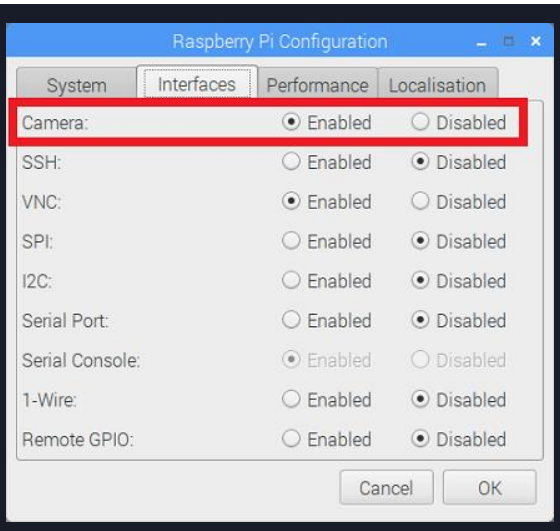


Figure 51: configuring the raspberry pi

4.1.8.2. Download this repository and create virtual environment

The repository contains the scripts we'll use to run TensorFlow Lite, as well as a shell script that will make installing everything easier.

```
git clone https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
```

This downloads everything into a folder called TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi. That's a little long to work with, so rename the folder to "tflite1" and then cd into it:

```
mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi tflite1
cd tflite1
```

4.1.8.3. Install TensorFlow Lite dependencies and OpenCV

Next, we'll install TensorFlow, OpenCV, and all the dependencies needed for both packages. OpenCV is not needed to run TensorFlow Lite, but the object detection scripts in this repository use it to grab images and draw detection results on them.

```
bash get_pi_requirements.sh
```

4.1.8.4. Set up TensorFlow Lite detection model

Next, we'll set up the detection model that will be used with TensorFlow Lite.
A detection model has two files associated with it: a detect.tflite file (which is the model itself) and a labelmap.txt file (which provides a labelmap for the model). My preferred way to organize the model files is to create a folder (such as "BirdSquirrelRaccoon_TFLite_model") and keep both the detect.tflite and labelmap.txt in that folder. This is also how Google's downloadable sample TFLite model is organized.

4.1.8.5. Using Google's sample TFLite model

Google provides a sample quantized SSDLite-MobileNet-v2 object detection model which is trained off the MSCOCO dataset and converted to run on TensorFlow Lite. It can detect and identify 80 different common objects, such as people, cars, cups, etc.

4.1.8.6. Download the sample model

```
wget https://storage.googleapis.com/download.tensorflow.org/models/tflite/coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip
```

```
unzip coco_ssd_mobilenet_v1_1.0_quant_2018_06_29.zip -d Sample_TFLite_model
```

4.1.8.7. The Model :

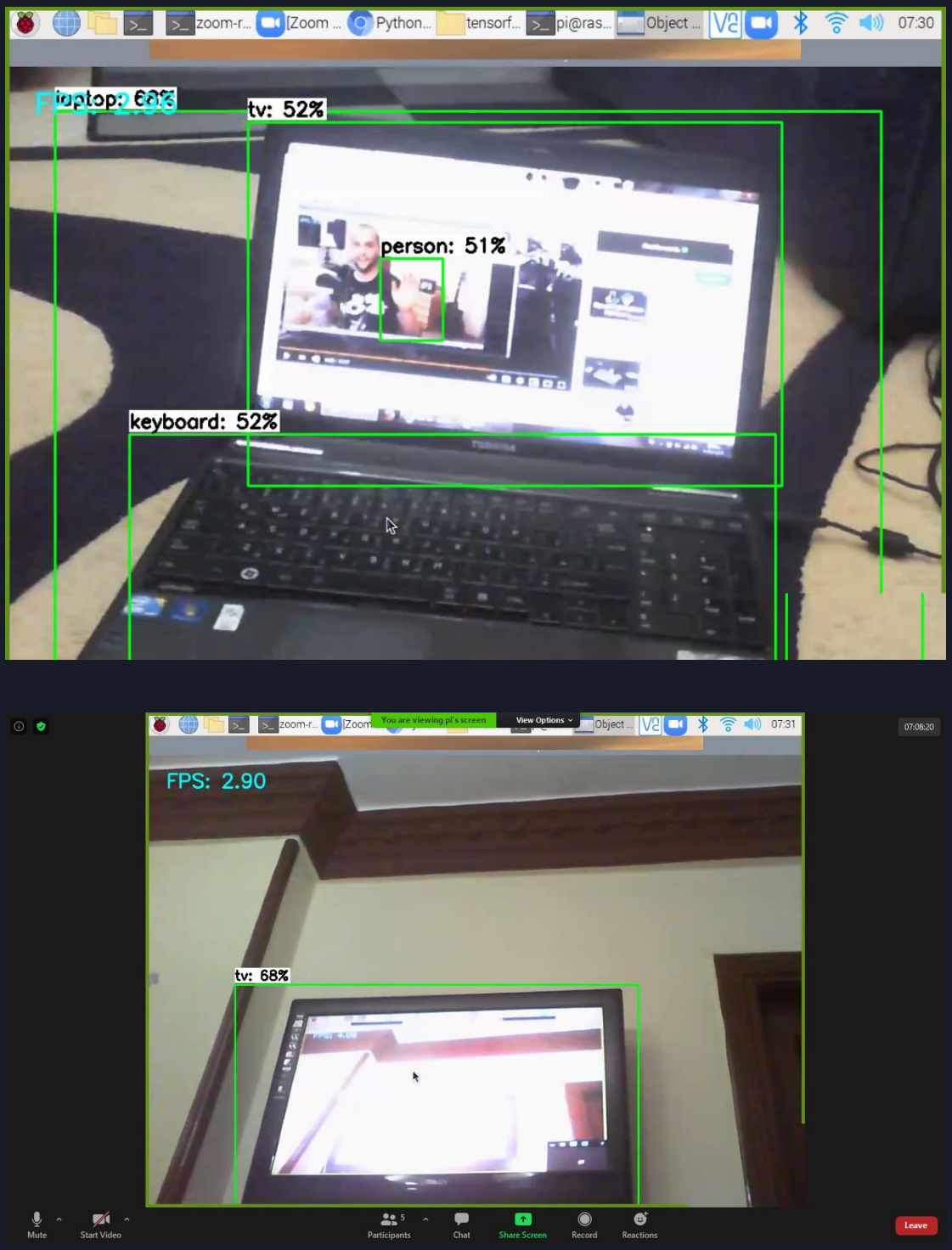


Figure 52: Pre-created model testing on the raspberry

4.2. training our model

In general, there are 3 steps to training a machine learning model:
When you hear the words machine learning, you probably think of face recognition, robotics or self-driving cars. But it's so much more than that. You don't have to be inventing the next big thing to leverage the power of machine learning in your business. In fact, you should be considering all the ways machine learning could work for you today.
Machine learning is not a way to solve the problems you're already familiar with. It's a way to solve new problems, business issues and tasks with data-driven predictions. To understand how you can apply machine learning, you need to first understand how it works. Let's start by training a machine learning model.

Step 1: Begin with existing data

Machine learning requires us to have existing data—not the data our application will use when we run it, but data to learn from. You need a lot of real data, in fact, the more the better. The more examples you provide, the better the computer should be able to learn. So just collect every scrap of data you have and dump it and voila! Right?
Wrong. In order to train the computer to understand what we want and what we don't want, you need to prepare, clean and label your data. Get rid of garbage entries, missing pieces of information, anything that's ambiguous or confusing. Filter your dataset down to only the information you're interested in right now. Without high quality data, machine learning does not work. So take your time and pay attention to detail.

Step 2: Analyze data to identify patterns

Unlike conventional software development where humans are responsible for interpreting large data sets, with machine learning, you apply a machine learning algorithm to the data. But don't think you're off the hook. Choosing the right algorithm, applying it, configuring it and testing it is where the human element comes back in.
There are several platforms to choose from both commercial and open source. Explore solutions from Microsoft, Google, Amazon, IBM or open source frameworks like TensorFlow, Torch and Caffe. They each have their own strengths and downsides, and each will interpret the same dataset a different way. Some are faster to train. Some are more configurable. Some allow for more visibility into the decision process. In order to make the right choice, you need to experiment with a few algorithms and test until you find the one that gives you the results most aligned to what you're trying to achieve with your data.
When it's all said and done, and you've successfully applied a machine learning algorithm to analyze your data and learn from it, you have a trained model.

Step 3: Make predictions

There is so much you can do with your newly trained model. You could import it into a software application you're building, deploy it into a web back end or upload and host it into a cloud service. Your trained model is now ready to take in new data and feed you predictions, aka results.

These results can look different depending on what kind of algorithm you go with. If you need to know what something is, go with a classification algorithm, which comes in two types. Binary classification categorizes data between two categories. Multi-class classification sorts data between—you guessed it—multiple categories.

When the result you're looking for is an actual number, you'll want to use a regression algorithm. Regression takes a lot of different data with different weights of importance and analyzes it with historical data to objectively provide an end result.

Both regression and classification are supervised types of algorithms, meaning you need to provide intentional data and direction for the computer to learn. There is also unsupervised algorithms which don't require labeled data or any guidance on the kind of result you're looking for.

One form of unsupervised algorithms is clustering. You use clustering when you want to understand the structure of your data. You provide a set of data and let the algorithm identify the categories within that set. On the other hand, anomaly is an unsupervised algorithm you can use when your data looks normal and uniform, and you want the algorithm to pull anything out of the ordinary that doesn't fit with the rest of the data.

Although supervised algorithms are more common, it's good to play around with each algorithm type and use case to better understand probability and practice splitting and training data in different ways. The more you toy with your data, the better your understanding of what machine learning can accomplish will become.

Ultimately, machine learning helps you find new ways to make life easier for your customers and easier for yourself. Self-driving cars not necessary.

4.3. Used Libraries

There're a lot of libraries to train machine learning models: Most used are:

- **NumPy**

NumPy stands for Numerical Python. It is one of the most basic (yet advance) Python libraries available for scientific computing and can be used as a multi-dimensional container for data. One can perform Linear Algebra computations which are necessary for Machine learning Algorithms like Linear Regression, Logistic Regression, Naïve Bayes and so on. It is mostly written in C language (low-level language), due to which it is faster.

Pros

- NumPy uses less memory.
- Faster as compared to lists in Python.
- Mathematical operations can be performed using Numpy unlike Lists.

Cons

- NumPy arrays are homogeneous contiguous blocks of memory i.e., it will hold only one kind of data.

- **Pandas**

Pandas is one of the most important statistical library mainly used in the field of Statistics, Finance, Economics and Data Analysis. It is similar to Excel. It is known for processing large datasets.

Pros

- Pandas creates fast and effective Data frame objects with pre-defined and customized indexing.
- Can be used to manipulate large datasets. It deals with the missing values present in the dataset as well.
- Provides in-built features for creating Excel charts and performing complex data analysis task like data wrangling, data transformation and so on.

Cons

- Pandas does not persist data.
- Can only handle results that fit in memory, and is easy to fill.

- **Matplotlib**

Matplotlib is one of the most popular libraries for data visualization. It provides support for a wide variety of graphs like histogram, bar charts, scatter plot, pie charts and so on. It is basically a two-dimensional graphical library which produces very concise and clear graphs that are important for exploratory analysis. Nowadays I am getting more inclined to **Plotly** also :)

Pros

- Easy to plot graphs using matplotlib by providing functions for choosing suitable line styles, different font styles, changing and formatting axes.

- Easy to understand these graphs.
- Contains the Pyplot module that provides a basic interface, similar to MATLAB interface.
- Provides Object-Oriented API module that will help in integration of graphs into applications and tools.
- Matplotlib produces research quality graphs as it uses vectors instead of pixels.

Cons

- Graphs in Matplotlib are not interactive.
- Limited variety of visualization.

- **Scikit-learn**

One of the most effective library for machine learning, data modelling and model evaluation. It is built on the top of SciPy. It contains lot of functions for the purpose of model creation. Consisting of Supervised and Unsupervised Machine Learning algorithms.

Pros

- Provides a set of standard datasets, to help people to get started machine learning. E.g: Iris dataset.
- It has in-built functions to carry out both supervised and unsupervised learning which includes Clustering, Classification, Regression, Data Mining, Anomaly detection and so on.
- It consists functions for Feature Extraction and Feature Selection, which helps in identifying significant attributes or variables in data.
- It also has functions for Cross-Validation to estimate the performance of the model.
- It integrates really well with NumPy.

Cons

- Sometimes, it becomes really slow, especially during the training of models.
- Less flexible.
- Limited model tuning capabilities for few Machine Learning Model

- **Tensorflow**

A Deep Learning maintained by Google. It is a library that is used for building string and precise Neural Networks (algorithms are inspired by the structure of the brain). To represent Tensorflow programs tensor data structure is used. It supports programming languages like

C++, Python and R. It can used in Natural Language Processing (NLP), forecasting, text summarization, image/video analytics and handwriting recognition. It is known for faster deployment of algorithms while retaining the same APIs.

Pros

- Tensorflow allows to train multiple neural networks which help to accommodate large datasets.
- It provides functions and methods that provide basic statistical analysis.
- It also provides layered components that perform layer operations on weights and bias.
- Tensorflow also comes with a visualization tool called tensorboard.
- Pre-trained models availability
- Multi-GPU deployment support

Cons

- Can get really slow if you don't know what is happening under the hood.
- Due to its structure, debugging can get difficult.
- Steep learning curve (although it has become better since when I started)

- **Keras**

Keras provides full support for creating, analyzing, evaluating and improving neural networks. It is built on top of Theano or Tensorflow libraries which provide additional features to complex and large-scale deep learning models.

Pros

- Keras provide support for building all types of neural networks.
- Lightweight and easy to use.
- It is really straightforward when it comes to building a deep learning model by stacking multiple layers – Keras in a nutshell.
- It has several pre-processed datasets and trained models like MNIST.
- It is easily extensible and provide support to add new modules.
- François Chollet is the primary author and maintainer (great guy, follow him on twitter)

Cons

- Errors are difficult to debug.
- It is difficult to customize your layer because Keras already have pre-configured layers.

- **PyTorch**

PyTorch is an open-source Python-based scientific computing library, mainly used to implement deep learning techniques and neural networks on large datasets. It competes with Tensorflow. This library is developed by Facebook's AI Research lab (FAIR).

Pros

- It provides easy to use APIs.
- Efficient due to underlying scripting language LuaJIT and C/Cuda implementation.
- Can creates dynamic computation graphs.
- Good documentation and community support.

Cons

- It lacks visualization such as tensorboard in Tensorflow. Therefore, third-party is needed.
- API server needed for production.

- **XGBoost**

XGBoost stands for eXtreme Gradient Boosting. It was written in C++. This library was named after XGBoost algorithm. It is an application of gradient boosted decision trees created for speed and performance.

Pros:

- XGBoost has faster execution speed.
- It provides better model performance.
- The core XGBoost algorithm is parallelizable and it can use the power of multi-core computers.
- It can process very large datasets and it can work across a network of datasets.
- It also provides internal parameters to perform cross-validation for performing regularization, cross-validation, handling missing values and so on.

Cons:

- It is computationally expensive.
- It is less interpretable.

- **OpenCV**

OpenCV (Open Source Computer Vision) is a library used for computer vision. It supports Python, C++ and Java. In OpenCV, all the images are converted to NumPy arrays as this process will make it easier to integrate with other libraries that uses NumPy.

Pros

- OpenCV is written in C++, making it fast.
- Portable library

Cons

- OpenCV lacks in memory management.

- **NLTK**

NLTK (Natural Language Toolkit) is a leading library for building python programs to work with human language data and it also provides easy to use interfaces.

Functions performed using this library are classification, tokenization, parsing and so on. Some of its applications are text processing, recommendation system and sentiment analysis.

There many more libraries which are doing great work if you are looking to work with text data like **SpaCy** & **Gensim**.

Pros

- It supports the maximum number of languages as compared to other libraries.

Cons

- Slow speed.
- Can get difficult to use.
- It directly splits the sentences without analyzing the semantic structure of the sentence.

4.4. Conclusion

For most of the standard and out-of-the-box stuff Keras stands way ahead. If you are looking to go deeper and build your own layer go for PyTorch or Tensorflow. For other Machine Learning specific tasks like Text analytics there are great libraries like NLTK, or SpaCy.

So make the wise decision based on the current requirement when using the Machine Learning libraries for personal projects or for your Company. I will conclude by quoting Occam's razor principle

So, we used TensorFlow lite to train our model.

What is TensorFlow? Its Applications ?

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Its applications:

1. Voice/Sound Recognition

2. Text-Based Applications

3. Image Recognition

4. Time Series.

5. Video detection.

4.5. Training Our Model's Steps:

4.5.1. Install Anaconda, CUDA, and cuDNN

Anaconda is a software toolkit that creates virtual Python environments so we installed and used Python libraries without worrying about creating version conflicts with existing installations. Anaconda works well on Windows, and enables you to use many Python libraries that normally would only work on a Linux system. It provides a simple method for installing TensorFlow It also automatically installs the CUDA and cuDNN versions you need for using TensorFlow on a GPU. Anaconda will automatically install the correct version of CUDA and cuDNN for the version of TensorFlow you are using, so you shouldn't have to worry about this.

4.5.2. Set up TensorFlow Directory and Anaconda Virtual Environment

The TensorFlow Object Detection API requires using the specific directory structure. It also requires several additional Python packages, specific additions to the PATH and PYTHONPATH variables, and a few extra setup commands to get everything set up to run or train an object detection model.

- **Download TensorFlow Object Detection API repository**

By Creating a folder directly in C: and name it “tensorflow1”. This working directory will contain the full TensorFlow object detection framework, as well as your training images, training data, trained classifier, configuration files, and everything else needed for the object detection classifier.
Then, downloading the full TensorFlow object detection repository located at https://github.com/tensorflow/models by clicking the “Clone or Download” button and downloading the zip file.
Opening the downloaded zip file and extracting the “models-master” folder directly into the C:\tensorflow1 directory you just created. Rename “models-master” to just “models”.

- **Download the Faster-RCNN-Inception-V2-COCO model from TensorFlow's model**

TensorFlow provides several object detection models (pre-trained classifiers with specific neural network architectures) in its model zoo. Some models (such as the SSD-MobileNet model) have an architecture that allows for faster detection but with less accuracy, while some models (such as the Faster-RCNN model) give slower detection but with more accuracy. we initially started with the SSD-MobileNet-V1 model, but it didn’t do a very good job. we re-trained our detector on the Faster-RCNN-Inception-V2 model, and the detection worked considerably better, but with a noticeably slower speed.

Figure 53: Faster RCNN vs MobileNet

We can choose which model to train your objection detection classifier on. For a device with low computational power (such as a smart phone or Raspberry Pi), We should use the SDD-

MobileNet model. If we will be running our detector on a decently powered laptop or desktop PC, so we use one of the RCNN models.
So, we used the SDD-MobileNet model.

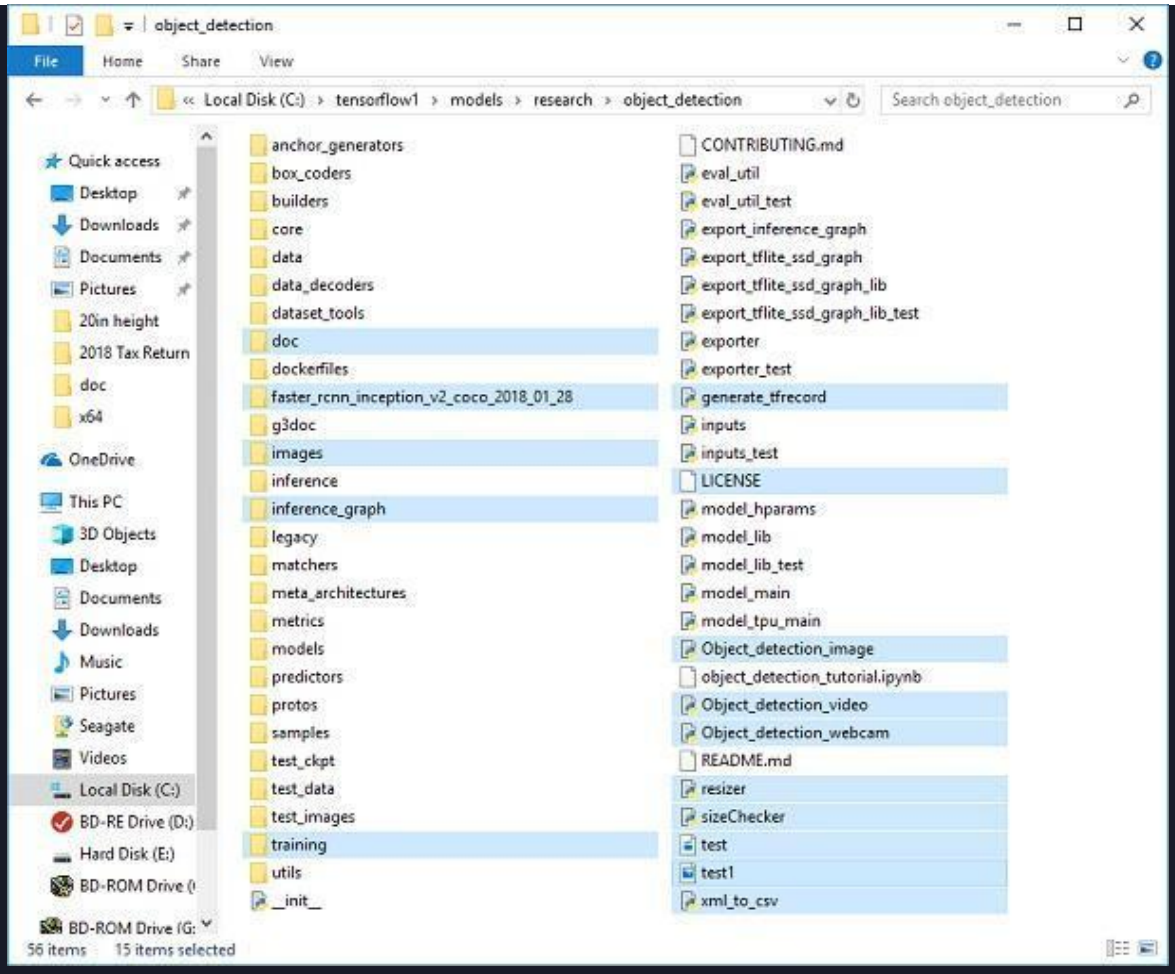


Figure 54: TensorFlow Github Repository

- **Set up new Anaconda virtual environment**

Next, we'll work on setting up a virtual environment in Anaconda for tensorflow-gpu.

In the command terminal that pops up, createating a new virtual environment called "tensorflow1" by issuing the following command:

C:\> conda create -n tensorflow1 pip python=3.5
Then, activating the environment and update pip by issuing:

C:\> activate tensorflow1

(tensorflow1) C:\>python -m pip install --upgrade pip
Installing tensorflow-gpu in this environment by issuing:

(tensorflow1) C:\> pip install --ignore-installed --upgrade tensorflow-gpu

Since we're using Anaconda, installing tensorflow-gpu will also automatically download and install the correct versions of CUDA and cuDNN.

(Note: We can also use the CPU-only version of TensorFow, but it will run much slower. If we want to use the CPU-only version, just use "tensorflow" instead of "tensorflow-gpu" in the previous command.)

Installing the other necessary packages by issuing the following commands:

(tensorflow1) C:\> conda install -c anaconda protobuf
(tensorflow1) C:\> pip install pillow
(tensorflow1) C:\> pip install lxml
(tensorflow1) C:\> pip install Cython
(tensorflow1) C:\> pip install contextlib2
(tensorflow1) C:\> pip install jupyter
(tensorflow1) C:\> pip install matplotlib
(tensorflow1) C:\> pip install pandas
(tensorflow1) C:\> pip install opencv-python
(Note: The 'pandas' and 'opencv-python' packages are not needed by TensorFlow, but they are used in the Python scripts to generate TFRecords and to work with images, videos, and webcam feeds.)

- **Configure PYTHONPATH environment variable**

A PYTHONPATH variable must be created that points to the \models, \models\research, and \models\research\slim directories. We done this by issuing the following commands (from any directory):

(tensorflow1) C:\> set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim
(Note: Every time the "tensorflow1" virtual environment is exited, the PYTHONPATH variable is reset and needs to be set up again. You can use "echo %PYTHONPATH% to see if it has been set or not.)

- **Compile Protobufs and run setup.py**

Next, compiling the Protobuf files, which are used by TensorFlow to configure model and training parameters. Unfortunately, the short protoc compilation command posted on TensorFlow's Object Detection API installation page does not work on Windows. Every .proto file in the \object_detection\protos directory must be called out individually by the command.

In the Anaconda Command Prompt, changing directories to the \models\research directory:

(tensorflow1) C:\> cd C:\tensorflow1\models\research
Then copy and paste the following command into the command line and press Enter:

protoc --python_out=. .\object_detection\protos\anchor_generator.proto
.\object_detection\protos\argmax_matcher.proto
.\object_detection\protos\bipartite_matcher.proto
.\object_detection\protos\box_coder.proto .\object_detection\protos\box_predictor.proto
.\object_detection\protos\eval.proto .\object_detection\protos\faster_rcnn.proto
.\object_detection\protos\faster_rcnn_box_coder.proto
.\object_detection\protos\grid_anchor_generator.proto
.\object_detection\protos\hyperparams.proto .\object_detection\protos\image_resizer.proto
.\object_detection\protos\input_reader.proto .\object_detection\protos\losses.proto
.\object_detection\protos\matcher.proto
.\object_detection\protos\mean_stddev_box_coder.proto
.\object_detection\protos\model.proto .\object_detection\protos\optimizer.proto
.\object_detection\protos\pipeline.proto .\object_detection\protos\post_processing.proto
.\object_detection\protos\preprocessor.proto
.\object_detection\protos\region_similarity_calculator.proto
.\object_detection\protos\square_box_coder.proto .\object_detection\protos\ssd.proto
.\object_detection\protos\ssd_anchor_generator.proto
.\object_detection\protos\string_int_label_map.proto .\object_detection\protos\train.proto
.\object_detection\protos\keypoint_box_coder.proto
.\object_detection\protos\multiscale_anchor_generator.proto
.\object_detection\protos\graph_rewriter.proto .\object_detection\protos\calibration.proto
.\object_detection\protos\flexible_grid_anchor_generator.proto
This creates a name_pb2.py file from every name.proto file in the \object_detection\protos folder.

Finally, we run the following commands from the C:\tensorflow1\models\research directory:

(tensorflow1) C:\tensorflow1\models\research> python setup.py build
(tensorflow1) C:\tensorflow1\models\research> python setup.py install

- **Test TensorFlow setup to verify it works**

The TensorFlow Object Detection API is now all set up to use pre-trained models for object detection, or to train a new one. we can test it out and verify your installation is working by launching the object_detection_tutorial.ipynb script with Jupyter. From the \object_detection directory, issue this command:

(tensorflow1) C:\tensorflow1\models\research\object_detection> jupyter notebook object_detection_tutorial.ipynb

This opens the script in our default web browser and allows you to step through the code one section at a time. we can step through each section by clicking the “Run” button in the upper toolbar.

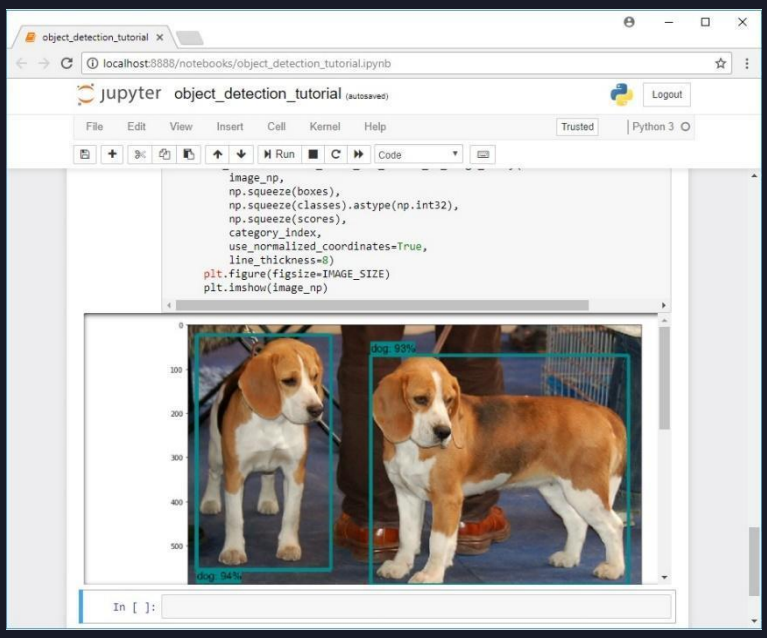


Figure 55: Pre-created model testing on an Image

4.6. Gather and Label Pictures

Now that the TensorFlow Object Detection API is all set up and ready to go, we need to provide the images it will use to train a new detection classifier.

4.6.1. Gather Pictures

TensorFlow needs hundreds of images of an object to train a good detection classifier. To train a robust classifier, the training images should have random objects in the image along with the desired objects, and should have a variety of backgrounds and lighting conditions. There should be some images where the desired object is partially obscured, overlapped with something else, or only halfway in the picture.

Figure 56: Training samples

276 photo (40 for each object) 218 for train and 58 for test.

We maked sure the images aren't too large. They should be less than 200KB each, and their resolution shouldn't be more than 720x1280. The larger the images are, the longer it will take to train the classifier.

After we have all the pictures we need, we moved 20% of them to the \object_detection\images\test directory, and 80% of them to the \object_detection\images\train directory.

4.6.2. Label Pictures

Here comes the fun part! With all the pictures gathered, it's time to label the desired objects in every picture. We used a tool called LabelImg.

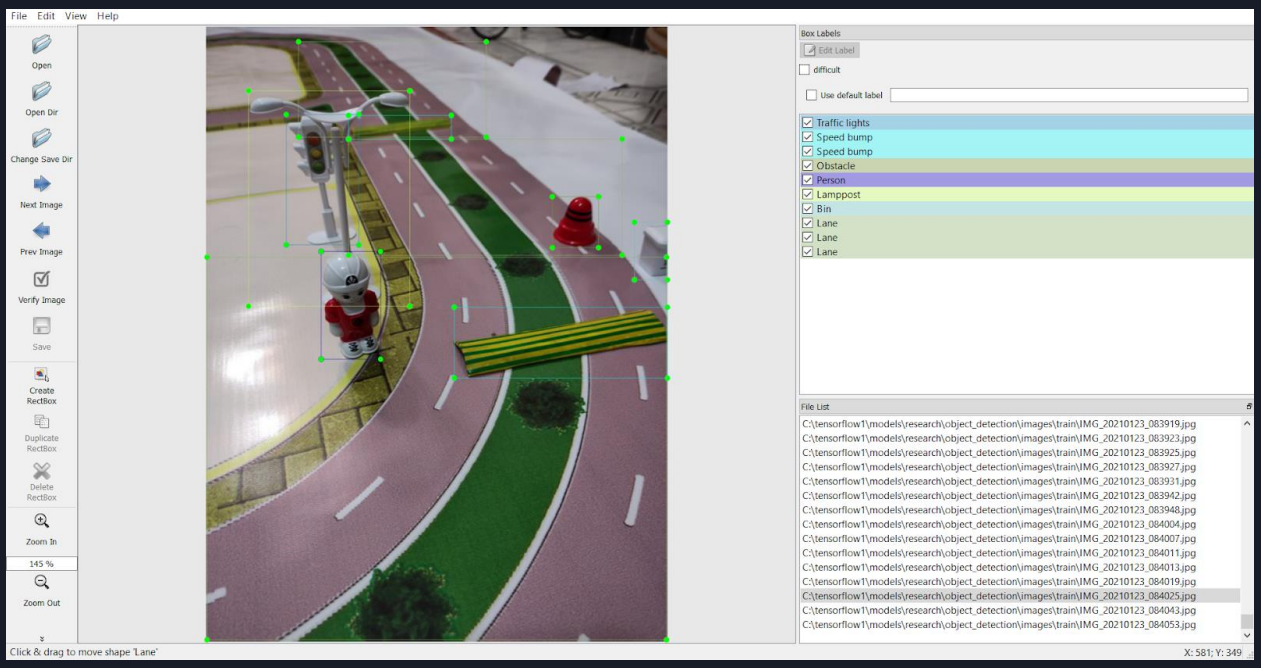


Figure 57: Labelling images

LabelImg saves a .xml file containing the label data for each image. These .xml files will be used to generate TFRecords, which are one of the inputs to the TensorFlow trainer. Once we have labeled and saved each image, there will be one .xml file for each image in the \test and \train directories.

4.6.3. Generate Training Data

With the images labeled, it's time to generate the TFRecords that serve as input data to the TensorFlow training model.

First, the image .xml data will be used to create .csv files containing all the data for the train and test images. From the \object_detection folder, issue the following command in the Anaconda command prompt:

(tensorflow1) C:\tensorflow1\models\research\object_detection> python xml_to_csv.py

This creates a train_labels.csv and test_labels.csv file in the \object_detection\images folder.

Next, opening the generate_tfrecord.py file in a text editor. Replacing the label map starting at line 31 with your own label map, where each object is assigned an ID number.

generate_tfrecord.py:

```
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'Speed bump':
        return 1
    elif row_label == 'Traffic lights':
        return 2
    elif row_label == 'Lane':
        return 3
    elif row_label == 'Lamppost':
        return 4
    elif row_label == 'Person':
        return 5
    elif row_label == 'Bin':
        return 6
    elif row_label == 'Obstacle':
        return 7
    else:
        return 0
```

Figure 58: TFRecord python file configuring

Then, generating the TFRecord files by issuing these commands from the \object_detection folder:

python generate_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\train --output_path=train.record

python generate_tfrecord.py --csv_input=images\test_labels.csv --image_dir=images\test --output_path=test.record

These generate a train.record and a test.record file in \object_detection. These will be used

train_ labels.csv

	filename	width	height	class	xmin	ymin	xmax	ymax
2	cam_imag	480	270	queen	173	24	260	137
3	cam_imag	480	270	queen	165	135	253	251
4	cam_imag	480	270	ten	255	96	337	208
5	cam_imag	960	540	ten	501	116	700	353
6	cam_imag	960	540	queen	261	124	453	370
7	cam_imag	960	540	nine	225	96	490	396
8	cam_imag	960	540	king	362	149	560	389
9	cam_imag	960	540	jack	349	142	550	388
10	cam_imag	960	540	jack	297	167	512	420
11	cam_imag	960	540	ace	367	181	589	457
12	cam_imag	960	540	ace	303	155	525	456
13	cam_imag	960	540	ace	316	125	547	451
14	cam_imag	960	540	ace	390	86	605	365
15	cam_imag	960	540	jack	357	97	578	379
16	cam_imag	960	540	queen	291	119	542	422
17	cam_imag	960	540	queen	319	54	556	346
18	cam_imag	960	540	queen	296	94	522	370
19	cam_imag	960	540	queen	286	108	517	389
20	cam_imag	960	540	king	444	113	685	411
21	cam_imag	960	540	king	329	94	584	410
22	cam_imag	960	540	king	350	89	587	383
23	cam_imag	960	540	jack	328	145	555	440
24	cam_imag	960	540	ten	268	115	502	426

Figure 59: CSV created data labels

4.7. Create Label Map and Configure Training

The last thing to do before training is to create a label map and edit the training configuration file.

4.7.1. Label map

The label map tells the trainer what each object is by defining a mapping of class names to class ID numbers. We used a text editor to create a new file and save it as labelmap.pbtxt

```
item {
  id: 1
  name: 'Speed bump'
}

item {
  id: 2
  name: 'Traffic lights'
}

item {
  id: 3
  name: 'Lane'
}

item {
  id: 4
  name: 'Lamppost'
}

item {
  id: 5
  name: 'Person'
}

item {
  id: 6
  name: 'Bin'
}

item {
  id: 7
  name: 'Obstacle'
}
```

Figure 60: Labelmap file

The label map ID numbers should be the same as what is defined in the generate_tfrecord.py file.

4.7.2. Configure training

Finally, the object detection training pipeline must be configured. It defines which model and what parameters will be used for training. This is the last step before running training. So, we made some in ssd_mobilenet_v1_coco.config, before start the training.

```
 7  model {
 8    faster_rcnn {
 9      num_classes: 7
10      image_resizer {
11        keep_aspect_ratio_resizer {
12          min_dimension: 600
13          max_dimension: 1024
14        }
15      }
```

Figure 61: Configuring number of classes

4.8. Run the Training

From the \object_detection directory, issue the following command to begin training:

python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_pets.config

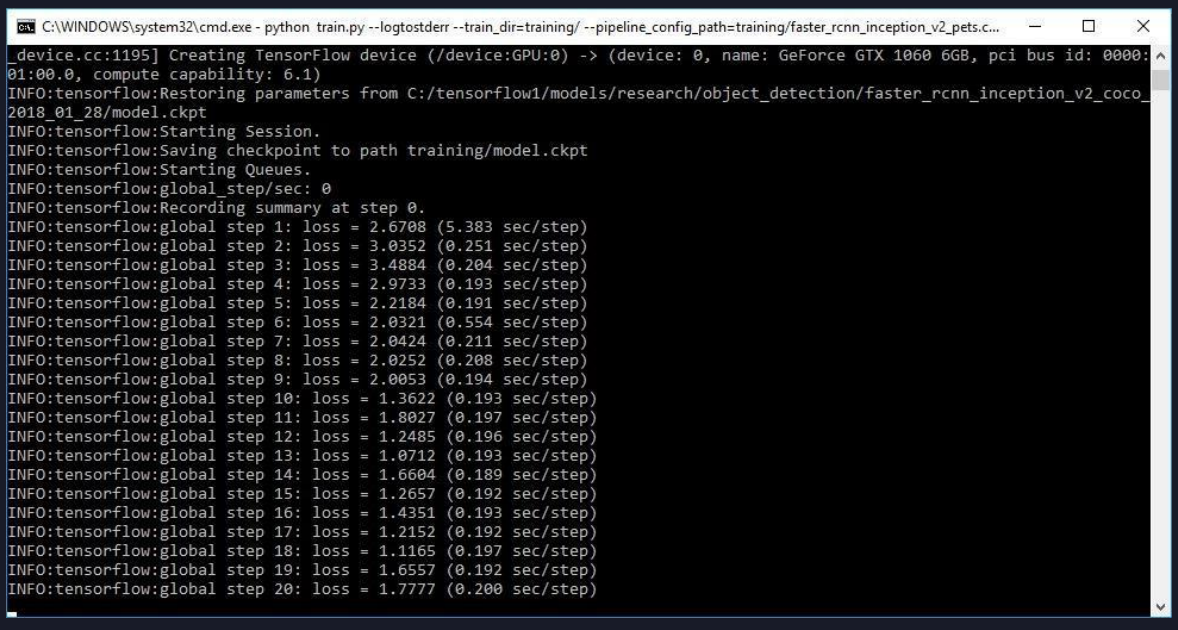


Figure 62: Running the training

Each step of training reports the loss. It will start high and get lower and lower as training progresses.

Checkpoints

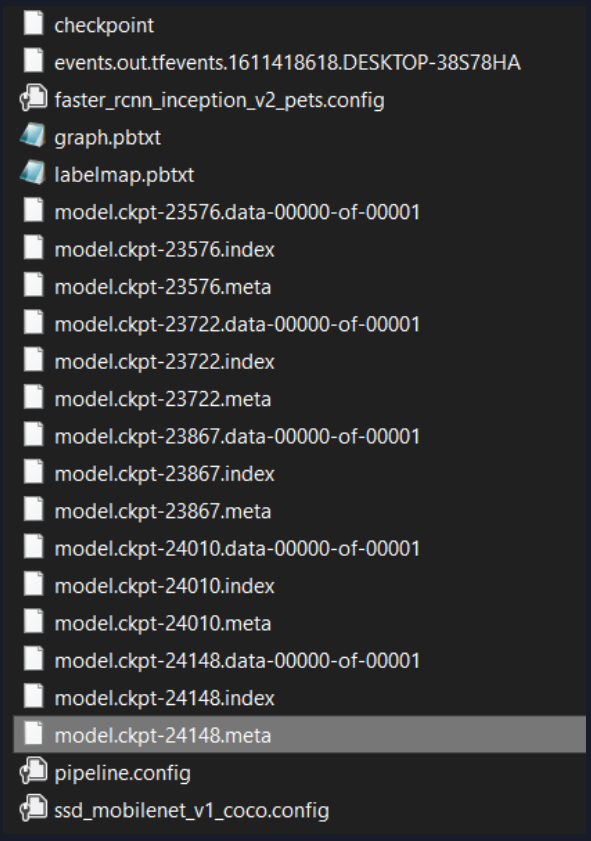


Figure 63: Checking the training file

We can view the progress of the training job by using TensorBoard. To do this, we opened a new instance of Anaconda Prompt, activating the tensorflow1 virtual environment, changed to the C:\tensorflow1\models\research\object_detection directory, and issue the following command:

(tensorflow1) C:\tensorflow1\models\research\object_detection>tensorboard --logdir=training

This will create a webpage on your local machine at YourPCName:6006, which can be viewed through a web browser. The TensorBoard page provides information and graphs that show how the training is progressing. One important graph is the Loss graph, which shows the overall loss of the classifier over time.

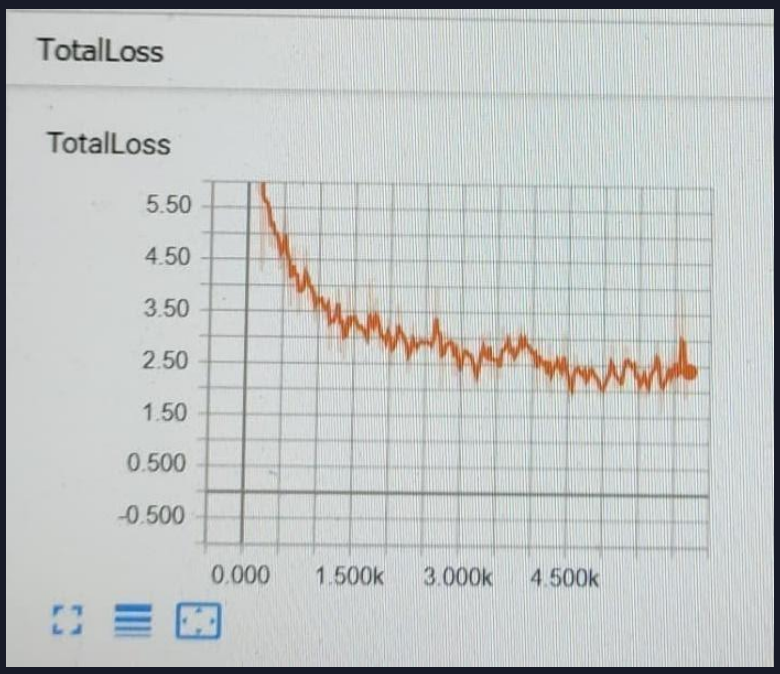


Figure 64: TensorBoard

The training routine periodically saves checkpoints about every five minutes.

The checkpoint at the highest number of steps will be used to generate the frozen inference graph.

4.9. All commands:

```
>conda create -n tensorflow1 pip python=3.6.4
>activate tensorflow1
>pip install --ignore-installed --upgrade tensorflow-gpu==1.15
>conda install -c anaconda protobuf
>pip install pillow
>pip install lxml
>pip install Cython
>pip install contextlib2
>pip install jupyter
>pip install matplotlib
>pip install pandas
>pip install opencv-python
>set PYTHONPATH=C:\tensorflow1\models;C:\tensorflow1\models\research;C:\tensorflow1\models\research\slim
>set PATH=%PATH%;PYTHONPATH
>cd C:\tensorflow1\models\research
>protoc --python_out=. .\object_detection\protos\anchor_generator.proto .\object_detection\protos\argmax_matcher.proto .\object_detection\protos\bipartite_matcher.proto .\object_detection\p
x_coder.proto .\object_detection\protos\ssd.proto .\object_detection\protos\ssd_anchor_generator.proto .\object_detection\protos\string_int_label_map.proto .\object_detection\protos\train.p
>python setup.py build
>python setup.py install
>cd C:\tensorflow1\models\research\object_detection
>python setup.py build
>python setup.py install
>cd C:\tensorflow1\models\research\object_detection
>jupyter notebook object_detection_tutorial.ipynb
>python xml_to_csv.py
>python generate_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\train --output_path=train.record
>python generate_tfrecord.py --csv_input=images\test_labels.csv --image_dir=images\test --output_path=test.record
>python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/ssd_mobilenet_v1_coco.config
>python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/ssd_mobilenet_v1_coco.config --trained_checkpoint_prefix training/model.ckpt-24148 --output_direc
>python export_tflite_ssd_graph.py --pipeline_config_path=%CONFIG_FILE% --trained_checkpoint_prefix=%CHECKPOINT_PATH% --output_directory=%OUTPUT_DIR% --add_postprocessing_op=true
>idle
(tensorflow1) C:\tensorflow1\models\research\object_detection>tensorboard --logdir=training
set CONFIG_FILE=C:\\tensorflow1\models\research\object_detection\training\ssd_mobilenet_v1_coco.config
set CHECKPOINT_PATH=C:\\tensorflow1\models\research\object_detection\training\model.ckpt-24148
set OUTPUT_DIR=C:\\tensorflow1\models\research\object_detection\TFLite_model
```

Figure 65: List of commands

4.10. Export Inference Graph

Now that training is complete, the last step is to generate the frozen inference graph (.pb file). From the \object_detection folder, issue the following command, where "XXXX" in "model.ckpt-XXXX" should be replaced with the highest-numbered .ckpt file in the training folder (in our case was 24148)

python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/faster_rcnn_inception_v2_pets.config --trained_checkpoint_prefix training/model.ckpt-24148 --output_directory inference_graph
This creates a frozen_inference_graph.pb file in the \object_detection\inference_graph folder. The .pb file contains the object detection classifier.

Inference Graph

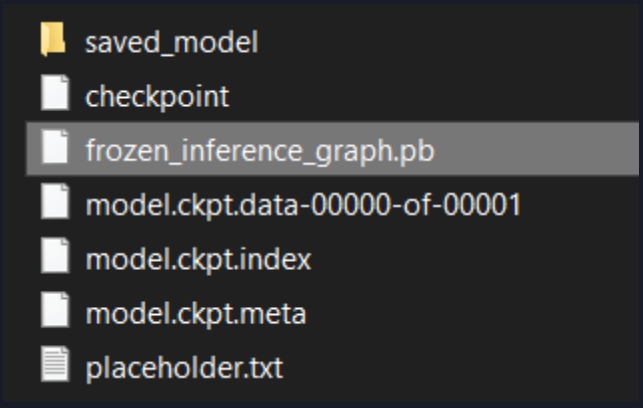


Figure 66: Inference Graph

4.11. Use Our Newly Trained Object Detection Classifier

To test our object detector, move a picture of the object or objects into the \object_detection folder, and change the IMAGE_NAME variable in the Object_detection_image.py to match the file name of the picture. Alternatively, we can use a video of the objects (using Object_detection_video.py), or just plug in a USB webcam and point it at the objects (using Object_detection_webcam.py)
(the last method that we used).

To run any of the scripts, typying "idle" in the Anaconda Command Prompt (with the "tensorflow1" virtual environment activated) and press ENTER. This will open IDLE, and from there, we can open any of the scripts and run them.

The object detector initialized for about 10 seconds and then displaed a window showing any objects it's detected in the image!

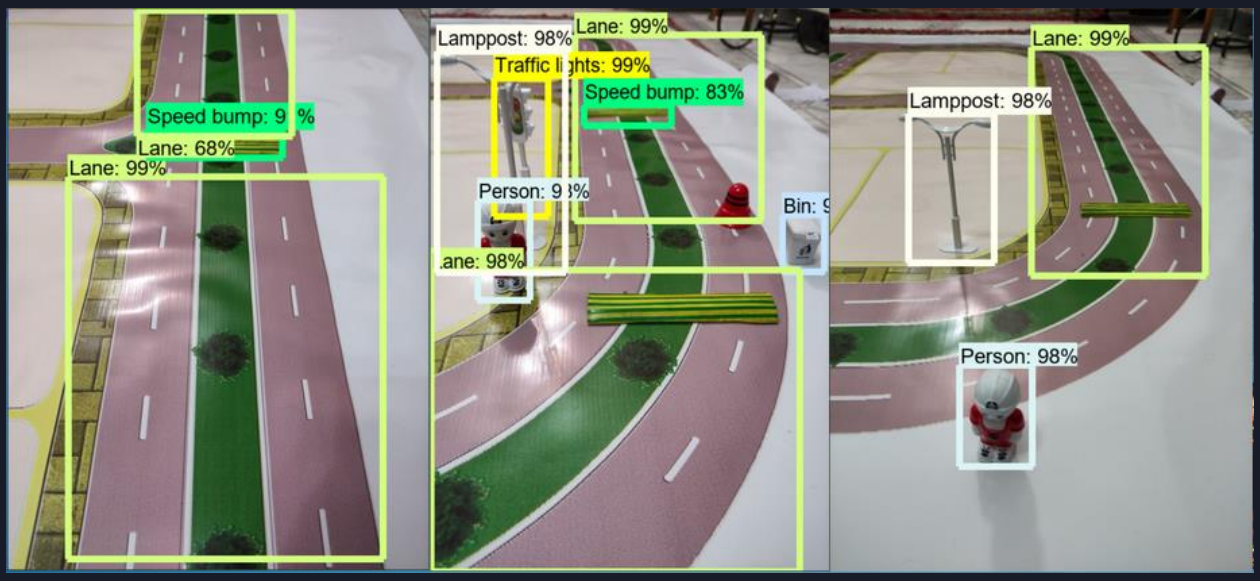


Figure 67: Testing our model on images

4.12. Testing Our Tensorflow Lite model on Raspberry Pi

Setting up TensorFlow Lite on the Raspberry Pi is much easier than regular TensorFlow! As the TensorFlow Lite consumes lower power and has higher speed than regular TensorFlow.

These are the steps needed to set up TensorFlow Lite:

1. Connecting the raspberry pi.
2. Updating the raspberry pi.
3. Downloading specific repository and creating virtual environment.
4. Installing TensorFlow and OpenCV
5. Setting up TensorFlow Lite detection model.
6. Running TensorFlow Lite model.

- **Connecting the raspberry pi.**

RJ45 Ethernet cable is used to connect raspberry to laptop.

Software programs used:

- (1Advanced IP Scanner to know the IP of RPI.

- 2(MobaXterm to see the RPI.

Figure 69: Advanced IP App

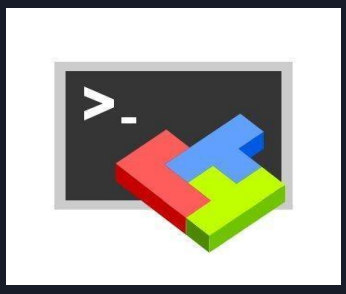


Figure 68: MobaXterm App

a. Advanced IP Scanner to know the IP of RPI.

This program is used to detect the ip address of the raspberry pi which will be used later in MobaXterm program.

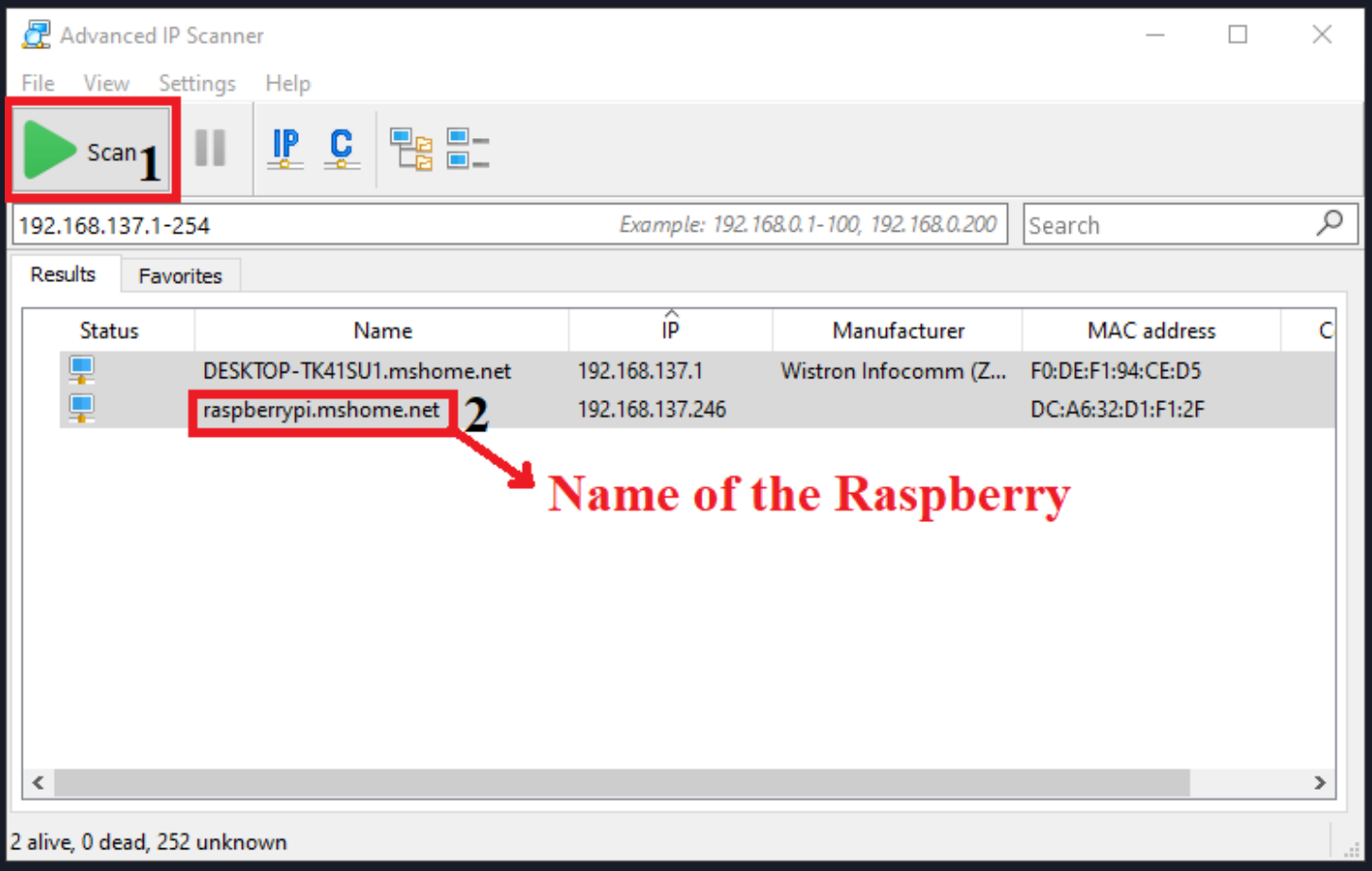


Figure 70: Getting the Raspberry Pi IP

b. MobaXterm to open the RPI.

- The steps are:
  - Press Session => SSH
  - Write the name of the raspberrypi which was detected by the advance ip scanner program.

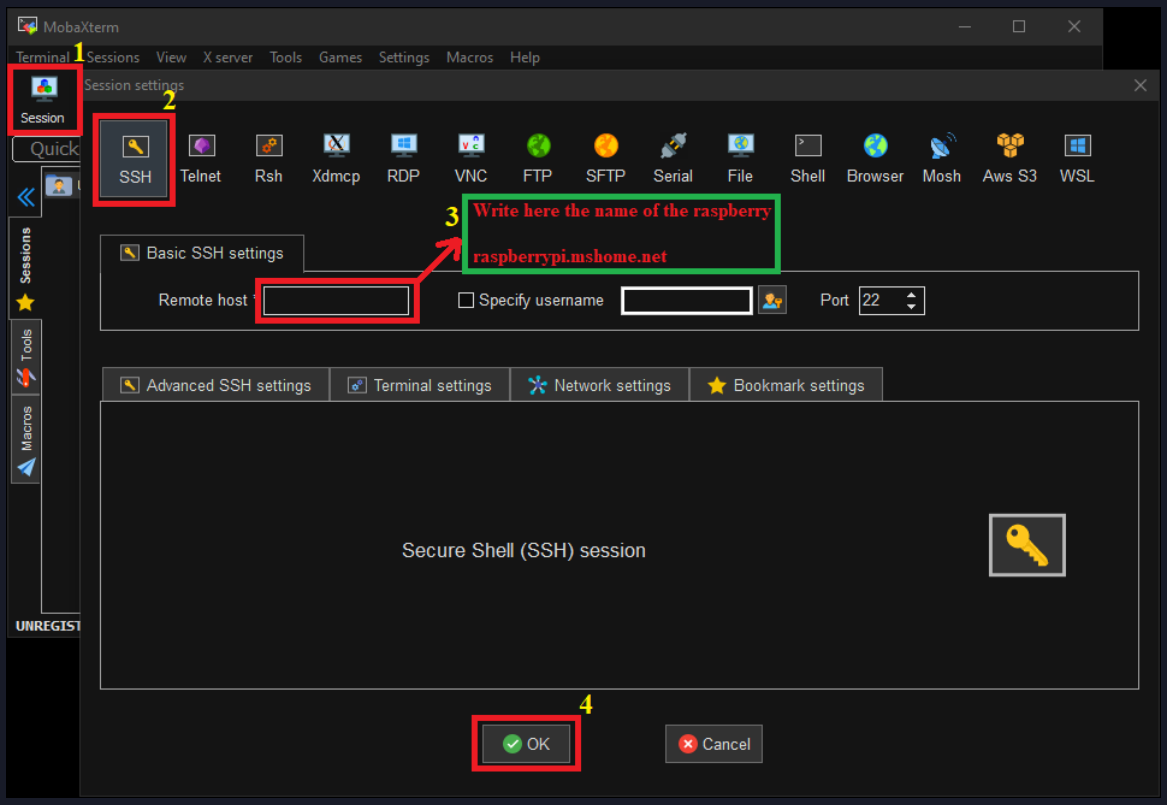


Figure 71: Connecting to the Raspberry Pi IP

- Enter the username and password of the raspberry pi.

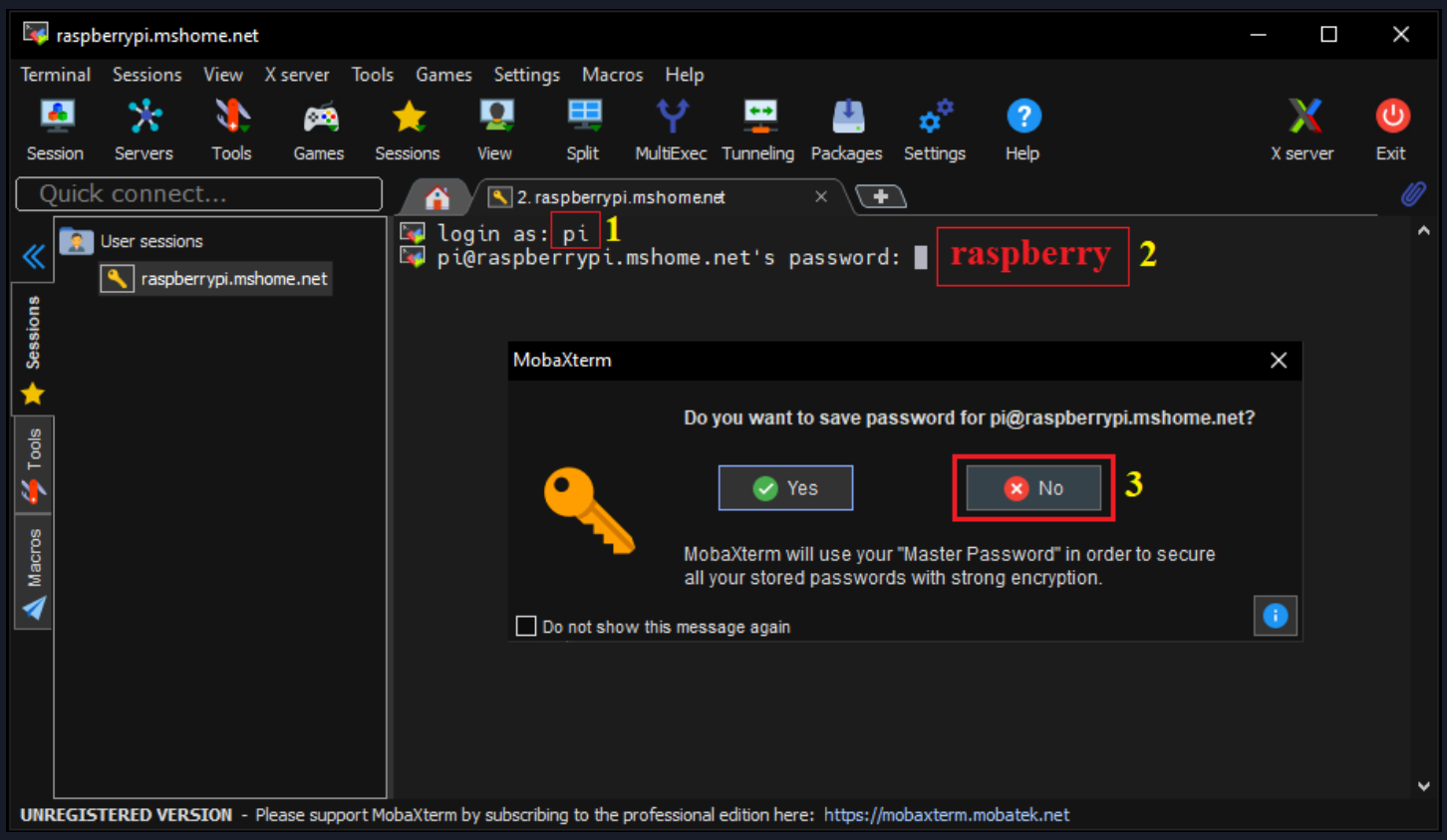


Figure 72: Logging into the Raspberry pi

- **The raspberry pi will be opened now.**

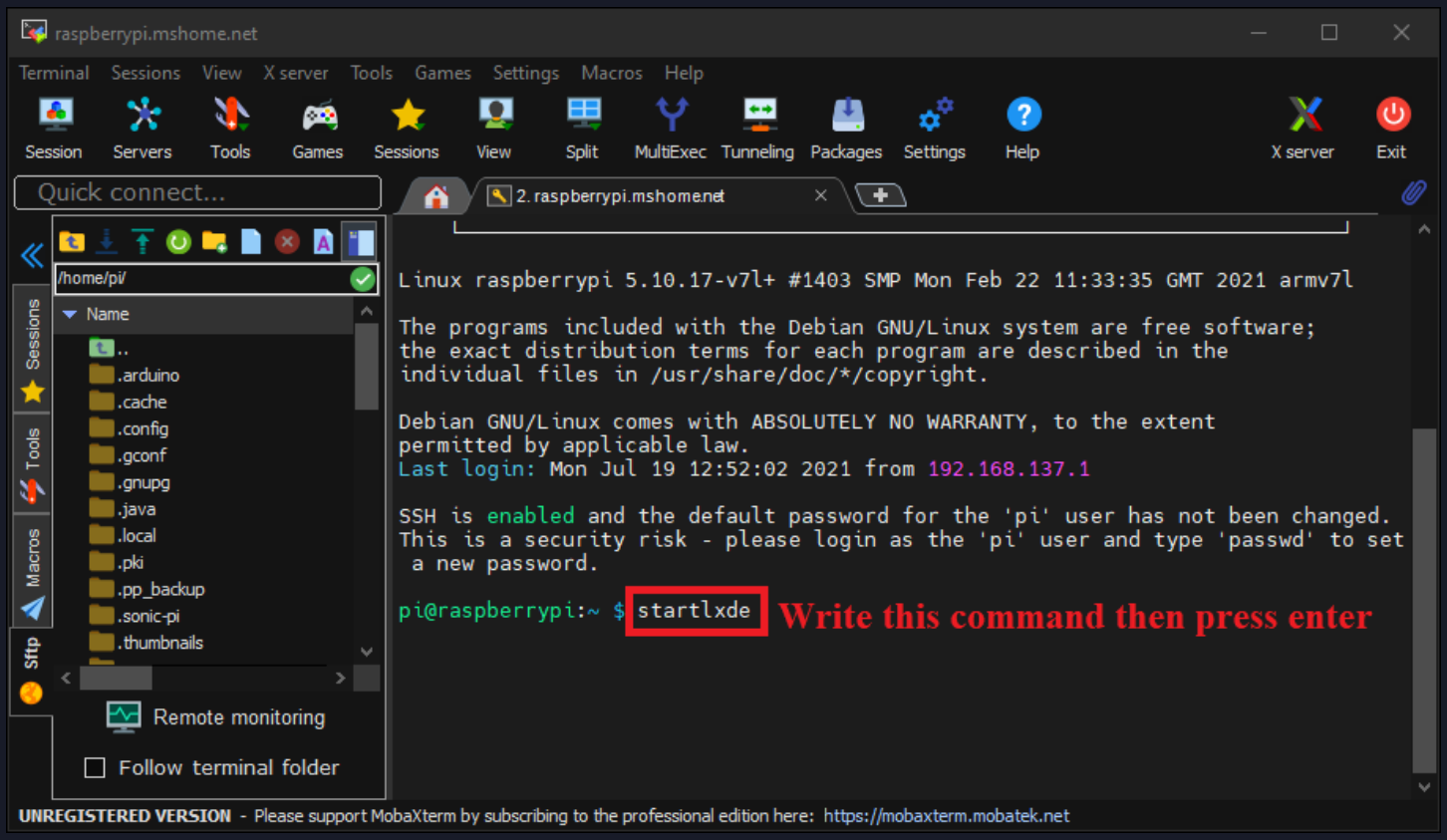


Figure 73: Opening the Raspberry Pi GUI

c. Updating the Raspberry Pi.

- The Raspberry Pi needs to be fully updated depending on how long it's been since we've updated our Pi.

```
sudo apt-get update
sudo apt-get dist-upgrade
```

- Enabling the camera interface in the Raspberry Pi.

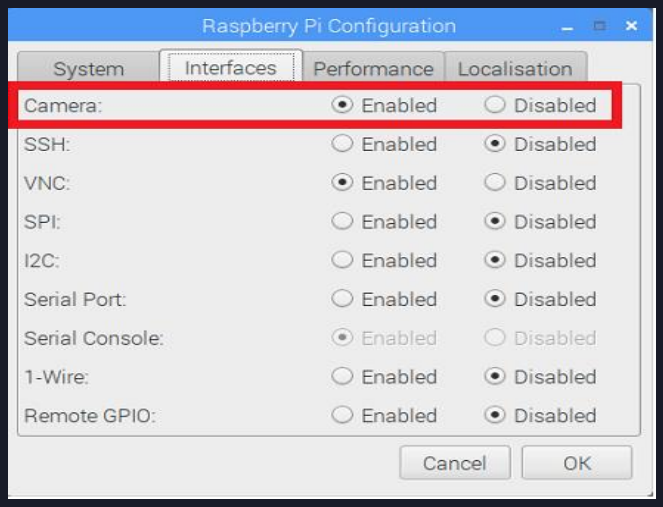


Figure 74: Configuring the Raspberry Pi

d. Downloading specific repository and creating virtual environment.

- **The repository contains the scripts we'll use to run TensorFlow Lite, as well as a shell script that will make installing everything easier.**

```
git clone https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi.git
```

This downloads everything into a folder called TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi. We renamed the folder to "tflite1" and then cd into it:

```
mv TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi tflite1
cd tflite1
```

- **Creating a virtual environment because it prevents any conflicts between versions of package libraries that may already be installed on the RPi. Keeping TensorFlow installed in its own environment allows us to avoid version conflicts. It's called "tflite1-env".**

```
sudo pip3 install virtualenv

python3 -m venv tflite1-env
```

- **This will create a folder called tflite1-env inside the tflite1 directory. The tflite1-env folder will hold all the package libraries for this environment. Next, activate the environment by issuing:**

```
source tflite1-env/bin/activate
```

e. Installing TensorFlow lite dependencies and OpenCV.

- **OpenCV is not needed to run TensorFlow Lite, but the object detection scripts in this repository use it to grab images and draw detection results on them.**
- **To make things easier, we wrote a shell script that will automatically download and install all the packages and dependencies:**

```
bash get_pi_requirements.sh
```

f. Setting up our TensorFlow Lite detection model.

- **A detection model has two files associated with it: a detect.tflite file (which is the model itself) and a labelmap.txt file (which provides a labelmap for the model).**

g. Running the model.

- **After making sure that our webcam or Picamera has been plugged in.**

```
python3 TFLite_detection_webcam.py --modeldir=Sample_TFLite_model
```

- **A window will appear showing the webcam feed. Detected objects will have bounding boxes and labels displayed on them in real time.**

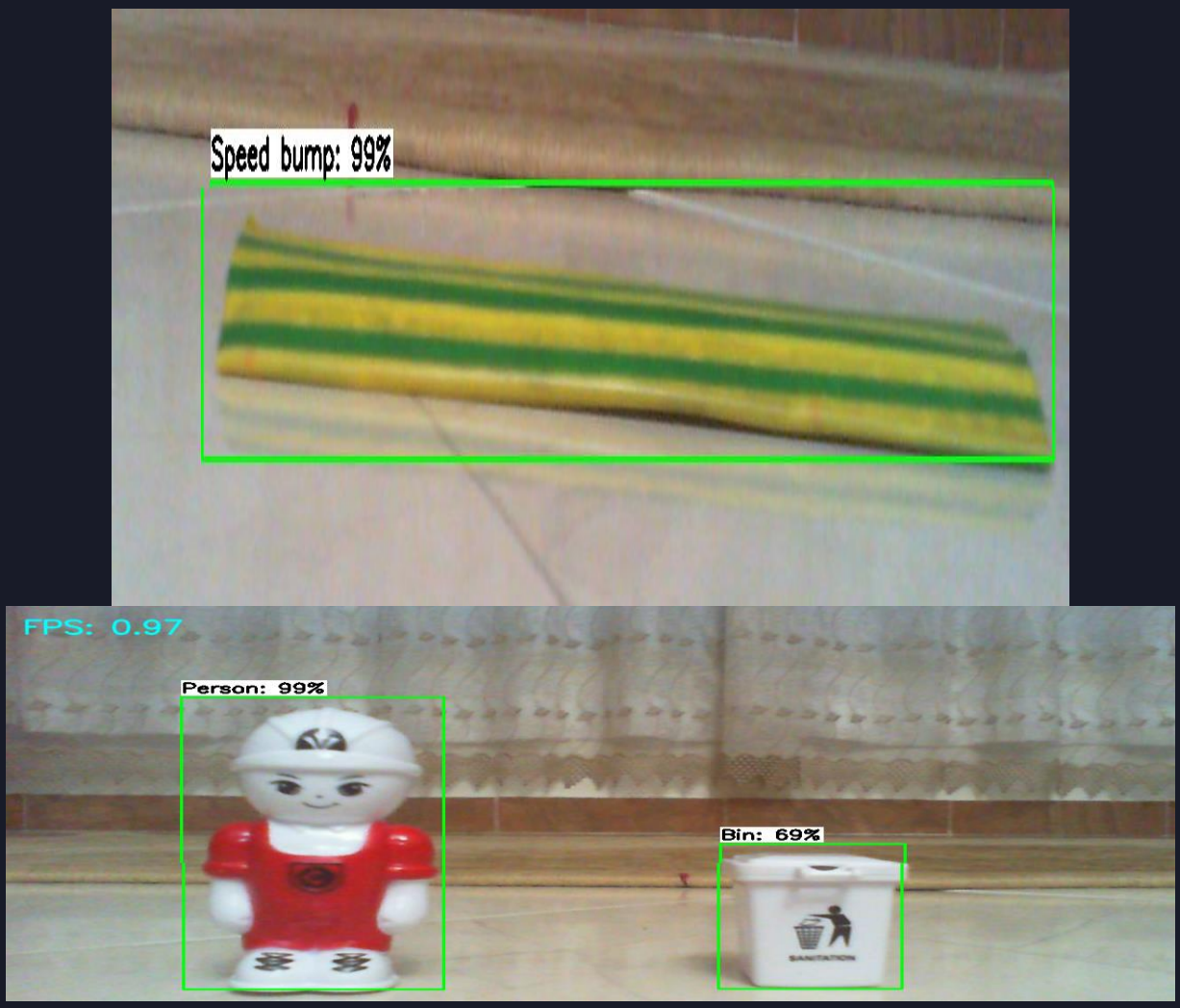
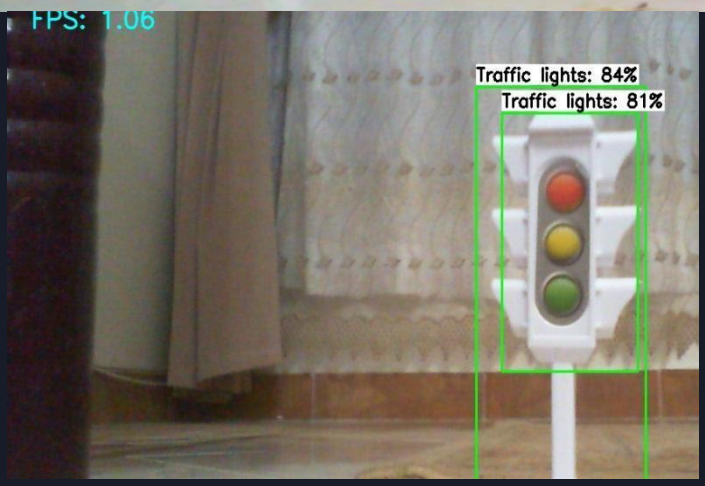
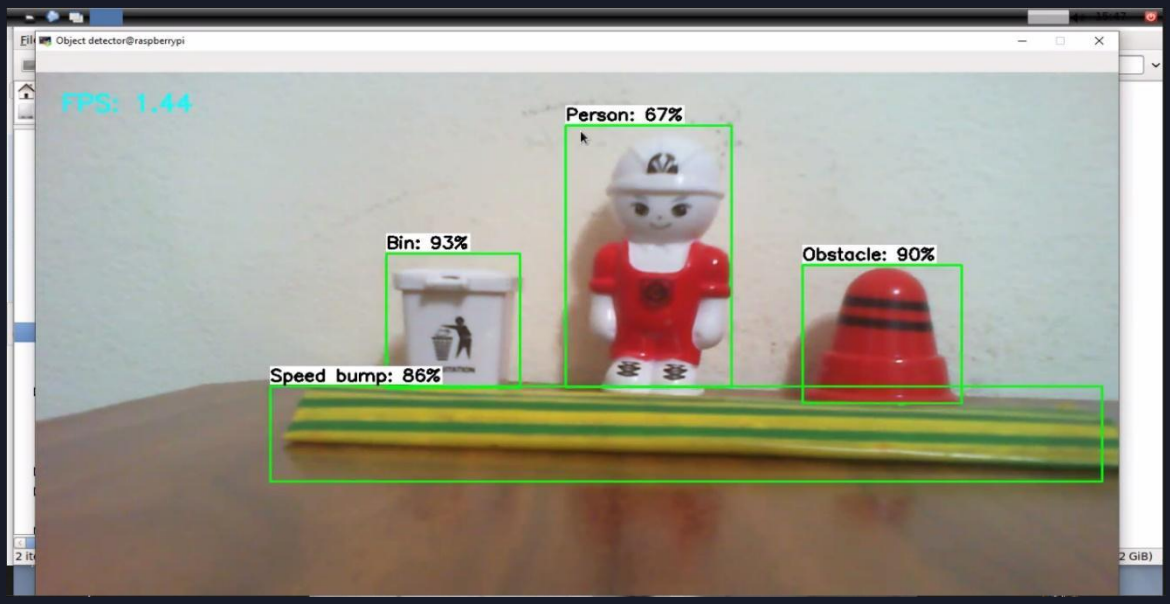
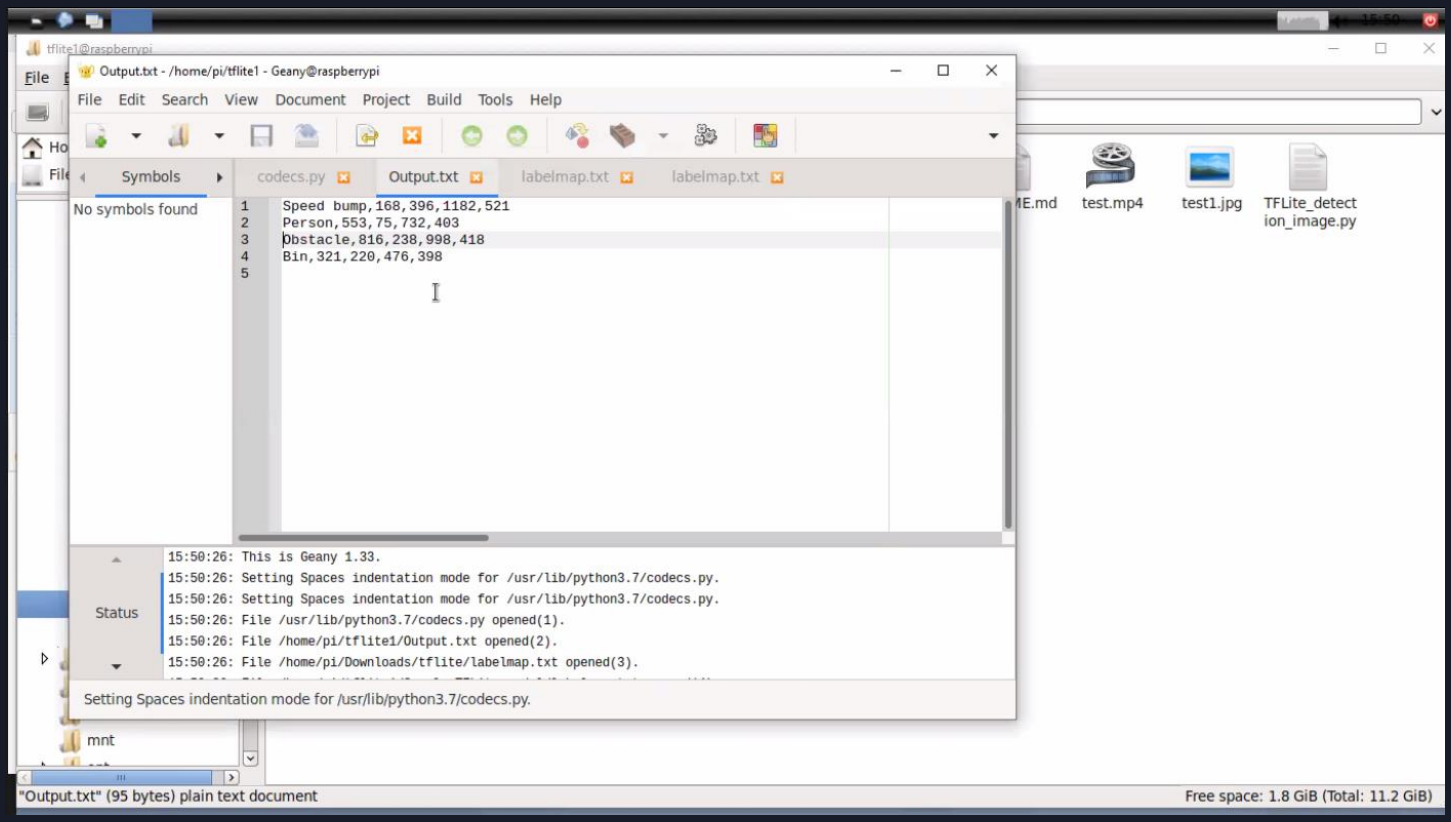


Figure 75: Testing Object-Detection model on the raspberry pi



h. Printing the detected objects' names in a text file to send it to TivaC later.





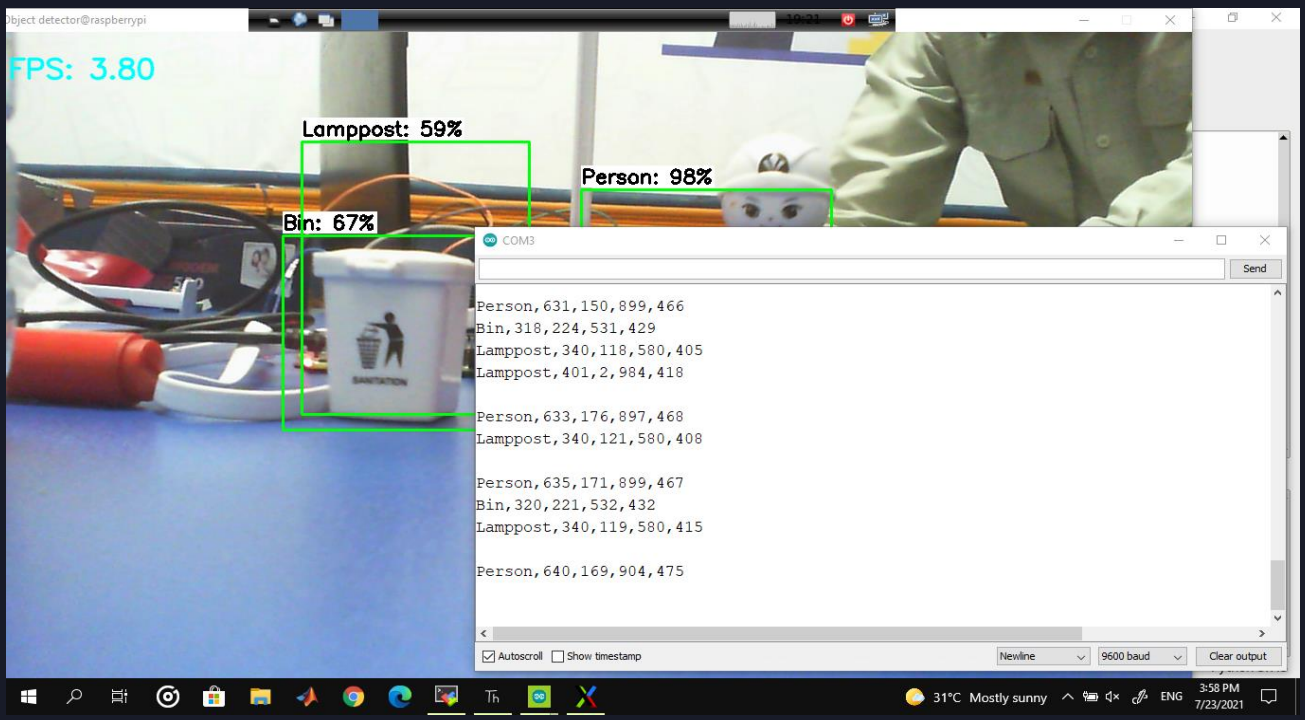


Figure 76:: Printing the Output data in a txt file

i. Webcam.py Flow Chart

Figure 77: Webcam.py Flowchart

```
    # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std

    # Perform the actual detection by running the model with the image as input
    interpreter.set_tensor(input_details[0]['index'],input_data)
    interpreter.invoke()

    # Retrieve detection results
    boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of detected objects
    classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected objects
    scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objects
    #num = interpreter.get_tensor(output_details[3]['index'])[0]  # Total number of detected objects (inaccurate and not needed)

    text_file = open("Output.txt", "w")

    # Loop over all detections and draw detection box if confidence is above minimum threshold
    for i in range(len(scores)):
        if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

            # Get bounding box coordinates and draw box
            # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min()
            ymin = int(max(1,(boxes[i][0] * imH)))
            xmin = int(max(1,(boxes[i][1] * imW)))
            ymax = int(min(imH,(boxes[i][2] * imH)))
            xmax = int(min(imW,(boxes[i][3] * imW)))

            cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

            # Draw label
            object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
            label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
```

```
            # Draw label
            object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
            label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
            labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
            label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
            cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED)
            # Draw white box to put label text in
            cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

            text_file.write("%s,%d,%d,%d,%d\n" %(object_name,xmin,ymin,xmax,ymax))

    # Draw framerate in corner of frame
    cv2.putText(frame,'FPS: {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)

    # All the results have been drawn on the frame, so it's time to display it.
    cv2.imshow('Object detector', frame)

    # Calculate framerate
    t2 = cv2.getTickCount()
    time1 = (t2-t1)/freq
    frame_rate_calc= 1/time1

    # Press 'q' to quit
    if cv2.waitKey(1) == ord('q'):
        break

        text_file.close()

# Clean up
cv2.destroyAllWindows()
videostream.stop()
```

Figure 78: configuring the python file to write detected data in the txt file

4.13. Optimizing our model

4.13.1. Overview:

Edge devices often have limited memory or computational power. Various optimizations can be applied to models so that they can be run within these constraints. In addition, some optimizations allow the use of specialized hardware for accelerated inference.

TensorFlow Lite and the TensorFlow Model Optimization Toolkit provide tools to minimize the complexity of optimizing inference.

4.13.2. Why models should be optimized ?

There are several main ways model optimization can help with application development.

- **Size reduction**

Some forms of optimization can be used to reduce the size of a model. Smaller models have the following benefits:

- **Smaller storage size:** Smaller models occupy less storage space on your users' devices. For example, an Android app using a smaller model will take up less storage space on a user's mobile device.
- **Smaller download size:** Smaller models require less time and bandwidth to download to users' devices.
- **Less memory usage:** Smaller models use less RAM when they are run, which frees up memory for other parts of your application to use, and can translate to better performance and stability.

Quantization can reduce the size of a model in all of these cases, potentially at the expense of some accuracy. Pruning and clustering can reduce the size of a model for download by making it more easily compressible.

- **Latency reduction**

Latency is the amount of time it takes to run a single inference with a given model. Some forms of optimization can reduce the amount of computation required to run inference using a model, resulting in lower latency. Latency can also have an impact on power consumption.

Currently, quantization can be used to reduce latency by simplifying the calculations that occur during inference, potentially at the expense of some accuracy.

- **Accelerator compatibility**

Some hardware accelerators, such as the Edge TPU, can run inference extremely fast with models that have been correctly optimized.

Generally, these types of devices require models to be quantized in a specific way. See each hardware accelerator's documentation to learn more about their requirements.

- **Trade-offs**

Optimizations can potentially result in changes in model accuracy, which must be considered during the application development process.

The accuracy changes depend on the individual model being optimized, and are difficult to predict ahead of time. Generally, models that are optimized for size or latency will lose a small amount of accuracy. Depending on your application, this may or may not impact your users' experience. In rare cases, certain models may gain some accuracy as a result of the optimization process.

4.14. Types of optimizations

TensorFlow Lite currently supports optimization via quantization, pruning, and clustering.

These are part of the TensorFlow Model Optimization Toolkit, which provides resources for model optimization techniques that are compatible with TensorFlow Lite.

● Quantization

Quantization works by reducing the precision of the numbers used to represent a model's parameters, which by default are 32-bit floating point numbers. This results in a smaller model size and faster computation.

The following types of quantization are available in TensorFlow Lite:

Technique	Data requirements	Size reduction	Accuracy	Supported hardware
Post-training float16 quantization	No data	Up to 50%	Insignificant accuracy loss	CPU, GPU
Post-training dynamic range quantization	No data	Up to 75%	Accuracy loss	CPU, GPU (Android)
Post-training integer quantization	Unlabelled representative sample	Up to 75%	Smaller accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP
Quantization-aware training	Labelled training data	Up to 75%	Smallest accuracy loss	CPU, GPU (Android), EdgeTPU, Hexagon DSP

Figure 79: Types of Quantization

The following decision tree helps you select the quantization schemes you might want to use for your model, simply based on the expected model size and accuracy:

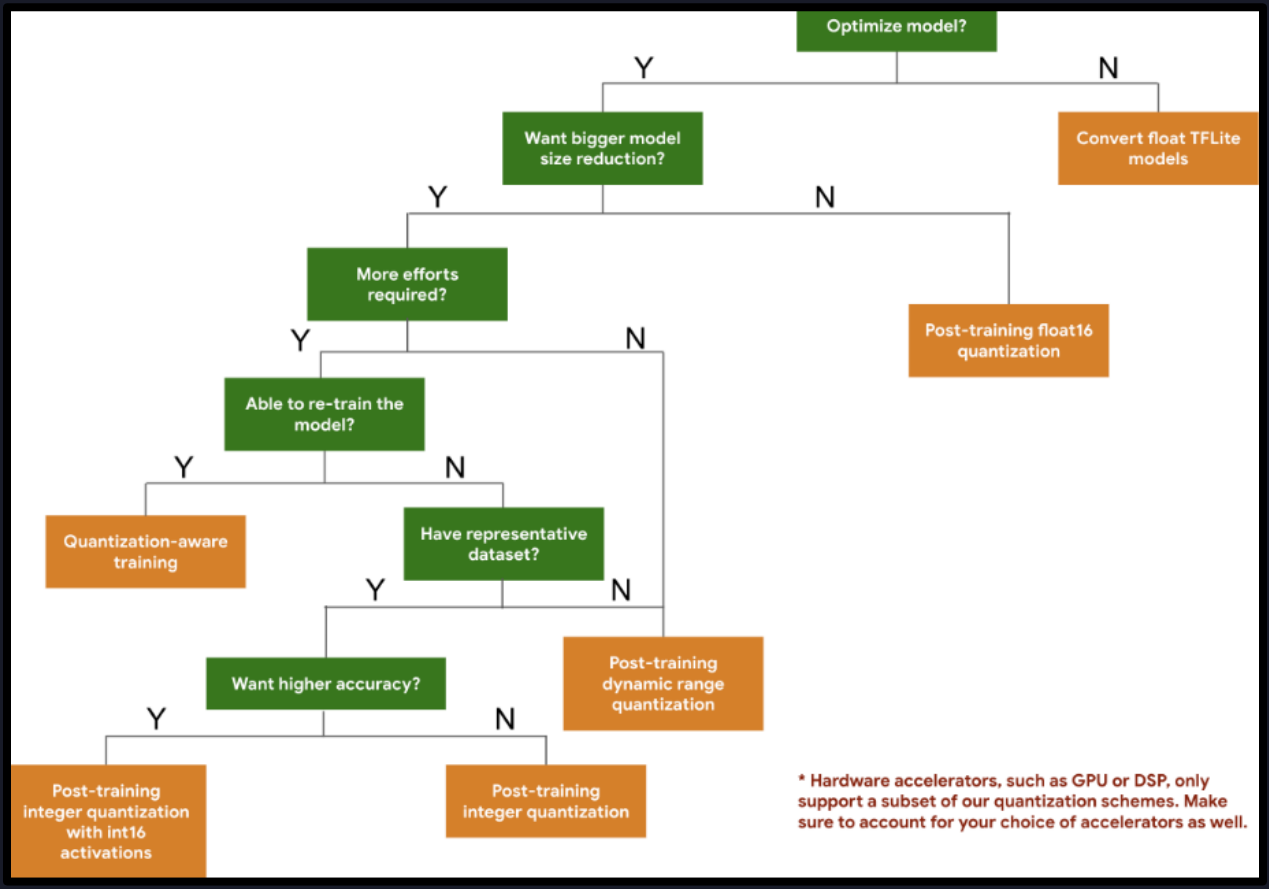


Figure 80: Quantization Algorithm

Below are the latency and accuracy results for post-training quantization and quantization-aware training on a few models. All latency numbers are measured on Pixel 2 devices using a single big core CPU. As the toolkit improves, so will the numbers here:

Model	Top-1 Accuracy (Original)	Top-1 Accuracy (Post Training Quantized)	Top-1 Accuracy (Quantization Aware Training)	Latency (Original) (ms)	Latency (Post Training Quantized) (ms)	Latency (Quantization Aware Training) (ms)	Size (Original) (MB)	Size (Optimized) (MB)
Mobilenet-v1-1-224	0.709	0.657	0.70	124	112	64	16.9	4.3
Mobilenet-v2-1-224	0.719	0.637	0.709	89	98	54	14	3.6
Inception_v3	0.78	0.772	0.775	1130	845	543	95.7	23.9
Resnet_v2_101	0.770	0.768	N/A	3973	2868	N/A	178.3	44.9

Table 1 Benefits of model quantization for select CNN models

Figure 81: Training models comparison

Full integer quantization with int16 activations and int8 weights:

Quantization with int16 activations is a full integer quantization scheme with activations in int16 and weights in int8. This mode can improve accuracy of the quantized model in comparison to the full integer quantization scheme with both activations and weights in int8 keeping a similar model size. It is recommended when activations are sensitive to the quantization.

Below are the accuracy results for some models that benefit from this mode.:

Model	Accuracy metric type	Accuracy (float32 activations)	Accuracy (int8 activations)	Accuracy (int16 activations)
Wav2letter	WER	6.7%	7.7%	7.2%
DeepSpeech 0.5.1 (unrolled)	CER	6.13%	43.67%	6.52%
YoloV3	mAP(IOU=0.5)	0.577	0.563	0.574
MobileNetV1	Top-1 Accuracy	0.7062	0.694	0.6936
MobileNetV2	Top-1 Accuracy	0.718	0.7126	0.7137
MobileBert	F1(Exact match)	88.81(81.23)	2.08(0)	88.73(81.15)

Table 2 Benefits of model quantization with int16 activations

Figure 82: Benefits of model Quantization

There are two other methods for model optimization, but they aren’t common and we didn’t use any of them for our model, which they are:

• Pruning

Pruning works by removing parameters within a model that have only a minor impact on its predictions. Pruned models are the same size on disk, and have the same runtime latency, but can be compressed more effectively. This makes pruning a useful technique for reducing model download size.
In the future, TensorFlow Lite will provide latency reduction for pruned models.

• Clustering

Clustering works by grouping the weights of each layer in a model into a predefined number of clusters, then sharing the centroid values for the weights belonging to each individual cluster. This reduces the number of unique weight values in a model, thus reducing its complexity.
As a result, clustered models can be compressed more effectively, providing deployment benefits similar to pruning.
For cases where the accuracy and latency targets are not met, or hardware accelerator support is important, quantization-aware training
If you want to further reduce your model size, you can try pruning and/or clustering prior to quantizing your models.

4.15. TFLite Model Optimization (Used Method):

Here, we will retrain our model using:

“ ssd_mobilenet_v2_quantized_300x300 _coco_2019_01_03 ”

Instead of using:

“ ssd_mobilenet_v1_coco_2017_11_17 ”

As Quantized models are faster as they use 8-bit integer values instead of 32-bit floating values within the neural network, allowing them to run much more efficiently on GPUs or specialized TPUs (TensorFlow Processing Units) like the Google Coral TPU Accelerator.

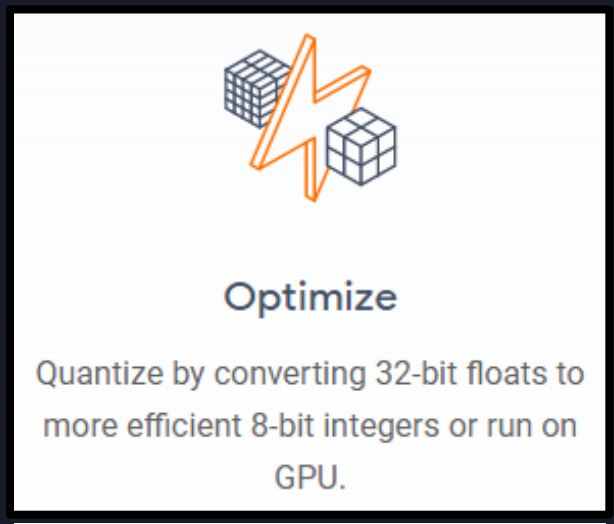


Figure 83: Used Optimization method

This time we used a google cloud (Google Colaboratory) instead of using our PCs as they give users better and faster resources have everything pre-installed and ready to do the job perfectly. Also, it’s easier to run codes on these machines with few or no errors.

4.16. What is Colaboratory?

Figure 84: Google Colaboratory (Colab)

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with:

- Zero configuration required
- Free access to GPUs
- Easy sharing

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them.

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from GitHub and many other sources.

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

4.17. Quantized Model Training Methods:

There were 2 specific training models for quantized TFLite model:

1. For Raspberry pi 3/4 (Faster but less accurate)(4-10 FPS depending on raspberry pi)
2. ONLY For Raspberry pi 4 (slower but more accurate)(1-2 FPS)

We've trained both of them, but the first method was better in accuracy and detection and actually it could be quantized unlike the second slower method.

4.18. Quantized Model Training:

Quantized Model Training will be the same as the non-quantized with just adding some parameters to the TFLite conversion code to be like that:

- tflite_convert --graph_def_file=tflite_graph.pb --output_file=detect.tflite --input_shapes=1,300,300,3 --input_arrays=normalized_input_image_tensor --output_arrays=TFLite_Detection_PostProcess,TFLite_Detection_PostProcess:1,TFLite_Detection_PostProcess:2,TFLite_Detection_PostProcess:3 --inference_type=QUANTIZED_UINT8 --mean_values=128 --std_dev_values=128 --change_concat_input_ranges=false --allow_custom_ops

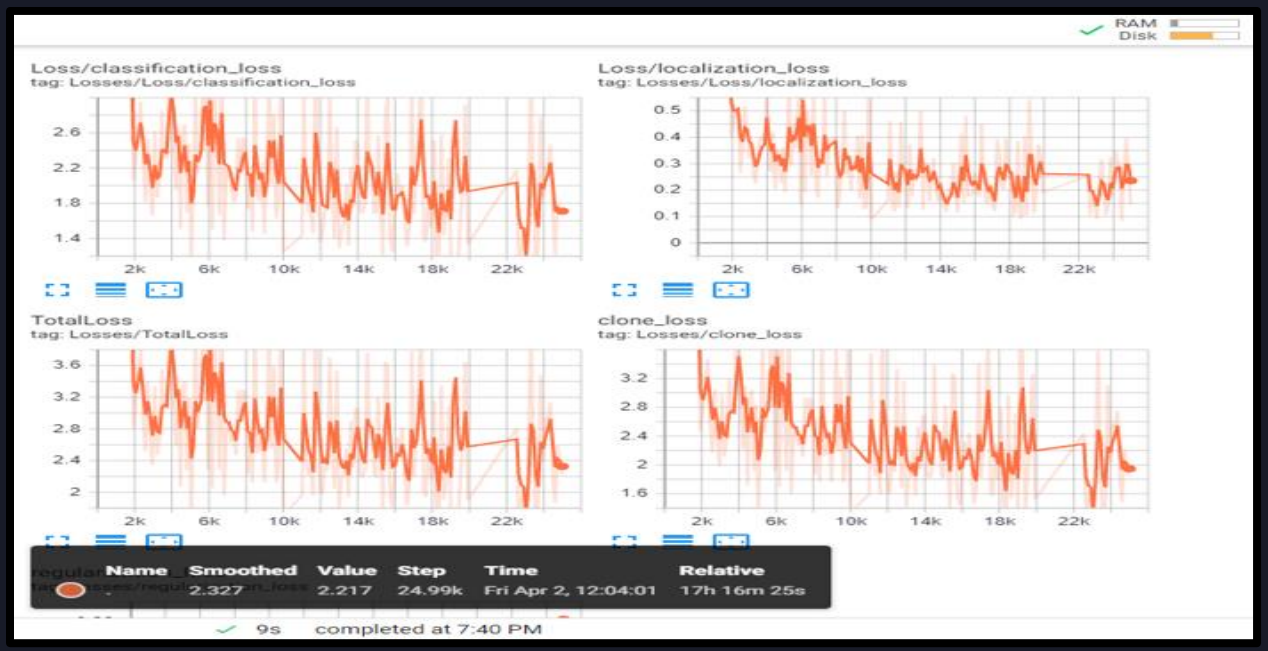
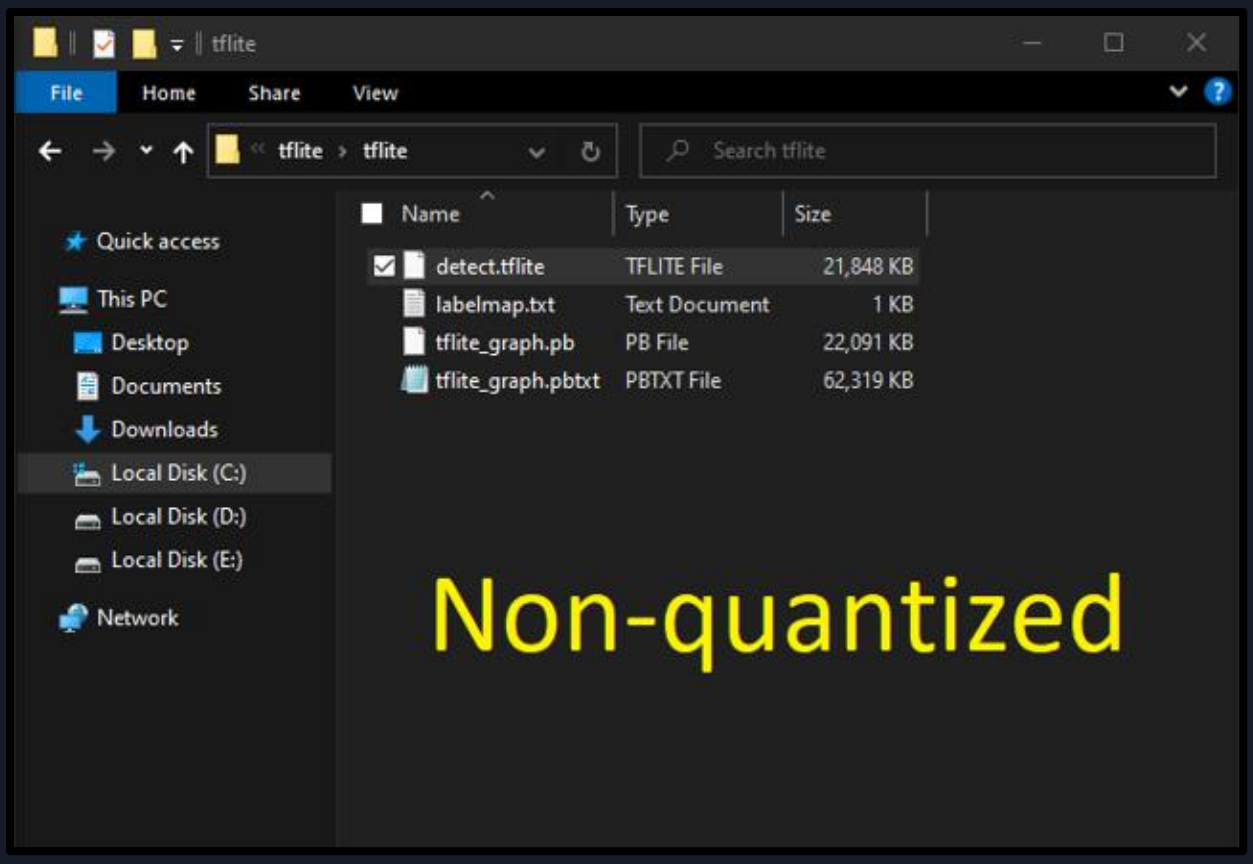


Figure 85: Quantization training Tensorboard

4.19. Quantized vs Non-quantized:



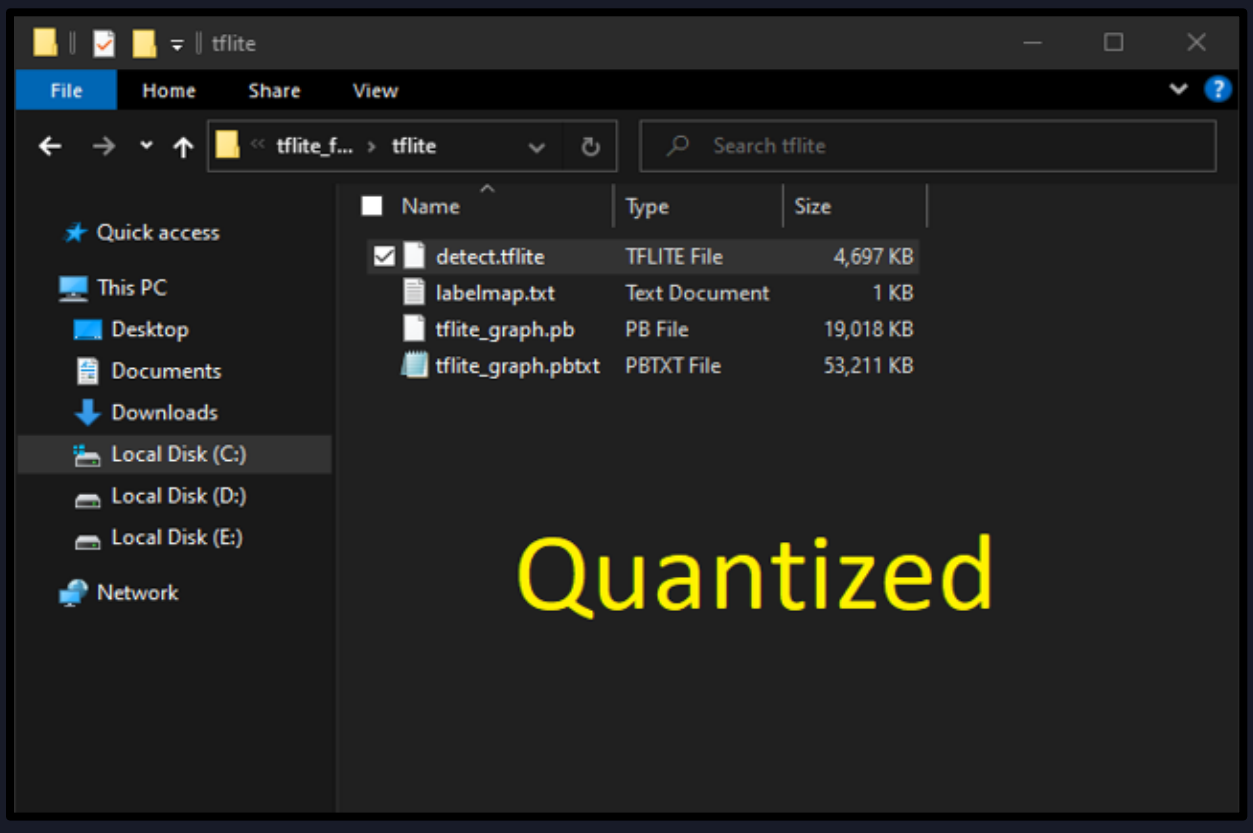


Figure 86: Quantized vs Non-Quantized model

4.20. Running this TFLite model:

Now, after we've finished and converted our model into a quantized model, we can then test our quantized model. We can use these devices for testing:

1. PC (Windows) with TensorFlow installed on it. Testing using a given photo, video, or a live-streaming webcam (if available), but it won't show you the FPS and won't know how this model is improved.

- Steps for testing will be the same as the normal-float TFLite used before.

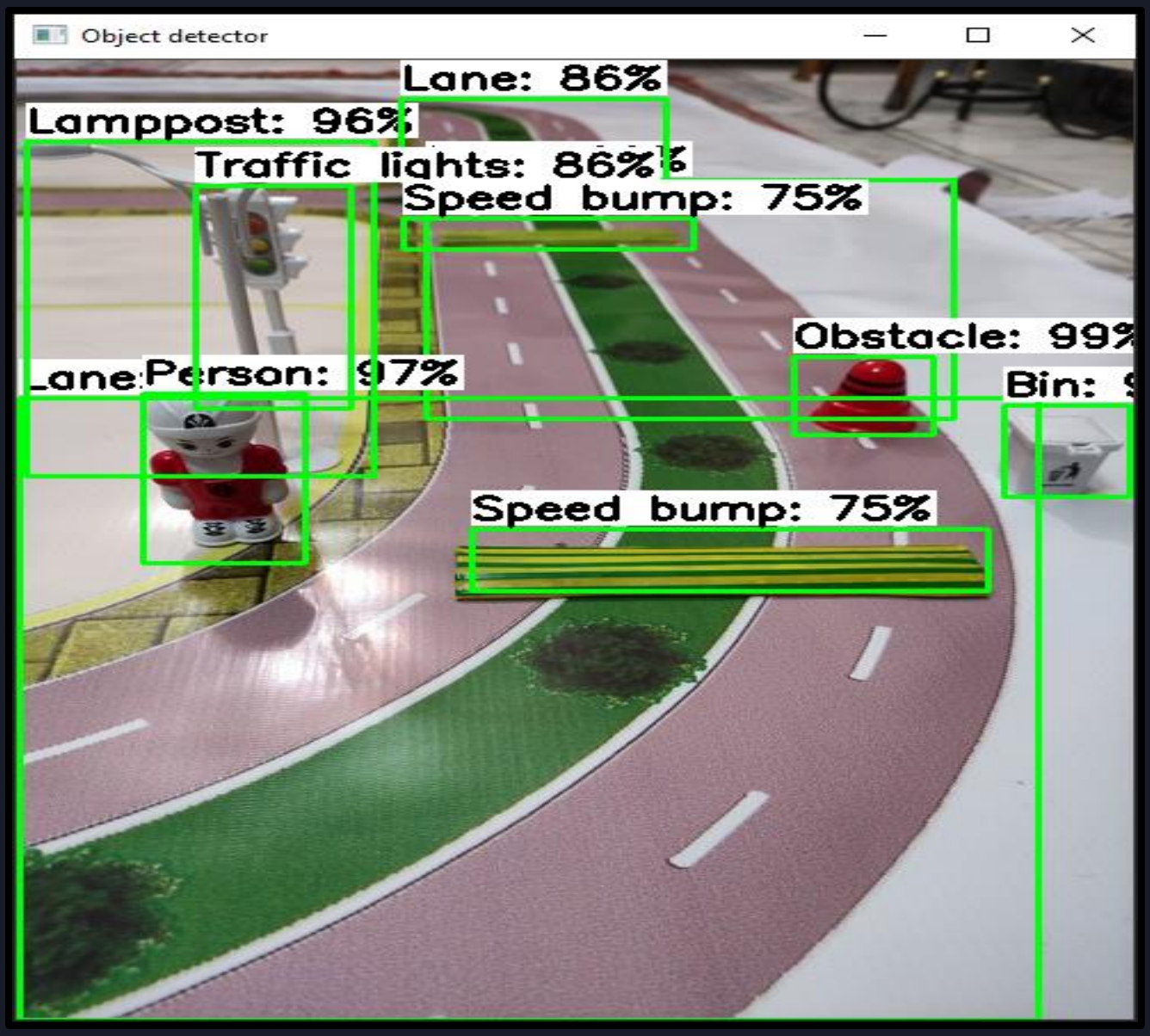


Figure 87: Running TFLite model on PC

4.21. Raspberry pi, which we will be using for our project.

Here, we can see the difference in FPS between the two TFLite models: the quantized & non-quantized one.

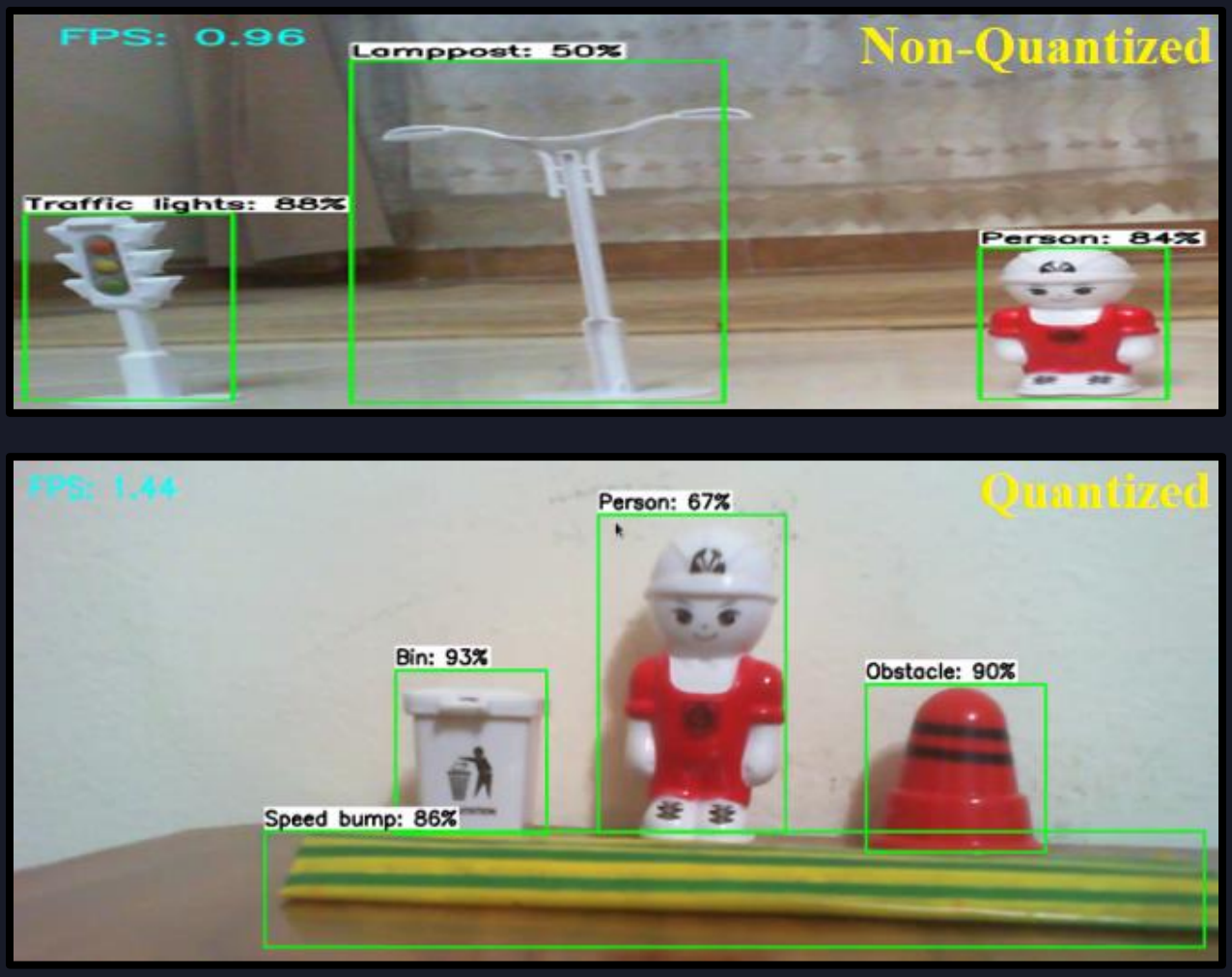


Figure 88: Running TFLite model on the Raspberry pi

4.22. . Android (FPS depends on "how modern is your android device ?")

For a Galaxy J1 Ace with android 5.1.1, we could get an FPS of about 2.5. We tested it also on other modern android devices, where the FPS got up to 20 !!

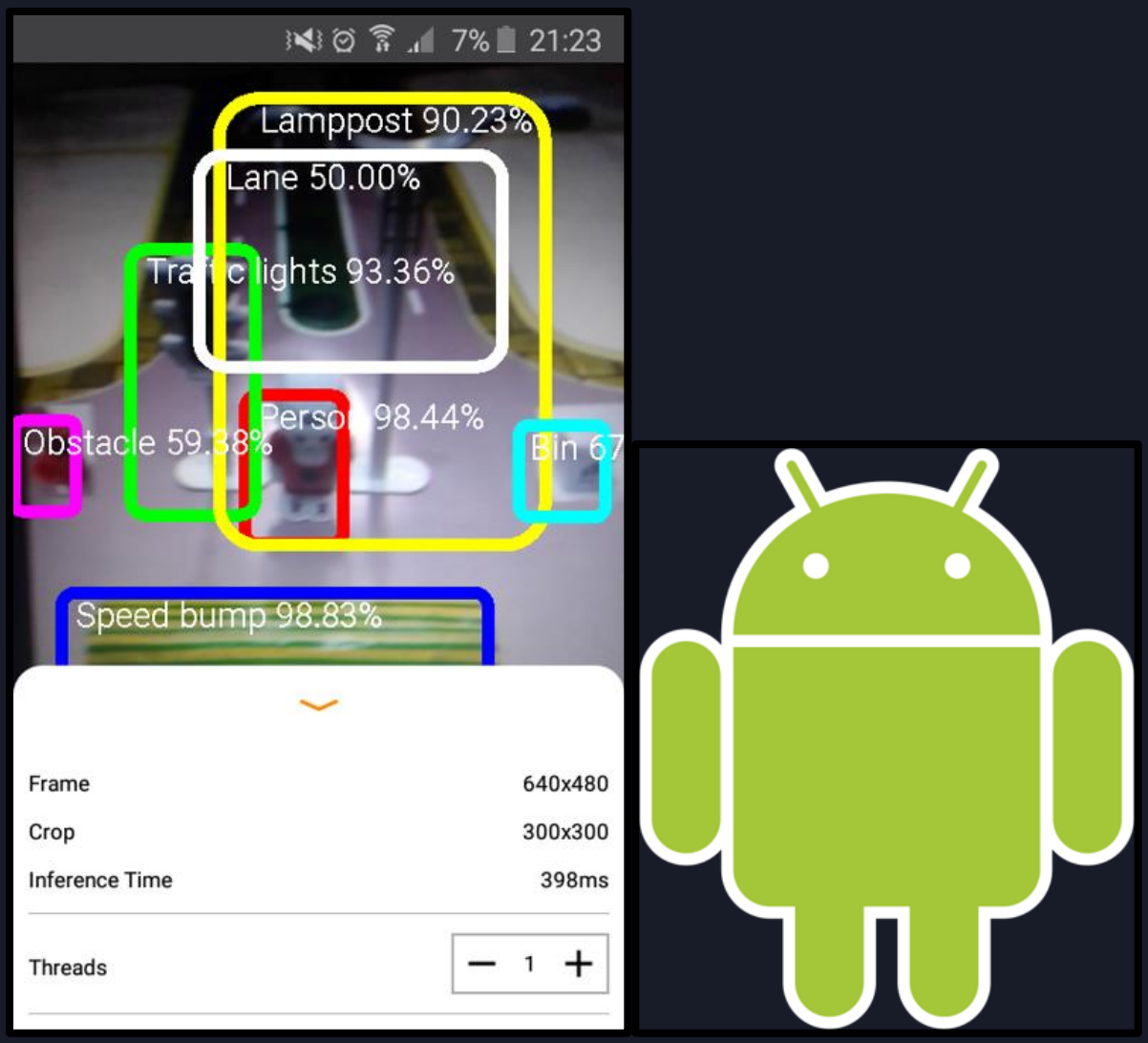


Figure 89: Runnig TFLite model on an Android device

However, we didn't use the android method as it will be difficult to communicate with other project components (android is not our profession, here).

4.23. How does the raspberry pi send the detected data?

- Sending data between raspberry pi and arduino via Serial communication
- We'll quickly explain what Serial communication is. Then we'll see how to setup hardware and software, and we'll dive into the Python code for raspberry pi and arduino code.
- Serial communication is simply a way to transfer data. The data will be sent sequentially, one bit at a time (1 byte = 8 bits), contrary to parallel communication, where many bits are sent at the same time.

4.24. Hardware setup for Serial communication

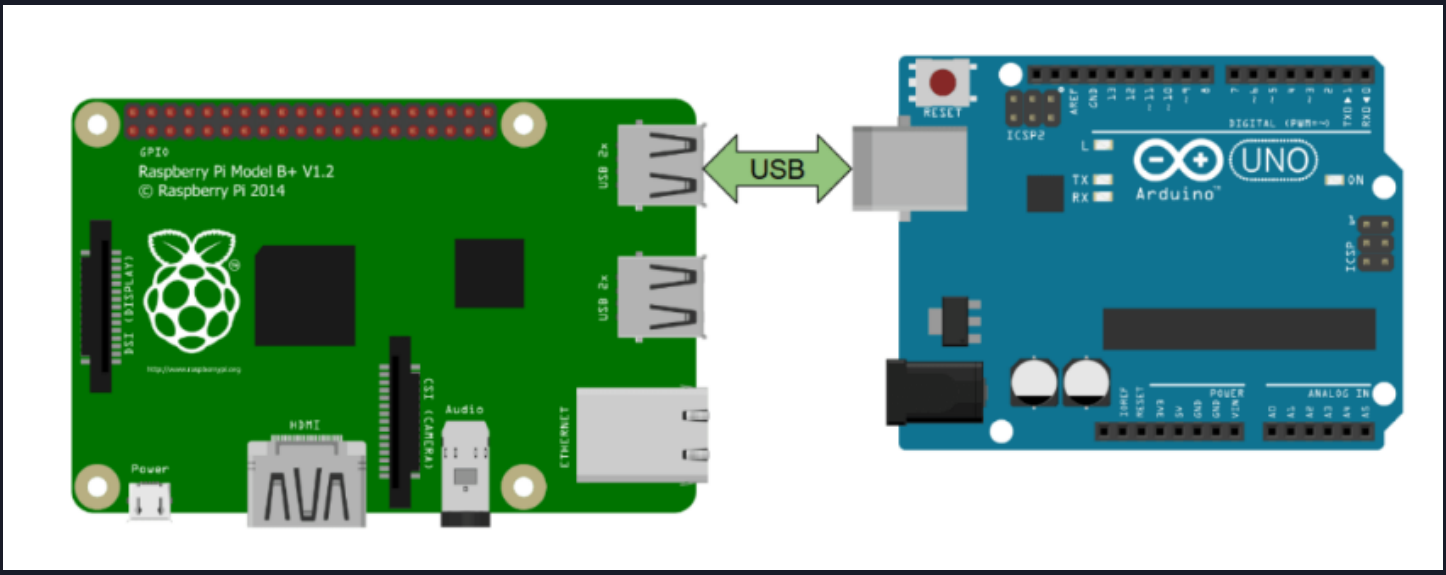


Figure 90: Hardware Setup for Serial Communication

There are more than one way to connect your Raspberry Pi and Arduino for Serial communication but we'll use USB cable.

• Serial via USB

The easiest way is to use a USB cable between both board.
On the Raspberry Pi side, a simple USB connector is all you need. You can choose any of the 4 USB ports available on the board.
For Arduino, you will use the USB port that you use to upload code from your computer (with the Arduino IDE) to your board. Here the USB connector will depend on which version you have.

• Detect the Arduino board

When connecting the Arduino with a USB cable, you should see it appear as /dev/ttyACM0, or /dev/ttyUSB0 (sometimes the number can be different, for example /dev/ttyACM1).

Install Python Serial library on Raspberry Pi

You need to install a library to be able to use the Serial interface with Python.

```
python3 -m pip install pyserial
```

This Python library is well-known and used in a lot of applications.

4.25. Raspberry Pi Python flow chart and code

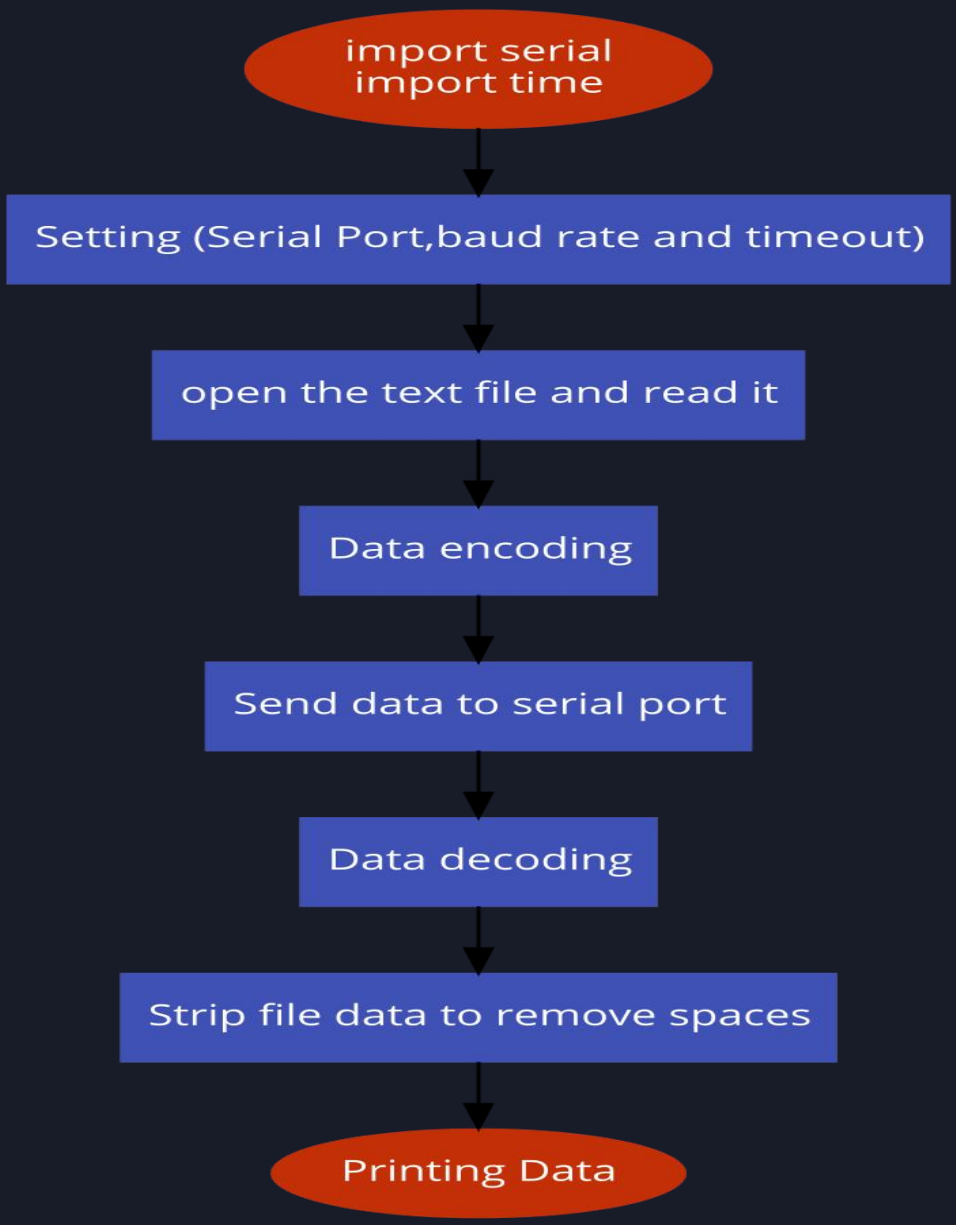


Figure 91: Raspberry Pi Python flow chart

```
import serial
import time
if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
    ser.flush()


    while True:
        f = open('/home/pi/Desktop/labelmap.txt','r')
        data = f.read()
        f.close()
        data_encode = data.encode()
        ser.write(data_encode)
        line = ser.readline().decode('utf-8').rstrip()
        print(line)
        time.sleep(1)
```

Figure 92: Serial Communication code

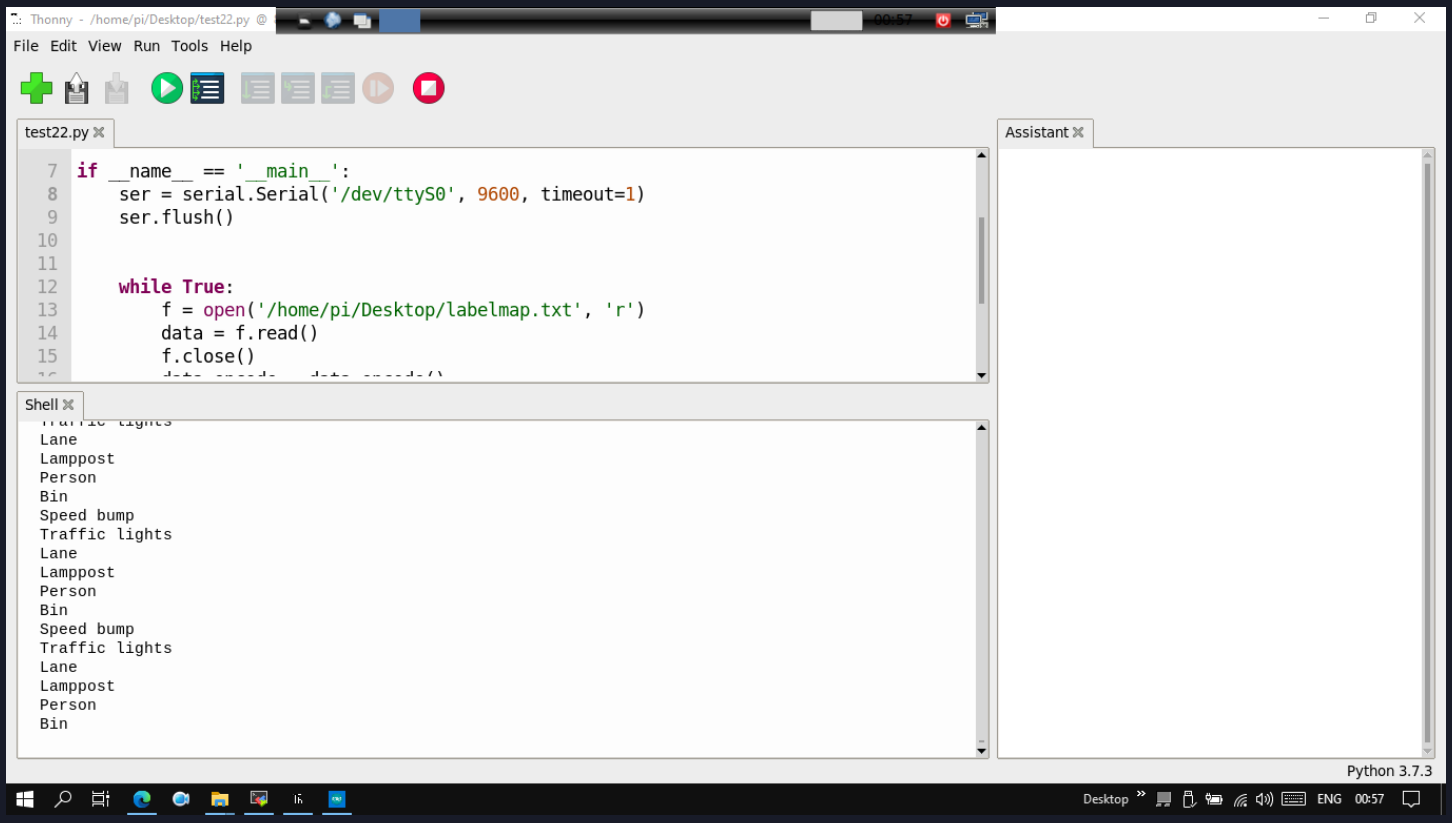


Figure 93:Sending data Serially

4.26. Arduino code and flow chart

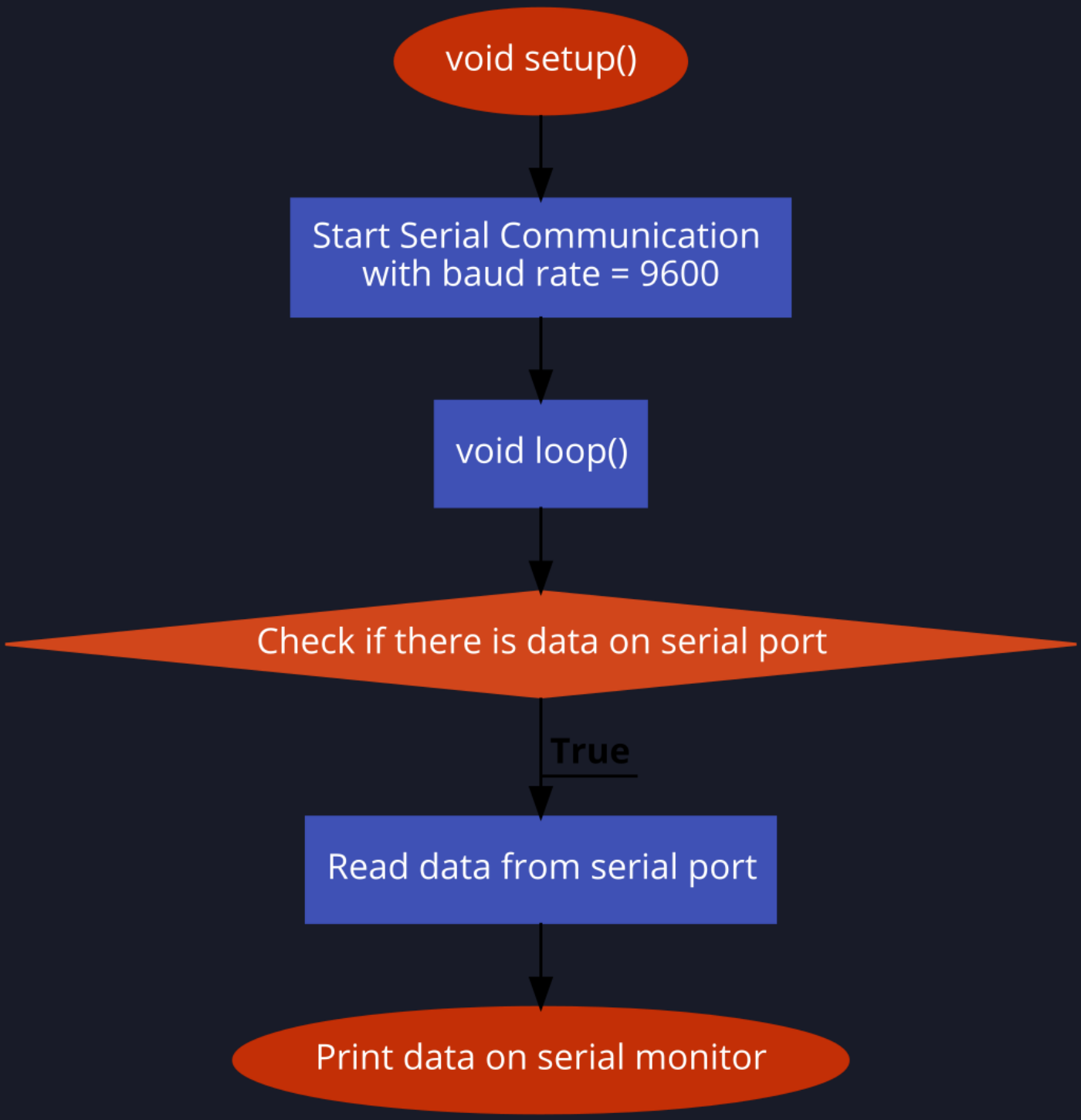


Figure 94: Arduino receiving data flowchart

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    String data = Serial.readStringUntil('\n');
    Serial.println(data);
  }
}
```

Figure 95: Arduino UART configuration

Output on arduino's serial monitor

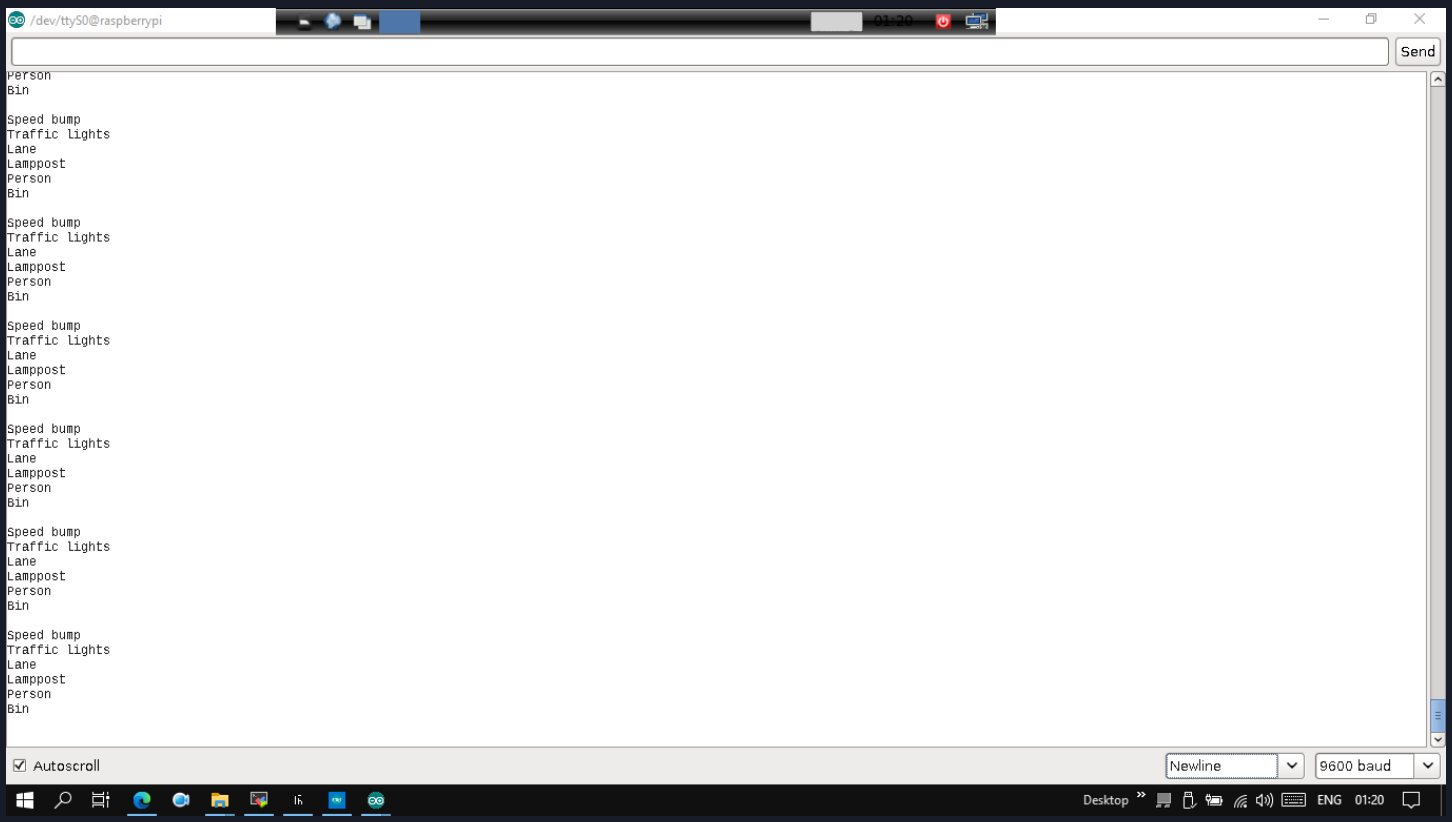


Figure 96:: Arduino serial Monitor

5

Embedded System

5.1. UART: Receiving Data With TivaC Receiving Code:

5.1.1. Function declaration and Initialization.

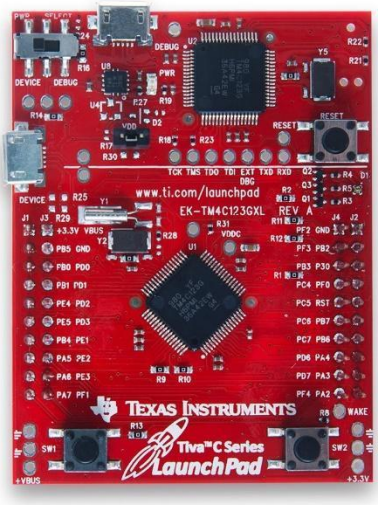


Figure 97: TM4C123GH6PM

```c
/*
 * main.c
 *
 *
 * This code reads data from RX5 pin whenever data is available
 * and echo back data through TX5 pin.
 *
 */
#include "inc/tm4c123gh6pm.h"
#include <stdint.h>
#include <stdlib.h>
void Delay(unsigned long counter);
char UART5_Receiver(void);
void UART5_Transmitter(unsigned char data);
void printstring(char *str);

int main(void)
{
    SYSCTL_RCGCUART_R |= 0x20;  /* enable clock to UART5 */
    SYSCTL_RCGCGPIO_R |= 0x10;  /* enable clock to PORTE for PE4/Rx and RE5/Tx */
    Delay(1);
    /* UART0 initialization */
    UART5_CTL_R = 0;         /* UART5 module disbable */
    UART5_IBRD_R = 104;      /* for 9600 baud rate, integer = 104 */
    UART5_FBRD_R  = 11;       /* for 9600 baud rate, fractional = 11*/
    UART5_CC_R  = 0;          /*select system clock*/
    UART5_LCRH_R = 0x60;     /* data lenght 8-bit, not parity bit, no FIFO */
    UART5_CTL_R = 0x301;     /* Enable UART5 module, Rx and Tx */


    /* UART5 TX5 and RX5 use PE4 and PE5. Configure them digital and enable alternate function */
    GPIO_PORTE_DEN_R = 0x30;      /* set PE4 and PE5 as digital */
    GPIO_PORTE_AFSEL_R = 0x30;    /* Use PE4,PE5 alternate function */
    GPIO_PORTE_AMSEL_R = 0;    /* Turn off analog function*/
    GPIO_PORTE_PCTL_R = 0x00110000;     /* configure PE4 and PE5 for UART */

```

Figure 98: function declaration and Initialization

5.1.2. Main Application.

```
      Delay(1);
     printstring("Hello World \n");
    Delay(10);
    while(1)
    {
       char c = UART5_Receiver();          /* get a character from UART5 */
       UART5_Transmitter(c);
    }
}
```

Figure 99: TivaC UARt Main App

5.1.3. Function Definition.

1. Transmitting Function

```
56 void UART5_Transmitter(unsigned char data)
57 {
58     while((UART5_FR_R & (1<<5)) != 0); /* wait until Tx buffer not full */
59     UART5_DR_R = data;                  /* before giving it another byte */
60 }
```

Figure 100: Transmitting Function

2. Receiving Function

```
48 char UART5_Receiver(void)
49 {
50     char data;
51       while((UART5_FR_R & (1<<4)) != 0); /* wait until Rx buffer is not full */
52     data = UART5_DR_R ;      /* before giving it another byte */
53     return (unsigned char) data;
54 }
```

Figure 101: Receiving Function

3. Printstring Function

```
62 void printstring(char *str)
63 {
64   while(*str)
65     {
66         UART5_Transmitter(*(str++));
67     }
68 }
69
```

Figure 102: printstring function

4. System Initialization Function

```
void SystemInit(void)
{

    /* Grant coprocessor access */
    /* This is required since TM4C123G has a floating point coprocessor */
    NVIC_CPAC_R |= 0x00F00000;
}
```

Figure 103: System initialization function

5. Delay Function

```
70 void Delay(unsigned long counter)
71 {
72     unsigned long i = 0;
73
74     for(i=0; i< counter; i++);
75 }
```

Figure 104: Delay Function

5.1.4. UART with LCD Flow Chart:

Figure 105: UART with LCD Flow Chart

5.2. SPEED SENSOR

Since v2v technology needs complete information about all the vehicles committed to it, it was necessary to calculate the speed of cars so that relative speed between vehicles on the road can be calculated and decisions can be taken according to that

5.2.1 Speed sensor work Theory:

LM 393 speed sensor is a module that is used to calculate the rotational speed of the rotating motors as follows:

Figure() shows the LM393 module which consists of two parts: the sensor part and the control part.

Sensor Part:

an Infrared LED and an NPN Photo Transistor. These two components are placed facing each other is a special housing made of black thermoplastic. This special housing ensures that the Photo Transistor receives light only from the Infrared LED and all the external source of light is eliminated.

Figure 106: Speed Sensor

Control Part:

LM393 Voltage Comparator and a few passive electronic components. The signal from the photo transistor is given to the LM393 and based on the presence or absence of an object between the Infrared LED and the Photo Transistor, the Output of the LM393 IC will either be HIGH or LOW. So in a nutshell it's the IR LED always on and passes an IR ray if it meets the motor's wheel slit so IR passes to the NPN photo transistor so the voltage comparator results HIGH, and if the IR ray meets blank areas between motor's wheel slits it blocks the ray so it doesn't reach the photo transistor so the voltage comparator results LOW.

5.2.2. Getting the speed sensor signal

The digital out put of the sensor is taken by the pinf2 from the tiva c as a gpio initialization is made to port f by the following code ..

5.2.3. Setting the data input pin for the tiva c :

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); // Enable the Gpio  for port f  by enaboling the crystal on the port
while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF)) //wait for port to be intalized
  {
  }
GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_3); //set pin pf3 input
```

```
GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_PIN_3, GPIO_RISING_EDGE); //set pinf3 inputas a  rising edge
GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_3); //Enable the pin interrupts
GPIOIntRegister(GPIO_PORTF_BASE, PortFIntHandler); //hand innterupt for portf
```

5.2.4. Interrupt handler for port f :

It is the function that initializes when data access port f

```
void PortFIntHandler(void)  //gpio innterupt handler
{
        count++;
   GPIOIntClear(GPIO_PORTF_BASE, GPIO_INT_PIN_3); // clear interrupt flag
}
```

5.2.5. Setting the timer:

We used timer 2 so we can divide the time into intervals so we can get represent the time for the velosity

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER2);// enable timer 2
while(!SysCtlPeripheralReady(SYSCTL_PERIPH_TIMER2))// Wait for the Timer2 module to be ready.
{
}
TimerConfigure(TIMER2_BASE,  TIMER_CFG_PERIODIC); // confiure the type of the timer to repeat every constant
period

TimerLoadSet(TIMER2_BASE, TIMER_A, period-1);
// Setup the interrupts for the timer timeouts.
TimerIntEnable(TIMER2_BASE, TIMER_TIMA_TIMEOUT);

// Enable the timers.
TimerEnable(TIMER2_BASE, TIMER_A);

TimerIntRegister(TIMER2_BASE,TIMER_A,Timer2IntHandler);
```

5.2.6. Setting the interrupt handler :

This is the function that will initialize when the timer reaches its over flow state to rise the timer flag

```
void Timer2IntHandler(void) //timer interupt handler
{
time=1;
// Clear the timer interrupt.
```

```
TimerIntClear(TIMER2_BASE, TIMER_TIMA_TIMEOUT);
}
```

5.2.7. Calculating the speed

To calculate the speed we need distance and time: distance is obtained by the perimeter of the motor's wheel and time is obtained by a timer handler, then all the information is ready now to calculate speed as follows:

speed = $\frac{distance}{time} = \frac{2\pi r * counts}{no.of\ slits * time} = \frac{2 * 3.14 * 1.3 * counts}{20*8}$

```
while(1){
        if (time==1)
        {
        GPIOIntDisable(GPIO_PORTF_BASE, GPIO_INT_PIN_3); // will distable the port to stop  the counting
        v = (int )pi*d*count*T/pulsesperturn/Ts; // will calculate the speed of the motor in integer
        sprintf(v_string , "%d", v); //will change the speed from integer state to char state to be displayed in the lcd
        count = 0;
        time = 0;
        GPIOIntEnable(GPIO_PORTF_BASE, GPIO_INT_PIN_3); // enable the port to receive the counts once again
        }
        }
```

5.2.7. Algorithm flow chart

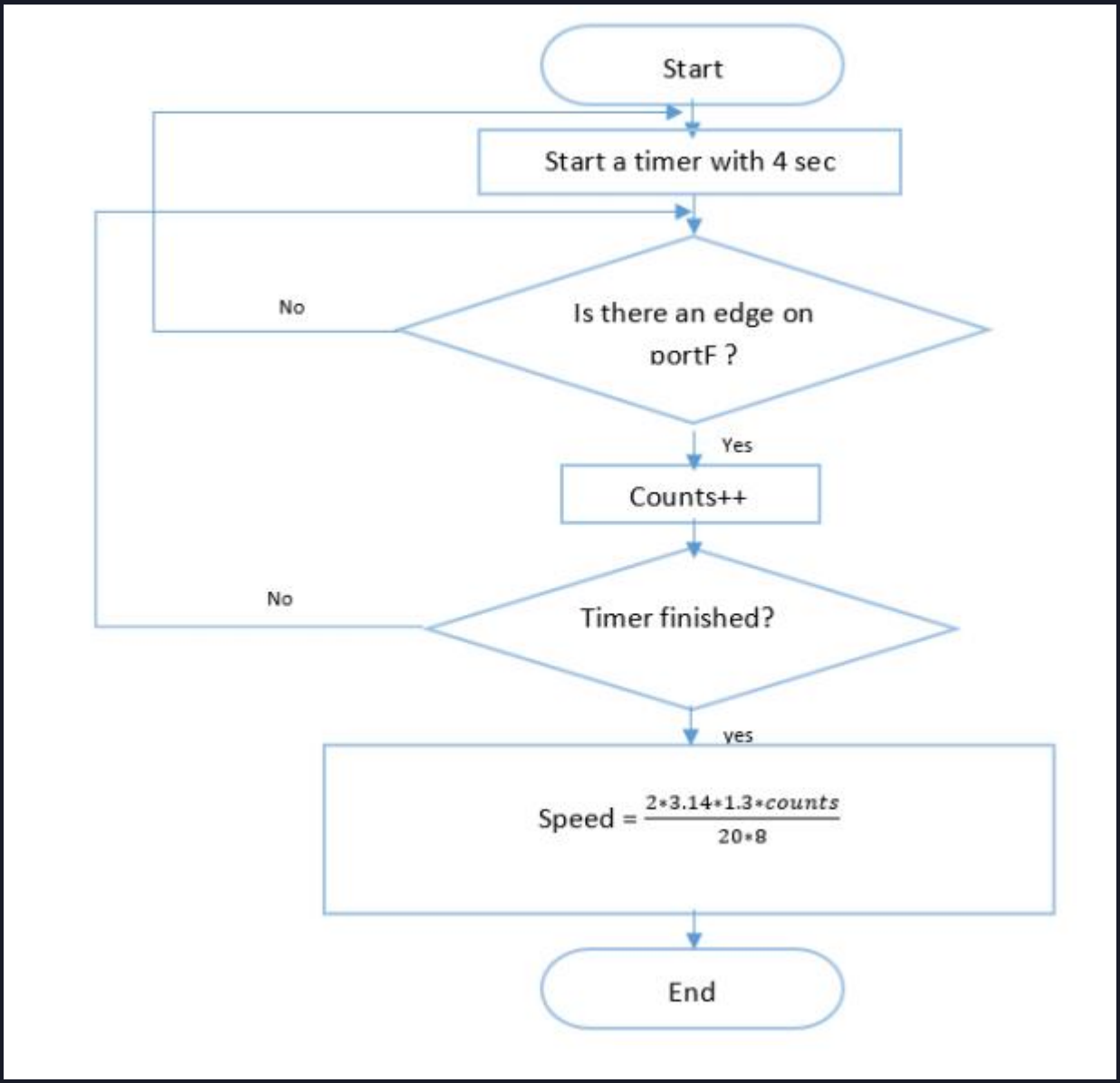
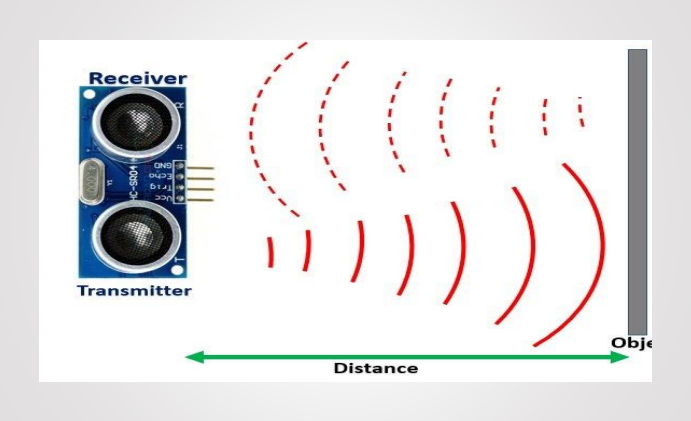


Figure 107: Speed Sensor Flowchart

5.3. ULTRASONIC SENSOR

5.3. 1 Ultrasonic theory of work



An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e., the sound that humans can hear).

The ultrasonic sensor works on the principle of SONAR system which is used to determine the distance to an object. SONAR basically stands for Sound Navigation and Ranging. An ultrasonic sensor generates the high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver. By measuring the time required for the echo to reach to the receiver, we can calculate the distance.

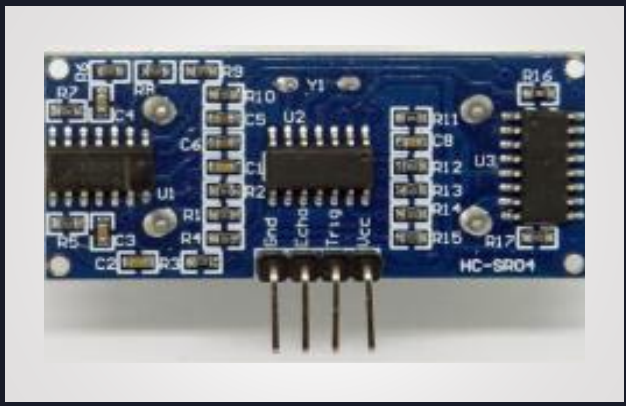


Figure 108: Ultrasonic Working Principle

5.3.2. HC-SR04 Ultrasonic module

HC-SR04 has an ultrasonic transmitter, receiver and control circuit.

Microcontroller U1

The heart of the unit is the EM78P153 8-bit microprocessor.

This handle:

- Interface to Trigger and Echo pins.
- Timing and sending antiphase burst for ping to send.
- Squelch control, where by during Ultrasonic transmit, a threshold for the incoming receiver is effectively disabled to avoid bogus echoes. This is important as whilst sending vibrations from the TX transducer will be received through PCB and by air between the transducers on the RX transducer.

- Receiving the processed signal from the Receiver section as an interrupt (Rising edge), this is actually a filtered and much amplified version of all the echoes received.

Transmitter U3

- Voltage drive to TX transducer from the antiphase TX signals from the micro (U1). By using antiphase signals, a differential voltage can be driven across the transducer effectively +/-5V across the transducer.
- The other part is two transistors with a common base pin, collectors available on other pins. This transistor forms part of the feedback loop on the final part of the receiver chain, to change an analog signal into a TTL type digital signal.

Receiver U2

The quad op-amp IC (U2) is a LM324 is 1 MHz unity gain bandwidth device, with limited range I/O. It is a ubiquitous and cheap device. Considering some of the gain levels used in the stages means that 40 kHz signals are passing through stages with around 100 kHz bandwidth.

HC-SR04 works at 40KHz Frequency which lies in ultrasound range, above 20 KHz. Working voltage is 5V DC. As the current drawn by the sensor is less than 15mA, so won't be affecting the current ratings of the controller. No need for external buffers.

5.3.3. Working Method

We need to transmit trigger pulse of at least 10 us to the HC-SR04 Trig Pin. 10µs is enough period for the controller, after which it starts to transmit ultrasonic signal. Then the HC-SR04 automatically sends Eight 40 kHz sound wave pulses and wait for rising edge output at Echo pin. As the number of pulses, the amplitude of the received signal increases and saturates at a point. After further increase in number of pulses results in increase in reflection peak, which are not required. When the rising edge capture occurs at Echo pin, start the Timer and wait for falling edge on Echo pin. As soon as the falling edge is captured at the Echo pin, read the count of the Timer. This time count is the time required by the sensor to detect an object and return back from an object.

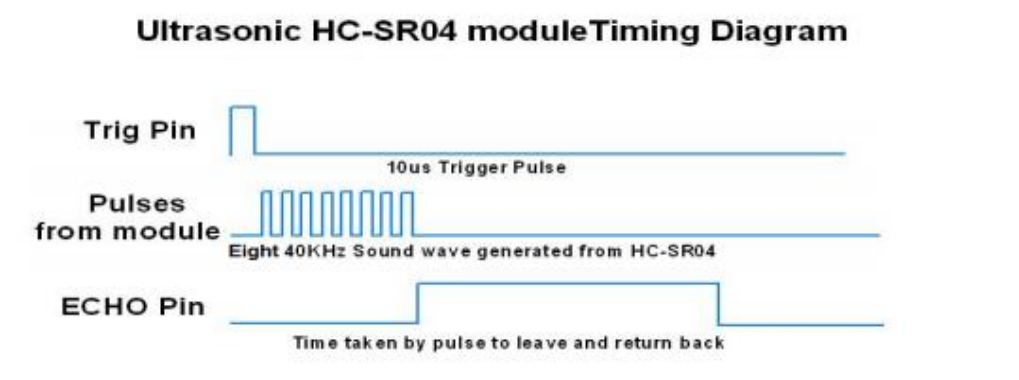


Figure 109: Ultrasonic Timing Diagram

Conversion from duration to distance can be done using the following formula:

Distance = (Travel time/2) x speed of sound

We need to divide the travel time by 2 because we have to take into account that the wave was sent, hit the object, and then returned back to the sensor.

If the HC-SR04 does not receive an echo then the output never goes low. Accordingly, some sensors timeout from 28ms to 36ms.

5.3.4. Contribution

In order to measure distance, we started by choosing two GPIO Pins. One for sending trigger and other for receiving echo. Both pins are either high or low, so it was better to be used in digital mode.

We start by sending trigger to the module by setting the Trigger pin high for 10 us, then low. After the trigger, the module starts transmitting ultrasound wave and Echo pin in set high until the wave hits and object and return back to the module.

In order to check whether the Echo pin is high or low it was configured to receive interrupts at both rising and falling edges. Using polling wasn't the best solution as it is time consuming and delays other tasks.

Once the Echo pin is high, we receive a rising edge interrupt. At this instance, we reset the timer to measure the duration in which Echo pin is high.

When the Echo pin returns low, we receive a falling edge interrupt, disable the timer and calculate the distance.

Since the operating frequency of controller is 80 MHz, so by diving number of ticks by 80 we get the duration at which Echo pin is high in µs. Assuming the speed of transmitted ultrasound

wave is approximately 343 m/s, so 343m/s = 0.0343 cm/µs = 1/29.1 cm/µs. Then we divide by 2 to get the required distance in cm.

5.3.5. Code

```
u32 Measure_Distance(void)
{
    Set_pin_value (PORTA , TRIGGER ,GPIO_LOW);    /*Set Trigger pin value to zero*/
    delay_Microsecond(10);                        /*Make a delay for 10 micro second*/
    Set_pin_value (PORTA , TRIGGER ,GPIO_HIGH);   /*Set Trigger pin value to one*/
    delay_Microsecond(10);                        /*Make a delay for 10 micro second*/
    Set_pin_value (PORTA , TRIGGER ,GPIO_LOW);    /*Set Trigger pin value to zero*/

    // Capture firstEgde i.e. rising edge
    timer0_GPTMICR =4;                            /*clear the flag*/
    while((timer0_GPTMRIS & 4)==0){};             /* wait till the flag will be 1 --> it means that the edge has come*/
    First_edge = timer0_GPTMTAR;                  /*store the time of the rising edge*/

    // Capture secondEdge i.e. falling edge
    timer0_GPTMICR =4 ;                          /*clear timer capture flag*/
    while((timer0_GPTMRIS & 4) ==0){};           /*wait till the flag will be 1 --> it means that the edge has come*/
    Second_edge = timer0_GPTMTAR;                /*store the time of the Falling edge*/

    /*calculate the Distance*/
    TIME = Second_edge - First_edge;
    Distance = (_16MHz_1clock )*(double) (VOICE_SPEED_CM/2) *(double)TIME;
    return Distance;
}
```

Figure 110: Ultrasonic Code

5.3.6. Ultrasonic Flowchart

5.4. GPS Tracking

5.4.1. GPS Tracking concept

What is GPS and how does it work?

The Global Positioning System (GPS) is a navigation system using satellites, a receiver and algorithms to synchronize location, velocity and time data for air, sea and land travel.

The satellite system consists of a constellation of 24 satellites in six Earth-centered orbital planes, each with four satellites, orbiting at 13,000 miles (20,000 km) above Earth and traveling at a speed of 8,700 mph (14,000 km/h).

While we only need three satellites to produce a location on earth's surface, a fourth satellite is often used to validate the information from the other three. The fourth satellite also moves us into the third-dimension and allows us to calculate the altitude of a device.

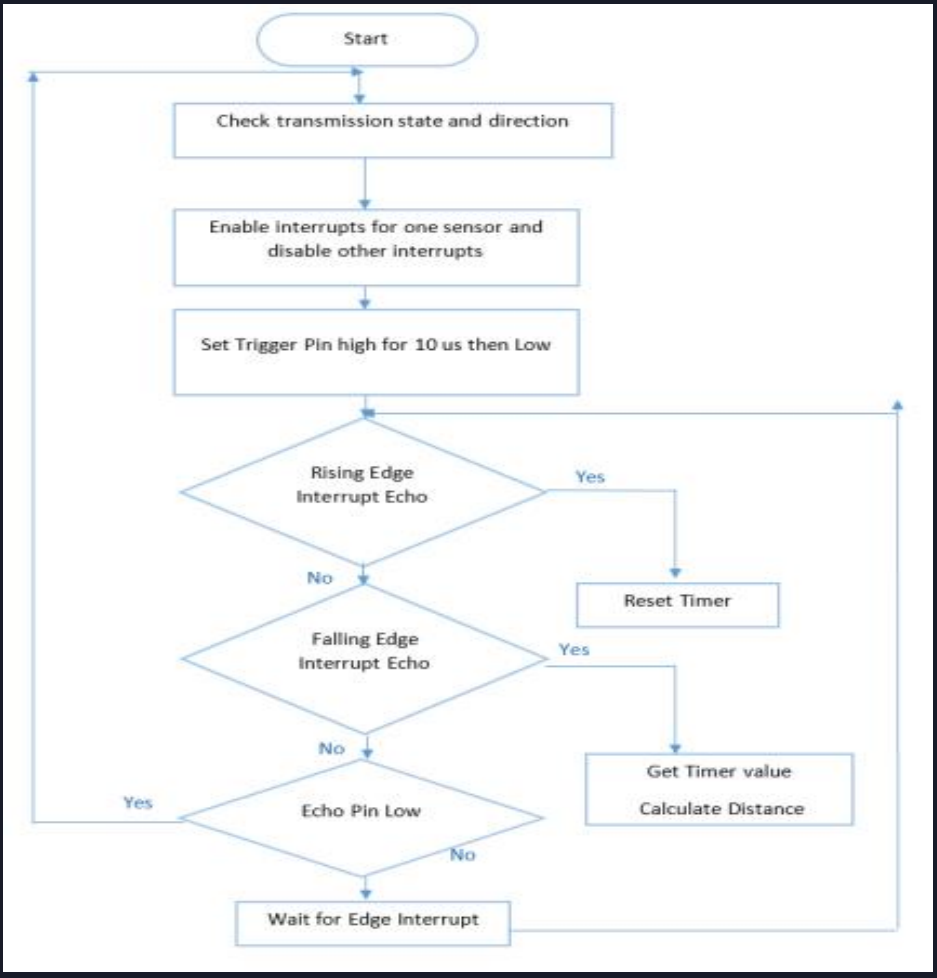


Figure 111: GPS Flowchart

What are the three elements of GPS?

GPS is made up of three different components, called segments, that work together to provide location information.

The three segments of GPS are:

- Space (Satellites) — The satellites circling the Earth, transmitting signals to users on geographical position and time of day.
- Ground control — The Control Segment is made up of Earth-based monitor stations, master control stations and ground antenna. Control activities include tracking and operating the satellites in space and monitoring transmissions. There are monitoring stations on almost every continent in the world, including North and South America, Africa, Europe, Asia and Australia.
- User equipment — GPS receivers and transmitters including items like watches, smartphones and telematic devices.

How does GPS technology work?

GPS works through a technique called trilateration. Used to calculate location, velocity and elevation, trilateration collects signals from satellites to output location information. It is often mistaken for triangulation, which is used to measure angles, not distances.

Satellites orbiting the earth send signals to be read and interpreted by a GPS device, situated on or near the earth's surface. To calculate location, a GPS device must be able to read the signal from at least four satellites.

Each satellite in the network circles the earth twice a day, and each satellite sends a unique signal, orbital parameters and time. At any given moment, a GPS device can read the signals from six or more satellites.

A single satellite broadcasts a microwave signal which is picked up by a GPS device and used to calculate the distance from the GPS device to the satellite. Since a GPS device only gives information about the distance from a satellite, a single satellite cannot provide much location information. Satellites do not give off information about angles, so the location of a GPS device could be anywhere on a sphere's surface area.

When a satellite sends a signal, it creates a circle with a radius measured from the GPS device to the satellite.

When we add a second satellite, it creates a second circle, and the location is narrowed down to one of two points where the circles intersect.

With a third satellite, the device's location can finally be determined, as the device is at the intersection of all three circles.

That said, we live in a three-dimensional world, which means that each satellite produces a sphere, not a circle. The intersection of three spheres produces two points of intersection, so the point nearest Earth is chosen.

Here is an illustration of satellite ranging:

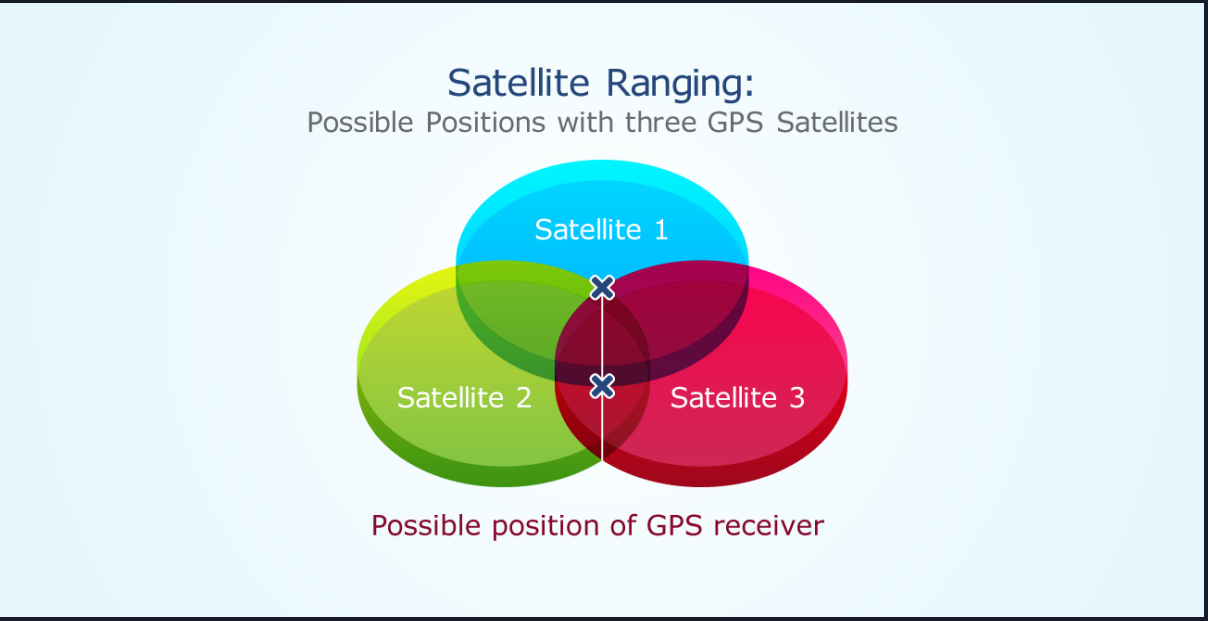


Figure 112: Satellite Ranging

5.4.2. GPS module

➢ Ublox NEO-6m GPS Module

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints. The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments.

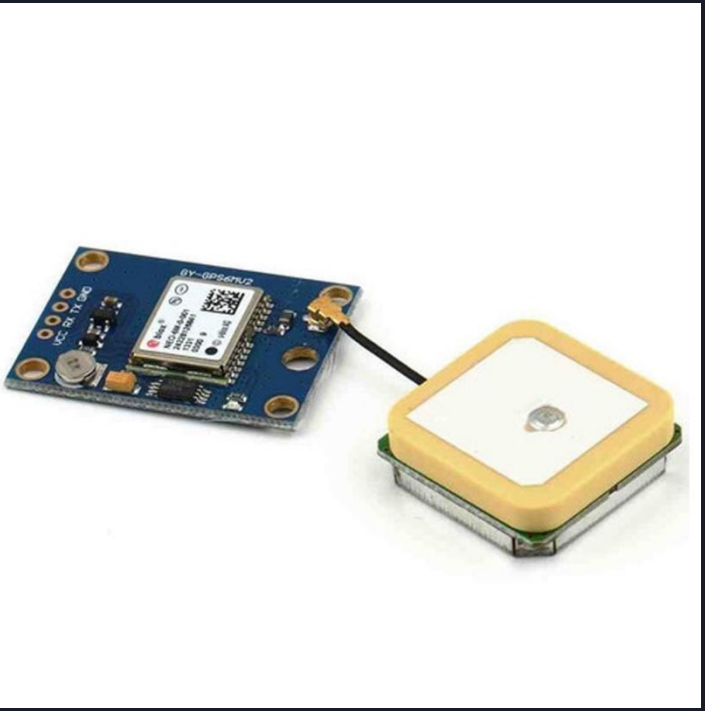


Figure 113: GPS Module

5.4.3. Specifications

Parameter	Specification			
Receiver type	50 Channels GPS L1 frequency, C/A Code SBAS: WAAS, EGNOS, MSAS			
Time-To-First-Fix[1]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Cold Start[2]	26 s	27 s	32 s
	Warm Start[2]	26 s	27 s	32 s
	Hot Start[2]	1 s	1 s	1 s
	Aided Starts[3]	1 s	<3 s	<3 s
Sensitivity[4]		NEO-6G/Q/T	NEO-6M/V	NEO-6P
	Tracking & Navigation	-162 dBm	-161 dBm	-160 dBm
	Reacquisition[5]	-160 dBm	-160 dBm	-160 dBm
	Cold Start (without aiding)	-148 dBm	-147 dBm	-146 dBm
	Hot Start	-157 dBm	-156 dBm	-155 dBm
Maximum Navigation update rate		NEO-6G/Q/M/T	NEO-6P/V	
		5Hz	1 Hz	
Horizontal position accuracy[6]	GPS	2.5 m		
	SBAS	2.0 m		
	SBAS + PPP[7]	< 1 m (2D, R50)[8]		
	SBAS + PPP[7]	< 2 m (3D, R50)[8]		
Configurable Timepulse frequency range		NEO-6G/Q/M/P/V	NEO-6T	
		0.25 Hz to 1 kHz	0.25 Hz to 10 MHz	
Accuracy for Timepulse signal	RMS	30 ns		
	99%	<60 ns		
	Granularity	21 ns		
	Compensated[9]	15 ns		
Velocity accuracy[6]		0.1m/s		
Heading accuracy[6]		0.5 degrees		
Operational Limits	Dynamics	≤ 4 g		
	Altitude[10]	50,000 m		
	Velocity[10]	500 m/s		

Figure 114: GPS module specs

5.4.4. Procedure

1.Interpreted sentences

The satellite sends parses of data(interpreted sentences) that holds information according to use as the following:

$GPBOD - Bearing, origin to destination
$GPBWC - Bearing and distance to waypoint, great circle
$GPGGA - Global Positioning System Fix Data
$GPGLL - Geographic position, latitude / longitude
$GPGSA - GPS DOP and active satellites
$GPGSV - GPS Satellites in view
$GPHDT - Heading, True
$GPR00 - List of waypoints in currently active route
$GPRMA - Recommended minimum specific Loran-C data
$GPRMB - Recommended minimum navigation info
$GPRMC - Recommended minimum specific GPS/Transit data
$GPRTE - Routes
$GPTRF - Transit Fix Data
$GPSTN - Multiple Data ID
$GPVBW - Dual Ground / Water Speed
$GPVTG - Track made good and ground speed
$GPWPL - Waypoint location
$GPXTE - Cross-track error, Measured
$GPZDA - Date & Time

In our application the $GPRMC parse was the most suitable one.

$GPRMC
Recommended minimum specific GPS/Transit data

eg1. $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
eg2. $GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

225446 Time of fix 22:54:46 UTC
A Navigation receiver warning A = OK, V = warning

4916.45,N Latitude 49 deg. 16.45 min North
12311.12,W Longitude 123 deg. 11.12 min West
000.5 Speed over ground, Knots
054.7 Course Made Good, True
191194 Date of fix 19 November 1994
020.3,E Magnetic variation 20.3 deg East
*68 mandatory checksum

eg3. $GPRMC,220516,A,5133.82,N,00042.24,W,173.8,231.8,130694,004.2,W*70
1 2 3 4 5 6 7 8 9 10 11 12

1 220516 Time Stamp
2 A validity - A-ok, V-invalid
3 5133.82 current Latitude
4 N North/South
5 00042.24 current Longitude
6 W East/West
7 173.8 Speed in knots
8 231.8 True course
9 130694 Date Stamp
10 004.2 Variation
11 W East/West
12 *70 checksum

eg4. $GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a*hh
1 = UTC of position fix
2 = Data status (V=navigation receiver warning)
3 = Latitude of fix
4 = N or S
5 = Longitude of fix
6 = E or W
7 = Speed over ground in knots
8 = Track made good in degrees True
9 = UT date
10 = Magnetic variation degrees (Easterly var. subtracts from true course)
11 = E or W
12 = Checksum

2. code

Program file

main.c | GPS_interface.h | GPS_program.c

```
/*********************************************************************************************************************
Author      : Graduation project 2021
Date        : 9 May 2021
Version     : V01
 *********************************************************************************************************************/
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "Data_Type.h"
#include "BIT_MATH.h"

#include "GPS_interface.h"
#include "UART_interface.h"

//$GPRMC,200751.00,A,2758.612,N,08210.515,E,9.879,105.80,230394,,,A*6C
void readGPS(void)
{
    u8 i=0,index;
    char array[100],GPS_Parts[10][20],*token,latitudeResult[10],longitudeResult[10];
    const char comma[2] = ",";
    char c0;
    float latitude=0.0,longitude=0.0,seconds=0.0,result=0.0,minutes=0.0;
    u8 degrees;

    UART_sendString(UART_0,"\r\n");
    c0=UART_ReadData(UART_5);
    if(c0=='$')
    {
        //UART_SendData(UART_0,c0);
        u8 c1=UART_ReadData(UART_5);                    /*read the data from the GPS*/
        //UART_SendData(UART_0,c1);
        if(c1=='G')
        {
            u8 c2=UART_ReadData(UART_5);                    /*read the data from the GPS*/
            //UART_SendData(UART_0,c2);
            if(c2=='P')
            {
                u8 c3=UART_ReadData(UART_5);                    /*read the data from the GPS*/
                //UART_SendData(UART_0,c3);
                if(c3=='R')
                {
                    u8 c4 =UART_ReadData(UART_5);                    /*read the data from the GPS*/
                    //UART_SendData(UART_0,c4);
                    if(c4=='M')
                    {
                        u8 c5=UART_ReadData(UART_5);                    /*read the data from the GPS*/
                        //UART_SendData(UART_0,c5);
```

Figure 115: GPS code

main.c | GPS_interface.h | GPS_program.c

```c
            //UART_SendData(UART_0,c5);
            if(c5=='C')
            {
                u8 c6=UART_ReadData(UART_5);                    /*read the data from the GPS*/
                //UART_SendData(UART_0,c6);
                if(c6==',')
                {
                    UART_sendString(UART_0,"&GPRMC,");
                    u8 c7=UART_ReadData(UART_5);                    /*read the data from the GPS*/
                    // Assign data to the GPSValues Array
                    while(c7!='*')
                    {
                        UART_SendData(UART_0,c7);
                        array[i]=c7;
                        c7=UART_ReadData(UART_5);
                        i++;
                    }
                    /*function to divide the NEMA code into parts*/
                    index=0;
                    token = strtok(array, comma);
                    while( token != NULL )
                    {
                        strcpy(GPS_Parts[index], token);
                        token = strtok(NULL, comma);
                        index++;
                    }

                    //check if the NEMA code is valid or not
                    if(strcmp(GPS_Parts[1],"A")==0)
                    {
                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"Valid");

                        /*convert string into double*/
                        latitude=string_to_double(GPS_Parts[2]);                        //good function

                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"pass");

                        degrees=latitude/100;
                        minutes=latitude-(double)(degrees*100);
                        seconds=minutes/60.00;
                        result=degrees+seconds;

                        float_to_string(result,latitudeResult);
                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"pass again");

                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"Longitude = ");
                        UART_sendString(UART_0,GPS_Parts[2]);

                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"Latitude = ");
                        UART_sendString(UART_0,GPS_Parts[4]);

                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"Latitude value float = ");
                        UART_sendString(UART_0,latitudeResult);

                        /*convert string into double*/
                        longitude=string_to_double(GPS_Parts[2]);                        //good function

                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"pass");

                        degrees=longitude/100;
                        minutes=longitude-(double)(degrees*100);
                        seconds=minutes/60.00;
                        result=degrees+seconds;

                        float_to_string(result,longitudeResult);
                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"pass again");

                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"Longitude = ");
                        UART_sendString(UART_0,GPS_Parts[2]);

                        UART_sendString(UART_0,"\r\n");
                        UART_sendString(UART_0,"Longitude value float = ");
                        UART_sendString(UART_0,longitudeResult);



                    }
                    else if (strcmp(GPS_Parts[1],"V")==0)
                    {
                        UART_sendString(UART_0,"InValid");
                    }
                    /*for (j=0;j<10;j++)
                    {
                        UART_sendString(UART_0,GPS_Parts[j]);
                    }
                     */
                    //UART_sendString(UART_0,"%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%");
```

main.c | GPS_interface.h | GPS_program.c

```c
int n_tu(int number, int count)
{
    int result = 1;
    while((count--) > 0)
    {
        result *= number;
    }

    return result;
}


void float_to_string(float f, char r[])
{
    long long int length, length2, i, number, position, sign;
    float number2;

    sign = -1;   // -1 == positive number
    if (f < 0)
    {
        sign = '-';
        f *= -1;
    }

    number2 = f;
    number = f;
    length = 0;  // Size of decimal part
    length2 = 0; // Size of tenth

    // Calculate length2 tenth part
    while( (number2 - (float)number) != 0.0 && !((number2 - (float)number) < 0.0) )
    {
        number2 = f * (n_tu(10.0, length2 + 1));
        number = number2;

        length2++;
    }

    // Calculate length decimal part
    for (length = (f > 1) ? 0 : 1; f > 1; length++)
        f /= 10;

    position = length;
    length = length + 1 + length2;
    number = number2;
    if (sign == '-')
    {
        length++;
        position++;
        position++;
    }

    for (i = length; i >= 0 ; i--)
    {
        if (i == (length))
            r[i] = '\0';
        else if(i == (position))
            r[i] = '.';
        else if(sign == '-' && i == 0)
            r[i] = '-';
        else
        {
            r[i] = (number % 10) + '0';
            number /=10;
        }
    }
}

/*function to convert string to double*/
double string_to_double(const char* s)
{    //definition
    double rez = 0, fact = 1;
    int point_seen,d;
    if (*s == '-'){
        s++;
        fact = -1;
    };
    for (point_seen = 0; *s; s++){
        if (*s == '.'){
            point_seen = 1;
            continue;
        };
        d = *s - '0';
        if (d >= 0 && d <= 9){
            if (point_seen) fact /= 10.0f;
            rez = rez * 10.0f + (float)d;
        };
    };
    return rez * fact;
}

```

3. Test Run:

*main.c | GPS_interface.h | GPS_program.c

```
/*************************************************************************************************************
Author      : Graduation project 2021
Date        : 31 / 5 / 2021
Version     : V01
Description : Code to take the data from UART5(GPS) and sent it to Putty terminal (UART0).

 *************************************************************************************************************/
/***************************WORK***************************/
#include "Data_Type.h"
#include "BIT_MATH.h"
#include "GPIO_interface.h"
#include "UART_interface.h"
#include "GPIO_private.h"
#include "timer_interface.h"

int main(void)
{
    /*BUTTY terminal initializations*/
    Port_Init(PORTA);                       /*Enable the clock for port A for UART0*/
    UART_Init(UART_0);                      /*initialize UART0*/
    Set_Pin_alternative(PORTA,PIN0,PIN1);   /*configure Pin 0 (RX) and pin 1 (TX) to work with alternative function*/

    /*GPS initializations*/
    Port_Init(PORTE);                       /*Enable the clock for port A for UART0*/
    UART_Init(UART_5);                      /*initialize UART0*/
    Set_Pin_alternative(PORTE,PIN4,PIN5);   /*configure Pin 0 (RX) and pin 1 (TX) to work with alternative function*/

    delay_Microsecond(10);

    while(1)
    {
        readGPS();
    }
}
```

Figure 116: Test Run

Using putty terminal to view the UART communication from TivaC to the GPS module:

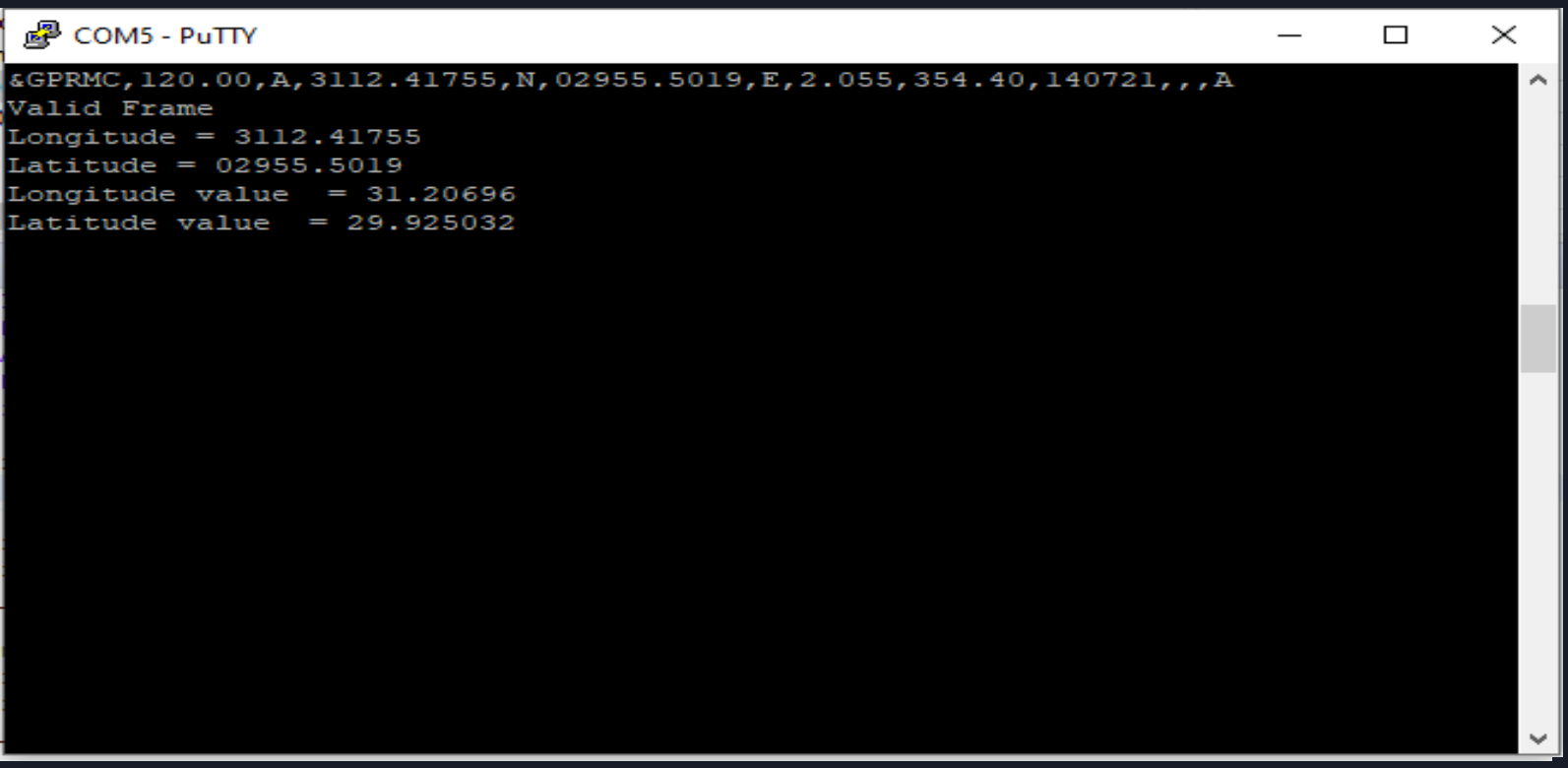


Figure 117: Putty terminal for GPS

Using google maps to convert the coordinates to a location:

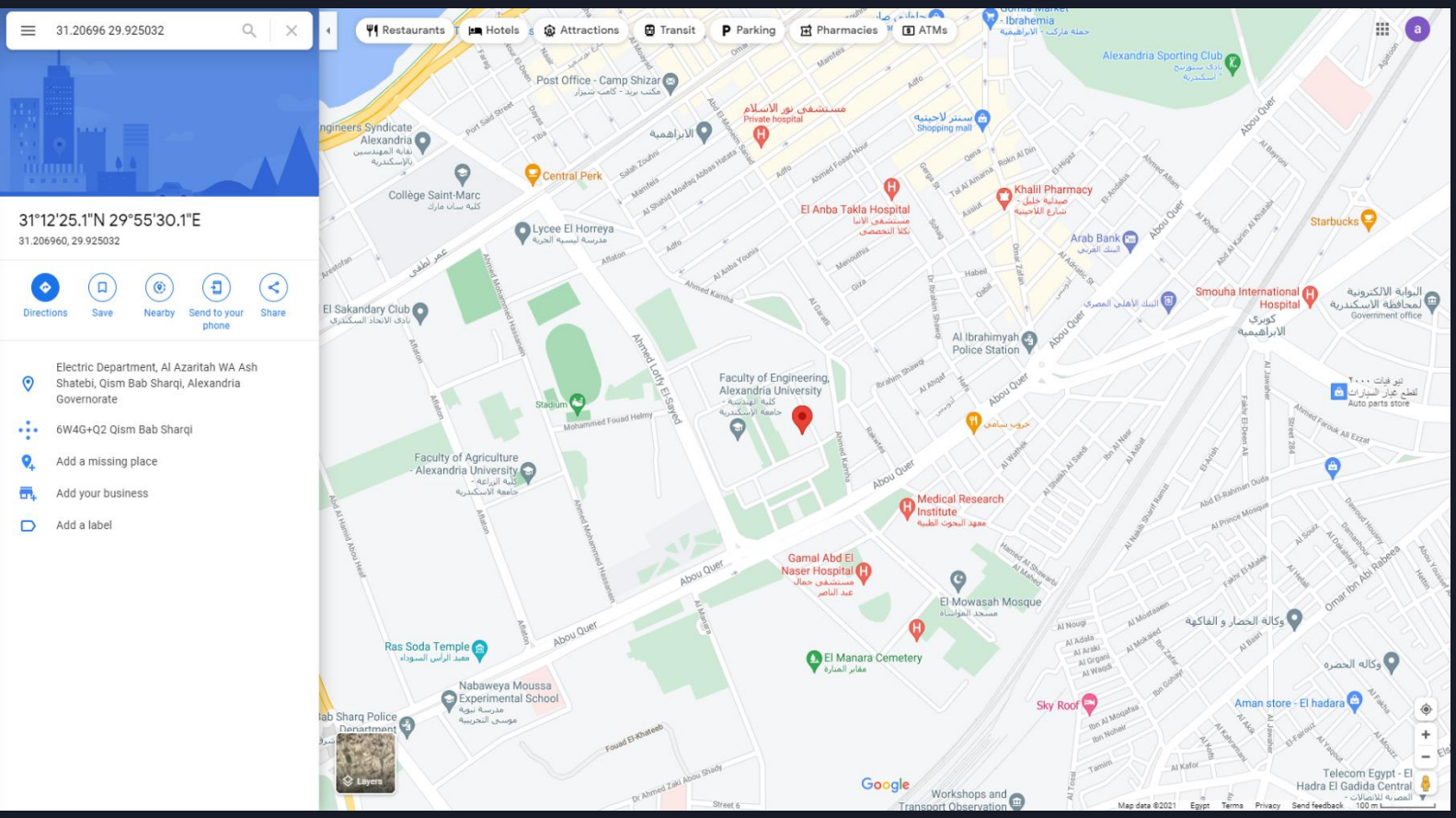


Figure 118: Location on Google Maps

5.5. COMMUNICATION PROTOCOLS

Vehicle-to-vehicle technology uses many technologies like WIFI, LTE and dedicated short-range communications (DSRC), the best of the is DSRC but the module isn't available so we used WIFI in order to transfer frames between cars.
Here we use ES8266 Module to take the data from Tivac, Upload data to Firebase server as the data will
Be available for other cars to know the state of surrounded cars.

5.5.1. WIFI MODULE

The NodeMCU ESP8266 development board comes with the ESP-12E module containing ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects. NodeMCU can be powered using Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

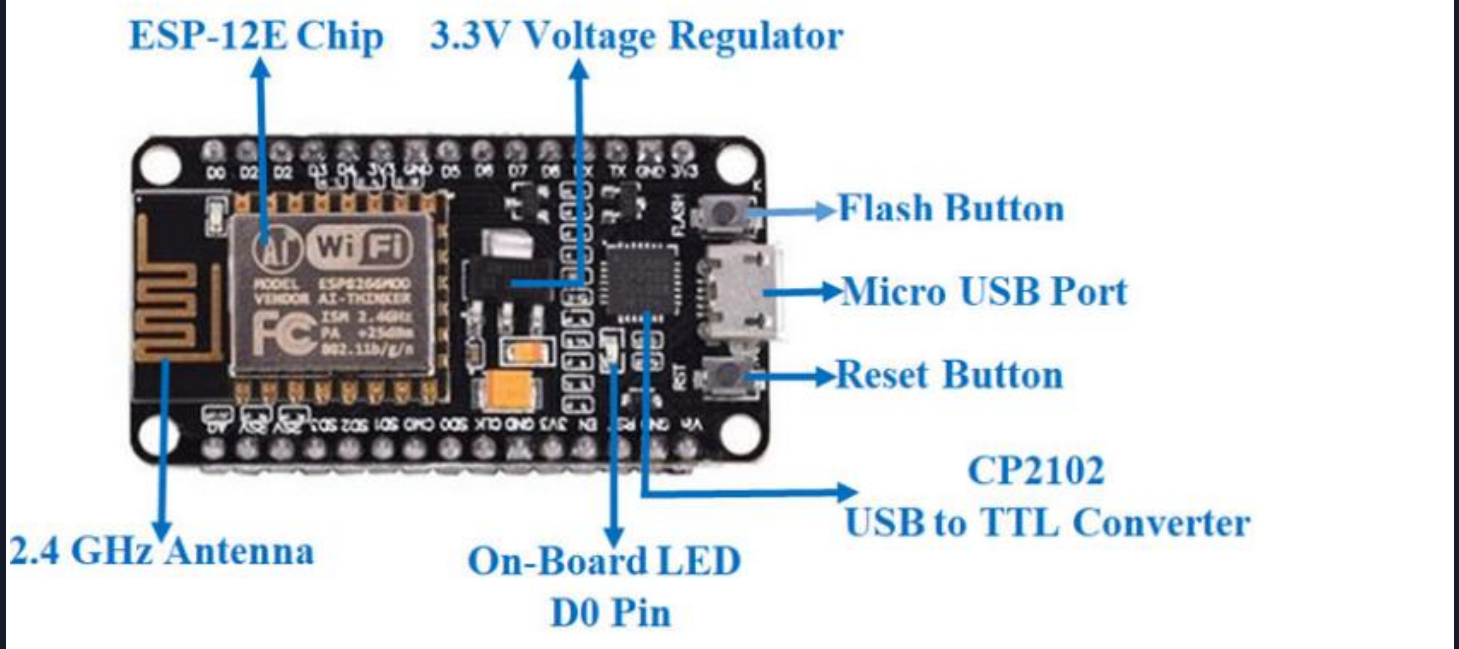


Figure 119: WiFi module

However, as a chip, the ESP8266 is also hard to access and use. You must solder wires, with the appropriate analog voltage, to its pins for the simplest tasks such as powering it on or sending a

keystroke to the “computer” on the chip. You also have to program it in low-level machine instructions that can be interpreted by the chip hardware. This level of integration is not a problem using the ESP8266 as an embedded controller chip in mass-produced electronics. It is a huge burden for hobbyists, hackers, or students who want to experiment with it in their own IoT projects.

But, what about Arduino? The Arduino project created an open-source hardware design and software SDK for their versatile IoT controller. Similar to NodeMCU, the Arduino hardware is a microcontroller board with a USB connector, LED lights, and standard data pins. It also defines standard interfaces to interact with sensors or other boards. But unlike NodeMCU, the Arduino board can have different types of CPU chips (typically an ARM or Intel x86 chip) with memory chips, and a variety of programming environments. There is an Arduino reference design for the ESP8266 chip as well. However, the flexibility of Arduino also means significant variations across different vendors. For example, most Arduino boards do not have WiFi capabilities, and some even have a serial data port instead of a USB port.

5.5.2. UART In ESP8266 :

- **Functional Overview**

  There are two group ESP8266 UART interfaces, respectively: • UART0: - U0TXD: pin26 (U0TXD) - U0RXD: pin25 (U0RXD) - U0CTS: pin12 (MTCK) - U0RTS: pin13 (MTDO) • UART1: - U1TXD: pin14 (GPIO2) The basic working process of transmission FIFO: As long as there has data filling into transmission FIFO, it will immediately start sending process.Since transmission itself is a relatively slow process,other data can be sent to the transmission FIFO simultaneously. Data sending should be paused when the transmission FIFO is full ,or it will cause data loss.Transmission FIFO will sent out one by one in accordance with the order of the data filling in,until the transmission FIFO is completely empty.Data has been sent will be automatically cleared, at the same time transmission FIFO will be more of a vacancy. The basic working process of receiver FIFO: When the hardware logic receives the data, it will fill them into receiver FIFO. Program should withdraw the data timely ,the data-dequeue is also a process of deleting data from FIFO automatically,thus, there will be one more vacancy in receiver FIFO. If the data in the receiver FIFO can not be removed in time, the receiver FIFO will be full which makes data loss.

- **Parameter Configuration**

  UART attribute parameters are all in UART_CONF0 register, can be found in the Uart_register.h. You can configure UART properties through modifying the different bit of the register.

  1. **The Baud Rate** The serial of ESP8266 can support the baud rate range from 300 to 115200 * 40.

2. **Parity Bit** #define UART_PARITY_EN (BIT(1)) Enable check: 1: enable; 0: disable #define UART_PARITY (BIT(0)) Check type setting 1: Odd parity; 0: Even parity Interface: void UART_SetParity(uint8 uart_no, UartParityMode Parity_mode);
3. **Data Bit** #define UART_BIT_NUM 0x00000003 //Length of data bit occupies two bit: Setting these two bit can configure data length 0: 5bit ; 1: 6bit ; 2: 7bit ; 3: 8bit #define UART_BIT_NUM_S 2 //Offset register is 2 (2 bit start) Interface: void UART_SetWordLength(uint8 uart_no, UartBitsNum4Char len)
4. **Stop Bit** #define UART_STOP_BIT_NUM 0x00000003 //The length of data bit occupies two bit: Configure the length of stop bits through setting these two bit can 1 : 1 bit ; 2 : 1.5 bit ; 3 : 2 bit #define UART_STOP_BIT_NUM_S 4 //Register offset is 4 (start from 4th bit) Interface: void UART_SetStopBits(uint8 uart_no, UartStopBitsNum bit_num)
5. **Inverting** Each input and output UART signals can reverse configuration internal. #define UART_DTR_INV (BIT(24)) #define UART_RTS_INV (BIT(23)) #define UART_TXD_INV (BIT(22)) #define UART_DSR_INV (BIT(21)) #define UART_CTS_INV (BIT(20)) #define UART_RXD_INV (BIT(19)) Set the corresponding register,you can reverse the corresponding signal line input / output. Interface: void UART_SetLineInverse (uint8 uart_no, UART_LineLevelInverse inverse_mask)

5.5.3. Cloud computing

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server. Clouds may be limited to a single organization, or be available to multiple organizations (public cloud).
Cloud computing relies on sharing of resources to achieve coherence and economies of scale. Advocates of public and hybrid clouds note that cloud computing allow companies to avoid or minimize up-front IT infrastructure costs. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and that it enables IT teams to more rapidly adjust resources to meet fluctuating and unpredictable demand, providing the burst computing capability: high computing power at certain periods of peak demand.
Cloud providers typically use a "pay-as-you-go" model, which can lead to unexpected operating expenses if administrators are not familiarized with cloud-pricing models.

The availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture and autonomic and utility computing has led to growth in cloud computing. As of 2017, most cloud computers run a Linux-based operating system.

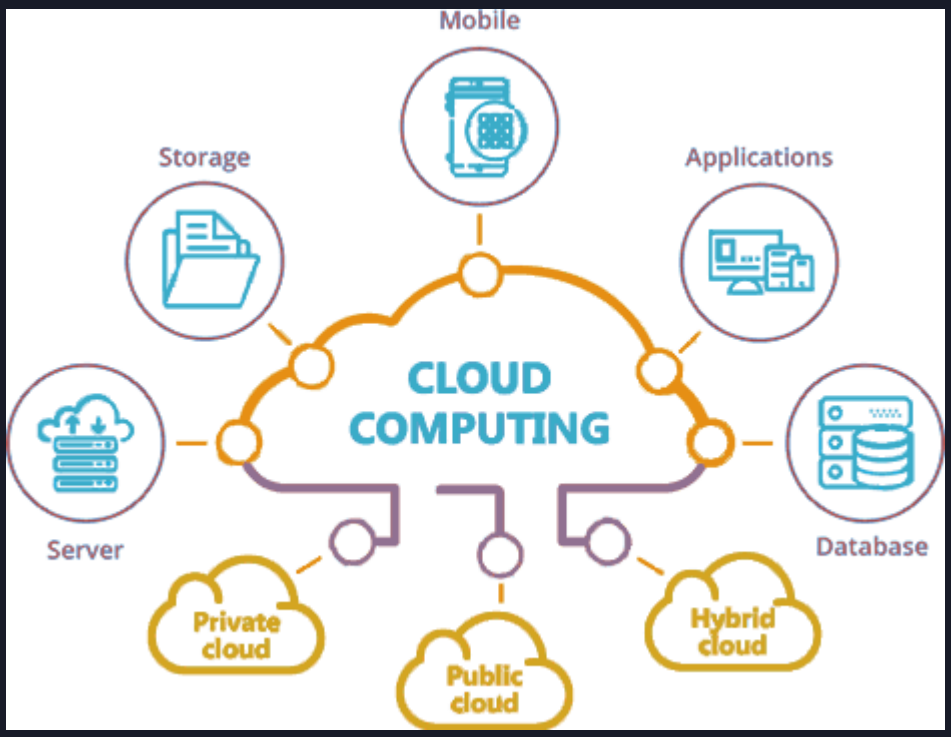


Figure 120: Cloud Computing

5.5.4. History

Cloud computing was popularized with Amazon.com releasing its Elastic Compute Cloud product in 2006.

References to the phrase "cloud computing" appeared as early as 1996, with the first known mention in a Compaq internal document.

The cloud symbol was used to represent networks of computing equipment in the original ARPANET by as early as 1977, and the CSNET by 1981—both predecessors to the Internet itself. The word *cloud* was used as a metaphor for the Internet and a standardized cloud-like shape was used to denote a network on telephony schematics. With this simplification, the implication is that the specifics of how the endpoints of a network are connected are not relevant to understanding the diagram.

The term *cloud* was used to refer to platforms for distributed computing as early as 1993, when Apple spin-off General Magic and AT&T used it in describing their (paired) Tele script and Personal ink technologies. In *Wired's* April 1994 feature "Bill and Andy's Excellent Adventure II", Andy Herzfeld commented on Tele script, General Magic's distributed programming language:

"The beauty of Tele script ... is that now, instead of just having a device to program, we now have the entire Cloud out there, where a single program can go and travel to many different sources of information and create a sort of a virtual service. No one had conceived that before. The example Jim White uses now is a date-arranging service where a software agent goes to the flower store and orders flowers and then goes to the ticket shop and gets the tickets for the show, and everything is communicated to both parties.

5.5.5. Similar Concepts

The goal of cloud computing is to allow users to take benefit from all of these technologies, without the need for deep knowledge about or expertise with each one of them. The cloud aims to cut costs and helps the users focus on their core business instead of being impeded by IT obstacles.[53] The main enabling technology for cloud computing is virtualization. Virtualization software separates a physical computing device into one or more "virtual" devices, each of which can be easily used and managed to perform computing tasks.
With operating system–level virtualization essentially creating a scalable system of multiple independent computing devices, idle computing resources can be allocated and used more efficiently. Virtualization provides the agility required to speed up IT operations and reduces cost by increasing infrastructure utilization. Autonomic computing automates the process through which the user can provision resources on-demand. By minimizing user involvement, automation speeds up the process, reduces labor costs and reduces the possibility of human errors.
Cloud computing uses concepts from utility computing to provide metrics for the services used. Cloud computing attempts to address QoS (quality of service) and reliability problems of other grid computing models.

5.5.6. Cloud computing shares characteristics with:

- **Client–server model** Client–server computing refers broadly to any distributed application that distinguishes between service providers (servers) and service requestors (clients).[54]
- **Computer bureau** A service bureau providing computer services, particularly from the 1960s to 1980s.
- **Grid computing** A form of distributed and parallel computing, whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large tasks.
- **Fog computing** Distributed computing paradigm that provides data, compute, storage and application services closer to the client or near-user edge devices, such as network routers. Furthermore, fog computing handles data at the network level, on smart devices and on

the end-user client-side (e.g. mobile devices), instead of sending data to a remote location for processing.

- **Mainframe computer** Powerful computers used mainly by large organizations for critical applications, typically bulk data processing such as census; industry and consumer statistics; police and secret intelligence services; enterprise resource planning; and financial transaction processing.
- **Utility computing** The "packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility, such as electricity."[55][56]
- **Peer-to-peer** A distributed architecture without the need for central coordination. Participants are both suppliers and consumers of resources (in contrast to the traditional client-server model).
- **Green computing** Study and practice of environmentally sustainable computing or IT.
- **Cloud sandbox** A live, isolated computer environment in which a program, code or file can run without affecting the application in which it runs.

5.5.7. Services Models

1. Infrastructure as a service (IaaS)

Infrastructure as a service" (IaaS) refers to online services that provide high-level APIs used to abstract various low-level details of underlying network infrastructure like physical computing resources, location, data partitioning, scaling, security, backup, etc.
A hypervisor runs the virtual machines as guests. Pools of hypervisors within the cloud operational system can support large numbers of virtual machines and the ability to scale services up and down according to customers' varying requirements. Linux containers run in isolated partitions of a single Linux kernel running directly on the physical hardware.
Linux cgroups and namespaces are the underlying Linux kernel technologies used to isolate, secure and manage the containers. Containerisation offers higher performance than virtualization because there is no hypervisor overhead. IaaS clouds often offer additional resources such as a virtual-machine disk-image library, raw block storage, file or object storage, firewalls, load balancers, IP addresses, virtual local area networks (VLANs), and software bundles.

2. Platform as a service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

PaaS vendors offer a development environment to application developers. The provider typically develops toolkit and standards for development and channels for distribution and payment. In the PaaS models, cloud providers deliver a computing platform, typically including operating system, programming-language execution environment, database, and web server. Application developers develop and run their software on a cloud platform instead of directly buying and managing the underlying hardware and software layers. With some PaaS, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manual.
Some integration and data management providers also use specialized applications of PaaS as delivery models for data. Examples include iPaaS (Integration Platform as a Service) and dPaaS (Data Platform as a Service). iPaaS enables customers to develop, execute and govern integration flows. Under the iPaaS integration model, customers drive the development and deployment of integrations without installing or managing any hardware or middleware. dPaaS delivers integration—and data-management—products as a fully managed service. Under the dPaaS model, the PaaS provider, not the customer, manages the development and execution of programs by building data applications for the customer. dPaaS users access data through data-visualization tools.

1. Software as a service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.
In the software as a service (SaaS) model, users gain access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis or using a subscription fee. In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs. This eliminates the need to install and run the application on the cloud user's own computers, which simplifies maintenance and support. Cloud applications differ from other applications in their scalability—which can be achieved by cloning tasks onto multiple virtual machines at run-time to meet changing work demand. Load balancers distribute the work over the set of virtual machines. This process is transparent to the cloud user, who sees only a single access-point. To accommodate a large number of cloud users, cloud applications can be multitenant, meaning that any machine may serve more than one cloud-user organization.

The pricing model for SaaS applications is typically a monthly or yearly flat fee per user, so prices become scalable and adjustable if users are added or removed at any point. It may also be free. Proponents claim that SaaS gives a business the potential to reduce IT operational costs by outsourcing hardware and software maintenance and support to the cloud provider. This enables the business to reallocate IT operations costs away from hardware/software spending and from personnel expenses, towards meeting other goals. In addition, with applications hosted centrally, updates can be released without the need for users to install new software. One drawback of SaaS comes with storing the users' data on the cloud provider's server. As a result, there could be unauthorized access to the data. Examples of applications offered as SaaS are games and productivity software like Google Docs and Word Online. SaaS applications may be integrated with cloud storage or File hosting services, which is the case with Google Docs being integrated with Google Drive and Word Online being integrated with Onedrive. Modern industrial enterprise performance management software solutions are typically delivered as a suite of software-as-a-service (SaaS) applications, providing enterprise-level visibility and business intelligence insights.

Figure 121: Cloud Computing Algorithm

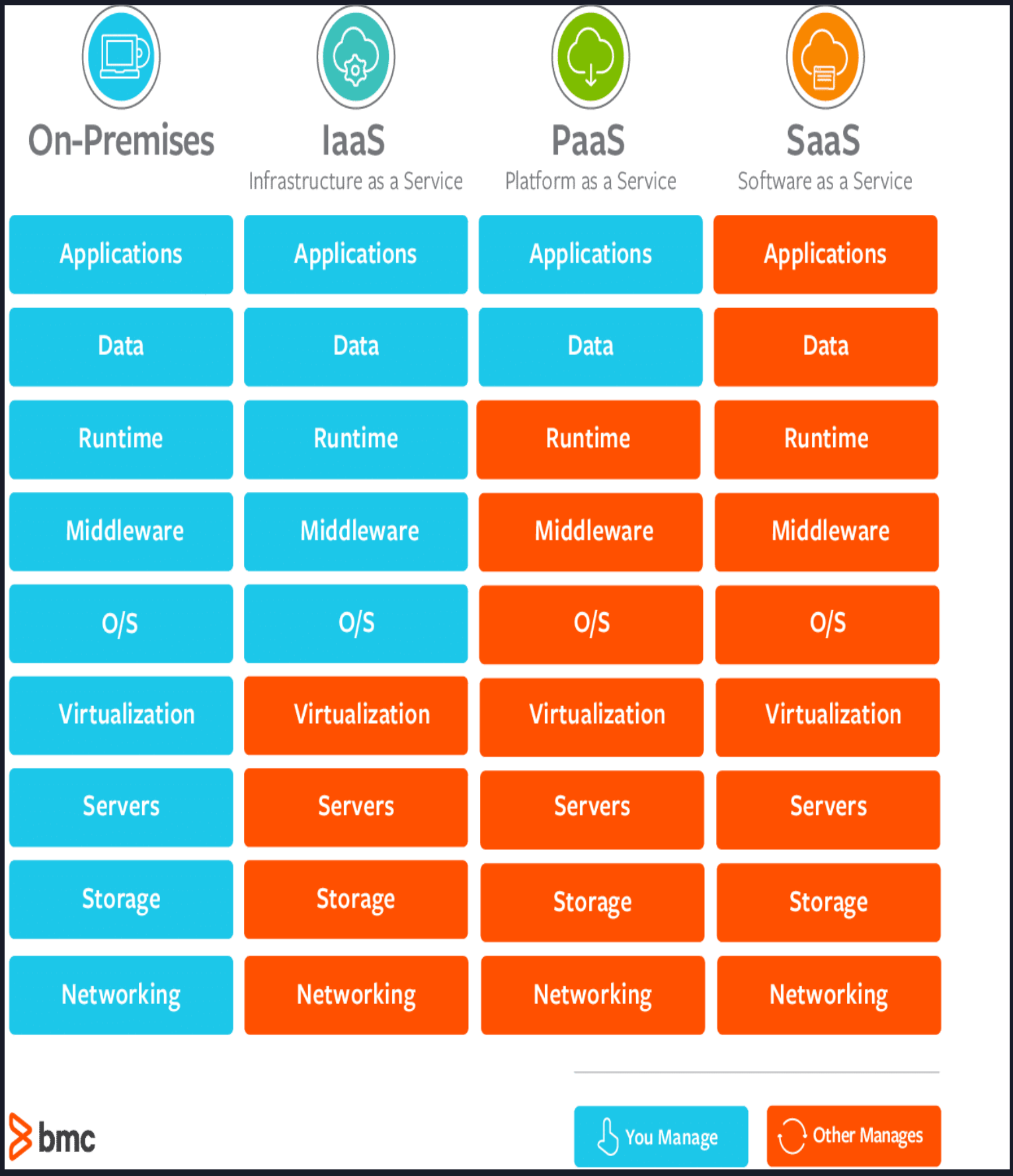


Figure 122: Software As Service (SAS)

5.5.8. Deployment models

1. Private cloud

Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third party, and hosted either internally or externally. Undertaking a private cloud project requires significant engagement to virtualize the business environment, and requires the organization to reevaluate decisions about existing resources. It can improve business, but every step in the project raises security issues that must be addressed to prevent serious vulnerabilities. Self-run data centers are generally capital intensive. They have a significant physical footprint, requiring allocations of space, hardware, and environmental controls. These assets have to be refreshed periodically, resulting in additional capital expenditures. They have attracted criticism because users "still have to buy, build, and manage them" and thus do not benefit from less hands-on management, essentially "[lacking] the economic model that makes cloud computing such an intriguing concept".

2. Public cloud

Cloud services are considered "public" when they are delivered over the public Internet, and they may be offered as a paid subscription, or free of charge. Architecturally, there are few differences between public- and private-cloud services, but security concerns increase substantially when services (applications, storage, and other resources) are shared by multiple customers. Most public-cloud providers offer direct-connection services that allow customers to securely link their legacy data centers to their cloud-resident applications.
Several factors like the functionality of the solutions, cost, integrational and organizational aspects as well as safety & security are influencing the decision of enterprises and organizations to choose a public cloud or on-premises solution.

3. Hybrid cloud

Gartner defines a hybrid cloud service as a cloud computing service that is composed of some combination of private, public and community cloud services, from different service providers. A hybrid cloud service crosses isolation and provider boundaries so that it can't be simply put in one category of private, public, or community cloud service. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service.
Varied use cases for hybrid cloud composition exist. For example, an organization may store sensitive client data in house on a private cloud application, but interconnect that application to a business intelligence application provided on a public cloud as a software service. This example of hybrid cloud extends the capabilities of the enterprise to deliver a specific business service through the addition of externally available public cloud services. Hybrid cloud adoption depends on a number of factors such as data security and compliance requirements, level of control needed over data, and the applications an organization uses.

Another example of hybrid cloud is one where IT organizations use public cloud computing resources to meet temporary capacity needs that can not be met by the private cloud. This capability enables hybrid clouds to employ cloud bursting for scaling across clouds. Cloud bursting is an application deployment model in which an application runs in a private cloud or data center and "bursts" to a public cloud when the demand for computing capacity increases. A primary advantage of cloud bursting and a hybrid cloud model is that an organization pays for extra compute resources only when they are needed. Cloud bursting enables data centers to create an in-house IT infrastructure that supports average workloads, and use cloud resources from public or private clouds, during spikes in processing demands. The specialized model of hybrid cloud, which is built atop heterogeneous hardware, is called "Cross-platform Hybrid Cloud". A cross-platform hybrid cloud is usually powered by different CPU architectures, for example, x86-64 and ARM, underneath. Users can transparently deploy and scale applications without knowledge of the cloud's hardware diversity. This kind of cloud emerges from the rise of ARM-based system-on-chip for server-class computing.
Hybrid cloud infrastructure essentially serves to eliminate limitations inherent to the multi-access relay characteristics of private cloud networking. The advantages include enhanced runtime flexibility and adaptive memory processing unique to virtualized interface models .

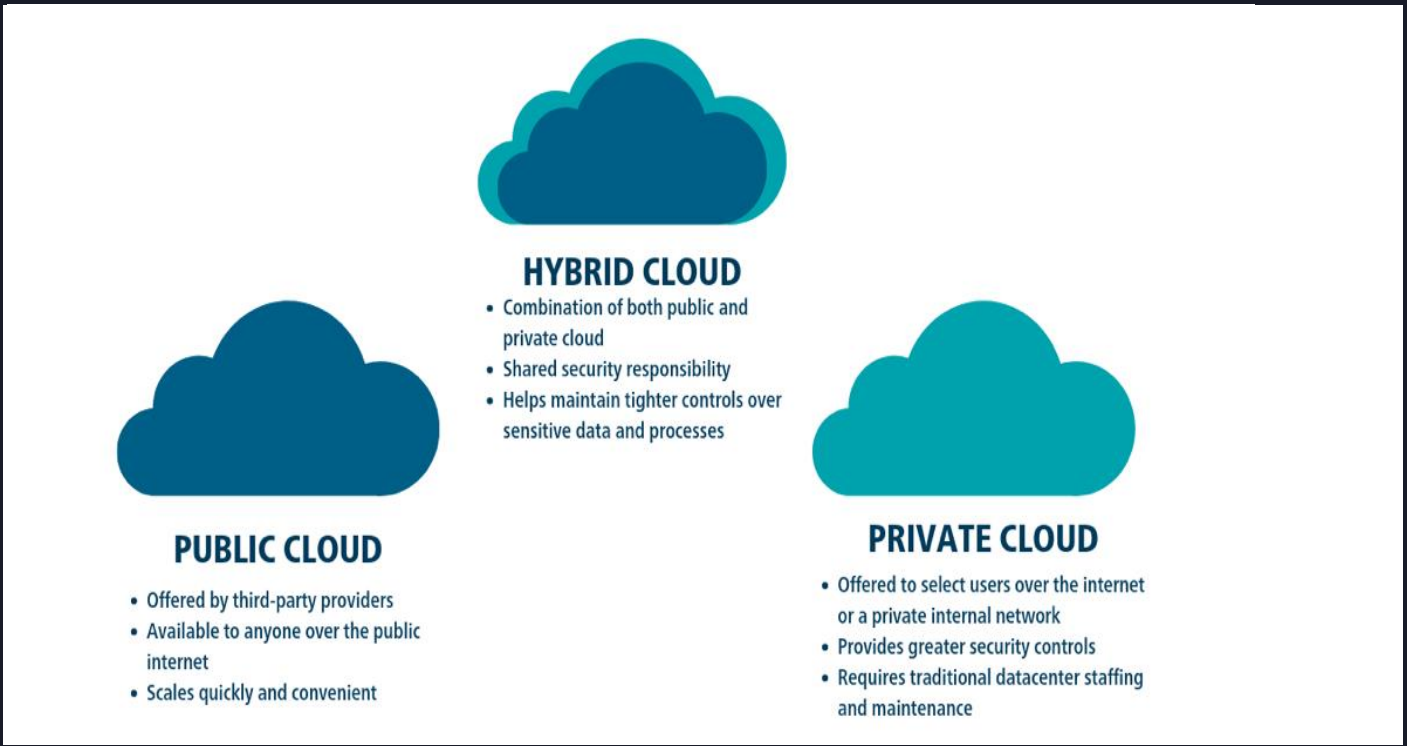


Figure 123: Hybrid cloud

5.5.9. Firebase:

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure.
Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

Key Features

1. Authentication

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

2. Realtime database

Data is synced across all clients in realtime and remains available even when an app goes offline.

3. Hosting

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

4. Test lab

The application is tested on virtual and physical devices located in Google's data centers.

5. Notifications

Notifications can be sent with firebase with no additional coding.

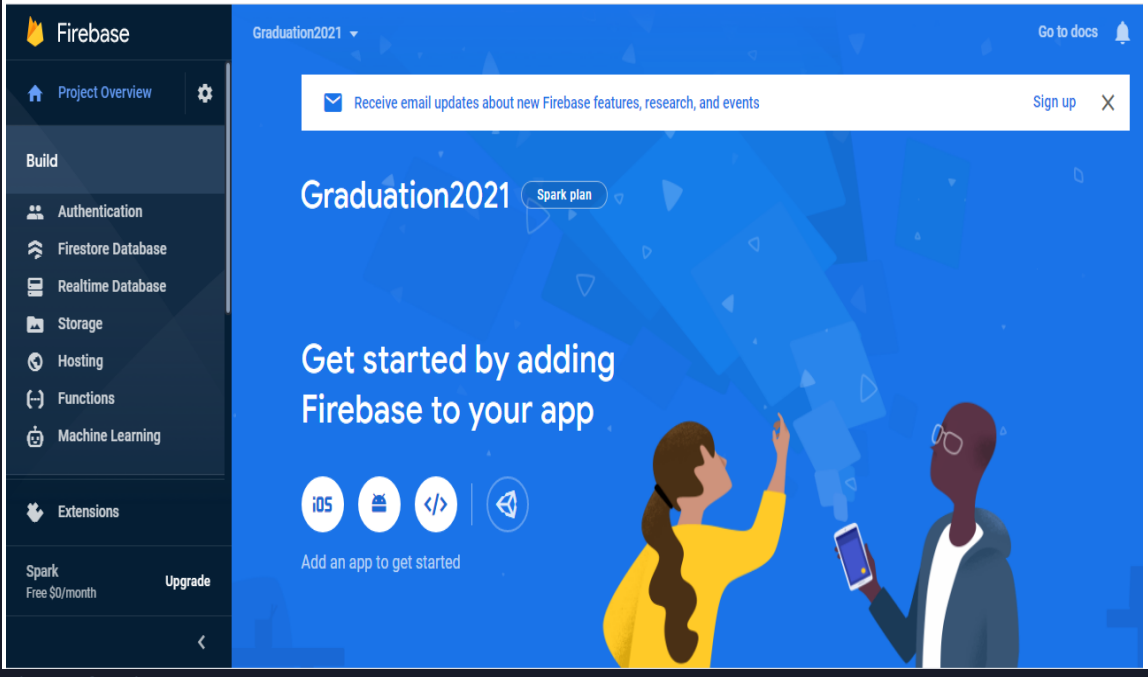


Figure 124: FireBase

5.5.10. Understand Firebase projects

This page offers brief overviews of several important concepts about Firebase projects. When available, follow the links to find more detailed information about features, services, and even other platforms. At the bottom of this page, find a listing of general best practices for Firebase projects.

1. Relationship between Firebase projects, apps, and products

A Firebase project is the top-level entity for Firebase. In a project, you create Firebase apps by registering your iOS, Android, or web apps. After you register your apps with Firebase, you can add the Firebase SDKs for any number of Firebase products, like Analytics, Cloud Firestore, Performance Monitoring, or Remote Config.

2. Relationship between Firebase projects and Google Cloud

When you create a new Firebase project in the Firebase console, you're actually creating a Google Cloud project behind the scenes. You can think of a Google Cloud project as a virtual container for data, code, configuration, and services. **A Firebase project is a Google Cloud project that has additional Firebase-specific configurations and services.** You can even create a Google Cloud project first, then add Firebase to the project later.

Since a Firebase project *is* a Google Cloud project:

- Projects that appear in the Firebase console also appear in the Google Cloud Console and Google APIs console.
- Billing and permissions for projects are shared across Firebase and Google Cloud.
- Unique identifiers for a project (like the project number and the project ID) are shared across Firebase and Google Cloud.
- You can use products and APIs from both Firebase and Google Cloud in a project.
- Deleting a project deletes it across Firebase and Google Cloud.

Note: If you're using the Firebase Management REST API to programmatically create a Firebase project, you must first create a Google Cloud project, then add Firebase services to the existing project.

3. Setting up a Firebase project and registering apps

You can set up a Firebase project and register apps in the Firebase console (or, for advanced use cases, via the Firebase Management REST API or the Firebase CLI). When you set up a project and register apps, you need to make some organizational decisions and add Firebase-specific configuration information to your local projects.

Make sure to review some general project-level best practices (at the bottom of this page) before setting up a project and registering apps.

- **The project name**

When you create a project, you provide a **project name**. This identifier is the *internal-only name* for a project in the Firebase console, the Google Cloud Console, and the Firebase CLI. The project name is not exposed in any publicly visible Firebase or Google Cloud product, service, or resource; it simply serves to help you more easily distinguish among multiple projects.

Note: The project number and the project ID are the truly *unique identifiers* for a project across all of Firebase and Google Cloud.

You can edit a project name at any time in the settings **Project settings** of the Firebase console. The project name is displayed in the top pane.

- **The project number**

A Firebase project (and its associated Google Cloud project) has a **project number**. This is the Google-assigned globally unique canonical identifier for the project. Use this identifier when configuring integrations and/or making API calls to Firebase, Google, or third-party services.

- **API calls and the project number**

For many API calls, you need to include a unique identifier for a project. Although many APIs accept the project ID, it's recommended that you use the **project number** for making API calls to Firebase, Google, or third-party services.

Learn more about using project identifiers, especially the project number, in Google's AIP 2510 standard.

- **Find the project number**

- Firebase console: Click settings **Project settings**. The project number is displayed in the top pane.
- Firebase CLI: Run firebase projects:list. The project number is displayed along with all the Firebase projects associated with your account.
- Firebase Management REST API: Call projects.list. The response body contains the project number in the FirebaseProject object.

- **The project ID**

A Firebase project (and its associated Google Cloud project) has a **project ID**. This is a user-defined unique identifier for the project across all of Firebase and Google Cloud. When you create a Firebase project, Firebase automatically assigns a unique ID to the project, but you can edit it during project setup. This identifier should generally be treated as a convenience alias to reference the project.

If you delete a project, the project ID is also deleted and can never be used again by any other project.

4. Firebase resources and the project ID

The project ID displays in publicly visible Firebase resources, for example:

- Default Hosting subdomain — *PROJECT_ID*.web.app and *PROJECT_ID*.firebaseapp.com

- Default Realtime Database URL — *PROJECT_ID*-default-rtdb.firebaseio.com or *PROJECT_ID*-default-rtdb.*REGION_CODE*.firebasedatabase.app
- Default Cloud Storage bucket name — *PROJECT_ID*.appspot.com

For all of the aforementioned resources, you can create non-default instances. The publicly visible names of non-defaults are fully-customizable. You can connect custom domains to a Firebase-hosted site, shard the Realtime Database, and create multiple Cloud Storage buckets (visit the platform-specific Get Started page).

After Firebase provisions resources for a Firebase project, you cannot change its project ID. To use a specific identifier for Firebase resources, you must edit the project ID during the initial creation of the project.

5. The Firebase CLI and the project ID

For some use cases, you might have multiple Firebase projects associated with the same local app directory. In these situations, when you use the Firebase CLI, you need to pass the --project flag with the firebase commands to communicate which Firebase project you want to interact with.

You can also set up a project alias for each Firebase project so that you don't have to remember project IDs.

6. API calls and the project ID

For many API calls, you need to include a unique identifier for a project. Although many APIs accept the project ID, it's recommended that you use the **project number** for making API calls to Firebase, Google, or third-party services.

Learn more about using project identifiers, especially the project number, in Google's AIP 2510 standard.

- ## Find the project ID

- Firebase console: Click settings **Project settings**. The project ID is displayed in the top pane.
- Firebase CLI: Run firebase projects:list. The project ID is displayed along with all the Firebase projects associated with your account.
- Firebase Management REST API: Call projects.list. The response body contains the project ID in the FirebaseProject object.

- ## Firebase config files and objects

When you register an app with a Firebase project, the Firebase console provides a Firebase configuration file (iOS/Android apps) or a configuration object (web apps) that you add directly to your local app directory.

- For iOS apps, you add a GoogleService-Info.plist configuration file.
- For Android apps, you add a google-services.json configuration file.
- For web apps, you add a Firebase configuration object.

At any time, you can obtain an app's Firebase config file or object.

A Firebase config file or object associates an app with a specific Firebase project and its resources (databases, storage buckets, etc.). The configuration includes "Firebase options", which are parameters required by Firebase and Google services to communicate with Firebase server APIs and to associate client data with the Firebase project and Firebase app. Here are the required, minimum "Firebase options":

- API key: a simple encrypted string used when calling certain APIs that don't need to access private user data (example value: AIzaSyDOCAbC123dEf456GhI789jKl012-MnO)
- Project ID: a user-defined unique identifier for the project across all of Firebase and Google Cloud. This identifier may appear in URLs or names for some Firebase resources, but it should generally be treated as a convenience alias to reference the project. (example value: myapp-project-123)
- **Application ID ("AppID")**: the unique identifier for the Firebase app across all of Firebase with a platform-specific format:
- Firebase iOS apps: GOOGLE_APP_ID (example value: 1:1234567890:ios:321abc456def7890)
  This is *not* an Apple bundle ID.
- Firebase Android apps: mobilesdk_app_id (example value: 1:1234567890:android:321abc456def7890)
  This is *not* an Android package name or Android application ID.
- Firebase Web apps: appId (example value: 1:65211879909:web:3ae38ef1cdcb2e01fe5f0c)

Caution: We do not recommend manually modifying an app's Firebase config file or object. If you initialize an app with invalid or missing values for any of these required "Firebase options", then your end users may experience serious issues.

The content of the Firebase config file or object is considered public, including the app's platform-specific ID (iOS bundle ID or Android package name) and the Firebase project-specific values, like the API Key, project ID, Realtime Database URL, and Cloud Storage bucket name. Given this, **use Firebase Security Rules** to protect your data and files in Realtime Database, Cloud Firestore, and Cloud Storage.

For open source projects, we generally do not recommend including the app's Firebase config file or object in source control because, in most cases, your users should create their own Firebase projects and point their apps to their own Firebase resources (via their own Firebase config file or object).

- **Managing a Firebase project**

Make sure to review the general project-level best practices (at the bottom of this page) for considerations that might affect how you manage a Firebase project.

- **Tools to manage a project**
- **Firebase console**

The Firebase console offers the richest environment for managing Firebase products, apps, and project-level settings.

Figure 125: FireBase Console

The left-side panel of the console lists the Firebase products, organized by top-level categories. At the top of the left-side panel, access a project's settings by clicking settings. A project's settings include integrations, access permissions, and billing.
The middle of the console displays buttons that launch setup workflows to register various types of apps. After you start using Firebase, the main area of the console changes into a dashboard that displays stats on the products you use.

7. Firebase CLI (a command line tool)

Firebase also offers the Firebase CLI for configuring and managing specific Firebase products, like Firebase Hosting and Cloud Functions for Firebase.
After installing the CLI, you have access to the global firebase command. Use the CLI to link your local app directory to a Firebase project, then deploy new versions of Firebase-hosted content or updates to functions.

- ## Firebase Management REST API

Using the Firebase Management REST API, you can programmatically manage a Firebase project. For example, you can programmatically register an app with a project or list the apps that are already registered (iOS | Android | web).

- ## General best practices

Ensure that all apps within a project are platform variants of the same application from an end-user perspective. It's advisable to register the iOS, Android, and web versions of the same app or game with the same Firebase project. All the apps in a project generally share the same Firebase resources (database, storage buckets, etc.).
If you have *multiple build variants* with different iOS bundle IDs or Android package names defined, you can register each variant with a separate Firebase project. However, if you have variants that share the *same* Firebase resources, register them with the *same* Firebase project.
Note: For each Android app, if you provide a SHA-1 key for the app, you need to provide a package name and SHA-1 key combination that is globally unique across all of Google Cloud.
Here are some general limits for Firebase projects, apps, and sites:

- **Number of projects per account**
- Spark pricing plan — Project-creation quota is limited to a lower count of projects (usually around 5-10).
- Blaze pricing plan — Project-creation quota per account increases substantially as long as the associated Cloud Billing account is in good standing.
  The limit on project-creation quota is rarely a concern for most developers, but if needed, you can request an increase in project quota.
  Be aware that the complete deletion of a project requires 30 days and counts toward project quota until the project is fully deleted.
- **Number of apps per project**
  Firebase restricts the total number of Firebase Apps within a Firebase project to 30.
  You should ensure that all Firebase Apps within a single Firebase project are platform variants of the same application from an end-user perspective. Read more about best practices for multi-tenancy below.
  Learn more about the limit on apps per project in the FAQ.
- **Number of Hosting sites per project**
  The Firebase Hosting multisite feature supports a maximum of 36 sites per project.

Multi-tenancy

Connecting several different logically independent apps and/or web sites to a single Firebase project (often called "multi-tenancy") is not recommended. Multi-tenancy can lead to serious configuration and data privacy concerns problems, including unintended issues with analytics aggregation, shared authentication, overly-complex database structures, and difficulties with security rules.

Generally, **if a set of apps don't share the same data and configurations, strongly consider registering each app with a different Firebase project.**
For example, if you develop a white label application, each independently labelled app should have its own Firebase project, but the iOS and Android versions of that label can be in the same project. Each independently labeled app shouldn't (for privacy reasons) share data with the others.

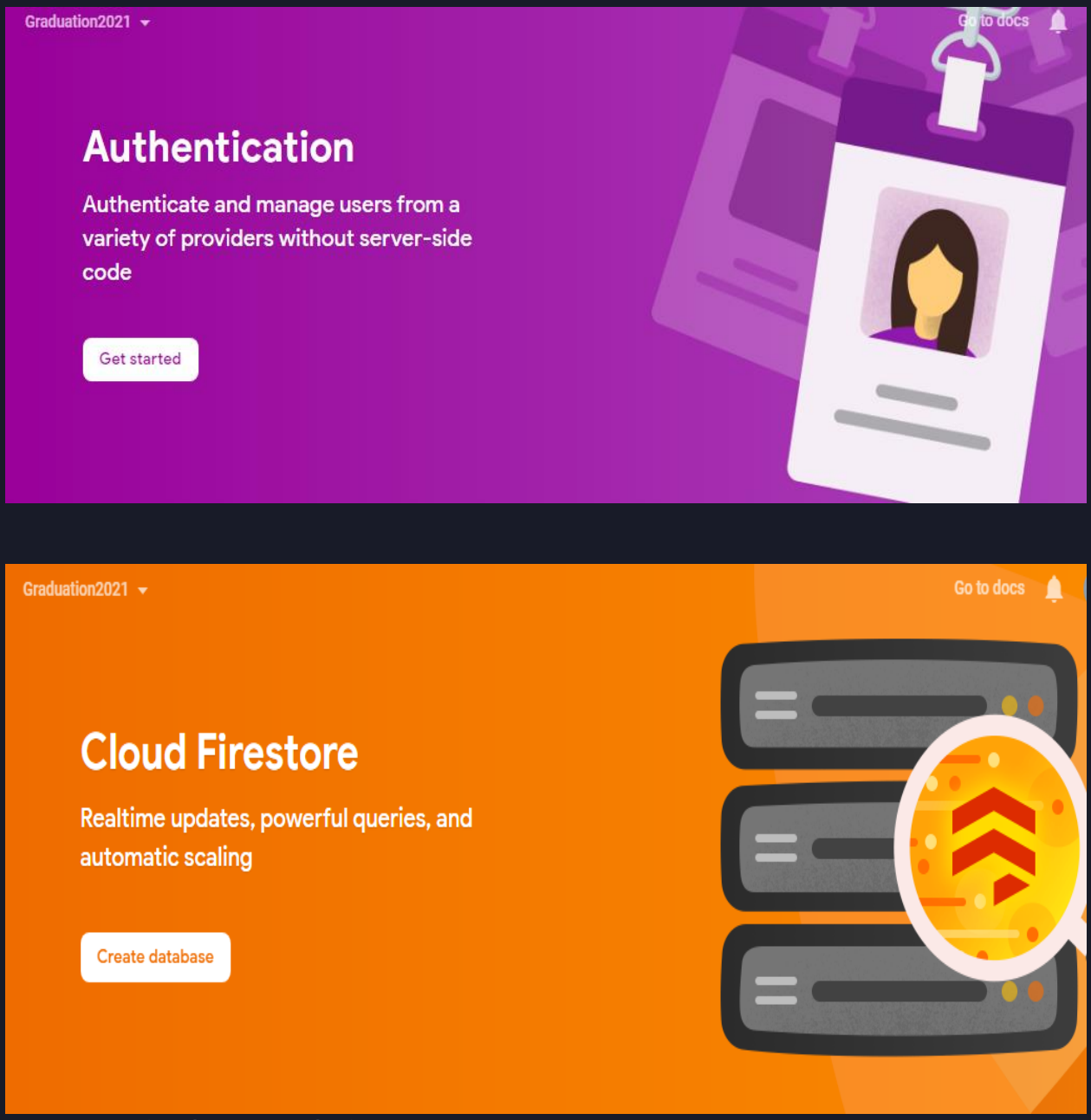


Figure 126: Number of Hosting sites for the project

- **Coding to upload data to firebase :**

File Edit Sketch Tools Help

Upload

sketch_jul13a

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

// Set these to run example.
#define FIREBASE_HOST "graduation2021-50b34-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "eTNGFW5N54uv8Txx67cSO0LhuYUih6eGiGr3RS6U"
#define WIFI_SSID "GATEWAYUNIVERSE"
#define WIFI_PASSWORD "GATEway12345GW"


String longitude1;
String latitude1;
void setup() {
  Serial.begin(9600);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}
```

File Edit Sketch Tools Help

sketch_jul13a

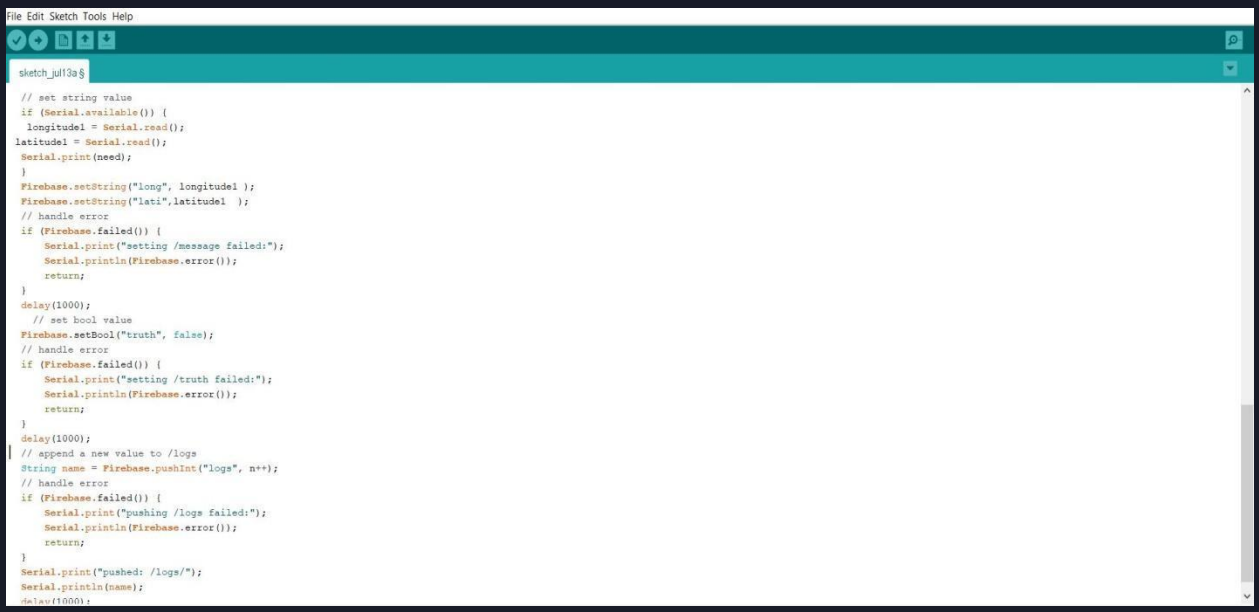
```
int n = 0;

void loop() {
  // set value
  Firebase.setFloat("number", 42.0);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /number failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

  // update value
  Firebase.setFloat("number", 43.0);
  // handle error
  if (Firebase.failed()) {
      Serial.print("setting /number failed:");
      Serial.println(Firebase.error());
      return;
  }
  delay(1000);

  // get value
  Serial.print("number: ");
  Serial.println(Firebase.getFloat("number"));
  delay(1000);

  // remove value
  Firebase.remove("number");
  delay(1000);
```



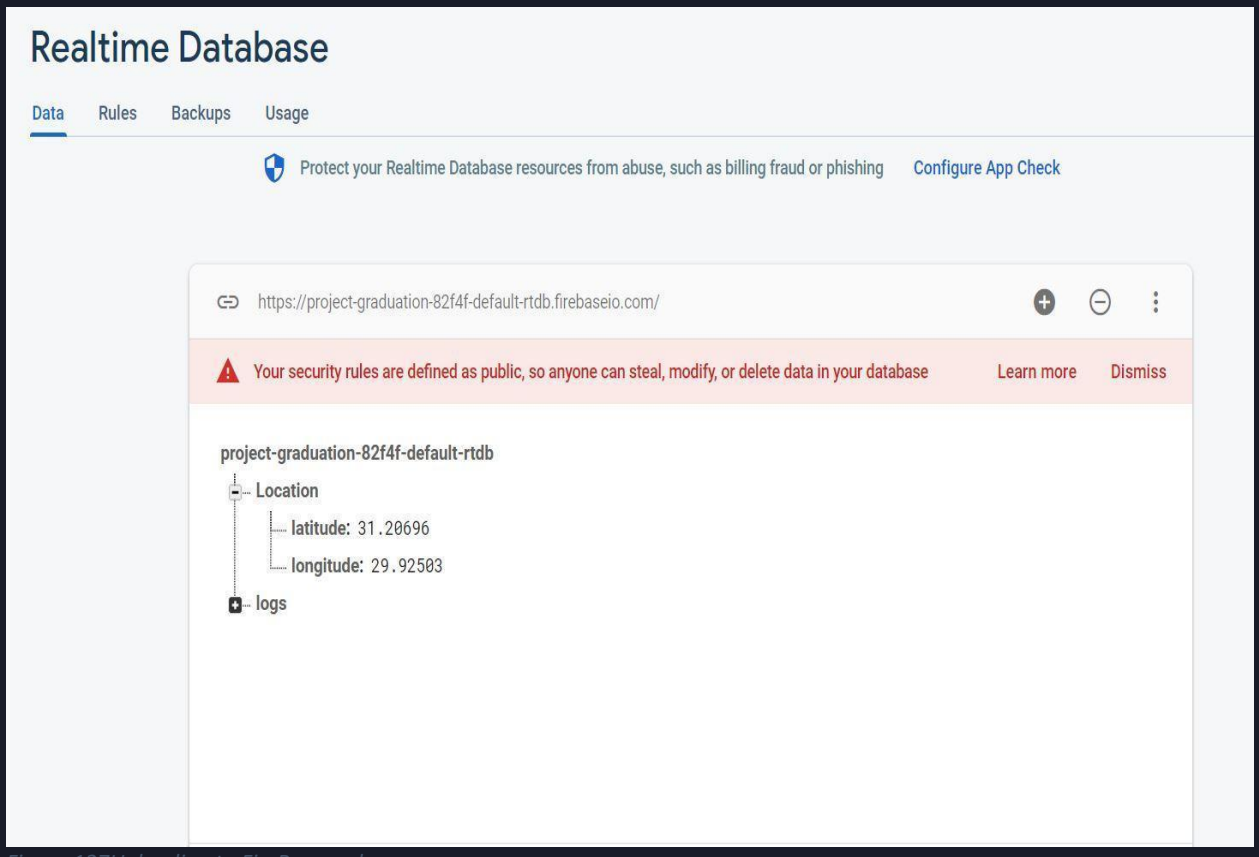


Figure 127Uploading to FireBase code

5.11. Mobile Application

A mobile application, most commonly referred to as an app, is a type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Mobile applications frequently serve to provide users with similar services to those accessed on PCs. Apps are generally small, individual software units with limited function. This use of app software was originally popularized by Apple Inc. and its App Store, which offers thousands of applications for the iPhone, iPad and iPod Touch.

A mobile application also may be known as an app, web app, online app, iPhone app or smartphone app.

Mobile applications are a move away from the integrated software systems generally found on PCs. Instead, each app provides limited and isolated functionality such as a game, calculator or mobile web browsing. Although applications may have avoided multitasking because of the limited hardware resources of the early mobile devices, their specificity is now part of their desirability because they allow consumers to hand-pick what their devices are able to do.

The simplest mobile apps take PC-based applications and port them to a mobile device. As mobile apps become more robust, this technique is somewhat lacking. A more sophisticated approach involves developing specifically for the mobile environment, taking advantage of both its limitations and advantages. For example, apps that use location-based features are inherently built from the ground up with an eye to mobile given that the user is not tied to a location, as on PC.

Apps are divided into two broad categories: native apps and web apps. Native apps are built for a specific mobile operating system, usually iOS or Android. Native apps enjoy better performance and a more finely-tuned user interface (UI), and usually need to pass a much stricter development and quality assurance process before they are released.

Web apps are used in HTML5 or CSS and require minimum device memory since they’re run through a browser. The user is redirected on a specific web page, and all information is saved on a server-based database. Web apps require a stable connection to be used.

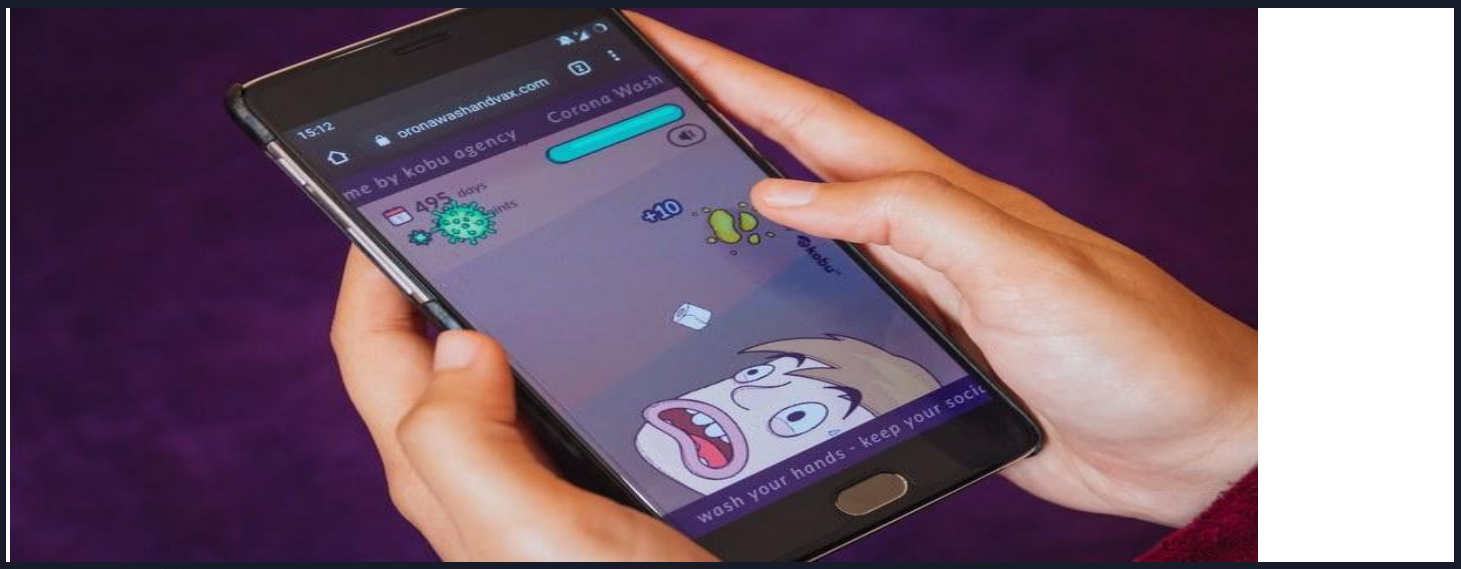


Figure 128: Mobile Apps

5.11.1. Types of mobile application

Mobile applications come in numerous shapes and sizes. Here are the most famous mobile application types to assist you with understanding the latest things in the mobile scene:

- **Mobile gaming applications**

This is the most famous classification of portable applications. You would be astonished to figure out the number of clients who install games on their telephones. Organizations invest/use a huge amount of time and assets into making games and mobile versions of well-known stationary games since it is a particularly lucrative market. According to a new report from Sensor Tower, mobile game downloads reach 12 billion, which is nearly 7 times higher than the second most downloaded category on Google Play. The share of mobile game would reach 40% in 2020 in the total mobile application downloaded. Of all application downloads, 84% of casual games were downloaded, and the rest of them spent on core game mobile applications. The best mobile games like Candy Crush Saga or Angry Birds have become known everywhere in the world.

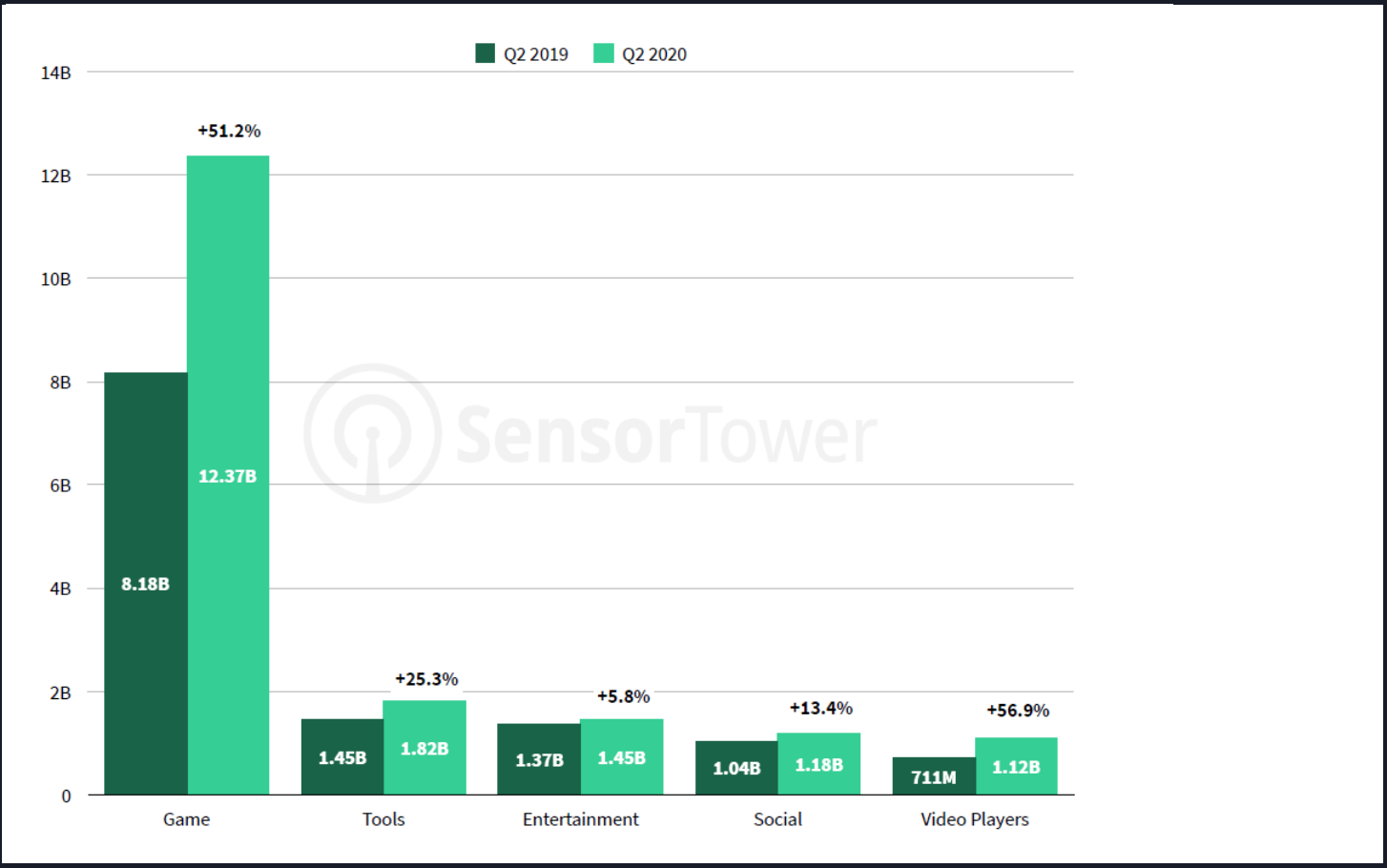


Figure 129: Chart of mobile usage for gaming

- **Educational applications**

This category incorporates portable applications that help clients acquire new abilities and information. For instance, language learning applications like Duolingo have become staggeringly mainstream since they give clients the adaptability they search for in learning. Educational game applications are an incredible apparatus for youngsters. Numerous educational applications end up being famous among educators as well, who use them to make their teaching process better or teach themselves further.

Figure 130: Mobile usage in business

- **Business or productivity applications**

These applications hold an enormous piece of the market today since individuals are progressively inclined to utilizing their smartphones and tablets to perform numerous intricate tasks in a hurry. For instance, applications can assist them with booking tickets, sending messages, or tracking their work progress. Business applications are equipped to boost profitability and limit costs as they permit clients to finish a wide scope of assignments, from purchasing new cartridges for office printers to enlisting another office director.

- **M-commerce applications**

The most famous shopping applications like Amazon or eBay offer the experience of their working assistant forms to mobile users. Mobile commerce applications furnish clients with advantageous admittance to items, as well as many consistent installment strategies for an ideal shopping experience.

- **Lifestyle applications**

This general classification of applications traverses shopping, style, virtual fitting rooms, exercise, dating, and diet applications. These applications essentially center around different parts of the individual way of life.

- **Entertainment applications**

These applications permit clients to transfer video content, look for occasions, talk, or watch content on the web. Online media applications like Facebook or Instagram are incredible models. Additionally, streaming applications, for example, Netflix or Amazon Prime Video have gotten unimaginably well known with clients everywhere in the world. These applications help their mobile users with the various forms and versions of entertaining methods, along with the continuous modification to meet the demand of users.

- **Travel applications**

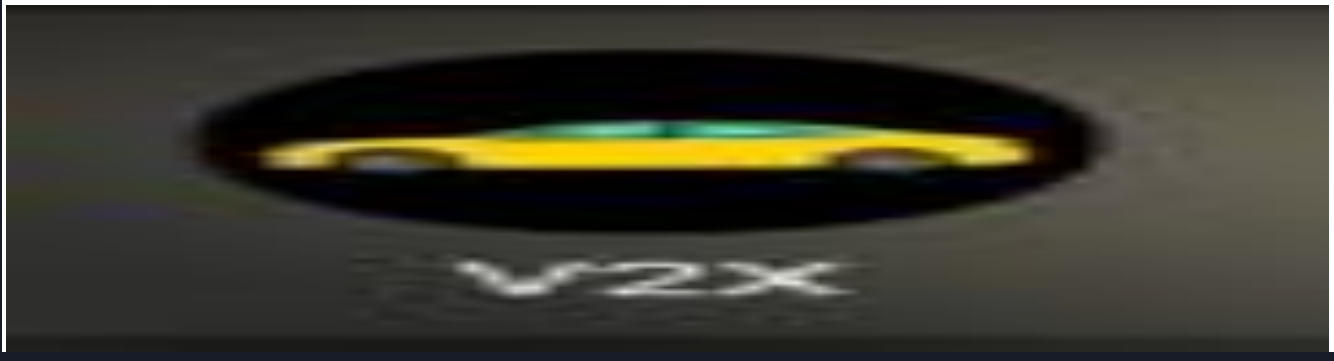
The primary thought behind this classification is assisting clients with traveling without any problem. Travel applications' users might change a cell phone or tablet into a movement journal status so that they would receive very helpful instructions, guidance and preferences. The greater part of the sightseers is carefully sagacious voyagers who realize how to utilize applications for their potential benefit.

- **Utility applications**

these are clear to such an extent that we scarcely even understand that we are utilizing them. Indeed, utility applications typically have the shortest user session times – individuals use them to complete things and afterward proceed onward. The most mainstream sorts of utility applications are standardized identification scanners, trackers, or medical services applications.

5.11.2. SIMULATION

In our project we used Mobile application to show the data that uploaded in the firebase , Normally , Driver won't open the server to see the places of pumps , so mobile application will make it easy as it will show the data on the mobile screen .
By only small steps, he can open the application icon , and show the data if pumps locations.



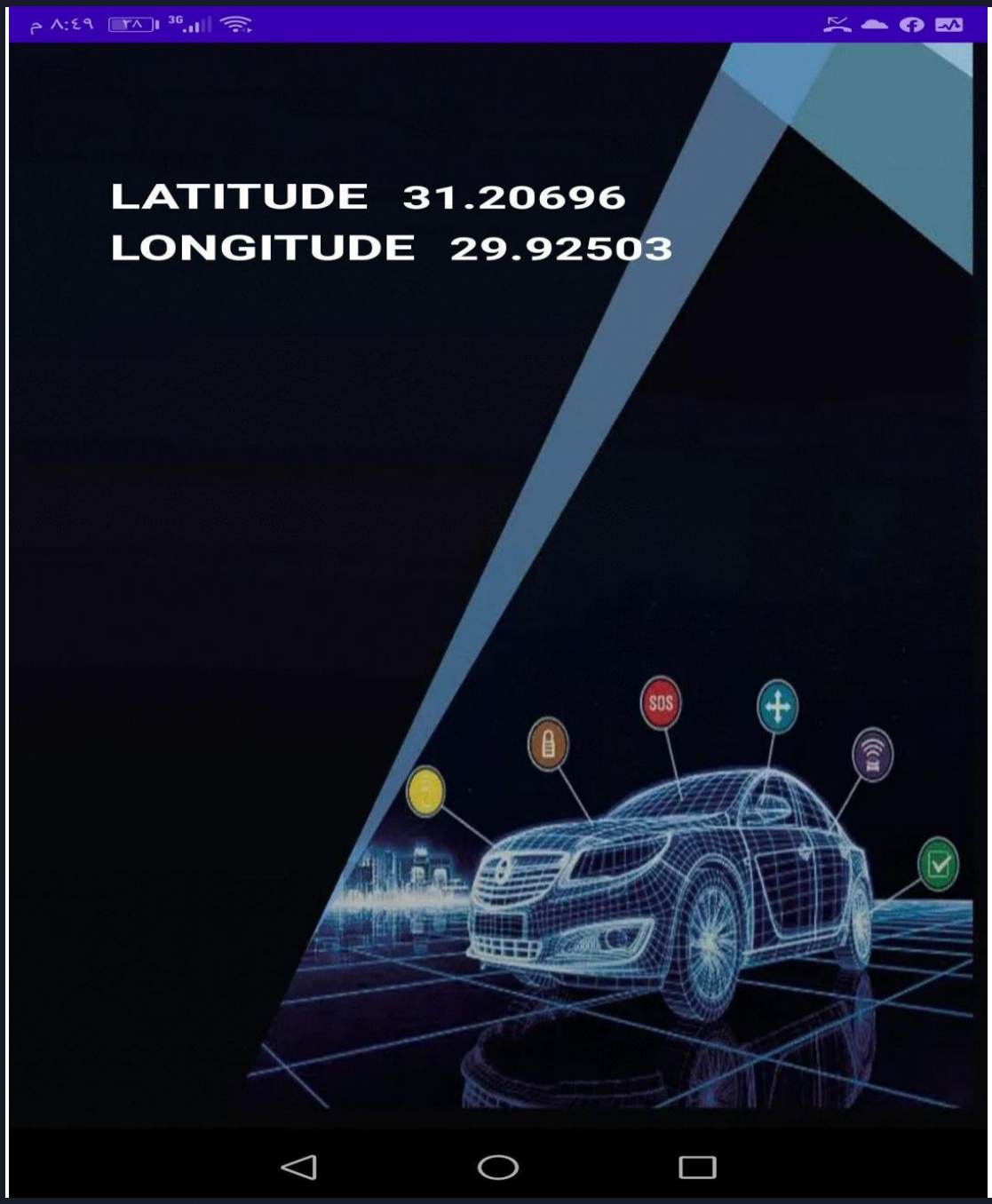
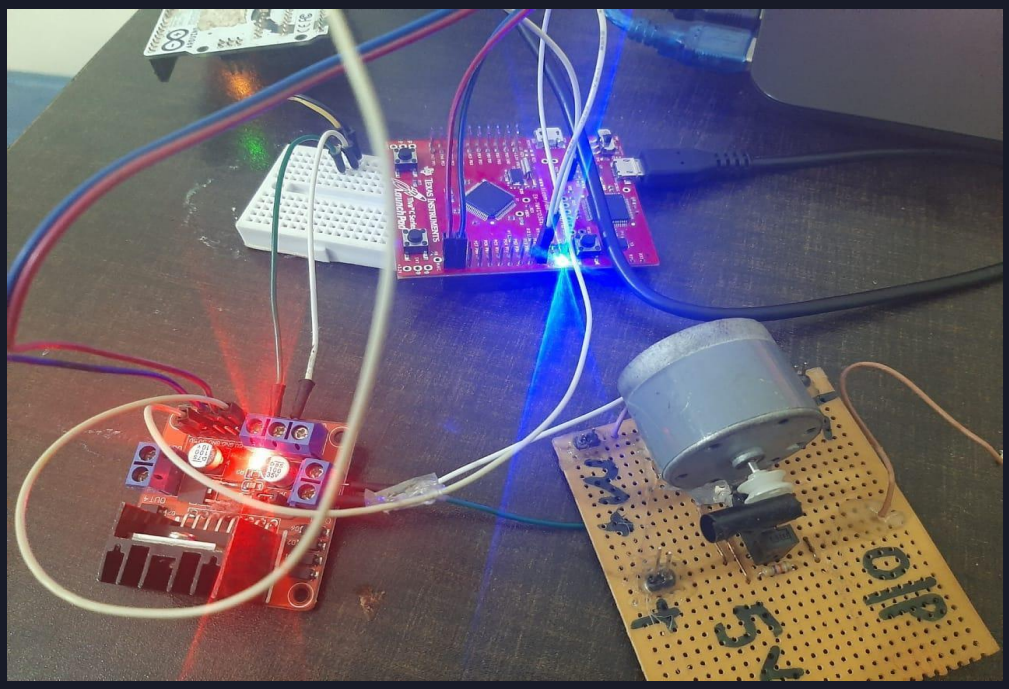


Figure 131: V2X Simulating mobile app

5.12. MOTORS



: Motor circuit132 Figure

5.12.1. DC Motor Introduction

A direct current (DC) motor is a type of electric machine that converts electrical energy into mechanical energy. DC motors take electrical power through direct current, and convert this energy into mechanical rotation.

DC motors use magnetic fields that occur from the electrical currents generated, which powers the movement of a rotor fixed within the output shaft. The output torque and speed depends upon both the electrical input and the design of the motor.

5.12.2. Speed Control

The speed of rotation of motors is directly related to the input voltage. The higher the input voltage, the higher will be the rotational speed of the motor. But the voltage should be within the operating voltage range.

One important point to note here is that if we want to control the speed of a DC motor, we will need to provide a variable voltage to the DC motor. For example, If the operating voltage range of a motor is between 3 – 6 volts. If we apply 3 volts input, the motor will run at its lowest rated speed. Similarly, if we apply 6 volts, the DC motor will run at its highest rated

speed. In short, we can control the speed of rotation by giving a variable input voltage to a DC motor. Usually, a pulse width modulation technique is used to generate a variable dc voltage from constant dc voltage source.

5.12.3. Direction Control

There is a concept of polarity such as positive and negative terminals of a battery. The polarity of input voltage source determines the direction of rotation of a DC motor. For instance, if we connect the positive terminal of battery with one terminal and the negative terminal of battery with another terminal of DC motor, it will start rotating in clockwise direction. But if we interchange the connection of the battery by connecting opposite polarity, the DC motor will start rotating in another direction as shown in the simulation below.

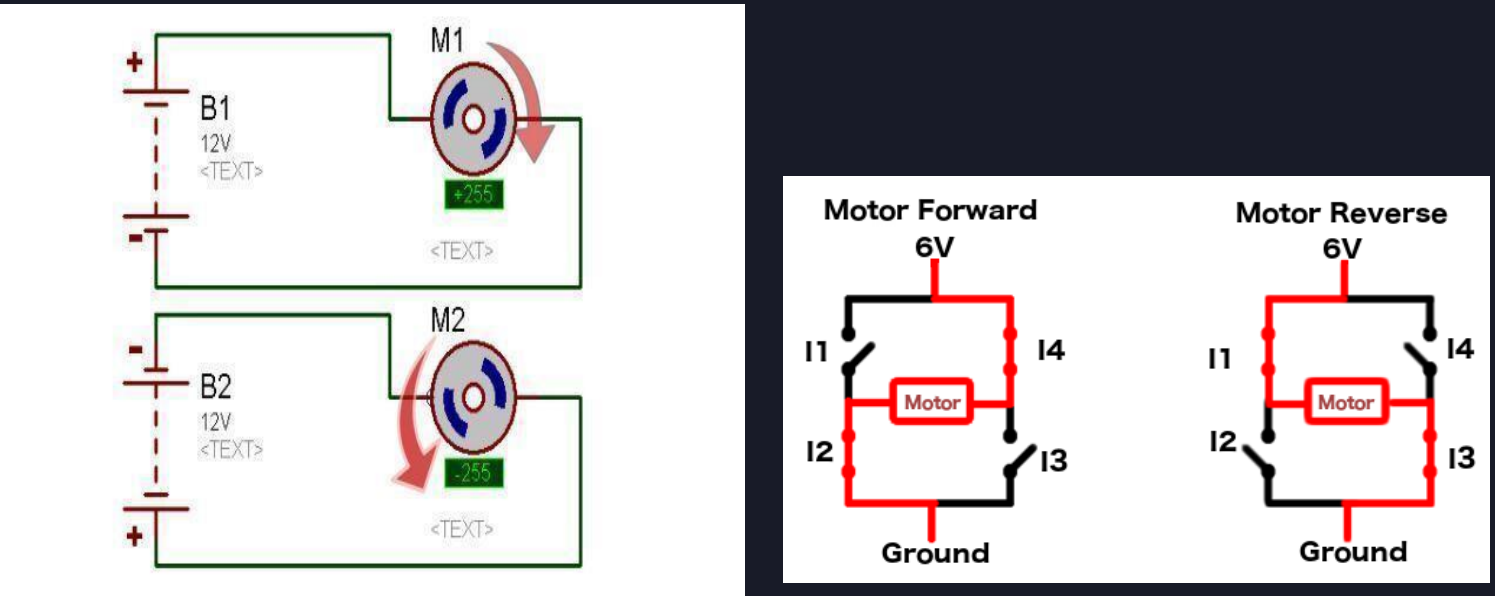
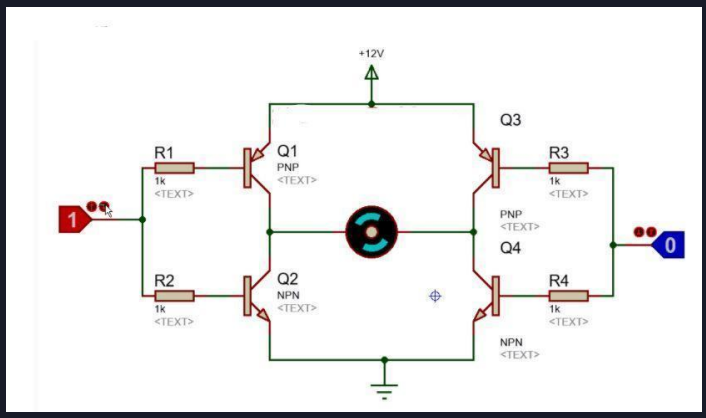


Figure 133: Motor circuit with directions

Generally, an H-Bridge circuit is used to provide opposite polarity power to a DC motor without changing actual power supply connections. An H-Bridge consists of four switches, resistors, and two control signals.



Two logic probes control the direction of a DC motor. When left probe logic signal is on, the motor rotates in an anti-clockwise direction and when right probe logic signal is on, the motor rotates in a clockwise direction.

5.12.4. 9V DC Motor

In this tutorial, we will use a hobby DC motor as shown in the figure below. It consists of two terminals such as terminal_1 and terminal_2. There is no concept of marked polarity in this motor. Because these two terminals are internally connected through a single coil. Therefore, we can reverse the direction of the motor by applying opposite polarity to these terminals.

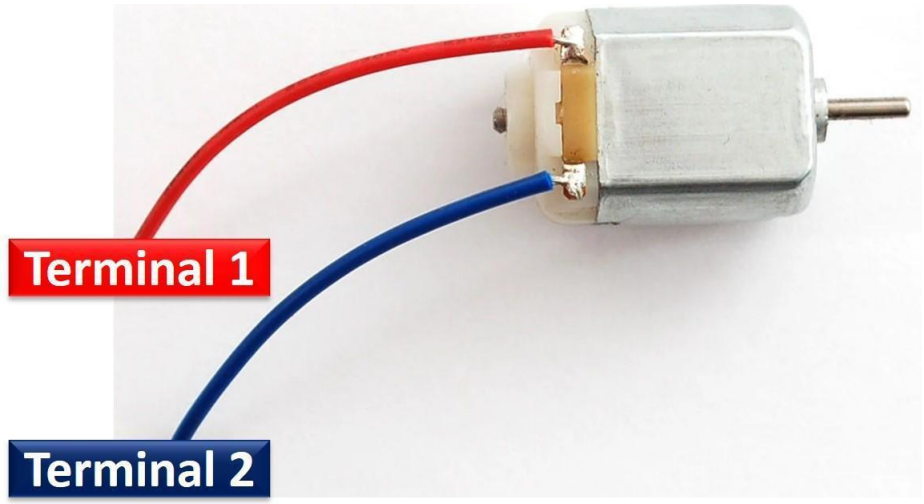


Figure 135: DC motor

The operating voltage of this toy motor is 4.5 to 9 volts. Hence, by applying a variable voltage between 4.5-9 volts, we can control its speed.

5.12.5. L298N Motor Driver Introduction

L298N motor driver shield consists of an L298 motor driver IC, 7805 voltage regulators, pinouts to provide input signals from microcontrollers, and pins to connect two DC motors.

There are two terminals two connection two DC motors.

- IN1 and IN2 are Motor_1 direction control pins of internal H-Bridge. Similarly, IN3 and IN4 are Motor_2 direction control pins of the second internal H-Bridge of the L298N motor driver.
- ENA and ENB control motor_1 and motor_2 respectively. Keeping these pin logic high will let the DC motors rotate. On the contrary, pulling these pins to an active low state will turn off the motors. But by removing jumpers, we can apply a PWM signal to each pin instead of just an ON/OFF signal.It consists of two H-Bridge circuits to drive two separate DC motors. On top of that each H-Bridge has an enable pin which is used to

provide ON/OFF signal. But we can also use Enable pins to provide PWM signal to each DC motor.

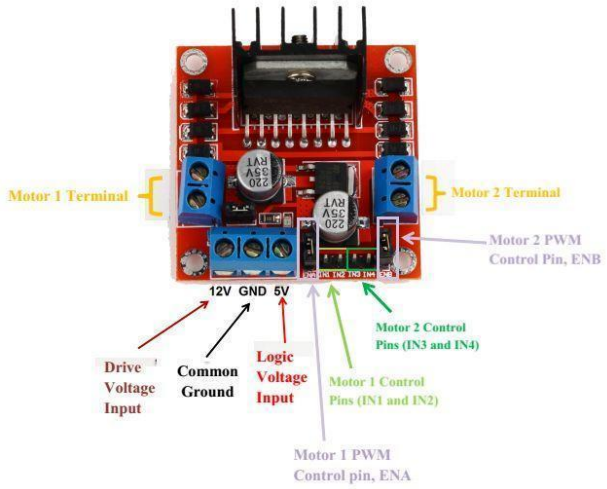


Figure 136: Motor Driver (L298N)

5.12.6. Speed Control with L298N and TM4C123

As we discussed earlier, the speed of a DC motor is directly proportional to the applied input voltage. But DC voltage is generally applied to a DC motor from a constant voltage source such as battery or adapter. Therefore, we use a special technique known as pulse width modulation to get variable output voltage from a constant DC voltage source.

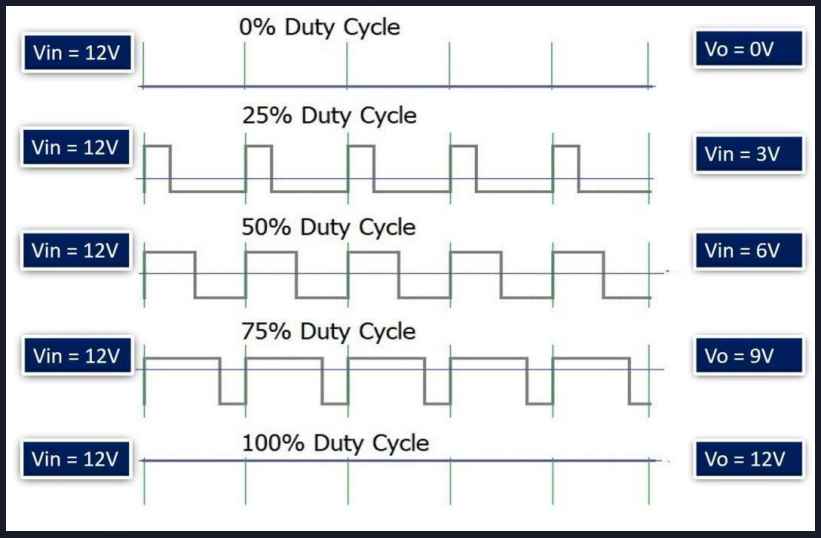
In a PWM method, we apply the DC voltage to a DC motor through a series of on-off pulses which control the voltage magnitude appearing across motor terminals. The duty cycle or on-time of these pulses determines the average voltage which appears across motor terminals. Higher the duty cycle, higher will be the average output voltage. The lower the duty cycle, the lower will be the average voltage across a motor.

The formula for calculating the duty cycle of square wave:

%Duty Cycle = (TON / (TON + TOFF)) * 100

TON is the time in which the square wave will be on.

TOFF is the time in which the square wave will be off.



The following figure depicts the relationship between output and input voltage according to the duty cycle of PWM pulses.

5.12.7. TM4C123 PWM Pins

TM4C123 Tiva Launchpad comes with a TM4C123GH6PM microcontroller. This microcontroller has 8 built-in PWM generators. These PWM generators can be used to generate variable duty cycle PWM signals. In response, we can use these variable duty cycle PWM signals from TM4C123 to get variable DC voltage and ultimately to control the speed of a DC motor.

We will feed the TM4C123 PWM signal to enable the pin (EnA or EnB) of the L298N motor provide to provide variable DC voltage to the DC motor.

Figure 137: PWM Working principle

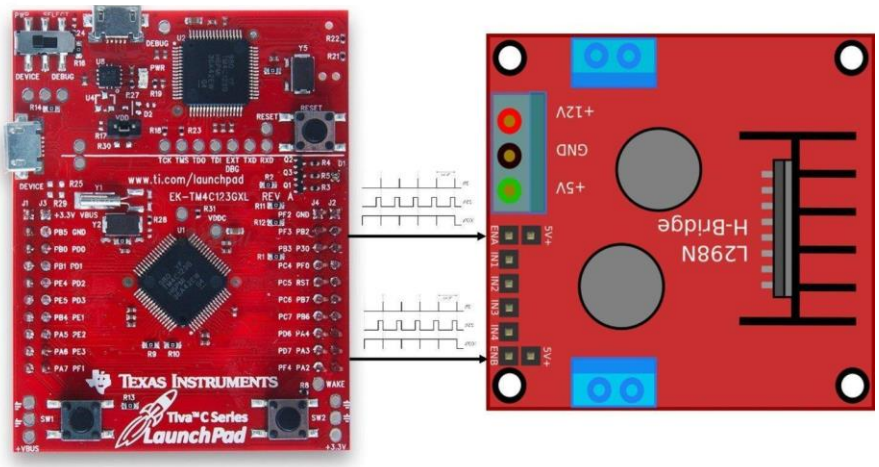


Figure 138: TivaC connection with L298N

5.12.8. DC Motor Interfacing with TM4C123 Tiva Launchpad

To interface a DC motor with TM4C123 through the L298N motor driver, we need to provide three signals to the L298N motor driver from TM4C123 GPIO pins. Two signals to control the direction and one PWM signal. To control direction, we just provide an active HIGH/LOW signal to L298N. Therefore, we can use two GPIO pins of TM4C123 as digital output pins.

- GPIO Pins TM4C123 Tiva Launchpad

As discussed earlier, TM4C123 microcontroller have 8 PWM generators. We can use any PWM generator of TM4C123 to provide a PWM signal.

5.12.9. Codes :

1 . Motor code initialization :

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB); //enable he crystal on the port b of the tiva c
while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB)) //wait until the crystal is enabled on the port
  {
  }
GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0);//pin B0 output
GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_1);//pin B1 output
GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);//pin B2 output
GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);//pin B3 output
```

By this we have enable the pin B0 ,B1,B2,B3 to be the inputs of the motor driver so we can detect the direction of the motion of the motor

2 . Pwm initialization

```
PWMGenConfigure(PWM0_BASE, PWM_GEN_1, PWM_GEN_MODE_DOWN | PWM_GEN_MODE_NO_SYNC);

//Set the Period (expressed in clock ticks)
   PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, 400);

//Set PWM duty-50% (Period /2)
 PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2,200);//PIN b4
```

```
 PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3,200);//PIN b5
 // Enable the PWM generator
   PWMGenEnable(PWM0_BASE, PWM_GEN_1);

 // Turn on the Output pins
   PWMOutputState(PWM0_BASE, PWM_OUT_2_BIT | PWM_OUT_3_BIT , true);
```

By this code we configured the pwm pins to work on pin b4 and b5 which will connect on the enable of the motor driver

3. The controlling functions of the motor

```
void forword  (void){
        while(pwmNow<400){
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_0, GPIO_PIN_0); //left side
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_1, 0x0);
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_2, GPIO_PIN_2); //right side
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_3, 0x0);
        SysCtlDelay( (SysCtlClockGet()/(3*1000))*20 ) ;
        pwmNow += 10;
   PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2,pwmNow);//PIN b4
   PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3,pwmNow);//PIN b5
}}
void backword  (void){
        while(pwmNow<400){
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_0, 0x0); //left side
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_1, GPIO_PIN_1);
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_2, 0x0); //right side
```

```
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_3, GPIO_PIN_3);
        SysCtlDelay( (SysCtlClockGet()/(3*1000))*100 ) ;
                pwmNow += 10;
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2,150);//PIN b4
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3,pwmNow);//PIN b5
        }
}
void TurnRight  (void){
        while(pwmNow<400){
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_0, GPIO_PIN_0);// left side
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_1, 0x0);
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_2, 0x0);//right side
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_3, GPIO_PIN_3);
        SysCtlDelay( (SysCtlClockGet()/(3*1000))*20 ) ;
                pwmNow += 10;
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2,pwmNow);//PIN b4
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3,pwmNow);//PIN b5
        }
}
void Turnleft  (void){
        while(pwmNow<400){
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_0, 0x0);//left side
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_1, GPIO_PIN_1);
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_2, GPIO_PIN_2);//right side
        GPIOPinWrite(GPIO_PORTB_BASE,GPIO_PIN_3, 0x0);
        SysCtlDelay( (SysCtlClockGet()/(3*1000))*20 ) ;
  pwmNow += 10;
```

```
  PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2,pwmNow);//PIN b4
  PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3,pwmNow);//PIN b5
        }
}
```

5.13. BLUETOOTH MODULE

5.13.1 Bluetooth theory of work:

Bluetooth technology is a short-range wireless communications technology to replace the cables connecting electronic devices. The Bluetooth RF transceiver (or physical layer) operates in the unlicensed ISM band centered at 2.4 GHz (the same range of frequencies used by microwaves and Wi-Fi). The core system employs a frequency-hopping transceiver to combat interference and fading.

Bluetooth's short-range transmitters are one of its biggest plus points. They use virtually no power and, because they don't travel far, are theoretically more secure than wireless networks that operate over longer ranges. Bluetooth devices automatically detect and connect to one another and up to eight of them can communicate at any one time. They don't interfere with one another because each pair of devices uses a different one of the 79 available channels. If two devices want to talk, they pick a channel randomly and, if that's already taken, randomly switch to one of the others.

When a group of two or more Bluetooth devices are sharing information together, they form a kind of ad-hoc, mini computer network called a piconet. Other devices can join or leave an existing piconet at any time. One device (known as the master) acts as the overall controller of the network, while the others (known as slaves) obey its instructions. Two or more separate piconets can also join up and share information forming what's called a scatternet.

5.13.2. HC-05 Introduction

HC-05 is a two-way full-duplex Bluetooth to serial module. In other words, it uses UART serial communication to transmit and receive data packets through a classic Bluetooth standard. Its CSR Bluecore 04-External single chip is already configured to communicate with other Bluetooth devices through serial communication.

The default baud rate set for HC-05 is 9600. But it can be programmed with AT commands and it supports 9600,19200,38400,57600,115200,230400,460800 baud rates.

Modules operate in the industrial, scientific and medical (ISM) radio bands: 2.40GHz—2.48GHz.

Operating voltage ranges from 3.3 to 6v and operating current 40 mA

can reach a range up to 9 meters.

This module can be used to perform low-cost wireless communication between microcontrollers such as Arduino, TM4C123, Pic Microcontroller, Raspberry Pi, STM32, and Arduino Mega, etc. Also, we can use it for communication between a microcontroller and PC or Android application. There are many Android applications available in the Google Play store which we can use to send and receive data to/from HC05.

5.13.3 Pinout diagram

The following figure shows the pinout diagram of the HC-05 Bluetooth module. Its output header contains six pins; namely Enable/Key, Vcc, GND, Tx, Rx, state. The description of all these pins are given in the next section.

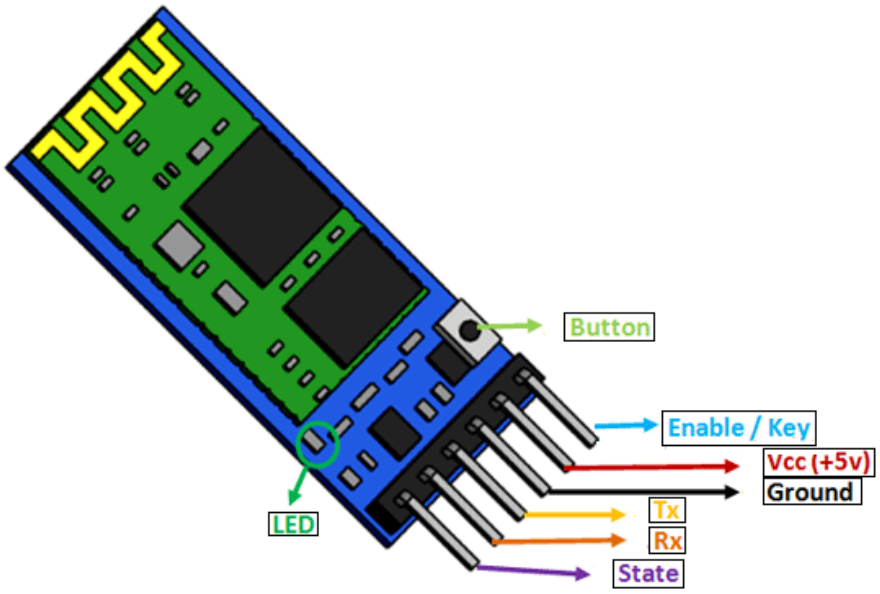


Figure 139: Bluetooth Module Pinout Diagram

5.13.4 HC-05 Interfacing with TM4C123 Tiva Launchpad

in order to communicate with the HC-05 Bluetooth module, we need to use a UART communication port of the TM4C123GH6PM microcontroller. This microcontroller has on-chip 5 UART ports such as UART0 to UART5. We can use any one of these UART ports to interface TM4C123 with the BT device.
The following table shows all UART ports of TM4C123 Tiva launchpad and their corresponding GPIO pins.

UART Module	Rx Pin	Tx Pin
UART0	PA0	PA1
UART1	PC4	PC5
UART2	PD6	PD7
UART3	PC6	PC7
UART4	PC4	PC5
UART5	PE4	PE5
UART6	PD4	PD5
UART7	PE0	PE1

Figure 140: UARTs in the TivaC

5.13.5 HC-05 and TM4C123 Connection Diagram

We will use a UART3 port. For the UART5 port, the PC6 GPIO pin is a receiver pin (Rx) and the PC7 pin is a transmitter pin (Tx).

Now make connections with HC-05 Bluetooth module and Tiva Launchpad according to this schematic diagram.

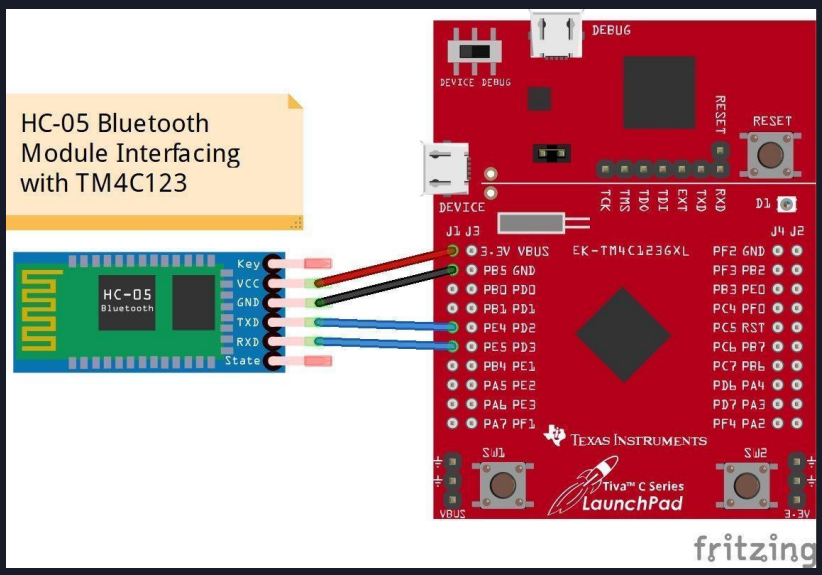


Figure 141: Bluetooth module connection with the TivaC

HC-05	TM4C123
Tx	PC7 (Rx3)
Rx	PC6(Tx3)
VCC	3.3 volts
GND	GND

We use the UART3_Receiver() and UART3_Transmitter() functions
The UART3_Receiver() routine is used to read data which will receive on PC6(Rx3) pin of TM4C123 and the UART3_Transmitter() routine is used to transmit data serially using PC7(Tx3) pin of Tiva Launchpad.

5.13.6 Uart code initialization

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC); // this will enable the crystal for the port c

GPIOPinConfigure(GPIO_PC6_U3RX);    // will configure the pins oc6 to work with its alternative function

GPIOPinConfigure(GPIO_PC7_U3TX);    //configure pc7 to work with its alternative function TX
```

```
GPIOPinTypeUART(GPIO_PORTC_BASE,GPIO_PIN_6|GPIO_PIN_7);  //will enable uart 3 pins

SysCtlPeripheralEnable(SYSCTL_PERIPH_UART3);    //will enable the crystals to pulse for uart 3

UARTConfigSetExpClk(UART3_BASE,SysCtlClockGet(),9600,(UART_CONFIG_WLEN_8 | UART_CONFIG_PAR_NONE | UART_CONFIG_STOP_ONE));

UARTEnable(UART3_BASE); // enable the uart 3 to work
```

5.13.7. HC-05 and Andriod App

Android Application

To communicate between Android mobile and HC-05 Bluetooth modules which will be interfaced with TM4C123 Tiva Launchpad, we will need an Android application. There are many Bluetooth terminal Android applications available in google play store. For this tutorial, we will use "Bluetooth RC Controller" Android application.

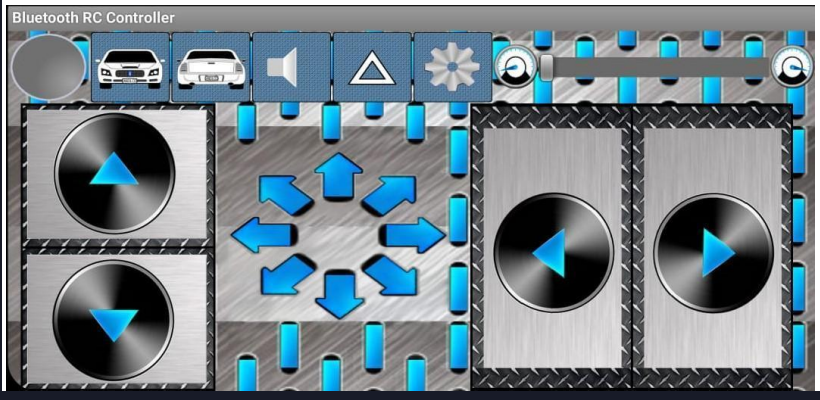


Figure 142: RC car Bluetooth android app

The application used for this is an Android based application. The app is easily available on the internet and there are many other applications like this available on the internet.

The app uses the interface GUI to convert your choice command as into corresponding string which is read using UART(Universal Asynchronous Receiver Transmitter) circuit in the Tiva micro-controller

The uart receives the string and the code is executed on the micro-controller accordingly.

In our project, we used the pins of the Tiva board as GPIO pins, controlling the operation of the motors through the variation of logic levels on the pins.

For the Bluetooth module, the pins were configured for alternate functionality as UART pins. In order to vary the speed of the motors, the pins were configured in the alternate functionality as Timer pins and the PWM mode was used. The speed was controlled by varying the duty cycle of the input clock cycle.

Commands used to control motors of Tiva Launchpad are listed below:

And these are the commands for speed control

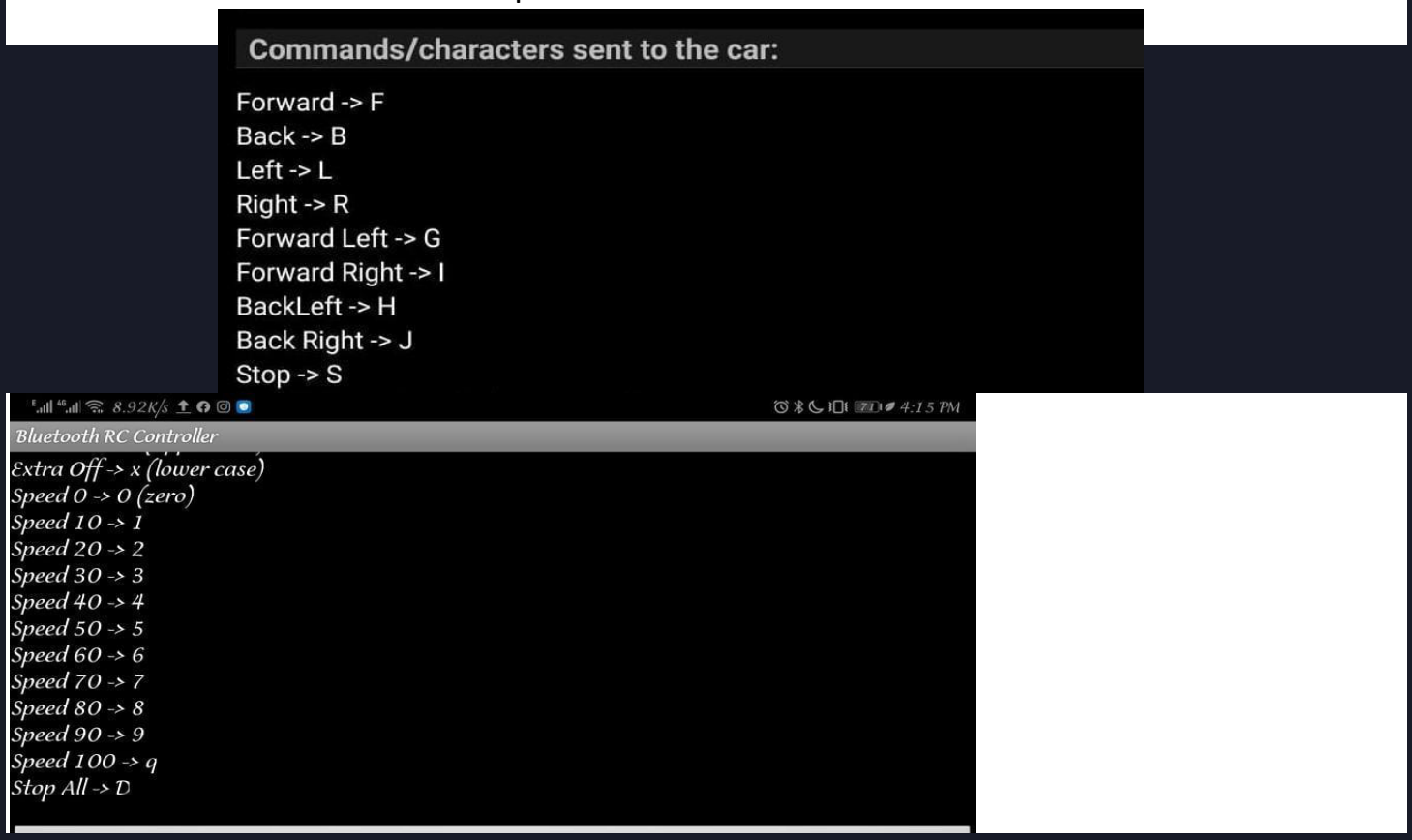


Figure 143: Speed control commands

So, first of all we connect our bluetooth module to the android application

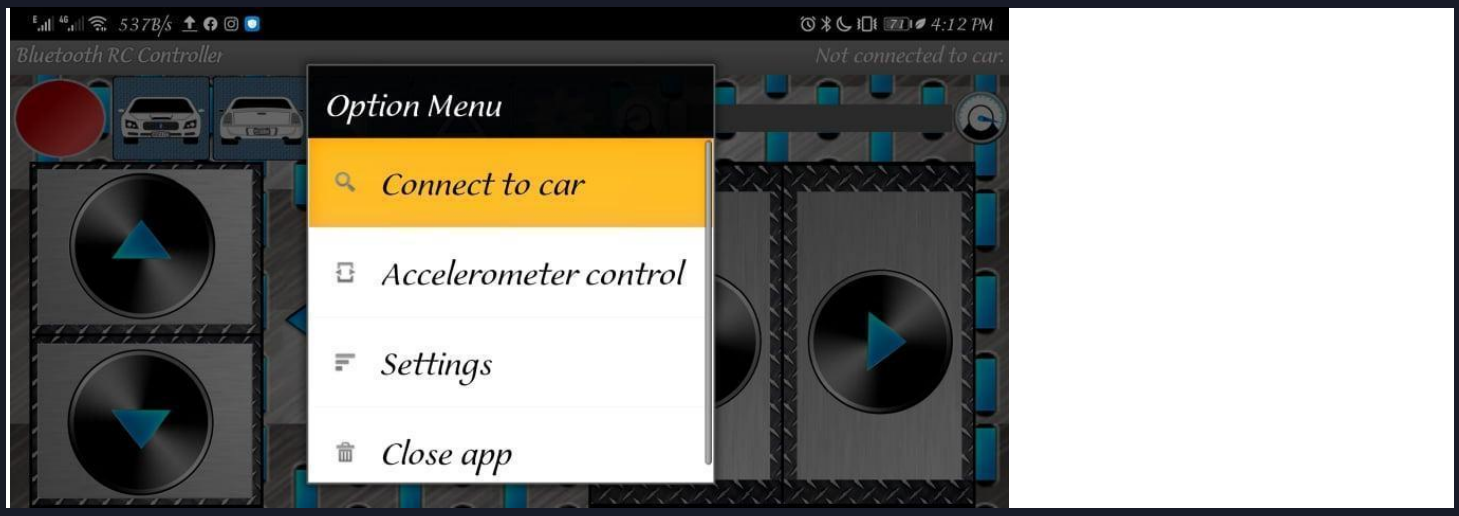


Figure 144: connecting the Bluetooth app to the car remotely

-Scanning for devices and pairing.

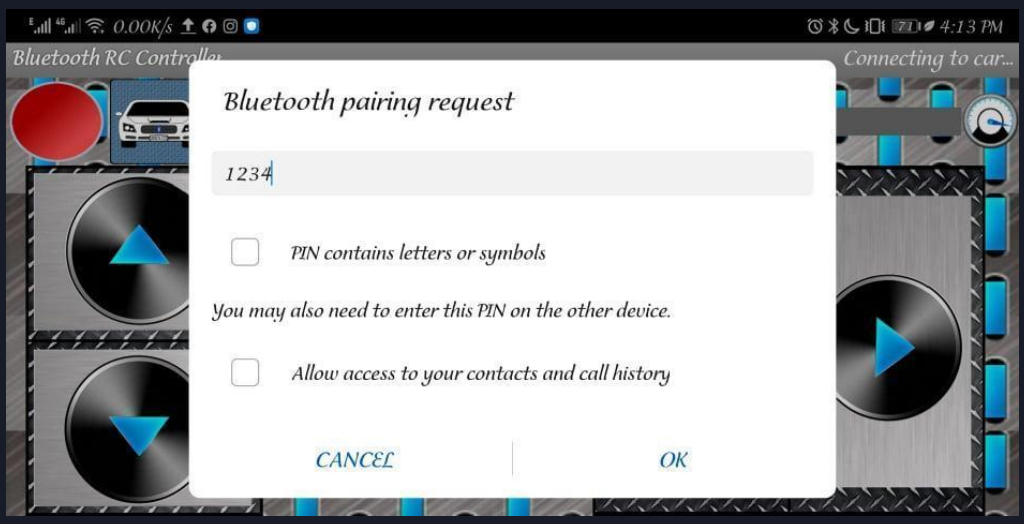


Figure 145: Bluetooth app pairing with the car

4.27. Connected with module.

Figure 146: car connected with the RC Bluetooth app

Now try press and button (Forward/Reverse/Right/Left), you will see the car moving as u pressed.

5.14. LCD

5.14.1. LCD Theory of work

A liquid-crystal display (LCD) is a flat panel display, electronic visual display, or video display that uses the light modulating properties of liquid crystals. Liquid crystals do not emit light directly. 20x4 means that 20 characters can be displayed in each of the 4 rows of the 20x4 LCD, thus a total of 80 characters can be displayed at any instance of time.

LCD accepts two types of signals, one is data, and another is control. These signals are recognized by the LCD module from status of the RS pin. Now data can be read also from the LCD display, by pulling the R/W pin high. As soon as the E pin is pulsed, LCD display reads data at the falling edge of the pulse and executes it, same for the case of transmission. There are two types character LCD and graphical LCD.

5.14.2. About the module

We used monochromatic 20x4 alphanumeric LCD.

- 5x8 dots
- +5V power supply
- 1/16 duty cycle
- LED Backlight

Figure 147: LCD (4 x 16)

5.14.3. -Pins Configuration

Data pins D7-D0 : Bi-directional data /command pins alphanumeric characters are sent in ASCII format

RS (register select)

RS =0 (command register is selected) RS =1 (data register is selected)

R/W (Read or Write) 0 (Write) /1(Read)

E (Enable data)

Used to enable the LCD display V0 (contrast control)

For maximum brightness V0=0

-Display Data RAM (DDRAM) Addresses

- Display Data RAM (DDRAM) Addresses inside LCD stores display data represented in 8-bit character codes.
- Its extended capacity is 80*8 bits or 80 characters

Pin No.	Pin Name	Descriptions
1	VSS	Ground
2	VDD	Supply voltage for logic
3	V0	Input voltage for LCD
4	RS	H : Data signal, L Instruction signal
5	R/W	H : Read mode, L : Write mode
6	E	Chip enable signal
7	DB0	Data bit 0
8	DB1	Data bit 1
9	DB2	Data bit 2
10	DB3	Data bit 3
11	DB4	Data bit 4
12	DB5	Data bit 5
13	DB6	Data bit 6
14	DB7	Data bit 7
15	LED_A	Backlight Anode
16	LED_K	Backlight Cathode

- The area in Display Data RAM (DDRAM) that is not used for display can be used as general data RAM.

- So whatever you send on the DDRAM is actually displayed on the LCD.

		COLUMN										
		1st	2nd	3rd	4th	5th	. . . . .	16th	17th	18th	19th	20th
ROW	1st	00h	01h	02h	03h	04h	. . . . .	0Fh	10h	11h	12h	13h
	2nd	40h	41h	42h	43h	44h	20 x 4 Characters (5x8 dots font)	4Fh	50h	51h	52h	53h
	3rd	14h	15h	16h	17h	18h		23h	24h	25h	26h	27h
	4th	54h	55h	56h	57h	58h	. . . . .	63h	64h	65h	66h	67h

DDRAM Address Map

Figure 149: LCD DDRAM Address Map

- GROM Memory

Character Generator ROM memory contains a standard character map all characters that can be displayed on the screen .Each character is assigned to one memory location.

- LCD commands

Command	Function
0F	LCD ON, Cursor blinking ON, Cursor ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Cursor blinking OFF, Display ON
80	Force cursor to the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
08	Display OFF, Cursor OFF
83	Cursor line 1 position 3
3C	Activate second line
38	Use 2 lines and 5x7 matrix
C1	Jump to second line, position1
OC	Cursor OFF, Display ON
C2	Jump to second line, position2

Figure 150: LCD commands

5.14.4. Contribution

At first we interfaced the LCD with controller in 8 it mode, which caused problems later as the number of modules used increased.

To use LCD some of the above commands must be sent for initialization before sending data, so we implemented initialization function and a function to clear

LCD screen. For better code organization we have 2 independent functions one to print string variables, other to print numeric variables. Besides a function to set the cursor position to start writing at desired location.

Optimization

In order to decrease number of GPIO pins used to interface between LCD and controller we used LDC in 4 bit mode in which data is sent in two cycles.

6

Printed Circuit Board (PCB) Design

6.1. Introduction

Although our hardware system implementation is modular and has the least number of interface connections between devices, yet it still had large number of wires. And with the many wires and breadboard connection, it was very difficult to handle the system and keep it neat and reliable, it was also very difficult to detect a bad connection or misplaces wires. Besides, the area of connection increased, Hence, it was not a practical method.

To improve compatibility and reliability, the PCB design and implementation had to be held to achieve so and solve the other mentioned Problems.
With the proper PCB design, we could achieve:

- **Compact design**

  al PCB contains many electronic components, most of which are very small in size. Without a PCB, it would be nearly impossible to connect such components together with wires. A PCB provides a convenient platform to arrange the electronic components in a compact and efficient way. This compactness allows development of large and complicated electronic circuits in small form factors, taking less space in devices.
- **Good fixation**

  Since components on a PCB are held fixed to the board by solder flux, they do not move, irrespective of the movement of the board. This enables the electronic circuit to be placed in devices that are moving or shaking without worrying about the possibility of component displacement and subsequent electronic short circuits.
- **Ease in Diagnostics and Repair**

  PCBs are helpful in performing diagnostics for several reasons. The components and their polarities on a well-designed PCB are clearly labelled on the board, which is convenient for installation as well as repair. For diagnostics, one often needs to trace signal paths, which would be very difficult to perform if the traces were not exposed and well organized.
- **Low Electronic Noise**

  When properly laid out, a PCB minimizes electronics noise that could significantly degrade performance. The electrical components on a PCB are organized in such a way that the path lengths of the electrical current between them are minimized, leading to low radiation and pickup of electromagnetic waves. This ensures lower crosstalk between components and between different traces, which is a major concern in electronic circuits.

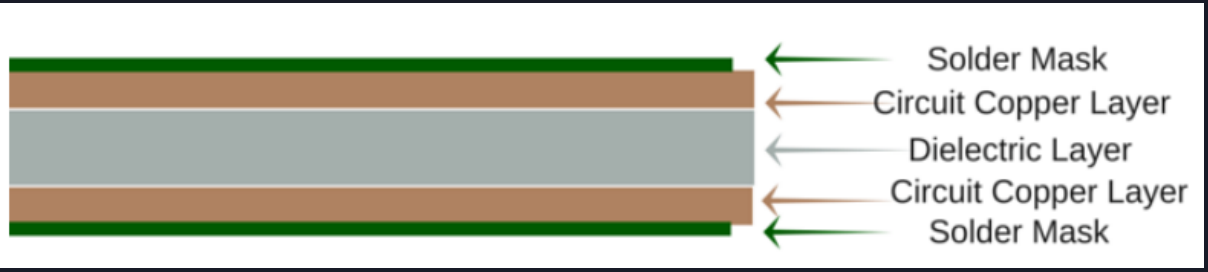


Figure 151: PCB layers

6.2. PCB design software tool

Among many PCB design software tools, we chose Altium designer as it is more flexible, professional, and it can provide 3D modelling.

6.2.1. Altium designer

Figure 152: Altium Designer

Altium Designer is a PCB and electronic design automation software package for printed circuit boards. Altium Designer's suite encompasses four main functional areas: schematic capture, 3D PCB design, Field-programmable gate array (FPGA) development and release/data management. Altium makes designing complex, high-quality projects easier, faster, and more accurate from concept to production.

The strength of Altium's design tools begins with their foundation, a solid platform built on a unified design environment. Another core strength of Altium that it provides the most useful library and up-to-date user interface for design workflow. Both interactive and automatic, routing tools help to finish routing in less time and with better results.

It gives complete error checking and design verification. The best rules driven design, as well as the ability to verify all aspects of your design including your circuit board's power delivery add speed to the workflow.

The version we used was V14.3

6.2.2. System PCB requirements

To have more neat hardware design and easy troubleshooting procedures, we decided to design two separate PCBs:

1- Controller interfacing board

2- System power supply board

Design steps

1- Requirements and features
2- Board size limitations
3- Component select
4- Circuit Schematic
5- PCB layout
6- PCB fabrication

6.2.3. Controller interfacing board

1. Requirements and features

As our hardware system is modular, the main functionality of this PCB is to interface modules, sensors, actuators, and communication links with each other and with the main controller of the system (The Tiva C).

2. Board size limitation

To have a good fixation of the board on the robot car chassis, the board size could not exceed a certain length and width of the chassis.
So, the board sized was selected to be: (8cm width * 15cm length)

3. Component select

Component	count	Function
Tiva C (TM4C123G Launch pad)	1	The main controller
NodeMCU ESP8266	1	WI-FI connection with server
Bluetooth module (HC-05)	1	Bluetooth module connection
GPS (4-Pin header)	1	Interfacing with GPS module
Speed Sensor (4-Pin header)	1	Interfacing with speed sensor
Ultrasonic (4-Pin header)	1	Interfacing with Ultrasonic
LCD (16-Pin header)	1	Interfacing with the LCD
L289 Motor driver (6-Pin header)	1	Interfacing with motor driver
2-pin Rosetta	1	Input power (5V)
Raspberry pi UART (3-pin header)	1	Serial communication between Tiva C and Raspberry Pi

4. Circuit Schematic

The following screenshot shows the complete board schematic with all components and connections are labelled

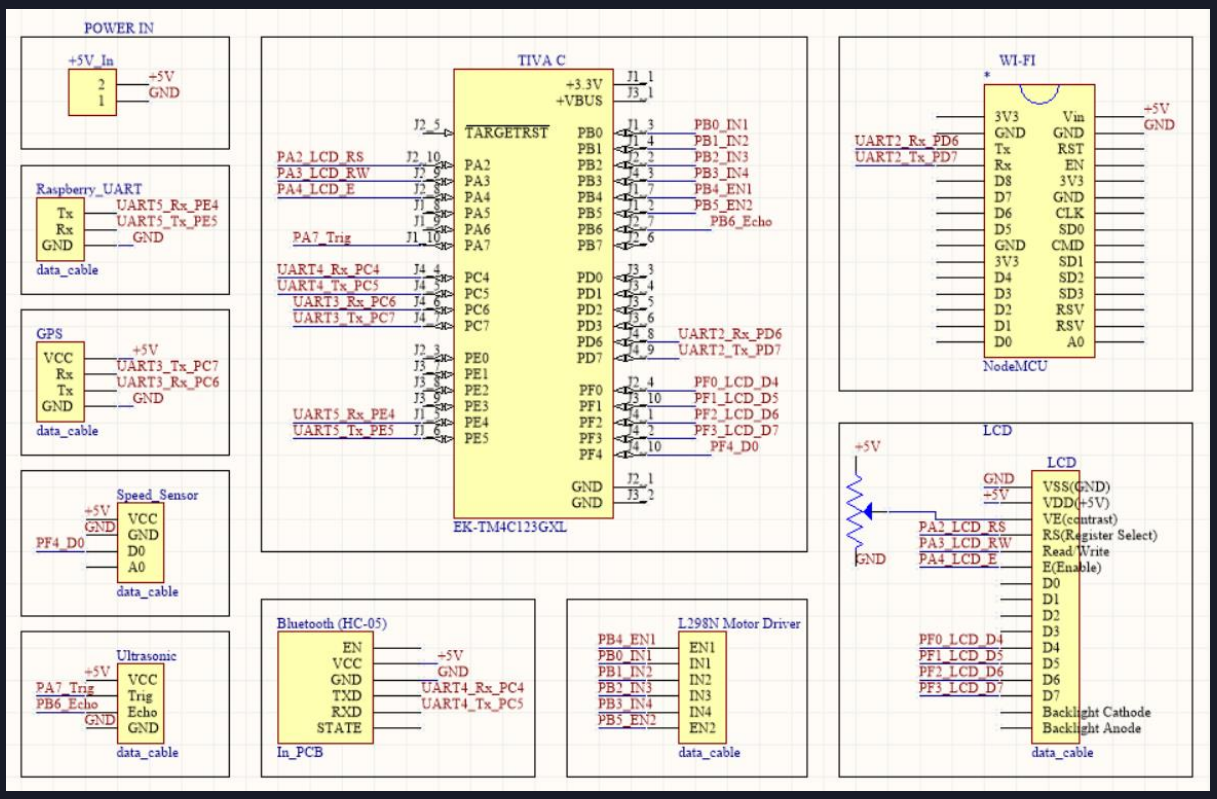


Figure 153: PCB circuit schematic

5. PCB layout and 3D modelling

The process of PCB layout includes Placing component, defining board size, and routing the tracks with the proper width and clearance values.

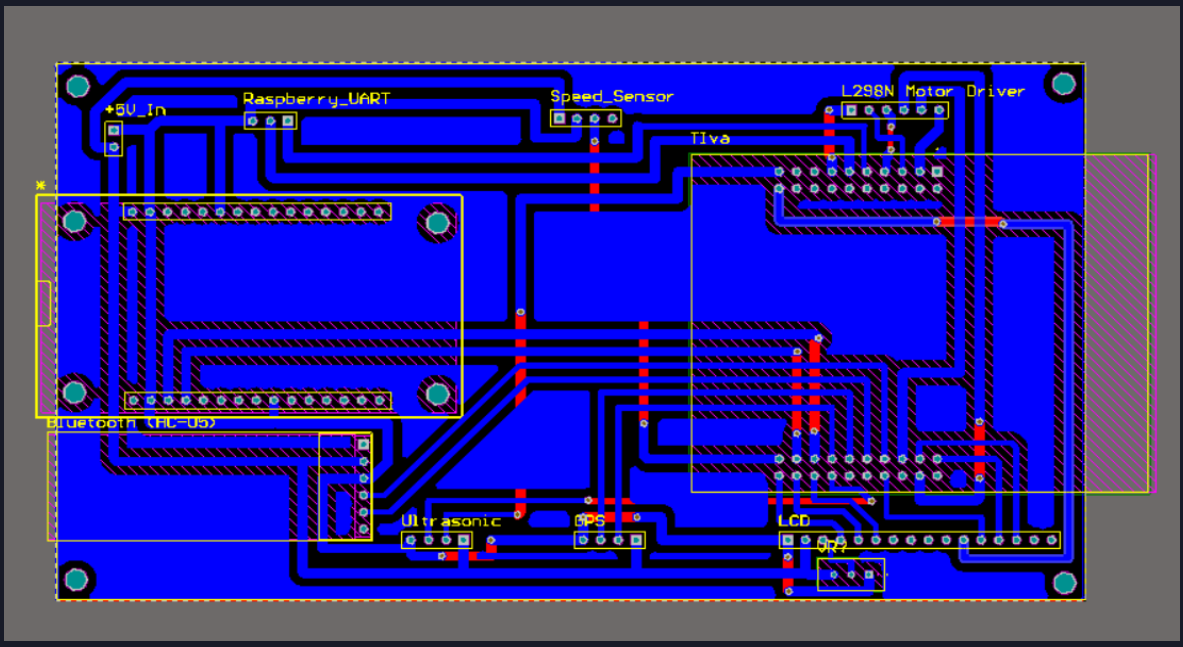


Figure 154: PCB layout and 3D modelling (Back)

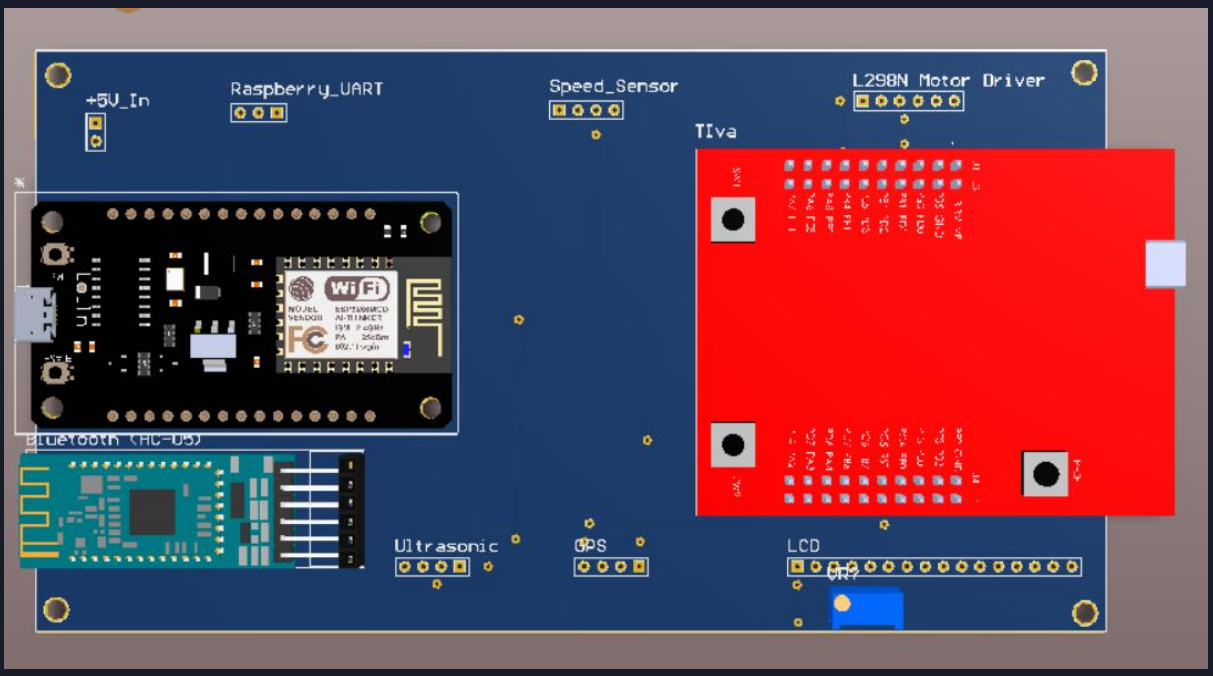


Figure 155: PCB layout and 3D modelling (Front)

6.2.3. SYSTEM POWER supply board

1. Requirements and Features

After considering the ideal system operating supply voltages, and considering the driven current of each component in the system, along with safety current supply margin, a power supply PCB is to be designed to:

- Voltage supply regulation
- Easy plug-in connections
- Overcurrent fuse protection

Hence the robot is portable and completely wireless, we used batteries as our main power source.

Figure 156: USB high voltage supplier

And as mentioned before, we used three lithium-ion batteries in series (3.7V each) providing total of 11.1V, this is this 11.1V cannot be supplied directly to the system, because each module has its own operating voltage range that cannot be exceeded.

USB Tester

For precise operating current monitoring, we used USB tester to make sure we are designing the system based on the real power values

The following table shows the system components and the operating voltage and current values of each.

USB tester tool

Power calculation at 5V operating voltage:

Component	Maximum Driven current
Tiva C (TM4C123G Launch pad)	200mA
NodeMCU ESP8266	60mA
Bluetooth module (HC-05)	40mA
GPS	60mA

Speed Sensor	30mA
Ultrasonic	40mA
LCD	20mA
L289 Motor driver (6-Pin header)	60mA
Raspberry pi	1A
Total	**1510mA**

With adding Safety factor 125%: 1.25*1510 = 1887.5mA

Safety fuse value = 2A

2. Board size limitations

For compact design and good fixation of the board the size chosen was as small as possible which was (5cm*5cm)

3. Component select

The following table illustrates the component list and functionality of each.

Component	count	Function
DC jack Female	1	Input power connector
Fuse	1	Overcurrent protection
Buck converter (3A)	1	Voltage regulator
Rosetta (Motor Power)	1	Output power connector
USB Female	2	Output power connector for Raspberry Pi and Tiva C
Pin header	2	Output power to the other board

4. Circuit schematic

The following screenshot shows the complete board schematic with all components and connections are labelled

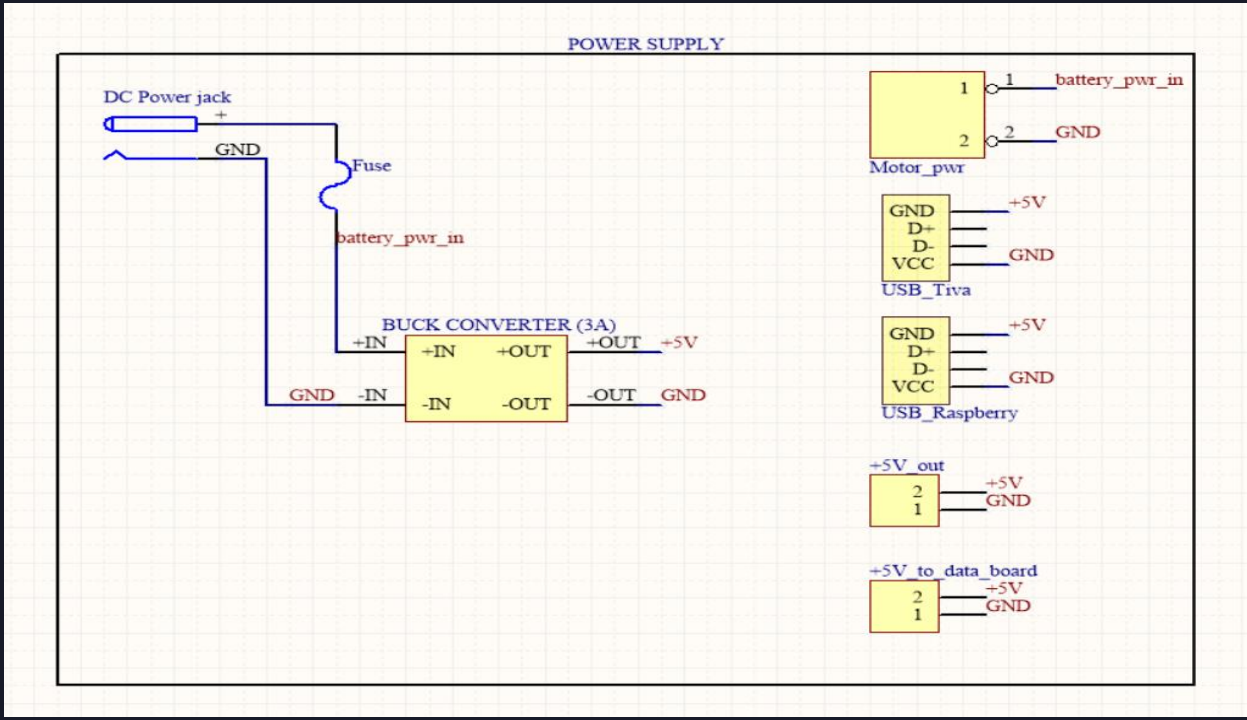


Figure 157: PCB power supply circuit schematic

5. PCB layout and 3D modelling

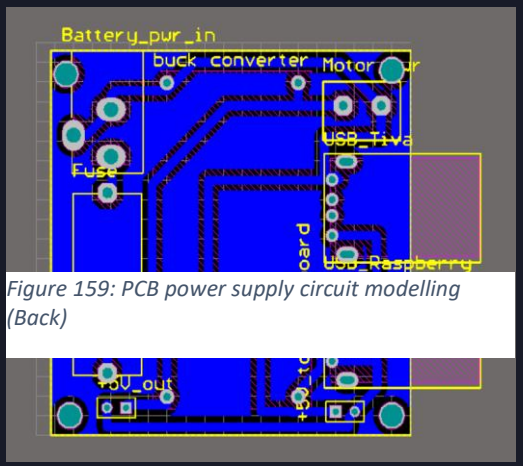


Figure 159: PCB power supply circuit modelling (Back)

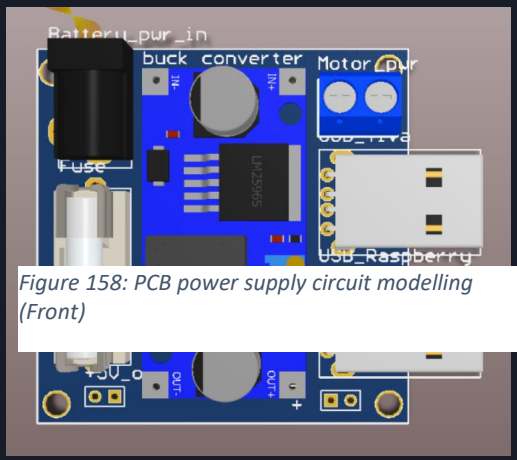


Figure 158: PCB power supply circuit modelling (Front)

7

Future Improvements and Costs

7. Future Improvements:

7.1. Computer Vision:

In the computer vision part, where we converted our model from a normal TensorFlow model into a TensorFlow Lite model then quantized it in order to increase the object-detection process performance (lower power consumption, higher accuracy, fast detection). But with all of these optimizations, our detection speed could hardly reach 1.5 FPS maximum, which is still so bad !!

So, there are some improvements, which should make our model better in detection whether in speed or accuracy.

Here, we will discuss three options to improve our model-detection performance:

7.1.1. CanaKit Raspberry Pi 4 8GB Starter Kit - 8GB RAM

Figure 160: CanaKit Raspberry Pi 4 8GB Starter Kit - 8GB RAM

Does the type of the Raspberry Pi affect the performance ?

Of course, yes. A Raspberry Pi is a hardware that is treated like the normal PC. Means that the more the (Storage size – RAM), the better the performance.

Features of this kit package:

- Includes Raspberry Pi 4 8GB Model B with 1.5GHz 64-bit quad-core CPU (8GB RAM)

  “We used a Raspberry Pi 4 2GB RAM instead of 8GB RAM as it was the available one”

- 32GB Samsung EVO+ Micro SD Card (Class 10) Pre-loaded with NOOBS, USB MicroSD Card Reader

  “We used a Kingston SD card instead, but this won’t affect the performance”

- CanaKit Premium High-Gloss Raspberry Pi 4 Case with Integrated Fan Mount, CanaKit Low Noise Bearing System Fan

  “Fans are added to the Raspberry Pi in order to cool down the Raspberry Pi while it’s working (This increases the performance and assures that the speed of detection process won’t get reduced along the time)”

- CanaKit 3.5A USB-C Raspberry Pi 4 Power Supply with Noise Filter, Set of Heat Sinks, Micro HDMI to HDMI Cable - 6 foot (Supports up to 4K 60p)

  “The Set of Heat Sinks are used with the Fans for a more cooling for the hardware to assure the best performance for a long time”

- CanaKit USB-C Pi Switch (On/Off Power Switch for Raspberry Pi 4)

Link: https://www.amazon.com/gp/product/B07V2B4W63

7.1.2. Coral USB Accelerator

Figure 161: Coral USB Accelerator

A USB accessory that brings machine learning inferencing to existing systems. Works with Linux, Mac, and Windows systems.

- **Description**

The Coral USB Accelerator adds an Edge TPU coprocessor to your system, enabling high-speed machine learning inferencing on a wide range of systems, simply by connecting it to a USB port.

- Performs high-speed ML inferencing

The on-board Edge TPU coprocessor is capable of performing 4 trillion operations (tera-operations) per second (TOPS), using 0.5 watts for each TOPS (2 TOPS per watt). For example, it can execute state-of-the-art mobile vision models such as MobileNet v2 at almost 400 FPS, in a power efficient manner.

- Supports all major platforms

Connects via USB to any system running Debian Linux (including Raspberry Pi), macOS, or Windows 10.

- Supports TensorFlow Lite

No need to build models from the ground up. TensorFlow Lite models can be compiled to run on the Edge TPU.

- Supports AutoML Vision Edge

Easily build and deploy fast, high-accuracy custom image classification models to your device with AutoML Vision Edge.

➢ **Tech specs**

ML accelerator	Google Edge TPU coprocessor: 4 TOPS (int8); 2 TOPS per watt
Connector	USB 3.0 Type-C* (data/power)
Dimensions	65 mm x 30 mm
Availability	Australia, European Union (except France, Czech Republic), Ghana, Hong Kong, India, Indonesia, Israel, Japan, Kenya, Malaysia, New Zealand, Oman, Philippines, Singapore, South Korea, Taiwan, Thailand, United States, Vietnam

* Compatible with USB 2.0 but inferencing speed is slower.

Figure 162: Coral USB Accelerator Specs

Link: https://www.amazon.com/gp/product/B07S214S5Y

7.1.3. Logitech HD Pro Webcam C920, Widescreen Video Calling and Recording, 1080p Camera, Desktop or Laptop Webcam

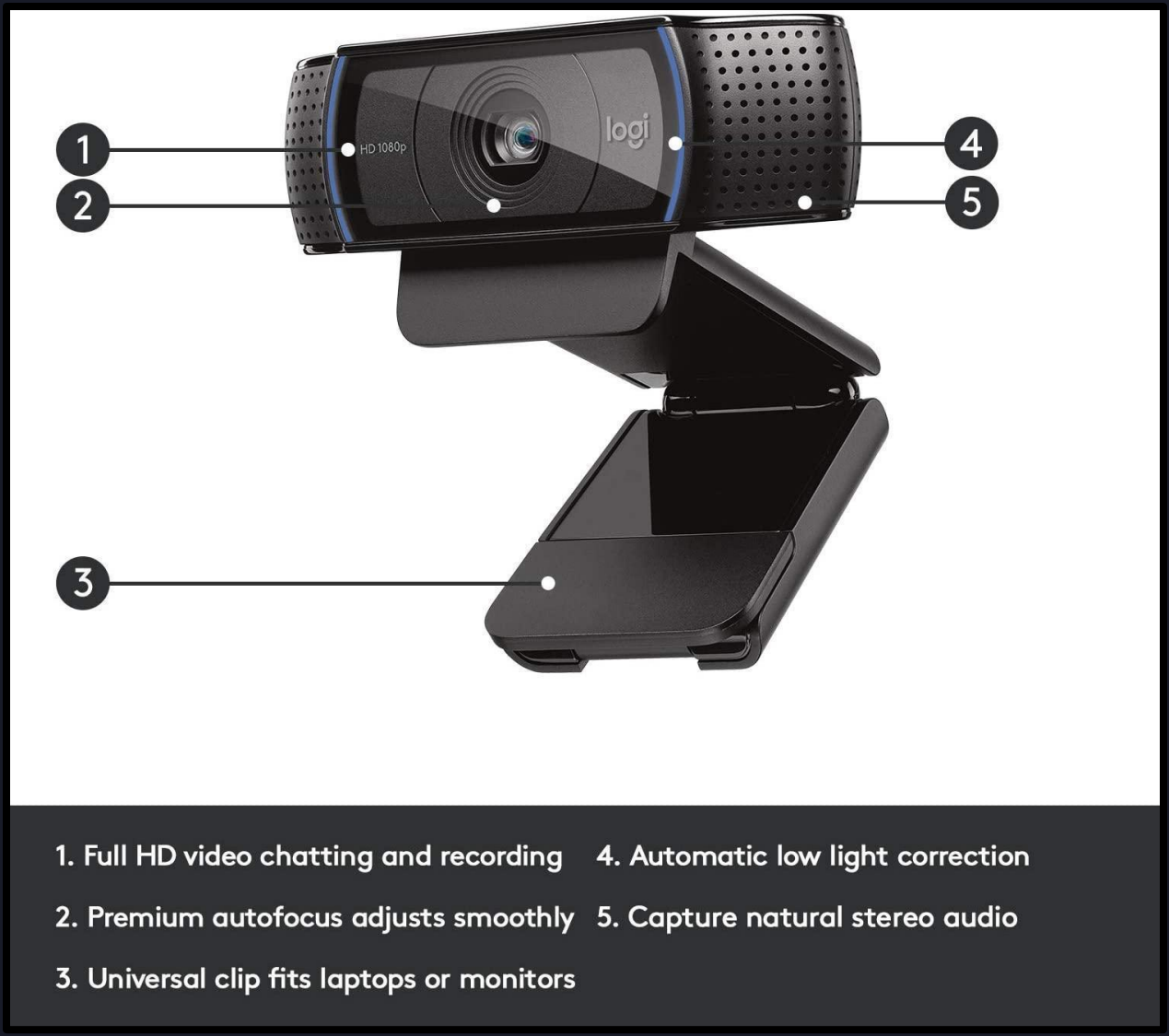


Figure 163: Logitech HD Pro Webcam C920

Features:

- Full HD 1080p video calling (up to 1920 x 1080 pixels) with the latest version of Skype for Windows; Webcam with 5-foot cable
- 720p HD video calling (up to 1280 x 720 pixels) with supported clients; Full HD video recording (up to 1920 x 1080 pixels). Max Resolution: 1080p/30fps 720p/30fps
- Video compression, Built in dual stereo mics with automatic noise reduction; Automatic low light correction, Tripod ready universal clip fits laptops, LCD or monitors

- Compatible with: Windows 7, Windows 8, Windows 10 or later; Works in USB Video Device Class (UVC) mode: Mac OS 10.10 or later (HD 720p on FaceTime for Mac or other supported video

calling clients; Full HD 1080p video recording with QuickTime Player) Chrome OS, Android v 5.0 or above (with supported video calling clients),USB port, Internet connection

This can triple the FPS compared to the normal Raspberry Pi camera.

Link: https://www.amazon.com/gp/product/B006JH8T3S

7.2. FPS comparison:

The figure below shows a brief comparison between the Raspberry Pi 3 & the Raspberry Pi 4 with various models (TensorFlow vs TensorFlow Lite vs TensorFlow Lite accelerated by the Edge TPU)

“Note that: this comparison is based on the Logitech Webcam and MobileNet V2”

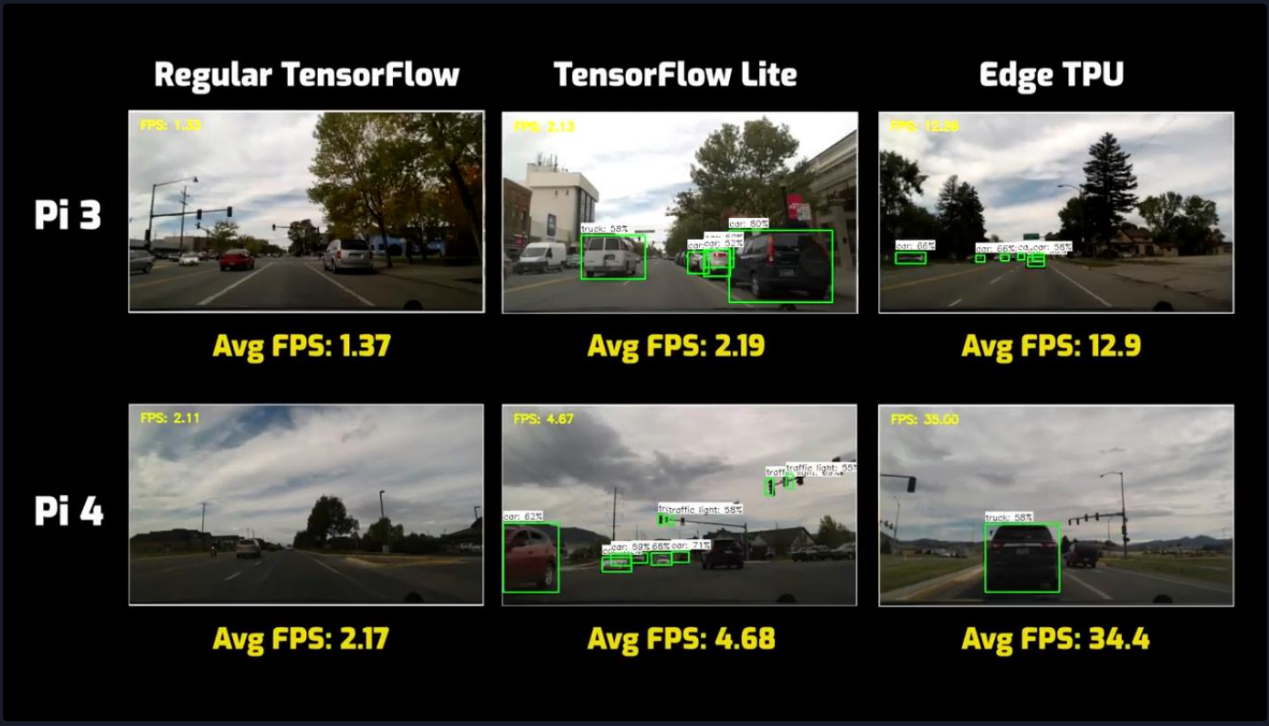


Figure 164: FPS comparison

Here, we can notice that the average FPS of our model should be 4.68. but due to using the normal Raspberry Pi camera, the FPS reached only 1.5 (≅ 4.68 / 3).

7.3. Cost and price

Component	**price**
ultrasonic X6	210
lcd size (20x 4)	120
speed sensor	45
car body (4 wheels) +platform	290
2 h bridge	110
memmory &capble	110
cable type c	40
testing ground and its components	**230**
Wifi module (offer)	55
Battery x4	100
ARM® Cortex®-M4F Based MCU TM4C123G Launchpad	600
Fidi module usb to ttlx2	120
batterycase	20
battery charger	35
holders *6	90
Raspberry pi controller	1550
RP Camera	290
ESP8266 WIFI module x2	250
Ublox NEO-6mV2 GPS Module	320
Total	4535

8

References

8. Reference

1) WIFI

- ESP8266EX, wifi module datasheet (2018) chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
- ESP8266 AT Instruction Set (2018) chrome-extension://oemmndcbldboiebfnladdacbdfmadadm/https://ieee-sensors.org/wp-content/uploads/2018/05/4a-esp8266_at_instruction_set_en.pdf
- Texas Instruments support (2019) https://e2e.ti.com/support/microcontrollers/other/f/908/p/798412/2954736#2954736

2) GPS

- Global Positioning System (2019) https://en.wikipedia.org/wiki/Global_Positioning_System
- World Geodetic System (2019) https://en.wikipedia.org/wiki/World_Geodetic_System
- NEO-6 seriesVersatile u-blox 6 GPS modules (2019) https://www.u-blox.com/sites/default/files/products/documents/NEO-6_ProductSummary_%28GPS.G6-HW-09003%29.pdf
- u-blox 6 GPS ModulesData Sheet (2019) https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6- HW-09005%29.pdf
- u-centerGNSS evaluation software Product summary (2019) https://www.u-blox.com/sites/default/files/u-center_ProductSummary_%28UBX-13003929%29.pdf
- u-centerGNSS evaluation software for Windows, User Guide (2019) https://www.u-blox.com/sites/default/files/u-center_UserGuide_%28UBX-13005250%29.pdf
- Stanly, How to change U-Blox GPS default baud rate and data rate (2015) https://freematics.com/forum/viewtopic.php?f=11&t=1759&sid=3e9b9a9f158afa2a0fce64f7b82d e273
- Handheld Group, Enable/Disable NMEA dataset's (2018) https://www.handheldgroup.com/support-rugged-computers/knowledgebase-KB/24331/?fbclid=IwAR0nDd0E_SjwHyXis39oVSTOETtf4w1GsNydBEmU4nsE_m9lzhCuUY LP6-8
- Distance Between Coordinates Theory https://www.movable-type.co.uk/scripts/latlong.html
- Distance Between Coordinates Calculator (2019) https://gps-coordinates.org/distance-between-coordinates.php?fbclid=IwAR1jl8zPtho8G6MrTzghCOecO859HTLVc-MCzoDI4KLngQ98uEIwGwprP2o

3) Ultrasonic

- Science Direct, Ultrasonic sensors (2019), https://www.sciencedirect.com/topics/engineering/ultrasonic-sensors
- Electronic Wings, Sensors & Modules, Ultrasonic Module HC-SR04 (2019), https://www.electronicwings.com/sensors-modules/ultrasonic-module-hc-sr04

- Introductory guide to sensors, Ultrasonic sensors (2019), https://www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/
- PC Services, Circuit Diagram Ultrasonic Distance Sensor HC-SR04 (2018), http://www.pcserviceselectronics.co.uk/arduino/Ultrasonic/electronics.php#faults
- Luis Rafael, Google Sites, GPIO timers (2014), https://sites.google.com/site/luiselectronicprojects/tutorials/tiva-tutorials/tiva-general-purpose-timers/timer-periodic-mode---srf04
- Cytron Technologies, "Product User's Manual – HCSR04 Ultrasonic Sensor" (2013), http://raspoid.com/download/datasheet/HCSR04

4) Bluetooth

- Scientific American, How does Bluetooth work? (2019), https://www.scientificamerican.com/article/experts-how-does-bluetooth-work/
- Components 101, HC-06 Bluetooth module (2019), https://components101.com/wireless/hc-06-bluetooth-module-pinout-datasheet
- e-Gizmo Mechatronix Central, Bluetooth Modules, "Hardware Manual & AT Commands Reference Manual Rev. 1r0" (2019), https://cdn.instructables.com/ORIG/FQ1/CUVZ/HXA9PUVQ/FQ1CUVZHXA9PUVQ.pdf
- Module143, HC06 and HC05 (Bluetooth Module) (2019), http://invent.module143.com/hc06-and-hc05-bluetooth-module-how-to-use-it/
- Instructables Circuits, "HC-03/05 Embedded Bluetooth Serial Communication Module AT command set" (2014) , https://cdn.instructables.com/ORIG/FKY/Z0UT/HX7OYY7I/FKYZ0UTHX7OYY7I.pdf

5) Motors and H-bridge

- Electrical4u, Operating principle of DC motor(2019), https://www.electrical4u.com/working-or-operating-principle-of-dc-motor/
- Roland Pelayo, Microcontroller tutorials, How to Use L298N Motor Driver (2019), https://www.teachmemicro.com/use-l298n-motor-driver/
- All about circuits, Pulse width modulation (2019), https://www.allaboutcircuits.com/textbook/semiconductors/chpt-11/pulse-width-modulation/
- STMicroelectronics, "DUAL FULL-BRIDGE DRIVER-L298"(2019), https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf
- Fairchild Semiconductor,"74125 Quad 3-STATE Buffer" (2019), http://www.datasheetcatalog.com/datasheets_pdf/7/4/1/2/74125.shtml

6) Altium-PCB

- PCB way, Advantages of double sided PCB (2019), https://www.pcbway.com/blog/Engineering_Technical/Advantages_of_Double_Sided_Pcb.html
- Amitron, Double sided PCB (2019), https://www.amitroncorp.com/printed-circuit-boards/double-sided.html
- Robert Ferance, Fedvel academy (2011),

https://www.fedevel.com/welldoneblog/2011/12/5x-why-to-choose-altium-designer-for- electronic-hardware-design/

7) lcd

- Lumisense Technologies, Chennai,"20*4 LCD" (2019), http://www.lumisense.in/pdfs/20-4%20LCD.pdf
- Elprocus, LCD Interfacing (2019), https://www.elprocus.com/lcd-interfacing-with-8051-microcontroller/
- Winstar, 20x4 Character LCD Display (2018), https://www.winstar.com.tw/products/character-lcd-display-module/20x4-lcd-display.html
- VISHAY INTERTECHNOLOGY," LCD-020N004L" (2017), https://www.vishay.com/docs/37314/lcd020n004l.pdf
- Hitachi," Systronix 20x4 LCD" (2000), http://www.systronix.com/access/Systronix_20x4_lcd_brief_data.pdf

8) Speed sensor

- Electronics Hub, Interfacing LM393 Speed Sensor (2019), https://www.electronicshub.org/interfacing-lm393-speed-sensor-with-arduino/

9) FreeRTOS

- API Reference (2019) https://www.freertos.org/a00106.html

10) AUTOSAR

- Autosar Layered Architecture https://www.embitel.com/tag/autosar-layered-architecture

11) Machine learning

- Simplilearn, Convolutional neural networks (2019), https://www.simplilearn.com/convolutional-neural-networks-tutorial
- Edureka, Developing An Image Classifier In Python Using TensorFlow (2019), https://www.edureka.co/blog/convolutional-neural-network/
- Ehab Essa, Neural network lectures (2019), https://drive.google.com/drive/folders/1Dv3mzafx9l3Ogwyi1y50vsFMVZCvMfJj?fbclid=IwAR0 8Cq1DPrncB5Y19NDSoMQrK9k7uHEIDrpZ7xoLfYQLNVGvv5yT4m3n4
- CS231n Convolutional Neural Networks for Visual Recognition (2019), http://cs231n.github.io/convolutional-networks/#fc
- Keras, The python deep learning library(2019),
- https://keras.io/?fbclid=IwAR14aZ5mkbJZ-cRHA5PrbQvmK4nLRFNI6MO3GnK1GHfYM_RApziFsaaSrIw
- Mahmoud Badry, Convolutional neural networks (2019),

https://drive.google.com/drive/folders/19VGXK1DGXjAscY8Og4LvwxY2FDD4IjF8?fbclid=Iw AR1wh2X-CiFw8R6AX4FP3TMALMqmDTPVT3kuY8eD4ay_augbVr4VxHM96JU

- Sabyasachi Sahoo, Towards data science, Residual blocks — Building blocks of ResNet (2018), https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
- Anish Singh, Towards Data Science, Activation functions (2017), https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f