



NUST COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING

Autonomous Unmanned Ground Vehicle

A PROJECT REPORT

DE-40 (EE)

Submitted by

PC Muhamad Qasim Khan
ASC Syed Faizan Ali Haider
PC Ammar Omar Khan
GC Syed Muhammad Khurram Wasti

BACHELORS IN ELECTRICAL ENGINEERING YEAR 2022

PROJECT SUPERVISOR

Dr. Fahad Mumtaz Malik

**NUST COLLEGE OF
ELECTRICAL AND MECHANICAL ENGINEERING
PESHAWAR ROAD, RAWALPINDI**

ACKNOWLEDGEMENTS

We are thankful to Allah Almighty, the most merciful and Al Fattah, for the blessings, courage, and the intellect He has bestowed us with, because of which it became possible for us to accomplish such remarkable goal. We are thankful to our parents for heartening and being there for us and at each thick and thin. We are much obliged to our supervisor Dr. Fahad Mumtaz Malik for his continuous guidance and the valuable time he spent on providing us with the support and the encouragement through each stage. We are honored to have received his supervision in the project, enabling us to achieve the milestone. Moreover, we are grateful to all our fellows, friends who supported us intellectually and emotionally. Many thanks for all those who contributed in it, without all their support and appreciation we would be lacking a major source of inspiration and motivation. We are hopeful that this project will add value to the peoples' life and would be developed further to adopt this technology on the massive level acting as a benchmark for future technologies of Pakistan.

ABSTRACT

Automation technologies are bringing revolutionary changes in the world. Autonomous driving is subject to high academic research as well as high investments from the corporate sector. Demand for unmanned ground vehicles in industry, agriculture, and military has increased due to advances in manufacturing and automation. Waypoint guidance and navigation are indispensable fields in the research in autonomous vehicles. Following user-defined paths and seeking goal locations is the main objective in waypoint guidance. In this work, an autonomous Unmanned Ground Vehicle (UGV) is commanded to drive itself along a path defined by a series of waypoints, which will be user-defined through a mapping interface. In autonomous navigation, the UGV use Sharp IR proximity sensors, GPS receivers and monocular and stereo vision cameras to circumvent obstacles and navigate through a path defined by GPS waypoints. It has to autonomously navigate through its surroundings and execute its mission hence including obstacle detection and avoidance. This report consists of implementing a system that can detect any obstacles in the way in real-time. Autonomous UGV are poised for accelerated adoption, with an ability to create and keep a map of their environment based on the input from the variety of sensors located at multiple parts of the UGV. The software then processes all these sensory inputs, determining the instructions to be sent for the UGV's actuation. UGV uses VIO for state estimation, convolutional neural networks and computer vision for perception and stereo vision-based 3D object detection for localization of obstacles, along with path planning algorithms to plan the local and global path, which is followed by the actuation of low-level controller. All these technologies enable efficient autonomous navigation.

Table of Contents

ACKNOWLEDGEMENTS	1
ABSTRACT.....	2
Table of Contents	3
List of Figures.....	7
List of Tables	9
Chapter 1: INTRODUCTION.....	10
1.1 Background	10
1.1.1 Perception	11
1.1.2 Path Planning.....	11
1.1.3 Control System	12
1.1.4 State Estimation.....	12
1.2 Project Objectives	12
1.3 Design Workflow.....	12
1.4 Technical Details Summary	13
1.5 Project Benefits	14
Chapter 2: LITERATURE REVIEW	15
2.1 Waypoint guidance	15
2.2 Position Control	16

2.3	Perception	17
2.3.1	Developments in Perception	17
2.3.2	Research in Perception	17
2.3.3	Neural Network Based Object Detection	18
2.3.4	Semantic Segmentation	18
2.4	State Estimation.....	19
2.4.1	Kalman Filter.....	19
2.4.2	Extended Kalman Filter.....	20
2.5	Application for further study: Surveillance Missions with Multiple UGVs	21
2.6	Applications of Unmanned Ground Vehicle for Fighting and Detection	21
Chapter 3: HARDWARE METHODOLOGY		23
3.1	UGV Circuitry and Components	23
3.1.1	DC Motors	23
3.1.2	Motor Drivers (H BRIDGES).....	24
3.1.3	ARDUINO.....	24
3.1.4	JETSON NANO	28
3.1.5	GPS UBLOX NEO M8N	30
3.1.6	Magnetometer (QMC5883l).....	33
3.1.7	Stereo Vision Camera.....	34

3.1.8 BLUETOOTH SENSOR (HCO5-HC06)	35
3.1.9 Buck Converter(XL4015)	36
3.1.10 GPS External Antenna.....	38
3.1.11 SparkFun IMU	39
Chapter 4: EXPERIMENTAL TECHNIQUES AND METHODS	41
4.1 Software Simulations	41
4.2 GPS code testing	41
4.3 Waypoint Selection Code.....	44
4.4 Waypoint Guidance.....	46
4.5 Obstacle Avoidance	50
Chapter 5: SOFTWARE METHODOLOGY	52
5.1 Implementation of Perception stack.....	52
5.1.1 Camera Calibration.....	52
5.1.2 Object Detection	53
5.2 Graphical User Interface Design.....	57
5.2.1 RC Control Buttons	58
5.2.2 Connect to Arduino	58
5.2.3 Run Camera	59
5.2.4 Load Map and Load mission	60

5.2.5 Autonomous Button.....	61
4.6 System integration using ROS	61
Chapter 6: RESULTS AND DISCUSSION	64
6.1 Sources of Error	65
Chapter 7: CONCLUSIONS	66
Chapter 8: FUTURE WORK	67
REFERENCES	69

List of Figures

Figure 1 Neural Network Model for FSCNN	19
Figure 2 Motor driver H-Bridge BTS-7960.....	24
Figure 3 Arduino Mega and UNO	28
Figure 4 Nvidia Jetson nano	30
Figure 5 GPS UBLOX NEO M8N	33
Figure 6 Magnetometer (QMC5883L).....	34
Figure 7 Waveshare IMX-219 Stereo Camera.....	35
Figure 8 Bluetooth Module HC-06.....	36
Figure 9 Buck Converter XL4015.....	38
Figure 10 GPS External Antenna.....	39
Figure 11 SparkFun 9DoF Razor IMU	40
Figure 12 GPS Code testing output	41
Figure 13 I2C Communication between 2 arduinos	43
Figure 14 GPS data transfer.....	43
Figure 15 Ardupilot mission planner	44
Figure 16 Saved Waypoints from mission planner.....	45
Figure 17 Waypoint selection flowchart.....	45
Figure 18 Waypoint guidance algorithm	48
Figure 19 Bearing angle calculation	48
Figure 20 Waypoint navigation flowchar	49
Figure 21 Waypoint guidance flowchart.....	50
Figure 22 Stereo Camera Calibration MATLAB.....	53

Figure 23 Deep Learning Performance Jetson nano	54
Figure 24 SSD-Mobilenet-v2 architecture.....	54
Figure 25 Convolution layers Mobilenet v2	55
Figure 26 TensorRT optimization.....	56
Figure 27 Object detection Mobilenet v2 inference	57
Figure 28 Graphical user interface.....	58
Figure 29 GUI RC control options.....	58
Figure 30 GUI serial communication	59
Figure 31 GUI Camera preview options.....	60
Figure 32 Mission planner through GUI.....	61
Figure 33 GUI integration using ROS	62
Figure 34 ROSCORE launch command	63
Figure 35 ROS topics for data communication.....	63

List of Tables

Table 1 DC motor specification.....	23
Table 2 Arduino specification.....	26
Table 3 Nvidia Jetson nano specification	29
Table 4 Buck converter specification	37

Chapter 1: INTRODUCTION

1.1 Background

A UGV (Unmanned Ground Vehicle) is a motorized machine that can operate without any onboard human guidance. UGVs can be of two types i.e. remote controlled and autonomous. It can be operated manually through wireless communication with the end-user in real-time or autonomously through proper programming and sensor data definition. They have been increasingly used for military purposes and emergency response units. In addition, they are also used commercially within industries and factories to transport large objects across large distances.

Successful deployment of UGVs requires programming including path planning and tracking algorithms. The controller should be capable of performing the functions of analysis of sensor data, localization, GPS data analysis, and combining all of the above and guide the UGV. The path-following control of the UGV can be funneled down to waypoint guidance.

Waypoint guidance – the following of different waypoints with an undefined trajectory is very important in UGV control, making it possible to navigate to certain provided points. A waypoint is a point of reference that can be used for the location of certain areas. They are of a specific latitude and longitude value.

The objective of this paper is the development of a waypoint-guided algorithm suitable for UGVs; which can follow waypoints, detect obstacles, and avoid them suitably. The ground control was established using Arduino. The waypoints would be user-defined and would convert mapping points to a Cartesian space for easier vehicular navigation. With satellite values for global maps used in the present time, using an API for mapping – Mission

Planner –within the system would present precise positioning.

This thesis comprises of an elaborate approach to design the high- and low-level controllers, and the description of autonomous and electrical systems of the vehicle. The objective is elaborated in this chapter which shall be followed by a detailed literature review. The hardware methodology is elaborated in chapter 3, while experimental techniques and methods in chapter 4. The software methodology in chapter 5. The operating instructions in chapter 6. The design of autonomous control is a highly complex problem and the major areas of research include Vehicle Dynamics, Control Systems, Geometric Computer Vision and Deep learning. The areas for active research for the Final Year Project including the objectives can be broadly classified in the following technological domains:

1.1.1 Perception

The lynchpin for the success of the Self Driving Car will be its ability to Perceive and comprehend the environment around it, in its entirety. The requirement is fulfilled 3D Object detection and tracking and the most widely used AI algorithm is the Convolution Neural Network (CNN). The objective for the FYP is to collect datasets for our road environment and perform inference to implement the task of perception. The performance of autonomous operation also requires efficient detection of obstacles, lane lines and pedestrians.

1.1.2 Path Planning

The path planning for Autonomous vehicles has been extensively explored and these range from the classical space configuration and curve-based path planning methods, adapted for autonomous navigation. For our UGV the path-following control of the UGV is

funneled down to waypoint guidance.

1.1.3 Control System

The controller regulates some of the states of the vehicle by sensing the current state variables (feedback) and generating suitable inputs for the plant to achieve its desired state. The task is to develop a controller to achieve desired heading and velocity.

1.1.4 State Estimation

The determination of the vehicle state is the primary requirement for further planning and controls, which is composed of vehicle velocity, absolute position, and heading and relative displacement. This is accomplished by visual inertial odometry which is loosely coupled with GPS and encoder for improved estimate through sensor fusion. Each of the individual sensors are prone to noise and drift, but the combined result of state estimate is an accurate prediction, utilizing several optimizations based and Kalman filter approach.

1.2 Project Objectives

The specific scope and objectives of the Final year project are as follows:

- Development of a prototype electric autonomous UGV.
- Achieving high level of autonomy wherein the UGV itself is capable of navigating through GPS waypoints from a start point to a destination.
- Integrating maps, perception sensors and motion sensors for the UGV to be able to perceive. While reducing sensor dependencies, to only camera-based perception.

1.3 Design Workflow

The implementation methodology for the project would be two dimensional. One of the aspects would be development of the UGV prototype, embedding the existing technologies and algorithms. The other one would be extension of state-of-the-art research in this

domain. Eventually both the research and development algorithms will be embedded so that a prototype of UGV with efficient algorithms is produced as an outcome.

The selection of suitable locations for electronic drives and microcontroller will be performed. The UGV dynamic state estimation sensors, including MEMS Inertial measurement units (IMUs) along with GPS will be installed. These sensors are required for determination of UGV dynamics and response which is necessary for generation of acceleration, braking and steering inputs.

The actuation decisions are taken by control law on the basis of required linear and angular positions, and velocities determines control inputs. The trajectories for the control system will be determined by the perception sensors.

Scene recognition puts the emphasis on object detection. Object detection is finding object and their types within a scene, bounding boxes are used to localize objects by using Computer Vision algorithms. The models have to be trained for our requirements, to yield efficient results within our operational design domain.

The Path planning system determines trajectories for the low-level control system. The user provides set of waypoints which the UGV follows and reach goal location. the waypoints are sent to the microcontroller containing the code for low level control that generates instructions for UGV actuators which control the heading, braking and acceleration of the UGV.

1.4 Technical Details Summary

UGV powertrain constitutes of a battery, two motor controllers, two motors and wheel. The power of the two motors is transmitted to the rear tires and the front tires are driven. The UGV is differential drive so heading is basically changed by varying the speeds of the

motor.

Equipment selection for the drive train of the Autonomous UGV include wiper motors controlled using motor drivers. Motors RPM are 112 r/min.

The vehicle will be powered by rechargeable batteries. Lithium-ion batteries will be used as they have a high-power density, high energy efficiency and better performance. The requirements of the battery are 12V and 8.4Ah, which will be achieved through 9 cell configuration considering the capacity of single cell to be 2.8Ah.

For perception, cameras will be used. The stereo cameras are used for depth measurement and it provides depth from 0.4 to 19 meters. The sensor requirements include U-blox NEO-M8N GPS, and SparkFun Razor M0 orientation sensor that produces attitude and heading estimates from triaxial accelerometer, gyroscope, and magnetometer data.

Every sensor data is collected on separate Arduino UNO microcontroller so that there is no lag.

1.5 Project Benefits

These UGVs can be employed for commercial or military uses; and with waypoint navigation, the semi-autonomous UGV will be useful for a variety of tasks that are too dangerous or inefficient for humans to carry out. The fact that they can work around the clock adds to the advantages it has over humans in areas such as harvest in agriculture and pesticide and insecticide spraying. They can also be used to transport materials along with long distances with several stops made with user-defined waypoints to pick up or drop off items. This in turn increases productivity and saves time performing operations that would otherwise take longer to complete

Chapter 2: LITERATURE REVIEW

This chapter will throw light on previous work done within the domain of robotics on Unmanned Ground Vehicles (UGVs) and their path planning.

2.1 Waypoint guidance

Path planning is the main component within autonomous vehicles whether they may be UAVs or UGVs and it's an established field. The pure pursuit algorithm is one of the most tried and tested ones, resulting in large accuracy and rare errors. The waypoints can be set up by selecting specific points. The vehicle kinematic model, on which the odometrical equations depend, will be set up; and hence the waypoint locations in relation to the vehicle position will be calculated[1].

Pure pursuit uses cross-track errors and look ahead distances (as explained further on) to make the steering as humanly as possible, in terms of measuring the error before time and hence closing the error during driving the vehicle. The cross-track error and the look-ahead distance are not measured using sensory data – they can be – but are mostly calculated through odometrical equations. Many applications of pure pursuit are dependent on GPS and IMU. IMU uses gyroscopes to measure yaw, roll, and pitch angles for information on vehicular position; they also provide linear acceleration information. The values from the IMU are used with odometrical equations and dead reckoning to correct any inaccuracies from the GPS positioning. They can be made more accurate using Kalman or Extended Kalman Filters; however, it should be noted that even though there is substantial literature on state estimation on a vehicular level, it would not be addressed in this thesis, as it is not the main focal point of the paper. In addition to this, the pure pursuit controller used below

is based on the traditional Ackermann steering model. An advantage to the pure pursuit controller is that they have no restriction on whatever path is set by the front-end user – unless there is a hardware boundary limit set. The paths can be set as continuous curvatures that can be potentially discretized into piece-wise linear segments[2].

2.2 Position Control

The UGV can be controlled using two modes of operation – teleoperated and autonomously. The teleoperation has been achieved using a GUI designed using PyQt, vehicle was operated with ease at the Department of Electrical, NUST using custom made GUI..

Navigation is to calculate and estimate the current position and orientation of UGV from the left and right wheel encoders and IMU. For controlling position, a method called Dead Reckoning is used that utilizes the differential drive kinematics model. Position using this method is found by finding speed and time. Wheel encoders are attached to the wheels. Encoder is incremented with each rotation of the wheel. Encoder increments as ticks per centimeter moved by the UGV. Speed of UGV is calculated through specific equations that accounts for the increments in encoder counts within a sampling period. Dead reckoning is very prone to cumulative and approximation errors. In other automotive applications with dead- reckoning navigation systems, the vehicle is equipped with sensors and encoders that record the steering direction and the wheel rotations. The sensor data is then passed through a Kalman filter so that the sensor data is decipherable by the dead reckoning system⁶. Dead Reckoning is also used to reduce the need for sensors and position sensing technology such as GPS, which reduces the complexity of programming and the computational power needed to run the UGV[3].

2.3 Perception

2.3.1 Developments in Perception

Perception and Understanding of the environment using vision is one of the most important and demanding domains of computer vision. It incorporates a lot of concepts and ideas. Scene recognition stresses on providing a global explanation of a scene, commonly summed up at a level of single scene categorization. Object detection primarily focuses finding and categorizing objects within a scene by effectively localizing and representing using bounding boxes. Semantic segmentation focus attention on giving a finer-grained prediction of the category every pixel belongs to, representing the boundaries of each object present in the scene. The era of major developments in the computer vision field has brought a revolution in the field, majorly through the advent of advanced Deep learning techniques which have surpassed most of the traditional techniques. The pinnacle of deep learning algorithms is outperforming many techniques for the task of scene understanding and recognition. Training of deep learning models requires a lot of data and computational power. And seeing the circumstances the large-scale datasets such as Cityscapes [4] and Mapillary vistas [5] are of great importance, for the scientific and the automotive industry. For understanding the complicated interactions between traffic contributors in street images, a lot of research efforts and funding have recently been done into developing datasets to train deep models for this task, such as the KITTI Vision Benchmark Suite [6].

2.3.2 Research in Perception

Extensive research has been conducted on camera topologies and visual perception algorithms incorporating several computer vision techniques involved in the perception of obstacles and pedestrians while also testing the performance of various perception

algorithms available for autonomous driving for these complex tasks. Extensive literature review has been conducted on and semantic segmentation, 3D object detection, multi-object tracking, which are the Deep learning and CNN based algorithms. Significant study on depth sensing for stereo vision, and stereo correspondence has allowed the understanding of underlying principles for geometric computer vision and perspective geometry.

2.3.3 Neural Network Based Object Detection

Focus of attention in object detection is finding object instances at a single scene category level. Methodologies of object detection are significant in determining the location of the objects in the image, and eventually categorizing them into respective labels to adequately classify them. We began our Literature review neural networks starting from basic Logistic Regression and advanced our research and simulations to advanced algorithms using CNNs and YOLO algorithms. For conceptualizing the domain of NNs and enforce our understanding, classification and detection neural networks were designed, coded manually accompanying all the details associated with forward propagation, back propagation, and developing and training NN with SGD and ADAM algorithms. The advanced concepts including regularization and hyper parameter optimization were incorporated to train efficient neural networks with good accuracy of classification. The training using custom datasets is performed in order to achieve good results in our environment.

2.3.4 Semantic Segmentation

Semantic Segmentation is an efficient technique for the segmentation of several objects within the environment. It is a highly supervised problem requiring large amounts of

completely annotated datasets, by classifying each pixel and identifying their location and class. This is a highly complex problem requiring deep learning models to adequately solve the problem.

An efficient approach to semantic segmentation includes Fast SCNN algorithm [7], which has a dual pipeline architecture to find features on coarse level and finer scale which are added to recognize the entire scene, and label the boundaries based on intra- class variations.

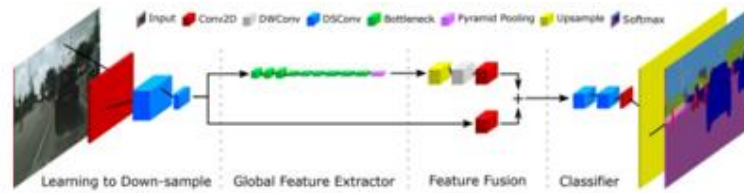


Figure 1 Neural Network Model for FSCNN

The Fast SCNN algorithm has the advantage of good real time performance and better accuracy due to the encoder decoder structure. Combination of convolution and pooling operations are used to extract deep CNN features by encoder structure. The spatial details from the sub-resolution features, and the prediction of the object labels is done by the decoder structure.

2.4 State Estimation

State estimation so critical for UGV's and the state is an indication to where the vehicle is present, where it is headed, and determination of motion parameters such as velocity and angular speeds. State estimation remains a fundamental requirement to ascertain these parameters and inform the behavioral planner whether the goal point has been reached.

2.4.1 Kalman Filter

As machine learning algorithms are becoming advance, the use of Kalman filter still

remains useful as it fuse measurements from several sensors to find the state of UGV in real-time. The Kalman filter algorithm [8] was published in 1960 by Rudolf E. Kalman. It was this ground-breaking innovation that played a pivotal role in success of many space missions. Kalman filter is able to find changing state. The Kalman filter makes an estimate of state using probabilistic approach and refresh it in real time by prediction and correction. From our probabilistic measurement model, current prediction is corrected by fusing the current sensor information with previous prediction by using gain matrix k i.e. optimized. At the end we got for our states an updated probabilistic estimate at time step k . The recursively squares and Kalman filter are similar, but recursively squares also adds a motion model that defines how our state evolves over time and is the best linear unbiased estimator.

2.4.2 Extended Kalman Filter

For nonlinear systems Extended Kalman Filter is designed, and it is often considered one of the workhorses of state estimation,, because of numerous domains of its implementation. The filter uses a measurement model to predict what measurements should arrive based on the state estimate and compares these predictions with the measurements that arrive from the sensors. The Kalman gain determines the weight all these pieces of information so that we can optimally combine them into a corrected estimate. This is called a predictor corrector architecture. The filter performs linearization of function to extract the first-order terms of the Taylor series expansion. The Jacobian matrix is evaluated which is the matrix of all first-order partial derivatives of a vector valued function. The linearized model exactly coincides with the non-linear model at the operating point and secondly, the appearance of the L and M Jacobians are related to the process and measurement noise.

The EKF estimates the result of the prediction step for the mean and covariance of the state which is moreover make of use by the path planning system to find the required trajectory and control systems uses it to regulate the state variables.

2.5 Application for further study: Surveillance Missions with Multiple UGVs

Due to a set of UGVs and a rhombic area, and paths for all UGVs consisting of waypoints, in the area each spot is visible from a spot on a path and for parallel search execution the time that is taken is minimized [10]. There, the camera's field of view is considered to be blocked by the hurdles and confined by a maximum range of the used sensor. The confined implementation of communication expands the first by giving access to the information database caused by the sensors to be connected when the monitoring function of UGVs stop to perform. The security and shared network – I2C – are of importance when need for shared awareness is the case, in order to relay the obtained sensor data, or to protect the team from spiteful people attempting to reach idle UGVs.

2.6 Applications of Unmanned Ground Vehicle for Fighting and Detection

Due to UGVs tremendous ability in very high-risk operations, they have gained a lot of interest and importance within the autonomous and emergency response field [11]. The inter-vehicle harmony and coaction is a important hurdle to a fruitful project in emergency response works. A harmonious mechanism for control is needed to handle the decision-making process at the system level and discretized and understandable the flow of sensor data across the connected vehicles so that the joint task is fulfilled. For UGV hierarchical model A management system is launched. A UGV just needs to be connected to the same network as a UAV and be able to receive optimized sensor data from it, process it, and run the correct commands on an urgent basis. The UGV and UAV positions are constantly

updated with a GPS and IMU system working together using odometrical data and tracking positions. Of course, the UAV has to be either tele operated or completely autonomous, while the UGV is completely autonomous in that regard due to a higher danger factor.

Chapter 3: HARDWARE METHODOLOGY

3.1 UGV Circuitry and Components

The UGV is equipped with ideal sensors throughout because it has been designed to move through simple terrains, so that it can overcome track problems over unknown paths. To help achieve a smooth path over different topologies and terrain features the UGV needs to be equipped with the correct array of sensors, physical, electrical, and mechanical components. They are as follows:

3.1.1 DC Motors

2 motors have been used in the UGV for the purpose of track movement. It would be difficult these motors and would require multiple motor drivers to communicate with each other. To avoid this, we connect the right one to the front and back and the left one to front and back.

This wiper dc motor is usually used for wipers of a car it is mounted in the area under the windshield base. The motor is used to drive the mechanism which rocks the wiper arm. If vehicle has a rear wiper a separate motor is used for it. The issues that arise generally with wiper dc motors in a vehicle are due to fuse of the motor, trip of the circuit breaker, interior switch that is responsible for wiper failing and wires in the circuit are damaged. All these issues were kept in mind while use of wiper dc motors in our project. Corrosion and debris may also block the moving parts and they may need lubrication.

Table 1 DC motor specification

Motor Voltage	12V
Speed (Rpm)	45

Power (Watts or HP)	14
Rated Voltage	13.5 V
Current No load	1.5 A
Speed (On load)	38 RPM

3.1.2 Motor Drivers (H BRIDGES)

H-bridges are used to switch polarity of voltage. The use of H-bridges in robotic applications is because with them one can easily control motor in forward and backward direction. The name “H” is due to branches of a letter "H" represent four switching and the load connected as the cross-bar.

H bridge can be used as power supply to a device with two terminals. If switch are properly arranged the polarity of the power to the devices can be changed. DC motor Driver and transformer of switching regulator are the two mostly used application.

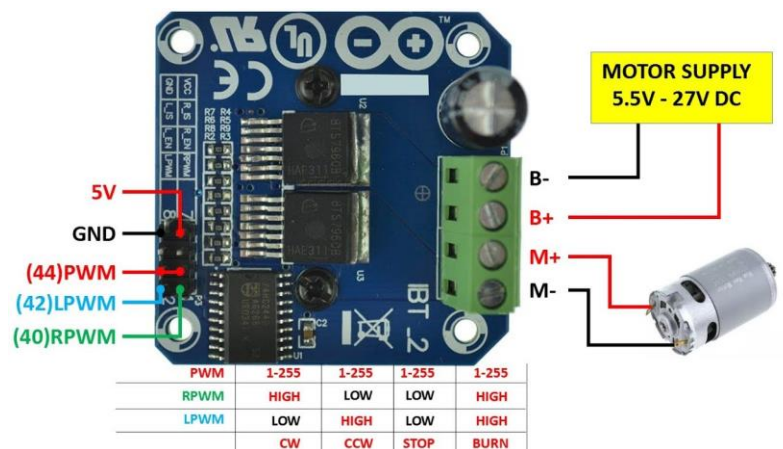


Figure 2 Motor driver H-Bridge BTS-7960

3.1.3 ARDUINO

We use 4 Arduinos in our UGV, 3 of them are UNO and one is Mega.

The name “Uno” of the Arduino uno is based on the Italian word UNO which means one.

It is a ATmega328 based microcontroller board. This board comprises of a power jack, an RST button, 14 digital input /output pins, a USB connection, resonator-16 MHz, and an ICSP header. These components act as a support for the microcontroller for operation when it is connected to the computer. The board can be powered with the help of a battery, adapter or USB.

The features of Arduino Uno ATmega328 includes the following..

- Operating voltage is 5V
- Recommended input voltage will range from 7v to 12V
- Input voltage ranges from 6v to 20V
- Digital input/output pins are 14
- Analog i/p pins are 6
- DC Current for each input/output pin is 40 mA
- DC Current for 3.3V Pin is 50 mA
- Flash Memory is 32 KB
- SRAM is 2 KB
- EEPROM is 1 KB
- CLK Speed is 16 MHz
- The applications of Arduino Uno include the following:
 - DIY projects.
 - Code-based controlled projects.
 - Automation system development.
 - Basic circuit designing.

Table 2 Arduino specification

MICROCONTROLLER	<u>ATmega2560</u>
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	54 (of which 15 provide PWM output)
ANALOG INPUT PINS	16
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
CLOCK SPEED	16 MHz
LED_BUILTIN	13
LENGTH	101.52 mm
WIDTH	53.3 mm
WEIGHT	37

The Arduino Mega 2560 is a ATmega2560 based microcontroller board that contains 54 digital I/O pins, 16 analog inputs, 4 UARTs serial ports, a 16 MHz oscillator, a USB connection, a power jack, an ICSP header, and a reset button. All these componens are

used to support the operation of this microcontroller. It can be powered via a computer, AC/DC adapter or battery to get started. It is also compatible with shields designed for the Arduino Uno.

The Mega 2560 is an upgrade version and replacement to the Arduino Mega.

Features/Specifications

- Operating voltage: 5V
- Input voltage (recommended): 7-12V
- Input voltage (limits): 6-20V
- Digital I/O pins: 54 (of which 14 provide PWM output)
- Analog input pins: 16
- DC current per I/O pin: 40mA
- DC current for 3.3V pin: 50mA
- Flash Memory: 256 KB, 8KB used by bootloader
- SRAM: 8 KB
- EEPROM: 4 KB
- Clock Speed: 16 MHz
- Advantages:
 - Fast startup with low power consumption.
 - Ease of use, with 8-bit microcontroller being less complex than 32/64-bit versions
 - ease of exploring, developing and debugging own touch applications using the QTouch Suite.
- Patented Adjacent Key Suppression technology allows for unambiguous detection of key events

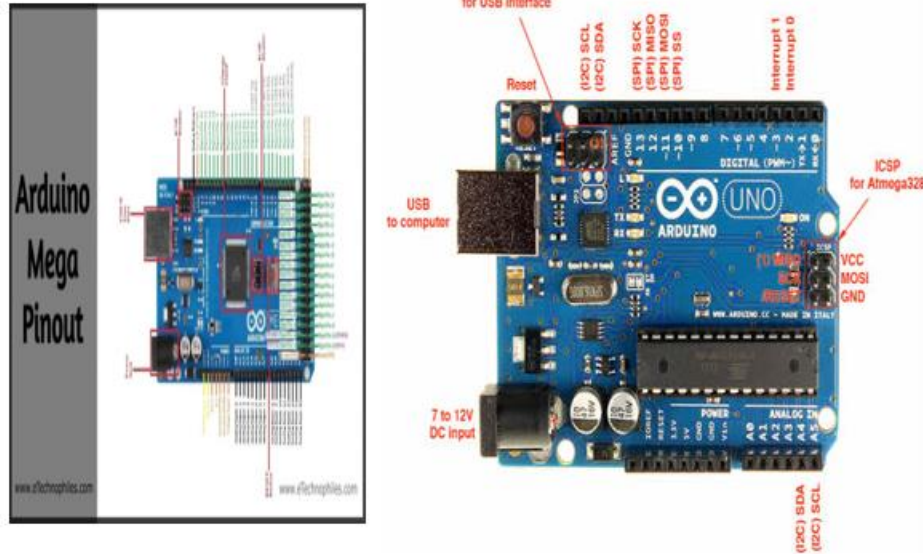


Figure 3 Arduino Mega and UNO

3.1.4 JETSON NANO

In this UGV we use NVIDIA Jetson Nano. Jetson Nano is a small and powerful single board computer with 138 cuda cores GPU for engineering applications . It is a devices that delivers the computational capability of modern AI algorithm in a relatively cheap price. The Jetson Nano board because of it's 128 cores GPU is specialized for fast computation of deep learning and AI applications. It can easily perform microcontroller level tasks similar to the Raspberry Pi boards because it contains 40-pin general purpose input output header.

The Jetson Nano Development Board is an small single board computer optimized for AI application. It can efficiently perform neural networking, image processing, and many more tasks that has made it suitable to create millions of intelligent systems. It can be connected both by wire or wirelessly so it can also be used for embedded projects. It has a high-speed Ethernet port for high-speed network connectivity. It can be remotely used without an external monitor because it also has a dedicated camera and display connector.

SPECIFICTAION

Table 3 Nvidia Jetson nano specification

GPU	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
CPU	Quad-core ARM Cortex-A57 MPCore processor
Memory	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Storage	16 GB eMMC 5.1
Video Encode	250MP/sec 1x 4K @ 30 (HEVC) 2x 1080p @ 60 (HEVC) 4x 1080p @ 30 (HEVC) 4x 720p @ 60 (HEVC) 9x 720p @ 30 (HEVC)
Video Decode	500MP/sec 1x 4K @ 60 (HEVC) 2x 4K @ 30 (HEVC) 4x 1080p @ 60 (HEVC) 8x 1080p @ 30 (HEVC) 9x 720p @ 60 (HEVC)
Camera	12 lanes (3x4 or 4x2) MIPI CSI-2 D-PHY 1.1 (1.5 Gb/s per pair)

Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I ² C, I ² S, SPI, UART
Mechanical	69.6 mm x 45 mm 260-pin edge connector



Figure 4 Nvidia Jetson nano

3.1.5 GPS UBLOX NEO M8N

NEO-M8N Evaluation board is used for easy and accurate evaluation of the high performance of the u-blox 8 positioning. For power supply and high-speed data transfer it

contains a built-in USB interface eliminating the need for an external power supply. The NEO-M8N Breakout is suited for applications like use in vehicles , laboratories and outdoor locations because of it's compact size and power supply options. The NEO-M8N board is very convenient through all stages of design-in projects as it can also be used with a notebook PC.

NEO-M8N features:

72-channel u-blox M8 engine GPS/QZSS L1 C/A, GLONASS L10F, BeiDou B1 SBAS
L1 C/A: WAAS, EGNOS, MSAS Galileo-ready E1B/C (NEO-M8N)

Position accuracy 2.0 m CEP

Acquisition

- Cold starts: 26 s
- Aided starts: 2 s
- Reacquisition: 1.5 s
- Sensitivity
- Tracking & Nav: -167 dBm
- Cold starts: -148 dBm
- Hot starts: -156 dBm
- Assistance AssistNow GNSS Online
- AssistNow GNSS Offline (up to 35 days)
- AssistNow Autonomous (up to 6 days)
- OMA SUPL & 3GPP compliant
- Oscillator TCXO
- RTC crystal Built-In
- Anti-jamming Active CW detection and removal.
- Extra onboard SAW band pass filter
- Memory Flash
- Supported antennas Active and passive
- Odometer Travelled distance
- Data-logger for position, velocity, and time



Figure 5 GPS UBLOX NEO M8N

3.1.6 Magnetometer (QMC5883L)

The QMC5883L is a multi-chip three-axis magnetic sensor. It is a compact, surface-mounted chip that has magnetic sensors and signal condition application specific integrated chip, it is designed for applications that require high accuracy such as compassing, navigation, robotics applications, and mobiles.

QMC5883L uses high resolution, magneto-resistive technology licensed from Honeywell AMR. It has a custom-designed 16-bit ADC ASIC which offers it the advantages of low noise, high accuracy, low power consumption, offset cancellation and temperature compensation. 1° to 2° compass heading accuracy is possible with QMC5883L . Easy interface is made possible through the I²C serial bus.

FEATURES

- 3-Axis Magneto-Resistive Sensors in a 3x3x0.9 mm³ Land Grid Array Package (LGA), guaranteed to operate over an extended temperature range of -40 °C to +85 °C.
- 16 Bit ADC With Low Noise AMR Sensors Achieves 5 Milli-Gauss Field Resolution
- Wide Magnetic Field Range (± 8 Gauss)
- Temperature Compensated Data Output and Temperature Output

- I2C Interface with Standard and Fast Modes
- Wide Range Operation Voltage (2.16V To 3.6V) and Low Power Consumption (75 μ A)
- Lead Free Package Construction

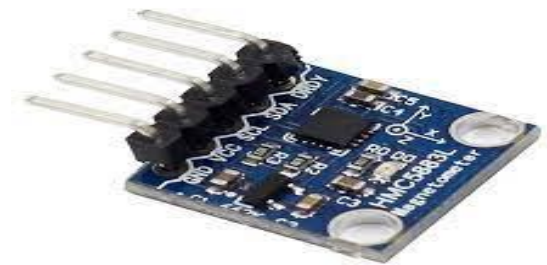


Figure 6 Magnetometer (QMC5883L)

3.1.7 Stereo Vision Camera

The stereo camera used in this project is waveshare IMX-219. It is a 8 Megapixel camera with 2.6 mm focal length of each camera and 60 mm baseline. It is designed especially for AI vision applications like depth estimation, SLAM and 3d reconstruction.

Each camera of this module provide a resolution of 3280×2464 . It has a field of view of 83/73/50 degree (diagonal/horizontal/vertical). It is a all-in-one package for robotics applications as it also comprises of a gyroscope, accelerometer and magnetometer.

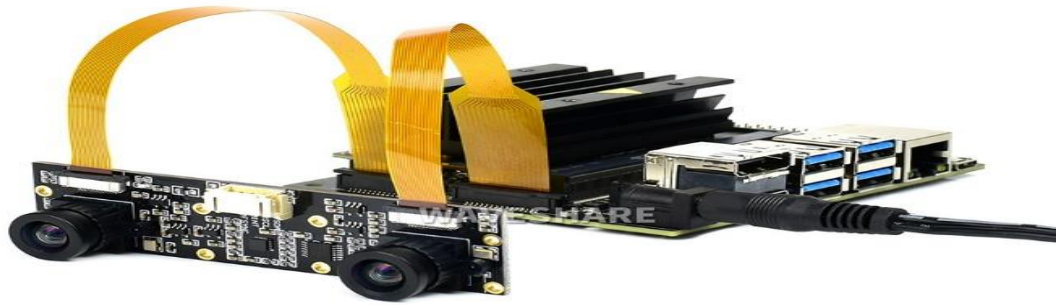


Figure 7 Waveshare IMX-219 Stereo Camera

3.1.8 BLUETOOTH SENSOR (HCO5-HC06)

The HC-05 module is a Bluetooth module that can communicate serially with Arduino through its serial port protocol. This module involves a Bluetooth IC named BC417 and supports Compute Module POE Board because it is compliant with Bluetooth v2.0 standards.

Specifications

- Low operating voltage: 4V to 6V (Typically +5V)
- Frequency: 2.45 GHz
- Range: <100m
- Follows IEEE 802.15.1 standardized protocol
- Supported baud rates are 9600,19200,38400,57600,115200,230400 and 460800.
- Can operate in Master, Slave or Master/Slave mode

HC-06 Bluetooth module is used to make short-range wireless data communication between two microcontrollers or systems. A PC or a smartphone can be used as a master Bluetooth device to pair it, so that it has a transparent operation to the user. All the serial input data that is received is immediately transmitted over the air.

Specifications

- Input voltage: 3.3~6V (Typically +5V)
- Built-in antenna
- Operating Current: 30mA
- Uses Frequency-Hopping Spread Spectrum (FHSS)
- Works with Serial communication (USART) and TTL compatible
- 2.4GHz ISM band frequency
- Default baud rate: 9600



Figure 8 Bluetooth Module HC-06

3.1.9 Buck Converter(XL4015)

The XL4015 buck converter is used in this project to step down 12 volts to 5.1 volts. 12 volts are coming from the battery source but Jetson nano voltage rating is 5.2 volts so this buck is use for this purpose. XL4015 power module used the DC to DC step-down (BUCK) power module, it operates at a switching frequency of 180kHz. It provides smaller sized filter components it operate at high frequency.

Table 4 Buck converter specification

Pin Name	Description
IN+	Positive input (Unregulated or Regulated)
IN-	Negative input (Ground)
OUT+	Positive Output (Regulated)
OUT-	Negative Output (Ground)

Features and Specifications

- Input voltage: 4 - 38V
- Output voltage: 1.25 - 36V (adjustable)
- Output current: Maximum output current 5A
- **Note: The higher the voltage, the load current increases. Try to use it within 4.5A.**
- Output power: It is recommended to use within 75W
- Note: Heat sink required if output power exceeds 50W
- Efficiency of this regulator up to 96%
- **Note: Efficiency is related to input, output voltage, current, and the voltage difference**
- Load regulation: 0.8%
- Voltage regulation: 0.8%
- Adjustable potentiometer onboard for output voltage adjustment.

- Power LED indicator
- Thermal Protection enabled.
- Short circuit protection: limited to 8A.
- Dimension: 54*23*18 mm (L*W*H)



Figure 9 Buck Converter XL4015

3.1.10 GPS External Antenna

An external GPS antenna is used with NEO-M8 module for fast and accurate operation of GPS. It comes with a SMA connector. It is a compact size user friendly antenna that helps the GPS in taking the signal more accurately and efficiently.



Figure 10 GPS External Antenna

3.1.11 SparkFun IMU

The SparkFun 9DoF Razor IMU M0 combines a SAMD21 microprocessor with an MPU-9250 9DoF (nine degrees of freedom) sensor to be used in the creation of a tiny, re-programmable, multi-purpose inertial measurement unit (IMU). It can be programmed to be able to monitor and log motion, transmit Euler angles over a serial port, or to act as a step-counting pedometer.

Quick Spec

- Integrated MPU-9250 IMU and SAMD21 microprocessor
- LiPo battery charger
- μ SD card socket
- Preprogrammed example firmware that streams and/or logs
- Accelerometer, gyroscope and magnetometer data and/or quaternions, and Euler angles
- Arduino-programmable via USB
- New MPU-9250 Arduino library with support for the chip's digital motion

processing capabilities

- Extra SAMD21 pins broken out
- System on/off switch
- **Arduino Compatible**
- The Arduino compatible Atmel SAMD21 is the onboard microprocessor it is a 32-bit ARM Cortex-M0+ microcontroller and is also featured on the Arduino Zero and SAMD21 Mini Breakout boards.



Figure 11 SparkFun 9DoF Razor IMU

Chapter 4: EXPERIMENTAL TECHNIQUES AND METHODS

Several experimental techniques were applied for the testing and experimentation of the hardware and software. This included the use of

Software Simulations

Algorithm Implemented Code

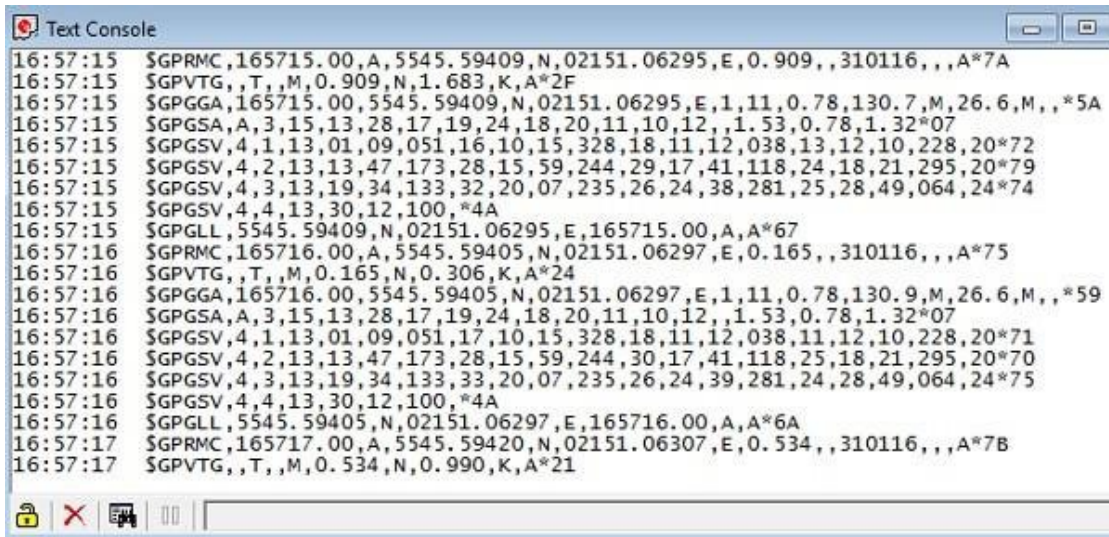
4.1 Software Simulations

The software simulations were only to be carried out for the waypoint selection, the GPS module testing code, and the code for the kinematic model which needed to be tested out before it can be implemented.

4.2 GPS code testing

The GPS needed to be tested so that we get correct values for the location of the UGV, instead of the just garbage values, this was done using the following method-

First tx pin of GPS-neo-m8n was connected to rx pin of Arduino and blank code was uploaded. Display is shown in figure below.



```
16:57:15 $GPRMC,165715.00,A,5545.59409,N,02151.06295,E,0.909,,310116,,A*7A
16:57:15 $GPVTG,,T,,M,0.909,N,1.683,K,A*2F
16:57:15 $GPGGA,165715.00,5545.59409,N,02151.06295,E,1,11,0.78,130.7,M,26.6,M,,*5A
16:57:15 $GPGSA,A,3,15,13,28,17,19,24,18,20,11,10,12,,1.53,0.78,1.32*07
16:57:15 $GPGSV,4,1,13,01,09,051,16,10,15,328,18,11,12,038,13,12,10,228,20*72
16:57:15 $GPGSV,4,2,13,13,47,173,28,15,59,244,29,17,41,118,24,18,21,295,20*79
16:57:15 $GPGSV,4,3,13,19,34,133,32,20,07,235,26,24,38,281,25,28,49,064,24*74
16:57:15 $GPGSV,4,4,13,30,12,100,,*4A
16:57:15 $GPGLL,5545.59409,N,02151.06295,E,165715.00,A,A*67
16:57:16 $GPRMC,165716.00,A,5545.59405,N,02151.06297,E,0.165,,310116,,A*75
16:57:16 $GPVTG,,T,,M,0.165,N,0.306,K,A*24
16:57:16 $GPGGA,165716.00,5545.59405,N,02151.06297,E,1,11,0.78,130.9,M,26.6,M,,*59
16:57:16 $GPGSA,A,3,15,13,28,17,19,24,18,20,11,10,12,,1.53,0.78,1.32*07
16:57:16 $GPGSV,4,1,13,01,09,051,17,10,15,328,18,11,12,038,11,12,10,228,20*71
16:57:16 $GPGSV,4,2,13,13,47,173,28,15,59,244,30,17,41,118,25,18,21,295,20*70
16:57:16 $GPGSV,4,3,13,19,34,133,33,20,07,235,26,24,39,281,24,28,49,064,24*75
16:57:16 $GPGSV,4,4,13,30,12,100,,*4A
16:57:16 $GPGLL,5545.59405,N,02151.06297,E,165716.00,A,A*6A
16:57:17 $GPRMC,165717.00,A,5545.59420,N,02151.06307,E,0.534,,310116,,A*7B
16:57:17 $GPVTG,,T,,M,0.534,N,0.990,K,A*21
```

Figure 12 GPS Code testing output

GPGGA Line was taken and copied to website:

(https://swairlearn.bluecover.pt/nmea_analyser) to decode NMEA lines.

Latitude and Longitude provided by website were verified from google maps, during all these testing antennae was in outside environment.

After verification Arduino library TinyGPSPlus-master was used to get latitude and longitude.

This method simulated the GPS values that we would be getting, so it was easier to understand the amount of change in the values of the latitudes and the longitudes. The code was designed to get valid data, which it did, and knowing the differences between different positions in lat and long terms made it easier to program the UGV accordingly in terms of sample rate from the sensors and the GPS module.

The satellite led light should be on whenever the code has been run so we are sure that the GPS module has been locked onto a satellite where it can get its values from.

Unfortunately, the GPS module only works when it's outdoors directly under the open sky because it only locks down on a satellite then. Hence, the usage of the UGV has been limited to the outdoors and cannot be used in closed interiors.

GPS is connected to Arduino uno and total four Arduinos are used, reason for using separate Arduino was due to low computational power of Arduino mega (Main controller Arduino), in main Arduino code length is of approx. 750 lines implanting GPS code in same Arduino was producing delays that is why UGV was unable to fetch correct bearing angle. I2C communication was used between uno and mega and parsing was done because Arduino is 8-bit Architecture and cannot transfer and receive values having more than 8 bits (max 255) so each byte was transfer one by one.

I2C Communication

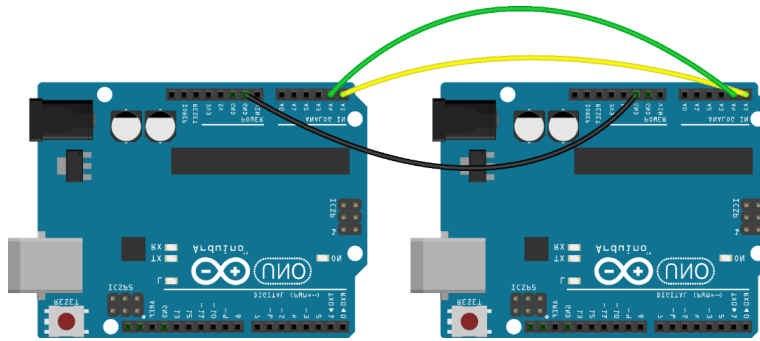


Figure 13 I2C Communication between 2 arduinos

GPS Data Transfer technique.

Master Arduino(unno).

```
int a = currLat;
float b = ( currLat - a) * 1000000;
int q = b / 10000;
int w = b / 100 - q * 100;
float r = q * 100 + w;
float y = b / 100 - r;
int i = y * 100;
//////////
int z = currLon;
float x = ( currLon - z) * 1000000;
int v = x / 10000;
int j = x / 100 - v * 100;
float m = v * 100 + j;
float k = x / 100 - m;
int d = k * 100;
//////////

Wire.beginTransmission(7);
Wire.write(a);
Wire.write(q);
Wire.write(w);
Wire.write(i);
//////////
Wire.write(z);
Wire.write(v);
Wire.write(j);
Wire.write(d);

Wire.endTransmission();
```

Figure 14 GPS data transfer

4.3 Waypoint Selection Code

The waypoint selection was done using Ardupilot mission planner. Initially we wanted to integrate Google map in python GUI, but issue was with purchasing of google map API and another option was to use folium maps but they are not user friendly and identifying correct waypoints might not be possible for user so considering all these issues we come with our own idea to use ardupilot mission planner basically it's for Pixhawk and allows user to select waypoints .Waypoints selected by user are saved in txt file and can be read with python to select target lat and log, python code was designed in such a way that user that send 999 waypoints in to Arduino and can be increased by just changing few lines of codes.

Ardupilot mission planner:



Figure 15 Ardupilot mission planner

Txt file having waypoints coordinates:

qasim3 - Notepad

File	Edit	Format	View	Help
0	WPL	110		
0	1	0	16	0 0 0 0 33.6198404 72.9569489 0.000000 1
1	0	3	16	0.00000000 0.00000000 0.00000000 0.00000000 33.61978680 72.95688850
2	0	3	16	0.00000000 0.00000000 0.00000000 0.00000000 33.61988170 72.95682950
3	0	3	16	0.00000000 0.00000000 0.00000000 0.00000000 33.61994980 72.95691130
4	0	3	16	0.00000000 0.00000000 0.00000000 0.00000000 33.61991300 72.95695020

Figure 16 Saved Waypoints from mission planner

Selected waypoints are sent from jetson nano to Arduino mega using rosserial and then by using UART communication these waypoints were stored in main controller Arduino.

FLOWCHART

Target waypoints transfer technique

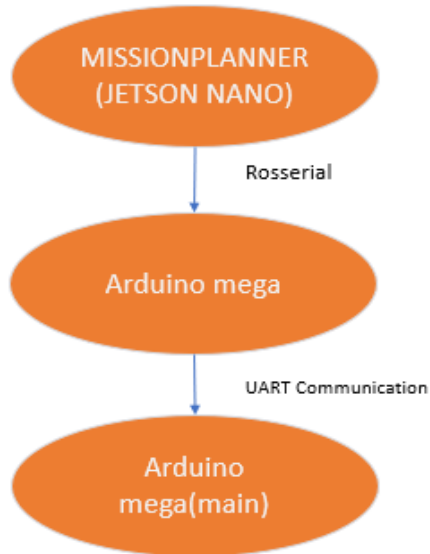


Figure 17 Waypoint selection flowchart

Rosserial Communication:

Rosserial act as bridge between jetson nano and Arduino mega and allows communication between two devices. Selected waypoints were sent to Arduino using parsing because Arduino is 8-bit architecture and delays were carefully managed.

UART Communication:

UART stands for Universal Asynchronous Reception and Transmission and is a simple communication protocol that allows the Arduino to communicate with serial devices. The UART system communicates with digital pin 0 (RX), digital pin 1 (TX), and with another computer via the USB port.

4.4 Waypoint Guidance

The waypoint guidance algorithm was personally developed in which mathematics played a large role. The distances needed to be calculated and the system needed constant updates to run smoothly and continuously.

The algorithm used to approach the solution was to calculate the distance between the current position of the UGV, and the distance between it and the next waypoint. The bearing angle (heading angle) was also calculated alongside the movement of the UGV which will be explained further on. While the distance between the waypoint and the current position of the UGV is not zero, the UGV uses the bearing angle (heading angle) to compute the direction of the movement of the UGV, a full flowchart will be shown to explain the algorithm fully and concisely.

The system uses the Haversine Formula⁸ to calculate the distance between the latitude longitude values that we have.

$$a = \sin^2(\Delta\phi/2) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = R \cdot c$$

ϕ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km/6371000m).

The Haversine formula utilizes the spherical law of cosines to compute the distance between the latitudes and the longitudes of the coordinates. There is a slight error while computing the value for the distance between the coordinates – 5% - and this is because the Haversine formula computes the distance based on the assumption that the globe is a perfect sphere when it is an ellipsoid.

There have been studies on these which led to the rise of another formula called the Vincenty formula which based the calculation based on the ellipsoid shape, but this formula was beyond the scope of this project due to the heavy number of iterations that needed to be done to calculate the distance which increases the computational power needed to run the system., which in turn lags the system.

The heading angle was calculated using simple Pythagorean math because of the smaller scale and constant update rate. The area between the waypoints is assumed to be small, hence Pythagorean math was used. The algorithm is as shown below –

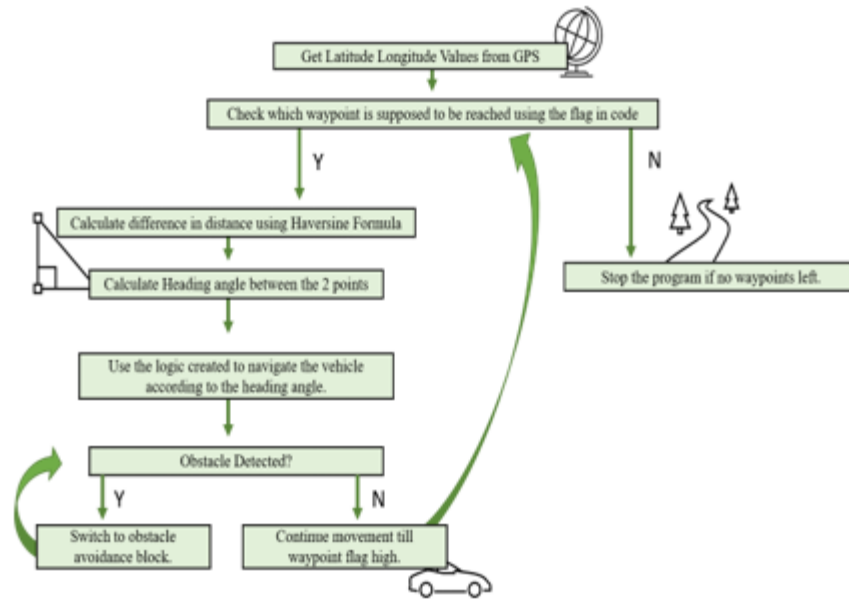


Figure 18 Waypoint guidance algorithm

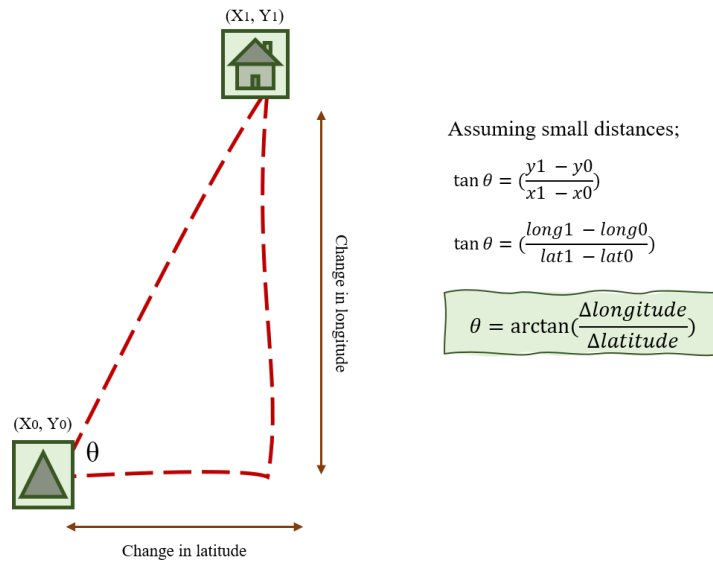


Figure 19 Bearing angle calculation

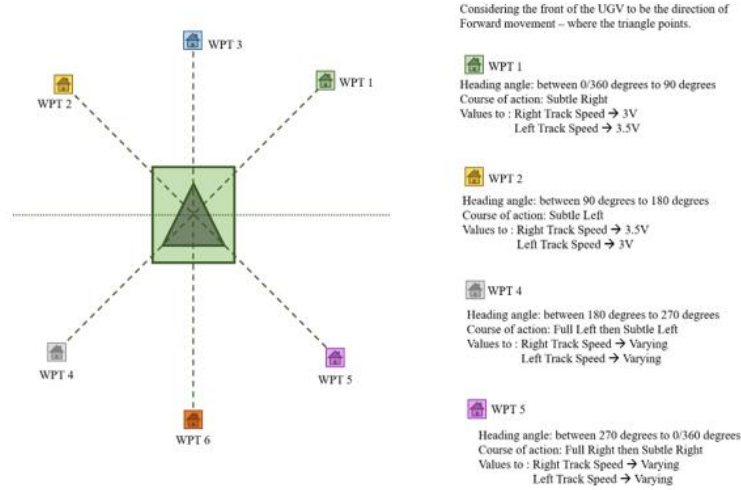


Figure 20 Waypoint navigation flowchar

To find the heading/bearing angle between two longitudes and latitudes, we have implemented the following formulae.

Let's assume we have two points A and B. The A points corresponds to the UGV's location, and its latitude and longitude is denoted as Θ_a and L_a respectively. The similar is assumed for point B as well that is Θ_b and L_b . R is the radius of the Earth and β is the heading angle/bearing angle. The angle is measured from the North direction, that is

North \square 0 degrees East \square 90 Degrees South \square 180 degrees West \square 270 degrees

Following are the equations, the heading angle is given by:

$$\beta = \tan^{-1}(X, Y)$$

Where X and Y can be calculated as:

$$X = \cos(\theta_b) * \sin(\Delta L)$$

$$Y = \cos(\theta_a) * \sin(\theta_b) - \sin(\theta_a) * \cos(\theta_b) * \cos(\Delta L)$$

Here, $\Delta L = L_b - L_a$

Complete workflow is show in figure below:

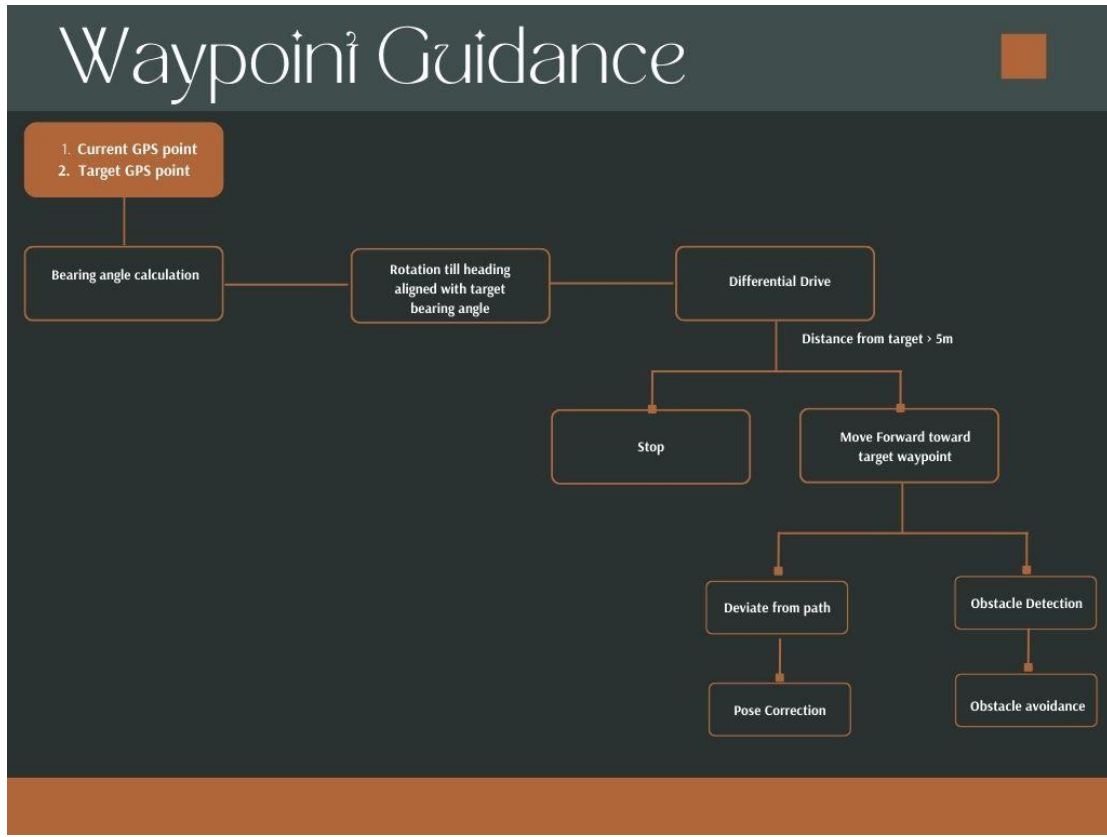


Figure 21 Waypoint guidance flowchart

4.5 Obstacle Avoidance

For an obstacle avoidance algorithm to be implemented, one needs the obstacles to be detected as well. Obstacle avoidance is very essential in local path Planning autonomous system must be having obstacle avoidance capabilities. Researcher and developer have proposed many methods to avoid obstacle in real time. Robot must be able to avoid obstacle and must start its motion toward target location after avoidance. Multiple options were available but considering our computational power and due to shortage of time we develop fuzzy Logic for obstacle avoidance which is explained in contents mentioned below:-

Obstacle Avoidance Tools:

- Stereo Vision Camera
- Bounding Box approach

- Tensor RT
- ROS
- Arduino
- Three controllers

Approach towards Avoidance :

Basically, whenever obstacle (human) comes within bounding box code generate 1's otherwise 0's, these values are transfer from Jetson nano to Arduino using rosserial and then by using UART communication these values are transmitted to main Arduino mega. Obstacle detected once activates function controller2 which will stop UGV and will count value of 1's ,if there sum is up to 35 it will activate function controller3 and obstacle avoidance code will come into action these function will rotate UGV until it start to receive 0's and will stop and will move forward until values of 0's are add up to 11 and if any obstacle is again detected controller3 function will call controller2 again and whole process will repeat. After moving forward till value of 0's is added up to 11 controller1(pose correction) will become active and after correcting path via rotation left or right depending on steering angle, UGV will continue its journey towards waypoints, suppose if obstacle is detected and suddenly moved, then UGV must not move because there are chances that 0 might be published because of missing frames so UGV will move forward only if 0's added are greater than 5.

Chapter 5: SOFTWARE METHODOLOGY

The algorithms used in waypoint guidance in this project have spanned over multiple platforms in the software world. They have been integrated to the best of their abilities and are as described in the following report –

5.1 Implementation of Perception stack

Perception and Understanding of the environment using vision is one of the most challenging task of computer vision. It incorporates different levels of conception. Scene recognition focuses on offering a universal understanding of the environment, commonly sum up at a single scene level. Detection primarily focuses on localizing objects and their respective category within a frame, effectively localized and represented using bounding boxes.

For this task we used a stereo vision camera to detect an object and accurately measure it's distance.

5.1.1 Camera Calibration

The initial step is the camera calibration, which is achieved using a checkboard grid of size 6x9, which is printed on an A4 sheet and the camera matrices are determined. A calib file is created from these observations which is used to undistort any incoming camera images. The calibration process is performed using MATLAB camera calibration toolbox with multiple images of the checkboard takes as dataset. The calibration results are shown below:

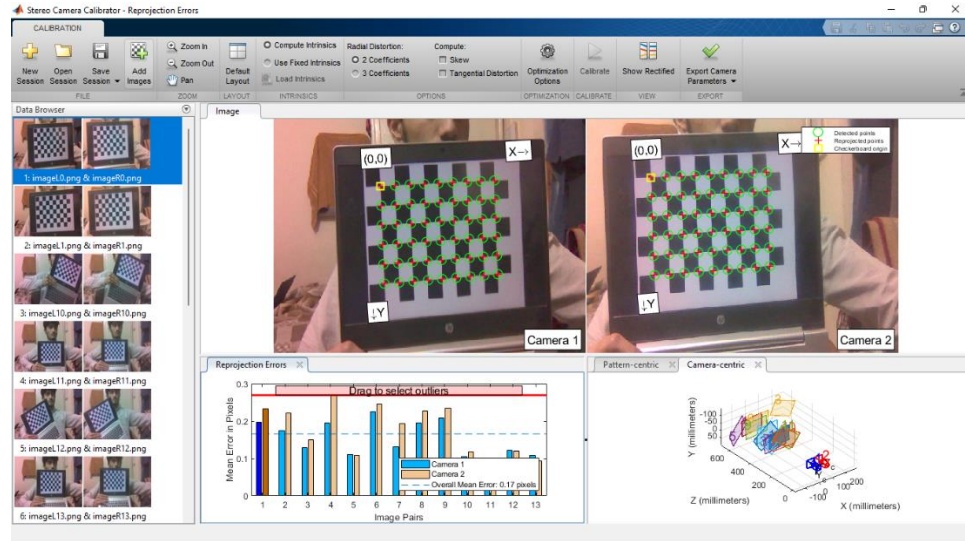


Figure 22 Stereo Camera Calibration MATLAB

5.1.2 Object Detection

Object detection is a fundamental requirement of localization of static and dynamic obstacles in an environment, and an important problem for autonomous systems. The algorithm processes the image obtained through stereo cameras as estimates the depth of the scene by geometry techniques. The use of stereo camera is generally much easier, economical, and it is preferred for affordable operation for autonomous vehicles.

5.1.2.1 SSD-MobileNet-v2

In perception of autonomous robots obtaining accurate object detection with good fps is very crucial. The benchmark model for object detection in 2021 is YOLOR which has a mean average precision of 56.7. It is trained on famous MS COCO dataset. But keeping in view computational complexity of our Nvidia Jetson nano we were unable to achieve good frame rate. So out of all available models ssd mobilenet v2 provided good accuracy along with considerable frame rates.

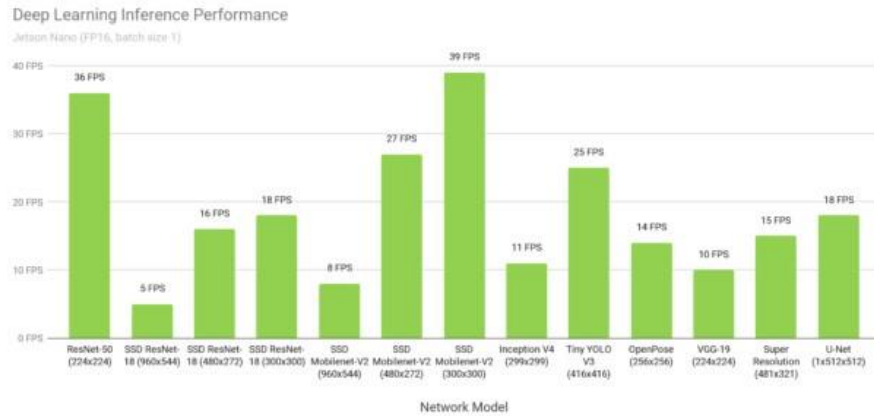


Figure 23 Deep Learning Performance Jetson nano

The framework that is used for the purpose of object detection is SSD-Mobilenet-v2 [34]. It is a deep neural network designed to be light weight because of it's depth-wise convolution filters. It is especially designed for use in mobiles and embedded vision. The vision tasks need to be carried out at good frame rate on a limited specs device it is very crucial in many real-world applications such as a self-driving car.

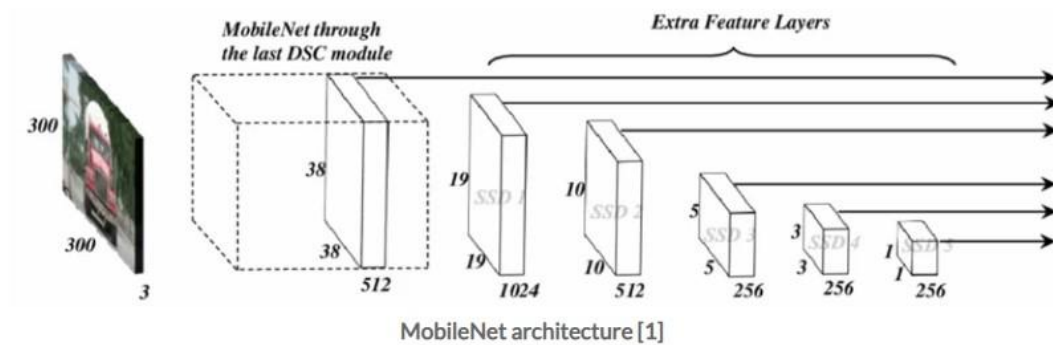


Figure 24 SSD-Mobilenet-v2 architecture

MobileNet contains depth-wise separable convolution filters in it's layers. SSD or Single Shot object detection uses one single look technique instead of passing a template over

learning inference applications, such as object detection. Less precision inference minimizes application latency but increases speed of computational processing that results in high throughput, which is a requirement for applications that are run in real-time, such as autonomous systems.

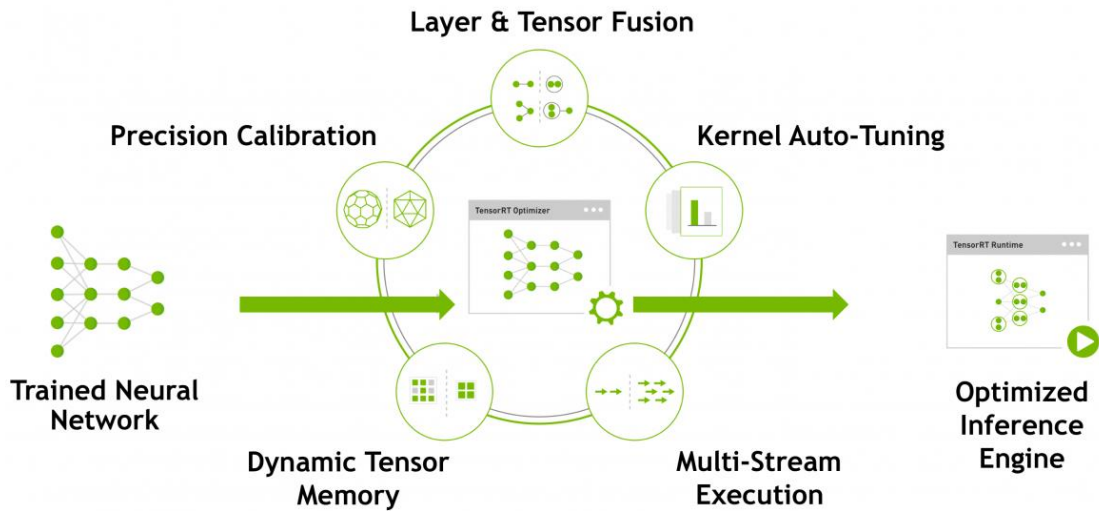


Figure 26 TensorRT optimization

5.1.2.3 Inference of SSD-MobileNet-v2 object detection

We used “dusty-nv” optimized SSD-MobileNet- v2. He used NVIDIA TensorRT for optimizing deployment of SSD v2 onto the Jetson nano, improving performance and power efficiency using low latency, kernel fusion, and FP16/INT8 quantization. The inference of SSD-MobileNet V2 Object Detection is performed on Nvidia Jetson nano, with an inference time of 0.3 seconds. The code is modified to create an inference node from the available test code such that the python code takes real-time feed from the camera instead of loading the images from database. The output of the inference node is shown below indicating the position of the oobstacle bounded by boxes. The output of the object detection is passed to the controller for planning of feasible trajectories to avoid the

obstacle.



Figure 27 Object detection Mobilenet v2 inference

5.2 Graphical User Interface Design

A graphical user interface (GUI) is visual representation of communication for easy interaction with machine. It is a system of interactive display of the complete system. It displays options can be taken by the user and convey information. A system with a well-designed GUI can be used by almost anybody, regardless of how technically savvy the user might be. So, for easy control and monitoring of UGV from work station we designed a graphical user interface in PyQt5. PyQt is a Graphical User Interface toolkit. PyQt is a product of a combination of Python language and Qt library. PyQt comes with Qt designer. We used Qt designer that allowed us to easily design a GUI by just selecting pushbuttons, dialogue boxes and many other options, and then got a python code for that GUI. Then we easily connected callback functions using simple python coding. PyQt supports all platforms including Windows, macOS and UNIX. PyQt can be used to build stylish GUIs, a modern and portable python framework.

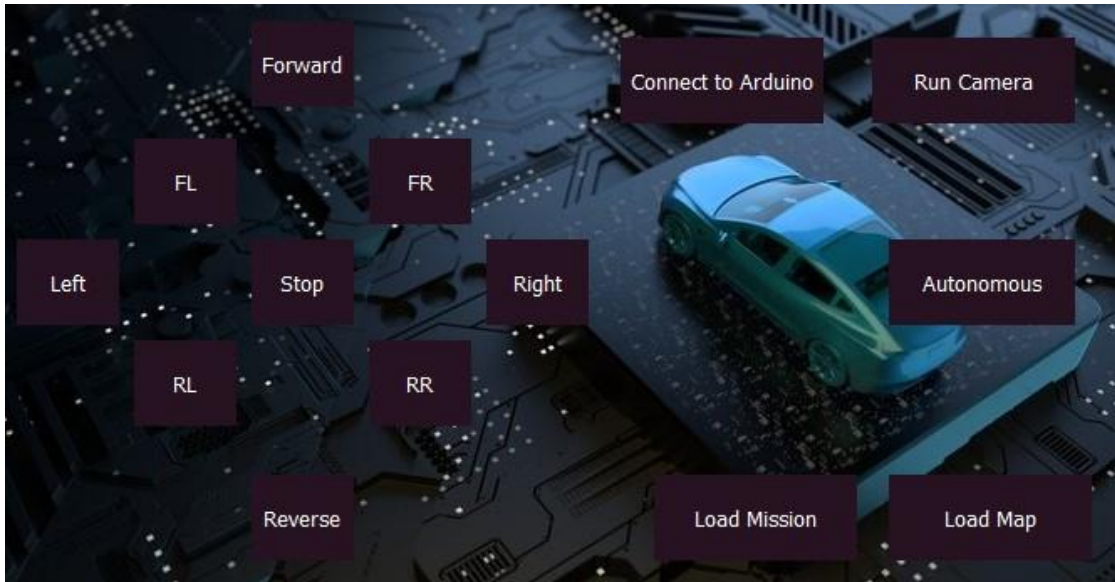


Figure 28 Graphical user interface

5.2.1 RC Control Buttons

Graphical user interface contains 9 buttons for the remote control of the UGV. As our UGV is differential drive it contains some extra buttons such as forward-right and reverse-left etc. for easy and smooth remote operation.



Figure 29 GUI RC control options

5.2.2 Connect to Arduino

This button enables the user to start communication between Nvidia Jetson and main Arduino controller. Nvidia Jetson nano contains waypoints entered by the user and message

that perception stack releases if an obstacle is detected. For this ROS command need to be run on the command prompt to start the communication. This button does this work for the user



Figure 30 GUI serial communication

5.2.3 Run Camera

This button is for remote monitoring of UGV from work station. It enables the user to see a live stream of video being recorded and processed by the perception stack. The user can easily view the environment around the UGV and make appropriate action if there is any haphazard situation.



Figure 31 GUI Camera preview options

5.2.4 Load Map and Load mission

This button opens up a map for the user to entered the desired waypoint. Because of unavailability of free google maps we used map from mission planner of the arduopilot software. When user enters desired waypoint these waypoints are saved in a text file in the system. Then Load mission buttons is used to read these waypoints from the text file and send them to main Arduino controller for waypoint guidance.

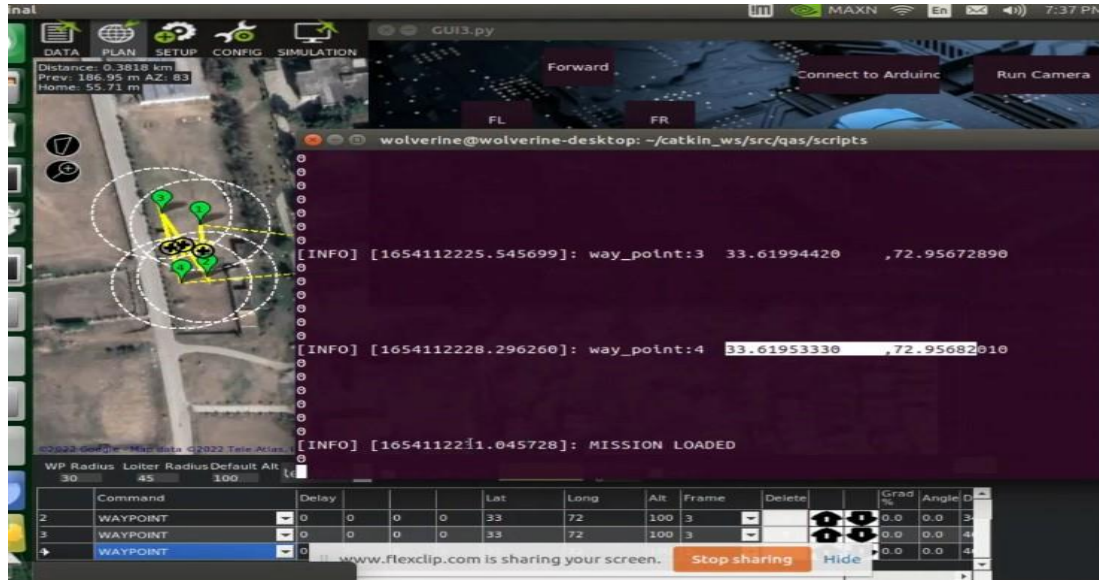


Figure 32 Mission planner through GUI

5.2.5 Autonomous Button

This button starts the autonomous navigation of the UGV. When a set of desired waypoints are loaded to main controller by the user this buttons simply starts the autonomous navigation through those waypoint while obstacle detection can be monitored through the camera preview.

4.6 System integration using ROS

The Robot Operating System is an open-source framework that help researcher and developers build and reuse code between robotic application. Basically, we need bridge between jetson nano and Arduino, and that bridge is ROS.

ROS Version:

In our project we are using ROS-melodic because in jeton nano we are using ubuntu 18.04, So it is better to use Ros-melodic instead of ROS-noetic. ROS-melodic uses python2 so codes and library were used keeping in view that it must be supported by python2.

GUI Integration :

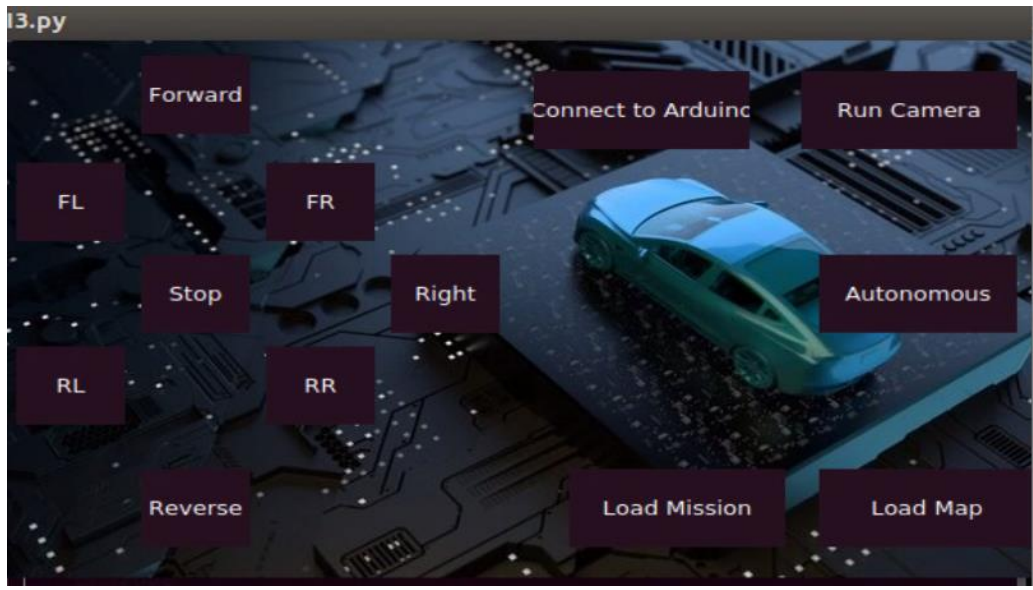


Figure 33 GUI integration using ROS

Approach towards ROS integration:

- ROSCORE
- ROS-Topic
- ROS-Nodes
- Subprocess.Popen

ROSCORE

roscore is a collection of [nodes](#) and programs that are pre-requisites of a ROS-based system. You **must** have a roscore running in order for ROS nodes to communicate. It is launched using the roscore command.

roscore will start up:

- a ROS Master
- a ROS Parameter Server
- a rosout logging node

There are currently no plans to add to roscore.

The roscore can be launched using the roscore executable:

```
roscore
```

Figure 34 ROSCORE launch command

ROS-Topic

Topics are [named](#) buses over which [nodes](#) exchange [messages](#). Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data *subscribe* to the relevant topic; nodes that generate data *publish* to the relevant topic. There can be multiple publishers and subscribers to a topic.

There are three ROS-Topics:

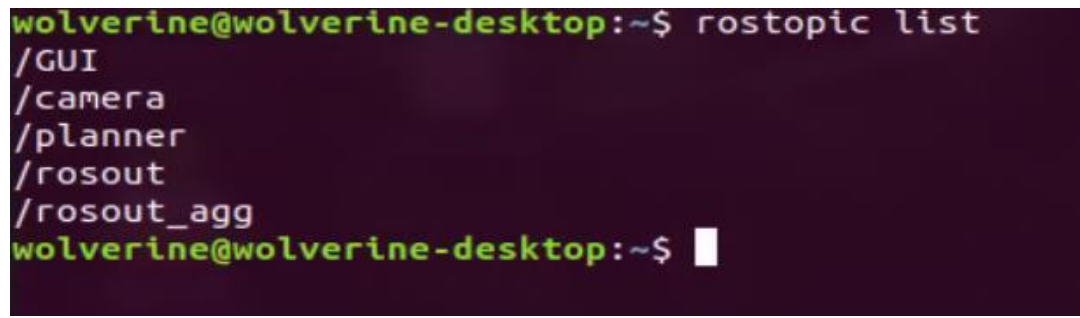
A terminal window with a dark purple background. The prompt is 'wolverine@wolverine-desktop:~\$'. The command 'rostopic list' has been entered, and the output is a list of topics: '/GUI', '/camera', '/planner', '/rosout', and '/rosout_agg'. The prompt is now 'wolverine@wolverine-desktop:~\$' with a cursor.

Figure 35 ROS topics for data communication

Camera Topic :

Topic camera is sending 1 or 0 depending on whether obstacle is detected or not.

```
pub = rospy.Publisher('camera', Int8, queue_size=12)
rospy.init_node('talker', anonymous=True)
```

Planner Topics :

Topic planner is sending targets waypoints from jetson nano to Arduino.

GUI Topic :

Topic GUI is doing teleoperation.

Chapter 6: RESULTS AND DISCUSSION

The code was simulated every step of the way and the results were as expected in the simulations. These simulations were done on MATLAB, Turtlesim, RViz and Gazebo. The algorithm for selecting the waypoints on the browser and the storage of the waypoints was done on a GPU using Mission Planner and ROS. The kinematic model made was simulated on MATLAB on which the system worked as expected. This was made using the odometrical equations as discussed in the experimental techniques above and then simulated on Turtlesim and Gazebo.

Similarly, the Haversine formula and the math for calculating the heading angle were calculated and tested using RViz. This was tested out and worked as expected for the system. The latitudes and longitudes were taken from Google Maps and were within the premises of the university. They are as follows –

1. Premise ATM - 33.62136, 72.95881
2. United Café – 33.62267, 72.96010
3. NCRA – 33.62339, 72.95936
4. Auditorium – 33.62456, 72.96167

Using the above coordinates, the values were a bit different than the values from Google Maps for the distance between the waypoints, but there are enough sources of error for that to be justified. The sources of error will be discussed later on in this chapter.

Similarly, when the code for obstacle avoidance was tested on Gazebo, the blockades were shown as expected and the resolution for the system for that particular situation was also tested and the results came out as expected.

6.1 Sources of Error

The sources of error in this project mostly lie in the software and the hardware used. The GPS – UBLOX NEO 8M– has a smaller accuracy than most GPS modules in the market but had to be used due to compromise with the current situation.

Using the Haversine formula is in itself, a source of error because the formula algorithm assumes that the system is, in fact, a sphere. This is done to promote easier computation over efficiency. The Haversine formula has a 5% rate of inaccuracy due to the globe being an ellipsoid and not a sphere.

Chapter 7: CONCLUSIONS

The waypoint code was made on Arduino IDE and tested using Arduino mega 2560. This was fitted into the chassis of the UGV. GPS sensor was connected to Arduino Uno.

The UGV was Operated through a GUI using a wireless network between the GPU Nvidia Jetson nano and laptop. All the sensors, actuators and GUI were efficiently connected using ROS nodes.

Waypoints were selected using mission planner and stored in a text file. Which was then read and the waypoints values sent to Arduino Mega 2560 using Rosserial. The waypoints were stored and then fed into an algorithm that converts the longitudes and the latitudes to specific Cartesian coordinates using the Haversine formula, and the same is done with the changing location of the UGV.

Using the output from the above algorithm, the heading angle and the distance to the waypoint are calculated. We use the heading angle to give certain values to the motor driver which moves the UGV with different speeds and angles by controlling the velocities of the left and right motor tracks, which in turn allows the UGV to move to that specific waypoint easily.

The wave share IMX219 Stereovision camera was used with Nvidia Jetson nano to detect and calculate depth of obstacle which helps in the autonomous navigation of the UGV so that it can avoid obstacles accordingly.

Chapter 8: FUTURE WORK

With the advancement in technology, especially in the field of autonomous navigation, there is always room for future improvements. In section 7.1, some sources of errors were discussed. These errors were found after rigorous testing. Therefore, improving/removing the errors would greatly improve the UGV's performance and its ability to autonomously carry out its missions.

Employing use of a better GPS sensor would greatly increase the accuracy of the algorithm. As discussed earlier, the code uses the UGV location (as provided by the GPS), to decide its next course of action. So getting a precise location, with the minimum possible lag, would give a performance boost.

When testing, the controller would sometimes disconnect itself from the network.

Therefore, a better router would improve the network connectivity. Similarly, a fail-safe can be added, to turn the motors OFF in case the network disconnects, or the UGV goes out of range. Similarly, an improved connection would reduce the lag between the commands being sent and received (during testing the lag was quite minimal but can be further improved).

The UGV would be deployed in large environments, therefore, improved battery life would increase the working hours of the UGV. This can be achieved by code optimization and using a battery with a greater mAH rating. Moreover, the UGV can be made application-specific by adding any necessary ADD-ONS. For example, a manipulator. This would justify the algorithm's purpose, which is to autonomously complete different tasks. Other than that precise state estimation is very important for UGV autonomous navigation especially in environments in which there is no GPS connectivity. for that implementing

visual inertial odometry is very important.

Powerful GPU is a must for developing autonomous UGV as high computation power is required for developing and running perception stack and VIO in real time.

REFERENCES

- [1] Mahmood, A. and Bicker, R.(2016). *Path Planning, Motion Control, And Obstacle Detection Of Indoor Mobile Robot*.
<https://www.researchgate.net/publication/332082708_Path_Planning_Motion_Control_and_Obstacle_Detection_of_Indoor_Mobile_Robot>
- [2] Campbell, S. (2020). *Steering Control of an Autonomous Ground Vehicle with Application to the DARPA Urban Challenge*. Massachusetts Institute of Technology, pp.1-193.
- [3] *GUIDANCE NAVIGATION AND CONTROL IMPLEMENTATION FOR ...* (n.d).
<https://rc.library.uta.edu/uta-ir/bitstream/handle/10106/26162/SRINIVASAN-THESIS-2016.pdf?sequence=1&isAllowed=y>.
- [4] M. O. S. R. T. R. M. E. R. B. U. F. S. R. a. B. S. M. Cordts, "The Cityscapes Dataset for Semantic UrbanScene Understanding,," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [5] G. a. O. T. a. B. S. R. a. K. P. Neuhold, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [6] A. G. a. P. L. a. R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [7] S. L. R. C. RPK Poudel, "Fast-scnn: Fast semantic segmentation network," *Computer Vision and Pattern Recognition*, 2019.
- [8] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME--Journal of Basic Engineering*, vol. 82, pp. 35-45, 1960.
- [9] F. E. Daum, "Extended Kalman Filters," in *Encyclopedia of Systems and Control*,

2015.

[10] Luca, G. D. (2020, December 13). *Haversine Formula*. Baeldung on Computer Science. <https://www.baeldung.com/cs/haversine-formula>.

[7]

[11]. *Pololu - Sharp GP2Y0A21YK0F Analog Distance Sensor 10-80cm*. Pololu Robotics & Electronics. (n.d.). <https://www.pololu.com/product/136>.

[12]. "NEO-6 Series, NEO6M." *u*, 21 June 2019, www.u-blox.com/en/product/neo-6-series.

[13] "HMC5883L Magnetometer Module Pinout, Features, Specs & Brief Overview." *Components101*, components101.com/ics/hmc5883l-magnetometer-module.

[14] "Differential Drive - Robotics Toolbox - MATLAB." *Differential-Drive Vehicle Model - MATLAB*, www.mathworks.com/help/robotics/ref/differentialdrivekinematics.html.

[15] Bleything, Talesa. "Reading Magnetometer Data ." *Digilent Blog*, 10 June 2021, blog.digilentinc.com/how-to-convert-magnetometer-data-into-compass-heading/.

[16] "Creating a Simulation Scene (Robotics Module)." *Creating a Simulation Scene (Robotics Module) - LabVIEW 2014 Robotics Module Help - National Instruments*, zone.ni.com/reference/en-XX/help/372983F-01/lvrobogsm/robo_sim_creating/.

[17] "Optimal Waypoint Guidance, Trajectory Design and Tracking." *IEEE Xplore*, ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6579936.

[18] Capello, Elisa, et al. "A Waypoint-Based Guidance Algorithm for Mini UAVs." *IFAC Proceedings Volumes*, Elsevier, 21 Apr. 2016, www.sciencedirect.com/science/article/pii/S1474667015402812.

[19]. "NEO-6M GPS Module - An Introduction." *ElectroSchematics.com*, 4 Apr. 2019, www.electroschematics.com/neo-6m-gps-module/.

[20] A. Kelly, "Mobile Robotics Mathematics, Models and Methods", "

Cambridge University Press, 32 Avenue of the Americas, New York NY 10013-2473, USA, "Kinematics of Wheeled Mobile Robots", 2013.

[21] J. R. Nikolic, "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes," in *Conference: 2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[22] L. W. M. L. Yuxuan Liu, "YOLOStereo3D: {A} Step Back to 2D for Efficient Stereo 3D Detection," *CoRR*, 2021.

[23] S. L. R. C. RPK Poudel, "Fast-scnn: Fast semantic segmentation network," *Computer Vision and Pattern Recognition*, 2019.

[24] J. S. P. L. X. W. a. X. T. Xingang Pan, "Spatial As Deep: Spatial CNN for Traffic Scene Understanding," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[25] Z. a. W. H. a. L. X. Qin, "Ultra Fast Structure-aware Deep Lane Detection," in *The European Conference on Computer Vision (ECCV)*, 2020.

[26] N. Amer, K. Hudha and H. Zamzuri, " "Adaptive modified Stanley controller with fuzzy supervisory system for trajectory tracking of an autonomous armoured vehicle", "*Robotics and Autonomous Systems*, DOI: 10.1016/j.robot.2018.03.006, April 2018.

[27] "Introduction Self Driving Cars," Coursera, University of Toronto, [Online]. Available: <https://opencourser.com/course/klq6i8/introduction-to-self-driving-cars>.

[28] A. Kelly, "“A Feedforward control approach to Local Navigation problem for Autonomous Vehicles”, "*The Robotics Institute Carnegie Mellon University*, CMU-RI-TR-94-17..

[29] H. Wu, S. Cheng and S. Cui, "A Controller for Brushless DC motor for Electric Vehicle," *IEEE Transactions on Magnetics*, vol. 41, no. 1, January 2005.

[30] M. Park, S. Lee and W. Han, "“ Development of Steering Control System for Autonomous Vehicle Using Geometry-Based Path Tracking Algorithm”, "*ETRI Journal*, Volume 37, Number 3, June 2015.