Development of a platform for evaluating a GNSS positioning system

Simon Gollbo
Abstract

Development of a platform for evaluating a GNSS positioning system

Simon Gollbo

The project aims to develop a platform for the demonstration and evaluation of Global Navigation Satellite Systems (GNSSs). The GNSS receivers used in the project are two uBlox NEO-M8P modules, one of which is used as a reference receiver in a differential GNSS setup.

The entire platform has been implemented and consists of a network-controlled car (NetCar) which has a GNSS receiver and a 4G module on it, a separate reference GNSS receiver, a computer hosting a control interface for controlling the NetCar and a relay server for relaying data between the control interface and the NetCar.

The platform has been tested for evaluating the GNSS. The performance was evaluated in terms of the time to first fix (TTFF), stationary positioning accuracy and tracking accuracy which includes qualitative evaluation of accuracy, statistical evaluation of accuracy and evaluation of absolute accuracy after moving. The test results show that the platform performed well for the demonstration and evaluation of the GNSS. The evaluation results show that the TTFF for the uBlox NEO-M8P used in the project averaged 24.2 s with a worst case of 33 s. The stationary positioning accuracy was found to drift in the order of tens of centimeters when using differential GNSS in RTK float mode and in the order of centimeters in RTK fixed mode. The relative tracking accuracy between points during either RTK mode was found to be on the order of centimeters while the absolute accuracy was found to be drifting, probably due to the reference receiver position being acquired through self-survey. The current project can be extended further by including automatic navigation, sensor fusion, pathfinding and the like.
Populärvetenskaplig sammanfattning

I samhället idag kan en trend att automatisera system och processer observeras. Arbete som tidigare utförts manuellt av människor utförs nu i större utsträckning av självgående maskiner eller flottor av flera maskiner som styrs på håll av en ensam människa. För att genomföra ett arbete automatiskt eller med begränsad hjälp av människan krävs bland annat att maskinerna har information om var de befinner sig för att kunna ta beslut om var de bör röra sig häpnäst.


Denna rapport beskriver hur ett GNSS kan demonstreras och utvärderas. Den beskriver också hur ett system konstruerades för att åstadkomma demonstration och utvärdering av GNSS i form av en nätverksstyrda bil, d.v.s. en bil som styrs via uppkoppling till internet. I projektet som rapporten beskriver utvecklades också programvaran som körs ombord på bilen samt programvaran för styrsystemet som användaren sitter vid, vilket körs på en dator. Slutligen beskriver rapporten hur den nätverksstyrda bilen användes för att demonstrera och utvärdera ett specifikt GNSS genom olika tester.
Nomenclature

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC</td>
<td>Electronic Speed Controller</td>
</tr>
<tr>
<td>GNSS</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>RTK</td>
<td>Real Time Kinematic</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>1.1</td>
<td>Background</td>
</tr>
<tr>
<td>1.2</td>
<td>Purpose statement</td>
</tr>
<tr>
<td>1.3</td>
<td>Project specifications and tasks</td>
</tr>
<tr>
<td>1.4</td>
<td>Report outline</td>
</tr>
<tr>
<td>2</td>
<td>Global Navigation Satellite System and Devices</td>
</tr>
<tr>
<td>2.1</td>
<td>Global Navigation Satellite System</td>
</tr>
<tr>
<td>2.2</td>
<td>Least-squares estimation and Kalman filtering</td>
</tr>
<tr>
<td>2.3</td>
<td>GNSS Positioning Devices</td>
</tr>
<tr>
<td>2.4</td>
<td>GNSS Augmentations and DGNSS</td>
</tr>
<tr>
<td>2.5</td>
<td>Map projection</td>
</tr>
<tr>
<td>2.6</td>
<td>GNSS Evaluation Criteria</td>
</tr>
<tr>
<td>3</td>
<td>Computer and mobile communications</td>
</tr>
<tr>
<td>3.1</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>3.2</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>3.3</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>3.4</td>
<td>Multitasking and multithreading</td>
</tr>
<tr>
<td>3.5</td>
<td>4G</td>
</tr>
<tr>
<td>4</td>
<td>Implementation</td>
</tr>
<tr>
<td>4.1</td>
<td>Overview of the platform</td>
</tr>
<tr>
<td>4.2</td>
<td>Control interface (A)</td>
</tr>
<tr>
<td>4.3</td>
<td>Relay server (B)</td>
</tr>
<tr>
<td>4.4</td>
<td>GNSS receivers</td>
</tr>
<tr>
<td>4.5</td>
<td>NetCar (D)</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Raspberry Pi</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Connectivity</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Webcam</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Power supply</td>
</tr>
<tr>
<td>4.5.5</td>
<td>Direct Current motors</td>
</tr>
<tr>
<td>4.5.6</td>
<td>Car chassis</td>
</tr>
<tr>
<td>4.6</td>
<td>Development tools</td>
</tr>
<tr>
<td>4.7</td>
<td>Control interface (A) software</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Graphical user interface</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Back-end communications</td>
</tr>
</tbody>
</table>
4.8 NetCar onboard computer software.................................................................28
4.8.1 Video streaming.............................................................................................28

5 Results and discussions.........................................................................................30
5.1 NetCar ...............................................................................................................30
5.2 Relay server........................................................................................................30
5.3 uBlox NEO-M8P performance............................................................................31
  5.3.1 Fix time .......................................................................................................31
  5.3.2 Stationary accuracy......................................................................................31
  5.3.3 Tracking accuracy.......................................................................................33
5.4 Further development..........................................................................................34
  5.4.1 Automatic navigation..................................................................................34
  5.4.2 Inertial measurements and sensor fusion ....................................................35
  5.4.3 Pathfinding..................................................................................................35
  5.4.4 Battery monitoring......................................................................................35
  5.4.5 Area mapping..............................................................................................35

6 Conclusions..........................................................................................................36
References..................................................................................................................37
1 Introduction

1.1 Background
In a world where tasks are increasingly automated and where manual labor is replaced by control centers where multiple remote-controlled devices carry out the physical work there is a need to accurately determine the position of automated and remote-controlled devices. Using accurate position data enables fully automatized fleets of devices to carry out tasks independently as well as large scale supervised control where a human controller is able to monitor a fleet of devices single-handedly.

One way to track the position of an object is to use Global Navigation Satellite Systems (GNSSs) that utilize signals transmitted by satellites for positioning and tracking. The American Global Positioning System or GPS as a term has become equated with positioning in daily use. GPS as a system has evolved, along with the advent of smartphones, from a military and aviation positioning system to a day-to-day civilian positioning system, for example able to help users navigate through a new city.

In addition to the American GPS, other nations have also developed and deployed their own GNSSs, for instance, the Russian Globalnaya Navigatsionnaya Sputnikovaya Sistema (GLONASS), the Chinese BeiDou Navigation Satellite System (BDS) and the European Galileo. GLONASS and BeiDou are fully operational whereas Galileo provides initial services but is not yet at full operational capacity. [1]

The precision of GNSSs has been improved from being able to estimate a position within a radius in the order of tens of meters to now being able to provide accuracy in the order of centimeters with different augmentation techniques. This increase in accuracy enables applications that were previously impossible. With, for instance, centimeter level accuracy anything from automated survey of crops to automated quarry operations becomes possible.

The company Syntronic wishes to have a method for demonstrating and evaluating GNSS receivers. Of particular interest to Syntronic is the uBlox NEO-M8P differential GNSS receiver, which promises centimeter level accuracy when used in differential mode.
1.2 Purpose statement
The purpose of this thesis project is to investigate how to evaluate and demonstrate the positioning and tracking capabilities of GNSSs. In addition, the results from this investigation are to be used to realize a system intended for the evaluation and demonstration of the positioning and tracking capabilities of a specific GNSS receiver, which is to simultaneously receive GPS and GLONASS signals.

To evaluate a GNSS, some criteria of evaluation need to be established. These criteria may vary depending upon the intended application for the GNSS. For the purposes of this thesis, however, some criteria of general interest are to be suggested and discussed.

For an observer to appreciate, i.e. to demonstrate, the capabilities of a GNSS it is necessary to visualize GNSS’s real position in relation to the GNSS’s own estimation of where it is. This should be performed while the GNSS is stationary as well as when the GNSS is moving.

1.3 Project specifications and tasks
In Figure 1.1 an overview of the project is shown. The overview outlines the workflow of the project (in the dashed rectangle on the upper left) and the solution for evaluating and demonstrating GNSS, a network-controlled car (in the solid rectangle on the right). The realization of the solution, namely the development of the network-controlled car, constitutes the main bulk of the project.

Figure 1.1: Block overview showing the workflow of the project and introducing the chosen solution for demonstrating and evaluating a GNSS, a network-controlled car.
From the workflow shown in Figure 1.1, the project starts with research into GNSS and how to demonstrate and evaluate the positioning capabilities of a GNSS. Taking this research into account, a hardware platform is chosen. In this project, a remote-controlled car is chosen as the base hardware platform on which the rest of the system is mounted. The system in its entirety is referred to as a network-controlled car or NetCar and consists of the remote-controlled car and a control application.

The whole project involves theoretical study of GNSS and practical work associated with hardware and software. The specific tasks include:

- Research on GNSS and related topics, including map projection
- Research on technical aspects of the platform, including network communication protocols and concurrent programming
- Drafting a bill of materials, including a GNSS receiver
- Implementation of the software for the onboard computer on the NetCar
- Implementation of the software for the control interface
- Implementation of the software for the relay server
- Interfacing hardware and software functionality on the NetCar
- Testing of software implementations and functionality of hardware in conjunction with software
- Assembly of the NetCar
- Using the system to gather data on the performance of the GNSS receiver

1.4 Report outline
The report first introduces the theoretical and technical aspects of how a GNSS works as well as a system overview of GNSSs. This is then used to formulate the GNSS evaluation criteria to be used in the project which concludes the first section.

The second section explains some theoretical aspects of the computer and mobile communications underlying the functionality of the network-controlled car.

The third section outlines the practical development of the network-controlled car and its implementation. It applies the theory explained in the previous section.

This is followed by a section on results when testing a specific GNSS using the network-controlled car as well as discussions about what could be improved.

The report ends with conclusions drawn from the project.

2 Global Navigation Satellite System and Devices

2.1 Global Navigation Satellite System
A Global Navigation Satellite System (GNSS) is a system that consists of a constellation of satellites to provide global positioning. Currently, close to 100 GNSS satellites orbit the Earth at altitudes between 19,000 and 23,000 km. A small-sized electronic receiver associated with the GNSS uses timed signals transmitted from the satellites to estimate the position of the object the receiver is attached to. The satellites continuously transmit so-called almanacs, which contain rough orbital data on the entire satellite constellation, and ephemerides (ephemeris in singular),
which contain precise information on the transmitting satellite’s orbit. The almanac is the same for all satellites in a constellation, while the ephemeris is unique to each satellite. The roughness of the almanac means it is considered valid for a much longer time than the ephemeris, which needs to be reacquired on a regular basis to maintain a positional fix. [2]

To achieve the high precision in time that is required for accurate positional fix, the satellites are equipped with extremely precise atomic clocks that are designed to correct for relativistic effects. The reduced gravity field intensity as well as the orbital speed are compensated for, which would otherwise cause clock drifts. The high precision timing is used for positioning by measuring the time it takes for a signal from a satellite, travelling at close to the speed of light, to reach the receiver. This information is used to calculate the distance to each satellite. Using multiple such distance measurements along with precise information on where the satellites are located allows for estimation of the receiver position.

Consider first an ideal scenario, where the distance to each satellite can be determined with no error. A single satellite allows a receiver to establish that it is located at any point on a sphere whose center is the satellite and whose radius is the distance between the receiver to the satellite. Having access to measurements from two satellites allows the receiver to know it is located at any point on the circle which is the intersection of two spheres whose centers are the two satellites, as shown to the left in Figure 2.1. Having access to a third satellite, the receiver can further narrow its position down to two distinct possibilities. This is shown to the right in Figure 2.1.

Figure 2.1: The actual receiver position is marked by the black X.
Left: Having access to two satellites allows the receiver to narrow its position down to the circle intersecting the two spheres.
Right: Having access to three satellites allows the receiver to narrow its position down to two distinct positions at the intersection points of the third sphere with the circle established using two satellites.

With access to three satellites the receiver is typically able to determine its location when it is located within the Earth’s atmosphere. Access to a fourth satellite is, however, still required to achieve a three-dimensional fix. The fourth satellite is used to synchronize the receiver’s clock to the atomic clocks on-board the satellites. The clock synchronization enables precise timing of the signal travel time, which is essential to accurately calculate the distances to the satellites.
In reality, however, the measurements of the signal travel time cannot be 100% accurate because of influence from more factors than clock synchronization. Signal interaction with the different layers of the atmosphere and noise in the receiver electronics are two examples of such factors. Some of these factors can be somewhat corrected for, but it is impossible to get 100% correction. This leads to an uncertainty in the distances calculated and in turn in the determined position of the receiver. To reduce the uncertainty in the determined position, as many satellites as possible should be used once they are available and the calculated distances weighted together.

2.2 Least-squares estimation and Kalman filtering

GNSS generally uses Kalman filtering or a least-squares approach to estimate the position of the GNSS receiver. Kalman filtering is more common. Both methods allow multiple measurements to be used in calculations to achieve greater accuracy in the position estimation. [3]

Kalman filtering and the least-squares approach can both be described as estimating a state vector. In the case of positioning the states typically represent the position of the receiver. The methods have in common that they estimate the most probable value for the state vector at a given time. What value is considered most probable varies slightly between the approaches due to differing mathematical formulations. The following theoretical description is based on [4] and [5].

We consider that state vector $\hat{x}_i$ is to be determined from the measure data $\tilde{z}_i$, and they are related in the following manner

$$\tilde{z}_i = H_i \hat{x}_i + \tilde{v}_i$$

(2.1)

where $H_i$ is the system transfer matrix describing the measurement system and $\tilde{v}_i$ represents the measurement errors that occur at time instance $i$.

The measurement errors $\tilde{v}_i$ are assumed to have zero mean. Let $\hat{x}$ be the estimate of the state vector. The residuals of the estimate are then defined as

$$\tilde{r}_i = \tilde{z}_i - H_i \hat{x}_i$$

(2.2)

where $\tilde{r}_i$ represents the residuals from the estimation. The residuals should be interpreted as the error in the estimated state vector $\hat{x}_i$. The least squares approach minimizes the square of the residuals, thus minimizing the error between the measurements $\tilde{z}_i$ and the predicted measurement $H_i \hat{x}_i$. This is expressed as

$$\hat{x}_{LS,i} = \arg\min_{\hat{x}_i} \left( \tilde{z}_i - H_i \hat{x}_i \right)^T \cdot \left( \tilde{z}_i - H_i \hat{x}_i \right)$$

(2.3)

where $\hat{x}_{LS,i}$ is the least-squares estimate of the state vector. Minimizing the right-hand side expression in Eq. (2.3) is done by differentiating Eq. (2.3) with respect to $\hat{x}_i$ and finding where it’s equal to 0. The $\hat{x}_i$ that yields the differentiated function equal to 0 is the least-square estimate of $\hat{x}_i$, denoted $\hat{x}_{LS,i}$. The resulting expression can be differentiated a second time to ensure that the $\hat{x}_i$ found indeed yields the function minimum.

The Kalman filter takes a different approach. Assuming again the measurements are related to the state as described in Eq. (2.1), the Kalman filter assumes the system dynamics can be described as

$$\hat{x}_{i+1} = \Phi_{i+1} \hat{x}_i + \bar{w}_i$$

(2.4)
where $\tilde{x}_{t+1}$ and $\tilde{x}_i$ are the system states at time instance $i + 1$ and $i$ respectively, $\Phi_{t+1}$ is the system transition matrix at time instance $i + 1$ and $\tilde{w}_i$ is a white-noise sequence.

The Kalman filter models the measurement errors $\{\tilde{v}_i\}$ as a white-noise sequence. The assumption is made that the white-noise sequences $\{\tilde{w}_i\}$ and $\{\tilde{v}_i\}$ are independent, have zero mean and variances

$$E[\tilde{w}_j \tilde{w}_k^T] = Q_j \delta_{j,k}$$  \hspace{1cm} (2.5)$$

$$E[\tilde{v}_j \tilde{v}_k^T] = R_j \delta_{j,k}$$  \hspace{1cm} (2.6)$$

where $\delta_{j,k}$ is the Kronecker delta.

The Kalman filter works by performing an a priori estimate which is compared to the next measurement. The degree of agreement between the a priori estimation and the measurement influences an adjustment term which weights the discrepancy relative to the measurement and model variance and adds it to the a priori estimate to form an a posteriori estimate. The update equation is expressed as

$$\hat{x}_{i}^{a \text{ posterior}} = \hat{x}_{i}^{a \text{ priori}} + K_i (\tilde{z}_i - H_i \hat{x}_{i}^{a \text{ priori}})$$  \hspace{1cm} (2.7)$$

where $K_i$ is the Kalman gain. The optimal Kalman gain is found by minimizing the a posteriori variance, i.e. minimizing the uncertainty in the a posteriori estimate. To perform this minimization, the covariance matrix $P_{\tilde{x}_{i}i-1}$ of the error in the estimated state at time instance $i$ given measurements up to time step $i - 1$ is introduced as

$$P_{\tilde{x}_i i-1} = E \left[ (\tilde{x}_i - \hat{x}_{i|i-1})(\tilde{x}_i - \hat{x}_{i|i-1})^T \right] = \Phi_{i|i-1} P_{\tilde{x}_i i-1} \Phi_{i|i-1}^T + Q_i$$  \hspace{1cm} (2.8)$$

where $P_{\tilde{x}_i i}$ is the covariance matrix of the error in the estimated state at time instance $i$ given measurements up to time instance $i$. It is expressed as

$$P_{\tilde{x}_i i} = E \left[ (\tilde{x}_i - \hat{x}_{i|i})(\tilde{x}_i - \hat{x}_{i|i})^T \right] = P_{\tilde{x}_i i-1} - K_i H_i P_{\tilde{x}_i i-1}$$  \hspace{1cm} (2.9)$$

Returning to the Kalman gain, the optimal Kalman gain $K_i$ is then given as

$$K_i = P_{\tilde{x}_i i-1} H_i^T (H_i P_{\tilde{x}_i i-1} H_i^T + R_i)^{-1}$$  \hspace{1cm} (2.10)$$

which minimizes the a posteriori variance.

Summarizing the Kalman filtering process, it consists of performing an a priori estimation of what the system state is expected to be in the next time step. Next a measurement of the system is input. Using the measurement and estimation error covariances, a weighting is chosen that minimizes the a posteriori error covariance. Using the chosen weighting, an a posteriori estimate of the system state is produced.

In the summaries of the least-squares approach and the Kalman filter above, the system state has been assumed to be linearly related to the measurements of the system. In a non-linear case, linearization is often utilized to be able to apply the above principles.
2.3 GNSS Positioning Devices

A GNSS positioning device is a GNSS receiver for receiving GNSS satellite signals. The receiver can be constructed to receive signals from a specific GNSS system, such as GPS, or to receive signals from multiple GNSS systems. As established in section 2.1, utilizing signals from additional satellites can increase the accuracy of the position estimation.

A minimum requirement for a receiver to be able to receive GNSS signals is an antenna able to receive signals in the correct spectrum. The GNSS data signal is superimposed on top of a carrier wave, as is common practice in radio communication. The carrier wave is removed using down conversion and the transmitted data signal is extracted and digitized. The data signal is then subjected to digital signal processing to determine the data content of the signal. The different GNSS constellations use different carrier waves to transmit their signals, meaning a flexible receiver should be provided with an antenna capable of receiving at multiple frequencies.

Due to the uncertainties described in section 2.1, the position of the receiver is calculated as a minimization of a set of error equations. Given that the receiver is able to receive data from a sufficient number of satellites, the solution to this set of equations converges to a positional fix. The variance of the solution can be reduced by a larger number of measurements.

Examples of GNSS receivers available on the market is the uBlox NEO-M8P and the NVS Technologies NV08C-RTK. Both receivers have in common the capacity for differential GNSS operation, outlined in the next section. Their technical specifications differ slightly, with the uBlox NEO-M8P having 72 channels, meaning it can receive signals from 72 satellites simultaneously, as compared to the NV08C-RTK's 32 channels. They promise similar levels of accuracy in their respective documentations. Both receivers are available mounted in evaluation kits.

2.4 GNSS Augmentations and DGNSS

To improve the functionality of the base GNSS system, various augmentations have been proposed and developed. For instance, a ground-bound approach to improving the positional accuracy reported by the positioning system is the differential GNSS with Real-Time Kinematics (RTK).

Differential GNSS (DGNSS) utilizes two separate receivers that can communicate. One type of differential GNSS setup consists of a moving part, a rover, and a stationary reference receiver with a known position. The rover determines its position relative to the stationary reference receiver and adds the reference receiver position to determine its absolute position. This differential mode of operation eliminates some of the constant error sources present in the position estimates produced by normal GNSS operation. [6]

The reference receiver can estimate its position using the GNSS signals and compare the resulting position to its known position. The resulting difference is a result of various error sources, such as signal distortion and small inaccuracies in the satellite atomic clocks. The reference receiver can then compute corrections for these factors and transmit them to the rover. The correction terms are typically transmitted using one of the Radio Technical Commission for Maritime services (RTCM) standardized protocols to ensure device interoperability. The protocols contain a host of different message types, including messages for correcting different types GNSSs.
The RTK method attempts to calculate the number of wavelengths between the receiver and the transmitting satellite. To be able to perform this calculation, the correction terms transmitted by the reference receiver are used to resolve phase ambiguities in the satellite signal. If enough satellites are available, the ambiguities can be resolved and the number of wavelengths fixed to an integer, giving the highest accuracy. This is called RTK fixed mode. If fewer satellites are available, the ambiguities cannot be resolved fully, but RTK can still be used for improved positioning. This less accurate mode of operation is called RTK float mode. [6]

The greater the distance between reference receiver and rover, the greater the difference in the distortions to the GNSS signals received by either party. If the rover is within a radius of about 10 to 20 km from the reference receiver, the distortions are similar enough that the correction terms will be useful in correcting the positional estimate. [7]

If the exact position of the reference receiver is unknown, it can be made to survey its own position. This results in a less accurate absolute position correction for the rover, but the rover’s relative position can still be determined. [8]

2.5 Map projection

The GNSS positioning device used in the project returns the position of the device in terms of latitudes and longitudes. For a position on a sphere, the latitude describes the north-south position. It is the angle measured between the equatorial plane and a vector pointing from the center of the sphere to the position. The longitude of the position is the angle of rotation around the line connecting the north and south poles of the sphere, describing the east-west position on the sphere. In Figure 2.2 a sphere with the latitude and longitude angles for some position on the sphere is shown.

![Figure 2.2](image)

Figure 2.2: The latitude ($\phi$) and longitude ($\lambda$) angles of a position on the sphere surface.
The coordinates of the device are to be displayed on a user interface, i.e. a 2-dimensional plane. On this plane, a map of the surrounding region is also to be displayed. For the reported coordinate latitudes and longitudes of the device to map correctly to the flat map displayed on the user interface, projection must be used.

The maps retrieved for displaying on the user interface are from the Google Maps API. Google Maps uses the Mercator projection to map the ellipsoidal (almost spherical) earth onto a cylinder. Because of this projection some features are distorted, notably toward the poles distances in the longitudinal direction are drawn out.

A great circle on a sphere is defined as a circle which has the same radius as the sphere, encircling the sphere at its widest points. The shortest distance between two points on a sphere is the great circle distance between them. The great circle distance is defined as follows: Given two points on a sphere, a great circle can be constructed that contains both points. The great circle distance between the points is the length of the arc section of the great circle that has the two points in question as endpoints.

To calculate the great circle distance, the angle between the two different points needs to be determined. Assume the angle between two points \( A \) and \( B \) on a sphere to be \( \Delta \alpha \), their latitudes \( \phi_A \) and \( \phi_B \), respectively, and their longitudes \( \lambda_A \) and \( \lambda_B \), respectively. Because a right-hand oriented spherical coordinate system is easier to work with, the colatitude is used when describing the points in place of the latitude. The colatitude is defined as the right angle complement to the latitude, i.e. the difference between \( \pi/2 \) and the latitude. The spherical coordinate of a point on a sphere takes the form \((r, \text{colatitude}, \text{longitude})\). In this case, the spherical coordinates of points \( A \) and \( B \) are assumed to have the following forms, respectively,

\[
A = \left(r, \frac{\pi}{2} - \phi_A, \lambda_A \right) \tag{2.11}
\]

\[
B = \left(r, \frac{\pi}{2} - \phi_B, \lambda_B \right) \tag{2.12}
\]

The spherical coordinates can be transformed to Cartesian coordinates using the formulae as follows

\[
x = r \sin \left(\frac{\pi}{2} - \phi \right) \cos(\lambda) = r \cos(\phi) \cos(\lambda) \tag{2.13}
\]

\[
y = r \sin \left(\frac{\pi}{2} - \phi \right) \sin(\lambda) = r \cos(\phi) \sin(\lambda) \tag{2.14}
\]

\[
z = r \cos \left(\frac{\pi}{2} - \phi \right) = r \sin(\phi) \tag{2.15}
\]

Assuming \( r = 1 \) the Cartesian coordinates of points \( A \) and \( B \), respectively, become

\[
A = (\cos(\phi_A) \cos(\lambda_A), \cos(\phi_A) \sin(\lambda_A), \sin(\phi_A)) \tag{2.16}
\]

\[
B = (\cos(\phi_B) \cos(\lambda_B), \cos(\phi_B) \sin(\lambda_B), \sin(\phi_B)) \tag{2.17}
\]

The angle between \( A \) and \( B \) can now be determined using the vector scalar product in the following manner
\[ A \cdot B = |A||B| \cos(\Delta \alpha) \rightarrow \Delta \alpha = \cos^{-1}(A \cdot B) \]
\[ = \cos^{-1} \left( \cos(\phi_A) \cos(\phi_B) \cos(\lambda_A - \lambda_B) + \sin(\phi_A) \sin(\phi_B) \right) \]  

(2.18)

The great circle distance \( d \) between \( A \) and \( B \) on a sphere of radius \( r \) is calculated as \( d = r \cdot \Delta \alpha \).

This formula has a limitation when implemented on a computer system which inherently has limited numerical accuracy during float point operations, namely when calculating small distances (small relative to the radius of the Earth, in the order of a few meters), the round-off error can be magnified by the inverse cosine operation. This problem can be avoided by using the so-called haversine formula, detailed in [9].

An additional inherent limitation in the accuracy of both previously mentioned methods for calculating the distance between two points is the assumption that the Earth is spherical, leading to accuracy problems at sub-meter distances. More advanced formulae can be used to overcome this limitation. One such formula is Vincenty’s formula, which instead assumes an ellipsoidal shape.

Vincenty’s formula consists of a set of equations which are iterated until the change in the distance calculated is sufficiently small between iterations. The formula, formulated in 1975, is optimized for computer systems with limited storage capacity and computing power. The size of the change in distance between iterations is correlated to the accuracy in the calculated distance. [10]

2.6 GNSS Evaluation Criteria

The performance parameters of interest when evaluating a GNSS can be derived from the Required Navigation Performance (RNP) that is placed upon navigation systems to be used in airborne applications. RNP defines four parameters that are required to guarantee safe operations. These parameters are: Accuracy, availability, continuity and integrity. Specific to GNSS, the time it takes the receiver to estimate its position to within a certain level of accuracy after being switched on is also of interest. This is referred to as Time To First Fix (TTFF). [11]

This project will focus on the accuracy and the TTFF parameters. The accuracy, as in how well the position estimated by the GNSS receiver coincides with the actual position, will be evaluated both with a stationary receiver and a moving receiver. Also, the accuracy of regular GNSS as compared to differential mode GNSS will be evaluated.

3 Computer and mobile communications

3.1 Transmission Control Protocol

The Transmission Control Protocol (TCP) was first described in 1974 as a way of reliably transmitting data over an unreliable packet switched network, with an example of such a network being the public internet. The functionality of this initial iteration of TCP is described in RFC675. This initial description was later added to by the Defense Advanced Research Projects Agency (DARPA) internet program in 1981 in the document RFC793. While additional extensions and improvements have been made over the years, the core functionality has remained as described in these two documents. [12], [13]
TCP implementations are expected to provide basic data transfer, reliability, flow control, multiplexing, connections as well as precedence and security options. In the next paragraph, the Java implementation of TCP is discussed.

The data transfer of TCP works in terms of streams. Streams transfer octets of data in both directions between two hosts and TCP ensures that all pieces of data that are transmitted are received. This is done using acknowledgements of received data and retransmissions will be utilized until both hosts are certain that all data has been transmitted and received correctly. To ensure correctness of received data, checksums are used. TCP also rearranges out of order data to ensure that the data received is ordered the same way as it was sent.

3.2 User Datagram Protocol

The User Datagram Protocol (UDP) aims to provide a “datagram mode of packet-switched computer communication” [14] and is defined in RFC768. A program utilizing UDP constructs a datagram packet and sends it off on the network. UDP does not guarantee that the datagram packet arrives at its destination. If for some reason the datagram packet is dropped during routing to the destination, UDP does not make the sender aware of this fact. The minimal nature of UDP means less communication overhead is part of the communication protocol itself.

Routers have a maximum packet size that will be routed. If a router receives a packet larger than its maximum packet size it will fragment the packet. A packet will not be routed to the next link in the route path until an entire packet has been received. If a fragment from a fragmented packet is lost in the network, the entire packet will be dropped. Comparing a single large packet and several smaller ones (assumed to be smaller than potential fragments of the single larger packet), the single large packet will have larger delays between links due to the entire packet needing to be transferred before being able to be transmitted to the next link in the route path and a larger risk of being dropped due to missing fragments. The smaller packets come with the drawback of requiring a header section for each packet, resulting in a larger header field overhead for the transmission. When using UDP, packet size should be compared to header field overhead for balanced communication.

3.3 Network Address Translation

For two hosts to be able to communicate over the internet, they need to know each other’s IP address. In a client and server scenario, the client typically accesses a Domain Name Server, DNS, to be able to translate a URL given by the user into a destination IP address. This requires the server to be listed on some DNS and for its IP to be public knowledge.

In a peer to peer connection scenario, the two hosts or endpoints trying to communicate are typically not listed on a DNS server. Instead, one host needs to know the address of the other or the hosts need to connect to some third party to exchange address information.

The number of devices connected to the internet has increased at a high rate since the invention of the internet. Previously it was possible for all devices to receive a unique IP in the IPv4 domain. This is no longer possible due to the number of connected devices.

To reduce the amount of required IP addresses, many routers employ Network Address Translation or NAT. Put simply, NAT enables a single IP address to be used by multiple different endpoints behind a router. The router has some external IP address, but internally distributes its own IP addresses among the users of the network. To enable connecting to an
outside network, commonly the public internet, the router keeps track of any internally connected host currently maintaining a networking session with any outside host.

**NAT Traversal**

NAT traversal is the act of establishing or maintaining a connection across a NAT table. NAT can be implemented multiple ways and these different types of NAT each provide different methodologies for how connections are established and maintained. Depending on what type of NAT is used, the available guaranteed methods for traversing the NAT, i.e. connecting to a host on the other side of the NAT, vary. In the most prohibitive case, a third party with an externally known address needs to be used to set up a peer to peer communication session between hosts behind different NATs.

### 3.4 Multitasking and multithreading

A computer system with a single processor can perform multiple tasks seemingly at the same time by using multitasking. Typically, a computer system runs a set of tasks called processes. The processes are by convention independent from one another and occupy separate memory spaces within the computer architecture, being unable to access the memory space of other processes.

On a single processor core system, only one process can be run at any instance in time. To perform multitasking, the operating system employs a scheduling strategy to determine what process should be run when and what priority it should carry in the process time scheduling. Switching between processes is called context switching and enables multiple processes to achieve progress without having to wait for other processes to fully complete execution (which may or may not happen). If a process is waiting for some system resource or user input it can give up its allocated execution time to other waiting processes.

Within a process, multiple threads can be run. Threads have similarities to processes in that they can run concurrently by scheduling. They differ in the sense that threads typically share memory space with each other, enabling quick inter-thread communication. A disadvantage of threads as compared to processes, is the potential introduction of memory consistency problems. Since thread execution can arbitrarily be paused to run another thread, scenarios where shared memory is altered by two threads simultaneously become possible. These scenarios can for example result in so called race conditions, where the output of the program becomes dependent on which thread manages to access the shared memory first.

### 3.5 4G

The International Telecommunications Union or the ITU is an agency under the United Nations concerned with coordinating radio, networks and related technologies. Included in the mission description of the ITU is the ensuring of there being standards for communication using networks and technologies, allowing them to interact. [15]

The fourth generation of broadband cellular networking, abbreviated 4G, is standardized by the ITU. The 4G standard provides end user data rates of 100 Mbit/s when the receiver and base station move with high speed relative to each other and data rates of up to 1 Gbit/s when they are stationary relative to each other.
4 Implementation

4.1 Overview of the platform

The platform chosen for evaluating the criteria formulated in section 2.6 and demonstrating a GNSS is a network-controlled car with surrounding control functionality. The platform consists of four parts as shown in Figure 4.1. A laptop (A) runs the control interface that transmits data to and receives data from the relay server (B) via the public Internet. The relay server (B) relays data between the control interface (A) and the car platform (D). The reference GNSS receiver (C) communicates with the GNSS receiver mounted on the car platform (D) via radio. During testing, the reference GNSS receiver (C) is connected to the control interface (A), this however is not required; the reference GNSS receiver can be powered from an arbitrary power source during normal operation.

![Diagram of platform components](image)

Figure 4.1: An overview of the hardware constituents of the platform.

A: Laptop or other computer running the control interface which controls the car.
B: Relay server which is used to relay data between the desktop software and the computer mounted on the car.
C: The reference GNSS receiver. Communicates with the GNSS receiver mounted on the car.
D: The network-controlled car which has a 4G module and a GNSS receiver mounted. Communicates both with the desktop software via the relay server as well as with the reference GNSS receiver.

The network-controlled car allows the GNSS receiver to be moved for evaluating tracking performance as well as providing a means to demonstrate the GNSS functionality on a controllable moving object.
The network-controlled car should be able to move at variable speeds to evaluate tracking. The car should also be outfitted with a front-facing camera to allow for control when not within sight and it should transmit its position as reported by the GNSS receiver. It should also be usable in an outside environment.

The car is to be controlled from a software control interface also to be developed within the project. The control interface should allow a user to steer the car and should present the user with a video feed from the front-facing camera mounted on the car. It should also show the reported position from the GNSS receiver mounted on the car.

### 4.2 Control interface (A)

The control interface is a human-device interface (HDI) for controlling the NetCar (D). It consists of (1) a Java software program, which can be run on any computer that can run the Java virtual machine; (2) a keyboard for user input; (3) a screen to display the graphical user interface (GUI); and (4) an internet connection to be able to connect to the NetCar (D) via the relay server (B).

### 4.3 Relay server (B)

The server is connected to the internet via a router and is given an internal IP address by the router. The router is configured to forward certain ports to the internal IP address of the server, meaning that all incoming data from the external net on these ports are sent to those same ports on the server. If the router is publicly addressable, the server will then be addressable by any party that is connected to the public internet on the chosen specific ports. The motivation for and the purpose of the relay server is explained below.

The 4G modem on the NetCar (D) is connected to the public internet via a carrier grade NAT. The NAT enables the initiation of outgoing connections, but will not allow incoming connection sessions to be established. The NAT is run by the service provider and cannot be configured by the user.

It is desirable for the control interface (A) to be runnable regardless of what type of network the computer running the control interface (A) is connected to. If the control interface (A) computer is connected to a router, it will most likely also be connected to the public internet via NAT. Depending on the type of NAT, an incoming connection session could potentially be established, but assuming the worst-case scenario implies that only outgoing connection sessions can be established.

The resulting problem is that neither the NetCar (D) nor the computer running the control interface (A) can accept an incoming connection from the other part. The solution to this problem is NAT traversal. One way of implementing NAT traversal is to set up a relay server which has a publicly accessible IP address that is known to both parties trying to communicate. After establishing outgoing connection sessions to the relay server, both parties behind NAT protocols can transmit data to the relay server, which then simply relays the data to the other party. This approach can traverse all types of NAT protocols.
4.4 GNSS receivers

The GNSS to be evaluated is the uBlox NEO-M8P differential GNSS, which was chosen by Syntronic. The NEO-M8P is to be evaluated using the C94 M8P evaluation kit, which comes with two C94 M8P modules, one to be run in rover configuration and the other to be run in reference configuration. The two configurations are physically identical and differ only in software options. The reference module (C) is stationary and is placed in the operating region of the NetCar and the rover module is mounted on the NetCar (D).

The GNSS receivers consist of a uBlox NEO-M8P GNSS receiver chip mounted on the C94 M8P evaluation kit. A close-up of the evaluation kit is shown in Figure 4.2 and a close-up of the NEO-M8P is shown in Figure 4.3. The C94 M8P comes equipped with an array of different interfaces for communicating with the NEO-M8P as well as multiple ways to provide power to the chip.

The C94 M8P uses a patch antenna with an attached ground plane for communication with the GNSS satellite constellations. Throughout the project the receivers are configured to receive satellite data from the GPS and GLONASS satellite constellations. The reference GNSS receiver communicates with the GNSS receiver mounted on the NetCar (D) using UHF radio. The UHF radio uses a second antenna.

The uBlox NEO-M8P is a 72-channel receiver, meaning it can simultaneously use up to 72 satellites from up to 2 different constellations for positioning. When used in a stationary reference-moving rover pair it has a position update rate of 5 Hz and a convergence time of less than 60 seconds. When used standalone, the specification lists the horizontal accuracy as 50% of position estimates being within $2.5 \text{ m}$ of the actual position. When used in a reference-rover pair, the horizontal accuracy is given as 50% of position estimates being within $0.025 \text{ m} + 1 \text{ ppm}$ for reference to rover distances up to $10 \text{ km}$. [16]
4.5 NetCar (D)

Consists of a remote-controlled car chassis which comes with a brushless DC motor, an ESC for the brushless motor and a steering servo pre-installed. On top of the chassis, additional components are mounted. An overview of the NetCar’s subsystems is shown in Figure 4.4. In addition to the previously mentioned pre-installed components, the NetCar also has a uBlox NEO-M8P GNSS positioning system, a webcam used for the video feed, a 4G modem used for communication and a Raspberry Pi used to coordinate the other subsystems.

![Figure 4.4: Overview of the NetCar subsystems. The 4G module is used to communicate with the desktop via the relay server. The NEO-M8P](image)

To be able to communicate with the desktop application, the NetCar is equipped with a 4G module. The 4G module provides internet data rates high enough to enable streaming of video.

The NetCar chassis is a Turnigy Trooper car, which comes pre-equipped with a brushless motor, an electronic speed controller for driving the motor and a steering servo. The Turnigy Trooper comes with a drivetrain that powers all 4 wheels with a single motor as well as differentials for both rear and front wheel pairs. The steering servo is connected to the front wheel pair, allowing the NetCar to turn.

As outlined in the previous section, the GNSS system to be evaluated is the uBlox NEO-M8P GNSS receiver module. One of the two C94 M8P modules included in the evaluation kit was mounted on the NetCar.

4.5.1 Raspberry Pi

The main processing unit to be mounted on the NetCar should have enough computing power to be able to encode and stream video while simultaneously performing other less complex, but time-critical, tasks. It should also be able to interface with low-level electronics interfaces. The processing unit chosen for this task was the Raspberry Pi 3 model B.

The Raspberry Pi is a computer fitted on a single circuit board. It comes in a variety of models, varying in size and processing power. For the NetCar a Raspberry Pi 3 model B is used, which at
the time of this writing is the newest and most powerful of the Raspberry Pi models. The Raspberry Pi 3 model B features an ARM Cortex-A53 64-bit 1.2 GHz quad-core CPU and 1 GB of RAM providing ample processing power. It also features several USB ports and GPIO pins for interfacing with other devices. [17]

The Raspberry Pi was chosen as the control circuit for the NetCar as it can interact with low-level electronics via GPIO ports as well as performing more computationally complex operations, such as encoding and sending video.

4.5.2 Connectivity
The chosen mode of communication between the NetCar and the control interface (A) is through the public internet. A 4G modem is mounted on the NetCar for public internet communications.

In addition to the 4G mode of communication, the WiFi chip on the Raspberry Pi is configured to act as a wireless access point to enable direct peer-to-peer communication when proximity between the NetCar and the computer running the control interface (A) allows for this.

4.5.3 Webcam
The webcam chosen for streaming video from the front of the NetCar is the Logitech C525 HD webcam. The C525 is able to record in 720p resolution and has a frame capture rate of 30 frames per second. It is compatible with the Universal Video Capture (UVC) standard for webcams.

4.5.4 Power supply
The NetCar chassis is powered by a 4000 mAh LIPO battery. As this power source is already mounted on the vehicle voltage regulation, rather than a separate power supply, was used to power the Raspberry Pi and other electronics mounted on the NetCar.

Lithium polymer (LIPO) batteries have a high specific energy, i.e. energy per mass, compared to other battery technologies. This means LIPO batteries are typically employed where the weight of the battery is of concern to the system being powered, for example when powering a moving system which also transports the battery. LIPO batteries are commonly used by hobbyists to power remote controlled vehicles but are also employed in modern electric cars.

LIPO batteries do not use a liquid electrolyte as other comparable battery technologies do, instead using a solid polymer as the electrolyte connecting the anode and cathode of the battery. The liquid electrolytes in other batteries typically allows current to flow through the battery. This is not the case with the solid polymer electrolyte, which instead allows ions to flow between the anode and cathode. [18]

The LIPO battery used contains two LIPO cells in series, meaning it can safely hold a total maximum charge at 8.4 V (meaning the cells are individually charged to 4.2 V safely). During operation the voltage can be safely discharged to a total level of 6 V (meaning each cell can be safely discharged to a level of 3 V). This means the voltage regulation system needs to be able to accept a varying voltage level between 8.4 V and 6 V and output a steady 5 V required to power the Raspberry Pi and the other electronics.

The voltage regulation system that supplies the Raspberry Pi and all peripherals connected to it is made up of two BA50DD0T linear regulators that can handle a maximum of 25 V as input and a maximum dropout voltage of 0.5 V, meaning they should output 5 V in the voltage input
range 5.5 – 25 V. They are capable of outputting 2 A each, which is an ample supply of current for the Raspberry Pi including the current drawn by the peripherals. The reason for using 2 regulators is the power dissipation: without heat-sinks the regulators are able to dissipate 2 W at room temperature. The current draw of the Raspberry Pi with peripherals is maximally in the order of 1 A, meaning regulating the voltage of a fully charged battery at 8.4 V down to 5 V requires 3.4 W\(^1\) to be dissipated, more than a single regulator can handle.

4.5.5 Direct Current motors

Direct current (DC) motors convert a direct current into mechanical rotation. This conversion is accomplished using the interaction between magnetic fields where at least one of two fields is generated by the direct current. The resulting Lorentz force interaction is what drives the rotation.

A DC motor consists of two main parts, a field source and an armature. The armature is the rotating part of the motor and it sits within the field generated by the field source. For full rotation to be possible, the direction of one of the two magnetic fields needs to change. In a DC motor this is accomplished by reversing the current direction at the correct times. The process of reversing the current is called commutation.

Types of DC motors

The source of the magnetic fields used vary between different types of DC motors. DC motors can be divided into 4 different types based on differences in how the magnetic fields involved are generated. There are permanent magnet motors, series wound motors, shunt wound motors and compound motors.

Permanent magnet motors utilize a permanent magnet as the source of magnetic flux for one of the two interacting magnetic fields. Using a permanent magnet in place of a second electromagnet means the motor can spend less energy magnetizing and demagnetizing a second set of coils. Due to this, permanent magnet motors are generally more efficient compared to the electromagnet type of DC motor. The main drawback of the permanent magnet motor is cost; strong permanent magnets are made up of rare earth minerals that come at a higher cost than additional copper windings for a second set of coils. [19]

Series wound motors have the field source and armature linked in series. This results in a, compared to the other variants, high initial torque. The rotational speed of a series-wound motor varies with the load. A high load results in a slower rotational speed and conversely a light load results in a faster rotational speed. [20]

Shunt wound motors have the field source and armature linked in parallel. Because of this the current going through the field source and armature are separate. Due to this, a shunt wound motor is more or less independent of the load it’s driving. The torque of the motor and the speed at which it rotates have an inverse proportional relationship.

Compound motors have field sources linked both in series and in parallel with the armature. This results in a motor with combined characteristics from both approaches. A compound motor can output a higher starting torque than a shunt wound motor and is able to better regulate speed than the series-wound motor.

\[ P = U \times I \] where \( P \) is the power dissipated, \( U \) is the voltage difference and \( I \) is the current
Commutation

Direct current inherently only flows in a single direction. For a DC motor to rotate fully, it is necessary to alter the magnetization of either the field source or the armature. To change the direction of magnetization in a DC motor, commutation is used, which changes the direction the current flows in the motor. There are two main ways of achieving commutation and these result in a further classification of DC motors, namely brushed and brushless DC motors.

**Brushed motors**

A brushed DC motor has brushes typically made of carbon that conduct electricity to the armature via a commutator. The commutator consists of disconnected sections which are connected to the armature coil windings such that the direction of the current through the armature coil windings alternate as the commutator rotates and the brushes are connected to different parts of the commutator. An overview of a brushed DC motor circuit is shown in Figure 4.5. As the commutator (A.) rotates with the armature, the brushes (B.) are able to alternate the direction of current to the armature coil windings.

![Figure 4.5: Overview of a brushed DC motor circuit. A. is the commutator and B. are the brushes conducting current into the armature windings via the commutator.](image)

A drawback of using a brushed motor is that the brushes wear out over time due to friction. When designing a brushed DC motor, care must also be taken to minimize sparking between the commutator and the brushes, which further increases deterioration of the brushes during use.

**Brushless motors**

Brushless motors achieve commutation by electrical rather than mechanical means. Instead of having a mechanical commutator mechanism, brushless DC motors have multiple sets of field source coils that can be magnetized separately. By using a sensor to keep track of the armature rotation, the correct set of coils can be energized at the correct time to achieve continuous
rotational motion. A sketch of the principal workings is shown in Figure 4.6. The coil-pairs C1 through C3 are energized in sequence, depending on the sensor reading of the current armature rotation. The armature is visualized as a magnet in the middle of Figure 4.6.

![Brushless DC Motor Diagram]

Figure 4.6: Overview of a brushless DC motor. The coil-pairs C1 through C3 are energized in sequence to achieve continuous rotational motion of the armature visualized as a magnet in the center of the figure.

A brushless DC motor requires additional electronics for controlling what coil-pairs should be magnetized at what time. These additional electronics are typically packaged into a single component called an Electronic Speed Controller (ESC) and are often separate from the motor itself.

**Motor selection**

The NetCar is powered by battery, meaning an efficient motor is desirable. A permanent magnet motor is more efficient than a wound motor and are the motors of choice for remote-controlled applications. As to the commutation, a brushless motor was decided upon due to comparatively higher torque and the absence of brushes that get worn down over time.

**4.5.6 Car chassis**

The chassis chosen for the NetCar is the Turnigy Trooper. The Turnigy Trooper comes with a brushless motor pre-installed as the primary drive motor, an ESC for controlling the brushless motor and a drive train that powers all 4 wheels using the primary motor, i.e. it has 4-wheel drive using a single motor. It also comes with a servo for controlling the steering using front wheel pair.
4.6 Development tools
For developing the Java software, the integrated development environment (IDE) Eclipse was used. Eclipse is an open source, fully fledged development environment which has an extensive plugin support and ecosystem allowing customization.

JavaFX was used in developing the desktop application. “JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.” [21]

Scene Builder was used to construct the graphical user interface (GUI). Scene Builder provides a drag and drop interface for building GUIs and produces FXML code, an XML-based markup language, which a JavaFX application is able to load. With the use of variable injection, handles to the various elements of the GUI defined in the FXML code become accessible in the Java code. This enables development of event driven functionality based on user interaction with the GUI.

The Raspberry Pi was controlled using the Secure Shell (SSH) command line client PuTTy during the development process. PuTTy is a free and open-source client for remote terminal access and was written and is maintained primarily by Simon Tatham. [22]

The file transfer protocol (FTP) was used to transfer files to the Raspberry Pi. The free FTP application FileZilla was used for this purpose.

For configuring and performing some of the testing of the specific GNSS in the project, the uBlox NEO-M8P, uBlox’s software u-center was used. U-center allows the user to inspect all messages sent by the GNSS chip and offers a host of configuration options to customize system operation for suitability in performing varying tasks.

4.7 Control interface (A) software
The control interface software can be subdivided into two separate parts, the GUI that is presented to the user and the backend which listens for and interprets user input and communicates user actions to the NetCar (D).

4.7.1 Graphical user interface
The graphical user interface (GUI) for the control interface consists of a main screen window which contains a video view for viewing the video stream sent from the Raspberry Pi. Also in the main window are some heads-up display indicators that indicate when the user presses a control button and a minimap which draws the current location of the NetCar, using position data reported from the GNSS receiver mounted on the NetCar. Figure 4.7 shows a sketched overview of the GUI for the desktop application.
4.7.2 Back-end communications

The back-end of the control interface is what enables the control interface to receive the video stream and the GNSS position data from the NetCar (D). This back-end also takes care of sending user commands to the NetCar.

The back-end consists of several modules: A command sender, a video receiver, a position receiver and a minimap fetcher. The command sender, video receiver and position receiver modules each establish a connection to the Raspberry Pi and they will attempt to reestablish this connection when it is lost. Due to the overlapping nature of the behavior of the modules, polymorphism was used in the implementation. The superclass takes care of the connection-reconnection logic and the subclasses handle their specific data types that need to be transmitted or received. The minimap fetcher fetches a map from the internet using an HTTP request.

**Command sender module**

User commands are encoded into a binary command string which includes the logical complement of the commands. The command string structure is shown in Figure 4.8, where command (Cmd) 1 is in the active state (1 0) and the other commands are in the inactive state (0 1). When a user presses or releases a key, the event is detected and the corresponding command is switched to the active or inactive state depending on the type of event detected.
The command string is encoded as an integer and transmitted to the NetCar via TCP only when either an additional key is pressed or a key previously pressed is released. The command string enables multiple key presses to be registered and actuated by the NetCar, which for example enables steering at the same as moving forward. Sending a single command upon detecting a key press or release does not as easily handle multiple events and the potential reduction in network traffic from using the single command approach as compared to the command string approach is negligible as the command string sent is only 4 bytes in size.

The available commands with their mapped keys within parenthesis are:

- Forward (Arrow up)
- Backward (Arrow down)
- Steer left (Arrow left)
- Steer right (Arrow right)
- Brake (Spacebar)
- Nitro (Left Shift)
- Gear up (W)
- Gear down (S)

These 8 commands result in the command string being 16 bits long, or 2 bytes. In Java, the size of an integer is 4 bytes and while the ‘short’ datatype, which is 2 bytes, could have been used, it was decided that leaving space for additional commands would be beneficial.

The command sender is run on its own thread and can asynchronously from the rest of the program place commands on an outbound link whenever they are detected.

**Video receiver module**

The video receiver module receives, as the name implies, the video stream transmitted via TCP from the webcam mounted on the front of the NetCar. The video frames are received, encapsulated into the JavaFX image format sent to the JavaFX engine for display on the GUI. The video receiver module is run in its own thread and can asynchronously receive video frames.

**Position receiver module**

The position receiver module receives position data from the GNSS receiver mounted on the NetCar. The position data is received as NMEA messages, which is a standardized message form for communicating satellite positioning data. Aside from the latitude and longitude, the messages...
also include whether the position is considered valid based on the receiver satellite lock, detailed data on the satellites currently acquired and current ground speed among others.

The NMEA messages are parsed to retrieve the latitude and longitude of the receiver which are then projected onto a 2-dimensional map using Vincenty’s formula to be displayed on the GUI.

4.8 NetCar onboard computer software

The Raspberry Pi mounted on the NetCar (D) runs software which handles communication with the control interface. The software fetches video frames from the webcam and streams the video frames to the control interface, it sends position data from the GNSS receiver to the control interface and it receives commands from the control interface and actuates commands to move the car by using Pulse Width Modulated (PWM) signals to communicate with the ESC for the brushless motor and steering servo. To perform these tasks simultaneously, multithreading is utilized.

4.8.1 Video streaming

The Raspberry Pi streams video captured by a webcam to the desktop application. This task can be broken down into three separate steps. First the video frames, the actual image data, need to be retrieved from the webcam. Secondly, an optional step, the video frames can be encoded to reduce their size. Finally, the frames are sent to the desktop using the network protocol of choice for the application. These three steps are discussed in greater detail in the following paragraphs.

The first step, capturing the video frames, is done by interfacing the software written for the Raspberry Pi, referred to as the Raspberry Pi application, to a driver for the used webcam. The webcam used supports the USB Video Class (UVC) driver, a standardized driver format for video capturing devices, and is the driver used in the project.

The second step, encoding the video frames, is not essential for achieving basic functionality. Encoding is a useful tool when attempting to reduce the end-to-end delay of the stream and increase the video frame image quality.

Encoding can work on single frames, reducing their size in either a lossless or a lossy manner. Lossless encoding implies that no information is lost during the encoding process, reducing the frame size while still being able to fully reconstruct the original frame. Lossy encoding typically utilizes a quality index to tune how much information is lost in the encoding process. During lossy encoding, the original frame cannot be fully reconstructed. In return for this tradeoff, lossy encoding is expected to generate smaller frame sizes compared to lossless encoding.

Some types of encoding can utilize the fact that the stream is a video rather than a still image. One way of utilizing this information is to only send a full frame at certain intervals. In between intervals, so-called delta frames can be sent. Delta frames contain only information about what changed in the frame since the last full or delta frame and are smaller than a full frame. The delta frame size will depend on how much of the frame changed since the last capture.

The final step of sending the frames is carried out using a chosen network protocol. Some network protocols are designed with real-time transmissions in mind, while others are more general in their implementations.

Various solutions for streaming the video were tested, including FFMPEG, a WebRTC-based solution called UV4L and a direct TCP stream implementation in Java. Testing showed the direct
TCP stream implementation in Java gave the best performance for live streaming with an emphasis on low end-to-end latency as well as being most easily manageable. This seems counterintuitive and therefore the following caveat is appended: both FFmpeg and UV4L contain a wide range of functionality and it is entirely possible that some options were overlooked during the testing.

The Raspberry Pi application has several tasks that include interacting with the desktop application over the internet. Video streaming, compared to the other tasks, consumes the majority of networking traffic carried out between the two platforms. Video streaming is also time-critical in the sense that for an operator to drive the NetCar fluidly, the end-to-end delay of the video stream must not be too large. That is, when the user accelerates the car, the visual feedback via video stream needs to be responsive enough for the user to be able to react to changes in the NetCar environment as the car drives around.

To minimize end-to-end delay, thereby maximizing responsiveness, the video streaming process was split into the parts outlined above and timed separately. Using this data, the software was constructed to optimize the process.

A sequential implementation of the entire process would look as in Figure 4.9. This should be compared to a multithreaded approach detailed in Figure 4.10. Assuming that multiple cores are available, the task of reading the webcam frames can be accomplished simultaneously as encoding and sending. While the end-to-end delay for the first frame will be the same in both cases, the multithreaded implementation will for the following frames reduce the end-to-end delay, since the encoding and sending thread does not have to wait for the entire frame to be read from the webcam.

![Sequential Timeline](image)

Figure 4.9: Timeline of a sequential approach to retrieving frames from a webcam, encoding the frames and then sending them.

![Multithreaded Timeline](image)

Figure 4.10: Timeline of a multi-threaded approach to retrieving frames from a webcam, encoding the frames and then sending them.

The interthread communication between the webcam reader thread and the encoding and sending thread is based on a memory structure with two buffer memory locations. The encoding and sending thread accesses the latest fully available frame and begins encoding. The webcam
reading thread replaces the older of the two stored frames with the latest frame read from the 
webcam. The memory spaces are synchronized, meaning that if a thread starts reading from or 
writing to one of the memory spaces, the other thread is unable to access that memory space 
until the first thread has completed its operation. This ensures that the threads do not corrupt 
each other’s data.

5 Results and discussions

5.1 NetCar
The NetCar along with its control interface constructed in the project is able to drive around, 
while streaming a live feed video to the user at the control interface. The GUI displays the live 
feed video, the controls input by the user and a map where the position of the NetCar is drawn.

The control interface is able to accept user control input which is sent to the NetCar and then 
actuated. The NetCar is able to receive instructions for moving forward and backward, for 
steering left and right and instructions for switching a speed setting referred to as a gear. These 
instructions are then translated into corresponding hardware level instructions in the form of 
PWM signals using the standard servo protocol to actuate the instructions.

The NetCar also receives input from the mounted GNSS in the form of NMEA-messages. These 
NMEA-messages are sent to the control interface and contain information on the current GNSS 
estimation of where the NetCar is located.

In one test the NetCar was driven around at variable speeds for about 400 seconds. All systems 
except the 4g-modem were active during the test and WiFi was used for internet connectivity. 
During these 400 seconds the battery voltage level dropped from 8.16 V per cell to 7.94 V per 
cell. The discharge curve for a LIPO battery in this interval is approximately linear.

The operating range of the NetCar when using the WiFi access point as the mode of 
communication was tested to be about 50 m when using a laptop with its onboard WiFi receiver. 
At this range the packet loss is considerable and the video stream at this range has high delay. 
While it is possible to control the NetCar at this range, the user must wait to receive visual 
feedback after sending move commands.

5.2 Relay server
The relay server was tested using the internal network provided by a router and did not result in a 
oticeable increase in delay when compared to directly connecting the control interface to the 
NetCar via the same internal network.

Unfortunately, the router configuration was handled by a 3rd party and while the relevant ports 
were supposed to be forwarded to enable NAT traversal from the public internet, something 
prevented the relay server from working. Seeing as the relay server worked on the internal 
network, some configuration in the router can be suspected to be the reason for being unable to 
establish connection to the relay server from an external connection. Due to this the relay server 
was not able to be tested as a means of NAT traversal within the time frame of the project.

While the testing of the relay server did not introduce noticeable delays when connected via the 
internal network of the router, this testing only introduces one additional routing node in the
end-to-end connection path. Connecting via the relay server from external connections introduces several additional nodes. With these additional nodes the end-to-end delay will increase. Whether this increase in delay would lead to the NetCar being too unresponsive to drive in any practical manner remains unanswered.

5.3 uBlox NEO-M8P performance
The uBlox NEO-M8P was tested using the NetCar both standalone and in DGNSS-mode, using a second NEO-M8P as a reference receiver. The testing was conducted with the receiver having a close to full 360° unobstructed view of the sky and being configured to receive the GPS and GLONASS constellations. It is important to note that uBlox themselves maintain that the patch antennas delivered with the C94 M8P evaluation kit have a difficult time maintaining an RTK fixed lock and recommend providing the modules with more sensitive antennas. The testing was in spite of this performed using the aforementioned patch antennas.

5.3.1 Fix time
The time to first fix (TTFF) was measured by sending a cold-start command via the u-center interface to the NEO-M8P. This drops all current knowledge of position and time, resetting the receiver. Over a set of 10 tests, the average time to resynchronize the receiver clock to the satellites was found to be 7.4 s and the average TTFF was found to be 24.2 s, with a worst case TTFF of 33 s during the 10 tests. The system seems adept at maintaining a lock while moving.

5.3.2 Stationary accuracy
When the NetCar is stationary, the accuracy of the GNSS can be evaluated using the deviation map provided in the u-center utility program made by uBlox. Figure 5.1 shows the positions reported while the system was in RTK float mode, sometimes dipping into RTK fixed mode. The cluster of points at the center of the map were all generated during the short periods of RTK fixed mode. Of note is the scale of the deviation map, the innermost circle has a radius of 10 cm. It would seem RTK fixed mode is able to maintain an accuracy as promised by uBlox. When in RTK float mode the position drifts in the order of tens of centimeters.
Figure 5.1: Deviation map during mixed RTK float and fixed lock. The small cluster of points at the center of the map were generated during fixed lock, the others during float.

Removing one of the GNSS receivers, thereby disabling the RTK mode, the test was done again. The deviation map for this second test is shown in Figure 5.2. It should be pointed out that the scale in Figure 5.2 is 10 times the scale in Figure 5.1, and its innermost and outermost circles have radii of 1 and 4 meters, respectively. The position drifts much more than in the case where RTK is enabled.

Figure 5.2: Deviation map without a second receiver, thereby disabling RTK mode. Note that the outermost circle has a radius of 4 m, 10 times larger than the outermost radius in Figure 5.1.
5.3.3 Tracking accuracy
Aspects of the tracking accuracy were evaluated in different tests. The tracking accuracy was tested with the positioning system set to produce readings at 5 \text{Hz}.

Qualitative evaluation of accuracy
Qualitatively the tracking accuracy was evaluated while driving the car using the position drawing of the control interface. The tracking is fast enough to track turning of the car and is responsive to fast turns. As an example, the positions reported when driving about first in a circle followed by a random pattern is visualized in 5.3, where the black box is drawn for scale with sides of length 1\text{m}. The starting circle was driven by using maximum steering to the right. This implies that the positioning system can accurately track the car when turning the tightest possible corners for the car.

![Figure 5.3: Positions reported when driving first in a circle followed by a random pattern. The black box is present for scale and has sides that are each of length 1\text{m}.](image)

Statistical evaluation of accuracy
To statistically evaluate the accuracy in the positional fixes of the NetCar, the following test was performed. The NetCar was driven in an approximately straight line at constant speed. The distance between a reported position and the previously reported position was calculated for the reported positions along the straight line and averaged. The standard deviation of these distances was also calculated. The distance was calculated between 27 latitude and longitude pairs using Vincenty’s formula. The average calculated distance was 0.329\text{m} with a standard deviation of 0.017\text{m}. The maximum distance was 0.358\text{m} and the minimum distance was 0.298\text{m}. The standard deviation in distance of 0.017\text{m} suggests centimeter level relative accuracy.

Absolute accuracy evaluation after moving
A test to evaluate whether the positional fix of the NetCar drifted over time was performed by driving the NetCar in circles and measuring the distance between a fixed point on the NetCar at...
the starting location and the finishing location. The distance measured between these points was compared to the distance calculated between the reported coordinates at the two points. The results from these tests are presented in Table 5.1.

Table 5.1: Results from driving the NetCar in a circle and measuring the distance between the starting position and the finishing position. The distance between the reported starting and finishing positions is calculated using Vincenty’s formula and compared to the measured distance between the points.

<table>
<thead>
<tr>
<th>Calculated distance (cm)</th>
<th>Measured distance (cm)</th>
<th>Error (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>35.5</td>
<td>17.0</td>
<td>18.5</td>
</tr>
<tr>
<td>28.1</td>
<td>7.0</td>
<td>21.1</td>
</tr>
<tr>
<td>34.2</td>
<td>16.5</td>
<td>17.7</td>
</tr>
<tr>
<td>16.7</td>
<td>22.5</td>
<td>5.8</td>
</tr>
</tbody>
</table>

The measurements reported by the GNSS can be described as $P$ given by

$$P = P_{\text{actual}} + \delta$$

where $P_{\text{actual}}$ is the real position and $\delta$ is the error in the estimate given by the GNSS. In Table 5.1 the calculated distances are based on double position measurements, this means errors in position are doubly represented in the errors between the calculated and measured distances as

$$P_A - P_B = P_{A,\text{actual}} - P_{B,\text{actual}} + \delta_A - \delta_B$$

The double error is limited in magnitude by the double maximum error in the following manner

$$|\delta_A - \delta_B| \leq 2 \times \delta_{\max}$$

Assuming the error in position is Gaussian, the errors in Table 5.1 are likely to be larger than the errors in the position estimate. A possible explanation for the higher error magnitude compared to the statistical evaluation could be that the positional fix for the reference receiver, used in a self-survey mode, has more time to drift while driving a full circle with the NetCar as compared to the time between two neighboring measurements of position in the case of the statistical evaluation.

5.4 Further development

This section explores some potential further developments that could be added to the NetCar to extend its capabilities.

5.4.1 Automatic navigation

The groundwork to implement basic automatic navigation for the NetCar is present within the project. A simple implementation would simply accept coordinates from a user and try to steer the vehicle toward these coordinates.

Automatic navigation can be made more complex by including obstacle avoidance. Obstacle avoidance would either require image processing from the streamed video, additional sensors or a combination of both. Obstacle avoidance would increase the NetCar’s ability to navigate to a certain set of coordinates due.

Automatic navigation could be used to increase the GNSS evaluation aspect of the NetCar by having a pre-surveyed track which the NetCar would be tasked to follow using the GNSS.
magnitude of deviations from the track would give an indication as to the performance of the GNSS.

5.4.2 Inertial measurements and sensor fusion
Adding an inertial measurement unit (IMU) could be used for a second estimation of the position of the NetCar. This estimation would be relative in its nature, but potentially with a faster update rate compared to the GNSS positioning. The relative position data from the IMU could then be fused with the position data obtained from the GNSS positioning, achieving a higher degree of accuracy in position.

As an alternative to fusing the second estimation of position, the two estimates could be compared and used to evaluate GNSS performance. This requires high accuracy in the IMU position estimation and since the IMU position estimate is relative and unanchored by itself, drifts in the position estimate from the IMU need to be considered and compensated for.

The relative position estimation could be made by implementing basic kinematics formulae by integrating accelerometer data from the IMU over time. A higher accuracy positional fix would potentially also improve automatic navigation performance (see section Automatic navigation).

5.4.3 Pathfinding
Pathfinding using satellite imagery could be implemented alongside automatic navigation, to optimize paths from identifiable static obstacles. Combined with obstacle avoidance, this could lead to successful automatic navigation.

5.4.4 Battery monitoring
An analog to digital converter could be integrated in the system to monitor the lithium polymer (LIPO) battery voltage level. LIPO batteries should not be discharged lower than 3.6 V per cell as doing so can damage the battery. To avoid accidental over-discharge during operation, the battery monitor circuitry could cut power to minimum when dangerously low levels are approached.

5.4.5 Area mapping
By mounting a LIDAR on the system, the system could be driven around and map LIDAR ranging data together with the location data reported from the GNSS. This would enable visualizing areas in 3D or 2D depending on how the implementation of the LIDAR is made.
6 Conclusions

The NetCar is able to demonstrate the functionality of a GNSS and is able to evaluate its performance to a certain extent, but would require additional features to be able to evaluate the centimeter-level accuracy. Especially difficult is the evaluation of delay in tracking accuracy.

There is a large difference, meter-level accuracy versus centimeter-level accuracy, in GNSS performance when comparing a non-assisted GNSS to a differential GNSS augmented using RTK. RTK as an augmentation method has the drawback of requiring a reference receiver on the ground.

The uBlox NEO-M8P seems to be able to deliver on its promise of centimeter level accuracy, although there are some ambiguities in the results which prevent a full validation of the performance. For practical use, depending on the demand for precision, improved antennas for the uBlox NEO-M8P should be considered. A better antenna would mean a more consistent RTK fixed mode lock, which should be able to deliver the promised GNSS accuracy more predictably.
References


