

The raw measurement of the pseudoranges and position of each satellite must be known to estimate the pseudorange corrections by the general DGPS model. Acquisition of these data and following application of the PRC becomes impossible with the default setting. Due to this reasons, a special algorithm was developed to enable DGPS functionality with this type of receivers. (Kim, 2013) This algorithm is described in Section (4.3).

3.2.4 DGPS errors

The DGPS can mitigate or reduce mutually correlated errors. It means that errors are projected to the number of receivers, which are close to each other, in the same manner. It involves clock ephemerides and atmospheric effects. As seen from the multipath description in Section 2.6.5, the multipath effect cannot be mitigated even because of the random geometry. For example, the static RS located at the known position and moving rover station. The RS can be located in the open area with a minimal multipath error, but the rover is moving, and it can enter the area with frequent multipath error (near the building). The RS has no chance to predict or reduce this kind of mistakes.

Satellite clock biases and clock errors generated by SA are eliminated. The effect of these errors is projected into the pseudorange measurement, which is same for different receivers at different locations and they can be mutually subtracted.

Ephemeris and atmospheric errors can be reduced but not mitigated. It is because, at a different location, the different error will be measured. The variance of the error depends on the distance between receivers. With increased distance, the variance will also be higher. The amount of error suppression also depends on the age of PRC. The minimum age⁴ was required especially for SA since the errors caused by SA was varying very frequently.

When talking about the distance between receivers, it means the distance between reference and rover station. The positioning error should not exceed 10m in the distant area 100km from the RS. (Madron, 2009)

The table below shows the mean amount of error reduction using DGPS presented by (Forssell, 2008).

Table 6. Error difference between GPS and DGPS (Forssell, 2008)

	GPS C/A code range (m RMS)	DGPS C/A code range (m RMS)
Satellite clock errors	1-3	0
Ephemeris errors	2.5-7	0-0.1
Ionospheric errors	2-15	0.1-1.5
Tropospheric errors	0.4-2	0.1-1.5
Multipath propagation	2-4	2-5
Resulting range error	4-18	2-6
Resulting position error H	6-27	3-9
Resulting position error V	10-45	5-15

⁴ Age – mean the time between the issued PRC and time, when the PRC was applied at the rover station

4 DGPS system proposal

This section will describe the proposal of the new low-cost DGPS system. In the former part, the principal scheme, and functional proposal, which describing the core algorithms used by the both modes. This is followed by necessary GPS receiver setup respectively for RS and rover station. The latter part of the section is focused on correction data transmission and proposal of self-accuracy monitoring.

In general, the system of Arduino DGPS consists of two devices. At the RS's side, the main module considered for all the computations was Arduino MEGA. However, there were lacks of computational precision because MEGA module is equipped with 16MHz ATmega2560, which has limited memory and performance. Due to that, the MEGA has restricted the size of the double variable to 4-bytes (same as float). This lack of precision makes the calculation of PRC and satellite position very problematic. To mitigate this major issue, the Arduino DUE board is used. The Arduino Due support full sized double variables (8-bytes), and thanks to the enhanced microcontroller, the performance is improved.

The rover uses one Arduino device (version Nano or Micro), which receive the data coming from the GPS receiver and establish the Wi-Fi connection with the RS. The rover device only parses the GPS data, send them to the RS, and apply corrected position received from RS since these "small" versions of Arduino are equipped only with ATmega32. This microcontroller has only 32KB of flash memory and limited performance 16MHz. The reason for the use of this device is weight since the application of the Arduino DGPS may be utilized by drones where every gram counts. The key scheme is shown in Figure 4.1.

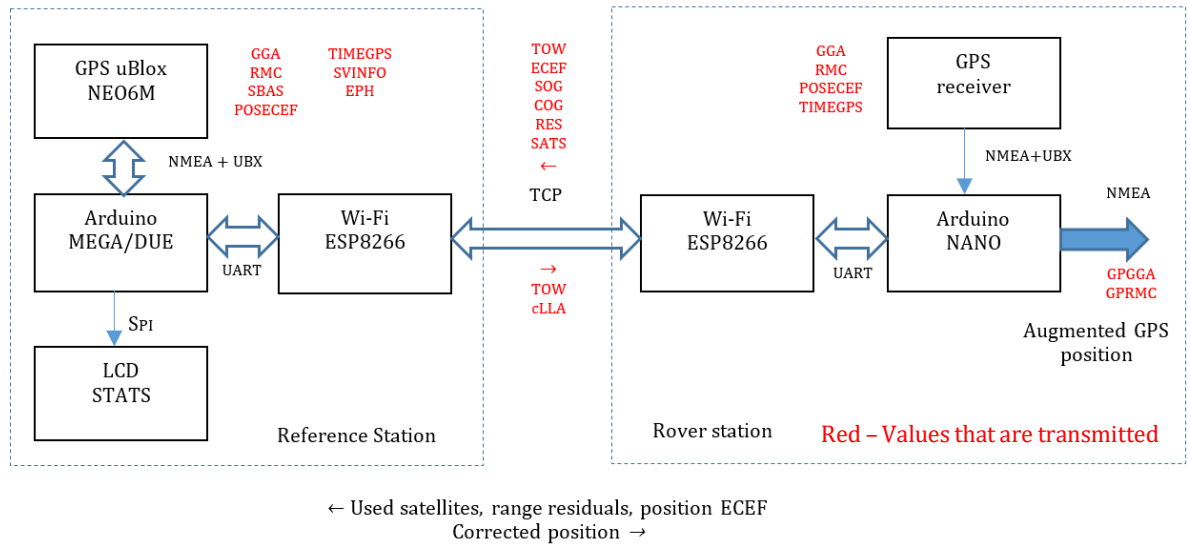


Figure 4.1. Arduino DGPS System proposal - Block Scheme

The communication between rover and RS is dedicated. Therefore, the RS has to compute corrected position for each rover station individually since the position correction processing is done in RS. In the following Arduino DGPS development, only one rover station is assumed.

4.1 Functional proposal

The system of Arduino DGPS is designed to operate in two modes. The first mode supports the Code Range based DGPS idea described in Section 3.2.2.1.

4.1.1 Range Residuals (MODE1)

The main goal in the MODE1 is to generate the PRCs and apply them to the position computed by the rover's GPS receiver. The PRC is calculated as the difference between observed pseudorange and range computed to the true position estimated from the map or by long-term measurement.

4.1.1.1 PRC estimation

The method of deriving PRC is based on the geometrical ranges from the actual measured position ρ_{RS}^j , true position $\rho_{RS,true}^j$ to the particular satellites, which are used at the rover's side, and range residuals ΔR_{RS}^j between observed pseudorange and computed pseudorange from the actual position. The computed pseudorange is modeled as a geometrical range ρ_{RS}^j , satellite clock bias $\Delta\delta_s$, and range bias $\Delta\rho_r^j$, which includes the ephemeris, relativistic and atmospheric errors (4.1). Since the satellite clock bias and range bias are correlated with the rover, they will be subtracted in PRC computation. This is shown in (4.2).

$$\begin{aligned} R_{RS,comp}^j &= \rho_{RS}^j - c \Delta\delta_s + \Delta\rho_r^j \\ R_{RS,observed}^j &= R_{RS,comp}^j + \Delta R_{RS}^j \end{aligned} \quad (4.1)$$

Residuals are contained in the NMEA GRS message and UBX-SVINFO.

Obtained observed pseudorange $R_{RS,observed}^j$ differs from the RAW observable since it does not contain the receiver clock bias. It is the value, which is directly used for position estimation.

The required PRC is computed as same as in the generic range domain DGPS as shown in following equation.

$$\begin{aligned} PRC^j &= \rho_{RS,true}^j - R_{RS,observed}^j \\ R_{r,corrected}^j &= R_{r,observed}^j + PRC^j \\ R_{r,corrected}^j &= \rho_r^j + \rho_{RS,true}^j - \rho_{RS}^j + \Delta R_{RS}^j - \Delta R_r^j \end{aligned} \quad (4.2)$$

The limitation is that the residuals are computed and carried by the messages only for the set of satellites, which are currently used for the navigation solution. To have the correction valid, the set of satellites used by the rover must be a part of the set of satellites used by RS (the higher elevation mask was applied at the rover side to assure that).

4.1.2 SBAS corrections (MODE2)

Another option, how to obtain the PRC is SBAS. Satellite-based augmentation was described in Section 3.1. The PRCs are provided directly by the receiver via unified UBX protocol, more about UBX in Section 2.3.4. MODE2 represents a simplified type of repeater, which repeats the SBAS corrections to the rover.

The big advantage of this method is that RS does not have to know the true position because the PRCs are already calculated. The RS just use them to compute a position

correction according to rover set of used satellites. The need for the use of RS is also because the SBAS signal is weak (especially EGNOS in Europe) and it does not have to be always available for a rover itself. The RS has to be located stationary at the open area to establish the SBAS connection.

4.1.3 Application of the PRC

Calculated PRCs is applied differentially in each mode. The MODE2 involves the Correction Projection Algorithm described in Section 4.2. The solution of the position projection is then added to the measured rover position, which generates the corrected position. This method is useful since the PRC is already calculated without regard to the actual measured position by the RS. Thus, it will improve even filtered position generated by the receiver.

The MODE1 is bit trickier. It computes the difference between actual ranges (defined by the measured position of the rover) and corrected rover ranges defined by equation (4.2). This residual is then projected to the position domain and resulting position shift is added to the actual measured position. To be able to compute the corrected rover position, the range residuals have to be transmitted together with the calculated position from the rover station to RS.

The corrected position is not dependent on the receiver output filtering so the precision will be worse than actual measured position, but with improved accuracy. To get the corrected position with better precision, Kalman filter can be optionally applied at the output.

4.2 PRC Projection to Position Domain

The original idea how to use GPS receivers, which does not provide any raw pseudorange output was presented by (Kim, 2013) and (Yoon, 2014). The main formulation consists of the projection of the gathered pseudorange corrections into position domain. The algorithm allows application of the position correction directly to the rover's position coordinates. Thus, the rover's GPS device does not have to be modified and for the correction, the NMEA output is sufficient.

It is necessary to mention the method of computing the corrected coordinates using least-squares procedure in measurement domain to understand better the position projection.

$$\begin{bmatrix} \vec{X}_{DGPS} \\ B \end{bmatrix} = (H^T H)^{-1} H^T \begin{bmatrix} -e^j \rho^j + (R^j + PRC^j) \\ \vdots \\ \vdots \end{bmatrix} \quad (4.3)$$

Whereas e^j is a unit vector of line-of-sight directions to the j -th satellite. The ρ^j is a position vector from receiver to the j -th satellite. The R^j and PRC^j respectively denote the observed pseudorange (3.3) and computed pseudorange correction (3.4). B represents a receiver clock bias.

The equation (4.3) can be decomposed as follows.

$$\begin{bmatrix} \vec{X}_{DGPS} \\ B \end{bmatrix} = (H^T H)^{-1} H^T \begin{bmatrix} -e^j \rho^j + R^j \\ \vdots \\ \vdots \end{bmatrix} + (H^T H)^{-1} H^T \begin{bmatrix} PRC^j \\ \vdots \\ \vdots \end{bmatrix} \quad (4.4)$$

Where the former part denotes a standalone GPS position and the latter part is a correction in position domain according to equation (3.1). The last part is a crucial one and gives the desirable solution

$$\vec{\delta}_x = (H^T H)^{-1} H^T \overline{PRC^j} \quad (4.5)$$

The important thing to this solution is to control the satellite selection as same as in the position domain DGPS described in (3.2.1). However, in the position domain DGPS, the correction is done by differencing already computed position coordinates. Due to that, it is impossible to coordinate the set of used satellites for navigation solution in both stations.

The DGPS correction projection algorithm (DGPS-CP) (stated in (4.5)) can set-up the position domain correction with an arbitrary set of satellites. To have the position correction valid, the set of j satellites must be the same as in the rover station. Otherwise, the different constellation creates definite H matrix and PRC vector. Thus, the position correction generated by (4.5) can cause a much bigger error as proved in (Kim, 2013).

4.2.1 Projection matrix H

The remaining things are to estimate parts of the equation (4.5). Projection matrix composition is the same to the observation matrix described in Section 2.5.3. H matrix consist of line-of-sight vector cosines also called direction vectors e_j (Eq. (2.15) - yellow part). It represents the direction from the receiver position to the satellite in ECEF. Figure 4.2 illustrates the situation in ECEF coordinates system.

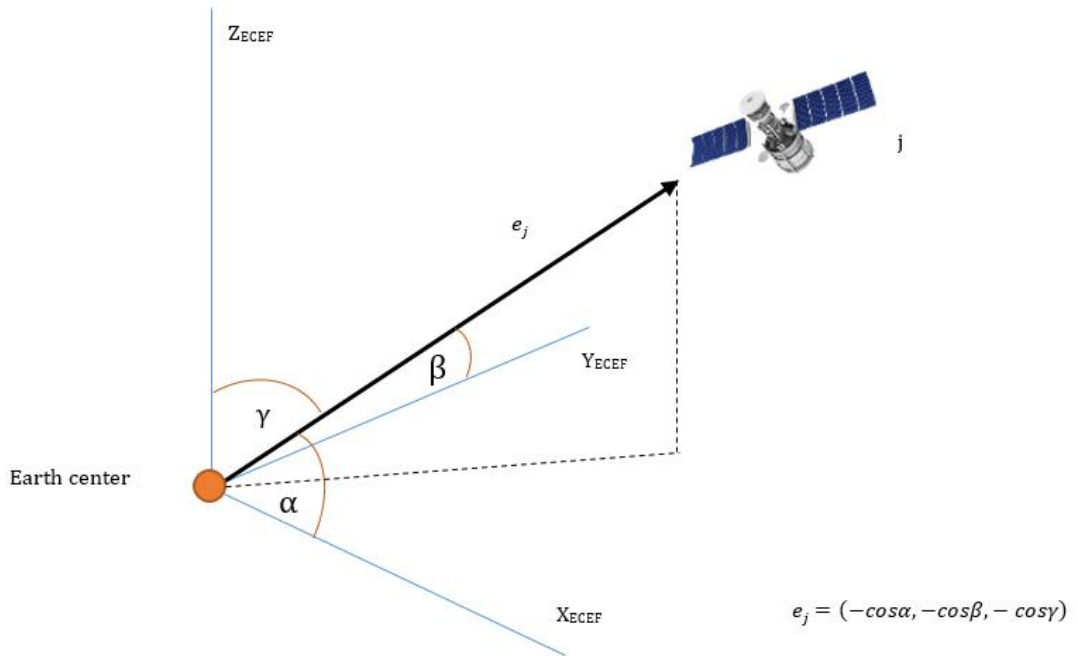


Figure 4.2. Line-of-sight vector representation in ECEF

The remaining question is how to get these parameters. There are two possible ways.

4.2.1.1 Satellite Position Calculation

The line of sight vector can be computed using the satellite position calculated from ephemeris parameters (Section 2.3.1.1). Satellite position is calculated based on actual GPS time. At first, the difference between reference time t_{ref} and actual epoch t , have to be derived.

$$\Delta t = t - t_{ref} \quad (4.6)$$

Where, both timestamps are in seconds and t_{0e} represent an ephemeris reference GPS time of the week.

$$t_{ref} = (GPSweek - 1) \times 168 \times 3600 + t_{0e} [s] \quad (4.7)$$

Mean motion represents the average angular speed required to complete one orbit, which the body completes in variable speed.

$$n = \Delta n - n_0 [rad/s] \quad (4.8)$$

Where n_0 is a reference mean motion based on gravitational constant and earth mass.

$$n_0 = \sqrt{\frac{GM}{a^3}} \quad (4.9)$$

The variable GM is based on the geodetic model definition (WGS84). It represents the product of the earth mass (including the atmosphere) and gravitational constant.

In the next step, the position of the spacecraft on the ellipse will be computed based on the Mean and True anomaly. The mean anomaly is an angle between body and pericenter, which the body has while moving on the circular orbit. True anomaly denotes a position on the elliptical orbit represented by the angle between directions of periapsis⁵ and body seen from the focus point.

$$M = M_0 + n\Delta t [rad] \quad (4.10)$$

To calculate True anomaly, the eccentric anomaly⁶ has to be computed by iteration of Kepler's equation of eccentric anomaly.

$$E_k = M + e \sin E_{k-1} [rad] \quad (4.11)$$

True anomaly,

$$\begin{aligned} \cos v &= \frac{\cos E - e}{1 - e \cos E} \\ \sin v &= \frac{\sqrt{1 - e^2} \sin E}{1 - e \cos E} \end{aligned} [rad] \quad (4.12)$$

Argument of latitude describes the position of the body moving along the Kepler's orbit in term of angle between the body and ascending node,

⁵ Periapsis – is the nearest extreme point on the elliptical orbit

⁶ Eccentric anomaly – is the angle between the periapsis and body, but observed from the ellipsoid center.

$$\begin{aligned}\delta u &= C_{uc} \cos 2(v + \omega) + C_{us} \sin 2(v + \omega) \text{ [rad]} \\ u &= v + \omega + \delta u \text{ [rad]}\end{aligned}\tag{4.13}$$

Actual radius and radius correction from the ellipsoid center and the body is defined as.

$$\begin{aligned}\delta r &= C_{rc} \cos 2(v + \omega) + C_{rs} \sin 2(v + \omega) \text{ [m]} \\ r &= a(1 - e \cos E) + \delta r \text{ [m]}\end{aligned}\tag{4.14}$$

Moreover, the inclination correction and actual inclination between the reference plane and the ellipsoid are computed.

$$\begin{aligned}\delta i &= C_{ic} \cos 2(v + \omega) + C_{is} \sin 2(v + \omega) \text{ [rad]} \\ i &= i_0 + i\Delta t + \delta i \text{ [rad]}\end{aligned}\tag{4.15}$$

Longitude of ascending node defines an angle between the ascending node and the reference direction in the reference plane.

$$\Omega = \Omega_0 + (\dot{\Omega} - \omega_e)\Delta t - \omega_e t_{0e} \text{ [rad]}\tag{4.16}$$

Where ω_e is defined by WGS84 reference geodetic system and denotes the angular speed of the earth.

Finally, the orbital plane coordinates and coordinates of the spacecraft can be derived as follows.

$$\begin{aligned}X' &= r \cos u \text{ [m]} \\ Y' &= r \sin u \text{ [m]}\end{aligned}\tag{4.17}$$

$$\begin{aligned}X &= X' \cos \Omega - Y' \sin \Omega \cos i \text{ [m]} \\ Y &= X' \sin \Omega - Y' \cos \Omega \cos i \text{ [m]} \\ Z &= Y' \sin i \text{ [m]}\end{aligned}\tag{4.18}$$

By using the true position, the computed satellite position and regarding range between these, it is possible to get the full representation of LOS vector.

As shown, the calculation of the satellite position using ephemeris is quite difficult. The calculation has to be done for each SV and epoch to get the position for all SVs required for the line of sight vectors. It consumes much time and memory because each value of the ephemeris must be stored since the position changes with time. Nevertheless, the advantage of this method is a precise line of sight vector, which is useful for more accurate position correction.

Another limitation using this approach is that low-cost receivers do not provide the raw ephemeris data by default. However, there is way how to get limited (but sufficient) ephemeris information even with the low-cost receiver. (Section 4.3.2)

There is also a supplementary solution to get the ephemeris by using CORS data. CORS (Continuously Operated Reference Station) is online service, which provides ephemeris measurement on different time scales mostly in RINEX format. Delayed data are free of charge. The minimum data update rate is typically 15 minutes, which seems to be enough for the processing since the ephemeris data are valid up to an hour. On the other hand, the RS has to establish the internet connection and periodically download the ephemeris data, which is not always possible.

4.2.1.2 Satellite Azimuth and Elevation

Another way, how to get the LOS vector is to use the NMEA GSV or UBX-SVINFO message. GSV message contains angular coordinates of each satellite with the base at the receiver location (Local coordinates frame).

The structure of the NMEA GSV message is described in Section 2.3.2. The satellite azimuth and elevation angles are used to estimate the direction vector.

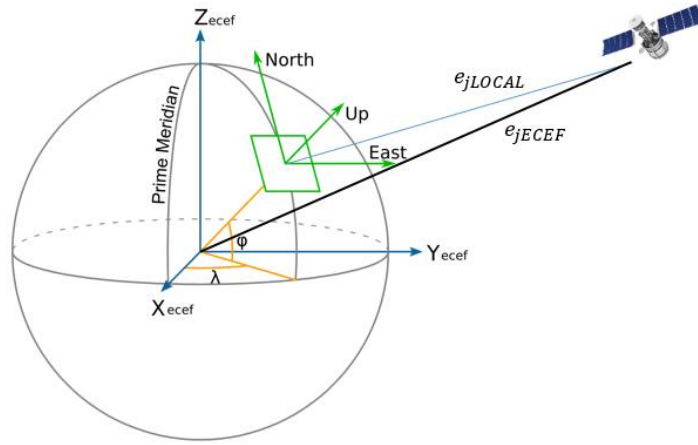


Figure 4.3. Local and ECEF coordinates relation

The local coordinates of LOS are computed directly from the azimuth Az_j and elevation El_j angles.

$$\overrightarrow{e_{j,LOCAL}} = \begin{bmatrix} \sin Az_j \cos El_j \\ \cos Az_j \cos El_j \\ \sin El_j \end{bmatrix} \quad (4.19)$$

The rotation matrix R is used to convert the local coordinates to ECEF coordinates system.

$$\overrightarrow{e_{j,ECEF}}^T = \begin{bmatrix} -\sin \lambda & -\cos \lambda \sin \varphi & \cos \lambda \cos \varphi \\ \cos \lambda & -\sin \lambda \sin \varphi & \sin \lambda \cos \varphi \\ 0 & \cos \varphi & \sin \varphi \end{bmatrix} \times \overrightarrow{e_{j,LOCAL}} \quad (4.20)$$

Where λ and φ refer to the longitude and latitude of the receiver position. This method is very easy to implement, but it also brings some difficulty in precision. The NMEA GSV sentence provides only azimuth and elevation in full degrees (integers) that cause uncertainty in the satellite direction vector. However, as proved in (Park, 2014), the error posed by this uncertainty in position was not higher than 0.2m.

4.3 GPS Receivers Setting

The uBlox GPS receivers are used for application and testing in both stations. There are several models provided by uBlox with different prices. Concrete model for the application is uBlox NEO6m (Figure 3.3). It is a basic GPS receiver, which does not provide any RAW data output.

4.3.1 Rover Station

The main goal is to make just the minimal modification from default functionality of the GPS receiver to make the system easy to implement.

The one thing that should be configured is the minimal satellite elevation threshold. It allows filter out SVs, which have a low elevation (default 5°). By setting a higher elevation mask at the rover side, it secures that the number of the used satellite will always correspond to the RS.

The minimum GPS requirements are to have enabled NMEA GPGLL, GGA, UBX-SVINFO, UBX-POSECEF, and UBX-TIMEGPS (to get the actual TOW for synchronization).


4.3.2 Reference Station (RS)

The GPS receiver at the RS side requires a larger intervention into its setting. To enable the ephemeris, the specific setup code is sent to the device during start-up and after defined time interval. Some setting can be done directly in U-Center (selection of particular NMEA sentence, output frequency, and so on). There are two types of messages. **Polled** is refreshed when request code is sent and **Periodic**, which is updated every cycle defined by output frequency. Since the receiver sending the subframe data via UART, it is better to set the higher speed of the serial channel. In this case, the baud rate was set to 57600Bd.

To enable full functionality, following messages have to be enabled.

Table 7. Messages required by RS

Protocol	Type	Name	Description
NMEA	Periodic	GPGLL, GPRMC	Provides minimum positioning information such as (Latitude, Longitude, UTC)
UBX	Periodic	NAV-SVINFO	Provides range residual for active satellites
UBX	Periodic	NAV-SVINFO	Provides information about satellites in view (Azimuth, Elevation)
UBX	Polled	AID-EPH	Gives subframe data 1-3 from navigation message (Ephemeris)
UBX	Periodic	NAV-POSECEF	Directly provides ECEF coordinates
UBX	Periodic	NAV-TIMEGPS	Contains the GPS week/time
UBX	Periodic	NAV-SBAS	Provides SBAS correction data

 Require request message

Other messages, which are provided by the receiver should be disabled to reduce the load of the Arduino device, while parsing the data. However, it will not affect the functionality if a strange frame will be received.

4.4 Algorithm Implementation

This Section discusses the application of the key parts of the algorithm to the Arduino environment. It is done separately for RS and rover station.

The Arduino environment is based on C#/C++ language with a lot of simplification, which makes them very easy to use. Arduino and the most of the developed libraries are open source, so it allows the user to realize many different projects.

Both stations using the same GPS receiver and the data, which are transmitted, are parsed by the Arduino units. For this purpose, the TinyGPS++ library was used. In general, TinyGPS provides the reading of two basic NMEA message GLL and RMC. However, the library also offers a function to call any parameter from arbitrary NMEA sentence (more in (Hart, 2016)). To be able to read the UBX data frames, the library was modified by adding special conditions. These conditions are used to identify the beginning of the UBX frame (0xB5, 0x62), which allows parsing the UBX data separately from NMEA. The parsing is done by saving the UBX stream to the array, which is then parsed to representative variables. Variables are then called from the main program as needed. Each UBX frame is recorded to the array individually.

4.4.1 Rover Station

The rover station acts as the initiator of the whole correction calculation process. The triggering event is when the new GPS measurement is received. The end of the GPS measurement is indicated by the Boolean flag (`time.end`), which is set to HIGH when the last word of the last frame is parsed. In the case when only important messages are enabled (Table 7), the last sentence in the queue is NMEA GPGGA and last word parsed is the geoid separation, so the flag is located there. It is done in the modified TinyGPS library. It is not a rule that GPGGA will always be the last, so it is important to check the message flow by using uCenter and place the flag to appropriate position in the library.

In the main sketch, there is a function `programSend()`, which is continuously called. The function is waiting for the triggering flag by calling the library function `gps.time.isFinished()`. When the respond is positive, the function `programSend()` will proceed once. The main purpose of this feature is to collect all the necessary data for the correction computation, pack them into an HEX string and send it to the RS. The first flow chart represents the algorithm principal in Figure 4.4.

The second function in the main loop `programRec()` is used for sending the received data from RS to the user. The function proceeds in specified interval `OUT_GAP`, which is equal to GPS receiver output frequency (1Hz). The function offers two options of which data will be sent to the user. The first option occurs when there is new corrected data from RS. The reception of corrected data must be done in the interval defined by `OUT_GAP`. Otherwise, the new corrected data will be discarded and function switch to the second option. The second alternative will just send the original, uncorrected NMEA data to the output.

This principal ensures that user will always be provided by the actual GPS data whether corrected or uncorrected. The rover unit provides only the basic NMEA sentences GPGGA and GPRMC since these messages are mostly used by the GPS-equipped devices because it provides a valuable positioning data.

When the corrected position is applied, it is indicated in the GPGGA fix quality word (the 6th word in the sentence). In addition, the RS ID and time from the last received correction is captured by the GPGGA (the 13th and 14th word in the sentence).

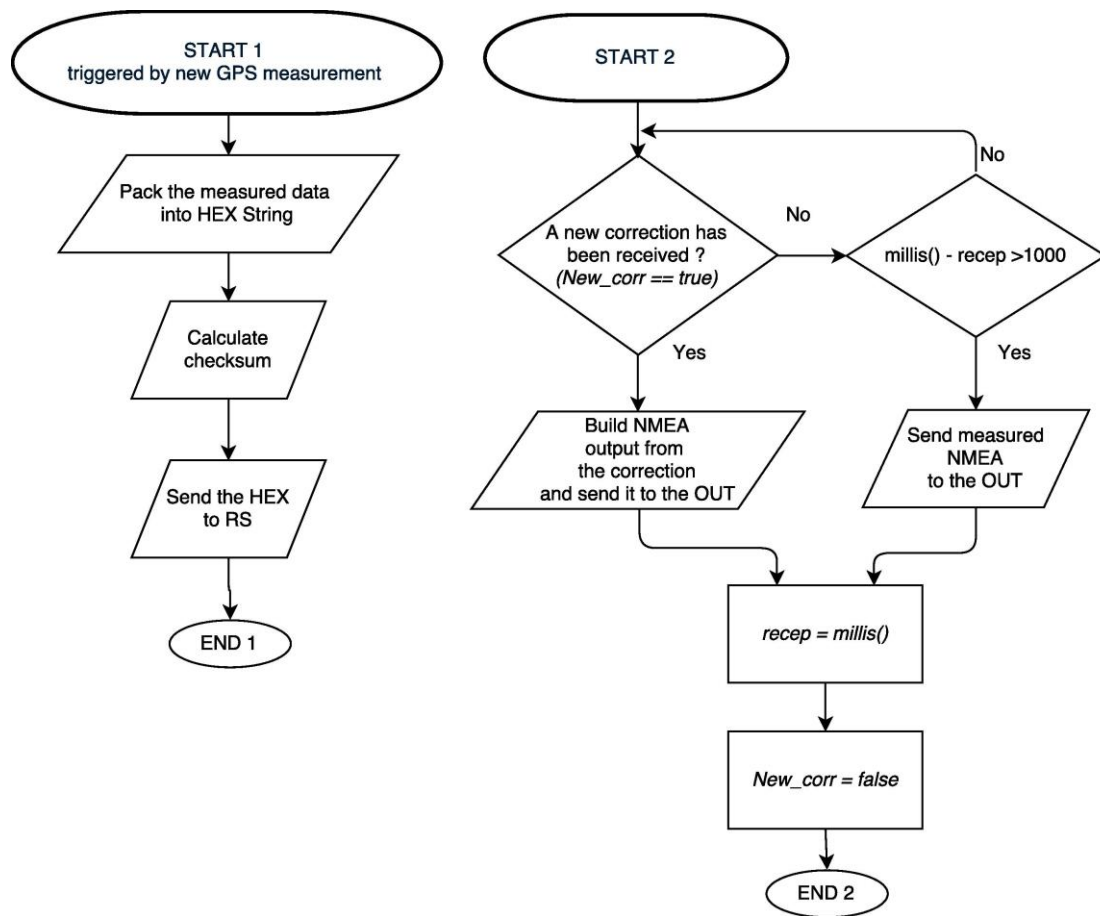


Figure 4.4. Rover algorithms *programSend*(left) and *programRec*(right)

4.4.2 Reference Station (RS)

The RS is responsible for all calculations and provides system status info via attached TFT display. As same as the rover, the RS continuously capturing the GPS data from the connected GPS receiver and parsing them using the GPS library. The libraries used by RS and rover are not the same since both devices require different data. The library is adjusted individually for a particular station to save the memory space, especially on the rover side.

In the first part of the main sketch, there are optionally adjustable variables. These values should not be changed since it is set according to the shield hardware design. Section 5

The program also offers the debugging options, which can be enabled by uncommenting the line `#define DEBUG`. It will transmit a basic GPS data and RS status via the main Serial interface.

As described in Section 4.1 the RS will offer two modes of position correction computation. The main algorithm of the both modes is shown in Figure 4.5 and Figure 4.6. respectively.

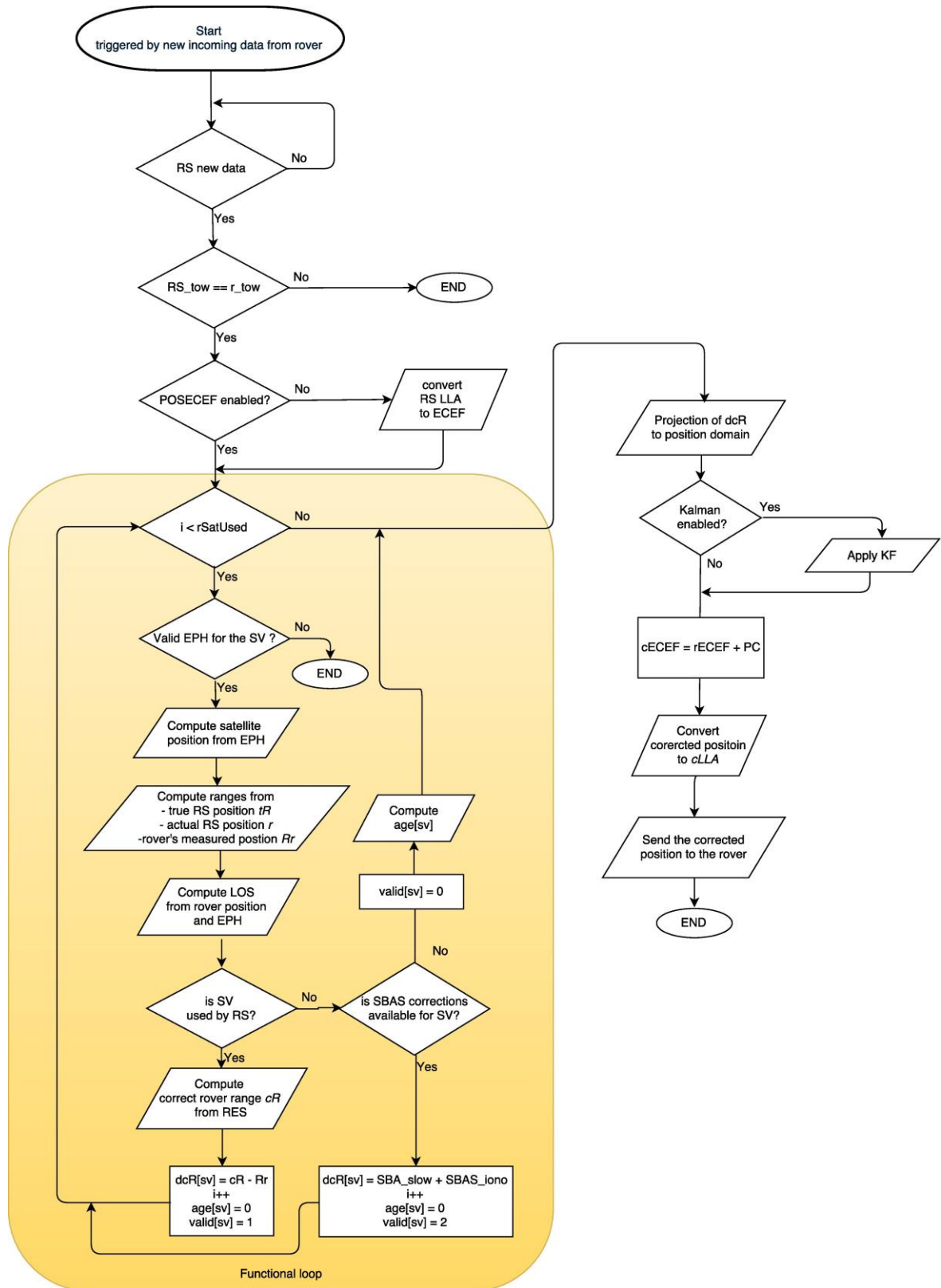


Figure 4.5. RS MODE 1

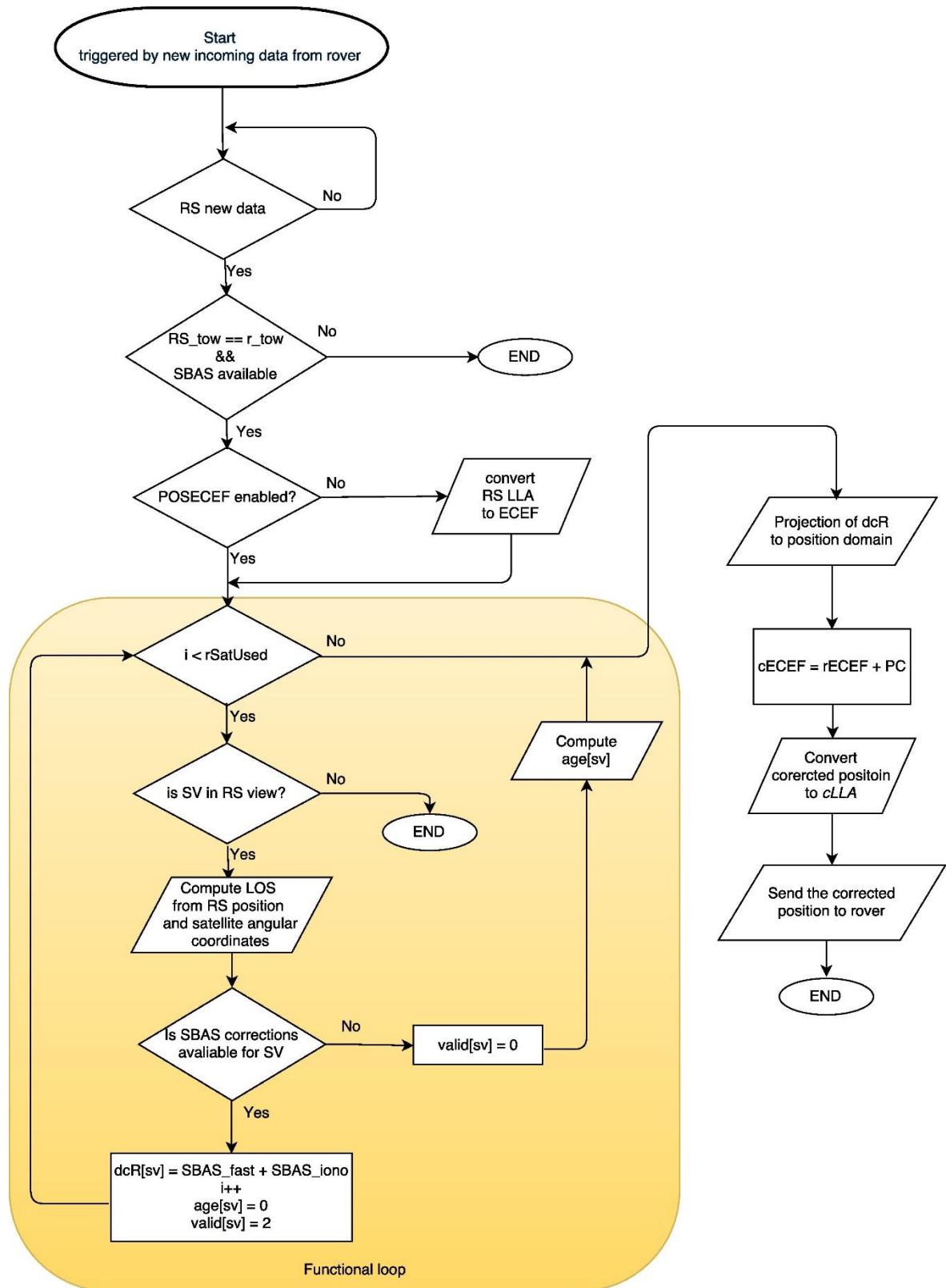


Figure 4.6. RS MODE2

4.4.2.1 MODE 1 (Range residuals)

The MODE1 algorithm is represented by the function `coreRES()` in the main Arduino sketch. Once RS receives new data from the rover, it calls the function. The next step is to check if the data are from the same sequence. This is done by comparing the timestamps `tow` from both stations. If the signal from the receiver is delayed and the data sets will be from the different sequence, the correction computation terminates here and rover will not be provided by any corrected position. The GPS modules in both stations are set to synchronize the output by the GPS time. Thus, the time of the week generated by both receivers should be the same. This simple rule ensures the synchronization of both stations and avoids generating corrections for data, which are from different sequences.

In the next step, the algorithm checks if there are actual ECEF coordinates of the RS available. It is done by calling the variable `posecef`, which is set to HIGH when new UBX frame containing the ECEF coordinates has arrived. When the variable is false, the ECEF position is calculated from geodetic coordinates set.

When all input conditions are fulfilled, the algorithm proceeds to generate LOS and range corrections for all SVs used by the rover. The number of SVs used by rover is contained in the variable `rSatUsed` and specific SV values is stored in array `rSatUsedList`. The function goes through all the SVs stored in the array (Functional loop) and checks if there are valid ephemerides. If no, the function terminates, and no correction will be generated. The valid ephemerides are necessary to get the satellite coordinates (as described in Section 4.2.1.1) and for the ranges computations. Otherwise, if all ephemerides are available the program will continue to compute the satellite position, LOS from rover position, and ranges:

- From the true RS position to the satellite (τ_r)
- From the actual RS position to the satellite (r)
- From actual rover position to the satellite (R_r)

The next step is to check if RS also uses the rover's satellite. This is done in the same manner as the previous loop. However, the list of SVs used by rover is now compared with satellites employed by RS. If the result is positive, the PRC is computed by equation (4.2). In case that rover's SV is not used by RS, the function tries to check if there is a SBAS correction available and optionally use that correction as PRC for the currently listed SV. It is indicated by setting the flag `valid[sv]` to appropriate value.

- Rover SV is used by the RS (`valid[sv] = 1`)
- Rover SV is not used by the RS, but SBAS is available (`valid[sv] = 2`)
- Rover SV is not used by the RS and SBAS is not available (`valid[sv] = 0`)
 - In case that `valid[sv] = 0` the function uses the previous value of PRC and update the age.

When the Functional loop is finished, it proceeds to the projection of PRC to position domain. The position correction PC is added to the actual rover position, which creates the corrected rover position $cECEF$. As discussed in Section 4.1.3, this method generates corrected position from originally observed pseudoranges, so the position is improved only in terms of accuracy. The optional Kalman filter can be applied, to improve the precision of corrected position coordinates,

The corrected position is then packed to HEX String and sends back to the rover, which closes the whole process of correction computation.

4.4.2.2 MODE 2 (SBAS)

The second mode principal is shown in Figure 4.6. As seen from the figure, it works similarly as the mode 1. The initial stage of achieving synchronization has one more condition that checks if the SBAS correction data are available. If not the function terminates because the SBAS data are crucial for this mode since SBAS provides range corrections. The MODE2 functional loop is simpler since PRCs are already computed. It goes through all the satellites used by the rover and checks whether the SV used by rover is also in RS view. It is a critical condition because the RS must have the SV in view to get angular coordinates of the satellite from UBX-NAV-SVINFO. These angular coordinates are required for LOS computation, as described in Section 4.2.1.2.

When SBAS correction is available, and SV is in RS view, the functional loop proceeds to compute the LOS from Azimuth and Elevation angles and range corrections from SBAS. When the functional loop is finished, the last remaining thing is to project the range corrections to position domain and generates the corrected position, which is sent back to the rover.

Note that the UBX-NAV-SBAS provides fast and slow corrections mixed as one variable.

4.5 Data Transmission

Data transmission between rover and RS is realized by two cheap Wi-Fi modules ESP8266.

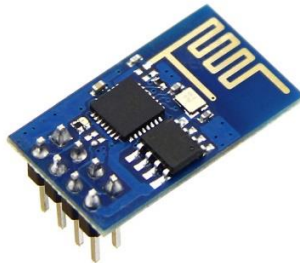


Figure 4.7. ESP8266 Wi-Fi module (www.seeedstudio.com)

This module is controlled via the serial connection by a set of AT commands. The AT commands are special ASCII strings, which are predefined for different devices (i.e. GSM). The set of AT commands defined for the ESP8266 are described in (Espressif, ESP8266 AT Instruction Set, 2015). The weakness of AT commands is the defined set of AT commands has to be sent to the unit at each startup to establish the functionality. This drawback can be removed since the ESP8266 offers direct programming in LUA language. Programming the device directly via LUA is less intuitive and more difficult. However, the program is saved in the ESP EEPROM and it is automatically loaded at each startup (LUA code for both stations is included in Appendix).

The ESP module at the RS side runs as an Access point, and rover is set to a client mode. Immediately after the startup, the units are set to their roles and client try to connect to the AP with defined SSID and password. When the connection is established, specific bytes are sent through the channel mutually to the both stations (RS, rover). It

informs the stations that connection is established and the device is ready. The connection status is indicated on the display (RS) and by LED illumination (rover).

4.5.1 Data Formats

Data, which are sent over the TCP connection are carrying the position correction are using the simplified UBX communication protocol.

Sync CHAR 0xB5	Sync CHAR 0x62	Class 0xCC	Type	Payload (including length)	CK_A	CK_B
----------------------	----------------------	---------------	------	-------------------------------	------	------

Figure 4.8. UBX packet format used for correction transmission

Reasons for using the UBX protocol is that the same algorithm can parse it as the other UBX data frames. That makes it simpler and consumes less MC memory.

The position correction packet is denoted by class 0xCC, which is not reserved by any other UBX packet. The next byte represents the data type. There several types of the transmission correction packet:

- 0xFF⁷ – This type informs the station that there is no connection between rover and RS. This packet is periodically sent in a one-second interval.
- 0xFE⁷ – denotes that connection was established. This byte is followed by the station ID in HEX. This type of sentence sends only once.
- 0x01 – denotes a new data from rover
- 0x02 – denotes a position correction packet

The payload length of message type 0x01 is variable, and it is derived from a number of used satellites. The payload structure is shown in *Figure 4.9*.

Used Sats 1B	Voltage (*10) 1B	TOW 4B	SOG (*100) 4B	COG 4B	ECEF_X 4B	ECEF_Y 4B	ECEF_Z 4B	SVs 1B*used_sats	Residuals 2B*used_sats	0 1B
--------------------	------------------------	-----------	---------------------	-----------	--------------	--------------	--------------	---------------------	---------------------------	---------

Figure 4.9. Payload of message type 0x01

The payload of message type 0x02 has a constant length and contains correct position in LLA⁸.

0 1B	TOW 4B	LAT 4B	LON 4B	HAE 4B	0 1B
---------	-----------	-----------	-----------	-----------	---------

Figure 4.10. Payload of message type 0x02

The data are sent as the ASCII string, which represents the HEX data divided by comma. This method shows better performance than sending the binary data itself and it is easier to parse the data on the other side. The data are thus packed into a HEX string and the last character “*” trigger the ESP to send the string. The receiver side then parses the data and sends it to the station in binary format via UART.

⁷ This type of packet does not include the payload and checksum since they are not carry any information. They are used just for the connection status indication

⁸ Corrected position is send in format dddmm.mmmm*10E5

4.6 Self-Accuracy Monitoring

This section discusses a proposal of accuracy and integrity monitoring of the Arduino DPGS.

4.6.1 Integrity Monitoring

The station monitoring is necessary to be able to detect any deficiencies, which may make the system unstable or unable to run in a proper way. The main integrity checks are done by the RS, which is more vulnerable for any malfunctions. The RS integrity monitoring runs in two steps. The first step is done in the boot sequence, which occur right after the startup. In the boot sequence, the RS checks whether there are all key features available, which are needed for proper functionality.

- RS GPS data availability
 - Check if the GPS receiver is connected
 - Check if there are enabled all required UBX sentences
- Position fix of RS (must be 3D)
- True position⁹

The RS GPS receiver activity check is done by calling the Arduino `Serial.availability()` function. This function will return zero if there are no incoming data to the defined serial port. It is useful when there is a wiring problem or basically if the GPS receiver does not work properly.

The UBX sentences are checked by calling a special boolean flag, which was set for each message type in the library. Once the UBX packet arrives, the flag is set to true, which indicates that particular messages are enabled. The 3D position fix is recognized from the NMEA GPWGA (6), and the true position of the RS must be entered by the user to enable the MODE 1.

All these requirements must be fulfilled before the RS will accept any data from the rover. It ensures that all parts of the system are accessible, and the correction computation will be done properly. All these initial steps are displayed on LCD. Thus, a user knows, which exact part is not accessible.

The second part of integrity monitoring is done when the device is in operation. It continuously checks whether there are all required parameters available for selected mode.

- MODE 1 (Range residuals)
 - Ephemerides
 - Rover satellite is used by RS
 - SBAS correction is available
 - Ranger residuals
- MODE 2 (SBAS)
 - SBAS signal availability
 - SBAS correction for particular rover satellites
 - Rover satellites are in RS view

For the MODE 1, the ephemerides are key features since they are used for satellite position computation. The satellite position is necessary for ranges computation, which is important for range corrections estimation. Due to that importance, each

⁹ Only for MODE 1

satellite ephemeris is equipped with a flag in the GPS library. It indicates whether the ephemeris was downloaded.

The comparison of used satellites is done in the functional loop. When there is a match, it sets the variable `valid[sv]`, which also indicates how the corrections are obtained. The variable also says whether there are range residuals available for defined SV. When the SV is in the RS navigation solution, this means, that the range residual is automatically computed by the receiver.

For the MODE2 the integrity monitoring is composed by checking the SBAS signal availability and the correction for particular satellites. It is done by checking the availability flag, which is carried by the UBX-NAV-SBAS message and correction validity by reading the boolean flag, which was set for each SV. The comparison of rover satellites set is made in the functional loop similarly as in MODE1.

All these values are displayed on LCD (See section 5.3)

4.6.2 Accuracy Monitoring

The accuracy monitoring is realized by computation of statistical values, which describes the actual correction behavior and give an idea about actual position error.

The RS computes its accuracy easily from the actual measured position and true position. This is done by transforming the coordinates to local coordinates system with origin in the true position. Actual measured position then shows an actual position error. The RS store the accuracy values in an array, which has following structure.

actual H error	actual V error	Horizontal DRMS	Horizontal 2DRMS	Vertical RMS	Horizontal CEP	SEP
----------------------	----------------------	--------------------	---------------------	-----------------	-------------------	-----

Figure 4.11. RS Accuracy monitoring array structure

In the case of MODE2, there is no information about the true position. The accuracy values are then computed with respect to SBAS range correction.

The rover does not measure its accuracy because there is no information about the true position. The actual error measurements are computed concerning the average position. It results in precision measurements, which is stored in the array (figure 4.12). In other words, the RS is capturing the rover position from UBX frame via Wi-Fi and then computes the average. The actual position is then compared with this average, which results in actual precision measurements. This precision measurement is only valid for static measurements. When the rover produces any movement, this measurement will produce a nonsense values.

actual H precision	actual V precision	Horizontal RMS	Vertical RMS
--------------------------	--------------------------	-------------------	-----------------

Figure 4.12. Rover precision monitoring array structure

With each new measurement, the position is added to the previous one to be able to compute the standard deviations as described in Section 2.4.2. Standard deviations are computed for each direction. The values are stored in 8bytes variables thus; the computation of the average position is limited to approximately 1E10 measurements¹⁰. This number regards to hundreds of years of continuous operation. The range, over

¹⁰ The number was computed from approximate value of ECEF coordinates

which the average is computed, can be set in the main sketch. The average variables are set to zero when the range is reached, and the measurement starts from the beginning.

5 Hardware Design

This chapter discusses the proposal of hardware structure and design of both stations. The first section briefly describes the individual parts of which the whole system is composed. It is followed by the wiring diagrams of both stations and design of the PCB shield. The last section shows how the system should be controlled and describes an information displayed on LCD.

5.1 Parts Description

This section briefly describes particular units used by the rover and RS. Both stations are using the same GPS receiver. It is the uBlox receiver NEO 6M.

5.1.1 GPS Receivers

The uBlox NEO6M is shown in Figure 3.3. NEO6M modules are designed for operational voltage 3.3V. Nevertheless, the board shown in Figure 3.3 is equipped with 3.3V stabilizer so that 5V can supply it. The device communicates with the Arduino board via Serial communication (UART). Since the module is supplied by 3.3V, the Serial connection must be shifted to 3.3V to avoid the module destruction.

From the functional point of view, the receiver offers basic positioning service in NMEA and UBX form. It can either accept the RTCM DGPS corrections. The NEO6 is equipped with SBAS engine, which can reserve up to three channels for SBAS (uBlox A. , 2011). The GPS module is one band only. It can receive the GPS L1 C/A code only. More about NEO6M can be found in the product summary (uBlox A. , 2011).

5.1.2 Wi-Fi ESP8266

See Section 4.5.

5.1.3 QVGA ILI9341

The information about station status and other information are provided via an LCD. LCD also works as GUI, which allows an initial setup of the device.

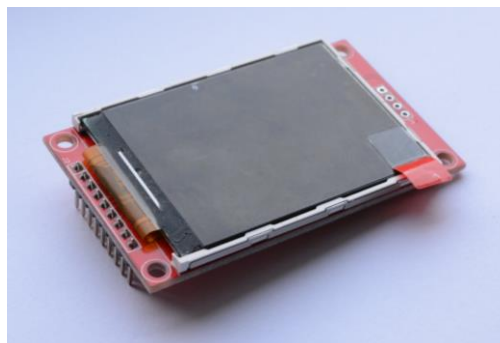


Figure 5.1. ILI9341 QVGA

The LCD has resolution 320x240px with 262K color. The supply voltage is as same as the GPS module 3.3v. The communication is provided via 8bits serial interface (SPI). More about technical data can be found in technical documentation (ILITEK).

The access to the LCD from the main program in RS is arranged via optimized ILI9341 library. This library is specially optimized for ARM microcontrollers, and it provides excellent performance than other libraries. The LCD control is very easy using this library. There are many functions, which are used for printing text, shapes, or numbers. Follow the official web pages http://pjrc.com/store/display_ili9341.html for more information about the library.

5.1.4 Arduino

The Arduino unit is a core of the whole system. It provides all the computations and control over the entire system.

The RS is using the Arduino Due. It is an enhanced unit, which is equipped with ARM 32-bit microcontroller, 54 digital IO, 4 UARTs, and 12 analog IO (Arduino, Arduino Due, 2016). The important feature of the ARM processor is that its supply power is 3.3V. That makes it easier for GPS and Wi-Fi modules since they also operate on 3.3V.

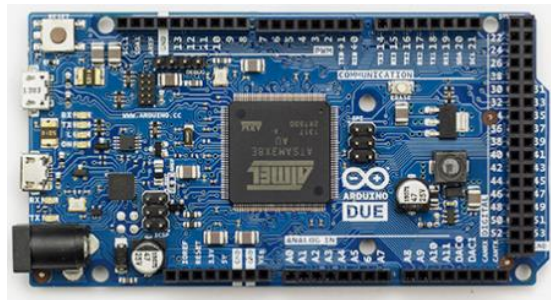


Figure 5.2. Arduino Due (Arduino, Arduino Due, 2016)

The rover station is using Arduino Mini. This device is equipped with Atmel AVR MEGA328. It is a 16-bit processor equipped with 14 digital, eight analog IO and one UART. The AVR processors operate on 5V levels. It means that the communication with GPS and Wi-Fi must be adjusted with the passive level shifter (Arduino, Arduino Pro Mini, 2016). The most important thing is to avoid overvoltage of the 3.3V modules. However, the output from these devices does not have to be adjusted since the 5V Arduino is capable of reading 3.3V logic because the minimum voltage of logic “1” is 2.7V.

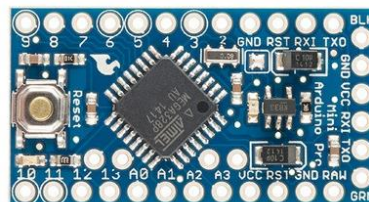


Figure 5.3. Arduino Pro Mini (Arduino, Arduino Pro Mini, 2016)

5.1.5 Wiring Diagram

The RS is designed as a shield, which fits the Arduino Due device. The RS shield consists of only of pull-up resistors and a voltage regulator. The additional Voltage

regulator was used due to high power requirements of the ESP8266 unit, which internal power regulator, cannot provide.

The wiring diagrams were done in EAGLE environment. They are presented for both stations in following figures.

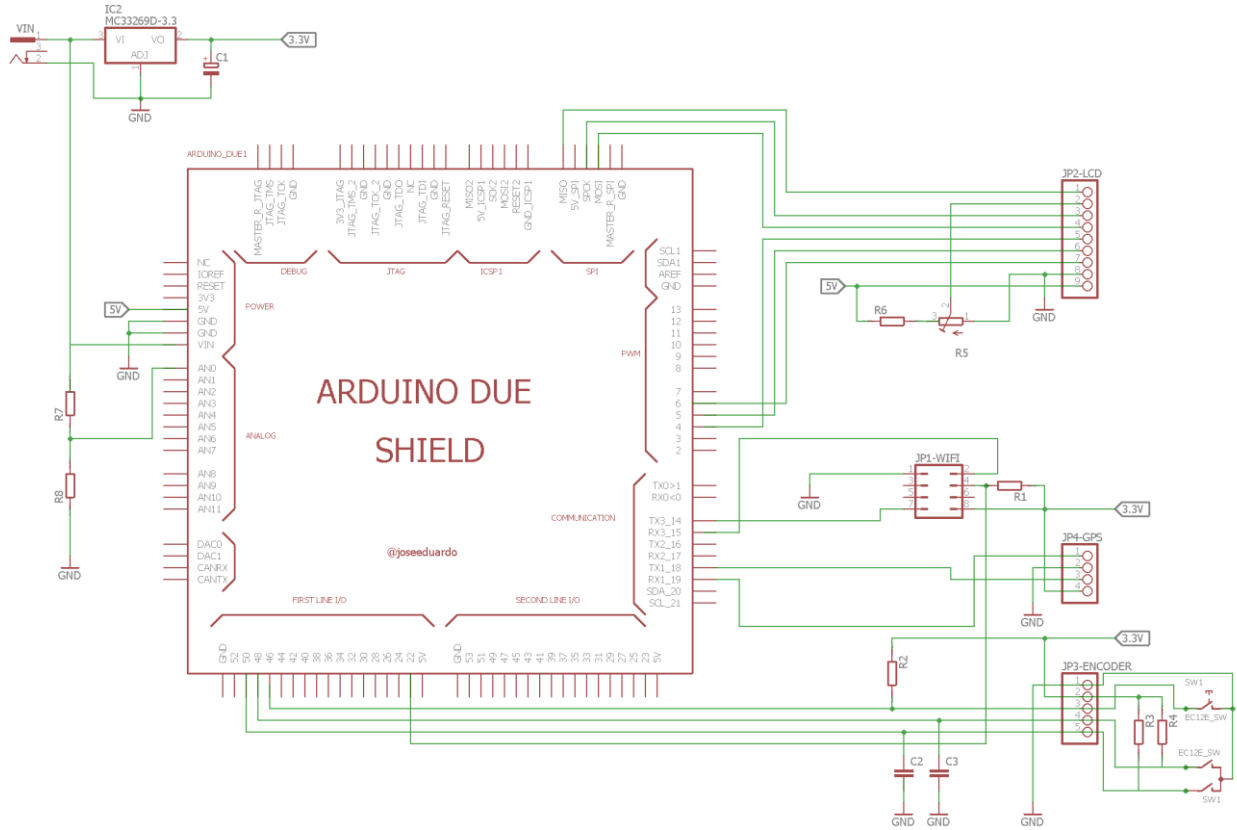


Figure 5.4. RS shield Wiring Diagram

A rover device shield (Figure 5.5) consists of a 3.3V voltage regulator for powering the Wi-Fi and GPS module. Due to that Arduino Pro MINI is a 5V logic, the signal wires must use a voltage divider to avoid overloading of the GPS and Wi-Fi modules. The input of the rover device is represented by the JP4-GPS and JP5-Wi-Fi, which collects a GPS data and RS corrections. The output is defined by JP1, which provides the NMEA GGA, and RMC sentences with the corrected position. The input voltage of both devices can be up to 16V.

The component list of both devices is included in Appendix B.

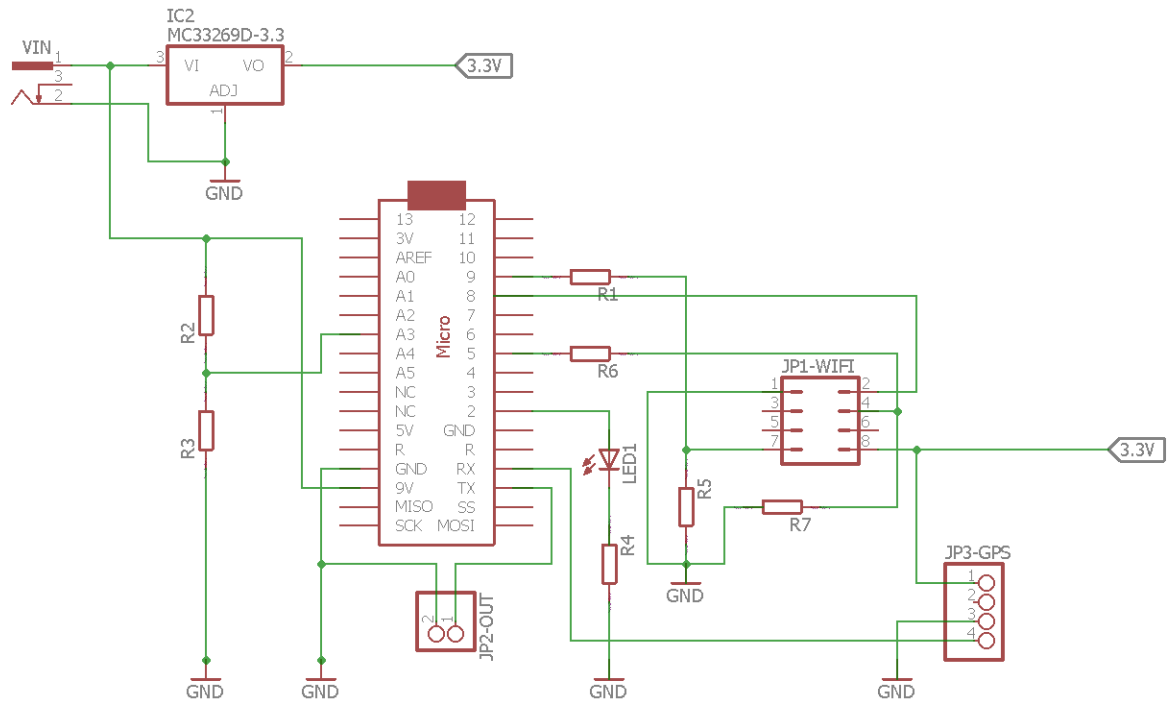


Figure 5.5. Rover shield Wiring Diagram

5.2 PCB Design

The physical shield proposal is intended on 1-sided PCB board. The PCB design was done in Sprint Layout (Figure 5.6).

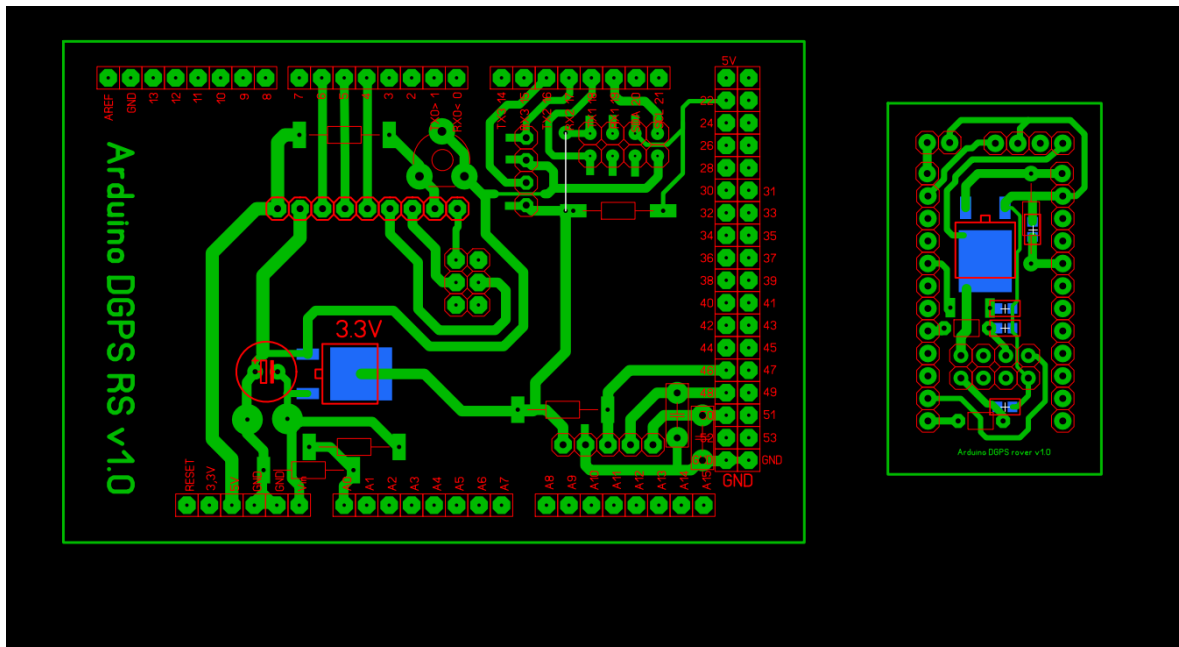


Figure 5.6. PCB layout (RS right, Rover left)

5.3 In Operation

This section contains operational instructions, which shows how to set the Arduino DGPS into operation. It will be described together with a legend, which helps to understand values on the LCD.

It is better to power up the RS first. It ensures that the RS Wi-Fi module will be set to AP mode before the rover starts surveying the network. When the rover is powered later, it may cause that connection pending will take a longer time.

After the RS is powered up, the welcome screen will be shown for a few seconds followed by the initial screen.

The initial screen shows the results of the integrity monitoring (step one) as described in Section 4.6.1. After the primary device check, a user is asked to select the mode, under which the RS will operate.

5.3.1 RS MODE1 (RES)

When Mode1 is selected, the RS checks whether there is GPS 3D position fix available. If yes, the form for the setting of the true position will appear. The user then set the true position of the RS derived from a map or gathered by a long-term measurement. Controlling of the RS is done by the rotary encoder with a confirmation button. To edit the value of the true position, the user has to select the desired value and then by pressing the button enter the editing mode. To exit the editing mode a user just presses the button again.

After the true position is set, a user selects whether the Kalman filter should be enabled or not. The selection is done by confirming the K_f Off or K_f On button on the screen. The program proceeds to the main screen after one of these selections is confirmed.

The main screen is divided into four sections. The upper (section one) is used for general information about current time and date gathered from the GPS. In the right upper corner, there is an annunciator dot, which indicates the communication with the rover. Every time the new rover data arrive, it is signaled by one blink of the dot. Similarly, when new correction is sent to the rover, it is indicated by the second blink of the dot.

The second section is used for actual condition monitoring of the RS (Section 4.6.1). This is done by displaying the most important values needed for smooth operation. See Figure 6.7.

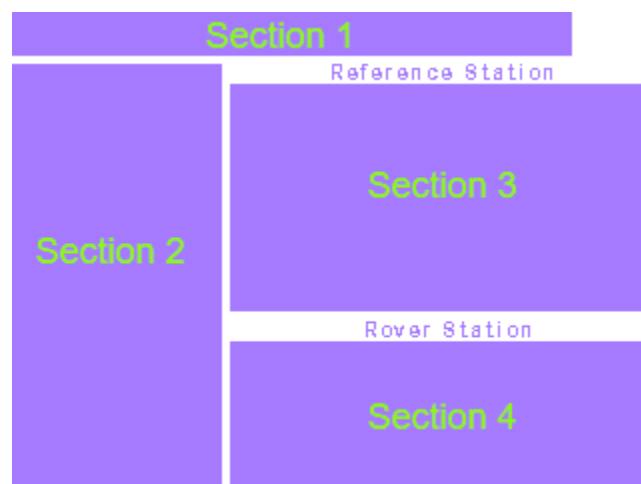


Figure 5.7. Display sections

The third and fourth section is used for the RS and rover status indication. It is structured as shown in Table 8 and Table 9.

Table 8. RS display section

Row No.	Displayed info
1	Actual measured position in LLA format
2	RS accuracy measurement
3	RS state (Week, TOW, number of used satellites, SBAS, Voltage)
4	Running time

Table 9. Rover displays section

Row No.	Displayed info
1	Corrected rover position in LLA format
2	Rover precision values
3	Rover status (Age of last correction, SOG, COG, Satellites, Voltage)

5.3.2 RS MODE2 (SBAS)

The main screen for MODE2 is practically the same as for MODE1. There is only one difference in the second section where there is only one flag signaling the SBAS corrections availability for defined SV.

5.3.3 Rover Station

The rover is equipped with signalization LED. The LED can signalize four states.

- OFF – The device has not established connection with RS
- ON – Connection with RS is established, but no data being sent to the RS
- One Blink per sec – data has been forwarded to the rover, but there is no respond from the RS
- Two Blinks per sec – the first blink signalizes that information has been sent as from the previous case and the second blink represents the received correction.

When the rover is powered up, it will automatically start surveying the network and connect to the RS SSID. After the connection was established and GPS fix is available, it automatically starts to send the data to the RS for correction. Then, the rover waits for corrected data. This algorithm proceeds as described in Section 4.4.1.

6 Test results

This chapter evaluates the Arduino DGPS solution regarding static and dynamic measurements, precision, and accuracy estimates, device technical requirements and comparison with similar projects.

Measurements were done on small parking lots near to the Roxen lake in Östergötland. There were two different places as shown in the figure below.

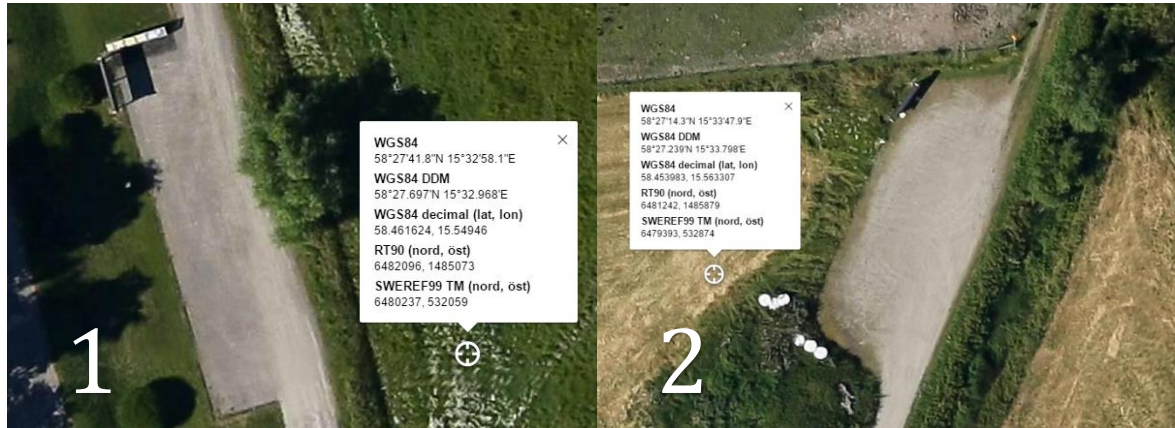


Figure 6.1. Location of the test fields (<http://kartor.eniro.se>)

The reason of selection of these two locations was the amount of reference points, which are used as a ground true. The Eniro maps were chosen as a source since they provide much more resolution over Sweden than i.e. Google.

6.1 Static measurements

The static measurements were done for both modes, each at different time and place. The measurements take approximately 45 minutes and during that, the rover and RS preserved its position.

6.1.1 MODE1 (RES)

The static measurement for the mode1 was done during the early afternoon on spot two, on Tuesday 31st of May 2016. The weather was partly cloudy, approximately 23°C, and light wind.



Figure 6.2. RS (left) Rover (right)

The result of the static measurement is presented in Figure 6.3. As seen from the illustration there is obvious accuracy improvement. However, the precision of the corrected position is a bit worse. The cyan measurement represents the filtered position by linear Kalman filter. The filter gives a small precision improvement, but it is not that precise as the original measurement.

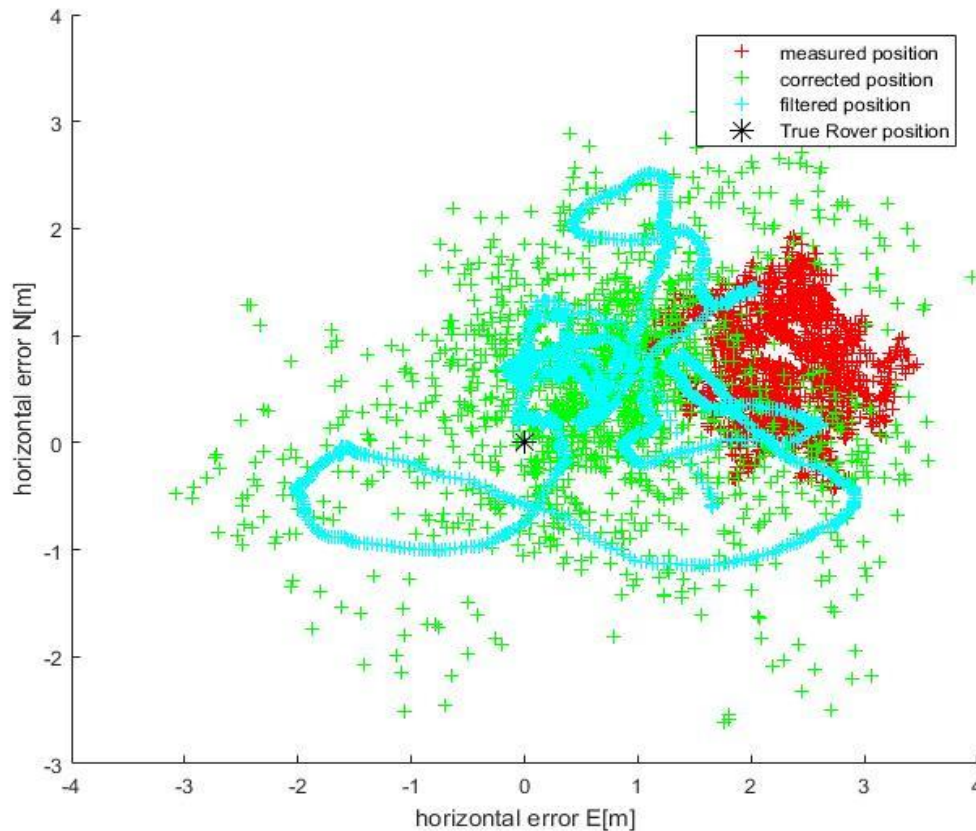


Figure 6.3. MODE1 - Static measurement

In the vertical plane, the accuracy improvement is not so apparent. The vertical data are very noisy. However, in the later part of the measurement the vertical error is quite low. From the Figure 6.4 is evident, that the largest error is resolved by the change of used satellites for which, the correction was calculated.

The horizontal accuracy improvement is also apparent from standard deviations and its RMS values as shown in table

Table 10. Accuracy comparison MODE1

Horizontal	Original	Corrected	Filtered
DRMS	2.47 m	1.72 m	1.57 m
CEP (50%)	1.93 m	1.43 m	1.31 m
Vertical			
σ_{up}	1.22 m	2.83 m	2.78 m

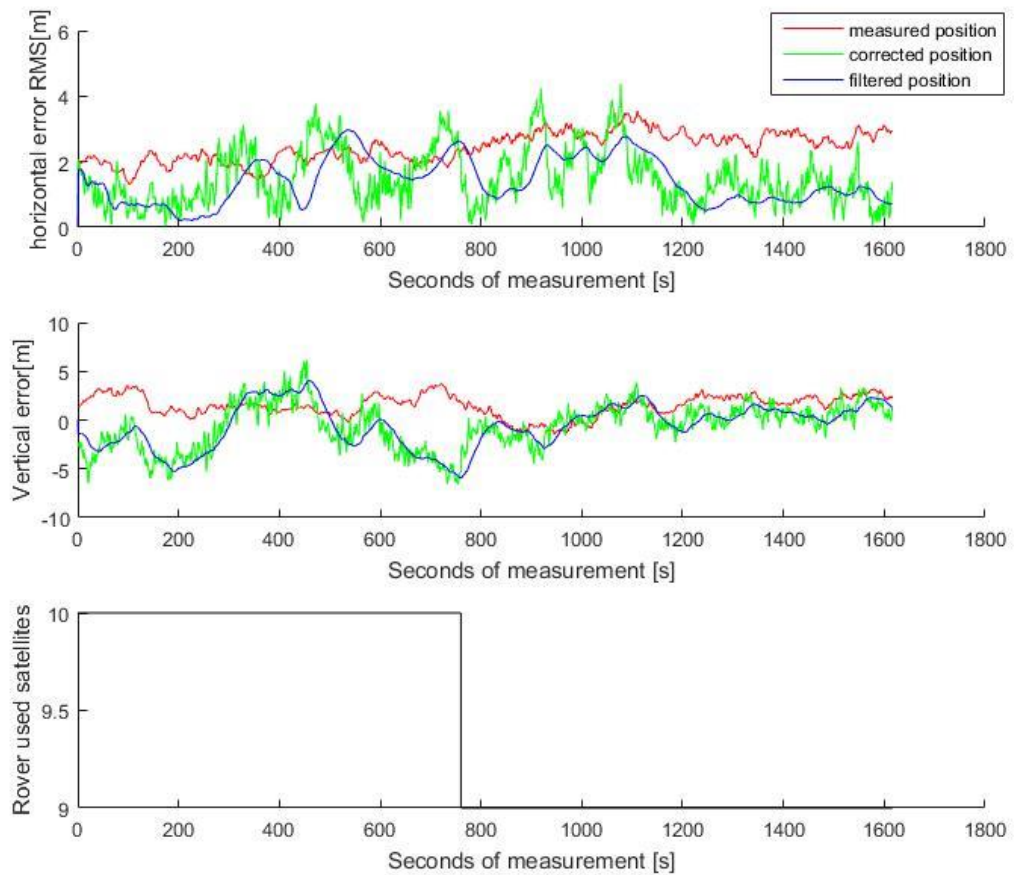


Figure 6.4. MODE1 Error comparison

6.1.2 MODE 2 (SBAS)

The SBAS static measurement was sorted out in the same conditions as the MODE1. However, it was measured at the first spot. The measurement also took approximately 45 minutes, and the results are shown in the figure bellow.

There is no such a big improvement since the original measurement was quite accurate itself. In compare with original measurement, the corrected position estimates are even slightly worse than original measurement. In the former part of the measurement, there was an error caused by unavailability of SBAS correction for the SV8, which occurs for approximately 5 minutes (figure 6.6). The time when this error was observed is also evident from the comparison figure 6.6. During this time, the correction was not generated, and it is projected as a gap in the graph. Another significant error happened when the satellite change occurred.

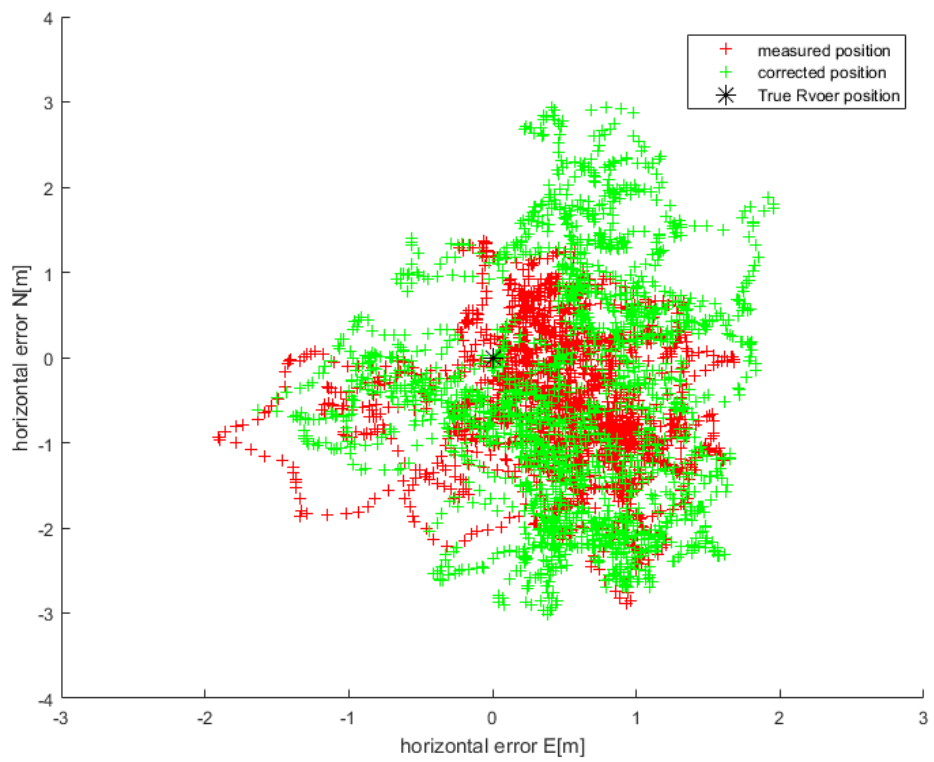


Figure 6.5. MODE2 – Static measurement

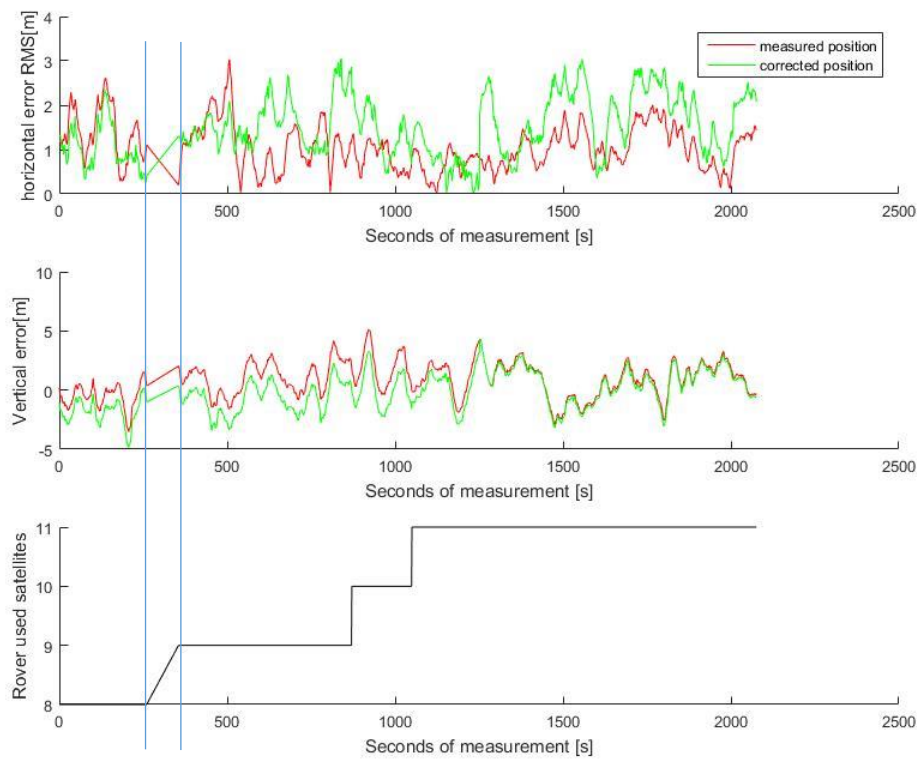


Figure 6.6. MODE2 – Error comparison

A status of the measurement was nicely recognizable from the LCD (figure 6.7)



Figure 6.7. MODE2 Error indication screenshots (occurred [Left] - resolved [right])

As can be observed from Figure 6.7, the SBAS range corrections were not significant. That was also projected to the vertical domain where the measured vertical error was slightly improved, but almost the same as an original measurement. The concrete values of accuracy are summarized in Table 11.

Table 11. Accuracy comparison MODE2

Horizontal	Original	Corrected
DRMS	1.18 m	1.65 m
CEP (50%)	0.97 m	1.34 m
Vertical		
σ_{up}	1.76 m	1.65 m

6.2 Dynamic measurements

The dynamic measurements were accomplished by walking in the defined pattern, which was delineated by reference points from Eniro maps. The pattern has the shape of the rectangle defined by the reference points in its corners.

6.2.1 MODE1 (RES)

The dynamic measurement was done during the midnight on the spot two on May 31, 2016. There was approximately 10°C, partly cloudy, and wind was calm. The measurement was done in two steps.

The first step was at a slow speed approximately 2km/h and the second about 5km/h. It can be seen in Figure 6.9 that the corrected position is closer to the ground truth, but the data are more spread around. The Kalman filter, which was used to improve this drawback, is plotted by cyan color. In the first case when speed was lower, the line is characterized by a large inertia. It is due to Kalman model, which does not take into account the actual speed of the body. In the other case, the line is conversely more smoothed, and it does not precisely follow the corrected position data when a sudden direction change occurs.

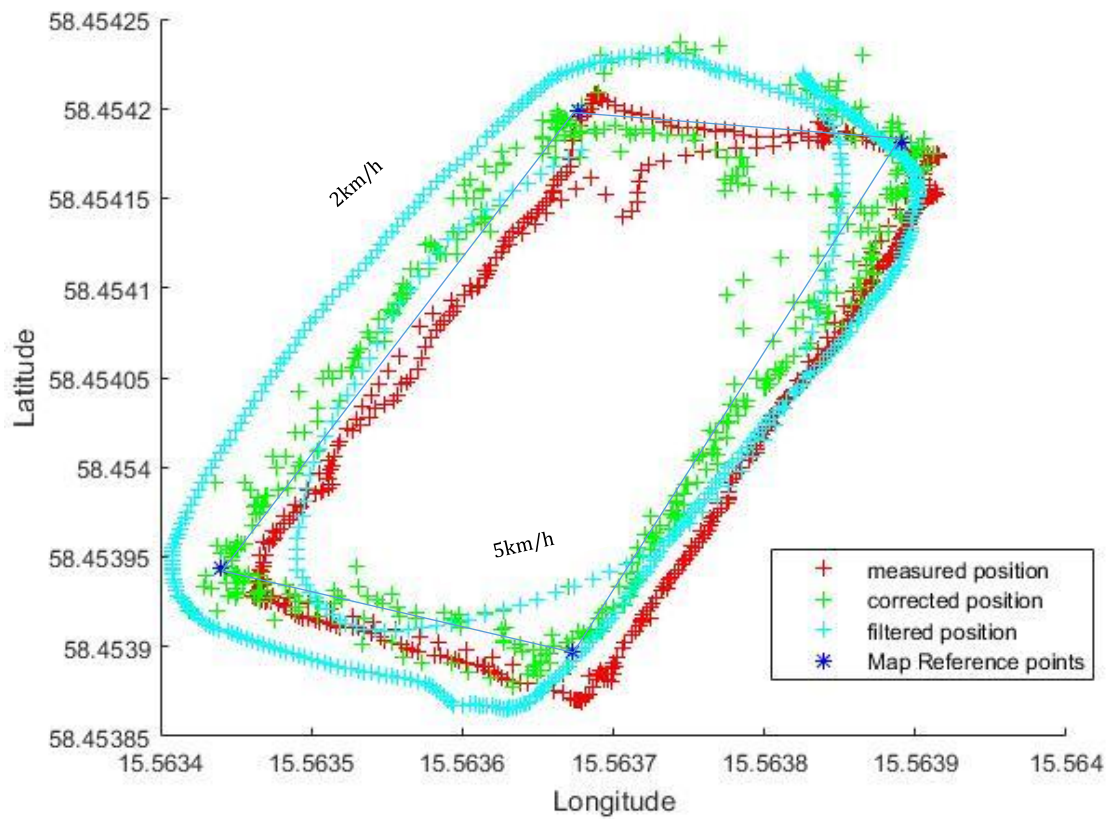
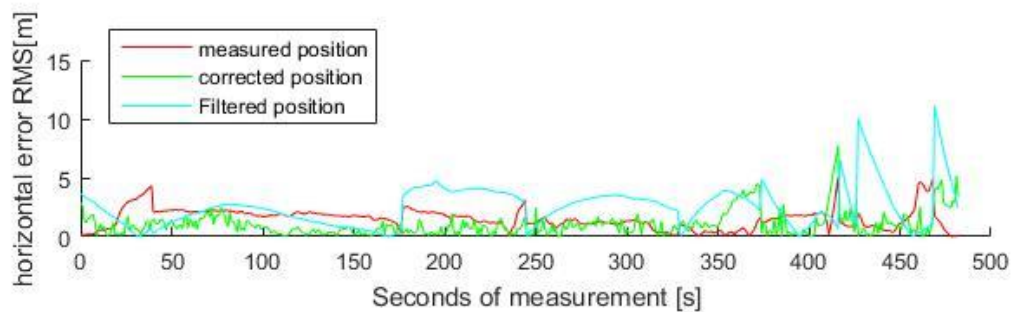


Figure 6.8. MODE1 - Dynamic measurement

The error comparison of each measurement is shown in Figure 6.9. The former part of the dynamic measurement (time 0-350s) represents the slower measurement where the filtered position is characterized by large inertia and small delay. The rest (time 350-480s) is characterized by large delay of the filtered position, which results in high inaccuracy represented by peaks, where a direction change occurs.



Horizontal	Original	Corrected	Filtered
Arithmetic Average	1.51 m	1.049 m	2.54 m
Vertical			
σ_{up}	1.49 m	2.37 m	2.14 m

Figure 6.9. MODE1 Error comparison

6.2.2 MODE2 (SBAS)

The dynamic measurement for the SBAS correction was done on spot one during late afternoon right after the static measurement. As seen from the Figure 6.10, the corrected position behaves in the proper way. It clearly follows the measurement, and because there were only small range corrections from SBAS, there is not a huge difference between measured and corrected position. However, it is apparent, that there is a small shift in the correct direction, so the accuracy in dynamical measurement was slightly improved. The measurement was carried out in the same manner as the dynamic measurement for MODE1. However, there is not a big dilution from the true position even for fast and slow measurement. It is because of application of SBAS

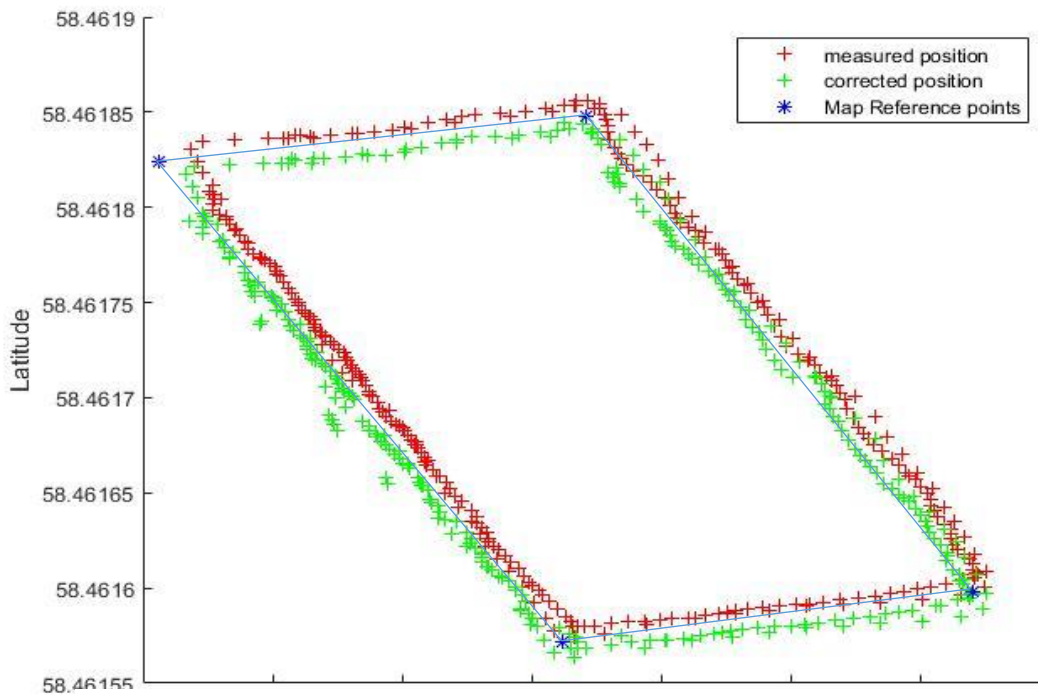


Figure 6.10. MODE2 - Dynamic measurement

correction data to original filtered receiver position. It results in a position shift, which behaves similarly as the original measurement since the SBAS corrections are not so dynamic. During the dynamic measurement, there were not any satellite data unavailability.

The error comparison for MODE2 is shown on Figure 6.11. The generated error was not so significant since there was not a big dilution during the measurement.