UNIVERSITY OF TURKISH AERONAUTICAL ASSOCIATION
INSTITUTE OF SCIENCES AND TECHNOLOGY

AN ENHANCED RTK PROTOCOL FOR L1 GPS RECEIVERS

MASTER THESIS

Cemil Baki KIYAK

Department of Electrical and Electronics Engineering

JULY 2018

UNIVERSITY OF TURKISH AERONAUTICAL ASSOCIATION
INSTITUTE OF SCIENCES AND TECHNOLOGY

AN ENHANCED RTK PROTOCOL FOR L1 GPS RECEIVERS

MASTER THESIS

Cemil Baki KIYAK

1303627003

Department of Electrical and Electronics Engineering

Thesis Supervisor: Prof. Dr. Doğan ÇALIKOĞLU

Cemil Baki KIYAK, having student number 1303627003 and enrolled in the Master Program at the Institute of Science and Technology at the University of Turkish Aeronautical Association, after meeting all of the required conditions contained in the related regulations, has successfully accomplished, in front of the jury, the presentation of the thesis prepared with the title of: An Enhanced RTK Protocol For L1 GPS Receivers.

Supervisor	: **Prof. Dr. Doğan ÇALIKOĞLU** **University of Turkish Aeronautical Association**	[illegible] ………………
Jury Members	: **Assoc. Prof. Dr. Ahmet KARAARSLAN** **Yıldırım Beyazıt University**	[illegible] ………………
	: **Asst. Prof. Hassan SHARABATY** **University of Turkish Aeronautical Association**	[illegible] ………………
	: **Prof. Dr. Doğan ÇALIKOĞLU** **University of Turkish Aeronautical Association**	[illegible] ………………

Thesis Defense Date: 05.07.2018

THE UNIVERSITY OF TURKISH AERONAUTICAL ASSOCIATION
INSTITUTE OF SCIENCE AND TECHNOLOGY

I hereby declare that all the information in this study I presented as my Master's Thesis, called "An Enhanced RTK Protocol For L1 GPS Receivers" has been presented in accordance with the academic rules and ethical conduct. I also declare and certify on my honor that I have fully cited and referenced all the sources I made use of in this present study.

05/07/2018

Cemil Baki KIYAK

[illegible]

ACKNOWLEDGEMENTS

I would like specially to thank to my advisor Prof. Dr. Doğan ÇALIKOĞLU for his valuable advises, helps, directives, encouragments and suggestions. I would also like to thank to my family Prof. Dr. Tülin DOĞANAY, Prof. Dr. Suat KIYAK, Ahmet Şemsi KIYAK, Yaprak ÖRNEK KIYAK, Cahit Emir KIYAK and my friends Gizem Büke ÖZTÜRK, Şükrü GÜRSES, Burcu DEĞİRMENCİOĞLU, Ömer Meriç BEŞLER, Onur KILIÇ, Basri YAŞAR and Suzan YAŞAR who encouraged me and helped preparing the thesis documents as well as the researches.

July 2018 Cemil Baki KIYAK

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

GPS	: Global Positioning System
UART	: Universal Asynchronous Receiver Transmitter
IC	: Integrated Circuit
RTK	: Real Time Kinematic
RTCM	: Radio Technical Commission for Maritime Services
PPP	: Precise Point Positioning
GNSS	: Global Navigation Satellite System
D-GPS	: Differantial GPS
S-GPS	: Simultaneous GPS
A-GPS	: Assisted GPS
WAAS	: Wide Area Augmentation System
EGNOS	: European Geostationary Navigation Overlay Service
SBAS	: Satellite-Based Augmentation Systems
MSAS	: Multi-functional Satellite Augmentation System
GSM	: Global System for Mobile communications
RF	: Radio Frequency
NMEA	: National Marine Electronics Association
GPGGA	: Global Positioning System Fix Data
RX Pin	: Serial UART Receiveing Pin
TX Pin	: Serial UART Transmitting Pin
uC	: Microcontroller
I/O Pin	: Input/Output Pin
AT Commands	: Attention Commands for Operations
FTDI	: Future Technology Devices International
TTL	: Transistor-to-Transistor Logic
PC	: Personal Computer
IDE	: Integrated Development Environment
Lat	: Latitude
Lon	: Longitude
Alt	: Altitude
CR	: Carriage Return
LF	: Line Feed

ABSTRACT

AN ENHANCED RTK PROTOCOL FOR L1 GPS RECEIVERS

KIYAK, Cemil Baki
Master Thesis, Department of Electrical and Electronics Engineering
Thesis Supervisor: Prof. Dr. Doğan ÇALIKOĞLU
July 2018, 71 page

There are three known bands around the world that GPS receivers analyse. They are named as L1, L2 and L5. L1 and L2 are for commercial use while L5 is for military use. Widely available cheap L1 receivers in the market produces a coordinate result with an error of 1 to 20 meters because of weather conditions and the other similar effects. In mapping sector and its applications, where precise and accurate computation is needed, much more expensive L1 L2 band receivers are used. These receivers are named as D-GPS. D-GPS receivers compute its coordinate point with an accuracy of up to 30 cms by waiting on a point for a long time. In cases where more precise measurement is needed, a fixed reference D-GPS station is installed on the land that to be measured for multiple measurements, so the instantaneous measurement error in the current area is determined. This error as direction and distance information, is sent with a wireless data link module to the other (traveling) D-GPS that measures the field. The protocol for this verification information is a complex binary protocol called RTCM, implemented for expensive L1 L2 D-GPS receivers. Having acquired this verification information in RTCM format, the mobile device corrects its own error with this information. This verification process is called RTK (Real Time Kinematics). With this process, the accuracy of about 30 cm is raised to 3 cm. The purpose of this study is to develop an alternative protocol that is simpler and more understandable than the RTCM format that can only be applied to the much cheaper L1 GPS devices, which is the RTK method for very expensive L1 L2 D-GPS devices, to reduce the measurement error to 20 ~ 50 cm levels. In order to code the necessary protocols and perform experiments, a base GPS equipment design will be done. As a result of the study, it is shown that, a device and a method that can work at low cost in the cartography sector can be produced by developing a system that can obtain a position with medium sensitivity with cheap GPS receivers.

Key Words: GPS, L1, L2, RTK, RTCM, D-GPS, RF Modem

ÖZET

L1 BANDI GPS ALICILARI İÇİN GELİŞMİŞ BİR RTK PROTOKOLÜ

KIYAK, Cemil Baki
Yüksek Lisans Tezi, Elektrik Elektronik Mühendisliği Bölümü
Tez Danışmanı: Prof. Dr. Doğan ÇALIKOĞLU
Temmuz 2018, 71 sayfa

Dünyada GPS alıcılarının çözümlediği bilinen üç frekans vardır. Bunlar L1, L2 ve L5'tir. L1 ve L2 ticari amaçlı olup L5 askeri amaçlar için kullanılmaktadır. Piyasada bulunan ucuz L1 alıcıları, hava koşulları ve benzer etkenler dolayısıyla 1 metre ile 20 metre arası hata ile sonuç vermektedir. Haritacılık sektörü uygulamalarında ise hassas ölçümlemeye ihtiyaç duyulan alanlarda çok daha pahalı olan L1 L2 alıcıları kullanılmaktadır. Bu alıcılara D-GPS adı verilir. Bu alıcılar bir noktada uzun süre bekletilerek gerekli filtrelemeler sonucu 30 cm hassasiyete kadar sonuç verebilmektedir. Daha hassas ölçüme ihtiyaç duyulan durumlarda ise, birden çok ölçümlemenin yapılacağı araziye bir adet sabit referans D-GPS istasyonu kurularak, mevcut bölgedeki anlık ölçüm hatası tespit edilir. Bu hata, yön ve mesafe bilgisi olarak arazide ölçüm yapan diğer (gezici) D-GPS cihazına kablosuz bir data link modülüyle gönderilir. Bu doğrulama bilgisinin protokolü, pahalı L1 L2 D-GPS alıcıları için uygulanan RTCM adlı binary formatlı karmaşık bir protokoldür. RTCM formatındaki bu doğrulama bilgisini edinen gezici cihaz, kendi hatasını bu bilgi ile düzeltir. Bu doğrulama işlemine RTK (Real Time Kinematics) adı verilir. Bu işlem ile 30 cm civarındaki doğruluk 3 cm'ye kadar yükseltilir. Bu çalışmada amaç, sadece çok pahalı L1 L2 D-GPS cihazları için geçerli olan RTK yönteminin çok daha ucuz olan L1 GPS cihazları için de uygulanabileceği ve RTCM formatından daha basit ve anlaşılır bir formata sahip alternatif bir protokol geliştirerek, çok hassas olmasa da ucuz GPS cihazlarının ölçüm hatasını 20~50 cm seviyelerine indirmektir. Gerekli protokolleri yazmak ve deneyleri icra etmek için bir baz GPS donanımı tasarımı yapılacaktır. Çalışma sonucu görülmüştür ki, ucuz GPS alıcıları ile orta hassasiyette konum elde edebilecek bir sistem geliştirerek haritacılık sektöründe düşük maliyetlerle çalışma yapabilecek bir cihaz ve yöntem üretilebilmektedir.

Anahtar Kelimeler: GPS, L1, L2, RTK, RTCM, D-GPS, RF Modem

CHAPTER ONE

INTRODUCTION

1.2 Presentation of the Work

In this study, two circuits were prepared for two GPS receiver modules. Two transceiver modules have been implemented in both modules. At this point, two modules can communicate with each other. The first of these modules is considered as a fixed reference GPS station. The other is set as a rover GPS device which communicates with the first one to verify itself in order to find the accurate coordinate of itself. Fixed GPS fixes its own coordinate precisely by making continuous measurements and taking the average of these measurements. Then, by subtracting the deviation in the instantaneous readings from the computed fixed position, the instantaneous error in that region is obtained. Fixed GPS sends this error to rover GPS. Rover GPS improves its measurement error by adding error information from fixed GPS to its own position. The reason for this exercise is the lack of an existing device that keeps the price / efficiency ratio on the market at an optimal level because the sensitive GPS devices on the market are very expensive and the cheap GPS devices have high flaws. By using this new RTK protocol and methodology obtained from this study, it will be possible to obtain cheaper and more accurate verification by making any L1 band GPS device with RTK.

1.2 RTK Protocol as the Main Component

RTK (Real Time Kinematics) is the name of a procedure that is used to refine a non-accurate GPS data in order to obtain more accurate coordinate readings. It simply carries the correction data for coordinates as distances on axes of up-down, east-west and north-south. With these data, D-GPS devices corrects the coordinate data of their positions. The procedure has a specific protocol called RTCM (Radio Technical Commission for Maritime Services) that has many versions. These protocols are binary protocols with many parameters as they have many complex structures which serves the same simple idea of carrying the position errors in three axes as distance values.

RTK procedure has two sides in the process. First is its producing stage with a fixed station and the second is the applied stage of a rover device. In producing process, a fixed reference station is installed on the land to be surveyed and starts gathering continuous measurements. It collects the measurements and takes their average value in order to find its actual coordinate. After finding its actual coordinate, reference station will be able to find the error between the actual point and the currently read point. Then sends this error to the rover device, so that the rover device will be able to correct its point relatively as the error is time and location dependent.

1.3 Comprising Class

It is seen in the literature review that; the studies were done chronologically on one of the two D-GPS devices as a reference station and sending RTK data wirelessly to the other. Afterwards, academic studies were carried out on the issue of data communication at longer distances by connecting GSM modules to these devices. Later, a network consisting of fixed stations was established, where a common area was constantly monitored for errors and establishment of a validation network is studied. In this way, any rover D-GPS device can connect to these networks and get confirmation via internet. Later on, instead of establishing physical stations, studies were made on the establishment of virtual fixed stations with various interpolation methods taking into consideration the distributions of these stations on the map. As a result, it is noticed that developing a simple format RTK protocol for cheap single channel (L1) GPS devices is a new and important issue.

1.4 The Main Aspects of the Work and its Importance

Two GPS receivers were used in this study. Two Bluetooth modules and one wireless data link module are connected to both of these receivers. The first circuit is assigned as the reference fixed station. The task of this circuit is to constantly read in the region where it is established and to filter out the results, to find out every instant reading error and to publish this error. The second circuit is assigned as a rover device. The task of this circuit is to minimize the error by applying the relative error value from the reference fixed station to its own instantaneous readings. In this view, the second circuit can find its position at low assignment with the help of the first circut. With this method, a new RTK protocol for low cost L1 band GPS receivers has been achieved.

In chapter two, literature studies were carried out to investigate similar studies and differences.

In the third and fourth chapter; communication devices with GPS modules, communication with smart phones via bluetooth, communication between two devices via wireless data link, codes, positioning algorithms, process stages, filtering and verification data are explained.

In the fifth and sixth chapter, the differences between the raw data and the processed data are compared. Error minimization has been numerically tested and demonstrated. The needs of the market have been analyzed by comparing their advantages in terms of cost and portability. Ease of use and comprehensibility are mentioned in the market.

CHAPTER TWO

LITERATURE REWIEV

It is seen in the literature review that; the studies were done generally on two D-GPS devices. One is set as a reference station and sending RTK data wirelessly to the other.

"*RTK has represented the peak of GPS performance for several years, but there have been severe limitations - with fewer than 5 satellites in view RTK does not work at all, or works so slowly as to be almost no better than DGPS in many applications. Now, by combining GPS and GLONASS in an RTK product, you can do RTK, for the first time ever, in places such as open pit mines, urban canyon, river valleys, etc., where GPS-only RTK simply will not work.*" [Diggelen, 1997].

"*The paper introduces a new approach to determine azimuth and elevation dependent phase center biases through a field measurement in an absolute sense. It takes special care of the multipath effects. The model, the conditions for the field procedure and preliminary analysis of results are presented. The absolute antenna phase center calibration procedure is implemented in the GPS processing package GEONAP.*" [Wübbena, et.al., 1, 2014].

Furthermore, there are many studies about data communication at longer distances by connecting GSM modules to these D-GPS devices.

"*The latest evolution in digital wireless technology, third-generation (3G) Code Division Multiple Access/Single Carrier (CDMA2000/1X) wireless network, is applicable for transmitting real-time kinematic (RTK) GPS correction messages. Fast and reliable, publicly available wireless networks, combined with highly accessible Internet connectivity, allows the multicasting of messages to mobile users, who are no longer restricted to traditional private UHF wireless networks. Nationwide public wireless network systems continue to expand and can provide an inexpensive infrastructure for the emerging multi-reference network system.*

Transmission performance via the Internet-based CDMA2000/1X outperforms UHF technology in transmission throughput and latency, as well as in the RTK initialization time and positional accuracy." [Lui, 2004].

"*The Internet as a basis for Real-Time Kinematic (RTK) and differential Global Positioning System (DGPS) service provides many advantages for worldwide GPS users. Among these advantages are service unification, open architecture, bidirectional communication, and scalability. The current development of this service allows users to use RTK and DGPS through the Internet with conventional accuracy over the short and medium baselines. The perspective for this service lies in the field of wide-area augmentation systems (WASS). At this stage of the Internet-based RTK and DGPS service project, the general concept, system components, draft standards, and software are developed.*" [Hada, 2000].

Some developments are accomplished on a network consisting of fixed stations, where a common area was constantly monitored for errors and establishment of a validation network. So, any moving D-GPS device could connect to these networks and get confirmation via internet.

"*This paper describes the design, operation and testing of a RTK GPS system based on the use of a multi- reference station approach. The use of a multi- reference station network, as opposed to a single reference station, results in a larger service area coverage, a lower number of reference stations, increased robustness, and a higher positioning accuracy.*" [Lachapelle, et.al., 2000].

"*The accuracy of today's RTK is limited by the distance dependent errors from orbit, ionosphere and troposphere as well as station dependent influences like multipath and antenna phase center variations. The basic idea of Geo++ ® GNSMART (GNSS − State Monitoring And Representation Technique) is to analyze the data from a reference station network to estimate and represent the state of individual components of the GPS error budget in real−time. All stations of a network are processed simultaneously for best estimation of global parameters and to increase the reliability of the results. The complete state can normally not be used by the rover directly. Therefore GNSMART can derive several types of representations from the complete state model, adequate for special transmission or rover requirements to reduce the GNSS error budget significantly. The implementation was operable before the current solar activity maximum, and is* *currently installed on many reference stations around the world under different ionospheric conditions. Recent results show the capabilities of GNSMART. Horizontal accuracy of 1 centimeter can be achieved with initialization times of 30 seconds, often even within 10 seconds over distances of more than 30 kilometers.*" [Wübbena, et.al., 2, 2001].

Then it is noticed that, some studies were carried out to establish virtual reference stations based on physical D-GPS reference stations in order to minimize costs and mixmize corrections in a correction network.

"*Virtual Reference Stations lead to substantial improvements for real-time positioning by reducing atmospheric, orbital and multipath errors resulting in a performance for long baseline as experienced from short baselines using only one reference station.*" [Vollath, et.al., 2000].

As a result, it is obvious that developing a simple and understandable RTK protocol for cheap single channel (L1) GPS devices is a new and important issue.

CHAPTER THREE

THE BASIC STRUCTURE OF THE PROTOTYPE

3.1 Basic Working Principle

The scenario of the L1 band GPS RTK system is set to be a verification system with two devices. One is set to be a reference station and the other is the measuring rover device. The reference station's mission is to gather GPS coordinate data by continuous readings and collect them with a filter that calcutes the average over time. Its coefficient for past data should be more than 50% while the coefficient for the current measurements should be lesser. By this method, the reference station will find its own coordinate more accurate over time.

The rover device's mission is to make fast measurements by making GPS measurement readings and correct its readings by the reference station's correction broadcast. After the rover device corrects itself with the broadcasted data, the result will be filtered again with a coefficient less than %50 for its past coordinate data as it will not be a fixed device. By this method, the rover device will reduce its positioning error with the help of the reference station. The system's demonstration is shown below in Fig. 3.1.

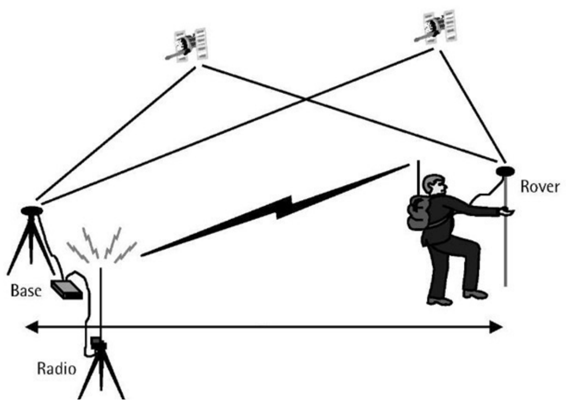


Fig. 3.1: RTK System Demonstration.

3.2 Circuit Design

Using Bluetooth module (M1) to communicate with an Android smart phone, it has to be connected to a UART port with RX and TX pins. The selected microcontroller, Arduino Pro Mini (U1), has only one hardware UART port. So, in software design, a software UART port has to be defined on any I/O pin for Bluetooth's communication with uC. The GPS IC (M2), uBlox - NEO6MV2, has its communication via UART again. As it is the main component of the circuit, it is assigned as the primary module. So, it is connected to Arduino's hardware UART in order not to miss any data in communication as the valuable informations of the system are primarily the GPS readings. After connecting them, for RF Data Link (M3), RFD900+, another software UART will be defined on another couple of I/O pins again to provide communication between rover device and reference station.

Fig.3.2. shows the connections of the base circuit design as both the rover device and the reference station will have the same setup. Because both devices will need to communicate with each other via an RF Link, with smart phones to observe the parameters and with GPS satellites; the circuit setups remains same.

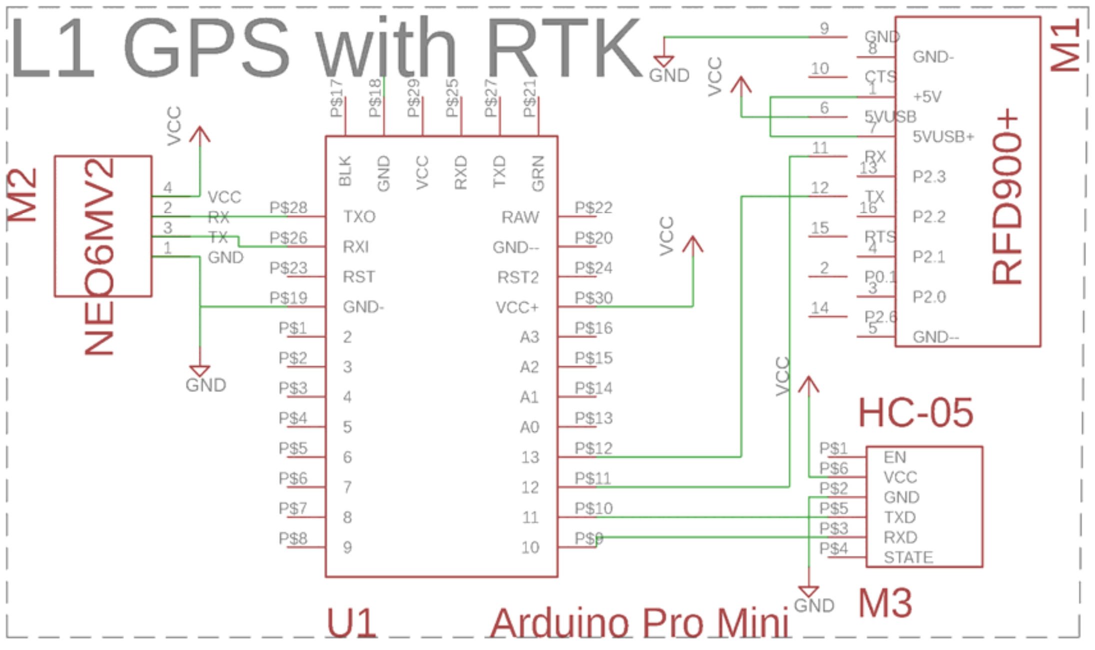


Fig. 3.2: RTK System Circuit Design for both Rover Device and Reference Station.

3.3 Algorithms

Fig.3.3. shows the algorithms and working principle for both RTK rover device and RTK reference device. The reference device calculates the difference between the measured value and the actual value to broadcast it as a correction value. The rover device measures the GPS coordinate and corrects itself with the reference station's broadcasted correction value.

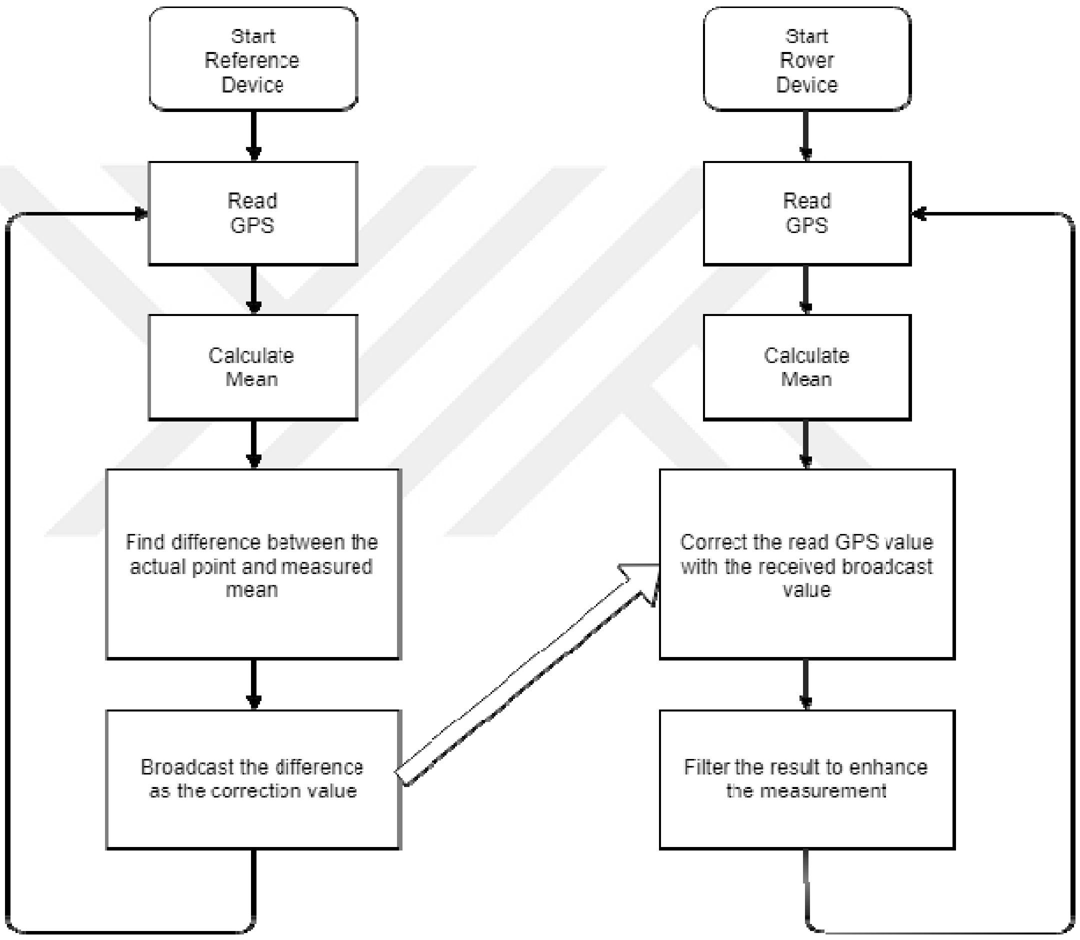

Start Reference Device
Read GPS
Calculate Mean
Find difference between the actual point and measured mean
Broadcast the difference as the correction value
Start Rover Device
Read GPS
Calculate Mean
Correct the read GPS value with the received broadcast value
Filter the result to enhance the measurement

Fig. 3.3: Block Diagrams of RTK Devices' Algorithms.

CHAPTER FOUR

DESIGNING THE ALGORITHMS AND ELECTRONICS

4.1 Component Selection

For the desired base circuit, there should be an L1 band GPS IC, a Bluetooth module to communicate with the smart phone as the observing device, a wireless data link and a microcontroller unit to execute all the needed algorithms and unite all the modules.

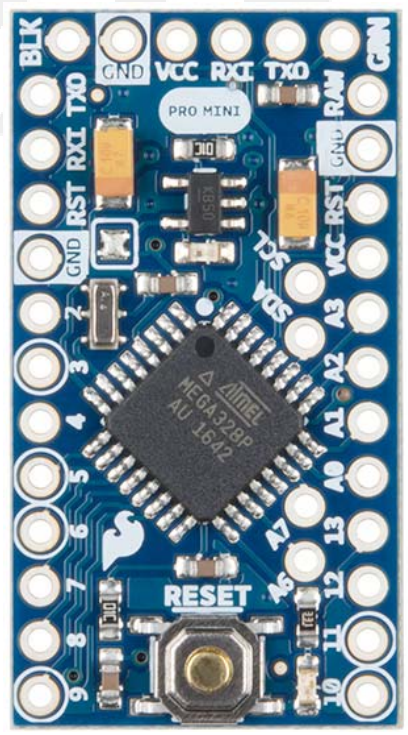


Fig. 4.1: Arduino Pro Mini as Microcontroller.

4.1.1 Chosing the L1 GPS Module

There are some known brands for L1 GPS ICs. Some of them are Sirfstar, Globalsat, Trimble and uBlox. The technical spesifications and market price will be compared in this section. The best price/performance value can be seen on uBlox module. So, it is selected for the project as the receiver.

4.1.1.1 Sirfstar - 20 Channel EM-408 SiRF III Receiver

Channels: 20
Sensitivity: -159dBm
Position accuracy: 5m to 10m
Hot Start Time: 8s
Warm Start Time: 38s
Cold Start Time: 42s
Power Consumption: 75mA at 3.3V
Weight: 20 grams
Price: 124,43 $

4.1.1.2 Globalsat - GPS receiver - EM-506 (48 channel)

Channels: 48
Sensitivity: -163dBm
Position accuracy: 2.5m
Hot Start Time: 1s
Warm Start Time: 15s to 35s
Cold Start Time: 15s to 35s
Power Consumption: 45-55mA at 4.5-6.5V
Weight: 16 grams
Price: 51,94 $

4.1.1.3 Trimble - copernicus II DIP (12 channel)

Channels: 12

Sensitivity: -148dBm

Position accuracy: 2m to 5m

Hot Start Time: 3s

Warm Start Time: 35s

Cold Start Time: 38s

Power Consumption: 44 mA at 3.0 V

Weight: - (no info as a module)

Price: 111,39 $

4.1.1.4 uBlox - GY-NEO6MV2

Channels: **50**

Sensitivity: -160dBm

Position accuracy: **1m to 2.5m**

Hot Start Time: **1s**

Warm Start Time: 27s

Cold Start Time: **27s**

Power Consumption: **44 mA at 3.0 V**

Weight: - (no info as a module)

Price: **14,15** $

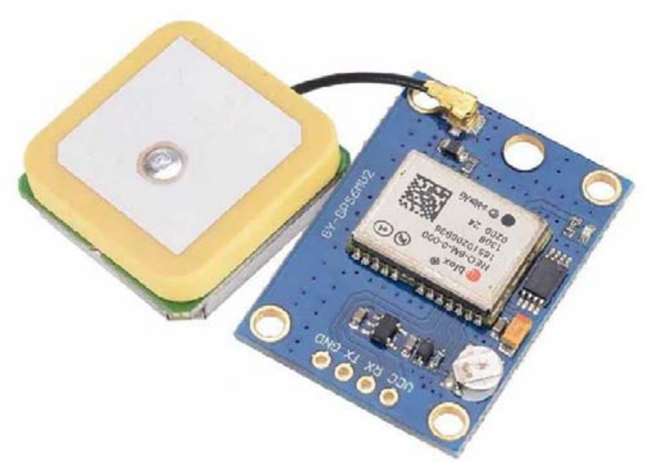


Fig. 4.2: uBlox - GY-NEO6MV2

4.1.2 Chosing the Bluetooth Module

There are not so much alternatives on Bluetooth on the market. The only known model is called HC-05 and is to be sold as a module with an easy to use interface with an UART protocol and ready to use structure. So, it will be chosen as the device’s communication module for smart phones. Its price is 5.9 $s.

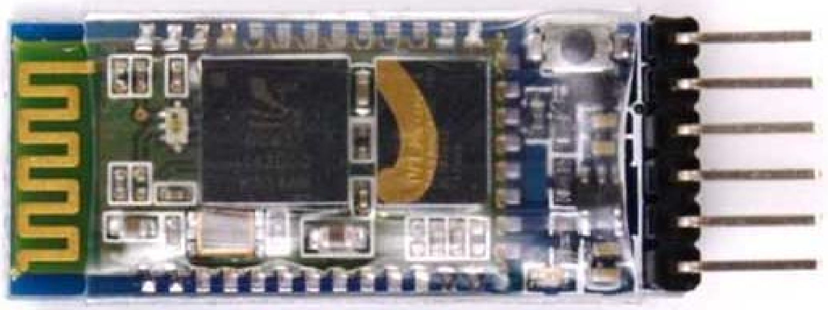


Fig. 4.3: HC-05 Bluetooth Module.

4.1.3 Chosing the Wireless (RF) Data Link Module

There are many lesser known companies produces RF Data Link modules. The market leading company in the sector is the brand named Digi and its product groups name is called XBee. I have chosen RFD900+ module of an Australian company named RFDesign because a couple of modules were available in my workshop. As they are the most expensive parts of the projects, I chose to use them in my thesis project. In order to compare its spesifications, Digi’s equivalent XBee module will be used in the following section.

4.1.3.1 Digi - XBee-PRO 900HP

Frequency Band 902 To 928 Mhz,
Rf Data Rate 10 Kbps Or 200 Kbps
Indoor/Urban: 10 Kbps: Up To 610 M; 200 Kbps Up To 305 M
Outdoor/Line-Of-Sight: 10 Kbps Up To 15.5 Km; 200 Kbps Up To 6.5 Km
Transmit Power Up To 24 Dbm (250 Mw)
Receiver Sensitivity -101 Dbm @ 200 Kbps, -110 Dbm @ 10 Kbps
Price: 101,48 $

4.1.3.2 RFDesign – RFD 900+ Modem

Long range >**40km** depending on antennas and GCS setup

1 Watt (**+30dBm**) transmit power.

Power: +5v, ~800mA max peak (at maximum transmit power)

Frequency Range: 902 - 928 MHz (USA) / 915 - 928 MHz (Australia)

UART data rates: 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps

Receive Sensitivity: >**121 dBm** at low data rates

Weight: 14.5g

Price: **75,3 $**

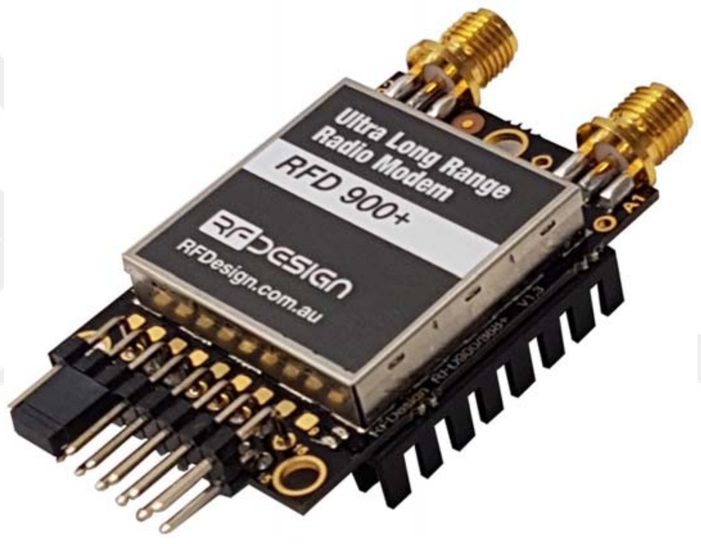


Fig. 4.4: RFDesign – RFD 900+ Modem.

4.2 Circuit Setup

The circuit design is mentioned on section 3.2 Circuit Design. In this section the setup will be menitoned.

Each electronic board contains a 5V Li-ion battery, uBlox GPS module, HC-05 Bluetooth Module, Arduino pro mini, and RF Design Ultra Long Range Radio Modem.

- Arduino pro mini: Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- uBlox GPS module: uBlox module is a positioning module, the position is obtained by GPS.

- HC-05 Bluetooth Module: The HC-05 module is a hardware component that provides a wireless connection via Bluetooth protocol.
- RF Design Ultra Long Range Modem: The Ultra Long Range Modem is a hardware component that provides an ultra long range wireless connection.

Each electronic board is connected to a breadboard. The Arduino is the main controller on the breadboard. It has VCC, GND, 2 RX, 2 TX and IO pins. The 5V battery supplies power to Arduino, uBlox, HC-05, and RF Design Modem via VCC and GND pins. uBlox component has VCC, GND, RX, and TX pins. It is connected to Arduino via the serial connection on RXI and TXO ports and it is connected to the 5V battery from VCC and GND pins. uBlox GPS component outputs GPGGA sentence for obtaining the position from the uBlox component. Arduino reads the GPGGA string and parses the required information from the string which are latitude, longitude, altitude, HDOP value, GPS time in seconds and the number of satellite connection. The HC-05 component has VCC, GND, RX, and TX pins. It is connected to Arduino via the serial connection on pins 10 and 11. It is connected to the 5V battery on VCC and GND pins. The RF Design modem has VCC, GND, RX, and TX pins. It is connected to Arduino via the serial connection on 12 and 13 ports and It is connected to the 5V battery on VCC and GND pins.

4.3 Embedded Software Design

As mentioned on Section 3.3 Algorithms, there are two algorithms for each device.

4.3.1 Common Processes

There are several common processes in both devices. Both the devices communicates with uBlox GPS receiver and parses data from the given sentence string which is called GPGGA. Both of them communicates with a Bluetooth module which is called HC-05 and with an RF module called RFD900+ which need to be configured for the desired communication process. Another needed process is to build a software serial communication port as the Arduino Pro Mini microcontroller has only one hardware serial port while the circuit setup needs three serial ports for each GPS module, Bluetooth module and RF transceiver.

4.3.1.1 Parsing GPGGA sentence

GPGGA string is a universal common data format for all GPS devices. It carries the needed coordinate information and other needed peripherals.

The embedded software is designed to parse the latitude, longitude, altitude, time, satellite and horizontal dilution of precision values. The code is built to be a letter processor which splits the string to its components between commas that is used to seperate the needed values. After splitting the needed values, they are assigned to the corresponding variables as integers or decimal values in order to make them ready to process for the needed calculations. In this code, an arduino compatible, ready to use library called TinyGPS+ is downloaded and used for communicating the GPS module. The example sentence and explanations are made below on Table 4.1.

Example: $GPGGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh

Table 4.1: GPGGA String Data Format and Definitions.

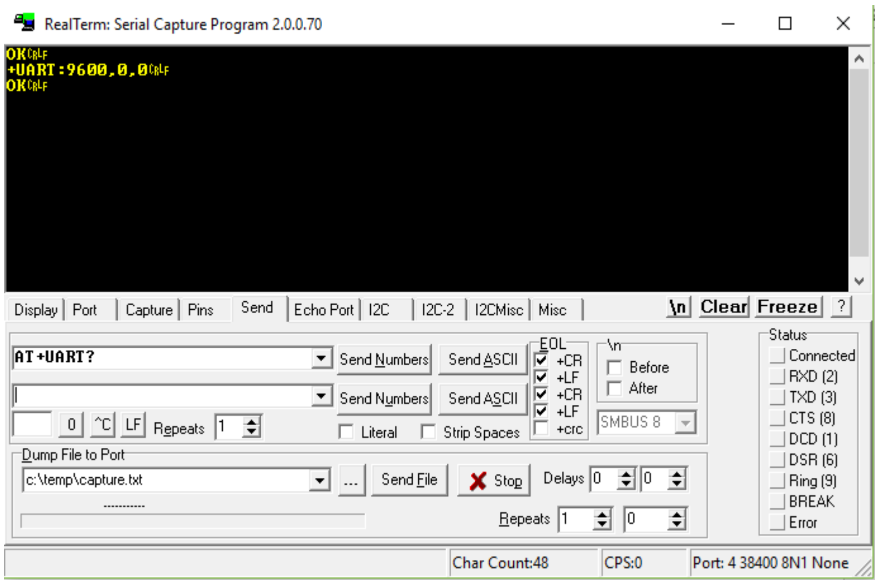
Name	Example Data	Description
Sentence Identifier	$GPGGA	Global Positioning System Fix Data
Time	170834	17:08:34 Z
Latitude	4124.8963, N	41d 24.8963' N or 41d 24' 54" N
Longitude	08151.6838, W	81d 51.6838' W or 81d 51' 41" W
Fix Quality: - 0 = Invalid - 1 = GPS fix - 2 = DGPS fix	1	Data is from a GPS fix
Number of Satellites	05	5 Satellites are in view
Horizontal Dilution of Precision (HDOP)	1.5	Relative accuracy of horizontal position
Altitude	280.2, M	280.2 meters above mean sea level
Height of geoid above WGS84 ellipsoid	-34.0, M	-34.0 meters
Time since last DGPS update	blank	No last update
DGPS reference station id	blank	No station id
Checksum	*75	Used by program to check for transmission errors

4.3.1.2 Configuring bluetooth module

The Bluetooth module is prepared to communicate with a baud rate of 9600 bps with no parity bit and 1 stop bit. Its speed has to be set low as the software serial code can’t be built with an automatic buffer mechanism as there is no hardware background to sense any incoming character. So, any incoming bit of a serial data has to be caught by the software in one loop period. This is the reason why the communication speed has to be set low. 9600 bps baud rate means 9600 bits/second which allows main loop period to be 1/9600=1.042 ms.

In order to set the baud rate and Bluetooth module’s broadcast name, module has to be powered up while the button on the module is hold pressed. This is the process to start the HC-05 module in configuration mode. After starting the module in configuration mode, AT commands are used to set up the needed configurations for the module. For sending the AT commands to the module, it is wired to an FTDI-USB to TTL Serial converter module which is connected to the PC. For communicating serially with the module, the software named Realterm is used on PC side.

To change the baud rate, AT+BAUD=9600,0,0 command is sent to the module. To change the module’s broadcast name, AT+NAME=RTKReference command is sent for the reference station while RTKRover is set for the rover device. The Fig.4.5. shows how to configure HC-05 Bluetooth module.


RealTerm: Serial Capture Program 2.0.0.70

```
OKCRLF
+UART:9600,0,0CRLF
OKCRLF
```

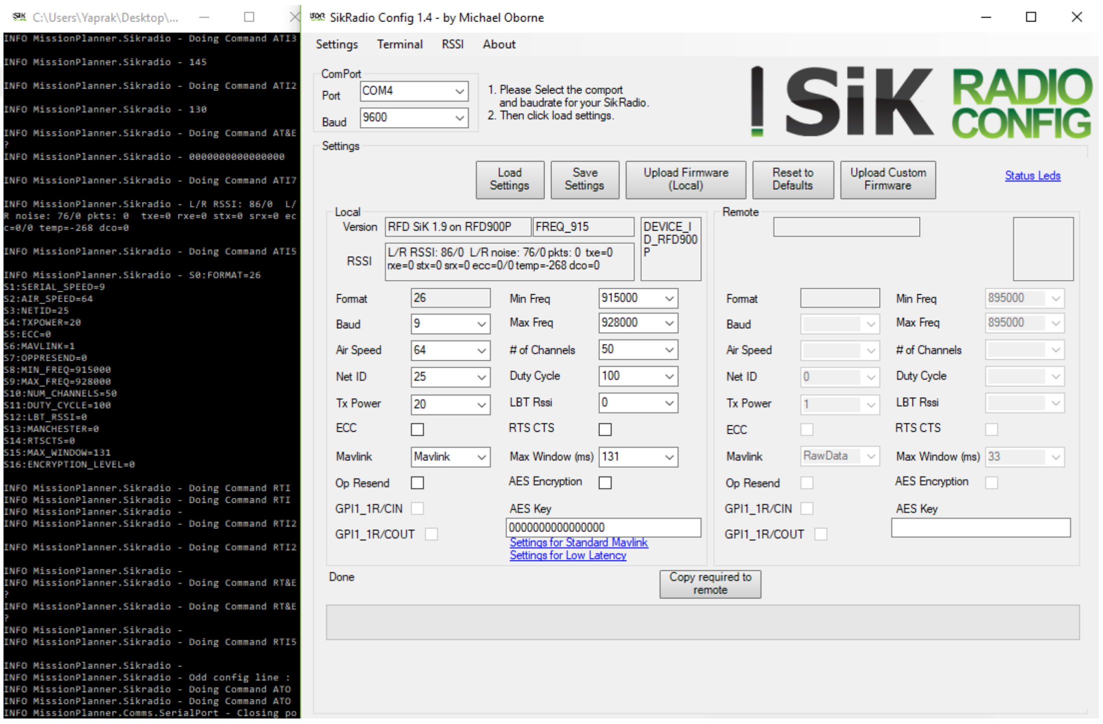
AT+UART?

Char Count:48 CPS:0 Port: 4 38400 8N1 None

Fig. 4.5: Configuring HC-05 Bluetooth Module with Realterm.

4.3.1.3 Configuring RF module

RF module is set to communicate with a baud rate of 9600 bps again as RF module is connected to a non-hardware serial pin and assigned as a software serial port. The software named "3DR Radio Config" is used to communicate with the RFD900+ module. It enters configuration mode by sending a "+++" string as a command. Then it uses AT commands to configure the peripherals. 3DR Radio Config software does the required commands automatically for wirelessly paired modules. The Fig.4.6. below shows the configuration software.



```
INFO MissionPlanner.Sikradio - Doing Command ATI3

INFO MissionPlanner.Sikradio - 145

INFO MissionPlanner.Sikradio - Doing Command ATI2

INFO MissionPlanner.Sikradio - 130

INFO MissionPlanner.Sikradio - Doing Command AT&E
?
INFO MissionPlanner.Sikradio - 0000000000000000

INFO MissionPlanner.Sikradio - Doing Command ATI7

INFO MissionPlanner.Sikradio - L/R RSSI: 86/0  L/
R noise: 76/0 pkts: 0  txe=0 rxe=0 stx=0 srx=0 ec
c=0/0 temp=-268 dco=0

INFO MissionPlanner.Sikradio - Doing Command ATI5

INFO MissionPlanner.Sikradio - S0:FORMAT=26
S1:SERIAL_SPEED=9
S2:AIR_SPEED=64
S3:NETID=25
S4:TXPOWER=20
S5:ECC=0
S6:MAVLINK=1
S7:OPPRESEND=0
S8:MIN_FREQ=915000
S9:MAX_FREQ=928000
S10:NUM_CHANNELS=50
S11:DUTY_CYCLE=100
S12:LBT_RSSI=0
S13:MANCHESTER=0
S14:RTSCTS=0
S15:MAX_WINDOW=131
S16:ENCRYPTION_LEVEL=0

INFO MissionPlanner.Sikradio - Doing Command RTI
INFO MissionPlanner.Sikradio - Doing Command RTI
INFO MissionPlanner.Sikradio -
INFO MissionPlanner.Sikradio - Doing Command RTI2

INFO MissionPlanner.Sikradio - Doing Command RTI2

INFO MissionPlanner.Sikradio -
INFO MissionPlanner.Sikradio - Doing Command RT&E
?
INFO MissionPlanner.Sikradio - Doing Command RT&E
?
INFO MissionPlanner.Sikradio -
INFO MissionPlanner.Sikradio - Doing Command RTI5

INFO MissionPlanner.Sikradio -
INFO MissionPlanner.Sikradio - Odd config line :
INFO MissionPlanner.Sikradio - Doing Command ATO
INFO MissionPlanner.Sikradio - Doing Command ATO
INFO MissionPlanner.Comms.SerialPort - Closing po
```

ComPort
Port: COM4
Baud: 9600

1. Please Select the comport and baudrate for your SikRadio.
2. Then click load settings.

Local
Version: RFD SiK 1.9 on RFD900P | FREQ_915 | DEVICE_ID_RFD900P
RSSI: L/R RSSI: 86/0 L/R noise: 76/0 pkts: 0 txe=0 rxe=0 stx=0 srx=0 ecc=0/0 temp=-268 dco=0

Format	26	Min Freq	915000
Baud	9	Max Freq	928000
Air Speed	64	# of Channels	50
Net ID	25	Duty Cycle	100
Tx Power	20	LBT Rssi	0
ECC	☐	RTS CTS	☐
Mavlink	Mavlink	Max Window (ms)	131
Op Resend	☐	AES Encryption	☐
GPI1_1R/CIN	☐	AES Key	0000000000000000
GPI1_1R/COUT	☐		

Remote

Format		Min Freq	895000
Baud		Max Freq	895000
Air Speed		# of Channels	
Net ID	0	Duty Cycle	
Tx Power	1	LBT Rssi	
ECC	☐	RTS CTS	☐
Mavlink	RawData	Max Window (ms)	33
Op Resend	☐	AES Encryption	☐
GPI1_1R/CIN	☐	AES Key	
GPI1_1R/COUT	☐		

Done

Fig. 4.6: Configuring RFD900+ RF Module with 3DR Radio Config.

4.3.1.4 Embedding “Software Serial Library” and usage

In the project, in order to communicate with the devices with I/O Pins, it is required to use a software serial library. Normally, a hardware library has its own interrupt peripheral. This interrupt peripheral has a mechanism that senses the incomming bytes whenever there is a 0-bit coming. By sensing the first 0-bit, hardware serial structure catches the byte on background and stores the data in a bank called buffer. In the code, that byte in the buffer can be read when needed. So, a hardware serial structure always stores the incoming bytes without a need to be aware when it is coming.

A software serial structure on the other hand, does not have any interrupt structure. So, it is important to set the devices to communicate at low baud rates which are connected to software serial pins. This is because to prevent missing any incoming bit while the code is running for other purposes as one bit’s period should last longer than the main program’s one loop period.

In this project, Arduino’s ready to use embedded software serial library is used. Then RF module and Bluetooth module is set to communicate with 9600 bits per seconds. That will provide one bit to last long as about 1 microseconds. Which will allow 1 loop of the program to process at most 1ms.

The program’s main loop is coded to check the incoming bits everytime it prints the information to the smart phone via Bluetooth. The checking function’s code is shown below:

```
static void smartDelay(unsigned long ms){
  unsigned long start = millis();
  do {
    BT.listen();
    BToku();
    delay(1);
    while(Serial.available()>0)
    gps.encode(Serial.read());
    } while (millis() - start < ms);
}
```

As shown on the code above, this function acts as a delay function in main loop function. Its main mission is to check the incoming bits in software serial pins while it delays the main function. As there is a do-while statement, even if the delay function is called with an input of 0 value, it still performs the required command lines once so it checks the incoming bits once.

So, even if the function is called as "*smartDelay(0)*" in the main function, it executes the reading once to catch the data.

4.3.1.5 Building a string parsing function for communication between devices

The strings transceived between the rover device, reference device and the smart phone are constructed to start with a '$' character and end with a '*' character. Following are the constructed strings for communication between devices:

- The string generated by the reference device that to be sent to the rover device. This string carries the correction values that is used to correct the rover device.
    - $R,[*altitude*],[*latitude*],[*longitude*],[*gps-second*]*
        - R: Identifier for the string type
        - [*altitude*]: The difference between the actual altitude and the measured altitude.
        - [*latitude*]: The difference between the actual latitude and the measured latitude.
        - [*longitude*]: The difference between the actual longitude and the measured longitude.
        - [*gps-second*]: The GPS time's second part that is measured by the refence GPS module.
- The string generated by the mobile application and sent to the reference device via Bluetooth. This string is the command for the reference device that make the reference device to calculate its actual coordinates by filtering the continuous measurements and generate a fixed coordinate when there is no accurate known coordinate for the reference device's point.

- $A*
  - A: Command’s identifier name

- The string generated by mobile application that to be sent to the reference device via Bluetooth. This string is the command for the reference device to start transmitting the information string to the smart phone. This is necessary as the bluetooth module’s buffer stores all the data to be transmitted to the smart phone even if it is not connected to the smart phone. So, when the smart phone connects to the reference device, it collects the past data so it won’t be able to a real time monitor. So, the smart phone sends this string in order to start the communication when the application starts and connects to the reference device.
  - $B *
    - B: Command’s identifier name
- The string generated by mobile application that to be sent to the reference device via Bluetooth. This string carries the actual coordinate that manually entered by user to set the reference device to assign itself its actual point.
  - $C ,[*latitude*],[*longitude*], [*altitude*]*
    - C: Identifier for the string type
    - [*latitude*]: The actual latitude of the reference device.
    - [*longitude*]: The actual longitude of the reference device.
    - [*altitude*]: The actual altitude of the reference device
- The string generated by the reference device that to be sent to smart phone via Bluetooth. This string carries all the information in order to monitor the device process on smart phone.
  - $DAT,[*satellite*],[*hdop*],[*latitude*],[*longitude*],[*gps-seconds*],[*avg-altitude-cm*],[*act-alt-cm*],[*latitude-cm*],[*avg-lat-cm*],[*longitude-cm*],[*avg-lon-cm*],[*fail*],$R,[*cor-alt-cm*],[*cor-lat-cm*],[*cor-lon-cm*],[*gps-seconds*]*
    - DAT: Identifier for the string type
    - [*satellite*]: The number of satellites connected by GPS.
    - [*hdop*]: Horizontal dilution of precision value of the device.
    - [*latitude*]: Measured raw latitude.
    - [*longitude*]: Measured raw longitude.

        - [*gps-seconds*]: Measured GPS seconds.
        - [*avg-altitude-cm*]: Average of measured altitudes.
        - [*act-alt-cm*]: Actual altitude in centimeters (filtered calculation or entered manually).
        - [*latitude-cm*]: Measured raw latitude in centimeters.
        - [*avg-lat-cm*]: Actual latitude in centimeters (filtered calculation or entered manually).
        - [*longitude-cm*]: Measured raw longitude in centimeters.
        - [*avg-lon-cm*]: Actual longitude in centimeters (filtered calculation or entered manually).
        - [*fail*]: Number of failed strings, communication errors.
        - $R: The correction string's identifier.
        - [*cor-alt-cm*]: Altitude correction in centimeters.
        - [*cor-lat-cm*]: Latitude correction in centimeters.
        - [*cor-lon-cm*]: Longitude correction in centimeters.
        - [*gps-seconds*]: Measured GPS seconds to be sent to the rover device.

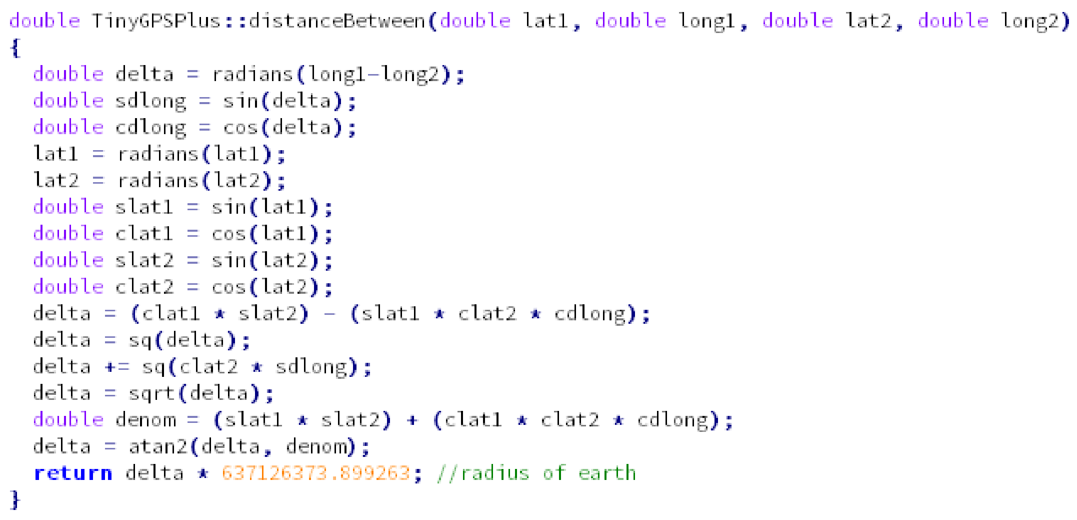
- The string generated by the rover device that to be sent to smart phone via Bluetooth. This string carries all the information in order to monitor the device process on smart phone.
    - $DAT,[satellite],[hdop],[latitude],[longitude],[gps-seconds],[altitude-cm],[cor-alt-cm],[latitude-cm],[cor-lat-cm],[longitude-cm],[cor-lon-cm],[fail],$R,[fil-alt-cm],[fil-lat-cm],[fil-lon-cm],[gps-seconds]*
        - DAT: Identifier for the string type
        - [satellite]: The number of satellites connected by GPS.
        - [hdop]: Horizontal dilution of precision value of the device.
        - [latitude]: Measured raw latitude.
        - [longitude]: Measured raw longitude.
        - [gps-seconds]: Measured GPS seconds.
        - [altitude-cm]: Measured raw altitude in centimeters.
        - [cor-alt-cm]: Corrected altitude in centimeters.
        - [latitude-cm]: Measured raw latitude in centimeters.
        - [cor-lat-cm]: Corrected latitude in centimeters.
        - [longitude-cm]: Measured raw longitude in centimeters.

- [cor-lon-cm]: Corrected longitude in centimeters.
- [fail]: Number of failed strings, communication errors.
- $R: The correction string’s identifier.
- [fil-alt-cm]: Filtered and corrected altitude in centimeters.
- [fil-lat-cm]: Filtered and corrected latitude in centimeters.
- [fil-lon-cm]: Filtered and corrected longitude in centimeters.
- [gps-seconds]: Measured GPS seconds that is received from rover device.

The parsing function adds the received characters together into a string. Then checks for the ‘$’ character and ‘*’ character in that string. Whenever the function hits ‘$’ character, it assigns the starting point of the string after the character. Then when it hits ‘*’ character, it assigns the finishing point of the string before that character. So the string is fully obtained with all the elements seperated by commas. So the parsing function splits the strings using the commas in order to obtained all the elements.

4.3.2 RTK Reference Device Embedded Software Design

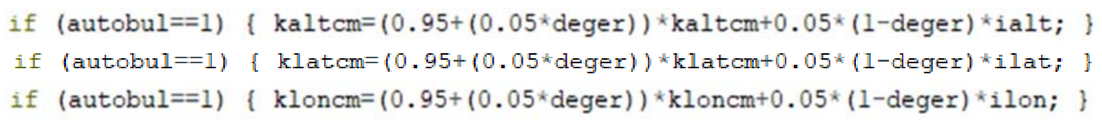
RTK reference device first reads the GPS module by using “TinyGPS+” library. Then retreives the latitude, longitude and altitude values with the embedded string parser of the library and converts them into centimeter unit with respect to 0th degree latitude and 0th degree longitude. Eqn.4.1. shows the converting algorithm from latitude and longitude to centimeters.



```
double TinyGPSPlus::distanceBetween(double lat1, double long1, double lat2, double long2)
{
  double delta = radians(long1-long2);
  double sdlong = sin(delta);
  double cdlong = cos(delta);
  lat1 = radians(lat1);
  lat2 = radians(lat2);
  double slat1 = sin(lat1);
  double clat1 = cos(lat1);
  double slat2 = sin(lat2);
  double clat2 = cos(lat2);
  delta = (clat1 * slat2) - (slat1 * clat2 * cdlong);
  delta = sq(delta);
  delta += sq(clat2 * sdlong);
  delta = sqrt(delta);
  double denom = (slat1 * slat2) + (clat1 * clat2 * cdlong);
  delta = atan2(delta, denom);
  return delta * 637126373.899263; //radius of earth
}
```

Eqn. 4.1: Latitude, Longitude to Centimeters Converstion Algorithm.

As the reference device needs to be set with the actual coordinate, it is programmed to obtain the actual coordinates by two methods. First is to find it itself, the second is to enter it manually by smart phone. To find the actual value itself, device is programmed to read continuous GPS data and filter the result for many minutes until fixing the coordinate values as shown on Eqn.4.2. If the coordinate of the reference device is already known, it is manually entered by smart phone and the values are obtained by the reference device via Bluetooth module.

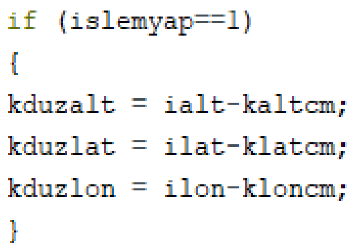


```
if (autobul==1) { kaltcm=(0.95+(0.05*deger))*kaltcm+0.05*(1-deger)*ialt; }
if (autobul==1) { klatcm=(0.95+(0.05*deger))*klatcm+0.05*(1-deger)*ilat; }
if (autobul==1) { kloncm=(0.95+(0.05*deger))*kloncm+0.05*(1-deger)*ilon; }
```

Eqn. 4.2: Filtering Algorithm to Find the Most Accurate Actual Point.

The variable “deger" shown on Eqn.4.2. is the HDOP value received from the GPS module. This variable is equal to HDOP/1000 and multiplied by the past readings in order to give more weight to the most current reading inversely proportional with the precision value.

After obtaining the coordinates, in order to generate RTK correction broadcast, the actual coordinates are simply substracted from the measured coordinates as shown on Eqn.4.3. Then generate a string with those calculated corrections of latitude, longitude and altitude to send them to the rover device via RF module.



```
if (islemyap==1)
{
kduzalt = ialt-kaltcm;
kduzlat = ilat-klatcm;
kduzlon = ilon-kloncm;
}
```

Eqn. 4.3: Substraction Process to Generate the Correction Broadcast by Reference Device.

After all, to monitor the device’s process, all needed calculations and measurements are put together into another string to be sent to the smart phone via Bluetooth.

4.3.3 RTK Rover Device Embedded Software Design

RTK rover device reads the GPS module by using “TinyGPS+” library. Then retreives the latitude, longitude and altitude values with the embedded string parser of the library and converts them into centimeter unit with respect to 0th degree latitude and 0th degree longitude as shown on Eqn.4.1.

As the rover device needs to correct its raw coordinate measurement and find its actual coordinate, it is programmed to obtain the RTK broadcast from the reference device via RF module and split the string that carries the correction values into its sub-elements as they are latitudal, longitudal and altitudal shifts provided by the refence device.

After obtaining the coordinate corrections, in order to find the most accurate coordinates, the raw coordinates are again substracted by the incoming coordinate corrections as shown on Eqn.4.4.

```
if (islemyap==1)
{
  kduzalt = ialt - RFDalt;
  kduzlat = ilat - RFDlat;
  kduzlon = ilon - RFDlon;
  kduzsec = gps.time.second() - RFDsec;
}
```

Eqn. 4.4: Substraction Process to Find the Corrected Coordinate by Rover Device

Then, the founded accurate coordinates are filtered to produce the most precise and stable values for latitude, longitude and altitude values as shown on Eqn.4.5.

```
kaltcm=(0.75+(0.25*deger))*kaltcm+0.25*(1-deger)*kduzalt;
klatcm=(0.75+(0.25*deger))*klatcm+0.25*(1-deger)*kduzlat;
kloncm=(0.75+(0.25*deger))*kloncm+0.25*(1-deger)*kduzlon;
```

Eqn. 4.5: Filtering Process to Find the Most Precise Values by Rover Device.

After all, to monitor the device’s process, all needed calculations and measurements are put together into another string to be sent to the smart phone via Bluetooth.

4.3.4 App Inventor 2 – Mobile Application Design

The mobile application for RTK GPS devices, is developed by using App Inventor 2 IDE.

App Inventor 2 is a drag&drop type coding platform for Android mobile devices. It is a WEB based IDE that easily connects to an Android device as a real time emulator that is connected to the same wireless network with the PC. App Inventor just connects to the mobile device with an easy QR code integration process then it starts compiling the code on the smart phone. It is required to download the "MIT AI Companion" application to test the applications developed.

4.3.4.1 Designing the visual appearance of the application

Three buttons are assigned for sending some commands in the mobile application to the devices. And there is another button for selecting the device to connect via Bluetooth. There is a layout to show the measured and calculated points visually on the phone user interface.

Some textboxes are put on the interface in order to assign the known latitude, longitude and altitude values to the reference GPS device. Then there are some text labels that are put on the interface in order to show the devices' measurements and calculations for monitoring the variables and parameters.

The visual design is shown on Fig.4.7.a. and Fig.4.7.b.

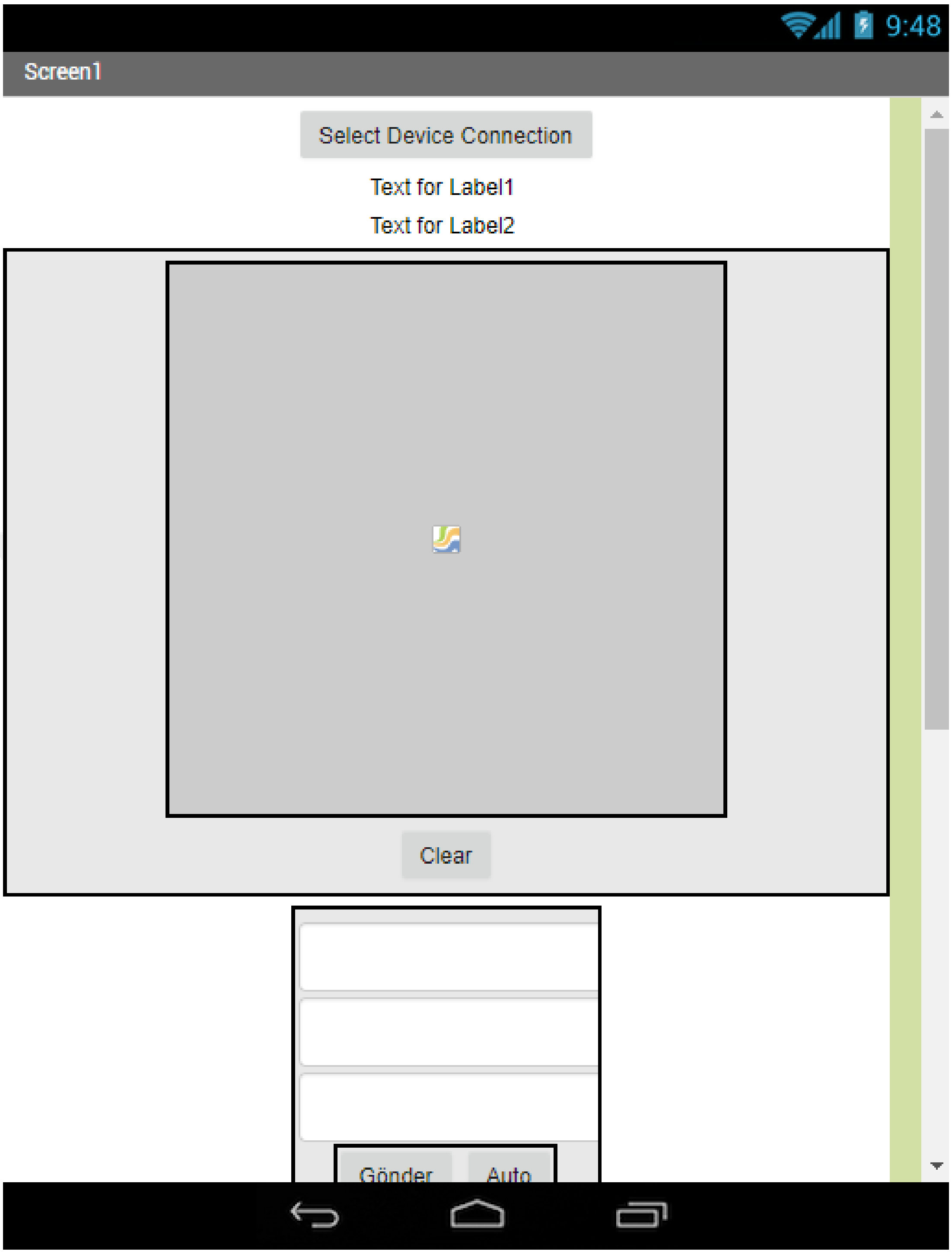
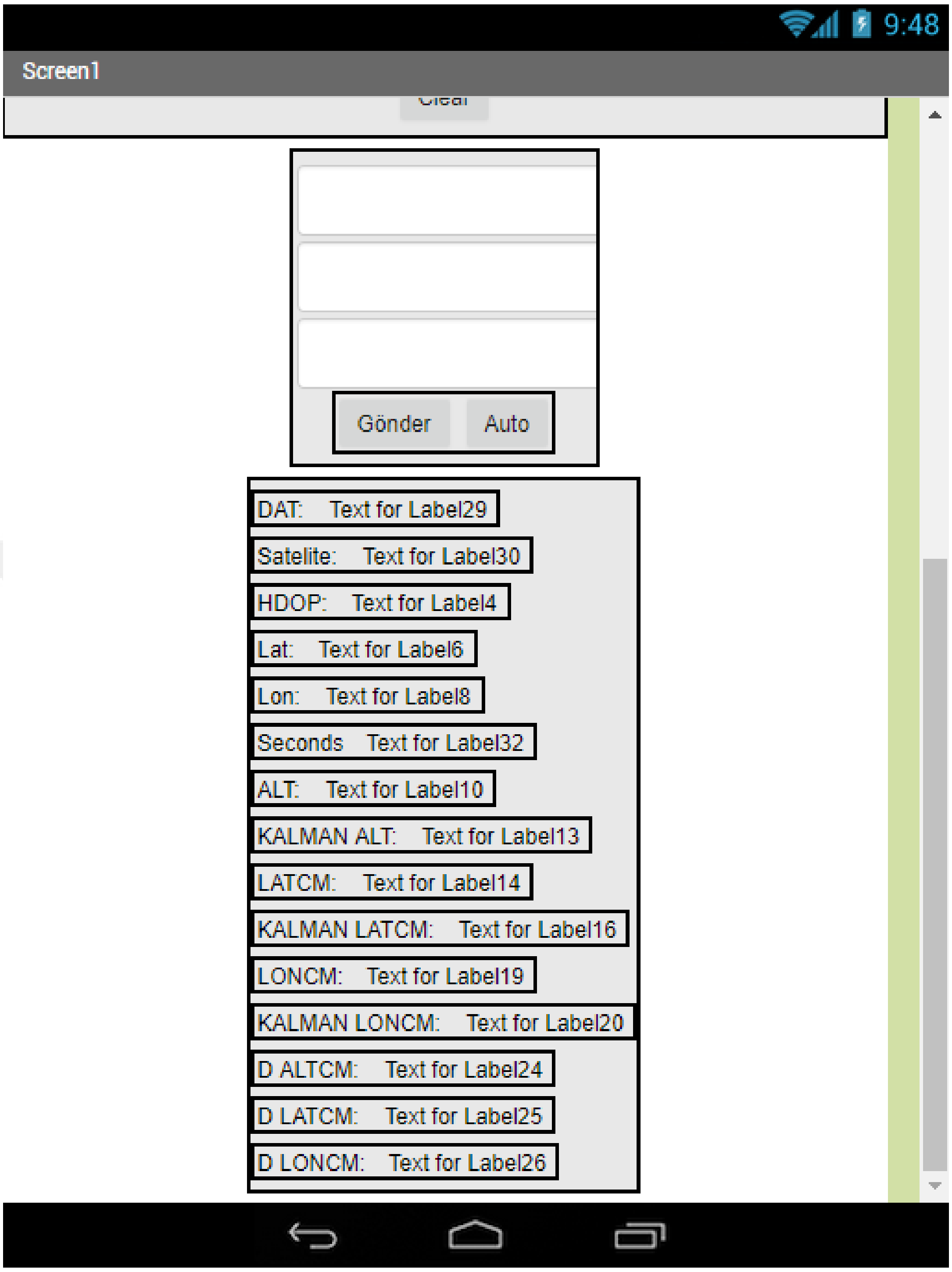


Fig. 4.7.a: 1st Screenshot of Visual Design of the Android Application of RTK Devices.



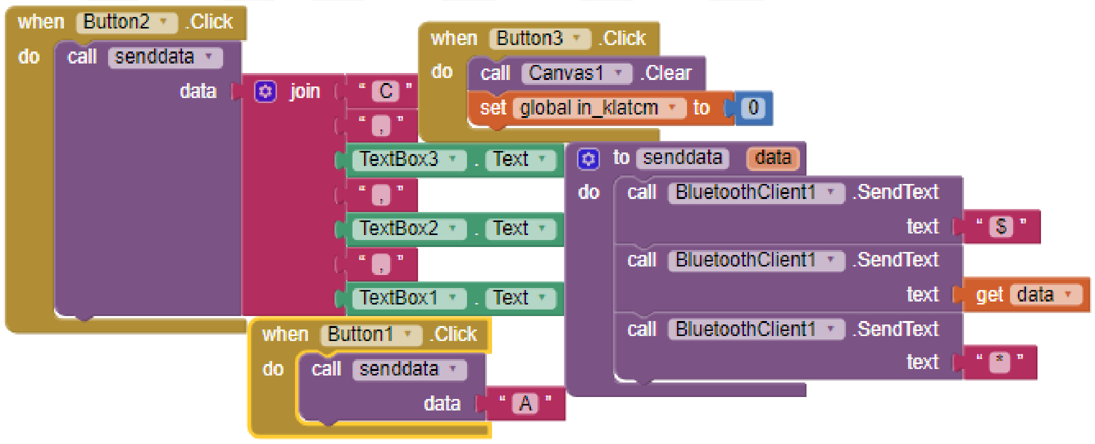
DAT: Text for Label29
Satelite: Text for Label30
HDOP: Text for Label4
Lat: Text for Label6
Lon: Text for Label8
Seconds Text for Label32
ALT: Text for Label10
KALMAN ALT: Text for Label13
LATCM: Text for Label14
KALMAN LATCM: Text for Label16
LONCM: Text for Label19
KALMAN LONCM: Text for Label20
D ALTCM: Text for Label24
D LATCM: Text for Label25
D LONCM: Text for Label26

Fig. 4.7.b: 2nd Screenshot of Visual Design of the Android Application of RTK Devices.

4.3.4.2 Coding the Buttons' functions

There are three buttons for some functions. The button named 'Clear' is coded to center the actual point or calculated corrected point in the layout's canvas. The other buttons 'Send' and 'Auto' is to send the device 2 commands. The 'Send' button is to send the latitude, longitude and altitude values to the reference device for the known coordinates which is to be sent by the user. The ‘Auto’ button is to command the reference device that the reference coordinate is not known and it is required to find the coordinate by the device itself by filtering and having average values. So, the ‘Auto’ button sends the command ‘$A*’ string to the reference device and the button ‘Send’ sends the command ‘$C,[*lat*],[*lon*],[*alt*]*’ to the device as mentioned on the section 4.3.1.5.

To send the data, there is a function called spesifically as “procedures” is named as ‘senddata’ in the coding. Its function is to send any data by adding ‘$’ character to the beginning and ‘*’ character to the ending in order to set up the string as a procedure. The code blocks for needed buttons’ functions and sending data procedure’s function is shown on Fig.4.8.



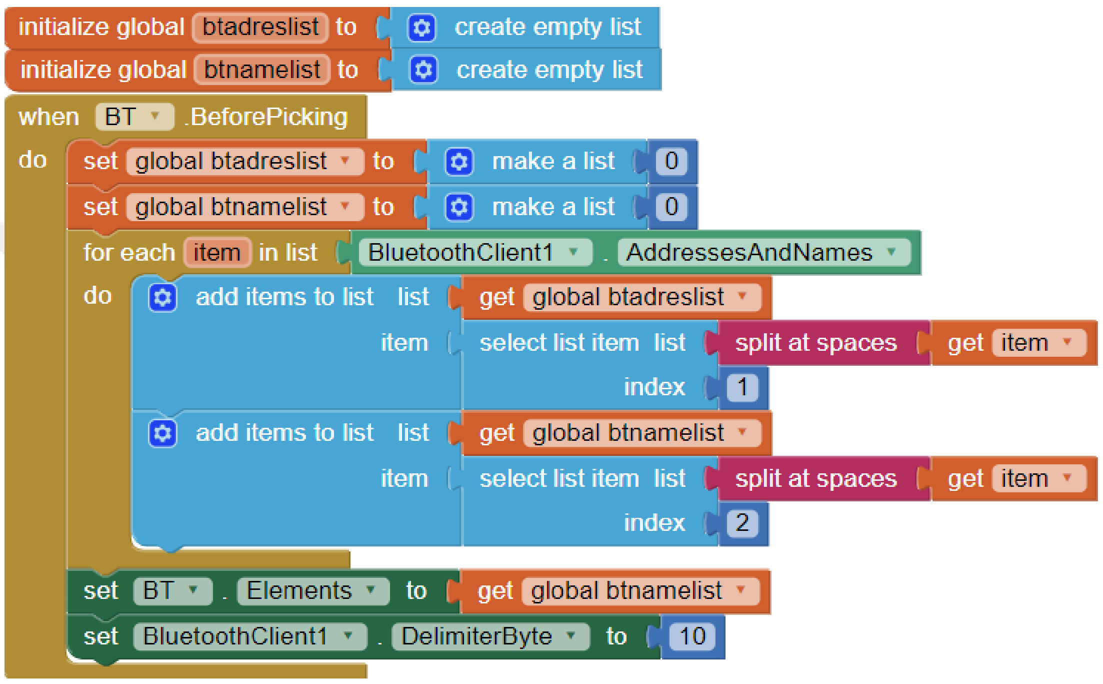
```
when Button2 .Click
do  call senddata
         data  join  " C "
                      " , "
                      TextBox3 . Text
                      " , "
                      TextBox2 . Text
                      " , "
                      TextBox1 . Text

when Button3 .Click
do  call Canvas1 .Clear
    set global in_klatcm to 0

to senddata data
do  call BluetoothClient1 .SendText
         text " $ "
    call BluetoothClient1 .SendText
         text get data
    call BluetoothClient1 .SendText
         text " * "

when Button1 .Click
do  call senddata
         data " A "
```

Fig. 4.8: Buttons and the Procedure for Sending Data to a Bluetooth Device.

4.3.4.3 Coding the bluetooth connection and device selection functions

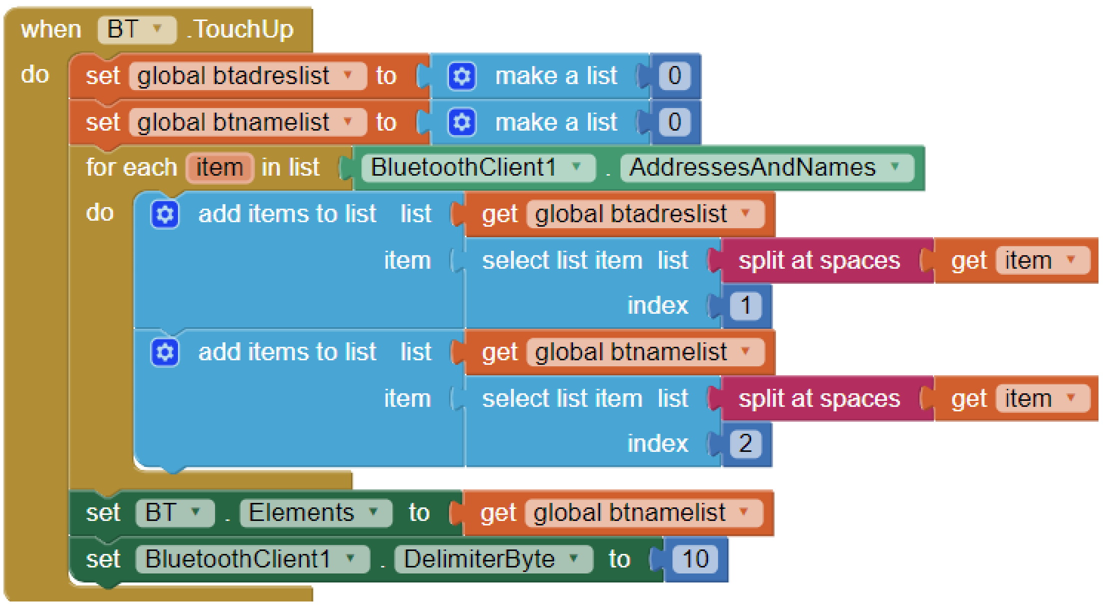
There is a Bluetooth connection button for selecting and connecting the device to be connected via Bluetooth. It is named as “Bluetooth Device Selection”. That button is a “list picker” element which is used to set up a list for the user to choose the needed “element”. This list is coded to show the user all the Bluetooth devices in range.

When it is first loaded on the screen, with the help of the “BeforePicking” function, it fills the list with the available bluetooth devices names in range and their paired addresses to another list. When this button is touched, the “TouchUp” function is activated and it sets the needed Bluetooth device to be connected. After setting the selection, the function “AfterPicking” is activated and it is coded to connect to the device that selected. And if it connects to the device, it sends the command “$B*” to this device in order to start the device send its informations to be monitored as it is a need for not filling the buffer as it is mentioned before on section 4.3.1.5. All the coding are shown below on Fig.4.9.a., Fig.4.9.b. and Fig.4.9.c.



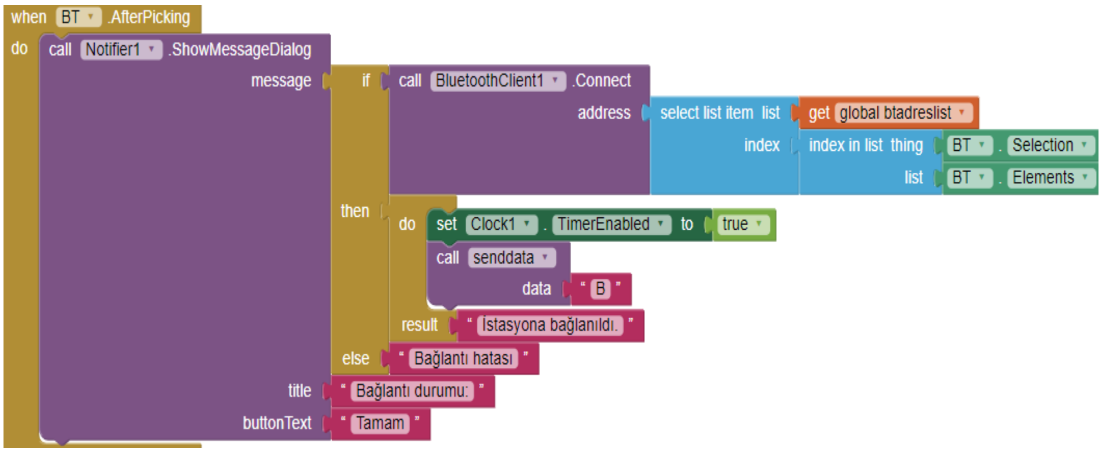
```
initialize global btadreslist to create empty list
initialize global btnamelist to create empty list
when BT .BeforePicking
do  set global btadreslist to make a list 0
    set global btnamelist to make a list 0
    for each item in list BluetoothClient1 . AddressesAndNames
    do  add items to list  list get global btadreslist
                           item select list item  list split at spaces get item
                                                  index 1
        add items to list  list get global btnamelist
                           item select list item  list split at spaces get item
                                                  index 2
    set BT . Elements to get global btnamelist
    set BluetoothClient1 . DelimiterByte to 10
```

Fig. 4.9.a: 1st Screenshot of Selecting the Bluetooth Device to Connect with App Inventor.



```
when BT .TouchUp
do  set global btadreslist to make a list 0
    set global btnamelist to make a list 0
    for each item in list BluetoothClient1 . AddressesAndNames
    do  add items to list  list get global btadreslist
                           item select list item  list split at spaces get item
                                                  index 1
        add items to list  list get global btnamelist
                           item select list item  list split at spaces get item
                                                  index 2
    set BT . Elements to get global btnamelist
    set BluetoothClient1 . DelimiterByte to 10
```

Fig. 4.9.b: 2nd Screenshot of Selecting the Bluetooth Device to Connect with App Inventor.

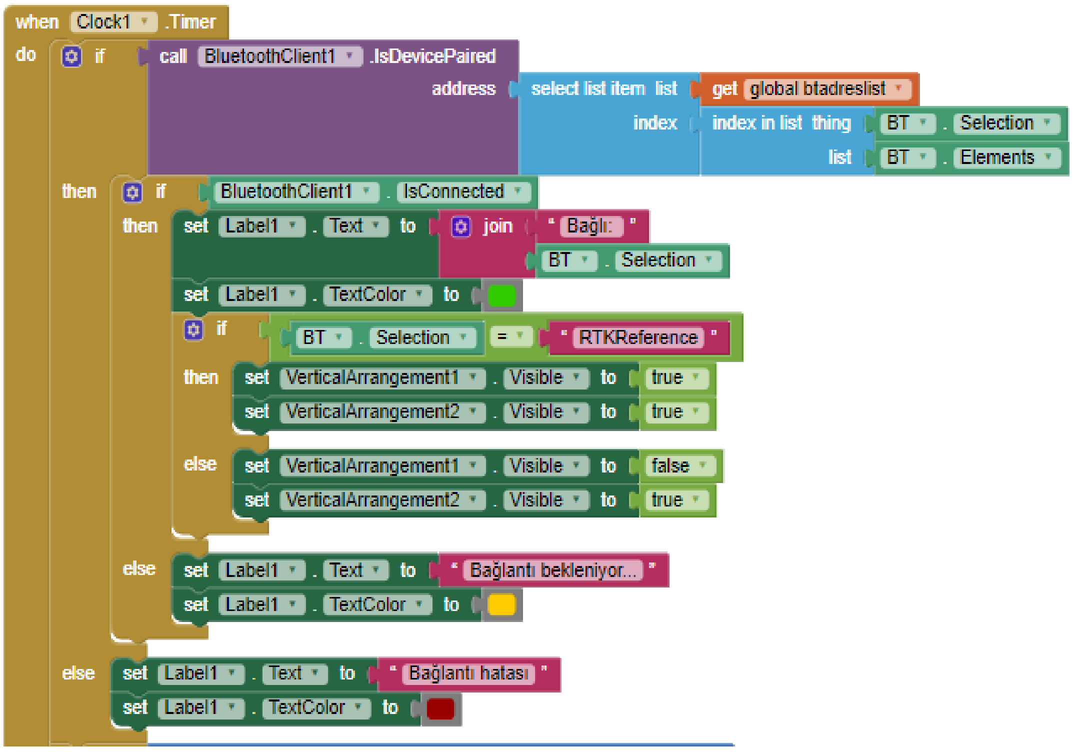


```
when BT .AfterPicking
do  call Notifier1 .ShowMessageDialog
        message  if    call BluetoothClient1 .Connect
                            address  select list item  list   get global btadreslist
                                                       index  index in list  thing  BT . Selection
                                                                             list   BT . Elements
                 then  do  set Clock1 . TimerEnabled to  true
                           call senddata
                                data  " B "
                       result  " İstasyona bağlanıldı. "
                 else  " Bağlantı hatası "
        title       " Bağlantı durumu: "
        buttonText  " Tamam "
```

Fig. 4.9.c: 3rd Screenshot of Selecting the Bluetooth Device to Connect with App Inventor.

4.3.4.4 Coding the main loop function

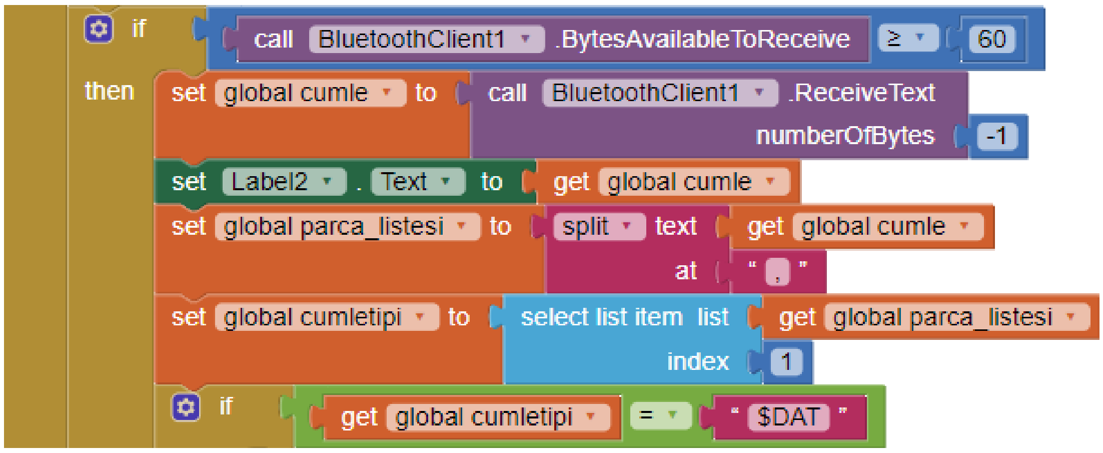
The main loop, spesifically called as “Timer” function is set to be fired every second. It first checks the Bluetooth-device connection. If it exists, it checks if it is connected to the reference device or rover device. Then, if it is connected to the rover device, it is coded to hide the layout of the part that allow user to assign the reference device its actual point. Then it informs the user about connection status. The code blocks about Bluetooth connection are shown on Fig.4.10.



```
when Clock1 .Timer
do  if    call BluetoothClient1 .IsDevicePaired
               address  select list item  list   get global btadreslist
                                          index  index in list  thing  BT . Selection
                                                                list   BT . Elements
    then  if    BluetoothClient1 . IsConnected
          then  set Label1 . Text to  join  " Bağlı: "
                                            BT . Selection
                set Label1 . TextColor to 🟩
                if    BT . Selection  =  " RTKReference "
                then  set VerticalArrangement1 . Visible to  true
                      set VerticalArrangement2 . Visible to  true
                else  set VerticalArrangement1 . Visible to  false
                      set VerticalArrangement2 . Visible to  true
          else  set Label1 . Text to  " Bağlantı bekleniyor... "
                set Label1 . TextColor to 🟨
    else  set Label1 . Text to  " Bağlantı hatası "
          set Label1 . TextColor to 🟥
```

Fig. 4.10: Checking Bluetooth Connection in App Inventor.

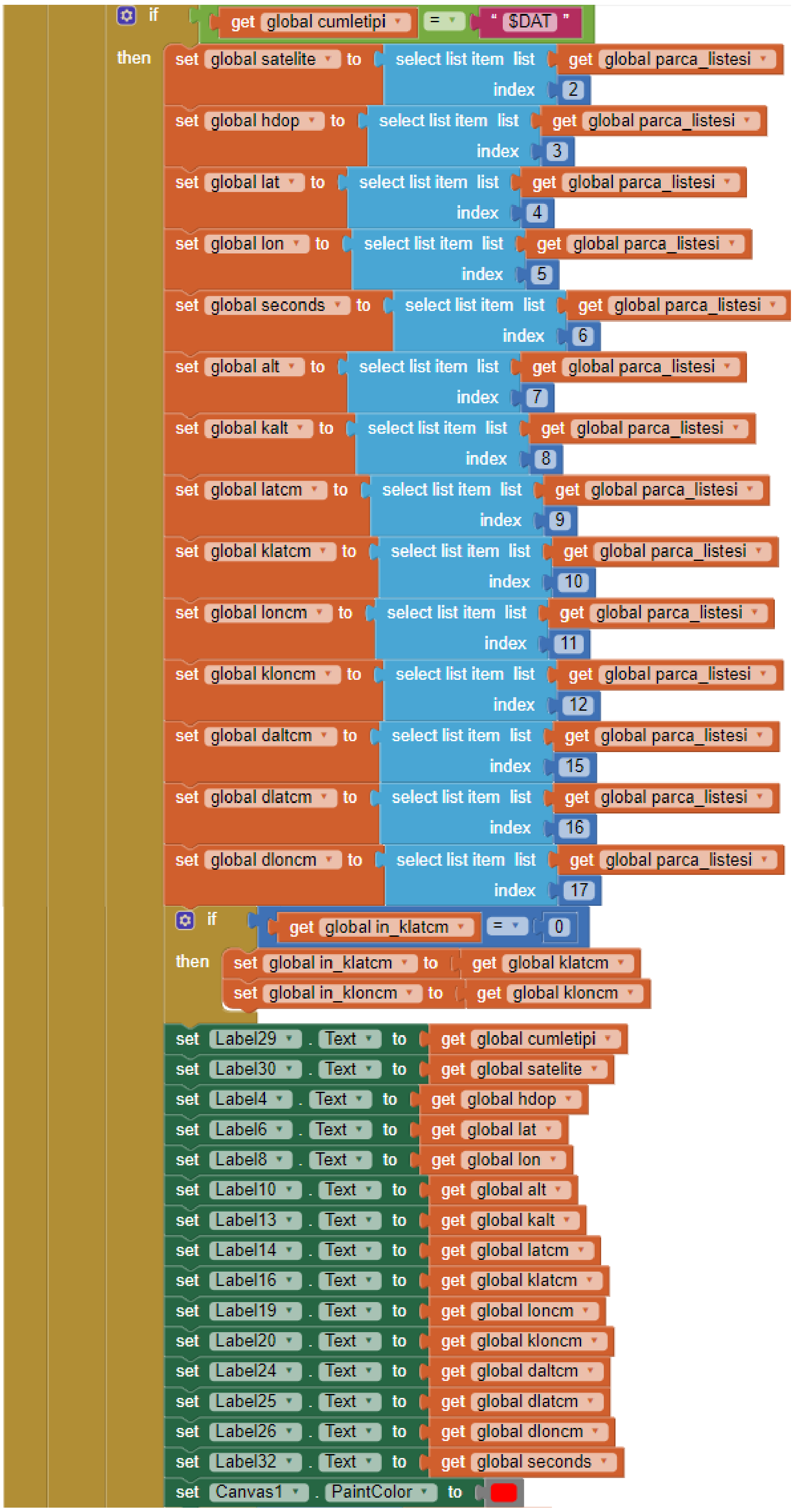
After connection is established, it checks the availability of characters in serial communciation buffer. If there are characters received, it starts reading the buffer until the end of the line which is delimited by the special character “CR” that corresponds to a byte value of ‘10’, so it gets the string completely. Related code blocks are shown on Fig.4.11.



```
if    call BluetoothClient1 .BytesAvailableToReceive ≥ 60
then  set global cumle to call BluetoothClient1 .ReceiveText
                              numberOfBytes -1
      set Label2 . Text to get global cumle
      set global parca_listesi to split text get global cumle
                                         at " , "
      set global cumletipi to select list item list get global parca_listesi
                                              index 1
      if    get global cumletipi = " $DAT "
```

Fig. 4.11: Receiving Bytes with App Inventor.

After getting the string, it is coded to parse the string and show the information on the corresponding text label. The code blocks of the parsed information classification are shown on Fig.4.12.

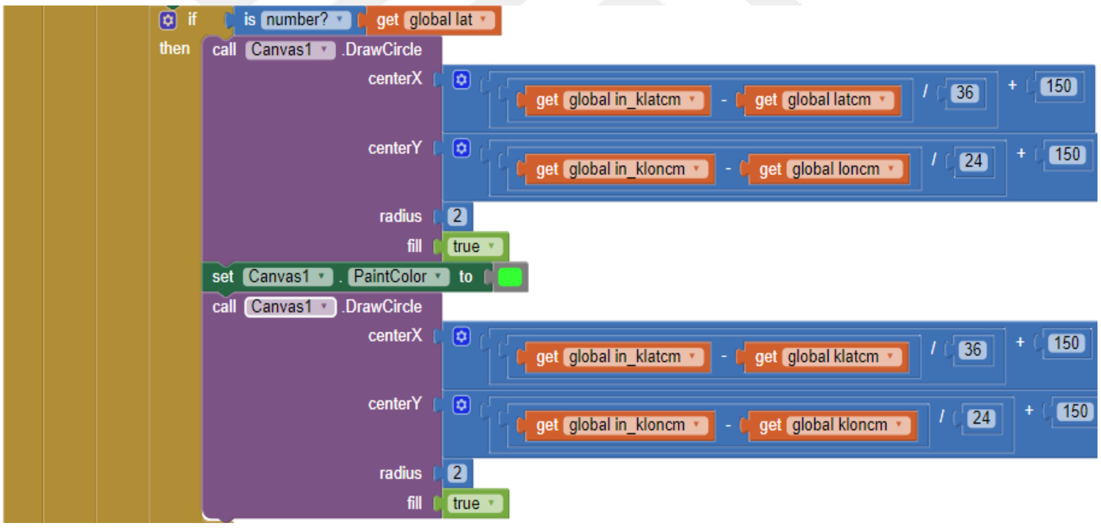


```
if   get global cumletipi = " $DAT "
then set global satelite to select list item list get global parca_listesi
                                              index 2
     set global hdop to select list item list get global parca_listesi
                                          index 3
     set global lat to select list item list get global parca_listesi
                                         index 4
     set global lon to select list item list get global parca_listesi
                                         index 5
     set global seconds to select list item list get global parca_listesi
                                             index 6
     set global alt to select list item list get global parca_listesi
                                         index 7
     set global kalt to select list item list get global parca_listesi
                                          index 8
     set global latcm to select list item list get global parca_listesi
                                           index 9
     set global klatcm to select list item list get global parca_listesi
                                            index 10
     set global loncm to select list item list get global parca_listesi
                                           index 11
     set global kloncm to select list item list get global parca_listesi
                                            index 12
     set global daltcm to select list item list get global parca_listesi
                                            index 15
     set global dlatcm to select list item list get global parca_listesi
                                            index 16
     set global dloncm to select list item list get global parca_listesi
                                            index 17
     if   get global in_klatcm = 0
     then set global in_klatcm to get global klatcm
          set global in_kloncm to get global kloncm
     set Label29 . Text to get global cumletipi
     set Label30 . Text to get global satelite
     set Label4 . Text to get global hdop
     set Label6 . Text to get global lat
     set Label8 . Text to get global lon
     set Label10 . Text to get global alt
     set Label13 . Text to get global kalt
     set Label14 . Text to get global latcm
     set Label16 . Text to get global klatcm
     set Label19 . Text to get global loncm
     set Label20 . Text to get global kloncm
     set Label24 . Text to get global daltcm
     set Label25 . Text to get global dlatcm
     set Label26 . Text to get global dloncm
     set Label32 . Text to get global seconds
     set Canvas1 . PaintColor to 🟥
```

Fig. 4.12: Parsing the String and Assigning Them into Text Labels in App Inventor.

Then, the code is developed to show the raw coordinates and the processed actual or calculated coordinates on and around the center of the canvas layout. This is done by adding the half of 300x300 pixel canvas' value (150) to the mapped fixed value of the coordinates.

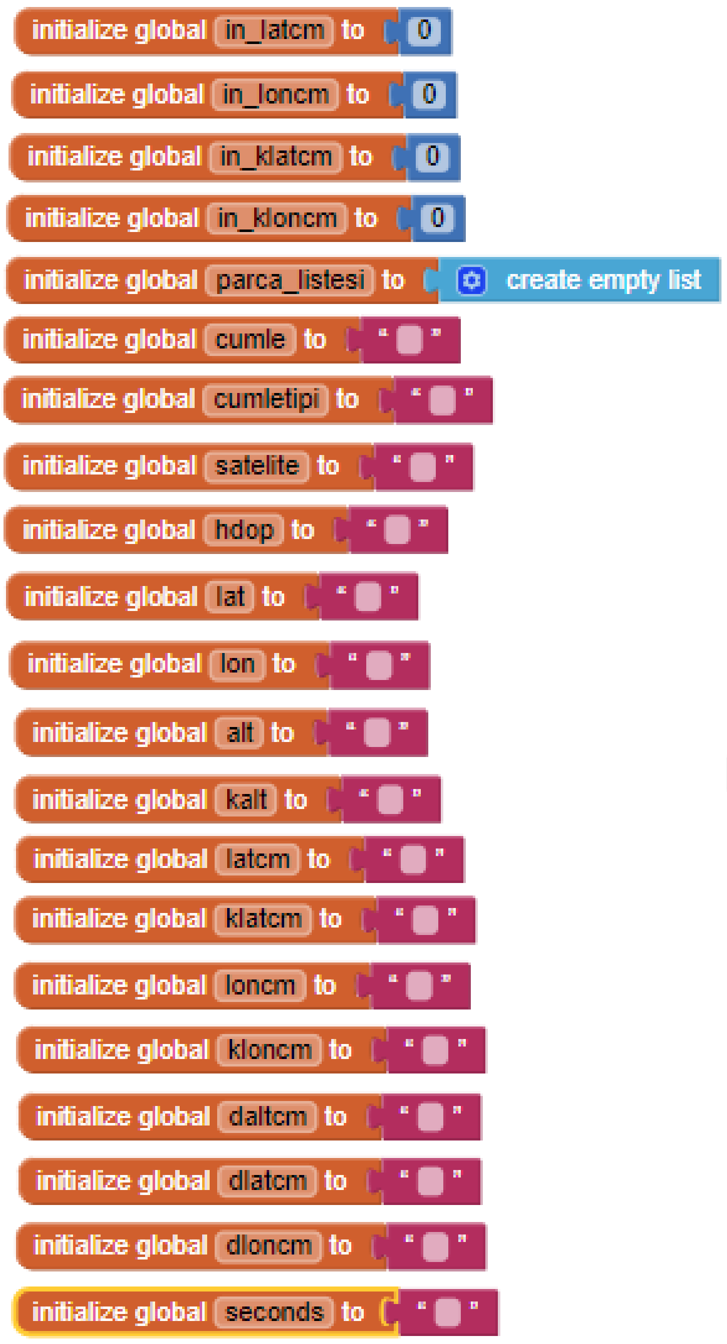
Mapping is done by substracting the new calculated and measured values from the first fixed coordinate value in order to place all the points around the center as the difference between all points will be around '0' cms. Then the difference between the new coordinate values and first fixed coordinate, which is assigned by user by touching the 'Clear' button, is divided by 36 on vertical axis and by 24 on horizontal axis in order to make 1 pixel to represent 36 cms on vertical axis and 24 cms on horizontal axis that means to set up the visualising resolution. The parameter values are assigned as 36 and 24 as the horizontal error was generally greater than the vertical error because of the experiment field. The code blocks of the calculations mentioned are shown on Fig.4.13.



```
if   is number?  get global lat
then call Canvas1 .DrawCircle
          centerX  ( get global in_klatcm - get global latcm ) / 36 + 150
          centerY  ( get global in_kloncm - get global loncm ) / 24 + 150
          radius   2
          fill     true
     set Canvas1 . PaintColor to 🟩
     call Canvas1 .DrawCircle
          centerX  ( get global in_klatcm - get global klatcm ) / 36 + 150
          centerY  ( get global in_kloncm - get global kloncm ) / 24 + 150
          radius   2
          fill     true
```

Fig. 4.13: The Code Blocks for Visualising Measurements and Calculations on Smart Phone Application.

The required variable definitions for the application is shown below on Fig.4.14.



```
initialize global in_latcm to 0
initialize global in_loncm to 0
initialize global in_klatcm to 0
initialize global in_kloncm to 0
initialize global parca_listesi to create empty list
initialize global cumle to " "
initialize global cumletipi to " "
initialize global satelite to " "
initialize global hdop to " "
initialize global lat to " "
initialize global lon to " "
initialize global alt to " "
initialize global kalt to " "
initialize global latcm to " "
initialize global klatcm to " "
initialize global loncm to " "
initialize global kloncm to " "
initialize global daltcm to " "
initialize global dlatcm to " "
initialize global dloncm to " "
initialize global seconds to " "
```

Fig. 4.14: Variables for the RTK Devices' Smart Phone Application.

CHAPTER FIVE

EXPERIMENTAL RESULTS

As the GPS module reads with an accuracy of 2.5 meters as stated on section 4.1.1.4, it would be better to make the measurments in a weak signal area for a better visibility. In order to provide this, measurements and tests are made in an indoor place.

5.1 Results of Rover Device with a Calculated Reference Value

As the first experiment, reference device is set on a fixed point and configured to calculate its own coordinate in auto mode by calculating the averages. After reference device finds its slowly changing average coordinates it started to broadcast the difference between the raw GPS measurements and the calculated average. Then the rover device is set and fixed on another point to observe the raw readings and the corrected coordinates via the broadcasts read.

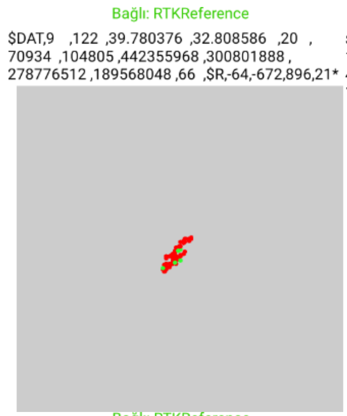
Reference device first gets the raw GPS measurements. Then, it starts to find an accurate fixed point by the wegihting average kalman-based filter iterations which lasts for 3 to 5 minutes.

After it finds a slowly floating mostly fixed coordinate, it assumes this point as the actual coordinate. Then it starts to calculate the average of last 25 GPS readings one per second.

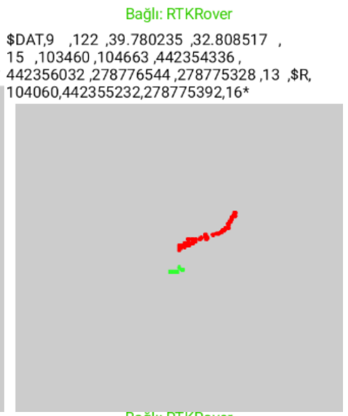
Then the reference device calculates the difference between the calculated actual point and the last 25 seconds average and broadcasts it as the correction value.

Rover device, again calculates the average of its own raw GPS readings every 25 seconds. Then substracts the broadcasted correction values from the average of 25 GPS readings in order to find the corrected coordinates.

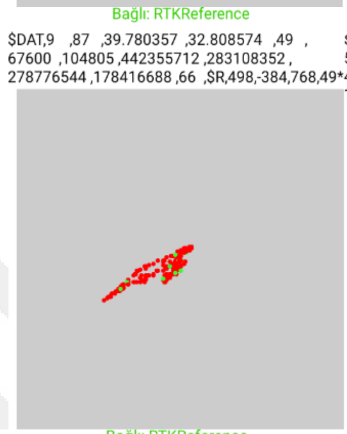
Fig.5.1.a. and Fig.5.1.b. shows the comparison between the rover and reference devices' data with 9 screenshots of user interfaces.


Bağlı: RTKReference

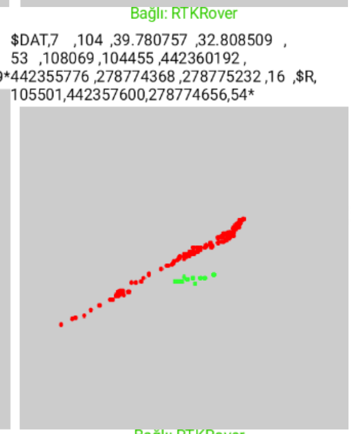
```
$DAT,9   ,122  ,39.780376  ,32.808586   ,20   ,
70934  ,104805 ,442355968 ,300801888 ,
278776512 ,189568048 ,66  ,$R,-64,-672,896,21*
```


Bağlı: RTKRover

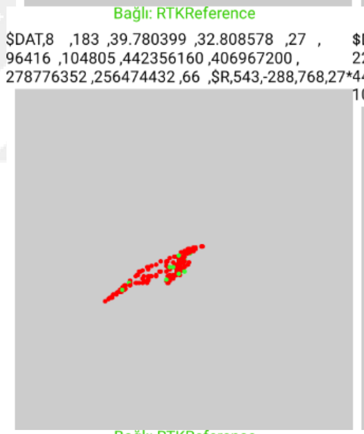
```
$DAT,9   ,122  ,39.780235  ,32.808517   ,
15   ,103460 ,104663 ,442354336 ,
442356032 ,278776544 ,278775328 ,13  ,$R,
104060,442355232,278775392,16*
```


Bağlı: RTKReference

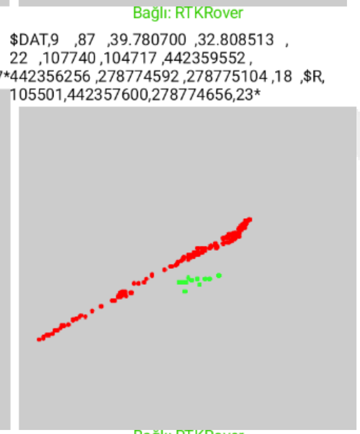
```
$DAT,9   ,87   ,39.780357  ,32.808574   ,49   ,
67600  ,104805 ,442355712 ,283108352 ,
278776544 ,178416688 ,66  ,$R,498,-384,768,49*
```


Bağlı: RTKRover

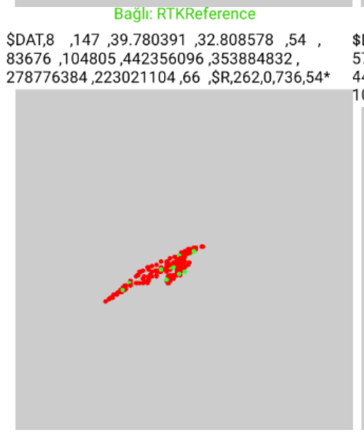
```
$DAT,7   ,104  ,39.780757  ,32.808509   ,
53   ,108069 ,104455 ,442360192 ,
442355776 ,278774368 ,278775232 ,16  ,$R,
105501,442357600,278774656,54*
```


Bağlı: RTKReference

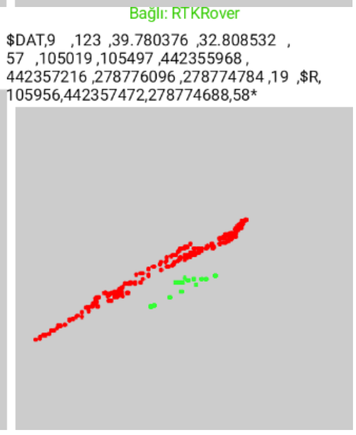
```
$DAT,8   ,183  ,39.780399  ,32.808578   ,27   ,
96416  ,104805 ,442356160 ,406967200 ,
278776352 ,256474432 ,66  ,$R,543,-288,768,27*
```


Bağlı: RTKRover

```
$DAT,9   ,87   ,39.780700  ,32.808513   ,
22   ,107740 ,104717 ,442359552 ,
442356256 ,278774592 ,278775104 ,18  ,$R,
105501,442357600,278774656,23*
```

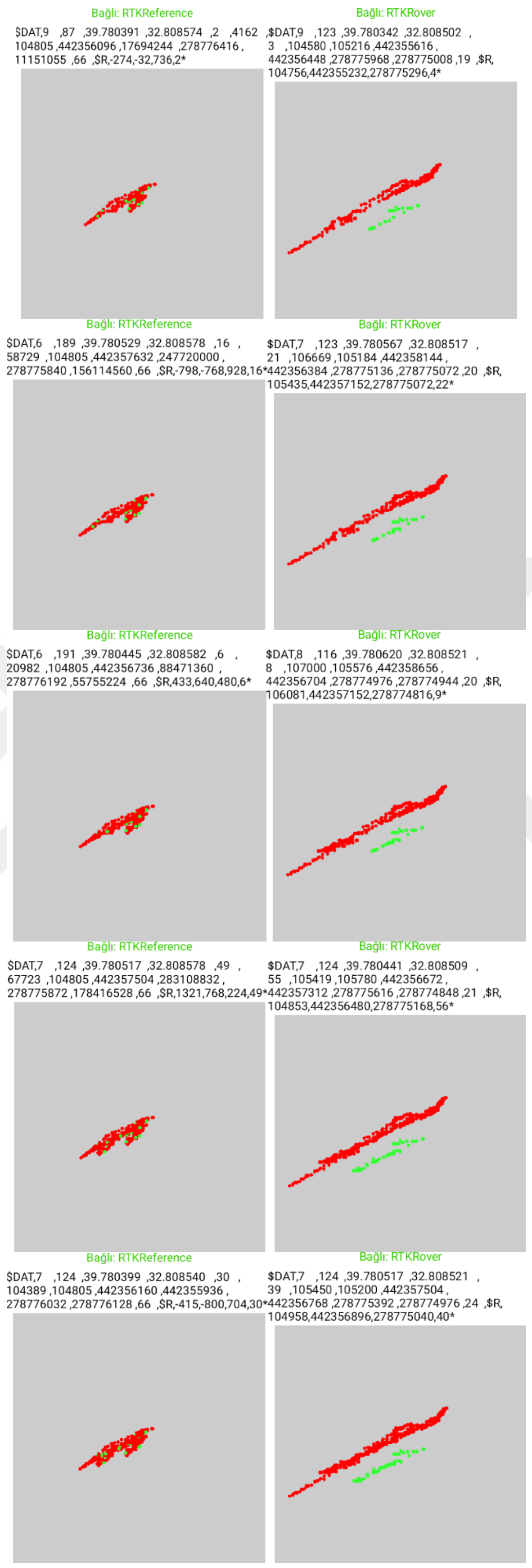

Bağlı: RTKReference

```
$DAT,8   ,147  ,39.780391  ,32.808578   ,54   ,
83676  ,104805 ,442356096 ,353884832 ,
278776384 ,223021104 ,66  ,$R,262,0,736,54*
```


Bağlı: RTKRover

```
$DAT,9   ,123  ,39.780376  ,32.808532   ,
57   ,105019 ,105497 ,442355968 ,
442357216 ,278776096 ,278774784 ,19  ,$R,
105956,442357472,278774688,58*
```

Fig. 5.1.a: Screenshots 1st to 4th of the Smart Phone User Interface for Both of the Devices for an Unknown Point.



```
$DAT,9   ,87   ,39.780391  ,32.808574  ,2   ,4162  ,
104805 ,442356096 ,17694244  ,278776416 ,
11151055  ,66  ,$R,-274,-32,736,2*
```

```
$DAT,9   ,123  ,39.780342  ,32.808502  ,
3   ,104580 ,105216 ,442355616 ,
442356448 ,278775968 ,278775008 ,19  ,$R,
104756,442355232,278775296,4*
```

```
$DAT,6   ,189  ,39.780529  ,32.808578  ,16  ,
58729  ,104805 ,442357632 ,247720000 ,
278775840 ,156114560 ,66  ,$R,-798,-768,928,16*
```

```
$DAT,7   ,123  ,39.780567  ,32.808517  ,
21   ,106669 ,105184 ,442358144 ,
442356384 ,278775136 ,278775072 ,20  ,$R,
105435,442357152,278775072,22*
```

```
$DAT,6   ,191  ,39.780445  ,32.808582  ,6   ,
20982  ,104805 ,442356736 ,88471360  ,
278776192 ,55755224  ,66  ,$R,433,640,480,6*
```

```
$DAT,8   ,116  ,39.780620  ,32.808521  ,
8   ,107000 ,105576 ,442358656 ,
442356704 ,278774976 ,278774944 ,20  ,$R,
106081,442357152,278774816,9*
```

```
$DAT,7   ,124  ,39.780517  ,32.808578  ,49  ,
67723  ,104805 ,442357504 ,283108832 ,
278775872 ,178416528 ,66  ,$R,1321,768,224,49*
```

```
$DAT,7   ,124  ,39.780441  ,32.808509  ,
55   ,105419 ,105780 ,442356672 ,
442357312 ,278775616 ,278774848 ,21  ,$R,
104853,442356480,278775168,56*
```

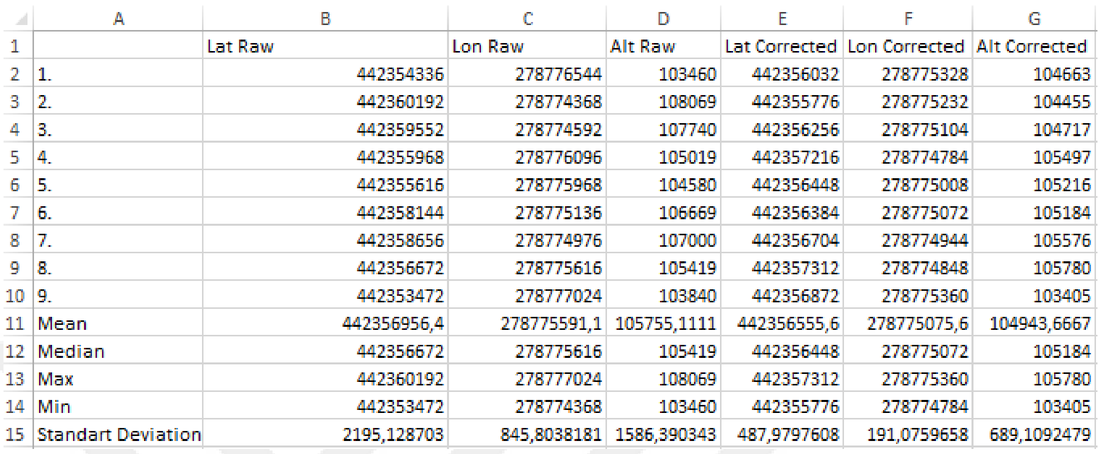
```
$DAT,7   ,124  ,39.780399  ,32.808540  ,30  ,
104389 ,104805 ,442356160 ,442355936 ,
278776032 ,278776128 ,66  ,$R,-415,-800,704,30*
```

```
$DAT,7   ,124  ,39.780517  ,32.808521  ,
39   ,105450 ,105200 ,442357504 ,
442356768 ,278775392 ,278774976 ,24  ,$R,
104958,442356896,278775040,40*
```

Fig. 5.1.b: Screenshots 5th to 9th of the Smart Phone User Interface for Both of the Devices for an Unknown Point.

After those experiments, the rover device's raw and corrected measurements were processed in Microsot Excel in order to calculate mean, median, max, min and standard deviation values as shown on Table 5.1.

Table 5.1: Rover Device's Raw and Corrected Coordinates in Auto Mode.

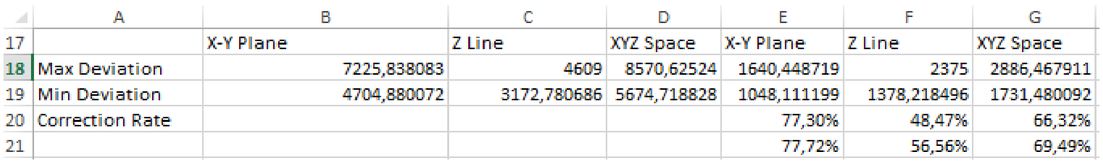


	A	B	C	D	E	F	G
1		Lat Raw	Lon Raw	Alt Raw	Lat Corrected	Lon Corrected	Alt Corrected
2	1.	442354336	278776544	103460	442356032	278775328	104663
3	2.	442360192	278774368	108069	442355776	278775232	104455
4	3.	442359552	278774592	107740	442356256	278775104	104717
5	4.	442355968	278776096	105019	442357216	278774784	105497
6	5.	442355616	278775968	104580	442356448	278775008	105216
7	6.	442358144	278775136	106669	442356384	278775072	105184
8	7.	442358656	278774976	107000	442356704	278774944	105576
9	8.	442356672	278775616	105419	442357312	278774848	105780
10	9.	442353472	278777024	103840	442356872	278775360	103405
11	Mean	442356956,4	278775591,1	105755,1111	442356555,6	278775075,6	104943,6667
12	Median	442356672	278775616	105419	442356448	278775072	105184
13	Max	442360192	278777024	108069	442357312	278775360	105780
14	Min	442353472	278774368	103460	442355776	278774784	103405
15	Standart Deviation	2195,128703	845,8038181	1586,390343	487,9797608	191,0759658	689,1092479

As it is observed on Table 5.1. the average mean of raw GPS coordinates is (lat: 442356956.4cms; lon: 278775591.1cms) and the standard deviation for raw coordinates is (lat: 2195.1cms; lon: 845.8cms) while the mean of corrected coordinates is (lat: 442356555.6cms; lon: 278775075.6cms) and standart deviation for corrected coordinates is (lat: 488cms; lon: 191.1cms)

On Table 5.2. the rover device's maximum and minimum deviations about raw and corrected coordinates are calculated in order to calculate the error reduction percentage of the overall system.

Table 5.2: Rover Device Measurement Deviations and Correction Rate in Auto Mode.



	A	B	C	D	E	F	G
17		X-Y Plane	Z Line	XYZ Space	X-Y Plane	Z Line	XYZ Space
18	Max Deviation	7225,838083	4609	8570,62524	1640,448719	2375	2886,467911
19	Min Deviation	4704,880072	3172,780686	5674,718828	1048,111199	1378,218496	1731,480092
20	Correction Rate				77,30%	48,47%	66,32%
21					77,72%	56,56%	69,49%

As observed on Table 5.2. maximum deviation on X-Y plane is calculated by adding squares of the maximum and minimum values of latitude and longitude together and taking square root of this addition. Then the minimum deviaton is calculated by multiplying the standard deviation by 2. After obtaining the maximum and minimum devaitons, they are assigned as the diameter of the error circle. So, the error circles can be demonstrated around the mean values of coordinates on the graphic shown on Fig.5.2.

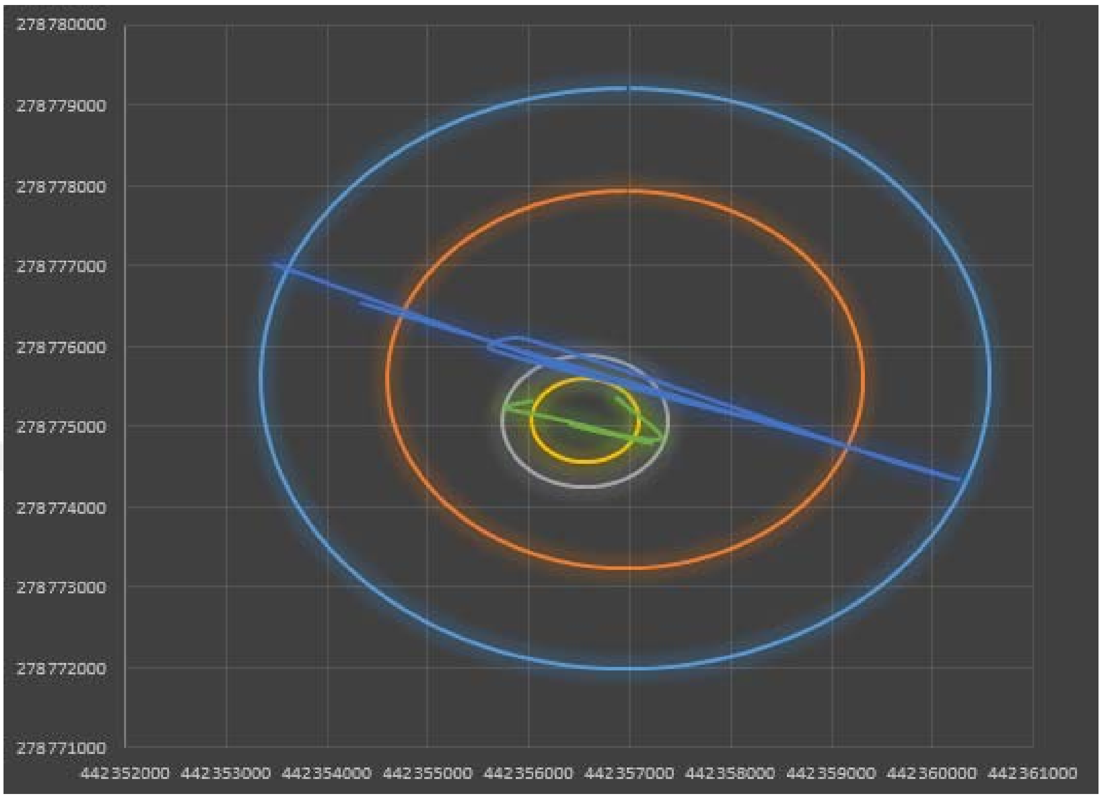


Fig. 5.2: Stability Errors of Raw GPS Coordinates Versus Corrected Coordinates with the First Method of Calculated the Reference Value.

As can be observed on Table.5.2. and Fig.5.2., raw GPS data correction with calculated reference GPS coordinate method reduces the precision and stability error of the measurements in a continuous process. As it is shown on Table 5.2. this method and devices are observed to be able to reduce the maximum deviation stability error by 77.30% around X-Y plane and reduce the minimum deviaton stability error by 77.72% around X-Y plane. With this method, accuracy error reduction couldn't be observed as the actual coordinate is not known.

Raw and calculated measurments and mean, median, maximum and minimum values are visualised on Fig.5.3. below.

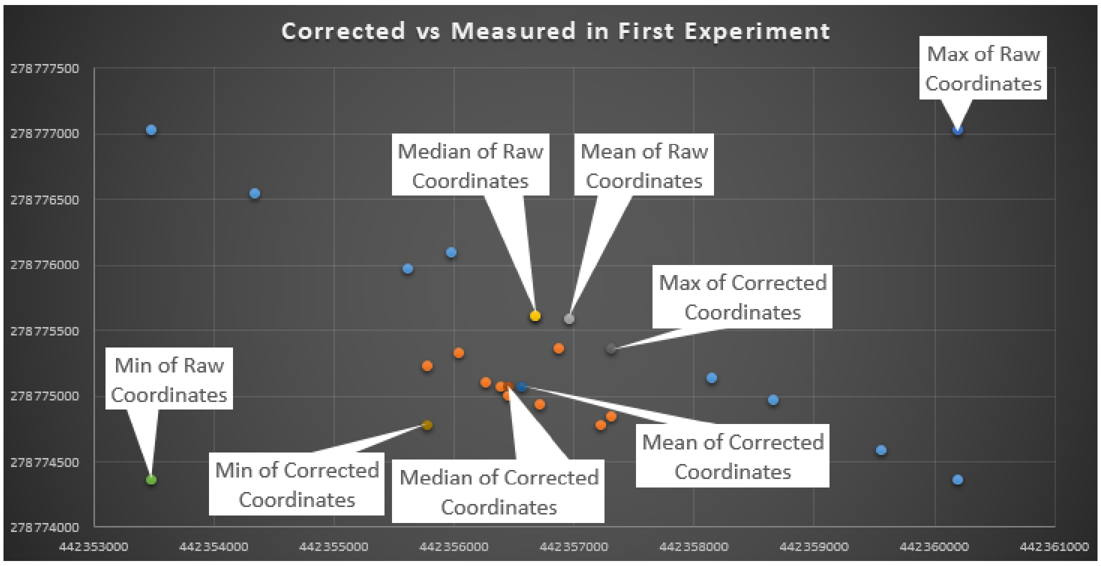


Fig. 5.3: Graphics of Raw GPS Measurements, Calculated GPS Coordinates, Mean Values, Median Values, Minimum Values and Maximum Values Obtained with the First Method of Calculated the Reference Value.

5.2 Results of Rover Device with a Known Reference Value

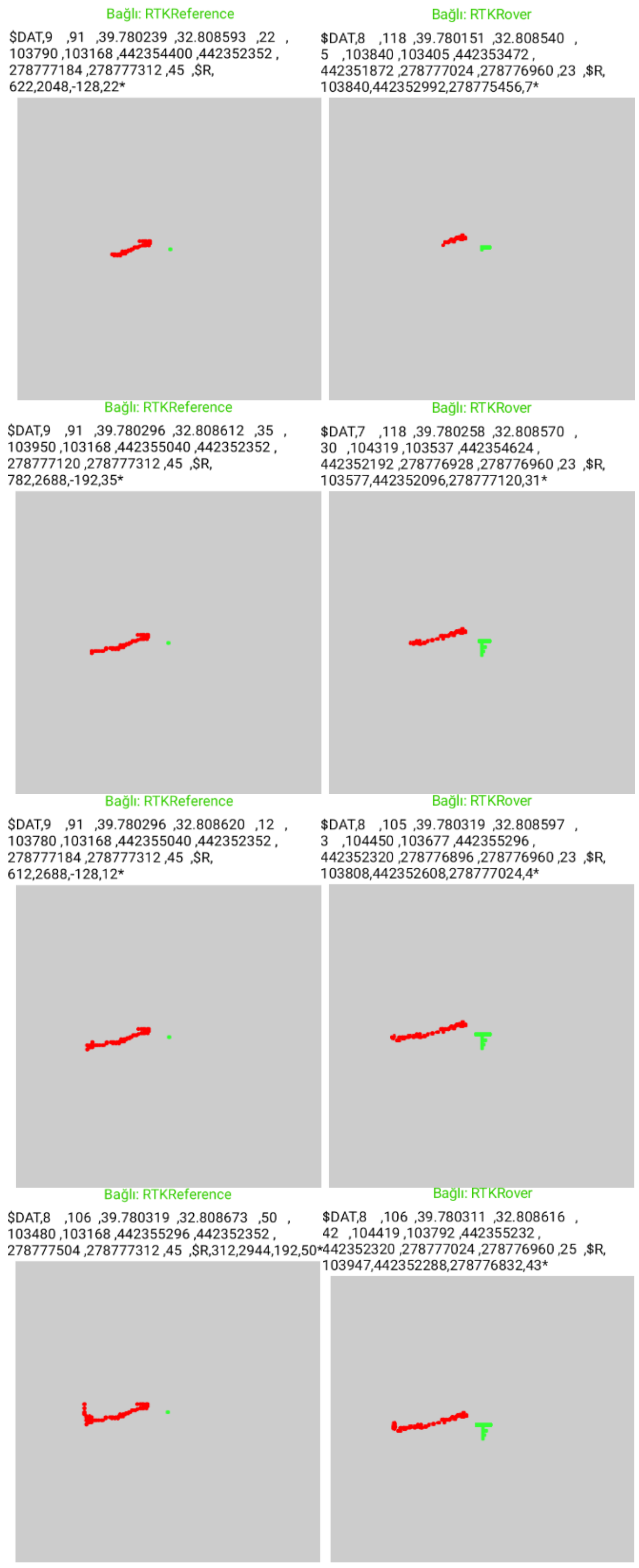
For the second experiment, reference device is set on a fixed point and its known actual coordinates are entered manually by smart phone Android application. After reference device is set, it started to broadcast the difference between the raw GPS measurements and the actual point that entered manually. Then the rover device is set and fixed on another point to observe the raw readings and the corrected coordinates via the broadcasts read.

Reference device first gets the raw GPS measurements. Then it starts to calculate the average of last 25 GPS readings one per second.

Then the reference device calculates the difference between the manually entered actual point and the last 25 seconds average and broadcasts it as the correction value.

Rover device, again calculates the average of its own raw GPS readings every 25 seconds. Then substracts the broadcasted correction values from the average of 25 GPS readings in order to find the corrected coordinates.

Fig.5.4.a. and Fig.5.4.b shows the comparison between the rover and reference devices’ data with 7 screenshots of user interfaces.


Bağlı: RTKReference

```
$DAT,9   ,91   ,39.780239 ,32.808593  ,22   ,
103790 ,103168 ,442354400 ,442352352 ,
278777184 ,278777312 ,45  ,$R,
622,2048,-128,22*
```

Bağlı: RTKRover

```
$DAT,8   ,118 ,39.780151 ,32.808540  ,
5   ,103840 ,103405 ,442353472 ,
442351872 ,278777024 ,278776960 ,23  ,$R,
103840,442352992,278775456,7*
```

Bağlı: RTKReference

```
$DAT,9   ,91   ,39.780296 ,32.808612  ,35   ,
103950 ,103168 ,442355040 ,442352352 ,
278777120 ,278777312 ,45  ,$R,
782,2688,-192,35*
```

Bağlı: RTKRover

```
$DAT,7   ,118 ,39.780258 ,32.808570  ,
30  ,104319 ,103537 ,442354624 ,
442352192 ,278776928 ,278776960 ,23  ,$R,
103577,442352096,278777120,31*
```

Bağlı: RTKReference

```
$DAT,9   ,91   ,39.780296 ,32.808620  ,12   ,
103780 ,103168 ,442355040 ,442352352 ,
278777184 ,278777312 ,45  ,$R,
612,2688,-128,12*
```

Bağlı: RTKRover

```
$DAT,8   ,105 ,39.780319 ,32.808597  ,
3   ,104450 ,103677 ,442355296 ,
442352320 ,278776896 ,278776960 ,23  ,$R,
103808,442352608,278777024,4*
```

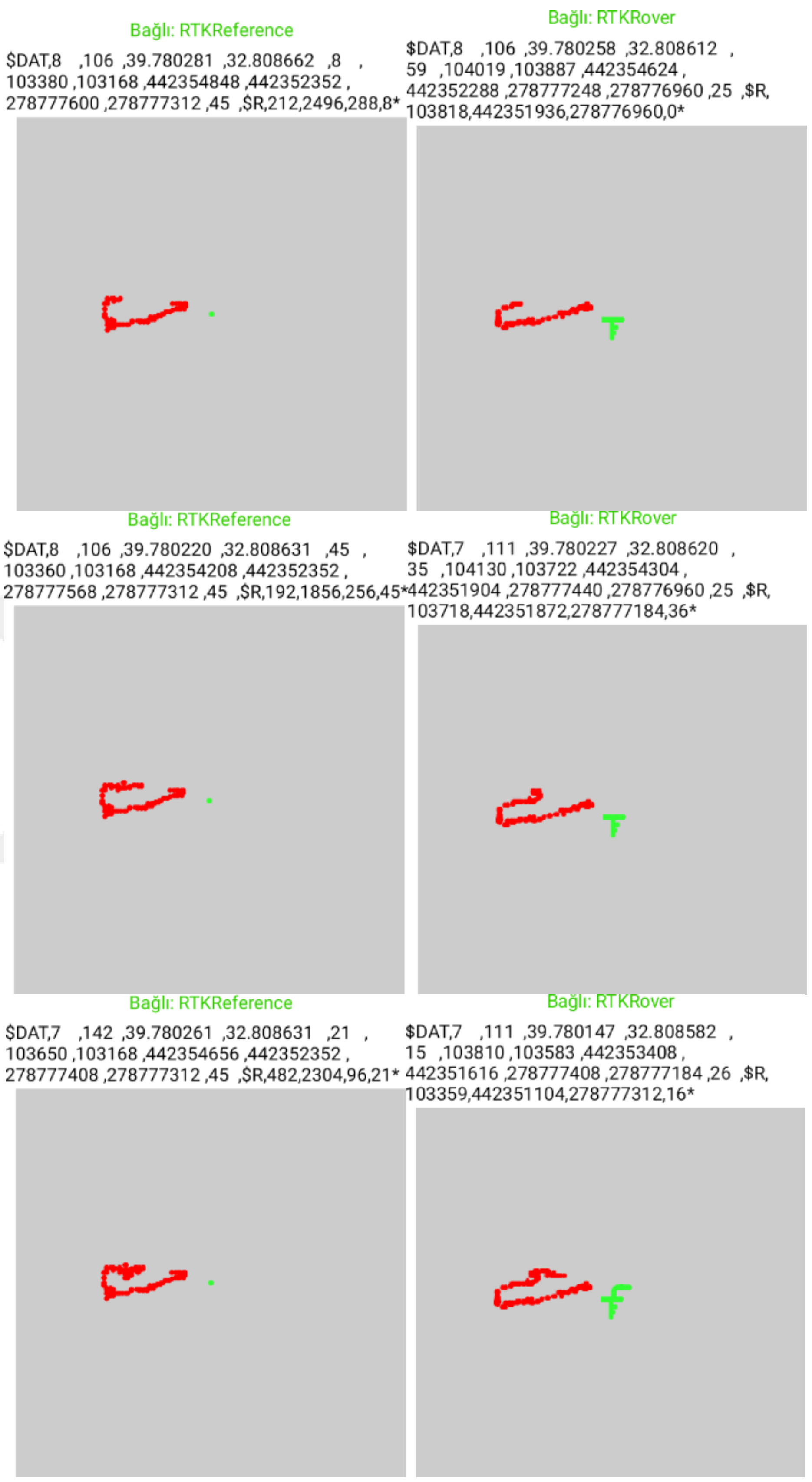
Bağlı: RTKReference

```
$DAT,8   ,106 ,39.780319 ,32.808673  ,50   ,
103480 ,103168 ,442355296 ,442352352 ,
278777504 ,278777312 ,45  ,$R,312,2944,192,50*
```

Bağlı: RTKRover

```
$DAT,8   ,106 ,39.780311 ,32.808616  ,
42  ,104419 ,103792 ,442355232 ,
442352320 ,278777024 ,278776960 ,25  ,$R,
103947,442352288,278776832,43*
```

Fig. 5.4.a: Screenshots 1st to 4th of the Smart Phone User Interface for Both of the Devices for a Known Point.



Bağlı: RTKReference

```
$DAT,8   ,106 ,39.780281 ,32.808662  ,8   ,
103380 ,103168 ,442354848 ,442352352 ,
278777600 ,278777312 ,45  ,$R,212,2496,288,8*
```

Bağlı: RTKRover

```
$DAT,8   ,106 ,39.780258 ,32.808612  ,
59  ,104019 ,103887 ,442354624 ,
442352288 ,278777248 ,278776960 ,25  ,$R,
103818,442351936,278776960,0*
```

Bağlı: RTKReference

```
$DAT,8   ,106 ,39.780220 ,32.808631  ,45  ,
103360 ,103168 ,442354208 ,442352352 ,
278777568 ,278777312 ,45  ,$R,192,1856,256,45*
```

Bağlı: RTKRover

```
$DAT,7   ,111 ,39.780227 ,32.808620  ,
35  ,104130 ,103722 ,442354304 ,
442351904 ,278777440 ,278776960 ,25  ,$R,
103718,442351872,278777184,36*
```

Bağlı: RTKReference

```
$DAT,7   ,142 ,39.780261 ,32.808631  ,21  ,
103650 ,103168 ,442354656 ,442352352 ,
278777408 ,278777312 ,45  ,$R,482,2304,96,21*
```

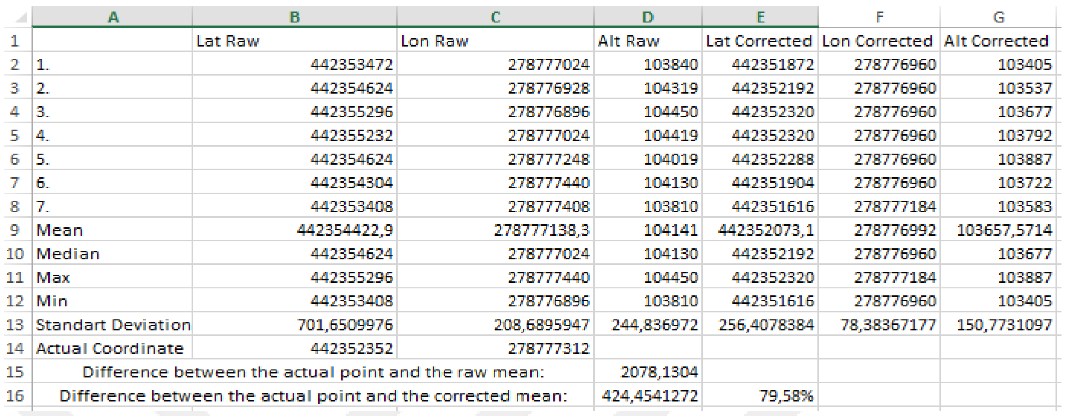
Bağlı: RTKRover

```
$DAT,7   ,111 ,39.780147 ,32.808582  ,
15  ,103810 ,103583 ,442353408 ,
442351616 ,278777408 ,278777184 ,26  ,$R,
103359,442351104,278777312,16*
```

Fig. 5.4.b: Screenshots 5th to 7th of the Smart Phone User Interface for Both of the Devices for a Known Point.

After those experiments, the rover device's raw and corrected measurements were processed in Microsot Excel in order to calculate mean, median, max, min and standard deviation values as shown on Table 5.3.

Table 5.3: Rover Device's Raw and Corrected Coordinates in Manuel Mode.

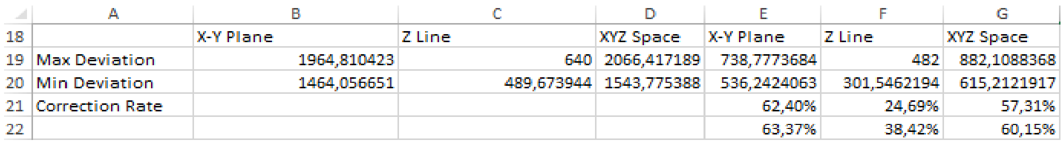


	A	B	C	D	E	F	G
1		Lat Raw	Lon Raw	Alt Raw	Lat Corrected	Lon Corrected	Alt Corrected
2	1.	442353472	278777024	103840	442351872	278776960	103405
3	2.	442354624	278776928	104319	442352192	278776960	103537
4	3.	442355296	278776896	104450	442352320	278776960	103677
5	4.	442355232	278777024	104419	442352320	278776960	103792
6	5.	442354624	278777248	104019	442352288	278776960	103887
7	6.	442354304	278777440	104130	442351904	278776960	103722
8	7.	442353408	278777408	103810	442351616	278777184	103583
9	Mean	442354422,9	278777138,3	104141	442352073,1	278776992	103657,5714
10	Median	442354624	278777024	104130	442352192	278776960	103677
11	Max	442355296	278777440	104450	442352320	278777184	103887
12	Min	442353408	278776896	103810	442351616	278776960	103405
13	Standart Deviation	701,6509976	208,6895947	244,836972	256,4078384	78,38367177	150,7731097
14	Actual Coordinate	442352352	278777312				
15	Difference between the actual point and the raw mean:			2078,1304			
16	Difference between the actual point and the corrected mean:			424,4541272	79,58%		

As it is observed on Table 5.3. the average mean of raw GPS coordinates is (lat: 442354422.9cms; lon: 278777198.3cms) and the standard deviation for raw coordinates is (lat: 701.7cms; lon: 208.7cms) while the mean of corrected coordinates is (lat: 442352073.1cms; lon: 278776992cms) and standart deviation for corrected coordinates is (lat: 256.4cms; lon: 78.4cms).

On Table 5.4. the rover device's maximum and minimum deviations about raw and corrected coordinates are shown.

Table 5.4: Rover Device Measurement Deviations and Correction Rate in Manuel Mode.



	A	B	C	D	E	F	G
18		X-Y Plane	Z Line	XYZ Space	X-Y Plane	Z Line	XYZ Space
19	Max Deviation	1964,810423	640	2066,417189	738,7773684	482	882,1088368
20	Min Deviation	1464,056651	489,673944	1543,775388	536,2424063	301,5462194	615,2121917
21	Correction Rate				62,40%	24,69%	57,31%
22					63,37%	38,42%	60,15%

As observed on Table 5.3. difference between 2 points on X-Y plane is calculated by adding squares of the difference between latitudes and longitudes of 2 points and taking square root of this addition. With this method, the difference between the actual point and the raw values' mean, and the difference between the actual point and the calculated values' mean are calculated and visualised on Fig.5.5. So, the accuracy error values are obtained with these calculations. Raw readings' accuracy is error is 2078.1 cms while the calculated GPS result's accuracy error is 424.5 cms. So, error reduction of accuracy of the system is 79.58%.

Fig. 5.5. below shows the stability and accuracy errors compared to each other.

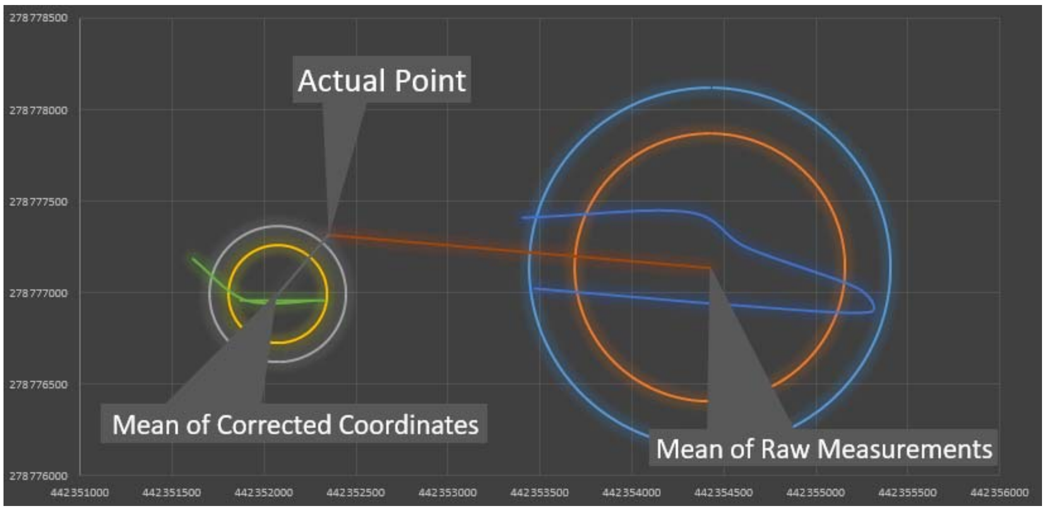


Fig. 5.5: Stability and Accuracy Errors of Raw GPS Coordinates Versus Corrected Coordinates with the Second Method with a Known Reference Value

As can be observed on Table 5.3. Table 5.4. and Fig.5.5. accuracy error is reduced by 79.58% from 2078 cms to 424 cms. Stability error is reduced by 63.37% from 1965 cms to 739 cms. And the measurements are more precise because of the filtering function that calculates a filtered results of 25 measurements.

Actual, raw and calculated measurments and mean, median, maximum and minimum values are visualised on Fig.5.6. below.

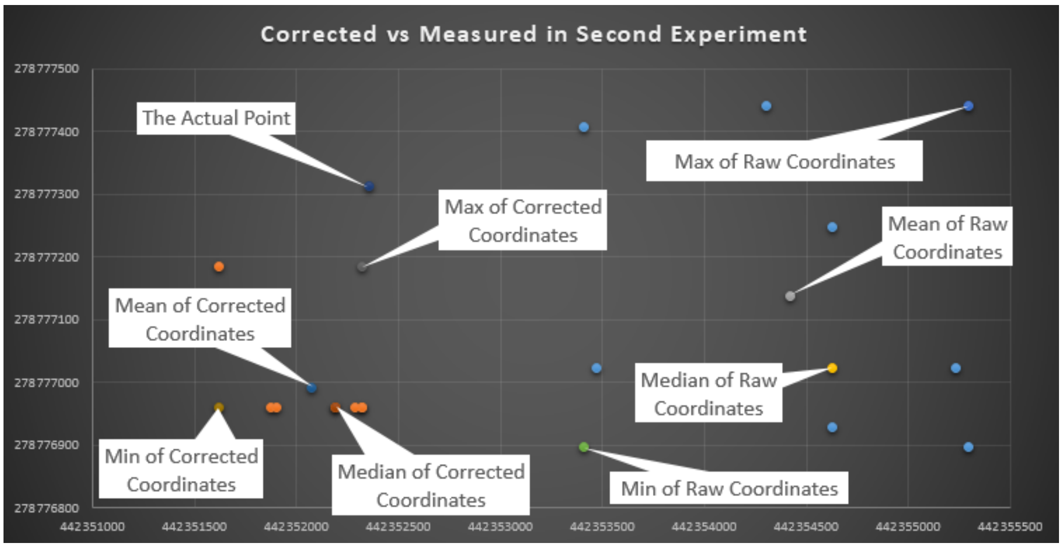


Fig. 5.6: Graphics of Raw GPS Measurements, Actual Value, Calculated GPS Coordinates, Mean Values, Median Values, Minimum Values and Maximum Values Obtained with the Second Method with a Known Reference Value.

5.3 Overall Results of the Experiments

It is observed that, both methods on these paired devices provided around 80% stability and accuracy reduction for raw GPS readings which is a successful correction for cheap L1 band GPS modules. The process reduced 20 meters of error to around 4 meters which would correspond a reduction from 2.5 meters to 50 cms or 1 meters to 20 cms on a strong signal environment. On the other hand, the precision is increased from around 24~36 cms to 1~2 cms with the average filtering algorithm.

CHAPTER SIX

CONCLUSIONS

In the market, D-GPS receivers with RTK functionality are sold at least around 6.500 $s which are able to provide coordinates with a 3 cms of accuracy [nivoteknik, 2018].

After purchasing those devices, it is required to sign up for a correction online broadcasting network which costs around 200$s per year. 3 cms of accuracy is required for many cases like bridge constructions, building construcitons and road constructions. Other usages in forestry, mining, dam construction, archaeological site construction and neat line calculations require lower accuracies.

So, the result of the study shows that the new method with 2 paired devices to provide RTK corrections is convenient for these kind of rough calculations. This system with 2 devices cost around 100$s per device so it will be a good alternative for most cases in mapping sector as it can provide 20 to 50 cms of accuracy according to the progress of measurement time.

REFERENCES

[Diggelen, 1997] Dr. Diggelen, F., V., 1997, "GPS and GPS+GLONASS RTK", New Product Descriptions, Ashtech Inc., California

[esa, 2018] esa, SBAS Systems, Retrieved April 30, 2018, from https://gssc.esa.int/navipedia/index.php/SBAS_Systems

[europe, 2018] europe, What is GNSS?, Retrieved May 5, 2018, from https://www.gsa.europa.eu/european-gnss/what-gnss

[europe, 2018] europe, What is SBAS, Retrieved June 13, 2018, from https://www.gsa.europa.eu/european-gnss/what-gnss/what-sba

[gpsags, 2018] gpsags, RTK Network FAQ's, Retrieved June 10, 2018, from http://www.gpsags.com/resources/rtk-faqs

[Hada, 2000] Hada, H., Sunahara, H., Uehara, K. et al., 2000, "DGPS and RTK Positioning Using the Internet", GPS Solutions, 4: 34. https://doi.org/10.1007/PL00012826, Nara Institue of Science Technology, Japan

[insidegnss, 2018] insidegnss, Single- versus Dual-Frequency Precise Point Positioning, Retrieved June 20, 2018, from http://insidegnss.com/single-versus-dual-frequency-precise-point-positioning/

[Lachapelle, et.al., 2000] Lachapelle, G., Alves, P., Luiz, P., F., Cannon, M., E., 2000, DGPS RTK Positioning Using a Reference Network, University of Calgary, Alberta, Canada

[Lui, 2004] Liu, G., C., 2003, "GPS RTK positioning via Internet-based 3G CDMA2000/1X wireless technology", GPS Solutions, 7: 222. https://doi.org/10.1007/s10291-003-0078-y, Engineering and Construction Department, Burnaby, Canada

[nivoteknik, 2018] nivoteknik, ÜRÜNLERİMİZ, Retrieved June 20, 2018, from http://www.nivoteknik.com.tr/sokkia-grx-1-gnss-cors-rtk-topcon-rtk-uyumlu/

[nps, 2018] nps, GPS Accuracy Levels, Retrieved May 18, 2018, from http://www.oc.nps.edu/oc2902w/gps/gpsacc.html

[racelogic, 2018] racelogic, How does DGPS (Differential GPS) work?, Retrieved June 12, 2018, from https://racelogic.support/01VBOX_Automotive/01General_Information/Knowledge_Base/How_does_DGPS_(Differential_GPS)_work%3F

[Vollath, et.al., 2000] Vollath, U., Buecherl, A., Landau, H., Pagels, C., Wagner, B., 2000, "Multi-base RTK positioning using virtual reference stations", In Proceedings of ION GPS (Vol. 95, pp. 123-131), Spectra Precision Terrasat GmbH

[wikipedia, 1, 2018] wikipedia, Global Positioning System, Retrieved May 22, 2018, from http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvR2xvYmFsX1Bvc2l0aW9uaW5nX1N5c3RlbQ

[wikipedia, 2, 2018] wikipedia, Assisted GPS, Retrieved June 18, 2018, from http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQXNzaXN0ZWRfR1BT

[wikipedia, 3, 2018] wikipedia, Simultaneous GPS, Retrieved June 8, 2018, from http://www.wiki-zero.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvUy1HUFM

[Wübbena, et.al., 1, 2014] Wübbena, G., Schmitz, M., Menge, F., Seeber, G., Völksen, C., 2014, A New Approach for Field Calibration of Absolute GPS Antenna Phase Center Variations, Geo++ Gesellschaft für satellitengestützte geodätische und navigatorische Technologien mbH, Garbsen, Germany

[Wübbena, et.al., 2, 2001] Wübbena, G., Bagge, A., Schmitz, M., 2001, "Network-based techniques for RTK applications", In GPS Symposium, GPS JIN (pp. 14-16), Gesellschaft für satellitengestützte geodätische und navigatorische Technologien mbH, Garbsen, Germany

APPENDICES

APPENDIX-A: Definitons of Current Methods and Technologies

GPS

"*The GPS concept is based on time and the known position of GPS specialized satellites. The satellites carry very stable atomic clocks that are synchronized with one another and with the ground clocks. Any drift from true time maintained on the ground is corrected daily. In the same manner, the satellite locations are known with great precision. GPS receivers have clocks as well, but they are less stable and less precise. GPS satellites continuously transmit data about their current time and position. A GPS receiver monitors multiple satellites and solves equations to determine the precise position of the receiver and its deviation from true time. At a minimum, four satellites must be in view of the receiver for it to compute four unknown quantities (three position coordinates and clock deviation from satellite time).*" [wikipedia, 1, 2018]

A-GPS and S-GPS

"*The data rate of the satellite signal is only 50 bit/s, so downloading orbital information like ephemerides and the almanac directly from satellites typically takes a long time, and if the satellite signals are lost during the acquisition of this information, it is discarded and the standalone system has to start from scratch. In A-GPS, the network operator deploys an A-GPS server, a cache server for GPS data. These A-GPS servers download the orbital information from the satellite and store it in the database. An A-GPS-capable device can connect to these servers and download this information using mobile-network radio bearers such as GSM, CDMA, WCDMA, LTE or even using other radio bearers such as Wi-Fi.*" [wikipedia, 2, 2018]

“*As the name implies, Simultaneous GPS allows a cellphone to receive both GPS and voice data at the same time, which improves sensitivity and allows service providers to offer location-based services.*” [wikipedia, 3, 2018]

SBAS (WAAS, MSAS, EGNOS)

“*SBAS improves the accuracy and reliability of GNSS information by correcting signal measurement errors and by providing information about the accuracy, integrity, continuity and availability of its signals. SBAS uses GNSS measurements taken by accurately located reference stations deployed across an entire continent. All measured GNSS errors are transferred to a central computing centre, where differential corrections and integrity messages are calculated. These calculations are then broadcast over the covered area using geostationary satellites that serve as an augmentation, or overlay, to the original GNSS message.*” [europe, 2018]

“*From all the SBAS systems in the world, three are already operational (WAAS, MSAS, EGNOS), three are under implementation.*” [esa, 2018]

PPP, GNSS, DGPS and RTK

“*Precise point positioning (PPP) is a technique that can compute positions with a high accuracy anywhere on the globe using a single GNSS receiver. It relies on highly accurate satellite position and clock data that can be downloaded from the International GNSS Service (IGS) or obtained in real-time from a number of service providers, using either the Internet or satellite links. The best possible GNSS accuracy, a few centimeters or better, is obtained by using carrier phase measurements from dual-frequency receivers.*” [insidegnss, 2018]

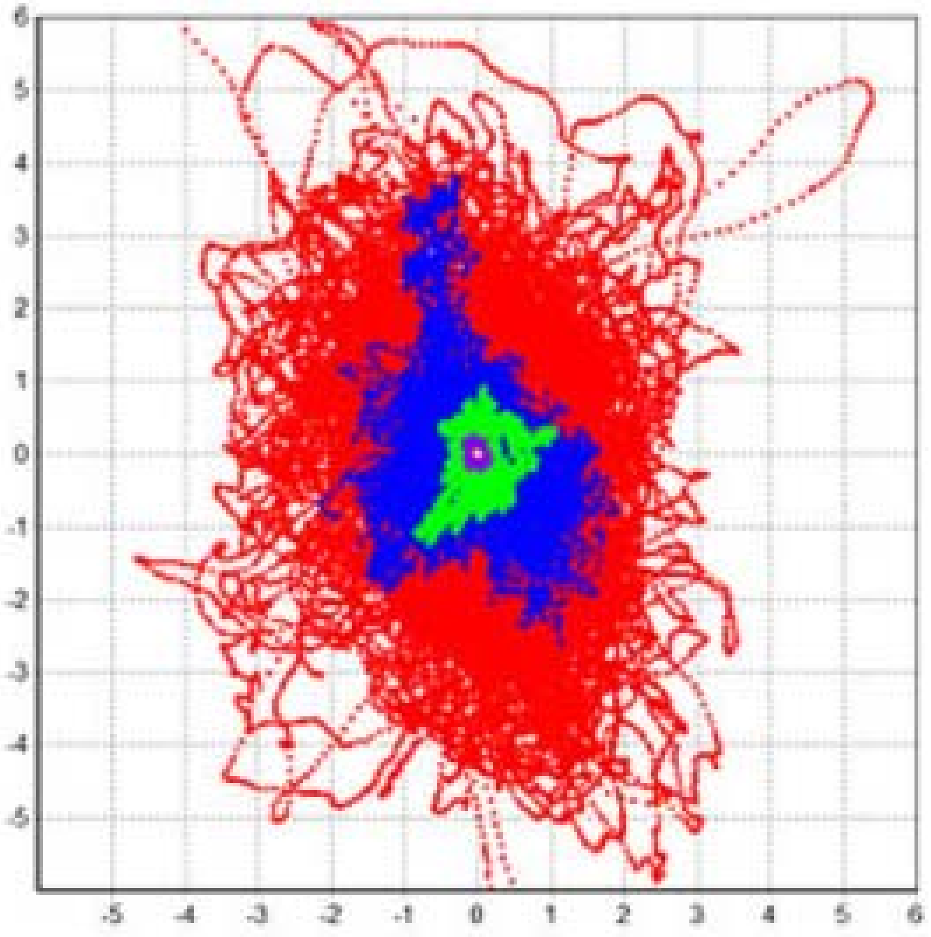
“*The post processed differential user typically uses L1 and L2 data from high end receivers. Special techniques (called Kinematics) make use of the phase to achieve very high accuracies. However they normally can only be used out to ranges of 25 to 50 km. Many manufacturers of high end receivers have vendor specific real time version of this. You almost always have to have the same type of receiver at each end and a dedicated communication link. Since Selective Availability was turned off, the main operational difference between the civilian and military systems (SPS vs PPS)* *is the availability of a L2 for real time removal of the ionosphere. Inexpensive receivers with only L1 still have to deal with the ionosphere that mainly affects the height. Even this difference is removed for the more expensive receivers that use complex techniques to track L2.*” [nps, 2018]

“*Global Navigation Satellite System (GNSS) refers to a constellation of satellites providing signals from space that transmit positioning and timing data to GNSS receivers. The receivers then use this data to determine location.*” [europe, 2018]

“*DGPS (Differential GPS) is essentially a system to provide positional corrections to GPS signals. DGPS uses a fixed, known position to adjust real time GPS signals to eliminate pseudorange errors.*” [racelogic, 2018]

“*With RTK, you need a base station placed on a known, surveyed point, and one or more mobile receivers within a ten kilometer range of your base station. The base station transmits corrections via radio to the mobile receivers in the field.*” [gpsags, 2018]

APPENDIX-B: Accuracy, Stability and Precision Comparison Between Used Systems



"*This 24 hour position scatter plot shows a commercial engine in red (Sat Nav's, Mobile Phones, etc.). an un-aided survey grade in blue (VBOX units), iaded by SBAS corrections in green, aided by a 20 cm base station in purple, and you can just make out an RTK 2 cm aided system in yellow!*" [racelogic, 2018]

APPENDIX-C: RTK Reference – Main Code:

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;
static const int RXPin1 = 12, TXPin1 = 13;
static const uint32_t GPSBaud1 = 9600;
String STR="", BTstring="", inString="", MSGtip="", BTlats, BTlons, BTalts;
double alt=0.0, kaltcm=0, klatcm=0, kloncm=0, BTlat, BTlon;
signed long hamlat=0, hamlon=0, kduzalt=0, kduzlat=0, kduzlon=0;
double ialt=0, ilat=0, ilon=0;
int deger=1, BTbagli=1, autobul=1, manuel=0, BTok=0, BTchar, ix=0, islemyap=0;
TinyGPSPlus gps;
SoftwareSerial rfd(RXPin1, TXPin1);
SoftwareSerial BT(RXPin, TXPin);
void setup()
{
  Serial.begin(9600);
  rfd.begin(GPSBaud1);
  BT.begin(GPSBaud);
  delay(5000);
}
void loop()
{
  if (BTbagli==1)
  {
  BT.listen();
  BT.print("$DAT,");
  printInt(gps.satellites.value(), gps.satellites.isValid(), 5);    BT.print(",");
  printInt(gps.hdop.value(), gps.hdop.isValid(), 5);                BT.print(",");
  printFloat(gps.location.lat(), gps.location.isValid(), 11, 6);    BT.print(",");
  printFloat(gps.location.lng(), gps.location.isValid(), 12, 6);    BT.print(",");
  printInt(gps.time.second(), 1, 5);                                BT.print(",");
  }
  if  (gps.hdop.value()<1000 && manuel==0){deger=gps.hdop.value()/1000;}
  else                                    {deger=1;}
  alt = gps.altitude.meters()*100;
  if (autobul==1) { kaltcm=(0.95+(0.05*deger))*kaltcm+0.05*(1-deger)*ialt; }
  if (BTbagli==1) {printInt(ialt,1,7);BT.print(",");
                   printInt(kaltcm, 1, 7);BT.print(",");}
    hamlat =
      (unsigned long)TinyGPSPlus::distanceBetween(
      gps.location.lat(),
      gps.location.lng(),
      0.0,
      gps.location.lng());
    hamlon =
      (unsigned long)TinyGPSPlus::distanceBetween(
      gps.location.lat(),
      gps.location.lng(),
      gps.location.lat(),
      0.0);
  if (BTbagli==1){printInt(hamlat, gps.location.isValid(),10);BT.print(",");}
  if (autobul==1){ klatcm=(0.95+(0.05*deger))*klatcm+0.05*(1-deger)*ilat;   }
  if (BTbagli==1){printInt(ilat, 1, 10);                      BT.print(",");}
  if (BTbagli==1){printInt(hamlon, gps.location.isValid(),10);BT.print(",");}
  if (autobul==1){ kloncm=(0.95+(0.05*deger))*kloncm+0.05*(1-deger)*ilon;   }
  if (BTbagli==1){printInt(ilon, 1, 10);                      BT.print(",");}
  if (islemyap==1)
  {
  kduzalt = ialt-kaltcm;
  kduzlat = ilat-klatcm;
  kduzlon = ilon-kloncm;
  }
  if (BTbagli==1){printInt(gps.failedChecksum(), true, 4);    BT.print(",");}

  STR = "$R," + (String)kduzalt + ",";
  STR = STR   + (String)kduzlat + ",";
  STR = STR   + (String)kduzlon + ",";
  STR = STR   + (String)gps.time.second() + "*";


  if (BTbagli==1)
  {
    Serial.println(STR);delay(1);
    rfd.listen();
    rfd.print(STR);     delay(1);
    BT.listen();
    BT.println(STR);
  }delay(1);
  if(ix==25&&((gps.time.second()>=30&&gps.time.second()<=36)
            ||(gps.time.second()>=0&&gps.time.second()<=6)))
  {ix=0; ialt=0; ilat=0; ilon=0;}
  if (gps.location.isValid()==1 && ix<25)
  {
    ialt+=alt/25;
    ilat+=hamlat/25;
    ilon+=hamlon/25;
    ix++;
    islemyap=0;
  }
  if (ix==25){islemyap=1;}
  smartDelay(1000);
}
static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    BT.listen();
    BToku();
    delay(1);
    while(Serial.available()>0)
      gps.encode(Serial.read());
  } while (millis() - start < ms);
}
static void printFloat(float val, bool valid, int len, int prec)
{
  if (!valid)
  {
    while (len-- > 1)
      BT.print('*');
    BT.print(' ');
  }
  else
  {
    BT.print(val, prec);
    int vi = abs((int)val);
    int flen = prec + (val < 0.0 ? 2 : 1); // . and -
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
    for (int i=flen; i<len; ++i)
      BT.print(' ');
  }
  smartDelay(0);
}
static void printInt(unsigned long val, bool valid, int len)
{
  char sz[32] = "*****************";
  if (valid)
    sprintf(sz, "%ld", val);
  sz[len] = 0;
  for (int i=strlen(sz); i<len; ++i)
    sz[i] = ' ';
  if (len > 0)
    sz[len-1] = ' ';
  BT.print(sz);
  smartDelay(0);
}
```

```
static void printDateTime(TinyGPSDate &d, TinyGPSTime &t)
{
  if (!d.isValid())
  {
    BT.print(F("********** "));
  }
  else
  {
    char sz[32];
    sprintf(sz, "%02d/%02d/%02d ", d.month(), d.day(), d.year());
    BT.print(sz);
  }
  if (!t.isValid())
  {
    BT.print(F("******** "));
  }
  else
  {
    char sz[32];
    sprintf(sz, "%02d:%02d:%02d ", t.hour(), t.minute(), t.second());
    BT.print(sz);
  }
  printInt(d.age(), d.isValid(), 5);
  smartDelay(0);
}
static void printStr(const char *str, int len)
{
  int slen = strlen(str);
  for (int i=0; i<len; ++i)
    BT.print(i<slen ? str[i] : ' ');
  smartDelay(0);
}
```

APPENDIX-D: RTK Reference – Bluetooth Serial String Reader and Parser Function:

```
void BToku()
{
  int hexs1, hexs2;
  int inChar, yildiz;
  char floatbuf[256];
  double f;
  unsigned char hexc1;
  unsigned char hexc2;
  int virgul,virgul2,basla,i;
  int sonlandi=0;
  int timed=0;
  inString = "";
  basla=0;
  if (BT.available()>0)
  {
    while (BT.available()>0 || basla==1)
    {
      inChar = BT.read();
      if (inChar!=-1)
      {
       Serial.print((char)inChar);
       if(inChar=='$')  {basla=1;}
       if (basla==1)     {inString += (char)inChar;}
       if (inChar=='*')  {break;}
      }
    }
    if (basla==1)
    {
      yildiz=inString.indexOf('*');
      virgul=inString.indexOf(',');
      if (virgul==-1)
      {
        MSGtip=inString.substring(1,yildiz);
        if (MSGtip=="A")  { autobul = 1;Serial.println("A"); }
        if (MSGtip=="B")  { BTbagli = 1;Serial.println("B"); }
      }
      else
      {
        MSGtip=inString.substring(1,virgul);
        for (i=1; virgul>0; i++)
        {
          virgul2=virgul+1;
          virgul = inString.indexOf(',',virgul2);
          if(MSGtip=="C")
         {autobul=0;Serial.println("C");
            switch(i){case1:
               BTlats=inString.substring(virgul2,virgul);BTlats.toCharArray(floatbuf,
               sizeof(floatbuf));BTlat=atof(floatbuf);Serial.println("btlat");break;
                                      case 2:
               BTlons=inString.substring(virgul2,virgul);BTlons.toCharArray(floatbuf,
               sizeof(floatbuf));BTlon=atof(floatbuf);Serial.println("btlon");break;
                                      case 3:
               BTalts=inString.substring(virgul2,yildiz);BTalts.toCharArray(floatbuf,
               sizeof(floatbuf));kaltcm=atof(floatbuf);Serial.println("btalt");break;
               }
          }
        }
        klatcm = (unsigned long)TinyGPSPlus::distanceBetween(BTlat,BTlon,0.0,BTlon);
        kloncm = (unsigned long) TinyGPSPlus::distanceBetween(BTlat,BTlon,BTlat,0.0);
      }
    }
  }
}
```

APPENDIX-E: RTK Rover – Main Code:

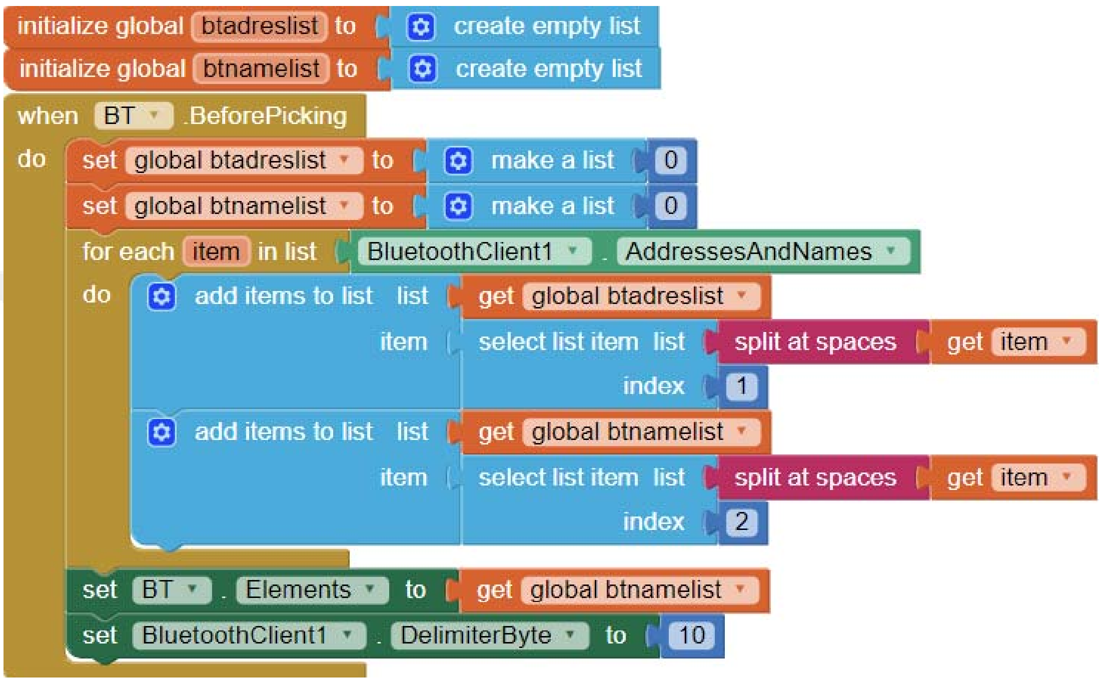
```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 9600;
static const int RXPin1 = 12, TXPin1 = 13;
static const uint32_t GPSBaud1 = 9600;
String STR="", BTstring="", inString="", MSGtip="", RFDlats, RFDlons, RFDalts,
RFDsecs;
double alt=0.0, kaltcm=0, RFDalt=0, klatcm=0, kloncm=0, RFDlat, RFDlon;
int    RFDsec=0, kduzsec=0;
double ialt=0, ilat=0, ilon=0;
int deger=1, BTbagli=1, autobul=1, manuel=0, BTok=0, BTchar, ix=0, islemyap=0,
ilkkez=0;
signed long kduzalt=0, kduzlat=0, kduzlon=0, latcm=0, loncm=0;
TinyGPSPlus gps;
SoftwareSerial rfd(RXPin1, TXPin1);
SoftwareSerial BT(RXPin, TXPin);
void setup()
{
  Serial.begin(9600);
  rfd.begin(GPSBaud1);
  BT.begin(GPSBaud);
  delay(5000);
}
void loop()
{
  if (BTbagli==1)
  {
  BT.listen();
  BT.print("$DAT,");
  printInt(gps.satellites.value(), gps.satellites.isValid(), 5);    BT.print(",");
  printInt(gps.hdop.value(), gps.hdop.isValid(), 5);                BT.print(",");
  printFloat(gps.location.lat(), gps.location.isValid(), 11, 6);    BT.print(",");
  printFloat(gps.location.lng(), gps.location.isValid(), 12, 6);    BT.print(",");
  printInt(gps.time.second(), 1, 5);                                BT.print(",");
  }
  if (gps.hdop.value()<1000 && manuel==0){deger=gps.hdop.value()/1000;}
  else                                  {deger=1;                    }
  alt = gps.altitude.meters()*100;
  if (BTbagli==1)
  {
    printInt(alt, 1, 7);                                            BT.print(",");
    printInt(kaltcm, 1, 7);                                         BT.print(",");
    Serial.print(alt); Serial.print(",");
  }
  latcm =
      (unsigned long)TinyGPSPlus::distanceBetween(
      gps.location.lat(),
      gps.location.lng(),
      0.0,
      gps.location.lng());
  loncm =
      (unsigned long)TinyGPSPlus::distanceBetween(
      gps.location.lat(),
      gps.location.lng(),
      gps.location.lat(),
      0.0);
  if (BTbagli==1){printInt(latcm, gps.location.isValid(),10);BT.print(",");
                  Serial.print(latcm);Serial.print(",");}
  if (BTbagli==1){printInt(klatcm, 1, 10);BT.print(",");}
  if (BTbagli==1){printInt(loncm, gps.location.isValid(),10);BT.print(",");
                  Serial.print(loncm);Serial.println(",");}
  if (BTbagli==1){printInt(kloncm, 1, 10);BT.print(",");}
  smartDelay(1000);
  if (BTbagli==1){printInt(gps.failedChecksum(), true, 4);BT.print(",");}
  STR = "$R," + (String)kduzalt + ",";
  STR = STR   + (String)kduzlat + ",";
  STR = STR   + (String)kduzlon + ",";
  STR = STR   + (String)gps.time.second() + "*";

  if (islemyap==1)
  {
    if (ilkkez==1)
    {
      kaltcm=kduzalt;
      klatcm=kduzlat;
      kloncm=kduzlon;
      ilkkez=2;
    }
    kaltcm=(0.75+(0.25*deger))*kaltcm+0.25*(1-deger)*kduzalt;
    klatcm=(0.75+(0.25*deger))*klatcm+0.25*(1-deger)*kduzlat;
    kloncm=(0.75+(0.25*deger))*kloncm+0.25*(1-deger)*kduzlon;
  }
  Serial.println(STR);
  if (BTbagli==1){BT.listen();BT.println(STR);}
  if ( ix==25 && ((gps.time.second() >= 30 && gps.time.second()<=36)
              ||(gps.time.second() >= 0 && gps.time.second()<=6)))
  {ix=0;ialt=0;ilat=0;ilon=0;}
  if (gps.location.isValid()==1 && ix<25)
  {
    ialt+=alt/25;
    ilat+=latcm/25;
    ilon+=loncm/25;
    ix++;
    islemyap=0;
  }
  if (ix==25)
  {
    if (ilkkez==0)
      ilkkez=1;
    islemyap=1;
  }
}
static void smartDelay(unsigned long ms)
{
  unsigned long start = millis();
  do
  {
    rfd.listen();
    RFDoku();

    while(Serial.available()>0)
      gps.encode(Serial.read());
  } while (millis() - start < ms);
}
static void printFloat(float val, bool valid, int len, int prec)
{
  if (!valid)
  {
    while (len-- > 1)
      BT.print('*');
    BT.print(' ');
  }
  else
  {
    BT.print(val, prec);
    int vi = abs((int)val);
    int flen = prec + (val < 0.0 ? 2 : 1); // . and -
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
    for (int i=flen; i<len; ++i)
      BT.print(' ');
  }
  smartDelay(0);
}
static void printInt(unsigned long val, bool valid, int len)
{
  char sz[32] = "*****************";
  if (valid)
```

```
    sprintf(sz, "%ld", val);
  sz[len] = 0;
  for (int i=strlen(sz); i<len; ++i)
    sz[i] = ' ';
  if (len > 0)
    sz[len-1] = ' ';
  BT.print(sz);
  smartDelay(0);
}
static void printDateTime(TinyGPSDate &d, TinyGPSTime &t)
{
  if (!d.isValid())
  {
    BT.print(F("********** "));
  }
  else
  {
    char sz[32];
    sprintf(sz, "%02d/%02d/%02d ", d.month(), d.day(), d.year());
    BT.print(sz);
  }
  if (!t.isValid())
  {
    BT.print(F("******** "));
  }
  else
  {
    char sz[32];
    sprintf(sz, "%02d:%02d:%02d ", t.hour(), t.minute(), t.second());
    BT.print(sz);
  }
  printInt(d.age(), d.isValid(), 5);
  smartDelay(0);
}
static void printStr(const char *str, int len)
{
  int slen = strlen(str);
  for (int i=0; i<len; ++i)
    BT.print(i<slen ? str[i] : ' ');
  smartDelay(0);
}
```

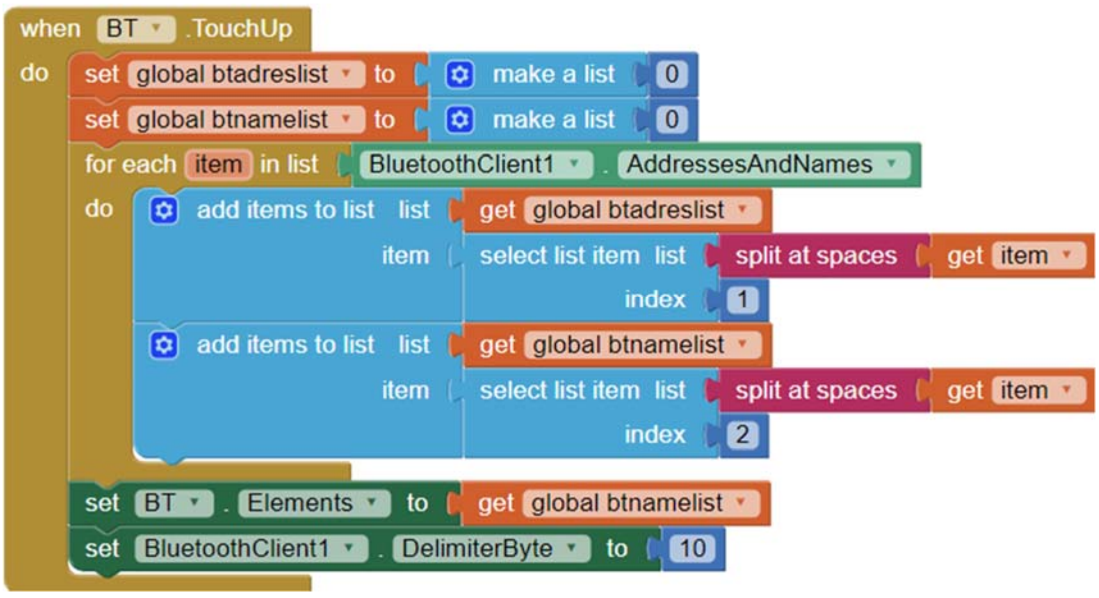
APPENDIX-F: RTK Rover– RF Data Link Serial String Reader and Parser Function:

```
void RFDoku()
{
  int hexs1, hexs2;
  int inChar, yildiz;
  char floatbuf[256];
  double f;
  unsigned char hexc1;
  unsigned char hexc2;
  int virgul,virgul2,basla,i;
  int sonlandi=0;
  int timed=0;
  inString = "";
  basla=0;
  if (rfd.available()>0)
  {
    while (rfd.available()>0 || basla==1)
    {
      inChar = rfd.read();
      if (inChar!=-1)
      {
        if (inChar=='$'){basla=1;}
        if (basla==1)   {inString += (char)inChar;}
        if (inChar=='*'){break;}
      }
    }
    if (basla==1)
    {
      yildiz=inString.indexOf('*');
      virgul=inString.indexOf(',');
      if (virgul>=0)
      {
        MSGtip = inString.substring(1,virgul);
        for (i=1; virgul>0; i++)
        {
          virgul2= virgul+1;
          virgul = inString.indexOf(',',virgul2);
          if (MSGtip=="R") {switch(i){case 1:
RFDalts=inString.substring(virgul2,virgul);RFDalts.toCharArray(floatbuf,
sizeof(floatbuf));RFDalt=atof(floatbuf);break;
                                      case 2:
RFDlats=inString.substring(virgul2,virgul);RFDlats.toCharArray(floatbuf,
sizeof(floatbuf));RFDlat=atof(floatbuf);break;
                                      case 3:
RFDlons=inString.substring(virgul2,virgul);RFDlons.toCharArray(floatbuf,
sizeof(floatbuf));RFDlon=atof(floatbuf);break;
                                      case 4:
RFDsecs=inString.substring(virgul2,yildiz);RFDsec=RFDsecs.toInt();break;}
                           }
        }
        if (islemyap==1)
        {
          kduzalt = ialt - RFDalt;
          kduzlat = ilat - RFDlat;
          kduzlon = ilon - RFDlon;
          kduzsec = gps.time.second() - RFDsec;
        }
      }
    }
  }
}
```

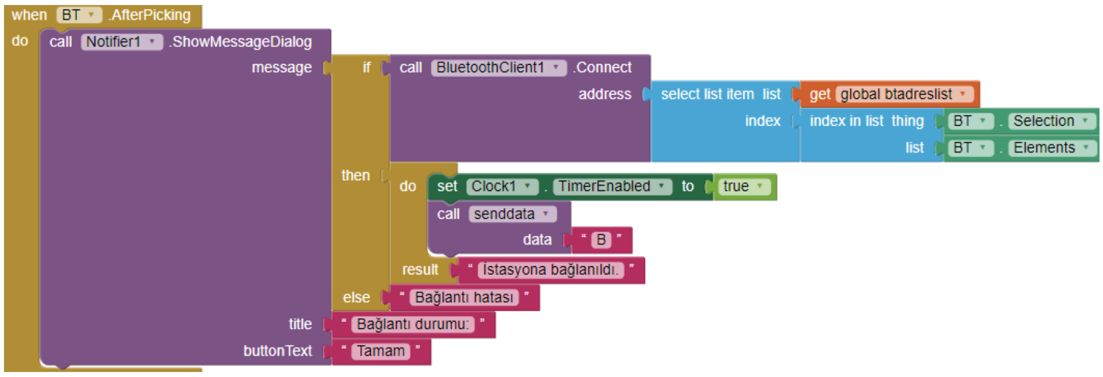
APPENDIX-G: RTK Devices' Android Application User Interface - App Inventor 2 Code Blocks:



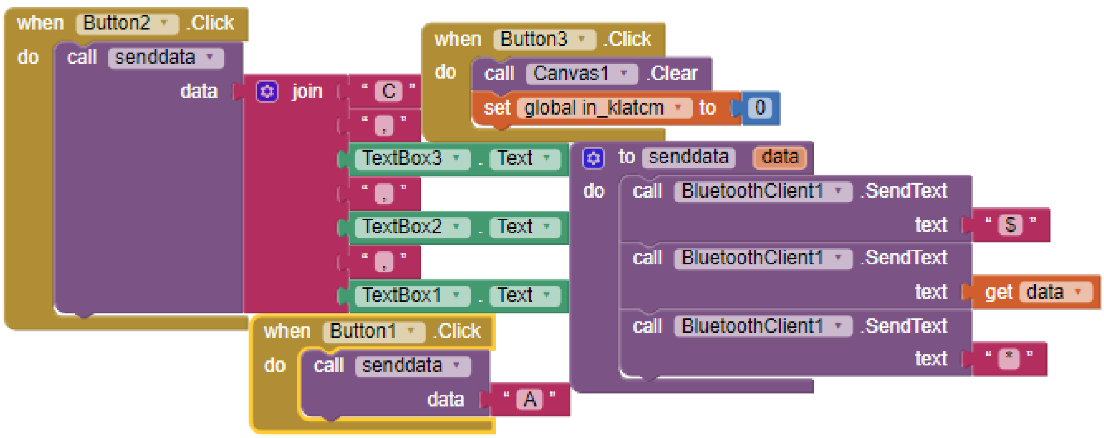
```
initialize global btadreslist to  create empty list
initialize global btnamelist to  create empty list
when BT .BeforePicking
do  set global btadreslist to  make a list  0
    set global btnamelist to  make a list  0
    for each item in list  BlueToothClient1 . AddressesAndNames
    do  add items to list  list  get global btadreslist
                           item  select list item  list  split at spaces  get item
                                                   index 1
        add items to list  list  get global btnamelist
                           item  select list item  list  split at spaces  get item
                                                   index 2
    set BT . Elements to  get global btnamelist
    set BluetoothClient1 . DelimiterByte to  10
```



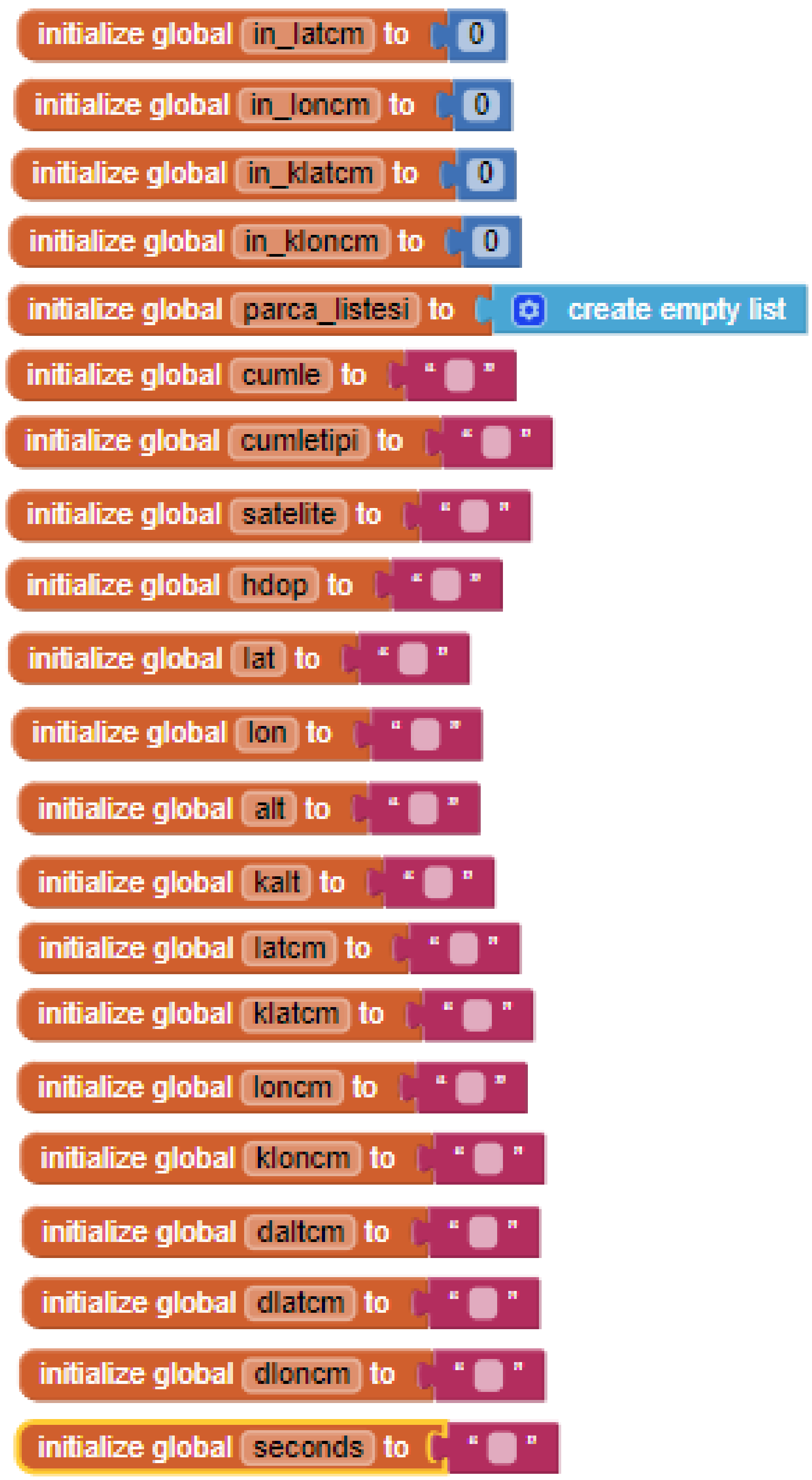
```
when BT .TouchUp
do  set global btadreslist to  make a list  0
    set global btnamelist to  make a list  0
    for each item in list  BlueToothClient1 . AddressesAndNames
    do  add items to list  list  get global btadreslist
                           item  select list item  list  split at spaces  get item
                                                   index 1
        add items to list  list  get global btnamelist
                           item  select list item  list  split at spaces  get item
                                                   index 2
    set BT . Elements to  get global btnamelist
    set BluetoothClient1 . DelimiterByte to  10
```



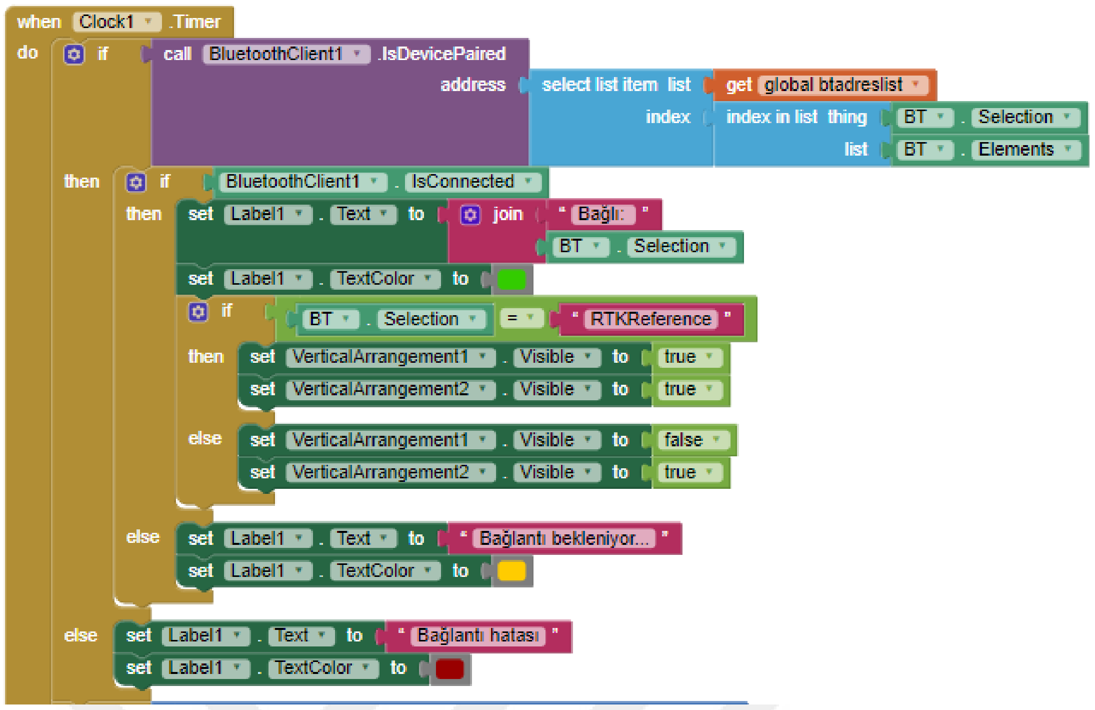
```
when BT .AfterPicking
do  call Notifier1 .ShowMessageDialog
        message  if  call BluetoothClient1 .Connect
                         address  select list item  list  get global btadreslist
                                                     index  index in list  thing  BT . Selection
                                                                           list   BT . Elements
                 then  do  set Clock1 . TimerEnabled to  true
                           call senddata
                                data  " B "
                       result  " İstasyona bağlanıldı. "
                 else  " Bağlantı hatası "
        title       " Bağlantı durumu: "
        buttonText  " Tamam "
```



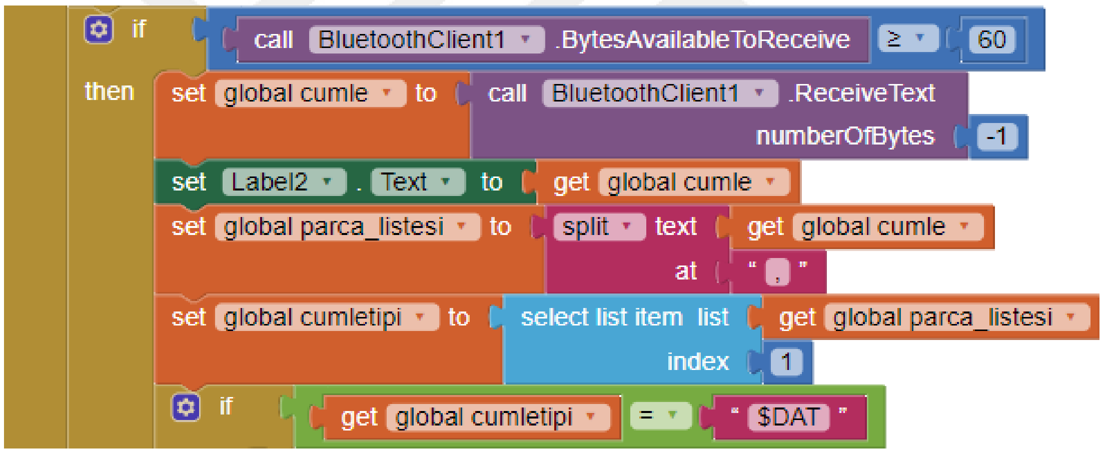
```
when Button2 .Click
do  call senddata
         data  join  " C "
                     " , "
                     TextBox3 . Text
                     " , "
                     TextBox2 . Text
                     " , "
                     TextBox1 . Text

when Button3 .Click
do  call Canvas1 .Clear
    set global in_klatcm to  0

to senddata  data
do  call BluetoothClient1 .SendText
         text  " S "
    call BluetoothClient1 .SendText
         text  get data
    call BluetoothClient1 .SendText
         text  " * "

when Button1 .Click
do  call senddata
         data  " A "
```



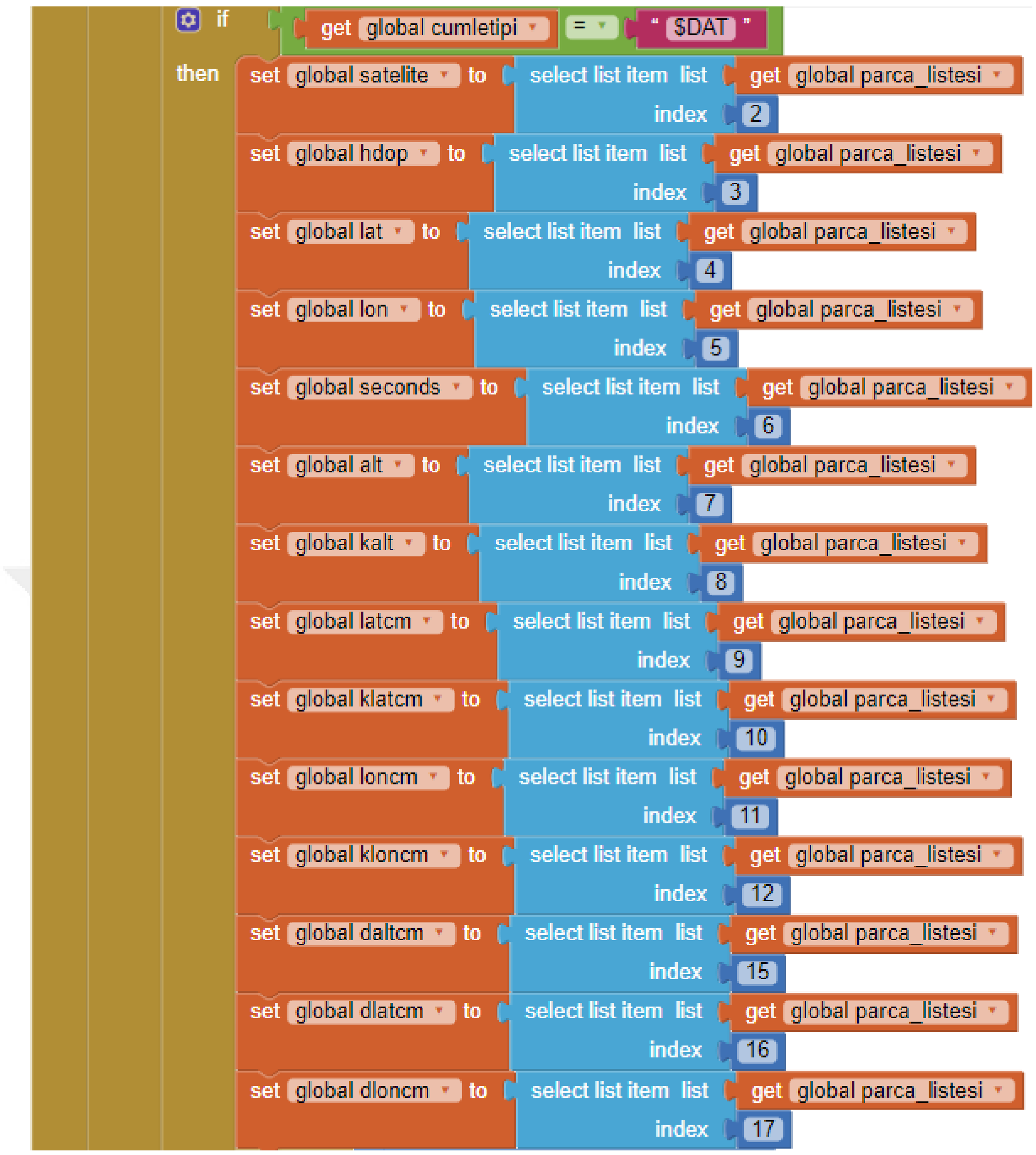
```
initialize global in_latcm to 0
initialize global in_loncm to 0
initialize global in_klatcm to 0
initialize global in_kloncm to 0
initialize global parca_listesi to create empty list
initialize global cumle to " "
initialize global cumletipi to " "
initialize global satelite to " "
initialize global hdop to " "
initialize global lat to " "
initialize global lon to " "
initialize global alt to " "
initialize global kalt to " "
initialize global latcm to " "
initialize global klatcm to " "
initialize global loncm to " "
initialize global kloncm to " "
initialize global daltcm to " "
initialize global dlatcm to " "
initialize global dloncm to " "
initialize global seconds to " "
```



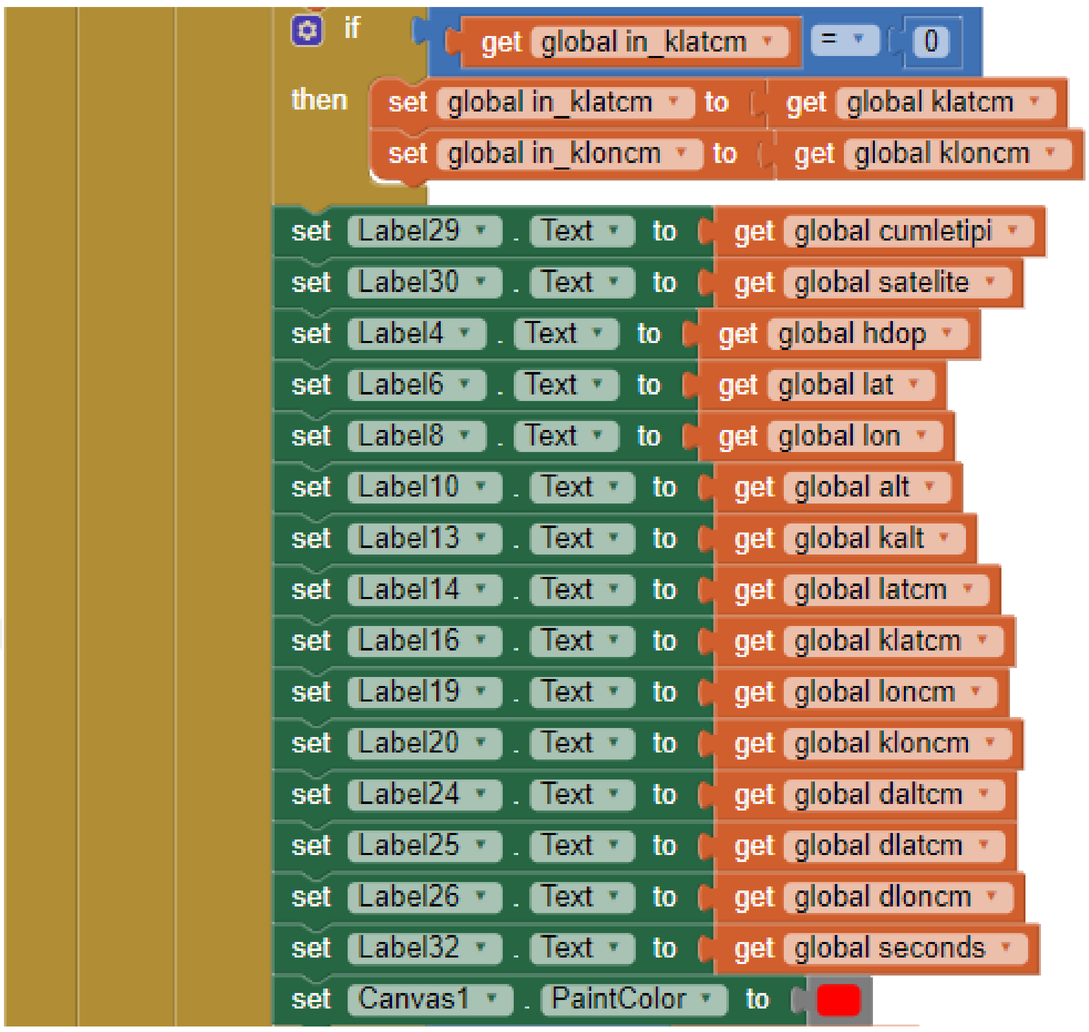
```
when Clock1 .Timer
do  if  call BluetoothClient1 .IsDevicePaired
            address  select list item  list  get global btadreslist
                                      index  index in list  thing  BT . Selection
                                                            list   BT . Elements
    then  if  BluetoothClient1 . IsConnected
          then  set Label1 . Text to  join  " Bağlı: "
                                            BT . Selection
                set Label1 . TextColor to 🟩
                if  BT . Selection  =  " RTKReference "
                then  set VerticalArrangement1 . Visible to true
                      set VerticalArrangement2 . Visible to true
                else  set VerticalArrangement1 . Visible to false
                      set VerticalArrangement2 . Visible to true
          else  set Label1 . Text to " Bağlantı bekleniyor... "
                set Label1 . TextColor to 🟨
    else  set Label1 . Text to " Bağlantı hatası "
          set Label1 . TextColor to 🟥
```



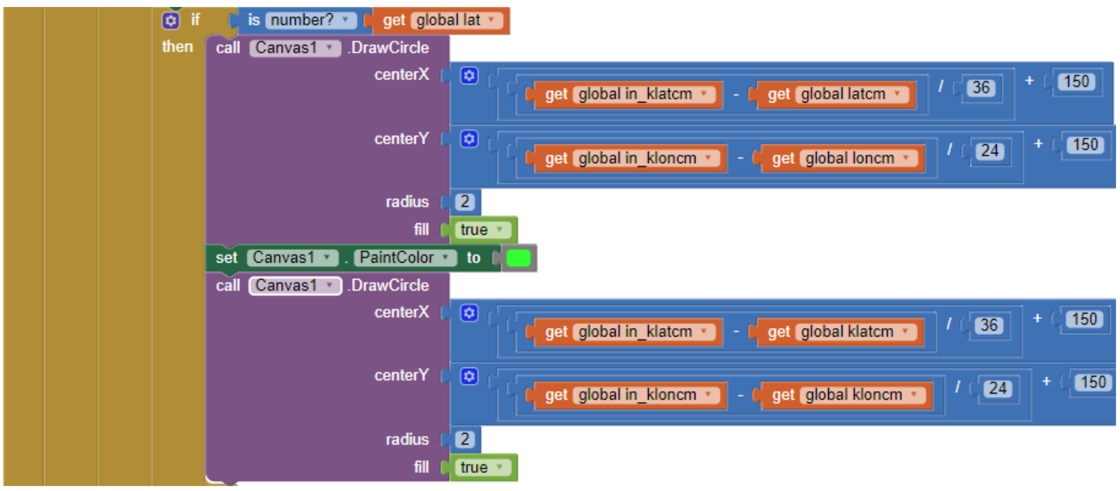
```
if  call BluetoothClient1 .BytesAvailableToReceive  ≥  60
then  set global cumle to  call BluetoothClient1 .ReceiveText
                                numberOfBytes  -1
      set Label2 . Text to  get global cumle
      set global parca_listesi to  split  text  get global cumle
                                          at    " , "
      set global cumletipi to  select list item  list  get global parca_listesi
                                                 index 1
      if  get global cumletipi  =  " $DAT "
```



```
if    get global cumletipi = " $DAT "
then  set global satelite to  select list item  list  get global parca_listesi
                                                index 2
      set global hdop to  select list item  list  get global parca_listesi
                                            index 3
      set global lat to  select list item  list  get global parca_listesi
                                           index 4
      set global lon to  select list item  list  get global parca_listesi
                                           index 5
      set global seconds to  select list item  list  get global parca_listesi
                                               index 6
      set global alt to  select list item  list  get global parca_listesi
                                           index 7
      set global kalt to  select list item  list  get global parca_listesi
                                            index 8
      set global latcm to  select list item  list  get global parca_listesi
                                             index 9
      set global klatcm to  select list item  list  get global parca_listesi
                                              index 10
      set global loncm to  select list item  list  get global parca_listesi
                                             index 11
      set global kloncm to  select list item  list  get global parca_listesi
                                              index 12
      set global daltcm to  select list item  list  get global parca_listesi
                                              index 15
      set global dlatcm to  select list item  list  get global parca_listesi
                                              index 16
      set global dloncm to  select list item  list  get global parca_listesi
                                              index 17
```



```
if   get global in_klatcm  =  0
then set global in_klatcm to  get global klatcm
     set global in_kloncm to  get global kloncm
set Label29 . Text to  get global cumletipi
set Label30 . Text to  get global satelite
set Label4 . Text to  get global hdop
set Label6 . Text to  get global lat
set Label8 . Text to  get global lon
set Label10 . Text to  get global alt
set Label13 . Text to  get global kalt
set Label14 . Text to  get global latcm
set Label16 . Text to  get global klatcm
set Label19 . Text to  get global loncm
set Label20 . Text to  get global kloncm
set Label24 . Text to  get global daltcm
set Label25 . Text to  get global dlatcm
set Label26 . Text to  get global dloncm
set Label32 . Text to  get global seconds
set Canvas1 . PaintColor to 🟥
```



```
if   is number?  get global lat
then call Canvas1 .DrawCircle
          centerX  get global in_klatcm - get global latcm / 36 + 150
          centerY  get global in_kloncm - get global loncm / 24 + 150
          radius  2
          fill  true
     set Canvas1 . PaintColor to 🟩
     call Canvas1 .DrawCircle
          centerX  get global in_klatcm - get global klatcm / 36 + 150
          centerY  get global in_kloncm - get global kloncm / 24 + 150
          radius  2
          fill  true
```

RESUME

PERSONAL INFORMATION

Surname, Name	:	KIYAK, Cemil Baki
Nationality	:	T.C.
Date and Place of Birth	:	1987, Ankara
Phone	:	+90 312 266 4664
E-mail	:	cemilbakikiyak@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
Master Program	THK University	2018
License	Atılım University	2011
High School	Ankara Atatürk Lisesi	2005

WORK EXPERIENCE

Year	Place	Enrolment
2013-2018	Devr-i Robotik R&D Co. Ltd.	General Manager

FOREIGN LANGUAGES

English

CERTIFICATES, COURSES, CONFERENCES, PUBLICATIONS

- TÜBİTAK (2004) Lise Öğrencileri Arası Proje Yarışması Elektronik Dalı Katılım: "Elektrik Dalgalanmalarının Vereceği Hasarın Engellenmesi: AC Gerilim Kontrol Rölesi"
- TÜBİTAK (2005) Orta Öğretim Öğrencileri Arası Araştırma Projeleri Yarışması Yerbilimi Dalı (İç Anadolu Bölgesi) Katılım Belgesi : "Paslanarak Yıkılan Binalara Son!"

- TÜBİTAK (2005) Orta Öğretim Öğrencileri Arası Araştırma Projeleri Yarışması Yerbilimi Dalı Final Yarışması Katılım Belgesi : "Paslanarak Yıkılan Binalara Son!"
- TÜBİTAK (2005) Orta Öğretim Öğrencileri Arası Araştırma Projeleri Yarışması Yerbilimi Dalı Final Yarışması (Türkiye) Birincilik Belgesi : "Paslanarak Yıkılan Binalara Son!"
- (Sofya 2005), Doğal Bilimler Alanında (Balkan Ülkeleri Arası) Pilot Birinci Bölgesel Genç Yetenekler Yarışması Katılım Belgesi : "No More Collapsing Buildings Because of Rust!"
- (Sofya 2005), Doğal Bilimler Alanında (Balkan Ülkeleri Arası) Pilot Birinci Bölgesel Genç Yetenekler Yarışması Üçüncülük Ödülü Belgesi : "No More Collapsing Buildings Because of Rust!"
- ATO (2005), Ankara'nın Altın Çocukları, Ankara Valiliği Milli Eğitim Müdürlüğü Başarı Belgesi
- Ankara Atatürk Lisesi (2005), TÜBİTAK Proje Yarışması Bölge Finalistliği Teşekkür Belgesi
- Atılım Üniversitesi (2006), "Akademik Başarıyı Arttırma ve Verimli Ders Çalışma Yöntemleri" Kurs Programı Katılım Belgesi
- Birinci MEB Robot Yarışması Katılım (2007)
- Kıyak, C.B., ÇESKO (2007), Fatih Üniversitesi, Üniversite Öğrencileri 2. Çevre Sorunları Konferansı, "Paslanmayla yıpranan betonarme yapılara doğal katkı maddeleri ile çözüm önerileri"
- Kıyak, C.B., 11. Otomotiv Sempozyumu (2009) "Kalıpçılık ve Tasarım" Bildirileri, "Çelik Yüzeylerdeki Pirol Katkılı Epoksi Boyaların Korozyona Karşı Koruma Refleksinin Tahribatsız ve Tahribatlı Elektro Kimyasal Testlerle Belirlenmesi"
- Atılım Üniversitesi (1., 2., 3., 4., ve 5.) Mekatronik Günleri Katılım Belgeleri (2007,2008,2009,2010,2011)
- Kıyak, C.B., MeMÖK(2011), Atılım Üniversitesi, 2. Mekatronik Mühendisliği Öğrenci Kongresi, "Uçan Robotlar İçin Gömülü Birimler"
- CATIA V5 R19 Başarı Belgesi (2011)
- Atılım Üniversitesi Robot Kulübü Asil Üyeliği (2006-2011)
- Sanayi Bakanlığı Teknogirişim Sermayesi Desteği Başarı Belgesi (2014)

- Sanayi Bakanlığı Teknopazar Tanıtım ve Pazarlama Desteği – ZEPS 22. Uluslararası Teknoloji Fuarı Sunumu (2015) Zenica, Bosna Hersek
- Ankara Kalkınma Ajansı 2017 Proje Yarışması İlk 9 ödülü olarak ticarileştirme programıyla İrlanda iş gezisi ve International Development Ireland ve Enterprise Ireland ile iş geliştirme görüşmeleri (2017)