אוניברסיטת בן גוריון בנגב המחלקה להנדסת תוכנה מאי 2015

פרויקט גמר

פרויקט הלוויין
אוניברסיטת בן גוריון

מסמך עיצוב.

מגישים : הוד עמרן,שמעון לחאיני,יניב רדומסקי ועידן מור.

מנחה אקדמי : דר' גרא וויס.

מנחה מקצועי : מר אבירן סדון.

תומכים מקצועיים (תעשייה אווירית) : מר שלמה סבירי, גב' עידית וקסלר.

פרק א': תרחישי שימוש.

Use-cases:

המשתמש העיקרי והיחיד בצד תחנת הקרקע הינו מפעיל המערכת.

יכול לבצע איתחול למערכת תחנת הקרקע, פענוח המידע הטמון בקובץ שהתקבל וכן יציאה סטנדרטית מן המערכת.

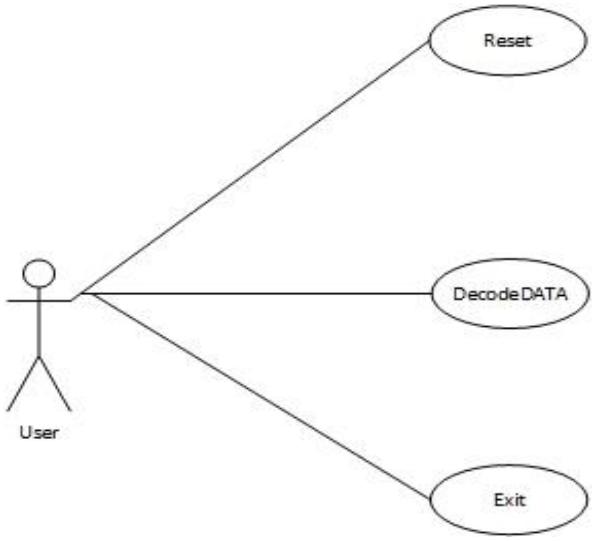


Figure 1.1 Use-Case Diagram

1. Reset

שחקן ראשי – משתמש תוכנת תחנת הקרקע (חוקר /מפתח).

תיאור- איפוס כלל המערכת, הנתונים האגורים בה, התצוגה כולה , חזרה למצב תחילת ריצת המערכת.

תנאי התחלה – התוכנה עלתה.

תנאי סיום – הנתונים של המערכת מאופסים , התצוגה מאותחלת .

תרחיש הצלחה עיקרי :

1. המשתמש בוחר לאפס את המערכת.
2. המערכת עוצרת את הפעולות שרצות דרכה (אם ישנם כאלה)
3. המערכת מאפסת את הנתונים המוחזקים בידה (טלמטריה שנאספה).
4. המערכת מאפסת את תצוגת המשתמש למצב התחלתי (אין ערכי מדידה ומידע מחיישנים)

תרחיש אלטרנטיבי:

-אין.

2. DecodeDATA

שחקן ראשי – משתמש תוכנת תחנת הקרקע (חוקר /מפתח).

תיאור- בחירת קובץ wav .

תנאי התחלה – התוכנה עלתה.

תנאי סיום – הנתונים מאוחסנים במערכת לאחר decode ו parsing, ומוצגים זה לאחר זה בתצוגת המשתמש .

תרחיש הצלחה עיקרי :

1. המשתמש בוחר באפשרות בחירת קובץ.
2. המערכת מציגה חלון לבחירת קובץ מסוג wav בלבד.
3. המשתמש בוחר את הקובץ שברצונו לפענח.
4. המערכת מייצרת תהליך חדש לשם תחילת הפענוח.
5. תהליך הפענוח רץ במקביל לתוכנה שרצה ומפענח את קובץ ה wav הנבחר.
6. המערכת נגשת לתוצאת תהליך הפענוח ומפרסרת את התוצאות לאובייקטי המערכת .
7. המערכת אוגרת את האובייקטים החדשים שפורסרו.
8. המערכת מציגה את התוצאות בזו אחר זו בהתאם לסדר הגעתם.

תרחיש אלטרנטיבי:

- קובץ ה wav אינו תקין לפורמט הלוויין- במקרה זה התהליך של decodeיעצור והמשתמש יצטרך לחזור ל 1.
- המשתמש סגר את תהליך ה decode שנפתח בעת ריצתו – במקרה זה המשתמש יצטרך לחזור ל 1, נתוני המערכת לא השתנו מכפי שהיו בתחילה.

3. Exit

שחקן ראשי – משתמש תוכנת תחנת הקרקע (חוקר /מפתח).

תיאור- יציאה מהמערכת.

תנאי התחלה – התוכנה עלתה.

תנאי סיום – המערכת סגורה .

תרחיש הצלחה עיקרי :

1. המשתמש בוחר לצאת מן המערכת.
2. המערכת נסגרת (באם תהליכים אחרים רצים ברקע, הם ימשיכו לרוץ – כמו decode)

תרחיש אלטרנטיבי:

-אין.

פרק ב': ארכיטקטורת המערכת.

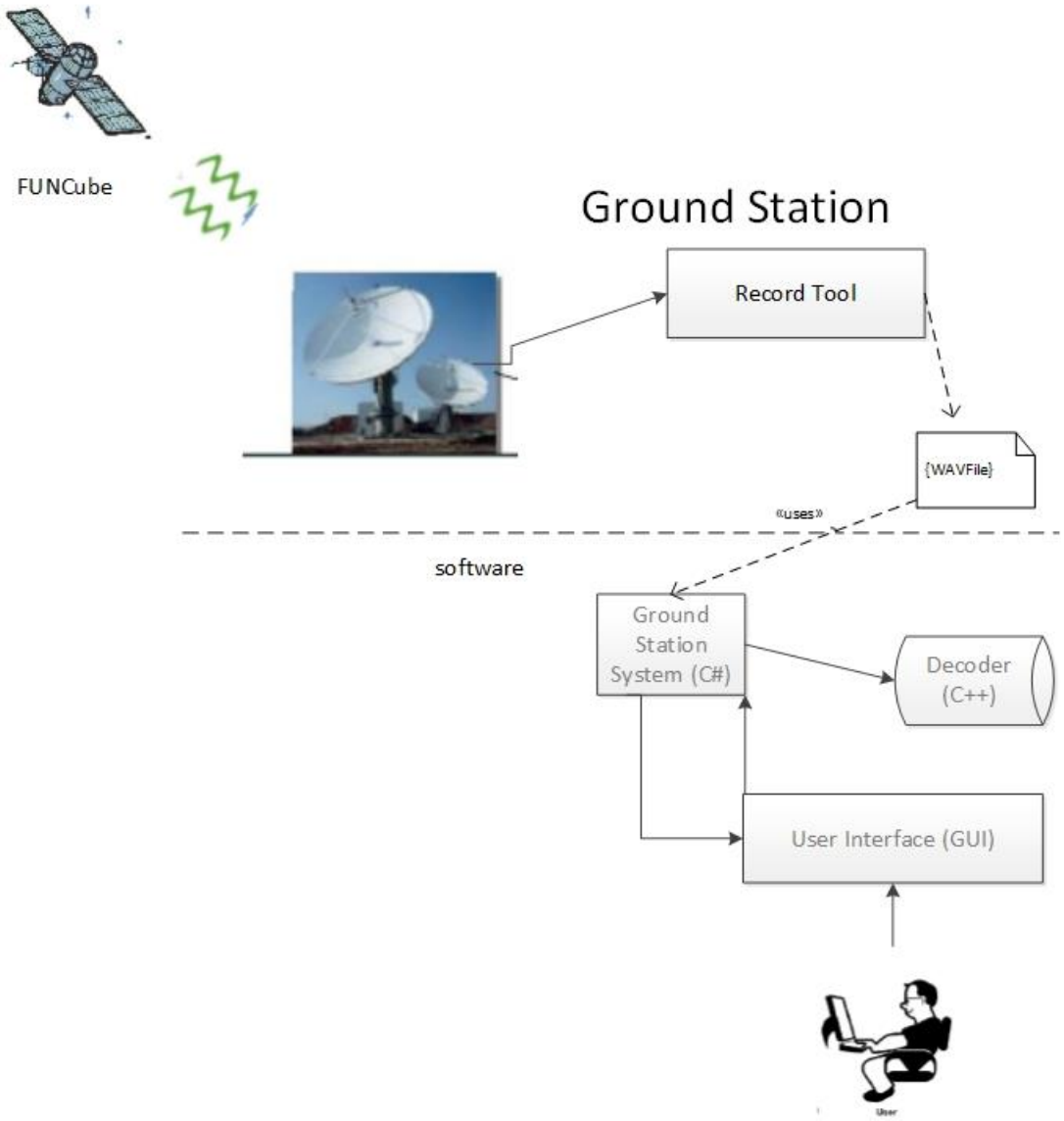


Figure 2.1 High-Level System Architecture Diagram

המערכת מורכבת ממספר רכיבי תוכנה ומספר רכיבי חומרה העובדים יחד:

<u>רכיב הלוויין</u> - FunCube הינו לוויין בריטי חינמי הנועד לצורכי מחקר, העשרה, חובבי לוויינים ורדיו. הוא חג סביב כדור הארץ כאשר בכל כ 24 שעות ישנה חליפה מול תחנת הקרקע האורכת כ - 9 דקות בזמן זה הלוויין "מוריד" את המידע שאגר כולל מצב החיישנים מצב פריסת הכנפיים הסולריות מצב סוללה ועוד.
משדר הלוויין משדר בקצבים של 1200bps באפנון BPSK.
הלוויין משדר את המידע כפאקטות, כאשר 8 הביטים הראשונים הינם מספר הסידור של הפאקטה 440 ביט של Real Time Telemetry ו1600 ביט של Payload. סה"כ 2048 bits (256 byte) לפאקטה יחידה.
במחזור שידור קיימים 24 פאקטות.

בין כל פאקטה לפאקטה ישנו "ביפ" המעיד על סיום ומעבר השידור לפאקטה הבאה כל פאקטה מתקבלת במשך 5 שניות.

תיאור סדר קבלת הפאקטות:

Frame Type	Frame Id	Frame type value
RTT + Whole Orbit	WO1	01
RTT + Whole Orbit	WO2	02
RTT + Whole Orbit	WO3	03
RTT + Whole Orbit	WO4	04
RTT + Whole Orbit	WO5	05
RTT + Whole Orbit	WO6	06
RTT + Whole Orbit	WO7	07
RTT + Whole Orbit	WO8	08
RTT + Whole Orbit	WO9	09
RTT + Whole Orbit	WO10	10
RTT + Whole Orbit	WO11	11
RTT + Whole Orbit	WO12	12
RTT + High Resolution	HR1	13
RTT + Fitter Message	FM1	14
RTT + Fitter Message	FM2	15
RTT + Fitter Message	FM3	16
RTT + High Resolution	HR2	17
RTT + Fitter Message	FM4	18
RTT + Fitter Message	FM5	19
RTT + Fitter Message	FM6	20
RTT + High Resolution	HR3	21
RTT + Fitter Message	FM7	22
RTT + Fitter Message	FM8	23
RTT + Fitter Message	FM9	24

מקלט תחנת הקרקע – המקלט הינו דונגל הנקרא FUNcube Dongle pro+ המתחבר למחשב בממשק USB. הדונגל אחראי לקליטת אותות הרדיו המתקבלות מן הלוויין וקידודם לאות דיגיטלי. לאחר קליטת האותות ניתן לשמור את תוכן המידע שהתקבל מן הלוויין כקובץ wav. זוהי הקלטה גולמית ביותר של קליטת אותות הרדיו אשר התקבלו ע"י הדונגל מהלוויין. ניתן להשתמש בכל תוכנה המנגנת קבצי wav על מנת להשמיע את השידור באיטרציה זו עשינו שימוש בתוכנה # SDR בה ניתן להגדיר את טווח השמעת התדרים.

בתמונה בעמוד הבא רואים את ניגון הקובץ כאשר תחום התדרים הרלוונטי בלבד מושמע. החלק התחתון (הצהוב) נועד לצורך הצגת תמונה מאותות רדיו, במידה ונשלחה כזו.

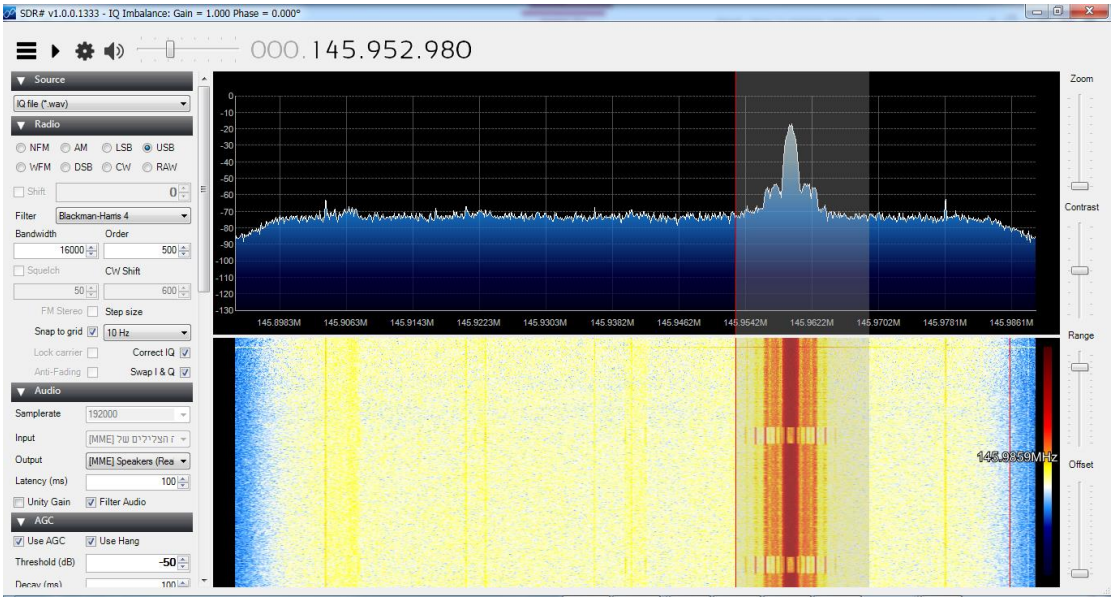


Figure 2.2 SDR#

Decoder – קטעי קוד ++c שבבסיסם נכתבו ע"י צוות תוכנה של חברת Yahoo בשיתוף עם צוות ה FunCube, ולהם ביצענו שינויים והתאמות עבור האפליקציה שלנו, תפקיד התכנית לבצע עיבוד לאות אנלוגי שנתקבל מהלוויין ולהמירו לדיגיטאלי לפי המודולציה בה משדר הלוויין בכדי שנוכל לבצע פרסור למידע שהוא מפיק. אנו משתמשים בה כתוכנה עצמאית המופעלת כרכיב חיצוני לאפליקציה שלנו. האפליקצה שלנו תפעיל את תכנית ה Decoder ותקבל כקלט את התוצרים. (רצפים של בתים).

Ground Station System – הרכיב העיקרי במערכת התוכנה מנהל את שטף המידע, הפעלת הרכיבים בסדר הנדרש, ניהול התהליכים דלגציה סנכרון ועוד. זוהי ליבת התוכנית.

רכיב GUI – רכיב הבא במגע ישיר עם המשתמש, מאפשר למשתמש בצורה ידידותית לטעון את קובץ ה wav הנדרש לצורך עיבוד, ומציג את הנתונים הסופיים אשר התקבלו מתעדכן על בסיס קבוע כל עוד תהליך פענוח המידע אשר הגיע מהלוויין נמשך. עובד כתהליך מקביל לתהליך עיבוד המידע.

פרק ג': Data Model.

תחנת קרקע:

GUI Package:

TelemetryWindow – החלון המרכזי שמציג את הנתונים שנתקבלו מהלוויין, חלון זה למעשה הינו המנשק משתמש למול הלוויין. בחלון זה ניתן לבחור את קובץ ה wav שהוקלט מהלוויין ולעבד את המידע שהוא מכיל לפי הפורמט שמתואר באתר הלוויין.

Domain Package :

Parser – מחלקה סטטית המשמשת כמחלקת עזר לביצוע פרסור המידע שנתקבל לאחר תהליך ה decode. מכיל מטודות סטטיות לצרכי פרסור בלבד. לא נוצר מופע שלה במהלך ריצת המערכת.

GroundStation – אובייקט מסוג singleton, מכיל את פונקציית ה main, ממנו רצה המערכת כולה. מאתחל את החלון של מנשק המשתמש וכן את המשתמשים המשמשים אותו בריצת המערכת.

BitStream- מחלקה שבאמצעותה אנו מייצגים מידע שנתקבל בפריים מסוים וממנו בונים אובייקט telemetry באמצעות רצף סיביות. יורשת ממחלקת הספרייה IO.Stream, מכילה מיני מטודות לצורך ייצוג המידע בדרכים שונות, כתיבה וקריאה לאובייקט זה.

Telemetry – מכיל מידע אודות כל חיישני הלוויין ב real-time וכן payload שמתווסף לכל פריים שנשלח מהלוויין. המידע מאורגן באמצעות dictionary הממפה מאינדקס החיישן ,שזהו המפתח, לבין ערך המידע TelemetryValue שהוא מדד.

TelemetryValue – מחלקה אבסטרקטית המייצגת מידע שנדגם מחיישן לוויין כלשהו שאותו ניתן להציג בכמה דרכים : כמשתנה מספרי, כמשתנה מספרי עם דיוק לאחר הנקודה העשרונית, כטקסט ועוד. מחלקת זו היא למעשה מחלקת אב לכל דגימת חיישן כלשהו מהלוויין, בהמשך נראה כי היא מייצגת המון דגימות.

כל המחלקות הבאות הינן מחלקות יורשות מהמחלקה הנ"ל וממשות את כל המטודות הווירטואליות שלה, כל אחת מייצגת מידע מסוג שונה (לרוב מדובר מחיישן מסויים) , ההבדלים ביניהן אינן משמעותיים ולרוב הן נוצרו לצרכי נוחות ע"מ לבצע הפרדה ברורה בין מקורות המידע של הלוויין:

BoolOnOffTelemetryValue – מציג מידע אודות חיישן או רכיב ברמה הבוליאנית האם הוא עובד או לא.

NullTelemetryValue- מציג מידע לא מוכר שהגיע מהלוויין, בשימוש בעיקר לצרכי בדיקות.

AntsDeployTelemetryValue- מציג האם הלוויין פרס את הזרוע שלו (ישנן 4 זרועות הממוספרות מ 0 עד 3) ועבור כל אחת נקבל מיגע האם נפרסה או לא.

SunSensorTelemetryVallue- מציג את תוצאת המדידה מכל אחת מקולטי השמש לפי טבלת ההמרה שמצוינת בטבלת המשוואות והערכים.

DeviceDataValidTelemetryValue- מציג האם המידע שמספק הלוויין הינו תקין, גם כאן המידע הינו בוליאני , אך מוצג כטקסט בצורה שונה לעומת BoolOnOffTelemetryValue.

EPSResetCauseTelemetryValue- מציג את הגורם לאיפוס חיישן ה EPS :

Power on– דולק.

External– גורם חיצוני

Brown Out– בעיית טמפרטורה.

Watchdog– תהליך תוכניתי-חומרתי.

Jtag– חיבור חומרתי.

Other

EPSPowerTrackingTelemetryValue – מציג את 'מוד' הפעולה של חיישן זה:

Hw default

MPPT

SW Fixed

Unknown

SoftwareAbfTelemtryValue – מציג האם חיישן זה מאופשר או לא.

MultuplierOfssetTelemetryValue – מחלקה המייצגת תוצאות מדידות של חיישנים שונים אשר להם נוסחת המרה קבוע המכפילה בקבוע כלשהו (שדה ה Multiplier של מחלקה זו) ומתווסף ערך סף מסוים (שדה ה offset של מחלקה זו) דוגמה לחיישן כזה הינו חיישן מדידת טמפרטורה של פאת לוויין.

MultuplierPowerTelemetryValue- בדומה למחלקה הקודמת, מייצג חיישנים אשר את הערך שמפיקים ממדידותיהן צריך להעלאות בחזקה. דוגמת חיישן כזה הינו PaPower.

PaCurrentTelemetryValue- יורש מ MultuplierOfssetTelemetryValue, תוך השמת ערכים שונים להכפלה בקבוע והוספת סף בהתאם לחיישן.

PaPowerTelemetryValue- יורש מ MultuplierPowerTelemetryValue, תוך השמת ערכים להעלאה חזקה עבור בסיס מסויים בהתאם לנוסחת החיישן.

MultiplierTelemetryValue- בדומה ל MultuplierOfssetTelemetryValue רק ללא offset , ז"א שימוש עבור MultuplierOfssetTelemetryValue עם שדה ה offset=0 יפיק את אותה תוצאה.

FrameIdTelemetryValue- מציג את סוג הפריים שנתקבל :

Whole orbit data.

High resolution data.

Fitter messages.

PaTemperatureTelemetryValue- מציג את טמפרטורת חיישן ה pa לפי טבלת ההמרה שמצורפת בקובץ הנוסחאות של הלוויין.

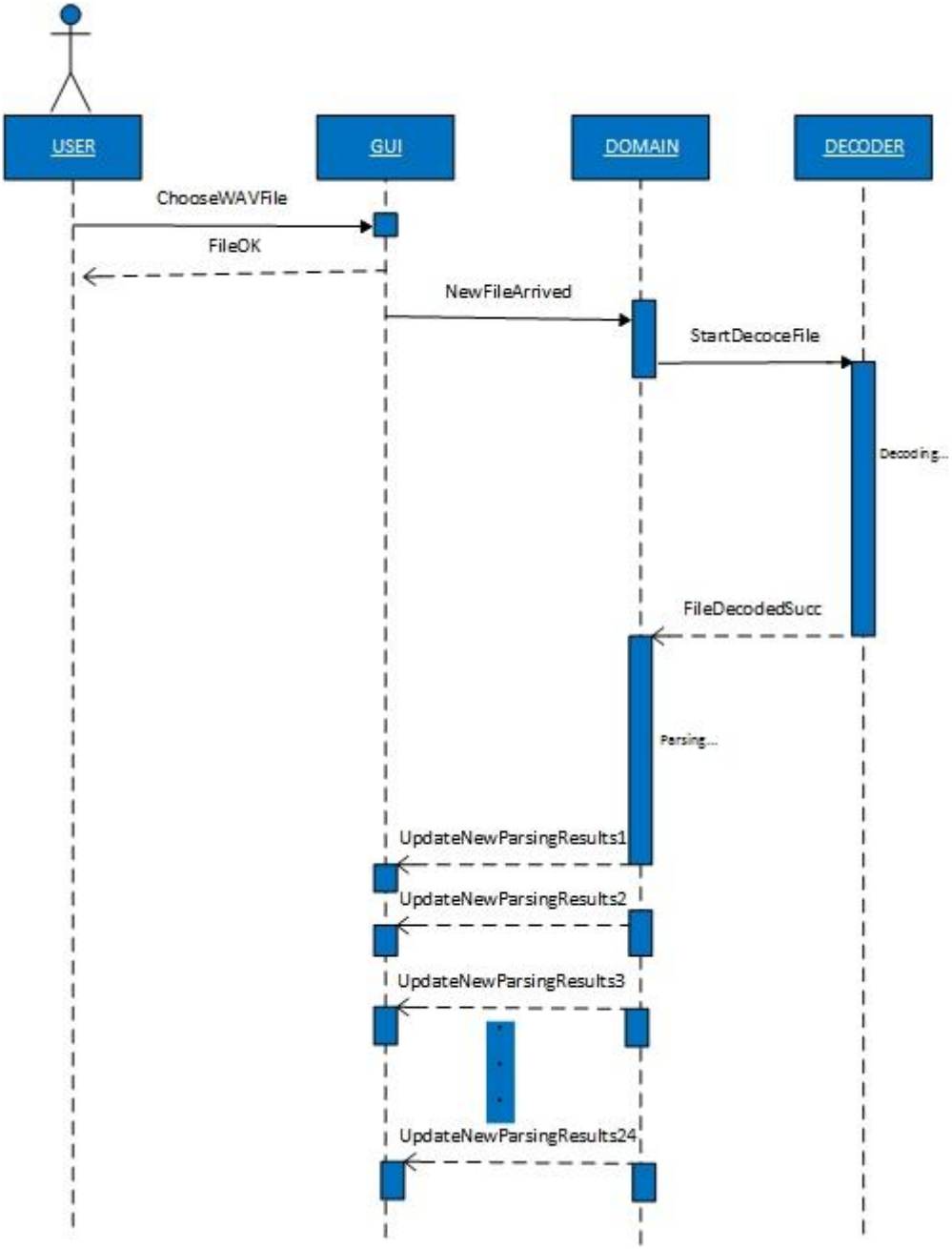
פרק ד': Behavioral Analysis

Sequence diagrams:

DecodeDATA:

מייצג את הליבה של התוכנה באיטרציה זו – מקבלת המידע הגולמי מהלוויין ועד הצגתו למשתמש.

המשתמש בוחר בממשק הGUI את הקובץ אשר ברצונו לקודד, שכבת ה GUI מעבירה את נתיב הקובץ לשכבת הDOMAIN אשר מפעילה את רכיב ה DECODER. לאחר קידוד המידע ושמירתו על קובץ שכבת ה DOAMIN מפרסרת את המידע ומעבירה לתהליך ה GUI את המידע המעובד להצגתו למשתמש.



פרק ה': Object-Oriented.

תרשים מחלקות עבור רכיב ה GroundStaion:
מכוון שתרשים המחלקות רחב מצ"ב קובץ המציג את תרשים המחלקות ע"מ שיהיה קריא:

רכיב ה GroundStaion שהינו סינגלטון מחזיק וקטור של טלמטריה כך שכל איבר בווקטור הינו טלמטריה – פקטה המוכנה להצגה ע"פ סוגי הפקטות שהתקבלו.
כל איבר בוקטור הטלמטריה הינו מסוג ***TelemetryValue*** כפי שניתן לראות בתרשים ישנם מחלקות רבות היורשות ממחלקה זו. למעשה כול סוגי המידע האטומי שהתקבלו מהלוויין הינם יורשים ממחלקת ***TelemetryValue*** שכבת ה GUI תציג אותם בהתאם.
בנוסף ישנה מחלקת static אשר משמשת כמחלקת עזר מחלקה זו עוזרת בפרסור המידע, ניקוי מחרוזות וכדומה.

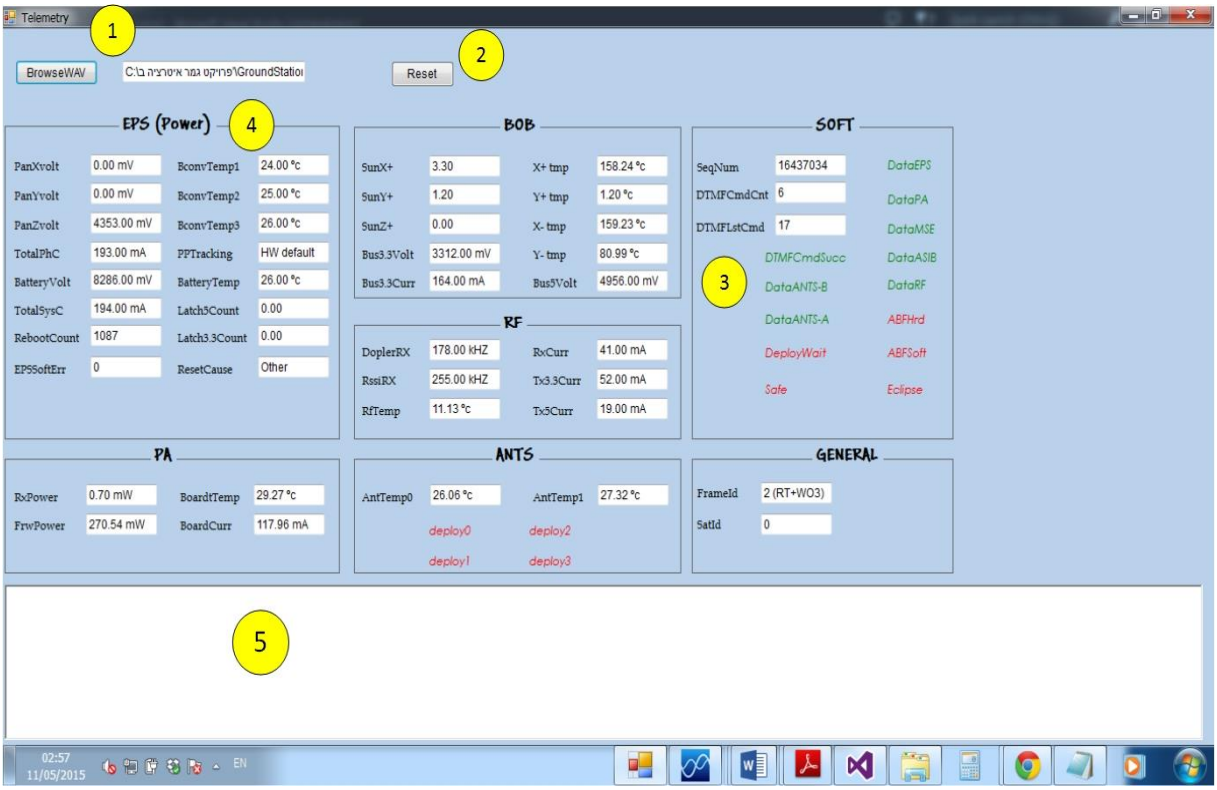
ע"מ שניתן יהיה לקרוא בבהירות את תרשים המחלקות צירפנו אותו כלינק לקובץ.

קישור לתרשים מחלקות.

המחלקה העיקרית – מחלקת ה - ground station מכילה בתוכה את כל המידע שהתקבל ועבר דמודולציה. היא מחזיק וקטור של של Telemetry.
המחלקה Telemetry מחזיקה אף היא רשימת ערכים (TelemetryValue) המייצגים את פאקטת הטלמטריה.
ממחלקת TelemetryValue יורשות מחלקות רבות, כאשר כל מחלקה יורשת מייצגת קריאת חיישן או מידע אטומי אחר כגון טמפרטורת כנף, מתח בסוללות וכדומה.

פרק ו' :User Interface Draft

חלון התוכנה העיקרי:



החלון לעיל הינו חלון התוכנה העיקרי מולו עובד המשתמש.

בכפתור בסימון 1 (מסומן בעיגול צהוב ובתוכו המספר 1) מתבצע טעינת קובץ ה wav אשר עליו תעבוד המערכת – קידוד פרסור ולבסוף הצגתו למשתמש. ניתן לבחור קבצי wav בלבד.

ע"י לחיצה על כפתור ה reset המסומן בספרה 2 המערכת תתאפס וכל החלוניות המציגות מידע ינוקו. אם המערכת בהרצה דהיינו פענוח הקובץ או משימה אחרת המערכת תעצור ותתאפס וניתן יהיה לבחור קובץ מחדש.

הסימון מספר 3 מציג את מצב רכיבי הלוויין כפי שהתקבלו בחיישני הלוויין למשל פריסת הכנפיים הסולריות וכדומה. ירוק מסמן תקין אדום יסמן לא תקין\לא בוצע\לא התקבל מידע עבור חיישן זה.

סימון מספר 4 מציג את הנתונים המספריים שהתקבלו מחיישני הלוויין לאחר שהועברו קידוד – כל חיישן עם קידוד שונה ע"מ להציג את הערך המספרי בין אם זה טמפרטורה, מתח , kHZ וכו'.

סימון מספר 5 – עבור פקטות מסוימות הלוויין שולח payload המכיל טקסט, כאשר תתקבל פקטה כזו הטקסט יוצג באזור זה המסומן בספרה 5.

פרק ז': Testing

לצורכי בדיקות עשינו שימוש ב UnitTest.לכל מחלקה הוגדר מספר בדיקות כך שבעתיד ניתן יהיה להוסיף בדיקות על אלו הקיימות בצורה נוחה.

End-to-End Testing :

רוב הבדיקות בוצעו כבדיקות "קופסא שחורה" לקיחת מספר טלמטריות שהתקבלו מחיישני הלוויין ביצוע דמודולציה פרסור והצגתם.

אינדיקציה לתוצאות הבדיקות יכולנו לבצע מול התוכנה המסופקת עם הFunCube. התוכנה נקראת DashBoard.

תוכנה זו יודעת לקחת את נתוני הלוויין ולפענח אותם, נעזרו בתוצאות שהתקבלו בתוכנה והשוואתם מול התוצאות שהתקבלו באפליקציה שלנו כך יכולנו לדעת מה מידת הסטייה והאם ביצענו דמודולציה והמרת יחידות תקינה.

Unit Testing / Component Testing :

ברכיבים המודולרים אשר ניתן היה לבדוק אותם כיחידה עצמאית דימינו מספר קלטים שגויים ותקינים לכל רכיב ובדקנו כיצד המערכת מגיבה –
קליטת קובץ wav לא תקין למשל קובץ שהוא בעצם סאונד של שיר או הקלטה של שידור רדיו כאשר עוצמת האות נמוכה או שגויה.
סידור לא נכון של פאקטות , הפרה של פרוטוקול התקשרות וסדר קבלת המידע.
בנוסף נבדקה רכיב הפרסור הן עבור ערכים תקינים והן עבור ערכים שאינם תקינים, בדיקה של משתני הקצה.

דוגמא לביצוע מספר בדיקות עבור רכיב ה – parser.
לאחר שבוצע דמודולציה, בדיקה של ערכי החיישנים והשוואתם לתוצאות התוכנה Dashboard.

UnitTest1.cs | TelemetryWindow.Designer.cs | TelemetryWindow.resx

UnitTestProject1.UnitTest1 | TestMethod4()

```
[TestMethod]
0 references
public void TestMethod4()
{
    List<Telemetry> AllTelems = new List<Telemetry>();
    string[] Packets = File.ReadAllLines(@"ssssssss.txt");
    for (int i = 0; i < Packets.Length; i++)
    {
        byte[] bytes = Parser.StringToByte(Packets[i]);
        AllTelems.Add(new Telemetry(bytes));
    }

    Telemetry TestTelm = AllTelems[0];
    Assert.AreEqual(1087, TestTelm.Get("RebootCount").AsInt);
}

[TestMethod]
0 references
public void TestMethod5()
{
    List<Telemetry> AllTelems = new List<Telemetry>();
    string[] Packets = File.ReadAllLines(@"sssssssss.txt");
    for (int i = 0; i < Packets.Length; i++)
    {
        byte[] bytes = Parser.StringToByte(Packets[i]);
        AllTelems.Add(new Telemetry(bytes));
    }
```

Test Explorer

Search

Streaming Video: Improving quality with

Run All | Run... | Playlist : All Tests

Passed Tests (6)

TestMethod1	45 ms
TestMethod2	13 ms
TestMethod3	8 ms
TestMethod4	7 ms
TestMethod5	8 ms

TestMethod1

Source: UnitTest1.cs line 15

Test Passed - TestMethod1

Elapsed time: 45 ms

Output