






פרויקט תקשורת


מבנה מחשבים ספרתיים להנדסת
מחשבים


Terminal Application

 **Light
Terminal**

 Parameters

 Command

 Send Files

 Recieve Files

> First choose PORT connection !

Choose PORT ▾

9600 ▾

8 ▾

1 ▾

non ▾

Connect

Disconnect

Disconnected

Baud rate

Bits per byte

Stop bit

Parity bit

Send parameter's

Change parameter's

מאור אסייג 318550746
רפאל שטרית 204654891
מנחה : חנן ריבוא

תוכן עניינים

3-4 עמודים תיאור המשימה

5-18 עמודים תיאור האפליקציה והקוד C#

19 עמודים תרשים זרימה צד PC

20 - 22 עמודים תיאור הקוד C

נספח – הקוד כמכלול

תיאור כללי

תיאור המשימה

תיאור כללי

מטרת פרויקט התקשורת הינה יישום תקשורת א-סינכרונית דו-כיוונית, המיישמת את סטנדרט RS-232 (יורחב בהמשך) בין בקר Freescale KL25Z למחשב.

לכל רכיבי החומרה השונים ניתנו מדריכי קנפוג ועבודה בהתאם ע"י המנחה, חלקים נרחבים מהם אינם מצויינים במסמך זה.

התקשורת תמומש באמצעות פסיקות (ולא באופן סינכרוני) בשני הצדדים. המשדר והמקלט ברכיבי החומרה שבמחשב ובבקר יתמכו באופן מלא בתקשורת בעזרה פסיקות.

המערכת תכלול את אפשרויות התקשורת הבאות :

- בחירת פרמטרי התקשורת.
- חלון פקודות מובנות מהמחשב לבקר.
- העברת קבצים מהמחשב לבקר, ולהפך.
- שליחת קבצי Script מהמחשב לבקר, וביצוע הקובץ מהבקר.

בצד המחשב פיתחנו תוכנית עם ממשק משתמש נקי, בשפת C# (סביבת עבודה Visual Studio 2017) אשר תנהל את התקשורת.

בצד הבקר תוכנית בשפת C לבקר בסביבת עבודה CodeWarrior.

הערה: בצד המחשב הספריה serialport ובפרט רכיבי התקשורת תומכים ב 5-8 databit ואילו הבקר איתנו ביצענו את הפרוייקט תומך ב-8 databit 10. עובדה זו מונעת משינוי רוב פרמטרי התקשורת בשני הצדדים ולהמשך תקשורת תקינה (מלבד baudRate).

מאפייני התקשורת

שכבה פיזית: פרוטוקול RS-232 ללא תמיכה במודם באמצעות חיבור 3 קווים בלבד.

שכבת הערוק (Data link layer): באמצעות בקר ה- UART ב- KL25Z ובקר התקשורת ב- PC.

שכבת האפליקציה:

- ממשק להעברת תווים בין הצדדים, העברת פקודות מה- PC לבקר, העברת הודעות ACK מהבקר ל- PC והעברת קובץ בין הבקר ל- PC.
- כתיבת ממשק גרפי המאפשר קונפיגורציה של פרמטרי התקשורת, העברת פקודות והצגת נתונים שנאספו מהבקר על מסך ה- PC.

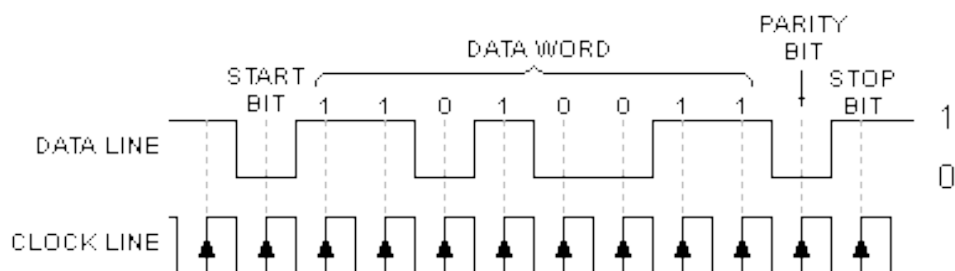
פרוטוקול RS-232: הפרוטוקול מקשר בין DTE (אחראי לשלוח מידע) לבין ה- DCE (אחראי לקלוט את המידע שנשלח).

הפרוטוקול מגדיר את הפרמטרים הבאים:

- מתחי עבודה
- התנגדויות קצה וקו
- זמני עליה וירידה
- קצב שידור וקצב מקסימלי
- קבוליות מקסימלית לקו התקשורת

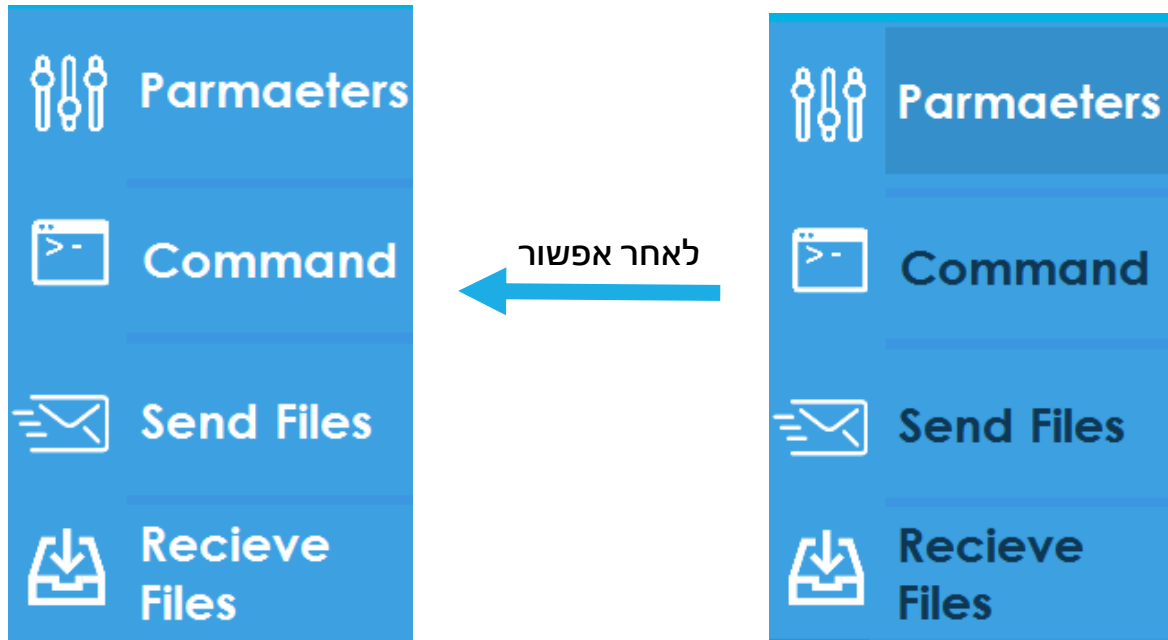
המידע העובר על הקו הוא byte בודד, תוך כדי סדרת תיחול (ראשית Start bit). הבית מלווה ב Parity bit המשתמש בקרת שגיאות ונגמר ב Stop Bits המעידים על סיום התשדורת.

הפרמטרים הנ"ל ניתנים לבחירה על ידי המשתמש באפליקציה ב- PC.



תיאור האפליקציה והקוד C#

מעבר בין חלונות



הסמן מצביע על Parameters.

לכל כפתור יש שדה **Enabled** המאפשר רק כאשר מקומו בתהליך התשדורת אפשרי עבור המשתמש. ביצירת כפתורים מוגבלי גישה הזנו לשדה זה `false`. כך למשל, חלון **Command** אינו מאפשר גישה ללא התחברות ל Port וקביעת פרמטרי התקשורת הרצויים (ישנם פרמטרים המהווים ברירת מחדל). בעת שגישתו מתאפשרת, פשוט מאפשרים אותו למשתמש על ידי שינוי שדה `Enabled` שלו.

```
private void openCommunication()
{
    ConnectButton.Enabled = false;
    DissconnectButton.Enabled = true;
    BaudrateBox.Enabled = true; //baud rate
    DatabitBox.Enabled = true; //BFS
    StopbitBox.Enabled = true; //stop bit
    ParityBox.Enabled = true; //pairty box
    CommandBox.Enabled = true; //otherwise you cant write commands
    CommandPanelButton.Enabled = true; //you can inter the command
}
```

תצוגה חלקית של פונקציית עזר לפתיחת התקשורת.

כפי שניתן לראות, בפתיחת התקשורת מתאפשרים קופסאות הבחירה\כפתורים למשתמש.

חלון default – חלון Parameters

ראשית על המשתמש לבחור באיזה PORT מתבצעת התקשורת, ולאחר מכן ללחוץ על **Connect** – לחצן שיבצע את ההתחברות. במידה והתקשורת תקינה, תיווצר גישה לאחד מפורטי התקשורת במחשב.

סטטוס בר Disconnected/connected – משתנה צבעו והטקסט בו בהתאם למצב החיבור (מחובר – ירוק, מנותק – אדום).

לאחר חיבור תקין לאחד מהפורטים, נפתחת האפשרות לבחור את פרמטרי התקשורת השונים וגישה לשאר החלונות.

Baude rate : 1200 /2400 /4800 /9600 /19200 /38400 /57600 /115200

Bits per byte(on transmit) : 5 /6 /7 /8

Stop bit : 1 /2

Parity bit : non /even /odd

- במידה והמשתמש שינה את אחד מערכי ברירת המחדל של פרמטרי התקשורת, תתאפשר לחיצה על **Send Parmater's**.
בלחיצה על כפתור זה נוצר מערך המורכב מ 8 בתים, כאשר הבית הראשון יכול סימן מוסכם עבור הבקר לתחילת תקשורת המהווה שינוי פרמטרי התקשורת. 4 הבתים הבאים יסמנו את קצב השידור החדש (Baud rate), ולאחר מכן שאר פרמטרי התקשורת יקבלו בית אחד.

```
SendBytes = new byte[8];
// Starting sign - 'B'
SendBytes[0] = BitConverter.GetBytes('B')[0];
//Baud Rate
byte[] temp = BitConverter.GetBytes(baudRate);
SendBytes[1] = temp[0];
SendBytes[2] = temp[1];
SendBytes[3] = temp[2];
SendBytes[4] = temp[3];
```

ראשית נמיר לביטים את הסימן המוסכם לתחילת מערך הפרמטרים שיישלח לבקר בעזרת **BitConverter**, נמיר את בחירת המשתמש מהקופסא של ה baudrate ולבסוף נעתיק את התאים הרלוונטיים (המכילים קידוד לביטים) אל מערך התשובה שישלח לבקר.

- כאשר משתנה בחירת המשתמש באחת מקופסאות הבחירה, נשמרים הפרמטרים הרצויים לתקשורת כמשתנים גלובליים לשימוש מאוחר יותר.

```
private void BaudrateBox_SelectedValueChanged(object sender, EventArgs e)
{
    if (baudRate != Int32.Parse(BaudrateBox.Text))
    {
        baudRate = Int32.Parse(BaudrateBox.Text);
        SendPara.Enabled = true;
        PcParaChange = false;
        ControllerParaChange = false;
    }
}
```

דוגמא לשמירת **StopBit** החדש (אובייקט מסוג StopBits) לפי בחירת המשתמש :

```
StopBit = int.Parse(StopbitBox.SelectedItem.ToString());
switch (StopbitBox.SelectedItem.ToString())
{
    case "1":
        actualStopBit = StopBits.One;
        break;
    case "2":
        actualStopBit = StopBits.Two;
        break;
}
```

תחילה מומר הערך הנבחר למספר **int**, ולאחר מכן לפי הבחירה נשמר הטיפוס הרצוי.

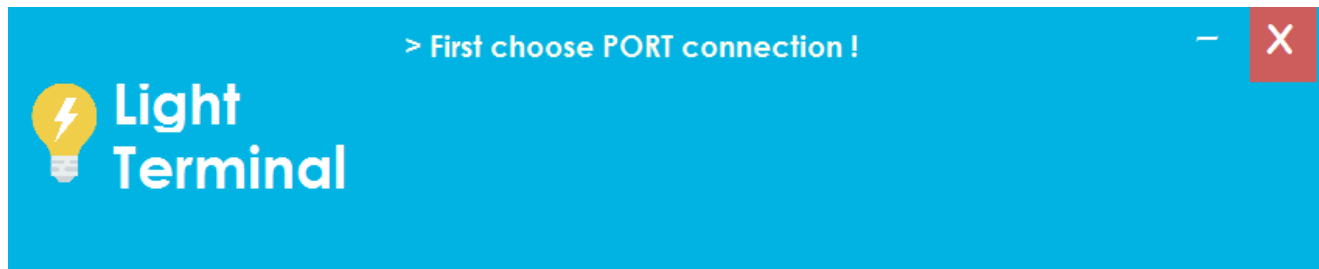
- לאחר שהבקר יקבל את חבילת המידע לגבי פרמטרי התקשורת החדשים הוא יישלח אות מוסכם חזרה למחשב כ **Acknowledge** "a" – ויתאפשר למשתמש ללחוץ על כפתור "Change Parameters" וסוגר את התקשורת הנוכחית.

```
case "a": //port parameters changed
    currentPort.DiscardInBuffer();// discard current data, if exist
    TopMessageBox.AppendText("Controller Parameters changed !\n");
    closePort(); //close the port
    ChangePara.Enabled = true; // enabled change in pc parameters
    SendPara.Enabled = false;
    ControllerParaChange = true;// the controller has change his parameters
    break;
```

- בלחיצה על **Change Parameters** נסגרת ופתחת תקשורת חדשה במחשב לפי פרמטרי התקשורת הגלובליים החדשים ומוזדפסת הודעה לאזור ההודעות העליון.

```
private void ChangePara_Click(object sender, EventArgs e)
{
    closePort();
    openPort();
    TopMessageBox.AppendText("Connected with new parameters !\n");
}
```


החלון העליון הינו חלון הודעות על המתרחש בתקשורת \ פירוט פקודות שניתנות לשליחה לבקר כפי שיפורט בהמשך.



סמן העכבר מצביע על כפתור הסגירה.



כמו כן בכדי לאפשר עיצוב "נקי", **עיצבנו לחלוטין מחדש** את כל פונקציונליות החלון כמו כפתור הסגירה, מזעור, גרירה מהאזור העליון של האפליקציה ועוד.

```
private void ExitButton_Click_1(object sender, EventArgs e)
{ this.Close(); }

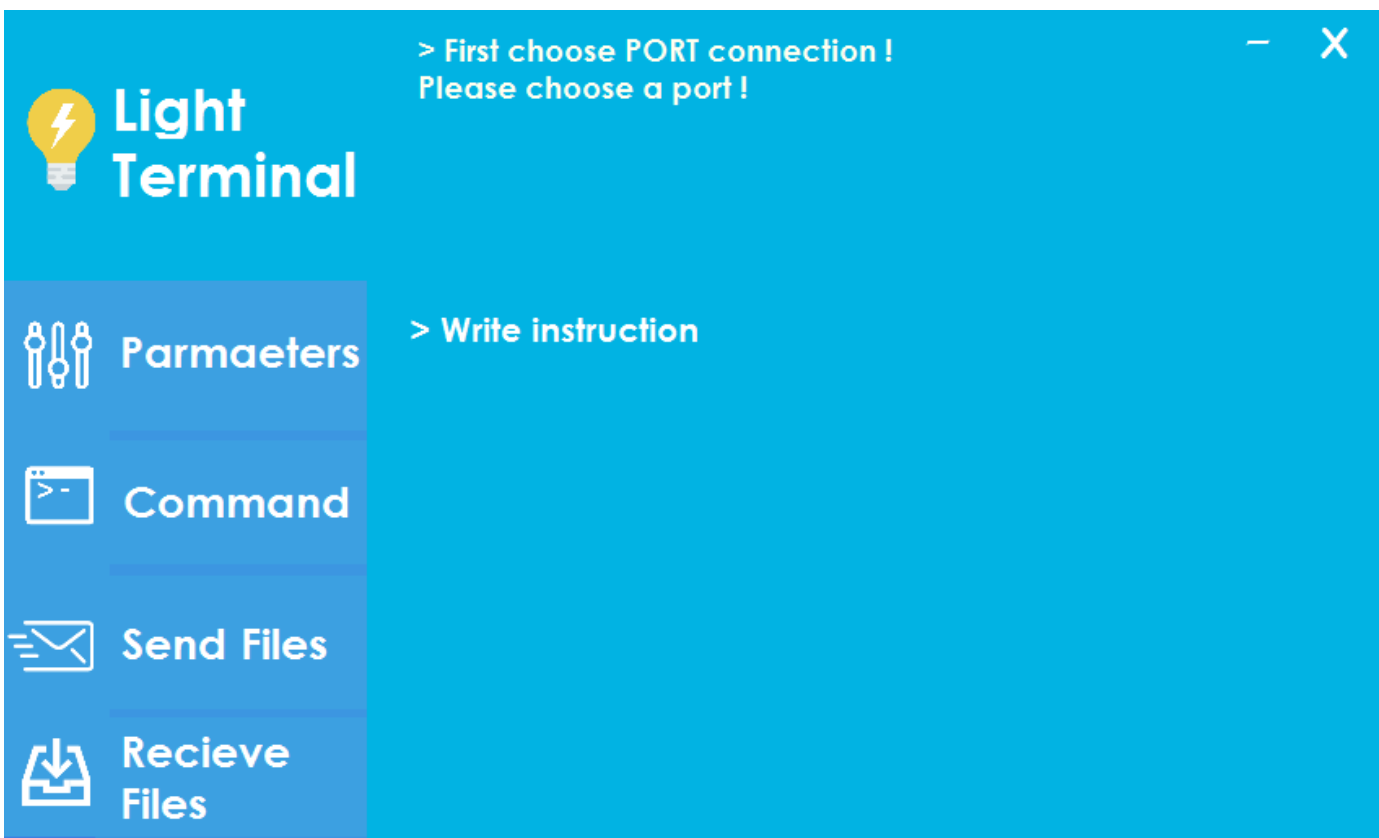
private void MinimizeButton_Click(object sender, EventArgs e)
{ this.WindowState = FormWindowState.Minimized; }

// mouse can move the form
int mouseX = 0, mouseY = 0;
bool mouseDown;
private void pictureBox6_MouseMove(object sender, MouseEventArgs e)
{
    if (mouseDown)
    {
        mouseX = MousePosition.X - 300 ;
        mouseY = MousePosition.Y - 4;
        this.SetDesktopLocation(mouseX, mouseY);
    }
}

private void pictureBox6_MouseUp(object sender, MouseEventArgs e)
{ mouseDown = false; }
```

כאשר **ExitButton** הינו סמן היציאה X, **MinimizeButton** הינו סמן המיזעור -, והאזור העליון של האפליקציה (**pictureBox6**) מאפשרת גרירת החלון ושינוי המיקום שלו על מסך המחשב.

חלון Command

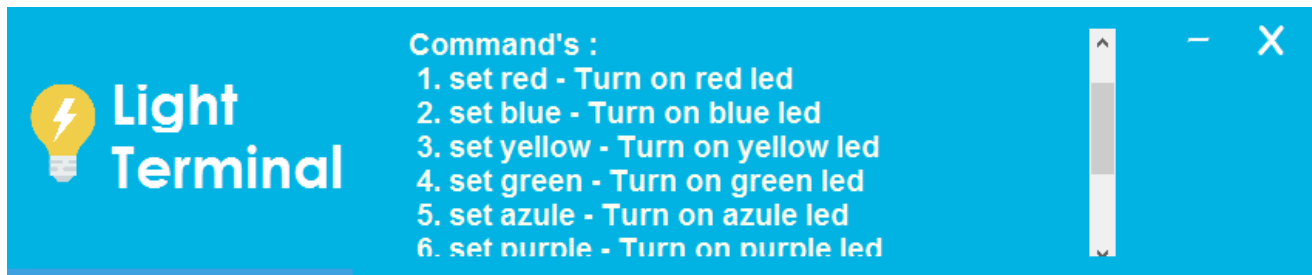


לאחר חיבור תקין אל PORT וקביעת פרמטרי התקשורת הרצויים, ניתן לכתוב פקודות מובנות מראש אל הבקר בחלון זה.

מבנה ההודעות הידועות מראש:

- תו ENTER מפריד בין פקודה לפקודה (ירידת שורה = ENTER).
- פקודת הדלקת צבע לד RGB (מתוך 7 אפשרויות) לדוגמה – set .blue
- פקודת כיבוי לד clear rgb – RGB.
- פקודת השהייה (לכתיבה בסקריפט בלבד), מספר באורך 4 ספרות (בבסיס 10) ביחידות של msec לדוגמה – delay 400.
- פקודת reset להתחלת הצ'ט מחדש – reset.

לאחר כל לחיצת **ENTER** תופיע רשימת הפקודות המובנות בחלון ההודעות העליון.



במידה וכתבנו פקודה מסוימת, בסיום זיהוי מוצלח יישלח לבקר סימן מוסכם של הפקודה (ובסיום הסימן המוסכם אות סיום '<'); ;

```
switch (CommandBox.Lines[CommandBox.Lines.Length - 2])
{
    case "set red":
        port.Write("1");
        port.Write(">");
        break;
```

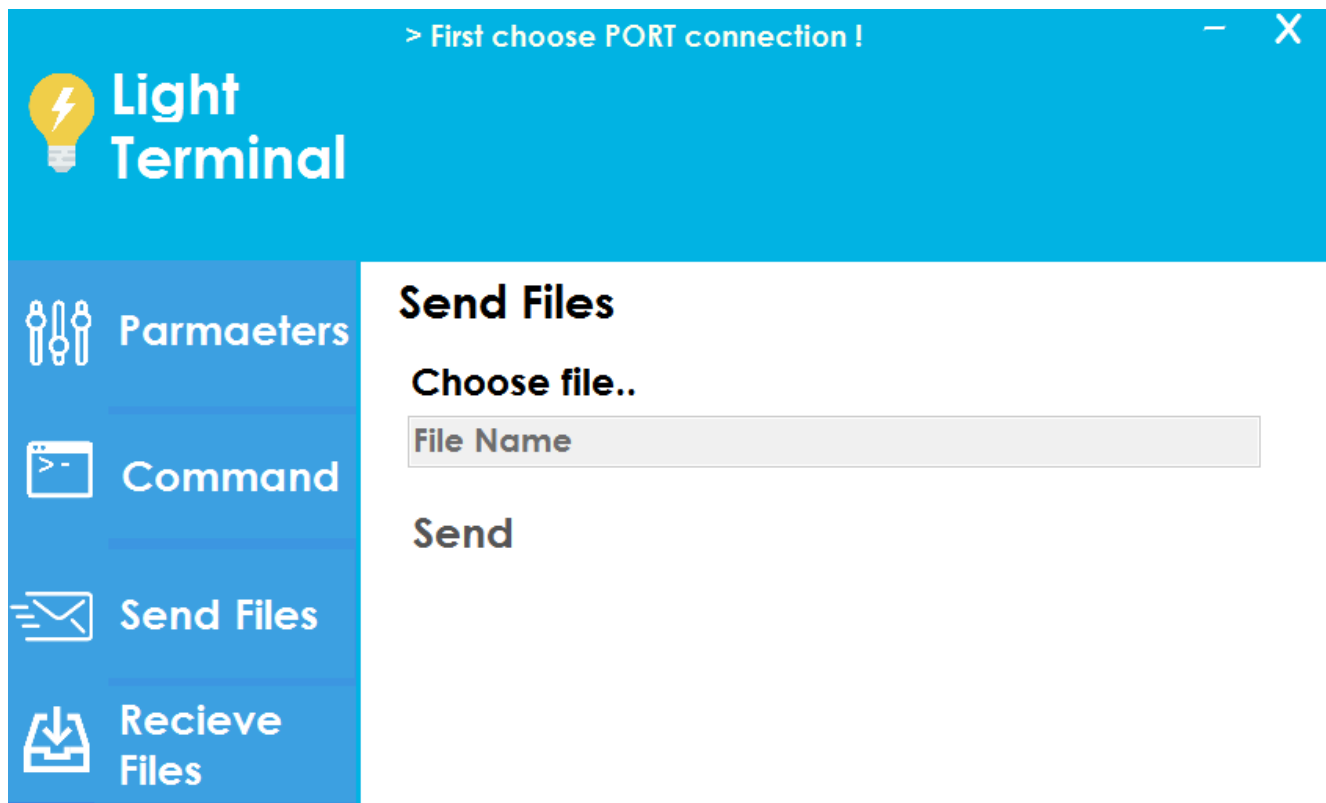
בצד הבקר תתבצע הפקודה לאחר פענוח האות המוסכם (בצורה זהה).

רשימת פקודות מובנות מראש :

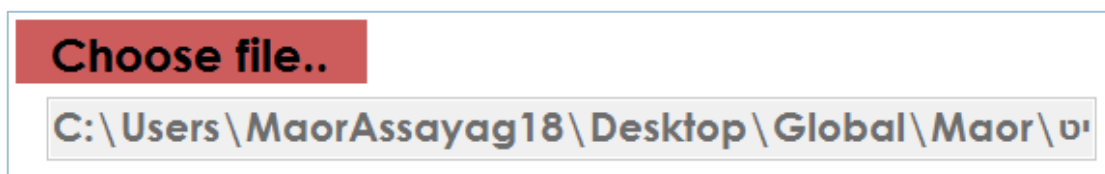
```
TopMessageBox.AppendText(" Valid Command's : ");
TopMessageBox.AppendText(" 1. set red - Turn on red led ");
TopMessageBox.AppendText(" 2. set blue - Turn on blue led ");
TopMessageBox.AppendText(" 3. set yellow - Turn on yellow led ");
TopMessageBox.AppendText(" 4. set green - Turn on green led ");
TopMessageBox.AppendText(" 5. set azule - Turn on azule led ");
TopMessageBox.AppendText(" 6. set white - Turn on white led ");
TopMessageBox.AppendText(" 7. set purple - Turn on purple led ");
TopMessageBox.AppendText(" 8. clear rgb - Turn off rgb led ");
TopMessageBox.AppendText(" 9. reset - To reset the controller ");
TopMessageBox.AppendText(" 10. in Script only : delay X milliseconds ");
```

לכל פקודה ישנו סימן מובנה מראש שיישלח אל הבקר לזיהוי הפקודה שנכתבה על ידי המשתמש.

חלון Send Files



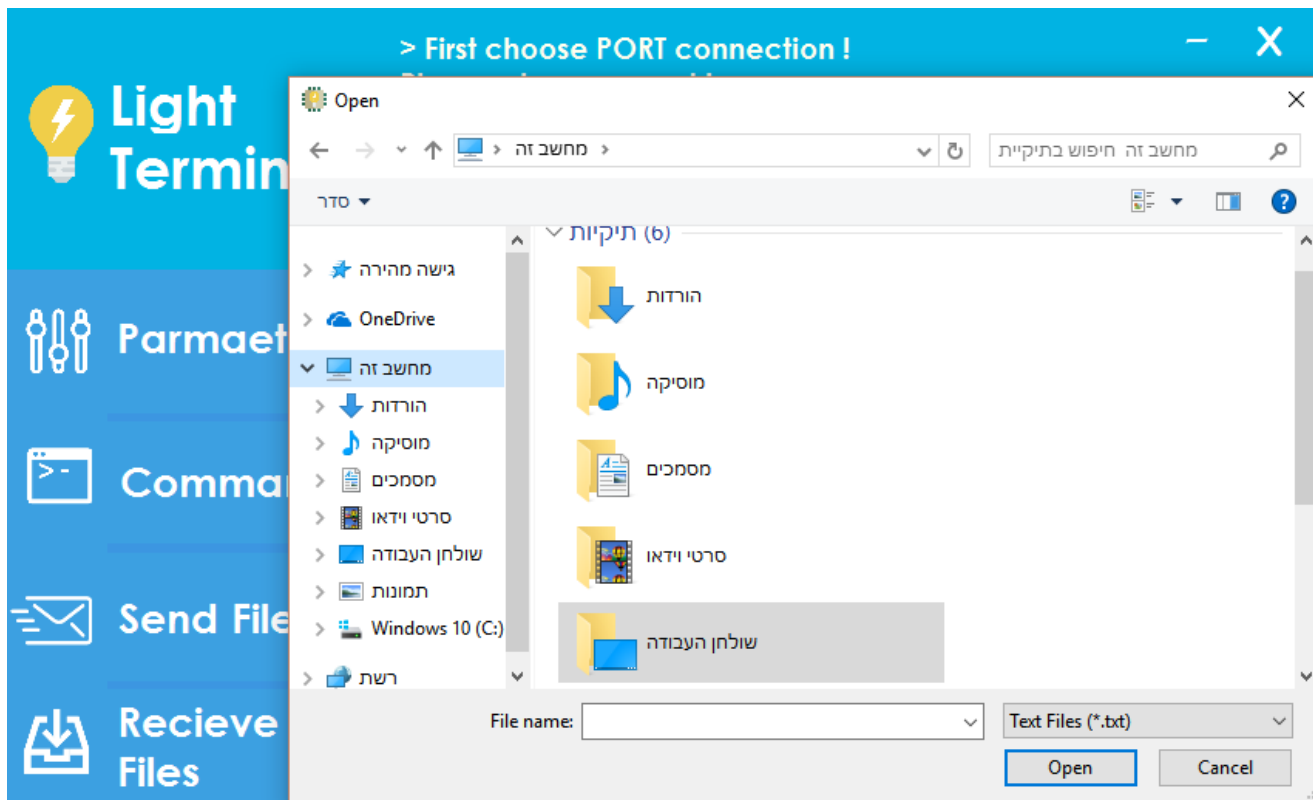
המשתמש בוחר קובץ txt שברצונו לשלוח לבקר, קובץ יכיל אוסף תווים בפורמט ASCII וסיום של קובץ יאופיין ע"י תו EOF מתאים. עבור כל קובץ שנשלח לצד השני הצד השולח יקבל הודעה עם סיום קבלת הקובץ ויציג אותה על המסך. אופן עבודה זה יאפשר העברת קובץ נתונים מהבקר לצד המחשב אחסונו והצגתו ב- PC.



הסמן על כפתור Choose file..

לאחר מכן המשתמש ילחץ על Send להתחלת תהליך ההעברה ובסיומה תוצג הודעה באזור העליון של האפליקציה.

בחירת הקובץ מתבצעת באמצעות פתיחת חלון דיאלוג סטנדרטי לחיפוש קובץ:



```
private void ChoosefileButton_Click(object sender, EventArgs e)
{
    using (OpenFileDialog window = new OpenFileDialog())
    {
        window.Filter = "Text Files (*.txt)|*.txt";
        if (window.ShowDialog() == DialogResult.OK)
        {
            FileNameBox.Text = window.FileName; // show the file name
        }
        SendfileButton.Enabled = true; // you can send the file
    }
}
```

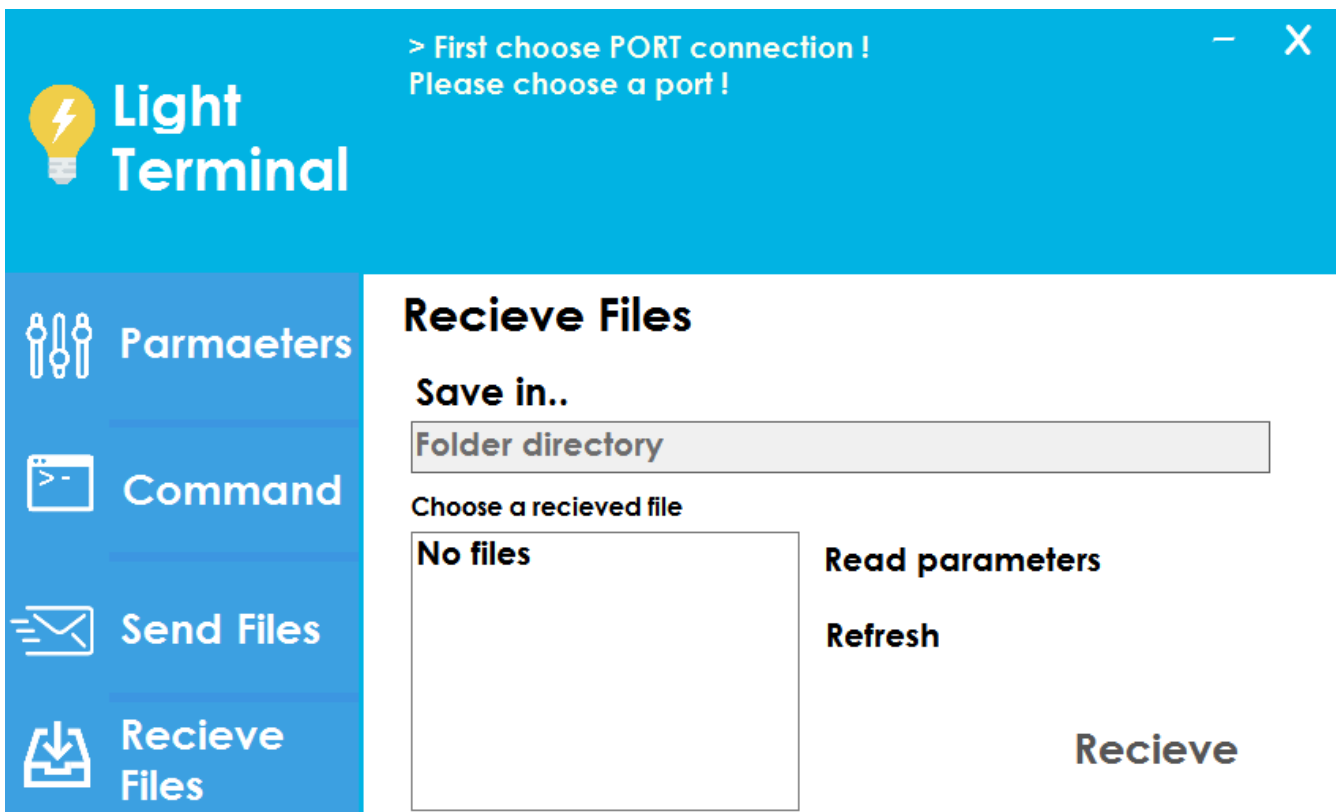
נשים לב שיש מנגנון בדיקה האם הקובץ עם סיומת txt. לאחר בדיקה זו נאפשר את כפתור **Send**.

בשליחת הקובץ ראשית שלח לבקר סימן מוסכם "A" המעיד על תחילת תהליך שליחת הקובץ, לאחר מכן נעזר בפונקציית **FileStream** שמקודדת את הקובץ לבתים. נשלח את גודל הקובץ, שמו והמידע עצמו ונסיים בתו סיום ובהודעה למשתמש.

```
private void SendfileButton_Click(object sender, EventArgs e)
{
    if(FileNameBox.Text== "File Name")
    {
        TopMessageBox.AppendText("Please choose a file ! \n");
        return;
    }
    port.Write("A");
    // using statement is for ensure that even if an exception occurs the code will be execute
    using (FileStream file = new FileStream(FileNameBox.Text, FileMode.Open, FileAccess.Read, FileShare.Read))
    {
        BinaryReader binary = new BinaryReader(file);
        SendBytes = new byte[4];
        byte[] size = BitConverter.GetBytes((int)file.Length);
        SendBytes[0] = size[0];
        SendBytes[1] = size[1];
        SendBytes[2] = size[2];
        SendBytes[3] = size[3];
        port.Write(SendBytes, 0, 4); //send the number of bytes (size) of the file
        port.Write(Path.GetFileName(FileNameBox.Text) + "\n");//send the name of the file
        port.Write(binary.ReadBytes((int)file.Length), 0, (int)file.Length);//send the data of the file
    }
    port.Write(">");
    TopMessageBox.AppendText("File sent ! \n");
}
```

בסיום התהליך תוצג הודעה בצג העליון של האפליקציה שהקובץ נשלח.

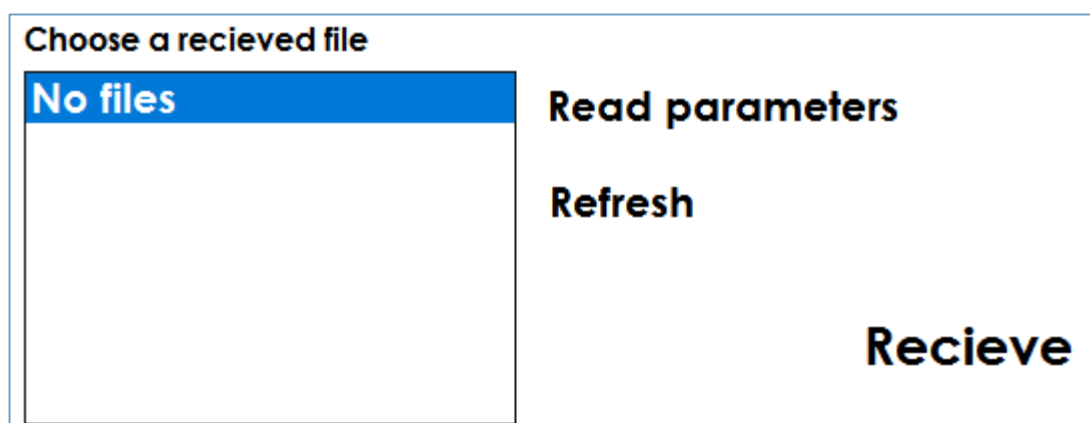
חלון Receive Files



בחלון זה המשתמש יוכל לבחור תיקייה לשמירת קבצים, ולאחר מכן לגשת לקבצים **השמורים בבקר** ולבקש שמירה במחשב.

כפתור Save in.. עובד בצורה זהה לכפתור Choose File.. מהחלון Send Files.

תיבת הקבצים תציג את שמות הקבצים שכבר הועברו לבקר, ואליהם אנו יכולים לבקש העברה חזרה מהבקר.



כפתור **Read parameters** – שולח סימן מוסכם אל הבקר בבקשה לשליחה חזרה אל המחשב את פרמטרי התקשורת העכשוויים. לאחר קבלת המידע (שתחילתו באות מוסכמת "B") נעבד את הנתונים לString ונציג אותה על הצג.

קבלת הפרמטרים אל המחשב מהבקר עם אות מוסכם "U" :

```
case "U": // Recieve controler port parameters
    string parameters = "Baud rate : " + currentPort.ReadLine() + " BPS, "
        + currentPort.ReadLine() + " Data bits,";
    parameters += " No start," + StopBit.ToString() + " Stop bit,";
    currentPort.ReadLine();
    parameters += currentPort.ReadLine() + " Parity";
    sign = currentPort.ReadLine(); // for the final signal
    TopMessageBox.AppendText(parameters);
    break;
```

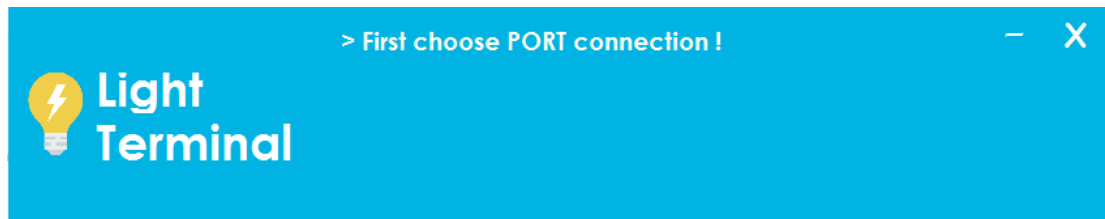
נשים לב שאנו מעבדים את הנתונים שנכנסו לפי הסדר אל תוך String, שאותה בסיום התהליך נציג באזור ההודעות של האפליקציה.

הערה : את הסימונים המוסכמים מהבקר אל המחשב ראשית מקבלים, ורק לאחר מכן מעבדים את הנתונים בצורה המתאימה.

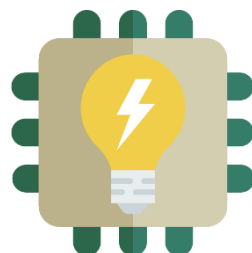
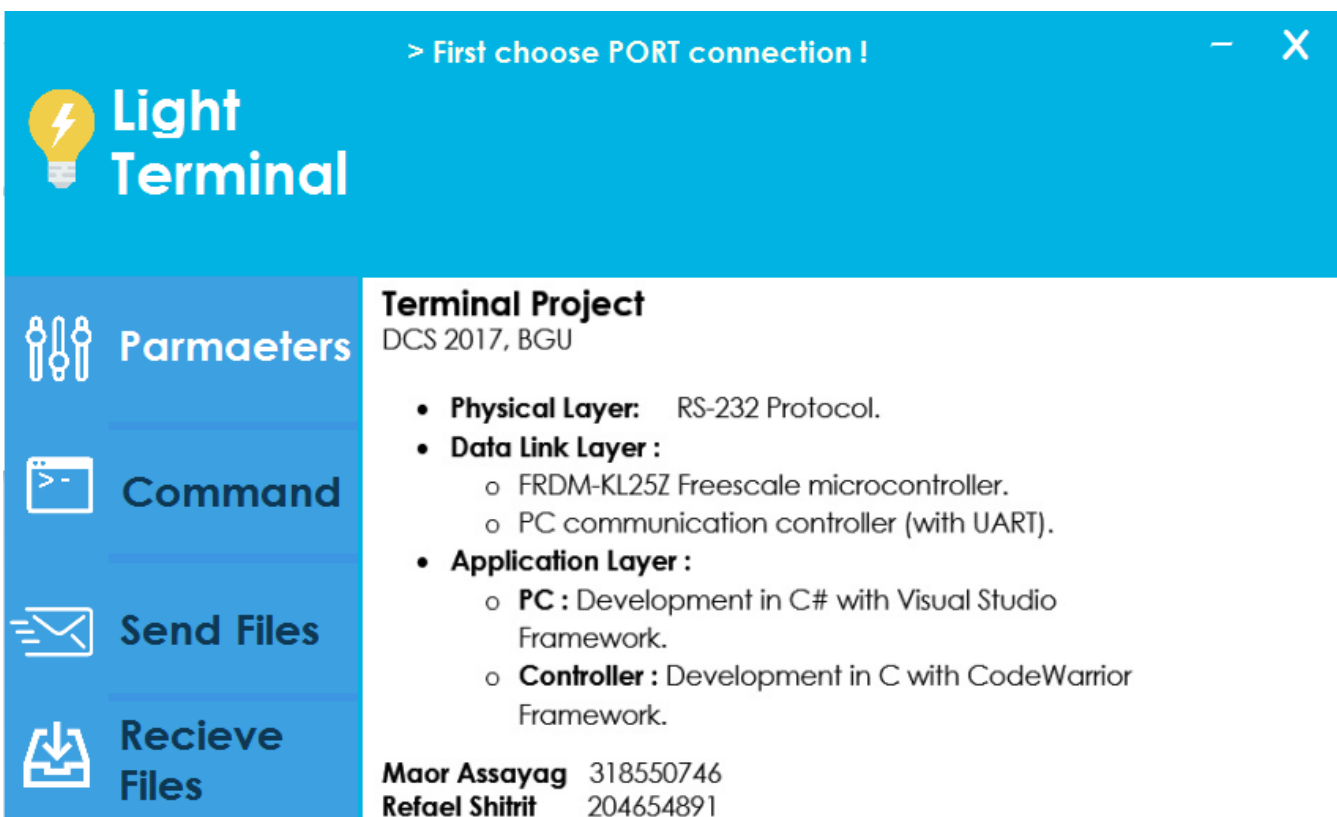
```
private void port_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    SerialPort currentPort = (SerialPort) sender;
    String sign = currentPort.ReadLine();
```


חלון info

בלחיצה על הנורה

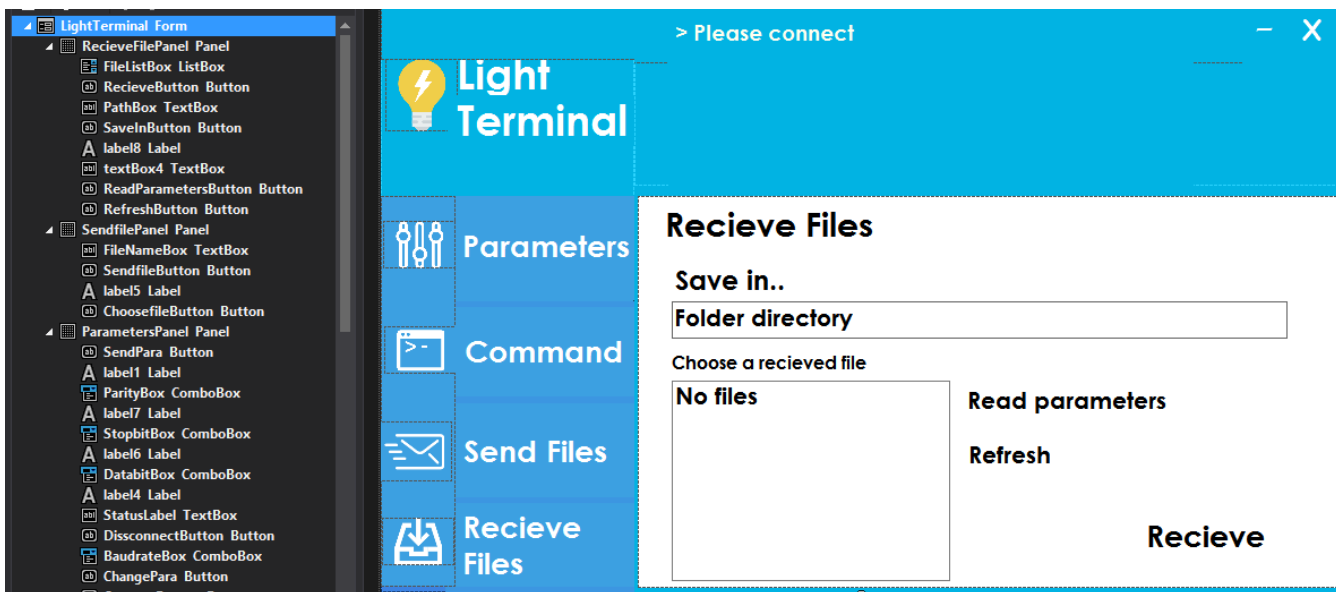


יפתח חלון "מוסתר" המכיל מידע לגבי אופי התשדורת והפרטים המזהים שלנו.

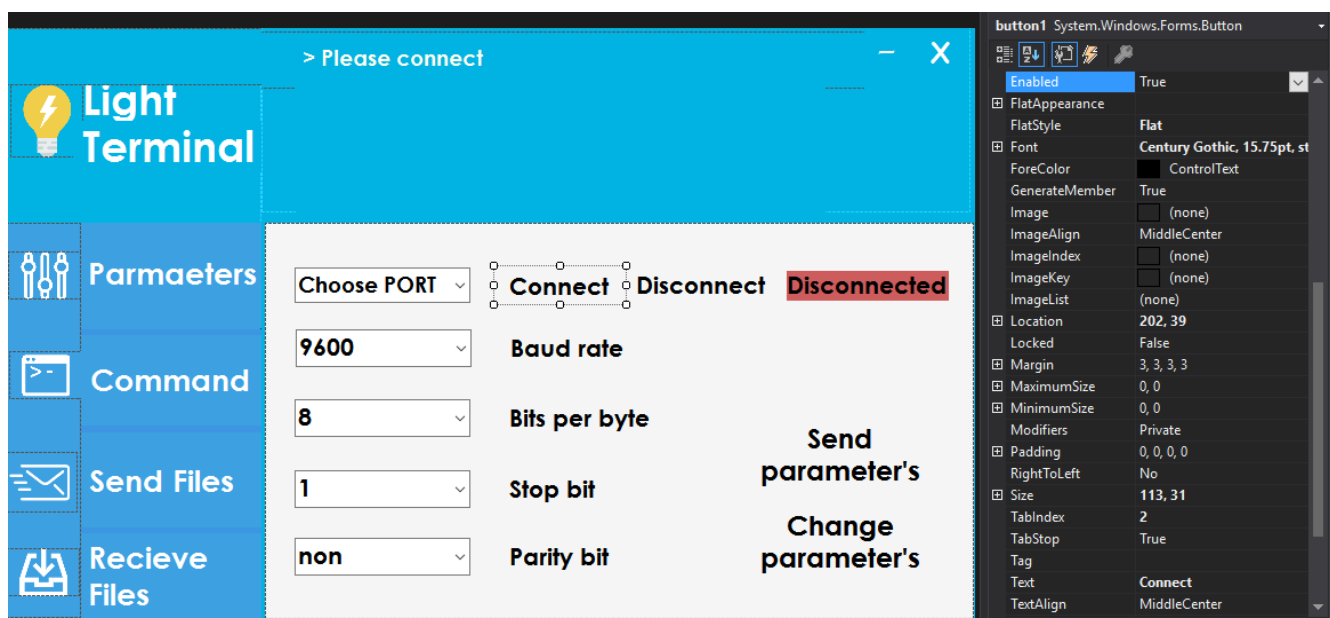


Light Termianl App - icon

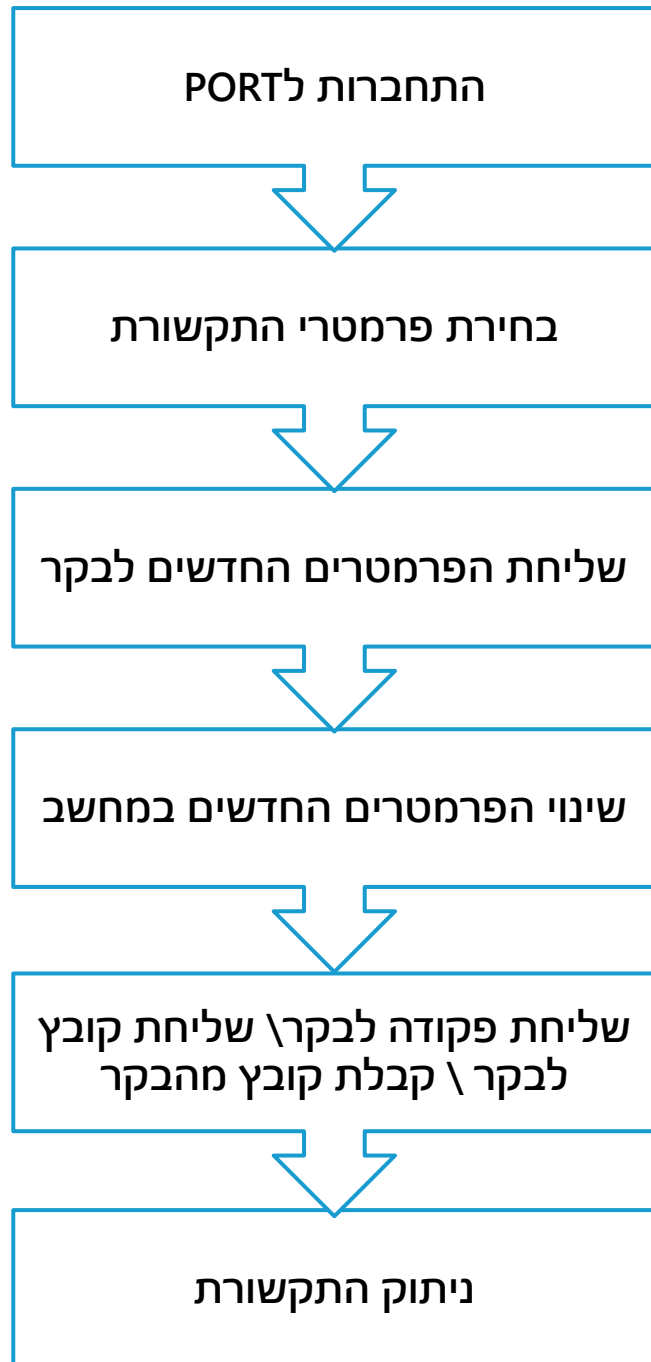
עיצוב ה **UI** (User Interface) כחלונות מרובות התאפשר בזכות היררכית האלמנטים שמציע Visual Studio Form Design



הפיכת הכפתורים (וכן שאר האלמנטים) להיות מאופשרים תחת אילוצים מסוימים, העיצוב שלהם, הEvents השונים שהם מעוררים בלחיצה\סמן העכבר מעליהם התאפשרו בזכות השדות הרבים שסביבת העבודה מציעה.



תרשים זרימה – סדר פעולות צד PC



תיאור הקוד C

פונקציות עיקריות- פירוט מקוצר (ראה קוד בעמודים המפורטים)

קובץ main

- **main :**

מחכה להעלאת הדגל flag ל1, מסמל קבלת מידע ותחילת פעילות מתאימה.

קובץ GameOn

- **sendPara :**

שולחת את הפרמטרים של ה uart למחשב.

- **updatePara :**

מקבלת פרמטרים של ה uart המיועדים לשינוי ומשנה אותם.

- **ChangePara :**

משנה את הפרמטרים הדרושים ב uart. הקריאה תתבצע ממערך המידע שהתקבל, ובתיאום מלא מתכנון האפליקציה ב C#.

- **gameon :**

בהתאם לתו הראשון במידע שנשלח ל UART בתחילת תקשורת חדשה, נוכל לזהות את המשימה שהתבקשה ע"י המשתמש (פקודה לבקר\ קבלת קובץ\ בקשת קובץ\ בקשת שמות הקבצים השונים\ שינוי פרמטרים וכדומה).

קובץ array

- **initarray :**

מאתחלת מבנה נתונים מערך בגודל 400 בתים לקבלת מידע מה UART. גודל זה הינו דינמי בצורה ידנית.

- **Char out()** :

מחזיר את התו הנוכחי במערך עליו מצביע currarray – אינדקס שרץ בצורה מעגלית על המערך.

- **void in(char a)** :

הכנסת תו a למערך המידע במיקום currnt שרץ בצורה מעגלית על המערך.

- **char outT()** :

הוצאת תו ממערך זמני לשמירת המידע של הקובץ במיקום currarryf ודריסת המידע שהיה שם.

- **char inT()** :

הכנסת תו למערך זמני לשמירת המידע של הקובץ במיקום currarryf ודריסת המידע שהיה שם.

- **void inStringT(char* str)** :

הכנסת מחרוזת למערך זמני לשמירת המידע של הקובץ החל ממיקום currarryf ודריסת המידע שהיה שם.

קובץ file

- **addFile** :

ראשית נחלץ מהמידע (המערך הזמני המכיל את מידע הקובץ) את גודלו (ב-C# ראשית נשלח מספר הבתים של מידע הקובץ), נקצה מקום בזיכרון למידע הקובץ. נחלץ את שם הקובץ ונוסיף אותו למערך שמות הקבצים במקום פנוי, נחלץ את המידע ונכניס אותו החל מהכתובת שמצאנו להקצאת המידע – ונעדכן את מערך המצביעים לתחילת המידע של כל קובץ בכתובת ההתחלתית. בכתובת האחרונה נכניס תו סיום מוסכם.

- **sendFile** :

מה-C# נשלח השם של הקובץ הרצוי, נחלץ מידע זה, ונחפש אותו במערך שמות הקבצים. ניגש לאינדקס שמצאנו אך במערך הכתובות – נשלח את מידע הקובץ הרצוי החל מהכתובת ההתחלתית ועד שנמצא תו הסיום.

- **:search**

מחפש את הקובץ שלנו לפי הstring שהכנסנו שמתאר את השם.

- **:filelist**

שולחת לנו את רשימת הקבצים שיש בזיכרון בעזרת מערך שמות הקבצים. בין כל שם נשלח גם '\n' שיאפשר להבדיל את שמות הקבצים על ידי ירידת שורה בport.

נספח – הקוד כמכלול

ניתן לזהות את שמות הקבצים על פי הכותרות, או
לחלופין לפי הסדר הבא :

C

קובץ main	אתחול הפונקציונאליות
קובץ array	מבנה המערך בו מתקבל מידע מהUART
קובץ UART	פונקציות של רכיב ה UART
קובץ GameOn	פעילות בהתאם למידע שהתקבל
קובץ Files	מערך מצביעים לזיכרון, ומערך המכיל את שמות הקבצים בתיאום אינדקסים.

C#

קובץ form LightTerminal	עיצוב ופעולות האפליקציה בצד המחשב
-------------------------	--------------------------------------

```

#include "TFC.h"
#define SDA_SERIAL_BAUD 9600
#define CORE_CLOCK 48000000
//-----
//  UART0 configuration
//-----
void InitUARTs(){

    SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK; // Make sure clock for PORTA is enabled
    SIM_SCGC4 |= SIM_SCGC4_UART0_MASK; // Enable peripheral clock

    PORTA_PCR1 = PORT_PCR_MUX(2) | PORT_PCR_DSE_MASK; // RX
    PORTA_PCR2 = PORT_PCR_MUX(2) | PORT_PCR_DSE_MASK; // TX

    //Select PLL/2 Clock
    SIM_SOPT2 &= ~(3<<26);
    SIM_SOPT2 &= ~SIM_SOPT2_UART0SRC_MASK;
    SIM_SOPT2 |= SIM_SOPT2_UART0SRC(1);
    SIM_SOPT2 |= SIM_SOPT2_PLLFLLSEL_MASK;

    //We have to feed this function the clock in KHz!
    Uart0_Br_Sbr(CORE_CLOCK/2/1000, SDA_SERIAL_BAUD);
    //Enable receive interrupts

    UART0_C2 = UARTLP_C2_TE_MASK | UART_C2_RE_MASK | UART_C2_RIE_MASK; // Enable Transmitter, Receive interrupt
    set_irq_priority(INT_UART0-16,0);
    enable_irq(INT_UART0-16);

}

//-----
//  UART0 - Selection of BR (Baud Rate) and OSR (Over Sampling Ratio)
//-----
void Uart0_Br_Sbr(int sysclk, int baud){

    uint8 i;
    uint32 calculated_baud = 0;
    uint32 baud_diff = 0;
    uint32 osr_val = 0;
    uint32 sbr_val, uart0clk;
    uint32 baud_rate;
    uint32 reg_temp = 0;
    uint32 temp = 0;

    SIM_SCGC4 |= SIM_SCGC4_UART0_MASK;

    // Disable UART0 before changing registers
    UART0_C2 &= ~(UART0_C2_TE_MASK | UART0_C2_RE_MASK);

    // Verify that a valid clock value has been passed to the function
    if ((sysclk > 50000) || (sysclk < 32))
    {
        sysclk = 0;
        reg_temp = SIM_SOPT2;
    }
}

```



```
    reg_temp &= ~SIM_SOPT2_UART0SRC_MASK;
    reg_temp |= SIM_SOPT2_UART0SRC(0);
    SIM_SOPT2 = reg_temp;

    // Enter infinite loop because the
    // the desired system clock value is
    // invalid!!
    while(1);

}

// Initialize baud rate
baud_rate = baud;

// Change units to Hz
uart0clk = sysclk * 1000;
// Calculate the first baud rate using the lowest OSR value possible.
i = 4;
sbr_val = (uint32)(uart0clk/(baud_rate * i));
calculated_baud = (uart0clk / (i * sbr_val));

if (calculated_baud > baud_rate)
    baud_diff = calculated_baud - baud_rate;
else
    baud_diff = baud_rate - calculated_baud;

osr_val = i;

// Select the best OSR value
for (i = 5; i <= 32; i++)
{
    sbr_val = (uint32)(uart0clk/(baud_rate * i));
    calculated_baud = (uart0clk / (i * sbr_val));

    if (calculated_baud > baud_rate)
        temp = calculated_baud - baud_rate;
    else
        temp = baud_rate - calculated_baud;

    if (temp <= baud_diff)
    {
        baud_diff = temp;
        osr_val = i;
    }
}

if (baud_diff < ((baud_rate / 100) * 3))
{
    // If the OSR is between 4x and 8x then both
    // edge sampling MUST be turned on.
    if ((osr_val > 3) && (osr_val < 9))
        UART0_C5 |= UART0_C5_BOTHEDGE_MASK;

    // Setup OSR value
    reg_temp = UART0_C4;
```

```

    reg_temp &= ~UART0_C4_OSR_MASK;
    reg_temp |= UART0_C4_OSR(osr_val-1);

    // Write reg_temp to C4 register
    UART0_C4 = reg_temp;

    reg_temp = (reg_temp & UART0_C4_OSR_MASK) + 1;
    sbr_val = (uint32)((uart0clk)/(baud_rate * (reg_temp)));

    /* Save off the current value of the uartx_BDH except for the SBR field */
    reg_temp = UART0_BDH & ~(UART0_BDH_SBR(0x1F));

    UART0_BDH = reg_temp |  UART0_BDH_SBR(((sbr_val & 0x1F00) >> 8));
    UART0_BDL = (uint8)(sbr_val & UART0_BDL_SBR_MASK);

//      /* Enable receiver and transmitter */
//      UART0_C2 |= (UART0_C2_TE_MASK
//                  | UART0_C2_RE_MASK );
    }
    else
    {
        // Unacceptable baud rate difference
        // More than 3% difference!!
        // Enter infinite loop!
        while(1);
    }

}

/*****
/*
 * Wait for a character to be received on the specified uart
 *
 * Parameters:
 *   channel      UART channel to read from
 *
 * Return Values:
 *   the received character
 */
char uart_getchar (UART_MemMapPtr channel)
{
    /* Wait until character has been received */
    while (!(UART_S1_REG(channel) & UART_S1_RDRF_MASK));

    /* Return the 8-bit data from the receiver */
    return UART_D_REG(channel);
}

/*****
/*
 * Wait for space in the uart Tx FIFO and then send a character
 *
 * Parameters:
 *   channel      UART channel to send to
 *   ch           character to send
 */
void uart_putchar (UART_MemMapPtr channel, char ch)

```

```

{
    /* Wait until space is available in the FIFO */
    while(!(UART_S1_REG(channel) & UART_S1_TDRE_MASK));

    /* Send the character */
    UART_D_REG(channel) = (uint8)ch;
}
/*****
/*
 * Check to see if a character has been received
 *
 * Parameters:
 *   channel      UART channel to check for a character
 *
 * Return values:
 *   0           No character received
 *   1           Character has been received
 */
int uart_getchar_present (UART_MemMapPtr channel)
{
    return (UART_S1_REG(channel) & UART_S1_RDRF_MASK);
}
/*****
/*
 * Wait for space in the uart Tx FIFO and then send a string
 *
 * Parameters:
 *   channel      UART channel to send to
 *   str          string to send
 */
void UARTprintf(UART_MemMapPtr channel,char* str){
    volatile unsigned char i;

    for (i=0 ; str[i] ; i++){

        while(!(UART_S1_REG(channel) & UART_S1_TDRE_MASK)); /* Wait until space
        is available in the FIFO */

        UART_D_REG(channel) = str[i]; /* Send the character */
    }
}
void UART0_IRQHandler(){

    uint8_t Temp;

    if(UART0_S1 & UART_S1_RDRF_MASK){ // RX buffer is full and ready for reading

        Temp=UART0_D;
        if(Temp=='>')
            f++;
        in(Temp);
    }
    if((UART0_S1 & UART_S1_TDRE_MASK)&(UART0_C2 & UART_C2_TIE_MASK)){ // TX buffer
    is empty and ready for writing
        UART_D_REG(UART0_BASE_PTR) =outT();
    }
}

```

```
        if(file[currarryf]=='&')
            UART0_C2 &= ~UART_C2_TIE_MASK;
    }

}

void Ack(){
    while(!(UART0_S1 & UART_S1_TDRE_MASK));
    UART_D_REG(UART0_BASE_PTR)='a';
    while(!(UART0_S1 & UART_S1_TDRE_MASK));
    UART_D_REG(UART0_BASE_PTR)=0x0A;
    while(!(UART0_S1 & UART_S1_TDRE_MASK));
    UART_D_REG(UART0_BASE_PTR)=0x1A;
    while(!(UART0_S1 & UART_S1_TDRE_MASK));
    UART_D_REG(UART0_BASE_PTR)=0x0A;
    while(!(UART0_S1 & UART_S1_TDRE_MASK));
}
```

```
/*
 * array.c
 *
 * Created on: Aug 08, 2017
 * Author: refael
 */

#include "ARRAY.h"

void initarray(){
    int i;
    for (i = 0; i < 400; i++) {
        game[i]='&';
        file[i]='&';
    }
}

char out(){
    char a = game[currarry];
    game[currarry]='&';
    currarry=(currarry+1)%400;
    return a;
}

void in(char a){
    game[currnt]=a;
    currnt=(currnt+1)%400;
}

char outT(){
    char a = file[currarryf];
    file[currarryf]='&';
    currarryf=(currarryf+1)%400;
    return a;
}

void inT(char a){
    file[currntf]=a;
    currntf=(currntf+1)%400;
}

void inStringT(char* str){
    char* temp=str;
    while(*temp!='\n'&&*temp!='\0'){
        inT(*temp);
        temp++;
    }
    inT(*temp);
}
```

```
# include "TFC.h"
void addFile(){
    i=0;
    PIT_TCTRL0 &= ~PIT_TCTRL_TEN_MASK;//enable PIT0
    char sizearr[4];
    int j;
    for(j=0;j<4;j++){
        sizearr[j]=out();
    }
    int* size=&sizearr;
    char name [40];
    char *filenametemp=malloc(41);
    filename[indexfilename]=filenametemp;//array of name
    char a=out();
    while(a!='\n'){
        *filenametemp=a;
        filenametemp++;
        a=out();
    }
    *filenametemp='\0';
    indexfilename++;
    char *filerem=malloc(*size+1);
    fileob[indexfileob]=filerem;//array of pointers to the memo
    char b;
    b=out();
    while(b!='>'){
        *filerem=b;
        filerem++;
        if(b=='s'){
            color=1;
        }
        if(b=='d'){
            if(game[(count+1)%400]=='e')
                del=1;
        }
        if(color==1)
            if(b=='w' || b=='r' || b=='b' || b=='y' || b=='a' || b=='p' || b=='g')
                Runfilechar(b);
            if(b>='0' && b<='9')
                Rundel(b);
        b=out();
    }
    *filerem=b;
    numOfFile++;
    i=0;
    delstap=0;
    PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK;//enable PIT0
}

void sendFile(){
    char name[40];
    char b=out();
    int index=0;
    while(b!='>'){
        name[index]=b;
        b=out();
        index++;
    }
}
```

```
    }
    name[index]='\0';
    int indextemp=search(name);
    char* temp=fileob[indextemp];
    while(*temp!='>'){
        inT(*temp);
        temp++;
    }
    inT(0x1A);
    UART0_C2 |= UART_C2_TIE_MASK; //enable transmit interrupt for start sending
}

int search(char* name){
    int temp=0;
    while(temp<numOfFile){
        if(strcmp(filename[temp],name)==0){
            return temp;
        }
        temp++;
    }
    return;
}

void filelist(){
    int countemp=0;
    if(numOfFile==0){
        inT(' ');
        inT('\n');
    }
    else while(countemp<numOfFile){
        inStringT(filename[countemp]);
        countemp++;
        inT(0x1A);
        inT('\n');
    }
    inT('>');
    inT('\n');

    UART0_C2 |= UART_C2_TIE_MASK; //enable transmit interrupt for start sending
}
```

```

/*
 * GameOn.c
 *
 * Created on: Aug 16, 2017
 * Author: refael
 */
#include "GAMEON.h"
void updatePara(){
    UART0_C2 &=~UART_C2_TIE_MASK;
    char baudratePara[4];
    int PSSBPara[4];
    int i;
    // get the baud rate and ppsb
    for(i=0;i<8;i++){
        if(i<4)
            baudratePara[i]=out();
        else
            PSSBPara[i-4]=(int)out();
    }
    //send ack
    out();
    Ack();
    int baudRate=(baudratePara[0]-'0')*1000+(baudratePara[1]-'0')*100+(baudratePara[2]-'0')*10+(baudratePara[3]-'0')*1;
    ChangePara(PSSBPara,baudRate);
    UART0_C2 |= UART0_C2_TE_MASK |UART0_C2_RE_MASK;
}
void ChangePara(int ChangeArr[],int newbaudRate){
    UART0_C2 &= ~(UART0_C2_TE_MASK |UART0_C2_RE_MASK);
    //update the baud rate
    Uart0_Br_Sbr(48000000/2/1000, newbaudRate);
    if(ChangeArr[0]<2){
        UART0_C4 &=~UART0_C4_M10_MASK;// change the bit of 10 bit trans
        if(ChangeArr[0]==0)
            UART0_C1 &=~UART0_C1_M_MASK;// 8 bit trans
        else{
            UART0_C1 |=UART0_C1_M_MASK;// 9 bit trans
            RED_LED_ON;
        }
    }else{// 10 bit trans
        UART0_C1 &=~UART0_C1_M_MASK;
        UART0_C4 |=UART0_C4_M10_MASK;
    }
    if(ChangeArr[1]==1)
        UART0_BDH &=~UART0_BDH_SBNS_MASK;//1 stop bit
    else if(ChangeArr[1]==2)
        UART0_BDH |=UART0_BDH_SBNS_MASK;//2 stop bit
    if(ChangeArr[2]==0){
        UART0_C1 &=~UART0_C1_PE_MASK;//no parity bit
    }else{
        UART0_C1 |=UART0_C1_PE_MASK;//yes parity bit
        if(ChangeArr[2]==1){
            UART0_C1 &=~UART0_C1_PT_MASK;//even bit
        }else if(ChangeArr[2]==2){
            UART0_C1 |=UART0_C1_PT_MASK;// odd bit
        }
    }
}

```



```
}

}
void gameon(){
    char a=out();
    RGB_LED_OFF;
    PIT_TCTRL0 &= ~PIT_TCTRL_TEN_MASK;//enable PIT0
    switch (a){
        case '1':
            RGB_LED_OFF;
            RED_LED_ON;
            out();
            break;
        case '2':
            RGB_LED_OFF;
            BLUE_LED_ON;
            out();
            break;
        case '3':
            RGB_LED_OFF;
            RED_LED_ON;
            GREEN_LED_ON;
            out();
            break;
        case '4':
            RGB_LED_OFF;
            GREEN_LED_ON;
            out();
            break;
        case '5':
            RGB_LED_OFF;
            RED_LED_ON;
            BLUE_LED_ON;
            out();
            break;
        case '6':
            RED_LED_ON;
            GREEN_LED_ON;
            BLUE_LED_ON;
            out();
            break;
        case '7':
            RGB_LED_OFF;
            BLUE_LED_ON;
            GREEN_LED_ON;
            out();
            break;
        case '8':
            inStringT("l\n");
            filelist();
            out();
            break;
        case '9':
            inStringT("S\n");
            sendFile();
            out();
    }
}
```

```

        break;
        case 'A':
            addFile();
        break;
        case 'B':
            updatePara();
            out();
        break;
        case 'R':
            __iar_program_start();
        break;
        case 'C':
            RGB_LED_OFF;
            out();
        break;
    }
    f--;
}
void Runfilechar(char a){
    switch(a){
        case 'r':
            funarry[i]=1;
        break;
        case 'w':
            funarry[i]=2;
        break;
        case 'b':
            funarry[i]=3;
        break;
        case 'y':
            funarry[i]=4;
        break;
        case 'a':
            funarry[i]=5;
        break;
        case 'p':
            funarry[i]=6;
        break;
        case 'g':
            funarry[i]=7;
        break;
    }
    color=0;
    i++;
    count++;
}
void Rundel(char a){
    Delaysum[delarry]=a;
    if(delarry==3){
        int* delayNum=&Delaysum;
        funarry[i]=(Delaysum[0]-'0')*1000+(Delaysum[1]-'0')*100+(Delaysum[2]-'0')
            *10+(Delaysum[3]-'0')*1;
        i++;
        count++;
        del=0;
        delstap=0;
    }
}

```

```
    delarry=0;
}else
    delarry++;
}
```

```
/*#include "derivative.h" */ include peripheral declarations */
#include "TFC.h"
int main(void){
    //timer
    ClockSetup();
    InitPIT();
    //led
    InitGPIO();
    //uart
    InitUARTs();
    //data structures
    initarray();
    while(1){
        if(f>0){
            gameon();
        }
    }
    return 0;
}
```

```
using System;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;
using System.Threading;

//*****//
// '>' is the recognized sign for the End of DATA in a current communicaiton //
//*****//

namespace Terminal
{
    public partial class LightTerminal : Form
    {
        #region Fields
        //-----//

        System.IO.Ports.SerialPort port;
        byte[] SendBytes;
        //default parameters;
        string portName;
        int baudRate = 9600;
        int WordLength = 8;
        Parity actualParity = Parity.None;
        StopBits actualStopBit = StopBits.One;
        //for send parameters
        int paritySend;
        int StopBit;
        bool PcParaChange = false;
        bool ControllerParaChange = false;

        //-----//
        #endregion

        #region Initialization
        //-----//

        public LightTerminal()
        {
            InitializeComponent();
            TopMessageBox.Text = "> First choose PORT connection !\n";
        }

        //-----//
        #endregion

        #region Port methodes
        //-----//

        private void openPort() // open port by parameters
        {
            port.BaudRate = baudRate;
            port.Parity = actualParity;
            port.StopBits = actualStopBit;
```

```

        port.DataBits = WordLength;
        port.Open();
        port.DtrEnable = true;
        port.RtsEnable = true;
        try
        {
            PcParaChange = true; //the parameters change in the computer
            if (PcParaChange && ControllerParaChange) { openCommunication
                (); } // only than apply communication
        }
        catch
        {
            TopMessageBox.AppendText("Something didn't work, Please check your
                connection !\n");
        }
    }
    private void openCommunication()
    {
        TopMessageBox.Text = "> Connected to " + portName;
        ConnectButton.Enabled = false;
        DisconnectButton.Enabled = true;
        BaudrateBox.Enabled = true; //baud rate
        DatabitBox.Enabled = true; //BFS
        StopbitBox.Enabled = true; //stop bit
        ParityBox.Enabled = true; //pairty box
        CommandBox.Enabled = true; //otherwise you cant write commands
        CommandPanelButton.Enabled = true; //you can inter the command
        SendPanelButton.Enabled = true; // files enabled
        RecievePanelButton.Enabled = true;
        StatusLabel.Text = "Connected";
        StatusLabel.BackColor = Color.LightGreen;
        PcParaChange = false;
        ControllerParaChange = false;
        SendPara.Enabled = false;
        ChangePara.Enabled = false;
        ReadParametersButton.Enabled = true;
    }
    private void closePort()
    {
        if (port != null && port.IsOpen)
        {
            port.DiscardInBuffer();
            port.Dispose();
            port.Close(); // close the port
        }
        StatusLabel.Text = "Disconnected";
        StatusLabel.BackColor = Color.IndianRed;
        DisconnectButton.Enabled = false;
        BaudrateBox.Enabled = false; //baud rate
        DatabitBox.Enabled = false; //BFS
        StopbitBox.Enabled = false; //stop bit
        ParityBox.Enabled = false; //pairty box
        CommandPanelButton.Enabled = false; //you can inter the command
        SendPanelButton.Enabled = false; // files enabled
        RecievePanelButton.Enabled = false;
        SendPara.Enabled = false;
    }

```

```

    if (ControllerParaChange != true)
    { ConnectButton.Enabled = true; }
}

//-----//
#endregion

#region DataReceived
//-----//

private void port_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    SerialPort currentPort = (SerialPort) sender;
    Thread.Sleep(200);
    try
    {
        String sign = currentPort.ReadLine();
        switch (sign)
        {
            case "a": //port parameters changed
                currentPort.DiscardInBuffer();// discard current data, if exist
                TopMessageBox.AppendText("Controller Parameters changed !");
                closePort(); //close the port
                ChangePara.Enabled = true; // enabled change in pc parameters
                SendPara.Enabled = false;
                ControllerParaChange = true;// the controller has change his parameters
                break;

            case "l": //recieve list of files
                FileListBox.BeginInvoke(new EventHandler(delegate { FileListBox.Items.Clear(); }));
                string list = currentPort.ReadLine();
                if (list == " ") { FileListBox.Items.Add("No files"); }
                else
                {
                    while (list != ">") // ='>'.
                    {
                        FileListBox.BeginInvoke(new EventHandler(delegate { FileListBox.Items.Add(list.ToString()); }));
                        Thread.Sleep(150);
                        list = currentPort.ReadLine();
                    }
                }
                break;

            case "U": // Recieve controler port parameters - optional
                string parameters = "Baud rate : " + currentPort.ReadLine() + " BPS, "
                    + currentPort.ReadLine() + " Data bits,";
                parameters += " 1 Start," + StopBit.ToString() + " Stop bit,";
        }
    }
}

```

```

        currentPort.ReadLine();
        parameters += currentPort.ReadLine() + " Parity";
        sign = currentPort.ReadLine(); // for the final signal '>'
        TopMessageBox.AppendText(parameters);
        break;

    case "S": // Recieve file
        string current = currentPort.ReadExisting();
        current = current.Substring(0, current.Length - 3); // dont
        save the last char '>' and 0x1A
        string path = PathBox.Text;
        string namefile = FileName.Text + ".txt";
        int Duplicate = 0;
        try
        {
            while (File.Exists(Path.Combine(path, namefile)))
            {
                Duplicate++;
                namefile = FileName.Text + Duplicate.ToString() +
                ".txt";
            }

            using (FileStream fileToSave = File.Create(Path.Combine
            (path, namefile)))
            {
                Byte[] data = new UTF8Encoding(true).GetBytes
                (current);
                fileToSave.Write(data, 0, data.Length);
            }
            TopMessageBox.AppendText("File saved !");
        }
        catch { TopMessageBox.AppendText("Error oucerd !"); }
        break;

    case "D":
        TopMessageBox.AppendText("File saved in the controller !
        \n");
        currentPort.ReadLine();
        break;

    default: // discard current data, if exist
        currentPort.ReadExisting();
        break;
    }
}
catch { port.DiscardInBuffer(); }
}

//-----//
#endregion

#region UserInterface
//-----//

private void ConnectButton_Click(object sender, EventArgs e)
{

```



```

        if (PortBox.Text == "Choose PORT")
        {
            TopMessageBox.AppendText("Please choose a port !");
        }
        else
        {
            port = new System.IO.Ports.SerialPort(portName, baudRate,
                actualParity, WordLength, actualStopBit);
            port.ReadTimeout = 2000;
            port.ReceivedBytesThreshold = 1500;
            port.Open();
            port.DataReceived += new SerialDataReceivedEventHandler
                (port_DataReceived);
            openCommunication();
        }
    }

    private void PortBox_DropDown(object sender, EventArgs e)
    {
        string[] ports = SerialPort.GetPortNames();
        PortBox.Items.Clear();
        for (int i = 0; i < ports.Length; i++) // Display each port name to the
            console.
        {
            PortBox.Items.Add((string)ports[i]);
        }
    }

    private void ChangePara_Click(object sender, EventArgs e)
    {
        closePort();
        openPort();
        TopMessageBox.AppendText("Connected with new parameters !\n");
    }

    private void ReadParametersButton_Click(object sender, EventArgs e)
    {
        //print the parameters
        TopMessageBox.Clear();
        TopMessageBox.AppendText("Baud rate : " + baudRate.ToString() + " BPS,
            " + WordLength.ToString() + " Data bits," +
            "1 Start," + StopBit.ToString() + " Stop bit," + actualParity.ToString
                () + " Parity");
    }

    private void DissconnectButton_Click(object sender, EventArgs e)
    {
        closePort();
    }

    private void CommandBox_KeyUp(object sender, KeyEventArgs userKey)
    {
        if (userKey.KeyCode == Keys.Return) // Enter key
        {
            switch (CommandBox.Lines[CommandBox.Lines.Length - 2])// the last
                line(with indes [Length-1]) will be empty (the user press enter)

```

```

{
    case "set red":
        port.Write("1");
        port.Write(">");
        break;
    case "set blue":
        port.Write("2");
        port.Write(">");
        break;
    case "set yellow":
        port.Write("3");
        port.Write(">");
        break;
    case "set green":
        port.Write("4");
        port.Write(">");
        break;
    case "set purpule":
        port.Write("5");
        port.Write(">");
        break;
    case "set white":
        port.Write("6");
        port.Write(">");
        break;
    case "set azule":
        port.Write("7");
        port.Write(">");
        break;
    case "clear rgb":
        port.Write("C");
        port.Write(">");
        break;
    case "reset":
        port.Write("R");
        port.Write(">");
        break;
    default: // else
        TopMessageBox.Clear();
        TopMessageBox.AppendText(" Valid Command's : ");
        TopMessageBox.AppendText(" 1. set red - Turn on red led ");
        TopMessageBox.AppendText(" 2. set blue - Turn on blue led ");
        TopMessageBox.AppendText(" 3. set yellow - Turn on yellow ");
        TopMessageBox.AppendText(" 4. set green - Turn on green led ");
        TopMessageBox.AppendText(" 5. set azule - Turn on azule led ");
        TopMessageBox.AppendText(" 6. set white - Turn on white led ");
        TopMessageBox.AppendText(" 7. set purple - Turn on purple ");
        TopMessageBox.AppendText(" 8. clear rgb - Turn off rgb led ");
        TopMessageBox.AppendText(" 9. reset - To reset the ");

```

```

        controller ");
        TopMessageBox.AppendText(" 10. in Script only : delay X
        milliseconds ");
        break;
    }
}

private void SendfileButton_Click(object sender, EventArgs e)
{
    if(FileNameBox.Text== "File Name")
    {
        TopMessageBox.AppendText("Please choose a file !");
        return;
    }
    port.Write("A");
    // using statement is for ensure that even if an exception occurs the
    // code will be execute
    using (FileStream file = new FileStream(FileNameBox.Text,
        FileMode.Open, FileAccess.Read, FileShare.Read))
    {
        BinaryReader binary = new BinaryReader(file);
        SendBytes = new byte[4];
        byte[] size = BitConverter.GetBytes((int)file.Length);
        SendBytes[0] = size[0];
        SendBytes[1] = size[1];
        SendBytes[2] = size[2];
        SendBytes[3] = size[3];
        port.Write(SendBytes, 0, 4); //send the number of bytes (size) of
        // the file
        port.Write(Path.GetFileName(FileNameBox.Text) + "\n");//send the
        // name of the file
        port.Write(binary.ReadBytes((int)file.Length), 0, (int)
        // file.Length); //send the data of the file
    }
    port.Write(">");
    TopMessageBox.AppendText("File sent !");
}

private void RecieveButton_Click(object sender, EventArgs e)
{
    if (FileListBox.SelectedItem.ToString()=="No files")
    {
        TopMessageBox.AppendText("Please refresh the list and choose a
        file !");
    }
    else if (Directory.Exists(PathBox.Text))
    {
        port.Write("9"); // ask for this file
        port.Write(FileListBox.SelectedItem.ToString()); // send
        // filename.txt
        port.Write(">");
    }
    else
    {
        TopMessageBox.AppendText("Please choose a valid directory !");
    }
}

```

```
    }  
}  
  
private void ChoosefileButton_Click(object sender, EventArgs e)  
{  
    using (OpenFileDialog window = new OpenFileDialog())  
    {  
        window.Filter = "Text Files (*.txt)|*.txt";  
        if (window.ShowDialog() == DialogResult.OK)  
        {  
            FileNameBox.Text = window.FileName; // show the file name  
        }  
    }  
    SendfileButton.Enabled = true; // you can send the file  
}  
  
private void SaveInButton_Click(object sender, EventArgs e)  
{  
    using (FolderBrowserDialog window = new FolderBrowserDialog())  
    {  
        if (window.ShowDialog() == DialogResult.OK)  
        {  
            PathBox.Text = window.SelectedPath;  
        }  
    }  
    RecieveButton.Enabled = true;  
}  
  
private void RefreshButton_Click(object sender, EventArgs e)  
{  
    port.Write("8"); // ask for the list of files names  
    port.Write(">");  
}  
  
private void TopMessageBox_TextChanged(object sender, EventArgs e)  
{  
    TopMessageBox.AppendText("\r\n");  
    TopMessageBox.ScrollToCaret();  
}  
  
private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)  
{  
    SendPara.Enabled = true;  
    PcParaChange = false;  
    ControllerParaChange = false;  
}  
  
private void CommandPanelButton_Click(object sender, EventArgs e)  
{  
    CommandPanel.Visible = true; // command page  
    ParametersPanel.Visible = false;  
    SendfilePanel.Visible = false;  
    RecieveFilePanel.Visible = false;  
    InfoPanel.Visible = false;  
    CommandBox.Enabled = true;  
}
```

```
private void InfoButton_Click_1(object sender, EventArgs e)
{
    InfoPanel.Visible = true; // info page
    ParametersPanel.Visible = false;
    CommandPanel.Visible = false;
    SendfilePanel.Visible = false;
    RecieveFilePanel.Visible = false;
}

private void ParametersPanelButton_Click(object sender, EventArgs e)// parameter button
{
    ParametersPanel.Visible = true; //parameter page
    CommandPanel.Visible = false;
    SendfilePanel.Visible = false;
    RecieveFilePanel.Visible = false;
    InfoPanel.Visible = false;
}

private void PortBox_SelectedIndexChanged(object sender, EventArgs e)
{
    portName = PortBox.Text;
}

private void SendPanelButton_Click(object sender, EventArgs e)//files button
{
    SendfilePanel.Visible = true; // send files
    ParametersPanel.Visible = false;
    CommandPanel.Visible = false;
    RecieveFilePanel.Visible = false;
    InfoPanel.Visible = false;
}

private void RecievePanelButton_Click(object sender, EventArgs e)//info button
{
    RecieveFilePanel.Visible = true; // recevie file
    ParametersPanel.Visible = false;
    CommandPanel.Visible = false;
    SendfilePanel.Visible = false;
    InfoPanel.Visible = false;
}

private void ExitButton_Click_1(object sender, EventArgs e)
{
    this.Close();
}

private void MinimizeButton_Click(object sender, EventArgs e)
{
    this.WindowState = FormWindowState.Minimized;
}

// mouse can move the form
int mouseX = 0, mouseY = 0;
bool mouseDown;
private void pictureBox6_MouseMove(object sender, MouseEventArgs e)
{

```

```
        if (mouseDown)
        {
            mouseX = MousePosition.X - 300 ;
            mouseY = MousePosition.Y - 4;
            this.SetDesktopLocation(mouseX, mouseY);
        }
    }

    private void pictureBox6_MouseUp(object sender, MouseEventArgs e)
    { mouseDown = false; }

    private void SendPara_Click(object sender, EventArgs e)
    {
        SendBytes = new byte[8];
        // Starting sign - 'B'
        SendBytes[0] = BitConverter.GetBytes('B')[0];
        //Baud Rate
        byte[] temp = BitConverter.GetBytes(baudRate);
        SendBytes[1] = temp[0];
        SendBytes[2] = temp[1];
        SendBytes[3] = temp[2];
        SendBytes[4] = temp[3];
        //WordLength
        SendBytes[5] = BitConverter.GetBytes(8- WordLength)[0]; // 0,1,2,3 for 7
        //8,7,6,5 data bit
        //Stop Bit
        SendBytes[6] = BitConverter.GetBytes(StopBit)[0];
        //Parity Bit
        SendBytes[7] = BitConverter.GetBytes(paritySend)[0];
        port.Write(SendBytes, 0, 8);
        port.Write(">");
        TopMessageBox.AppendText("Parameters has been sent !");
    }

    private void BaudrateBox_SelectedValueChanged(object sender, EventArgs e)
    {
        if (baudRate != Int32.Parse(BaudrateBox.Text))
        {
            baudRate = Int32.Parse(BaudrateBox.Text);
            SendPara.Enabled = true;
            PcParaChange = false;
            ControllerParaChange = false;
        }
    }

    private void DatabitBox_SelectedValueChanged(object sender, EventArgs e)
    {
        if (WordLength != int.Parse(DatabitBox.SelectedItem.ToString()))
        {
            WordLength = int.Parse(DatabitBox.SelectedItem.ToString());
            SendPara.Enabled = true;
            PcParaChange = false;
            ControllerParaChange = false;
        }
    }
}
```

```
private void StopbitBox_SelectedValueChanged(object sender, EventArgs e)
{
    if (StopBit != int.Parse(StopbitBox.SelectedItem.ToString()))
    {
        StopBit = int.Parse(StopbitBox.SelectedItem.ToString());
        switch (StopbitBox.SelectedItem.ToString())
        {
            case "1":
                actualStopBit = StopBits.One;
                break;
            case "2":
                actualStopBit = StopBits.Two;
                break;
        }
        SendPara.Enabled = true;
        PcParaChange = false;
        ControllerParaChange = false;
    }
}

private void ParityBox_SelectedValueChanged(object sender, EventArgs e)
{
    if (paritySend != ParityBox.SelectedIndex)
    {
        String parity = ParityBox.SelectedItem.ToString();
        switch (parity)
        {
            case "None":
                actualParity = Parity.None;
                paritySend = 0;
                break;
            case "Even":
                actualParity = Parity.Even;
                paritySend = 1;
                break;
            case "Odd":
                actualParity = Parity.Odd;
                paritySend = 2;
                break;
        }
        SendPara.Enabled = true;
        PcParaChange = false;
        ControllerParaChange = false;
    }
}

private void pictureBox6_MouseDown(object sender, MouseEventArgs e)
{
    mouseDown = true; }

//-----//
#endregion
}
```