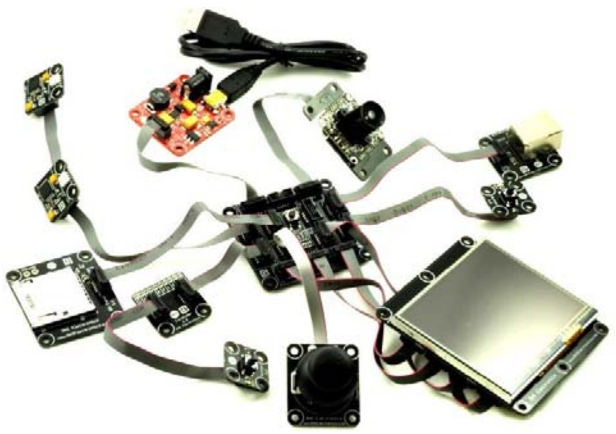
System Design Specification

For:

Communication System

A FEZ Spider Application



TA:

Guido Albertengo

Development Group:

Jiawei Wu

Luca Graglia

Date:

6/11/2014

Document History

Version Number	Date Updated	Author	Brief Description of Changes
0.1	5/5/2014	Jiawei Wu	Started SDS from template, up to chapter 2.2
0.2	5/7/2014	Jiawei Wu	Add chapter 2.4, 2.5, 2.6
0.3	5/9/2014	Jiawei Wu	Add chapter 4, 5
0.4	5/14/2014	Jiawei Wu	Fill chapter 5.1,5.2,5.3.3,5.3.4,5.3.5,5.3.6,5.3.7
0.5	5/16/2014	Jiawei Wu	Modify the content about email sending function in chapter 1.4
0.6	5/21/2014	Jiawei Wu	Add and fill chapter 6.2.1
0.7	5/22/2014	Jiawei Wu	Modify 5.4.6 parent class name into ServerWin Add chapter 5.4.7 Email Attachment Setting Full fill chapter 5.4.3, 5.4.5, 5.4.7,5.4.8 Delete chapter 8 Finish the definition for server side
0.8	5/30/2014	Jiawei Wu	Fill chapter 3.1
0.9	6/4/2014	Jiawei Wu	Fill chapter 3.2, 3.3, 3.4, 3.5 Add chapter 3.6 Delete image taken Change figure1 in section 2.1, delete the node of button Modify section 2.2, add feature of delete image taken Fill chapter 5.3.1
1.0	6/11/2014	Jiawei Wu	Modify section 2.2 change to selected image Modify section 3.1 authentication Fill section 5.3.1, 5.3.2 Fill section 5.4.1, 5.4.2 Add chapter 8 Summary Complete

Table of Contents

1 Introduction 1
- 1.1 Purpose 1
- 1.2 Document Conventions 1
- 1.3 Intended Audience and Reading Suggestions 1
- 1.4 Project Scope 1
- 1.5 References 1

2 Overall Description 2
- 2.1 Product Perspective 2
- 2.2 Product Features 2
- 2.3 User Classes and Characteristics 3
- 2.4 Operating Environment 3
- 2.5 Design and Implementation Constraints 3
- 2.6 Assumptions and Dependencies 4

3 System Features 4
- 3.1 Connect wireless available 4
- 3.2 Display the current time 4
- 3.3 Display current IP address 4
- 3.4 Photograph through camera 4
- 3.5 Send image to Server 4
- 3.6 Delete image taken 4
- 3.7 A terminal for receiving data from FEZ Spider 5
- 3.8 Display image on PC 5
- 3.9 Save images in database 5
- 3.10 Send images to a specified email address 5

4 Database Design 5
- 4.1 Tables and Fields 6
  - 4.1.1 Databases 6
  - 4.1.2 Tables 6
- 4.2 Data Migration 6

5 Objects and Characteristics 6
- 5.1 Class Diagrams 7
- 5.2 Use Case Diagram 7
- 5.3 Use Cases 8
  - 5.3.1 Take picture 8
  - 5.3.2 Send picture 8
  - 5.3.3 Visualize data received 9
  - 5.3.4 Query data 10

5.3.5 Email setting 11
5.3.6 Send email with attachment 11
5.3.7 Display picture 12
5.4 Key Function and Methods Highlight 13
5.4.1 Take picture 13
5.4.2 Send picture to server 13
5.4.3 Data receiving status visualization 14
5.4.4 Load Image into database 15
5.4.5 Query from database 15
5.4.6 Send email 16
5.4.7 Email Attachment Setting 16
5.4.8 Display picture 16
6 User Interface Design 17
6.1 FEZ Spider 18
6.2 Server 18
6.2.1 Structure Diagram 19
7 Nonfunctional Requirements 20
7.1 Performance Requirements 20
7.2 Safety Requirements 20
8 Summary 20

1 Introduction

1.1 Purpose

The purpose of this document is define the system specifications by using the FEZ Spider Mainboard with 72 MHz 32-bit ARM7 as Microcontroller to be placed in the final end system, concatenating various components. Currently, the only software involved in this project is the .NET Micro Framework and .NET Gadgeteer programming. The low-level components are designed, coded and tested.

1.2 Document Conventions

For this SDS embedded software design standards shall be used. As of yet there are no special sections which needs to be specifically pointed out. Regardless of concerning the requirements in details, here, requirements shall exist in a triage similar to the following: mandatory, nice-to-have, optional.

1.3 Intended Audience and Reading Suggestions

This document is intended for any user wishing to understand the underlying concepts in the software developed for Communication System. Readers will most likely include electrical and computer engineering professors and students as well as any interested embedded software developers. The rest of this SDS shall contain functional requirements for the Communication System and their relationship to its Microcontroller and components. It is suggested that the reader first visit the .NET Micro framework of Gadgeteer to get an overview of architecture of this product and methods definitions.

1.4 Project Scope

The embedded application being specified in this SDS shall be developed in the C# programming language using Visual Studio 2012 integrated .NET Gadgeteer kit. The purpose of the application to fulfill all functional requirements as described in the following.

This application implements a communication procedure among local PC and FEZ Spider kit. The application is able to connect wireless when it scans an available wireless network connection by filling appropriate authorization certification information automatically. While the application boots, it will pop-up an interface on the screen with options. By clicking the button from end user, the application will capture an image from camera and displays on the screen. Then, by activating the dispatcher, this image will be sent to PC, meanwhile, on the PC, it will wait for receiving data from FEZ Spider. After receiving data, on the terminal of PC, the data will be saved in the database first, certainly, it also could be displayed by selecting on the list manually. After selecting image on the data list, and setting the email configuration, user is allowed to send email to a specified email address with more than one attachment.

1.5 References

This document shall not be published on the internet, but you could still find couples of description about the relevant topics. Such as http://mikedodaro.net/ , at this website, you will be able to view the various projects based on Gadgeteer kit. And also you could find instructions for describing each devices separately, including tons of functionalities methods on the website http://gadgeteer.codeplex.com.

2 Overall Description

2.1 Product Perspective

The overall product being specified in this document is the Communication System. More importantly is the fact that this system is based on an embedded devices. All the functions which concerned will be described in details in this document. The following block diagram shall illustrate the whole architecture of our system.

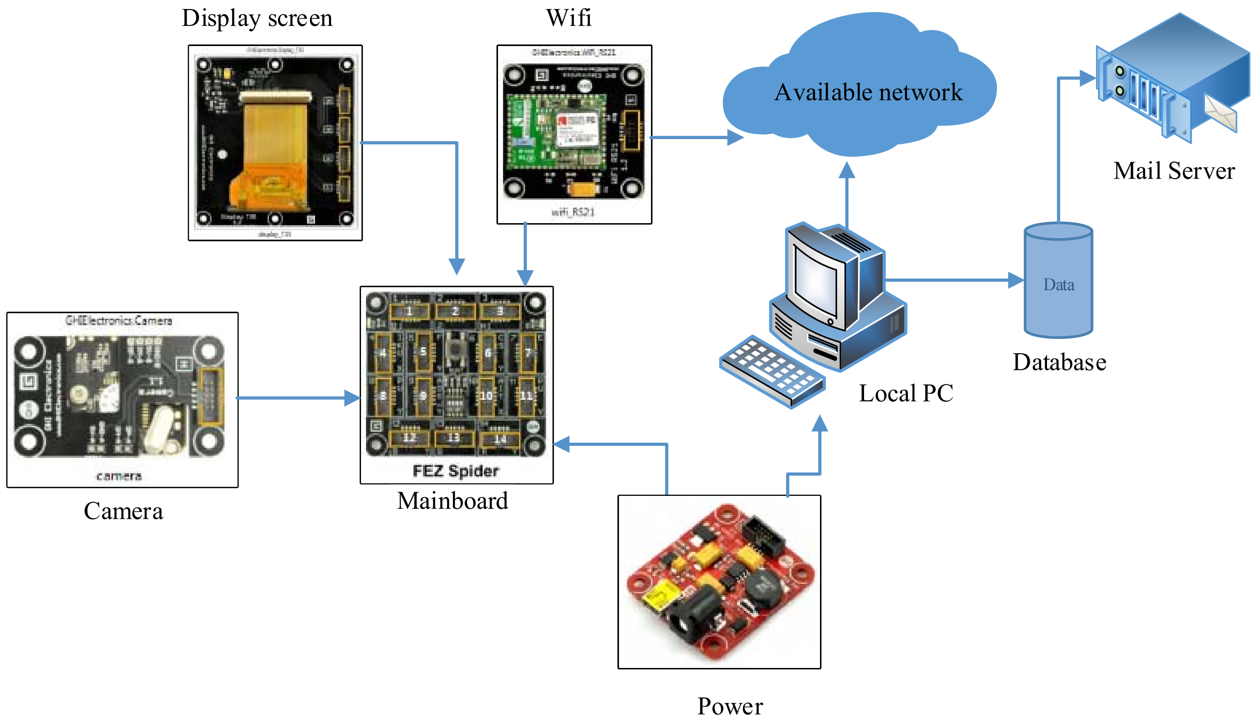


Figure 1 System architecture

2.2 Product Features

The following is a brief description of the design objectives of the project: Communication System:

- Connect available wireless connection network
- Display the current/correct time in 12 or 24 hour format
- Display the current IP address
- Photograph through camera and display
- Send one image to PC (Server)
- Delete image taken
- Terminal on PC receive images dynamically
- Save received images in Database

- Display received image on PC interface
- Send selected images to a specified email address

The underlying software in this system must guarantee the wireless connection, and ensuring that local PC and FEZ Spider could connect the same network. On the terminal of PC, the standalone interface which will act as a receiver waiting for data from FEZ Spider automatically. The visualization function of terminal on PC, it requires to be refresh the list of received data manually. Meanwhile, the internal software must guarantee the connection with database when the database is available. User could manipulate the available data for displaying individual. And also the dispatcher function for sending data to email server will be activated automatically which will be defined internal.

2.3 User Classes and Characteristics

The intended users are those who know C# program syntax. It can also be used by those who would like to learn C# programming skills through a practical application. This should not be used by children due to certain dangers such small parts and electrical components.

This object is a communication system meant to communication among different devices and terminals, here, in our case, it refers to FEZ Spider and PC, which through the available wireless connection. It is also extendable by adding alternate functions and methods to diverse intentions.

With respect to software, no user shall have access once the main program is “burned” into the FEZ and also the server side interface. On the hardware side, there will be no user class built, instead, on the server side, we defined a class to make a graphical interface, and distinct function requirement.

2.4 Operating Environment

The operating environment of the FEZ Spider is unlimited, with good usability and friendly interface. The environment should be able to supply the appropriate power needed through a USB cable connection with PC. It should not be used in areas that contain a lot of moisture such as the bathroom or pool room. It should not be in an area that it could potentially be dropped in a sink such as a kitchen or laundry room, due to the safety reason of electronic units.

2.5 Design and Implementation Constraints

The software required for the FEZ Spider communication system falls into the category of embedded systems. As such there are certain design and implementation constraints posed to the software developers. The first constraint is based upon the hardware chosen. A FEZ Spider is capable of executing C# program. The second constraint is the development environment. Any C# compiler and associated IDE may be used but a specific programmer is necessary to pack the functions inside the executable file.

High quality, error free and well documented code is desired. This document as well as final documentation shall assist groups in the future who wish to use the FEZ Spider communication system project as a reference.

2.6 Assumptions and Dependencies

It is assumed that the software shall be written in the C# programming language using Visual Studio. It is also assumed that the software is being written for a FEZ Spider embedded device and shall fulfill all functional requirements in the scope part.

3 System Features

3.1 Connect wireless available

According to the wireless configuration, FEZ Spider wireless device does not support the polito wireless authentication policy, so we choose wireless network from mobile phone which has the fixed SSID (FEZSpider) and password (projects). In our case, the connection characters will differ between distinct connection fields, however, we fixed the server IP address (192.168.43.27) and port (1100), and assume the WIFI connection is stable, static and never changed, so it is not necessary to compile the source again.

3.2 Display the current time

Assuming the successful WIFI connection, we get the internet time from "time-a.nist.gov" website, and display in HH:mm format. Certainly, the time is also relative to the FEZ Spider hardware clock cycle, in our device, it will always one hour later than the international time, obviously, it proposed a possible improvement in the future.

3.3 Display current IP address

Assuming the successful WIFI connection, we get the current IP address which has been assigned, meanwhile, displaying on the main interface.

3.4 Photograph through camera

After the main windows raises up, the function of camera runs by clicking the button on the touch screen. While completing the photograph, the image will be stored in a public static variable which is used to send to server side later.

3.5 Send image to Server

Assuming the camera works in a proper way, and also we get the value of the parameter which is used to save the image data. After user click the button of "Send" on the main screen, it will call the function that sends the image data by TCP socket. In this case, we assume the FEZ Spider and server side connect the same network, and obviously, the IP address and connection port is known by FEZ Spider before, which will be used to create socket serving the communication between server and client.

3.6 Delete image taken

In this section, it does not indicate a particular feature of our system, it only presents the function which is essentially required by communication process. We use this function to clean the public static parameter which is used to store the image data after raises up the camera. Logically, it is necessary to clean the 'container' after performing the data transmission each time, maintaining the efficiency and availability.

3.7 A terminal for receiving data from FEZ Spider

On the server side, it will get current IP address by connecting the available wireless connection, meanwhile, it will open a port for providing the connection of client. Then, it employs multiple threads to watch the port and transact the event. By convenience, on the view of listbox, it will pop-up necessary message for different intentions to provide the friendly and usability.

3.8 Display image on PC

Here, we assume there are sorts of images which have already been saved in database in binary format, serialized by an integer counter which is treat as an index. By clicking the header of rows which list in the datagridview component, it will activate the data retrieving from database, then, display image by employing picturebox. The image will be distinct while clicking different header each time.

3.9 Save images in database

It mainly refers to image data which will be received from client through the listener on server side. The data transmission will be performed after user activate the function of sending data. In this case, we transit the image data in package by using TCP socket in the format of byte array.

3.10 Send images to a specified email address

On the user interface, we provide several options of SMTP server, meanwhile, setting the SMTP port and SSL. It requires to input sender and receiver email address, and also password in mask. All the fields on the interface must be filled. Then, by clicking the send button, the progress bar will show the status of transaction.

4 Database Design

This section describes the technics which are used for storing images in database (SQL Server). It has been performed by using BLOB data type in this communication system. Essentially, it depicts the key points about how to store sorts of images, and meanwhile, maintaining considerable Server performance.

Generally, the following technics are available for storing binary information:

- Saving files into a fixed directory, meanwhile saving directory path and name in database
- Using blob or varbinary datatype in database to store files
- Using OLE storage structure to store files locally

Here, we select second strategy to satisfy the data storage requirement. After saving files in database, apparently, the local temporary files are useless and could be deleted immediately, and it also does not require complex management policy for managing binary files, besides, all the files will be stored on the network server, it is highly convenience for data sharing.

In this case, due to the reason of displaying image by activating the function on the interface, it requests a good Server and interactive performance.

4.1 Tables and Fields

In this project, we only use one table which is named ImageData and used for storing image which received from client.

4.1.1 Databases

Here, we give the connection string of SQL Server database.

```
private string ConnStr = "Server=BERLIN\\SQLEXPRESS;Database=CSysDB;User
ID=sa;Password=*****";
```

4.1.2 Tables

Index	Field Name	Field Type	Remark
1	SeqID	Integer	not allowed null
2	ImgFEZ	Image	not allowed null
3	ReceivedDate	Date	not allowed null

Table 1 Database table

4.2 Data Migration

Here, we indicates the procedure of how to migrate database in different destinations. First of all, it requires a backup file which has generated before, or received from source server. Then, creating a database case which is named 'CSysDB' , right click this database, selecting task, restore, and database, on the pop-up windows, choose device option, then click the button on the right side. At this moment, clicking add button, selecting the backup file in the directory tree, then clicking ok. On the window of restore database, selecting destination database which you have built before, then complete. Remembering that, for running the database, you need to create the user account which named sa, with password spider. For the details techniques, you can find information on the MSDN website, we do not expand here.

5 Objects and Characteristics

This section describes all required classes, attributes included, and also each methods in details. Objects are entities in a software system which represent instances of real-world and system entities. Objects derive from things (e.g., email, image, etc.), roles (e.g., classes of actors in systems like client who will operate in our system, etc.), and events (e.g., visualize data, query data from database, send email with attachment, etc.) and interactions (e.g., close our interface on PC, etc.).

Objects are created according to some class definition. A class definition serves as a template for objects and includes declarations of all the attributes and operations which should be associated with an object of that class. The operations associated with the object provide services to other objects, which request these services when some functionality is required.

5.1 Class Diagrams

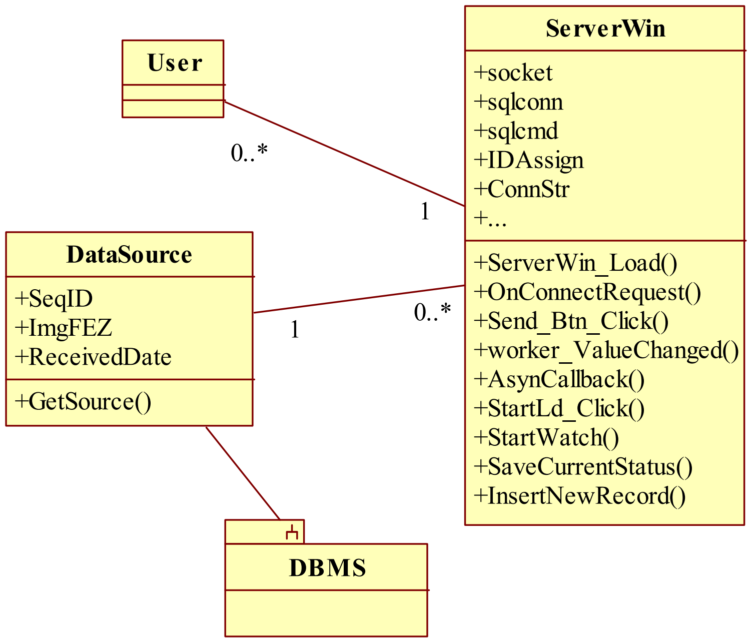


Figure 1 Class diagram

5.2 Use Case Diagram

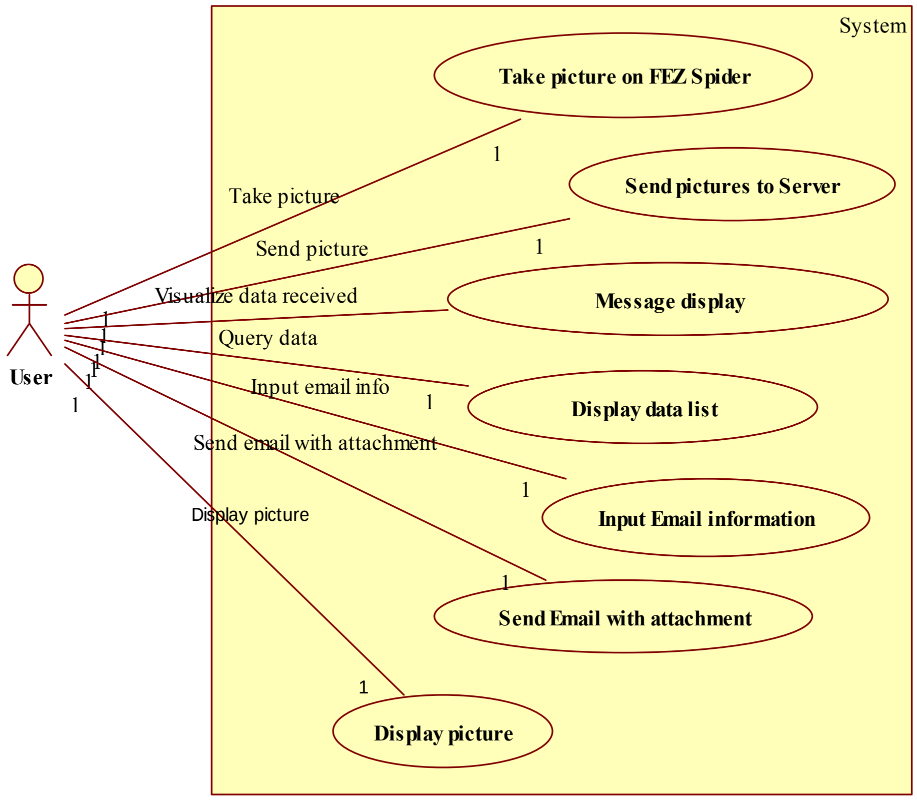


Figure 2 Use case diagram

5.3 Use Cases

5.3.1 Take picture

Use case name: Take picture		**ID:** UC1	**Priority:** Medium
Primary actor: User	**Sources:** Jiawei, Wu	**Use case type:** Business	**Level:** Overview
Brief description: This use case describes how to take a picture singly by camera on hardware (FEZ Spider) dynamically, which is an essential function of the system. In this use case, the actor's goal is to see the camera raises up and taking picture successfully by clicking the touch screen. It will generate a public static state which is used to store the image taken by camera.			
Goal: • The successful displaying of camera raises up and picture taken			
Success Measurement: • The dimension of that public static variable is not empty			
Prediction: • On the side of FEZ Spider, the main windows runs correctly • There must be a connection with camera through cable correctly **Trigger:** • User clicking the camera button on touch screen			
Typical flow of events: 1. User click the camera button 2. The camera opens and ready for taking picture 3. User click the screen again to save the image data 4. Camera closes and return to main windows			
Assumptions: 1. It is assumed that the hardware of camera works well, there should be no problem inside 2. It is assumed that the touch screen is sufficient about sensitive to sense the clicking 3. It is assumed that there is no extra interactive while one user click the touch screen			
Implementation Constraints and Specifications:			

5.3.2 Send picture

Use case name: Send picture		**ID:** UC2	**Priority:** Medium
Primary actor: User	**Sources:** Jiawei, Wu	**Use case type:** Business	**Level:** Overview
Brief description: This use case describes how to send one picture individually on the hardware (FEZ Spider) side, which is an essential function of the system. In this use case, the actor's goal is to sense the function for data sending on the FEZ Spider touch screen, simultaneously, sending the data to server side. There will be no extra data generated.			
Goal: • The successful of the while data package sending			

Success Measurement: • Server receives all the image package correctly
Prediction: • Client (FEZ Spider) and server connect the same network which will assign the proper IP address automatically to client and server • The IP address and open port on server side have already known by client • The socket connection works well on two sides (client and server) • All the image package will be sent successfully in the format: package dimension and the whole package • The server is always running and waiting for client connection **Trigger:** • User click the button "Send" on touch screen on FEZ Spider side
Typical flow of events: 1. User click the "Send" button 2. Camera opens and ready to take a picture 3. By clicking the screen again, capture on picture 4. Camera closes and return to main windows
Assumptions: 1. It is assumed that the hardware of camera works well, there should be no problem inside 2. It is assumed that the touch screen is sufficient about sensitive to sense the clicking 3. It is assumed that there is no extra interactive while one user click the touch screen
Implementation Constraints and Specifications:

5.3.3 Visualize data received

Use case name: Visualize Data Received		**ID:** UC3	**Priority:** Medium
Primary actor: User	**Sources:** Jiawei, Wu	**Use case type:** Business	**Level:** Overview
Brief description: This use case describes the visualization of the data receiving status from hardware (FEZ Spider) dynamically, which is an essential function of the system. In this use case, the actor's goal is to see the data receiving status. There is no extra data which will be generates.			
Goal: • The successful visualization of data receiving status			
Success Measurement: • User could see the data receiving status through interface on PC			
Prediction: • FEZ Spider and PC connect in the same network (regardless the availability of internet connection) • FEZ Spider and PC could find each other by ping command using IP address • FEZ Spider takes pictures successfully • FEZ Spider connect remote server on PC using socket method successfully with a particular server IP address and port			

• FEZ Spider pack all the images which will be sent successfully (even if it sends only one picture to server) • Server must be booted before taking pictures on FEZ Spider **Trigger:** • User open interface on PC successfully
Typical flow of events: 1. User open system interface on PC firstly 2. Connect device with PC through USB cable and boot 3. Take sorts of pictures using camera on device 4. Select pictures which will be sent later on device 5. Pack all the pictures which will be sent later 6. Send package to server by interacting with touch screen 7. List connection information and insert successful message on the panel inside interface
Assumptions: 1. It is assumed that the connection among device and PC works in a proper way, that support the data transmission between device and server 2. It is assumed that the RAM is efficient to pack the required data on device, which will be sent to server
Implementation Constraints and Specifications:

5.3.4 Query data

Use case name: Query data		**ID:** UC4	**Priority:** Medium
Primary actor: User	**Sources:** Jiawei, Wu	**Use case type:** Business	**Level:** Overview
Brief description: This use case describes the query data operation which will be performed on the interface which is an essential function of system. In this case, the actor's goal is to display all the available data which has already been stored in database successfully before.			
Goal: • The successful visualization of data list which locates in database			
Success Measurement: • User could see the data list through interface on PC			
Prediction: • Database server is running in a proper way with efficient performance • Interface could connect database successfully • Inside database, the essential tables and fields have already been built before • The query operation could be performed in database • The data is available for displaying, or empty tables with fixed fields **Trigger:** • User startup the main user interface			
Typical flow of events:			

1. User open system interface on PC firstly			
Assumptions: 1. It is assumed that database is running on the same location with interface			
Implementation Constraints and Specifications:			

5.3.5 Email setting

Use case name: Email setting		**ID:** UC5	**Priority:** High
Primary actor: User	**Sources:** Jiawei, Wu	**Use case type:** Business	**Level:** Overview
Brief description: This use case describes the email setting operation which will be performed on the interface which is an essential function of system. In this case, the actor's goal is to set all the essential data for email.			
Goal: • Filling all the fields for email on the panel			
Success Measurement: • Satisfying all the necessary attributes and fields for email class in C#			
Prediction: • Email sending function has already been defined before inside system with the fields on panel • User input the correct fields and attributes values **Trigger:** • User open interface and operates			
Typical flow of events: 1. User open system interface on PC firstly 2. User input sender email address in the field of 'From', SMTP server info, display name, email server port of sender, password, receiver email address, subject of email, and the body content of email			
Assumptions: 1. It is assumed that user has already known his own email address, SMTP server configuration variables and values, password, and the receiver's email address exactly 2. It is assumed that the email address of receiver exists and guarantee the availability			
Implementation Constraints and Specifications:			

5.3.6 Send email with attachment

Use case name: Send email with attachment		**ID:** UC6	**Priority:** High
Primary actor: User	**Sources:** Jiawei, Wu	**Use case type:** Business	**Level:** Overview
Brief description: This use case describes the sending email with attachment which will be performed on the interface which is an essential function of system. In this case, the actor's goal is to send an			

email with the selected data from database which will be visibility on the interface successfully.
Goal: • Send an email with attachment
Success Measurement: • Return a successfully sending message
Prediction: • User has already filled the necessary data fields on email setting panel • User has selected the required data from database panel on the interface which using a data grid view for visualization proper info • User perform attachment function firstly **Trigger:** • User click attachment, then send
Typical flow of events: 1. Select object on 'database' panel 2. Click 'Send' button
Assumptions: 1. It is assumed that data is existence and selectable 2. It is assumed that internet is available due to the reason for supporting SMTP service
Implementation Constraints and Specifications:

5.3.7 Display picture

Use case name: Display picture		**ID:** UC8	**Priority:** High
Primary actor: User	**Sources:** Jiawei, Wu	**Use case type:** Business	**Level:** Overview
Brief description: This use case describes the function about display picture which locates in database and available and it is an essential function of our system. In this case, the actor's goal is to display one picture in database which is taken from camera on device			
Goal: • Display picture			
Success Measurement: • Displaying one picture successfully			
Prediction: • There is at least one picture inside database • The system function for calling 'windows photo viewer' is available • The data ends with the extension of '.jpg' or '.bmp' **Trigger:** • Click display picture button			
Typical flow of events: 1. Click the row header in the data grid view component in the database panel			

Assumptions:

1. It is assumed that there is at least one picture which has been stored in database
2. It is assumed that the function of windows photo viewer is available
3. It is assumed that user will only open one photo at each time
4. It is assumed that the resolution of local system supports the visualization of picture

Implementation Constraints and Specifications:

5.4 Key Function and Methods Highlight

5.4.1 Take picture

Parent Class Name: MainWindow

Brief Description:
This is a method to achieve the function of camera, which we just call the GHI Gadgeteer internal API, but we create an event to trigger this action while user click the camera button

Program Description Language

```
CameraManager.CameraModule.BitmapStreamed += new
GT.Modules.GHIElectronics.Camera.BitmapStreamedEventHandler(CameraModule_Bi
tmapStreamed);
```

5.4.2 Send picture to server

Parent Class Name: MainWindow

Brief Description:
This is a method to achieve the function of data sending, which is the most important and challenge part. Inside the function definition, we implemented the asynchronous technique to solve the problem about asynchronous socket connection, obviously, at the same time, we must maintain the usability of server interface. In order to solve the problem of package sending, first of all, we get the exact dimension of our image data, then, using 4 bits to store it, while connecting socket, we send the size data first, then, using a while loop to receive all the image data package.

Program Description Language

1.Function

```
while (!ConnectToSocket()) { }
SendPicture(cameraImage);
```

2.Socket connection

```
//set parameter for the connection
ipep = new IPEndPoint(IPAddress.Parse(ip), port);
clientSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);

try
{
   //connect to socket
   clientSocket.Connect(ipep);
   Debug.Print("Socket connected to " +
clientSocket.RemoteEndPoint.ToString());
}
catch (SocketException se)
{
   Debug.Print("SocketException : " + se.ToString());
   return false;
}
   return true;
```

3.Data sending

```
// Encode the picture into a byte array.
Bitmap bmp = picture.Bitmap;
Byte[] outputBytes = new Byte[bmp.Width * bmp.Height * 3 + 54];
Util.BitmapToBMPFile(bmp.GetBitmap(), /*bmp.Width*/320, /*bmp.Height*/240,
outputBytes);

// Send the data through the socket. The image is in the byte array
'outputBytes'
int bytesSent = SendBuffer(outputBytes);
```

4.Data Convert

```
//send the size of the byte array of the image on 4 bytes...an integer
byte[] tmp = new byte[4];
for (int i = 0; i < 4; i++)
{
  tmp[i] = (byte)(buffer.Length >> (i * 8) & 0xFF);
  Debug.Print("byte " + i + ": " + tmp[i].ToString());
}
clientSocket.Send(tmp);

//send the bitmap picture in the form of a byte array
int numBytes = clientSocket.Send(buffer);
return numBytes;
```

5.4.3 Data receiving status visualization

Parent Class Name: ServerWin

Brief Description:
This is a method to achieve the function of message displaying, which we implementing the multiple threads strategies to listen open port in server side, meanwhile, deal with the across access of components on UI by using delegate method

Program Description Language

1.Listener:

```
ListenSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
ListenSocket.SetSocketOption(SocketOptionLevel.Socket,
SocketOptionName.ReuseAddress, true);
localEP = new IPEndPoint(myIP, ServerPrt);
ListenSocket.Bind(localEP);
this.listBox1.Items.Add("Server is running!");
this.listBox1.Items.Add("Server IP: " + myIP.ToString() + ":" +
ServerPrt.ToString());
ListenSocket.Listen(10);

myThread = new Thread(StartWatch);
myThread.Start();
this.listBox1.Items.Add("Wating for connection!");
```

2.Socket Acceptance:

```
sokMsg = ListenSocket.Accept();

// Display message
CrossThreadOperationControl CrossAddMsg = delegate()
{
this.listBox1.Items.Add("A Client connected!");
this.listBox1.Items.Add("ClientIP: " + sokMsg.RemoteEndPoint.ToString());
};
listBox1.Invoke(CrossAddMsg);
```

5.4.4 Load Image into database

Parent Class Name: ServerWin

Brief Description:

Load single image file which ends with .jpg extension into SQL server database

Program Description Language

```
byte[] blob = GetPhoto("c:\\test1.jpg");

sqlConn = new SqlConnection(strConn);
sqlCmd = new SqlCommand("INSERT INTO
ImageData(SeqID,ImgName,ImgFEZ,ReceivedDate)"+"Values(@SeqID,@ImgName,@ImgF
EZ,@ReceivedDate)",sqlConn);

// Parameters list
sqlCmd.Parameters.Add("@SeqID",SqlDbType.Int).Value = 1;
sqlCmd.Parameters.Add("@ImgName",SqlDbType.NChar,10).Value="test1";
sqlCmd.Parameters.Add("@ImgFEZ",SqlDbType.Image).Value = blob;
sqlCmd.Parameters.Add("@ReceivedDate",SqlDbType.Date).Value = RecvDate;

sqlConn.Open();
sqlCmd.ExecuteNonQuery();
```

5.4.5 Query from database

Parent Class Name:

Brief Description:

This is a method used to achieve the function of visualizing data which retrieved from SQL server data table

Program Description Language

```
// Initial datagridview component
List<DataSource.DataBaseFields> list = DataSource.GetSource();
_source.DataSource = list;
dataGridView1.AutoGenerateColumns = false;
dataGridView1.RowCount = list.Count;

for (int i = 0; i < list.Count; i++)
{
dataGridView1.Rows[i].Cells["SeqID"].Value = list[i].SeqID;
dataGridView1.Rows[i].Cells["ImgFEZ"].Value = list[i].ImgFEZ;
dataGridView1.Rows[i].Cells["ReceivedDate"].Value = list[i].ReceivedDate;
dataGridView1.Rows[i].Cells["IsAttached"].Value =
MyLib.SelectedStatus.Selected;
}
```

5.4.6 Send email

Parent Class Name: ServerWin

Brief Description:
This is a method used to achieve email sending with necessary attributes based on the smtpclient class

Program Description Language

```
// Object under SmtpClient
// Used to set email subject and content
MailMessage mymail = new MailMessage();

// Sender email address
mymail = new MailMessage(SenderAddr, ReceiverAddr);
mymail.Subject = Subject;
mymail.IsBodyHtml = true;
mymail.Body = Body;

// Create sender object
// Verify email server address,port,username and pwd
SmtpClient client = new SmtpClient(SMTPServer,
Convert.ToInt32(ServerPort));
client.UseDefaultCredentials = false;
client.EnableSsl = true;
client.Credentials = new System.Net.NetworkCredential(SenderAddr, Pwd);
client.Send(mymail);        }
```

5.4.7 Email Attachment Setting

Parent Class Name: ServerWin

Brief Description:
This is a method used to achieve the function of attaching an file to an email

Program Description Language

```
byte[] imgbyte = (byte[])imgObj;
MemoryStream ms = new MemoryStream(imgbyte);

Attachment myAttach = new
Attachment(ms,System.Net.Mime.MediaTypeNames.Image.Jpeg);
myAttach.ContentDisposition.FileName = j.ToString() + ".jpg";

mymail.Attachments.Add(myAttach);
ms.Flush();
ms.Seek(0, SeekOrigin.Begin);
```

5.4.8 Display picture

Parent Class Name: ServerWin

Brief Description:
This is a method used to achieve the function of displaying an image which got from database into picturebox component

Program Description Language

```
byte[] imgbyte = (byte[])imgObj;
MemoryStream ms = new MemoryStream(imgbyte);
Image img = Image.FromStream(ms);
this.pictureBox1.Image = img;
ms.Flush();
```

6 User Interface Design

This section describes all of the requirements that have been defined, and specified in the following table. Here, we separate from hardware (FEZ Spider) and software (Server-side) part, using different design technics with distinct topologies, mainly based on the task which referring to requirements.

First of all, for the design of hardware side, user operates through the touch screen, the layout of the operating screens will focus on ease of use, comprehension, and safety. The design will avoid elements that cover or hide information, such as pop-up messages boxes, resize menus, radio and check boxes, sliders, spin controls. Simple controls like click buttons, selectable lists will be favored over edit controls that require typing in numbers or characters.

Then, to the design of software side, user operates through the graphical UI on local PC, the layout of the operating interface will focus on the requisite tasks. The design will cover buttons, pop-up message dialogs, sub-windows and sorts of serviceable components. User could perform kinds of manipulation, such as, button click, list box select.

The use of keyboard and mouse will simplify interaction with interface. An operator error is defined as an unexpected or inappropriate data entry, e.g. button press, syntax error. This is superseded by any other defined error response.

Index	Requirement description	Remark
1	Have different buttons to trigger the function of picture taking, data sending, and clean container	–
2	Display the internet time	–
3	Display the current IP address	–

Table 2 Interface Requirements of Hardware

Index	Requirement description	Remark
1	Visualize the data received from sender dynamically, displays in the type of list box, including the received date, ID which is used for guaranteeing the uniquely	After receiving the data, it will be inserted into database immediately
2	Visualize the connection status with database, ensure remaining the essential connection	–
3	Database connection button	–
4	Visualize the list of data which has been already stored in database	–
5	Selectable of data inside database with check box	–
6	Set or select sender email address	–
7	Set or select receiver email address	–
8	Write message in text box which will be sent through email at the same moment	–
9	Button for send email	–
10	Message notification about email sent successfully	–

Table 3 Interface requirements of Server Side

6.1 FEZ Spider

On the side of hardware, in order to provide the usability and friendly on interface, we design three different windows. The first windows will raise up while the system try to find the network and synchronize the time and acquire the current IP address, then, while the former works correctly, the main window will raise up and show the proper information, such as the button of camera, send and delete for diverse operation.

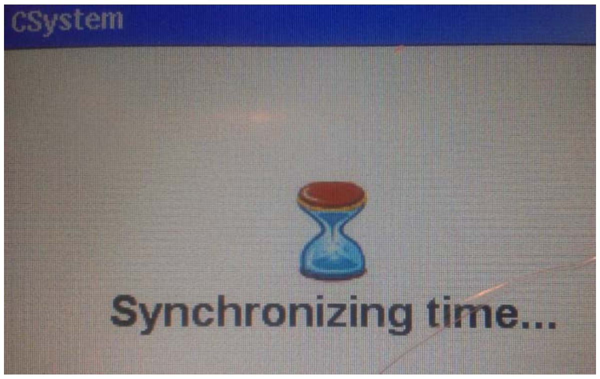


Figure 3 Initial window

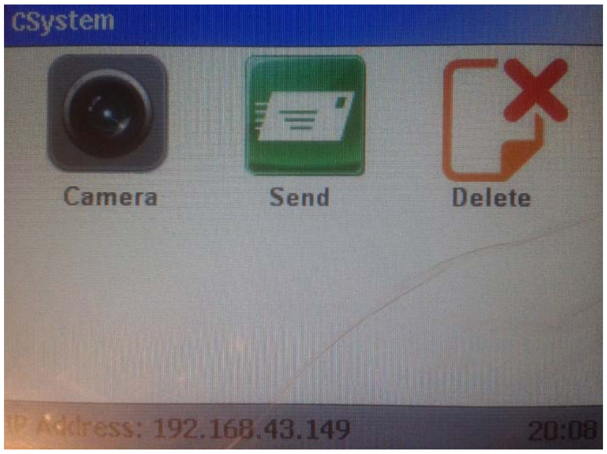
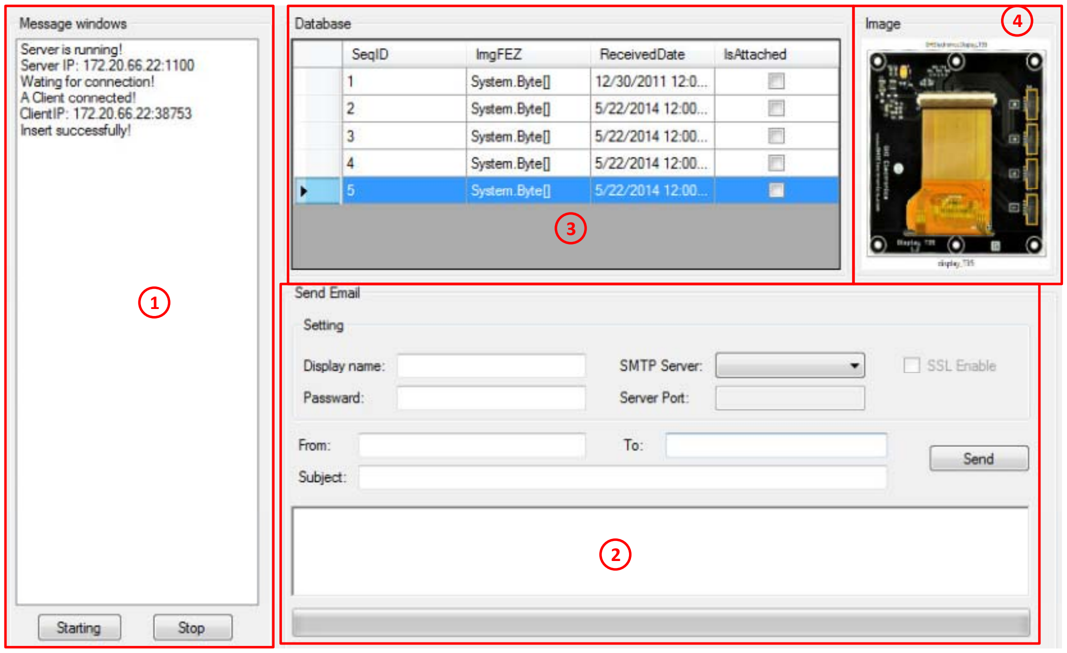


Figure 4 Main window

6.2 Server

There will be only one mainly interface during the execution and interactive with the program. All the activities will be performed in this panel which will be designed using windows form graphical design tool in Visual studio. Below are the details for describing the fields on the panel in the view of functionalities.



Server is running!
Server IP: 172.20.66.22:1100
Wating for connection!
A Client connected!
ClientIP: 172.20.66.22:38753
Insert successfully!

	SeqID	ImgFEZ	ReceivedDate	IsAttached
	1	System.Byte[]	12/30/2011 12:0...	☐
	2	System.Byte[]	5/22/2014 12:00...	☐
	3	System.Byte[]	5/22/2014 12:00...	☐
	4	System.Byte[]	5/22/2014 12:00...	☐
▶	5	System.Byte[]	5/22/2014 12:00...	☐

Figure 5 Server-side Interface

Index	Region Name	Function Description
1	Message windows	1.Pop-up the status of connection clients, including remote client IP address and port 2.Pop-up message after one client connected 3.Pop-up message after data receiving and inserting successfully 4.Pop-up message after email sending successfully
2	Send Email	1.Set Email information, including display name, SMTP server, password, server port, sender address, receiver address, subject, and email body 2.Progressbar is used to display the status of email sending
3	Database visualize	1.List existence data in database 2.Support record selection option 3.Fresh list view after insert new record
4	Image Display	Used to display image after user click the header of left-side list view component, refresh automatically, non-interactive

Table 4 UI Function list

6.2.1 Structure Diagram

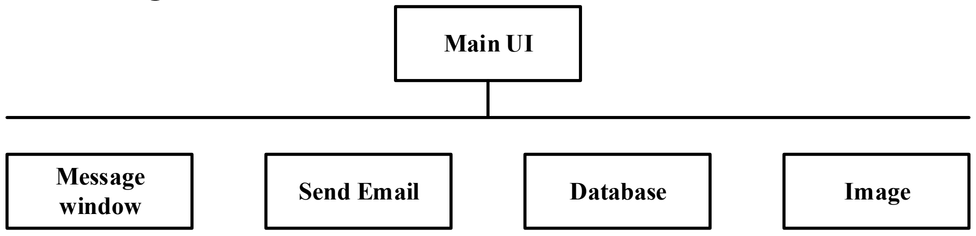


Figure 6 Structure Diagram

7 Nonfunctional Requirements

7.1 Performance Requirements

A minimalistic approach shall also be used to preserve limited amount of memory. The size of the program written must be not exceed memory size and must execute reasonably quickly. There shall be no software maintenance or evolution for this project.

7.2 Safety Requirements

N/A

8 Summary

In order to build a system, the first step is meaningful requirement analysis, and make the scope of the project. In our case, we would like to achieve a more ‘simple’ goal which is achievable and less complex. Certainly, the first step is to know ‘how to climb’, then, we will know ‘how to run’. Thanks to the basic course about C# language and presentation of hardware, we have already acquired sorts of techniques, such as programming syntax, how to build the connection channel between client and server through TCP socket, how to startup the first project of FEZ Spider, how to debug and track the error when the application is running, and so on.

The difficulty in our project is that we must make clear about how to create the connection channel between client and server, especially we implement the asynchronous strategy, because of the reason about suspension of interface while server is always waiting for the new client connection. And also, we must make sure, how much package we need to transmit, how to transit. Because data package will transmit in distinct ways automatically, we cannot control the data transmission process, in this situation, we set the package format manually by setting the first bytes and the following bytes content. In this way, we are sure that we could receive the correct dimension of package, avoiding the data lost.

At the end, we would like to thanks again to Professor Guido, he gave us many ideas about the structure building and couples of programming suggestion. It help us save a lot of time while we are struggling about programming syntax. Certainly, any kinds of improvement of our code is appreciated and welcomed.