Runsen Ma

Remote Controller for TV from Windows Phone

WP TV Controller

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Electronics

Bachelor's Thesis

21-08-2017

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Runsen Ma Remote controller for TV from Windows Phone (WP) 32 pages + 2 appendices 5 May 2017
Degree	Bachelor of Engineering
Degree Programme	Electronics
Specialisation option	
Instructor(s)	Kai Lindgren, Senior Lecturer

The objective of this study was to design a windows phone remote controller for a TV with microcontrollers,a touch screen keyboard for the WP that sends commands to the microcontrollers.

In this study, codes for Windows Phone 8.1 Bluetooth controlling NETMF Gadgeteer were made, to run in the visual studio environment. Modificantions were made to create the buttons for the TV controller. Signals were sent to spiders through bluetooth to control the source materials from GHI Electronics. The microcontrollers are programmed to a .NET Micro Frame work (MFW) with C#.

The goal of was basically achieved despite some errors running in visual studio. Commands can be sent to controll LEDs successfully although there were some irregular colors on the display of LEDs.

Keywords	Spider microcontrollers , windows phone,Visual Studio,.NET Micro Framework C#

Contents

Abbreviations.

WP	Windows Phone
CMD	CMD
BT	Bluetooth
LED	Light-Emitting Diode
API	Application programming interface
SPP	Serial Port Profile
USBClientDP	The USB Client Dual Power
SDK	Software development kit
EMX System on Module	Eberhard Mattes eXtender System on Module

1 Introduction

The goal of this project was to remote control a TV by a WP with microcontrollers. In other words, to turn the WP a Remote-controller for TV, i.e. a TV remote-controller alternative. Similar to a TV remote controller, it transfers all of the functions that a TV remote controller has to a single WP. The WP screen simulates the TV Remote- controller's keyboard and buttons in a connection between internet /WLAN and microcontrollers which flash LEDS to react the commands from WP.

Modifications added to the original program were aimed to make the WP almighty.This is nowadays the trend which does all of the unnecessary parts away and simply merges their functions into one.

This device was operated in visual studio. The microcontrollers are programmed to a .NET Micro Frame work (MFW) with C#. The inspiration and source information of the study is called “Windows Phone 8.1 Bluetooth controlling NETMF Gadgeteer” from GHI electronics [1].

The inventor of the whole system is RoSchmi. My thesis supervisor and I had a lot of contact with him during this study and we got updates and some practical help from him. The record of out contacts can be found at the forum at [2].

2 Programming

The goal of the study was to control the TV resembled FEZ Spiders LEDs. They are manipulated by the WP/WP Simulator- BT SPP -WinPhoneClient. The dominating parts were mainly modification of source code related tasks which are divided into following four sections

BlueWinPhoneClient
BT_CerbuinoXBeeCtrlByWinPhone
BT_SpiderCtrlByWinPhone
BT_Spider_Scratch

The above codes are from Roschmi in GHI Electronics, the inventor of the gadget. [3]. Finally, the manipulation of the remote controller was basically achieved but there are some errors detected in the running Visual Studio. It was at the occurring of errors when the inventor of the gadget was contacted intensively.

3 Prerequisites

3.1 Software and Installations for the Operations of the WP

There are a few key software for the microcontrollers and WP. They are:

Visual Studio 2013 Community
Microsoft .NET Micro Framework 4.3 (QFE2)
Microsoft .NET Gadgeteer Core
GHI Electronics NETMF SDK 2016 R1

Above mentioned software are installed by following the instructions given by GHI electronics [4].

The windows SDK and emulator are downloaded and installed [5].
The whole system is operated under Windows 8.1.

4 System Overview

In general, the system contains Windows Phone that is capable of Bluetooth connectivity which allows Windows Phone to communicate with FEZ Spider. The command that FEZ Spider receives from the Windows Phone are executed by FEZ Spider and related actions are performed to change the colors of corresponding LEDs. Figure 1 shows the overview of the system.

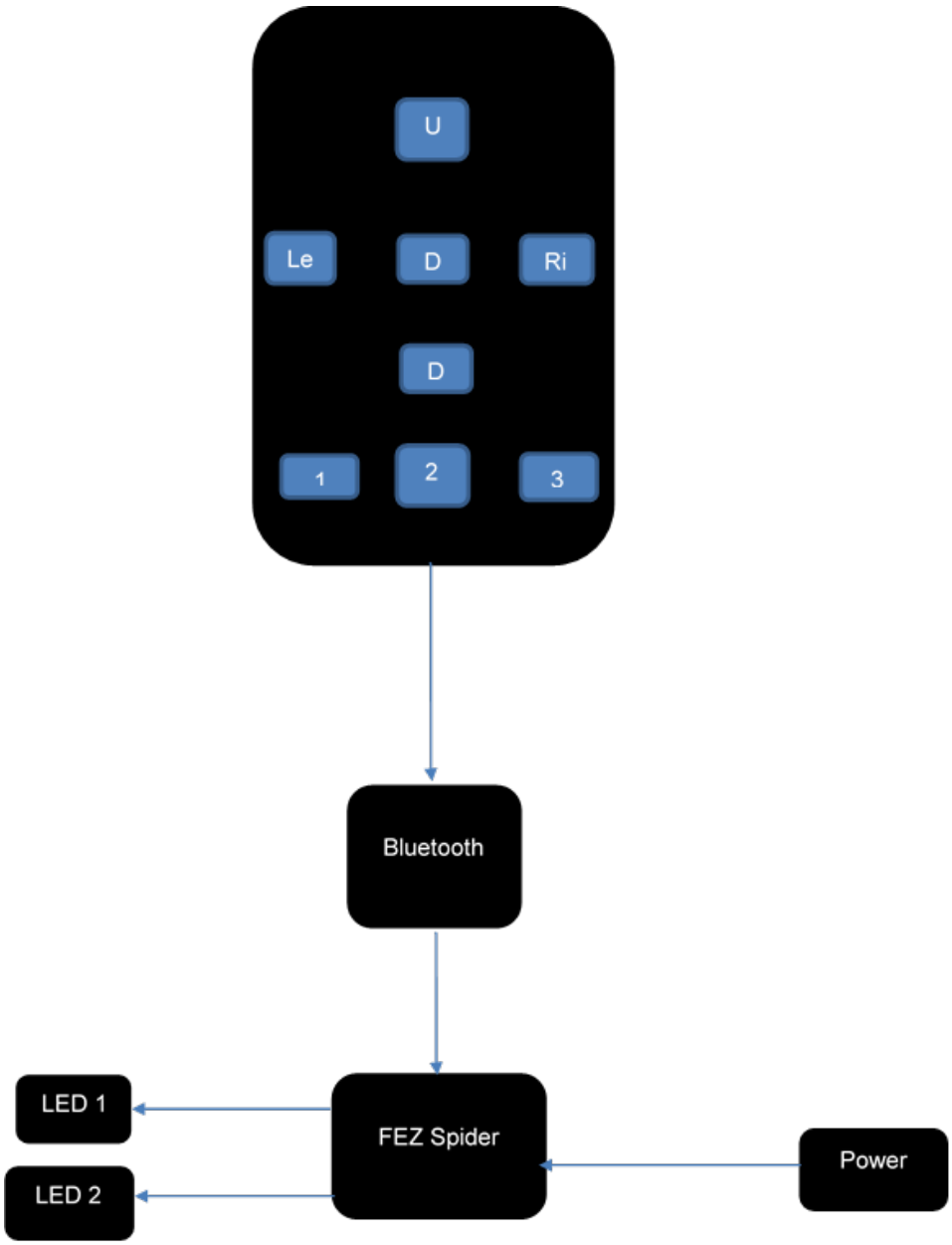

Figure 1: System Overview

5 Hardware Units

5.1 FEZ Spider 1.0

The key component of the GHI Electronics FEZ Spider Kit is the FEZ Spider Mainboard which includes a processor and memory 14 sockets. The sockets are framed by white squares with the socket numbers. It puts the socket number and set of letters into categories which indicate the specific modules that can be connected to the socket. Figure 2 gives the details of names to every sockets.

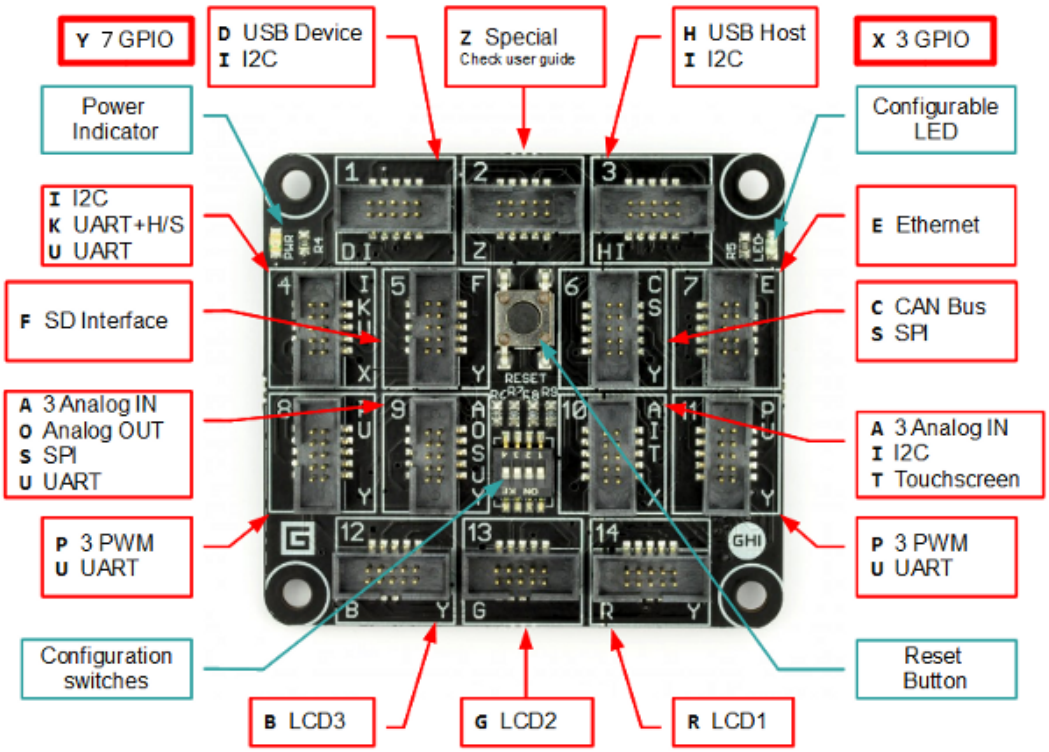

Figure 2: FEZ Spider ports description(copied from GHI Electronics , https://www.ghielectronics.com/catalog/product/269)*

Microsoft's Visual Studio-compatible hardware modules connected to the FEZ Spider mainboard have functionalities such as communication, user interaction,
Sensing, and actuation capabilities.

The FEZ Spider mainboard also has a Reset button to reboot the system. There is a small LED which lights up whenever the FEZ Spider has power.

The FEZ Spider is based on the EMX System on Module. This board is included in the FEZ Spider Starter Kit. [6].

5.2 Power Module.

The USB Client Dual Power Device Module has 2 ports 7 -30Volt external power input and the USB input The USB Client Dual Power (USBClientDP) module (coloured red) is able to connect the FEZ Spider Mainboard to computer for programming and debugging. The dual-powered module can be powered either by a USB port on USBClientDP module supplies power to the FEZ Spider and to any other modules that are connected to it. It can be plugged in both power sources of the USBClientDP module to program and to power at the same time.The Power device is connected to Socket 1 on the FEZ Spider.Figure 3 shows the exact USB Client DP 1.3 with the actual sockets on it.

Figure 3: USB Client DP (Dual Power) Module(copied from https://www.ghielectronics.com/downloads/man/FEZSpider_Starter_Kit_Guide.pdf)

5.3 Bluetooth 1.1 (GM-312)

The Bluetooth module enables Gadgeteer mainboards with a wireless connection to connect to any other Bluetooth device with SPP (Serial Port Profile), such as mobile phones and laptops. This module can host a connection or search and hook onto other connections.Figure 4 shows the Bluetooth 1.1 (GM-312)

Figure 4: Bluetooth Module(copied from https://www.ghielectronics.com/catalog/product/312)

5.4 Multi-colour LED module

Figure 5 shows the Multi-colour LED module.

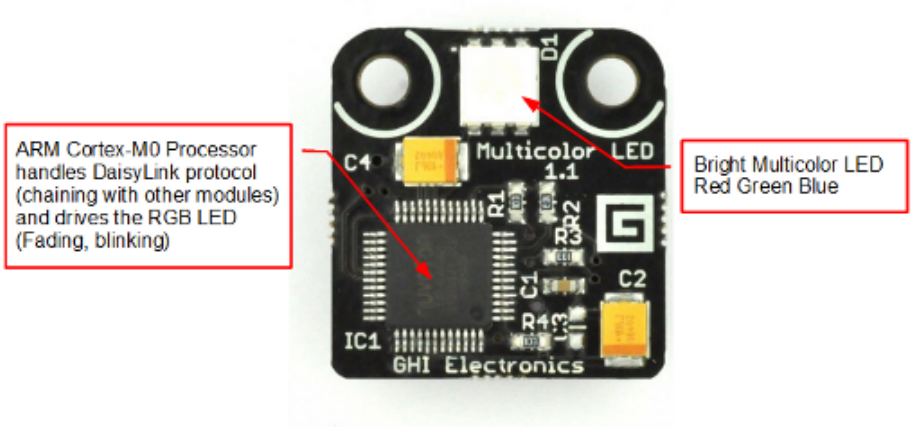

Figure 5: Multi color LED module(copied from (https://www.ghielectronics.com/downloads/man/FEZSpider_Starter_Kit_Guide.pdf)

This smart module includes a 32bit microcontroller. The stock firmware controls a bright multicolour LED (fade, blink, etc) through.NET Gadgeteer DaisyLink Protocol Commands sent from the .NET Gadgeteer main board. It also allows chaining with other DaisyLink modules. Advanced users may program or customize the on-board microcontroller. [7].

5.5 Display T35 1.5

The display is an optional device for the study time.However, it is listed on the hardwares of GHI Electronics.

Figure 6 shows the LCD Module.

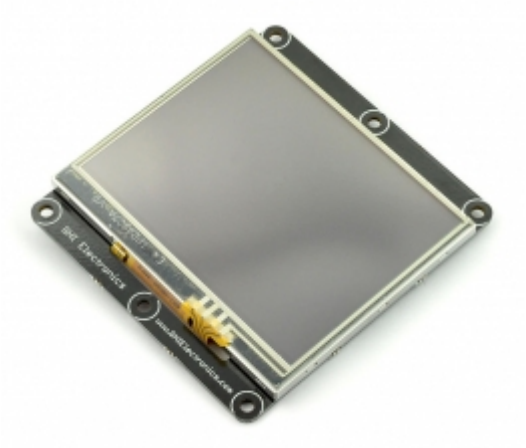

Figure 6: Multi color LED module(copied from (https://www.ghielectronics.com/downloads/man/FEZSpider_Starter_Kit_Guide.pdf)

6 Schematics

The schematics are built in the Spider Scratch code. It shows the connections of the hardware in visual studio (Figure 7). The aim is to generate new lines to adapt new added component -the LED 12 (For Details see Page 21).

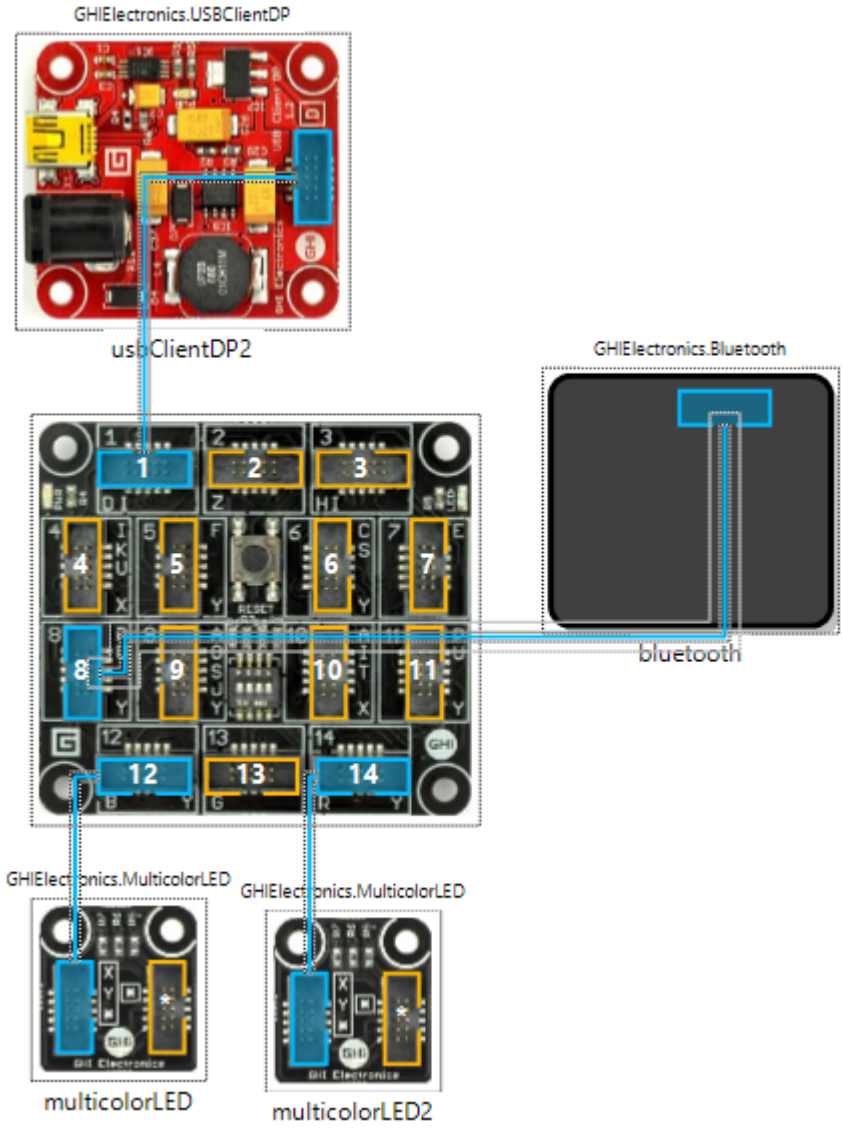

Figure 7: Connection scheme(This schematic was from the Spider_Scratch which I created in visual studio)

Software in phone app connects to Bluetooth module which is connected to spider. Phone sends instructions to spider and spider converts it to LEDS signals.

7 BlueWinPhoneClient

7.1 Installation of BT-SPP WindowsPhoneClient on Windows Phone

One of the aims of this study was to become acquainted with programming the remote controller on WP. However, we came to use ready-made code for WP remote controller which could also be downloaded as a ready-made executable program. To demostrate

how the WP controller would finally work it is better at this point to download the executable now for the beginning from the Windows App Store.

Steps

1 Windows Phone Store
2 BT -SPP WindowsPhoneClient and Installation as shown in Figure 8.

Figure 8: Bluetooth client application

8 The Creation of Buttons on BlueWinPhoneClient in Window Phone

There are two pages in the interface in WP remote controller. No change has been made on the first page below (Figure 9). There are eight buttons, plus the Back button, on the screen on the second page in total (Figure 10). Some functionalities were added to Button 1 and Button 2 and button 3, and a new button, Button 4.

For every button there is a program which is executed. The commands executed are sent from WP to Spider; that means that WP sends them in text form to Spider. The communication between WP and Spider happens through Bluetooth link as letters (as shown in Figure 9, “Hello from MySpider”). The Spider will decipher the command text and find any request that certain LED to be lighten up.

Figure 9: Responce from Spider

Figure 10: Application Interface(Photographed from my WP Phone)

Table 1 shows details of the Structure of Button Design

Table 1: Button design structure

Column / Row	0	1	2
0		Up	
1	Left	Do	Right
2		Down	
3	1	2	3

The Width of every Button is always 85, The Content shows the name of the Button. Columns and Rows are highlighted by Yellow in the following Code, all of the Letters are in Center.

9 Designing Screens in Visual Studio

BlueWinPhoneClient.sln contains the source code that creates user interface of the application.

9.1 In “MainPage.xaml”

Details of how the buttons in the windowsphoneclient is created are described is listing 1.

Listing 1 shows the code for creating buttons from MainPage.xaml

```
</Grid.RowDefinitions>
                                    <Button x:Name="UpButton"
Grid.Column="1" Grid.Row="0" HorizontalAlignment="Center"
Content="Up" Margin="0,5,0,5" Width="85"
Click="UpButton_Click"/>
                                    <Button
x:Name="LeftButton" Grid.Column="0" Grid.Row="1"
HorizontalAlignment="Left" Content="Left"
Margin="0,5,5,5" Width="85" Click="LeftButton_Click"/>
                                    <Button
x:Name="CenterButton" Grid.Column="1" Grid.Row="1"
HorizontalAlignment="Center" Content="Do"
Margin="0,5,5,5" Width="85" Click="CenterButton_Click"
Background="{x:Null}"/>
                                    <Button
x:Name="RightButton" Grid.Column="2" Grid.Row="1"
HorizontalAlignment="Left" Content="Right"
Margin="0,5,10,5" Width="85" Click="RightButton_Click"/>
                                    <Button
x:Name="DownButton" Grid.Column="1" Grid.Row="2"
HorizontalAlignment="Center" Content="Down"
Margin="5,5,10,5" Width="85" Click="DownButton_Click"/>
                                    <Button x:Name="Button_1"
Grid.Column="0" Grid.Row="3" HorizontalAlignment="Left"
```

```
Content="1" Margin="0,5,5,5" Width="85"
Click="Button_1_Click"/>
                                    <Button x:Name="Button_2"
Grid.Column="1" Grid.Row="3" HorizontalAlignment="Center"
Content="2" Margin="0,5,5,5" Width="85"
Click="Button_2_Click"/>
                                    <Button x:Name="Button_3"
Grid.Column="2" Grid.Row="3" HorizontalAlignment="Left"
Content="3" Margin="0,5,5,5" Width="85"
Click="Button_3_Click" Background="{x:Null}"/>
```

Listing 1. Code for creating buttons.

9.2 The Execution of the Program

The following is the part of a source code of “MainPage.xaml.cs?” file which shows how it is executed. ("CMD Up"); means the function in bracket (CMD-Up) is sent to Spider. They are highlighted in blue.

Listing 2 shows the code for the execution of the program in “MainPage.xaml.cs?” file.

```
#region Control Panel Command Button Clicks
        private void UpButton_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_Up");
        }

        private void LeftButton_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_Left");
        }
```

```
        private void CenterButton_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_Do");
        }

        private void RightButton_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_Right");
        }

        private void DownButton_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_Down");
        }

        private void Button_1_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_1");
        }

        private void Button_2_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_2");
        }

        private void Button_3_Click(object sender,
RoutedEventArgs e)
        {
            sendCommand("CMD_3");
```

Listing 2. Functions Execution.

10 The Chatbox

The following part shows what happens when clicking the created button and how the command is send to the Spider. Figure 11 shows sending commands to Spider.

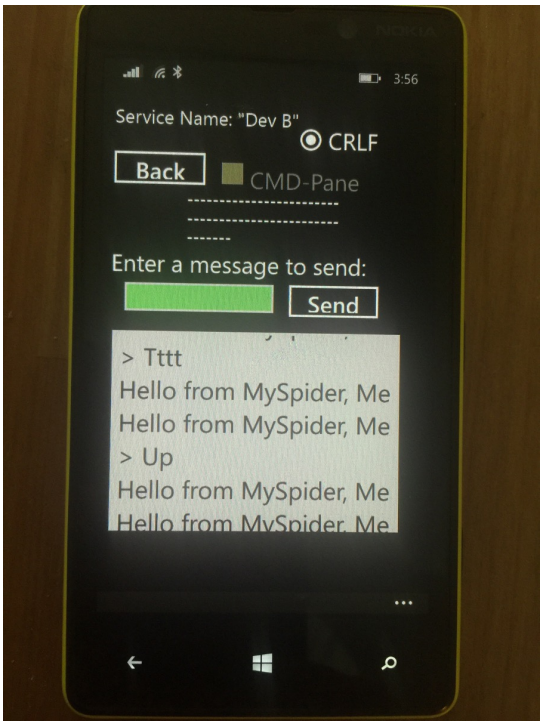

Figure 11: Sending commands to Spider.

Write any Letters like ‘Tttt’ or ‘Up’ and then send text by pressing button SEND when pressing ‘SEND’ the text is sent to spider, and Spider reads all the lines.Lines can also be changed and many lines can be sent. When spider gets the text it takes ONE line at a time even if these commands don’t have any meaning or function. Now, try writing ‘CMD_UP’ and pressing SEND which has the same effect as pressing button ‘Up’ on page 2. Send one line, the spider takes it and puts it to that variable which has name ‘to data line 2’.

Next program also goes to where it is examined.It tells if the Letters of Command ‘LEFT’ is in the text.Then it texts if Command ‘DO’ is in the text and also Command ‘UP’……..

‘0’ is shown on the Control Panel if the text wanted is not in Command Line.

Finally it finds that the text exists, the function gives the numbers of the first ring.
The LEDs come up when the spider received it. The Function datalineindexof tells where the first letter of text is. All of the commands can be put at the same time to the line and all of them are executed.

Listing 3 shows here some important code snippets and to get correct indentation and formatting used in program codes.

You can easily see that this code is used in connection of the buttons of the first page of the WP interface. The important function names are emphasised with blue background colour.

The reader can try to identify the function names SendButton_Click, sendCommand, GoBack_Click, CloseApp_Click with the corresponding buttons.

Listing 3 shows code for sending button click and sending command.

```
#region SendButton_Click and sendCommand
        private async void SendButton_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                chatWriter.WriteString(MessageTextBox.Text + AppendString);
                await chatWriter.StoreAsync();
                this.updateChatBox("> " + MessageTextBox.Text);
                MessageTextBox.Text = "";
            }
            catch (Exception)
            {
                NotifyUser("Cannot send, no Bluetooth Connection",
                   NotifyType.StatusMessage);
                ClearNotify(4);
            }
        }

        private async void sendCommand(string pMessage)
        {
            try
            {
                chatWriter.WriteString(pMessage + "\r\n");
```

```
            await chatWriter.StoreAsync();

            this.updateChatBox("> " + pMessage);
        }
        catch (Exception)
        {
            NotifyUser("Cannot send, no Bluetooth Connection",
               NotifyType.StatusMessage);
            ClearNotify(4);
        }
    }
    #endregion

    #region GoBack_Click

    private void GoBack_Click(object sender, RoutedEventArgs e)
    {
        GoBack.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
        ServiceSelector.Visibility = Windows.UI.Xaml.Visibility.Visible;
        HeadGrid.Visibility = Windows.UI.Xaml.Visibility.Visible;
    }

    #endregion

    #region CloseApp_Click
    private async void CloseApp_Click(object sender, RoutedEventArgs e)
    {
        StreamReadManager.cancelSocket();
        await Task.Delay(TimeSpan.FromMilliseconds(100));

        if (FluxToggle.IsOn)
        {
            FluxToggle.IsOn = false; //Release Display Request
        }

        if (chatReader != null)
        {
            try
            {
                chatReader.Dispose();
                chatReader = null;
            }
            catch (Exception ex5)
            {
                NotifyUser("Error: " + ex5.HResult.ToString() + " - " + ex5.Message,
                NotifyType.ErrorMessage);
            }
        }

        if (chatWriter != null)
```

```
    {
        try
        {
            chatWriter.DetachStream();
            chatWriter.Dispose();
            chatWriter = null;
        }
        catch (Exception ex6)
        {
            NotifyUser("Error: " + ex6.HResult.ToString() + " - " + ex6.Message,
           NotifyType.ErrorMessage);
        }
    }

    lock (this)
    {
        if (chatSocket != null)
        {
            try
            {
                chatSocket.Dispose();
                chatSocket = null;
            }
            catch (Exception ex7)
            {
                NotifyUser("Error: " + ex7.HResult.ToString()
                                                + " - " + ex7.Message,
      NotifyType.ErrorMessage);
            }
        }
    }

    tbData.Text = "";
    Application.Current.Exit();
}
#endregion
```

Listing 3. Code for button clicks and sending command.

11 Details of the Creation of Control Buttons in Spider controller

11.1 New LED

To make some real change, a new LED to Socket 12 - LED 2 was added to accompany the original, multicolour LED module using Socket 14 of the main board. Private gadgeteer module and the followings were created. To change the colour of LED 2, 'what I copied code 3' and emphasised with yellow were added,emphasised with green the important function names which the reader can try to identify with the button names on WP programs interface page 2.

You can identify the following functions from listing 4.

Both of the LEDS go off If 'DO' is pressed (Figure 12). Both of them become blue if 'UP' is pressed. Multicolour LED goes to yellow when pressing CMD 'RIGHT'.

Figure 12: Second LED module additon

Listing 4 shows code for the adding of the new LED

```
#region  Eventhandler --  Receive data as Byte Arrays
            // Select this output mode with the
                            bluetooth.SetReadMode_ByteArray(true) command
            void bluetooth_DataByteArrayReceived(BluetoothBee sender, byte[] data, int
                            startIndex, int ByteCount)
                            // This command receives Bluetooth data and save it
            {
                if (ByteCount < 100)   // for < 100 it is assumed,
                            that ccommand strings are received
                {
                            // larger Dataarrays may be from e.g. file transfer
                             and must be handeld otherwise
                            //Put data in buffer, we get them in the following
                             while loop after a hex \0A was received
                    mySerialBuffer.LoadSerial(data, startIndex, ByteCount);
                }
                else                  // for high speed data we only send the statistics
                {
                    // Do something else with the incoming data
                }
                while ((dataLine2 = mySerialBuffer.ReadLine()) != null)
                {

                    Debug.Print(dataLine2 + "\r\n");

                    // parse the data sentence here if needed
                    if (dataLine2.IndexOf("CMD_Left") >= 0)
                    {
                        //multicolorLED.TurnRed();
                        multicolorLED.TurnWhite();
                        multicolorLED2.TurnWhite();//----------------------------------
                    }

                    if (dataLine2.IndexOf("CMD_Do") >= 0)
                    {
                        multicolorLED.TurnOff();
                        multicolorLED2.TurnOff();//-------------------------------------
                    }

                    if (dataLine2.IndexOf("CMD_Right") >= 0)
                    {
                       // multicolorLED.TurnGreen();
                        multicolorLED.TurnColor(GT.Color.Yellow);
                        multicolorLED2.TurnColor(GT.Color.Yellow);//--------------------
--
                    }

                    if (dataLine2.IndexOf("CMD_Up") >= 0)
```

```
            {
                multicolorLED.TurnBlue();
                multicolorLED2.TurnBlue();// what I copied 3
            }

            if (dataLine2.IndexOf("CMD_Down") >= 0)
            {
                multicolorLED.TurnOff();
                multicolorLED.BlinkRepeatedly(GT.Color.Blue);
                multicolorLED2.TurnOff();//-------------------------------------
                multicolorLED2.BlinkRepeatedly(GT.Color.Blue);//----------------
            }

            if (dataLine2.IndexOf("CMD_1") >= 0)
            {
                // multicolorLED.TurnGreen();
                multicolorLED.TurnColor(GT.Color.White);
                multicolorLED2.TurnColor(GT.Color.White);//---------------------
            }
            if (dataLine2.IndexOf("CMD_2") >= 0)
            {
                // multicolorLED.TurnGreen();
                multicolorLED.TurnColor(GT.Color.Magenta);
                multicolorLED2.TurnColor(GT.Color.Magenta);//-------------------
            }

            if (dataLine2.IndexOf("CMD_3") >= 0)
            {
                // multicolorLED.TurnGreen();
                multicolorLED.TurnColor(GT.Color.Cyan);
                multicolorLED2.TurnColor(GT.Color.Cyan);//----------------------
            }

            if (dataLine2.IndexOf("CMD_4") >= 0)
            {
                // multicolorLED.TurnGreen();
                multicolorLED.TurnColor(GT.Color.Blue);
                multicolorLED2.TurnColor(GT.Color.Blue);//----------------------

            }
        }
    }
    #endregion
```

Listing 4. Code for the adding of the new LED

The above is about how the WP‘s function is sent to the LED and how the LEDs are changed according to the Commands.

The lines with' ----------'are created to add button functions for more colours.

12 The New LED in Spider-Scratch Code

In the Program.generated.cs

Listing 5 explains how multicolour LED2 was created in the code to adapt the change of adding components.

Listing 5 shows code for the adaption of the new LED

```
public partial class Program : Gadgeteer.Program {

        /// <summary>The Button module using socket 11 of the mainboard.</summary>
        private Gadgeteer.Modules.GHIElectronics.Button button;

        /// <summary>The USB Client DP module using socket 1 of the mainboard.</summary>
        private Gadgeteer.Modules.GHIElectronics.USBClientDP usbClientDP;
                                        e
                              /// <summary>The Multicolor LED module
                               using socket 14 of the mainboard.</summary>

        private Gadgeteer.Modules.GHIElectronics.MulticolorLED multicolorLED;
                              /// <summary>The Multicolor LED module
                               using socket 12 of the mainboard.</summary>

        private Gadgeteer.Modules.GHIElectronics.MulticolorLED multicolorLED2;
                              // what I copied 2


             /// <summary>This property provides access to the Mainboard API.
              This is normally not necessary for an end user program.</summary>

        protected new static GHIElectronics.Gadgeteer.FEZSpider Mainboard {

            get {
                              return
              ((GHIElectronics.Gadgeteer.FEZSpider)(Gadgeteer.Program.Mainboard));
            }

            set {
                Gadgeteer.Program.Mainboard = value;
            }
```

Listing 5. Code for the adaption of the new LED

13 The Creation of BT- Spider- Scratch in Visual Studio

BT- Spider- Scratch is created in visual studio. Starting a new project with Spider, the change of the code is examined in the software. For that there is no man-made code in it, we can see how the device is taken into use. It tells how to take into use the modules. Every components are put together in the graphical interface. The components are connected on the screen of Visual Studio. The new button is able to be detected in the scratch code. The highlighted lines are added into the original code to control the new added component -LED 2 in Socket 12.
So this is the easiest way. After adding the hardware the scratch code generates lines that are changed. The very same lines in original code also need to be changed. The new lines are added in this driver program to send commands to the new hardware in Program .generated .cs

Listing 6 contains what was defined. The Spider Scratch was created to generate new lines.

```
public partial class Program : Gadgeteer.Program {

        /// <summary>The USB Client DP module using socket 1 of the mainboard.</summary>

        private Gadgeteer.Modules.GHIElectronics.USBClientDP usbClientDP2;

        /// <summary>The Bluetooth module using socket 8 of the mainboard.</summary>

        private Gadgeteer.Modules.GHIElectronics.Bluetooth bluetooth;

              /// <summary>The Multicolor LED module using socket 12
                            of the mainboard.</summary>
        private Gadgeteer.Modules.GHIElectronics.MulticolorLED multicolorLED;

              /// <summary>The Multicolor LED module using
                           socket 14 of the mainboard.</summary>

              private Gadgeteer.Modules.GHIElectronics.MulticolorLED multicolorLED2;

              /// <summary>This property provides access to the Spider Mainboard API.
               This is normally not necessary for an end user program.</summary>
        protected new static GHIElectronics.Gadgeteer.FEZSpider Mainboard {
            get {
              return ((GHIElectronics.Gadgeteer.FEZSpider)(Gadgeteer.Program.Mainboard));
            }
```

```
            set {
                Gadgeteer.Program.Mainboard = value;
            }
        }

          /// <summary>This method runs automatically when the device
           is powered, and calls ProgramStarted.</summary>

        public static void Main() {
            // Important to initialize the Mainboard first
            Program.Mainboard = new GHIElectronics.Gadgeteer.FEZSpider();
            Program p = new Program();
            p.InitializeModules();
            p.ProgramStarted();
            // Starts Dispatcher
            p.Run();
        }

        private void InitializeModules() {
            this.usbClientDP2 = new GTM.GHIElectronics.USBClientDP(1);
            this.bluetooth = new GTM.GHIElectronics.Bluetooth(8);
            this.multicolorLED = new GTM.GHIElectronics.MulticolorLED(12);
            this.multicolorLED2 = new GTM.GHIElectronics.MulticolorLED(14);
        }
    }
}
```

Listing 6. Generating new lines.

The lines above initializes USB Client, Bluetooth and 2 LEDs.

14 Problems Encountered and Solution

14.1 Problem 1

The frequent problems occurred in the practical usage are the errors such as Error message:

Cannot display the base assembly 'mscorlib', or any of his satellite assemblies, to device-USB: Gadgetteer twice. Assembly 'mscorlib' on the device has version 4.1 2821.0,

The Solution gotten from the GHI Forum is that the Bluetooth Module in the designer to be connected to the Spider Mainboard. This is forbidden in the project because the driver in the module is contained separately. Attention should be paid to that the version of 'mscorlib' be matched. The progrom can not go on if the errors occur.

14.2 Problem 2

The need for the updating of the firmware that were too old to be operated

The Solution is from GHI Electronics website
Go to Support - .NET Micro Framework and look for help or change to an older version of Windows. Because of the lack of the old version of the hardwares,it posed a challenge to update them but fortunately the updated hardwares were obtained before the company totally stopped the offering.

14.3 Problem 3

The emulator cannot find the BT ´device during the running of the program in visual studio.

An example error occurred during the running of the program in Visual Studio.

Figure 13 shows the problem in finding bluetooth device

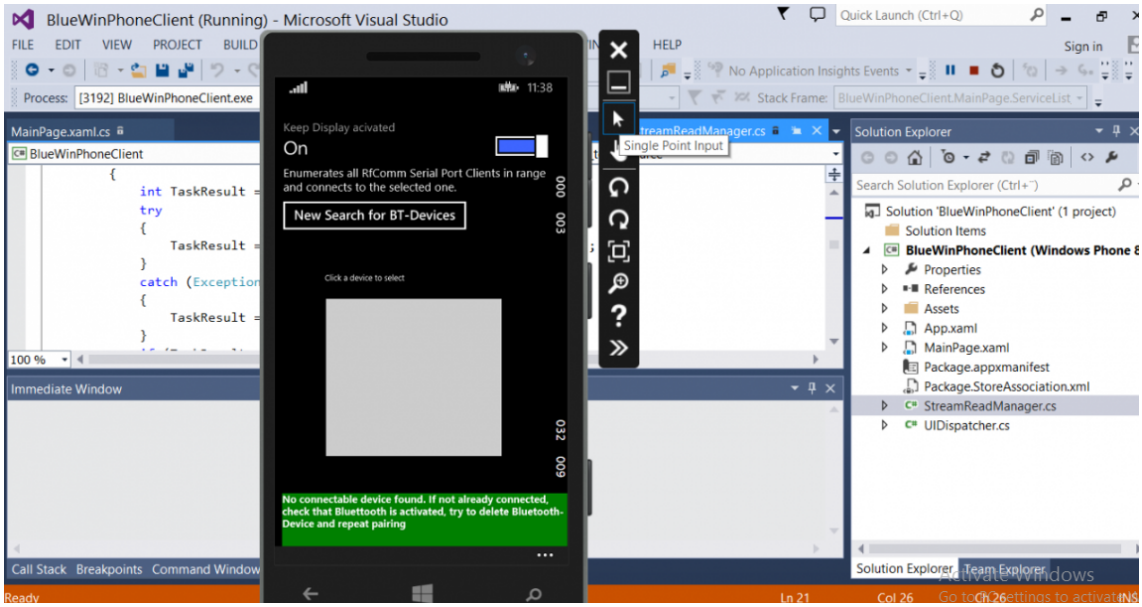

Figure 13: Problem in finding a bluetooth device

The solution

Finally the emulator option abandoned .BT-SPP-WinPhoneClient is downloaded in the WP to control the Spider.

14.4 Problem 4

The specific U connector for connector 11

Solution

According to RoSchmi, it is set to Socket 8. And set the correct #define directive for the GHI Bluetooth Module in the class BluetoothBee.cs

```
#define GHI_Bluetooth_Module
//#define Bluetooth_Bee_Gadgeteer
//#define Bluetooth_Bee_Cerbuino
```

14.5 Problem 5

Failed to connect to the device as it is pin locked.

Solution

Enter pin code.

14.6 Problem 6

Debugging the Windows phone code and the spider code

Solution

Examine Breakpoints and the content of variables. Install version 2015 with update 2. For details see[8].

15 Final Implementation

Figure 14 shows UP button response

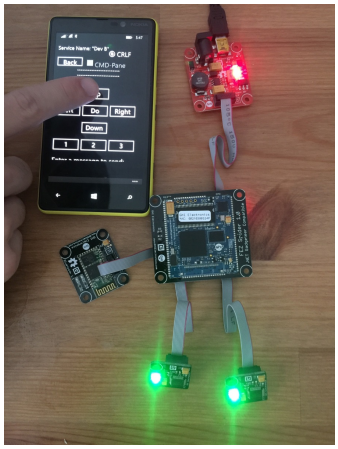

Figure 14: UP button response

When pressing Up Button, both LEDS turn green.

Figure 15 shows Down button response

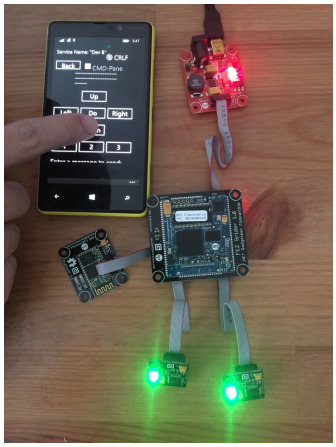

Figure 15: Down button response

When pressing Down button, both LEDS turn green

Figure 16 shows LEFT button response

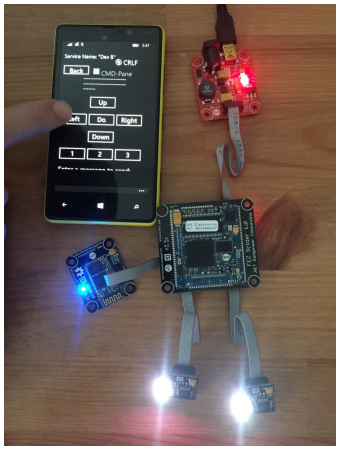

Figure 16: LEFT button response

When pressing Left Button, Both LEDS turn White.

Figure 17 shows RIGHT button response

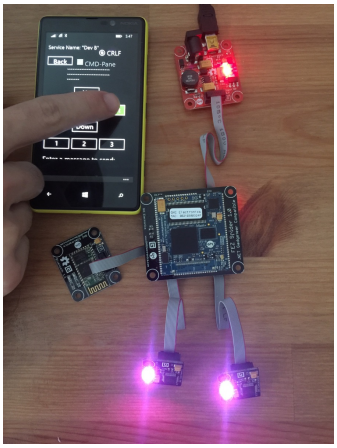

Figure 17: RIGHT button response

When pressing Right button, both LEDS turn purple.

Figure 18 shows DO button response

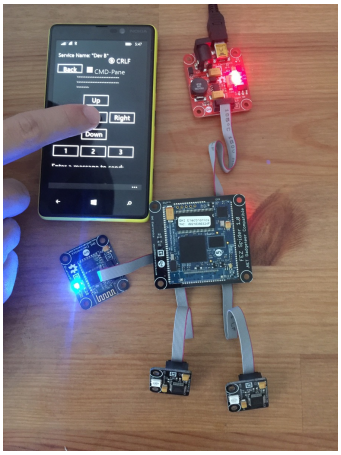

Figure 18: DO button response

When pressing Do button, both LEDS turn off.

Figure 19 shows 1 button response

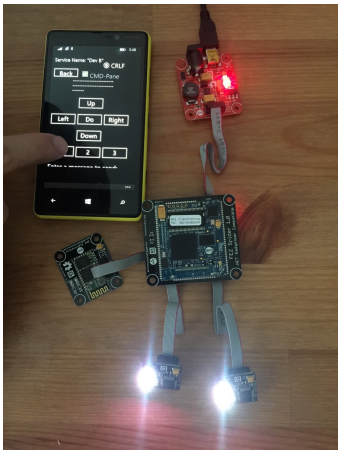

Figure 19: 1 button response

When pressing Button1, both LEDS turn white.

Figure 20 shows 2 button response

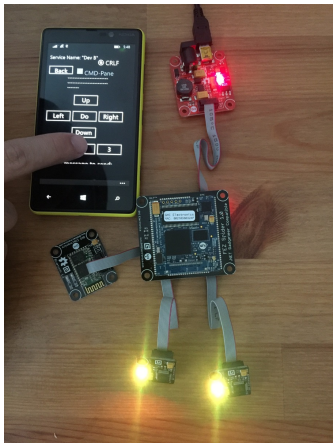

Figure 20: 2 button response

When pressing Button2, both LEDS turn yellow.

Figure 21 shows 3 button response

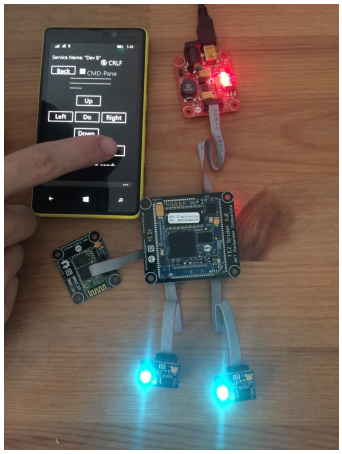

Figure 21: 3 button response

When pressing Button3, both LEDS turn blue.

16 Conclusion

During the building of the gadget successes were achieved and the WP was able to simulate the TV remote- controller to send commands to spider in which multicolour LEDs are switched on and off. The spider now works independently in visual studio with the code and it needs only power. The spider obeys the command sent by real Windows phone. Loading the program to Windows phone from Microsoft store or by visual studio was availlable.

The running of Visual Studio has some problem with debugging but it works in general. Although the colour of the LEDs sometimes don't match quite well when sending commands, the overall goal of controlling was finally achieved.

17 References

1. https://www.ghielectronics.com/community/codeshare/entry/1004 [Accessed on 20-08-2017].

2. https://www.ghielectronics.com/community/forum/topic?id=18204&page=3 [Accessed on 20-08-2017].

3. *https://github.com/akshay2000/XBMCRemoteRT* [Accessed on 20-08-2017].

4. https://www.ghielectronics.com/support/netmf [Accessed on 20-08-2017].

5. https://developer.microsoft.com/en-us/windows/downloads/sdk-archive [Accessed on 20-08-2017].

6. https://www.ghielectronics.com/catalog/product/269 [Accessed on 20-08-2017].

7. https://www.ghielectronics.com/catalog/product/272 [Accessed on 20-08-2017].

8. https://www.ghielectronics.com/community/forum/topic?id=18204&page=3 [Accessed on 20-08-2017].