**Content**

Abstract ......................................................................................................................................2

ACKNOWLEDGEMENT .........................................................................................................3

Abbreviation ......................................................................................................................5

1 Theoretical Background ............................................................................................6

1.1 Types of OCR .............................................................................................5

1.1.1 Offline Handwritten Text ................................................................5

1.1.2 Online Handwritten Text ................................................................5

1.1.3 Machine Printed Text .....................................................................5

1.2 Uses of OCR ..........................................................................................................5

1.3 Benefits and Pitfalls of OCR ...................................................................................6

1.4 Current state of OCR technology ............................................................................6

2 Hardware Requirement ..............................................................................................7

2.1 What is Gadgeteer .......................................................................................7

2.2 FEZ Spider Kit .............................................................................................7

3 Setup Development Environment .............................................................................13

4 Project Analysis ......................................................................................................15

5 Limitation ..............................................................................................................18

Conclusion .............................................................................................................................19

References .....................................................................................................................20

**Abstract**

The scope of the project is to develop an Optical Character Recognition program. **Optical character recognition** usually abbreviated to OCR, involves a computer system designed to translate images of typewritten text (usually captured by a scanner) into machine editable text or to translate pictures of characters into a standard encoding scheme representing them. OCR began as a field of research in artificial intelligence and computational vision. The process of taking images is developed on FEZ Spider kit and the process of image processing and character recognition is done on a computer/laptop.

**ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

A special gratitude towards Prof. Dr. Andreas Pech, for his valuable encouragement and guidance.

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report.

**Abbreviation**

**LED**

Light Emitting Diode

**NET**

Network Enabled Technology

**OCR**

Optical Character Recognition

**SDK**

Software Development Kit

**USB**

### [Universal Serial Bus](https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCQQFjAA&url=http%3A%2F%2Fde.wikipedia.org%2Fwiki%2FUniversal_Serial_Bus&ei=S1RRVf7IOseqUZDEgbgG&usg=AFQjCNG_KrtI9KY80cIWKEAWXvfyn5d0qw)

**1 Theoretical Background**

Optical Character Recognition, usually abbreviated to OCR, is the mechanical or electronic translation of images of handwritten, typewritten or printed text into machine-editable text. The images are usually captured by a scanner. However, throughout the text, we would be referring to printed text by OCR. Data Entry through OCR has faster speed, more accuracy, and generally more efficiency than keystroke data entry.[1]

**1.1 Types of OCR**

Basically, there are three types of OCR. They are briefly discussed below:

**1.1.1 Offline Handwritten Text**

The text produced by a person by writing with a pen/ pencil on a paper medium and which is then scanned into digital format using scanner is called Offline Handwritten Text.

**1.1.2 Online Handwritten Text**

Online handwritten text is the one written directly on a digitizing tablet using stylus. The output is a sequence of x-y coordinates that express pen position as well as other information such as pressure (exerted by the writer) and speed of writing.

**1.1.3 Machine Printed Text**

Machine printed text can be found commonly in daily use. It is produced by offset processes, such as laser, inkjet and many more.

**1.2 Uses of OCR [[1]](#footnote-1)**

Optical Character Recognition is used to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data.

The OCR technology can also be used for the following purposes:

a. Processing checks

b. Documenting library materials

a. Storing documents, searching text and extracting data from paper based documents

**1.3 Benefits and Pitfalls of OCR**

Every coin has two parts; so does the OCR technology. The following are the benefits of OCR:

a. It can be used to scan and preserve historical documents.

b. It can be used to convert scanned documents into searchable text.

c. It can be used in obtaining the scanned data at the minimum time, with greater ease.

The pitfalls of OCR have been listed below:[[2]](#footnote-2)

a. It doesn’t work well the materials having font size less than 10.

b. It is useless in taking the proper threshold value. It may end up losing the necessary pixels.

c. It is difficult to take the scan sample of the forms which includes lots of boxes.

**1.4 Current state of OCR technology**

The accurate recognition of Latin-script, typewritten text is now considered largely a solved problem. Typical accuracy rates exceed 99%, although certain applications demanding even higher accuracy require human review for errors. Other areas—including recognition of hand printing, cursive handwriting, and printed text in other scripts (especially those with a very large number of characters)--are still the subject of active research.

Optical Character Recognition (OCR) is sometimes confused with *on-line* character recognition. OCR is an instance of off-line character recognition, where the system recognizes the fixed *static shape* of the character, while on-line character recognition instead recognizes the *dynamic motion* during handwriting.

Recognition of cursive text is an active area of research, with recognition rates even lower than that of hand-printed text. Higher rates of recognition of general cursive script will likely not be possible without the use of contextual or grammatical information. For example, recognizing entire words from a dictionary is easier than trying to parse individual characters from script. Reading the *Amount* line of a cheque (which is always a written-out number) is an example where using a smaller dictionary can increase recognition rates greatly. Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy. The shapes of individual cursive characters themselves simply do not contain enough information to recognize all handwritten cursive script accurately (greater than 98%).

It is necessary to understand that OCR technology is a basic technology also used in advanced scanning applications. Due to this, an advanced scanning solution can be unique and patented and not easily copied despite being based on this basic OCR technology.

For more complex recognition problems, intelligent character recognition systems are generally used, as artificial neural networks can be made indifferent to both affine and non-linear transformations.

**2 Hardware Requirement**

### 2.1 What is Gadgeteer

Microsoft .NET Gadgeteer is an open-source toolkit for building small electronic devices using the .NET Micro Framework. It combines the advantages of object-oriented programming, solderless assembly of electronics, and support for customizable physical design.[2]

**2.2 FEZ Spider Kit**

The FEZ Spider Kit consists of components, which are called *modules*, and cables that can use to create various types of functionality in device. To create first .NET Gadgeteer device, following parts will be needed:

* FEZ Spider Mainboard[[3]](#footnote-3)
* A Red USB Client Dual Power (USBClientDP) module
* A Button module
* A Camera module
* A Display\_T35 module
* Module connector cables

##### **The FEZ Spider Mainboard**

The FEZ Spider Mainboard includes a processor and memory, as well as 14 sockets. The sockets are outlined by a white box that surrounds the socket number [3](**1** through **14**) and groups the socket number with a set of letters that indicate which modules can be connected to the socket.

.NET Gadgeteer-compatible hardware modules connected to the FEZ Spider mainboard by these connectors allow you to extend the FEZ Spider mainboard with communication, user interaction, sensing, and actuation capabilities.[4]

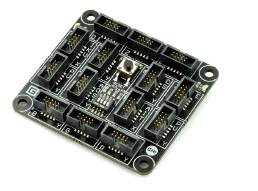
The FEZ Spider mainboard includes a Reset button to reboot the system. There is also a small LED (labelled D1) which lights up whenever the FEZ Spider has power.

fig-1: Mainboard

##### **The USB Client Dual Power Device Module**

The USB Client Dual Power (USBClientDP) module (coloured red) enables you to connect the FEZ Spider Mainboard to your computer for programming and debugging. The dual-powered module is itself powered either by a USB port on a computer or by a 7 -30 volt DC power source. A USBClientDP module supplies power to the FEZ Spider and to any other modules that are connected to it. You can plug in both power sources of the USBClientDP module to program and to power at the same time.[[4]](#footnote-4)

##### **Warning**

Never connect more than one red module to the FEZ Spider Mainboard at the same time. This will damage the hardware.

The USBClientDP module has a black socket, identical to the sockets on the FEZ Spider Mainboard. Next to the connector, there is a letter **D**. **This means that this particular module can only be connected to a socket labelled D on the mainboard.**

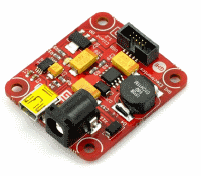
In a similar way, all .NET Gadgeteer-compatible modules have letters next to their sockets that identify which mainboard sockets they can be connected to. Many modules are labelled with multiple letters. This means that they can be connected to **any**of the labelled sockets.[5]

fig-2: Red USB Device Module

##### **The Module Connector Cable**

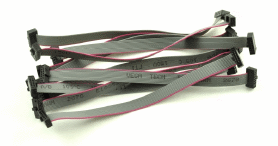
Your hardware kit includes many module connector cables of different lengths. Apart from the length, these cables are all identical and can be used interchangeably to connect modules to the FEZ Spider Mainboard. Note that all sockets have a notch and the cable headers have a protrusion that fits into this notch, so the cables can only be inserted one way.[[5]](#footnote-5)

fig-3: A .NET Gadgeteer-compatible Module Connector Cable

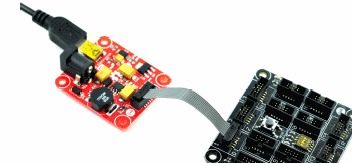
Connect the red USB Device module to socket number 1 on the FEZ Spider Mainboard, which is the only socket that has the letter D. Then, connect the small end of the mini USB cable provided with the kit to the USBClientDP module. However, do not connect the other end to your computer yet.

fig-4: Connecting the FEZ Spider to the USB Client module

##### **Warning**

When plugging or unplugging any module into a FEZ Spider socket, always make sure that power is not connected, by unplugging either end of the mini USB cable. The mini USB cable supplies power to the FEZ Spider; if you plug or unplug a module on the FEZ Spider while it is powered, the hardware could be damaged.[6]

After ensuring that the FEZ Spider mainboard is not powered, continue by getting a Button module from your hardware kit.

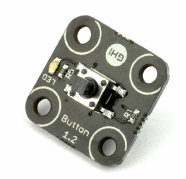
****

fig-5: A Button with Multicolour LED

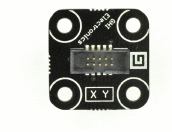
Turn the Button module over. Next to the connector are the letters **X Y**. This means that a Button module can be connected to one of the sockets labelled X or Y on a main board.

fig-6: The Reverse Side of a Button Module

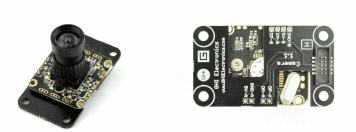
Get a Camera module from your hardware kit. Turn the Camera module over. This module has a [SINGLE](http://blogs.msdn.com/b/uk_faculty_connection/archive/2011/12/08/getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx) connector labelled **H**. Only socket **3** on the FEZ Spider mainboard supports modules labelled with the letter **H**. Plug one end of a connector cable to the socket on the Camera module and the other end to the FEZ Spider on socket number **3**, also labelled **HI**.

fig-7: Camera with Socket labelled H

##### **Connecting the Modules to the Mainboard[[6]](#footnote-6)**

Connect the Button and the Camera to the mainboard using module connector cables as described in the previous sections.

The .NET Gadgeteer designer can identify the sockets on modules and on the mainboard that are compatible. This method is described in the following section titled **Using the .NET Gadgeteer Designer UI**. In this example we will connect the remaining Display\_T35 module manually.

The Display\_T35 module has four connectors. Connect sockets 14, 13, and 12 to the Display\_T35 module sockets labelled **R**, **G**, and **B**. As you might expect, these letters signify the colour distribution of the Display\_T35 module. Connect socket **10** on the mainboard to socket **T** on the Display\_T35 module. Socket **T** on the Display\_T35 module facilitates the touch screen features of this module.

##### **Warning**

Make the actual connections between all modules and the mainboard before you connect the USBClientDP to the USB port on your computer.[7]

The following illustration shows the mainboard connected to a Display\_T35 module, a Button module, a Camera module, and the USBClientDP module. Now, with all the other modules connected, you can connect the USBClientDP module to the USB port on your computer.

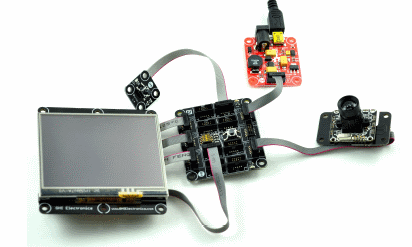
****

fig-8: FEZ spider Modules with Connectors

**3 Setup Development Environment**

This section lists the steps required to setup your development environment. Note that some steps are specific to your particular .NET Micro Framework based hardware and you will need to refer to instructions provided by your hardware vendor.

**Step 1**: Install Microsoft Visual Studio 2013

Any version of Visual Studio 2013 can be used including the Visual Studio Express editions.

**Step 2**: Uninstall any previous versions of the .NET Micro Framework SDK

Make sure you don't have any other versions of the .NET Micro Framework already installed. The .NET Micro Framework SDK cannot coexist with other versions of the SDK.

**Step 3**: Uninstall any third party SDK software[[7]](#footnote-7)

If you have used other .NET Micro Framework software or hardware in the past, make sure to uninstall any software included with your hardware. You will need to get UPDATED SOFTWARE from your hardware manufacturer that are compatible with Visual Studio 2013. The software will be installed in a later step.

**Step 4**: Download the latest version of the .NET Micro Framework

You can download the .NET Micro Framework SDK from the Codeplex site.

**Step 5**: Unzip the downloaded file

Unzip the file that you downloaded in the previous step. Note the location of the unzipped file.

**Step 6**: Run the MicroFrameworkSDK.MSI

Run the MicroFrameworkSDK.msi that was unzipped in the previous step by double clicking the file from a File Explorer window.

**Step 7**: Install the .NET Micro Framework VSIX Extensions

1. In Visual Studio go to Tools Extensions and Updates.
2. Under the Online category, find the .Net Micro Framework extensions for Visual Studio 2013.
3. Select the Download button to download and install the extensions.

At this point you have completed the step to install the core of the .NET Micro Framework.

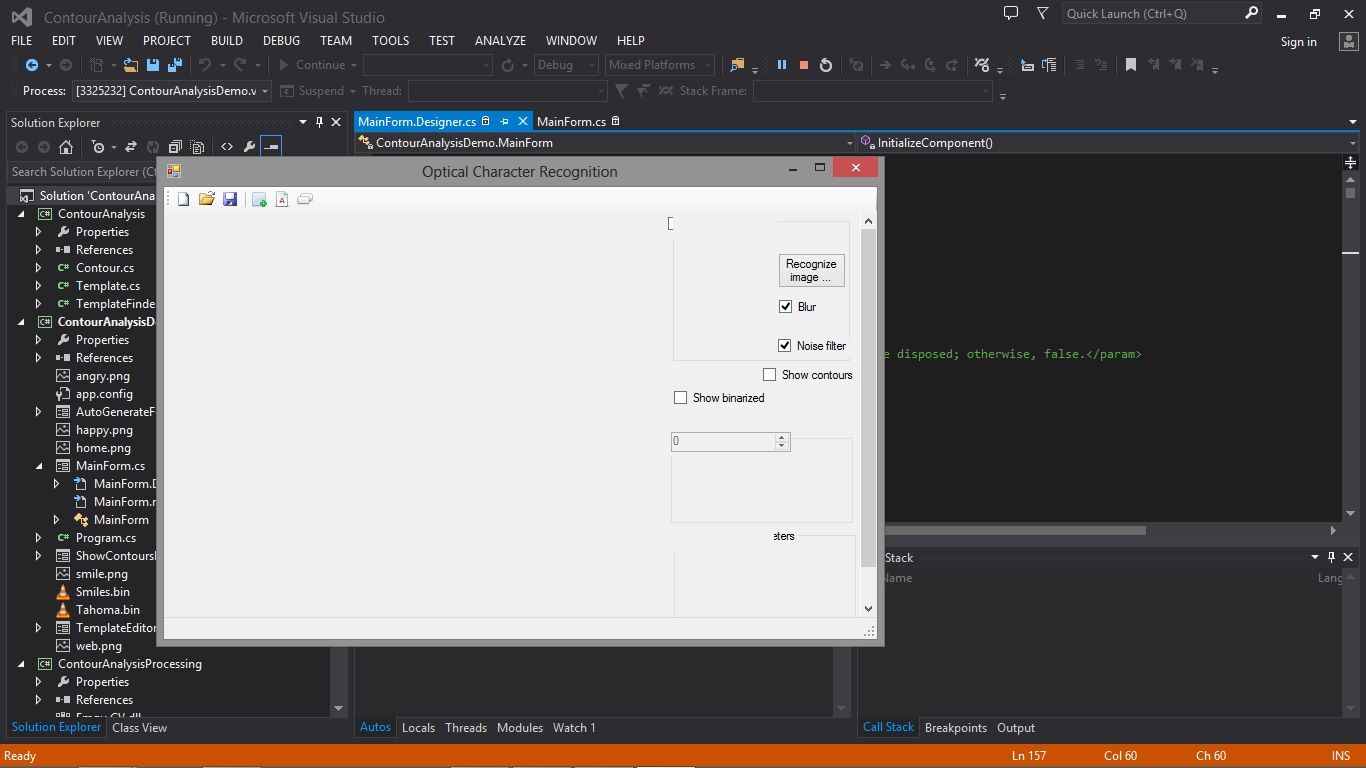
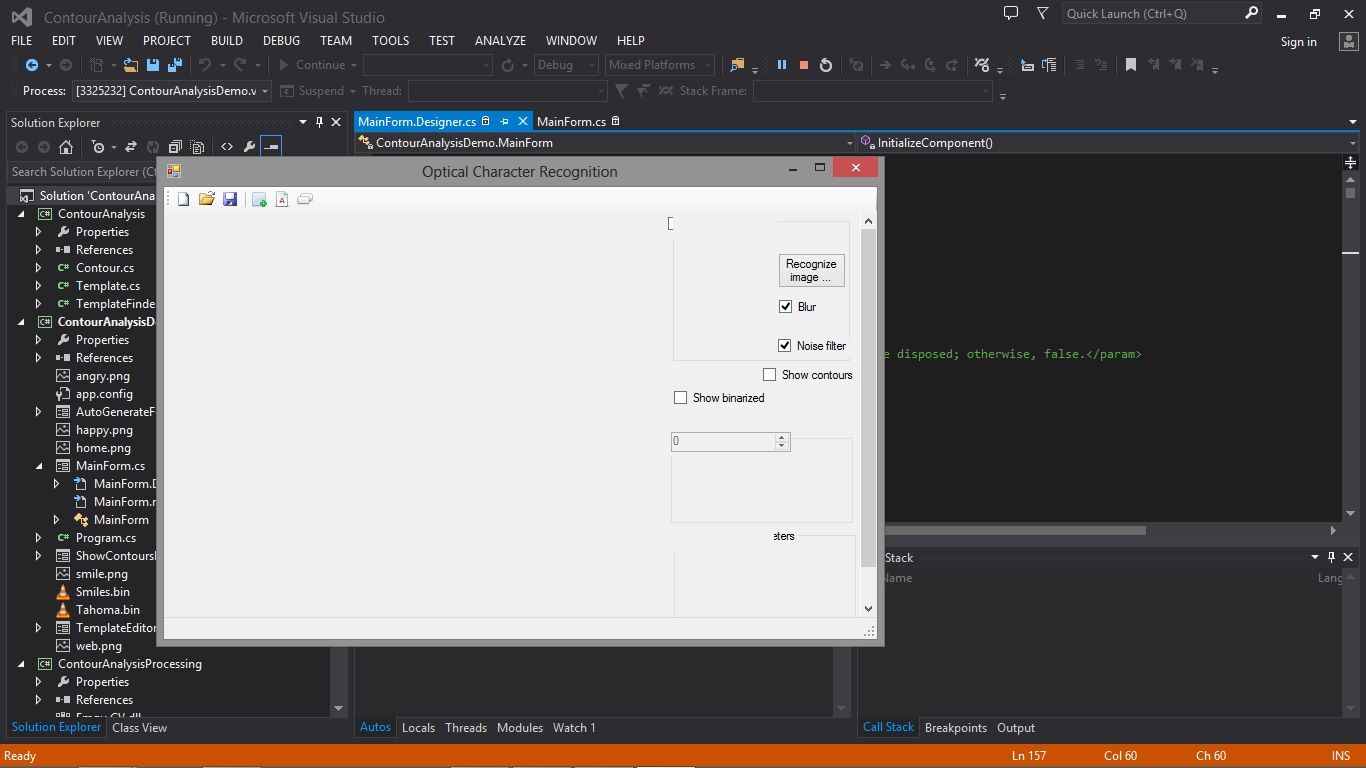
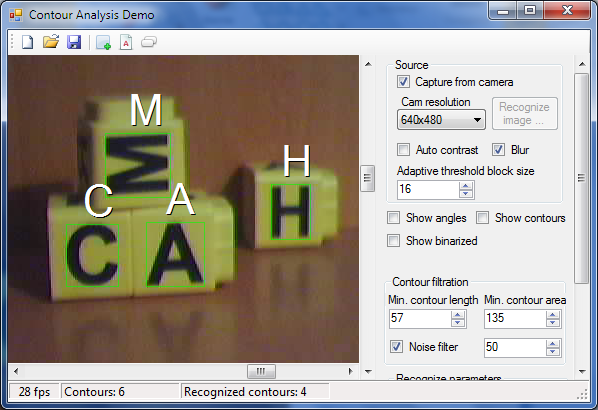
**Step 8**: Install hardware specific software

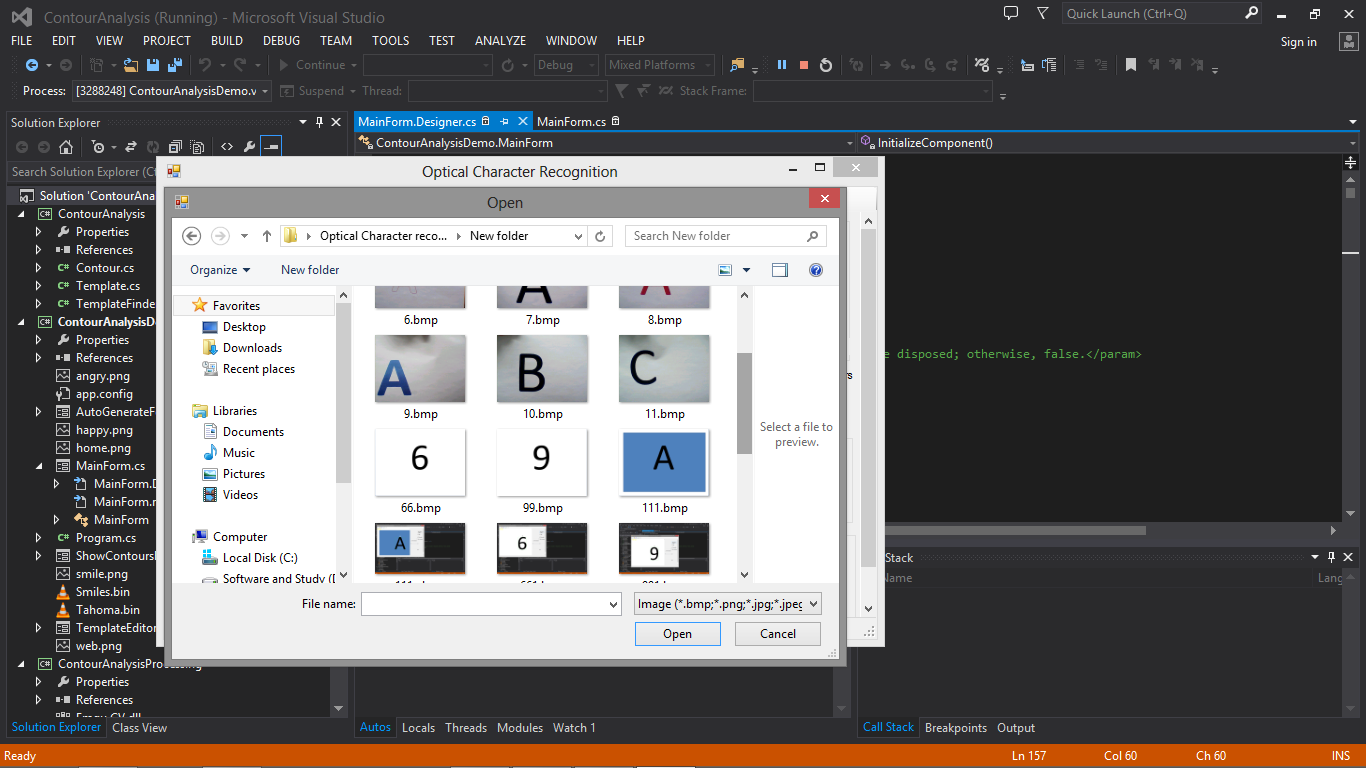
In this step you need to install the hardware specific libraries and drivers that APPLY to your particular hardware so that your development machine recognizes and Visual Studio are configured correctly for your hardware. Please consult with your hardware provider for detailed instructions.[8]

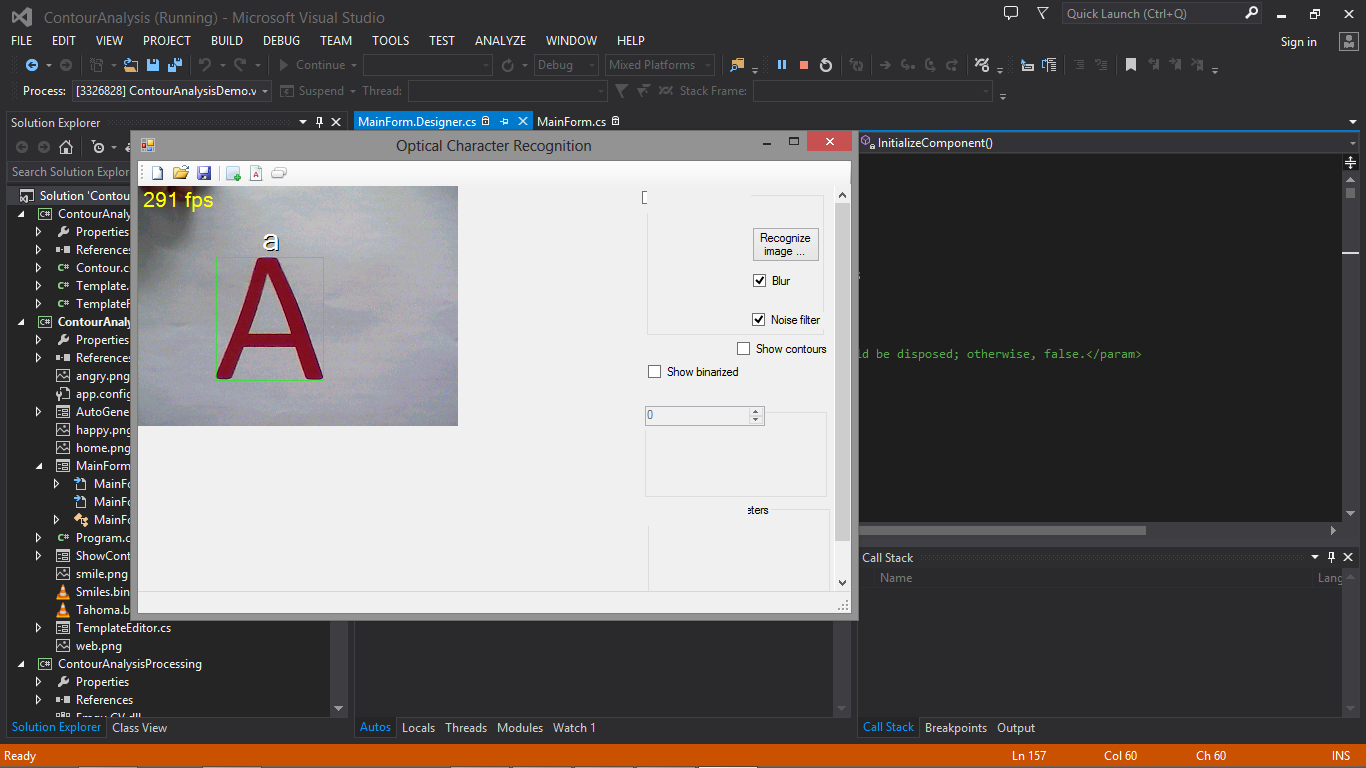
**Step 9**: Write your first application

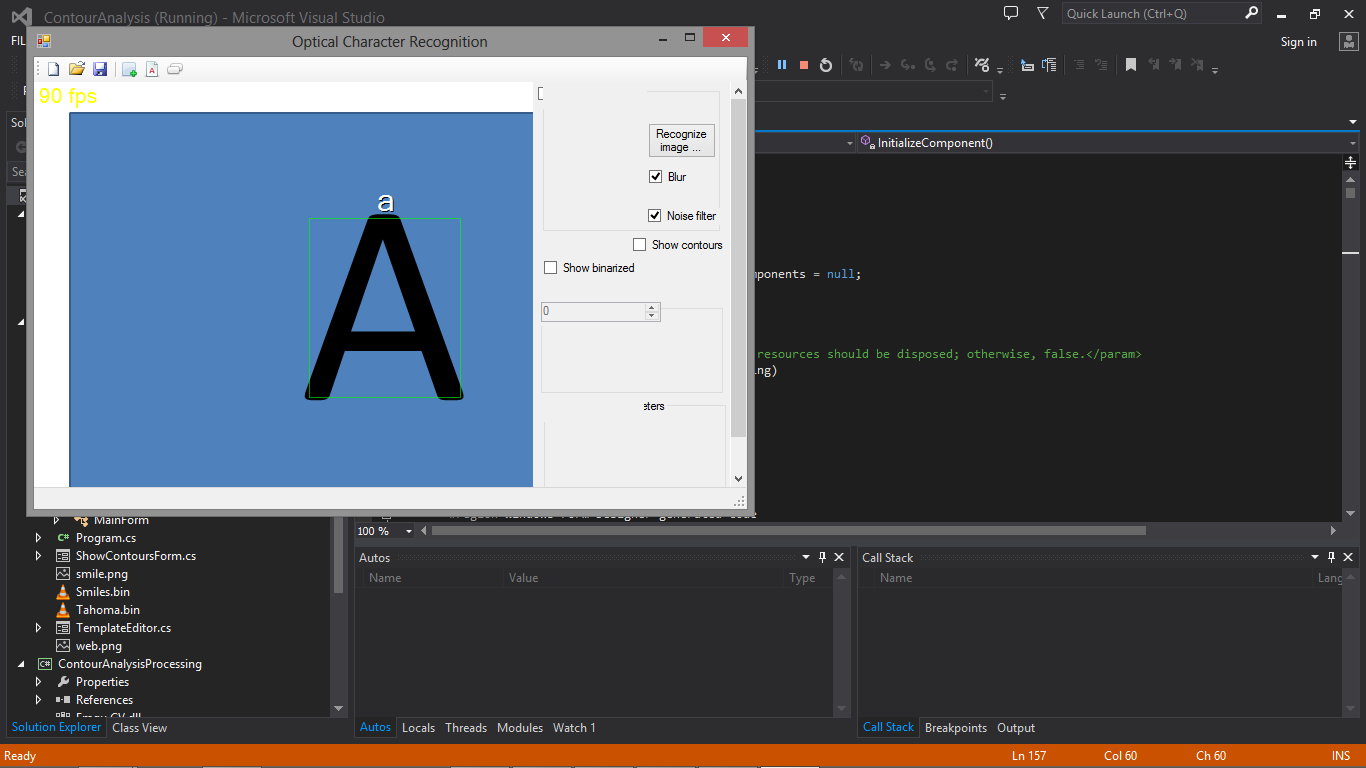
After installing your hardware specific software you can create a new project in Visual Studio, write an application, and deploy the application to your hardware.

**4 Project Analysis**

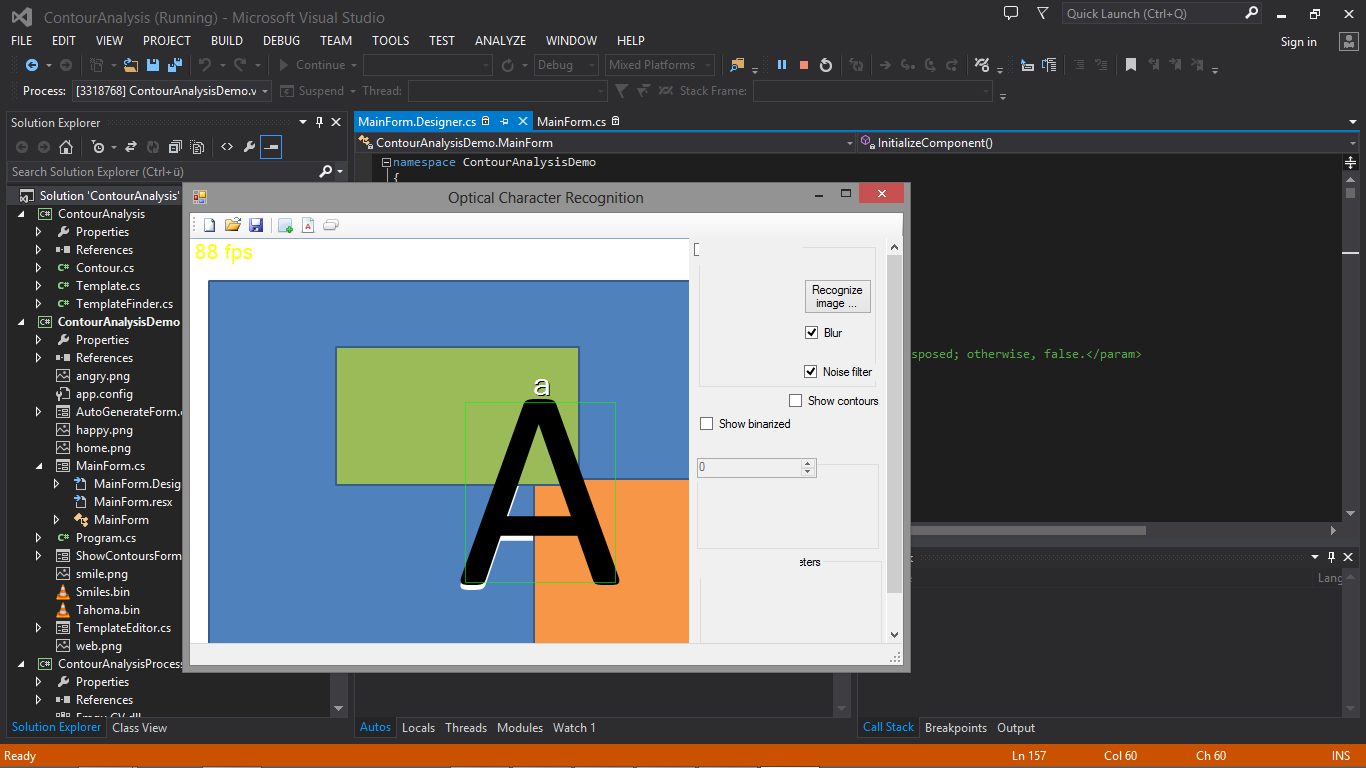
In this section I would like to show how the project will work. After all the setup, now we are ready to test our project.

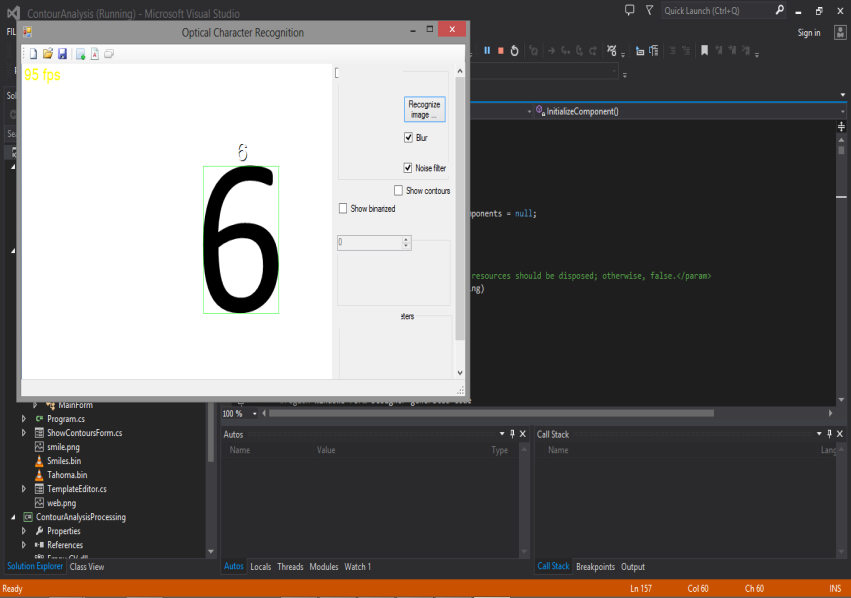
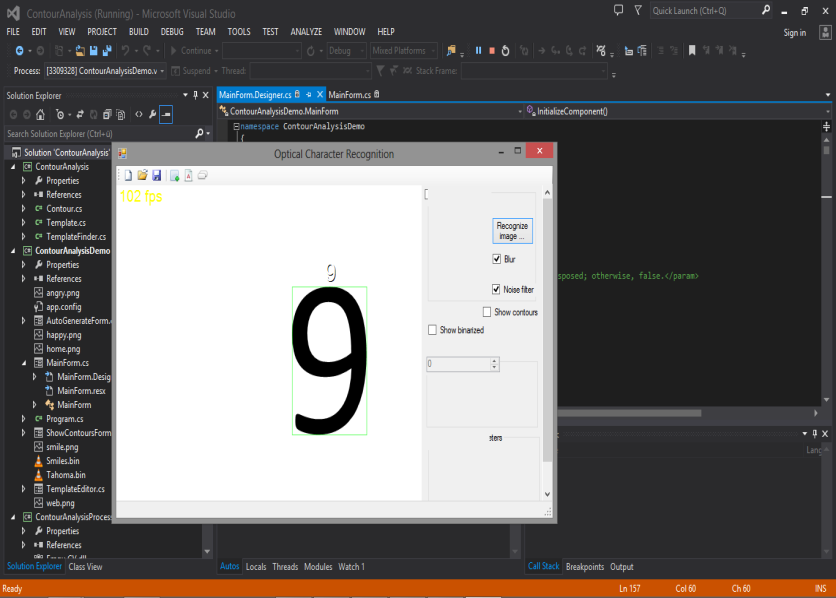
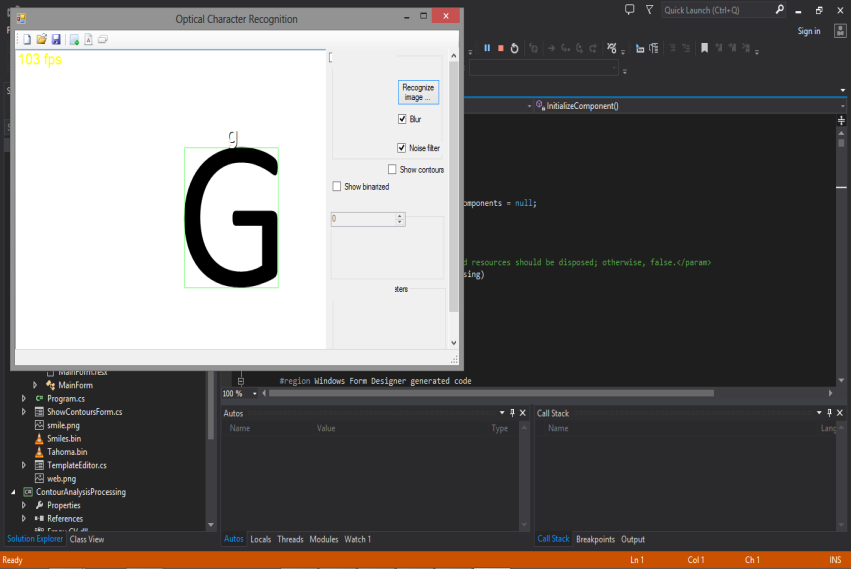
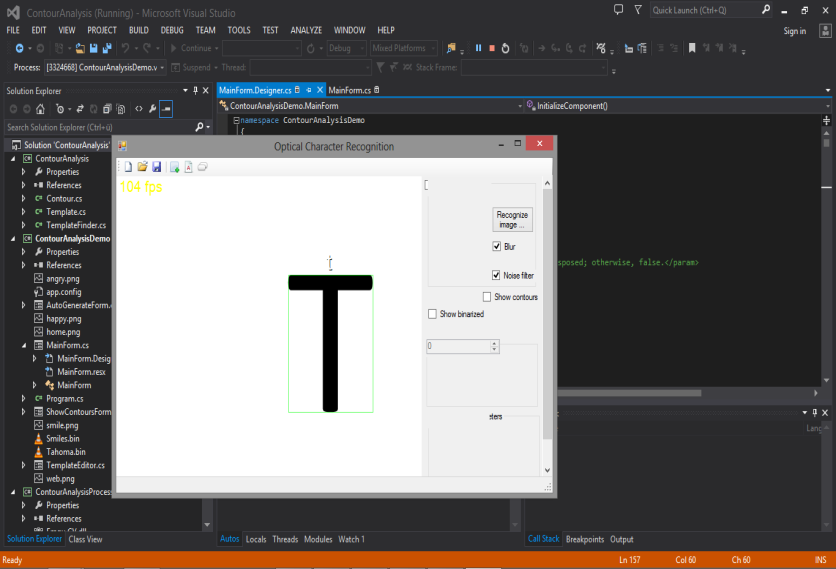
Now i will try to choose one alphabet or number which i have take through FEZ spider camera.

let's start with alphabet "A" without any kind of background.

As we can see, we are able to find our object "A". Now let's start again with "A" and with a color background.

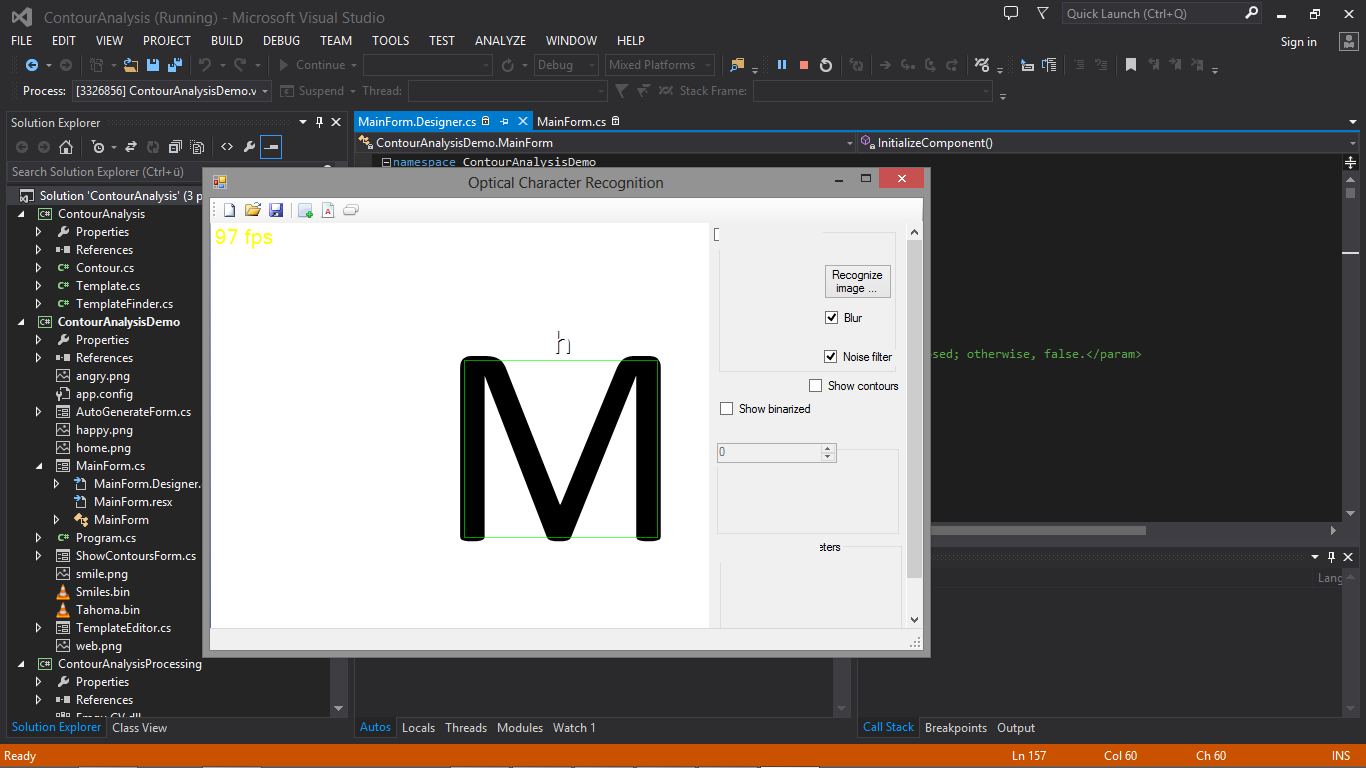
We can see this project can detect "A" with and without color background. Now let's try with multiple color background.

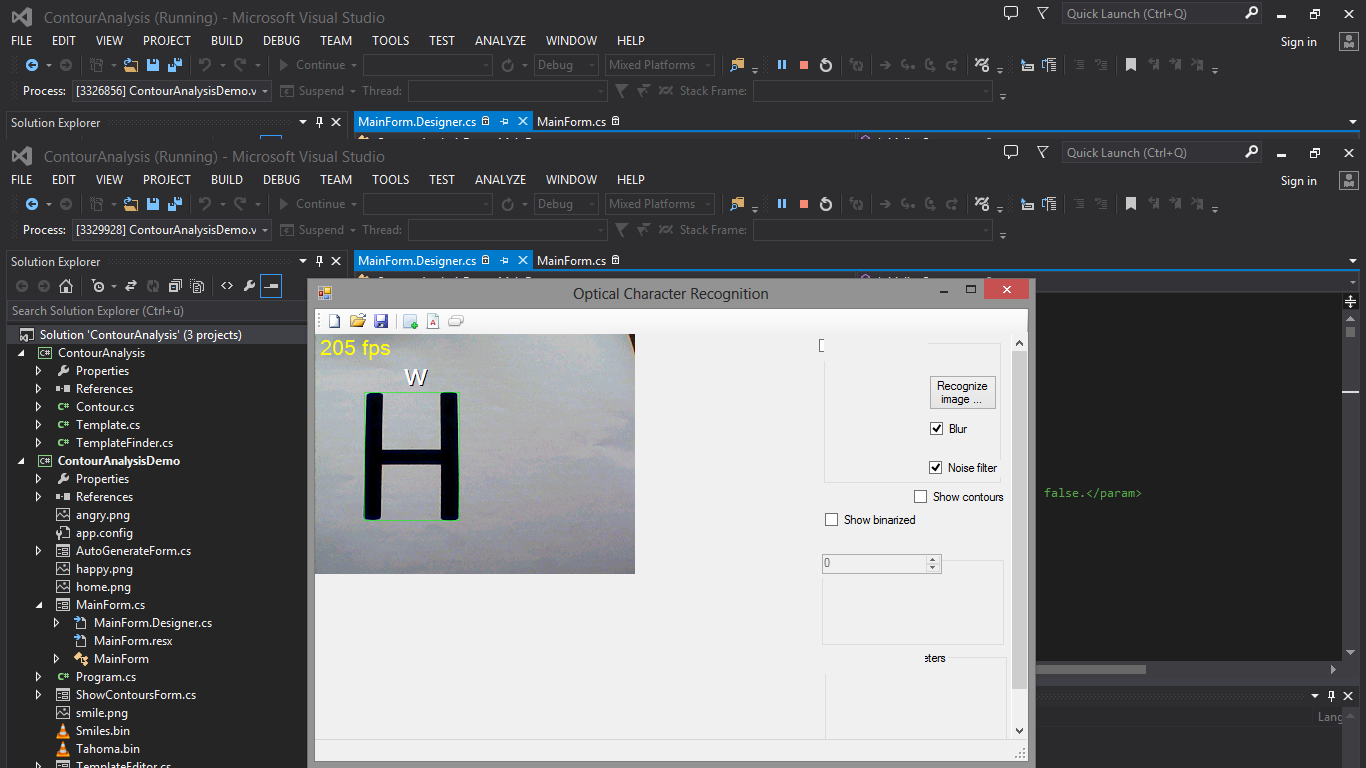
We can see it's working properly.

Now we will try to find some other alphabet and number together.

**5 Limitation**

As we can see throughout my report everything is going well. But I would like to share some limitation what I have faced during my project work. Till now what I have choose, my project can identify every single alphabet and number. But there are some alphabet which my project couldn't identify. Even its gave me some other output.



****

Hopefully as a future work I will be find out the solution very soon.

**Conclusion**

The task of the project was to develop an Optical Character Recognition program. The code for the whole program was developed in C# using .Net framework libraries. The process of taking images was accomplished with the FEZ Spider kit and image processing and character recognition was done on computer/laptop. The Optical Character Recognition program is developed in such a way that it is able to recognize both – grayscale and color images along with different sorts of backgrounds.

**References**

|  |  |
| --- | --- |
| [1] | Nepali Optical Character Recognition  https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&cad=rja&uact=8&sqi=2&ved=0CGEQFjAG&url=http%3A%2F%2Fwww.nr.no%2F~eikvil%2FOCR.pdf&ei=BxVRVcLOJJL5yQTitoHQDg&usg=AFQjCNEtX588wvAvOSH\_\_JMGKylawQgoNA&bvm=bv.92885102,d.bGg |
| [2] | http://www.netmf.com/gadgeteer/ |
| [3] | http://blogs.msdn.com/b/uk\_faculty\_connection/archive/2011/12/08/getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx |
| [4] | https://www.google.de/search?q=getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx&oq |
| [5] | http://blogs.msdn.com/b/uk\_faculty\_connection/archive/2011/12/08/getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx |
| [6] | getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx |
| [7] | http://blogs.msdn.com/b/uk\_faculty\_connection/archive/2011/12/08/getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx |
| [8] | http://www.netmf.com/get-started/ |

1. Nepali Optical Character Recognition- page 6 [↑](#footnote-ref-1)
2. Nepali Optical Character Recognition- page 7 [↑](#footnote-ref-2)
3. ### Getting Started with the FEZ Spider Kit for Microsoft .NET Gadgeteer

   [↑](#footnote-ref-3)
4. ### FEZ Spider Kit for Microsoft .NET Gadgeteer

   [↑](#footnote-ref-4)
5. http://blogs.msdn.com/b/uk\_faculty\_connection/archive/2011/12/08/getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx [↑](#footnote-ref-5)
6. getting-started-with-the-fez-spider-kit-for-microsoft-net-gadgeteer.aspx [↑](#footnote-ref-6)
7. www.netmicroframework.com [↑](#footnote-ref-7)