# How to build and use

* Clone the source code, open it in Visual Studio. In the Solution Explorer, there are 2 projects, ExcelAddIn and ExcelFunctions:

Graphical user interface, text, application

Description automatically generated

* Right click to one project and choose Build to build and run it:

A screenshot of a computer screen

Description automatically generated

* See the result after building the project, pay attention the path of the result file:

Text

Description automatically generated

Text

Description automatically generated

* To run ExcelAddIn project, open one Excel workbook, on the ribbon bar, choose File -> Options, the Excel Options window will appear, in this window, choose Add-ins -> Manage, choose COM Add-ins, and Go:

Graphical user interface, text, application, email

Description automatically generated

* Add the Dynamic Link Library (.dll file) to the COM Add-ins windows, the path of this file is the build result of the ExcelAddIn project:

Graphical user interface, text, application, email

Description automatically generated

* Click OK, on the current workbook, the new tab Add-ins will appear on the ribbon bar(if not, you may have to restart the Excel workbook):

Application, table

Description automatically generated

* To run ExcelFunctions project, on the Manage section of the Excel Options window, choose Excel Add-ins and Go:

Graphical user interface, text, application, email

Description automatically generated

* Click Browse button on the new window Add-ins to lead to the path of the build result file (ExcelFunction-AddIn64-packed.xll) of the ExcelFunctions project:

Graphical user interface, text, application

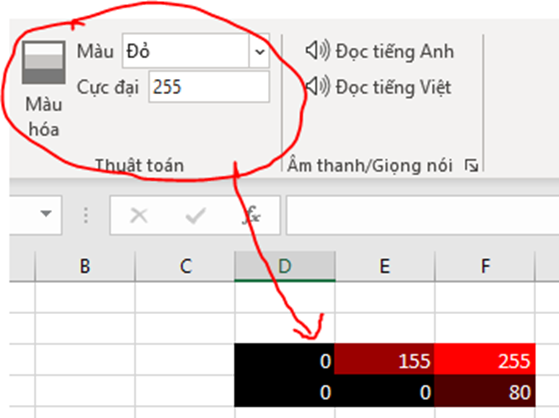
Description automatically generated

* Click OK. To test, choose one cell on the workbook, on the Formula Bar, input =QRCode(<option>,”<shape>”,”<string to QRCode>”), for example: =QRCode(1,"shape1","testString").

# Main progress

## Cells colorization

* About (03/10/2020):
* User chooses a range of cells, with selected color (red, green, blue) and saturation.
* User clicks on “Colorize”.
* The chosen cells will be colorized with corresponding color and saturation.



* Update 04/10/2020: Add gray to color, change font color based on cell color, if sat >128 -> black font, if sat<=128 -> white font.
* Detail:
* In group Algorithm (Thuật toán) on Add-ins tab, there are a button Colorize (Màu hóa), drop down ColorRGB (Màu), edit box editSaturationPeak (Cực đại) as the figure:

Graphical user interface, application

Description automatically generated

* In MyRibon.Designer.cs, I used click event handler to call the method buttonColorize\_Click in MyRibon.cs with arguments color and saturation:

this.buttonColorize.Click += (sender,e) => this.buttonColorize\_Click(dropDownColorRGB.SelectedItem.OfficeImageId,editSaturationPeak.Text);

* In method buttonColorize\_Click in MyRibon.cs, I got selected cells:

//get selected cells

Range currentRange = (Range)Globals.ThisAddIn.Application.Selection as

Microsoft.Office.Interop.Excel.Range;

if (currentRange == null) return;

* I checked if the saturation value is valid:

int saturationInt; //saturation value in integer

bool isNumeric = int.TryParse(saturation,out saturationInt); //boolean to check if saturation is a valid number

if (isNumeric) //check if the saturation is a number

if (saturationInt >= 0 && saturationInt <= 255) //check if the saturation value is in range 0 to 255

* If valid, I checked the color to colorize and assigned the saturation value to the cells:

switch (color) // color to display with saturation

{

case "AppointmentColor1": currentRange.Interior.Color = Color.FromArgb(saturationInt, 0, 0); break; //red

case "AppointmentColor2": currentRange.Interior.Color = Color.FromArgb(0, 0, saturationInt); break; //green

case "AppointmentColor3": currentRange.Interior.Color = Color.FromArgb(0, saturationInt, 0); break; //blue

default: break;

}

currentRange.Value = saturationInt; // cell value = saturation value

Example:

Graphical user interface, application, table, Excel

Description automatically generated

* If invalid, the error messages will be displayed:

//saturation is not in range 0 to 255

MessageBox.Show("Please input a number between 0 and 255 at saturation box", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

//saturation is not a valid number

MessageBox.Show("Invalid type, please input a number at saturation box", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

Example:

Table

Description automatically generated

Table

Description automatically generated

* Update 04/10/2020: done.

## QRCode generator function

* About (09/10/2020):

Using ZXing library (<https://github.com/zxing/zxing>) (instead of Google API) to generate QRCode image by input text.

* Details:
* ZXing library can be installed on Visual Studio as NuGet package:

On Solution Explorer, right click on one project, choose “Manage NuGet Packages…”, browse and install “ZXing.NET”:

A screenshot of a computer screen

Description automatically generated

Graphical user interface, text

Description automatically generated

* I created an excel function QRCodeZ (Z stands for ZXing), with a string Text argument:

//Generate QRCode using ZXing library

[ExcelDna.Integration.ExcelFunction(Description = "QRCode generator by ZXing lib")]

public static object QRCodeZ(

[ExcelDna.Integration.ExcelArgument(Description = "Text to be transformed to QRCode. Example: \"hello\"")]

string Text)

* I used ZXing library to read the Text string, transform it to QRCode image, save it to a file, then load it to the current excel file.
* Update (12/10/2020): not necessary to save the QRCode image to a file, just save it to clipboard and paste to the worksheet.
* Update (23/10/2020): Add a parameter int option to the function QRCode, with option 1 to use ZXing library to generate QRCode image, option 2 to use Google API. The ZXing option uses Path.GetTempFileName() to save the QRCode image to local storage.