.NET Gadgeteer Graphics 2.

Distributed Embedded Systems Lab

Daniel Stojcsics, PhD
stojcsics.daniel@nik.uni-obuda.hu

2015 autumn

The application - Basic image processing

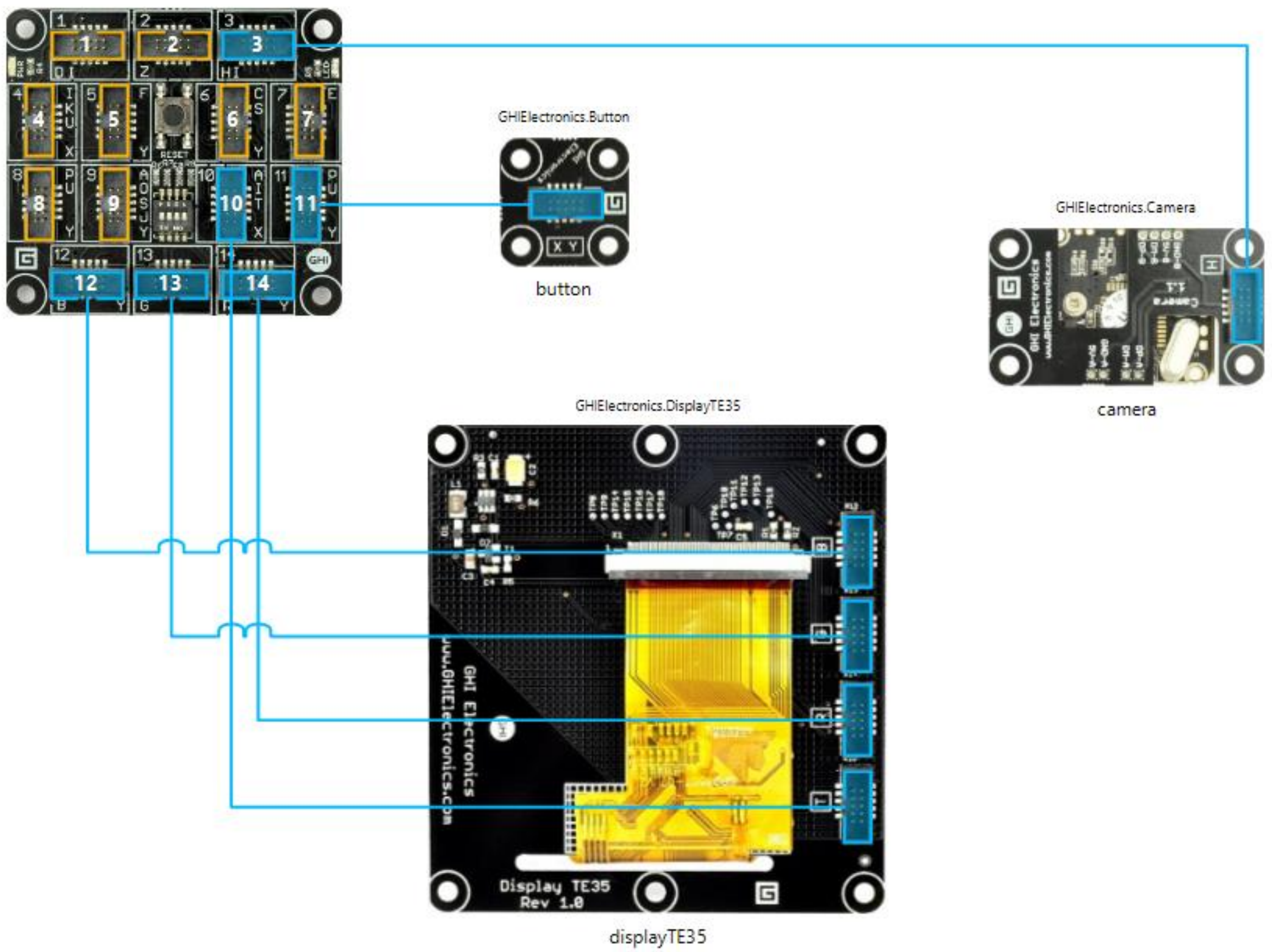
Modules:

- USBClient DP
- FEZ Spider
- Button
- Camera
- DisplayTE35

Application:

- Take a picture with a „tuch” on the display
- Rescale the picture for faster image procession
- Create greyscale, black&white, inverted images

2. Hardware configuration



3. The code: Tuch event – taking a picture (1)

```
private Bitmap buffer;
private GTM.Module.DisplayModule.Window mainWindow;
private bool isStreaming = false;

void ProgramStarted()
{
   mainWindow = display.WPFWindow;
   buffer = new Bitmap(display.Width, display.Height);
   camera.CameraConnected += camera_CameraConnected;
   camera.BitmapStreamed += camera_BitmapStreamed;
}

void camera_BitmapStreamed(Camera sender, Bitmap e)
{
   display.SimpleGraphics.DisplayImage(e, 0, 0);
}

void camera_CameraConnected(Camera sender, EventArgs e)
{
   mainWindow.TouchUp += mainWindow_TouchUp;
   camera.StartStreaming(buffer);
   isStreaming = true;
}

void mainWindow_TouchUp(object sender, Microsoft.SPOT.Input.TouchEventArgs e)
{
   if (isStreaming) camera.StopStreaming();
   else camera.StartStreaming(buffer);

   isStreaming = !isStreaming;
}
```

Tuch event – taking a picture (2)

- `Buffer:` Bitmap object to store the image
- `Window` object: to detect tuch event
- `TuchUp & isStreaming:` to handle states (taking or processing the bitmap)

Adding more bitmap objects (1)

- Complete the existing code with the yellow lines:

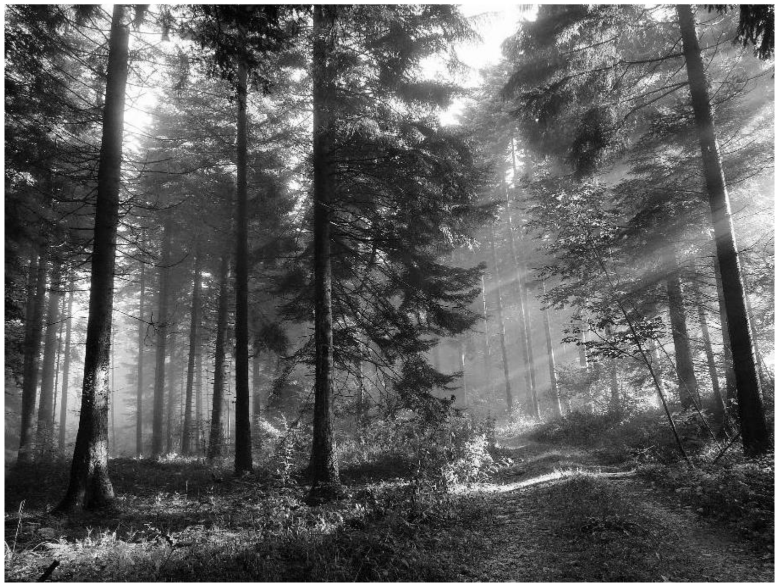
```
private const int SCALE_VALUE = 3;
private Bitmap scaledBitmap;
private Bitmap grayScale;
private Bitmap blackAndWhite;
private Bitmap inverted;

void ProgramStarted()
{
   mainWindow = display.WPFWindow;
   buffer = new Bitmap(display.Width, display.Height);
   camera.CameraConnected += camera_CameraConnected;
   camera.BitmapStreamed += camera_BitmapStreamed;

   buffer = new Bitmap(display.Width, display.Height);
   scaledBitmap = new Bitmap(buffer.Width / SCALE_VALUE, buffer.Height / SCALE_VALUE);
   grayScale = new Bitmap(scaledBitmap.Width, scaledBitmap.Height);
   blackAndWhite = new Bitmap(scaledBitmap.Width, scaledBitmap.Height);
   inverted = new Bitmap(scaledBitmap.Width, scaledBitmap.Height);
}
```

Adding more bitmap objects (2)

- FEZ Spider has a slow core. To speed up the image processing, the bitmaps has to be scaled down with a `SCALE_VALUE`
- The bitmap objects for the manimpulated images:
  - `scaledBitmap`: scaled version of the original
  - `grayScale`
  - `blackAndWhite`
  - `inverted`

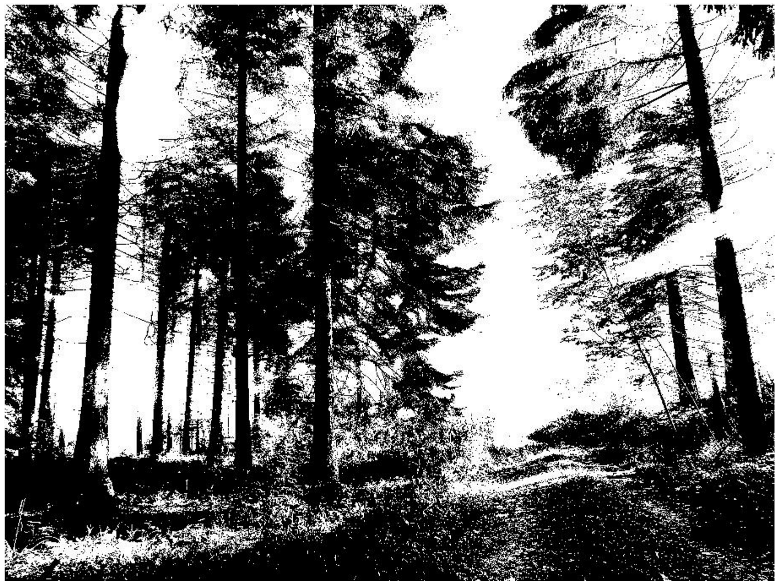
Greyscale image

- (Weighted) average of R+G+B pixels *(note: R/G/B values = [0-255])*



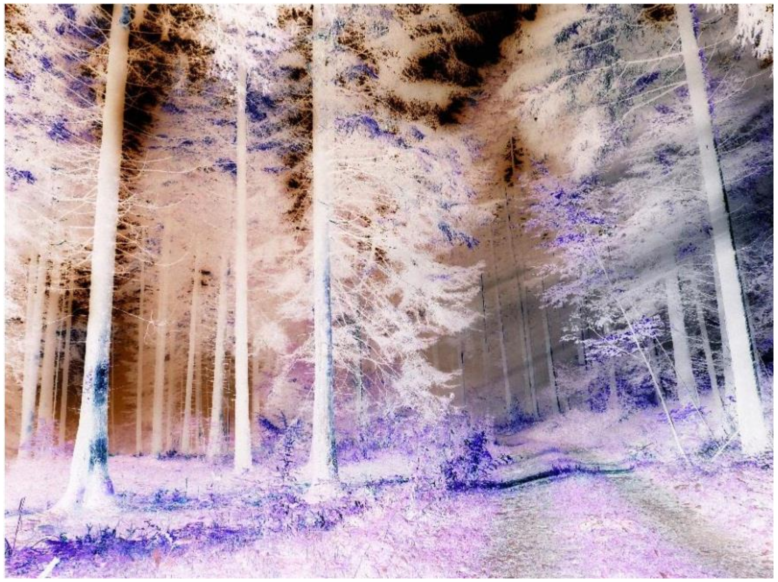
Black & white

- Only black or white pixels. A treshold value has to be declared (e.g. 128). Every pixel over this value will be white (255), others black (0).



Inverted

- Each R/G/B value = 255 - R/G/B value



Create the following method:

```
void ProcessImages()
{
   scaledBitmap.StretchImage(0, 0, scaledBitmap.Width, scaledBitmap.Height, buffer, 0, 0,
     buffer.Width, buffer.Height, 255);

   int width = scaledBitmap.Width;
   int height = scaledBitmap.Height;
   for (int x = 0; x < width; x++)
   {
      for (int y = 0; y < height; y++)
      {
         GT.Color pixel = scaledBitmap.GetPixel(x, y);
         //gray
         byte value = (byte)((pixel.R + pixel.G + pixel.B) / 3);
         grayScale.SetPixel(x, y, GT.Color.FromRGB(value, value, value));
         //black&white
         byte value2 = (value > 128) ? (byte)255 : (byte)0;
         blackAndWhite.SetPixel(x, y, GT.Color.FromRGB(value2, value2, value2));
         //inverted
         byte r = (byte)(255 - pixel.R);
         byte g = (byte)(255 - pixel.G);
         byte b = (byte)(255 - pixel.B);
         inverted.SetPixel(x, y, GT.Color.FromRGB(r, g, b));
       }
    }
}
```

Modify the TouchUp metod:

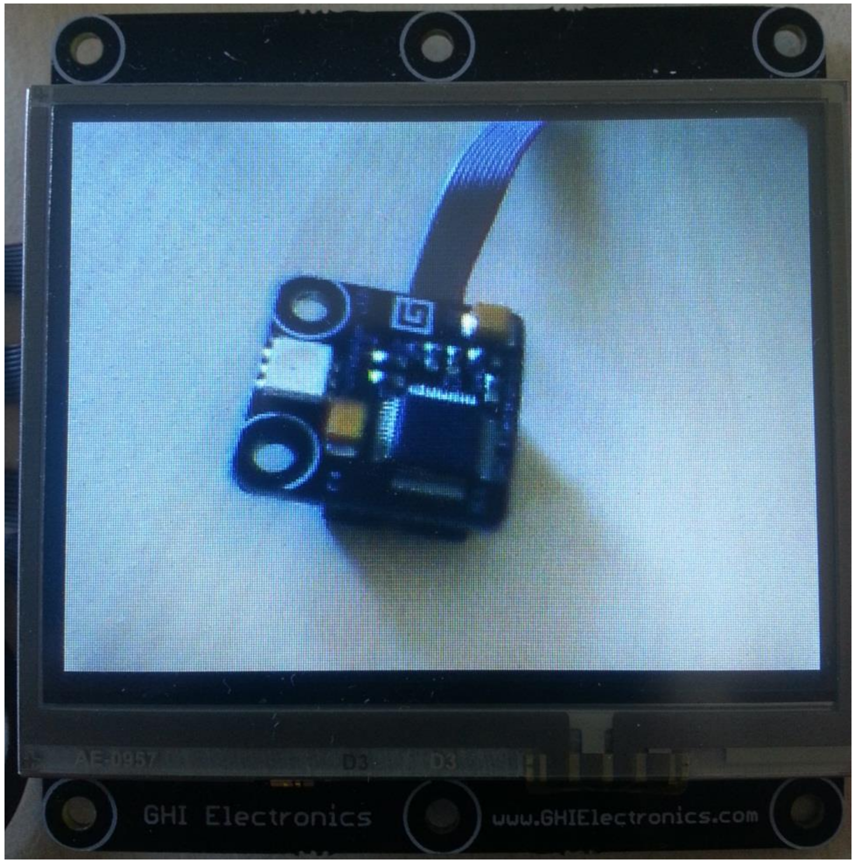
```
private enum Modes { NORMAL, GRAYSCALE, BLACKWHITE, INVERT };
private int currentMode = 0;

void mainWindow_TouchUp(object sender, Microsoft.SPOT.Input.TouchEventArgs e)
{
   if (isStreaming)
   {
      camera.StopStreaming();
      button.TurnLedOn();
      ProcessImages();
      button.TurnLedOff();
   }
   else
   {
      currentMode = 0;
      camera.StartStreaming(buffer);
   }

   isStreaming = !isStreaming;
}
```

ButtonPressed method:

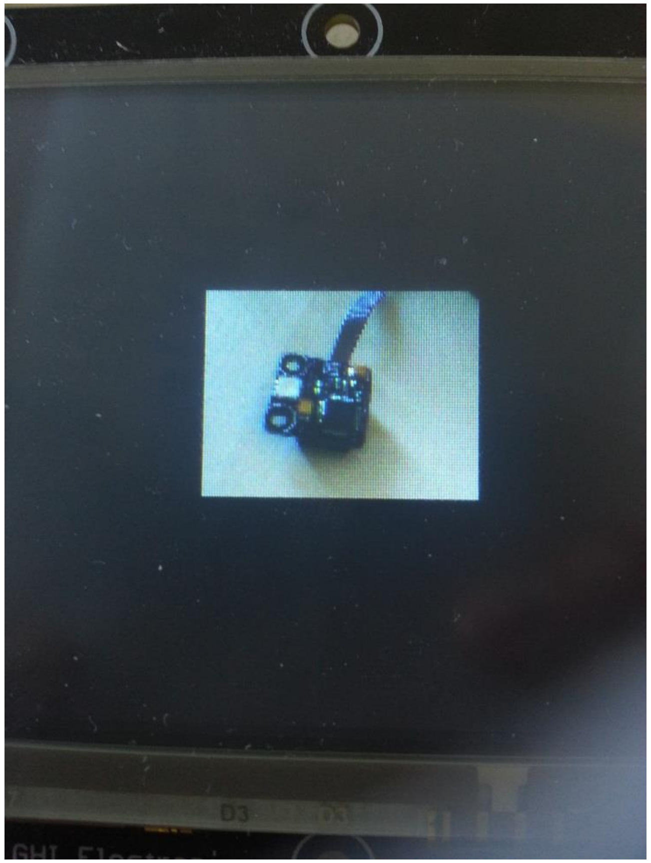
- Create the *button.ButtonPressed* method in *ProgramStarted()*!

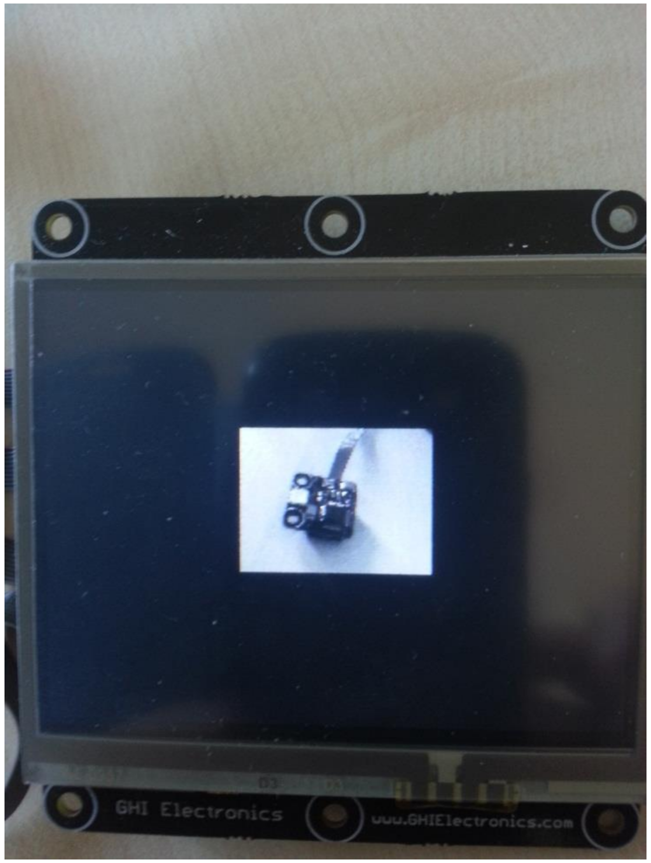
```
void button_ButtonPressed(Button sender, Button.ButtonState state)
{
   if (!isStreaming)
   {
      currentMode++;
      currentMode %= 4;
      Modes current = (Modes)currentMode;
      Bitmap selected = scaledBitmap;
      switch (current)
      {
         case Modes.NORMAL:
            selected = scaledBitmap;
            break;
         case Modes.GRAYSCALE:
            selected = grayScale;
            break;
         case Modes.BLACKWHITE:
            selected = blackAndWhite;
            break;
         case Modes.INVERT:
            selected = inverted;
            break;
      }

      int w = display.Width / 2;
      int h = display.Height / 2;
      display.SimpleGraphics.Clear();
      display.SimpleGraphics.DisplayImage(selected, w - selected.Width / 2, h - selected.Height / 2);
   }
}
```

Initial picture



Initial and greyscale image





Black & white and inverted image