.NET Gadgeteer Intro

Distributed Embedded Systems Lab

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

2015 autumn

Lecture 2.

Camera and Display basics

- T35 Display

- USB camera

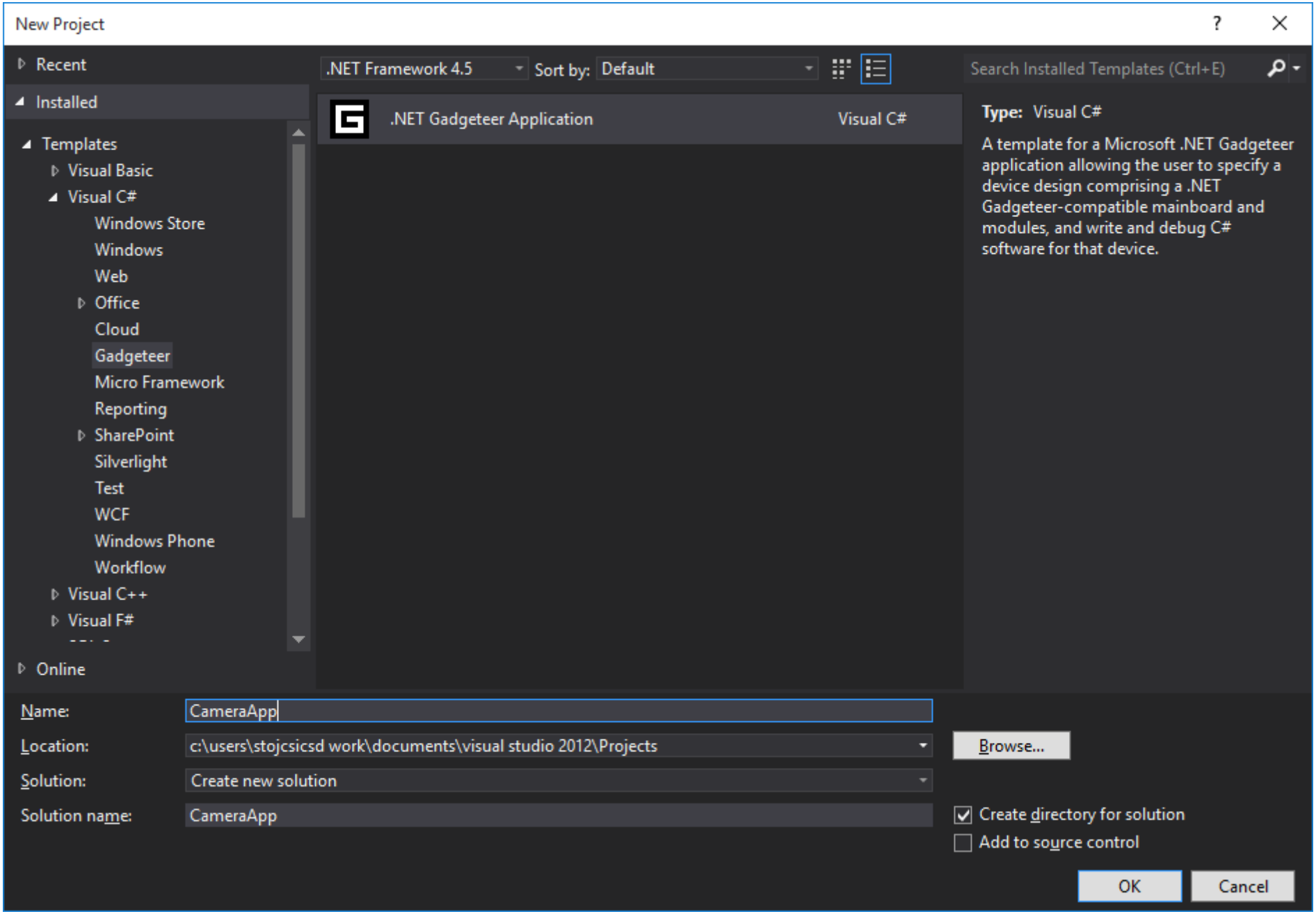
Mainboard is only @ 72MHz!! ☺

T35 Display

- 320x240px RGB
- Tuch available
- R,G,B,T sockets
- T must be connected even if tuch is not used! (otherwise we get null reference exception)

Display_T35 Class

- Width
- Height
- SimpleGrapics
  - Text
  - Shapes
  - Images


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows Store
Windows
Web
Office
Cloud
Gadgeteer
Micro Framework
Reporting
SharePoint
Silverlight
Test
WCF
Windows Phone
Workflow
Visual C++
Visual F#
Online
.NET Framework 4.5
Sort by:
Default
Search Installed Templates (Ctrl+E)
.NET Gadgeteer Application
Visual C#
Type: Visual C#
A template for a Microsoft .NET Gadgeteer application allowing the user to specify a device design comprising a .NET Gadgeteer-compatible mainboard and modules, and write and debug C# software for that device.
Name:
CameraApp
Location:
c:\users\stojcsicsd work\documents\visual studio 2012\Projects
Browse...
Solution:
Create new solution
Solution name:
CameraApp
Create directory for solution
Add to source control
OK
Cancel

.NET Gadgeteer Application Wizard

Choose a mainboard:

FEZ Cerberus

FEZ Cerbuino Bee

FEZ Cerbuino Net

FEZ Cobra II Eco

FEZ Hydra

FEZ Raptor

FEZ Spider

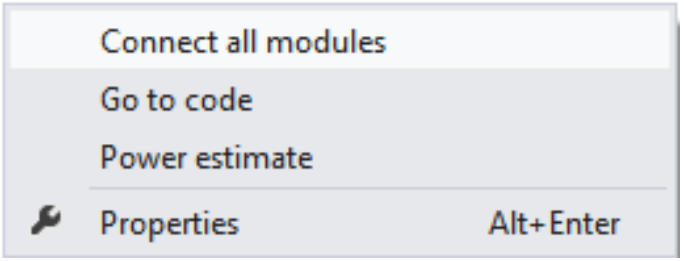
G400HDR Breakout

Choose from .NET Micro Framework versions supported by this mainboard: 4.3

Create Cancel

Autoconnect the modules

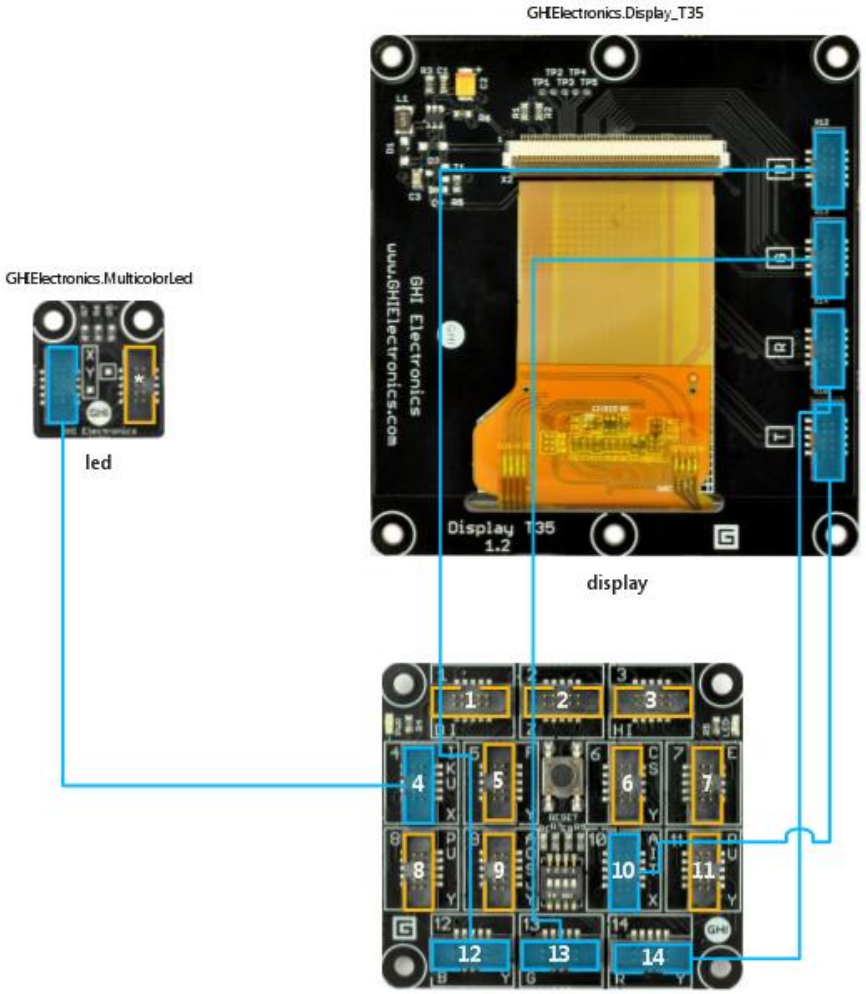
- Right click, then „connect all modules”



- Connect the modules phisically exacly the same as in the graphic designer!

Simple Application

- Create a random color in every second second (2000ms)!
- Switch on the multicolor led with that color
- Draw a full screen rectangle to the display with that color (simple shape)



```
public void DisplayRectangle
  (Color outlineColor,
  uint thicknessOutline,
  Color fillColor,
  uint x, uint y,
  uint width, uint height);
```

```
// Summary:
//     Displays a rectangle.
//
// Parameters:
//   outlineColor:
//     The color for the outline of the rectangle.
//
//   thicknessOutline:
//     The thickness of the outline.
//
//   fillColor:
//     The color to fill the rectangle with.
//
//   x:
//     The X coordinate for the top left corner of the rectangle.
//
//   y:
//     The Y coordinate for the top left corner of the rectangle.
//
//   width:
//     The width of the rectangle.
//
//   height:
//     The height of the rectangle.
```

```
public partial class Program
    {
         private GT.Timer timer = new GT.Timer(2000);

        private byte r, g, b = 0;


        void ProgramStarted()
        {
            timer.Tick += timer_Tick;
            timer.Start();
            multicolorLED.GreenBlueSwapped = true; // ONLY IF NECESSARY!
        }

        void timer_Tick(GT.Timer timer)
        {
            Random rand = new Random();
            r = (byte)rand.Next(255);
            g = (byte)rand.Next(255);
            b = (byte)rand.Next(255);

            GT.Color c = new GT.Color();
            c = GT.Color.FromRGB(r, g, b);
            multicolorLED.SetRedIntensity(r);
            multicolorLED.SetGreenIntensity(g);
            multicolorLED.SetBlueIntensity(b);
            displayTE35.SimpleGraphics.DisplayRectangle(c, 0, c, 0, 0,
                                                        displayTE35.Width, displayTE35.Height);
        }
    }
```

void SimpleGraphicsInterface.DisplayRectangle(**GT.Color outlineColor**, uint thicknessOutline, GT.Color fillColor, uint x, uint y, uint width, uint height)

```
display.SimpleGraphics.DisplayText
   ("Some text",
   Resources.GetFont(Resources.FontResources.NinaB),
   GT.Color.White, 0, 0);
// Parameters:
//   text:
//     The text to display.
//
//   font:
//     The font to use for the text display.
//
//   color:
//     The color of the text.
//
//   x:
//     The X cooridinate to begin the text display.
//
//   y:
//     The Y cooridinate to begin the text display.


display.SimpleGraphics.ClearNoRedraw();
//     Clears the display, but does not redraw it.
```

Display Date & Time

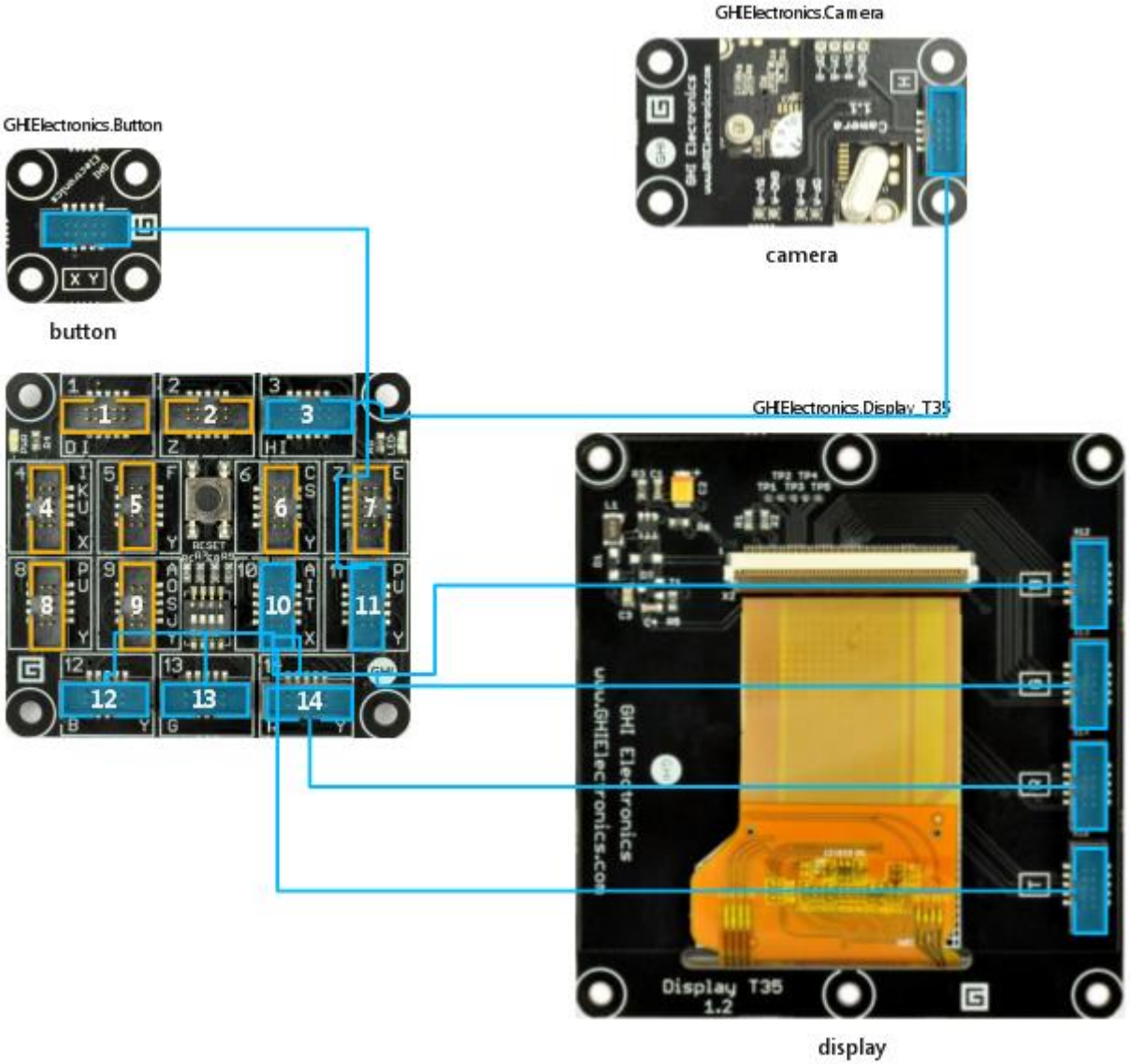
- `public struct DateTime`
  - `public DateTime(int year, int month, int day, int hour, int minute, int second);`

```
DateTime time =
      new DateTime(2014, 11, 28, 11, 00, 00);
// lost after power cycle or reset:
Microsoft.SPOT.Hardware.Utility.SetLocalTime(time);
```

- Display the Date&Time on every new color change! ☺
  - `DateTime.Now.ToString()`

```
public partial class Program
{
    private GT.Timer timer = new GT.Timer(2000);
    private byte r, g, b = 0;
    DateTime time = new DateTime(2015, 10, 14, 8, 30, 00);

    void ProgramStarted()
    {
        timer.Tick += timer_Tick;
        timer.Start();
        Microsoft.SPOT.Hardware.Utility.SetLocalTime(time);
        //multicolorLED.GreenBlueSwapped = true;
    }

    void timer_Tick(GT.Timer timer)
    {
        Random rand = new Random();
        r = (byte)rand.Next(255);
        g = (byte)rand.Next(255);
        b = (byte)rand.Next(255);

        GT.Color c = new GT.Color();
        c = GT.Color.FromRGB(r, g, b);
        multicolorLED.SetRedIntensity(r);
        multicolorLED.SetGreenIntensity(g);
        multicolorLED.SetBlueIntensity(b);
        displayTE35.SimpleGraphics.DisplayRectangle(c, 0, c, 0, 0,
                displayTE35.Width, displayTE35.Height);
        displayTE35.SimpleGraphics.DisplayText(DateTime.Now.ToString(),
                Resources.GetFont(Resources.FontResources.NinaB), GT.Color.White, 0, 10);
    }
}
```

Simple digital camera



To do:

- When program starts, display:

  `"Press button to take picture!"`
- On ButtonPressed event, take a picture!
- On PictureTaken event display the picture on the display!

- `public class SimpleGraphicsInterface`
  - `public void DisplayImage(Picture picture, uint x, uint y);`

Image Stream

```
public class Camera : Module
    public event Camera.BitmapStreamedEventHandler BitmapStreamed;
      // Summary:
      //     Event raised when the camera has completed streaming a
        bitmap.
    public void StartStreamingBitmaps(Bitmap bitmap);
         // Summary:
         // Starts streaming the bitmap identified by the bitmap
    parameter.
         //
         // Parameters:
         //   bitmap:
         //     Bitmap of the same dimensions as the
                 Gadgeteer.Modules.GHIElectronics.
                 Camera.CurrentPictureResolution
         //     property.

public sealed class Bitmap : MarshalByRefObject, IDisposable
    public Bitmap(int width, int height);
```

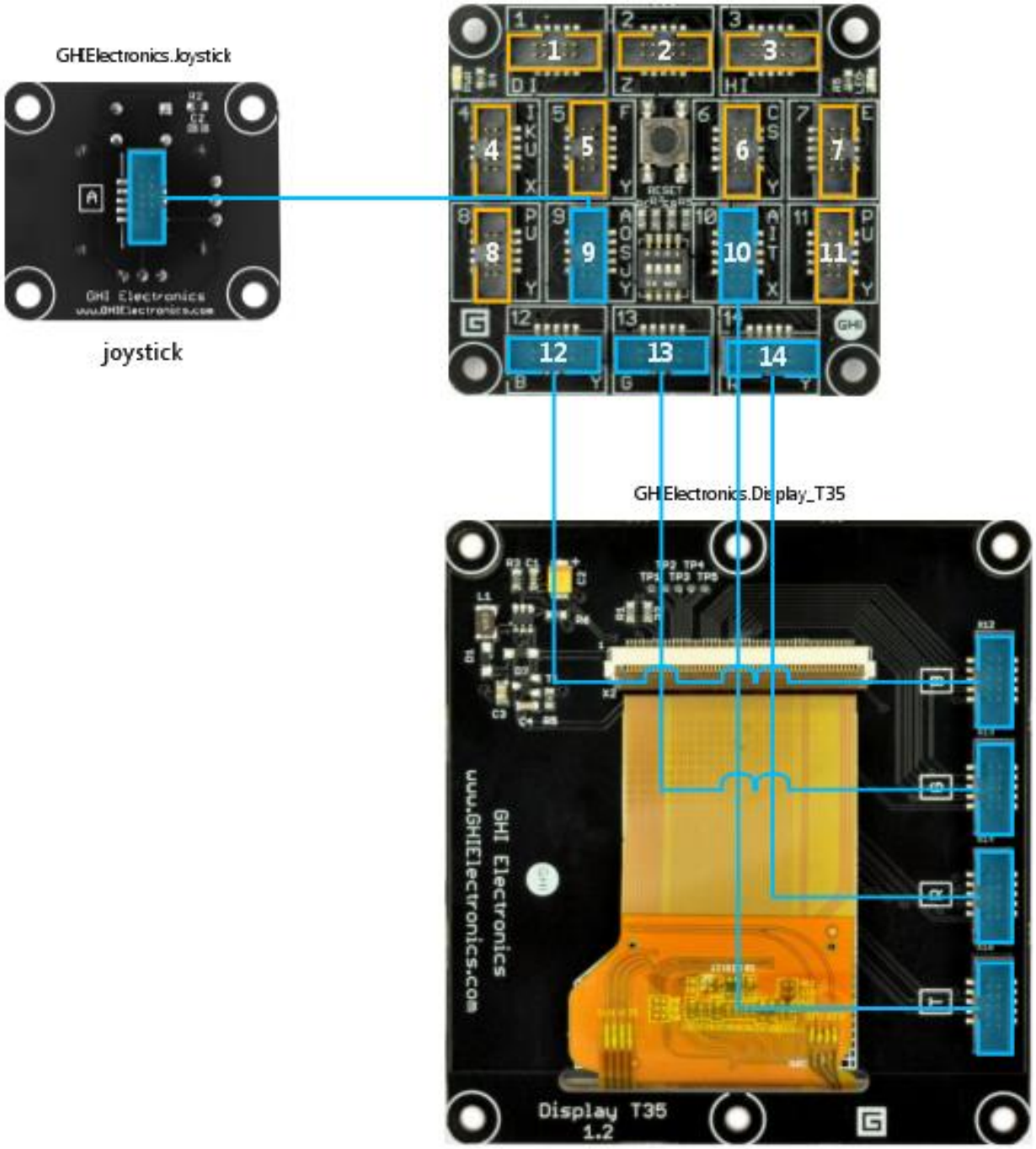
```
public partial class Program
  {
    private Bitmap _buffer;

    void ProgramStarted()
    {
      Debug.Print("Started");
      _buffer = new Bitmap(displayTE35.Width, displayTE35.Height);
      camera.BitmapStreamed += camera_BitmapStreamed;
      button.ButtonPressed +=button_ButtonPressed;
      displayTE35.SimpleGraphics.ClearNoRedraw();
      displayTE35.SimpleGraphics.DisplayText("Press Button to take picture!",
            Resources.GetFont(Resources.FontResources.NinaB), GT.Color.White, 0, 10);
    }

    void camera_BitmapStreamed(GTM.GHIElectronics.Camera sender, Bitmap e)
    {
      displayTE35.SimpleGraphics.DisplayImage(e, 0, 0);
      sender.StopStreaming();
    }

    void button_ButtonPressed(Button sender, Button.ButtonState state)
    {
      camera.StartStreaming(_buffer);
    }
  }
```

Lecture 3.

2015

Joystick

(ADC, basic signal filtering)


GHIElectronics.Joystick
GHI Electronics
www.GHIElectronics.com
joystick
GHIElectronics.Display_T35
GHI Electronics
www.GHIElectronics.com
Display T35
1.2
display

Joystick parameters:

- joystick.GetPostion().X
  - X coordinate of the Joystick (-1…1; 0 is center)
- joystick.GetPostion().Y
  - Y coordinate of the Joystick (-1…1; 0 is center)
- joystick.JoystickPressed EVENTHANDLER

- Create a timer (100ms)
- Draw a circle representing joystick position
  - Center: Width/2; Height/2
  - Radius: 10px

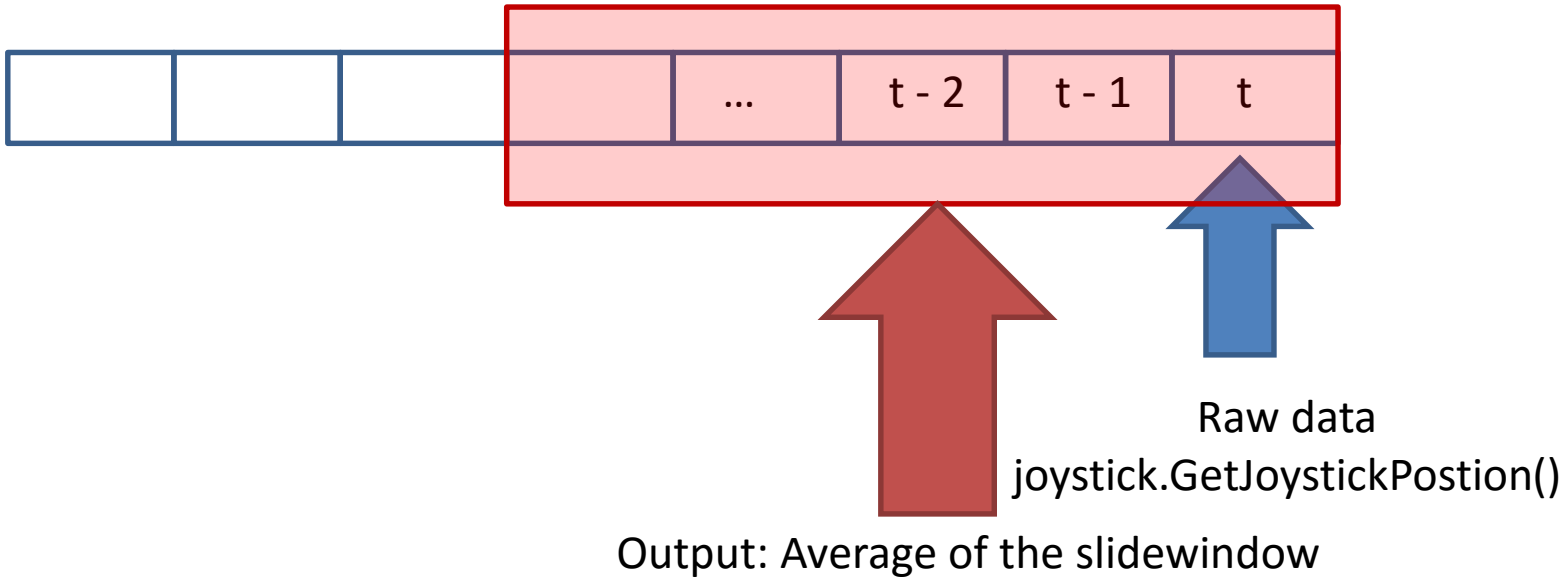
```
public void DisplayEllipse(Color outlineColor, int
thicknessOutline, Color fillColor, int x, int y, int xRadius,
int yRadius                 //   outlineColor:
          //     The color of the ellipse outline.
          //
          //   thicknessOutline:
          //     The thickness of the outline.
          //
          //   fillColor:
          //     The color of the ellipse fill.
          //
          //   x:
          //     The X coordinate of the center of the ellipse.
          //
          //   y:
          //     The Y coordinate of the center of the ellipse.
          //
          //   xRadius:
          //     The radius value for Y.
          //
          //   yRadius:
          //     The radius value for X.
```

- Write the X and Y pos to the screen

```
public partial class Program
  {

    int ballsize = 10;

    void ProgramStarted()
    {
      Debug.Print("Program Started");
      GT.Timer timer = new GT.Timer(100);
      timer.Tick += timer_Tick;
      timer.Start();

    }

    void timer_Tick(GT.Timer timer)
    {
      int xpos = displayTE35.Width/2 + (int)System.Math.Floor(joystick.GetPosition().X * displayTE35.Width/2);
      int ypos = displayTE35.Height/2 - (int)System.Math.Floor(joystick.GetPosition().Y * displayTE35.Height/2);
      displayTE35.SimpleGraphics.ClearNoRedraw();
      displayTE35.SimpleGraphics.DisplayText("xpos: " + xpos.ToString(),
          Resources.GetFont(Resources.FontResources.NinaB), GT.Color.White, 0, 10);
      displayTE35.SimpleGraphics.DisplayText("ypos: " + ypos.ToString(),
          Resources.GetFont(Resources.FontResources.NinaB), GT.Color.White, 0, 30);
      displayTE35.SimpleGraphics.DisplayText("Joy. X: " + joystick.GetPosition().X.ToString(),
          Resources.GetFont(Resources.FontResources.NinaB), GT.Color.White, 0, 50);
      displayTE35.SimpleGraphics.DisplayText("Joy. Y: " + joystick.GetPosition().Y.ToString(),
          Resources.GetFont(Resources.FontResources.NinaB), GT.Color.White, 0, 70);
      displayTE35.SimpleGraphics.DisplayEllipse(GT.Color.White,1,GT.Color.Black, xpos + ballsize / 2, ypos + ballsize / 2,
          ballsize, ballsize);
    }
  }
```

ADC noise

- Circle position ☹
- Create a slide window filter with 5 measurements
- Output is the average of the slide window
- Noise is always present!



```
public partial class Program
    {

        int ballsize = 10;
        const int filterlength = 5;
        double[] xfilter = new double[filterlength];
        double[] yfilter = new double[filterlength];
        Joystick.Position position;

        void ProgramStarted()
        {
            Debug.Print("Program Started");
            GT.Timer timer = new GT.Timer(100);
            timer.Tick += timer_Tick;
            timer.Start();

        }
...
```

```
 void timer_Tick(GT.Timer timer)
        {
            filter();
            int xpos = displayTE35.Width/2 +
                  (int)System.Math.Floor(position.X * displayTE35.Width/2);
            int ypos = displayTE35.Height/2 –
                  (int)System.Math.Floor(position.Y * displayTE35.Height/2);

            displayTE35.SimpleGraphics.ClearNoRedraw();
            displayTE35.SimpleGraphics.DisplayText("xpos: " +
xpos.ToString(), Resources.GetFont(Resources.FontResources.NinaB),
GT.Color.White, 0, 10);

            displayTE35.SimpleGraphics.DisplayText("ypos: " +
ypos.ToString(), Resources.GetFont(Resources.FontResources.NinaB),
GT.Color.White, 0, 30);

            displayTE35.SimpleGraphics.DisplayText("Joy. X: " +
position.X.ToString(), Resources.GetFont(Resources.FontResources.NinaB),
GT.Color.White, 0, 50);

            displayTE35.SimpleGraphics.DisplayText("Joy. Y: " +
position.Y.ToString(), Resources.GetFont(Resources.FontResources.NinaB),
GT.Color.White, 0, 70);

         displayTE35.SimpleGraphics.DisplayEllipse(GT.Color.White, 1,
GT.Color.Black, xpos + ballsize / 2, ypos + ballsize / 2, ballsize,
ballsize);
        }
...
```

```
void filter()
        {
            //Shifting data
            for (int i = 0; i < filterlength- 1; i++)
            {
                xfilter[i] = xfilter[i + 1];
                yfilter[i] = yfilter[i + 1];
            }
            xfilter[filterlength - 1] =
                        joystick.GetPosition().X;
            yfilter[filterlength - 1] =
                        joystick.GetPosition().Y;

            //Get filtered value
            for (int i = 0; i < filterlength - 1; i++)
            {
                position.X += xfilter[i];
                position.Y += yfilter[i];
            }

            position.X /= filterlength;
            position.Y /= filterlength;
        }
    }
```

When joystick button pressed, draw yellow dots ☺

Touch events

Event handlers:

- display.WPFWindow.TouchMove
- display.WPFWindow.TouchDown

- TuchEventArgs e:
  - Contains Touches X & Y coordinates
    - e.Touches[0].X;
    - e.Touches[0].Y;

- Can draw line to bitmap
  bitmap.DrawLine()

```
public partial class Program
  {
    Bitmap b;
    int prevX, prevY;

    void ProgramStarted()
    {
      Debug.Print("Program Started");
      displayTE35.WPFWindow.TouchMove += WPFWindow_TouchMove;
      displayTE35.WPFWindow.TouchDown += WPFWindow_TouchDown;
      b= new Bitmap(displayTE35.Width,displayTE35.Height);
    }
    void SetPrevTouch(Microsoft.SPOT.Input.TouchEventArgs e)
    {
      prevX = e.Touches[0].X;
      prevY = e.Touches[0].Y;
    }

    void WPFWindow_TouchDown(object sender, Microsoft.SPOT.Input.TouchEventArgs e)
    {
      b.Clear();
      SetPrevTouch(e);
    }

    void WPFWindow_TouchMove(object sender, Microsoft.SPOT.Input.TouchEventArgs e)
    {

      b.DrawLine(GT.Color.Yellow, 5, prevX, prevY, e.Touches[0].X, e.Touches[0].Y);
      displayTE35.SimpleGraphics.DisplayImage(b, 0, 0);
      SetPrevTouch(e);
    }
  }
```