Name : Welly
E-mail : wellytan09@gmail.com

Problem:

1. bufferedFrame is immediately dispose after event OnFrameUpdated is trigger means Frame object could be dispose while being used

   Solution:

```
public class FrameGrabber : IFrameCallback

{

    private byte[] _buffer;

    public delegate void FrameUpdateHandler( Frame rawFrame );

    public event FrameUpdateHandler OnFrameUpdated;

    public void FrameReceived( IntPtr frame, int width, int height )

    {

        if( _buffer == null )

        _buffer = new byte[width * height];

        // https://stackoverflow.com/questions/5486938/c-sharp-how-to-get-byte-from-intptr

        Marshal.Copy( frame, _buffer, 0, width * height );

        Frame bufferedFrame = new Frame( _buffer );
```

```
                OnFrameUpdated( bufferedFrame ); // Remove

                bufferedFrame.Dispose(); // Remove

                OnFrameUpdated?.Invoke(bufferedFrame); // ADD

        }

}


//Dispose after being used

private void OnTimerElapsed( object sender, ElapsedEventArgs e )

{

        If ( _receivedFrames.Count > 0 )

        {

                Frame frame = _receivedFrames.Dequeue();

                byte[] raw = frame.GetRawData();

                // https://stackoverflow.com/questions/29312223/finding-the-arithmetic-mean-of-an-array-c-sharp

                int sum = 0;

                for( int i = 0; i < raw.Length; i++ )

                sum += raw[i];

                int result = sum / raw.Length; // result now has the average of those numbers.
                _reporter.Report( result );
                frame.Dispose(); //ADD
            }
        }
```

2. Queue<Frame> _receivedFrames on **FrameCalculateAndStream** is access from one from the native library callback (which triggers HandleFrameUpdated) and the other from the timer's elapsed event (OnTimerElapsed). This can lead to race conditions and data corruption. The queue is not thread-safe.

   Solution: Use concurrentqueue instead, for thread-safe

   ```
   private ConcurrentQueue<Frame> _receivedFrames = new ConcurrentQueue<Frame>();
   ```

3. No Exceptions Handling
   Solution: add try-catch
   Ex:
   ```
   private void HandleFrameUpdated(Frame frame)
       {
         try
             {
               _receivedFrames.Enqueue(frame);
             }
         catch (Exception ex)
             {
               // Handle exceptions related to queue operations
               Console.WriteLine($"Error in HandleFrameUpdated: {ex.Message}");
   ```

```
            }
    }
```