

**CAMERA CAPTURE BASED ON AFORGE
PENGOLAHAN CITRA
MK401**

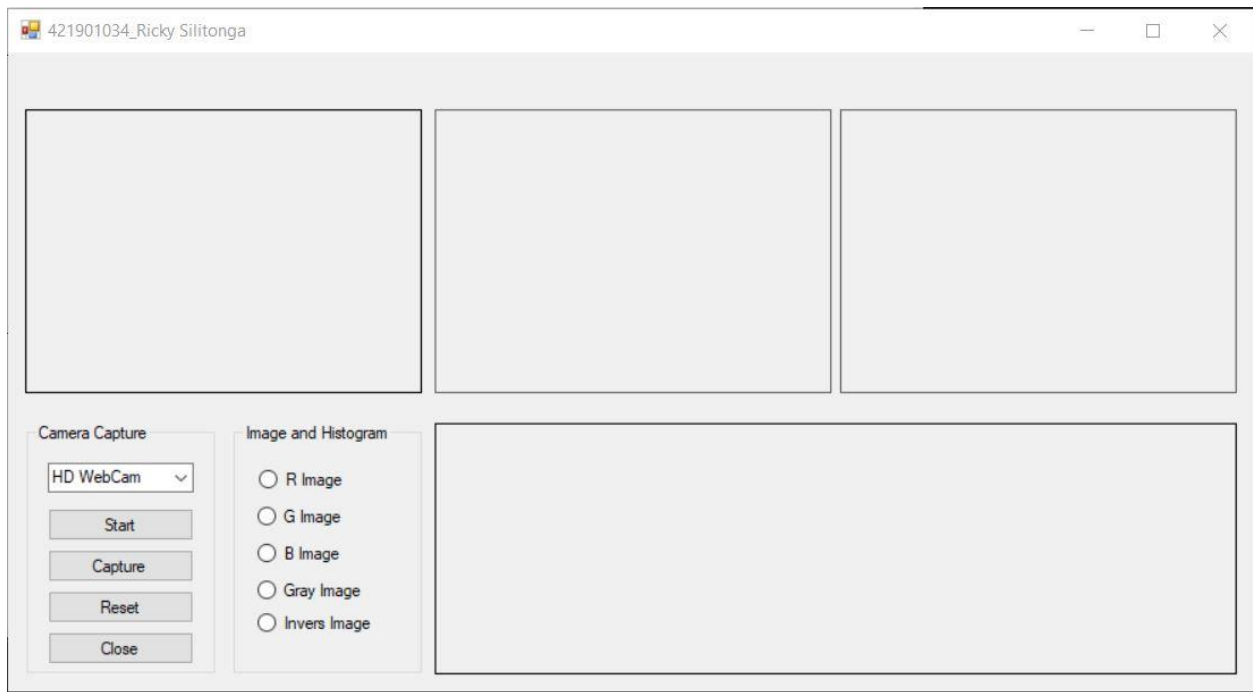


POLITEKNIK NEGERI Batam

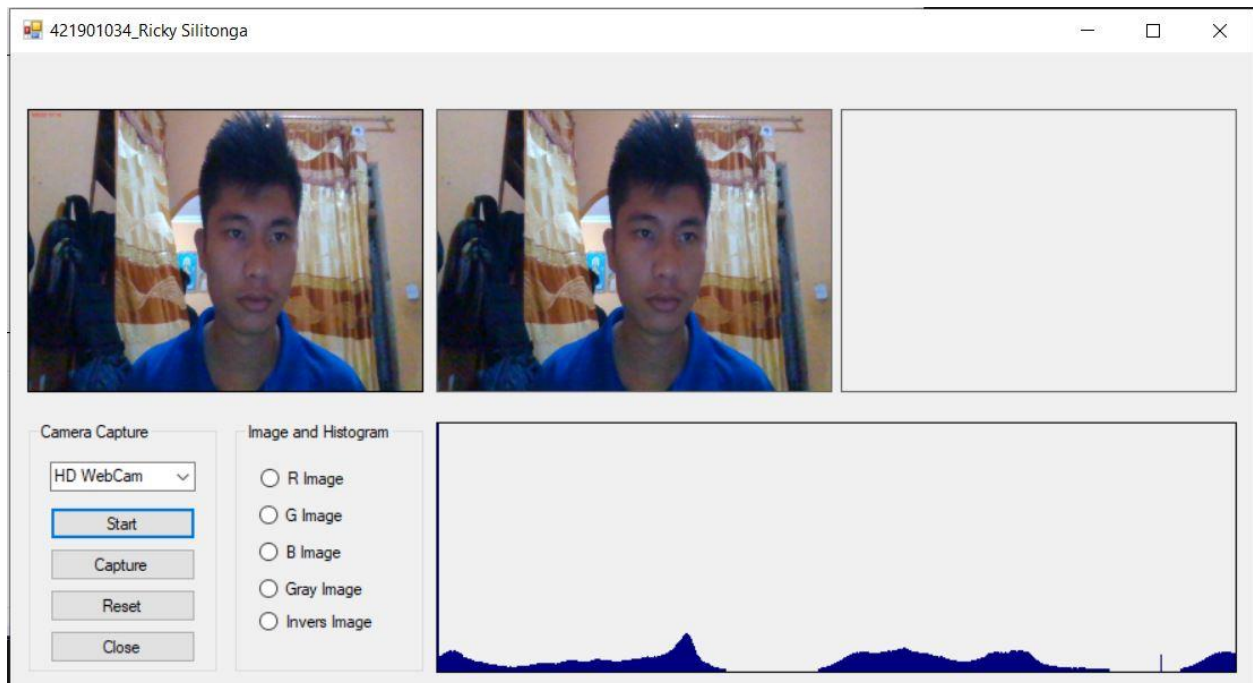
**Disusun oleh :
Ricky Silitonga (4211901034)**

**PROGRAM STUDI TEKNIK MEKATRONIKA
JURUSAN TEKNIK ELEKTRO
POLITEKNIK NEGERI BATAM
2021**

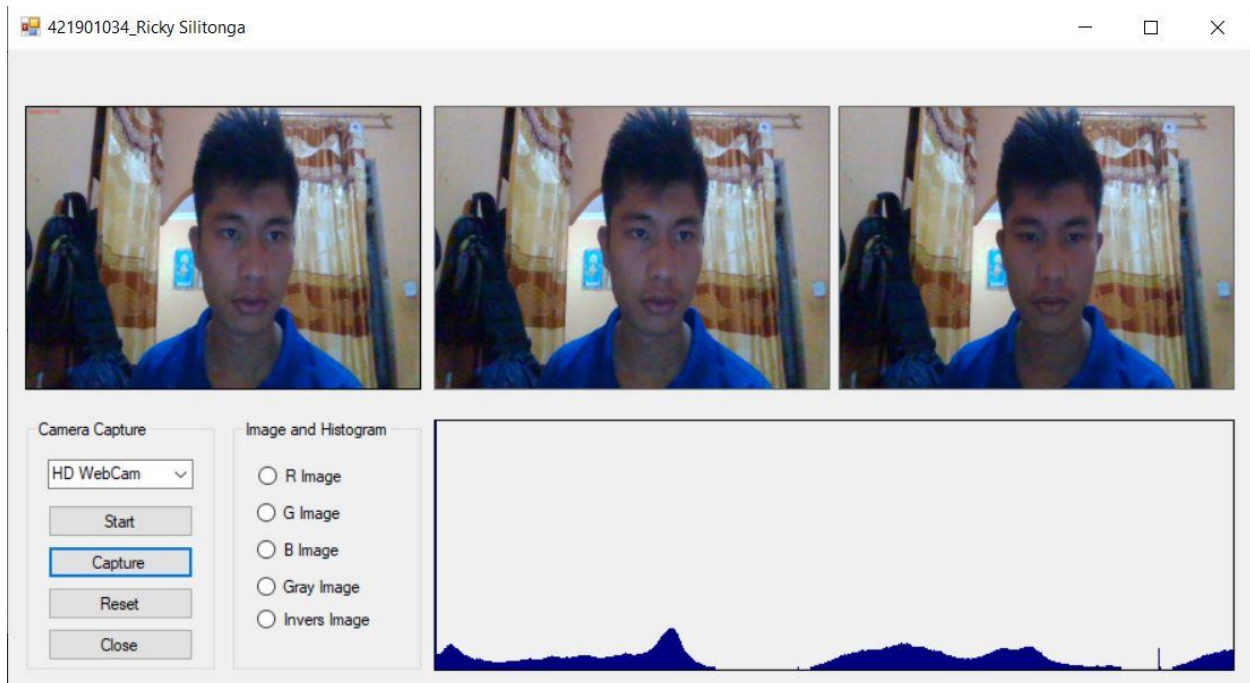
Introduction To Camera Capture Based On Aforge



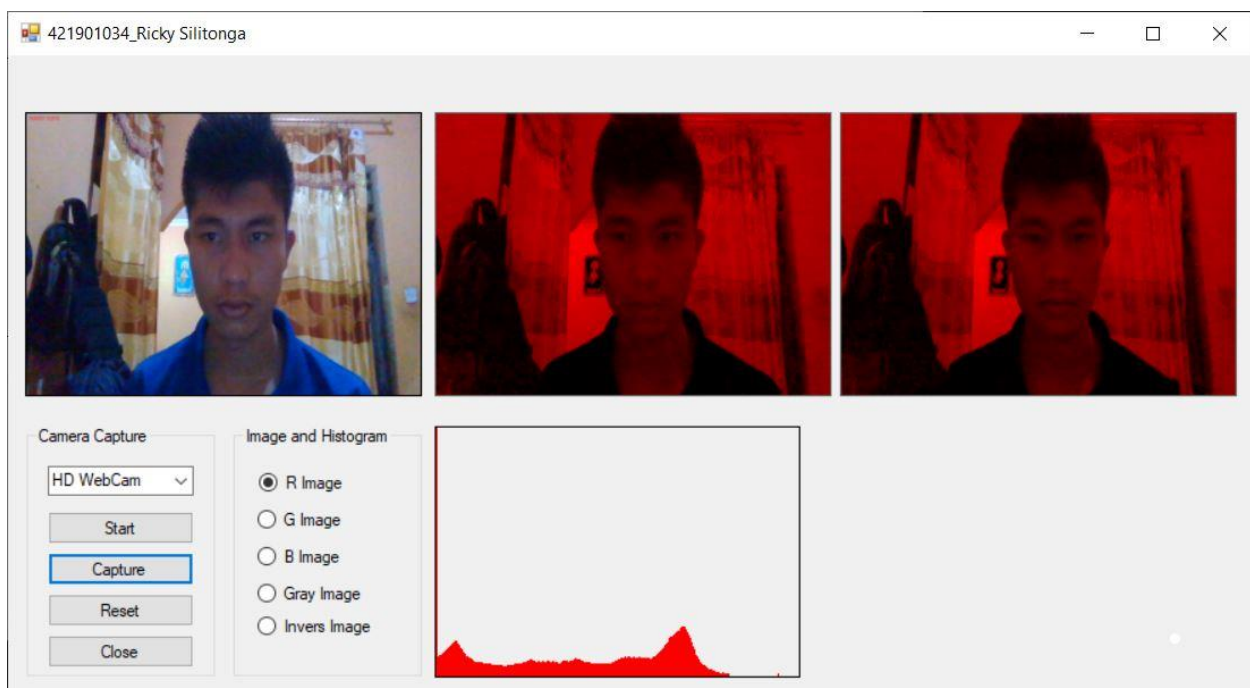
Gambar 1. Tampilan Awal



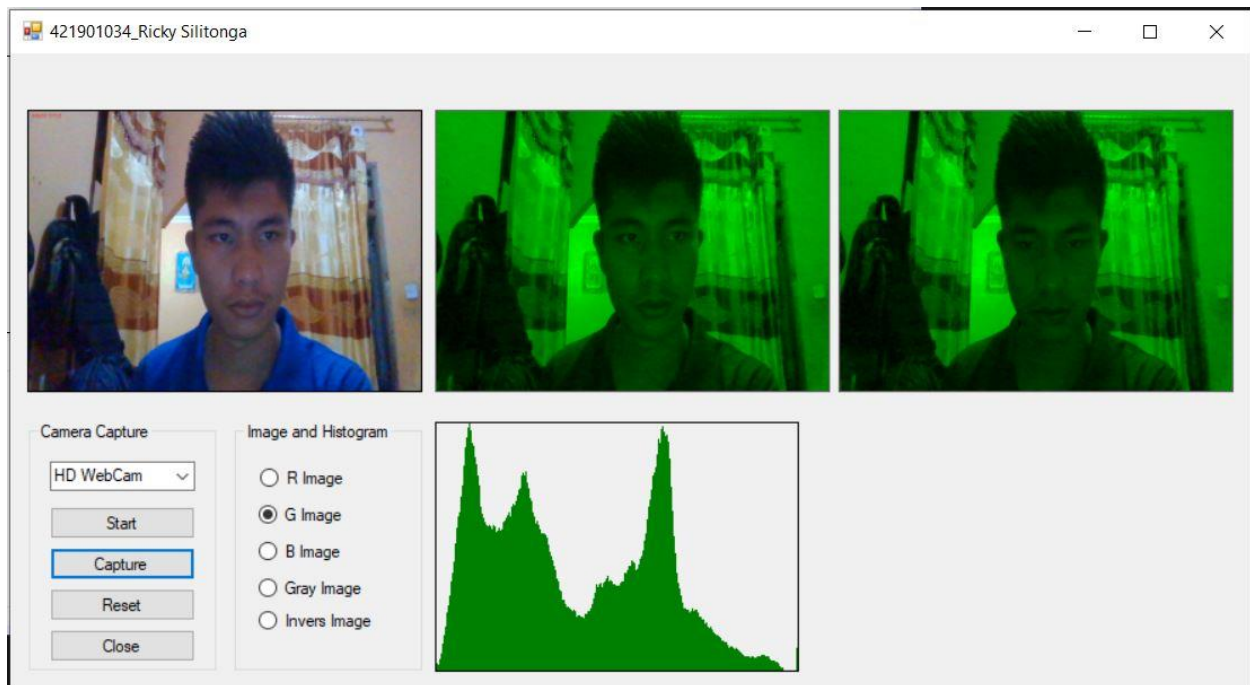
Gambar 2. Start Webcam



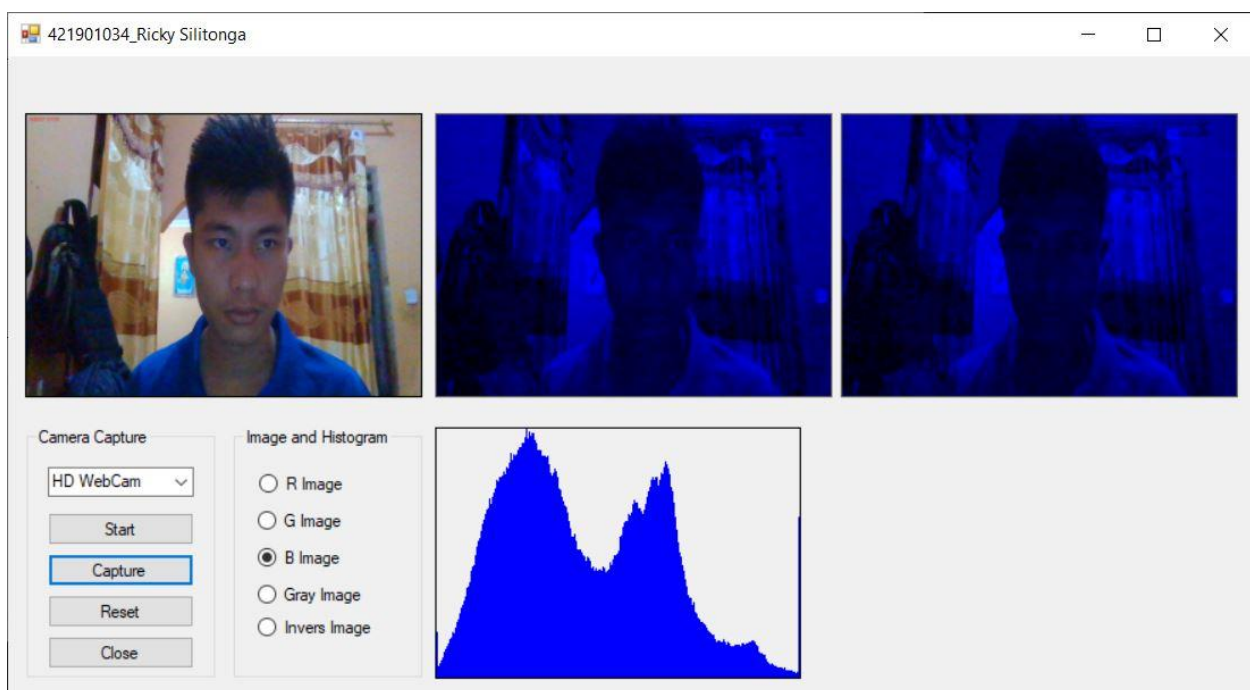
Gambar 3. Tombol Capture di klik



Gambar 4. R Image + Capture



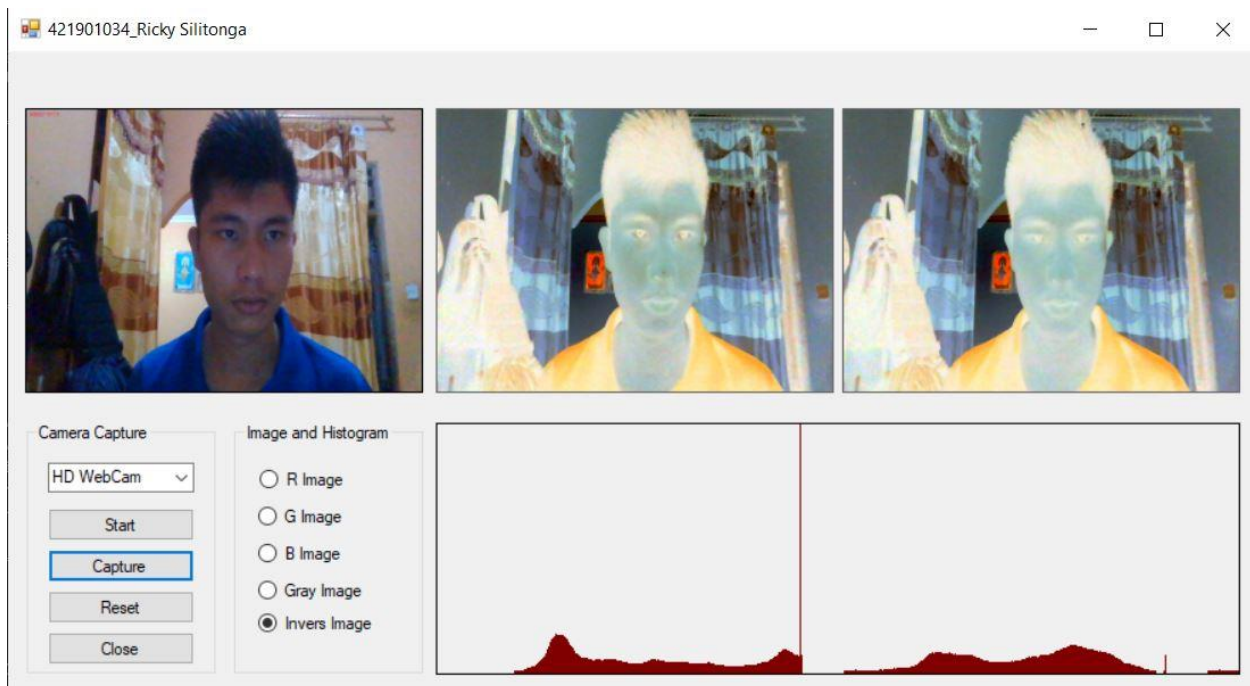
Gambar 5. G Image + Capture



Gambar 6. B Image + Capture



Gambar 7. Gray Image + Capture



Gambar 8. Invers Image + Capture

Source Code

```
4211901034.cs* 4211901034.cs [Design]*
Pertemuan9_4211901034
Pertemuan9_4211901034.Form1
hitungHistogram(int channel)

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10 using System.Threading;
11 using System.IO;
12 using System.IO.Ports;
13 using System.Collections;
14 using System.Drawing.Imaging;
15 using AForge;
16 using AForge.Imaging;
17 using AForge.Imaging.Filters;
18 using AForge.Video;
19 using AForge.Video.DirectShow;
20 using AForge.Math.Geometry;
21
22 namespace Pertemuan9_4211901034
23 {
24     3 references
25     public partial class Form1 : Form
26     {
27         // global
28         private FilterInfoCollection videoDevices;
29         private VideoCaptureDevice videoDevice;
30         private VideoCapabilities[] snapshotCapabilities;
31         private ArrayList listCamera = new ArrayList();
32         public string pathFolder = Application.StartupPath + @"\ImageCapture\";
33         //for capturing image
34         bool needSnapshot = false;
35         int imageChannel = 0;
36         //image variabel
37         Bitmap sourceImage = null;
38         Bitmap processedImage = null;
39         Bitmap grayImage = null;
40         Bitmap invertImage = null;
41
42     }
43
44     1 reference
45     public Form1()
46     {
47         InitializeComponent();
48         //list the available camera and add to comboBox
49         getListCameraUSB();
50     }
51
52     1 reference
53     private void getListCameraUSB()
54     {
55         videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);
56         if (videoDevices.Count != 0)
57         {
58             // add all devices to combo
59             foreach (FilterInfo device in videoDevices)
60             {
61                 comboBox1.Items.Add(device.Name);
62             }
63         }
64         else
65         {
66             comboBox1.Items.Add("No DirectShow devices found");
67         }
68         comboBox1.SelectedIndex = 0;
69     }
70
71     // usb camera definition
72     private static string _usbcamera;
73     2 references
74     public string usbcamera
75     {
76         get { return _usbcamera; }
77         set { _usbcamera = value; }
78     }
79
80     // opening the video source
81     1 reference
82     private void OpenVideoSource(IVideoSource source)
83     {
84         try
85         {
86             // set busy cursor
87         }
88     }
89
90     95 % No issues found Ln: 329 Ch: 36 SPC CRLF
```

```
4211901034.cs* 4211901034.cs [Design]*
Pertemuan9_4211901034
Pertemuan9_4211901034.Form1
hitungHistogram(int channel)

41 public Form1()
42 {
43     InitializeComponent();
44     //list the available camera and add to comboBox
45     getListCameraUSB();
46 }
47
48 1 reference
49 private void getListCameraUSB()
50 {
51     videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);
52     if (videoDevices.Count != 0)
53     {
54         // add all devices to combo
55         foreach (FilterInfo device in videoDevices)
56         {
57             comboBox1.Items.Add(device.Name);
58         }
59     }
60     else
61     {
62         comboBox1.Items.Add("No DirectShow devices found");
63     }
64     comboBox1.SelectedIndex = 0;
65 }
66
67 // usb camera definition
68 private static string _usbcamera;
69 2 references
70 public string usbcamera
71 {
72     get { return _usbcamera; }
73     set { _usbcamera = value; }
74 }
75
76 // opening the video source
77 1 reference
78 private void OpenVideoSource(IVideoSource source)
79 {
80     try
81     {
82         // set busy cursor
83     }
84 }
85
86 95 % No issues found Ln: 329 Ch: 36 SPC CRLF
```

```
4211901034.cs* 4211901034.cs [Design]*
Pertemuan9_4211901034
Pertemuan9_4211901034.Form1
hitungHistogram(int channel)

79      this.Cursor = Cursors.WaitCursor;
80      // stop current video source
81      CloseCurrentVideoSource();
82      // start new video source
83      videoSourcePlayer1.VideoSource = source;
84      videoSourcePlayer1.Start();
85      this.Cursor = Cursors.Default;
86  }
87  catch { }
88  }
89  private void OpenVideoSource([VideoSource source]
1 reference
90  public void CloseCurrentVideoSource()
91  {
92      try
93      {
94          if (videoSourcePlayer1.VideoSource != null)
95          {
96              videoSourcePlayer1.SignalToStop();
97              // wait ~ 3 seconds
98              for (int i = 0; i < 30; i++)
99              {
100                  if (!videoSourcePlayer1.IsRunning)
101                      break;
102                  System.Threading.Thread.Sleep(100);
103              }
104              if (videoSourcePlayer1.IsRunning)
105              {
106                  videoSourcePlayer1.Stop();
107                  videoSourcePlayer1.VideoSource = null;
108              }
109          }
110          catch { }
111      }
112  }
113  private void button1_Click(object sender, EventArgs e)
114  {
115      OpenCamera();
116  }
117  }
1 reference

95 % No issues found Ln: 329 Ch: 36 SPC CRLF
```

```
4211901034.cs* 4211901034.cs [Design]*
Pertemuan9_4211901034
Pertemuan9_4211901034.Form1
hitungHistogram(int channel)

118  private void OpenCamera()
119  {
120      try
121      {
122          usbcamera = comboBox1.SelectedIndex.ToString();
123          videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);
124          if (videoDevices.Count != 0)
125          {
126              // add all devices to combo
127              foreach (FilterInfo device in videoDevices)
128              {
129                  listCamera.Add(device.Name);
130              }
131          }
132          else
133          {
134              MessageBox.Show("Camera devices found");
135          }
136          videoDevice = new
137          VideoCaptureDevice(videoDevices[Convert.ToInt32(usbcamera)].MonikerString);
138          snapshotCapabilities = videoDevice.SnapshotCapabilities;
139          if (snapshotCapabilities.Length == 0)
140          {
141              MessageBox.Show("Camera Capture Not supported");
142          }
143          OpenVideoSource(videoDevice);
144      }
145      catch (Exception err)
146      {
147          MessageBox.Show(err.ToString());
148      }
149  }
150  }
151  public delegate void CaptureSnapshotManifest(Bitmap image);
152  1 reference
153  public void UpdateCaptureSnapshotManifest(Bitmap image)
154  {
155      try
156      {
157          needSnapshot = false;
158          pictureBox2.Image = image;
159      }
160  }

95 % No issues found Ln: 329 Ch: 36 SPC CRLF
```

```
4211901034.cs* x 4211901034.cs [Design]*
Pertemuan9_4211901034
pictureBox2.Update();
string namaImage = "sampleImage";
string nameCapture = namaImage + "-" +
DateTime.Now.ToString("yyyyMMddHHmmss") + ".bmp";
if (Directory.Exists(pathFolder))
{
    pictureBox2.Image.Save(pathFolder + nameCapture,
    ImageFormat.Bmp);
}
else
{
    Directory.CreateDirectory(pathFolder);
    pictureBox2.Image.Save(pathFolder + nameCapture,
    ImageFormat.Bmp);
}
}
try
catch { }
}
public void UpdateCaptureSnapshotManifest(Bitmap image)

1 reference
private void videoSourcePlayer1_NewFrame(object sender, ref Bitmap image)
{
    try
    {
        DateTime now = DateTime.Now;
        Graphics g = Graphics.FromImage(image);
        sourceImage = image.Clone() as Bitmap;
        //process the image
        processedImage = channelFiltering(imageChannel);
        hitungHistogram(imageChannel);
        //display the processed image
        pictureBox1.Image = processedImage;
        // paint current time
        SolidBrush brush = new SolidBrush(Color.Red);
        g.DrawString(now.ToString(), this.Font, brush, new PointF(5, 5));
        brush.Dispose();
        if (needSnapshot)
        {
            this.Invoke(new
            CaptureSnapshotManifest(UpdateCaptureSnapshotManifest), processedImage);
        }
    }
    g.Dispose();
}
```

```
4211901034.cs* x 4211901034.cs [Design]*
Pertemuan9_4211901034
}
try
catch
{
}
}
}
private void videoSourcePlayer1_NewFrame(object sender, ref Bitmap image)

1 reference
private void button2_Click(object sender, EventArgs e)
{
    needSnapshot = true;
}

1 reference
private Bitmap channelFiltering(int channel)
{
    if (sourceImage == null) return null;
    //image initialization
    Bitmap image = new Bitmap(sourceImage);
    // create filter
    ChannelFiltering filter = new ChannelFiltering();
    // RGB image
    if (channel == 0)
    {
        filter.Red = new IntRange(0, 255);
        filter.Green = new IntRange(0, 255);
        filter.Blue = new IntRange(0, 255);
        //apply the filter
        image = filter.Apply(sourceImage);
    }
    // R image
    else if (channel == 1)
    {
        filter.Red = new IntRange(0, 255);
        filter.Green = new IntRange(0, 0);
        filter.Blue = new IntRange(0, 0);
        //apply the filter
        image = filter.Apply(sourceImage);
    }
    // G image
    else if (channel == 2)
    {
    }
}
```



```

4211901034.cs* 4211901034.cs [Design]*
Pertemuan9_4211901034 Pertemuan9_4211901034.Form1 hitungHistogram(int channel)

239 // tambahkan coding
240 filter.Red = new IntRange(0, 0);
241 filter.Green = new IntRange(0, 255);
242 filter.Blue = new IntRange(0, 0);
243 //apply the filter
244 image = filter.Apply(sourceImage);
245 } else if (channel == 2)
246 // B image
247 else if (channel == 3)
248 {
249 // tambahkan coding
250 filter.Red = new IntRange(0, 0);
251 filter.Green = new IntRange(0, 255);
252 filter.Blue = new IntRange(0, 255);
253 //apply the filter
254 image = filter.Apply(sourceImage);
255 }
256 else if (channel == 4)
257 {
258 FiltersSequence filter1 = new AForge.Imaging.Filters.FiltersSequence();
259 filter1.Add(new Grayscale(0.299, 0.587, 0.144));
260 grayImage = filter1.Apply(sourceImage);
261 image = grayImage;
262 }
263 else if (channel == 5)
264 {
265 Invert filterInvert = new Invert();
266 //apply the filter
267 invertImage = filterInvert.Apply(sourceImage);
268 image = invertImage;
269 }
270 return image;
271 } private Bitmap channelFiltering(int channel)
272
273 2 references
274 int[] gabungHistogram(int[] r, int[] g, int[] b)
275 {
276 int[] c = new int[256 * 3];
277 for (int i = 0; i < 256; i++)
278 c[i] = r[i];
279 for (int i = 256; i < 512; i++)
280 c[i] = g[i - 256];
281
282 for (int i = 512; i < 768; i++)
283 c[i] = b[i - 512];
284 return c;
285 }
286 private void setImageChannel(int channel)
287 {
288 imageChannel = channel;
289 }
290 private void radioButtonReset()
291 {
292 radioButton1.Checked = false;
293 radioButton2.Checked = false;
294 radioButton3.Checked = false;
295 radioButton4.Checked = false;
296 }
297
298 1 reference
299 private void button3_Click(object sender, EventArgs e)
300 {
301 if (sourceImage == null) return;
302 setImageChannel(0);
303 radioButtonReset();
304 }
305 1 reference
306 private void hitungHistogram(int channel)
307 {
308 if (sourceImage == null) return;
309 ImageStatistics stat = new ImageStatistics(sourceImage);
310 // RGB histogram
311 if (channel == 0)
312 {
313 int[] redStat = stat.Red.Values;
314 int[] greenStat = stat.Blue.Values;
315 int[] blueStat = stat.Blue.Values;
316 int[] gab = gabungHistogram(redStat, greenStat, blueStat);
317 histogram1.Color = Color.Navy;
318 histogram1.Values = gab;
319 }
320 }

```

95 % No issues found Ln: 329 Ch: 36 SPC CRLF

```

4211901034.cs* 4211901034.cs [Design]*
Pertemuan9_4211901034 Pertemuan9_4211901034.Form1 hitungHistogram(int channel)

280 for (int i = 512; i < 768; i++)
281 c[i] = b[i - 512];
282 return c;
283 }
284 private void setImageChannel(int channel)
285 {
286 imageChannel = channel;
287 }
288
289 private void radioButtonReset()
290 {
291 radioButton1.Checked = false;
292 radioButton2.Checked = false;
293 radioButton3.Checked = false;
294 radioButton4.Checked = false;
295 }
296
297 private void button3_Click(object sender, EventArgs e)
298 {
299 if (sourceImage == null) return;
300 setImageChannel(0);
301 radioButtonReset();
302 }
303
304 private void hitungHistogram(int channel)
305 {
306 if (sourceImage == null) return;
307 ImageStatistics stat = new ImageStatistics(sourceImage);
308 // RGB histogram
309 if (channel == 0)
310 {
311 int[] redStat = stat.Red.Values;
312 int[] greenStat = stat.Blue.Values;
313 int[] blueStat = stat.Blue.Values;
314 int[] gab = gabungHistogram(redStat, greenStat, blueStat);
315 histogram1.Color = Color.Navy;
316 histogram1.Values = gab;
317 }
318 }

```

95 % No issues found Ln: 329 Ch: 36 SPC CRLF

```

4211901034.cs* x 4211901034.cs [Design]*
Pertemuan9_4211901034
Pertemuan9_4211901034.Form1
hitungHistogram(int channel)

318 } if (channel == 0)
319 // R histogram
320 else if (channel == 1)
321 {
322     int[] redStat = stat.Red.Values;
323     histogram1.Color = Color.Red;
324     histogram1.Values = redStat;
325 }
326 // G histogram
327 else if (channel == 2)
328 {
329     // tambahkan koding
330     int[] greenStat = stat.Green.Values;
331     histogram1.Color = Color.Green;
332     histogram1.Values = greenStat;
333 }
334 // B histogram
335 else if (channel == 3)
336 {
337     // tambahkan koding
338     int[] blueStat = stat.Blue.Values;
339     histogram1.Color = Color.Blue;
340     histogram1.Values = blueStat;
341 }
342 // Gray histogram
343 else if (channel == 4)
344 {
345     ImageStatistics grayStat = new ImageStatistics(grayImage);
346     int[] grayHis = grayStat.Gray.Values;
347     histogram1.Color = Color.Gray;
348     histogram1.Values = grayHis;
349 }
350 // Invers histogram
351 else if (channel == 5)
352 {
353     ImageStatistics invertStat = new ImageStatistics(invertImage);
354     int[] redStat = invertStat.Red.Values;
355     int[] greenStat = invertStat.Green.Values;
356     int[] blueStat = invertStat.Blue.Values;
357     int[] gab = gabungHistogram(redStat, greenStat, blueStat);
358     histogram1.Color = Color.Maroon;
359     histogram1.Values = gab;
360 }
361 } else if (channel == 5)
362 }
363 private void hitungHistogram(int channel)
364 {
365     //if the source image is not yet open..don't execute
366     if (sourceImage == null) return;
367     setImageChannel(1);
368 }
369
370 private void radioButton2_CheckedChanged(object sender, EventArgs e)
371 {
372     //if the source image is not yet open..don't execute
373     if (sourceImage == null) return;
374     setImageChannel(2);
375 }
376
377 private void radioButton3_CheckedChanged(object sender, EventArgs e)
378 {
379     //if the source image is not yet open..don't execute
380     if (sourceImage == null) return;
381     setImageChannel(3);
382 }
383
384 private void radioButton4_CheckedChanged(object sender, EventArgs e)
385 {
386     //if the source image is not yet open..don't execute
387     if (sourceImage == null) return;
388     setImageChannel(4);
389 }
390
391 private void radioButton5_CheckedChanged(object sender, EventArgs e)
392 {
393     //if the source image is not yet open..don't execute
394     if (sourceImage == null) return;
395     setImageChannel(5);
396 }
397
398 }
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

4211901034.cs* x 4211901034.cs [Design]*
Pertemuan9_4211901034
Pertemuan9_4211901034.Form1
hitungHistogram(int channel)

359     histogram1.Values = gab;
360 } else if (channel == 5)
361 }
362 private void hitungHistogram(int channel)
363 {
364     //if the source image is not yet open..don't execute
365     if (sourceImage == null) return;
366     setImageChannel(1);
367 }
368
369
370 private void radioButton2_CheckedChanged(object sender, EventArgs e)
371 {
372     //if the source image is not yet open..don't execute
373     if (sourceImage == null) return;
374     setImageChannel(2);
375 }
376
377 private void radioButton3_CheckedChanged(object sender, EventArgs e)
378 {
379     //if the source image is not yet open..don't execute
380     if (sourceImage == null) return;
381     setImageChannel(3);
382 }
383
384 private void radioButton4_CheckedChanged(object sender, EventArgs e)
385 {
386     //if the source image is not yet open..don't execute
387     if (sourceImage == null) return;
388     setImageChannel(4);
389 }
390
391 private void radioButton5_CheckedChanged(object sender, EventArgs e)
392 {
393     //if the source image is not yet open..don't execute
394     if (sourceImage == null) return;
395     setImageChannel(5);
396 }
397
398 }
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```
4211901034.cs* 4211901034.cs [Design]*
Pertemuan9_4211901034 Pertemuan9_4211901034.Form1 hitungHistogram(int channel)
396 } private void radioButton5_CheckedChanged(object sender, EventArgs e)
397
398 1 reference
398 private void Form1_FormClosed(object sender, FormClosedEventArgs e)
399 {
400     if (videoDevice != null && videoDevice.IsRunning)
401         videoDevice.Stop();
402 }
403
404 1 reference
404 private void button4_Click(object sender, EventArgs e)
405 {
406     Close();
407 }
408 } public partial class Form1 : Form
409 namespace Pertemuan9_4211901034
410
```