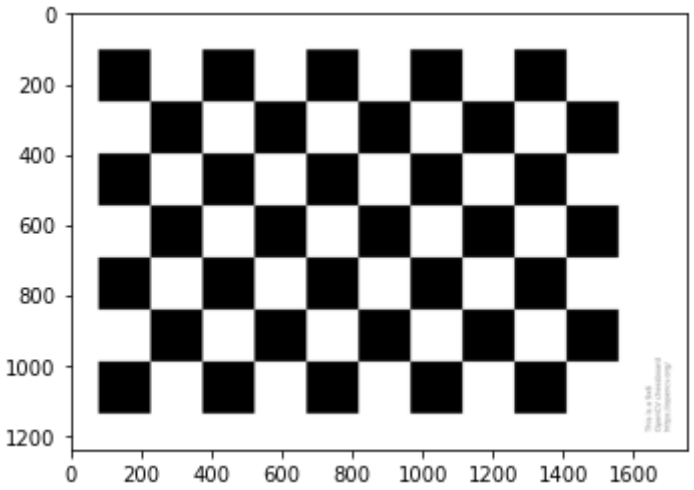
Imports

In [1]:
```
%matplotlib inline

import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import glob
import warnings

warnings.filterwarnings('ignore')
```

Patrón

Se utilizará el patrón de tablero de ajedrez de **9X6** de **opencv.org**.

In [2]:
```
pattern = cv.imread("pattern.png")
plt.imshow(pattern)
plt.show()
```

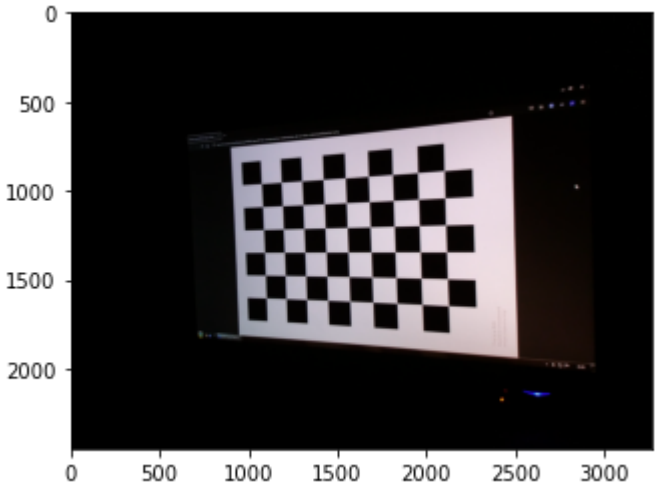


Fotos

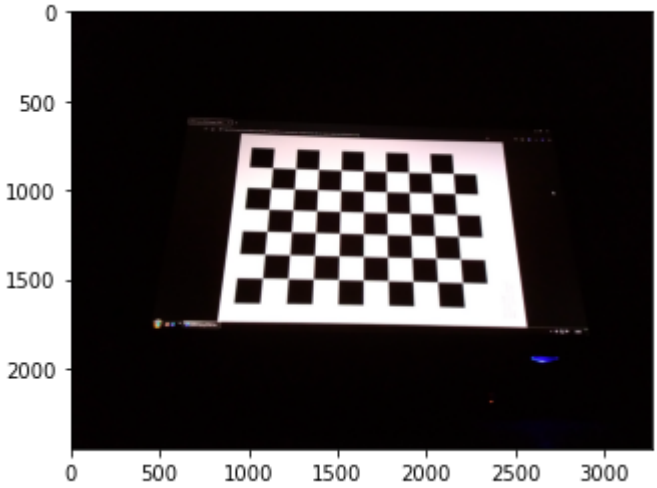
Se utilizarán las siguientes fotos para realiza la calibración.

In [3]:
```
filenames = glob.glob("./pictures/*")

for filename in filenames:
    print("processing image {0}...".format(filename))
    image = cv.imread(filename)
    plt.figure(filename)
    plt.imshow(image)
    plt.show()
```

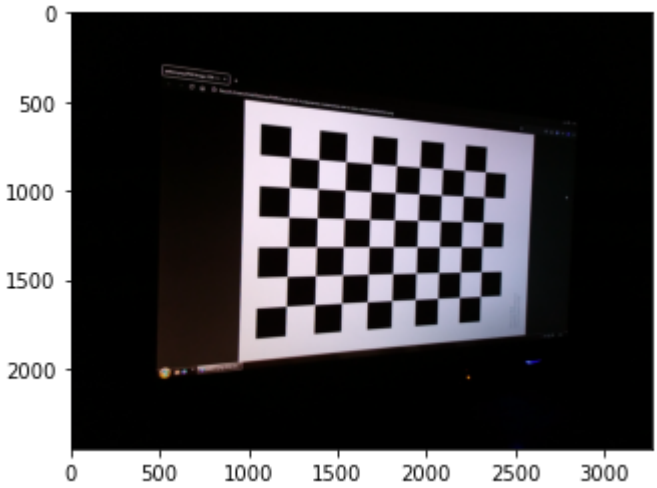
```
processing image ./pictures/06-center-right.jpg...
```



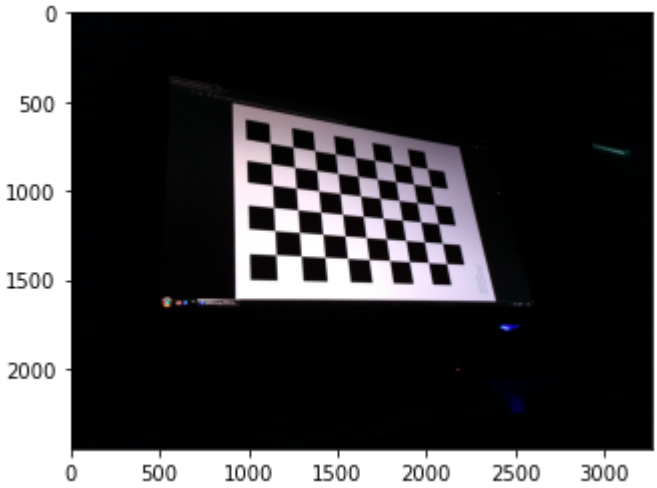

```
processing image ./pictures/08-bottom.jpg...
```



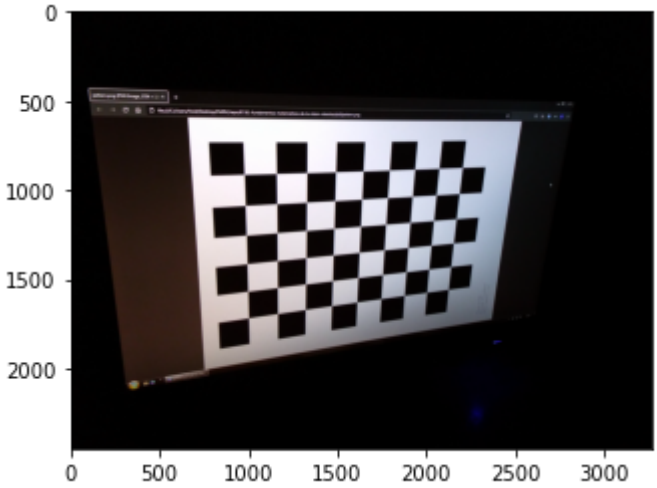

```
processing image ./pictures/04-center-left.jpg...
```



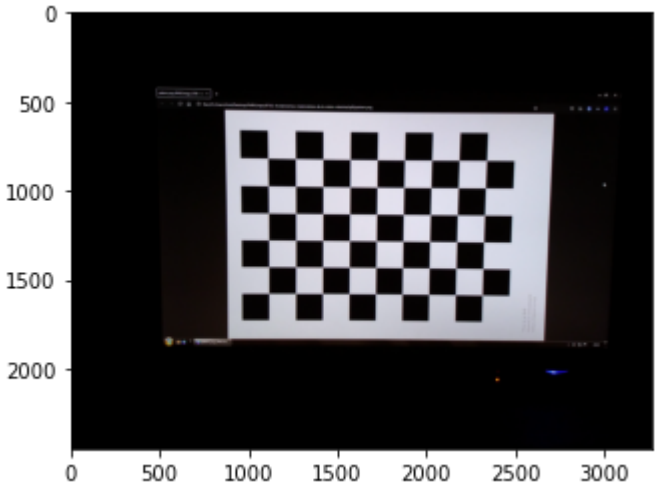

```
processing image ./pictures/07-bottom-left.jpg...
```



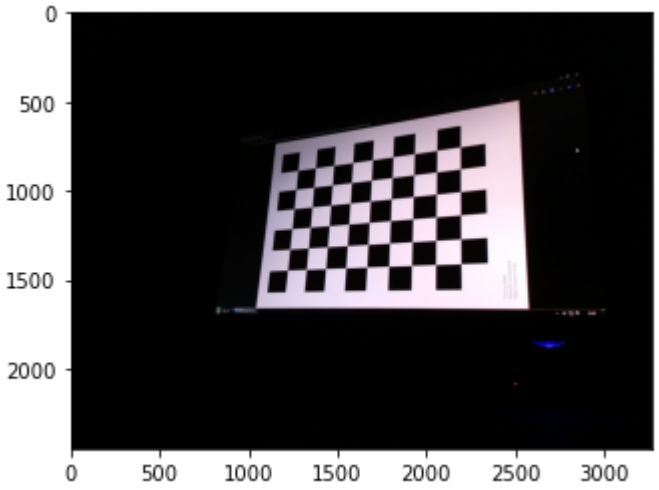

```
processing image ./pictures/01-top-left.jpg...
```



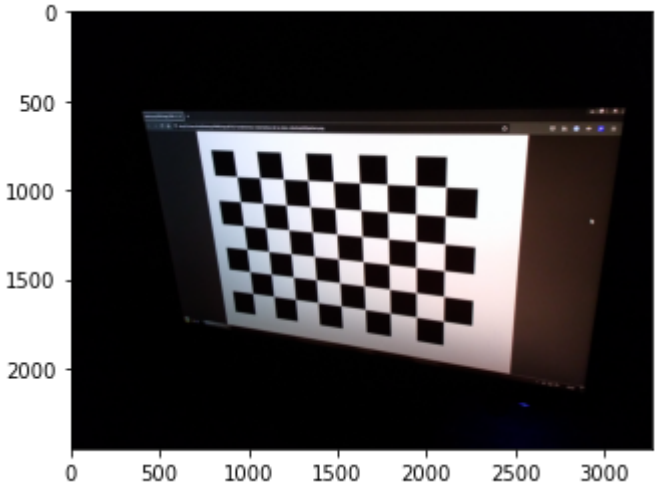

```
processing image ./pictures/05-center.jpg...
```



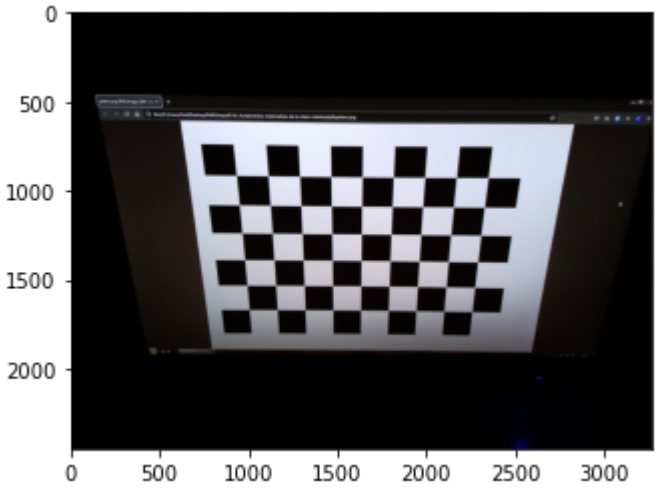

```
processing image ./pictures/09-bottom-right.jpg...
```




```
processing image ./pictures/03-top-right.jpg...
```




```
processing image ./pictures/02-top.jpg...
```



Prueba de flags para el método *findChessboardCorners*

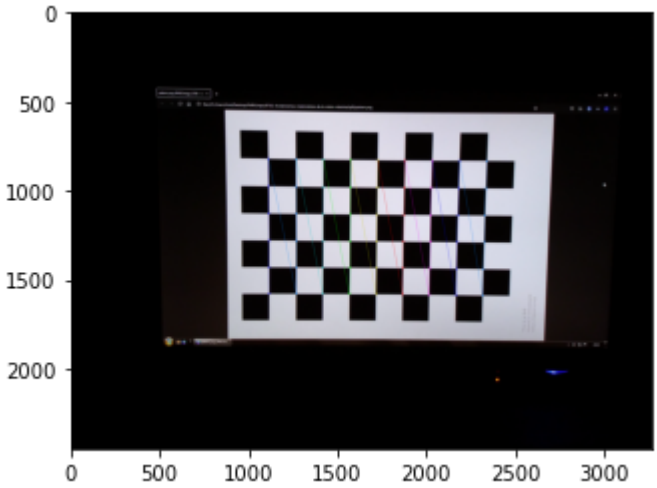
Para esta prueba se utilizará una sola imagen.

In [4]:
```
chessboard_size = (6, 9)
flag_names = ("cv.CALIB_CB_ADAPTIVE_THRESH", "cv.CALIB_CB_FILTER_QUADS",
              "cv.CALIB_CB_NORMALIZE_IMAGE", "cv.CALIB_CB_FAST_CHECK")

for flag_name in flag_names:
    print("Testing flag {0}...".format(flag_name))
```

```
    flag = eval(flag_name)
    original_image = cv.imread("pictures/05-center.jpg")
    grayscale_image = cv.cvtColor(original_image, cv.COLOR_BGR2GRAY)
    ret, corners = cv.findChessboardCorners(grayscale_image, chessboard_size,
    if ret:
        print("Results for flag {0}: ".format(flag_name))
        print(corners)
        cv.drawChessboardCorners(original_image, chessboard_size, corners, re
        plt.figure(flag_name)
        plt.imshow(original_image)
        plt.show()
    else:
        print("No results for flag {0}: ".format(flag_name))
```

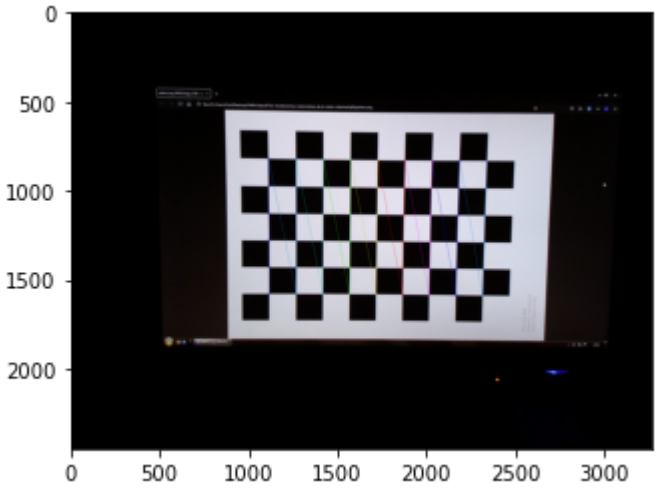
```
Testing flag cv.CALIB_CB_ADAPTIVE_THRESH...
Results for flag cv.CALIB_CB_ADAPTIVE_THRESH:
[[[1112.9852  1577.4095 ]]

 [[1112.3035  1429.0598 ]]

 [[1112.7931  1278.8617 ]]

 [[1110.981   1128.4026 ]]

 [[1109.8995   975.3975 ]]

 [[1107.9417   820.7854 ]]

 [[1263.5817  1578.1587 ]]

 [[1264.8599  1429.3958 ]]

 [[1264.5221  1280.7739 ]]

 [[1264.4453  1129.9733 ]]

 [[1263.485    977.67896]]

 [[1263.0122   823.1131 ]]

 [[1414.7375  1578.6123 ]]

 [[1415.1323  1430.8042 ]]

 [[1415.5928  1281.9642 ]]

 [[1416.4503  1132.0125 ]]

 [[1417.1158   979.54913]]

 [[1416.609    825.8532 ]]

 [[1563.3059  1579.4685 ]]

 [[1565.6012  1431.3553 ]]

 [[1566.2306  1282.9388 ]]

 [[1568.3417  1132.8981 ]]

 [[1568.8752   981.83936]]

 [[1570.6351   827.4061 ]]

 [[1713.266   1580.7281 ]]

 [[1714.5482  1433.1355 ]]

 [[1717.2826  1284.1808 ]]
```

```
[[1718.5841  1134.8912 ]]
[[1721.0111   982.8544 ]]
[[1722.725    829.75616]]
[[1862.0637  1582.4506 ]]
[[1864.8058  1434.1605 ]]
[[1866.3286  1285.5621 ]]
[[1869.4536  1136.0074 ]]
[[1872.2546   984.8714 ]]
[[1875.6019   830.74524]]
[[2012.142   1583.6978 ]]
[[2013.9513  1436.0046 ]]
[[2017.5585  1287.1648 ]]
[[2020.3809  1137.1628 ]]
[[2024.4387   985.7778 ]]
[[2028.6757   831.9389 ]]
[[2161.7676  1586.3134 ]]
[[2164.9753  1437.5162 ]]
[[2168.592   1288.9746 ]]
[[2173.2703  1138.4323 ]]
[[2177.5432   987.     ]]
[[2182.7153   832.4078 ]]
[[2311.959   1587.7694 ]]
[[2315.6448  1439.5238 ]]
[[2320.594   1290.3795 ]]
[[2325.      1140.0864 ]]
[[2330.7285   987.6259 ]]
[[2336.4985   833.5573 ]]]
```



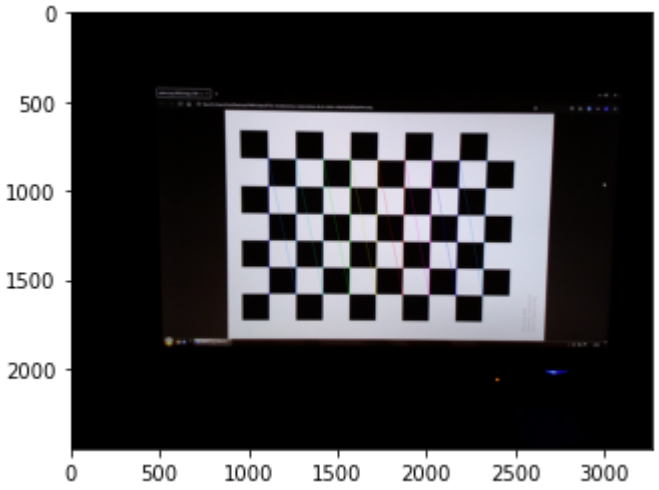

```
Testing flag cv.CALIB_CB_FILTER_QUADS...
Results for flag cv.CALIB_CB_FILTER_QUADS:
[[[1112.9463  1577.4326 ]]

 [[1112.5405  1429.2697 ]]

 [[1112.7844  1278.8738 ]]

 [[1111.0183  1128.4155 ]]

 [[1109.3641   975.9011 ]]

 [[1107.8162   820.69965]]

 [[1263.5608  1578.1421 ]]

 [[1264.8564  1429.3971 ]]

 [[1264.5221  1280.7739 ]]

 [[1264.4795  1129.9219 ]]

 [[1263.5067   977.6695 ]]

 [[1262.9471   823.18   ]]

 [[1414.7274  1578.6217 ]]

 [[1415.1323  1430.8042 ]]

 [[1415.9933  1281.5862 ]]

 [[1416.4503  1132.0125 ]]

 [[1417.1158   979.54913]]

 [[1416.5757   825.8357 ]]

 [[1563.314   1579.4812 ]]

 [[1565.6038  1431.352  ]]

 [[1566.3374  1283.1046 ]]

 [[1568.3418  1132.8982 ]]

 [[1568.8752   981.83936]]

 [[1570.6215   827.4189 ]]

 [[1713.266   1580.7281 ]]
```

```
[[1714.5162  1433.084  ]]

[[1717.2678  1284.1912 ]]

[[1718.524   1134.8038 ]]

[[1721.0095   982.8559 ]]

[[1722.725    829.75616]]

[[1862.254   1582.5748 ]]

[[1864.7612  1434.192  ]]

[[1866.3286  1285.5621 ]]

[[1869.1215  1136.3019 ]]

[[1872.2546   984.8714 ]]

[[1875.6019   830.74524]]

[[2012.142   1583.6978 ]]

[[2014.4541  1436.5356 ]]

[[2017.5682  1287.1564 ]]

[[2020.3674  1137.1763 ]]

[[2024.4387   985.7778 ]]

[[2028.6547   831.9026 ]]

[[2161.7722  1586.3329 ]]

[[2164.9739  1437.5172 ]]

[[2168.592   1288.9746 ]]

[[2173.2703  1138.4323 ]]

[[2177.5432   987.     ]]

[[2182.7153   832.4078 ]]

[[2311.824   1587.8728 ]]

[[2315.6448  1439.5238 ]]

[[2320.6138  1290.3604 ]]

[[2325.      1140.0864 ]]

[[2330.6362   987.68677]]

[[2336.4946   833.586  ]]]
```




```
Testing flag cv.CALIB_CB_NORMALIZE_IMAGE...
No results for flag cv.CALIB_CB_NORMALIZE_IMAGE:
Testing flag cv.CALIB_CB_FAST_CHECK...
Results for flag cv.CALIB_CB_FAST_CHECK:
[[[1112.9463  1577.4326 ]]

 [[1112.5405  1429.2697 ]]

 [[1112.7844  1278.8738 ]]

 [[1111.0183  1128.4155 ]]

 [[1109.3641   975.9011 ]]

 [[1107.8162   820.69965]]

 [[1263.5608  1578.1421 ]]

 [[1264.8564  1429.3971 ]]

 [[1264.5221  1280.7739 ]]

 [[1264.4795  1129.9219 ]]

 [[1263.5067   977.6695 ]]

 [[1262.9471   823.18   ]]

 [[1414.7274  1578.6217 ]]

 [[1415.1323  1430.8042 ]]

 [[1415.9933  1281.5862 ]]

 [[1416.4503  1132.0125 ]]

 [[1417.1158   979.54913]]

 [[1416.5757   825.8357 ]]

 [[1563.314   1579.4812 ]]

 [[1565.6038  1431.352  ]]

 [[1566.3374  1283.1046 ]]

 [[1568.3418  1132.8982 ]]

 [[1568.8752   981.83936]]

 [[1570.6215   827.4189 ]]
```

```
[[1713.266   1580.7281 ]]
[[1714.5162  1433.084  ]]
[[1717.2678  1284.1912 ]]
[[1718.524   1134.8038 ]]
[[1721.0095   982.8559 ]]
[[1722.725    829.75616]]
[[1862.254   1582.5748 ]]
[[1864.7612  1434.192  ]]
[[1866.3286  1285.5621 ]]
[[1869.1215  1136.3019 ]]
[[1872.2546   984.8714 ]]
[[1875.6019   830.74524]]
[[2012.142   1583.6978 ]]
[[2014.4541  1436.5356 ]]
[[2017.5682  1287.1564 ]]
[[2020.3674  1137.1763 ]]
[[2024.4387   985.7778 ]]
[[2028.6547   831.9026 ]]
[[2161.7722  1586.3329 ]]
[[2164.9739  1437.5172 ]]
[[2168.592   1288.9746 ]]
[[2173.2703  1138.4323 ]]
[[2177.5432   987.     ]]
[[2182.7153   832.4078 ]]
[[2311.824   1587.8728 ]]
[[2315.6448  1439.5238 ]]
[[2320.6138  1290.3604 ]]
[[2325.      1140.0864 ]]
[[2330.6362   987.68677]]
[[2336.4946   833.586  ]]]
```



Se observó que el flag **CALIB_CB_NORMALIZE_IMAGE** no devuelve resultados. Para los otros 3 flags, no se observan diferencias a simple vista lo que se corrobora al comparar los puntos encontrados.

Prueba de iteraciones para el método *cornerSubPix*

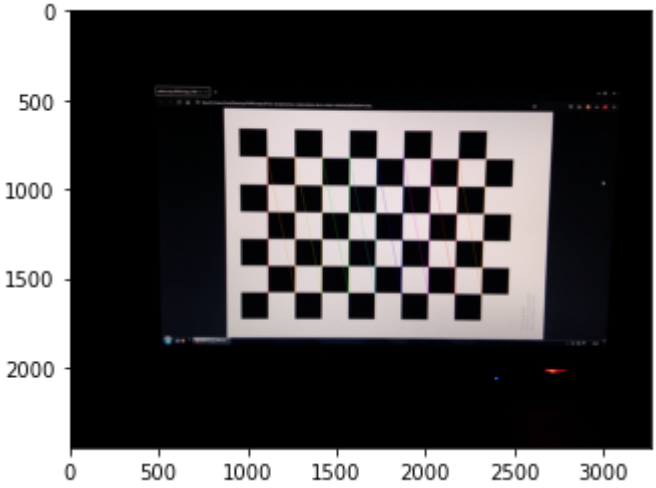
Para esta prueba se utilizará una sola imagen.

In [5]:
```
iterations = (5, 10, 15, 20, 25)

for iteration in iterations:
    print("Testing {0} iterations...".format(iteration))
    original_image = cv.imread("pictures/05-center.jpg")
    grayscale_image = cv.cvtColor(original_image, cv.COLOR_BGR2GRAY)
    ret, corners = cv.findChessboardCorners(grayscale_image, chessboard_size,
    if ret:
        print("Results for {0} iterations: ".format(iteration))
        print(corners)
        criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, iterati
        corners_subpix = cv.cornerSubPix(grayscale_image, corners, (5, 5), (-
        cv.drawChessboardCorners(original_image, chessboard_size, corners_sub
        plt.figure()
        plt.imshow(cv.cvtColor(original_image, cv.COLOR_BGR2RGB))
        plt.show()
```

```
Testing 5 iterations...
Results for 5 iterations:
[[[1112.9852  1577.4095 ]]

 [[1112.3035  1429.0598 ]]

 [[1112.7931  1278.8617 ]]

 [[1110.981   1128.4026 ]]

 [[1109.8995   975.3975 ]]

 [[1107.9417   820.7854 ]]

 [[1263.5817  1578.1587 ]]

 [[1264.8599  1429.3958 ]]

 [[1264.5221  1280.7739 ]]
```

```
[[1264.4453  1129.9733 ]]
[[1263.485    977.67896]]
[[1263.0122   823.1131 ]]
[[1414.7375  1578.6123 ]]
[[1415.1323  1430.8042 ]]
[[1415.5928  1281.9642 ]]
[[1416.4503  1132.0125 ]]
[[1417.1158   979.54913]]
[[1416.609    825.8532 ]]
[[1563.3059  1579.4685 ]]
[[1565.6012  1431.3553 ]]
[[1566.2306  1282.9388 ]]
[[1568.3417  1132.8981 ]]
[[1568.8752   981.83936]]
[[1570.6351   827.4061 ]]
[[1713.266   1580.7281 ]]
[[1714.5482  1433.1355 ]]
[[1717.2826  1284.1808 ]]
[[1718.5841  1134.8912 ]]
[[1721.0111   982.8544 ]]
[[1722.725    829.75616]]
[[1862.0637  1582.4506 ]]
[[1864.8058  1434.1605 ]]
[[1866.3286  1285.5621 ]]
[[1869.4536  1136.0074 ]]
[[1872.2546   984.8714 ]]
[[1875.6019   830.74524]]
[[2012.142   1583.6978 ]]
[[2013.9513  1436.0046 ]]
[[2017.5585  1287.1648 ]]
[[2020.3809  1137.1628 ]]
[[2024.4387   985.7778 ]]
[[2028.6757   831.9389 ]]
[[2161.7676  1586.3134 ]]
[[2164.9753  1437.5162 ]]
```

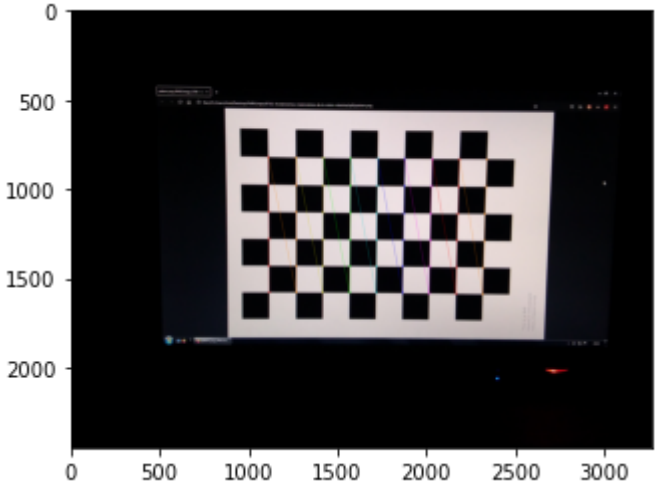
```
 [[2168.592   1288.9746 ]]

 [[2173.2703  1138.4323 ]]

 [[2177.5432   987.     ]]

 [[2182.7153   832.4078 ]]

 [[2311.959   1587.7694 ]]

 [[2315.6448  1439.5238 ]]

 [[2320.594   1290.3795 ]]

 [[2325.      1140.0864 ]]

 [[2330.7285   987.6259 ]]

 [[2336.4985   833.5573 ]]]
```




```
Testing 10 iterations...
Results for 10 iterations:
[[[1112.9852  1577.4095 ]]

 [[1112.3035  1429.0598 ]]

 [[1112.7931  1278.8617 ]]

 [[1110.981   1128.4026 ]]

 [[1109.8995   975.3975 ]]

 [[1107.9417   820.7854 ]]

 [[1263.5817  1578.1587 ]]

 [[1264.8599  1429.3958 ]]

 [[1264.5221  1280.7739 ]]

 [[1264.4453  1129.9733 ]]

 [[1263.485    977.67896]]

 [[1263.0122   823.1131 ]]

 [[1414.7375  1578.6123 ]]

 [[1415.1323  1430.8042 ]]

 [[1415.5928  1281.9642 ]]
```

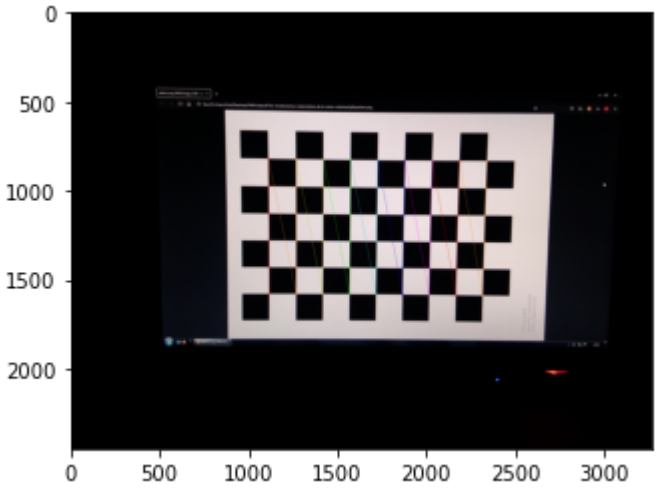
```
[[1416.4503  1132.0125 ]]
[[1417.1158   979.54913]]
[[1416.609    825.8532 ]]
[[1563.3059  1579.4685 ]]
[[1565.6012  1431.3553 ]]
[[1566.2306  1282.9388 ]]
[[1568.3417  1132.8981 ]]
[[1568.8752   981.83936]]
[[1570.6351   827.4061 ]]
[[1713.266   1580.7281 ]]
[[1714.5482  1433.1355 ]]
[[1717.2826  1284.1808 ]]
[[1718.5841  1134.8912 ]]
[[1721.0111   982.8544 ]]
[[1722.725    829.75616]]
[[1862.0637  1582.4506 ]]
[[1864.8058  1434.1605 ]]
[[1866.3286  1285.5621 ]]
[[1869.4536  1136.0074 ]]
[[1872.2546   984.8714 ]]
[[1875.6019   830.74524]]
[[2012.142   1583.6978 ]]
[[2013.9513  1436.0046 ]]
[[2017.5585  1287.1648 ]]
[[2020.3809  1137.1628 ]]
[[2024.4387   985.7778 ]]
[[2028.6757   831.9389 ]]
[[2161.7676  1586.3134 ]]
[[2164.9753  1437.5162 ]]
[[2168.592   1288.9746 ]]
[[2173.2703  1138.4323 ]]
[[2177.5432   987.     ]]
[[2182.7153   832.4078 ]]
[[2311.959   1587.7694 ]]
[[2315.6448  1439.5238 ]]
```

```
 [[2320.594   1290.3795 ]]

 [[2325.      1140.0864 ]]

 [[2330.7285   987.6259 ]]

 [[2336.4985   833.5573 ]]]
```



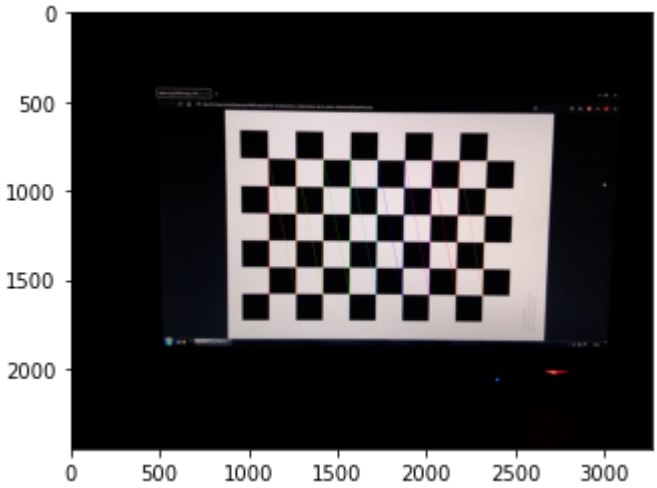

```
Testing 15 iterations...
Results for 15 iterations:
[[[1112.9852  1577.4095 ]]

 [[1112.3035  1429.0598 ]]

 [[1112.7931  1278.8617 ]]

 [[1110.981   1128.4026 ]]

 [[1109.8995   975.3975 ]]

 [[1107.9417   820.7854 ]]

 [[1263.5817  1578.1587 ]]

 [[1264.8599  1429.3958 ]]

 [[1264.5221  1280.7739 ]]

 [[1264.4453  1129.9733 ]]

 [[1263.485    977.67896]]

 [[1263.0122   823.1131 ]]

 [[1414.7375  1578.6123 ]]

 [[1415.1323  1430.8042 ]]

 [[1415.5928  1281.9642 ]]

 [[1416.4503  1132.0125 ]]

 [[1417.1158   979.54913]]

 [[1416.609    825.8532 ]]

 [[1563.3059  1579.4685 ]]

 [[1565.6012  1431.3553 ]]

 [[1566.2306  1282.9388 ]]
```

```
[[1568.3417  1132.8981 ]]
[[1568.8752   981.83936]]
[[1570.6351   827.4061 ]]
[[1713.266   1580.7281 ]]
[[1714.5482  1433.1355 ]]
[[1717.2826  1284.1808 ]]
[[1718.5841  1134.8912 ]]
[[1721.0111   982.8544 ]]
[[1722.725    829.75616]]
[[1862.0637  1582.4506 ]]
[[1864.8058  1434.1605 ]]
[[1866.3286  1285.5621 ]]
[[1869.4536  1136.0074 ]]
[[1872.2546   984.8714 ]]
[[1875.6019   830.74524]]
[[2012.142   1583.6978 ]]
[[2013.9513  1436.0046 ]]
[[2017.5585  1287.1648 ]]
[[2020.3809  1137.1628 ]]
[[2024.4387   985.7778 ]]
[[2028.6757   831.9389 ]]
[[2161.7676  1586.3134 ]]
[[2164.9753  1437.5162 ]]
[[2168.592   1288.9746 ]]
[[2173.2703  1138.4323 ]]
[[2177.5432   987.     ]]
[[2182.7153   832.4078 ]]
[[2311.959   1587.7694 ]]
[[2315.6448  1439.5238 ]]
[[2320.594   1290.3795 ]]
[[2325.      1140.0864 ]]
[[2330.7285   987.6259 ]]
[[2336.4985   833.5573 ]]]
```



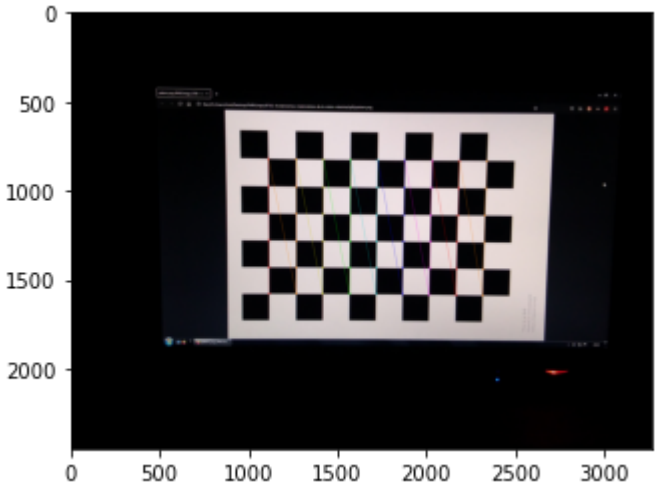

```
Testing 20 iterations...
Results for 20 iterations:
[[[1112.9852  1577.4095 ]]

 [[1112.3035  1429.0598 ]]

 [[1112.7931  1278.8617 ]]

 [[1110.981   1128.4026 ]]

 [[1109.8995   975.3975 ]]

 [[1107.9417   820.7854 ]]

 [[1263.5817  1578.1587 ]]

 [[1264.8599  1429.3958 ]]

 [[1264.5221  1280.7739 ]]

 [[1264.4453  1129.9733 ]]

 [[1263.485    977.67896]]

 [[1263.0122   823.1131 ]]

 [[1414.7375  1578.6123 ]]

 [[1415.1323  1430.8042 ]]

 [[1415.5928  1281.9642 ]]

 [[1416.4503  1132.0125 ]]

 [[1417.1158   979.54913]]

 [[1416.609    825.8532 ]]

 [[1563.3059  1579.4685 ]]

 [[1565.6012  1431.3553 ]]

 [[1566.2306  1282.9388 ]]

 [[1568.3417  1132.8981 ]]

 [[1568.8752   981.83936]]

 [[1570.6351   827.4061 ]]

 [[1713.266   1580.7281 ]]
```

```
[[1714.5482  1433.1355 ]]
[[1717.2826  1284.1808 ]]
[[1718.5841  1134.8912 ]]
[[1721.0111   982.8544 ]]
[[1722.725    829.75616]]
[[1862.0637  1582.4506 ]]
[[1864.8058  1434.1605 ]]
[[1866.3286  1285.5621 ]]
[[1869.4536  1136.0074 ]]
[[1872.2546   984.8714 ]]
[[1875.6019   830.74524]]
[[2012.142   1583.6978 ]]
[[2013.9513  1436.0046 ]]
[[2017.5585  1287.1648 ]]
[[2020.3809  1137.1628 ]]
[[2024.4387   985.7778 ]]
[[2028.6757   831.9389 ]]
[[2161.7676  1586.3134 ]]
[[2164.9753  1437.5162 ]]
[[2168.592   1288.9746 ]]
[[2173.2703  1138.4323 ]]
[[2177.5432   987.     ]]
[[2182.7153   832.4078 ]]
[[2311.959   1587.7694 ]]
[[2315.6448  1439.5238 ]]
[[2320.594   1290.3795 ]]
[[2325.      1140.0864 ]]
[[2330.7285   987.6259 ]]
[[2336.4985   833.5573 ]]]
```




```
Testing 25 iterations...
Results for 25 iterations:
[[[1112.9852  1577.4095 ]]

 [[1112.3035  1429.0598 ]]

 [[1112.7931  1278.8617 ]]

 [[1110.981   1128.4026 ]]

 [[1109.8995   975.3975 ]]

 [[1107.9417   820.7854 ]]

 [[1263.5817  1578.1587 ]]

 [[1264.8599  1429.3958 ]]

 [[1264.5221  1280.7739 ]]

 [[1264.4453  1129.9733 ]]

 [[1263.485    977.67896]]

 [[1263.0122   823.1131 ]]

 [[1414.7375  1578.6123 ]]

 [[1415.1323  1430.8042 ]]

 [[1415.5928  1281.9642 ]]

 [[1416.4503  1132.0125 ]]

 [[1417.1158   979.54913]]

 [[1416.609    825.8532 ]]

 [[1563.3059  1579.4685 ]]

 [[1565.6012  1431.3553 ]]

 [[1566.2306  1282.9388 ]]

 [[1568.3417  1132.8981 ]]

 [[1568.8752   981.83936]]

 [[1570.6351   827.4061 ]]

 [[1713.266   1580.7281 ]]
```

```
[[1714.5482  1433.1355 ]]
[[1717.2826  1284.1808 ]]
[[1718.5841  1134.8912 ]]
[[1721.0111   982.8544 ]]
[[1722.725    829.75616]]
[[1862.0637  1582.4506 ]]
[[1864.8058  1434.1605 ]]
[[1866.3286  1285.5621 ]]
[[1869.4536  1136.0074 ]]
[[1872.2546   984.8714 ]]
[[1875.6019   830.74524]]
[[2012.142   1583.6978 ]]
[[2013.9513  1436.0046 ]]
[[2017.5585  1287.1648 ]]
[[2020.3809  1137.1628 ]]
[[2024.4387   985.7778 ]]
[[2028.6757   831.9389 ]]
[[2161.7676  1586.3134 ]]
[[2164.9753  1437.5162 ]]
[[2168.592   1288.9746 ]]
[[2173.2703  1138.4323 ]]
[[2177.5432   987.     ]]
[[2182.7153   832.4078 ]]
[[2311.959   1587.7694 ]]
[[2315.6448  1439.5238 ]]
[[2320.594   1290.3795 ]]
[[2325.      1140.0864 ]]
[[2330.7285   987.6259 ]]
[[2336.4985   833.5573 ]]]
```

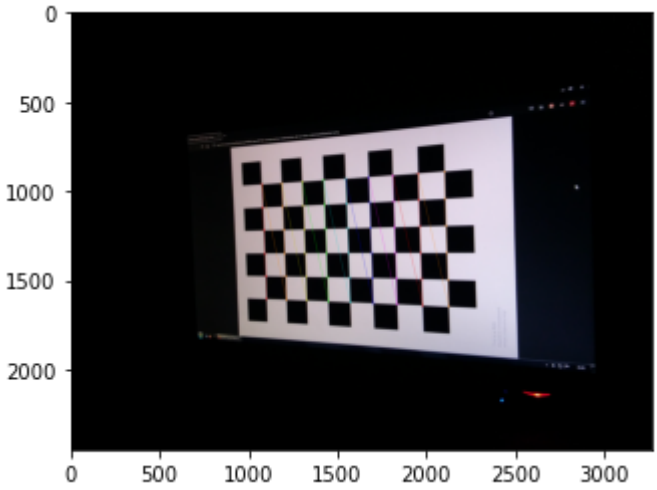


No se observan diferencias a simple vista lo que se corrobora al comparar los puntos encontrados. Se utilizarán **5** iteraciones.

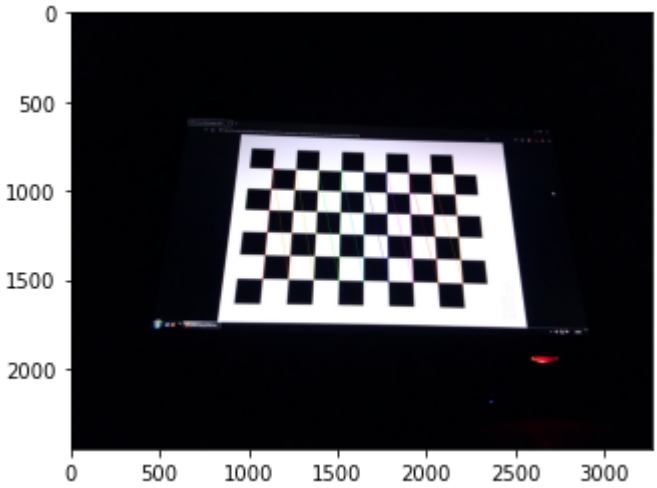
Identificación de puntos imagen y puntos objeto

In [6]:
```
max_iterations = 5
epsilon = 0.001
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, max_iterations,
world_points = list()
image_points = list()
chessborard_world_points = np.zeros((np.prod(chessboard_size), 3), dtype=np.f
chessborard_world_points[:, :2] = np.mgrid[0:chessboard_size[0], 0:chessboarc

for filename in filenames:
    print("processing image {0}...".format(filename))
    original_image = cv.imread(filename)
    grayscale_image = cv.cvtColor(original_image, cv.COLOR_BGR2GRAY)
    ret, corners = cv.findChessboardCorners(grayscale_image, chessboard_size,
    if ret:
        world_points.append(chessborard_world_points)
        corners_subpix = cv.cornerSubPix(grayscale_image, corners, (5, 5), (-
        image_points.append(corners_subpix)
        cv.drawChessboardCorners(original_image, chessboard_size, corners_sub
        plt.figure()
        plt.imshow(cv.cvtColor(original_image, cv.COLOR_BGR2RGB))
        plt.show()
```

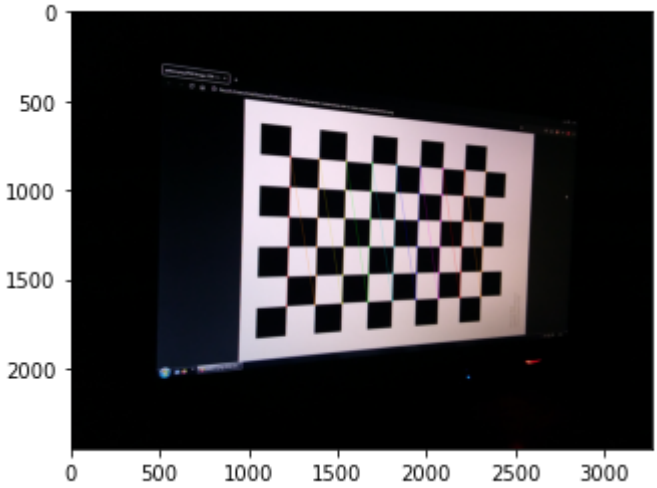
```
processing image ./pictures/06-center-right.jpg...
```



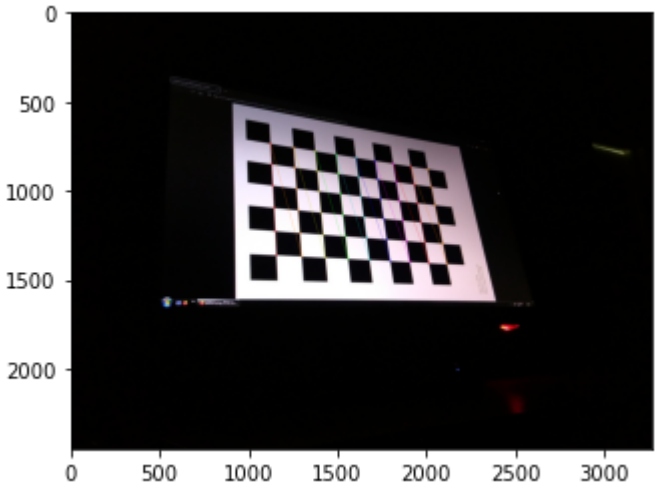

```
processing image ./pictures/08-bottom.jpg...
```



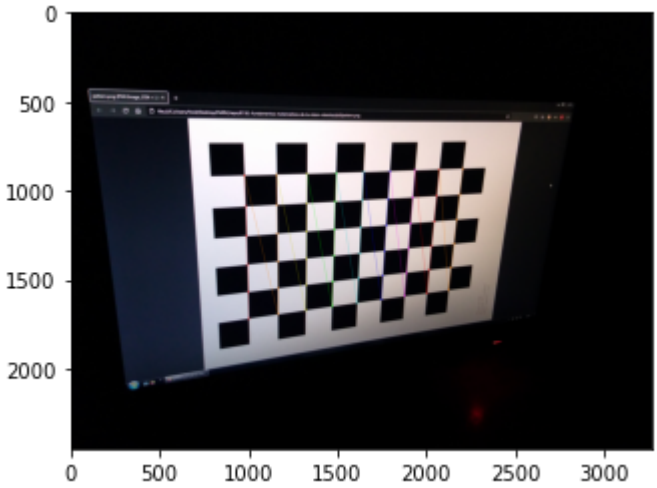

```
processing image ./pictures/04-center-left.jpg...
```



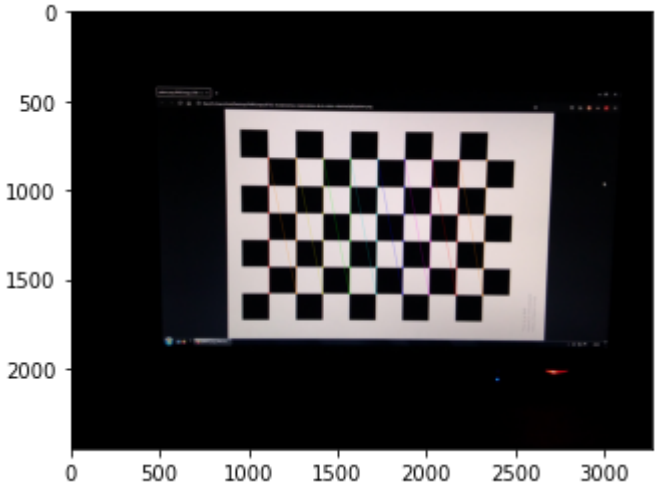

```
processing image ./pictures/07-bottom-left.jpg...
```



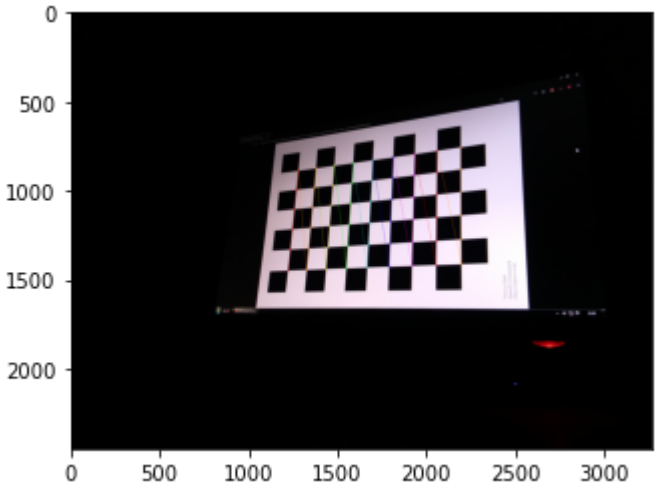

```
processing image ./pictures/01-top-left.jpg...
```



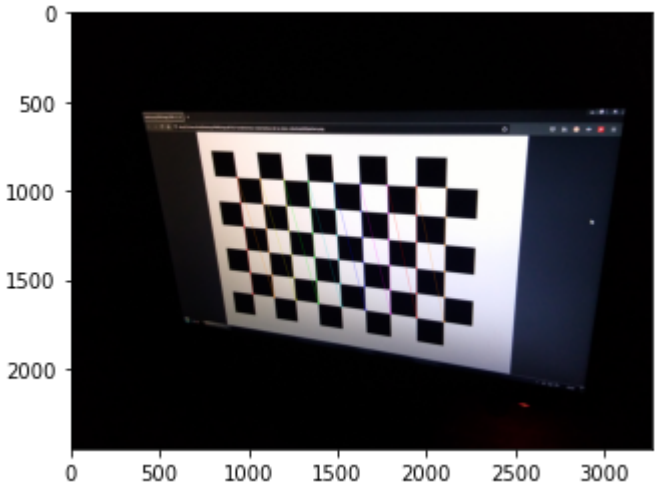

```
processing image ./pictures/05-center.jpg...
```



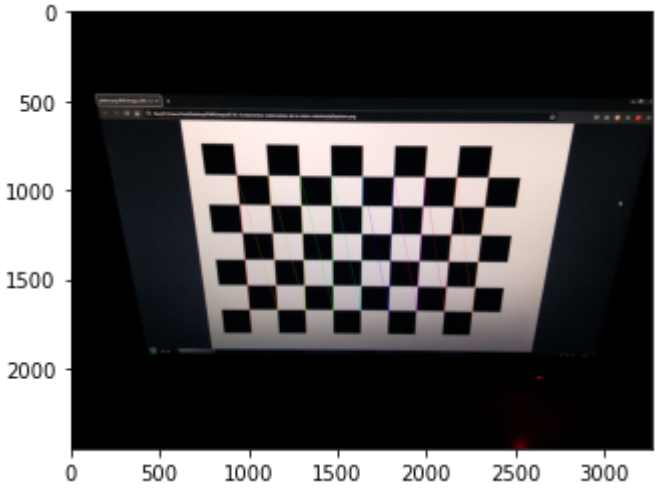

```
processing image ./pictures/09-bottom-right.jpg...
```




```
processing image ./pictures/03-top-right.jpg...
```




```
processing image ./pictures/02-top.jpg...
```



Se identificaron correctamente los puntos del tablero de ajedrez, ahora se procederá a la calibración de la cámara.

Calibración de cámara

In [7]:
```
image = cv.imread(filenames[0])
grayscale_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
image_height, image_width = grayscale_image.shape
ret, camera_matrix, distortion_coefficients, rvecs, tvecs = cv.calibrateCamer
    world_points, image_points,(image_width, image_height), None, None)

print("Camera Matrix: \n{0}".format(camera_matrix))
print("Distortion Coefficients: \n{0}".format(distortion_coefficients))
```

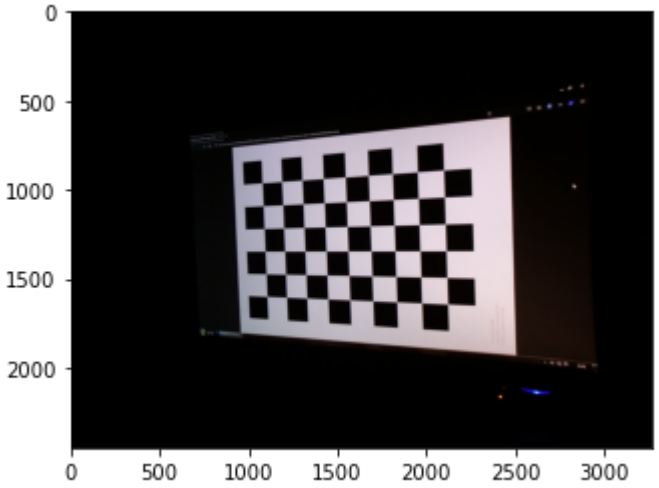
```
Camera Matrix:
[[2.59123832e+03 0.00000000e+00 1.61087290e+03]
 [0.00000000e+00 2.58703920e+03 1.20577695e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Distortion Coefficients:
[[ 0.15290914 -0.35897228 -0.00065981 -0.00110709 -0.3307477 ]]
```

Rectificación de imágenes

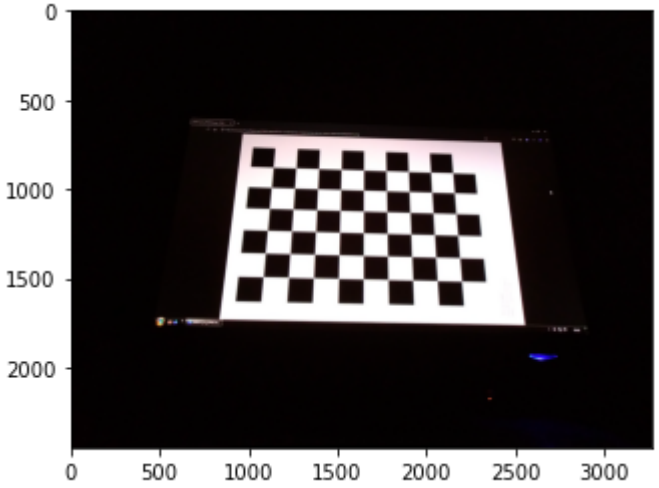
In [8]:
```
for filename in filenames:
    print("processing image {0}...".format(filename))
    image = cv.imread(filename)
    undistorted_image = cv.undistort(image, camera_matrix, distortion_coeffic
    plt.figure()
```

```
    plt.imshow(undistorted_image)
    plt.show()
```

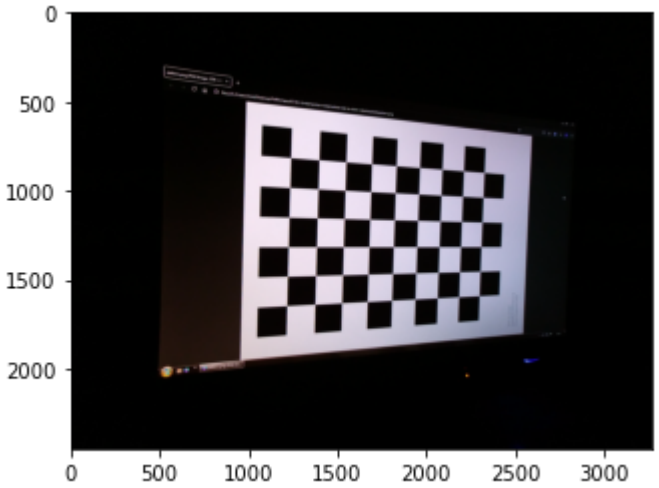
```
processing image ./pictures/06-center-right.jpg...
```



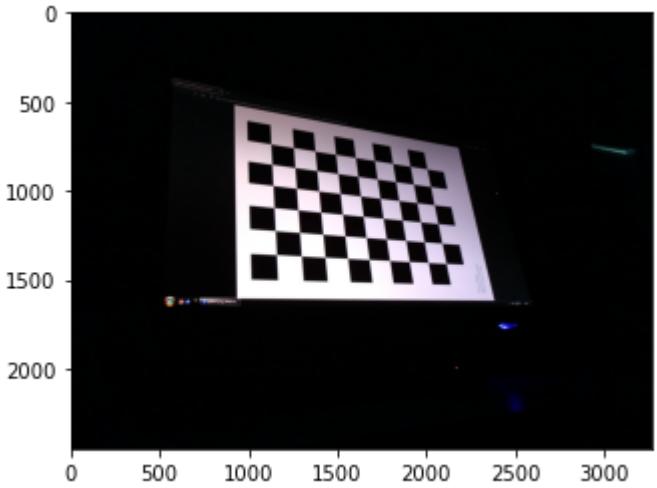

```
processing image ./pictures/08-bottom.jpg...
```



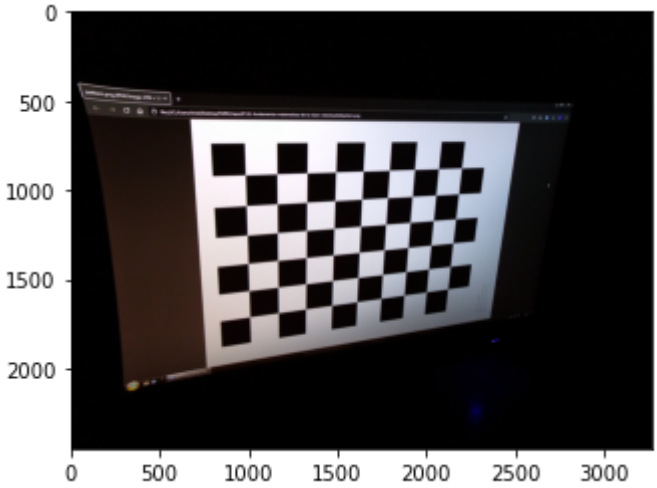

```
processing image ./pictures/04-center-left.jpg...
```



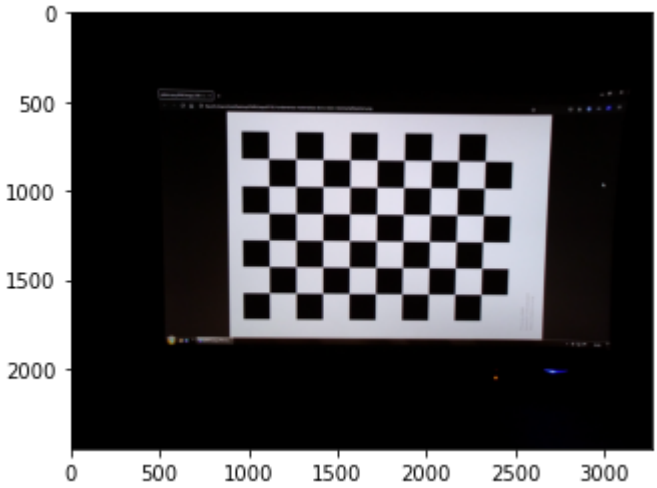

```
processing image ./pictures/07-bottom-left.jpg...
```



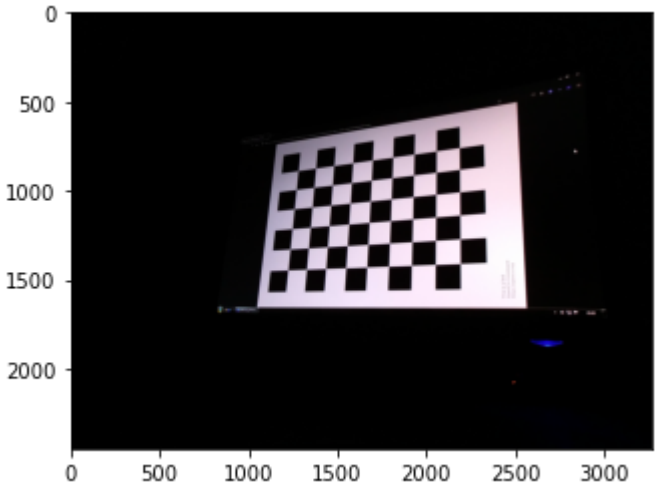

```
processing image ./pictures/01-top-left.jpg...
```



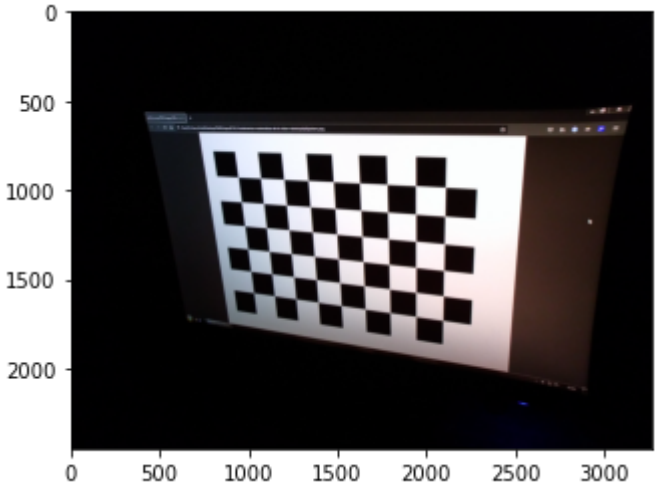

```
processing image ./pictures/05-center.jpg...
```



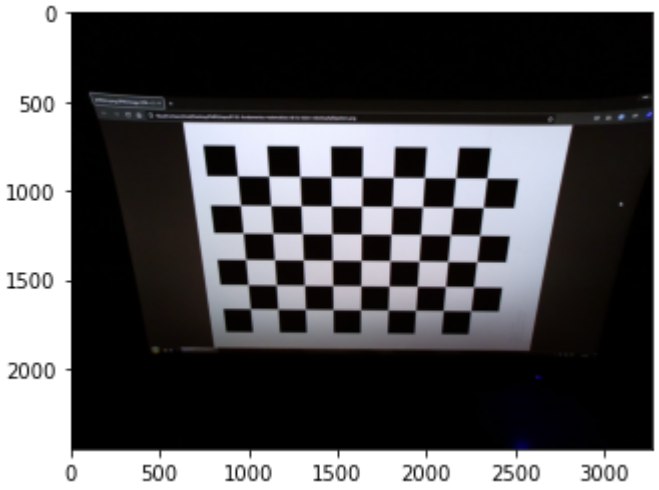

```
processing image ./pictures/09-bottom-right.jpg...
```




```
processing image ./pictures/03-top-right.jpg...
```




```
processing image ./pictures/02-top.jpg...
```



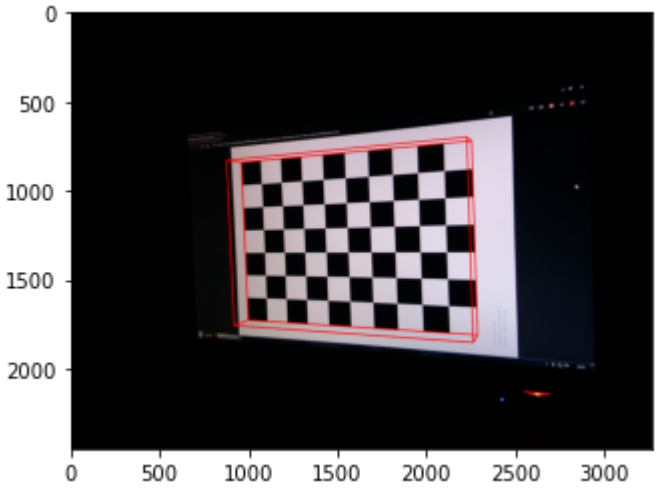
Chequeo de los resultados

Se dibujará un primsma que abarque el tablero de ajedrez para verificar que los resultados son correctos.

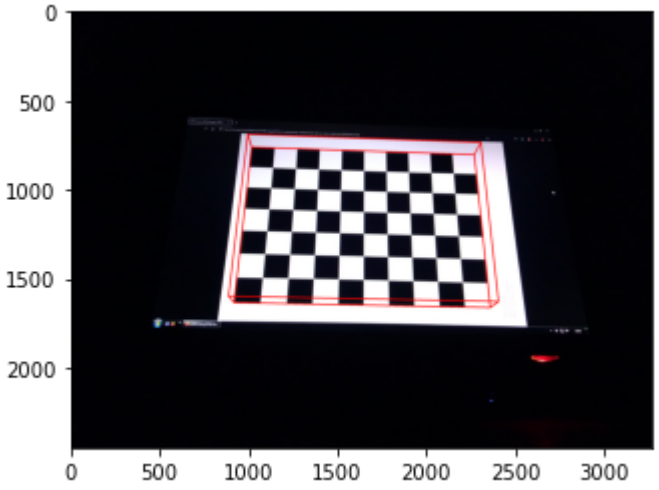
In [9]:
```
world_points = np.float32([[6, -1, 0], [6, 9, 0], [-1, 9, 0], [-1, -1, 0],
                           [6, -1, -1], [6, 9, -1], [-1, 9, -1], [-1, -1, -1]])
color = (0, 0, 255)
line_width = 5
```

```
for i in range(len(filenames)):
    print("processing image {0}...".format(filenames[i]))
    image = cv.imread(filenames[i])
    image_points = cv.projectPoints(world_points, rvecs[i], tvecs[i], camera_
                                    distortion_coefficients)[0]
    image_points = image_points[:, 0, :]
    cv.line(image, tuple(image_points[0]), tuple(image_points[1]), color, lir
    cv.line(image, tuple(image_points[1]), tuple(image_points[2]), color, lir
    cv.line(image, tuple(image_points[2]), tuple(image_points[3]), color, lir
    cv.line(image, tuple(image_points[3]), tuple(image_points[0]), color, lir
    cv.line(image, tuple(image_points[0]), tuple(image_points[4]), color, lir
    cv.line(image, tuple(image_points[1]), tuple(image_points[5]), color, lir
    cv.line(image, tuple(image_points[2]), tuple(image_points[6]), color, lir
    cv.line(image, tuple(image_points[3]), tuple(image_points[7]), color, lir
    cv.line(image, tuple(image_points[4]), tuple(image_points[5]), color, lir
    cv.line(image, tuple(image_points[5]), tuple(image_points[6]), color, lir
    cv.line(image, tuple(image_points[6]), tuple(image_points[7]), color, lir
    cv.line(image, tuple(image_points[7]), tuple(image_points[4]), color, lir

    plt.figure()
    plt.imshow(image[..., ::-1])
    plt.show()
```

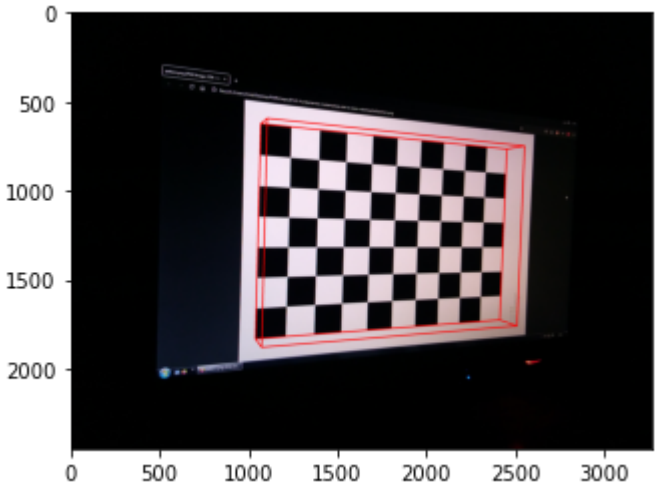
```
processing image ./pictures/06-center-right.jpg...
```



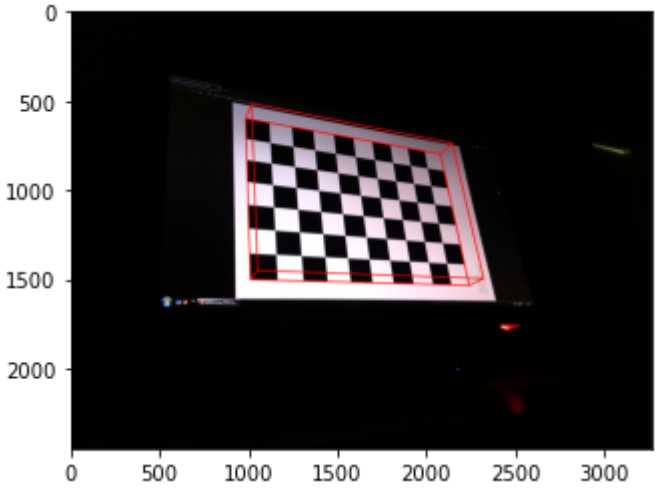

```
processing image ./pictures/08-bottom.jpg...
```



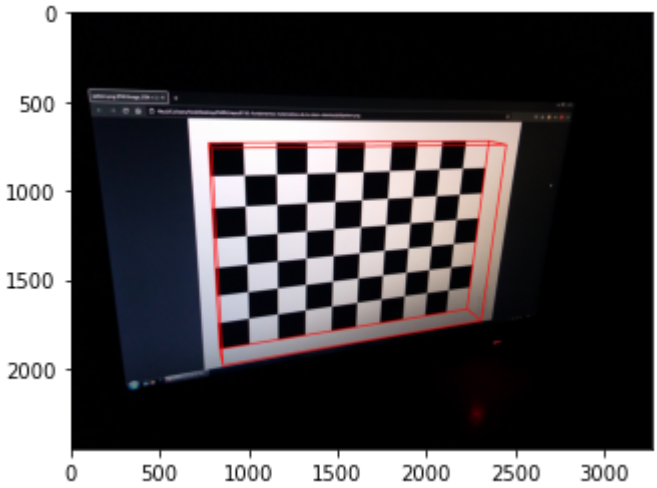

```
processing image ./pictures/04-center-left.jpg...
```



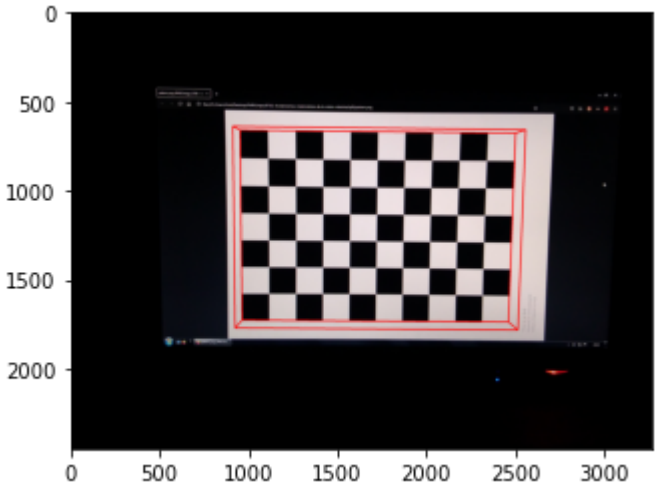

```
processing image ./pictures/07-bottom-left.jpg...
```



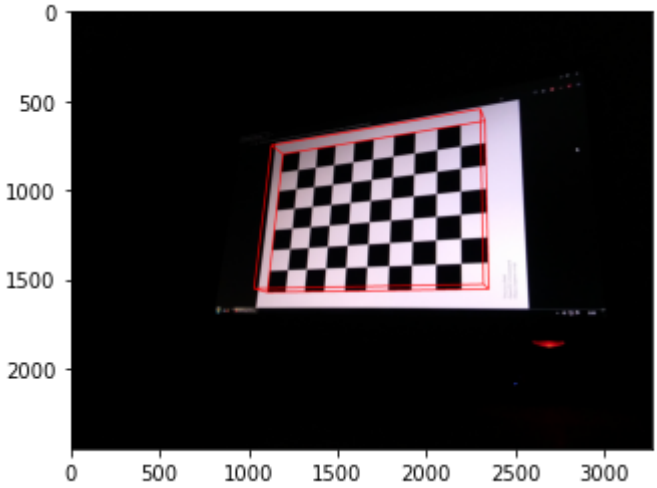

```
processing image ./pictures/01-top-left.jpg...
```



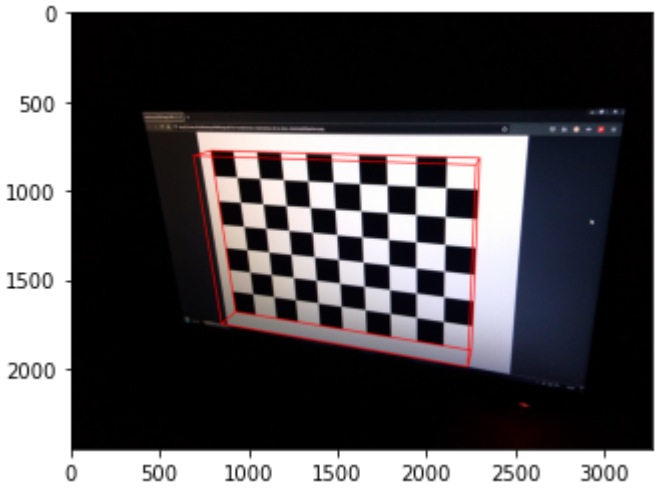

```
processing image ./pictures/05-center.jpg...
```



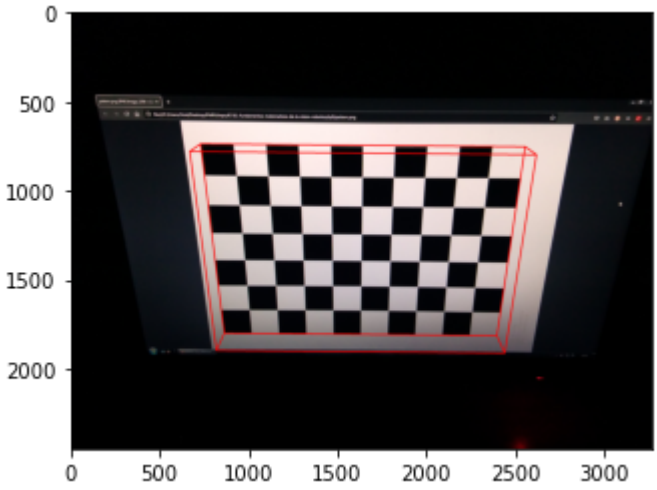

```
processing image ./pictures/09-bottom-right.jpg...
```




```
processing image ./pictures/03-top-right.jpg...
```




```
processing image ./pictures/02-top.jpg...
```



Se verifica que se se dibuja el prisma correctamente.