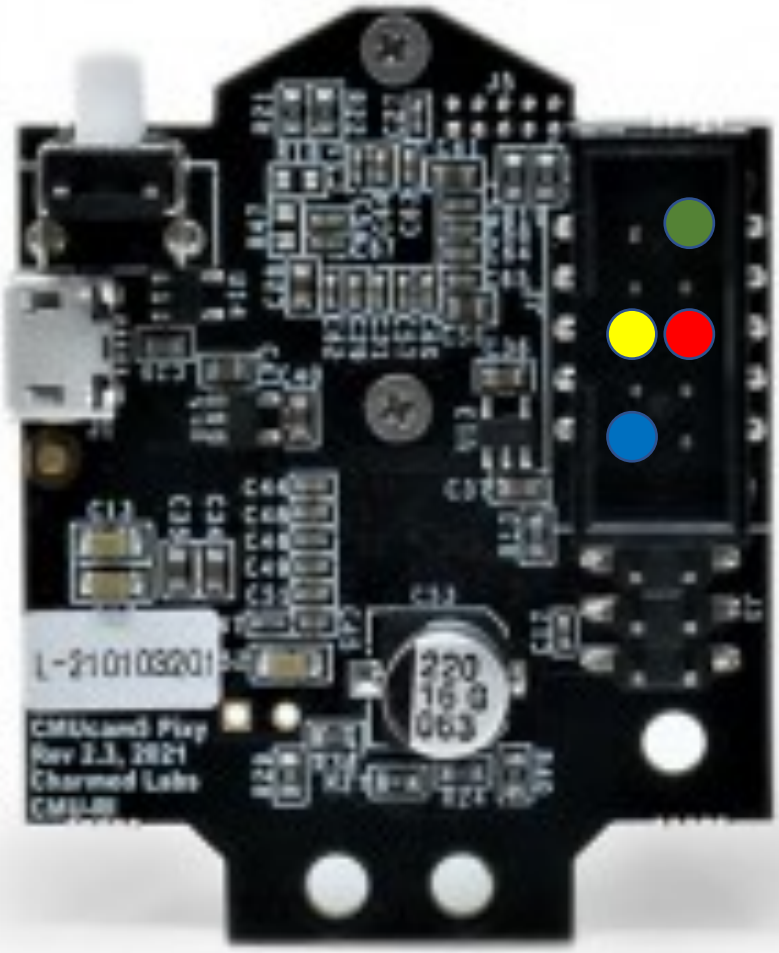
---Instructions for working with Pixy v2.1 camera---

Table of contents:

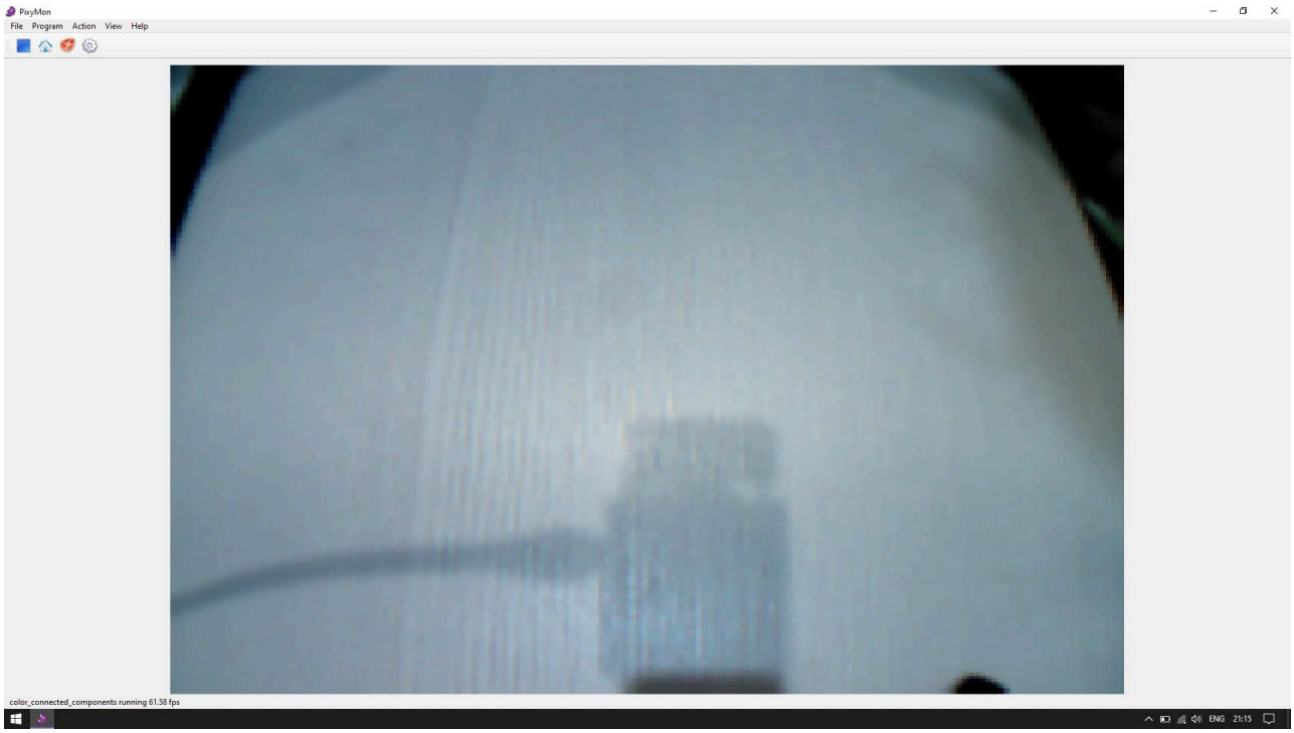
Connection

To connect the camera, we need to cut the insulation of the Ev3 wire and solder the female connectors. Next, connect them to the pins indicated in the image.
The color of the circle indicates the color of the wire to which you want to connect this pin. (green pin to green wire).

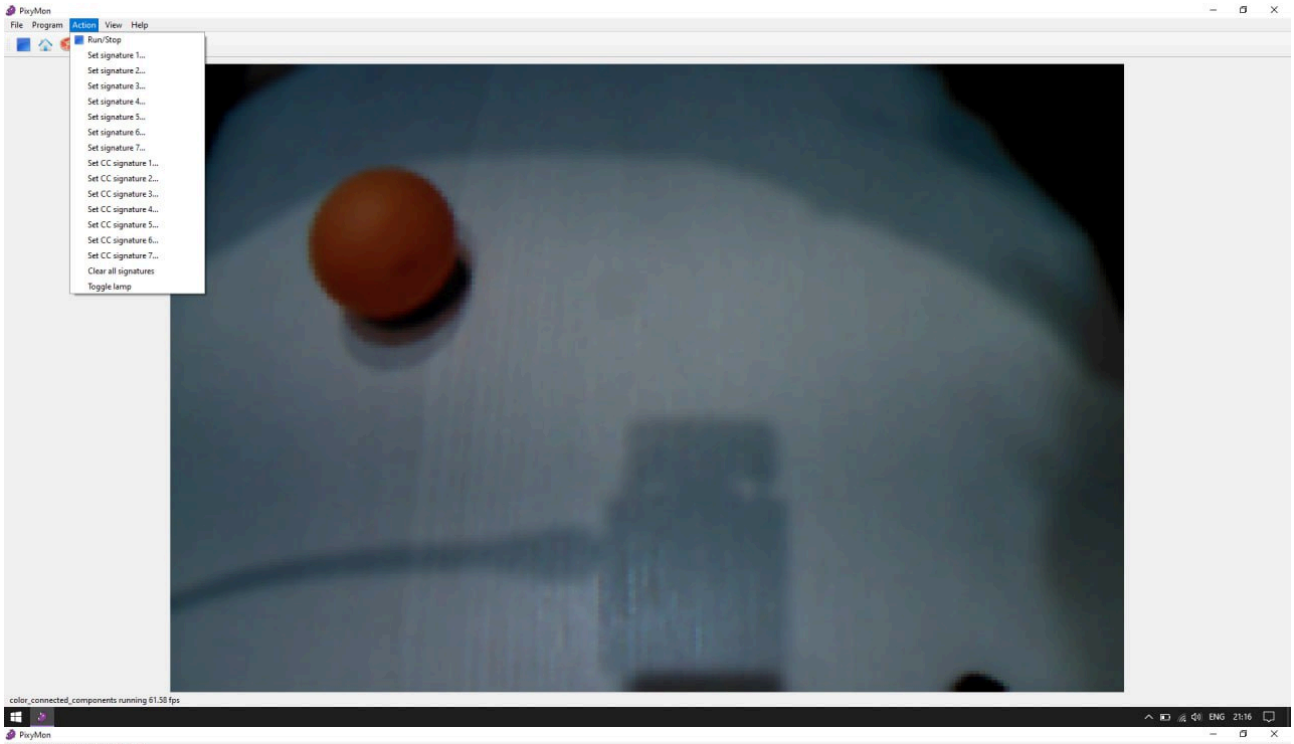


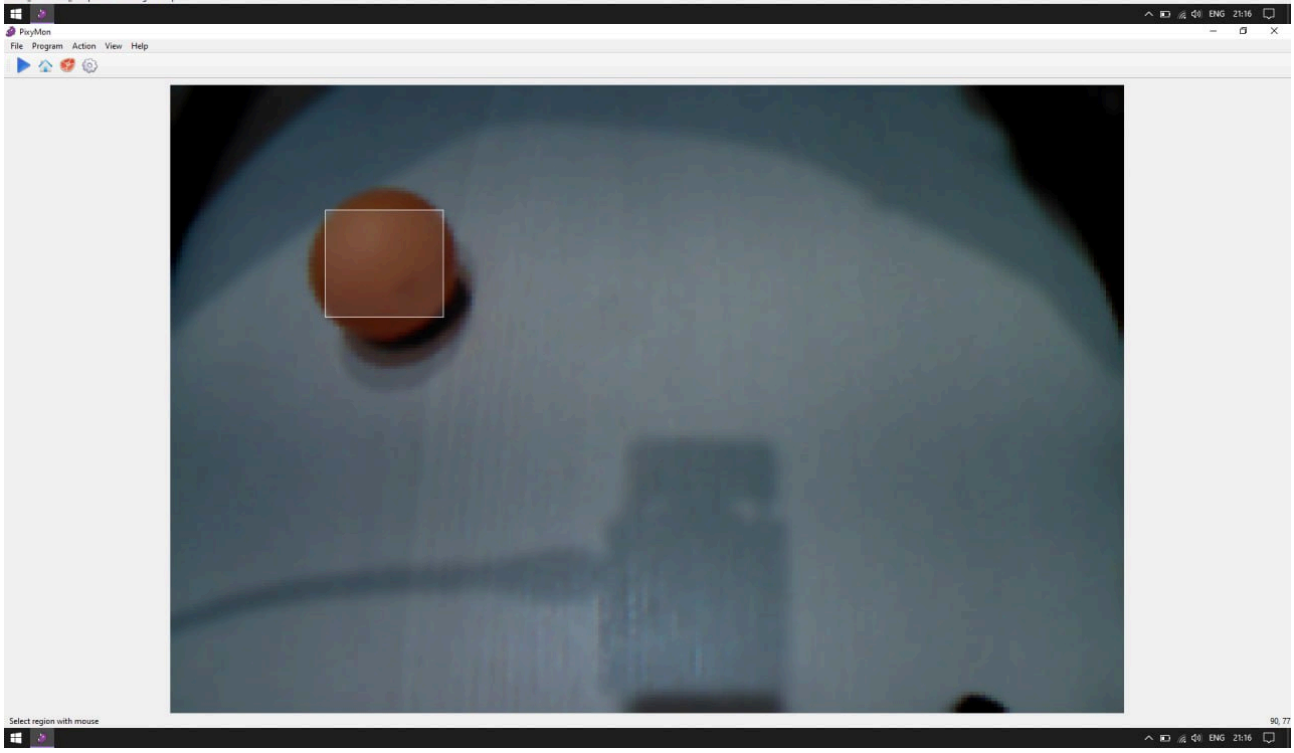
Camera setup

In this paragraph, I will tell you how to set the camera to detect an object of a certain color, in my case an orange tennis ball.
Download the PixyMon v2 program from the link (https://pixycam.com/downloads-pixy2/) and open it.
We connect the camera via microUSB wire to the computer and see our image from the camera.

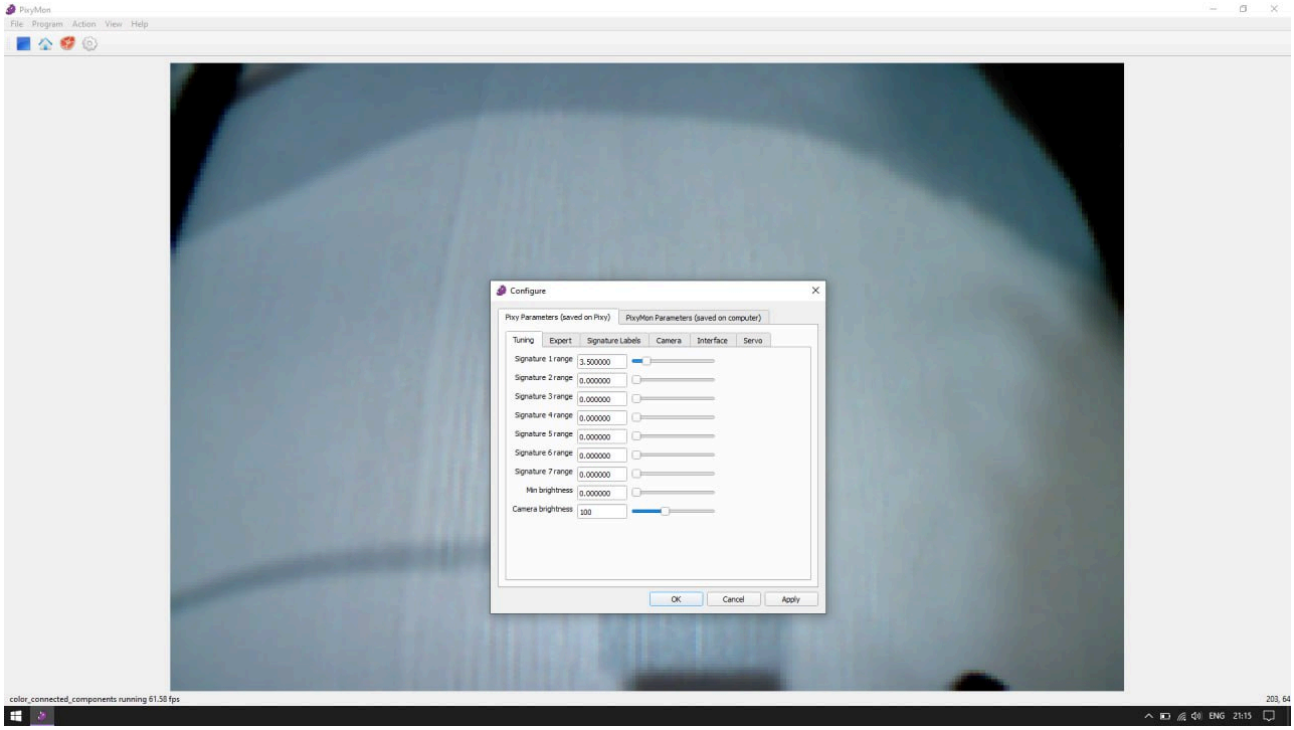


Next, we show the object to be found. Open the Action tab.
There we see functions for the object signature task. Press Set signature 1 and select the OBJECT COLOR, pay attention not to the object, but to its color. The larger the area of color you select, the more accurate the definition will be.

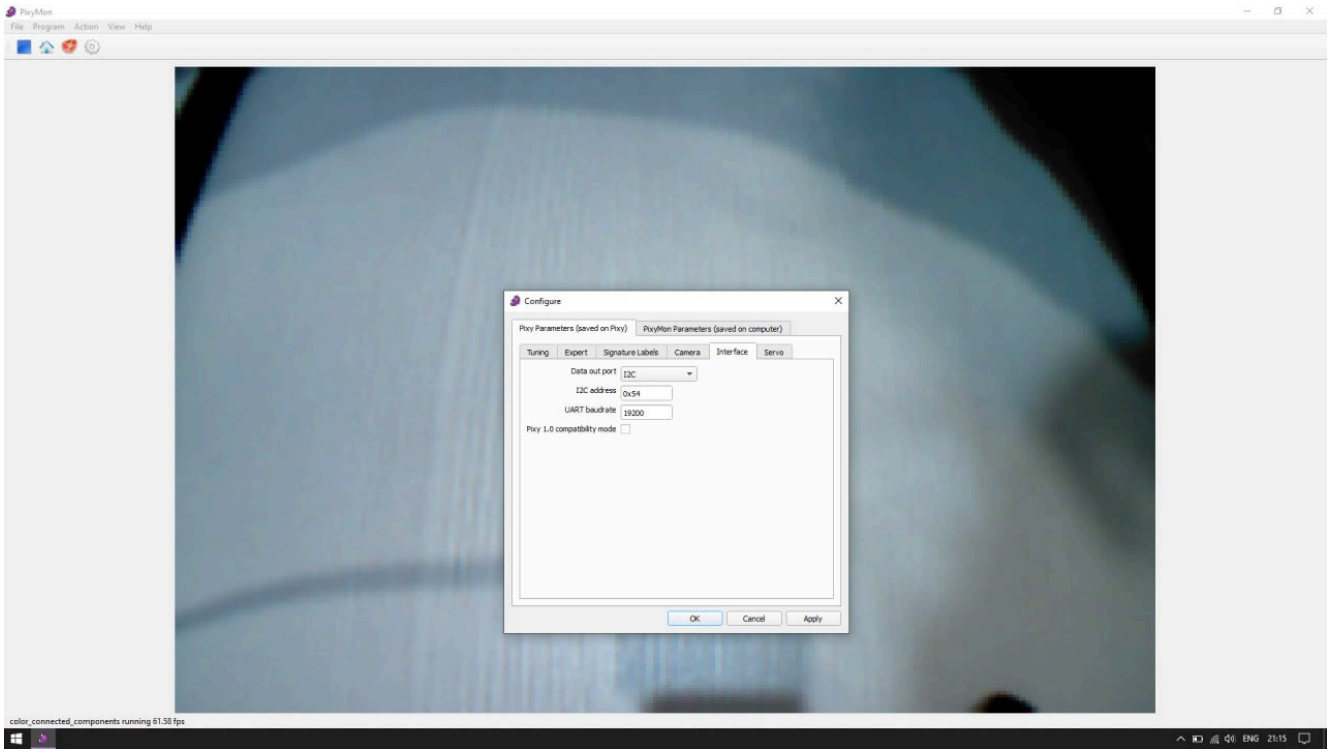




That's it, we have set up the definition of the object in the application. You can also explore the settings in the "Settings" and "Improve Definition" tab.
For example, if the camera saw too many additional objects before, then you can either reconfigure the camera or reduce the range of settings. The initial value is 3.5, it is recommended to experiment with changing this parameter yourself.



To work with the camera on ev3basic, we need to change the way the camera communicates with the computer to I2C, you can do this in the Interface - data out port tab.



Reading the readings from the camera

I will be reading in ev3basic using the clev3r programming environment. You can download clev3r from the link (https://clev3r.ru/) The code for reading looks like this: I will be reading in ev3basic using the clev3r programming environment. You can download clev3r from the link (https://clev3r.ru/) The code for reading looks like this:

At the beginning of the program, we need to set the constants:

```
I2C_adress = 84 'Камера
iCorrect = 0 'корректировка камеры
ArraySizeSend = 32
valuesSize = 19
ArraySend = Vector.Init(ArraySizeSend,0)
valuesCamera = Vector.Init(valuesSize, 0) 'массив данных с камеры
ArraySend[0] = 174
ArraySend[1] = 193
ArraySend[2] = 32
ArraySend[3] = 2
ArraySend[4] = 1
ArraySend[5] = 1
tvalues[0] = 0 'измененный массив
sti = 0 'коррекция массива
xcorrect = 0 'корректировка массива
```

Function for determining object coordinates:

```
Sub getCoordsBall
  While "true"
    iCorrect = 0

    valuesCamera = Sensor.CommunicateI2C(cameraPort,I2C_adress, ArraySizeSend, valuesSize, ArraySend)

    xcorrect=0
    iCorrect = 0
    While 16>= iCorrect + 3
      If valuesCamera[iCorrect] = 85 And valuesCamera[iCorrect+1] = 170 And valuesCamera[iCorrect+2] =85
And valuesCamera[iCorrect+3] = 170 Then
        xcorrect = iCorrect
        iCorrect = 99
      EndIf
      iCorrect = iCorrect + 1
    EndWhile

    If iCorrect <> 100 Then
      If valuesCamera[13] = 85 And valuesCamera[14] = 170 And valuesCamera[15] = 85 And valuesCamera[0]
= 170 Then
        xcorrect = 13
      EndIf
      If valuesCamera[14] = 85 And valuesCamera[15] = 170 And valuesCamera[0] = 85 And valuesCamera[1] =
170 Then
        xcorrect = 14
      EndIf
      If valuesCamera[15] = 85 And valuesCamera[0] = 170 And valuesCamera[1] = 85 And valuesCamera[2] =
170 Then
        xcorrect = 15
      EndIf
    EndIf

    If xcorrect <> 0 Then
      tvalues = Sensor.CommunicateI2C(cameraPort,I2C_adress, ArraySizeSend, valuesSize, ArraySend)
      iCorrect = xcorrect
      sti=  0
      While iCorrect <= 15
        valuesCamera[sti] = valuesCamera[iCorrect]
        iCorrect = iCorrect + 1
        sti = sti + 1
      EndWhile
      iCorrect = 0

      sti = 16 - xcorrect
      While iCorrect < xcorrect
        valuesCamera[sti] = tvalues[iCorrect]
        iCorrect = iCorrect + 1
        sti = sti + 1
      EndWhile
    EndIf

    centerX = valuesCamera[8] + valuesCamera[9] * 255
    centerY = valuesCamera[10] + valuesCamera[11] * 255
    width = valuesCamera[12] + valuesCamera[13] * 255
    hight = valuesCamera[14] + valuesCamera[15] * 255
  EndWhile
EndSub
```

Explanation of the code

```
valuesCamera = Sensor.CommunicateI2C(cameraPort,I2C_adress, ArraySizeSend, valuesSize, ArraySend)
```

Here we get the values from the camera

```
    xcorrect=0
    iCorrect = 0
    While 16>= iCorrect + 3
      If valuesCamera[iCorrect] = 85 And valuesCamera[iCorrect+1] = 170 And valuesCamera[iCorrect+2] =85
And valuesCamera[iCorrect+3] = 170 Then
        xcorrect = iCorrect
        iCorrect = 99
      EndIf
      iCorrect = iCorrect + 1
    EndWhile

    If iCorrect <> 100 Then
      If valuesCamera[13] = 85 And valuesCamera[14] = 170 And valuesCamera[15] = 85 And valuesCamera[0] =
170 Then
        xcorrect = 13
      EndIf
      If valuesCamera[14] = 85 And valuesCamera[15] = 170 And valuesCamera[0] = 85 And valuesCamera[1] =
170 Then
        xcorrect = 14
      EndIf
      If valuesCamera[15] = 85 And valuesCamera[0] = 170 And valuesCamera[1] = 85 And valuesCamera[2] =
170 Then
        xcorrect = 15
      EndIf
    EndIf

    If xcorrect <> 0 Then
      tvalues = Sensor.CommunicateI2C(cameraPort,I2C_adress, ArraySizeSend, valuesSize, ArraySend)
      iCorrect = xcorrect
      sti=  0
      While iCorrect <= 15
        valuesCamera[sti] = valuesCamera[iCorrect]
        iCorrect = iCorrect + 1
        sti = sti + 1
      EndWhile
      iCorrect = 0

      sti = 16 - xcorrect
      While iCorrect < xcorrect
        valuesCamera[sti] = tvalues[iCorrect]
        iCorrect = iCorrect + 1
        sti = sti + 1
      EndWhile
    EndIf
```

This is an input array adjustment. Sometimes the camera shifts the array by several indexes, so this code corrects the input array.

```
    centerX = valuesCamera[8] + valuesCamera[9] * 255
    centerY = valuesCamera[10] + valuesCamera[11] * 255
    width = valuesCamera[12] + valuesCamera[13] * 255
    hight = valuesCamera[14] + valuesCamera[15] * 255
```

We get the coordinates of the center of the object along the X and Y axes, we also get the length and width.

Use in the main program

We declare block and constants at the beginning of programs. Next, run it in a parallel loop:

```
Thread.Run = getCoordsBall
```

Everything, you can display the coordinates on the screen in this way:

```
While "true"
    LCD.StopUpdate()
    LCD.Clear()
    LCD.Write(0,50,centerX)
    LCD.Write(50,50,centerY)
    LCD.Write(0,100,width)
    LCD.Write(50,100,hight)
    LCD.Update()
EndWhile
```

Algorithm for following the ball

The algorithm consists of a normal PID motion controller, but the "error" is calculated differently. If for driving along the line we subtracted the readings of another from the readings of one sensor, then in our case we subtract the coordinate along the X axis to which we are striving from the X coordinate of the object. You must choose the coefficients yourself.

```
Thread.Run = getCoordsBall
While "True"
    err = centerX - centerNeed
    sum = sum + err - errOld
    dLine = (err - errOld)
    up = (err * kFollowingBall) + (dLine * kdFollowingBall) + (sum * kiFollowingBall)
    Motor.StartPower("B",vFollowingBall + up)
    Motor.StartPower("C",vFollowingBall - up)
    errOld = err
EndWhile
```

Outcome

Congratulations, you have learned how to work with the Pixy camera, set it up, get the coordinates of an object and follow it!

If you have any additional questions about working with the camera or you have a suggestion for supplementing the instructions with information, write to me by mail:

ilyavasiliev1@mail.ru

My channel on youtube:

https://www.youtube.com/channel/UCcH64ygbrEwDkqws_n1b7gQ

My boosty:

https://boosty.to/roadster