

# Building Machine Vision Applications using OpenMV

## Class 2: Writing our First OpenMV Application

June 9, 2020  
Jacob Beningo

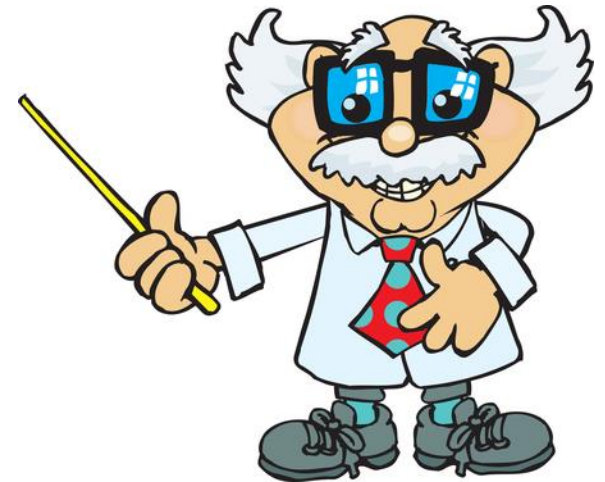
# Course Overview

## Topics:

- Introduction to Machine Vision and OpenMV
- **Writing our First OpenMV Application**
- Working with the OpenMV I/O
- Utilizing Machine Learning to Detect Objects
- Designing a Machine Vision Application

# Session Overview

- OpenMV Cam H7 Pin-Outs
- OpenMV Cam H7 LEDs
- Hello World Script
- Adjusting Image Resolution
- Adjusting pixel mode
- Detecting a Circle



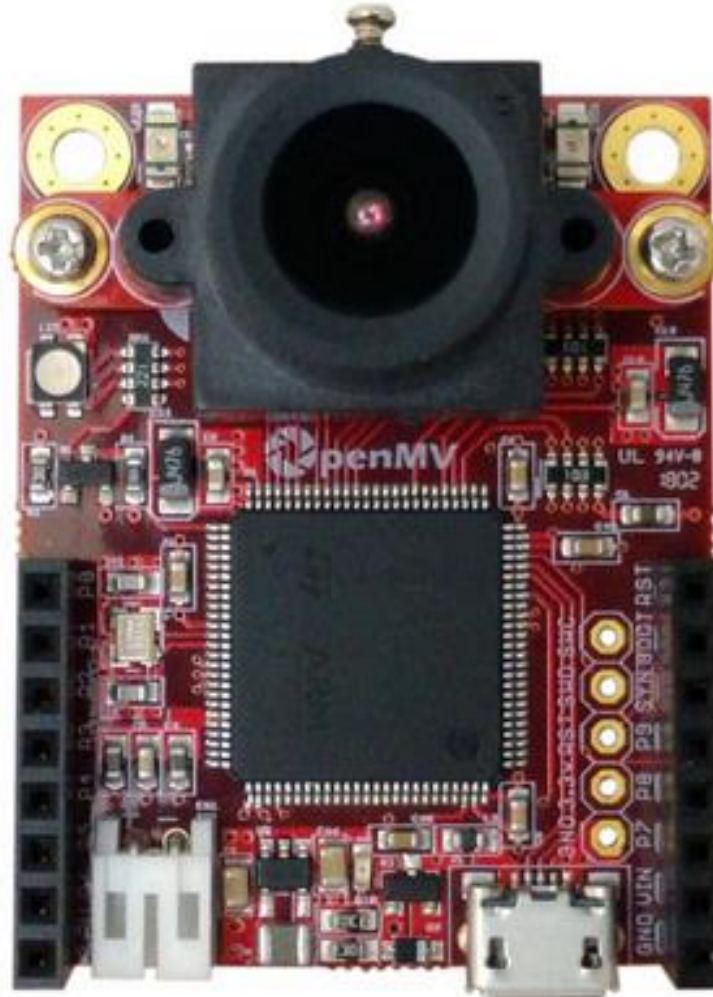
Presented by:

# OpenMV Cam H7 Pin-Out



By: Ibrahim Abdelkader & Kwabena W. Agyeman  
<https://openmv.io>

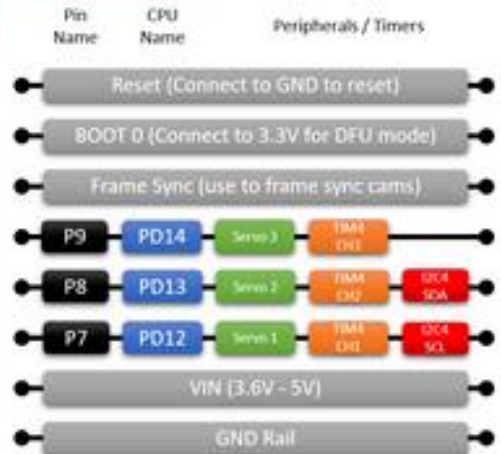
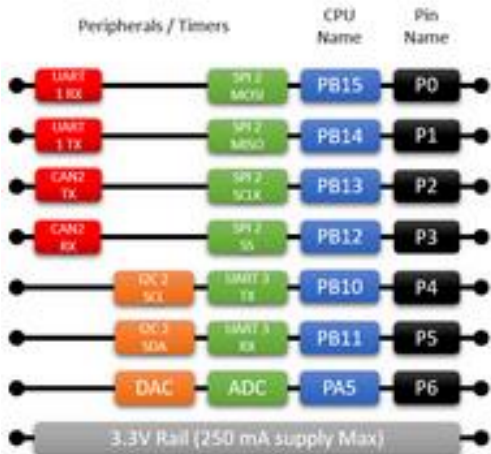
LED1 – Red  
 LED2 – Green  
 LED3 – Blue  
 LED4 – IR



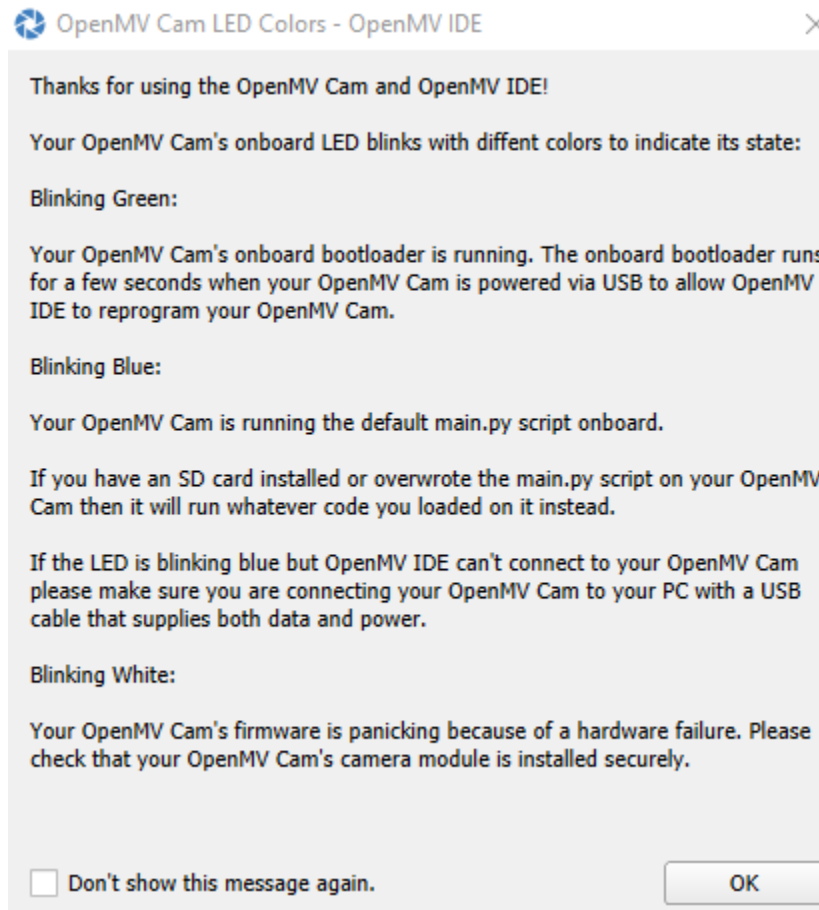
All pins are 5V tolerant<sup>1</sup> with a 3.3V output  
 All pins can sink or source up to 25 mA<sup>2</sup>  
<sup>1</sup> P6 is not 5V tolerant in ADC or DAC mode  
<sup>2</sup> Up to 120mA in total between all pins

Max current used w/o  $\mu$ SD card < 150 mA  
 Max current used w/  $\mu$ SD card < 250 mA

Micro SD Slot  
 SD < 2GB Max  
 SDHC < 32GB Max



# OpenMV Cam H7 LEDs



# “Hello World” Script

Script format follows a typical Python format:

*import ...*

*...*

*One-time initialization*

*...*

*while(True):*

*...*

# “Hello World” Script

```
# Hello World Example
```

```
#
```

```
# Welcome to the OpenMV IDE! Click on the green run arrow button below to run the script!
```

```
import sensor, image, time
```

```
sensor.reset() # Reset and initialize the sensor.
```

```
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
```

```
sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
```

```
sensor.skip_frames(time = 2000) # Wait for settings take effect.
```

```
clock = time.clock() # Create a clock object to track the FPS.
```

```
while(True):
```

```
    clock.tick() # Update the FPS clock.
```

```
    img = sensor.snapshot() # Take a picture and return the image.
```

```
    print(clock.fps()) # Note: OpenMV Cam runs about half as fast when connected  
# to the IDE. The FPS should increase once disconnected.
```

# “Hello World” Script

```
helloworld_1.py  |  | X |
1  # Hello World Example
2  #
3  # Welcome to the OpenMV IDE! Click on the green run arrow button below to run the script!
4
5  import sensor, image, time
6
7  sensor.reset() # Reset and initialize the sensor.
8  sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
9  sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
10 sensor.skip_frames(5) # Wait for settings take effect.
11 clock = time.clock() # Create a clock object to track the FPS.
12
13 while True:
14     # Update the FPS clock.
15     # Take a picture and return the image.
16     # Note: OpenMV Cam runs about half as fast when connected
17     # to the IDE. The FPS should increase once disconnected.
18     image = sensor.snapshot()
```

**sensor — camera sensor**

The sensor module is used for taking pictures.

Example usage:

```
import sensor

# Setup camera.
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames()

# Take pictures.
while True:
    sensor.snapshot()
```



# “Hello World” Script

The screenshot shows the OpenMV IDE interface. The main window displays a Python script named 'helloworld\_1.py' with the following code:

```
1 # Hello World Example
2 #
3 # Welcome to the OpenMV IDE! Click on the green
4
5 import sensor, image, time
6
7 sensor.reset() # Reset and
8 sensor.set_pixformat(sensor.RGB565) # Set pixel
9 sensor.set_framesize(sensor.QVGA) # Set frame
10 sensor.skip_frames(time = 2000) # Wait for a
11 clock = time.clock() # Create a
12
13 while(True):
14     clock.tick() # Update the
15     img = sensor.snapshot() # Take a pic
16     print(clock.fps()) # Note: Open
17 # to the IDE
18
```

The right side of the IDE shows a camera view of a person. Below the camera view are three histograms for the image, labeled 'Histogram' and 'RGB Color Space'. The histograms show the distribution of red, green, and blue colors. The red histogram has a peak around 120, the green histogram has a peak around 140, and the blue histogram has a peak around 140. The histograms are labeled 'Res (w:320, h:240)'.

The bottom of the IDE shows a Serial Terminal window with the following output:

```
53.33333
53.87931
54.39331
53.67794
54.15861
54.61394
53.95684
54.38597
54.88851
54.18719
54.57464
54.94505
```

The bottom status bar shows: Board: H7 Sensor: OV7725 Firmware Version: 3.6.2 - [latest] Serial Port: COM3 Drive: G:/ FPS: 18.5

Presented by:

# Adjusting Image Resolution

## Image Resolution with the OV7725

- Default image size is 320x240 (QVGA)
- Adjust image size to 640x480

*sensor.set\_framesize(sensor.QVGA)      # Set frame size to QVGA (320x240)*

To

*sensor.set\_framesize(sensor.VGA)      # Set frame size to QVGA (640x480)*

# Adjusting Image Resolution

The screenshot displays the OpenMV IDE interface for a file named 'helloworld\_1.py'. The code editor shows a Python script for image processing. The script includes comments and code for setting the sensor format to RGB565, setting the frame size to VGA, and capturing a snapshot. The histogram window shows the image's resolution as 640x480 pixels and provides statistical data for the Red, Green, and Blue channels.

```
1 # Hello World Example
2 #
3 # Welcome to the OpenMV IDE! Click on the green
4
5 import sensor, image, time
6
7 sensor.reset() # Reset and
8 sensor.set_pixformat(sensor.RGB565) # Set pixel
9 sensor.set_framesize(sensor.VGA) # Set frame
10 sensor.skip_frames(time = 2000) # Wait for e
11 clock = time.clock() # Create a c
12
13 while(True):
14     clock.tick() # Update the
15     img = sensor.snapshot() # Take a pic
16     print(clock.fps()) # Note: Oper
17 # to the IDE
18
```

Serial Terminal Output:

```
22.52816
22.89604
22.59215
22.95468
22.66289
22.99495
22.72727
23.04394
22.77433
23.08877
22.82879
29.41176
```

Statistics for RGB Color Space (Res: w:640, h:480):

Channel	Mean	Median	Mode	StDev
Red	115	123	132	45
Green	114	125	138	44
Blue	114	132	140	44

Additional statistics (Min, Max, LQ, UQ) are also provided for each channel.

Presented by:

# Adjusting the Grayscale

Image pixel format set to RGB565

- Default pixel format is 16-bit (RGB565)
- Adjust pixel format to 8-bit (Grayscale)

```
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565
```

To

```
sensor.set_pixformat(sensor.GRAYSCALE) # Set pixel format to (GRAYSCALE)
```

# Adjusting the Grayscale

The screenshot displays the OpenMV IDE interface. The main window shows a Python script named 'helloworld\_1.py' with the following code:

```
1 # Hello World Example
2 #
3 # Welcome to the OpenMV IDE! Click on t
4
5 import sensor, image, time
6
7 sensor.reset() # R
8 sensor.set_pixformat(sensor.GRAYSCALE) # S
9 sensor.set_framesize(sensor.VGA) # S
10 sensor.skip_frames(time = 2000) # W
11 clock = time.clock() # C
12
13 while(True):
14     clock.tick() # U
15     img = sensor.snapshot() # T
16     print(clock.fps()) # N
17 # t
18
```

The right side of the IDE shows a live video feed of a person's face in grayscale. Below the video feed is a histogram titled 'Histogram' in 'RGB Color Space' with a resolution of 'Res (w:640, h:480)'. The histogram shows three distributions for the Red, Green, and Blue channels. The Red channel is shown in red, the Green channel in green, and the Blue channel in blue. Each channel has a corresponding table of statistics:

Channel	Mean	Median	Mode	StDev	Min	Max	LQ	UQ
Red	125	140	148	46	0	255	115	156
Green	125	142	154	46	0	255	117	154
Blue	125	140	148	46	0	255	115	156

The bottom of the IDE shows a 'Serial Terminal' window with the following output:

```
40.0
38.86926
37.97468
39.03904
38.25137
39.16449
38.46154
39.26097
38.62661
39.33747
38.75969
39.39962
```

The status bar at the bottom indicates: Board: H7, Sensor: OV7725, Firmware Version: 3.6.2 - [ latest ], Serial Port: COM3, Drive: G:/, FPS: 18.9.

Presented by:

# Detecting a Circle

The screenshot shows a software interface with a menu on the left, a code editor in the center, and a camera view on the right. The menu is open to 'Examples', which is further expanded to 'Feature-Detection', where 'find\_circles.py' is highlighted with a red circle. The code editor shows Python code for a while loop that captures a sensor snapshot and processes it. The camera view shows a green circular object with a red outline.

```
15 while (True):  
16     sensor.snapshot()  
17     while (True):  
18         clock.tick()  
19         img = sensor.snapshot()  
20         # Circle objects have  
21         # center, radius, and area
```

how to find circles in the image using the Hough Transform  
le\_Hough\_Transform  
find circles which are com  
of the image/roi are ignor  
is faster  
edges.py  
find\_circles.py



# Detecting a Circle

```
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.RGB565) # grayscale is faster
sensor.set_framesize(sensor.QQVGA)
sensor.skip_frames(time = 2000)
clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot().lens_corr(1.8)

    for c in img.find_circles(threshold = 2000, x_margin = 10, y_margin = 10, r_margin = 10,
        r_min = 2, r_max = 100, r_step = 2):
        img.draw_circle(c.x(), c.y(), c.r(), color = (255, 0, 0))
        print(c)

print("FPS %f" % clock.fps())
```

# Detecting a Circle

```
import sensor, image, time, pyb
```

```
sensor.reset()
```

```
sensor.set_pixformat(sensor.RGB565) # grayscale is faster
```

```
sensor.set_framesize(sensor.QQVGA)
```

```
sensor.skip_frames(time = 2000)
```

```
clock = time.clock()
```

```
ledRed = pyb.LED(1)
```

```
while(True):
```

```
    clock.tick()
```

```
    img = sensor.snapshot().lens_corr(1.8)
```

```
    for c in img.find_circles(threshold = 2000, x_margin = 10, y_margin = 10, r_margin = 10,  
        r_min = 2, r_max = 100, r_step = 2):
```

```
        img.draw_circle(c.x(), c.y(), c.r(), color = (255, 0, 0))
```

```
        ledRed.on()
```

```
        print("Circle Detected!")
```

```
        pyb.delay(500)
```

```
        ledRed.off()
```



