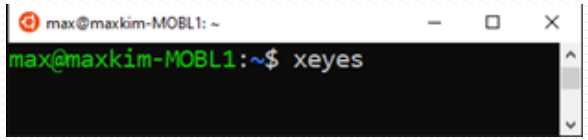
AI TRAINING SERIES

SMART FACTORY

DEMO & HANDS-ON

Pre-requisite

- Let's check WSL



- Install QT5
  - $ sudo apt install qt5-default
- Install Python packages
  - $ sudo python3 -m pip install --upgrade pip
  - $ sudo python3 -m pip install scikit-build cmake opencv-python opencv-contrib-python numpy pillow pyserial pyqt5
- Install iotdemo package
  - $ python3 -m pip install --user dist/iotdemo-0.0.1-py2.py3-none-any.whl
- Set PATH
  - $ echo "$PATH=$PATH:~.local/bin" >> ~/.bashrc
  - $ source ~/.bashrc
- Copy resource files

CONTENTS

DEMO SYSTEM

NUC
NUC10i7FN

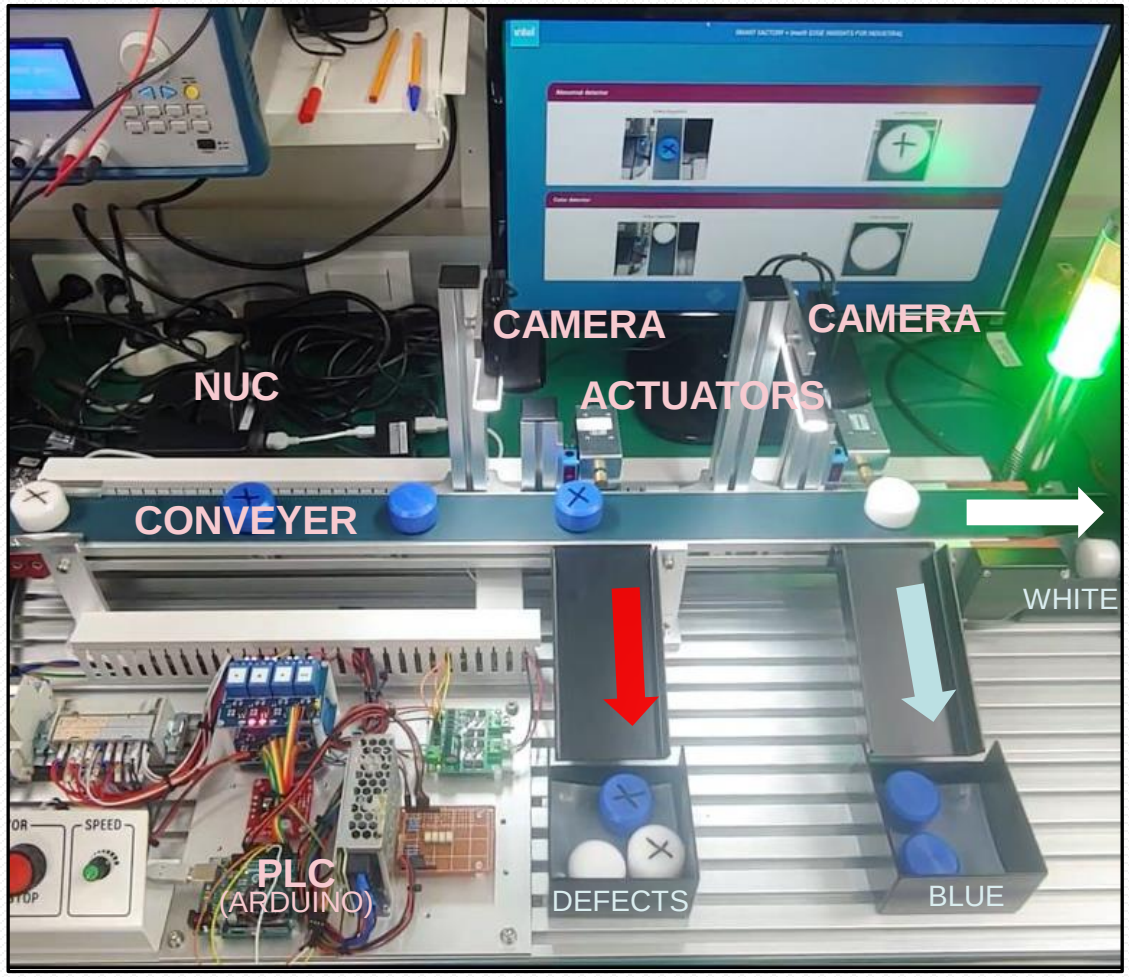
10th Generation Intel® Core™ i7-10710U
1.1 GHz – 4.7 GHz Turbo, 6 core, 12 thread, 12MB Cache
25W Intel® UHD Graphics, 300 MHz – 1.15 GHz

16GB RAM / 512GB SSD

CAMERA
LOGITECH HD PRO WEBCAM C920N

640x480 30 fps / up to 1920x1080 30 fps

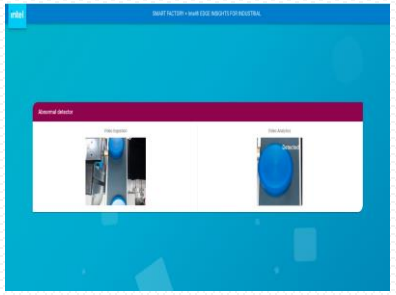
ARDUINO
Mega



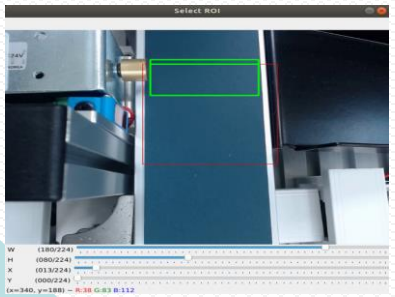
SYSTEM OVERVIEW

CAMERA INPUT

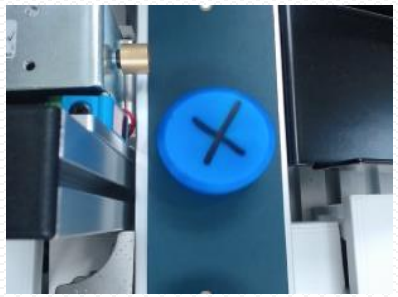
SHOW RESULTS



ROI SETTING



MOTION DETECT

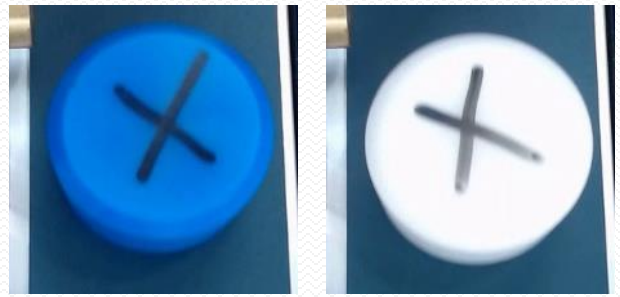


PLAIN OPENCV

DEFECT DETECT

MACHINE LEARNING

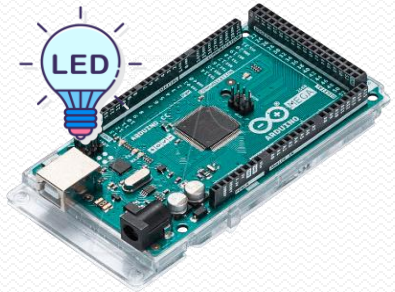
COLOR DETECT



BLUE? or WHITE?

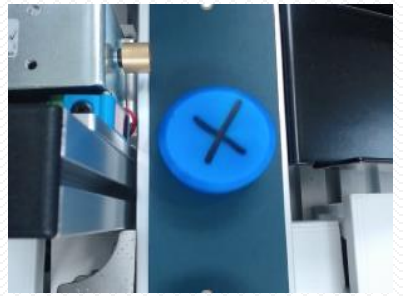
SIMULATION

ARDUINO CONTROL



VIDEO INPUT

MOTION DETECT



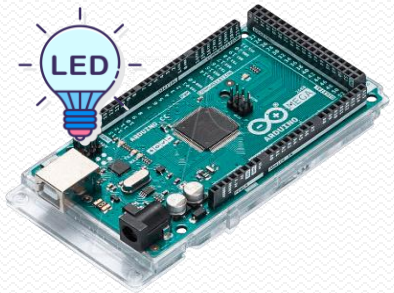
PLAIN OPENCV

COLOR DETECT

DEFECT DETECT

MACHINE LEARNING

APPLY to SMART FACTORY



HW CONTROL

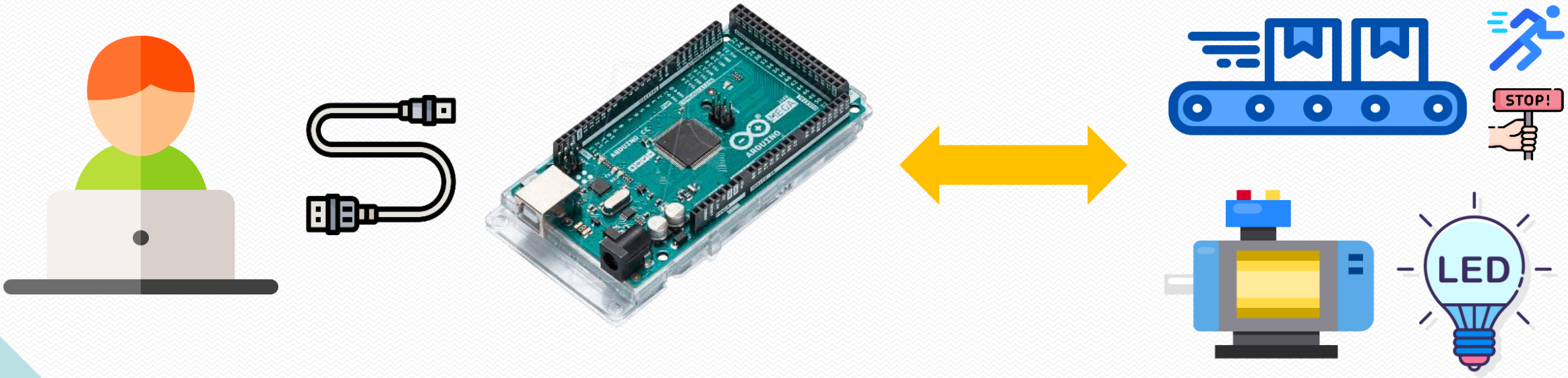
ARDUINO

***CONTROL the HW through ARDUINO** / UART*

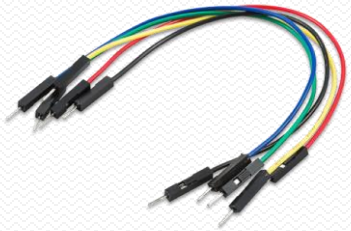
Conveyer Belt On/Off

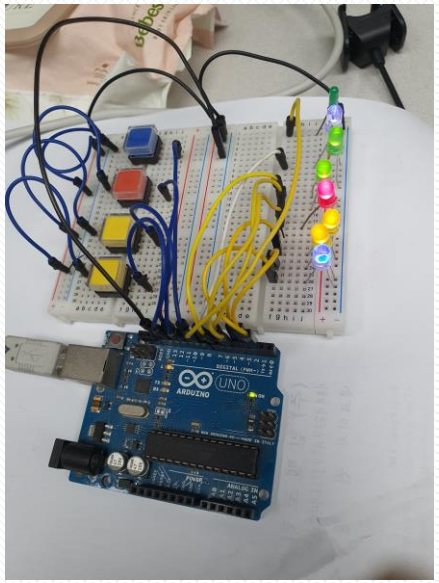
Actuator Motor Push/Release

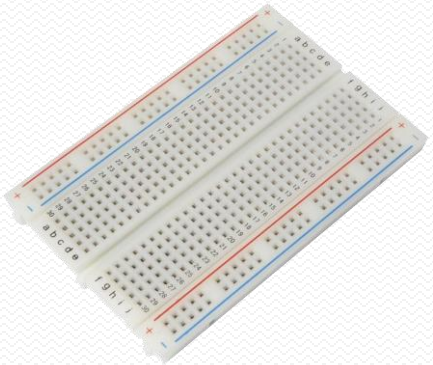
LED(Red/Orange/Green) Control



ARDUINO SETUP for SIMULATION

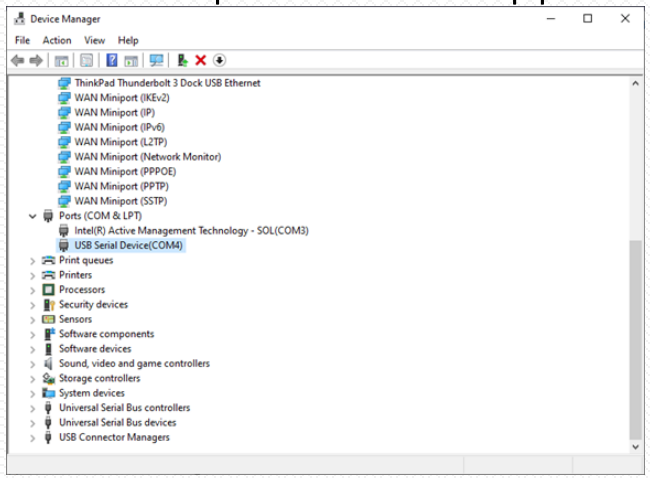






ARDUINO CONNECTION & FLASH

- COMx is mapped to /dev/ttySx in WSL
  - Each computers are mapped to different COM port.



- Flash Arduino FW
  - https://downloads.arduino.cc/arduino-cli/nightly/arduino-cli_nightly-latest_Windows_64bit.zip
  - arduino-cli config init
  - arduino-cli core install arduino:avr
  - arduino-cli compile -b arduino:avr:mega . -e -v
  - arduino-cli upload -p COM4 -b arduino:avr:mega -t -v -i build\arduino.avr.mega\arduino.ino.hex

ARDUINO CONTROL

PYTHON MODULE APIs

(from iotdemo import FatoryController)

Init the system

- *with FactoryController('/dev/ttySx') as ctrl:*

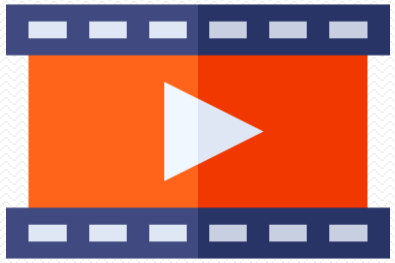
  or ctrl = FactoryController('/dev/ttySx')

Close the system

- *ctrl.close()*

For Micro Control the HW Module

Input	API
1	system_start()
2	system_stop()
3	red
4	orange
5	green
6	conveyor
7	push_actuator(1)
8	push_actuator(2)



VIDEO INPUT

Show video file

- Video file
  - conveyor.mp4
- Use cv2 library
  - import cv2
- Use these functions to show video file
  - VideoCapture(filename), read(), imshow()

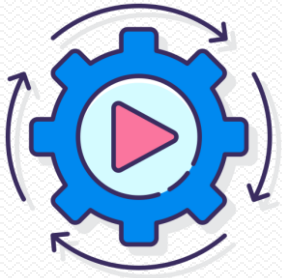
QUEUE with THREADs

THREAD 1
(COLOR CAMERA)

THREAD 2
(DEFECT CAMERA)

Queue the camera stream, receive it from the main and process it

QUEUE
BUFFER



MAIN

PYTHON MAIN (factory.py)

Enqueue (each Thread)

- *q.put(<data>)*

Dequeue (main)

- *try:*
  event = q.get_nowait()
  except Empty:
  continue

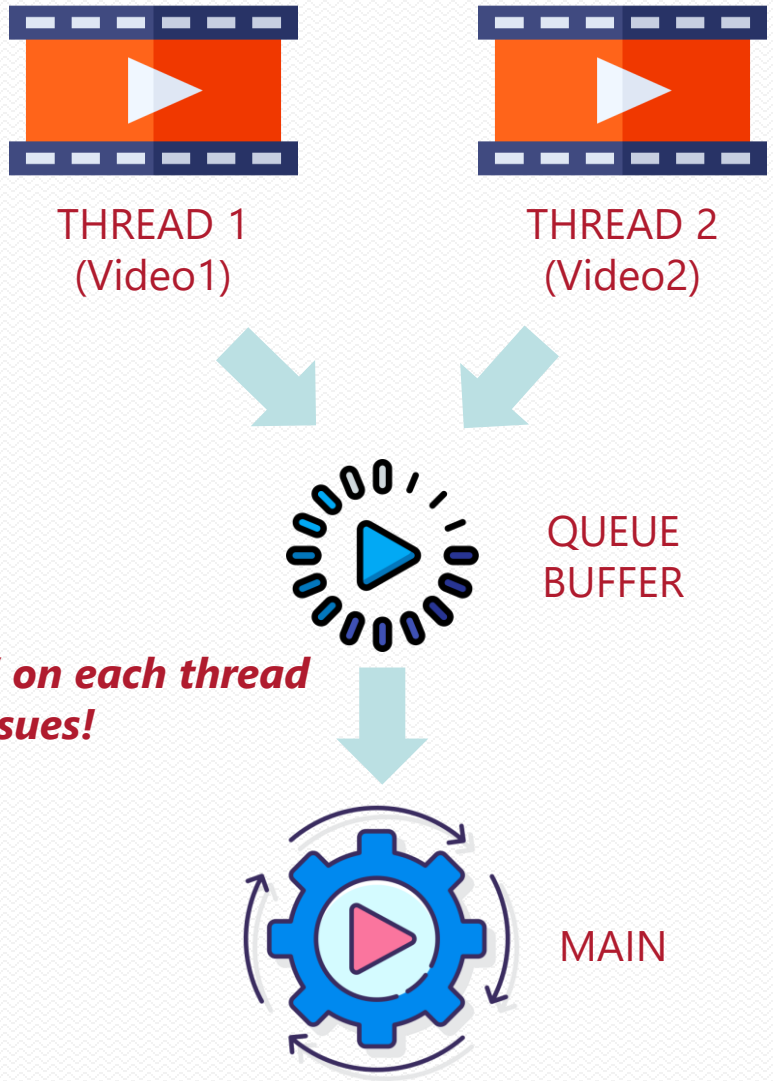
Task Done (main)

- *q.task_done()*

Print Data (main)

- *print(<data>)*

SHOW VIDEO using QUEUE



Running 'imshow' on each thread can cause crash issues!

PYTHON MAIN (factory.py)

Read Video Frame (each Thread)

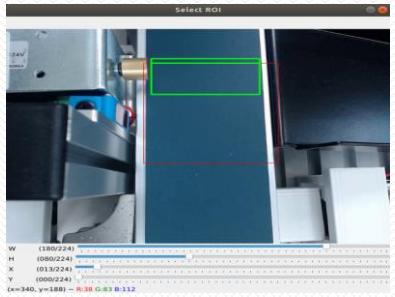
```
- _, frame = cap.read()
  if frame is None:
      break
```

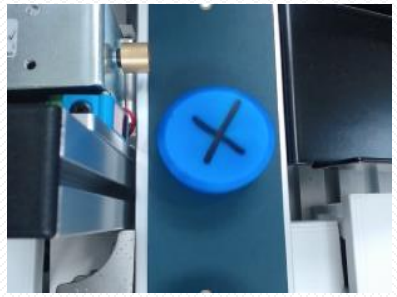
Enqueue (each Thread)

```
- q.put(('VIDEO: Cam1 live', frame))
- q.put(('VIDEO: Cam2 live', frame))
```

Dequeue (main)

```
- name, data = event
  if name.startswith('VIDEO:'):
      imshow(name[6:], data)
```

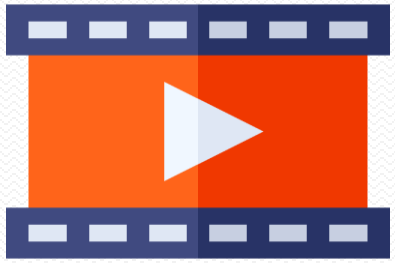




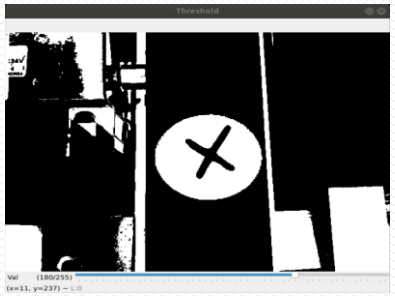

MOTION DETECTION

DETECTION FLOW

CAMERA INPUT

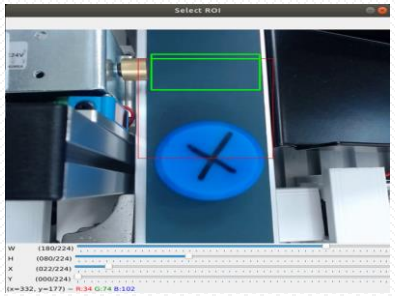


BINARIZATION



REGION OF INTEREST

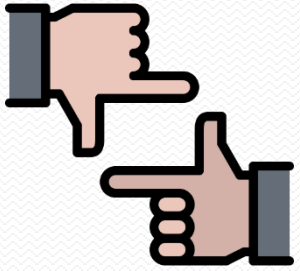
ROI SETTING



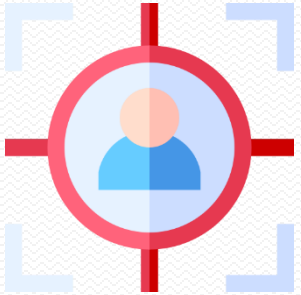
MOTION DETECT

CAMERA INPUT

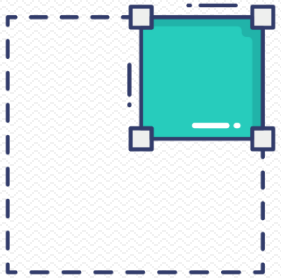
VALIDATE FRAME



MOTION DETECT

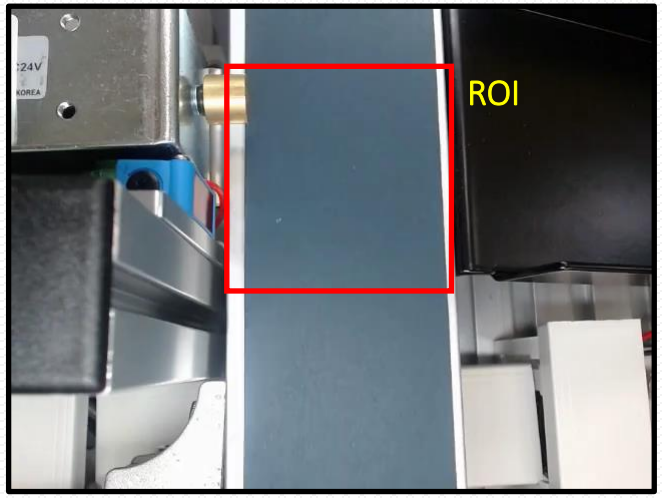


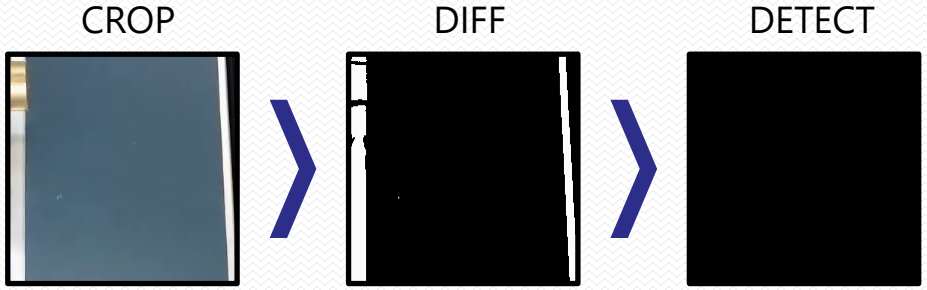
CROP IMAGE



SAVE IMAGE

DETAILED FLOW





CROP THE FRAME PER SELECTED ROI

CALCULATE THE FRAME DIFFERENCE

Apply custom threshold with brightness value

CHOOSE THE BEST FRAMES TO PASS

SELECT ROI TO DETECT MOTION

PYTHON TOOL (iotdemo-motion-detector)

iotdemo-motion-detector -l ./resource/factory/conveyor.mp4

PYTHON MAIN (factory.py)

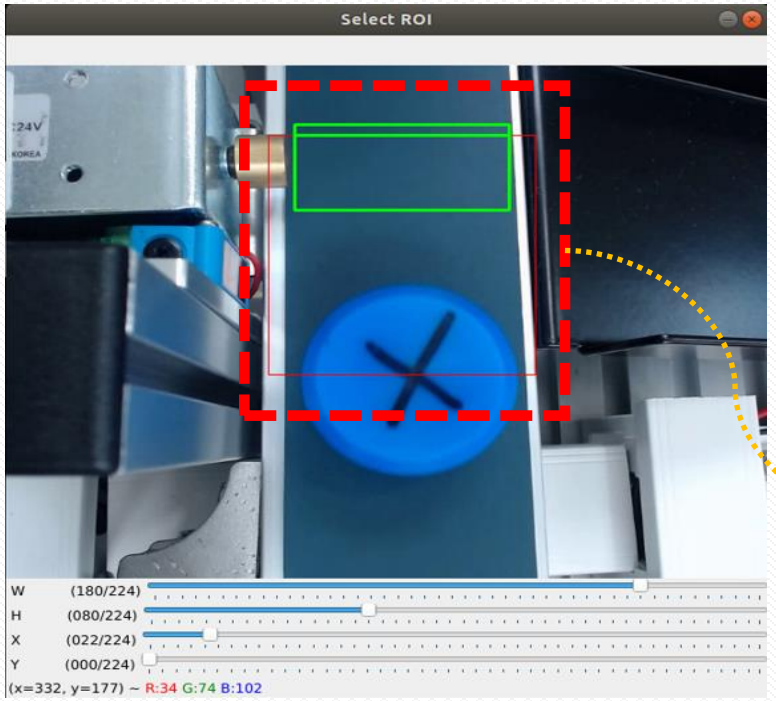
Detect the Frame (each Thread)

- *detected = det.detect(frame)*

if detected is None:
continue

Enqueue (each Thread)

- *q.put(('VIDEO: Cam1 detected', detected))*
- *q.put(('VIDEO: Cam2 detected', detected))*

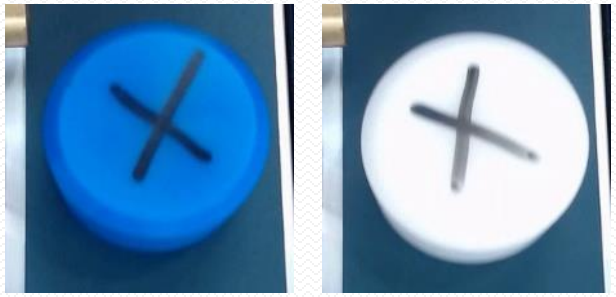


PRE-IMPLEMENTED TOOL

```
[default]
inverted = False
flipped = False
top_margin = 10
threshold = 127
top_ratio = 0.6
mid_ratio = 0.4
skip_time = 0.099
roi_top_left = (12, 0)
roi_bottom_right = (212, 90)
crop_top_left = (0, 10)
crop_bottom_right = (224, 234)
```

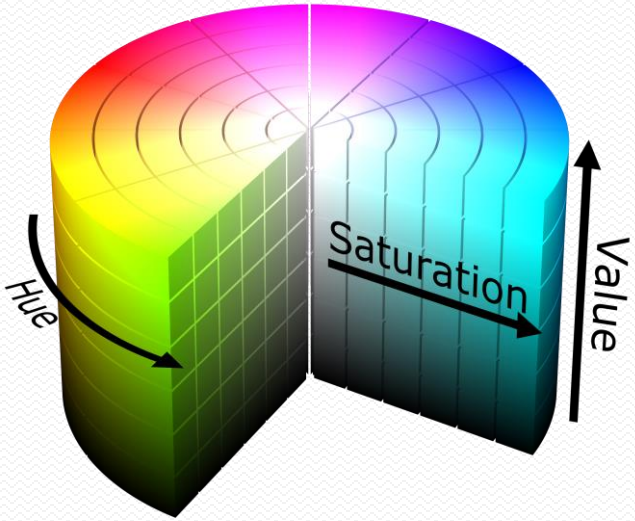
motion.cfg

CROPPED IMAGE



COLOR DETECTION

RGB vs HSV



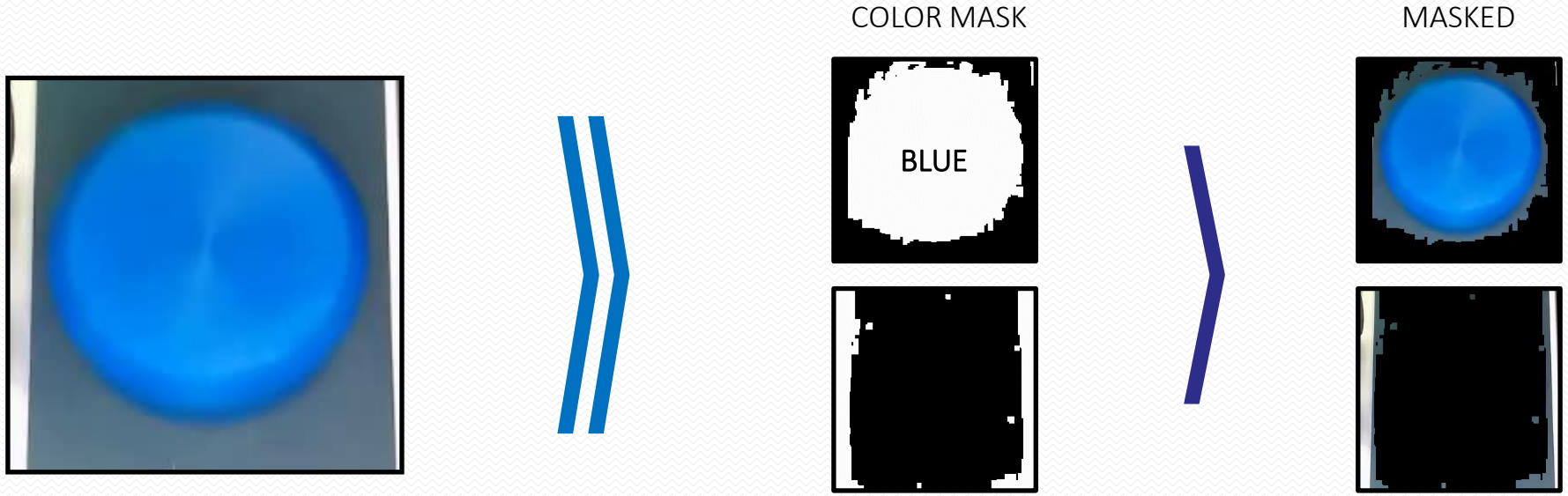
RGB (Red, Green, Blue)

additive color model in which the RGB primary colors of light are added together in various ways to reproduce a broad array of colors

HSV (Hue, Saturation, Value)

more closely aligned with the way human vision perceives color-making attributes

COLOR DETECTION FLOW



APPLY COLOR MASKS TO THE FRAMES

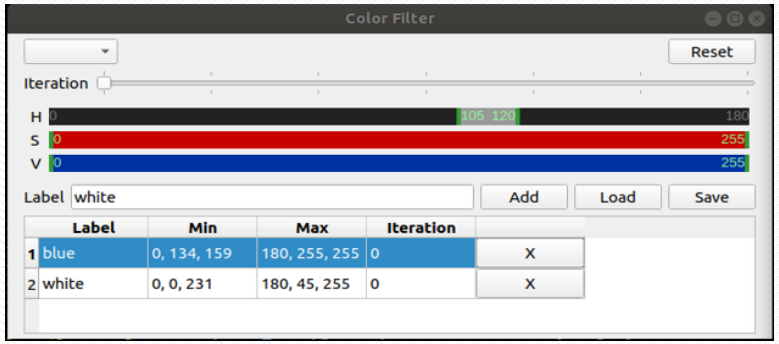
Using the HSV color model to specify the color position and color "purity"

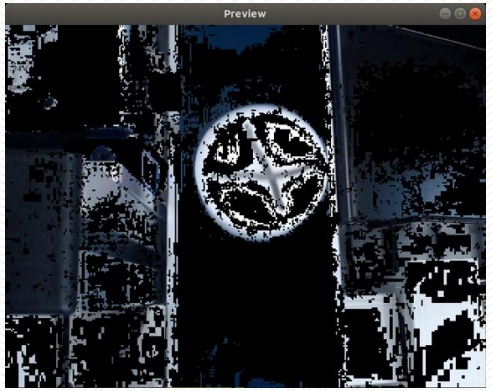
COUNT THE MASKED PIXELS AND PREDICT COLORS

COLOR DETECTION

PYTHON TOOL (iotdemo-color-detector)

iotdemo-color-detector
./resource/factory/conveyor.mp4





PYTHON MAIN (factory.py)

Color Detect (Thread 2)

- *predict = color.detect(detected)*

Get the Predict Ratio (Thread 2)

- *name, ratio = predict[0]*
  *ratio = ratio * 100*

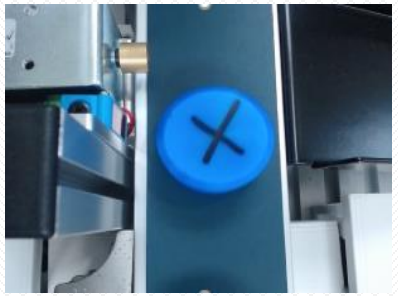
Print the Predict Ratio (Thread 2)

- *print(f"{name} : {ratio:.2f}")*

DEFECT DETECTION

TRAINING & DETECTION FLOW

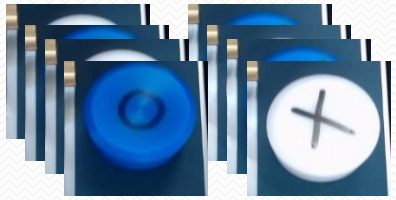
MOTION DETECT



DEFECT CLASSIFICATION

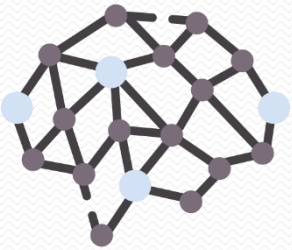
DEFECT DETECT

IMAGES UPLOAD



EXPORT IR FILES

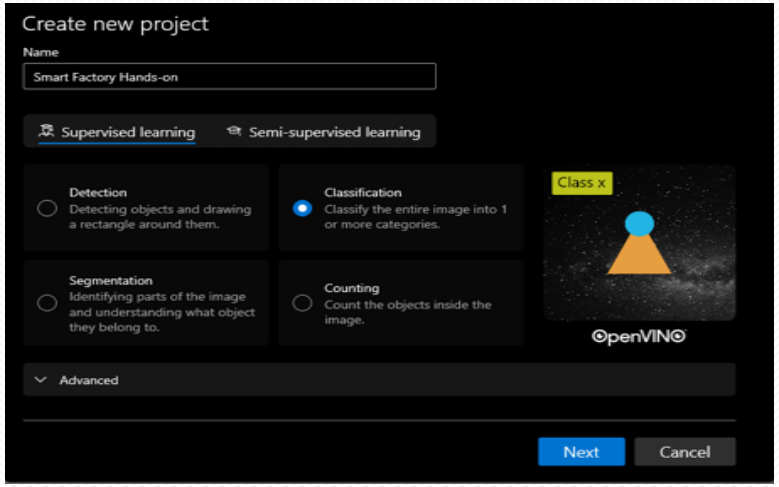
EXTERNAL
TRAINING SERVER



DATA TRAINING

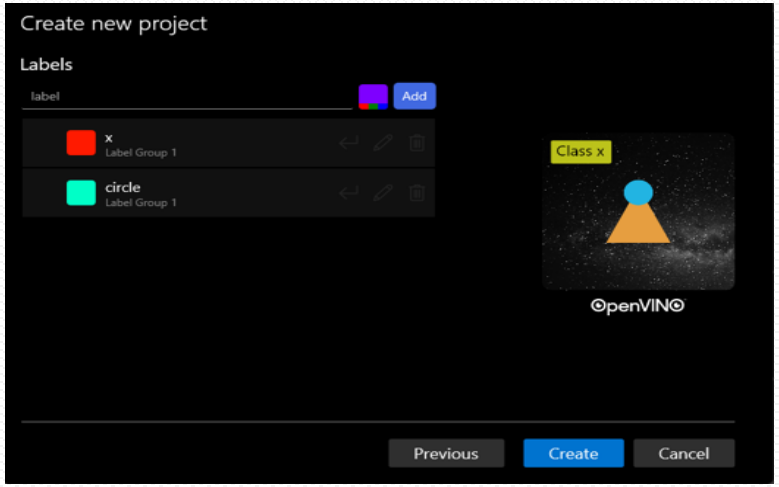
TRAINING / PREPARE PROJECT

CREATE TRAINING PROJECT WITH TOOL



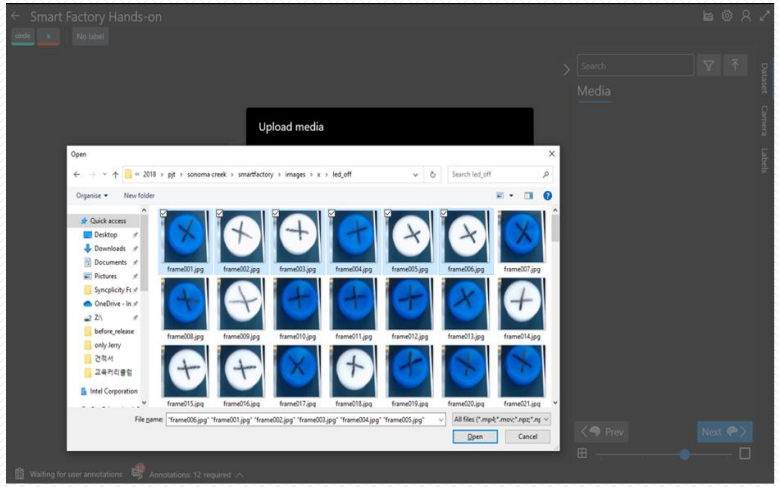
CREATE THE PROJECT

CLASSIFICATION



SET THE LABELS

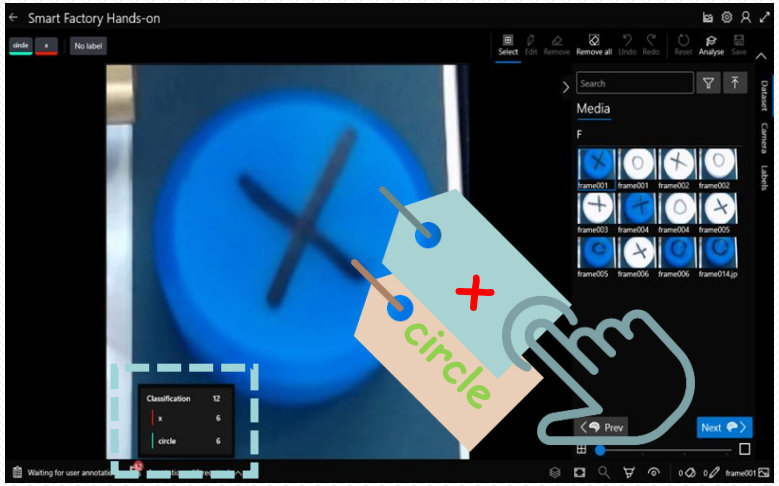
X, CIRCLE



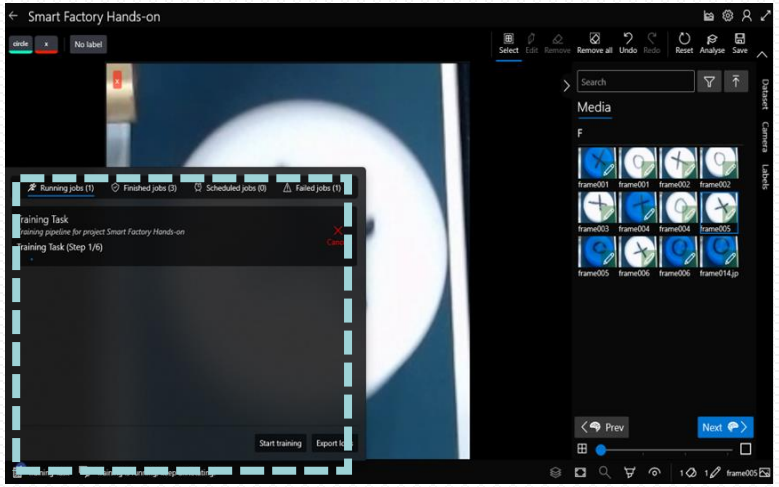
UPLOAD THE IMAGES

TRAINING / ANNOTATION

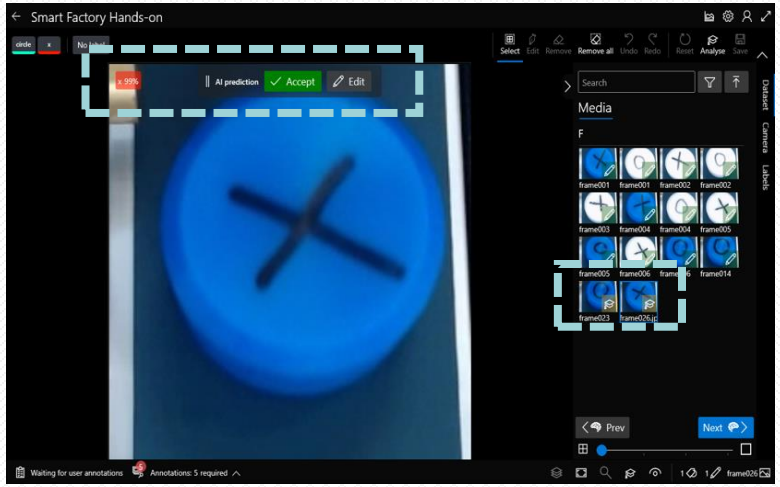
DATASET LABELING & PREDICT THE RESULT



MANUAL ANNOTATION



START THE TRAINING



PREDICT RESULT

TRAINING WITH Colab

- https://colab.research.google.com/drive/1RpqElb6px8eVmZ9kmooryyFfCUGAYw9-?usp=sharing

- Classify images to images/circle, images/x directories.
- Zip and upload to colab.

INFERENCING

RUN & TEST TRAINING RESULT

TRAINING SERVER

OPENVINO IR FILE

*.XML, *.BIN

HOST MACHINE

DEFECT DETECT

PYTHON MAIN (factory.py)

Load IR FILEs (Thread 1)

```
- ie = IECore()
  net = ie.read_network(<IR path>)
  exec_net = ie.load_network(
                    network=net,
                    device='CPU')
input_name = next(iter(net.input_info))
out_name = next(iter(net.outputs))
```

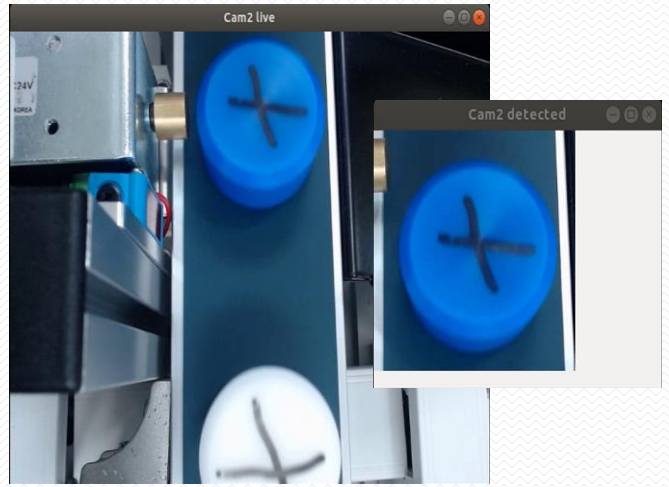
Inference the Detected Data (Thread 1)

```
- res = exec_net.infer(
      inputs={input_name: batch_tensor})
  predict = res[out_name][0]
```

Print the Predict Ratio (Thread 1)

```
- x_ratio = np.float32(predict[0])*100
  circle_ratio = np.float32(predict[1])*100
  print(f"{x_ratio:.2f}% {circle_ratio:.2f}%")
```

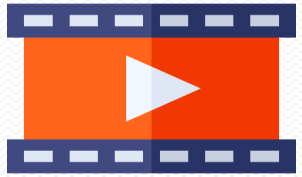
```
blue: 51.95
X = 26.19%, Circle = 73.81%
white: 54.42
X = 38.04%, Circle = 61.96%
white: 53.70
X = 40.47%, Circle = 59.53%
blue: 51.23
X = 25.25%, Circle = 74.75%
white: 53.93
X = 41.91%, Circle = 58.09%
blue: 50.11
X = 73.07%, Circle = 26.93%
blue: 50.51
X = 44.54%, Circle = 55.46%
```



INTEGRATE INTO SMART FACTORY

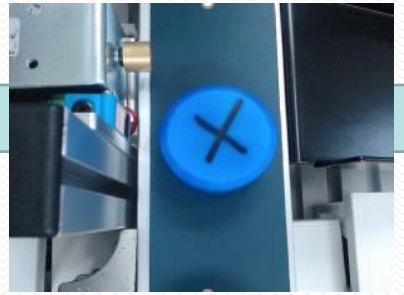
FINAL SIMULATION

VIDEO INPUT



./resources/factory/conveyor.mp4

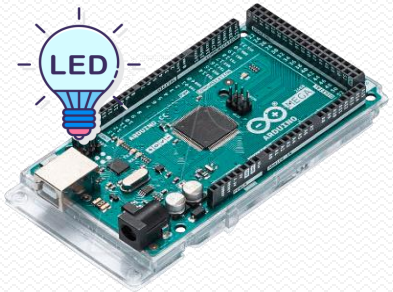
MOTION DETECT



COLOR DETECT

DEFECT DETECT

ARDUINO SIMULATION

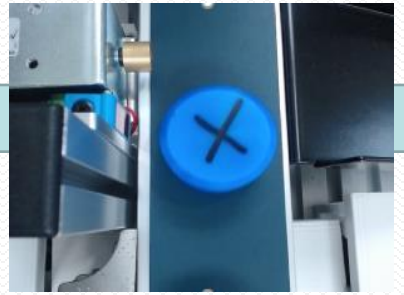


APPLY to SMART FACTORY

CAMERA INPUT

/dev/video*

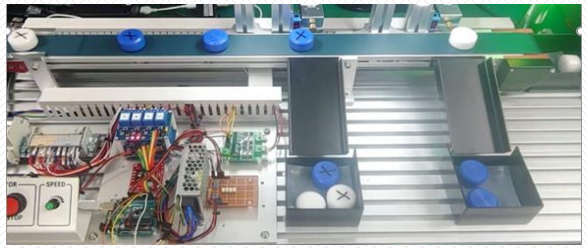
MOTION DETECT



COLOR DETECT

DEFECT DETECT

SMART FACTORY



THANK YOU