CEC Continuing Education Center

DesignNews

Getting Hands-On With Automated Inspection Concepts Using AI-Based Smart Cameras

The M5Stack AI-V Camera and the V-Function Model

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Dr. Don Wilcher

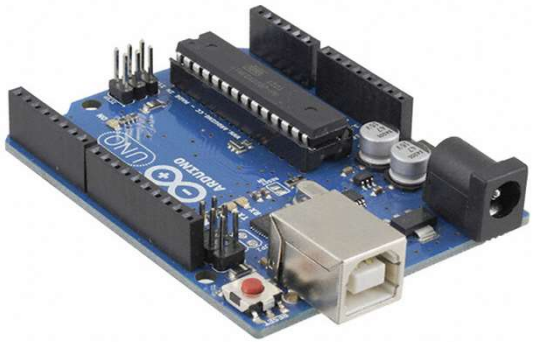
Visit ‘Lecturer Profile’ in your console for more details.

Course Kit and Materials

Pixy2 CMUCAM5

Pan/Tilt2 Servo Motor Kit for Pixy2

Arduino Uno Rev 3



M5Stack AI Camera

M5GO IoT Starter Kit V2.7

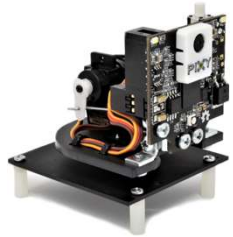
Agenda:

- V-Function Model Explained
  - a) Definition
  - b) Equation
- Bellman Equation
  - a) Definition
  - b) Equation
- The M5Stack AI-V Camera
- UiFlow Software
- Lab: Building an Automated Visual Object Detection Device with the M5Stack AI-V Camera

Seminal Research Perspective

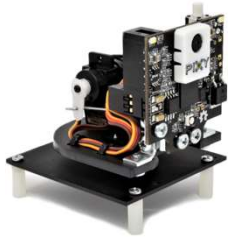
"Inspections are performed in virtually every production system. Their purpose is to verify that the production operations were carried out properly and that the production output meets the expectations of the customer" (Ben-Gal et al., 2002).

V-Function Explained: Definition



- The V-Function determines whether a point in the 3D space is visible from the camera's viewpoint or if it is blocked by other objects in the scene.
- The V-Function, in the context of a camera and image processing is known as the Visibility Function.
- The V-Function helps explain the visible different points in a scene from the camera's perspective.
- The V-Function helps render scenes accurately by computing the visibility.
  a) only the object's visible parts are rendered.
  b) helps in optimizing the rendering process and ensures visual correctness.

V-Function Explained: Equation. . .



ChatGPT Inquiry of the V-Function Equations

Please provide the equations, with explanation(s) of the V-Function.

Partial ChatGPT Response

General Visibility Function

6. **Visibility Function** $V(\mathbf{P}, \mathbf{Q})$:

$$V(\mathbf{P}, \mathbf{Q}) = \begin{cases} 1 & \text{if line segment } \overline{\mathbf{PQ}} \text{ is unobstructed} \\ 0 & \text{if line segment } \overline{\mathbf{PQ}} \text{ is occluded} \end{cases}$$

- $\mathbf{P}$: Point in space (e.g., camera or light source)
- $\mathbf{Q}$: Point in the scene

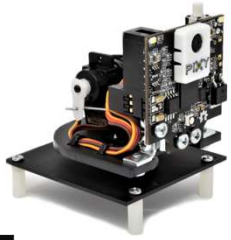
This binary function determines if $\mathbf{Q}$ is visible from $\mathbf{P}$.

Question 1

In the Visibility Function V(P, Q), the P is for

a) Precision
b) Performance
c) Point in the object
d) Point in space

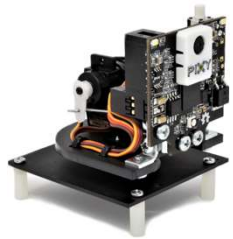
V-Function Explained: Equation. . .



Equivalent General Visibility Function Equation modeled in Python

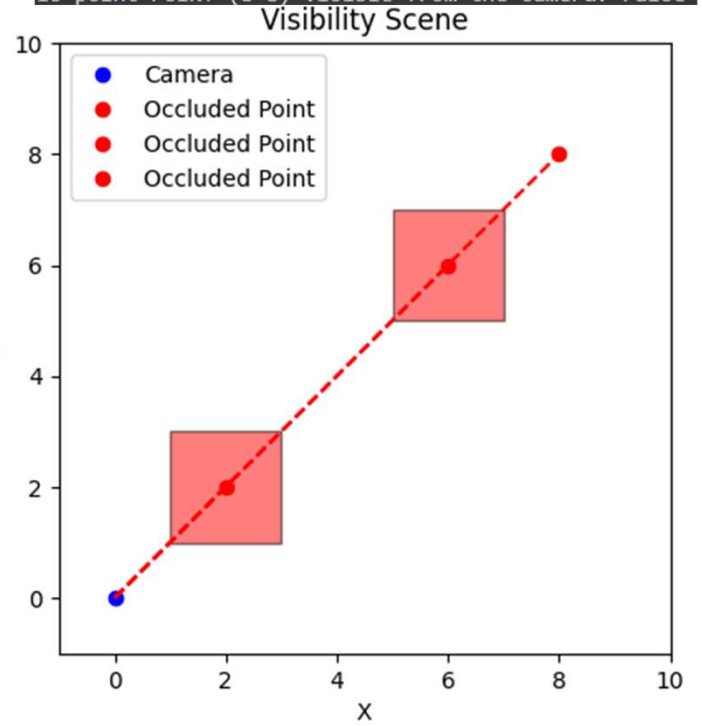
```
# Plot points and visibility lines
for point in points:
    if self.is_visible(camera, point):
        ax.plot([camera.x, point.x], [camera.y, point.y], 'g-')
        ax.plot(point.x, point.y, 'go', label='Visible Point')
    else:
        ax.plot([camera.x, point.x], [camera.y, point.y], 'r--')
        ax.plot(point.x, point.y, 'ro', label='Occluded Point')
```

V-Function Explained: Equation. . .

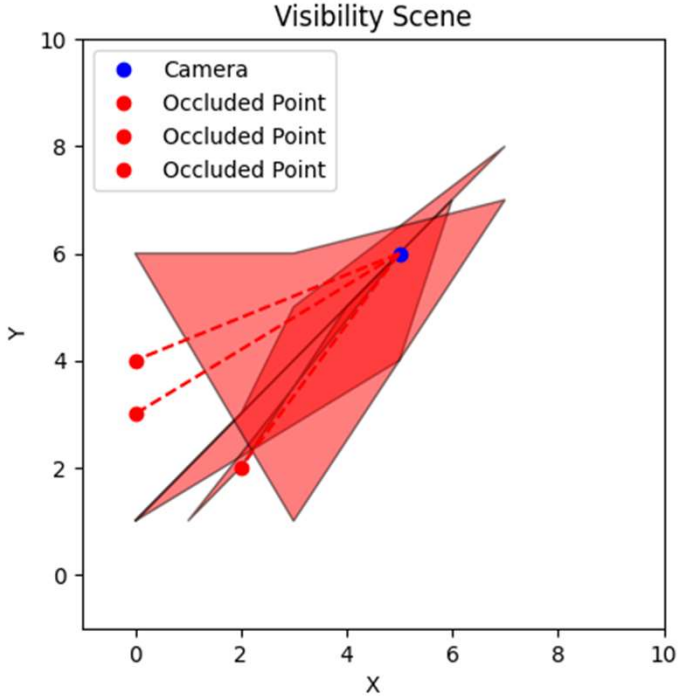


Example Plots of V-Function modeled in Python

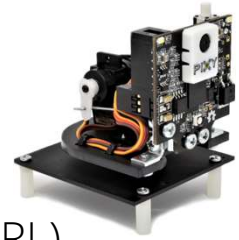
```
Is point POINT (2 2) visible from the camera? False
Is point POINT (6 6) visible from the camera? False
Is point POINT (8 8) visible from the camera? False
```




```
Is point POINT (0 3) visible from the camera? False
Is point POINT (2 2) visible from the camera? False
Is point POINT (0 4) visible from the camera? False
```

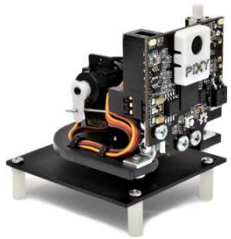


Bellman Equation: Definition . . .



- The Bellman Equation – One of the central elements of many Reinforcement Learning (RL) algorithms and required for calculating the value function.
- The Bellman Equation decomposes the value function[V(s)] into two parts.
  a) the immediate reward
  b) discounted future values
- The Bellman equation benefits automated visual inspection and detection systems using the following approaches
  a) Optimized Inspection strategies – Focuses on States: current frame or image captured by a camera.
  b) Reinforcement Learning (RL) for defect detection- Trained on a data set of images with known defects and non-defective samples.
  c) Provides a learning process policy to improve its inspection and detection performance over time.

Bellman Equation: Definition . . .



Bellman Equation

The Bellman Equation relates the value of a state to the values of its successor states. For a given policy π, the Bellman Equation for the value function $V^{\pi}(s)$ is:

$$V^{\pi}(s) = \sum_{a \in A} \pi(a \mid s) \left[R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V^{\pi}(s') \right]$$

In this equation:

- $\pi(a \mid s)$ is the probability of taking action a in state s under policy π.
- $R(s,a)$ is the immediate reward received after taking action a in state s.
- $P(s' \mid s,a)$ is the transition probability of moving to state s' from state s after taking action a.
- $V^{\pi}(s')$ is the value of the successor state s'.

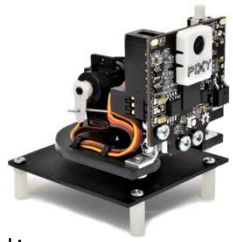
Question 2

The Bellman Equation relates the valve of a state to the values of its proceeding states.

a) True

b) False

Bellman Equation: Definition: Image Classifier . . .



The Bellman Equation can be implemented as an image classifier based on the reward policy.

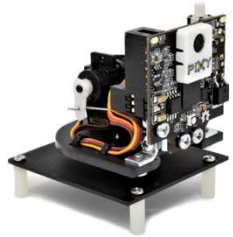
- If the value function "-V^pi (s)" is negative the result is a misclassification of the image feature(s).
- A positive"+ V^pi (s)" in the reward policy has identified the image feature(s) correctly.
- To implement this reward policy in a software agent using Python, the snippet of code is shown next.

```
12      def bellman_update(self):
13          new_value_function = np.zeros_like(self.value_function)
14          for s in range(len(self.states)):
15              value_list = []
16              for a in range(len(self.actions)):
17                  value = self.rewards[s][a] + self.discount_factor * np.sum(self.transition_probabilities[s][a] * self.value_function)
18                  value_list.append(value)
19              new_value_function[s] = max(value_list)
20          self.value_function = new_value_function
```

Line 17 implements the Bellman Equation

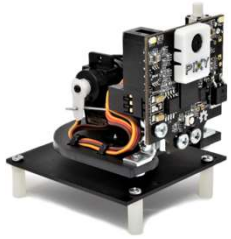
Partial Python Code in Google Colaboratory

Bellman Equation: Definition: Image Classifier . . .



```python
import numpy as np

class BellmanAgent:
    def __init__(self, states, actions, transition_probabilities, rewards, discount_factor=0.9):
        self.states = states
        self.actions = actions
        self.transition_probabilities = transition_probabilities
        self.rewards = rewards
        self.discount_factor = discount_factor
        self.value_function = np.zeros(len(states))

    def bellman_update(self):
        new_value_function = np.zeros_like(self.value_function)
        for s in range(len(self.states)):
            value_list = []
            for a in range(len(self.actions)):
                value = self.rewards[s][a] + self.discount_factor * np.sum(self.transition_probabilities[s][a] * self.value_function)
                value_list.append(value)
            new_value_function[s] = max(value_list)
        self.value_function = new_value_function

    def value_iteration(self, iterations=100):
        for _ in range(iterations):
            self.bellman_update()
        return self.value_function
```

Bellman Equation: Definition: Image Classifier . . .



Partial Python Code in Google Colaboratory: Output Results
100 Iterations, negative reward policy applied, Misclassification of apple and orange features.

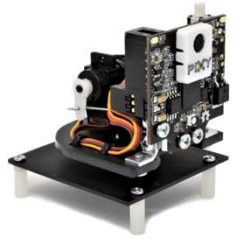
```
Initial Value Function:
Value of state apple_feature_1: 0.00
Value of state apple_feature_2: 0.00
Value of state orange_feature_1: 0.00
Value of state orange_feature_2: 0.00

Value Function after Value Iteration:
Value of state apple_feature_1: -100.00
Value of state apple_feature_2: -100.00
Value of state orange_feature_1: -100.00
Value of state orange_feature_2: -100.00
```

Question 3

In reviewing slide 17, which feature received a positive reward?

a) apple_feature 1
b) apple_feature 2
c) orange_feature 2
d) none of the above

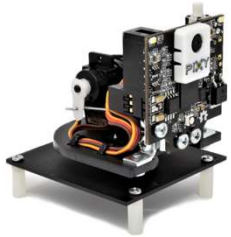
Bellman Equation. . .
Markov Decision Process



The Markov Decision Process (MDP) is a mathematical framework used to model decision-making events. The MDP outcomes are partially random and under the control of a decision-making policy.

Note: The relationship between the V-Function and the Bellman Equation is the MDP.

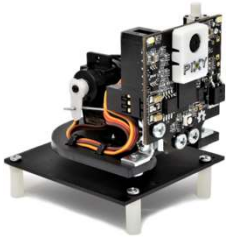
The M5Stack AI-V Camera



- The M5Stack AI-V is a compact, versatile, and AI-powered camera module developed by M5Stack.
- The camera is designed to facilitate machine vision applications and is equipped with various features that make it suitable for use in AI.

M5Stack AI-V Camera

The M5Stack AI-V Camera

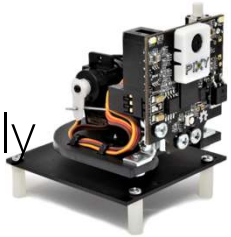


The M5Stack AI-V camera can be deployed in various applications. Listed below are typical applications for the M5Stack AI-V camera.

- Smart Surveillance – Security systems for real-time monitoring and alerting.
- Industrial Automation – Visual inspection with defect detection capabilities can enhance automation systems.
- Smart Home Devices – Tasks like object recognition and monitoring can be integrated with home automation systems.
- Educational Projects – Hands-on tools for learning about AI and computer vision can be built using the AI-V camera.

The UIFlow software will be used to program the M5Stack AI-V Camera for object detection and recognition.

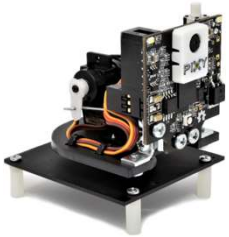
UIFlow Software. . .



- UIFlow software is a graphical programming interface designed primarily for creating Internet of Things (IoT) applications.
- The UIFlow software was developed by the company M5Stack.
- The goal of UIFlow is to simplify the programming for various embedded applications using a range of M5Stack development modules.

M5GO IoT Starter Kit V2.7

UIFlow Software



The UIFlow Software is implemented in Blockly created by Google.

No Code Development

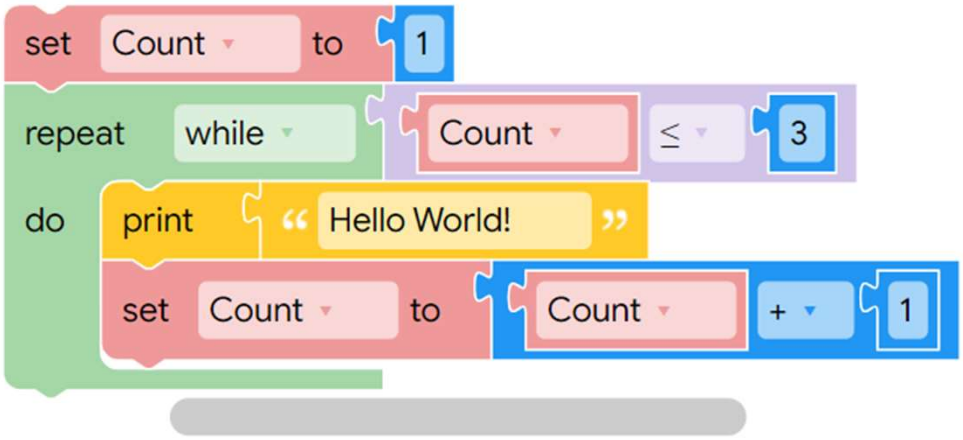
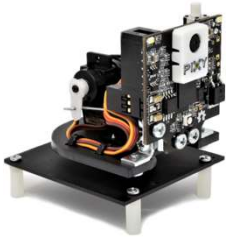


Image Courtesy of Google Blockly: https://developers.google.com/blockly

UIFlow Software

The UIFlow Software and Firmware Burning Tool.



Documents Page:
https://docs.m5stack.com/en/download

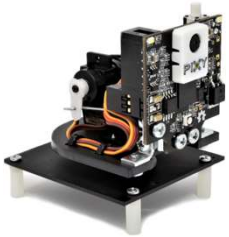
UIFLOW SOFTWARES

NO	Name	Download
1	UIFlow Web IDE	
2	Desktop IDE Win10 x64 (update is terminated)	
3	Desktop IDE MacOS (update is terminated)	
4	Desktop IDE Linux (update is terminated)	
5	UIFlow Local Server for Windows11 x64	
6	UIFlow Local Server for MacOS	
7	UIFlow Local Server for Ubuntu22.04	
8	UIFlow Local Server for Linux arm (Support CM4Stack)	

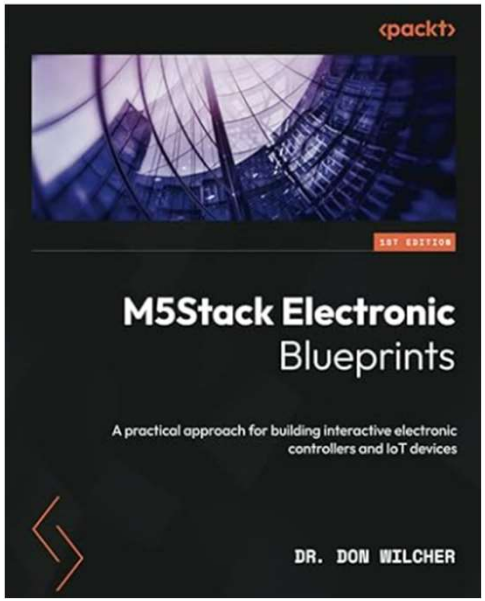
UIFLOW FIRMWARE BURNING TOOL

NO	Name	Download
1	M5Burner Win10 x64 v3.0	
2	M5Burner MacOS x64 v3.0	
3	M5Burner Linux x64 v3.0	

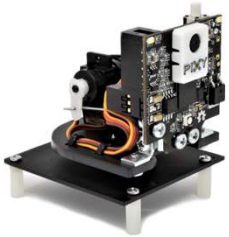
UIFlow Software



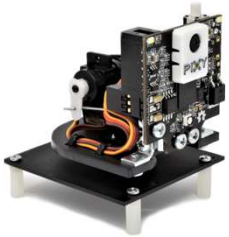
Additional information on UIFlow Software, M5Stack development modules and projects can be found in the following book.



Lab: Build an Automated Visual Object Detection Device with Feature Extractor



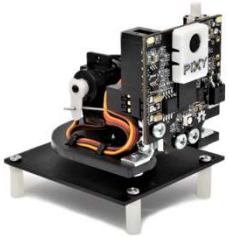
Lab: Build an Automated Visual Object Detection Device with Feature Extractor. . .



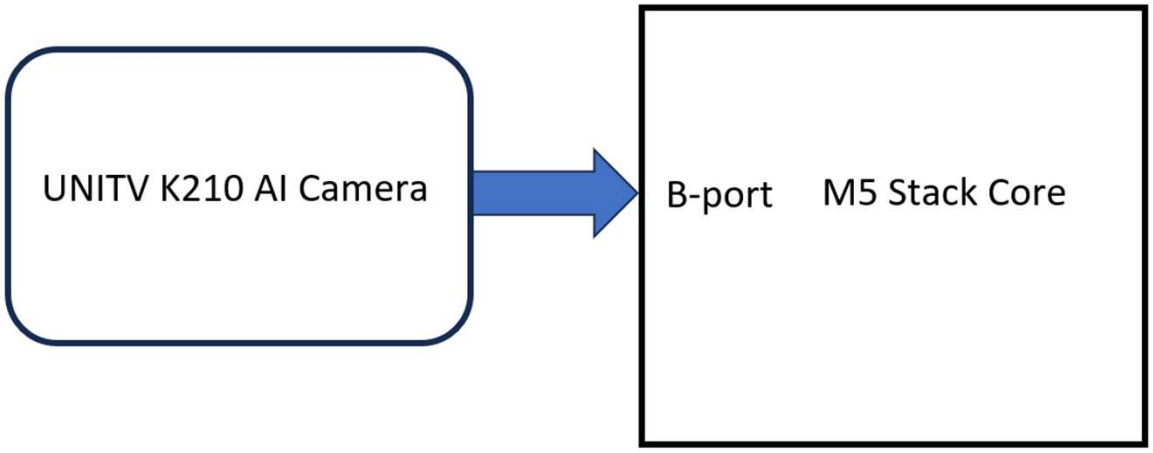
Lab Objectives:

- Participants will learn to burn ColorTracker firmware into the M5Stack UNITV camera.
- Participants will learn to record LAB Color features from an object using the M5Stack LAB Color tool.
- Participants will learn to run UIFlow Color-Track Blockly code on the M5Stack UNITV camera.
- Participants will learn to extract features from an object and display the values on the M5Stack Core.

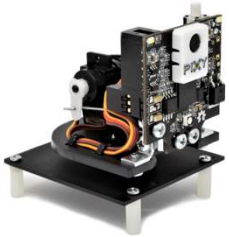
Lab: Build an Automated Visual Object Detection Device with Feature Extractor. . .



Concept Block Diagram

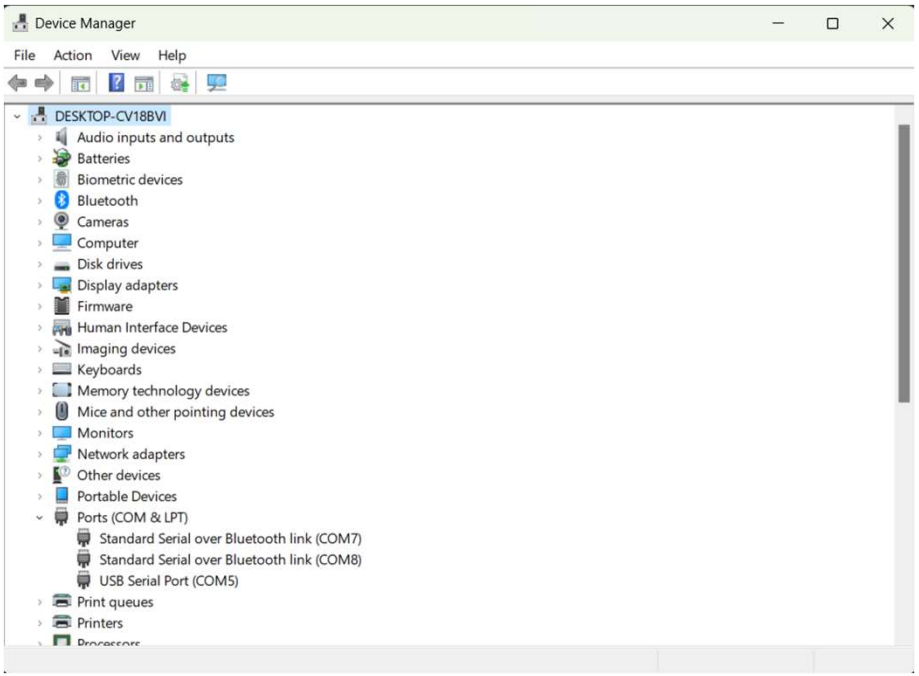


Lab: Build an Automated Visual Object Detection Device with Feature Extractor. . .

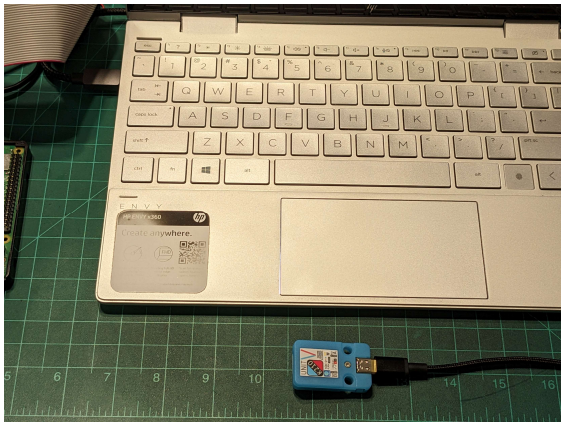


Burning Object Tracker Firmware to M5Stack AI Camera

UNITV Camera's COM port



UNITV Camera Attached to COM Port

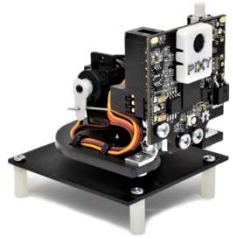


Question 4

In reviewing slide 29, which COM port is the UNITV camera attached to?

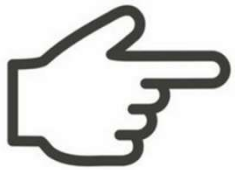
a) 7
b) 8
c) 3
d) 5

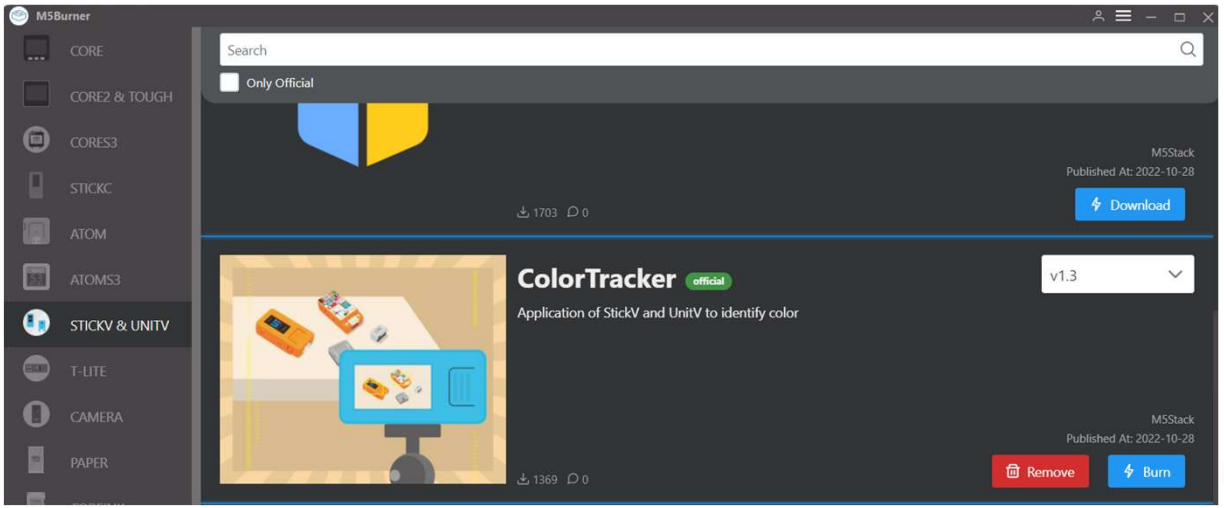
Lab: Build an Automated Visual Object Detection Device with Feature Extractor…



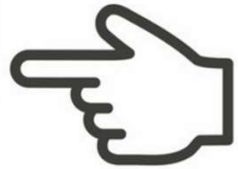
Open the M5Burner Tool and Select STICKYV & UNITV.
Click Burn button to install ColorTracker Firmware onto M5Stack AI Camera

Select STICKY V & UNITV option

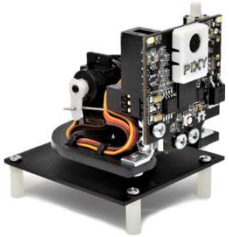




Click the Burn button to install ColorTracker Firmware



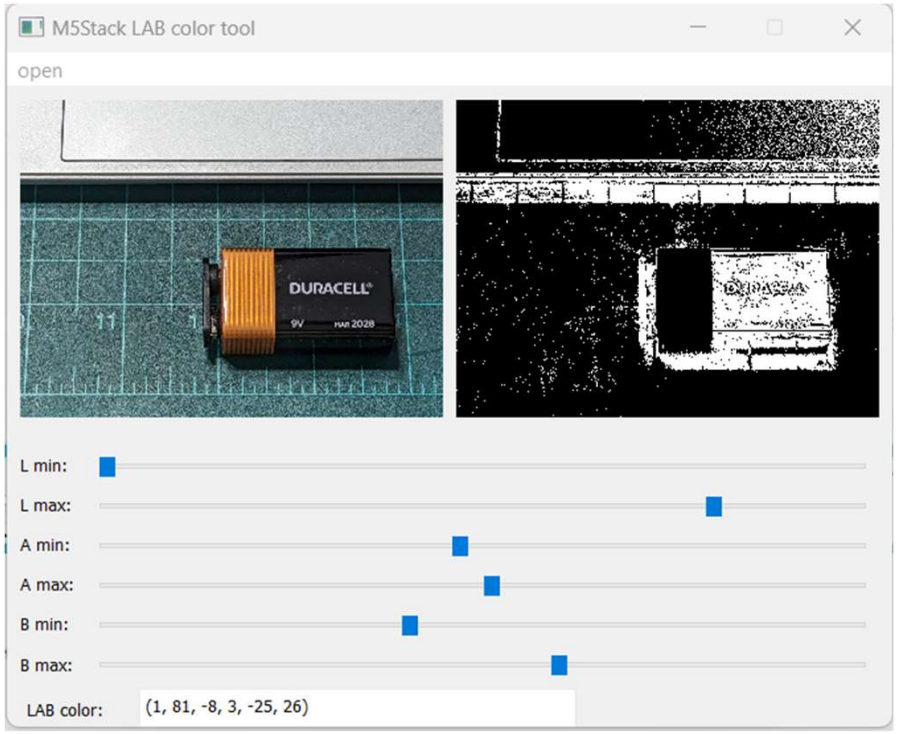
Lab: Build an Automated Visual Object Detection Device with Feature Extractor…



Obtaining LAB Color data for V-Function UIFlow code.

Click on image to obtain initial LAB color data

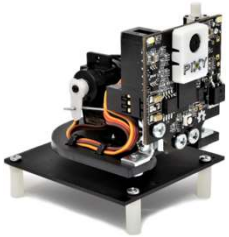
Adjust sliders to obtain the best image



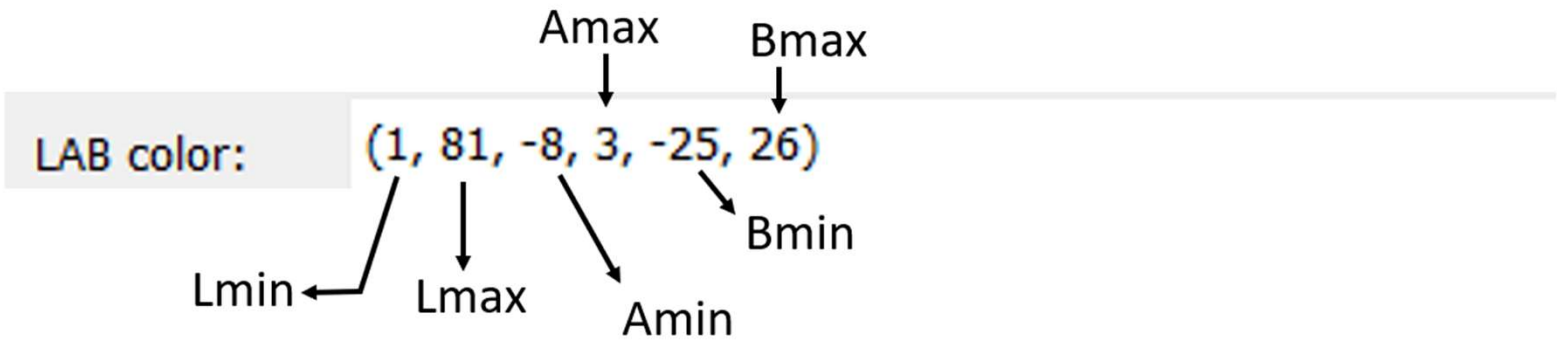
Best Image

Record LAB color data

Lab: Build an Automated Visual Object Detection Device with Feature Extractor…

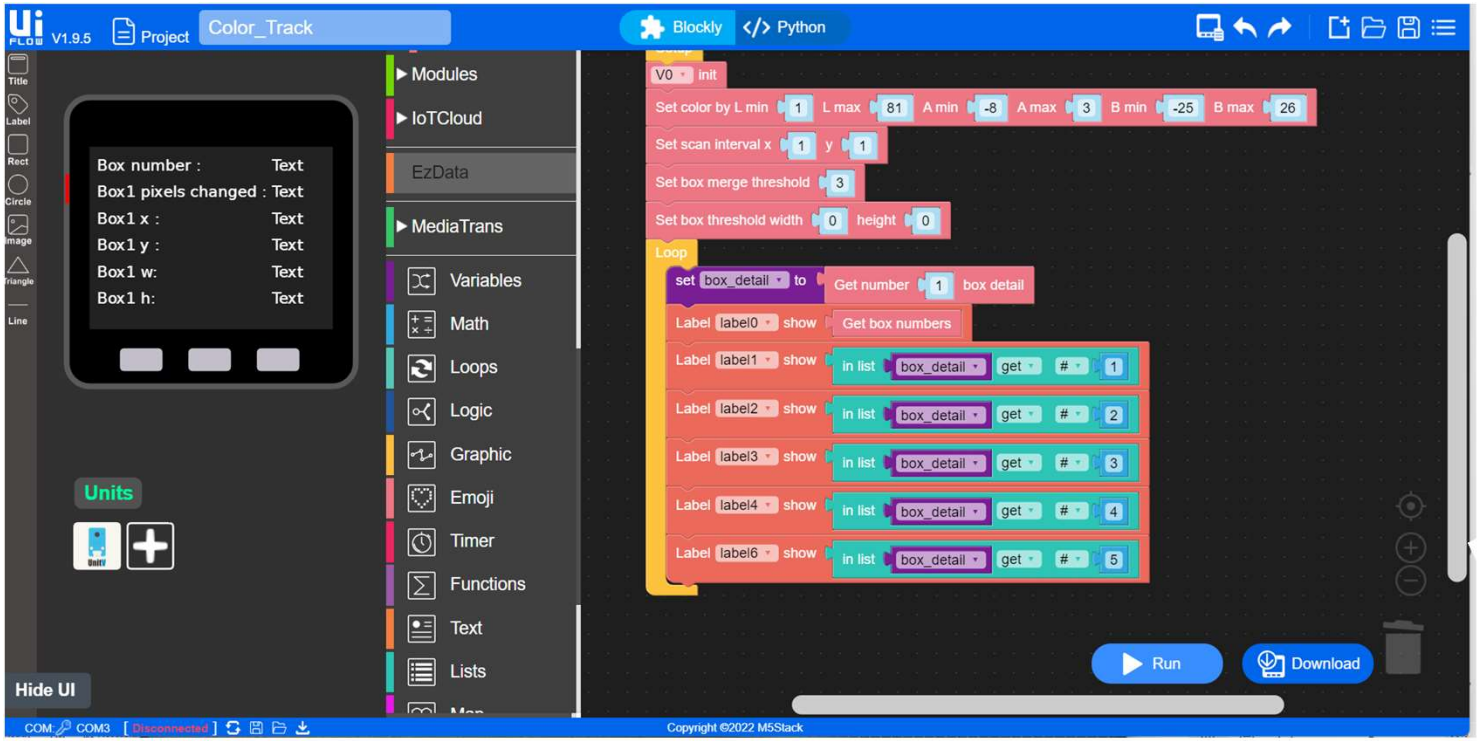


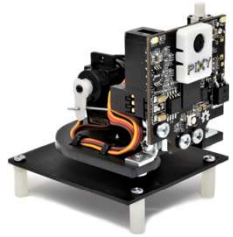
Slider Data Values



Lab: Build an Automated Visual Object Detection Device with Feature Extractor…

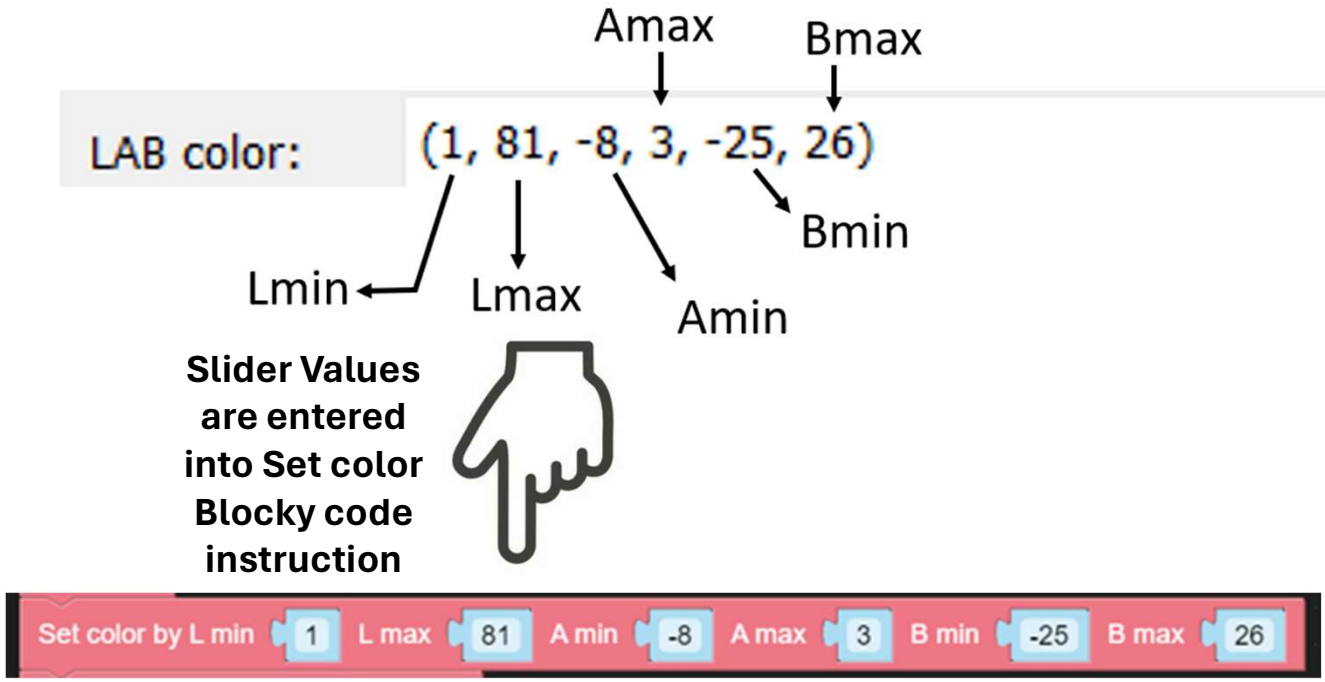
Open UIFlow software, then open the Color_Track Blockly Code.



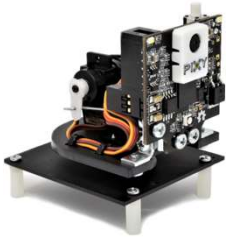


Click the RUN button to operate the Blocky Code on the UNITV camera

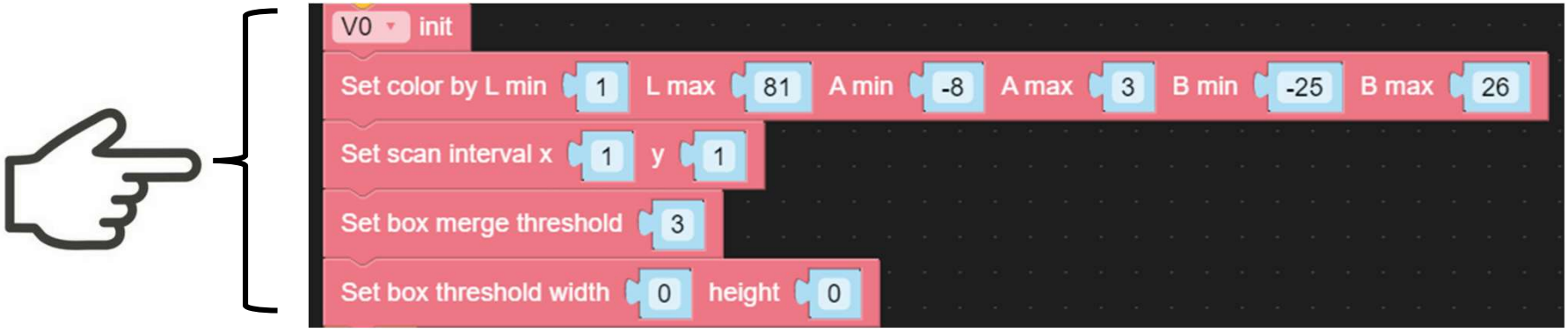
Lab: Build an Automated Visual Object Detection Device with Feature Extractor…



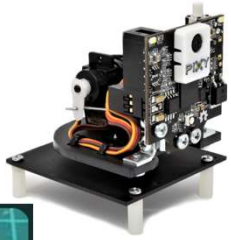
Lab: Build an Automated Visual Object Detection Device with Feature Extractor…



Experiment with these Blockly code parameters for better feature extraction and detection

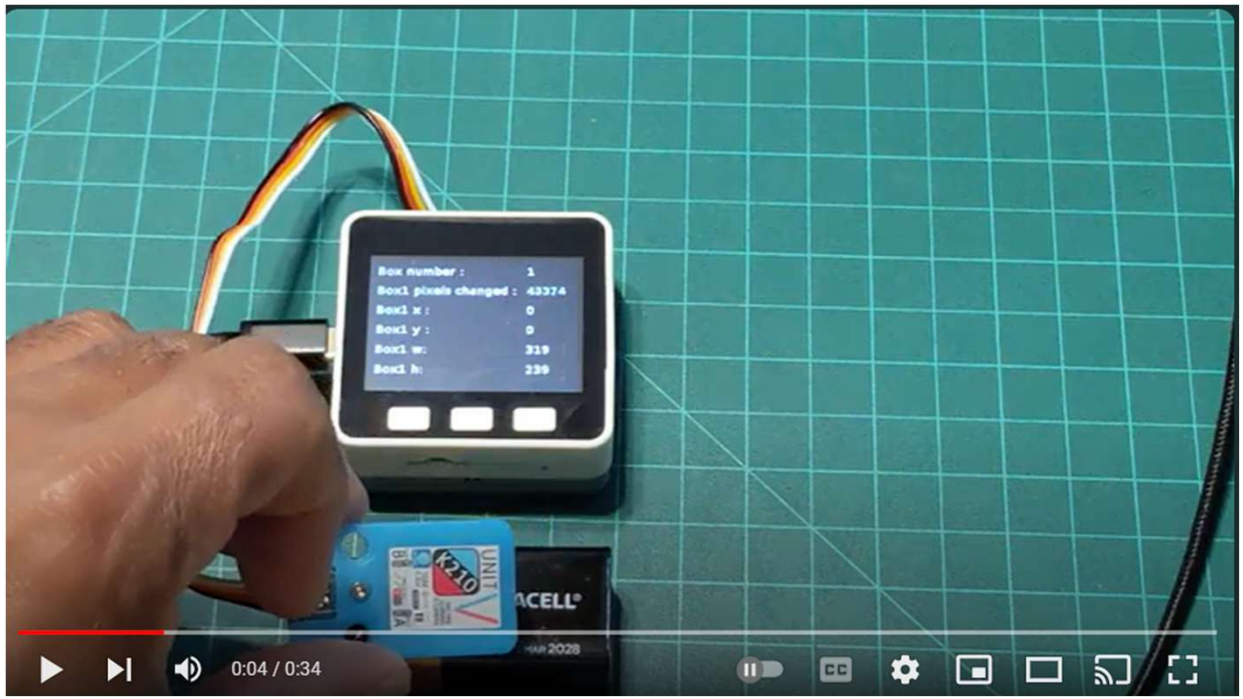


Lab: Build an Automated Visual Object Detection Device with Feature Extractor…



Watch YouTube Video to see the device in action!

https://youtu.be/wk0aLaIlOec



Question 5

In reviewing slide 35, which Color parameter has a -25 value?

a) L min
b) L max
c) B max
d) B min

Thank you for attending

Please consider the resources below:

Ben-Gal, I, Herer, Y. T., & Raz, T. (2002). Self-correcting inspection procedure under errors. *IIE Transactions, 34*, 529 – 540. https://www.academia.edu/12922699/Self-correcting_inspection_procedure_under_inspection_errors

Bozinovski, S. (2020). Reminder of the first paper on transfer learning in neural networks, 1976. *Informatics 44*, 291-302. https://www.researchgate.net/publication/346435488_Reminder_of_the_First_Paper_on_Transfer_Learning_in_Neural_Networks_1976

Chin, R.T., & Harlow, C. A. (1992). Automated visual inspection: A survey. IEEE Transactions On Pattern Analysis and Machine Intelligence, 4 (6), 557-573. https://ieeexplore.ieee.org/document/4767309

Gounaridou, A., Pantraki, E., Dimitriadis, A.T., Ioaannidis, D., & Tzovaras, D. (2023). Semi-automated visual quality control inspection during construction or renovation of railways using deep learning techniques and augmented reality visualization. *Proceedings of the 23rd International Conference On Construction Applications of Virtual Reality*, 865 -976. https://www.researchgate.net/publication/378535268_Semi-Automated_Visual_Quality_Control_Inspection_During_Construction_or_Renovation_of_Railways_Using_Deep_Learning_Techniques_and_Augmented_Reality_Visualization

Panella, F., Lucy, J., Fisk, E., Huang, S.T., & Loo, Y. (2023). Computer vision and machine learning for cost-effective automated visual inspection of tunnels: A case study. https://www.taylorfrancis.com/chapters/oa-edit/10.1201/9781003348030-340/computer-vision-machine-learning-cost-effective-fully-automated-visual-inspection-tunnels-case-study-panella-lucy-fisk-huang-loo

Thank you for attending

Please consider the resources below:

Rahimi, H.N., & Nazemizadeh, M. (2013). Dynamic analysis and intelligent control techniques for flexible manipulators: A review. *Advanced Robotics*, 1- 14. https://www.academia.edu/32830488/Dynamic_analysis_and_intelligent_control_techniques_for_flexible_manipulators_a_review

Github Code: https://github.com/DWilcher/DesignNews-WebinarCode/blob/main/June_24_Webinar_code.zip

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets