CEC Continuing Education Center

DesignNews

Exploring Smart AI Lens with the Micro:bit

DAY 4: Building a Recognition Application Workflow

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.

LinkedIn Page:
https://www.linkedin.com/in/dr-don-wilcher-ed-d-mseit-ee-ceta-2735151/

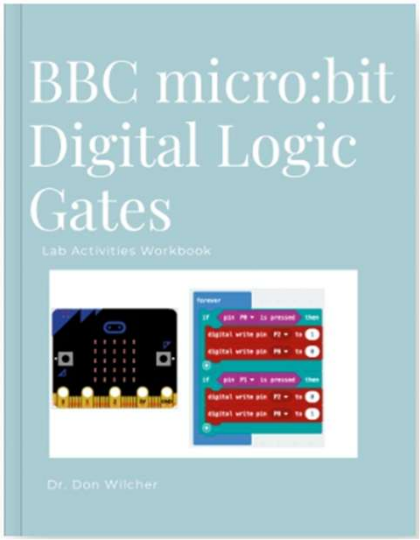
Patreon Page:
https://www.patreon.com/c/DrDon683

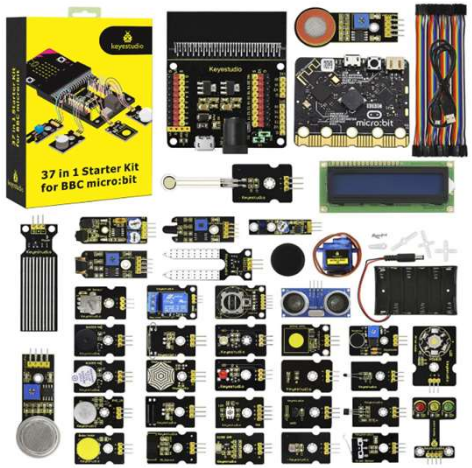
Course Kit and Materials

Micro:bit Version 2

Smart-AI Lens

BBC Micro:bit Digital Logic Gates Lab Activities Workbook





Keyestudio 37-In-1 Sensor Starter Kit

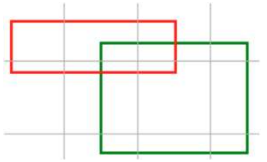
Research Perspective

"A single neural network predicts bounding boxes and class probabilities, directly from full images in one evaluation[1]."

Agenda:

- What Is a Recognition Application Model Workflow?
- Microsoft MakeCode Python Smart AI-Lens APIs
- Lab: Enhanced Characteristic Learn Device

What Is a Recognition Application Model Workflow



A Recognition Application Model Workflow is a structured process used to:

a) design
b) train
c) evaluate,
d) and deploy models.

The purpose of the workflow is to identify or classify patterns, objects, signals, or behaviors from input data (images, audio, text, sensor data, etc.).

Question 1

Which statement is correct?

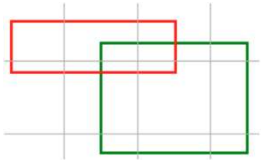
a) A Recognized Application Model Workflow is a structured process.

b) A Recognized Application Model Workflow is a stochastic process.

c) A Recognized Application Model Workflow is a linear process.

d) none of the above

What Is a Recognition Application Model Workflow. . .



1. Define the Problem

Objective: What needs to be recognized? (e.g., objects in images, keywords in audio, gestures, equipment faults, or handwritten digits).

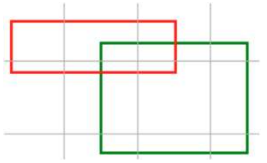
Type of Recognition Model:

a) Classification (assign input to a category)

b) Detection (locate and classify objects within input)

c) Identification/Verification (confirm identity or authenticity)

What Is a Recognition Application Model Workflow. . .



2. Data Collection

Gather datasets relevant to the problem:

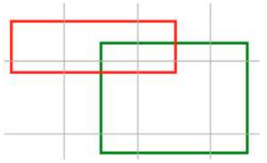
a) Images: Labeled photos for object or face recognition.

b) Text: Documents, speech transcripts, or labeled phrases.

c) Signals: Sensor readings, EEG, vibration data, etc.

Ensure data quality and variety (different lighting, backgrounds, noise conditions).

What Is a Recognition Application Model Workflow. . .

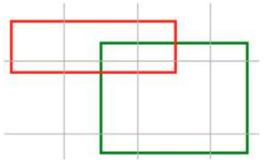


3. Data Preprocessing
   a) Cleaning: Remove duplicates, fix corrupted samples, normalize values.
   b) Augmentation: Generate variations (rotated images, noisy signals, paraphrased text) to improve robustness.
   c) Feature Engineering: Extract meaningful features (e.g., MFCC for speech, edges for vision, statistical measures for signals).
   d) Labeling: Ensure accurate, consistent annotations (often the bottleneck in recognition workflows).

What is MFCC?

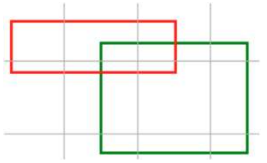
Mel-Frequency Cepstral Coefficients are numerical features extracted from audio signals that represent the sound's spectral characteristics in a way that mimics human hearing.

What Is a Recognition Application Model Workflow. . .



4. Model Selection
   Traditional Methods (feature-based):
      a) K-Nearest Neighbors (KNN)
      b) Support Vector Machines (SVM)
      c) Hidden Markov Models (HMMs)
   Deep Learning Methods:
      a) Convolutional Neural Networks (CNNs) → Image/object recognition
      b) Recurrent Neural Networks (RNNs), Transformers → Speech/text recognition
      c) Hybrid architectures → Multimodal recognition (e.g., vision + language)

What Is a Recognition Application Model Workflow. . .



5. Model Training

Split dataset:

a) Training,

b) Validation,

c) Testing (common ratios: 70/15/15).

- Choose loss function (e.g., cross-entropy for classification).
- Use optimization algorithm (SGD, Adam).
- Monitor metrics (accuracy, F1-score, precision, recall, AUC).
- Apply regularization (dropout, weight decay, data augmentation).

What is SGD?

Stochastic Gradient Descent (SGD) is an iterative optimization algorithm used in machine learning to train models by minimizing a loss function

What is Adam?

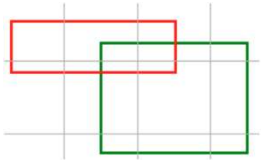
Adaptive Moment Estimation (Adam) is a popular and efficient optimization method used to train deep learning models by adjusting each parameter's learning rate.

Question 2

Which statement is correct about a Stochastic Gradient Descent?

a) An interactive optimization algorithm used in AL
b) An interactive optimization algorithm used in AI/ML.
c) An interactive optimization algorithm used in ML.
d) none of the above

What Is a Recognition Application Model Workflow. . .



6. Model Evaluation

- Compare model predictions with ground truth labels.
- Use confusion matrices for classification problems.
- Measure robustness under variations:
  - a) noise
  - b) distortion,
  - c) adversarial conditions.
- Perform cross-validation for stability check.

What Is a Recognition Application Model Workflow. . .

Confusion Matrix Example

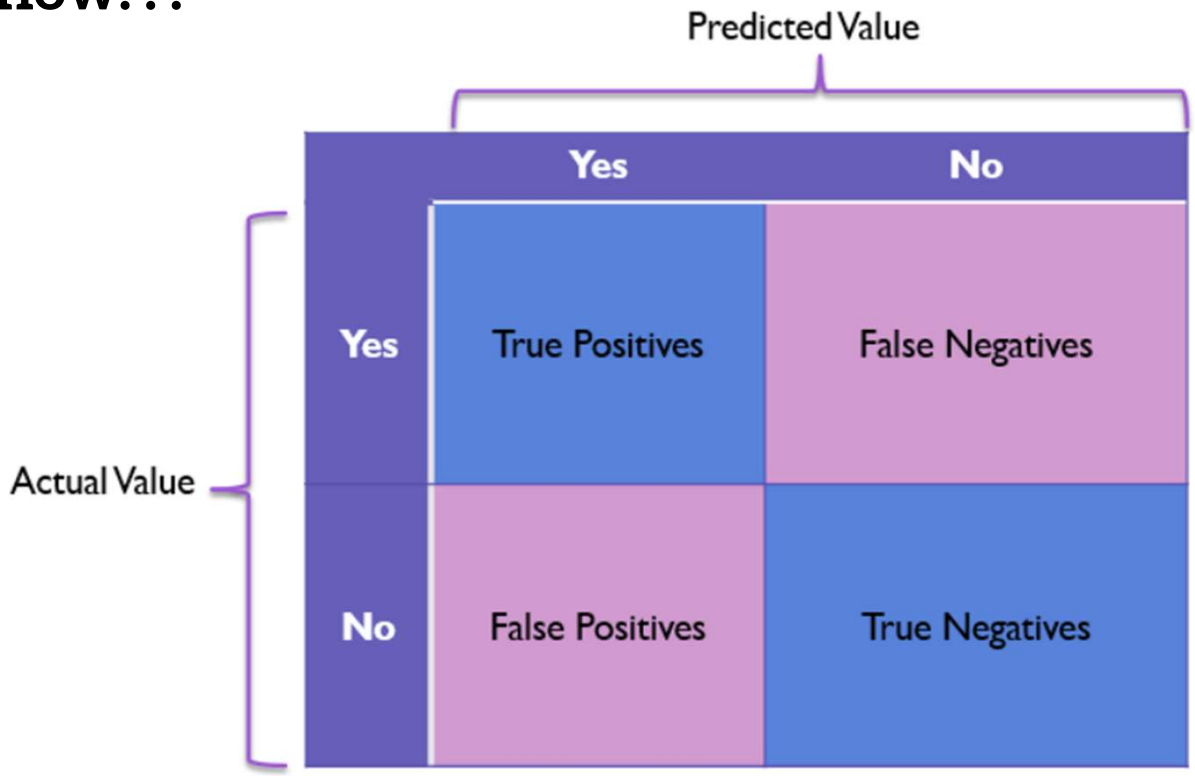
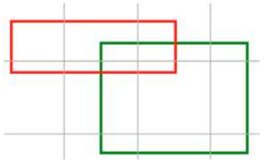


Image: Accessible AI

What Is a Recognition Application Model Workflow. . .



7. Model Deployment

- Export trained model into production environment:
  - a) On-device (edge AI): Lightweight, optimized models (TensorFlow Lite, ONNX).
  - b) Cloud-based: REST APIs for large-scale recognition.
- Integrate with user applications (e.g., mobile app, industrial monitoring dashboard).

What is an ONNX?

Open Neural Network Exchange (ONNX) is a standard format and open standard for machine learning models that allows for interoperability between different deep learning frameworks like PyTorch and TensorFlow. It provides a common, intermediate representation (IR) for models, making them portable and deployable across various platforms and devices, and includes a high-performance ONNX Runtime for efficient deployment and inference.

What Is a Recognition Application Model Workflow. . .

ONNX Interoperability

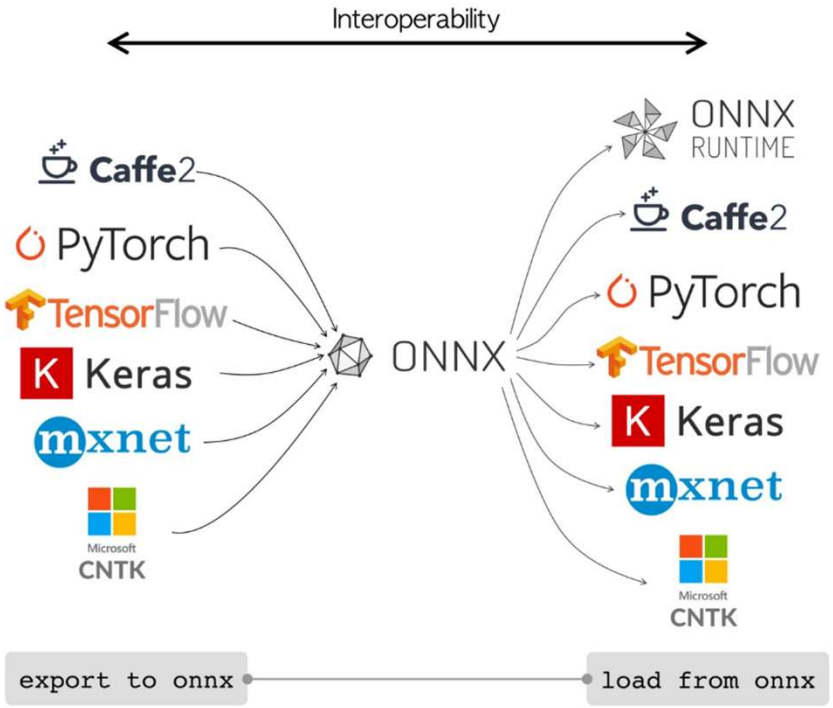
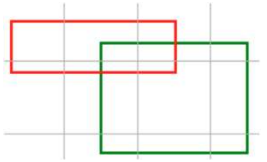


Image: ONNX

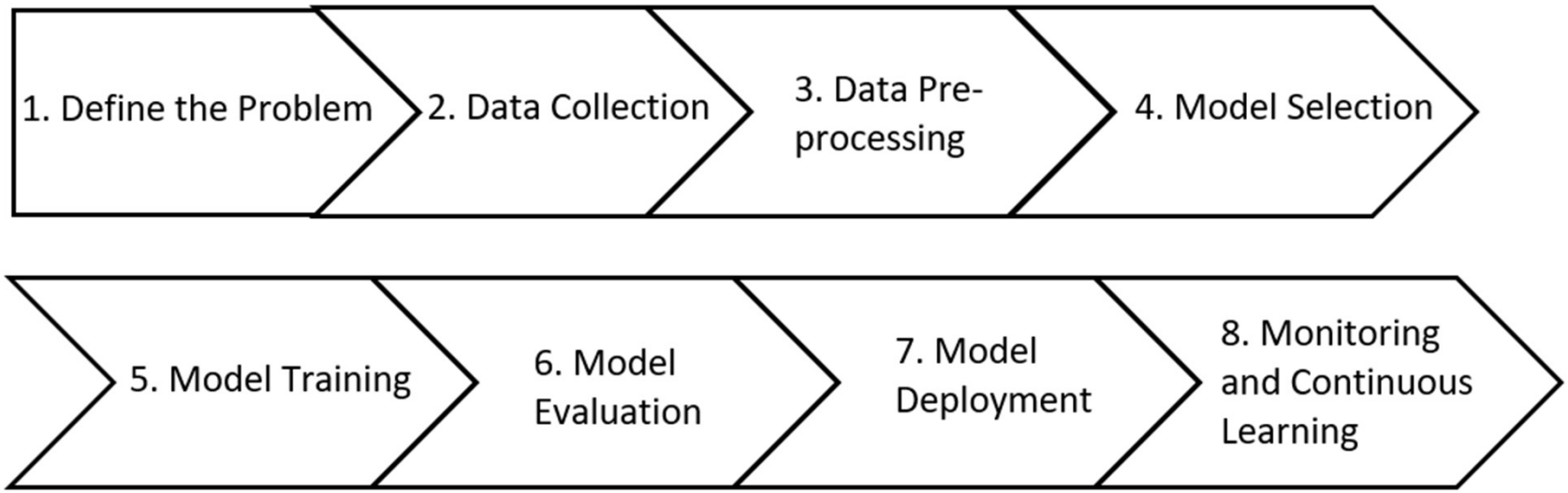
What Is a Recognition Application Model Workflow. . .



8. Monitoring & Continuous Learning
    a) Monitor real-world performance (drift in data distribution).
    b) Collect feedback to update training datasets.
    c) Re-train periodically with new data to maintain accuracy.

What Is a Recognition Application Model Workflow. . .

Recognition Application Model Workflow Visual Diagram



Question 3

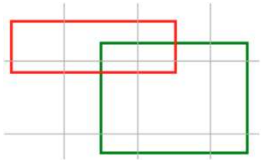
In reviewing slide 24, which step is responsible for integrating with user applications?

a) Data Collection
b) Data Preprocessing
c) Model Evaluation
d) Model Deployment

Microsoft MakeCode Python Smart AI Lens APIs

The Microsoft MakeCode Python Smart AI Lens APIs enable the Smart AI-Lens to function as a multi-mode recognition sensor (objects, faces, gestures, traffic signs, colors, tags, and more).

Microsoft MakeCode Python Smart AI Lens APIs. . .



📌 Smart AI-Lens APIs

1. General Setup

- `PlanetX_AILens.init()`
  - Initialize the Smart AI-Lens module.
- `PlanetX_AILens.switch_function(mode: AILensFuncList)`
  - Switch AI-Lens to a specific recognition mode.
  - `mode` is selected from the **AILensFuncList enum** (see below)

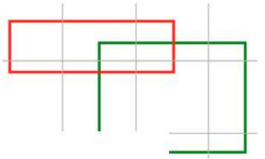
2. Image Capture

- `PlanetX_AILens.get_image()`
  - Capture an image frame for processing.

3. Object Recognition

- `PlanetX_AILens.learn_object(id: number)`
  - Train AI-Lens to recognize an object with an assigned ID.
- `PlanetX_AILens.get_learn_data() -> List[number]`
  - Retrieve learned object recognition results.

Microsoft MakeCode Python Smart AI Lens APIs. . .



4. Face & Gesture Recognition

- `PlanetX_AILens.face_recognize() -> number`
  - Detect faces and return result codes.
- `PlanetX_AILens.gesture_recognize() -> number`
  - Detect hand gestures.

5. Color & Tag Recognition

- `PlanetX_AILens.color_recognize() -> number`
  - Detect basic colors (red, green, blue, etc.).
- `PlanetX_AILens.tag_recognize() -> number`
  - Detect AprilTags / markers.

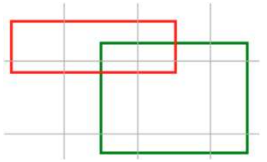
6. Traffic & Card Recognition

- `PlanetX_AILens.traffic_recognize() -> number`
  - Recognize traffic signs.
- `PlanetX_AILens.card_recognize() -> number`
  - Recognize number or symbol cards.

7. Line Following & Ball Detection

- `PlanetX_AILens.line_recognize() -> number`
  - Line-following mode.
- `PlanetX_AILens.ball_recognize() -> number`
  - Detect balls (used in robotics navigation).

Microsoft MakeCode Python Smart AI Lens APIs. . .



📌 Enum Types

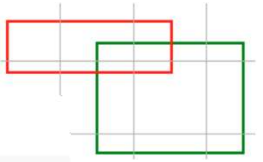
AILensFuncList

Used with `switch_function()` to select modes:

- `Learn`
- `Face`
- `Gesture`
- `Color`
- `Tag`
- `Traffic`
- `Card`
- `Line`
- `Ball`

Microsoft MakeCode Python Examples

Microsoft MakeCode Python Smart AI Lens APIs. . .



1. Initialization

```python
from PlanetX_AILens import *

# Initialize Smart AI-Lens
PlanetX_AILens.init()
```

2. Switch Function Mode

```python
from PlanetX_AILens import *

PlanetX_AILens.init()
# Switch AI-Lens to Face Recognition mode
PlanetX_AILens.switch_function(AILensFuncList.Face)
```

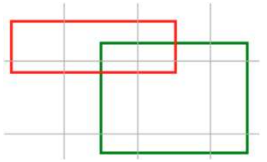
3. Image Capture

```python
from PlanetX_AILens import *

PlanetX_AILens.init()
PlanetX_AILens.switch_function(AILensFuncList.Learn)

while True:
    # Capture frame
    PlanetX_AILens.get_image()
```

Microsoft MakeCode Python Smart AI Lens APIs. . .



Microsoft MakeCode Python Examples

4. Object Recognition

```python
from PlanetX_AILens import *

PlanetX_AILens.init()
PlanetX_AILens.switch_function(AILensFuncList.Learn)

# Teach the lens object ID 1
PlanetX_AILens.learn_object(1)

# Get recognition results
result = PlanetX_AILens.get_learn_data()
if result[0] == 1:
    display.show(Image.HAPPY)
else:
    display.show(Image.SAD)
```

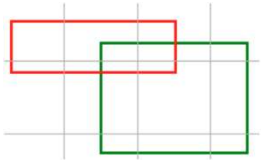
5. Face Recognition

```python
from PlanetX_AILens import *

PlanetX_AILens.init()
PlanetX_AILens.switch_function(AILensFuncList.Face)

while True:
    if PlanetX_AILens.face_recognize() == 1:
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

Microsoft MakeCode Python Smart AI Lens APIs. . .



Microsoft MakeCode Python Examples

6. Gesture Recognition

python

```python
from PlanetX_AILens import *

PlanetX_AILens.init()
PlanetX_AILens.switch_function(AILensFuncList.Gesture)

while True:
    gesture = PlanetX_AILens.gesture_recognize()
    if gesture == 1:
        display.show(Image.ARROW_N)
    elif gesture == 2:
        display.show(Image.ARROW_E)
```

7. Color Recognition

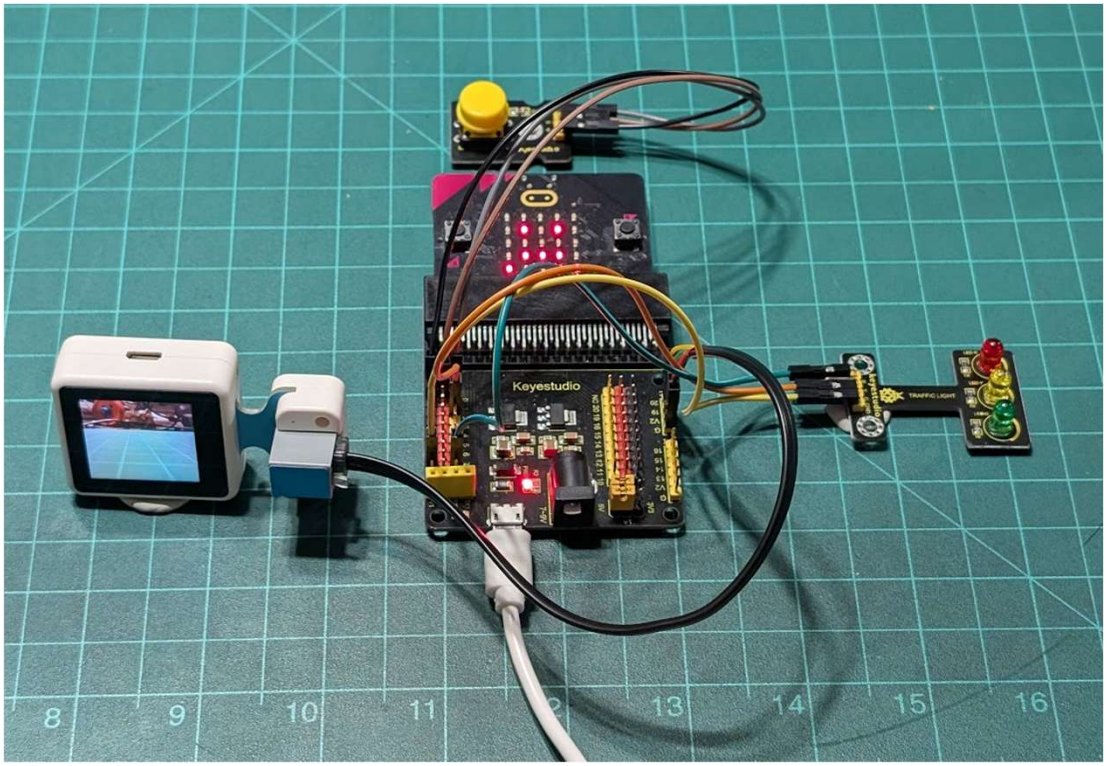
python

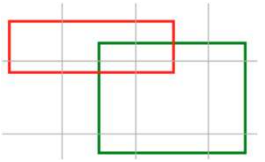
```python
from PlanetX_AILens import *

PlanetX_AILens.init()
PlanetX_AILens.switch_function(AILensFuncList.Color)

while True:
    color = PlanetX_AILens.color_recognize()
    if color == 1:
        display.show(Image.HEART)  # Red
    elif color == 2:
        display.show(Image.HAPPY)  # Green
```

Lab: Enhanced Characteristics Learn Device





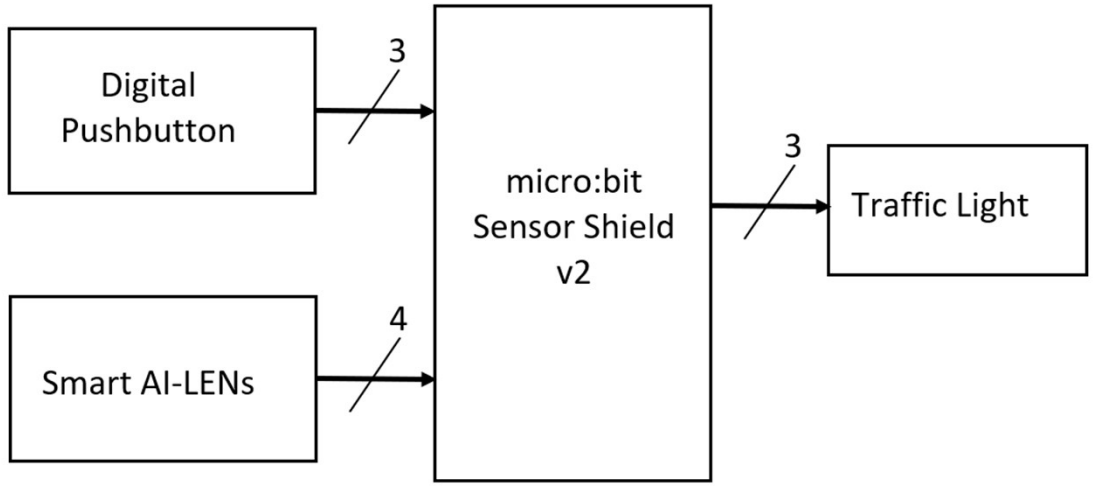
Lab: Enhanced Characteristics Learn Device

Participant Learning Objectives:

- Participants will learn to wire the Enhanced Characteristics Learn Device.
- Participants will learn how to use the Microsoft MakeCode Python to build the Enhanced Characteristics Learn Device.
- Participants will learn to implement a learning event on recognizing an object using the Smart AI-Lens to Micro:bit Sensor Shield.
- Participants will learn to view the Confidence Level on the Characteristic Learn Device.

Lab: Enhanced Characteristics Learn Device. . .

Concept Block Diagram for an Enhanced Characteristics Learn Demonstrator

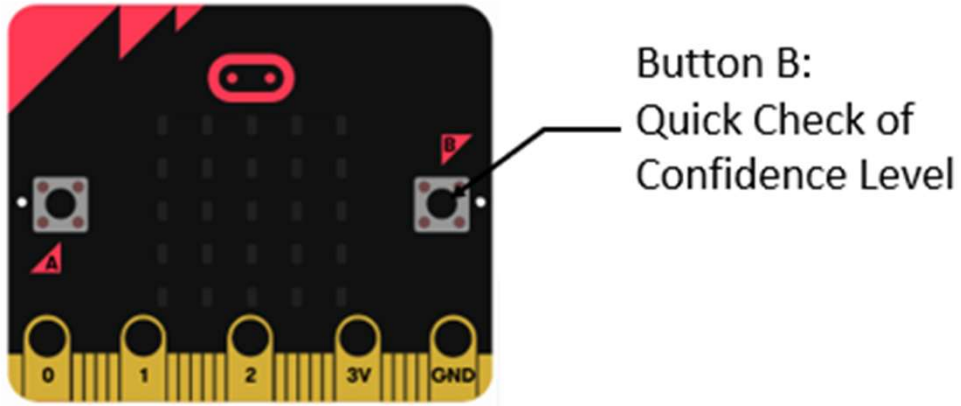


Operation:

A press/release on Digital Pushbutton, will allow the Smart-AI Lens to capture and store an object's image. Placing the object in front of the Smart-AI Lens will allow the micro:bit to display a Smiley face and turn on the Green LED. No object present or the incorrect object will display a Sad face on the micro:bit's 5x5 LED matrix and turn off the Green LED.

Lab: Enhanced Characteristics Learn Device. . .

User Interface (UI) of Enhanced Characteristics Learn Device

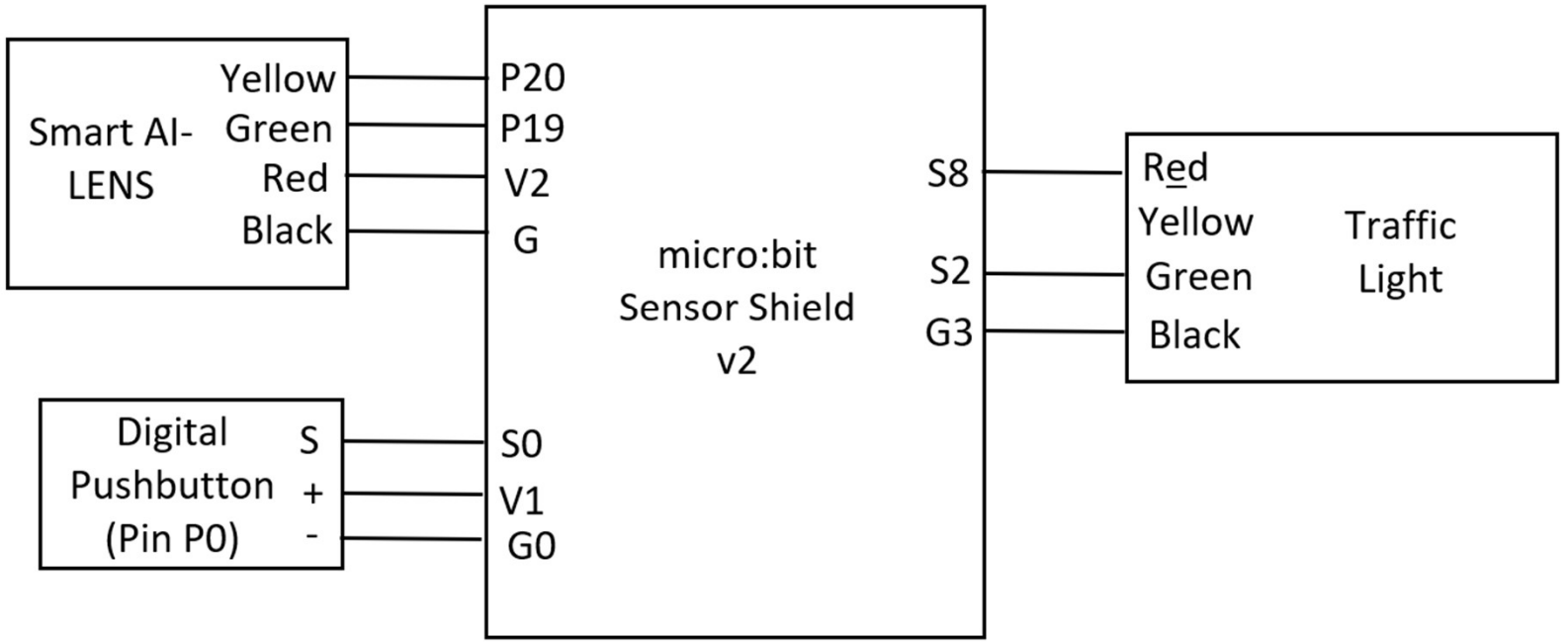


Operation:

Pressing Button B will display a confidence level (in percentage value) on recognizing the image/object placed in front of the Smart-AI Lens.

Lab: Enhanced Characteristics Learn Device. . .

Electrical Wiring Diagram



Question 4

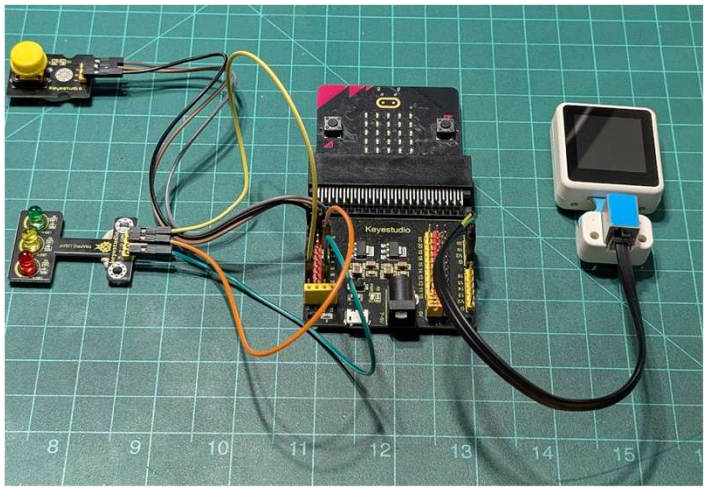
How many wires are required to attach the Traffic Light to the micro:bit sensor shield?

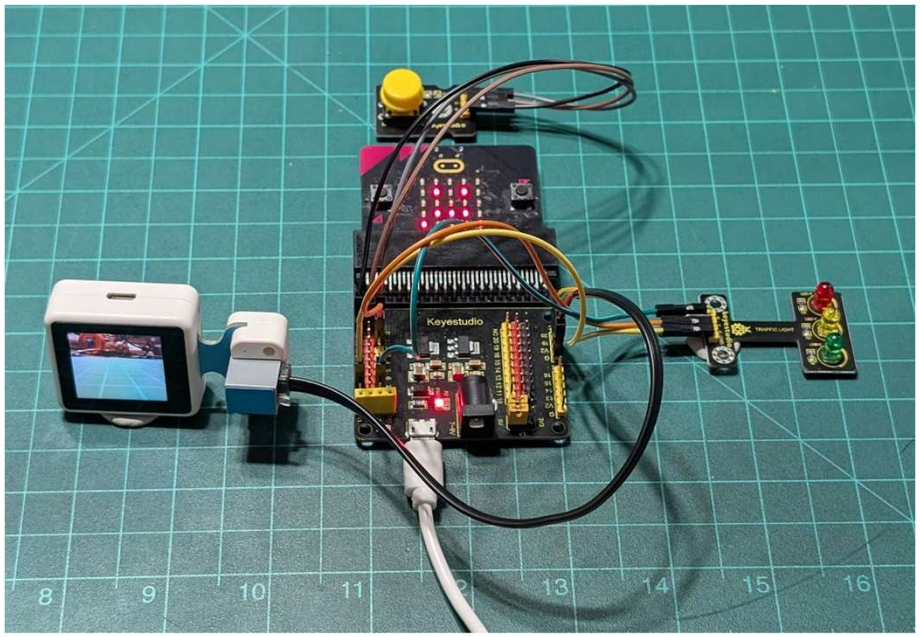
a) 4
b) 3
c) 2
d) none of the above

Lab: Enhanced Characteristics Learn Device. . .

Enhanced Characteristics Learn Device Demonstrator: Final Wired Device

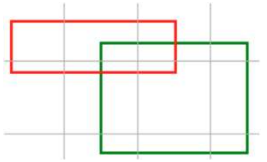
Smart AI-Lens, Digital Pushbutton Switch, and Traffic Light wired to the Micro:bit Sensor Shield





Powered ON Enhanced Characteristics Learn Device

Lab: Enhanced Characteristics Learn Device. . .

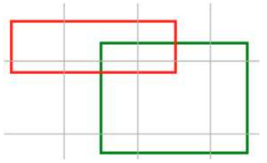


Setup Procedure

1. Go to Microsoft MakeCode: micro:bit and open a new project.
2. Click on the Python button to open a new editor.
3. Copy and paste the Enhanced Characteristics Learn Python code from the text file into the editor.
4. Add the PlanetX-AI extension in the Make Code Python project before compiling. Note: The extension provides the PlanetX-AI Lens API to the programming environment.
5. Attach the micro:bit to the code development laptop computer or desktop PC USB port.
6. Download the Enhanced Characteristics Learn code to the micro:bit (Click the Download button)
7. Insert the micro:bit into the Sensor Shield. Attach the USB cable to the Sensor Shield's USB connector.

Learn Mode

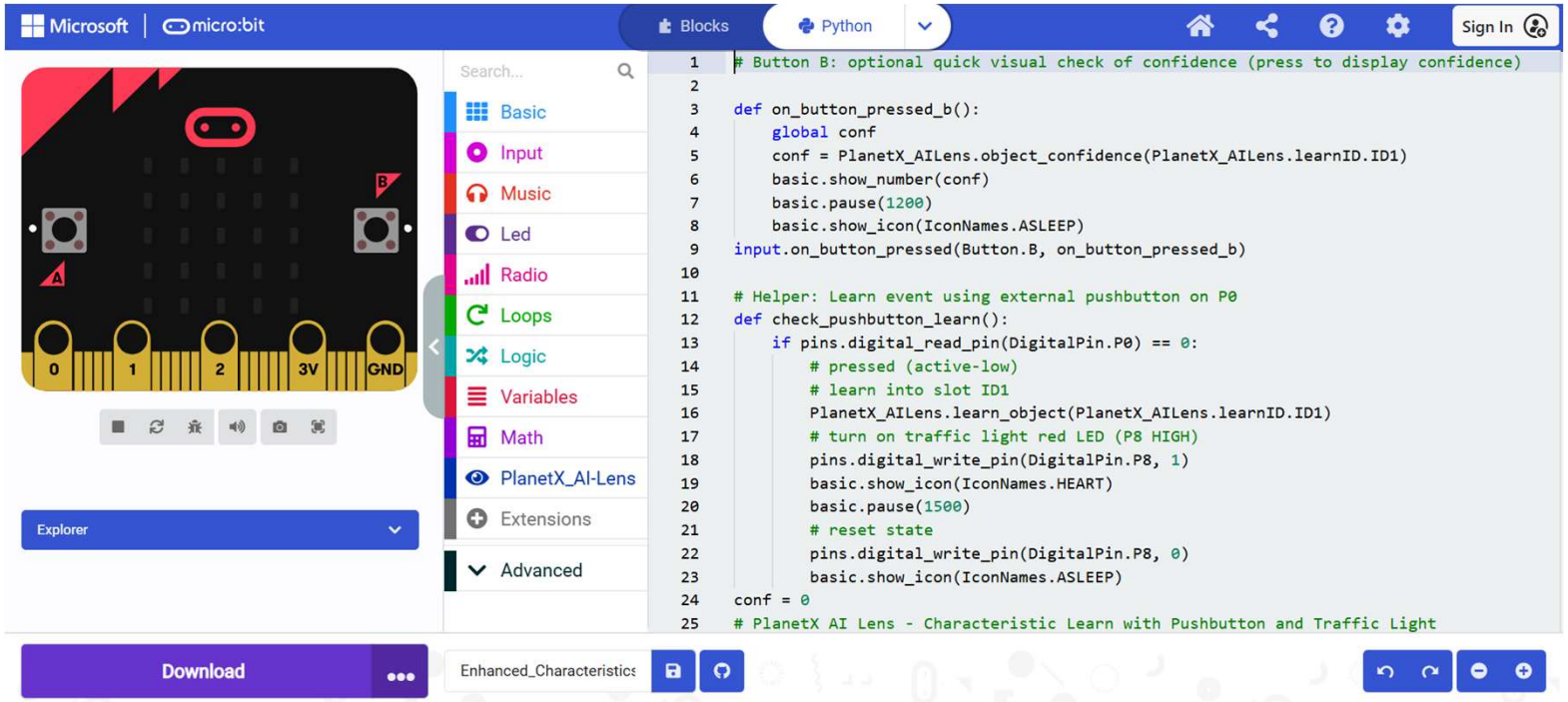
Lab: Enhanced Characteristics Learn Device. . .



1. Place an object in front of the Smart AI-Lens.
2.Press/release Digital Pushbutton. A heart will be displayed on the micro:bit's 5x5 LED display, and the Red LED will turn on briefly.
   Note: the object is stored as an image on the micro:bit.
2. Press Button B. The number 79 will scroll on the LED matrix. This number is the confidence level..
4. Place the learned object from Step 1 in front of the Smart AI-Lens.
5. A smiley face will be displayed on the micro:bit's LED matrix. The Green LED will turn on.

Lab: Enhanced Characteristics Learn Device. . .

Enhanced Characteristics Learn Python Code pasted onto the editor

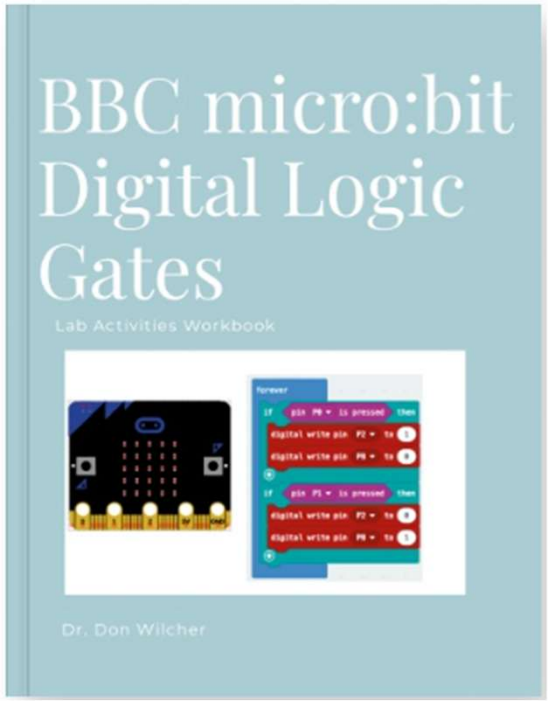


Lab: Enhamced Characteristics Learn Device. . .

BBC Micro:bit Digital Logic Gate Lab Activities Workbook Challenge

Modify the Enhanced Characteristics Learn Device code, where pressing a crash sensor will turn display a check mark on an external red LED.

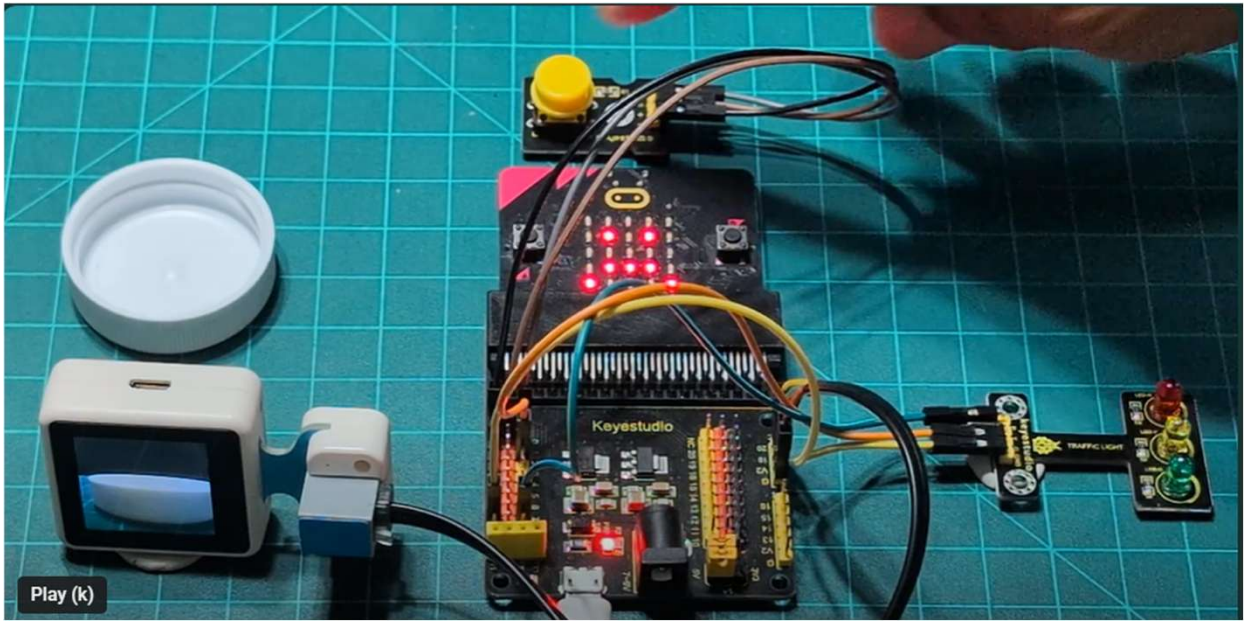
Page 34 provides an electrical wiring diagram to attach the crash sensor to the sensor shield.



Lab: Enhanced Characteristics Learn Device. . .

Assembled and Functionally Enhanced Characteristics Learn Device Demonstrator

Watch the Video Clip!



https://www.youtube.com/watch?v=PqCyLkF2OPI

Question 5

Which statement is incorrect?

a) Press/release Crash Sensor, a heart will be displayed on the micro:bit.

b) Press Button B, and the number 79 will scroll on the LED matrix.

c) A smiley face will be displayed on the micro:bit's LED matrix.

d) none of the above

Thank you for attending

Please consider the resources below:

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *arXiv*:1506.02640 [cs.CV], Jun. 2016. [Online]. Available: https://arxiv.org/abs/1506.02640
[2] [1] D. Wilcher, “Designs News September 25 webinar code,” GitHub repository, Sep. 2025. [Online]. Available: https://github.com/DWilcher/DesignNews-WebinarCode/blob/main/September_25_Webinar_Code.zip

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets