DOI: https://doi.org/10.62985/j.huit_ojs.vol26.no2E.416

ADAPTIVE COLOR SORTING ROBOT APPLYING MACHINE LEARNING ON EMBEDDED SYSTEMS

Dan-Huy Truong[1], Hien-Nam Phan Phuc[1], Thanh-Tuan Nguyen[1,*], Xuan-Huy Nguyen[1], Mong-Fong Horng[2], Chin-Shiuh Shieh[2], Thanh-Lam Nguyen[2]

[1]*Faculty of Electrical and Electronics, Nha Trang University, Nha Trang 650000, Vietnam*

[2] *Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 807618, Taiwan*

*Email: *tuantn@ntu.edu.com*

Received: 12 August 2025; Revised: 18 March 2026; Accepted: 4 April 2026

ABSTRACT

Automation in manufacturing relies heavily on accurate product classification. However, affordable color sensors often suffer from interference caused by ambient lighting changes, leading to errors in traditional systems based on fixed thresholds. This study presents an adaptive sorting robot arm integrated with a Programmable Logic Controller and an embedded microcontroller. The novelty of this research lies in the implementation of the K Nearest Neighbors machine learning algorithm directly on the microcontroller to process color data. Instead of using fixed logic, the system features a calibration mode where it learns reference color features in the current environment. During operation, the algorithm calculates the Euclidean distance between the detected object and the learned samples to perform classification. Experimental results demonstrate that this approach significantly improves recognition accuracy under varying lighting conditions compared to methods using static logic, offering a robust and economical solution for industrial automation.

Keywords: Adaptive Color Sorting, K-Nearest Neighbors, Embedded Machine Learning, PLC, Industrial Automation.

1. INTRODUCTION

The rapid progression of the Fourth Industrial Revolution has established automation as a primary foundation of modern manufacturing [1]. In this context, industrial robotic arms play a crucial role in enhancing productivity and precision across various production stages [2]. Among the control units used in these systems, Programmable Logic Controllers (PLCs) are widely preferred in industrial environments due to their high stability, superior noise immunity, and durability under harsh operating conditions [3]. Consequently, integrating robotic mechanisms with PLCs remains a standard approach for automated tasks involving material handling and product sorting. This combination ensures that the mechanical operation remains robust even in environments with electrical noise or vibration.

Product classification based on color is a fundamental requirement in many industries, particularly in food processing, packaging, and logistics. To minimize implementation expenses, especially for small and medium enterprises, engineers often utilize affordable color sensors such as the TCS3200 [4]. This sensor operates by converting light intensity into frequency, which a microcontroller can read and process. However, a significant challenge involves the sensitivity of these sensors to environmental lighting changes. Traditional control methods typically rely on fixed threshold logic. In this approach, the

programmer manually defines specific ranges for Red, Green, and Blue values in the source code. This method lacks flexibility, leading to frequent classification errors when the ambient light intensity fluctuates or when the sensor position shifts slightly. Furthermore, whenever the physical environment changes, the system requires manual recalibration and code updates, causing unwanted system downtime.

Existing research in the field of automatic color sorting typically follows two main directions. The first direction employs Computer Vision techniques using cameras and advanced image processing algorithms, such as OpenCV or deep learning models like YOLO [5]. While this method offers high accuracy and versatility, it requires expensive hardware and substantial computational power, such as a Raspberry Pi or an industrial computer [6]. This makes the Computer Vision approach unsuitable for economical embedded applications where cost and power consumption are critical constraints. The second direction utilizes discrete color sensors interfaced with microcontrollers. Although this approach is affordable, it traditionally suffers from low stability. Most studies in this category rely on static logic statements that fail to adapt to new conditions, resulting in lower reliability compared to systems utilizing cameras.

To address these limitations, this paper proposes an adaptive color sorting robot system that combines the industrial reliability of a PLC with the flexibility of an embedded microcontroller. The primary contribution of this study is the implementation of the K Nearest Neighbors (KNN) machine learning algorithm directly on an 8 bit Arduino microcontroller. Unlike complex models requiring heavy processing, KNN is a lightweight algorithm suitable for limited hardware resources [7]. Instead of relying on rigid conditional logic, the proposed system features a calibration mode where it learns reference color features in the actual operating environment. By calculating the Euclidean distance between the detected object and the learned samples, the system can adapt to lighting variations effectively. This hybrid architecture ensures both the robust motion control provided by the PLC and the intelligent signal processing capability of the microcontroller, offering a practical solution for low budget industrial automation.

2. SYSTEM HARDWARE DESIGN

To achieve the objective of a stable yet economical color sorting system, the hardware design is divided into two main subsystems: the mechanical structure and the electrical control unit. The overall architecture follows a Master Slave topology, where the PLC manages the sequence of operations while the microcontroller handles sensor data processing. The robot arm is designed with 3 degrees of freedom to perform pick and place tasks effectively. Instead of using heavy gears, the transmission system utilizes synchronous timing belts. This design choice reduces the inertia of the moving parts and minimizes mechanical noise during operation. The rotational motion from the stepper motors is converted into linear motion or arm rotation through a transmission ratio of 1:4.5. This ratio helps increase torque and ensures the robot moves with high precision. As illustrated in Figure 1, the mechanical frame is constructed to ensure rigidity. The end effector is equipped with a pneumatic suction cup, allowing the robot to grip objects with flat surfaces securely.

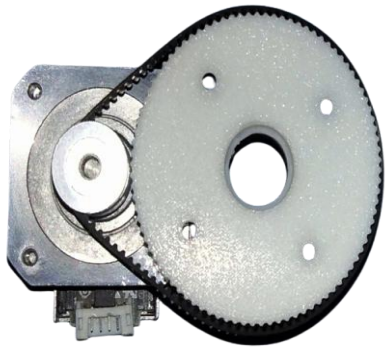

Fig. 1. Mechanical structure of the robot arm using belt transmission

The control center of the system is the Samkoon PLC - FAS 32MT, which offers high immunity to industrial electrical noise. The PLC acts as the Master controller, responsible for generating pulses to drive the stepper motors and managing the pneumatic valves. For color detection, the system employs the TCS3200 sensor. This sensor consists of an array of photodiodes with red, green, blue, and clear filters [8]. It converts light intensity into a square wave frequency. Since the PLC inputs are designed for standard industrial logic levels and are not optimized for reading high frequency signals from the TCS3200 directly, an Arduino microcontroller is introduced as an intermediate processing unit. The connection diagram of the system is depicted in Figure 2. The Arduino reads the frequency from the TCS3200, processes the data using the proposed algorithm, and sends the result to the PLC via digital output pins. Optical isolators are used between the Arduino and the PLC to protect the microcontroller from high voltage surges.

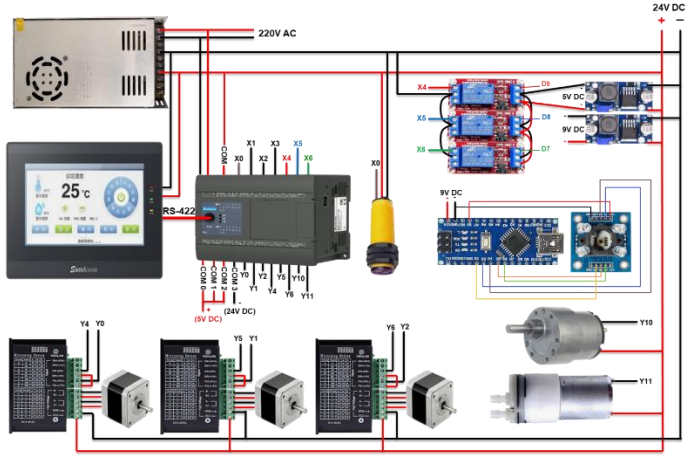

Fig. 2. Diagram of the control system connection

The selection of specific hardware components is critical to achieving the design goals of stability and cost efficiency. The system integrates standard industrial equipment with accessible embedded devices to form a cohesive unit. The Programmable Logic Controller serves as the robust core for logic operations, ensuring durability in industrial environments, while the microcontroller bridges the gap between complex sensor data and digital control logic. Additionally, the chosen stepper motors and drivers provide precise positioning capabilities required for the sorting task without the high expense associated with servo systems. To provide a clear and comprehensive overview of the experimental setup, the

detailed technical specifications and the functional roles of each key component are summarized in Table 1.

Table 1. Technical specifications of the proposed system

Component	Model	Key Specification	Function
Central Controller	Samkoon FAS 32MT	16 Inputs / 16 Outputs, 24V DC	Master logic control
Microcontroller	Arduino Uno R3	ATmega328P, 16 MHz Clock	Sensor data processing
Color Sensor	TCS3200	4 channel photodiode array	Color to frequency conversion
Actuators	Stepper Motor 57	Step angle 1.8 degrees	Robot arm motion
Motor Driver	TB6600	9V to 42V DC supply, 4A Peak	Motor current control
Transmission	GT2 Timing Belt	Pitch 2mm, Width 6mm	Power transmission

3. CONTROL STRATEGY AND ALGORITHM

The efficacy of the proposed system relies on the seamless coordination between the signal processing unit and the motion control unit. This section details the mathematical formulation of the adaptive classification algorithm and the kinematic control logic governing the robotic actuation.

3.1. Adaptive Color Recognition Algorithm

The core innovation of this study is the replacement of static conditional logic with a distance based classification method. We model the color recognition problem within a three dimensional Euclidean space.

Let S be the color space where every distinct color is represented as a vector. The input vector x captured by the sensor is defined by its Red (R), Green (G), and Blue (B) intensity components:

$$x = [R, G, B]^T \tag{1}$$

During the calibration phase, the system establishes a set of reference classes $C = \{c_1, c_2, \ldots, c_k\}$, where k represents the number of target colors. For each class c_i, the system computes a centroid vector μ_i based on N sample measurements learned from the environment:

$$\mu_i = \frac{1}{N} \sum_{j=1}^{N} x_{i,j} \tag{2}$$

Where $x_{i,j}$ denotes the j^{th} sample vector belonging to color class i. This centroid vector μ_i serves as the adaptive reference point for that specific lighting condition. In the classification phase, when an unknown object with feature vector x_{new} is detected, the algorithm calculates the Euclidean distance $d(x_{new}, \mu_i)$ between the input vector and each reference centroid [9]. The distance metric is expressed as:

$$d(x_{new}, \mu_i) = \| x_{new} - \mu_i \|_2 = \sqrt{(R_{new} - \bar{R}_i)^2 + (G_{new} - \bar{G}_i)^2 + (B_{new} - \bar{B}_i)^2} \quad (3)$$

The decision rule follows the Nearest Neighbor principle. The system assigns the label L to the object by identifying the class that minimizes the calculated distance [10]:

$$L = *arg\,min_{c_i \in C} \{d(x_{new}, \mu_i)\} \quad (4)$$

This mathematical approach ensures that the classification is based on relative similarity rather than absolute thresholds, providing robustness against linear shifts in light intensity. The complete flow of this adaptive algorithm is illustrated in Figure 3.

Algorithm 1 Adaptive Color Classification using K-NN Algorithm

Input: Sensor raw values $x = [R, G, B]^T$
Output: Classification Label $L \in \{Red, Green, Blue, None\}$
Variables: μ_i (Centroid of class i), d_i (Euclidean distance)

procedure SystemCalibration ▷ Run once at startup
  for each color class c_i in $\{Red, Green, Blue\}$ **do**
    Place sample object of color c_i
    Collect N samples $S = \{x_1, x_2, ..., x_N\}$
    Compute Centroid: $\mu_i \leftarrow \frac{1}{N} \sum_{j=1}^{N} x_j$ ▷ Learn reference features
    Store μ_i in EEPROM
  end for
end procedure

procedure MainLoop ▷ Continuous operation
  loop
    $x_{new} \leftarrow ReadTCS3200()$
    if x_{new} indicates Empty Belt **then**
      $L \leftarrow None$
    else
      $d_{min} \leftarrow \infty$
      for each color class c_i **do**
        Calculate distance: $d_i \leftarrow \|x_{new} - \mu_i\|_2$
        if $d_i < d_{min}$ **then**
          $d_{min} \leftarrow d_i$
          $L \leftarrow c_i$
        end if
      end for
      Output Control signal for class L to PLC
      Activate corresponding LED indicator
    end if
    Delay for stable reading
  end loop
end procedure

Fig. 3. Adaptive color recognition algorithm

3.2. Kinematics and Control Logic Implementation

Upon determining the object label, the microcontroller transmits the result to the PLC to execute the physical sorting task. The motion control is governed by the relationship between the stepper motor rotation and the linear displacement of the timing belt. Let θ_{step} be the fundamental step angle of the motor and M be the micro step resolution setting of the driver. The angular rotation per pulse α is given by:

$$\alpha = \frac{\theta_{step}}{M} \quad (5)$$

The transmission system converts this angular motion into linear displacement via a pulley with radius r. The relationship between the number of control pulses P and the target travel distance D_{target} is derived as follows:

$$P = \frac{D_{target}}{2\pi r \eta} \times \frac{360}{\alpha} \quad (6)$$

Where η represents the mechanical transmission ratio. In this system, the PLC utilizes this kinematic model to generate the exact number of pulses required to move the robot arm from the home position to the designated sorting bin. The control sequence ensures synchronization between the conveyor belt and the robotic arm. When the sensor detects an object, the PLC triggers a hardware interrupt to halt the belt, executes the pick and place routine defined by P, and subsequently resets the system. The detailed control logic is depicted in Figure 4.

Algorithm 2 PLC Motion Control and Actuation Sequence

Input: Classification Signal L from Arduino, Position Sensor S_{pos}
Output: Physical Sorting Operation
Initialize: Execute Homing Procedure ($P \leftarrow 0$)
Start: Activate Conveyor Belt **and** Status Light
while System is Active **do**
  if Stop Button is Pressed **then**
    Halt Conveyor **and** Terminate Operation
  end if
  if $L \neq None$ **and** S_{pos} is Active **then**
    Stop Conveyor Belt immediately
    ▷ Determine target pulses based on color signal
    $P_{target} \leftarrow$ GetTargetCoordinates(L)
    ExecutePickAndPlace(P_{target})
    Resume Conveyor Belt
    Reset Signal L
  end if
end while
procedure ExecutePickAndPlace($Target$)
  Generate pulses to move Arm to Pickup Position
  Activate Suction Cup (Vacuum ON)
  Generate pulses to move Arm to $Target$ Bin
  Deactivate Suction Cup (Vacuum OFF)
  Return Arm to Home Position
end procedure

Fig. 4. Algorithm of the robot control logic implemented on the PLC

4. EXPERIMENTAL RESULTS AND DISCUSSION

Before detailing the experimental data, it is essential to describe the physical realization of the proposed system used for testing. The complete experimental setup is depicted in Figure 5. The system integrates the PLC control cabinet responsible for overall logic and power distribution, the customized 3-DOF robotic arm with belt transmission mechanisms, a conveyor belt system for product feeding, and the TCS3200 color sensor station interfaced with the Arduino microcontroller.

Fig. 5. The physical experimental model

In the first phase, the system operates in the calibration mode to establish the reference feature vectors. We placed standard objects (Red, Green, Blue) and an Empty

belt condition under the sensor to record the RGB frequency response. The sensor was configured with a 20 percent output frequency scaling to match the processing capability of the Arduino microcontroller.

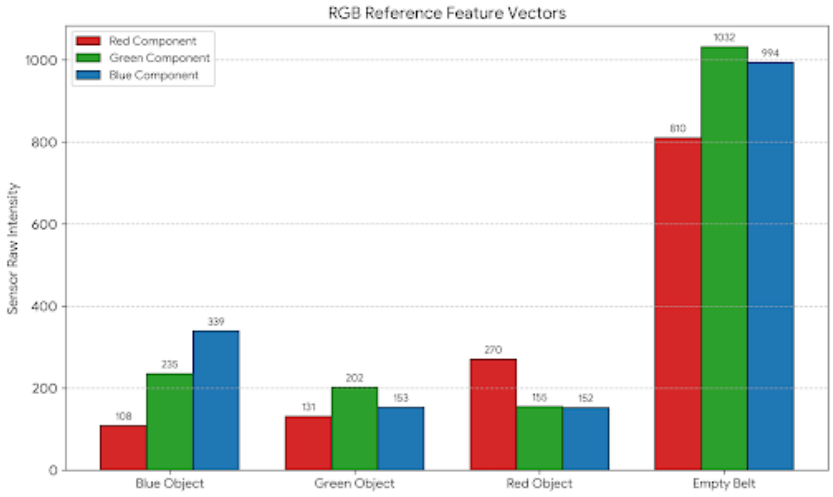

Fig. 6. Average RGB reference values obtained during calibration.

As observed in Figure 6, the Euclidean distance between these color vectors is sufficiently large, allowing the algorithm to distinguish them clearly. However, relying solely on average values can be misleading because industrial sensor readings often fluctuate due to noise. To demonstrate how the K-NN algorithm handles these fluctuations, particularly between color classes with similar characteristics, Figure 7 visualizes the calibration data in a 3D feature space.

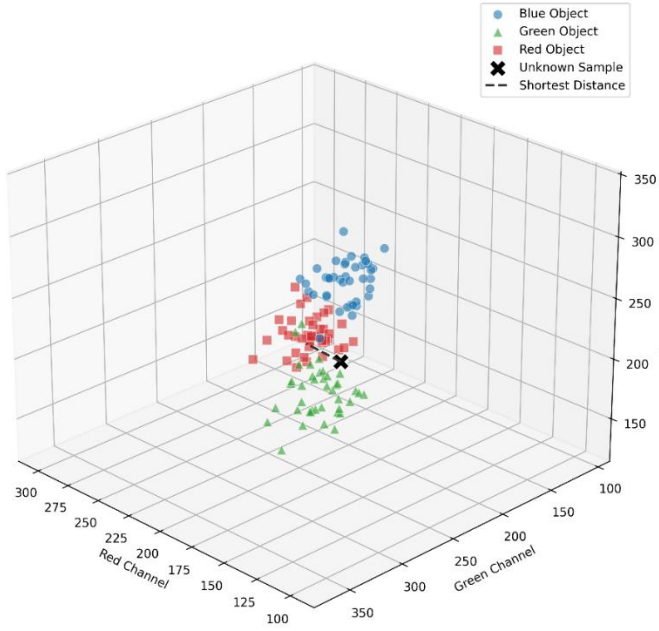

Fig. 7. 3D visualization of the K-NN feature space

To demonstrate the superiority of the proposed adaptive method over the traditional fixed threshold method, we set up a comparative experiment. The experiment compares two control logic scenarios:

- Scenario A (Fixed Logic): The robot uses hard-coded if statements with fixed boundaries derived from the calibration data.
- Scenario B (Adaptive KNN): The robot uses the Euclidean distance algorithm described in Section 3, using the calibrated centroids.

Both methods were tested in two environments:

Standard Light: The environment where the calibration was originally performed.

Altered Light: The ambient light intensity was reduced by approximately 20 percent to simulate a shadow or a change in time of day. Table 2 presents the success rate of the sorting task over 20 trials for each color in both scenarios.

Table 2. Accuracy comparison between fixed logic and proposed method.

Test Condition	Fixed Logic Accuracy	Proposed Method Accuracy
Standard Light	95.0%	96.6%
Altered Light	65.0%	91.6%

In the Standard Light condition, both methods perform well with high accuracy. However, in Altered Light conditions, the accuracy of the Fixed Logic method drops significantly to 65%. This failure occurs because the light intensity fluctuations cause the raw sensor values to shift beyond the hard-coded thresholds defined in the code.

In contrast, the Adaptive KNN method maintains a high reliability of 91.6%. This robustness is achieved because the algorithm evaluates the relative similarity (distance) between the color vectors rather than checking absolute values. Even when the light intensity changes, the correlation between the RGB components remains consistent, allowing the system to reassign the object to the correct class by finding the nearest centroid. This result confirms that the proposed machine learning approach effectively compensates for environmental changes without requiring complex hardware shielding.

5. CONCLUSION

This paper presented the design and implementation of an adaptive color sorting robot arm that effectively integrates industrial reliability with embedded intelligence. By combining a Programmable Logic Controller for robust motion control and an Arduino microcontroller for flexible signal processing, the system offers a cost-efficient solution for small and medium enterprises. The primary contribution of this study is the successful application of the K Nearest Neighbors algorithm on a low resource 8-bit microcontroller. The experimental results demonstrated that the proposed adaptive method significantly outperforms traditional fixed threshold logic. Specifically, while the traditional method suffered a major drop in accuracy (falling to 65.0 percent) under altered lighting conditions, the adaptive algorithm maintained a high reliability of 91.6 percent. This confirms that the Euclidean distance-based classification approach provides superior robustness against environmental variations without the need for expensive hardware modifications like dark chambers. Future work will focus on optimizing the algorithm to further reduce the processing time and integrating Computer Vision capabilities to enable sorting based on product shape and size. Additionally, we plan to implement Industrial Internet of Things protocols to allow remote monitoring and data logging of the production process.

Acknowledgment: This research is partly funded by Nha Trang University (NTU), Vietnam under grant number “SV2024-13-60”.

REFERENCES

[1] L. D. Xu, E. L. Xu, and L. Li, "Industry 4.0: State of the art and future trends," *Int. J. Prod. Res.*, vol. 56, no. 8, pp. 2941–2962, 2018, doi: https://doi.org/10.1080/00207543.2018.1444806.

[2] R. Rakholia, A. L. Suarez-Cetrulo, M. Singh, and R. Simon Carbajo, "Advancing Manufacturing Through Artificial Intelligence: Current Landscape, Perspectives, Best Practices, Challenges, and Future Direction," *IEEE Access*, vol. 12, pp. 131621–131637, 2024, doi: https://doi.org/10.1109/ACCESS.2024.3458830.

[3] E. R. Alphonsus and M. O. Abdullah, "A review on the applications of programmable logic controllers (PLCs)," *Renew. Sustain. Energy Rev.*, vol. 60, pp. 1185–1205, Jul. 2016, doi: https://doi.org/10.1016/j.rser.2016.01.025.

[4] A. R. M. Khairudin, M. H. A. Karim, A. A. Samah, D. Irwansyah, M. Y. Yakob, and N. M. Zian, "Development of colour sorting robotic arm using TCS3200 sensor," in *Proc. IEEE 9th Conf. Syst., Process Control (ICSPC)*, 2021, pp. 108–113, doi: https://doi.org/10.1109/ICSPC53359.2021.9689114.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788, doi: https://doi.org/10.1109/CVPR.2016.91.

[6] I. Alrushidy, Y. Bahumid, R. Bin Shujaa, A. Abdullah, A. Kurd, and M. Abdullah, "Design and fabrication of color sorting machine based on computer vision," *J. Sci. Technol.*, vol. 30, no. 6, pp. 107–115, May 2025, doi: https://doi.org/10.20428/jst.v30i6.2954.

[7] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-low-power Microcontrollers*. O'Reilly, 2020. [Online]. Available: https://books.google.com.vn/books?id=sB3mxQEACAAJ

[8] AMS-TAOS USA Inc., "TCS3200, TCS3210 Programmable Color Light-to-Frequency Converter," 2009, *Plano, TX, USA*.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley, 2012. [Online]. Available: https://books.google.com.vn/books?id=Br33IRC3PkQC

[10] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: https://doi.org/10.1109/TIT.1967.1053964.