A low-cost wireless extension for object detection and data logging for educational robotics using the ESP-NOW protocol (#88894)

First submission

Guidance from your Editor

Please submit by **29 Sep 2023** for the benefit of the authors (and your token reward) .

Structure and Criteria
Please read the 'Structure and Criteria' page for general guidance.

Raw data check
Review the raw data.

Image check
Check that figures and images have not been inappropriately manipulated.

If this article is published your review will be made public. You can choose whether to sign your review. If uploading a PDF please remove any identifiable information (if you want to remain anonymous).

Files
Download and review all files from the materials page.

20 Figure file(s)
2 Latex file(s)
9 Raw data file(s)

Structure and Criteria

Structure your review

The review form is divided into 5 sections. Please consider these when composing your review:

1. **BASIC REPORTING**
2. **EXPERIMENTAL DESIGN**
3. **VALIDITY OF THE FINDINGS**
4. General comments
5. Confidential notes to the editor

You can also annotate this PDF and upload it as part of your review

When ready submit online.

Editorial Criteria

Use these criteria points to structure your review. The full detailed editorial criteria is on your guidance page.

BASIC REPORTING

- Clear, unambiguous, professional English language used throughout.
- Intro & background to show context. Literature well referenced & relevant.
- Structure conforms to PeerJ standards, discipline norm, or improved for clarity.
- Figures are relevant, high quality, well labelled & described.
- Raw data supplied (see PeerJ policy).

EXPERIMENTAL DESIGN

- Original primary research within Scope of the journal.
- Research question well defined, relevant & meaningful. It is stated how the research fills an identified knowledge gap.
- Rigorous investigation performed to a high technical & ethical standard.
- Methods described with sufficient detail & information to replicate.

VALIDITY OF THE FINDINGS

- Impact and novelty not assessed. *Meaningful* replication encouraged where rationale & benefit to literature is clearly stated.
- All underlying data have been provided; they are robust, statistically sound, & controlled.
- Conclusions are well stated, linked to original research question & limited to supporting results.

Standout reviewing tips

The best reviewers use these techniques

Tip	*Example*
Support criticisms with evidence from the text or from other sources	*Smith et al (J of Methodology, 2005, V3, pp 123) have shown that the analysis you use in Lines 241-250 is not the most appropriate for this situation. Please explain why you used this method.*
Give specific suggestions on how to improve the manuscript	*Your introduction needs more detail. I suggest that you improve the description at lines 57- 86 to provide more justification for your study (specifically, you should expand upon the knowledge gap being filled).*
Comment on language and grammar issues	*The English language should be improved to ensure that an international audience can clearly understand your text. Some examples where the language could be improved include lines 23, 77, 121, 128 – the current phrasing makes comprehension difficult. I suggest you have a colleague who is proficient in English and familiar with the subject matter review your manuscript, or contact a professional editing service.*
Organize by importance of the issues, and number your points	*1. Your most important issue* *2. The next most important item* *3. …* *4. The least important points*
Please provide constructive criticism, and avoid personal opinions	*I thank you for providing the raw data, however your supplemental files need more descriptive metadata identifiers to be useful to future readers. Although your results are compelling, the data analysis should be improved in the following ways: AA, BB, CC*
Comment on strengths (as well as weaknesses) of the manuscript	*I commend the authors for their extensive data set, compiled over many years of detailed fieldwork. In addition, the manuscript is clearly written in professional, unambiguous language. If there is a weakness, it is in the statistical analysis (as I have noted above) which should be improved upon before Acceptance.*

A low-cost wireless extension for object detection and data logging for educational robotics using the ESP-NOW protocol

Emma Capaldi Corresp. 1

1 Phillips Academy Andover, Andover, Massachusetts, United States of America

Corresponding Author: Emma Capaldi
Email address: ecapaldi25@andover.edu

In recent years, inexpensive and easy to use robotics platforms have been incorporated into middle school, high school, and college educational curricula and competitions all over the world. Students have access to microprocessors and advanced sensor systems that engage, educate, and encourage their creativity. In this study, the capabilities of the widely available VEX Robotics System are extended using wireless ESP-NOW protocol to allow for real time data logging and to extend the computational capabilities of the system. Specifically, this study presents an open source system that interfaces a VEX V5 microprocessor, an OpenMV camera, and a computer. Images from OpenMV are sent to a computer where object detection algorithms can be run and instructions sent to the VEX V5 microprocessor while system data and sensor readings are sent from the VEX V5 microprocessor to the computer. System performance as a function of distance between transmitter and receiver, data packet round trip timing, and object detection using YoloV8 were evaluated. Three sample applications are detailed including the evaluation of a vision based object sorting machine, a drivetrain trajectory analysis, and a PID control algorithm tuning experiment. It was concluded that the system is well suited for real time object detection tasks and could play an important role in improving robotics education.

A Low-Cost Wireless Extension for Object Detection and Data Logging for Educational Robotics Using the ESP-NOW Protocol

Emma Capaldi[1]

[1]**Phillips Academy, Andover, MA 01810, USA**

Corresponding author:
Emma Capaldi[1]

Email address: ecapaldi25@andover.edu

ABSTRACT

In recent years, inexpensive and easy to use robotics platforms have been incorporated into middle school, high school, and college educational curricula and competitions all over the world. Students have access to microprocessors and advanced sensor systems that engage, educate, and encourage their creativity. In this study, the capabilities of the widely available VEX Robotics System are extended using wireless ESP-NOW protocol to allow for real time data logging and to extend the computational capabilities of the system. Specifically, this study presents an open source system that interfaces a VEX V5 microprocessor, an OpenMV camera, and a computer. Images from OpenMV are sent to a computer where object detection algorithms can be run and instructions sent to the VEX V5 microprocessor while system data and sensor readings are sent from the VEX V5 microprocessor to the computer. System performance as a function of distance between transmitter and receiver, data packet round trip timing, and object detection using YoloV8 were evaluated. Three sample applications are detailed including the evaluation of a vision based object sorting machine, a drivetrain trajectory analysis, and a PID control algorithm tuning experiment. It was concluded that the system is well suited for real time object detection tasks and could play an important role in improving robotics education.

INTRODUCTION

Machine learning (ML) and artificial intelligence (AI) are changing the way goods and people travel with autonomous vehicles (Bathla et al., 2022), the way people write (Kasneci et al., 2023), the way people debug programs (Surameery and Shakor, 2023), the way people create art (Mazzone and Elgammal, 2019), the way teaching is done (Adiguzel et al., 2023), and the way we work (J. et al., 2017).

Many middle schools and high schools around the United States have adopted engineering courses that have a robotics component (Harrell et al., 2004; Darmawansah et al., 2023). In addition, participation in Robotics in middle school and high school has been shown to encourage students to pursue (Hendricks et al., 2012; Sullivan and Bers, 2019) and prepare students for (Karim et al., 2015) careers in STEM. They have also been shown to be effective at improving academic performance (Jurado-Castro et al., 2023) and promoting strong computational-thinking (Evripidou et al., 2020). Robotics competitions introduce students to the fields of Mechanical Engineering, Electrical Engineering, Computer Science, and Mathematics in an exciting competitive sports-like environment.

Recently, there has been a push to incorporate AI and machine learning into middle school and high school curriculum (Xiong et al., 2018; Zhai et al., 2021; Dai et al., 2020; Knox, 2020; Ali et al., 2019). Project based learning in robotics has been successful at the university level (Zhang et al., 2020) and both Game- and project-based learning have been shown to be effective in teaching students about AI (Leitner et al., 2023; Martins and Wangenheim, 2022). Guiding students through projects involving image recognition is an excellent way to introduce them to the basics of machine learning and prepare them for a world that is quickly integrating AI into everyday life (Sophokleous et al., 2021). This paper provides a platform and a few examples that can be used to teach project based AI or ML at the middle school or high school level.

The platform presented integrates with the VEX V5 robotics system. The Robotics Education and Competition (REC) Foundation runs robotics competitions for elementary school, middle school, high school and college students using VEX robotics equipment. The REC Foundation reports that over 100,000 students participated at 2,600 international events in the 2022 - 23 competition season. They also claim that 1.1 million students participated in their various programs which distribute educational materials to classrooms (REC Foundation, 2023). The VEX robotics system used in competitions has been purchased and used by many high schools throughout the United States and abroad, and can be used beyond the competitions to teach STEM concepts through hands-on learning activities. However, the VEX robotics hardware includes a only rudimentary vision sensor capable of detecting competition objects by color at 50 fps, and the Cortex A9 microprocessor running the competition equipment has limited processing capability and a fixed ecosystem of plug and play peripherals. It can not run complex machine learning models without additional computational power.

In order to run AI or ML applications, the VEX equipment must be paired with a more capable processor and camera. There has been some success pairing the previous generation VEX Cortex microprocessor with a Raspberry Pi (He and Hsieh, 2018), and pairing the VEX V5 microprocessor with a Raspberry Pi (Zietek et al., 2022). The Raspberry Pi is a capable processor which has been used for many IOT applications (Nguyen et al., 2022) and even used to run convolutional neural networks (Sabri and Li, 2021). However, the Raspberry Pi requires an additional power source and lacks the computational power to execute complex vision models. In this paper, the VEX V5 microprocessor is paired with an ESP32 to communicate wirelessly with a computer system. ESP32s are cheap, widely available microcontrollers that communicate wirelessly with each other over long distances using the ESP-NOW protocol. The low overhead and fast through rate for the protocol makes it fast enough to use in real time robotic control applications. ESP32s have already been used for a wide range of projects, such as the remote monitoring of bee colonies (Kviesis et al., 2023), environmental monitoring (Winkler, 2021), air quality monitoring (Truong et al., 2021) and low-energy wireless networks (Labib et al., 2021). Others have also incorperated OpenMV (Abdelkader et al., 2017) machine vision modules, which integrate a camera with a basic low power microprocessor for vision applications (Guo et al., 2020; Wei-Peng et al., 2020).

Educational robotics also benefits from real time data visualization. Logging and visualization can improve students' investigative, analytical, and interpretive skills (McFarlane and Sakellariou, 2002), improve student understanding of key physics concepts (Alimisis and Boulougaris, 2014), and change the way students think about experiments (Barton, 2004). Access in the python programming language to libraries for data analysis, visualization, computer vision, and machine learning makes it well suited for this task (Fraanje et al., 2016).

This paper outlines the construction, coding, and usage of a low-cost communication interface between the widely available VEX V5 microprocessor, a computer system, and a camera system for the purpose of implementing machine learning algorithms and data logging in educational environments. The bidirectional system allows data transfer from the robotic system to the visualization system and for commands to be sent from the computer system to the robotic system. Due to the increased processing capability, this system extends the functionality and usage cases for the VEX V5 system, allowing students to run more advanced code on their robot. The wireless communication between the ESP32s also gives students real time access to data streams from their robot. This paper discusses possible ways that this data could be used to introduce students to important and interesting topics in robotics, such as PID controller optimization and trajectory analysis. The study also integrates a widely available OpenMV camera which provides both a photodetector and an onboard microprocessor. This gives the system the ability to run real time image recognition programs and provides an excellent platform for teaching students about object detection algorithms.

MATERIALS AND METHODS

Hardware Configuration

ESP-NOW The ESP32 (Espressif Systems, 2022) supports a proprietary communication protocol called ESP-NOW which utilizes the 2.4 GHz band and is optimized for sending short messages of up to 250 bytes to other ESP32 nodes with very little overhead, a quick response time on the order of a millisecond, low power usage, and a range between nodes of over 200 meters (Yukhimets et al., 2020). These characteristics make the ESP32 ideal for reading data from and sending instructions to real time control systems and data logging (Linggarjati, 2022). The onboard ESP32 does not require a separate battery and can be

powered by the output voltage on any of the VEX robot 3-wire ports which support an output of 5 Volts at a maximum of 2 Amps. The ESP-NOW link was operated at a bit rate of 1 Mbps.

The ESP32 also has Wi-Fi communication built into the board. The Wi-Fi communication would work well for the transmission of data from the board to a computer for applications such as data logging. However, the overhead associated with this communication would make it ill-suited for usage in a real time control system where the external computer is driving the robot and responding to sensor data transmitted by the robot.

OpenMV The H7 OpenMV camera (Abdelkader et al., 2017) module used in this study has an ARM 32-bit Cortex M7 CPU and consumes 140 mA when active. It can be powered using the 5 Volt 3-wire port power on the VEX V5 microprocessor. The USB connection between the camera and the computer can transmit data at rates up to 12 Mb/s. The camera can take 16-bit RGB565 images at a resolution of 640x480 at 75 frames per second. The onboard processor is capable enough to run small tensorflow lite models, run the opencv image processing library, and perform basic image preprocessing if needed. Programs can be written for and run on the OpenMV hardware using python.

Connectivity Three different hardware configurations were envisioned for this system, as shown in Figure 1. Each of these configurations has strengths that would be make it useful for different applications.

The first configuration, Figure 1a, has the VEX V5 microprocessor connected to an ESP32 with the OpenMV camera, computer, and another ESP32 connected to each other. The direct OpenMV to computer connection via USB supports rapid transmission of images. This is useful for real time computationally intensive image analysis. Commands are then sent to the robot via the ESP-NOW connection. This configuration is suitable for a case where the robot actuation is occurring in response to a visual que in a fixed region, such as machine automation tasks and quality control. This first configuration was selected for the sorting experiments detailed later in this paper.

The second configuration, Figure 1b, has the OpenMV and VEX V5 microprocessor connected to the same ESP32. The VEX V5 is connected via a RS485 bridge while the OpenMV camera is connected via a serial connection. The vision analysis would be limited to tensor flow lite models which can be run directly on the OpenMV camera. This configuration has the advantage that the OpenMV camera, the VEX V5 microprocessor, and the ESP32 can be powered using the VEX battery.

The third configuration, Figure 1c, has the VEX V5 microprocessor, the computer, and the OpenMV camera each connected to one of three independent ESP32s. If the camera and robot were both remotely located and the camera was not traveling with the robot, this configuration might be considered. This would be suitable for a case where a tensor flow lite model can be used and run directly on the OpenMV camera, negating the need for heavy computation on the computer.

All three of these configurations are viable and benefit from the added computational power gained by having a computer in the loop. In addition, each ESP32 can be used to interface with additional sensors such as lidar, temperature, or GPS sensors via I2C. These types of sensors are unavailable through the official VEX manufacturer and I2C is inaccessible on the VEX V5 microprocessor. These additional sensor types would greatly increase the scope and type of experiments that could be conducted using the VEX V5 microprocessor.

Wiring The VEX V5 microprocessor has 21 smart ports, of which 20 are located on the front of the device as shown in Figure 2. These smart ports can communicate with external devices using the RS485 standard and provide power at 12 Volts. The VEX V5 microprocessor also has eight 3-wire ports located on the side of the device which can be used to supply power at 5 Volts. The ESP32 power input pin is connected to a 5 Volt power supply via the 3-wire port on the VEX V5 microprocessor. The RS485 signal emitted via the smart port on the VEX V5 microprocessor is decoded by a MAX485 chipset. The power to the RS485 decoder is supplied via the 3.3 Volt power output pin on the ESP32. The RS485 decoder used in this study can be set to transmit or receive a signal based on the state of the DE and RE pins. Bidirectional communication is achieved by driving these pins using the ESP32. When a signal is received via the ESP-NOW protocol, the ESP32 sets the state of the RS485 to transmit mode. A message is then sent via RS485 to the VEX V5 microprocessor. The ESP then sets the RSS485 decoder back to receive mode so that the VEX V5 microprocessor may transmit messages through the ESP32 back to the computer system. The wiring diagram showing the connection between the ESP32, RS485 decoder and VEX V5 microprocessor is shown in Figure 2.

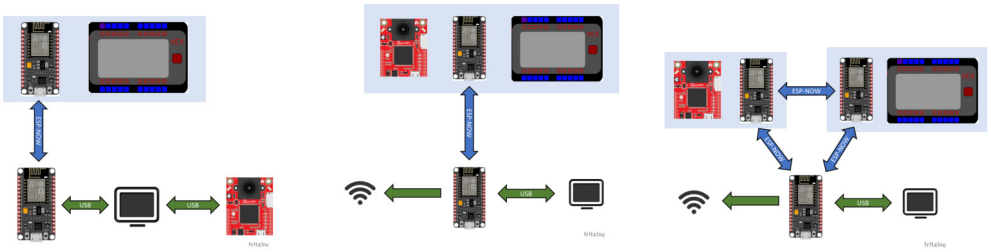


Figure 1. Potential configurations for the remote ESP-NOW enabled connection between the computer, OpenMV camera, and the VEX V5 microprocessor. (a) OpenMV camera is directly connected to the computer via a USB cable while the computer and VEX V5 microprocessor communicates via ESP-NOW. (b) The OpenMV camera and VEX V5 microprocessor both communicate to the computer through a single ESP-NOW link. (c) The OpenMV camera and the VEX V5 microprocessor are each linked to a separate ESP32 allowing communication via ESP-NOW between camera, VEX V5 microprocessor, and the computer. In all cases, one ESP32 was connected to the VEX processor and one ESP32 was hardwired to the computer.

Software

Three distinct pieces of software were written and are available in the supplemental data section of this paper. This includes a driver for communication between the two ESP32s, a code for running the VEX V5 microprocessor, and a GUI for use on the computer system.

ESP32 Communication Code The Arduino IDE was used to code a communication link between two ESP32s using the ESP-NOW communication protocol. The ESP32 connected to the VEX V5 microprocessor continuously transmits system status data via the ESP-NOW protocol at a fixed interval to another ESP32. The ESP32 linked via USB to the computer receives the data, and repeats it to the computer via a serial link. Commands sent by the computer through the USB are echoed in the opposite direction back to the ESP32 connected to the VEX V5 microprocessor.

VEX Code A code template written in C++ is provided which runs the communication link on the VEX V5 Microprocessor. It provides a starting point for users interested in modifying and extending the capabilities of this system. The code auto detects the presence of VEX peripherals and transmits data from each of these peripherals at a regular interval to the computer. The code also provides a mechanism for the VEX V5 microprocessor to recognize custom commands sent from a computer and to run predefined tests in response to instruction from the computer. The code can be edited using Visual Studio Code with the VEX VS Code Extension installed and then downloaded onto the VEX V5 Microprocessor.

Python GUI A graphical user interface was written in python using the Tkinter graphical user interface (GUI) library. This library is included in the standard installation of Python. Data received from the VEX V5 microprocessor is plotted in real time using MatplotLib (Hunter, 2007). This data can be stored or retrieved from files on the hard drive, plotted, and analyzed using the python code. The python code can also capture and display images from an OpenMV camera connected via USB to the computer. It can be used to perform machine learning operations on the images and send commands back to the VEX V5 microprocessor. Users can modify the code to create hybrid control systems that are running some processing on the computer platform and some on the VEX V5 microprocessor while sharing data and commands between the two. The Python multiprocessor module is used to isolate the GUI operations from data transmission, resulting in a smooth GUI and much improved responsiveness when compared to a single processor algorithm.

Cost

The system hardware was designed with minimal cost and footprint. It is intended to be used by high schools that already have a VEX program in place. A VEX starter kit costs on the order of $1,000

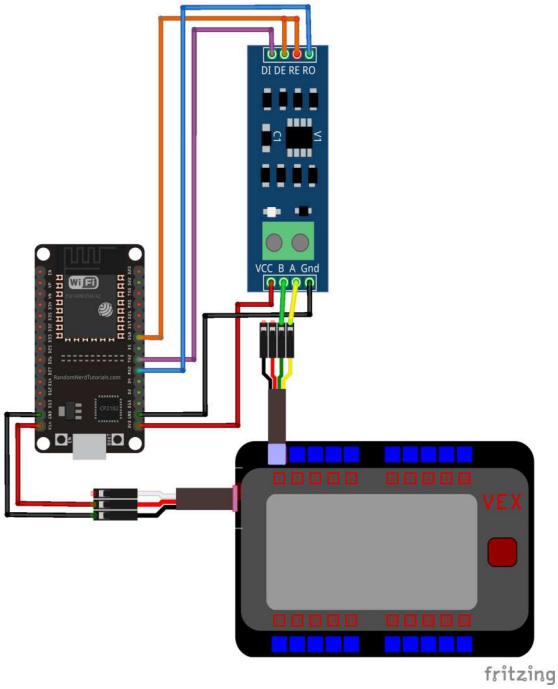


Figure 2. Wiring diagram between ESP32, RS485 decoder and VEX V5 microprocessor. As shown, the ESP32 is connected to a 3 wire port on the microcontroller which supplies power. The RS485 decoder receives and transmits signals to the microcontroller via the smart port.

USD. The parts needed to extend the system to use machine learning and sophisticated machine vision cost less than $100 USD, as shown in Table 1. The ESP32/RS485 package connected to the VEX V5 microprocessor fits within an electronics case with dimensions 2.4 x 1.4 x 1 inches. The ESP32 connected to the computer fits within an electronics case with dimensions 2.36 x 1.42 x 0.67 inches.

Item	Quantity	Cost (in US$)
ESP-32S Wifi Development Board	2	$ 5.67
TTL to RS-485 MAX485 Module	1	$ 1.48
2.4 x 1.4 x 1 inch Electronics Case	1	$ 1.44
2.36 x 1.42 x 0.67 inch Electronics Case	1	$ 1.40
OpenMV Camera Module	1	$ 82.00
Total		$ 97.66

Table 1. Device cost breakdown as determined on Amazon.com on 6/14/2023.

RESULTS AND DISCUSSION

Message Transit Time The time required for a message to be sent from the computer to the VEX V5 microprocessor, processed by the VEX V5 microprocessor, and then for a response to be sent back to the computer was determined. The mean round trip message time was determined from 10 trials to be 187 ms with a standard deviation of 57 ms.

Packet Loss The practical distance limits between the computer and the VEX V5 microprocessor described in this experiment were evaluated by determining the wireless packet loss as a function of distance. A VEX V5 microprocessor was attached to a drivetrain as shown in Figure 9a. The robot was then placed on the ground and a laptop with a receiving ESP32 connected to it was moved at intervals of 10 m away from the robot. Data was transmitted from the VEX V5 microprocessor at intervals of 50 ms and received on the computer end. The number of data packets received by the computer in a 15 second interval was recorded. The packet loss, *PL*, is calculated as follows:

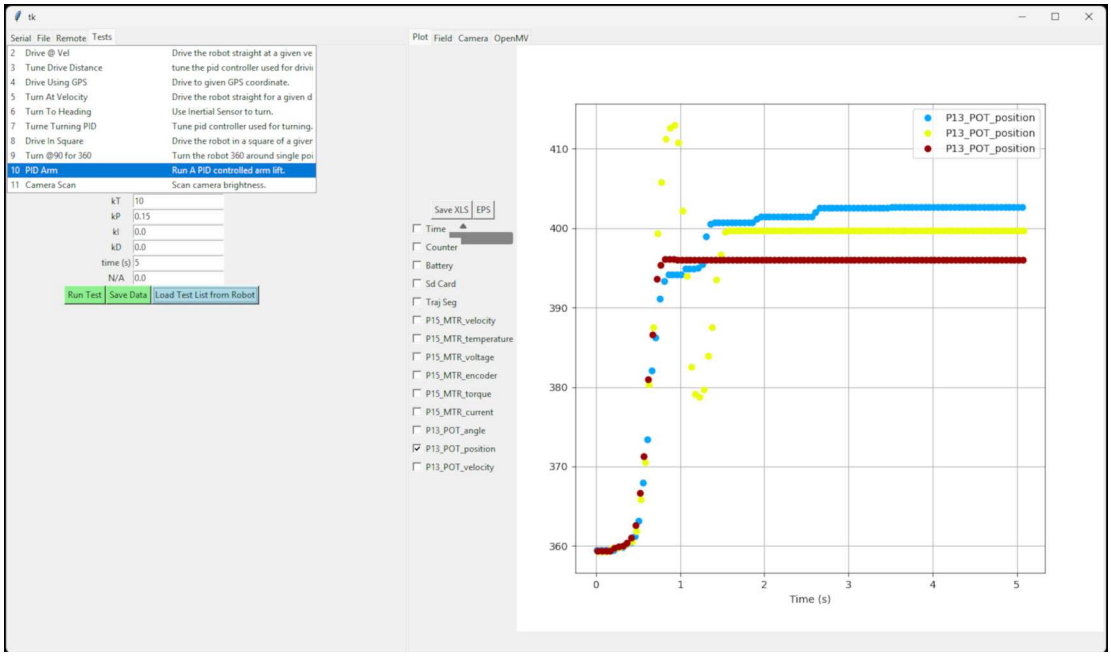


Figure 3. Screenshot showing the Python GUI running on the computer system which can be used to send commands wirelessly via the ESP-NOW connection to the robot and to display data in real time from sensors and motors on the VEX V5 microprocessor. Here the results of multiple runs are displayed simultaneously and can be compared with one another.

$$PL = \frac{n_e - n_a}{n_e} \times 100\% \tag{1}$$

where n_e is the number of expected packets received from the ESP-32, and n_a is the number of packets actually received. The mean and standard deviation of the percent loss for each distance interval is shown in Table 2. The packet loss was less than 1% between 0 and 40 meters. For data logging, the functionality is sufficient up to 60 meters, but for control applications where the computer is in the control loop, the distance between transmitter and receiver should be 40 meters or less.

Distance (m)	Mean [%]	Standard Deviation [%]
0	0.00	0.00
10	0.197	0.139
20	0.571	0.410
30	0.861	1.22
40	0.126	0.178
60	13.2	10.4

Table 2. Average and standard deviation of the percent of data messages lost between transmitter and receiver outdoors as a function of distance.

Machine Learning Sorting Application

As an example of an application that uses machine vision to identify objects and drive the motion of a physical system, a simple sorting experiment was created. As shown in Figure 5, the sorting machine has an OpenMV camera placed above a platform on which VEX game pieces are placed. The machine can then sort the pieces to the left side or right side of the platform.

VEX game pieces were collected from the past five game seasons, as shown in Figure 4. These game pieces differ significantly from one another and make for a good proof of concept experiment. An

OpenMV H7 camera was held 19 inches above a platform onto which the VEX competition objects were placed. The camera was used to obtain 640x480 pixel RGB images of objects on the platform. The images from the camera were analyzed by a computer, the YoloV8 algorithm was used to determine the object type, and then a signal was sent to the VEX V5 microprocessor. In response to signals from the microprocessor, a motor swings an arm across the platform, sweeping objects either to the left or the right. The system can be programmed to sort one VEX game object type to the left and another VEX game object type to the right. To test the accuracy of the model, we programmed the system to swing the arm to the left if a box was detected, to the right if an acorn was detected, and not to move the arm if no object was detected. Objects were placed on the platform or the platform was left empty for 2 second intervals to allow the arm an appropriate time to react. Each object was placed on the platform 10 times, for a total of 30 trials. A random number generator was used to determine the order in which objects would be placed on or withheld from the platform.

Figure 4. VEX game pieces used for sorting experiment. The object types listed from left to right include in the top row the acorn, box, container and in the bottom row the ring, disc, and ball.

Training The YOLO (You Only Look Once) v8 image object identification algorithm was selected for use in this project because of its ability to identify objects with high accuracy at high speed. The model was trained on pictures of VEX game objects from multiple years of competitions, as shown in Figure 4. A set of 116 images were taken at a size of 640x480 pixels using the OpenMV camera of the game objects in various positions, orientations, groupings, and at various angles. Figure 6a shows one of the images that were taken. The training images were imported within LabelStudio (Tkachenko et al., 2022), an image labelling software that makes it easy to label objects for image recognition algorithms. Each distinct VEX game object was assigned a tag. For each image, all identifiable objects were assigned a tag and labelled with a bounding box. After the images were labeled, they were split into training, a testing, and a validation sets which consisted of 86, 16, and 14 images respectively. Training was run for 50 epochs and the best set of YoloV8 parameters from these 50 epochs was used as the final model.

After training was completed, the accuracy of the best resulting YoloV8 model was then measured

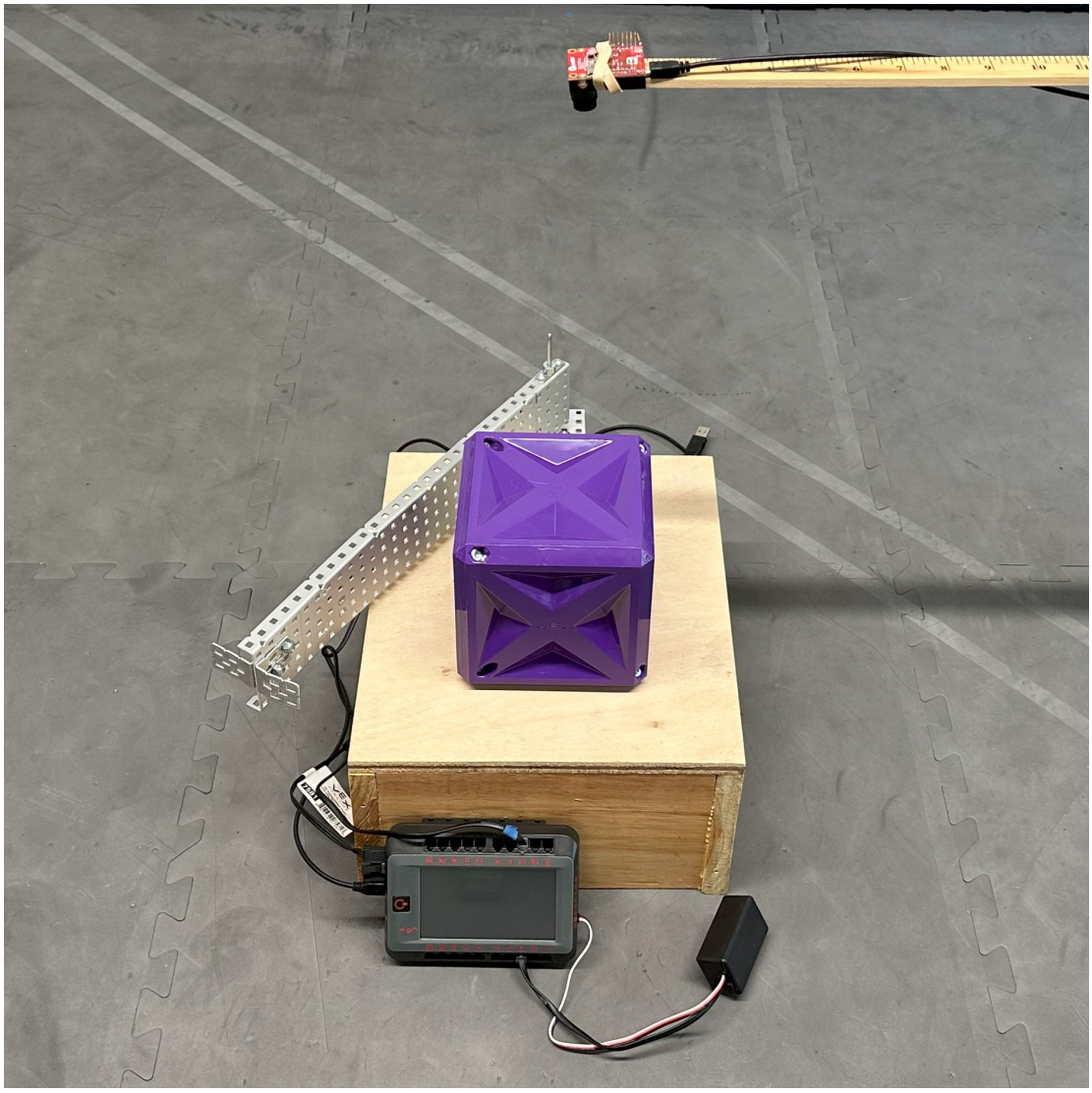


Figure 5. Simple sorting machine used to test visual recognition of VEX game pieces. Objects are pushed to the right or the left depending on their identified type.

using the validation set of images. Figure 6b illustrates how the model can label identified objects within an image. The name of each object type and the confidence for each prediction is shown at the top of the bounding box for each object. The sample image illustrates that most objects can be identified with high confidence (above 90%). It also shows that some objects can be missed, such as the yellow disc shown behind a purple ring, and some objects can be identified twice, such as the double bounding boxes around the purple ring. To quantify these errors and labeling performance, the precision, recall, mAP, and processing time of the model were evaluated.

Precision and Recall Precision and recall are useful values for evaluating the performance of a machine learning model. Precision, P, is a measure of how many identifications made were correct, while recall, R is a measure of how many objects were identified correctly, and are defined as follows:

$$P = \frac{TP}{TP+FP} \tag{2}$$

$$R = \frac{TP}{TP+FN} \tag{3}$$

Figure 6. (a) Raw image taken using OpenMV H7 camera at 640x480 resolution of partially occluded VEX game pieces. (b) Predicted object labels using YOLOV8 algorithm and trained weights.

where TP refers to the number of true positives, or number of labeled objects that were correctly identified, FP to the number of false positives, or number of predicted objects that did not exist, and FN to the number of false negatives, or number of labeled objects that were not identified. TP, FP, and FN are calculated from the confusion matrix, a table that shows the number of each object that were labeled as each of the possible object labels, as shown in Figure 7a. Precision and recall scores lie between 0 and 1, and a more accurate model has both a higher precision score and recall score. The precision-recall curve for each of the tested object categories is shown in Figure 7b. All objects had precision and recall scores close to 1, demonstrating that the model correctly identified objects most of the time. The precision and recall values can also be combined to generate an F1 score, another common way to evaluate the performance of a model. The F1 score is defined as follows:

$$F1 = \frac{2 \times P \times R}{P + R} \tag{4}$$

F1 scores lie between 0 and 1, and a more accurate model has a higher F1 score. The F1 scores of each object category were used to generate an F1 curve, as shown in Figure 7c. All classes had F1 scores close to 1 at most confidence levels, and all classes had an F1 above 0.99 at a confidence level of 0.882.

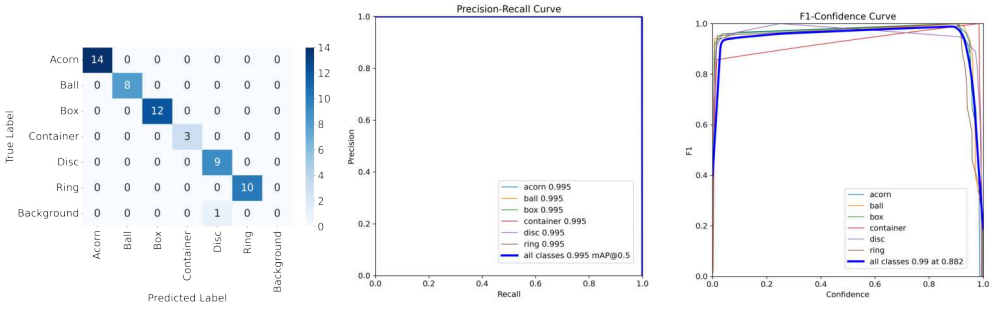
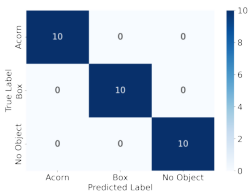


Figure 7. Evaluation of the validation images using the best YoloV8 weights gives (a) the confusion matrix for the objects detected in the validation images, (b) the precision-recall curve, and (c) the F1 curve.

A confusion matrix and precision, recall, and accuracy scores were also calculated for the sorting experiment described earlier. The results are shown in Figure 8. The system identified both objects and an empty platform correctly with 100% accuracy.

mAP Score Another value of interest for evaluating the model's performance is the mean average precision value, mAP, which is equal to the area enclosed by the precision-recall curve and the coordinate axis. It is calculated as follows:



(a)

Metric	Value [%]
Recall (ball)	100
Precision (ball)	100
Recall (acorn)	100
Precision (acorn)	100
Accuracy	100

(b)

Figure 8. (a) The confusion matrix, and (b) metric results obtained from the object sorting experiment.

$$mAP = \frac{\sum_{k=1}^{k=n} AP_k}{n} \tag{5}$$

where n is the number of classes the model is trained on, and AP_k is the average precision of class k. mAP scores lie between 0 and 1, and a more accurate model has a mAP score closer to 1. The difference between a mAP50 and mAP50-95 score lies in the way that true and false positives are identified. A label is classified as true positive when a certain percentage of the labeled area overlaps with the true object boundary. If the overlapping area does not meet this threshold, it is labelled as a false positive. This overlap area is known as the intersection over union, or IoU, value. To compute the mAP50 score, the IoU value of a predicted object must be greater than or equal to 0.5 to be classified as a true positive. To compute the mAP50-95 socre, the number of true positives is computed as the average of the number of true positives for each IoU threshold value between 0.5 and 0.95. The mAP50 and mAP50-95 values are shown in Table 3, and are all close to 1. The mAP50 scores are all larger than the mAP50-95 scores, which is expected as there should be fewer true positives as the IoU threshold value increases.

Class	Instances	mAP50	mAP50-95
All	57	0.955	0.924
Acorn	14	0.995	0.940
Ball	8	0.995	0.941
Box	12	0.995	0.940
Container	3	0.995	0.995
Disc	10	0.995	0.865
Ring	10	0.995	0.863

Table 3. mAP values computed from validation training set of 14 randomly selected images using the weights from the best set of weights obtained after 50 training epochs. The column labeled instances gives the number of labeled objects of each type found within the set of 14 images.

Inference Time The YoloV8 algorithm takes an image as input, then conducts a preprocessing step that resizes the image, pads the image to have a square shape, normalizes the pixel values and converts the pixel array to a pytorch tensor. The inference step identifies objects within the processed image, and then post-processing conducts a non-maximal suppression (NMS) step. The processing time for each of these steps was recorded. Table 4 shows the time taken for preprocessing, inference, and post-processing for a single image on a set of devices with the algorithm running on either the CPU or GPU averaged over the set of validation images.

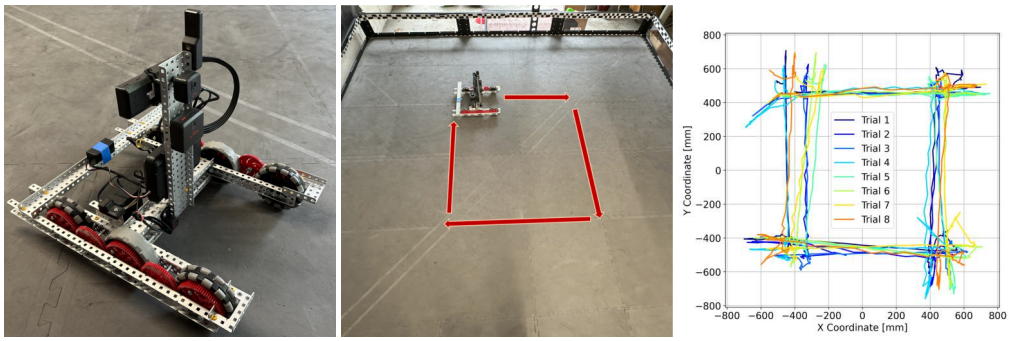
Real Time Data Logging

Trajectory Analysis Repeatability and reliability are a recurrent problem in middle school and high school robotics competitions and in educational curriculum. Understanding how design choices and control algorithms influence the reproducibility of a particular motion is particularly useful. In this section, a simple six wheel drivetrain powered by two motors, as shown in Figure 9a, is created and outfitted with a VEX GPS sensor. The VEX GPS sensor allows the drivetrain's position to be determined within a competition field. Though the name implies this sensor relies on GPS, it is in fact vision based and uses images of an encoded strip placed along the field perimeter to determine the location.

Device	Type	Preprocessing mean (stdev) [ms]	Inference mean (stdev) [ms]	Postprocessing mean (stdev) [ms]
AMD Ryzen 9 5950X	CPU	1.09 (0.325)	62.8 (2.49)	0.53 (0.450)
NVIDIA Quadro P2200	GPU	0.750 (0.493)	8.41 (1.67)	1.64 (0.585)
Apple M2	CPU	0.772 (0.351)	52.1 (14.4)	0.472 (0.957)

Table 4. Average inference times obtained by executing the YoloV8 algorithm on various hardware platforms.

The drivetrain was programmed to travel around the field in a prescribed square pattern, as shown in Figure 9b. The tests are triggered by the GUI executing on the computer and are parameterized, allowing students to adjust the travel distances. Sensor data is sent back at a rate of 20 readings per second to the computer. This allows students to review real time data giving the robot's position and orientation, which can be graphed and analyzed in a variety of ways. A graph displaying the real time position and orientation of the drivetrain is provided. Data from multiple trials can be saved and graphed simultaneously giving a clear graphical indication of the drivetrain repeatability, as shown in Figure 9c, where the position of the drivetrain completing the square path shown in Figure 9b during eight trial runs are graphed simultaneously.



(a) (b) (c)

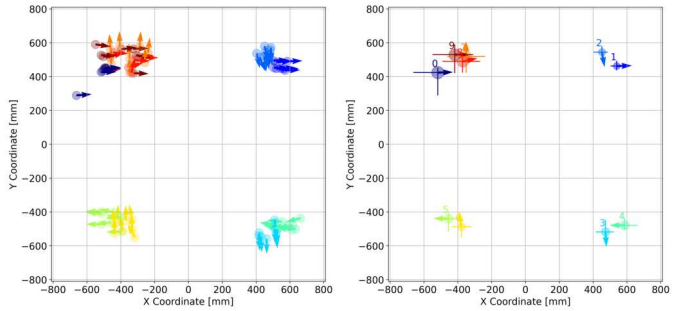
Figure 9. (a) Drivetrain used in testing. The ESP32 transmitter is located on the back of the drivetrain held in place with blue tape. (b) Photograph of the VEX field taken from above with the robot path labelled with red arrows. (c) Repeat trajectories collected in the system.

Segement ID	Description
0	Starting position
1	Drive forward 40 inches (1016 mm)
2	Turn right by 90 degrees
3	Drive forward 40 inches (1016 mm)
4	Turn right by 90 degrees
5	Drive forward 40 inches (1016 mm)
6	Turn right by 90 degrees
7	Drive forward 40 inches (1016 mm)
8	Turn right by 90 degrees
9	Final heading adjustment.

Table 5. Drivetrain movement segment definitions.

Beyond the visualization of trajectories, a jupyter notebook is provided which can be used to evaluate the repeatability using saved data in more detail. The VEX code that runs the drivetrain is built so that each segment of a trajectory is labeled independently. For example, segment 1 involves moving the drivetrain along a straight path for a given distance while segment 2 involves turning the robot to the right by 90 degrees. The position and orientation of the drivetrain at the endpoint of each segment is plotted in

Figure 10a for all 8 trials. Figure 10b displays the position and orientation of the drivetrain at the end of each segment averaged over all 8 trials, as well as an error bar that indicates the maximum and minimum position. This data can be used to assess the reliability and repeatability of the drivetrain and compare the performance of design changes. The provided jupyter notebook can also be used to tabulate all of the trajectory averages as shown in Table 6.



(a) **(b)**

Figure 10. (a) Drivetrain position and orientation at the endpoint of each trajectory segment shown for all 8 trials. (b) Drivetrain position and orientation averaged over 8 trials. The min and max values of the drivetrain s and y position are shown with the error bars.

S_{ID}	x mean (stdev) [mm]	y mean (stdev) [mm]	θ mean (stdev) [degree]	x range $[min, max]$ [mm]	y range $[min, max]$ [mm]	θ range $[min, max]$ [degree]
0	-518 (55)	423 (51)	88 (0)	$[-662, -486]$	$[289, 452]$	$[86, 89]$
1	539 (29)	462 (21)	88 (1)	$[503, 584]$	$[437, 494]$	$[86, 91]$
2	452 (27)	544 (21)	170 (5)	$[403, 487]$	$[515, 580]$	$[158, 179]$
3	473 (43)	-519 (32)	176 (1)	$[415, 523]$	$[-558, -448]$	$[174, 179]$
4	587 (49)	-479 (21)	269 (10)	$[509, 662]$	$[-506, -437]$	$[256, 281]$
5	-455 (50)	-441 (38)	271 (4)	$[-516, -396]$	$[-517, -398]$	$[265, 281]$
6	-379 (45)	-488 (38)	353 (9)	$[-437, -317]$	$[-550, -444]$	$[-28, 4]$
7	-350 (80)	518 (52)	0 (14)	$[-461, -238]$	$[422, 578]$	$[-4, 6]$
8	-372 (78)	489 (38)	77 (25)	$[-490, -268]$	$[423, 542]$	$[13, 97]$
9	-418 (95)	529 (50)	91 (6)	$[-549, -309]$	$[420, 590]$	$[82, 100]$

Table 6. Table of segment endpoints. x, y, and θ are the robot's field coordinates at the end of each movement segment respectively.

Tuning a PID Controller A proportional-integral-derivative controller, or PID controller, is a control feedback loop that corrects for deviations from the desired value. PID controllers are commonly used to precisely control parameters such as temperature, speed, or pressure. In robotics, PID controllers can be used to control the velocities of a drivetrain, for lifting objects, and for spinning flywheels. For example, the angular velocity of a flywheel might need to be precisely controlled to propel a projectile a particular distance. The controller would adjust the flywheel motor power as a function of the difference between the target flywheel angular velocity and the actual angular velocity. Here the angular velocity could be determined via a rotation sensor on the flywheel axle.

Another common example would be lifting an object to a desired height. The experimental setup shown in Figure 11a illustrates the case where an arm has been fitted with a circular disc weight. The arm is driven directly via a motor at the base. A rotation sensor is attached to the axle passing through the arm and motor to monitor the arm configuration. The arm is to be lifted to a particular target angle. The motor power can be determined using a PID controller that responds to the difference, $e(t)$, between the target arm angle, θ_o and the actual arm angle as measured by the rotation sensor, $\theta(t)$ where the difference is defined by:

$$e(t) = \theta(t) - \theta_o \tag{6}$$

A PID controller determines the output response as a function of a term that is proportional to the difference between the target and actual value, a term that is proportional to the integral of this difference, and a term that is proportional to the derivative of this difference.

The equation for the instantaneous motor output, $u(t)$, is then defined as follows:

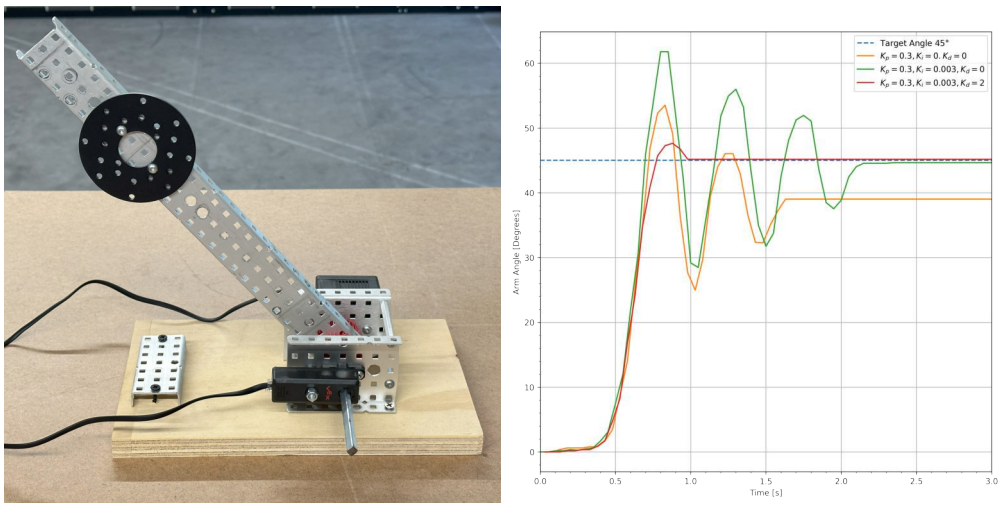
$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{7}$$

where K_p is the coefficient of the proportional term, $e(t)$ is the difference between the measured value and the desired value, K_i is the coefficient of the integral term, and K_d is the coefficient of the derivative term.

Each of the three coefficients K_p, K_i and K_d must be found for each application independently, as the optimal values depend on the characteristics of the system being driven. This process is known as tuning, and is often done by entering approximate coefficient values into the equation, observing the system response, and then further refining the coefficients.

Tuning is greatly simplified when real time data logging is possible. Students can observe the system response as they modify PID parameters to determine if a particular set of coefficients is effective or not. Figure 11b displays the system response with various PID parameters and a target angle of 45 degrees.

Real time data logging shows much more accurately whether a particular set of coefficients result in an effective controller. Figure 11b illustrates that a controller with only a proportional term never reaches the target angle for this system. While the proportional component on its own is an effective controller in many cases, it is not suitable for lifting applications where the output power may never be large enough to hold the weight at the target angle. The addition of an integral term allows the system to settle near the target value and the addition of a derivative term eliminates the oscillations by slowing down the response when the measured value changes too quickly. This allows the system to settle at the target value much faster. All of this information is captured visually as the student modifies parameters of the PID model.



(a) **(b)**

Figure 11. (a) Photograph of the weighted arm system. (b) System response given different PID parameters.

CONCLUSIONS

In this paper, we have demonstrated that the ESP-NOW protocol can be used to extend functionality of the VEX V5 microprocessor to allow real time object detection and real time data logging. Further,

the described system provides the ability to interface sensors and motors outside the VEX ecosystem, enhancing their capabilities for experimentation. The system can wirelessly log sensor data from the VEX V5 microprocessor in real time at a rate of 20 samples per second reliably to distances of 40 meters from the transmitter. The transmission distance can probably be increased with better placement of the transmitter relative to the ground and additional distance between the transmitter and the surrounding metallic components of the drivetrain.

Object detection in real time was achieved using the YoloV8 algorithm. Round trip messages from computer to the VEX V5 microprocessor require 187 ms and total prediction times for the YoloV8 algorithm range from 10.8 ms to 64.4 ms depending on the hardware used. Therefore, 4 to 5 sequential object identifications can be made per second using this system. The current speeds are sufficient to enable students to build real time object manipulation applications. Future work will focus on optimizing the transmission protocol to reduce the round trip message time further.

It was also shown that the YoloV8 algorithm, trained on a dataset of 116 images of VEX game pieces, was able to identify game pieces in a sample sorting application. This methodology can easily be extended to create robots that seek out particular objects, avoid objects, or manipulate objects depending on object identification. It is also straightforward to extend the methodology to larger groups of object types which are more difficult to distinguish from one another by increasing the number of training images used.

The real time wireless data logging enabled by this system also allows for the VEX V5 microprocessor to be used for more advanced experimentation in the classroom. Students can compute statistics related to repeatability as was done with the trajectory analysis. It was also demonstrated that the sometimes difficult task of tuning a PID controller can be tackled visually by students using graphs generated using the developed python GUI.

The low system cost, ease of construction, and the supplied software in this study make the ESP-NOW enabled system an excellent choice for schools that have existing VEX equipment and a desire to work with ML or conduct more advanced experimentation.

REFERENCES

Abdelkader, I., El-Sonbaty, Y., and El-Habrouk, M. (2017). Openmv: A python powered, extensible machine vision camera.

Adiguzel, T., Kaya, M. H., and Cansu, F. K. (2023). Revolutionizing education with ai: Exploring the transformative potential of chatgpt. *Contemporary Educational Technology*, 15:ep429.

Ali, S., Payne, B. H., Williams, R., Park, H. W., and Breazeal, C. (2019). Constructionism, ethics, and creativity: Developing primary and middle school artificial intelligence education.

Alimisis, D. and Boulougaris, G. (2014). Robotics in physics education: fostering graphing abilities in kinematics. pages 2–10.

Barton, R. (2004). *Using IT effectively in teaching and learning*, chapter Does data logging change the nature of children's thinking in experimental work in science?, pages 71–80. Routledge.

Bathla, G., Bhadane, K., Singh, R. K., Kumar, R., Aluvalu, R., Krishnamurthi, R., Kumar, A., Thakur, R. N., and Basheer, S. (2022). Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. *Mobile Information Systems*, 2022:1–36.

Dai, Y., Chai, C.-S., Lin, P.-Y., Jong, M. S.-Y., Guo, Y., and Qin, J. (2020). Promoting students' well-being by developing their readiness for the artificial intelligence age. *Sustainability*, 12:6597.

Darmawansah, D., Hwang, G.-J., Chen, M.-R. A., and Liang, J.-C. (2023). Trends and research foci of robotics-based stem education: a systematic review from diverse angles based on the technology-based learning model. *International Journal of STEM Education*, 10:12.

Espressif Systems (2022). Esp32 datasheet. Available at https://www.espressif.com/sites/default/files/documentation/ESP32_datasheet_en.pdf. (accessed on 14 June 2023).

Evripidou, S., Georgiou, K., Doitsidis, L., Amanatiadis, A. A., Zinonos, Z., and Chatzichristofis, S. A. (2020). Educational robotics: Platforms, competitions and expected learning outcomes. *IEEE Access*, 8:219534–219562.

Fraanje, R., Koreneef, T., Mair, A. L., and de Jong, S. (2016). Python in robotics and mechatronics education. pages 14–19. IEEE.

Guo, X., Wu, J., and Fang, J. (2020). Baby-follower: a child-care robot system based on openmv and iot. *Journal of Physics: Conference Series*, 1651:012121.

Harrell, D., Bataineh, M., El-Sheikh, E., and Spolski, J. (2004). The development of a pre-college engineering curriculum for high school students: design and implementation. pages 828–832. IEEE.

He, S. and Hsieh, S.-J. (2018). Multi-sensors for robot teaming using raspberry pi and vex robotics construction kit.

Hendricks, C., Alemdar, M., and Ogletree, T. (2012). The impact of participation in vex robotics competition on middle and high school students' interest in pursuing stem studies and stem-related careers. pages 25.1312.1–25.1312.16. ASEE Conferences.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9:90–95.

J., G., H., L., and Evans-Greenwood, P. (2017). Cognitive collaboration: Why humans and computers think better together. *Deloitte Review*, pages 8–29.

Jurado-Castro, J. M., Vargas-Molina, S., Gómez-Urquiza, J. L., and Benítez-Porres, J. (2023). Effectiveness of real-time classroom interactive competition on academic performance: a systematic review and meta-analysis. *PeerJ Computer Science*, 9:e1310.

Karim, M. E., Lemaignan, S., and Mondada, F. (2015). A review: Can robots reshape k-12 stem education? pages 1–8. IEEE.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., and Kasneci, G. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274.

Knox, J. (2020). Artificial intelligence and education in china. *Learning, Media and Technology*, 45:298–311.

Kviesis, A., Komasilovs, V., Ozols, N., and Zacepins, A. (2023). Bee colony remote monitoring based on iot using esp-now protocol. *PeerJ Computer Science*, 9:e1363.

Labib, M. I., ElGazzar, M., Ghalwash, A., and AbdulKader, S. N. (2021). An efficient networking solution for extending and controlling wireless sensor networks using low-energy technologies. *PeerJ Computer Science*, 7:e780.

Leitner, M., Greenwald, E., Wang, N., Montgomery, R., and Merchant, C. (2023). Designing game-based learning for high school artificial intelligence education. *International Journal of Artificial Intelligence in Education*.

Linggarjati, J. (2022). Design and prototyping of temperature monitoring system for hydraulic cylinder in heavy equipment using esp32 with data logging and wifi connectivity. *IOP Conference Series: Earth and Environmental Science*, 998:012042.

Martins, R. M. and Wangenheim, C. G. V. (2022). Findings on teaching machine learning in high school: A ten - year systematic literature review. *Informatics in Education*.

Mazzone, M. and Elgammal, A. (2019). Art, creativity, and the potential of artificial intelligence. *Arts*, 8:26.

McFarlane, A. and Sakellariou, S. (2002). The role of ict in science education. *Cambridge Journal of Education*, 32:219–232.

Nguyen, L. T. T., Ha, S. X., Le, T. H., Luong, H. H., Vo, K. H., Nguyen, K. H. T., Nguyen, A. T., Dao, T. A., and Nguyen, H. V. K. (2022). Bmdd: a novel approach for iot platform (broker-less and microservice architecture, decentralized identity, and dynamic transmission messages). *PeerJ Computer Science*, 8:e950.

REC Foundation (2023). Available at https://roboticseducation.org/about-us/. (accessed on 23 July 2023).

Sabri, Z. S. and Li, Z. (2021). Low-cost intelligent surveillance system based on fast cnn. *PeerJ Computer Science*, 7:e402.

Sophokleous, A., Christodoulou, P., Doitsidis, L., and Chatzichristofis, S. A. (2021). Computer vision meets educational robotics. *Electronics*, 10:730.

Sullivan, A. and Bers, M. U. (2019). Vex robotics competitions: Gender differences in student attitudes and experiences. *Journal of Information Technology Education: Research*, 18:097–112.

Surameery, N. M. S. and Shakor, M. Y. (2023). Use chat gpt to solve programming bugs. *International Journal of Information technology and Computer Engineering*, pages 17–22.

Tkachenko, M., Malyuk, M., Holmanyuk, A., and Liubimov, N. (2022). Label studio data labeling software.

458 Truong, T. V., Nayyar, A., and Masud, M. (2021). A novel air quality monitoring and improvement
459 system based on wireless sensor and actuator networks using lora communication. *PeerJ Computer*
460 *Science*, 7:e711.
461 Wei-Peng, Z., Li-Sha, C., Er-Min, L., and Feng-Chun, Z. (2020). Design and production of tracking
462 system based on openmv image recognition. pages 1198–1202. IEEE.
463 Winkler, R. (2021). Meteomex: open infrastructure for networked environmental monitoring and
464 agriculture 4.0. *PeerJ Computer Science*, 7:e343.
465 Xiong, Y., Wang, J., and Huang, J. (2018). *Textbook series on artificial intelligence for elementary and*
466 *middle schools*. East China Normal University Press.
467 Yukhimets, D., Sych, A., and Sakhnenko, A. (2020). Designing a method for constructing distributed
468 open acs based on the esp-now wireless protocol. pages 642–647. IEEE.
469 Zhai, X., Chu, X., Chai, C. S., Jong, M. S. Y., Istenic, A., Spector, M., Liu, J.-B., Yuan, J., and Li, Y.
470 (2021). A review of artificial intelligence (ai) in education from 2010 to 2020. *Complexity*, 2021:1–18.
471 Zhang, Z., Zhang, A., Zhang, M., and Esche, S. (2020). Project-based courses for b.tech. program of
472 robotics in mechanical engineering technology. *Computers in Education Journal*, 11:1–10.
473 Zietek, J., Wade, N., Roberts, C., Malek, A., Pylla, M., Xu, W., and Patil, S. (2022). Pac-man pete: An
474 extensible framework for building ai in vex robotics. *arxiv*.