ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Bench Organizer

Team #12

XIAOHU MU

(xiaohum2@illinois.edu)

LIANGCHENG SUN

(ls25@illinois.edu)

TA: Maanas Sandeep Agrawal

May 7, 2025

Abstract

Tools are frequently misplaced or missed due to human error in shared workspaces such as maker spaces, engineering labs, and mechanical workshops. Users often forget to return tools to their designated locations after use. This typically leads to inefficiencies, wasted time, and workflow disruptions. Existing solutions, such as manual sign-out sheets, RFID-based systems, and barcode scanning, require extra effort from users

This project focuses on building an automated system to help track tools and keep workspaces organized. It uses a Raspberry Pi 4 with YOLOv5 object detection and two webcams—one watching the bench and one watching the drawer. When a tool goes missing for more than 30 seconds, the system sends a signal to an Arduino Nano, which turns on an LED and buzzer to alert the user.

The detection accuracy reached about 93% for tools it was trained on, and each detection took less than 2 seconds. While features like learning new tools and working well in different lighting are not fully finished, the current system still shows a working and fast way to track tools in real time.

Contents

1 Introduction

In many labs and workshops, tools are often misplaced or forgotten after use, leading to disorganization, delays, and potential safety risks. Traditional tracking methods such as sign-out sheets or manual checklists are time-consuming and prone to human error. As workspaces grow more complex, there is an increasing need for automated systems that can reliably monitor tool usage and quickly alert users when items are missing.

This project introduces a smart bench organizer system that uses computer vision and embedded hardware to address this issue. The system automatically detects the presence or absence of tools using a Raspberry Pi 4 running a YOLOv5 object detection model and two webcams. One camera monitors tools on the workbench while the other observes tools in a drawer. When a tool is detected as missing for more than 30 seconds, the system sends a signal to an Arduino Nano, which activates a buzzer and an LED to alert the user.

This project combines real-time image recognition with a simple alert system to provide a low-cost, modular solution that can help users manage their tools more efficiently. The system is designed to be scalable, responsive, and easy to control remotely through a laptop interface.

2 Design

2.1 High Level Requirements

1. Accuracy and Responsiveness: The most important success indicator of our project is the accuracy of recognition and system responsiveness. Using a pre-trained machine learning model, we expect the system to recognize 90Bluetooth module, we anticipate that the recognition process will take less than two seconds.

2. Robustness: In real-world scenarios, the system may be affected by varyinglighting conditions. In this case, we must ensure that the systemis robust enough to function reliably without being impacted by these variations. Additionally, the system should be able to differentiate whether a user is taking or placing an item based on the movement of their hands

3. Extended Functionality: We also expect that when an uncategorized itemappears in the camera's view, users will receive a notification and be able toscan it into the system. The next time the item is detected, the systemshouldbe able to recognize it.

2.2 Design Procedure

The project was designed with modularity in mind, and we divided the system into three major blocks: tool detection, alert system, and control device. Several design approaches were considered for each block, and we selected solutions that best balanced accuracy, simplicity, and robustness.

For tool detection, we considered both ESP32-CAM and Raspberry Pi. We chose the Raspberry Pi 4 due to its greater processing power and compatibility with the YOLOv5 object detection framework. This enabled fast, on-device detection without needing cloud inference.

For the alert system, we originally planned to use Bluetooth with the ESP32 to send notifications, but connection stability was a major issue. We switched to a wired GPIO signal approach between the Pi and an Arduino Nano. The Arduino was programmed to trigger a buzzer and LED in response to a HIGH signal, which simplified debugging and improved reliability.

For the control device, we used a laptop that connects to the Pi via SSH over Wi-Fi. This approach allowed users to start detection scripts, monitor tool status, and view real-time camera feeds from anywhere on the same network.

This overall architecture was chosen because it reduces unnecessary complexity while still meeting our functional and performance goals.

2.3 Design Details

The tool detection system runs on a Raspberry Pi 4. Two USB webcams are fixed—one above the workbench and one above the drawer. These cameras capture images every 10 seconds. The Pi uses a custom-trained YOLOv5 model to detect the presence or absence of tools. Model training was performed using Python and PyTorch, with images labeled for object categories like screwdrivers and pliers.

Each detected object is tracked over time. If a previously detected tool disappears for more than 30 seconds, the Pi considers it "missing." At that point, the Pi sends a HIGH signal from a GPIO pin to an Arduino Nano.

The Arduino receives this signal and triggers a buzzer and LED alert, which provides a clear, real-time physical response. This mechanism helps users quickly identify when and where a tool is missing, even without actively monitoring the control interface.

The control interface is a laptop connected via SSH. A Python script handles camera input,

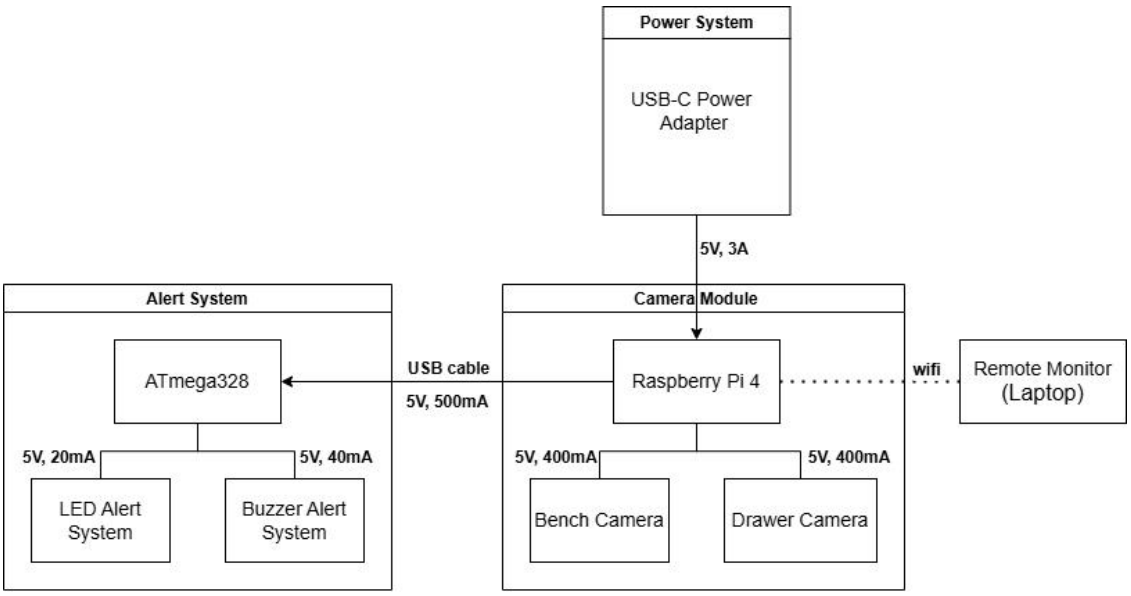


Figure 1: Block Diagram

model inference, and GPIO signaling. The live detection result is printed to the terminal and can be viewed in real time, along with the camera feed.

A simplified block diagram of the system is shown in Figure 1.

2.4 Subsystem Description

1. Camera Subsystem

   Functionality: The Camera Subsystem has one primary camera to monitor the user's activity on the tool rack to track the tool usage. Additionally, multiple cameras are integrated to Monitor the changes in items stored within drawers to capture additions or removals In real time.

   Contribution: This subsystem captures the activity of the tool rack and drawer contents and transmits the images/videos to the processing unit via Bluetooth. Continuous monitoring of tool interactions enables the system to detect missing or misplaced items. Furthermore, the camera subsystem also allows the system to recog-

nize new tools added by users

(a) Webcam

(b) Raspberry Pi

2. Processing Subsystem

   Functionality: The processing subsystem is powered by a microcontroller, Arduino Nano, which serves as the central computing unit. It runs the pre-trained classification model using OpenCVwhich allows real-time recognition and tracking of tools. The model could recognize the items on the desk, detect missing tools or misplaced tools, and determine the appropriate drawer for storing newly introduced objects.

   Contribution: This subsystem is responsible for analyzing visual data received from the Camera Subsystem, executing object detection algorithms, and making decisions based on tool presence and location. It also reduces tool misplacement and enhances workspace efficiency. Furthermore, it facilitates user feedback mechanisms, such as triggering LED indicators, updating the display, and logging tool usage history.

3. Power Subsystem

   The Raspberry Pi 4 is powered via a 5V, 3A USB-C power adapter or battery pack. • The Bluetooth Camera Module is powered by a 3.7V Li-ion battery with a 5VBoost

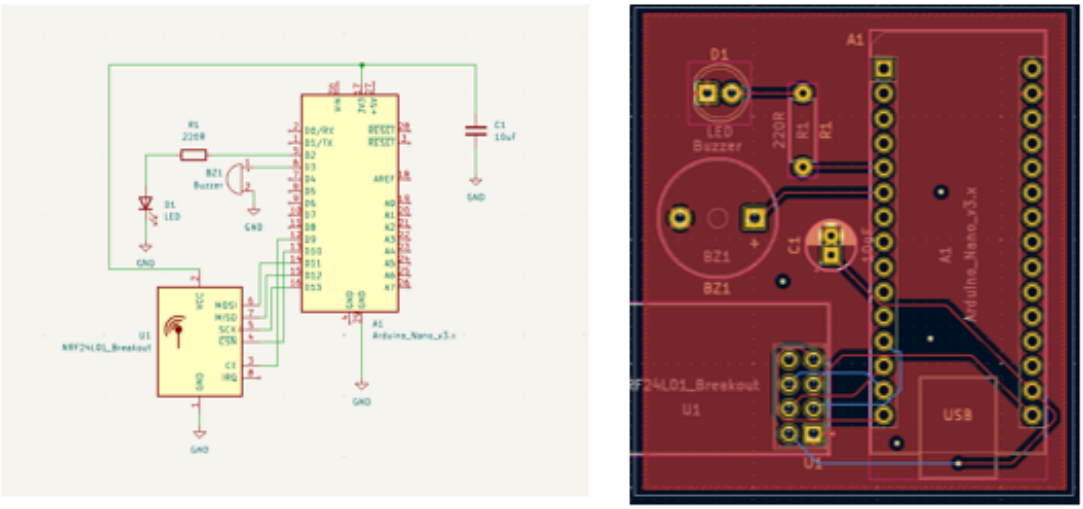


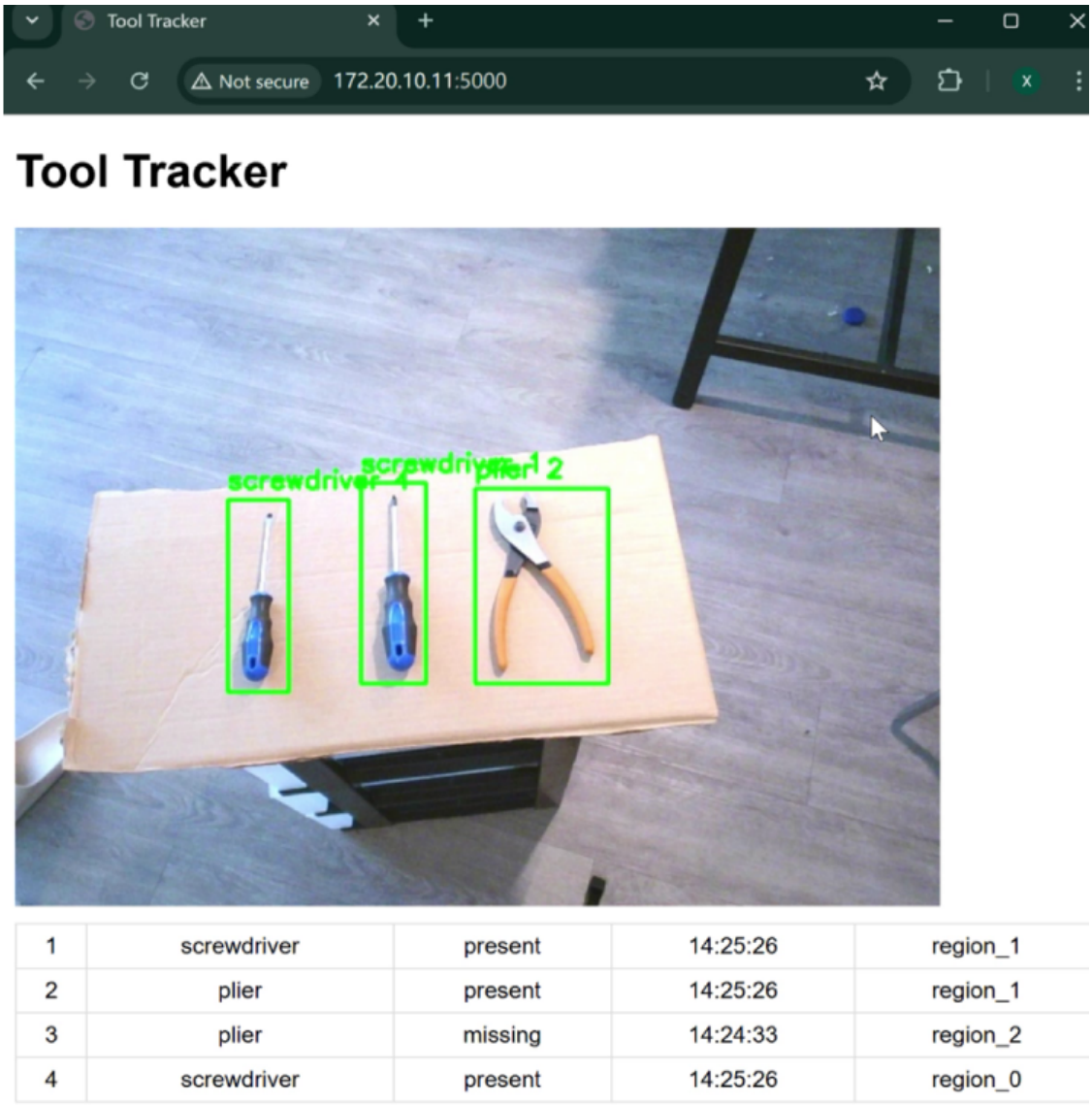
Figure 3: PCB Schematic

Converter. • The Display and LED Notification System are powered through the Raspberry Pi GPIO (5V/3.3V).

4. User Interface Subsystem

Functionality: This subsystem provides a visual platform for users to monitor and manage tool organization. It connects Raspberry Pi with a laptop via WIFI to present real-time item information, including tool status, missing or misplaced items, and storage guidance. Additionally, an LED indicator serves as a visual alert to notify users when a tool is missing or misplaced.

Contribution: This subsystem enhances user experience and accessibility by offering clear visual feedback and intuitive controls. By displaying real-time tool status and guiding proper organization, it helps reduce misplacement and ensures efficient workspace management. The LED indicator provides an immediate alert.



1	screwdriver	present	14:25:26	region_1
2	plier	present	14:25:26	region_1
3	plier	missing	14:24:33	region_2
4	screwdriver	present	14:25:26	region_0

Figure 4: User Interface

2.5 Verification

Each functional block was tested individually, and the complete system was evaluated through repeated use in a controlled workspace. Verification focused on performance metrics defined in the high-level requirements, including detection accuracy, system response time, alert triggering, and robustness.

1. Tool Detection: The YOLOv5 model was tested with over 100 tool placement and removal actions. For known tools under normal lighting, the model achieved an accuracy of approximately 93%, which satisfies the detection requirement (90%). Each inference completed in roughly 1.5 seconds, meeting the target latency of under 2 seconds.

2. Alert System: The Raspberry Pi correctly sent a GPIO signal to the Arduino Nano after a tool remained missing for 30 seconds. The Nano triggered both the buzzer and LED reliably. This mechanism was tested repeatedly and met expectations.

3. Control Device: SSH-based access from a laptop worked consistently. Users could view real-time detection logs and camera output, as well as control the system remotely.

However, the system did not meet all the extended functionality and robustness requirements:

1. The system does not support real-time learning of unknown tools. All tools must be pre-trained in the YOLO model.

2. User labeling features were planned but not implemented in the current version.

3. The detection model struggles under inconsistent or poor lighting conditions, which affects accuracy and limits deployment flexibility.

These unmet goals highlight areas for future development. The core functionality—detecting trained tools and issuing alerts—remains effective, but extending the system's adaptability will be necessary for real-world deployment.

A verification summary is provided below:

Feature Tested	**Target/Requirement**	**Result/Status**
Tool detection accuracy	$\geq$90%	$\sim$93%
Detection latency	$<$2 seconds	$\sim$1.5 seconds
Missing tool timeout	30 seconds	Met
Alert signal and activation	LED + Buzzer respond correctly	Met
Control via SSH	Stable remote access	Met
Real-time learning	Supported	Not implemented
User labeling support	Included	Not implemented
Lighting robustness	Works under varied lighting	Failed under harsh lighting

Table 1: System verification results

2.6 Cost

The total cost of the project includes both labor and hardware. Labor cost is calculated using the formula: ideal hourly rate × actual hours × 2.5.

Partner	**Hours Worked**	**Ideal Hourly Rate**	**Total**
Liangcheng Sun	60	$15	$900
Max Mu	60	$15	$900
Total	–	–	**$1800**

Table 2: Labor cost summary

The hardware costs are listed below:

Component	**Quantity**	**Unit Price**	**Total**
Raspberry Pi 4 Model B 4GB 8GB RAM	1	$67	$67
20W USB-C Power Adaptor	1	$14	$14
USB Webcams	2	$15	$30
Arduino Nano Board ATmega328P 5V 16M	4	$5	$20
LED + Buzzer + Wires	1 set	$5	$5
Total	–	–	**$136**

Table 3: Component cost summary

The total estimated project cost is $1936, with the majority coming from labor.

3 Conclusion

The system design successfully combines real-time computer vision with hardware alerts and remote control. While some features like environmental robustness and tool learning are still in progress, the current version demonstrates strong accuracy and responsiveness. The control interface over Wi-Fi adds convenience and flexibility, allowing users to interact with the system from any location within network range.

Future Works:

Planned future work includes expanding the system's capabilities by training a larger custom YOLO model to recognize additional tool types, integrating support for multiple drawers and camera feeds, and implementing persistent tool-tracking with timestamped logs. Once these enhancements are complete, the team aims to deploy the fully featured system in a real laboratory setting.

4 Reference

1. YOLOv5 Documentation – https://github.com/ultralytics/yolov5

2. Raspberry Pi GPIO – https://www.raspberrypi.com/documentation/computers/os.html

3. Arduino Nano Datasheet – https://docs.arduino.cc/hardware/nano

4. OpenCV Camera Capture – https://docs.opencv.org/4.x/dd/d43/tutorial$_{p}y_{v}ideo_{d}isplay.html$