YOLO-Based Binary Object Sorting System

An AI-Powered Cyber-Physical Prototype for Intelligent Object Classification

Sidh Gurnani

The Problem

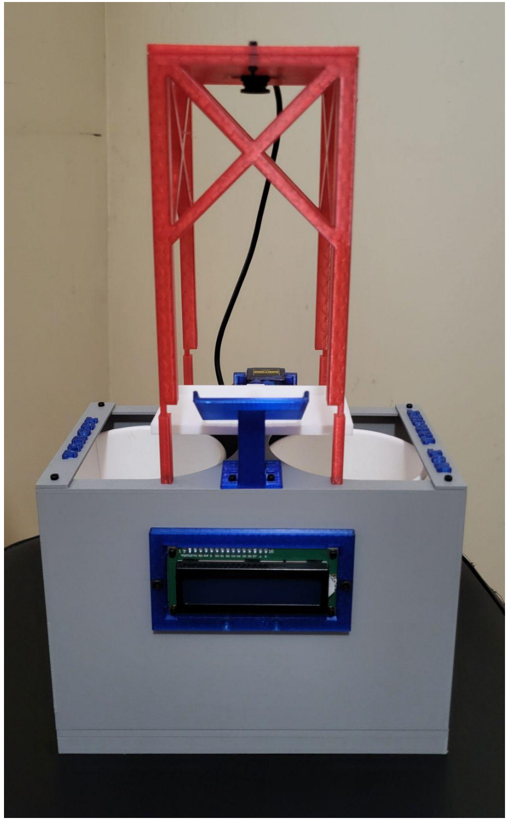
Inefficient Manual Sorting Processes

- Current manual sorting processes in certain sectors are time-consuming and labor-intensive
- There are limited ‘plug and play’ options
    - Sorting processes require lots of time and resources to develop for one specific application
- Possible applications include (not limited to):
    - Manufacturing (electronic components, hardware)
    - Agriculture (varying bean sizes/types)
    - Crime Scene Investigation (bullet casings)
    - Everyday use (LEGOs, coins, etc)

The Solution

AI-Powered Binary Sorting System

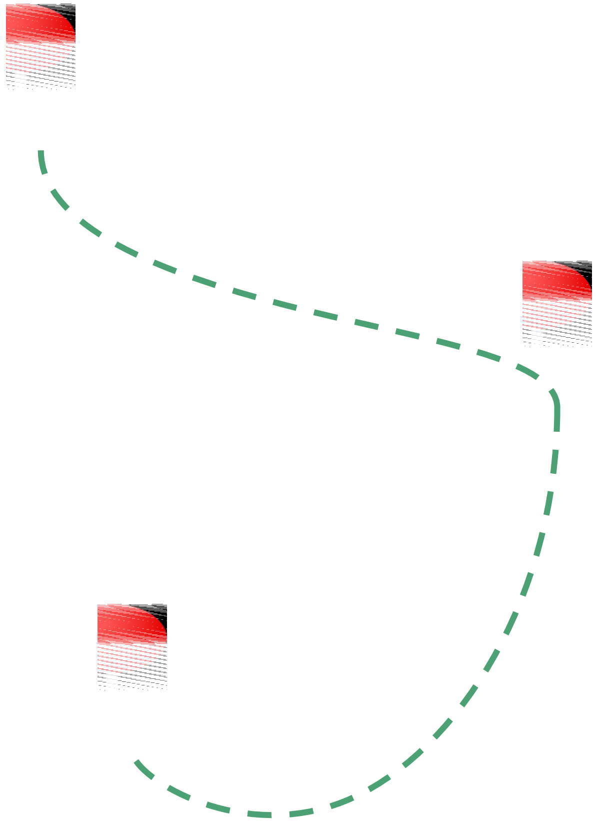
- **Computer Vision**: You Only Look Once (YOLO) Computer Vision Models
    - Offers real-time object classification and can be trained to many use cases
- **Binary Algorithm:** Sorts N objects with N-1 passes
    - No info on how many objects/passes needed with inclusion of an 'Object Detection' setup screen (takes in an image and outputs number and type of objects)
- **Physical Automation:** Servo-controlled platform
    - Directs objects into 'Target' or 'Not Target' bin
- **User-Friendly Interface:** Python GUI with setup wizard



Project Milestones

Project Milestones

1. Refresh Programming Knowledge
2. Understand Basics of Machine Learning
3. Develop Basic Software
4. 3D Model and Design Physical Product
5. Integrate Software and Hardware
6. Final Testing
7. Reflection and Next Steps



Development Journey

Iterative Design Process (click green bubble to view page)

1. **Research Phase:** Explored WebUSB, but pivoted to local Python app
2. **Software V1:** Basic GUI with tkinter; basic, limited functions
3. **Hardware V1:** Vibratory Bowl Feeder (shelved due to complexity)
4. **Software V2:** Modern GUI with Customtkinter; binary sorting framework
5. **Hardware V2:** Servo-based platform design with USB camera
6. **Software V3:** Enhanced V2, with multi-step wizard, live statistics, etc
7. **Final Integration:** Combined software + hardware to perform sorting

How it Works

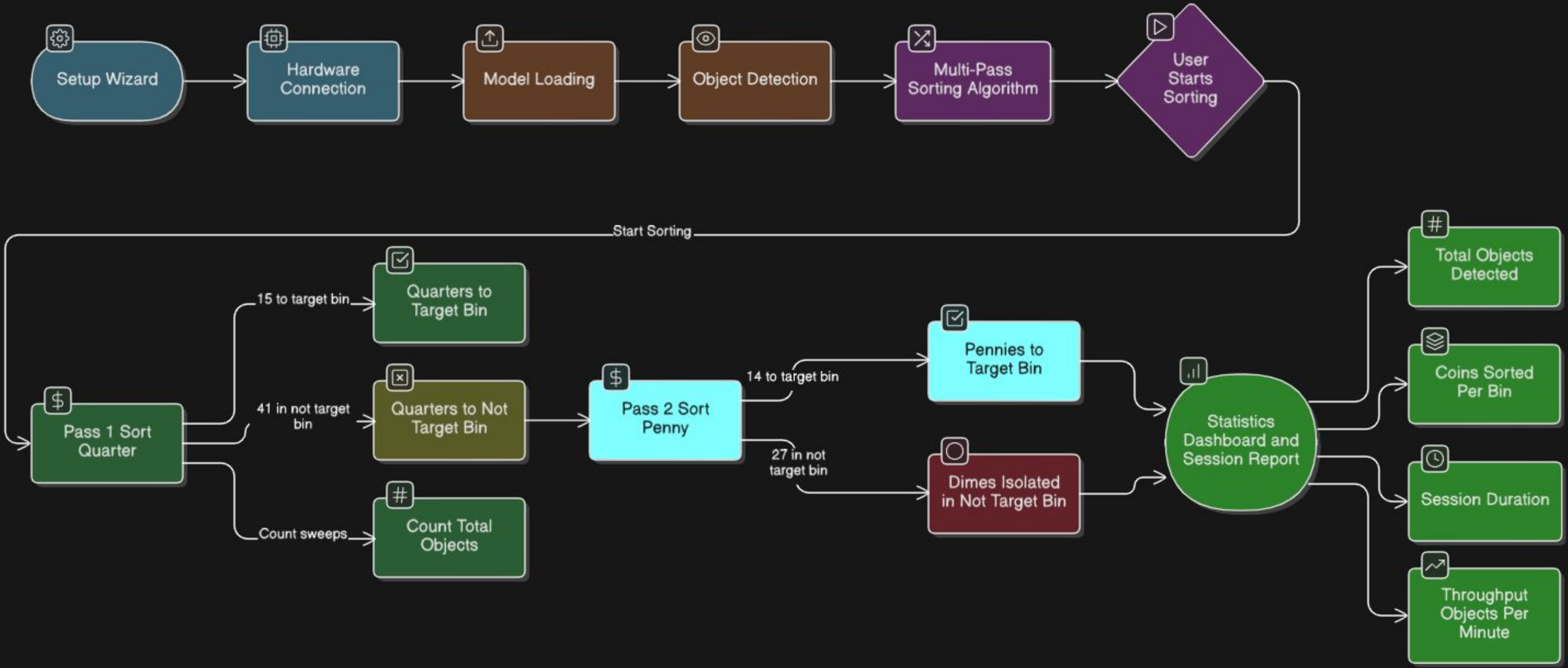
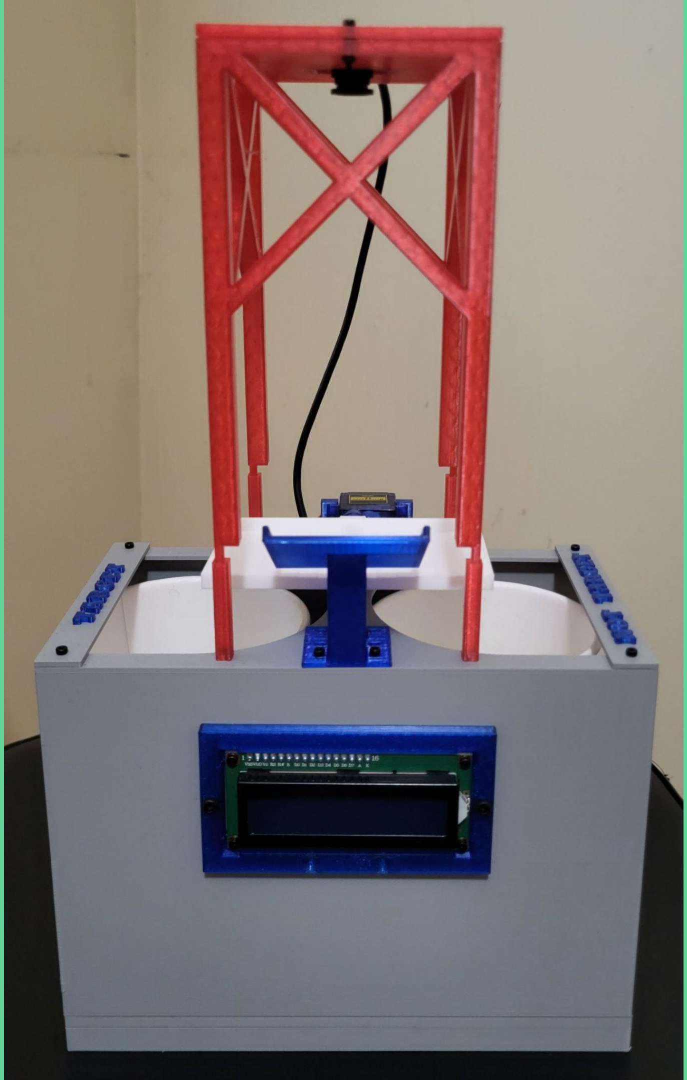
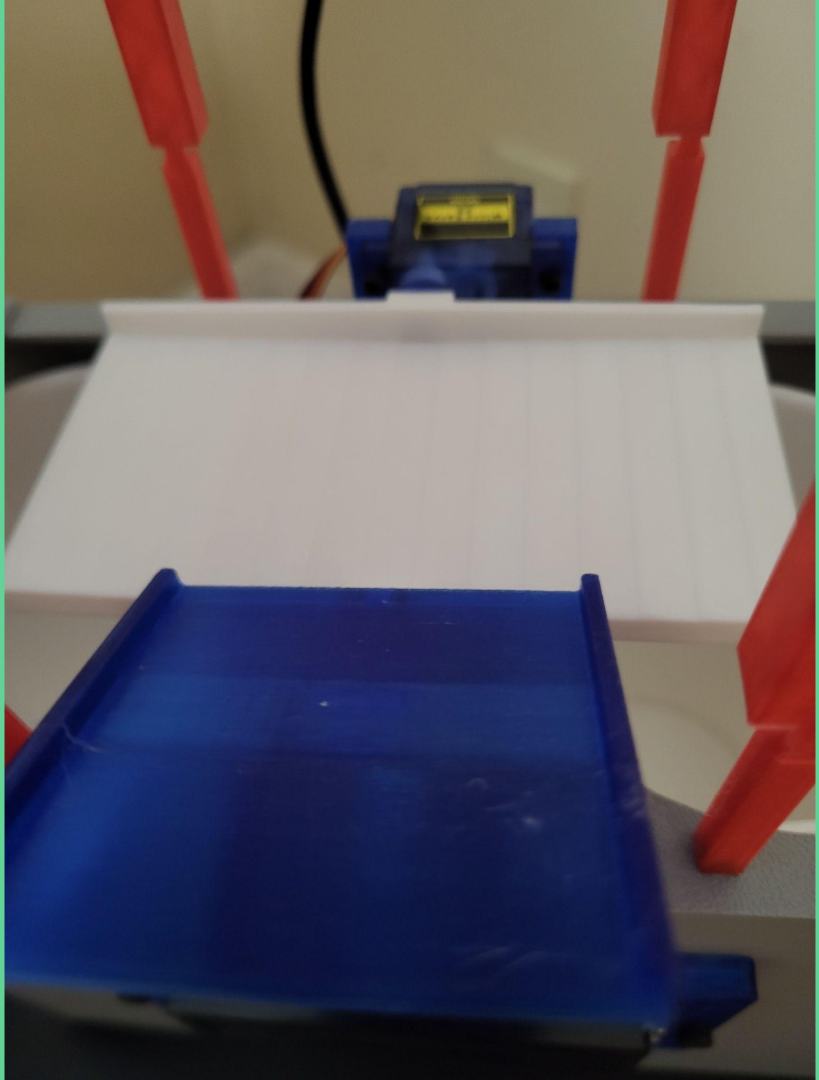
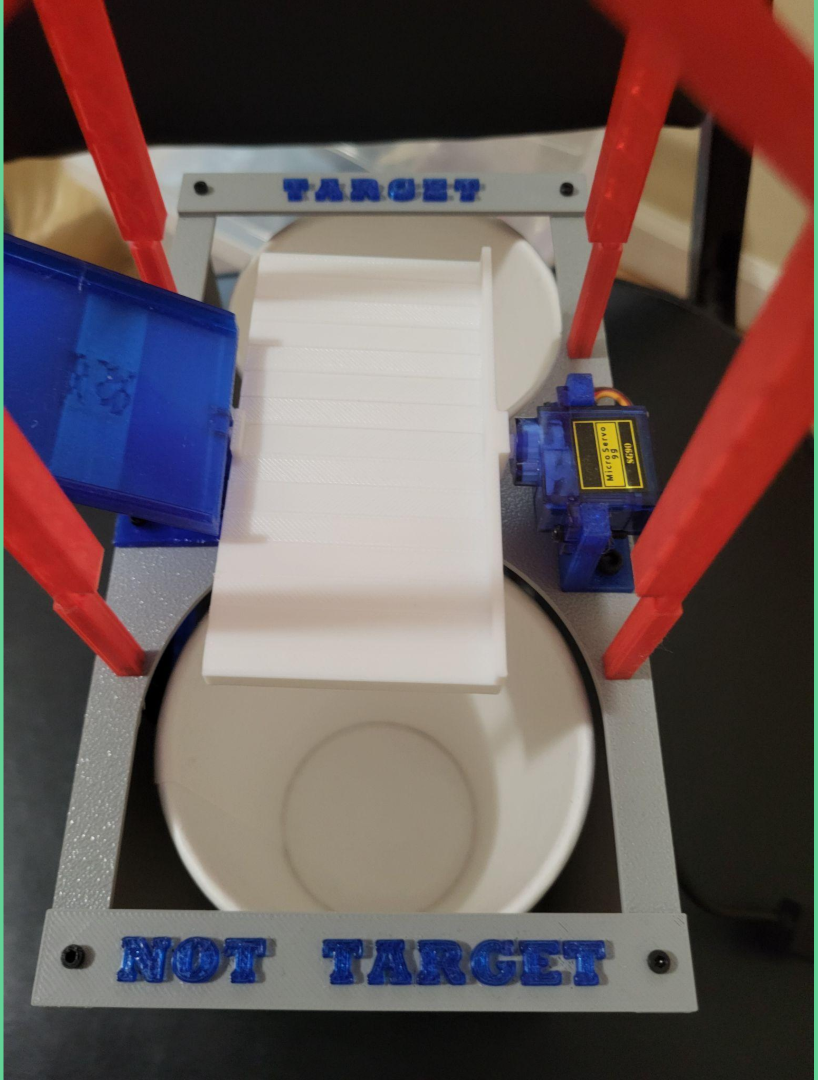


Diagram of how it works with an example using coins

Final Prototype






TARGET
Micro Servo 9g
SG90
NOT TARGET

MUST.
HAVE.
COFFEE.
EXPRESS
CAUTION – it's hot in here
© 2017 IHG. All rights reserved.
Most hotels are independently owned

Click icon to view demo

Testing

Test Setup + Scenario

I decided to test on coins, using an assorted pile of pennies, dimes, and quarters across all tests.

Goal: Compare the performance of the prototype against a human sorting the same set of objects in a binary and non-binary fashion

Testing Plan

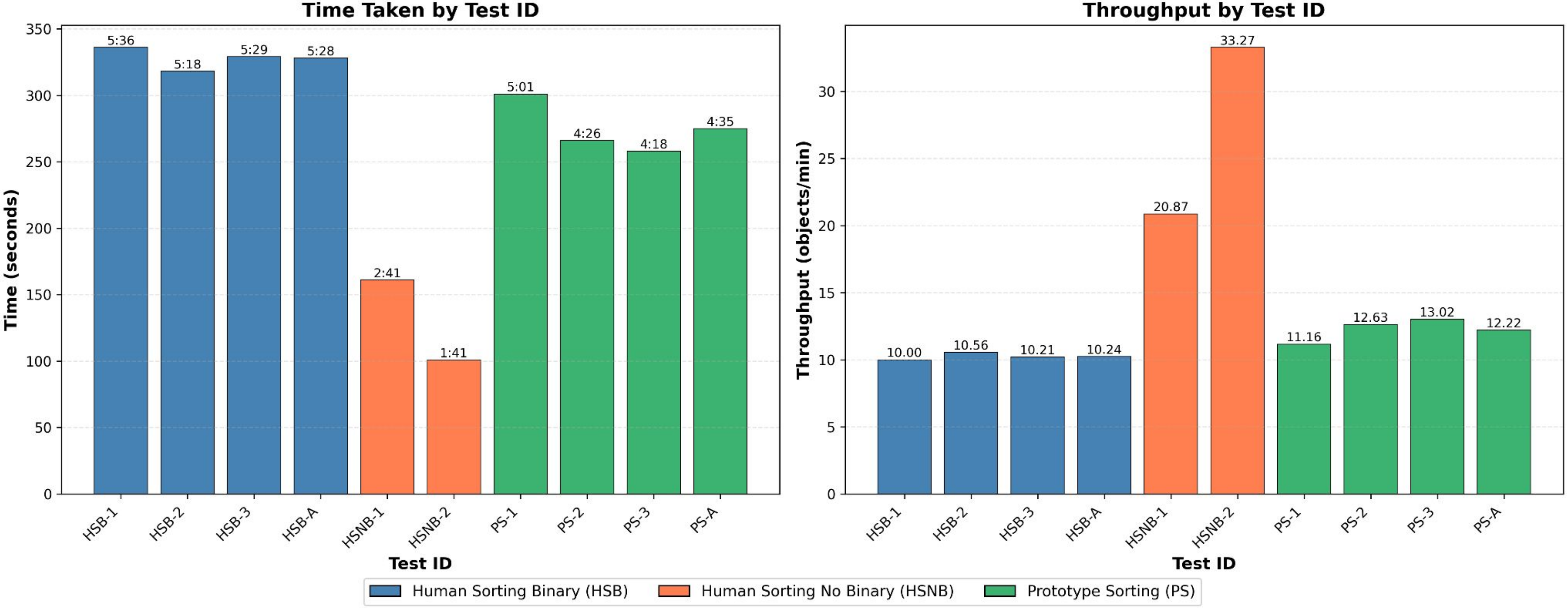
- **Pile Contents:** 15 Pennies, 15 Quarters, 26 Dimes → 56 coins total
- **Tests Ran:**
    - Human Sorting in Binary Fashion (HSB): Take from pile one by one (no looking) filter by target object, then either put into accept/reject pile; afterwards, redo reject pile until all objects sorted
        - HSB-1
        - HSB-2
        - HSB-3
        - HSB-A → Averaged time of above HSB trials
    - Human Sorting in Non-Binary Fashion (HSNB):
        - HSNB-1 → Take from pile one by one (no looking), sort into 3 smaller piles
        - HSNB-2 → See full pile, sort into 3 smaller piles
    - Prototype Sorting:
        - PS-1
        - PS-2
        - PS-3
        - PS-A → Averaged time of above PS trials

Bonus Test

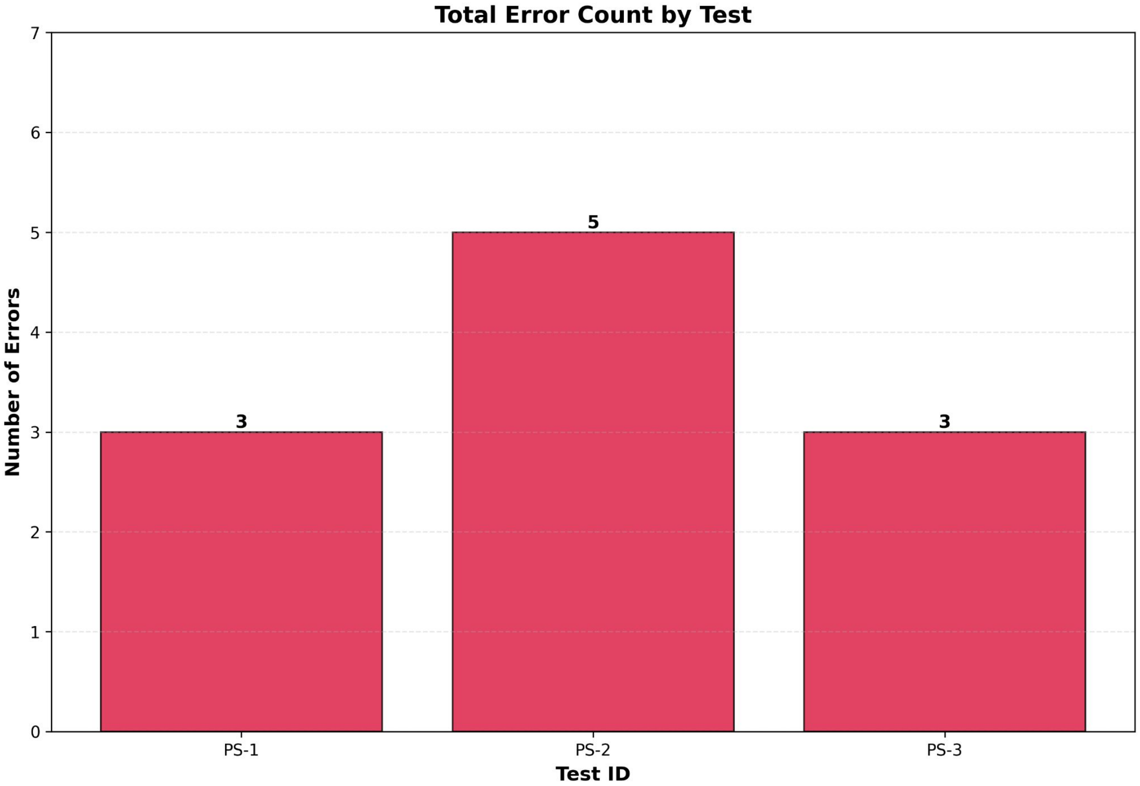
- As this is a binary sorter, one of the advantages is that it can filter out a specific kind of object from the larger pile of the objects
    - This can be configured before starting the sorting process

- **Which method is fastest to filter out pennies from the pile?**
    - **Prototype (PF-1)**
    - **Manual (HF-1)**
        - Take from pile one by one (no looking), see if it is penny; if yes, put in accept pile, otherwise, put in reject pile
    - **Manual (HF-2)**
        - See full pile, sort out all pennies

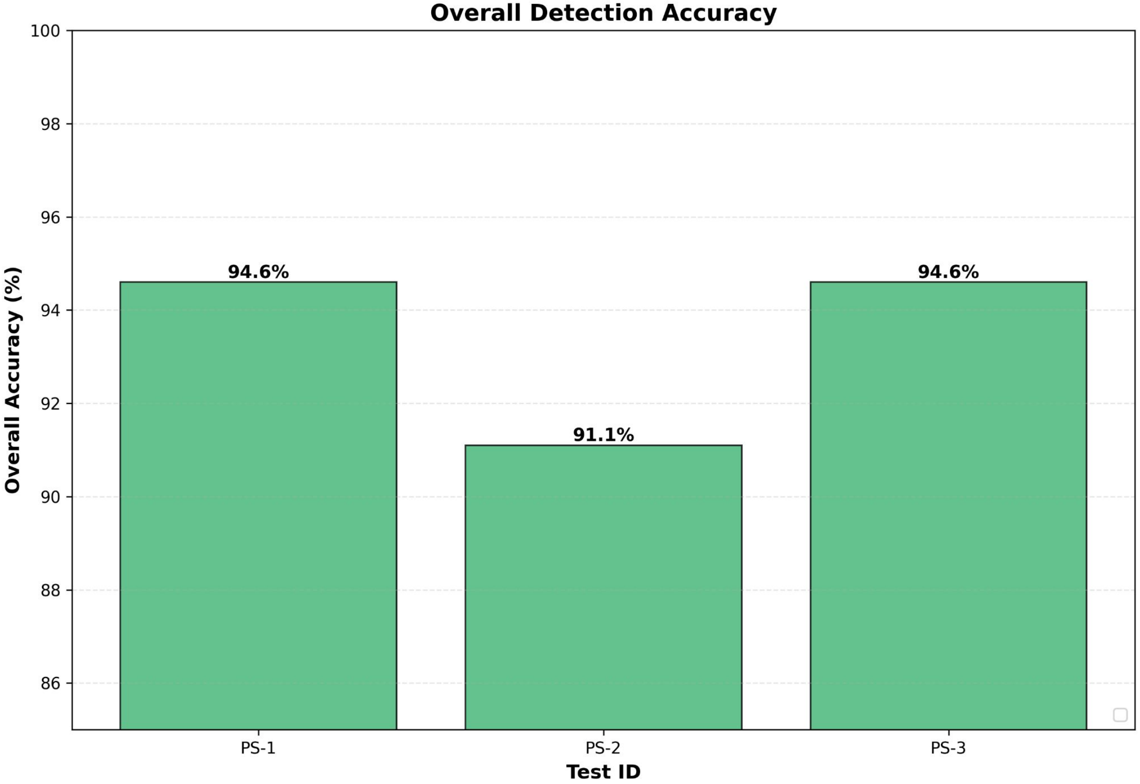
Results - Speed Comparison



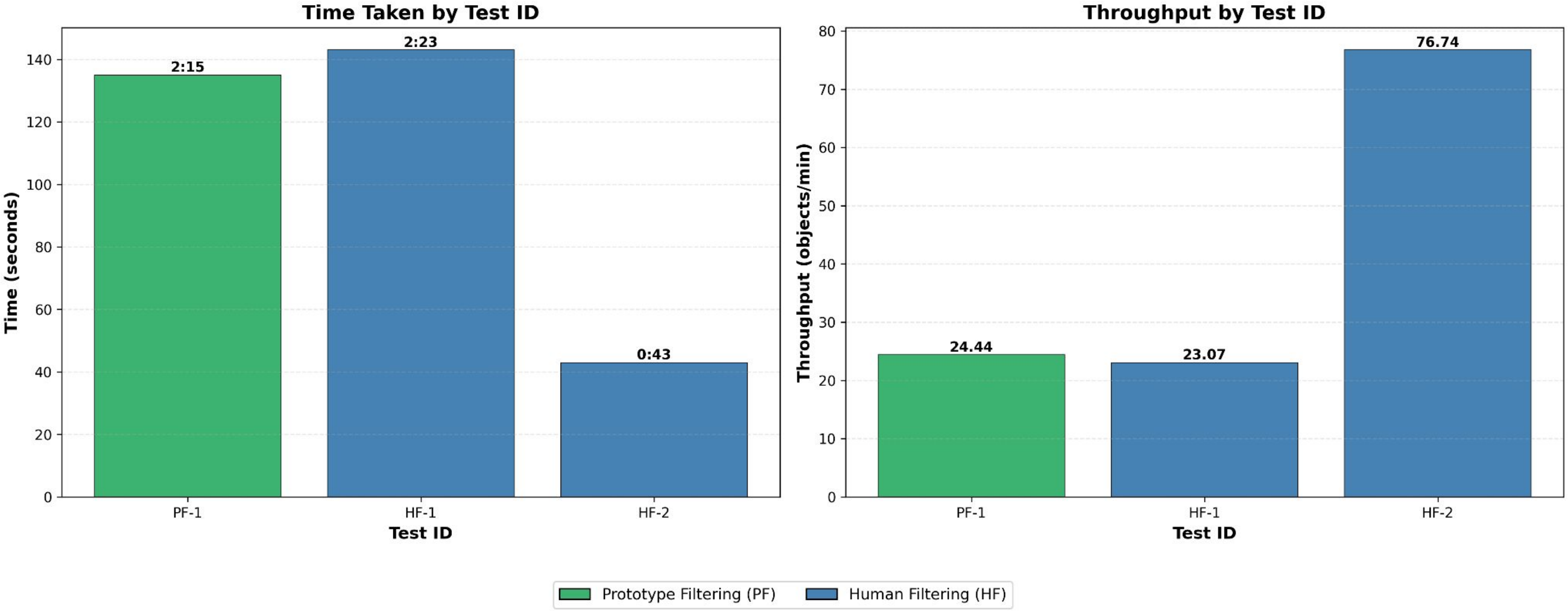
Results - Prototype Error Count



Results - Prototype Accuracy



Results - Penny Extraction Benchmark



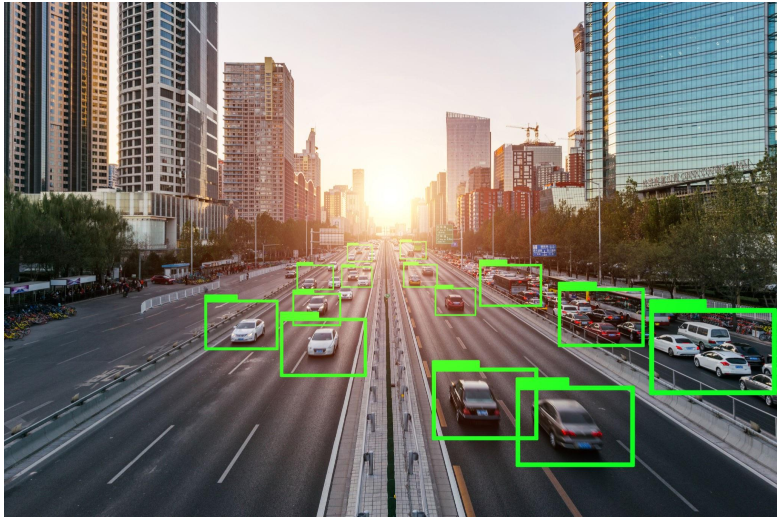
Remarks

- On average, the prototype was around 16% quicker and achieved above 90% accuracy, proving concept viability.
    - This number jumps to as high as 21.3% when comparing best and worst case scenarios between prototype and manual binary sorting
- Prototype was not optimized for speed, and can be made faster:
    - Improve the model being used
        - Can enable faster detection
    - Reduce time system waits for object to exit platform
        - If 0.5 seconds per object can be saved, that would translate to 25 seconds saved for a 50 object batch, which equates to around 8 mins 20 seconds per 1000 objects

Skills Refined/Acquired

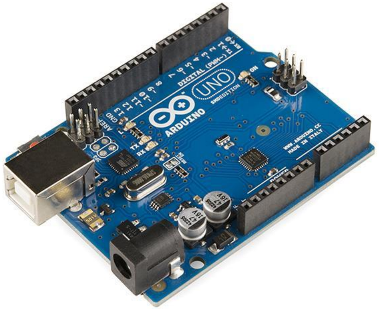
Software Development

- **Python**
  - CustomTkinter → GUI Development
  - OpenCV → Computer Vision for Machine Learning Tasks
  - PySerial → Communication between Arduino and Python App
- **Arduino IDE**
- **GitHub**
- **Machine Learning**
  - Ultralytics YOLO → Used to train models to detect objects
- **Real-time computer vision processing**
- **Multi-threaded communication protocols**
- **Data analysis & statistics tracking**

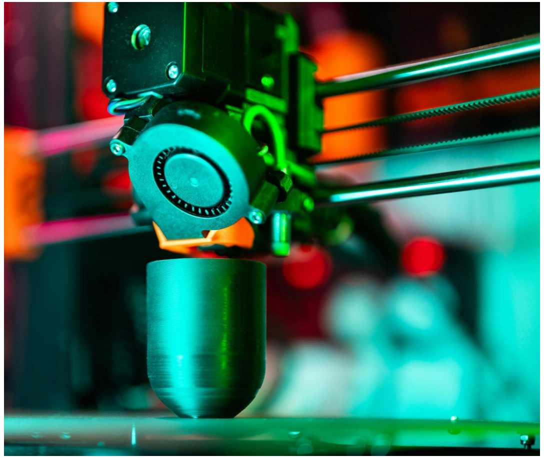


Hardware Integration

- **Arduino (C/C++)**
- **Integrating electrical components into larger design:**
  - Servo
  - LCD Display
  - LED Indicators
- **Software/Hardware Calibration**
  - Wrote test code to ensure all systems functioned correctly before testing
- **CAD Modelling**
  - Onshape
- **FDM 3D Printing**

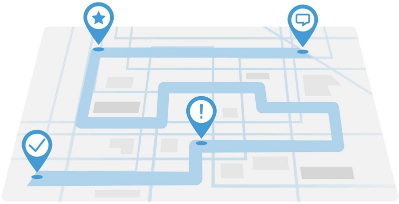


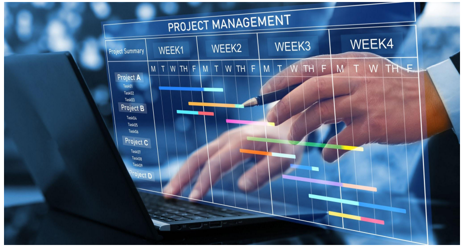
onshape®

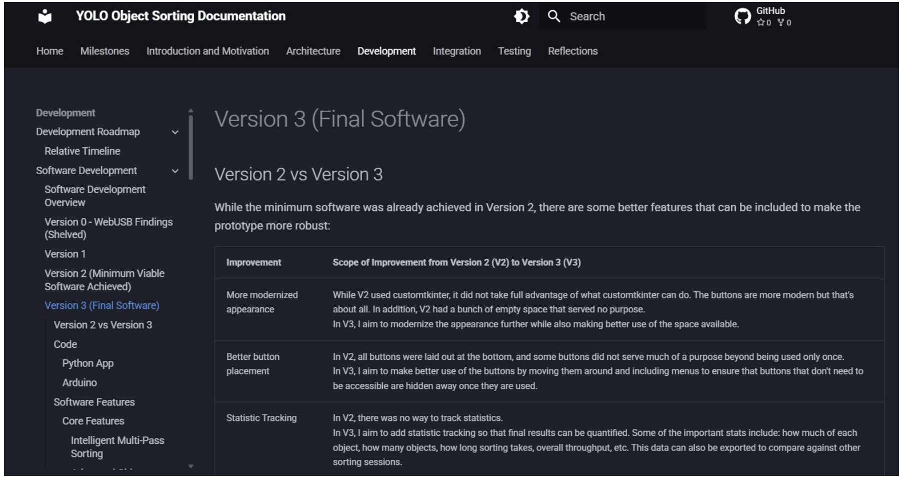


Miscellaneous Skills Developed

- **Project Management**
  - Scope Management - Shelved WebApp and Bowl Feeder to maintain focus on core functionality
  - Roadmap developed before starting any part of the project
- **Technical Communication**
  - Documentation - detailed MkDocs page with knowledge, development guide, tutorials, troubleshooting, etc
  - Knowledge Transfer - reproducible guides on YOLO model training and system setup







Version 3 (Final Software)

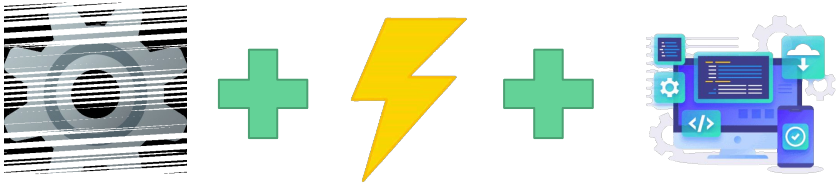
Version 2 vs Version 3

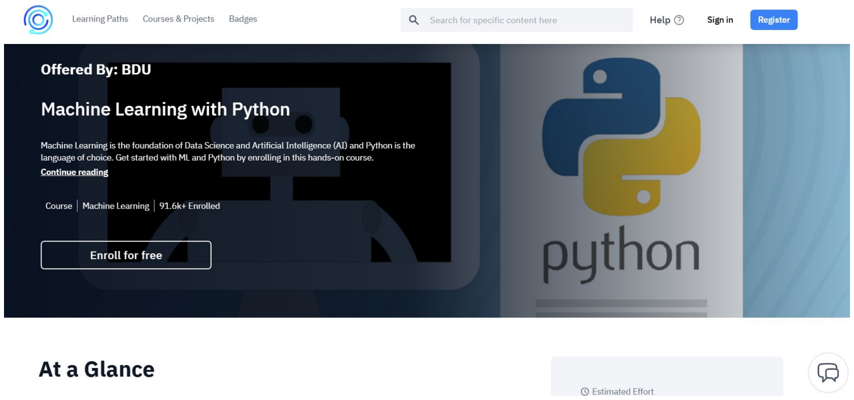
While the minimum software was already achieved in Version 2, there are some better features that can be included to make the prototype more robust:

Improvement	Scope of Improvement from Version 2 (V2) to Version 3 (V3)
More modernized appearance	While V2 used customtkinter, it did not take full advantage of what customtkinter can do. The buttons are more modern but that's about all. In addition, V2 had a bunch of empty space that served no purpose. In V3, I aim to modernize the appearance further while also making better use of the space available.
Better button placement	In V2, all buttons were laid out at the bottom, and some buttons did not serve much of a purpose beyond being used only once. In V3, I aim to make better use of the buttons by moving them around and including menus to ensure that buttons that don't need to be accessible are hidden away once they are used.
Statistic Tracking	In V2, there was no way to track statistics. In V3, I aim to add statistic tracking so that final results can be quantified. Some of the important stats include: how much of each object, how many objects, how long sorting takes, overall throughput, etc. This data can also be exported to compare against other sorting sessions.

Miscellaneous Skills Developed (cont’d)

- **Systems-Level Design + Integration**
  - Mechanical Subsystem
  - Electrical Subsystem
  - Software Subsystem
- **Trade-off Analysis**
  - Balanced speed, accuracy, cost, and complexity throughout project
- **Test Planning**
  - Designed testing protocol across various situations with repeatability
- **Self-Directed Learning**
  - Took courses to fill knowledge gaps
  - Self-taught GitHub, Roboflow, ML, Serial Protocols, etc





Offered By: BDU

Machine Learning with Python

Machine Learning is the foundation of Data Science and Artificial Intelligence (AI) and Python is the language of choice. Get started with ML and Python by enrolling in this hands-on course.

Continue reading

Course | Machine Learning | 91.6k+ Enrolled

Enroll for free

At a Glance

Estimated Effort

Reflection

Key Learnings & Insights

1. **Scope Management:** Reprioritized object feeding from “must have” to “nice to have”, allowing greater focus on core sorting functionality, and resulted in a working prototype that can accommodate a feeding mechanism
2. **Environment-Specific Training:** No need to generalize when creating a system that is super specific, and the best way to ensure reliability is to train model in the deployment environment
3. **User Experience Importance:** End user experience is a core focus, and making the system intuitive important, even for a prototype/proof-of-concept
4. **Hardware-Software Integration:** One-way communication and repeated messages reduced the complexity compared to bidirectional communication

What would the next version/iteration look like?

Software:

- Error handling and bug fixes
- More robust data storage
- Performance modes: conservative, normal, rapid
- Mode to allow for easy video recording (for model training)

Hardware:

- Overall redesign for tool access and assembly purposes
- Better looking mounting for screen
- Move dimming knob (potentiometer) to actual body rather than breadboard
- Put all electronics in a ‘black box’ to hide away all components
- Better cable management within prototype
- Add lighting to make viewing objects easier, so webcams can capture most details

What would I change if I had to do it again?

- Source a higher quality camera
- Further explore idea of a web app to run on lower end devices
- Switch away from a pure binary concept and opt for having multiple targets instead of a singular target
- Slim down the size of the physical product even further
- More upgraded camera housing module, where objects are fed into an enclosed, well-lit box
- Platform geometry/size to allow for a wider selection of objects
- Experiment with a different microcontroller
- Design a system for object feeding

Project Resources

Click or scan the QR Codes to access the resource

MkDocs Documentation

All Project Files (GitHub Repo)

Demo Video (YouTube)

Thank you!

Email: sidh.gurnani@gmail.com
LinkedIn: https://www.linkedin.com/in/sidh-gurnani/
Portfolio: https://sidhgurnani.github.io/sidh-ME-portfolio/