Tackling Parking Lot Congestion at the University at Buffalo's North Campus with Car Automated Recognition System (CARS)

Jordan Wang, Ananta Sharma
September 29, 2023
CSE321 Realtime and Embedded Systems
Bina Ramaurthy

Table of Contents

Hardware Component Prototype

a. Hardware Components List

Part	**Description**	**Connections**
Raspberry Pi 4	A small microprocessor capable of running object classification models.	Connected to the power supply, Pi Camera, and LCD display. See below for detailed connections.
Raspberry Pi Camera Module 3 Wide NoIR	A camera, built specifically for Pi's, with wide angle lens and low light capabilities.	Connected to the Camera Serial Interface Type 2 (CSI-2) on the Pi.
LCD1602 with I2C	A two row LCD display, attached with a I2C backpack, capable of displaying 16 characters per row.	Connected to Pi's GPIO pins. • LCD GND → GPIO6 (GND) • LCD VCC → GPIO2 (+5V) • LCD SDA → GPIO3 (SDA) • LCD SCL → GPIO5 (SCL)
Power Supply	Either a wall outlet or a portable charger.	Connected to the Raspberry Pi 4 power port through a USB-C to USB-A connection.

b. Hardware Connection Diagram

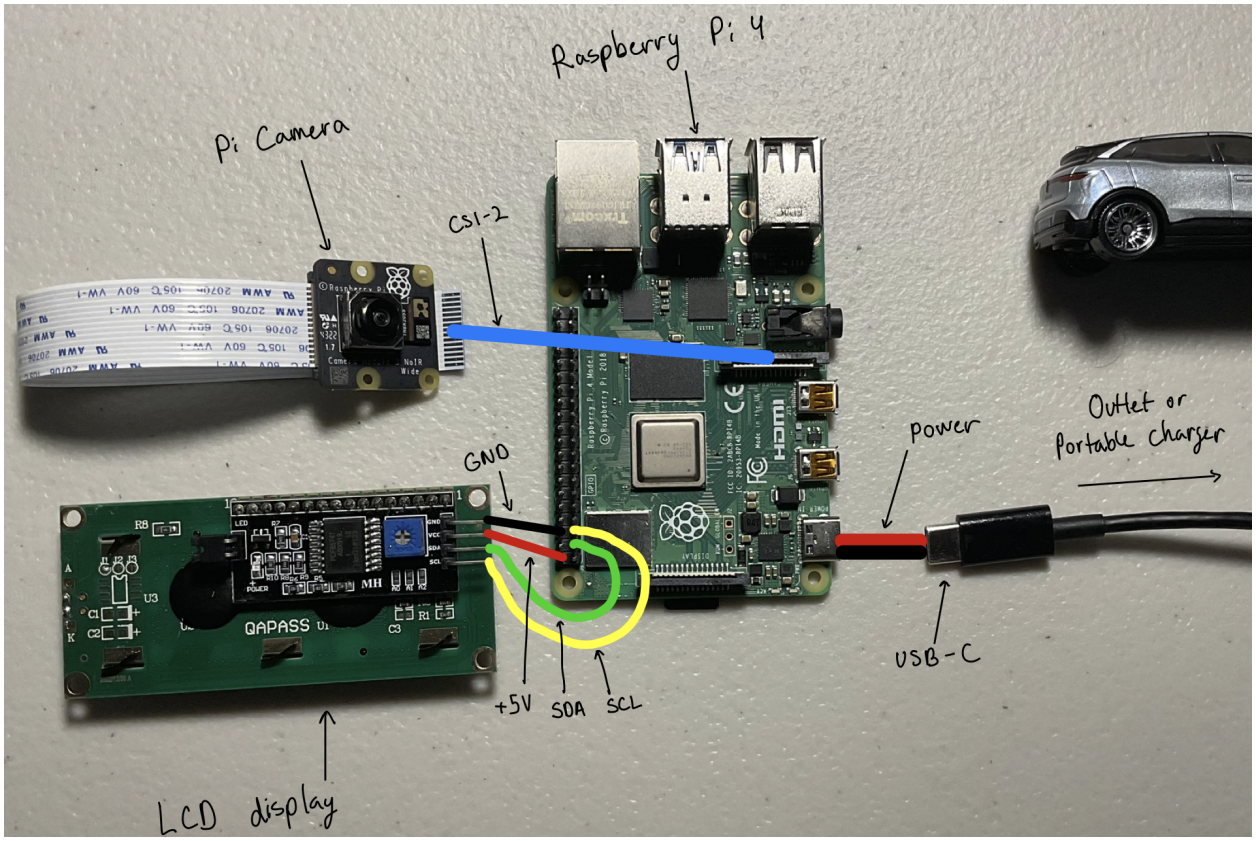

Figure 1: *System Parts and Connections*
Source: Jordan Wang

Software Component Prototype

a. Car Detection Script

```
from ultralytics import YOLO
from picamera2 import Picamera2
import time
import os
import subprocess
from RPLCD.i2c import CharLCD

# setup LCD
lcd = CharLCD(i2c_expander='PCF8574', address=0x27, port=1, cols=16, rows=2,
dotsize=8)
lcd.clear()

# only log errors for camera
os.environ["LIBCAMERA_LOG_LEVELS"] = "3"

# load model
model = YOLO('yolov8s.pt')

# initialize camera object
picam2 = Picamera2()
picam2.start()

# setup camera configuration for taking image
capture_config = picam2.create_still_configuration()
time.sleep(1)

while True:

    # number of cars
    count = 0

    # capture and save image
    picam2.switch_mode_and_capture_file(capture_config, "image.jpg")

    # run inference on image
    results = model("image.jpg")
    # get results
    cls = results[0].boxes.cls
```

```
    print(cls)
    # print out number of cars detected
    for result in results:
        boxes = result.boxes
        cls = boxes.cls
        # if cls.data[0] == 2:
        for i in enumerate(cls):
            print("car!")
            count = count + 1
    #child.communicate(count)
    lcd.write_string(f'cars: {count}')
    time.sleep(1)
    lcd.clear()
```

b. Model Training (Google Colab)

```
!nvidia-smi

!pip install ultralytics

import IPython
from ultralytics import YOLO
import os
from IPython.display import display, Image
from IPython import display
display.clear_output()
!yolo checks

!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="MY_API_KEY")
project = rf.workspace("WORKSPACE").project("PROJECT")
dataset = project.version(1).download("yolov8")

!yolo detect train model=yolov8m.pt data={dataset.location}/data.yaml epochs=20
imgsz=640 workers=0 patience=50 verbose=False

!yolo detect val model=/content/runs/detect/train/weights/best.pt
data={dataset.location}/data.yaml

!yolo predict model=/content/runs/detect/train/weights/best.pt
source=/content/PKLot-2/custom/images/cars.jpg
```

Test Scripts and Patterns

Test Suite 1 - Basic Functionality

Test Case 1 - Camera Function

1. Start the camera preview program.
2. Expect that the camera boots up and displays a preview.
3. **Result: Camera displays preview.**

Test Case 2 - LCD Display Function

1. Print “Hello World!” to LCD display.
2. Expect “Hello World!” to be displayed.
3. **Result: “Hello World!” is displayed**

Test Case 3 - Model Detection

1. Run inference on image of cars in parking lot
2. Expect that the model detects cars and indicates in output.
3. **Result: Model indicates cars are present in the image.**

Test Case 3 - Empty Parking Lot.

1. Upload image of empty parking lot and run script.
2. Expect that the LCD displays “cars: 0”.
3. **Actual: Nothing is displayed.**

Test Case 4 - Camera shows partially filled parking lot.

1. Start the program.
2. Expect that the model picks up multiple empty lots in the parking lot.
3. **Actual: Model does not pick up on empty lots**
4. Expect that the camera picks up multiple filled up lots in the parking lot on the lcd display.
5. **Actual: Model detects cars.**

Test Suite 2 - Program Execution

Test Case 1 - Program keeps running until turned off

1. Start the program.
2. Expect the program to keep running.
3. **Actual: Program keeps running.**

Test Case 2 - Disrupt Camera functions

1. Start the program.
2. Block the camera view.
3. Expect the “error” output on the LCD display.
4. **Actual: No error output.**
5. Unblock view and see the program continues as normal.

6. **Actual: Program continues as normal.**

Test Case 3 - Interrupt the program

1. Start the program.
2. Force reboot the program.
3. Expect that the LCD display is reset and does not show previous results.
4. **Actual: LCD does not show previous results.**
5. Let the program continue and see its working normally.
6. **Actual: Program runs normally.**

Test Suite 3 - Edge Cases **(Future Testing)**

Test Case 1 - Camera Obstruction

1. Start the program.
2. Leave some obstacles in the empty lots in the parking lot.
3. Expect that the display shows that the lot is occupied.
4. Now let a person occupy the space.
5. Expect that the LCD display shows that the lot is occupied.
6. Empty the lot and check the lcd display shows the lot is vacant.

Test Case 2 - Snowy Weather Conditions

1. Start the program in snowy weather conditions i.e. snow falling and on the ground.
2. Expect that the camera works properly and outputs correct information to the LCD display.

Test Case 3 - Low Light Conditions

1. Start the program in low light conditions i.e. night time.
2. Expect that the camera works as intended and outputs correct information on the LCD display.

Test Case 4 - Multiple interactions in parking lot

1. Start the program.
2. Make sure at least 2 cars are moving out and 2 cars are moving into vacant lots.
3. Expect that the camera detects both actions and displays correct results to the LCD display

Photos of the Prototype

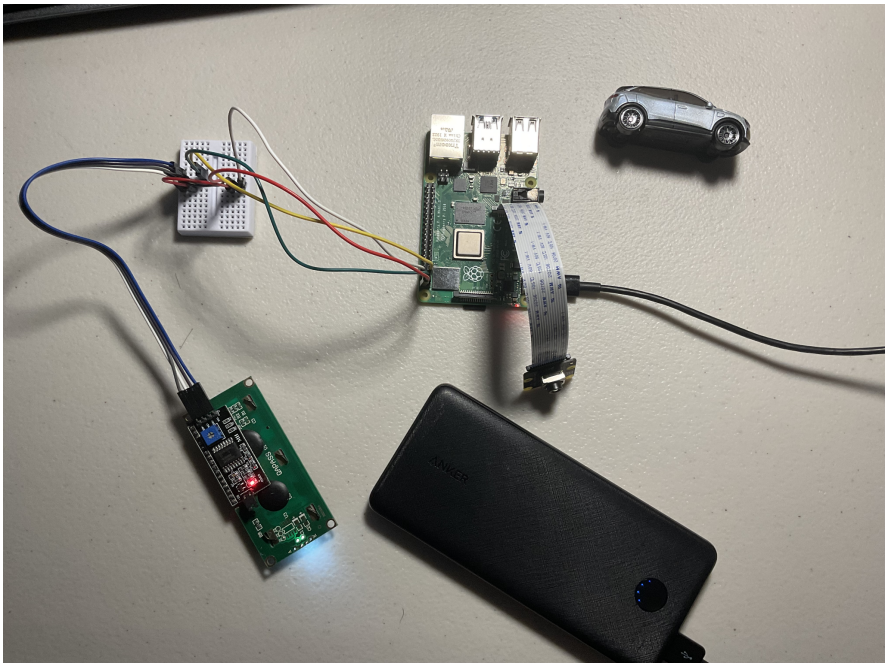

Figure 2: *Complete Prototype*
Source: Jordan Wang

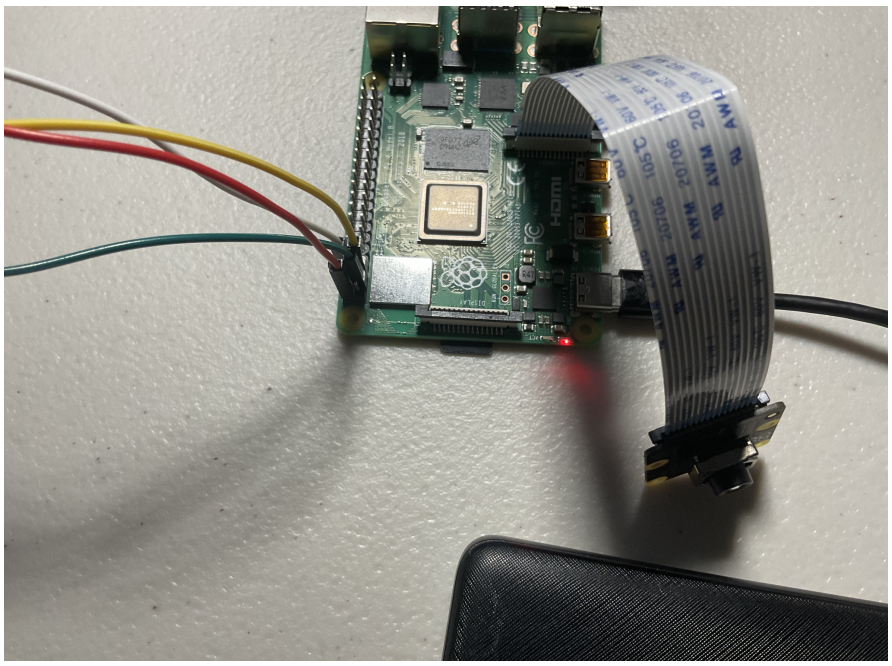

Figure 3: *GPIO Connections*
Source: Jordan Wang

Figure 4: *Camera CSI-2 Connection*
Source: Jordan Wang

```
pi@raspberrypi:~ $ ls
Bookshelf  Documents  Music     Public     Videos
Desktop    Downloads  Pictures  Templates
pi@raspberrypi:~ $ cd Documents/pytorch_setup/
pi@raspberrypi:~/Documents/pytorch_setup $ source env/bin/activate
(env) pi@raspberrypi:~/Documents/pytorch_setup $ ls
bus.jpg         env         opencv_stream.py  runs        yolov8n.pt
display_lcd.py  image.jpg   requirements.txt  stream.py   yolov8s.pt
(env) pi@raspberrypi:~/Documents/pytorch_setup $ python stream.py

image 1/1 /home/pi/Documents/pytorch_setup/image.jpg: 384x640 1 person, 1 dining
 table, 1 cell phone, 2581.4ms
Speed: 35.5ms preprocess, 2581.4ms inference, 48.9ms postprocess per image at sh
ape (1, 3, 384, 640)
```

Figure 6: *Setup and Successful Script Execution*
Source: Jordan Wang

Figure 5: *One Car Test with LCD and Terminal Output*
Source: Jordan Wang

Observations and Notes

1. Currently the camera is picking up cars but not accurately so we still need to fix that.
2. Had to think about different kinds of edge cases for this project, one of them was obstruction on the parking lots. For example, a group of people can be occupying one empty lot, and the camera may not be able to determine if the lot is empty as it covers the whole not and not just a portion of it. In this case, since a group of people are currently in that lot, the camera should mark it as occupied.
3. Camera currently inaccurately displays how many cars are present or not in the LCD display.
4. We are currently using toy cars to demo this project, but the idea was for real parking lots and that may be a challenge for us to test it out. We will look into testing on UB parking lots.
5. We will continue to work on training the model for this project so it displays accurately if a lot is occupied or not.
6. We had some trouble accessing the camera output since OpenCV camera capture was no longer supported for Raspbian Bullseye.
7. We noticed that the original LCD display required a lot of wiring so we decided to get one with a I2C backpack.
8. We ran into segmentation faults while trying to run inference on the model. We fixed this by downgrading our torch and torchvision libraries to an earlier release.
9. We had trouble accessing python libraries installed using apt-get so the fix was to recreate the virtual environment with flag --system-site-packages.
10. In the future, have the script run on startup so there is no need to login to the Pi.
11. In the future, have the Pi transmit data wirelessly so that it can be mounted to a higher place, but the output is still accessible.

CRC Cards

Car Automated Recognition System (CARS)	
1. Real-time Car detection (input). 2. Update LCD display with the number of cars (output). 3. Continuous monitoring and counting until the system is turned off (computation).	1. LCD Display (Jordan) 2. Pi Camera Module (Jordan) 3. Raspberry Pi 4 (Ananta)

Pi Camera (Input)	
1. Continuous image capture of parking lots. 2. Camera works in low light conditions.	1. Raspberry Pi 4

Raspberry Pi (Computation)	
1. Camera integration 2. Image processing and classification 3. Data storage 4. Counting logic for number of cars in the parking lot.	1. Camera 2. LCD Display

LCD Display (Output)	
1. Display how many cars are occupying empty lots. 2. Get the output from the Raspberry Pi. 3. Outputs when cars are found or if there is an error. 4. Display update in real-time.	1. Raspberry Pi 4

References

- https://github.com/niconielsen32/YOLOv8-Class/blob/main/YOLOv8InferenceClass.py
- https://github.com/ultralytics/ultralytics/tree/main
- https://medium.com/@thedyslexiccoder/how-to-set-up-a-raspberry-pi-4-with-lcd-display-using-i2c-backpack-189a0760ae15
- https://rplcd.readthedocs.io/en/stable/
- https://docs.ultralytics.com/
- https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf
- https://www.tomshardware.com/how-to/use-picamera2-take-photos-with-raspberry-pi
- https://stackoverflow.com/questions/55600132/installing-local-packages-with-python-virtualenv-system-site-packages
- https://github.com/ultralytics/ultralytics/issues/5274#issuecomment-1765965550
- https://stackoverflow.com/questions/73573891/hide-libcamera-info-in-python