Backup Camera and Sensor

Joanna Dorval
CS370: Operating Systems
Colorado State University
Fort Collins, United States
joanna.dorval@colostate.edu

Paul Gaudin
CS370: Operating Systems
Colorado State University
Fort Collins, United States
Paul.Gaudin@colostate.edu

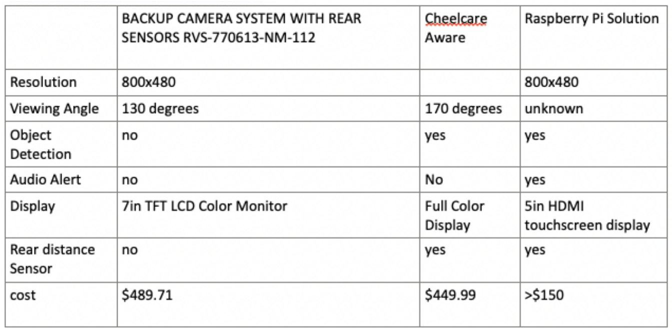
Anthony Juvera
CS370: Operating Systems
Colorado State University
Fort Collins, United States
Anthony.Juvera@colostate.edu

***Abstract*—Wheelchair users face significant challenges when traveling in reverse due to limited visibility, leading to increased risk of collision, damage, or injury. Existing commercial backup-assistance systems can cost hundreds if not thousands, making them inaccessible for many users. This project proposes a low-cost, effective solution utilizing a Raspberry Pi-based system. The system incorporates SanDisk 128GB SSD, Raspberry Pi Camera Module V2, 5 inch HDMI touchscreen, and various peripherals from a Sunfounder starter kit like an ultrasonic distance sensor. The system provides real-time visual and proximity feedback, significantly reducing the risk of accidents. Ultimately, this solution significantly enhances wheelchair users' safety and independence, demonstrating a viable pathway toward accessible assistive technology.**

***Index Terms*—Wheelchair navigation, Raspberry Pi 5, Backup camera system, Ultrasonic sensor, Affordable accessibility.**

I. Introduction

Individuals who require the use of a wheelchair have a considerable number of barriers to overcome while maneuvering inside constrained spaces [1]. Wheelchair users often face limited rear visibility leading to a risk of collision when backing up. Bumping into obstacles or people, can cause injury or damage and can compromise independence. High-end commercial solutions exist, however, these enhanced obstacle detection electronics are often expensive costing up to $10,000 [2]. Wheelchairs equipped with backup cameras often come at a premium, with full systems costing users thousands of dollars or more. Even aftermarket add-ons, like the Cheelcare AWARE system, are between $400 and $500 [8].



	BACKUP CAMERA SYSTEM WITH REAR SENSORS RVS-770613-NM-112	Cheelcare Aware	Raspberry Pi Solution
Resolution	800x480		800x480
Viewing Angle	130 degrees	170 degrees	unknown
Object Detection	no	yes	yes
Audio Alert	no	No	yes
Display	7in TFT LCD Color Monitor	Full Color Display	5in HDMI touchscreen display
Rear distance Sensor	no	yes	yes
cost	$489.71	$449.99	>$150

Fig. 1. Comparison of Backup Camera Solutions [14] [15]

.

This report outlines a project designed to directly address these issues by developing a low-cost, effective backup assistance system that utilizes widely available technology. Centered around a Raspberry Pi 5, paired with a SanDisk Extreme pro 128GB SSD card for fast boot and storage. The system incorporates a Raspberry Pi Camera Module v2, a 5-inch touchscreen display, and a buzzer to provide real-time visual and auditory feedback based on proximity data captured by an HC-SR04 ultrasonic sensor. All peripherals are mounted on a breadboard from the SunFounder Raphael Ultimate Starter kit, which may also supply additional sensors for future upgrades. This proposed solution and configuration provides users with real-time visual and proximity feedback, significantly improving spatial awareness and reducing the risk of accidents during reverse navigation.

The primary objectives of this project are to provide an affordable and effective solution that empowers wheelchair users by increasing their independence while improving their overall quality of life. Through integration of accessible technology and user-centric design, our system presents a viable alternative to costly commercial products, opening opportunities towards greater safety and inclusive design for wheelchair users. In the rest of this report, we present a detailed technical characterization of the problem, our proposed solution along with its implementation strategy, the outcome of the developed prototype, and conclusion.

II. Problem Characterization & Technical Requirements

The consequences of reversing without adequate feedback can be severe. Colliding with unexpected objects or people may result in significant damage, serious injury, or even cause the wheelchair to tip over. Research indicates that maneuvering backward is among the most frequently reported challenges in daily wheelchair navigation [1]. Without adequate sensory feedback, users are left navigating with insufficient data, an issue worsened by poor lighting, noise, or spatial congestion. With environments being dynamic and obstacles can move, as well as the wheelchair itself, this problem requires the system to operate effectively in both indoor and outdoor settings.

To address these technical challenges, our system had to meet several key requirements. By integrating a camera for rear visual and a distance sensor to accurately detect obstacles within a reasonable distance behind the user (TODO distance), while displaying this information to the user without a long delay (less than 300 milliseconds latency). The system must operate continuously and concurrently capturing live video, monitoring distance sensors, and issuing alerts all at once. The system should be lightweight, compact, and easy to mount onto a wheelchair frame, aiming to improve safety by addressing these challenges. The system turns a regular wheelchair into a smarter, context-aware device that reduces the risk of collision when backing up.

The following sections describes our proposed solution, the peripherals used in our system as well as the libraries associated with them.

A. Proposed Solution

Our purposed solution is a modular, low-cost system centered around a Raspberry Pi 5 paired with a SanDisk Extreme Pro 128GB SSD. It uses sensor input, camera for vision, and user feedback components integrated on a breadboard from the SunFounder Raphael Ultimate Starter kit.

B. Component breakdown and purpose

- `Raspberry Pi 5`: Featuring a 2.4GHz quad-core 64bit Arm Cortex-A76 CPU. The Raspberry Pi 5 offers processing speeds that are "over twice as fast as its predecessor" according to Upton from the Raspberry Pi foundation [3]. Previous models were built on a monolithic AP architecture that was becoming both technically and economically unsustainable. However, the Raspberry Pi 5 is built on a disaggregated chiplet architecture where only major fast digital functions and interfaces are provided by the AP. RP1, the I/O controller for Raspberry Pi5 provides improved throughput reading from peripherals [3]. With our video-driven and sensor-intensive workload, this new and significantly improved performance directly benefits this project.
- `SanDisk Extreme Pro 128GB SSD`: This flash drive delivers maximum performance and has the ability to transfer huge files extremely fast. The sequential read performance is up to 420mb/s and write speeds up to 380mb/s [4]. This high-speed data transfer capability ensures rapid boot times and efficient handling of real-time data logging and video processing tasks essential for this system. Its solid-state design offers enhanced durability, making it suitable for the mobile and potentially rugged environment of a wheelchair mounted system.
- `Raspberry Pi Camera module V2`: This camera module comes with a Sony IMX219 8-megapixel sensor. This compact, high-quality camera integrates seamlessly with the Raspberry Pi systems. This module provides a live video feed of the area behind the wheelchair, which enhances the user's spatial situational awareness and safety.
- `Ultrasonic ranging module HC-SR04`: This ultrasonic transmitter, receiver and control circuit module has the ability to detect a maximum range of 4m and a minimum range of 2cm [5]. This serves as a low-cost critical component with excellent range accuracy and stable readings [6] for detecting obstacles behind the wheelchair.
- `Buzzer` This audio signaling devise is commonly used for various technologies and can can be categorized as active or passive. Active buzzers have a built-in oscillating circuit, allowing them to produce consistent tone when electrified. [7]. However, we went with a simple passive buzzer.
- `5-Inch HDMI Touchscreen`: The ELECROW 5-inch Capacitive USB touchscreen is a compact and versatile display designed for Raspberry Pi, can easily be mounted. With a resolution of 800 x 480 pixels, but can be configured to the maximum resolution of 1920 x 1080 pixels, and a refresh rate of 60Hz, this display gives the user real-time visual feedback of the objects and obstacles present behind the wheelchair [9].
- `Breadboard and wires`: All sensors and output devices are interconnected using a solderless breadboard and male-to-male jumper wires from the Sunfounder Starter Kit. The breadboard provides the means to distribute power and organize signal routing. These components allow for easy configuration of the circuit without permanent connections for our prototyping purposes.

C. Software Library Usage

- `Pigpio`: This library we us for the General Purpose Input Outputs (GPIO). Pigpio is one of the most widely used libraries designed for precise control of the Raspberry Pi GPIO pins. Scalable to the simplest LED blinking set up to the most complex sensor networks [10] Pigpio integration with hardware peripherals is ideal for newcomers and make it simple for developers to integrate and control other devices.
- `PiCamera2`: This library, the successor to the legacy PiCamera module, offers more direct access to the Raspberry Pi's camera system. It improves integration with modern hardware and simplifies the development of Python-based applications that utilize real-time video capture [11]. In our implementation, PiCamera2 captures the live rear video feed from the Raspberry Pi Camera Module v2, processed then through the OpenCV and displayed on the touchscreen interface. This real-time feedback is crucial for spacial awareness and user safety.
- `OpenCV-Python`: Open Source Computer Vision Library framework for Python bindings designed to solve computer vision problems [12]. This powerful toolkit is used for manipulation and analyzing visual data like real-time computer vision and image processing. In our backup assistance system, OpenCv-Python renders the live video feed from Raspberry Pi Camera Module v2.
- `GPIO Zero`: This Python library, which is used for our distance sensor, simplifies interaction with the Raspberry Pi's GPIO pins and builds on top of libraries like RPi.GPIO and pigpio. With very little code, this library provides quick connections to components, and easy access to beginner-friendly interfaces [13].

III. Implementation

A. Ultrasonic Sensor

- `Sensor Set up:` To connect the HC-SR04 Ultrasonic sensor to the Raspberry Pi, we utilized the 40-pin GPIO extension board, referencing the GPIO pins using the BCM (Broadcom) pin numbering convention [16]. The sensor was connected with the GPIO pin 23 for the Trigger and GPIO pin 24 for the Echo. Typically, the Echo pin of the HC-SR04 outputs a 5v signal, while the Raspberry Pi's GPIO pins are only 3.3V tolerant [17]. In most cases, a voltage divider would be necessary to step down the Echo pin's output to a safe 3.3V level and protect the Raspberry Pi from damage. The particular model of the HC-SR04 we used features a built-in voltage divider, which internally reduces the Echo pins voltage to the 3.3V safe level, eliminating the need for external resistors [18].
- `Software:` The initial test script for the ultrasonic sensor utilized the RPI.GPIO library, which is widely used for interfacing with raspberry Pi GPIO pins. However, when running the script on the Raspberry Pi5, we encountered a runtime error. The error, "cannot determine peripheral base address", was an indication the RPI.GPI library is not compatible with Raspberry Pi 5's new Hardware architecture [20].
- `Trouble Shooting:` In response to the compatibility issue, we switched to using the GPIO Zero library, which was better suited for newer Raspberry pi models. The GPIO Zero library includes classes for mock pins [21], allowing us to test our code remotely which was extremely beneficial for our project as well as classes for specific devices such as the ***DistanceSensor*** class on top of their basic Input/Output classes [22]. Live testing with GPIO Zero confirmed the effectiveness of our new set up. The sensor returned distance measurements and printed them to the console approximately every .3 seconds. The switch to GPIO Zero resolved the compatibility issues, allowing us to proceed with further integration of the sensor into the overall system.

B. Camera

- `Camera Set Up:` For the camera integration, we used the Raspberry Pi Camera Module V2 and the 15-pin FFC (flat flexible cable), connecting the camera module to the CSI (Camera serial interface) port on the Raspberry Pi board [23]. This connection method allowed us to reserve the GPIO pins for the other essential components, such as the ultrasonic sensor and the buzzer. Additionally, the raspi-config tool was required to enable the camera interface. Overall, the hardware set up was successful and straight forward.
- `Sofware Integration:` Initially, we chose to use the OpenCV (cv2) library to capture and display the live video feed. OpenCV is a robust, open-source computer vision library that supports real-time video capture and advanced processing features like object, which we intended to integrate into our system [24].
- `Trouble Shooting:` However, during live testing, we encountered significant performance issues. While OpenCV offered the functionality we needed, it also consumed considerable system resources. We attempted to allocate additional GPU memory, but still experienced issues including GPU memory errors and failures to initialize the video stream. The sensor process, being more lightweight, often "took over," causing the camera to stall or crash.
  As a solution, we switched to using the native PiCamera2 library, which is optimized for Raspberry Pi hardware. PiCamera2 provided a more stable and efficient interface for capturing video frames [25]. We then were able to pipe video frames from PiCamera2 into OpenCV using NumPy arrays and maintaining our original functionality [26]. This hybrid approach gave us the best of both worlds: efficient camera handling and advanced object detection.

C. Buzzer

- `Buzzer Set Up:` The buzzer we were originally going to use was an active buzzer. However, when testing our code we noticed that it was not working. Therefore, as part of our tests, we decided to try another buzzer. We decided on a passive buzzer that works by modulating the voltage. This means that the more voltage is passed through it, the louder it will be. Through research we found that the cathode (positive connection) should be connected to the GPIO pin 17 and the anode to any ground on the board [27] [28]. The passive buzzer is what ended up in our final solution.
- `Software:` The logic we used for the frequency of the buzz is fairly simple. Probably the simplest of all the peripherals that we used in our final solution. It consisted of a series of comparisons that depended on the distance gathered from the HC-SR04 distance sensor. Specifically, if the distance is greater than 50.0 cm, there is no buzzer activity. If the distance is between 20.0 and 50.0 cm, there is buzzer activity, with the frequency controlled by the import time.sleep(). This was given a value of 0.2 seconds. If it was between 10.0 cm and 20.0 cm, the value of sleep() was given 0.1 seconds, doubling the frequency of buzzes. Finally, if it was within 10.0 cm, the value provided to sleep() was 0.05 seconds. This was the final comparison, since 10.0 cm is about 4 inches and the HC-SR04 sensor only goes to 2cm.
- `Trouble Shooting:` While our solution for the buzzer works, there is one fairly noticeable, or maybe not, flaw in the design. The buzzer is not getting enough voltage to make it a noticeable sound. Although you can hear it, and some would find it more pleasant than a typical tiny buzzer, it is not very loud. Meaning that the solution in a real world scenario would have to involve

putting the buzzer right next to the user's ear. However, some might view this as a benefit.

IV. Results

Our system was centered around a modular, threaded design principle as a response to the resource constraints of the Raspberry Pi. The project aimed to integrate three primary components: the ultrasonic sensor, the camera module, and the buzzer. Each component was connected to the Raspberry Pi through separate interfaces. The ultrasonic sensor was wired via GPIO pins (GPIO 23 for the trigger and GPIO 24 for Echo)., the Raspberry Pi Camera Module V2 was connected using a 15-pin FFC cable to the CSI port and the buzzer was connected to GPIO pin 17.

Our initial prototype relied on sequential, procedural scripting the use of cv2, RPi.GPIO, and basic Python logic. During our live testing, we experienced major resource allocation issues, encountering a "GPU memory allocation" error, among many others. This was specifically due to the demands of OpenCV and camera streaming running in a single thread alongside the ultrasonic sensor. The ultrasonic sensor's read calls dominated the loop. We then attempted to allocate more memory to the camera process. This approach was unsuccessful as well.

The solution was a multi-pronged approach:

- First, we refactored our architecture using Python's threading module. Each component was encapsulated in its own thread class: SensorThread, CameraThread, and AlertThread.
- We then introduced the shared global state structure with synchronization logic to ensure the threads had access to updated data without race conditions. This was a crucial development for the stability of our program.
- Finally, we replaced OpenCV with PiCamera2 for image processing in order to mitigate the GPU memory allocation issues we were having in the camera's capture pipeline. Then, using NumPy arrays, we were still able to feed the frames into cv2 for object detection.

This hybrid approach offered us the advantage of reliable video streaming, efficient memory usage, and the ability to perform frame-by-frame object detection, all from a single Raspberry Pi.

V. Conclusion

In conclusion, our team was successful in providing a responsive prototype, capable of providing its user timely warnings and identifying obstacles in their path. We overcame significant compatibility and performance challenges, including library mismatches and GPU memory limitations. The shift from sequential to threaded design not only resolved critical runtime conflicts and allowed each component to function independently and responsively, but it allowed us to demonstrate what we have learned in CS 370 this semester. This project solidified the importance of modular architecture, resource-aware programming, and selecting tools that align with the platforms capabilities. Through iterative design, live testing, and adaptive troubleshooting, our team was able to designing a low cost, real-time obstacle detection system for wheelchair users. Our compact solution integrated the ultrasonic sensor, live camera feed, and audio alert system, improving mobility safety and spatial awareness. While our current implementation fulfills the core functionality requirements, we've identified several areas for further enhancement:

- Improvements such as tapping into the wheelchairs onboard power supply would eliminate the need for external batteries and improve portability.
- A proper mounting system for the mechanisms display screen would increase usability and integrating dual cameras and ultrasonic sensors (one per wheel) would offer a wider field of view and better spatial awareness for the user.
- Long-term goals include upgrades such as partial autonomy where the wheelchair might intelligently slow or stop in response to obstacles in its path.
- Connectivity features such as remote monitoring and caregiver alerts would be another great addition to the system.

Ultimately, this prototype serves as a step toward developing a more robust, affordable assistive technology that promotes independence and safety for individuals with mobility impairment.

References

[1] Torkia, C., Reid, D., Korner-Bitensky, N., Kairy, D., Rushton, P. W., Demers, L., and Archambault, P. S. (2014). Power wheelchair driving challenges in the community: a users' perspective. Disability and Rehabilitation: Assistive Technology, 10(3), 211–215. https://doi.org/10.3109/17483107.2014.898159

[2] Moulton, Cyrus. (2024). Wheelchair sensors can cost $10,000. Here's how Northeastern engineering students built a better version – for $87. Northeastern Global News. https://news.northeastern.edu/2024/04/18/wheelchair-backup-sensor-system/

[3] Upton, Eben. (2023) Introducing: Rasberry Pi 5!, The Raspberry Pi Foundation. https://www.raspberrypi.com/news/introducing-raspberry-pi-5/

[4] SanDisk. "SanDisk Extreme Pro USB 3.2 Solid State Flash Drive – 128GB" SanDisk Official Store. https://shop.sandisk.com/en-ua/products/usb-flash-drives/sandisk-extreme-pro-usb-3-2?

[5] SparkFun Electronics. "Ultrasonic Ranging Module HC-SR04". https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[6] Handson Technology. "HC-SR04 Ultrasonic Sensor Module User Guide". https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

[7] SunFounder. "Buzzer." SunFounder Component Documentation. https://docs.sunfounder.com/projects/sf-components/en/latest/component_buzzer.html

[8] Cheelcare. Cheelcare AWARE Rear-View Camera System for Power Wheelchairs and Scooters. https://cheelcare.com/products/cheelcare-aware-rear-view-camera-system-for-power-wheelchairs-and-scooters?variant=41393957077174

[9] Elecrow. Elecrow RC050S: 5-Inch 800×480 Capacitive Touch LCD Display for Raspberry Pi. https://www.elecrow.com/5-inch-hdmi-800-x-480-capacitive-touch-lcd-display-for-raspberry-pi-pc-sony-ps4.html

[10] RaspberryPiBox. "Getting Started with pigpio: Installation and Use Cases." https://www.raspberrypibox.com/getting-started-with-pigpio-installation-and-use-cases/

[11] Raspberry Pi Foundation. Picamera2 Documentation. https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf

[12] OpenCV.org. "OpenCV-Python Tutorials." https://docs.opencv.org/4.x/d0/de3/tutorial_py_intro.html

[13] Raspberry Pi Foundation. GPIO zero - A Simple Interface to GPIO devices. https://gpiozero.readthedocs.io/en/stable/
[14] Quantum. Quantum's Wheelchair Backup Camera. https://www.quantumrehab.com/quantum-accessories/quantum-backup-camera.asp
[15] Rear View Safety. Backup Camera System With Rear Sensors. https://www.rearviewsafety.com/amfile/file/download/file/2343/product/2418/
[16] SunFounder. (2021). GPIO Extension Board. In RasPad 3 Documentation. https://docs.sunfounder.com/projects/raspad3/en/latest/appendix/gpio_extension_board.html
[17] Hawkins, M. (2012, December 30). Ultrasonic distance measurement using Python – Part 1. Raspberry Pi Spy. https://www.raspberrypi-spy.co.uk/2012/12/ultrasonic-distance-measurement-using-python-part-1/
[18] Pfahler, L. Connecting an HC-SR04 Ultrasonic Sensor to a Raspberry Pi. https://youtu.be/C02oB1n7rcg?si=JpeYnxeEMNqaC5uk
[19] SunFounder. (2023). 2.2.8 Ultrasonic Sensor Module. In Ultimate Raphael Kit for Raspberry Pi Documentation. https://docs.sunfounder.com/projects/raphael-kit/en/latest/python_pi5/pi5_2.2.8_ultrasonic_sensor_module_python.html
[20] babylonbadders. (2023, December 10). Pi 5 Python RuntimeError: Cannot determine SOC peripheral base address [Online forum post]. Raspberry Pi Forums. https://forums.raspberrypi.com/viewtopic.php?t=361218
[21] GPIO Zero Developers. (n.d.). API – Pins: Changing the pin factory. GPIO Zero Documentation. https://gpiozero.readthedocs.io/en/latest/api_pins.html#changing-pin-factory
[22] GPIO Zero Developers. (n.d.). Migrating from RPi.GPIO. GPIO Zero Documentation. Retrieved April 28, 2025, from https://gpiozero.readthedocs.io/en/latest/migrating_from_rpigpio.html
[23] SunFounder. (n.d.) Camera Module. Sunfounder Documentation. https://docs.sunfounder.com/projects/pidog/en/latest/hardware/cpn_camera.html
[24] Ghosh, A. (Jan 9, 2024). Moving Object Detection with OpenCV using Contour Detection and Background Subtraction. LearnOpenCV. https://learnopencv.com/moving-object-detection-with-opencv/
[25] Raspberry Pi Ltd. (n.d.) PiCamera2 Manual. https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf
[26] Rosebrock, A. (March 30, 2015). Accessing the Raspberry Pi Camera with OpenCV and Python. PyImageSearch .https://pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/
[27] RaspberryPi Ltd. (Dec 5, 20024). Passive Buzzer to Raspberry Pi5. RaspberryPi Forums. https://forums.raspberrypi.com/viewtopic.php?t=380516
[28] Raspberry Pi Pinout Ltd. (n.d). The Raspberry Pi GPIO pinout guide. https://pinout.xyz/