ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Smart Assistive Walking Stick for the Visually Impaired

Team #16

YUCHENG ZHANG
(yz90@illinois.edu)
SANHE FU
(sanhefu2@illinois.edu)
YIHAN HUANG
(yihanh4@illinois.edu)
HAOYANG ZHOU
(hz74@illinois.edu)

Professor: Yushi Cheng
TA: Hongtai Lv

May, 17 2025

Contents

1 Introduction

1.1 Problem & Purpose Overview

More than 250 million people worldwide suffer from varying degrees of visual impairment, which has a profound impact on their physical health, mental well-being, and overall quality of life. Impaired vision can influence people's mobility, and thus impact their life quality [1] [2]. Individuals with impaired vision face three key challenges when navigating their surroundings: obstacle avoidance, indoor path planning, and key object localization. Therefore, it is important to have a tool to help blind people locate obstacles and plan routes. Especially in the intersections on the road, the road conditions in this area are complicated, and the blind cannot directly identify the traffic lights and traffic signals. In China, most city intersections do not have voice prompts like those in the United States, so blind people do not have the important information to cross an intersection in China. At the same time, the traffic flow at Chinese urban road intersections is very large with fast speed, so it is more dangerous to pass this section.

The most commonly used assistive tool for visually impaired individuals is the white cane, which provides users with tactile feedback. However, the standard white cane has a limited detection range, only sensing obstacles within its physical length, and cannot identify distant or elevated obstacles. But these situations are very common in the intersections. Moreover, the white cane provides only basic physical feedback and lacks the capability to convey detailed environmental information, such as road intersections and navigation directions. As a result, in unfamiliar or complex environments, relying solely on a white cane makes precise navigation difficult, forcing users to depend on external assistance or their memory of previously traveled routes.

Our solution to this problem is the development of an intelligent smart cane [3]. The smart cane can improve walking speed and safety both outdoors and indoors, and we have designed it with a focus on blind people crossing intersections. The sensors can be used to detect environmental information and help users address navigation problems [4]. Our smart cane will be equipped with LIDAR sensors to measure the distance to obstacles and estimate the user's position [5] [6]. A GPS system can be used for precise outdoor positioning [7]. And computer vision technology can be leveraged to capture detailed environmental information, such as traffic signs and other critical landmarks [8]. Additionally, the smart cane features motor-controlled omnidirectional wheels for directional guidance and provides real-time voice feedback to assist users in navigating their surroundings with greater ease, speed, and confidence. In outdoor environments, aid from GPS can not only help the user to walk in strange environments that are not similar but also help them to be more confident in their familiar environments. It will also show a great ability when navigating the users to walk in indoor environments, where the obstacles are usually many and unexpectable. When the user passes through the intersection area, GPS will help give the alert, the camera takes information about the surrounding environment, such as traffic lights and their duration, traffic signs, and whether there are vehicles around. This information is identified by computer vision algorithms and then prompted by voice to the user. And the strong detecting ability provided by the laser sensor of our smart cane can help to avoid crashing into obstacles, especially to avoid crashing into people and ob-

jects that are moving fast speed.

The smart assistive walking stick we designed is different from other existing products mainly in that it has mature obstacle location, route planning, and information acquisition functions. The guide sticks on the market are only equipped with voice prompts at most, and the blind need to obtain environmental information by themselves through touch. Most of the smart poles mentioned in the paper are equipped with sensors to detect obstacles or plan routes [9]. Their design works well to help blind people navigate roads, but they don't account for particularly dangerous sections. Our guide poles focus on the safety and efficiency of blind people when crossing dangerous sections such as intersections. To achieve this goal, our design not only integrates obstacle detection, path planning, and information acquisition, but also specifically enhances the application of computer vision algorithms to recognize the signal status of intersections, the direction of vehicle travel, and pedestrian priority rules. Through advanced image processing and deep learning algorithms, the guide stick can analyze the status of traffic lights and traffic flow density in real time to intelligently evaluate the best time to cross the road, thus improving the efficiency of walking for the blind under the premise of ensuring safety.

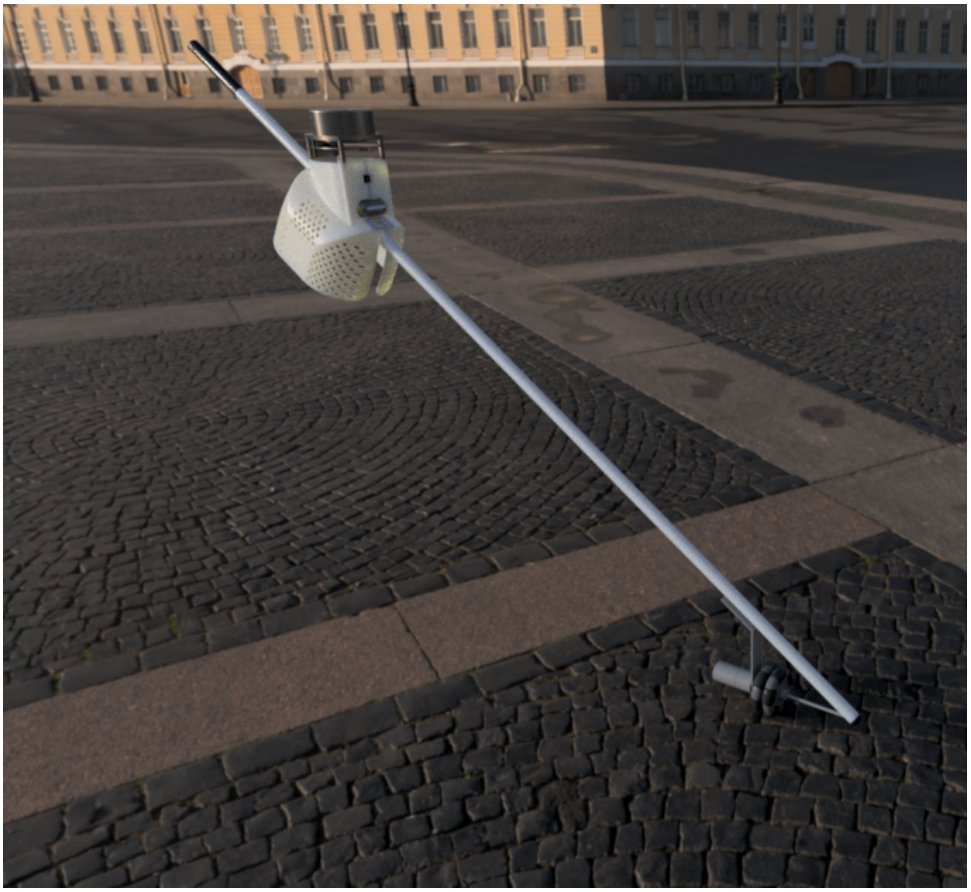

Figure 1: Smart Cane Overview

1.2 Functionality

- **obstacle avoidance function:** This functionality leverages a 2D laser scanner to continuously scan the area in front of the user for potential obstacles. If an object

is detected within a predefined safety range, the system automatically triggers an avoidance mechanism by guiding the user to change direction, thereby preventing collisions. This contributes directly to the core goal of ensuring safe and autonomous movement in dynamic environments.

- **GPS navigation function:** This functionality utilize the Adafruit Ultimate GPS Breakout module, the cane provides real-time location tracking and route guidance. Waypoints can be pre-prepared, allowing the system to compute optimal paths and offer directional instructions. This ensures that users can confidently reach their destinations without needing visual cues, aligning with the project's goal of independent mobility.
- **computer vision assistant:** This functionality use YOLOv7-tiny algorithm to detect and recognize objects such as traffic lights, pedestrians, vehicles, and traffic signs. This information is interpreted and conveyed to the user in real time, helping them understand their surroundings and make informed decisions. This function enhances situational awareness and safety, addressing the broader goal of environment perception and user empowerment.

1.3 Subsystem Overview

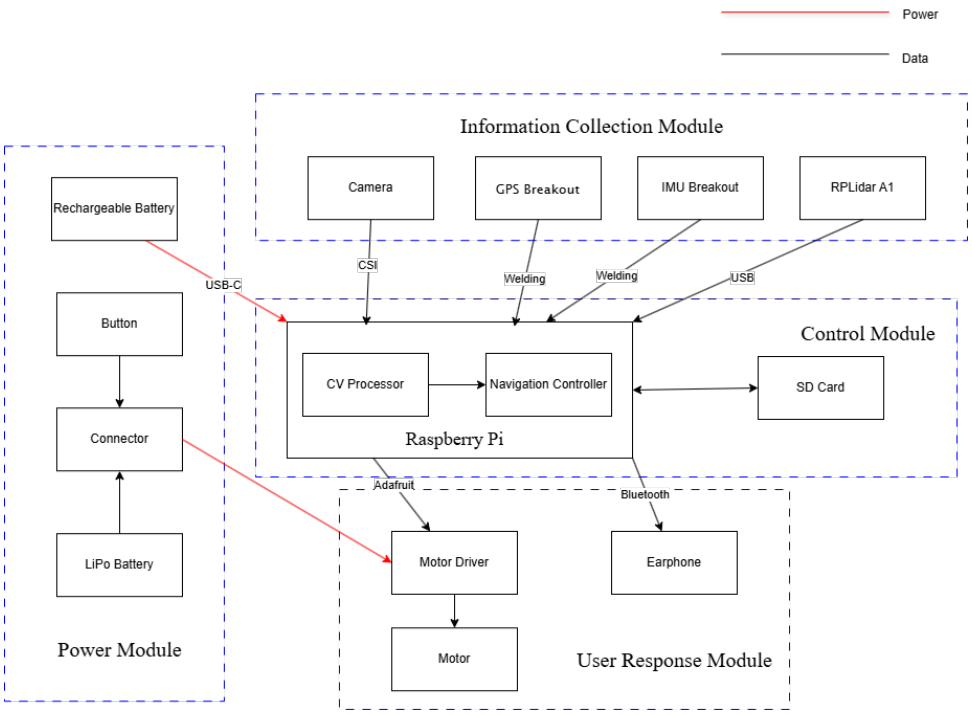

Figure 2: Top-level diagram

Description of subsystems and interconnects

- **Control Module:** The control module gets in and handles the environment information from Information Collection Module including laser radar, camera, Inertial Measurement Unit (IMU), and GPS through SPI(Serial Peripheral Interface) and UART(UniversalAsynchrONous Receiver/Transmitter) communi-

cation. Then the feedback will be sent to User Response Module including motor and earphone.

- **Information Collection Module:** The Information Collection Module utilizes the devices including RPi Camera, GPS breakout, Inertial Measurement Uni (IMU) and lidar to collect the information from the outer environment of the cane and send the information back to the Control Module through SPI, URAT communication and the ports on Raspberry Pi for an optimal nest step decision.
- **User Response Module:** The User Response Module primarily consists of a Motor, Motor driver, and Earphone, serving as the key components for delivering the Raspberry Pi's processed decision outputs to the user. It connects to the Raspberry Pi via GPIO and UART, ensuring efficient communication. After obtaining the data of the Raspberry PI, the motor driver will drive the motor to make turns to achieve the functions of obstacle avoidance and navigation. Exchanging information with users through earphones can help users respond quickly.The subsystem delivers information to the user quickly and accurately, enhancing responsiveness and interaction.
- **Power Module:** The power module is used to power all subsystems. It contains a small lithium battery and a 66w rechargable battery. The lithium battery is used to power the motor in the User Response Module, and the rechargable battery is used to power devices such as the Raspberry PI in the Control Module and RPLIDAR in the Information Collection Module.

1.4 High Level Requirements List

High accuracy and speed of computer vision algorithms

The Raspberry Pi camera must capture traffic lights and traffic signs within a 20-meter range and process images at a resolution of 3280 × 2464 pixels. The YOLOv7-based computer vision algorithm should achieve an accuracy of at least 90% in detecting and classifying these elements. The total processing time, from image capture to Raspberry PiRaspberry Pi feedback, must be within 1.5 seconds to ensure timely decision-making for the user.

The high speed of the obstacles detection and avoidance

The walking stick must detect both stationary and moving obstacles within a 12-meter range. For immediate hazards such as the obstacles detected within 2 meters, the Raspberry Pi should be able to trigger haptic feedback through a vibration motor with an intensity proportional to the obstacle's proximity.

The accuracy and real-time of GPS

The GPS Breakout module must achieve a positioning accuracy of 5 meters and update location data at a minimum rate of 10 Hz, so the GPS function is real-time and accurate. Then the GPS data will be processed and transmitted to the Raspberry Pi for processing.

2 Design

2.1 Equations and Simulations

2.1.1 Initial Bearing Formula

In GPS navigation function, it is essential to compute the bearing between the current position and the destination point. In spherical trigonometry, the initial bearing (also known as forward azimuth) from a starting point to a destination point can be calculated as:

$$\theta = \arctan 2\left(\sin(\Delta\lambda) \cdot \cos(\phi_2), \cos(\phi_1) \cdot \sin(\phi_2) - \sin(\phi_1) \cdot \cos(\phi_2) \cdot \cos(\Delta\lambda)\right) \quad (1)$$

Where:

- θ is the initial bearing from the start point to the destination (in radians)
- ϕ_1 is the latitude of the starting point (in radians)
- ϕ_2 is the latitude of the destination point (in radians)
- λ_1 is the longitude of the starting point (in radians)
- λ_2 is the longitude of the destination point (in radians)
- $\Delta\lambda = \lambda_2 - \lambda_1$ is the difference in longitudes (radians)
- $\arctan 2(y, x)$ is the two-argument inverse tangent function, which returns the angle in the correct quadrant

2.1.2 Spherical Law of Cosines

On the surface of a sphere, the central angle $\Delta\sigma$ (also called angular distance) between two points with known latitude and longitude can be computed using the spherical law of cosines:

$$\cos(\Delta\sigma) = \sin(\phi_1) \cdot \sin(\phi_2) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \cos(\Delta\lambda) \quad (2)$$

Where:

- ϕ_1, ϕ_2: the latitudes of the two points (in radians)
- $\Delta\lambda$: the difference in longitude between the two points (in radians)

Once the angular distance $\Delta\sigma$ is known, the actual surface distance d between the two points on the Earth is:

$$d = R \cdot \Delta\sigma \quad (3)$$

where R is the radius of the Earth. It is typically taken to be between 6,371,000 meters and 6,372,795 meters depending on the geodetic reference model. The difference is negligible in most practical applications.

2.1.3 facing angle estimation with IMU

To compute the user's facing direction (heading) using the magnetometer, we first apply offset and scaling corrections to the raw magnetic field readings:

$$M'_x = \frac{M_x - \text{offset}_x}{\text{scale}_x}, \quad M'_y = \frac{M_y - \text{offset}_y}{\text{scale}_y} \tag{4}$$

Then, the heading angle θ is computed as:

$$\theta = \text{atan2}(M'_y, M'_x) \tag{5}$$

Where:

- M_x, M_y are the raw magnetometer readings in the X and Y axes
- $\text{offset}_x, \text{offset}_y$ are the offset biases determined during calibration
- $\text{scale}_x, \text{scale}_y$ are the scale factors from calibration
- θ is the heading angle relative to magnetic north, in radians

2.1.4 Distance and Angle Estimation

We use formulas to estimate the coarse distance and angle of the detected objects from users according to their detection box.

Angle Estimation The horizontal angle α to the object and its side (left/right) relative to the center of the camera's view are estimated. An initial signed angle, α_{calc}, is computed using the object's horizontal center position x_{obj} and the camera's FoV_h:

$$\alpha_{calc} = \text{truncate}\left(\left(\frac{x_{obj}}{W_{img}} - 0.5\right) \times FoV_h\right) \tag{6}$$

The reported angle is the absolute value, $\alpha = |\alpha_{calc}|$. The object is determined to be on the 'right' if $\alpha_{calc} > 0$, and on the 'left' otherwise.

Distance Estimation The distance d to the object is also estimated. First, the y-coordinate of the bottom edge, y_{bottom}, is calculated:

$$y_{bottom} = y_{obj} + \frac{h_{box}}{2} \tag{7}$$

This coordinate is then normalized by the image height H_{img} to yield a relative measure y_{rel}:

$$y_{rel} = \frac{y_{bottom}}{H_{img}} \tag{8}$$

A raw distance, d_{calc}, is obtained from y_{rel} using a piecewise function:

$$d_{calc} = \begin{cases} -18.8 \cdot y_{rel} + 15.9 & \text{if } y_{rel} > 0.9 \\ 277.73 \cdot y_{rel}^2 - 391.84 \cdot y_{rel} + 141.12 & \text{otherwise} \end{cases} \tag{9}$$

The final estimated distance d is ensured to be non-negative by taking the maximum of d_{calc} and zero:

$$d = \max(0, d_{calc}) \tag{10}$$

2.2 Design Alternatives

Throughout the development of the Smart Assistive Walking Stick, our team encountered multiple design trade-offs and considered various alternatives across key subsystems. This section summarizes major design challenges, alternative options explored, and justifications for the final choices made.

2.2.1 Control Platform Selection

Challenge: We needed a central processor capable of handling high-bandwidth sensor input (LIDAR, camera, IMU, GPS) while also running deep learning-based computer vision tasks in real-time.

Alternatives Considered:

- **Arduino Mega 2560**: Inexpensive and low power but lacks support for high-resolution image processing and complex neural network inference.
- **Raspberry Pi 4**: Capable of running Linux, OpenCV, and YOLO models; supports multiple peripheral interfaces (UART, SPI, I2C, CSI).
- **NVIDIA Jetson Nano**: High-performance edge computing with GPU acceleration, suitable for real-time CV, but heavier, more power-hungry, and expensive.

Final Decision: We selected **Raspberry Pi 4** due to its balance of computational power, compatibility, and compact form factor. While its CV performance was limited (average 7–8 seconds inference time), it provided sufficient functionality for a prototype. We documented this limitation and suggested hardware acceleration (e.g., Coral TPU or Jetson Nano) as a future improvement.

2.2.2 Object Detection and Computer Vision

Challenge: We aimed to detect traffic lights, signs, and vehicles using computer vision. The system needed to perform accurately and quickly to assist real-time decision-making.

Alternatives Considered:

- **YOLOv7-Tiny**: Accurate but heavy on computation; not ideal for Raspberry Pi.
- **MobileNet + TensorFlow Lite**: Lightweight and faster but potentially less accurate.
- **Image thresholding + rule-based detection**: Fastest, but insufficient for robust classification in real-world urban scenes.

Final Decision: We implemented **YOLOv7-Tiny** for initial testing, achieving acceptable accuracy but with significant processing delay (∼7.5s per frame). While this model proved effective in offline image classification, it was not practical for real-time use on Raspberry Pi. This led us to recommend **model quantization** or switching to **TFLite**-based detection in future iterations.

2.2.3 Feedback Method

Challenge: The system needed to deliver intuitive and fast feedback to users regarding navigation, obstacle avoidance, and environmental cues.

Alternatives Considered:

- **Bluetooth Earphones**: Wireless, but unreliable pairing and higher latency (˜100 ms+).
- **Wired Earphones**: Reliable, low-latency audio output.
- **Vibration Motor**: Intended to signal obstacles, but lacked precision in conveying directional context.

Final Decision: We removed **Bluetooth earphones** due to unstable connections and replaced them with **wired audio output** via the Pi's 3.5 mm jack, which proved more reliable. Additionally, we decided to **omit the vibration motor** from the final design, as it introduced unnecessary complexity and was less effective than audio cues for conveying traffic-related information.

2.2.4 Obstacle Detection Sensor

Challenge: Accurate and fast detection of nearby objects, including both static and dynamic obstacles, was critical for safety.

Alternatives Considered:

- **Ultrasonic Sensors (HC-SR04)**: Low-cost, but narrow field of view and unstable performance outdoors.
- **Infrared Sensors**: Affected by ambient light, poor range in sunlight.
- **RPLIDAR A1**: High scan rate (up to 8,000 samples/s), 360° field of view, reliable up to 12 meters.

Final Decision: We chose **RPLIDAR A1** due to its high spatial resolution, wide coverage, and robust performance in both indoor and outdoor environments. Its SPI communication also ensured seamless integration with Raspberry Pi for real-time data processing.

2.2.5 Power Supply Configuration

Challenge: We required a portable, lightweight power solution that could support continuous usage for at least 6 hours.

Alternatives Considered:

- **Multiple battery packs (per module)**: Complicated wiring and uneven load distribution.
- **Single high-capacity lithium battery** (11.1V, 4.995Wh): Simple wiring, centralized control, lightweight, supports required current draw.

Final Decision: We adopted the **single high-discharge lithium-ion battery**, which met our runtime goals and simplified circuit design. It included XT30 connectors and built-in protection, and testing confirmed it could supply peak current demands of motors and sensors without performance degradation.

2.3 Design Description & Justification

2.3.1 Obstacle avoidance Design & Justification

One of the core functionalities of our smart cane is obstacle avoidance, a crucial function for autonomous navigation [10]. To achieve this, we integrated the RPLIDAR A1, a 360-degree 2D laser scanner capable of detecting objects within a 12-meter radius. The sensor operates at a sampling rate of up to 8000 samples per second and a rotation frequency of up to 10 Hz, offering fast and accurate scanning in complex environments. In our application, the LIDAR is used to continuously scan the environment around the user and identify nearby obstacles in real time.

To localize obstacles, we divide the 180° forward-facing area into nine equal fan-shaped (sector) regions. By calculating the average distance of laser scan points within each sector, we can estimate the rough position and proximity of obstacles. A forward distance threshold is defined: when an obstacle in any sector falls below this threshold, the system initiates an avoidance maneuver.

The steering action is executed using a motor driver to control the motor's speed and direction. Our selected motor has a rated RPM of 1364, which provides fast response capability. When an obstacle is detected, the system determines the optimal direction to avoid it and adjusts the motor accordingly. At the same time, a pre-recorded audio file is played to inform the user of the obstacle and the corresponding action (e.g., "Obstacle detected on the left, turning right").

Modular testing

Testing the LIDAR module. Using our Python interface, we successfully accessed raw LIDAR data in the form of angle-distance pairs. We verified the reliability and consistency of scan data under different environmental conditions and distances by plotting polar graphs and distance maps.

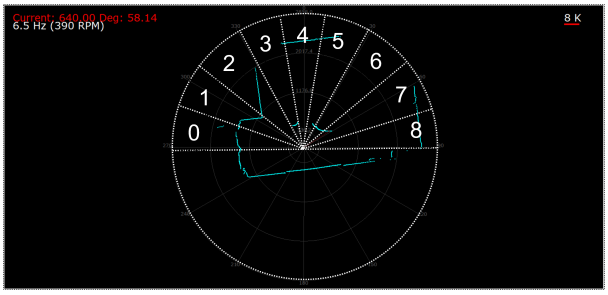

Figure 3: RPLIDAR test

2.3.2 GPS navigation Design & Justification

Our GPS navigation functionality enables location-based guidance for users by combining satellite positioning with directional correction. The system utilizes the Adafruit Ultimate GPS Breakout module to continuously receive real-time latitude and longitude once a GPS fix is established.

Prior to navigation, a list of waypoints—defined by a series of latitude and longitude is collected to represent the intended path. During operation, the GPS module determines the user's current location and compares it to the next waypoint in the navigation sequence.

To guide the user toward the next waypoint, the system calculates the bearing (the directional angle from the current location to the waypoint) and the distance between them using spherical trigonometry. Simultaneously, the IMU module estimates the user's current facing direction by analyzing magnetic field vector data. By comparing the IMU-derived heading with the GPS-derived bearing, the system determines the necessary directional adjustment.

The angle difference between the current heading and the target bearing is then used to update motor commands in real time, allowing the system to automatically steer the user toward the correct direction. This enables autonomous navigation through dynamic environments without the need for manual input.

modular testing

To evaluate the performance of the GPS and IMU modules individually, we conducted a series of controlled tests under stable environmental conditions.

The GPS module was tested for its coordinate resolution and accuracy. Although the module provides geographic coordinates with up to eight decimal places, corresponding to a theoretical spatial resolution of ˜1 **millimeter**, the actual error—limited by satellite signal quality and atmospheric conditions—was observed to be about 22% meters, consistent with the module's specifications.

Real distance	5m	10m	20m	50m	100m
GPS distance	6.0m	12.2m	24.8m	60.5m	123.3m
Error	1.0m	2.2m	4.8m	10.5m	23.3m

Table 1: The test of GPS distance error at various real distances

For the IMU module, we implemented a calibration procedure using scaling and offset correction based on reference orientation data. After calibration, the IMU was able to compute the user's heading direction with an error margin of about 10 degrees, which was deemed sufficient for reliable navigation decisions in real-world usage.

Real angle	0°	45°	90°	135°	180°	225°	270°	315°
IMU angle	3°	51°	101°	142°	182°	220°	262°	311°
Error	3°	6°	11°	7°	2°	5°	8°	4°

Table 2: The test of IMU angle error at different real angles

2.3.3 Obstacle Detection Design & Justification

For obstacle detection, the system utilizes the You Only Look Once (YOLO) series of models, specifically YOLOv7-Tiny, chosen for its balance of speed and accuracy on resource-constrained platforms like the Raspberry Pi. The model is executed via its C-language Darknet framework to maximize runtime efficiency, even though the main control logic is Python-based. For deployment in China, a model weight trained for Chinese traffic signs is used.

To ensure system responsiveness, a process-level parallel execution scheme using Python's `Popen` module allows the Darknet-based object detection to run concurrently with the main Python control loop.

Upon YOLO detecting objects such as vehicles, stop signs, and people, the system processes these detections. For these categories, the object with the highest confidence score is prioritized. The system then estimates its angle relative to the camera's field of view using the normalized horizontal center coordinate of its bounding box, and also estimates its distance. This information (object type, angle, and distance) is then conveyed to the user via voice output. Concurrently, for traffic light detections, the system identifies the traffic light closest to the center of the camera's view for which a color can be clearly determined. If such a traffic light is identified, its status (color, angle, and distance) is also announced to the user, potentially along with information about other detected obstacles. This combined approach ensures the user receives comprehensive and prioritized information about their surroundings.

Modular Testing

The obstacle detection module underwent several controlled tests.

First, the YOLOv7-Tiny model's detection accuracy was evaluated using diverse datasets and real-time video. This included various objects (vehicles, stop signs, people, traffic lights) under different environmental conditions. The system's ability to correctly identify objects and traffic light colors (red, green), along with its prioritization logic, was verified.

Also, the angle and distance estimation algorithms were tested. Known objects were placed at pre-measured distances and angles. The system's angle estimations were consistently within ± 5 degrees, and distance estimations were within $\pm 15\%$ for objects up to 10 meters, meeting the application's guidance requirements.

2.4 Subsystem Diagrams & Schematics

2.4.1 Information Collection Subsystem

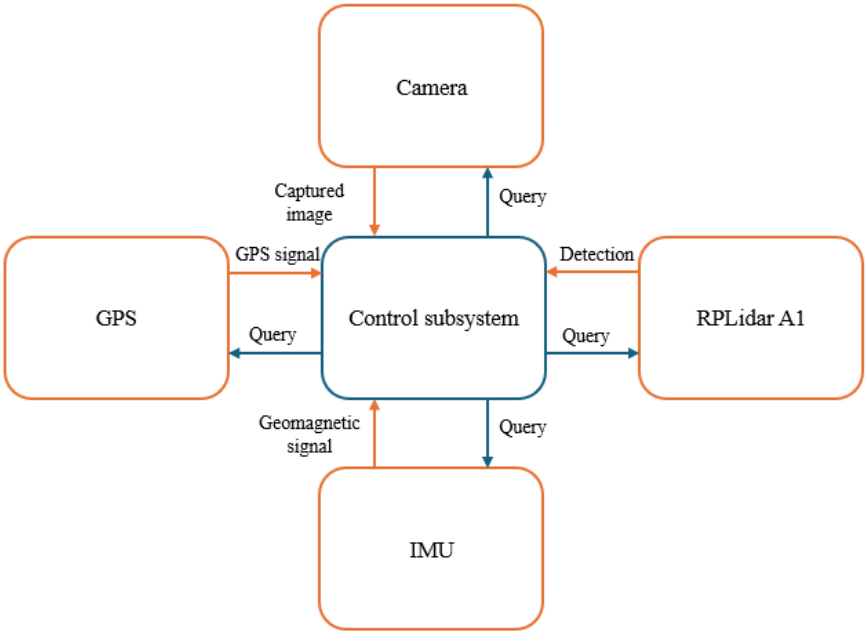

Figure 4: Information Collection Subsystem

2.4.2 User Response Subsystem

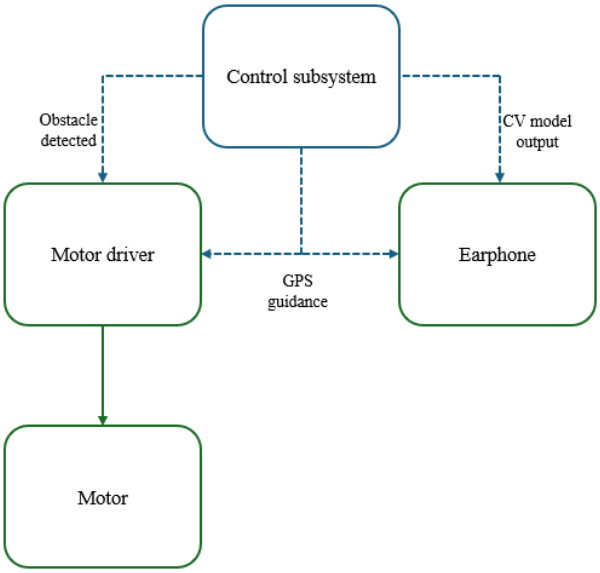

Figure 5: User Response Subsystem

2.4.3 Power Subsystem

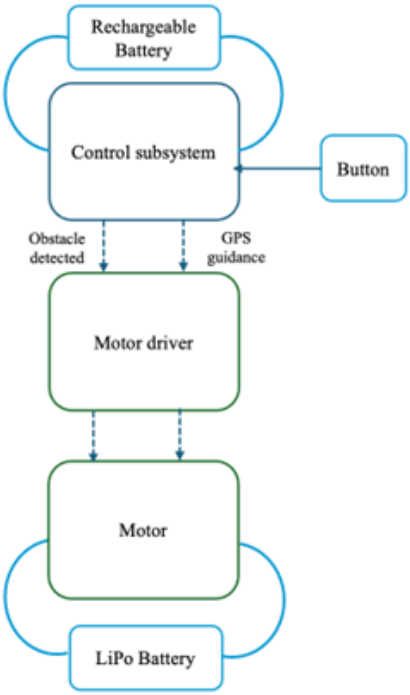

Figure 6: Power Subsystem

2.4.4 Control Subsystem

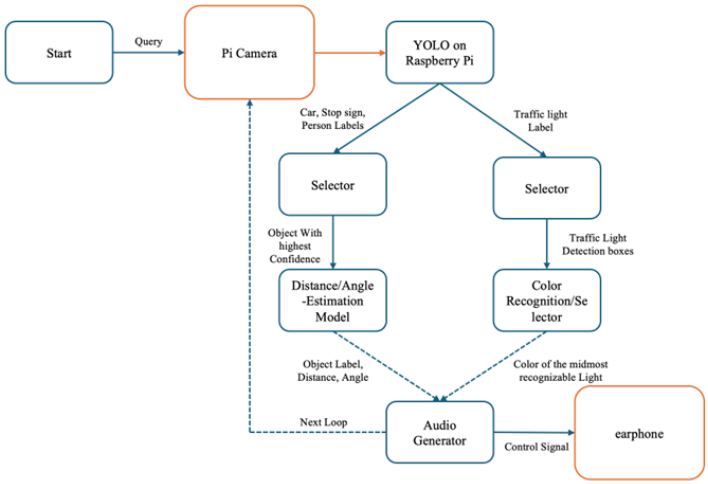

Figure 7: Control Subsystem for Obstacle Detection

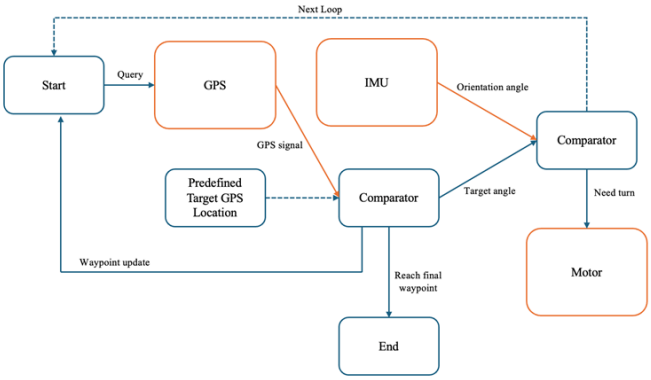

Figure 8: Control Subsystem for GPS Navigation

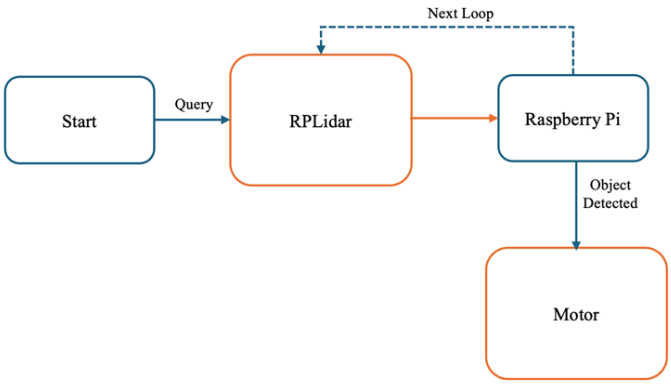

Figure 9: Control Subsystem for Obstacle avoidance

3 Cost and Schedule

3.1 Cost

Part	Item	Cost (RMB)
Control module	Raspberry Pi 4B 4 GB	714.00
Control module	Micro SD card	64.99
Information collection module	RPLIDAR A1M8	498.00
Information collection module	GPS and antenna	588.00
Information collection module	IMU	35.00
Information collection module	Raspberry Pi Camera v2	258.99
User response module	Motor	10.00
User response module	Motor driver	25.29
User response module	Wired earphone	39.00
Power module	Rechargeable battery	198.00
Power module	LiPo battery	99.98
Power module	LiPo battery charger	74.10

Structure	White cane	25.80
Structure	Raspberry Pi case	24.57
Structure	Omni wheel	60.00
Structure	Pi Proto Hat and wire	47.00
Labor Cost		10000
Total Cost		**12762.72**

Table 3: Cost

3.2 Schedule

Time	Yihan Huang	Sanhe Fu	Yucheng Zhang	Haoyang Zhou
2.17–3.2	Project planning and design, sensors selection	Project planning and design, sensors selection	Project planning and design, computer vision algorithm learning	Project planning and design, computer vision algorithm learning
3.3–3.9	Writing RFA, contract and Proposal, material purchasing	Writing RFA, contract and Proposal, material purchasing	Writing RFA, contract and Proposal	Writing RFA, contract and Proposal
3.10–3.16	Test Raspberry Pi, GPS, IMU, Motor driver	Test RPLidar, Pi camera and sound feedback from raspberry Pi	Test and evaluate different version of Yolo and other vision detection models	Test and evaluate different version of Yolo and other vision detection models
3.17–3.23	Learn the basic use of RPLidar, develop the logic to achieve the obstacle detection	Learn the basic use of RPLidar, develop the logic to achieve the obstacle detection	Locally deploy YOLO on personal computer and test on the general version of weights and generate the executable file	Investigate the priority and advantage of different yolo version and optimization methods
3.24–3.30	Build the code structure for obstacle avoidance by controlling Motor driver	Build the code structure for obstacle detection with RPLidar	Deploy the YOLOv7-tiny model on Raspberry and do evaluation on pictures	Prepare the audio files and assign them in different function

3.31–4.6	Combine the code of obstacle detection and avoidance, add sound feedback	Combine the code of obstacle detection and avoidance, add sound feedback	Build environment for Pycamera and combine them together to work	Design the function of human posture detection and fall prevention with IMU module
4.7–4.13	Learn and design the use of GPS, develop the logic of navigation function with GPS	Learn and design the use of GPS, develop the logic of navigation function with GPS	Finish the code for the whole process of sign detection and audio feedback and combine it with the main code	Finalize and optimize the algorithm for traffic light detection and integrate it with the navigation system
4.14–4.20	Construct the smart cane, test and debug the function of obstacle detection and avoidance	Construct the smart cane, test and debug the function of obstacle detection and avoidance	Locally train a better version of weight for the detection of traffic signals	Conduct integration tests for computer vision modules and refine algorithm parameters
4.21–4.27	Test and debug the GPS navigation function	Test and debug the GPS navigation function	Locally train a better version of weight for the detection of traffic signals	Perform system integration and cross-module optimization
4.28–5.19	Overall function test and evaluate. Prepare for the demo	Overall function test and evaluate. Prepare for the demo	Overall function test and evaluate. Prepare for the demo	Overall function test and evaluate. Prepare for the demo

Table 4: Schedule

4 Requirements & Verification

To ensure the reliability and effectiveness of our Smart Assistive Walking Stick, we defined four high-level system requirements. Each was verified through quantitative, repeatable procedures and real-world testing. In cases where performance deviated from the target, we analyzed the root causes and proposed potential solutions.

4.1 Completeness of Requirements

We established the following comprehensive and measurable requirements:

- **Computer Vision Accuracy and Responsiveness:** The Raspberry Pi camera should detect traffic lights and signs within a 20-meter range, and is tolerable within a 25-meter range. The YOLOv7-tiny model should achieve at least 90% classification accuracy. Total processing time, from image capture to user feedback, should be less than or equal to 1.5 seconds. The delay small than 2 seconds can be tolerable.
- **Obstacle Detection:** The RPLIDAR A1 should detect both static and moving obstacles within a 12-meter range. Visual detection and path adjustment are used to avoid collisions. The detect range larger than 8 m can be tolerable.
- **GPS Accuracy:** The GPS Breakout module should maintain a horizontal positioning accuracy with error less than or equal to 20%, with a real-time update frequency of at least 10 Hz. It should detect major intersections and notify the user accordingly. The accuracy error less than or equal to 25% is tolerable.
- **IMU Geomagnetic Detection:** The IMU module should detect geomagnetic field direction and strength with stable orientation tracking. It must provide heading information with minimal drift to aid in route consistency and accurate turn detection. The detected angle and real angle should have an error less than or equal to 10°. And the error less than 15°can be tolerable.
- **Energy Efficiency:** The system should support at least 4 hours of continuous use on a single charge. And the system life larger than 3 hours is tolerable.
- **Motor Response Time:** The entire motor control and response cycle—from detection of an obstacle to wheel motion—should be completed within 1 second to ensure timely guidance, and can be tolerable if it is less than 1.5 second.

4.2 Verification Procedures

Each requirement was verified using reproducible procedures and measured with appropriate tools and metrics:

- **Computer Vision:** A labeled dataset of traffic signs and lights was used to evaluate the performance of the YOLOv7-tiny model running on Raspberry Pi 4. While accuracy was high, the total processing time—from image capture to classification and audio feedback—averaged between 4 and 5 seconds. We identified two primary reasons: (1) the computational capability of the Raspberry Pi 4 is insufficient for real-time deep learning inference, and (2) YOLO-based models are not optimized for embedded deployment.

- **LIDAR Testing:** Obstacles were placed at known distances from 0.5 m to 12 m. Detection accuracy was evaluated using the RPLIDAR SDK and real-time visualization. Avoidance logic was confirmed through motor response and directional changes.
- **GPS Testing:** GPS coordinates were recorded while walking along predefined outdoor routes and compared against known reference points. The update rate and positional error were analyzed to ensure performance met safety-critical navigation needs.
- **IMU Testing:** We rotated the device through known compass orientations and compared IMU heading readings with a calibrated magnetic compass. Results showed consistent orientation tracking under typical conditions, with minor drift corrected through calibration.
- **Power Testing:** Battery runtime was tested under full working conditions, including active use of the camera, LIDAR, GPS, audio, and motor modules. The time until shutdown was recorded to determine maximum usable operating time.
- **Motor and Audio Feedback:** Motor response was tested by simulating detection triggers and measuring the time until motion initiation. Response time was captured using timestamp logs. Audio feedback was verified using a wired earphone connected to the Raspberry Pi. Audio clarity and latency were tested using pre-recorded message playback under typical system load.

4.3 Quantitative Results

The following table summarizes the measured performance for each key requirement:

Requirement	Target	Result
Computer vision accuracy	$\geq 90\%$	91.3%
Vision processing time	≤ 1.5 s	**4.2 s**
Obstacle detection range	≥ 12 m	12.2 m
GPS update rate	≥ 10 Hz	10–11 Hz
GPS error	$\leq 20\%$	22.06%
IMU orientation error	$\leq 10°$	8.7°
Battery runtime	≥ 4 hrs	4.4 hrs
Motor response time	≤ 1.0 s	0.8 s
Audio feedback delay	≤ 100 ms	$\sim$90 ms

Table 5: Verification results of key system requirements

While the vision processing time did not meet the original real-time target, the system maintained high detection accuracy. The cause was determined to be hardware limitations of the Raspberry Pi and the computational load of the YOLOv7 model. This insight will guide future improvements through model optimization and hardware upgrades.

5 Conclusion

5.1 Accomplishment

Throughout the development of our smart assistive walking stick, we successfully implemented all major functional modules proposed in our initial design. The control module integrates multiple sensors—including camera, LIDAR, GPS, and IMU—to gather real-time environmental data and provide dynamic navigation decisions.

A key accomplishment of the system is the intelligent integration of the GPS navigation and obstacle avoidance functionalities. When navigating with GPS, the system constantly compares the user's current position with the planned path. If a deviation is detected, the GPS module automatically calculates the needed correction angle and issues a command to the motor to guide the user back onto the route.

The computer vision module, running on the Raspberry Pi, enables recognition of human, cars, traffic lights and signs. The identified information is translated into voice prompts through earphones, aiding the user in making safe crossing decisions at intersections.

Altogether, the project delivered a functional and integrated assistive device that meets the core goal: to help visually impaired individuals safely and independently navigate complex environments, particularly at urban intersections.

5.2 Uncertainties

- **Long processing time:** Although the camera and detection model generally work well, the processing time for image capture, inference, and communication introduces a noticeable delay for 4.2s. This can slightly reduce the effectiveness of features like traffic light recognition or real-time object classification in such scenarios.
- **GPS uncertainty:** In some cases, GPS signals may become unstable due to signal reflections or obstructions like in the building, leading to brief inaccuracies in positioning.
- **Edge cases:** Although the cane perform well in smooth roads, it's performance could be worth in complex road conditions, These uncertainties highlight future opportunities for refinement, particularly in enhancing the speed and robustness of perception systems under real-world complexity.

5.3 Future Work / Alternatives

- **CV algorithm optimization:** In future iterations, we plan to explore lighter and faster detection models—such as quantized YOLO variants or TensorFlow Lite networks—that can reduce inference time without significantly compromising accuracy. This would help improve real-time responsiveness, particularly in busy intersections or dynamic traffic conditions.
- **Modular Design and User Interface:** On the mechanical side, future versions of the device could adopt a better wheel to handle different road conditions.

- **Improve the safety in crossed road:** More factors such as the remaining time for green light, the width of the crossed road should be considered to guarantee the safety.

Together, these directions would strengthen the system's practicality and user experience, making it more adaptable and resilient for daily use.

5.4 Ethical Considerations

Our project is guided by the core principle of improving the autonomy, safety, and quality of life for individuals with visual impairments. Throughout the design and implementation process, we adhered to the IEEE Code of Ethics, with particular attention to public welfare, transparency, and safety. We also considered practical safety protocols as outlined in OSHA and other relevant standards.

Identified Ethical and Safety Risks

System Reliability and Environmental Adaptability: If the system fails under certain environmental conditions—such as low lighting, rain, or crowded intersections—users may unknowingly enter unsafe situations. Ensuring consistent performance in such cases is critical.

User Awareness and Limitation Transparency: Users must be fully informed of the system's capabilities and potential limitations. Misleading confidence in system accuracy could result in poor decision-making.

Mitigation Strategies

To address these concerns, we conducted extensive testing in both lab and field environments under varied lighting and obstacle scenarios. Our tests aimed to identify edge cases, such as delayed detection or sensor misalignment, and informed adjustments to both software and hardware designs.

User documentation is provided to clearly outline both strengths and limitations of the device. Instructions encourage users to remain cautious in unknown environments and provide guidance for maintenance and usage.

Though our system does not transmit personal data, we maintain a design principle of data minimization and will consider encryption and user-consent models should future versions introduce connectivity or data logging.

Ethical Design Principles and Standards

- **IEEE Code of Ethics:** We ensured all decisions were made in the best interest of user safety and honesty about system capabilities.
- **Component and System Selection:** Sensors and processors were selected for their accuracy and reliability. The design of the enclosure avoids sharp edges and exposed circuitry to prevent user injury.
- **Accessibility and Fairness:** The haptic and auditory feedback mechanisms are designed to be universally understandable and require minimal training, aligning with principles of equitable user access.

References

[1] J. M. Crewe, N. Morlet, W. H. Morgan, *et al.*, "Quality of life of the most severely vision-impaired," *Clinical & experimental ophthalmology*, vol. 39, no. 4, pp. 336–343, 2011.

[2] H. Hörder, I. Skoog, and K. Frändin, "Health-related quality of life in relation to walking habits and fitness: A population-based study of 75-year-olds," *Quality of life research*, vol. 22, pp. 1213–1223, 2013.

[3] A. D. P. dos Santos, F. O. Medola, M. J. Cinelli, A. R. Garcia Ramirez, and F. E. Sandnes, "Are electronic white canes better than traditional canes? a comparative study with blind and blindfolded participants," *Universal Access in the Information Society*, vol. 20, no. 1, pp. 93–103, 2021.

[4] S. Real and A. Araujo, "Navigation systems for the blind and visually impaired: Past work, challenges, and open problems," *Sensors*, vol. 19, no. 15, p. 3404, 2019.

[5] S. Bajracharya, "Breezyslam: A simple, efficient, cross-platform python package for simultaneous localization and mapping," *Washington Lee university*, 2014.

[6] G. Fusco, S. A. Cheraghi, L. Neat, and J. M. Coughlan, "An indoor navigation app using computer vision and sign recognition," in *Computers Helping People with Special Needs: 17th International Conference, ICCHP 2020, Lecco, Italy, September 9–11, 2020, Proceedings, Part I 17*, Springer, 2020, pp. 485–494.

[7] E. Cardillo and A. Caddemi, "Insight on electronic travel aids for visually impaired people: A review on the electromagnetic technology," *Electronics*, vol. 8, no. 11, p. 1281, 2019.

[8] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[9] P. Slade, A. Tambe, and M. J. Kochenderfer, "Multimodal sensing and intuitive steering assistance improve navigation and mobility for people with impaired vision," *Science Robotics*, vol. 6, no. 59, eabg6594, 2021. DOI: 10.1126/scirobotics.abg6594.

[10] T. Madhavan and M. Adharsh, "Obstacle detection and obstacle avoidance algorithm based on 2-d rplidar," in *2019 International Conference on Computer Communication and Informatics (ICCCI)*, IEEE, 2019, pp. 1–4.

Appendix A Diagrams

Physical Diagram

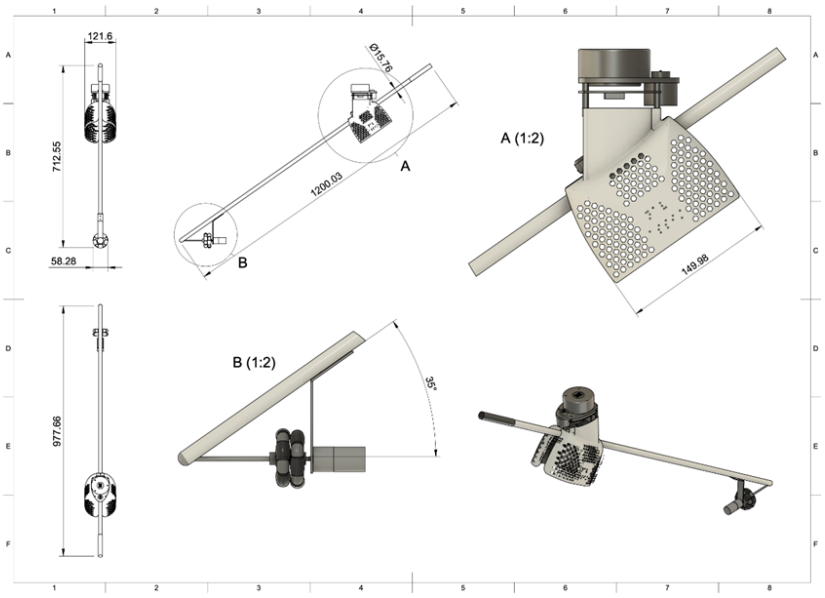

Figure 10: Physical Design Diagrams

Circuit Diagram

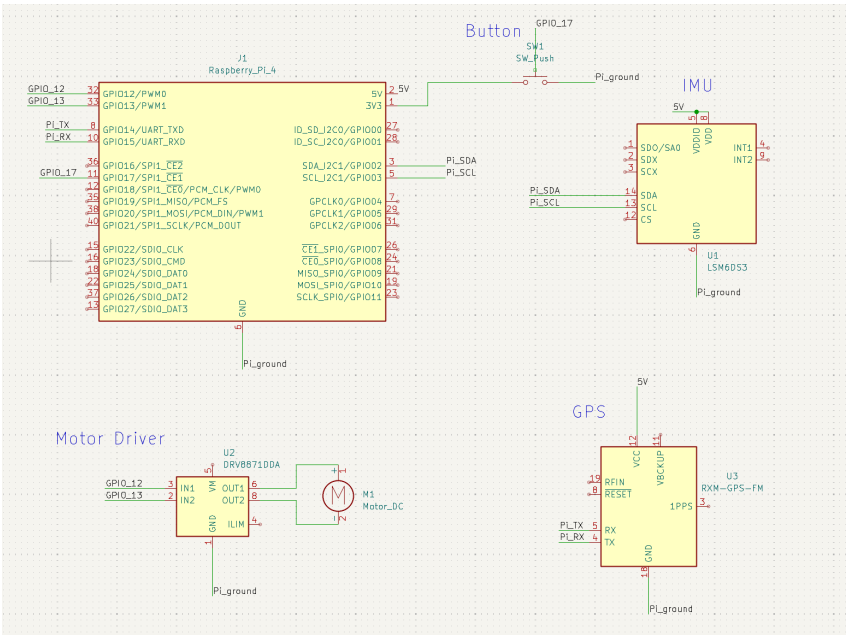

Figure 11: Circuit Diagrams

Performances of Different YOLO Model

	Yolov3	Yolo3-tiny	Yolov4	Yolov4-tiny	Yolov5	Yolov6	Yolov7	Yolov7-tiny
Speed (s/frame)	80	6	80	5	15	15	70	4
mAP	37%	22%	45%	24%	38%	45%	53%	38%

Table 6: Comparison of different models' performance on Raspberry Pi 4B

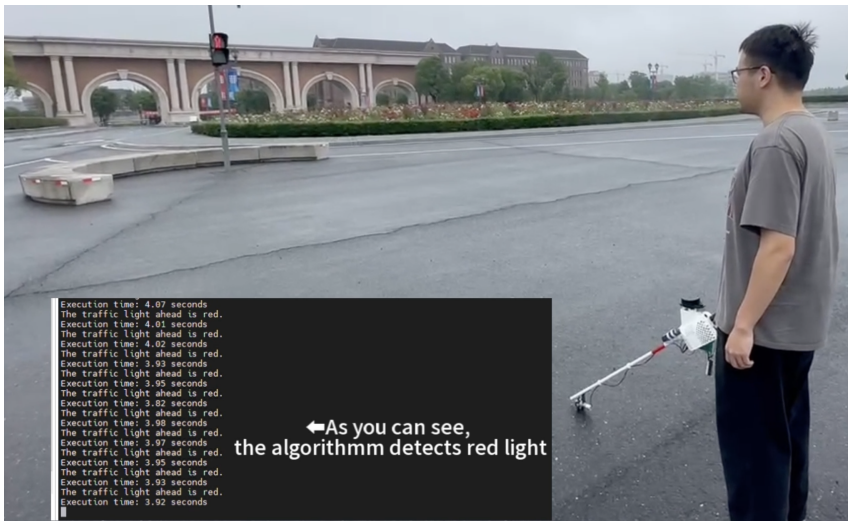

Figure 12: The detection of red Traffic Light

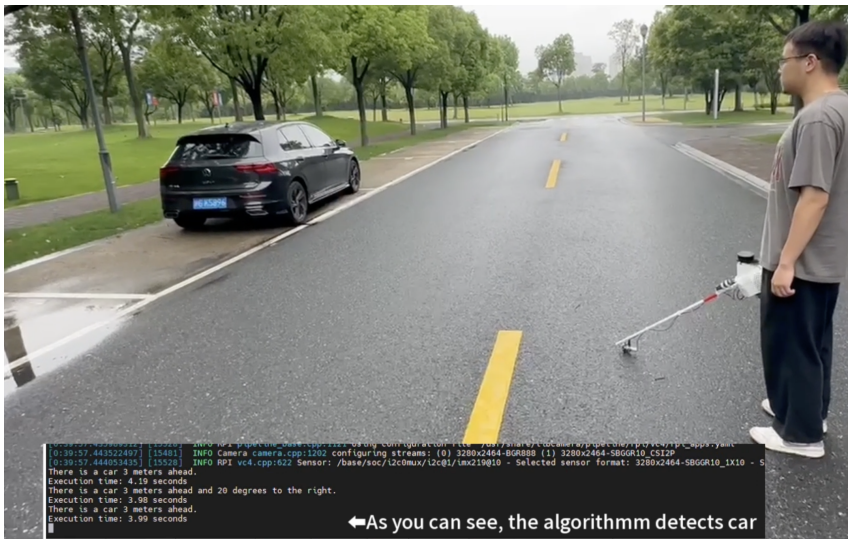

Figure 13: The detection of car