Final Project Report

PilotClean: An ultimate FPGA dual mode cleaning robot

Team 23:
112062129 Song-Ze, Yu
112062232 Yun-Zhong, Lai

Table of Contents

1. Introduction

1.1 Abstraction

PilotClean is a dual-mode cleaning robot which integrates manual control with autonomous cleaning into a single FPGA-based solution. Its dual-mode functionality, supported by mobile app control and advanced hardware, ensures efficiency and user convenience.

▲ Figure.1-1 Prototype of our sweeping robot design

1.2 Motivation

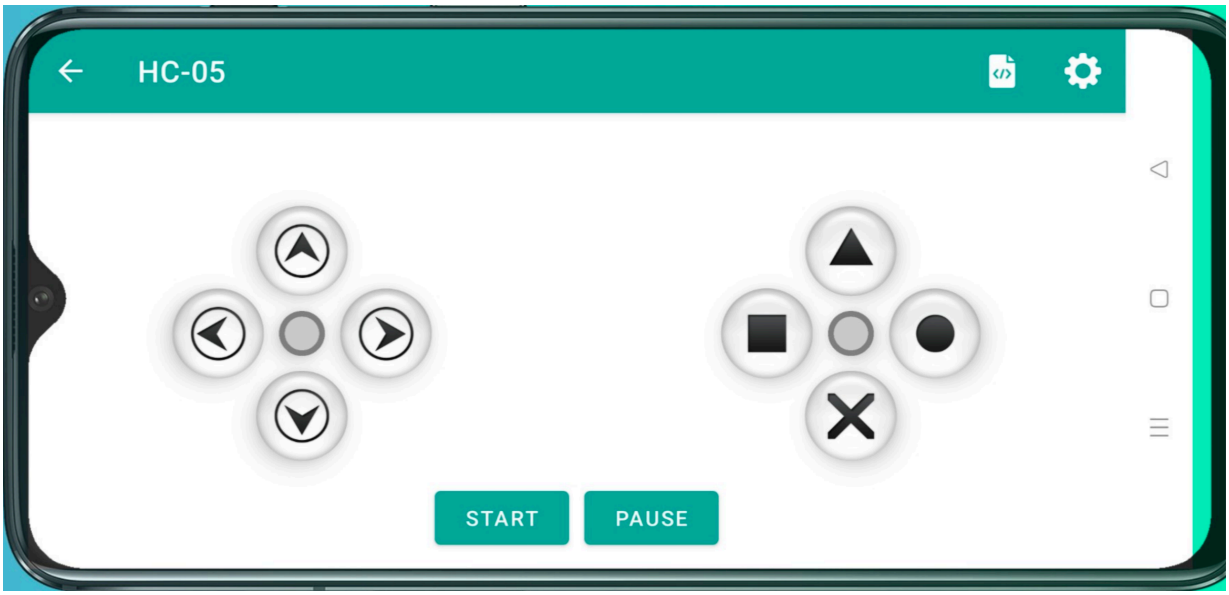
Though the usage of sweeping robot is thriving, we still observe some significant limitations of current sweeping robots: (1) While they are already feature-rich, they lack remote-control capabilities. (2) Even with advanced algorithms or techniques, they can't efficiently handle sudden, localized messes compared to manual control.

Due to the reason above, our vision is to design a dual-mode sweeping robot that combines the flexibility of manual control with the efficiency of automatic cleaning, delivering a more effective solution.

1.3 Features

1. App Control

PilotClean can be controlled via a smartphone app. We highly recommend *Arduino Bluetooth Controller* as it is not only free but also includes a fully functional gamepad that allows you to define the ASCII code for each button.



▲ Figure.1-2 Gamepad screenshot of Arduino Bluetooth Controller

- **Movement controls**: forward (`), backward (p), left (x), and right (|).
- **Power control (on/off)**: triangle (default)
- **Suction (fan) control (on/off):** square (default)
- **Mode selection (remote/random)**: X (default)
- **Speed level selection (LOW, MID, HIGH):** circle (default)

2. Dual-Modes Operation

2.1 Bluetooth remote control mode

PilotClean can be fully controlled by remote app with the connection between smartphone and HC-06 on our FPGA. We implement a UART protocol to complete the communication between the bluetooth module (HC-06) and our FPGA. The detail will be mentioned later on in our report.

2.2 Random mode

PilotClean will generate random walk generally by utilized LFSR. The detail will be mentioned later on in our report.

3. Obstacle Detection and Avoidance

PilotClean uses two infrared sensors which places in the front and back side to detect obstacles in its path when random walking. By doing so, the robot can avoid collide or stuck at a certain position.

2. Overall Design Details

2.1 Dust Collection System

We use an 8x8 12V 0.6A square fan, selected for its effective suction performance after multiple experiments. Power is supplied through an L298N module with three 18650 lithium-ion batteries connected in series, providing an nearly 12V output.

To enhance airflow, the middle filter paper layer was removed, while an outer masking cover was retained as the picture shown. By doing so, (1) it protects our internal components, such as the FPGA board and Dupont wires, from dust intrusion; (2) it also guarantees enough suction power for collecting the dusts.

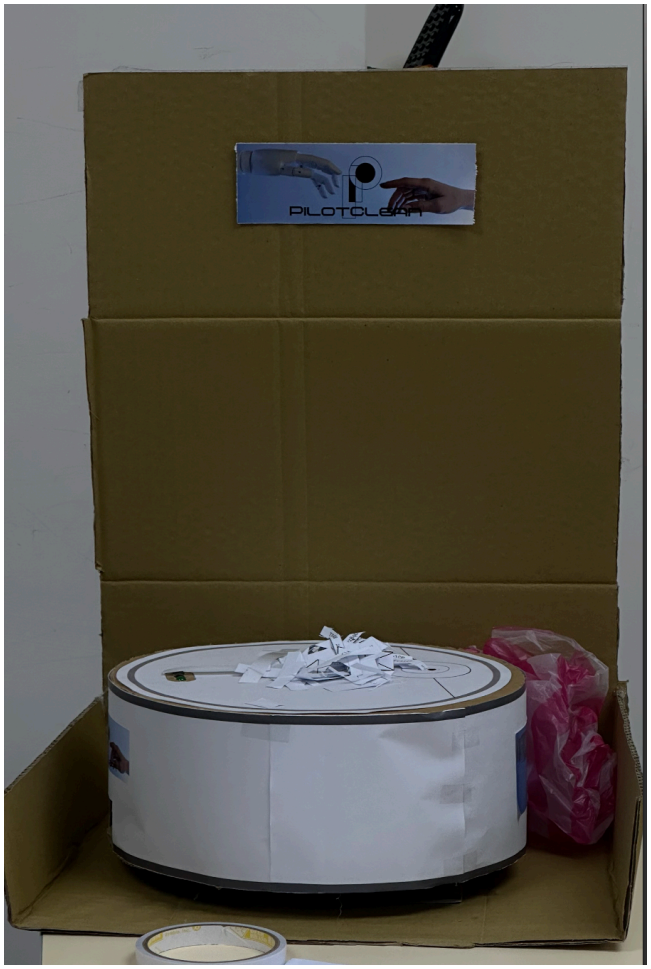
▲ Figure.2-1 Our flat design of our dust collection system

2.2 Trash Disposal Method

The initial design of the trash disposal system involved a vertical airflow mechanism that directed debris into a replaceable bag. However, this approach posed several challenges, including (1) the risk of dust intrusion into the intricate Dupont wiring and FPGA board, making maintenance difficult. Additionally, (2) disassembling the robot for trash replacement after every use proved impractical. Furthermore, (3) the reliance on a 12V power supply was found to be slightly underpowered for this design after many experiments.

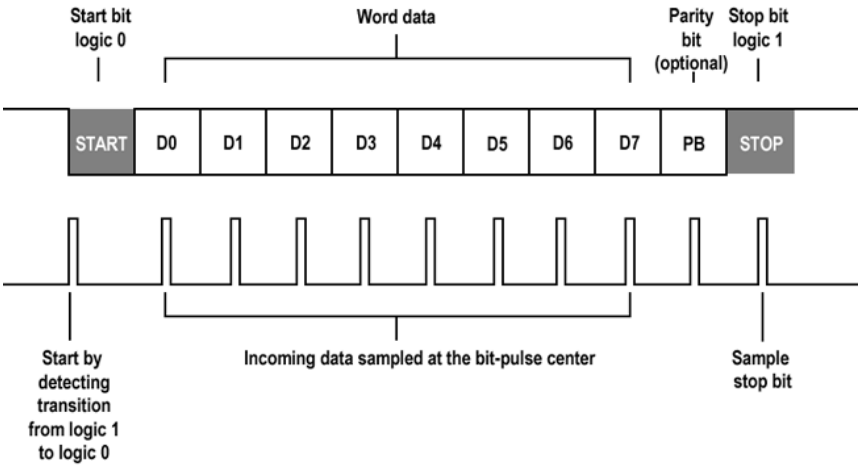
To address these issues, the final design incorporates a return-to-home system, where the robot, controlled via remote control, deposits collected trash at its base station as the picture shown.

This solution simplifies maintenance while protecting internal components and ensuring efficient trash management.



▲ Figure.2-2 Our base station design

2.3 Bluetooth and UART Protocol

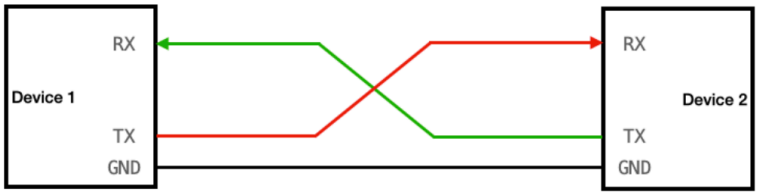


▲ Figure.2-3 An overview of UART protocol (source: web)

We use UART as our protocol to communicate between FPGA and HC-06 module.

UART (Universal Asynchronous Receiver/Transmitter): It enables serial communication by transmitting data bit by bit without a shared clock. It uses a start bit, 8-bits data , and a stop bit to frame and synchronize data, making it ideal for point-to-point communication.

Challenge: We spent three days troubleshooting a wiring mistake where the RX pin of the FPGA was incorrectly connected to the RX pin of the HC-05 Bluetooth module. Since both are receivers, no communication occurred. Correcting this by connecting RX to TX resolved the issue, enabling proper data exchange.

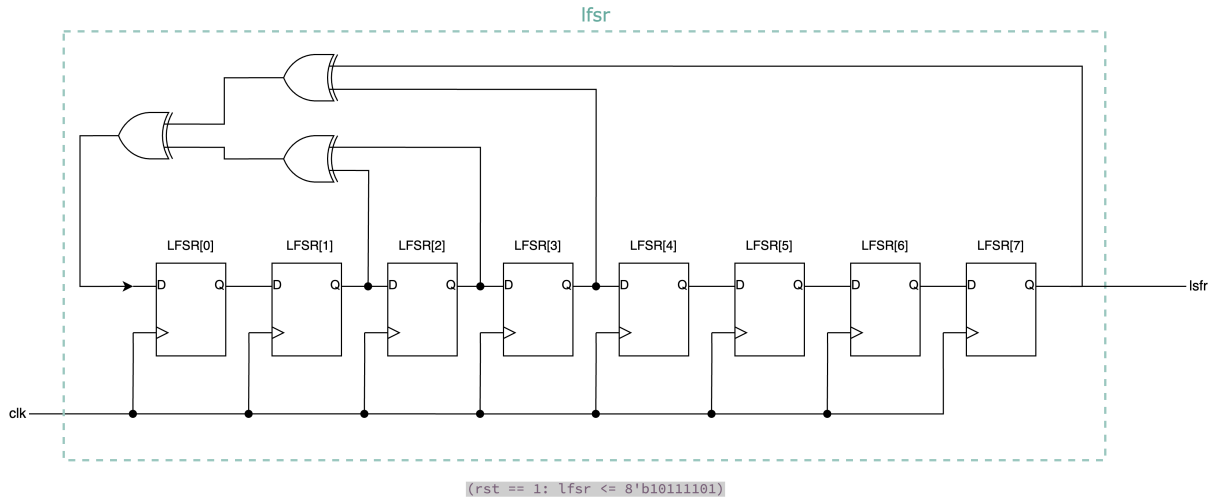


▲ Figure.2-4 Right connection of UART (source: web)

Experiment: By the way, we have tried to implement parity bit in verilog, but the app can't fully control the way to transmit as PUTTY or any other terminal in computer, so we didn't use it at the very end.

2.4 Random Walk Generation

We implemented a Linear Feedback Shift Register (LFSR) as below to generate random directions for the robot's movement ({lfsr[3], lfsr[0]}). The LFSR provided a cost-effective and efficient way to ensure unpredictability in the robot's path.



▲ Figure.2-5 Our LFSR design and initialization which guarantees four bits appears

Challenge: At first, I only use a 4-bit register and using lsfr[0] ^ lsfr[1] as our LFSR design. However, since the design and the initial seed for the LFSR was poorly designed, leading to limited randomness. It ends up with never act the behavior of FORWARD since the design did not account for every bit, reducing the diversity of generated paths.

After identifying these issues, we refined the initial seed and adjusted the LFSR logic to ensure all bits were properly utilized. I also print all the possible states which contains four bits to choose the one with better distribution (FORWARD and BACKWARD have higher possibility) as the picture shown.

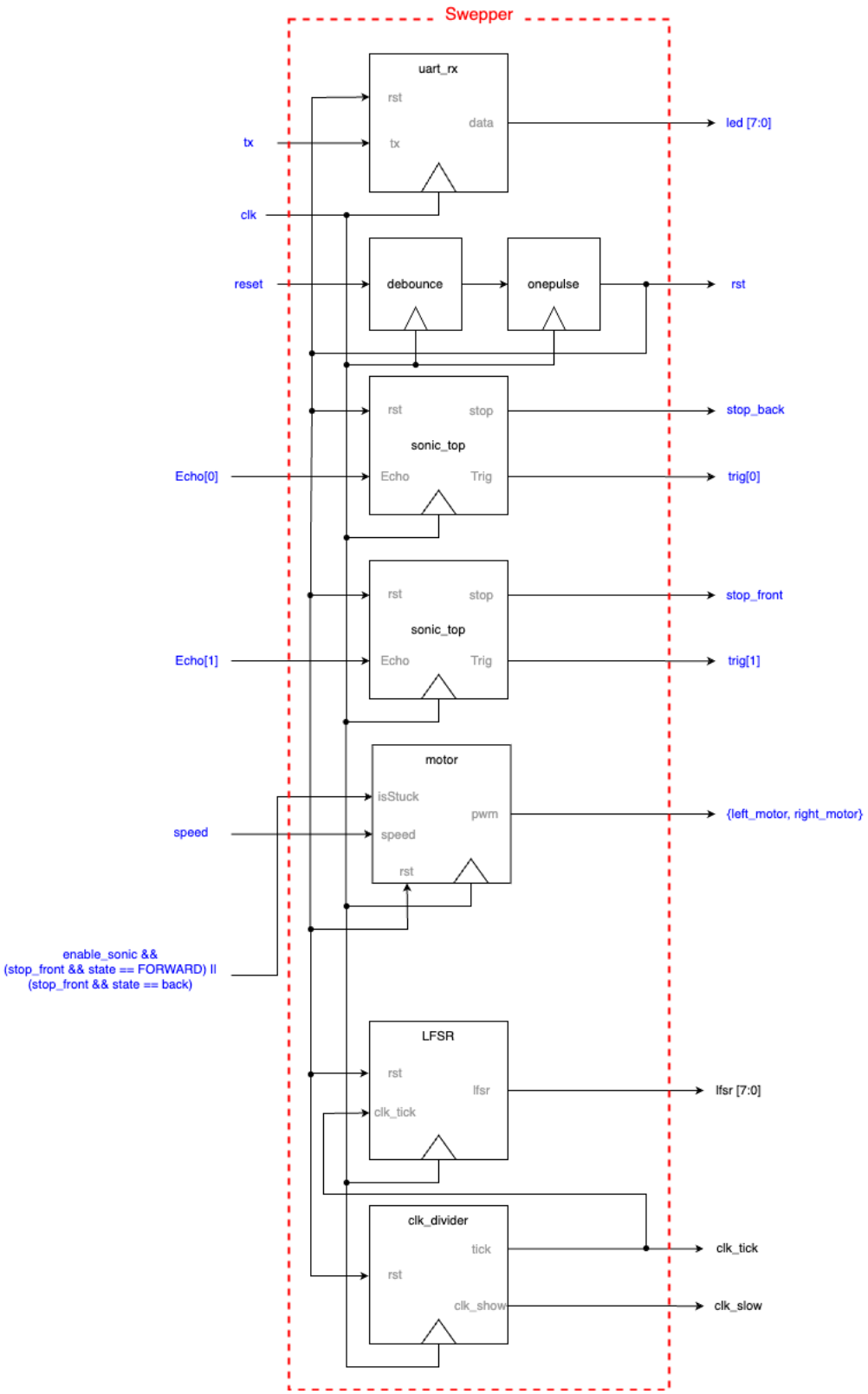
This improvement resulted in more varied and truly random movement patterns, enhancing the effectiveness of the random walk functionality.

0	2	14	11	10	15
0	3	14	11	9	16
0	4	11	14	12	13
0	5	10	15	12	13

▲ Figure.2-6 The distribution (idx1, idx2, [0], [1], [2], [3])

2.5 Our verilog implementation

Module: Sweeper

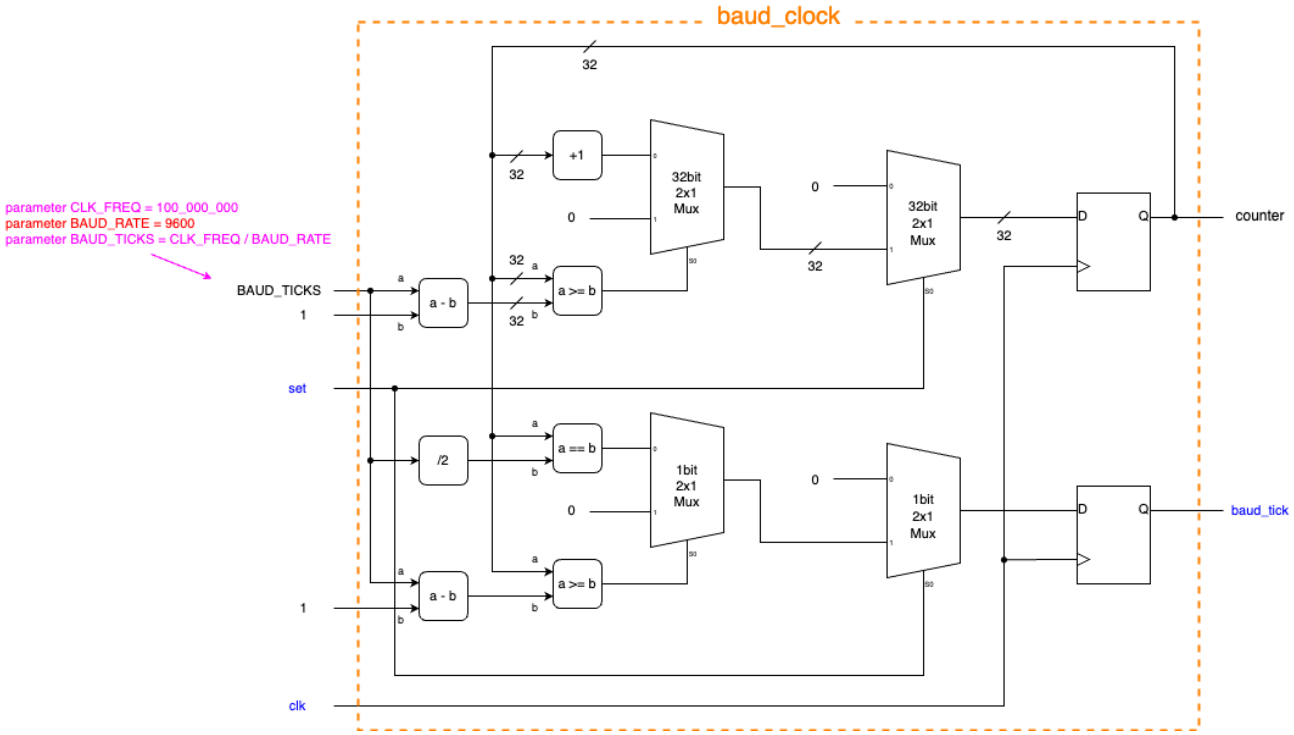


▲ Figure.3-1 Our circuit design of our top module

- Inputs: clk, reset, tx, [1:0] echo, enable_switch, enable_sonic
- Outputs: left_motor, right_motor, [1:0] left, [1:0] right, [7:0] led, [1:0] fan, en_led, [1:0] trig, [1:0] LED_mode
- Wires: [19:0] dis [1:0], stop_front, stop_back, isStuck, rst, rst_pb, clk_tick, clk_tick_17, clk_slow_17, [7:0] en_sig_pb;
- parameters: BACK, LEFT, RIGHT, STRAIGHT, STOP, SLOW, MID, FAST, CONTROL, RANDOM (motor behavior, motor speed, sweeper mode)

- **Purpose:** The **Sweeper** module is a top-level controller for a robotic sweeper, integrating motor control, ultrasonic sensors, Bluetooth communication, and LED indicators. It receives commands from a Bluetooth module (HC-06) to control the robot's movements, speed, and operating mode. The module also utilizes ultrasonic sensors to detect obstacles and adjust behavior accordingly. It supports two modes: manual control (CONTROL) and random movement (RANDOM). The system status is indicated via LEDs and a seven-segment display, and the fan operation can be toggled for enhanced cleaning functionality.

Sub-module: baud_clock



▲ Figure.3-2 The diagram of our baud_clock design

- **Inputs:** clk, set
- **Output:** baud_tick
- **Register:** counter
- **Parameters:** CLK_FREQ, BAUD_RATE, BAUD_TICKS = CLK_FREQ / BAUD_RATE
- **Purpose:** The module divides the high-frequency system clock to create a much slower clock signal (baud_clk) at the desired baud rate (9600 Hz), which is necessary for bluetooth's UART communication protocol. This is done using a counter that counts system clock cycles.
  BAUD_TICKS indicates how many baud_clk will be triggered in one system clock period.

Sub-module: uart_rx

- **Inputs:** clk, rst, tx
- **Output:** [7:0] data
- **Purpose**: to receive and interpret ASCII data from the Bluetooth module. In the UART transmission protocol, the tx signal from HC-06 is normally held at 1. When a 0 is received, it signifies a start bit, indicating that data is about to be read. The following 8 bits represent the data, and it is concluded with a 1-bit stop bit.

Sub-module: motor

- **Inputs**: clk, rst, [1:0] speed, isStuck
- **Outputs**: [1:0] pwm
- **Registers: left_motor, right_motor**
- **Wires**: left_pwm, right_pwm
- **Purpose:** This module controls a motor by generating PWM signals to adjust the speed and behavior. It dynamically modifies the duty cycle for the left and right motors based on input speed parameters and the "stuck" condition. Activates a specific duty cycle to reduce motor strain when a "stuck" condition is detected (**isStuck** is high).

Sub-module: sonic

- **Inputs**: clk, rst, echo
- **Outputs**: Trig, stop, [19:0] dis
- **Wires**: clk_1M, clk_2_17
- **Purpose:** This is an ultrasonic distance measurement module using a sensor (e.g., HC-SR04) to measure distances and trigger a stop signal when the distance falls below 25cm.

3. Conclusion

PilotClean successfully demonstrates the integration of FPGA technology into a dual-mode cleaning robot, addressing limitations in flexibility and efficiency found in traditional robot vacuums. By incorporating features such as remote control, random path generation, and a robust dust collection system, the project highlights the potential of combining manual and autonomous capabilities in one device. Despite challenges in power limitations, dust management, and initial hardware configurations, the team overcame these obstacles to deliver a functional and innovative solution.

Possible Improvements:

1. Upgrade Power Supply: Consider replacing the L298N module with a power board capable of delivering a higher output voltage. This would allow the use of more powerful and efficient fans, enhancing suction performance.

2. Optimize Size: Reduce the overall size of the robot to improve maneuverability. The current design is slightly bulky, and optimizing internal space could lead to a more compact and efficient form factor.

3. Develop a Custom App: Create a dedicated mobile app to enable more advanced and customizable features. This could include adding a parity bit for data transmission, custom controls for specific modes, and seamless integration of additional functionalities tailored to user needs.

4. Better pathfinding algorithm: Since our RANDOM mode can't traverse the whole field in reasonable time now, we need a good algorithm (like A*) to help our robot act more reasonable.

The GitHub repository for the Verilog codes is available at **vaclisinc/PilotClean**. Feel free to explore, use, or contribute to the project. Thank you!

4. Contributions

(1) Song-Ze, Yu

- Write the verilog codes of our whole design
- Responsible for designing the prototype, logo, and presentation slides for PilotClean, with a focus on coding and whole visual design
- Write respective parts of report Introduction, Design Details and Conclusion

(2) Yun-Zhong, Lai

- Write the verilog sub-module of UART protocol
- Most of the hands-on tasks, such as cutting cardboard, drilling holes, and connecting Dupont wires, focusing on hardware-related assembly and prototyping
- Write respective parts of report Design Details
- Draw block diagrams of our PilotClean design