Automatic Pet Door System

Team 27

Haijian Wang (haijian4)

Haoran Zheng (haoranz8)

Zhihao Xu (zhihaox4)

TA: Wang, Yixuan

Professor: Arne Fliflet

May 3th, 2023

Abstract

Our project aims to build a system that can be mounted on regular pet doors to automatically unlock the door when a cat or dog approaches. We successfully finished this project by using a Raspberry Pi with pre-trained AI model, an electricity-powered latch, and other complementary electronic components. This provides a detailed record and explanation of project functionality, individual components, subsystem connections, AI mechanisms, design processes, and test results.

Contents

1. Introduction

1.1 Problem

For those people living near small natural ecosystems, human-wildlife conflicts could frequently happen, and one type of such conflict is intruding wild animals. As pet doors are designed for free entry and exit for medium-sized pets like cats and dogs, wild animals of similar size like raccoons, skunks, opossums, and even coyotes can potentially intrude into people's houses through pet doors [1].

Although most intruding animals' incidents are not fatal, some of these wildlife animals can pose a threat to pets and humans: KCAL9 News reported that a coyote trespassed into a house in Buena Park through the pet door and killed a pet dog in 2019 [2]. Since wildlife animals' intrusion is hard to predict, the most intuitive action to prevent wildlife animals from intruding is to lock the pet door. However, locking the door diminishes the benefits of having pet doors in the first place, as pets can no longer freely enter or leave the house on their own.

1.2 Solution

The cause of the above-mentioned problem is that, unlike the pet's owner, a pet door can neither distinguish between wild animals and pets nor lock/unlock without human intervention. Our goal is to design a pet door to automatically make the correct decision: permitting pets to enter while preventing wild animals from intruding. When an object approaches our pet door, our system will execute the following three sequential steps without manual controls: Detection, Evaluation, and Action.

Detection: Our pet door automatically monitors the motion of any approaching object in real-time. The captured data will be stored for further analysis.

Evaluation: After successfully capturing the data, our pet door begins to analyze whether the approaching object is a pet or not. Currently, our pet door only categorizes cats and dogs as pets.

Action: After successfully identifying the species, our pet door unlocks itself if the result is a pet and locks itself if the result is not a pet.

1.3 Visual Aid

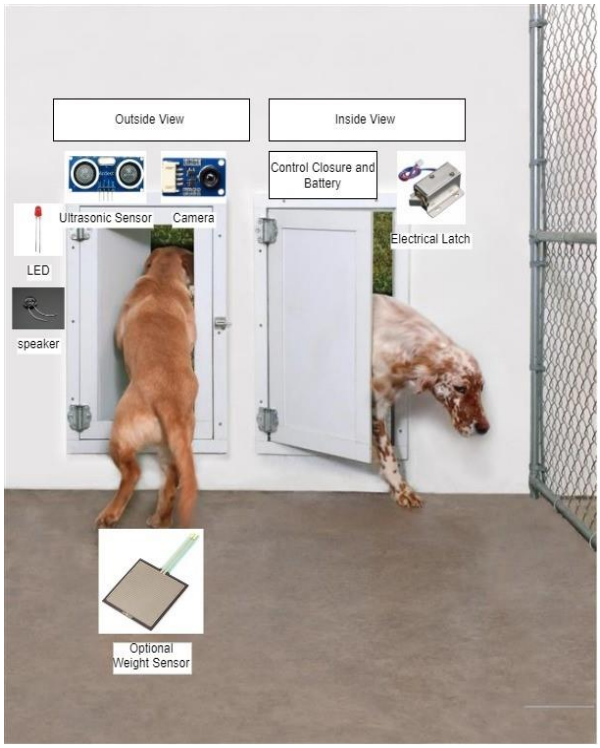


Figure 1: The conceptual pet door on both sides

1.4 High-level Requirements

Our latest iteration of design can achieve the following requirements:

1. If the approaching object is a cat or dog, our pet door can categorize it as a pet and unlock within 1±0.5 seconds. Once the door is unlocked, it starts a 10±0.1 seconds countdown to give pets enough time to enter the house before locking itself again. If the pet stays in camera's view, the counter will repeatedly reset itself to 10±0.1 seconds to ensure that the latch stays unlocked until the pets enter the house. If the approaching object is anything else, our pet door can correctly categorize the object as non-pet and keep the latch locked.

2. Our pet door only unlocks if the linear distance between the door frame and the identified pet is equal to or less than three meters. The distance measured by our pet door only deviates 0.2 meter at maximum from the actual distance measured by standard rulers.

3. Our system has a rated power of 17.39 watts(W) and can work for 4.83 hours on average before recharging, assuming using the default Howell Energy (HWE) Battery. If powering the Raspberry Pi with external power source such as a socket, the average functional time before recharging will be extended to 35.14 hours, which is a desirable working time.

2. Design

2.1 Block Diagram

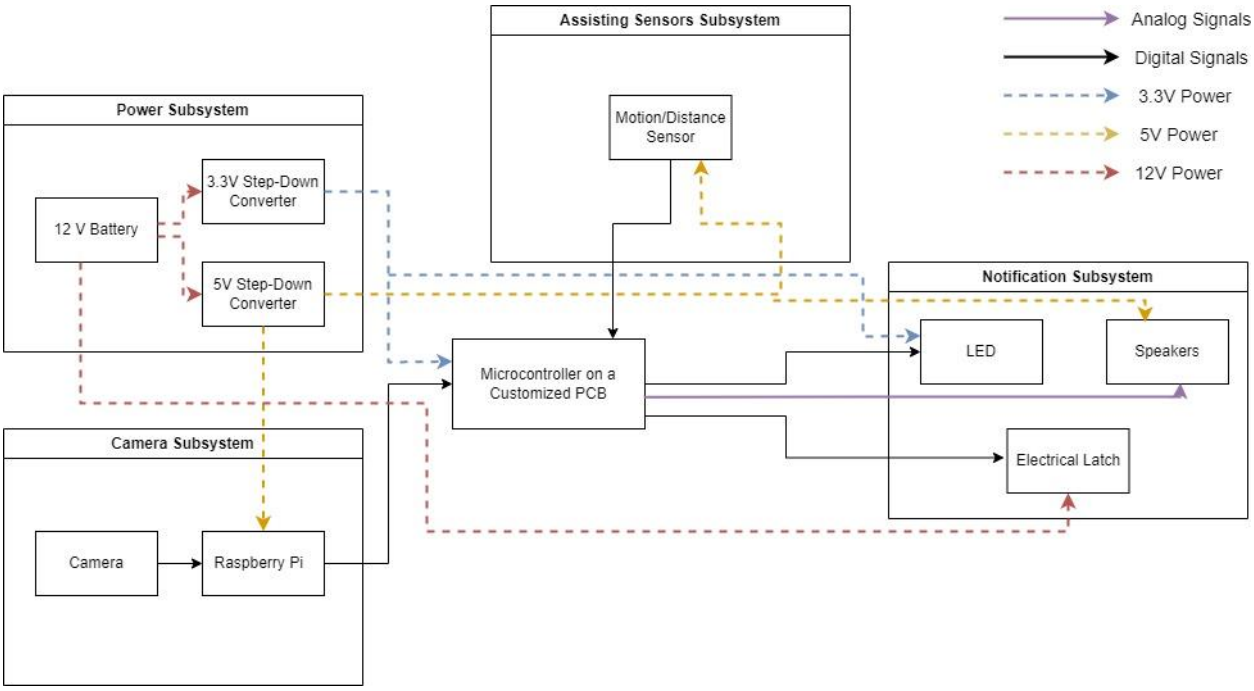


Figure 2. Block Diagram of our design's latest iteration

Our design can be divided into five subsystems: camera subsystem, motion sensor subsystem, microcontroller & Printed Circuit Board (PCB), notification subsystem, and power subsystem. Our system does not include the pet door itself, and all the components can be mounted on most normal pets' doors as accessories or extensions.

2.2 Camera Subsystem

This subsystem consists of a Raspberry Pi 4 Model B responsible for previously mentioned **Evalution** step and a Raspberry Pi Camera Module 3 responsible for previously mentioned **Detection** step.

2.2.1 Training Mechanism

An Artificial Intelligence (AI) model is trained and tested on recognizing pet's images with many batches of labeled photos as inputs. The photos include both the image of pets and other non-pet objects. Those input photos are randomly divided into training sets, development sets, and testing sets with no intersection between each set to detect overfitting errors. Each photo is interpreted as an input vector, and each pixel is interpreted as one element of this input vector. The training process is manifest as a multi-layered neural network built on a linear regression model. In this neural network, adjacent layers are convolutionally connected.

In forward propagation, the AI makes its judgment and generates its own labeling. Then the AI compares it to the assigned labels. After finishing forward propagation, the AI begins the backward propagation: it adjusts the weight of each pixel to reduce the error gradually and thus improve labeling accuracy.

We used YOLOv3-tiny (You Only Look Once) as the backbone of our model. The idea of using YOLO is inspired by some online blogs for object detections [7][8] On the neural network level, YOLO adopts a few

layers of convolution 3x3 and convolution 1x1 layers to satisfy the fundamental requirement of image interpretation. This YOLO model is relatively small, so it excels in high processing speed at the cost of minor accuracy decrease. Since our system is not required to perform more demanding identification tasks such as distinguishing individual animals from the same species, the accuracy provided by this YOLO model is still sufficient, making it ideal for our embedded system. We used the COCO dataset for training, a publicly available dataset for objects in a common context[12]. Thus, our model is able to identify most animals including cats and dogs, and as an additional benefit, it can also identify objects other than animals such as cups and desks. At the end, we also added a final layer at the end to limit our outputs to certain objects for better efficiency.

2.2.2 Raspberry Pi 4 Model B and Raspberry Pi Camera Module 3

After finishing the training, the AI is loaded onto the Raspberry Pi 4 Model B. The Raspberry Pi Camera Module 3 monitors the outside of the door and sends captured image data to Raspberry Pi as input. We use OpenCV to grab images from the camera sensor and display the video stream on the screen. If the object is classified as a "pet", the Raspberry Pi will generate a high signal and send it to the ESP32 microcontroller. Otherwise, it will send a low signal. In some unusual cases during our test, the AI failed to correctly label the pets, but it would always adjust to a correct label within two seconds. Figure 11 shows the schematics of this subsystem, and the detailed subsystem requirements and verification of this subsystem can be seen in Table 1.

2.2.3 Design Decisions and Alternatives

This subsystem's general design and mechanism are unchanged throughout the project because a camera and a minicomputer are mandatory to achieve the desired functionality described in high-level requirements. However, there were some alternative models to our chosen cameras and minicomputers. Initially, we believed that an Arduino board could interpret the imagery data streamed from Arducam Mini Module Camera and identify the species. However, after doing some online research, we abandoned these two components, as the Arduino board's computational power proved not sufficient to support the AI models. Therefore, we decided to use the Raspberry Pi as our minicomputer due to its great computation power and excellent compatibility with the Raspberry Pi camera.

For the AI model, there are some alternatives such as Res-Net and YOLOv7. We eventually decided to pick YOLOv3-tiny as our model for its simplicity and high performance. Its parameter size is only around 30 Megabytes (MB), so our Raspberry Pi could complete image interpretation at a rate of 2.4 frames per second (FPS) in real-time. It took the best tradeoff between speed and accuracy and was the best option available for our project. However, there are still some parts to improve upon to optimize the architecture further: pruning and quantization could be taken for better speed.

2.3 Motion Sensors Subsystem

This subsystem consists of an ultrasonic distance sensor, and this component is also responsible for the previously mentioned **Detection** step.

2.3.1 Ultrasonic Distance Sensor - HC-SR04

In some scenarios, pets can be far away from the door and have no intention to enter the house, but the camera still captures the image of the pets and unlock the latches, causing unnecessary power consumption and potential danger. To avoid this undesirable situation, an ultrasonic distance sensor powered by one

RED WOLF 12 volts to 5 volts (V) converter is used to detect the presence of objects in front of our pet door. The ultrasonic distance sensor emits sound waves as a signal and receives the reflected waves after hitting an obstacle within the practical range. If the reflected signal is different from the original signal, the ultrasonic sensor will consider it meets an obstacle. The time lapse between sending and reflected rave will be sent to the microcontroller. The detailed subsystem requirements and verification of this subsystem can be seen in Table 2.

2.3.2 Design Decisions and Alternatives

This subsystem's purpose had not changed throughout our project, but it originally consisted of a weight sensor instead of our current ultrasonic distance sensor. In our previous plan, this weight sensor should be put right in front of the pet door to determine whether the approaching object was close enough by measuring the weight. However, unlike the ultrasonic sensor, the weight sensor required the pet to stay at a specific location to trigger, and it also needed a much larger space than the HC-SR04 ultrasonic sensor currently in use. Therefore, although the weight sensor could theoretically achieve our desired functionality, it was inferior to the ultrasonic sensor in our current design.

2.4 Microcontroller & PCB

2.4.1 Microcontroller

An ESP32-S3-WROOM-2 microcontroller powered by a 3.3V converter is the main control of the whole system. This microcontroller receives signals from Raspberry Pi and pulse duration measured in microseconds (μs) from the ultrasonic distance sensor respectively. After reading the pulse duration of the obstacle by using "pulseIn()" function in Arduino library, the microcontroller can calculate the distance between pet door and approaching object measured in centimeter (cm) with Equation (2.1):

$$D = (Pd/2) * 0.0343 \text{ cm/μs} \quad (2.1)$$

The symbol "D" stands for distance between approaching object and the pet door. The symbol "Pd" means pulse duration mentioned above, and it is divided by two in the equation because the soundwave travels twice the distance between the pet door and the approaching object, as it first reaches the object, and then returns to the door. The "0.0343 cm/μs" is the soundwave speed in air.

After calculating the distance, our microcontroller compares it to a manually preset distance, which is 300 centimeters in our test case. If the obstacle's distance is smaller than the preset distance, and the signal from Raspberry Pi is high at the same time, the microcontroller will generate and send a high signal to the Gate pins of the MOSFETs on the PCB to power the latch and the speaker. The duration measured in seconds (s) of the high signal is determined by Equation (2.2):

$$Sd = t + 10\text{s} \quad (2.2)$$

The symbol "*t*" is the time duration the pet stays in camera's view and the three meters range. Symbol "*Sd*" means the signal duration. As shown in Equation (2), the high signal will not cease as long as the camera can still capture the image of pets within three meters, and after the pet disappears from the camera, the signal will also continue for 10 seconds before ending.

Our microcontroller also sends wave-shaped output to the speaker by using the "tone()" function provided by the Arduino library, so the speaker can receive waved signals instead of monotonous signals to properly generate sounds.

Physically, this microcontroller is mounted on the left edge of our PCB board. The block diagram and schematics of this microcontroller can be seen in Figure 9 and Figure 10 respectively.

2.4.2 PCB

Our initial PCB design is inspired by some online blogs and follows the official reference design requirement: we use a 40MHz crystal oscillator with two 20pF capacitors for the chip as required by the datasheet[9][10]. We include a debounced reset button at the CHIP_EN pin. For the USB connection, we use a micro-USB connector and a CP2102 chip to convert the USB signals to UART signals. The detailed subsystem requirements and verification of this subsystem can be seen in Table 3.

2.4.3 Design Decisions and Alternatives

We initially decided to use the STM32U5 microcontroller for its great performance and low power consumption. After some research and discussion, however, we decided to switch to the ESP32 microcontroller, as it was much more affordable to purchase on a large scale, and its guidelines were clear and easy to follow.

Our final product uses the second iteration of our PCB, which solves many problems our first iteration had. In our original PCB design (seen in Figure 8), the wires routes produced right-angle turnings and were poorly organized, which hindered the movement of electrons and consequently lowered the currents. Even worse, our 2102 chips and some other capacitors were difficult to be directly hand-soldered on the PCB. Also, one linear regulator was proved not enough to convert 12 V down to 5 V, because the conversion generates an excessive amount of heat to damage our board and components.

Thus, we completely redesigned our PCB in the latest iteration (seen in Figure 7). We removed the CP2102 chip and implemented two connectors to allow external RX/TX connections, so we could use a USB-to-UART conversion adapter to program our chip. We added an XT60 connector for the 12V power input because its unique shape could prevent mixing the ground and power pins. We also added one more linear voltage regulator to convert 12 V to 5 V before converting it down to ESP32's desired 3.3 V, which solved the overheating problem. Another essential design choice of our PCB is the two MOSFETs (Q1 and Q3) to control the power of our latch and speakers. The current from the microcontroller pin was limited to 250 mA, which was not enough to power the 350 mA latch. Thus, we powered these components with our 12V battery and the two MOSFETs acted as switches for the circuit. The speaker and electrical latch will be connected to the board through two screw terminals.

2.5 Notification Subsystem

This subsystem consists of an electrical-powered latch, a speaker, and two MOSFETs. It is responsible for the previously mentioned **Action** step.

2.5.1 Uxcell DC 12V 0.35A Electric Lock Cabinet Door Latch & MOSFET 1

This latch is in the "locked" state if no voltage is applied. Once receiving a 12V input voltage, it will unlock the pet door and let the pets freely enter the house. Our HWE Battery will provide enough voltage and power for this latch. MOSFET 1 controls the electricity flow to this latch, and if the microcontroller sends a high signal to the Gate of MOSFET 1, it will close the circuit and allow latch to receive the 12V voltage and thus unlock the pet door. The latch is only installed on the inner side of the pet door, so the pet door is only locked from the outside, and anything in the house can freely go outside regardless of the state of the latch.

2.5.2 Uxcell a15080600ux0275 Round Internal Magnet Speaker & MOSFET 2

This speaker will generate sounds if a 5V voltage is applied. MOSFET 2 controls the electricity flow to this speaker, and if the microcontroller sends a high signal to the Gate of MOSFET 2, it will close the circuit and allow this speaker to receive a 5V regulated pulse from the microcontroller and thus make melodies. When sounds are generated, it indicates that the object in front of the door is recognized as a pet and the latch is unlocked. Otherwise, no sounds will be generated and indicate that the latch is locked. The speaker is directly mounted on the PCB board inside the house. The detailed subsystem requirements and verification of this subsystem can be seen in Table 4.

2.5.3 Design Decision and Alternatives

Originally, we used a much larger latch that required 1.1 ampere (A) instead of 0.35 ampere to work properly. Our previous latch choice was problematic, as our microcontroller could only produce 5V voltage to the Gate pin of MOSFETs, which was not high enough to fully connect the Drain and Source pin of the MOSFET, thus lowering the current flowing through the latch to 0.34±0.04 ampere. As a result, the previous latch could not respond to commands from the microcontroller due to the low current. After we replaced the 1.1 ampere latch with our current 0.35 ampere latch, the problem was solved.

2.6 Power Subsystem

This subsystem consists of an HWE 12.8V 7Ah Battery and two RED WOLF Adjust DC converters. Although this subsystem does not directly participate in any of the three steps, it provides power to the rest of the system, so all other components can function properly.

2.6.1 HWE 12.8V 7Ah Battery & RED WOLF Adjust DC converter

This battery serves as the main power source of the system. When powering to the electrical-powered latch, it can directly supply its 12V voltage. When powering the ESP32 microcontroller, it is connected to a RED WOLF Adjust DC converter with the output voltage set to 3.3V for safety concerns. When powering the Raspberry Pi, the speaker, and the ultrasonic distance sensor, it is connected to a RED WOLF Adjust DC converter with the output voltage set to 5V for safety concerns.

2.6.2 Design Decisions and Alternatives

In our initial design, we used a Lithium-Ion 3.7V Battery Pack and the MT3608 DC-DC Step-Up Boost Converter to boost the voltage to power the Raspberry Pi and latch. We originally decided to use this low-voltage battery mainly for size concerns, as our current HWE Battery has a three-dimensional size of 5.94*2.56*3.7 inches, which cannot be mounted on the pet door. However, the Lithium-Ion 3.7V Battery Pack only had a 22-watt-hour capacity, which was much inferior to our current 84-watt-hour capacity. As a result, we made a tradeoff to favor capacity over portability.

3. Requirement & Verification:

In this section, we will go over our verification methodology for this project in general. We already had a discussion about requirements and verifications in previous sections. In this section, we will look at them in a more systematic manner. To ease our debugging process, we tested each subsystem individually for every single PCB board first. To test our power conversion circuit, we would give it 12 volts from the power supply directly and then check the output pins' voltages. A functional circuit normally would give us 3.32V and 4.54V after each conversion. An example read of a functional converted power voltage for the microcontroller is shown in Figure 3(3.3V ESP32). Then, we would test our MOSFETs circuits. In fact, we designed our MOSFETs circuit in configurable manner and each gate can be connected externally, which means that we could change the input signal at the gate by connecting to different pins. To test the proper functionality of this part of our PCB, we chose to power the circuits with the 12V battery and input a 3.3V signal from the power supply at the gates. As shown in Figure 4(12V mos), the MOSFETs can supply 12V power between their drain and source. Furthermore, we would make sure that every other signal can switch between high and low properly. An example here would be the EN pin for the ESP32. As shown in Figure 5(raspberry high and low), it can stay high at around 3.25V with 100mV peak-to-peak difference, and pressing the button can make it low at around 100mV with 125mV peak-to-peak difference, which is acceptable for our case.

For our object detection model, we got a high true positive rate under our special condition. We tested 30 different dog pictures from Google and they were all classified as "dog." Sometimes, the model might classify the dog as something else and take a bit more time to reach the correct classification, which we would still consider as a true positive for our use case. Out of 30 non-pet pictures we picked, our model only made one mistake and corrected it after some time. If we count it as a false positive, our true positive rate is over 96% and we consider it as a success.

4. Cost

4.1 Parts

Cost analysis:

Description	Quantity	Manufacturer	Extended price
HWE 12.8V 7Ah Battery	1	HWE Energy	$34.56
RED WOLF Adjust DC 12V to 3.3V 5V 6V DC Step Down Converter Power Supply Voltage Adapter Reducer Regulator Fit	2	Red Wolf	$13.99
ESP32-S3--WROOM-2 Microcontroller	1	Espressif	$7.5
HC-SR04 ultrasonic distance sensor	1	Seeed Technology Co., Ltd	$4.3
Raspberry Pi Camera Module 3	1	Adafruit	$35.5
Raspberry Pi 4 Model B	1	Raspberry Pi	$128
Uxcell DC 12V 1.1A Electric Lock Cabinet Door Lock	1	Uxcell	$12.9
Uxcell a15080600ux0275 Metal Shell Round Internal Magnet Speaker	1	Dragonmarts - BISS	$10.55
Optoelectronics LED Indication – Discrete	10	American Opto Plus LED	$1.14
IRF540PBF MOSFET	3	ECE supply center	$1.97
Total costs: $250.41			

4.2 Labor

The Average hourly salary for a graduate student is $40 per hour, and we expect to spend 12 hours per week for 10 weeks. Equation （2.3） shows the results.

$$(\$40/\text{hour}) * 2.5 * 10 * 12 = \$12000 \quad (2.3)$$

$$\text{Total costs} = 12000 + 250.41 = \$12250.41$$

5. Conclusion

5.1 Accomplishments

In the last iteration, our project satisfy all previously mentioned high-level requirements.

In our power subsystem, we successfully build a 12V, 5V, and 3.3V conversion to satisfy all components' needs without damaging them or causing any safety issues. Even without the external power source, the system can work 4.83 hours without recharging, which is a decent working time for a fully automatic system involved in complicated AI processing.

In our camera subsystem, our Raspberry Pi Camera Module 3 can successfully detect the approaching animal and send the images to the Raspberry Pi. Then the Raspberry Pi 4 Model B is able to identify the approaching animal within the range of fixed animal categories including raccoons, elephants, giraffes, cats, and dogs in different breeds. If the identified object is a cat or a dog, the Raspberry Pi will send signals to microcontroller to unlock the latch.

For our microcontroller, it provides all necessary controls in our system to achieve complete automation. The functionality of reading from ultrasonic, outputting melodies from speakers, and unlocking the latch are all successfully automated. The house owner only needs to connect the power wire from the battery and boost the Raspberry Pi, and then the system can function on its own.

In our notification subsystem, the latch is able to finish the unlocking process in 1±0.5 seconds when a cat or dog was within three meters' range. The unlocked time duration is determined by Equation 2, and the speaker will keep ringing for the whole duration to notify the house owner that the pet door is in unlocked state.

5.2 Uncertainties

Through our design process, we contained several uncertainties, but they were all solved.

Firstly, we failed to upload our Arduino code to the microcontroller. We tried several different methods including resoldering components and switching to new USB connectors, but we kept getting error message "No serial data connection" from Arduino Integrated Development Environment (Arduino IDE). Eventually, we figured out that the connection of RX/TX pins on USB side was different from UART side: USB's RX pin should be connected to UART's TX pin, while USB's TX pin should be connected to UART's RX pin. Additionally, we should boot up the ESP32 with proper strapping pins: both GPIO 0 and EN had to be connected to ground so the previous code could be cleaned, allowing new codes to be uploaded.

Another problem came from our latch, which hindered us from making any progress for days. As mentioned in the design alterative section, we initially used a latch that could only function at 12V and 1.1 A, which our microcontroller and MOSFETs could not provide. As a result, the latch did not respond to any command from our microcontroller. We tried several different methods to resolve this problem, including adding several parallelly-connected MOSFETs and additional transistors to provide enough current. However, these methods only worked on breadboard and could not be ported to PCB due to the limited soldering spaces, so we abandoned those methods. This problem was solved when we switched to a new and smaller latch than could function at 0.35A.

Aside from these two main problems, we also encountered many smaller problems such as bad soldering and loose connection. Among all these trivial problems, the most unpredictable one was the broken

microcontrollers, as there was no visual difference between a functional microcontroller and a broken one. We had broken five microcontrollers in total, and every case was caused by excessive intensity of current as a result of short circuits.

5.3 Ethical Consideration

Ethics and safety are our priority when designing the system. Before implementing the physical design, we had considered risks related to our subsystem components including sensors, latches, and power supply as many as possible to ensure that our system would not harm the user, the pets, or itself. There were some minor safety issues we did not consider at first, such as the previously mentioned overheated linear regulator, but as soon as we found any safety issue during testing processes, we would try every possible way to fix it. During the test, we also paid attention to safety. The electrical components in our system were kept away from dangerous factors including water, fire, and sharp objects to prevent harm all the time.

We strictly followed the 7.8 IEEE Code of Ethics to hold paramount the safety, health, and welfare of the public. To protect the privacy of others, we also need to disclose promptly factors that might endanger the public or the environment. During the previous tests, we only built a proto pet door, which was much smaller than the real pet door. Therefore, we need to adjust the wire length and mounting location to fit the actual pet doors of varied sizes in the future.

According to the 7.8 IEEE Code of Ethics III. 10, we should ensure this code is upheld by colleagues and co-workers. Every teammate followed the code of ethics and was honest with each other. Any information related to our design was open to each other. Each teammate in our group finished the Laboratory Safety Training and paid attention to every dangerous factor when working in the lab. We always kept at least two people staying in the laboratory and were careful about the actions of soldering and wiring.

5.4 Future Work

In the future, we plan to implement Multithreading in our AI model to enhance the processing speed. The improved processing speed is expected to significantly increase the image frames our Raspberry Pi can process, so it can interpret more precise pictures during the detection process and improve smoothness. Additionally, the accuracy of our project will be enhanced to a new level.

Furthermore, we can build a more integrated PCB which includes buck converters, auto reset, auto boot, and USB connection. The auto boot and USB connection are more convenient for people to upload code into the microcontroller soldered on our PCB.

We can also build a more customized trained AI model for specific pets. For our AI portion, we are currently using a YOLOv3 pre-trained on the COCO dataset. Although this dataset includes common animals like cats, dogs, and raccoons, it does not have a comprehensive list of wild animals, so similar animals like coyotes and dogs may confuse our current AI model. We can further improve our AI model by training it to identify the specific pets of the house owner, so our door will also distinguish between the house owner’s pets and random stray cats or dogs.

Finally, we also want to build some wireless communication and application to make our automatic pet door more intelligent and convenient. The current speaker notification is not as advanced as wireless communication, as it produces louds sounds, which may not be desirable at night. With the wireless

application implemented, the house owner can receive the signal from the phone text using blue tooth. This promotion also eliminates the disturbance that covers the voice of the speaker, so the information about the approaching pet will no longer be delayed. The house owner can get a message immediately when the pet is entering the house.

Reference:

[1] “Keep Raccoons from Using Pet Doors.” *The Humane Society of the United States*, https://www.humanesociety.org/resources/keep-raccoons-using-pet-doors.

[2] Caplan, Christy. “Coyote Snuck through 'Doggie Door' to Kill Family Dog, Injure Another.” *Pets*, 14 July 2022, https://www.wideopenpets.com/coyote-kills-dog-after-entering-through-doggie-door/.

[3] Winningham, Cathleigh. “Caught on Camera: Coyote Sneaks into Home through Doggie Door.” *WKMG*, WKMG News 6 & ClickOrlando, 30 May 2022, https://www.clickorlando.com/news/weird-news/2022/05/30/caught-on-camera-coyote-sneaks-into-home-through-doggie-door/.

[4] “IEEE Code of Ethics.” *IEEE*, https://www.ieee.org/about/corporate/governance/p7-8.html.

[5] Raspberry Pi. “Buy A Raspberry Pi Camera Module 3.” *Raspberry Pi*, https://www.raspberrypi.com/products/camera-module-3/.

[6] Raspberry Pi. “Raspberry Pi 4 Model B Specifications.” *Raspberry Pi*, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

[7] Damen, Ryder. “Let the Dogs out! How to Make a Raspberry Pi Pet Detector.” *Tom's Hardware*, Tom's Hardware, 3 Apr. 2021, https://www.tomshardware.com/how-to/raspberry-pi-pet-detector.

[8] “03. Predict with Pre-Trained YOLO Models¶.” *03. Predict with Pre-Trained YOLO Models - Gluoncv 0.11.0 Documentation*, https://cv.gluon.ai/build/examples_detection/demo_yolo.html.

[9] “How to Make Your Own esp32 Breakout Board with Minimal Circuit - PCB Design Tutorial.” *PCBway*, https://www.pcbway.com/blog/PCB_Design_Tutorial/How_to_make_your_own_ESP32_breakout_board_with_minimal_circuit.html.

[10] BOY, Electro. “How I Made Own ESP32 Development Board.” *Hackster.io*, 12 May 2022, https://www.hackster.io/electroboy001/how-i-made-own-esp32-development-board-8af733.

[11] “ESP32-S3-WROOM-2-N16R8V ESPRESSIF Systems: Mouser.” *Mouser Electronics*, https://www.mouser.com/ProductDetail/Espressif-Systems/ESP32-S3-WROOM-2-N16R8V?qs=Rp5uXu7WBW%2Fbmj6VRWU1MQ%3D%3D.

[12] “Common Objects in Context.” *COCO*, https://cocodataset.org/#home

Appendix

Appendix A - Requirements and Verification Tables

Table 1: Camera Subsystem – Requirements & Verification

Requirement	Verification
• We should connect and adjust the camera correctly to the Raspberry Pi 4 to catch and present the graph of pets clearly, and our coding part should also ensure the accurate transmission of imagery data.	• Use the 15 Pin cable to Connect the Raspberry Pi Camera Module 3 to the Raspberry Pi 4 Model B board. • Connect the Raspberry Pi board to a monitor with a micro-HDMI cable. • Launch the Raspbian OS and run our Python test scripts. • The script should properly decode the input data from the sensor. • The script should calculate the frame rate. • The script should display the video stream and frame rate on the monitor. Check if the stream is stable and consistent. • Move around an object in front of the camera to test auto-focus.
• Our AI model should identify the category of a given object. When the AI is enabled, the video frame rate should be at least 3 FPS and at most 10 FPS for energy saving.	• With the camera and monitor connected to the Raspberry Pi board, run our AI model's Python test script in the Raspbian OS. • The video frame rate should be above 3 FPS and below 10 FPS. Otherwise, we should optimize our code further with multithreading. • A colored box will be drawn around the identified object and its category will be displayed in text near the box. • Use real objects and pictures from the COCO dataset to test the accuracy of our model, which should be over 90%.
• Our AI model should generate a digital high signal while a target category object stays in the frame. In our case, the target categories are cats and dogs.	• Set the target categories in our Python script. For testing purposes, we will use common objects such as apples and bags. • Connect an LED test circuit to the output pin so that we can verify the signal visually. • Put the target object in front of the camera, and then remove the objects. The output signal should be low within 10s. Check if the behaviors of the LED satisfy our design requirements.

• We have to ensure the stability of our circuit's connection, which means our design should function properly for at least two days.	• Test our OS and hardware stability by running the OS for two days. We will monitor the circuit status and test judgement correctness once per 3 hours except for the sleeping time.

Table 2: Motion Sensors Subsystem – Requirements & Verification

Requirement	Verification
• Our ultrasonic distance sensor should be installed in the correct position facing the outside direction, so its emitted sound wave could detect obstacles in a long range.	• The ultrasonic sensor should be powered by a 12V to 5V converter in the power subsystem to work with enough voltage. • The location of ultrasonic sensor is set up near the camera and they are set up at a common horizontal line to make sure the signal is received simultaneously. • The signal is sent to the microcontroller to unlock the latch which presents whether the signal is reflected correctly.

Table 3: Microcontroller – Requirements & Verification

Requirement	Verification
• We should ensure our coding part function properly, so the microcontroller will generate control signals properly when the according to the signal from the sensor.	• The input signals from the Raspberry Pi and distance sensor should be recorded properly. We will use an unused pin and an LED to visualize if the signals are detected and the circuits function properly.
• Our designed PCB should be physically fitting and support the functionality of the rest of the system.	• Pass the DRC verification test • Satisfy the physical dimension limit. • Specifications of the official datasheet and hardware design manual are satisfied. • When all other components work properly, the PCB should ensure the functionality of the whole system.
• The microcontroller should be programmed through the Type-C/micro-USB port.	• We will use Arduino IDE with a customized pin assignment to program our microcontroller. • The USB port and UART soldering connection should be correct and precise. Otherwise, we will use a multimeter to test each soldered pin to debug.

• The whole board, especially the microcontroller, should operate at the recommended temperature range.	• We will use an IR thermometer to measure the temperature of the device under high loads and light loads. • If the temperature went above 90 °C, we would add a copper heat pipe in our design to dissipate heat.

Table 4: Notification Subsystem – Requirements & Verification

Requirement	Verification
• The Latch, LED, and speaker should be correctly controlled by the microcontroller's signal. To achieve this, we need to connect this subsystem with microcontroller and MOSFETs properly.	• They should be powered by different converters to satisfy enough voltage. LED should be powered by a 12V to 3.3V converter. Speaker, the latch should be powered by 12 V to 5 V converter. • The Latch, LED, and speaker should be connected to diverse MOSFETs separately. All three MOSFETs are connected to the microcontroller. • Microcontroller sends a high signal to MOSFET, the Latch will be unlocked, the LED will light up and the speaker will generate sound. Otherwise, the latch will keep locked, the LED will not light and there are no sound generated.

Table 5: Power Subsystem – Requirements & Verification

Requirement	Verification
• Despite being powered all by a 12V battery, each electrical component should run under their designed rated voltage. With the help of voltage converters, we should connect the circuits in such a way that the power subsystem will provide proper 3.3V to the microcontroller and LED, provide 5V to Raspberry Pi, ultrasonic distance sensor, and the speaker, provide 12V to the electrical-powered latch.	• Use a 12.8V 7Ah Battery and connect it with a DC step-down converter from 12 V to 3.3V/5V. • Use the voltmeter to measure the voltage after converting to ensure that it reaches the requirement of the microcontroller, optional weight sensor, LED / Raspberry Pi, Ultrasonic Distance Sensor, latch, and the speaker. If all of these measured voltages' errors are within ± 0.3V, it will be considered success.

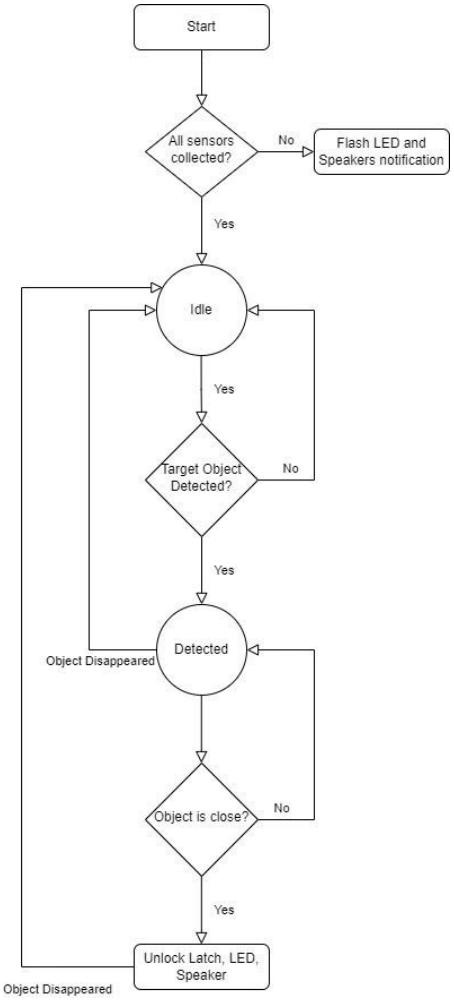
Appendix B- Schedule：

Week	**Task**	**Person**

February 20 – February 24	Finish design document	Everyone
February 27 – March 6	Design review and order components	Everyone
	Prototype design on the bread board and check the problems in our design	Haoran Zheng
	Revise our design and check each module's feasibility	Haijian Wang
March 6 – March 13	Begin PCB design and PCB review	Haoran Zheng
	Revise our PCB design	Zhihao Xu
	Place the order for PCB and team evaluation	Everyone
March 13 – March 20	Spring break	N/A
March 20 – March 27	Coding our AI part for our project and building connection with the sensor	Haijian Wang
	Integrate sensors with PCB and test the whole system	Haoran Zheng
	Revise the PCB design according to the testing results	Zhihao Xu
March 27 – April 3	Place the second order of PCB	Everyone
	Individual progress reports	Haijian Wang
April 3 – April 10	Finalize the camera system and integrate other sensor modules	Zhihao Xu
	Check the functionality of the whole system and debug the system	Haijian Wang
April 10 – April 17	Finalize the document and notebook	Everyone
	Prepare for the mock demo	Everyone
April 17- April 24	Mock demo	Everyone
April 24 – May 1	Finish final paper	Everyone

Table 6 schedule list

Appendix C flow chart:



Appendix D schematics and PCB layout:

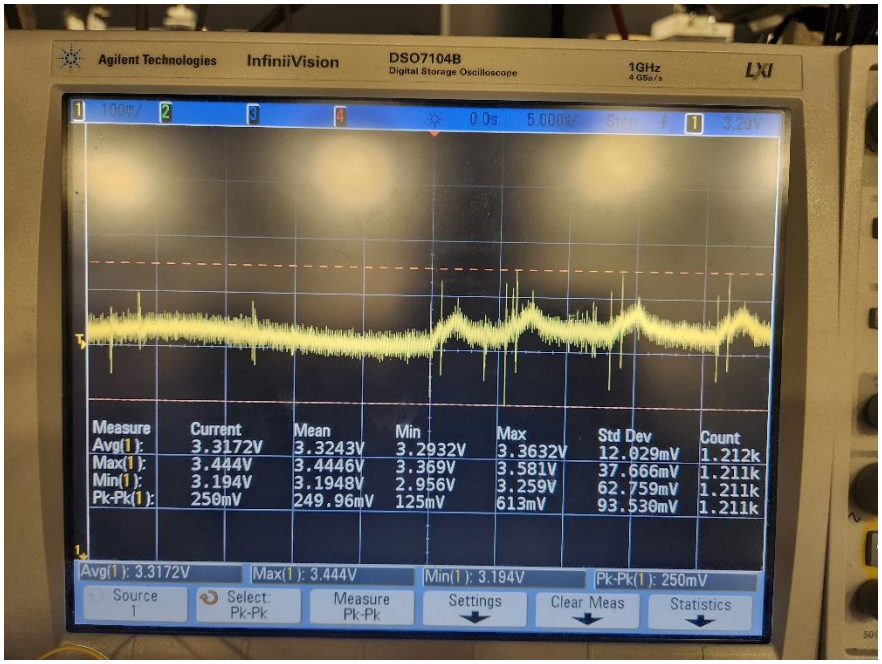


Figure 3：3.3 v ESP 32

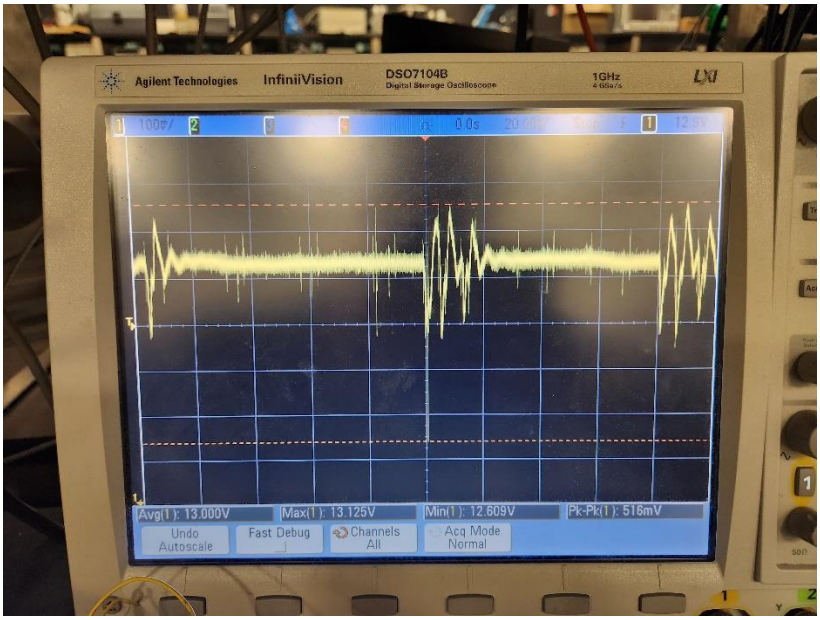


Figure 4：12v MOSFET

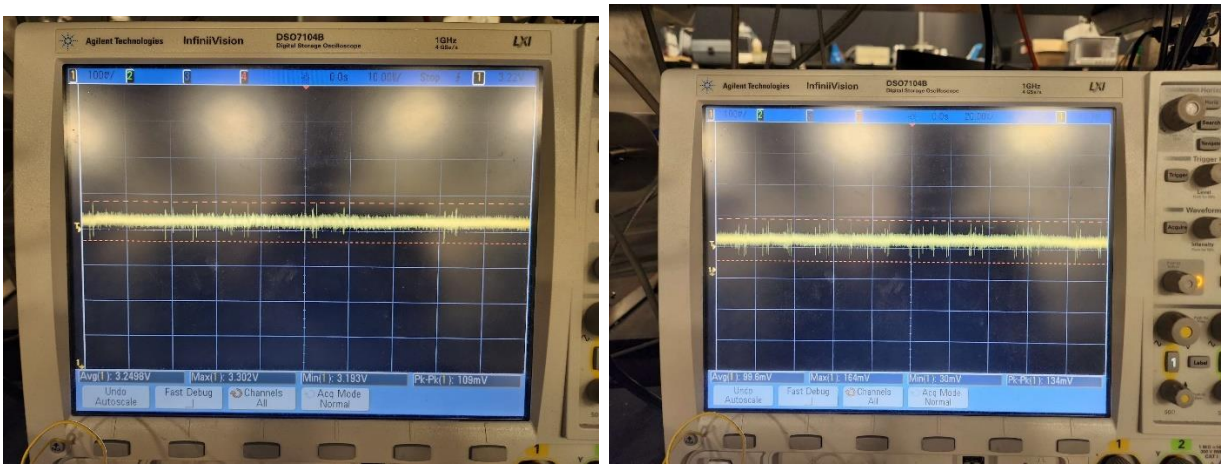


Figure 5：raspberry high or low

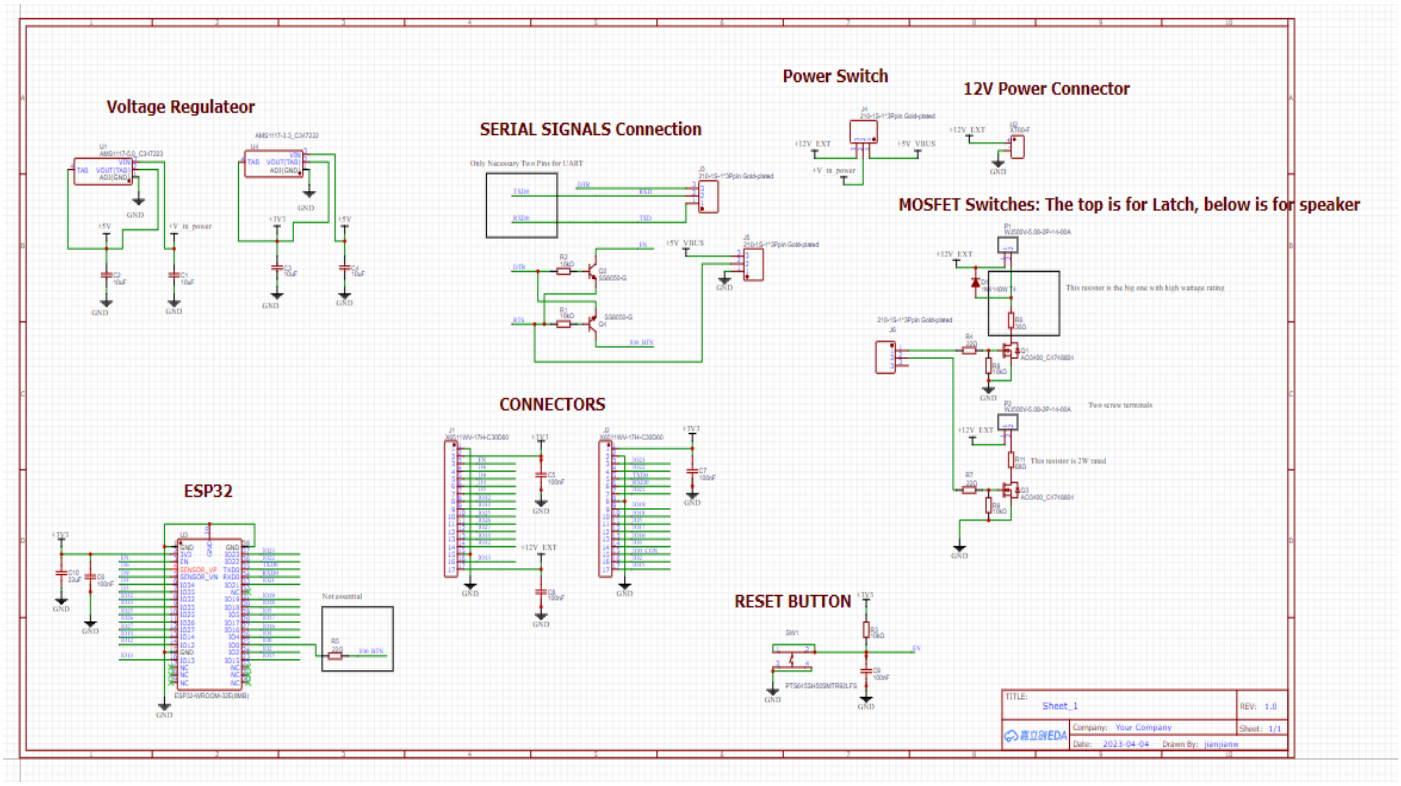


Figure 6： Schematics of PCB

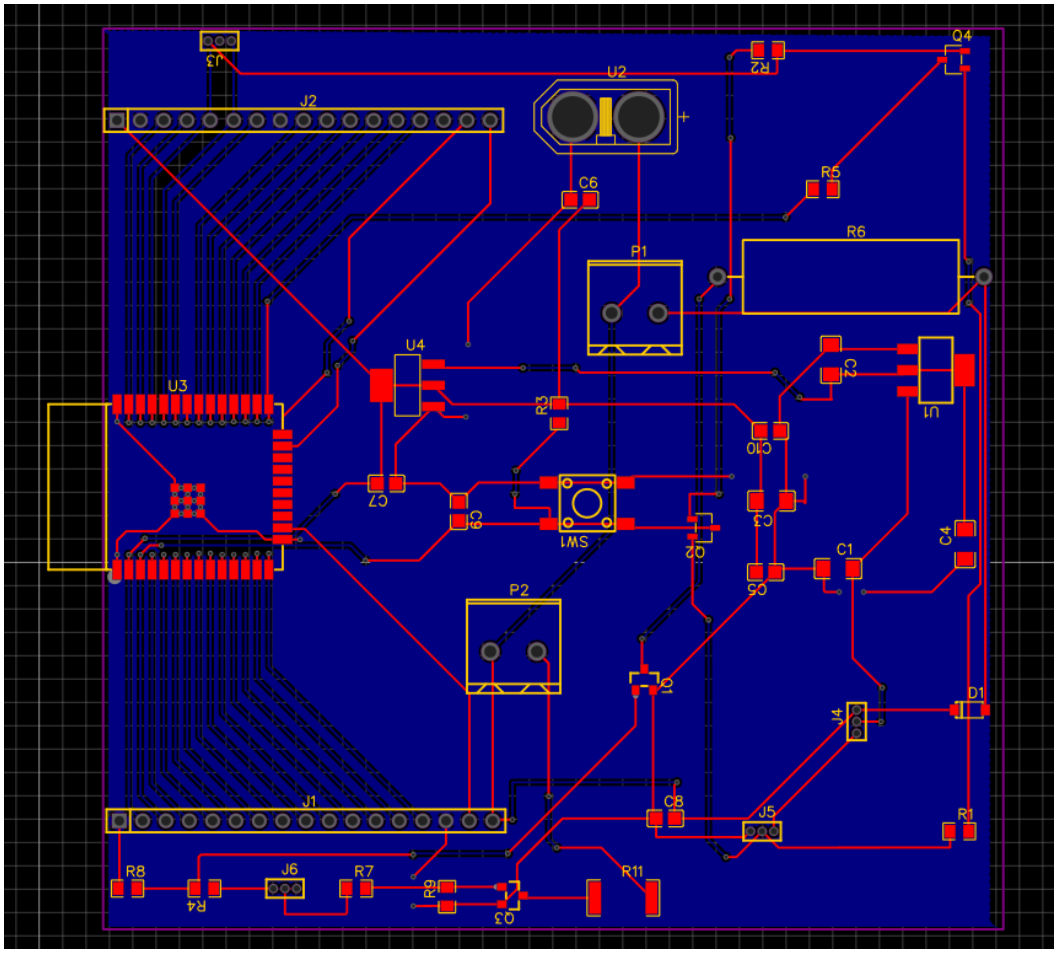


Figure 7： new Layout of PCB

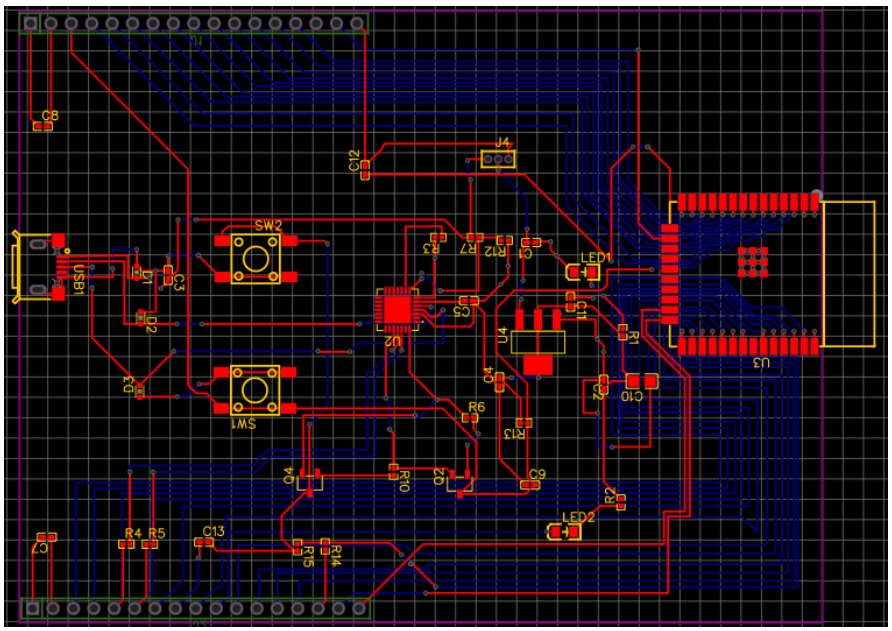


Figure 8：old Layout of PCB

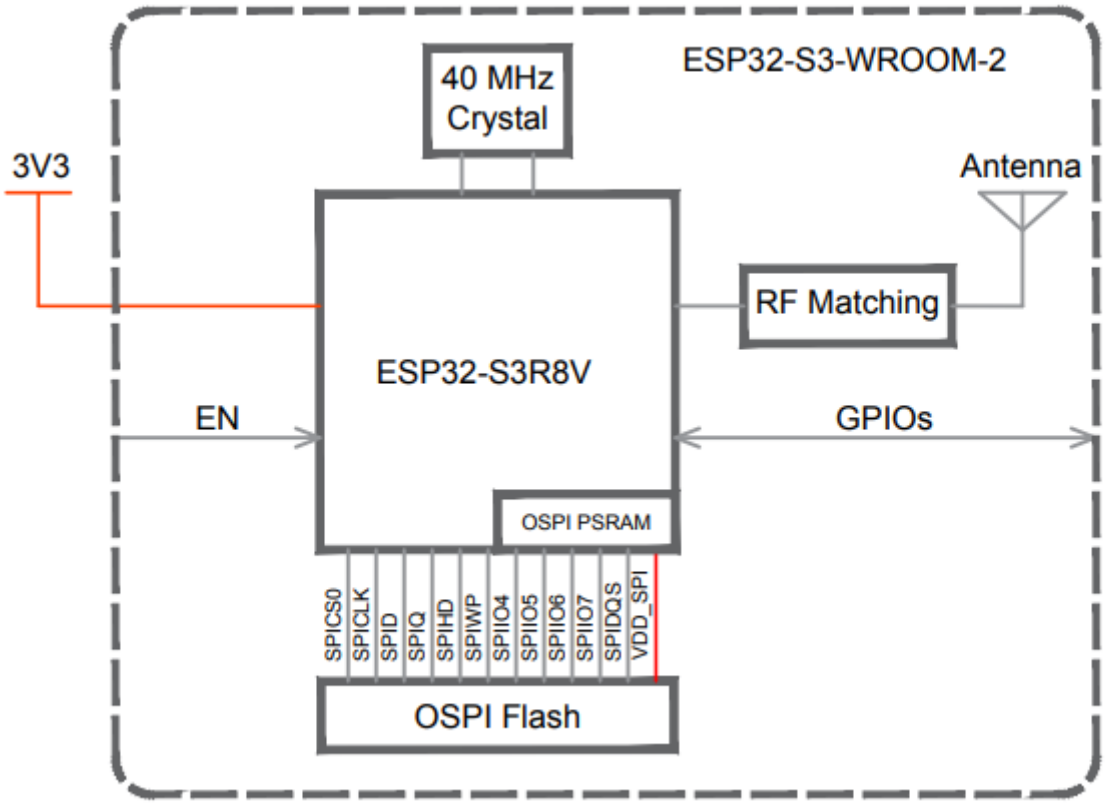


Figure 9: ESP32-S3-WROOM-2 Block Diagram

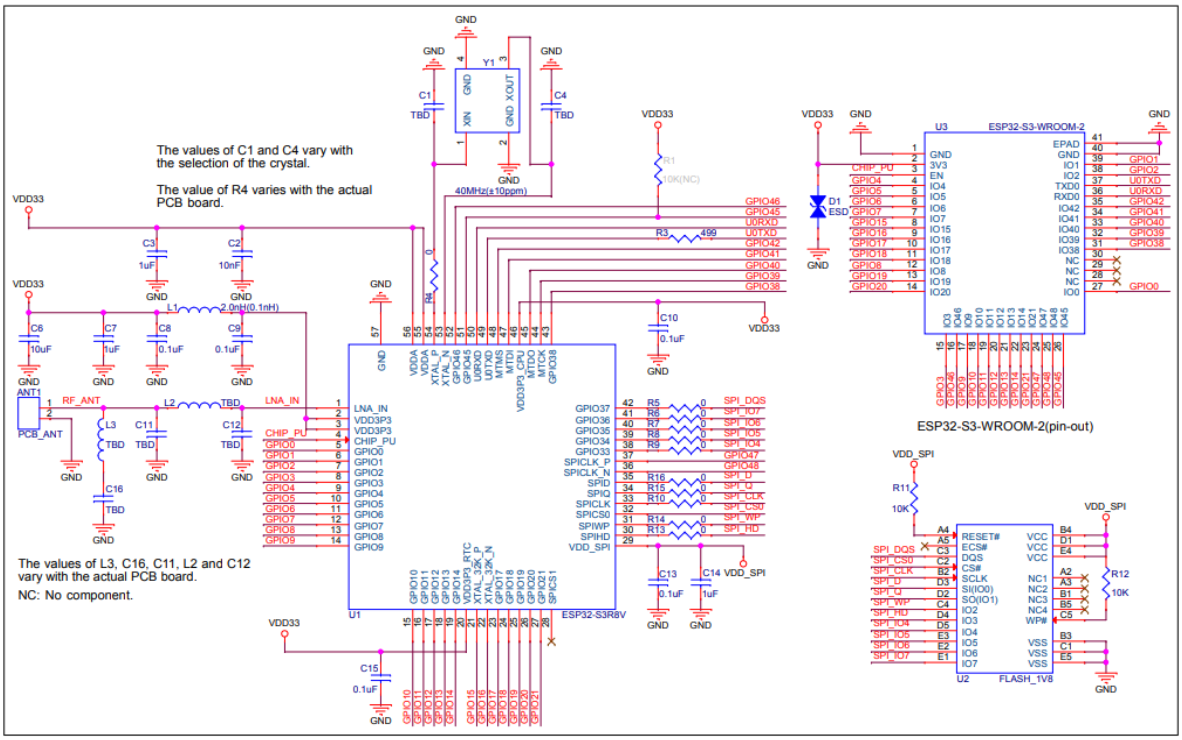


Figure 10: ESP32-S3-WROOM-2 Schematics

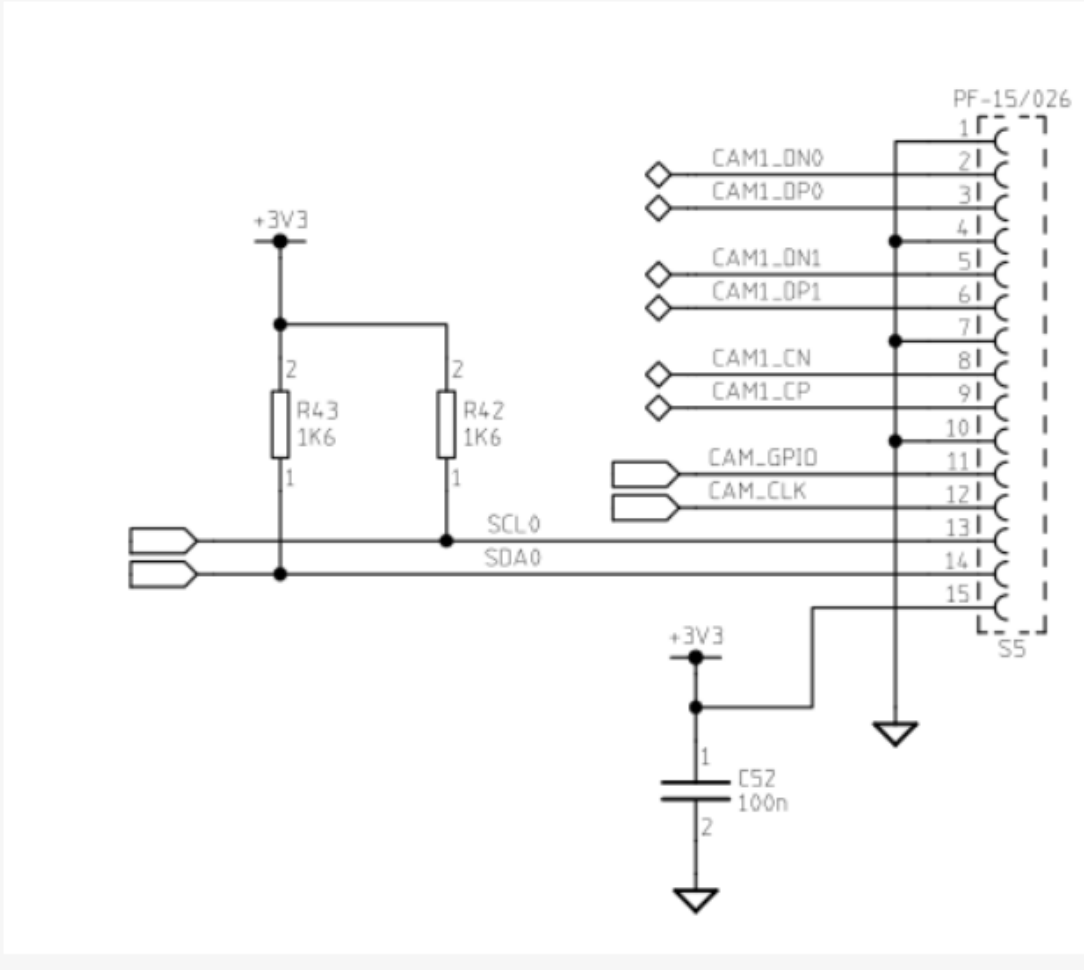


Figure 11: Schematic of the Raspberry Pi CSI camera connector