B31DD Report – Mini Project- Cat Catering Machine

Student Name: Yixin Cao
Student Number: H00315680

1.0 Project introduction

At present, young people are often too busy working overtime to take care of their pets. And in a long trip inconvenient to take the cat, will throw a lot of food. The grain is thus susceptible to damp spoilage. Cats get sick, too. Hence, young people do not have enough time to take care of their pets due to busy work and other reasons, so I developed an Arduino-based cat catering machine. There is timing feeder, and a toy module to the feeder, which can fully help owners to play with their pets. The whole project has four modes.

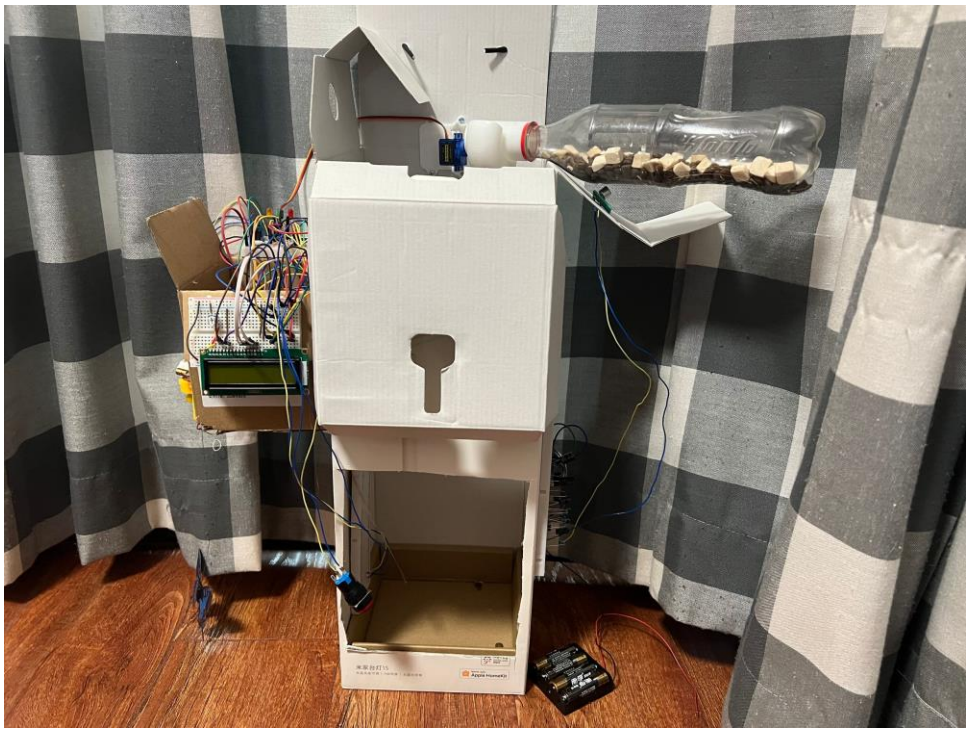


Figure 1 Machine structure overall

Here some components are needed, such as Arduino, motor, servo, and so on. Besides, there are some modules that did not used in class, such as LM393 and laser.

Table 1 Components used

Components	Number
Arduino uno	1
Jumper	∞
Motor	1
laser	3
L293D	1
resister	2
16*2LCD	1

switch	2
Breadboard	4
Servo	1
LM393 Soil moisture detection	1
3D Print items	2

2.0 Functionalities

There are four modes of this machine. The First mode is the playing mode, which is selected according to the pet's toy preferences (After investigation found that many cats and dogs like to chase laser). And because cats and dogs have the nature of hunting, so that this mode also have a module of flying butterfly which can help them train the skills of hunting. This little butterfly is connected to the motor by a wire, and we can use a potentiometer to adjust the speed of the motor and a button to control the direction of rotation. When the motor starts, the module appears to have a small butterfly flying around the device. The module has three laser generators that then shoot lasers at the ground at intervals of 1 seconds to 1s, because the laser is random, it can fully achieve the purpose of playing with pets.

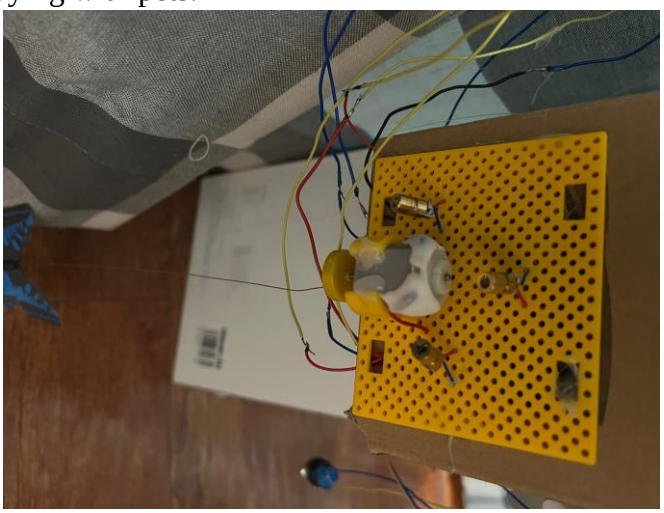


Figure 2 Assembled pattern for modeA

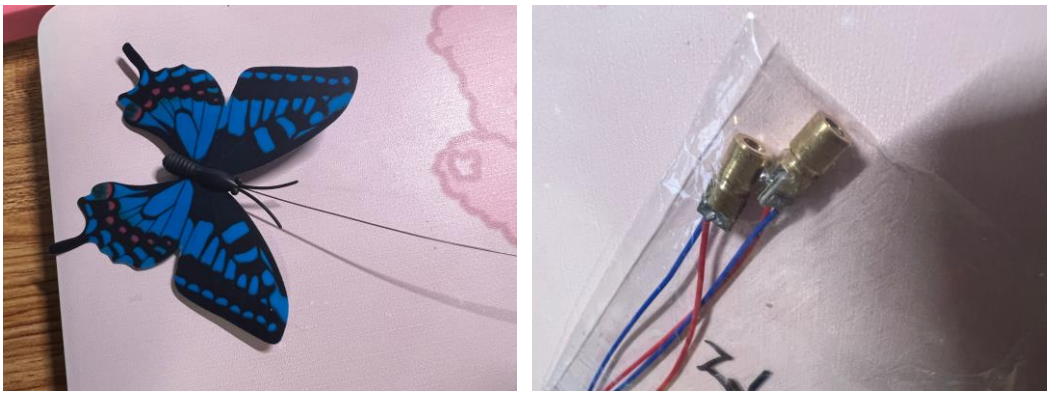


Figure 3 Toys and laser

The second mode is the feeding mode, which can feed the pets at a fixed time every day. Here are 4 time period can be set, that is, one hour, two hours, three hours and four hours. For convenience, here timer is set 1s instead of 1 hour. You can set the interval

time for the feeding module to feed food at regular intervals, the specified interval is 1h, 2h, 3h and 4h, and the set time will be displayed on the 16*2 LCD screen.

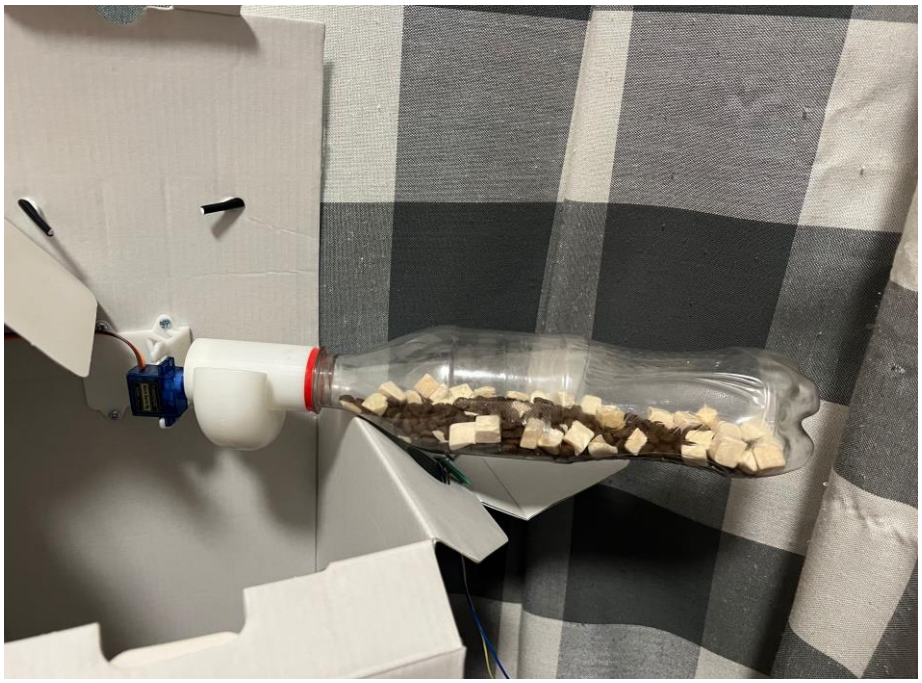


Figure 4 Assembled pattern for modeB

Third mode is noting mode. People can leave a note on LCD to remind something important. For example, leave a note about buying cat food, or clean the bowl of cat.

And fourth mode is humidity detection mode. In this mode, the sensor detects in real time whether the cat's food is wet or not. If the humidity is lower than 500, it is judged to be wet and the LCD warns. This can prevent the cat from eating loose bowels.

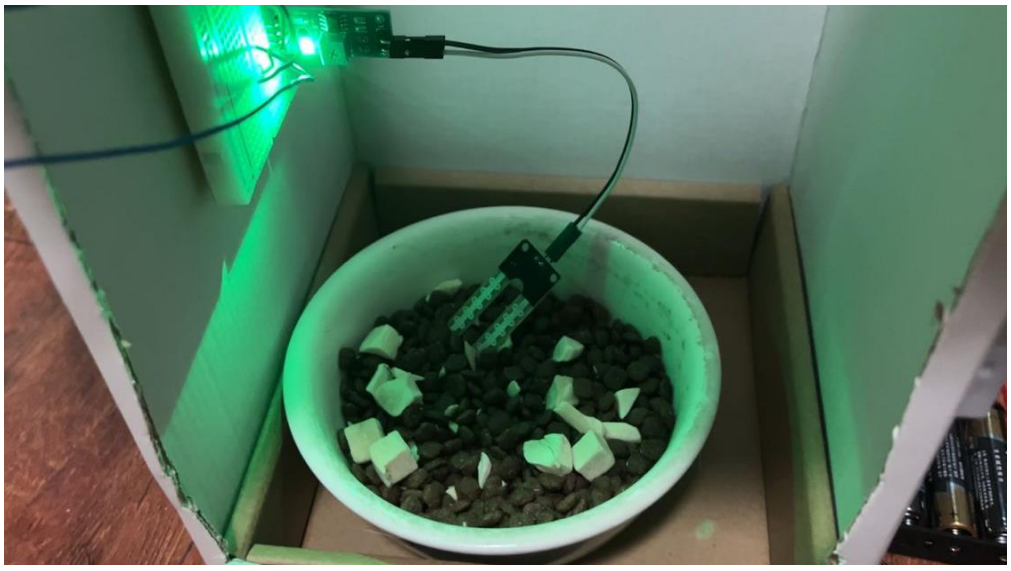


Figure 5 Assembled pattern for modeD

Finally, there is a speaker module, which can amplify your voice at any time. This module is designed by myself, with external power supply. You can speak through the microphone in real time, and the buzzer plays your voice.

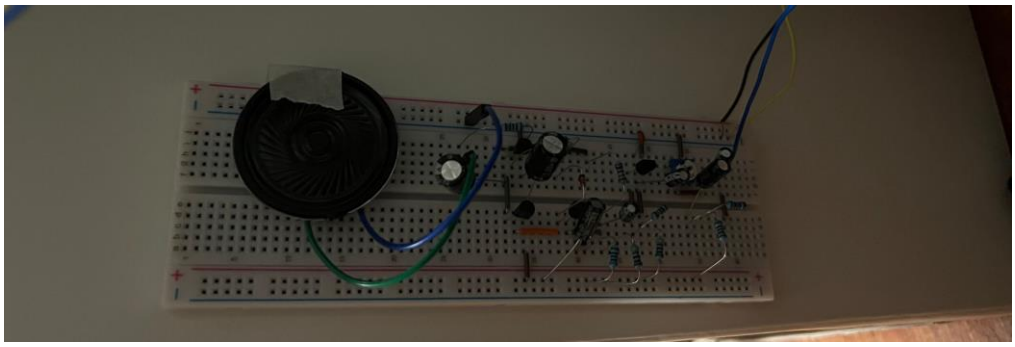


Figure 6 The speaker

Table 2 Function of each mode

Mode name	Mode code	Function
Playing mode	ModeA	Laser and toys play with cat
Feeding mode	ModeB	Feed cat at fixed time
Noting mode	ModeC	Leave note as reminder
Humidity detection mode	ModeD	Detect humidity of cat food
Speaker module	/	Amplify your voice

3.0 Arduino schematics diagram

3.1 Schematics diagram for speaker

R1 provides the bias voltage for the electret simplified MIC. The audio signal is added through C1 to RP1, which is the potentiometer for the volume control and can be used to adjust the output volume. C3 is the audio coupling capacitance. V1, R3 and R4 form a typical voltage parallel negative feedback circuit. After amplification by V1, the audio signal is coupled to the push stage constituted by V2 through C5. R5 and R7 provide a stable operating point for V2, in which R7 is connected to the output midpoint voltage. V3, V4 composed of power amplifier, because V2 and push-pull tube V3, V4 is directly coupled, resistance R7 such connection, plays a deep negative feedback role, so that the circuit can work more stable. At the same time, R6 is the emitter feedback resistance of V2, which further ensures the stability of the static working point of the circuit. C6 is the emitter bypass capacitor of V2, which provides a path for THE AC signal, so that the AC signal is not affected by the lock. VD1, R8 and R9 are the load resistances of the collector V2. And diode VD1 certain temperature compensation function, ensure the circuit work stably.

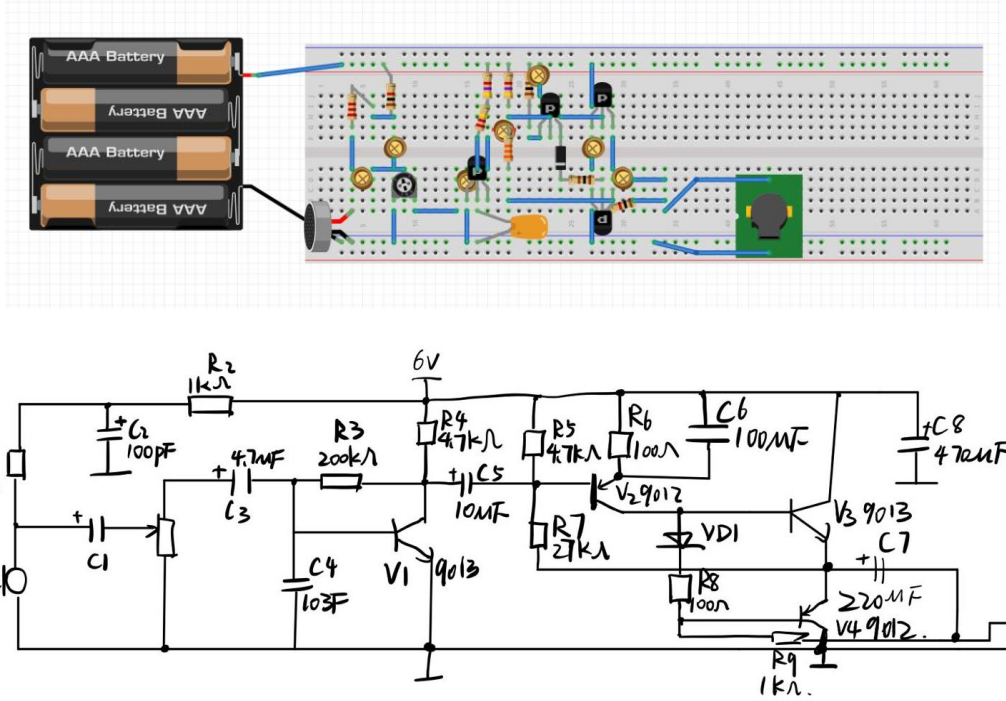


Figure 7 Schematics diagram for speaker

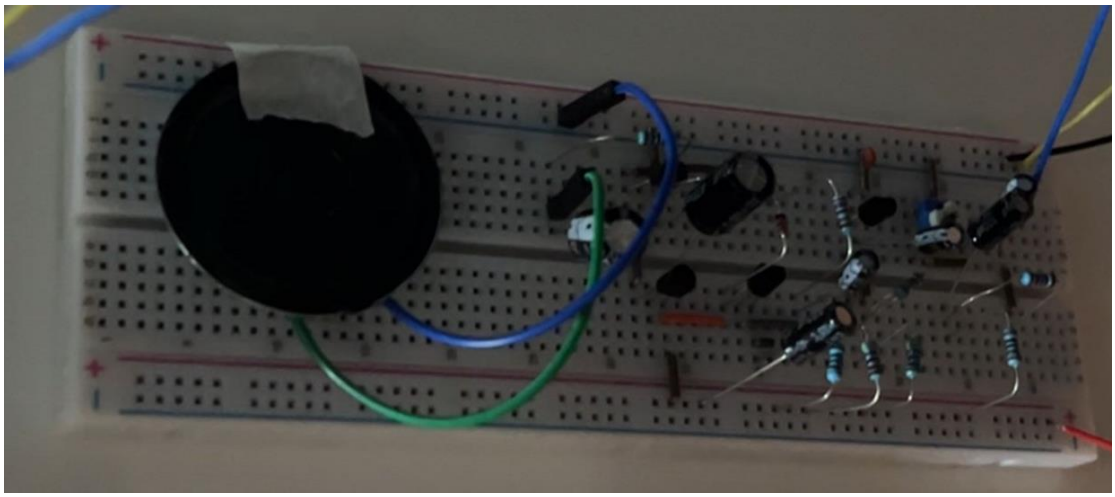


Figure 8 Photo of speaker circuit

Table 3 Components used for speaker

Components	Number
MIC	1
Capacitor 100uF	2
Capacitor 4.7uF	2
Capacitor 470uF	1
Capacitor 220uF	1
Capacitor 10uF	1
Capacitor 103F	1
Resistor 2kΩ	1
Resistor 1kΩ	2
Resistor 200kΩ	1

Resistor 4.7kΩ	2
Resistor 27kΩ	1
Resistor 10kΩ	1
Resistor 100Ω	1
Triode 9013	2
Triode 9012	2
Diode 1N4148	1
Buzzer speaker	1

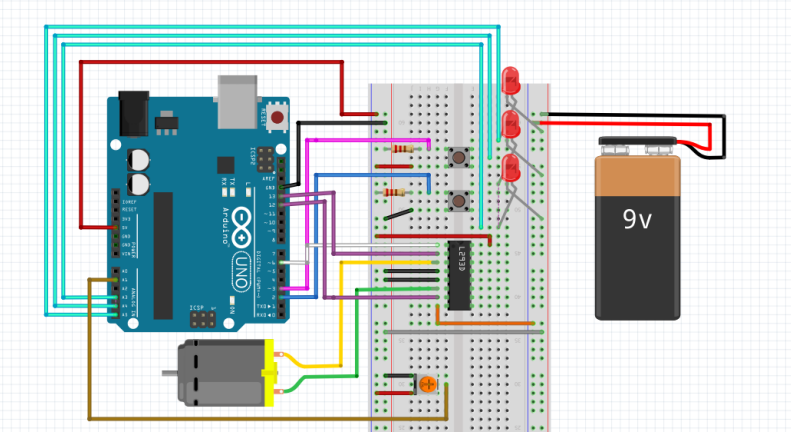
3.2 Schematics diagram for ModeA

For mode A, using two switches and a potentiometer to control the overall program. The switch connected with Pin2 controls the direction of motor rotation. The switch connected with Pin3 controls the exit procedure. The potentiometer controls the motor speed. And use A3, A4, and A5 ports to control the light and dark of the laser. Lasers are replaced by LEDS here. ADC1 is used to control speed. So, potentiometer is connected to A1.

Table 4 Pins for H-bridge

H-bridge Pins number	1	2	7	3	6	16	8	4 5	//////
Pin means	1,2 EN	1A	2A	1Y	2Y	Vcc1	Vcc2	Ground (Pin4 Pin5)	Common ground
Arduino pins	Pin6 (PD6) White	Pin13 (PB5) Purple	Pin12 (PB4) Purple	DC motor Yellow	DC motor Green	5V Red	9V Red	GND Black	9V and 5V Grey

The Pin1 of the H bridge is connected to the Arduino's PD6 because the PD6 is the output of the PWM wave. Pin4 and Pin5 are connected to the ground of the Arduino. In addition, the Arduino GND is connected to the battery GND. Pin16 is connected to an Arduino 5V pin, and Pin8 to a 9V battery. Pin7 and Pin2 control the direction of the motor and connect PB4 and PB5. Pin6 and Pin3 are the average terminals of the dc motor, so they are connected to the DC motor. Pin16 is connected to an Arduino 5V pin, and Pin8 to a 9V battery.



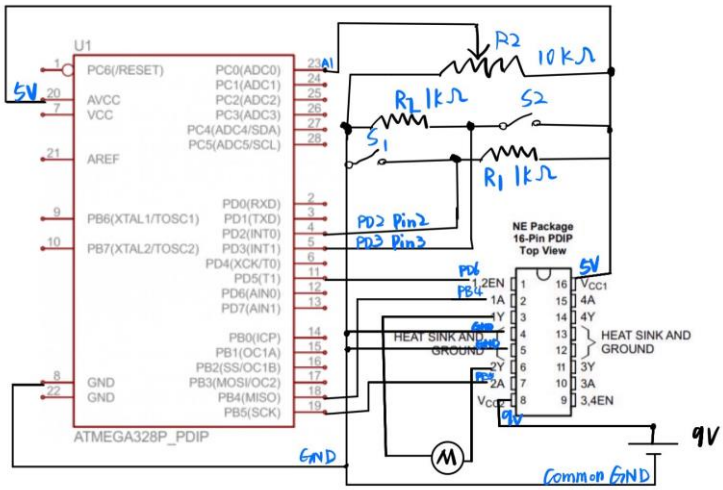


Figure 9 Schematics diagram for ModeA

Table 5 Pins for modeA

Output	Pin6 (PD6) White	Pin12 (PB4) Purple	Pin13 (PB5) Purple
Output laser	A3(PC3) Light blue	A4(PC4) Light blue	A5(PC5) Light blue
Interrupt pin	Pin2 (PD2) Blue		
Input	Pin3 (PD3) Pink	A1(PC1) Brown	
Power	5V Red	GND Black	
Resistor	1k Ω for switch		

3.3 Schematics diagram for Mode

For mode B, using two switches to control the overall program. The switch connected with Pin2 controls the set time of servo rotation. The switch connected with Pin3 to controls the change time and quit the mode.

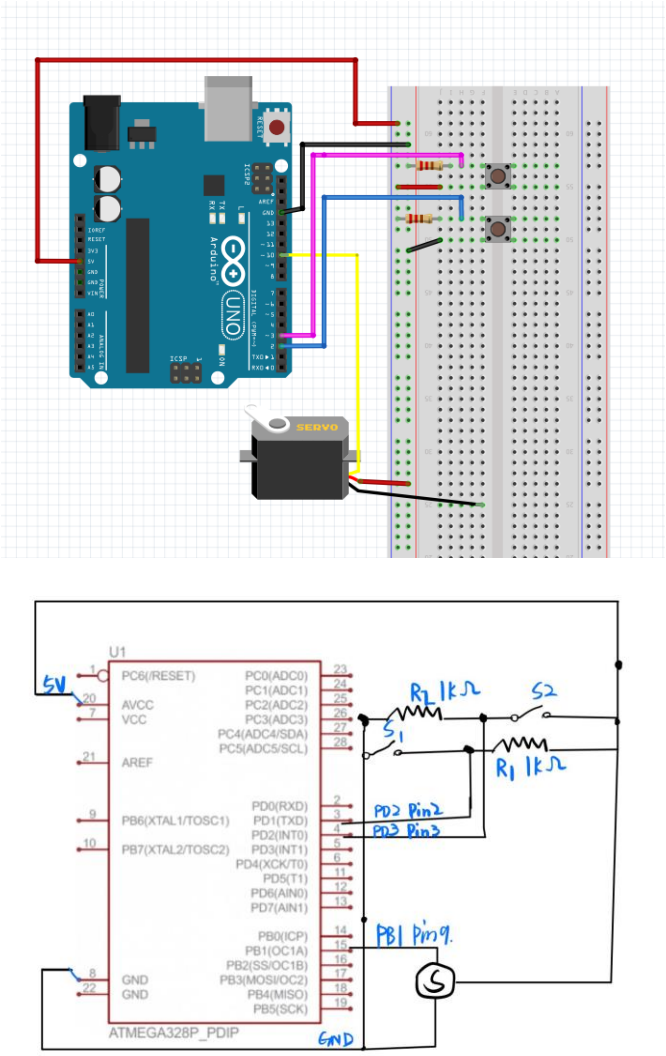


Figure 10 Schematics diagram for ModeB

Table 6 Pins for modeB

Output	Pin9 (PB1) Yellow	
Interrupt pin	Pin2 (PD2) Blue	
Input	Pin3 (PD3) Pink	
Power	5V Red	GND Black
Resistor	1k Ω for switch	

3.4 Schematics diagram for ModeC

For mode C, using one switch to control the overall program. The switch connected with Pin2 controls to quit the mode. Here using 4-bit mode.

For LCD, the D4-D7 pins on the LCD are used to send and receive data. So, they connect PD4, PD5, PD7, and PB3 on Arduino, respectively. LCD grounding Connects to Arduino GND. Vss and Vcc can power the LCD and connect GND and 5V in Arduino. V_{EE} is a pin that uses the potentiometer's voltage to control LCD contrast. This voltage is between 0 and 5 volts. The RW pins control whether the Arduino sends or receives data, and connect to GND so that it always receives data. Enable pins are used to activate pin receive and send. So, connect it with PB2 of the Arduino. The RS

pin controls register selection, i.e.RS=1, loaded data is stored in the data register, RS=0, loaded data is stored in the command register. This is connected to PB0 on Arduino.

Table 7 Pins for ModeC

LCD Output	Pin7 (PD7) Yellow	Pin11 (PB3) Yellow	Pin5 (PD5) Yellow	Pin4 (PD4) Yellow
Output for enable and write	Pin8 (PB0) Green		Pin10 (PB2) Blue	
Power	5V Red		GND Black	
Interrupt pin	Pin2 (PD2) Blue			
Potentiometer	5V Red	GND Black	V_{EE} Pink	

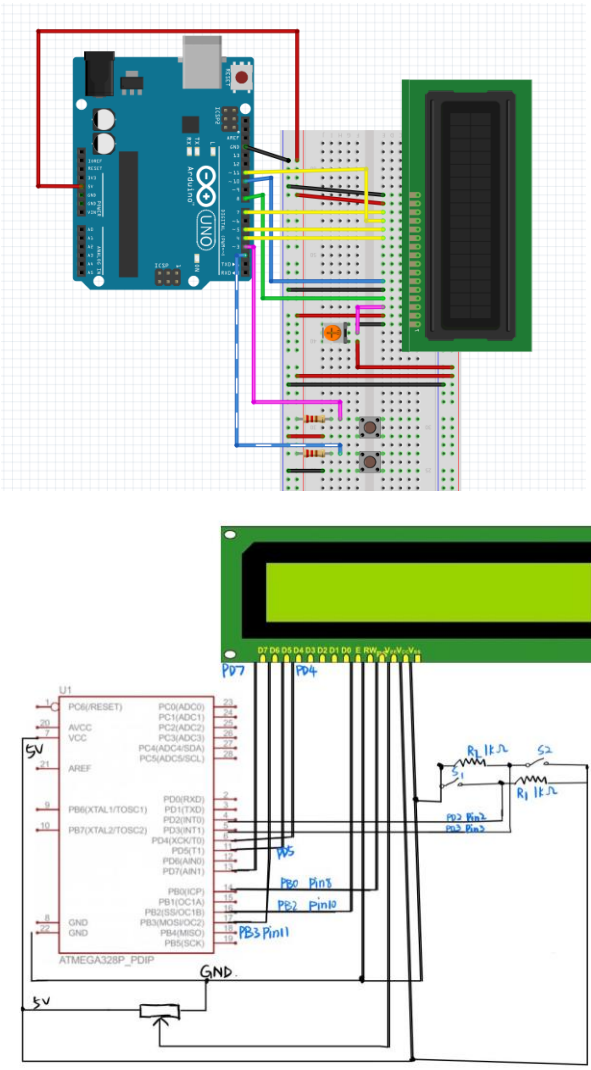


Figure 11 Schematics diagram for ModeC

Table 8 Pins for LCD

Vss	Vcc	V_{EE} /Vo	RS	RW	Enable	LED(GND)	LED(5V)

GND Black	5V Red	Potentiometer Pink	Pin8 (PB0) Green	GND Black	Pin10 (PB2) Blue	GND Black	5V Red

3.5 Schematics diagram for ModeD

For mode D, using one switch to control the overall program. The switch connected with Pin2 controls quit the mode. Using ADC0 to get data from LM393.

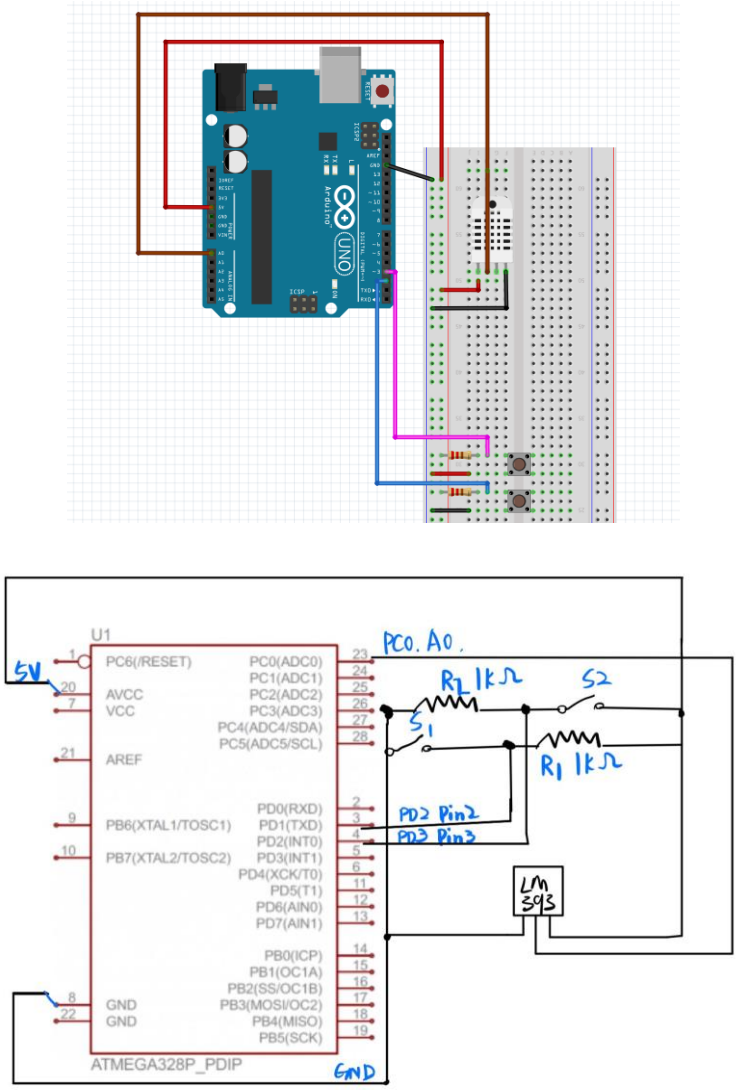


Figure 12 Schematics diagram for ModeD

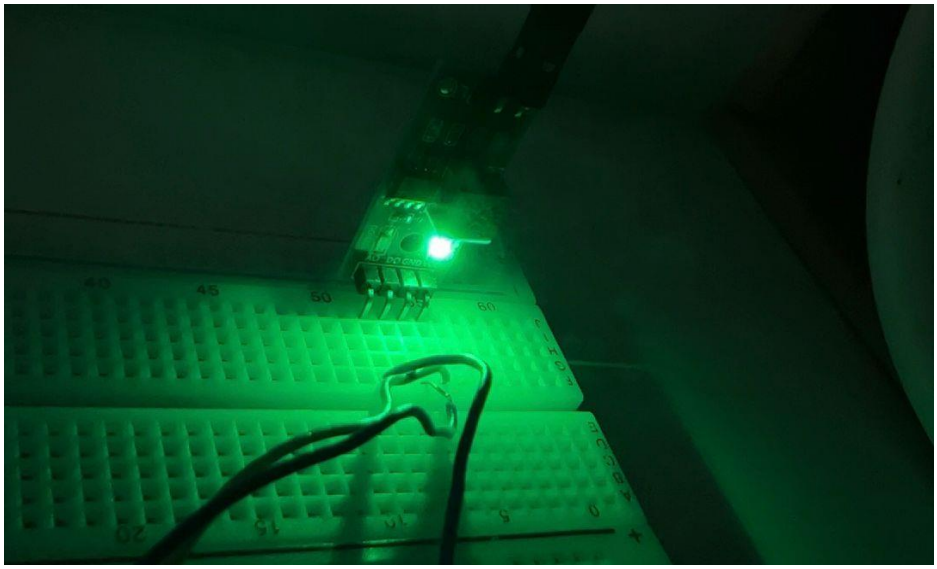


Figure 13 Photo of LM393 circuit

Table 9 Pins for modeD

Interrupt pin	Pin2 (PD2) Blue	
Input	Pin3 (PD3) Pink	A0 (PC0) Brown
Power	5V Red	GND Black
Resistor	1k Ω for switch	

3.6 Schematics diagram for Overall structure

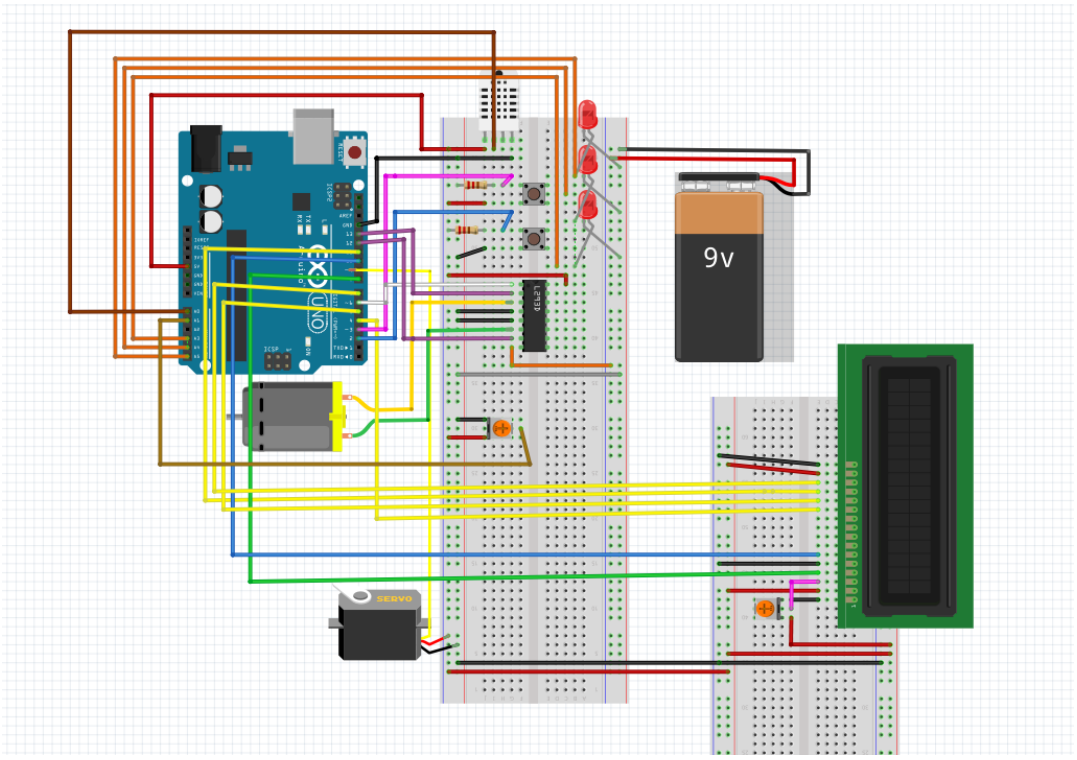


Figure 14 Schematics diagram for Overall structure

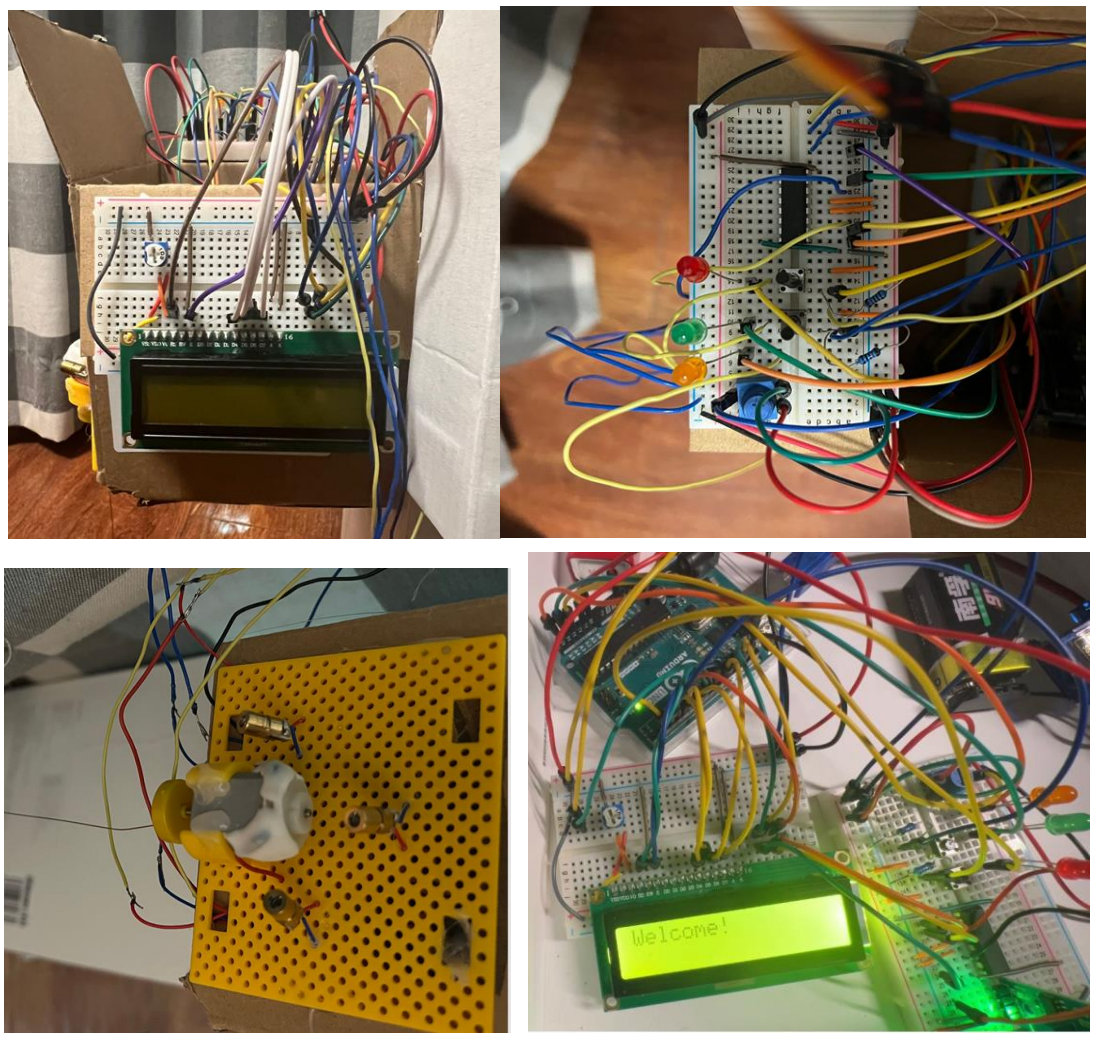


Figure 15 Photo of Overall structure circuit

3.7 The mechanical structure

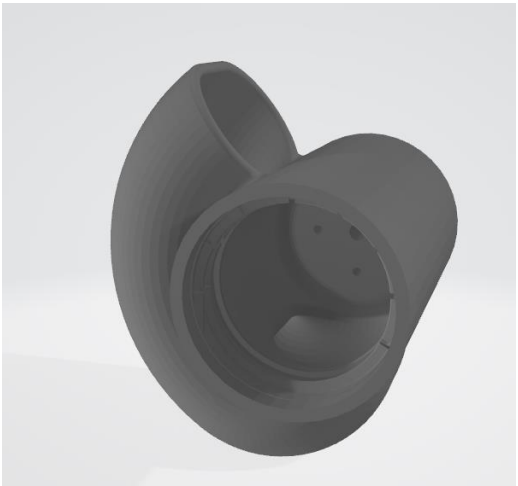


Figure 16 3D model for servo

The structure can be nested on top of a normal beverage bottle, which greatly reduces the cost of replacing the food box and can automatically drops food from the device when the servo motor works.

The whole machine is attached to a cardboard box. The motor and laser are fixed to the hole plate (See 3.7). Other circuits are fixed to the breadboard. There was an opening at the bottom of the box for the cat to eat in.

And switches are changed into switches has long button. This is very convenient for user to press.

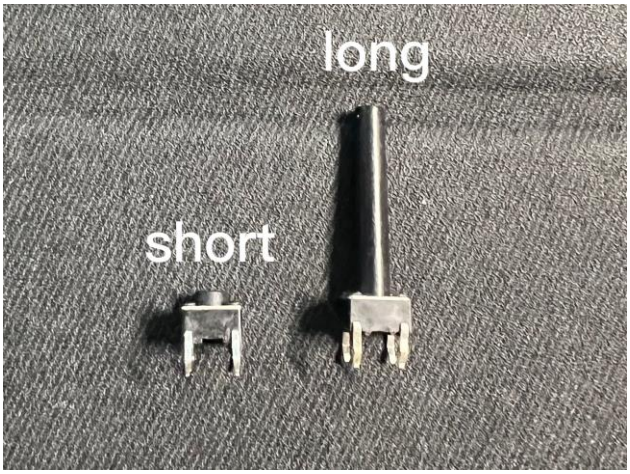


Figure 17 Switches used

The motor and laser are fixed to the hole plate. The laser has two wires, one is GND and the other is positive.

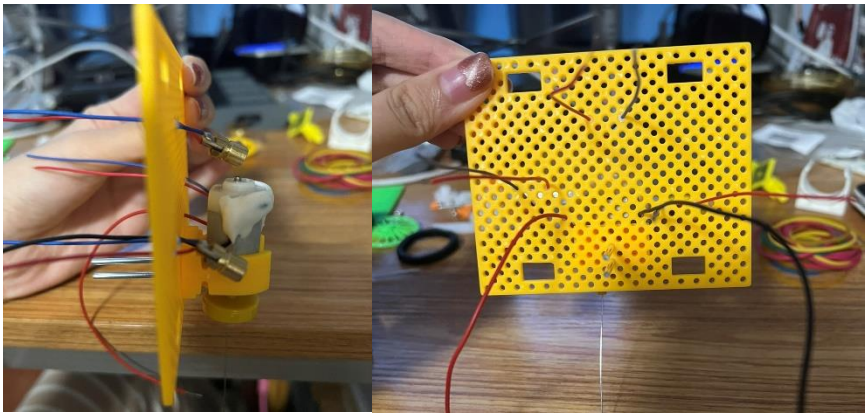


Figure 18 Motor is fixed on a board

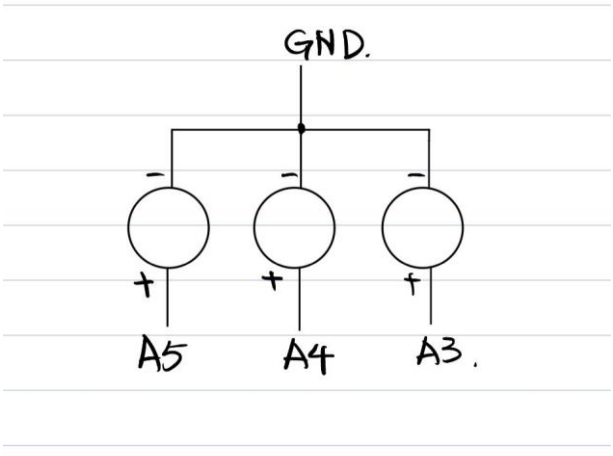


Figure 19 Pins for lasers

As the wire was not long enough, part of the circuit and parts were welded.

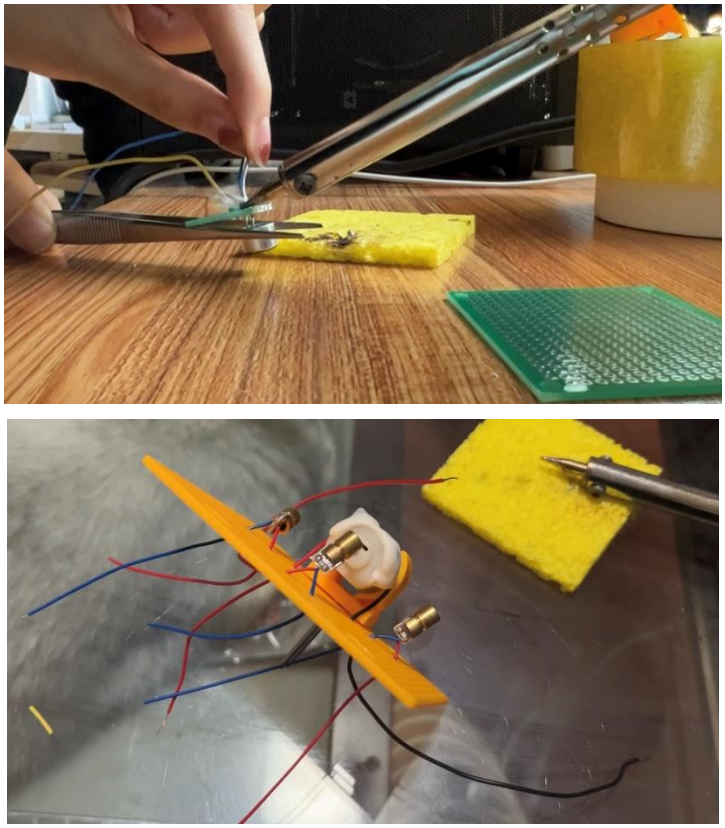


Figure 20 The welding equipment

4.0 Explanation of techniques used

Here, all the techniques that learned at class are used. Input and output are used to connect Arduino and other parts. ADC0 and ADC1 are used to get data from LM393 and potentiometer, respectively. Motor is used in ModeA to rotate the butterfly to play with cat. Servo is used in ModeB to rotate the bottle to feed cat. Interrupt is used to quit mode and stop the machine for all modes, which will show in a single module in this report. USART is used in each mode to communicate with PC and send some reminder to PC. This will also be shown in a single module. For LCD, it is used in each mode to remind which mode that is running. And especially in ModeC, LCD can get string from PC and display it. For timer, it is used in ModeB to achieve some delay. Here are details about them.

4.1 Explanation of techniques used in ModeA

In ModeA, motor is controlled by one switch and a potentiometer. This is quite similar to Lab2. Switch is used to change direction and potentiometer is used to change speed. Pin6 is used to generate PWM wave. So, OCR0A is used. Here are some formular to calculate OCR0A to control the speed. Here non-inverting mode and fast PWM mode are used. So, the duty cycle can be calculated:

$$Duty\ Cycle\ = \frac{OCR0A+1}{256} \times 100\%$$

N: prescaler can be calculated, if generate 61.04Hz PWM wave.

$$F_{oco} = \frac{f_{clk}}{N(256)} = 61.04Hz = \frac{16MHz}{N(256)}$$

$$N = \frac{16MHz}{61.04\,Hz \times 256} \approx 1024$$

For 1024 prescaler and timer0, set WGM02=0, CS02=1 CS01=0 CS00=1. So, TCCR0B = 00000101=0x05.

For non-inverting mode, set COM0A1=1, COM0A0=0. So, TCCR0A = 10000011=0x83. For fast PWM mode, set WGM02=0, WGM01=1, WGM00=1.

And here ADC1 is used to set the OCR0A at any time when in mode A. PinA1 (ADC1) gets the voltage value of potentiometer. For this 10-bit convert, the resolution of ADC1 can be obtained.

$$Resolution = \frac{5V}{1024} \approx 0.0049\,V\ per\ unit = 4.9\,mV\ per\ unit$$

The OCR0A need in the range of 0-255. So, get the factor to use adjust the range. The ADC1 initial range is:

$$initial\ range = 0 - 1023$$

$$N\ factor = \frac{1023}{255} \approx 4.01 = 4$$

Besides, to change direction of motor, using H-bridge to control. When pressing the button, detect which direction pin is on. Turn off the current direction pin by setting it Low, and turn on the direction pin which is off now by setting it High.

And here three laser is used to play with cat. A3, A4 and A5 are connected with these three lasers. And to make cat more confused and happier, these three pins' output are random. Here using rand() function to generate random number. And use srand() function to select a seed for random generating. This can make sure the random number is not a pseudo random sequence.

4.2 Explanation of techniques used in ModeB

For ModeB, timer and servo are used. Timer is used to achieve delay. Here a delay for 1s, 2s, 3s and 4s are used. Because timer is used to generate PWM wave, timer0 is used to set delay. Here are details for calculated.

Step1: Calculate period for original.

$$Period_{original} = {}^{1}/_{Frequency} = {}^{1}/_{XTAL} = {}^{1}/_{16MHz}$$

Step 2: Calculate period when using 1024 prescaler. For 1024 prescaler, set TCCR1B=0x05=5. And normal mode is used, so set TCCR1A=0x00=0.

$$Period_{actual} = Period_{original} \times 1024 = {}^{1024}/_{16MHz} = 64us$$

Step 3: Calculate the number of machine cycles for delay 1 ms.

$$number\ of\ machine\ cycles\ for\ 1\ ms = {}^{1\ ms}/_{Period_{actual}} = 15.625$$

Step 4: Calculate the time delay when number of machine cycles is 255

$$\text{Time delay max} = Period_{actual} * 255 = 0.01632s$$

Step 5: Calculate loop:

$$\text{Loop for 1s delay} = 1s/\text{Time delay max} \approx 61.27 \approx 61$$
$$\text{Loop for2s delay} = 2s/\text{Time delay max} \approx 122.55 \approx 123$$
$$\text{Loop for 3s delay} = 3s/\text{Time delay max} \approx 183.82 \approx 184$$
$$\text{Loop for 4s delay} = 4s/\text{Time delay max} \approx 245.10 \approx 245$$

Table 10 Loop for delay

Time delay	Loop number
1s	61
2s	123
3s	184
4s	245

For servo, set it to rotate 180 degrees at each time. And hold this position for 1 second, then back to 0 degree.

Using fast PWM mode 14 and timer1 control servo system. ICR1 has a maximum value of 0xFFFF. Here ICR1 is set to 40000. Therefore, the frequency wave generated in fast PWM mode can be calculated:

$$F_{oco} = \frac{f_{clk}}{N(40000)}$$

Duty cycle of PWM can be calculated:

$$Duty\ Cycle = \frac{OCR1A}{C_{ICR}} \times 100\% = \frac{OCR1A}{40000} \times 100\%$$

Theoretically, the angle corresponding to the control servo angle is as follows.

Table 11 The parameters of servo

Angle(degrees)	-90(0)	-45(45)	0(90)	45(135)	90(180)
Pulse width	1ms	1.25ms	1.5ms	1.75ms	2ms
Duty cycle	2.5%	5%	7.5%	10%	12.5%

The servo system requires a PWM wave with a period of 20ms. So, T is equal to 20 milliseconds. Here are the steps to calculate the register values.

Find N: prescaler:

$$ICR1 = 40000$$
$$F_{oco} = \frac{f_{clk}}{N(40000)} = \frac{16MHz}{N(40000)} = \frac{1}{T} = \frac{1}{20ms} = 50Hz$$
$$N = \frac{16MHz}{50\ Hz \times 40000} = 8$$

Find OCR1A for -90 and 90 degree:

$$Duty\ Cycle\ = \frac{OCR1A_{-90}}{C_{ICR}} \times 100\% = \frac{OCR1A_{-90}}{40000} \times 100\% = 2.5\%$$

$$OCR1A_{-90} = 1000$$

$$Duty\ Cycle\ = \frac{OCR1A_{90}}{C_{ICR}} \times 100\% = \frac{OCR1A_{90}}{40000} \times 100\% = 12.5\%$$

$$OCR1A_{90} = 5000$$

Table 12 Value of OCR1A

Angle(degrees)	-90(0)	90(180)
OCR1A	1000	5000
OCR1A offset	1100	4900

Moreover, switch is used to set time. When pressing switch, add one to the global variable C. And using C to select a case from switch function to achieve the time delay change. There are four length of time delay to select. So, only need four different C. Set C equal to 0 when C is larger than 4.

4.3 Explanation of techniques used in ModeC

For ModeC, LCD and USART are used. LCD is used to display the note from PC. USART are used to set communication between PC and Arduino. Here 4-bit mode of LCD is used to save the pins.

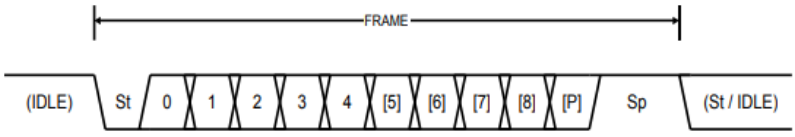
Here Baud rate is 9600 bps. Asynchronous Normal mode (U2Xn = 0) is used. Now BAUD_PRESCALLER can be obtained.

$$Desired\ Baud\ rate\ = \frac{F_{osc}}{(X + 1) \times 16}$$

X is the value of UBRR register that is USART baud rate register.

$$X\ = \frac{F_{osc}}{\text{Baud rate}\ \times 16} - 1 = \frac{16\ MHz}{9600\ \text{bps}\ \times 16} - 1 \approx 103.1667 \approx 103$$

$$F_{osc} = 16\ MHz$$



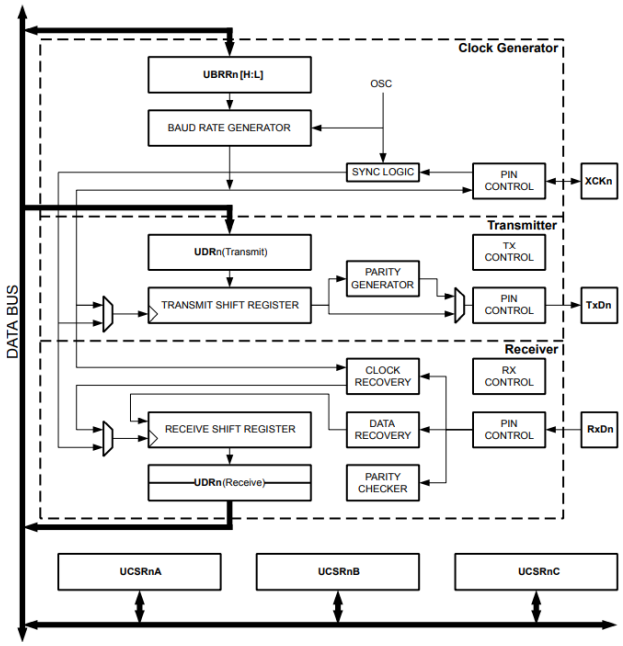


Figure 21 USART Block

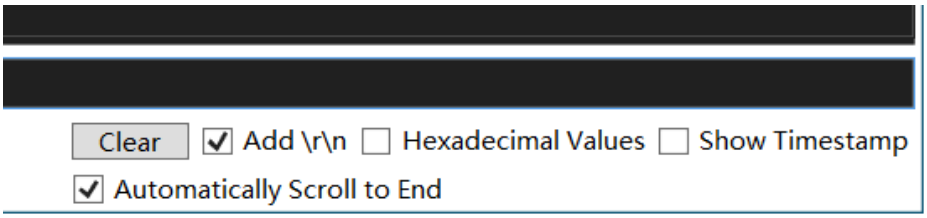


Figure 22 Data Visualizer setting

Select \r\n to the sent string, so that stop character can be detected.

Table 13 Meanings of \n and \r

Character	ASCII code	Meaning
\n	10=0x0a	LF/NL(Line Feed/New Line)
\r	13=0x0d	CR (Carriage Return)

4.4 Explanation of techniques used in ModeD

For ModeD, LCD, ADC0, LM393 and USART are used. LCD is used to display the humidity value from LM393. USART are used to set communication between PC and Arduino. Here 4-bit mode of LCD is used to save the pins. LM393 is used to detect the humidity of cat food. ADC0 is used to get data from LM393.

When using ADC0, change MUX0 of setting.

```
ADMUX |= (1 << REFS0) | (0 << MUX0); //reference voltage on AV_CC, and use ADC1
ADCSRA |= (1 << ADPS1) | (1 << ADPS0); //ADC clock prescaler / 8
ADCSRA |= (1 << ADEN); //enables the ADC
```

PinA0 (ADC0) gets the voltage value of potentiometer. For this 10-bit convert, the resolution of ADC1 can be obtained.

$$Resolution = \frac{5V}{1024} \approx 0.0049\ V\ per\ unit = 4.9\ mV\ per\ unit$$

The data of humidity has no need to adjust the range. So, use the ADC1 initial range 0-1023. When humidity is larger than 600, that means the cat food is dry. When humidity is smaller than 600, that means the cat food is wet, and Arduino will send "wrong" to LCD.

In terms of LM393, this is a sensor that detects humidity. It has its own potentiometer to regulate sensitivity. It determines humidity by measuring the electrical conductivity of the two probes. Because more water makes it easier to conduct electricity. It has four pins. One is Vcc, a 5V connected to Arduino. One is the GND connection ground. One is D0, which is a digital output pin that connects the alarm or indicator light. The last one is the A0 pin, which is the output voltage pin, that is, the output data pin. This pin needs to be connected to the ADC0 of Arduino.

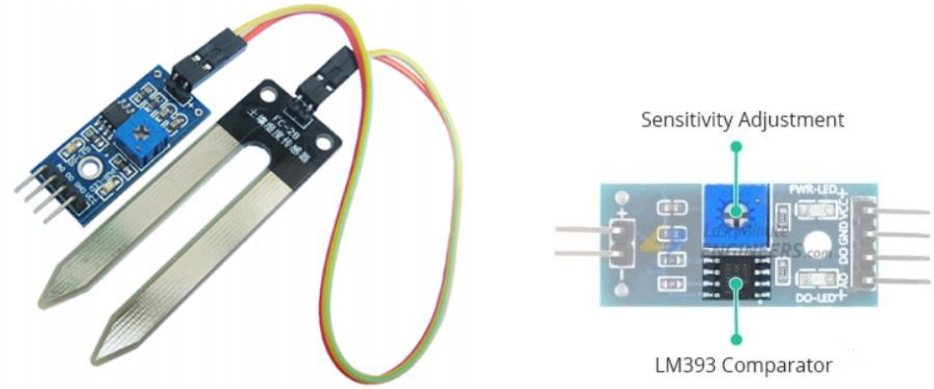


Figure 23 LM393A

4.5 Explanation of interrupt

In each mode, interrupt INT0 is used to quit the mode at any time. So, a pull-up register is used and switch is used. When press interrupt button, the LCD will be cleared and delay for 1s to reset the whole program. Then back to while(1) and waiting for mode select.

4.6 Explanation of USART

USART in this project has three functions.

Firstly, select mode. When sending "A", the code will get into the ModeA. When sending "B", the code will get into the ModeB. When sending "C", the code will get into the ModeC. When sending "D", the code will get into the ModeD.

Secondly, transmit information to PC. When get into a mode, it will send ModeX to PC. For example, when get into ModeA, it will send "ModeA" to PC. And when quit a mode, it will also send "Quit" to PC. And especially for ModeC, when sending a note from PC, it will send "Receive message" to PC, and send "Change message" later.

Thirdly, when in ModeC, USART is used to transmit string from PC to LCD.

4.7 Pseudo-code

Here is the pseudo-code:

Pseudo-code for Task2 switch control speed

```
1.  #define D4-D7 RS EN
2.  #define BAUDRATE BAUD_ PRESCALLER
3.
4.  in main event;
5.  set_ output(PORTD and PORTB0 1);
6.  USART_init();
7.
8.  while(1){
9.      if(A){
10.       while(1){
11.         Lcd4_Write_String(ModeA);
12.         Motor rotate;
13.         Speed change by potentiometer;
14.         If switch on, change direction;
15.         If interrupt on, Quit;
16.       }
17.     if(B){
18.       while(1){
19.         Lcd4_Write_String(ModeB);
20.         Set time by press button;
21.         servo rotate;
22.         Delay by timer0;
23.         If interrupt on, Quit;
24.       }
25.     if(C){
26.         Lcd4_Write_String(ModeC);
27.       while(1){
28.           Lcd4_Clear();
29.           stop USART send and receive;
30.           Lcd4_Init();
31.           Lcd4_Set_Cursor(1,3);
32.           Lcd4_Write_String(string);
33.           USART_init();
34.           transform(string);
35.           USART_putstring("Received data\n");// tips for terminal
36.           _delay_ms(1000);
37.         If interrupt on, Quit;
38.       }
39.     if(D){
40.       while(1){
41.         Lcd4_Write_String(ModeD);
42.         Get ADC0 value from LM393;
43.         If <600, send wrong to LCD;
```

```
44.             If interrupt on, Quit;
45.           }
46.
47. in interrupt event 0;
48. if (INT0){ //switch on
49.             Lcd4_Clear();
50.             delay();//delay 0.5s to prevent error caused by key jitter
51. }
52. return main event while(1);
```

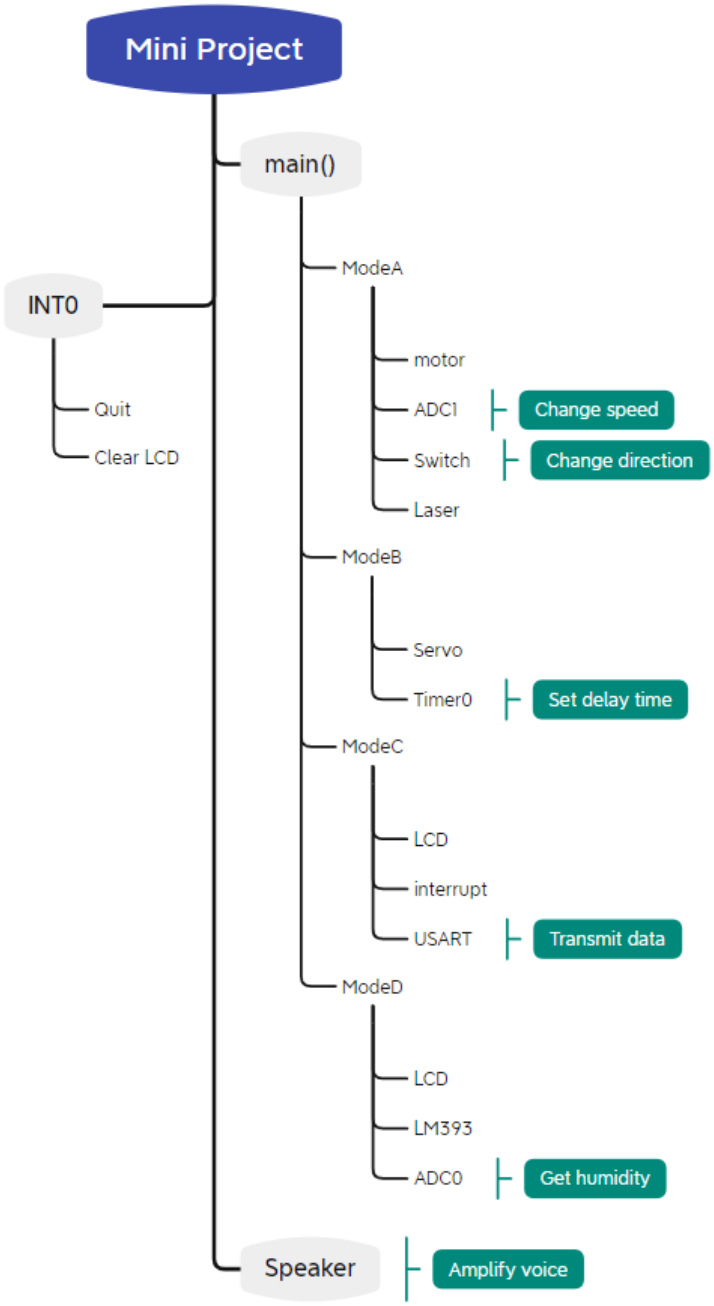


Figure 24 Flowchart of whole structure

5.0 Demo results (e.g., screenshots)

5.1 Demo results for LCD

When entering into each mode, LCD will display.

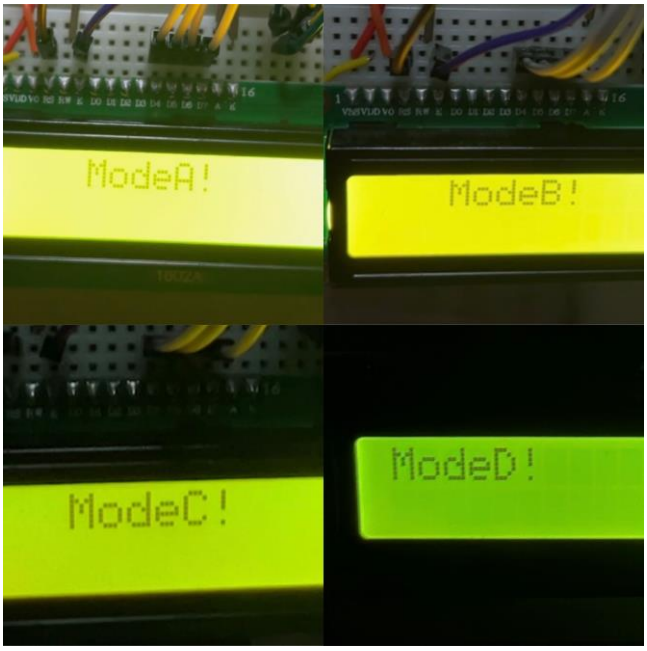


Figure 25 LCD display Mode

When in ModeB, selecting time, LCD will display.

Figure 26 LCD display in ModeB

When in ModeC, sending note, LCD will display.

Figure 27 LCD display in ModeC

When in ModeD, getting data from LM393, LCD will display.

Figure 28 LCD display in ModeD

5.2 Demo results for ModeA

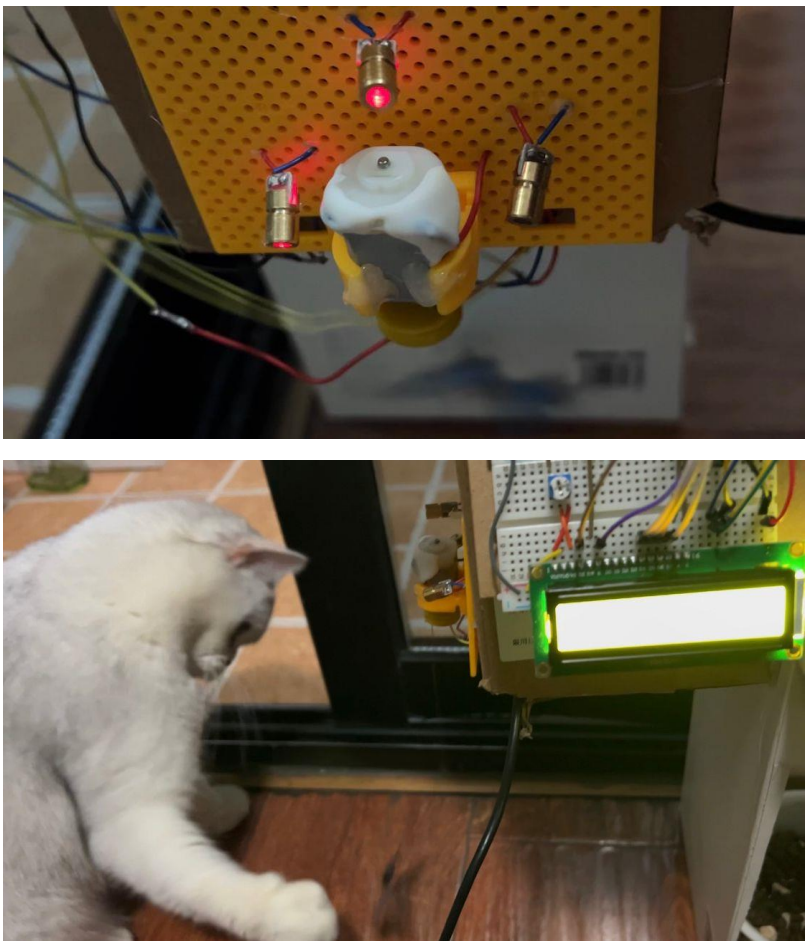


Figure 29 Demo results for ModeA

5.3 Demo results for ModeB

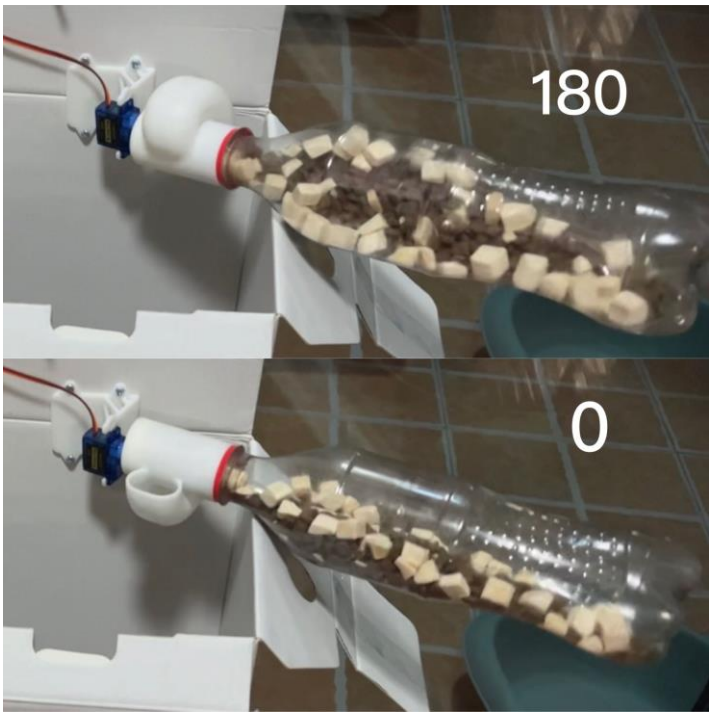


Figure 30 Servo rotate

Demo results for ModeC and ModeD are shown in LCD result in 5.1.

5.4 Demo results for USART

When connecting Arduino, it will send “welcome to use” to PC. When entering into a mode, Arduino will send message to PC. When quitting a mode, Arduino will also send message to PC.

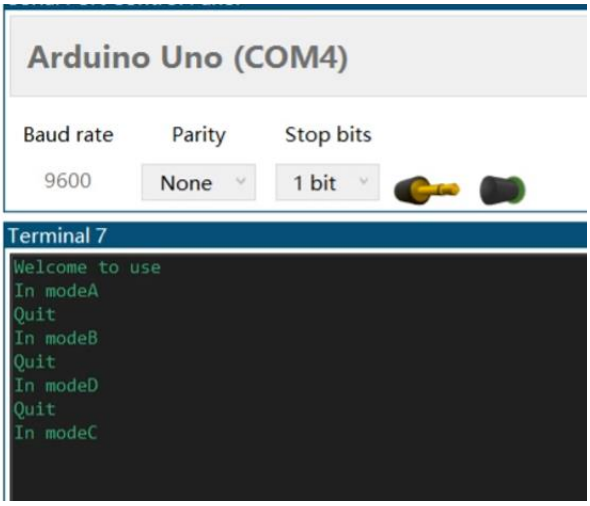


Figure 31 USART

When sending note from PC to Arduino, the Arduino will send reminder to PC to inform that message is received.

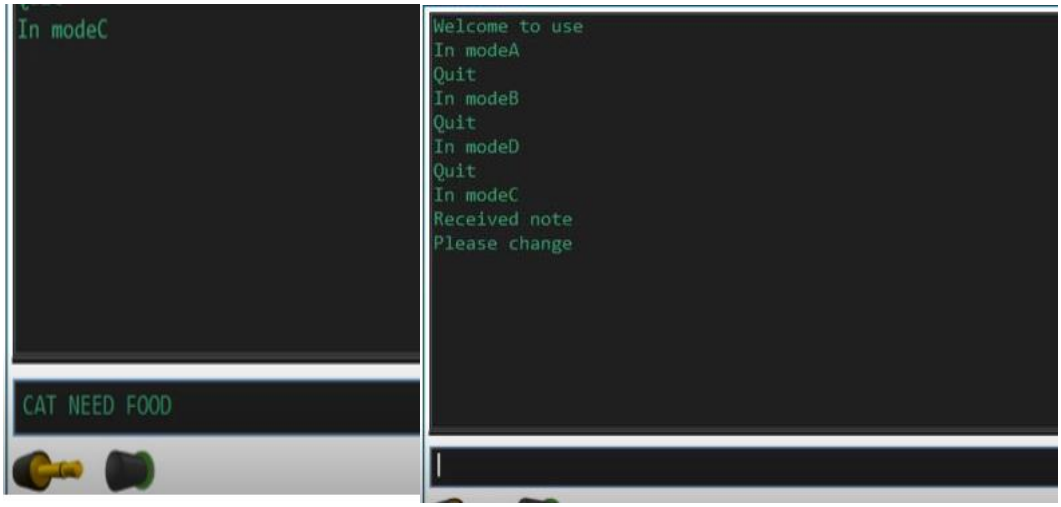


Figure 32 USART in ModeC

5.5 Method of use

Here four switches are used in the whole program. Here is picture of them. For S4, it is used to set the contrast degree of LCD.

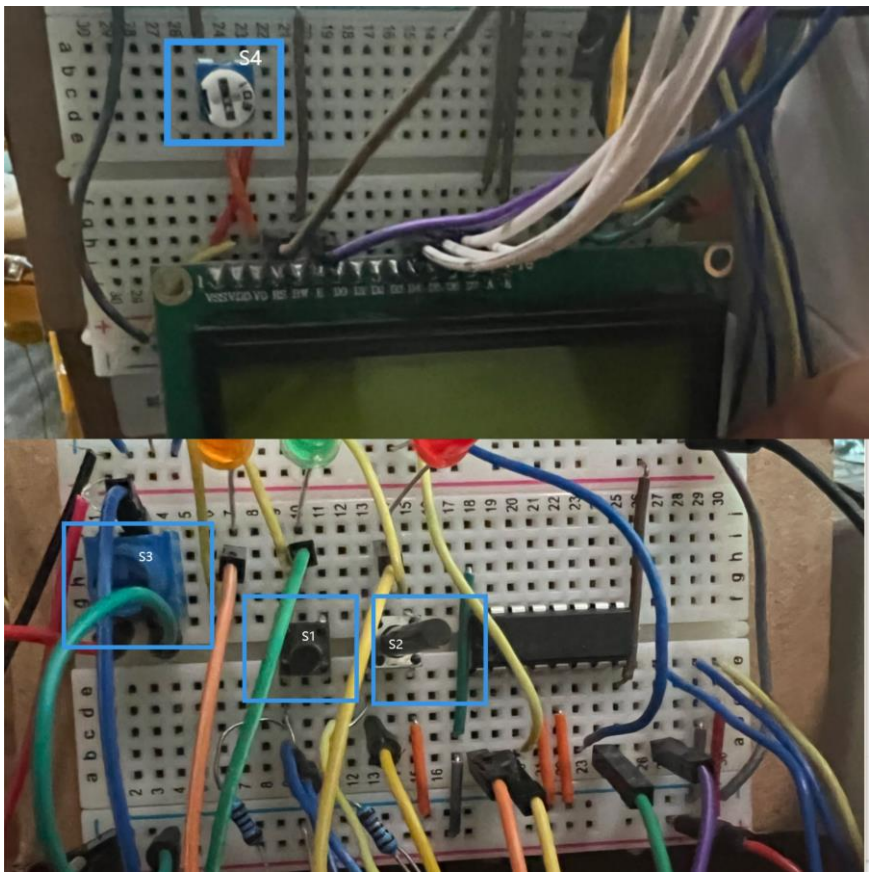


Figure 33 Switches control whole system

In ModeA, S1 is used to change direction of motor. And S3 is used to set the speed of motor. S2 is used to quit the mode.

In ModeB, S1 is used to change set time of timer0. And S3 is not used here. S2 is used to quit the mode.

In ModeC, S1 is not used here. And S3 is not used here. S2 is used to quit the mode.

In ModeD, S1 is not used here. And S3 is not used here. S2 is used to quit the mode.

6.0 Performance evaluation

Overall, the device conforms to the expected design and achieves the functional objectives, but there are still some areas that can be improved.

1. In the later stage, WiFi module can be added, and home intelligent LAN can be added, so that the device can be remotely controlled, such as remote setting time, remote recording and so on.
2. A camera can be added to the device so that the cat's condition can be monitored remotely in case of accidents. Bread boards can be replaced with welded hole boards. The whole project will be stronger and less susceptible to damage.
3. Stronger wires can be used to prevent the circuit from becoming loose. Bread boards can be replaced with welded hole boards. The whole project will be stronger and less susceptible to damage.

4. A stronger material can be chosen for the frame so that it could not be damaged by the cat's playful behavior during their daily activities. For example, acrylic sheet is beautiful and strong.
5. Switch can be replaced by button, in order to press remote and easily.

Figure 34 Button can be used

7.0 Source code

```
#define D4 eS_PORTD4
#define D5 eS_PORTD5
#define D6 eS_PORTB3
#define D7 eS_PORTD7
#define RS eS_PORTB0
#define EN eS_PORTB2
#define F_CPU 16000000UL

#include "lcd.h"
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <string.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <time.h>


#define BAUDRATE 9600// 9600 bps
#define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) - 1)//baud prescaller

//---------------------------------------LCD sub function
unsigned char rec = '0';//set receive char

void USART_init(void){
	UBRR0H = (uint8_t)(BAUD_PRESCALLER>>8); //set the baud rate of USART
	UBRR0L = (uint8_t)(BAUD_PRESCALLER);//set baud rate of USART
	UCSR0B = (1<<RXEN0)|(1<<TXEN0); //enable transmit of USART
	UCSR0C = (3<<UCSZ00); //use 8-bit (default) of USART
}

void USART_send(unsigned char data){
	while(!(UCSR0A & (1<<UDRE0))); // check that if data is sent
	UDR0 = data; // if sent, load new data to transmit
}

unsigned char USART_receive(void){
	while(!(UCSR0A & (1<<RXC0))); // Wait to receive the data
	return UDR0; // Read data from UDR data register
}

int get_length(char str[])//find length of string
{
	char *q = str;
	int cnt = 0;
	while (*q++ != '\0')
	{
		cnt=cnt+1;
```

```
        }
        return cnt;
}

void USART_putstring(char *str)
{
        int c;
        c=get_length(str);//get length of string
        for (int i=0; i<c; i++){
                if (str[i] != 0){
                        USART_send(str[i]); //display string
                }
        }
}

void transform(char *str){//get string from terminal
        int i = 0;
        char s=0;//use s to control the loop
        while(s!=0x0d&&s!=0x0a){
                char s = USART_receive();//receive data
                if(s==0x0d||s==0x0a){//0x0d=13 means enter, 0x0a=10 means /r /n
                        str[i++] =0;//stop transform
                        return;//get string
                }
                else{
                        str[i++] =s;//transform
                }
        }
}
//--------------------------------------LCD sub function

//--------------------------------------motor sub function
void delaym()//Use timer1 to prevent error caused by key jitter
{
        TCNT1=0xe17a; //load TCNT1
        TCCR1A=0x00;//TCCR1A=0 use timer1 normal mode
        TCCR1B=0x04; //Timer1, normal mode, 1024 prescaler
        while((TIFR1 & 0x1)==0); //wait for setting flag
        TCCR1B=0;//stop timer1
        TIFR1=0x01; //initialize timer1
}

void initADC(void) {
        ADMUX |= (1 << REFS0) | (1 << MUX0); //reference voltage on AV_CC, and use ADC1
        ADCSRA |= (1 << ADPS1) | (1 << ADPS0); //ADC clock prescaler / 8
        ADCSRA |= (1 << ADEN); //enables the ADC
}
void initADC1(void) {
        ADMUX |= (1 << REFS0) | (0 << MUX0); //reference voltage on AV_CC, and use ADC0
        ADCSRA |= (1 << ADPS1) | (1 << ADPS0); //ADC clock prescaler / 8
        ADCSRA |= (1 << ADEN); //enables the ADC
}
```

```
void direction(void)    //change direction by using PORTD4 and PORTD5
{
        if ((PINB&0x20)==0x20)//if PortD5=1, PortD4=0
        {
                PORTB &=~(1<<5);//make PortD5=0, PortD4=1
                PORTB|=1<<4;
        }
        else if ((PINB&0x10)==0x10)//if PortD5=0, PortD4=1
        {
                PORTB&=~(1<<4);//make PortD5=1, PortD4=0
                PORTB|=1<<5;
        }
                delaym();//delay 0.5s to prevent error caused by key jitter
}
//---------------------------------------motor sub function
//---------------------------------------servo sub function
void delays1()//use timer0 to delay 1s
{
        int i=0;//let i=0 to make a loop
        while(i!=61){
                TCNT0=0x00; //load tcnt0
                TCCR0A=0x00;//timer0
                TCCR0B=0x05; //timer0, 1024 prescaler
                while((TIFR0 & 0x1)==0); //wait
                TCCR0B=0;//stop timer0
                TIFR0=0x01; //clear tov0 flag
                i=i+1;
        }
}
void delays2()//use timer0 to delay 2s
{
        int i=0;//let i=0 to make a loop
        while(i!=122){
                TCNT0=0x00; //load tcnt0
                TCCR0A=0x00;//timer0
                TCCR0B=0x05; //timer0, 1024 prescaler
                while((TIFR0 & 0x1)==0); //wait
                TCCR0B=0;//stop timer0
                TIFR0=0x01; //clear tov0 flag
                i=i+1;
        }
}
void delays3()//use timer0 to delay 3s
{
        int i=0;//let i=0 to make a loop
        while(i!=183){
                TCNT0=0x00; //load tcnt0
```

```
                TCCR0A=0x00;//timer0
                TCCR0B=0x05; //timer0, 1024 prescaler
                while((TIFR0 & 0x1)==0); //wait
                TCCR0B=0;//stop timer0
                TIFR0=0x01; //clear tov0 flag
                i=i+1;
        }
}
void delays4()//use timer0 to delay 4s
{
        int i=0;//let i=0 to make a loop
        while(i!=244){
                TCNT0=0x00; //load tcnt0
                TCCR0A=0x00;//timer0
                TCCR0B=0x05; //timer0, 1024 prescaler
                while((TIFR0 & 0x1)==0); //wait
                TCCR0B=0;//stop timer0
                TIFR0=0x01; //clear tov0 flag
                i=i+1;
        }
}
void delays()//use timer0 to delay 2s
{
        int i=0;//let i=0 to make a loop
        while(i!=50){
                TCNT0=0x00; //load tcnt0
                TCCR0A=0x00;//timer0
                TCCR0B=0x05; //timer0, 1024 prescaler
                while((TIFR0 & 0x1)==0); //wait
                TCCR0B=0;//stop timer0
                TIFR0=0x01; //clear tov0 flag
                i=i+1;
        }
}
//--------------------------------------servo sub function

int main()
{

DDRD=0xFF;// Set all output
DDRB=0xFF;// Set all output
PORTB = 0x00;// Set all low
PORTD = 0x00;// Set all low
TCCR0A=0xc3;// Use fast PWM mode with inverting mode
TCCR0B=0x05;// use timer0 with 1024 prescaler
OCR0A=0xFF;//OCR0A=0xFF=255
```

```
int main()
{

DDRD=0xFF;// Set all output
DDRB=0xFF;// Set all output
PORTB = 0x00;// Set all low
PORTD = 0x00;// Set all low
TCCR0A=0xc3;// Use fast PWM mode with inverting mode
TCCR0B=0x05;// use timer0 with 1024 prescaler
OCR0A=0xFF;//OCR0A=0xFF=255

Lcd4_Init();//initialization of LCD using 8-bit mode
Lcd4_Clear();          //clean LCD
USART_init();// Initialize USART
char string[255]="Wait ";//The first letters to appear
Lcd4_Set_Cursor(1,0);//set cursor at row 1 and column 0
Lcd4_Write_String("ModeD!");// display string on LCD
USART_putstring("Welcome to use\n");//display on terminal

        while (1)//wait here
        {
                USART_init();//USART initialization
                rec=USART_receive();
                Lcd4_Init();//initialization of LCD using 4-bit mode

                if(rec == 'A'){//ModeA
                        Lcd4_Clear();          //clean LCD
                        Lcd4_Init();//initialization of LCD using 4-bit mode
                        Lcd4_Set_Cursor(1,3);//set cursor at row 1 and column 3
                        Lcd4_Write_String("ModeA!");// display string on LCD
                        USART_putstring("In modeA\n");//display on terminal

                        DDRD &=~(1<<2);// Set PD2 input
                        DDRD &=~(1<<3);// Set PD3 input
                        DDRC|=(1<<3)|(1<<4)|(1<<5);// Set PC3 PC4 PC5 output
                        PORTB|=(1<<5);//PB5=1 to control direction of motor
                        PORTB|=(1<<2);//PB2=1

                        TCCR0A=0x83;// Use fast PWM mode with inverting mode
                        TCCR0B=0x05;// use timer0 with 1024 prescaler
                        uint16_t pValue;
                        int value;
                        initADC();//function initADC()
                        int c=0;//initialize c
                        while(1){
                                        if((PIND&0x08)==0x08){// Quit
                                        Lcd4_Clear();          //clean LCD
```

```
USART_putstring("Quit\n");//display on terminal
PORTC &=~(1<<4);//PC3 PC4 PC5 as 0
PORTC &=~(1<<5);
PORTC &=~(1<<3);
OCR0A = 0;//OCR0A=0 stop motor
Lcd4_Set_Cursor(1,0);//set cursor at row 1 and column 0
Lcd4_Write_String("Welcome!");// display string on LCD
break;
}
ADCSRA |= (1 << ADSC); //art ADC conversion
while((ADCSRA & (1 << ADSC))); //wait until ADSC bit is clear, i.e., ADC conversion is done
uint8_t theLowADC = ADCL;//the low value is in ADCL register
pValue = ADCH << 8 | theLowADC;//the high value is in ADCH register
value = pValue/4;//make value in the range of about 0-255
if (value > 255) {
        OCR0A = 255;//if OCR0A overflow make it equal to 255
}
else if (value < 0)
{
        OCR0A = 0;//if OCR0A is negative make it equal to 0
}
else {
        OCR0A = value;//else make it equal to ADC value
}
if((PIND&0x04)!=0x04){
        direction();
}
delaym();//delay 0.5s to prevent error caused by key jitter

//srand ();
c=1+rand()%6;//get random number
switch (c)//select laser case
{

        case 0:{
                PORTC|=(1<<4);
                PORTC &=~(1<<5);
                PORTC &=~(1<<3);
                delaym();
                break;//jump out of switch
        }
        case 1:{
                PORTC&=~(1<<4);
                PORTC|=(1<<5);
                PORTC &=~(1<<3);
                delaym();
                break;//jump out of switch
        }
```

```
                                        }
                                        case 2:{
                                                PORTC&=~(1<<4);
                                                PORTC&=~(1<<5);
                                                PORTC|=(1<<3);
                                                delaym();
                                                break;//jump out of switch
                                        }
                                        case 3:{
                                                PORTC&=~(1<<4);
                                                PORTC|=(1<<5);
                                                PORTC|=(1<<3);
                                                delaym();
                                                break;//jump out of switch
                                        }
                                        case 4:{
                                                PORTC|=(1<<4);
                                                PORTC|=(1<<5);
                                                PORTC|=(1<<3);
                                                delaym();
                                                break;//jump out of switch
                                        }
                                        case 5:{
                                                PORTC|=(1<<4);
                                                PORTC &=~(1<<5);
                                                PORTC|=(1<<3);
                                                delaym();
                                                break;//jump out of switch
                                        }
                                        case 6:{
                                                PORTC|=(1<<4);
                                                PORTC|=(1<<5);
                                                PORTC &=~(1<<3);
                                                delaym();
                                                break;//jump out of switch
                                        }
                                }
                        }
                }

        if(rec == 'B'){//ModeB
                DDRD &=~(1<<2);//PD2 PD3 input
                DDRD &=~(1<<3);
                PORTB|=(1<<2);//PB2 high

                Lcd4_Clear();          //clean LCD
                Lcd4_Init();//initialization of LCD using 4-bit mode
                Lcd4_Set_Cursor(1,3);//set cursor at row 1 and column 3
                Lcd4_Write_String("ModeB!");// display string on LCD
```

```
USART_putstring("In modeB\n");//display on terminal
delays1();//delay for 2s
Lcd4_Clear();         //clean LCD
Lcd4_Write_String("Set time");// display string on LCD
TCCR1A |= (1 << WGM11) | (1 << COM1A1);//set fast pwm 14 mode with non-inverting mode
TCCR1B |= (1 << WGM13) | (1 << WGM12) |(1 << CS11);//set pre-scale of 8 cs11
ICR1 = 40000; //set icr1max = 40000 icr1 is the top defining pwm period
int c=0;
while(1){//duty cycle need offset to correct
if((PIND&0x04)!=0x04){//set time
        c=c+1;
        if(c>4){
                c=0;
        }
_delay_ms(50);//Avoid error
}
switch (c){
        case 0:{
                Lcd4_Clear();         //clean LCD
                Lcd4_Write_String("Set time");// display string on LCD
                _delay_ms(500);
                break;//jump out of switch
        }
        case 1:{
                Lcd4_Clear();         //clean LCD
                Lcd4_Write_String("Set one hour");// display string on LCD
                _delay_ms(500);
                break;//jump out of switch
        }
        case 2:{
                Lcd4_Clear();         //clean LCD
                Lcd4_Write_String("Set two hours");// display string on LCD
                _delay_ms(500);
                break;//jump out of switch
        }
        case 3:{
                Lcd4_Clear();         //clean LCD
                Lcd4_Write_String("Set three hours");// display string on LCD
                _delay_ms(500);
                break;//jump out of switch
        }
        case 4:{
                Lcd4_Clear();         //clean LCD
                Lcd4_Write_String("Set four hours");// display string on LCD
                _delay_ms(500);
                break;//jump out of switch
        }
}
```

```
                }
                        while((PIND&0x08)!=0x08){
                        switch (c)
                        {
                                case 0:{
                                        OCR1A = 1000;//-90 degree
                                        break;//jump out of switch
                                }
                                case 1:{
                                        //Lcd4_Clear();         //clean LCD
                                        //Lcd4_Write_String("Set one hour");// display string on LCD
                                        OCR1A = 1000;//-90 degree
                                        delays1();//delay for 1s
                                        OCR1A = 5000;//90 degree
                                        delays();//delay for 1s
                                        OCR1A = 1000;//-90 degree
                                        delays1();//delay for 1s
                                        break;//jump out of switch
                                }
                                case 2:{
                                        OCR1A = 1000;//-90 degree
                                        delays2();//delay for 2s
                                        OCR1A = 5000;//90 degree
                                        delays();//delay for 1s
                                        OCR1A = 1000;//-90 degree
                                        delays2();//delay for 2s
                                        break;//jump out of switch
                                }
                                case 3:{
                                        OCR1A = 1000;//-90 degree
                                        delays3();//delay for 3s
                                        OCR1A = 5000;//90 degree
                                        delays();//delay for 1s
                                        OCR1A = 1000;//-90 degree
                                        delays3();//delay for 3s
                                        break;//jump out of switch
                                }
                                case 4:{
                                        OCR1A = 1000;//-90 degree
                                        delays4();//delay for 4s
                                        OCR1A = 5000;//]90 degree
                                        delays();//delay for 1s
                                        OCR1A = 1000;//-90 degree
                                        delays4();//delay for 4s
                                        break;//jump out of switch
                                }
                        }
                }
        }
        }
        if(rec == 'C'){//ModeC
                DDRD &=~(1<<2);//PD2 PD3 input
```

```
                DDRD &=~(1<<3);
                PORTB|=(1<<2);//PB2 high
                Lcd4_Clear();          //clean LCD
                Lcd4_Set_Cursor(1,3);//set cursor at row 1 and column 3
                Lcd4_Write_String("ModeC!");// display string on LCD
                USART_putstring("In modeC\n");//display on terminal
                delays1();
                EIMSK |= (1 << INT0); //enable external interrupt 0
                sei(); //enable interrupts
                while (1){
                        Lcd4_Clear();          //clean LCD
                        UCSR0B &= ~((1<<RXEN0)|(1<<TXEN0));//stop USART send and receive
                        Lcd4_Init();//initialization of LCD using 8-bit mode
                        Lcd4_Set_Cursor(1,3);//set cursor at row 1 and column 3
                        Lcd4_Write_String(string);// display string on LCD
                        USART_init();//USART initialization
                        transform(string);// get string from terminal one by one
                        USART_putstring("Received note\n");//display on terminal
                        delays1();//delay for 1s
                        USART_putstring("Please change note\n");//display on terminal
                }
        }
        if(rec == 'D'){//ModeD
                DDRD &=~(1<<2);//PD2 PD3 input
                DDRD &=~(1<<3);
                uint16_t pValue;
                int value;
                initADC1();//function initADC()
                DDRB =0xFF;//output
                DDRD =0xFF;//output
                DDRC =0x00;//input
                PORTD &= ~(1<<3);//PD3
                Lcd4_Init();                    //Initialize LCD
                char data[5];
                while (1)//wait here
                {
                        Lcd4_Clear();                   //Clear LCD
                        char* s1="Humidity =";// set Humidity
                        Lcd4_Set_Cursor(1,0);   //set cursor at row 1 and column 0
                        Lcd4_Write_String(s1);//
                        PORTD|=(1<<3);//PD3 high
                        ADCSRA |= (1 << ADSC); //art ADC conversion
                        while((ADCSRA & (1 << ADSC))); //wait until ADSC bit is clear, i.e., ADC conversion is done
                        uint8_t theLowADC = ADCL;//the low value is in ADCL register
                        pValue = ADCH << 8 | theLowADC;//the high value is in ADCH register
                        value = pValue;//make value in the range of about 0-1023
                        PORTD &= ~(1<<3);//PD3=0
                        itoa(value,data,10);//change data type to char
                        Lcd4_Set_Cursor(1,11);//set cursor at row 1 and column 11
                        Lcd4_Write_String(data);//display data
```

```
                        if (value > 600) {
                        }
                        else if (value < 600)
                        {
                        Lcd4_Set_Cursor(2,0);//set cursor at row 2 and column 0
                        Lcd4_Write_String("wrong");//display wrong
                        }
                        if((PIND&0x08)==0x08){//Quit
                                USART_putstring("Quit\n");//display on terminal
                                Lcd4_Clear();//Clear LCD
                                Lcd4_Set_Cursor(1,0);//set cursor at row 1 and column 0
                                Lcd4_Write_String("Welcome!");// display string on LCD
                                break;
                        }
                        delays1();
                }
        }
    return 0;
}

ISR (INT0_vect) //ISR for external interrupt 0
{
                                        Lcd4_Clear();//Clear LCD
                                        USART_putstring("Quit\n");//display on terminal
                                        Lcd4_Set_Cursor(1,0);//set cursor at row 1 and column 0
                                        Lcd4_Write_String("Welcome!");// display string on LCD
                                        delays1();//delay 1s
}
```

Here is video for how to use it. I uploaded video on Bilibili and YouTube.
YouTube:
https://www.youtube.com/watch?v=lUdwOqK1184
Bilibili:

【B31DD】Miniproject| Cat catering machine| Based on Arduino| H00315680| Yixin Cao_哔哩哔哩_bilibili