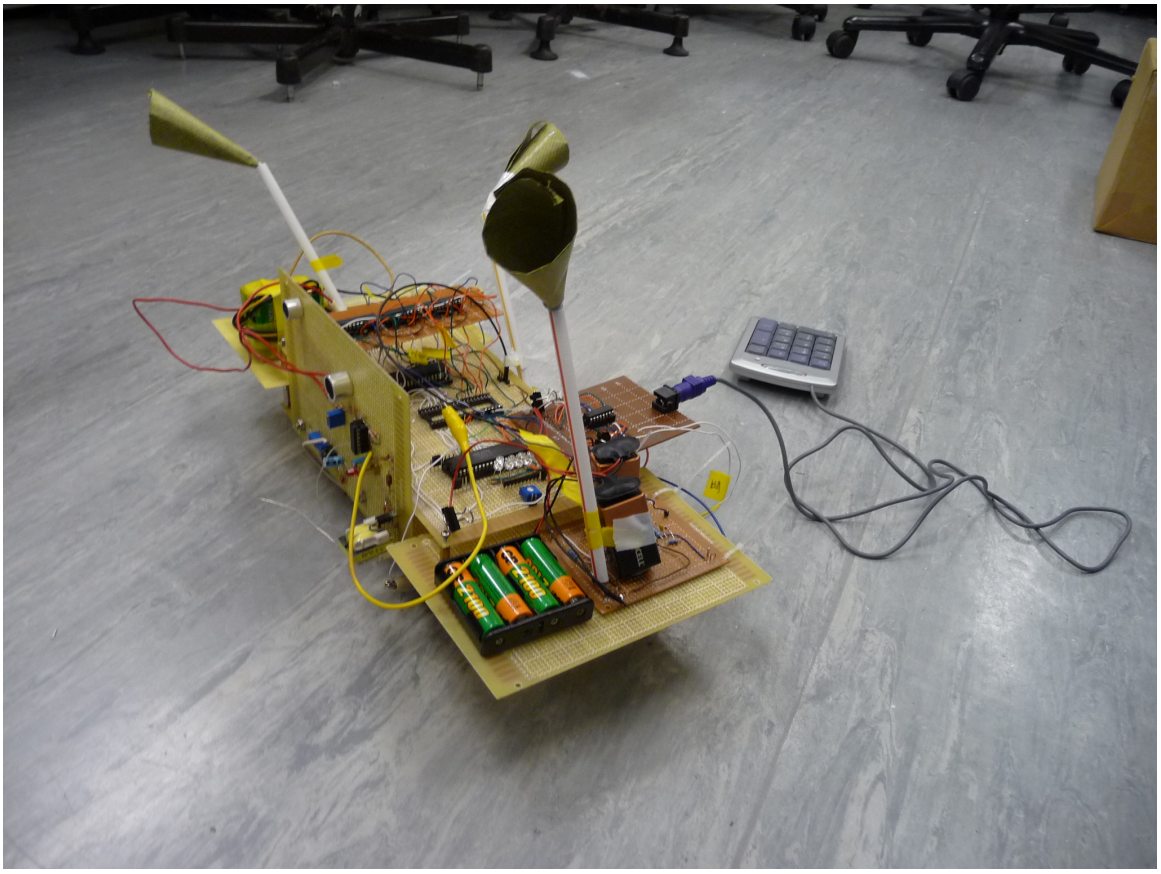
ELEC254 MICROPROCESSOR EXPERIMENTS
COURSE PROJECT REPORT

Mobile Island:
A Multifunctional Sound Control Car



Zhuo Wei, 05719905, *ee_zwxaa@stu.ust.hk*
Wang Zhen, 05711977, *ee_wzx@stu.ust.hk*
Zhu Shucheng, 05729558, *eg_zsxac@stu.ust.hk*

Contents

1. Introduction

In this course project, we have made fully use of only a few kinds of common and inexpensive ICs, and build a sound control car, named "Mobile Island", because we think that the microphones on the car bear the resemblance of coconut trees on a island. "Mobile Island" is multifunctional because it can be operated under three control modes: Sound Control, Keypad Control, and Free Run.
The main functions of these modes are:

- Mode 1 - sound control: Island can detect the source of the sound. After detecting the sound source, it will turn and move to the sound source.
- Mode 2 - keypad control: Island can be controlled by the keypad.
- Mode 3 - free run: Island can move by itself. It can also detect the obstacle in front of it while moving so that it can avoid crashing on the obstacle by itself.

The rest of this report is organized in following way: Section 2.1 and Section 2.2 mainly presents non-technical information on hardware and software. Section 2.3 describes the overall algorithm of voice localization. Section 3 provide the actual setup procedures and operation instructions. It also states some limitations of our project. Section 4 provide detailed information on hardware, including all the schematics and brief explanations. Section 5 provided detailed and complete implementation on software, including program list and flowchart. Finally, we conclude this project experience in Section 6.

For readers' convenient preview, we post a video for our project on YouTube, you are welcome to watch it. [1]

[1] http://www.youtube.com/v/lgQ4HIu8d6E&hl=en&fs=1

2. Overall Description

2.1. **Overall Description on Hardware.**

We built this project in a bottom-up style. Implementation begins with basic modules that do not need any other modules. i.e. we first built and test amplification circuits of capacitor mic and ultrasound, then worked towards analysis the signal from them.

For the sound control part, we first collected information about different types of microphones being widely used. Generally there are three kinds of microphones: electret condenser mic (capacitor mic), dynamic mic, and carbon mic. We decide to use capacitor mic because it is relatively small and the most available on market. However, the raw signal from capacitor mic is much smaller than that from carbon mic or dynamic mic, so we carefully built a amplification circuit for capacitor mic in order to read signals from it. Section 4 shows that, for each capacitor mic, at least three operational amplifier (LM324) are needed to achieve a output voltage range from 0 to 5V.

For the Keypad controlled mode, we use a PS/2 keypad connecting to a PS/2 female connector soldered on the board. The PS/2 female connector is connected to the socket holding the emulator directly.

In the Free Run mode and obstacle detection, we decided to employ an ultrasonic circuit. The distance detection is based on the time interval between ultrasound transmission and reflection. Though infrared is a more robust way to detect object compared to ultrasound, however, the distance information is difficult to obtain from infrared sensor and the whole system is also relatively expensive. The whole ultrasound transmitting and detecting circuit is operating under a 9V voltage supply.

In all the three modes, Four LEDs are used to indicate different modes. Also, two DC motors are used to move the car forward, backward, left and right. The two DC motors are controlled by a motor driver LS293D. The motor driver LS293D uses a separate +9V power source as the two DC motors consume a lot of power.

The control pins of the motor driver, the LEDs and the ultrasonic control pin are connected to an Intel 8255 Programmable Peripheral Interface chip[2] [3]. The 8255 chip is used to extend the limited pins on 8051. All the pins on 8255 chip are used as output pins. The ADCs used to convert the analog sound signals to digital signals are connected to another Intel 8255 Programmable Peripheral Interface chip,

[2] `http://en.wikipedia.org/wiki/Intel_8255`
[3] `http://en.wikipedia.org/wiki/Physical_interface`

so that the whole microphones amplification and ADC circuits use only 17 pins on 8051.

Finally, the more detailed implementation, including schematic and a complete component list are provided in Section 4.

2.2. **Overall Description on Software.**

The whole project is written in C language by using Keil uVision[4] IDE. Like many other team projects, each member of us is responsible for the programming in one control mode, and we later integrate them together in the main file. From the perspective of functionality, the main functions are:

- main: for demonstration
- motor: for motor control
- mic: for sound collection and voice localization
- keypad: for keypad control
- ultrasound: for obstacle detection
- setting8255: for using 8255 chip controlling motors, LEDs and ultrasound control pin
- delay: for time delays

From the perspective of algorithm design, The main features of this project are

(1) Environment Sampling: first sample the environment and set the threshold as the maximum value obtained during sampling.
(2) Voice Localization: After A/D conversion, the received the signals from the three electret condenser microphones is analyzed and proper action is triggered. Our car can achieve 30 degrees of resolution in sound source detection.
(3) Obstacle detection: By using ultrasonic, the car gets the information of how far the obstacle is from the front of the car every three seconds (the sampling period is adjustable) and modify its retreating step size according to it.

Note that Voice Localization has always been an interesting and important topic in both industrial and research sides of electronic engineering. In this project, we make effort to combine voice localization with robotics control: the machine makes different actions according to the different reponse patterns from microphones. To our knowledge, no previous group projects have looked into this aspect before.

2.3. **Principles of Operation.**

In the sound control mode, three microphones are placed on left side, right side and rear part of the car, as shown in Figure 1, and they are about 25cm apart

[4] `www.keil.com`

from each other. The responses from the rear, left and right microphones are denoted as mic1, mic2 and mic3 in the program. The function *mic_cont_check*() is used to update the values of mic1, mic2 and mic3. In order to let the car perform voice localization in different environments, i.e. sometimes background noise could be large or small in different environment, three threshold values correspond to three microphones: threshold1, threshod2 and threshold3, are computed by the function *mic_sampleEnvironment*(), and a microphone's response is compared to its threshold value plus a predetermined bias value. Following is the algorithm of

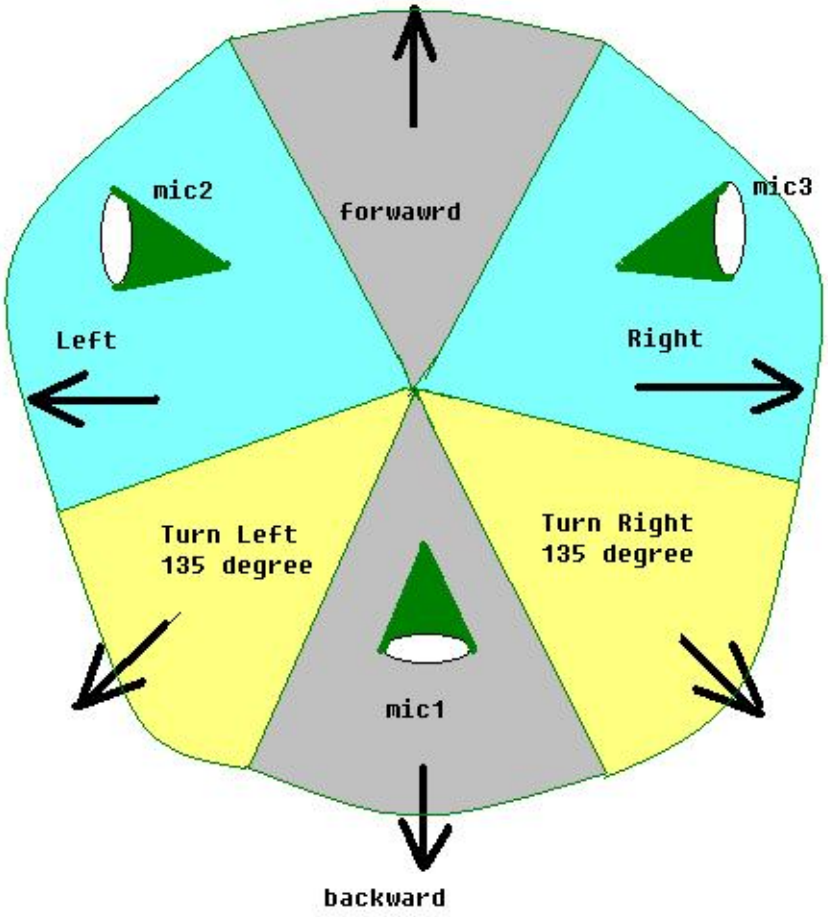


FIGURE 1. Voice localization scheme

our voice localization scheme:

```
if(mic1<threshhold1+bias1 && mic2<threshhold2+bias2 && mic3<threshhold3+bias2)
       {return;}
if (mic2>mic1 && mic3>mic1 && mic2<mic3+bias2 && mic3<mic2+bias3) {
       motor_forward_s(3);
```

```
        delay_s(2);
        return;} //forward
if (mic1>mic2+bias1 && mic1>mic3+bias1 ){
        motor_backward_s(3);
        delay_s(2);
        return; } //backward
if (mic2>mic3 && mic1>mic3 && mic2<mic1+bias2 && mic1<mic2+bias1) {
        motor_left_s(3);
        delay_s(2);
        motor_forward_s(2);
        delay_s(2);
        return;} //southeast
if (mic3>mic2 && mic1>mic2 && mic3<mic1+bias3 && mic3<mic1+bias3) {
        motor_right_s(3);
        delay_s(2);
        motor_forward_s(2);
        delay_s(2);
        return;} //southwest
if (mic2>mic3+bias2 && mic2>mic1+bias2 ){
          motor_left_s(2);
          delay_s(2);
          motor_forward_s(3);
          delay_s(2);
         return; } //left
if (mic3>mic1+bias3 && mic3>mic2+bias3 ){
          motor_right_s(2);
          delay_s(2);
          motor_forward_s(3);
          delay_s(2);
          return;
          } //right
return;
```

Note that bias1, bias2 and bias3 are three parameters that should be determined during test trails. If the properties of three microphones are very different, the three bias values tend to be very different, either. In our project, the bias values are selected to 50 or 60.

3. Setup Instructions, Operation Instructions and Limitations

3.1. **Setup Instructions.**

Following is the actual setup steps of our project:

(1) Put the car on the stable ground.
(2) Plug the emulator onto the car and connect the emulator to the computer properly.
(3) Connect the +5V battery to the ADCs and 8255 chips.
(4) Connect +9V and -9V batteries to the microphones.
(5) Connect another +9V battery to the motor driver LS293D.
(6) Rebuild and run the program.
(7) The detailed operation instructions will be given in the next subsection.

3.2. **Operation Instructions.**

The detailed operation procedure is illustrated as follows:
After finish the setup steps,

(1) The 4 LEDs will be on and off following certain pattern. This indicates that the car starts to work
(2) Then 4 LEDs will be all on. Press "Enter" on the keypad to enter the menu part
(3) The green LED indicates that the car is in menu part and is waiting for further instruction from the keypad
(4) Press "NUM" for certain times to make sure that the NUM-Lock LED on the keypad is on
(5) Then press "1" to enter mode 1 (sound control); press "2" to enter mode 2 (keypad control); press "3" to enter mode 3 (free run)
(6) When the car is in any of the 3 modes, press "Enter" to go back to the menu part

When enter Mode 1 (sound control) :

(1) The car is waiting for the sound source
(2) Shout to The car. The car will turn to you first; then move to you for 3 seconds
(3) Then the car will wait for further sound source
(4) Press "Enter" to exit mode 1 and go back to menu

When enter Mode 2 (keypad control) :

(1) The car is waiting for the keypad instruction.
(2) Press "NUM" for certain times to make sure that the NUM-Lock LED on the keypad is off. The car only responds to function keys such as "up-arrow", "down-arrow", "left-arrow" and "right-arrow". It will NOT respond number keys.
(3) Press "up-arrow" to move the car forward.
(4) Press "down-arrow" to move the car backward.
(5) Press "left-arrow" to turn the car to the left.
(6) Press "right-arrow" to turn the car to the right.
(7) Press "Enter" to exit mode 2 and go back to menu.

When enter Mode 3 (free run):

(1) The car will move forward by itself.
(2) The car will check the obstacle in front of it every 3 seconds.
(3) If the car detects the obstacle in front of it, it will move backward first and then turn left. After turning left, it will keep moving forward.
(4) The car can detect different distance from the obstacle. The car will move backward for longer distance if the obstacle is closer to it.
(5) Press "Enter" to exit mode 3 and go back to menu.

3.3. **Limitations.**
Here we identified some limitations of our project.

First of all, the car can not tell a human voice's location when the sound source is far from the car. This is because the amplitude of capacitor microphone's response falls rapidly as distance increases, as shown in Figure 2. When sound source is

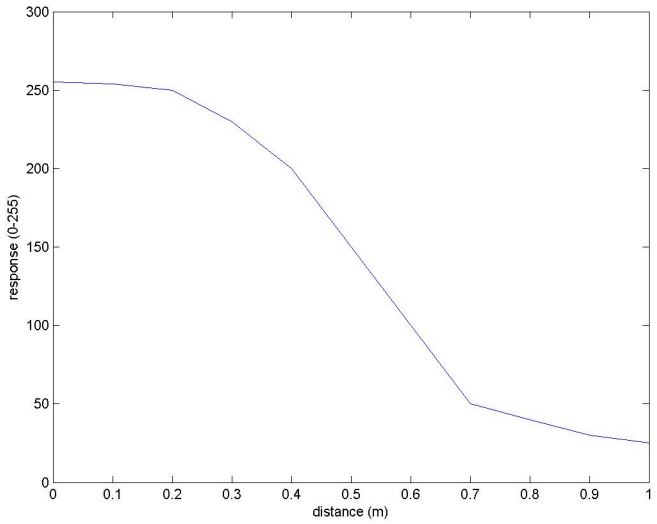


FIGURE 2. Amplitude vs distance, data collected from the speaker on Nokia N5300

above 1.5m away from the car, it becomes very difficult for the program to tell the difference between responses.

Also in mode 3, the ultrasound detection is affected by the noise generated by the two DC motors. So the car has to detect the obstacle only when it stops.

Finally, we need three different kinds of power resources, i.e. +5V, +9V, -9V, for different devices. And the motor driver LS293D needs a separate +9V power

resource for large power consumption from the two DC motors. However, we think this problem can be solved by changing component selection in the future.

4. Detailed Hardware Description

First of all, the whole architecture of the car is shown in Figure 3. Then detailed explanation on different hardware parts will be presented in following subsections.

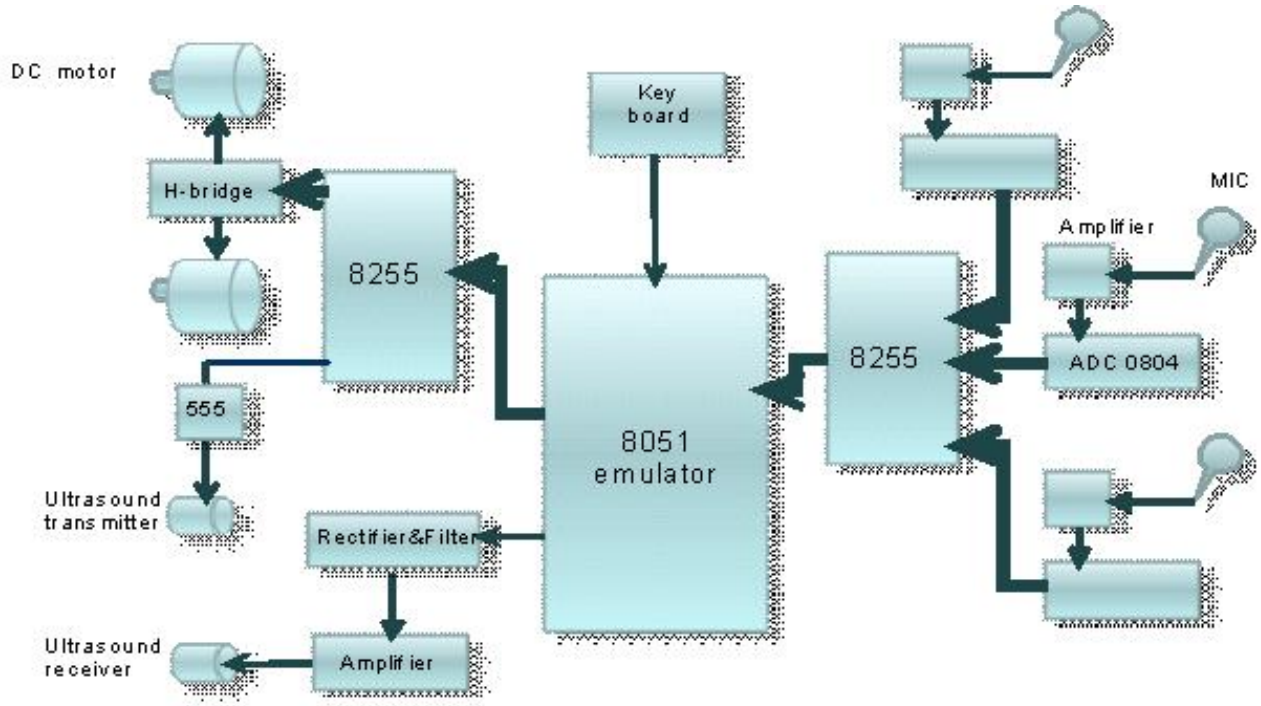


Figure 3. Overall hardware architecture

4.1. **Circuits related to Sound Control.**

Figure 4 shows the preamplifier of capacitor microphone. The three operational amplifiers are connected in series and provide a total amplify ratio about $15^3 \approx 3400$. The 103 capacitor across the opAmp is used to filter out jitters in the received signal. Without 103 capacitor, the received signal would jump to a large value easily even there is no perceivable sound source exists. Then the amplified signal passed a rectifier (IN4148) and a low pass filter and output to the input of ADC0804. In Figure 5, the Write signal lines of three ADC are wired together so to ensure the A/D conversion start at the same time. The variable resistor connect to Vref/2 is used to adjust the precision of conversion. The smaller the Vref/2, the higher the precision is, however, the conversion range will be reduced. In our demo, Vref/2 is adjusted to 2.5V.

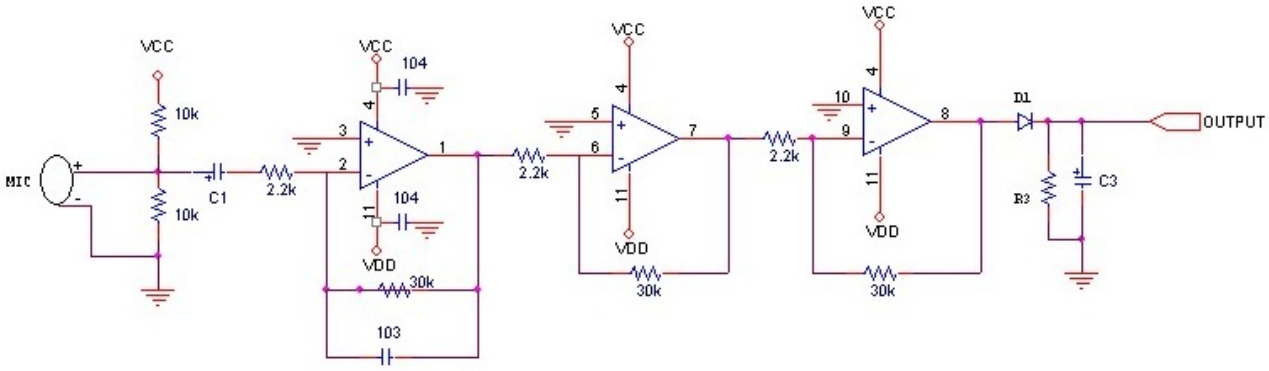


FIGURE 4. Preamplifier circuit for capacitor microphone

4.2. **Keypad Interfacing.**
We use standard PS/2 keypad in this project. The PS/2 protocol is applied for communication between the keypad and the emulator. The CLK line of the keypad is connected to P3.2 (external interrupt 0) of 8051 and the DATA line of the keypad is connected to P3.1 (normal input pin) of 8051.The keypad implements a bi-directional protocol. The keypad can send data to the host and the host can send data to the Keypad. The host has the ultimate priority over direction.

The timing diagram for keypad to host is shown as Figure 6. It takes 11 clock cycles to transmit each byte. The data should be sampled on the data bus at each falling edge of the clock. Bit 2 to 9 of the transmitted packet is the data. In each falling edge of the clock, the external interrupt will occur and the bit will be recorded. When it counts to a total 11 bits, the data will be stored in a buffer waiting for decoding. Decoding process will be called in the main function. After decoding process, the data will be deleted from the buffer. The keypad to host protocol is used to get keypad values.

The timing diagram for the host to keypad is shown as Figure 7. The host to keypad protocol is initiated by taking the keypad data line low. The keypad will start generating a clock signal on its clock line. After the first falling edge has been detected, the first bit can be loaded on the data line. This bit will be read into the keypad on the next falling edge, after which you can place the next bit of data. This process is repeated for the 8 data bits. After the data bits comes a parity bit. Once the parity bit has been sent and the keypad data line is in an idle (high) state for the next clock cycle, the keypad will acknowledge the reception of the new data. The keypad does this by taking the keypad data line low for the next clock transition. The host to keypad protocol is used to turn on and off the NUM-Lock LED.

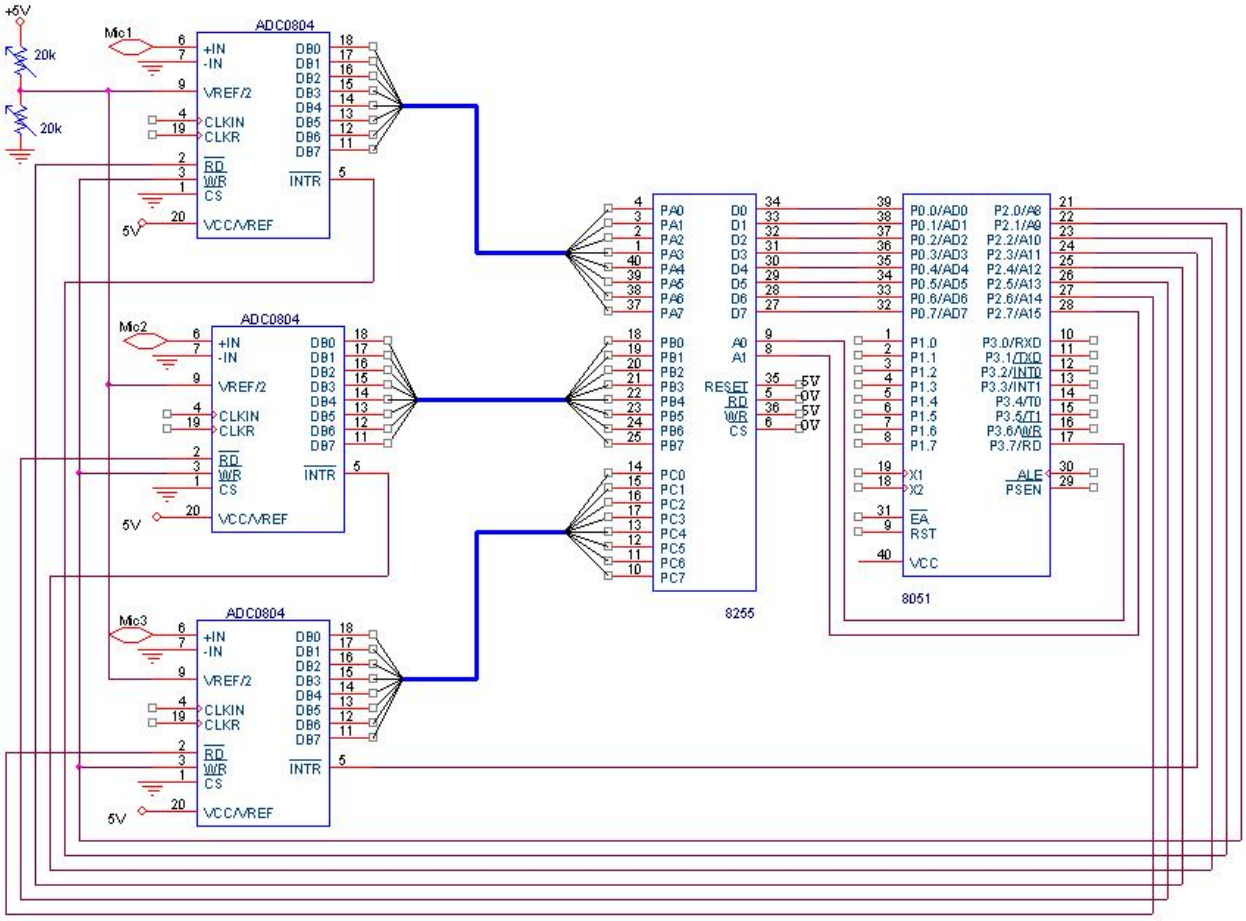


FIGURE 5. Sound ADC and control circuit

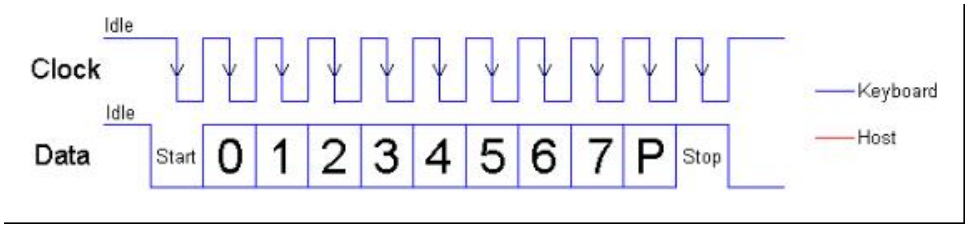


FIGURE 6. Timing diagram for keypad to host

There are two kinds of code in a PS/2 keypad. They are make code and break code. Make code is an 8 bit data and break code is 0xF0 plus the previous 8 bit data. We need to detect the two codes to determine the key status.

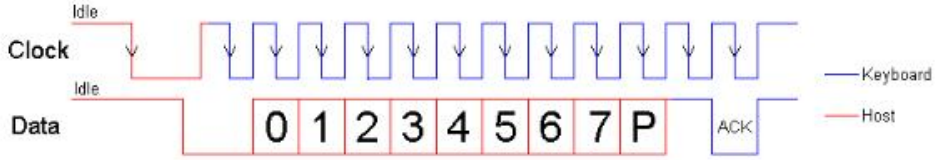


FIGURE 7. Timing diagram for host to keypad

4.3. **DC Motors.**

Referred to Figure 8, the two DC motors are controlled by the motor driver LS293D. So the control to the motors is actually the control to the motor driver. The program will enable the two DC motors at the very beginning. When motor functions are called, the values on the controlled pins of the motor driver will be changed in order to spin the DC motor.

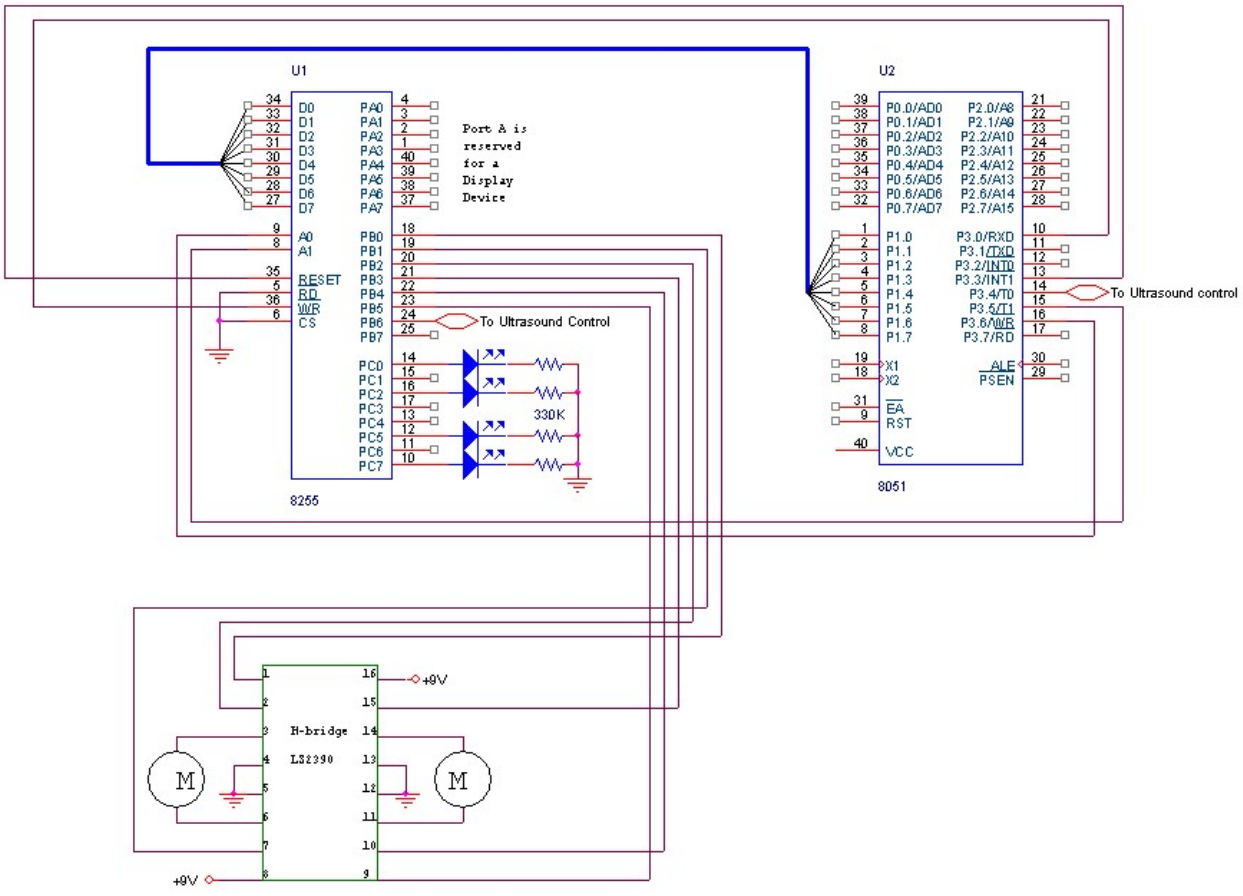


FIGURE 8. motor control circuit

4.4. **Ultrasonic Receiver and Transmitter.**
As shown in Figure 9, the ultrasound is generated by a 555 chip, whose pulse frequency is set to 40kHz by a adjustable resistor. And it is turned on or off by software through the pin 1 of the 555 timer. The ultrasound transmitter can only work at a frequency around 40kHz, which means the amplitude of the sound generated by the transmitter is maximized when the 555 timer is set to 40kHz. As for the ultrasound receiver, the operating frequency is the same as that of the transmitter. Actually, the receiver is structurally the same as a transmitter, but with an inversed function, converting mechanical energy into electrical energy. After the ultrasound is detected by the receiver, its transformed into a sinusoidal waveform at the very beginning. This sinusoidal signal is then amplified by 1100 times, which is achieved by using two operational amplifiers in the chip LM324N. Every time after an op-amp, the magnitude of this signal is multiplied by a factor of 33. A diode is then acting as a rectifier to transform the AC signal into a DC one. For the convenience of signal detection, the DC signal is filtered by a low-pass filter to make the fluctuating signal a stabilized one, which is either 7V or 0. Eventually, compared with a voltage around 4.5V by a comparator in the LM324N chip, the circuit is able to tell whether an ultrasound signal is detected just by checking whether the output is high.

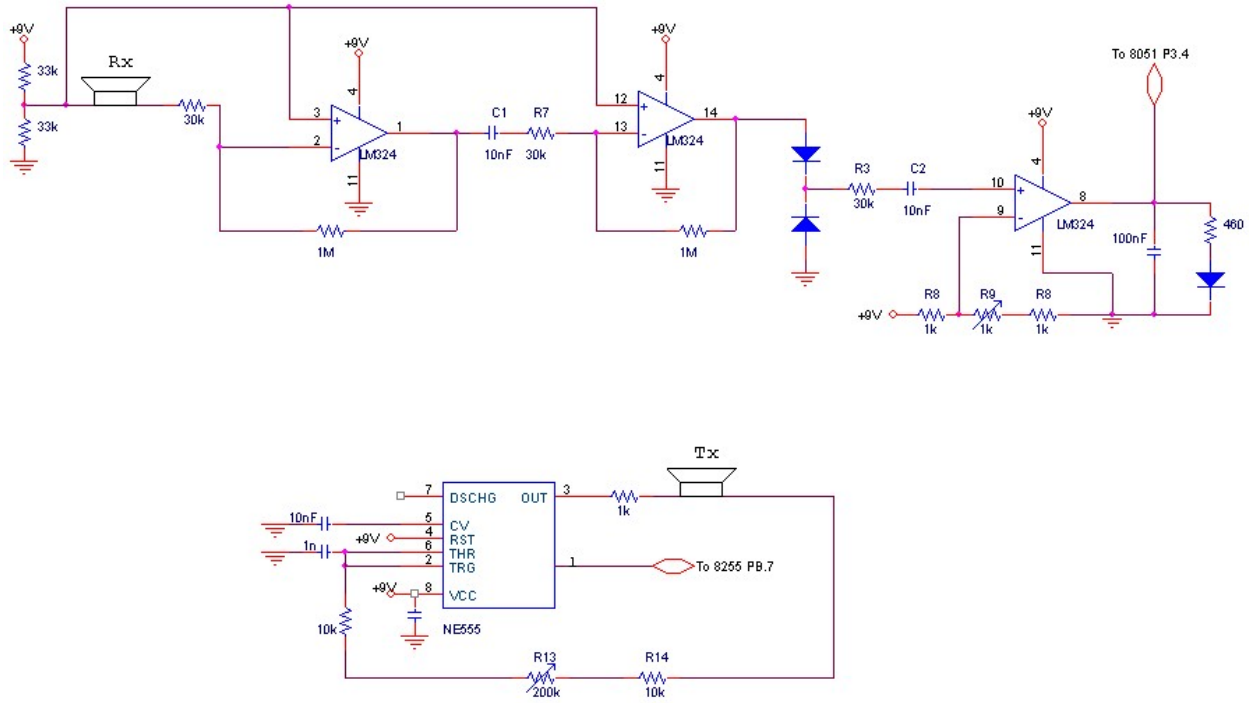


FIGURE 9. ultrasonic receiver and transmitter circuit

4.5. **Part List.**
Following is a complete list of the hardware components used in our project. Figure 10 and 11 show a mapping of some major hardware components with pictures of the car.

```
8051 emulator
8255*2
ADC0804*3
Motor H-bridge LS293D
Gear box*2
NE55 timer
LM324N operation amplifier*4
Capacitor mic*3
Ultrasound transmitter
Ultrasound receiver
Number keypad
DC motor*2
+9V, -9V and +5V voltage supply
LED*4
Circuit boards
IC sockets
Resistors(normal & adjustable), inductors, capacitors
```

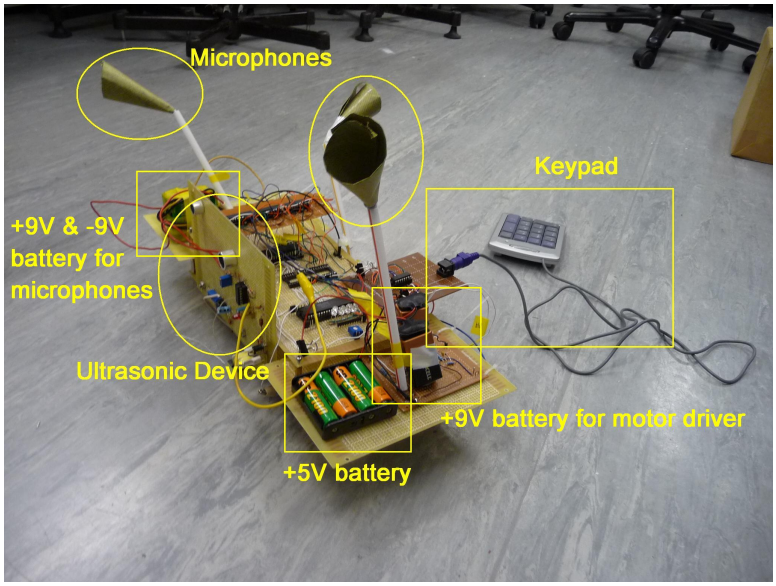


FIGURE 10. Hardware devices 1

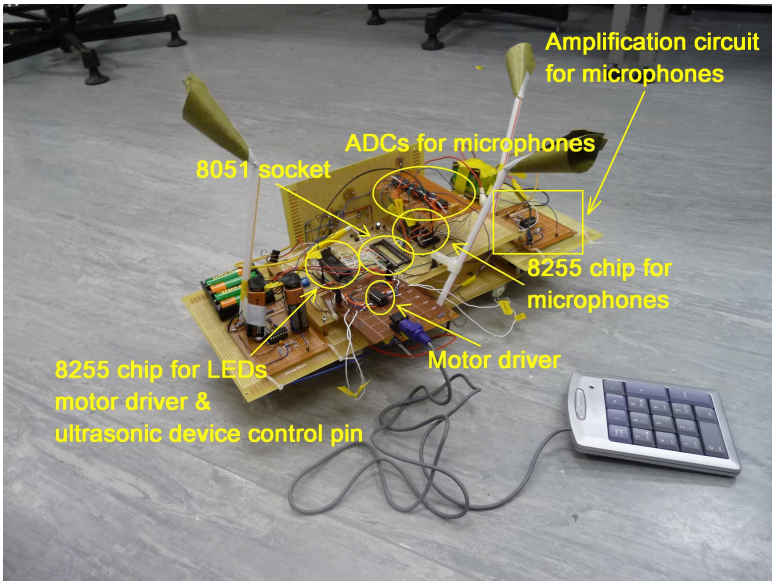


FIGURE 11. Hardware devices 2

5. Detailed Software Description

This section gives the detailed software description mainly in forms of flowcharts. The total lines in source files is about 1000. The size of compiled .hex file is under 8KB.

Figure 12 shows the overall flow of the main program running in the car.

Referred to previous description on Keypad Interfacing, Figure 13 and 14 shows the flowcharts of keypad control program.

Figure 15 shows the flowchart of sound control mode. Note that the the A/D conversion start at the same time for three microphones, however, the data collections are performed one by one.

Lastly, Figure 16 shows the flowchart of ultrasonic control.

5.1. **Program List.**

The whole software package contains following files

(1) main.h, main.c
(2) mic.h, mic.c
(3) keypad.h, keypad.c
(4) ultrasound.h, ultrasound.c

The package also contain three basic modules that are constantly called by above modules. They are

(1) setting8255.h, setting8255.c
(2) motor.h, motor.c
(3) delay.h, delay.c

The main functions and lines of these modules are:

- main: for demonstration; 148 lines
- motor: for motor control; 186 lines
- mic: for sound collection and voice localization; 151 lines
- keypad: for keypad control; 233 lines
- ultrasound: for obstacle detection; 106 lines
- setting8255: for using 8255 chip controlling motors, LEDs and ultrasound control pin; 144 lines
- delay: for time delays; 58 lines

6. Summary and Conclusion

Our project aims to build up a car with multiple control modes. In Sound Control mode, the car can lead itself toward the sound source. For example, if a person stand at the left side of the car and shout to the car, it will turn left and proceed towards the person. This is also called voice localization. In Keypad Control mode, the car will guided by keypad. In Free Run mode, the car will go straight forward unless it meets an obstacle. When it meet obstacle, it go backwards a variable distance, then turn left and go straight forward again. All these functionalities are achieved by simply using one 8051 micro-controller and other necessary peripheral devices, so this project tests our design ability to the best.

More importantly, we have learnt and experienced a complete flow from planning, implementation to integration and testing. We started this project early in the end of July 2008 and spent the first several weeks testing different implementations and learning various kinds of protocols. The project is divided into three relatively independent parts: Sound control, Keypad control and Ultrasonic Detection. Each team member is in charge of one part and works to continuously refine the functionality in the part assigned. It is not until the last two weeks that we start work extensively to integrate these parts together, and finally come up with a multifunctional sound control car.

Through this project, we also gained valuable hardware and software knowledge. The technology and experience we gained from this project experience will certainly be helpful to our FYP as well as future careers as engineers.

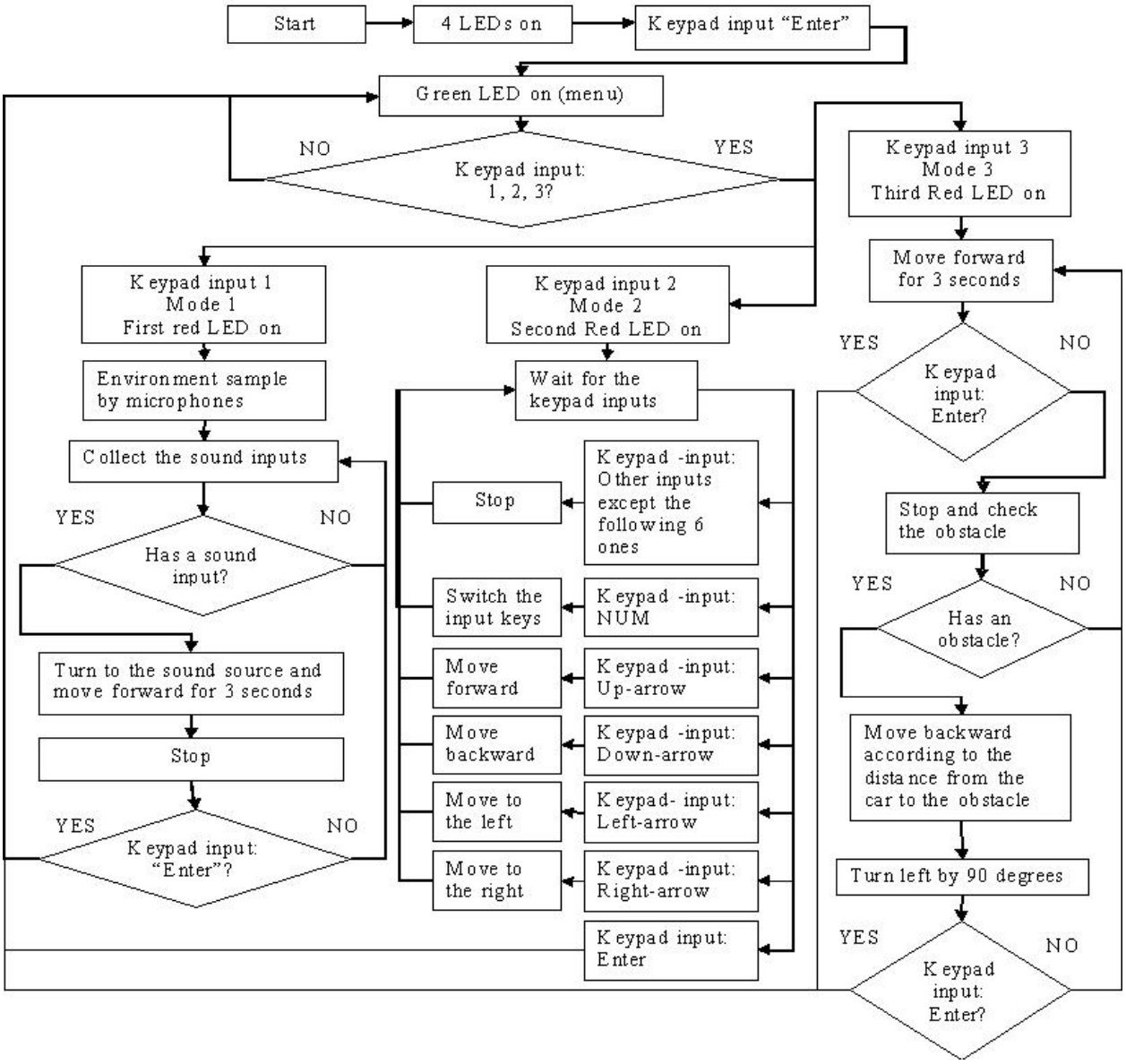


FIGURE 12. Main function flowchart

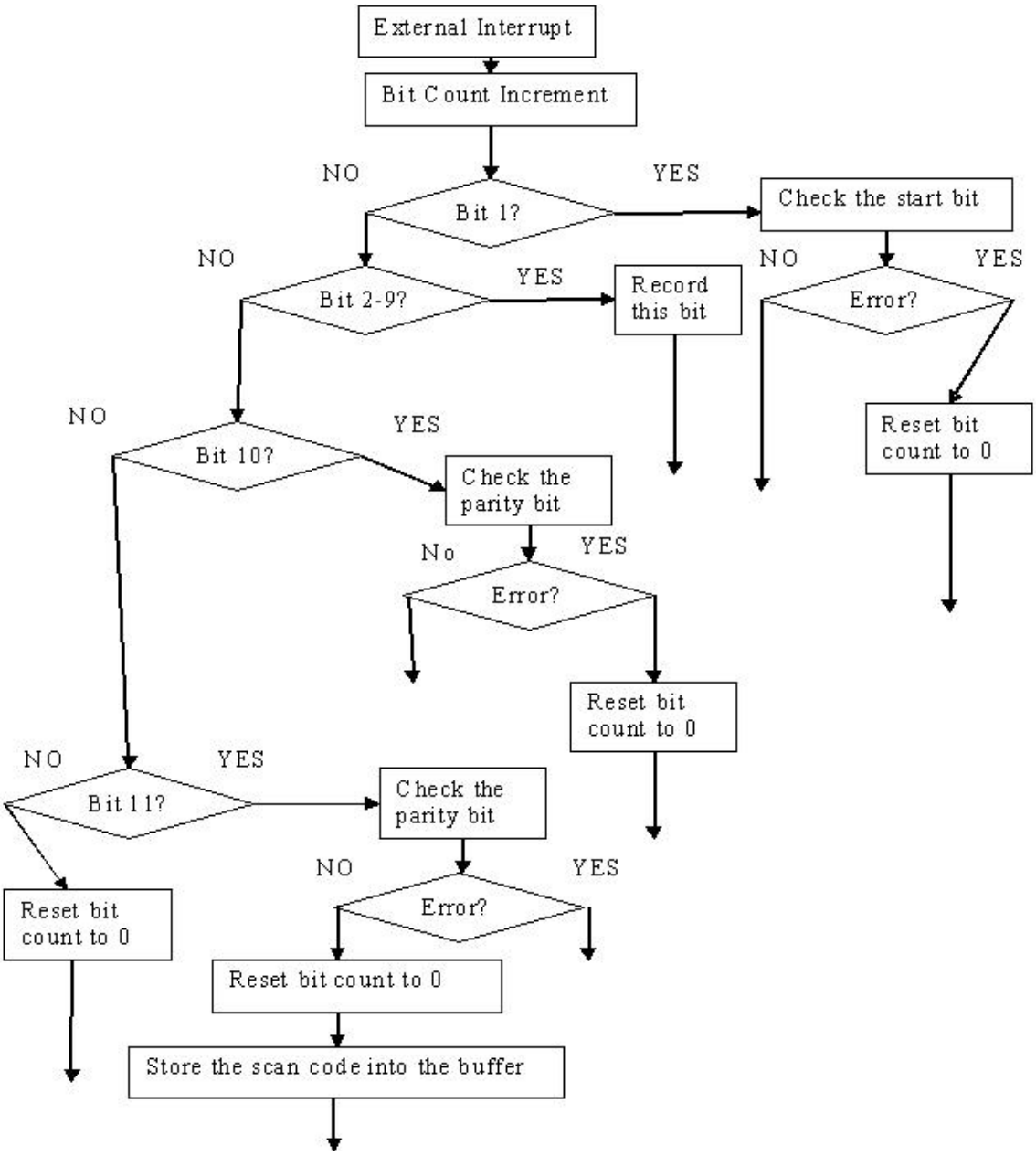


FIGURE 13. Flowchart for keypad to host

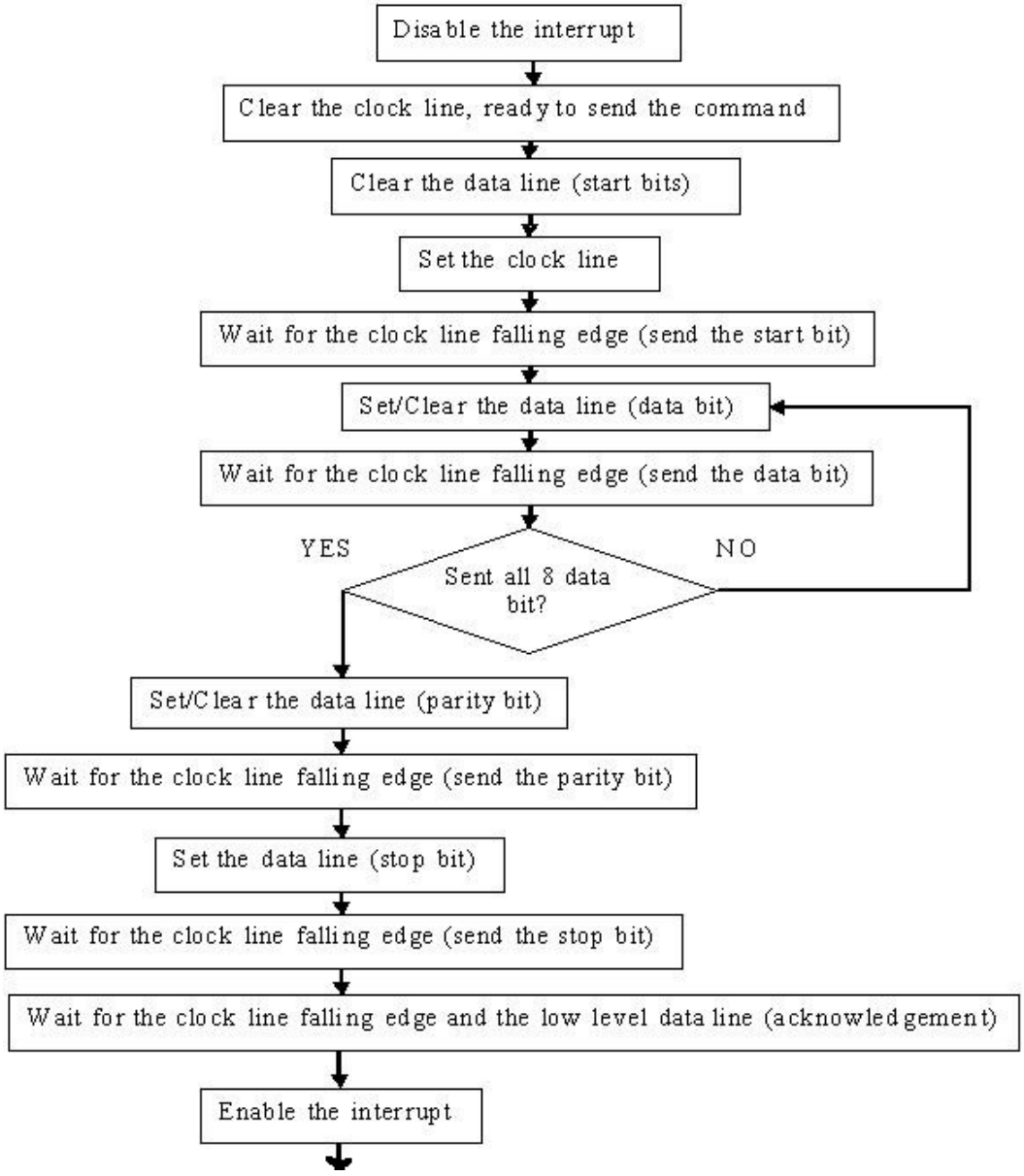


FIGURE 14. Flowchart for host to keypad

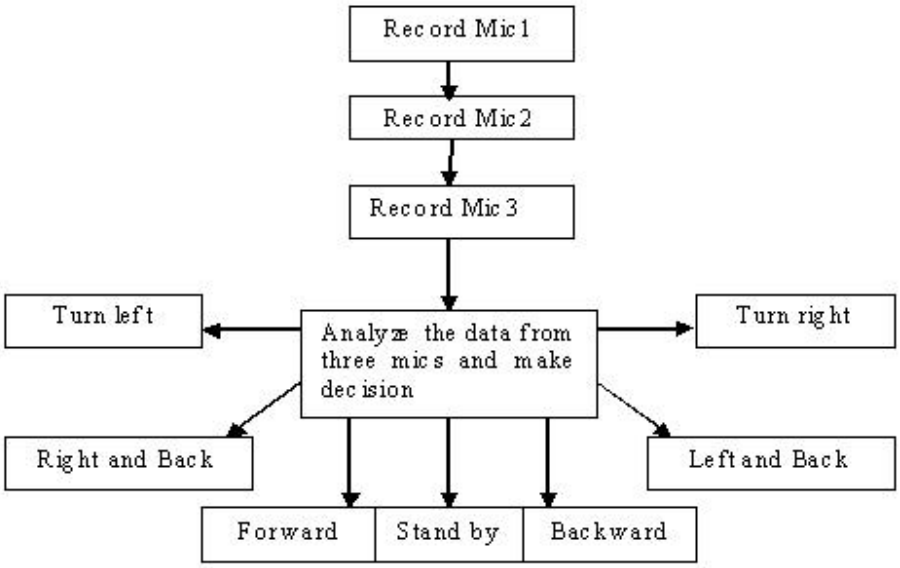


FIGURE 15. Flowchart of sound control

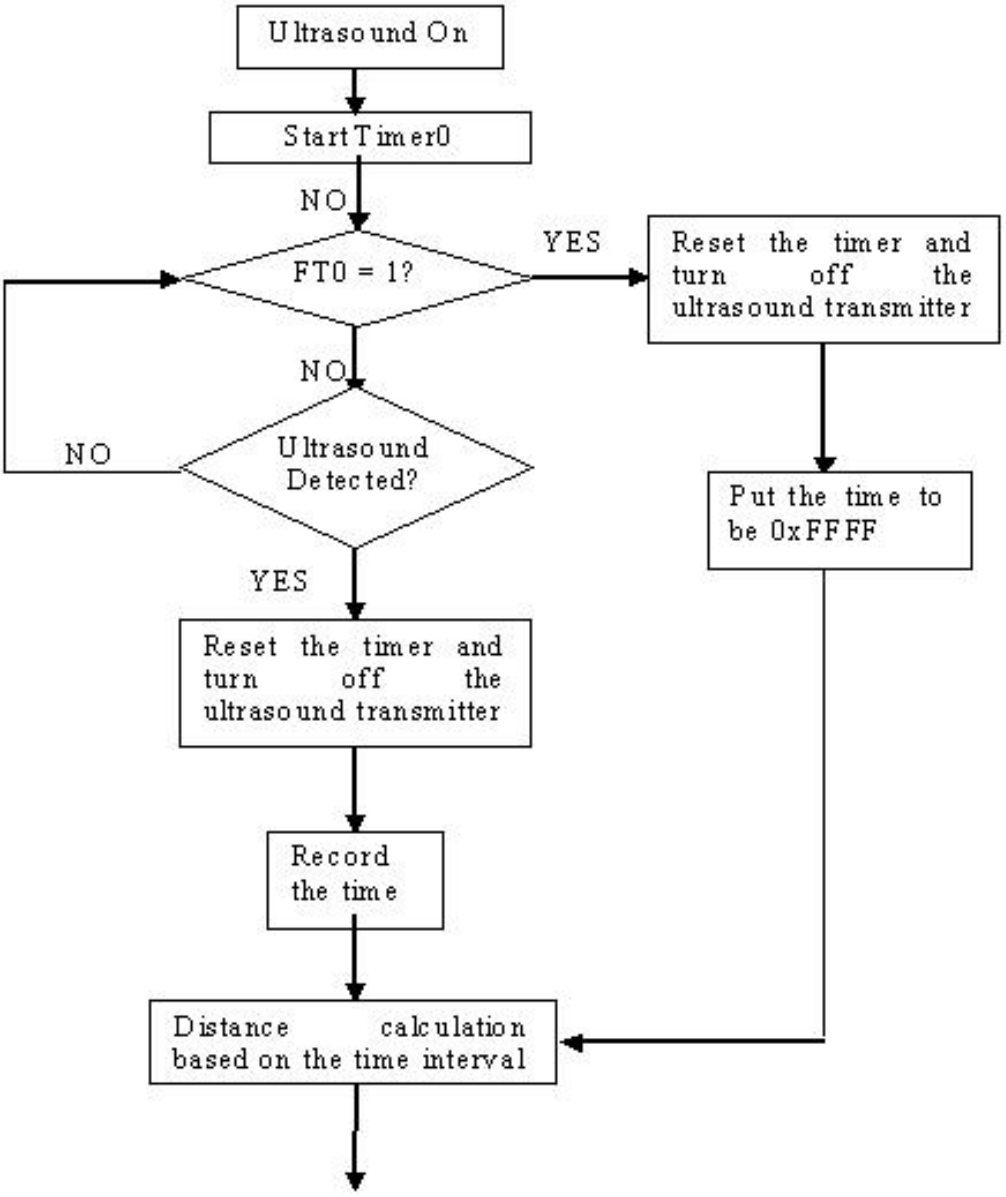


FIGURE 16. Flowchart of ultrasonic control