Lab 2 First Steps to Design

Leqi Ding
dlq991118@sjtu.edu.cn

Ming Cheng
chengming@sjtu.edu.cn

Wanda Guo
wdguo0424@gmail.com

July 25, 2019

Contents

Abstract

As solving any problems should follow a simple to complex process, we also need to follow the same procedure in doing lab 2 – implementing a four-function calculator, which consists of three phases. Lab 2 aims to utilize several peripheral drivers used in Lab1 in more complex design and get more familiar with the C language, the Arduino IDE and Arduino hardware. It requires to finally implement a four-function and three-phase calculator.All the phases use LCD as output and serial monitor as input.Phase 1 only implements the calculation of one-digit numbers while phase 2 takes two-digit numbers into account. What's more, negative numbers are also involved to the calculator in phase 3.

Keywords: four-function calculator, Arduino UNO Board, Serial Monitor, LCD

1 Introduction

This lab has two main purposes. One is to utilize the devices we used in lab 1 to implement a more complicated application. The other one is to learn more about how to design a useful program step by step.

The project is to design a four function calculator in three phases which can executes addition, subtraction, multiplication and division. All the input and output, including the two operands input, the operator and the result of the calculation are displayed on the LCD. Operands and operator are in the 1st row when the result is in the 2nd. We use Arduino UNO board with a Bus Shield on it and an LCD brick to realize it. Phase 1 requires calculation of two one-bit numbers and phase 2 and 3 require calculation of two-bit numbers. Especially, Phase 3 takes negative numbers into consideration.

2 Discussion of the lab

2.1 Brief Design Specification:

The whole project aims to use the Arduino UNO board to design and implement a basic four-function calculator, with the function of performing addition, subtraction, multiplication, and division. The results will be displayed on the LCD (Liquid Crystal Display).

2.1.1 Phase1

In this phase, numbers to be calculated shall be single-digit, positive integers in the range of 0 9. We read each character as input from the Serial Monitor, send it to the Arduino board and display it one by one in the first line of LCD. (The conversion from ASCII to integer is needed here.) The specific kind of arithmetic operation depends on whether the input is '+', '-', '*' or '/'. Especially, when the arithmetic operation is '/', the result will become a floating-point number if possible, keeping two decimal places. And after '=' is inputted, the result will be displayed in the next line of LCD.

2.1.2 Phase2

As a further experiment based on phase1, the numbers to be calculated will be double-digit instead of single-digit. The reading process and displaying process are similar to those in phase1.

2.1.3 Phase3

Two Arduino boards will be used in this phase, with one managing the input from the user, conversion from ASCII to integer, and all arithmetic calculations while the other one managing all display functions. The inputted numbers can be either positive or negative, in the range of 00 99. What's more, the reading process and displaying process are similar to those in previous phases while the results will be displayed on the LCD connected to the second board.

2.2 Hardware Implementation

The schematic diagram of the hardware implementation is as above. On the Arduino board, port 0 to 13 are used to the LCD brick via Bus Shield. The +5V and GND ports are used to connect the board to a voltage source and ground. The Port A4 and A5 are used for communication between two boards when the calculator works in phase 3. Last, in order to enable the whole board to work, it is indispensable to connect the board to PC, enabling the interaction between PC and the board as well.

2.3 Software Implementation

Lab2 is divided into three phases. Phase 2,3 are based on the phase 1. Therefore, we need to start from phase 1 to achieve the simplest case. Initially, we read and analyze the questions, and find the backbone of the questions. Then we will design the methods that can slove the problem we meet. After designing the idea with pseudocode, we write it into the computer code and run it. We observe the results of the codes, use the debugger to debug if possible, and finally solve the whole problem. In the meanwhile, if we need to use the knowledge that has not been learnt yet, we will turn to the references for help and apply related functions and algorithms to the code. In phase 2 and phase 3, we need to consider the more complex case on the basis of phase 1.The solution process is the same as above. Figure shows the structural diagram on software.

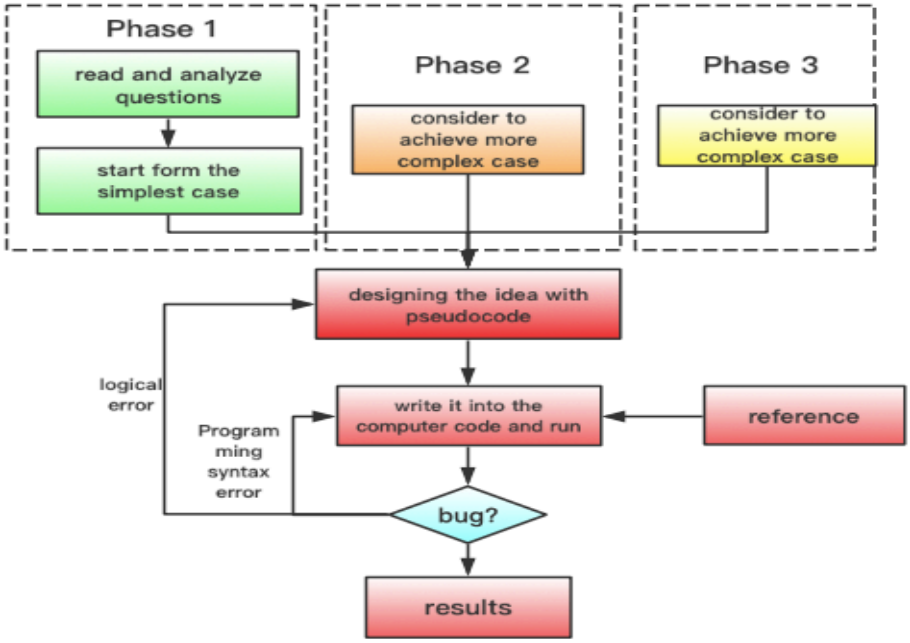

Figure 1: Structural diagram of software

There are three phases in this project and later phases are based on previous ones. Since there is a complete problem in each phase, we need to follow the entire process of solving a problem in each phase: from reading, analyzing to finally getting the results. The entire process of solving a problem is shown in the figure below.

In addition, since phase2, 3 are based on phase1. The code for phase1 is the basis of the whole project. In phase1, we defined variable incomingByte to store the incoming byte. When the incoming byte is a number, it will be displayed in lcd directly. lcd.print(incomingByte - '0'); can achieve the display of the number. In addition, in order to distinguish the first number and the second number, we define variable Operation. The Initial value of the variable is 0 which means operator has not been entered yet. Then, if the variable's ASCII code value is equal to one of the four operators, the variable Operation will be given a new value,

which represents the type of the operation. Finally, to calculate the value of the result, we used 'switch' structure. According to the type of the operator, the corresponding calculation is performed to obtain the value of result.

In phase2, we use an array to store the input data. For different input cases, we adopt the idea of classification discussion. The position where the '=' and the operator appear are used as the basis for the classification. For the double digit integers, we store most significant digit and least significant digit respectively. In phase3, we made improvements on the basis of phase2. In order to judge whether the first number is positive or negative, we only need to judge whether the input of the starting position is '-'. Since the second number and the first number are separated by operators, we set a flagofnum variable to indicate whether an operator is present, so that we can know the positive or negative of the second number.

3 Test Plan

For three different phases, we have three specific test plans.

In phase1,what we need to verify is that when two positive integers with a range of 0 9 and the operator are inputted whether the input is displayed on the first line and the result of the operation is displayed on the second line.

In phase2, what we need to verify is that when two positive integers with a range of 00 99 and the operator are inputted whether the input is displayed on the first line and the result of the operation is displayed on the second line.

In phase3, there are two Arduino UNO microprocessors to be used: one is master, the other is slave. We need to verify whether the master can accept input and manage all arithmetic calculations successfully. In the meanwhile, we need to verify whether the slave can manage all display functions.

4 Presentation, discussion and analysis of the results

All of the calculation results are handled as the method below. If the result is an integer, it will be displayed without fractional part. If it contains fractional part, we reserve two decimal fractions when displaying the result.

Phase 1: In this part, we need to test the basic four function and try some special cases to ensure the robustness of the program. When we input '3+5=' in the serial monitor, the LCD displays the same characters on 1st row and the character '8' on 2nd row. To test whether it can output negative results, we input '3-5='. Then the LCD displays the same character on 1st row and '-2' on 2nd row. When we input '5*3=', the input is shown on 1st row and on 2nd row there is '15'. To test the function of division, we input '9/7=', and the result is the input on 1st row and '1.29' on the 2nd row. When we input '5/8=', '5/8=' is on 1st row and '0.63' the 2nd row. The test results mentioned above prove that our design works without mistake.

Phase 2: In this part, we also need to test four basic function and special cases. Initially, we input two double-digit numbers to check the basic function, such as '67+20=' and '97/19='. Then we get the satisfactory result such as '87' and '5.11' on the 2nd row of LCD. What to emphasize is that our program accepts the input in phase 1 and the input including a one-digit number and a two-digit number, which is essential to verify our design. After testing all typical cases, we get the same results as phase 1 in the former case and the same results as theoretical values in the latter case.

Phase 3: First, we test all the cases of phase 2 and get satisfactory results. Second, we test several special cases. One is the four functions for two negative numbers and the other one is four functions of a negative number and a positive number. For the first one, we input expressions such as '-12+-32=' and '-98/-23='. Then we get the result '-44' and '4.26' on the 2nd row of LCD. For the second one, we input '-12+34=', '-23/52=', etc. Then we get the original expression in the 1st row and calculation result '22' and '-0.44' ,which are on the 2nd row. All these results meet our requirement.

5 Analysis of any errors

In this experiment, we mainly encountered two problems.

The first problem is that the operator entered in the serial monitor does not display properly. To solve this problem, we used the Serial.read() function to read the ASCII value of the input character. Thus, when implementing the display function, we only need to compare the ASCII code value of the input character and the ASCII code value of the operation symbol to know the operation type and output the operation result properly.

The second problem is that when the division operation is performed, if the result of the operation is a floating-point number, only the integer part of the display will be displayed. The reason for this problem is that we don't realize that the results of two integers are not necessarily integers. In order to solve this problem, the first thing we need to do is to define the type of two numbers inputted and the output result as double. Then we use the knowledge related to the type conversion to determine whether the result of the operation is an integer or a decimal. If it is an integer, it is directly outputted; while the precision is two digits after the decimal point if it is a floating point number.

6 Summary and Conclusion

In this project, we successfully achieve utilizing several of the peripheral drivers that designed in the previous lab in a more complex design - implementing the arithmetic operation of two numbers, either positive or negative, single-digit or double-digit. We use the Serial Monitor as input and the LCD as output.

The project consists of three phases. In the first one, the numbers to be calculated are single-digit and positive while they are changed into double-digit and positive in the second one. In the last phase, however, two Arduino boards are used, with one is to implement reading input from the user and all arithmetic calculations while the other one managing all display functions. In addition, the numbers to be calculated in this phase can be either positive or negative, in the range of 00 99. Also, the conversion from ASCII to integer is needed in all phases.

In phase 3, we use two Arduino UNO boards to implement the process of information sending and receiving. According to the references, there is some knowledge about the communicating process of two boards:

Two boards are programmed to communicate with one another in a Master Writer/Slave Receiver configuration via the I2C synchronous serial protocol. Several functions of Arduino's Wire Library are used to accomplish this.

The I2C protocol involves using two lines to send and receive data: a serial clock pin (SCL) that the Arduino or Genuino Master board pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two devices. As the clock line changes from low to high (known as the rising edge of the clock pulse), a single bit of information - that will form in sequence the address of a specific device and a command or data - is transferred from the board to the I2C device over the SDA line. When this information is sent - bit after bit -, the called upon device executes the request and transmits it's data back - if required - to the board over the same line using the clock signal still generated by the Master on SCL as timing.

In this project, we get more familiar with the Arduino UNO board and the ability to solve problems following the process from simple to complex is also improved. As a recommendation of the project, we think it essential to have more introductive content of Arduino and correct some errors of instructions in the Lab Intro document.

7 Appendix

Code for Phase1

```
#include <LiquidCrystal.h>
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7, d8 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7, d8);
```

```
int incomingByte = 0; // for incoming serial data
double number1,number2,result;
int yunsuan = 0;//+:1 -:2 *:3 /:4
int test;
void setup()
{
Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
lcd.begin(16, 2);
}
void loop()
{
// send data only when you receive data:
if (Serial.available() > 0) {
// read the incoming byte:
incomingByte = Serial.read();
// say what you got:
Serial.print("I received: ");
if (incomingByte == 43)
{
Serial.println('+');
lcd.print('+');
yunsuan = 1;
}//+
else if(incomingByte == 45)
{
Serial.println('-');
lcd.print('-');
yunsuan = 2;
}//-
else if(incomingByte == 42)
{
Serial.println('*');
lcd.print('*');
yunsuan = 3;
}//*
else if(incomingByte == 47)
{
Serial.println('/');
lcd.print('/');
yunsuan = 4;
}// /
else if(incomingByte == 61)
{
Serial.println('=');
lcd.print('=');
lcd.setCursor(0,1);
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
```

```
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
lcd.print(result);
else
lcd.print(test);
} // =
else
{
Serial.println(incomingByte - '0');
lcd.print(incomingByte - '0');
if(yunsuan == 0)
number1 = incomingByte - '0';
else
number2 = incomingByte - '0';
}
}
}
```

Code for Phase2

```
#include <LiquidCrystal.h>
const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7, d8 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7, d8);

int incomingByte = 0; // for incoming serial data
double number1,number2,result=0;
int yunsuan = 0;//+:1 -:2 *:3 /:4
int test;
int data[10]=0;//store the expression of formulation
int start = 0;
void setup()
{
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
  lcd.begin(16, 2);
}
void loop()
{
  // send data only when you receive data:
  if (Serial.available() > 0)
  {
  incomingByte = Serial.read();
  data[start] = incomingByte;
  start++;
  if((incomingByte - '0') >= 0 && (incomingByte - '0') <= 9)
  lcd.print(incomingByte - '0');
  else if(incomingByte == 43)
{
Serial.println('+');
lcd.print('+');
yunsuan =1;
} else if(incomingByte == 45)
```

```
{
Serial.println('-');
lcd.print('-');
yunsuan =2;
}
else if(incomingByte == 42)
{
Serial.println('*');
lcd.print('*');
yunsuan =3;
}
else if(incomingByte == 47)
{
Serial.println('/');
lcd.print('/');
yunsuan =4;
}
else if(incomingByte == 61)
{
Serial.println('=');
lcd.print('=');
if(data[5] == 61)//number1,number2 are double digit
{
number1 = (data[0]-'0') * 10 + (data[1]-'0');
number2 = (data[3]-'0') * 10 + (data[4]-'0');
lcd.setCursor(0,1);
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
lcd.print(result);
else
lcd.print(test);
} // =
else if(data[3] == 61)//number1,number2 are single digit
{
number1 = (data[0]-'0');
number2 = (data[2]-'0');
lcd.setCursor(0,1);
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
```

```
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
lcd.print(result);
else
lcd.print(test);
} // =
else if(data[1] -'0' > 9 || data[1] -'0' < 0)//number1 is single digit, number2 is double digit
{
number1 = data[0]-'0';
number2 = (data[2]-'0') * 10 + (data[3]-'0');
lcd.setCursor(0,1);
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
lcd.print(result);
else
lcd.print(test);
} // =
else if(data[2] -'0' > 9 || data[2] -'0' < 0)//number1 is double digit, number2 is single digit
{
number1 = (data[0]-'0') * 10 + (data[1]-'0');
number2 = data[3]-'0';
lcd.setCursor(0,1);
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
```

```
if((double)test != result)
lcd.print(result);
else
lcd.print(test);
} // =
}
}
```

Phase 3

```
  //master
#include <Wire.h>
int incomingByte = 0; // for incoming serial data
double number1,number2,result=0; int yunsuan = 0;//+:1 -:2 *:3 /:4
int test;
int data[10]={0};//store the expression of formulation
int start = 0;
int tmp;
int polar1 = 0,polar2 = 0;//polarity of two numbers,0:positive;1:negative
int flagofnum = 0;//0 is number1, 1 is number2
void setup()
{
Wire.begin();
Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}
void loop()
{
// send data only when you receive data:
if (Serial.available() > 0)
{
incomingByte = Serial.read();
if(start == 0 && incomingByte == 45)
{
polar1=1;
Wire.beginTransmission(4); // transmit to device #4
Wire.write('-'); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
else if(flagofnum == 1 && incomingByte == 45)
{
polar2=1;
Wire.beginTransmission(4); // transmit to device #4
Wire.write('-'); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
else
{
data[start] = incomingByte;
start++;
tmp=start;
if((incomingByte - '0') >= 0 && (incomingByte - '0') <= 9)
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write(incomingByte - '0'); // sends five bytes
Wire.endTransmission();
```

```
}
else if(incomingByte == 43)
{
Serial.println('+');
Wire.beginTransmission(4); // transmit to device #4
Wire.write('+'); // sends five bytes
Wire.endTransmission(); // stop transmitting
yunsuan =1;
flagofnum = 1;
}
else if(incomingByte == 45)
{
Serial.println('-');
Wire.beginTransmission(4); // transmit to device #4
Wire.write('-'); // sends five bytes
Wire.endTransmission(); // stop transmitting
yunsuan =2;
flagofnum = 1;
}
else if(incomingByte == 42)
{
Serial.println('*');
Wire.beginTransmission(4); // transmit to device #4
Wire.write('*'); // sends five bytes
Wire.endTransmission(); // stop transmitting
yunsuan =3;
flagofnum = 1;
}
else if(incomingByte == 47)
{
Serial.println('/');
Wire.beginTransmission(4); // transmit to device #4
Wire.write('/'); // sends five bytes
Wire.endTransmission(); // stop transmitting
yunsuan =4;
flagofnum = 1;
}
else if(incomingByte == 61)
{
Serial.println('=');
Wire.beginTransmission(4); // transmit to device #4
Wire.write('='); // sends five bytes
Wire.write('h');
Wire.endTransmission(); // stop transmitting
}
if(data[5] == 61)//number1,number2 are double digit
{
if(polar1 == 0)
number1 = (data[0]-'0') * 10 + (data[1]-'0');
else
number1 = -(data[0]-'0') * 10 - (data[1]-'0');
if(polar2 == 0)
number2 = (data[3]-'0') * 10 + (data[4]-'0');
else
```

```
number2 = -(data[3]-'0') * 10 - (data[4]-'0');
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
Wire.beginTransmission(4); // transmit to device #4
Wire.write((int)result); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
else
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write(test); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
} // =
else if(data[3] == 61)//number1,number2 are single digit
{
if(polar1 == 0)
number1 = (data[0]-'0');
else
number1 = -(data[0]-'0');
if(polar2 == 0)
number2 = (data[2]-'0');
else
number2 = -(data[2]-'0');
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write((int)result); // sends five bytes
Wire.endTransmission(); // stop transmitting
```

```
}
else
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write(test); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
} // =
else if(data[1] -'0' > 9 || data[1] -'0' < 0)//number1 is single digit, number2 is double digit
{
if(polar1 == 0)
number1 = data[0]-'0';
else
number1 = -(data[0]-'0');
if(polar2 == 0)
number2 = (data[2]-'0') * 10 + (data[3]-'0');
else
number2 = -(data[2]-'0') * 10 - (data[3]-'0');
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write(int(result)); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
else
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write(test); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
} // =
else if(data[2] -'0' > 9 || data[2] -'0' < 0)//number1 is double digit, number2 is single digit
{
if(polar1 == 0)
number1 = (data[0]-'0') * 10 + (data[1]-'0');
else
number1 = -(data[0]-'0') * 10 - (data[1]-'0');
if(polar2 == 0)
number2 = data[3]-'0';
else
number2 = -(data[3]-'0');
```

```
switch(yunsuan)
{
case 1: result = number1 + number2;
break;
case 2: result = number1 - number2;
break;
case 3: result = number1 * number2;
break;
case 4: result = number1 / number2;
break;
default: break;
}
test = (int)result;
if((double)test != result)
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write((int)result); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
else
{
Wire.beginTransmission(4); // transmit to device #4
Wire.write(test); // sends five bytes
Wire.endTransmission(); // stop transmitting
}
} // =
}
}
}
  //slave
#include <Wire.h>
#include <LiquidCrystal.h> const int rs = 2, en = 3, d4 = 4, d5 = 5, d6 = 6, d7 = 7, d8 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7, d8);
void setup()
{
Wire.begin(4); // join i2c bus with address #4
Wire.onReceive(receiveEvent); // register event
Serial.begin(9600);
lcd.begin(16, 2);
}
void loop()
{
// put your main code here, to run repeatedly:
delay(100);
}
void receiveEvent(int howMany)
{
while(0 <Wire.available()) // loop through all but the last
{
char c =Wire.read(); // receive byte as a character
if(c != 'h')
{
lcd.print(c);
Serial.print(c); // print the character
```

```
}
else
lcd.setCursor(0,1);
}
}
```

8 Reference

[1] The C Programming Language, Brian W. Kernighan, Dennis M. Ritchie
[2] https://www.arduino.cc [Online]