University of Washington
EE 299 Lab 2
First Steps to Design

Mitchell Szeto
Bert Zhao
Feifan Qiao

TABLE OF CONTENTS

ABSTRACT

This lab tasked us to create a calculator that can handle integer arithmetic and helped us increase our understanding of the project development cycle. The way that the lab was structured guided us to develop the calculator in small steps which made the project easier to manage. Most of the errors that we came across occured when handling edge cases and sending the different data types through I2C. But we managed to handle them fairly easily. This lab helped us practice using the I2C protocol and standard debugging practices such as having intermediate serial outputs through the calculations to a final solution. These skills will be important as we move on lab 3 and the final project for this class.

INTRODUCTION

The main goal of this lab is to create a four function calculator using two Arduino Unos, serial monitor, and an LCD screen. This lab covers two main goals: work with peripherals in a more complex manner than in Lab 1 and begin to learn the formal design process during the development cycle.

DISCUSSION OF LAB

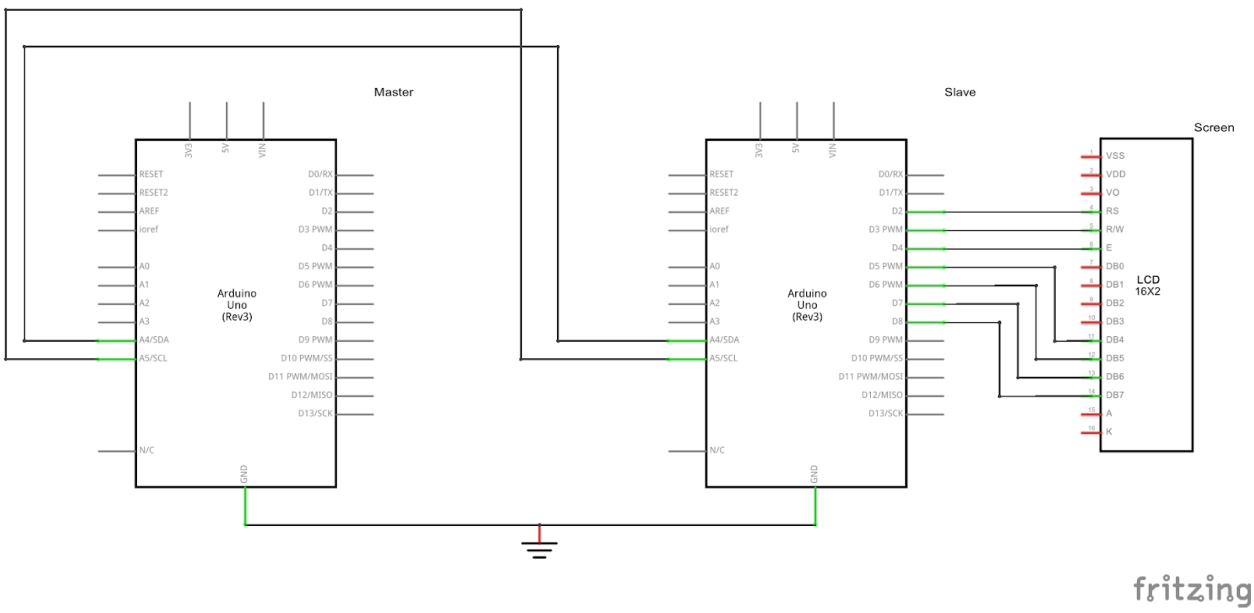
Design Specification

The completed four function calculator needs to support integer addition, subtraction, multiplication, and division. The numbers being calculated can be negative or positive double digit numbers. The functionality of this design is split into three parts: user input, arithmetic calculation, and display user input and calculation results.

For user input, the user needs to have the ability to input a simple equation through serial monitor. The user can input the equation in two ways: writing the equation in a single line like 3+3= or entering in each character on a different line. For arithmetic calculation, integer math will be used so there won't be any decimals. Finally for the display, the user entered equation will display on the first line of the LCD and the answer will be shown on the second line. The program also needs to support two Arduinos that are connected through the I2C protocol. The master will handle user input and the calculations. The slave will display the user input and calculation results.

Hardware Implementation

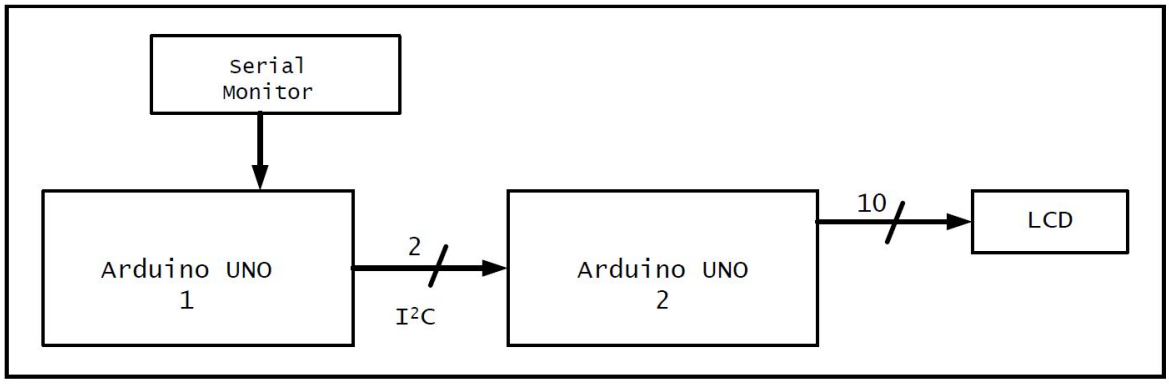
The setup we are using is comprised of a PC, two Arduinos and a LCD display. The two Arduinos, one as master device and the other one as slave device, are connected via analog pin 4 and 5 and communicate through the I^2C protocol. Both Arduinos use the same computer as ground. The LCD screen is connected to the slave Arduino via BUS 1. The master Arduino is plugged into the computer and is responsible for getting inputs from serial monitor.



Software Implementation

Master device is responsible for processing the user inputs and deliver those inputs, along with answer to the calculation, to the slave device. Master's code can then be broken down into three parts, receiving inputs, calculating the answer, and communicate to the slave device. For receiving inputs, it is implemented using the serial monitor. Serial monitor communication is started by setting the baud rate to 9600. After it is set up, user input will be delivered to the device as long as the input is not space or line feed. The program calculates the answer first by filtering out the two numbers, their sign, and the operator. Once this is done, the program performs the calculation using generic C code. After the result is calculated, it is communicated to the slave device using the I2C protocol.

The slave device is responsible for receiving information from the master device and sending the information to the LCD. The slave receives information from the master in two parts, and they are separated by the equals character. All data sent to the slave before and including the equals sign, is sent and received as a character. The slave can receive this information well within the one byte limit of a single receive event. On the other hand, the final answer is sent and received as two bytes. This is necessary because numbers over 255 need two or more bytes in a single receive event for the slave to completely receive the information. The slave also handles the divide by zero error and out of bounds error, properly displaying the error on the LCD if they occur.



See Appendices for code screenshot

ANALYSIS OF ANY ERRORS

The calculator that we programmed in this lab is very preliminary in terms of its functionality. The C language itself handles addition, subtraction, multiplication, and division, and since the calculation is handled by the generic language, there is very little room for error. However, there can still be errors in the way our program processes user inputs, especially in some edge cases. One of the edge cases we encountered is the division by 0 error. If we perform a division by 0 calculation in C, the C language is not going to return an error, instead, it is going to output some random number. This kind of error is catastrophic since the most important part of a calculator is its accuracy. To handle this situation, we wrote an if statement that will return an error message once a division by 0 calculation is performed.

Another edge case that we encountered is overflow/out of bounds. In the spec, upper bound for a user input is 99 while the lower bound is -99. While programming the calculator, we had to be conscious of the upper and lower bound even though this bound is way smaller than INT_MAX and INT_MIN. We created an if statement that tests if user inputs are in bounds and will return out of bounds message if not.

SUMMARY AND CONCLUSION

In this lab, we solved the design problem of creating a four function calculator. The serial monitor served as our primary input and the LCD served as our primary output. We worked more in depth with the I2C protocol between two Arduinos and how to send different types of data between the two Arduinos. We utilized more standard debugging practices in this lab such as having intermediate serial outputs through the calculations to a final solution. These skills will be important as we move on lab 3 and the final project for this class.

APPENDICES

Calculator

```
/*
 * EE 299 Lab 2 Master Code
 *
 * Feifan Qiao, Mitchell Szeto, Bert Zhao
 *
 * This is the master version. It asks user for a simple mathmatic operation
 * and sends the result to the slave device
 *
 * Last modified: 10/29/2018
 */

#include <Wire.h>

const int LINE_FEED = 10;     // ascii code for line feed
const int SPACE = 32;         // ascii code for space

int incomingByte = 0;         // for incoming serial data

char data = 0;
char prevData = 0;
bool firstNegative = false;   // checks if the first number is negative
bool secondNegative = false;  // checks if the second number is negative
int firstNum = 0;             // stores the value of the first number
int secondNum = 0;            // stores the value of the second number
bool onFirst = true;          // checks if current step is on the first number
char operation = 0;           // stores the operator
bool clearScreen = false;     // checks if it's time to clear screen
```

```
void setup() {
  Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
  Wire.begin();
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    if (incomingByte != LINE_FEED && incomingByte != SPACE) {
      prevData = data;
      data = (char)incomingByte;

      if (clearScreen) {
        clearScreen = false;
      }

      sendData(data);

      if (onFirst) {
        if (prevData == 0 && data == '-') {  // checks if the first input is negative sign
          firstNegative = true;
        } else if (!((prevData != '+' || prevData != '-' || prevData != '*' || prevData != '/') &&
            (data == '+' || data == '-' || data == '*' || data == '/'))) {
          firstNum *= 10;
          firstNum += (data - 48);
        } else {
          operation = data;
```

```
          onFirst = false;
        }
      } else {
        if ((prevData == '+' || prevData == '-' || prevData == '*' || prevData == '/') &&
            (data == '-')) {  // checks if the second number is negative
          secondNegative = true;
        } else if ((data != '+' || data != '-' || data != '*' || data != '/') && data != '=') {  // checks if input is a number
          secondNum *= 10;
          secondNum += (data - 48);
        }

        if (data == '=') {
          calculate();
          reset();
          clearScreen = true;
        }
      }
    }
  }
}

// calculates the final result and calls sendDataInt() function
void calculate() {
  if (firstNegative) {
    firstNum *= -1;
  }
```

```
  if (secondNegative) {
    secondNum *= -1;
  }

  if (operation == '+') {
    sendDataInt((firstNum + secondNum));
  } else if (operation == '-') {
    sendDataInt((firstNum - secondNum));
  } else if (operation == '/') {
    if (0 == secondNum) {
      sendDataInt(9802);  // add some comments
    } else {
      sendDataInt((firstNum / secondNum));
    }
  } else if (operation == '*') {
    sendDataInt((firstNum * secondNum));
  }
}
```

```
// resets all the variable values
void reset() {
  data = 0;
  prevData = 0;
  firstNegative = false;
  secondNegative = false;
  firstNum = 0;
  secondNum = 0;
  onFirst = true;
  operation = 0;
}

// send inputs as characters
void sendData(char data) {
  Wire.beginTransmission(4);
  Wire.write(data);
  Serial.println(data);
  Wire.endTransmission();
}
```

```
// send answer as two bytes
void sendDataInt(int dataInt) {
  byte data[2];
  data[0] = (dataInt >> 8) & 0xFF;
  data[1] = dataInt & 0xFF;
  Serial.println(dataInt);
  Wire.beginTransmission(4);
  Wire.write(data, 2);
  Wire.endTransmission();
}
```

StandAloneCal

```
/*
 * EE 299 Lab 2 Stand alone calculator
 *
 * Feifan Qiao, Mitchell Szeto, Bert Zhao
 *
 * This is a stand alone calculator that takes user input
 * and prints calculation result on a lcd screen
 *
 * Last modified: 10/30/2018
 */
#include <LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7, 8); // bus 1

const int LINE_FEED = 10;     // ascii code for line feed
const int SPACE = 32;         // ascii code for space

int incomingByte = 0;         // for incoming serial data

char data = 0;
char prevData = 0;
bool firstNegative = false;   // checks if the first number is negative
bool secondNegative = false;  // checks if the second number is negative
int firstNum = 0;             // stores the value of the first number
int secondNum = 0;            // stores the value of the second number
bool onFirst = true;          // checks if current step is on the first number
char operation = 0;           // stores the operator
bool clearScreen = false;     // checks if it's time to clear screen
```

```
void setup() {
  lcd.begin(16, 2);
  Serial.begin(9600);         // opens serial port, sets data rate to 9600 bps
  lcd.display();
}

void loop() {
  if (Serial.available() > 0) {
    incomingByte = Serial.read();
    if (incomingByte != LINE_FEED && incomingByte != SPACE) {
      prevData = data;
      data = (char)incomingByte;

      if (clearScreen) {
        lcd.clear();
        lcd.setCursor(0, 0);
        clearScreen = false;
      }

      lcd.print(data);

      if (onFirst) {
        if (prevData == 0 && data == '-') {
          firstNegative = true;
        } else if (!((prevData != '+' || prevData != '-' || prevData != '*' || prevData != '/') &&
            (data == '+' || data == '-' || data == '*' || data == '/'))) {
          firstNum *= 10;

        firstNum += (data - 48);
      } else {
        operation = data;
        onFirst = false;
      }
    } else {
      if ((prevData == '+' || prevData == '-' || prevData == '*' || prevData == '/') &&
          (data == '-')) {  // checks if the second number is negative
        secondNegative = true;
      } else if ((data != '+' || data != '-' || data != '*' || data != '/') && data != '=') {  // checks if input is a number
        secondNum *= 10;
        secondNum += (data - 48);
      }

      if (data == '=') {
        calculate();
        reset();
        clearScreen = true;
      }
    }
  }
}
```

```
// calculates and prints the final result
void calculate() {
  lcd.setCursor(0, 1);  // sets the cursor on the second row

  if (firstNegative) {
    firstNum *= -1;
  }

  if (secondNegative) {
    secondNum *= -1;
  }

  if (operation == '+') {
    lcd.print(firstNum + secondNum);
  } else if (operation == '-') {
    lcd.print(firstNum - secondNum);
  } else if (operation == '/') {
    if (0 == secondNum) {
      lcd.print("Div by 0 error");
    } else {
      lcd.print(firstNum / secondNum);
    }
  } else if (operation == '*') {
    lcd.print(firstNum * secondNum);
  }
}
```

```
// resets all variable values
void reset() {
  data = 0;
  prevData = 0;
  firstNegative = false;
  secondNegative = false;
  firstNum = 0;
  secondNum = 0;
  onFirst = true;
  operation = 0;
}
```