

MREN 178 - Lab 1

Introduction to Arduino, LCD Keypad Shield, and Linked Lists

Jano Fu

Matthew Pan, PhD

A. Introduction

A.1 Objectives

Welcome to Lab 1. The goal of this lab is to explore some of the basic C programming concepts along with Arduino coding targeted towards an LCD display and button input platform that will be useful for all the other labs in this course. Additionally, you will gain experience with pointers, arrays, and linked list data structures taught in MREN 178 lectures.

You will note that some of the lab sections ask you to read Arduino sketches (i.e., programs), with the goal of understanding how the code works. These programs illustrate some of the basic ideas discussed in lectures of the course. Besides reading the source code, you should compile and run the programs and make experimental changes to program with the 1602 LCD Keypad Shield and Arduino to provide some more in-depth programming experience. I would suggest that you try running the default program first – so that you get a feeling of what the lab does and can then correlate the code with the program's behaviour.

A.2 Topics Covered

- Arduino hardware setup (Arduino UNO R3 printed circuit board)
- Arduino software setup (Arduino integrated development environment)
- 1602 LCD Keypad Shield interaction
- Arrays
- Pointers
- Dynamic memory allocation – malloc() and free()
- Serial port monitoring
- C Structures
- Linked lists

A.3 Pre-Requisite Knowledge

This lab assumes that you have some basic experience with the Arduino ecosystem and know how to program, compile, and upload sketches to an Arduino board. In particular, you should know:

- the structure of Arduino .ino sketches – that is, that there are always setup and a loop functions in a sketch; and,
- how to set pin modes.

Most importantly, you should have completed the instructions set out in the Lab 1: Prelab document before attempting Lab 1. You will need to have setup the Arduino IDE and the *LiquidCrystal* library before continuing.

A.4 Required Resources

Most everything that you need to complete this lab is included.

From your MREN 178 lab kit, you will need:

- Arduino UNO R3 board
- 1602 LCD Keypad Shield
- Male USB-A to Male USB-B Cable

From the OnQ MREN 178 course site, you will need:

- Lab Instructions (this document)
- MREN 178 – Lab 1: Prelab document (for reference)
- A zip file containing source code for the lab. The packing_list.txt inside the zip will have an inventory of all files and folders that should be included.

You will need to provide:

- A computer with a USB-A port
 - Note: MREN 178 lab instructions are tailored towards Microsoft Windows systems. Although you can program Arduinos using Apple/macOS or Linux computers, TAs may have limited knowledge on how to diagnose any issues you might encounter.
 - Lab computers will also be pre-loaded with the Arduino IDE through the AppsAnywhere system. While there are two options available in AppsAnywhere, you must use the Arduino IDE loader indicated below in order to communicate with the Arduino Uno board. See Figure 1.
- Internet access

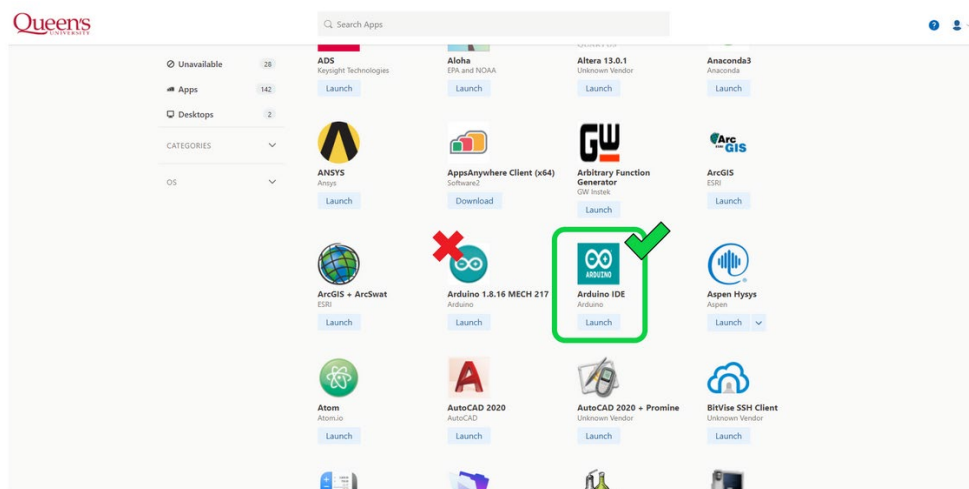


Figure 1. Use the Arduino IDE as highlighted in the Queen's AppsAnywhere system if you are using a lab computer. Do not use Arduino 1.8.16 MECH 217.

B. Lab Procedure

There are numerous steps in this first lab. Some of you may find it easier if you tackle one or two steps and then leave the lab for something else, coming back to it later. You can intermittently work on small pieces of the lab before the due date or complete the lab all at once. Figure out which technique works for you.

Part 1 – Test the 1602 LCD Keypad Shield

Introduction

In this section, you will compile and run sample code to familiarize yourself with the 1602 LCD Keypad shield. This first coding exercise will use the `analogRead()` function to read the value of an analog pin. As mentioned, button inputs on the 1602 LCD Keypad Shield affect the value of pin A0 on the Arduino. Value ranges for each of the buttons are shown in the Prelab.






Follow the setup procedure below.

Setup Procedure

1. Extract the lab01.zip into an appropriate folder on your machine.
2. Navigate to:

[extracted directory]/lab1/lab1_1/

Open the lab1_1.ino Arduino sketch.

3. Connect the Arduino with the LCD keypad shield installed to your computer using the USB cable included in your lab kit.
4. Click  to compile the code. If no errors appear, click  to upload the code to the Arduino board. Note: Clicking  actually compiles and uploads the code. Thus, clicking  is usually not required unless you're trying to debug syntax errors.
5. Click  to open up the serial monitor window to read the output of the program.
6. Press different buttons on the shield to see how the reading differs as different buttons are pressed.
7. Read the source code to better understand the structure of the program. Pay attention to how button responses are read and what are the key functions that interacts with LCD.



Every time you upload code to your Arduino, you'll need to wait a few seconds before the Arduino self-resets and starts running your freshly updated code. You can also press the reset button on the Arduino/shield to manually reset the hardware.



If you don't see anything on your LCD display after plugging in the Arduino, it is likely that the LCD contrast is turned way down; new displays are shipped like this. To rectify this issue, you'll need to use a small screwdriver to turn the small brass screw on the blue potentiometer on the top left-hand side of the LCD Keypad Shield clockwise. This may require several full rotations in order to see the display.

Exercise

Modify the lab1_1.ino code to create the following custom functionality:

- When the Arduino is powered on, the first row should display the text "number:" The second row starting at the 2nd character, should show "100", representing a default number value of 100 as shown in Figure 2.



Figure 2. LCD screen showing initial expected output display.

- Program the buttons such that the value displayed:
 - Increases by 10 when the UP button is pressed
 - Decreases by 10 when the DOWN button is pressed
 - Increases by 1 when the RIGHT button is pressed
 - Decreases by 1 when the LEFT button is pressed
- Once each button is pressed, we should see the updated number value directly on the LCD screen.
- A button press of any duration will increment or decrement the value only once – i.e., pressing and holding buttons should not continuously increment or decrement the value.
- Have a TA check your program functionality, if necessary.

Hints

- Key Bounce (also known as 'chattering') occurs in mechanical switches and can cause one switch press to be detected as multiple presses. Think about how you might choose to de-bounce key presses in case they happen.
- It is always good coding practice to provide comments to keep note of key functionality within your code.

Troubleshooting

If your IDE shows errors when compiling sketches, check your code for any syntax errors. If your code fails to upload to the board, you

1. In the Arduino IDE, go to “tools” tab, make sure you see “Board: Arduino”
2. Go to “Port” Make sure you are running on a operating Serial port, you can tell if there is a bracket saying “(Arduino Uno)”. See 1.3) if there is a problem.

Part 2 – Input/Output Operations and Array Storage

Introduction

Serial operations such as reading and writing form one of the most fundamental ways of interacting with the Arduino from your computer. There are four serial library functions that we'll be using: `Serial.read()`, `Serial.write()`, `Serial.print()`, and `Serial.println()` (shown in Table 1). You should have used some of these functions in the prelab. The Arduino Serial Monitor can be used to see what is being outputted over serial from the Arduino, as well as providing an interface for sending data from the computer to the Arduino.


Table 1. Table of commonly used Serial library functions. More functions can be found in the Arduino Serial reference: <https://www.arduino.cc/reference/en/language/functions/communication/serial/>

Function	Description
<code>Serial.read()</code>	Returns the first byte of incoming serial data available (or -1 if no data is available)
<code>Serial.write()</code>	Writes binary data to the serial port as a byte or series of bytes.
<code>Serial.print()</code>	Prints data to the serial port as human-readable ASCII text.
<code>Serial.println()</code>	Same as above, but appends a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n').

We will be using these commands in this program.

Setup Procedure

1. Make a copy of the lab1_1.ino file you modified in Part 1 and rename it lab1_2.ino. Ensure that the sketch is in its own folder named 'lab1_2' (there should already be an empty lab1_2 folder included in the zip file) – Arduino sketches always have to reside in a folder with the same name.

2. Open the Serial Monitor in the Arduino IDE by clicking the  button on the top right corner or using the Ctrl+Shift+M shortcut.
3. Modify the lab1_2.ino sketch to print out the value displayed on the LCD over serial as well, using `Serial.println()` during every loop. Check if the lcd screen value matches.

Exercise

Modify the code in lab1_2.ino such that the SELECT button stores a user-defined value into an array that stores a maximum of five numbers. The sketch should have the following features:

- The sketch should retain the UP, DOWN, LEFT, and RIGHT button functionality.
- Every time the SELECT button is pressed, insert the value displayed on screen into the next open position in the array.
- Use `Serial.print()` and `Serial.println()` to show all numbers in the array separated by spaces.
- If the number of array items exceeds 5, print “array is full” over both serial and LCD.

Show the resulting Arduino behaviour and the code to the TA to have them check functionality.

Part 3 – Creating a Linked List

Introduction

In this part of the lab, we will be implementing a linked list data structure on the Arduino which can be manipulated by button inputs using the LCD shield. Linked lists (LL) are one of the most common type of data structures. In this lab, each LL node will contain an int and a pointer to the next node, as shown in Figure 3.

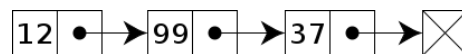


Figure 3. Diagram representation of a linked list.

The code for this part of the lab can be found in the folder ‘lab1_3’. If take a peek at the header file *linked_list.h*, we are able to see how the LL struct is implemented:

```

struct node {
    struct node *p_next_node; // a pointer pointing to the next node
    int data_val; // data stored in each node, this will be an integer ranged 0000 - 9999
};
  
```

Here we have the integer `data_val` storing the relevant data, and `p_next_node` is the pointer to the next node. The first node that is created will have a `p_next_node` pointing to NULL. If a second node is appended to the end of the LL, the first node’s pointer will point to the second node. To keep track of the linked list we have created, we will be

using global `*p_head` and `*p_tail` pointers to keep track – the declaration of these pointers can also be found in the `linked_list.h`.

The code you are provided in this lab creates an interface for creating and interacting with an LL consisting of 4-digit numbers as values. From the menu (shown in Figure 4), using LEFT and RIGHT buttons you can: Add, Show, and Delete items from a linked list. Click SELECT to choose the desired function to be used. When selecting add, you can choose to add items to the back, front, or after a node with a specific value. When selecting delete, you can choose to delete an item with a particular value, or all items.



Figure 4. LCD Keypad Shield showing the menu for this lab component.

Exercises

For this lab, students are required to complete the code in four functions found in “`linked_list.cpp`”

`insert_data_at_head (int val)`

- Inserts a new node with the value `val` at the beginning of the linked list and updates `p_head` and `p_back` appropriately

`insert_data_at_tail (int val)`

- Inserts a new node with the value `val` at the end of the linked list and updates `p_head` and `p_tail` appropriately

`insert_data_at_middle (int search_val, int val)`

- Inserts a new node with the value `val` after a node with value `search_val` only if it exists

`find_and_delete_data (int val)`

- Searches for a node with value `val` and deletes that node if it exists

`delete_all_data()`

- Deletes all nodes

C. Lab 1 Deliverables and Grading Rubric

For your lab submission, you'll need to create a zip archive containing the following files:

- lab1_1.ino [from lab1_1]
- lab1_2.ino [from lab1_2, which you'll need to create]
- linked_list.cpp [from lab1_3]

Do not change the name of these files. In each of these files, fill out the comment block, (e.g., names and student numbers of each member of the lab team, date of submission, etc.). Ensure that any external resources used in completing the lab are listed - this includes anyone outside of your group that provided help to your group. It is totally ok to get help with coding problems, as long as proper credit is given. Failing to do this will be a departure from academic integrity.

The zip archive that each group submits will need to have the default required name lab1_groupXX (so the actual file will be lab1_groupXX.zip) where XX is your assigned group number. For example, if you are group 7, the name of the zip file should be lab1_group07.zip.

Each group must submit one zipped copy of the deliverables on OnQ. The lab deliverables are due before the next lab session.

Component	Item	Mark	Max
lab1_1	Use LCD keypad shield to create user-generated numbers	5	
lab1_2	Use LCD keypad shield to input numbers into an array	5	
lab1_3	Linked list creation		
	insert_data_at_head	7	
	insert_data_at_tail	7	
	insert_data_at_middle	7	
	find_and_delete_data	7	
	delete_all_data	7	
Total			45