This lesson will teach use how to use a keypad on the Intel Galileo. We will also be using the LCD display and learning more about its functionality.

**Keypad**

A keypad is comprised of several buttons. If each button had its own pin, the keypad below would require 16 pins.

![Figure 1: Keypad GH5004-ND](image)

To reduce pin count, keypads commonly have a row/column arrangement as shown below.

![Figure 2: High-Level Connection diagram for Keypad](image)
Each row has a pin (R1-R4), and each column has a pin (C1-C4), for a total of 8 pins. Pressing a button uniquely connects one column pin with one row pin. For example, pressing the upper-left button connects pin C1 with pin R1. Pressing the bottom-right button connects C4 and R4.

To accomplish accepting input from 16 buttons with only 8 pins a technique known as time multiplexing is employed. The idea is simple, we shall use common row wires and common column wires the achieve our lower pin count. This however causes a problem, by sharing the rows and columns we have cross talk. To overcome this, we will selectively enable one column at a time, check the 4 pins connected to that row, and then continue by enabling the next column, repeating the process for all columns. This is time multiplexing -- simultaneous transmission of several messages along a single channel of communication by having those signals transmit at specific times (in this case a specific sequence).

We can get away with this because the microcontroller can operate much faster than humans can react/perceive. In the time it takes a person to press one of the buttons, the microcontroller can make many passes of the keypad to check for input. Thus the process is transparent to the user.

Connect the keypad to the Arduino pins as shown.
Keypad Connections

<table>
<thead>
<tr>
<th>Keypad Pin #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>N/A</td>
<td>R1</td>
<td>R2</td>
<td>R3</td>
<td>R4</td>
<td>C1</td>
<td>C2</td>
<td>C3</td>
<td>C4</td>
<td>N/A</td>
</tr>
<tr>
<td>Galileo Port</td>
<td>N/A</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>A0</td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>N/A</td>
</tr>
</tbody>
</table>

LCD Display Pin Connections

<table>
<thead>
<tr>
<th>LCD PIN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>11-14</th>
<th>15-16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>GND</td>
<td>5V</td>
<td>Potentiometer (10KΩ) thru. to GND</td>
<td>Pin 12</td>
<td>GND</td>
<td>Pin 11</td>
<td>Pins 5 - 2</td>
<td>Vcc-GND</td>
</tr>
</tbody>
</table>

Libraries

Keypad:

Download the GalileoKeypad library provided and extract the zipped contents to your Arduino libraries directory, [https://www.dropbox.com/s/f5e5yymzkmd8tr/GalileoKeypad.zip?dl=0](https://www.dropbox.com/s/f5e5yymzkmd8tr/GalileoKeypad.zip?dl=0)

Two important functions:

Initialization:

```cpp
1  //Keypad
2  #define ROWS 4
3  #define COLUMNS 4
4  #define c1 A0
5  #define c2 A1
6  #define c3 A2
7  #define c4 A3
8  #define r1 11
9  #define r2 10
10 #define r3 9
11 #define r4 8
12 GalileoKeypad keypad(ROWS, COLUMNS, c1, c2, c3, c4, r1, r2, r3, r4);
```
Receiving keypad characters:

```c
getKeypad();
```

34
35
36
37
38

```c
  incoming = keypad.getKeypad();
  if(incoming != '0'){
    lcd.print(incoming);
  }
```  

LCD Screen:

Use the Liquid Crystal library to operate the LCD screen. Refer to Lesson 4 for more detailed instructions on how to set up and begin using the LCD screen.

You will be using more of the libraries functions for this Lesson. Make sure to refer to the library’s documentation here:

https://www.arduino.cc/en/Tutorial/LiquidCrystal

Video Demonstration: https://youtu.be/Epi2FL8c6V0

Exercise 1

Start by printing an exclamation point on the first row and first column of the LCD screen. Using the keypad, print the characters that are pressed onto the LCD screen. Make sure that the cursor increments following every key press so that it wraps around the entire LCD screen overwriting the characters that are currently displayed.

Exercise 2

Use the LCD code, along with a button and/or time delay to display the message "Roads? Where we’re going, we don’t need roads.” The string will not fit on the display all at once, so you will need to come up with some way to paginate or scroll the text.

Note: If your LCD is exceptionally dim, adjust the resistance provided by the potentiometer connected to Pin #3.

Exercise 3

Combine the functionality of the keypad and LCD so when keypad is pressed and released, the character of the button pressed is displayed on the LCD, and stays displayed until a different button press occurs (May be accomplished with two tasks: LCD interface & modified test harness).
Exercise 4 (Challenge)

Notice that you can visually see the LCD refresh each character (display a lengthy string then update to a different lengthy string). Design a system where a single character is updated in the displayed string rather than the entire string itself. Use the functions provided in the Liquid Crystal library.

An example behavior would be to initially display a lengthy string, such as “Congratulations!” The first keypad button pressed changes the first character ‘C’ to the button pressed. The second keypad press changes the second character to the second button pressed, etc. No refresh should be observable during the character update.

Exercise 5 (Challenge)

Using both rows of the LCD display, design a game where a player controlled character avoids oncoming obstacles. Three buttons are used to operate the game.

Criteria:
- Use the cursor as the player controlled character.
- Choose a character like ‘#’, ‘*’, etc. to represent the obstacles.
- One button is used to pause/start the game.
- Two buttons are used to control the player character. One button moves the player to the top row. The other button moves the player to the bottom row.
- A character position change should happen immediately after pressing the button.
- Minimum requirement is to have one obstacle on the top row and one obstacle on the bottom row. You may add more if you are feeling up to the challenge.
- Choose a reasonable movement speed for the obstacles (100ms or more).
- If an obstacle collides with the player, the game is paused, and a “game over” message is displayed. The game is restarted when the pause button is pressed.

Hints:
- Due to the noticeable refresh rate observed when using lcd.print(), make a function that turns off the blink and cursor display while updating. Turn them back on at the end of the function to blink your character.
- LCD cursor positions range between 0, 15 and you will be using lcd.SetCursor(x, y) to modify it.
- As always, dividing the design into multiple, smaller synchSMs can result in a cleaner, simpler design.