Braille Keyboard

A Braille keyboard is a specialist input device that allows the user to type and enter text or instructions for the computer in Braille. Here we connect an Arduino Leonardo with 7 buttons to LCD Nokia5110 to make braille keyboard. Combinations of buttons are representing one braille letter. These combinations are shown in Fig.1 (represents the English and braille letter) and Fig.2 (shows English and Braille numbers).

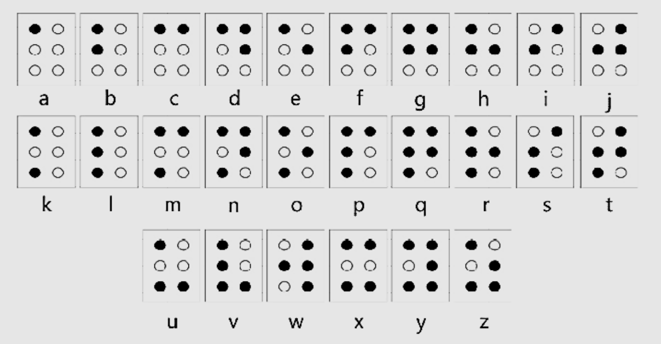


Fig.1 English and braille letter

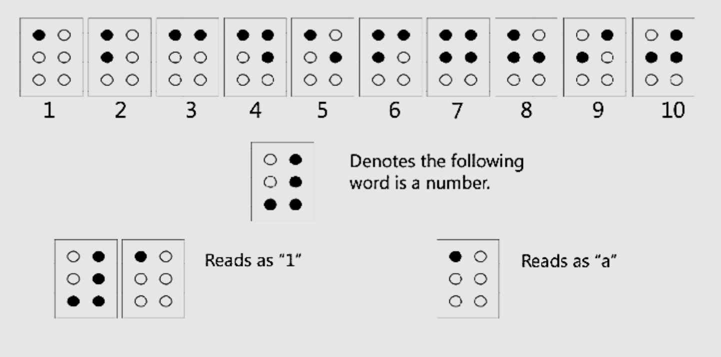


Fig.2 shows English and Braille numbers

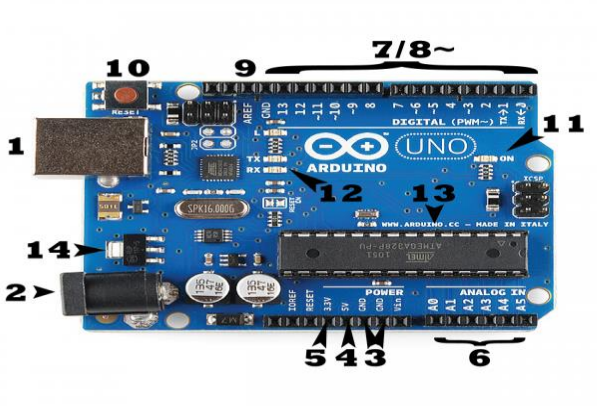
Here we have 7 buttons that 6 of them are assigned to show letters and 7th is assigned for separation of each letter. Send the corresponding letter over the serial console to LCD to show letters.

For example, according to Fig.1 for showing letter “A” we should press button number 1 and for letter “B” press button number 1 and 2 and so on. In addition, after each letter press 7th button to recognize pressing button for this letter finished and now we want to type another letter.

For this aim we need to introduce some components to make a circuit. Bellow we mention these components:

- **Arduino board**

There are many varieties of Arduino boards that can be used for different purposes. Most Arduinos have the majority of these components in common:



- Power (USB / Barrel Jack): Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).
- GND (3): Short for 'Ground'.
- 5V (4) & 3.3V (5): As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power.
- Analog (6): These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- Digital (7): These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- PWM (8): You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). These pins as being able to simulate analog output (like fading an LED in and out).
- AREF (9): Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

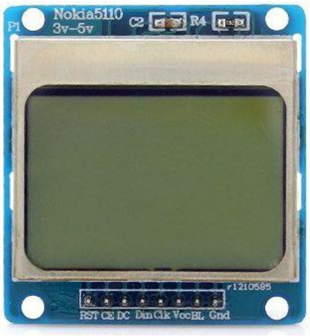
- Reset Button (10)**:** Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times.
- Power LED Indicator 'ON' (11): This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!
- TX RX LEDs: TX is short for transmit, RX is short for receive. TX and RX indicator LEDs (12)**:** These LEDs will give us some nice visual indications whenever Arduino is receiving or transmitting data (like when we're loading a new program onto the board).
- Main IC (13): The black thing with all the metal legs is an IC, or Integrated Circuit**.** Think of it as the brains of our Arduino.
- Voltage Regulator (14): It controls the amount of voltage that is let into the Arduino board.

- **Arduino Leonardo**

Here we use Arduino Leonardo.

- **LCD (Nokia 5110)**

It is a 84×48 pixel monochrome LCD display. This display is made of 84×48 individual pixels, so you can use it for graphics, text or bitmaps. These displays are inexpensive, easy to use, require only a few digital I/O pins and are fairly low power as well. To drive the display, you will need 5 digital output pins.

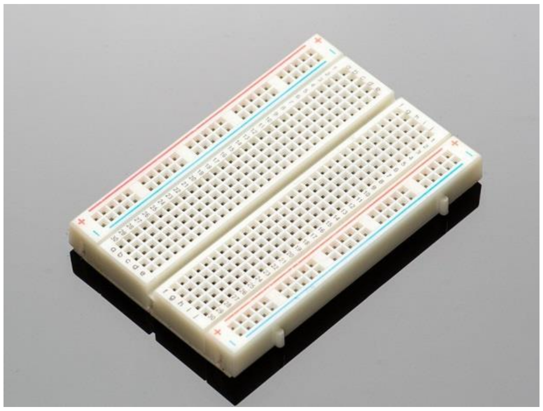


- **Push buttons**

The pushbutton is a component that connects two points in a circuit when you press it. When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to 5 volts and we read a HIGH. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to ground, so that we read a LOW. (The pin is still connected to 5 volts, but the resistor in-between them means that the pin is "closer" to ground.) [1]

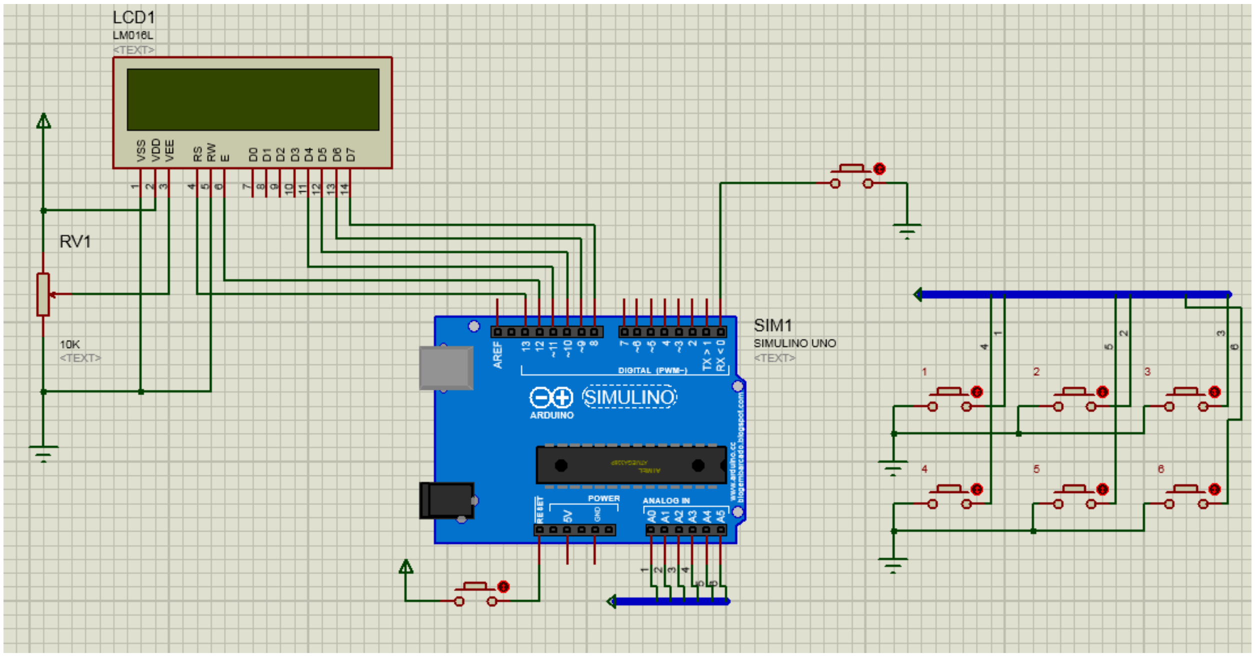
- **Breadboard**

Breadboards are one of the most important pieces to build circuits. Common use of breadboards is testing circuit. When you are trying to figure out how a part works.



- **Resistor**
- **Cable**

Circuit



Pseudo-Code

```
Declare the entire button constant
Setting values initially
Checking Complete Button
Check sum value
Check Number Format
lastDebounceTime = 0;
debounceDelay = 200;
void setup() {
    put your setup code here, to run once
    Initialize pinMode for the entire button
    pinMode(button0, INPUT);
void loop() {
    Read All The Button Pressed
        if (complete == 1 && isNumber == 0) {
            switch (sumvalue) {
                case 1:
                    Serial.print('A');
                break;
            default:
        Serial.print("Invalid Alphabet");
  complete = 0;
  sumvalue = 0;
 }
    if (complete == 1 && isNumber == 1) {
        switch (sumvalue) {
            case 2:
                Serial.print('0');
            break;
        complete = 0;
        sumvalue = 0;
        isNumber = 0;
    }
        lastDebounceTime = millis();
}
```

References:

[1] https://www.arduino.cc

[2] https://en.wikipedia.org