ARDUINO BASED TIMER: 16X2 LCD AND BUTTON INTERFACING

Submitted by Edwin Jose, Greeshma Manikandan, Jithin Raj

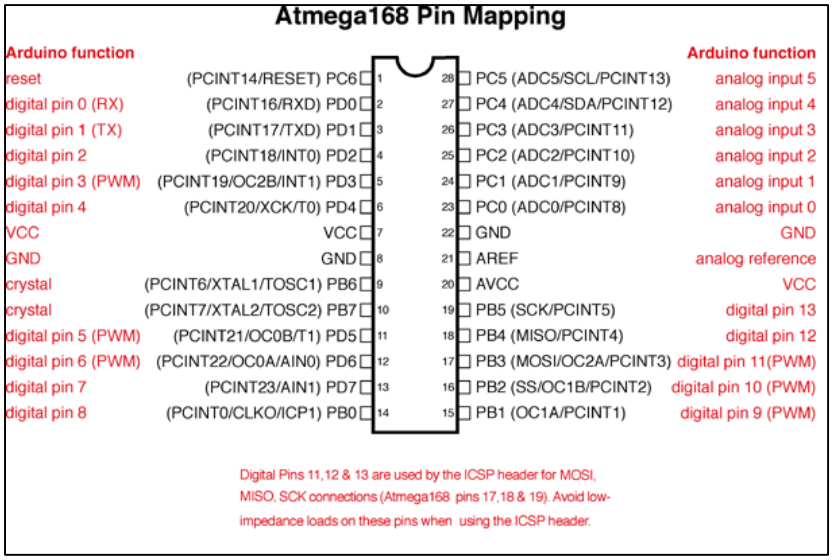
AIM

Make a Timer program using Arduino and display the results in a16x2 LCD, also add four button for :setting the timer(adding ones and tens),start the counter,and resetting the counter.

COMPONENTS REQUIRED

- Arduino uno(atmega328)
- LCD Screen - 16x2 LCM
- pin headers to solder to the LCD display pins
- 4 x Buttons
- 10k ohm potentiometer
- 220 ohm resistor
- hook-up wires
- breadboard

ARDUINO



Arduino 328 and arduino 128 have the same pin mapping. The Arduino UNO is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9 volt battery, though it accepts voltages between 7 and 20 volts.

16X2 LCD MODULE

Liquid crystal displays (LCDs) are a commonly used to display data in devices such as calculators, microwave ovens, and many other electronic devices. It has 16 pins and the first one from left to right is the Ground pin. The second pin is the VCC which we connect the 5 volts pin on the Arduino Board. Next is the Vo pin on which we can attach a potentiometer for controlling the contrast of the display.

The RS pin or register select pin is used for selecting whether we will send commands or data to the LCD. For example if the RS pin is set on low state or zero volts, then we are sending commands to the LCD like: set the cursor to a specific location, clear the display, turn off the display and so on. And when RS pin is set on High state or 5 volts we are sending data or characters to the LCD.

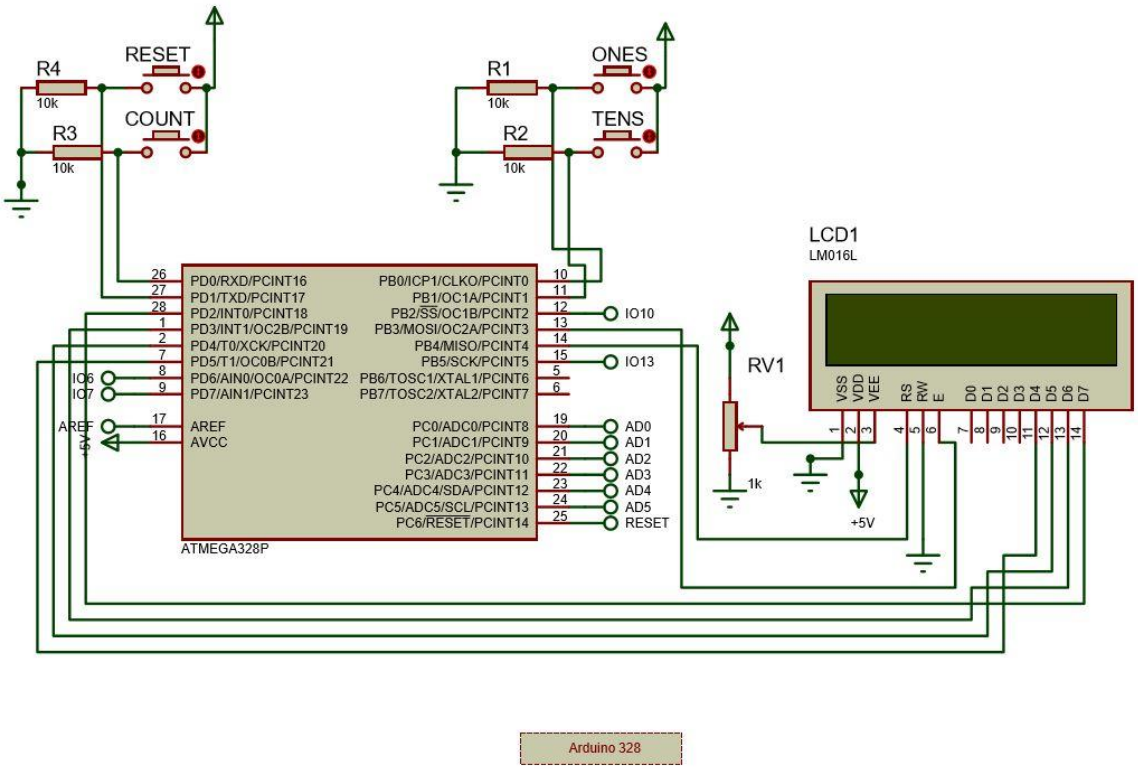
The R / W pin which selects the mode whether we will read or write to the LCD. Here the write mode is obvious and it is used for writing or sending commands and data to the LCD. The E pin which enables the writing to the registers, or the next 8 data pins from D0 to D7

The ARDUINO IDE allows the user to use LCD in 4 bit mode. This type of communication enables the user to decrease the pin usage on ARDUINO, unlike other the ARDUINO need not to be programmed separately for using it in 4 it mode because by default the ARDUINO is set up to communicate in 4 bit mode. In the circuit you can see we have used 4bit communication (D4-D7).

Terminal 1	GND
Terminal 2	+5V
Terminal 3	Mid terminal of potentiometer (for brightness control)
Terminal 4	Register Select (RS)
Terminal 5	Read/Write (RW)
Terminal 6	Enable (EN)
Terminal 7	DB0
Terminal 8	DB1
Terminal 9	DB2
Terminal 10	DB3
Terminal 11	DB4
Terminal 12	DB5
Terminal 13	DB6
Terminal 14	DB7
Terminal 15	+4.2-5V
Terminal 16	GND

CIRCUIT DIAGRAM

The following circuit diagram is made in proteus and simulated succecfully. For more efficiency a not fate from the reset button could be connected to the reset pin of the microcontroller for better efficiency.



PROGRAM DESCRIPTION:

The program for the arduino based counter is added in the APPENDIX A.

The following are the details of the libraries used in the program:

MILLIS()

Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

- Syntax: time = millis()
- Parameters: Nothing
- Returns: Number of milliseconds since the program started (unsigned long)

LIQUIDCRYSTAL LIBRARY

This library allows an Arduino board to control Liquid Crystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

- Autoscroll: Shift text right and left.
- Blink: Control of the block-style cursor.
- Cursor: Control of the underscore-style cursor.
- Display: Quickly blank the display without losing what's on it.
- Hello World: Displays "hello world!" and the seconds since reset.
- Scroll: Scroll text left and right.
- Serial Display: Accepts serial input, displays it.
- Set Cursor: Set the cursor position.
- Text Direction: Control which way text flows from the cursor.

PUSH BUTTON: SOLVING DEBOUNCE ISSUE

Pushbuttons often generate spurious open/close transitions when pressed, due to mechanical and physical issues: these transitions may be read as multiple presses in a very short time fooling the program. This example demonstrates how to debounce an input, which means checking twice in a short period of time to make sure the pushbutton is definitely pressed. Without debouncing, pressing the button once may cause unpredictable results.Hence a single physical act of pressing a button could be read as multiple high states hence we should make sure that the buttun is only read once for a single physical act of pressing the button

Solution :the button is only read once for one single physical press,for this the present state and previous state is read and as soon as the function is complete or when the super loop starts its next iterations the present state is loaded into the last state variable, hence only when the present state is not equal to the next state we consider if the button is pressed or not.

WORKING

Function of each button:

- ONES button: On pressing increase the timer count by one(adds 1)
- TENS button: On pressing increase the timer by ten(adds 10)
- COUNT button: On pressing the counter starts counting backwards with a delay of one second between each count
- REST button: On pressing the counter gets interrupted and resets to zero, can also be set to reset the overall count of the counter.

CONCLUSION

The timer circuit was successfully implemented on proteus and tested with various use case scenarios. Also were successful in learning the interfacing of the 16x2 LCD module and buttons, through this project.

Scan the QR code to watch the demo .

APPENDIX:PROGRAM

```
#include<LiquidCrystal.h>


LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // sets the interfacing pins
int num = 0;
int bs1 = 0;
int bs2 = 0;
int bs3 = 0;
int bs4 = 0;
int lbs1 = 0;
int lbs2 = 0;
int lbs3 = 0;
int lbs4 = 0;
void setup()
{
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  lcd.begin(16, 2);  // initializes the 16x2 LCD
}


void loop()
{
  //delay(20);
  lcd.setCursor(0, 0);         //sets the cursor at row 0 column 0
  lcd.print("COUNTER Program"); // prints 16x2 LCD MODULE
  lcd.setCursor(2, 1); //sets the cursor at row 1 column 2
  bs1 = digitalRead(8);
  bs2 = digitalRead(9);
```

```
bs3 = digitalRead(0);
bs4 = digitalRead(1);
if (bs1 != lbs1 && bs1 == HIGH)
{
  num = num + 1;
  lcd.setCursor(2, 1);
  lcd.print(num);   // prints HELLO WORLD
}
if (bs2 != lbs2 && bs2 == HIGH)
{
  num = num + 10;
  lcd.setCursor(2, 1);
  lcd.print(num);     // prints HELLO WORLD
}
if (bs3 != lbs3 && bs3 == HIGH)
{
  int numc = num;
  num=0;
  int count =millis();
  lcd.setCursor(0, 0);
  lcd.print("               ");
  lcd.setCursor(0, 0);
  lcd.print("COUNTER RUNNING>>>>>");
  lcd.setCursor(2, 1);
  lcd.print(numc);
  while (numc >0)
  {
    if((millis()-count)>999){
 lcd.setCursor(2, 1);
```

```
 lcd.print("      ");
 count = millis();
 numc = numc - 1;
 lcd.setCursor(2, 1);
 lcd.print(numc);
 //delay(1000);
   }
   lbs4 = bs4;
   bs4=digitalRead(1);
   if(bs4==HIGH && lbs4!=bs4)
 break;
 }
 lcd.setCursor(2, 1);
 lcd.print("      ");
 numc=0;
 num=0;
 lcd.setCursor(0, 0);
 lcd.print("              ");
 lcd.setCursor(0, 0);
 lcd.print("RESET");
 lcd.setCursor(2, 1);
 lcd.print(num);
 delay(1000);
}
if(bs4==HIGH && lbs4!=bs4){
 lcd.setCursor(2, 1);
 lcd.print("      ");
 //numc=0;
 num=0;
```

```
    lcd.setCursor(0, 0);
    lcd.print("               ");
    lcd.setCursor(0, 0);
    lcd.print("RESET");
    lcd.setCursor(2, 1);
    lcd.print(num);
    delay(1000);
   }
  lbs1 = bs1;
  lbs2 = bs2;
  lbs3 = bs3;
  lbs4 = bs4;

}
//8,9 pin
```