Arduino Lab

Robert Barron
Jorel Lalicki

For this lab we will be walking you through installing Arduino and running your first program. Then you will be making a Digital LCD Clock that shows the current time on an LCD screen.

Part 1:
    Digital Control - Blinking an LED
Part 2:
    LCD Display - Hello World
Part 3:
    Serial Communication - Print characters on screen as received over serial
Part 4:
    Serial Communication Again - Serial APIs - Designing and implementing commands

        -Set Timer
        -Start Timer
        -Stop Timer
Part 5:
    Creating the digital clock

Part 6: More Peripherals!

Appendix: LCD sample code

Arduino API Reference - You'll want to take a look at the Arduino API for some parts of this lab, particularly string manipulation which is a little different than C++.

Example LCD Screen code - You'll need this for part 2..

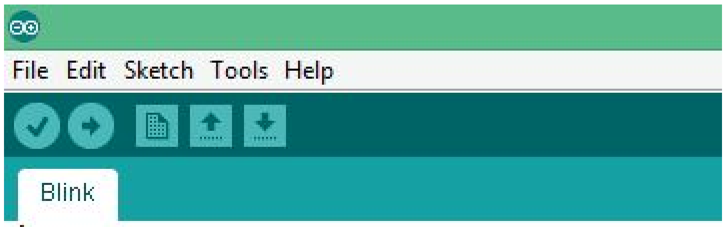
Part 1: Blinky

To start, we’ll need to download the Arduino IDE and confirm our Arduino board is working.

Once you have downloaded and installed Arduino, trying plugging your board in. Depending on your OS, it might install drivers.

To confirm your Arduino is working, we will be programming it with the “Hello World” of embedded programs, “Blink”. This will blink an LED every second on your Arduino board.

1. First, open the Arduino program, then navigate to *file->examples->basics->blink*
2. Now we need to set our board type. Navigate to *tools->Board-> Arduino/Genuino Mega or Mega 2560*
3. Next we need to set our COM port. Go to *tools->port*. On Windows you should see the the name of the Arduino under one of the ports, so go ahead and pick that one. On some Linux installations you might need to run the Arduino software as root to get the board to show up in ports.
4. Finally you can compile the program and program your Arduino. The *check mark* is the compile button, and the right pointing arrow uploads the program to the Arduino.



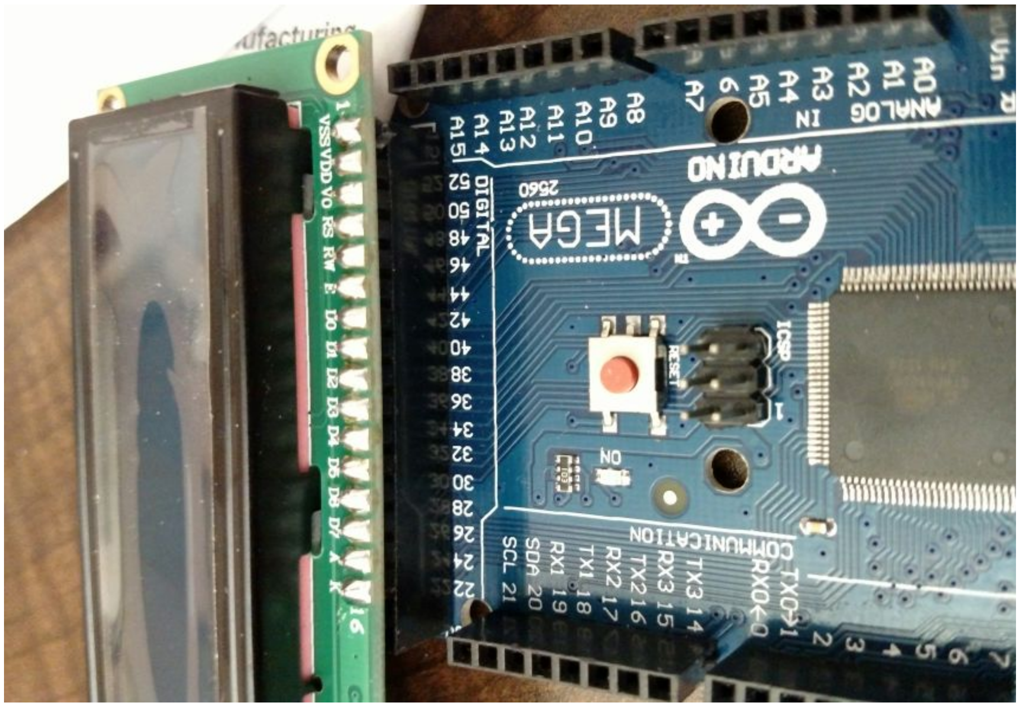
5. Your Arduino should now have an LED blinking every second. Take a look at the code to see how simple controlling Digital I/O is. Digital I/O is basically an ON or OFF signal. On this Arduino board writing a pin HIGH corresponds to a 5V signal, and LOW corresponds to a 0V signal. On some other types of boards HIGH might be 3.3V instead. The LED on this Arduino is on pin 13 so we are simply turning it on and off with our Digital Write commands. Try changing the speed the LED flashes.

}

Part 2: LCD Screen

First you'll need to download our LCD example code. We've created a special Arduino library that allows you to plug the LCD display directly into the Arduino with no additional wiring.

You'll need to plug the LCD screen into a specific row of pins on the Arduino as shown in the picture below. Do this before uploading our sample code to the Arduino.



Now run our sample arduino project code. You should see "Hello World" displayed on the screen. Try playing with the LCD commands until you get a feel for how printing and moving the cursor works. The relevant code is here:

```
void loop() {

        lcd.home();//return cursor to starting position

        lcd.clear();//clear the lcd screen

        lcd.print("Hello World");//displays a string on the lcd screen

        ...
 }
```

Part 3: Printing over Serial

For this part your goal is to print any string received over serial to the LCD screen. You'll want to play with a serial example project to get a feel for how serial works. To bring up the Serial monitor go to *tools->serial monitor*. This monitor lets you send and receive strings from the Arduino.

```
//add this code to your loop function to send and receive strings from the serial monitor
//edit this so you can display received strings on the lcd screen
void loop(){

        String message = Serial.readString();// read a string sent from the computer
        Serial.print("hello world");//prints hello world to the serial monitor
}
```

To complete this part you should be able to type "hello world" into the serial monitor and have it displayed on the LCD screen. You can also try implement handling of the newline "\n" character so that you can use both lines of the lcd screen automatically.

Part 4: Serial APIs - Implementing functions

Now we'll want to make a command structure over serial. You'll be able to send commands over the serial monitor, which will be used to tell the Arduino to do something, instead of just printing to the LCD. Specifically, we want to create a command that tells the Arduino a time to start counting from. This will be used later to put our clock together.

An example command structure (you can do this however you want) might be "t1234" where the "t" is a "set time command" and the numbers after is the time to start counting from. You'll need to write serial parsing code so that when the Arduino receives the "t" command it should then expect the numbers after to be the time.

For this part you should make a "Set Timer" command that will set the current time as detailed above.

If you send "t1230" over the serial monitor, the Arduino should display "12:30" on the lcd screen. In part 5 you'll make use of this command and two others to make a working digital clock.

Hint: Use Arduino's built in String API to make this easier.

Part 5: Putting it all together

Now we get to put this all together to make a digital lcd clock.

When the Arduino receives a Start Timer command it should begin displaying the time. The Arduino itself should count the time and update the display each second. You can count time using the “millis()” function. Refer to the Arduino reference for more information.

You’ll need to make two more commands in addition to the Set Timer command you made in part 4.

Start Timer - Begin counting from the set time
Stop Timer - Stop counting from the current time

So you’ll want your final result to go like this:

-Send a command over serial to tell the Arduino what the current time is
-Send a command over serial to tell the Arduino to start displaying the time

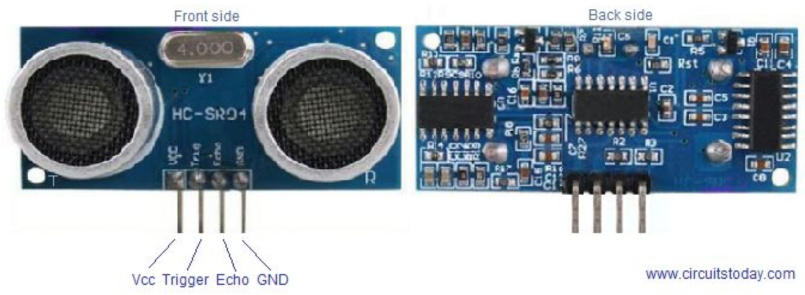
The LCD should display a result (using both lines of the LCD):
Current time:
12:30::12

- The Arduino should count time on its own once you send the Start Timer command. You shouldn’t need to send another command to update the time.
- It should stop counting the time (just display the time when it stopped) when you send a command over serial telling it to stop.
- If you never entered a start time, have the display blink “1200” when you start the timer

Part 6: More Peripherals

Once you've finished the rest of the lab we have a few different peripherals for you to experiment with:

1. **Ultrasonic Ranger** - This sensor lets you measure how far it is from an object. Our particular sensor is the **HC-SR04**. Ultrasonic sensors send out a ping and then count how long it takes for the ping to reflect off an object. Because we know how fast the wave travels we can then use that time to calculate how far away that object is.



For this part you'll want to implement a power saving "wakeup gesture" feature. The Arduino should only display the current time for a few seconds on the screen when you wave your hand closer than 10cm to the ultrasonic ranger. After you move you hand away it should go back to "sleep" and stop displaying the time

The Arduino IDE has a built-in example project for taking readings from this sensor.

2. **Joystick Shield** - This is what's called an Arduino shield. These can be plugged directly into the top of the Arduino without any additional wiring. Joysticks work by moving an adjustable resistor called a potentiometer. We will measure the voltage across the resistor by using the Arduino's Analog pins. An analog pin lets us measure a voltage between 0-5V, which it then converts into a digital number between 0-1023 (10 bit resolution). This lets us know how far the joystick has been pushed rather than just getting an on/off (joystick versus d-pad on a game controller).

3. **Servo Motors** - These are motors that can spin from 0-180 degrees, and hold position. These are great for manipulating real world objects (like opening a box, or creating something like our face tracker project). These motors are controlled by a 50hz PWM signal, but using the built-in Arduino library all you have to do is set their angle.

4. **RGB LED Shield -** This is another Arduino shield. This shield has an array of 40 RGB LED's. You can control each one individually and set them to any color. It's easy to display info or create animations with them. Tutorial and libraries are on Adafruit.

5. **Piezo Buzzer -** These buzzers can be controlled with PWM to emit an annoying tone.

Appendix: LCD sample code

This sample code will get you started with the LCD screen. Copy paste this into your Arduino IDE.

```
#include <LiquidCrystal.h>

/******************************************************************************************************************
******/
/***********     LCD Initialization Code - Don't change unless you want stuff to stop working!
********************/
/******************************************************************************************************************
******/
/*
 *     void init_LCD();     Call this function to set up the LCD screen for use
 *     void LCD_off();      Call this function to turn off the backlight of the screen. Note that this
leaves the LCD still on, but not visible
 *     void LCD_on();       Call this function to turn the backlight back on. The LCD starts on by
default
 *
 *
 *
 *
 *
 *
 *
 */






// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28);
int __counter = 0; //Global counter variable for contrast PWM
void init_LCD(){__init_LCD();}

void __init_LCD(){
  pinMode(24, OUTPUT); //K
  pinMode(26, OUTPUT); //A
  pinMode(54, OUTPUT); //VSS
  pinMode(52, OUTPUT); //VDD
  pinMode(50, OUTPUT); //Contrasty pin
  digitalWrite(24, LOW); //Backlight
  digitalWrite(26, HIGH); //Backlight
  digitalWrite(54, LOW); //GND
  digitalWrite(52, HIGH); //VDD
 // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Timer0 is used for millis() - we'll just interrupt
  // in the middle and call the compA
  OCR0A = 0x01;
  TIMSK0 |= _BV(OCIE0A);
}
SIGNAL(TIMER0_COMPA_vect)
{
   __counter++;
   if (__counter > 14){
      digitalWrite(50,HIGH);
      __counter = 0;
   }
   else if (__counter > 3){
      digitalWrite(50, LOW);
   }
}


//turn lcd backlight off
void lcd_off(){
  digitalWrite(26, LOW); //Backlight
}

//turn lcd backlight on
void lcd_on(){
  digitalWrite(26, HIGH); //Backlight
}
/*****************************************************************************************************************
*******/
/*****************************************************************************************************************
*******/
/*****************************************************************************************************************
*******/
```

```
void setup() {
  // initialize the serial communications:
  Serial.begin(9600);
  init_LCD();

}

void loop() {

  lcd.home();
  lcd.print("Hello World");
  lcd.setCursor(0, 1);

}
```