Name: _____ Date: _____

Serial LCD Reaction Timer

Introduction

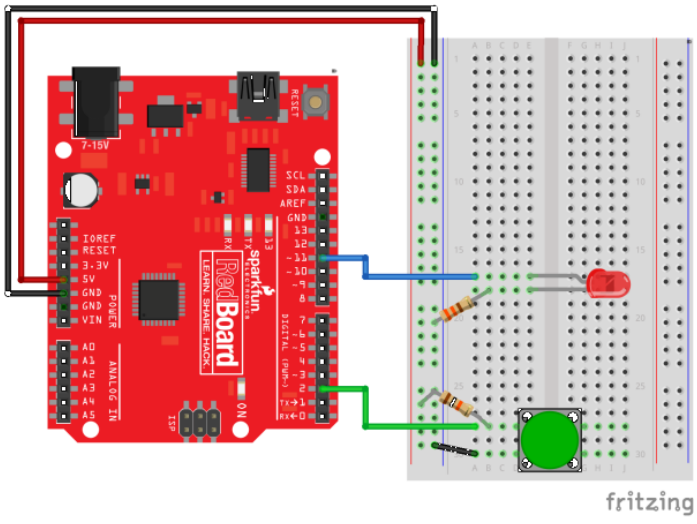
Mental chronometry is the study of how fast humans react to different inputs. It takes a few hundred milliseconds for the signal to get from your eyes, to your brain, out to your limbs to respond. The reaction timer is a great project to demonstrate this time delay. It also makes for a fun game between friends!

According to the team at Human Benchmark[1], the average human reaction time is about 215 ms. How fast are you?

The principal of the reaction timer is simple: when the user sees the light turn on, press the button! A microcontroller is perfect for this because it can time milliseconds very accurately

For this project, we will use an Arduino to be the time-keeper. An Arduino is a small, low-cost, and fast microcontroller that is capable of performing command / instructions at a rate of 16 MHz or roughly 62.5 ns per instruction.

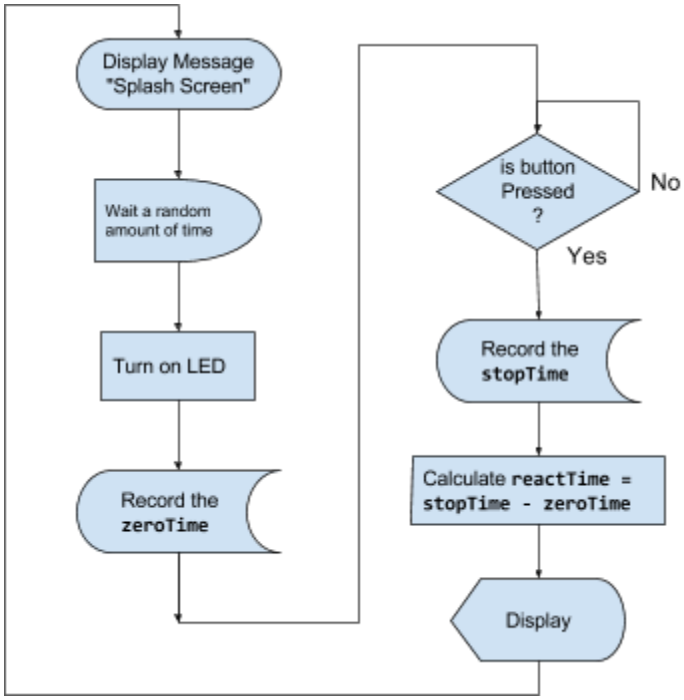
Wiring / Hook-up Guide



- Connect the 5V and GND rails on the breadboard to 5V and GND on the Arduino.
- Connect the Indicator LED between pin 11 and the GND rail using a 330 ohm current limiting resistor. Make sure you use the right size resistor, or the LED will be dim.
- Now, wire up the push button. Insert the push button on the very last row of the breadboard. You may need to bend the legs slightly so that the button sits flat.
- Connect one side of the button to 5V through a 10k pull-up resistor. Connect this same pin also to pin 2 on the Arduino.
- Finally, connect the other side of the button to the GND rail. When you push the button, you will connect pin 2 to GND.

[1] http://www.humanbenchmark.com/tests/reactiontime/index.php

Program Flow and Control



Here is a simple flow chart illustrating the programming for the reaction timer.

Open up a web browser and go to codebender.cc or use the regular Arduino IDE to write this code. We've pre-written some code to get you started at:

https://codebender.cc/sketch:349488

After you have the code typed out, select the **SparkFun RedBoard** as the board type and the appropriate port. Click the **Run on Arduino** button (CTRL+U) in either codebender or the Arduino IDE, and then open up the Serial Monitor.

Explore! How fast are you? What is your fastest reaction time? What is your average reaction time?

How it works

The reaction timer program uses what we call a blocking loop to wait for the button press. You can see this on line 50-52. It uses a command called the `while()` loop. What this does is ***while*** the `buttonPin` reads HIGH, it's going to stay in this loop. Notice that the loop doesn't actually have any code. It's like a holding loop. It won't exit this loop and move to line 53 until the `buttonPin` reads LOW.

```
50  while(digitalRead(buttonPin) == HIGH)  // blocking loop until button is pressed.
51  {
52  }
```

This is a neat trick if you want the program to stop and wait for something like a button press. It is what we call a ***blocking*** function. Note that this means that nothing else happens until the button pressed. This includes blinking LEDs, reading other sensors, or anything else you might be doing in the main `loop()`.

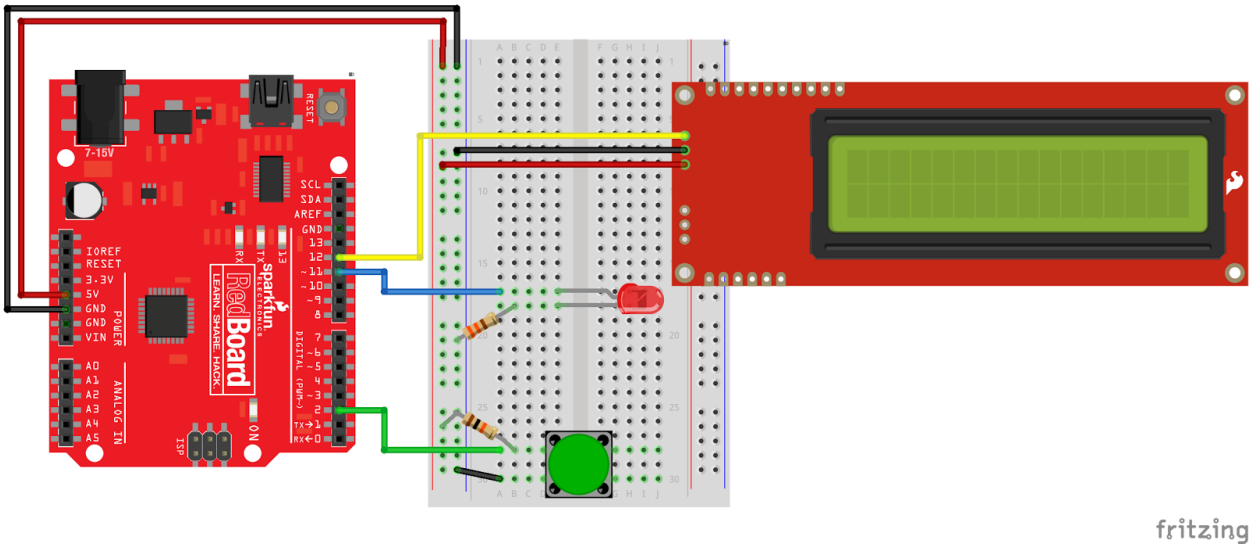
Making it better

You may notice that the reaction timer game starts immediately and doesn't wait for a button press to start. This means that if you're not watching it, it's going to automatically start the game with or without you. Change line 39 from `Serial.println("Get Ready!");` to `Serial.println("Push to start.");`

Now, see if you can figure out how to re-use the same code example from lines 50-52 to check for a button press at the beginning of the loop before starting the game. This will make the game flow a lot better.

Adding the Serial LCD

The Serial LCD is a nifty display device that uses just three wires to connect to your Arduino. The wires are color coded Red, Black, and Yellow for Power, GND, and Signal. Connect the Signal wire to pin 12, the Power to the 5V rail, and GND to the GND rail.



Modifying the Code

In order to send data to the LCD through pin 12, we need to add a few lines of code. First, we need to include a library that will allow us to add a second serial port on the Arduino. Add these two lines of code near the top starting on line 7.

```
7  #include <SoftwareSerial.h>
8  SoftwareSerial LCD(13, 12);
```

This sets up pins 13 and 12 to used as receive and transmit serial communication lines. The pin 12 (transmit) will transmit data to the LCD.

Similar to the normal Serial port on the Arduino, you need to setup this one with the speed. Add this line to your setup() function on line 27. This will initialize a Serial port to the LCD at 9600 bits per second.

```
27  LCD.begin(9600);
```

Now, each time you want to send or print data to the LCD, you simply use the code:

```
LCD.print("My message here");
```

Try this out. Just after you print out the display to the Serial monitor, add to line 40, the code:

```
40  LCD.print("Reaction Timer");
```

Upload this and see what happens! You should now see the text on your LCD.

SerLCD Special Commands / Controls

Any text that you print using the `LCD.print("");` command will show up on the display. Here are a few special commands you can use to clear the display and move the cursor:

```
// clear the LCD Display
LCD.write(0xFE);
LCD.write(0x01);

// move cursor to the 2nd line
LCD.write(0xFE);
LCD.write(0xC0);
```

A complete application note / reference guide to the Serial LCD command set is available at: http://sfe.io/r113

Use these commands and the `LCD.print();` to add a visual user feedback to your Reaction Timer. Once you've finished, you should have a stand alone Reaction Timer game that you can put into an enclosure and test out with your friends.

Additional Explorations / Examples / Investigation Ideas

You now have a reaction timer. Great, now what? Here are a few ideas of things to explore:

- **Statistics:**
  - What is your fastest reaction time? What is your slowest reaction time?
  - What is the best measure of your actual reaction time? (Measures of central tendancy -- mean, median, mode)
  - How do you identify an “outlier”? What does standard deviation mean?
- **Biology / Social / Other Science:**
  - **Age**. Does reaction time vary with age?
  - **Handedness**. Does reaction time vary with dominant vs. non-dominant hand?
  - **Color**. Does the color of the light make a difference?
  - **Gender**. Do males vs. females have a difference in reaction time?
  - **Internal Clock**. How good is yours? How well can you estimate 5 seconds? How about 30 seconds?

Going Further

See if you can add sound to your reaction timer. You’ll need a buzzer and a few extra lines of code. The buzzer has just two connections on it. Connect one side of the buzzer to pin 9 and the other side to GND.

In the `setup()`, add a line that reads:

```
pinMode(9, OUTPUT);
```

And, to make a tone, use the command:

```
tone(9, 440);  // 9 indicates the OUTPUT pin, and 440 is the frequency
```

To stop the tone, use this command:

```
noTone(9);
```

Want to see a complete working code example of this project?
https://codebender.cc/sketch:349329

Troubleshooting Tips

- Check code for syntax errors.
- Make sure that the Arduino is plugged into your computer.
- Make sure that the status shows "Upload Successful"
- Double check the wiring to the drawings and illustrations.
- Double check to make sure that the power light is on on the Arduino. If the power light is not on, it's likely you have a short circuit or the device is not plugged in. Double check your 5V and GND wires to make sure that there's not a short.
- Check the Serial port and the Board settings in Arduino. For PCs, the Serial port should be a **COM#**, and on Macs, the Serial port will be something like: **/dev/cu.usbserial-Axxxx**