Exported for Brian O'Connell on Tue, 27 May 2025 19:29:39 GMT

Programming 3 Homework - Digital Signals

For programming homework, we will begin the activities in class and use them to go over and reinforce the pre-lab content.

For this homework assignment, you will need your Sparkfun Inventor's Kit or other approved kit.

WARNING:

When making adjustments to your circuit, it's best practice to disconnect it from power and double check your wiring before re-connecting it. The RedBoard is fairly robust but this will minimize the risk of causing it any damage.

In anticipation of my son continuing to be sick, I've pre-recorded some of this in case I had to be remote or asynchronous.

Pre-lab review

Video

Please visit the textbook on a web or mobile device to view video content.

Review - Buttons

Video

Please visit the textbook on a web or mobile device to view video content.

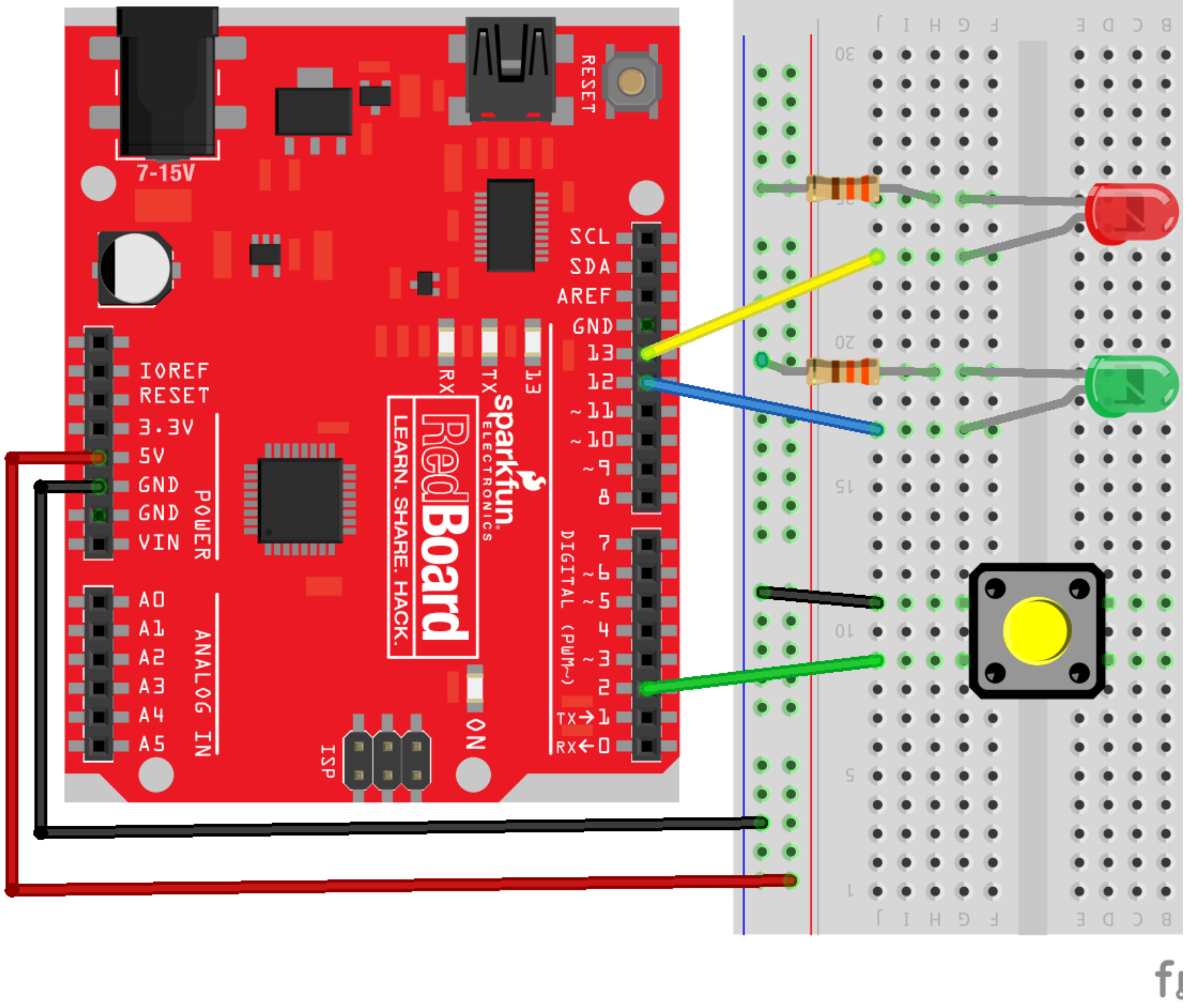
P3H1: Blink-Button

For this part, we will build an example that demonstrates how to use state variables to implement a button switch and debounce and how to use the command millis() to control delays in your code, allowing for multiple timed activities that aren't inhibited by the delay command.

Example:

Start with the following circuit and the example code found here:

https://raw.githubusercontent.com/boconn7782/CourseCode/refs/heads/main/P3H1_Start.ino



This was covered purely in class. Please see the presentation for details.

INPUT_PULLUP

Video

Please visit the textbook on a web or mobile device to view video content.

Millis Timer

Video

Please visit the textbook on a web or mobile device to view video content.

NOTE FOR THE ABOVE VIDEO: In the slides, the code is just a subset of the given example, the part we're working on. Also, I included `if led1state == HIGH  { ...` in the slide. You need the the conditional to be in `(   )` in your code. I've fixed that in the slides, but haven't re-recorded the video.

NOTE: Clarifications were added to some slides in class due to questions from students, to clarify that you are not doing anything to the button example portion of the code.

For reference to create multiple timers, this tutorial goes over Millis Timers but includes how to handle multiple: https://learn.adafruit.com/multi-tasking-the-arduino-part-1

Requirements:

Need to have implemented:

- INPUT_PULLUP
- State Variables
- Button Switch
- Debounce

- Millis Timer

Intended behavior

Independently of one another:

- LED 1 will blink every 1 second
- LED 2 will switch on or off whenever the button is pressed
  - This will occur with the press and release, not needing to be held down.

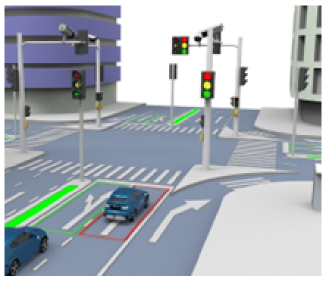
Complete this part of the assignment.

- Take a video of your working circuit
  - Try to describe the millis() timer, button switch, and debounce concepts
- Print a pdf of your code
  - Make sure you've included a title block and comments explaining your updates.

End of Part 1

P3H2: Pedestrian Crossing Traffic Light

This part is to build upon our prior Arduino experience and extrapolate it to meet a new goal. We'll will use the Blink LED's, RGB LED , and the ***button*** examples to create a board that implements an intelligent traffic light system and pedestrian light that uses a button to grant road access to pedestrians. This will aid in practicing the core C++ concepts and application of the Arduino components.



Recommendation: Follow along, referring back to the presentation and TinkerCAD example directly to see the effects of interference and the importance of using a pull-down resistor while using a button. All the attributes discussed below MUST function in one code file. Copy and Paste from class Sparkfun code is acceptable for this assignment.

Part 1: Algorithm & TinkerCAD

For this assignment, start off by creating an algorithm that implements the functionality discussed in Part 2 and Part 3. The pseudocode or flowchart must be typed or drawn using a computer program. Next, methodically, figuring out the connections and logic of your program. Use TinkerCAD it that's a userful development tool for you. In some instances, the simulations capabilities may be limited such that it may be impossible to recreate the real world system. However, for this assignment, TinkerCAD should be able to hand the scenarios provided. Once you are done with your TinkerCAD simulation implementation, you can make similar connections with your kit components as a one-one direct translation. To complete this part, see part 2 and 3 below for more details on the assignment requirements.

Part 2: The lights

Build the circuit using 3 LEDs (Red, Yellow, and Green) to create your Traffic Light (TL). Once you are done with the connections, develop your code to perform the following tasks:

1. Start with the programming of a traffic light (TL):
   1. Green LED (lights/ turn on) for 5 seconds and then off.
   2. Immediately, Yellow LED (lights/ turn on) for 2.5 seconds and then off
   3. Immediately, Red LED (lights/ turn on) for 7.5 seconds and then off.
2. Get this working using millis() timers and State Variables
3. Record (video) your circuit working before moving on
   1. Recording is a just in case measure, to show you at least got this far. Not required if you're confident you'll get the whole thing done but good to have contingencies in place.
4. Connect the RGB LED as the Pedestrian Light(PL).
   1. Don't remove the LED's connections for step 1!
5. Now, modify your code such that, as soon as the TL turns red, the PL cycle begins as follows:
   1. PL turns on with a white color for 3 seconds to allow pedestrians to cross the street.
   2. After, PL turns purple[1] and blinks (for 2 seconds total with 0.25 seconds delay between each blink.
   3. Immediately after, PL turns solid purple and stays next cycle when red light turns on.
6. Again, get this to work and record(video) the circuit working before you move on.

[1]The standard pedestrian signals specified by the US Department of Transportation Federal Highway Administration are a White walking person and a "Portland Orange" Stop Hand (not red). These colors were chosen so as to be conspicuous against a backdrop of red, green and yellow lights at intersections and to avoid giving drivers false cues of a green light (a good example of a

resilient design). We're using Purple because Orange requires some knowledge of analog signals and that's next week.

Part 3: The button.

1. Add/connect a button to your circuit to allow pedestrians access to the street.
2. Now, modify your code so the button makes the program to behave as follows.
    1. The Push button will have no effect if the TL is yellow or red.
    2. If the button is pressed when the TL is on green Light, the TL should be immediately interrupted and turn yellow (for 2.5 sec), then red (7.5 secs) following part 2 sequence.
    3. As soon as the car TL turns red, the pedestrian light turns white (3 sec cross time), then turns purple and blinks (for 2 sec, 0.25 delay), then turns solid purple and stays purple until the next red light, at which point the PL's cycle resets as in part 2.
3. You are done, so, record (video) the circuit working with narration.

I have not provided a detailed pseudocode or flowchart nor a wiring diagram since, you should start off the homework in small groups to work together to develop your own and may even begin the circuit in TinkerCAD to generate your wiring diagram with your initial code. This will help with creating the circuit and the logic too. I left that for you to do a little research on the component yourself and determine/ decide how to wire it up. Remember you are not starting from 0 as most of the concepts we already did in class. For the RGB, see the vender documentation, which is a good move for any component you're not familiar with. Don't forget ballast resistors.

Debugging Tip:

- Use constant variables to store pin numbers, push button state, etc.
- Your name as well as any collaborators should be in the title block/header.
- Your code should have enough comments to help us understand what each line of your code does, and how the hardware is connected.
- Change the connection of the LEDs and Pins to match the code and circuit.
- Use the flat jump wires when possible, to create a cleaner design

The challenging parts:

There are some logical elements to consider though. For instance, the timing has to be impeccable so as not have gaps in the transitions. Additionally, keep in mind that the button only interrupts that sequence when it is on Green Light, otherwise, nothing happens when it is on any of the other options. There is options to apply `logical operators` that are discussed here(https://www.arduino.cc/reference/en/). You ***may*** find that you have to reorder things so that it actually work. Remember that the first true statement is the one that occurs first. Multiple statements could be true, but it's the first one that matters. Do not forget to include comments in your code describing what's supposed to occur. Someone not familiar with Arduino should be able to read your code and know what's happening based on the comments.

Your Algorithm

You must also complete your algorithm (pseudocode/flowchart) to reflect your implementation, specifically capture the logical as discuss in class and how your implemented the button interruption.

If you're having issues, use the provided class examples code. Don't ignore the algorithm where each line can help understand what comes next. The starter code is meant to be for the beginning of class so reference some of the slides to see where some of the logical issues are pointed out so you don't forget to address them.

Extra Credit

Achieve all of the above goals without using any delay commands, only Millis Timers.

Submission Requirements:

P3H1

- Raw code
    - Includes Title Block
    - Includes comments
    - Based on the provided pseudocode
    - Relies on Programming Lab 3 content
- PDF
    - An photo of your circuit

- A copy of your raw code - Ensure formatting is preserved
- Video
  - Shows face or Student ID
  - Shows upload code
  - Describes the running circuit

P3H2

- Raw code
  - Includes Title Block
  - Includes comments
  - Based on the provided pseudocode
  - Relies on Programming Lab 3 content
- PDF
  - An photo of your circuit
  - Completed algorithm (Pseudocode/flowchart)
  - A copy of your raw code - Ensure formatting is preserved
- Video
  - Shows face or Student ID
  - Shows upload code
  - Describes the running circuit
- PNG
  - TinkerCAD circuit Diagram.

Rubric

P3H1: Blink - Button

25% Total

- Circuit - 10%
  - Circuit Correct
- Code - 15%
  - Runs without errors and meets Assignment requirements
  - Appropriate Commands
  - Commented throughout and cleanly
- Video - 10%
  - Identification, video upload and format
  - Introduction describes the circuit and functionality

    - Blink functions independently
    - Button press functions without delay
    - Circuit runs as described

P3H2: Intelligent Traffic Lights

75% Total

- Pseudocode - 15%
    - Logic built off the provided steps
    - Reflects the actual functioning of your circuit
    - Adheres to course pseudocode syntax
- Circuit - 10%
    - Circuit Correct
- Code - 30%
    - Runs without errors and meets Assignment requirements
    - Appropriate Commands
    - Commented throughout and cleanly
- Video - 20%
    - Identification, video upload and format
    - Introduction describes the circuit and functionality
    - Traffic light timing
    - Clearly shows button push functionality
    - Circuit runs as described
- EC: No Delays, Only Millis Timers - 5%

The following survey is for my reference to help improve future assignments. The results are not checked until after the semester is completed and participation is not required.