RoboJackets Firmware Training Week 2 Lab Guide

Andrew Rocco, Andrew Roach

September 1, 2024
v1.1

Contents

1 Background

1.1 Topics

The important topics being discussed this week in lab include state machines, interrupts, and more complex C++.

1.2 Premise

The lab premise is to make a state machine that implements a simple counter. At the start of the program, the counter should start at 0 and should be able to count up to 5 and back down to 0. This state machine will change states based on the 2 button inputs and will display state using the 5 controllable LEDs. One button input will be able to increment and the other decrement.

1.3 Interrupt Service Routines

Interrupt Service Routines (ISR) are functions that are called when interrupts are activated. These are usually short functions that should be used to update variables in global scope. Remember that interrupts are when your microcontroller gets a signal that makes it stop whats its doing, run the ISR, and then return to the code it was running. Interrupt-based programming is very common for microcontrollers and robots and is usually more efficient that other methods.

1.4 Simulation

If you are using a simulation instead of the hardware, do not worry. The steps are exactly the same. Go to the TinkerCad link and you will see the circuit that is a subset of the the hardware. The Arduino you see will be what you use, with the LEDs and buttons replicated as they would be on the actual board.

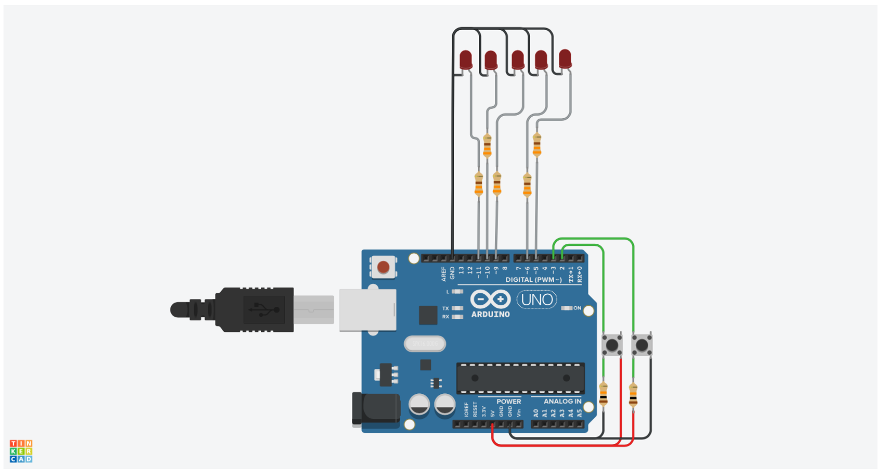


Figure 1: The circuit window of TinkerCAD for this project

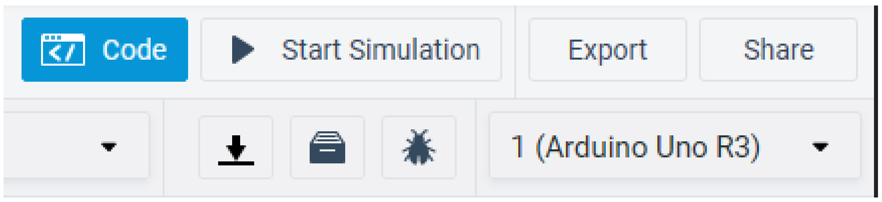


Figure 2: The area which you can use to select your target and compile

1.5 Firmware Training Board

If you are using the hardware, there are a few steps to complete to ensure that you can program the Arduino Nano on the circuit board. You may need to download the CH340 driver for your operating system which can be found here. If you completed this in Week 1 there is no need to do this again. Remember in Tools>Board Type, ensure that you have set the board to Arduino Nano. Also make sure that you have selected the correct COM port in Tools>Port. For these Arduino Nanos you also need to select Tools>Processor>ATmega328P (Old Bootloader).

2 Materials

- If running Simulation: AutoDesk Education Account and TinkerCAD
- If running on Firmware Training Board: Firmware Training Board Template Code and Arduino IDE

3 Objectives

3.1 Task 1 - Create State Machine

1. Plan and draw out the full state machine.
   - Make sure you have all the states and transitions marked.
   - As a programming convention, we will number our states from 0 upwards, which you will find more useful in the next step.
   - Don't forget that no LEDs can be on for our counter.
   - A refresher on state machines and a template to get started can be found in Section 4.1.
2. Write code to display state in `loop`.
   - You will need to go through every LED to set it state, so use a `for` loop and the `pinArray` array.
   - Make sure to use the fact your states are starting counting at 0, which is similar to arrays.
   - You will need to use your state variable to know how to do the update.

3.2 Task 2 - Create Interrupts

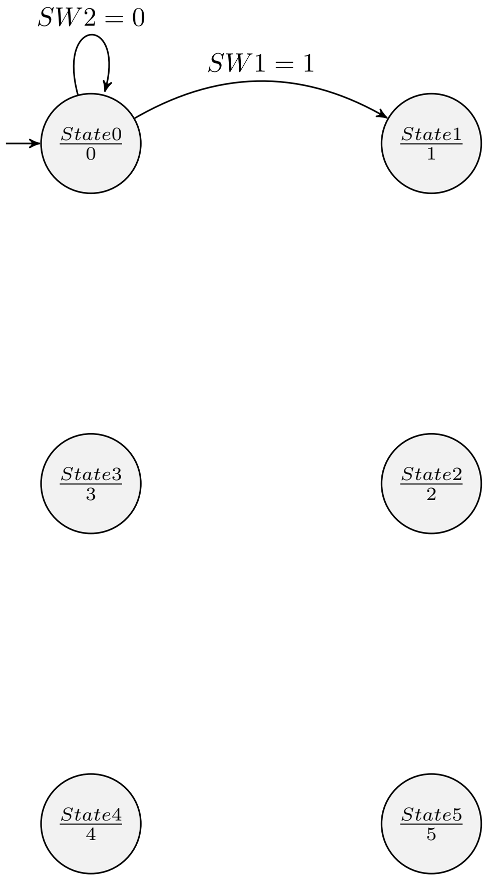
1. Write an ISR for each button.
   - SW1 is a button that when pressed sends a `HIGH` value to the Arduino, but is otherwise `LOW`. You should make sure this button should increase the number of LEDs turned on.
   - SW2 is a button that when pressed sends a `LOW` value to the Arduino, but is otherwise `HIGH`. This button should decrease the number of LEDs turned on.
2. Write code in `setup` for the buttons and interrupts.
   - You will also need to setup the buttons as normal digital inputs first.
   - Make sure to understand how each button has a different type of change, which will affect how the interrupt triggers.
   - A refresher on setting up interrupts can be found in Section 4.2.

4 Relevant Information

4.1 State Machines

State machines are tools to organize the behavior of code, based around a number of ”states” or points the code can be at. We specifically are looking at state machines that have output behavior depend only on current state, meaning the number of behaviors equals the number of states. The state machine inputs are used to trigger transitions to different states, and there no transitions to an impossible state. A graphical representation is often used with bubbles representing states and labelled arrows representing transitions. In addition, a state-transition table can be used to show what state a finite-state machine will move to based on the input and current state. Templates for both the table and graphical representation are given with the first two transitions filled out.

Current State	Input	Next State	LED Output
State 0	SW2 = 0	State 0	0
State 0	SW1 = 1	State 1	1
State 1	SW2 = 0		
State 1	SW1 = 1		
State 2	SW2 = 0		
State 2	SW1 = 1		
State 3	SW2 = 0		
State 3	SW1 = 1		
State 4	SW2 = 0		
State 4	SW1 = 1		
State 5	SW2 = 0		
State 5	SW1 = 1		



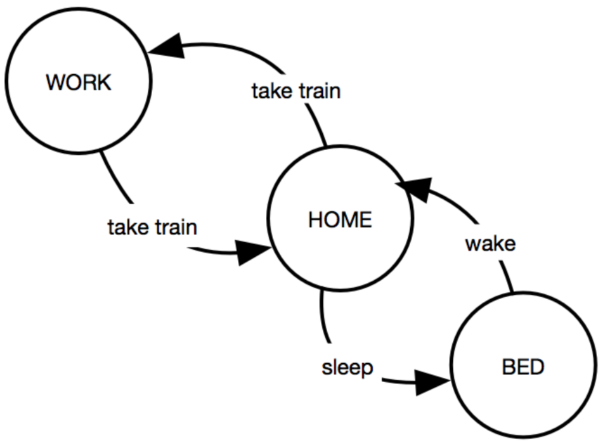
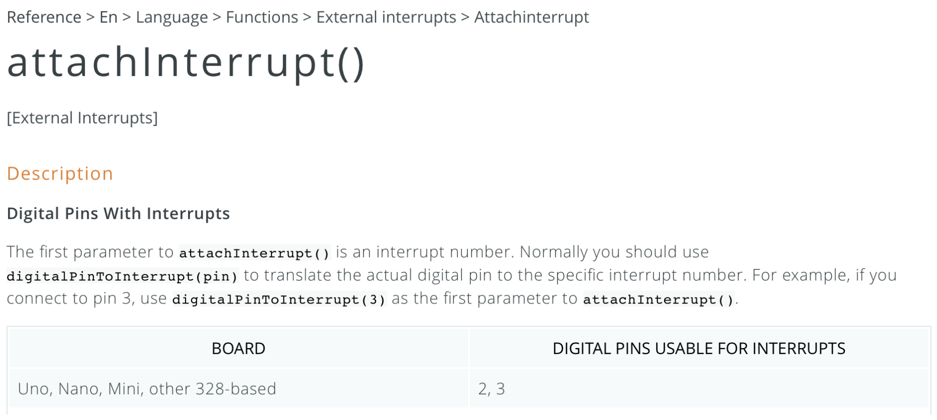


Figure 3: Example State Machine

4.2 `attachInterrupt` Function

The function we will be using for setting up interrupts is the `attachInterrupt` Function. You can refer to the reference page here for understanding it along with examples. Specifically, you need to look into how to convert a pin to a interrupt number and to set the mode that triggers the interrupts. Note that you should use the recommended syntax using the `digitalPinToInterrupt` function.



Reference > En > Language > Functions > External interrupts > Attachinterrupt

attachInterrupt()

[External Interrupts]

Description

Digital Pins With Interrupts

The first parameter to `attachInterrupt()` is an interrupt number. Normally you should use `digitalPinToInterrupt(pin)` to translate the actual digital pin to the specific interrupt number. For example, if you connect to pin 3, use `digitalPinToInterrupt(3)` as the first parameter to `attachInterrupt()`.

BOARD	DIGITAL PINS USABLE FOR INTERRUPTS
Uno, Nano, Mini, other 328-based	2, 3

Figure 4: `attachInterrupt` Function on the Arduino Website

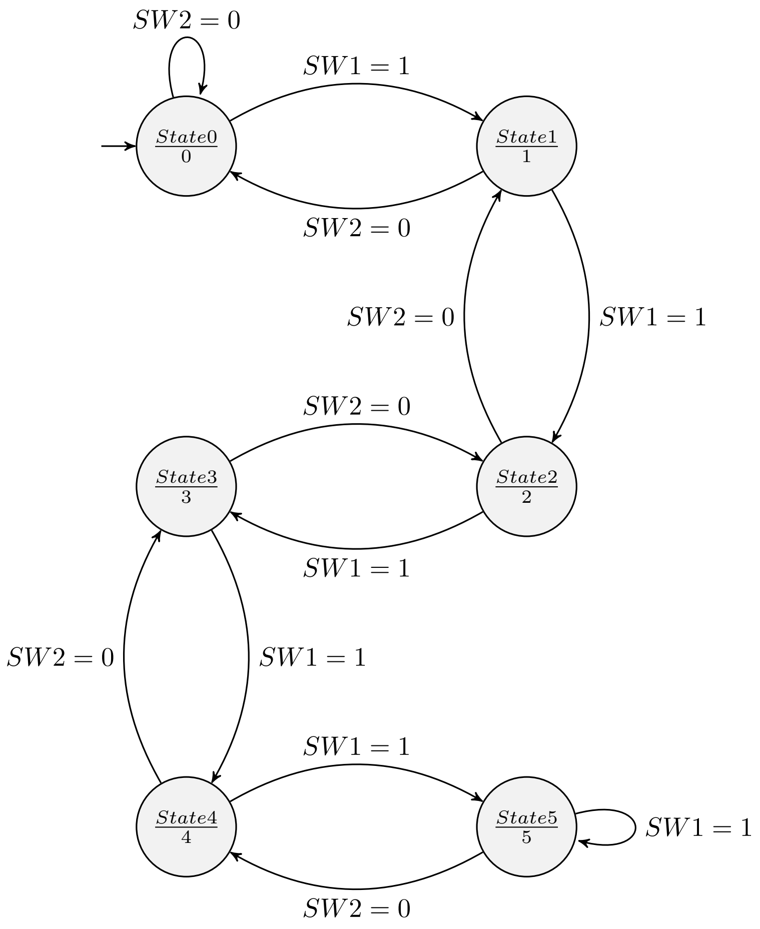
5 Troubleshooting

5.1 Solutions

We have included the solutions below if you do not complete the lab during the session or if you want to verify your answer. If you need help during the lab ask an instructor!

- TinkerCAD Solution
- Firmware Training Board Template Solution
- State Machine Solution

Current State	Input	Next State	LED Output
State 0	SW2 = 0	State 0	0
State 0	SW1 = 1	State 1	1
State 1	SW2 = 0	State 0	0
State 1	SW1 = 1	State 2	2
State 2	SW2 = 0	State 1	1
State 2	SW1 = 1	State 3	3
State 3	SW2 = 0	State 2	2
State 3	SW1 = 1	State 4	4
State 4	SW2 = 0	State 3	3
State 4	SW1 = 1	State 5	5
State 5	SW2 = 0	State 4	4
State 5	SW1 = 1	State 5	5



6 Appendix

6.1 Installing the Arduino IDE

We will be using the legacy version of the Arduino IDE (version 1.8.19). Follow the installation instructions based on your platform.

6.1.1 Windows and Mac

1. Go to the Arduino Software page and scroll down to the "Legacy IDE (1.8.X)". Download the binary corresponding to your platform.

2. Run the installer. Allow the installer to install everything, including drivers.

6.1.2 Linux

1. Go to the Arduino Software page and scroll down to the "Legacy IDE (1.8.X)". The Linux Arduino IDE should download as a `tar.xz` file.

2. Run the following commands, replacing `<arduino-ide>` with the name of the downloaded file.

```
tar -xJvf <arduino-ide>.tar.xz -C /home/$(whoami)
cd /home/$(whoami)/<arduino-ide>
sudo ./install.sh
```

6.2 Verify CH340 Drivers

Our Arduino Nanos are kinda old, so they often don't work out of the box without installing the correct drivers first. Your operating system or the installation of the Arduino IDE might have installed the correct drivers, so it's a good idea to check if you have the driver before reinstalling them.

6.2.1 Windows

Checking if driver is present: type `driverquery` into the command prompt. You should see a driver called `CH341SER`.

Checking if driver loads: Plug the Arduino Nano into your computer. Go to `Device Manager` and look under `Ports`. You should see the Arduino Nano as `USB-SERIAL CH340`, followed by the `COM` port it's associated with.

6.2.2 Linux

Checking if module is present: type `modinfo ch341` into your shell. The command should return a bunch of information about the `ch341` kernel module.

Checking if driver is present: Plug the Arduino Nano into your computer. Enter the command `sudo dmesg` into your shell.

- If you see the following, you need to uninstall `br1tty`. After uninstalling `br1tty`, unplug and plug in the Arduino Nano and run `sudo dmesg` again.
  - Ubuntu: `sudo apt remove br1tty`.
- If you see the following, type `ls /dev/tty*`. You should see the Arduino Nano show up as `/dev/ttyUSB[0-9]` or `/dev/ttyACM[0-9]`. This indicates that the kernel module has loaded successfully.

6.3 Install CH340 Drivers

If you could not find the `CH340` drivers, you may have to install them manually.

6.3.1 Windows and Mac

Follow the instructions