ECE 177 – Programming I: From C Foundations to Hardware Interaction Lecture 15

Vince Weaver

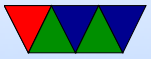
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

2 March 2026

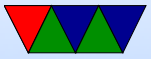
Announcements

- HW#4
- There is going to be a midterm review for this class at 5pm on Wednesday March 4th in Barrows 130 with pizza
- Lab#5 this week



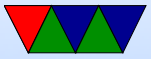
Homework Questions

- "Toggle Switch". If you have a true (not-zero) value how can you switch it to false (0) and vice-versa
- One way is to have if/else or even case/switch but that's overkill
- Can you think of a way to do it with one of the logical operators?
- Note: not bitwise (in theory you *could* do it with XOR but only if it's only the 1/0 case)
- Remember the NOT or '!' operator



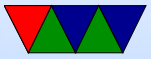
Lab5 – Larson Scanner

- Light pattern where lights go right to left with a bit of a pause
- To be technically correct the moving light should be a few lights wide, dimmer on either side, but that's harder to do than just having one light on
- Named for Glen A. Larson who created many TV shows, including both Knight Rider and Battlestar Galactica (where the cylons have one)



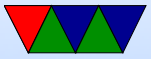
Lab5 – Knight Rider

- `https://www.youtube.com/watch?v=oNyXYPhnUIs`
- Can't tell from the intro, but the car's controlled by an AI named K.I.T.T. that talks
- Also "turbo boost" button
- Drives into the back of a semi-truck on the highway
- One of the bad guys is another AI car named K.A.R.R. so I find the recent KAR license plates amusing



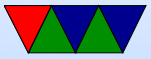
Lab5 – Lighting up LED

- LED – Light Emitting Diode
  - Diode is a semiconductor device that only lets electricity flow one direction
  - It turns out specially crafted one, due to quantum effects, if electrons jump from one voltage to another can release a photon while doing it
  - Depending on the voltage drop this can be a variety of colors



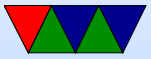
Lab5 – Powering an LED

- It's not just voltage, but current
- You don't want to put too much current across them or it will emit light *very* brightly and briefly until the "magic smoke" will escape
- To prevent that, use a "current limiting resistor" to keep the current low enough. Ohm's Law $V = IR$, $I = V/R$ V=3.3, R=510 Ohms so I is roughly 6mA?
- Also good to keep current low as the Pico itself can be hurt if you draw too much current from it



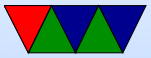
Lab5 – GPIOS

- How do you turn on or off a pin coming off the Pico?
- Use something called "General Purpose Input/Output (GPIO)
  - Allow setting output to 3.3V or 0V
  - Can also use as inputs (to read 3.3V of 0V)
  - Lots of fancy stuff you can do with them that isn't important for this lab



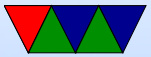
Lab5 – Memory Mapped I/O

- On embedded boards like this, with ARM processors, the way you talk to hardware is via Memory Mapped I/O (MMIO)
- The CPU monitors certain memory addresses, and instead of treating them like RAM it sees the reads and writes and takes action
- You can see the manual for this
- There are a series of registers in memory controlling the GPIO pins



Lab5 – Pi Pico GPIO

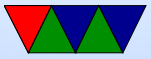
- Only so many pins (40 on this board) so not all GPIOs make it to the pins, and also some of them are used for other purposes (like the keypad, and SPI for the display)
- GP0-GP22, GP26-GP28 available on pins of Pico
- Our board is configured so GPIO13..GPIO22 connect to the 10 LEDs in the bargraph LED



Lab5 – Turning on a GPIO Line

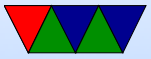
- If coding this yourself can involve some really low level messing with MMIO registers, bit-shifts, and other things
- You will see that in ECE271
- For this first part we can use some helper routines that are provided for you

```
gpio_init(26);                  // initialize GPIO 26
gpio_put(26,0);                 // set GPIO26 output to 0
gpio_set_dir(26,GPIO_OUT);      // set GPIO26 to OUTPUT mode
gpio_put(26,1);                 // set GPIO26 output to 1 (3.
```



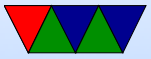
Lab5 – Toggling a GPIO Line

```
int outval=0;
gpio_init(26);                  // initialize GPIO 26
gpio_put(26,0);                 // set GPIO26 output to 0
gpio_set_dir(26,GPIO_OUT); // set GPIO26 to OUTPUT mode
while(1) {
    gpio_put(26,outval);
    sleep_ms(1000);
    outval=!outval;
}
```



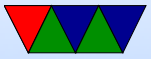
Lab5 – Scanner

- To do the Larson scanner, want to start with the rightmost (GPIO13) on
- Then turn off 13 and turn on 14, then turn off 14 and turn on 15, etc
- Once you hit 22 stop, then move to left
- You could in theory do this with the routines we've discussed, but we want you to use another way



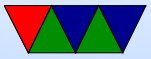
Lab5 – Using a bitmap for output

- The GPIO interface also lets you write many pins at once like a bitmap
- To learn how this works we're going to have to learn about structs



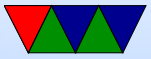
structs

- We talked about arrays, where you can group together multiple of same data type, i.e. 10 ints `int a[10]`
- What if you want to group together variables of different types?
- Or what if you want to group together same types but have them named instead of indexed?
- In C this is a struct (other languages have a somewhat similar idea called a `class`



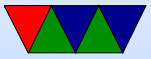
defining a struct

```
struct location {
    int x;
    int y;
    int color;
};
struct location my_location;
```



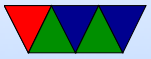
setting values

```
struct location my_location;
my_location.x=4;
my_location.y=3;
my_location.color=16;
```



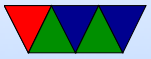
reading values

```
printf("X=%d Y=%d color=%d\n",
     my_location.x,
     my_location.y,
     my_location.color);
```



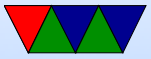
Advanced: initializing

```
struct fancy {
    int x;
    double rate;
    char letter;
} fancy1 = { 10, 4.5, 'C' };
```



Advanced: arrays of structs

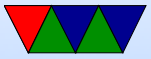
```
struct fancy our_fancy[10];
our_fancy[5].x=15;
```



Advanced: pointers to structs

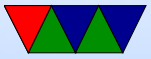
```
struct fancy *fancy_ptr;
struct fancy fancy1;
fancy_ptr=&fancy;

/* if pointer, need to use -> to dereference */
/* instead of . */
fancy->x=15;
```

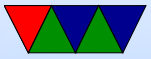


Lab5 – Accessing Memory like a Struct

- `pico-sdk/src/rp2040/hardware_structs/include/hardware/structs/sio.h`
- Sets up a struct that matches the MMIO registers
- These MMIO registers live at a predefined ADDRESS in memory
- So if we set up a STRUCT that matches the layout of this, and then set up a POINTER that points to the address, then we can use struct notation to access memory like this more easily
- So that's what we do, in this case it is called "sio" and



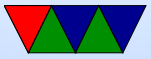
various GPIO registers live in there.



Lab5 – Outputting using a Struct

```
spio_hw->gpio_out;
```

- This struct element is a 32-bit integer
- These bits map 0 to GPIO0, 1 to GPIO1, etc
- So if we write a 32-bit value to it, that bit pattern gets put on the GPIO pins
- BE CAREFUL! Only set pins you want to set, if we set some to 0 or 1 that are doing other things can break things



Lab5 – Setting use the Struct

- So to turn on GPIO15 we just need to set bit 15 to 1

```
spio_hw->gpio_out=spio_hw->gpio_out|(1<<15);
```

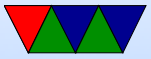
- What if want to set two of them on?

```
spio_hw->gpio_out=spio_hw->gpio_out|(1<<15)|(1<<16);
```

- Remember you can shortcut this,

```
spio_hw->gpio_out|=(1<<15)|(1<<16);
```

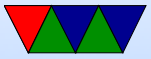
- The key part is a read-modify-write is happening, you read the value, change it, then write it back out



Lab5 – What if you want to clear?

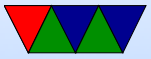
- Want to turn off GPIO15?
- Need to use a mask. We want to bitwise AND it with a value with all bits 1 except for bit 15. How can we do that?

  `~(1<<15)`
- So want to load this, and with the mask above, then save it out `spio_hw->gpio_out&=~(1<<15);`



Lab5 – Finally Larson Scanner

- The code wants you to start with a bit in GPIO13 `out=(1<<13)`
- To go left, use shifting to move left, so `out=out<<1;`
- Until it gets to GPIO22
- Then have a second loop that shifts it right until it gets to 13
- Then repeat.
- Note while doing this you're going to have to mask so you don't touch any other GPIO bits



- To clear all the bargraph bits you'd read value, then MASK with

```
          33222222 22221111 111111
          10987654 32109876 54321098 76543210
       0b11111111 10000000 00011111 11111111
0xFF801FFF
```

- get value from GPIO, mask it, or in your shifted value, then write back out