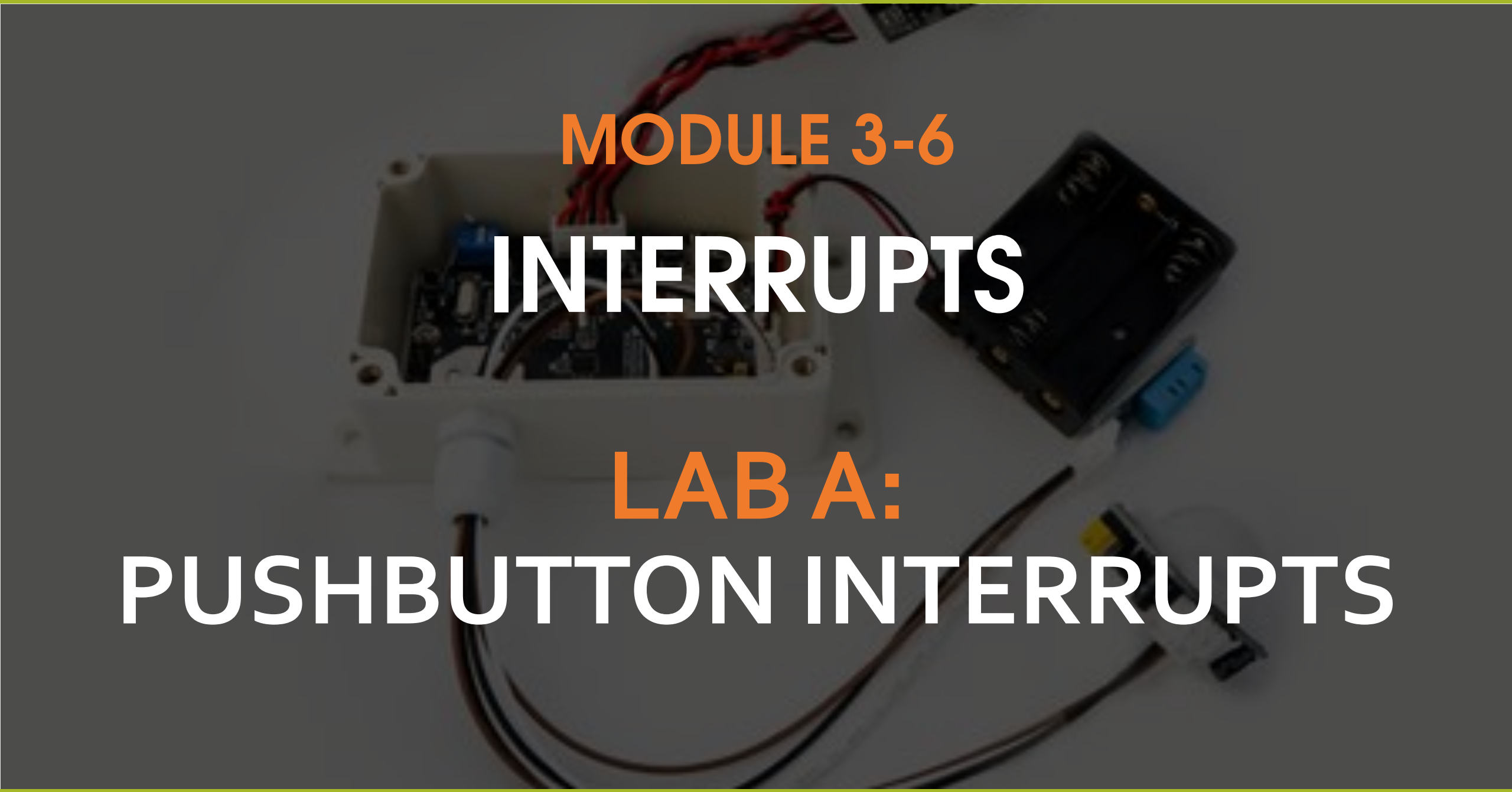

VIRTUAL COURSE
BUILD YOUR OWN
DATA LOGGER
WILDLABS.NET
[The conservation technology network]
FREAKLABS



MODULE 3-6

INTERRUPTS

LAB A:

PUSHBUTTON INTERRUPTS

Goals

- Learn about pushbuttons and the quirks we need to adjust for in code

Program a pushbutton to turn an LED on and off to learn how to:

- Initialize an interrupt
- Write an interrupt service routine (ISR)
- How to “handle” an interrupt event

Why Is This Important?

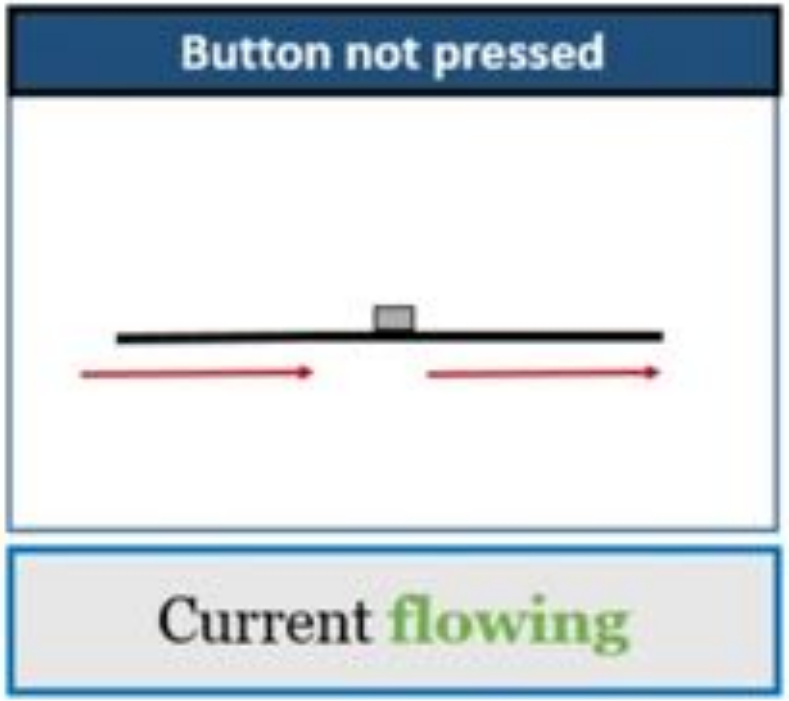
- Pushbuttons are a simple way to learn about interrupts
  - Give us full control of when our interrupt event happens (ie. button press)
  - Can easily test our code
  - Gives more confidence in the reliability of our application
- Pushbuttons are useful in all kinds of applications
  - User interface
  - Start/stop of an application, or specific functions
  - Change device settings / parameters

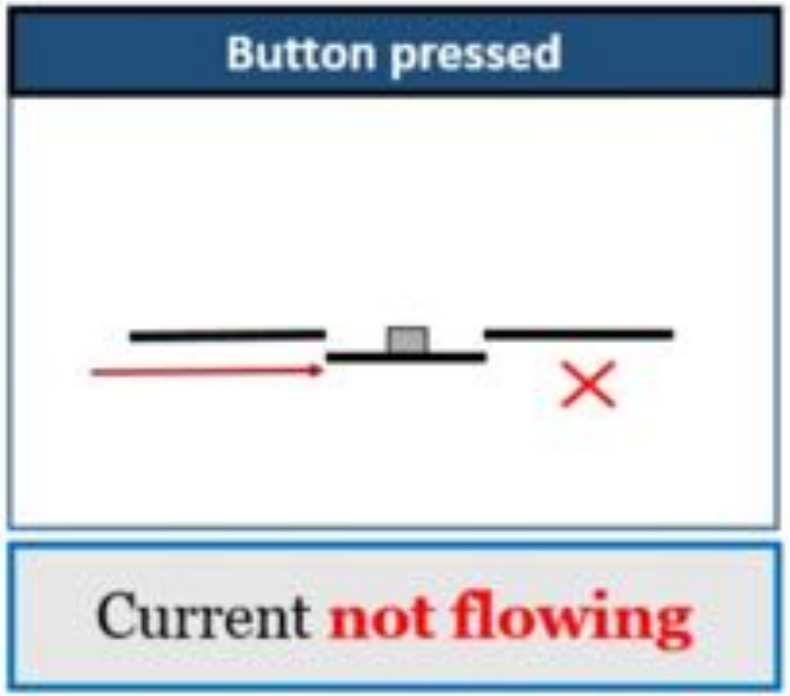
Pushbuttons

Pushbuttons – Momentary Switches

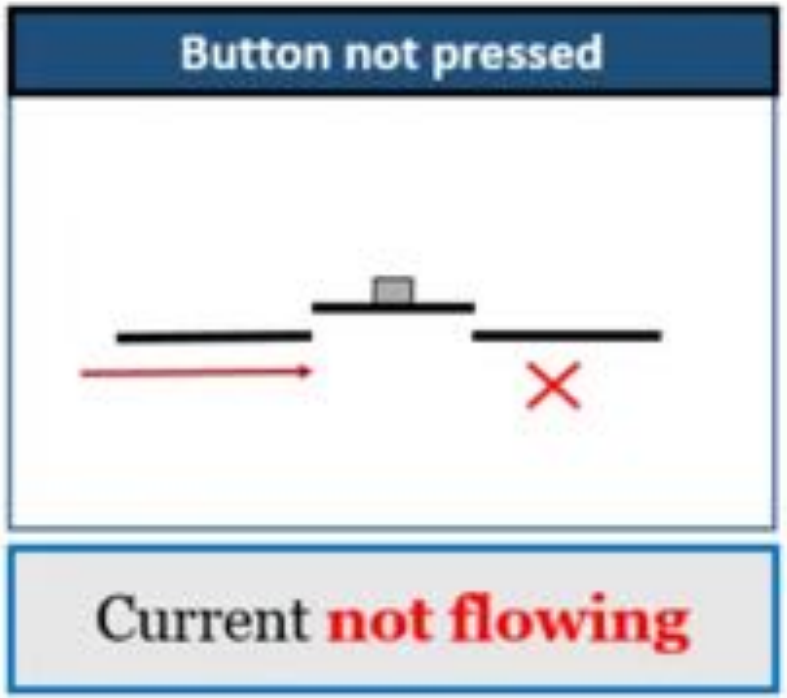
vs

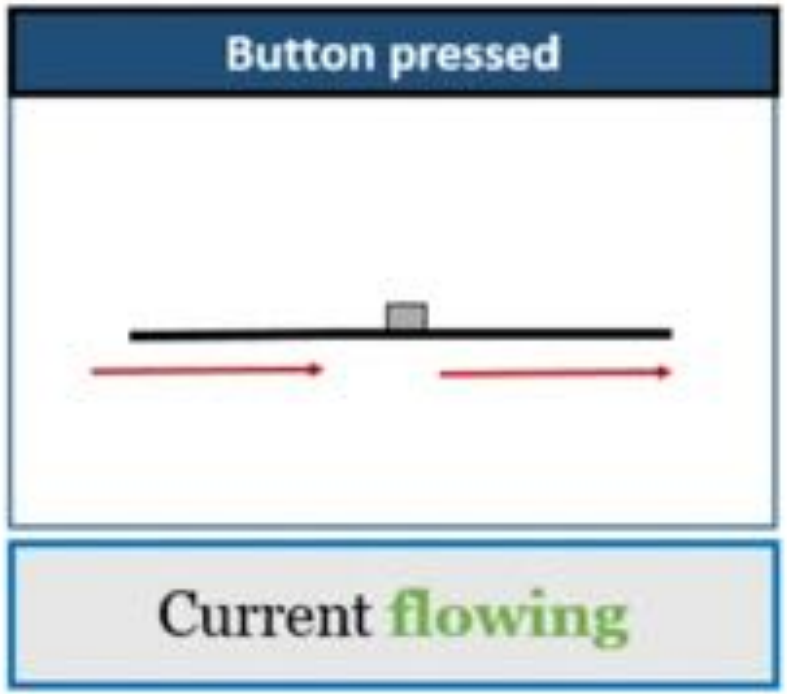
Pushbuttons - Active Low



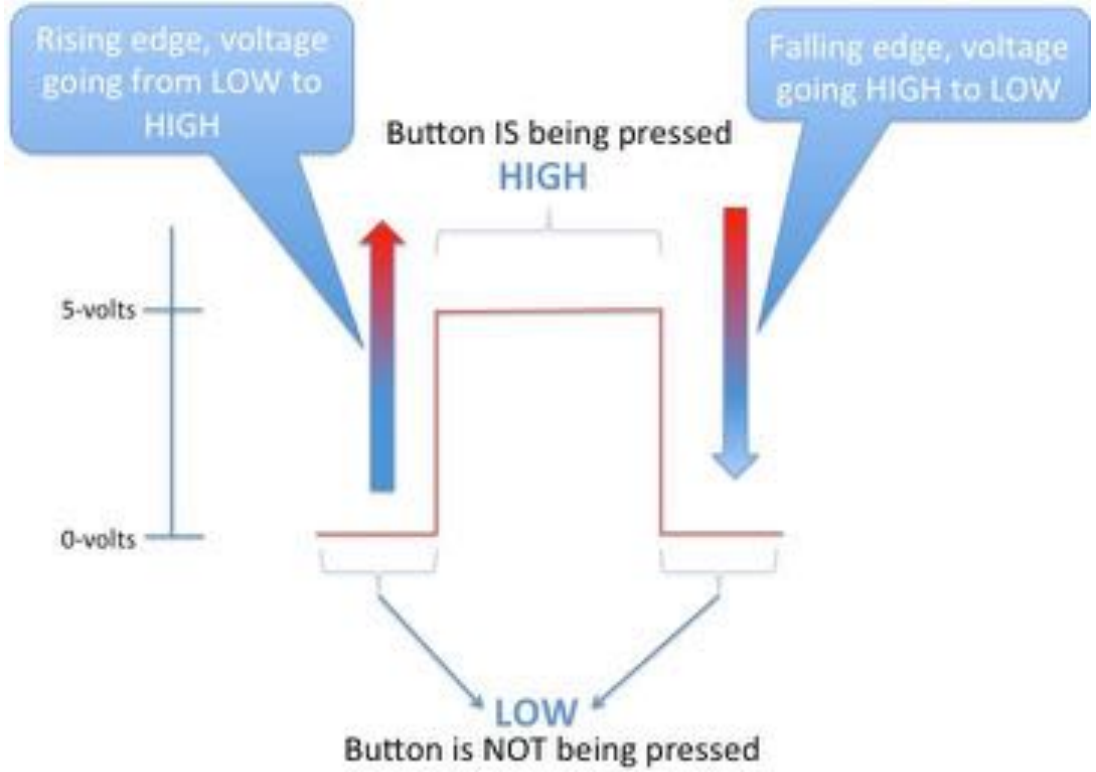


Pushbuttons - Active High



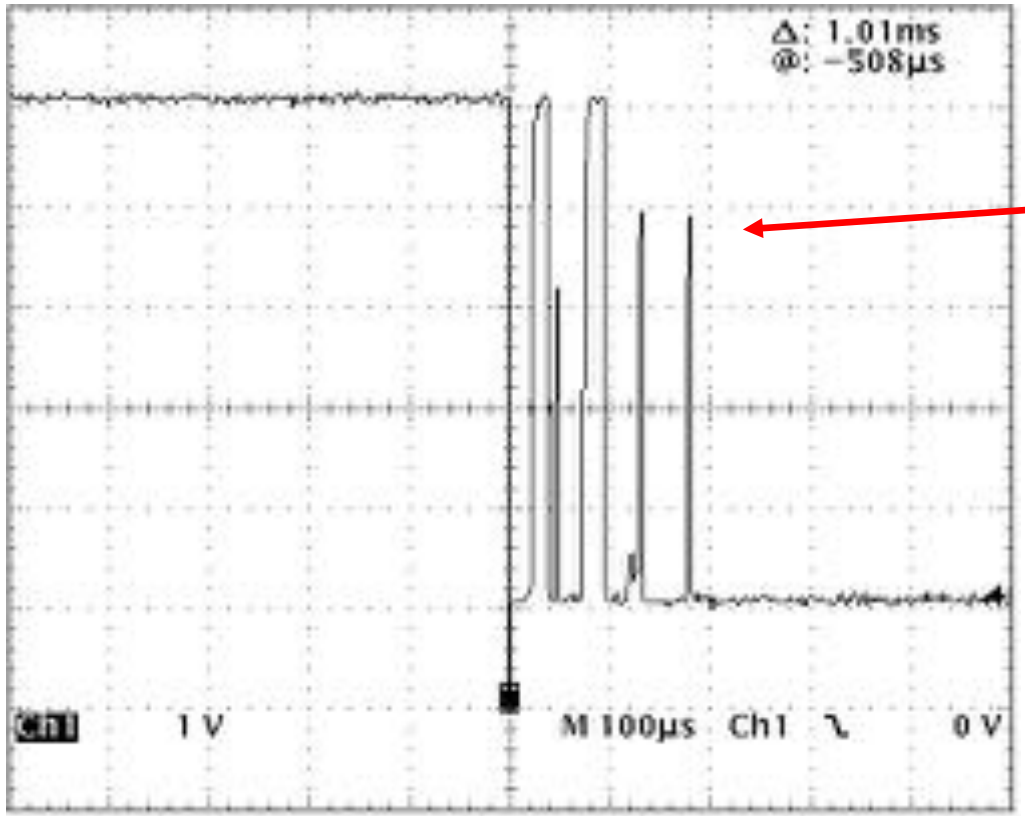


Pushbuttons – Transitions or Edges



From: http://www.programmingelectronics.com/tutorial-19-debouncing-a-button-with-arduino-old-version

Pushbuttons – Debounce Delay



micro connections

IDLE BOUNCE PUSHED

Wild Logger Interrupt Mapping

Used in attachInterrupt()

Used to access interrupt pin's properties

Interrupt Source	Interrupt Number	Pin Number
Real Time Clock	0	2
PIR Motion Sensor/Aux Intp	1	3
Pushbutton 0	2	6

What Do I Need to Know?

- **attachInterrupt(interruptNum, isrFunction, type)**
  - interruptNum = interrupt number
  - isrFunction = the name of our interrupt service routine function that handles the interrupt
  - type:
    - FALLING = falling edge
    - RISING = rising edge
    - CHANGE = any edge
- **void isrFunction()**
  - Boilerplate function for interrupt service routine
- **volatile int varName**
  - variables used inside ISRs need to be declared as volatile along with their data type
  - This tells the compiler not to delete the variable when optimizing our code

Our Code

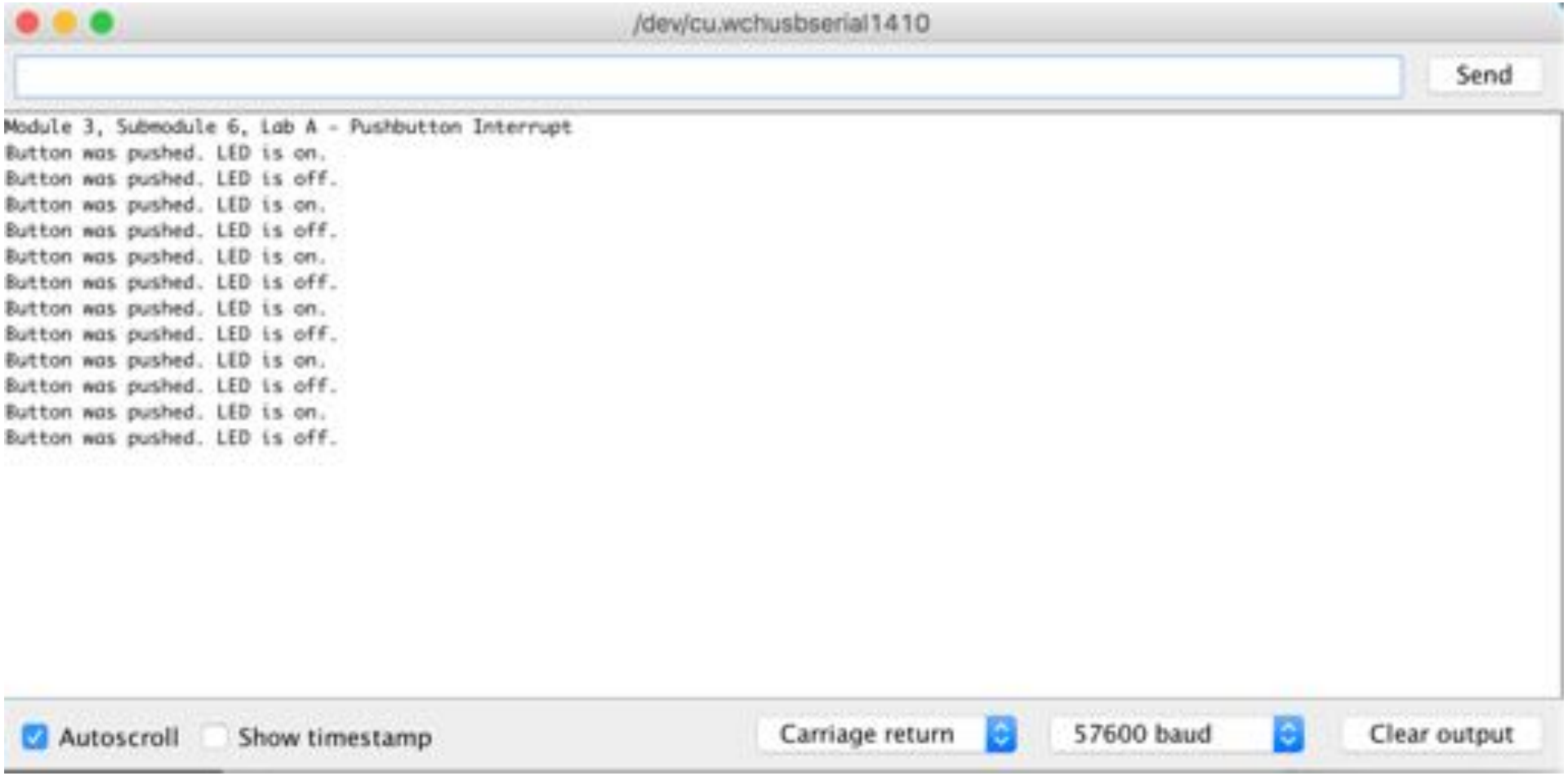
```
#define DEBOUNCE_TIME 50

int intpButton = 2;
int pinLed = 4;
volatile int flagButton;

void setup()
{
    pinMode(pinLed, OUTPUT);
    digitalWrite(pinLed, LOW);

    Serial.begin(57600);
    Serial.println("Module 3, Submodule 6, Lab 6a - Pushbutton Interrupt");

    attachInterrupt(intpButton, isrButton, FALLING);
}
```

Continues on next page ...

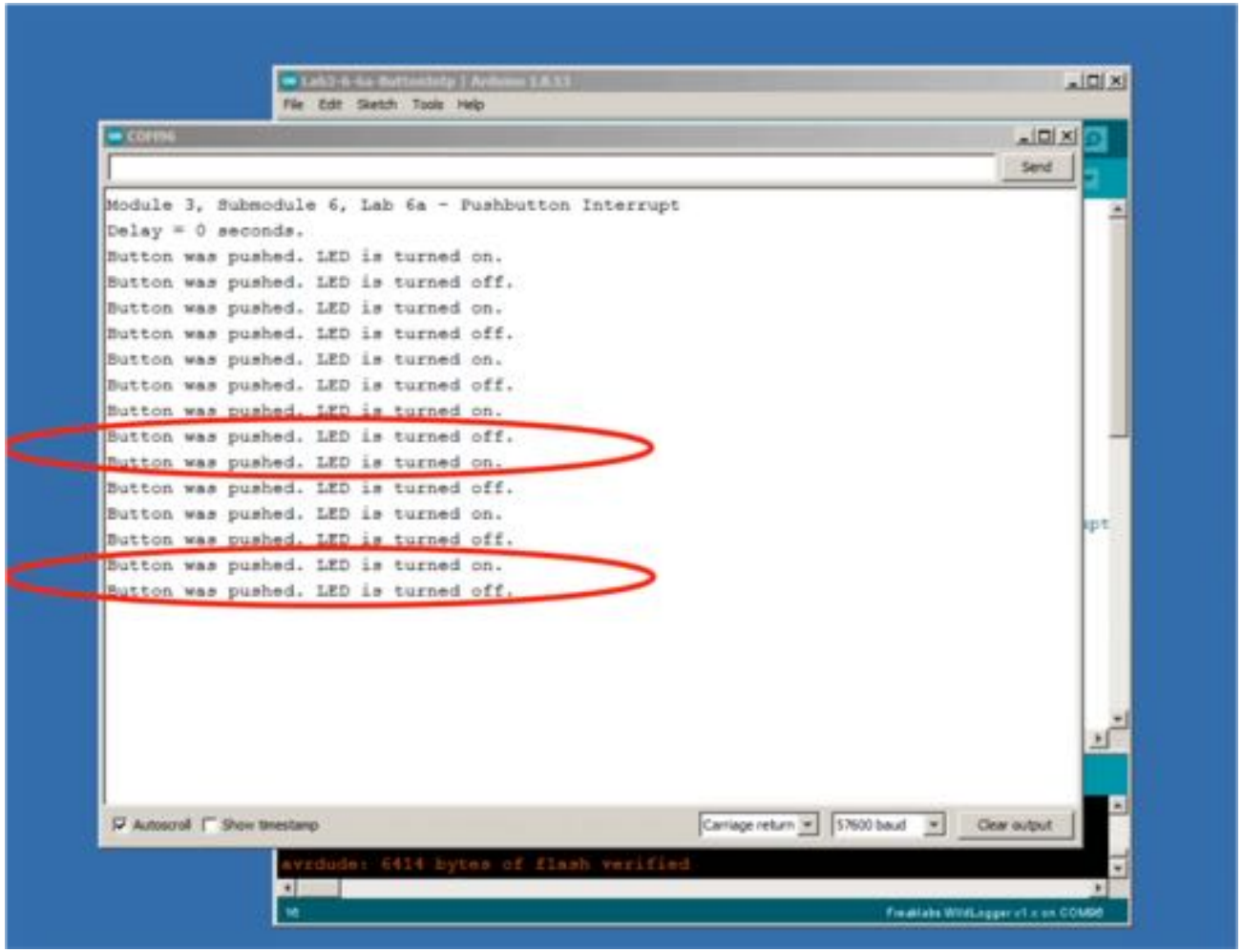
Our Code


3.6.2-LabA §
18
19 void loop()
20 {
21 if (flagButton == 1)
22 {
23 int ledVal = digitalRead(pinLed);
24
25 delay(DEBOUNCE_TIME);
26 flagButton = 0;
27
28 if (ledVal == 0)
29 {
30 Serial.println("Button was pushed. LED is on.");
31 digitalWrite(pinLed, HIGH);
32
33 }
34 else
35 {
36 Serial.println("Button was pushed. LED is off.");
37 digitalWrite(pinLed, LOW);
38 }
39
40 }
41
42 }
43
44 void isrButton()
45 {
46 flagButton = 1;
47 }
48

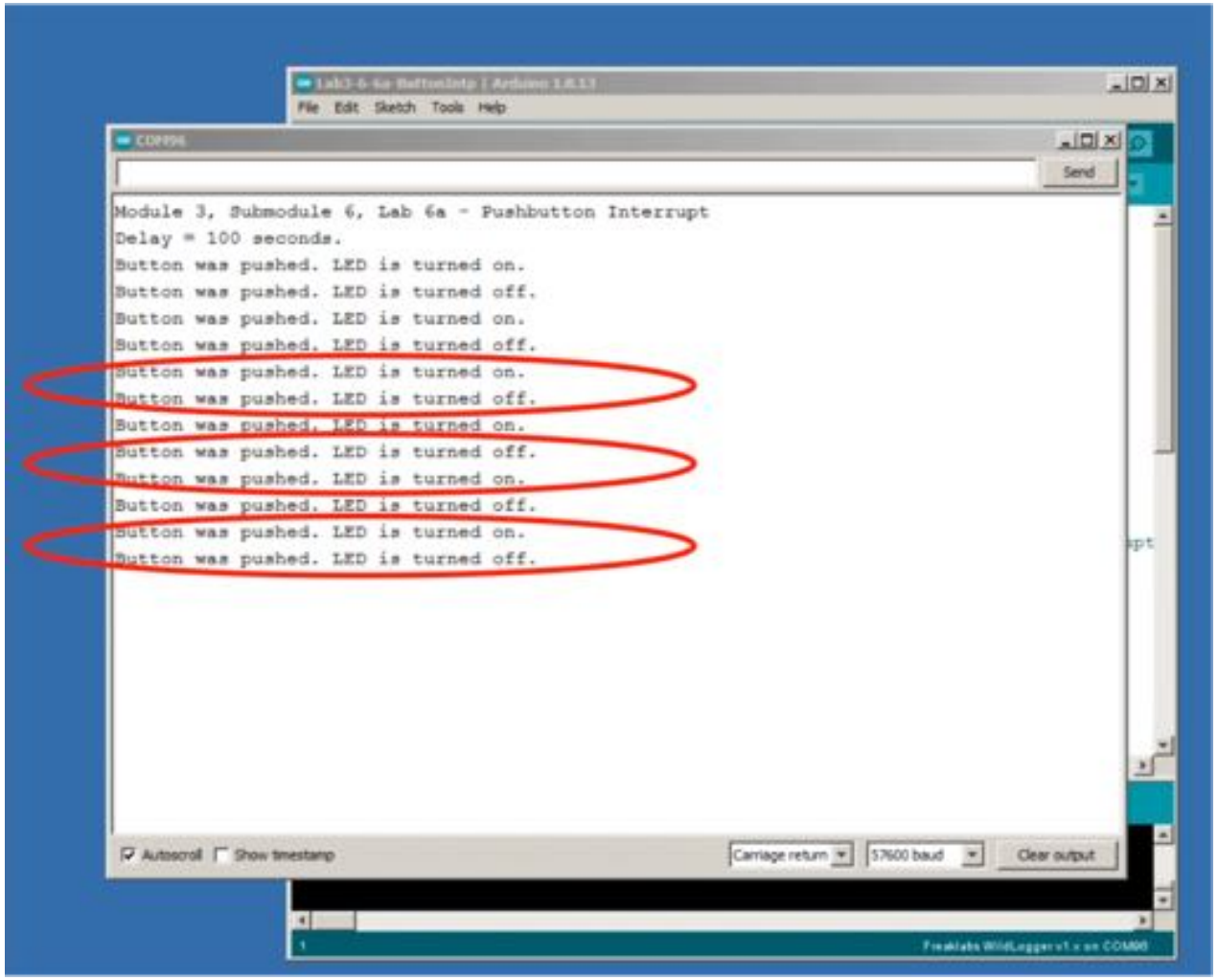
Output


/dev/cu.wchusbserial1410
Send
Module 3, Submodule 6, Lab A - Pushbutton Interrupt
Button was pushed. LED is on.
Button was pushed. LED is off.
Button was pushed. LED is on.
Button was pushed. LED is off.
Button was pushed. LED is on.
Button was pushed. LED is off.
Button was pushed. LED is on.
Button was pushed. LED is off.
Button was pushed. LED is on.
Button was pushed. LED is off.
Button was pushed. LED is on.
Button was pushed. LED is off.
Autoscroll
Show timestamp
Carriage return
57600 baud
Clear output

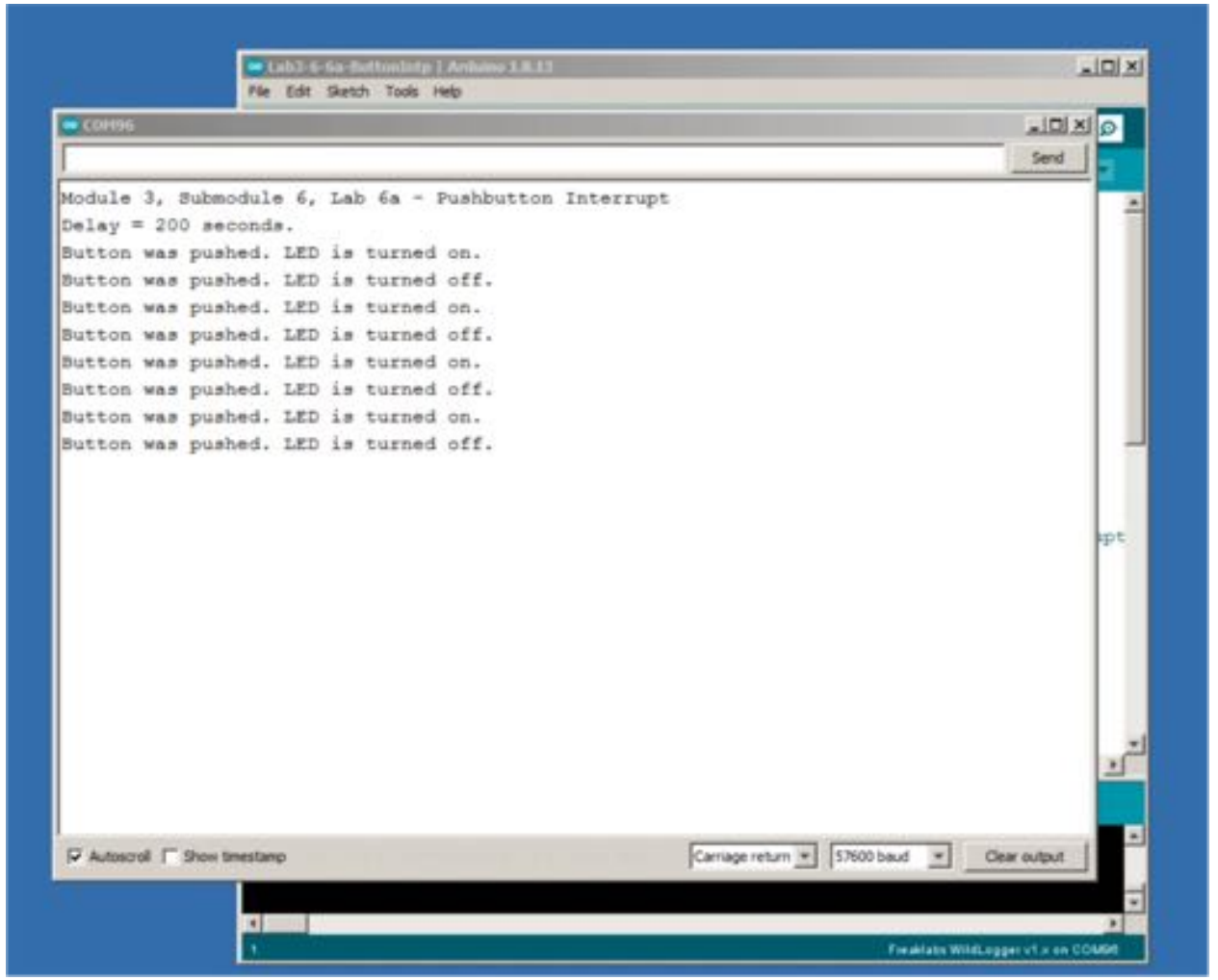
Output –Debounce Delay Value = 0

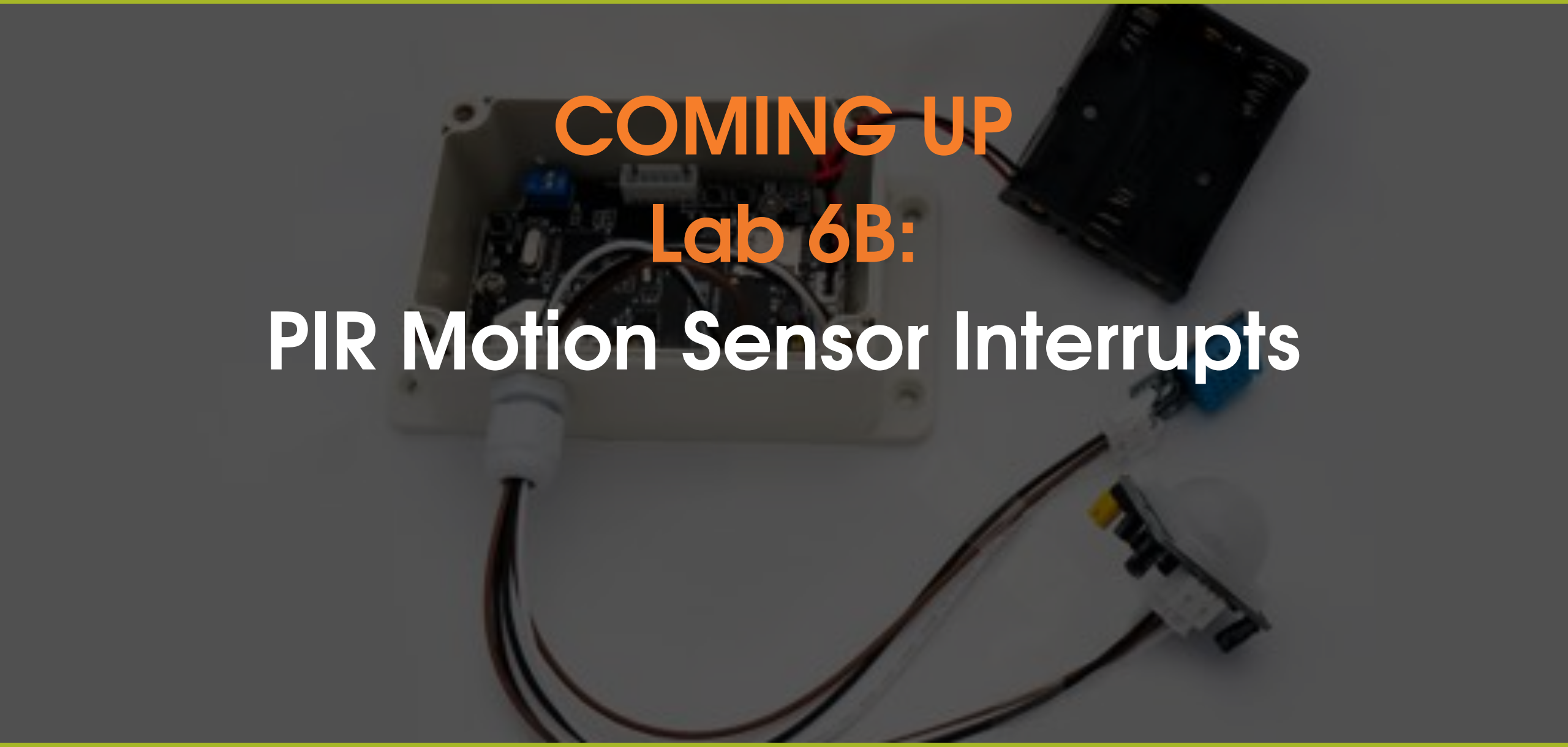


Output –Debounce Delay Value = 100 ms



Output –Debounce Delay Value = 200 ms





COMING UP

Lab 6B:

PIR Motion Sensor Interrupts