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

- Button not pressed: Current not flowing
- Button pressed: Current flowing
Pushbuttons – Transitions or Edges

Pushbuttons – Debounce Delay
### Wild Logger Interrupt Mapping

- **Used in attachInterrupt()**
- **Used to access interrupt pin’s properties**

<table>
<thead>
<tr>
<th>Interrupt Source</th>
<th>Interrupt Number</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Time Clock</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>PIR Motion Sensor/Aux Intp</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Pushbutton 0</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
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

```c
#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);
}
```
void loop()
{
  if (flagButton == 1)
  {
    int ledVal = digitalRead(pinLed);
    delay(DEBOUNCE_TIME);
    flagButton = 0;
    if (ledVal == 0)
    {
      Serial.println("Button was pushed. LED is on.");
      digitalWrite(pinLed, HIGH);
    }
    else
    {
      Serial.println("Button was pushed. LED is off.");
      digitalWrite(pinLed, LOW);
    }
  }
}

void isrButton()
{
  flagButton = 1;
}
Output

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.
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