Implementing Interrupts with the Arduino

Objectives

- Understand what interrupts are and how they can be implemented

Interrupts – an analogy

Think of some repetitive task as a program

For example, driving to school:

There are multiple steps, done in a sequence, with time constraints…

1. Start the car
2. Move down the driveway
3. Turn at proper intersections
4. Stop at stop signs
5. Find a parking place (hopefully)

The interrupt

However, a police car is following you, and last time you got pulled over for speeding with just a warning – this time it will be a ticket!

So, what will you do differently / more often??

- Check rearview
- Check speedometer

What is important about these actions?

- They must be done *often*
- They must be done *very quickly*

- Consequences of not following above rules:
  - You might not see you're speeding (not keep track of your speed)
  - You might hit the car in front of you (not enough time doing main task)

What does this have to do with Arduino programming??

- There are times in a program where some event can occur quickly, but we can't miss it.
- For example, a pushbutton used for a stopwatch…
- If we don't capture the event at the exact moment it occurs, it may not work properly
- Do we know ***when*** the button will be pressed?
  - These are "Asynchronous" events…

One way to solve this problem is using *interrupts*

- An interrupt can occur at *any* time
- An interrupt must be VERY short so it doesn't cause the main program to be delayed

Interrupts on the Arduino

- Two types of interrupts:
- Hardware (external)
  - Triggered by a hardware input pin
- Software (internal)
  - Triggered by an internal timer or other event

Hardware Interrupts on the Arduino

- Two main parts of an interrupt:
- Interrupt hardware definitions
  - Cause / Condition
- Interrupt Service Routine (ISR)
  - Short function that runs when interrupt occurs

Interrupts on the Arduino

- Interrupt hardware definitions

attachInterrupt (0, switchPushed, FALLING);

Arduino built-in command – done in setup

Which pin is used (hardware interrupt) NOTE:
0 = pin 2
1 = pin 3

Name of ISR (function that runs when interrupt is activated)

Condition of hardware interrupt (falling edge of pulse)

What does the interrupt look like?

Interrupt occurs at this point in program

```
void loop()
{
  val = analogRead(A0);
  if (val > 100)
     digitalWrite(LED1, HIGH);
  else
     digitalWrite(LED1,LOW);
}
```

Loop runs forever *unless* an interrupt occurs

Program jumps out of the loop to the ISR

When ISR is completed, program continues at the point it left off…

```
void switchPushed()
{
   digitalWrite(LED2,HIGH);
}
```

Example Program - setup

```
int redLED = 13;
volatile int buttonPressed = 0;  // use 'volatile'
                                 //  with ISR


void setup()
{
    pinMode(redLED, OUTPUT);
    attachInterrupt (0, switchPushed, FALLING);
}
```

Example Program

```
void loop()
{

 if (buttonPressed == 1)
 {
      digitalWrite (redLED,HIGH);        // turn on LED for
      delay(500);                        // ½ second
      digitalWrite (redLED,LOW);
      buttonPressed = 0;
  }
  else
     digitalWrite(redLED, LOW);
  }
```

Example Program

```
void switchPushed()          // ISR
{
   buttonPressed = 1;
}
```

Software interrupts

- http://www.hobbytronics.co.uk/arduino-timer-interrupts
- http://homediyelectronics.com/projects/arduino/arduinotimerinterruptexample/