ME 133 Winter 2022
Lab 2: *ADC and Interrupts*
January 20, 2022
Due: 1/31/2022

Submit a `zip` file named `yourFirstName-Lab2.zip` on Canvas with your code, a lab report (following the format in syllabus) – which must include **a schematic of the wiring** – and a short video proving the working hardware-software integration.

Exercise 1

Material required

- Arduino
- 2 x 330 Ohm to 1 K Ohm Resistor
- 2 x LEDs
- 1 x Potentiometer
- 1 x 10 K Ohm Resistor
- 1 x 10 K Ohm Photoresistor
- 1 x Button

Coding together

In this exercise we will learn how to read analog values from an analog sensor and look at serial communication, which allows the Arduino to send data directly to your computer.

Functions we will learn about

- `analogRead(pin)`: Reads the value from the specified analog pin. The Arduino board contains a 6 channel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, .0049 volts (4.9 mV) per unit.
- `analogWrite(pin, value)`:

  Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite() on the same pin). (NB: value: the duty cycle: between 0 (always off) and 255 (always on)).
- `map(value, fromLow, fromHigh, toLow, toHigh)`:

  Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.
- `Serial.begin(speed)`:

  Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.
- `Serial.print()` and `Serial.println()`:

  Prints data to the serial port as human-readable ASCII text.

Assignment

Connect two LEDs and a Potentiometer to the Arduino. Control the light emitted by the LEDs thought the potentiometer in a complementary pattern (the bigger the resistance in the potentiometer, the brighter one LED has to become, while the second become less bright). Have the value of the potentiometer printed on the serial monitor.

Extra credit

Have a third LED attached and control its status (ON/OFF) through a photoresistor. That is: the LED starts OFF, but when you cover it with your hand it turns on (put a 10 K Ohm resistor in series with the photoresistor!). Congrats, you created an automated light sensitive switch!

Exercise 2

Material required

- Arduino
- 1 x Button
- 2 x 330 Ohm resistors
- 2 x LEDs

Coding together

We will learn about external interrupts.

Functions we will learn about

- `attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)`:

  interrupt: the number of the interrupt. Allowed data types: int. pin: the Arduino pin number. ISR: the ISR (its a quick function) to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine. mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

  LOW to trigger the interrupt whenever the pin is low,

  CHANGE to trigger the interrupt whenever the pin changes value

  RISING to trigger when the pin goes from low to high,

  FALLING for when the pin goes from high to low.

Assignment

Explain what is the difference between the behavior of the button in Part 2 of Lab 1 with this case. What are the advantages of this later approach? Connect at least two LEDs and have one button to get them go from ON-ON -> ON-OFF -> OFF-ON -> OFF-OFF -> back to the start, but each action is triggered only when the user RELEASES the button! Show on the Serial Monitor when a change in the LED happens.

It is mandatory to use this as your `loop()` function:

```
void loop() {
  delay(10000);
}
```