AD32600 Physical Computing
Prof. Fabian Winkler
Fall 2014

Arduino Workshop 02 – Input: Sensors

This workshop is about connecting simple sensor to the Arduino board, exploring fundamental differences between digital and analog sensors and linking basic sensor input to LED output.

Review

Before we begin, here are two great resources to review some of the concepts introduced in the Arduiono workshop 01 last week (both are from the folks at Fritzing as part of their "Learning Arduino with the Fritzing Starter Kit" Youtube channel):

- Episode 003 – Digital Out with Blink
  https://www.youtube.com/watch?v=XQ_hDEoX5w8&index=3&list=PL8CD32146ED5CD04E
- Episode 005 – Breadboard Prototyping
  https://www.youtube.com/watch?v=CssOBb-qaX0&index=5&list=PL8CD32146ED5CD04E

A good reference and further reading for this workshop is Dan O'Sullivan and Tom Igoe's book *Physical Computing* – chapter 6: "The 'Big Four' Schematics, Programs and Transducers" (pp. 87 -137). You have online access to this book through the Purdue library.

Sensors Introduction

Sensors are electronic devices that measure a physical quality such as light or temperature and convert it to a voltage. This process of changing one form of energy into another is called transduction. Often, sensors are also referred to as transducers.

Sensors can be broadly classified in two categories: **digital** sensors and **analog** sensors. A digital sensor's output can only be in one of two possible states. It is either ON (1) often +5V, or OFF (0), 0V. Most digital sensors work with a threshold. Is the incoming measurement below the threshold, the sensor will output one state, is it above the threshold, the sensor will output the other state.

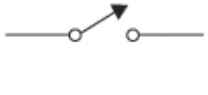
In contrast to a digital sensor, an analog sensor's output can assume any possible value in a given range. Very often the output of an analog sensor is a variable resistance that can be used to control a voltage. Rather than only being able to toggle between two states (digital sensor) the analog sensor can output an almost infinite range of values.

Digital Sensors

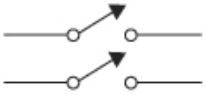
You can watch a good introductory video about digital sensors and the Arduino board from the folks at Fritzing here: https://www.youtube.com/watch?v=zyvCVN6p1M4&list=PL8CD32146ED5CD04E&index=6 (Episode 006 – Digital in with a button)

We will begin with the simplest digital sensor, the switch. When a switch is open, no current flows. In contrast, when a switch is closed, current flows (i.e. closed = ON). A switch that stays in the position it was put is called a latching switch. Switches can be spring loaded (e.g. microswitches/snap action switches), in this case they are called momentary. A simple switch can be Normally Open (NO) or Normally Closed (NC).

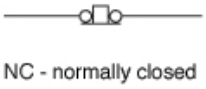
NO - normally open



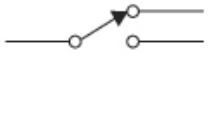
SPST - single pole, single throw



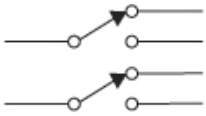
DPST - double pole, single throw



NC - normally closed



SPDT - single pole, double throw



DPDT - double pole, double throw

tactile switch/
pushbutton switch

subminiature switch/
snap action switch

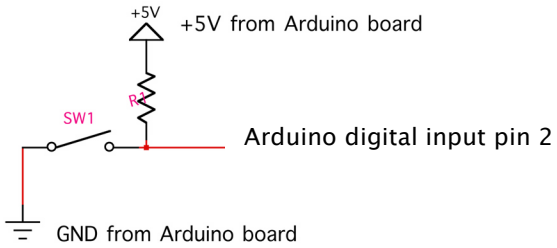
NO NC C

miniature toggle switch

Connecting a Switch to the Arduino Board

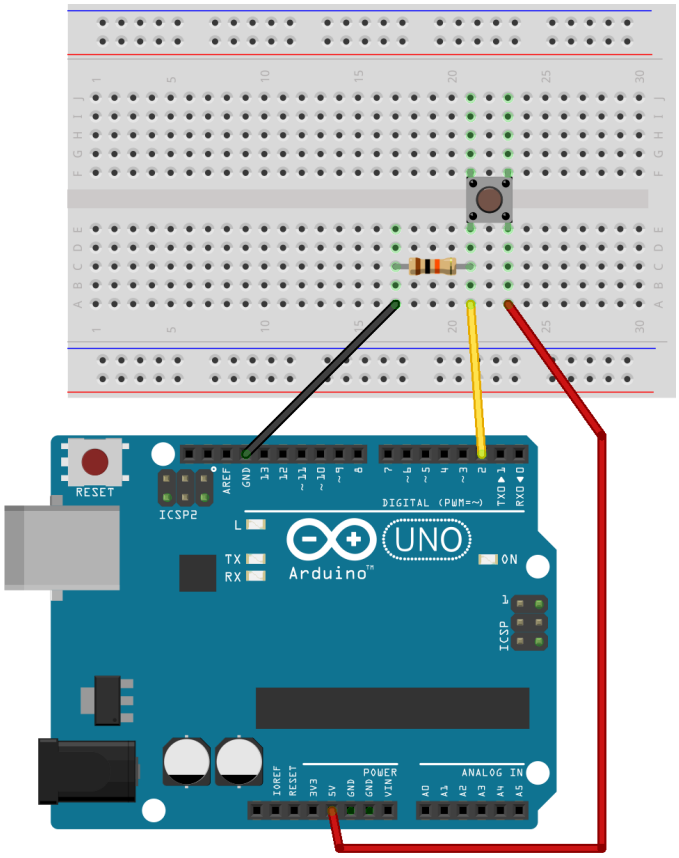
This is probably the simplest possible example to get started with Arduino. It uses an external switch and the Arduino board to turn ON or OFF the on-board LED (see: http://www.arduino.cc/en/Tutorial/Pushbutton):

1. Connect a switch (you can replace the switch with a plain jumper wire) to the Arduino board in the following way:



R1 = 10kΩ
SW1 can be any type of push button switch or just a wire

And translated into your breadboard view this circuit looks like this:



Why do we need the resistor R1? R1 guarantees that the Arduino's digital input pin 7 is connected to a constant voltage of +5V whenever the push button is not pressed. If the push button is pressed, the signal on pin 7 drops to ground (GND), at the same time the Arduino's +5V power is connected to GND, we avoid a shorted circuit by limiting the current that can flow from +5V to GND with a resistor (1 – 10 KΩ). Also, if there were no connection from pin 7 to +5V at all, the input pin would be "floating" whenever the pushbutton is not pressed. This means that it is connected neither to GND nor to +5V, picking up electrostatic noise leading to a false triggering of the input.

Now upload the following sketch to your Arduino board:
`File > Examples> 02. Digital > Button`

This example code turns on and off a light emitting diode(LED) connected to digital pin 13, when pressing a pushbutton attached to pin 2.

Fun with Switches

OK, you now know how to turn on and off an LED but you wouldn't really need an Arduino board to do that. Using the Arduino board really makes a difference once you start thinking about different ways of processing the digital input, doing more than just routing it 1:1 to your output. Here are a couple of examples of how you can already do more interesting things with just one switch and one LED:

- toggle the LED on and off: pushing the switch once turns the LED on, pushing it a second time turns it off.
- Incrementally fading the LED: each push of the button fades the LED either in or out using a set increment/decrement.
- Setting up a counter, only every third time you push the button the LED will turn on.
- Setting the LED to random brightness values (or using the tri-color RGB LED to set it to random colors) every time the button is pushed.

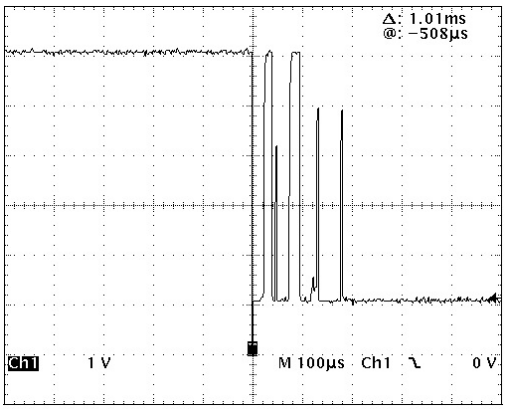
Let's take a look at the first scenario, and try to toggle the LED on and off based on the pushing of the button switch. First we need to make sure we only detect changes of the button state (`File > Examples > 02. Digital > StateChangeDetection` has an in depth discussion of this). However, rather than outputting the result of the state change detection to the serial window (we will use this window later in the workshop) we would like to visualize the state change by turn on or off the LED depending on its previous state (turn it on if it was off, turn in off if it was on).

Here is the code for performing this task:
http://web.ics.purdue.edu/~fwinkler/AD32600_F14/LEDtoggle.zip

Feel free and try to realize some of the other ideas mentioned above, the one where each push of the button generates a randomized color with the RGB LED would be a good one to get started with since it will require you to connect the LED and the switch to the Arduino board as well as coming up with the code yourself (http://arduino.cc/en/Reference/Random will give you information on how to generate random numbers with the Arduino board).

Switch Debouncing

Sometimes – in rare cases – when toggling the switch from on to off or vice versa, there is noise which could trigger the switch multiple times in what is perceived as only one push or release. On top of the next page is a visual representation (taken from an oscilloscope reading in the millisecond range) of this behavior. You can get rid of this noise and possible malfunctioning of the switch by "debouncing" it. `File > Examples > 02. Digital > Debounce` has the code and comments for it.

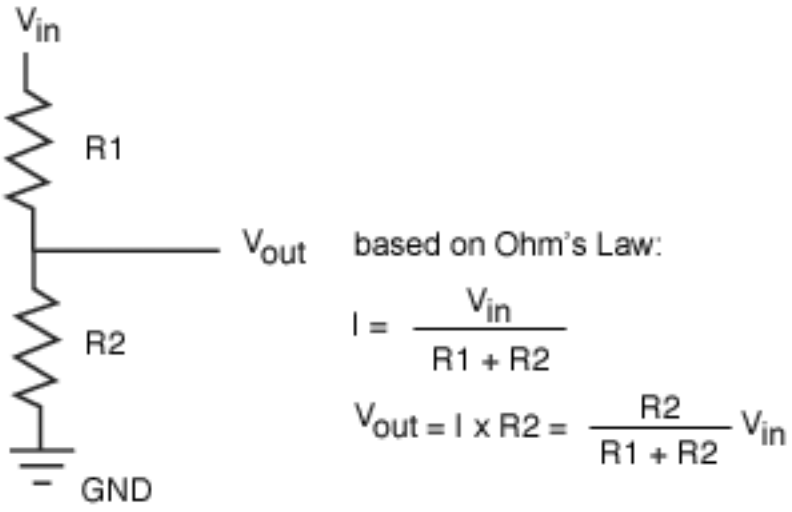


Noise of a “bouncing” switch, voltage range 0-5V, time scale 1ms per unit.

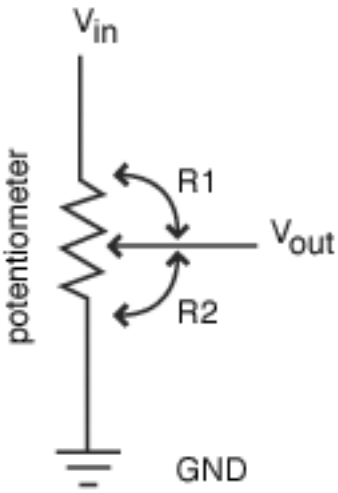
Analog Sensors

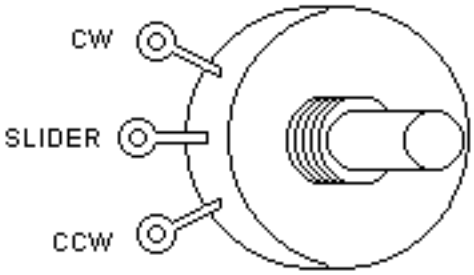
Analog sensors, such as a photocell (LDR) often work like a variable resistor the resistance of which is directly influenced by the condition it measures (in the case of the photocell it is the amount of light). Often a voltage divider circuit is used to turn the sensor’s changing resistance into a proportional change of voltage that can be understood better by ICs/microcontrollers (such as the Arduino) that you connect the sensor to.

In the following circuit diagram, a potentiometer is used as a voltage divider, with its slider dividing the overall voltage into two parts R1 and R2. We see that a voltage divider produces a predictable fraction of the input voltage as the output voltage.



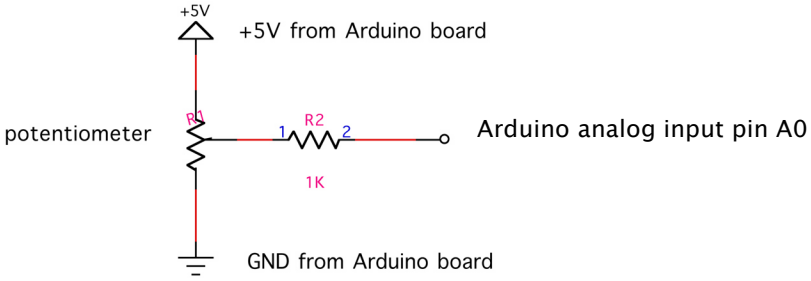
This general description of a voltage divider is related to the potentiometer as follows:



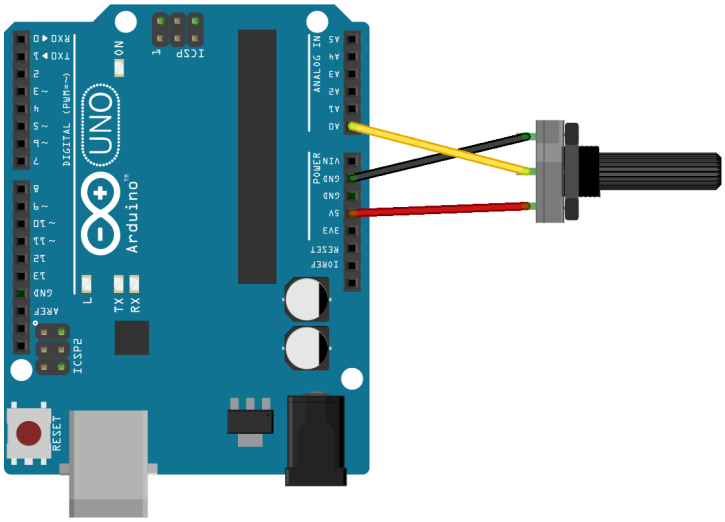


Connecting a Potentiometer to the Arduino Board

We use the following circuit to connect the potentiometer to the Arduino board. You may need to solder some wires on the potentiometer's pins to be able to connect it to the Arduino board.



And this is what it will look like in your breadboard view:



We are interested to see how Arduion is turning the rotation of the potentiometer's knob into numbers internally to represent interaction with the knob.

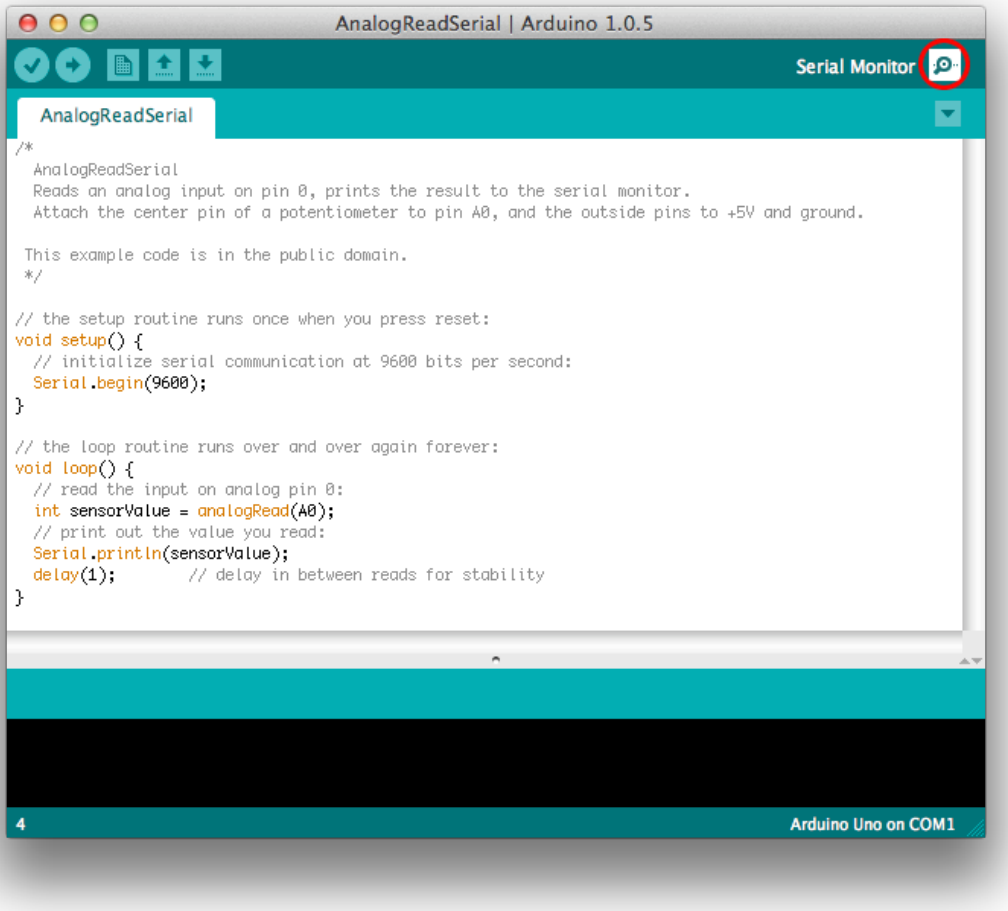
Open the following example code:

`File > Examples > 01. Basics > AnalogReadSerial`

Serial Communication

The above code example uses a communication strategy between the Arduino board and the host computer (PC or Mac). called "serial communication" to exchange the necessary data. In serial communication, data is transmitted one bit at a time over a single path. In order for the Arduino board and the host computer to understand each other, we need to work with a specific serial data protocol that defines e.g. communication speed (baud rate) and how to begin/end reading data. For now we use the serial output from the Arduino board as a convenient way to monitor data from sensors, later we will use a more complex serial communication protocol (Arduino Firmata) to communicate back and forth between the Arduino board and Processing.

After uploading the above example code, open the Serial Monitor window by clicking on the icon in the top right corner of the Arduino software:

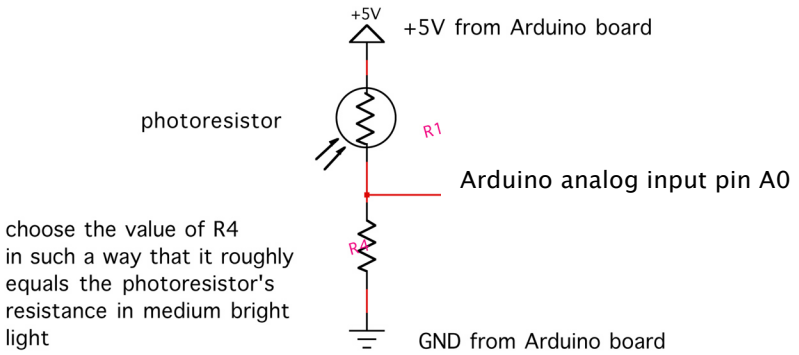


This will open up the serial monitor window which should display numbers from 0 – 1023 for the potentiometer's values (10bit range).

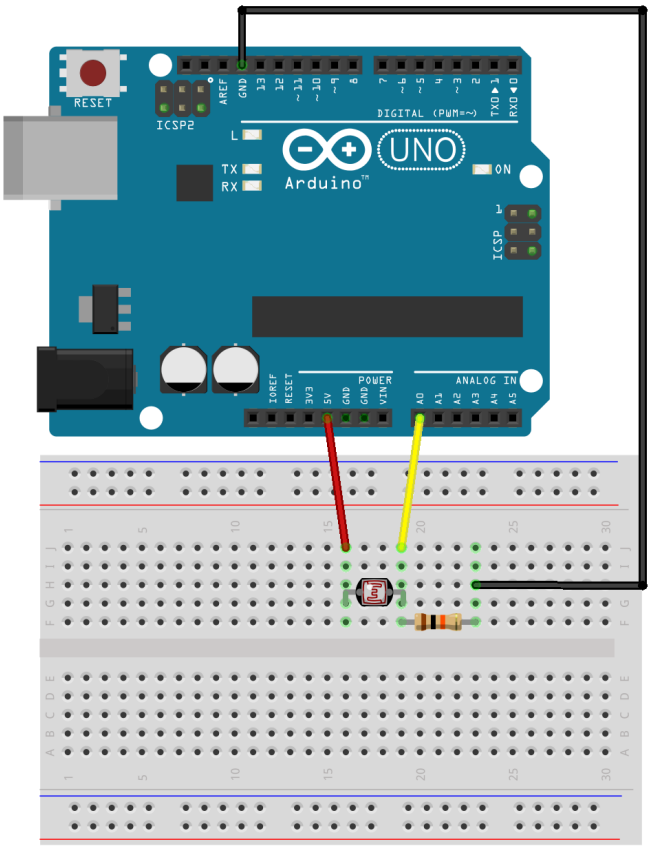
Experiment: Can you setup a circuit that dims an external LED based on the readings from a potentiometer? `File > Examples > 03. Analog > AnalogInOutSerial` *can give you some clues…*

Connecting a Photocell (LDR) to the Arduino Board

This example shows you how to read a photocell (photoresistor or LDR) with the Arduino. The code is analog to the example above.



Here is the breadboard view:



A good starting point for R4 is a resistor value of 10kΩ, if you would like to make the sensor more sensitive in brighter light conditions, reducing R4 to 1kΩ will help. You can always use the serial monitor to check in with the sensor reading and adjust the value of R4 accordingly. If you would like to be flexible, connecting a potentiometer instead of R4 will be helpful since you can change its resistance on the fly.

Experiment: Can you setup a circuit that dims an external LED based on the readings from an LDR similar to a night light, e.g. the LED gets brighter the darker the surrounding light gets? You can use `File > Examples > 03. Analog > AnalogInOutSerial` *as a starting point again and also working with the map function: http://arduino.cc/en/Reference/Map will be helpful.*

More Sensors

Of course there are many more interesting digital and analog sensors than just switches, potentiometers and LDRs. This incomplete list introduces the names of some of them, so you can do some further research and experimentation:

- accelerometer (acceleration and tilt angle measuring)
- PIR sensor (detects presence of people, think automatic house lights)
- IR proximity sensor (short range, up to 3-5 ft.)
- Ultrasonic distance sensor/rangefinder (long distance up to 20 ft.)
- Flex/bend sensor (used in 1990s cybergloves!)
- Heartbeat/pulse sensor
- Temperature sensor (thermistor)
- Touch sensor, capacitive sensor
- Anemometer (wind speed sensor)
- Moisture sensor
- Hall effect sensor (detecting the presence of magnets. Magnetic fields)
- RFID sensor (think anti theft devices)

Sparkfun has a great selection of the above and more sensors for you to explore, check: https://www.sparkfun.com/categories/23 for ideas.

Review and Outlook

Please review these two video about analog sensors and the Arduino board (again from the folks at Fritzing):

- https://www.youtube.com/watch?v=m9J3mDYy0Sg&index=10&list=PL8CD32146ED5CD04E (Episode 009 -- Analog in with Servo and Knob)
- https://www.youtube.com/watch?v=m9J3mDYy0Sg&list=PL8CD32146ED5CD04E&index=10 (Episode 010 – Analog in with Servo and LDR)

These videos already give you a preview of the next workshop, which is about Arduino and output: actuators in which we will take a look at different types of actuators, output amplification and serial control of an iRobot Create robot platform.