AD32600 Physical Computing
Prof. Fabian Winkler
Fall 2014

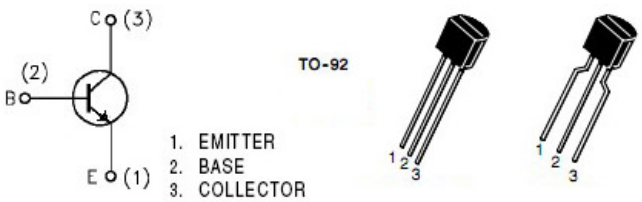
Arduino Workshop 03 – Output: Actuators

This workshop introduces important strategies for the control of actuators (motors, lights, other forms of output) with the Arduino. It is organized in the following sections:

1) Transistors: Output amplification
2) DC motor control
3) Servo motor control
4) Robot control: Arduino and the iRobot Create
    a) Connecting the Arduino to the iRobot Create
    b) Getting sensor values from the robot
    c) Making the robot move: controlling speed with a potentiometer
    d) Putting it all together: simple robot navigation with bump and go

(1) Transistors: Output amplification

Remember the problems we ran into when we tried to drive more than 3 LEDs from one Arduino output pin? Connecting them in series required more than 5V (max. voltage output per Arduino pin) if we wanted to drive more than 3 LEDs and connecting them in parallel required more than 40mA of current (40mA is the maximum current output of an Arduino pin) when trying to connect more than 2 LEDs (review series and parallel circuits in the Arduino workshop 01 if this does not sound familiar to you).
Transistors help us overcome the voltage and current limitations of Arduino pins. Transistors are polarized electronic components that amplify amount of electrons going through a circuit. In a transistor, a weak current (i.e. few electrons) controls a larger current (lots of electrons).



The presence/absence of a current at the transistor's base determines the opening closing of the collector/emitter gate for amplification. Look at the component drawing on the right (above) – this is the pinout for the 2222 general purpose NPN transistor. A weak input current at pin 2 would open the emitter-collector gate and allow a much

larger current to flow from V_{ss} through a load (actuators, such as LEDs connected in series or parallel), the transistor's collector, emitter and finally to ground (GND).

The following picture from my old electronics junior project book illustrates this:

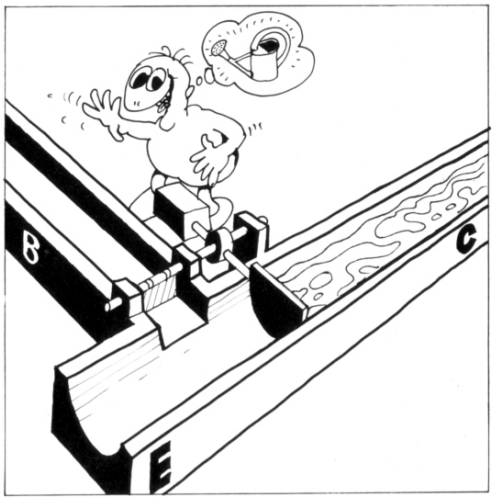
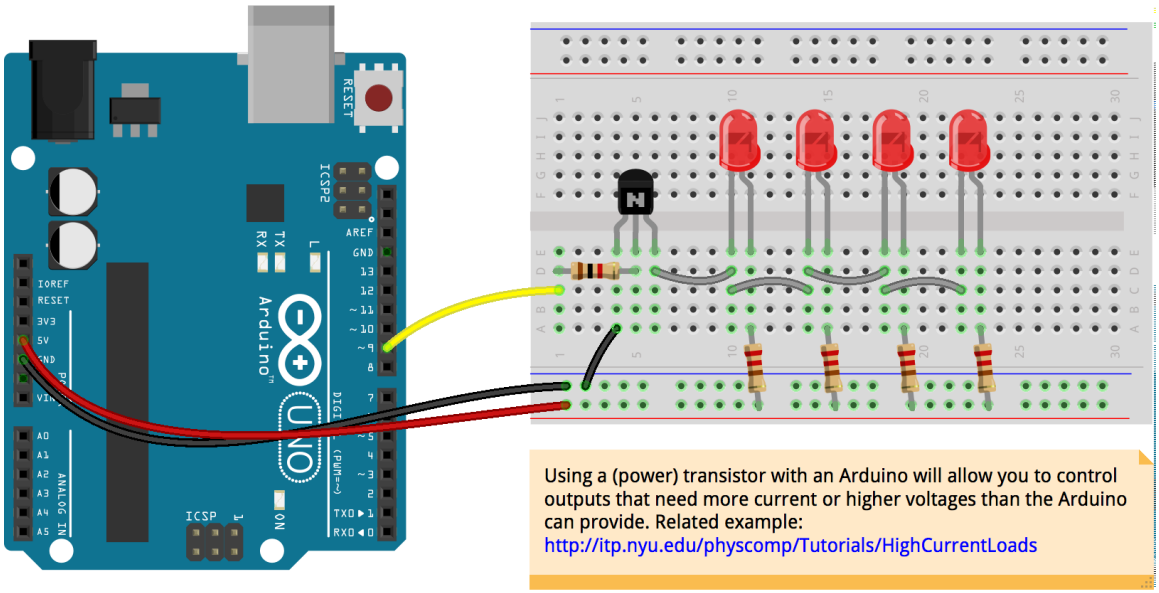


Bild 60. „Wasser-Transistor" (gesperrter Zustand)

Bild 61. „Wasser-Transistor" (leitender Zustand)

Here is a simple circuit that shows how to use a 2222 transistors to control 4 LEDs connected in parallel. Important: The NPN 2222 transistor has a current rating of 200mA. If each LED has an operating current of 20mA (general purpose LED, clear LEDs usually require more current) you can drive a maximum of 10 LEDs in parallel this way. Other transistors, such as the TIP120 that we'll use for the motor control example in section 2) can control much more current than the 2222.

This breadboard view is slightly modified from this Fritzing example:
`File > Open Example > Arduino > Power > NPN Power Transistor`



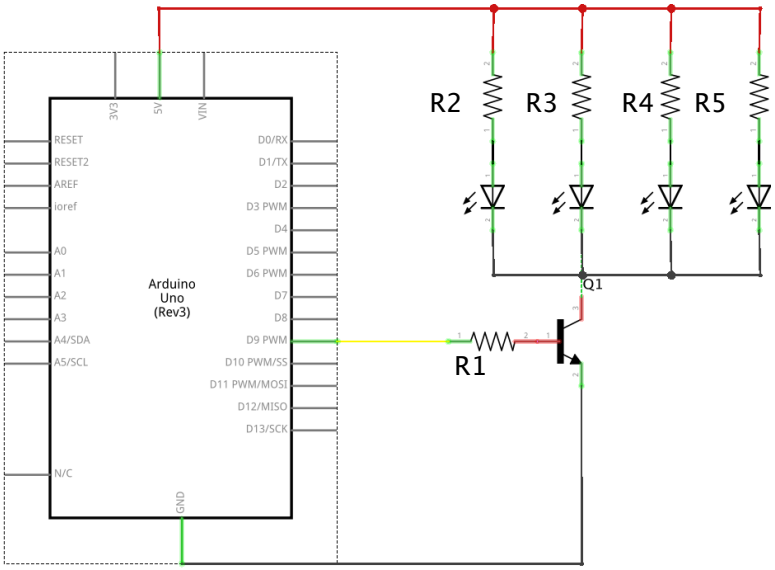
Using a (power) transistor with an Arduino will allow you to control outputs that need more current or higher voltages than the Arduino can provide. Related example:
http://itp.nyu.edu/physcomp/Tutorials/HighCurrentLoads

You can rewrite the blink example (make sure to define pin 9 instead of pin 13 as the

LED output pin) to quickly test your circuit. In the Arduino software:
`File > Examples > 01. Basics > Blink`

Or if you would like to have the LEDs fade:
`File > Examples > 03. Analog > Fading`

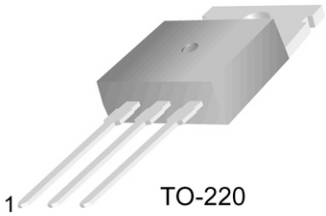
The resistor at the transistor's base (R1) can be any value between 1 – 10kΩ, the resistors for the LEDs (R2 – R5) are 220Ω:



(2) DC motor control

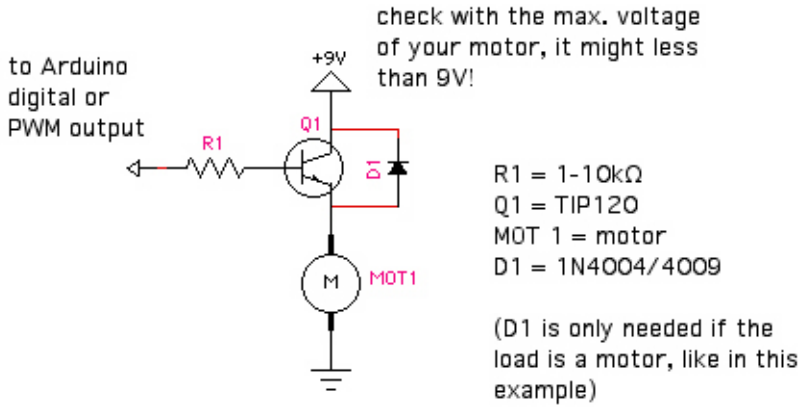
The Arduino's output pins can only supply (sink or source) a very weak current of a maximum of 40mA per pin. This means that you cannot connect a larger load (such as a motor, larger number of LEDs, relay, etc.) directly to the output pin. You need an amplifier that is connected to the pin. For smaller loads a regular NPN2222 switching transistor works (up to 200mA current, see example in section above) for larger loads you should consider a power transistor such as the TIP120 (collector current up to 5A, or pulsed up to 8A)

TIP120/121/122



1.Base 2.Collector 3.Emitter

A very simple Arduino controlled TIP120 circuit would look like this:

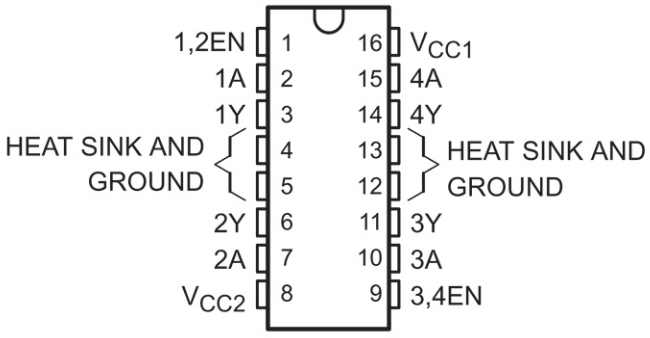


You can get the +9V voltage directly from the Vin pin on your Arduino (as long as you use a (V power supply of course). If you need a +5V voltage, use one of your 7805 voltage regulators to convert the output from your 9V power supply to +5V. Please look at the section "power supply decoupling" on p. 254 in "Physical Computing" if you intend to driver larger loads (multiple LEDs, high brightness LEDs, motors, relays, etc…)

DC motor control with a SN754410 motor driver IC

The SN754410 is a handy IC that allows you to control the speed and direction of a DC motor with only one PWM output and two digital outputs from your Arduino board

SN754410
QUADRUPLE HALF-H DRIVER



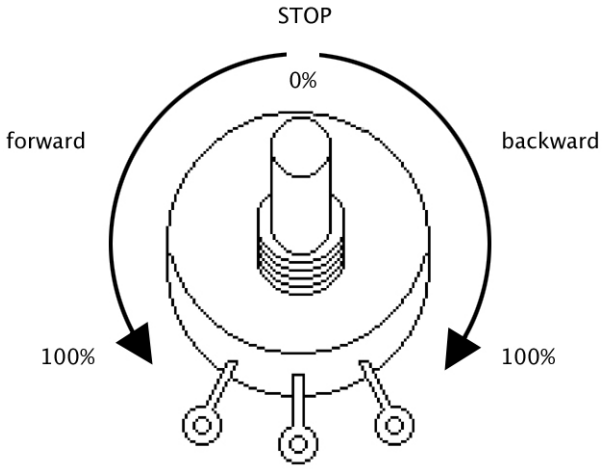
FUNCTION TABLE
(each driver)

INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high-level, L = low-level
X = irrelevant
Z = high-impedance (off)
† In the thermal shutdown mode, the output is in a high-impedance state regardless of the input levels.

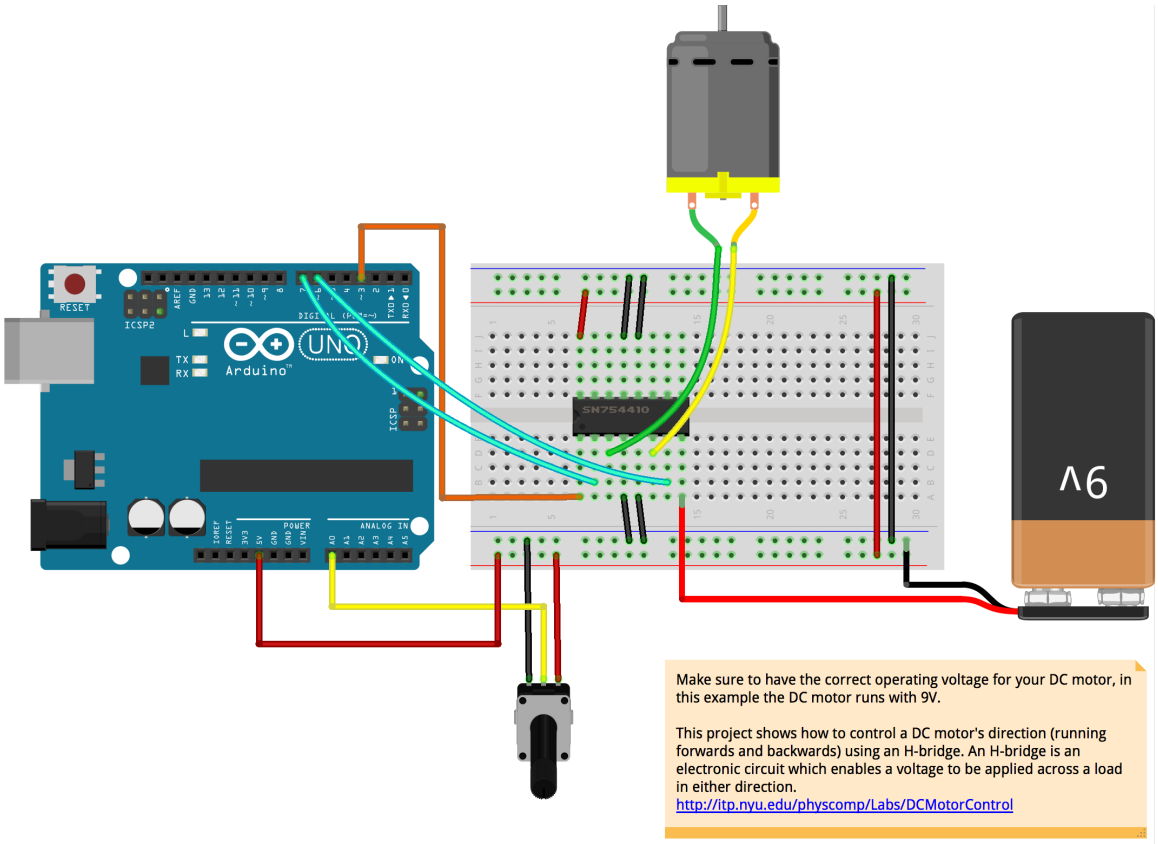
Please read pp.255 -260 in "Physical computing for how to use the SN754410 motor driver IC with a microcontroller.

In the following example we will control a DC motor with a potentiometer. The first half of the 270° degree turning angle of the potentiometer will make the motor run forward. The motor's speed will decrease the more the knob is turned toward the potentiometer's center position. The second half of the turning angle will make the motor run in reverse, with increasing speed toward the 270° mark. The percentages in the following picture are related to the motor's speed.



Here is the Arduino code:
http://web.ics.purdue.edu/~fwinkler/AD32600_F14/Arduino_SN754410.zip

and here is the breadboard view of the circuit:



Regular DC motors vs. gear motors

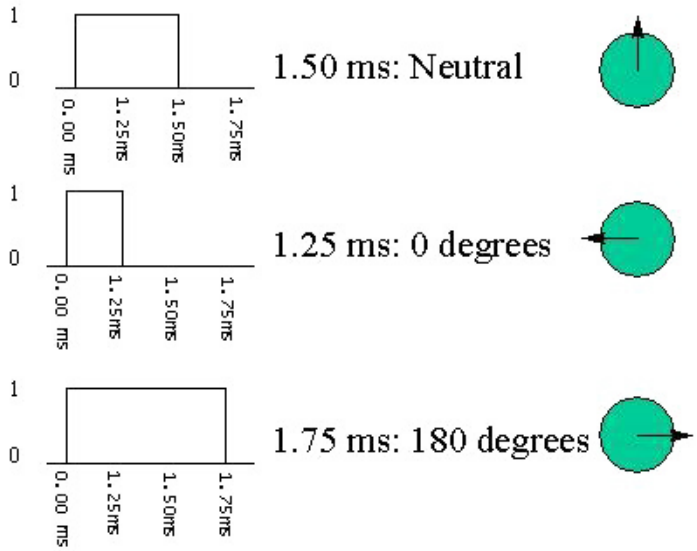
In most of your application when a rotary movement is necessary you will need force (torque) over speed. In this case, use a gear motor instead of a regular DC motor. The gearbox attached to a motors output shaft amplifies its torque and slows down its speed. If you are using a regular DC motor and adjust its speed with the Arduino's PWM output you will just slow down its speed and reduce its torque! You might consider hacking a servo motor for an inexpensive continuously rotating gear motor if you need one and only have a servo available (see: http://www.seattlerobotics.org/guide/servohack.html). Solarbotics has inexpensive gear motors if you need to purchase some genuine gear motors: https://solarbotics.com/catalog/motors-servos/gear-motors/

(3) Servo motor control

Look at pp.121-123 in "Physical Computing" for a description of servo motors. You should also read "What is a servo" from the Seattle Robotics Society, http://www.seattlerobotics.org/guide/servos.html

The following is from the Seattle Robotics Society's intro to servo motors. It describes timing, the most important aspect of controlling a servo motor's 180 degree movement with a microcontroller:

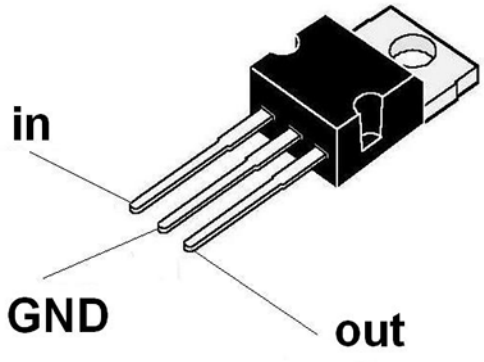
How do you communicate the angle at which the servo should turn? The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation. The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90 degree position (often called the neutral position). If the pulse is shorter than 1.5 ms, then the motor will turn the shaft to closer to 0 degrees. If the pulse is longer than 1.5ms, the shaft turns closer to 180 degrees.



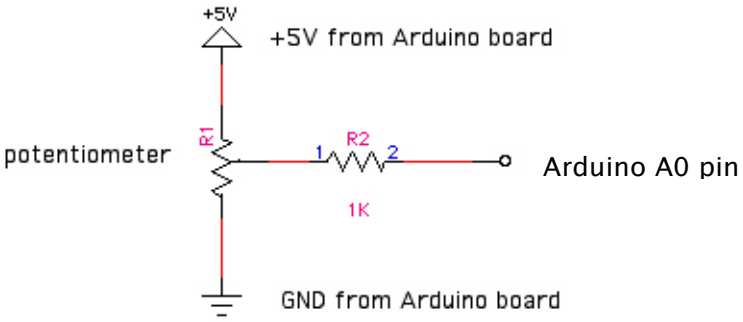
As you can see in the picture, the duration of the pulse dictates the angle of the output shaft (shown as the green circle with the arrow). Note that the times here are illustrative, and the actual timings depend on the motor manufacturer. The principle, however, is the same. You can start with a range from 1000-2000ms and then work your way toward 500ms to 2500ms.

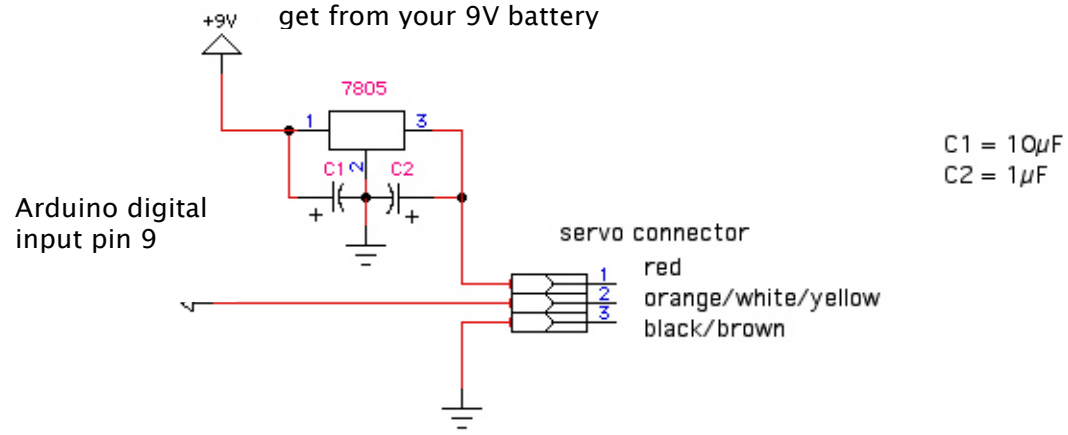
Powering your servo
The servo doesn't need a regulated power supply. You can use any power supply between 4.5V and 6V to power the servo (however, keep in mind that if you power the servo from a power supply of less than +5V, that you will need to put a 1kOhm resistor between the I/O line and the servo's control line. If you plan to use the same power supply for the Arduino and the servo, make sure that it can supply at 1000mA of current – servos can be quite power-demanding! If you are using two different power supplies, one for the Arduino and one for the servo make sure to connect both grounds (GND) together.
Important: do not try to power the servo motor with the Arduino board's USB power, it will draw too much current and might disable your computer's USB port!

Here is the pinout for the 7805 voltage regulator IC:



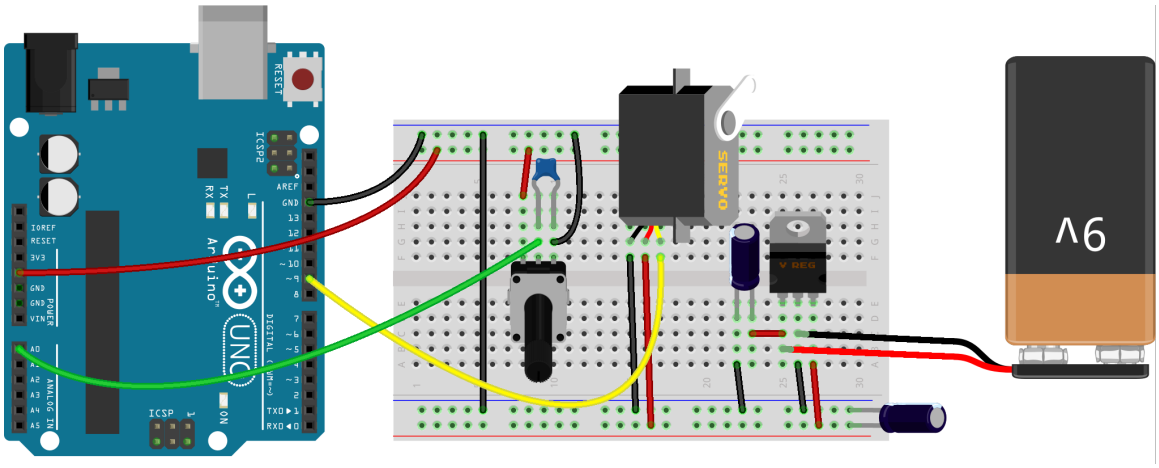
A simple example
In this simple example we connect a potentiometer to one of the Arduino's analog inputs. The turning of the potentiometer's knob should result in a proportional turning of the servos output shaft.





Arduino code: The Arduino code makes use of a built-in library for servo control. In Arduino open: `File > Examples > Servo > Knob`

Here is the breadboard view:



If you notice an unsteady or jerky behavior of the servo motor's output, put a 0.1µF capacitor between the Arduino's analog pin A0 and GND. This will smooth out any possible voltage spikes from the potentiometer.

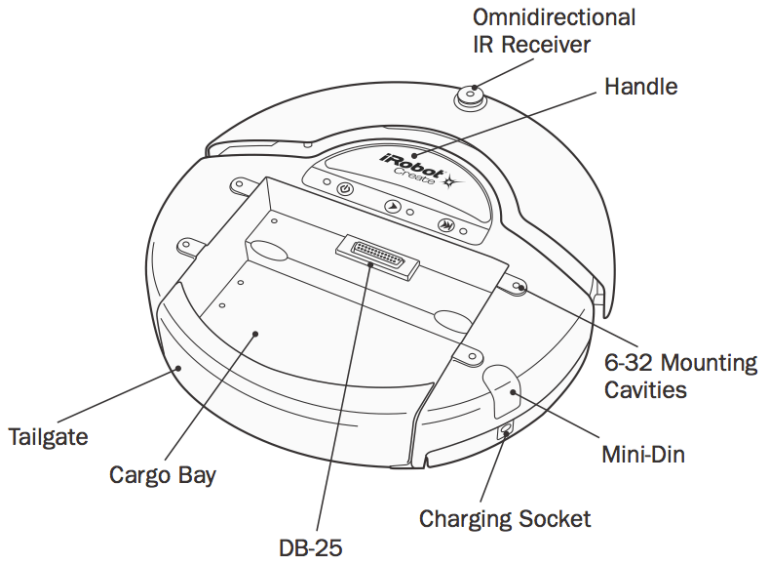
(4) Robot control: Arduino and iRobot Create

The iRobot Create (http://store.irobot.com/education-research-robots/shop.jsp?categoryId=3311368) is an inexpensive educational robot platform that can be controlled with an Arduino board using serial commands. These commands are based on the iRobot Create's Open Interface (OI) protocol, a documentation of which can be found here:
http://www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Open%20Interface_v2.pdf

This section is broken up into the following smaller steps:

e) Connecting the Arduino board to the iRobot Create
f) Getting sensor values from the robot
g) Making the robot move: controlling speed with a potentiometer
h) Putting it all together: simple robot navigation with bump and go

Let's start by first taking a look at the anatomy of the iRobot Create:



More information about the iRobot Create can be found in the robot's manual:
http://www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Manual_Final.pdf

Sensors
The iRobot Create comes with some basic sensors mounted on its platform, they are:

- 2x bump sensors (left, right)
- 4x cliff sensors (left, front left, right, front right)
- 3x wheel drop sensors (left, right, caster (center))
- 2x buttons (labeled advance and play)
- 1x infrared sensor (receiving bytes sent from Roomba remote, home base or user created devices)
- 1x wall sensor (used with the virtual wall accessory: wall detection/signal strength)

Furthermore the iRobot Create allows access to sensor monitoring some of its internal functions:

- distance the robot has traveled since the distance was last requested (in millimeters)
- angle the robot has turned since the last time the angle was requested (in degrees)
- charging state
- battery voltage (in mV)
- current (flowing into the robot's battery, in mA)
- battery temperature (in degrees Celsius)
- battery charge (in mAh)
- battery capacity (in mAh)
- velocity information (left, right, in mm/s)

Actuators

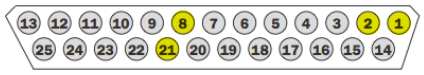
The iRobot Create Open interface allows access to the following on-board actuators

- wheel motors (speed left, speed right)
- LEDs (play, advance, bi-color power LED)
- speaker (define musical notes)

Using the Arduino board as a controller, other sensors and actuators can be added easily to the Arduino board's I/O pins (e.g. photocells, IR proximity sensors, ultrasonic range finders, servo motors, etc.)

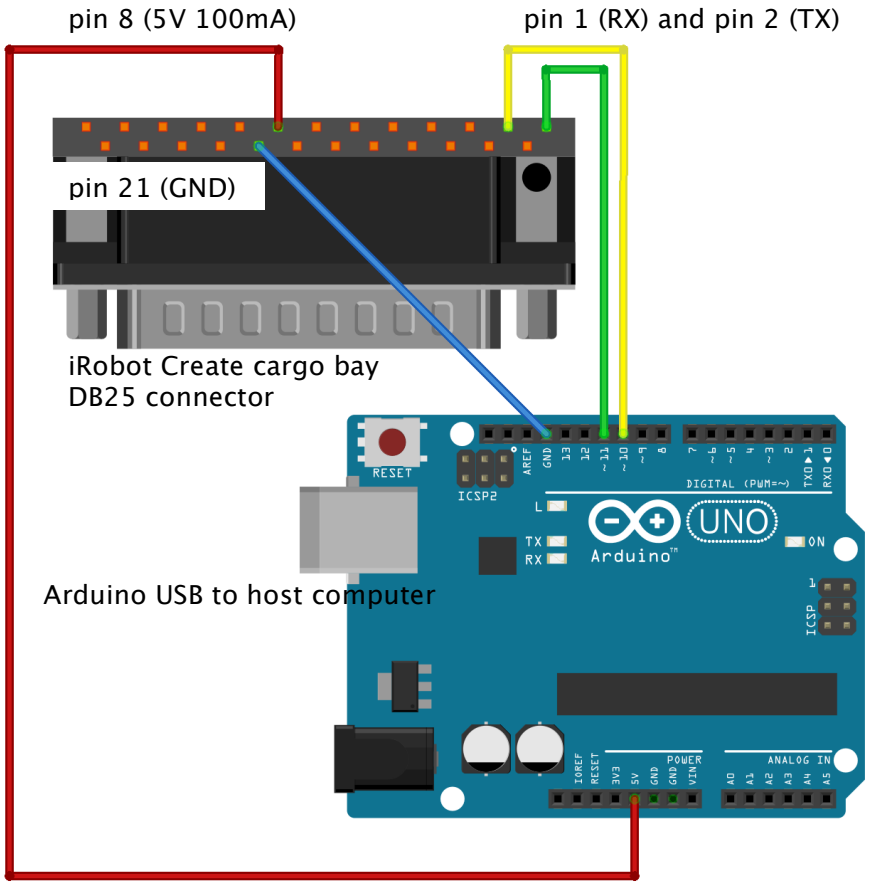
a) Connecting the Arduino board

Connecting the Arduino board is easy through the cargo bay DB25 connector, in the diagram below we see the necessary pins:



Pin	Name	Description
1	RXD	0 – 5V Serial input to Create
2	TXD	0 – 5V Serial output from Create
3	Power control toggle	Turns Create on or off on a low-to-high transition
4	Analog input	0 - 5V analog input to Create
5	Digital input 1	0 - 5V digital input to Create
6	Digital input 3	0 - 5V digital input to Create
7	Digital output 1	0 - 5V, 20 mA digital output from Create
8	Switched 5V	Provides a regulated 5V 100 mA supply and analog reference voltage when Create is switched on
9	Vpwr	Create battery voltage (unregulated), 0.5A
10	Switched Vpwr	Provides battery power @ 1.5 A when Create is powered on.
11	Switched Vpwr	Provides battery power @ 1.5 A when Create is powered on.
12	Switched Vpwr	Provides battery power @ 1.5 A when Create is powered on.
13	Robot charging	When Create is charging, this pin is high (5V)
14	GND	Create battery ground
15	Device Detect/Baud Rate Change Pin	0-5V digital input to Create which can also be used to change the baud rate to 19200 (see below)
16	GND	Create battery ground
17	Digital input 0	0 - 5V digital input to Create
18	Digital input 2	0 - 5V digital input to Create
19	Digital output 0	0 - 5V, 20 mA digital output from Create
20	Digital output 2	0 - 5V, 20 mA digital output from Create
21	GND	Create battery ground
22	Low side driver 0	0.5A low side driver from Create
23	Low side driver 1	0.5A low side driver from Create
24	Low side driver 2	1.5A low side driver from Create
25	GND	Create battery ground

Connecting an Arduino UNO board to the iRobot Create:



b) Reading basic sensor values

We are making use of Arduino's softSerial library, so we can work with two independent serial ports on one Arduino board. The reason for this is simple: in order to see if the Arduino can read OI sensor values we need to be able to send requests to OI that return values (serial call and response) and at the same time communicate these readings back to the Arduino serial monitor so we can see what values we get. In a later step we can then trigger driving commands based on the sensor readings. For now however we are just interested in reading sensor values.

Why do we need two serial ports for this?
It's simple – we do not want to get our serial values/commands mixed up – OI commands that we send to the iRobot Create should not show up in the serial monitor and values we send back to the serial monitor should not be interpreted as OI commands.

Why don't we connect the Arduino UNO's hardware serial port (RX/TX)?

> "The serial port output TXD from the Roomba/Create is too weak to drive the RX serial port (hardware serial port) input of an Arduino properly. This is because of the USB-Serial converter on the Arduino: it also tries to drive the RX serial port input via a pullup resistor, but the Roomba does not have enough drive to pull the RX down below about 2.5 volts, which is insufficient to be reliably detected as a TTL serial input of 0."

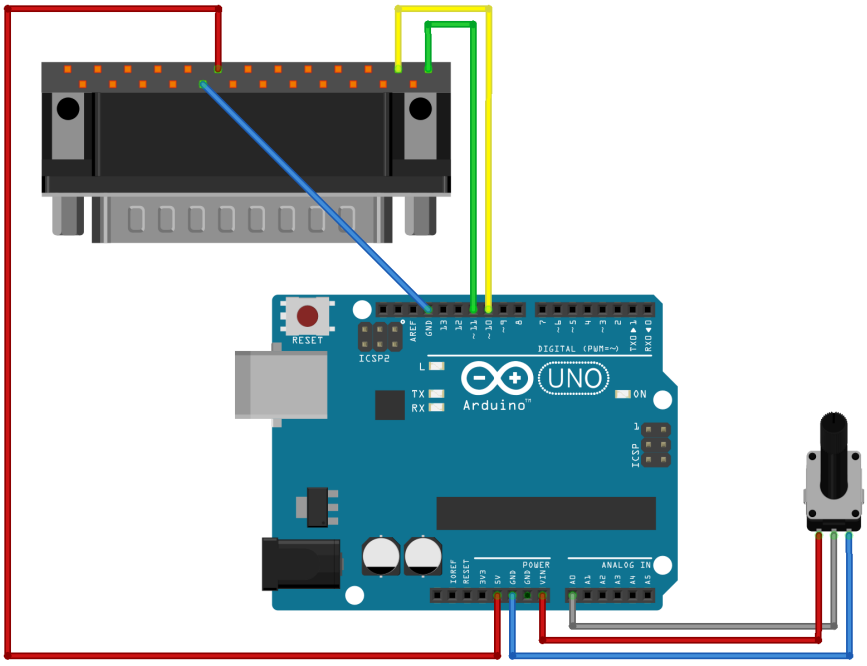
(from: http://projectsfromtech.blogspot.com/2013/11/irobot-create-arduino-control.html).

Notes on the Arduino Mega: the Arduino Mega has 4 serial ports, use **Serial1**(TX1/RX1), **Serial2**(TX2/RX2) or **Serial13**(TX3/RX3) instead of **Serial0**. You can still use Serial0 to communicate back to a serial monitor via USB or XBee for debugging if needed.

Here is the Arduino code to read the Create's bump Sensors:
http://web.ics.purdue.edu/~fwinkler/AD32600_F14/bump.zip

c) Making the robot move
In this section we develop Arduino code that makes the robot move, it is based on the drive serial sequence documented in the Open Interface manual on page 9:
`[137] [Velocity high byte] [Velocity low byte] [Radius high byte] [Radius low byte]`

We would like to connect a potentiometer to Arduino's analog pin A0 for speed control and map the potentiometer's reading like this: the robot should be at speed 0 when the potentiometer is in the middle position, it should be at the maximum speed backwards when we turn the potentiometer all the way to the left and it should be at its maximum forward speed when the potentiometer is turned all the way to the right. The following drawing shows the hardware setup for this example and the full Arduino code example can be downloaded here:
http://web.ics.purdue.edu/~fwinkler/AD32600_F14/create_go.zip.



d) Putting it all together: simple robot navigation with bump and go
Based on the function for moving the robot forward and backward and the documentation from the Create's Open Interface manual we can think of further helpful functions that can become part of almost all code examples that make the robot move:

(1) driveStraight (speed)
(2) drive(speed, radius)

(3) stop()
(4) spinLeft(speed)
(5) spinRight(speed)

The final code example has all of these functions implemented and makes use of them in the bump and go behavior:
http://web.ics.purdue.edu/~fwinkler/AD32600_F14/bump_and_go.zip

Outlook
This is just the beginning of what you can do with Arduino and the iRobot Create platform. Feel free to combine Arduino examples from previous and future workshops with the iRobot Create to develop exciting robot-based applications. For example, you can think about drawing a line in Processing and having the iRobot Create execute this line by driving along it in the real world. In the next workshop we will take a closer look at how to integrate Processing and Arduino, so that something like the idea mentioned above could be soon realized by you!

Further reading:
Kurt, Tod E. *Hacking Roomba*. Indianapolis, IN: Wiley,2007.