



Introduction to the Arduino and physical computing

Class 2

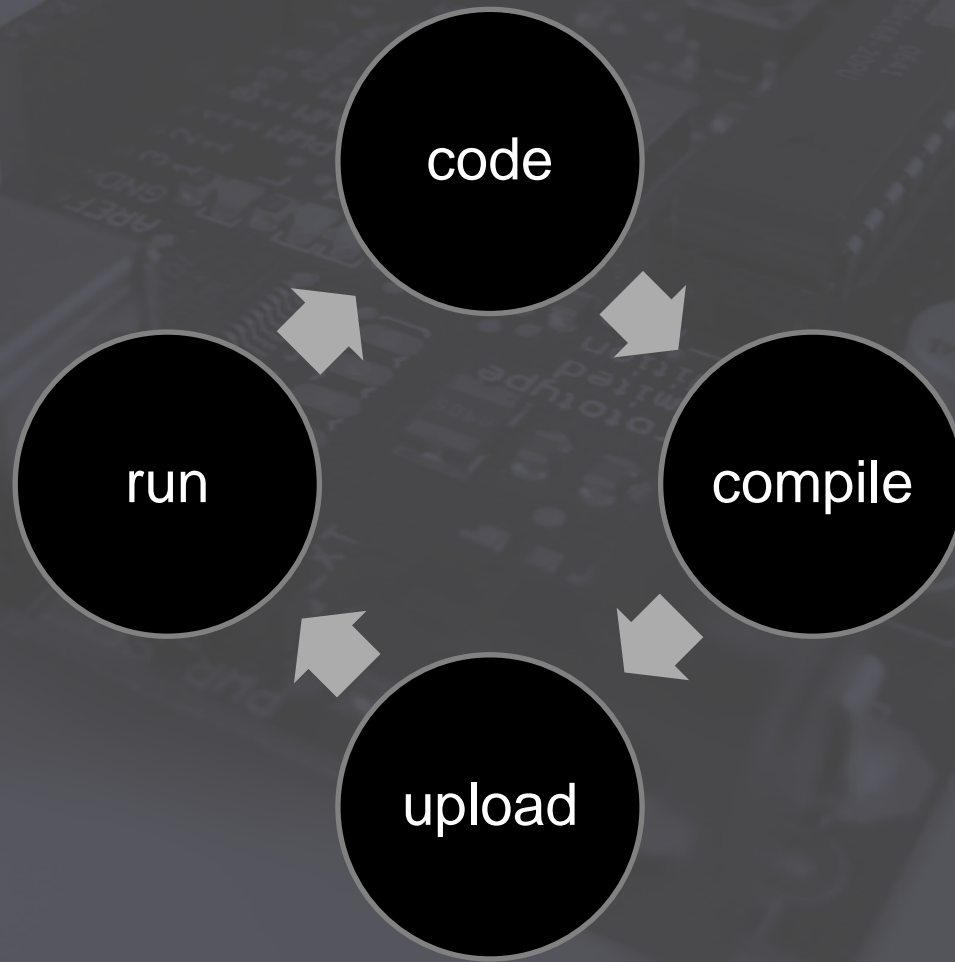
With Josh Kopel

Arduino IDE



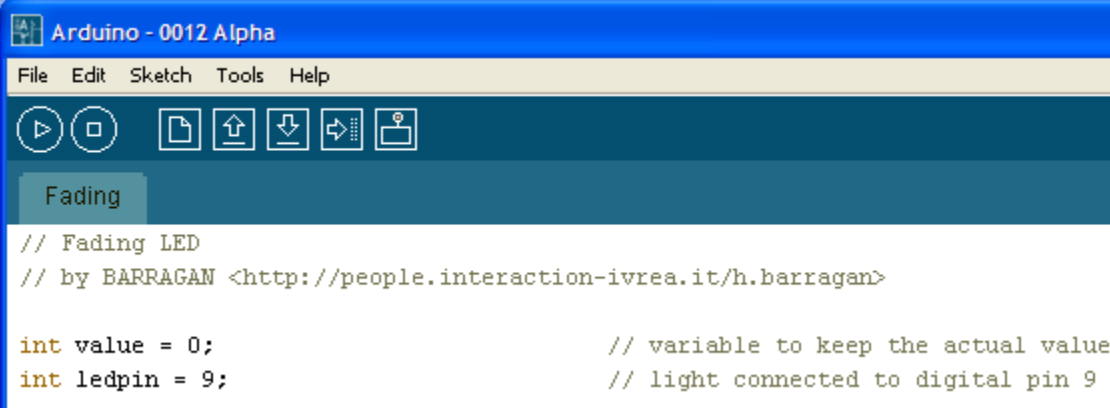
- Open source IDE
- Automates avr-g++ tool chain
- Subset of C++
- Simplified syntax
- Full language reference included in IDE

Arduino development cycle



Arduino programming

- Declare variables, defines, and includes at top



```
Arduino - 0012 Alpha
File Edit Sketch Tools Help
Fading
// Fading LED
// by BARRAGAN <http://people.interaction-ivrea.it/h.barragan>

int value = 0; // variable to keep the actual value
int ledpin = 9; // light connected to digital pin 9
```

Arduino programming

- `setup()` – run once at beginning, set pins, open communications

```
void setup() {
  // set the switch as an input:
  pinMode(switchPin, INPUT);

  // set all the other pins you're using as
  pinMode(motor1Pin, OUTPUT);
  pinMode(motor2Pin, OUTPUT);
  pinMode(speedPin, OUTPUT);
  pinMode(ledPin, OUTPUT);

  // set speedPin high so that motor can tu
  digitalWrite(speedPin, HIGH);

  // blink the LED 3 times. This should hap
  // if you see the LED blink three times,
  // reset itself,. probably because the mo
  // or a short.
  blink(ledPin, 3, 100);
}
```

Arduino programming

- loop() – run repeatedly, after setup()

```
void loop() {  
  // if the switch is high, motor will turn on one direction:  
  if (digitalRead(switchPin) == HIGH) {  
    digitalWrite(motor1Pin, LOW); // set leg 1 of the H-bridge low  
    digitalWrite(motor2Pin, HIGH); // set leg 2 of the H-bridge high  
  }  
  // if the switch is low, motor will turn in the other direction:  
  else {  
    digitalWrite(motor1Pin, HIGH); // set leg 1 of the H-bridge high  
    digitalWrite(motor2Pin, LOW); // set leg 2 of the H-bridge low  
  }  
}
```

Arduino programming

- User defined functions

```
/*  
  blinks an LED  
*/  
void blink(int whatPin, int howManyTimes, int milliSecs) {  
  int i = 0;  
  for ( i = 0; i < howManyTimes; i++) {  
    digitalWrite(whatPin, HIGH);  
    delay(milliSecs/2);  
    digitalWrite(whatPin, LOW);  
    delay(milliSecs/2);  
  }  
}
```

Common constructs

- ```
if (i < 10) { // statement(s) }
else if(i > 10){ // statement(s) }
else { // statement(s) }
```
- ```
for ( i = 0; i < 10; i++) { // statement(s) }
```
- ```
while(expression){ // statement(s) }
```

# Unique to Arduino

- `pinMode()` – set a pin as input or output
  - `digitalWrite()` – set a digital pin high/low
  - `digitalRead()` – read a digital pin's state
  - `analogRead()` – read an analog pin
  - `analogWrite()` – write an “analog” value
  - `delay()` – wait an amount of time
- (And others)

# Simple program

```
// Fading LED
int value = 0; // variable to keep the actual value
int ledpin = 9; // light connected to digital pin 9

void setup()
{ // nothing for setup }

void loop() {
 for(value = 0 ; value <= 255; value+=5) // fade in (from min to max) {
 analogWrite(ledpin, value); // sets the value (range from 0 to 255)
 delay(30); // waits for 30 milli seconds to see the dimming effect
 }
 for(value = 255; value >=0; value-=5) // fade out (from max to min) {
 analogWrite(ledpin, value);
 delay(30);
 }
}
```

# Physical computing “patterns”



- Wait for input
- Check limits
- Send/receive
- React using feedback
- Blind output

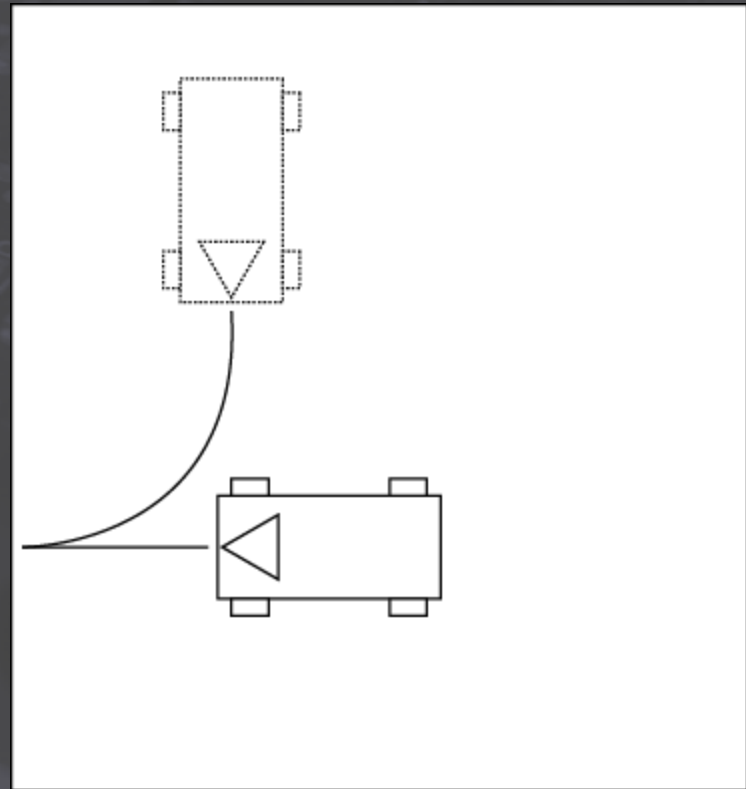
# What is going on out there?

- Polling – test a sensor (or sensors) inside a loop. Wait for a value or state change.
- Interrupts – “attached” to a pin or timer. Code runs when a pin changes state or timer overflows
  - External
  - Internal (timer)

# Real world polling

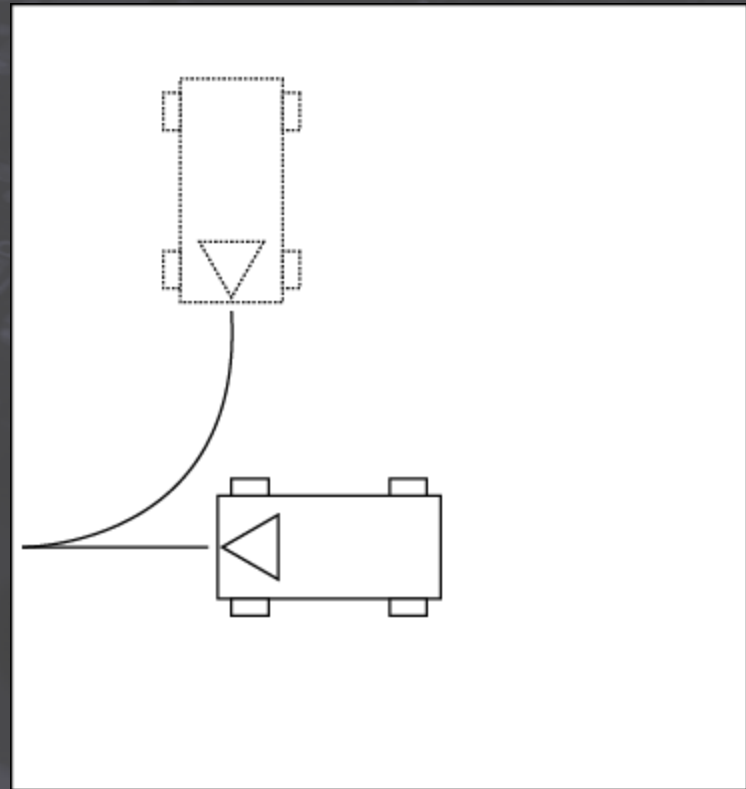
## Obstacle avoidance

- Move until hit
- Backup and turn
- Move until hit
- Repeat...



# Pseudo-code

```
If(no obstacle){
 go forward
}else{
 turn wheels
 for(i = 0; i < 100; i++){
 backup
 }
 straighten wheels
}
```



# Polling

```
int inputPin = 2; // choose the input pin (for a pushbutton)
void setup() {
 pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop(){
 val = digitalRead(inputPin); // read input value
 if (val == HIGH) {
 // button is pressed, do something
 }else{
 //not pressed, do something else
 }
}
```

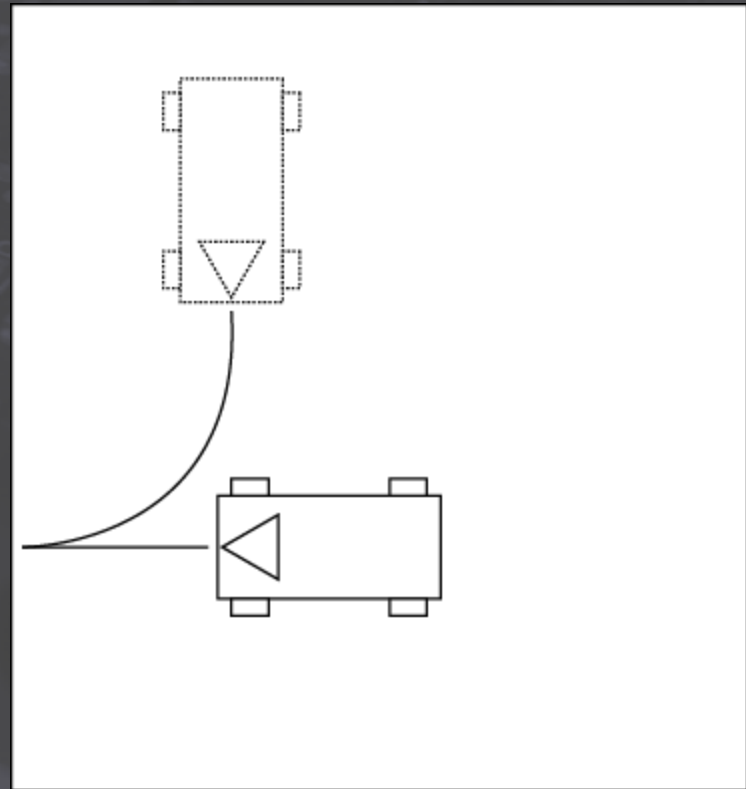
# External interrupt

- Attach interrupt function to one of the available pins
- Digital pin **2** = interrupt **0**
- Digital pin **3** = interrupt **1**
- Set the type of transition
  - **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.

# Pseudo-code

```
loop(){
 go forward
}

interrupt(){
 turn wheels
 for(i = 0; i < 100; i++){
 backup
 }
 straighten wheels
}
```



# Attaching interrupt

```
int inputPin = 2; // choose the interrupt pin (must be 2 or 3)
void setup() {
 pinMode(inputPin, INPUT); // declare input
 attachInterrupt(0, backup, HIGH); // attach interrupt 0 to function
}
void loop(){ Go forward }

void backup(){
 Turn wheels
 Backup
 Straighten wheels
}
```

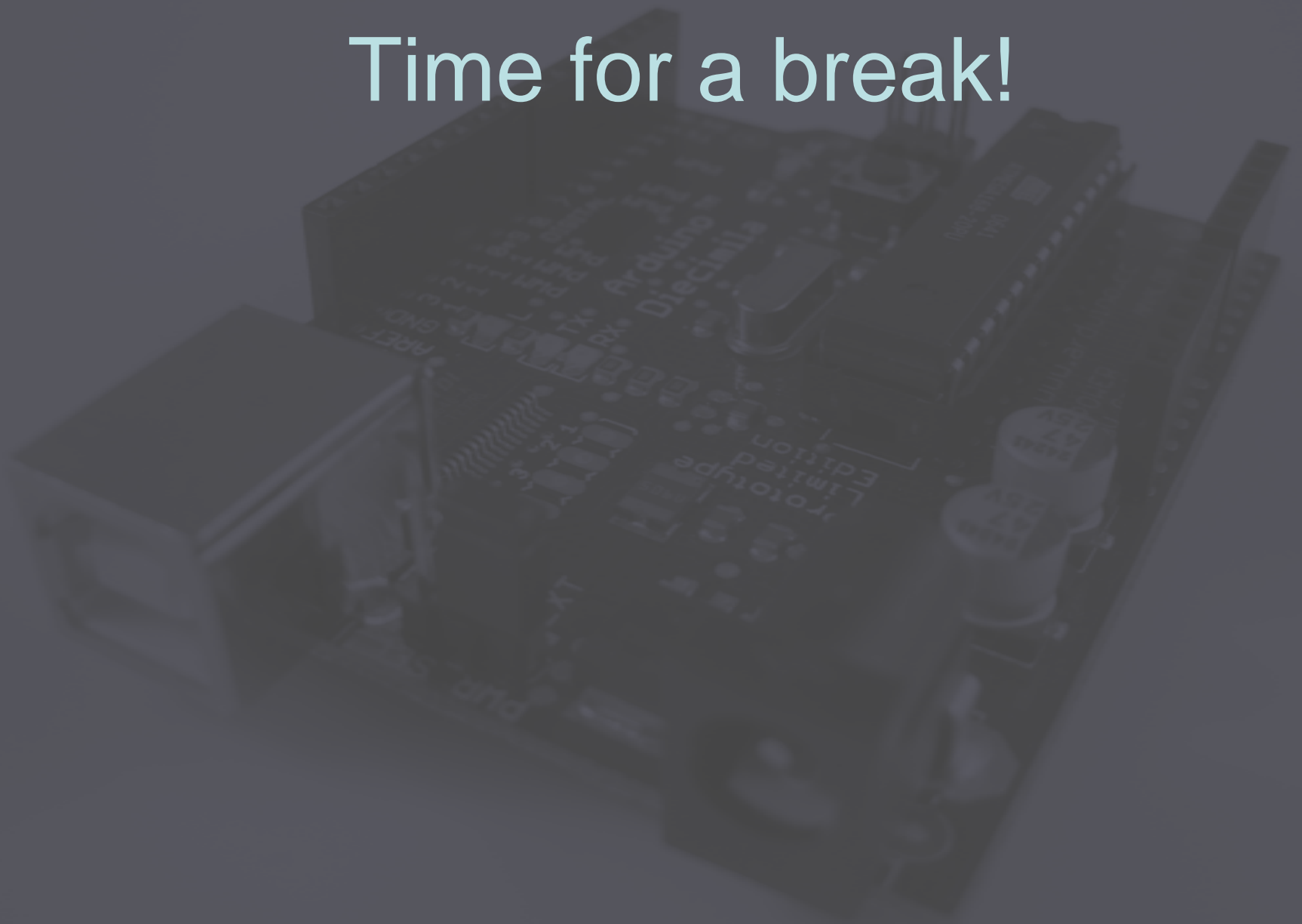
# Timer Interrupt

- Make something happen at regular intervals
- atMega168/328 has several internal timers
- All are used by various Arduino functions
  - **Timer0** (System timing, PWM 5 and 6)
  - **Timer1** (PWM 9 and 10)
  - **Timer2** (PWM 3 and 11)

# Timer Interrupt

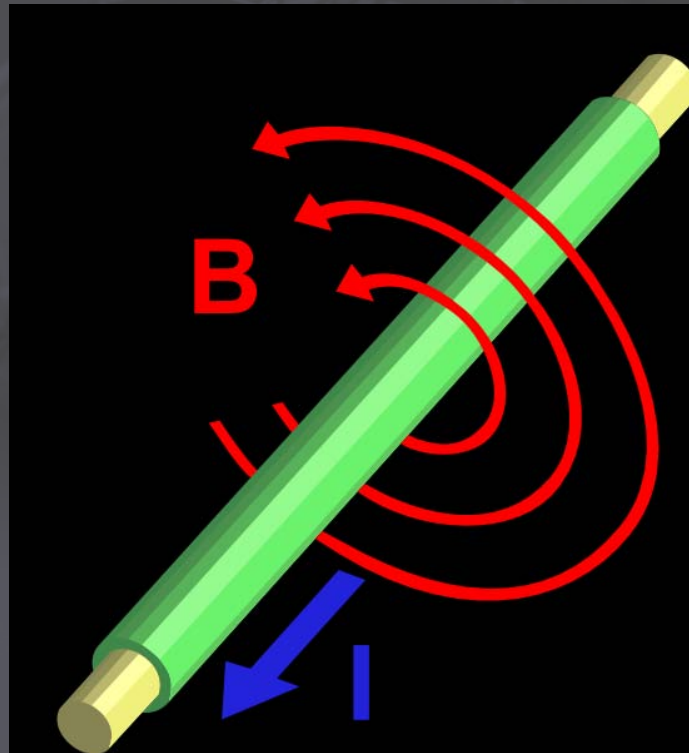
- Enable an interrupt on a timer
- Set the timer going and when it exceeds a pre-set value (overflows) it runs a function
- Actual code too complicated for this venue
- Lots of online resources...

Time for a break!



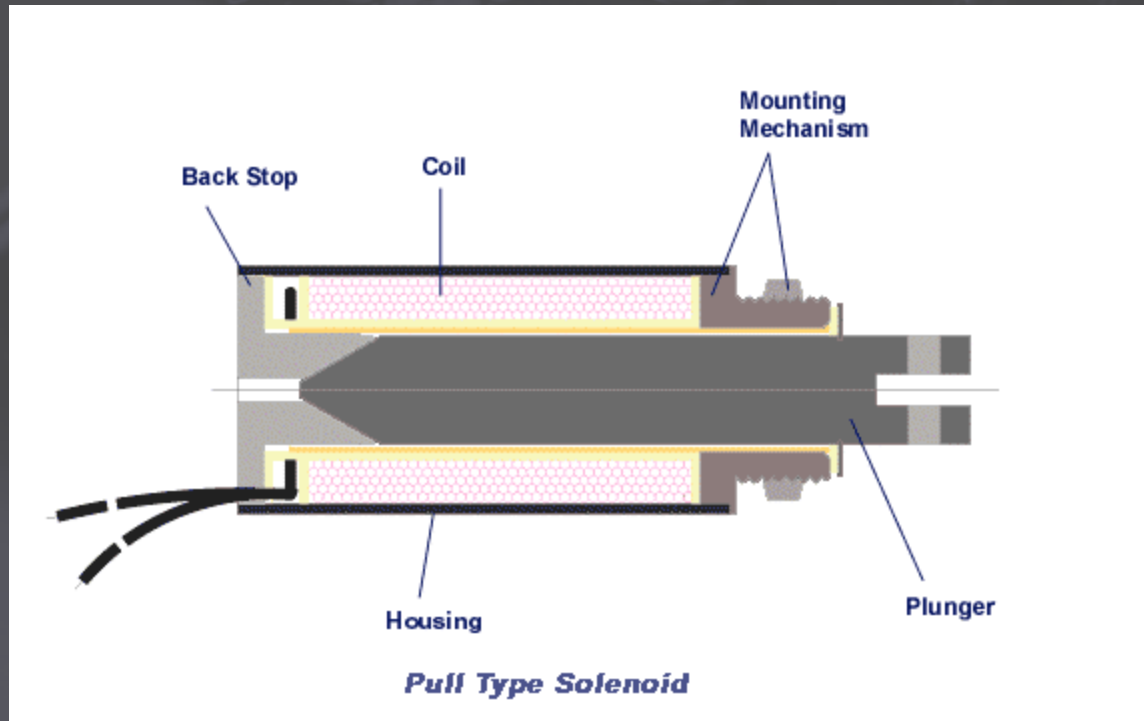
# Making things move

- Conductors generate a magnetic field when a current flows through them



# Simple linear motion

- Solenoid moves a metal plunger in a coil



# Motors!

- DC motor and Gear motor
- “Hobby” Servo
- Stepping motor



# DC Motor

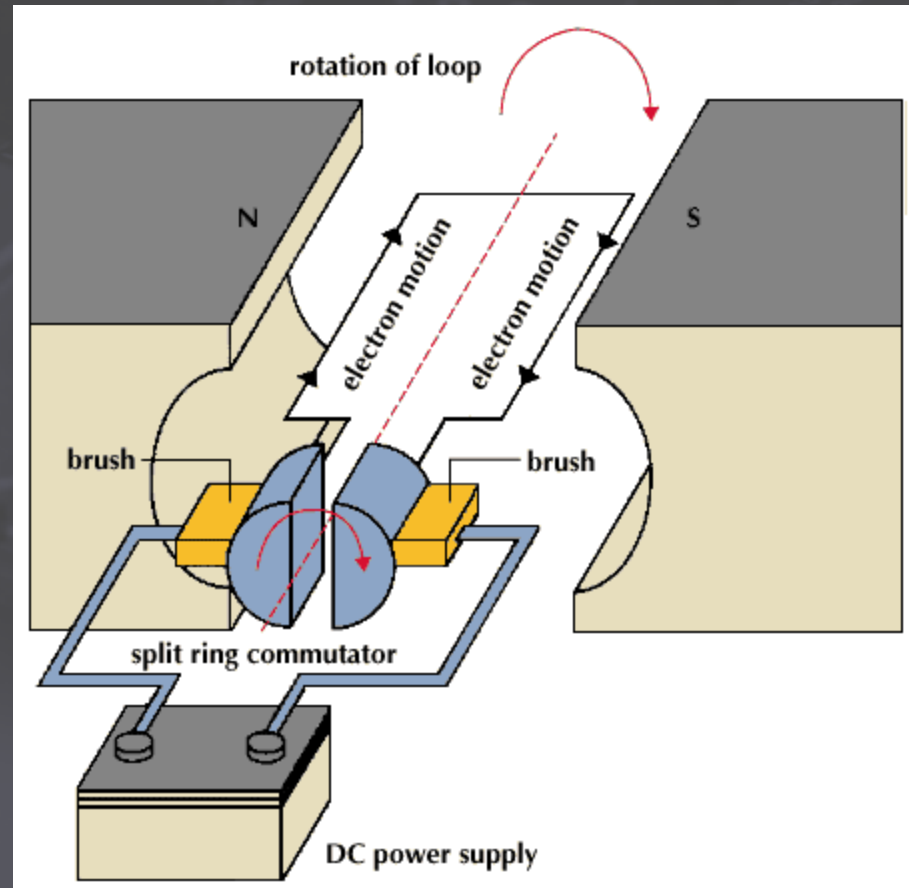
## Pros

- Simple & cheap (only 2 wires)
- Fast & be geared down for higher torque
- Easy to reverse
- Speed control through PWM

## Cons

- No built-in feedback

# DC Motor



# Hobby Servo

(DC motor with feedback)

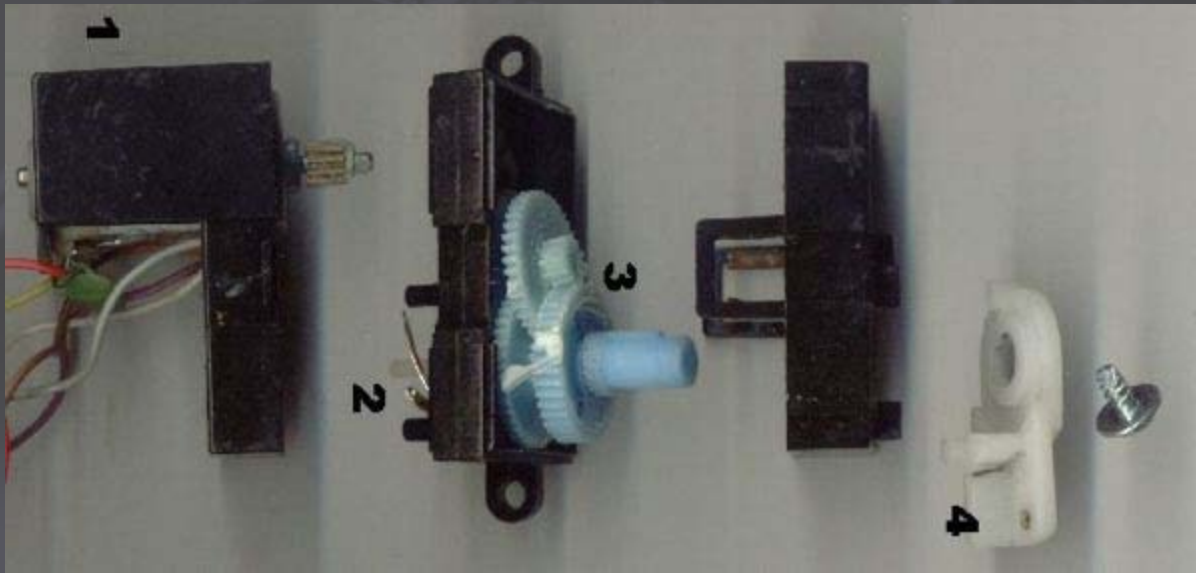
## Pros

- Simple (3 wire) and “Standard”
- accurate even under load
- High torque

## Cons

- Slower (geared)
- More expensive (rare as junk)
- Must be modified for 360 deg. rotation

# Hobby Servo



# Stepping motor

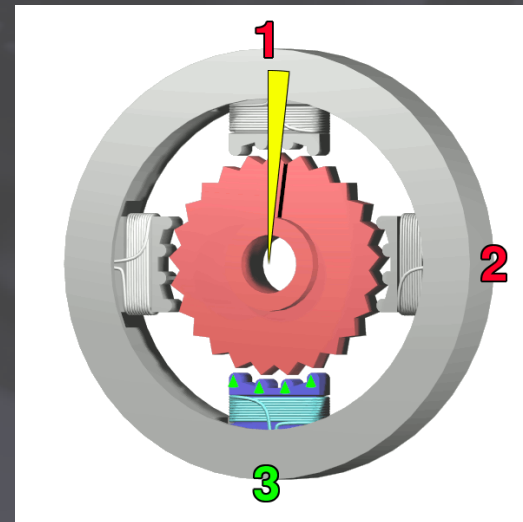
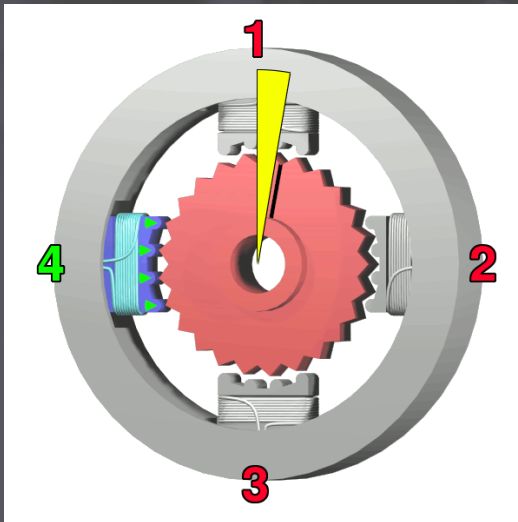
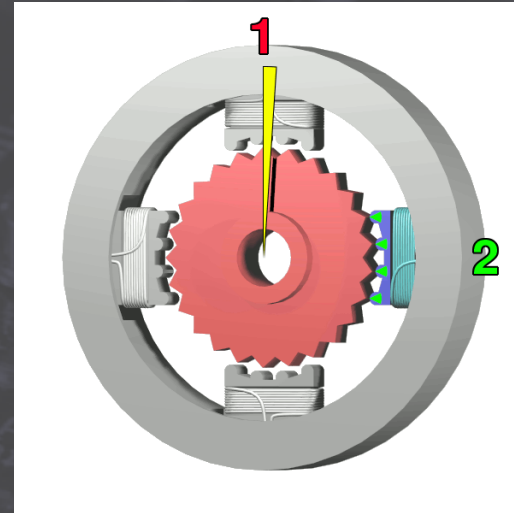
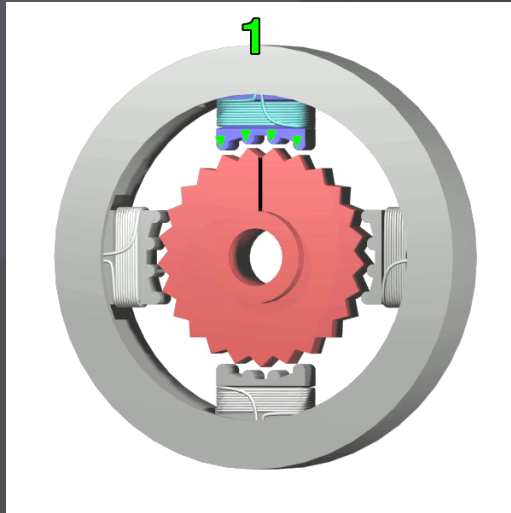
## Pros

- Very accurate positioning
- Speed, step size, and direction control
- (Can be) High torque and position hold

## Cons

- Slower & can skip under load
- More complicated & expensive
- Needs more pins

# Stepping motor

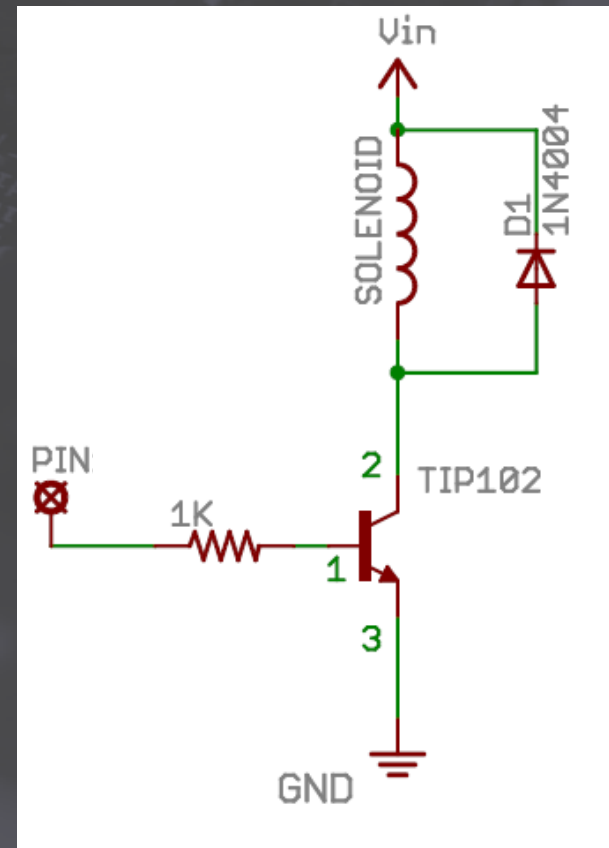


# Driving a motor

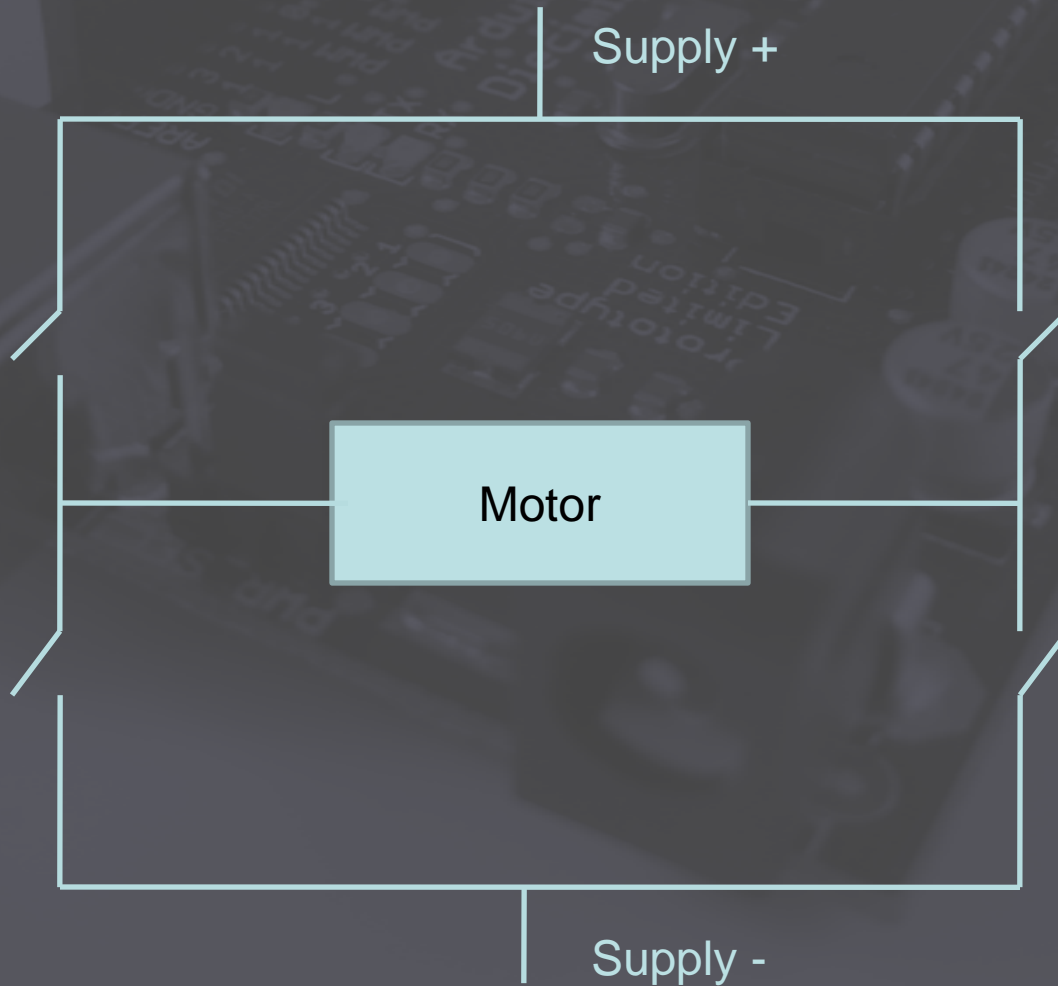
- Motors are inductive loads and require a high current
- Very high current pulse on start up or stall
- Must account for “back EMF” when switched off

# Simple driver

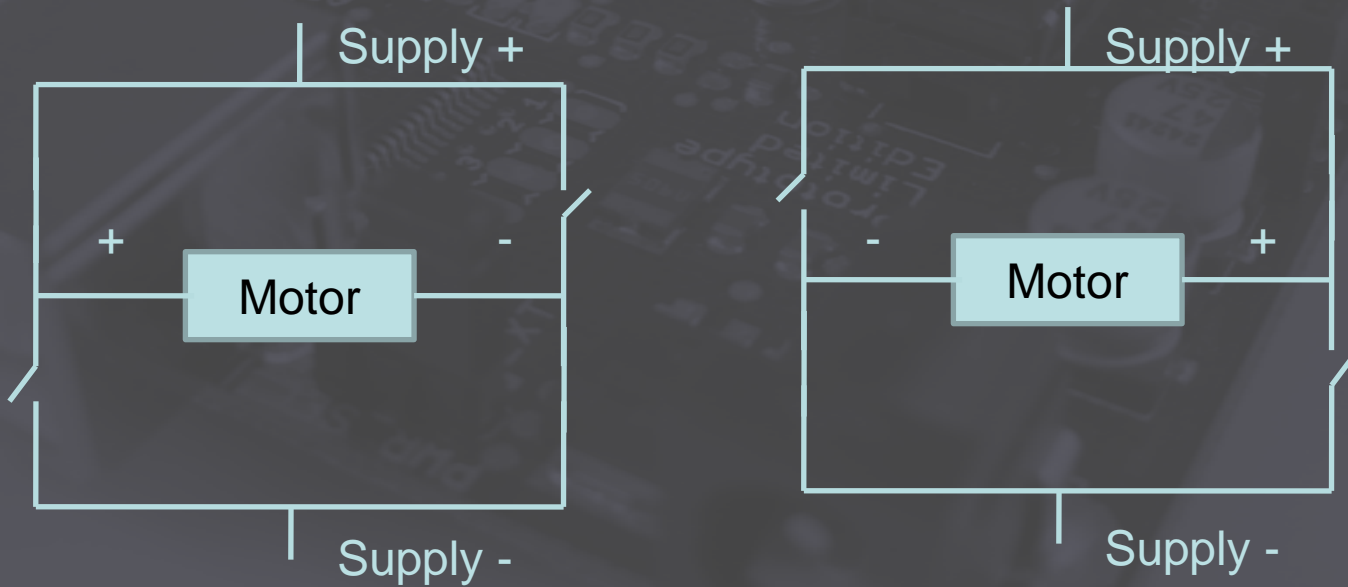
- Use a transistor to switch the high-current load
- $V_{in}$  can be any voltage up to about 24V
- Total power about 24W (24V@1A, 12V@2A, etc.)
- Will get HOT (use a heat sink)



# H-Bridge



# Direction control



A grayscale, slightly blurred image of an Arduino Uno R3 board. The board is oriented vertically, showing the USB Type-C port on the left, the ATmega328P microcontroller in the center, and the 5V and GND pins on the right. The text "Play time!" is overlaid in a white, sans-serif font in the center of the image.

Play time!