

Physical Computing

DC Motor, Servo Motor & Communication with Computer

Kening Zhu (Ken)
Assistant Professor
School of Creative Media
City University of Hong Kong
Email: keninzhu@cityu.edu.hk

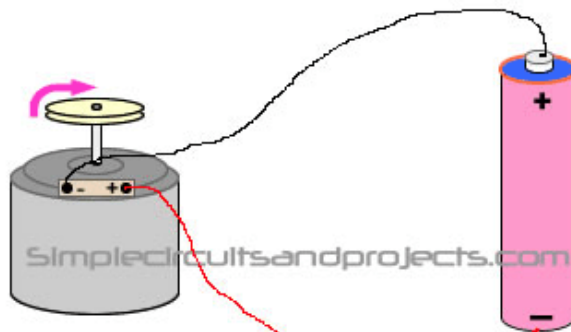


DC Motor

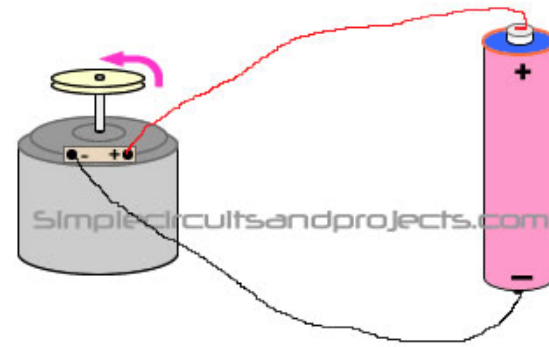


How to control a DC motor (Direction & Speed)

- “DC” means Direct Current (Single Direction), which can be generated by a normal battery.



(a) Polarity reversal test of DC motor



(b) Polarity reversal test of DC motor

- The higher the current output by the battery, the faster the motor will be.

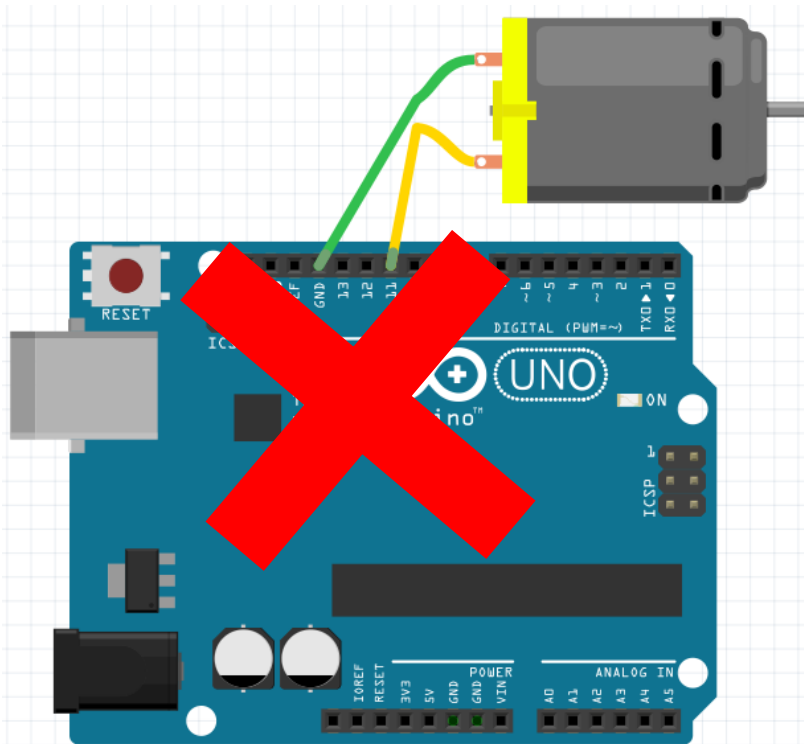
Current \uparrow Speed \uparrow

Then can we use the analog output from Arduino to control the motor?

Yes! But...

- Can we connect a motor directly to Arduino just like a LED?

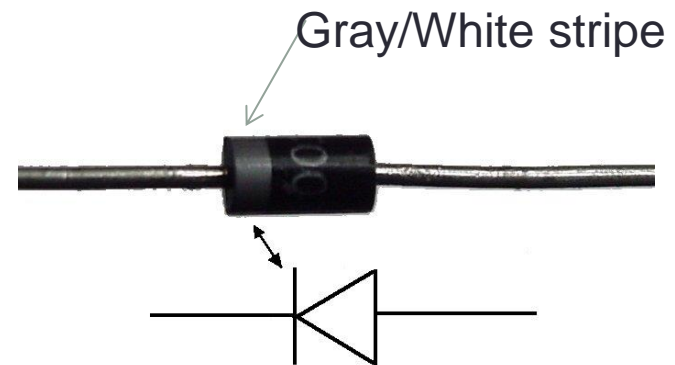
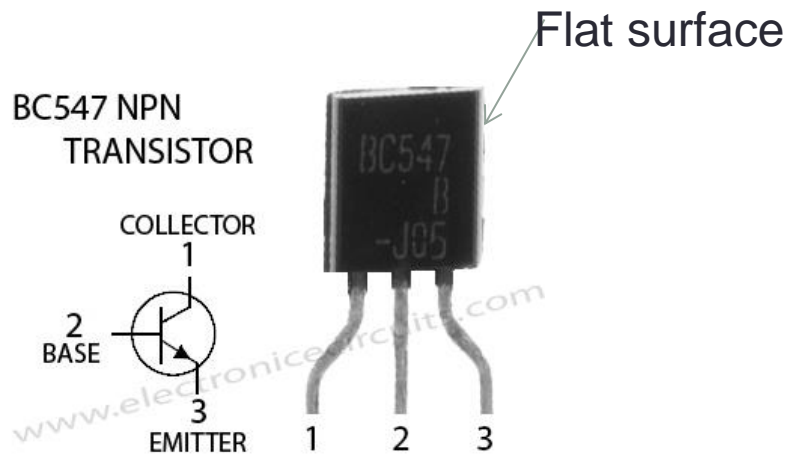
No!!!

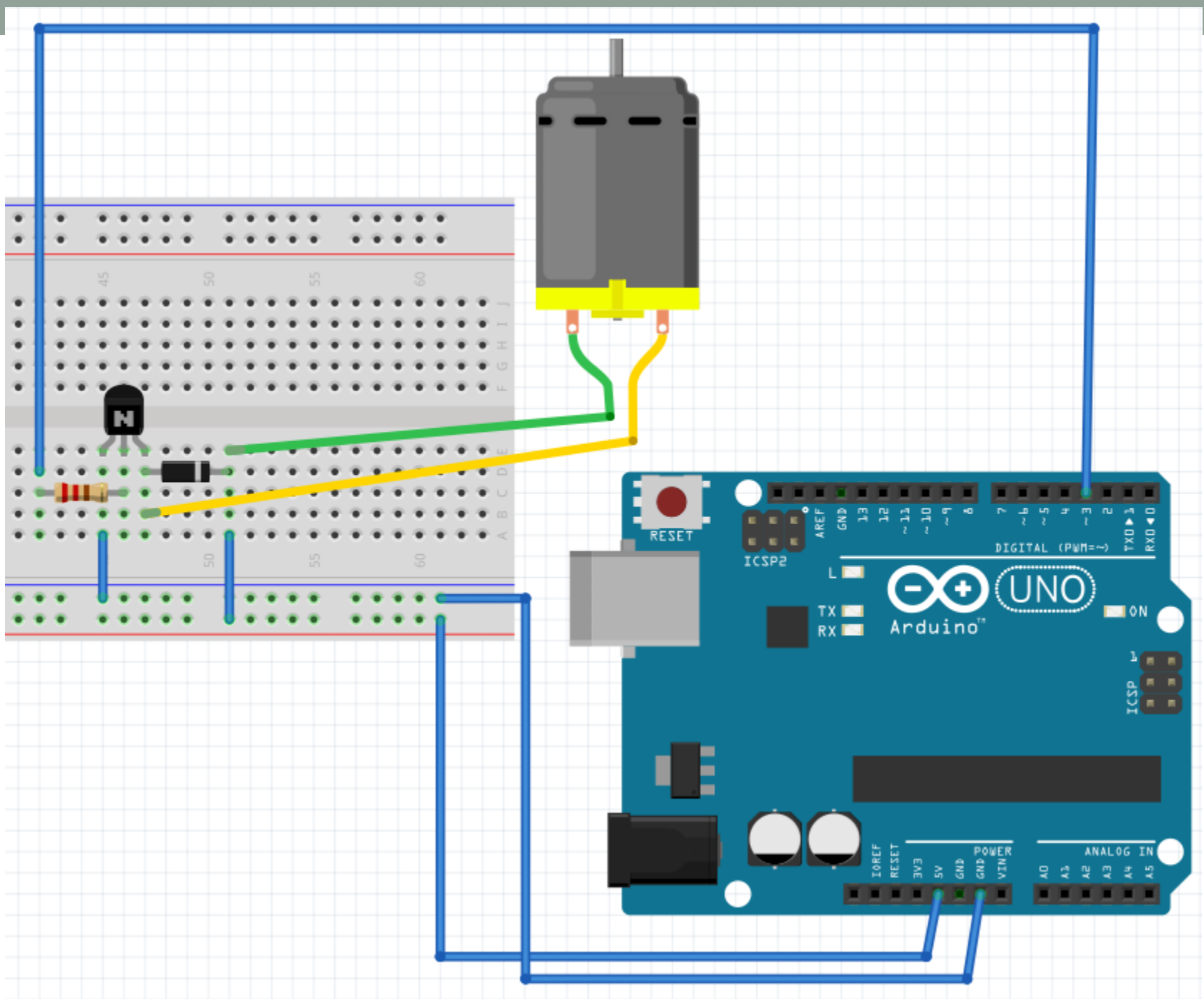


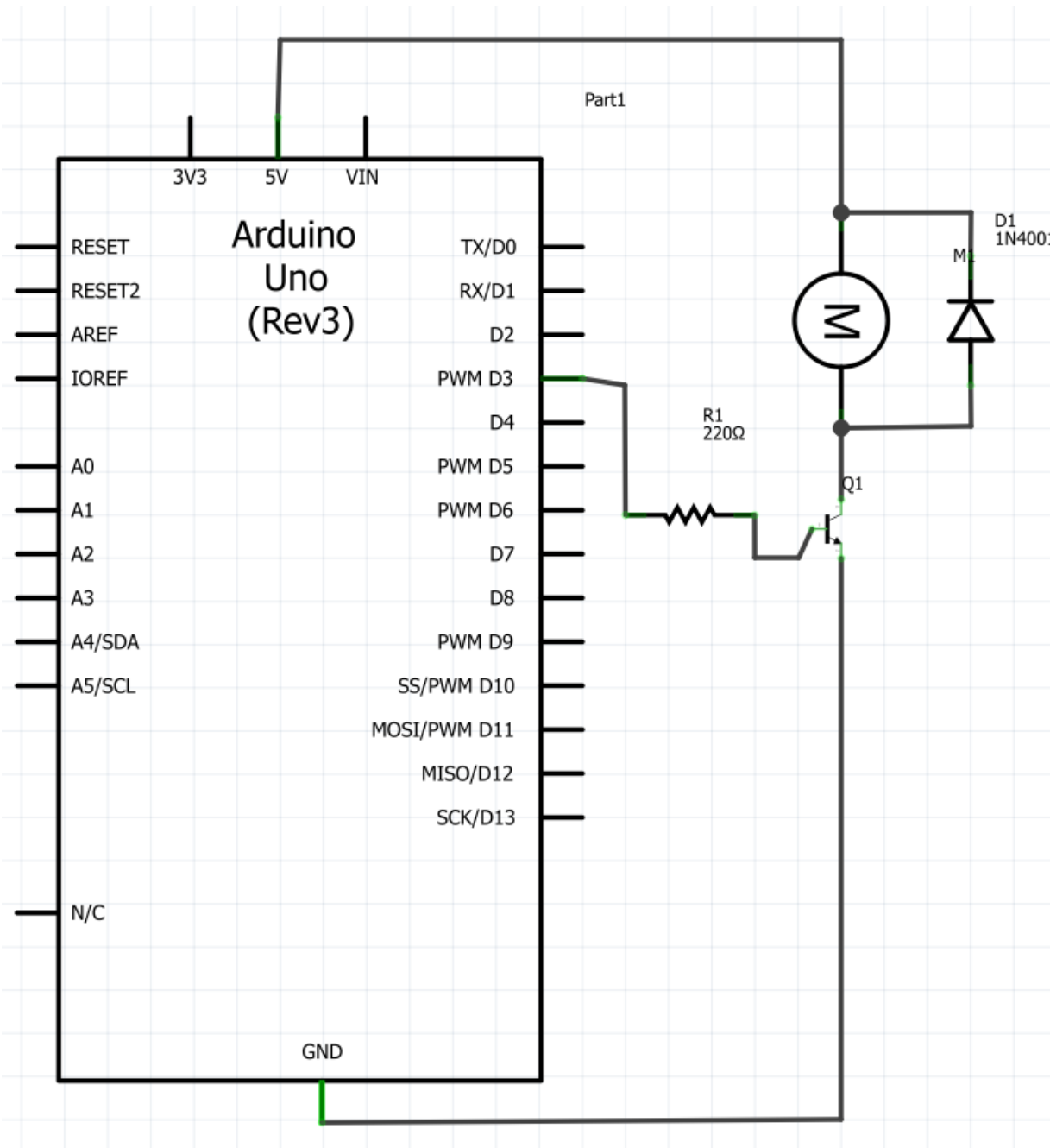
- The small DC motor, is likely to use more power than an Arduino output can handle directly.
- If we tried to connect the motor straight to an Arduino pin, it may damage the Arduino.

How to connect a DC Motor to Arduino

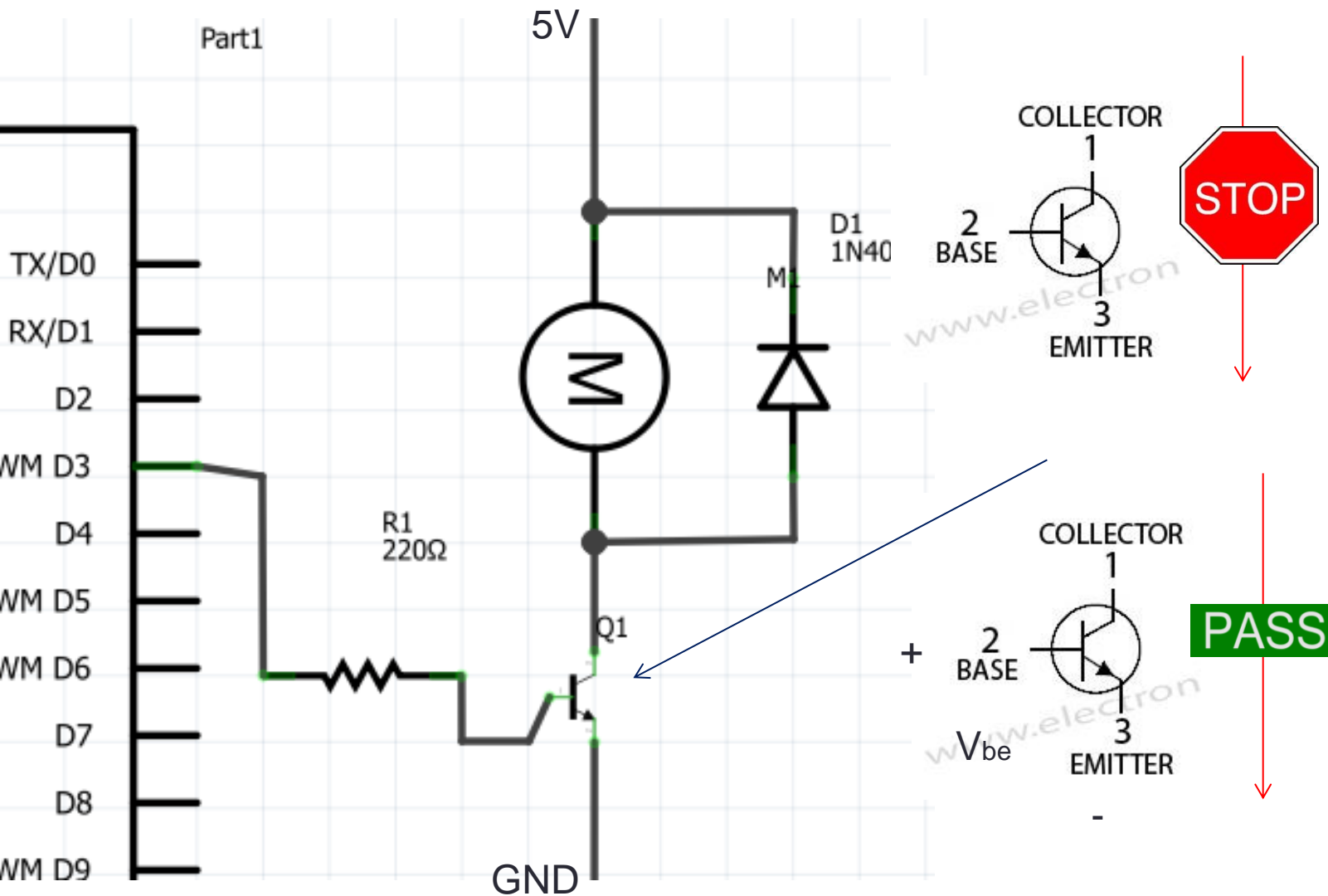
- We use a transistor as a switch (Week 2) to driver the DC motor.
- A diode (Week 2) to protect the Arduino



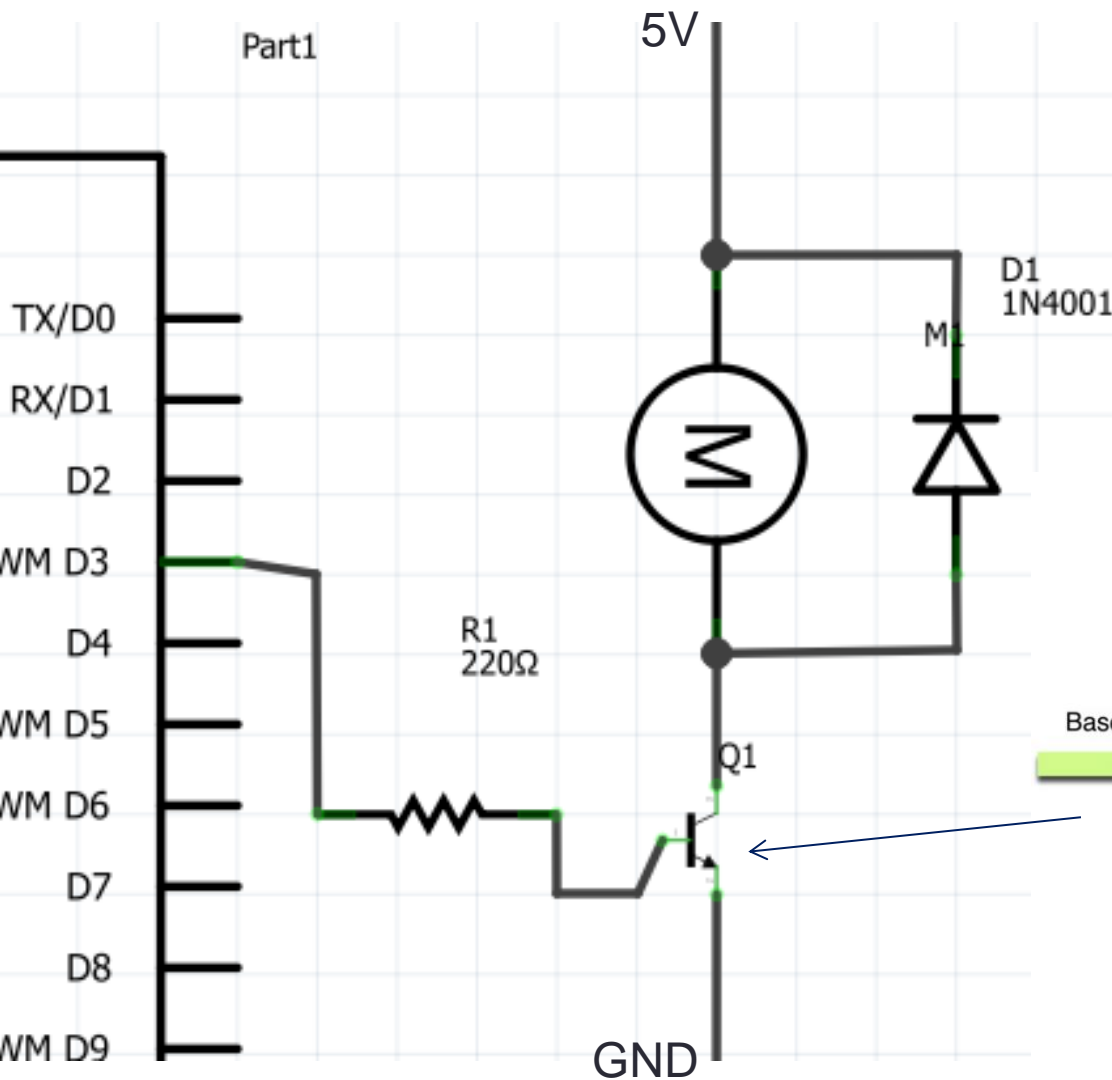




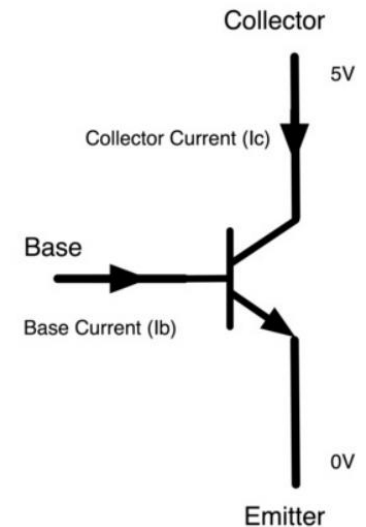
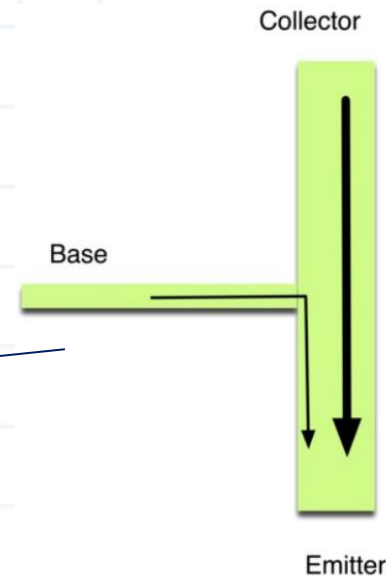
The transistor in the circuit



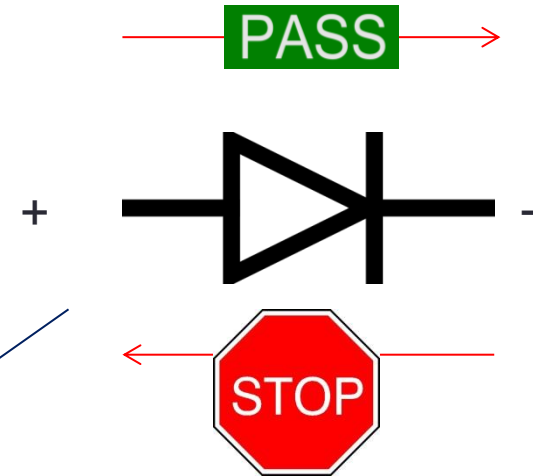
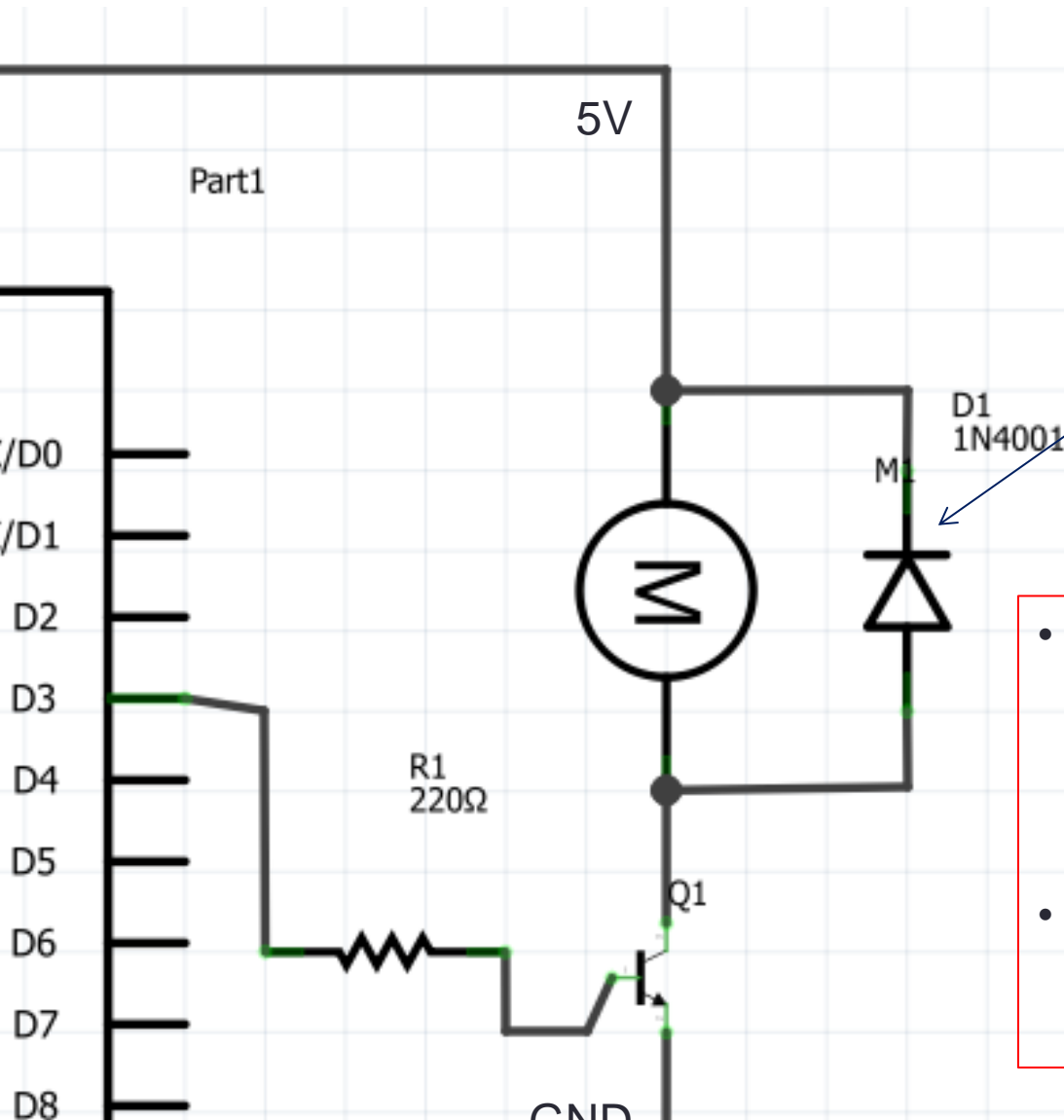
Control the transistor to control the motor



The bigger the input to Base pin, the bigger the current that is allowed to pass from Collector Pin.



The Diode in the circuit



- When you turn the power off to a motor, the motor generates a negative spike of voltage, that can damage your Arduino or the transistor.
- The diode protects against this, by shorting out any such reverse current from the motor.

Now the code

```
int motorPin= 9; //output pin number
void setup()
{
}

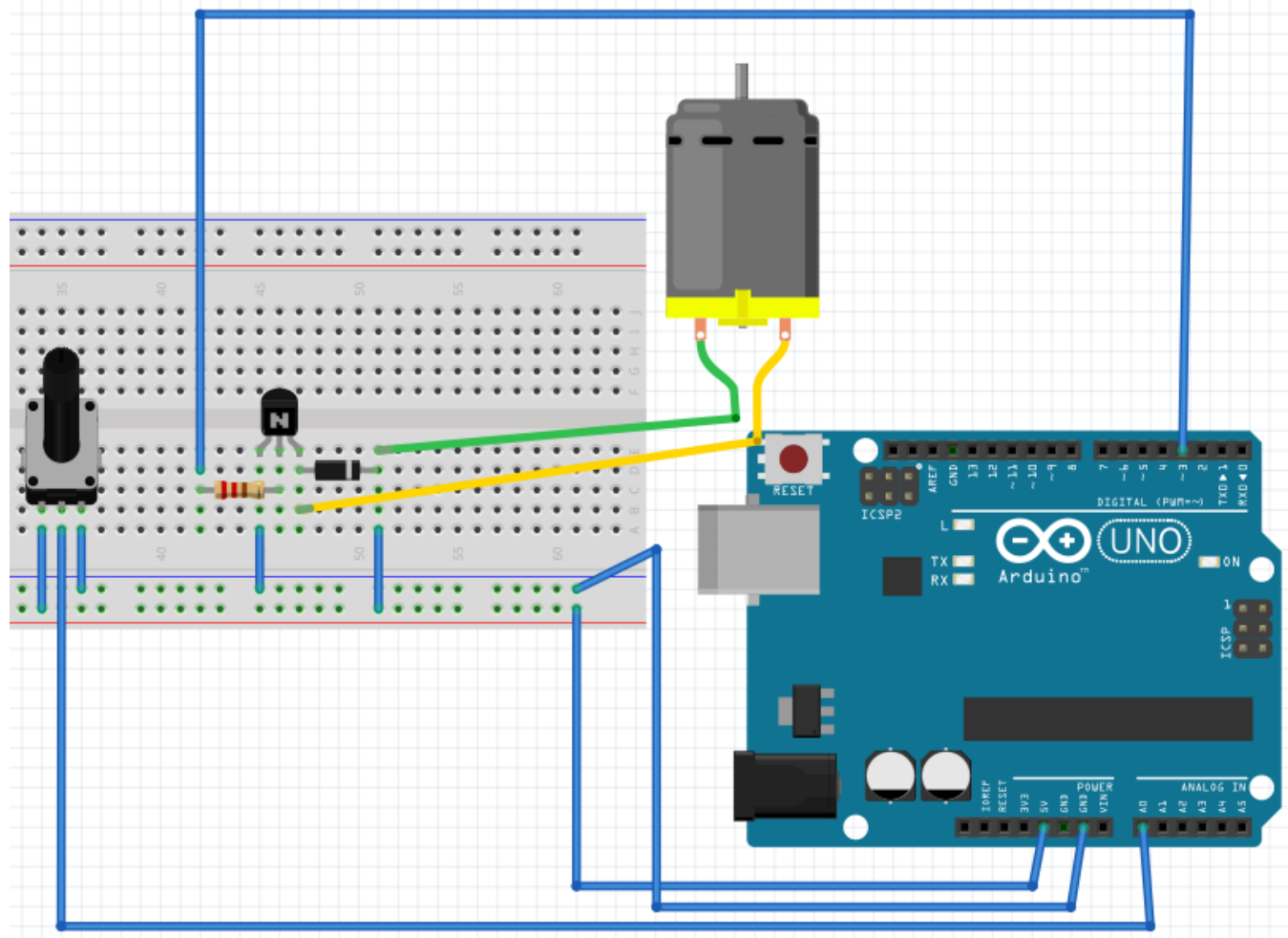
void loop()
{
    //analogWrite(pin, value) is the funtion of generating
    //analog signal
    // fade in from min to max in increments of 5 points:
    for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
        // sets the value (range from 0 to 255):
        analogWrite(motorPin, fadeValue);
        // wait for 30 milliseconds to see the dimming effect
        delay(3000);
    }

    // fade out from max to min in increments of 5 points:
    for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
        // sets the value (range from 0 to 255):
        analogWrite(motorPin, fadeValue);
        // wait for 30 milliseconds to see the dimming effect
        delay(3000);
    }
}
```

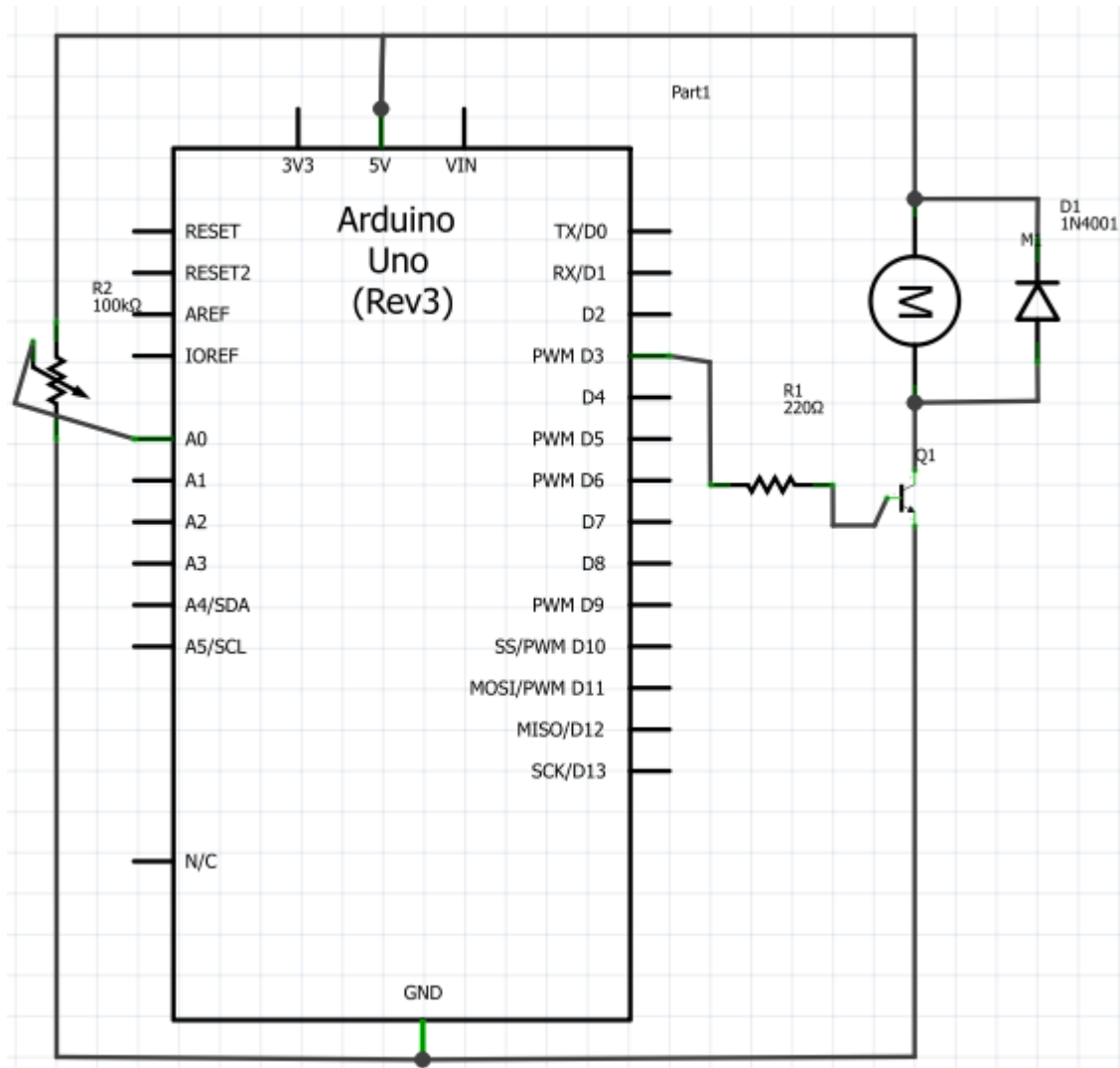
- It is actually the same as the code for Analog output to fade an LED/speaker
- Now you can see a motor spinning fast and slow continuously

You can also control the DC motor with sensors

- Combine what we have learnt last week with DC motor circuit



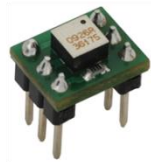
- Then we need to write the new code as well



Try to connect others sensors



Potentiometer
(Changeable Resistor)



Orientation Sensor



Infrared Motion Sensor



Light Dependent
Sensor (Resistance
changed by light)



Ultrasound Distance
Sensor

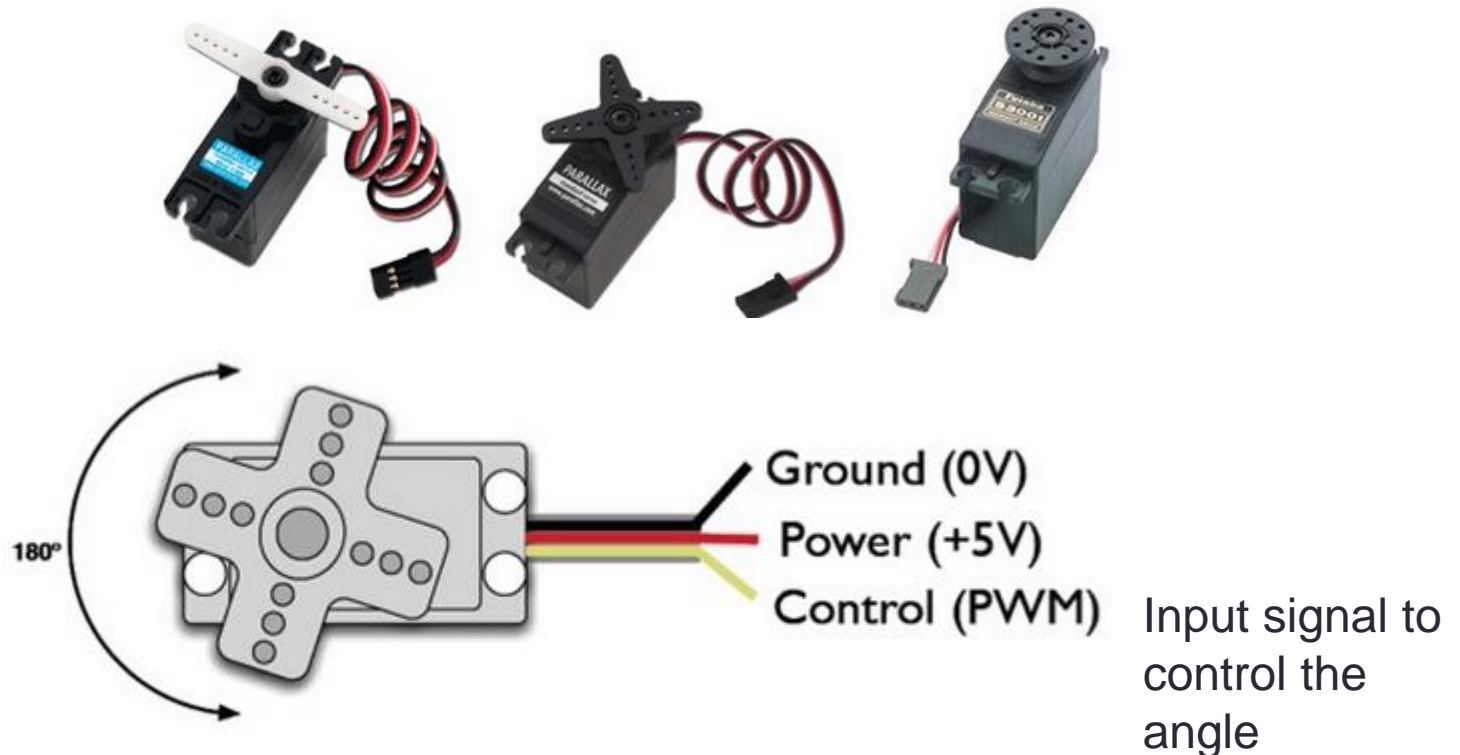


Flexible Vibration
Sensor

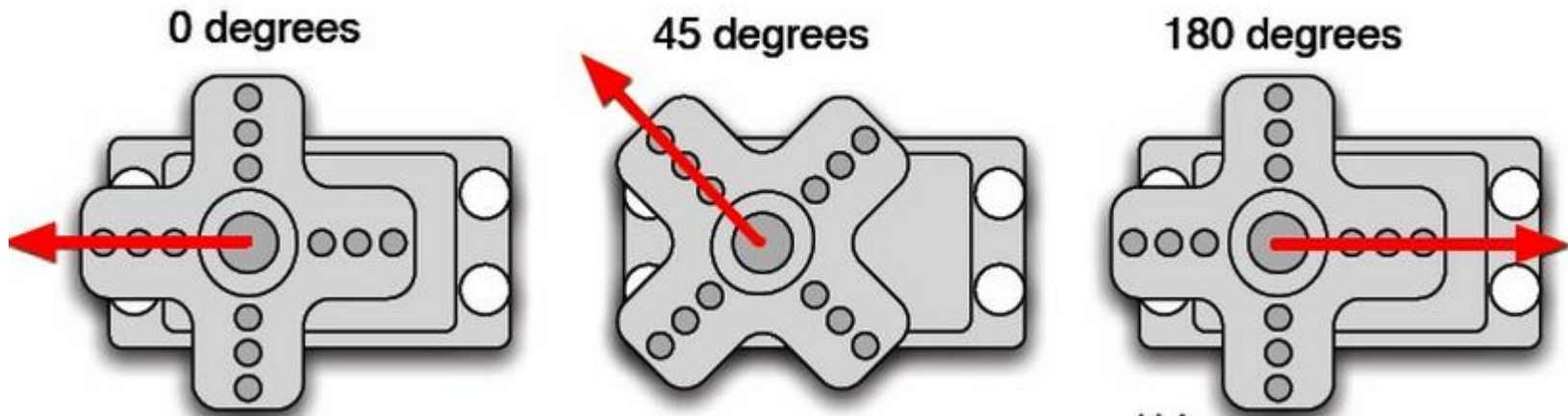
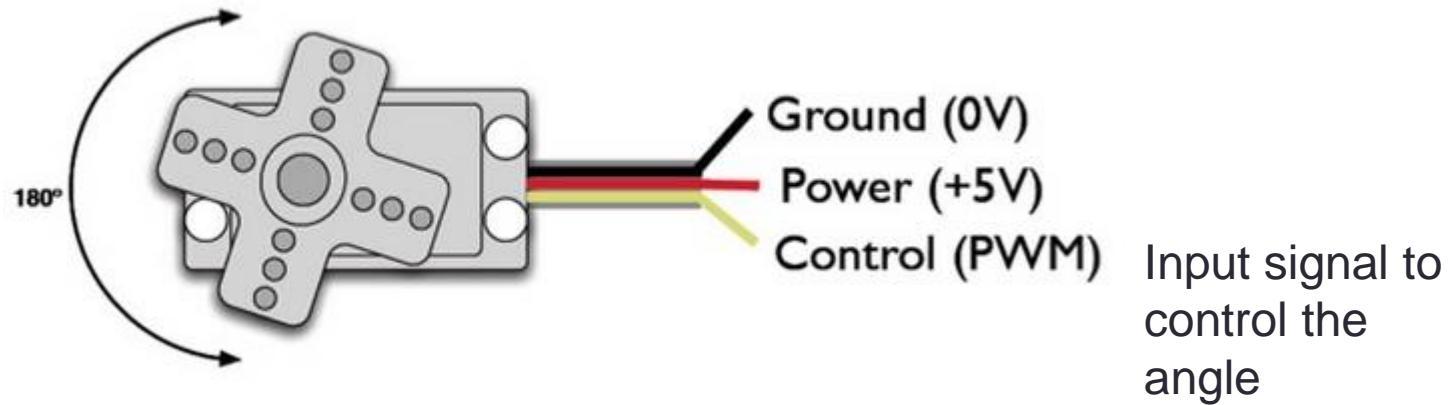


Servo Motor

- It is hard to control the angle and the direction of a DC motor
- But we can control the angle and the direction of a Servo motor easily.

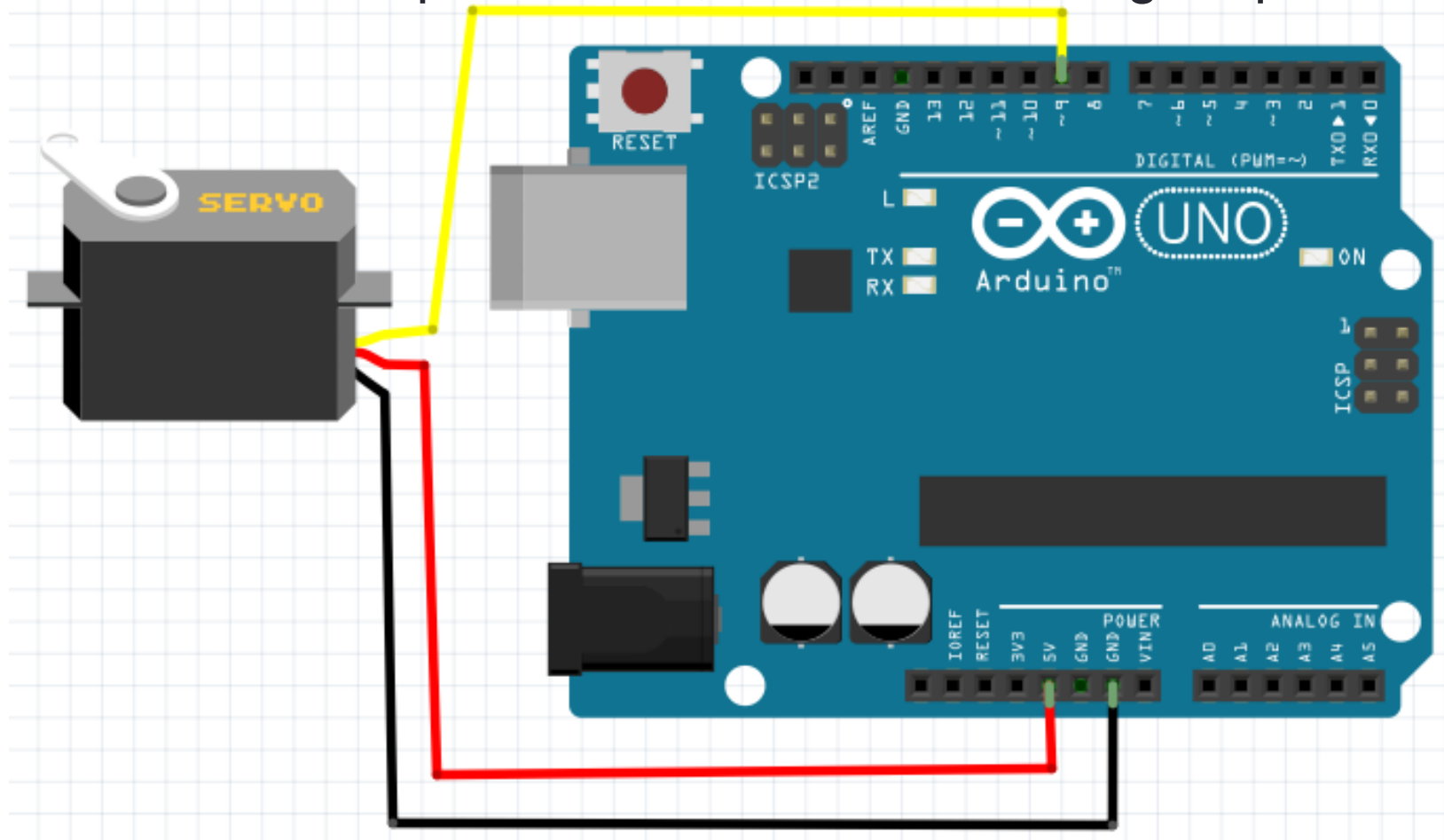


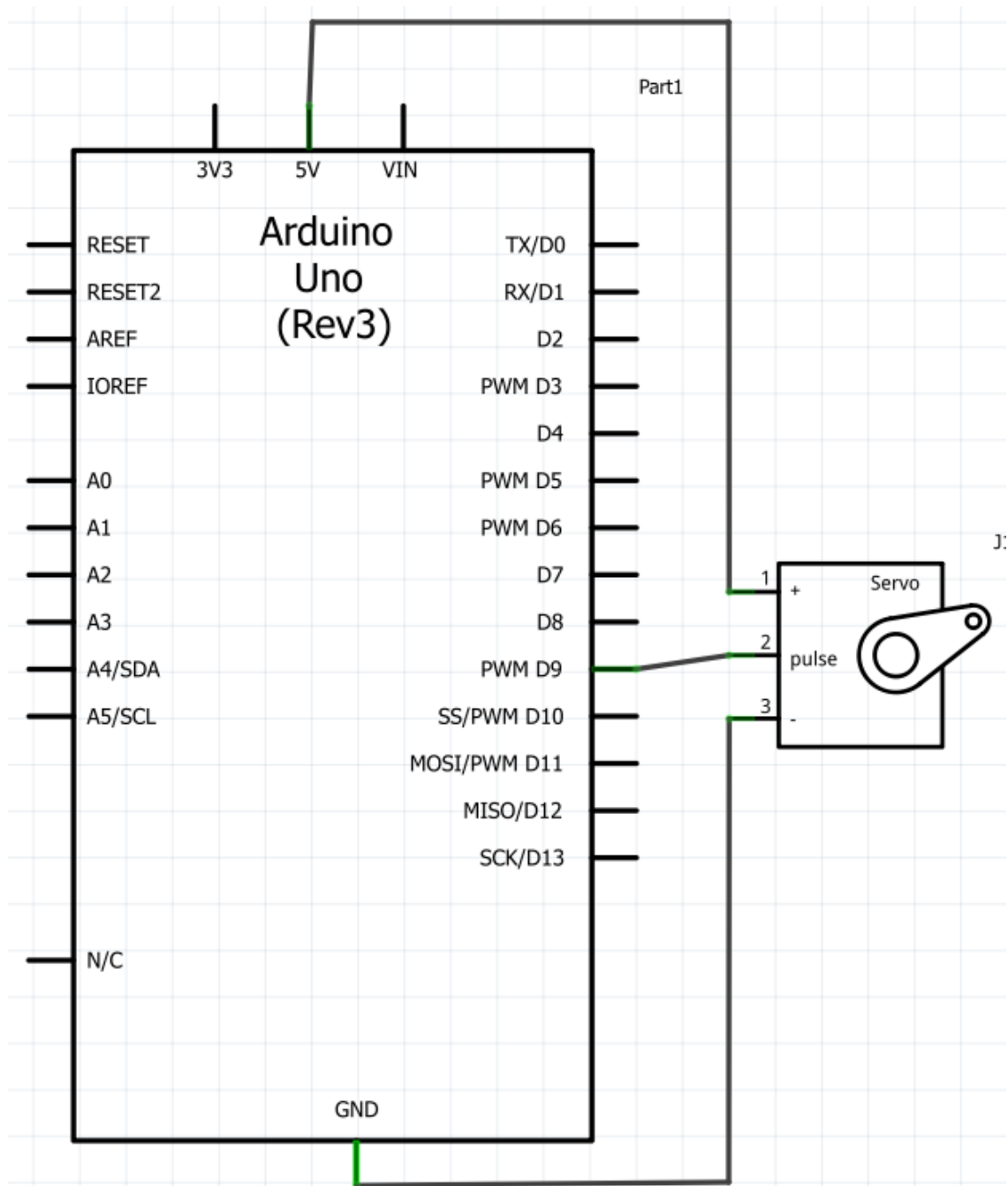
We can control the angle from the code



Connect the servo motor to Arduino

- Connect to the pin with “~”, to receive analog output





Connection is easy, but coding is a bit tricky

```
#include <Servo.h>
```

```
int servoPin = 9;
```

```
Servo servo;
```

```
int angle = 0; // servo position in degrees
```

```
void setup()
```

```
{  
  servo.attach(servoPin);  
}
```

```
void loop()
```

```
{  
  // scan from 0 to 180 degrees  
  for(angle = 0; angle < 180; angle++)  
  {  
    servo.write(angle);  
    delay(15);  
  }  
  // now scan back from 180 to 0 degrees  
  for(angle = 180; angle > 0; angle--)  
  {  
    servo.write(angle);  
    delay(15);  
  }  
}
```

- We need to use the Servo library, so we can use the functions to control the motor

```
#include <Servo.h>
```

- Create Servo Motor Object

```
Servo servo;
```

- Connect the motor to a particular pin on the board

```
servo.attach(pin);
```

- Set the angle of the motor

```
servo.write(angle)
```

You can connect the sensors to control as well!!



Potentiometer
(Changeable Resistor)



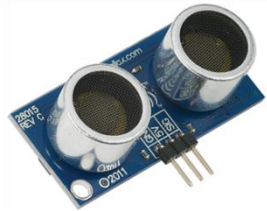
Orientation Sensor



Infrared Motion Sensor



Light Dependent
Sensor (Resistance
changed by light)



Ultrasound Distance
Sensor



Flexible Vibration
Sensor



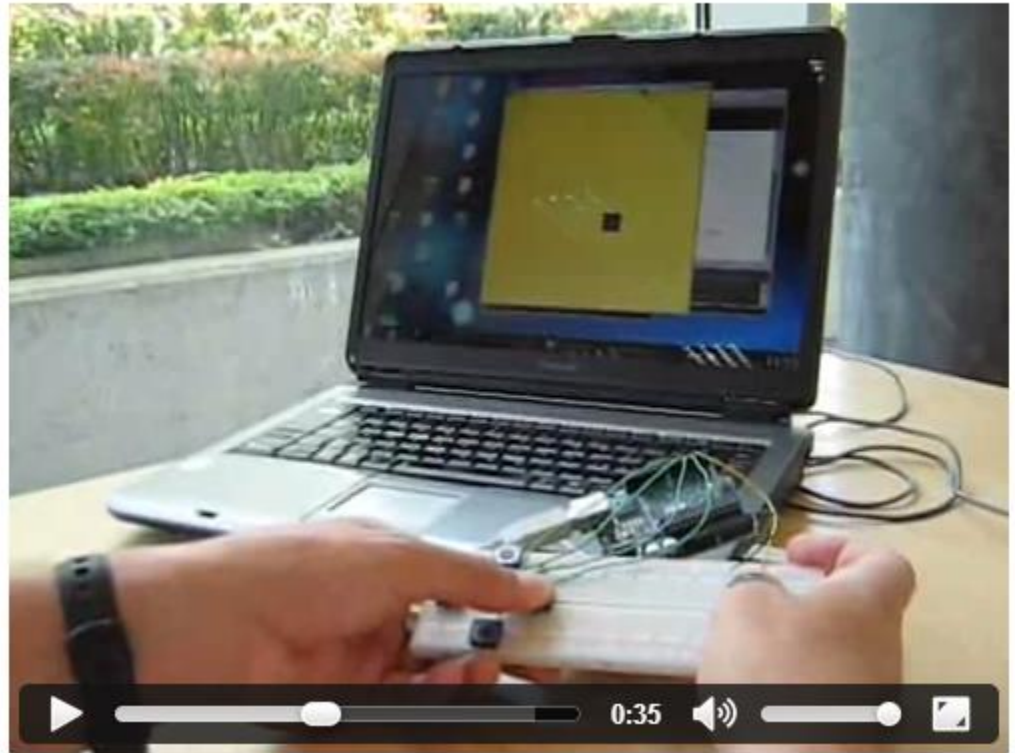
Now we have seen how Arduino can help us to...

- Control LEDs, Speakers, and Motors
- Sense the input with..
 - Buttons
 - Potentiometers
 - Light sensors
 - Vibration sensors
 - Distance sensors
 - Infrared Motion sensors
 - Tilt sensors

How about using Arduino to control our computer?

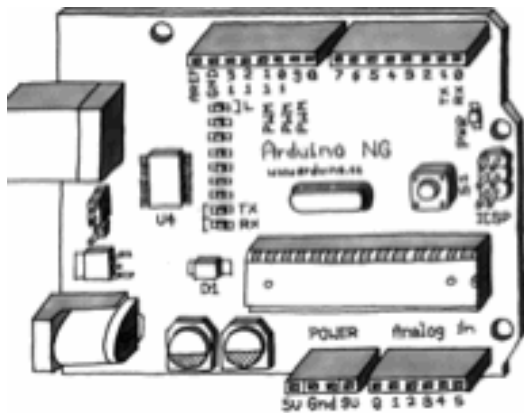
- Color
- Games
- Animations
- Music

Following a demonstration of the Arduino controller and the Processing program:



<http://www.varesano.net/blog/fabio/serial-communication-arduino-and-processing-simple-examples-and-arduino-based-gamepad-int>

We send the signal to computer through USB serial connection



Before talking about the communication, we need to know...

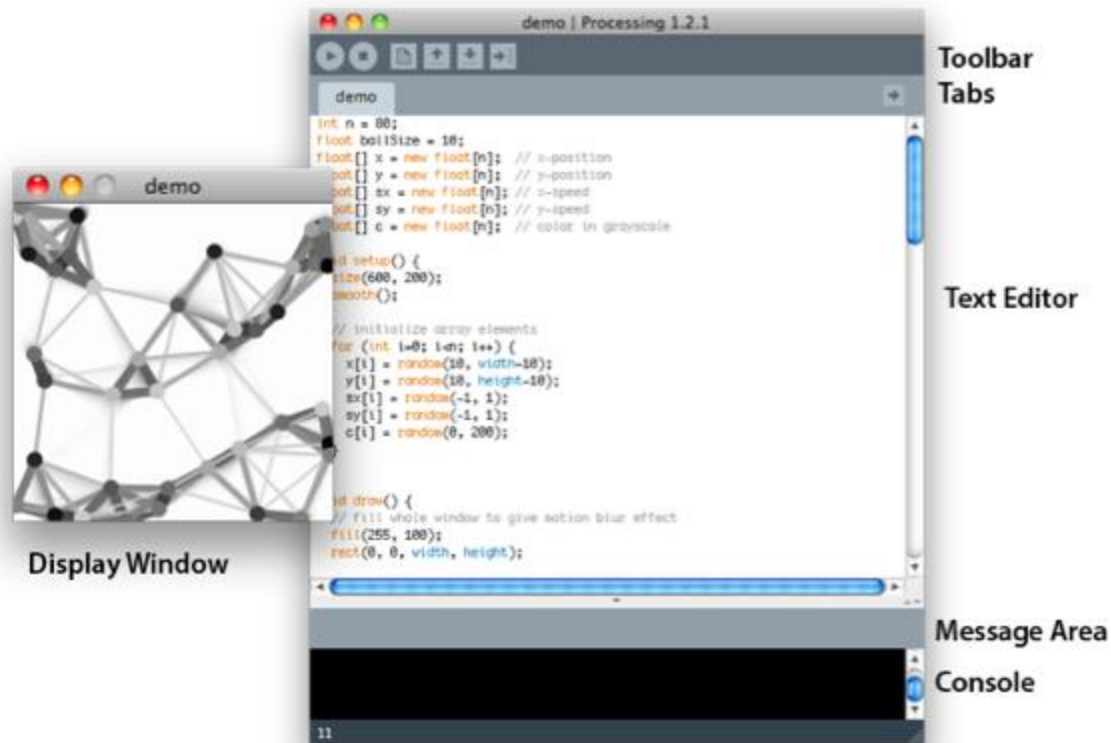
- Processing

- Processing is a programming language, development environment, and online community that since 2001 has promoted software literacy within the **visual arts**.
- You will mainly learn it in the second half of the semester.
- Here I will only talk about some basic concept, and how it connects the Arduino and the computer



Basic Processing Programming

- Processing Development Environment (PDE)

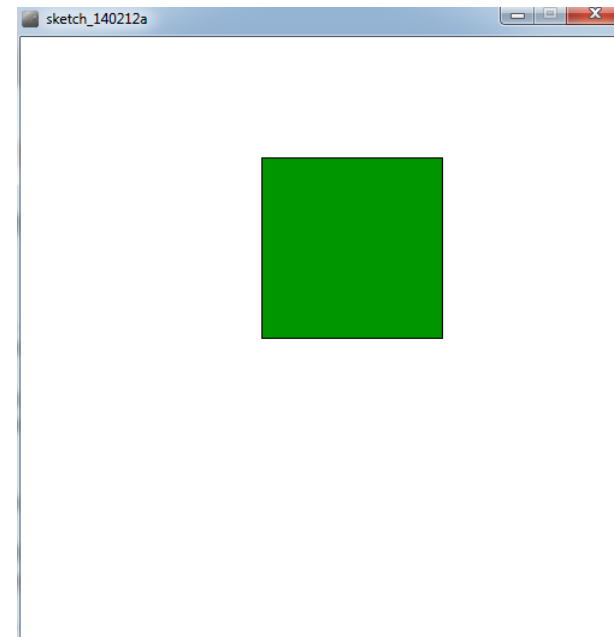


Basic Processing Programming

- A first simple Processing drawing

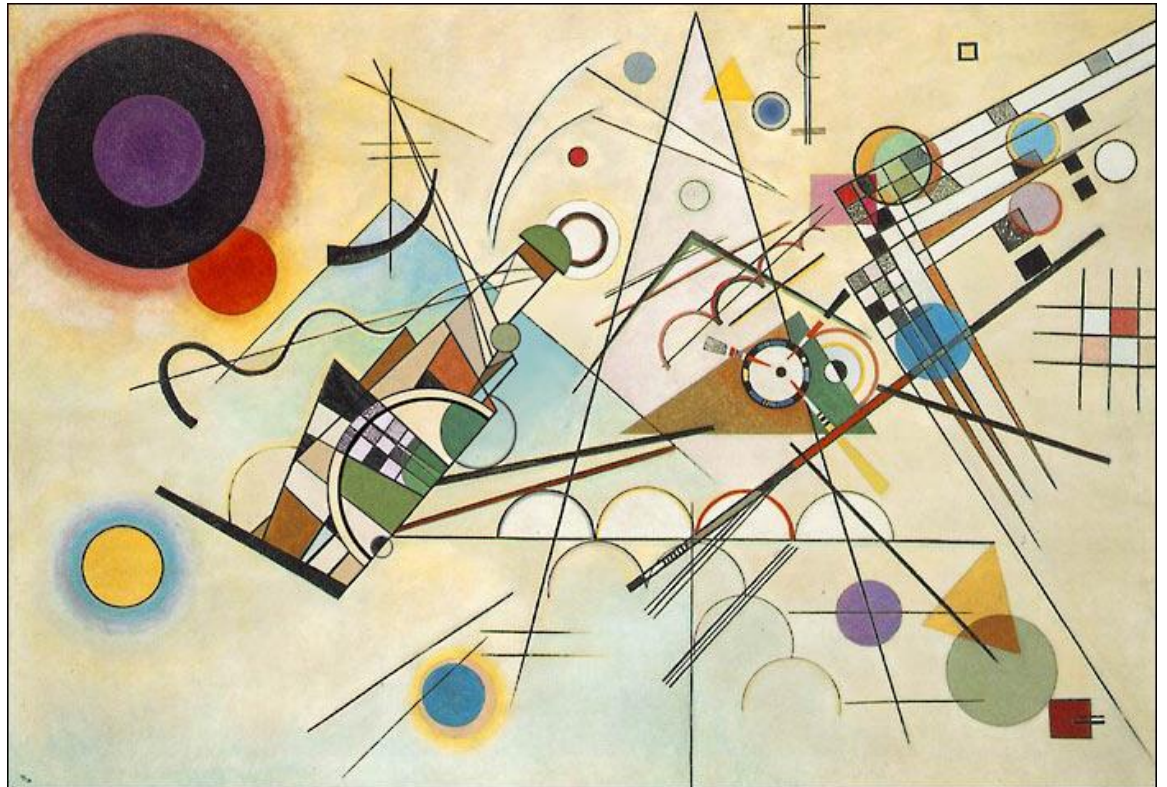
```
void setup()
{
  background(255,255,255);
  size(500,500);
}

void draw()
{
  fill(0,150,0);
  rect(200,100,150,150);
}
```



Basic Processing Programming

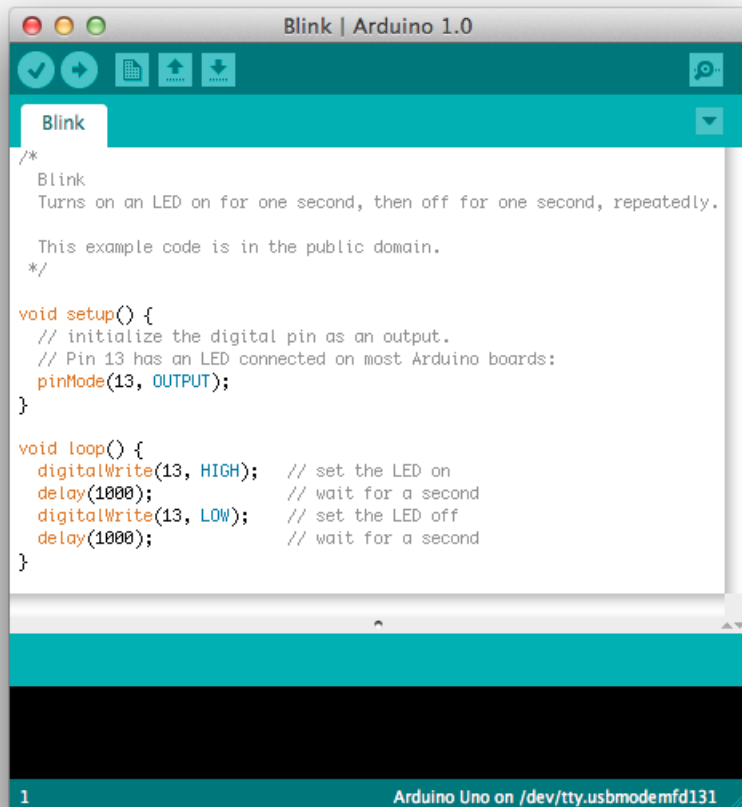
- Basic drawing
 - Coordinates
 - Size
 - Shapes
 - Line
 - Circle
 - Box
 - Curve
 - etc.
 - Color



Processing communicates with Arduino

- **We use Processing Serial Library**
- The Processing **serial library** allows for easily reading and writing data to and from external machines.
- It allows two devices to send and receive data and gives you the flexibility to communicate with custom microcontroller devices,
- using them as the input or output to Processing programs. (<http://processing.org/reference/libraries/serial/index.html>)

Connect the Arduino using USB (Week 3)



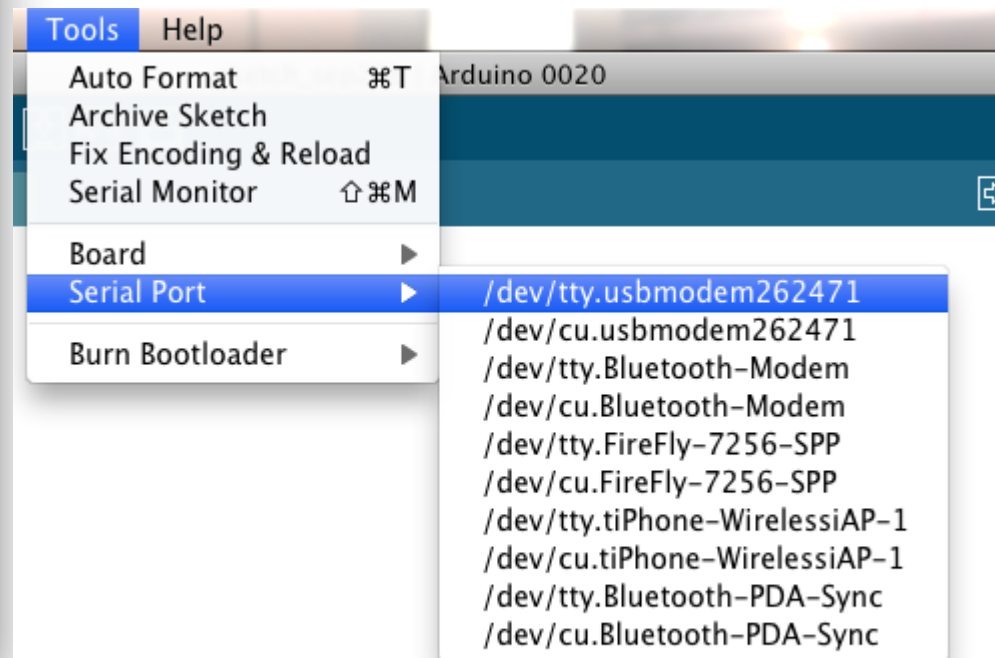
The screenshot shows the Arduino IDE window titled "Blink | Arduino 1.0". The code editor contains the following code:

```
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */

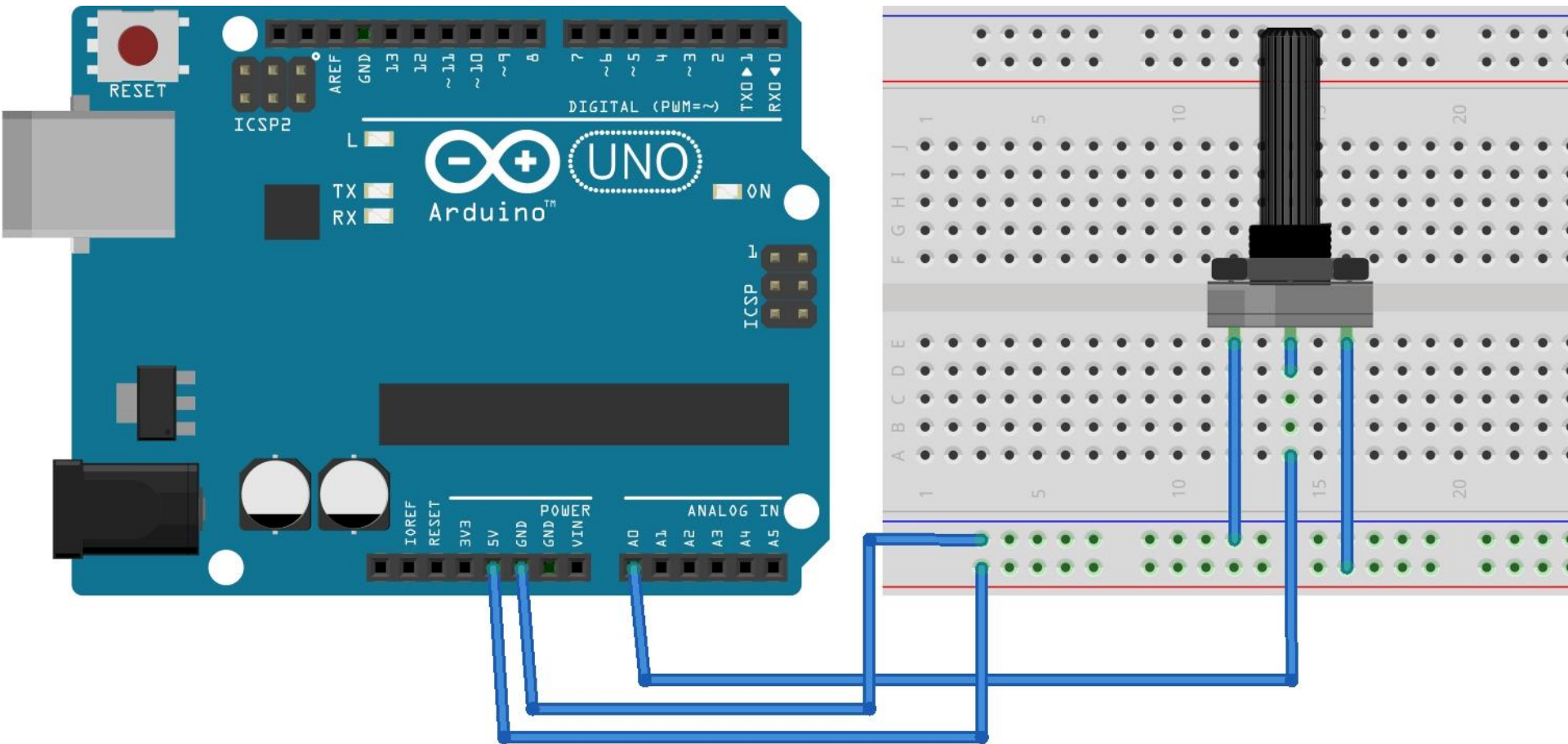
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);            // wait for a second
}
```

The status bar at the bottom indicates "1" and "Arduino Uno on /dev/tty.usbmodemfd131".



Make the circuit (Here is an example)



Setup the serial communication in Arduino code

```
void setup()
{
  Serial.begin(9600); //set up the serial connection
}

void loop()
{
  //get value from sensors/buttons
  int value = analogRead(0);
  value = map(value, 0, 1023, 0, 255); //re-map the value to a different range
  //print the value to serial connection, send to the computer
  Serial.write(value);
  delay(100);
}
```

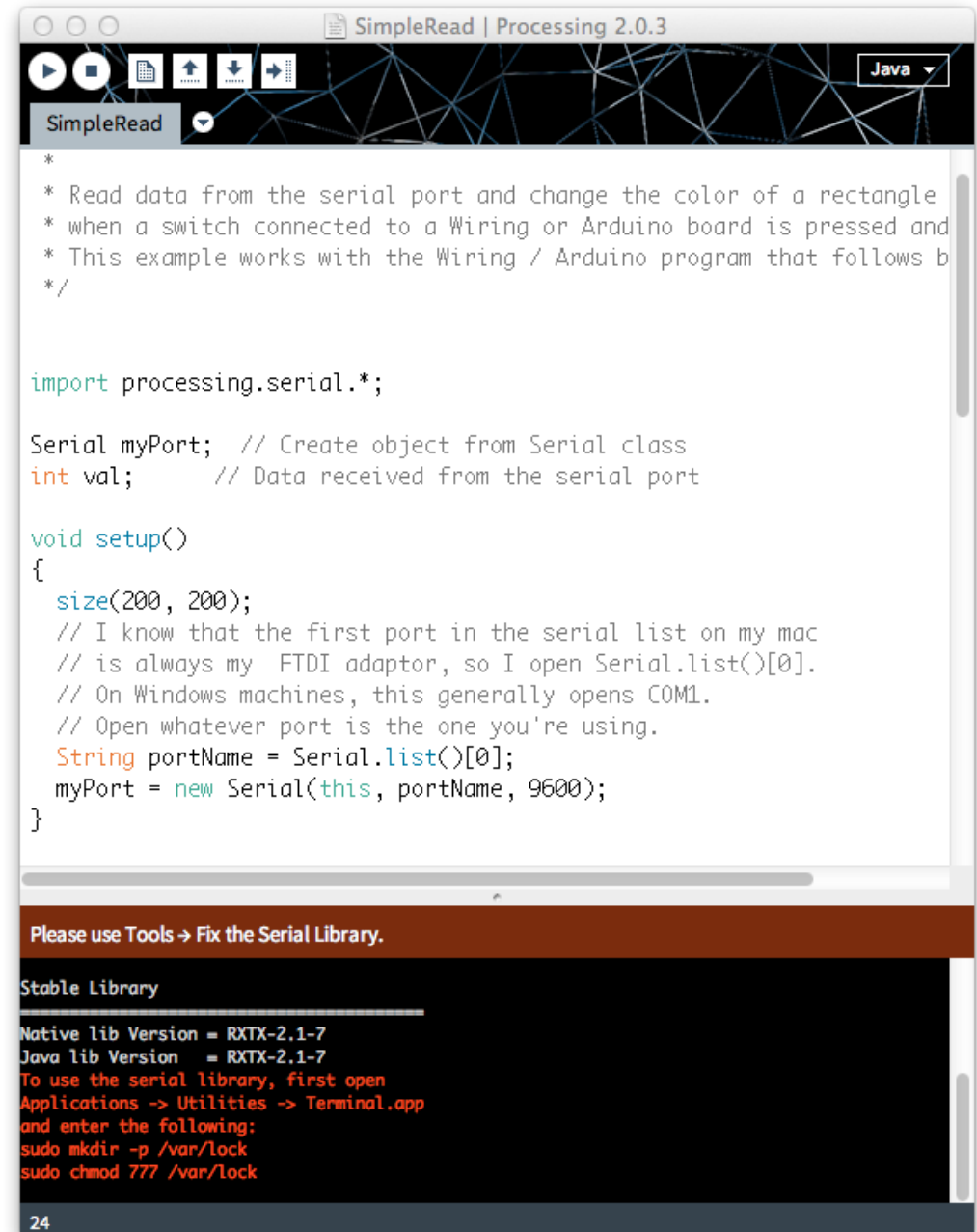
New Functions

`serial.write(byte)`

- write a byte (numerical value in binary format) to serial port.

Upload Arduino code to board (remember NOT to open the Serial Monitor of Arduino IDE)

Configuration to use Serial Library on Mac



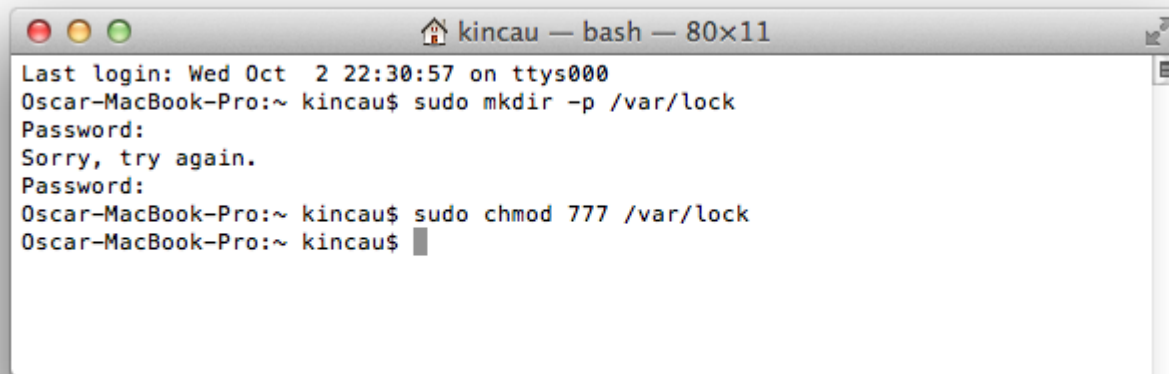
- To use serial library on Mac machine, you may need to create a folder for storing lock file:

1. Open Application => Utilities => Terminal

2. Enter the following commands (you may need to enter admin's password):

```
sudo mkdir -p /var/lock
```

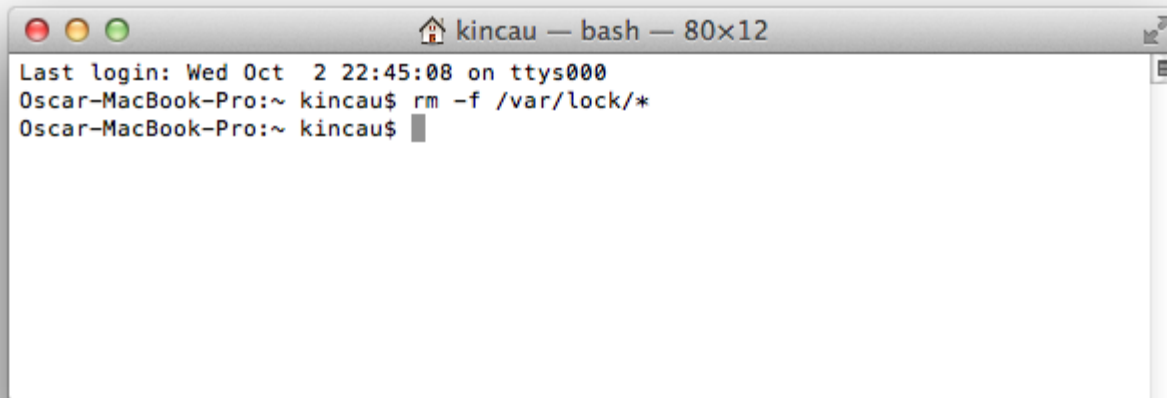
```
sudo chmod 777 /var/lock
```

A screenshot of a macOS Terminal window. The title bar shows 'kincau — bash — 80x11'. The terminal output shows the user 'kincau' at 'Oscar-MacBook-Pro' with a tilde prompt. The first command is 'sudo mkdir -p /var/lock', which prompts for a password. The user enters a password, but it is rejected with 'Sorry, try again.' and prompts for the password again. The second command is 'sudo chmod 777 /var/lock', which is executed successfully. The prompt returns to 'kincau\$'.

```
kincau — bash — 80x11
Last login: Wed Oct  2 22:30:57 on ttys000
Oscar-MacBook-Pro:~ kincau$ sudo mkdir -p /var/lock
Password:
Sorry, try again.
Password:
Oscar-MacBook-Pro:~ kincau$ sudo chmod 777 /var/lock
Oscar-MacBook-Pro:~ kincau$
```

- **Remove old lock files:** Sometime you will see a warning like "RXTX Warning: Removing stale lock file. /var/lock/LK.017.033.006", in this case you need to remove the lock file by using the command below to remove the lock files.

`rm -f /var/lock/*`

A screenshot of a macOS terminal window. The title bar shows a home icon, the text "kincau — bash — 80x12", and standard window control buttons. The terminal content shows a login message "Last login: Wed Oct 2 22:45:08 on ttys000", followed by the prompt "Oscar-MacBook-Pro:~ kincau\$". The command "rm -f /var/lock/*" has been entered and executed, with the prompt now showing "Oscar-MacBook-Pro:~ kincau\$".

```
kincau — bash — 80x12
Last login: Wed Oct 2 22:45:08 on ttys000
Oscar-MacBook-Pro:~ kincau$ rm -f /var/lock/*
Oscar-MacBook-Pro:~ kincau$
```

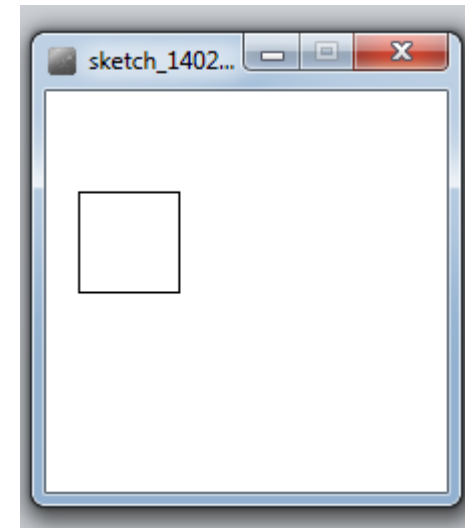
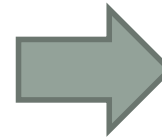
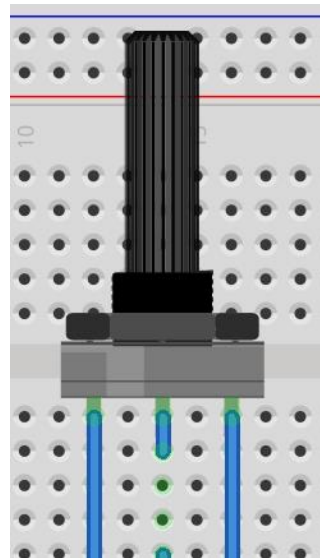
Setup the serial communication in Processing code in the computer side

```
import processing.serial.*;
```

```
Serial myPort;  
char val;
```

```
void setup()  
{  
  size(200, 200);  
  
  // the name of port connecting the arduino  
  println(Serial.list());  
  myPort = new Serial(this, Serial.list()[0], 9600);  
}
```

```
void draw()  
{  
  // read data if available  
  if (myPort.available() > 0) {  
    val = myPort.readChar();  
  }  
  
  // Set background to white  
  background(255);  
  //draw a box using the value from Arduino  
  rect(val, 50, 50, 50);  
}
```



Run processing code

So...



Finally...we are done with Physical Computing!

- Please do come to talk to me if you have any question
 - It is hard to cover all the details in less than 6 weeks

Tutorial for this week

- Making of the mini project
 - Spend time of brainstorming and making your mini project
 - I will be here to help if you have any problem/difficulty.