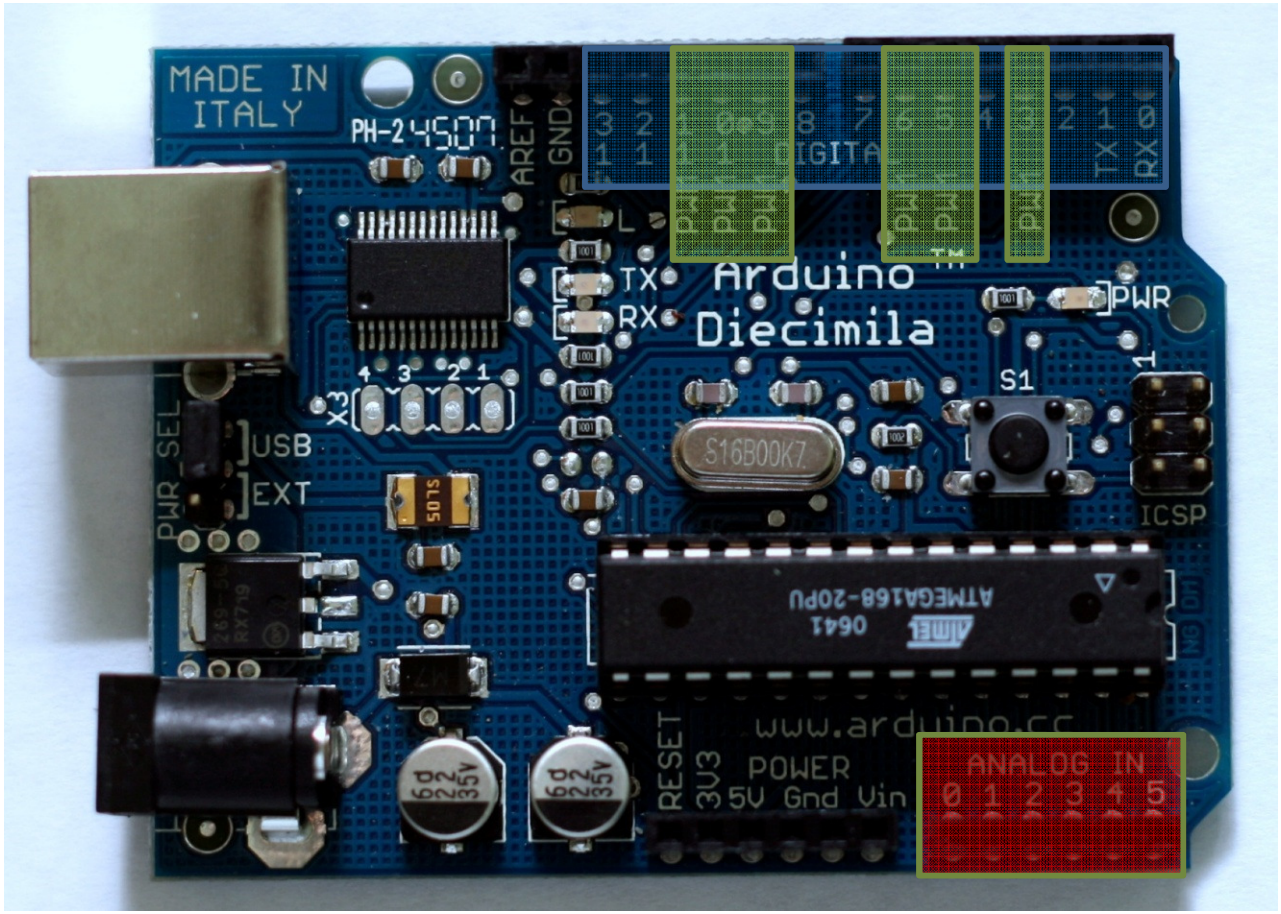
Digital I/O
Analog I/O

Quick Recap

Closer look at the board



Digital In:

High / Low

Digital Out:

High / Low

Analog In:

0 – 1023

Analog Out:

0 – 255

Recap

ADC’s

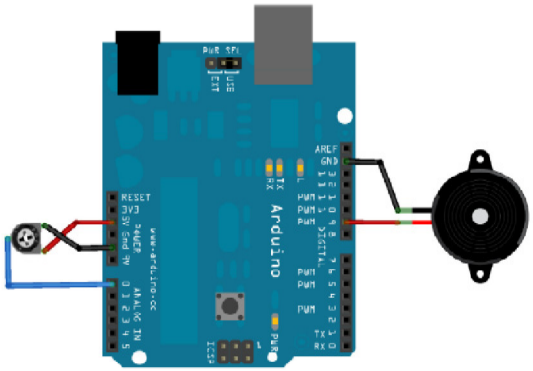
PWM

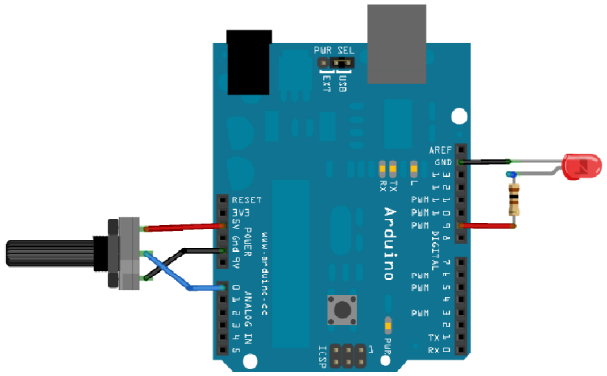
Potentiometer

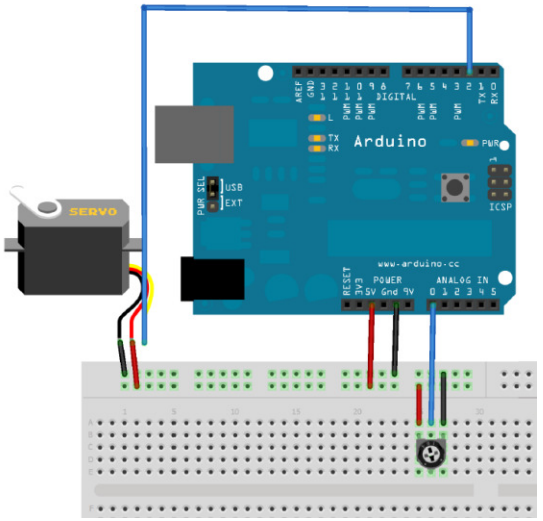
LED dimming

Piezo Speaker

Servo Motor







Coding in Arduino

- Brief introduction:
  - on the project, references, types of electronics used, Arduino pins used etc.
- Include external library:
  - #include <filename.h>
- Declaring Variables:
  - Introducing all different numeric, character types used in the program
- Structure:
  - Setup Function:
    - Start of Sketch, One time Run
    - Initialization of variables, assigning pins etc.
  - Loop Function:
    - Keeps looping the program written in this function
    - Main coding area
- Custom Functions:
  - Write our own functions

```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

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

Introduction and Commenting

```
/*
    Basic Intro to project (function / usefulness)
    Type of electronics used in project
    Considerations / Assumptions / Cautions /
    Limitations
    Creation date / Author
    References
*/

// Single line description of the current line in the code
```

External library files

#include <filename.h> //Simplifies My Code

To load external libraries (set of functions)

```
int servoPin = 2;    // Control pin for servo motor
int minPulse = 500;   // Minimum servo position
int maxPulse = 2500;  // Maximum servo position
int pulse = 0;       // Amount to pulse the servo
int lastPulse = 0;    // the time in milliseconds of the last pulse
int refreshTime = 20; // the time needed in between pulses
int analogValue = 0;  // the value returned from the analog sensor
int analogPin = 0;    // the analog pin that the sensor's on
void setup() {
  pinMode(servoPin, OUTPUT);  // Set servo pin as an output pin
  pulse = minPulse;          // Set the motor position value to the minimum
  Serial.begin(9600);
}
void loop() {
  analogValue = analogRead(analogPin);     // read the analog input pulse = (analogValue / 10) * 19 + 500;
                                           // convert the analog value
                                           // to a range between minPulse
                                           // and maxPulse.
  // pulse the servo again if rhe refresh time (20 ms) have passed:
  if (millis() - lastPulse >= refreshTime) {
    digitalWrite(servoPin, HIGH);   // Turn the motor on
    delayMicroseconds(pulse);      // Length of the pulse sets the motor position
    digitalWrite(servoPin, LOW);    // Turn the motor off
    lastPulse = millis();          // save the time of the last pulse
  }
}
```

```
#include <SoftwareServo.h>
SoftwareServo myservo; // create servo object to control a servo
int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin
void setup()
{
  myservo.attach(2);  // attaches the servo on pin 2 to the servo object
}
void loop()
{
  val = analogRead(potpin);          // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 179);     // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
  SoftwareServo::refresh();
}
```

All Arduino libraries are in the following folder :

\arduino-0017\arduino-0017\hardware\libraries

Add new Header files: copy files in the same folder

#define = #include

Declaring Variables

- Storage packets of values
- Types:
  - Constants
    - HIGH / LOW
    - INPUT / OUPUT
    - TRUE / FALSE
  - Data Types
  - Conversion

Data Types
int variable; int variable = value;

- boolean - boolean x = true; x = !x; (TRUE / FALSE)
- char - to assign Decimal ASCI character.
- byte - 8-bit number.
- Int - integers (-32768 to 32767) (-2^15 to 2^15-1)
- unsigned int - integers (0 to 65535)
- long - extended sized values (-2147483648 to 2147483647)
- unsigned long -0 to 2^32 -1
- float /double - takes decimal values (-π to π with 38 decimal digits)
- String - arrays of char
- array - collection of variables accessed with index number
- void - used for function declaration

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	STX
^C	3	03	♥	ETX
^D	4	04	♦	EOT
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HT
^J	10	0A	◙	LF
^K	11	0B	♂	VT
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	DLE
^Q	17	11	◄	DC1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^T	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	ETB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SUB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	SP
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂†

† ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL+BKSP key.

Structure

- Setup() & Loop()
- Control Structures:
  - If / if… else
  - For
  - Switch case
  - While / do … while
  - Break
  - Continue
  - return

Functions

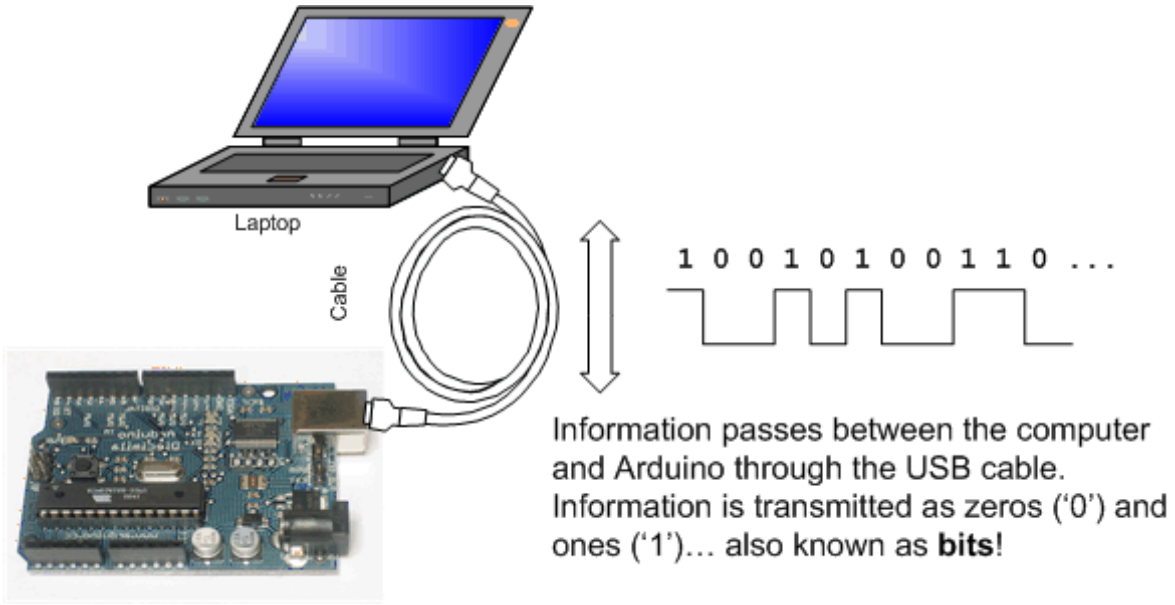
- System functions (http://arduino.cc/en/Reference/HomePage)
  - Digital I/O
  - Analog I/O
  - Advanced I/O
  - Time
  - Math
  - Trigonometry
  - Random Numbers
  - Communication
- Custom functions (http://www.arduino.cc/en/Reference/FunctionDeclaration)

Serial Communication

Bits, Bytes, Data Rates and Protocols
ASCI interpretation
Using terminal to view serial Data
Serial Out from Arduino
Serial In to Processing, PD, Max/MSP, Flash

Serial?

Communication done one after the other.

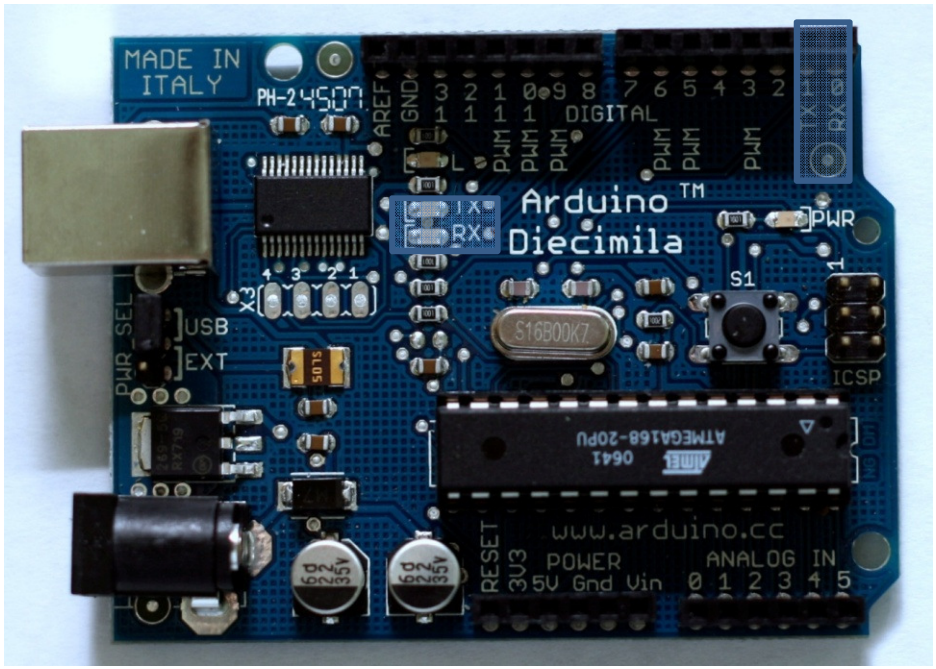


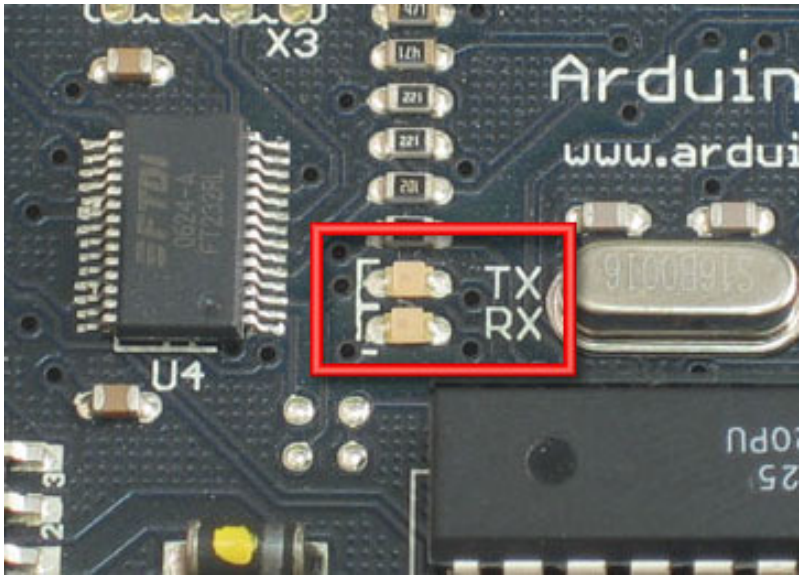
01100010011010010111010001110011001011000010
00000110001001111001011101000110010101110011

- The world isn’t run by weapons anymore, or energy, or money. It’s run by little 1’s and 0’s…
- BIT = 1 / 0
- BYTE = 8 BITs (10010010)
- KB = 1024 B
- MB = 1024 KB
- GB = 1024 MB

Serial Communication: Arduino

- RX – Receiving Data
- TX – Transmitting Data



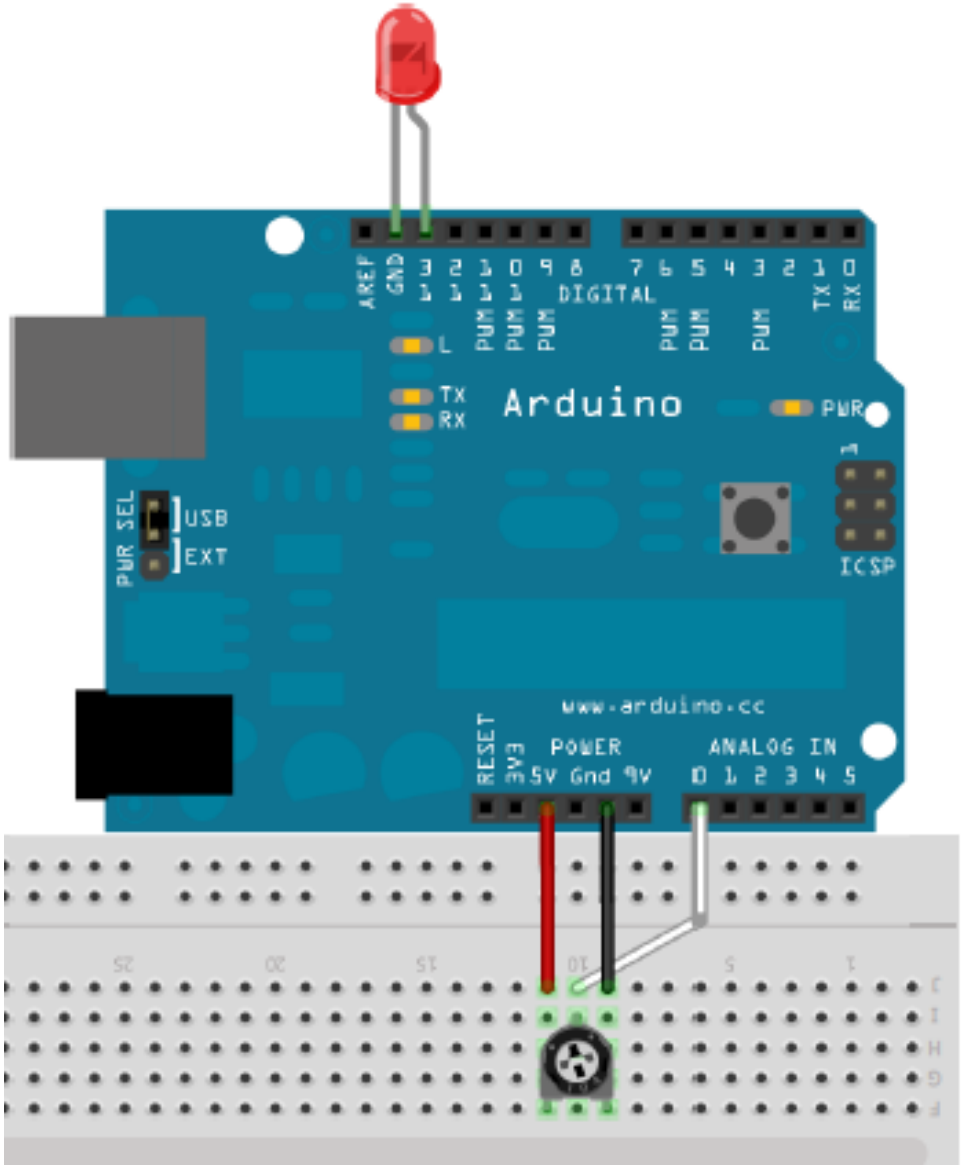


Time to Talk with Arduino board!

- Program to let Arduino say “Hello World!”
- Bit per second (baud rate)

- Note that this communication baud rate is independent of the upload process, which is fixed at 19200 bps.

Analog I/O: LED



Analog I/O: LED

```
int sensorPin = 0;                  // select the input pin for the potentiometer
int ledPin = 9;                     // select the PWM pin for the LED
int sensorValue = 0;                // variable to store the value coming from the sensor
Int outputData = 0;                 // Assigning integer value to the variable outputData

void setup() {
  Serial.begin(9600);
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // map it to the range of the analog out:
  outputData = map(sensorValue, 0, 1023, 0, 255);
  // turn the ledPin on according to potentiometer's modulation
  analogWrite(ledPin, outputData);
Serial.println("sensorValue = ");
Serial.print(sensorValue);
Serial.println("outputData = ");
Serial.print(outputData);
}
```

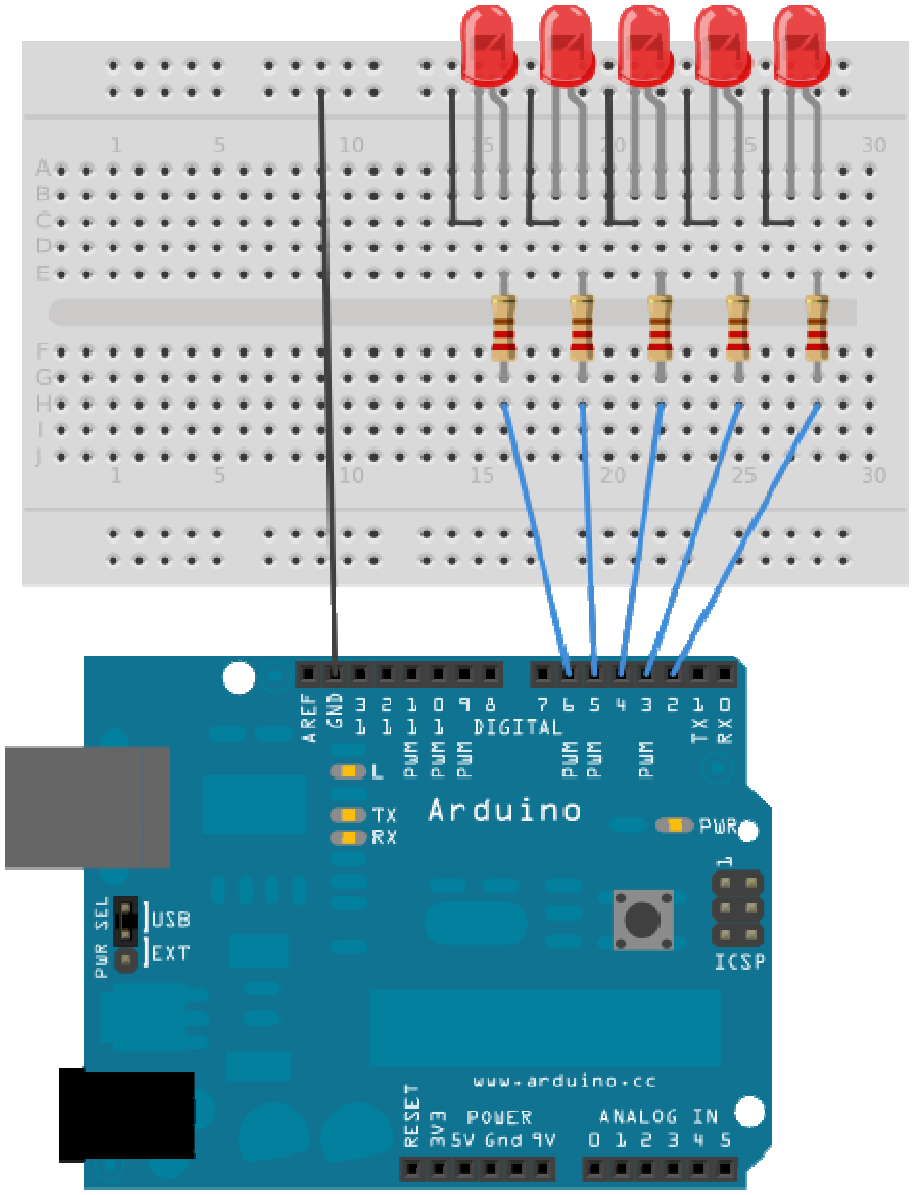
Serial Available

```
int incomingByte = 0;          // for incoming serial data

void setup() {
        Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
}

void loop() {

        // send data only when you receive data:
        if (Serial.available() > 0) {
                // read the incoming byte:
                incomingByte = Serial.read();

                // say what you got:
                Serial.print("I received: ");
                Serial.println(incomingByte);
        }
        else {
        Serial.println(“Nothing”);
        }
}
```

Serial Read

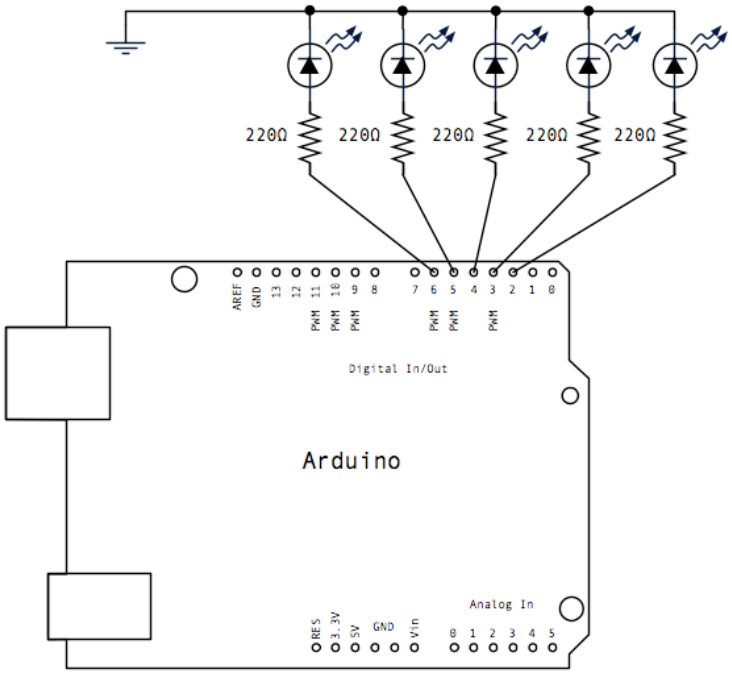
```
int incomingByte = 0;     // for incoming serial data

void setup() {
      Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
}

void loop() {

      // send data only when you receive data:
      if (Serial.available() > 0) {
                  // read the incoming byte:
                  incomingByte = Serial.read();

                  // say what you got:
                  Serial.print("I received: ");
                  Serial.println(incomingByte);
      }
}
```

Serial Read



Code:

http://arduino.cc/en/Tutorial/SwitchCase2



Serial Flush

- Flushes the buffer of incoming serial data.
- That is, any call to Serial.read() or Serial.available() will return only data received after all the most recent call to Serial.flush().

Serial Write

- Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print() function instead.