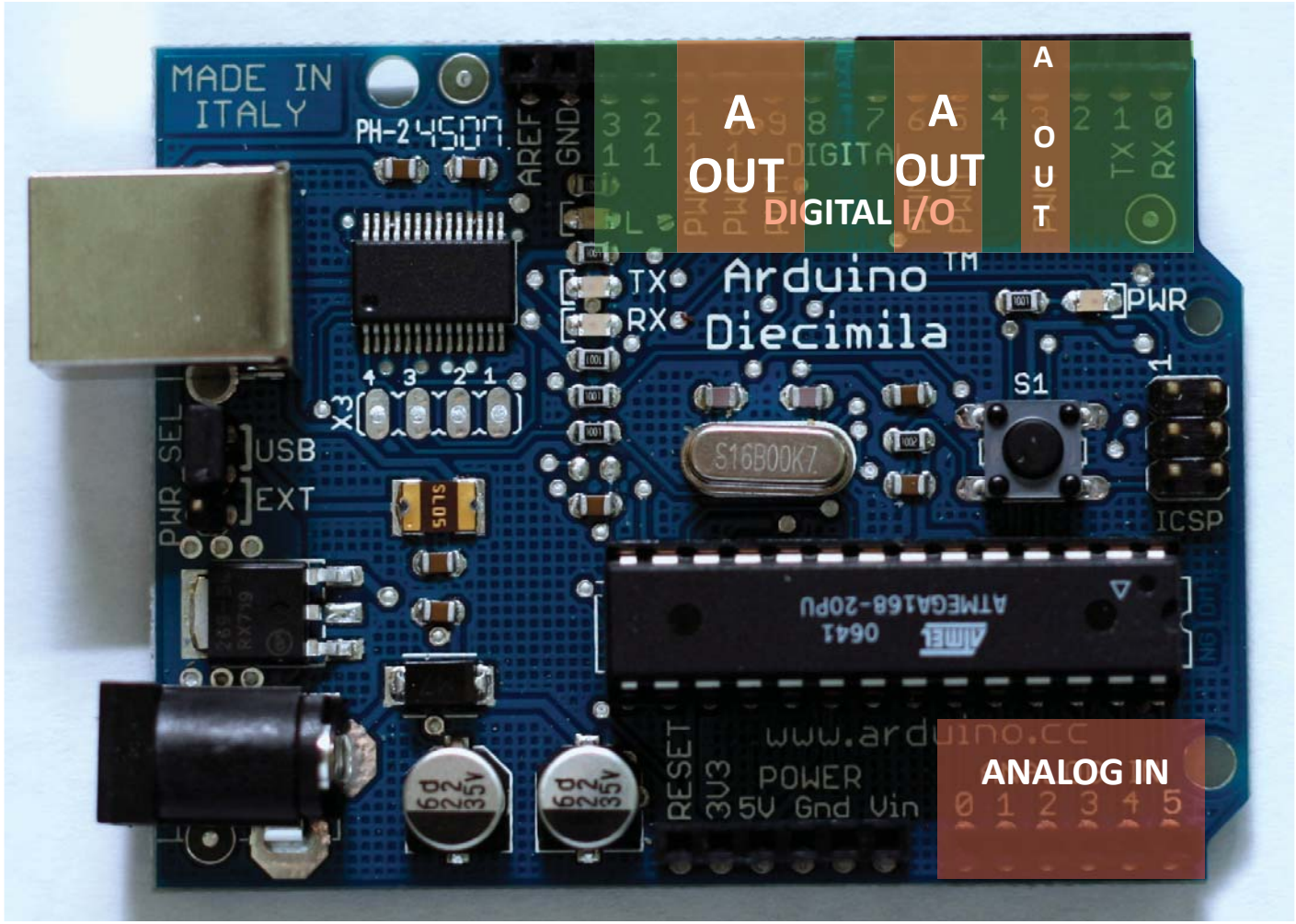
DIGITAL & ANALOG I/O_QUICK RECAP

DIGITAL AND ANALOG I/O ON THE BOARD



Digital In:
High / Low

Digital Out:
High / Low

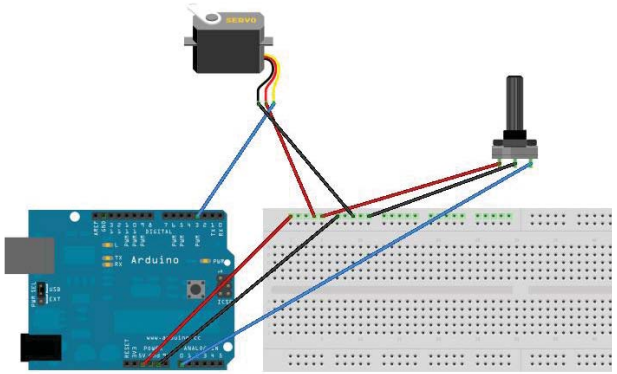
Analog In:
0 – 1023

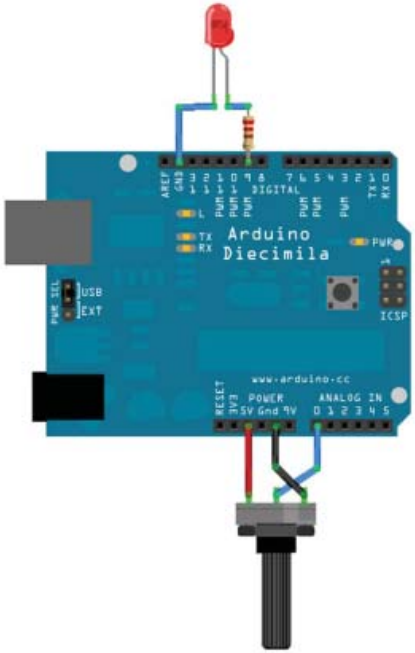
Analog Out:
0 – 255

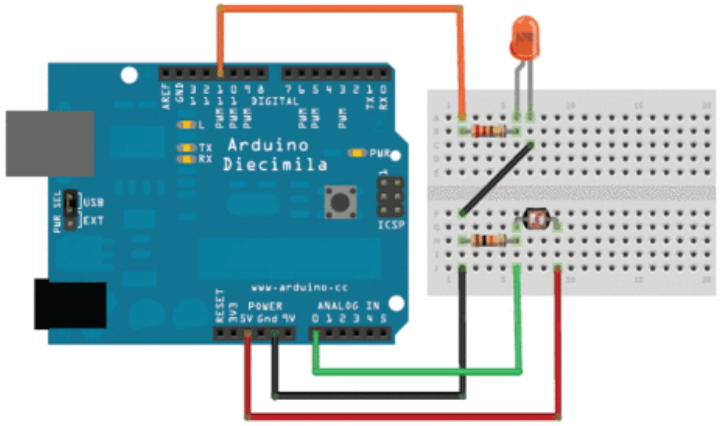
RECAP

ADC
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

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
```

LIBRARIES

#include <filename.h> //imports outside code into code
To load external libraries (set of functions)

http://www.arduino.cc/playground/ComponentLib/Servo

```
#include <Servo.h>

Servo myservo;          //create servo object to control a servo
int potpin = 0;                 // analog pin used to connect the potentiometer
int val = 0;            // 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, 180);  // 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
}
```

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

int variable; int variable = value;

boolean: true or false values
declaration boolean x = true; x = !x;
char: to assign Decimal ASCI character
declaration: char letter = 'A'; char letter = 65;
byte: 8-bit number
declaration: byte b = B10010; // (B10010 = 18 decimal)
int: integers (-32768 to 32767) (-2^15 to 2^15-1)
delaration: int degrees = -365;
unsigned int: integers (0 to 65535)
delaration: unsigned int days = 365;
long: extended variable size for integers (-2147483648 to 2147483647)
declaration: long val = -2147483647;
unsigned long: 0 to 2^32 -1
declaration: unsigned long val = 2147483647;
float /double: takes decimal values (-π to π with 38 decimal digits)
declaration: float pi = 3.14;
array: collection of variables accessed with index number
declaration: int pins[6];
int pins[] = {2, 4, 8, 3, 6};
int pins[6] = {2, 4, -8, 3, 2};
string: arrays of char
declaration: char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};
char Str4[ ] = "arduino";
void: used for function declaration
declaration: void name(_)

String(object): allows you to use and manipulate strings of text in more complex ways than character arrays do
String(object) declarations:

```
// using a constant String
String stringOne = "Hello String";
 // concatenating two strings
String stringOne =  String(stringTwo + "
with more");
// using a constant integer
String stringOne =  String(13);
// using an int and a base
String stringOne =  String(analogRead(0),
DEC);
// using an int and a base (hexadecimal)
String stringOne =  String(45, HEX);
// using an int and a base (binary)
String stringOne =  String(255, BIN);
// using a long and a base
String stringOne =  String(millis(), DEC);
```

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	SIX
^C	3	03	♥	EIX
^D	4	04	♦	EOI
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HI
^J	10	0A	◙	LF
^K	11	0B	♂	VI
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	SLE
^Q	17	11	◄	CS1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^T	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	EIB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SIB
^[	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

STRUCTURE

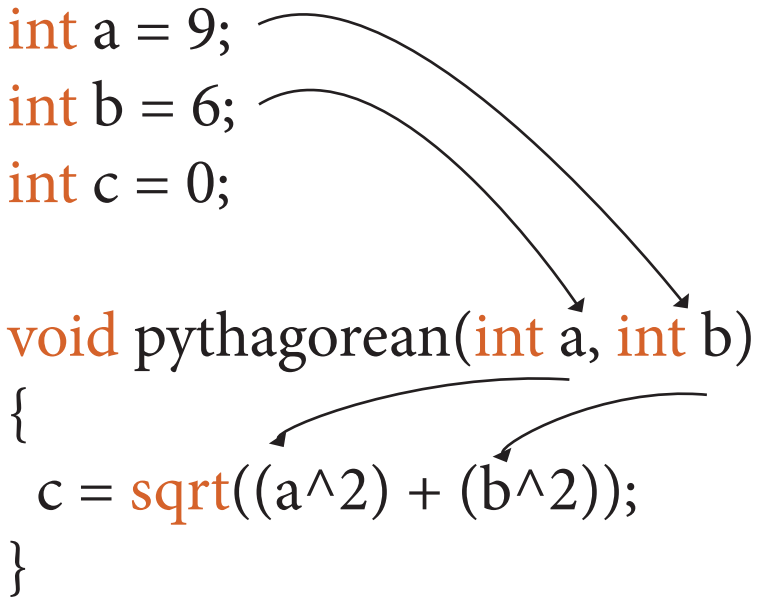
```
int buttonState =0;
unsigned long count = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
    /*buttonState = digitalRead(2);
   while(buttonState == HIGH)            //infinite loop
   {
      count = (count+1);
   }*/
   while(digitalRead(2) == HIGH)
   {
      count = (count+1);
     if((count/1000) > 600)        //stop counting when count reach 600
     {
        break;
     }
   }
   if(count>0)
   {
    Serial.println(count/1000); //convert milliseconds to seconds
   }
   count = 0;

}
```

STRUCTURE

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)

CUSTOM FUCTIONS

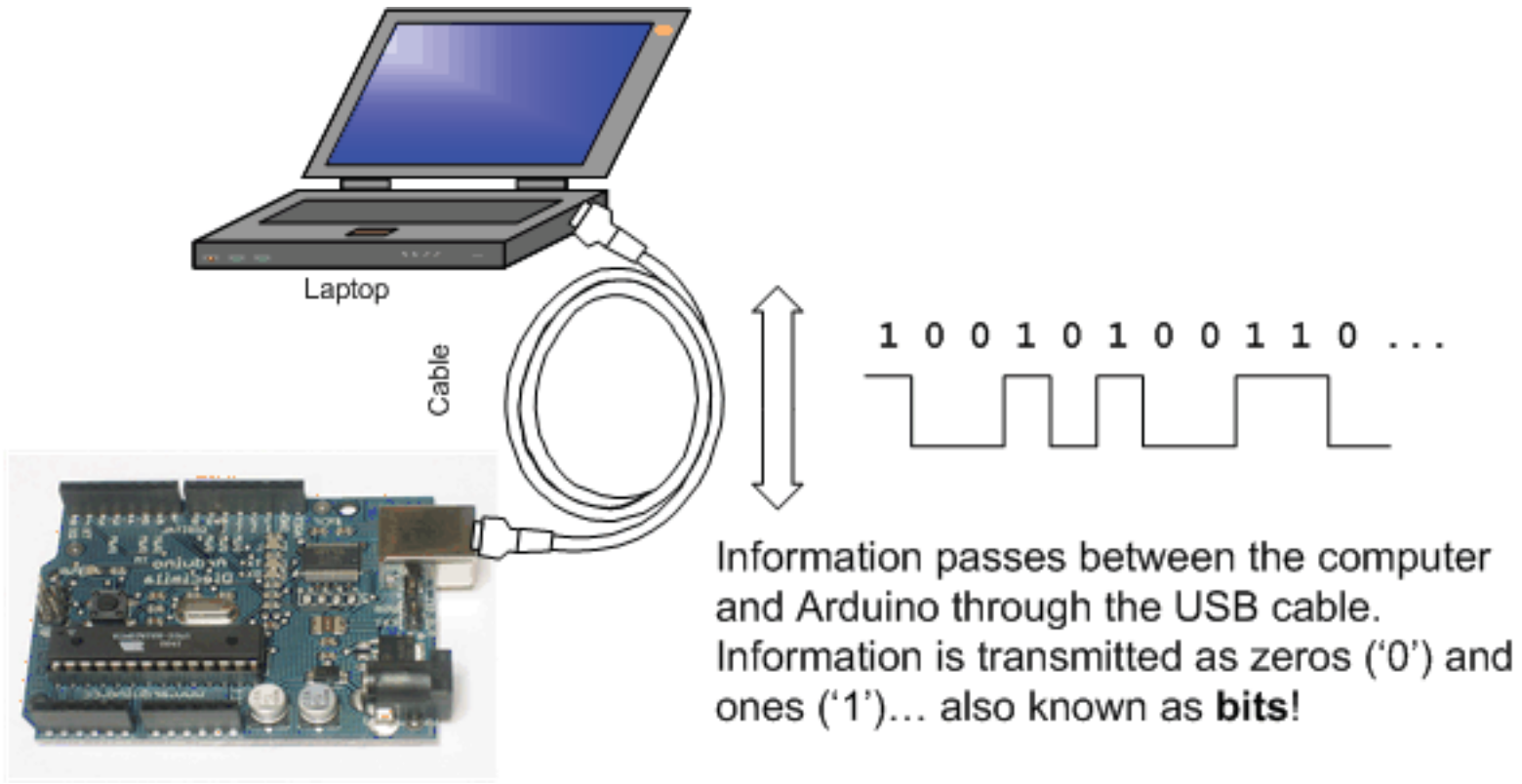


SERIAL COMMUNICATION

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

Communication done one after the other.

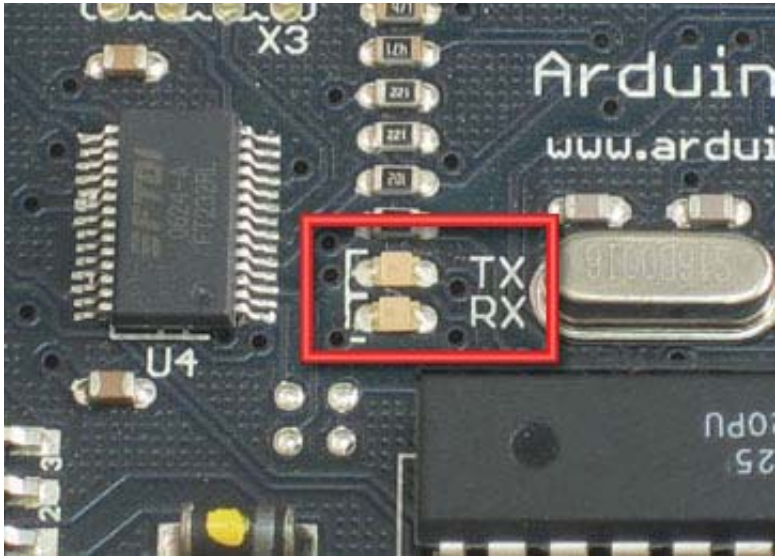


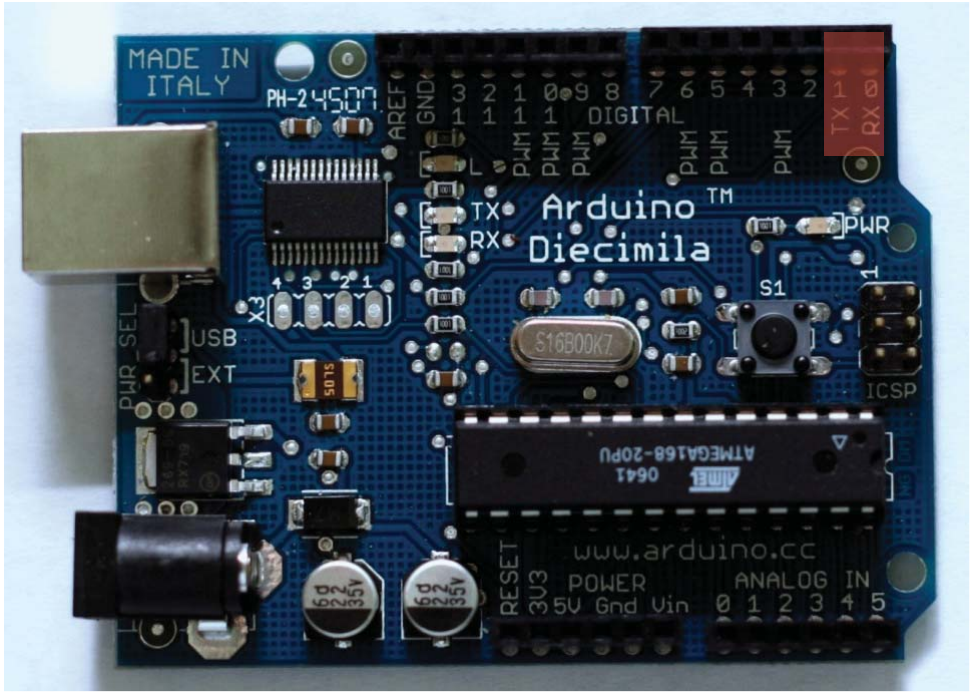
01100010011010010111010001110011001011000010
00000110001001111001011101000110010101110011

- 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





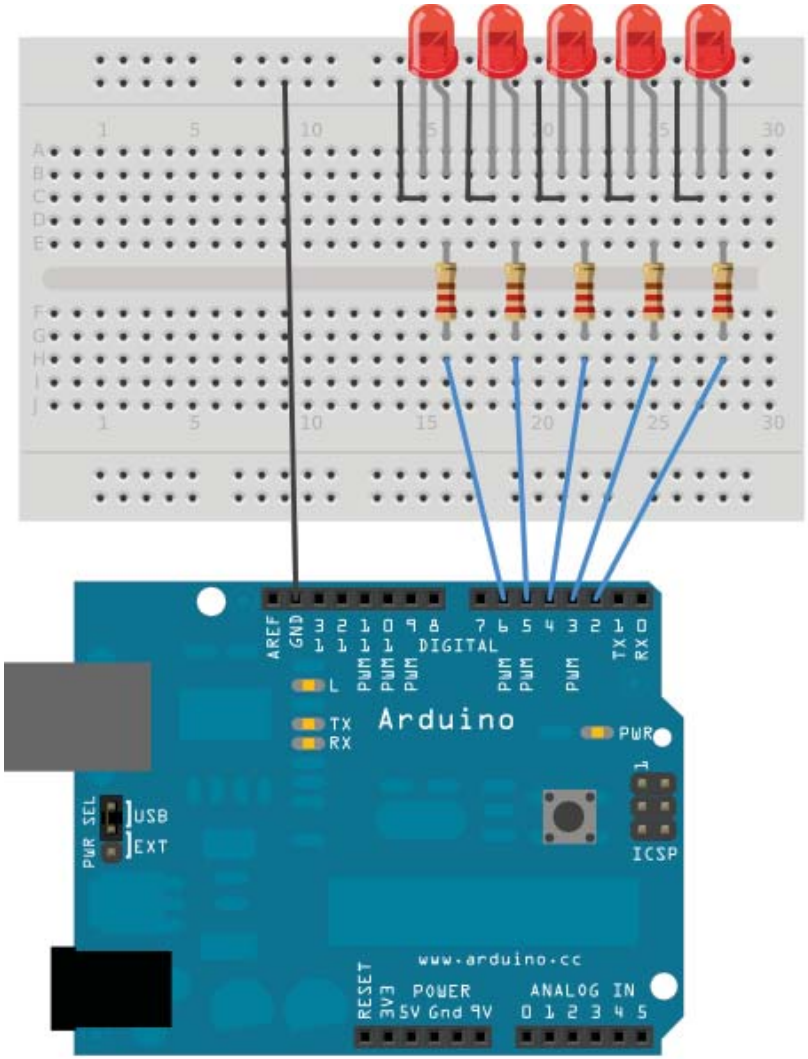
Time to Talk with Arduino board!

- 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.

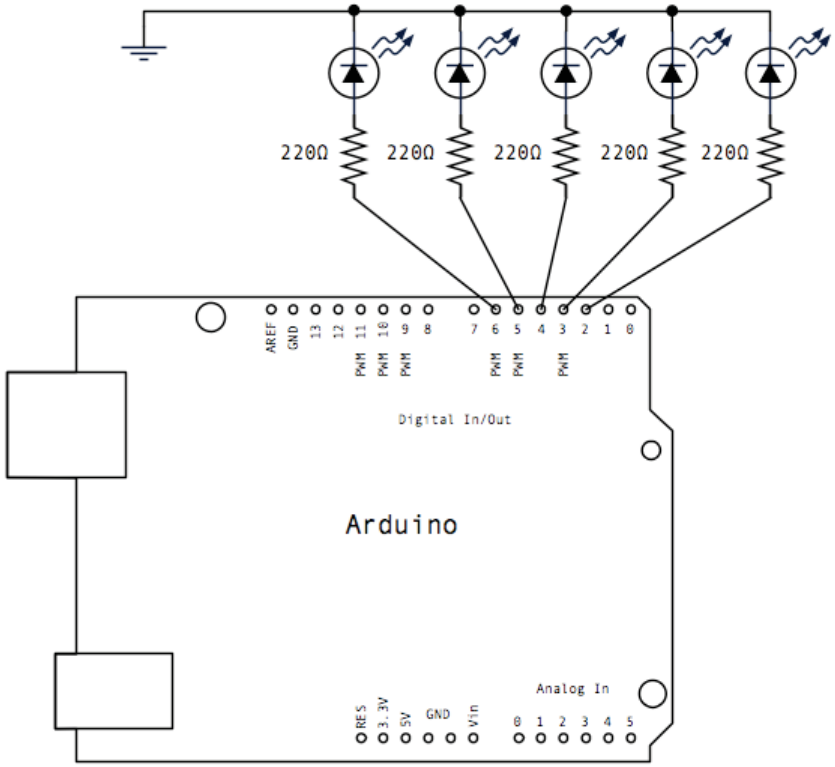
SERIAL AVAILABLE

```
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_switch case



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



SERIAL READ_switch case

EXAMPLES > CONTROL> SWITCHCASE2

```
void setup()
{
        // initialize serial communication:
        Serial.begin(9600);
        // initialize the LED pins:
        for (int thisPin = 2; thisPin < 7; thisPin++)
        {
                pinMode(thisPin, OUTPUT);
        }
}
```

```
void loop()
{
        // read the sensor:
        if (Serial.available() > 0)
        {
                int inByte = Serial.read();
                /*do something different depending on the character received.
                The switch statement expects single number values for each case;
                in this exmaple, though, you're using single quotes to tell
                the controller to get the ASCII value for the character.  For
                example 'a' = 97, 'b' = 98, and so forth:*/
                switch (inByte) {
                case 'a':
                        digitalWrite(2, HIGH);
                        break;
                case 'b':
                        digitalWrite(3, HIGH);
                        break;
                case 'c':
                        digitalWrite(4, HIGH);
                        break;
                case 'd':
                        digitalWrite(5, HIGH);
                        break;
                case 'e':
                        digitalWrite(6, HIGH);
                break;
                default:
                        // turn all the LEDs off:
                        for (int thisPin = 2; thisPin < 7; thisPin++) {
                        digitalWrite(thisPin, LOW);
                        }
                }
        }
}
```

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().

INTERFACING WITH OTHER SOFTWARE_serial in

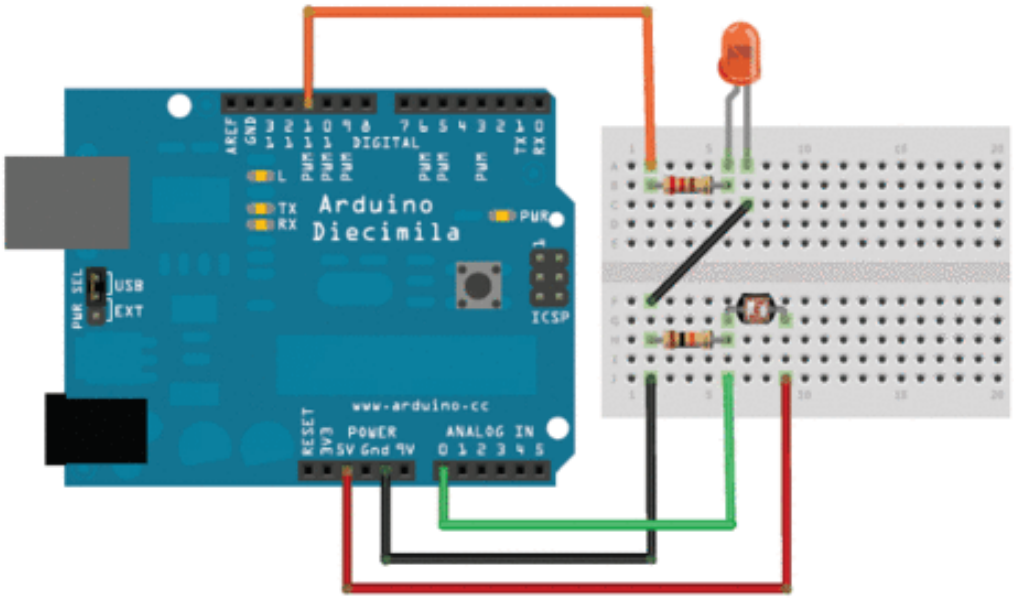
- Processing, PureDate(PD), Max/MSP, Flash

http://www.arduino.cc/playground/Main/InterfacingWithSoftware

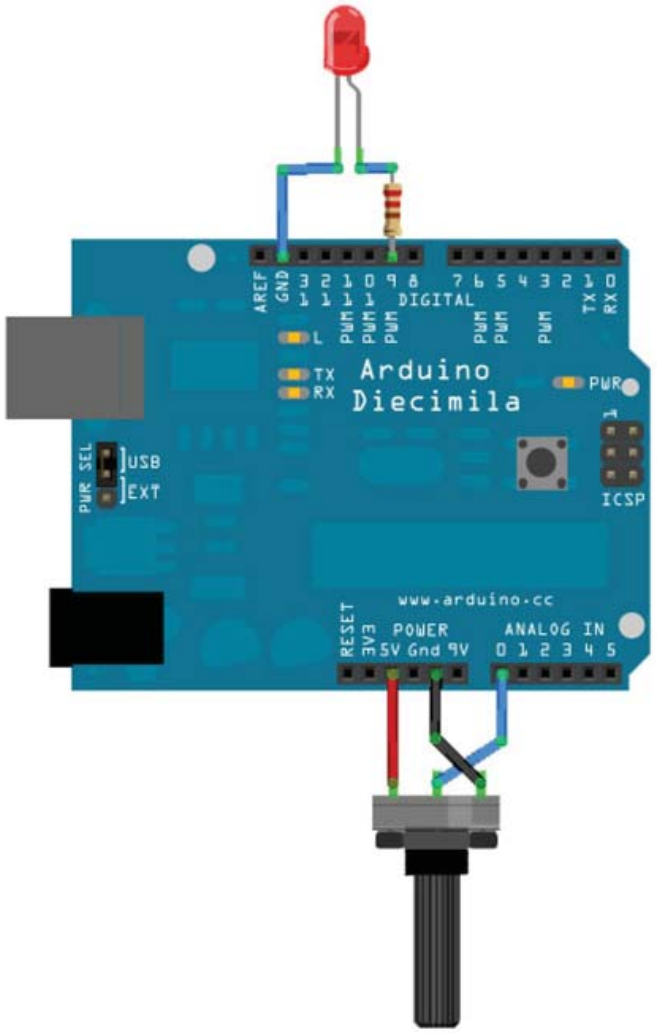
ARDUINO_SERIAL WRITE

- 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.

Photocell:



Potentiometer:



PROCESSING_serial read:image

EXAMPLES > LIBRARY> SERIAL> SIMPLE READ

```
import processing.serial.*;
Serial myPort;  // Create object from Serial class
int val;      // Data received from the serial port

void setup()
{
  size(200, 200);
  // I know that the first port in the serial list on my mac
  // is always my  FTDI adaptor, so I open Serial.list()[0].
  // On Windows machines, this generally opens COM1.
  // Open whatever port is the one you're using.
  String portName = Serial.list()[0];
  myPort = new Serial(this, portName, 9600);
}
void draw()
{
  if ( myPort.available() > 0)
  {  // If data is available,
    val = myPort.read();         // read it and store it in val
  }
  background(255);             // Set background to white
  if (val == 0) {              // If the serial value is 0,
    fill(0);                   // set fill to black
  }
  else {                       // If the serial value is not 0,
    fill(204);                 // set fill to light gray
  }
  rect(50, 50, 100, 100);
}
```

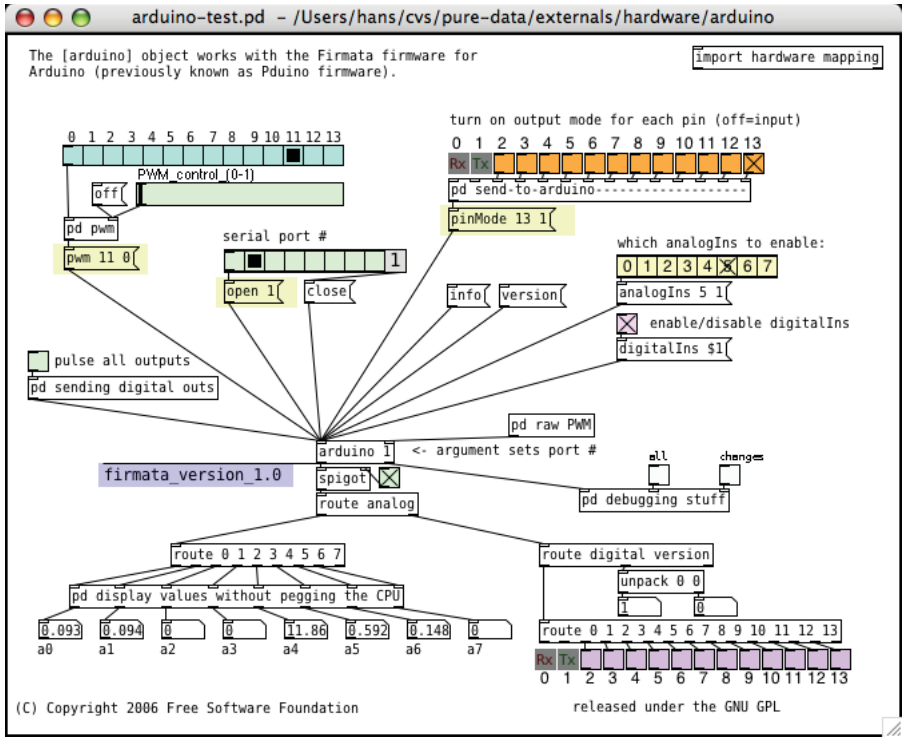
PROCESSING_arduino library:sound

ARDUINO> EXAMPLES> FIRMATA> UPLOAD STANDARD FIRMATA
PROCESSING> SKETCHBOOK> ARDUINO> EXAMPLES> ARDUINO_INPUT
PROCESSING> EXAMPLES> LIBRARY> MINIM(SOUND)

```
//import libraries
import processing.serial.*;        //serial library
import cc.arduino.*;               //arduino library
import ddf.minim.*;                //minim library_audio
import ddf.minim.signals.*;
//declare objects and varibles
Arduino arduino;
Minim minim;
AudioOutput out;
SineWave sine;

int val = 0;
void setup()
{
 size(512, 200, P2D);
 //println(Arduino.list());  to find com port to use
 //declare arduino object
 arduino = new Arduino(this, Arduino.list()[0], 57600);
 minim = new Minim(this); //declare minim object
 out = minim.getLineOut(Minim.STEREO);
 sine = new SineWave(440, 0.5, out.sampleRate());
 sine.portamento(200);
 out.addSignal(sine);

}
```

```
void draw()
{
 background(0);
 stroke(255);
  //draw the waveforms
  for(int i = 0; i < out.bufferSize() - 1; i++)
  float x1 = map(i, 0, out.bufferSize(), 0, width);
  float x2  = map(i+1, 0, out.bufferSize(), 0, width);
  line(x1, 50 + out.left.get(i)*50, x2, 50 + out.left.get(i+1)*50);
  line(x1, 150 + out.right.get(i)*50, x2, 150 + out.right.get(i+1)*50);
 }

 val = arduino.analogRead(0);
 println(val);
 float freq = map(val, 0, 1023, 1500, 60);
 sine.setFreq(freq);
 float pan = map(val, 0, 1023, -1, 1);
 sine.setPan(pan);
}


void stop()
{
 out.close();
 minim.stop();
  super.stop();
}
```

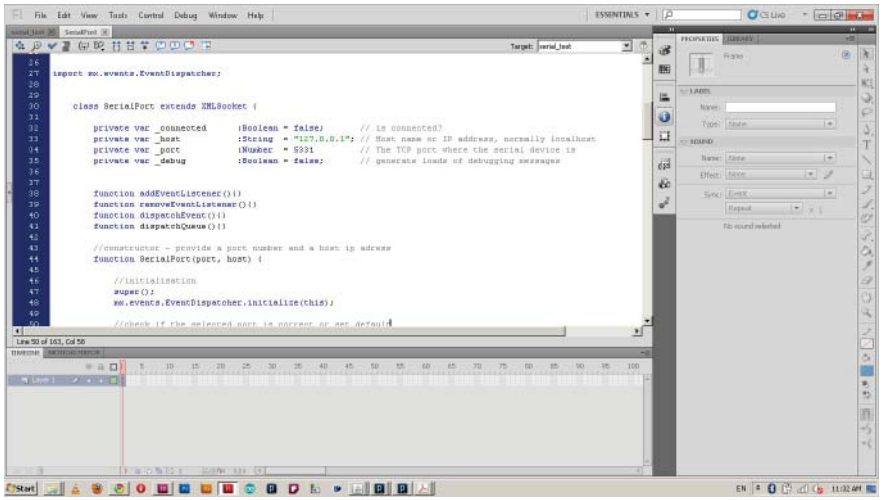
PureData(PD)



Max/MSP

```
max v2;
#N vpatcher 10 59 610 459;
#P user uslider 286 66 18 128 255 1 0 0;
#P window setfont "Sans Serif" 9.;
#P window linecount 1;
#P message 212 192 32 196617 print;
#P newex 286 217 71 196617 serial a 9600;
#P window linecount 2;
#P comment 316 148 100 196617 Slide the fader to dim the LED;
#P comment 69 192 125 196617 Click here to get a list of serial ports;
#P fasten 4 0 2 0 291 215 291 215;
#P fasten 3 0 2 0 217 212 291 212;
#P pop;
```

FLASH



LIBRARIES:
http://www.arduino.cc/playground/Main/InterfacingWithSoftware