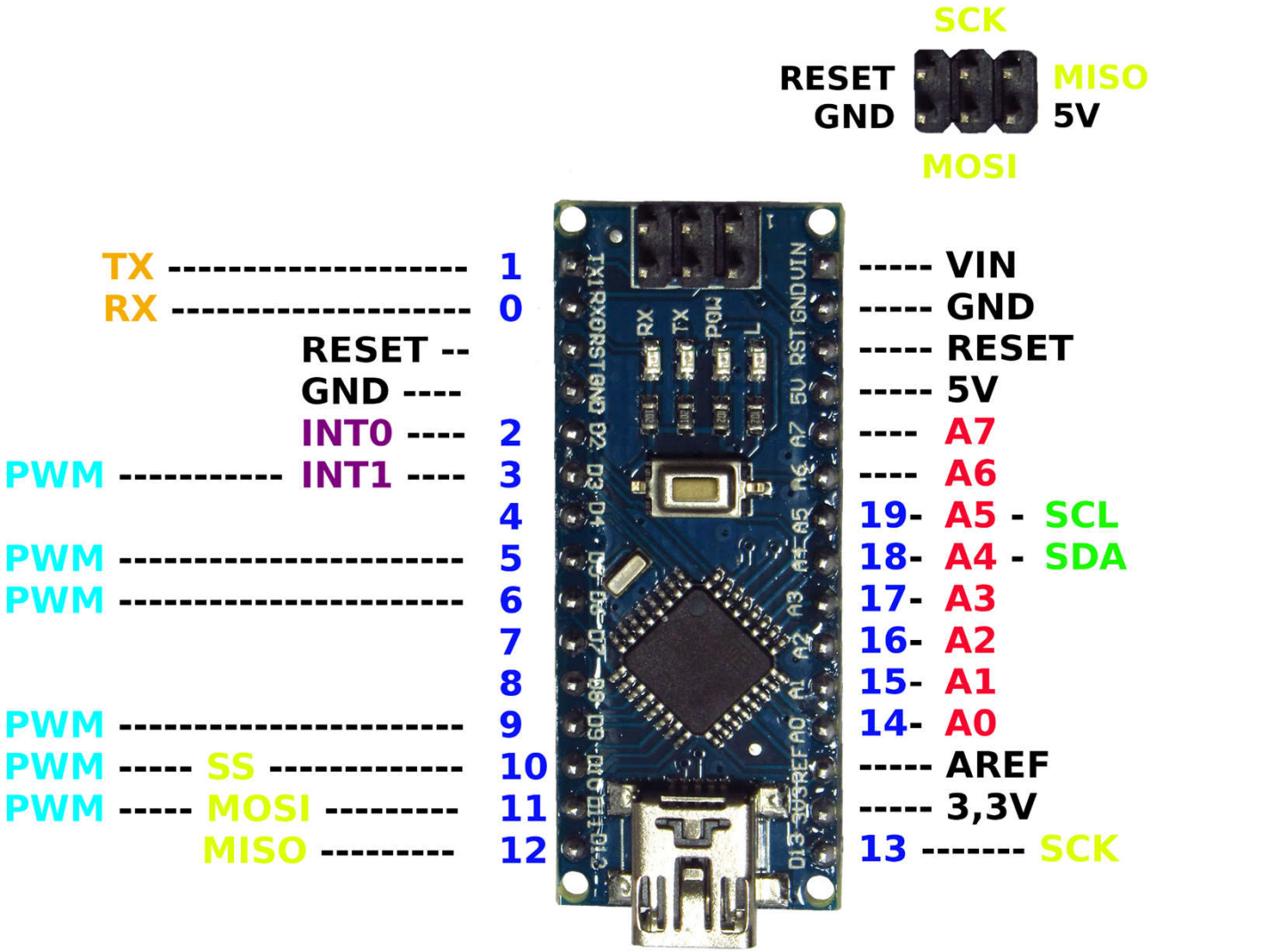
Intro to the Arduino

Topics:

- The Arduino
- Digital IO
- Analog IO
- Serial Communication

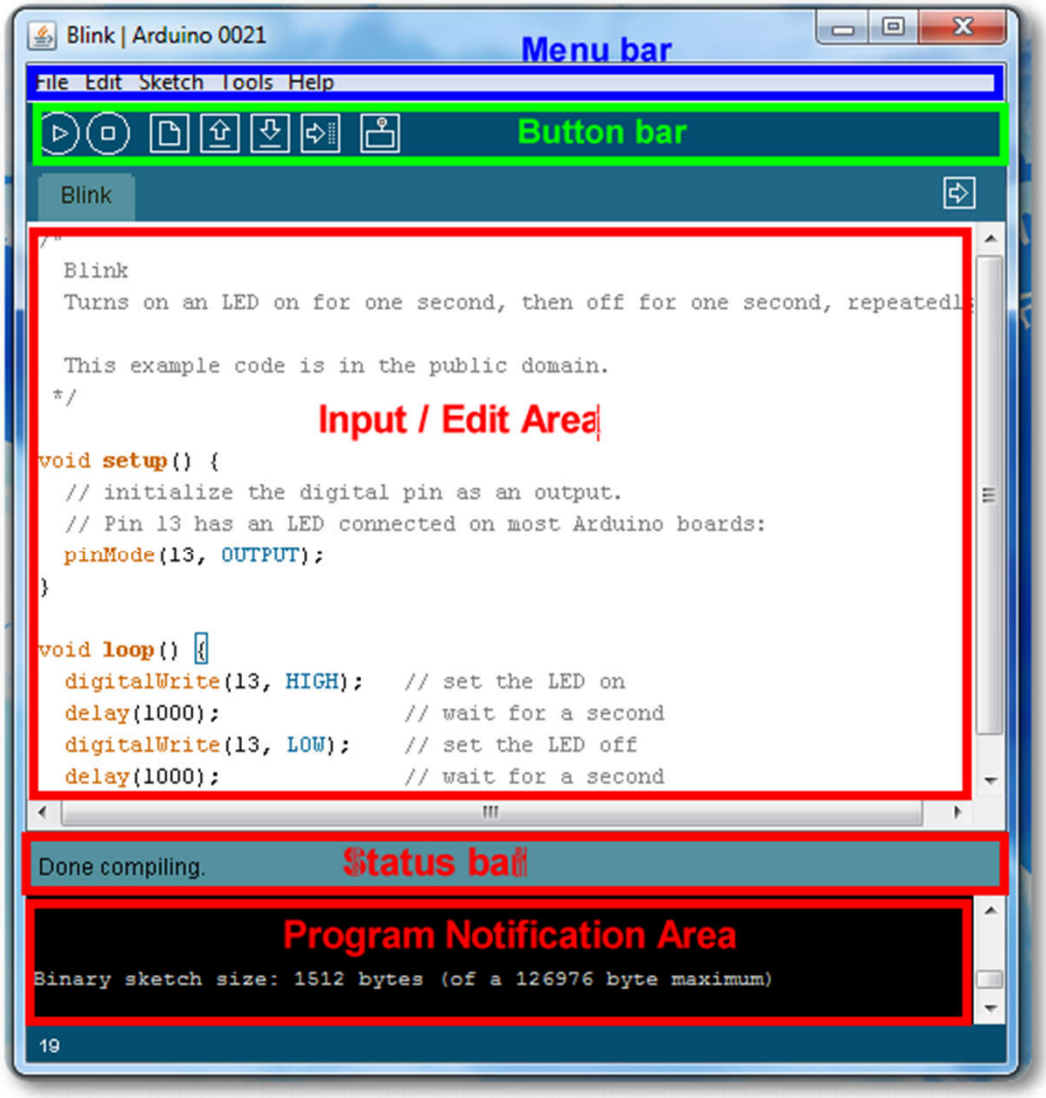
Topic 1: Meet Arduino nano



Getting Started

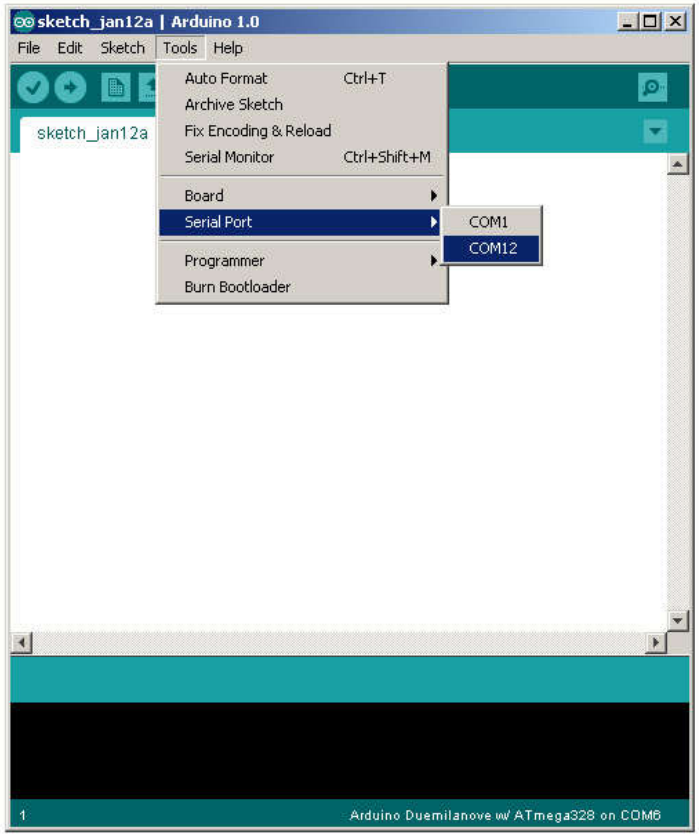
- Check out: http://arduino.cc/en/Guide/HomePage
  1. Download & install the Arduino environment (IDE) **(not needed in lab)**
  2. Connect the board to your computer via the USB cable
  3. If needed, install the drivers **(not needed in lab)**
  4. Launch the Arduino IDE
  5. Select your board
  6. Select your serial port
  7. Open the blink example
  8. Upload the program

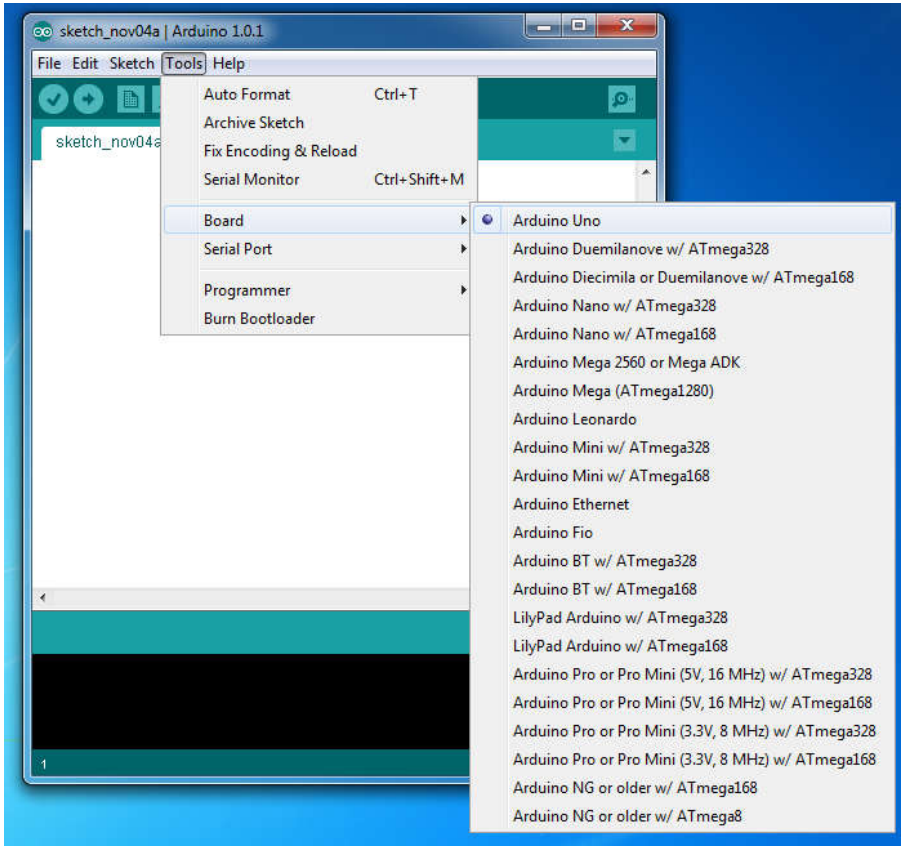
Arduino IDE



See: http://arduino.cc/en/Guide/Environment for more information

Select Serial Port and Board

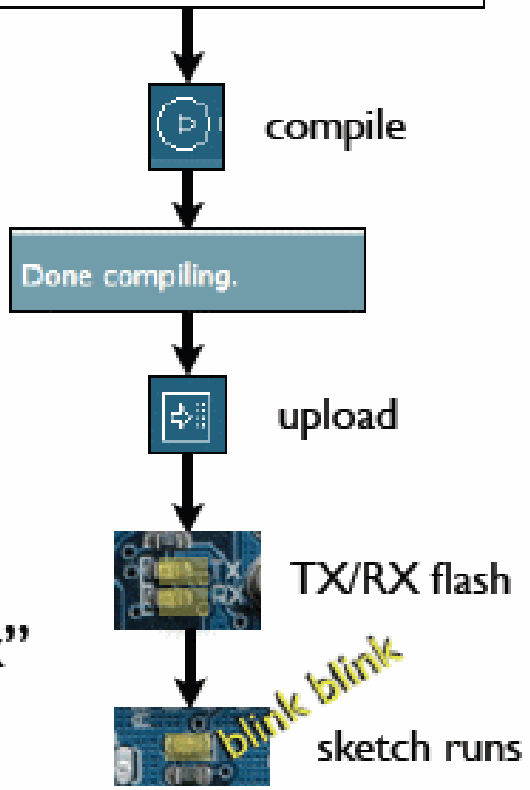




Using Arduino

- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

```
void setup() {
  pinMode(ledPin, OUTPUT);      // sets t
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets t
  delay(1000);                  // waits
  digitalWrite(ledPin, LOW);    // sets t
  delay(1000);                  // waits
}
```



Try it out with the "Blink" sketch!

Load "File/Sketchbook/Examples/Digital/Blink"

Input/Output

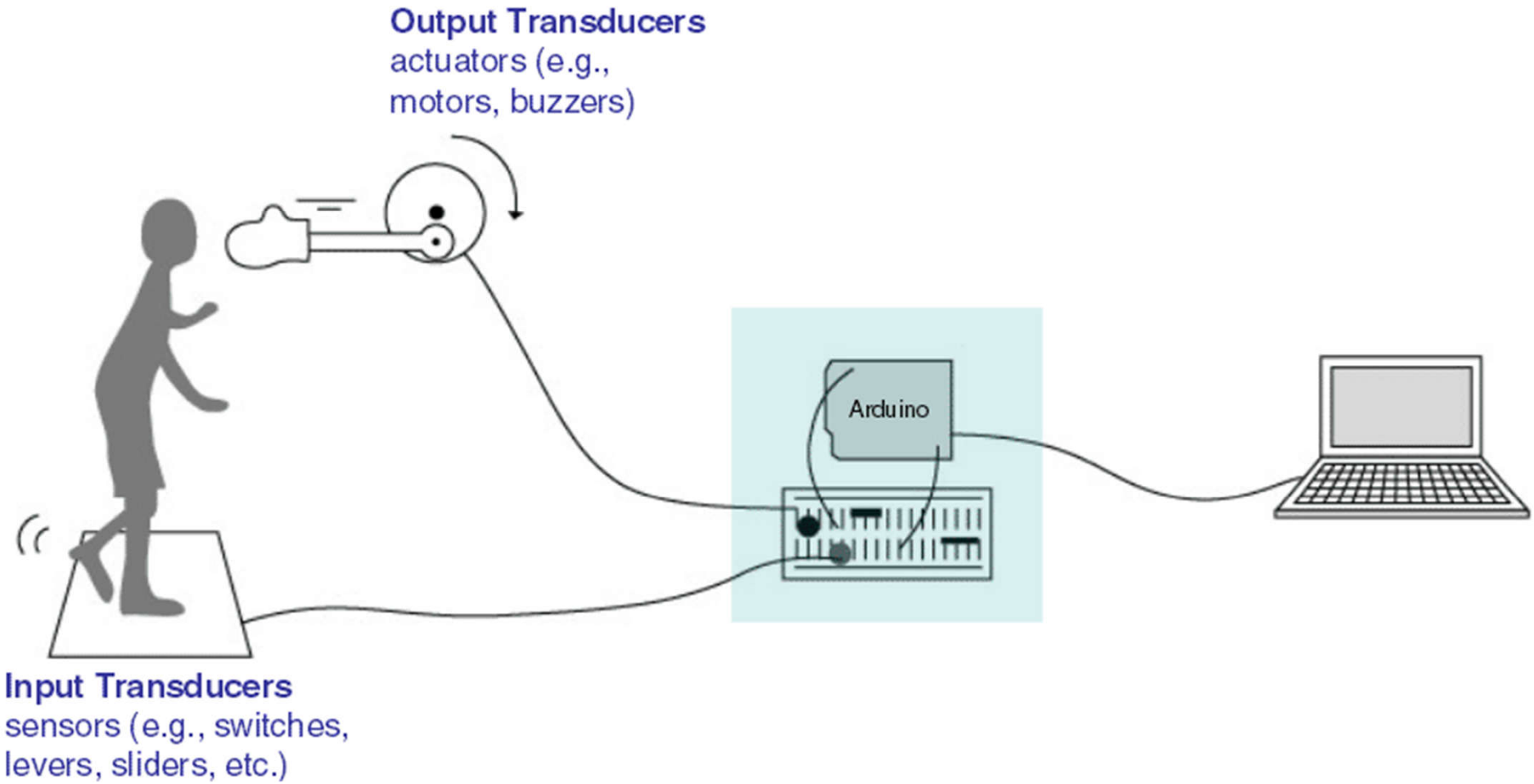
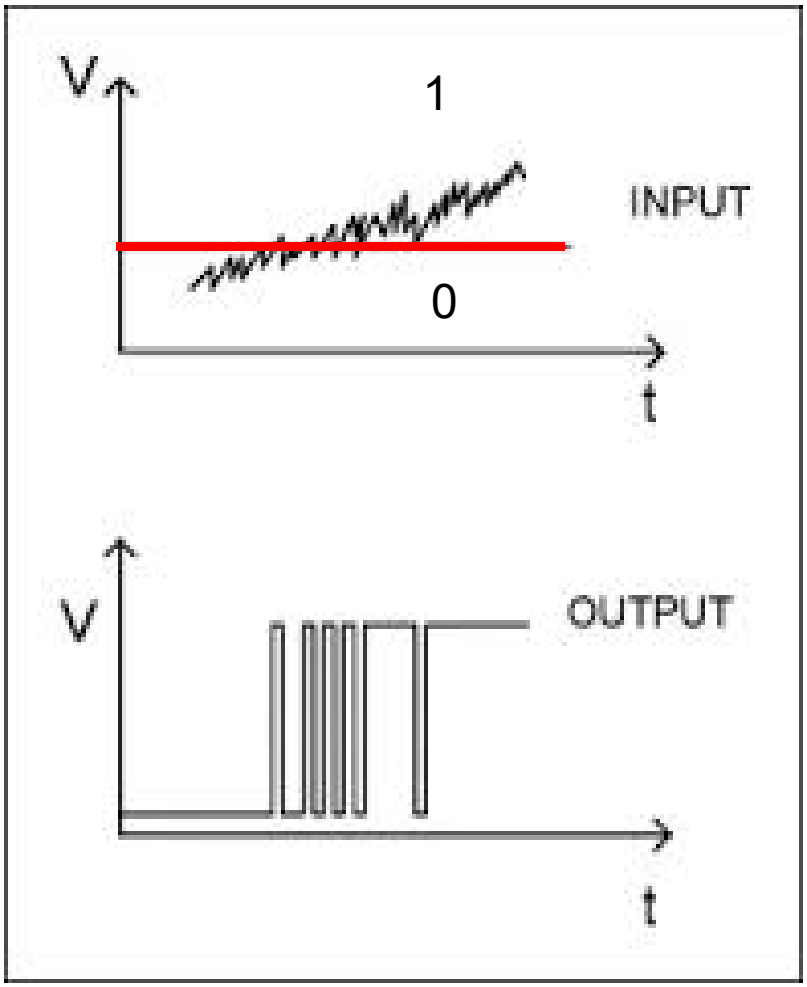
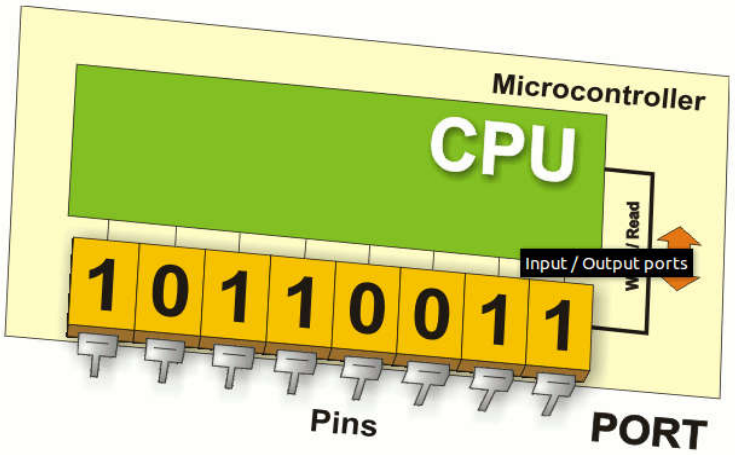


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

Topic 2: Digital Input/Output

- Digital IO is binary valued—it's either *on* or *off*, 1 or 0
- Internally, all microprocessors are digital, **why**?





www.mikroe.com/chapters/view/1

Arduino Digital I/0

`pinMode(`*`pin, mode`*`)`

Sets pin to either `INPUT` or `OUTPUT`

`digitalRead(`*`pin`*`)`

Reads `HIGH` or `LOW` from a pin

`digitalWrite(`*`pin, value`*`)`

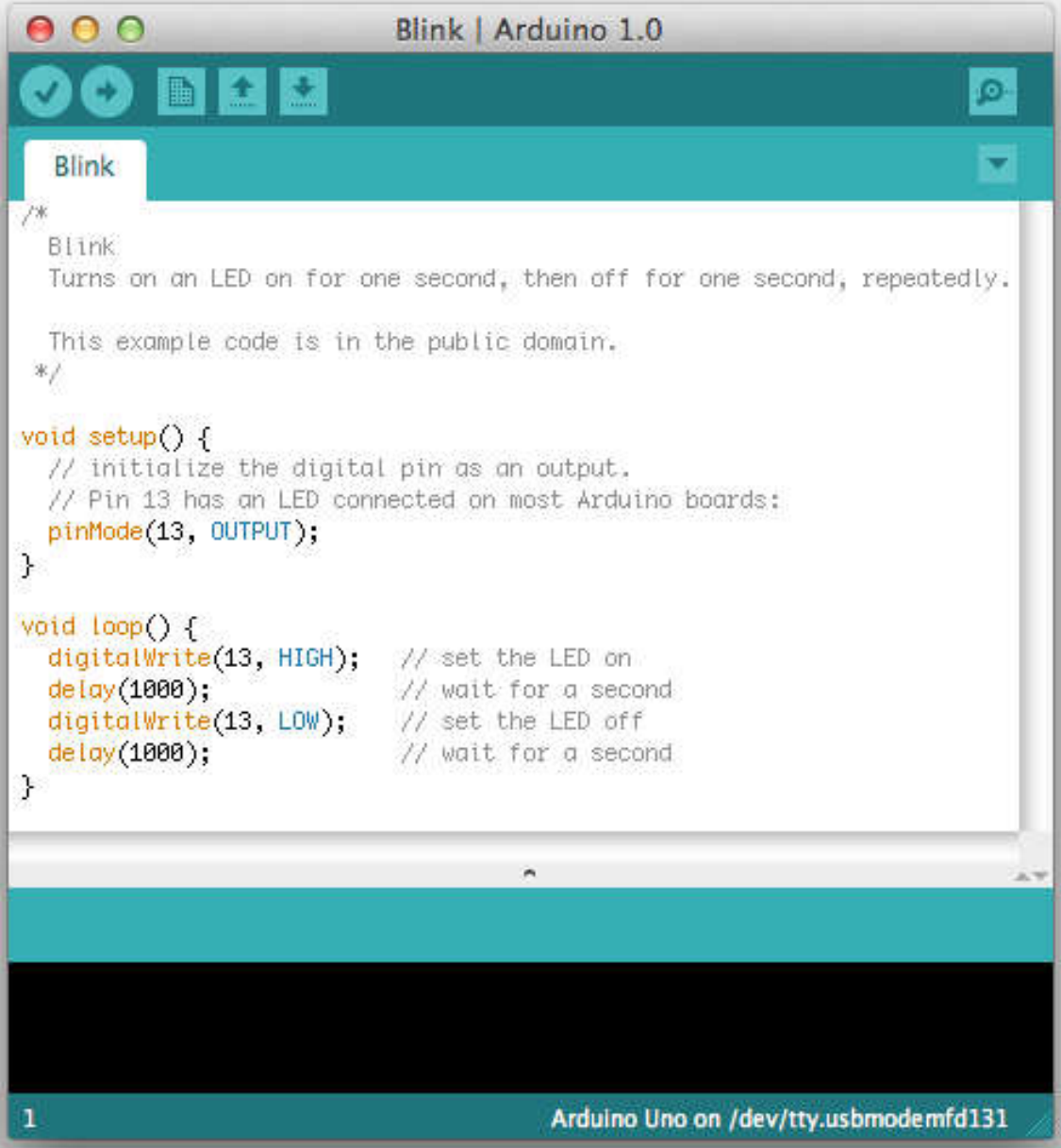
Writes `HIGH` or `LOW` to a pin

Electronic stuff

Output pins can provide 40 mA of current

Writing `HIGH` to an input pin installs a 20KΩ pullup

Our First Program



IO Pins

Two states (binary signal) vs. multiple states (continuous signal)

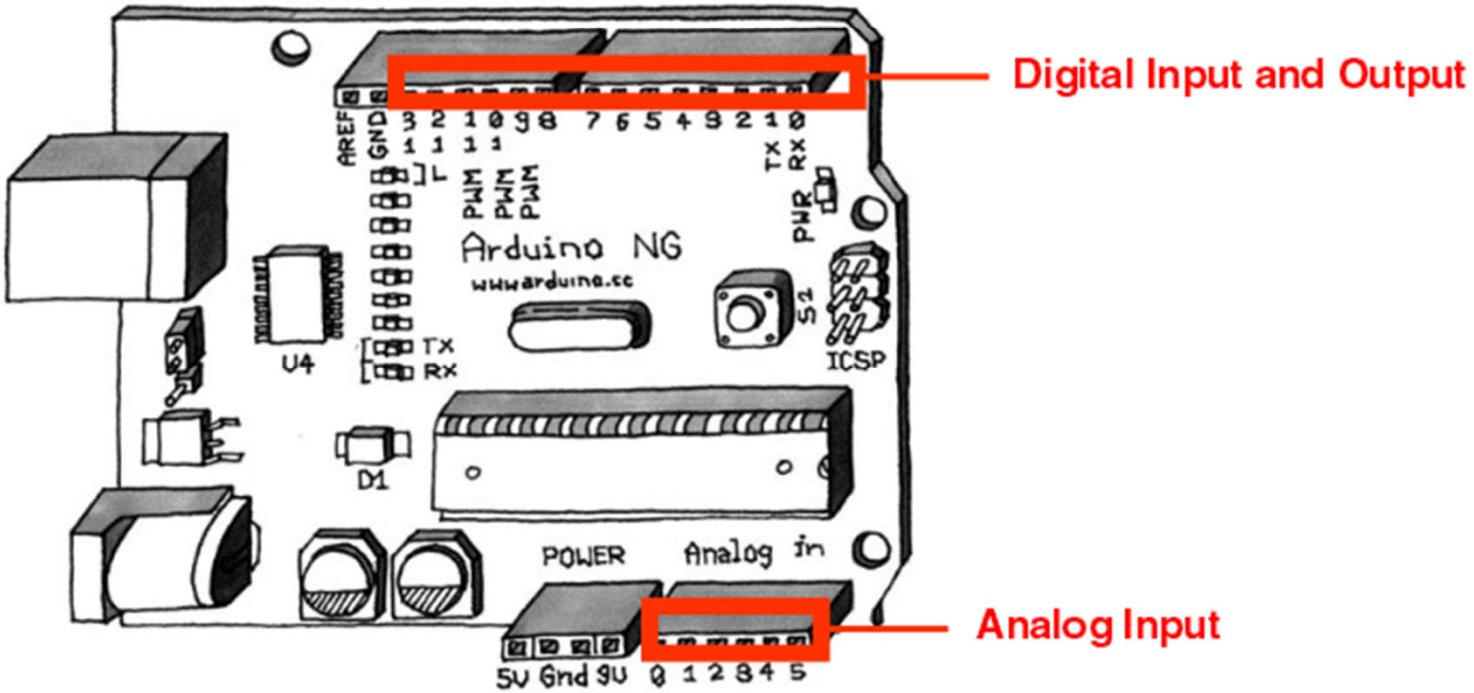
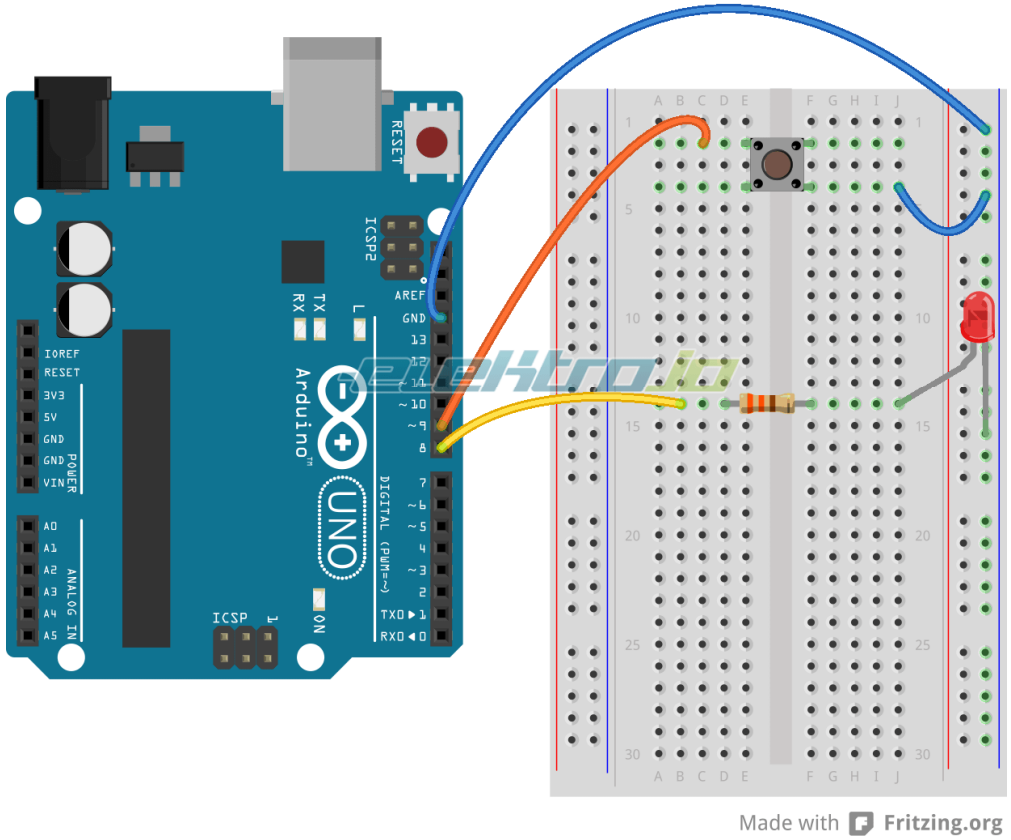


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

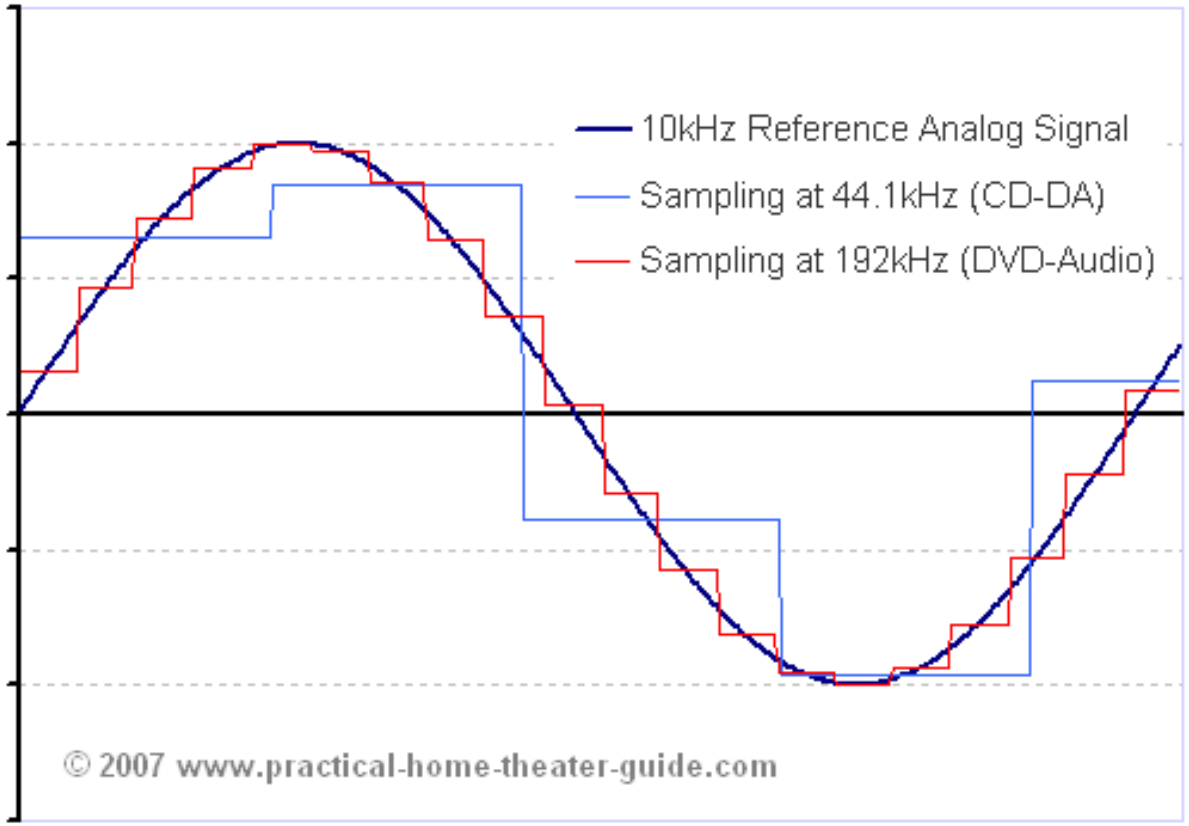
In-class Exercise 1: Digital IO



- Use a push-button to turn ON/Off LED

Topic 3: Analog Input

- Think about music stored on a CD---an analog signal captured on digital media
  - Sample rate
  - Word length



Arduino Analog Input

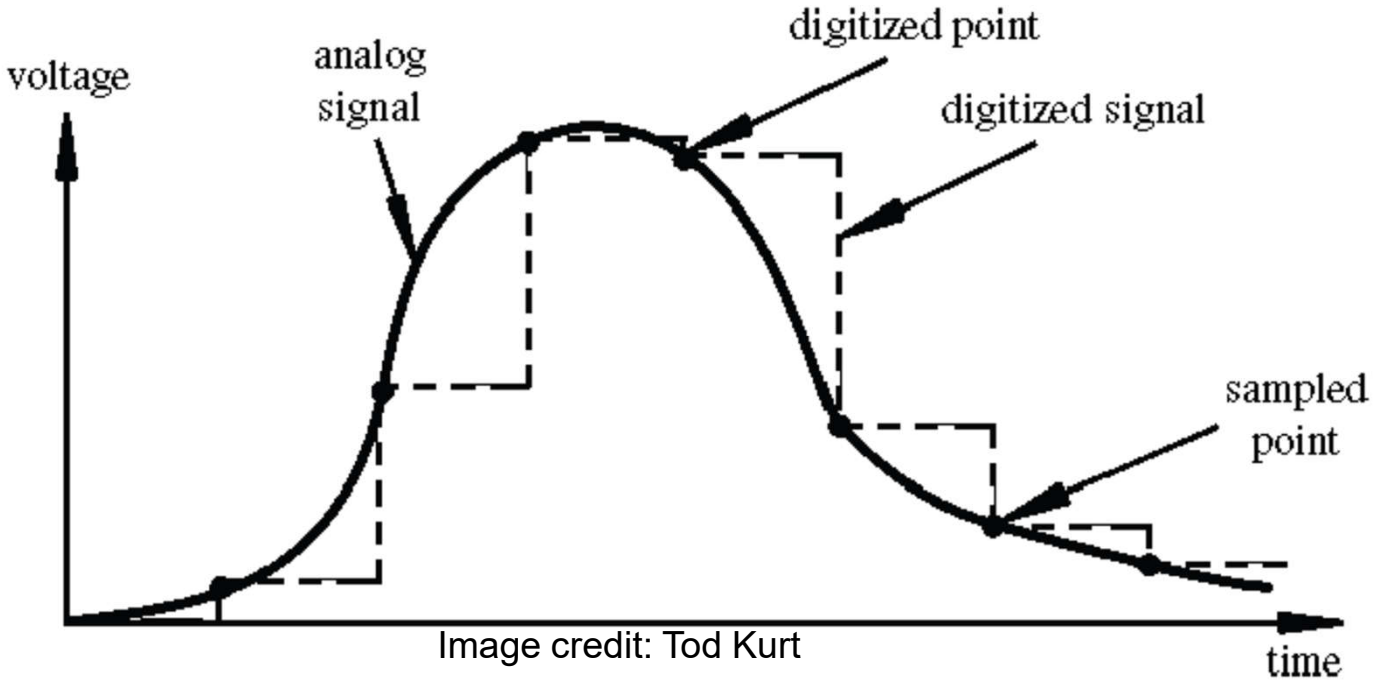
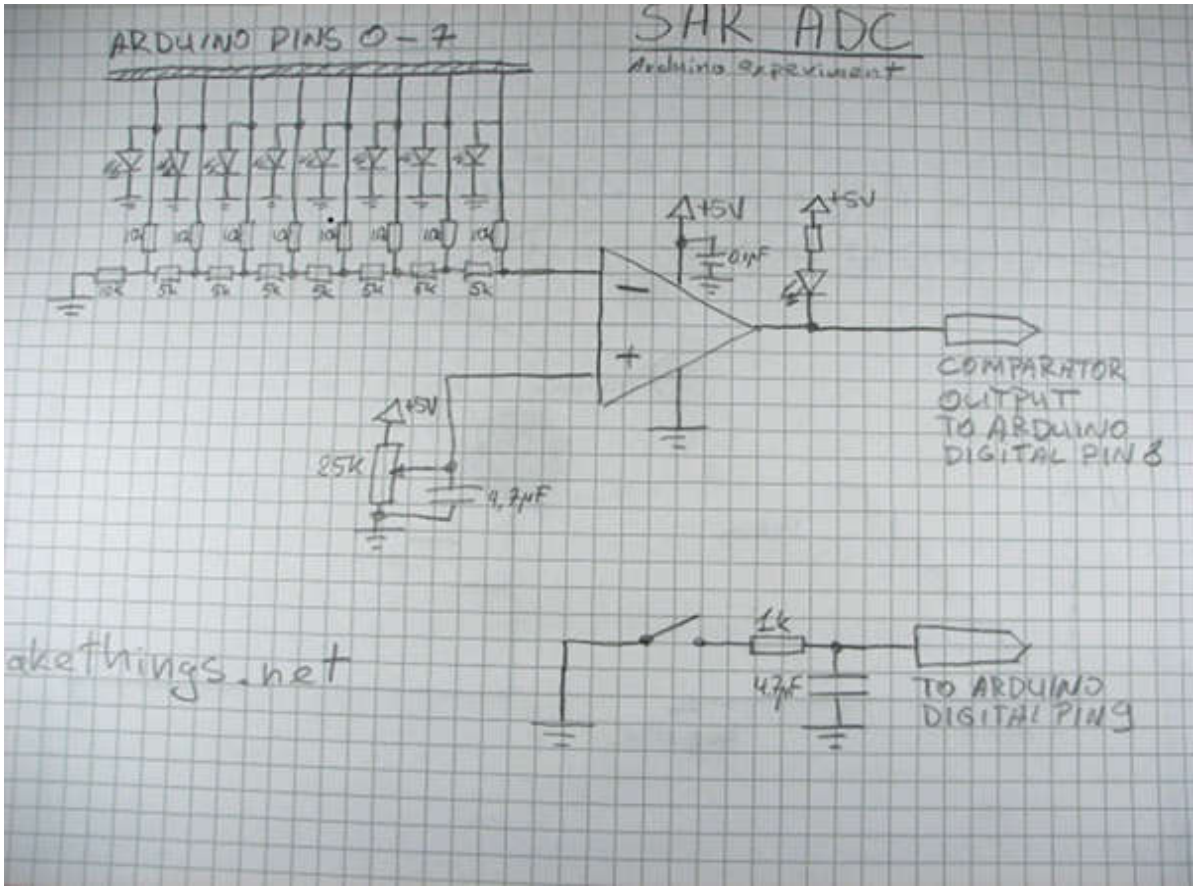


Image credit: Tod Kurt

- *Resolution*: the number of different voltage levels (i.e., *states*) used to discretize an input signal
- Resolution values range from 256 states (8 bits) to 4,294,967,296 states (32 bits)
- The Arduino uses 1024 states (10 bits)
- Smallest measurable voltage change is 5V/1024 or 4.8 mV
- Maximum sample rate is 10,000 times a second

How does ADC work?



- How does ADC work
- Excel Demonstration

Topic 3: Analog Output

- Can a digital devise produce analog output?

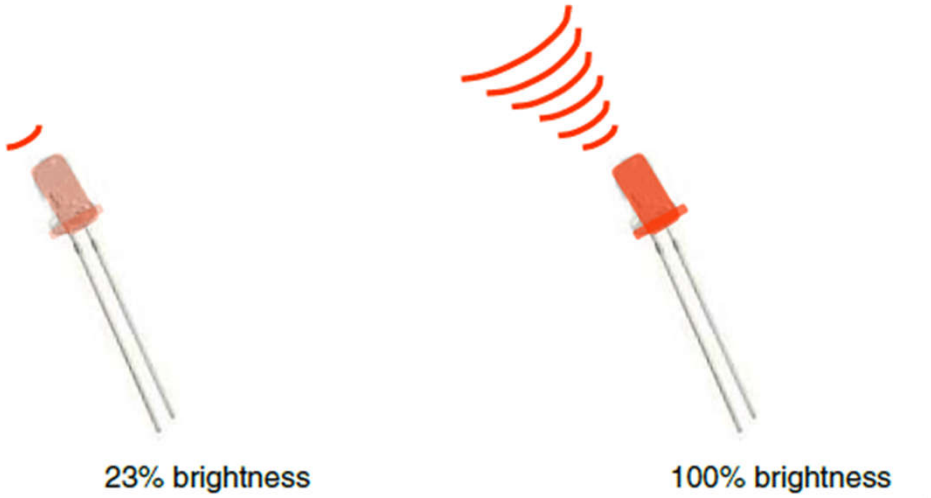


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

- Analog output can be simulated using pulse width modulation (PWM)

Pulse Width Modulation

- Can't use digital pins to directly supply say 2.5V, but can pulse the output on and off really fast to produce the same effect
- The on-off pulsing happens so quickly, the connected output device "sees" the result as a reduction in the voltage

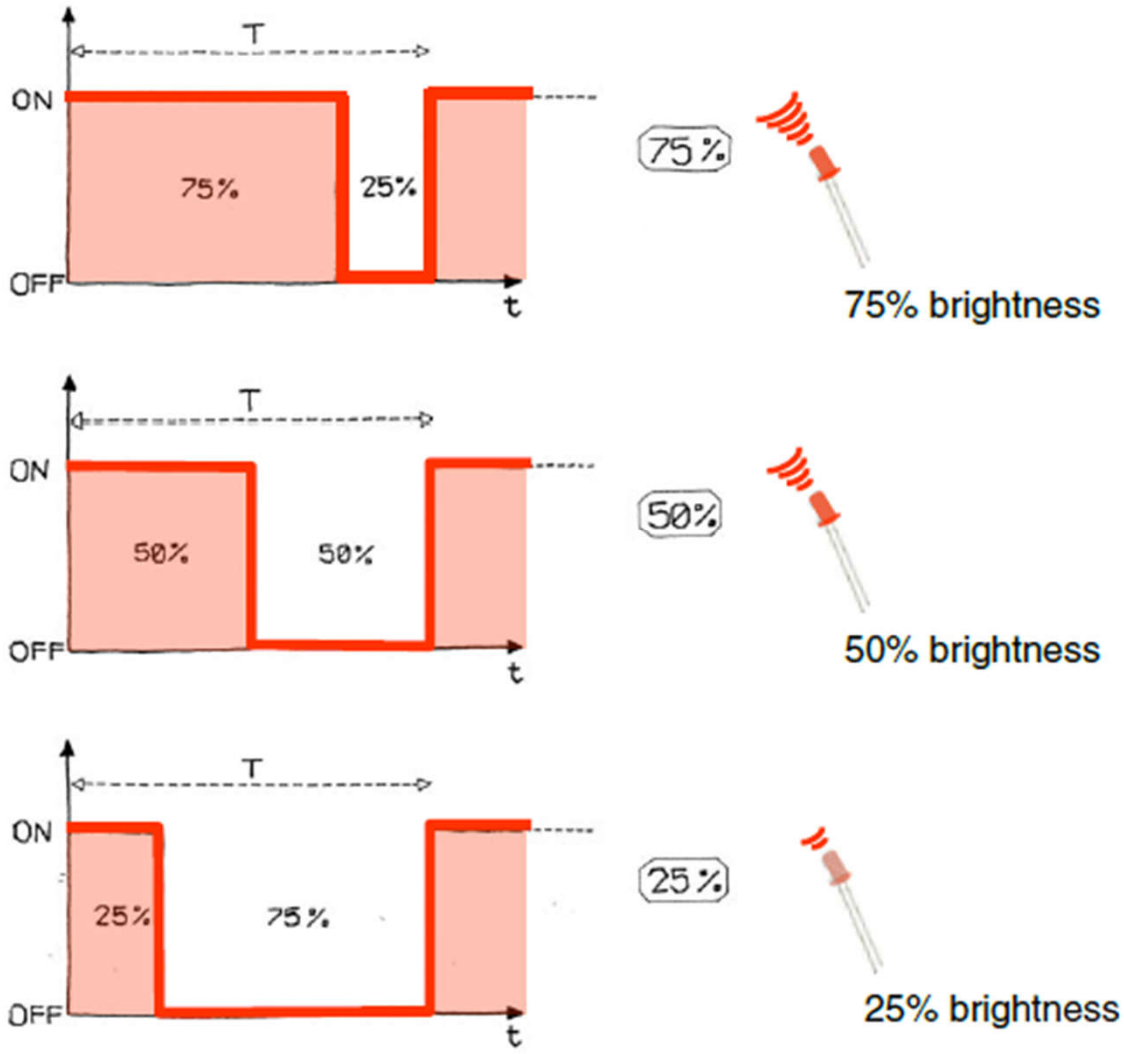


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

PWM Duty Cycle

output voltage = (on_time / cycle_time) * 5V

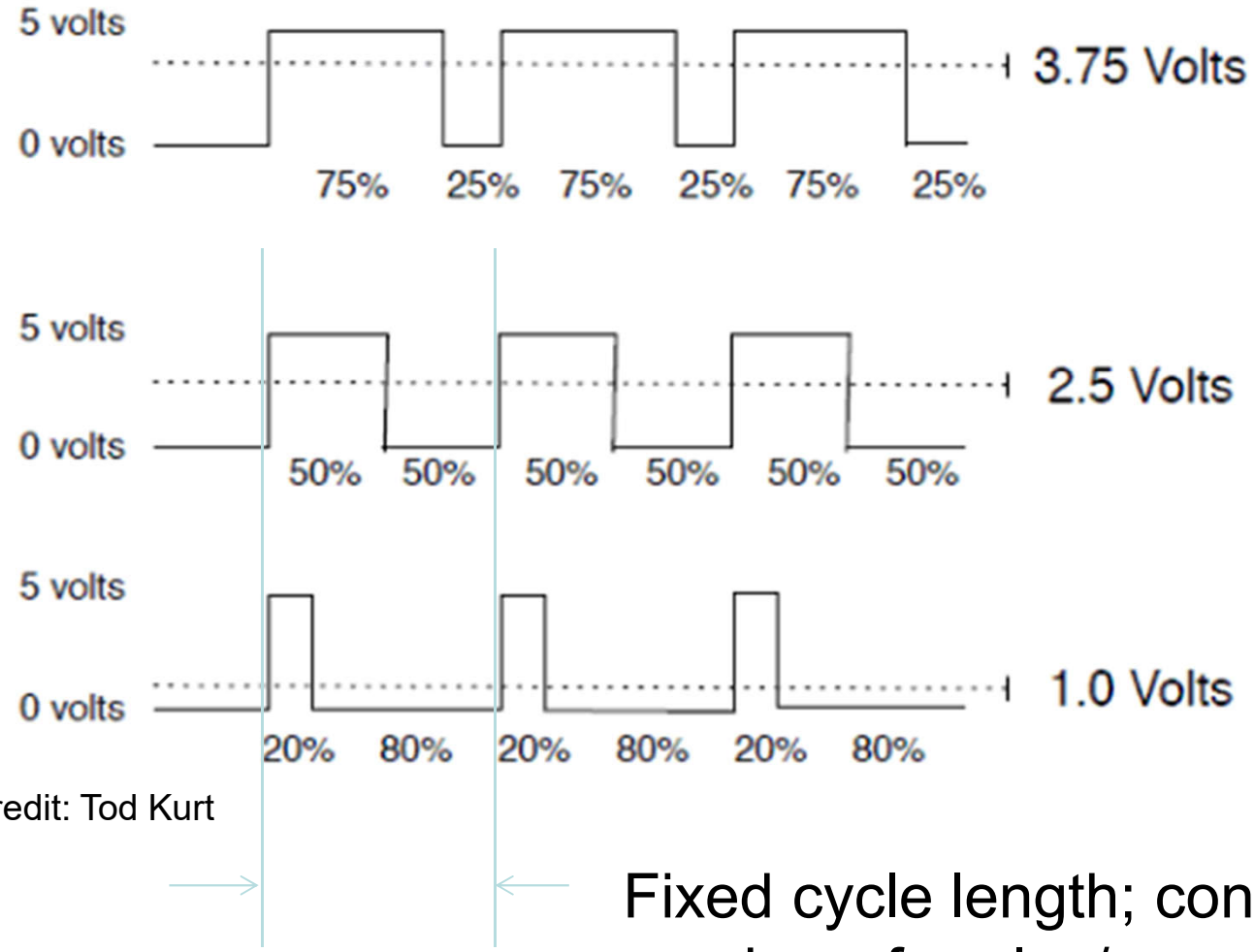
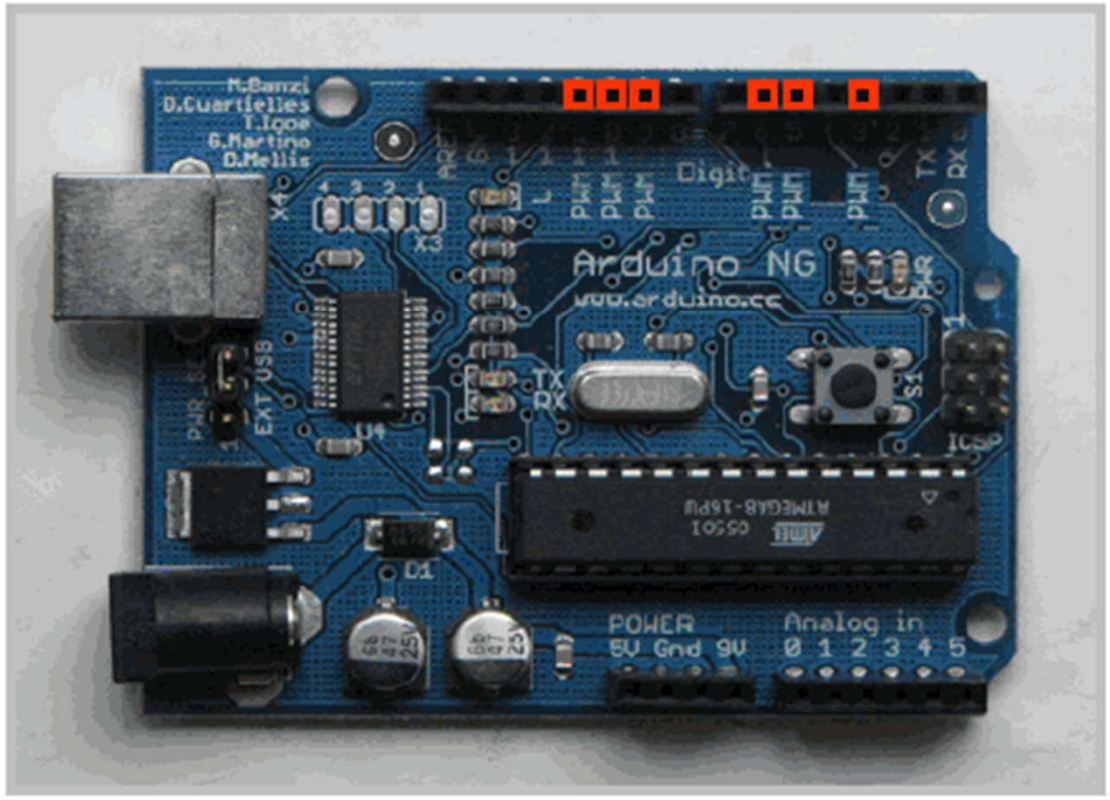


Image credit: Tod Kurt

Fixed cycle length; constant number of cycles/sec

PMW Pins

Your Arduino board has built in PWM circuits, on pins 3, 5, 6, 9, 10, and 11



- Command: **analogWrite(pin,value)**
- value is duty cycle: between 0 and 255
- Examples:
  analogWrite(9, 128)
  for a 50% duty cycle

  analogWrite(11, 64)
  for a 25% duty cycle

Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

In-class Exercise 2: Analog IO

Part 2: Add an LED

- Add a 330 ohm resistor and an LED to pin 9
- Using the analogWrite() command, set the intensity of the LED as a function of the value of prReading

Topic 4: Serial Communication

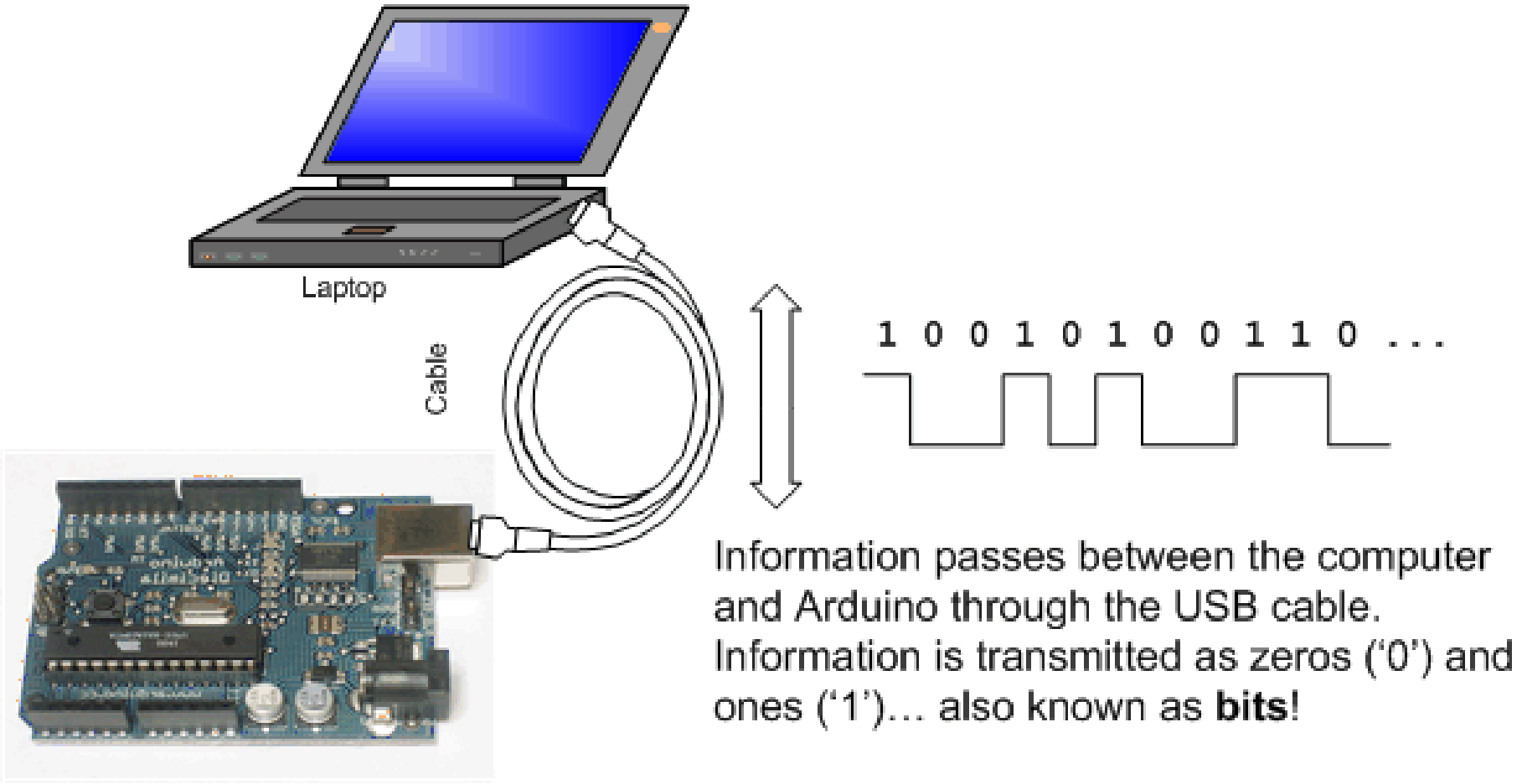


Image from http://www.ladyada.net/learn/arduino/lesson4.html

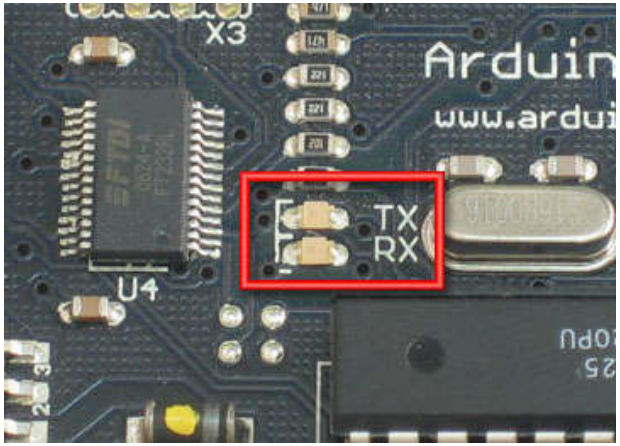
Serial Communications

- "Serial" because data is broken down into bits, each sent one after the other down a single wire.
- The single ASCII character 'B' is sent as:

```
'B' =  0 1 0 0 0 0 1 0
    =  L H L L L L H L
    =
```

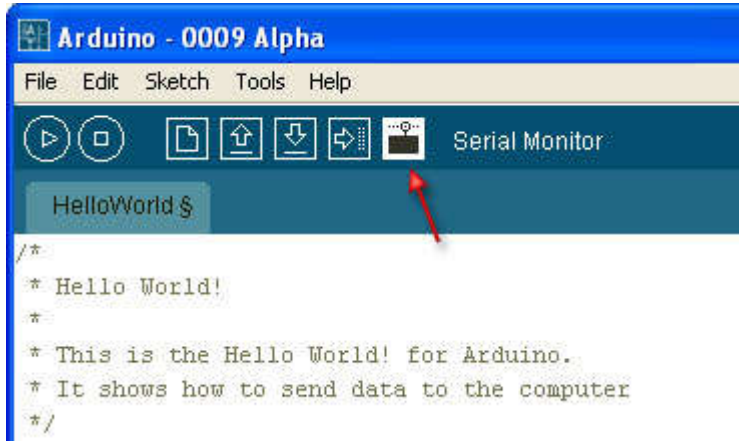
- Toggle a pin to send data, just like blinking an LED
- You could implement sending serial data with `digitalWrite()` and `delay()`
- A single data wire needed to send data. One other to receive.

Serial Communication



- ***Compiling*** turns your program into binary data (ones and zeros)
- ***Uploading*** sends the bits through USB cable to the Arduino
- The two LEDs near the USB connector blink when data is transmitted
  - **RX** blinks when the Arduino is receiving data
  - **TX** blinks when the Arduino is transmitting data

Open the Serial Monitor and Upload the Program

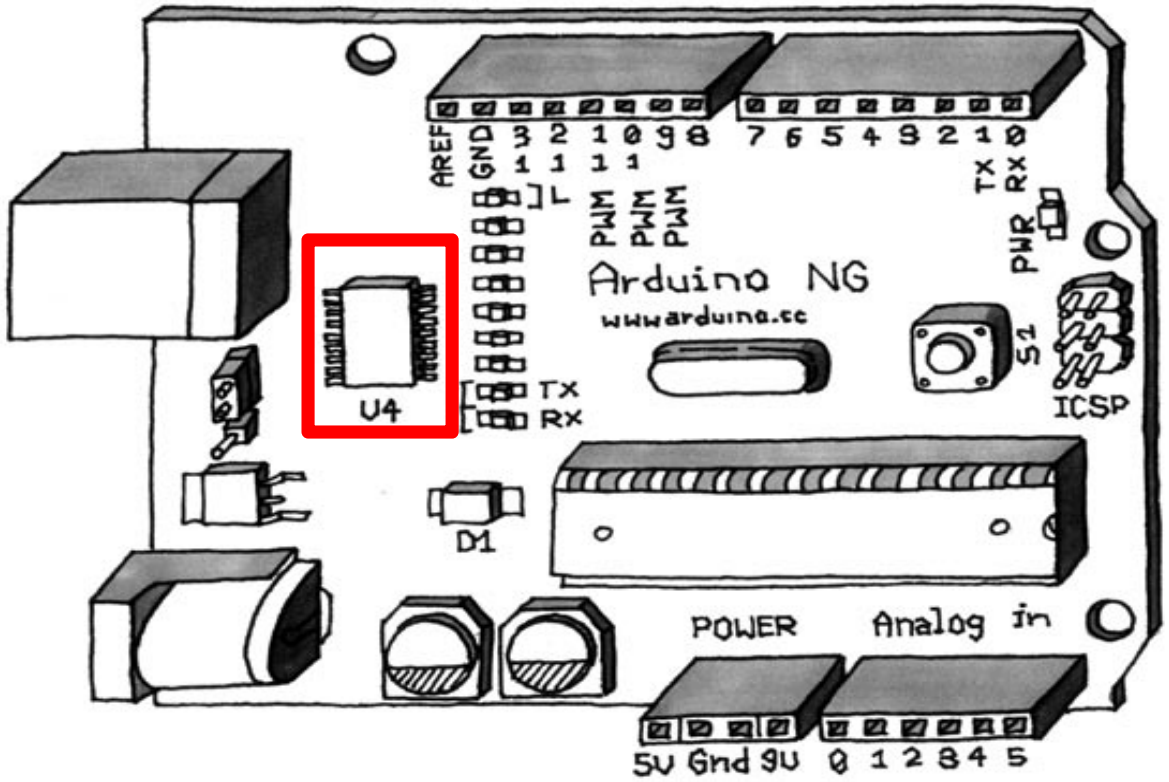


Some Commands

- Serial.begin()
  - e.g., Serial.begin(9600)
- Serial.print() or Serial.println()
  - e.g., Serial.print(value)
- Serial.read()
- Serial.available()
- Serial.write()
- Serial.parseInt()

- Example Program

Serial-to-USB chip---what does it do?



The LilyPad and Fio Arduino require an external USB to TTY connector, such as an FTDI “cable”.
In the Arduino Leonardo a single microcontroller runs the Arduino programs and handles the USB connection.

Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

Two different communication protocols

Serial (TTL):

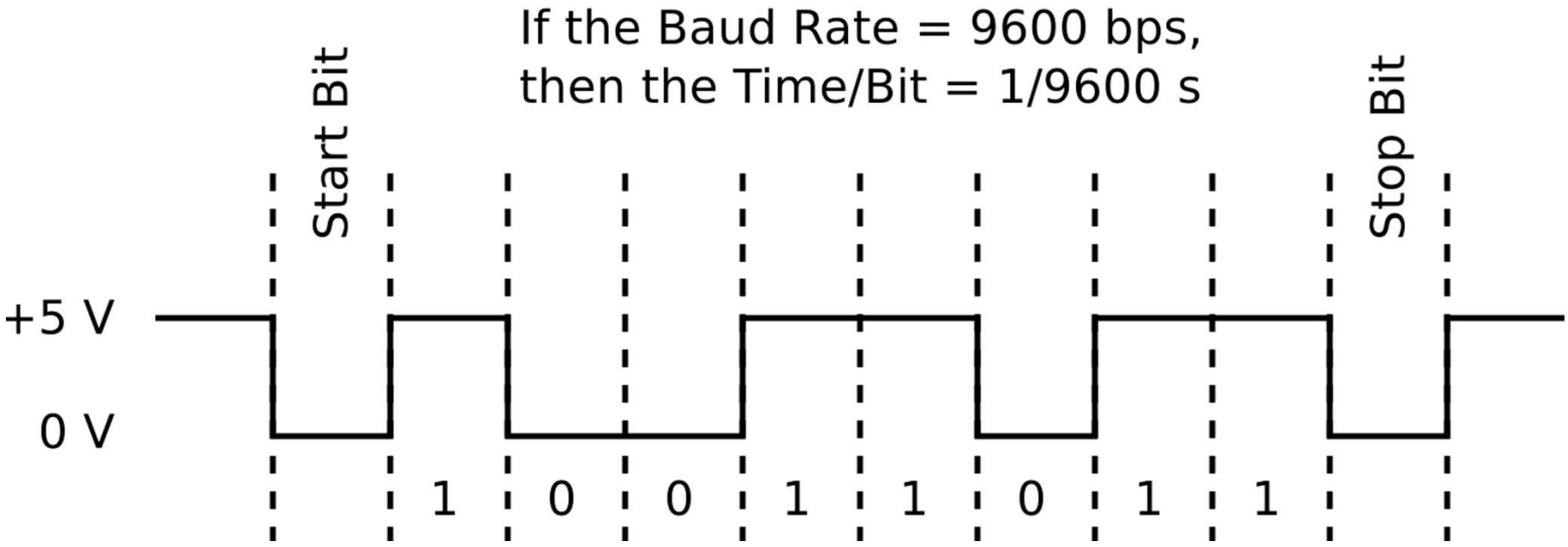


Image from http://www.fiz-ix.com/2013/02/introduction-to-arduino-serial-communication/

USB Protocol

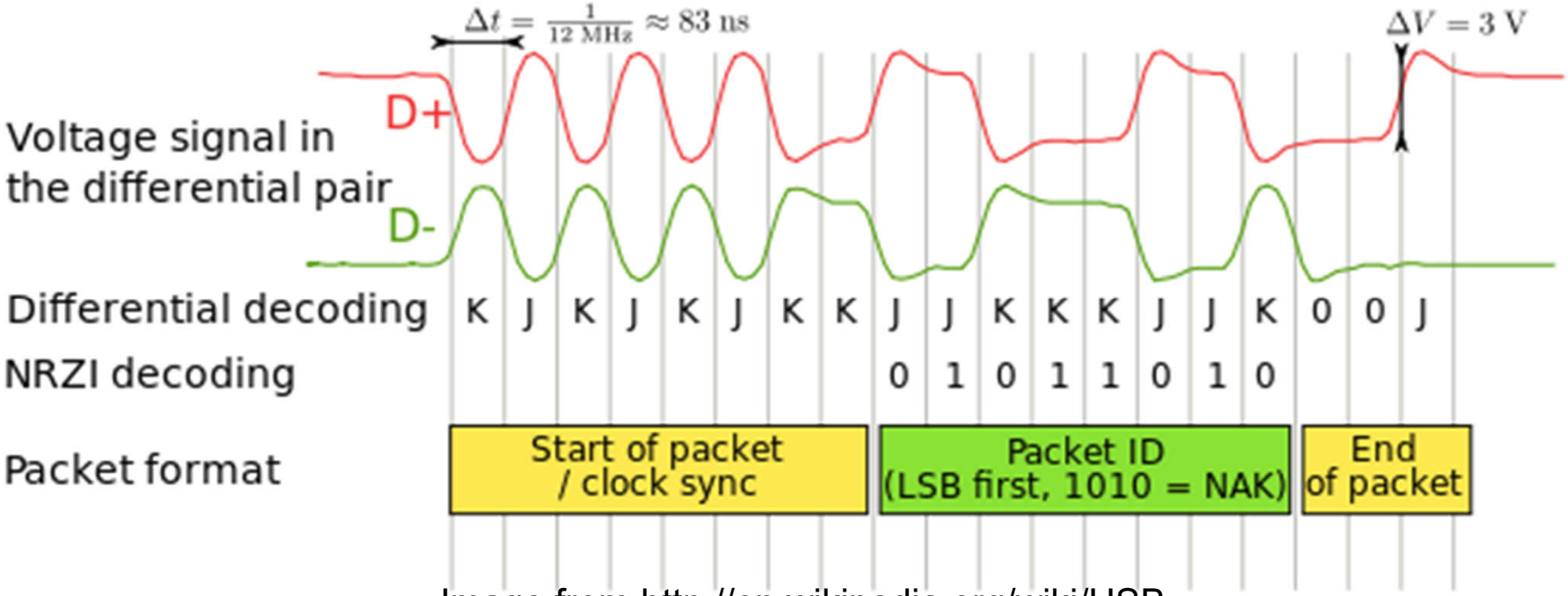


Image from http://en.wikipedia.org/wiki/USB

- Much more complicated