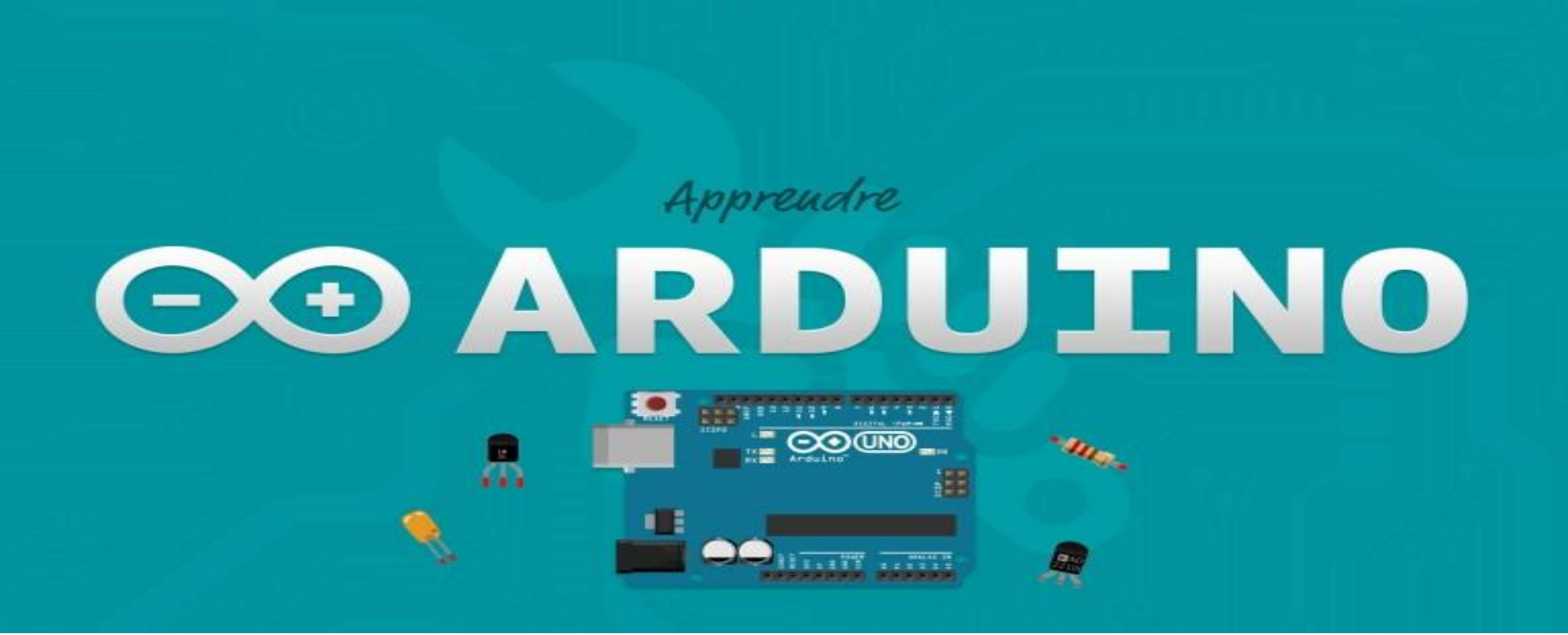
Jiangxi University of Science and Technology

C programming

Introduction To Arduino Types

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY



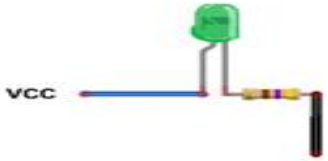
WLECOME To Arduino

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

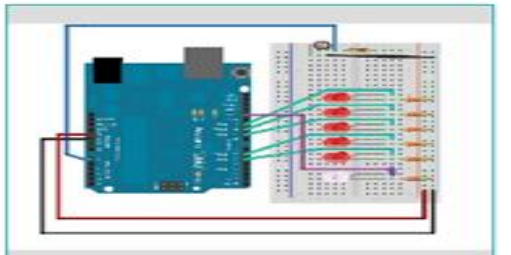
معرفی آردوینو و سخت افزار آن

نرم افزار arduino

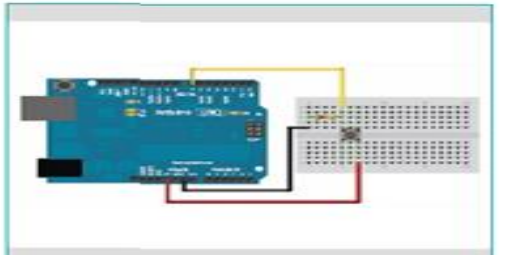
شروع کار با arduino



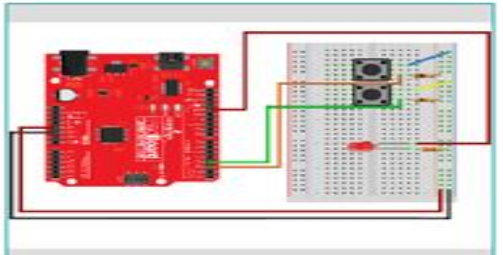
نحوه اجرای یک پروژه



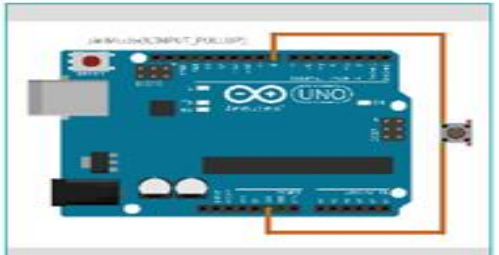
بررسی دستورات arduino



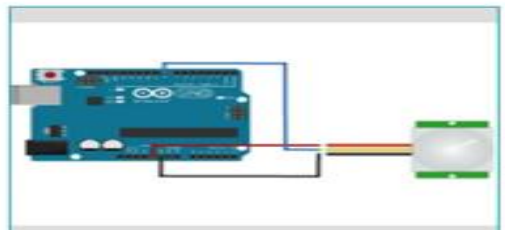
کار با کلید فشاری



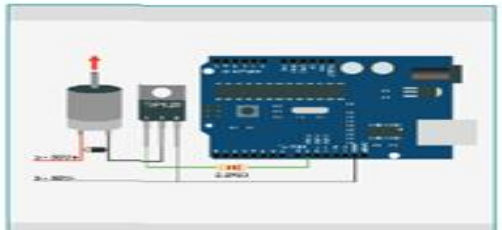
ادامه کار با کلید فشاری:



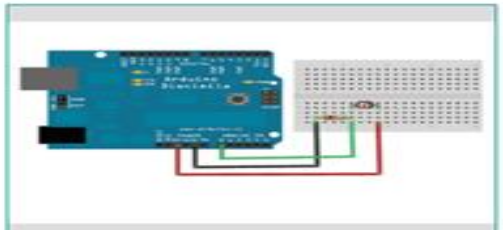
کار با LED و کلید



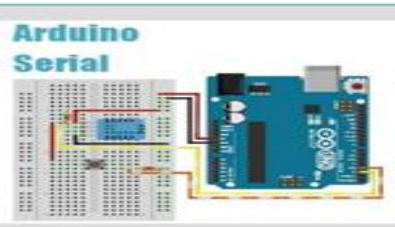
معرفی سنسورها



کار با موج های PWM



ورودی های آنالوگ در arduino

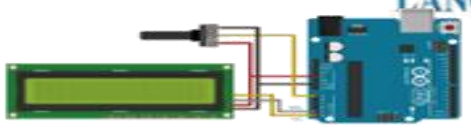


ارتباط سریال

فتوسل

ماژول آلتراسونیک

موتور سروو

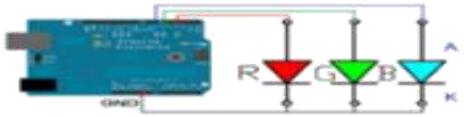


LCD کاراکتری

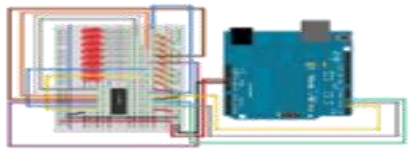
سون سگمنت 4 رقمی

ds18b20 با آردوینو

سنسور دمای DHT



RGB و نوسان رنگ



راه اندازی چند LED با شیفت رجیستر

Step 1: What is Arduino?

THE C PROGRAMMING LANGUAGE

- Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.
- You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.
- That actually says it all.

Step 2: Before you begin...

- Before you begin plugging things into your new Arduino, it may be good to know what can damage the board.
- Drawing more than 40mA from an output pin.
- An Arduino can only supply 40mA per output pin, so you cannot drive a motor or a speaker directly, for example, and you cannot connect an LED directly (without a resistor).
- Shorting an output in to the +5v, +3.3v or the ground pins, will also kill your board: If an output pin is at 5v for example, and you connect it to the ground, it draws an enormous amount of current, and kills your Arduino almost instantly.
- The pins go through the circuit board, so make sure you don't place the Arduino on a conductive (metal) surface, because it will short out the pins.

Step 2: Before you begin...

- Drawing more than 200mA from all output pins together.
  The ATmega chip on your Arduino can only supply 200mA in total, so driving more than 10 LEDs @ 20mA each, for example, will eventually damage your board.
- Supplying more than 5v (3.3v) to an input pin.
  Supplying more than the operating voltage of the Arduino on any pin is very dangerous. Some Arduinos that run at 3.3v have 5v tolerant pins, but that's about it. This also holds true for other devices, like sensors or wireless chips: always check the voltages: if you connect the output of a 5V Arduino to a 3.3V chip, you might kill it.

Step 2: Before you begin...

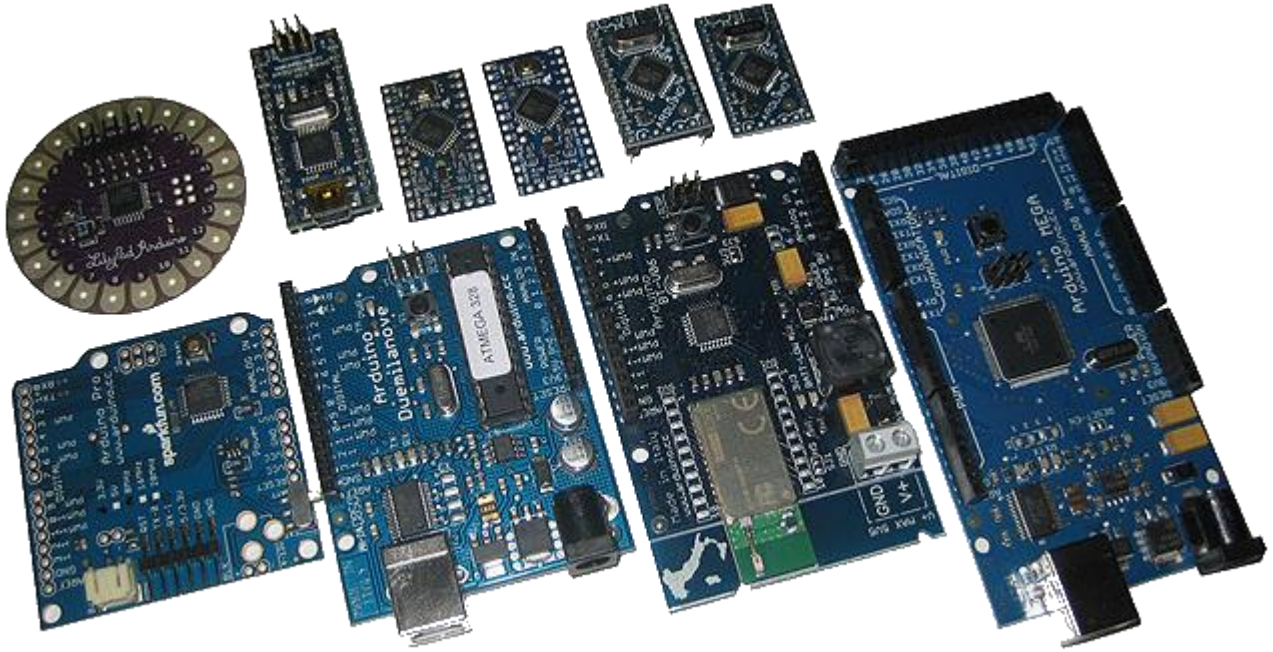
- Supplying more than 5v to the 5v pin.
  The 5v of the Arduino board goes directly to the ATmega chip, that is rated for an absolute maximum of 6v.
- Supplying more than 12v to the Vin pin.
  There's an onboard 5v voltage regulator on the board, that will overheat and die if you feed it with more than 12v.
- Drawing more than 500mA from the 5v pin (when running off an external power supply.
  The onboard 5v voltage regulator can only supply 500mA of current.
  The 5vUSB has a polyfuse to limit the current to 500mA.

Step 2: Before you begin...

- Drawing more than 50mA from the 3.3v pin.
  The onboard 3.3v voltage regulator can only supply 50mA of current. This means that you can not connect power hungry 3.3v devices like an ESP8266 or nRF24L01 directly to the Arduino: you need an external 3.3v voltage regulator.
- Reversing the polarity of the power supply.
  If you swap the 5v or Vin pin with the GND pin, you'll kill the board almost instantly. The barrel jack has a diode to protect against reverse polarity.
- Connecting a load to the Vin pin while using USB power.
  If you connect a load to the Vin pin while the 5v to the Arduino comes from the USB connection, current will flow backwards through the voltage regulator, damaging it.
- Static electricity
  Although most chips have clamping diodes as protection against ESDs (electrostatic discharges), it may be wise to us an anti-static wrist strap, or to remove the carpet under your desk.

Arduino Types

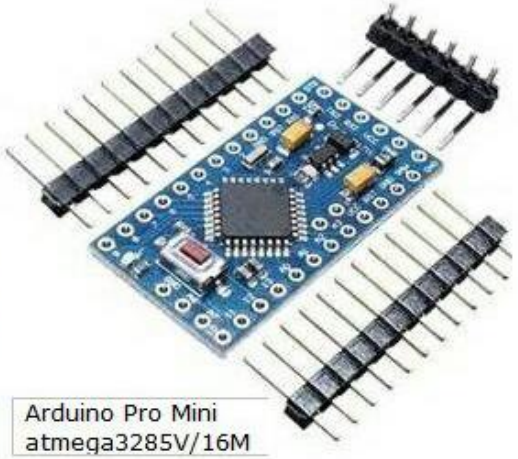
- Many different versions
  - Number of input/output channels
  - Form factor
  - Processor
- Leonardo
- Due
- Micro
- LilyPad
- Esplora
- Uno



THE C PROGRAMMING LANGUAGE

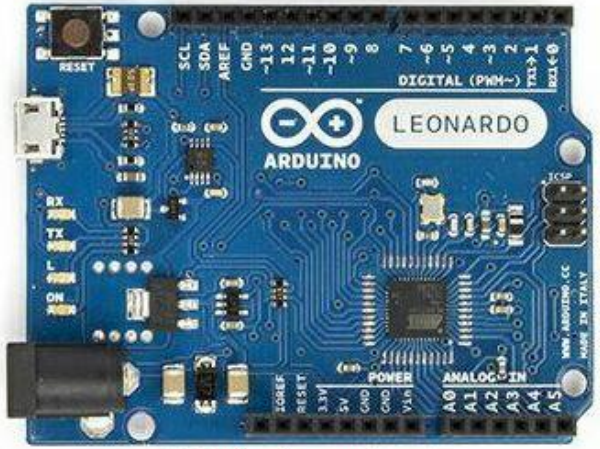
Arduino UNO R3
atmega328pwith USB cable

Arduino UNO R3
mega328pwith USB cable



Arduino Pro Mini
atmega3285V/16M

Arduino MEGA2560 R3
withUSB cable(With logo)

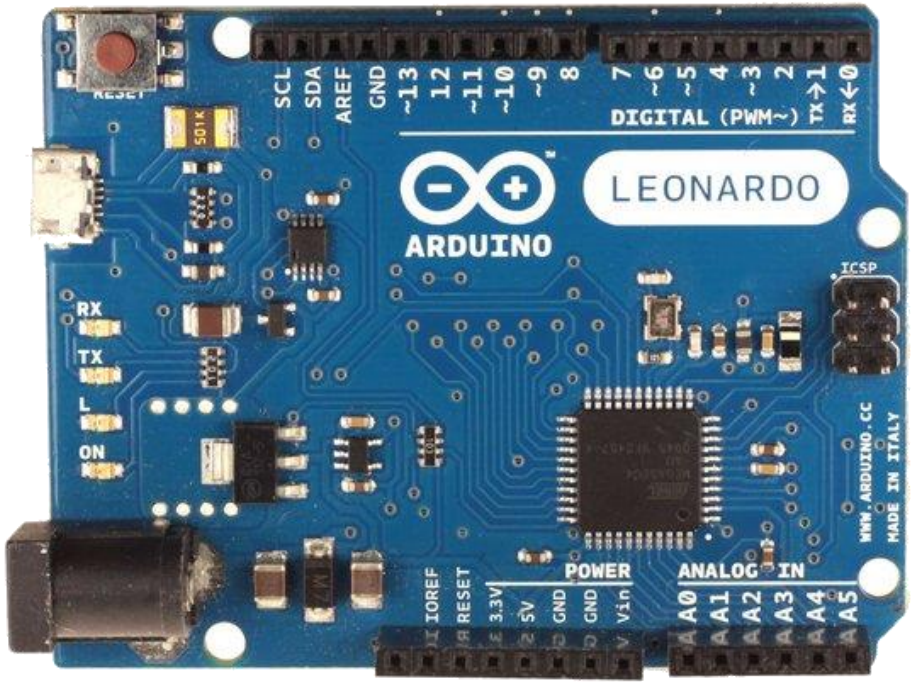


Arduino Leonardo
R3ATMEGA32U4
with USB cable

Arduino Nano
V3.0with USB
cable

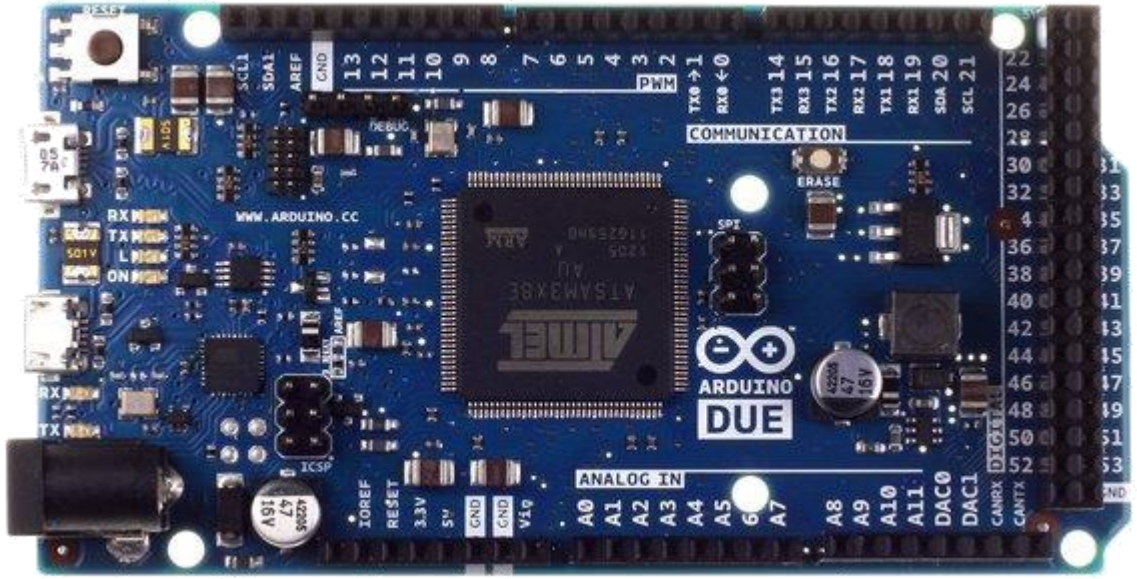
Leonardo

- Compared to the Uno, a slight upgrade.
- Built in USB compatibility
- Bugs?
  - Presents to PC as a mouse or keyboard



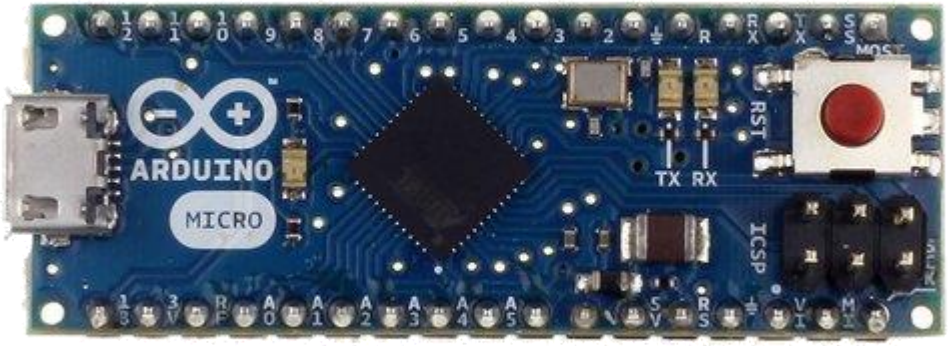
Due

- Much faster processor, many more pins
- Operates on 3.3 volts
- Similar to the Mega



Micro

- When size matters: Micro, Nano, Mini
- Includes all functionality of the Leonardo
- Easily usable on a breadboard

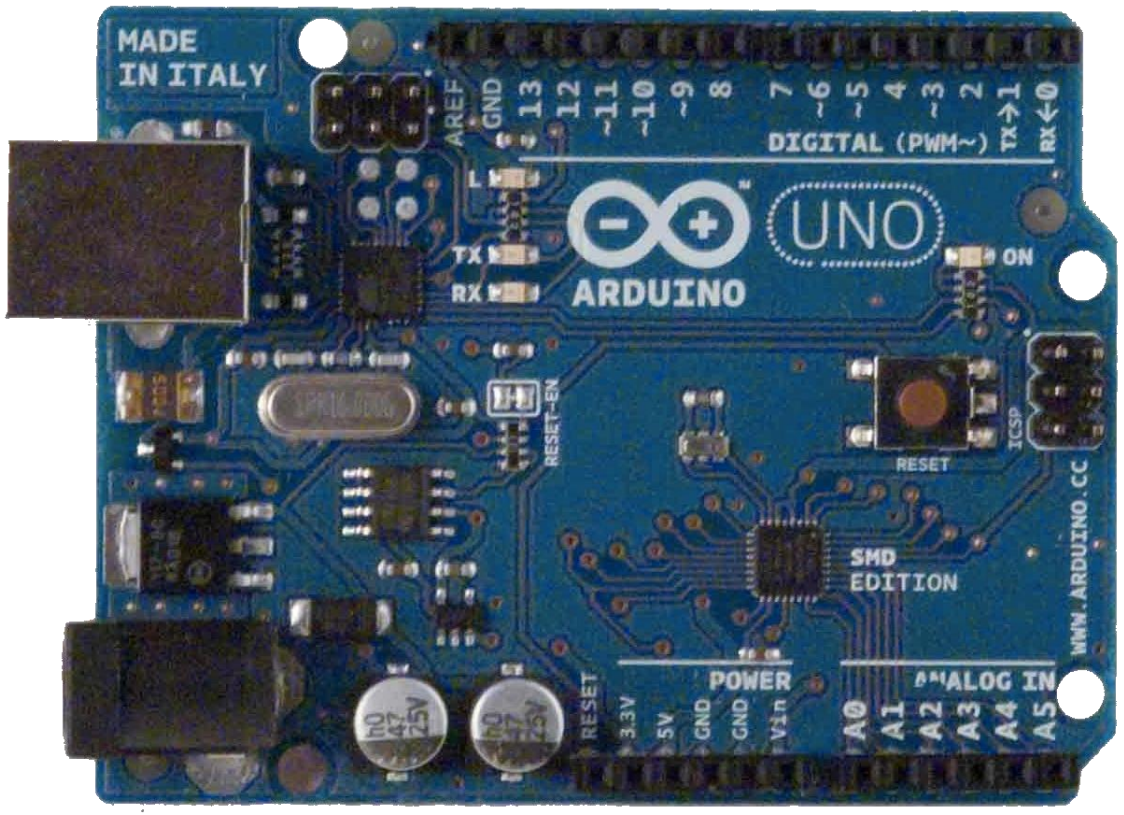


LilyPad

- LilyPad is popular for clothing-based projects.

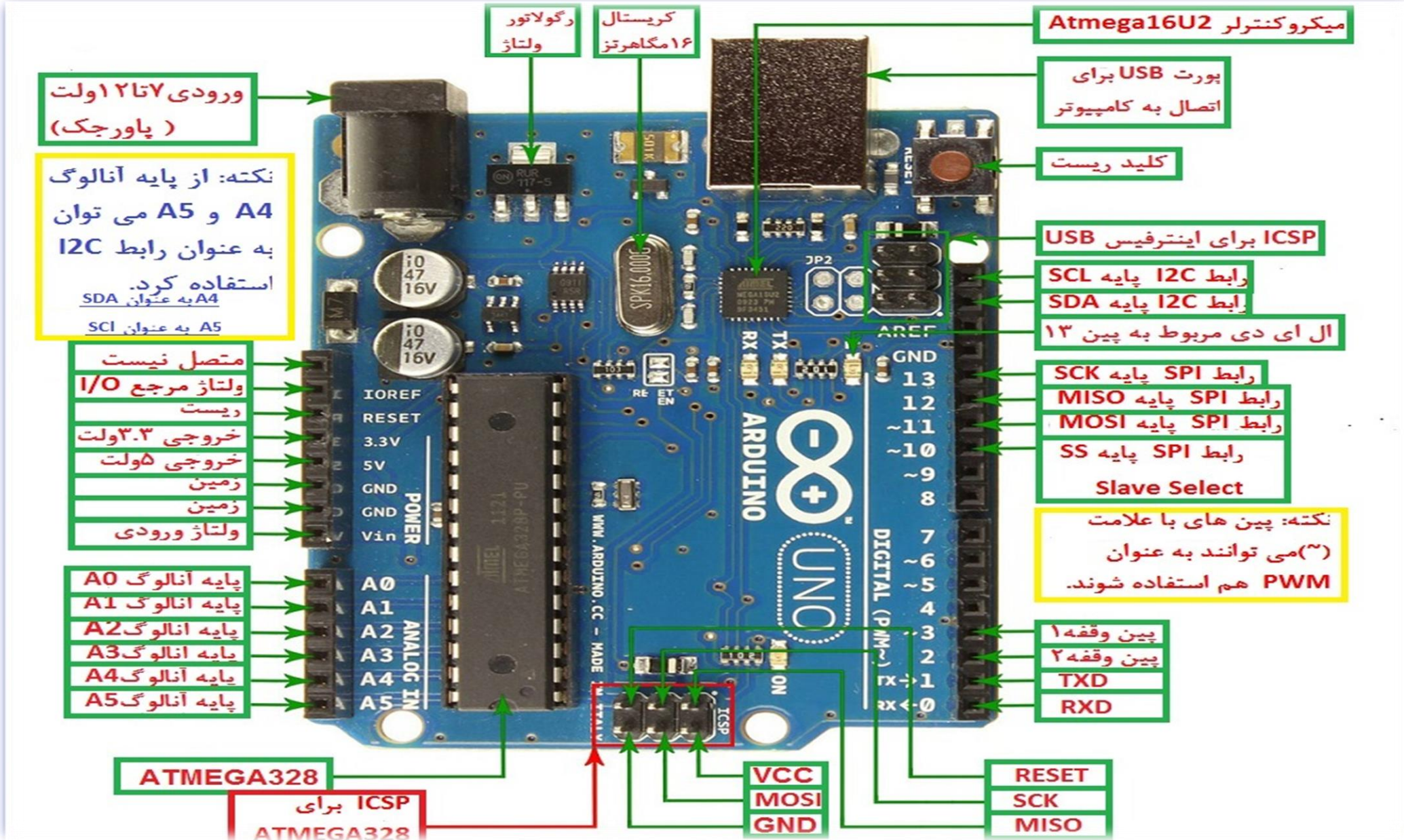
Uno

- The pins are in three groups:
  - Invented in 2010
  - 14 digital pins
  - 6 analog pins
  - power



THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

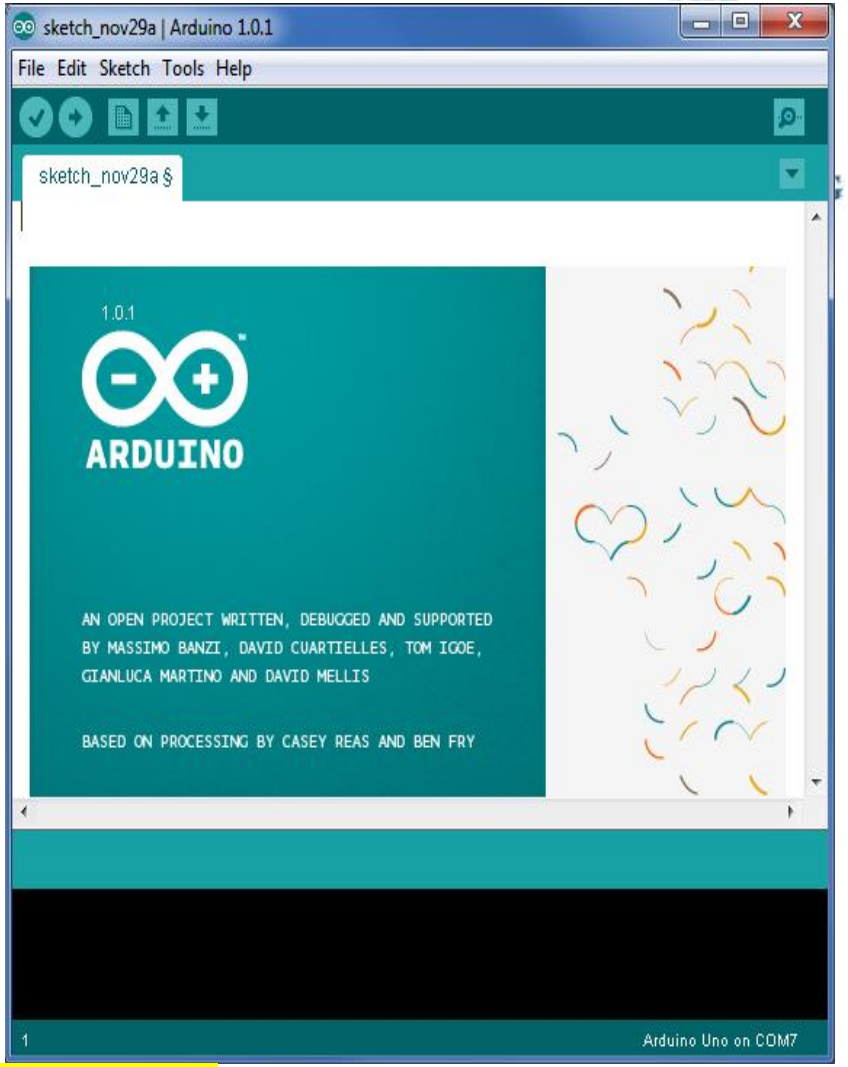


THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino Compiler

- **Download current compiler from:**
  arduino.cc/en/Main/software
- **Arrogantly refers to itself as an IDE .**
- **Run the software installer.**
- **Written in Java, it is fairly slow.**



Visit playground.arduino.cc/Main/ DevelopmentTools
for alternatives to the base arduino IDE

Arduino Program Development

- Based on C++ without 80% of the instructions.
- A handful of new commands.
- Programs are called 'sketches'.
- Sketches need two functions:

  void setup( )

  void loop( )

  - setup( ) runs first and once.
  - loop( ) runs over and over, until power is lost or a new sketch is loaded.

Arduino C Specific Functions

THE C PROGRAMMING LANGUAGE

- **pinMode(pin, mode)**
  - Designates the specified pin for input or output
- **digitalWrite(pin, value)**
  - Sends a voltage level to the designated pin
- **digitalRead(pin)**
  - Reads the current voltage level from the designated pin
- **analog versions of above**

  analogRead's range is 0 to 1023
- **serial commands**

  print, println, write

Jiangxi University of Science and Technology

How to spot a counterfeit Arduino

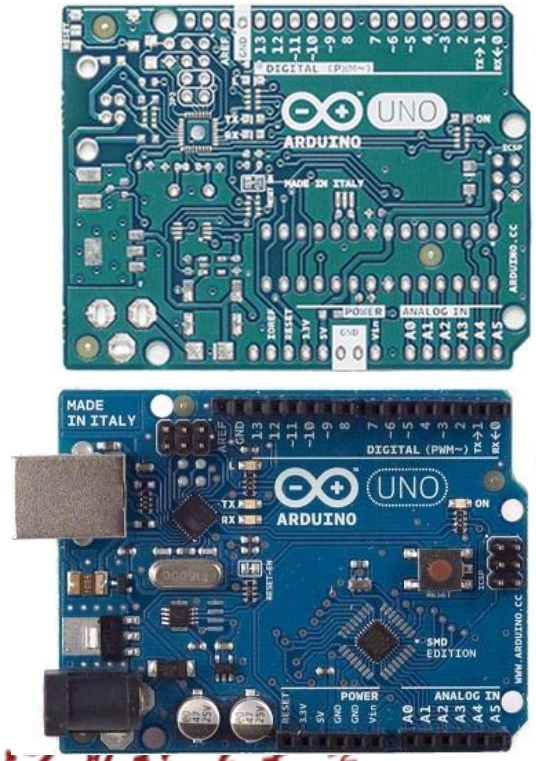
- https://www.arduino.cc/en/Products/Counterfeit

How to spot a counterfeit Arduino

Color

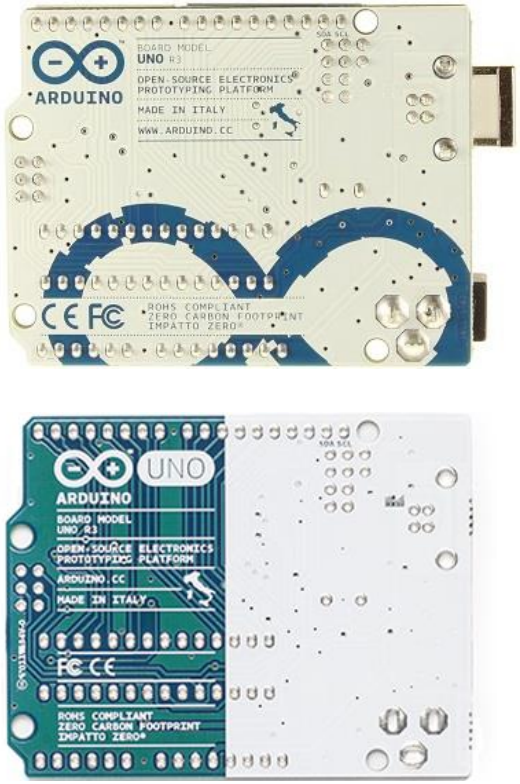
The authentic Arduino's color is either a mix of green and blue, or the classic Arduino teal.
The counterfeits usually are blue to deep blue color.

NEW Original board (Front)



Counterfeit (Front)

NEW Original board (Front)



Counterfeit (Front)

Font/silk

Original board

When you carefully observe the font in which the word 'Uno' or 'Arduino' is written we can find a difference.
Arduino font is custom designed and cloners tend to overlook or not be able to reproduce the same.
Observe the 'O'!

Logo

There is only one Arduino logo, which looks like this.

Original board

Look for the sharp edges and the hole inside the 'A' as compared to the following fakes:

The fake, has 'rounded' edges and the print is not sharp.

Made in Italy

Original board

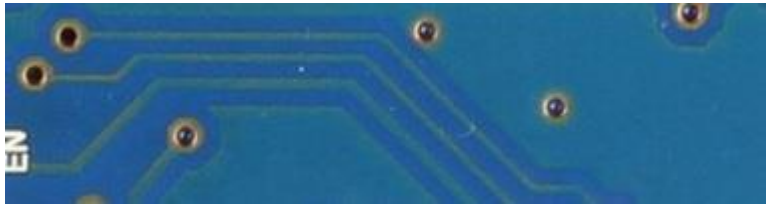
Turn your Arduino board around to see the map of Italy, saluting the place of its birth.

Counterfeit

Connections and connectors

In every original Arduino the connections between components look 'woven' and beautiful, not 'pasted' and ugly.

Original board



Counterfeit

Components

- **Here's a major fake-alert! The component 501 K located next to the voltage regulator is generally green on fake boards.**
- **The green one has similar properties as the original Arduino golden-black component and is available off the shelf.**
- **We get it specially made for Arduino to make it look different:**

Original board

Counterfeit

Which Arduino Do You Need for Your Project?

By **Brock Craft** from **Arduino Projects For Dummies**

The Arduino project has come a long way in only a few years. It started out with just a single simple board that offered basic features and did not have a USB connector. You can now find over a dozen Arduino boards, each with its own unique characteristics and features. The current full listing of Arduino products is on the **Arduino website**. Select the Arduino that best matches your needs.

Which Arduino Do You Need for Your Project?

By **Brock Craft** from **Arduino Projects For Dummies**

- **Arduino Uno:** This is the main workhorse in the Arduino family. The Uno is based on the ATMega328 microcontroller, and operates at 16MHz. It has 14 digital input/output (I/O) pins and 6 analog input pins. Power can be provided over the USB connection or DC Barrel connector, or by using power input header pins.
- The onboard power regulator is smart enough to know which one is being used. It has a handy onboard “utility” LED connected to digital Pin 13, and a reset button for when things get weird.
- A key difference of the Uno from all previous boards is that it has a USB controller chip integrated onboard. This feature makes it much easier to hit the ground running because you simply plug it into your computer and the device will be recognized. Previous versions required you to install software drivers for a USB interface provided by FTDI.
- **Arduino Mega:** The Mega 2560 is the Uno’s big brother. It has all of the basic functionality and is fully compatible, but benefits from a ton of extra connections — 54 digital IO pins and 16 analog input pins! It also has more pins that offer pulse-width modulation (PWM), which is useful for dimming LEDs and controlling motors.It costs a little more, but if you want to control an army of devices or read in a fistful of sensors, this Mega is the way to go.

Which Arduino Do You Need for Your Project?

By **Brock Craft** from **Arduino Projects For Dummies**

- **Arduino Leonardo:** The Leonardo is similar to the Uno, but has a more digital I/Os (20) and analog (12) inputs. Where it really stands apart, is that it has a second serial port. It can also do some nifty tricks like emulating a keyboard and a mouse. It can be programmed to control these input devices so your computer will act as if it’s receiving keyboard and mouse input.
- **Lilypad Arduino:** The Lilypad is an Arduino with personality! It’s a favorite of people who want a little style and designed for sewing into wearables and textiles and clothing. It has almost the same number of digital input/output pins as a regular Arduino, but they are arranged in a circle and the connections can be sewn into clothing with conductive thread. Running at 8MHz, it’s “fast fashion”!

Which Arduino Do You Need for Your Project?

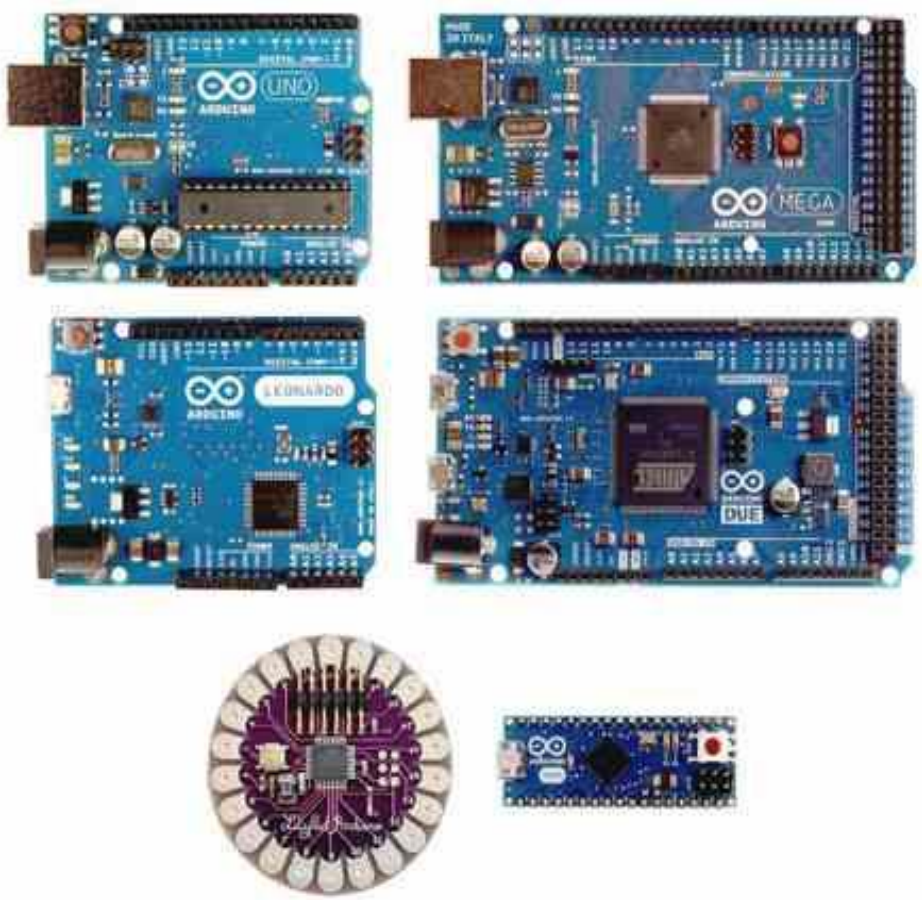
By **Brock Craft** from **Arduino Projects For Dummies**

THE C PROGRAMMING LANGUAGE

- **Arduino Micro:** The Micro is super cute and perfect for tight spaces. It has only the essential requirements — a built-in micro USB connector and 20 digital input/output pins. It has no pin headers, so you solder connections directly onto the board itself. You can also solder on headers so that it can be inserted into a breadboard.
- **Arduino Due:** The Arduino Due boasts a much beefier processor and is really a full-fledged computer on a board, similar to a Raspberry Pi or a BeagleBoard. It has an Atmel SAM3X8E ARM Cortex-M3 CPU that runs at the brisk pace of 84MHz, and 54 digital ports. It's more robust and uses 3.3V DC onboard rather than 5V DC, so you should avoid it for the projects here.

Which Arduino Do You Need for Your Project?

By **Brock Craft** from **Arduino Projects For Dummies**

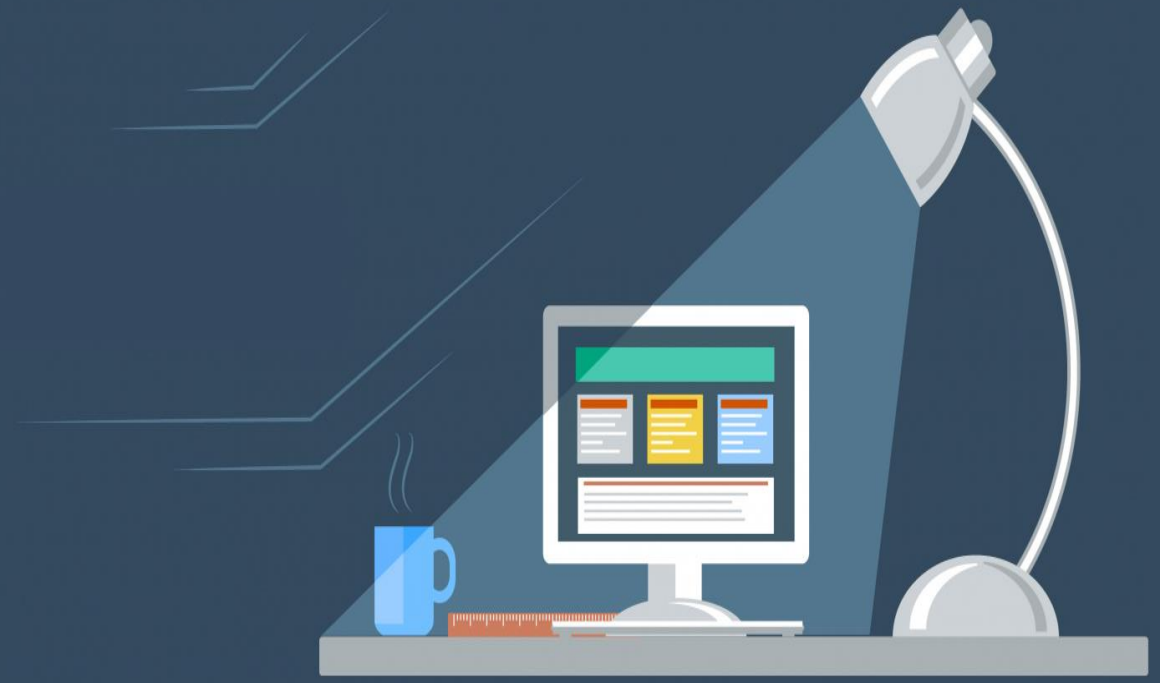


- You may also come across older boards and want to use them for your project. Older Arduinos you may encounter are Diecimila, Duemilanove, NG, and Bluetooth.
- Be aware that the Arduino IDE has changed a bit as the hardware has evolved and not all the older boards will work with the most current IDE. Also, some software libraries that offer extended features are not compatible with some of the older boards. If you have trouble with a project using one of the older boards, you may want to try it out with an Arduino Uno instead.

THE C PROGRAMMING LANGUAGE
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

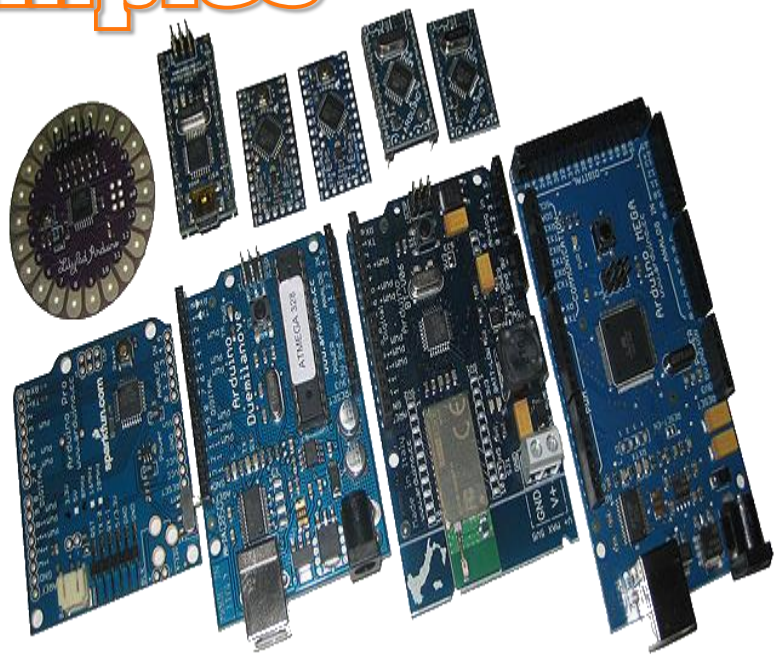
Reference

- MY BOOK
- Arduiuno .cc



Jiangxi University of Science and Technology

Lecture 2: Introduction To IDE and Examples



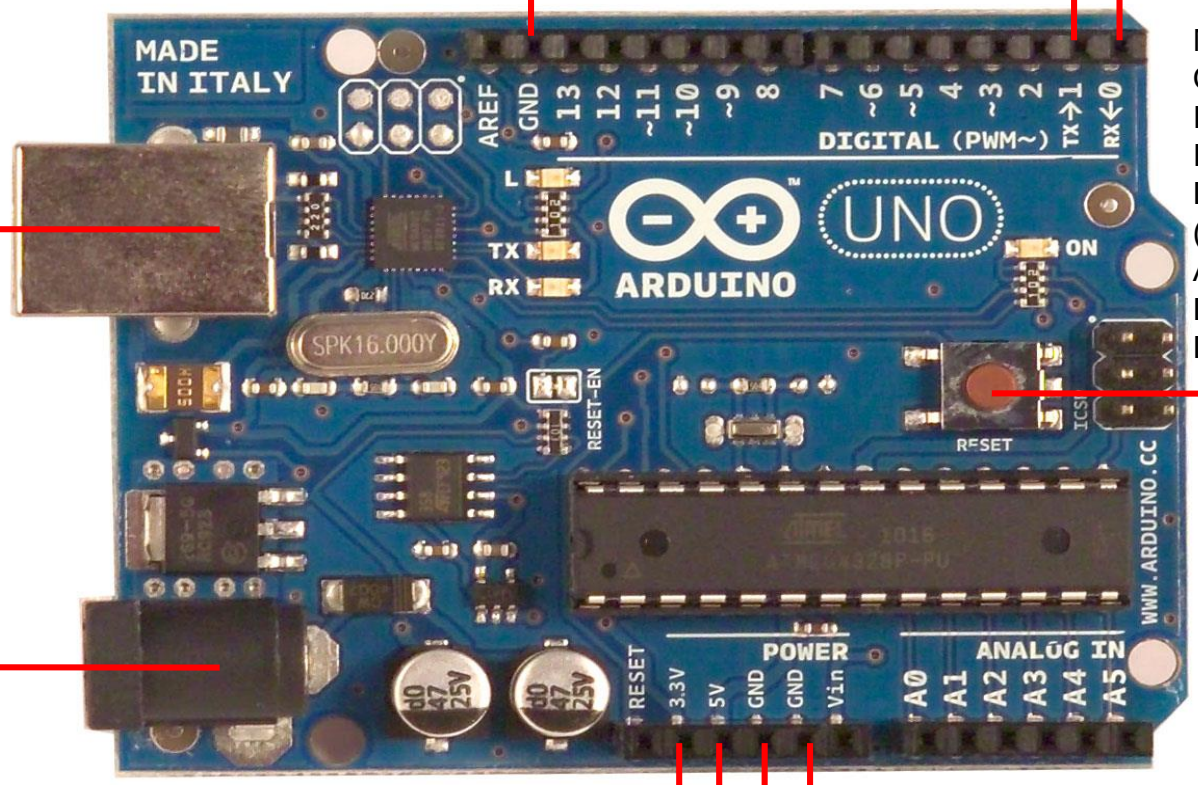
THE C PROGRAMMING LANGUAGE

UNO

Digital Input/Output Pins
Pins with ~ are PWM
[Analog Output]

GRD

Transmitter/Receiver
Serial Connection



USB

7-12 v

Microcontroller ATmega328
Operating Voltage 5V
Input Voltage (recommended)7-12V
Input Voltage (limits)6-20V
Digital I/O Pins 14
(of which 6 provide PWM output)
Analog Input Pins 6
DC Current per I/O Pin 40 mA
DC Current for 3.3V Pin 50 mA

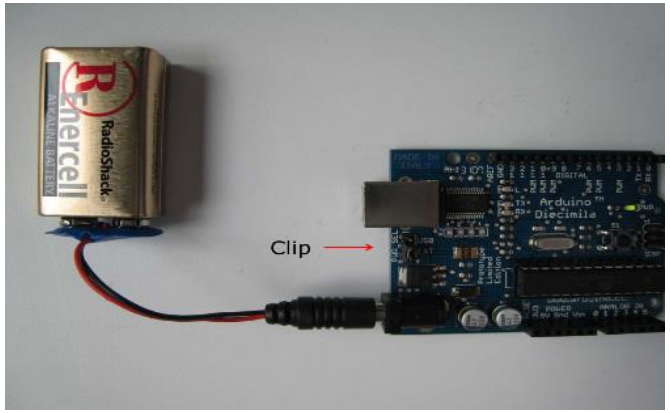
3 v 5 v
GRD

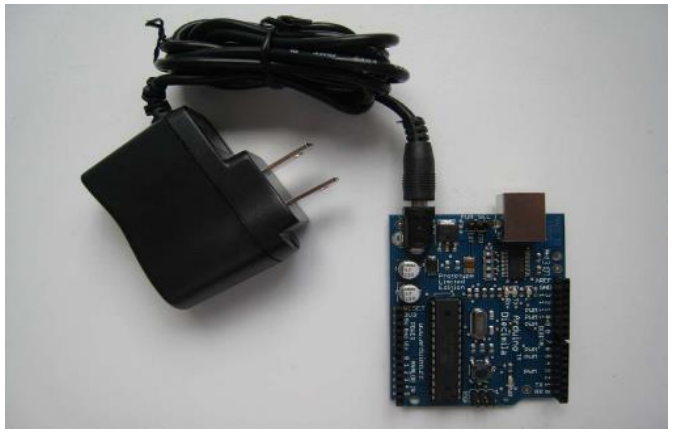
Analog Input Pins

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino Board: Arduino UNO





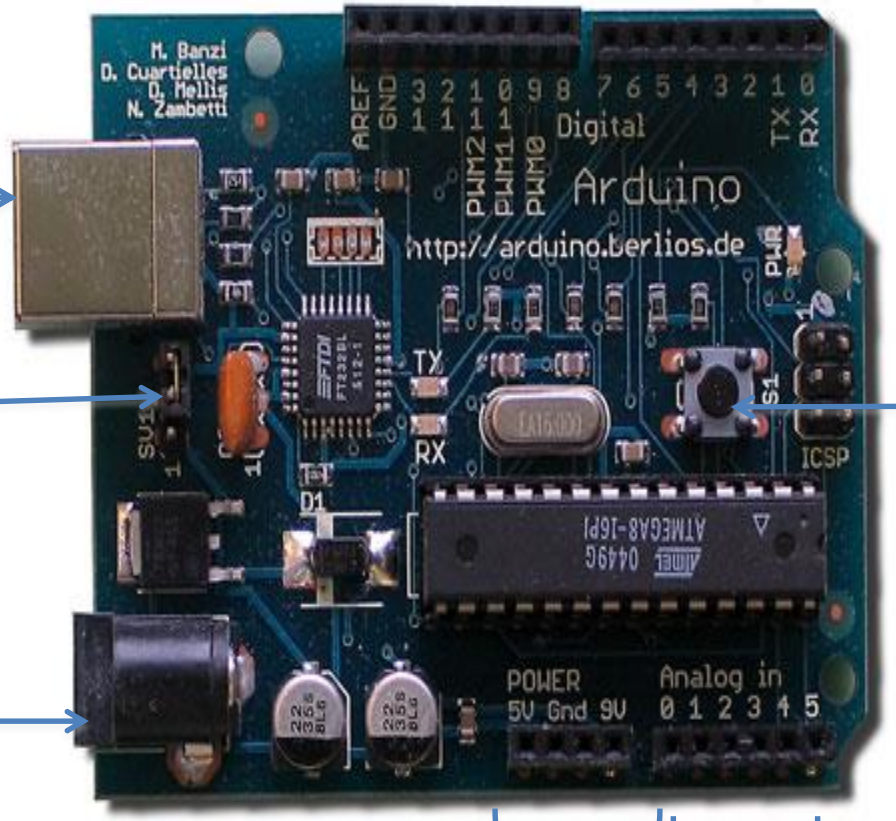
USB (Data & Power)

Power Source Jumper

Alternate Power (9V)

Digital I/O (2 – 13)

Serial Transfer (0 -1)



Reset

5V / 9V / GND (x2)

Analog Input (0 – 5)

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

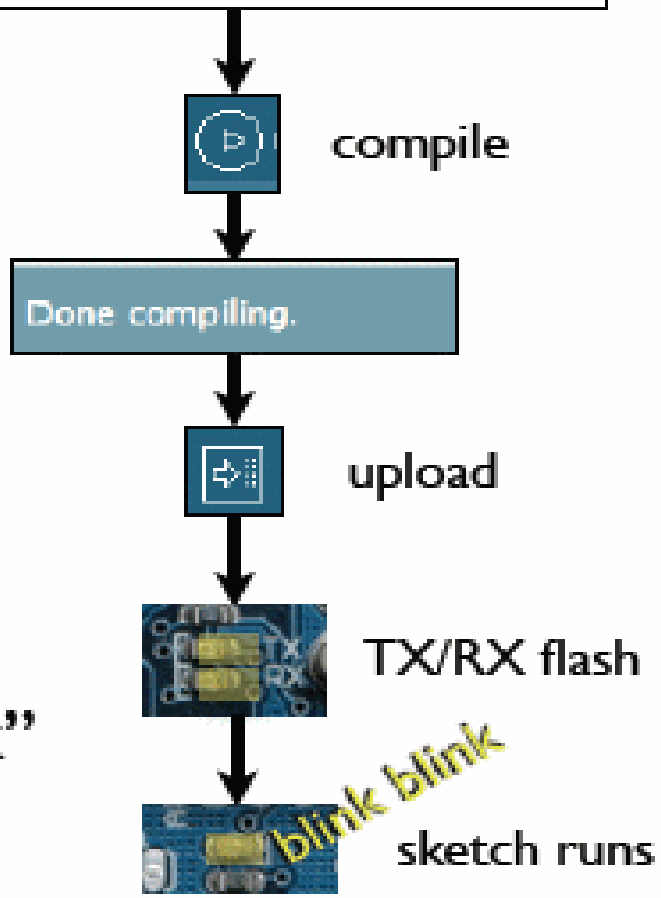
Using Arduino Board

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

Try it out with the "Blink" sketch!

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

```
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
}
```



THE
MMING
UAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino IDE Window

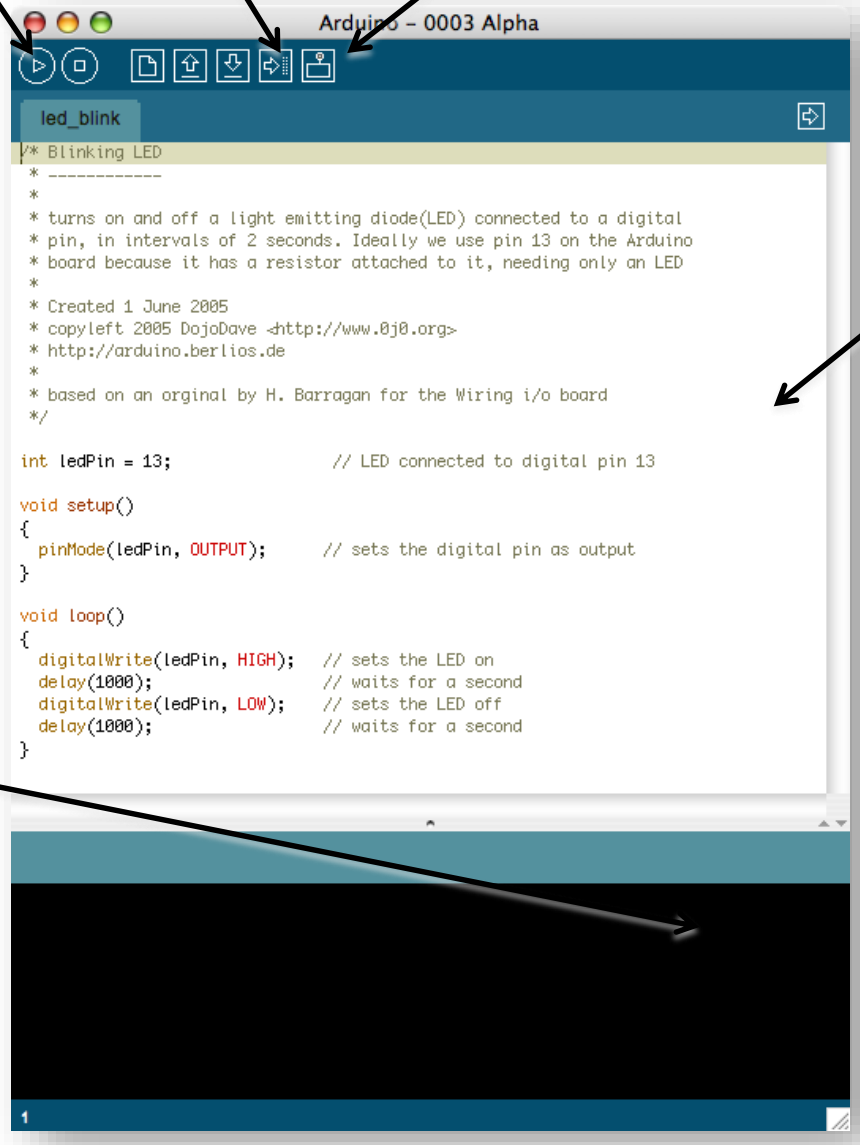
Compile

Upload to controller

Serial Monitor

Code Editor

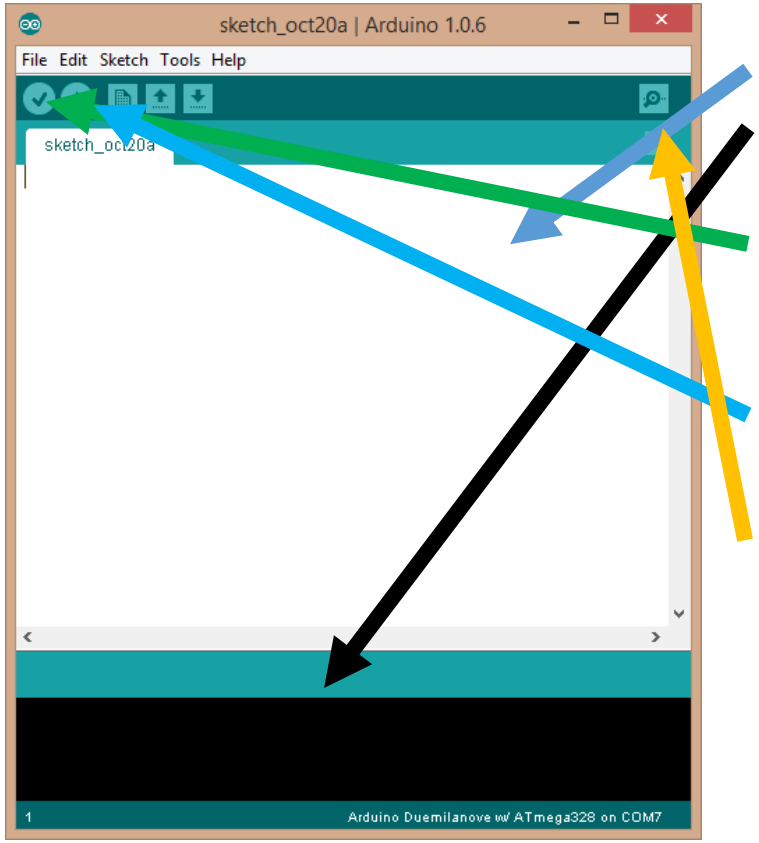
Text Output (Serial Data)



THE C PROGRAMMING LANGUAGE

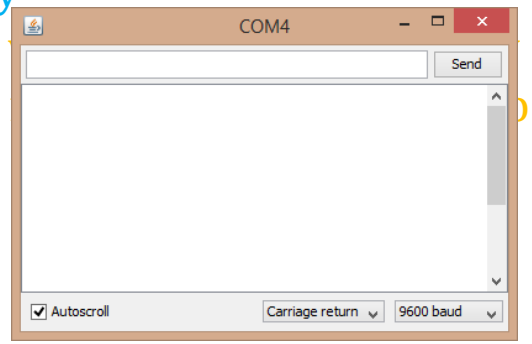
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

The Arduino IDE



The main features you need to know about are:

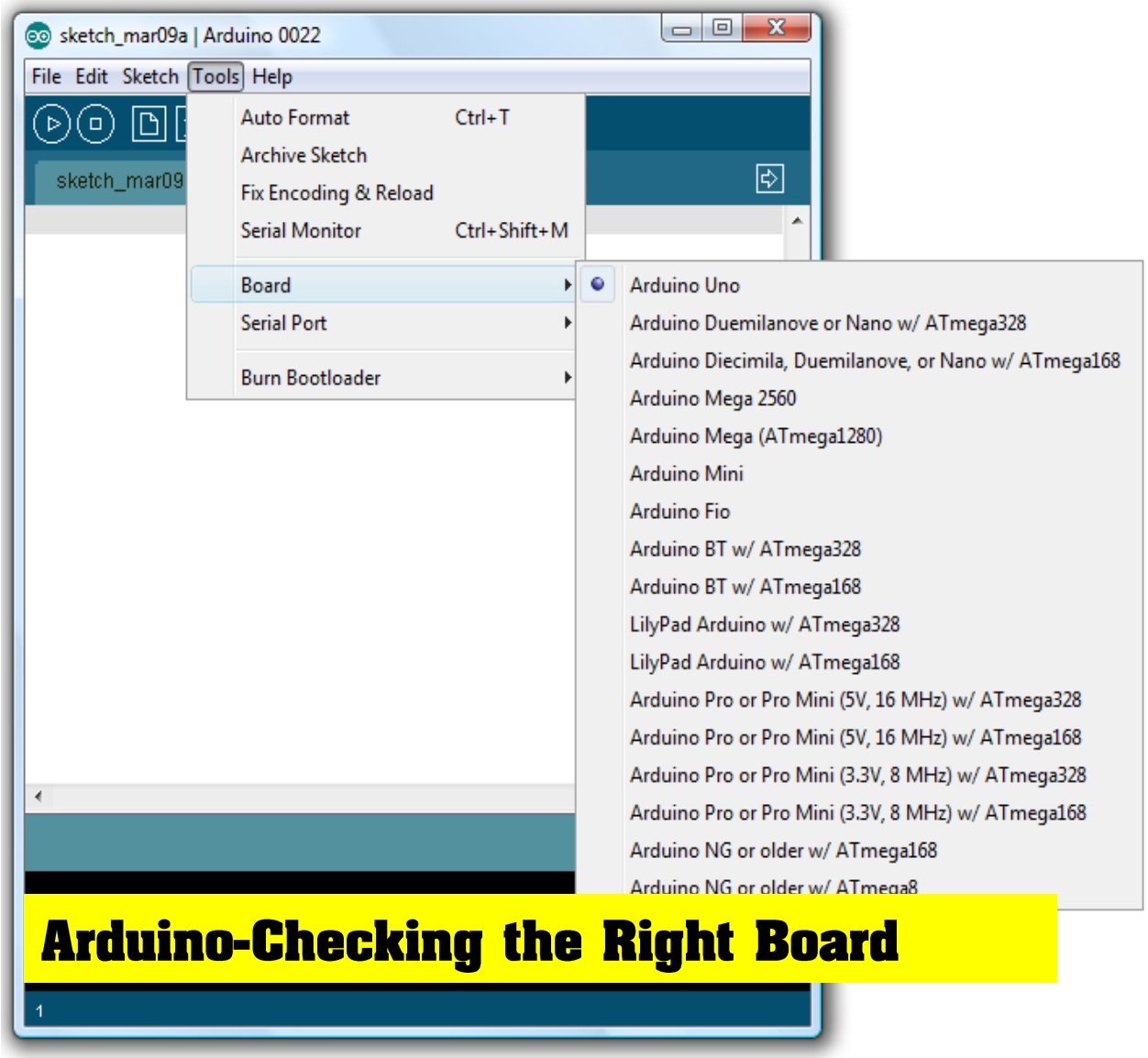
- **Code area:** This is where you will type all your code
- **Info panel:** This will show any errors during compiling or uploading code to your Arduino
- **Verify:** This allows you to compile your code to code the Arduino understands. Any mistakes you have made in the syntax of your code will be show in the info pannel
- **Upload:** This does the same as verify but will then send your code to your Arduino if the code is verified successfully
- **Serial Monitor:** This that allows you to send text to a . We will use this feature in la



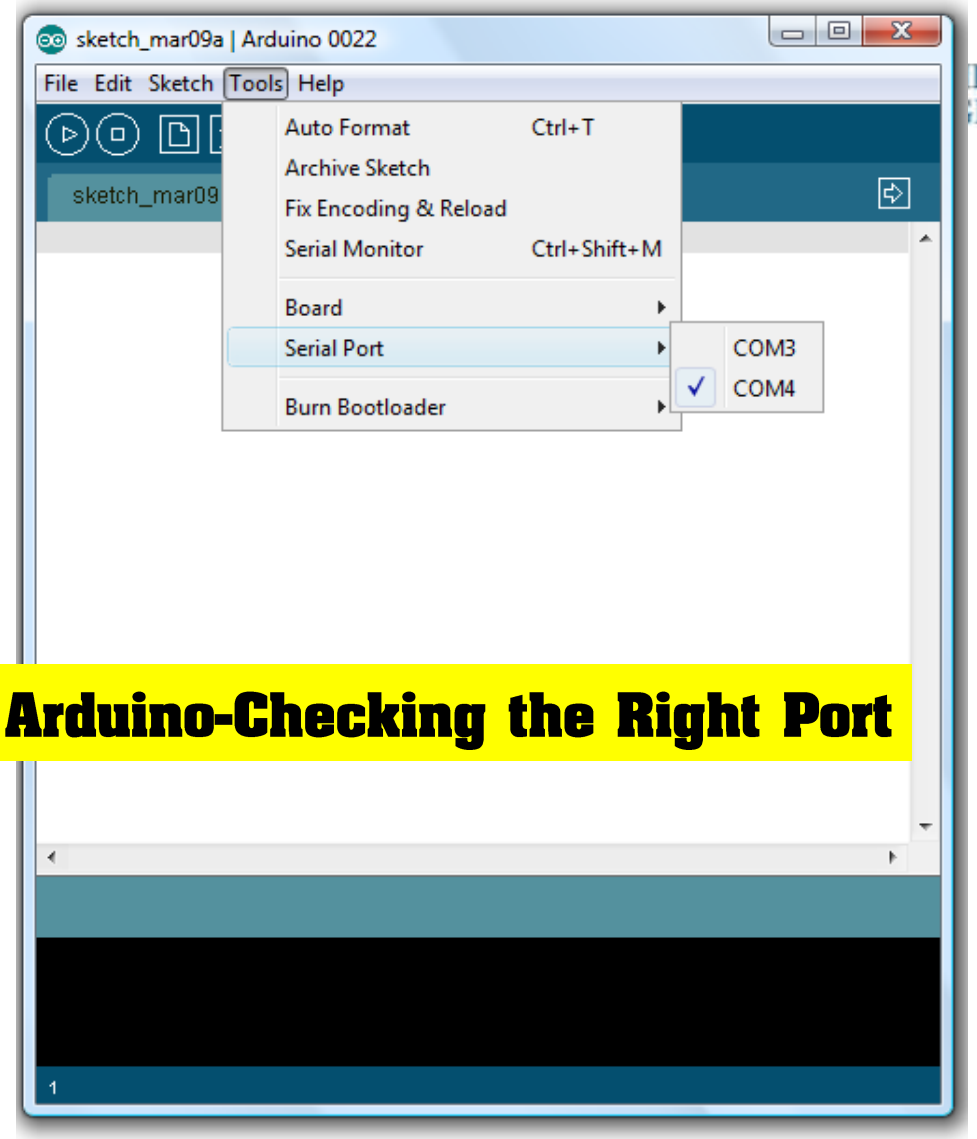
Important functions

- **Serial.println(value);**
  - **Prints the value to the Serial Monitor on your computer**
- **pinMode(pin, mode);**
  - **Configures a digital pin to read (input) or write (output) a digital value**
- **digitalRead(pin);**
  - **Reads a digital value (HIGH or LOW) on a pin set for input**
- **digitalWrite(pin, value);**
  - **Writes the digital value (HIGH or LOW) to a pin set for output**

Two common problem

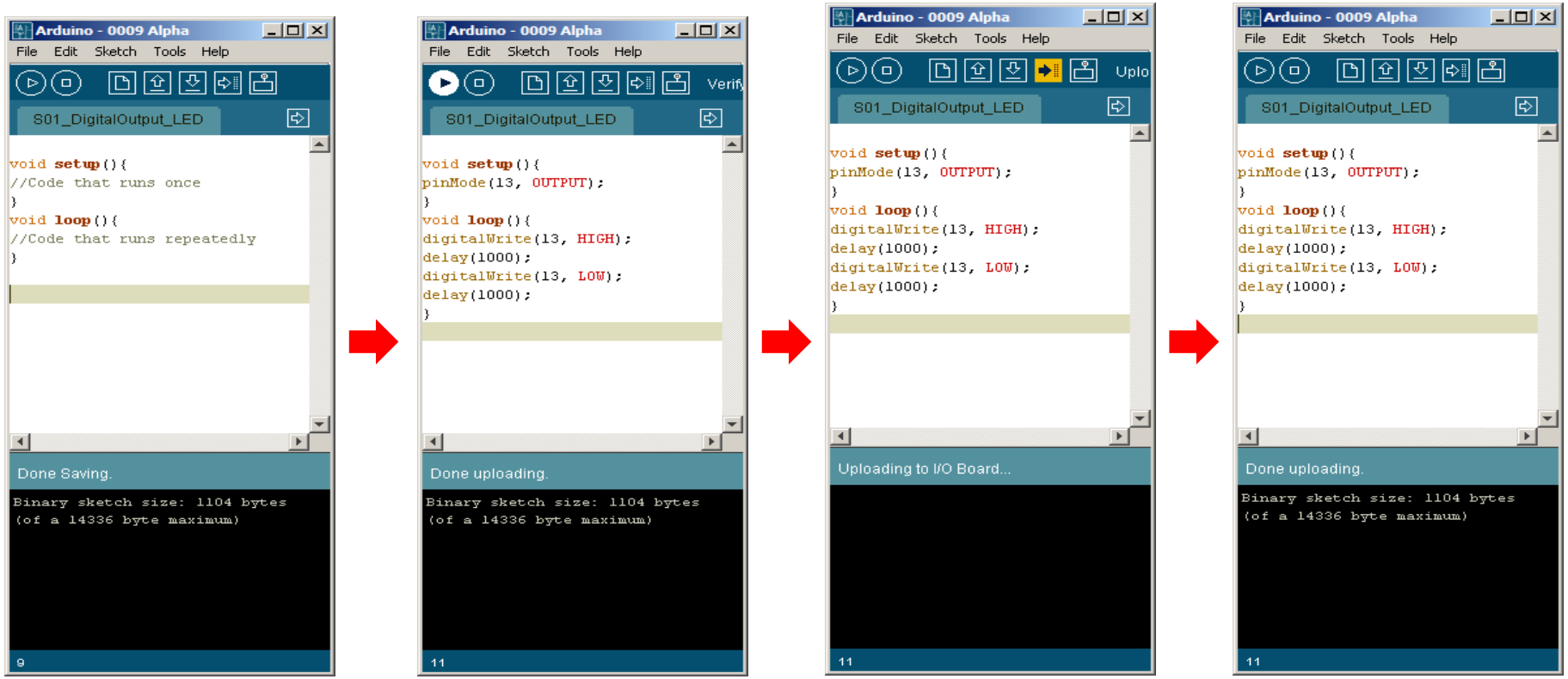


Arduino-Checking the Right Board



Arduino-Checking the Right Port

Arduino-Compiling and Uploading Code



1. Write the code
2. Compile the code
3. Check Arduino Port Connection
4. Upload the Code
5. The Arduino and Connected Circuits start to show behavior based on the uploaded code

THE C PROGRAMMING LANGUAGE
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino C is Derived from C++

- These programs blink an LED on pin 13

- avr-libc

```
#include <avr/io.h>
#include
  <util/delay.h>

int main(void){
  while (1){
    PORTB =0x20;
    _delay_ms(1000);
    PORTB =0x00;
    _delay_ms(1000);
  }
  return 1;
}
```

- Arduino C

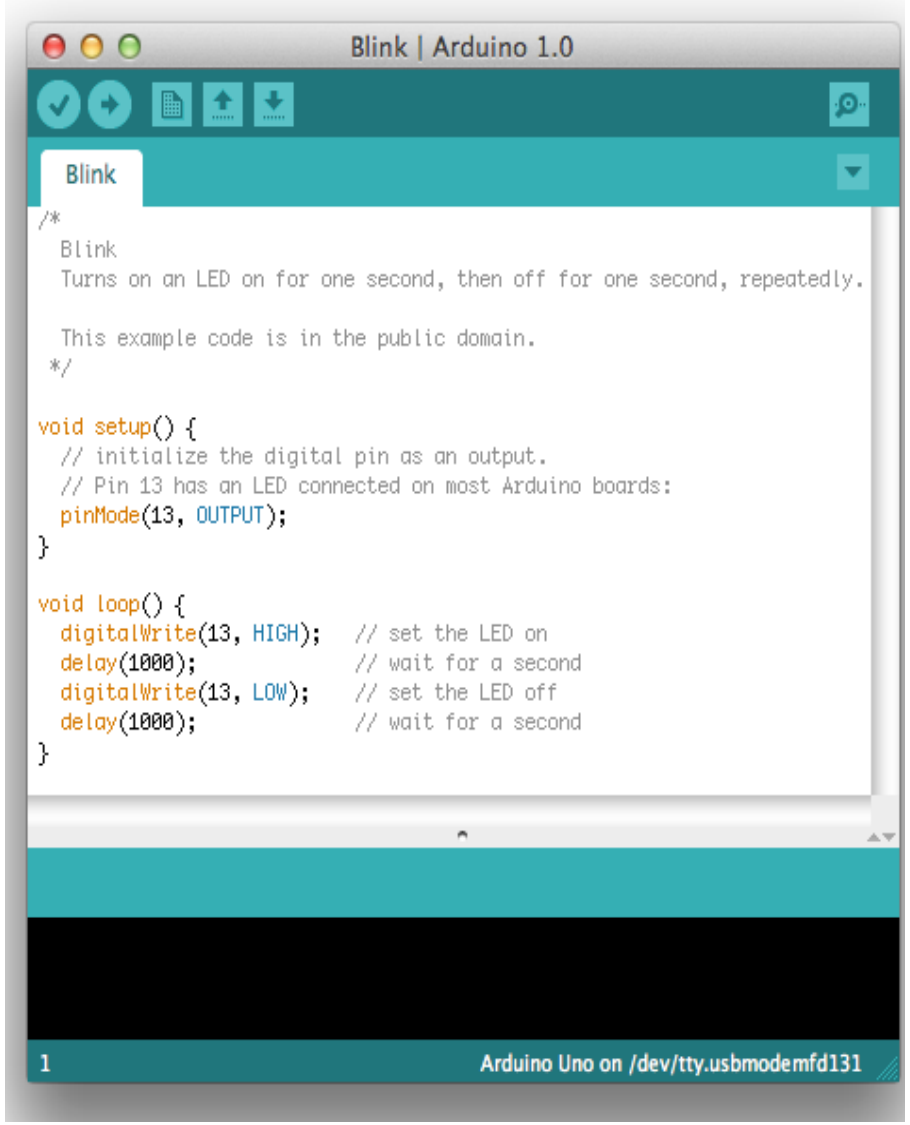
```
void setup(){
  pinMode(13,
  OUTPUT);
}
void loop(){
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
}
```

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

TASK 1

Using LEDs

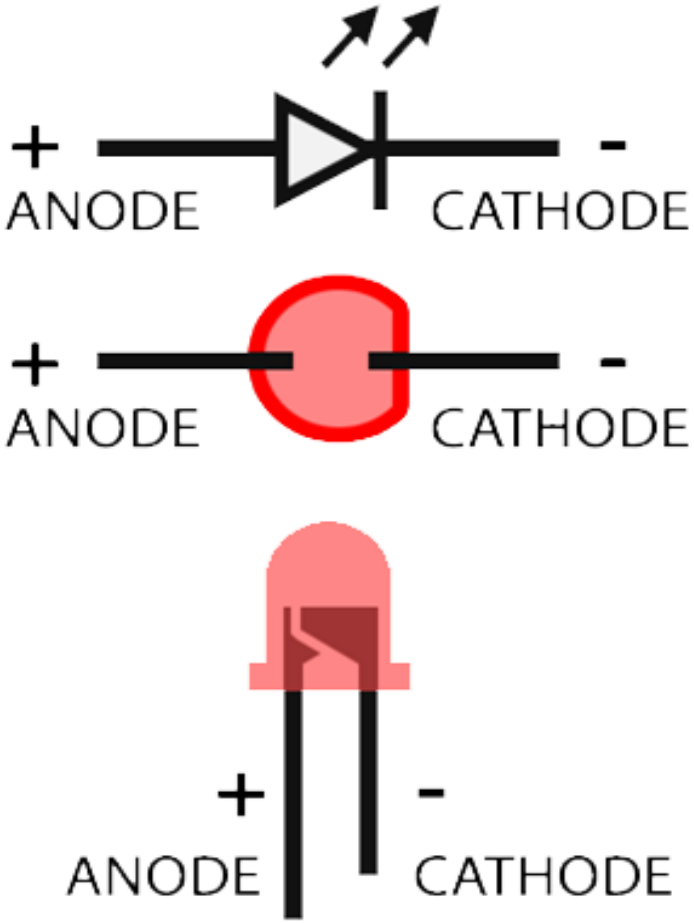
Arduino-Digital Output-LED

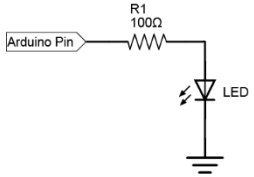


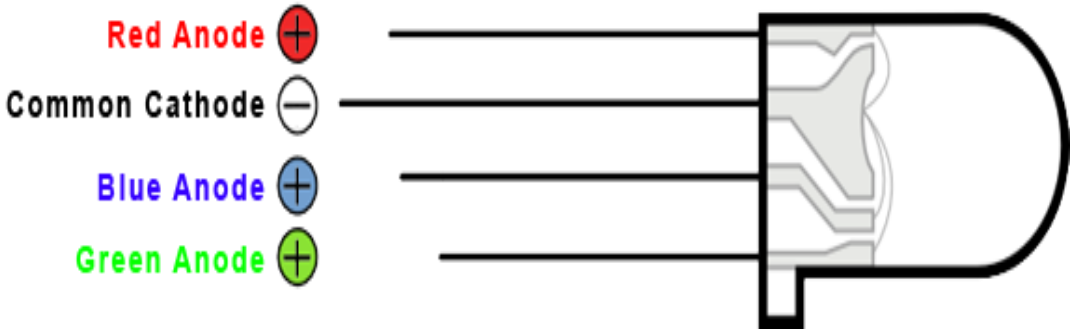
THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

LEDs Information & Specification



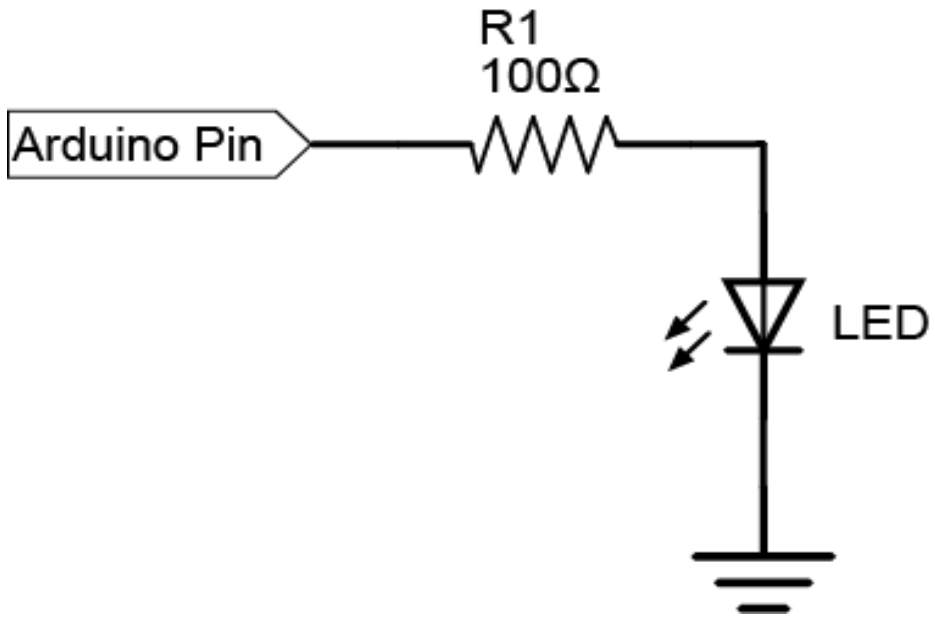


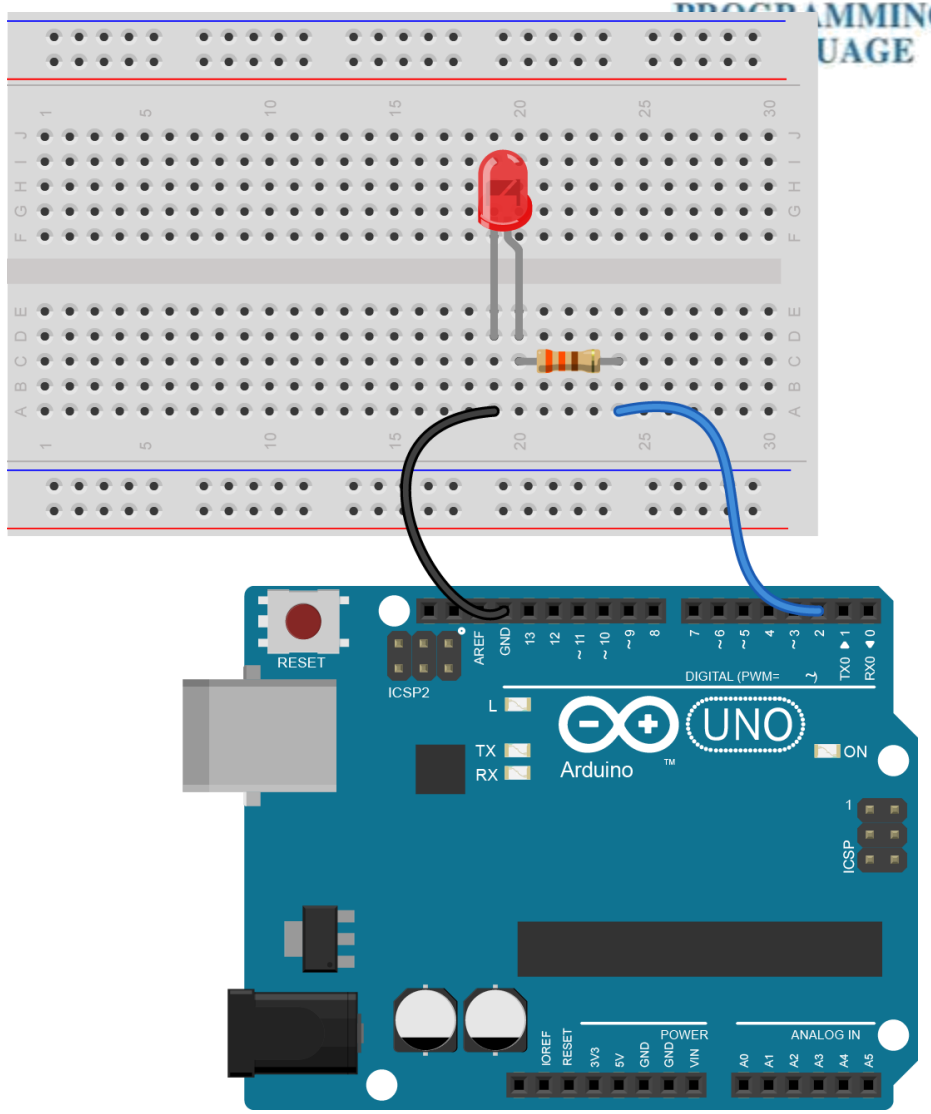


THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

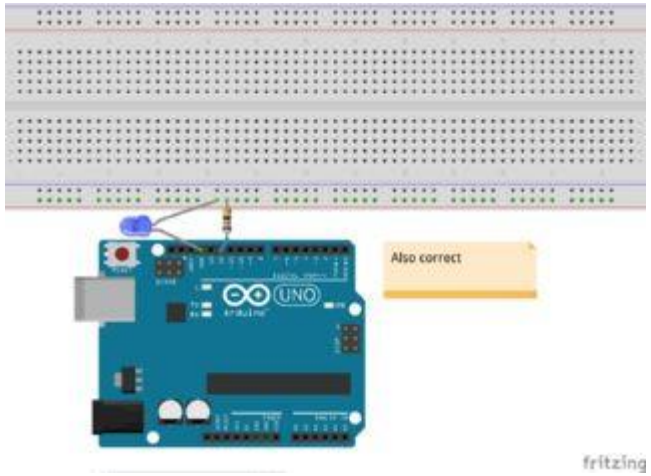
External LEDs

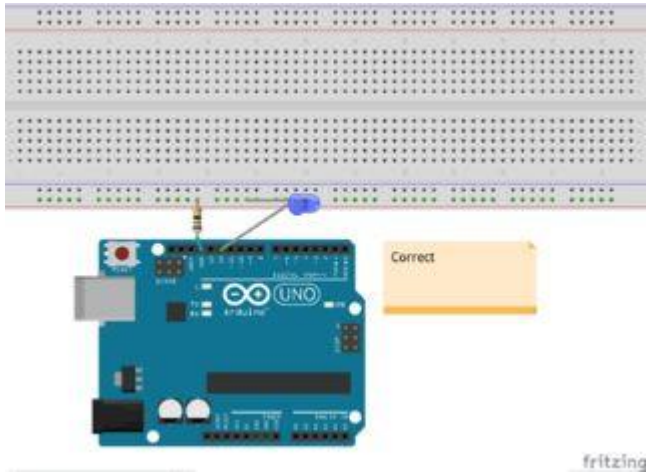


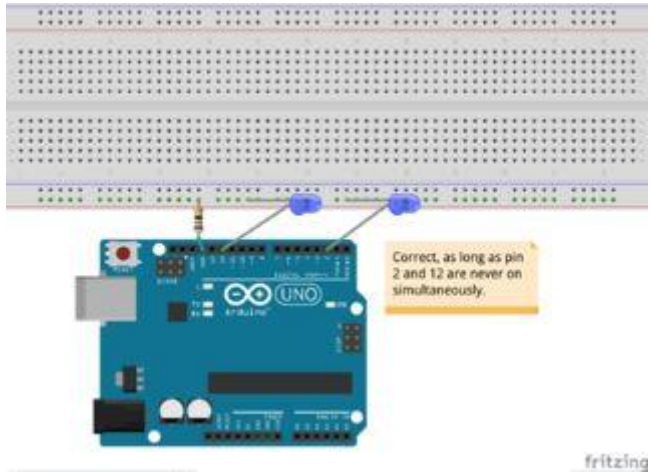


THE C PROGRAMMING LANGUAGE

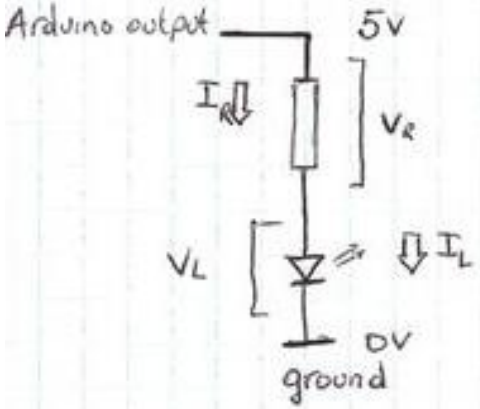
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY







Calculating an LED resistor



$V_R + V_L = 5v \Leftrightarrow V_R = 5v - V_L$

$I_R = I_L \leqslant 20mA$

OHM'S LAW: $R = \frac{V}{I}$

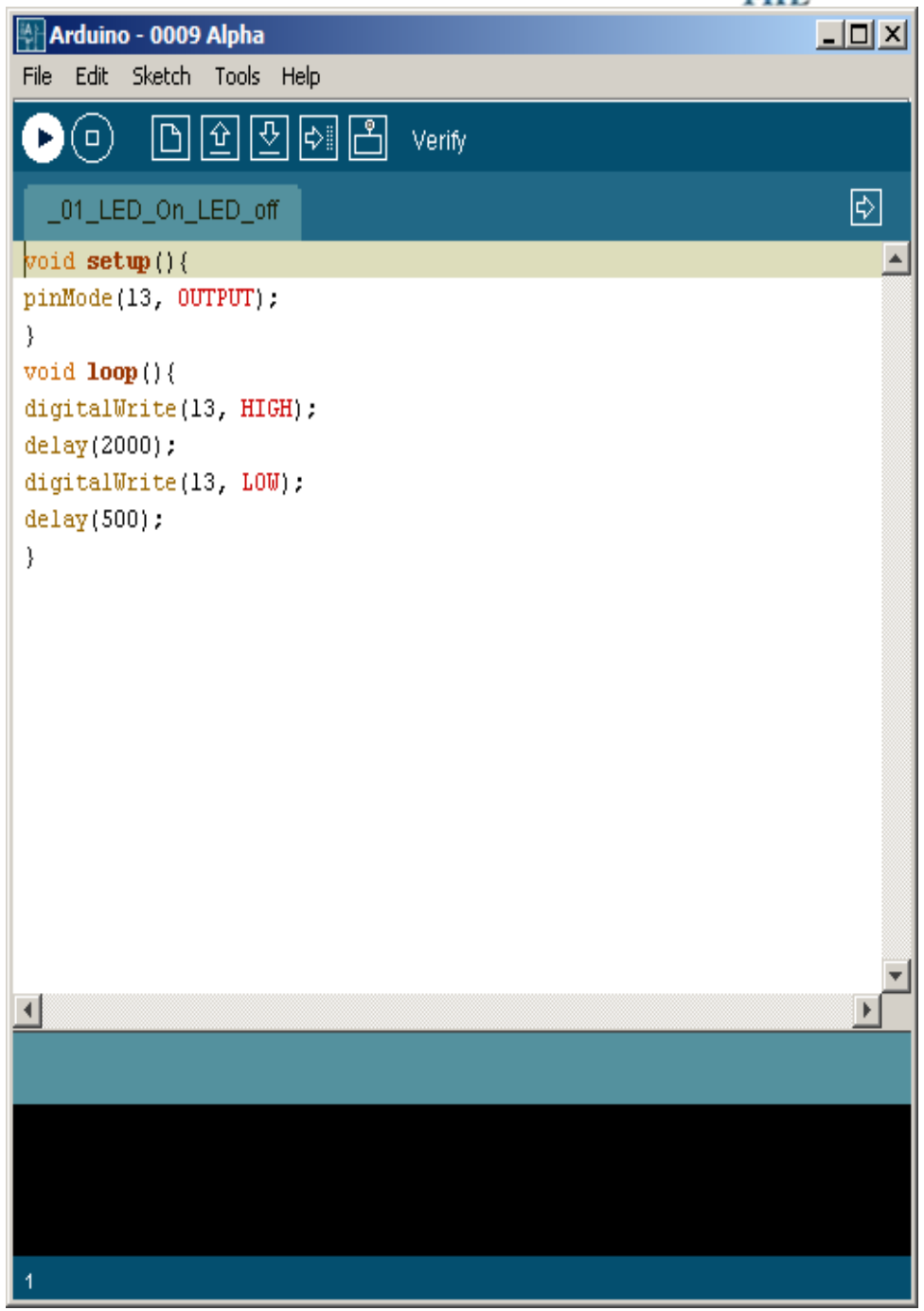
$\Rightarrow R = \frac{5v - V_L}{I_L}$

e.g. a blue LED @ 3.7v, 20mA $\Rightarrow R = \frac{5v - 3.7v}{0.02A} = \frac{1.3v}{0.02A} = 65\Omega$

Color	Forward voltage (V_L)
Red	1.7v - 2.2v
Orange	2.0v
Yellow	2.1v
Green	2.2v
Blue	3.2v - 3.8v
White	3.2v - 3.8v

Arduino-Digital Output-LED

```
void setup(){
pinMode(13, OUTPUT);
}
void loop(){
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```



Arduino Code Basics

- **Arduino programs run on two basic sections:**

```
void setup() {
        //setup motors, sensors etc
}
void loop() {
        // get information from sensors
        // send commands to motors
}
```

SETUP

- **The setup section is used for assigning input and outputs (Examples: motors, LED's, sensors etc) to ports on the Arduino**
- **It also specifies whether the device is OUTPUT or INPUT**
- **To do this we use the command "pinMode"**

```
void setup() {
        pinMode(9, OUTPUT);
}
```

port #

Input or Output

LOOP

```
void loop() {

    digitalWrite(9, HIGH);
    delay(1000);
    digitalWrite(9, LOW);
    delay(1000);

}
```

Port # from setup

Turn the LED on or off

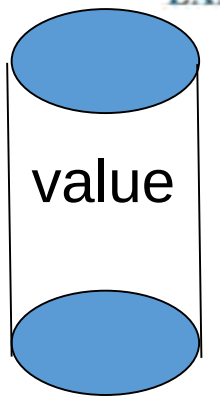
Wait for 1 second or 1000 milliseconds

THE C PROGRAMMING LANGUAGE

江西理工大學 JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Variables

- **A variable is like "bucket"**
- **It holds numbers or other values temporarily**



DECLARING A VARIABLE

```
int val = 5;
```

Type

variable name

assignment
"becomes"

value

USING VARIABLES

THE C PROGRAMMING LANGUAGE

```
int delayTime = 2000;
int greenLED = 9;
void setup() {
      pinMode(greenLED, OUTPUT);
}

void loop() {
      digitalWrite(greenLED, HIGH);
      delay(delayTime);
      digitalWrite(greenLED, LOW);
      delay(delayTime);
}
```

Declare delayTime Variable

Use delayTime Variable

Using Variables

```
int delayTime = 2000;
int greenLED = 9;

void setup() {
      pinMode(greenLED, OUTPUT);
}
void loop() {
      digitalWrite(greenLED, HIGH);
      delay(delayTime);
      digitalWrite(greenLED, LOW);
 delayTime = delayTime - 100;
      delay(delayTime);
}
```

subtract 100 from delayTime to gradually increase LED's blinking speed

VALUE COMPARISONS

THE C PROGRAMMING LANGUAGE

GREATER THAN	GREATER THAN OR EQUAL
a > b	a >= b
LESS	LESS THAN OR EQUAL
a < b	a <= b
EQUAL	NOT EQUAL
a == b	a != b

Conditions

- To make decisions in Arduino code we use an ‘if’ statement
- ‘If’ statements are based on a TRUE or FALSE question

```
if(true)
{
    “perform some action”
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

IF Example

```
int counter = 0;

void setup() {
    Serial.begin(9600);
}
void loop() {

    if(counter < 10)
    {
        Serial.println(counter);
    }
    counter = counter + 1;
     }
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

IF - ELSE Condition

```
if( “answer is true”)
{
    “perform some action”
}
else
{
    “perform some other action”
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

IF - ELSE Example

THE C PROGRAMMING LANGUAGE

```
int counter = 0;
void setup() {
      Serial.begin(9600);
}
void loop() {

      if(counter < 10)
      {
            Serial.println("less than 10");
      }
      else
      {
            Serial.println("greater than or equal to 10");
            Serial.end();
      }
      counter = counter + 1;
}
```

IF - ELSE IF Condition

THE C PROGRAMMING LANGUAGE

```
if( “answer is true”)
{
    “perform some action”
}
else if( “answer is true”)
{
    “perform some other action”
}
```

IF - ELSE Example

```
int counter = 0;
void setup() {
      Serial.begin(9600);
}
void loop() {

      if(counter < 10)
      {
            Serial.println("less than 10");
      }
      else if (counter == 10)
      {
            Serial.println("equal to 10");
      }
      else
      {
            Serial.println("greater than 10");
            Serial.end();
      }
      counter = counter + 1;
}
```

THE C PROGRAMMING LANGUAGE
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

BOOLEAN OPERATORS - AND

- If we want all of the conditions to be true we need to use ‘AND’ logic (AND gate)
- We use the symbols **&&**

  - Example

```
if ( val > 10 && val < 20)
```

BOOLEAN OPERATORS - OR

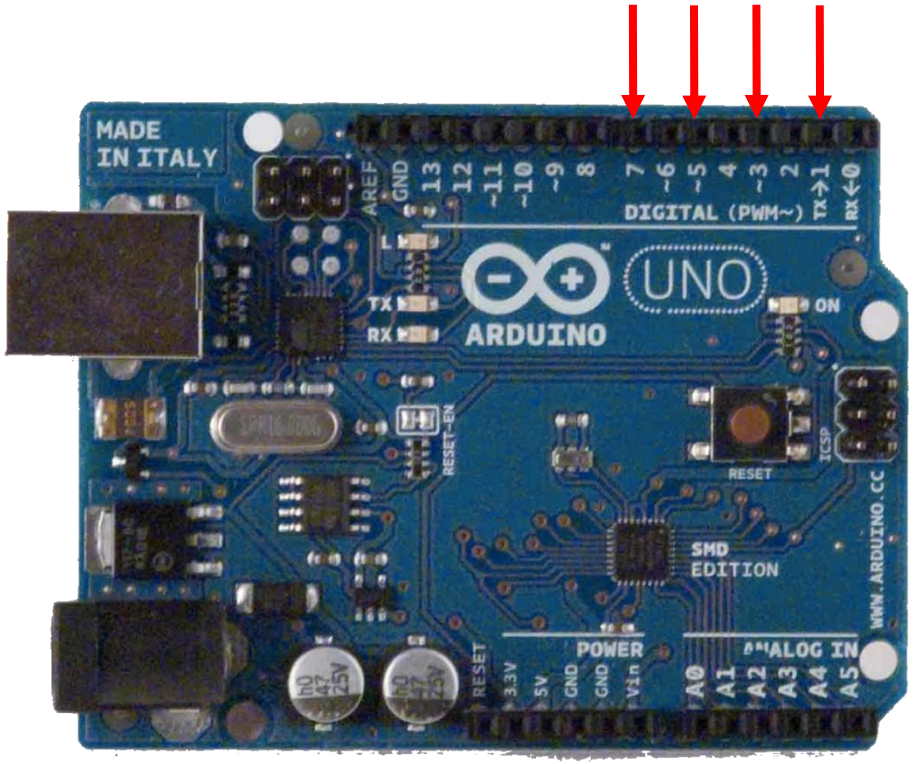
- If we want either of the conditions to be true we need to use ‘OR’ logic (OR gate)
- We use the symbols ||

  - Example

```
if ( val < 10 || val > 20)
```

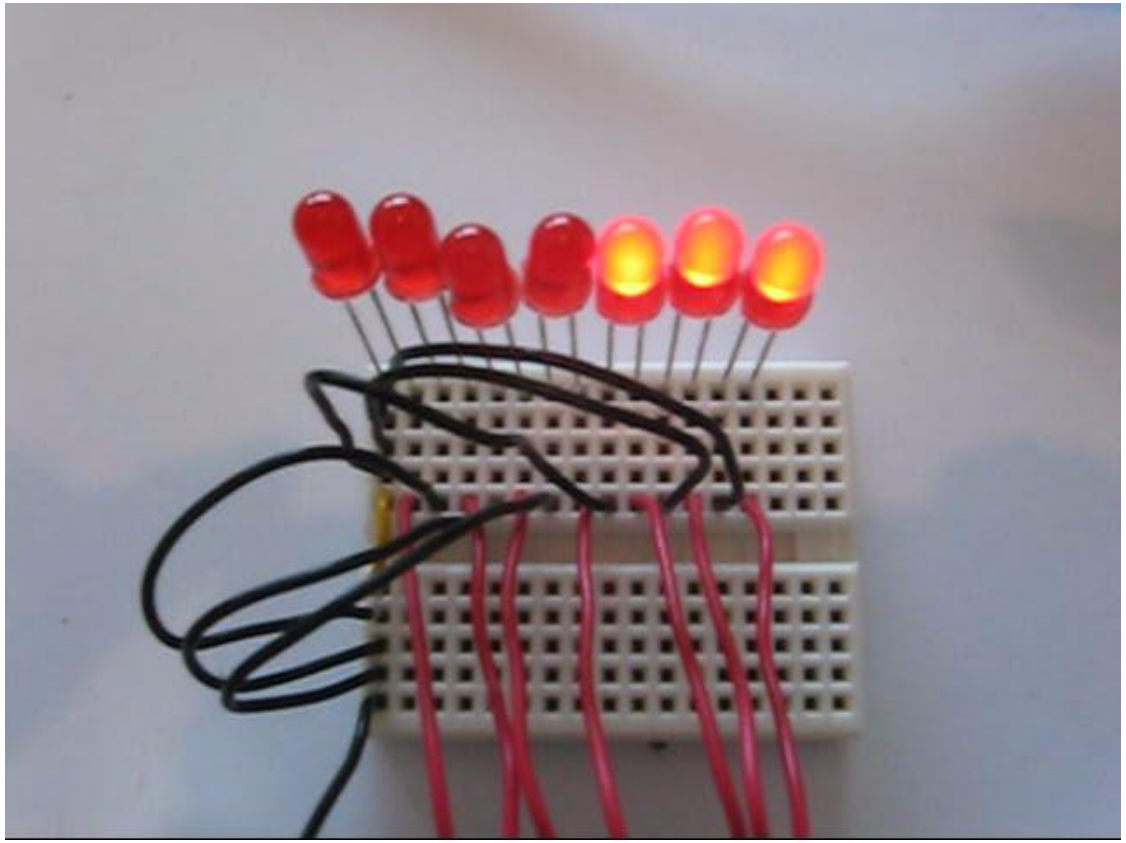
4 LED Blink Sketch

```
void setup(){
  pinMode(1,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(7,OUTPUT);
}
```




```
void loop(){
  digitalWrite(1,
    HIGH);
  delay(200);
  digitalWrite(1,LOW);
  digitalWrite(3,
    HIGH);
  delay(200);
  digitalWrite(3,LOW);
  digitalWrite(5,
    HIGH);
  delay(200);
  digitalWrite(5,LOW);
  digitalWrite(7,HIGH);
  delay(200);
  digitalWrite(7,LOW);
}
```

Arduino-Controlling Multiple Actuators separately from different output pins-Sequencing



THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-Controlling Multiple Actuators separately from different output pins-Random Patterns

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on/off randomly
  int signal=int(random(0,2));
  digitalWrite(i,signal); //Sending High Signal to Pin
  }
  delay(1000); //Wait 1 second
}
```

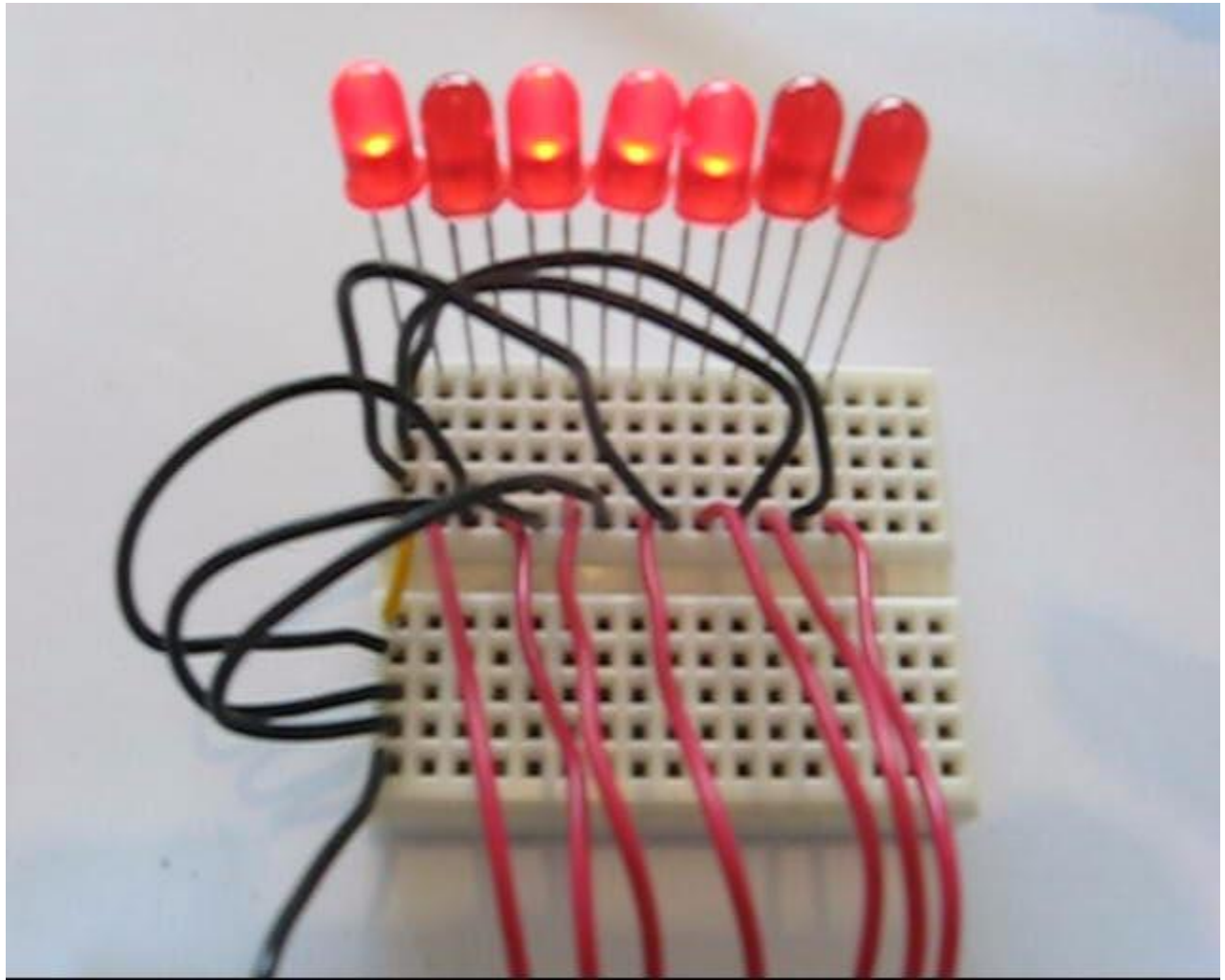
Arduino-Controlling Multiple Actuators separately from different output pins-Random Patterns

THE C PROGRAMMING LANGUAGE

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on/off randomly
  int signal=int(random(0,2));
  digitalWrite(i,signal); //Sending High Signal to Pin
  }
  delay(1000); //Wait 1 second
}
```

Arduino-Controlling Multiple Actuators separately from different output pins-Random Patterns

THE C PROGRAMMING LANGUAGE



江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-Controlling Multiple Actuators separately from different output pins-Random Patterns

THE C PROGRAMMING LANGUAGE

```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on/off randomly
  int signal=int(random(0,2));
  digitalWrite(i,signal); //Sending High Signal to Pin
  }
  delay(1000); //Wait 1 second
}
```

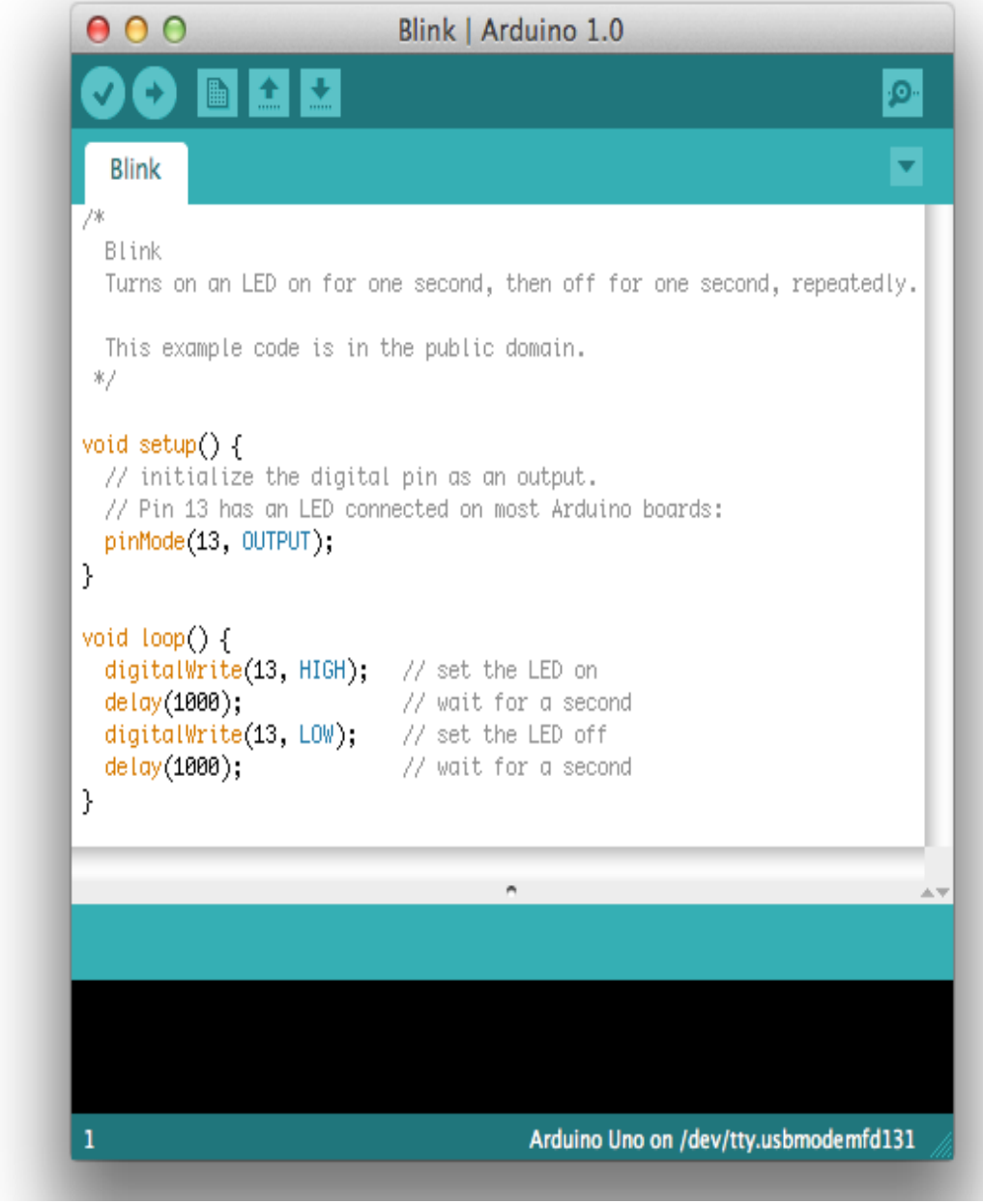
```
void setup(){
  pinMode(2, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(3, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(4, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(5, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(6, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(7, OUTPUT); // Specify Arduino Pin number and output/input mode
  pinMode(8, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=2; i<9; i++){//iterating through pin 2 to 8 and turning them on one by one
  digitalWrite(i,HIGH); //Sending High Signal to Pin
  delay(1000); //Wait 1 second
  }
  for(int i=9; i>2; i--){//iterating through pin 8 to 2 and turning them off one by one
  digitalWrite(i,LOW); //Sending LOW Signal to Pin
  delay(1000); //Wait 1 second
  }
}
```

Aside from introduction of randomness, payattention to how changing the place of delay() function can change the systems behavior. Here we put the delay function out side of the for loop. As a result instead of seeing the change for each actuator one by one in a sequence, which is the case in the previous exercise, here, at first all the actuators(LEDs) are configured together and then the system pauses for one second to let us see the over all configuration.

TASK 2

Using LEDs

Arduino-Analog Output-LED



Analog Output

- Can a digital devise produce analog output?

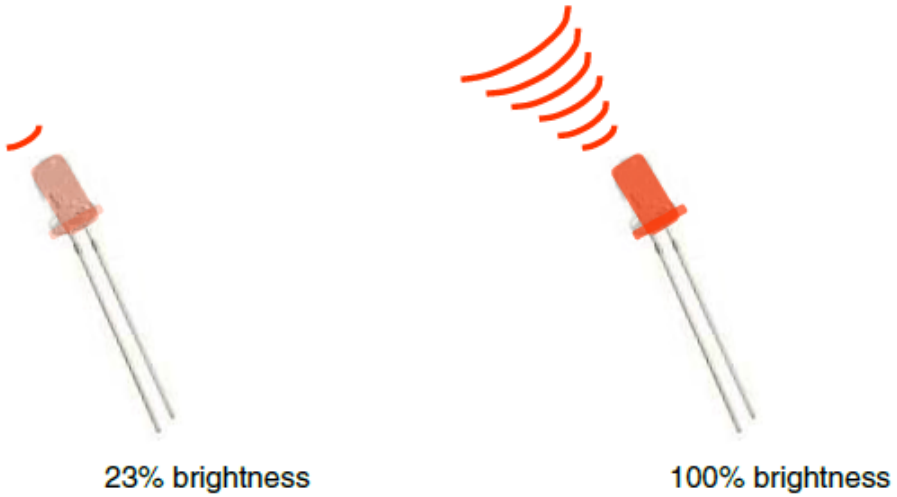


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

- **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

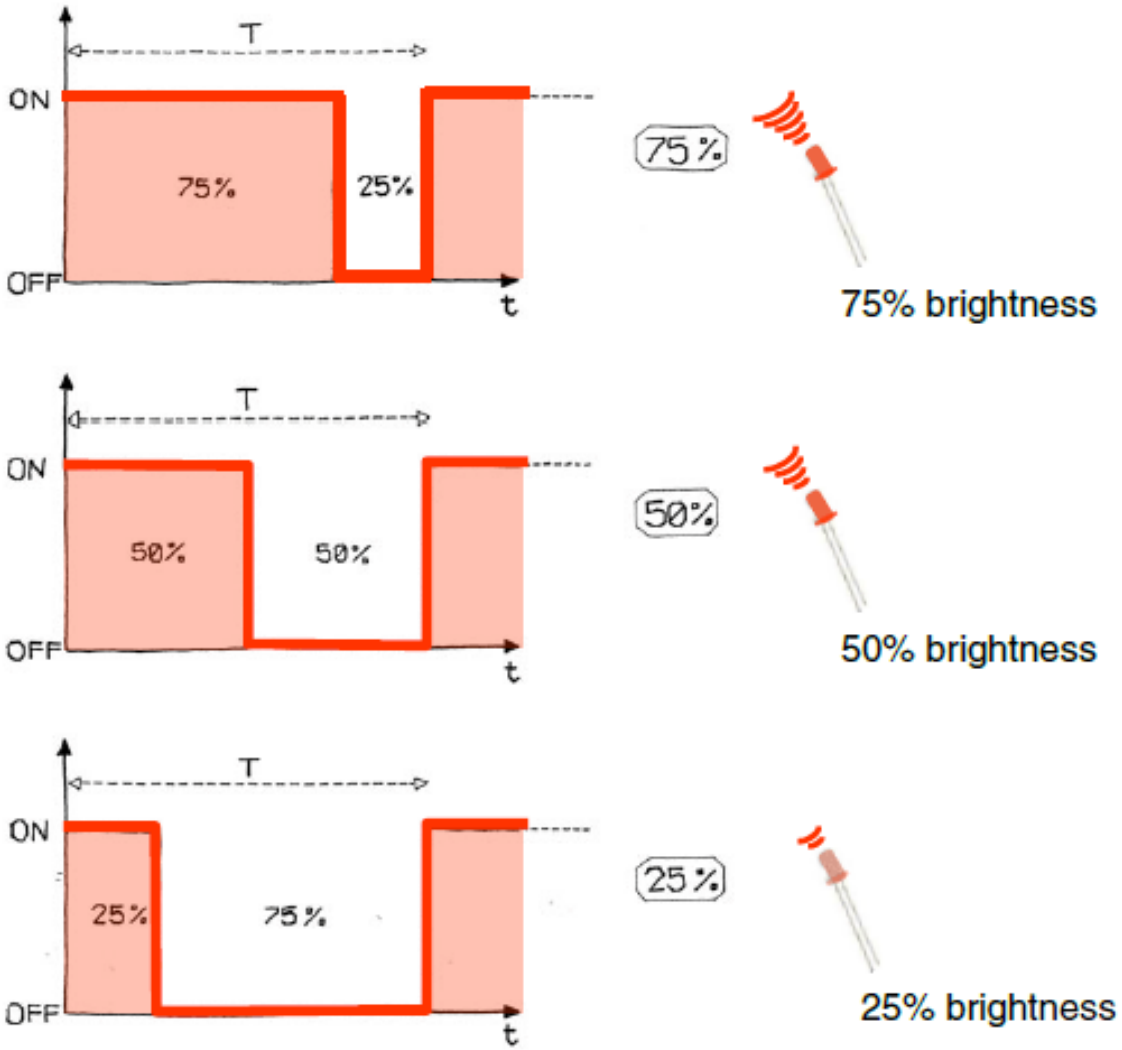


Image from *Theory and Practice of Tangible User Interfaces* at UC Berkley

PWM Duty Cycle

output voltage = (on_time / cycle_time) * 5V

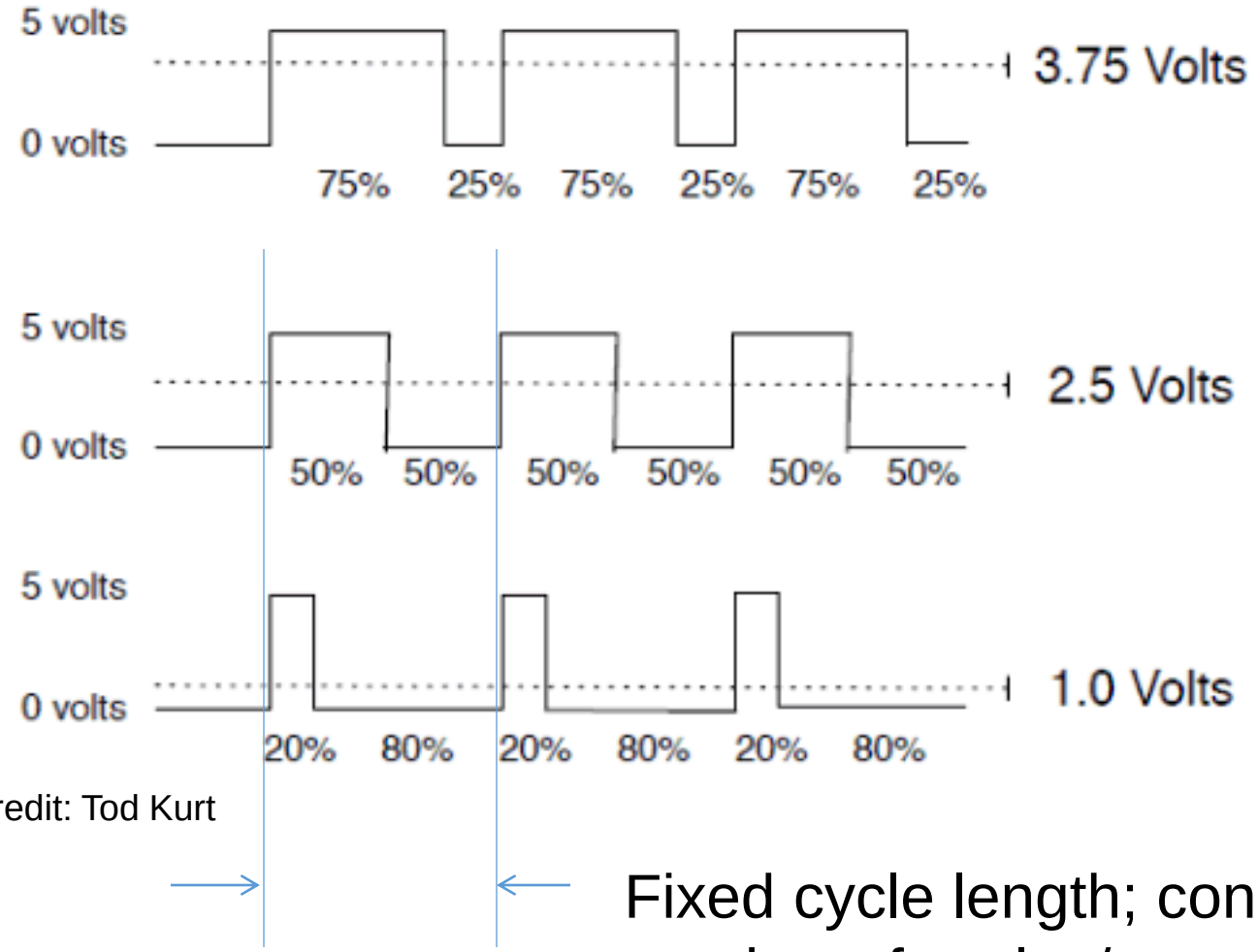
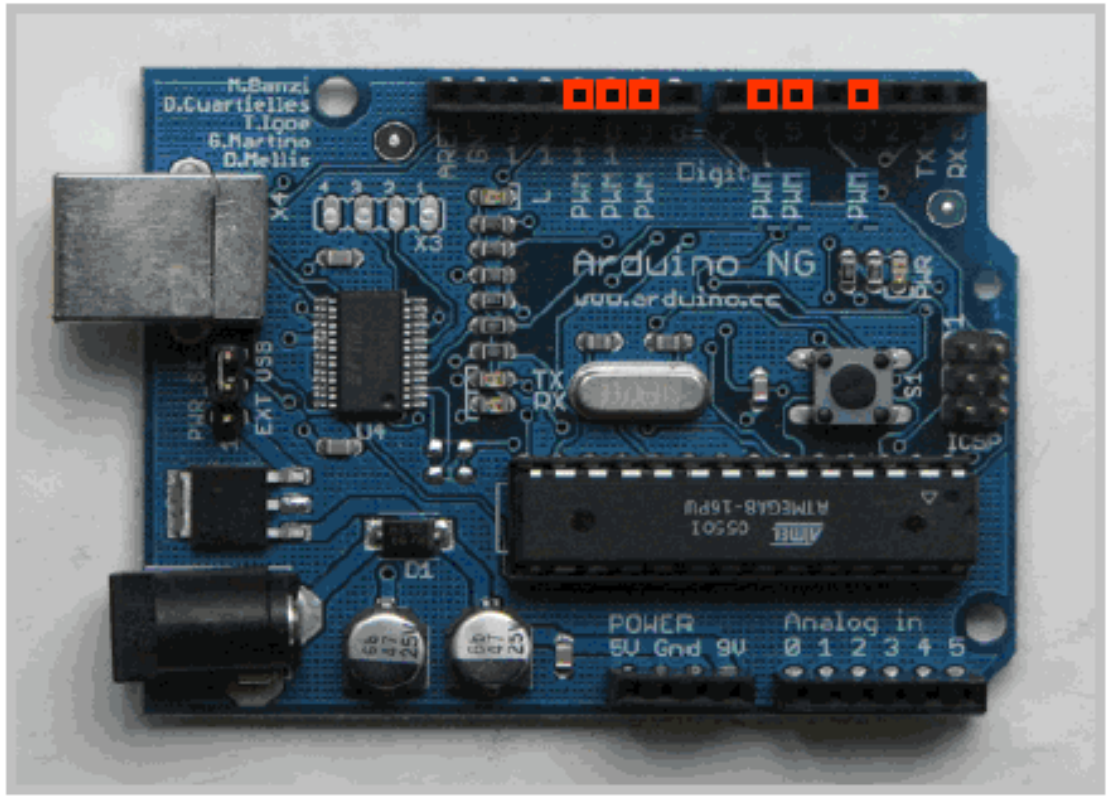


Image credit: Tod Kurt

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

THE C PROGRAMMING LANGUAGE

江西理工大学
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

PWM EXAMPLE

```
int ledPin;
void setup()
{
      ledPin = 10;
      //Note that PWM doesn't need a pinMode
}
void loop()
{
      analogWrite(ledPin, 50);
      delay(500);
      analogWrite(ledPin, 255);
      delay(500);
  }
```

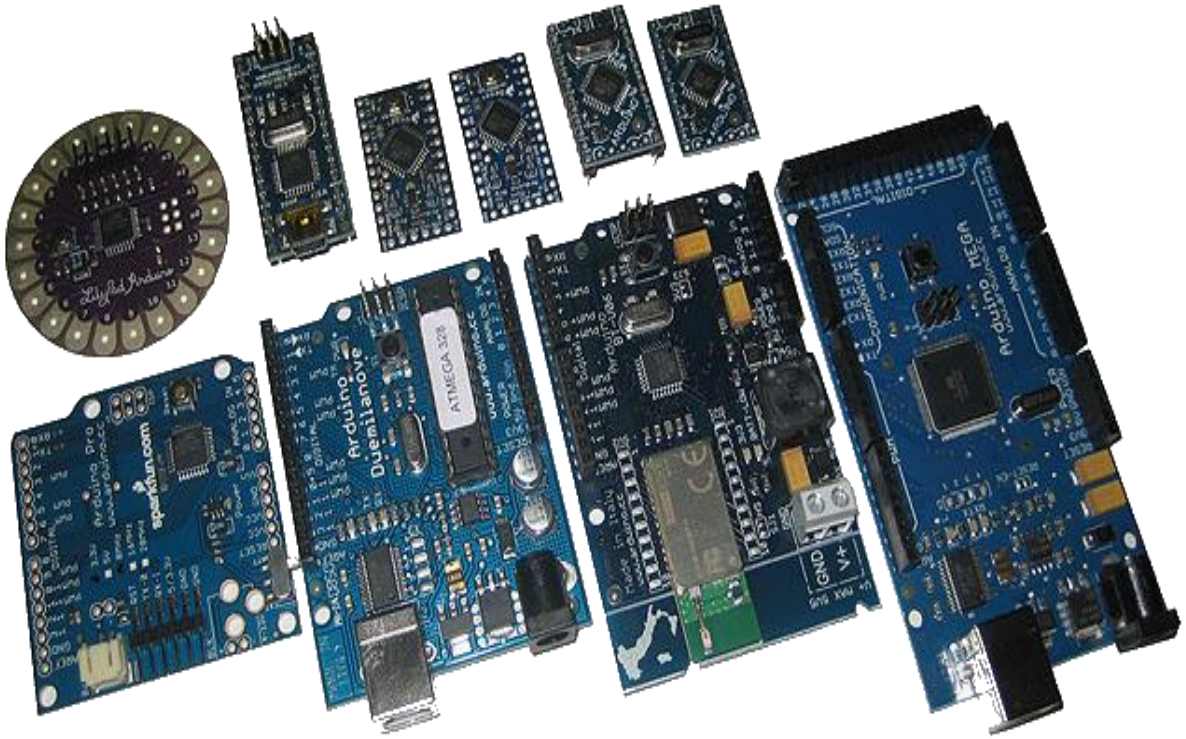
THE C PROGRAMMING LANGUAGE

江西理工大 JIANGXI UNIVERSITY OF SCIENCE AND TECH

مرکز آموزش علمی-کاربردی گروه انتخاب

Lecture 3:

Serial Communication & Introduction Analog Input



Serial Communication

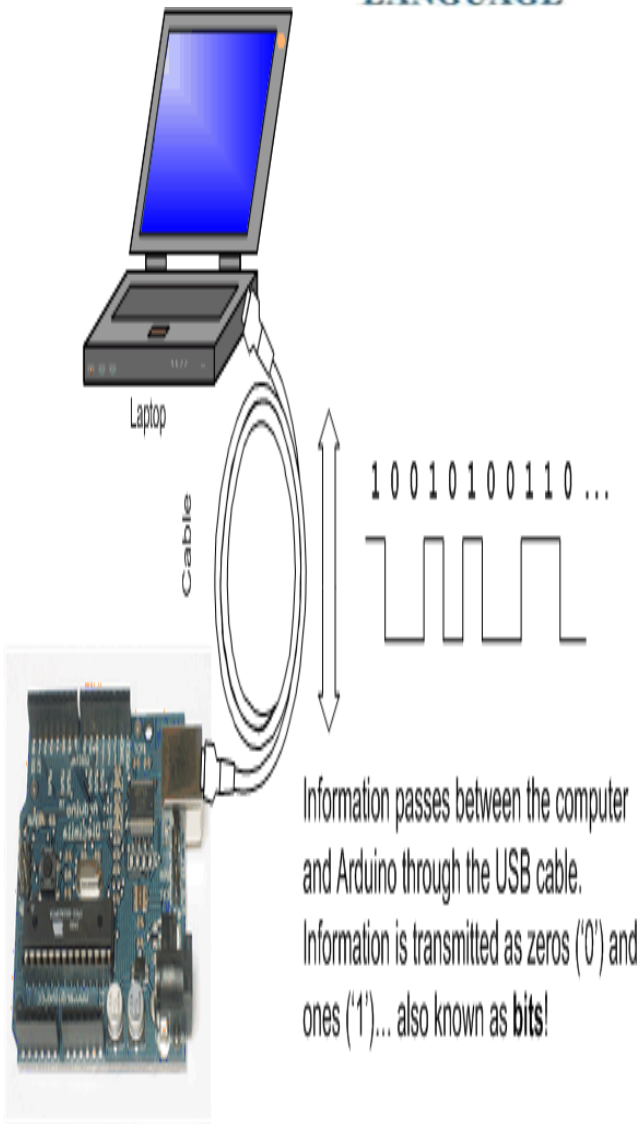
Serial Communication is the transferring and receiving of information between two machines, the Arduino dedicates pin # 0 to receiving information and pin 1 to transferring information

THE C PROGRAMMING LANGUAGE

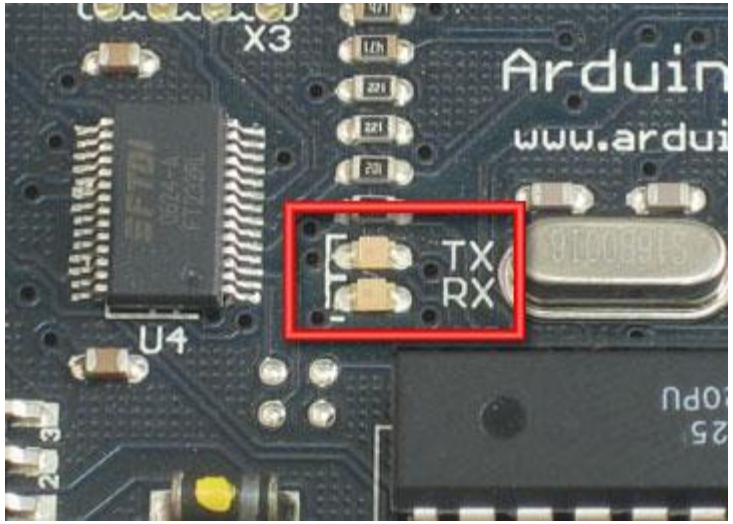
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

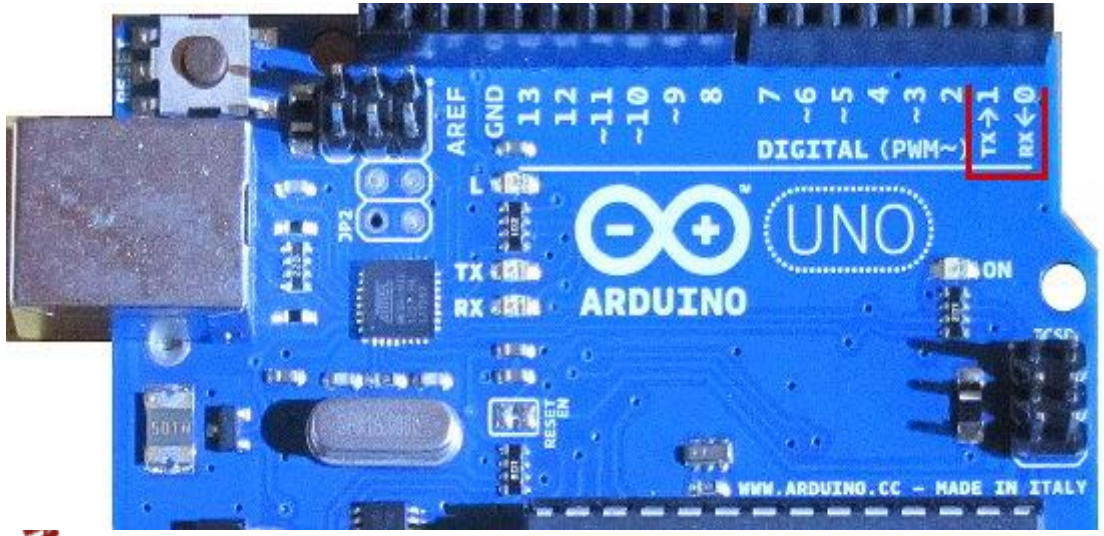
What is Serial?

- **Serial** may sound like a tasty breakfast food, but its actually quite different. The word **serial** means "one after the other." For example, a serial killer doesn't stop with one murder, but stabs many people one after the other. Serial data transfer is when we transfer data one **bit** at a time, one right after the other.
- Information is passed back & forth between the computer and Arduino by, essentially, setting a pin high or low. Just like we used that technique to turn an LED on and off, we can also send data. One side sets the pin and the other reads it.

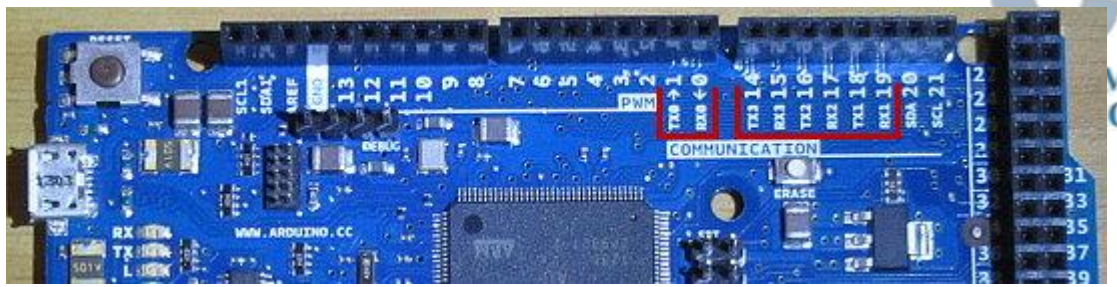


Next time you upload a sketch, look carefully at the two LEDs near the USB connector, they'll blink when data is being transmitted. One blinks when the Arduino is receiving data **(RX)** and one blinks when the Arduino is transmitting data (**TX**)

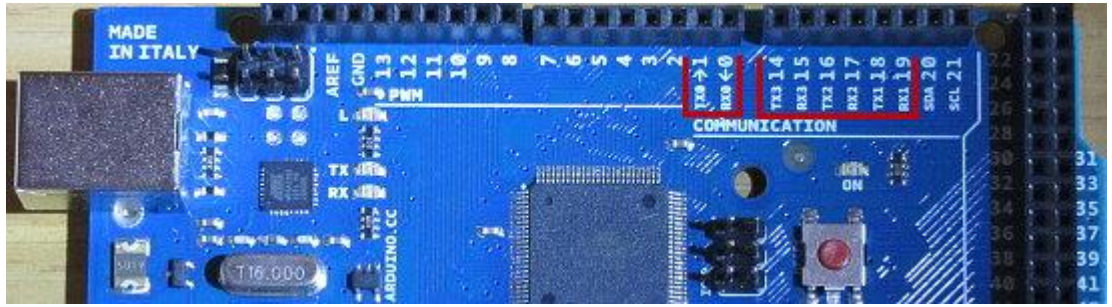




Arduino Due Serial Ports



Arduino MEGA 2560 Serial Ports



Both the MEGA 2560 and Due have 4 serial ports in total. One that connects through a USB port chip to the USB device port on the board and three extra serial ports that connect to pins on one of the pin headers of the board.

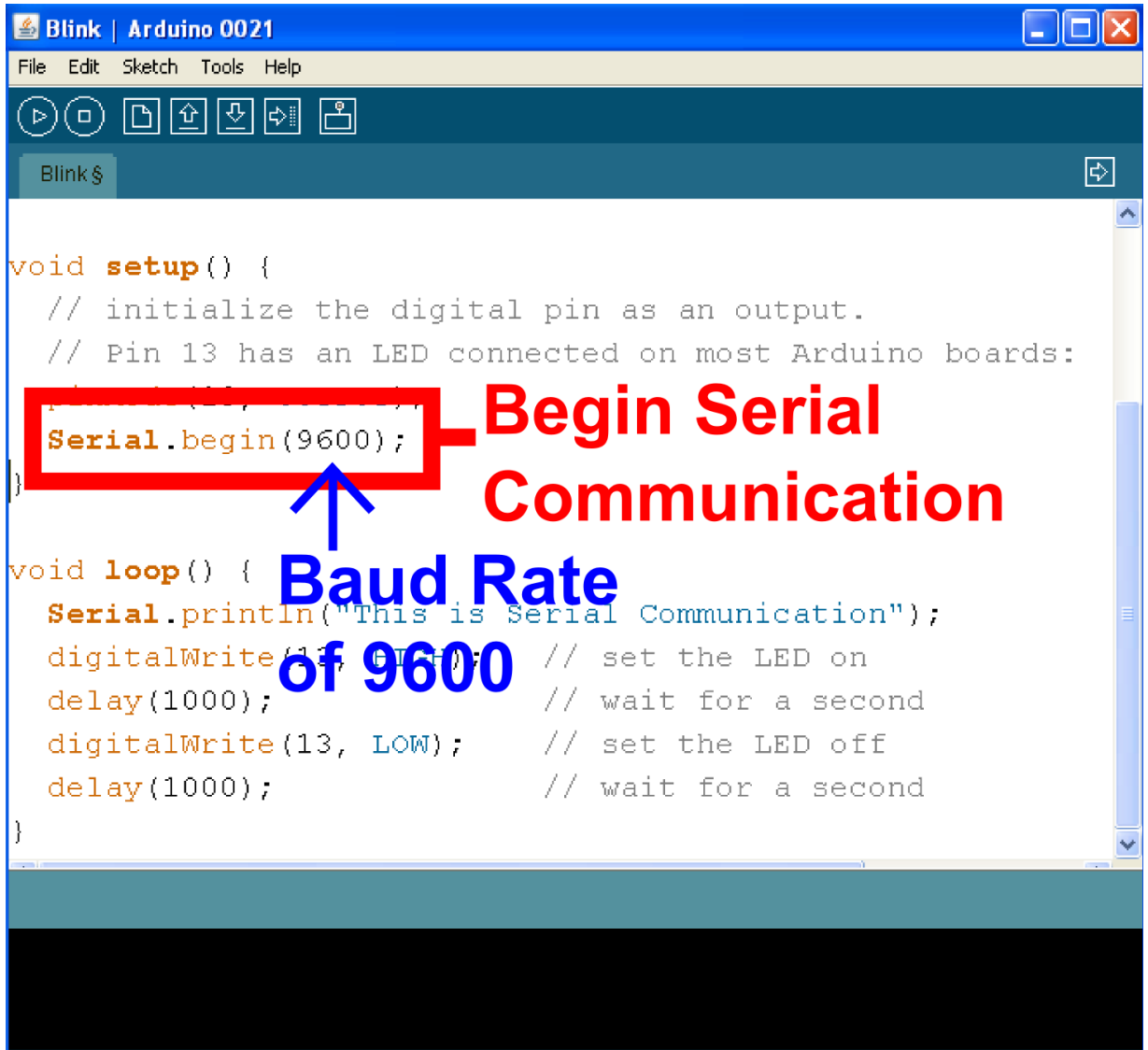
- Pins 0 and 1 of the Due and MEGA connect serial port 0 through to the USB device port so that these Arduino boards are compatible with the pin numbering of the Uno and therefore with Arduino shields.
- The extra serial ports are ports 1 to 3 with each port having a transmit and receive pin.
- It is important to be aware that the MEGA 2560 serial port pins use 5V voltage levels, but the Due uses 3.3V voltage levels

Input & Output

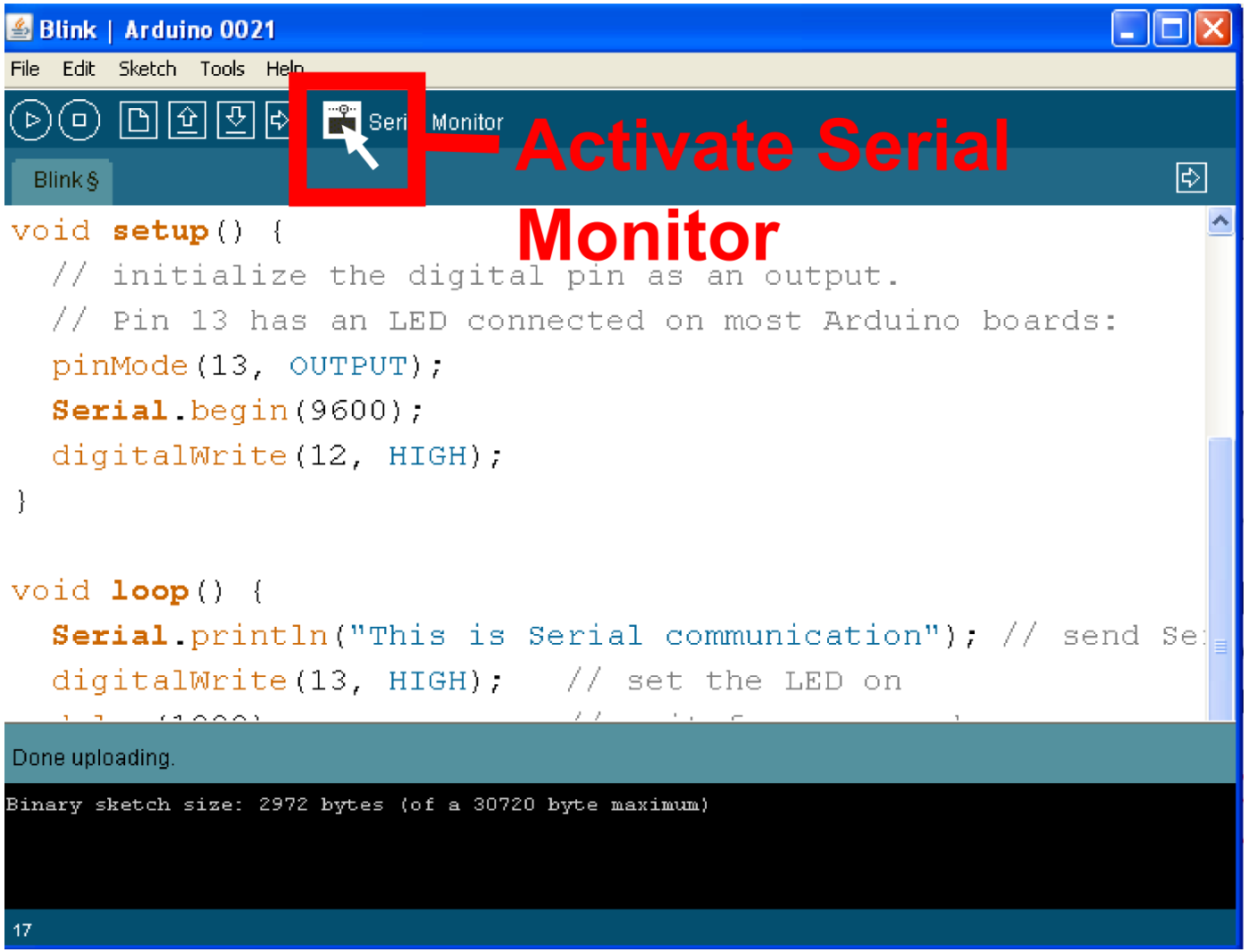
- Transferring data from the computer to an Arduino is done using **Serial Transmission**
- To setup Serial communication we use the following

```
void setup()
{
    Serial.begin(9600);
}
```

Serial in Setup



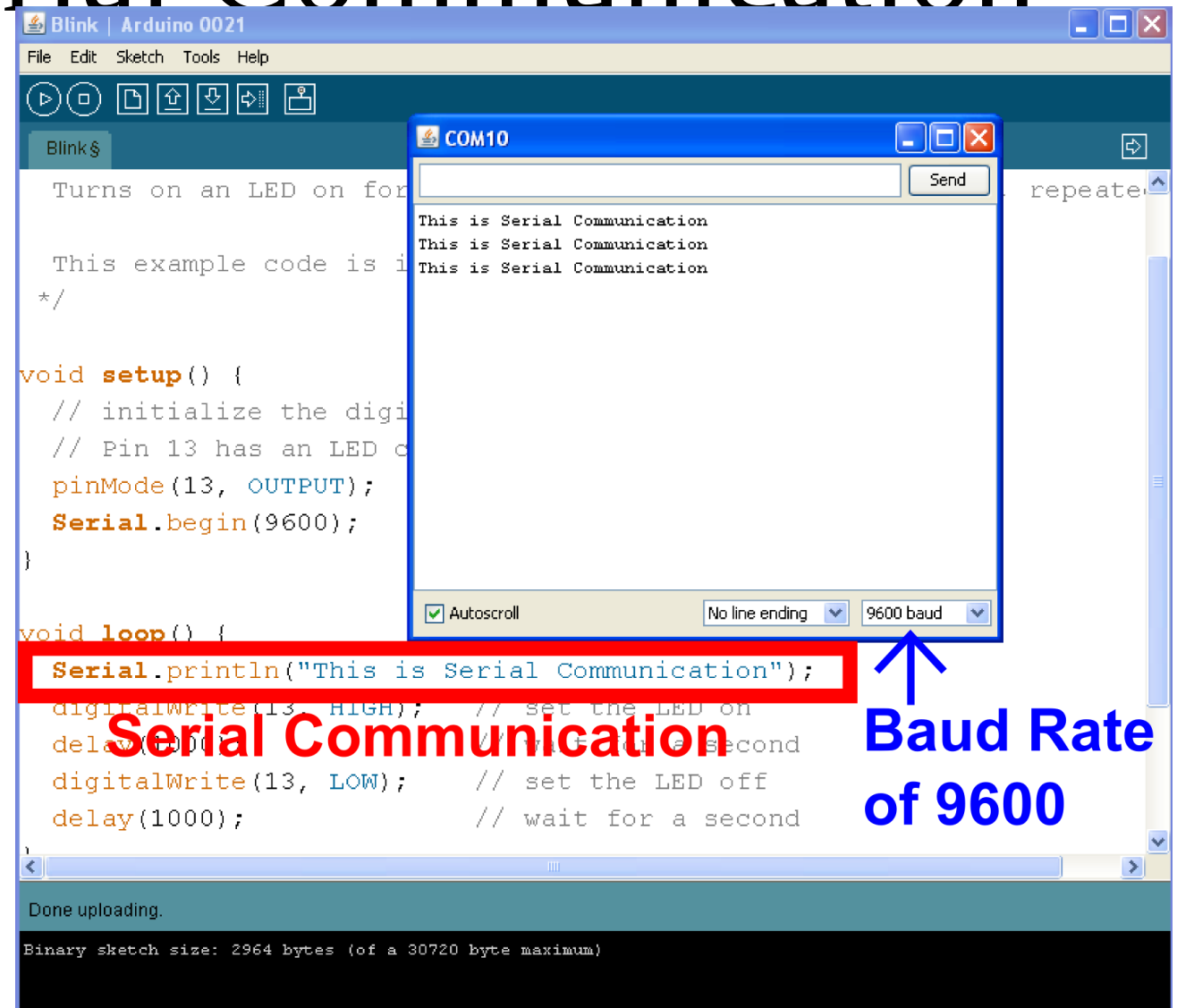
Serial Monitor



THE C PROGRAMMING LANGUAGE

江西理工大學 JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Serial Communication



Serial Communication

Baud Rate of 9600

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Serial Communication: Serial Setup

```
void setup ( ) {
Serial.begin ( 9600 ) ;
}
```

In this case the number 9600 is the baudrate at which the computer and Arduino communicate

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Serial Communication: Sending a Message

```
void loop ( ) {
Serial.print ( “Constructivism & “ ) ;

Serial.println ( “Mechatronics” ) ;
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Serial Communication: Serial Debugging

```
void loop ( ) {
int xVar = 10 ;
Serial.print ( “Variable xVar is “ ) ;
Serial.println ( xVar ) ;
}
```

THE C PROGRAMMING LANGUAGE

Serial Communication: Serial Troubleshooting

```
void loop ( ) {
Serial.print ( “Digital pin 9 reads “ ) ;
Serial.println ( digitalRead ( 9 ) ) ;
}
```

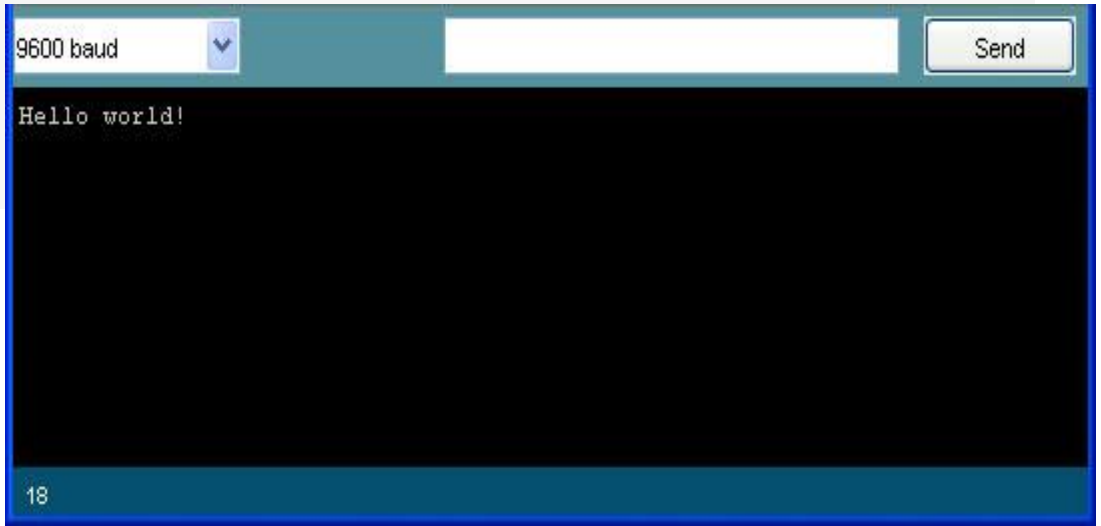
Serial Communication: Circuit 7 code

```
void loop ( ) {
buttonState = digitalRead(inputPin);
if (buttonState== HIGH){
        digitalWrite (ledPin, LOW)
        }
        else     {
        digitalWrite(ledPin, HIGH);
        Serial.print ("button state is ");
        Serial.println ( buttonState );
        }
}
```

Sending Data to PC from the Arduino

THE C PROGRAMMING LANGUAGE

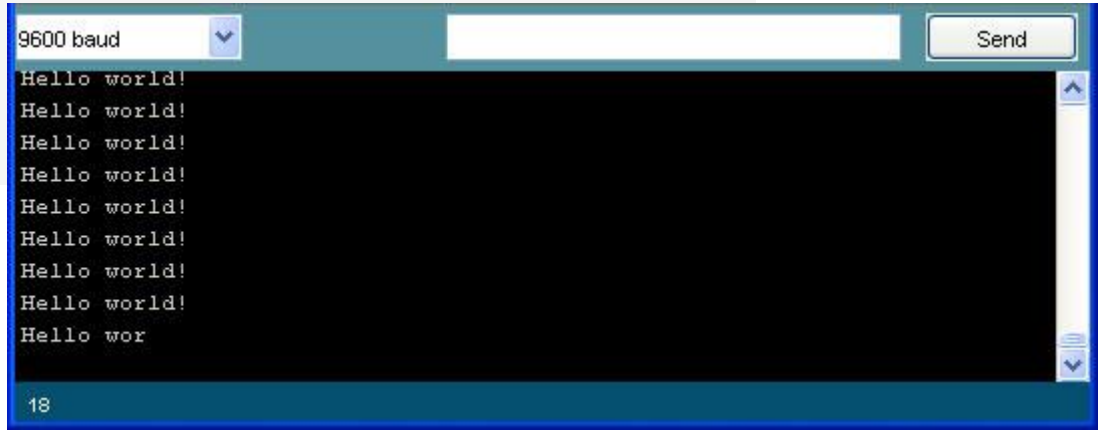
```
void setup() // run once, when the sketch starts
        {
Serial.begin(9600); // set up Serial library at 9600 bps
 Serial.println("Hello world!"); // prints hello with ending line break
        }
void loop() // run over and over again
        {
// do nothing!
        }
```



Hello world in Loop

```
void setup() // run once, when the sketch starts
 {
Serial.begin(9600); // set up Serial library at 9600 bps
 }
void loop() // run over and over again
 {
Serial.println("Hello world!"); // prints hello with ending line break
delay(1000);

 }
```



Sending Data from the Arduino to the PC

```
char rx_byte;
 void setup() {
 Serial.begin(9600); }
 void loop() {
 if (Serial.available() > 0) {
 rx_byte = Serial.read();
 Serial.print("You typed: ");
 Serial.println(rx_byte); } }
```

This program receives data from the PC and then transmits it back to the PC with an additional message. It demonstrates receiving and transmitting data on the Arduino board.

In the Arduino IDE, verify and then upload the serial_rx_msg program to the Arduino board. Start the serial monitor program and enter a text character or sentence in the top field (to the left of the Send button). Click the Send button (or press Enter) to send the character or sentence.

INPUT EXAMPLE

```
int val = 0;
int greenLED = 13;

void setup() {
   Serial.begin(9600);
  pinMode(greenLED, OUTPUT);
     }
void loop() {
  if(Serial.available()>0)  {
   val = Serial.read();
   Serial.println(val);
 }
 if(val == 53)  {
   digitalWrite(greenLED, HIGH);
 }
  else  {
  digitalWrite(greenLED, LOW); }}
```

THE C PROGRAMMING LANGUAGE

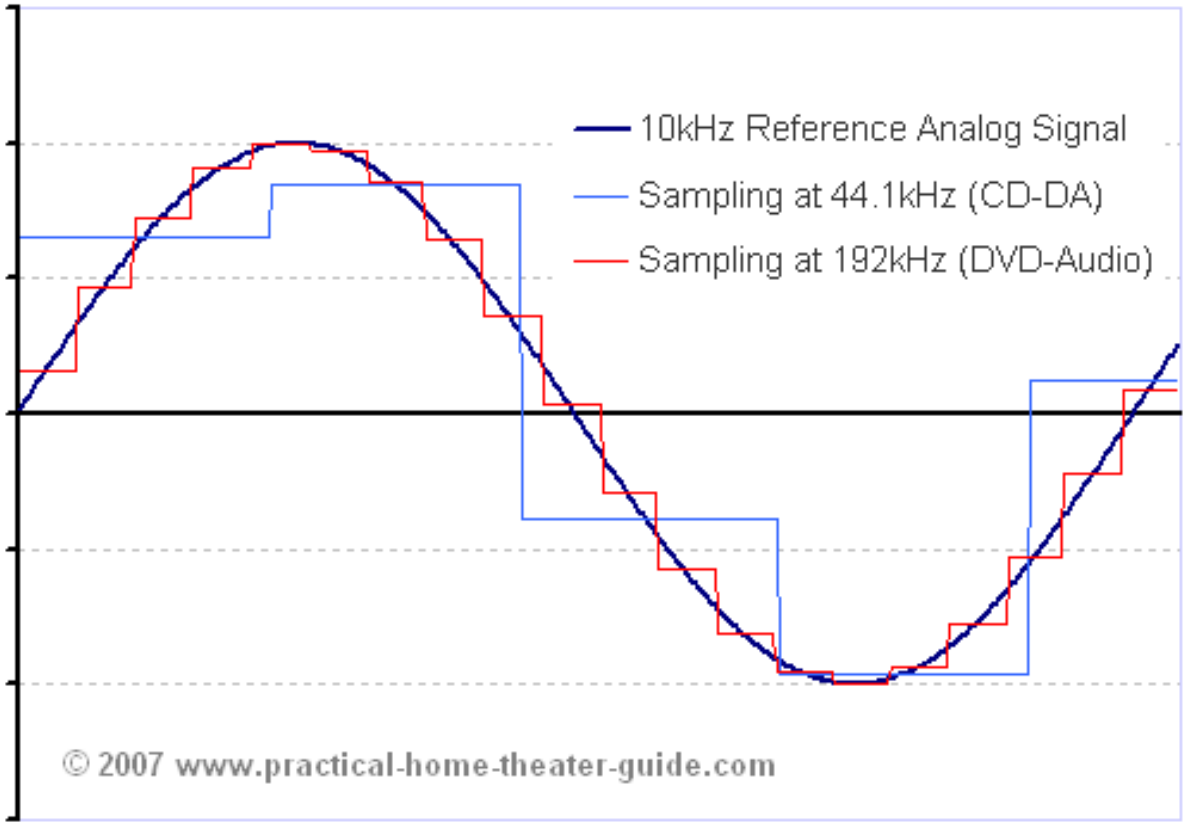
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Using switches and buttons

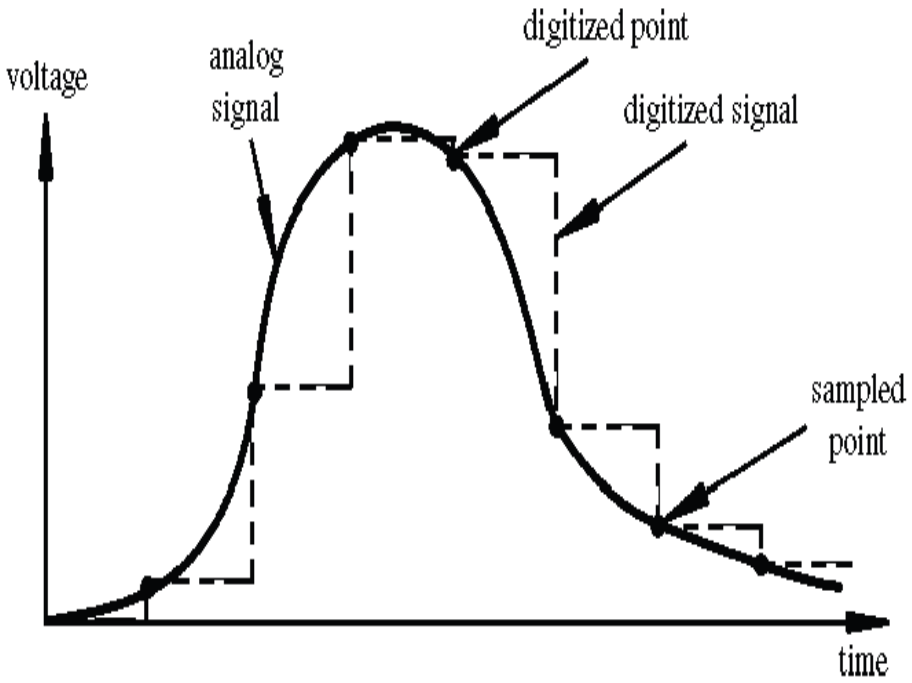
```
const int inputPin = 2; // choose the input
 pin
void setup() {
pinMode(inputPin, INPUT); // declare
 pushbutton as input
}
void loop(){
int val = digitalRead(inputPin); // read
 input value
}
```

Analog Read/Write

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



Arduino Analog Input



- **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**

Analog Inputs

- Most Arduinos have between 6 and 12 analog input pins. They can measure voltages ranging from 0v to the input voltage (5v or 3v3).
- An the board, these pins are labeled A0-A5, and in case of the Leonardo, some of the pins on the digital side can also be used as analog inputs. They are marked with a dot, and the name (A6-A11) is written on the back.

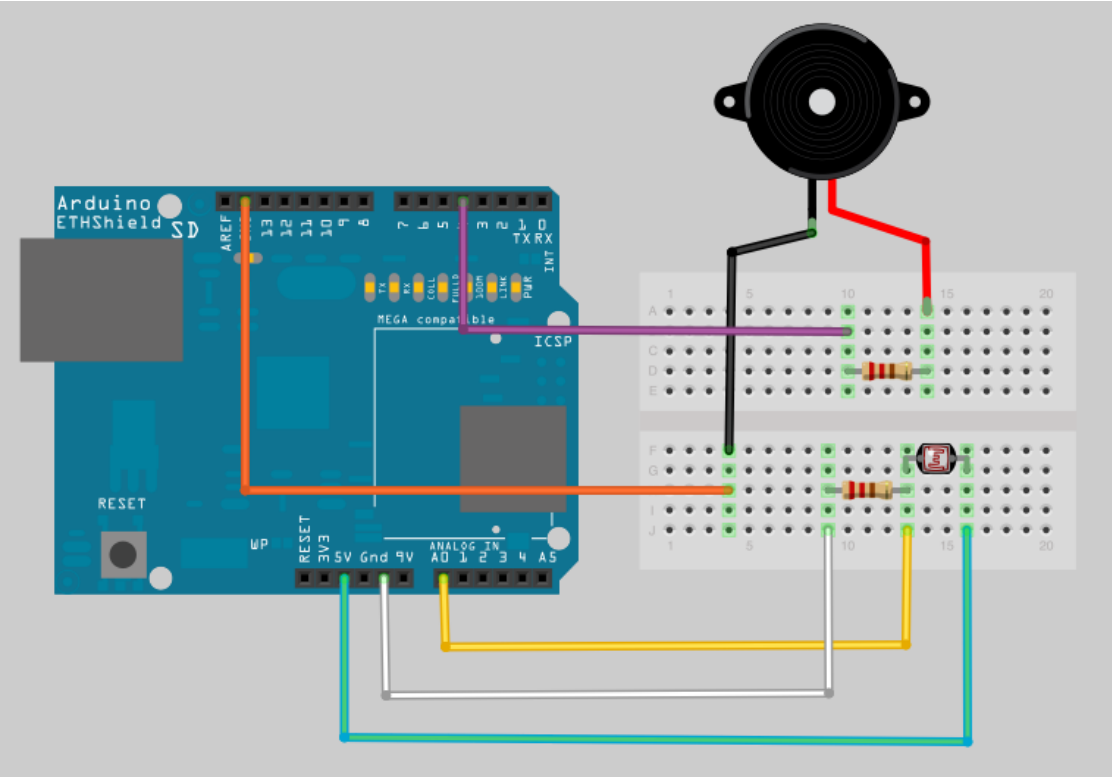
Analog Inputs

- Note that reading an analog input is (relatively) slow, compared to reading digital pins. This is because the Arduino measures voltages using an internal reference voltage generator (DAC, Digital-to-Analog Converter), and then comparing the input voltage to the reference voltage, then changing the reference voltage, comparing again, changing the reference voltage, comparing ... until the two voltages are equal.
- The resolution of the internal DAC is 10 bits.
- This means that the maximum number it can read is 10 bits long, or 210 = 1024, so a number from 0 (B0000000000) to 1023 (B1111111111).
- 1023 means the input is 5v (or 3v3) and 0 means 0v.

Inputs

- Digital inputs will come to the Arduino as either on or off (HIGH or LOW, respectively).
  - HIGH is 5VDC.
  - LOW is 0VDC.
- Analog inputs will come to the Arduino as a range of numbers, based upon the electrical characteristics of the circuit.
  - 0 to 1023
  - .0049 V per digit (4.9 mV)
  - Read time is 100 microseconds (10,000 a second)

Part 1: Analog IO



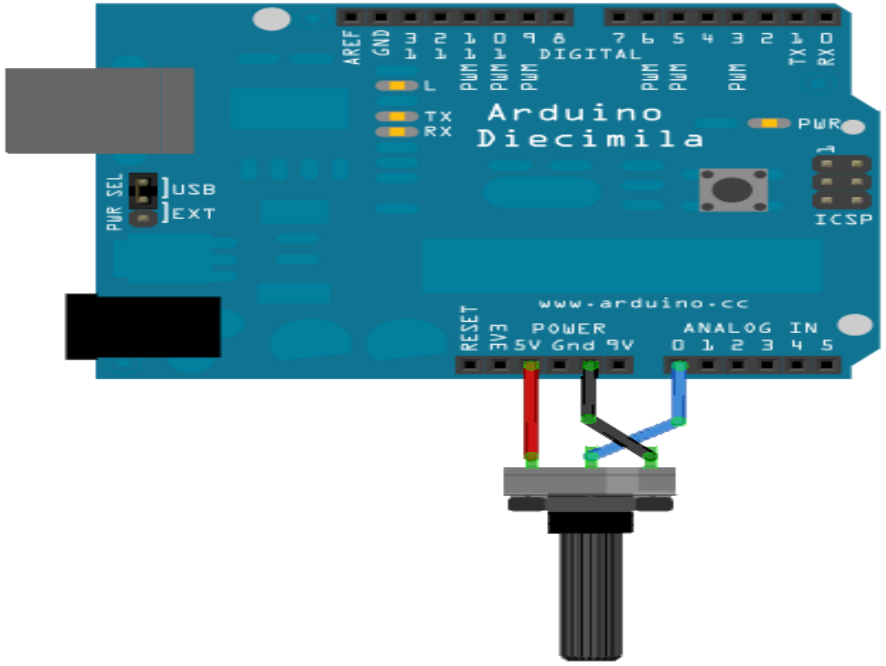
[A light theremin]

Analog Input-Application

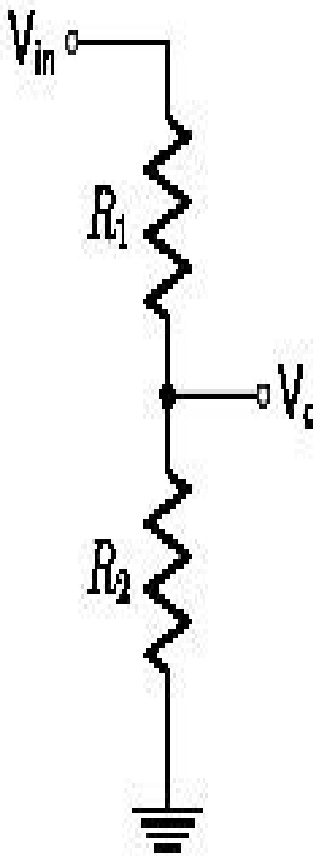
- The variable resistor can be replaced with a sensor.
- For example, a photo resistor.
  - Depending upon the light level at the photo resistor:
    - Turn on a light
    - Increase or decrease the brightness of an LED (or an LED array)
- Most sensors are simply variable resistors, but vary their resistance based on some physical characteristic.

Analog Input

- A potentiometer (variable resistor) is connected to analog pin 0 to an Arduino.
- Values presented to pin 0 will vary depending upon the resistance of the potentiometer.

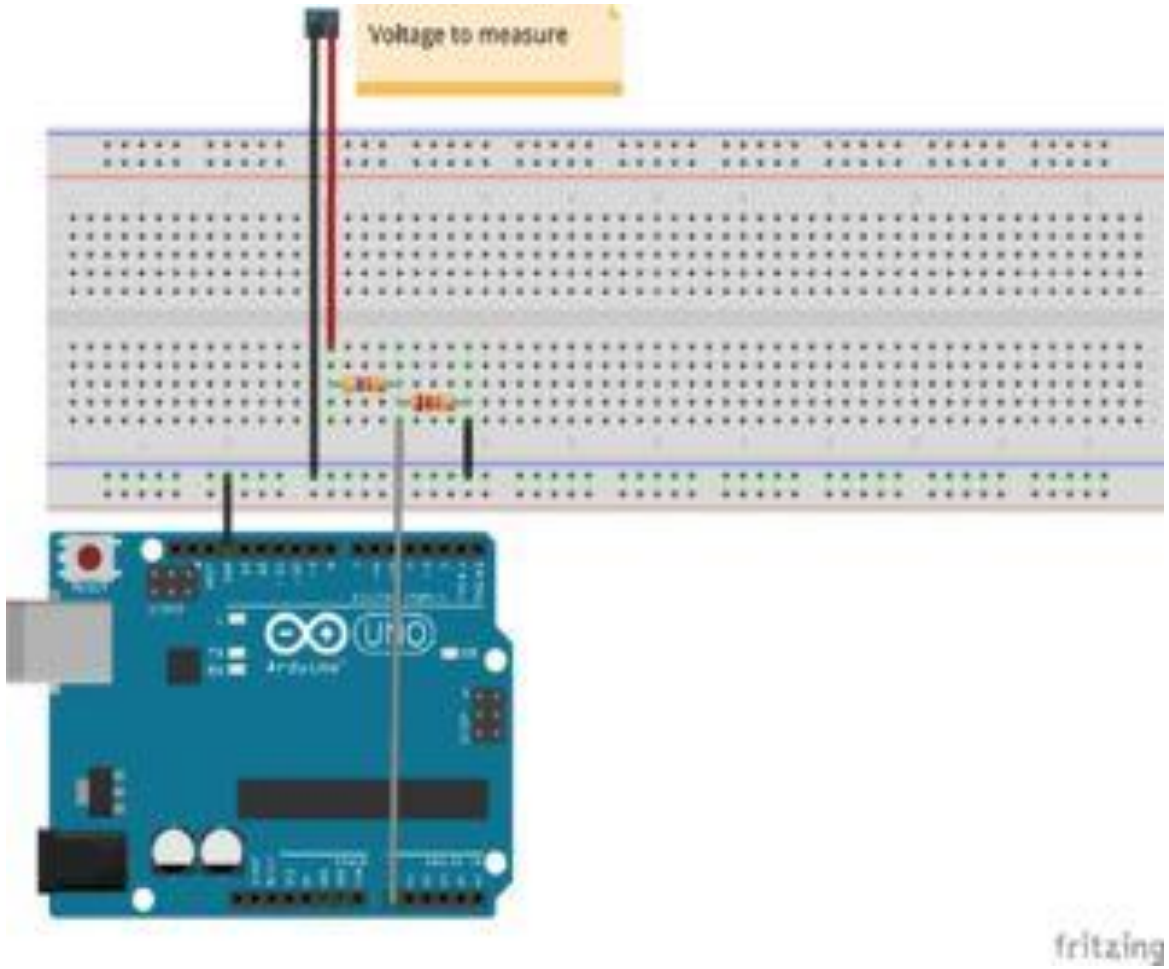


Analog inputs and outputs.



$$V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{\text{in}}$$

$$\Leftrightarrow \quad V_{\text{in}} = \frac{R_1 + R_2}{R_2} \cdot V_{out}$$



- ```
  void setup() {
    // open a serial connection
    Serial.begin(9600);
  }

  void loop() {
    // read the input on A0 at default resolution (10 bits)
    // and send it out the serial connection
    analogReadResolution(10);
    Serial.print("ADC 10-bit (default) : ");
    Serial.print(analogRead(A0));

    // change the resolution to 12 bits and read A0
    analogReadResolution(12);
    Serial.print(", 12-bit : ");
    Serial.print(analogRead(A0));

    // change the resolution to 16 bits and read A0
    analogReadResolution(16);
    Serial.print(", 16-bit : ");
    Serial.print(analogRead(A0));

    // change the resolution to 8 bits and read A0
    analogReadResolution(8);
    Serial.print(", 8-bit : ");
    Serial.println(analogRead(A0));

    // a little delay to not hog serial monitor
    delay(100);
  }
  ```

THE C PROGRAMMING LANGUAGE
  ```

Analog Read serial Out

```
void setup() {     // initialize serial communication at 9600 bits per second:
   Serial.begin(9600);
}
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

TASK

THE C PROGRAMMING LANGUAGE

Arduino-Analog Output-LED

Arduino-Analog Output-LED

```
//pin 11,10,9,6,5,3 can be used for Analog output
void setup(){
pinMode(11, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
analogWrite(11, 255);   // sending Analog output 255
delay(500);            // Wait for half a second
analogWrite(11, 200);   // Sending Analog output 200
delay(500);            // Wait for half a second
analogWrite(11, 150);   // Sending Analog output 150
delay(500);            // Wait for half a second
analogWrite(11, 100);   // Sending Analog output 100
delay(500);            // Wait for half a second
analogWrite(11, 50);   // Sending Analog output 50
delay(500);            // Wait for half a second
analogWrite(11, 0);     // sending analog output 0
delay(500);            // Wait for half a second
}
```

THE C PROGRAMMING LANGUAGE

Arduino-Analog Output-LED

```
//pin 11,10,9,6,5,3 can be used for Analog output
void setup(){
pinMode(11, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
analogWrite(11, 255);   // sending Analog output 255
delay(500);             // Wait for half a second
analogWrite(11, 200);   // Sending Analog output 200
delay(500);             // Wait for half a second
analogWrite(11, 150);   // Sending Analog output 150
delay(500);             // Wait for half a second
analogWrite(11, 100);   // Sending Analog output 100
delay(500);             // Wait for half a second
analogWrite(11, 50);   // Sending Analog output 50
delay(500);             // Wait for half a second
analogWrite(11, 0);     // sending analog output 0
delay(500);             // Wait for half a second
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-Analog Output-LED_Dimming Using Loop Structure

```
//pin 11,10,9,6,5,3 can be used for Analog output
void setup(){
  pinMode(11, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=255; i>0; i--){
    analogWrite(11, i);   // sending Analog output 255
    delay(20);
  }
  for(int i=0; i<255; i++){
    analogWrite(11, i);   // sending Analog output 255
    delay(20);
  }
}
```

Arduino-Analog Output-LED_Dimming Using Loop Structure

THE C PROGRAMMING LANGUAGE

```
//pin 11,10,9,6,5,3 can be used for Analog output
void setup(){
  pinMode(11, OUTPUT); // Specify Arduino Pin number and output/input mode
}
void loop(){
  for(int i=255; i>0; i--){
    analogWrite(11, i);   // sending Analog output 255
    delay(20);
  }
  for(int i=0; i<255; i++){
    analogWrite(11, i);   // sending Analog output 255
    delay(20);
  }
}
```

Reading analog inputs and scaling

```
const int potPin = 0;
// select the input pin for the potentiometer
void loop() {
int val;
// The value coming from the sensor
int percent; // The mapped value
val = analogRead(potPin);
// read the voltage on the pot (val ranges from 0 to 1023)
percent = map(val,0,1023,0,100);
 // percent will range from 0 to 100.
```

TASK

- Create a program that resets the delayTime to 2000 once it has reached 0

TASK

- Using 3 LED's (red, yellow and green) build a traffic light that
  - Illuminates the green LED for 5 seconds
  - Illuminates the yellow LED for 2 seconds
  - Illuminates the red LED for 5 seconds
  - repeats the sequence

- Note that after each illumination period the LED is turned off!

TASK 2

- Modify Task 1 to have an advanced green (blinking green LED) for 3 seconds before illuminating the green LED for 5 seconds

Task

- Replace all delay times with variables
- Replace LED pin numbers with variables

TASK

- Create a program that illuminates the green LED if the counter is less than 100, illuminates the yellow LED if the counter is between 101 and 200 and illuminates the red LED if the counter is greater than 200

Task

- Using input and output commands find the ASCII values of

#	ASCII	#	ASCII	@	ASCII	@	ASCII	@	ASCII	@	ASCII
1	49	6	54	a	97	h	104	o	111	v	118
2	50	7	55	b	98	i	105	p	112	w	119
3	51	8	56	c	99	j	106	q	113	x	120
4	52	9	57	d	100	k	107	r	114	y	121
5	53			e	101	l	108	s	115	z	122
				f	102	m	109	t	116		
				g	103	n	110	u	117		

Task

- Create a program so that when the user enters 1 the green light is illuminated, 2 the yellow light is illuminated and 3 the red light is illuminated

Create a program so that when the user enters ‘b’ the green light blinks, ‘g’ the green light is illuminated ‘y’ the yellow light is illuminated and ‘r’ the red light is illuminated

Task

- Light Mode Video
- Create a program that
  - when the user enters ‘1’ all lights turn on
  - when the user enters ‘2’ all lights flash
  - when the user enters ‘3’ lights cycle repeatedly
  - when the user enters ‘q’ or ‘e’ the lights turn off
  - when + or - is pressed the speed of the LED increase or decrease

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

BOOLEAN VARIABLES

THE C PROGRAMMING LANGUAGE

```
boolean done = true;
```

TASK

- Write a program that asks the user for a number and outputs the number that is entered. Once the number has been output the program finishes.
  - EXAMPLE:

```
Please enter a number: 1 <enter>
The number you entered was: 1
```

TASK

- Write a program that asks the user for a number and outputs the number squared that is entered. Once the number has been output the program finishes.

```
Please enter a number: 4 <enter>
Your number squared is: 16
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Christmas Light Assignment

- Using at least 5 LEDs create a program that has at least 4 different modes
- Must use if statements and user input to switch modes
- Marks:
  - Creativity [4 Marks]
  - Working user input [2 Marks]
  - Four Working modes [2 Marks]
  - Instructions [1 Mark]

Creating a bar graph using LEDs

```
const int NoLEDs = 8;
const int ledPins[] = { 70, 71, 72, 73, 74, 75, 76, 77};
const int analogInPin = 0; // Analog input pin const int wait = 30;
const boolean LED_ON = HIGH;
const boolean LED_OFF = LOW;
int sensorValue = 0; // value read from the sensor
int ledLevel = 0; // sensor value converted into LED 'bars'
void setup() {
for (int i = 0; i < NoLEDs; i++)
{
pinMode(ledPins[i], OUTPUT); // make all the LED pins outputs
}
}
```

Creating a bar graph using LEDs

```
void loop() {
sensorValue = analogRead(analogInPin); // read the analog in value
ledLevel = map(sensorValue, 0, 1023, 0, NoLEDs); // map to the number of
  LEDs
for (int i = 0; i < NoLEDs; i++)
{
if (i < ledLevel ) {
digitalWrite(ledPins[i], LED_ON); // turn on pins less than the level
}
else {
digitalWrite(ledPins[i], LED_OFF); // turn off pins higher than the level:
}
}
}
```

Measuring Temperature

```
const int inPin = 0; // analog pin
void loop()
{
int value = analogRead(inPin);
float millivolts = (value / 1024.0) * 3300;
 //3.3V analog input
float celsius = millivolts / 10; // sensor
 output is 10mV per degree Celsius
delay(1000); // wait for one second
}
```

Reading data from Arduino

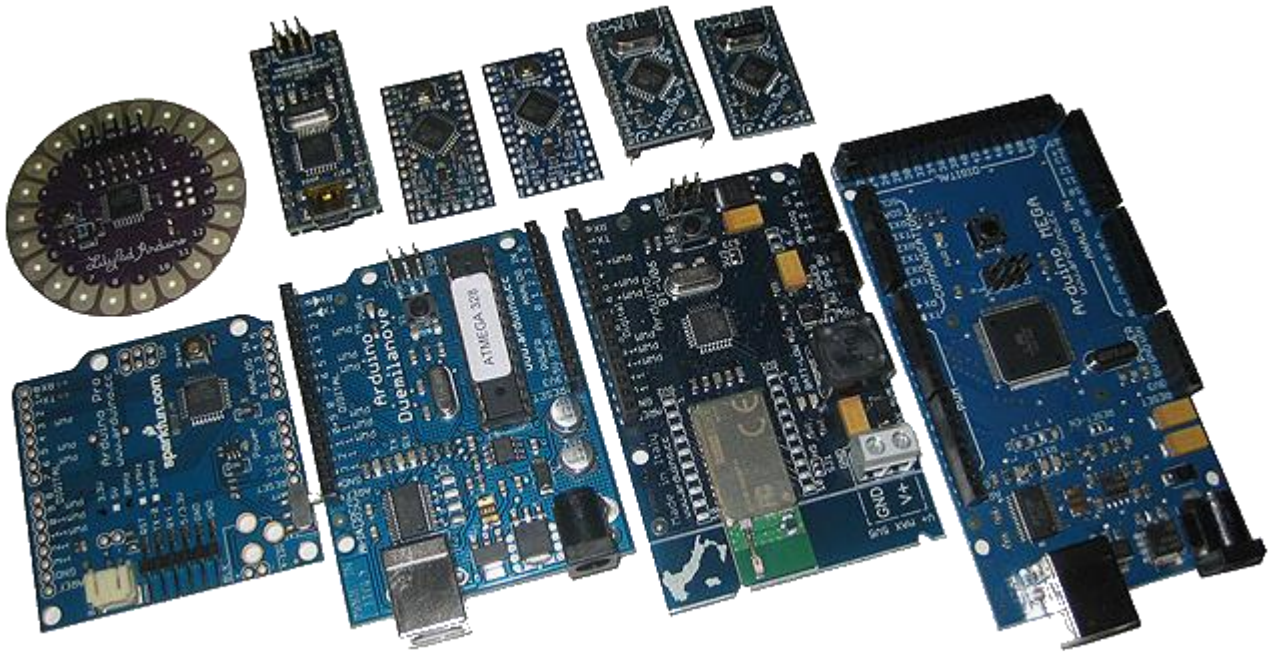
```
void setup()
{
Serial.begin(9600);
}
void serialtest()
{
int i;
for(i=0; i<10; i++)
Serial.println(i);
}
```

THE C PROGRAMMING LANGUAGE

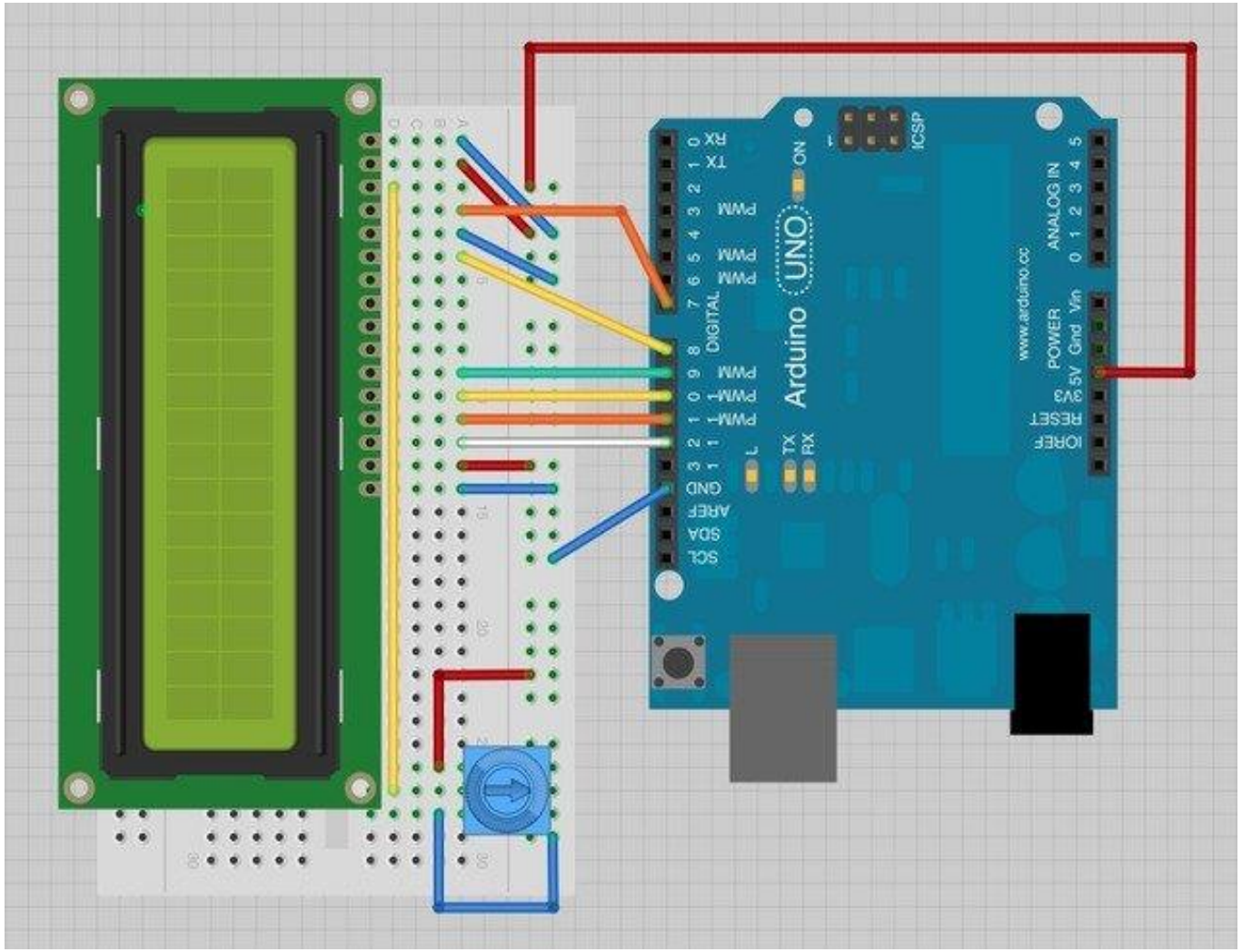
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

مرکز آموزش علمی-کاربردی گروه انتخاب

Lecture 4:
Introduction To LCD



اتصال LCD کاراکتری به آردوینو:



THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

THE C PROGRAMMING LANGUAGE

نام پین مربوط به LCD	شماره پایه مربوط به L C D	شماره پایه متصل شده به پین آردوینو
RS	4	7
E	6	8
D4	11	9
D5	12	10
D6	13	11

پایه شماره 5 LCDباید به زمین وصل شود

Connecting LCDs

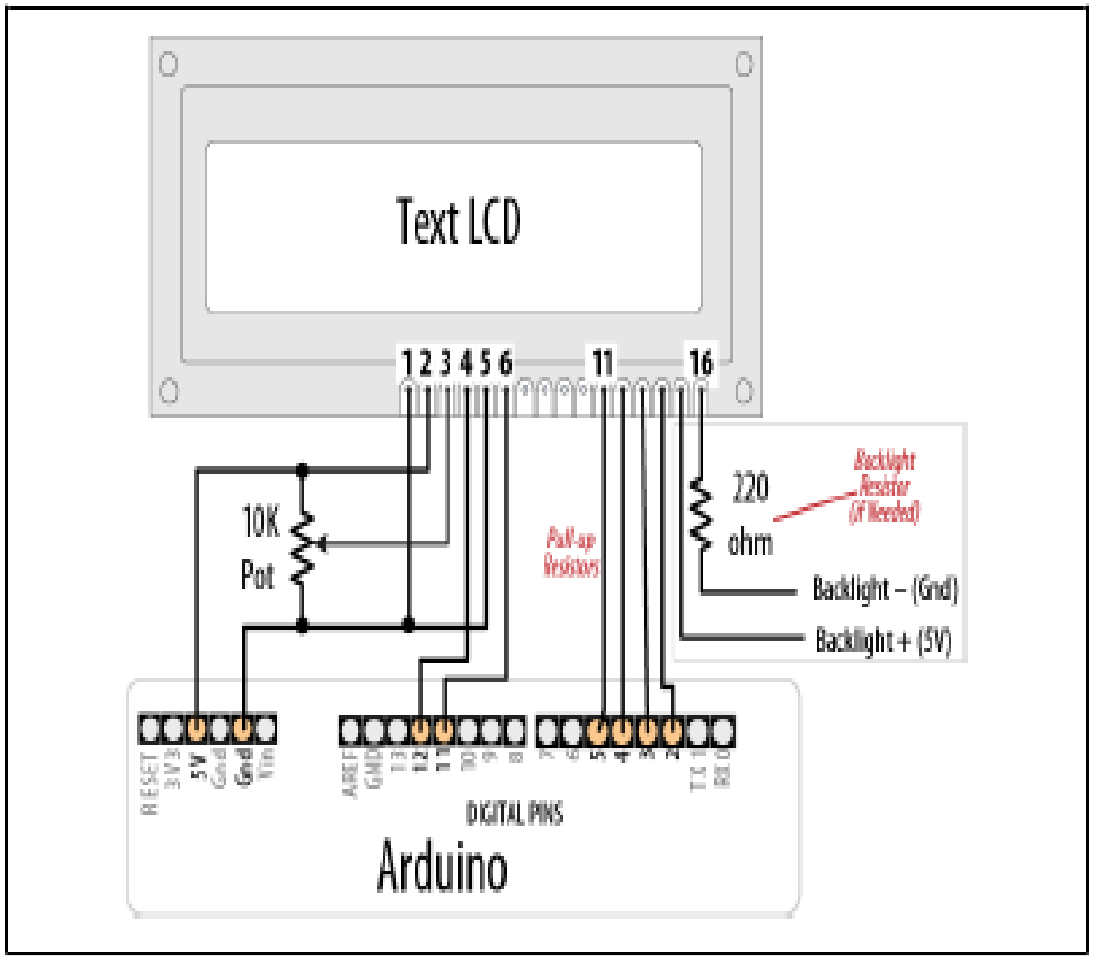


Table 11-1. LCD pin connections

LCD pin	Function	Arduino pin
1	Gnd or 0V or Vss	Gnd
2	+5V or Vdd	5V
3	Vo or contrast	
4	RS	12
5	R/W	
6	E	11
7	D0	
8	D1	
9	D2	
10	D3	
11	D4	5
12	D5	4
13	D6	3
14	D7	2
15	A or analog	
16	K or cathode	

THE
IMING
AGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Using LCDs

```
#include <LiquidCrystal.h> // include the library code
//constants for the number of rows and columns in the LCD
const int numRows = 2;
const int numCols = 16;
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
lcd.begin(numCols, numRows);
lcd.print("hello, world!"); // Print a message to the LCD.
}
```

```
include <liquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//LiquidCrystal lcd(7 , 8 , 9 , 10, 11 ,12);
lcd.begin(16, 2);
lcd.print("hello, world!");
lcd.setCursor(0, 1);
```

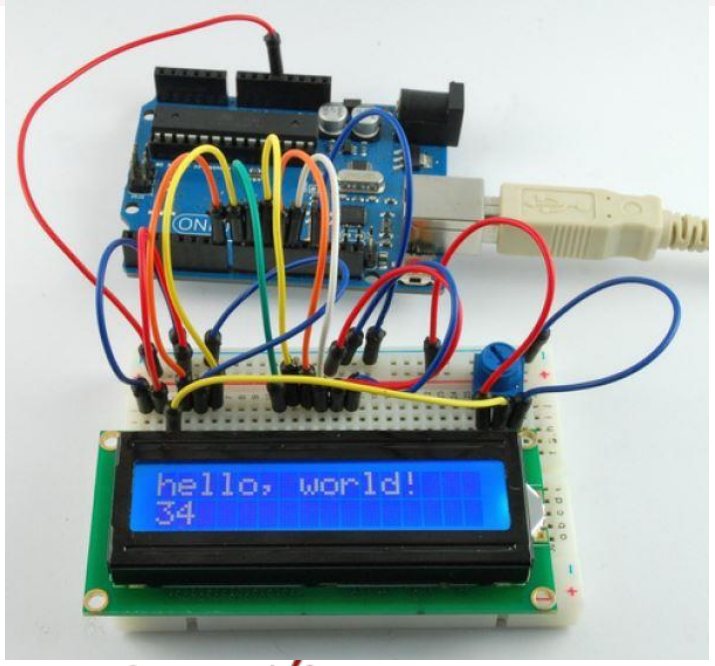
خط برای اضافه کردن کتابخانه مربوط به lcd کاراکتری هست.

کاراکتری lcd از این خط برای تعریف پایه هایی از آردوینو که به
متصل میشوند استفاده میشود. طبق این خط پایه های 12 و 11 و 5 و 4
کاراکتری وصل میشود lcd و 3 و 2 از آردوینو به
تذکر: ما شماره این پایه ها رو دوست نداریم و میخوایم خودمون
:عوضش کنیم به خاطر همین از شماره دستور زیر استفاده میکنیم

د پایه های 7 و 8 و 9 و 10 و 11 و 12 به lcd کاراکتری وصل میشود

ه که LCD که ما به آردوینو وصل کردیم یه 2*16 LCD هست.

با استفاده از این دستور عبارت hello , world روی LCD کاراکتری نمایش داده میشه.
طبیعی هست که اگه بخوایم متن دیگه ای رو نمایش بده باید به جای hello ,world متن مورد
نظر خودمون رو بنویسیم.
LCD میره به ردیف اول و سطر دوم و شروع میکنه به نوشتن داده ها.



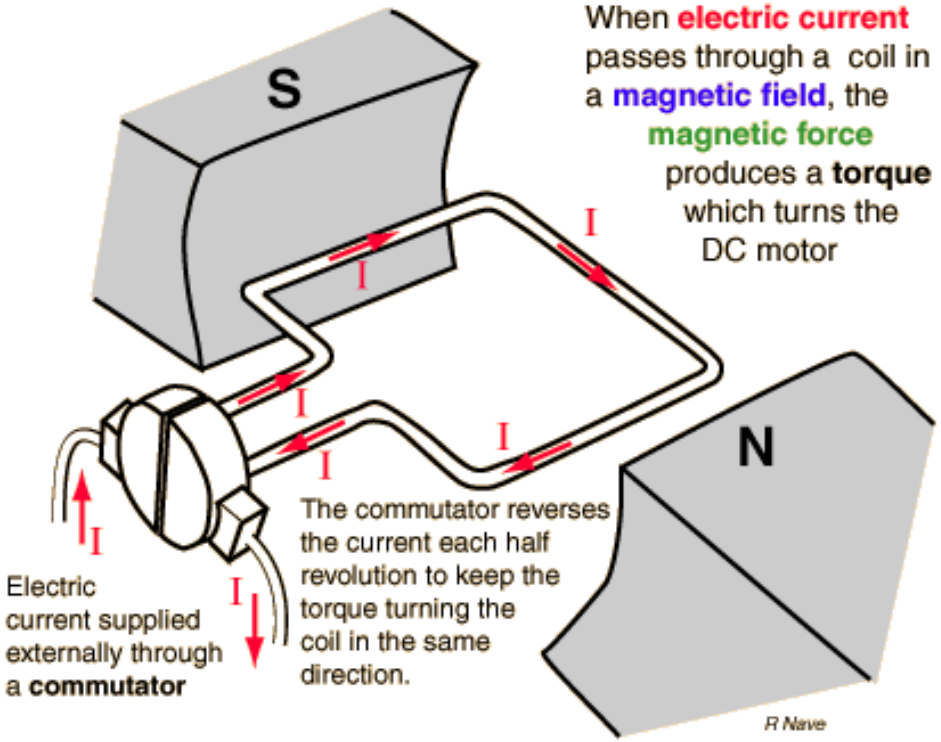
Actuators

- Device that turns energy (typically electrical) to motion
- Features
  - Force
  - Speed
  - Torque
  - Power
  - Efficiency

THE C PROGRAMMING LANGUAGE

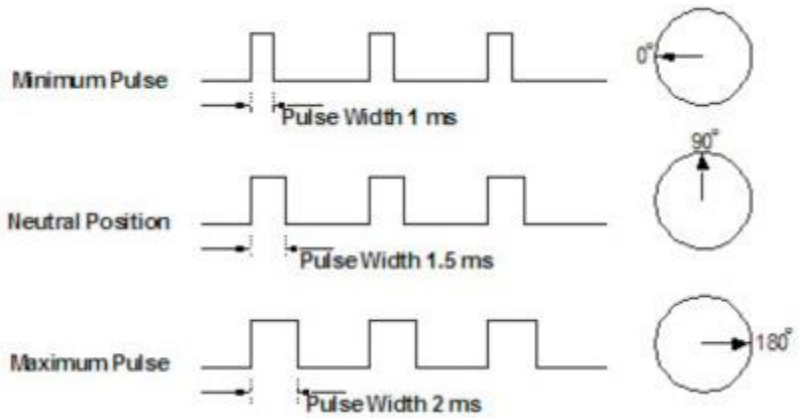
DC motor

- Force is produced (F=ILB) due to the electric current in a wire inside a magnetic field.
- Proportional to the current, therefore can be controlled by potentiometer
- Hard to control precisely



Servo motors

- A DC motor with a control circuit
- Servo motors are controlled by PWM through the control pin

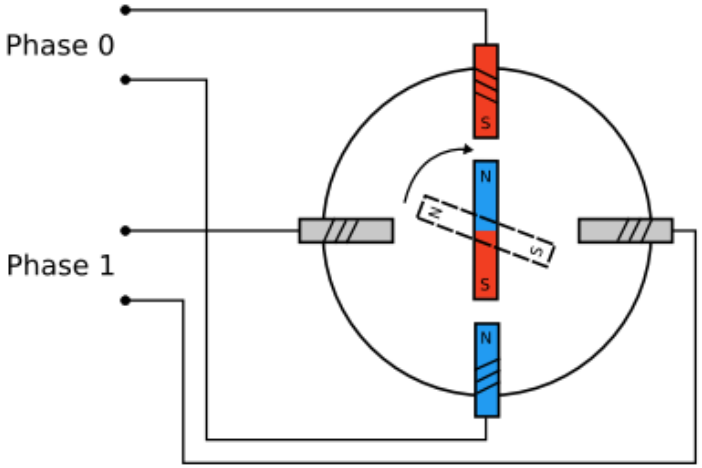


Servo motor control

```
#include <Servo.h>
   Servo myservo;  // create servo object to control a servo
   void setup() {
   int val = 180;    // variable to control servo
     myservo.attach(9);  // pin 9 is a PWM pin
   }
   void loop() {
     myservo.write(val);                    // constant servo speed
     delay(15);                     // waits for the servo to get there
   }
```

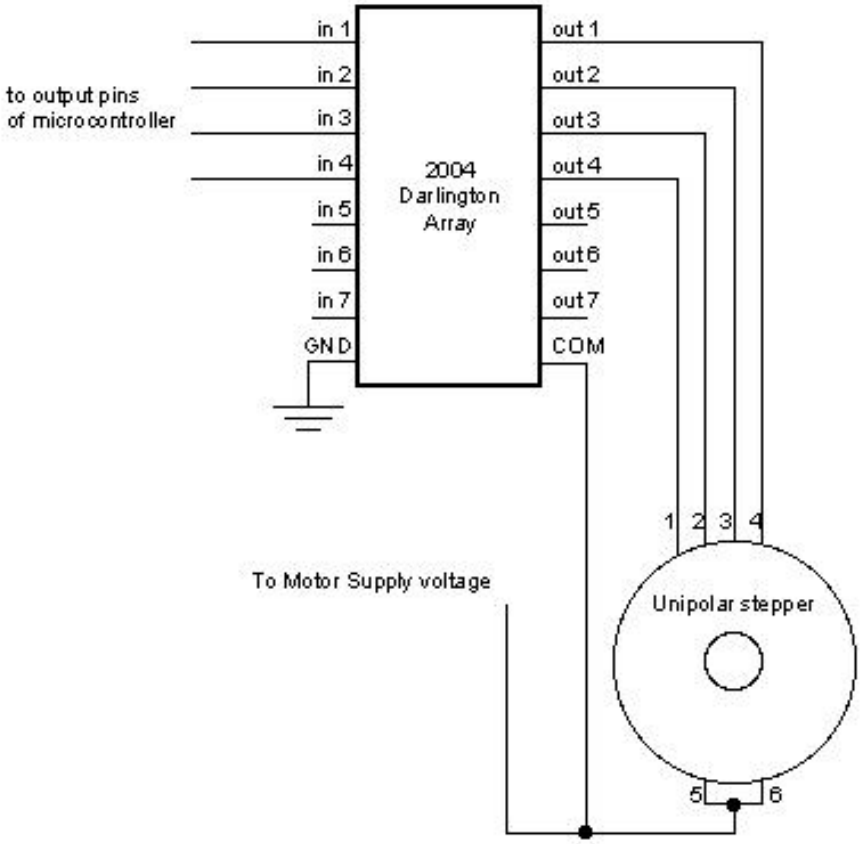
Stepper Motors

- motor controlled by a series of electromagnetic coils.
- The center shaft has a series of magnets mounted on it
- the coils are alternately given current or not, creating magnetic fields which repulse or attract the magnets on the shaft, causing the motor to rotate.
- allows for very precise control of the motor. it can be turned in very accurate steps of set degree increments
- two basic types
  - unipolar
  - bipolar



Example: Unipolar stepper motor

- Four digital pins required to control the motor
- Darlington transistor array used for supplying the power required



Stepper motor control

```
#include <Stepper.h> //the control sequence is in this library

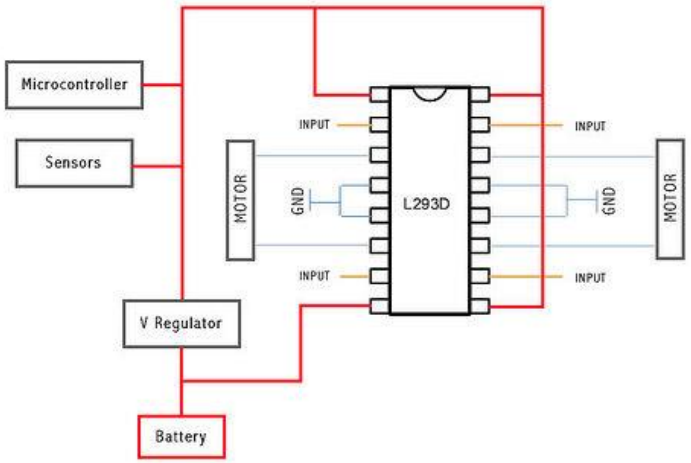
  const int stepsPerRevolution = 200;  // motor-dependent

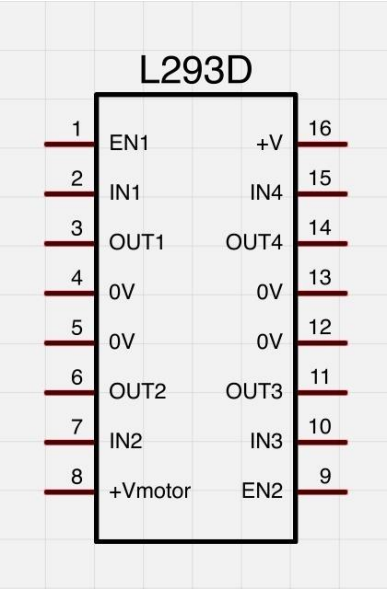
  Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11); //pins used

  void setup() {
    // set the speed at 60 rpm:
    myStepper.setSpeed(60);  //actually sets the delay between steps
  }

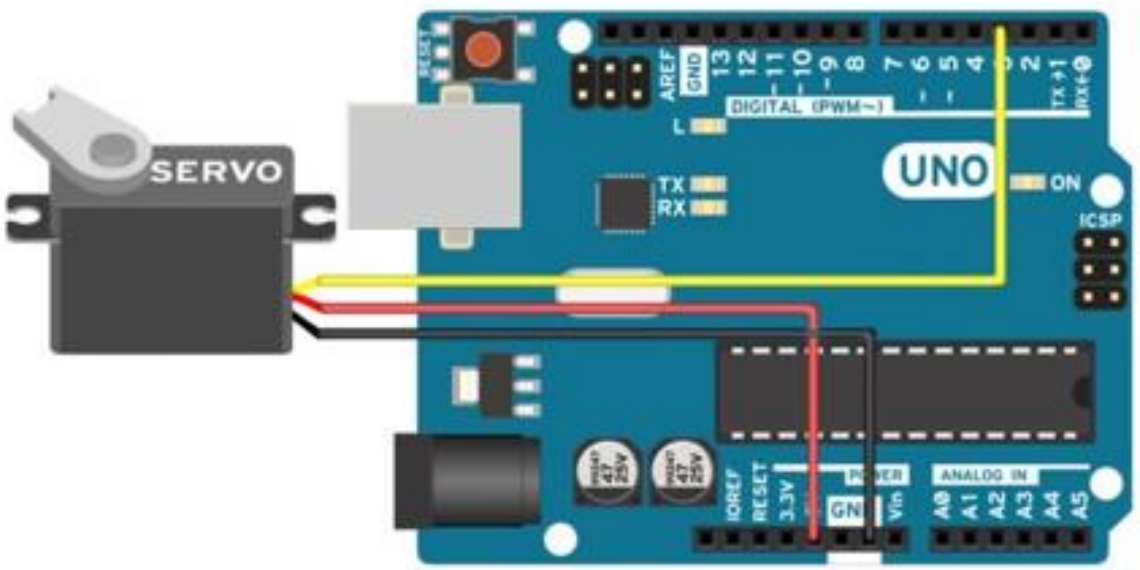
  void loop() {
    // step one revolution  in one direction:
    myStepper.step(stepsPerRevolution);
    delay(500);

}
```


Microcontroller
Sensors
V Regulator
Battery
MOTOR
GND
INPUT
L293D
INPUT
GND
MOTOR
INPUT
INPUT


L293D
1 EN1
2 IN1
3 OUT1
4 0V
5 0V
6 OUT2
7 IN2
8 +Vmotor
+V 16
IN4 15
OUT4 14
0V 13
0V 12
OUT3 11
IN3 10
EN2 9

سروو موتور



موتور سروو سه تا سیم مادگی داره که باید به آردوینو وصل کنیم.سیم اول VCC هست که باید وصل بشه به 5 ولت.(سیم قرمز رنگ در شکل بالا) سیم دوم سیم زمین موتور سروو هست که باید به پایه GND آردوینو وصل بشه(سیم مشکی رنگ) و در نهایت سیم سوم دیتا هست هم که باید وصل بشه به یکی از پایه های دیجیتال آردوینو که ما اینجا از پایه شماره 3 آردوینو استفاده کردیم(سیم زرد رنگ در شکل بالا)

کد راه اندازی موتور سروو:

```
include <servo.h>

 int servoPin = 3;

 Servo Servo1;
void setup() {

   Servo1.attach(servoPin);
}
void loop(){

   Servo1.write(0);
  delay(1000);

   Servo1.write(90);
  delay(1000);

   Servo1.write(180);
  delay(1000);
}
```

در ابتدای کار کتابخونه موتور سروو به برنامهاضافه میشود
.در خط بعد پایه دیجیتالی که موتور بهش وصل میشود رو به صورت یه متغیر integer تعریف میکنیم.
در خط بعدی یه objectبه نام Servo1تعریف میکنیم.
در تابع Setupهم تنها کاری که نیاز هست انجام بدیم اینه که موتور را به پایه 3 دیجیتال attachکنیم.
در نهایت هم در تابع loopبا استفاده از دستورات Servo1.write () وآرگومان ورودیش به موتور میگیم که چند درجه بچرخه.
در اینجا ما ابتدا به موتور گفتیم توی زاویه 0 درجه وایسا بعد یه ثانیه صبر کن و بعد از اون برو توی زاویه 90 درجه بچرخ ، دوباره یه ثانیه صبر میکنه و در نهایت هم بهش میگیم برو توی زاویه 180 درجه.دوباره یه ثانیه صبر میکنه و برمیگرده اول تابع loopو در نهایت میاد برمیگرده به حالت اولش و توی زاویه صفر قرار میگیره.

Servo

Example 2:

```
#include <Servo.h> // Include the Servo library
int servoPin = 3; // Declare the Servo pin
Servo Servo1; // Create a servo object
void setup() {
  Servo1.attach(servoPin);   // We need to attach the servo to the used pin number
}
void loop(){
  Servo1.write(0);   // Make servo go to 0 degrees
  delay(1000);
  Servo1.write(90);   // Make servo go to 90 degrees
  delay(1000);   // Make servo go to 180 degrees
  Servo1.write(180);
  delay(1000);
}
```

Controlling a servo position using a potentiometer (variable resistor)

Controlling a servo position using a potentiometer (variable resistor)

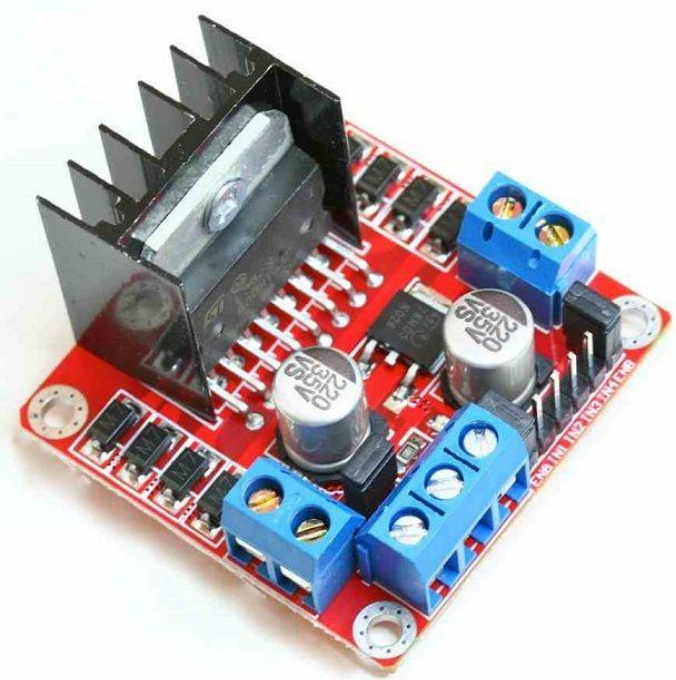
```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin
void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void 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
}
```

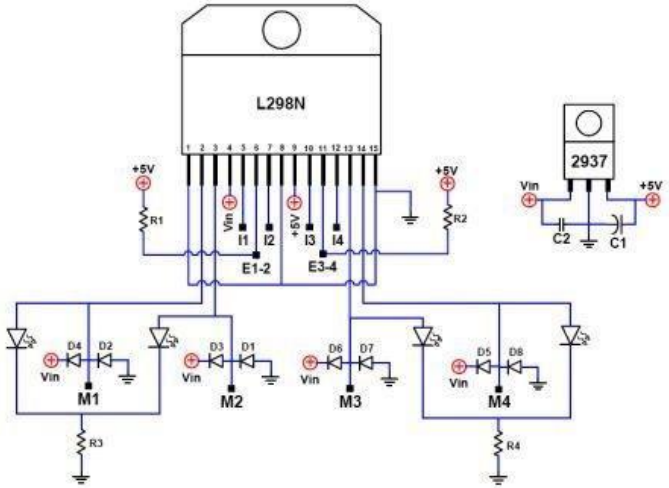
- // ----------

- / Use this code to test your motor with the Arduino board:

```
// if you need PWM, just use the PWM outputs on the Arduino
// and instead of digitalWrite, you should use the analogWrite command
// ------------------------------------------------------------------------------ Motors
int motor_left[] = {2, 3};
int motor_right[] = {7, 8};
// ------------------------------------------------------------------------------ Setup
void setup() {
Serial.begin(9600);
// Setup motors
int i;
for(i = 0; i < 2; i++){
pinMode(motor_left[i], OUTPUT);
pinMode(motor_right[i], OUTPUT);
}  }
// ------------------------------------------------------------------------------ Loop
void loop() {
drive_forward();
delay(1000);
motor_stop();
Serial.println("1");
drive_backward();
delay(1000);
motor_stop();
Serial.println("2");
turn_left();
delay(1000);
motor_stop();
Serial.println("3");
turn_right();
delay(1000);
motor_stop();
Serial.println("4");
motor_stop();
delay(1000);
motor_stop();
Serial.println("5");
}
// ------------------------------------------------------------------------------ Drive
void motor_stop(){
digitalWrite(motor_left[0], LOW);
digitalWrite(motor_left[1], LOW);
digitalWrite(motor_right[0], LOW);
digitalWrite(motor_right[1], LOW);
delay(25);
}
void drive_forward(){
digitalWrite(motor_left[0], HIGH);
digitalWrite(motor_left[1], LOW);
digitalWrite(motor_right[0], HIGH);
digitalWrite(motor_right[1], LOW);
}
void drive_backward(){
digitalWrite(motor_left[0], LOW);
digitalWrite(motor_left[1], HIGH);
digitalWrite(motor_right[0], LOW);
digitalWrite(motor_right[1], HIGH);
}
void turn_left(){
digitalWrite(motor_left[0], LOW);
digitalWrite(motor_left[1], HIGH);
digitalWrite(motor_right[0], HIGH);
digitalWrite(motor_right[1], LOW);
}
void turn_right(){
digitalWrite(motor_left[0], HIGH);
digitalWrite(motor_left[1], LOW);
digitalWrite(motor_right[0], LOW);
digitalWrite(motor_right[1], HIGH);
```

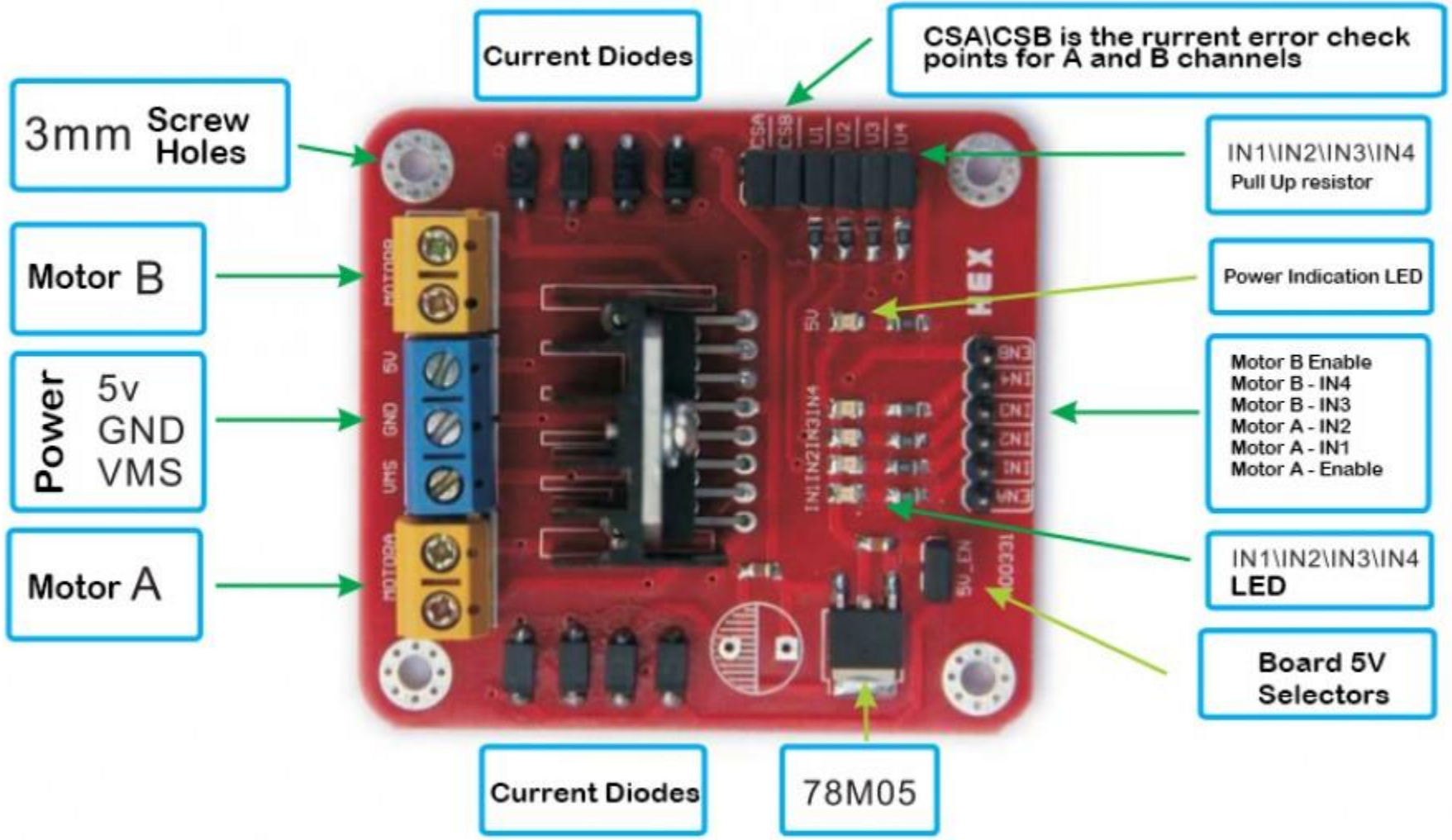
THE C PROGRAMMING LANGUAGE
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY





The L298N driver module, using ST' L298N chip, can directly drive two 3-30V DC motor, and provide a 5V output interface, power for 5V single-chip circuitry , support 3.3VMCU control. you can easily control the DC motor speed and direction, also control the 2-phase stepper motor. It is essential for making a smart car.

- Driver: L298N Dual H Bridge DC Motor Driver IC
- Driven part of the terminal supply area Vs: +5 V ~ +35 V; such as the need to take power within the board, the supply area Vs: +7 V ~ +35 V
  Driven part of the peak current Io: 2A
  The logical part of the terminal supply area Vss: +5 V ~ +7 V (can take power within the board +5 V)
  The logical part of the operating current range: 0 ~ 36mA
  Control signal input voltage range:
  Low:-0.3V ≤ Vin ≤ 1.5V
  High: 2.3V ≤ Vin ≤ Vss
  Enable signal input voltage range:
  Low: -0.3 ≤ Vin ≤ 1.5V (control signal is invalid)
  High: 2.3V ≤ Vin ≤ Vss (control signal active)
  Maximum power consumption: 20W (when the temperature T = 75 ℃)
  Storage temperature: -25 ℃ ~ +130 ℃
  Other Extensions: control of direction indicators, the logic part of the plate to take power interface.
  Driver Board Size: 55mm * 60mm * 30mm
-


Current Diodes
CSA\CSB is the rurrent error check points for A and B channels
3mm Screw Holes
IN1\IN2\IN3\IN4 Pull Up resistor
Motor B
Power Indication LED
Power 5v GND VMS
Motor B Enable
Motor B - IN4
Motor B - IN3
Motor A - IN2
Motor A - IN1
Motor A - Enable
Motor A
IN1\IN2\IN3\IN4 LED
Board 5V Selectors
Current Diodes
78M05

```
/ Use this code to test your motor with the Arduino board:
// if you need PWM, just use the PWM outputs on the Arduino
// and instead of digitalWrite, you should use the analogWrite command
// ------------------------------------------------------------------------------ Motors
int motor_left[] = {2, 3};
int motor_right[] = {7, 8};
// ------------------------------------------------------------------------------ Setup
void setup() {
Serial.begin(9600);
// Setup motors
int i;
for(i = 0; i < 2; i++){
pinMode(motor_left[i], OUTPUT);
pinMode(motor_right[i], OUTPUT);
}  }
// ------------------------------------------------------------------------------ Loop
```

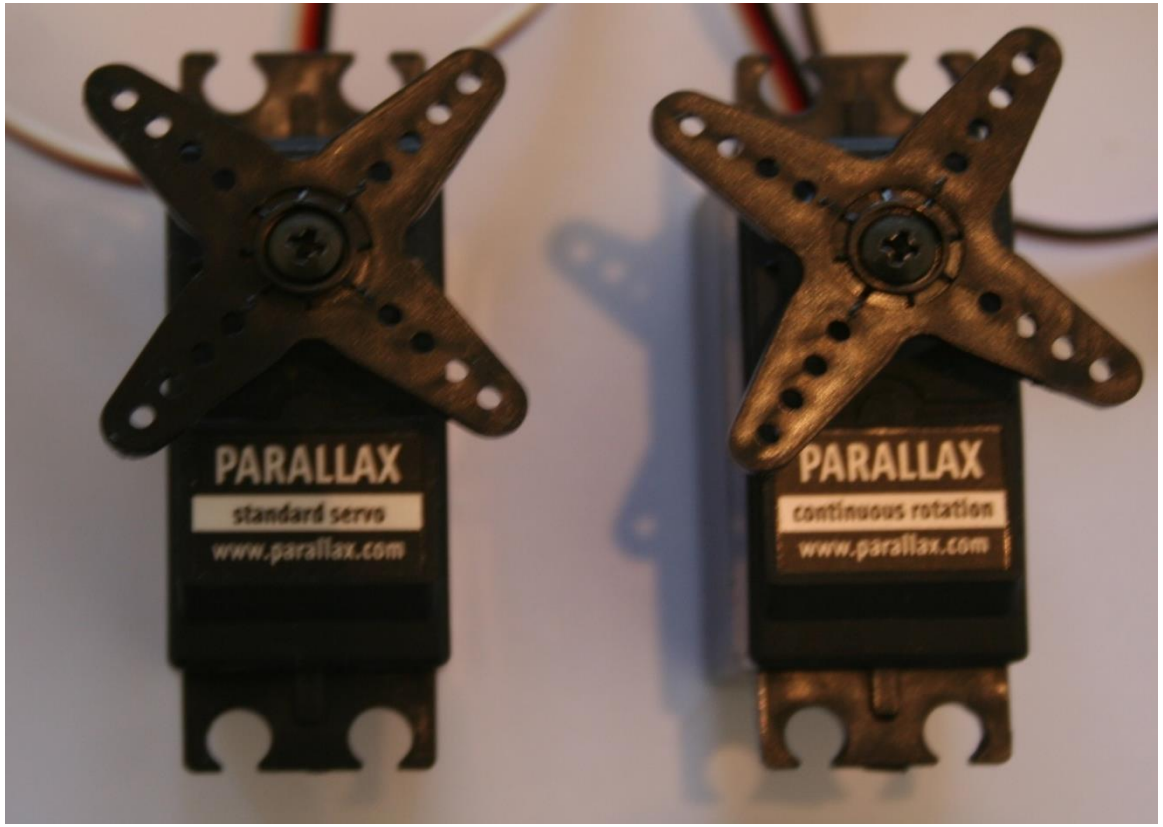
```
void loop() {
drive_forward();
delay(1000);
motor_stop();
Serial.println("1");
drive_backward();
delay(1000);
motor_stop();
Serial.println("2");
turn_left();
delay(1000);
motor_stop();
Serial.println("3");
turn_right();
delay(1000);
motor_stop();
Serial.println("4");
motor_stop();
delay(1000);
motor_stop();
Serial.println("5");
}
```

```
void motor_stop(){
digitalWrite(motor_left[0], LOW);
digitalWrite(motor_left[1], LOW);
digitalWrite(motor_right[0], LOW);
digitalWrite(motor_right[1], LOW);
delay(25);
}
void drive_forward(){
digitalWrite(motor_left[0], HIGH);
digitalWrite(motor_left[1], LOW);
digitalWrite(motor_right[0], HIGH);
digitalWrite(motor_right[1], LOW);
}
void drive_backward(){
digitalWrite(motor_left[0], LOW);
digitalWrite(motor_left[1], HIGH);
digitalWrite(motor_right[0], LOW);
digitalWrite(motor_right[1], HIGH);
}
void turn_left(){
digitalWrite(motor_left[0], LOW);
digitalWrite(motor_left[1], HIGH);
digitalWrite(motor_right[0], HIGH);
digitalWrite(motor_right[1], LOW);
}
void turn_right(){
digitalWrite(motor_left[0], HIGH);
digitalWrite(motor_left[1], LOW);
digitalWrite(motor_right[0], LOW);
digitalWrite(motor_right[1], HIGH);
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

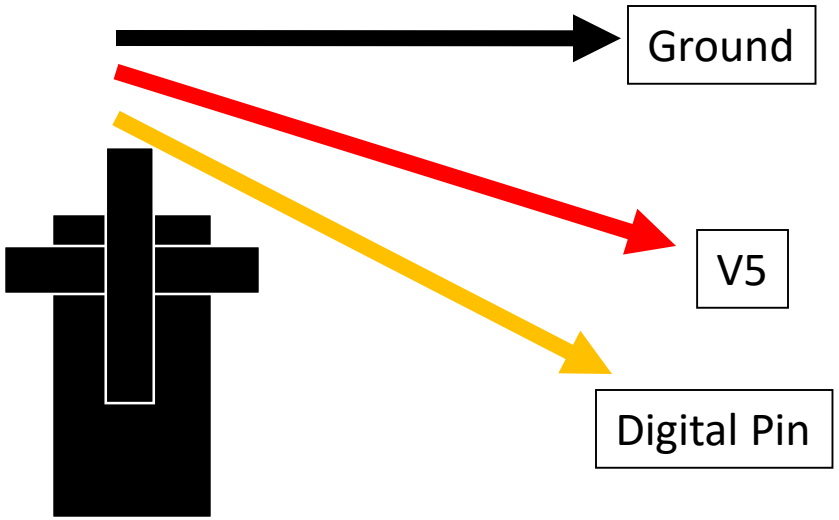
Arduino- DigitalOutput-Motion-Servo Motor

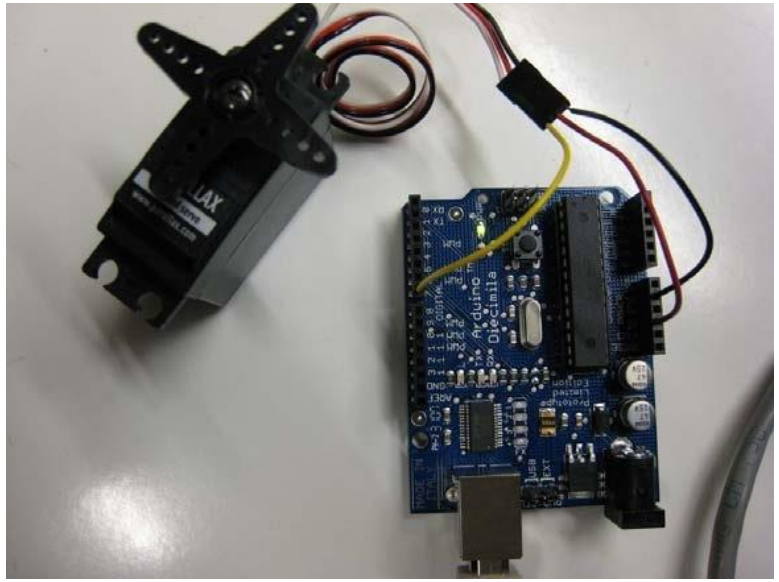


Servo Motors are electronic devices that convert digital signal to rotational movement. There are two sorts of servo motors: Standard servos that their rotation is limited to maximum of 180 degrees in each direction and Continuous Rotation Servos that can provide rotation unlimitedly in both directions

Arduino-DigitalOutput-Motion-Servo Motor

A servo motor is a motor that pulses at a certain rate moving its gear at a certain angle. It has three connections: **the black is ground, the red is connected to 5V, and the white (yellow wire here) is set to the digital pin.**





THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-
Standard Servo Rotation to Exact Angel

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop()
{
  myservo.attach(9);
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  myservo.detach();                  //Detach the servo if you are not controling it for a while
  delay(2000);
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-
Standard Servo Rotation to Exact Angel

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}
void loop()
{
  myservo.attach(9);
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  myservo.detach();                  //Detach the servo if you are not controling it for a while
  delay(2000);
}
```

THE C PROGRAMMING LANGUAGE
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-
Controlling Standard Servo with User Input

```
void setup(){
pinMode(5,OUTPUT); //set the pin to output
Serial.begin(9600); //open the serial to print
Serial.print("Ready"); //write a message
Serial.println();
}
void loop(){
int val = Serial.read(); //read the serial to see
if(val>='0' && val <= '9'){ //which key was pressed
val = val - '0'; //convert the character to an integer
val = val * (180/9); //9 divisions of 180 degrees
Serial.print("moving servo to ");
Serial.print(val);
Serial.println();
for(int a=0; a<100; a++){
int pulseWidth = (val*11)+500; // See the formula above
digitalWrite(5,HIGH);
delayMicroseconds(pulseWidth);
digitalWrite(5,LOW);
delay(10);
}
}
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-
Controlling Standard Servo with User Input

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
int angleValue=0;
int serialNumber=0;
void setup()
{
  Serial.begin(9600);
  myservo.attach(9);
}
void loop()
{
  int value=Serial.read();
  Serial.println(value);
  if(value!=-1 && value!=10){
    serialNumber=serialNumber*10+(value-48);
  }
  if(value==10){
    myservo.attach(9);
    angleValue=serialNumber%180;
    myservo.write(angleValue);            // tell servo to go to position in variable 'pos'
    Serial.print("Number Recieved from Serial Port:");
    Serial.println(serialNumber);
    serialNumber=0;
    delay(250);
  }
  myservo.detach();
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-
Controlling Servo with User Input

THE C PROGRAMMING LANGUAGE

Untitled - Notepad

sketch_mar09a | Arduino 0022

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;    // variable to store the servo position
int angleValue=0;
int serialNumber=0;
void setup()
{
  Serial.begin(9600);
  myservo.attach(9);
}
void loop()
{
  int value=Serial.read();
  Serial.println(value);
  if(value!=-1 && value!=10){
    serialNumber=serialNumber*10+(value-48);
  }
  if(value==10){
    myservo.attach(9);
    angleValue=serialNumber%180;
    myservo.write(angleValue);              // tell servo to go to position in variable 'pos'
    Serial.print("Number Recieved from Serial Port:");
    Serial.println(serialNumber);
    serialNumber=0;
    delay(250);
  }
  myservo.detach();
}
```

Done uploading.

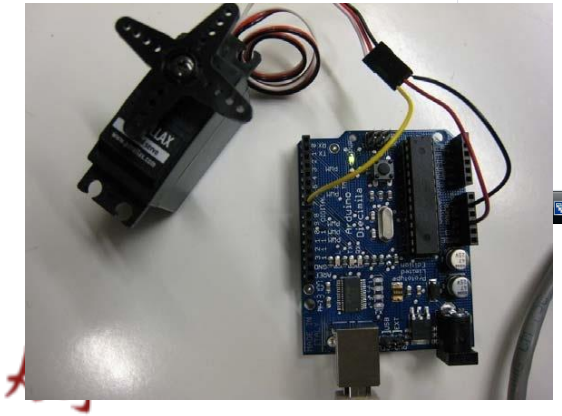
Binary sketch size: 4112 bytes (of a 32256 byte maximum)

COM4

180

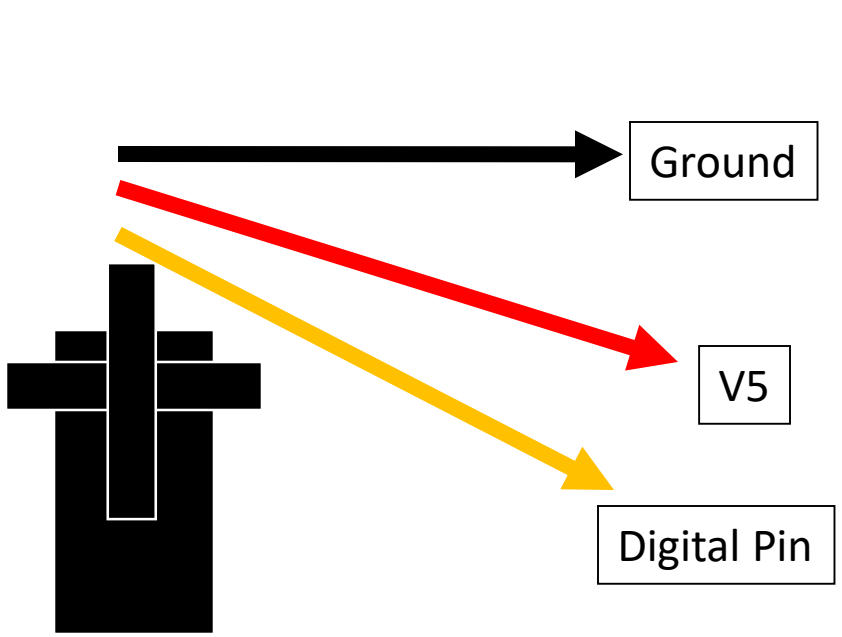
Send

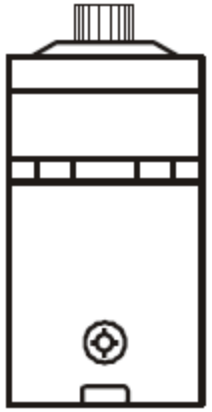
Autoscroll | Newline | 9600 baud

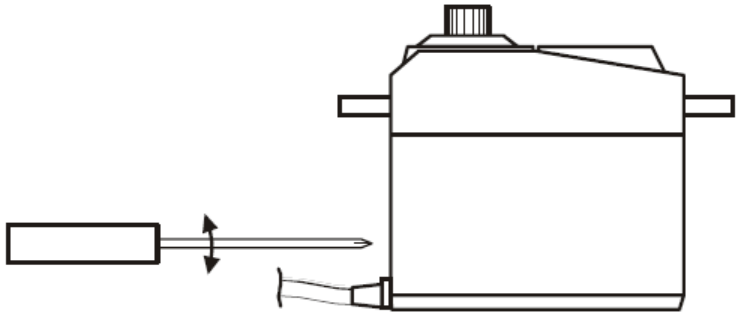


Arduino-
DigitalOutput - Continuous Rotation

As opposed to standard Servo that its rotation is limited to 180 degrees both ways, a continuous rotation servo can keep rotating unlimitedly-again both ways- based on the frequency that is pulsed out to it. There is a specific frequency at which the Servo motor should be static and beyond and before which the servo will change in its rotation direction.

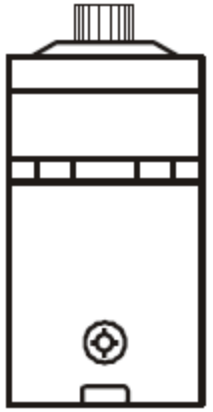


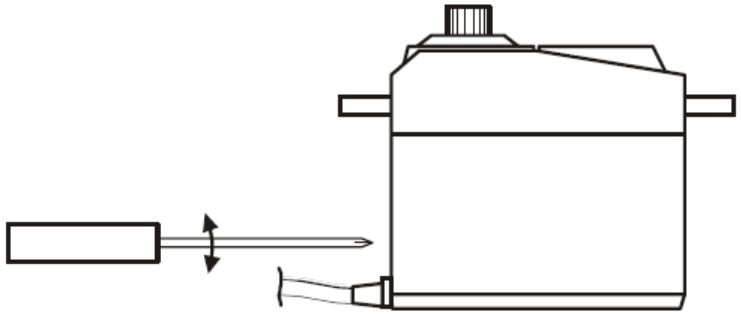




Arduino-
Digital Output - Continuous Rotation-Adjustment

As opposed to standard Servo that its rotation is limited to 180 degrees both ways, a continuous rotation servo can keep rotating unlimitedly-again both ways- based on the frequency that is pulsed out to it. There is a specific frequency at which the Servo motor should be static and beyond and before which the servo will change in its rotation direction.
There is a pin on the servo motor that enables us to adjust the servo for its static frequency.





Arduino-
Digital Output - Continuous Rotation-Adjustment

Upload the following code to the board and while the servo is connected, try to adjust the pin until the servo motor is static.
Once the servo is adjusted to this code any pulse grater than 1500 will result in rotation in one direction while any pulse less than 1500 will result in rotation in the other direction

```
void setup()
{
pinMode(5,OUTPUT);
}
void loop()
{
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1500); // 1.5ms This is the frequency at which the servo motor should be static
digitalWrite(5,LOW);
delay(20); // 20ms
}
}
```


```
void setup()
{
pinMode(5,OUTPUT);
}
void loop()
{
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1500); // 1.5ms This is the frequency at which the servo motor should be static
digitalWrite(5,LOW);
delay(20); // 20ms
}
}
```

Arduino-
Digital Output - Continuous Rotation-Direction Change

```
void setup()
{
pinMode(5,OUTPUT);
}
void loop()
{

//Rotating in One direction
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(20); // 20ms
}

//Stop
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1500);
digitalWrite(5,LOW);
delay(20); // 20ms
}

//Rotating in the other direction
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1200);
digitalWrite(5,LOW);
delay(20); // 20ms
}

//Stop
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1500);
digitalWrite(5,LOW);
delay(20); // 20ms
}
}
```

Once the servo is adjusted to this code any pulse grater than 1500 will result in rotation in one direction while any pulse less than 1500 will result in rotation in the other direction

Arduino-
Digital Output - Continuous Rotation-Direction Change

```
void setup()
{
pinMode(5,OUTPUT);
}
void loop()
{
//Rotating in One direction
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(20); // 20ms
}
//Stop
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1500);
digitalWrite(5,LOW);
delay(20); // 20ms
}
//Rotating in the other direction
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1200);
digitalWrite(5,LOW);
delay(20); // 20ms
}
//Stop
for (int i = 0; i <= 200; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1500);
digitalWrite(5,LOW);
delay(20); // 20ms
}
```

Once the servo is adjusted to this code any pulse grater than 1500 will result in rotation in one direction while any pulse less than 1500 will result in rotation in the other direction

THE C PROGRAMMING LANGUAGE

江西理工大学 JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-
Digital Output - Continuous Rotation-
Delayed Steps

Playing with delay() gives us pauses between rotation steps

```
void setup()
{
pinMode(5,OUTPUT);
}
void loop()
{
//Continious Rotation
for (int i = 0; i <= 20; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(1);
}
//Rotating with delayed steps
for (int i = 0; i <= 20; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(100);
}
//More Delay
for (int i = 0; i <= 20; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(200);
}
//More Delay
for (int i = 0; i <= 20; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(400);
}
//More Delay
for (int i = 0; i <= 20; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(800);
}
//More Delay
for (int i = 0; i <= 20; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(1800);
}
}
```

THE C PROGRAMMING LANGUAGE

Arduino-
Digital Output - Continuous Rotation-Controlling Rotation Angle

```
void setup()
{
pinMode(5,OUTPUT);
}
void loop()
{
for (int i = 0; i <= 10; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(20);
}
delay(1000);
for (int i = 0; i <= 20; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(20);
}
delay(1000);
for (int i = 0; i <= 30; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(20);
}
delay(1000);
for (int i = 0; i <= 40; i++)
{
digitalWrite(5,HIGH);
delayMicroseconds(1800);
digitalWrite(5,LOW);
delay(20);
}
delay(1000);
}
```

Playing with the number of steps in the for loop gives us variations in the span /Angle of the rotation

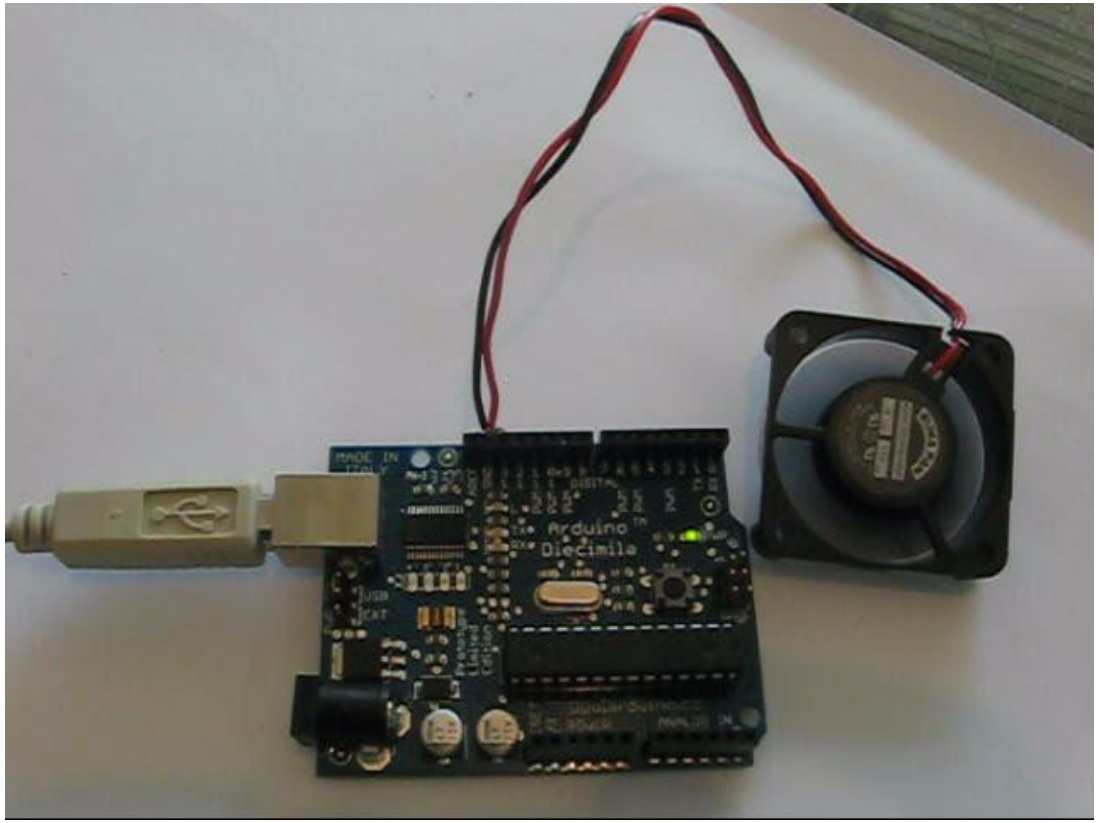
THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Arduino-
Digital Output – Wind –Controlling a Fan

Controlling a Fan is as easy as sending a HIGH or LOW Signal to the Pin that the fan is connected to.

```
// Connect the fan to Pin 13 and Ground
void setup(){
pinMode(13, OUTPUT); // Specify Arduino Pin number and
output/input mode
}
void loop(){
digitalWrite(13, HIGH); // Turn on Pin 13 sending a HIGH Signal
delay(1000);            // Wait for one second
digitalWrite(13, LOW);  // Turn off Pin 13 sending a LOW Signal
delay(3000);            // Wait for Three second
}
```



Arduino-
Digital Output – Rotation –Controlling a DC Motor

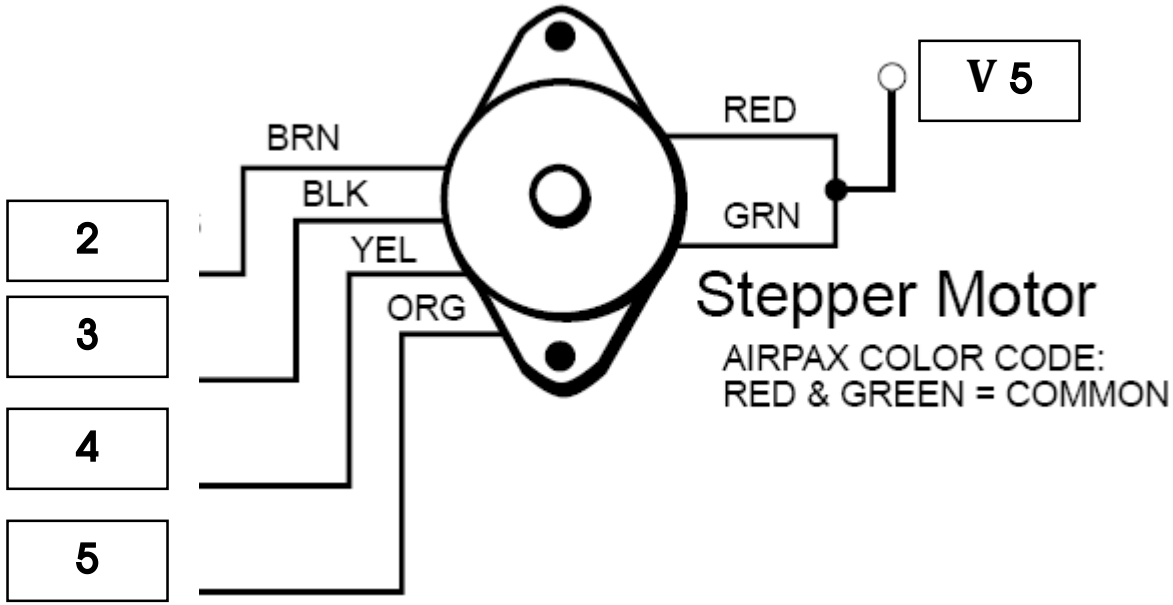
```
// Connect to Pin 13 and Ground
void setup(){
pinMode(13, OUTPUT); // Specify Arduino Pin number
and output/input mode
}
void loop(){
digitalWrite(13, HIGH); // Turn on Pin 13 sending a
HIGH Signal
delay(1000);          // Wait for one second
digitalWrite(13, LOW);  // Turn off Pin 13 sending a
LOW Signal
delay(3000);          // Wait for Three second
}
```

Code for Rotation/No Rotation

```
// Connect to Pin 13 and 12
void setup(){
pinMode(13, OUTPUT); // Specify Arduino Pin number
and output/input mode
pinMode(12, OUTPUT);
}
void loop(){
digitalWrite(13, HIGH); // Turn on Pin 13 sending a
HIGH Signal
digitalWrite(12, LOW);  //Make Pin 12 a Ground
delay(1000);          // Wait for one second
digitalWrite(13, LOW);  // Make Pin 13 a Ground
digitalWrite(12, HIGH); // Turn on Pin 12 sending a
HIGH Signal
delay(3000);          // Wait for Three second
}
```

Code for CW and CCW Rotation

Arduino- Digital Output–Rotation–Stepper Motor



Stepper motors translate digital switching sequences into motion. They are used in a variety of applications requiring precise motions under computer control.

Unlike ordinary dc motors, which spin freely when power is applied,steppers require that their power source be continuously pulsed in specific patterns. These patterns, or step sequences, determine the speed and direction of a stepper's motion.

For each pulse or step input, the stepper motor rotates a fixed angular increment; typically 1.8 or 7.5 degrees.

Steppers are driven by the interaction (attraction and repulsion) of magnetic fields. The driving magnetic field "rotates" as strategically placed coils are switched on and off. This pushes and pulls at permanent magnets arranged around the edge of a rotor that drives the output shaft.

Unipolar Stepper Motor

Full Step, Low Torque

Full Step, High Torque (standard application)

Half Step (best precision):

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Stepper Motor

When the on-off pattern of the magnetic fields is in the proper sequence, the stepper turns (when it's not, the stepper sits and quivers).

The most common stepper is the four-coil unipolar variety. These are called unipolar because they require only that their coils be driven on and off. Bipolar steppers require that the polarity of power to the coils be reversed.

	Step Sequence				
	1	2	3	4	1
coil 1	1	1	0	0	1
coil 2	0	0	1	1	0
coil 3	1	0	0	1	1
coil 4	0	1	1	0	0

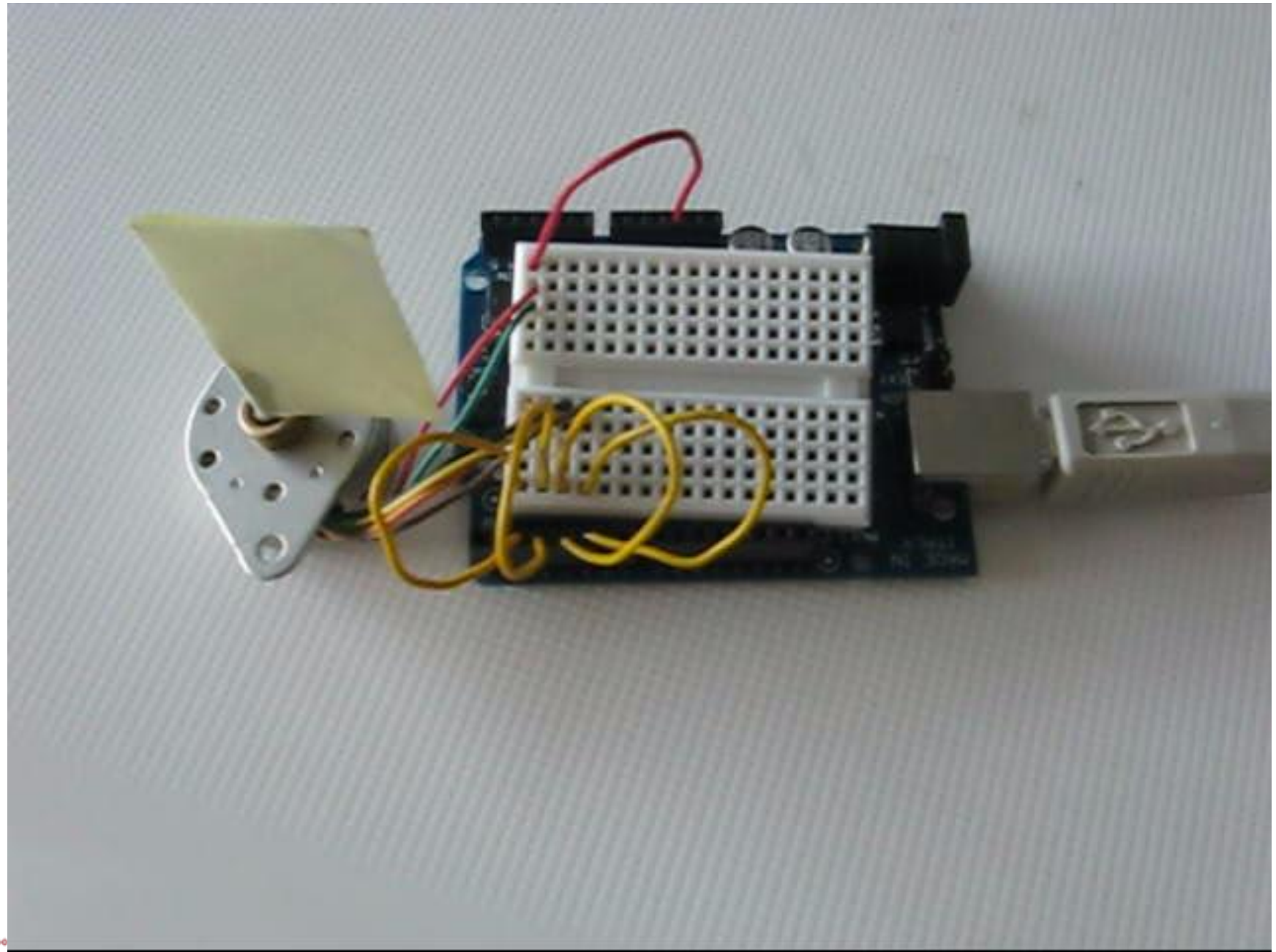
Figure 2. Normal stepping sequence.

The normal stepping sequence for four-coil unipolar steppers appears in the figure. If you run the stepping sequence in the figure forward, the stepper rotates clockwise; run it backward, and the stepper rotates counterclockwise.

The motor's speed depends on how fast the controller runs through the step sequence. At any time the controller can stop in mid sequence.

If it leaves power to any pair of energized coils on, the motor is locked in place by their magnetic fields. This points out another stepper motor benefit: built-in brakes.

Stepper Motor



THE C PROGRAMMING LANGUAGE

江西理工大
JIANGXI UNIVERSITY OF SCIENCE AND TECHNO

Stepper Motor-Direction and Speed

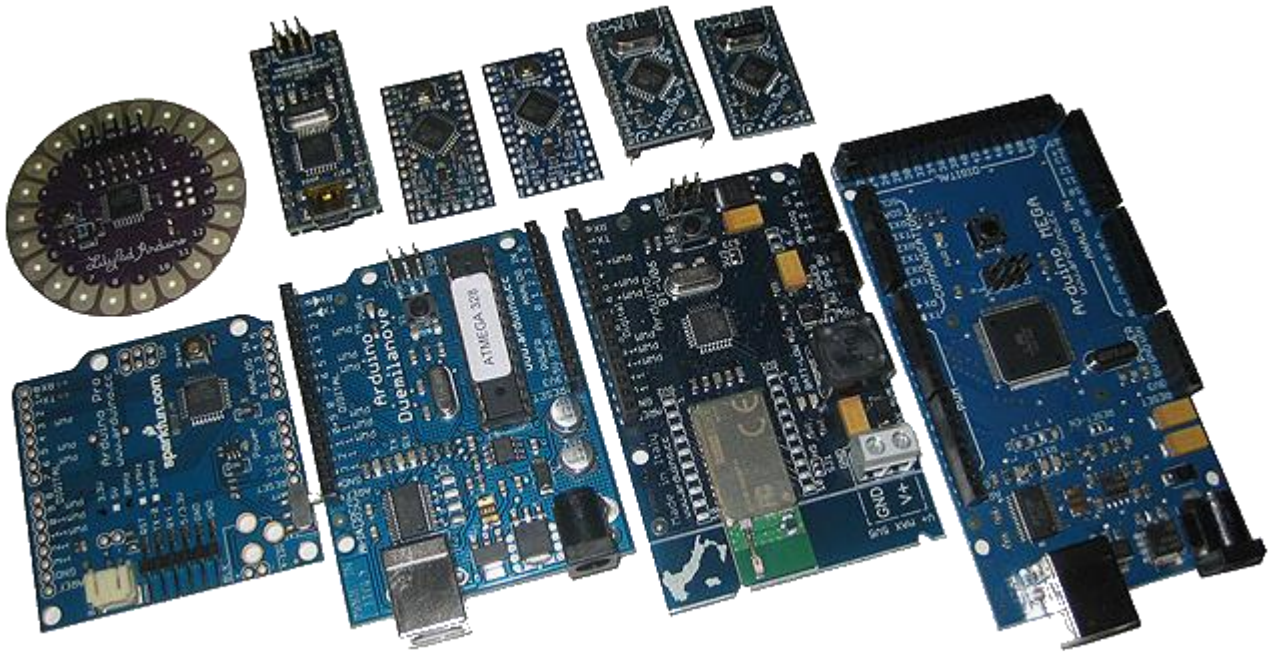
```
void setup(){
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
}
void loop(){
  // Pause between the types that determines the speed
  int stepperSpeed=200;// Change to change speed
  int dir=1;// change to -1 to change direction
  if (dir==1){  //Running Clockwise
  digitalWrite(2,HIGH);//Step 1
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  delay(stepperSpeed);// Pause between the types that determines the speed
  digitalWrite(2,HIGH);//Step 2
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  delay(stepperSpeed);// Pause between the types that determines the speed
  digitalWrite(2,LOW);//Step 3
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  delay(stepperSpeed);// Pause between the types that determines the speed
  digitalWrite(2,LOW);//Step 4
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  delay(stepperSpeed);// Pause between the types that determines the speed
  }
  if (dir==-1){  //Running CounterClockwise
  digitalWrite(2,LOW);//Step 4
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  delay(stepperSpeed);// Pause between the types that determines the speed
  digitalWrite(2,LOW);//Step 3
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  delay(stepperSpeed);// Pause between the types that determines the speed
  digitalWrite(2,HIGH);//Step 2
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  delay(stepperSpeed);// Pause between the types that determines the speed
  digitalWrite(2,HIGH);//Step1
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  delay(stepperSpeed);// Pause between the types that determines the speed
  }
}
```

THE C PROGRAMMING LANGUAGE

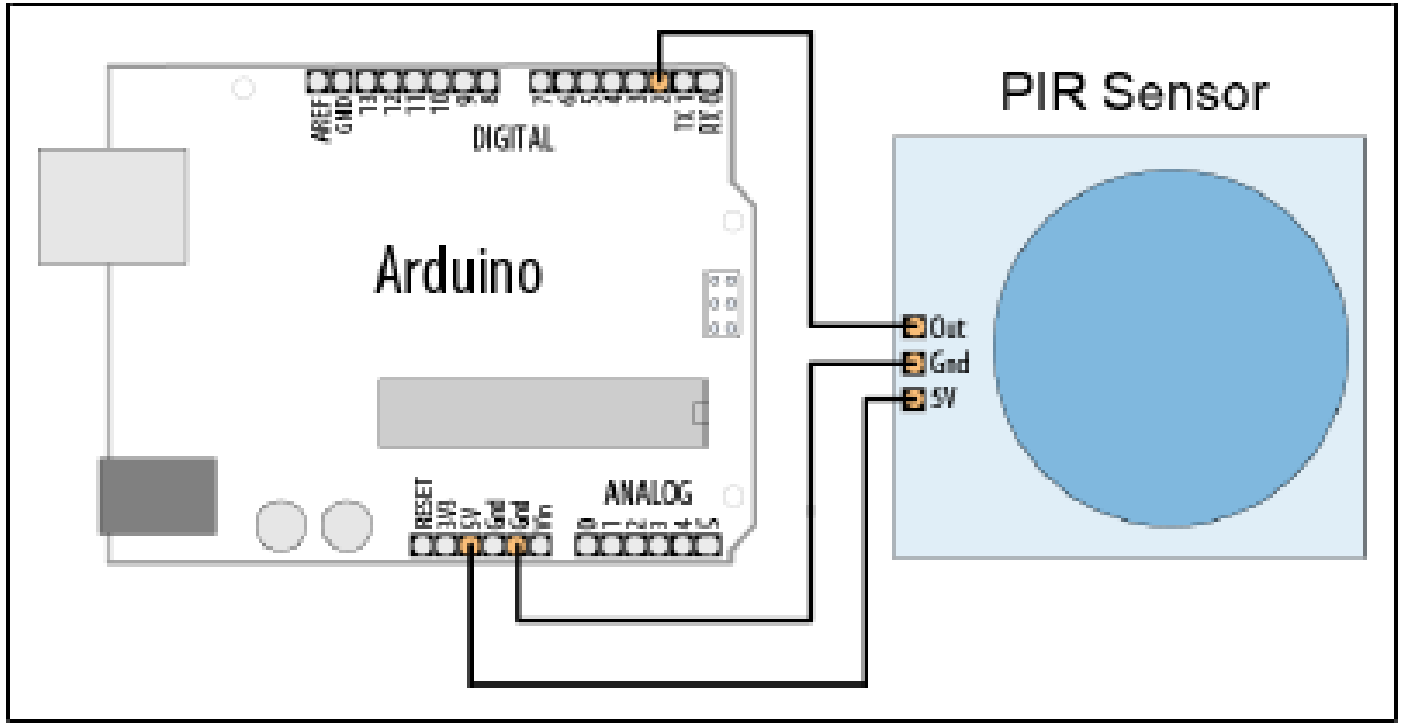
江西理工大学
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

مرکز آموزش علمی-کاربردی گروه انتخاب

Lecture 4:
Introduction To LCD



Using PIR motion sensors



Using PIR motion sensors

```
const int ledPin = 77; // pin for the LED
const int inputPin = 2; // input pin (for the PIR sensor)
void setup() {
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop(){
int val = digitalRead(inputPin); // read input value
if (val == HIGH) // check if the input is HIGH
{
digitalWrite(ledPin, HIGH); // turn LED on if motion detected
delay(500);
digitalWrite(ledPin, LOW); // turn LED off
}
}
```

Using ultrasonic sensors

- The “ping” sound pulse is generated when the pingPin level goes HIGH for two microseconds.
- The sensor will then generate a pulse that terminates when the sound returns.
- The width of the pulse is proportional to the distance the sound traveled
- The speed of sound is 340 meters per second, which is 29 microseconds per centimeter. The formula for the distance
- of the round trip is: RoundTrip = microseconds / 29

Using ultrasonic sensors

```
const int pingPin = 5;
const int ledPin = 77; // pin connected to LED
void setup()
{
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
}
void loop()
{
int cm = ping(pingPin) ;
Serial.println(cm);
digitalWrite(ledPin, HIGH);
delay(cm * 10 ); // each centimeter adds 10 milliseconds delay
digitalWrite(ledPin, LOW);
delay( cm * 10);
}
```

Using ultrasonic sensors

```
int ping(int pingPin)
{
long duration, cm;
pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(5);
digitalWrite(pingPin, LOW);
pinMode(pingPin, INPUT);
duration = pulseIn(pingPin, HIGH);
// convert the time into a distance
cm = microsecondsToCentimeters(duration);
return cm ;
}
long microsecondsToCentimeters(long microseconds)
{
// The speed of sound is 340 m/s or 29 microseconds per centimeter.
// The ping travels out and back, so to find the distance of the
// object we take half of the distance travelled.
return microseconds / 29 / 2;
}
```

Audio output

```
tone(speakerPin, frequency, duration); // play the tone
delay(duration); //wait for the tone to finish
```

THE C PROGRAMMING LANGUAGE

Example

- Write a program that plays an A note (440 Hz) for 1 second when a motion sensor detects motion.

THE C PROGRAMMING LANGUAGE

Using Interrupts

- On a standard Arduino board, two pins can be used as interrupts: pins 2 and 3.
- The interrupt is enabled through the following line:
- attachInterrupt(interrupt, function, mode)
  - attachInterrupt(0, doEncoder, FALLING);

Interrupt example

```
int led = 77;
  volatile int state = LOW;

  void setup()
  {
    pinMode(led, OUTPUT);
    attachInterrupt(1, blink, CHANGE);
  }

  void loop()
  {
    digitalWrite(led, state);
  }

  void blink()
  {
    state = !state;
  }
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Timer functions (timer.h library)

- int every(long period, callback)
  - Run the 'callback' every 'period' milliseconds. Returns the ID of the timer event.
- int every(long period, callback, int repeatCount)
  - Run the 'callback' every 'period' milliseconds for a total of 'repeatCount' times. Returns the ID of the timer event.
- int after(long duration, callback)
  - Run the 'callback' once after 'period' milliseconds. Returns the ID of the timer event.

Timer functions (timer.h library)

- int oscillate(int pin, long period, int startingValue)
  - Toggle the state of the digital output 'pin' every 'period' milliseconds. The pin's starting value is specified in 'startingValue', which should be HIGH or LOW. Returns the ID of the timer event.
- int oscillate(int pin, long period, int startingValue, int repeatCount)
  - Toggle the state of the digital output 'pin' every 'period' milliseconds 'repeatCount' times. The pin's starting value is specified in 'startingValue', which should be HIGH or LOW. Returns the ID of the timer event.

Timer functions (Timer.h library)

- int pulse(int pin, long period, int startingValue)
  - Toggle the state of the digital output 'pin' just once after 'period' milliseconds. The pin's starting value is specified in 'startingValue', which should be HIGH or LOW. Returns the ID of the timer event.
- int stop(int id)
  - Stop the timer event running. Returns the ID of the timer event.
- int update()
  - Must be called from 'loop'. This will service all the events associated with the timer.

Example (1/2)

```
#include "Timer.h"
Timer t;
int ledEvent;
void setup()
{
  Serial.begin(9600);
  int tickEvent = t.every(2000, doSomething);
  Serial.print("2 second tick started id=");
  Serial.println(tickEvent);
  pinMode(13, OUTPUT);
  ledEvent = t.oscillate(13, 50, HIGH);
  Serial.print("LED event started id=");
  Serial.println(ledEvent);
  int afterEvent = t.after(10000, doAfter);
  Serial.print("After event started id=");
  Serial.println(afterEvent);
}
```

Example (2/2)

```
void loop()
{
  t.update();
}

void doSomething()
{
  Serial.print("2 second tick: millis()=");
  Serial.println(millis());
}

void doAfter()
{
  Serial.println("stop the led event");
  t.stop(ledEvent);
  t.oscillate(13, 500, HIGH, 5);
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Chipkit Max32 arduino board

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ChipKit MAX32

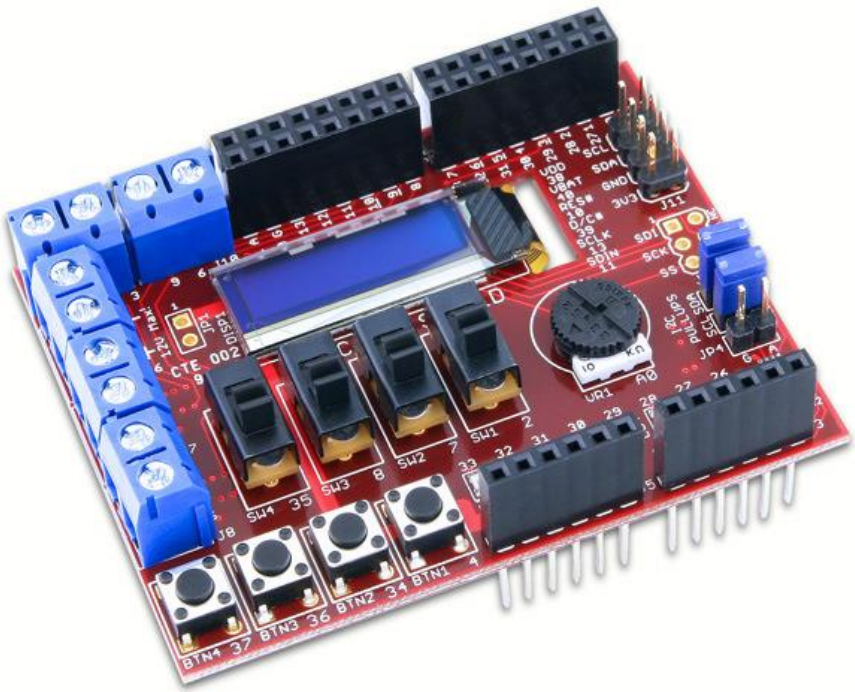
- Microcontroller: PIC32MX795F512L
- Flash Memory: 512K
- RAM Memory: 128K
- Operating Voltage: 3.3V
- Operating Frequency: 80Mhz
- Typical operating current: 90mA
- Input Voltage (recommended): 7V to 15V
- Input Voltage (maximum): 20V
- I/O Pins: 83 total
- Analog Inputs: 16
- Analog input voltage range: 0V to 3.3V
- DC Current per pin: +/-18mA
- Advanced peripherals:
- 10/100 Ethernet MAC
- USB 2.0 Full Speed OTG controller
- 2 CAN controllers.
- **External Interrupts**: Pin 3 (INT0), Pin 2 (INT1), Pin 7 (INT2), Pin 21 (INT3), Pin 20 (INT4)

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Basic I/O Shield

- Features
  - 128x32 pixel OLED graphic display
  - I2C temperature sensor
  - 256Kbit I2C EEPROM
  - I2C daisy chain connector
  - 4 push buttons
  - 4 slide switches
  - 8 discrete LEDs
  - 4 open drain FET drivers
  - Analog potentiometer



THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Temperature Sensor

- A digital temperature sensor is provided using a Microchip TCN75A 2-Wire Serial Temperature Sensor. The temperature sensor, IC2, is an I2C device, and is located in the lower right corner of the board.
- The TCN75A is rated for an accuracy of +/-1ºC and has selectable resolution from 0.5ºC down to 0.0625ºC. The seven bit device address is ‘1001000’.
- Digilent provides a library for accessing the temperature sensor. This library is available on the Digilent web site and in the third party library repository on github.
- Using the temperature sensor with the MAX32 board requires manually connecting the SDA and SCL pins to the basic I/O shield

Configuring Temperature sensor

- **void config(uint8_t configuration)**
- Parameters
- configuration - Value to be written to config register
- This function writes the configuration register with the given value. There are a number of defined
- values as described below that can be or’d together to set multiple parameters. For example if one
- wished to put the device in one shot mode and use 12 bit resolution the following call could be made.
- Config(ONESHOT | RES12)
- IOSHIELDTEMP_ONESHOT 0x80 //One Shot mode
- IOSHIELDTEMP_RES9 0x00 //9-bit resolution (0.5ºC)
- IOSHIELDTEMP_RES10 0x20 //10-bit resolution
- IOSHIELDTEMP_RES11 0x40 //11-bit resolution
- IOSHIELDTEMP_RES12 0x60 //12-bit resolution (0.0625ºC)
- IOSHIELDTEMP_FAULT1 0x00 //1 fault queue bits
- IOSHIELDTEMP_FAULT2 0x08 //2 fault queue bits
- IOSHIELDTEMP_FAULT4 0x10 //4 fault queue bits
- IOSHIELDTEMP_FAULT6 0x18 //6 fault queue bits
- IOSHIELDTEMP_ALERTLOW 0x00 //Alert bit active-low
- IOSHIELDTEMP_ALERTHIGH 0x04 //Alert bit active-high
- IOSHIELDTEMP_CMPMODE 0x00 ///comparator mode.
- IOSHIELDTEMP_INTMODE 0x02 //interrupt mode
- IOSHIELDTEMP_STARTUP 0x00 //Shutdown disabled
- IOSHIELDTEMP_SHUTDOWN 0x01 //Shutdown enabled
- IOSHEIDLTEMP_CONF_DEFAULT //Power up initial configuration

Reading Temperature sensor

- **float getTemp()**
- Retrieves the current temp from the temp sensor and converts the returned value into a signed floating point value.

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Example

```
void setup(){
IOShieldTemp.config(IOSHIELDTEMP_ONESHOT |
  IOSHIELDTEMP_RES11 | IOSHIELDTEMP_ALERTHIGH);
}


void loop()
{
  float temp;
  int celsius;
  char sign, msd_char, lsd_char;
  //Get Temperature in Celsius.
  temp = IOShieldTemp.getTemp();
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Potentiometer

- A potentiometer is provided on the board
- to be used as an analog signal source or
- analog control input. The pot is a 10Kohm
- trimmer pot connected between the VCC3V3
- supply and ground. The wiper of the pot is
- connected to analog input A0.
- The pot is read using the analogRead function.

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

OLED Display

- 128x32 pixels
- Each individual pixel can only be on (illuminated) or off (not illuminated)
- The display is a serial device that is accessed via an SPI interface.
- write-only device

THE C PROGRAMMING LANGUAGE

Using the OLED display

- Initialization
- Mode select
  - Character
  - Graphic
  - Drawing

THE C PROGRAMMING LANGUAGE

Example

```
void setup()
{
IOShieldOled.begin();
}

void loop()
{
char toprint;

  IOShieldOled.clearBuffer();
    IOShieldOled.setCursor(0, 0);

toprint = ‘A’;

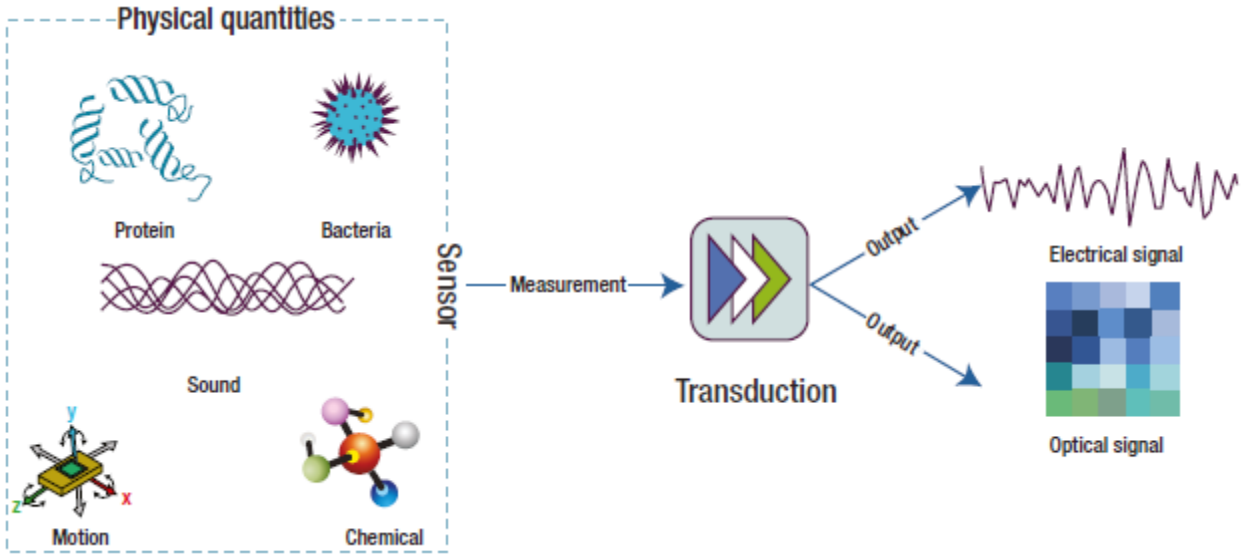
IOShield0led.putChar(toprint);
  }
```

THE C PROGRAMMING LANGUAGE

Introduction to Sensors

What is a sensor?

- *A device that receives a stimulus and responds with an electrical signal.*
- *A special type of transducer (device that converts one type of energy into another*



Common Sensors

- Mechanical
  - Accelerometers
  - Gyroscopes
- Optical
  - Photodetectors
  - Infrared
- Semiconductor
  - Gas
  - Temperature
  - Magnetic

THE C PROGRAMMING LANGUAGE

Example: Simple temperature sensor

- A RTD is a *thermoresistive temperature sensor. It is a metal element (in a ceramic tube) whose resistance* typically increases with temperature, according to a known function.
- A linear approximation is given by

$$R = R_0(1 + \alpha T)$$

- Where a is the temperature coefficient, T is the temperature in Kelvin and R0 the resistance in ture

Example

- Calculate the temperature for a copper RTD if R0 = 500Ω and room temperature and a = 0.0043. The measured resistance is 500.43Ω

THE C PROGRAMMING LANGUAGE

Sensor Characteristics (1/4)

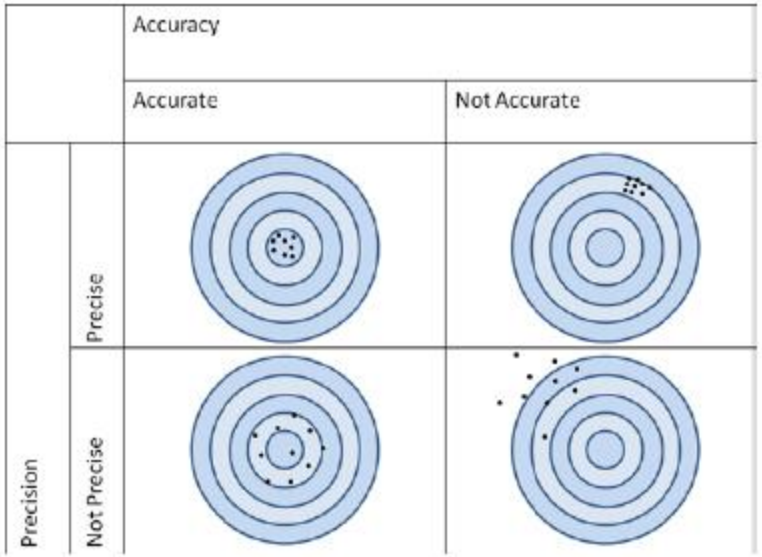
- Range
  - Full Scale Range
  - Operating Voltage Range
- Accuracy
- Transfer Function
  - S=F(x), where x is the measurand and S the electrical signal (commonly Voltage)
- Sensitivity
  - The change in input required to generate a unit change in output

Sensor Characteristics (2/4)

- Accuracy
- Precision

$$Percentage\ Relative\ Error = \frac{(Measured\ Value - True\ Value)}{(True\ Value)} \times 100$$

$$Percentage\ Standard\ Deviation = \frac{(Standard\ Deviation)}{(Mean)} \times 100$$

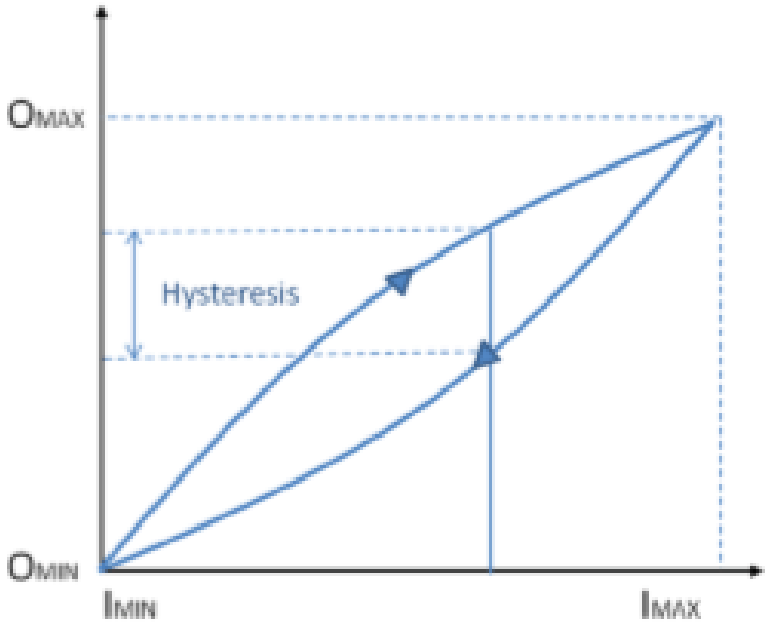


Sensor Characteristics (3/4)

- Error: the difference between the measured value and true value
- Systematic errors are reproducible inaccuracies that can be corrected with compensation methods
  - Interference errors
  - Operator errors etc.
- Random error
  - Noise

Sensor Characteristics (4/4)

- Hysterysis: the difference in output between the rising and falling output values for a given input



Example: Smoke sensor (1/2)

- An MQ-2 smoke sensor reports smoke by the voltage level it puts out.
- The more smoke there is, the higher the voltage.
- built-in potentiometer for adjusting sensitivity
- Three pins:
  - Vdd input
  - Ground input
  - Analog output

Smoke sensor (2/2)

```
const int smokePin = 54; //sensor input
void setup() {
  pinMode(smokePin, INPUT);
  Serial.begin(9600);
}
void loop() {
int smoke_level = analogRead(smokePin); //read sensor
Serial.println(smoke_level);
  if(smoke_level > 120) {     //calibrate accordingly
    Serial.println("Smoke detected");
  }
  delay(100); // ms
}
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

References

- M.J. McGrath, C. N. Scanaill and D. Nafus, “Sensor Technologies Healthcare, Wellness and Environmental Applications”, Apress, 2013
- C.W. de Silva, “Sensors and Actuators: Control System Instrumentation”, 2nd Edition, CRC Press, 2015
- T. Karvinen and K. Karvinen, ” Make: Sensors: A Hands-On Primer for Monitoring the Real World with Arduino and Raspberry Pi”, Maker Media, Inc, 2014

THE C PROGRAMMING LANGUAGE

Internet of things with arduino

Arduino with GPS (1/4)

Three modes:

- Stand-alone: GPS obtains information such as position and altitude with only the signal of the satellites
  - slowest of the three modes
  - AT+CGPSINFO command brings directly latitude, logtitude, date, UTC time, altitude and speed
- S-GPS: module connects to a GPS server
  - module calculates the position
- A-GPS: Three modes
  - MS-Assisted: Server sends data and calculates position
  - MS-based: Server sends aiding data, module calculates position
  - Stand-Alone: Module demodulates GPS data and calculates position

Arduino with GPS (2/4)

- AT+CGPSINFO returns the GPS info in a string:
- [<latitude>],[<N/S>],[<longitude>],[<E/W>],[<date>],[<UTC_time>],[<altitude>],[<speedOG>],[<course>]

Arduino with GPS (3/4)

```
int8_t answer;
int onModulePin= 2;
char gps_data[100];   //will perform 100 GPS data reads
int counter;
void setup(){
    pinMode(onModulePin, OUTPUT);
    Serial.begin(115200);
    Serial.println("Starting...");
    power_on();
   delay(5000); // starts GPS session in stand alone mode
  }
}
void power_on(){   //void power_on() should be placed AFTER void loop(), used here for lecture
    digitalWrite(onModulePin,HIGH);    //GPS module requires a 3 sec pulse on onModulePin
    delay(3000);
    digitalWrite(onModulePin,LOW);
}
```

Arduino with GPS (4/4)

```
void loop(){
    answer = sendATcommand("AT+CGPSINFO","+CGPSINFO:",1000);
    // request info from GPS
    if (answer == 1) {
        counter = 0;
        do{
         while(Serial.available() == 0);          //reading GPS data
         gps_data[counter] = Serial.read();
         counter++;
         }
        }
          Serial.print("GPS data:");              //printing GPS data
         Serial.print(gps_data);
    }
```

THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Connecting through 3G (1/2)

```
int8_t answer;
int onModulePin = 2, aux;
char aux_str[50];
void setup(){
pinMode(onModulePin, OUTPUT);
Serial.begin(115200);
Serial.println("Starting...");
power_on();
delay(3000); //sets the PIN code
sprintf(aux_str, "AT+CPIN=%s", pin_number);
sendATcommand(aux_str, "OK", 2000);
delay(3000);
```

Connecting through 3G (2/2)

```
while( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) ||
   sendATcommand("AT+CREG?", "+CREG: 0,5", 500)) == 0 ); // sets APN, user
   name and password
sprintf(aux_str, "AT+CGSOCKCONT=1,\"IP\",\"%s\"", apn);
sendATcommand(aux_str, "OK", 2000);
sprintf(aux_str, "AT+CSOCKAUTH=1,1,\"%s\",\"%s\"", user_name, password);
sendATcommand(aux_str, "OK", 2000);
}
```

Arduino-Controlling Actuators Based on Input from Arduino Serial Port

THE C PROGRAMMING LANGUAGE

sketch_mar09a | Arduino 0022

File Edit Sketch Tools Help

Serial Monitor

sketch_mar09a §

```
//pin 11,10,9,6,5,3 can be used for Analog output
int serialNumber=0;
int lightIntensityValue=0;
void setup(){
  Serial.begin(9600);
  pinMode(11, OUTPUT); // Specify Arduino Pin number and output/input mode
}

void loop(){
  int value=Serial.read();
  Serial.println(value);
  if(value!=-1 && value!=10){
    serialNumber=serialNumber*10+(value-48);
  }
  if(value==10){
    lightIntensityValue=serialNumber%255;
    Serial.print("Number Recieved from Serial Port:");
    Serial.println(serialNumber);
    serialNumber=0;
  }
  analogWrite(11,lightIntensityValue);
  delay(1000);
}
```

Done uploading.

```
Binary sketch size: 2918 bytes (of a 32256 byte maximum)
```

17

COM4

Send

```
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
49
50
51
10
Number Recieved from Serial Port:123
-1
-1
```

Autoscroll | Newline | 9600 baud

Arduino-Controlling Actuators Based on Input from Arduino Serial Port

```
//pin 11,10,9,6,5,3 can be used for Analog output
int serialNumber=0;
int lightIntensityValue=0;
void setup(){
  Serial.begin(9600);
  pinMode(11, OUTPUT); // Specify Arduino Pin number and output/input mode
}

void loop(){
  int value=Serial.read();
  Serial.println(value);
  if(value!=-1 && value!=10){
    serialNumber=serialNumber*10+(value-48);
  }
  if(value==10){
    lightIntensityValue=serialNumber%255;
    Serial.print("Number Recieved from Serial Port:");
    Serial.println(serialNumber);
    serialNumber=0;
  }
  analogWrite(11,lightIntensityValue);
  delay(1000);
}
```

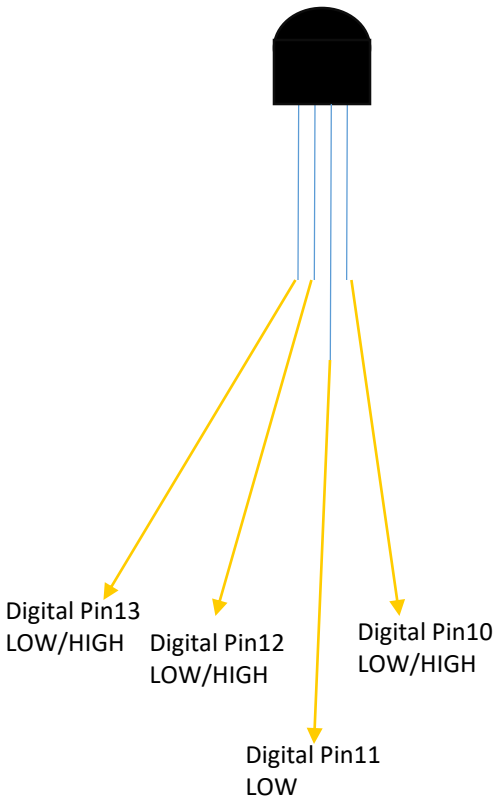
1. Data is received from Serial port as ASCII codes.
2. If data is numerical, each digit is sent separately.
3. ASCII code of zero is 48
4. To calculate the numerical value of a digit from its ASCI code: digit=**ASCII-48**
5. At the end of a package the serial port send a number **10**
6. If nothing is passed to the serial port, the port sends number **-1** as default

3-Color LED

```
void setup(){
pinMode(13,OUTPUT);
pinMode(12,OUTPUT);
pinMode(11,OUTPUT);
pinMode(10,OUTPUT);
}
void loop(){
digitalWrite(11,LOW);

digitalWrite(13,HIGH);
digitalWrite(12,LOW);
digitalWrite(10,LOW);
delay(1000);
digitalWrite(13,LOW);
digitalWrite(12,HIGH);
digitalWrite(10,LOW);
delay(1000);
digitalWrite(13,LOW);
digitalWrite(12,LOW);
digitalWrite(10,HIGH);
delay(1000);

}
```



**Make sure you are not doing the circuit vice versa!!!
** Sometimes the long leg should be high and the leg which is low would determine the color of the light

Arduino-
Digital Output-Sound-Piezo

A Piezo is an electronic piece that converts electricity energy to sound. It is a digital output device. You can make white noise or even exact musical notes (frequencies for musical notes) based on the duration that you iterate between HIGH and LOW signals.

A Piezo is a directional piece, meaning that it has a positive and negative pole. The positive pole should be connected to the digital output pin that you allocate to control the piezo and the negative pole should be connected to Ground pin

Arduino-
Digital Output-Sound-Piezo

```
//connect piezo to pin 13 and ground
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
void setup(){
  pinMode(13,OUTPUT);
}
void loop(){
  for(int i=0;i<8;i++){//iterating through notes
    for(int j=0;j<1000;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(freqs[i]);
    digitalWrite(13,LOW);
    delayMicroseconds(freqs[i]);
    }
  }
}
```

Arduino-
Digital Output-Sound-Piezo

```
//connect piezo to pin 13 and ground
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
void setup(){
  pinMode(13,OUTPUT);
}
void loop(){
  for(int i=0;i<8;i++){//iterating through notes
    for(int j=0;j<1000;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(freqs[i]);
    digitalWrite(13,LOW);
    delayMicroseconds(freqs[i]);
    }
  }
}
```

Arduino-
Digital Output-Sound-Piezo-Playing a melody

```
//connect piezo to pin 13 and ground
void playNote(int note)
{
  for(int j=0;j<60;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(note);
    digitalWrite(13,LOW);
    delayMicroseconds(note);
  }
  delay(60);
}
int pause=200;
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
//              i = { 0     1     2     3    4     5     6     7
//mi mi mi - mi mi mi - mi sol do re mi - - - fa fa fa fa fa mi mi mi mi re re mi re - sol - mi mi mi - mi mi mi - mi sol do re mi -- fa fa fa fa fa mi mi mi sol sol fa re do - - -
void setup(){
  pinMode(13,OUTPUT);


}
void loop(){
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);   playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);        delay(pause);         delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);   playNote(freqs[2]);
  playNote(freqs[2]);  playNote(freqs[1]);  playNote(freqs[1]);   playNote(freqs[2]);
  playNote(freqs[1]);  delay(pause);        playNote(freqs[4]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);   playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);        delay(pause);         delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);   playNote(freqs[2]);
  playNote(freqs[4]);  playNote(freqs[4]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[0]);  delay(pause);        delay(pause);         delay(pause);
}
```

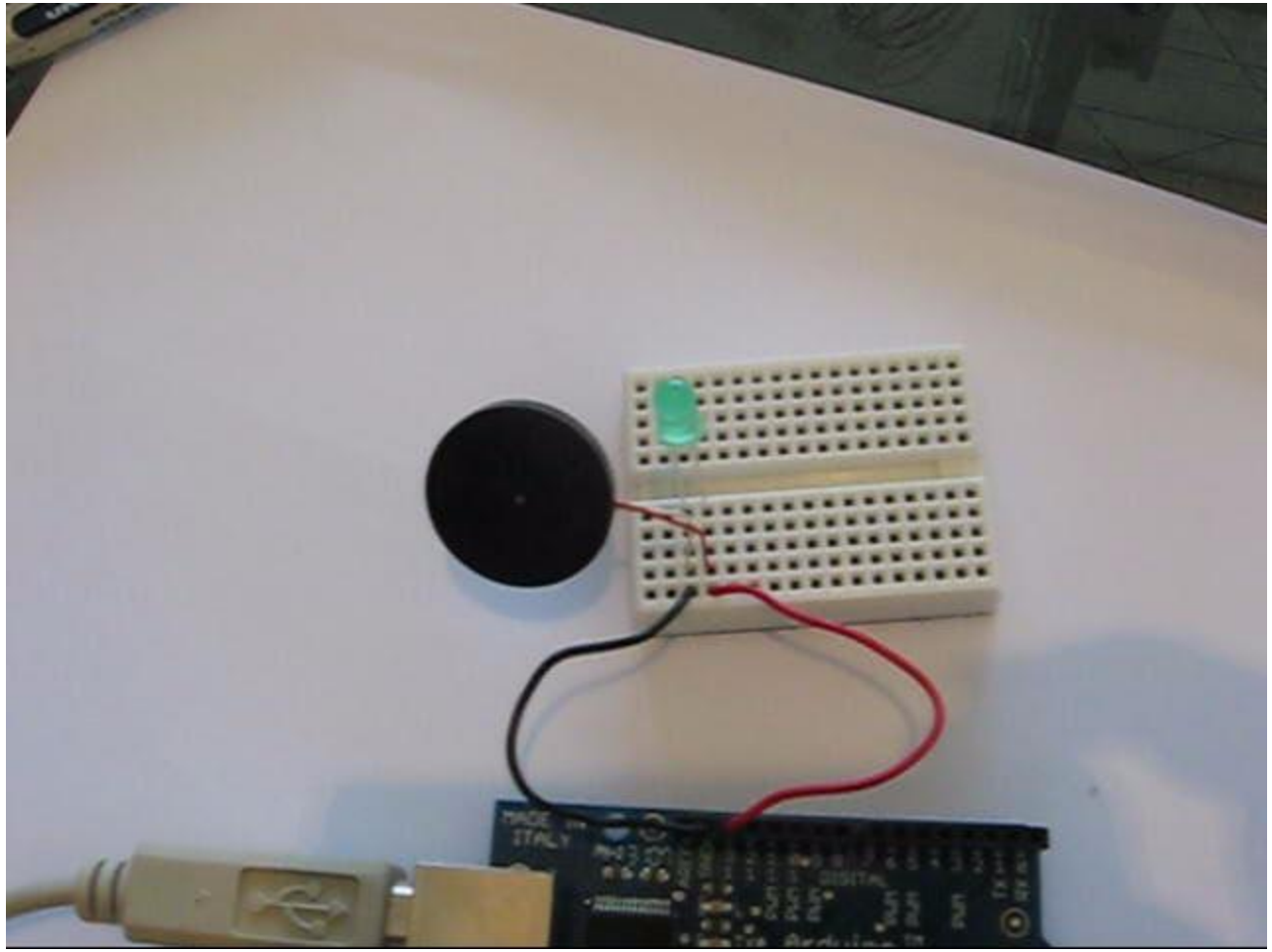
THE C PROGRAMMING LANGUAGE

江西理工大学
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

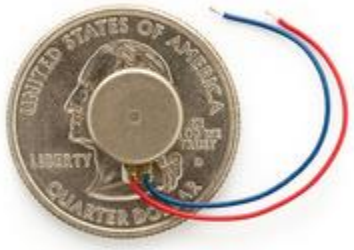
Arduino-
Digital Output-Sound-Piezo-Playing a melody

```
//connect piezo to pin 13 and ground
void playNote(int note)
{
  for(int j=0;j<60;j++){//the time span that each note is being played
    digitalWrite(13,HIGH);
    delayMicroseconds(note);
    digitalWrite(13,LOW);
    delayMicroseconds(note);
  }
  delay(60);
}
int pause=200;
int  freqs[] = {
  1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};
//string  tones[] = {"do", "re", "mi", "fa","sol"," la", "si", "do"};
//            i = { 0     1     2     3    4     5     6     7
//mi mi mi - mi mi mi - mi sol do re mi - - - fa fa fa fa fa mi mi mi mi re re mi re - sol - mi mi mi - mi mi mi - mi sol do re mi -- fa fa fa fa fa mi mi mi sol sol fa re do - - -
void setup(){
  pinMode(13,OUTPUT);
}
void loop(){
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);   playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);       delay(pause);        delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);   playNote(freqs[2]);
  playNote(freqs[2]);  playNote(freqs[1]);  playNote(freqs[1]);   playNote(freqs[2]);
  playNote(freqs[1]);  delay(pause);       playNote(freqs[4]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[2]);  playNote(freqs[2]);   delay(pause);
  playNote(freqs[2]);  playNote(freqs[4]);  playNote(freqs[0]);   playNote(freqs[1]);
  playNote(freqs[2]);  delay(pause);       delay(pause);        delay(pause);
  playNote(freqs[3]);  playNote(freqs[3]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[3]);  playNote(freqs[2]);  playNote(freqs[2]);   playNote(freqs[2]);
  playNote(freqs[4]);  playNote(freqs[4]);  playNote(freqs[3]);   playNote(freqs[3]);
  playNote(freqs[0]);  delay(pause);       delay(pause);        delay(pause);
}
```

Arduino-
Same Signal Multiple Interpretations



In the same setting if you connect an LED parallel to Piezo, you can see how the same signal can be interpreted differently using a different output device that accept the same type of signals(in this case digital signal)



Vibration Motor

: A vibration motor! This itty-bitty, shaftless vibratory motor is perfect for non-audible indicators. Use in any number of applications to indicate to the wearer when a status has changed. All moving parts are protected within the housing. With a 2-3.6V operating range, these units shake crazily at 3V. Once anchored to a PCB or within a pocket, the unit vibrates softly but noticeably. This high quality unit comes with a 3M adhesive backing and reinforced connection wires.

http://www.sparkfun.com/products/8449

MigaOne Linear Actuator and Muscle Wires

MigaOne is a unique electric actuator based on Shape Memory Alloy (SMA) wire, also known as muscle wire. Exposing the MigaOne to 10V and 2.7A for half a second will cause the wires to heat up, contract, and push what ever is attached to the arm with 2.5 lbs. (11N) of force. Surprisingly strong for such a slim actuator. This is a very new type of actuator with many interesting potential applications. Checkout the datasheet for more ideas and information.

http://www.sparkfun.com/products/8751
http://www.musclewires.com/Products.php

Firgelli Linear Actuator
They come in various shapes and sizes and powers. They are able to initiate straight locomotion.

http://www.robotshop.com/

THE C PROGRAMMING LANGUAGE

Flexible Heater
They come in various shapes and sizes and powers. They are able to regulate temperature.

http://www.omega.com/

Solenoid Water Valves
Allow to control flow of fluids electronically.

iklimnet.com
Honeywell.com
jelpc-pneumatic.com

Hagen-Laguna Ultrasonic Fog Generator
Generates mist

MarineDepot.com

THE C PROGRAMMING LANGUAGE

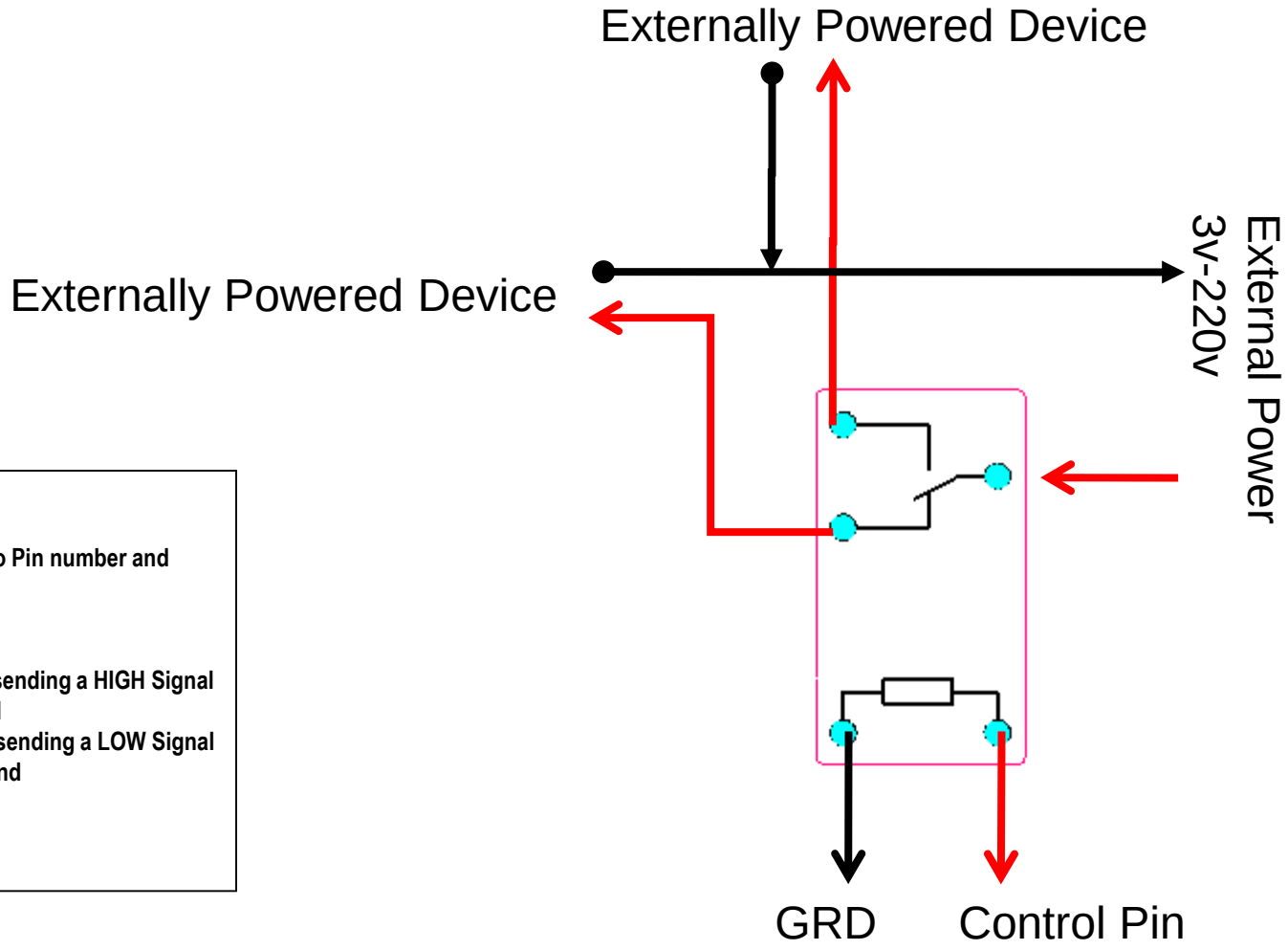
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Single/Double/Tri Color LED Matrix
7 segment LED Digital Display
LED Strips
RGB LEDs

Sparkfun.com
http://www.jameco.com/
www.ledsupply.com/
http://www.oznium.com/led-strip-flat-head

Arduino- Digital Output –

Controling any Electrical Device with any power needs using a relay



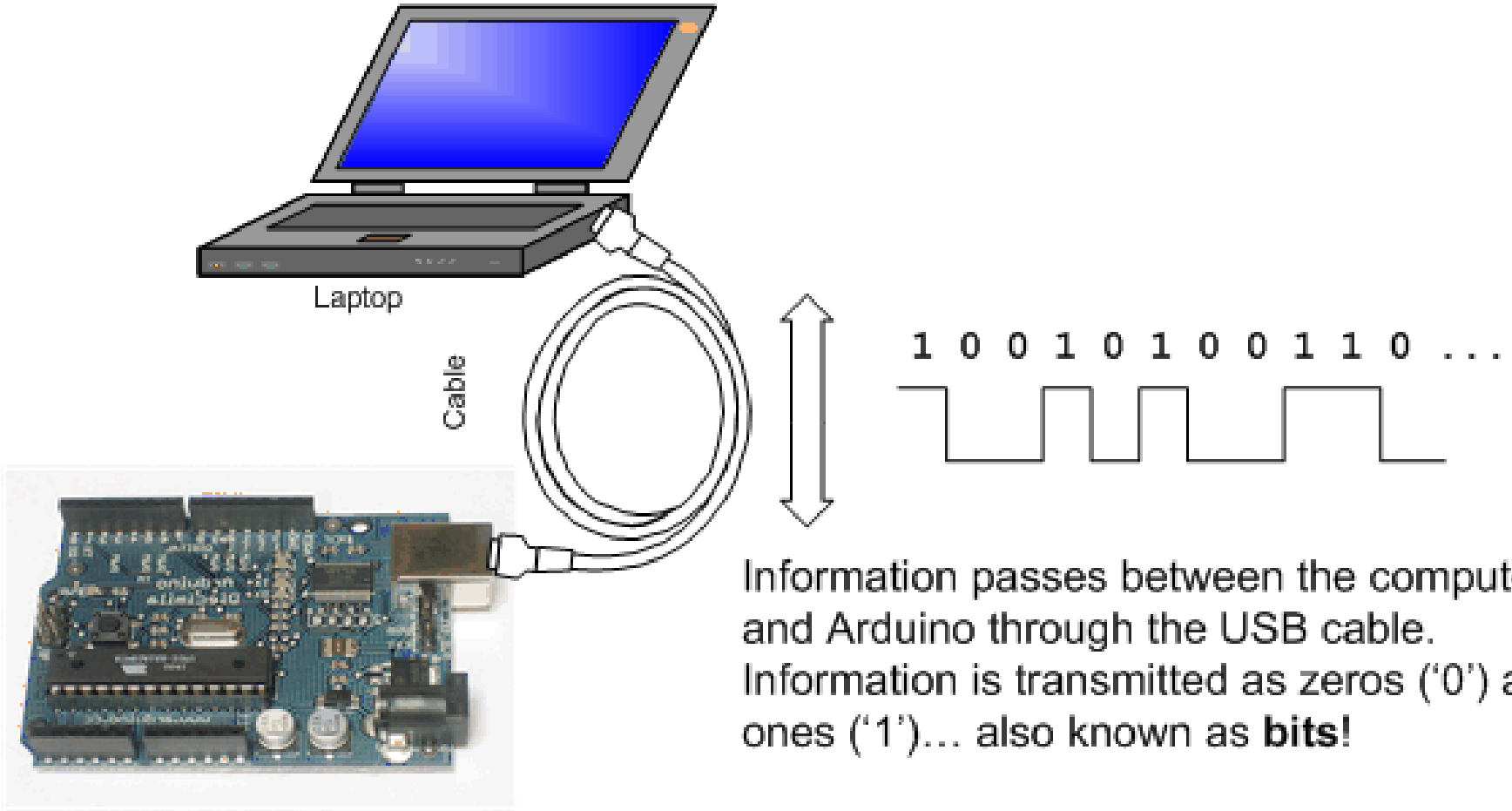

```
// Connect to Pin 13 and Ground
void setup(){
pinMode(13, OUTPUT); // Specify Arduino Pin number and
output/input mode
}
void loop(){
digitalWrite(13, HIGH); // Turn on Pin 13 sending a HIGH Signal
delay(1000);          // Wait for one second
digitalWrite(13, LOW);  // Turn off Pin 13 sending a LOW Signal
delay(3000);          // Wait for Three second
}
```

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



Information passes between the computer and Arduino through the USB cable. Information is transmitted as zeros ('0') and ones ('1')… also known as **bits!**

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

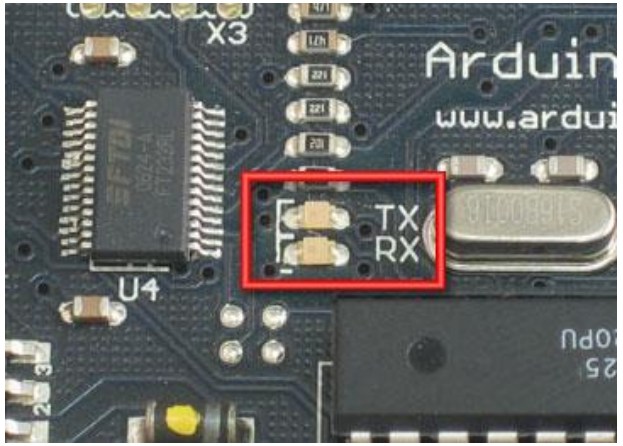
Serial Communications

- "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 <u>single data wire</u> 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

todbot.com/blog/bionicarduino

Open the Serial Monitor and Upload the Program

THE C PROGRAMMING LANGUAGE

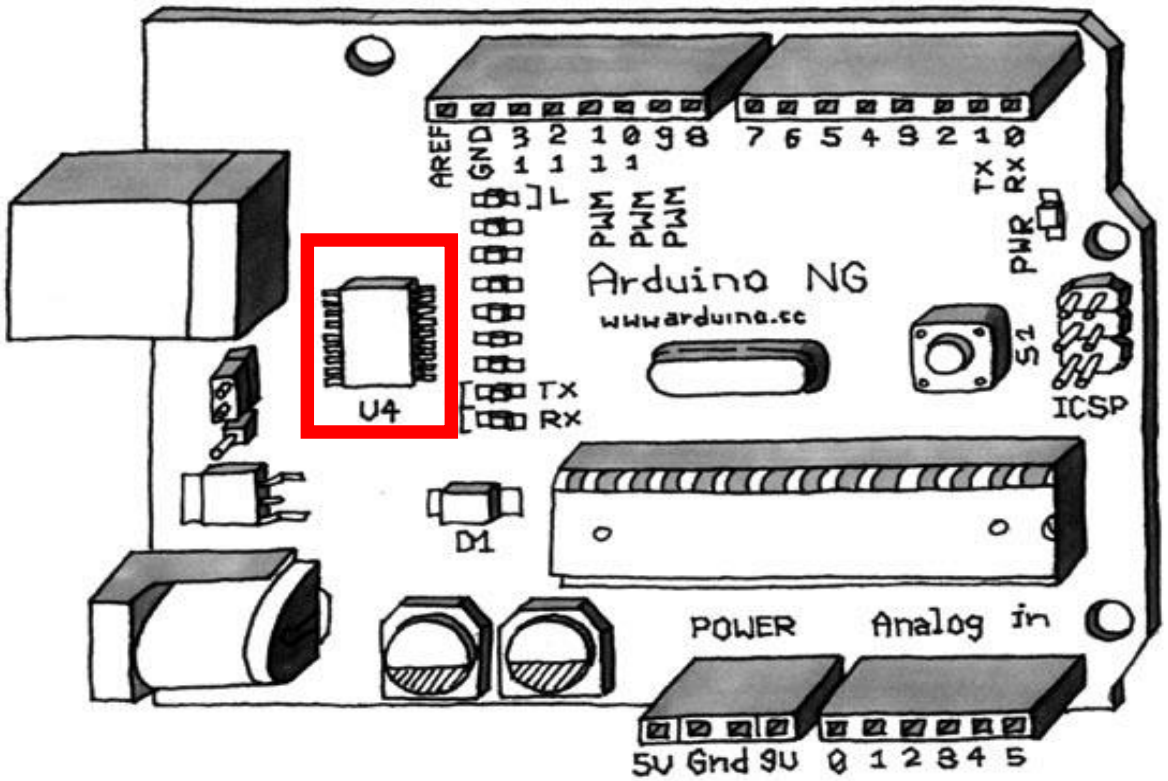
江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

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):

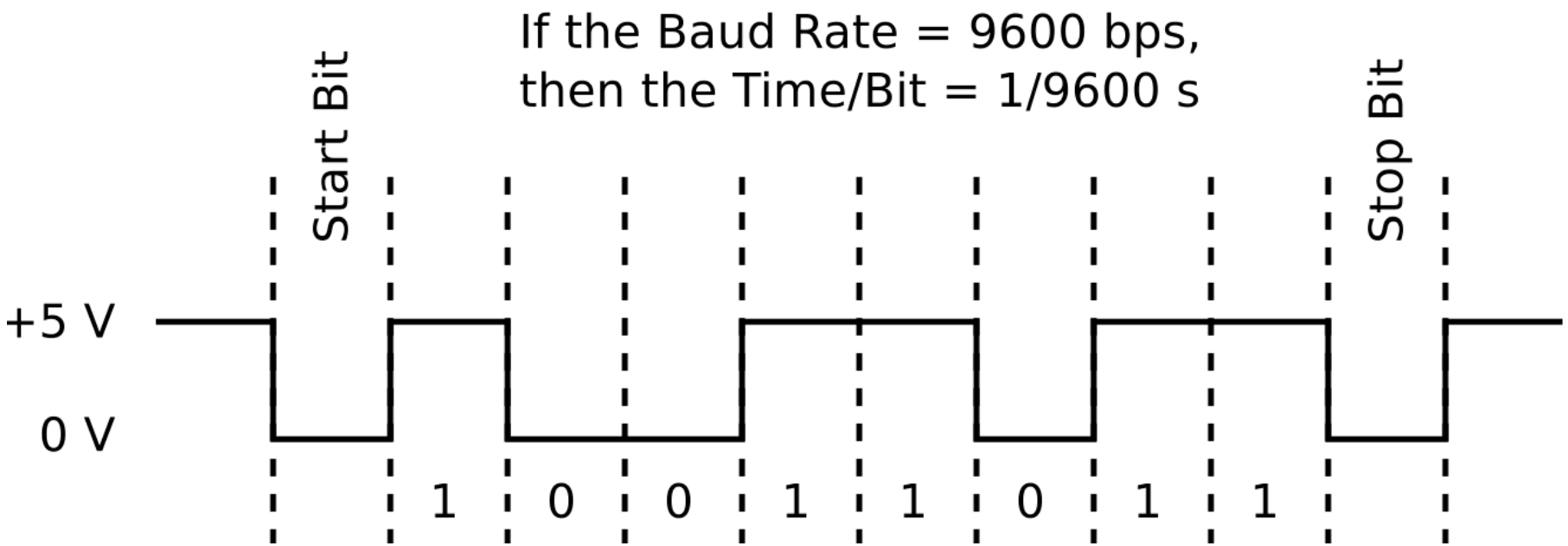


Image from http://www.fiz-ix.com/2013/02/introduction-to-arduino-serial-communication/

USB Protocol

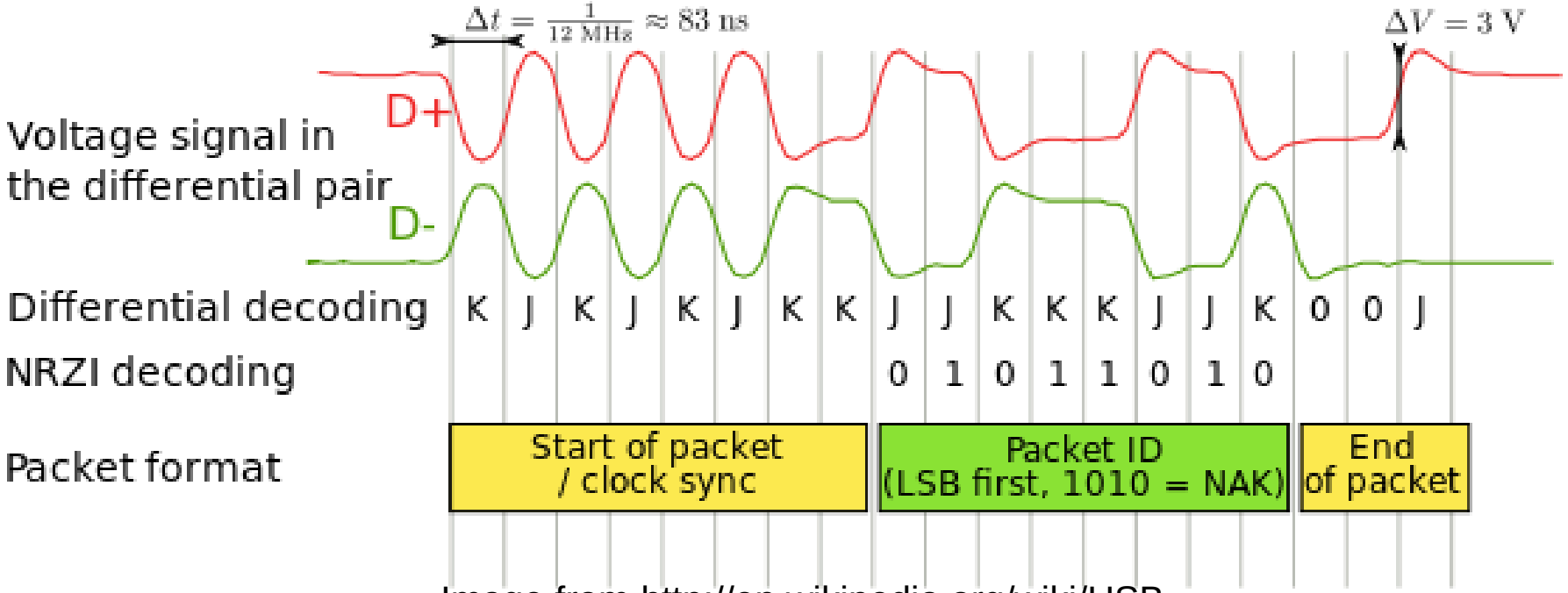


Image from http://en.wikipedia.org/wiki/USB

- Much more complicated

In-class Exercise 3: Serial Communication

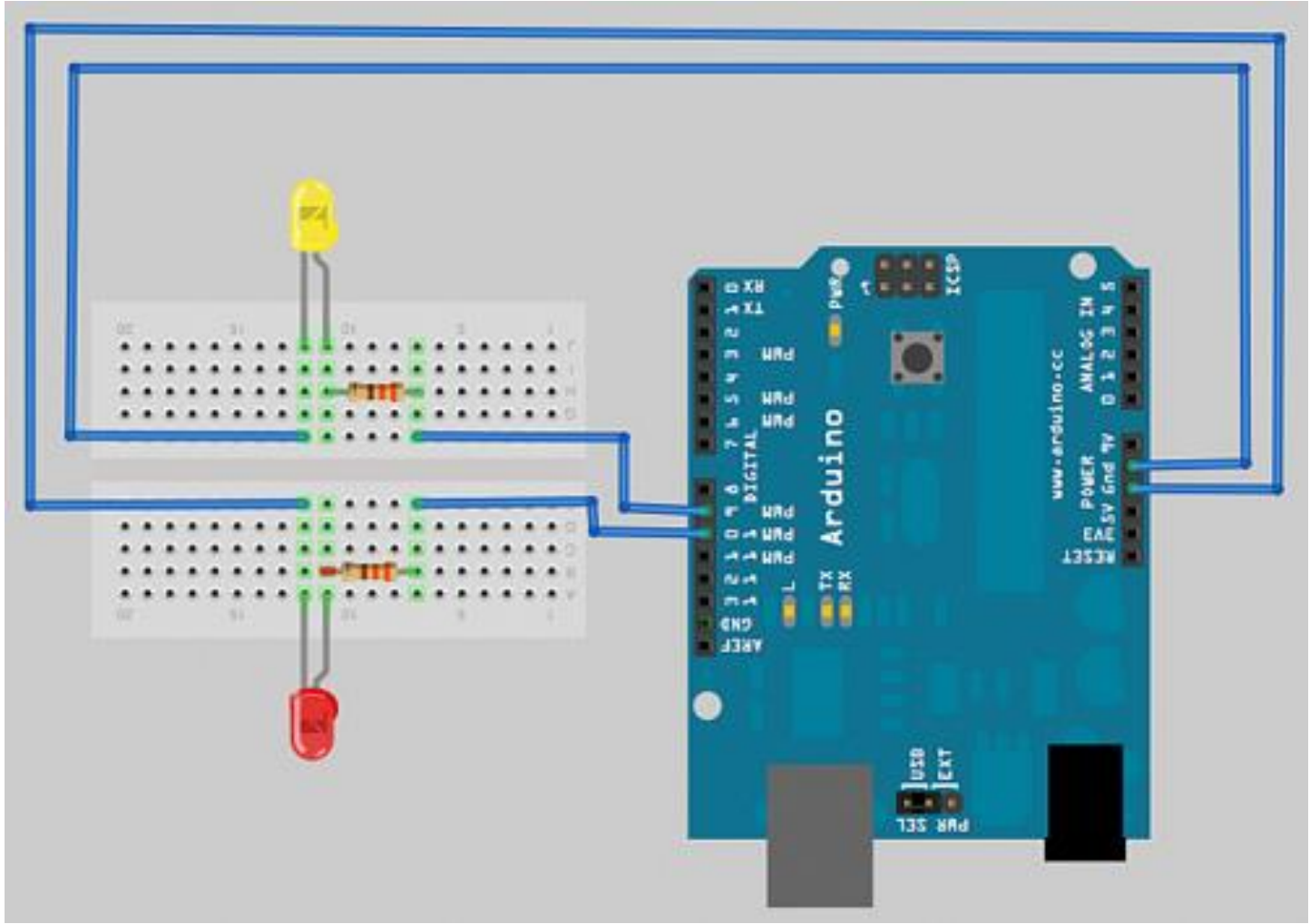
Modify your program from in-class exercise 2-part 2 to control the intensity of the LED attached to pin 9 based on keyboard input.

Use the Serial.parseInt() method to read numeric keyboard input as an integer.

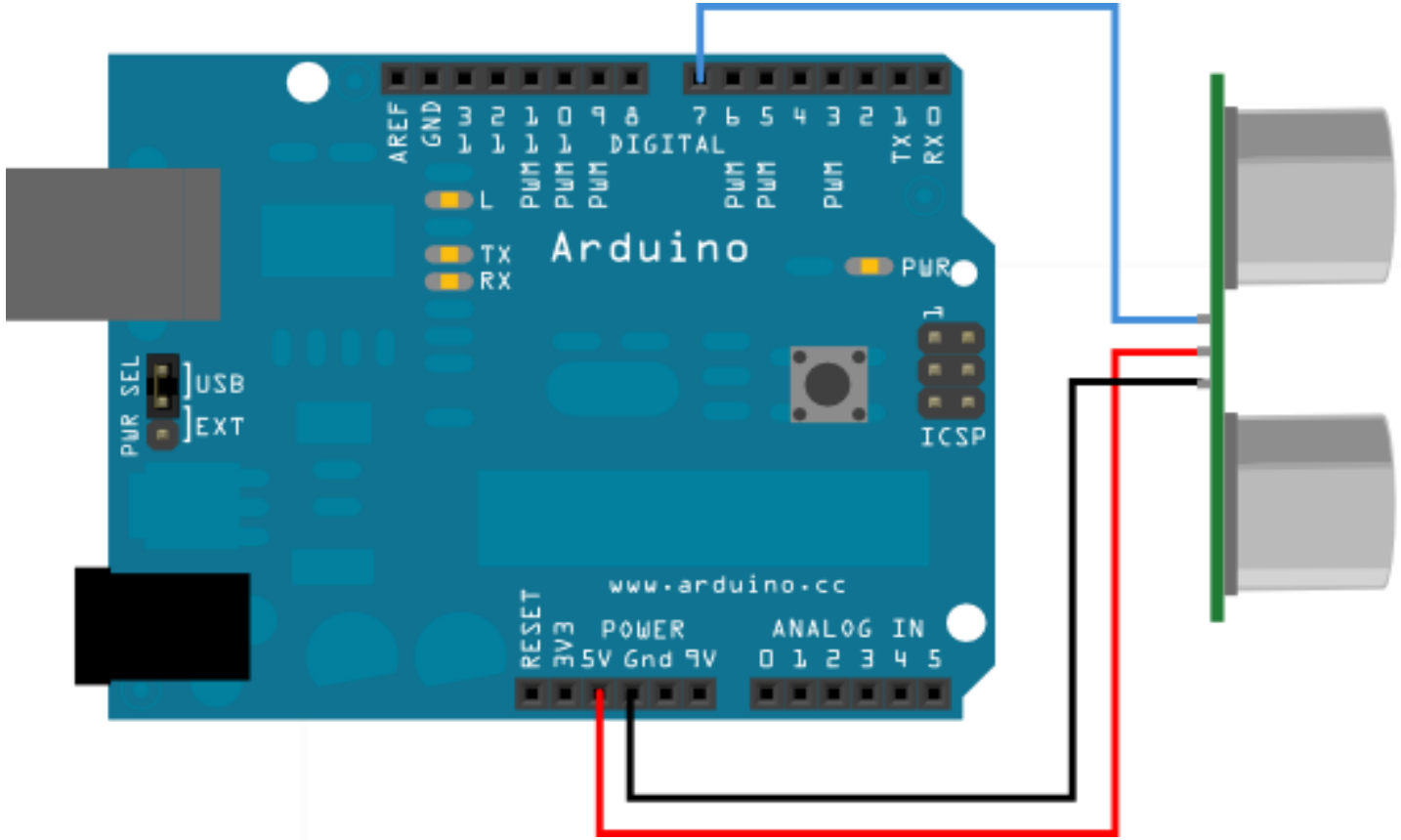
An input of 9 should produce full intensity and an input of 0 should turn the LED off.

Arduino Code Basics

THE C PROGRAMMING LANGUAGE



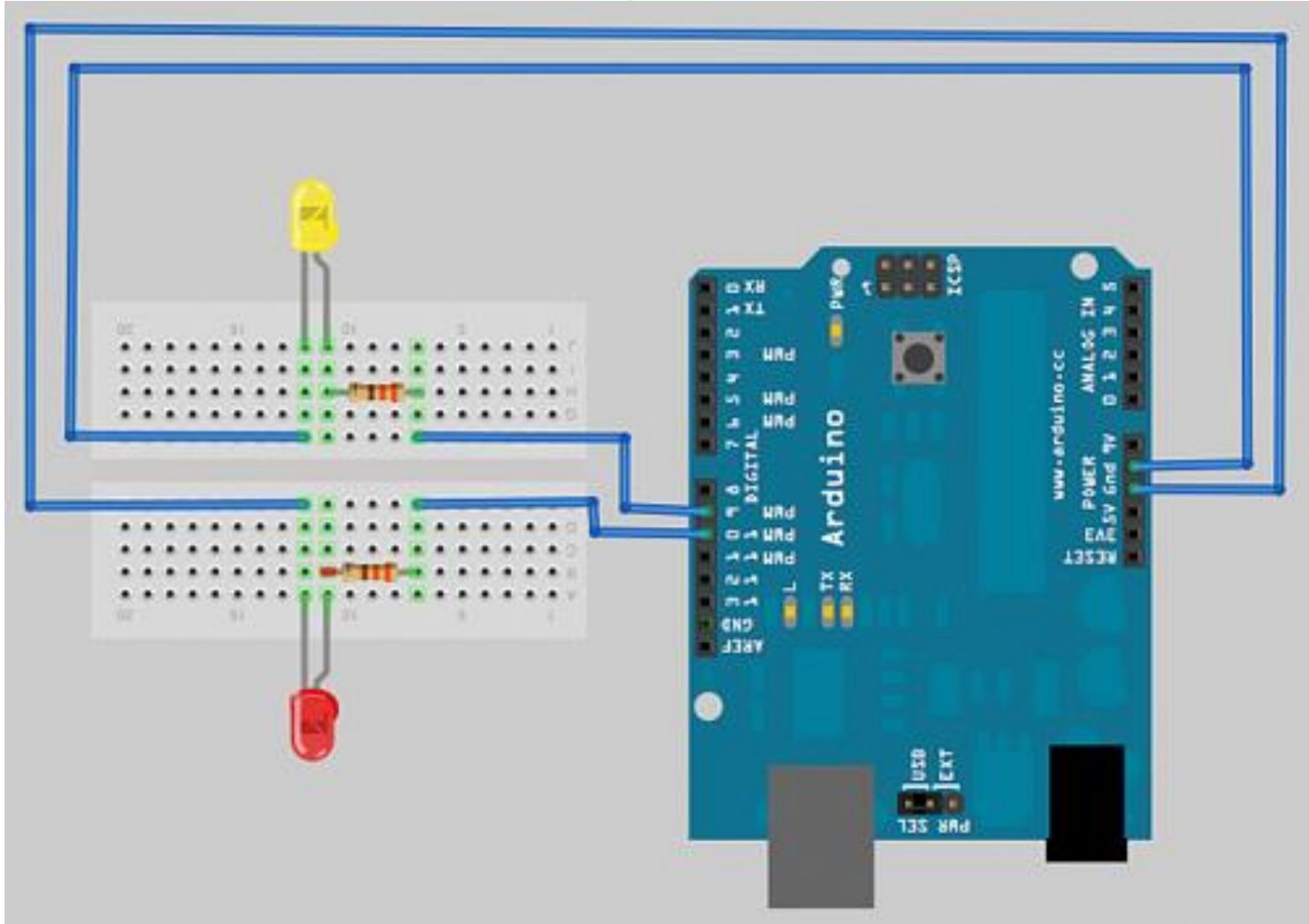
Arduino Code Basics



THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Setup - Adding an LED



THE C PROGRAMMING LANGUAGE

江西理工大學
JIANGXI UNIVERSITY OF SCIENCE AND TECHNOLOGY