Arduino

1

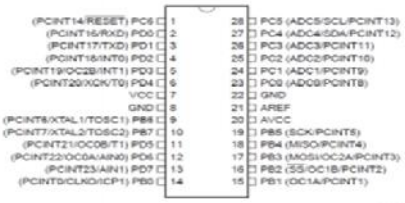
What is a Microcontroller (µC, MCU)

- Computer on a single integrated chip
  - Processor (CPU)
  - Memory (RAM / ROM / Flash)
  - I/O ports (USB, I2C, SPI, ADC)
- Common microcontroller families:
  - Intel: 4004, 8008, etc.
  - Atmel: AT and AVR
  - Microchip: PIC
  - ARM: (multiple manufacturers)
- Used in:
  - Cellphones,
  - Toys
  - Household appliances
  - Cars
  - Cameras

2

The ATmega328P Microcontroller
(used by the Arduino)

- AVR 8-bit RISC architecture
- Available in DIP package
- Up to 20 MHz clock
- 32kB flash memory
- 1 kB SRAM
- 23 programmable I/O channels
- Six 10-bit ADC inputs
- Three timers/counters
- Six PWM outputs



ARDUINO

3

So what *is* Arduino?

It's a ***movement***, not a microcontroller:

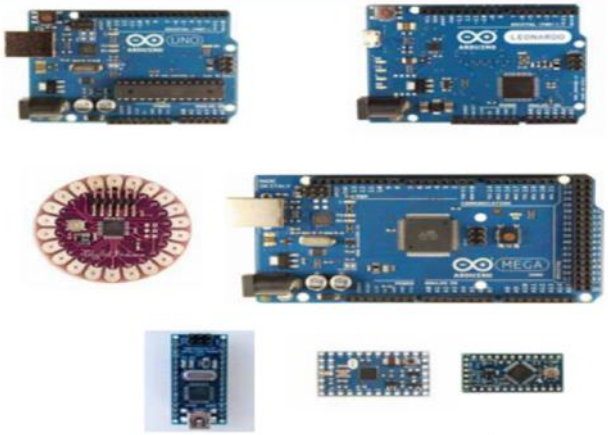
- Founded by Massimo Banzi and David Cuartielles in 2005
- Based on "Wiring Platform", which dates to 2003
- Open-source hardware platform
- Open source development environment
  - Easy-to learn language and libraries (based on Wiring language)
  - Integrated development environment (based on Processing programming environment)
  - Available for Windows / Mac / Linux

ARDUINO

4

The Many Flavors of Arduino

- Arduino Uno
- Arduino Leonardo
- Arduino LilyPad
- Arduino Mega
- Arduino Nano
- Arduino Mini
- Arduino Mini Pro
- Arduino BT

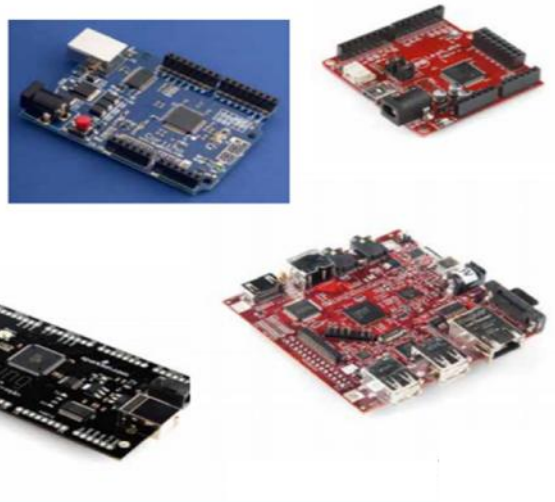


ARDUINO

5

Arduino-like Systems

- Cortino (ARM)
- Xduino (ARM)
- LeafLabs Maple (ARM)
- BeagleBoard (Linux)
- Wiring Board (Arduino predecessor)

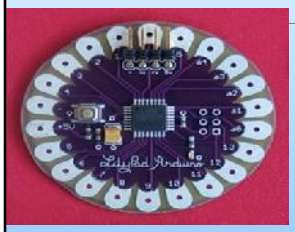


ARDUINO

6

Other Hardware Choices-Boards

Arduino BT
The Arduino BT is an Arduino board with built-in bluetooth module, allowing for wireless communication.



LilyPad Arduino
The LilyPad Arduino is a microcontroller board designed for wearables and e-textiles. It can be sewn to fabric and similarly mounted power supplies, sensors and actuators with conductive thread.

Arduino Nano
Arduino Nano is a surface mount breadboard embedded version with integrated USB. It is a smallest, complete, and breadboard friendly. It has everything that Diecimila has (electrically) with more analog input pins and onboard +5V AREF jumper.

7

Other Hardware Choices-Sheilds

Xbee Shield
The Xbee shield allows an Arduino board to communicate wirelessly using Zigbee. The module can communicate up to 100 feet indoors or 300 feet outdoors (with line-of-sight). It can be used as a serial/usb replacement or you can put it into a command mode and configure it for a variety of broadcast and mesh networking options.
The Xbee shield was created in collaboration with Libelium, who developed it for use in their SquidBee motes (used for creating sensor networks).
Adafruit Servo/Stepper/DC Motor shield
A shield that can control 2 hobby servos and up to 2 unipolar/bipolar stepper motors or 4 bi-directional DC motors.
Battery Shield
A shield from Liquidware that connects to the back of the Arduino, with a USB-rechargable lithium ion battery that can power an Arduino for 14-28 hours depending on the circuit
Liquidware TouchShield
OLED touch screen shield.
Adafruit Wave shield
Plays any size 22KHz audio files from an SD memory card for music, effects and interactive sound art
Adafruit GPS & Datalogging shield
Connects up a GPS module and can log location, time/date as well as sensor data to an SD memory flash card.
Adafruit XPort/Ethernet shield
Allows use of an XPort module for connecting to the Internet as a client or server.

8

Other Hardware Choices-Sheilds

Adafruit GPS & Datalogging shield
Connects up a GPS module and can log location, time/date as well as sensor data to an SD memory flash card.

Adafruit XPort/Ethernet shield
Allows use of an XPort module for connecting to the Internet as a client or server.

http://ladyada.net

Other Hardware Choices-Sheilds

Liquidware TouchShield
OLED touch screen shield.

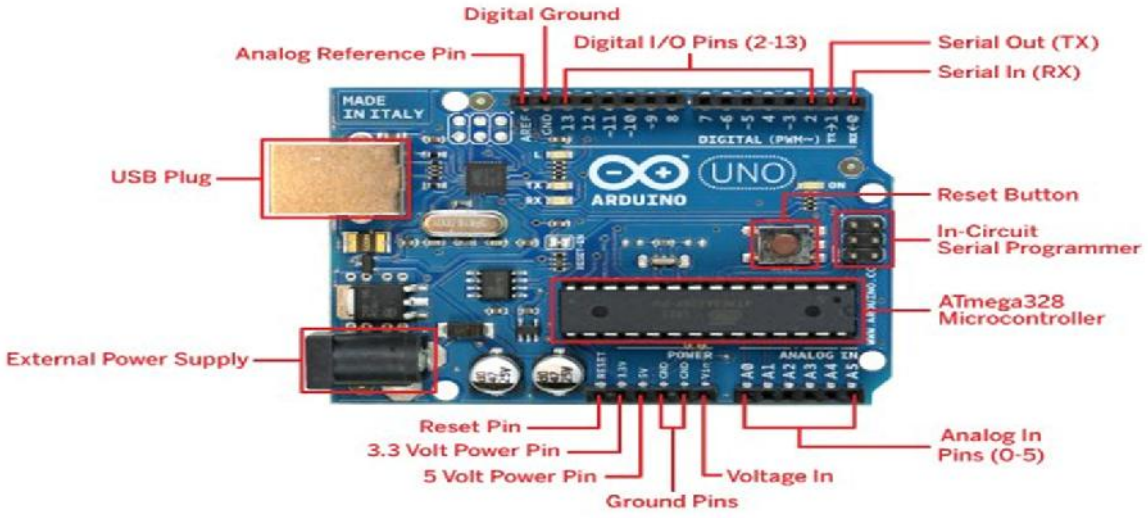
http://www.liquidware.com

Adafruit Servo/Stepper/DC Motor shield
A shield that can control 2 hobby servos and up to 2 unipolar/bipolar stepper motors or 4 bi-directional DC motors.

http://ladyada.net

10

Getting to know the Arduino: Electrical Inputs and Outputs



11

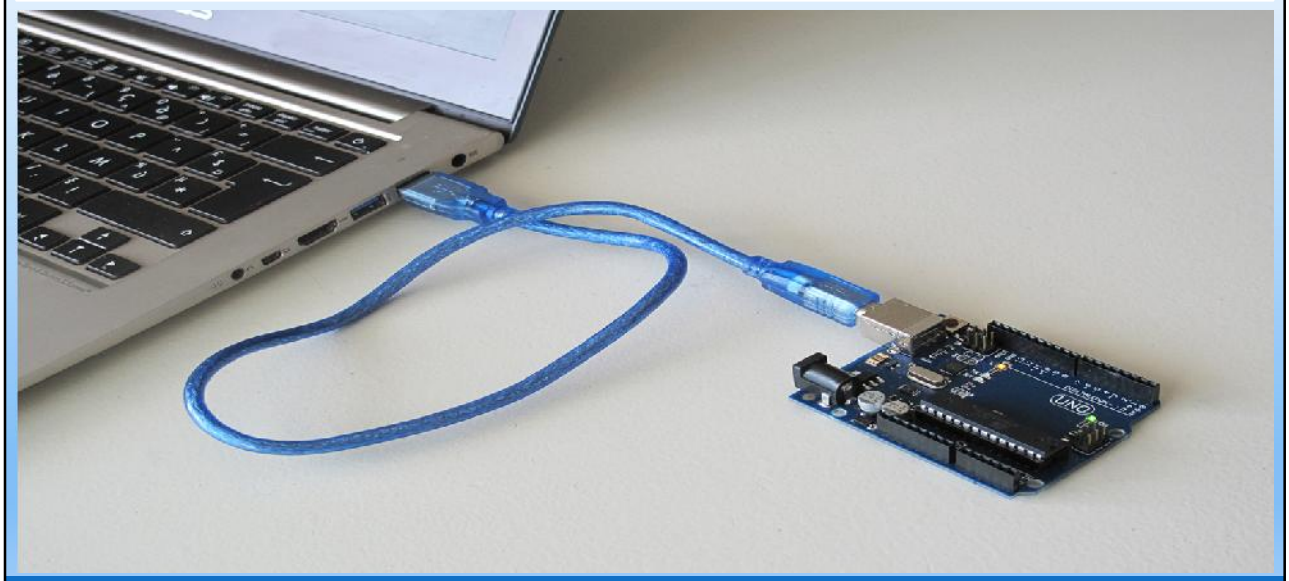
Download and Install

- Download Arduino compiler and development environment from: http://arduino.cc/en/Main/Software
- Current version: 1.0.1
- Available for:
  - Windows
  - MacOX
  - Linux
- No installer needed… just unzip to a convenient location
- ***Before running Arduino***, plug in your board using USB cable (external power is not necessary)
- When USB device is not recognized, navigate to and select the appopriate driver from the installation directory
- Run Arduino

ARDUINO

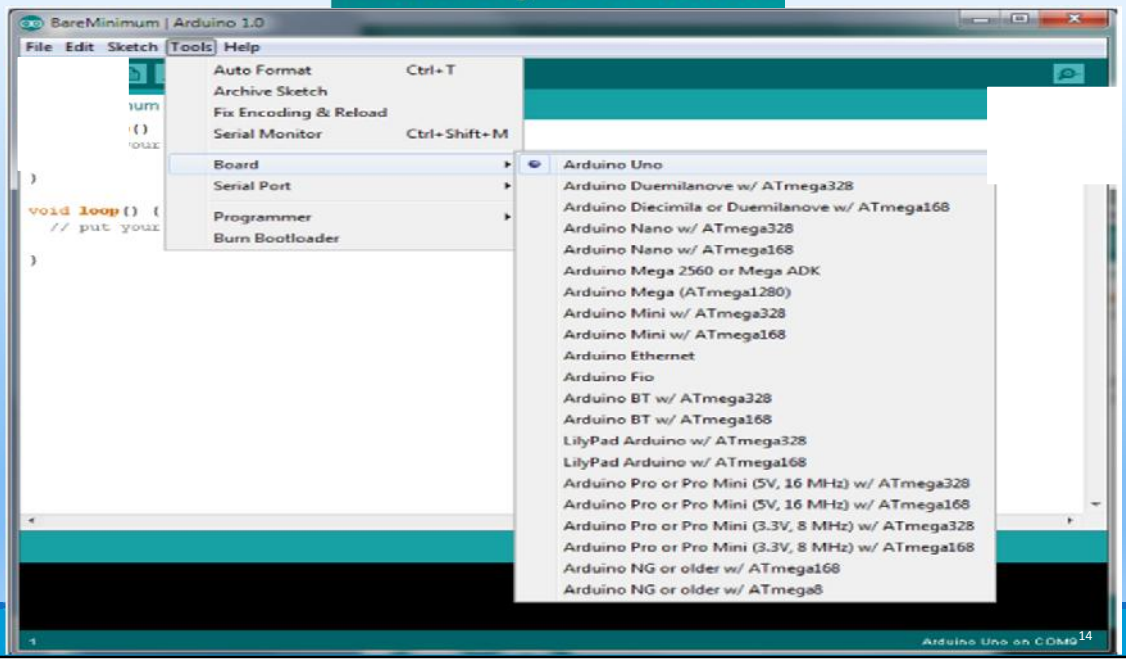
12

Connecting ARDUINO Uno with PC's USB Port



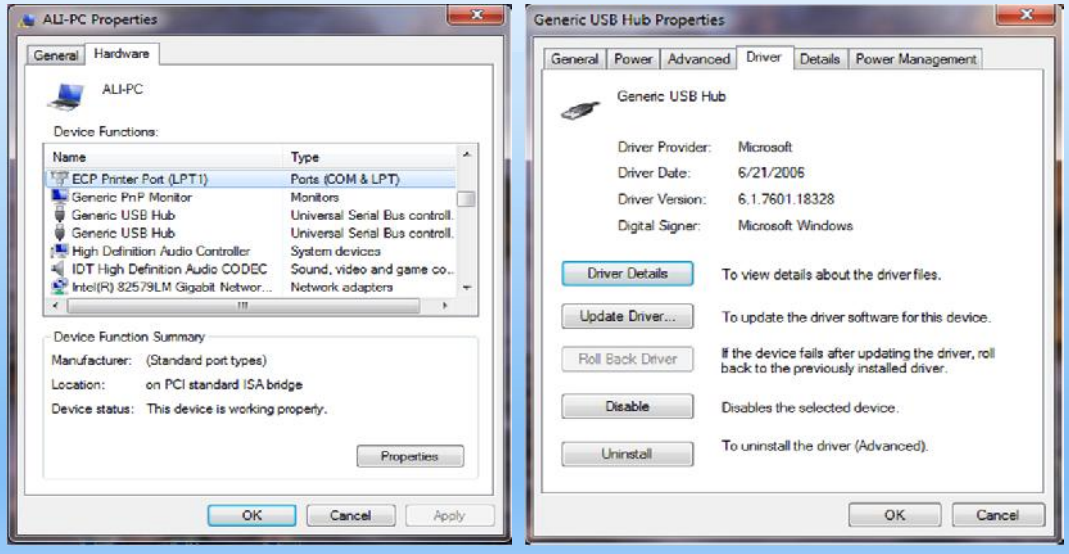
13

Select your Board



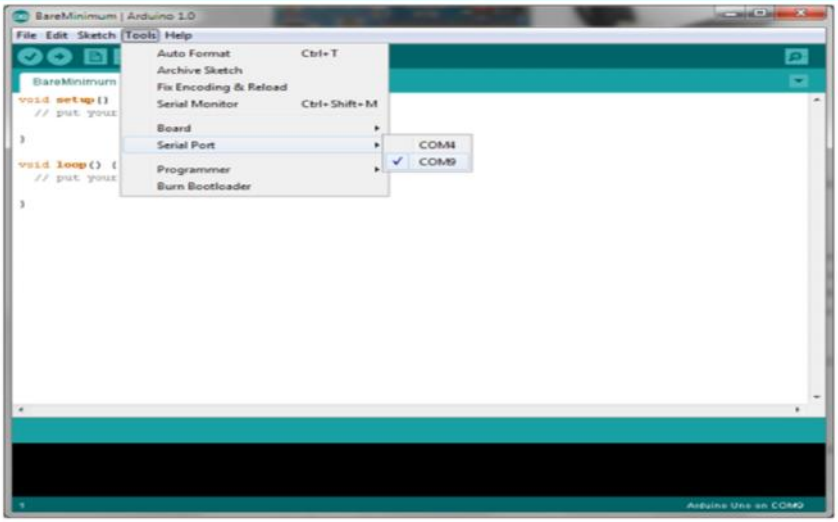
14

Selecting and Updating the Drivers if Needed



15

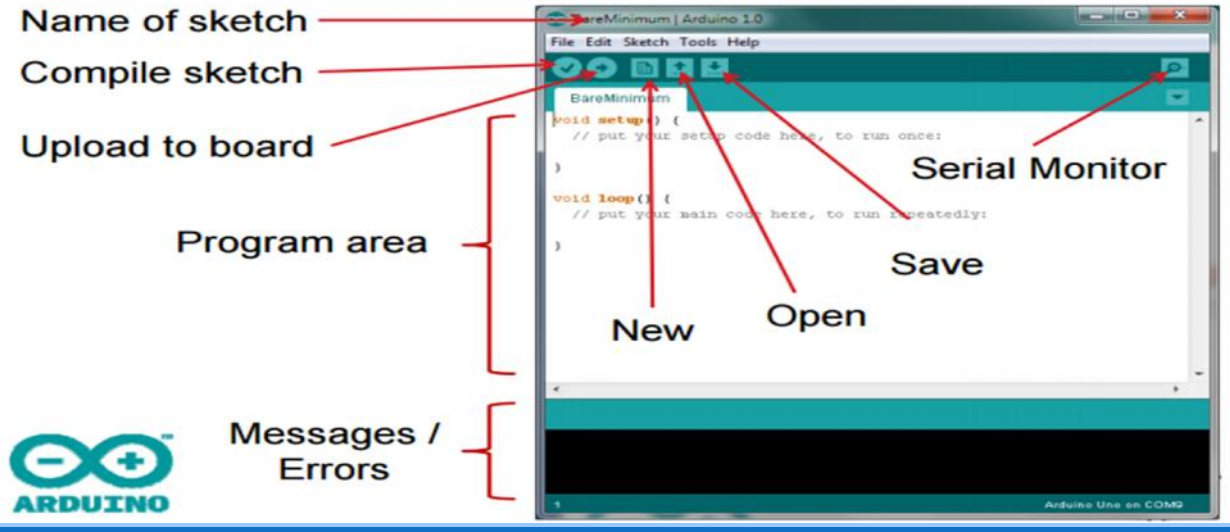
Select Serial Port



ARDUINO

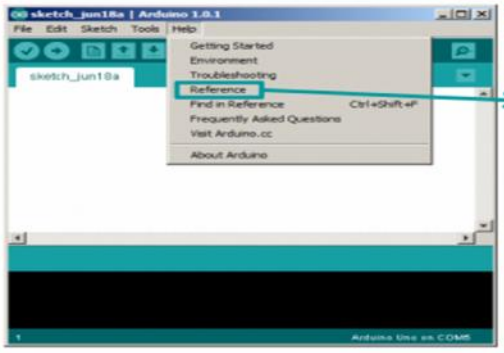
16

Using the Arduino IDE

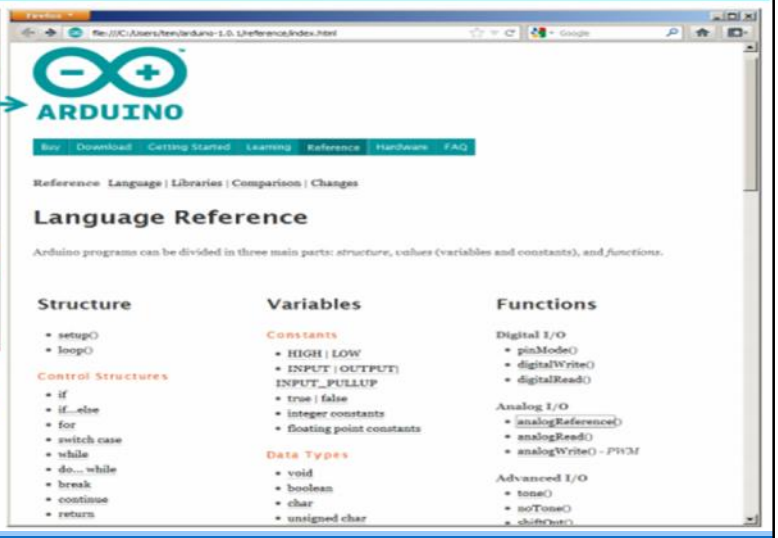


17

Arduino Reference



Arduino Reference is installed locally or available online at http://arduino.cc/



18

C++ Programming Structure and Syntax for ARDUINO

- Structure:

1. The basic programming for ARDUINO revolves around 2 simple parts.
2. These 2 parts are the necessary ingredients for your program to work.
3. Setup function in the figure run only once and is used for the initialization of Serial port and I/O's.
4. Loop function is used to trigger the conditions that run continuously like reading input or sending the data to output port if conditions are meet.

```
void setup()
{
  statements;
}

void loop()
{
  statements;
}
```

19

Basic Instruction Set for Digital I/O's and Conditional Loops

Arduino Programming Basics

Command	Description
`pinMode(n,INPUT)`	Set pin *n* to act as an input. One-time command at top of program.
`pinMode(n,OUTPUT)`	Set pin *n* to act as an output
`digitalWrite(n,HIGH)`	Set pin *n* to 5V
`digitalWrite(n,LOW)`	Set pin *n* to 0V
`delay(x)`	Pause program for *x* millisec, *x* = 0 to 65,535
`tone(n,f,d)`	Play tone of frequency *f* Hz for *d* millisec on speaker attached to pin *n*
`for()`	Loop. Example: for (i=0;i<3;i++){} Do the instructions enclosed by {} three times
`if (expr) {}`	Conditional branch. If *expr* true, do instructions enclosed by {}
`while (expr) {}`	While *expr* is true, repeat instructions in {} indefinitely

20

Acceptable Data Type

- Byte: 8-bits, ranges from 0 to 255. `byte someVariable = 180;`
- Integer (int): 16-bits, ranges from -32768 to 32767. `int someVariable = 1500;`
- Unsigned int: 16-bits, ranges from 0 to 65535.
- long: 32-bits, ranges from -2,147,483,648 to 2,147,483,647. `long someVariable = 90000;`
- float: 32-bits, ranges from 3.4028235E+38 to -3.4028235E+38

21

arithmetic

Arithmetic operators include addition, subtraction, multiplication, and division. They return the sum, difference, product, or quotient (respectively) of two operands.

```
y = y + 3;
x = x - 7;
i = j * 6;
r = r / 5;
```

comparison operators

Comparisons of one variable or constant against another are often used in if statements to test if a specified condition is true. In the examples found on the following pages, ?? is used to indicate any of the following conditions:

```
x == y    // x is equal to y
x != y    // x is not equal to y
x <  y    // x is less than y
x >  y    // x is greater than y
x <= y    // x is less than or equal to y
x >= y    // x is greater than or equal to y
```

compound assignments

Compound assignments combine an arithmetic operation with a variable assignment. These are commonly found in for loops as described later. The most common compound assignments include:

```
x ++      // same as x = x + 1, or increments x by +1
x --      // same as x = x - 1, or decrements x by -1
x += y    // same as x = x + y, or increments x by +y
x -= y    // same as x = x - y, or decrements x by -y
x *= y    // same as x = x * y, or multiplies x by y
x /= y    // same as x = x / y, or divides x by y
```

logical operators

Logical operators are usually a way to compare two expressions and return a TRUE or FALSE depending on the operator. There are three logical operators, AND, OR, and NOT, that are often used in if statements:

Logical AND:

```
if (x > 0 && x < 5)     // true only if both
                        // expressions are true
```

Logical OR:

```
if (x > 0 || y > 0)     // true if either
                        // expression is true
```

Logical NOT:

```
if (!x > 0)             // true only if
                        // expression is false
```

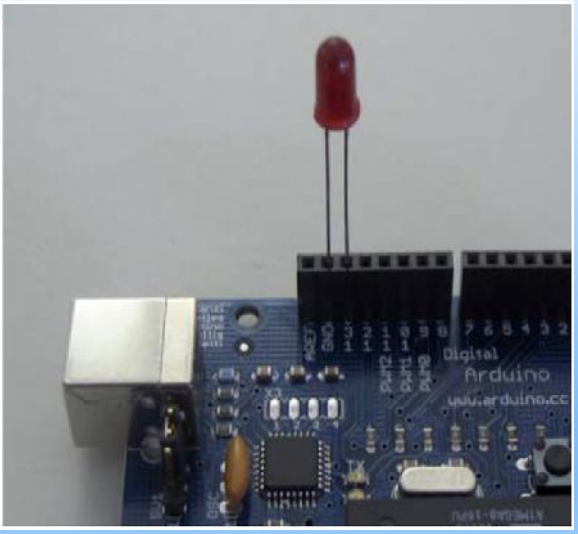
22

Example of LED blinking with 1sec delay

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

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

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



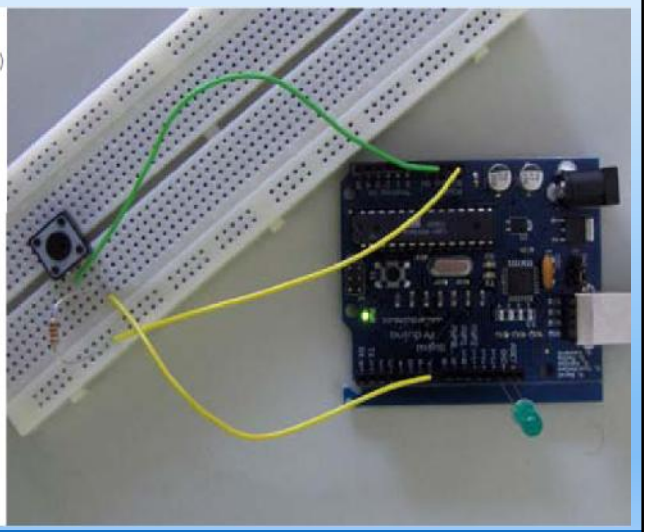
23

Button & LED Example

```
int ledPin = 13; // choose the pin for the LED
int inPin = 2;   // choose the input pin (for a pushbutton)
int val = 0;     // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT);  // declare LED as output
  pinMode(inPin, INPUT);    // declare pushbutton as input
}

void loop(){
  val = digitalRead(inPin);  // read input value
  if (val == HIGH) {         // check if the input is HIGH
    digitalWrite(ledPin, LOW);  // turn LED OFF
  } else {
    digitalWrite(ledPin, HIGH);  // turn LED ON
  }
}
```



24

Knight Rider Example

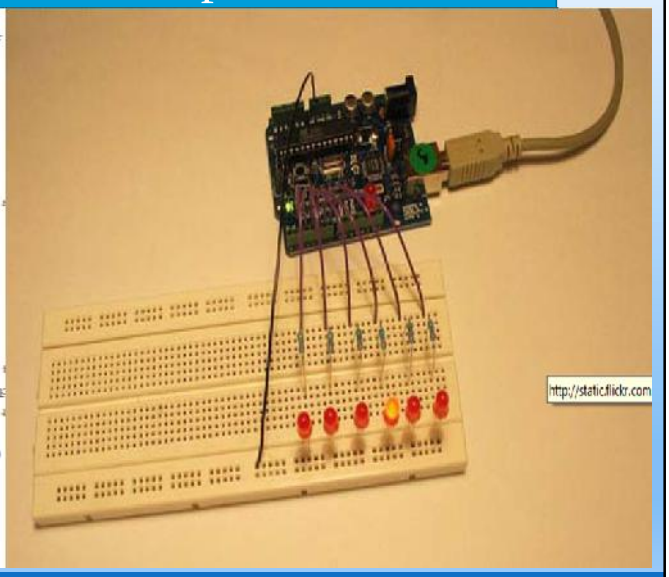
```
int timer = 100;
int pins[] = { 2, 3, 4, 5, 6, 7 };
int num_pins = 6;

void setup()
{
  int i;

  for (i = 0; i < num_pins; i++)
    pinMode(pins[i], OUTPUT);
}

void loop()
{
  int i;

  for (i = 0; i < num_pins; i++) {
    digitalWrite(pins[i], HIGH);
    delay(timer);
    digitalWrite(pins[i], LOW);
  }
  for (i = num_pins - 1; i >= 0; i--)
    digitalWrite(pins[i], HIGH);
    delay(timer);
    digitalWrite(pins[i], LOW);
  }
}
```



25

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

26

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]);
    }
  }
}
```

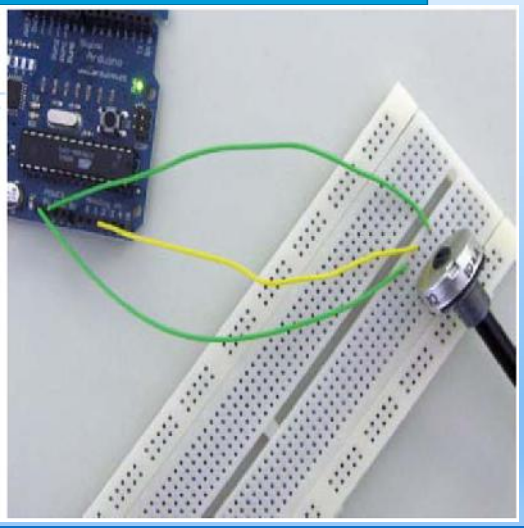
27

LED's Blinking rate using Analog Input (Potentiometer)

```
int potPin = 2;    // select the input pin for the potentiometer
int ledPin = 13;   // select the pin for the LED
int val = 0;       // variable to store the value coming from the sensor

void setup() {
  pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT
}

void loop() {
  val = analogRead(potPin);    // read the value from the sensor
  digitalWrite(ledPin, HIGH);  // turn the ledPin on
  delay(val);                  // stop the program for some time
  digitalWrite(ledPin, LOW);   // turn the ledPin off
  delay(val);                  // stop the program for some time
}
```



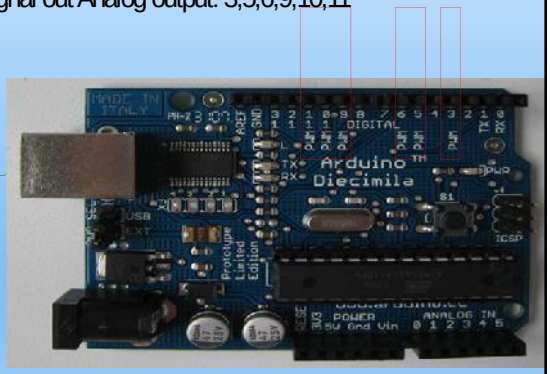
28

Arduino-Analog Output-LED

Analog Out put is defined as sending signals from one of the digital pins on the Aurduino board that range between two extremes:
0-255

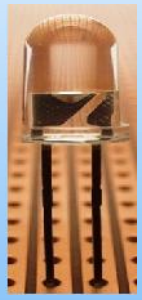
Out of 13 Digital pins on Arduino board the following pins can be used to signal out Analog output: 3,5,6,9,10,11

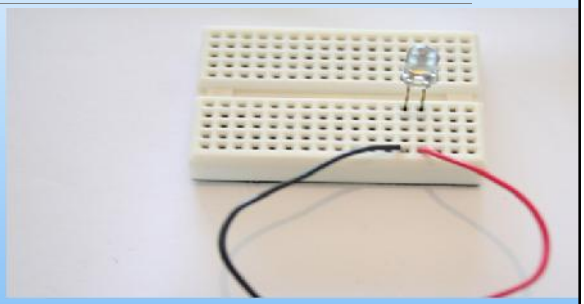
These are the pins with PWM label next to them on the board



29

Arduino-Analog Output-LED





For this exercise since we need to see the light variations , we are going to use a high intensity LED. High Intensity LEDs emit more light than normal LEDs and it is easier to detect light variations, using them.

30

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
}
```

31

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);
  }
}
```

32

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
}
}
```

33

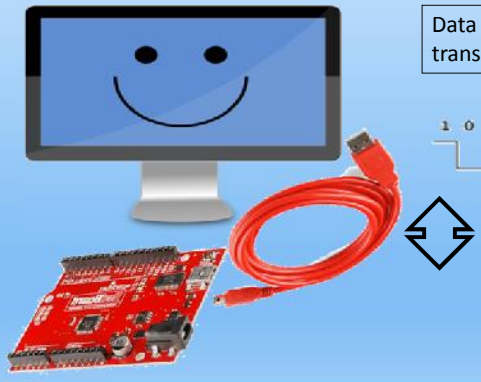
Creating a bar graph using LEDs

```
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:
}
}
}
```

34

Using Serial Communication

Method used to transfer data between two devices.

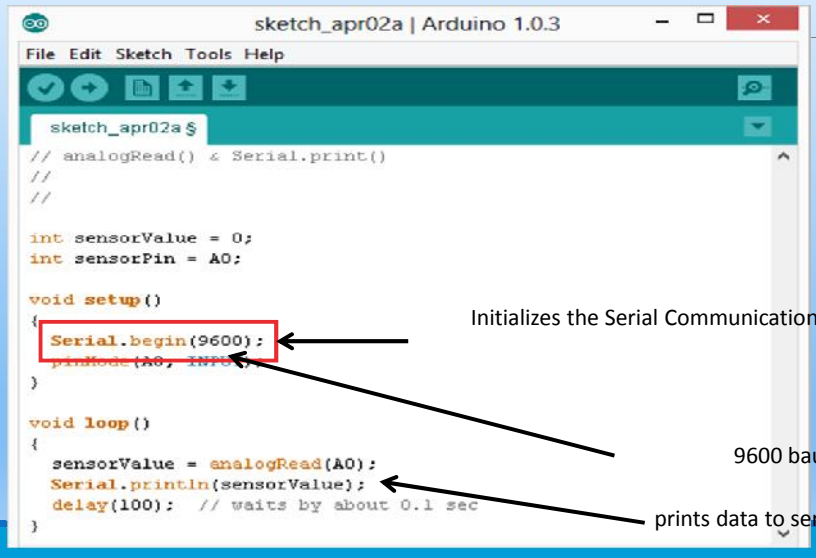


Data passes between the computer and Arduino through the USB cable. Data is transmitted as zeros (‘0’) and ones (‘1’) sequentially.

Arduino dedicates Digital I/O pin # 0 to receiving and Digital I/O pin #1 to transmit.

35

Serial Monitor & analogRead()



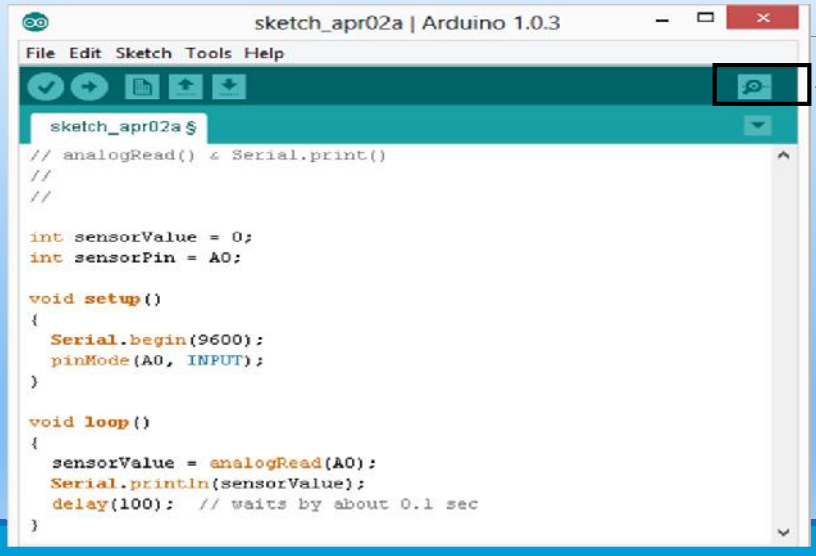
Initializes the Serial Communication

9600 baud data rate

prints data to serial bus

36

Serial Monitor & analogRead()



Opens up a Serial Terminal Window

37

Serial Communication

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

Note: When using serial communication, digital pins 0 (RX) and 1 (TX) cannot be used at the same time.

```
void setup()
{
  Serial.begin(9600);              // sets serial to 9600bps
}

void loop()
{
  Serial.println(analogRead(0)); // sends analog value
  delay(1000);                   // pauses for 1 second
}
```

38

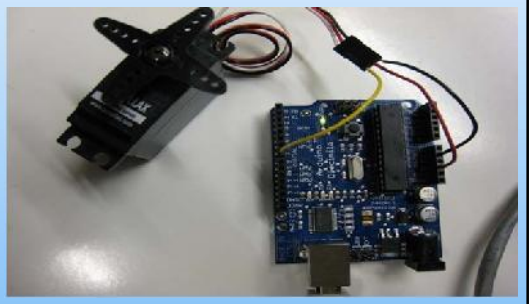
Arduino-Digital Output-Motion-Servo Motor

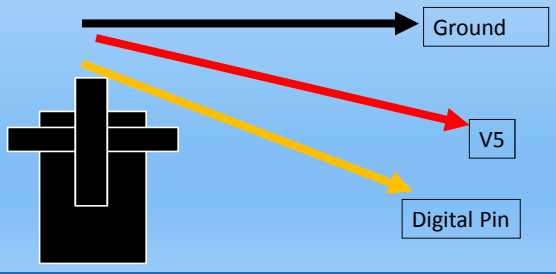
Servo Motors are electronic devices that convert digital signal to rotational movement. There are two sorts of servo motors: Standard servos that their rotation is limited to maximum of 180 degrees in each direction and Continuous Rotation Servos that can provide rotation unlimitedly in both directions

39

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.**





40

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);
}
```

41

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);
}
```

42

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);
}
}
}
```


Connecting LCDs

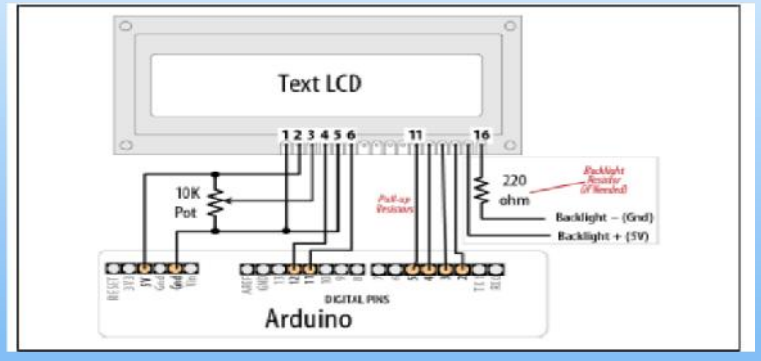


Table 11-1. LCD pin connections

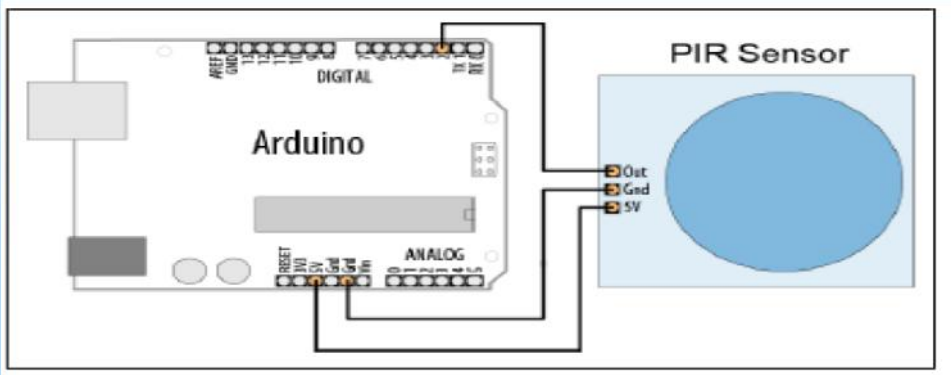
LCD pin	Function	Arduino pin
1	Gnd or 0V or Vss	Gnd
2	+5V or Vdd	5V
3	Vo or contrast	
4	RS	12
5	R/W	
6	E	11
7	D0	
8	D1	
9	D2	
10	D3	
11	D4	5
12	D5	4
13	D6	3
14	D7	2
15	A or analog	
16	K or cathode	

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.
}
```

45

Using PIR motion sensors



46

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
}
```

47

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

48

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); }
```

49

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 ;
}
```

50

Audio output

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

51

Queries

52


Thank you
53