


A decorative network diagram in the top-left corner of the slide. It consists of a series of interconnected nodes and lines. The nodes are represented by small circles, some of which are solid blue, some are solid grey, and some are hollow with a blue outline. The lines connecting them are thin and grey. The overall shape is organic and spreads out from the top-left towards the center.

Introduction to Microcontrollers - Arduino, Sensors and Programming

Katherine, Kee Wan Ting
Tech Director, Vice President
MAE Robotics Club

A decorative network diagram in the bottom-right corner of the slide. It is similar to the one in the top-left, featuring interconnected nodes and lines. The nodes are small circles, some solid blue, some solid grey, and some hollow with a blue outline. The lines are thin and grey. The shape is organic and spreads out from the bottom-right towards the center.

What is a Microcontroller

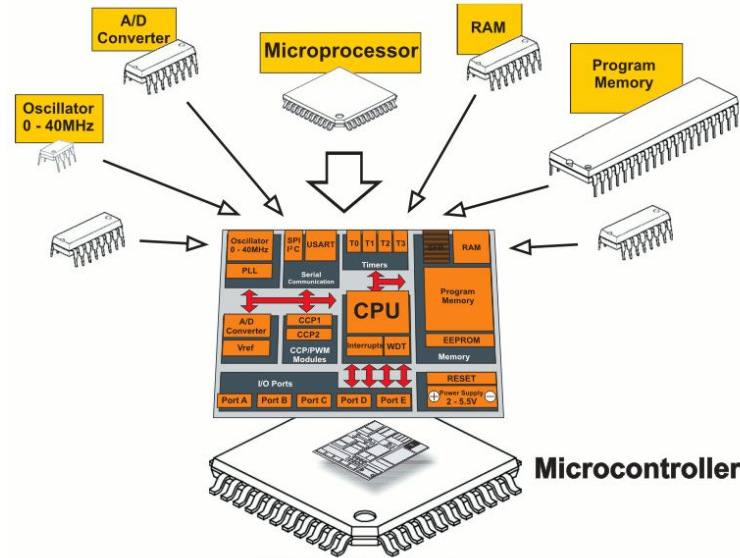
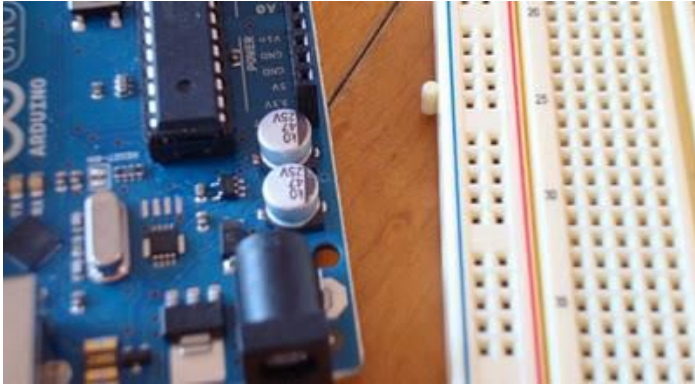


Fig. 0-1 Microcontroller versus Microprocessor

- ⊙ A small computer on a single chip containing a processor, memory, and input/output
- ⊙ Typically "embedded" inside some device that they control
- ⊙ A microcontroller is often small and low cost

What is a Development Board



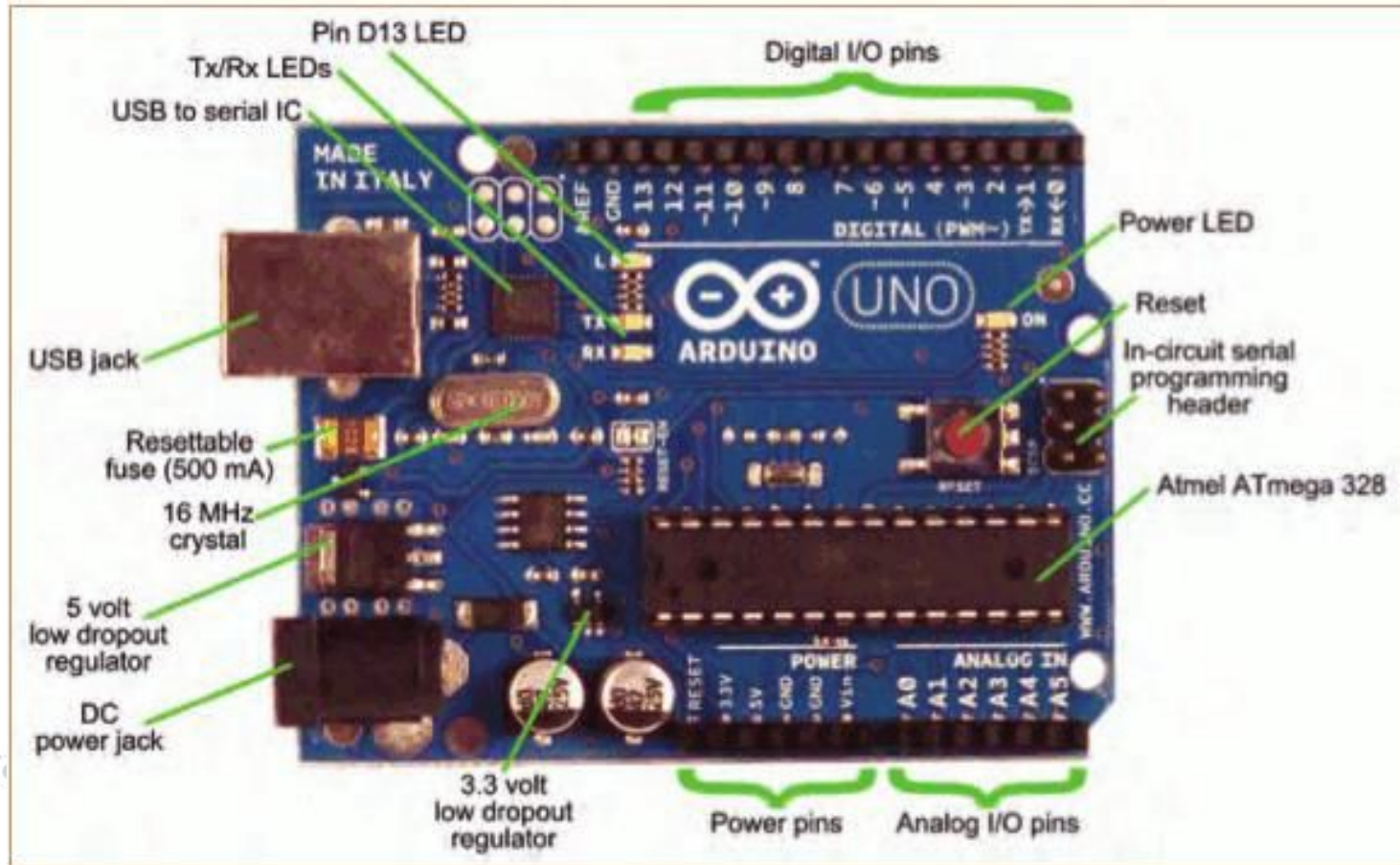
A printed circuit board designed to facilitate work with a particular microcontroller.

Typical components include:

- ⦿ power circuit
- ⦿ programming interface
- ⦿ basic input; usually buttons and LEDs

I/O pins (Input Output)

The Arduino Development Board



What is Arduino

The word “Arduino” can mean 3 things

A physical piece of hardware



A programming environment



A community & philosophy



Base Terminology

“*sketch*” – a program you write to run on an Arduino board

“*pin*” – an input or output connected to something.
e.g. output to an LED, input from a knob.

“*digital*” – value is either HIGH or LOW.
(aka on/off, one/zero) e.g. switch state

“*analog*” – value ranges, usually from 0-255.
e.g. LED brightness, motor speed, etc.

IDE

- ◎ To **program** the Arduino we need special software that will let us connect to the Arduino so we can code it.
- ◎ IDE stands for **Integrated Development Environment** . Let's break down that down:
- ◎ **Development** stands for software development or **coding** . We use code to create **programs**
- ◎ **Environment** is exactly what it means, humans exist in a environment that's made up of oxygen and water + lots of other stuff we need to live. In **computer programming** an environment includes all the things we need to code our Arduino in it's little Arduino world.
- ◎ **Integrated** : It just means that everything you need to code is all one place.

Getting Started

- 1 | Get an Arduino board and USB cable
- 2 | Download the Arduino Software (IDE)
- 3 | Connect the board
- 4 | Install the drivers
- 5 | Launch the Arduino application
- 6 | Open the blink example
- 7 | Select your board
- 8 | Select your serial port
- 9 | Compile the program
- 10 | Upload the program



2. Download Arduino Software (IDE)

Download the Arduino Software



ARDUINO 1.6.6

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer

Windows ZIP file for non admin install

Mac OS X 10.7 Lion or newer

Linux 32 bits

Linux 64 bits

[Release Notes](#)

[Source Code](#)

[Checksums](#)

<https://www.arduino.cc/en/Main/Software>

If the port is MISSING, it means that you need a driver.

- Go to <https://tinyurl.com/maerc-ch34x>
- Download the link in under “Solution 2”

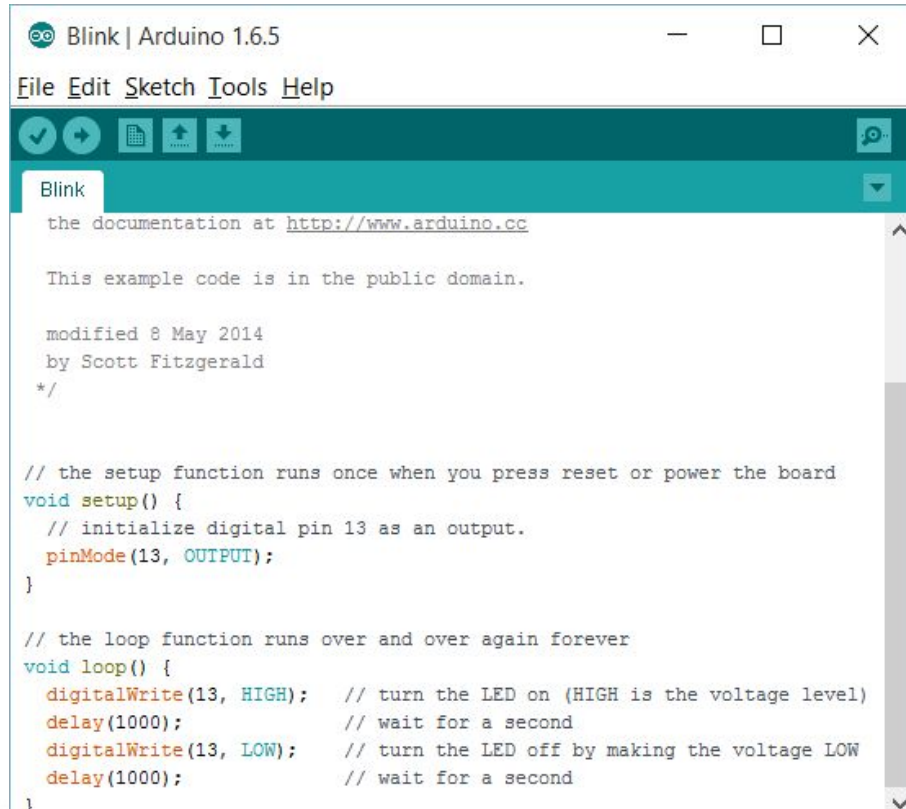
3. Try It: Connect the USB Cable

- ◎ The Arduino automatically draws power from either the USB connection to the computer or an external power supply.
- ◎ Connect the Arduino board to your computer using the USB cable. The green power LED (labelled PWR) should go on.



6. Open the LED Blink Example

- File > Examples > 01.Basics > Blink

A screenshot of the Arduino IDE window titled "Blink | Arduino 1.6.5". The window has a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checking, running, uploading, and downloading. The main text area shows the "Blink" example code. The code includes a comment about the documentation at <http://www.arduino.cc>, a note that the code is in the public domain, and the author's name "Scott Fitzgerald". The code defines a setup function to initialize pin 13 as an output and a loop function that turns the LED on and off every 1000 milliseconds.

```
Blink

the documentation at http://www.arduino.cc

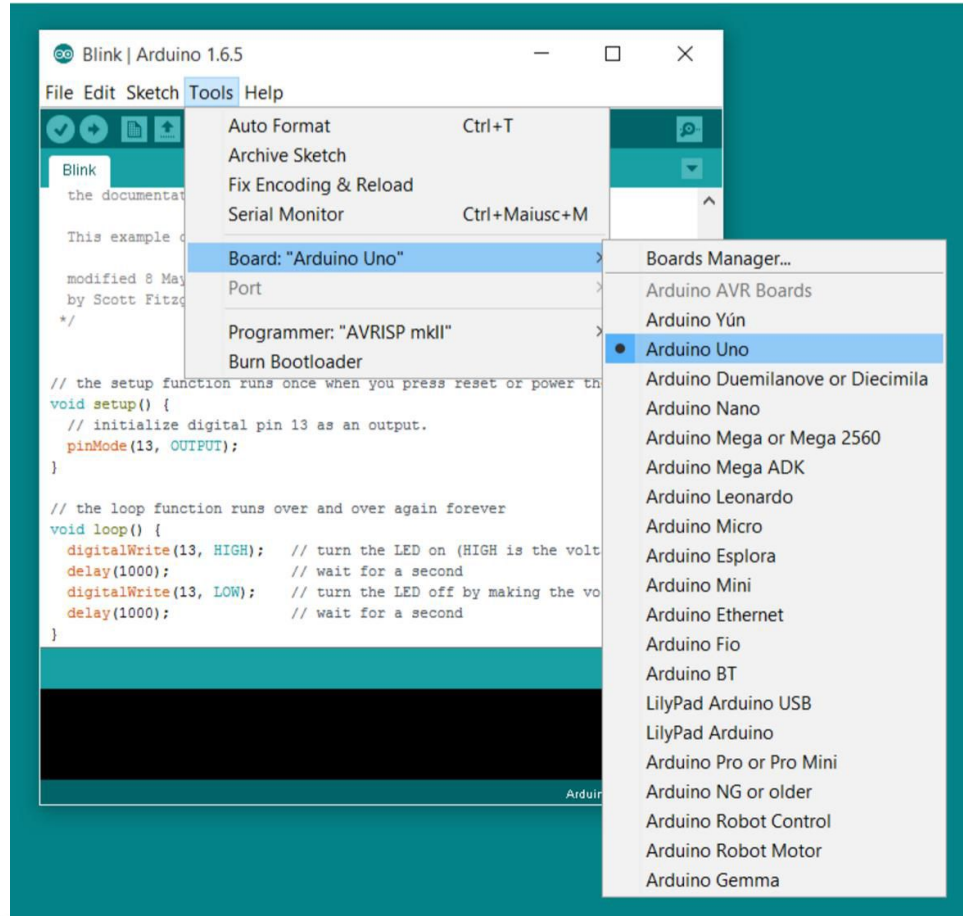
This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);            // wait for a second
}
```

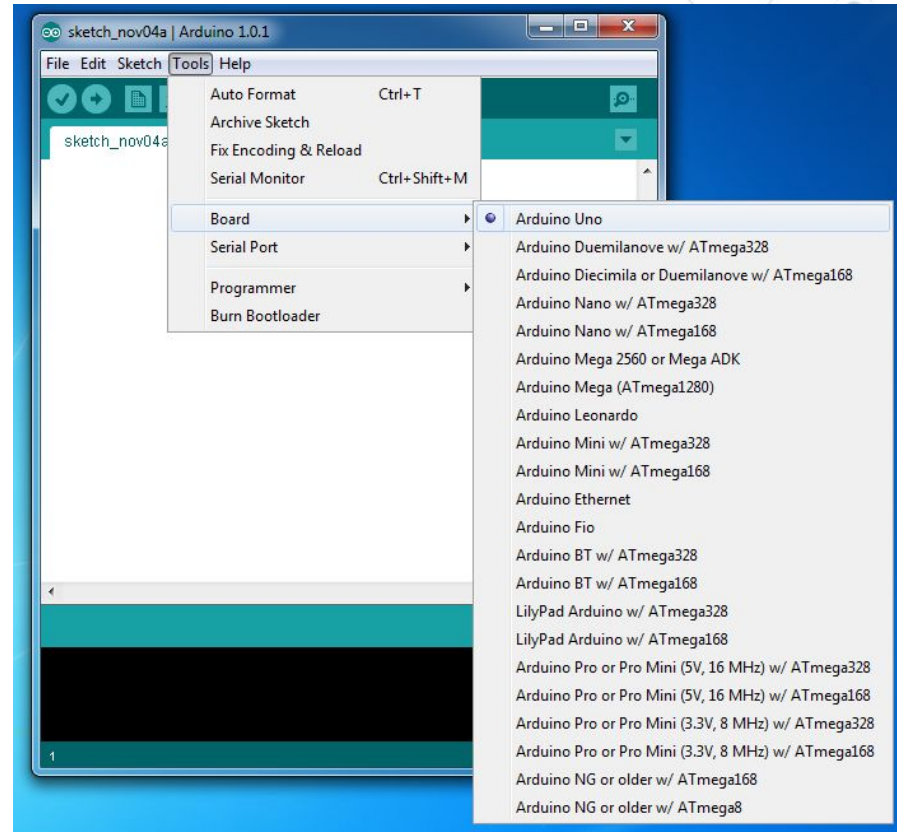
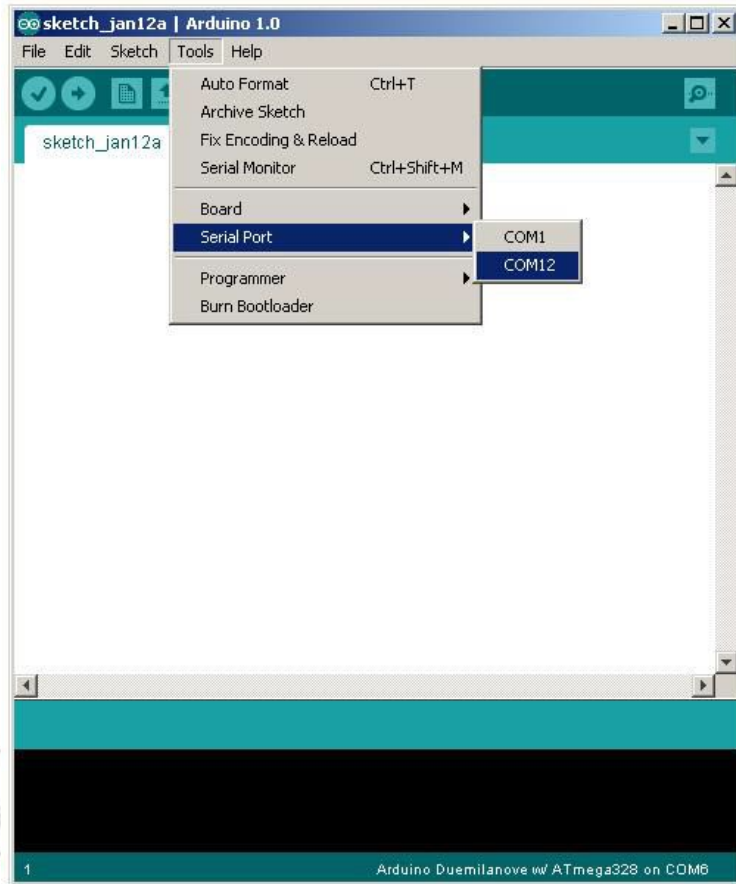
7. Select Your Arduino Board Type



8A. Select your COM (Serial) Port

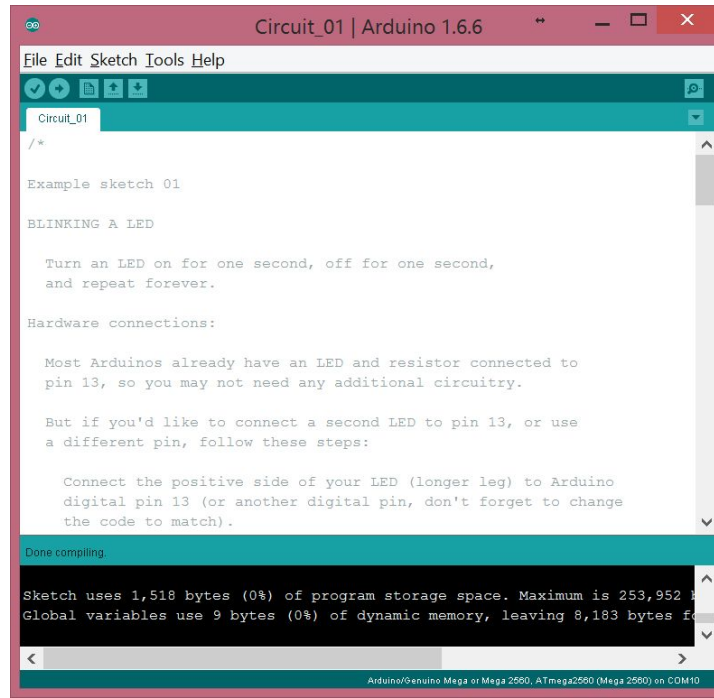
- ⦿ Select the serial device of the Arduino board from the Tools | Serial Port menu.
- ⦿ This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).
- ⦿ To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board.
- ⦿ Reconnect the board and select that serial port.

8B. Select Serial Port and Board



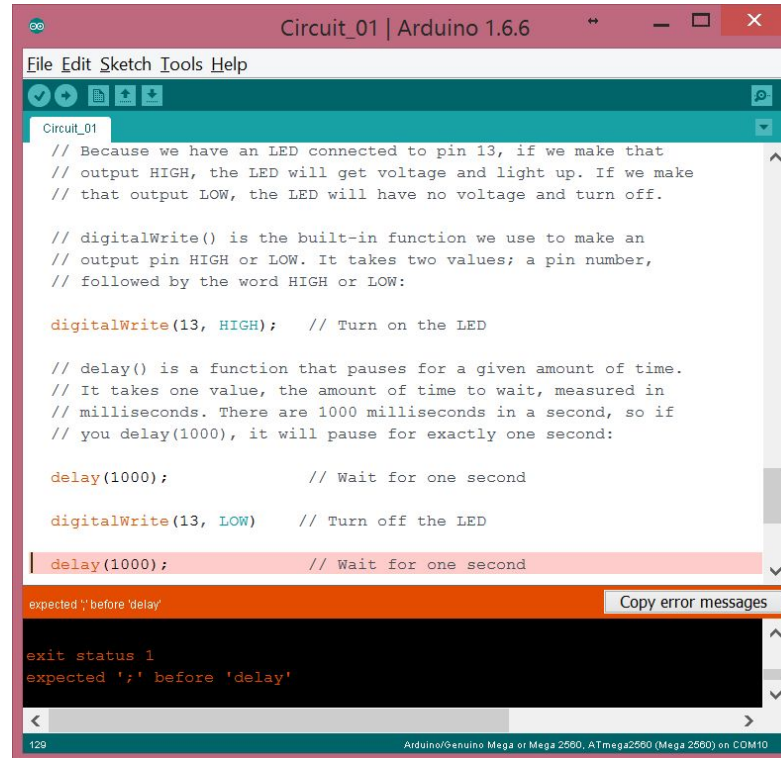
9A. Compile

Once the required file is open in the environment, press the verify button (a tick inside a circle) to compile your sketch. If the compilation is successful, the message "Done compiling." will appear in the status bar.



9B. Error in Compilation

If there is any error during the compilation process, the status bar will show the error message along with the line number where the error occurred.



The screenshot shows the Arduino IDE interface with a sketch named 'Circuit_01'. The code in the editor includes comments about an LED connected to pin 13 and uses the `digitalWrite` and `delay` functions. The line `delay(1000);` is highlighted in red, indicating an error. The error message at the bottom states: 'expected ';' before 'delay''. A 'Copy error messages' button is visible next to the error text.

```
File Edit Sketch Tools Help
Circuit_01
// Because we have an LED connected to pin 13, if we make that
// output HIGH, the LED will get voltage and light up. If we make
// that output LOW, the LED will have no voltage and turn off.

// digitalWrite() is the built-in function we use to make an
// output pin HIGH or LOW. It takes two values; a pin number,
// followed by the word HIGH or LOW:

digitalWrite(13, HIGH); // Turn on the LED

// delay() is a function that pauses for a given amount of time.
// It takes one value, the amount of time to wait, measured in
// milliseconds. There are 1000 milliseconds in a second, so if
// you delay(1000), it will pause for exactly one second:

delay(1000); // Wait for one second

digitalWrite(13, LOW) // Turn off the LED

delay(1000); // Wait for one second

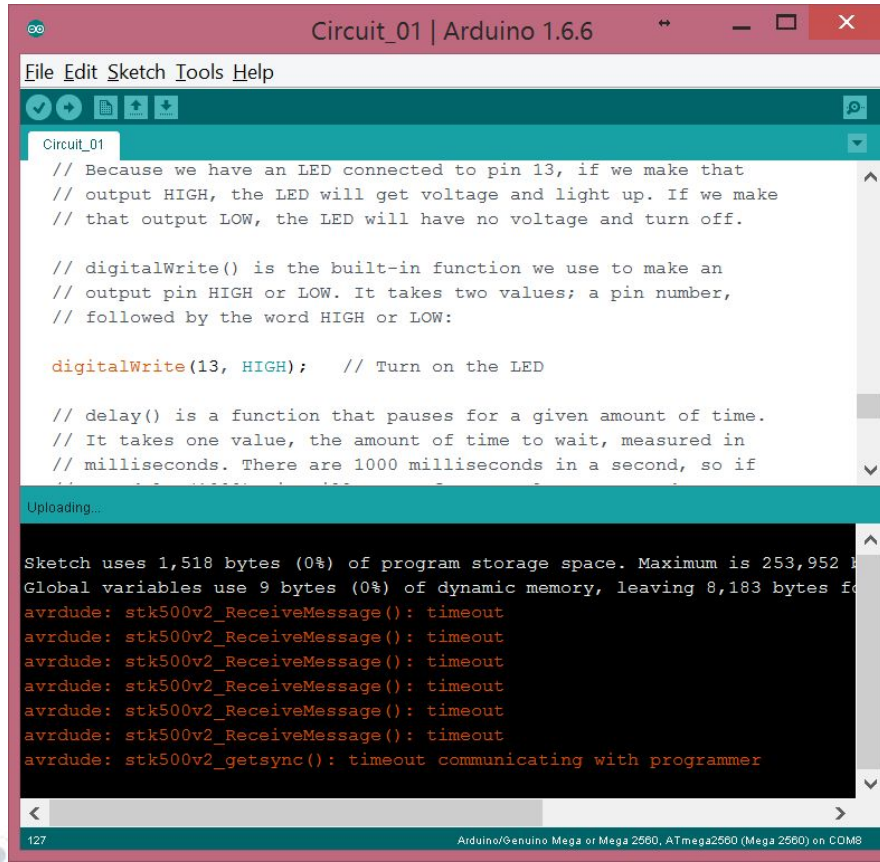
expected ';' before 'delay'
exit status 1
expected ';' before 'delay'
129 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM10
```

10A. Upload the Program to the Arduino Board

- ⦿ Now, simply click the "Upload" button in the environment.
- ⦿ Wait a few seconds you should see the RX and TX LEDs on the board flashing.
- ⦿ If the upload is successful, the message "Done uploading." will appear in the status bar.
- ⦿ A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange).
- ⦿ If it does, congratulations! You've gotten Arduino up and running.

10B. Error in upload

Upload error occurs mostly because of a **wrong COM port** or a **wrong board selected**.

A screenshot of the Arduino IDE interface. The title bar reads 'Circuit_01 | Arduino 1.6.6'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for opening, saving, and running. The main text area shows a sketch named 'Circuit_01' with the following code:

```
// Because we have an LED connected to pin 13, if we make that
// output HIGH, the LED will get voltage and light up. If we make
// that output LOW, the LED will have no voltage and turn off.

// digitalWrite() is the built-in function we use to make an
// output pin HIGH or LOW. It takes two values; a pin number,
// followed by the word HIGH or LOW:

digitalWrite(13, HIGH); // Turn on the LED

// delay() is a function that pauses for a given amount of time.
// It takes one value, the amount of time to wait, measured in
// milliseconds. There are 1000 milliseconds in a second, so if
```

The status bar at the bottom indicates '127' and 'Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM8'. The output window at the bottom shows the upload progress and error messages:

```
Uploading...

Sketch uses 1,518 bytes (0%) of program storage space. Maximum is 253,952 B.
Global variables use 9 bytes (0%) of dynamic memory, leaving 8,183 bytes for
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_ReceiveMessage(): timeout
avrdude: stk500v2_getsync(): timeout communicating with programmer
```

Structure

Each program (is normally called a sketch) has two required functions (called also routines).

void setup() { }

Contains all the code, between two curly brackets, that will be run once when your Arduino program is started.

void loop() { }

This function is run after the setup function has finished.

After it has run once it will be executed again, and again like a loop, until the power is removed from the microcontroller.

Syntax

`//` (single line comment)

Is useful to write notes to yourself as you need to put a remark to a line of code. Type two forward slashes and everything until the end of the line will be ignored by the program.

`/* */` (multi line comment)

`{ }`

Is used to define when a block of code starts and ends (is normally used in functions and in loops).

`;`

Each line of code must be ended with a semicolon (a missing semicolon is a quite common reason for a program not compiled).

Variables

A program is nothing more than a set of instructions to move data around the RAM of the microcontroller. Variables are used to store one or more information. Below are listed all the different type of data that can be used to program the microcontroller. The most important information that should be kept in mind when writing programs microcontrollers, is that normally the memory / space available is very limited, this means that should always used the smallest type variable (in bits).

int

Is the most important and used type of variable, it stores numbers using 2 bytes (16 bits). Has no decimal places and store a values between -32,768 and 32,767.

long (long)

When an integer is not big enough, the long is the next option possible. It takes 4 bytes (32 bits) of RAM and has a range between -2,147,483,648 and 2,147,483,647.

More Variables

boolean `boolean`

Is a simple variable that contain a True or False variable. It's useful because it only require one bit of RAM.

float `(float)`

Used for floating point math (with decimals). It takes 4 bytes (32 bits) of RAM and has a range between $3.4028235E+38$ and $3.4028235E+38$.

char `(character)`

Stores one character using the ASCII code (ie 'A' = 65). Uses one byte (8 bits) of RAM. The language handles strings as an array of char's.

String `(contains several characters)`

It stores several characters and is easy to use for beginners. The main disadvantage is that requires a lot of memory and in some versions of the programming IDE can break the execution of your program.

Math Operators

Operators are used to manipulate numbers

+ (addition)

- (subtraction)

* (multiplication)

/ (division)

= (assignment)

makes something equal to something else (eg . $x = 10 * 2$ (x now equals 20)).

% (modulo)

gives the remainder when one number is divided by another (ex. $12 \% 10 = 2$)

++ (increment) -- (decrement)

The variable is incremented / decremented of one unit.

(e.g. $4++ = 5$; $3-- = 2$;))

Comparison Operators

Operators used for logical comparison between variables.

`==` eg. `12 == 10` is FALSE or `12 == 12`

`!=` eg . `12 != 10` is TRUE or `12 != 12`

`<` eg . `12 < 10` is FALSE or `12 < 12` is FALSE or `12 < 14`

`>` eg . `12 > 10` is TRUE or `12 > 12` is FALSE or `12 > 14` is FALSE

Control Structure

```
if(condition){ } else if( condition ){ } else { }
```

Execute the code between the curly brackets if the condition is true, if not true it will test the else if condition if that is also false the else code will execute. Is possible to use several else if, but is important to remind that some time if the conditions are not well written the code can work improperly.

```
for( int i = 0; i < number_of_repetitions; i++) { }
```

If you would like to repeat a part of code a defined number of times you should use the for control structure you should use the for control structure (can count up i ++ or down i- or use any variable)

Inputs and Outputs

- ◎ The microcontroller, like a small computer, allows to interact with external resources, this allows our programs to read or execute some physical action outside the microcontroller.
- ◎ The type of interaction can be DIGITAL or ANALOG.
- ◎ Digital means that some data like 01010010 are sent over one line
- ◎ Analog is when an audio is played like in the old style tapes.
- ◎ Some examples are: to switch on or off a light, to read a temperature, play a melody or to send some data over a network.

Digital

pinMode (pin number, mode Input/Output);

- ⊙ Is used to set the functionality of a pin, pin is the pin number that you would like to address 0 19 (analog 0-5 are 14-19). The mode can either be INPUT or OUTPUT.

digitalWrite (pin, value);

- ⊙ When a pin is set as an OUTPUT, it can be set either HIGH (pulled to +5 volts) or LOW (pulled to ground, GND).

int digitalRead (pin);

- ⊙ When the pin is set as an INPUT, the information read can be HIGH (pulled to +5 volts) or LOW (pulled to ground, GND).

Analog

The microcontroller is a digital machine but it has also the ability to operate in the analog realm (using some tricks).

pinMode (pin, mode);

- Is used to set the functionality of a pin, pin is the pin number that you would like to address 0 19 (analog 0 5 are 14 19). The mode can either be INPUT or OUTPUT.

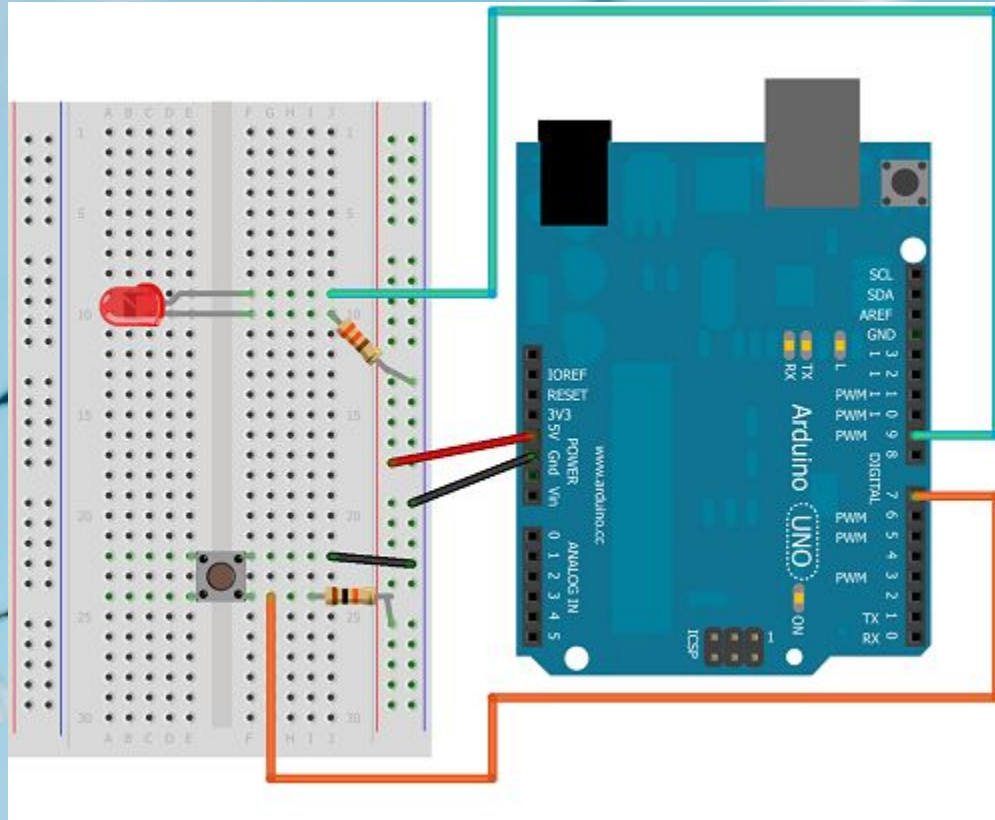
int analogWrite (pin, value);

- Some of the Arduino's pins support a special functionality that is the pulse width modulation (3, 5, 6, 9, 10, 11). This turns the pin on and off very quickly making it act like an analog output. The value is any number between 0 (0% duty cycle ~0v) and 255 (100% duty cycle ~5 volts).

int analogRead (pin);

When the analog input pins are set, is possible to read their voltage. A value between 0 (for 0 volts) and 1024 (for 5 volts) is returned.

Arduino



MICROCONTROLLERS

Brain of hardware
Have multiple lines, or connections

ATmega328P pin mapping

PC6	1	28	PC5
PD0	2	27	PC4
PD1	3	26	PC3
PD2	4	25	PC2
PD3	5	24	PC1
PD4	6	23	PC0
VCC	7	22	GND
GND	8	21	AREF
PB6	9	20	AVCC
PB7	10	19	PB5
PD5	11	18	PB4
PD6	12	17	PB3
PD7	13	16	PB2
PB0	14	15	PB1

(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)
VCC	7	22	GND
GND	8	21	AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)

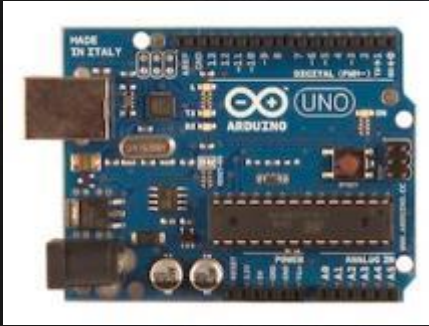




ARDUINO FAMILY



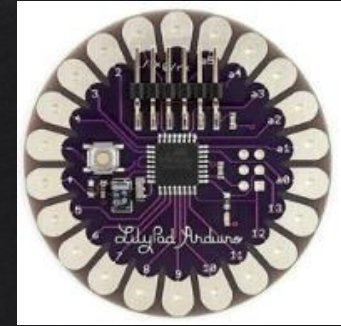
MAE Robotics Club



Uno



Mega



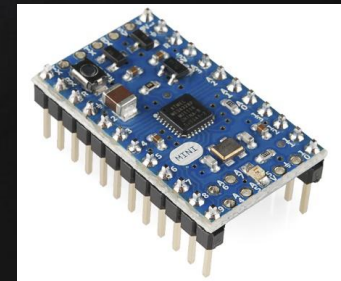
LilyPad



BT



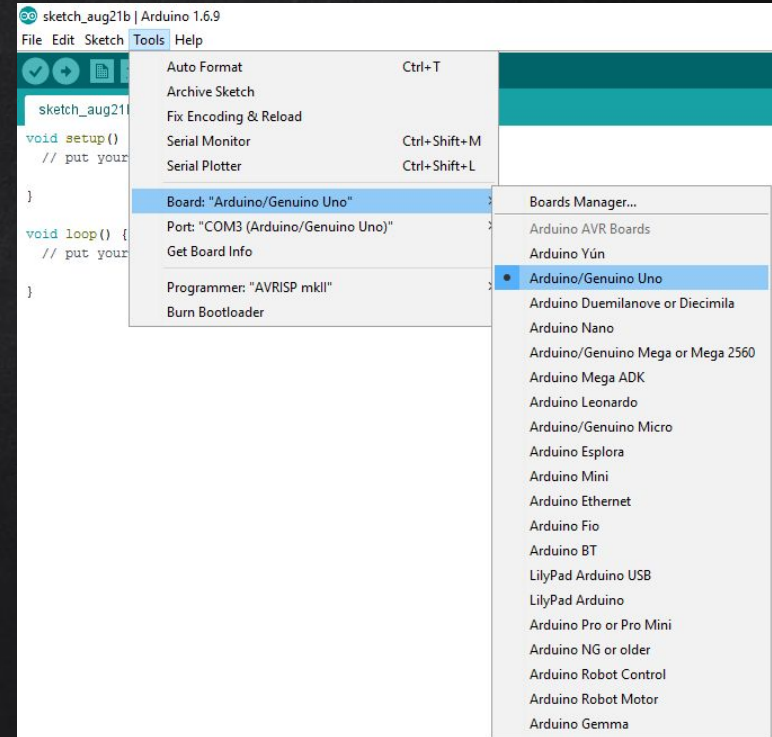
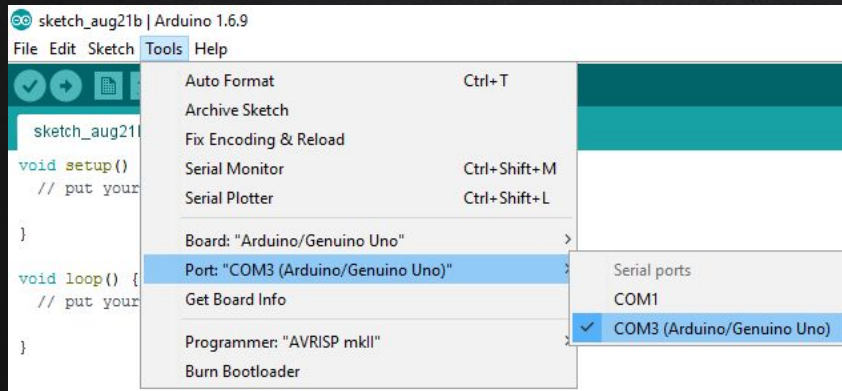
Nano



Mini

SETUP

- Select board (Arduino Uno)
- Select port (it changes if you change to a different USB port)





STANDARD FUNCTIONS



MAE Robotics Club

- `setup()`: A function present in every Arduino sketch. Run once before the `loop()` function. Often used to set pinmode to input or output -> variable declaration

The `setup()` function looks like:

```
void setup() {  
  //code goes here  
}
```

- `loop()`: A function present in every single Arduino sketch. The `loop()` is where (almost) everything happens. The one exception to this is `setup()` and variable declaration.

The `loop()` function looks like:

```
void loop() {  
  //code goes here  
}
```



Two required functions /
methods / routines:

```
void setup()
{
    // runs once
}
```

```
void loop()
{
    // repeats
}
```

KEY TERMINOLOGY

input:	A pin mode that intakes information	
output:	A pin mode that sends information	
HIGH:	Electrical signal present (5V for Uno).	-> ON /True
LOW:	No electrical signal present (0V).	-> OFF or False

BASIC SYNTAX

`;` Semicolon

Used to end a statement.

`{ }` Curly Braces / `()` Parentheses / `[]` Brackets

`//` or `/**/` Comments



STANDARD FUNCTIONS

- `PinMode(pin_number, INPUT/OUTPUT)`

This function declares a given pin to be input or output pin.

- `digitalWrite(pin_number, 0/1/HIGH/LOW):`

This function produces a digital signal of either HIGH(5V) or LOW(0V) on the given pin.

- `analogWrite(pin_number, value)`

This function produces an analog signal varying between 0 to 5V on the given pin.

- `analogRead(pin_number)`

This function reads the voltage on the given pin and outputs its value as an integer ranging from 0 to 1023.

- `digitalRead(pin_number)`

This function reads the voltage on the given pin and outputs its value as either 0 or 1.



STANDARD FUNCTIONS



MAE Robotics Club

- `Serial.begin(baud_rate)`

This function begins serial communication between the microcontroller and the arduino.









- `Serial.print()`

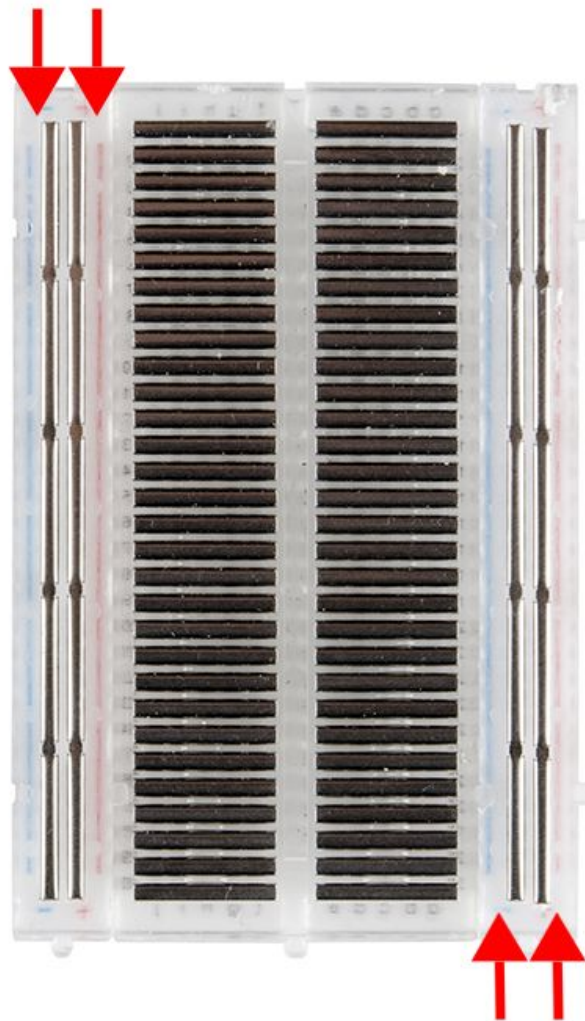
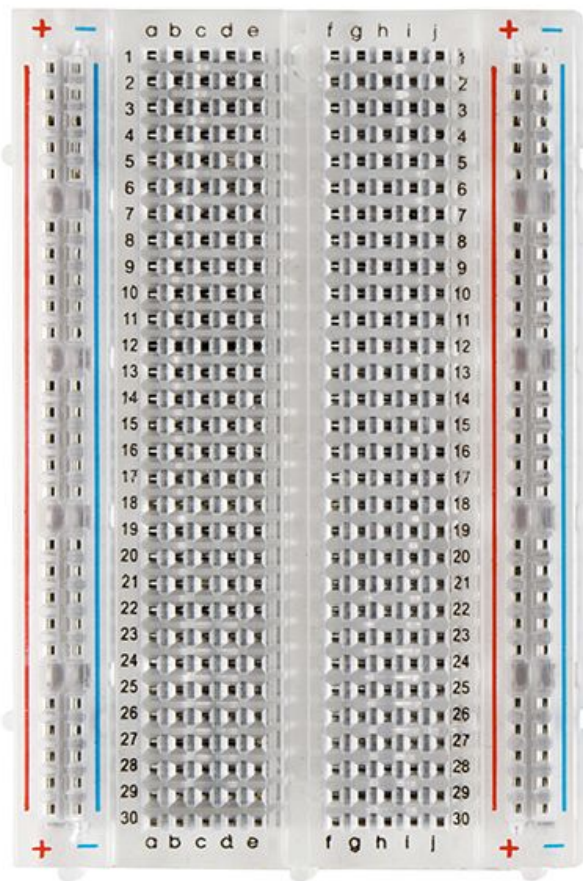
This function prints the value in its argument on the serial monitor of the arduino IDE.

- `delay(time)`

This function provides delay of given time (in milliseconds) during execution of the code.

- `void setup()` : The code inside this function is run only once, when new code has been uploaded.
- `void loop()` : The code inside this runs on a loop. Without any delay function, the default rate of delay is 1ms.

Name	Image	Type	Function	Notes
Push Button		Digital Input	Switch - Closes or opens circuit	Polarized, needs resistor
Trim potentiometer		Analog Input	Variable resistor	Also called a Trimpot.
Photoresistor		Analog Input	Light Dependent Resistor (LDR)	Resistance varies with light.
Relay		Digital Output	Switch driven by a small signal	Used to control larger voltages
Temp Sensor		Analog Input	Temp Dependent Resistor	
Flex Sensor		Analog Input	Variable resistor	
Soft Trimpot		Analog Input	Variable resistor	Careful of shorts
RGB LED		Dig & Analog Output	16,777,216 different colors	Ooh... So pretty.



BREADBOARD UNDERSTANDING TEST

1. Is A5 connected to E5?
2. Is A5 connected to A7?
3. Is E5 connected to F5?
4. Is D10 connected to H50?

BREADBOARD UNDERSTANDING TEST

- | | |
|-----------------------------|-----|
| 1. Is A5 connected to E5? | YES |
| 2. Is A5 connected to A7? | NO |
| 3. Is E5 connected to F5? | NO |
| 4. Is D10 connected to H50? | NO |

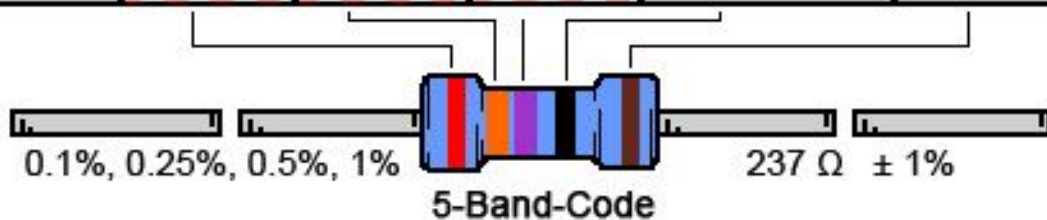


MAE Robotics Club

HOW TO READ RESISTORS



COLOR	1 ST BAND	2 ND BAND	3 RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)





MAE Robotics Club

HANDS-ON

Hardware

- ✕ Arduino Uno
- ✕ Breadboard
- ✕ Type A-B USB Cable
- ✕ 220 Ohm Resistor
- ✕ LED
- ✕ Push Button
- ✕ Buzzer
- ✕ Jumper Wires – 5

Software

- ✕ Arduino IDE –
<https://www.arduino.cc/en/Main/Software>
- ✕ Driver (May not need) –
<https://sparks.gogo.co.nz/ch340.html>

RMB TO ADD RESISTOR

Arduino current **< 50mA**

LED current **< 30mA**

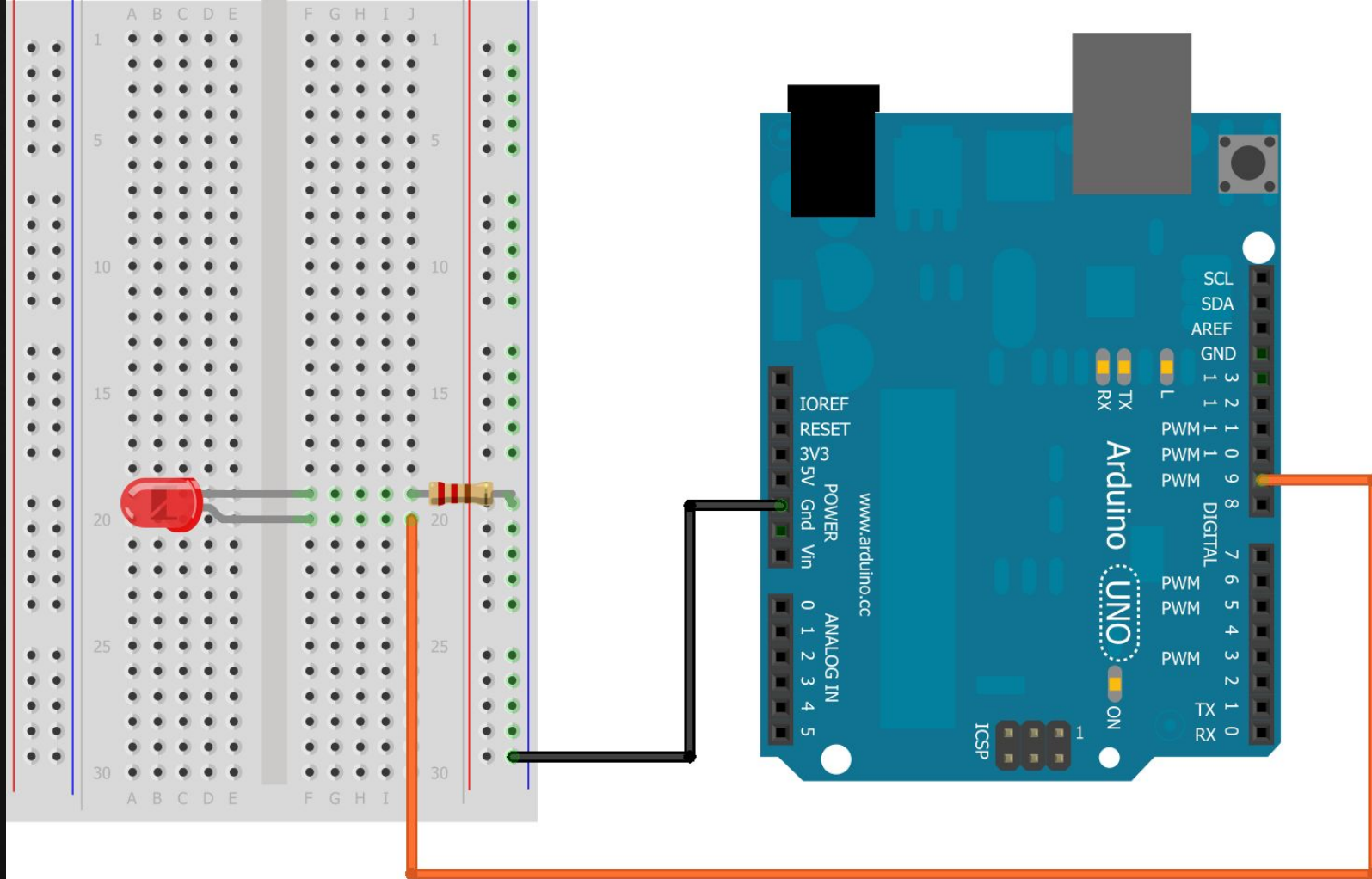
Source voltage = 5V

LED voltage drop = 2V

Resistance = 220 Ohm

$$V = iR$$

$$\text{Current} = (5V - 2V) / 220 \text{ Ohm} = \mathbf{14mA}$$



CODE



MAE Robotics Club

```
void setup() {  
    pinMode(9, OUTPUT); }  
  
//Initialize ledPin (pin 9) as OUTPUT pin
```

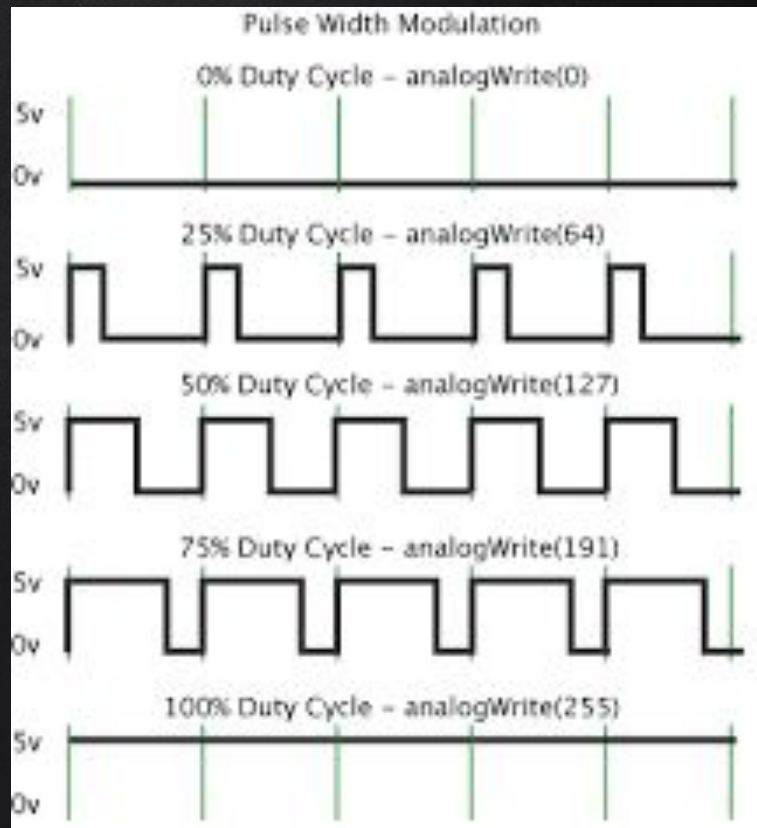
```
void loop() {  
    digitalWrite(9, HIGH);  
    delay(1000);  
    digitalWrite(9, LOW);  
    delay(1000);  
}
```

Turn LED on by setting voltage to high, sleep for 1s, then turn LED off by setting voltage to low

PWM

You can also use pulse width modulation to control the angle of a servo motor

. Servos have a shaft that turns to specific position based on its control line. Our servo motors have a range of about 180 degrees.



PWM

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means.

Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width.

ANALOG – PWM



- `analogRead(pin_number)`

This function reads the voltage on the given pin and outputs its value as an integer ranging from 0 to 1023.

- `analogWrite(pin_number, value)`

This function produces an analog signal varying between 0 to 5V on the given pin.

This allows you to get readings from analog sensors or interfaces that have more than two states.

This allows you to set output to a PWM value instead of just HIGH or LOW.

PWM: Stands for Pulse-Width Modulation, a method of emulating an analog signal through a digital pin. A value between or including 0 and 255. Used with `analogWrite`.

ANALOG RANGE: 0 – 255

A bit is a binary digit. So a byte can hold 2^8 numbers ranging from 0 to $2^8 - 1 = 255$.

A byte, by its standard definition, is 8 bits which can represent 256 values (0 through 255).

CODE



MAE Robotics Club

```
int x = 0;
void setup() {
    pinMode(9, OUTPUT); }

void loop() {
    analogWrite(9,x);
    if(x == 255) {
        x = 0; }
    else{
        x++; }
    delay(10);
}
```

BAUD RATE



```
Serial_01 | Arduino 1.6.7
File Edit Sketch Tools Help

Serial_01
int i = 0;

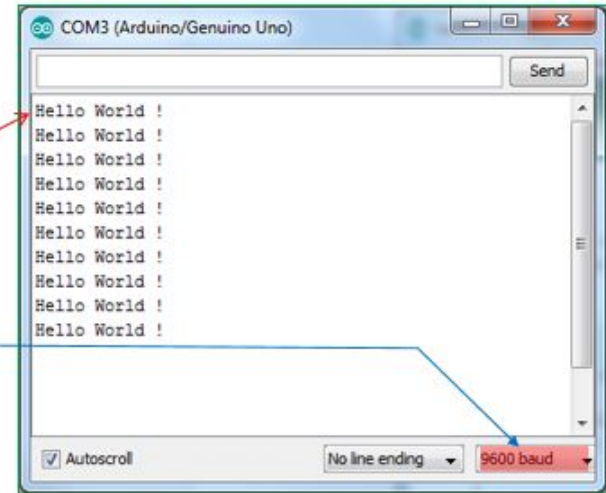
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if( i < 10) {
    Serial.println("Hello World !");
    delay(1000);
    i++;
  }
}

Done Saving.

Sketch uses 1,994 bytes (6%) of program storage space. Maximum is 32,
Global variables use 216 bytes (10%) of dynamic memory, leaving 1,832

5 Arduino/Genuino Uno on COM3
```



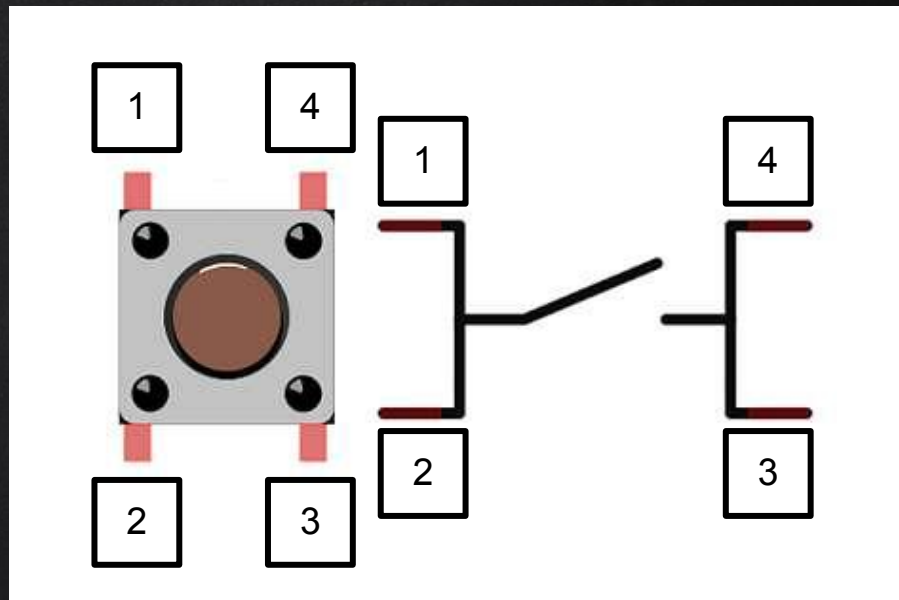
PUSH BUTTON

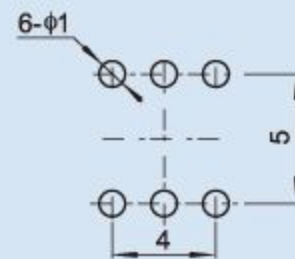
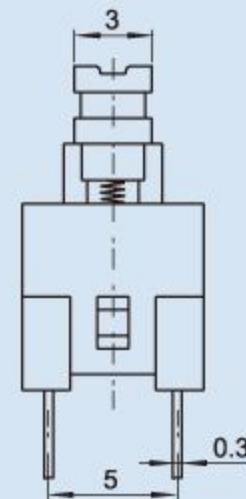
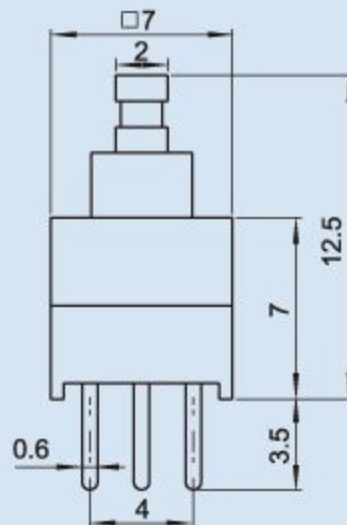
However, one must be careful to remember the orientation of the legs within the push button.

Legs 1 and 2 are always connected!

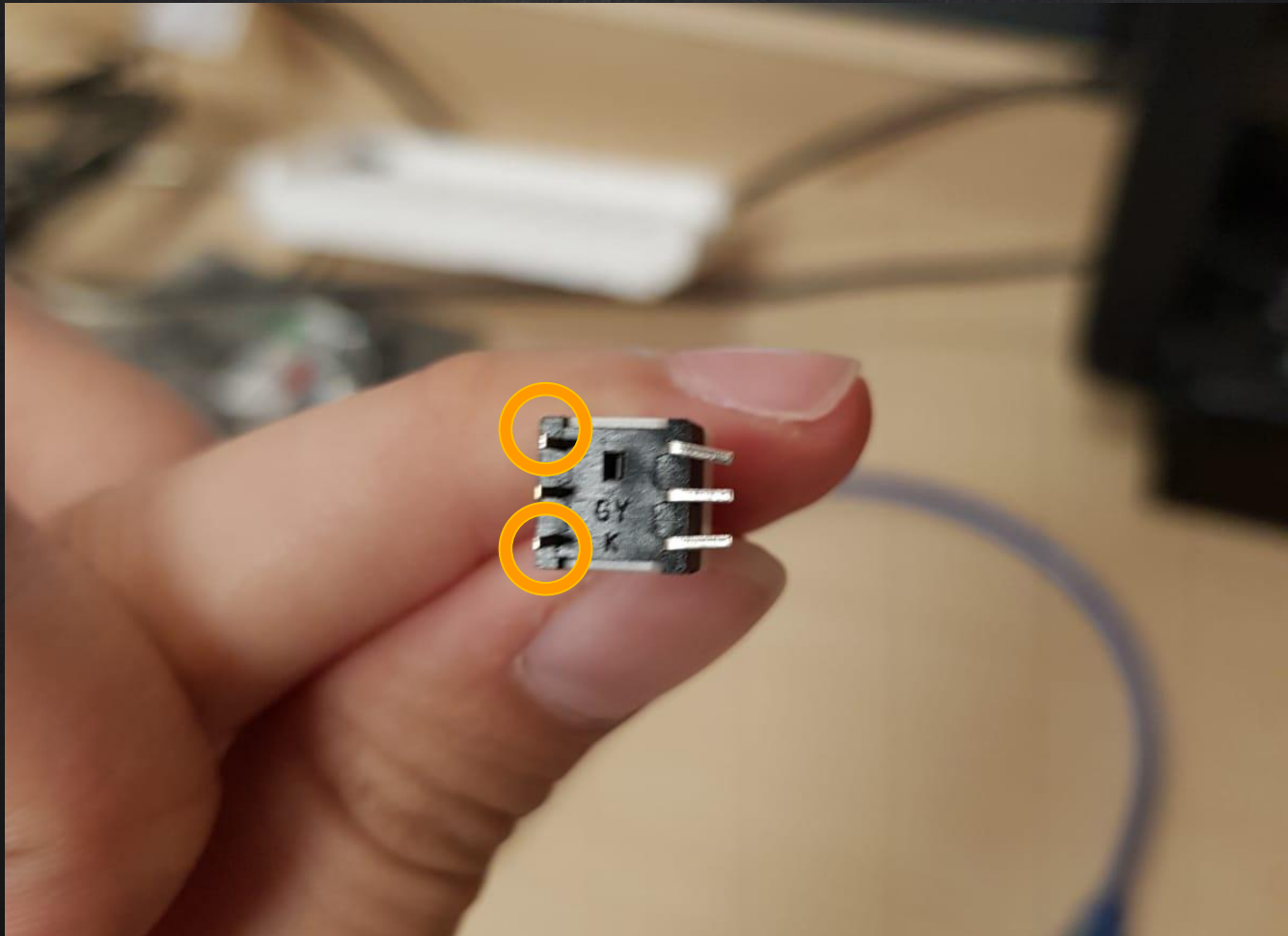
Legs 3 and 4 are always connected!

The disconnection is between 12 and 34!





KFC-7x7-B



BOOLEAN OPERATORS

<Boolean>	Description
() == ()	is equal?
() != ()	is not equal?
() > ()	greater than
() >= ()	greater than or equal
() < ()	less than
() <= ()	less than or equal

BUTTONS AND BOOLEANS

PB is pin 2

LED is pin 9

PB Set to high.

If PB Pressed, LED light up.

```
int signal1;

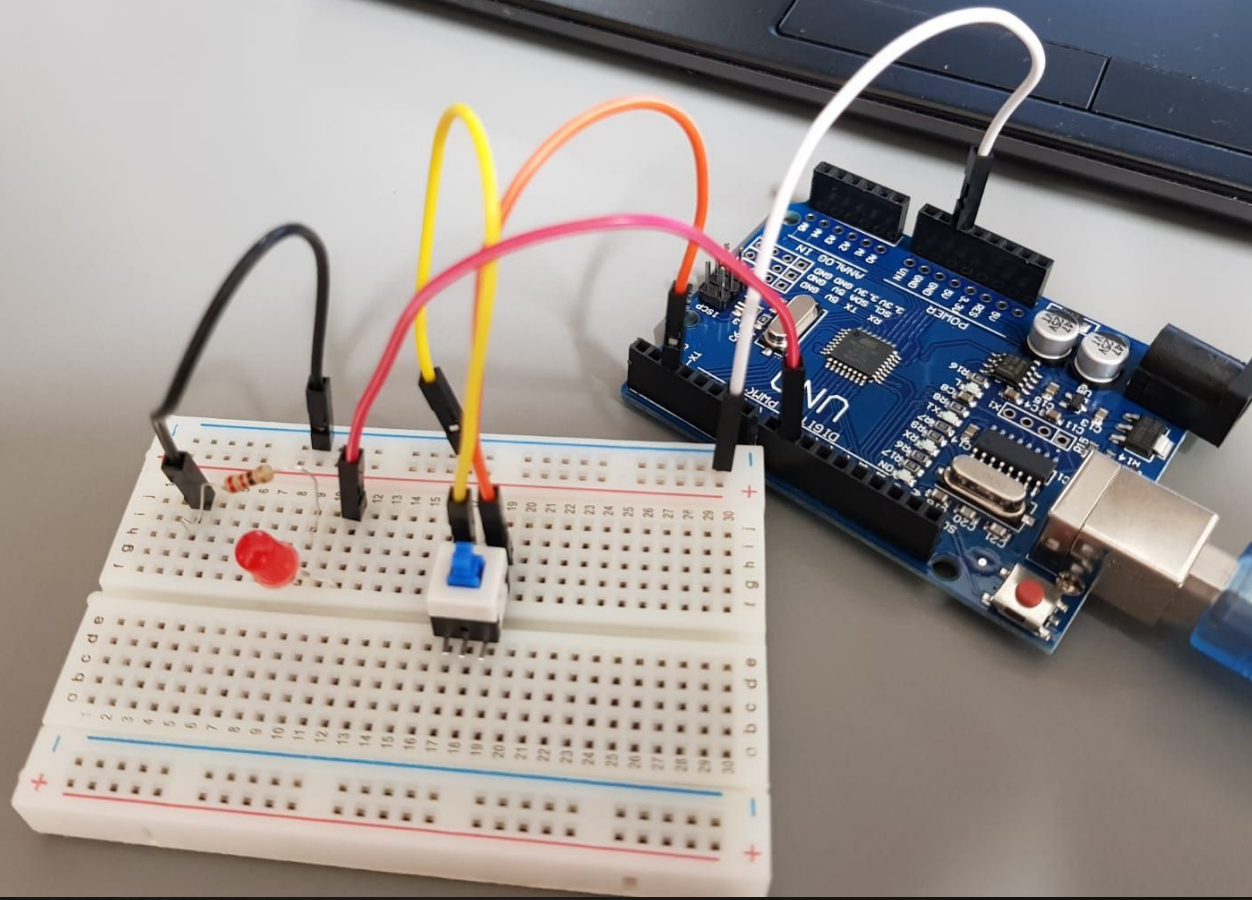
void setup() {
    pinMode(9, OUTPUT);
    pinMode(2, INPUT_PULLUP);
    Serial.begin(9600);
}

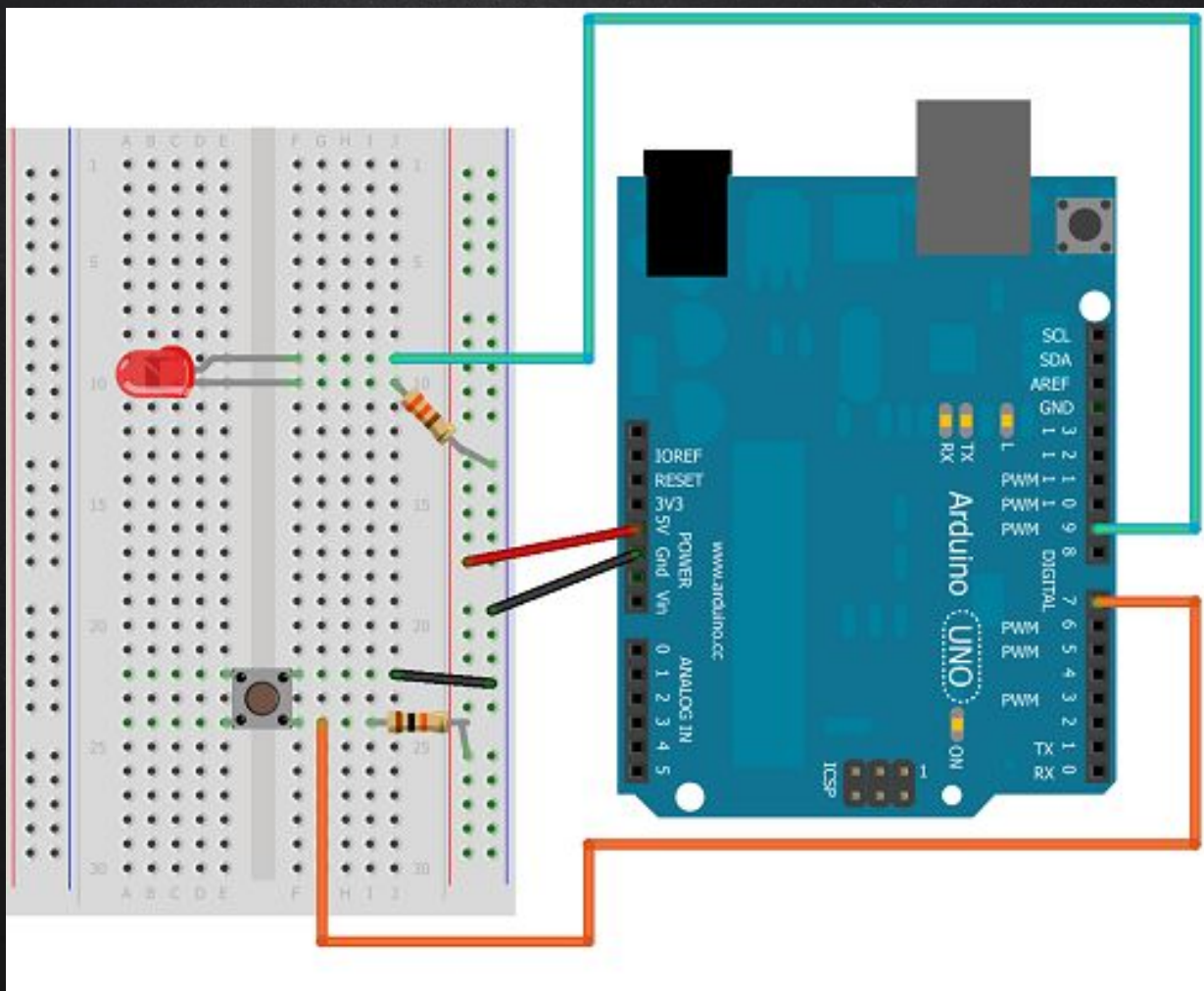
void loop() {
    signal1 = digitalRead(2);
    Serial.println(signal1);

    if (signal1 == 0){
        digitalWrite(9, HIGH);
        delay(10);
    }
    else{
        digitalWrite(9, LOW);
        delay(10);
    }
}
```



MAE Robotics Club





PIEZO BUZZER

`tone()`

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to `noTone()`. The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to `tone()` will have no effect. If the tone is playing on the same pin, the call will set its frequency.





TONE

tone()

Syntax

tone(pin, frequency)

tone(pin, frequency, duration)

noTone()

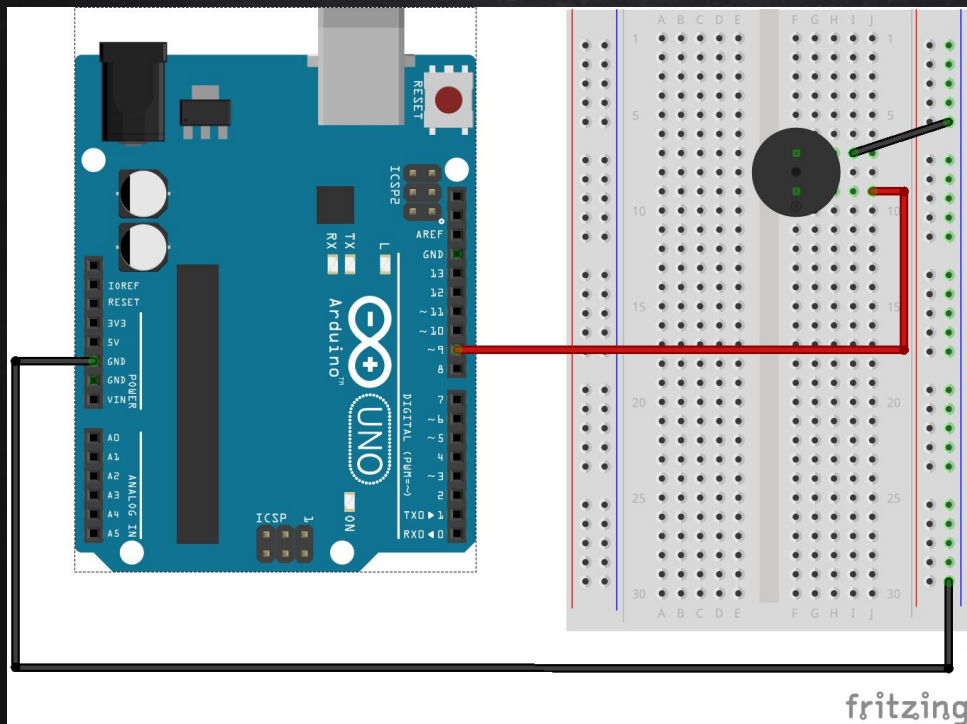
Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

NOTE: if you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

Syntax

noTone(pin)

BUZZER TIME



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

```
void loop() {  
    tone(9, 262, 500);  
    delay(1000);  
    noTone(9);  
    delay(500);  
}
```



DOORBELL

```
int signal1;
```

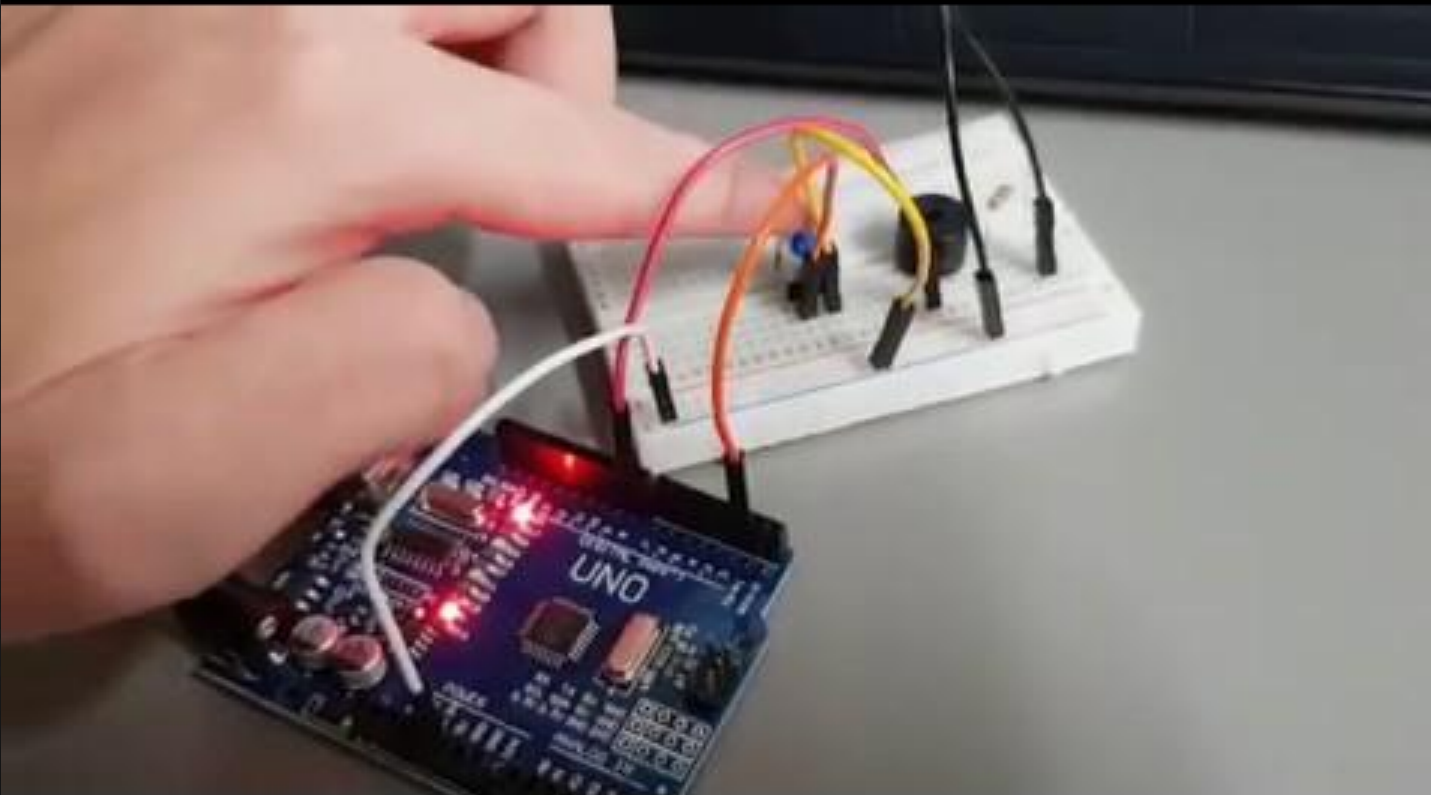
```
void setup() {  
  pinMode(9, OUTPUT);  
  pinMode(2, INPUT_PULLUP);  
}
```

```
void loop() {  
  signal1 = digitalRead(2);  
  if(signal1 == 0){  
    tone(9, 262, 500);  
    delay(600);  
    tone(9, 440, 500);  
    delay(600);  
    tone(9, 262, 1000);  
    delay(1200);  
  }  
  else{  
    noTone(9);  
  }  
}
```

DOORBELL DEMO



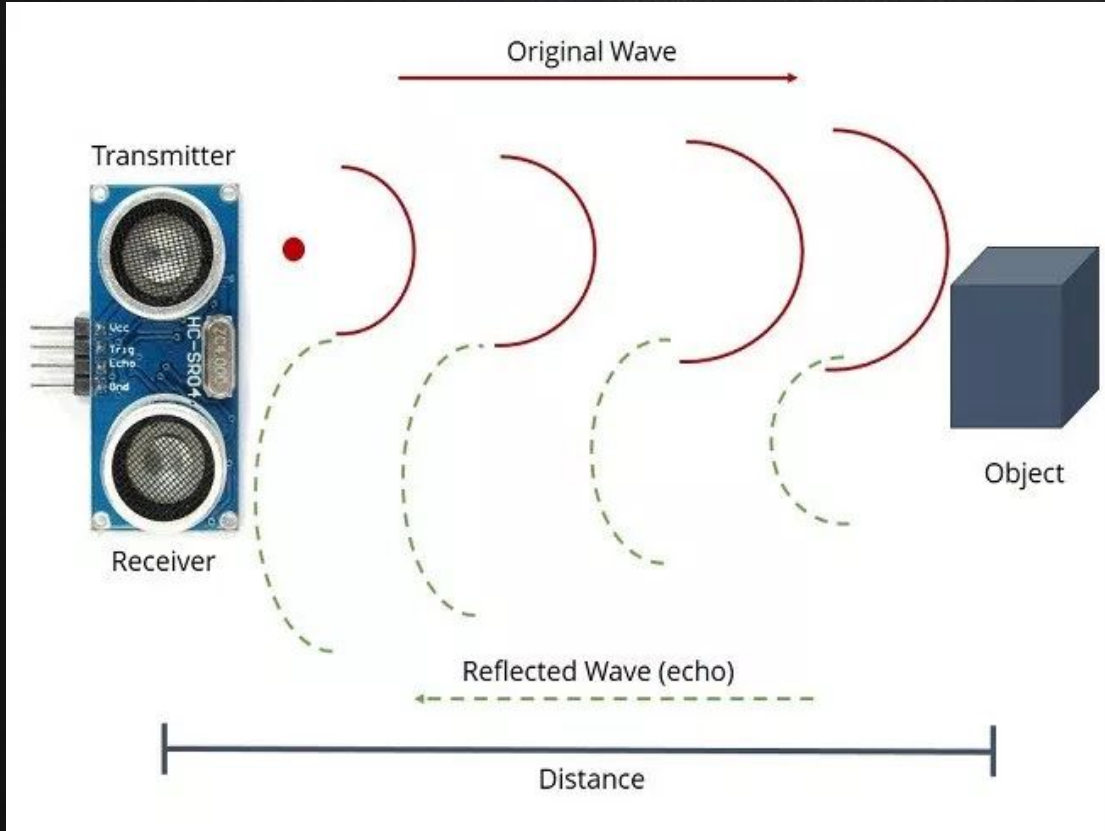
MAE Robotics Club



ULTRASONIC SENSOR



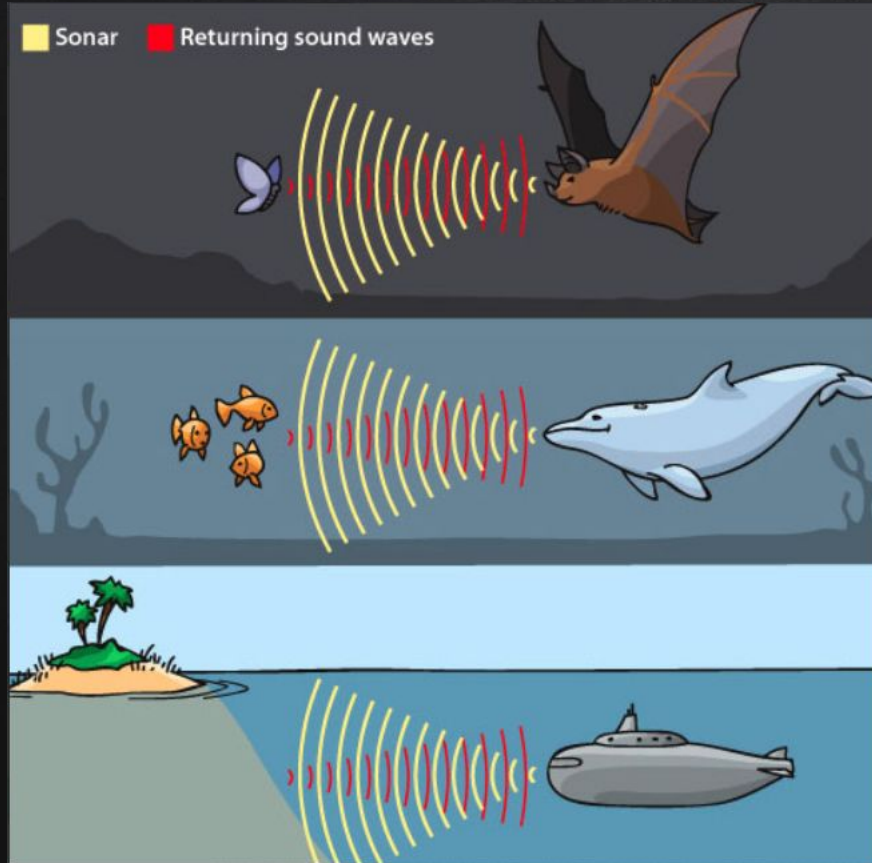
MAE Robotics Club



Ultrasound is sound waves with frequencies higher than the upper audible limit of human hearing.

Distance measured by time.

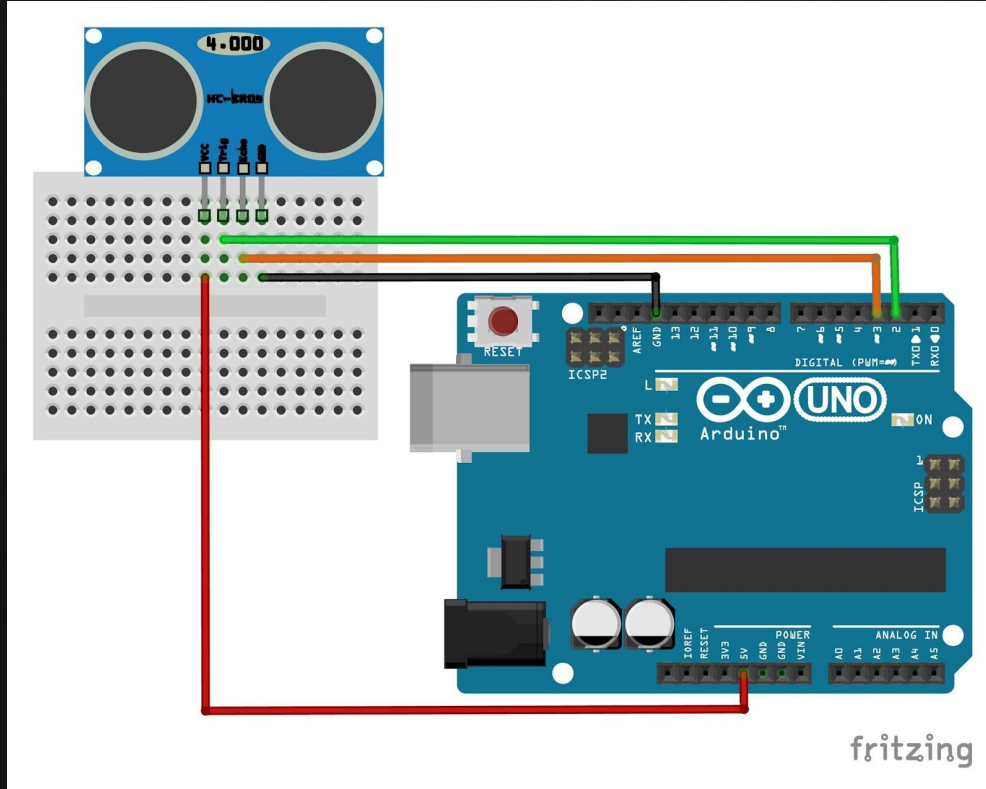
USES IN "NATURE"



Detects objects by measuring the time it takes for the returning sound waves to reach back to it after bouncing of it.

$$\text{Distance} = \text{Speed} * \text{Time}/2$$

HC-SR04 - ULTRASONIC SENSOR



4 pins: 5V, trigger, echo, GND
trigger pin > sound off a wave pulse,
echo pin > receive wave pulse

Code Part 1

Initialize serial communication



```
const int trigPin = 2;  
const int echoPin = 4;  
  
void setup() {  
  Serial.begin(9600);  
  pinMode(echoPin, INPUT);  
  pinMode(trigPin, OUTPUT);  
}
```

Establish variables



```
void loop()  
{  
  long duration, cm;
```

Code Part 2

'LOW' and 'High' pulse is produced to "clean" before every function call



```
digitalWrite(trigPin, LOW);  
delayMicroseconds(2);  
digitalWrite(trigPin, HIGH);  
delayMicroseconds(10);
```

Set the default to LOW



```
digitalWrite(trigPin, LOW);
```

Function Call: Read the signal

HIGH pulse whose duration is the time (in microseconds) from the sending of the ping to the reception of its echo off an object.



```
duration = pulseIn(echoPin,  
HIGH);
```

Code Part 3

sound waves travel at 340m/s



```
cm = duration * 0.034/2;
```

prints out



```
Serial.print(cm);  
Serial.print(" cm");  
Serial.println();
```

```
delay(100);  
}
```

Speed Facts				
	<i>Meter per Second</i>	<i>Kilometers per Hour</i>	<i>Feet per Second</i>	<i>Miles per Hour</i>
Sound* (at sea level)	340	1,225	1,116	761
Light	299,792,458	1,079,252,849	983,571,056	670,616,629

Car reverse warning system

Ultrasonic Sensor

Buzzer

LED

3 states:

Slow Blink - not near

Normal Blinking - near

Quick Blinking - at
limit



Car reverse warning system

```
trigPin = 2;  
echoPin = 4;  
led = 8;  
buzzer = 7;
```

```
const int trigPin = 2; //ultra trig int  
const int echoPin = 4; //ultra echo int
```

```
void setup() {  
  Serial.begin(9600);  
  pinMode(8, OUTPUT); //led  
  pinMode (7, OUTPUT); //buzzer
```

```
void loop()  
{  
  long duration, cm;
```

```
  pinMode(trigPin, OUTPUT);  
  digitalWrite(trigPin, LOW);  
  delayMicroseconds(2);  
  digitalWrite(trigPin, HIGH);  
  delayMicroseconds(10);  
  digitalWrite(trigPin, LOW);
```

```
  pinMode(echoPin, INPUT);  
  duration = pulseIn(echoPin, HIGH);  
  cm = duration * 0.034/2;
```

Car reverse warning system

if.. else loops

```
Serial.print(cm);  
Serial.print(" cm");  
Serial.println();
```

```
digitalWrite(8, LOW);  
delay(700);  
    if(cm >= 25){  
        tone(7, 500, 500);  
        digitalWrite(8, HIGH);  
        delay(500);  
        digitalWrite(8, LOW);  
        delay(500);  
    }    else if(cm >= 10){  
        tone(7, 500, 150);  
        digitalWrite(8, HIGH);  
        delay(150);  
        digitalWrite(8, LOW);  
        delay(150);  
    }    else {  
        tone(7, 700, 100);  
        digitalWrite(8, HIGH);  
        delay(10);  
        digitalWrite(8, LOW);  
        delay(10);  
    }  
}
```


Summary of Code

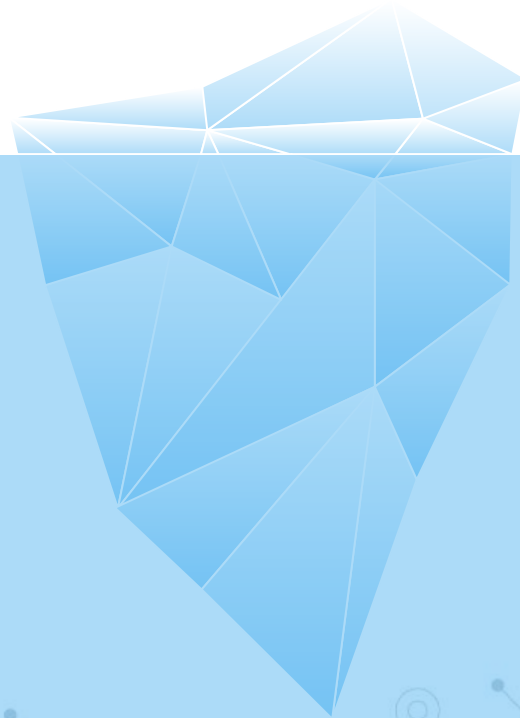
Initialize

```
trigPin = 2  
echoPin = 4  
led = 8, buzzer = 7  
variables:  
duration, cm
```

'Clearing' by
outputting
'HIGH' & 'LOW'
Calculating values

```
if >= 25cm  
    ...  
else if >= 10cm  
    ...  
else...
```

Thank you



Extra Sensors

Infrared IR Transmitter and Reciever



SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change line color, width and style.

Isn't that nice? :)

Examples:

