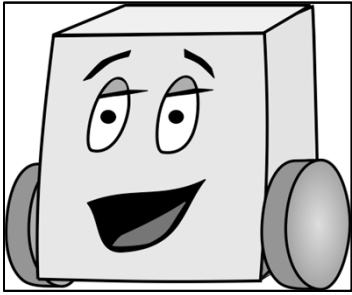


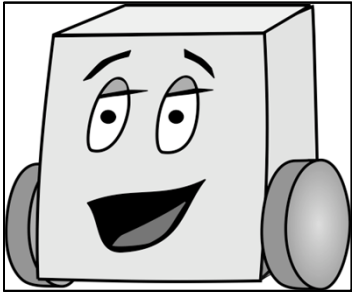
C Programming – Part 1

- What is C?
- Programming Target: Arduino
- Programming Basics
 - Simple C Program
 - Running a Program
- Programming Tools
 - Comments
 - Data Types
 - Variables
 - Console Inputs and Outputs
- More stuff you can do...

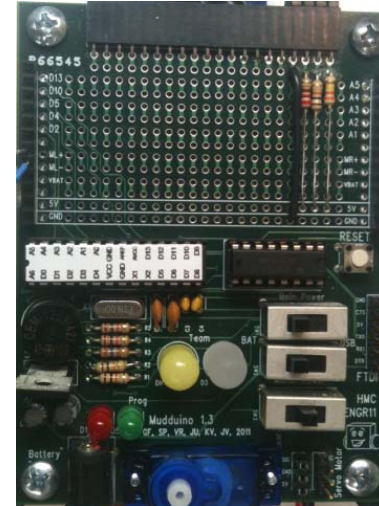


What is C?

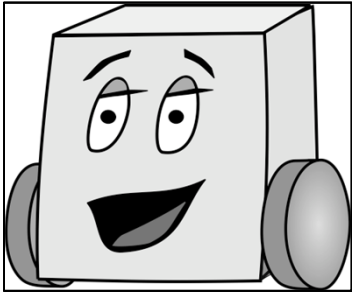
- Created by Dennis Ritchie at Bell Laboratories in 1972
- Programming language for making a computer/microcontroller do something.
- One of the most popular programming languages:
 - Available for many platforms (supercomputers to embedded microcontrollers)
 - Relatively easy to use, moderate level of abstraction, but programmer also has an idea of how code will be executed
 - Can interact with hardware directly



Programming Target: Arduino



- Arduino
 - type of microcontroller
 - we'll talk about this a lot more next time
- Overall syntax is same as C
- Some differences (we'll highlight them)
- FYI, Arduino's version of C is called "Arduino"



Simple C Program

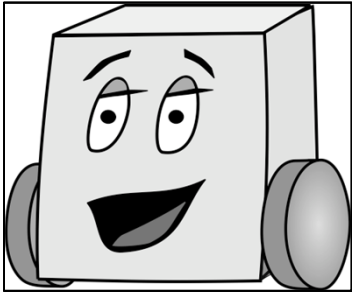
```
void setup()
{
}

void loop()
{
}
```

} This runs first
(sets things up)

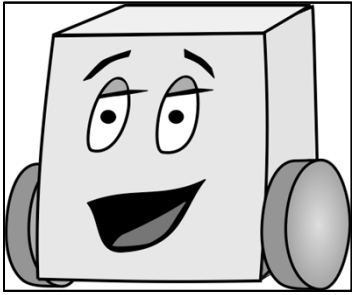
} Then this runs
repeatedly
(it loops)

All programs MUST contain these two functions



Simple C Program: Example

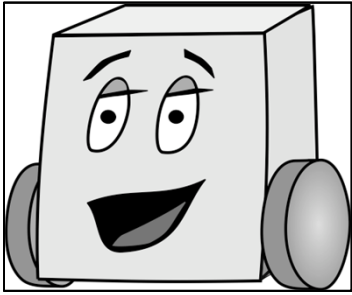
```
void setup()  
{  
  Serial.begin(9600);  
  Serial.println("Hello world!");  
}  
  
void loop()  
{  
}
```



Simple C Program: Example

Setup serial port to
run at 9600 baud
(bits/second)

```
void setup()  
{  
→ Serial.begin(9600);  
  
→ Serial.println("Hello world!");  
}  
  
void loop() ←  
{  
}
```



Simple C Program: Example

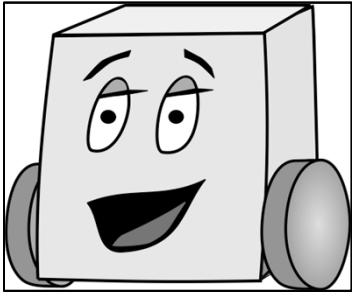
Setup serial port to
run at 9600 baud
(bits/second)

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

Print "hello world!" to
the serial port
(followed by a
carriage return)

```
  Serial.println("Hello world!");  
}
```

```
void loop() ←  
{  
}
```



Simple C Program: Example

Setup serial port to
run at 9600 baud
(bits/second)

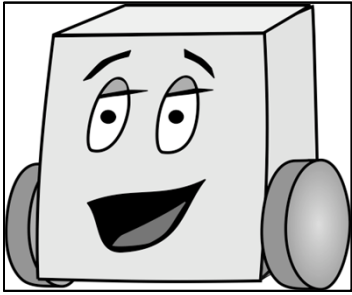
```
void setup()  
{  
  Serial.begin(9600);
```

Print "hello world!" to
the serial port
(followed by a
carriage return)

```
  Serial.println("Hello world!");  
}
```

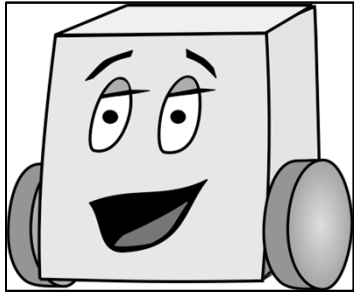
```
void loop() ←  
{  
}
```

In this program, the
loop() function does
nothing (but still must
be included!)



Running a program on the Arduino

- Run the Arduino software: *arduino.exe*
 - \\charlie.hmc.edu\Courses\Engineering\E11\fall2011\code\arduino-0022\arduino.exe
- Type the program into the *sketch*
- Save the file using a meaningful name – like “helloworld”
 - From the file menu: File -> Save As
 - The file will save with the .pde extension (helloworld.pde) and place it in a sketch folder
- Connect the Arduino board using an FTDI USB cable
- Change the settings to the correct device and port
 - From the file menu: Tools -> Board -> Arduino Duemilanove or Nano w/ Atmega328
 - Check the USB port settings with Tools -> Serial Port
- Verify the code
- Upload the code
- Open the Serial Monitor (after the code uploads)



Running a program on the Arduino

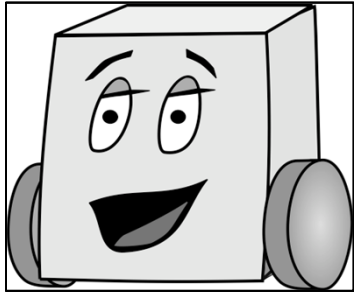
- Run the Arduino software: *arduino.exe*
- Type the program into the *sketch*

A screenshot of the Arduino IDE interface. The window title is "helloworld | Arduino 0022". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for play, stop, save, upload, download, and serial monitor. The main text area shows the following code:

```
// hello_world.pde
// 27 Aug 2011
// sarah_harris@hmc.edu

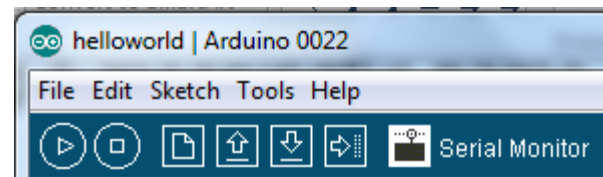
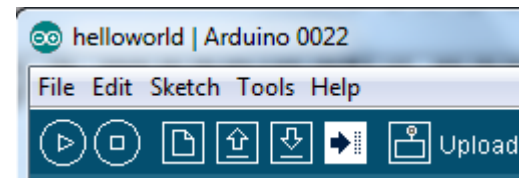
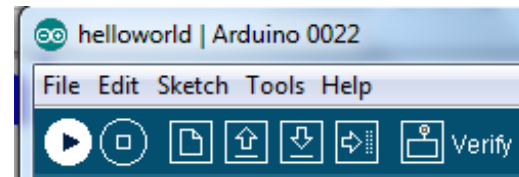
void setup()
{
  Serial.begin(9600);
  Serial.println("Hello world!");
}

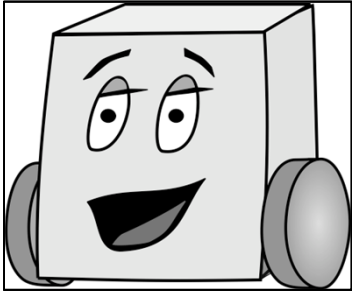
void loop()
{
}
```



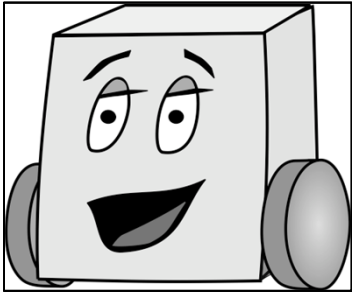
Running a program on the Arduino

- Save the file using a meaningful name – like “helloworld”
- Connect the Arduino board using a USB cable
- Change the settings to the correct device and port
- Verify the code
- Upload the code
- Open the Serial Monitor



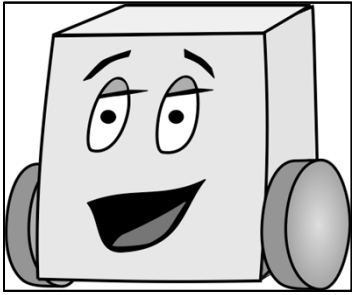


Demo



Coding: Your Turn!

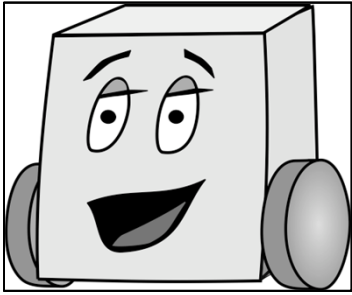
Write a program that repeatedly prints the phrase: "I love E11 already!"



Coding: Your Turn!

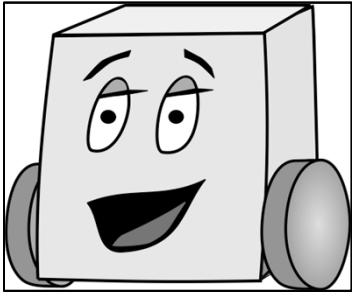
Write a program that repeatedly prints the phrase: "I love E11 already!"

```
void setup()  
{  
  Serial.begin(9600);  
}  
  
void loop()  
{  
  Serial.println("I love E11 already!");  
}
```



C Programming – Part 1

- What is C?
- Programming Target: Arduino
- Programming Basics
 - Simple C Program
 - Running a Program
- Programming Tools
 - Comments
 - Data Types
 - Variables
 - Console Inputs and Outputs
- More stuff you can do...



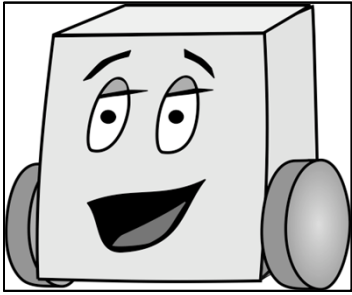
Comments

- Are ignored by the computer running the program
- But are **critical** for clarity and organization
- Single-line comment

```
// single-line comment
```

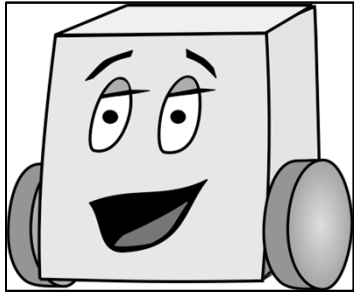
- Multiple-line comments

```
/* multiple-line  
comment */
```

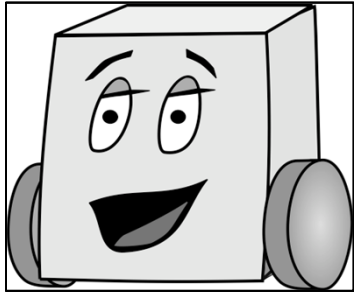
Data Types

- A data type tells us:
 - The **type** of values represented
 - The **range** of values



Data Types

Type	Size (bits)	Minimum	Maximum
char	8	-2^7 (-128)	$2^7 - 1$ (127)
unsigned char	8	0	$2^8 - 1$ (255)
int	16	-2^{15} (-32,767)	$2^{15} - 1$ (32,768)
unsigned int	16	0	$2^{16} - 1$ (65,535)
long	32	-2^{31} (-2,147,483,648)	$2^{31} - 1$ (2,147,483,647)
unsigned long	32	0	$2^{32} - 1$ (4,294,967,295)
float	32	$\pm 2^{-126}$	$\pm 2^{128} * (2 - 2^{-15})$
boolean	8	false	true



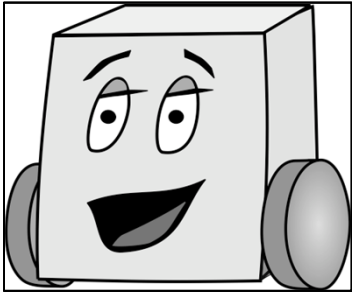
Data Types

Type	Size (bits)	Minimum	Maximum
char	8	-2^7 (-128)	$2^7 - 1$ (127)
unsigned char	8	0	$2^8 - 1$ (255)
int	16	-2^{15} (-32,767)	$2^{15} - 1$ (32,767)
unsigned int	16	0	$2^{16} - 1$ (65,535)
long	32	-2^{31} (-2,147,483,648)	$2^{31} - 1$ (2,147,483,647)
unsigned long	32	0	$2^{32} - 1$ (4,294,967,295)
float	32	$\pm 2^{-126}$	$\pm 2^{128} * (2 - 2^{-15})$
boolean	8	false	true

Note: byte = unsigned char

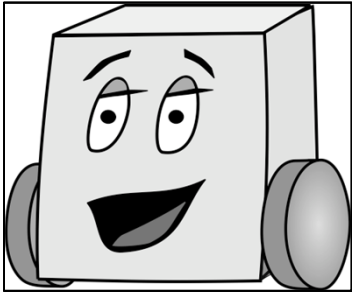
double = float

word = unsigned int



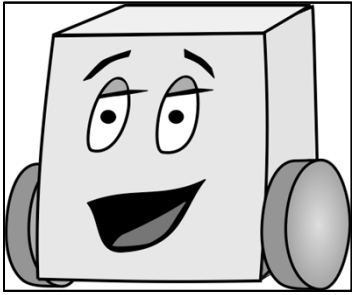
Binary numbers: range

- What happens when a result won't fit in that range?
 - Overflow!
 - For example, with only 2 bits: $11 + 01 = 100 = 00!$



Overflow Example

```
void setup()  
{  
  char x = 33;  
  char y = 257;  
  Serial.begin(9600);  
  
  Serial.print("The value of x is ");  
  Serial.println(x, DEC);  
  
  Serial.print("The value of y is ");  
  Serial.println(y, DEC);  
}  
  
void loop()  
{  
}
```



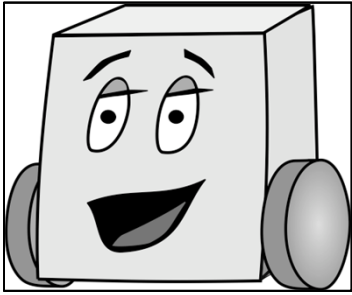
Variables

```
int cnt = 0;

void setup() {
  char x;
  float y = 7.8;
  boolean found = false;

  x = 12;
  ...
}

void loop()
{
}
```



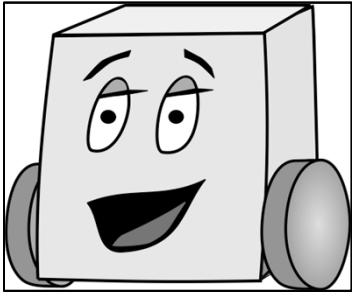
Variables

```
int count = 0; // global variable

void setup() {
  char x; // local variables
  float y = 7.8;
  boolean found = false;

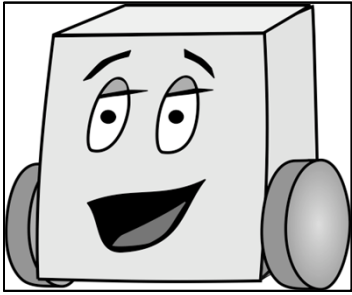
  x = 12; /* x is initialized
          after it is
          declared. */

  ...
}
```



Variables

- All variables must be *initialized* (set to a known value) before they are used
- Global variables:
 - are declared outside of all functions
 - are accessible anywhere in the program
- Local variables
 - are declared within a function
 - are only accessible within that function



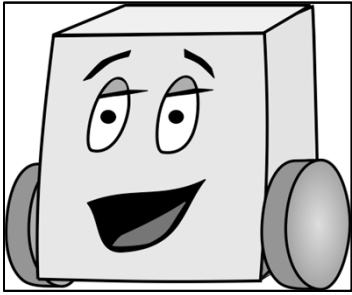
Variables

```
int cnt = 0;

void setup() {
  char x;
  float y = 7.8;
  boolean found = false;

  x = 12;
  ...
}

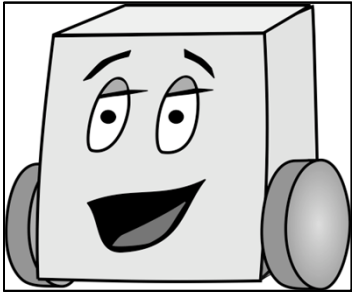
void loop()
{
  cnt = 42;
  x = 3;
}
```



Coding: Your Turn!

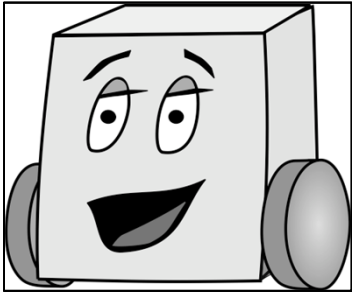
Write a program that converts the variable `x` from centimeters to inches and prints the value of `x` in both units.

```
// convert x from cm to in  
int x = 12;
```



Console Input and Output

- Output
 - `Serial.print(string or variable name);`
 - `Serial.println(string or variable name);`
- Input
 - `int Serial.read();`



Example: Console Input and Output

```
int incomingByte = 0; // incoming serial data

void setup()
{
  Serial.begin(9600); // opens serial port at 9600 baud
  Serial.println("Enter a value: ");
}

void loop() {
  // read user input
  if (Serial.available() > 0) {
    incomingByte = Serial.read();

    // print result:
    Serial.print("I received: ");
    Serial.println(incomingByte, BYTE);
  }
}
```

ASCII

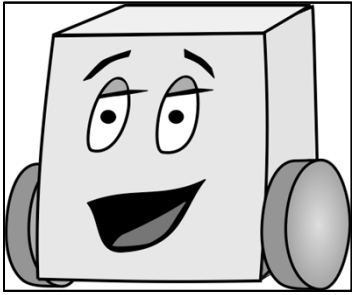
Binary	Octal	Decimal	Hexadecimal	Glyph
010 0000	040	32	20	space
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	\$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?

Binary	Octal	Decimal	Hexadecimal	Glyph
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D]
101 1110	136	94	5E	^
101 1111	137	95	5F	_

Binary	Octal	Decimal	Hexadecimal	Glyph
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	
111 1101	175	125	7D	}
111 1110	176	126	7E	~

In my eyes,
that should
be spelled
ASCIII





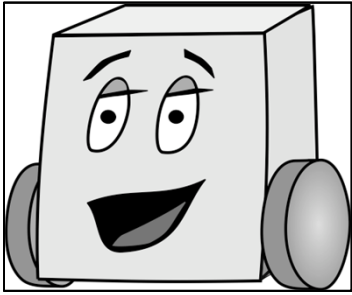
Example: Console Input and Output

```
int incomingByte = 0; // incoming serial data

void setup()
{
  Serial.begin(9600); // opens serial port at 9600 baud
  Serial.println("Enter a value: ");
}

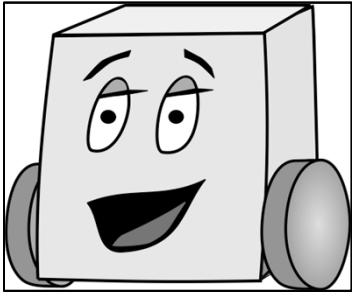
void loop() {
  // read user input
  if (Serial.available() > 0) {
    incomingByte = Serial.read();

    // print result:
    Serial.print("I received: ");
    Serial.println(incomingByte, DEC);
  }
}
```



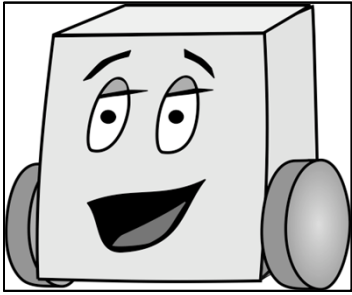
Physical Inputs and Outputs

- Add picture of Arduino



Physical Inputs and Outputs

- Setup:
 - `pinMode(pin, mode)`
 - *mode* is either: INPUT or OUTPUT
- Output – setting a pin value:
 - `digitalWrite(pin, value)`
 - *value* is either: HIGH or LOW
- Input
 - `digitalRead(pin)`



Physical Output: LED

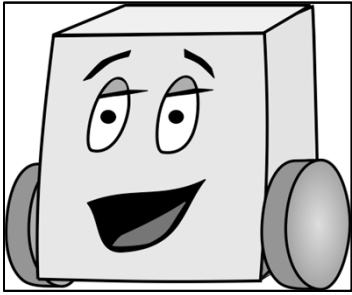
```
void setup()
{
  Serial.begin(9600);

  // set LED pin as output
  pinMode(13, OUTPUT); // LED pin
}

void loop()
{
  Serial.println("Testing LED");

  digitalWrite(13, HIGH); // turn the LED on
  delay(200); // delay 200 ms

  digitalWrite(13, LOW); // turn the LED off
  delay(200); // delay 200 ms
}
```



Physical Output: Speaker

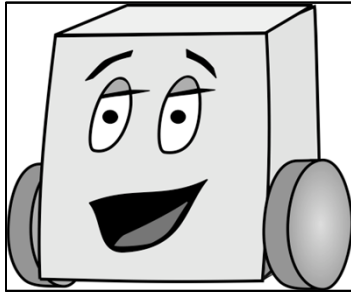
```
void setup()
{
  Serial.begin(9600);

  // set speaker pin and LED as outputs
  pinMode(4, OUTPUT); // speaker pin
  pinMode(13, OUTPUT); // LED pin
}

void loop()
{
  Serial.println("Testing speaker");

  tone(4, 440); // write tone of 440 Hz to speaker
  digitalWrite(13, HIGH); // turn the LED on
  delay(200); // delay 200 ms

  noTone(4); // turn the speaker (pin 4) off
  digitalWrite(13, LOW); // turn the LED off
  delay(200); // delay 200 ms
}
```



Useful Resource!!!

<http://arduino.cc/en/Reference/HomePage>

The screenshot shows a web browser window titled "Arduino - Reference" with the URL <http://arduino.cc/en/Reference/HomePage>. The page features a navigation menu with links for Buy, Download, Getting Started, Learning, Reference, Hardware, and FAQ. The main content area is titled "Language Reference" and provides an overview of Arduino program structure, variables, and functions. It is organized into three columns: Structure, Variables, and Functions.

Structure

- [setup\(\)](#)
- [loop\(\)](#)

Control Structures

- [if](#)
- [if...else](#)
- [for](#)
- [switch case](#)
- [while](#)
- [do... while](#)
- [break](#)
- [continue](#)
- [return](#)
- [goto](#)

Further Syntax

- [;](#) (semicolon)
- [{ }](#) (curly braces)
- [//](#) (single line comment)
- [/* */](#) (multi-line comment)
- [#define](#)
- [#include](#)

Arithmetic Operators

Variables

Constants

- [HIGH](#) | [LOW](#)
- [INPUT](#) | [OUTPUT](#)
- [true](#) | [false](#)
- [integer constants](#)
- [floating point constants](#)

Data Types

- [void](#)
- [boolean](#)
- [char](#)
- [unsigned char](#)
- [byte](#)
- [int](#)
- [unsigned int](#)
- [word](#)
- [long](#)
- [unsigned long](#)
- [float](#)
- [double](#)
- [string](#) - char array
- [String](#) - object
- [array](#)

Functions

Digital I/O

- [pinMode\(\)](#)
- [digitalWrite\(\)](#)
- [digitalRead\(\)](#)

Analog I/O

- [analogReference\(\)](#)
- [analogRead\(\)](#)
- [analogWrite\(\)](#) - *PWM*

Advanced I/O

- [tone\(\)](#)
- [noTone\(\)](#)
- [shiftOut\(\)](#)
- [pulseIn\(\)](#)

Time

- [millis\(\)](#)
- [micros\(\)](#)
- [delay\(\)](#)
- [delayMicroseconds\(\)](#)

Math

- [min\(\)](#)