Arduino 1: About Arduino

1. Download the Arduino software from http://www.arduino.cc
2. Install it and run it
3. Plug the Arduino into your computer

The Arduino is a small computer. The main chip on the Arduino circuit board is called a microcontroller. The microcontroller is a full computer on a chip with a CPU, memory, and other features all built into that one chip. Microcontrollers are used for small electronic devices like clocks, calculators, traffic lights, and microwaves.

Writing programs for a microcontroller is known as embedded programming. Embedded programming is different from normal programming because

- the system only does one thing and is not supposed to be used like a general computer
- the system does not have a screen, keyboard, mouse, or operating system, so you can't program directly on the system. You need to write your programs on a separate computer
- microcontroller chips are usually less powerful than a full computer but your programs have direct access to the hardware

This is the process that happens when you program an Arduino

a) You write a program on a normal computer

b) The program is translated into machine instructions that can be understood by the Arduino microcontroller

c) The machine instructions are transferred by USB to the Arduino's Read-Only Memory (ROM)

d) The Arduino is reset and restarted

e) When the Arduino starts up, it will run the program that was copied to its ROM

Once your program is copied to the Arduino, it no longer needs to be connected to your computer. Remember that the Arduino is a full computer in itself. You can plug the Arduino into a power source (like a battery or USB charger), and it will run the program that was stored in its ROM by itself.

Arduino 2: Getting Started

1. In the Arduino software, go to the Tools...Board menu and set the type of Arduino to match the Arduino circuit board that you have

2. Go to the Tools...Port menu and set it to COM3. Although COM3 usually works, different computers require different settings. If step 5 doesn't work for you, you may need to try a different port setting.

3. Type the following program into the Arduino software

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

4. This is what the program does:

   a) When the Arduino starts up, ...

   b) The Arduino will configures itself so that it can send messages along the usb cable back to the main computer

   c) It will send the message "hello"

5. Choose the Sketch...Upload menu to compile your program and upload it to the Arduino (you can also click on the right arrow button at the top of the Arduino window).

6. Choose the Tools...Serial Monitor menu to see the messages going between your computer and the Arduino. You should see a "hello" message there from the Arduino

7. Hit the reset button on the Arduino circuit board to restart the program. A new "hello" message should appear in the Serial Monitor

Arduino 3: LCD Keypad Shield

1. Unplug the Arduino
2. Let's plug some hardware into the Arduino computer so that the computer will do something. Find a LCD Keypad Shield and plug it into the Arduino. You may have to push and pull the pins or the circuit board to get everything to line up, but make sure you don't break any of the pins.
3. Plug the Arduino back into your computer
4. The LCD Keypad Shield has a small screen and some buttons. Lets have the Arduino show something on the screen when it starts up. We'll use a `lcd` code library to manage the lcd interface for us. Type up the program below:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    lcd.begin(16, 2);

    lcd.setCursor(0,0);
    lcd.print("Hello");
}
```

5. The code in italics is needed to setup the `lcd` library. The other code is for controlling what's printed on the screen

`lcd.setCursor(col, row)`	*Choose where to print things*
`lcd.print(data)`	*Show something on the screen*
`lcd.clear()`	*Clear the screen*

6. When you run the program, it will say "hello" when it starts up
7. Remember that the Arduino is a separate computer. Once you have uploaded the program into your Arduino, you can actually unplug the Arduino from your computer and then plug it into a battery or a usb charger, and it will show "hello" itself.

Arduino 3: LCD Keypad Shield as a Timer

1. Now lets get the Arduino to act as a timer. The `millis()` command tells you how many milliseconds have passed since the Arduino started
2. Lets get the Arduino to show that on its screen

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    lcd.begin(16, 2);
}

void loop() {
    lcd.setCursor(0, 0);
    lcd.print("Time is");
    lcd.print(millis());
}
```

3. Can you get the Arduino to show the amount of time in seconds instead of in milliseconds? There are 1000 milliseconds in a second.
4. Can you change the message to show the number of minutes and seconds that have passed?

```
Time is 0m 32s
```

You might find it useful to use the / and % instructions:

`7 / 3 = 2`	Division
`7 % 3 = 1`	Remainder after dividing (or modulo)

Arduino 4: LCD Keypad Shield Buttons

The LCD Keypad Shield has buttons on the bottom.

These buttons are connected to the analog input of the Arduino. The analog input lets the Arduino measure the "level" of something. If you hook up different sensors, it can tell you the level of sound or the level of light in a room.

The buttons on the LCD Keypad Shield are setup so that the Arduino reads different levels when different buttons are pressed.

1. Here is some code for showing what levels are pressed:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    lcd.begin(16, 2);
}

void loop() {
    lcd.setCursor(0, 0);

    int level = analogRead(0);
    lcd.print(level);

    lcd.print("   ");
}
```

2. Upload the program to the Arduino. Try pressing the different buttons (make sure you don't press the RST or reset button) and see what levels the Arduino sees.

Arduino 5: LCD Keypad Shield Button Detection

On the previous worksheet, we saw that the Arduino reads different levels on its analog input when different buttons are pressed.

Although the exact values are different for each Arduino, the levels for the different buttons should be in this range:

Button	**Range**	***Range (for v1.1 Keypad Shield)***
Right	0-50	*0-50*
Up	50-195	*50-250*
Down	195-380	*250-450*
Left	380-555	*450-650*
Select	555-790	*650-850*
No button	790-	*850-*

With this information, you can get the Arduino to tell you which button was pressed.

1. Here is a program that detects when the Select button is pressed

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup() {
    lcd.begin(16, 2);
}

void loop() {
    lcd.setCursor(0, 0);

    int level = analogRead(0);
    lcd.print(level);

    if (level > 555 && level < 790) {
        lcd.print(" SELECT");
    }

    lcd.print("   ");
}
```

2. Can you write a program that detects when some of the other buttons are pressed?

Arduino 6: LCD Keypad Shield Press any Key

1. Now that we know how to work with buttons, we can write programs that let the buttons do things. Let's start with a simple program that waits for you to press a button:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 4, 5, 6 , 7);

void setup() {
    lcd.begin(16, 2);

    lcd.setCursor(0, 0);
    lcd.print("Press a button");

    while (true) {
        int level = analogRead(0);
        if (level < 790) {
            break;
        }
    }

    lcd.clear();
    lcd.print("Ok");
}

void loop() {
}
```

2. The `while(true)` command in the middle makes the Arduino repeat the instructions inside until it sees a `break` command. The instructions inside will check the buttons. When a button is pressed, it will `break` and exit.

Arduino 7: LCD Keypad Shield Stopwatch

1. Here is some code for showing how much time has passed since the code was started:

```
unsigned long start = millis();
lcd.clear();
while (true) {
    lcd.setCursor(0, 0);
    lcd.print(millis() - start);
}
```

2. Can you combine the code with the button code from the previous worksheet so that when you press a button, the Arduino tells you how much time has passed?

3. Can you make the display nicer so that it shows you the number of seconds and minutes that have passed?

4. Can you program a button so that it stops the timer?

   TIP: When you see that one of the buttons has been pressed, you should then wait until the button is released before doing anything else. Otherwise, later on, your program might think the button has been pressed a second time even though it's still the first time it has been pressed

5. Can you program a button so that it resets the timer back to 0?

SUPER ADVANCED:

- Can you make a full clock that lets you set the time?
- Can you make a game with the screen?
- Can you make a 2-player game with the screen?