Name/NetID:	Teammate:

Module: Introduction to Using the Arduino

Module Outline

In this module you will be introduced to the microcontroller board included in your kit. It is manufactured by Sparkfun and is called the RedBoard for obvious reasons. It is a clone which means that it behaves the same way that the popular Arduino Uno works. All of the inputs and outputs behave the same in function though the electrical interface of the RebBoard is less robust than Arduino making it easily interfaced to fewer devices.

Arduino Hardware

As an introduction to the hardware the figure below shows your RedBoard (right) compared to an older version of the Arduino Uno. Price is often the reason for choosing a clone over the original, but the Arduinos are typically made of more robust connectors and the microprocessor is in a socket so it can be removed for some applications. An in-depth comparison can be found at the Sparkfun webpage https://learn.sparkfun.com/tutorials/redboard-vs-uno?_ga=1.150963894.1068174368.1405368730 .

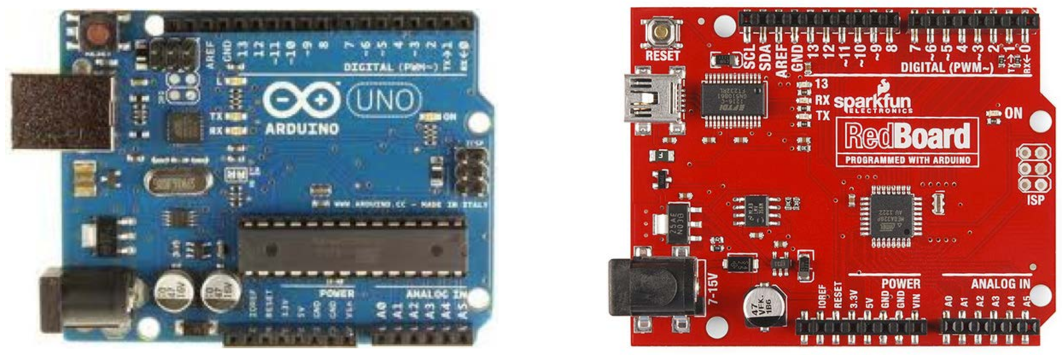


Figure 1: Arduino compared to Sparkfun's RedBoard

Notes:

In the margin is the figure that Sparkfun uses to highlight the parts of the board that most users need to know about to use the board effectively. Most of the listed parts are those directly accessible to the casual user.

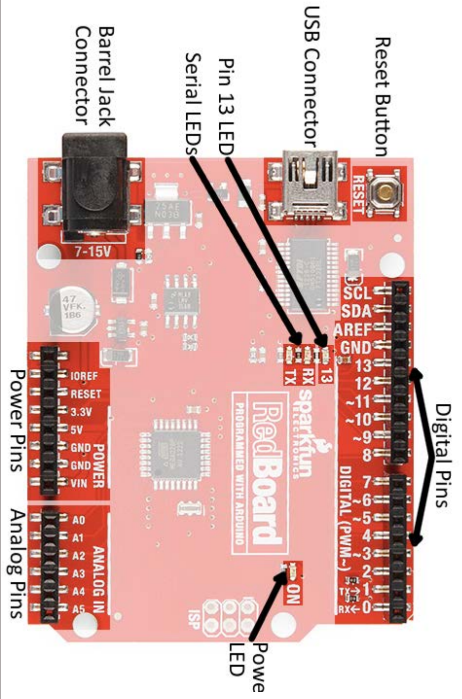
Hardware Interface Pins to External Circuitry

Digital Pins	Pins that can be configured as inputs or outputs. Digital means that the voltage measured can only have two values 0V or 5V. Some of the pins have additional functionality described in a later portion of the lab.
Analog Pins	Pins that input a continuous voltage that the onboard processor digitizes which means that the voltage range 0-5V is mapped onto integer values of 0-1023.
Power Pins	Pins that can be used to power the Arduino board and can source a regulated 3.3V and 5V. Several GND pins are provided.

Useful Board Features

Reset Button	This button interrupts the program running on the board and restarts it from the beginning.
USB Connector	This is a mini USB port that is used to connect the board to a USB port on a computer. This is the interface by which the programs loaded into the microcontroller are downloaded. Power is also provided through this port.
Pin 13 LED	An LED that is connected to pin 13 – if the LED is on then the voltage at pin 13 is 5V if it is off the voltage is 0V. This a useful debugging feature. But only 1???
Serial LEDs	The serial LEDs indicate the traffic on the receive and transmit pins. The computer and many other devices communicate to the board serially. As you upload programs to the board or communicate to the serial port using coded statements you can see the traffic as these pins flicker when data is being transmitted and received.
Barrel Jack Connector	This jack is used to connect a battery to provide power to the board so that the board can be disconnected from the computer.
Power LED	Indicates that the board has power.

But you are not the casual user. So let's look more closely at what the Arduino is and how to use it properly in an engineering setting.



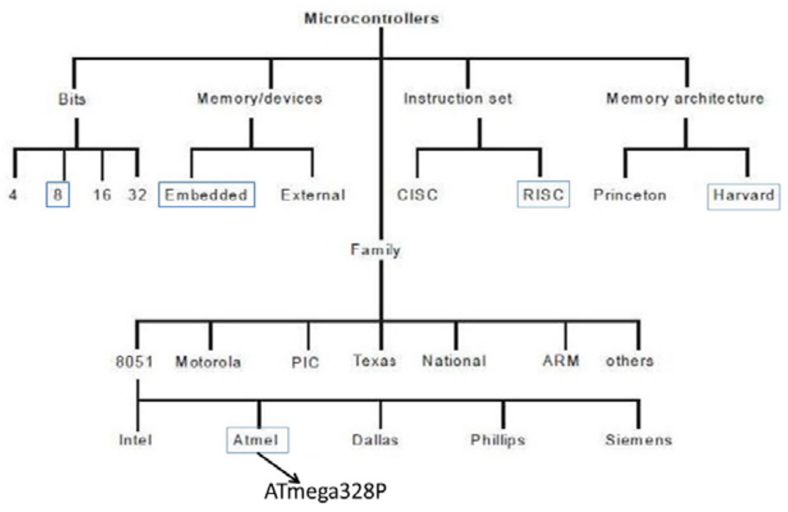
Notes:

The Embedded Processor

Most engineering products, even the humble light bulb, now comes with a computer of some sort *embedded* in the hardware. From the comfort of your bed you can remotely turn on and off your lights. Household appliances like your oven and refrigerator have not had hardware switches on them for decades. Nearly everything has a computer in it. How did the engineers who designed and built these products develop them to include these ubiquitous digital devices? The design of the analog part of the products is more obvious if you are skilled. Gather the parts, connect them together, and test them until they work properly. Then make a bunch of them. The presence of the microprocessor introduces a whole new level of complexity. It has many inputs and outputs that need interfacing, and how are you going to tell it what to do. It is just a chip – how do you put a program into its memory and start the process of running the program?

A microprocessor was included in your kits. You most likely bought an Arduino Uno or its clone the RedBoard from SparkFun. The Arduino is powered by the ATMega328P microprocessor. The figure below shows how this processor fits into the taxonomy of microprocessors. As you enter into your own designs you will become more familiar with the different types and will know what all the designations mean. The ATMega328 is produced Atmel Semiconductors. The processor uses an architecture based on a word length of 8-bits, it runs at a clock speed of 16Mhz, the memory is embedded onboard the processor itself, and it follows the more elegant RISC approach. The Reduced Instruction Set (RiSC pronounced “risk”) processor uses a simple architecture based on a few well-chosen assembly level instructions. Thereby, relying on speed and flexibility to perform more complex functions.

Types of Microcontrollers



Notes:

Question 1: Research online or in the library the distinction between the Princeton and Harvard architectures.

Evaluation vs. Development

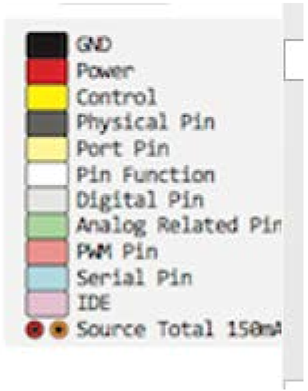
In the parlance of electrical and computer engineers, the Arduino Uno is a *development board* and the Sparkfun RedBoard is an *evaluation board*. The *evaluation boards* are usually cheaper and used to test the suitability of a particular microprocessor for a given application. The board provides an interface so that the microprocessor can be programmed easily and interfaced to hardware for fast proto-typing. The development board can be used to evaluate the microprocessor and once the hardware and software design are stabilized the processor chip can be removed and embedded in a more permanent proto-type. NOTE: If you look at all of the header pins both input and output they are connected DIRECTLY to the processor chip. With very few pieces of external circuitry – notably the voltage regulator and the clock – you can embed a fully programmed ATMega328P microprocessor into your hardware project.

Hardware Reference

The figure below summarizes the functionality of the Arduino Uno's interface. As stated in the title it is the definitive summary of the pinouts of the board and how they are connected to the microprocessor chip itself (sub-figure in the upper left corner). Once you learn to interpret the labeling it will become the only hardware reference you need.

The initial reaction to this figure is horror until it becomes clear that each pin on the board is connected directly to the processor chip and that some of the processor pins (and hence the pins on the board) can have several uses. For example, pins 0-13 as labeled on the board itself and echoed in the pink boxes are *digital input/output* (I/O) pins. This means that you the programmer must specify in your code the state of the I/O pins that you use as either *input* or *output*. You will come to understand this better as you work with the board.

The secret to understanding this diagram resides in the legend at the bottom (shown at right). The colors of each pin label indicates its functionality and how the pin is referenced in different circumstances.



Notes:

Question 2: Look at the labels in the grey boxes. These are the numbers of the physical pins on the microprocessor chip (ATMega328) that each pin is connected to. Which physical pin numbers are associated with the pins that are capable of providing a PWM signal? NOTE: you can get this information both from the diagram of the board layout and the smaller one of the microprocessor layout.

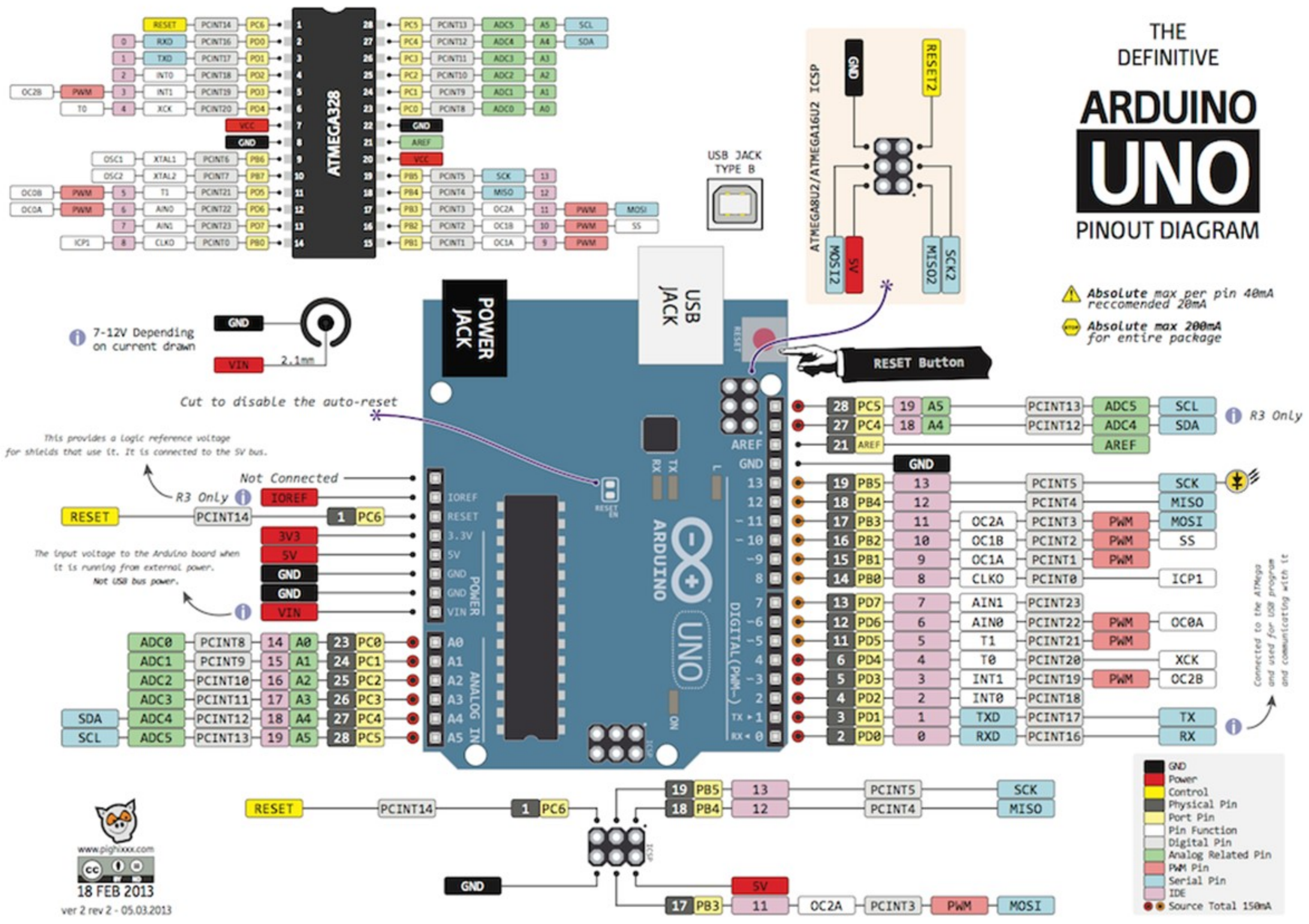
Question 3: If you interface to the board through the Arduino Integrated Development Environment (IDE) – the C-like language used by most casual users of the Arduino, the only pins designations that you will use are those in the pink boxes and those written on the board itself. How are the Analog Input pins identified when programming with the Arduino IDE?

The other pin labels are associated with functionality for the more advanced user. The yellow labels show the pin names used when identifying the pins as ports. The processor can be programmed at a lower level (assembly) where the inputs and outputs can be read and manipulated more quickly using the port designations rather than the designations used in the Arduino IDE.

Question 4: For the observant there are also rules of thumb for the amount of current that can be drawn from each pin and the entire processor chip. What are these suggested limits? Note: more current can be drawn from the battery through the V_{in} pin which is drawn directly from the battery completely bypassing the processor so there is a separate set of board level current limits.

Notes:

Question 5: The light blue labels are associated with the pins that can be configured to provide *serial communication* to and from a connected computer or other hardware with a serial interface. Do some research to find out what this means. It is a general interface that has been around from the beginning. It is the protocol and hardware interface used by the USB connection from the PC to the board.



Programming the Arduino Board

Arduino Development Environment – C-like programming environment

The Arduino Development Environment is the interface that you will use to interface with the board. To access this environment through a free program named ***Arduino*** and some driver software. There are several resources that are available to guide you through downloading the software onto your own computer.

Sparkfun Tutorial: https://learn.sparkfun.com/tutorials/redboard-hookup-guide

Arduino Tutorial: http://arduino.cc/en/Main/Software

Judy Culkin's Graphic Novel Turorial: http://www.jodyculkin.com/wp-content/uploads/2014/03/arduino-comic-2014.pdf

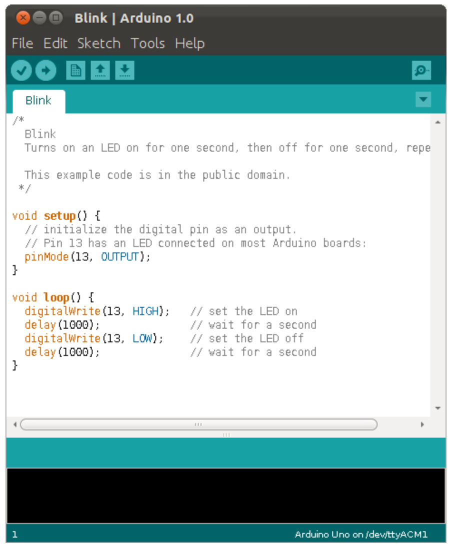
Washington DC's Hackerspace Tutorial: http://www.hacdc.org/summer-school-2013/

In the lab all of the computers have the Arduino software installed but it is recommended that you also have a copy on your own computer so you can work from home.

Arduino Libraries

The Sparkfun kit comes with a manual that includes 10 experiments that guide you through learning to build simple circuits, connecting them to the Arduino and learning simple programming. The software used in each experiment is provided and can be downloaded from the URL http://sparkfun.com/SIKcode . A zipped file will be downloaded into you Downloads folder. To have the library contents available inside the Development environment simple run the Arduino program. Find the menu item *Sketch* and inside the drop-down menu at the very top should be an item *Add Library…* Clicking on this item will bring up a window asking for you to find the library you wish to include. There is an enormous Arduino community and lots of free libraries. These instructions work for any library not just the Sparkfun library.

Notes:

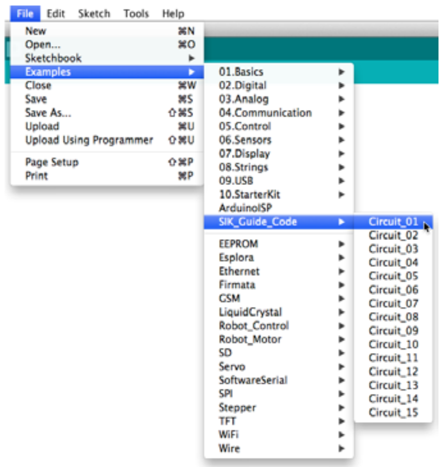


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```



Notes:

Assembly Language Programming

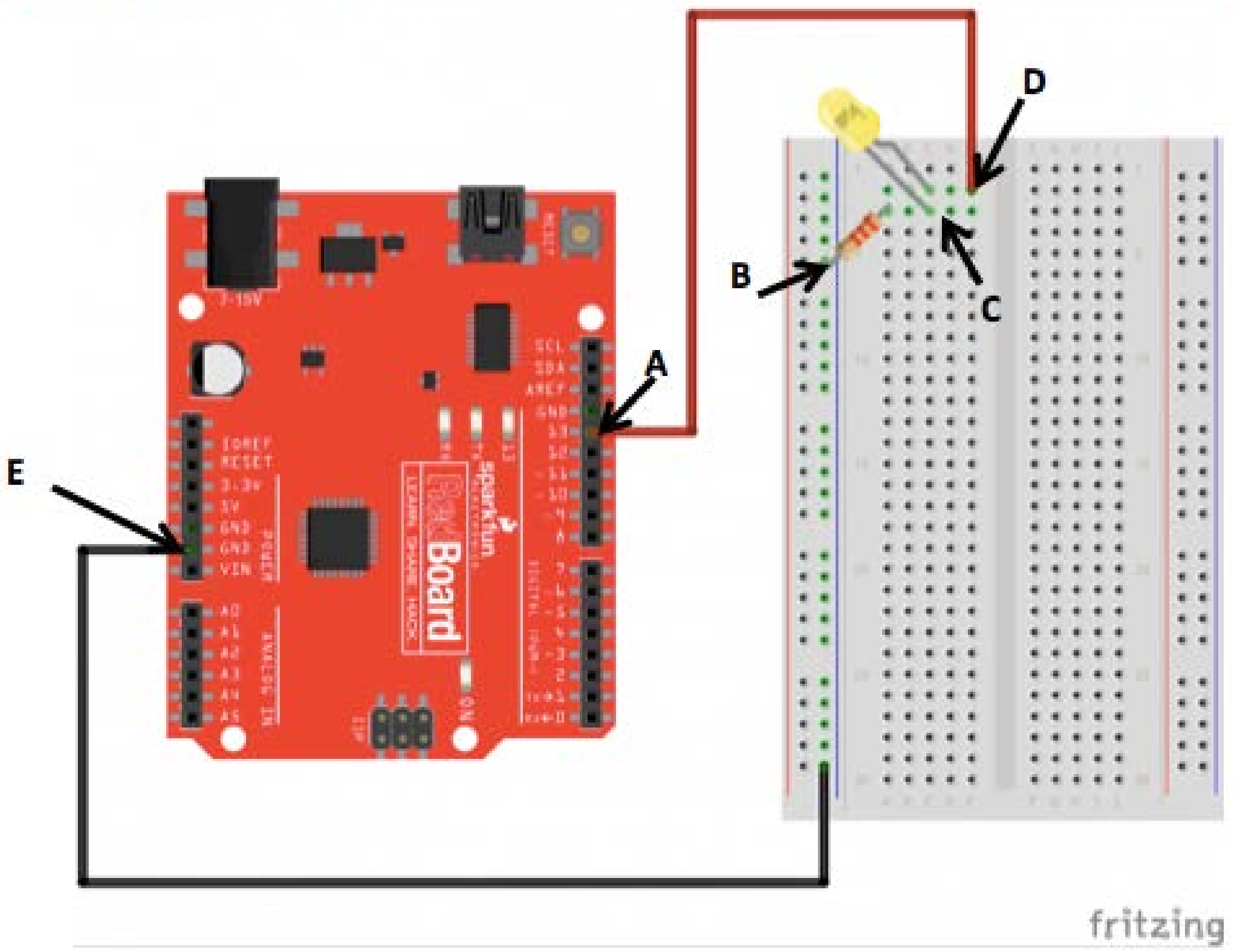
Code can be written in C or C++ and using the appropriate assembler convert the code to something that will run on the Arduino. Or you can write assembly code directly using an IDE like AVRDude that can be downloaded from the web.

Notes:

Uploading and Running code throught the Arduino IDE

Build the simple circuit below.

Fritzing Diagram for RedBoard



Notes:

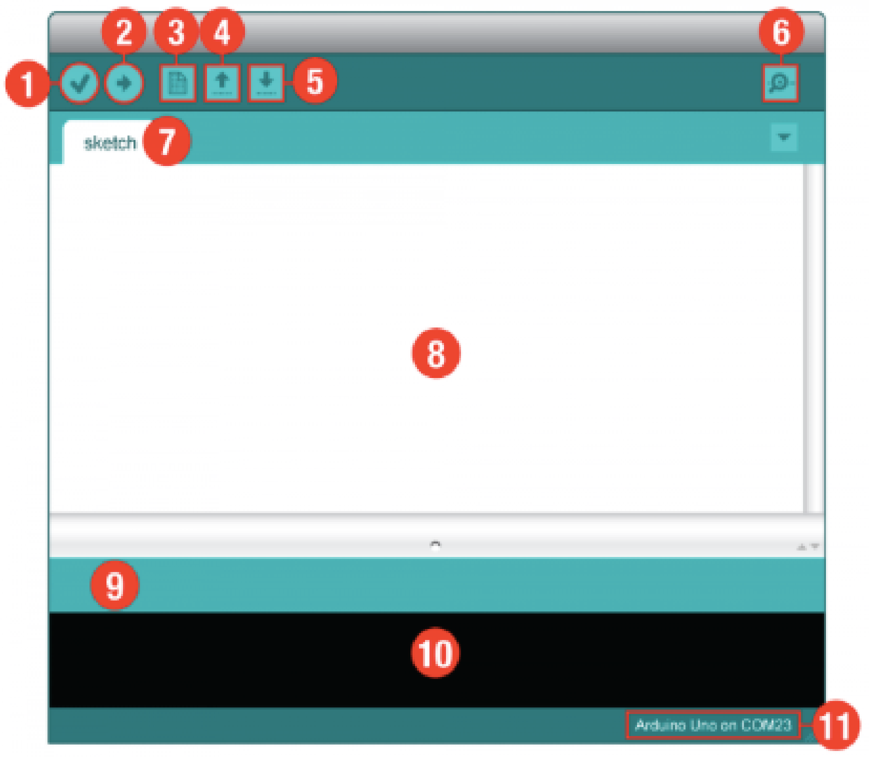
Some of you have already done this experiment so this may be a review.

Question 6: From the physical diagram draw a schematic of the circuit. Rather than draw the Arduino you may simply label the signals connected to the Arduino – one connection is made to the **GND** pin on the board and the other is made to the digital I/O **Pin 13**. Either way designate on the schematic the points labeled **A**, **B**, **C**, **D**, and **E.**

To make the LED blink you need to have a program running on the Arduino board that raises and lowers the voltage on Pin 13.

- ✓ Run the Arduino program on the computer that will be used to download the program and power the board.
- ✓ To find the program navigate the menu items File > Examples > 01.Basics > Blink and a window will open up with the software needed to blink the LED by using the Digital Pin 13 as an output.
- ✓ Attach the cable in your kit that has a mini USB connector on one end. The mini USB is connected to the board the other end goes into any available USB port on the computer.
- ✓ Upload the program to the board by clicking the upload button (labeled 2 in the diagram below). *NOTE: Common reasons for this step to fail are the Arduino does not know which of the many boards you are using or the COM port (USB port you plugged the board into) is incorrect. Both problems can be addressed by looking in the Tools option in the menu bar.*

The IDE interface

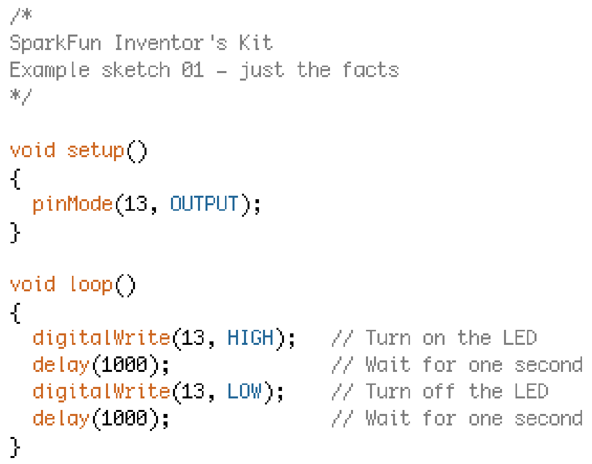


1. **Verify:** Compiles and approves your code. It will catch errors in syntax (like missing semi-colons or parenthesis).
2. **Upload:** Sends your code to the RedBoard. When you click it, you should see the lights on your board blink rapidly.
3. **New:** This buttons opens up a new code window tab.
4. **Open:** This button will let you open up an existing sketch.
5. **Save:** This saves the currently active sketch.
6. **Serial Monitor:** This will open a window that displays any serial information your RedBoard is transmitting. It is very useful for debugging.
7. **Sketch Name:** This shows the name of the sketch you are currently working on.
8. **Code Area:** This is the area where you compose the code for your sketch.
9. **Message Area:** This is where the IDE tells you if there were any errors in your code.
10. **Text Console:** The text console shows complete error messages. When debugging, the text console is very useful.
11. **Board and Serial Port:** Shows you what board and the serial port selections

Notes:

Notes:

Question 7: Look at the Arduino Code and all of the lovingly written comments that explain how to build the circuit, connect it to the processor board, run the 'sketch', and a short explanation of the structure of the code. Please read them – below is a condensed version without the comments so you can see more clearly the code structure. Using the information in the comments or on the Arduino website explain the function of the ***setup*** and ***loop*** portions of the code.



```
/*
SparkFun Inventor's Kit
Example sketch 01 - just the facts
*/

void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, HIGH);   // Turn on the LED
  delay(1000);              // Wait for one second
  digitalWrite(13, LOW);    // Turn off the LED
  delay(1000);              // Wait for one second
}
```

Congratulations!! You are ready to start using the Arduino in a project!