Arduino

What is it all about?

What is Arduino?

- Arduino is a tool for making computers that can sense our world and control devices.
- It is an open-source physical computing platform based on a simple microcontroller board.
- It is a development environment for writing software for an Arduino board.
- The boards can be assembled by hand or purchased preassembled.
- The open-source IDE can be downloaded for free. Latest version is v1.6.6
- http://arduino.cc/en/Main/Software

Where does Arduino come from?

- Arduino started in 2005 as a project for students at the Interaction Design Institute Ivrea in Ivrea, Italy. At that time program students used a "BASIC Stamp" at a cost of $100, considered expensive for students. Massimo Banzi, one of the founders, taught at Ivrea.[4] The name "Arduino" comes from a bar in Ivrea, where some of the founders of the project used to meet. The bar, in turn, has been named after Arduin of Ivrea, who was the margrave of Ivrea and king of Italy from 1002 to 1014.

Why Arduino?

- Inexpensive – Lots of Arduino boards are available for between $10 ~ $50.
- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers.
- The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to C programming language on which it's based.

Getting started with Arduino?

- Software projects in Arduino are called **Sketches**.
- **Sketches** are written in a text editor window.
- The console displays text output by the Arduino environment including complete error messages and other information.
- The bottom right-hand corner of the window displays the current board and serial port.
- The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

What are these Sketches?

- The Arduino environment uses the concept of a sketchbook: a standard place to store your programs (or sketches).
- The first time you run the Arduino software, it will automatically create a directory for your sketchbook.
- You can view or change the location of the sketchbook location from with the **Preferences** dialog.

Uploading an Arduino program?

- Before uploading your sketch, you need to select the correct items from the **Tools > Board** and **Tools > Serial Port** menus.
- Once you've selected the correct serial port and board, press the upload button in the toolbar or select the **Upload** item from the **File** menu.
- On most boards, you'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino environment will display a message when the upload is complete, or show an error.

What is the Arduino Bootloader?

- When you upload a sketch, you're using the Arduino **bootloader**, a small program that has been loaded on to the microcontroller on your board.
- The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller.
- The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

What are Arduino Libraries?

- Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data.
- To use a library in a sketch, select it from the **Sketch > Import Library** menu.
- This will insert one or more **#include** statements at the top of the sketch and compile the library with your sketch.
- There is a list of libraries in the reference. Some libraries are included with the Arduino software.

What is the Serial Monitor?

- Displays serial data being sent from the Arduino board (USB or serial board).
- Choose the baud rate from the drop-down menu that matches the rate passed to **Serial.begin** in your sketch.
- To send data to the board, enter text and click on the "send" button or press enter.

What Arduino boards are available?

- The board selection has two effects: it sets the parameters (e.g. CPU speed and baud rate) that are used when compiling and uploading sketches.
- It also sets the file and fuse settings used by the bootloader command.
- Arduino boards are available from many sources; locally you can buy them from **Microcenter on Metcalf Avenue and 93rd Street**.

What are the Core Functions?

- Simple programs that demonstrate basic Arduino commands.
- To open them, click the Open button on the toolbar and look in the **examples** folder.
- Basics, Digital, Analog, Communication, Control Structures, Sensors, Display, Strings, USB, Keyboard, and Mouse functions are core functions provide in the IDE.

What are Libraries?

- Examples from the libraries that are included in the Arduino software.
- EEPROM, Esplora (for Esplora board beginners), Ethernet, Firmata, GSM, LCD display, Robot, SPI, Servo, Software Serial (serial port emulation), Stepper, TFT display, Wire, Wi-Fi are the libraries provided with the IDE. Some of them require specific hardware boards added to the system.

Project Examples for Arduino

- Fan controller with temperature sensor and variable control
- Battery monitor and recharge controller
- Morse Code Encoder or Decoder
- Antenna Tuner or Switch Controller
- Antenna Rotator Controller
- ID Timer that counts up or down
- CW Beacon controller and keyer
- Temperature Reader and/or controller
- Light controller with pot control or wireless control
- Light controller for dimming/occupancy/wireless
- Serial port controller/display unit
- Compass/Altimeter/Elevation reader
- Lightning Detector
- LCD display controller w/ other functions
- HF radio remote head/controller with keys

First Arduino sketch?

- What does the sketch **have** to contain.

```
void setup() {
  // put your setup code here, it will run only once.
}

void loop() {
  // put your main code here, to run repeatedly.
}
```

What is Setup()?

- The setup() function is called when a sketch starts.
- Use it to initialize variables, pin modes, start using libraries, etc. The setup function will run only once, after each power-up or reset of the Arduino board.

What is Loop()?

- The loop() function does precisely what its name suggests, and loops continuously.
- Allows your program to change and respond as it runs.
- Code in the loop() section of your sketch is used to actively control the Arduino board.
- Any line that starts with two slashes (**//line comment**) will not be read by the compiler, so you can write anything you want after it.

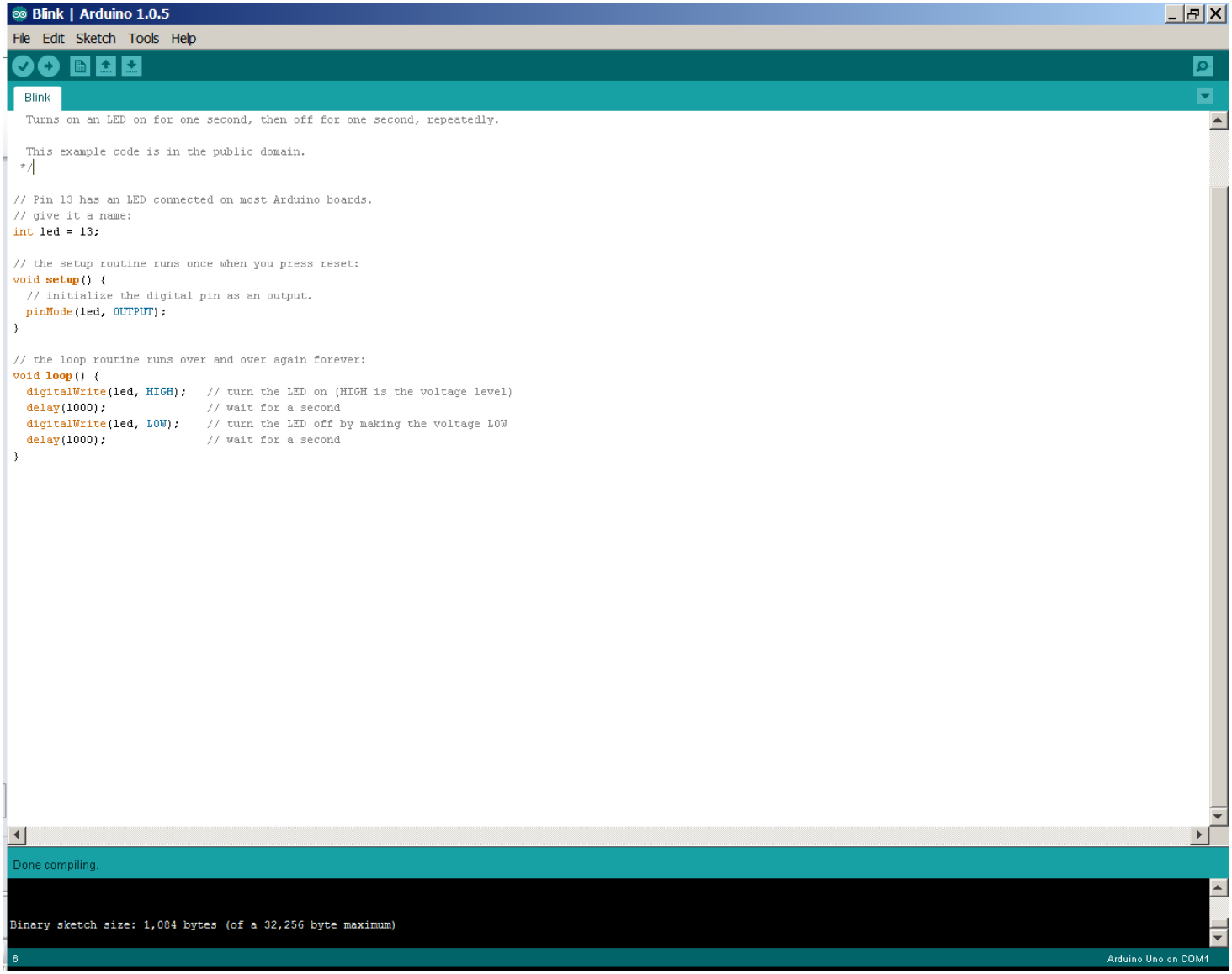
First program

```
/* -  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.  */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for 1,000 milliseconds  or 1 second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for 1,000 milliseconds  or 1 second
}
```

First program - functions?

- Int led = 13; // this defines which pin is connected to the LED that we are going to flash.
- pinMode(led, OUTPUT); // this makes the pin an output so that we can control the LED state.
- digitalWrite(led, HIGH); // this sets the state of the output pin to HIGH. 5V will be on pin after this command. Also, LOW will be at 0V.
- Delay(1000); // wait for 1,000 mSec.

First program!



Major Language Parts

- Structure – lots of syntax items
- Variables – dealing with data/variables
- Functions – smaller subroutines

- https://www.arduino.cc/en/Reference/HomePage

Control Structures

- If – decides True or False
- If..else – two paths, one True and other False
- For – usually a loop with counter
- Switch case – usually in a state machine
- While – loops until a condition is not true
- Do..while – similar to While but will always run 1 time
- Break – exit current code block
- Continue - skips the rest of the current iteration of a loop
- Return – terminate a function and return a value
- Goto – jumps to a Label (not preferred in C, stack issue)

Structure

- **Setup()** – first part of the application program that only runs once at startup
- **Loop()** – the main program that is the application that you want to work

Syntax

- **;** (semicolon which ends a line of code)
- **{}** (controls a group of code that goes together)
- **//** (single line comment)
- **/*** xxxxx ***/** (multi-line comment)
- **#define** (assigns value to symbol)
- **#include** (for adding other programs)

Comparison Operators

- **==** (equal to, this is not the same as “=“)
- **!=** (not equal to)
- **<** (less than)
- **>** (greater than)
- **<=** (less than or equal to)
- **>=** (greater than or equal to)

Boolean Operators (True/False)

- **&&** (AND function)
- **||** (OR function)
- **!** (NOT function)

Bitwise Operators

- **&** (bitwise AND) **1 & 1 = 1**
- **|** (bitwise OR) **1 | 0 = 1, 0 | 1 = 1**
- **^** (bitwise XOR) **0 ^ 0 = 1, 1 ^ 1 = 1**
- **~** (bitwise NOT) **~1 = 0, ~0 = 1**
- **<<** (bitshift left – same as multiple by 2)
- **>>** (bitshift right – same as divide by 2)

Compound Operators

- **++** (increment by 1)
- **--** (decrement by 1)
- **+=** (compound addition – **X += Y**)
- **-=** (compound subtraction – same as X = X - Y)
- ***=** (compound multiplication – careful here)
- **/=** (compound division)
- **&=** (compound bitwise AND)
- **|=** (compound bitwise OR)

Variables and how to control them

- Constants – fixed values
- Data Types – defining how big they are
- Conversion – converting to different type
- Variable Scope & Qualifiers – who sees it
- Utilities – special things we can do

Constants

- HIGH or LOW
- INPUT or OUTPUT or INPUT_PULLUP
- LED_BUILTIN
- True or False
- Integer constants
- Floating point constants
- **Ex. PI, √, zero, your own value**

Data Types

- Void (means no data is passed or received depending on placement)
- Boolean (True or False)
- Char (usually an ASCII character)
- Unsigned char (more specific form of char that is from 0 to 255)
- Byte => positive 8-bit number
- Int => +/- 16-bit number that is called an Integer
- Unsigned Int => positive 16-bit number
- Word => 16-bit or 32-bit positive number, careful
- Long => +/- 32-bit number
- Unsigned long => positive 32-bit number
- Short => +/- 16-bit number
- Float => +/- 32-bit floating point number
- Double => +/- 32-bit floating point number, but some versions are 64-bit
- String (character array) – has lots of additional features
- Array (group of variables)

Conversion

- Char() – converts to character
- Byte() – converts to byte
- Int() – converts to integer
- Word() – convert to word
- Long() - convert to long
- Float() – convert to floating point number

Variable Scope & Qualifiers

- Variable scope (not global so limits visibility)
- **Static** (static keyword is used to create variables that are visible to only one function and persist beyond the function call)
- **Volatile** (directs the compiler to load the variable from RAM or control register)
- **Const** (constant that can't be changed, read-only)

Utilities

- **Sizeof()** – returns the number of bytes in a variable type
- **PROGMEM** – stores data in flash (program) memory used for parameters that are not known ahead of time like parameters, settings or data.

Functions

- Digital I/O
- Analog I/O
- Advanced I/O
- Time
- Math
- Trigonometry
- Characters
- Random Numbers
- Bits and Bytes
- External Interrupts
- Interrupts
- Communication
- USB (ATMEGA32U4 based boards and Due/Zero Only)

Digital I/O

- **Pinmode()** – Pin and Mode setting
- **digitalWrite()** – puts HIGH or LOW to a pin
- **digitalRead()** – gets HIGH or LOW from a pin

Analog I/O

- **analogReference()** - configures the reference voltage used for analog input of ADC
- **analogRead()** - reads the value from the specified analog pin, returns integer
- **Analog(Write)** – PWM for LED or motor drive, outputs a duty cycle to the specified pin

Advance I/O

- **Tone()** – square wave at specified frequency
- **noTone()** – stops square wave
- **shiftOut()** – software SPI output function
- **shiftIn()** – software SPI input function
- **pulseIn()** – times a pin HIGH or LOW time in microseconds

Time

- **Millis()** – millisecond of running time
- **Micros()** – microsecond of running time
- **Delay()** – delay for xx number of milliseconds
- **delayMicroseconds()** – delay for xx microseconds

Trigonometry

- **Sin()** – sine function
- **Cos()** – cosine function
- **Tan()** – tangent function

Characters

- isAlphaNumeric()
- isAlpha()
- isAscii()
- isWhitespace()
- isControl()
- isDigit()
- isGraph()
- isLowerCase()
- isPrintable()
- isPunct()
- isSpace()
- isUpperCase()
- isHexdecimalDigit()

Random Numbers

- randomSeed()
- Random()

Bits and Bytes

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- Bit()

Interrupts

- External Interrupts
- attachInterrupts()
- detachInterrupts()

- Interrupts
- Interrupts()
- noInterrupts()

Communications

- Serial – lots of related functions
- Stream – lots of related functions

- USB (ATMEGA32U4 boards
- Keyboard (these are structures w/ elements)
- Mouse (these are structures w/ elements)

Conclusion

- Introduced Arduino.
- Explained how and what it does.
- Introduced the software environment.
- Completed first project.

- What is your first project going to do?

- Material from http://arduino.cc web site.