Communicating with Others

- Arduino can use same USB cable for programming and to talk with computers
- Talking to other devices uses the “Serial” commands
  - Serial.begin() – prepare to use serial
  - Serial.print() – send data to computer
  - Serial.read() – read data from computer

Can talk to not just computers.
Most things more complex than simple sensors/actuators speak serial.
Watch the TX/RX LEDs

- **TX** – sending to PC
- **RX** – receiving from PC
- **Used when programming or communicating**

*(and keep an eye on that pesky pin13 LED too)*
Arduino Says “Hi”

“serial_hello_world”

- Send “Hello world!” to your computer (and blink LED)
- Click on “Serial Monitor” to see output
- Watch TX LED compared to pin13 LED

This sketch is located in the handout, but it’s pretty short. Use on-board pin 13 LED, no need to wire anything up.
Telling Arduino What To Do

“serial_read_basic”

• You type “H” – LED blinks

• In “Serial Monitor” type “H”, press Send

• Watch pin 13 LED

This sketch is in “Examples/serial_comm/serial_read_basic”.
Notice how you might not always read something, thus the “-1” check.
Can modify it to print “hello world” after it receives something, but before it checks for ‘H’.
This way you can verify it’s actually receiving something.
Arduino Communications

is just serial communications

• Psst, Arduino doesn’t really do USB
• It really is “serial”, like old RS-232 serial
• All microcontrollers can do serial
• Not many can do USB
• Serial is easy, USB is hard

serial terminal from the olde days
Serial Communications

• “Serial” because data is broken down into bits, each sent one-by-one on a single wire:

```
'H'
= 0 1 0 0 1 0 0 0
= L H L L H L L L
= [HIGH]
   [LOW]
```

• Toggle a pin to send data, just like blinking an LED

• Only a single data wire is needed to send data. One other to receive.

Note, a single data wire. You still need a ground wire.
Arduino & USB-to-serial

Arduino board is really two circuits

USB to serial

Arduino microcontroller

Original Arduino boards were RS–232 serial, not USB.
New Arduino Mini

Arduino Mini separates the two circuits

Arduino Mini USB adapter
Arduino Mini

aka. “Arduino Stamp”
If you don’t talk with a computer, the USB-to-serial functionality is superfluous.
Arduino to Computer

USB is totally optional for Arduino
But it makes things easier

Original Arduino boards were RS–232 serial, not USB.
Arduino & USB

- Because Arduino is all about serial,
- And not USB,
- Interfacing to things like USB flash drives, USB hard disks, USB webcams, etc. is not possible

Also, USB is a host/peripheral protocol. Being a USB “host” means needing a lot of processing power and software, not something for a tiny 8kB microcontroller. It can be a peripheral. In fact, there is an open project called “AVR–USB” that allows AVR chips like used in Arduino to be proper USB peripherals. See: http://www.obdev.at/products/avrusb/
Controlling the Computer

• Can send sensor data from Arduino to computer with `Serial.print()`

• There are many different variations to suite your needs:

```c
int val = 123;
Serial.print(val);   // sends 3 ASCII chars "123"
Serial.print(val,DEC); // same as above
Serial.print(val,HEX); // sends 2 ASCII chars "7B"
Serial.print(val,BIN); // sends 8 ASCII chars "01111011"
Serial.print(val,BYTE); // sends 1 byte, the verbatim value
```
Controlling the Computer

You write one program on Arduino, one on the computer

**In Arduino: read sensor, send data as byte**

```cpp
void loop() {
  val = analogRead(analogInput);  // read the value on analog input
  Serial.print(val/4, BYTE);      // print a byte value out
  delay(50);                     // wait a bit to not overload the port
}
```

**In Processing: read the byte, do something with it**

```java
import processing.serial.*;

Serial myPort;  // The serial port

void setup() {
  String portname = "/dev/tty.usbserial-A3000Xv0";
  myPort = new Serial(this, myPort, 9600);
}

void draw() {
  while (myPort.available() > 0) {
    int inByte = myPort.read();
    println(inByte);
  }
}
```

But writing Processing programs is for another time
Controlling the Computer

- Receiving program on the computer can be in any language that knows about serial ports
  
  - C/C++, Perl, PHP, Java, Max/MSP, Python, Visual Basic, etc.

- Pick your favorite one, write some code for Arduino to control

If interested, I can give details on just about every language above.
Another Example

“serial_read_blink”

- Type in a number 1-9 and LED blinks that number
- Converts number typed into usable number

This sketch is also in the handout
Controlling Arduino, Again

"SerialReadBlink"

Type a number 1-9 and LED blinks that many times

Converts typed ASCII value into usable number

Most control issues are data conversion issues

Mysterious val = val – '0';

statement converts the byte that represents the character to a byte of that number

For example, if the character is '3', the ASCII code is 51

The ASCII code for '0' is 48

So, 51 – 48 = 3

This converts the character '3' into the number 3
Serial-controlled RGB

“serial_rgb_led”

Send color commands to Arduino

e.g. “r200”, “g50”, “b0”

Sketch parses what you type, changes LEDs

This sketch is located in the handout.

Color command is two parts: colorCode and colorValue

- colorCode is a character, ‘r’, ‘g’, or ‘b’.
- colorValue is a number between 0–255.

Sketch shows rudimentary character string processing in Arduino
Reading Serial Strings

- New Serial function in last sketch: "Serial.available()"
- Can use it to read all available serial data from computer
- Great for reading strings of characters
- The "readSerialString()" function at right takes a character string and sticks available serial data into it

```c
//read a string from the serial and store it in an array
//you must supply the array variable
void readSerialString (char *strArray) {
  int i = 0;
  if(!Serial.available()) {
    return;
  }
  while (Serial.available()) {
    strArray[i] = Serial.read();
    i++;
  }
}
```

Pay no attention to the pointer symbol ("*")
Must be careful about calling readSerialString() too often or you’ll read partial strings