

Communicating with Others

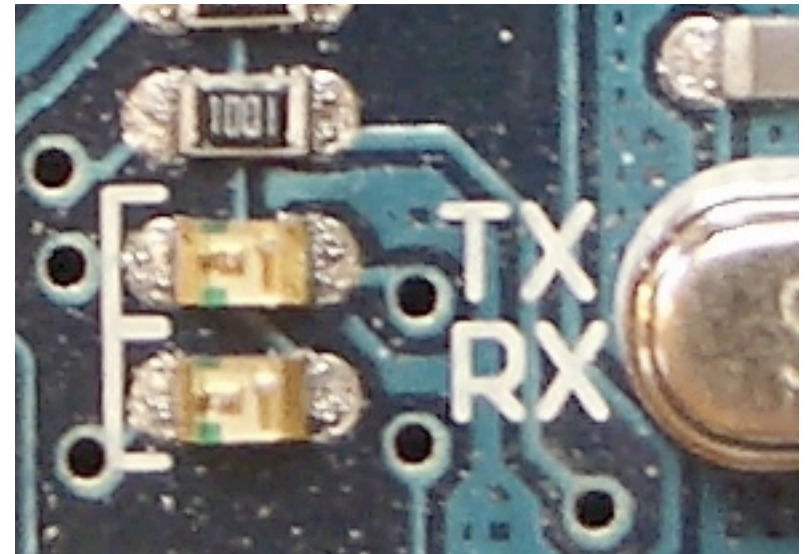
- Arduino can use same USB cable for programming and to talk with computers
- Talking to other devices uses the “Serial” commands
 - `Serial.begin()` – prepare to use serial
 - `Serial.print()` – send data to computer
 - `Serial.read()` – read data from computer

Can talk to not just computers.

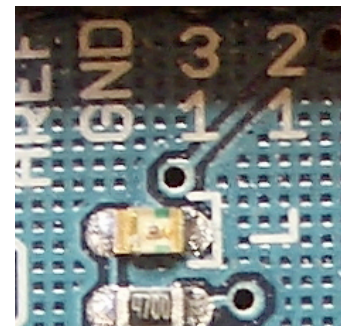
Most things more complex than simple sensors/actuators speak serial.

Watch the TX/RX LEDs

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



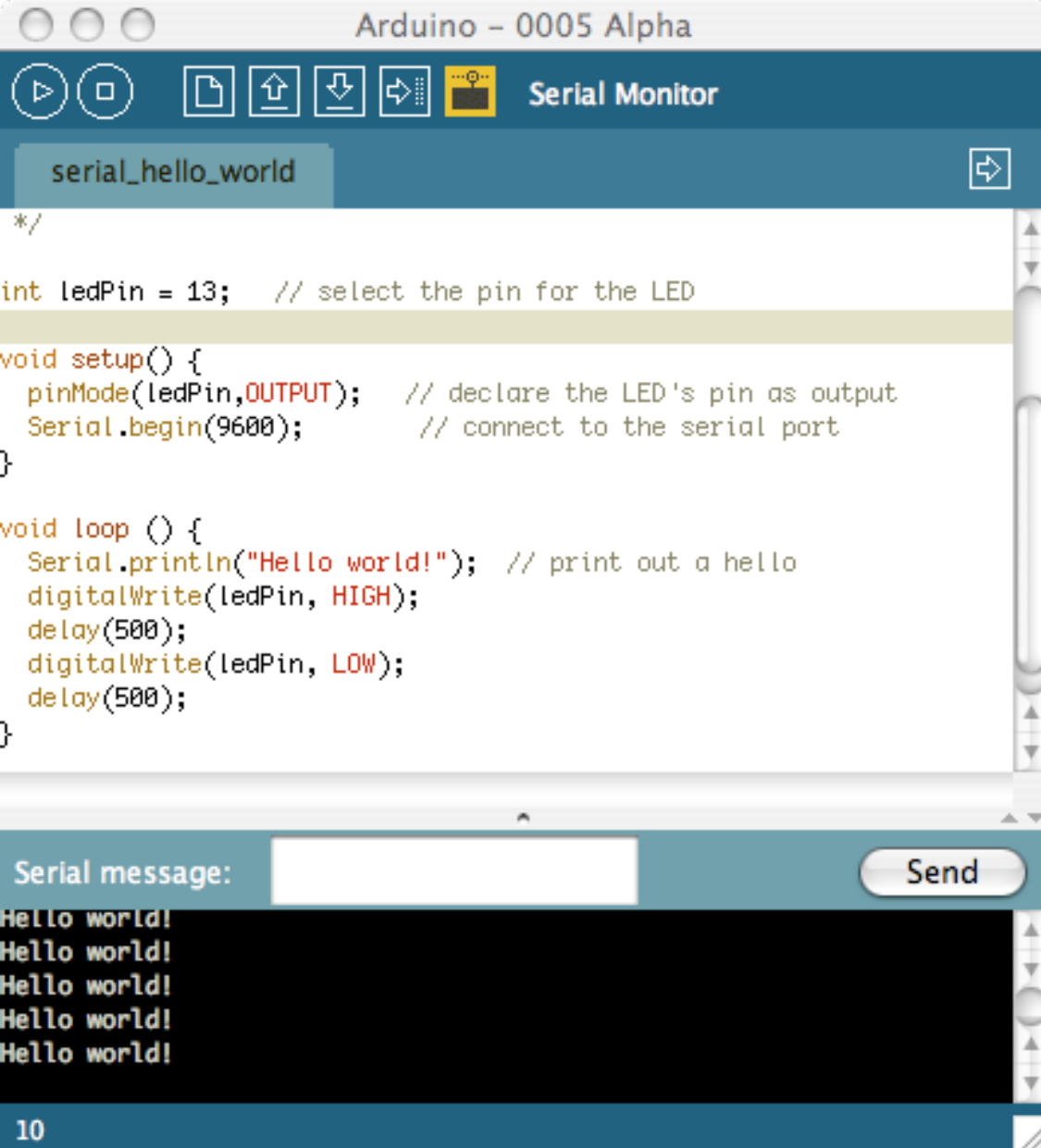
*(and keep an eye
on that pesky
pin 13 LED too)*



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 pin 13 LED



The screenshot shows the Arduino IDE interface. The title bar reads "Arduino - 0005 Alpha". The "Serial Monitor" window is open, displaying the sketch "serial_hello_world". The code in the editor is as follows:

```
*/  
  
int ledPin = 13; // select the pin for the LED  
  
void setup() {  
  pinMode(ledPin,OUTPUT); // declare the LED's pin as output  
  Serial.begin(9600); // connect to the serial port  
}  
  
void loop () {  
  Serial.println("Hello world!"); // print out a hello  
  digitalWrite(ledPin, HIGH);  
  delay(500);  
  digitalWrite(ledPin, LOW);  
  delay(500);  
}
```

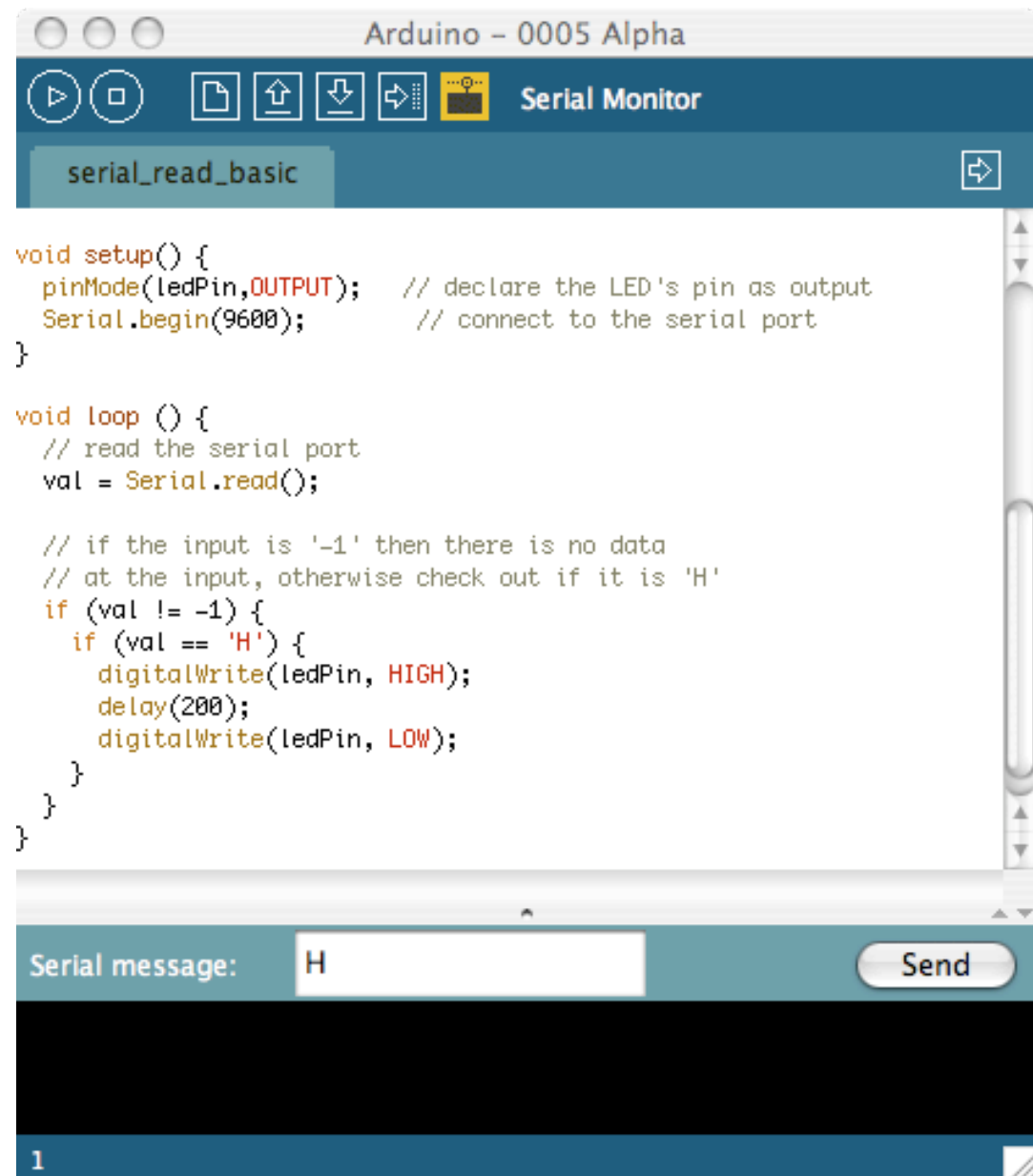
Below the code, the Serial Monitor shows the output "Hello world!" repeated five times. A "Send" button is visible to the right of the input field. The status bar at the bottom indicates line 10.

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



```
void setup() {
  pinMode(ledPin,OUTPUT); // declare the LED's pin as output
  Serial.begin(9600);     // connect to the serial port
}

void loop () {
  // read the serial port
  val = Serial.read();

  // if the input is '-1' then there is no data
  // at the input, otherwise check out if it is 'H'
  if (val != -1) {
    if (val == 'H') {
      digitalWrite(ledPin, HIGH);
      delay(200);
      digitalWrite(ledPin, LOW);
    }
  }
}
```

Serial message: H Send

1

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

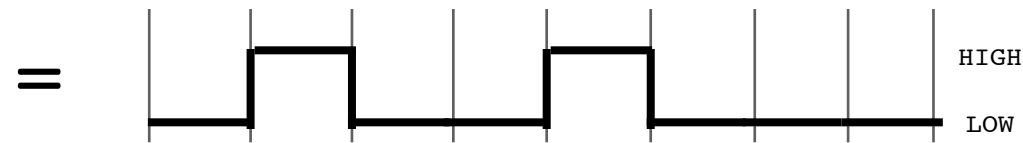
Serial Communications

- “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

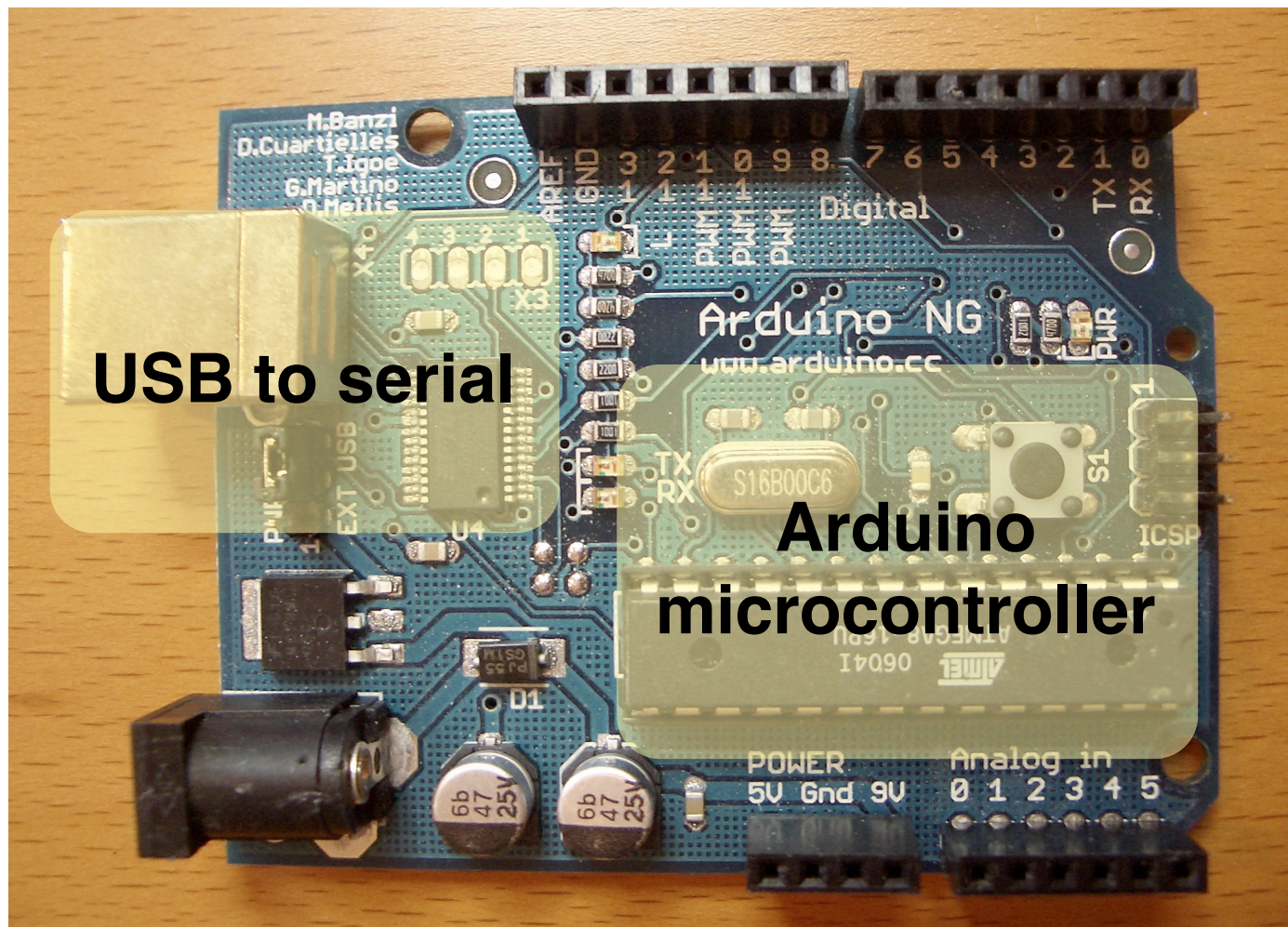


- Toggle a pin to send data, just like blinking an LED
- Only a single data wire is needed to send data. One other to receive.

Note, a single data wire. You still need a ground wire.

Arduino & USB-to-serial

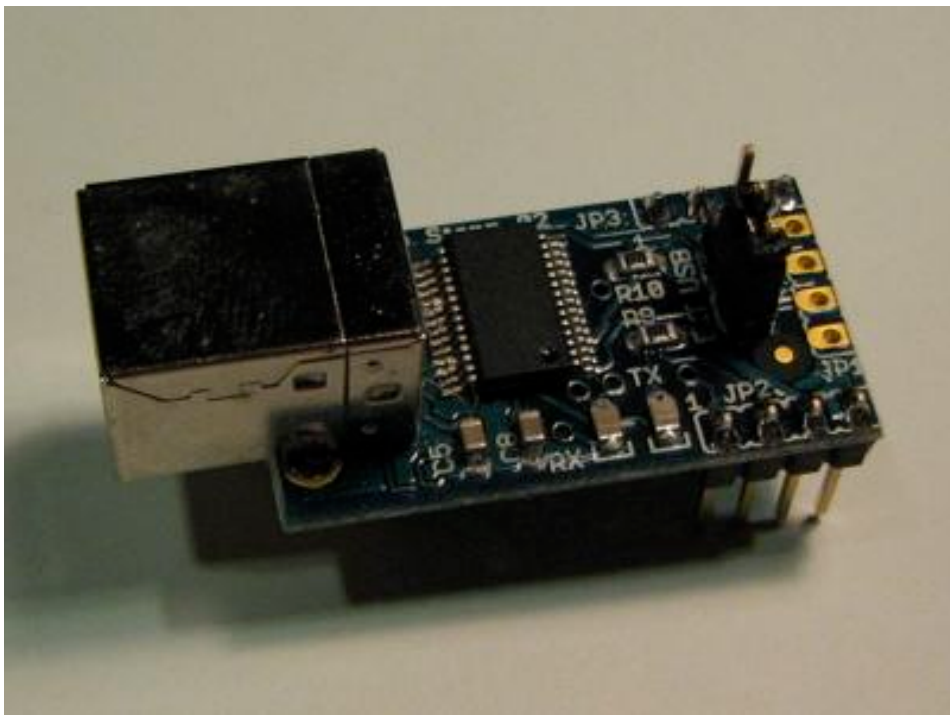
Arduino board is really two circuits



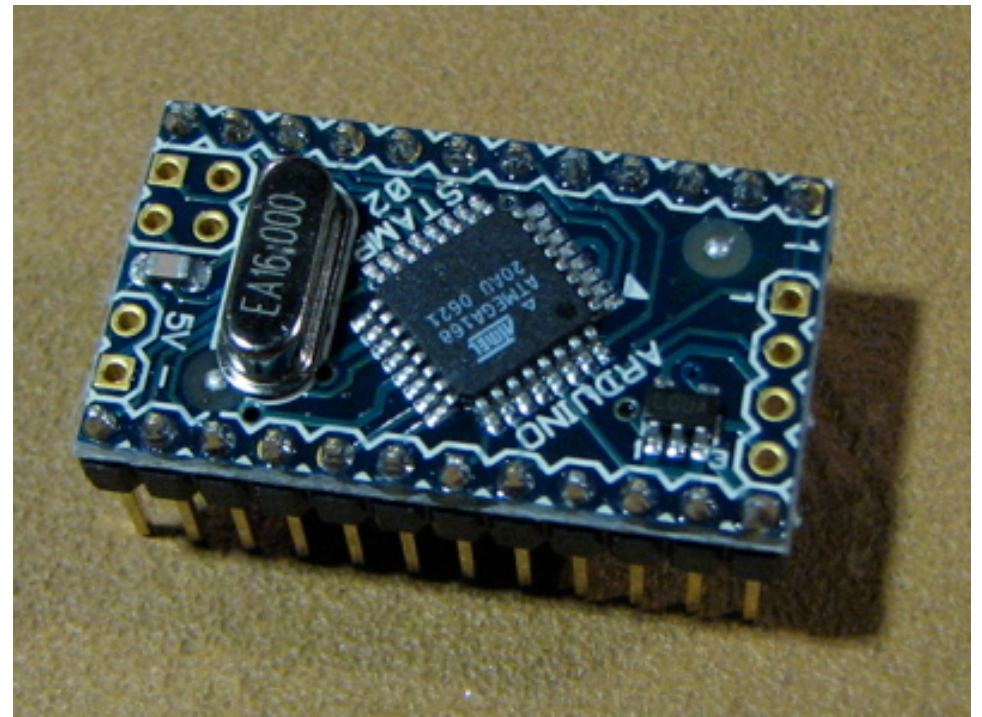
Original Arduino boards were RS-232 serial, not USB.

New Arduino Mini

Arduino Mini separates the two circuits



Arduino Mini USB adapter

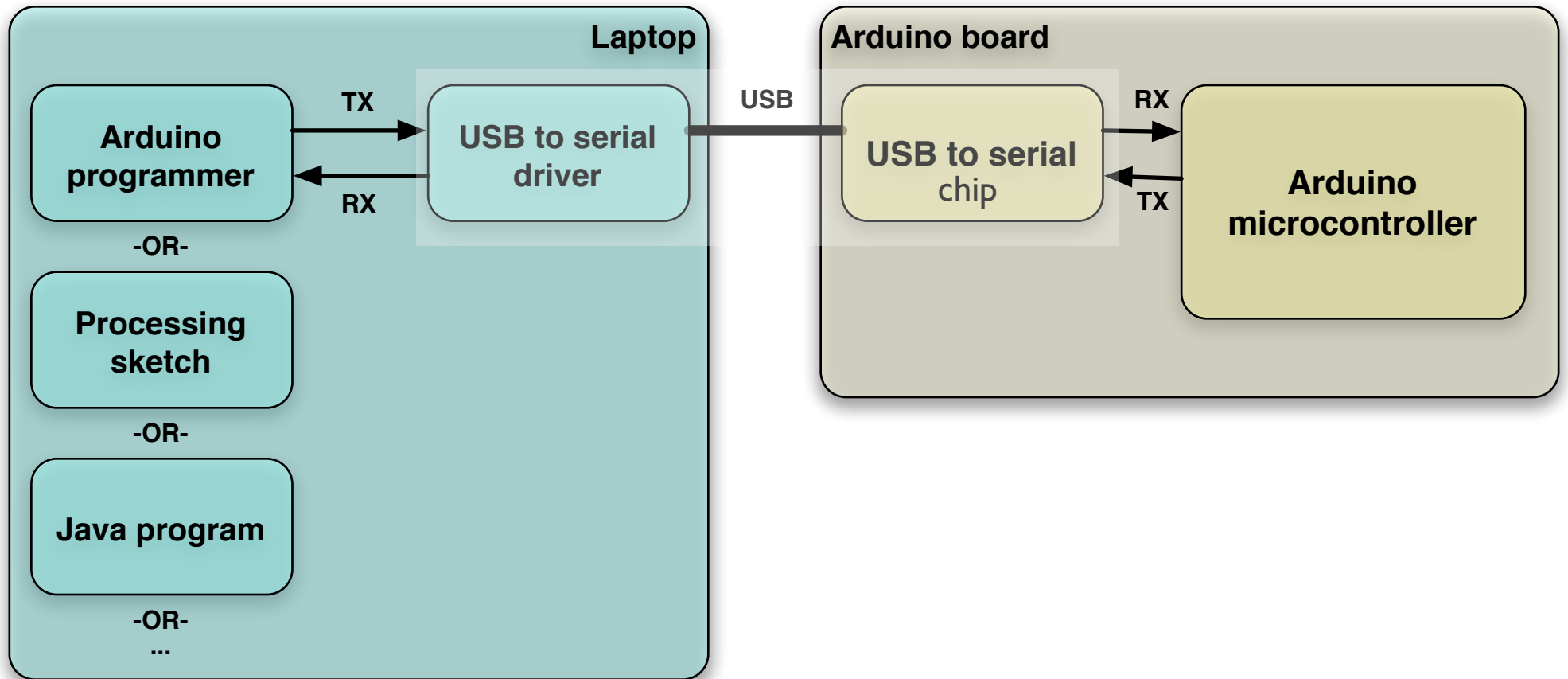


Arduino Mini

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:

```
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

```
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

```
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



```
void setup() {
  pinMode(ledPin,OUTPUT); // declare the LED's pin as output
  Serial.begin(9600); // connect to the serial port
}

void loop () {
  val = Serial.read(); // read the serial port

  // if the stored value is a single-digit number, blink the LED that number
  if (val > '0' && val <= '9' ) {
    val = val - '0'; // convert from character to number
    for(int i=0; i<val; i++) {
      Serial.println("blink!");
      digitalWrite(ledPin,HIGH);
      delay(75);
      digitalWrite(ledPin, LOW);
      delay(75);
    }
  }
}
```

Serial message: 3 Send

blink!
blink!
blink!

5

This sketch is also in the handout

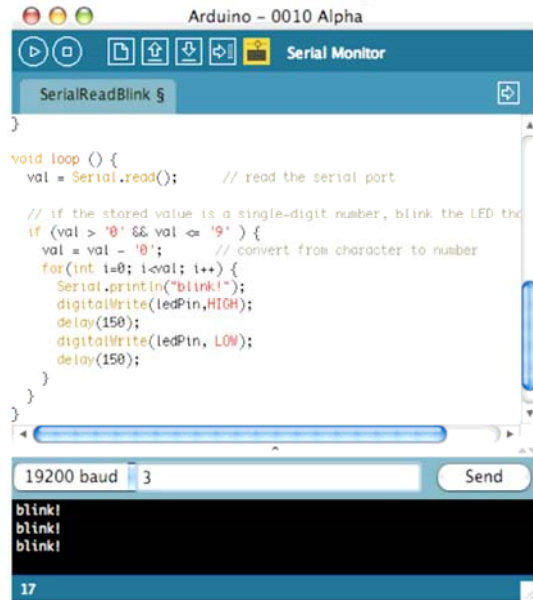
Controlling Arduino, Again

“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



Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20	!	64	40	@	96	60	'
^A	1	01		SOH	33	21	!	65	41	A	97	61	a
^B	2	02		STX	34	22	!"	66	42	B	98	62	b
^C	3	03		ETX	35	23	!#"	67	43	C	99	63	c
^D	4	04		EOT	36	24	!#\$	68	44	D	100	64	d
^E	5	05		ENQ	37	25	!#\$%	69	45	E	101	65	e
^F	6	06		ACK	38	26	!#\$%	70	46	F	102	66	f
^G	7	07		BEL	39	27	!#\$%'	71	47	G	103	67	g
^H	8	08		BS	40	28	!#\$%'(72	48	H	104	68	h
^I	9	09		HT	41	29	!#\$%'()	73	49	I	105	69	i
^J	10	0A		LF	42	2A	!#\$%'()*	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	!#\$%'()*+	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	!#\$%'()*+,	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	!#\$%'()*+,-	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	!#\$%'()*+,-.	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	!#\$%'()*+,-./	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	!#\$%'()*+,-./0	80	50	P	112	70	p
^Q	17	11		DC1	49	31	!#\$%'()*+,-./01	81	51	Q	113	71	q
^R	18	12		DC2	50	32	!#\$%'()*+,-./012	82	52	R	114	72	r
^S	19	13		DC3	51	33	!#\$%'()*+,-./0123	83	53	S	115	73	s
^T	20	14		DC4	52	34	!#\$%'()*+,-./01234	84	54	T	116	74	t
^U	21	15		NAK	53	35	!#\$%'()*+,-./012345	85	55	U	117	75	u
^V	22	16		SYN	54	36	!#\$%'()*+,-./0123456	86	56	V	118	76	v
^W	23	17		ETB	55	37	!#\$%'()*+,-./01234567	87	57	W	119	77	w
^X	24	18		CAN	56	38	!#\$%'()*+,-./012345678	88	58	X	120	78	x
^Y	25	19		EM	57	39	!#\$%'()*+,-./0123456789	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	!#\$%'()*+,-./0123456789:	90	5A	Z	122	7A	z
^[27	1B		ESC	59	3B	!#\$%'()*+,-./0123456789:;	91	5B	[123	7B	{
^\	28	1C		FS	60	3C	!#\$%'()*+,-./0123456789:;<	92	5C	\	124	7C	
^]	29	1D		GS	61	3D	!#\$%'()*+,-./0123456789:;<=	93	5D]	125	7D	}
^^	30	1E	▲	RS	62	3E	!#\$%'()*+,-./0123456789:;<=?	94	5E	~	126	7E	~
^_	31	1F	▼	US	63	3F	!#\$%'()*+,-./0123456789:;<=?>	95	5F	_	127	7F	~*

ASCII codes

Standard byte codes for
characters

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

ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

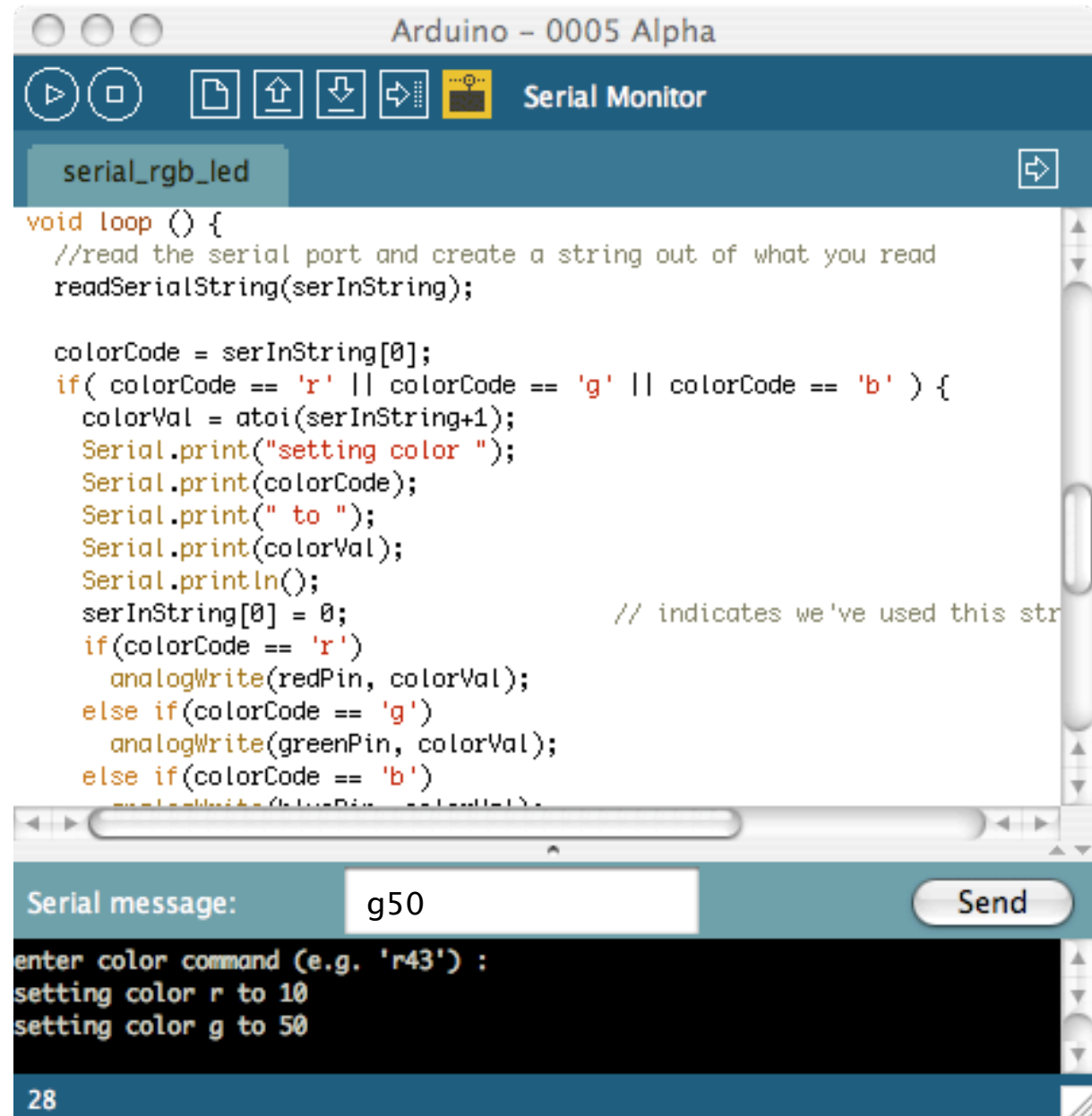
Serial-controlled RGB

"serial_rgb_led"

Send color
commands to
Arduino

e.g. "r200", "g50", "b0"

Sketch parses what
you type, changes
LEDs



```
Arduino - 0005 Alpha
Serial Monitor
serial_rgb_led

void loop () {
  //read the serial port and create a string out of what you read
  readSerialString(serInString);

  colorCode = serInString[0];
  if( colorCode == 'r' || colorCode == 'g' || colorCode == 'b' ) {
    colorVal = atoi(serInString+1);
    Serial.print("setting color ");
    Serial.print(colorCode);
    Serial.print(" to ");
    Serial.print(colorVal);
    Serial.println();
    serInString[0] = 0; // indicates we've used this str
    if(colorCode == 'r')
      analogWrite(redPin, colorVal);
    else if(colorCode == 'g')
      analogWrite(greenPin, colorVal);
    else if(colorCode == 'b')
      analogWrite(bluePin, colorVal);
  }
}
```

Serial message:

```
enter color command (e.g. 'r43') :
setting color r to 10
setting color g to 50
```

28

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

```
//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