Lecture 7
Arduino: analog input /output.
Serial communication.

IAT267 Introduction to Technological Systems

Organizational items

- Assignment 2 – marks will be available in about one week.
- Project Milestone 1 – due today, marks will be up on webct this week.
- Project milestone 2 – due October 26 (next Wednesday)
    - Equipment
        - From the Library and purchased
    - Task distribution between team members

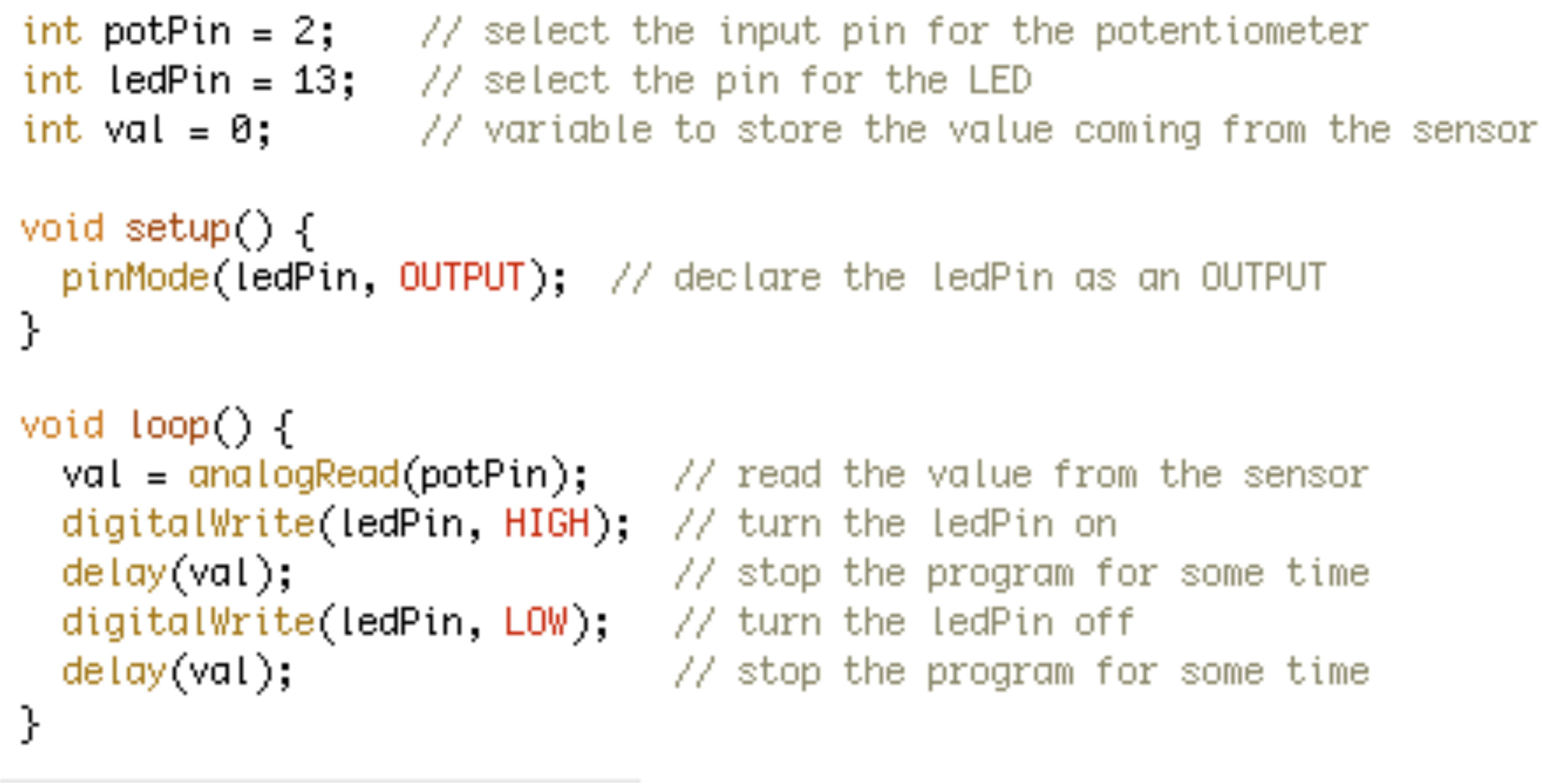
Quiz this week

- Will be available starting Friday until next Wednesday 5pm
- Can be done anytime in the availability interval
- We will not do the quizzes during the workshops.

Lecture topics for today

- Arduino – more on analog input / output
- Serial communication
- Code examples

Code from last week – adjust the blink frequency



```
int potPin = 2;    // select the input pin for the potentiometer
int ledPin = 13;   // select the pin for the LED
int val = 0;       // variable to store the value coming from the sensor

void setup() {
  pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT
}

void loop() {
  val = analogRead(potPin);    // read the value from the sensor
  digitalWrite(ledPin, HIGH);  // turn the ledPin on
  delay(val);                  // stop the program for some time
  digitalWrite(ledPin, LOW);   // turn the ledPin off
  delay(val);                  // stop the program for some time
}
```

How about brightness?

- So far we have seen the potentiometer / slider sensor / rotation sensor / light sensor in circuits used to modify the **blinking frequency** of an LED

- How can we adjust **the brightness of an LED** using the sensors connected to an analog input pin?

- LED connected to digital pin
  - Digital pin: pinMode is OUTPUT

AnalogWrite()

- **analogWrite(pin, value)**
- Writes an analog value to a pin.
- Arduino boards with an ATmega8 only support analogWrite() on **digital** pins 9, 10, and 11.
- For newer boards: Arduino Diecimilla: digital pins 3, 5, 6, 9, 10, 11 can be used for analogWrite()

AnalogWrite()

- Can be used to light a LED at varying brightness or drive a motor at various speeds.

- After a call to **analogWrite**, the pin will generate a steady wave until the next call to **analogWrite** (or a call to **digitalRead** or **digitalWrite** on the same pin).

Parameters of AnalogWrite()

- pin: the pin to write to.
- value: the duty cycle: between 0 and 255.
- A value of 0 generates a constant 0 volts output at the specified pin; a value of 255 generates a constant 5 volts output at the specified pin. For values in between 0 and 255, the pin rapidly alternates between 0 and 5 volts - the higher the value, the more often the pin is high (5 volts).

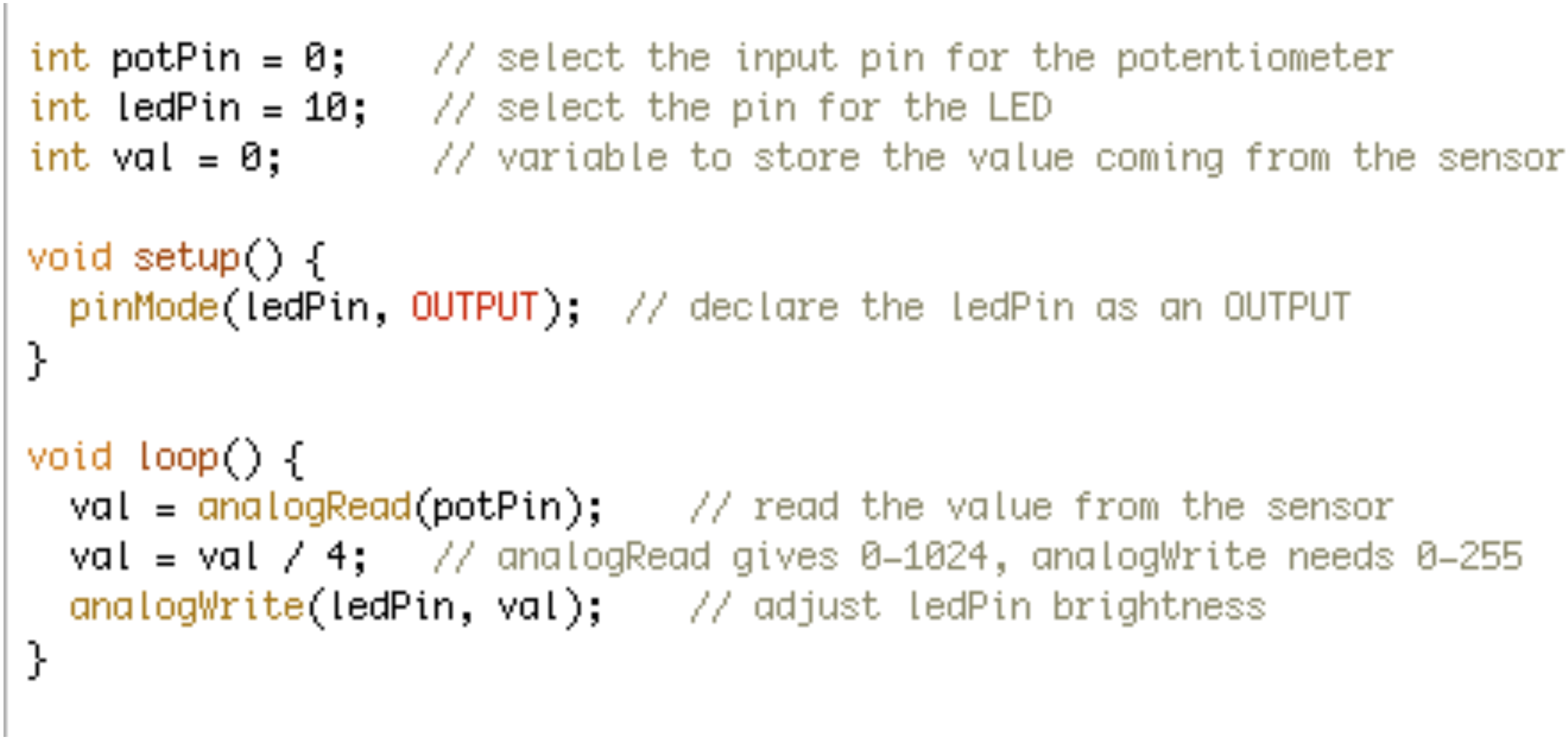
Code Example

```
int ledPin = 9;    // LED connected to digital pin 9
int analogPin = 3;    // potentiometer connected to analog pin 3
 int val = 0; // variable to store the read value

void setup()
{
   pinMode(ledPin, OUTPUT); // sets the pin as output
}
```

```
void loop()
{
    val = analogRead(analogPin); // read the input pin
    analogWrite(ledPin, val / 4);
// analogRead values go from 0 to 1023, analogWrite values from 0 to 255
}
```

Outcome: Sets the output to the LED proportional to the value read from the potentiometer.

code



```
int potPin = 0;    // select the input pin for the potentiometer
int ledPin = 10;   // select the pin for the LED
int val = 0;       // variable to store the value coming from the sensor

void setup() {
  pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT
}

void loop() {
  val = analogRead(potPin);    // read the value from the sensor
  val = val / 4;   // analogRead gives 0-1024, analogWrite needs 0-255
  analogWrite(ledPin, val);    // adjust ledPin brightness
}
```

- If instead of the potentiometer we have a light sensor: more light means less resistance → more voltage → ‘val’ will have a higher value so the LED will be brighter

- Darker: less brightness for the LED

- Automatic dimmer circuit - how can we obtain this behaviour? (brighter room should result in dimming of the LED, and in a darker room the LED should brighten).

Arduino and Serial Comm.

Serial Communication

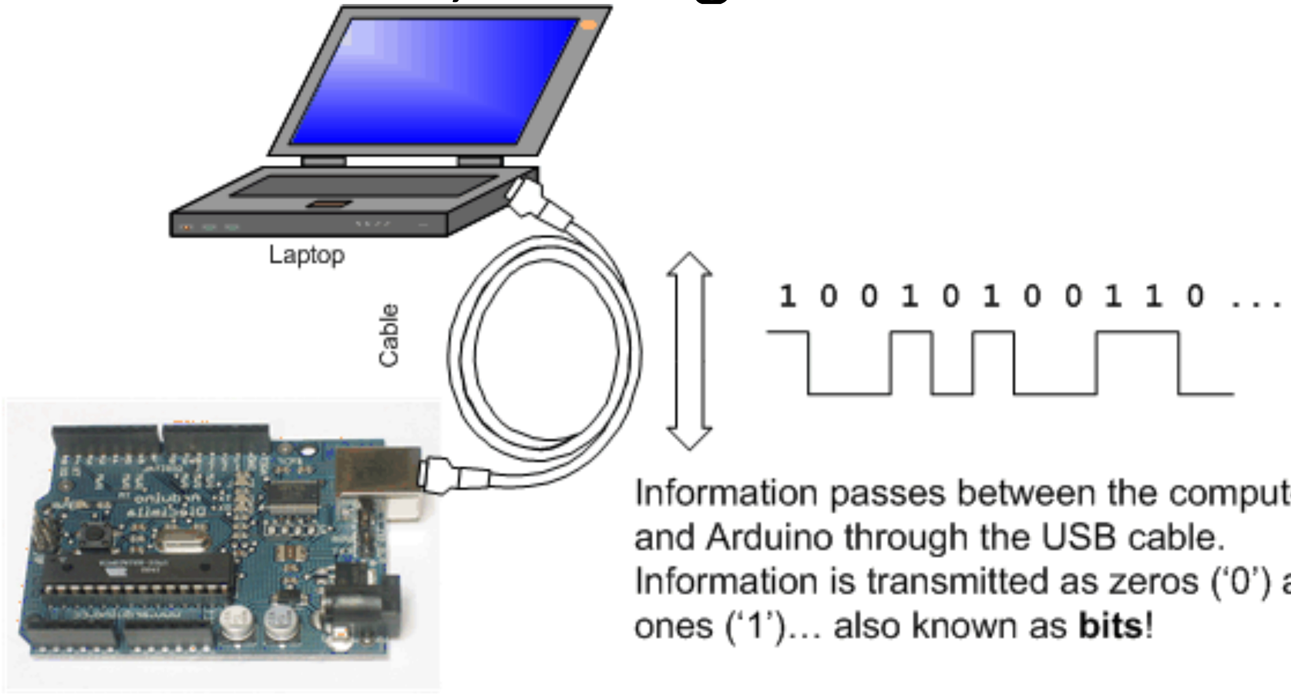
- The most common form of communication between electronic devices is *serial communication*.

- Communicating serially involves sending a series of digital pulses back and forth between devices at a mutually agreed-upon rate.

- The sender sends pulses representing the data to be sent at the agreed-upon *data rate*, and the receiver listens for pulses at that same rate.

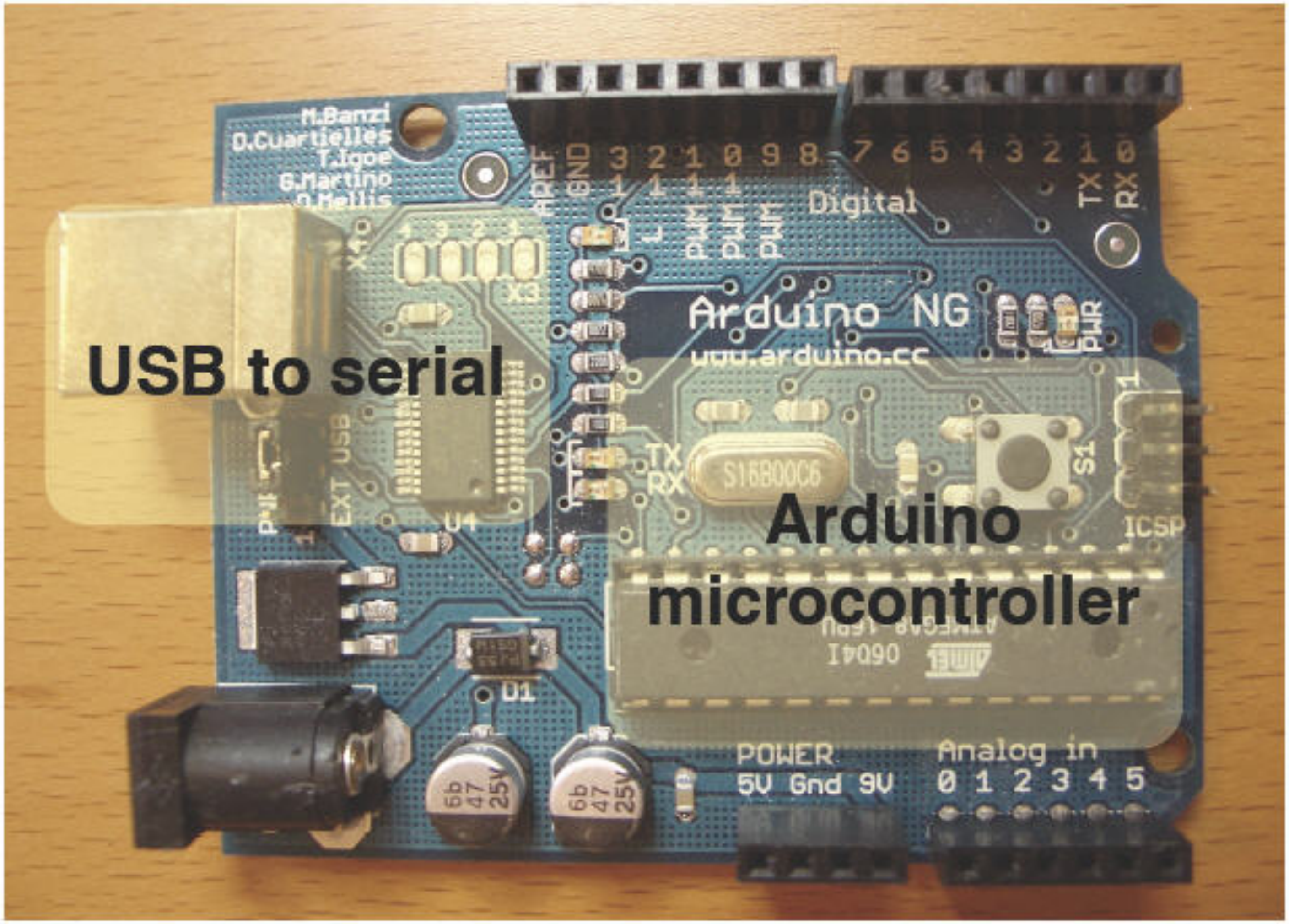
Serial Communication

- The word **serial** means "one after the other.“
- Serial data transfer is when we transfer data one **bit** at a time, one right after the other.



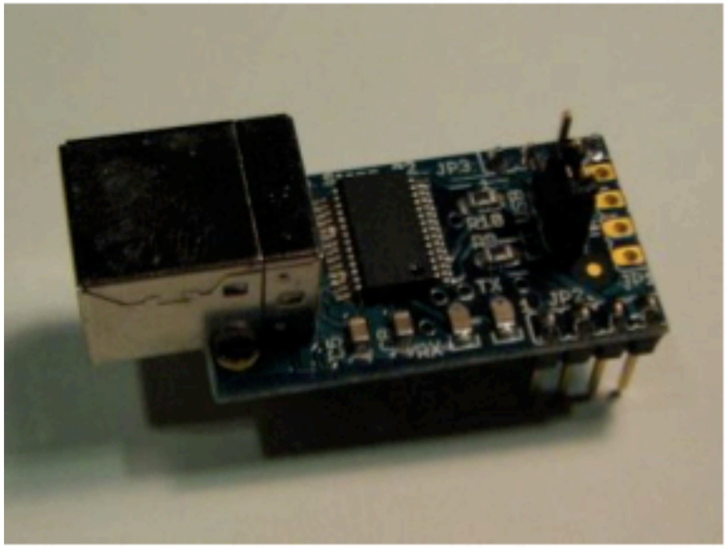
Information passes between the computer and Arduino through the USB cable. Information is transmitted as zeros (‘0’) and ones (‘1’)… also known as **bits**!

Arduino board is really two circuits

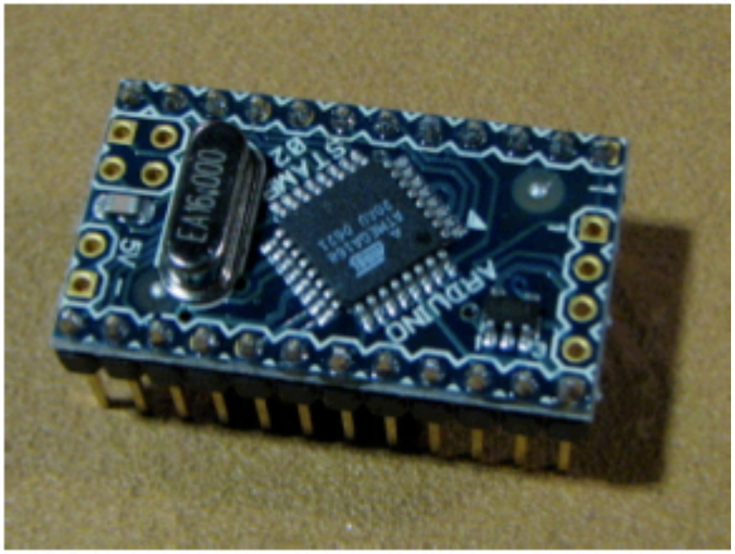


New Arduino Mini

Arduino Mini separates the two circuits

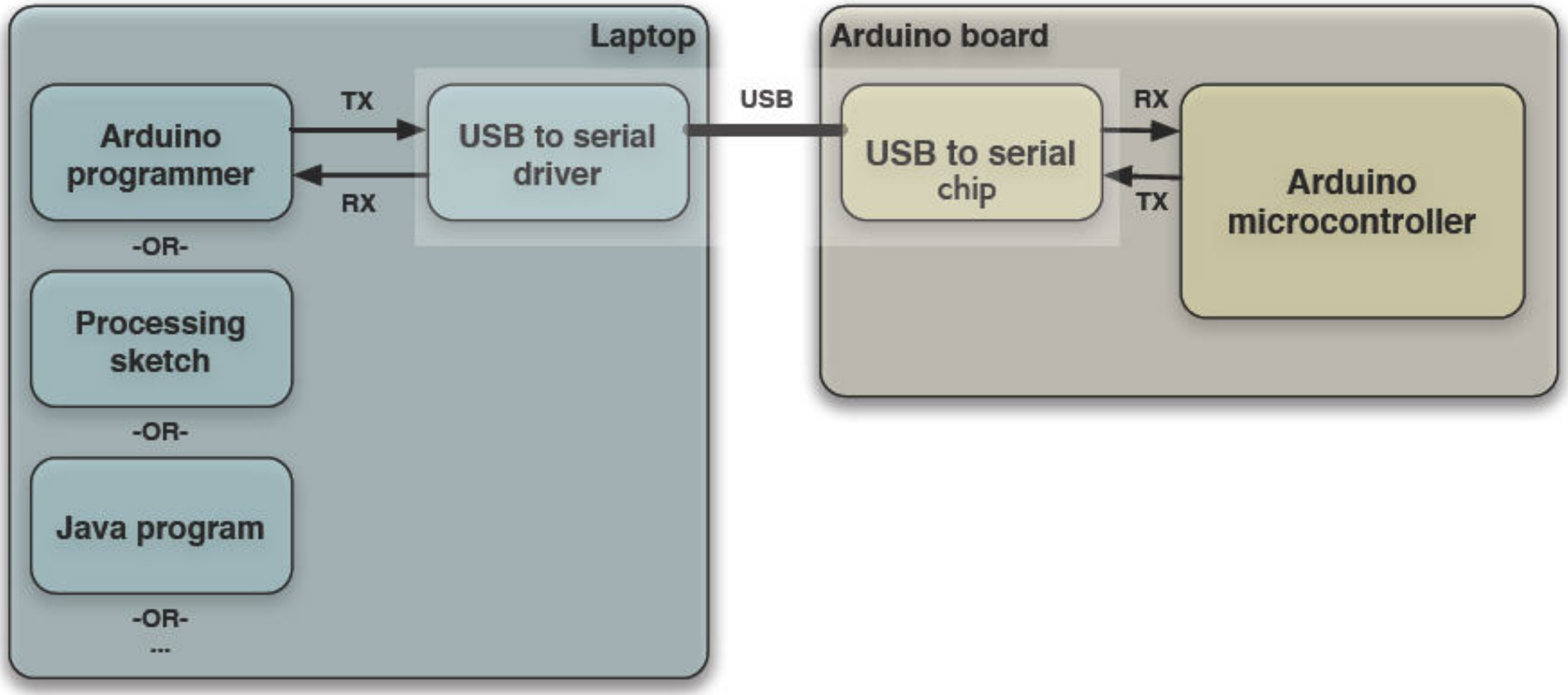


Arduino Mini USB adapter



Arduino Mini

Arduino to Computer



USB is totally optional for Arduino
But it makes things easier

Serial Communication Using Arduino

- Used for communication between the Arduino board and a computer or other devices.

- This communication happens via the Arduino board's serial or USB connection and on digital pins 0 (RX) and 1 (TX).

- Thus, if you use these functions, *you cannot also use pins 0 and 1 for digital i/o.*

TX/ RX LEDs

Arduino and USB

- Because Arduino is all about serial,
- And not USB:

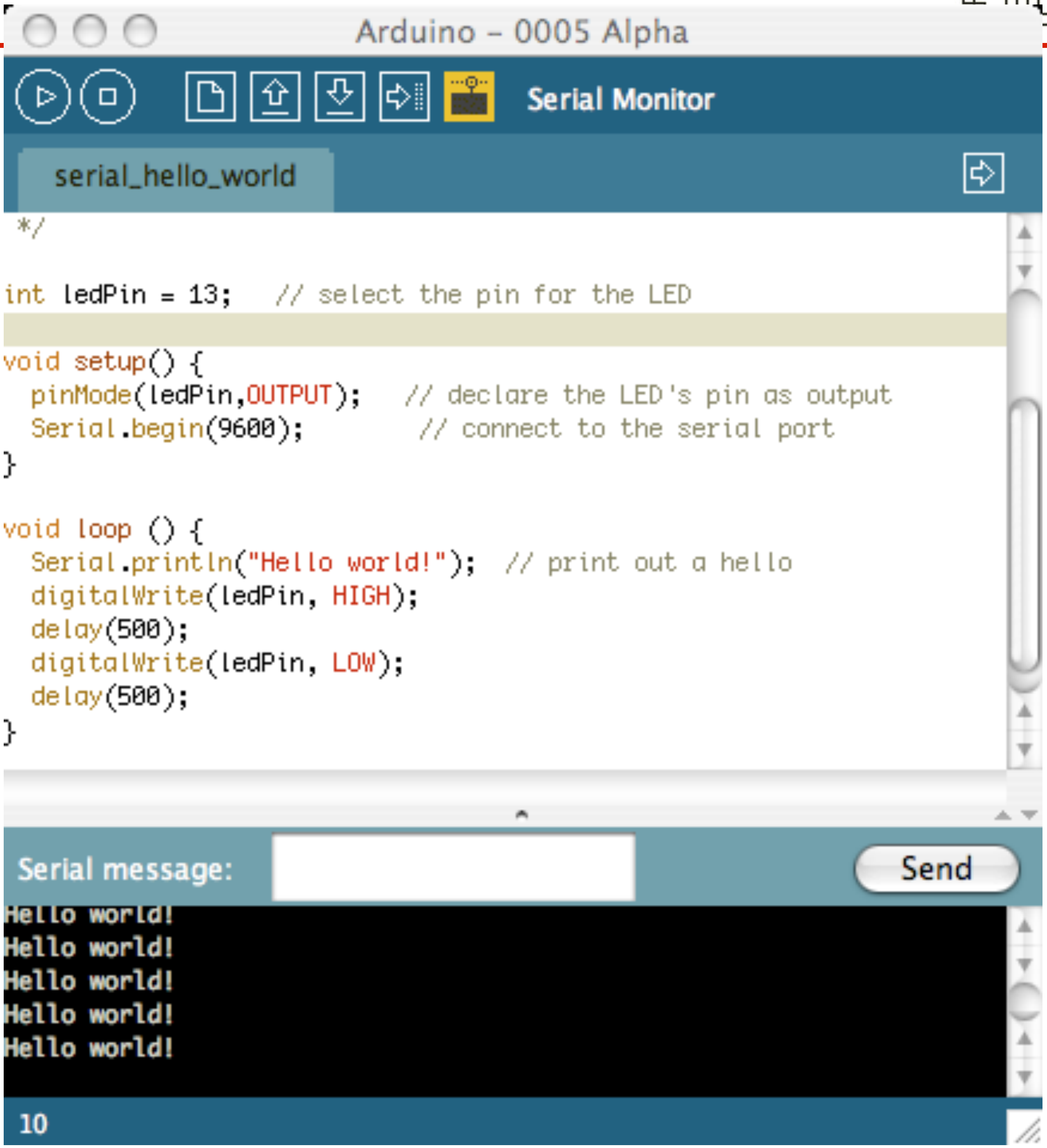
- Interfacing to things like USB flash drives, USB hard disks, USB webcams, etc. is *not possible*

USB / serial

- 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

Send/receive serial data

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



```
*/

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);
}
```

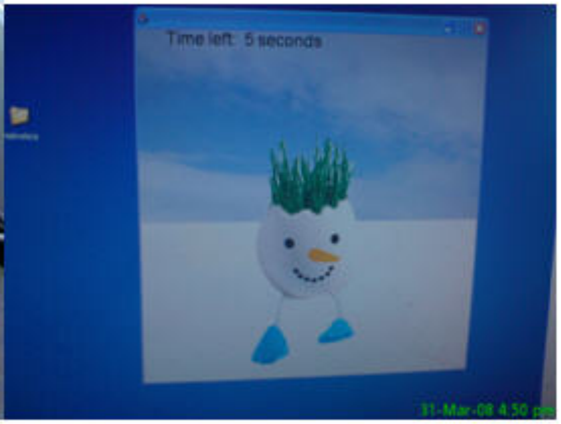
```
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
```

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

Applications of Serial Communication

- For many projects it is very common to have computers communicating with other devices
  - One of the most common configurations for physical computing systems is to have a microcontroller read a sensor, and then send the value of the sensor to a multimedia computer.

-inside of the pot

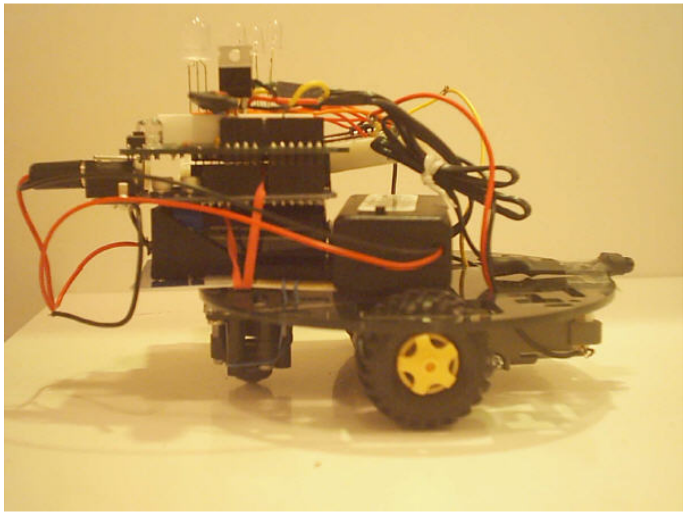


-what is displayed on screen

- The computer processes the input from the microcontroller and performs and action.
  - For example: the multimedia computer changes the playback of a video or the pitch of a sound, or activates some other multimedia response.

Fig 11
Screenshot of a video clip
Fig 13
Screenshot of a video clip
Fig 12
Screenshot of a video clip

- The reverse of this configuration is also common.
  - For example, a computer sends the coordinates of the mouse to the microcontroller to position a motor.



Protocols for serial communications

- A protocol is the set of parameters that the two devices agree upon in order to send information.

- There are many different protocols for serial communication, each suited to a different application.

Protocol = agreement between devices

- Physical Connection – serial port
- Timing – speed (bps)
- Electrical Connection
- Package size

Timing Agreement

- Timing of the pulses.
- This has to be set regardless of what serial protocol you're using.
- To be able to count the pulses, there has to be agreement about how fast they are coming.
- You will be using asynchronous serial communication, in which both devices have their own separate clock to keep track of time.

- The sender sends pulses representing the data being transmitted at an agreed-upon data rate, and the receiver listens for pulses at that same rate.

- The timing of the pulses is called the data rate or the baud rate.

- 9600 pulses per second → most frequently used

- Typically 8 pulses are grouped together. This means that one group of 8 pulses (also called a byte) is sent per millisecond, which is faster than human perception.

Bits and Bytes

- How data is measured:
  - A **single bit** is either a **zero** or a **one**.
  - You can group bits together into 8 bits which is 1 **byte**.
  - 1024 bytes is one **Kilobyte** (sometimes written KB).
  - 1024 KB (1048576 bytes) is one **Megabyte** (MB)
    1024 MB is 1 Gigabyte (GB)

Package Size

- There has to be some agreement as to how the sequence of pulses is interpreted.
- By interpreting them in groups of 8 (a byte), you can send numbers between 0 and 255.
- Serial data is passed byte by byte from one device to another. It’s up to the programmer to decide how each device (computer or microcontroller) should interpret those bytes: when the beginning of a message is, when the end is, and what to do with the bytes in between.

Speed of serial communication

- *Serial.begin(9600);   // set up Serial library at 9600 bps*
- bps = bits per second – **baud rate**

Example:

- If you're only sending one changing number (perhaps the value received from an analog sensor), and that number is less than 255, you know it can fit in a byte. This kind of message is easy.

- Just send the same byte over and over, and the computer can pick it up at any time.

- If you're sending more than that (and you usually are), things are a little more complicated. The receiving computer has to know when the message starts and when it ends.

Debugging Serial Communication

- Serial communication is difficult to debug because the problem could be in many different places:
  - microcontroller software or circuit
  - the multimedia computer software or hardware
  - or, the communication between the two.

Serial Communication and Arduino

- Where data comes from:
  - User entered data for Arduino to process:
    - Example: users enters data at the serial monitor and Arduino makes use of this data: how many times to blink an LED, set the brightness of an LED, set the speed of a servomotor, etc
  - Computer sends data serially to Arduino
  - Arduino sends data serially to computer

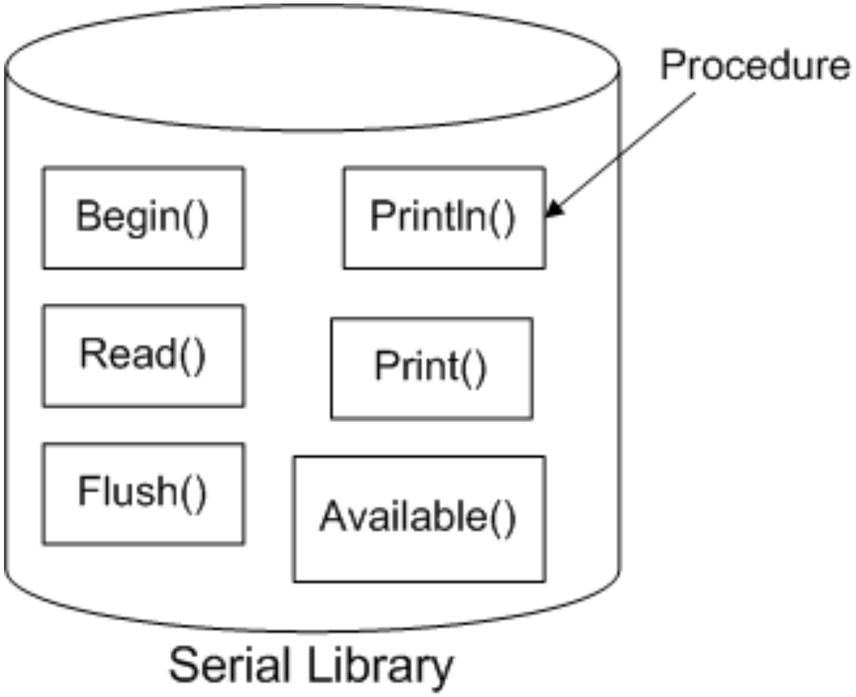
Serial comm. on the computer

- How to use serial communication to make a connection between a computer’s certain software environment and a microcontroller.
- We will use the Processing language for this purpose

Arduino Serial Library

- A library is a collection of procedures, where all the procedures are related.

- The library we will be using is the Serial Library, which allows the Arduino to send data back to the computer:



Serial Comm. Functions

- Serial.begin(speed)
- int Serial.available()
- int Serial.read()
- Serial.flush()
- Serial.print(data)
- Serial.println(data

Serial.begin(int speed)

- Sets the data rate in bits per second (baud) for serial data transmission.

- For communicating with the computer, use one of these rates: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.

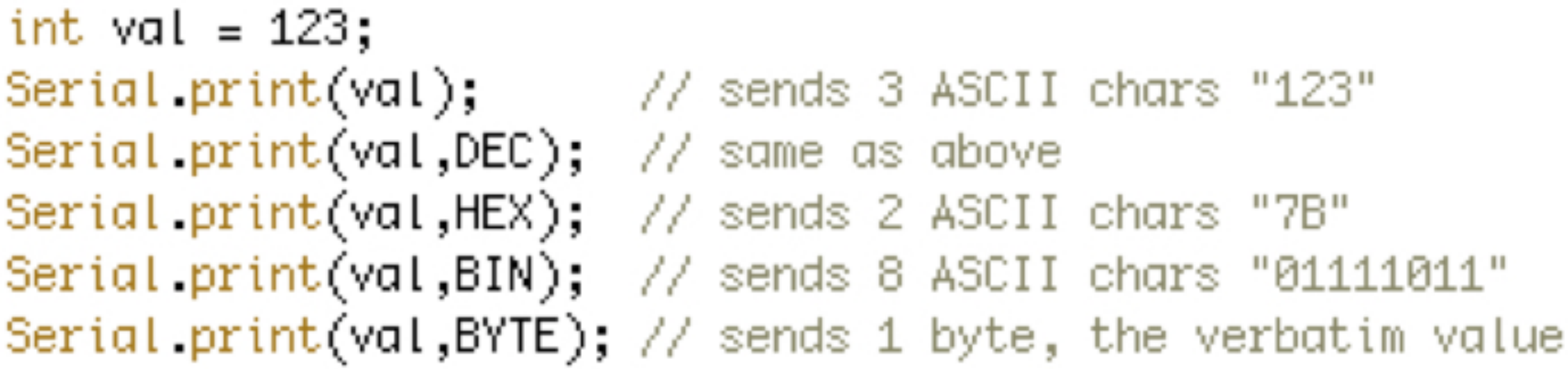
```
void setup()
     {
   Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
   }
```

int Serial.read()

- Reads incoming serial data.
- Returns an int, the first byte of incoming serial data available (or -1 if no data is available).

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
```

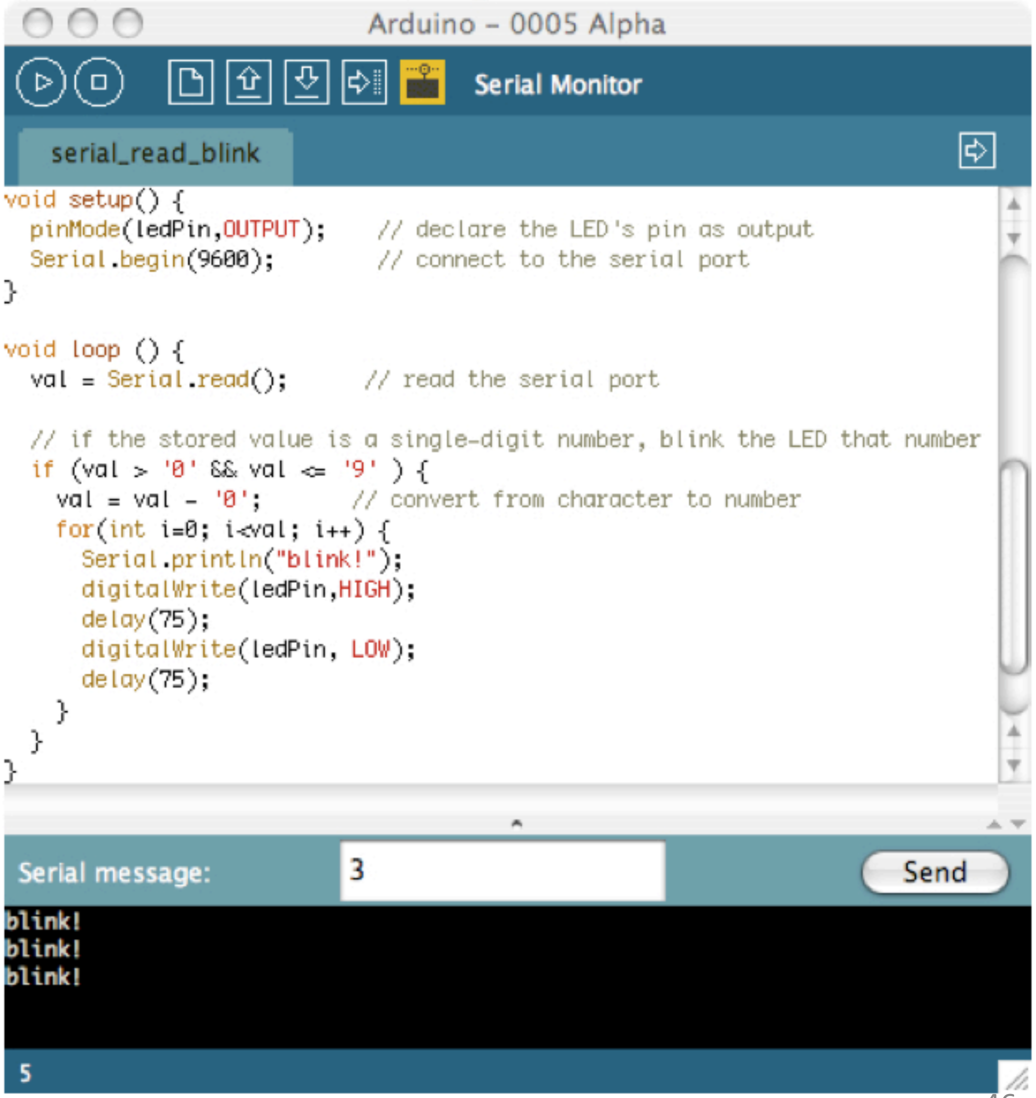
Controlling the computer

- 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.
- In this course we will use **Processing**

Example

"serial_read_blink"

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

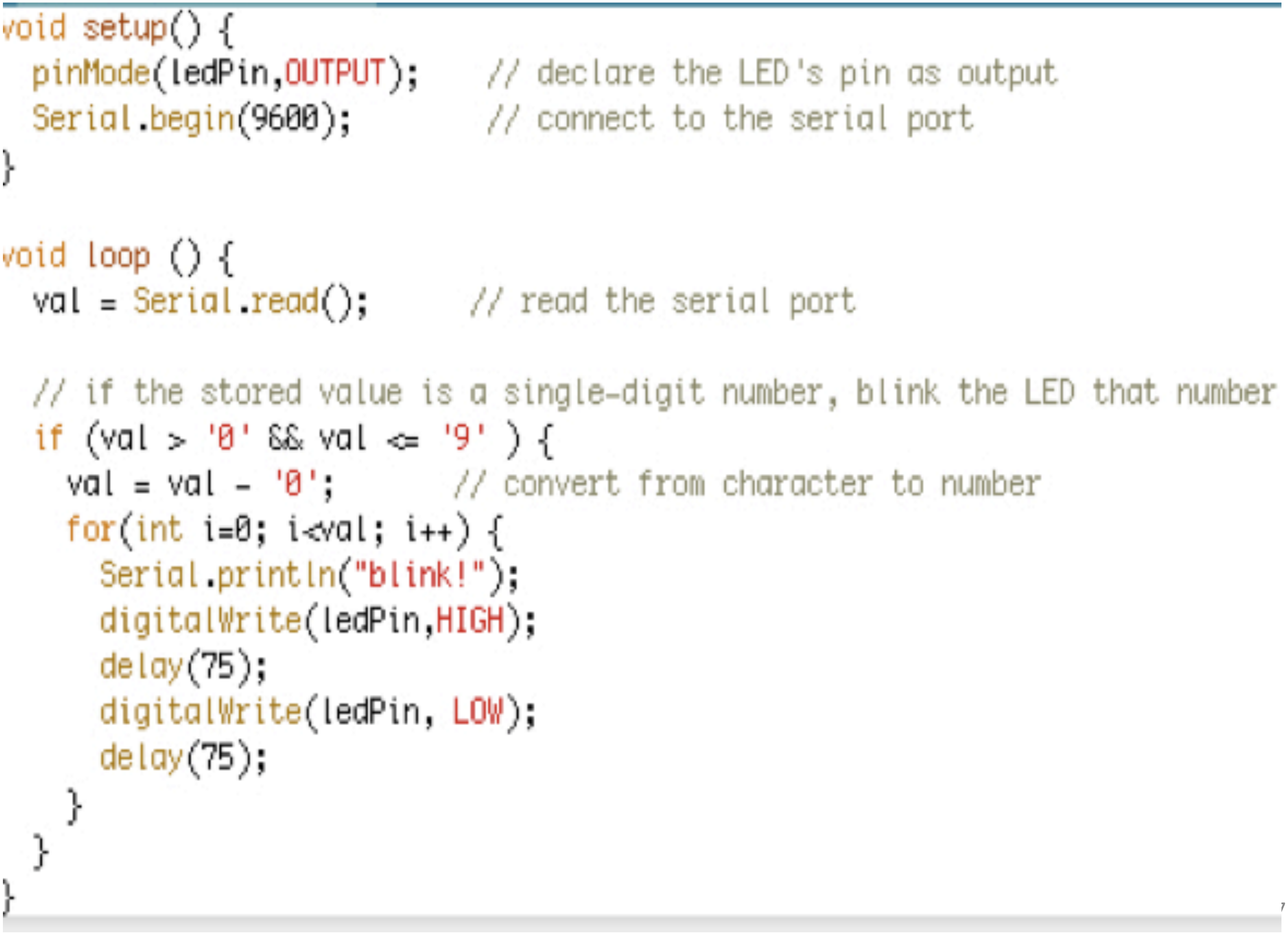


```
void setup() {
  pinMode(ledPin,OUTPUT);    // declare the LED's pin as output
  Serial.begin(9600);        // connect to the serial port
}

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

```
blink!
blink!
blink!
```

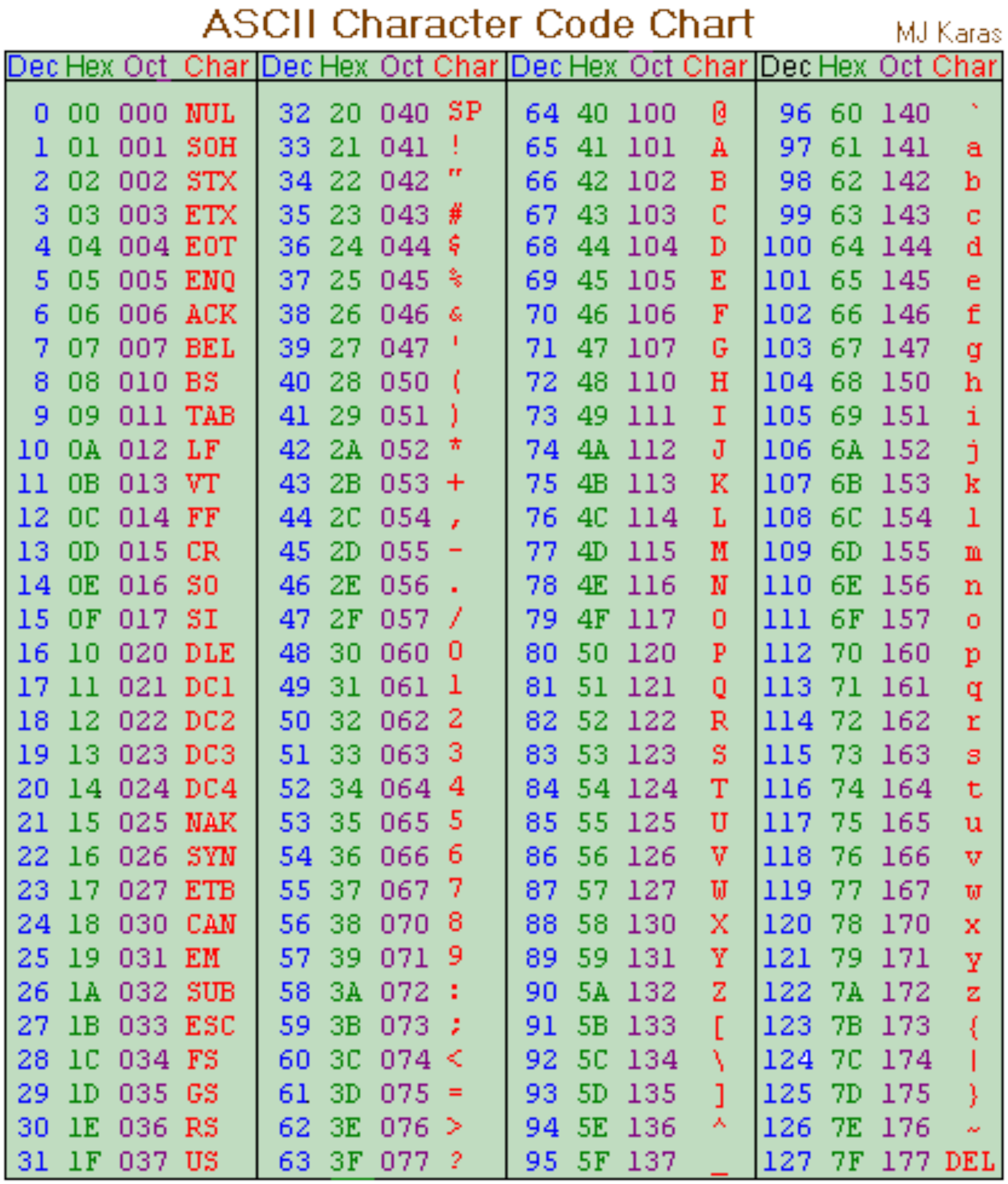
5



```
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);
    }
  }
}
```

char

- val =val – ‘0’ : converts from char to number.
- Characters are stored as numbers however. You can see the specific encoding in the ASCII cart.
- It is possible to do arithmetic on characters, in which the ASCII value of the character is used.



ASCII Character Code Chart

MJ Karas

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	00	000	NUL	32	20	040	SP	64	40	100	@	96	60	140	`
1	01	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	02	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	03	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	04	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	05	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	06	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	07	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	08	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	09	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	0A	012	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	0B	013	VT	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	0C	014	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	0D	015	CR	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	0E	016	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	0F	017	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Resources

- Arduino website: http://arduino.cc/
- Also see: http://arduino.cc/en/Tutorial/Links

Thank you

Questions?