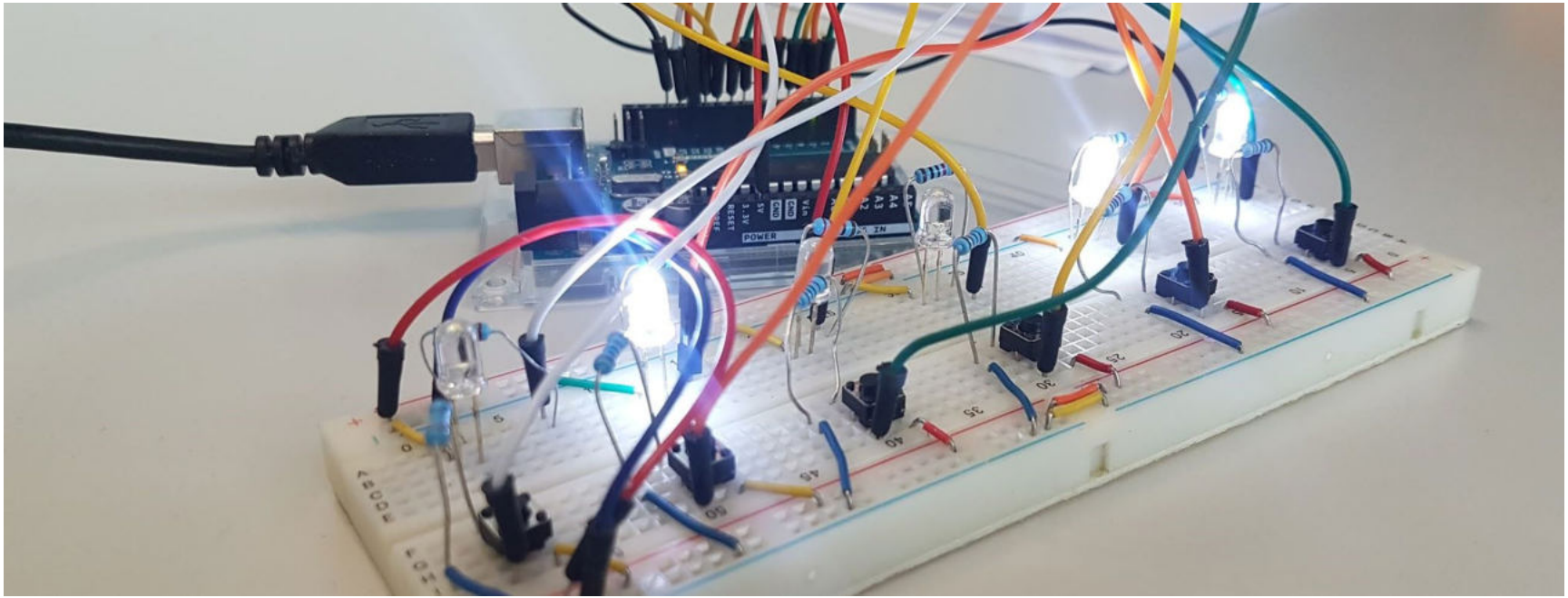
IAT 884 – Week 7 – Workshop 6

Alissa Antle and Annemiek Veldhuis (*ahv1@sfu.ca*)



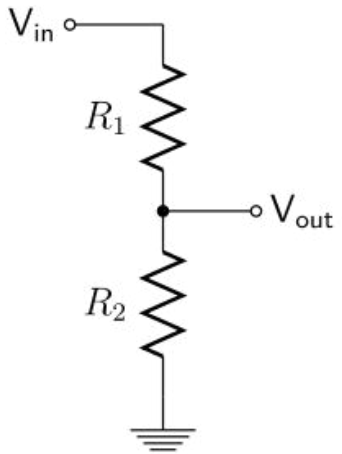
Recap

Recap

Voltage Divider

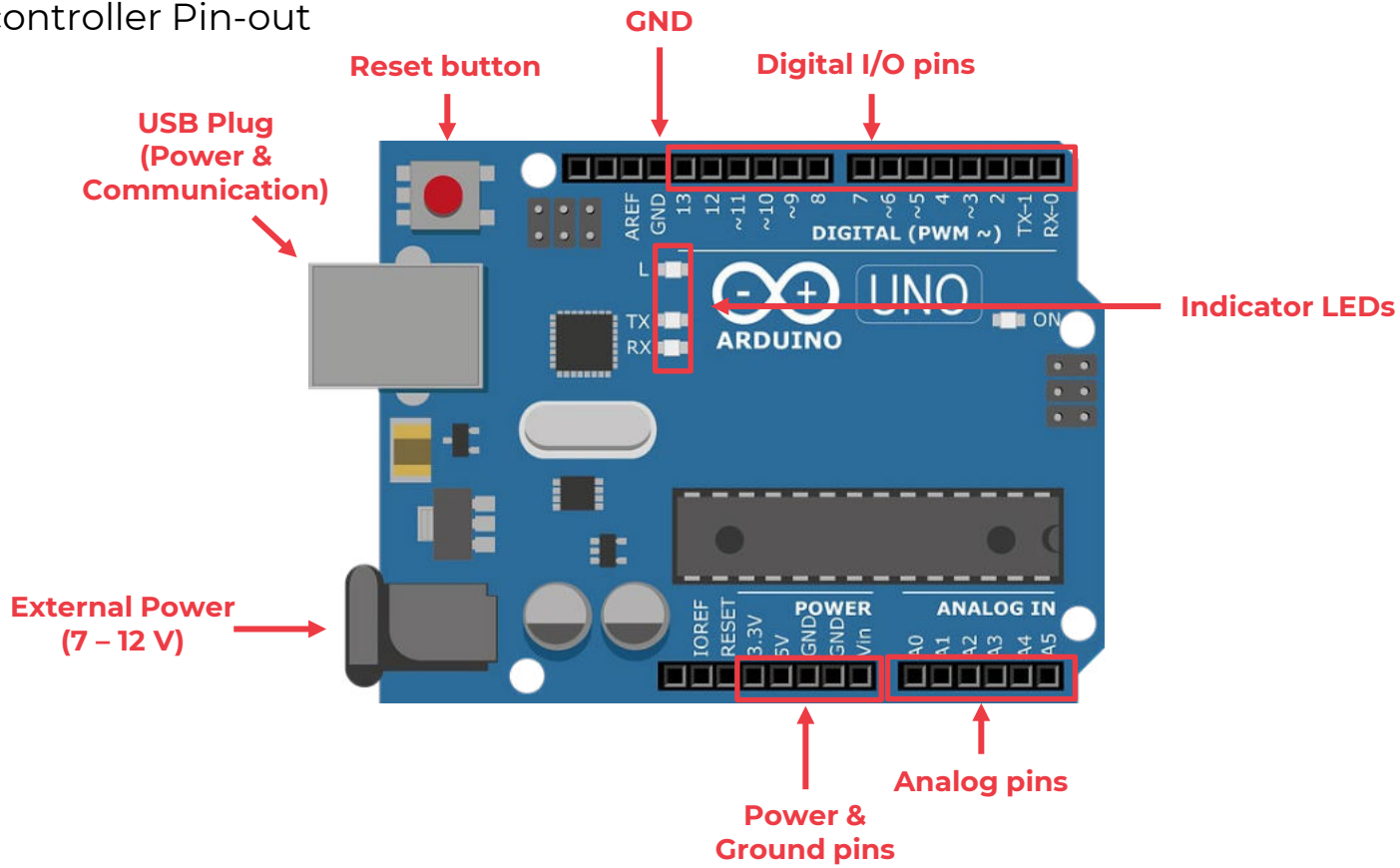
Voltage divider circuits are circuits that produce an output voltage (V_{out}) that is a fraction of its input voltage (V_{in}).

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}$$



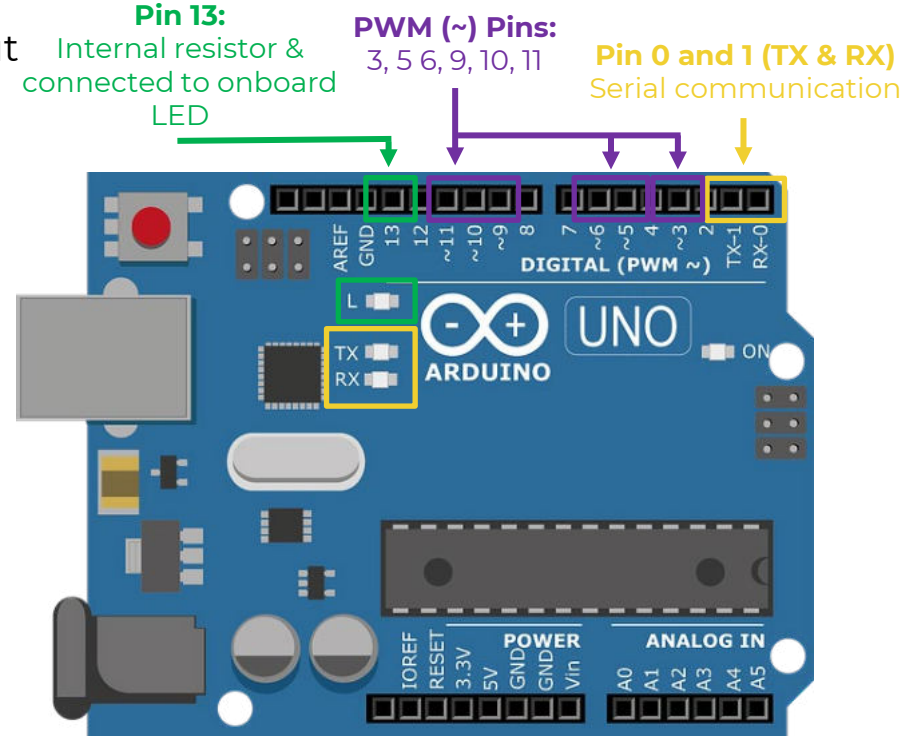
Recap

Microcontroller Pin-out



Recap

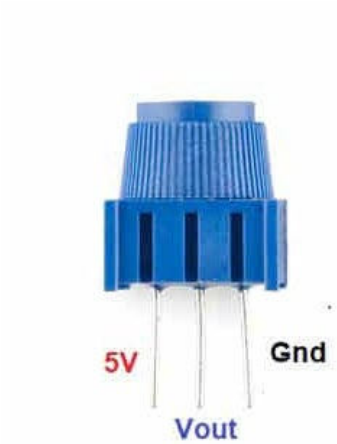
Microcontroller Pin-out



Recap

Digital & Analog I/O

digitalRead
digitalWrite
analogWrite



analogRead
analogWrite

Recap

Digital & Analog I/O

digital**Read**(pin);

input of 0V (<3V) = 0

input of 5V (>3V) = 1

digital**Write**(pin, KEYWORD);

output LOW = 0V

output HIGH = 5V

analog**Read**(pin);

Reads a number between 0-1023

0V is 0

5V is 1023

analog**Write**(pin, value);

Writes a number between 0 – 255

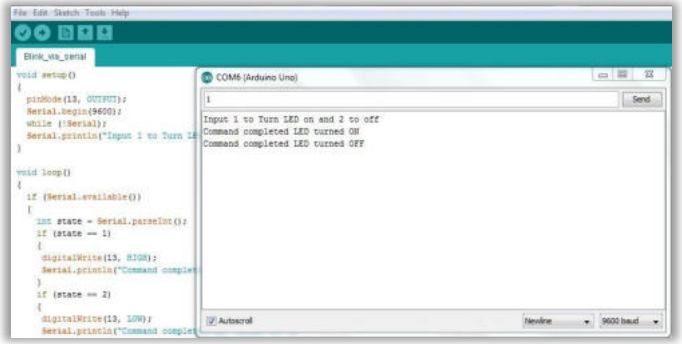
0 is 0V

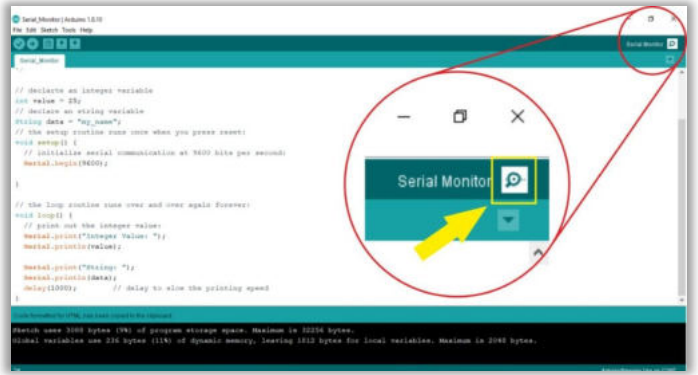
255 is 5V

Debugging

Serial monitor

Serial.print() and **Serial.println()**





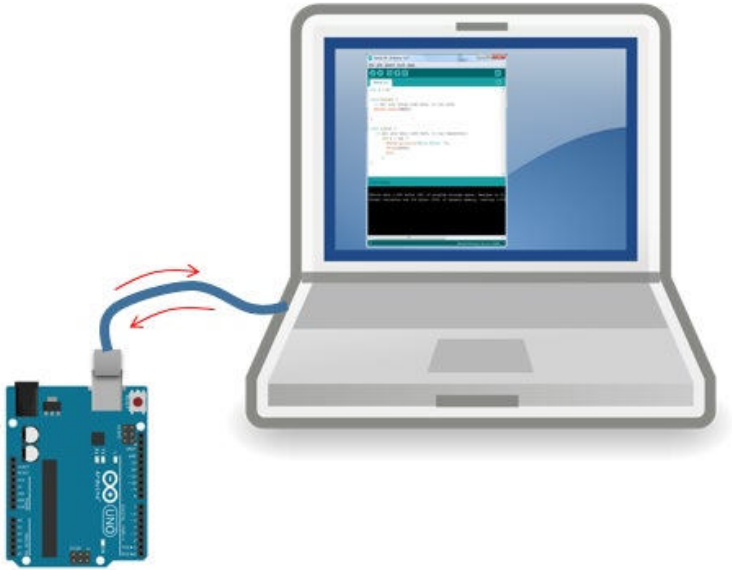
Serial Communication

Arduino

Serial Communication

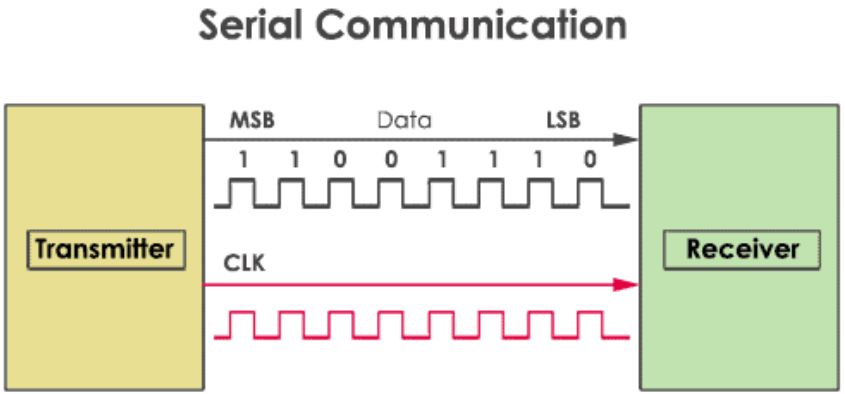
The most common form of communication between electronic devices is **serial communication**.

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



Serial Communication

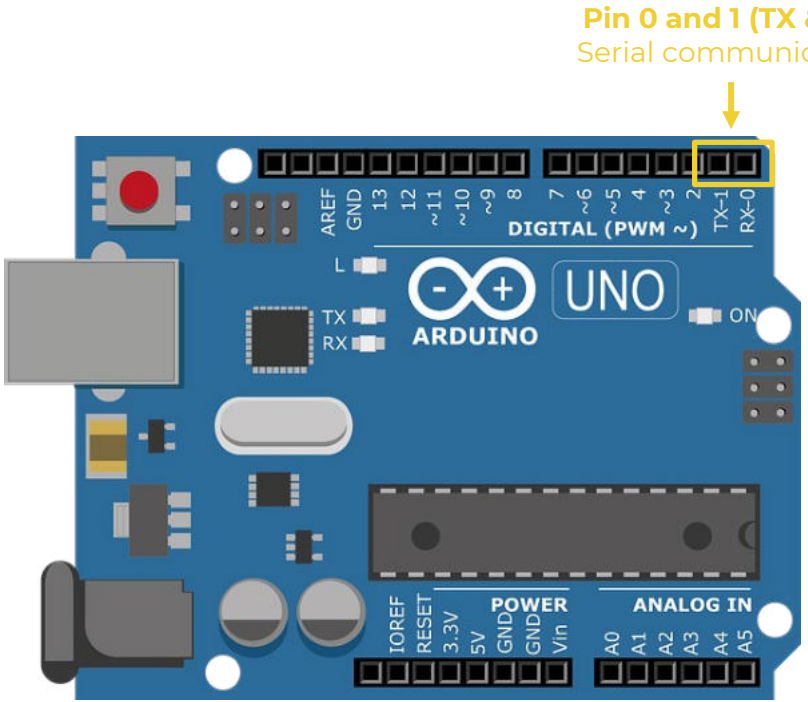
Serial data transfer is when we transfer data one bit at a time, one right after the other.



Serial Communication
Arduino

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

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



Serial Communication

Protocols

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

- Physical: Serial port
- Timing: Speed
- Electrical: Voltage level
- Package size: Bits per package

Serial Communication

Arduino Serial Library

https://www.arduino.cc/reference/en/language/functions/communication/serial/

Functions

- if(Serial)
- available()
- availableForWrite()
- begin()
- end()
- find()
- findUntil()
- flush()
- parseFloat()
- parseInt()
- peek()
- print()
- println()
- read()
- readBytes()
- readBytesUntil()
- readString()
- readStringUntil()
- setTimeout()
- write()
- serialEvent()

Serial Communication

Arduino Serial Library

Step 1: Initialize in **Setup**

```
// initialize serial communication at 9600 bits per second:
Serial.begin(9600);
```

Step 2: Print text and values in **Loop**

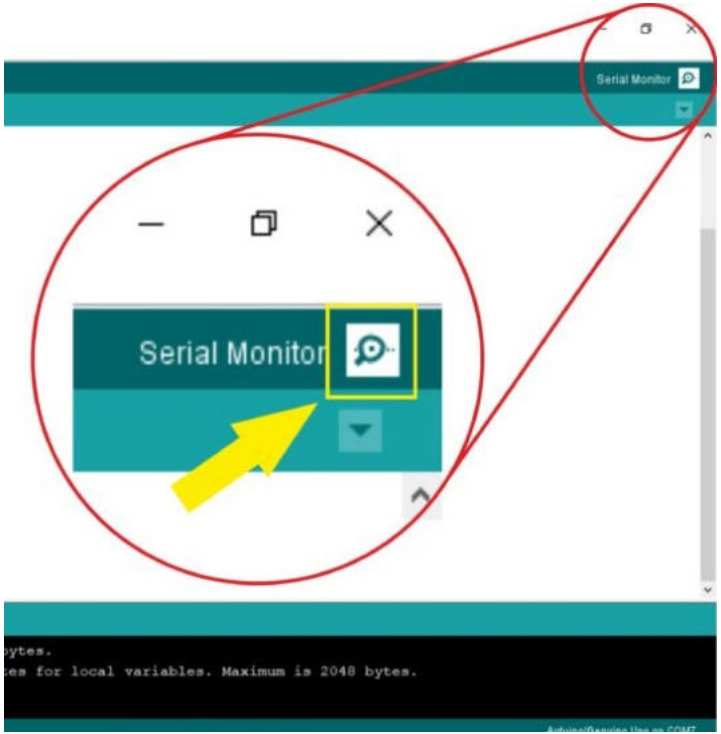
```
Serial.print("Integer Value: ");
Serial.println(value);
```

Serial Communication

Arduino Serial Library

Step 3: Run code

Step 4: open the Serial Monitor



Serial Communication

Arduino Serial Library

Prints data to the serial port as **human-readable ASCII text**.

Numbers are printed using an ASCII character for each digit.
Floats are similarly printed as ASCII digits, defaulting to two decimal places.
Bytes are sent as a single character.
Characters and strings are sent as is.

Serial.print(78) gives "78"

Serial.print(1.23456) gives "1.23"

Serial.print('N') gives "N"

Serial.print("Hello world.") gives "Hello world."

Serial Communication

Arduino Serial Library

When you want to send the integer value 154 you write the command:

Serial.print(154);

The Arduino then sends every single digit as a separate information package:

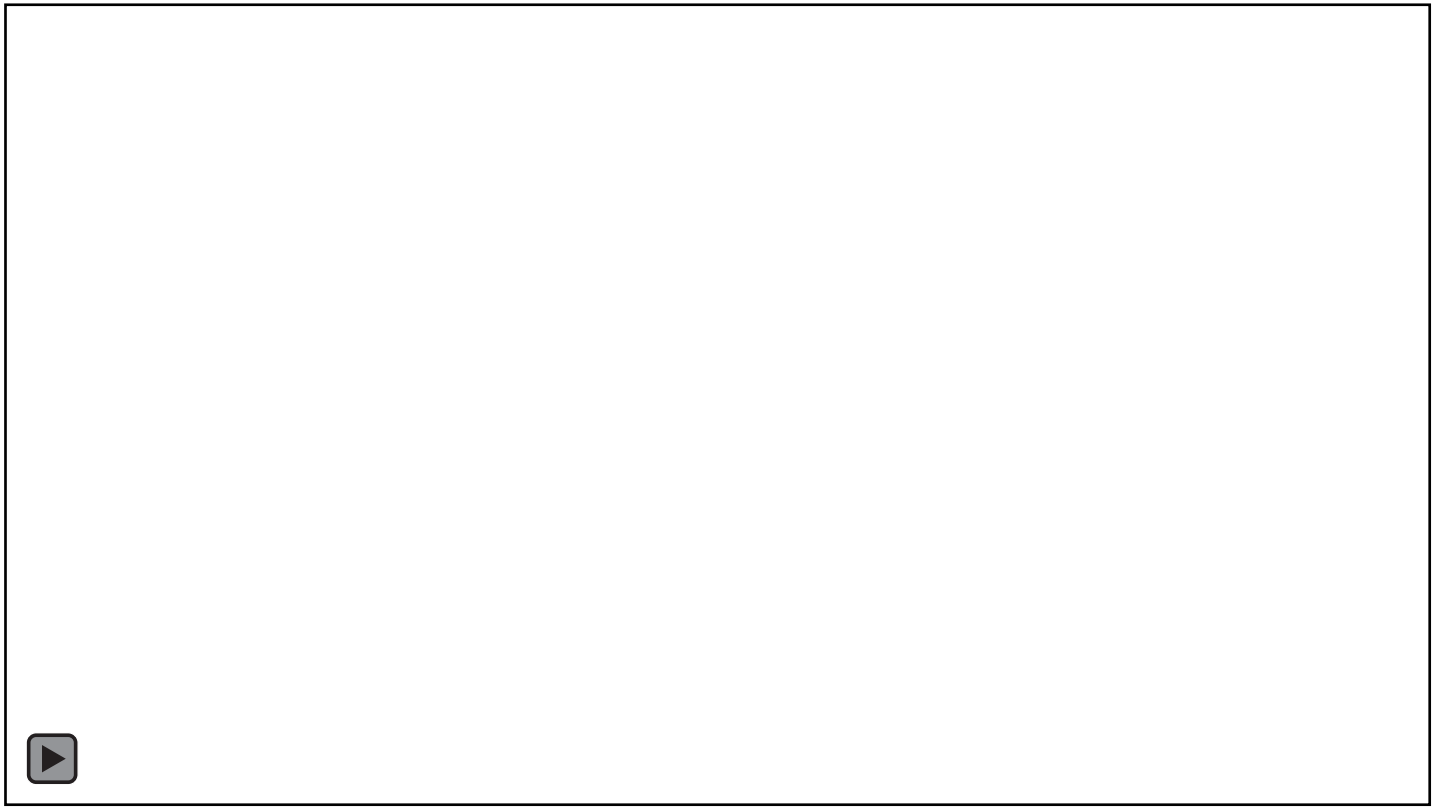
‘1’, ‘5’, ‘4’.

Each package corresponds to a specific **ASCII character** that has a value between 0 -255 (1 Byte).

In this case:

‘1’ = 49
‘5’ = 53
‘4’ = 52

Serial Communication



Serial Communication

Arduino Serial Library

An optional second parameter specifies the base (format) to use;
- BIN(binary, or base 2),
- OCT(octal, or base 8),
- DEC(decimal, or base 10),
- HEX(hexadecimal, or base 16).
- For floating point numbers, this parameter specifies the number of decimal places to use.

Serial.print(78, BIN) gives "1001110"

Serial.print(78, OCT) gives "116"

Serial.print(78, DEC) gives "78"

Serial.print(78, HEX) gives "4E"

Serial.print(1.23456, 0) gives "1"

Serial.print(1.23456, 2) gives "1.23"

Serial.print(1.23456, 4) gives "1.2345"

Serial Communication

Arduino Serial Library

Serial.print(); >

Serial.println();

Prints data to the serial port ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n').

Serial Communication

Arduino Serial Library

Serial.read();

Reads incoming serial data

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

Serial Communication

Arduino Serial Library

Serial.parseInt();

Looks for the next valid integer in the incoming serial data.

Serial Communication

Arduino Serial Library

Serial.available();

Get the number of bytes (characters) available for reading from the serial port.

This is data that has already arrived and is stored in the serial receive buffer (which holds 64 bytes).

Serial Communication

Arduino Serial Library

```
int incomingByte = 0;

void setup() {
        Serial.begin(9600);
}

void loop(){
        if (Serial.available() > 0){
                incomingByte = Serial.read();

                Serial.print("I received: ");
                Serial.println(incomingByte);
        }
}
```

Serial Communication

Arduino Serial Library

```
Int ledPin = 13;
int val = 0;

void setup() {
        pinMode(ledPin, OUTPUT);
        Serial.begin(9600);
}

void loop(){
        if (Serial.available() > 0){
                val = Serial.read();
                if ( val == 'H' ){
                        digitalWrite(ledPin, HIGH);
                } else {
                        digitalWrite(ledPin, LOW);
                }
        }
}
```

Serial Communication

Arduino Serial Library

```
int ledPin = 13;
int val = 0

void setup() {
        pinMode(ledPin,OUTPUT);
        Serial.begin(9600);
}

void loop () {
        val = Serial.read();
        if (val > '0' && val <= '9' ) {
                val = val - '0';
                for(int i=0; I < val; i++) {
                        Serial.println("blink!");
                        digitalWrite(ledPin,HIGH);
                        delay(150);
                        digitalWrite(ledPin, LOW);
                        delay(150);
                }
        }
}
```

Serial Communication

Arduino Serial Library

val = val – ‘0’ : converts from an ASCII character or char to number.

val = val – 48

Char:

A data type used to store a character value. Written like this: ‘A’. They are stored as numbers:

char myChar = 'A';
char myChar = 65; // both are equivalent

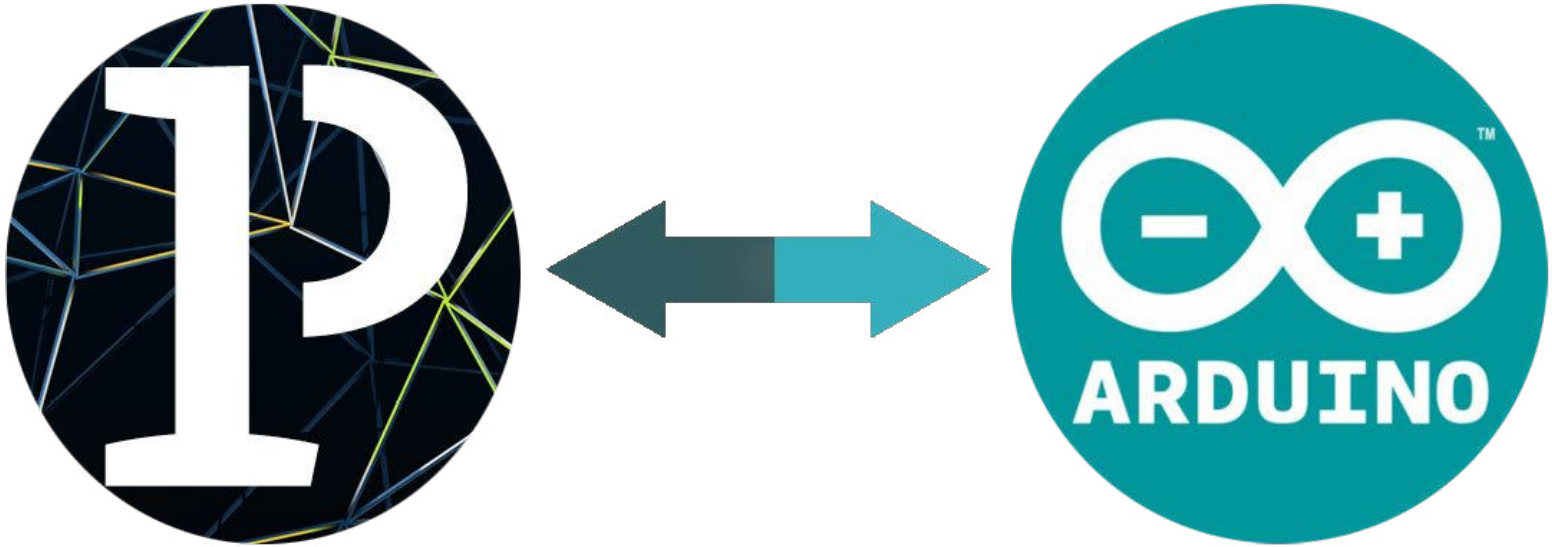
ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Serial Communication

Processing

Serial Communication



Serial Communication

There are several functions that you can use to receive serial data in Processing:

read(): Returns a number between 0 and 255 for the next byte that's waiting in the buffer.

readChar(): Returns the next byte in the buffer as a char.

readString(): Returns all the data from the buffer as a String. This method assumes the incoming characters are ASCII.

**Note that in this context a character value can be assigned to an integer variable (e.g. 'A' = 65), but that an integer value cannot be assigned to a character variable.*

Serial Communication

```
import processing.serial.*;

Serial myPort;

void setup() {:
        printArray(Serial.list());
        myPort = new Serial(this, Serial.list()[0], 9600);
}

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

Serial Communication

Bytes are sent out via the serial port using the **write( )** function.

- The following data types can be sent:
  byte, char, int, byte array, and String.

- Remember, if you are sending a String, the actual data sent are raw ASCII byte values of each character.

Serial Communication

```
import processing.serial.*;
Serial myPort;

void setup() {
        println(Serial.list());
        myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw(){
        myPort.write(65);
}
```

Exercises

Exercises

Download the Week 6 Handout on the wiki.

Get as far as possible with the exercises during the lecture time. Complete them at home if you can't finish.

Send a document with your name, photos of your circuits, and answers to the questions to ***ahv1@sfu.ca***.
Deadline: Sunday 11.59pm