Practical Electronics & Programming

with Arduino

Session 2: Communicate

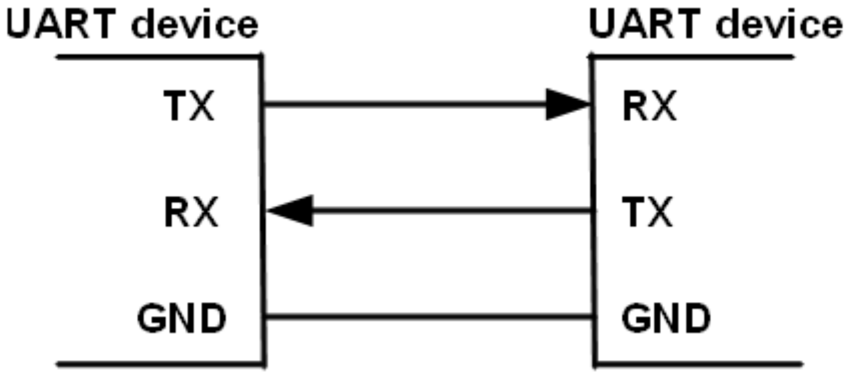
Review Last Class

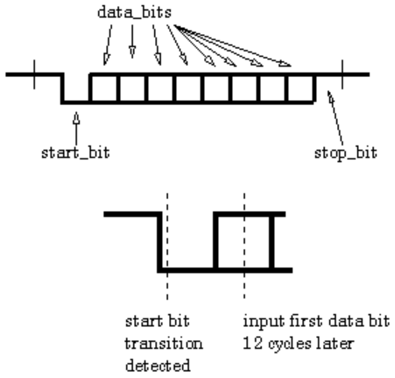
- Check homework projects

Session 2 Overview

- Hello Serial
- Custom outputs
- Simple Inputs
- Parsing Inputs

What is Serial?




data_bits
start_bit
stop_bit
start bit transition detected
input first data bit 12 cycles later

- How Arduino communicates with computer
- Old protocol implemented virtually on USB
- Two lines - Receive and Transmit (relative to device)
- Data sent in discrete chunks, usually 8 bits at a time

Things to know

- Baud rate - speed of transmit/receive
- Need same baud rate on both devices, otherwise gibberish
- Common baud rates:
  110, 150, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600
- Defacto standard settings (95%+ the time):
  9600 baud date 8 data bits, no parity, one stop bit (9600/8-N-1)

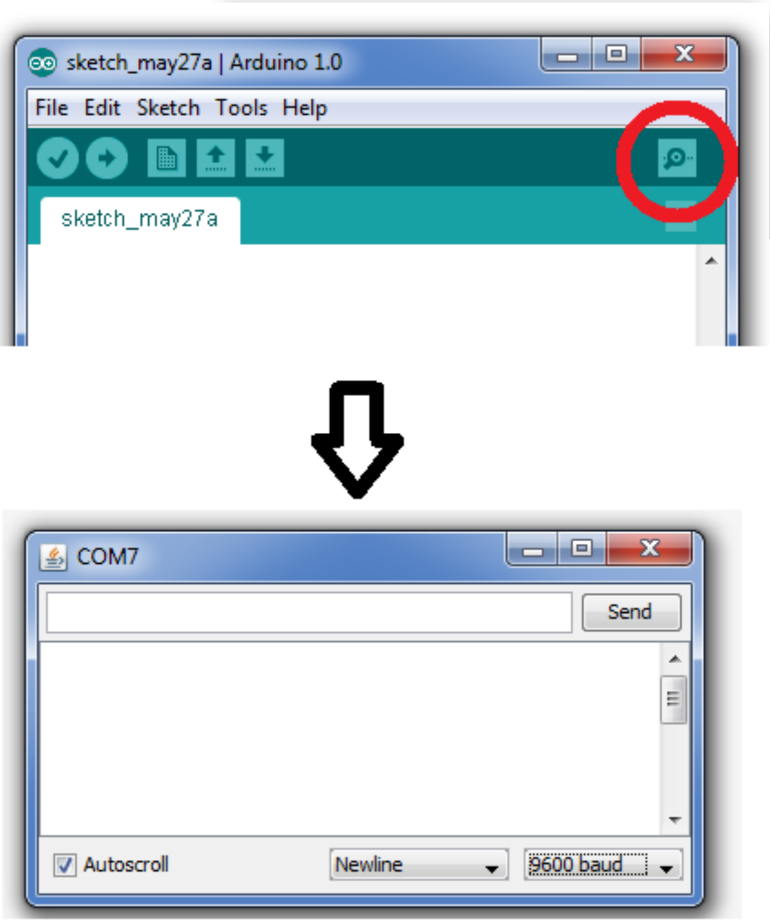
Things to know (Cont.)

As baud rate goes up:

- Faster data transfer
- More data transfer
- More processing power used
- Less reliable

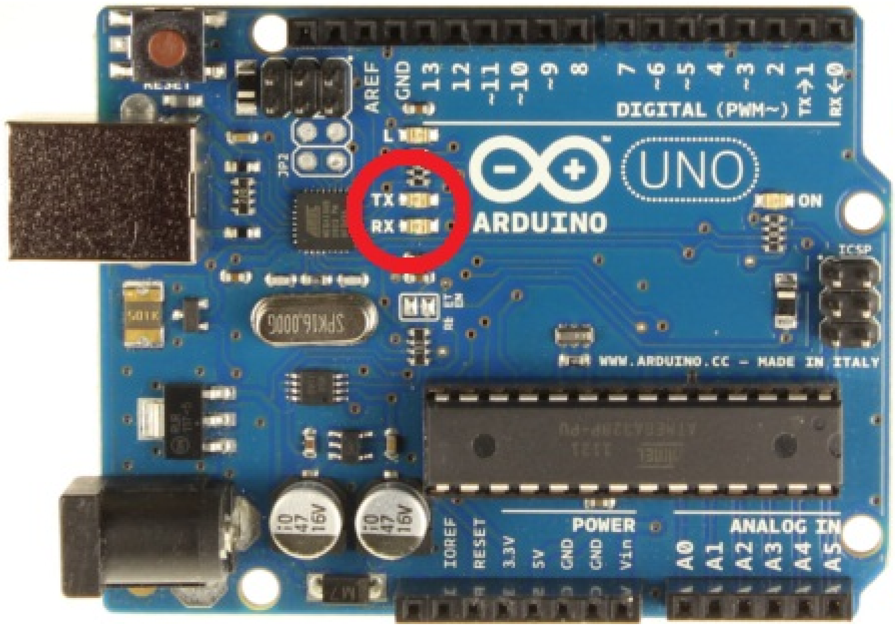
Serial on Arduino

Serial Monitor:



☑ Autoscroll   Newline   9600 baud

Serial TX /RX lights:

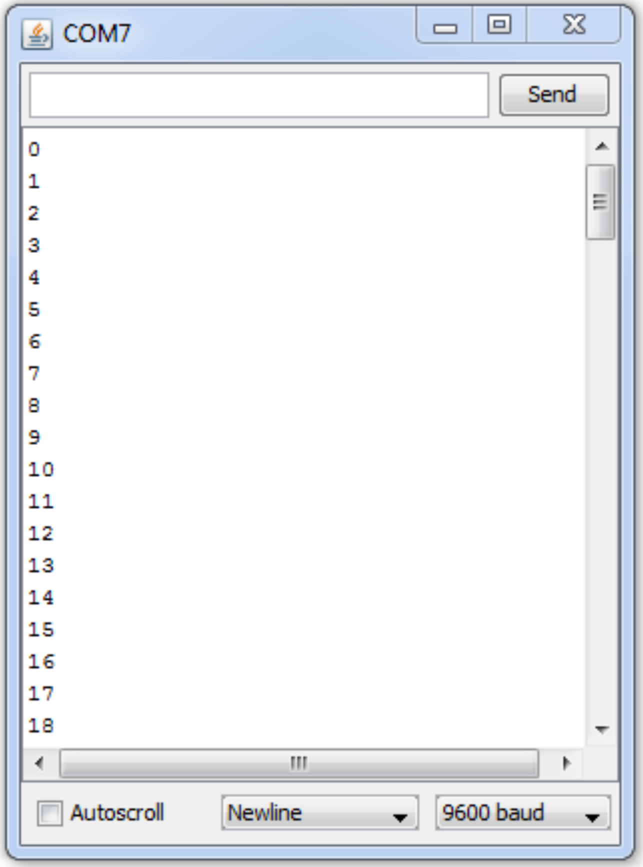


First Example

```
void setup() {
  Serial.begin(9600);        //Initialize serial and wait for port to open
}

int count = 0;
void loop() {
  Serial.println(count);     //print a character with a newline
  delay(500);                //wait 0.5 seconds (500 milliseconds)
  count = count + 1;         //increment count
}
```

First Example



```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

☐ Autoscroll

Newline

9600 baud

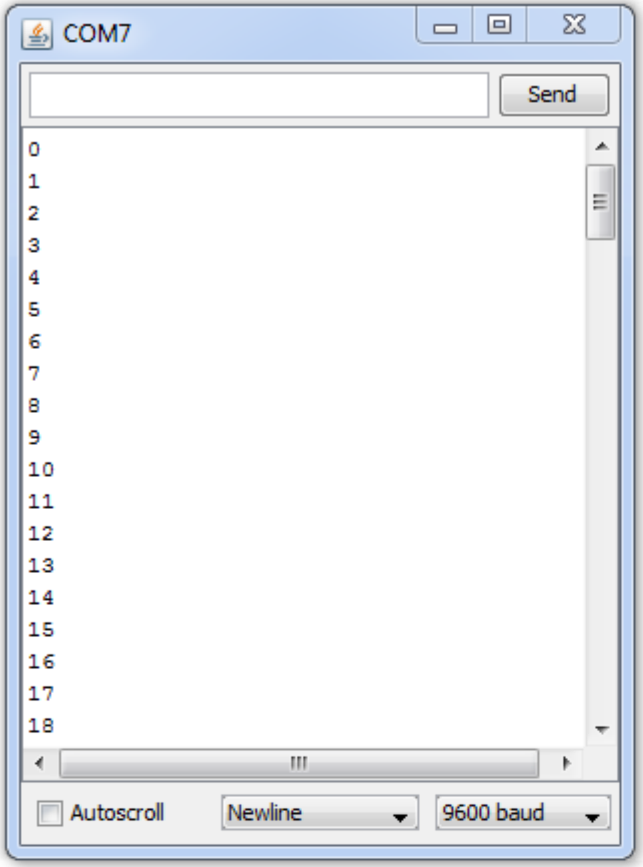
Should output increasing numbers forever, one every 0.5 seconds

While Loop

```
void setup() {
  Serial.begin(9600);         //Initialize serial and wait for port to open
}

int count = 0;
void loop() {
  while(true)
                  <-- loops forever, because the 'condition' is always true
  {
    Serial.println(count);    //print a character with a newline
    delay(500);               //wait 0.5 seconds (500 milliseconds)
    count = count + 1;        //increment count
  }
}
```

First Example



```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

□ Autoscroll

Newline

9600 baud

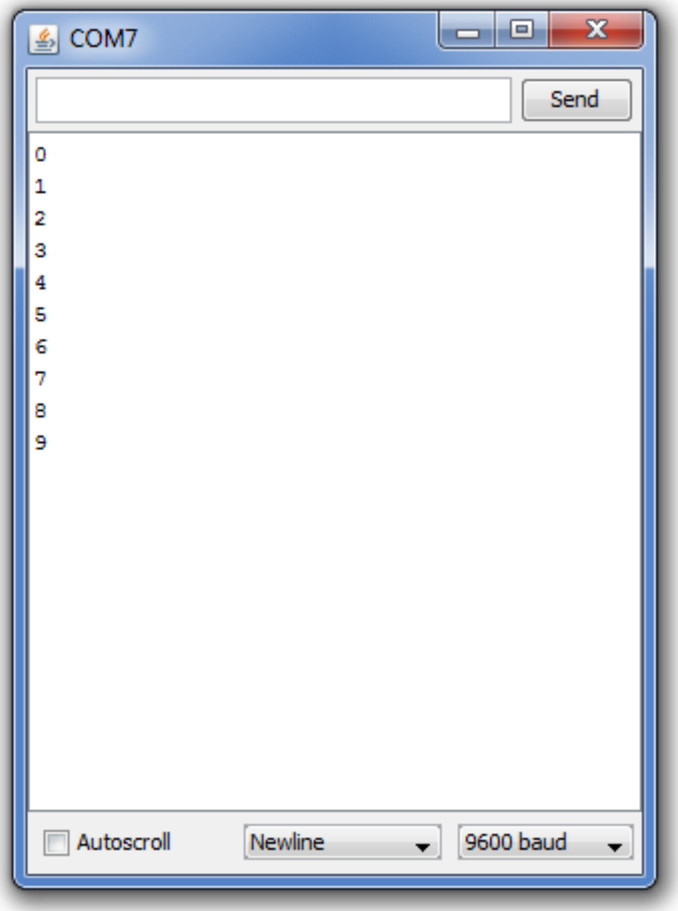
Should output increasing numbers till 9, one every 0.5 seconds, then loop back and repeat

While Loop

```
void setup() {
  Serial.begin(9600);         //Initialize serial and wait for port to open
}

int count = 0;
void loop() {
  while(count < 10)
                              <-- only counts up to 9, because then the count < 10 condition is false
  {
    Serial.println(count);    //print a character with a newline
    delay(500);               //wait 0.5 seconds (500 milliseconds)
    count = count + 1;        //increment count
  }
}
```

While Loop



```
0
1
2
3
4
5
6
7
8
9
```

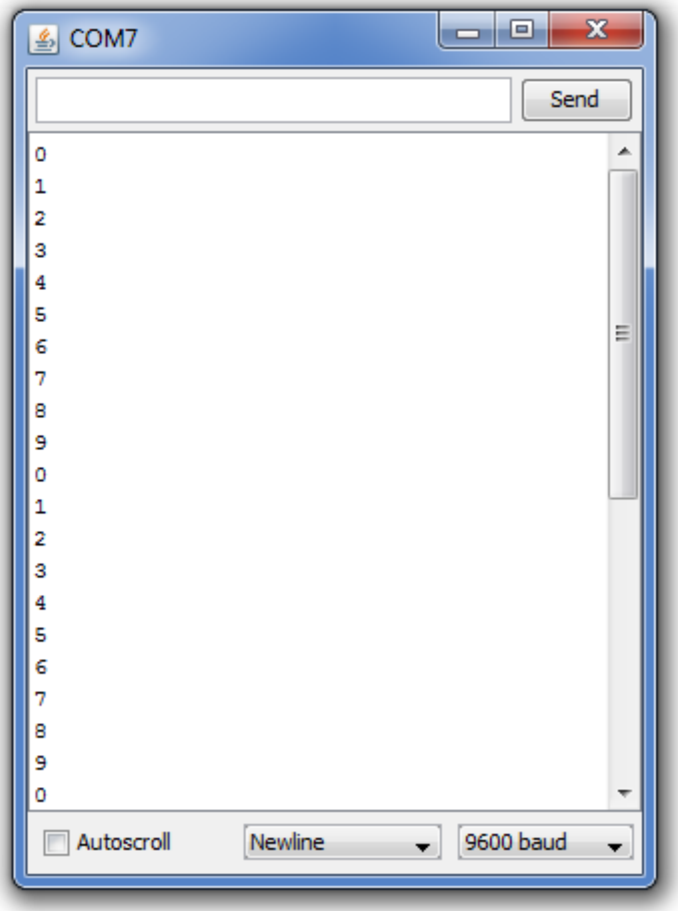
Should output increasing numbers till 9, one every 0.5 seconds, then stop

While Loop

```
void setup() {
  Serial.begin(9600);        //Initialize serial and wait for port to open
}

void loop() {
  int count = 0;       <-- now resets to zero after reaching 10
  while(count < 10)
  {
    Serial.println(count);   //print a character with a newline
    delay(500);              //wait 0.5 seconds (500 milliseconds)
    count = count + 1;       //increment count
  }
}
```

While Loop



```
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
0
```

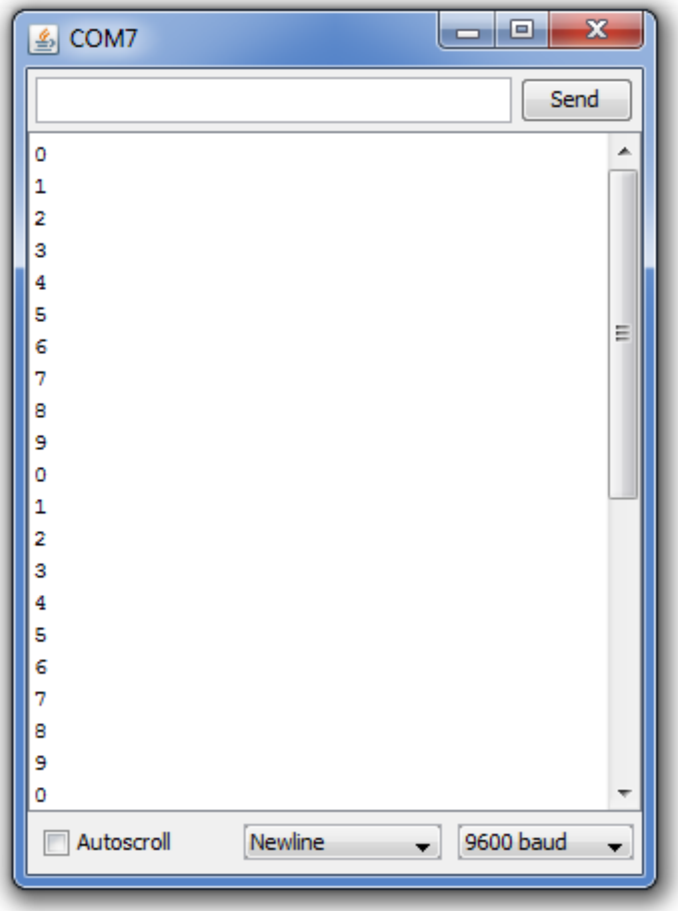
Should output increasing numbers till 9, one every 0.5 seconds, repeat

For Loop

```
void setup() {
  Serial.begin(9600);          //Initialize serial and wait for port to open
}

void loop() {                  //loop repeats infinitely
  for(int count=0; count < 10; count++)
  {
    Serial.println(count);     //print a character with a newline
    delay(500);                //wait 0.5 seconds (500 milliseconds)
  }
}
```

While Loop



```
0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
0
```

Should output increasing numbers till 9, one every 0.5 seconds, repeat

Variable Types

There are different types of 'variables', ways computers store data:

Data Type	Size in bytes (1 byte = 8 bits, or 1/0's)	Max/Min value (signed)	Max value (unsigned)
char	1 byte (8 bits)	-128 ... 127	0 ... 255
byte	1 byte (8 bits)	-128 ... 127	0 ... 255
short	2 bytes (16 bits)	-32,768 ... 32,767	0 ... 65,535
int	4 bytes (32 bits)	2,147,483,648 ... 2,147,483,647	0 ... 4,294,967,295
long	8 bytes (64 bits)	-4.61x10^{8} ... 4.61x10^{8}	0 ... 9.22x10^{8}
float	4 bytes (32 bits)	-3.4x10^{38} ... 3.4x10^{38}	N/A
double	8 bytes (64 bits)	-4.9x10^{324} ... 4.9x10^{324}	N/A

Variable Tradeoffs

More bits:

- Slower
- More data
- More precision for floating
- More likely to overflow

Less bits:

- Faster
- Less data
- Less precision for floating
- Less likely to overflow

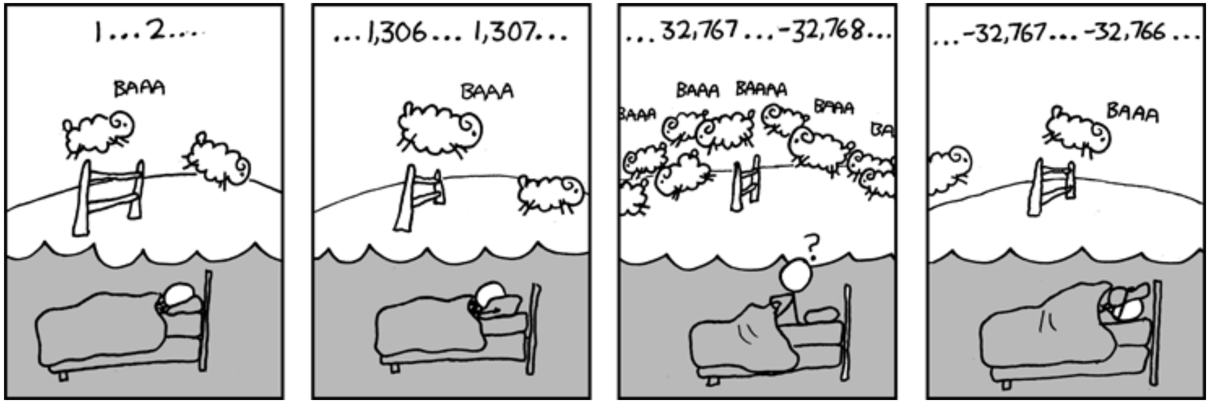
Use what size you have to, no more. But ONLY AFTER IT WORKS, and ONLY IF YOU NEED TO.

"Premature optimization is the root of all evil in programming."
- Donald Knuth, Computer Science Legend

Overflow

- Variables have 'max values', they can only store numbers so large.
- Go over this value, and they go back to their lowest value, THIS MESSES UP CODE BAD.

Example: should have used an int (32 bits) instead of a short (16 bits)!



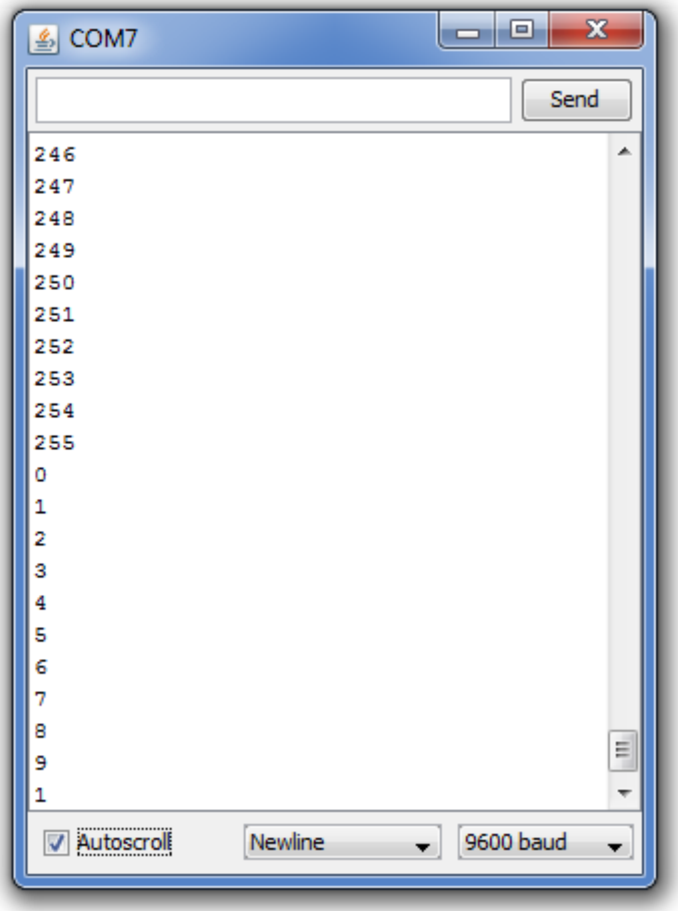
Overflow Example

```
void setup() {
  Serial.begin(9600);         //Initialize serial and wait for port to open
}

byte count = 0;               //only using 8 bits to store data (max of 255)
void loop() {
      while(count < 300) //should stop at 300, but won't because it will overflow
                         //at 255 back to zero
      {
         Serial.println(count);      //print a character with a newline
         delay(20);           //wait 0.02 seconds (20 milliseconds)
         count = count + 1;   //increment count
      }
}
```

Overflow Example

Overflows, and so will loop forever. It never falses on the < 300 condition to tell the while loop to stop.



```
246
247
248
249
250
251
252
253
254
255
0
1
2
3
4
5
6
7
8
9
1
```

Different Outputs

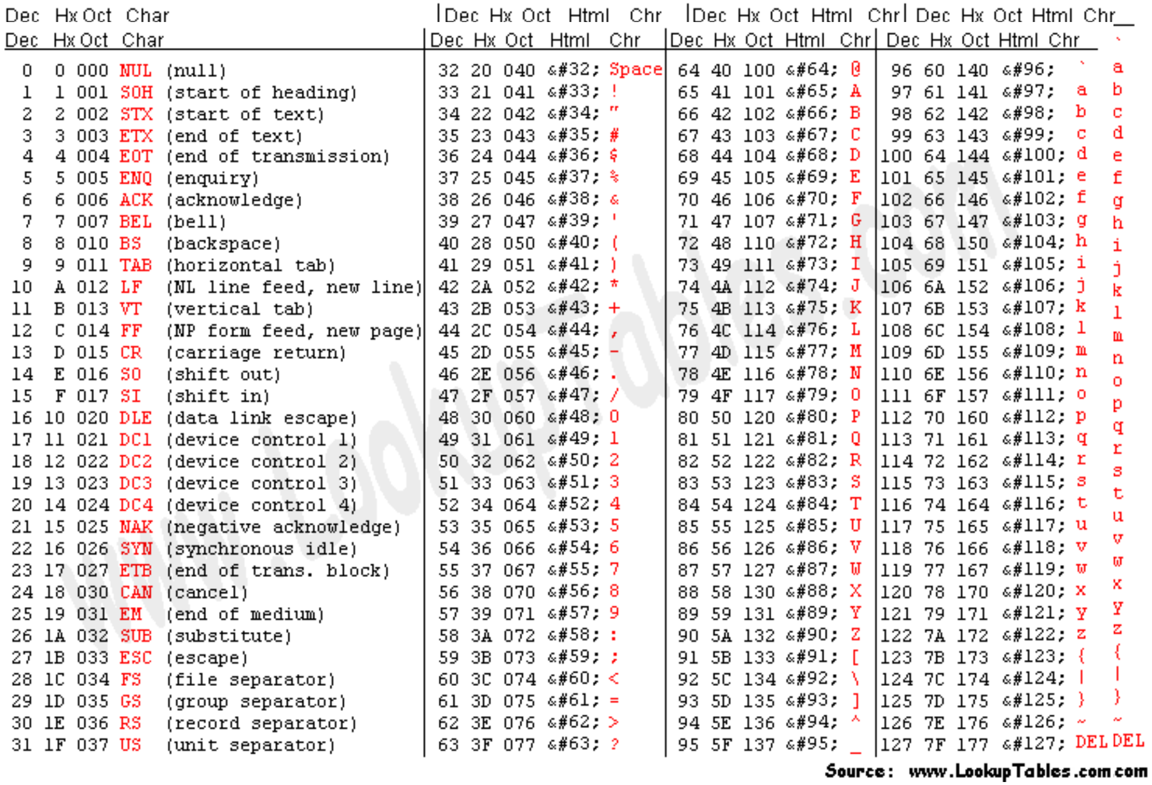
```
void setup() {
  Serial.begin(9600);           //Initialize serial and wait for port to open
}

int count = 0;

void loop() {
  Serial.println(count);        //print a character with a newline
  delay(500);                   //wait 0.5 seconds (500 milliseconds)
  count = count + 1;            //increment count
}
```

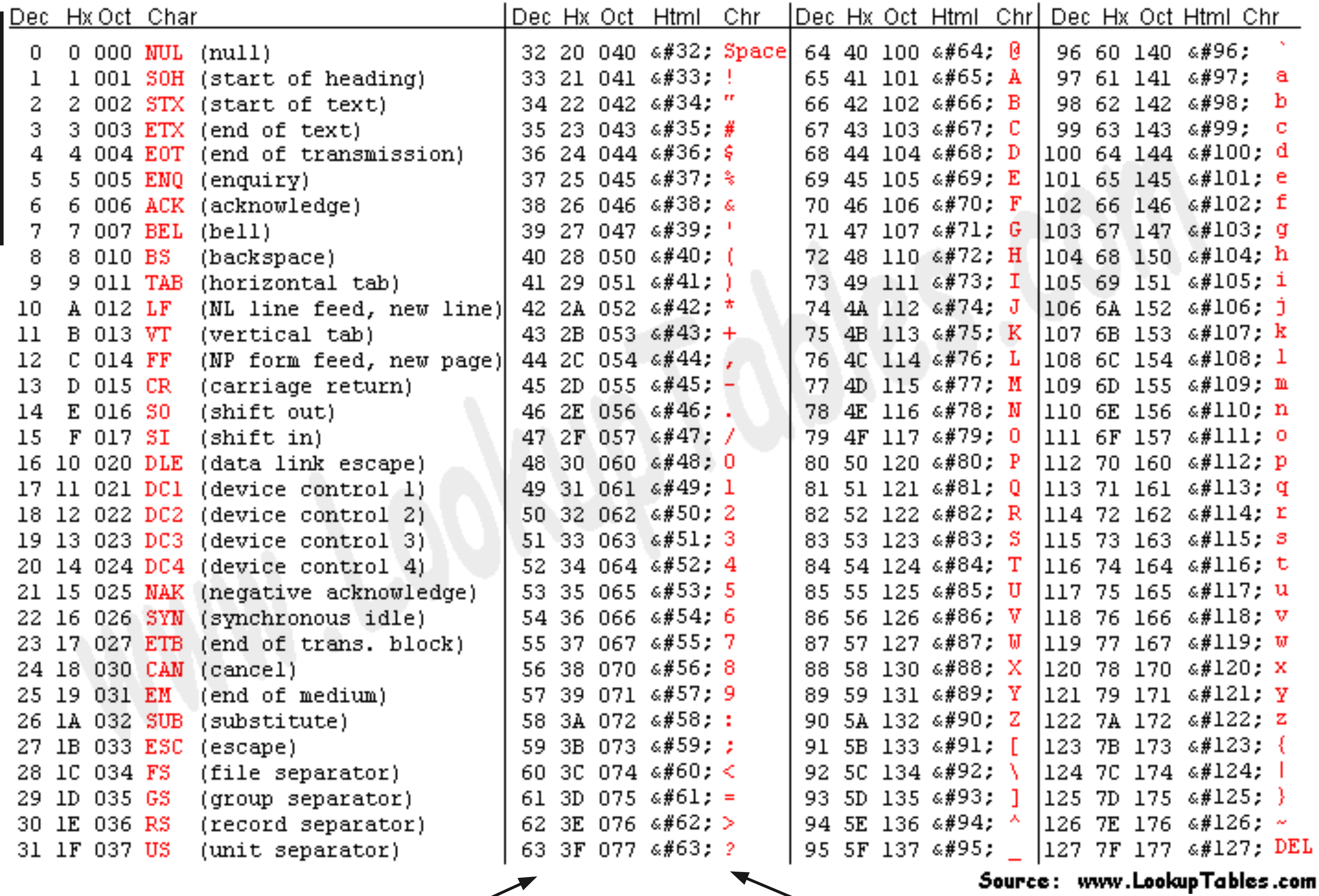
What are Characters

Characters are simply another way to think of bytes, mapping the number to the ASCII table:



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

Source: www.LookupTables.com com



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

Source: www.LookupTables.com

Decimal Number

Character

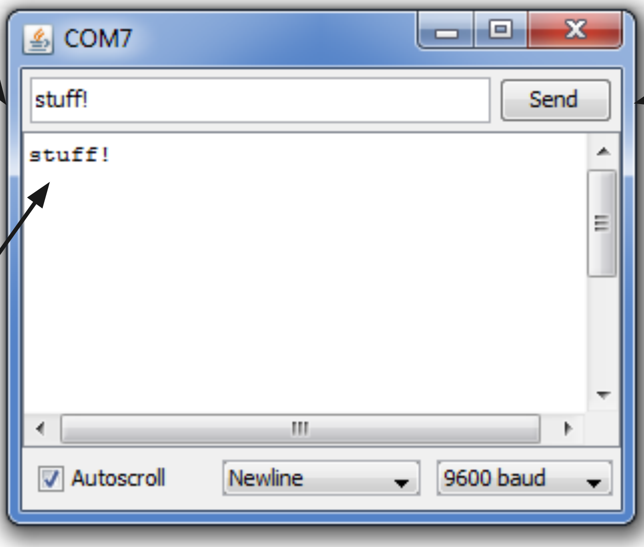
Input->Output Example

```
void setup() {
  Serial.begin(9600);        //Initialize serial and wait for port to open
}

char incoming;               //Create a temporary storage character (1 byte)
void loop() {
    if(Serial.available() != 0){   //If there is incoming Serial Data
        {
        incoming = Serial.read(); //Read a char from the serial line
        Serial.print(incoming); //Print it back
        }
}
```

Input->Output Example

1 - Type stuff here



COM7

stuff!

```
stuff!
```

☑ Autoscroll

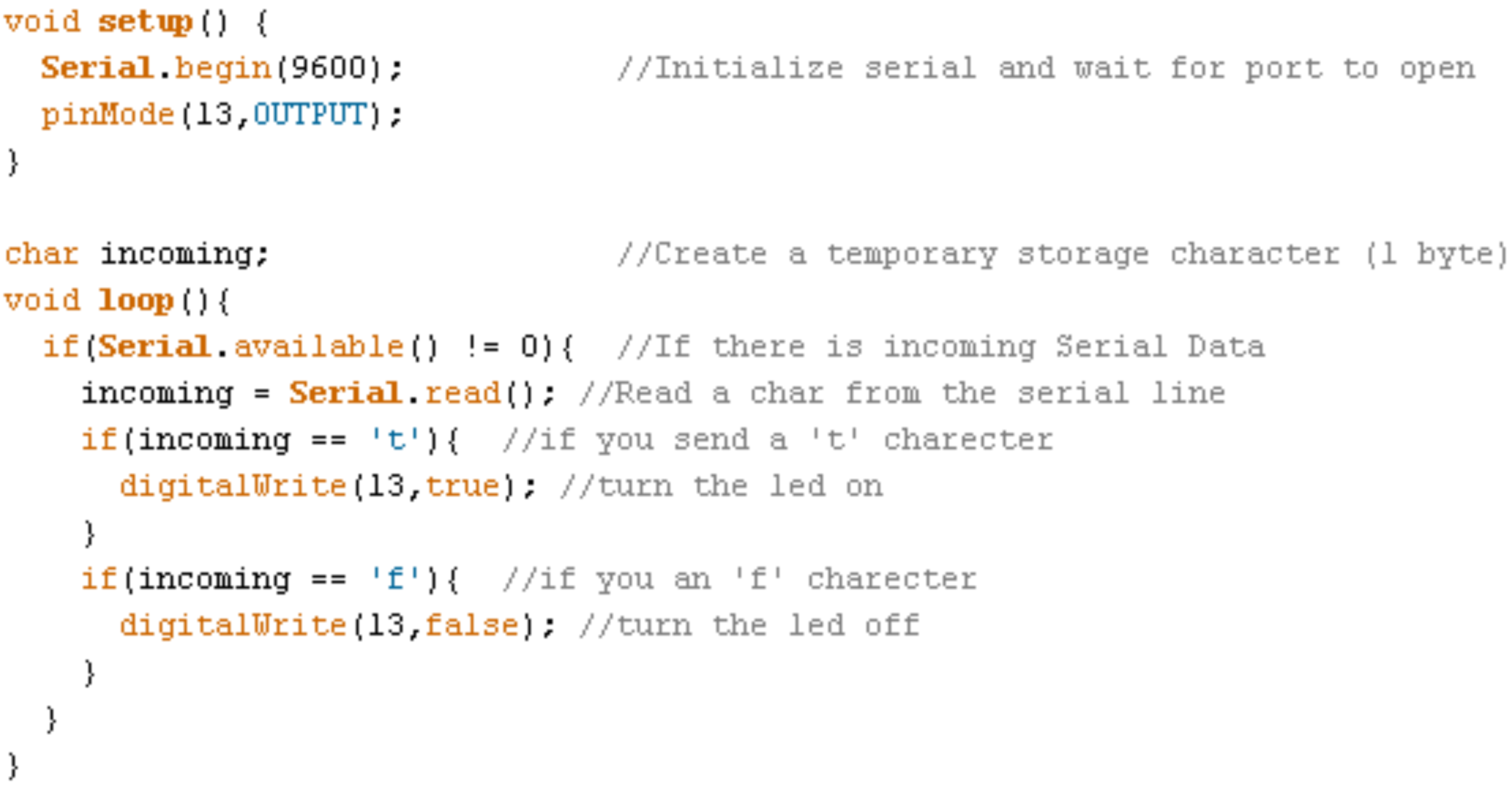
Newline

9600 baud

2 - Send it to the Arduino

3 - Arduino sends it back

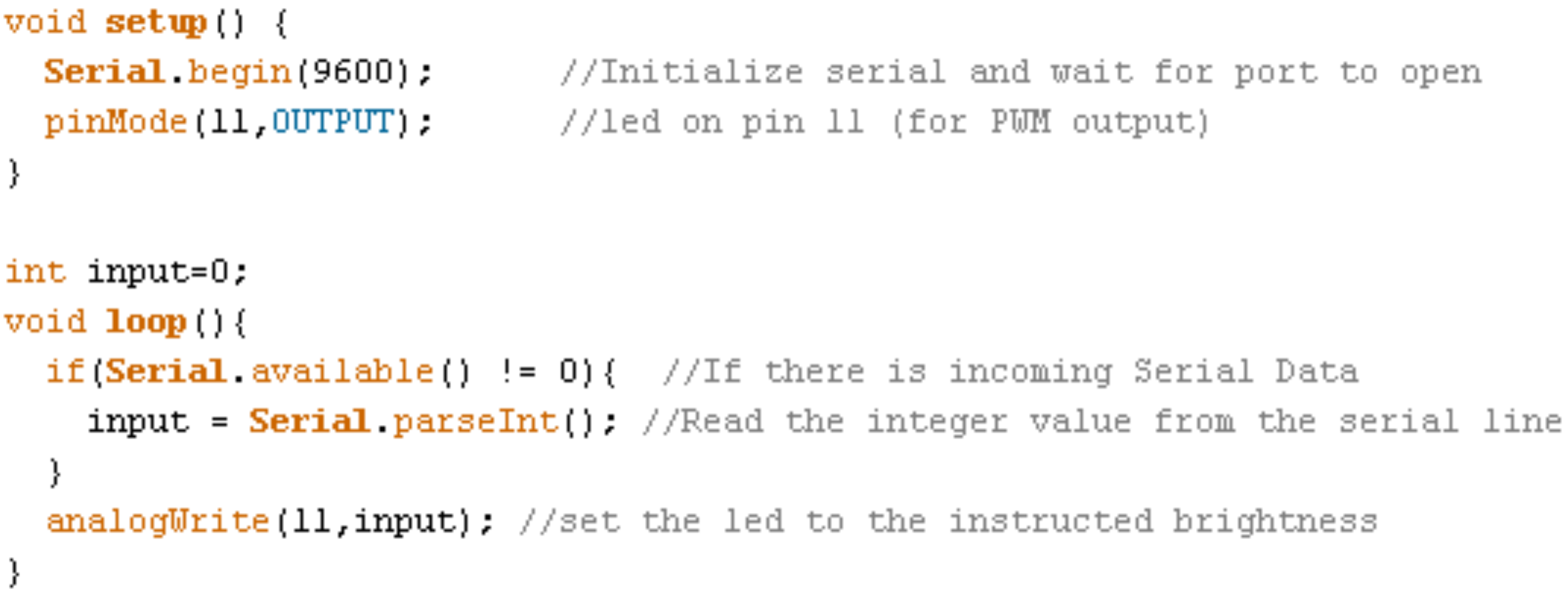
Character Parsing



```
void setup() {
  Serial.begin(9600);             //Initialize serial and wait for port to open
  pinMode(13,OUTPUT);
}

char incoming;                    //Create a temporary storage character (1 byte)
void loop(){
  if(Serial.available() != 0){  //If there is incoming Serial Data
    incoming = Serial.read(); //Read a char from the serial line
    if(incoming == 't'){  //if you send a 't' charecter
      digitalWrite(13,true); //turn the led on
    }
    if(incoming == 'f'){  //if you an 'f' charecter
      digitalWrite(13,false); //turn the led off
    }
  }
}
```

Integer Parsing



```
void setup() {
  Serial.begin(9600);      //Initialize serial and wait for port to open
  pinMode(11,OUTPUT);      //led on pin 11 (for PWM output)
}

int input=0;
void loop(){
  if(Serial.available() != 0){  //If there is incoming Serial Data
    input = Serial.parseInt(); //Read the integer value from the serial line
  }
  analogWrite(11,input); //set the led to the instructed brightness
}
```