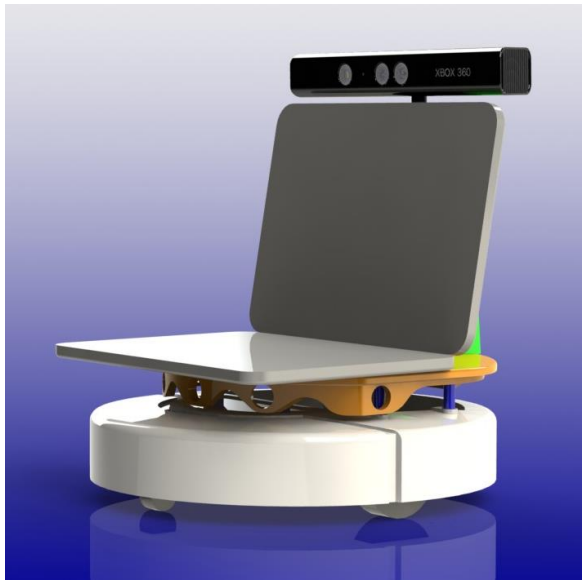
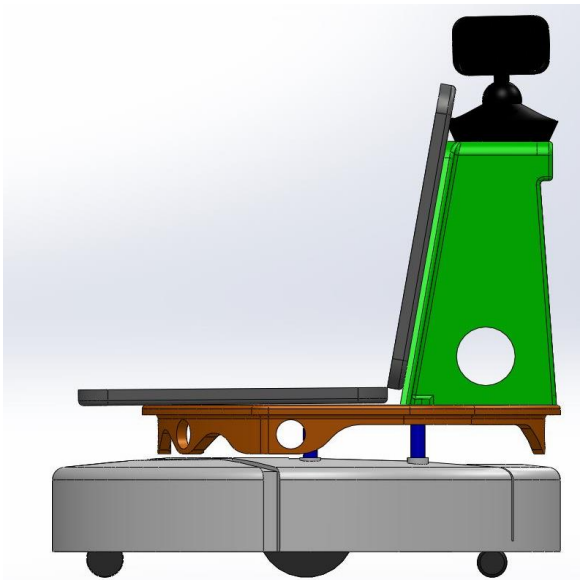
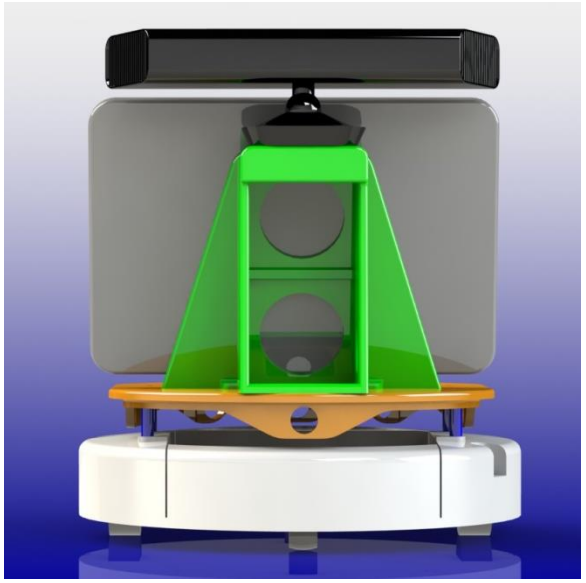
MAE 493G, CpE 493M, Mobile Robotics

3. Robot Communication

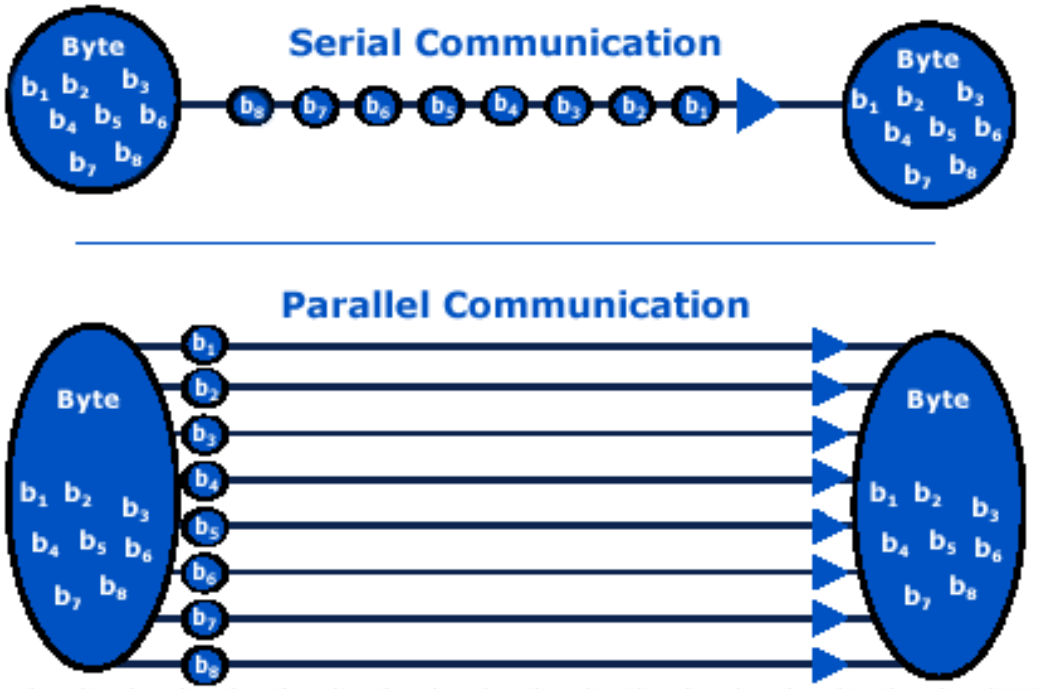






Parallel and Serial Communication

- Parallel communication is a method of conveying multiple binary digits (bits) simultaneously;
- Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus.



Asynchronous and Synchronous

- Whenever an electronic device transmits digital data to another electronic device, there must be a certain rhythm established between the two devices, i.e., the receiving device must have some way of knowing, within the context of the fluctuating signal that it's receiving, where each unit of data begins and where it ends;
- Synchronous transmissions are synchronized by an external clock, while asynchronous transmissions are synchronized by special signals along the transmission medium;
- Asynchronous transmissions are usually simpler and cheaper, but slower than synchronous transmissions.

Serial Peripheral Interface (SPI)

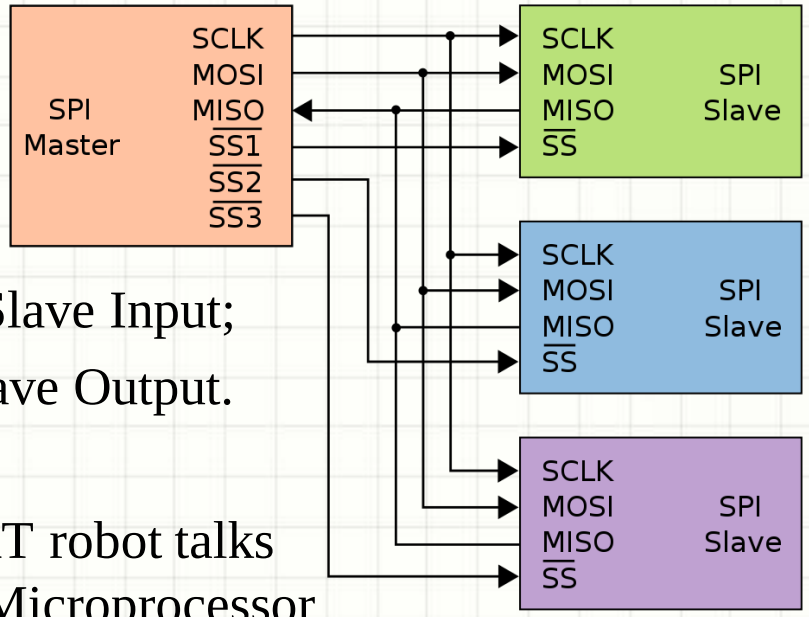
- The SPI bus is a synchronous serial data link, named by Motorola, that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines.

SCLK – Serial Clock;

SS – Slave Select

MOSI – Master Output, Slave Input;

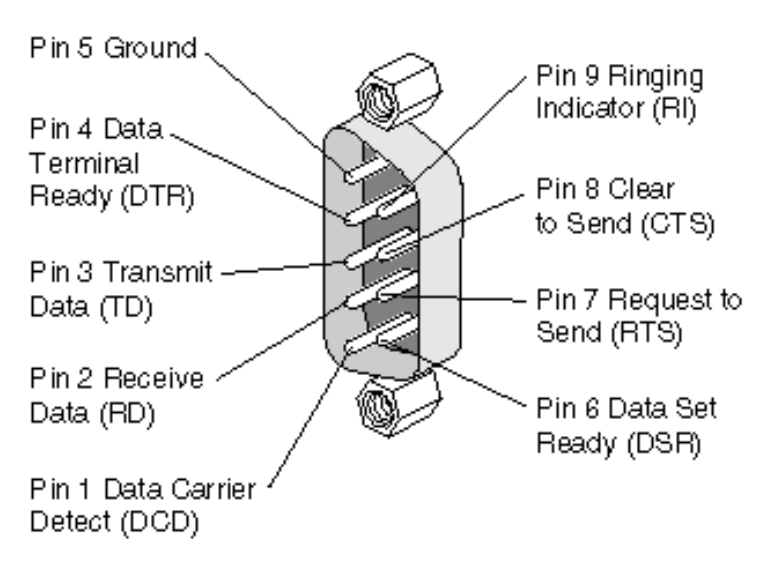
MISO – Master Input, Slave Output.



- The IMU Sensor on the SMART robot talks with the NetBurner Mod5213 Microprocessor with a SPI communication link.

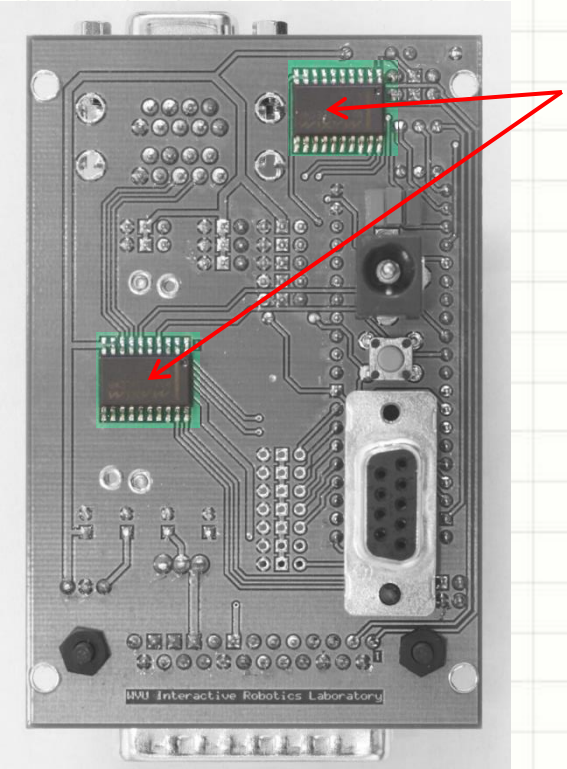
RS-232 Serial Port

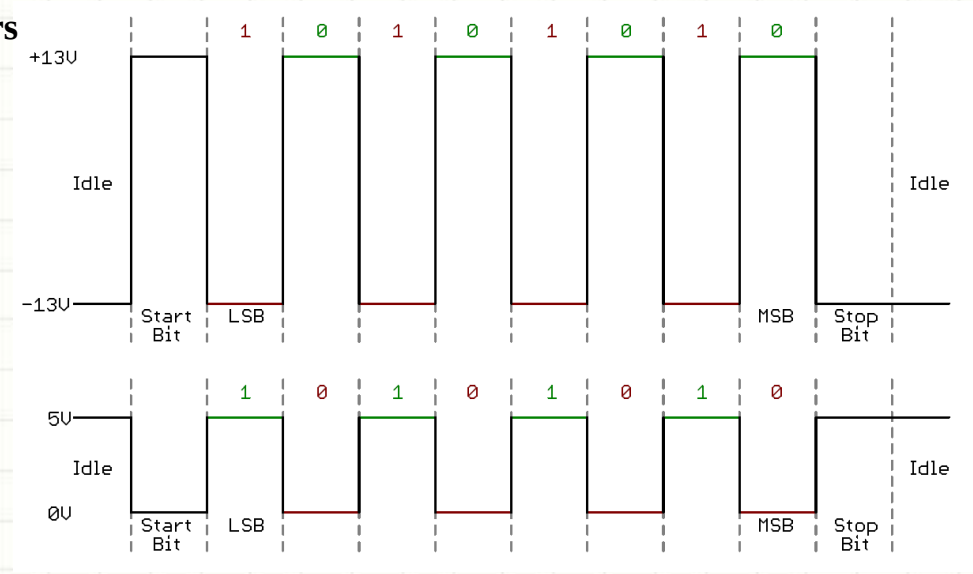
- RS-232 is the traditional name (first introduced in 1962!) for a series of synchronous serial communication standards between a DTE (Data Terminal Equipment) and a DCE (Data Communication Equipment).
- RS233 can no longer be found on your PC, but is still common for embedded microprocessors.
- A minimal "3-wire“ connection consisting only of transmit data, receive data, and ground, is commonly used.



RS232, TTL, and LVTTL

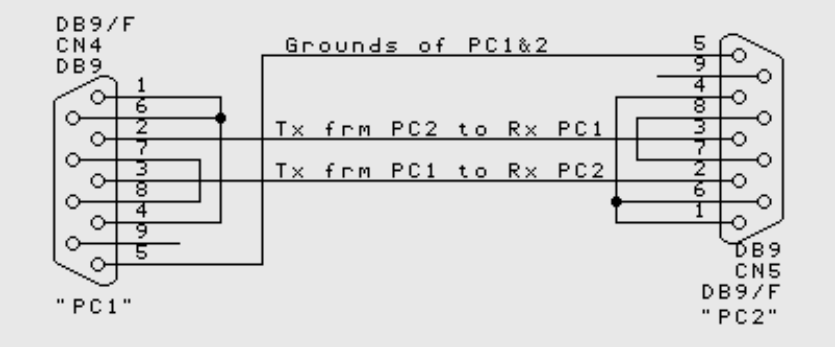
Type	Logic High	Logic Low
RS-232 (A Serial Communication Standard)	(-3.0)-(-25) V	3.0-25 V
TTL (Transistor-Transistor Logic)	2.0 - 5.0 V	0.0-0.8 V
LVTTL	2.0 - 3.3 V	0.0-0.8 V

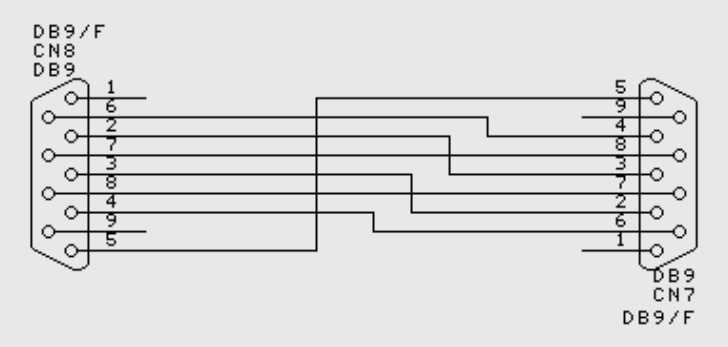




Null Modem and Loopback

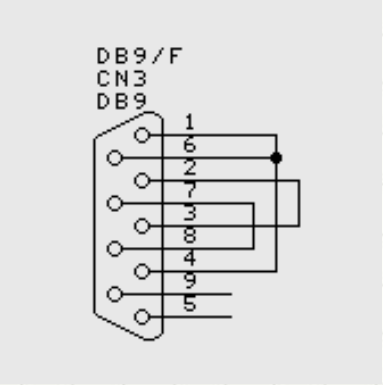
Use a Null Modem cable to connect between two computers.

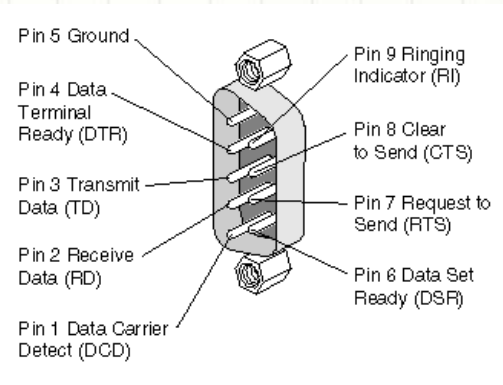




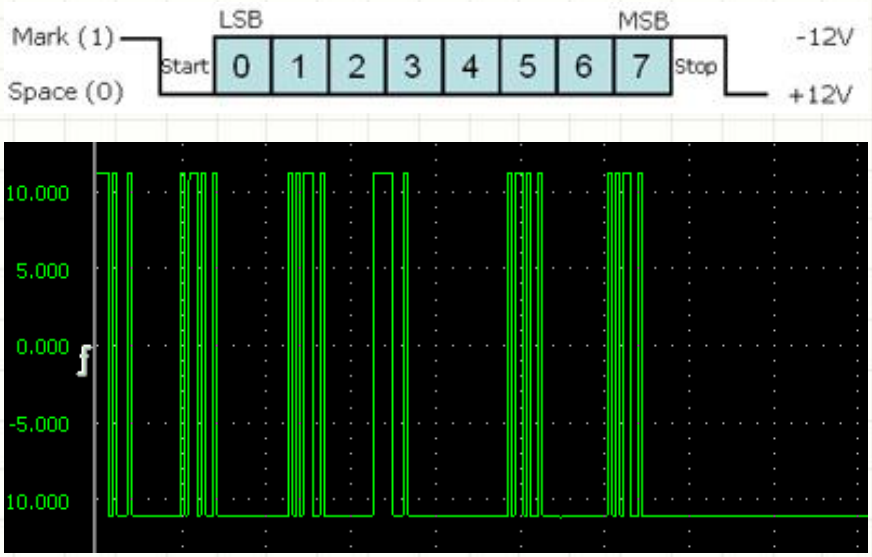
Without Handshaking **With Handshaking**

Loopback connectors can be used for debugging and testing.



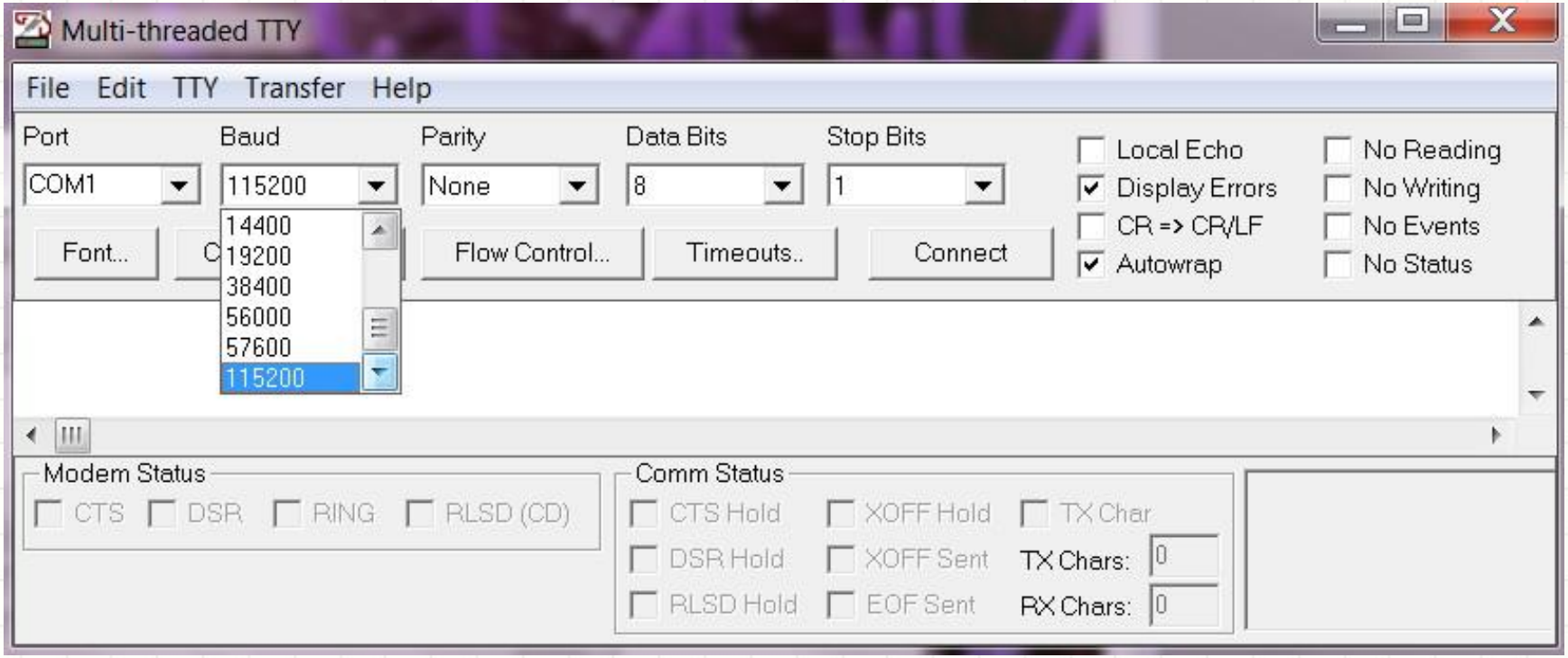


RS232 Signals



- A typical RS-232 logic waveform has 1 Start bit, 8 Data bits, No Parity and 1 Stop bit. The data transmission starts with a Start bit, followed by the data bits (LSB sent first and MSB sent last), and ends with a "Stop" bit;
- RS-232 connects the ground of 2 different devices together, which is the so-called “unbalanced" connection. An unbalanced connection is more susceptible to noise, and has a distance limitation of 50 ft (which is around 15 meters).

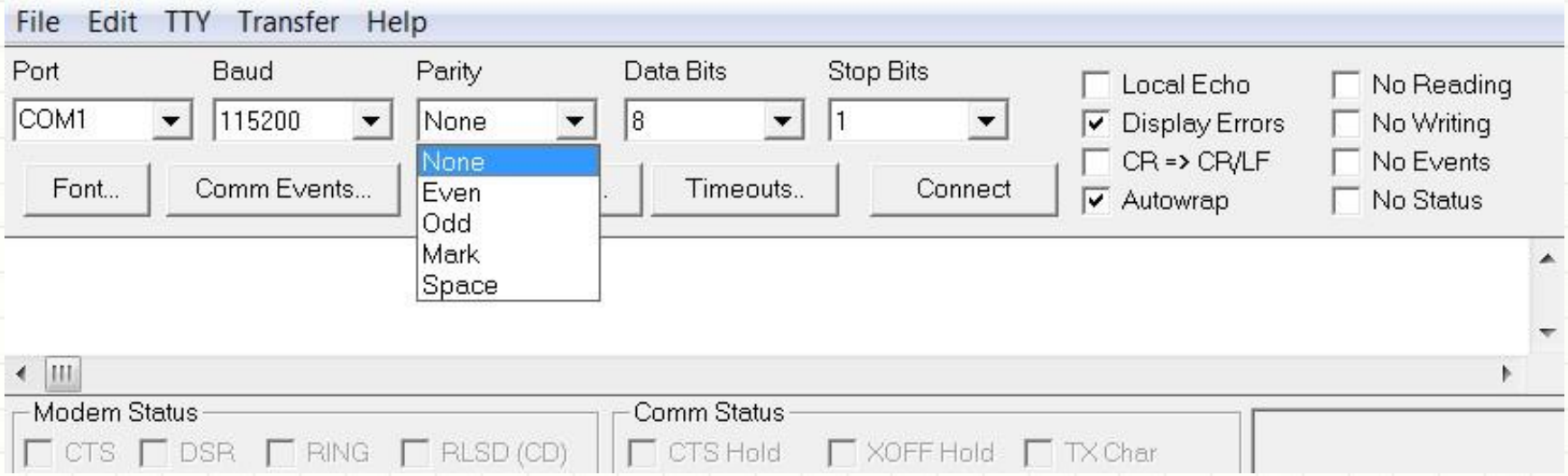
Serial Parameters



Port Number: pick which serial port to use.

Baud Rate: communication speed that measures the number of bit transfers per second. For example, 19200 baud is 19200 bits per second.

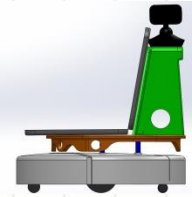
Serial Parameters (Cont.)



Parity: a method of detecting errors in transmission. For Even and Odd parity, the parity bit is set to a value to ensure that the data packet has an Even or Odd number of logic-high bits. Mark parity simply sets the parity bit to logic-high and Space sets the parity bit to logic-low.

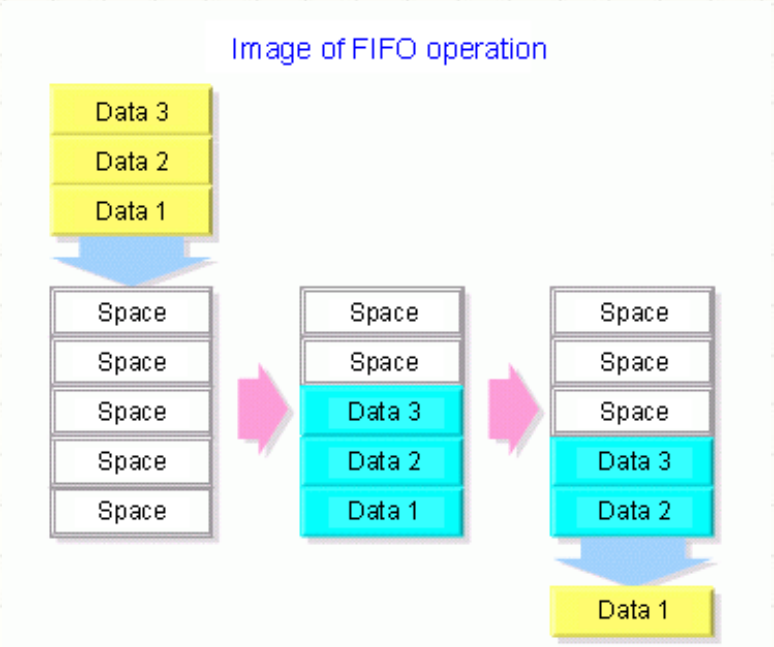
Data Bits: the number of data bits in a communication packet. Typically LSB is sent first, which is called "little endian."

Stop Bits: signal the end of a communication packet. This also helps to synchronize different clocks on the serial devices. Can be 1, 1.5, or 2.



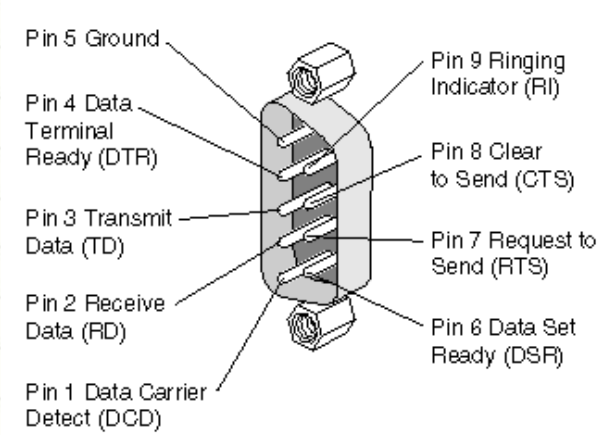
Serial Buffer

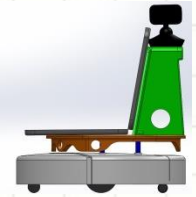
- A buffer is a region of memory used to temporarily store data while it is being moved from one place to another.
- The RS232 interface has at least two buffers each for sending and receiving data. First-In-First-Out (FIFO) buffers allow picking off individual bits for transmission.
- The operating system, such as Linux or Windows, sets aside part of memory for its own serial buffers.
- You might loose data if the allocated buffer is too small or have excessive delay if the buffer is too big.



Flow Control (Handshaking)

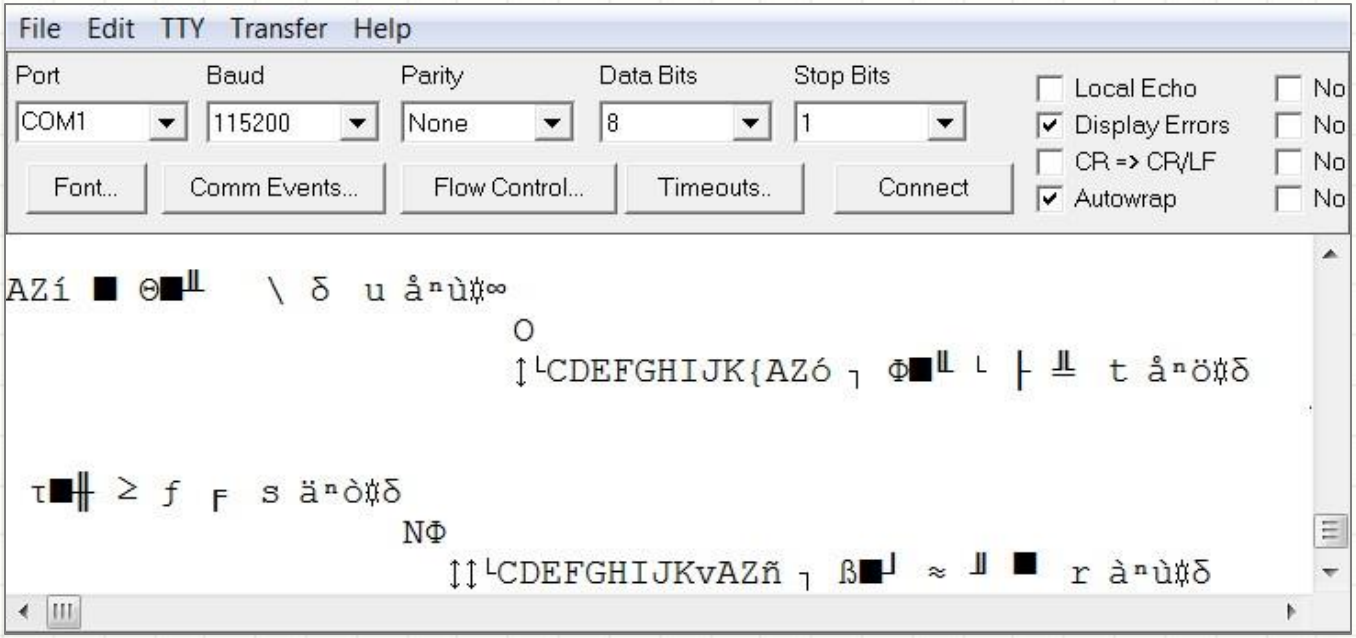
- Flow control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from overwhelming a slow receiver. By using Handshaking signals, receivers will be able to tell the sending device to pause data transmission if the receiver is overloaded.
- Hardware flow control often use the RS-232 RTS/CTS signal circuits. Generally, the RTS and CTS are turned off and on from alternate ends to control data flow, for instance when a buffer is almost full.
- In software flow control the XON/XOFF characters are sent by the receiver to the sender to control when the sender will send data. XON is decimal 17 and XOFF is decimal 19 in the ASCII chart. A drawback of software handshaking is that these two control characters can not be used in data.





Binary and ASCII

- Serial communication is always in binary (1s and 0s), but you can interpret it differently.
- Most time, if the message is in human language, we tend to use ASCII (American Standard Code for Information Interchange), which is a character encoding based on the English alphabet.
- If we want to send sensor data, it's more efficient to use binary bytes.



The Standard ASCII Table

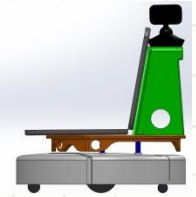
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of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

Source: www.LookupTables.com

The Extended ASCII Table

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡								
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±								
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥								
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤								
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠								
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡								
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷								
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈								
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°								
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙								
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·								
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√								
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	n								
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²								
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■								
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255									

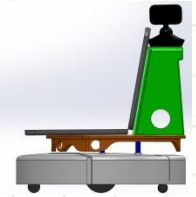
Source: www.LookupTables.com



Data Packet

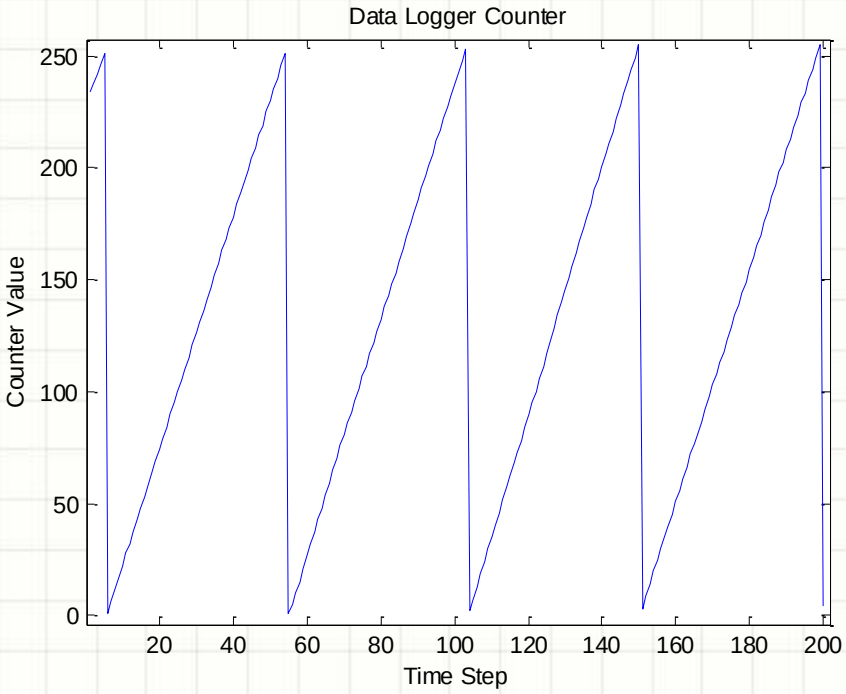
- Data to be transmitted are often grouped into a pre-negotiated format (communication packet).
- Click here for an example of the Novatel GPS (Page 21).
- For the SMART robot, we have a 50-byte communication packet between the sensor interface board and the laptop with the following configuration:

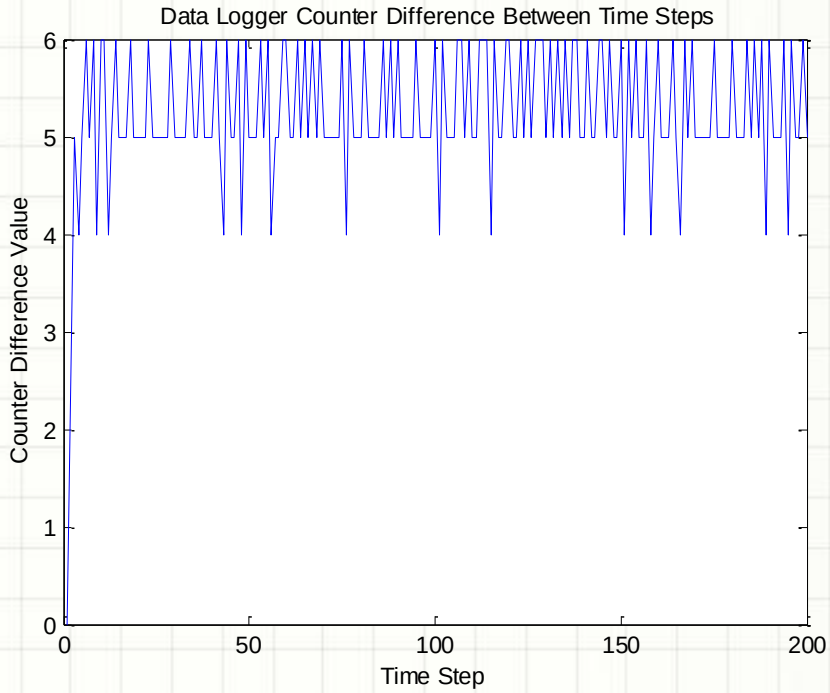
Byte	Content
0	Header Sync #1 (65_{10}, ASCII “A”)
1	Header Sync #1 (90_{10}, ASCII “Z”)
2	8-bit Counter (0-255)
3-48	Data bytes
49	Checksum



The Counter

The counter can help us to synchronize the data in time and can provide valuable information about the communication transmission quality.





50 Hz of Transmitted Data Received at ~10Hz at the SMART Laptop

Check Sum

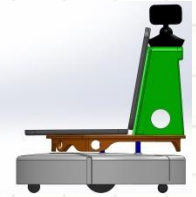
- Many serial protocols use checksum (additional bytes added at the end of the data packet) to check the data integrity, as errors might occur during data transmission;
- There are many types of checksum, from the simple Modula to sophisticated Cyclic Redundancy Checks (CRC) calculation;
- With the Modula method, the sender adds all command bytes together then mod it by 256 (decimal) to get an additional byte. This is to be added at the end of the command string. When the receiver receives the command string, it will first check the added byte to see whether data remain unchanged or not. If that is the case, it will accept the data, and if not, it will discard the data or ask the sender to resend the data;
- The SMART Robot uses a 1-byte Modula check sum.

Data Component (SMART)

Byte	Content
3, 4	IMU Acceleration x-axis
5, 6	IMU Acceleration y-axis
7, 8	IMU Acceleration z-axis
9, 10	IMU Angular Rate, p
11, 12	IMU Angular Rate, q
13, 14	IMU Angular Rate, r
15, 16	IMU Magnetic x-axis
17, 18	IMU Magnetic y-axis
19, 20	IMU Magnetic z-axis
21, 22	IMU Temperature
23, 24	Reserved A/D channel #1
25, 26	Reserved A/D channel #2

Byte	Content
27, 28	Front rangefinder
29, 30	Front Left rangefinder
31, 32	Left rangefinder
33, 34	Back rangefinder
35, 36	Right rangefinder
37, 38	Front Right rangefinder
39	LSB for switch #1, LS+1 for switch #2
40- 48	Reserved. Current values are 67-75 (ASCII ‘C’-’K’)

For a 16-bit data channel, the upper byte is transmitted first, then the lower byte:
Int16_Value = Upper_Byte *256 + Lower_Byte



Two's Complement

- Two's complement is the most common method of representing signed integers on computers.
- The two's complement of an N-bit number is defined as the complement with respect to 2^N; i.e., the result of subtracting the number from 2^N.
- An N-bit two's-complement numeral system can represent every integer in the range $-(2^{N-1})$ to $+(2^{N-1} - 1)$.
- For example, an 8-bit (1 byte) two's-complement number can range between -128 and +127:

Bits	Unsigned Integer	Two's Complement
0000 0000	0	0
0000 0001	1	1
0111 1110	126	126
0111 1111	127	127
1000 0000	128	-128
1000 0001	129	-127
1111 1111	255	-1

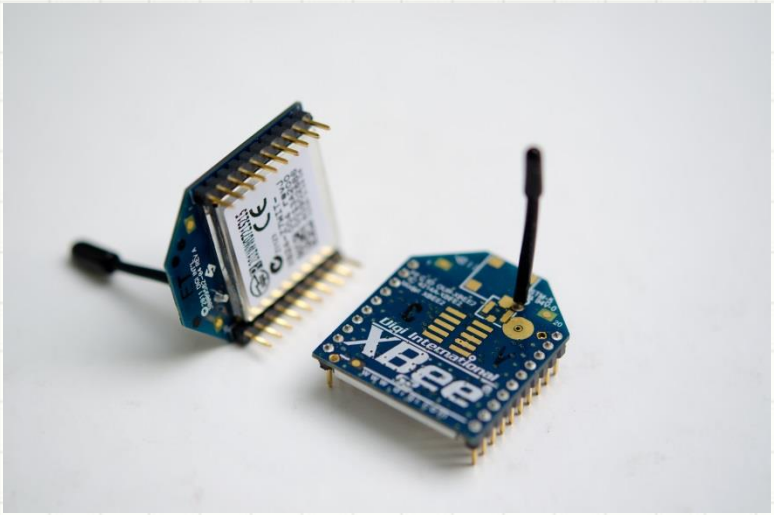
Polling Vs. Interrupt Driven

- The processor and data receiving devices are rarely synchronized;
- Polling means actively sampling the status of an external device;
- Polling is useful when something needs constant and urgent attention;
- Interrupt is an event external to the currently executing process that causes a change in the normal flow of instruction execution;
- Interrupts allows multiple devices or processes to co-exist;
- Interrupts are often ranked by priorities (backing up data on a laptop with a dying battery is often more important than receiving new keyboard inputs);
- Keep picking up a phone to check if someone called (Polling) Vs. work on something else and only pick up the phone when it rings (Interrupt Driven).

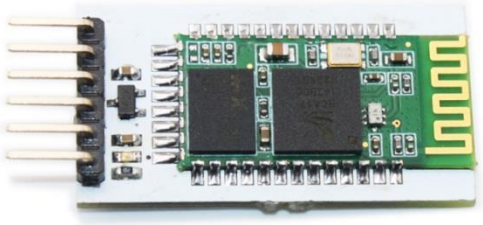
Wireless Communication

Many wireless communication devices have serial interfaces.

RF Modem



XBee



Blue-Tooth Modem

MATLAB Functions

```
>> S1= serial('com1') % Initialize the Serial Port
>> set(S1, 'BaudRate', 115200);  % Set the Baud Rate
>> fopen(S1);  % Open the Serial Port
>> fwrite(S1, [0, 12, 117, 251]); % Writing Binary Data
>> fprintf(S1, ‘Hello Word!') ;  % Writing ASCII
>> fread(S1,N); % Reading Binary Data
>> sentence = fscanf(S1, '%s'); % Reading ASCII
```

See here for an excellent explanation and sample code!

Summary

- Serial communication is the most common way of talking with sensors or other computers;
- RS232 is an serial communication standard that has been used for many years;
- A well designed communication packet can improve efficiency and reduce errors;
- Serial programing in MATLAB is quick and easy, as we will learn during our first lab session!

Further Reading

- Search Serial Communication, RS-232, Checksum, and SPI on Wikipedia;
- A tutorial on serial communication using MATLAB;
- Understand RS232.