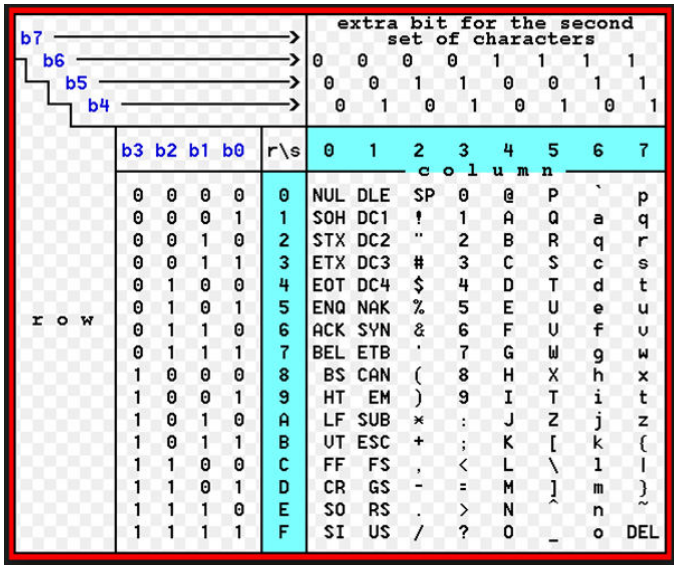
Chapter – 5: UART Based Serial Data Communication

1 Characters of the English Language are:

Upper-case Alphabets:	A, B, C, D, E, F, G, H, I, J, K, L, M, N , O, P, Q, R, S, T, U, V, W, X, Y, Z.		
Lower-case Alphabets:	a, b, c, d, e, f, g, h, I, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.		
Punctuation Marks:	! (Note of Exclamation)	" (Double Quote)	# (Numeric Sign)
	$ (Dollar Sign)	% (Percent Sign)	& (Ampersand Sign)
	' (Opening Single Quote)	((Opening Parenthesis)	) (Closing Parenthesis)
	* (Asterisk)	+ (Plus Sign)	, (Comma)
	- (Minus Sign)	. (Period)	/ (Forward Slash)
	: (Colon)	; (Semicolon)	< (Less Than)
	= (Equal)	> (Greater Than)	? (Question Mark)
	@ (At The Rate Of)	[(Opening Bracket)	\ (Backward Slash)
	] (Closing Slash)	^ (Exponentiation Sign)	_ (Underscore)
	' (Closing Single Quote)	{ (Opening Brace)	| (Pipe)
	} (Closing Brace)	~ (Tide)	
Numeral Characters:	0, 1, 2, 3, 4, 5, 6, 7, 8, 9.		
Control Characters:	NULL	SOH (Start of Heading)	STX (Start of Text)
	ETX (End of Text)	EOT (End of Trans.)	ENQ (Enquiry)
	ACK (Acknowledgement)	BEL (Bell Sound)	BS (back Space)
	HT (Horizontal Tab)	LF (Line feed)	VT (Vertical Tab)
	FF (Form Feed)	CR (Carriage Return)	SO (Shift Out)
	SI (Shift In)	DLE (Data Link Escape)	DC1 (Device Control 1)
	DC2 (Device Control 2)	DC3 (Device Control 3)	DC4 (Device Control 4)
	NAK (Negative Acknowledge)	SYN (Synchronous Idle)	ETB (End Trans. Block)
	CAN (Cancel)	EM (End of Medium)	SUB (Substitute)
	ESC (Escape)	FS (Form Separator)	GS (Group Separator)
	RS (Record Separator)	US (Unit Separator)	SP (Space)
	DEL (Delete)		

2 **(a)** What is the full name for ASCII?
(b) What is the purpose of assigning the so called ASCII Code to each of the characters of the English Language?

(a) ASCII stands for 'American Standard Code for Information Exchange'.
(b) Look at the character **A;** it is just an image or a sketch. How can we send it from one place to another (forget the Camera) using communication line? The solution is to assign an 8-bit value/ASCII Code (0100 1001) to the character **A**, and then send this code to other receiving end. At the receiving station, the display code will be placed back to the input of an of an ASCII type display; as a result, the image **A** will appear on the display unit. The ASCII Chart for the characters of English Language:



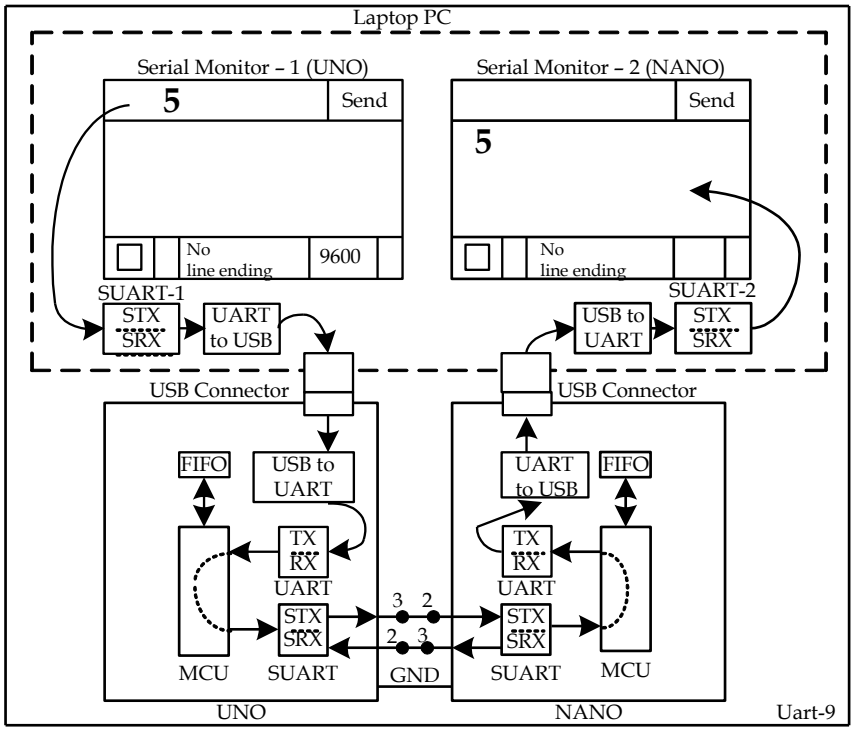
b7	b6	b5	b4	b3	b2	b1	b0	r\s	0	1	2	3	4	5	6	7
extra bit for the second set of characters																
0	0	0	0	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	0	0	1	0	2	STX	DC2	"	2	B	R	q	r
0	0	1	1	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	1	0	0	1	9	HT	EM	)	9	I	T	i	t
1	0	1	0	1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	1	1	1	1	F	SI	US	/	?	O	_	o	DEL

row — column

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

3 Draw Block Diagram to show how the character **5** starts its journey from the InputBox of Serial Monitor-UNO and then appears back on Serial Monitor-NANO.

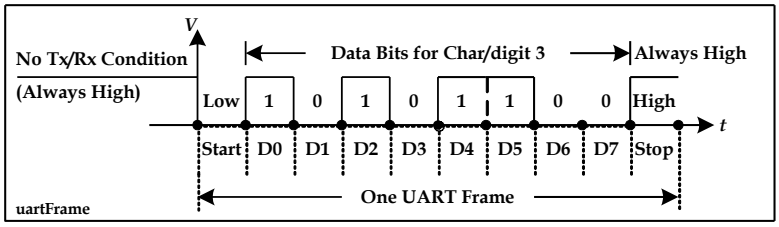


(1) After entering the character/digit/image 5 into the InputBox of Serial Monitor-1 (SM-1) from the Keyboard of PC, we click on the Send Button of the IDE.

(2) In response, the bit pattern (00110101 = 0x35) called ASCII Code is transmitted downward towards UNO (LSBit is transmitted first) by the 'Software UART Port-1 (SUART-1)'. SUART-1 is a software based UART Port which is usually found in the Laptop Computer; the Desktop Computer contains both Hardware Based UART Port (simply, UART) and SUART Port.

Before transmission, the SUART adds 'One START BIT (always Low)' before the 'ASCII Code of the character/digit' and 'One STOP Bit (always High)' after the 'ASCII Code of the character/digit'. The resultant 10-bit data is known as UART Frame which is shown below.

In the UART Frame, Logic-H is represented by 5V and Logic-L is represented by 0V, and hence it is known as TTLUARTFrame. This signal can travel up to 50 feet without distortion. If we want that the frame should travel up to 200 feet without distortion, we need to change its logic to RS232 Logic where Logic-H is represented by a voltage within -3.2V to -12V and Logic-L is represented by a voltage within 3.2V to 12V. The logic conversion is done with the help of MAX232 IC. The resultant signal is known as RS232UARTFrame.



(3) Let us (for this time) forget the inter-conversion processes of UART-to-USB within PC and then USB-to-UART within UNO.

(4) The UART Frame arrives to the RX-section of the UART Port of UNO where the START and STOP Bits are stripped out. The remaining 8-bit serial data takes the parallel form and activates a 'RX Ready' bit inside the UART Port. The RX Ready bit interrupts the MCU; the MCU goes to the side job (called Interrupt Sub Routine = ISR); the MCU reads the data byte from the RX-section of UART Port and saves it into a FIFO (first-in first-out) type data buffer. All these events happen automatically. We have the following Arduino instructions to deal with the FIFO buffer:

(a) To know the number of data bytes that the FIFO buffer has already accumulated, we can execute the following instruction:

```
byte n1 = Serial.available();    //n1 = 0 indicates that no data byte ha arrived via the UART Port
```

(b) To bring out the data byte (that has entered first into the FIFO) into the user variable *x1*, we may execute this instruction:

```
If (n1 != 0)
{
     byte x1 = Serial.read();   //data byte (the one that entered into the FIFO first)
}
```

(5) The UNO puts the data byte into STX-section of SUART Port for onward transmission to the NANO. The SUART adds (START Bit + ASCII Code + STOP Bit) to the frame. The UNO uses DPin-3 for transmission. SUART is software based UART Port which is created by executing the following instruction.

```
#include<SoftwareSerial.h>
SoftwareSerial SUART(2, 3);  //DPin-2 works as SRX-line; DPin-3 works as STX-line
SUART.begin(9600);           //bit transmission rate (Baud Rate = Bd) : 9600 bit/sec
SUART.write(x1);             //ASCII code is written into SUART Port
```

(6) The data byte reaches to SUART Port of NANO where the START/STOP bits are taken out and the 'SRX Ready' bit becomes active which interrupts the MCU of the NANO. The MCU goes to the ISR; the MCU reads data byte from SRX-section of the SUART Port and saves in the FIFO Buffer. The following codes carry out these steps:

```
byte n2 = SUART.available();
If (n2 != 0)
{
     byte x2 = SUART.read();
}
```

(7) The UNO puts the received data byte into TX-section of the UART Port for onward transmission of the UART Frame (START Bit + ASCII Code + STOP Bit) to SUART-2 Port of Laptop PC and then to SM-2 of NANO. The UART

Frame reaches to SUART-2 Port of the Laptop PC where the START/STOP bits are taken out. The 'SRX Ready' bit becomes active and interrupts the CPU. The CPU goes to the ISR; the CPU reads the data byte from SRX-section of SUART-2 Port and puts it to the Serial Monitor-2 of NANO. As a result, the character/digit/image 5 appears on SM-2. All these steps happen automatically except the first step which is brought into action by executing the following codes:

```
Serial.write(x2);
```

4 What is the difference between UART Port and SUART Port?

5 What is the ASCII Code for z?

6 What character will appear on S after the execution of the following instructions?

```
Serial.write(0x41) and Serial.write('A')?
```

7 Explain the working principle of **Serial.write();** instruction.

(1) The instruction will accept 8-bit ASCII code of a printable character; the code will be written to UART Port, which will ultimately appear to the input of SM; as a result, the corresponding character/digit/image will appear on the Serial Monitor. For example:

```
Serial.write(0x35);    //5 will appear on SM;
```

(2) The instruction will accept any printable character in this format: 'A'; the character will appear on SM. For example:

```
Serial.write('5');    //5 will appear on SM
```

(3) The instruction will accept any 8-bit binary code. If the code matches with the 8-bit ASCII code of a printable character, then that character will appear on SM; otherwise, the 8-bit value will be just transferred to the destination as a non-printable control byte or data byte.

```
Serial.write(0x35);    //5 will appear on SM
Serial.write (0x23);  //no character will appear on SM. It is a control byte  or data byte.
```

(4)

```
byte myData[3] = {0x31, 0x32, x033};
Serial.write(myData, sizeof(myData));//123 will appear on SM
```

9 **(a)** What is the full name of UART Port?

(b) What does it do?

Ans:

(a) UART stands for 'Universal Asynchronous Reception and Transmission' Port.

(b) The UART Port accepts 8-bit ASCII code of a character/digit in parallel format, converts it into serial format, sends 'One START Bit (always Low), send the ASCII code bit-by-bit with LSBit first, and then send 'One STOP Bit (always High).

10 Assume Bd (= baud rate = transmission rate = bit/sec) is 9600.

(a) Find 1-bit period.

(b) Find frame time.

(c) Find wastage of time in %.

11 Write code to put character 7 into UART Port.

```
UART.write(0x37);
```

However during programming, we do use the word **Serial** in place of **UART.** Now, the correct code is:

```
Serial.write(0x37);
```

12 What is the output of the following code?

```
Serial.print(1);
```

⇨ `Serial.print(12, DEC);`

The print() method will send code in such a way so that the characters/digits **12** (the decimal number 12) appear on SM. This comes this way:

* The ASCII code of 1 (0x31) is written to SM; as a result, 1 appears. After that, ASCII code of 2 (0x32) is written to SM; as a result, 2 appears on SM. Thus, the UART transmits 2 UARTFrame.

13 What will appear on SM after the execution of the following instruction:

Serial.print(12, HEX);

⇨ Serial.print(0x0C, HEX);

14 What character(s) will appear on SM/LCD after the execution of the following code?

Serial.write(0x37); //shows: 7

The write() method/function always sends the 8-bit ASCII code of the argument. If the bit pattern of the argument matches with an ASCII Code, then the corresponding character appears on the SM/LCD.

15 What character(s) will appear on SM/LCD after the execution of the following codes? (Remember that the print() method always shows the 'human friendly/understandable character(s)' on the SM/LCD as defined by the Base of the second argument.)

(1) Serial.print(37, DEC);
- **(a)** As the Base is 10, the characters 37 (a decimal number) will appear on SM/LCD.
- **(b)** To see 3 on SM, the PC must send 0x30 (the ASCII Code of 3) to UART Port using write() method.
- **(c)** To see 7 on SM, the PC must send 0x37 (the ASCII Code of 7) to UART Port using write() method.
- **(d)** Now, we see that the print() method is broken down into 2 write() methods to bring 37 on the SM/LCD.

(2) Serial.print(0x38, DEC);
SM will show 56 which is the decimal value of hex number: 56. There are two Serial.write() operations.

(3) Serial.print('A'); //A will appear on Serial Monitor.
There is no base as the base is only applicable if the 1st argument is an integer number; 'A' is a character.

(4) Serial.print('A', DEC); //will show 65
65 is the decimal value of the 8-bit (0x41) ASCII code of A.

(5) Serial.print("A"); //shows: A ; it is the 1st element of an 1-byte element array.

(6) Serial.print("A", DEC); //compilation error.

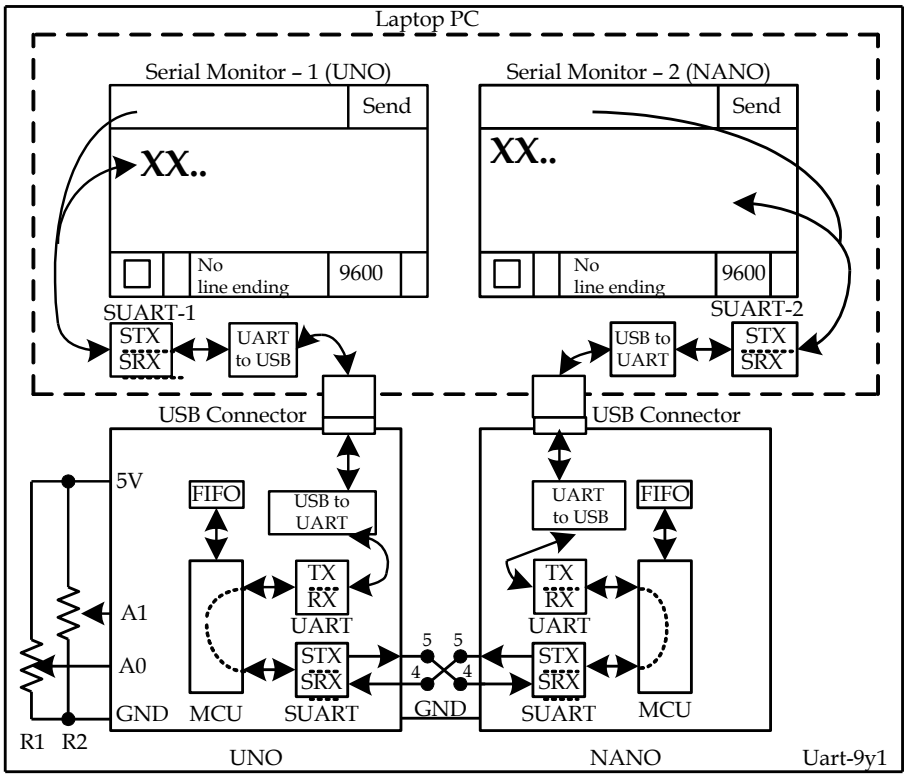
(7) Serial.print(temp, 10);
Given, unsigned int temp = 27;.
Ans: shows: 27 (the 2nd argument of the method is base 10 = DEC);

(8) Serial.print(temp, 2); //shows: 00011011 as the base is binary (2); 8 Serial.write() operations.; 8 Frames
Given, unsigned int temp = 27;

(9) Serial.print(temp, 2);
Given, float temp = 26.7895;.
Ans: Shows: 26.78. If the 1st argument is a floating point number, the 2nd argument indicates the number of digital to be printed after the decimal point (called precision).

(10) Serial.print("ABCD"); //shows: ABCD ; 4 write() operations; 4 ASCII frames

16 (1) Build circuit between UNO-NANO (or UNO) as per following diagram. In this circuit, UNO will acquire signal from R1 and R2 pots and then will display their values (in hex format) on Serial Monitor-1. The UNO will send these two values to NANO (or another UNO) using 'Software Serial Port (SUART Port)'; the NANO will receive the data and will show them on the Serial Monitor-2. If keep changing R1 and R2 slowly, you will see the changed values on both Serial Monitors simultaneously.



(2) Upload the following sketch in UNO

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(4, 5); //RX, TX, These pins will be used to send the data to another Arduino

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

void loop()
{
  unsigned int ValSensor0 = analogRead(A0); //R1 pot
  unsigned int ValSensor1 = analogRead(A1); // R2 pot

  //read figures in serial monitor
  Serial.print("handle Sensor 0: ");
  Serial.println(ValSensor0, DEC);
  Serial.print("handle Sensor 1: ");
  Serial.println(ValSensor1, DEC);
  //--send R1 value to NANO------
  mySerial.write(0x01); //R1 Pot Data is next one
  mySerial.print(ValSensor0, DEC);
  mySerial.write(0x02);  //end-of-message
  delay(1000);
}
```

(3) Upload the following sketch in NANO

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(4, 5); //RX, TX, These pins will be used to send the data to another Arduino
int i = 0;
char dataArray[50] = "";
bool flag1 = false;

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

void loop()
{
  byte n = mySerial.available();
  if (n != 0)
  {
    if (flag1 == false)
    {
      byte x = mySerial.read();
      if (x == 0x01)
      {
        Serial.print("Received Sensor 0: ");
        flag1 = true;
      }
    }
    else
    {
      byte x1 = mySerial.read();
      if (x1 != 0x02)
      {
        dataArray[i] = x1;
        i++;
      }
      else
      {
        dataArray[i] = 0x00;  //nul-byte
        Serial.println(dataArray);
        i = 0;
        flag1 = false;
```

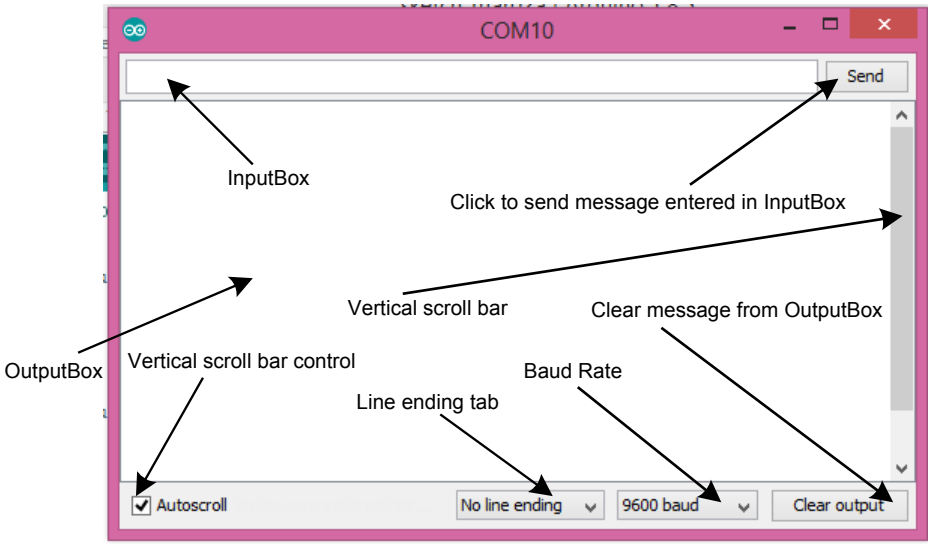
```
      }
    }
  }
}
```

(4) Check that Sensor 0 (for Pot R1) appears on Serial Monitor of NANO.
(5) Add codes with Sketch of UNO and NANO so that Sensor 1 (Pot R2) value appears on SM-1 and SM-2.

17 What does this command do: byte n = Serial.available();?
When a valid data item (8-bit ASCII code or binary code) arrives at the Receiver Register of the UNO's MCU, the data byte immediately enters into a FIFO type buffer. If we don't bring out the data byte from the FIFO into a user variable by executing a [b]Serial.read()[/b] instruction, it will remain there. Before we perform a Serial.read() operation on the FIFO, we naturally wish to check if the FIFO has actually accumulated any data item. The execution of the [b]byte n = Serial.available();[/b] instruction assigns an integer value to the variable [b]n[/b], which is equal to the number of 'data items' currently present in the FIFO. Therefore, the logical codes would like:

```
void loop()
{
   byte n = Serial.available();
   if (n != 0)       //FIFO has one data item provided we have read the previously arrived data item
   {
       byte x1 = Serial.read();  //there is no more data item in the FIFO
       Serial.println("5");          //send 5 and then non printable 'new line character'
   }
}
```

18 Draw a replica of the Serial Monitor Interface and then label its various fields
19 What happens when we click on the en Button of the Serial Monitor after enabling the following line ending options (one at a time) in the 'Line ending' Tab.



(1) **No line ending:** No data byte is transferred to the Arduino.
(2) **Newline:** The data byte 0A (00001010) is transmitted within a 10-bit UART Frame. As a result, the cursor moves to the home position and then goes one line below (can we call it Carriage Return and New Line?).
(3) **Carriage return:** The data byte 0D (00001101) is transmitted within a 10-bit UART Frame. As a result, the cursor neither moves to the home position nor goes one line below.
(4) **Both NL & CR:** The data byte 0D (00001101) is transmitted first within a 10-bit UART Frame and then the data byte 0A (00001010) is transmitted. As a result, the cursor moves to the home position and then goes one line below.

20 Given, byte x = 0x41;. What character (s) will appear on SM after the execution of the following instructions:
(1) Serial.write(x);
Ans: A will appear on SM. 01000001 will be transmitted to the ASCII-type Serial Monitor. 01000001 is the ASCII Code of the character A; so, A will appear on SM.

(2) Serial.print(x); ===> Serial.print(x, DEC);
Ans: 65 will appear on SM. x is given as a number (by explicit placement of the keyword byte); so, the compiler will associate a decimal base with it as the 2nd argument of the function. 0x41 will be converted into decimal number 65 (4x16 + 1 = 65). As a result, UNO will send two frames (0x36 and 0x35 = ASCII codes of 5 and 6) to SM and the SM will show 65.

(3) Serial.print((char)x);
Ans: A appears on Serial Monitor. The compiler prepares codes in such a way so that the UNO transmits only one UART Frame which is: Serial.write(0x41). The Compiler ignores the base (the 2nd argument of the function).

21 Given, char x = 'A';. What character (s) will appear on SM after the execution of the following instructions:

(1) Serial.print(x);
Ans: A will appear on SM. 01000001 will be transmitted to the ASCII-type Serial Monitor. Base is being ignored due to data type (char) attached with the variable x.

(2) Serial.print(x, DEC);
Ans: 65 will appear on SM. Though the data type of x is char, the user has forced to convert the value of x (0x41) from hex to decimal (65). As a result, two frames (Serial.write(0x36); and Serial.write(0x35)) will be transmitted SM; the SM will show 65.

22 Explain the meaning of the following instruction with example:

```
Serial.setTimeout(10000UL);    //10 sec unit in ms UL = unsigned long
long x = Serial.parseInt();         //-2 147 483 648 to +2 147 483 647 (signed 32-bit)
```

Let the incoming frames be formatted into valid data bytes and get entered into the FIFO buffer within the set time of 10 sec. Within this 10 sec time, the function keeps reading the FIFO. If no string is found to parse within this 10 sec, there is a timeout. The function output 0 results. The process repeats again. The function reads the data bytes from FIFO under the following rules.

(1) Beginning non-digits (other than 0 to 9) are skipped.

(2) Valid digits (+ - 0, …, 9) are taken until a non-digit or – (dash) occurs. There must be a non-digit for the function to terminate.

(3) Example:

```
void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(10000UL);  //10 sec unit ms unsigned long
}

void loop()
{
  long x = Serial.parseInt(); //real time reading from FIFO ; parse integer value (0000… to 9999…)
  Serial.print(x);
}
```

(a) Upload the above sketch.

(b) Open Serial Monitor with 'No line ending' option.

(c) Enter we234j in the InputBox of SM and then click on Send Button.

(d) Check that 234 has appeared on the SM.

23 What characters will appear on SM after the execution of the following instructions:

(1) Serial.print(1234); ⇒ Serial.print (1234, DEC); **Ans:** 1234

(2) Serial.print("1234");**Ans:** 1234 It is because the "1234" is coded as {'1', '2', '3', '4'}={0x31, 0x32, 0x33, 0x34}

24 Two integer values 25 and 37 would be sent to UNO from the SM by placing them in this format <25,37> in the InputBox and then clicking the Send button. Write program to receive the string (a series of characters), isolate the numbers from the string and show them on the SM.

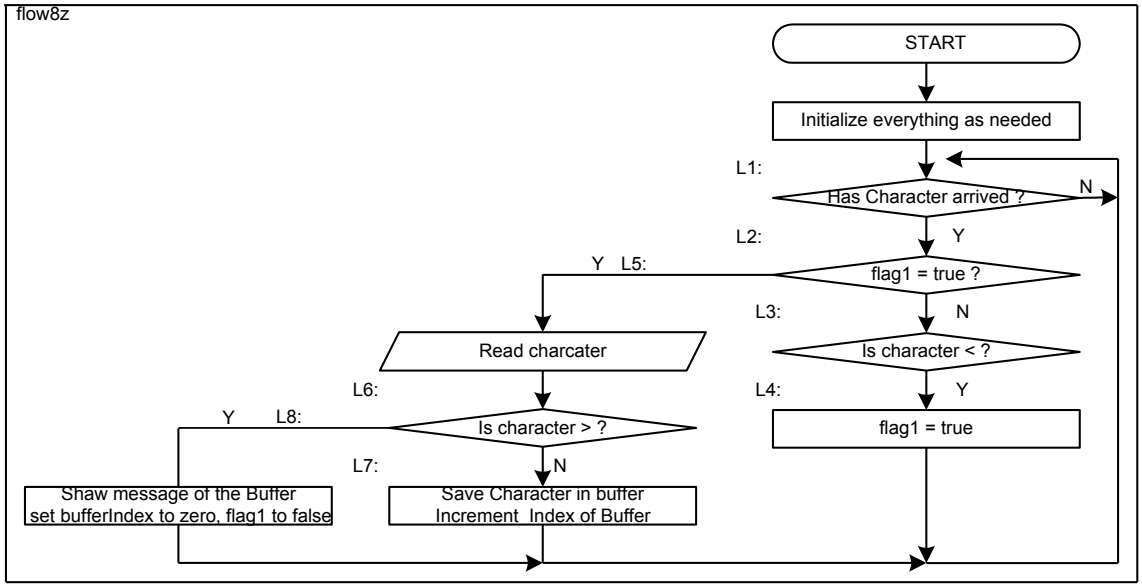
Solution Hints:

(1) Read all the characters excluding (>) from the FIFO buffer into a character type array named **char myData[50]** using this command: **Serial.readBytesUntil('>', myData, 50);**.

(2) Print content of myData[] array: SM shows: <25,37. (internally they are in ASCII Codes: 3C 32 35 2C 33 37.

(3) This function **int x = atoi(arrayContaingASCII);** converts ASCII codes into integer (limited within 0, 1, …, 9). In order to use this function, let us put 0x3F (ASCII code of 0) at location myData[0] to replace < and 0x00 (ASCII Code of NULL-byte) at location myData[3] to replace ,. Now, execute the atoi() (ASCII To Integer) function and get 25 into variable x. Print the value of x and get 25 on the SM.

25 Convert the following Flow Chart into programming codes.




```
char dataBuffer[50] = "";
int bufferIndex = 0;
bool flag1 = false;
byte x;

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

void loop()
{
  if (Serial.available() > 0) //L1:
  {
    if (flag1 != true)    //L2:
    {
      x = Serial.read();  //L3:
      if (x == '<') //Beginning Marker
      {
        flag1 = true;          //L4:
      }
    }
    else     //L5:
    {
      x = Serial.read();
      if ( x != '>')     //L6: Ending Marker
      {
        dataBuffer[bufferIndex] = x;  //L7:
        bufferIndex++;
      }
      else
      {
        Serial.println(dataBuffer);  //L8:
        bufferIndex = 0;
        flag1 = false;
      }
    }
  }
}
```

(1) In the InputBox of the 9600 Bd Serial Monitor, we enter <12345> and then press the Send button. After that the following events occur:

(2) The PC sends the ASCII Code of < which is (00111100 = 0x3C) in a 10-bit frame called UART Frame where LSBit is transferred first. The frame transmission time from PC to UNO is about : 10x1/9600 = 1041 us. The PC will send 7 such frames one after another at 1041 us apart.

(3) The UNO receives a frame and finishes the processing far less that 1041 us. The 8-bit character enters into the Receiver Section of the UART Module of the MCU and interrupts the processor. The processor goes to the ISR (Interrupt Sub Routine) and saves the data (character) byte into a 64-byte wide FIFO Buffer.

(Assume that we not reading data/character from the FIFO Buffer)

(4) The next character 1 enters into the next location of the FIFO Buffer and so on.

(5) The Arduino Platform has given us (that's why, the Arduino is so popular to the Logicians) us a function called byte x = Serial.available(); by which we can check how many charcaters so far been accumulated in the buffer.

(6) Arduino has also given us another function called [b]byte y = Serial.read();[/b] by which we can transfer a data/character from the FIFO buffer (the character that has entered first in the FIFO will come out first) into the variable x for processing.

(7) It is hoped that the above points are enough to clarify the meanings of the following commands:

```
if(Serial.available()>0)
{
   //do as needed
}
//--------------------
if(Serial.available() == 3)
{
   //do as needed
}
```

26 Write meanings and correct declarations for the following UART related functions/methods:

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