TeachEngineering

Ignite STEM learning in K-12

Oh Baby! Contractions and Calculations / Activity Part 3

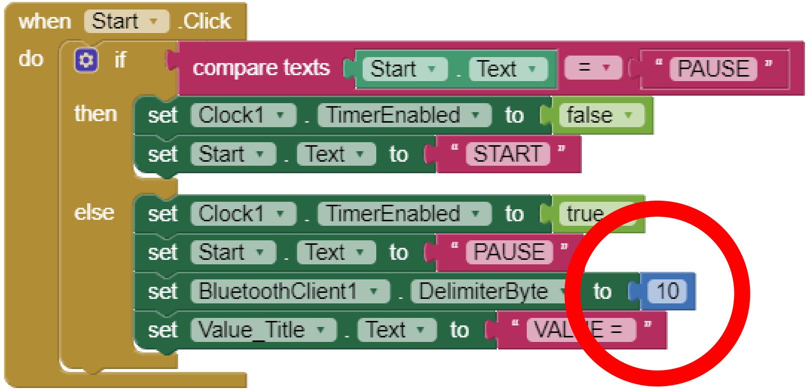
The Analog-to-Digital Converter

The Arduino has a built-in analog to digital converter capable of storing a 10-bit number. Therefore, the Arduino can recognize 2^{10} = 1024 different values ranging from 0–1023.

<table>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td colspan="10">Digital Value 735</td></tr>
</table>

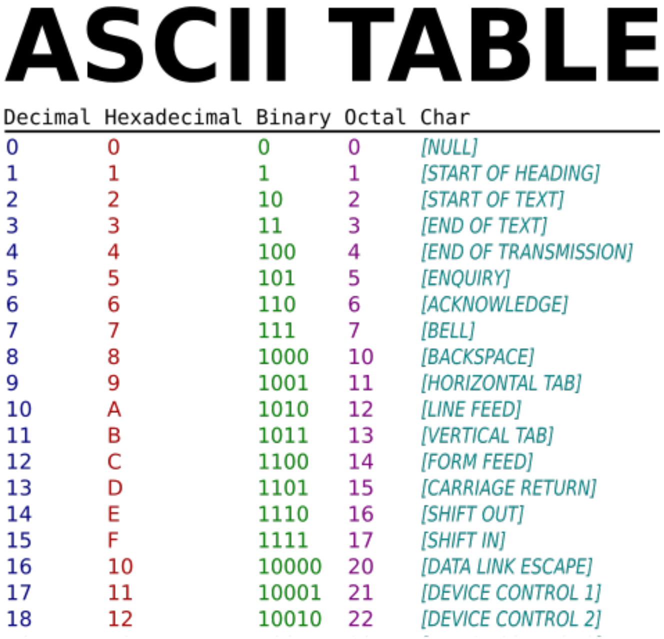
The Delimeter Character

In this activity, we are using serial communication to transmit data (FSR values) over Bluetooth. This means the Arduino sends each character as an 8-bit byte using the ASCII character value encoded as a decimal. The receiving end, the Android device, defaults to displaying byte values using the assigned ASCII value. Each force value (message) is separated by the decimal-encoded ASCII character for a line feed.



```
when Start .Click
do  if  compare texts  Start . Text  =  " PAUSE "
    then  set Clock1 . TimerEnabled to  false
          set Start . Text to  " START "
    else  set Clock1 . TimerEnabled to  true
          set Start . Text to  " PAUSE "
          set BluetoothClient1 . DelimiterByte to  10
          set Value_Title . Text to  " VALUE = "
```

ASCII Line Feed



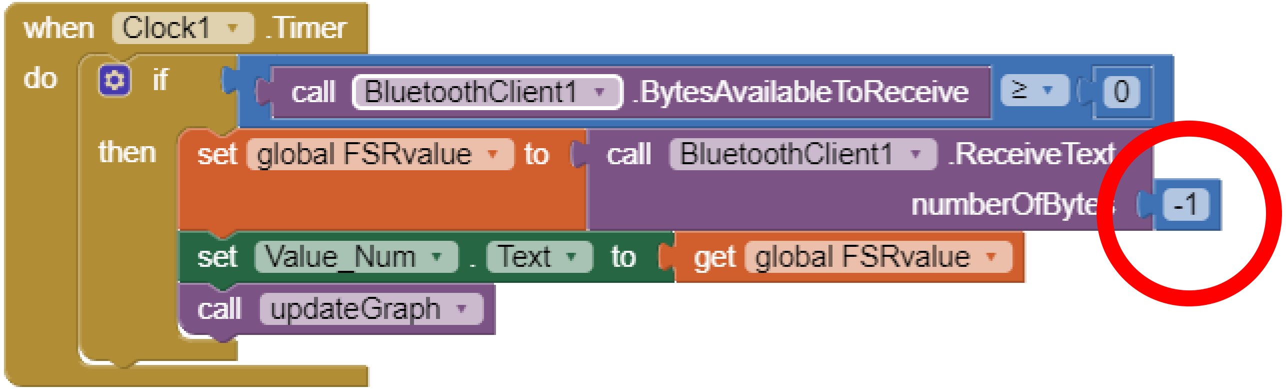
ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]

ASCII table

Buffers and Bytes

Setting the number of bytes to receive = -1 (or any negative number), signifies that there is no maximum number of bytes to the serial message.

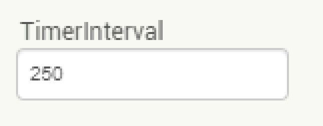


```
when Clock1 .Timer
do  if    call BluetoothClient1 .BytesAvailableToReceive ≥ 0
    then  set global FSRvalue to call BluetoothClient1 .ReceiveText
                                                  numberOfBytes -1
          set Value_Num . Text to get global FSRvalue
          call updateGraph
```

Delimeters, Buffers, and Timing

You will need a delay (1000 milliseconds) in the Arduino code and a clock timer (250 milliseconds) in the MIT App Inventor code.

You want the app timer to be faster than the Arduino delay. If it were the other way around, then there would be a lag on the app end. The app would not be able to process the information because data would arrive faster than it can process it. But if the app is faster than the Arduino delay, then there will never be any bytes in the buffer.

App Clock Interval (ms)	<	Arduino Delay (ms)
 TimerInterval 250	<	`delay(1000);`

A Serial Packet or Frame

Each character in a message has start and stop bits to indicate the beginning and end of the byte along with a parity bit (outside today’s scope) that basically checks for even/odd errors. We are concerned with the 8-bit byte of message.

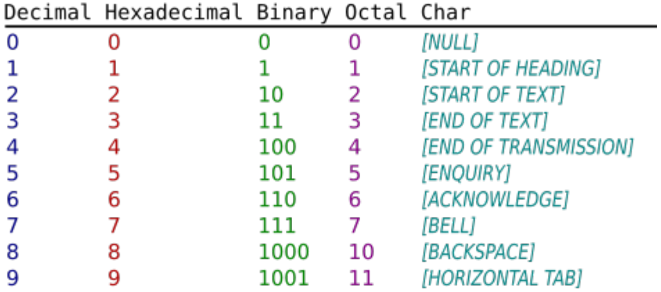
Start Bit	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Parity Bit	Stop Bit

Let's say that we had two sensor values in a row, separated by the line feed character. Suppose the sensor values were ADC 57 followed by ADC 864. The way the serial communication works is to send the character 7 encoded for ASCII as a decimal, followed by the 5, followed by the line feed character code, followed by the 4, then 6 then 8, then finally the line feed.

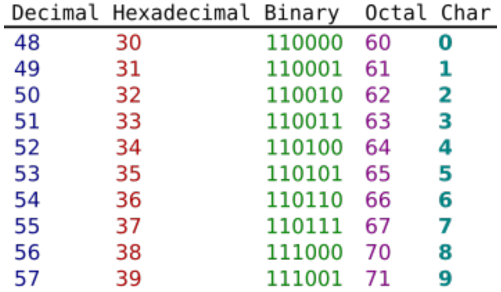
Two Readings and Two Delimeters

Let’s look at the ASCII table and discuss the values in the table below.

ASCII Characters		
Character	Decimal	Binary
7	55	0011 0111
5	53	0011 0101
LF	10	0000 1010
4	52	0011 0100
6	54	0011 0110
8	56	0011 1000
LF	10	0000 1010

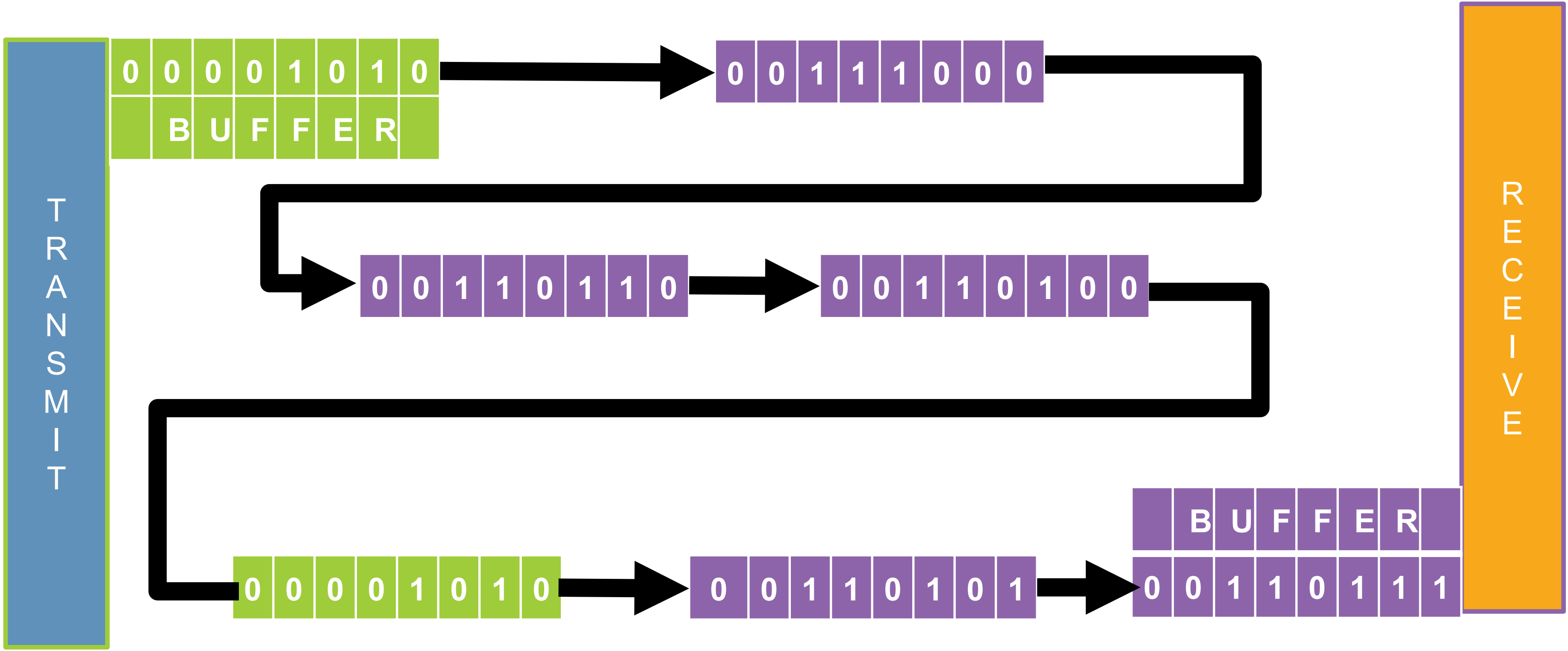


Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*



Decimal	Hexadecimal	Binary	Octal	Char
48	30	110000	60	**0**
49	31	110001	61	**1**
50	32	110010	62	**2**
51	33	110011	63	**3**
52	34	110100	64	**4**
53	35	110101	65	**5**
54	36	110110	66	**6**
55	37	110111	67	**7**
56	38	111000	70	**8**
57	39	111001	71	**9**

The Serial Flow



TRANSMIT

0 0 0 0 1 0 1 0

BUFFER

0 0 1 1 1 0 0 0

0 0 1 1 0 1 1 0

0 0 1 1 0 1 0 0

0 0 0 0 1 0 1 0

0 0 1 1 0 1 0 1

BUFFER

0 0 1 1 0 1 1 1

RECEIVE