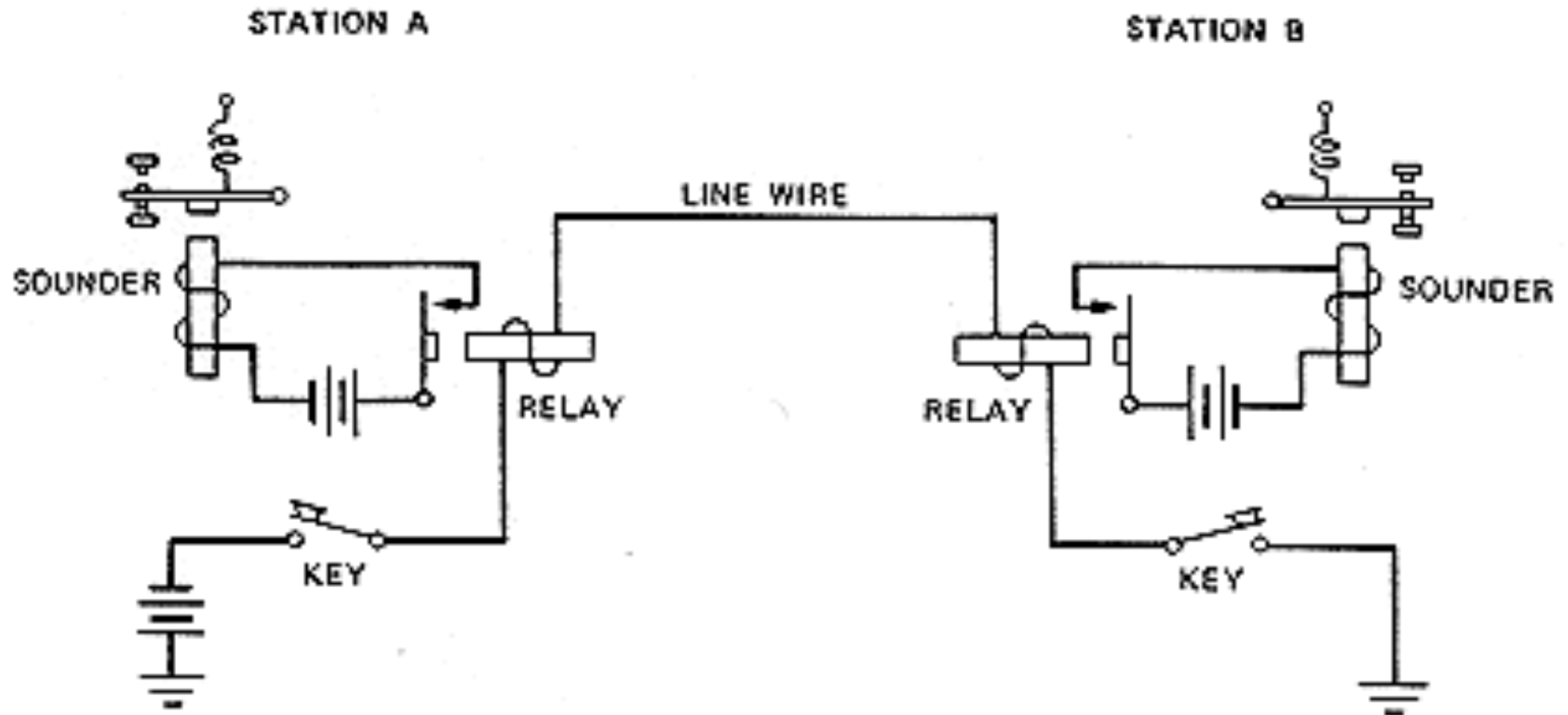


Communication

The Serial Protocol and ASCII Character Codes

SIMPLEX TELEGRAPH



Elementary neutral telegraph circuit.

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

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 — — ● ●

1 ● — — — —
2 ● ● — — —
3 ● ● ● — —
4 ● ● ● ● —
5 ● ● ● ● ●
6 — ● ● ● ●
7 — — ● ● ●
8 — — — ● ●
9 — — — — ●
0 — — — — —

blink.c -> sos.c

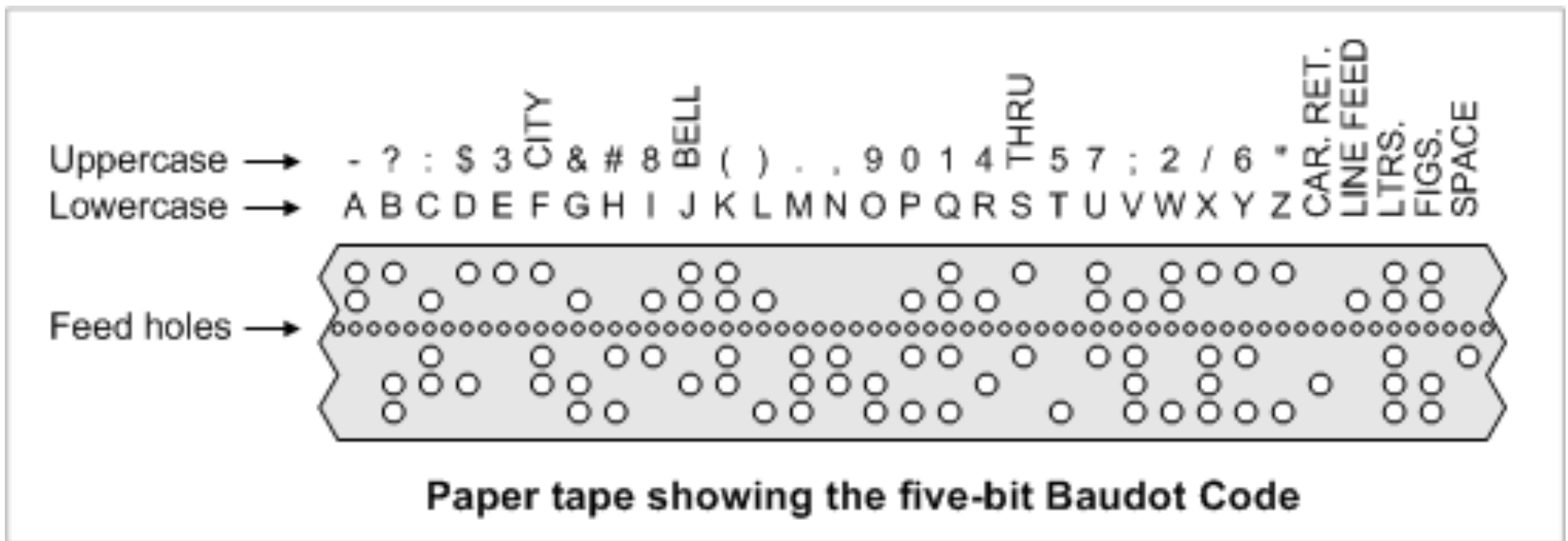
Teletype



http://www.smecc.org/police_-_fire_-_civil_defense_communications.htm

Baudot Code

<https://savzen.wordpress.com/tag/ baudot/>



Baud: Number of symbols per second

e.g. 9600 baud = 9600 bits/sec

% ascii

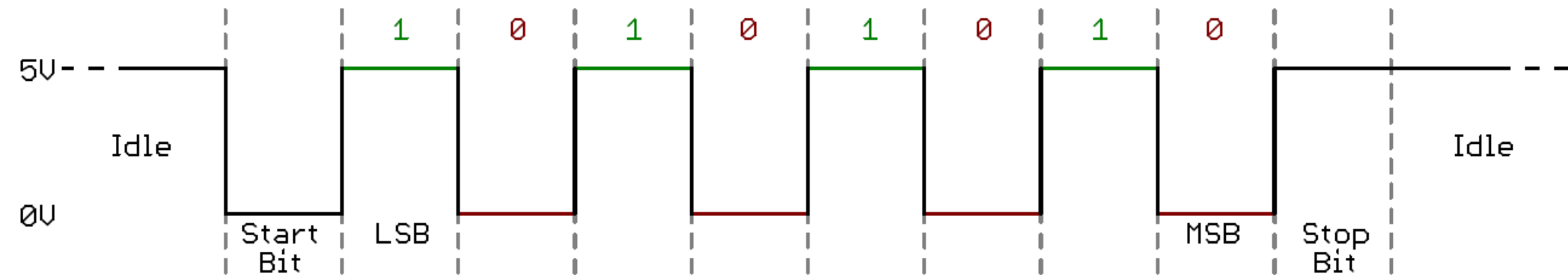
2 3 4 5 6 7

0: 0 @ P ' p
1: ! 1 A Q a q
2: " 2 B R b r
3: # 3 C S c s
4: \$ 4 D T d t
5: % 5 E U e u
6: & 6 F V f v
7: ' 7 G W g w
8: (8 H X h x
9:) 9 I Y i y
A: * : J Z j z
B: + ; K [k {
C: , < L \ l |
D: - = M] m }
E: . > N ^ n ~
F: / ? O _ o DEL

"cs107e" =

\0
64
37
30
31
73
63

Asynchronous Serial Communication



1 start bit (0), 8 data bits (lsb-first), 1 stop bit (1)

9600 baud = 9600 bits/sec

(1000000 usecs)/9600 ~ 104 usec/bit

<https://learn.sparkfun.com/tutorials/serial-communication>

sos.c -> serial.c

Logic Analyzer!



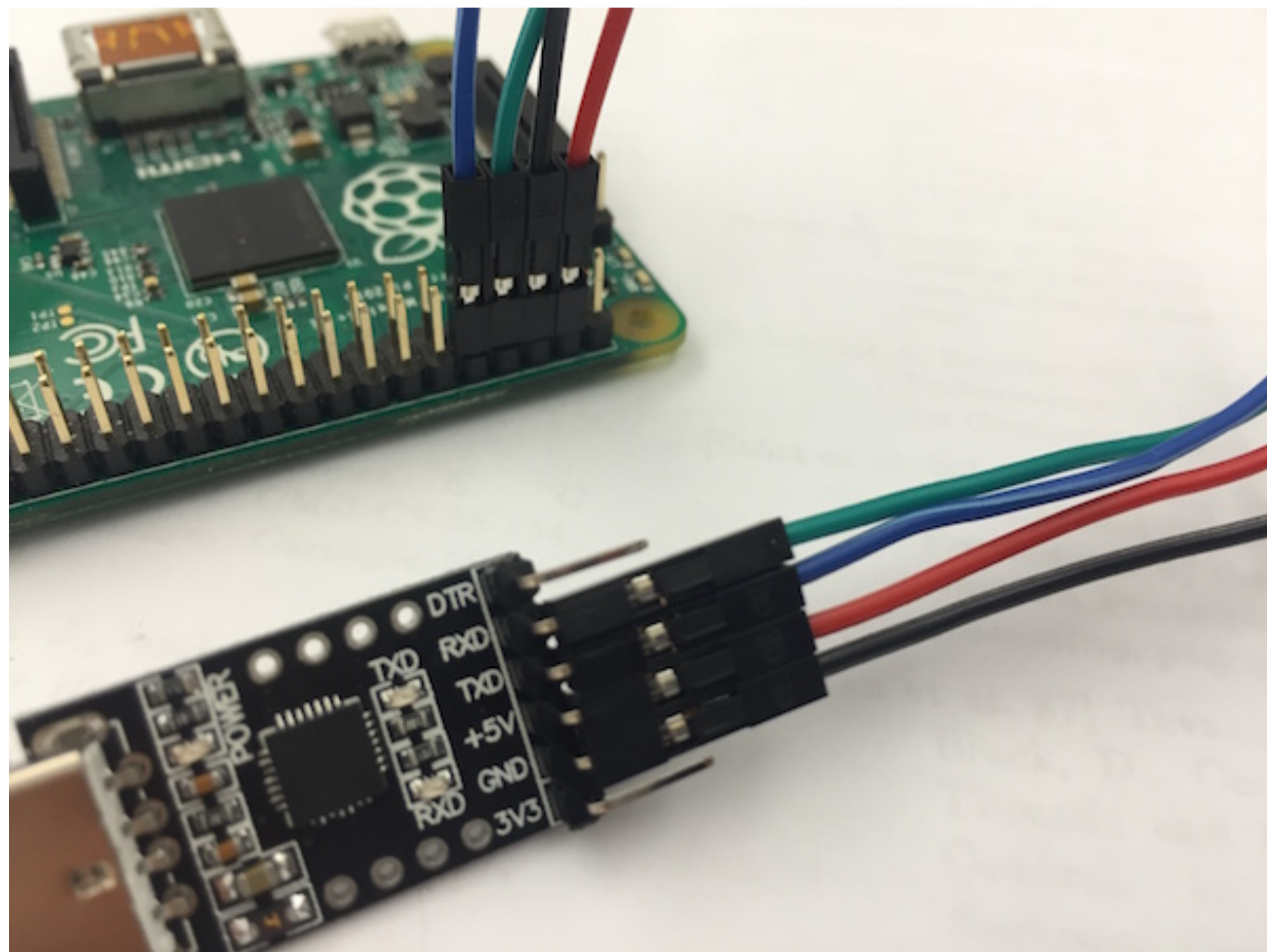
```
// hot wire TX
```

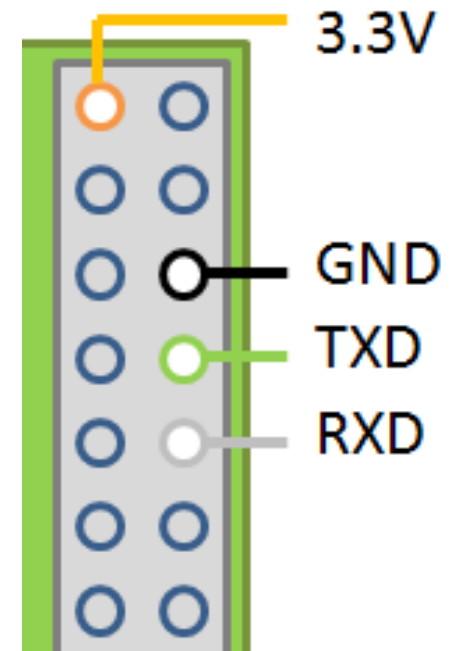
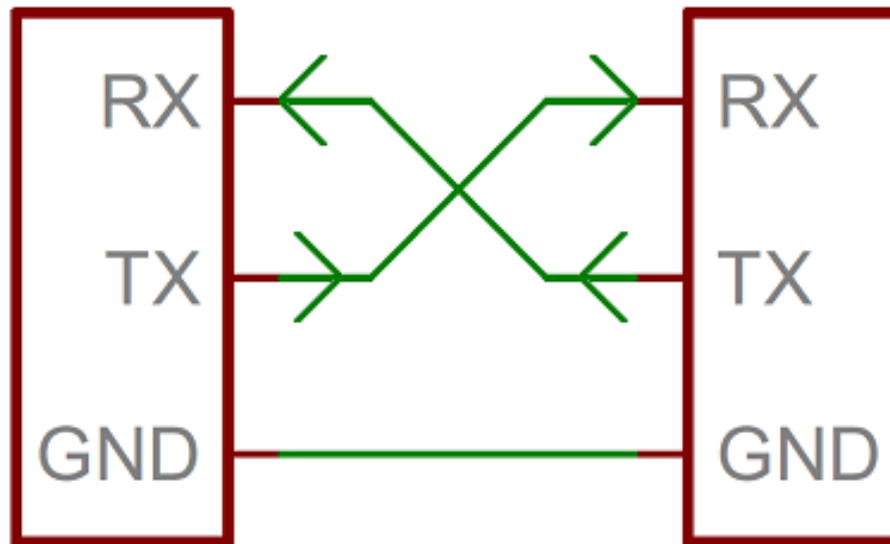
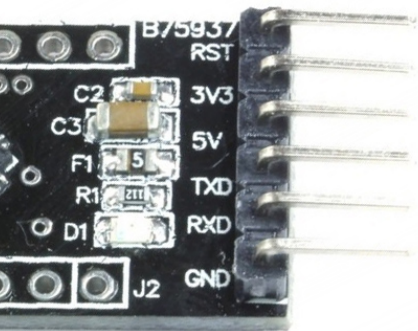
```
// device = tty (teletype)
```

```
// baud rate = 9600
```

```
% screen /dev/tty.SLAB_USBtoUART 9600
```

```
CTRL-A K - to exit
```





```
% screen /dev/tty.SLAB_USBtoUART 115200
```

uart.c

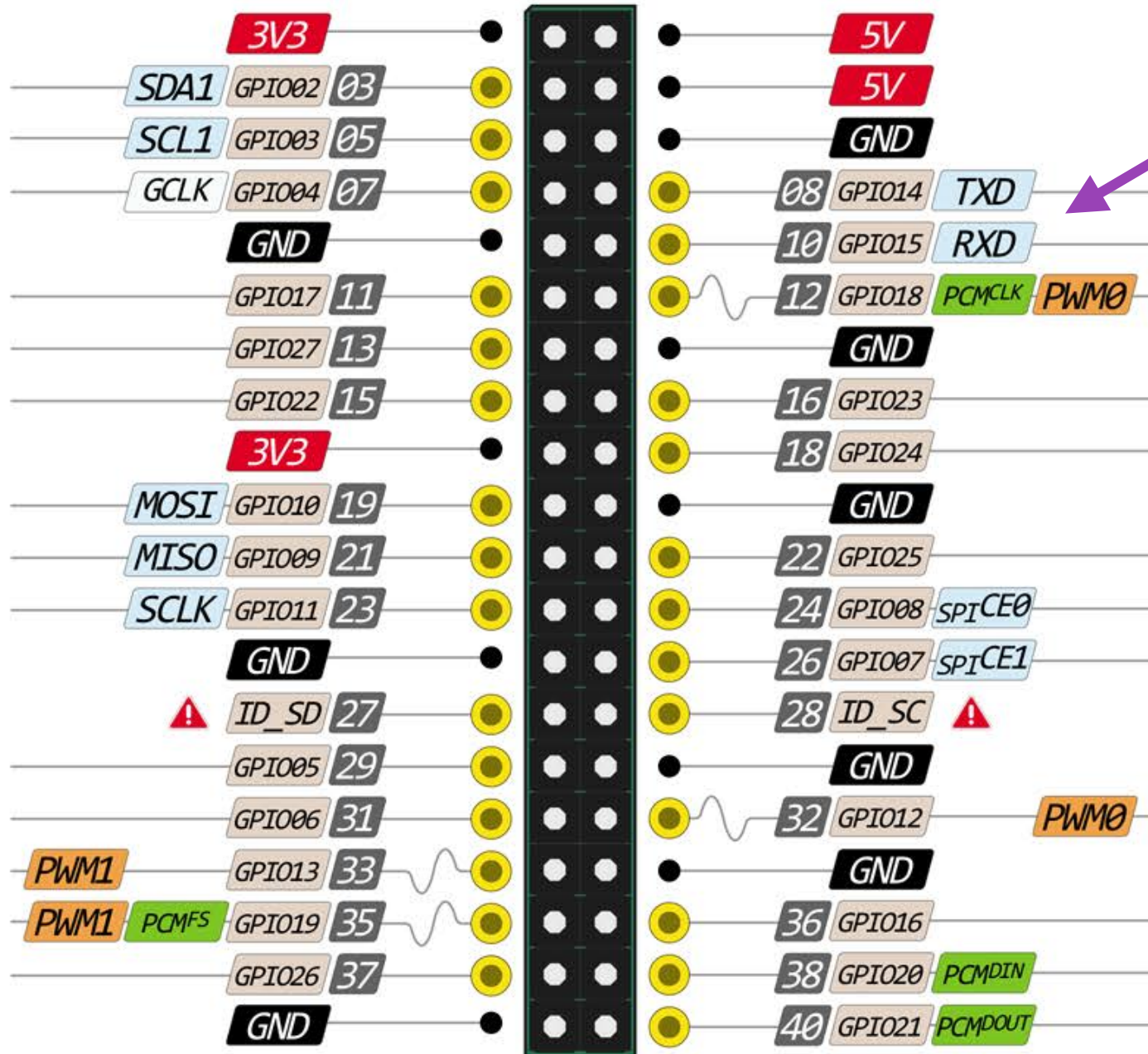
Universal Asynchronous Receiver-Transmitter

GPIO ALT Function

BCM2835 has 54 general-purpose I/O pins. Every pin can be input, output, or one of 6 special functions (ALT0-ALT5), specific to each pin.

PIN	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPI014	TXD0	SD6				TXD1
GPI015	RXD0	SD7				RXD1

GPIO Alternate Functions



Power of Types and Pointers

```
struct gpio {  
    unsigned int fsel[6];  
    unsigned int reservedA;  
    unsigned int set[2];  
    unsigned int reservedB;  
    unsigned int clr[2];  
    unsigned int reservedC;  
    unsigned int lev[2];  
};
```

Address	Field Name	Description	Size	Read/ Write
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	R/W
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	R/W
0x 7E20 0018	-	Reserved	-	-
0x 7E20 001C	GPSET0	GPIO Pin Output Set 0	32	W
0x 7E20 0020	GPSET1	GPIO Pin Output Set 1	32	W
0x 7E20 0024	-	Reserved	-	-
0x 7E20 0028	GPCLR0	GPIO Pin Output Clear 0	32	W
0x 7E20 002C	GPCLR1	GPIO Pin Output Clear 1	32	W
0x 7E20 0030	-	Reserved	-	-
0x 7E20 0034	GPLEV0	GPIO Pin Level 0	32	R
0x 7E20 0038	GPLEV1	GPIO Pin Level 1	32	R

```
volatile struct gpio *gpio = (struct gpio *)0x20200000;  
gpio->fsel[0] = ...
```

// BCM2835-ARM-Peripherals.pdf

// Sec 2: Mini-UART, SPI0, SPI1, pp 8-19

```
struct UART {  
    unsigned data; // I/O Data  
    unsigned ier;  // Interrupt enable  
    unsigned iir;  // Interrupt identify/fifo  
    unsigned lcr;  // line control register  
    unsigned mcr;  // modem control register  
    unsigned lsr;  // line status  
    unsigned msr;  // modem status  
    unsigned scratch;  
    unsigned cntl; // control register  
    unsigned stat; // status register  
    unsigned baud; // baud rate register  
} ;
```

echo.c

loop back test

C Strings

"cs107e" =

\0
64
37
30
31
73
63

```
char arr[] = ['c', 's', '1', '0', '7', 'e', '\0'];  
//char arr[] = "cs107e";  
char *ptr = "cs107e";
```

```
char ch;  
ch = arr[1]; // ok?  
ch = ptr[1];
```

```
arr = ptr;  
ptr = arr;
```

```
char **ptrptr;
```

```
ptrptr = &arr;  
ptrptr = &ptr;
```

String Functions in string.h

<code>strcat(s1,s2)</code>	Concatenate s2 to s1
<code>strncat(s1,s2,n)</code>	Concatenate at most n characters of s2 to s1
<code>strcpy(s1,s2)</code>	Copy s2 to s1; Note the direction of the copy!
<code>strncpy(s1,s2,n)</code>	Copy first n characters of s2 to s1
<code>strlen(s)</code>	Return length of string s, not counting '\0'
<code>strcmp(s1,s2)</code>	Compare s1 with s2; Return integer less than zero, equal to zero, or greater than zero
<code>strncmp(s1,s2,n)</code>	Compare only the first n characters of s1 and s2
<code>strchr(s,c)</code>	Return a pointer to first occurrence of character c in string s; return NULL if not found
<code>strrchr(s,c)</code>	Return a pointer to last occurrence of character c in string s; return NULL if not found
<code>strstr(s1,s2)</code>	Return a pointer to the first occurrence of string s1 in string s2; return NULL if not found
<code>strstr(s1,s2)</code>	Return a pointer to the first occurrence of string s1 in string s2; return zero if not found

```
size_t strlen(const char *str)
{
    for (const char *s = str; *s; ++s)
        ;
    return (s - str);
}
```

```
// strlen("a")?
// strlen(NULL)?
// strlen('a')?
```

// Assignment 3

```
printf(const char *format, ...);
```

```
printf(“%d, %d”, 1, 2);
```

```
printf(“%d, %d, %d”, 1, 2, 3);
```

```
printf(“%d, %d, %d”, 1, 2);
```

```
// Read about #include <stdarg.h>
```

```
// in the assignment writeup to
```

```
// to learn to use functions with
```

```
// variable numbers of arguments
```