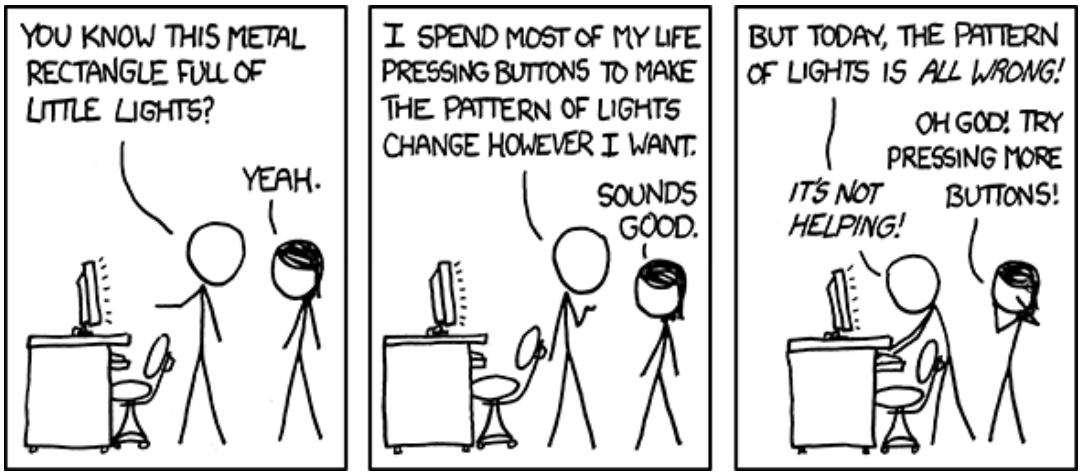
Admin

Assign 1 grades posted
Assign 2 due Tue

Lab 3 serial, gdb, strings
Assign 3 printf
Teach your Pi to talk, yeah!



Today: Serial communication

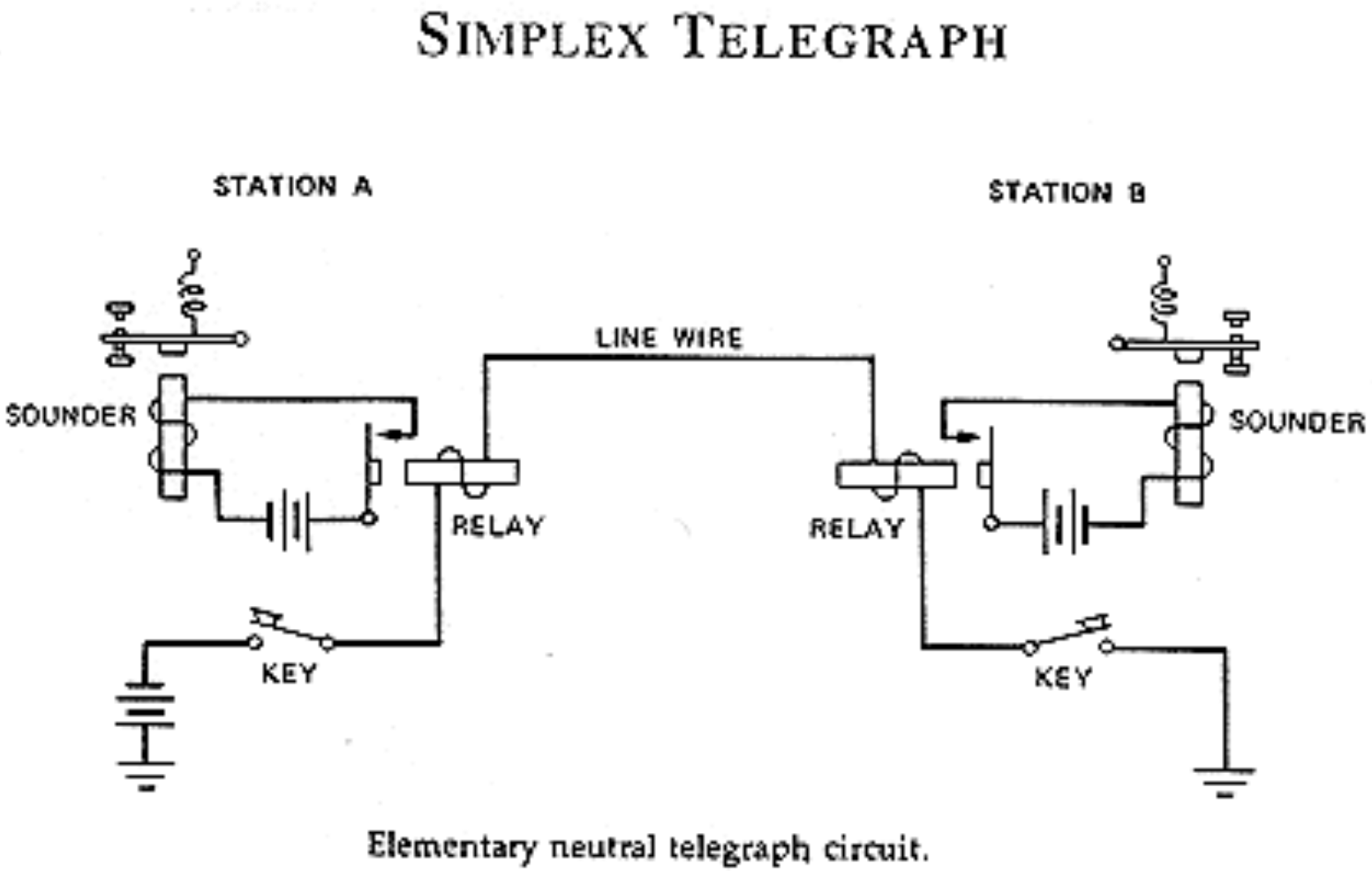
Serial protocol

ASCII character set, strings, printing

Uart peripheral

GDB debugger

Telegraph circuit

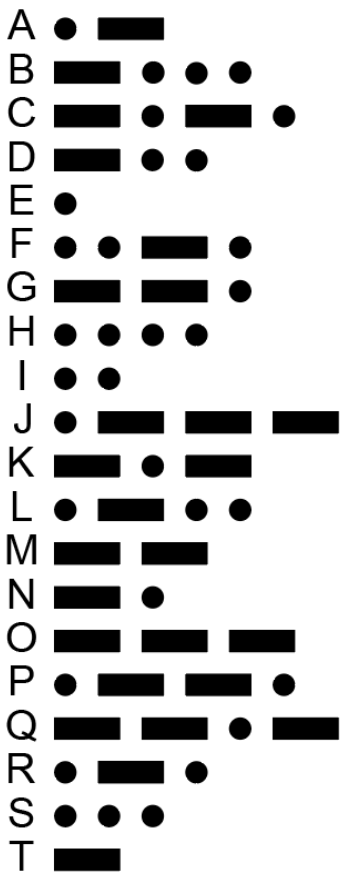


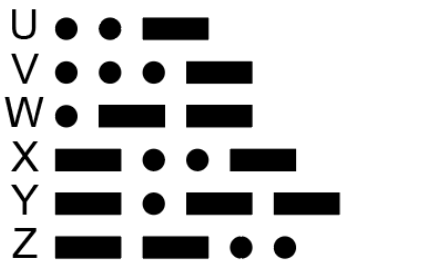
Elementary neutral telegraph circuit.

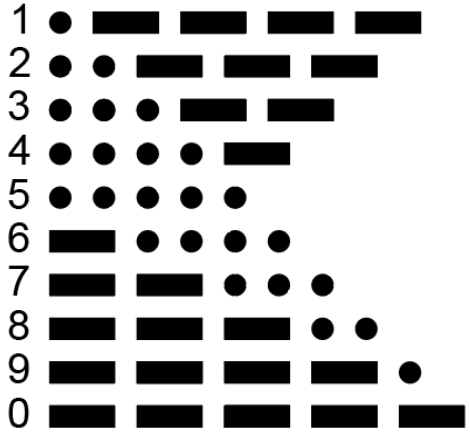
http://people.seas.harvard.edu/~jones/cscie129/nu_lectures/lecture5/elecmag_tel/morse_tel.html

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.



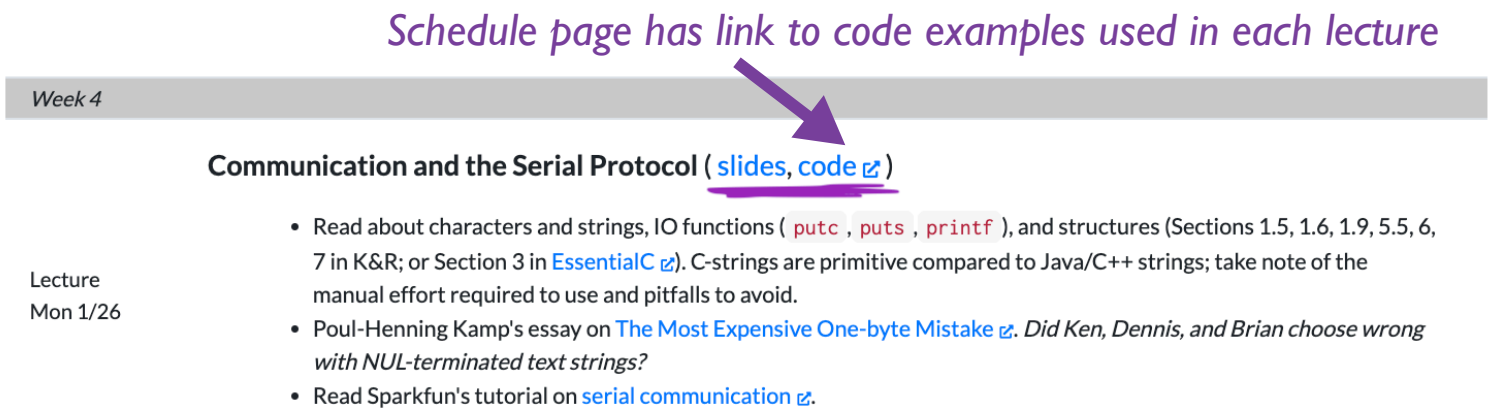




https://en.wikipedia.org/wiki/Morse_code

blink_gpio/blink.c
->
sos/sos.c

Schedule page has link to code examples used in each lecture



Week 4

Communication and the Serial Protocol (slides, code)

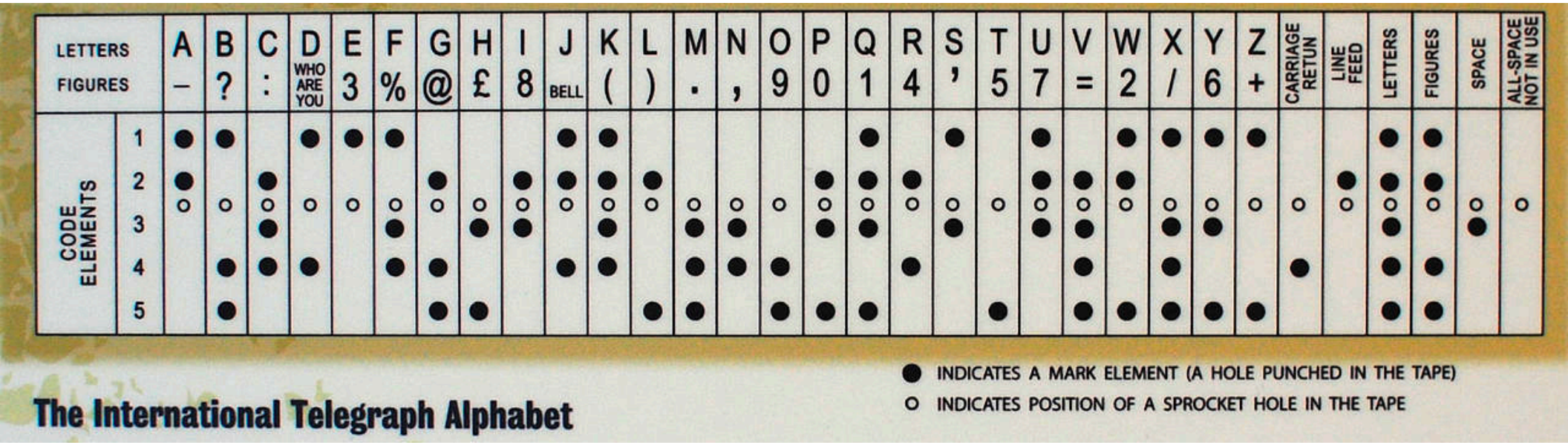
Lecture
Mon 1/26

- Read about characters and strings, IO functions (`putc` , `puts` , `printf`), and structures (Sections 1.5, 1.6, 1.9, 5.5, 6, 7 in K&R; or Section 3 in EssentialC). C-strings are primitive compared to Java/C++ strings; take note of the manual effort required to use and pitfalls to avoid.
- Poul-Henning Kamp's essay on The Most Expensive One-byte Mistake . *Did Ken, Dennis, and Brian choose wrong with NUL-terminated text strings?*
- Read Sparkfun's tutorial on serial communication .

Teletype

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

5-bit Baudot Code (1870)



LETTERS / FIGURES		A –	B ?	C :	D WHO ARE YOU	E 3	F %	G @	H £	I 8	J BELL	K (	L)	M .	N ,	O 9	P 0	Q 1	R 4	S '	T 5	U 7	V =	W 2	X /	Y 6	Z +	CARRIAGE RETUN	LINE FEED	LETTERS	FIGURES	SPACE	ALL-SPACE NOT IN USE
CODE ELEMENTS	1	●	●		●	●	●				●	●						●		●		●		●	●	●	●			●	●		
	2	●	○	●	○	○	○	●	○	●	●	●	●	○	○	○	●	●	●	○	○	●	●	●	○	●	○	○	●	●	●	○	○
	3	○		●			●	○	●	●	○	●	○	●	●		●	●	○	●		●	●		●	○				●	○	●	
	4		●	●	●		●	●			●	●		●	●	●	●		●				●		●		●			●	●		
	5		●					●	●				●	●		●	●	●			●		●	●	●	●	●				●		

● INDICATES A MARK ELEMENT (A HOLE PUNCHED IN THE TAPE)
○ INDICATES POSITION OF A SPROCKET HOLE IN THE TAPE

The International Telegraph Alphabet

https://en.wikipedia.org/wiki/Baudot_code

Baud = Number of symbols per second

ASCII Code

```
$ man 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" =

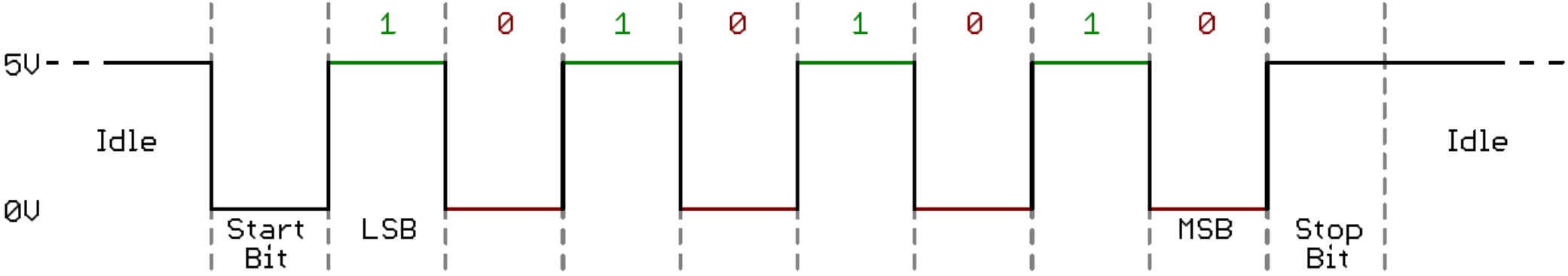
\0
65
37
30
31
73
63

0x68 represents character 'h'

ASCII 7-bit

(8th bit used by extended char sets)

Asynchronous Serial Communication



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

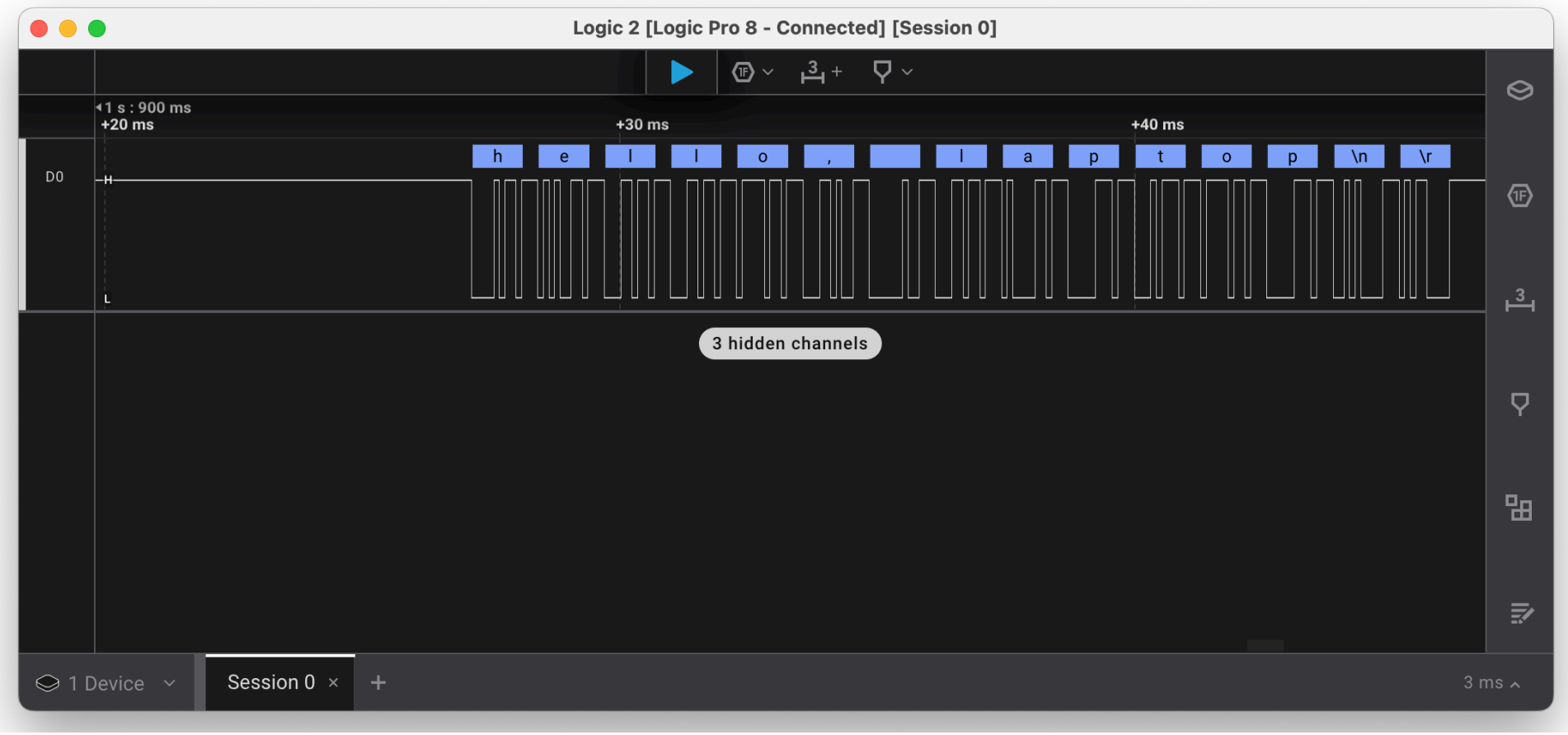
9600 baud = 9600 bits/sec

(1000000 usecs)/9600 ~ 104 usec/bit

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

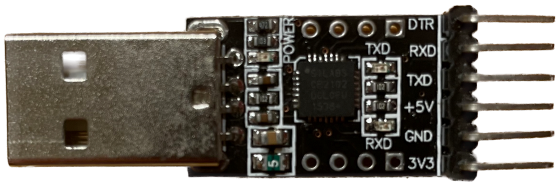
sos.c -> serial.c

Demo of Logic Analyzer

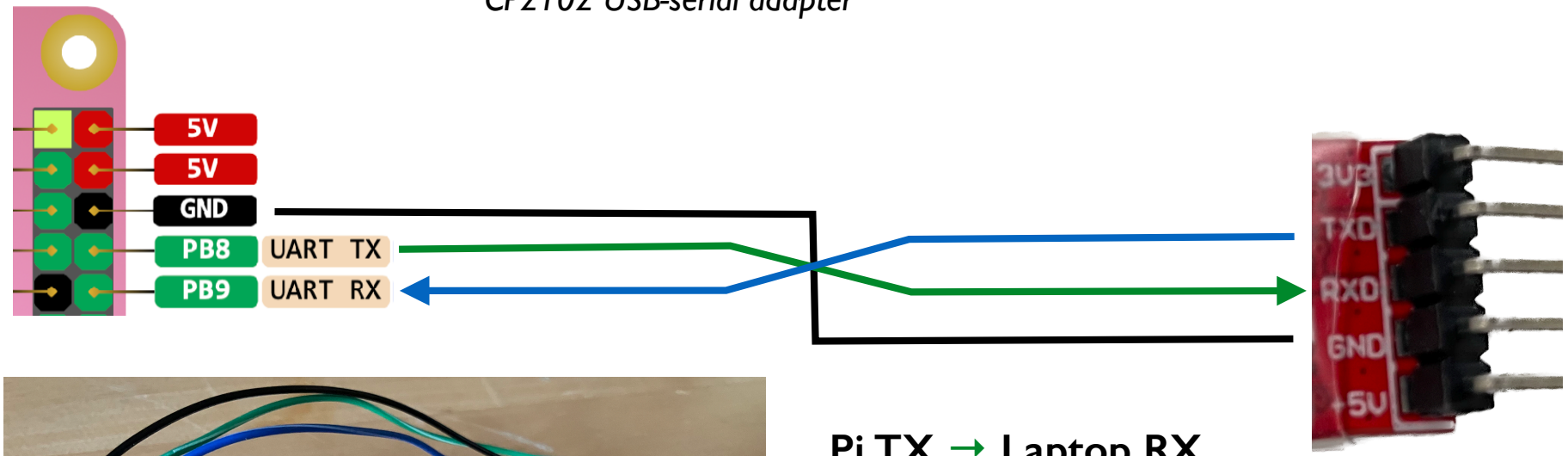


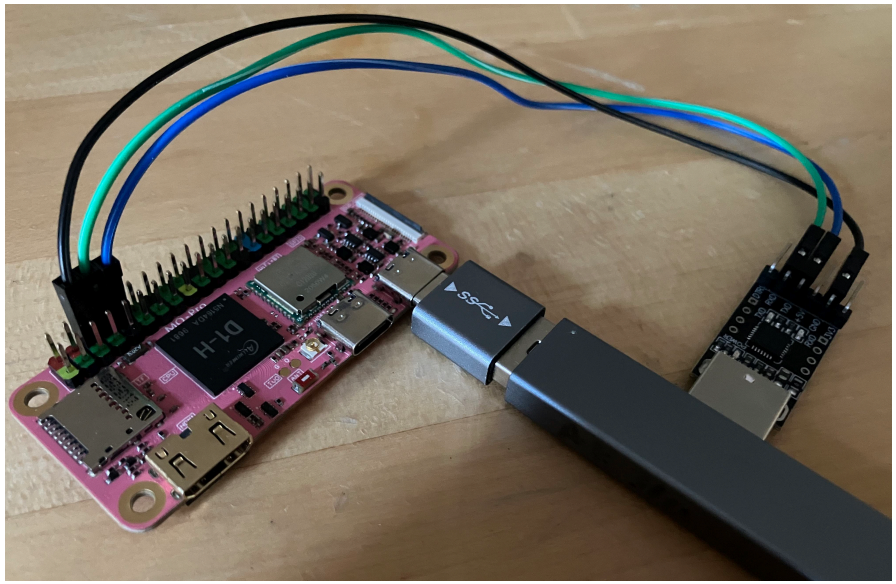
Making the invisible visible!

Serial connection



CP2102 USB-serial adapter





Pi TX → Laptop RX
Pi RX ← Laptop TX

Ref: CS107e uart guide

Universal Asynchronous Receiver-Transmitter

9.2 UART

9.2.1 Overview

The universal asynchronous receiver transmitter (UART) provides an asynchronous serial communication with external devices, modem (data carrier equipment, DCE). It performs serial-to-parallel conversion on the data received from peripherals and transmits the converted data to the internal bus. It also performs parallel-to-serial conversion on the data that is transmitted to peripherals.

The UART has the following features:

- Compatible with industry-standard 16450/16550 UARTs
- Supports IrDA-compatible slow infrared (SIR) format
- Two separate FIFOs: one is RX FIFO, and the other is TX FIFO
  - Each of them is 64 bytes (For UART0)
  - Each of them is 256 bytes (For UART1, UART2, UART3, UART4, and UART5)

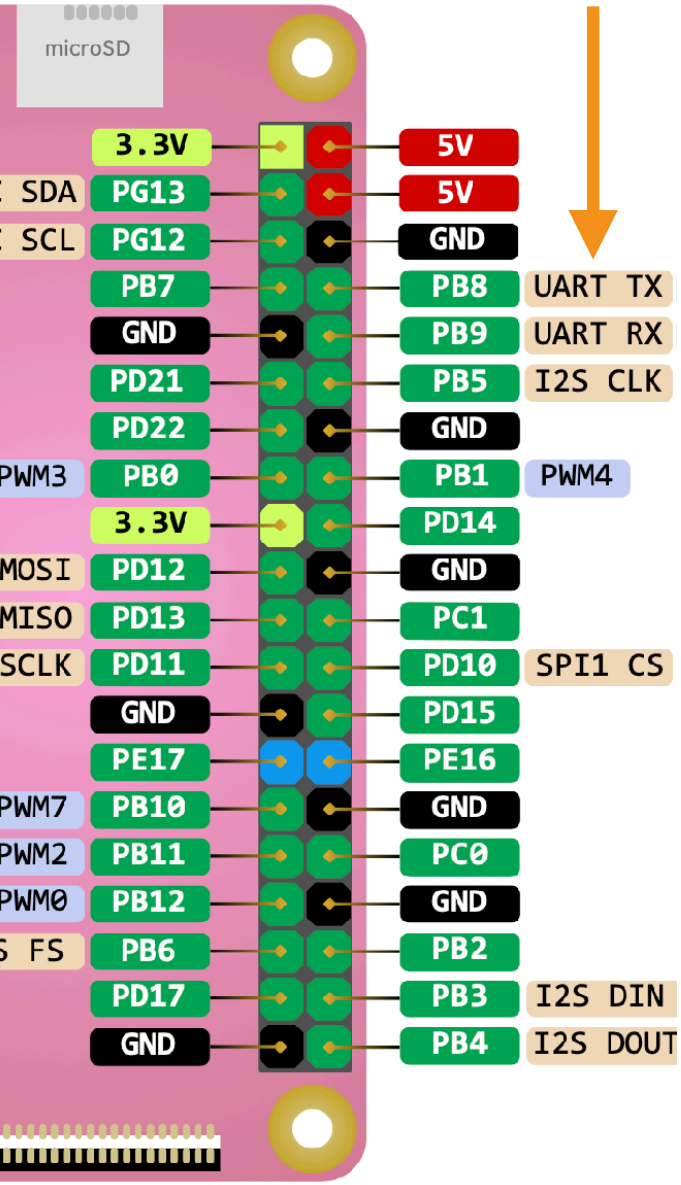
Ref: D1-H User Manual p.899

Register Name	Offset	Description
UART_RBR	0x0000	UART Receive Buffer Register
UART_THR	0x0000	UART Transmit Holding Register
UART_DLL	0x0000	UART Divisor Latch Low Register
UART_DLH	0x0004	UART Divisor Latch High Register
UART_IER	0x0004	UART Interrupt Enable Register
UART_IIR	0x0008	UART Interrupt Identity Register
UART_FCR	0x0008	UART FIFO Control Register
UART_LCR	0x000C	UART Line Control Register
UART_MCR	0x0010	UART Modem Control Register
UART_LSR	0x0014	UART Line Status Register
UART_MSR	0x0018	UART Modem Status Register
UART_SCH	0x001C	UART Scratch Register

```
struct {
   union {
       uint32_t rbr;   // receive buffer register
       uint32_t thr;   // transmit holding register
       uint32_t dll;   // divisor latch lo
    };
    union {
       uint32_t dlh;   // divisor latch hi
       uint32_t ier;   // interrupt enable register
    };
    union {
        uint32_t iir;   // interrupt identification register
        uint32_t fcr;   // FIFO control register
    };
    uint32_t lcr;       // line control register
    uint32_t mcr;       // modem control register
    uint32_t lsr;       // line status register
```

Ref: uart.c source

GPIO Alternate functions



7:4	R/W	0xF	PB9_SELECT PB9 Select 0000:Input 0010:DMIC-DATA2 0100:TWI2-SDA 0110:UART0-RX 1000:Reserved 1110:PB-EINT9	 0001:Output 0011:PWM6 0101:SPI1-MISO/DBI-SDI/DBI-TE/DBI-DCX 0111:UART1-RX 1001:Reserved 1111:IO Disable
3:0	R/W	0xF	PB8_SELECT PB8 Select 0000:Input 0010:DMIC-DATA3 0100:TWI2-SCK 0110:UART0-TX 1000:Reserved 1110:PB-EINT8	 0001:Output 0011:PWM5 0101:SPI1-HOLD/DBI-DCX/DBI-WRX 0111:UART1-TX 1001:Reserved 1111:IO Disable

Ref: D1-H User Manual p.1097

Hello, world!

```
$ tio -b 115200 /dev/cu.usbserial-0001
```

C-strings

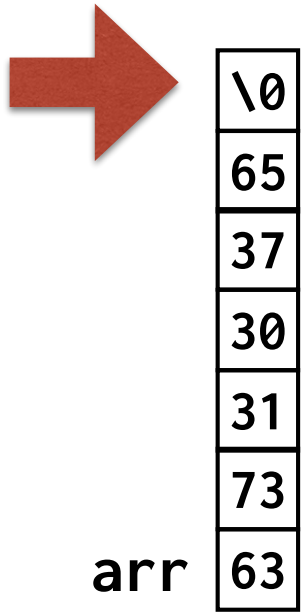
```
// Note '\0' at the end!
char arr[] = {'c','s','1','0','7','e','\0'};

// short cut
char arr[] = "cs107e";

char ch = arr[1]; // ok!
ch = arr[25];     // not ok, what happens?

char *ptr = arr; // ok!
arr = ptr;       // not ok, why?

const char *ptr = "cs107e";
ch = ptr[1];      // ok?
ptr[1] = 'd';     // ok?
```



Operations on C-strings

```
size_t strlen(const char *str) {
    int n = 0;
    while (str[n] != '\0') {
        n++;
    }
    return n;
}

// strlen("a")?
// strlen(NULL)?
// strlen(&ch)?
```

Strings module

```
$ man string
STRING(3)                    Library Functions Manual
STRING(3)

NAME
     index, rindex, stpcpy, strcasecmp, strcat, strchr, strcmp, strcpy, strcspn, strerror,
     strlen, strncasecmp, strncat, strncmp, strncpy, strpbrk, strrchr, strsep, strspn,
     strstr, strtok - string specific functions


     Standard C Library (libc, -lc)

SYNOPSIS
     #include <strings.h>

     char *
     index(const char *s, int c);

     char *
     rindex(const char *s, int c);

     int
     strcasecmp(const char *s1, const char *s2);
     ...
```

libc

strlcat(dst,src,n) Concatenate n chars of src to end of dst

strlen(s) Return length of s, not counting ‘\0'

strcmp(s1,s2) Lexicographic ordering of s1 vs s2

strtonum(str,endptr) Convert string to number

libmango!

Printf module

```
#include "printf.h"

// print formatted output

printf("%d, %d\n", 1, 2);
printf("%x\n", 0x20000030);
printf("%c\n", 'a');
printf("%s\n", "hello");
```

Printf implementation is almost entirely code to **construct formatted strings**!

Gdb debugger

Debugger is incredibly useful

Allows you to run your program in a monitored context
Can set breakpoints, examine state, change values, reroute control, and more

gdb has simulation mode where it pretends to be an RISC-V processor, running on your laptop 🙌

Good approximation (not exact, e.g. no peripherals)

Let's try it now!

Debugger FTW!

Invaluable tool for Assign3

Lab 3 has exercises to practice

👀 Read gdb guides 👀

http://cs107e.github.io/guides/gdb/
http://cs107e.github.io/guides/gdb-quickref