

ECE 471 – Embedded Systems

Lecture 24

Vince Weaver

<http://web.eece.maine.edu/~vweaver>

vincent.weaver@maine.edu

5 November 2018

Announcements

- HW#8 was posted
- Keep thinking about projects, topic due Friday.



HW#7 Review

- Questions
 - Disadvantage of SPI?
More wires, no standard, no errors
 - Advantage of SPI?
Lower Power, Full Duplex, No max speed
 - TMP36 on end of cable.
Voltage Drop, Noise?
Datasheet has two options, convert to current, or an extra resistor.



- Minimum frequency of 10kHz or results invalid. Maybe cannot go this fast if bitbanging via GPIO. Also context switch in middle, Linux not realtime?
- Linux
 - /dev/null
 - /dev/full
 - /dev/zero
 - /dev/random – give explanation on sources of randomness (entropy), pseudo-randomness, etc.
 - Mention related DOS/Windows compatibility issue



C string review

String manipulation is famously horrible in C. There are many ways to get the "YES" and "t=24125" values out of the text file for HW#8. Any you choose is fine.

- There are multiple ways to read files into a string in C

Assume `char string[1024];`

- `fd=open("filename",RD_ONLY);`

 - `read(fd,string,1023); close(fd);`

- `FILE *fff; fff=fopen("filename","r"); fread(b,1,1023,fff); fclose(fff)`



- You can also use `fgets(buffer, size, fff);`
- Advanced: use `mmap()`
 - C strings
 - In C, characters are NUL (0) terminated character arrays (usually 8-bit bytes). Usually ASCII or UTF8
 - Other languages might be unicode, 16-bit, `wchar`
 - You can use either pointer or array access to get a value (`string[0]` is the same as `*string`)
 - Note that double quotes indicate a string, while single quotes indicate a single character
 - It is very easy to accidentally go off the end of a string



and corrupt memory

- Alternatives? Fancy libraries? Pascal strings (where first char is the length?)
- Always be sure your strings are terminated, otherwise bad things can happen (and not all C string manipulation functions do this properly, see `strcpy()`, `strncpy()`, `strlcpy()`)
- Finding a location / substring in a larger string
 - If you trust the Linux kernel developers to keep a “stable ABI” you can assume the temperature will always be a fixed offset and hard code it. This can be



a bit dangerous.

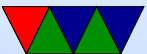
- You can use the `scanf()` series of functions to parse the string (either `fscanf()` directly, or `sscanf()` on the string)

One helpful hint, putting a '*' in a conversion (like `%*s` tells `scanf` to read in the value but ignore it.

- You can use the `strstr()` search for substring C-library function, maybe in conjunction with `strtok()`
- You can manually parse the array.

Using array syntax, something like:

```
i=0; while(string[i] != 0) {
```




```
if (string[i]=='t') break; i++ }
```

Using pointer syntax, something like:

```
char *a; a=string; while(*a!=0) {  
if (*a=='t') break; a++; }
```

- Pointing into a string
 - If you searched for "t=" you might now have a pointer a to something like "t=12345". To point to 12345 you can just add 2 to the string pointer.
 - `printf("%s\n",string+2);`
 - `printf("%s\n",&string[2]);`
- Converting string to decimal or floating point



- `atoi()` converts string to integer. What happens on error?
- `strtol()` will give you an error but is more complex to use
- `atof()` and `strtod()` will do floating point
- Comparing strings
 - Can you just use `==`? NO!
 - Be careful using `strcmp()` (or even better, `strncmp()`) they have unusual return value less than, 0 or greater than depending. 0 means match
So you want something like



```
if (!strcmp(a,b)) do_something();
```

