ECE471: Embedded Systems

Fall 2018 Second Midterm Exam

Name: Steve Castelle

Closed Book, Closed Notes.

This page intentionally left blank.

Q_2: 25/25
Q_3: 19/20
Q_4: 30/30
Q_5: 18/25

92/100
+2
94/100

25/25

1. Startup and Booting

(a) What part of the Raspberry Pi (hardware) is in charge of the initial boot process? Is this normal for an ARM board?

The GPU is in charge of the initial boot process on the Raspberry Pi. This is not normal for an ARM board and is why the Raspberry Pi has an unusual way of booting.

(b) What is the generic name for the code/program responsible for loading the operating system kernel into memory and running it?

The generic name is the bootloader

(c) Why is the `/boot` partition on the Pi a FAT32 filesystem?

It is a FAT32 filesystem because it is simple, compatable, and uses low resources

2. Real-time

(a) Would an embedded system responsible for automatically shutting down a nuclear reactor within 1ms of detection of dangerous temperatures that would lead to destruction of the plant be considered hard, firm, or soft real-time? Why?

Hard real-time because if it misses the 1ms deadline the results are deadly. Destruction of the planet deadly.

(b) Would an embedded system responsible for blinking LED lights in time with Christmas music be considered hard, firm, or soft real-time? Why?

Soft real-time because if it misses when it is supposed to play to the christmas music it becomes less useful as the time passes. It still will look cool, but the more it is off, the less useful it is.

19/20

3. Operating Systems and Security

(a) Operating Systems

i. What is one benefit of using an operating system on an embedded device?

The operating system does a lot of background stuff for you and makes things easier.

-1 what type of background stuff?

ii. What is one drawback of using an operating system on an embedded device?

The overhead of the operating system

✓

(b) Security

You are designing an entertainment system for the interior of a car. You want to support transmitting video via bluetooth to the in-car display, and have the car's interior lights automatically dim once the video starts playing. This involves sending messages from the in-car display directly onto the CANBUS of the car to control the lights.

i. What could go wrong with this embedded system if code correctness and security are not carefully maintained?

✓ If code correctness and security or worse are not maintained, an unauthorized user could get access to the interior lights and video display. This could be bad cause if the user messes with the lights and display, it could distract you from driving.

ii. You already have similar code from a related project where the video was transferred to a laptop and auto-dimmed the keyboard backlights. Why might you want to be careful re-using code from this application in the car application?

✓ This code from this application may work for the laptop, but doesn't mean it will do the same thing with the car. Copying it could cause something else to occur in your car which could be devastating.

30/30

4. Embedded Busses

For the following scenarios, say which bus you would use (*1-wire*, *SPI*, or I^2C) and one brief reason why you would use it in this situation.

(a) A device where you want to attach multiple displays and sensors with short cables, but only have two GPIOs available to drive everything.

I²C because it uses one clock line and one data line to communicate with multiple devices

(b) An outdoor temperature probe 40 yards (40m) away from your embedded system.

1-wire because it can go 3xxs meters, whereas I²C and SPI only can go a couple meters.

(c) The temperature/pressure probe in a low-power logging weather station located far from any power source, where battery life is the primary concern.

SPI because it uses low power

(d) Hooking a display up to a low-end STM32L embedded board and you have to bitbang the interface yourself without using libraries.

I²C because you can easily bit-bang it. We did it in homework too.

14/25

5. Linux SPI and C

(a) Assume the code below is running on a Raspberry Pi that has the SPI pins properly connected to a MCP 3008 Analog Digital Converter. Fill in the five missing code comments.

```
#define LENGTH 3

int spi_fd;
struct spi_ioc_transfer spi;
unsigned char data_out[LENGTH],data_in[LENGTH];

spi_fd=open("/dev/spidev0.0", O_RDWR); // opens the spi communication file
if (spi_fd < 0) {                      // It it didn't open properly return
   fprintf(stderr,"ERROR!\n");
   exit(1);
}

data_out[1] = ((2 & 0x7) << 4);        // We want to read channel 2
data_out[1]|=0x80;                     // Config for single ended
data_out[2] = 0;                       // Dummy value, ignored by device

memset(&spi,0,                         // reset the spi struct
       sizeof(struct spi_ioc_transfer));

spi.tx_buf    = (unsigned long)&data_out;
spi.rx_buf    = (unsigned long)&data_in;
spi.len               = 3;         // 3 bytes
spi.speed_hz          = 100000;    // 100kHz
spi.bits_per_word     = 8;         // 8 bits per word
spi.cs_change         = 0;         // don't auto change the chip-select

result = ioctl(spi_fd,                 // uses spi struct to send and receive data
                  SPI_IOC_MESSAGE(1), &spi);   (for spi communication)

value = ((data_in[1]&0x3) << 8)        // need mask to get the voltage from data in
                | (data_in[2] & 0xff);
value = (value*3.3)/1024.0;            // Convert back to voltage
                                       // See data sheet for details
printf("\t%.3lf",value);
```

(b) Why does the code use an ioctl() for the data transfer and not simply a `write()` as the i2c interface would do? This is because you need input output control to send the struct. With write(), you can't send the spi struct.

-5 need to read + write

6. Extra Credit

(a) In C, Which of the following will print a message if two independent strings have identical contents?

i.

```
if (string1==string2) printf("Match!\n");
```

ii.

```
if (&string1==&string2) printf("Match!\n");
```

iii.

```
if (strcmp(string1,string2)) printf("Match!\n");
```

iv. None of the above.

strcmp returns 0 if match

+2