Timothy Gedney
CSCE317 – Systems Engineering
April 22nd, 2021
Professor Bakos

Lab #3

Part 1: Interfaces

Given a microcontroller with 5 interfaces, 2 SPI transceivers with a clock rate up to 10MHz, 1 I2C transceiver with a clock rate up to 400kHz, and 2 UART transceivers capable of running at 115,200 baud. From there I calculated the maximum bits per second on each transceiver to be 10 million bps (1.25 million Bps), 400,000 bps (50,000 Bps), and 115200 bps (14,400 Bps) respectively. With this information calculated I could then determine which peripherals would be connected to which interfaces.

I started with calculating to find the largest transactions to find where the bandwidths might become bottlenecks. I found that the video camera and the wireless communication peripherals would require the largest bandwidths and began with finding places for them. With the current project parameters, the wireless communication can't fit into the bandwidth of any of the interfaces. On average, the wireless communication will be sending 153,634 bytes 10 times a second meaning it will be about 1,536,340 bits per second. This requires more bandwidth than even the SPI can provide, even though the wireless communication can't interface with it. The parameter for this peripheral will have to be changed to sending its data 1 every 5 seconds, sending out on average 30,727 bytes per second, which will fall into the bandwidth of the I2C. The wireless communication will be the sole connection of the I2C given its increased bandwidth.

After that the video camera had the second largest bandwidth requirement. I originally assumed the frames per second would be 10 but this led to a bandwidth requirement of 1,536,000 bytes per second which can't be met. I then decided to lower the frames per second to a more manageable bandwidth of 5 frames per second which leads to 768,000 bytes per second. With this, the video camera will be solely connected to one of the two SPI interfaces since they can handle 1.25 million bytes per second.

Timothy Gedney
CSCE317 – Systems Engineering
April 22nd, 2021
Professor Bakos

With the remaining peripherals it came down to making sure the interfaces could match the varying bitrates and send/receive from both sides. I decided that the accelerometer, gyroscope, and GPS should all be connected to the remaining SPI interface using the two interrupt lines connected to the gyroscope and GPS. I chose this because they collectively will be sending about 542 bytes per second and frequencies of 20 (and 1), 25, and 1 respectively so they shouldn't interfere with one another's transmissions. Also, since the accelerometer requires correction once every second it works well with the SPI interface since SPI is bidirectional and can receive the 2-bit correction on one of its returns.

Then, I decided that the battery controller should be connected to the first UART interface and that the motor controller should be connected to the second UART interface. I chose those connections because the battery transmits 8 bytes and the motor controller transmits 32 bytes which both are acceptable for the UART. Both peripherals also fall well within the UART's bandwidth since they all transmit less bytes per second than the UART's 14,400 bytes per second.

Excel table I used to calculate and make decisions for my design:

	A	B	C	D	E	F	G
1	DEVICE	COMMS	DIRECTION	FREQUENCY	FREQ HZ	DATA	Data per second
2	Motor Controller	Uart, I2c	RX	1 per 10ms	100hz	32b	3200 Bps
3	Battery Controller	Uart, I2C, SPI	TX	1 per second	1hz	8b	8 Bps
4	GPS Controller	Uart, SPI	TX	1 per 5 seconds	0.2hz	8b	~2 Bps
5	Accelerometer	I2C, SPI	TX	20 per second	20hz	12b	240 Bps
6			RX	1 per second	1hz	2b	2Bps
7	Gyro Scope	I2C, SPI	TX	25 per second	25hz	12b	300 Bps
8	Video Camera	SPI	TX	X	X	X	X
9	10 fps			10 per second	10hz	153,600b	1,536,000 Bps
10	8 fps (Max possible)			8 per second	8hz	153,600b	1,228,800 Bps
11	5 fps			5 per second	5hz	153,600b	768,000 Bps
12	Wireless Coms	Uart, I2C	TX	X	X	X	X
13				10 per second	10hz	153,634b	1,536,340 Bps
14				1 per 5 seconds	0.2hz	153,634b	~30,726.8 Bps
15							
16	Microcontroller					OUT per sec	IN per sec
17	MC	TX	100hz	Uart, I2C	32b	3200	
18	BC	RX	1hz	Uart, I2C, SPI	8b		8
19	GPS	RX	0.2hz	Uart, SPI	8b		2
20	Accel	RX	20hz	I2C, SPI	12b		240
21		TX	1hz	I2C, SPI	2b	2	
22	Gyro	RX	25hz	I2C, SPI	12b		300
23	VC	RX	10hz	SPI	153,600b		768,000
24	WC	RX	10hz	Uart, I2C	153,634b	30,727	
25						802479	
26							
27	COMS TYPE	COMS SPEED	bps	Bps			
28			How many bits per second?				
29	SPI 1	10Mhz	10 million bits per second	1.25M Bps	VIDEO CAMERA		
30	SPI 2	10Mhz	10 million bits per second	1.25M Bps	ACCELEROMETER / GYRO SCOPE		
31	I2C	400Khz	400,000 bits per second	50,000 Bps	WIRELESS COMS		
32	Uart 1	115200 baud	115200 bits per second	14,400 Bps	BATTERY		
33	Uart 2	115200 baud	115200 bits per second	14,400 Bps	GPS		

Timothy Gedney
CSCE317 – Systems Engineering
April 22nd, 2021
Professor Bakos

Provided parameters that the MCU takes 20 instructions to process one byte of data and each instruction takes 1 clock cycle. The maximum read/write that the MCU would require in my design is 802,479 bytes per second. With that it would take 16,049,580 instructions to process that data. Since it takes on average 1 clock cycle then the minimum required clock rate of my MCU would be 16.04MHz.

Part 2:

With three tasks as follows:

Task	Instructions	Period
1	500,000	20
2	10,000,000	1000
3	2,000,000	100

Based on the number of instructions and periods alone I would have the earliest deadline first scheduling approach but this doesn't meet the projects parameters. Since the first task is for communicating with sensors that interrupt the processor this means it will require priority over the other two, less immediate, tasks. Thus, these tasks should be scheduled using rate monotonic scheduling so that the shorter task 1 may interrupt the longer two tasks to complete and prevent any hold up in important processor interrupts.

As the clock rate stands, with the provided 1 instruction per 1 cycle provided, these tasks aren't schedulable so a new clock rate must be calculated. Brute forcing the system in simple terms provides a general starting point of 0.1MHz (100,000Hz) which provides a schedulability rating of 1.65. This can be further solved for with a final MCU clock rate of 0.072MHz (72,000Hz) with a schedulability rating of about 1.96.

Timothy Gedney
CSCE317 – Systems Engineering
April 22nd, 2021
Professor Bakos

Part 3:

Table of provided values:

Rate (λ per second)	Period (ms)	RAM (bytes)
200 (s)	30	64
10 (v)	500 (50ms per frame)	153,600
2 (c)	10	256

With the provided values you can calculate the required total allocated RAM. Using Little's Law you can determine the average number of events/frames/commands in the queue at any given time. These values equal out to 6 events, 5000 frames, 0.02 commands respectively. With that information you can determine that on average there needs to be that much RAM allocated to the system. With this the average required allocated RAM needs to be 768,000,390 bytes.

To get more realistic results I would start by decreasing the number of video frames sent per a second. The video frames are one of the biggest drags on the entire system so reducing them would provide adequate room for improvement. Dropping the frames per second to my calculated number in the part 1, 5 frames per second, reduces the load upon that system drastically. A reduction to 5 frames per second would decrease the average items in queue to 250 frames, which would reduce the RAM required for the entire system from 768,000,390 bytes to 38,400,390 bytes which is still quite large but is more reasonable.