Solutions for Midterm #1 - EECS 145M Spring 2001

1a

- Acquire a computer with a digital I/O port and an internal clock
- Connect a pushbutton switch to one line of the digital input port and one line of the output port to an amplifier and fast light (like an LED)
- Write a computer program that detects a pushbutton, waits a random delay, prompts the subject by turning on the light and measures the time until the pushbutton is pressed
- Select a large number of candidates representing group d (racecar drivers) and group p (jet fighter pilots)
- After initial training and a good nights sleep, take as much reaction time data as possible, allowing for rest periods
- Compute the average reaction times $\bar{d}$ and $\bar{p}$, the standard deviation of the samples σ_d and σ_p, and the standard errors of the mean $\sigma_{\bar{d}}$ and $\sigma_{\bar{p}}$ and compute Student's t:

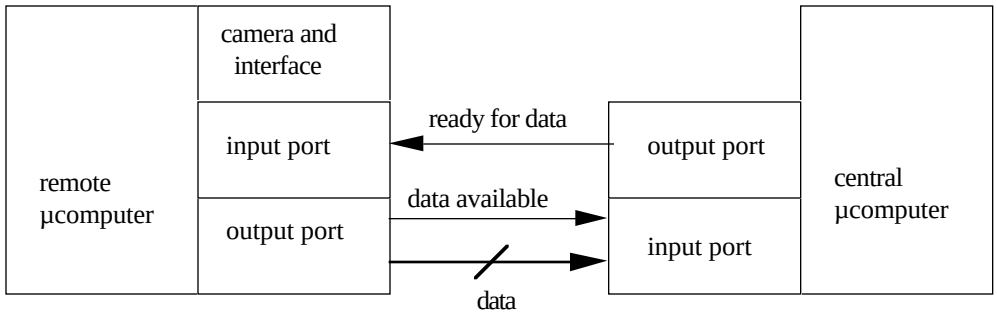
$$t = \frac{\Delta}{\sigma_\Delta} = \frac{\bar{d} - \bar{p}}{\sqrt{\sigma_{\bar{d}}^2 + \sigma_{\bar{p}}^2}} = \frac{\bar{d} - \bar{p}}{\sqrt{\sigma_d^2 / m_d + \sigma_p^2 / m_p}}$$

- Look up the probability of exceeding this value or t by chance. If the probability is < 0.1%, then the group with the lowest average has a faster reaction time.

[2 points off if only two subjects]

[1 point off if probability mentioned but not looked up or calculated]

2a



2b

1 car passes, remote computer acquires license plate number

2 remote computer stores license plate in local memory

3 remote computer waits until central computer signals "ready for data" TRUE (meanwhile steps 1 and 2 can continue and new license plate numbers can stack up in local memory)

4 remote computer writes license plate number to its output port and signals "data available" TRUE

5 central computer reads data, sets "ready for data" FALSE

6 remote computer sets "data available" FALSE

[4 points off for setting "data available" TRUE before the data have been asserted

A common mistake was to answer something like this:

1 car passes, remote computer acquires license plate number

2 remote computer stores license plate in local memory

3 remote computer signals "data available" TRUE (fundamental error here- "data available" should be set TRUE only **after** data are actually available to the receiver)

4 central computer detects "data available" TRUE and when ready sets "ready for data" TRUE

5 remote computer detects "ready for data" TRUE and asserts data on its input port

6 central computer reads data and sets "ready for data" FALSE

7 remote computer detects "ready for data" TRUE and sets "data available" FALSE

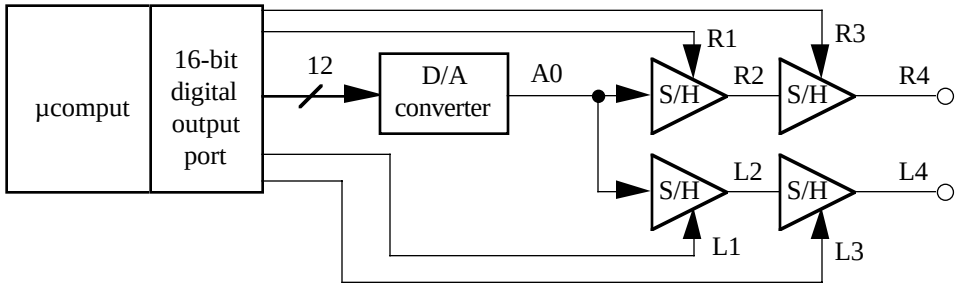
WHAT IS WRONG WITH THE ABOVE?

the central computer could perform steps 4 and 6 **before** step 5 is completed

Another common mistake was to think that the remote computer could "send" data to the central computer without any action on the part of the central computer. This appeared as the above with step 6 as

6 remote computer sends data to the central computer and sets "ready for data" FALSE

3a



R1, R3, L1, and L3 are digital control lines to S/H amps
[2 points off for using only two S/H amps, a design that does not update both left and right within << 1 µs]

3b

1 reset_clock; (this sets the clock to zero)

2 i=0

3 write to output port, sending val_right[i] to the D/A, SAMPLE to R1 and HOLD to R3, L1, and L3 (during this 1 µs step the D/A output A0 can convert, glitch, and settle down; this stable analog voltage appears at R2)

4 write to output port, sending val_right[i] to the D/A, HOLD to R1, R3, L1, and L3 (this holds the new analog value on R2)

5 write to output port, sending val_left[i] to the D/A, SAMPLE to L1 and HOLD to L3, R1, and R3, (during this 1 µs step the D/A output A0 can convert, glitch, and settle down; this stable analog voltage appears at L2)

6 write to output port, sending val_left[i] to the D/A, HOLD to R1, R3, L1, and L3 (this holds the new analog value on L2)

7 write to analog port, sending HOLD to L1 and R1, and SAMPLE to L3 and R3 (this simultaneously transfers R2 to R4 and L2 to L4)

8 write to analog port, sending HOLD to R1, R3, L1, and L3 (this holds the new analog values L2 and L4 until they are replaced the even newer values)

9 i = i + 1

10 val_time = time();

11 if val_time is less than i*25, go back to 10

12 go back to step 3 until all values are exhausted

[3 points off if the clock is not read and tested to perform each cycle in 25 μs]

3 c

A0	right[i+1]*	right[i+1]	left[i+1]*	left[i+1]	X	X
R1	SAMPLE	HOLD	HOLD	HOLD	HOLD	HOLD
R3	HOLD	HOLD	HOLD	HOLD	SAMPLE	HOLD
R2	right[i+1]*	right[i+1]	right[i+1]	right[i+1]	right[i+1]	right[i+1]
R4	right[i]	right[i]	right[i]	right[i]	right[i+1]	right[i+1]
L1	HOLD	HOLD	SAMPLE	HOLD	HOLD	HOLD
L3	HOLD	HOLD	HOLD	HOLD	SAMPLE	HOLD
L2	left[i]	left[i]	left[i+1]*	left[i+1]	left[i+1]	left[i+1]
L4	left[i]	left[i]	left[i]	left[i]	left[i+1]	left[i+1]

* analog values subject to glitches
X = don't care

3 c

1 2 hr is 7200 s or 228M updates at 40 kHz. This would require 1152 Mbytes of storage, which exceeds normal computer memory. Taking data from a large hard disk is possible, but block reads and double buffering would be needed to maintain the 40 kHz update rate.

2 the clock will reach 2^{32} μs in 4952 s so it is important to keep track of the clock overflow bit

[either of the above was taken for full credit]

[1 point off for stating that memory was a problem without estimating the actual amount of memory required]

Midterm #1 class statistics:

Problem	max	average	rms
1	20	18.0	2.0
2	20	16.7	2.7
3	60	47.4	9.4
total	100	82.1	10.8

Grade distribution:

Range	number	*approximate* letter grade
61-65	1	C–
66-70	2	C
71-75	0	C+
76-80	3	B
81-85	3	B+
86-90	4	A–
91-95	0	A
96-100	2	A+