CSE 466 Exam II 14 March 2008

Name: _____ ID#: _____

This is a take-home exam that should take you approximately one hour to complete, please don't spend more than two hours on it. There are 3 problems for a total of 100 points. The point value of each problem is indicated in the table below.

Complete the exam by editing this file directly (but please don't change the way the pages break). E-mail the resulting file (.doc or .pdf) to cse466-instructor@cs.washington.edu by 5PM on Wednesday, 19 March.

Good luck and have a great spring break. Thank you for a great time this quarter.

Problem	Max Score	Score
1	30	
2	30	
3	40	
TOTAL	100	

1. Definitions (30 points)

Define the following terms related to features of the iMote2 and the Linux operating system and provide a basic definition as well as an example of when each feature was used in your lab assignments.

a) Modules and device abstractions

<Your answer here – 1 or 2 sentences>

b) Packet buffers

<Your answer here – 1 or 2 sentences >

c) Describe an interaction you had between device modules and why you think it occurred.

<Your answer here – 1 or 2 sentences >

d) IRQ return values

<Your answer here – 1 or 2 sentences >

e) I^2C (TWI) read/write register commands

<Your answer here – 1 or 2 sentences >

f) Byte/word alignment in the packet format

<Your answer here – 1 or 2 sentences >

2. Soccer Enhancement (30 points)

There are many ways to enhance the soccer game that was done for the final project. Consider the enhancements below and discuss how you would go about implementing them, what parameters might be needed to be determined, and what problems do you foresee if packets are dropped.

a) Specialized players. A special goalie player on each team would be constrained to stay near the goal but would have the power to neutralize a merged player up to a size of 4. Special forward players would be able to move twice as fast as a goalie.

<Your answer here – 1 or 2 paragraphs>

b) Field worm holes. Worm holes on the field would allow a player to move rapidly from one end of the field to the other. Of course, the entrance and exits of the wormhole would also be visible to the other team.

<Your answer here – 1 or 2 paragraphs>

3. AirStick (40 points)

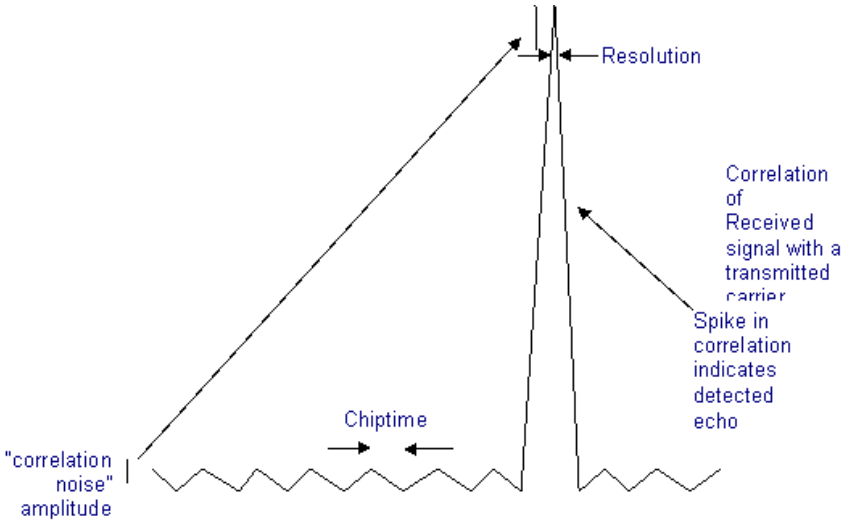
Light takes about 1ns (10^{-9} sec) to travel one foot. The waveforms we generated in our electric field sensing implementations had periods or chip-times on the order of 0.1 ms (corresponding to 10kHz frequency). Because the propagation time of the electric field was very small, the received signal was essentially synchronous with the transmitted signal: there was no noticeable phase shift or time delay between the received and transmitted signals.

Another common use of Pseudo-Noise signals (like those we generated with the LFSR) is to measure time delays in situations where time delays are meaningful. Examples include GPS, sonar range-finding, and laser range-finding. In these applications, Pseudo-Noise signals have a very clear advantage over periodic signals: the autocorrelation of a periodic signal has periodic maxima, so a peak in the correlation could indicate a reflection at many different distances. The autocorrelation of an LFSR signal has just one maximum, which identifies a reflection unambiguously.

Sound takes about 1ms (10^{-3} sec) to travel one foot. Suppose you are given a robot with a "dual" sonar rangefinder consisting of 2 ultrasound transmitters and 1 ultrasound receiver. Imagine that the transmitters are pointed in 2 slightly different directions (slightly to the left and right of the robot's heading, say). In this problem you will write code for CDMA spread-spectrum sonar range-finding. Assume you are given a 255-element array called LFSR with the values of a 255-element Maximum Length LFSR filled in (you can assume that the entries in the array are +1 or -1…if you assume differently, just specify your assumption). You can generate all the necessary LFSR sequences (for both TX and RCV) by indexing into this array.

Ignore attenuation effects. Just assume that you get equally strong echoes no matter what the distance. Also, assume that the ADC read operation is the only operation that takes any time.

a) Write code that generates the appropriate TX signals, collects (and buffers) the receive samples, does rolling correlations to detect echoes, and at the end returns 2 time delays, for the largest echo from each of the 2 TX signals. This code will be very different from your previous correlation code. To make the coding simpler, just transmit the entire Pseudo-Noise burst first, before starting to listen. When listening, you will have to maintain a buffer containing the last 255 samples you have seen. You will need to correlate this vector of samples with each of the 2 LFSR-generated carrier vectors. When there is a maximum in the correlation then that corresponds to a detected echo. Do not concern yourself with multiple echoes. Just return the time of the largest detected echo. So, for each of the 2 TXs, you will need one variable for the maximum correlation value seen on that channel so far, and one variable for the time at which the maximum correlation on that channel happened. Skeleton code is provided on the next page. Fill in the requested areas. Assume that main starts by calling tx(); and then correlate().



```
// Given: lfsr[i] for 0 <= i <= 254 is a ML LFSR sequence
// values of lfsr[i] are either +1 or -1;

int inc_headortail(int ht) { // increments & wraps head or tail pointer
  ht++;
  if (ht>254) {
    ht= 0;
  }
  return(ht);
}

int gai(int index, int h) { // gai == get_abs_index
  int abs_index;
  abs_index = (h+index) % 255;
  return (abs_index);
}

int lf1_h = 0;       // lfsr 1 head
int lf1_t = 254;     // lfsr 1 tail
int lf2_h = 10;      // Choosing lfsr 2 to be offset from 1 by 10
int lf2_t = gai(10, 254); // (10+254) % 255;

void tx() {
int i;
  for(i=0; i<255; i++) {
    output_PIN1(lfsr(gai(i,lf1_h)));
    output_PIN2(lfsr(gai(i,lf2_h)));
    READ_ADC; // Just for timing
  }
}

void correlate() {
int lf1_corr = 0;     // current correlation
int lf1_max_corr = 0; // max observed correlation
int lf1_peak_time = 0;// time of max observed correlation

int lf2_corr = 0;
int lf2_max_corr = 0;
int lf2_peak_time = 0;

int buf_h = 0;   // buf_h points to oldest sample
int buf_t = 254; // buf_t points to newest sample

int buf[255];
int i;
int t;
int T_MAX;

  // First fill buffer
  for (i=0; i<255; i++) {
    buf[i] = READ_ADC;
  }

  T_MAX = ___________; // How long are we going to wait for an echo?
```

```
  // Now start rolling correlation
  for (t=0; t<T_MAX; t++) {


<Your code goes  here – pseudo-code is fine.>
<It should not take more than this page.>



  } // t

  printf ("Chan 1: echo strength %d at time
                                        %d",lf1_max_corr,lf1_peak_time);
  printf ("Chan 2: echo strength %d at time
                                        %d",lf2_max_corr,lf2_peak_time);
}
```

b) Since you are transmitting first, and then listening, what is the closest object you can properly detect (by properly, we mean that you can hear the whole Pseudo-Noise sequence from). Assume 1 ft / ms speed of sound, length 255 LFSR, and chip time 0.1ms. What if the chip time was 0.01ms? What could you do to your code to give a reasonable minimum distance even with a 0.1ms chip time?

<Your answer here – 1 or 2 paragraphs>

c) What is the maximum range your sensor will detect an echo (this is determined by how long your code is willing to wait for an echo)?

<Your answer here – 1 paragraph>