The Raven Deconstructed

Background:

The Raven Deconstructed is based on the concept of Cellular Automata:

http://en.wikipedia.org/wiki/Cellular_automaton

and on the collective behavior of decentralized, self-organized systems:

http://en.wikipedia.org/wiki/Swarm_intelligence

An example of this is the famous BOIDS, which is a visual simulation of a flying flock of birds

The variant of using separate physical agents derives from field of Swarm Robotics:

http://www.Swarm-robotics.org/

We will appropriate the term "poetry slam", or "slam", to refer to our collective behavior in this project.

Description:

A Slam is a cloud of sound events of related phrases of a poem, in this case "The Raven" by Edgar Allen Poe, as read by James Earl Jones. The choices and the phrases derive from the execution of a set of common rules, without the intervention of any central control. Therefore, the resultant effect falls under the category of *Emergent Behavior*.

Slam sound-file phrases consist of several types:

Poem Phrases: 32 phrases from the poem, classed as complete thought (t), subject cluster (s), or verb cluster (v)

Raven Sounds: three distinct raven vocal sounds

Storm sounds: Thunder and rain sounds, two files

User Interface:

Each pair of partners will implement a GUI interface using the LCD and the rocker switch. This interface will allow for:

1. Manual Start/Stop. This will start the process. A global command packet can do this, as well.
2. Display: PhraseNum, Transmit Signal Strength, Avg. Received Signal Strength, other parameters as needed.

Each set of partners will implement their own version of a "Cooperative compositional agent" using the specifications listed in this document. Your compiled code will run on both you and your partner's "agents" for the Slams demonstration **on the final exam day, 10:30 AM, on Thursday, March 19th in the atrium**. Remember you need to qualify your "agent" before it can participate in the Slams demonstration. If for some reason you are unable to qualify your "agent", an alternative "agent" program will be provided so that you may receive your participation points for the demonstration.

Hints:

You should use enum types for default values and constants to simplify future modifications.

Try to minimize the number of divides and mod used in your code. Calculate the needed values only once and store the results. A memory read is a lot faster than a divide and/or mod.

NOTE: Not all of the agent implementations need to be the same. The implementations must only meet the specifications outlined. Differences in agent behavior will not be penalized as long as the behavior is within the specifications contained in this document. In fact, it is encouraged that each group's implementation be slightly different as long as they meet the specifications.

Addressing

You will be assigned a node ID that should uniquely identify your agent. This node ID is your agent's address on the radio network.

There are two special node IDs/addresses:

• 0x00 is the master controller
• 0xFF is the broadcast address

Your agent should ignore packets that weren't sent to its address or the broadcast address.

Rules governing phrase choice

```
Phrases are categorized as:
      a complete thought
      A subject cluster (noun)
      A verb cluster

This leads to choice algorithm:
      thought >> another thought or subject
      Subject >> verb only
      Verb >> thought or subject

Pick randomly:
x = (rand()% silence) + RSSIfactor
// note: RSSIfactor scaled from RSSI closer is higher probability
y = rand() % probability

if(x!=0) {
      don't play a phrase
      go to  WAIT_STATE

} else {

      choose phrase[y] from appropriate list
      try to send phrase packet
      if fail, go to WAIT_STATE;
      if success, go to PLAY_STATE;
}
```

•Note that Variables in bold are global parameters.

Trigger Phrases

Some phrases trigger a sound response—
Procedure: delay phrase_len – **trigvar** >> **1** +(rand( ) % **trigvar)** , then Play trigger sound

Kinds of trigger sounds (shown in filelst.xls):

- R = Raven sound 1 or 2 (pick one)
- T = Thunder
- W = Rain
- L = 10-Lenore
- N = 21, 27-Nevermore

State machine:

Within a individual "agent", the state machine that you will implement has the following states:

IDLE_STATE
(state machine starts in this state)
	set LED to RED
	Always {
	Wait for commands from controller and execute commands that arrive.
	Go to WAIT_STATE if start button is pushed, or if stop_and_listen packet received from controller
	}

WAIT_STATE	**On entering state** {
	set LED to YELLOW
	}
	Always {
	Listen for incoming packets
	Execute commands from controller
	If PACKET RECEIVED {
	    If Startled {
	        send Startled message
	        set LED to PURPLE
	        Set play_phrase to address of soundfile
	        go to PLAY_STATE
	    }
	    If **Phrase Message** {
	      decide if to play a phrase (see rules)
	      If result is PLAY {
	        delay phrase_len – **trigvar >> 1** +(rand( ) % **trigvar)***
	        If Carrier clear {
	            transmit **Phrase Message**
	            set LED to GREEN
	            Set play_phrase to address of soundfile
	            go to PLAY_STATE
	        } else go to WAIT_STATE
	    }
	    If Trigger Phrase {
	      set LED to BLUE
	      delay phrase_len – **trigvar >> 1** +(rand( ) % **trigvar)***
	      Set play_phrase to address of soundfile
	      go to PLAY_STATE
	    }
	}

PLAY_STATE	Listen for incoming packets Execute commands from controller Play play_phrase, then go to WAIT_STATE

***During delay above**, you should listen for packets
if packet == command: stop delay,do command, and go to directed state.
if packet == startle: stop delay, send startle packet (if needed), play startle
if packet == phrase: record info in circular queue, otherwise ignore

Message packet types:

The type field in the packet is a number to specify the type of the packet and the format of the rest of the data that follows.

Set Global Parameter

- This packet is a request from the controller for the agent to change one of its global parameters.
- It has a type of 50.
- uint8_t var The ID of the variable/parameter to alter
- int16_t value The new value for the parameter

Listen

- This packet from the controller tells the agent to stop any current sound and go to WAIT_STATE.
- If an agent is currently in IDLE_STATE and it receives a Listen packet, it should go to WAIT_STATE.
- It has a type of 51.
- It has no payload.

Do sound

- This packet from the controller tells the agent to stop any activity currently in progress and execute a particular sound, possibly send the phrase_message.
- It has a type of 52.
- uint8_t sound_num The number of the sound to execute.
- Uint8 send_start Boolean if true and packet not broadcast packet, send phrase_message packet upon play

Stop

- This packet from the controller tells the agent to stop any current sound and go to IDLE_STATE.
- It has a type of 53.
- It has no payload.

Phrase Message

- This is the main packet type sent out by your agent with information about itself, including the current phrase.
- It has a type of 42.
- uint8_t seq_num Sequence number. Start at 0 and increment on each packet sent.
- uint8_t phrase_num The current phrase number about to be played.
- uint8_t top_num The ID of the agent with the highest received signal strength. (from receive queue)
- uint8_t top_strength The received signal strength of the agent with the highest received signal strength. (from receive queue)
- uint8_t txpower The transmit power this node is currently using.

Startled Message (triggered by accelerometer)

- a message from some other agent. Stop what you are doing and play your startled sound. Then resend the startled message. Decrement the HopCount when you send the message. On reception, If (HopCount > 0) process message, else ignore message
- It has a type of 54.
- uint8_t transmittingNodeNum local # of startled node
- uint8_t hopCount Number of hops remaining.
- uint16_t startledSeqNum A randomly generated number that is stored to check to see if you have been startled by this startle packet previously

Receive queue

While listening, we collect the information we hear (#42 type packets) in an 8-entry circular FIFO queue.
Each queue entry looks as follows:

```
u_int16_t TransmittingNodeNum
u_int16_t PhraseNum
u_int16_t RSSI_strength
u_int16_t TxmtSigStrength
```

Each entry writes over the oldest entry in the queue.

Global Parameters

The following is a list of the parameters that the master controller should be able to set on your agent. Note that this list will probably change (particularly, new parameters may be added and the default values may change) as the project progresses.

Number	Name	Default Value
0	trigvar	100 msec
4	probability	10
5	silence	10
6	hop_cnt	3
7	txpower	20
8	LED color	0