SWARMS

Background:

Swarms is based on the AI concept of Swarm Intelligence:

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. It is characterized by simple rules:

- **separation**: steer to avoid crowding local flockmates
- **alignment**: steer towards the average heading of local flockmates
- **cohesion**: steer to move toward the average position of local flockmates

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

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

Description:

A Swarm is a cloud of sound events of related character stretching of a time period of 10s to 1000s of seconds. The choices and characteristics of the sounds 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.

Swarms consist of several types:

Swarms:

8 PD patches, 1 swarmless (silent) and 7 distinct patches, arranged in a continuous timbral space, **TBD**

User Interface:

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

1. Set Zone 1-6. Zone is a value that we can incorporate into the above choice algorithms.
2. Manual Start/Stop. This will start the process. A global command packet can do this, as well.
3. Display: SwarmNum, 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 Swarms demonstration **during the final class time, 12:30 PM, on Friday, Dec. 7th in the atrium**. Remember you need to qualify your “agent” before it can participate in the Swarms 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.

Implementation Details:

Addressing:

TBD

Swarms Algorithm:

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

```
A) SWARMLESS_WAIT STATE (Swarmless while waiting)

   When entering WAIT STATE {
     set Tri-Color LED to RED;
     start message timer to random time between [minWait, maxWait] seconds;
   }
   Listen for receive packets;
   IF(AdjustGlobals){
      Do so;
   }
   IF(Command Packet) {
      Perform command. Change LED;
      Stay in WAIT STATE;
   }
```

	IF (Timer Runout) {
	  Restart message timer timer to random time between [minWait, maxWait] seconds;
	  Evaluate rules;
	  Send Swarm_message;
	  Do result of evaluation;
	}

B)	Join_Swarm STATE Initialize for Swarm type and start it Go to SWARM STATE

C)	SWARM STATE
	Listen for receive packets;
	IF(*Command Packet*) {
	    Perform command.
	    Re-enter SWARM STATE
	}
	Restart message timer timer to random time between [minWait, maxWait] seconds;
	  Evaluate rules;
	  Send Swarm_message;
	  Do result of evaluation;

D)	END_SWARM STATE
	
	Gracefully end the Swarm you're doing
	Set listen timer for random time between [minWait, maxWait] seconds
	
	Change color of the Tri-Color LED.
	
	Continue to listen and collect information about what the neighbors are doing.
	
	IF(*Command Packet*) {
	    Perform command.
	}
	
	Go to SWARMLESS_WAIT STATE

Message types:

There are 4 types of Message packets your program must handle; they are as follows:

AM #	Flock Message / Packet

50	AdjustGlobals - A message from Node 0 containing global parameters for all birdies.
	
	`u_int16_t   Node0`
	`u_int16_t   Repetition          default 3`

	u_int16_t minWait default 3000 millisec. u_int16_t maxWait default 6000 millisec. u_int16_t Threshold default 600 u_int16_t minThreshold default 100 u_int16_t Probability default 10 u_int16_t Silence default 10 u_int16_t TransmitPower default 1
51	StopandListen - A message from Node 0 telling you to stop and listen. u_int16_t Node0 On receipt, go to SWARMLESS_WAIT STATE
42	**Swarm_message - The "I'm swarming" message; a message from some other agent indicating what swarm it is performing.** **You also send this packet each 3-6 seconds.** **u_int16_t TransmittingNodeNum local # of originating node** **u_int16_t SequenceNum start at 1, increment each time you send this packet** **u_int16_t swarmNum Swarm# that was performed** **u_int16_t swarmWeight usually same as Weight$_{max}$** **u_int16_t Weight$_{max}$swarmNum** **u_int16_t Weight$_{max}$** **u_int16_t Topswarm2Num** **u_int16_t Topswarm2Weight runner-up weight** **u_int16_t Weight$_{min}$swarmNum** **u_int16_t Weight$_{min}$** **u_int16_t TopNodeNum Strongest node you've heard** **u_int16_t TopNodeStrength Highest TOS_msg_strength** **u_int16_t Zone** **u_int16_t TxmtSigStrength**
52	DoSwarm - A message from Node 0 telling you to perform swarm N immediately. u_int16_t Node0 Go to Join_Swarm STATE u_int16_t SwarmN And do swarm N

While listening, we collect the information we hear (AM #42 type packets) in a 64-entry circular FIFO queue.
Each queue entry looks as follows:

u_int16_tTransmittingNodeNum
u_int16_tswarmNum
u_int16_tTOS_msg_strength
u_int16_tZone
u_int16_t**TxmtSigStrength**

Each entry writes over the oldest entry in the queue.

Evaluate rules

The algorithm for deciding the swarm to perform is as follows:

```
For all entries in our circular FIFO queue{
   // calculate $weight_i$ for swarmNum i == 0 to 7
   $Weight_i$ = sum of TOS_msg_strength in circular FIFO queue for each swarmNum
}

Find  $Weight_{max}$ == Largest Weight; $Weight_{max}$swarmNum is swarm with $Weight_{max}$

Find  $Weight_{min}$ == Smallest non-zero Weight; $Weight_{min}$swarmNum is swarm with $Weight_{min}$

x = rand() % Probability
y = rand() % Silence

if(x == 0 )
   SwarmNum = $Weight_{min}$swarmNum

else if(y == 0)
     Don't do a swarm-- go to SWARMLESS_WAIT STATE.

else{
   if($Weight_{max}$ < minThreshold) ||
    (($Weight_{max}$ > Threshold) && (You have already done $Weight_{max}$swarmNum more than Repetition times)) )
     SwarmNum = random swarm not among the last three swarms you've done more than Repetition times
   else
     SwarmNum = $Weight_{max}$swarmNum
}
```

Notes:

The threshold and Repetition count are meant to ensure that swarms are allowed to propagate through the performance space, but then die off after a while.
The repetition allows a strong swarm to propagate to a large number of nodes, but then once a swarm has played for a while, it should die off. This growth and die-off is accomplished by limiting the number of repetitions once the threshold is reached.

Zone is used by specific swarms to pick the average center frequency, similar to the use of the barometer to determine the floor.