# CSE 475 Creature Spec

Ryan Rowe, Austin Beaulieu

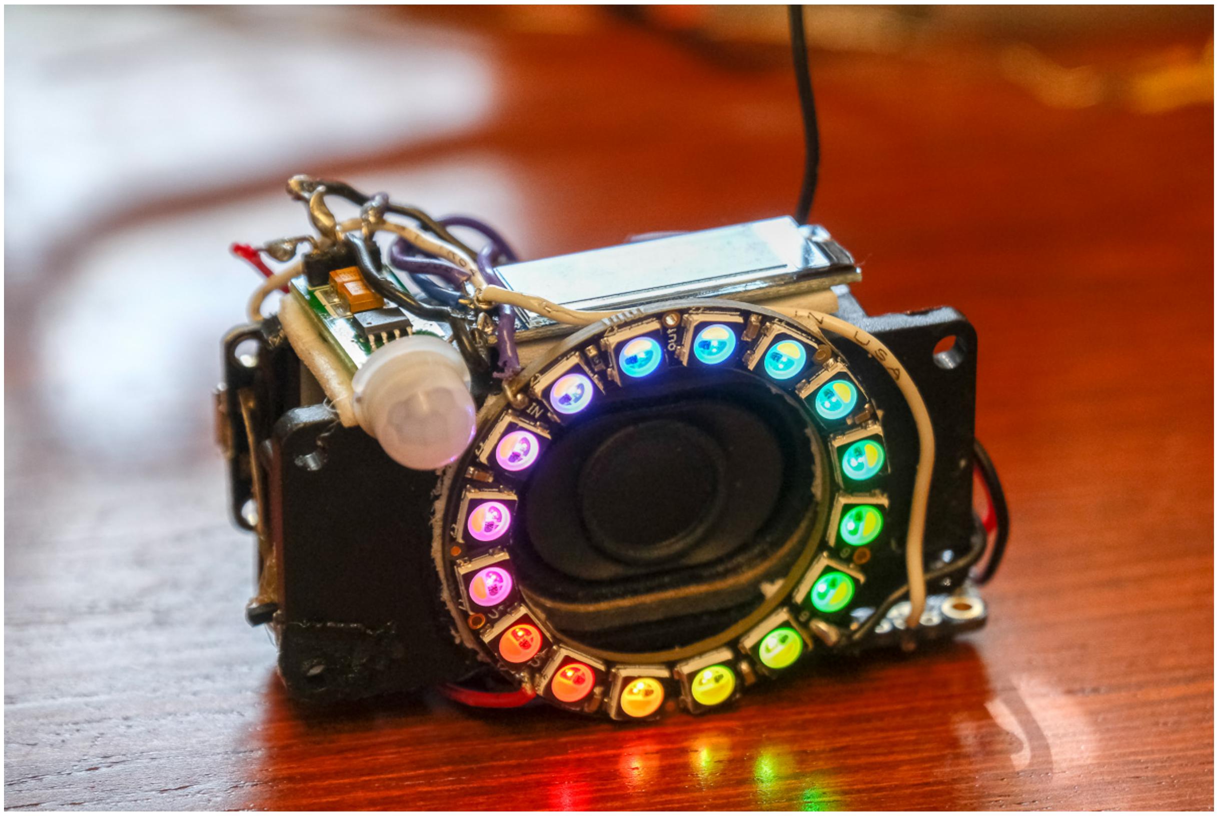
Rev. 2018-11-11 23:24:06

## 1 Summary

In this capstone we will function as a single mega-design team who have been commissioned to design and program a multi-nodal mesh radio-controlled set of “Creatures” who live in a large public space such as the Allen Center Atrium. The “Creatures” make sound, emit multi-colored light, sense motion, sense light levels and noise levels, and communicate to each other through 900MHz packet radio transmissions.

All “Creatures” obey a common set of rules of behavior, which encourages cooperation between “Creatures” and cumulatively produce emergent behavior of the group. This effect should be pleasing and relaxing, much more beneficial and entertaining than the usual background music.

We will explore such technologies as mesh networking, interrupt-driven multi-threaded processing, power-saving techniques, and rule-making to achieve leaderless emergent behavior. We will apply non-technical skills such as selecting colors and patterns and sounds for their expressive, evocative, and pleasing aesthetic effects.



# 2 The Creature Algorithm

## 2.1 Initialization

When each creature turns on, it must go through initialization:

1. Wait for communication with Controller
2. Receive & set data from AdjustGlobals packet
3. Initialize current state randomly from the list of standard states

## 2.2 Loop

For every `RepeatCycle` amount of time, run the following process:

1. If the controller sends PlaySound(N) or PlayEffect(N), play that sound and/or effect, else play current state
2. Start the radio and set listen timer for rand(`minListen`, `maxListen`) milliseconds for other creatures and/or the controller
3. Calculate the transition to the next state
4. Set a timer for `minListen` for sending the new state to other creatures when timer fired

## 2.3 Bookkeeping

Each creature needs to maintain values pertaining to the global state of the creature network and to the internal state of the creature itself. This section outlines these values.

### 2.3.1 Global Values

Creatures will receive state transition packets from other creatures. Each should maintain a list of the states other creatures are in and update this as it changes. We refer to this as the `StateList`. If a creature has not contacted another, they should assume it to be in the wait state (`0x00`) internally. The creature should also maintain a running average of RSSI from these packets. This average should serve as a reasonably accurate heuristic for the distances from other creatures. We refer to this as `DistanceList`.

### 2.3.2 Local Values

Several values must be maintained by the program pertinent to the creature itself. These values are used for computing state transitions & startles.

| **Variable** | **Description** |
|---|---|
| `uint32_t LAST_STARTLE` | Time in milliseconds since last startle |
| `uint8_t LAST_STARTLE_ID` | The last startle ID received |
| `uint8_t STARTLE_THRESHOLD` | Current numeric value of the startle threshold |
| `State CURRENT_STATE` | The current state object being maintained |
| `uint8_t REPEAT_STATE` | How many more state repetitions before calculating the next state |

# 3 Packet Protocol

Packets are sent and received in the following format. The first byte is the packet identifier, followed by the originating address (either $2 \times \text{kit}$ or $2 \times \text{kit} - 1$) and destination address. To send a packet to all creatures, use the broadcast address `0xFF`.

| 0 1 2 3 4 5 6 7 | 8 9 10 11 12 13 14 15 | 16 17 18 19 20 21 22 23 | 24 |
|---|---|---|---|
| `uint8_t pid` | `uint8_t src_addr` | `uint8_t destAddr` | `uint8_t payload[sizeof(packet) - 3]` |

All payload contents are listed by their byte offset inside the payload array.

## 3.1 Broadcast Packets

These are all packets that are broadcast by the controller to your creature. You should verify that `src_addr == 0` and `dst_addr` is either your creature's address or the broadcast address.

### PID `0x00` Set Globals

Updates a series of creature variables. Only valid if `src_addr == 0`.

| Byte | Variable | Default | Description |
|---|---|---|---|
| 0 | `uint16_t TX_POWER` | 1 | Radio transmit power |
| 2 | `uint8_t STARTLE_RAND_MIN` | 100 | Minimum bound when generating startle strength |
| 3 | `uint8_t STARTLE_RAND_MAX` | 200 | Maximum bound when generating startle strength |
| 4 | `uint8_t STARTLE_MAX` | 255 | Maximum startle strength |
| 5 | `uint8_t STARTLE_THRESHOLD` | 150 | Trigger threshold for being startled |
| 6 | `uint8_t STARTLE_DECAY` | 30 | Distance decay for startle packets |
| 7 | `uint8_t NUM_CREATURES` | 40 | Number of creatures in the network |
| 8 | `float STARTLE_THRESHOLD_DECAY` | 0.001 | Decay of threshold per millisecond |
| 12 | `uint16_t CYCLE_TIME` | 5000 | Amount of time between each play state |

### PID `0x01` Stop

Enter WAIT state. Only valid if `src_addr == 0`.

### PID `0x02` Start

Exit the WAIT state. Only valid if the creature is currently WAITing `src_addr == 0`. Note: the mode `0x0000` would indicate starting in state WAIT, but since this command is only accepted while we're already WAITing, it indicates starting randomly.

| Byte | Variable | Description |
|---|---|---|
| 0 | `uint8_t mode` | `0x00` Start in state specified<br>`0x01` Continue |
| 1 | `uint8_t stateId` | State to start in, unless continuing from last state. WAIT (`0x00`) indicates starting from a random sta |

### PID `0x03` Play Sound

Only valid if `src_addr == 0`. Tells a creature to temporarily stop its current process and play this specific sound for X cycles.

| Byte | Variable | Description |
|---|---|---|
| 0 | `uint8_t sound` | Sound number to play |

### PID `0x04` Play Effect

Only valid if `src_addr == 0`. Tells a creature to temporarily stop its current process and play this specific effect for X cycles.

| Byte | Variable | Description |
|---|---|---|
| 0 | `uint8_t effect` | Effect number to play |

#### PID `0x05` Broadcast States

Only valid if `src_addr == 0`. The controller here sends a list of the last observed state for all $N$ creatures. State identifiers are those described by §6.

| Byte | Variable | Description |
|---|---|---|
| 0 | `uint8_t state_1` | Creature 1's state ID |
| ... | ... | ... |
| $N-1$ | `uint8_t state_n` | Creature $N$'s state ID |

### 3.2 Creature Packets

These are all packets that are sent by your creature either to the controller or to other creatures. When you send these packets, be sure to set `src_addr` to your creature's address and set `dst_addr` correctly depending on the packet type and purpose.

#### PID `0x06` Startle

Sent when a creature transitions into a startled state. This should be packet should be sent to all creatures (a.k.a. broadcast address) with `destAddr == 0xFF`.

| Byte | Variable | Description |
|---|---|---|
| 0 | `uint8_t strength` | Startle strength |
| 1 | `uint8_t startle_id` | Startle identifier |

#### PID `0x07` Send State

Relays a creature's current state to the central controller. This packet should only be sent with `destAddr == 0` and should be sent whenever a creature transitions from one state to another.

| Byte | Variable | Description |
|---|---|---|
| 0 | `uint8_t old_state` | Previous state identifier as described in §6 |
| 1 | `uint8_t state` | New state identifier |

#### PID `0x08` Sound Played

TODO: Should we keep this?

#### PID `0x09` Effect Played

TODO: Should we keep this?

## 4 Global Values

TODO: Write descriptions & usages of globals

## 5 Startles

A creature can enter into the STARTLE state from one of two ways: a rising edge from the PIR sensor (0 to 1) or from a startle passed on from another creature. Each creature has an internal startle threshold that lowered proportional to the time since the last startle. When a startle occurs, a strength is randomly generated which decides how 'large' the startle is. This value is used to determine if the creature is actually startled by comparing it to the current startle threshold. If it does, it enters into a startle state, and then passes on this startle. This value could be used to determine how great a response is–perhaps by volume or length.
`STARLE_FACTOR` = Local decay of startle for each state
`STARTLE_THRESHOLD_DECAY` = Global decay of startle

PIR is called when the PIR sensor is activated.

```
function PIR
    if is not duplicate (This is handled for you automatically) then
        strength ← $\min\left\{255, \text{rand}(\texttt{STARTLE\_MIN}, \texttt{STARTLE\_MAX}) \times \left(1 - \frac{255}{\texttt{STARTLE\_TREHSOLD}} \cdot 0.5 + 1\right)\right\}$

        STARTLE(strength, rand(0, 255))
    end if
end function
```

RXSTARTLE is called when a startle packet is received from another creature. It decays the startle proportional to the heuristic distance. Note: We use the sigmoid function here: $\sigma(x) = 1/(1 + e^{-x})$

```
function RXSTARTLE(RSSI, strength, id)
    decay ← $\sigma\left(\frac{\texttt{STARTLE\_DECAY}-\text{RSSI}}{\texttt{STARTLE\_DECAY}}\right) \times \texttt{STARLE\_FACTOR}$
    STARTLE(decay × strength, id)
end function
```

STARTLE is called when this creature has successfully been startled, either by the PIR sensor or by receiving a startle packet.

```
last.id ← the id of the last startle to pass the threshold
last.time ← the time (in millis) since the last startle.
function STARTLE(strength, id)
    if id ≠ last then
        time ← millis()
        threshold ← threshold − threshold × (time − last.time) × STARTLE_THRESHOLD_DECAY × STARLE_FACTOR
        if strength ≥ state.threshold() then
            Transition into Startle State
            txStartlePacket(strength, id)
            last.id ← id
        end if
        last.time ← time
    end if
end function
```

# 6 States

To offer examples and build a potential scene for our network of creatures, lets imagine the scenery of a rain forest and how these states could be used to bring this scene to life in a pleasing way.

## 6.1 Pause And Listen State

When a creature receives a `Stop` packet from the controller, it enters into this state and does not perform any action. It stays in this state until it receives a `Start` packet from the controller and continues the general process of the creature algorithm. This is identified as state `0x00`

## 6.2 Startle State

When a creature crosses the `STARTLE_THRESHOLD` it will enter into this state, perform the appropriate action, then randomly decide a new start state and continue where it left off. The startle action should be identifiable from active states. The sounds should be very loud and active, and the visuals should convey something similar. To build onto the idea of the rain forest, an example of a startle could be the roar of a large cat or the thunderous strike of lightning (If several creatures are in a rain state) accompanied by a large flash of light. One could define the effect of the startle state as dependant on the state of other creatures in the network (i.e. thunder if several are in the ambient rain state, else a cacophony of animal sounds. This is identified as state `0xFF`

## 6.3 Normal States

### 6.3.1 Ambient

Each implementation must include 3 "ambient" states. These should have a high likelihood of being entered, but this likelihood should decrease linearly to the number of creatures in a "prominent" state. These states should convey calm and pleasing sounds that blend into the rest of the scene and with other creatures displaying an ambient state. The sounds should be subtle and the visual effects should be regular. An example of this could be raindrop sounds accompanied with slowly twinkling blueish lights in a rain forest scene, the sounds of small critters (like bugs and quieter birds), or the flow of water. One could even define the state transitions such that rain is unlikely to enter into, but if some creatures enter into it, it becomes increasingly more likely, creating a rainstorm!
These states should be identified by an odd hex value: `0x01`, `0x03`, ..., etc.

### 6.3.2 Active

Each implementation must include 3 "active" states. These should have a normal likelihood of being entered, and this likelihood should increase linearly to the number of creatures in active and prominent states. These states should convey that action is occurring within the scene. To further build on this idea of a rain forest scene, an example of this could be the sounds of leaves rustling or of louder birds or other animals accompanied with more active visual effects, possibly in sync with the sounds that are being output.
These states should be identified by an even hex value: `0x02`, `0x04`, ..., etc.

## 7 State Transitions

All creatures will have a predetermined number of states that they can enter into, with reasonably similar intention for the state itself. This does not mean that they will be similar in how they are implemented, but rather that certain states, like a startle state, convey a similar tone as that of other creatures. Each state will set `MaxRepeats`, which defines how many times a creature will repeat a certain state before calculating the next state transition. When entering into a state the number of repeats is defined as $\text{rand}(1, \texttt{MaxRepeats})$.

Let $\mathbb{S}$ be the set of all states a creature can enter into
Let $d$ be the number of possible creature states
Let $N$ be the number of all creatures

All creatures will have a global value defined by GLOBAL_WEIGHTS, passed by from the central controller. In this we refer to it as $\mathbb{L}$

$\mathbb{L} \in \mathbb{R}^d$: The set of scalars used for determining the weight that other creatures of the same state have on this creature's likelihood of transitioning into that state. If a scalar $\mathbb{L}_n > 0$, then it directly scales with respect to the number of creatures in state $\mathbb{C}_n$, else if $\mathbb{L}_n < 0$, it negatively scales such that the more creatures there are in state $\mathbb{C}_n$, the less likely one is to enter into it.

Each creature will be able to enter into one of $\mathbb{S}$. To compute this, we will a few things:

1. $\mathbb{C}$ : The set representing the record of the current state of all other $N$ creatures.
2. $\mathbb{W}_i \in \mathbb{R}^d$: The set of relative weights for transitions into its next state based on this creature's current state (higher weights being more likely).
3. $\mathbb{O}_i \in \mathbb{R}^N$ : The distance from this creature to the $N$ other creatures
4. $\mathbb{D}_i \in \mathbb{R}^N$ : The inverse distances of the $N$ creatures to this creature derived from $\mathbb{O}$. If $\mathbb{O}_i \neq 0$ then $\mathbb{D}_i = -1/\mathbb{O}_i$, else $\mathbb{D}_i = 0$

The final weight for a transition from state $\mathbb{S}_i$ to state $\mathbb{S}_j$ is calculated as follows:

$$\widetilde{\mathbb{L}}_j = \mathbb{L}_j \cdot \frac{N - \sum_{k=0}^{N} [\![\mathbb{C}_k = \mathbb{S}_j]\!]}{N}$$

$$\mathbb{P} = \left\{ \max(0, \mathbb{W}_{i,j} + \widetilde{\mathbb{L}}_j \cdot \textstyle\sum_{k=0}^{N} \mathbb{D}_{i,j} [\![\mathbb{C}_k = \mathbb{S}_j]\!]) \right\}_{j=1}^{N} \text{ for each creature } \mathbb{C}_i$$

$$\widetilde{\mathbb{P}} = \left\{ \textstyle\sum_{i=1}^{j} \mathbb{P}_i \right\}_{j=1}^{N}$$

Let $R = \text{rand}(0, \widetilde{\mathbb{P}}_N)$

```
1: for $i \in \{1, ..., N\}$ do
2:     if $R < \widetilde{\mathbb{P}}_i$ then
3:         Let $\mathbb{S}_i$ be the next state to transition to
4:     end if
5: end for
```