**Class Diagram Definitions:**

**Server –** Our Server will contain all the information needed for the feeder mechanism to work. This includes the Microcontroller IP Addresses, MAC Addresses, VLAN Switch identification information, ports and mesh network information. All the animal schedule information will also be stored on the server: Elephants on exhibit, feeds to be dispensed, time of the day, amounts to be dispensed. The Server will be working with the Coordinator, it will send it all the information

**GUI** – The GUI class will handle all graphical communications with the user. There will be a GUI for 3 types of users, Keepers, Managers and Admins. The GUI will retrieve information from the user such as the animals that will be in the exhibit, as well as display the feeding times for the current schedule.

**Feeder Allocator** – The Feeder Allocator will be responsible with generating the schedules for the feeds of the set time period. It will take as input the animals that will be in the exhibit during that time and use that to retrieve their data from the animal database. Once that is retrieved, the Feeder Allocator will use the Business (Biological) requirements to create a schedule for the system.

**Wifi Manager –** The wi-fi manager will handle the logic for socket-to-socket communication. This includes creating bi-directional sockets, connecting to other sockets to then send/receive data. The data sent is a list of schedules going from the Server to the Central Microcontroller.

**Coordinator –** The Coordinator will receive the necessary configurations for setting up the mesh network, and the daily schedules for each feeder. Using the connection information, the coordinator will set-up the mesh network. The coordinator will send the feeding schedule to its respective router/end-point. On a timed cadence, the coordinator will ping the routers/end-points to ensure connectivity.

**Endpoint-** The Endpoint has 2 main functionalities: Receiving its pertaining feeding schedule. And sending feeding status to the coordinator. Once the Endpoint receives a feeding schedule, it sends it to its Schedule Manager for it to be handled and stored internally in the dumb microcontroller. Once the dumb microcontroller has activated a feeder (successfully/unsuccessfully), the Endpoint sends that status to the coordinator.

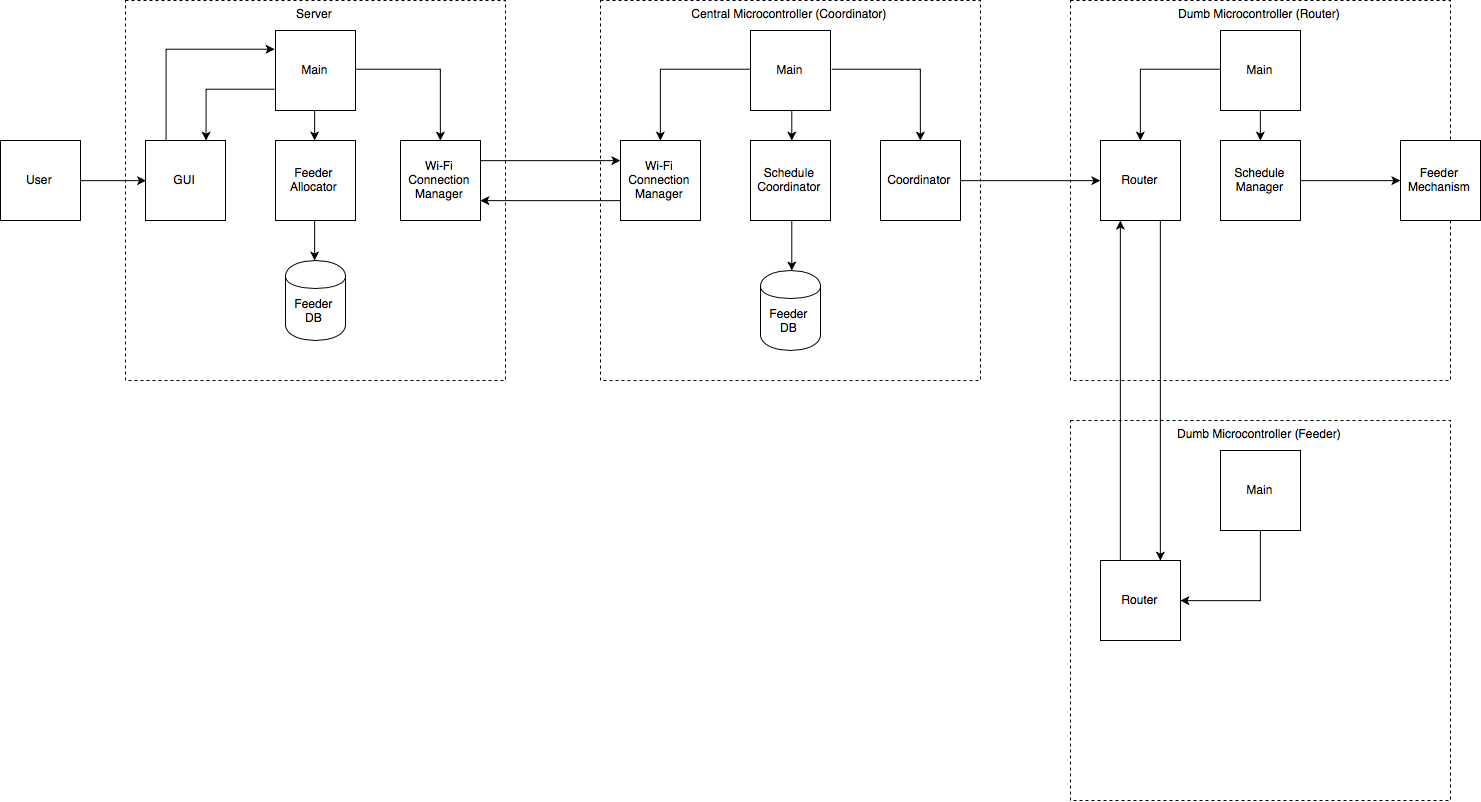
**Router –** The Router’s main functionality is to forward feeding schedules and statuses to the pertaining Endpoints in the mesh network.

**Schedule Coordinator (Coordinator)** – The schedule coordinator receives all schedules for the system. The coordinator stores the system schedule internally. The Class then Retrieves the schedules from the Schedule Coordinator and passes them to the Coordinator in order to send the pertaining schedule to its pertaining microcontroller (endpoint or router).

**Schedule Manager (Endpoint/Router)** – The schedule manager receives as input a schedule of feeds for its pertaining feeder. The schedule manager will store this schedule internally. When it is time to dispense a feed, the Schedule Manager will interface with the feeding mechanism to dispense a feed.

**Feeder Mechanism** – The feeder mechanism handles the logic for interfacing with the DC motor. The Feeder Mechanism takes an activation input from the schedule manager, when it is time to dispense the feed. When activated, the feeder mechanism turns on the DC motor for a set amount of time and turns the motor off when that time is exhausted. The feeder mechanism then communicates back to the schedule manager whether it was successful or not.

**Feeder Database (Server)** – The Feeder Database serves three purposes, storing the connection information for all feeders in the system, storing the scheduled feeds for the system and storing the biological information about the animals in the system.



**Server.Main Class:**

|  |
| --- |
| **Central Controller.Main** |
| - GUI  - Feeder Allocator  - Wi-Fi Connection Manager |
| create\_coordinator(configs)  create\_schedule\_coordinator(configs)  read\_schedule()  assign\_schedule()  send\_schedules() |

1. The central\_controller shall run a class: Main, that will control all the operations pertaining to the central microcontroller.
2. The central\_controller.main shall contain a reference to one and only one Coordinator object.
3. The central\_controller.main shall contain a reference to one and only one Schedule Coordinator object
4. The central\_controller.main shall signal the Coordinator object to begin to read input from the Server.
5. The central\_controller.main shall assign the schedule read from the Coordinator to the Schedule Coordinator.
6. The central\_controller.main shall instruct the send Coordinator object to send the pertaining schedule to its pertaining router, based on the information from the Schedule Coordinator.

**Central Controller.Main Class:**

|  |
| --- |
| **Central Controller.Main** |
| - Coordinator  - Schedule Coordinator  - Wi-fi Connection Manager |
| create\_coordinator(configs)  create\_schedule\_coordinator(configs)  read\_schedule()  assign\_schedule()  send\_schedules() |

1. The central\_controller shall run a class: Main, that will control all the operations pertaining to the central microcontroller.
2. The central\_controller.main shall contain a reference to one and only one Coordinator object.
3. The central\_controller.main shall contain a reference to one and only one Schedule Coordinator object
4. The central\_controller.main shall contain a reference to one and only one Wi-Fi connection manager object
5. The central\_controller.main shall signal the Coordinator object to begin to read input from the Server.
6. The central\_controller.main shall assign the schedule read from the Coordinator to the Schedule Coordinator.
7. The central\_controller.main shall instruct the send Coordinator object to send the pertaining schedule to its pertaining router, based on the information from the Schedule Coordinator.

**Central Controller.WiFi Manager Class:**

|  |
| --- |
| **Central Cotroller. Wi-fi Manager** |
| - IP Address  - Port  - Target Socket |
| write\_to\_socket()  read\_from\_socket() |

**Central Controller.Coordinator Class:**

|  |
| --- |
| **Central Cotroller.Coordinator** |
| - VLAN Config  - Xbee Config  - List<Routers> |
| - ~~send\_to\_server()~~  ~~- read\_from\_server()~~  - send\_to\_router()  - read\_from\_router() |

**Central Controller.Schedule Coordinator Class:**

|  |
| --- |
| **Central\_Cotroller.Schedule Coordinator** |
| - Feeder DB |
| - assign\_schedules()  - get\_feeders()  - get\_feeder() |

1. The Central\_Controller.Schedule Coordinator class shall connect to the Feeder Database inside the central controller.
2. The Central\_Controller.Schedule Coordinator shall receive a list of schedules as input.
3. Upon receiving a list of schedules, Central\_Controller.Schedule Coordinator shall assign the pertaining schedule to its pertaining feeder, by retrieving the feeder information from the Feeder Database.
4. Upon request, the Central\_Controller.Schedule Coordinator shall provide a list of all feeders with their pertaining connection and schedule information.
5. Upon request, the Central\_Controller.Schedule Coordinator shall provide a specified feeder with its pertaining connection and schedule information.

**Dumb Controller.Main Class:**

|  |
| --- |
| **Dumb Controller.Main** |
| - Router  - Schedule Manager |
| create\_router(configs)  create\_schedule\_manager(configs)  read\_schedule()  assign\_schedule()  forward\_schedules()  signal\_Router |

1. The dumb\_controller shall run a class: Main, that will control all the operations pertaining to the dumb microcontroller.
2. The dumb\_controller.main shall contain a reference to one and only one Router object.
3. The dumb\_controller.main shall contain a reference to one and only one Schedule Manager object
4. The dumb\_controller.main shall signal the Router object to begin to read input from the Coordinator object.
5. The dumb\_controller.main shall assign the schedule read from the Router to the Schedule Manager.
6. The dumb\_controller.main shall instruct the send Router object to send the pertaining schedule to its pertaining router, based on the information from the Schedule Coordinator.

**Dumb Controller.Router Class:**

|  |
| --- |
| **Dumb Controller.Router** |
| - Router  - ~~Schedule Coordinator~~  -Coordinator |
| ~~create\_coordinator(configs)~~  ~~create\_schedule\_coordinator(configs)~~  read\_schedule()  ~~assign\_schedule()~~  forward\_schedules()  send\_status() |

1. ~~The dumb\_controller.Router shall contain a reference to one and only one Schedule Coordinator object~~
2. The dumb\_controller.Router shall receive the feeding Schedule from the Coordinator
3. ~~The dumb\_controller.Router shall forward the feeding Schedule to the Main object.~~
4. The dumb\_controller.Router shall forward the feeding Schedule to the next Router
5. The dumb\_controller.Router shall receive the feeding status from the Main object.
6. The dumb\_controller.Router shall send the feeding status to the Coordinator object.

**Dumb Controller.ScheduleCoordinator Class:**

|  |
| --- |
| **Dumb Controller.ScheduleCoordinator** |
| ~~- Main~~  ~~- VLAN Config~~  ~~- Xbee Config~~  ~~- List<Routers>~~  - Feeder Mechanism |
| ~~create\_coordinator(configs)~~  ~~create\_schedule\_coordinator(configs)~~  ~~read\_schedule()~~  read\_completion\_status()  assign\_schedule()  ~~send\_schedules()~~  activate\_feeder()  store\_schedules() |

1. The dumb\_controller.ScheduleCoordinator shall contain a reference to one and only one Feeder Mechanism object.
2. ~~The dumb\_controller.ScheduleCoordinator shall receive the feeding schedule from the Main object~~
3. The dumb\_controller.ScheduleCoordinator shall store the Schedule internally
4. The dumb\_controller.ScheduleCoordinator shall ~~send the Schedule to~~ activate the Feeder Mechanism object at its pertaining time
5. The dumb\_controller.ScheduleCoordinator shall receive the completion status from the Feeder Mechanism

**Dumb Controller.FeederMechanism Class:**

|  |
| --- |
| **Dumb Controller.FeederMechanism** |
| - DC Motor  - ~~Schedule Manager~~ |
| ~~create\_coordinator(configs)~~  ~~create\_schedule\_coordinator(configs)~~  interface\_DCMotor()  signal\_DCMotor\_activation()  ~~send~~save\_completion\_status() |

1. The dumb\_controller.FeederMechanism shall interface with the DC Motor.
2. ~~The dumb\_controller.FeederMechanism shall contain a reference to one and only one Schedule Coordinator object.~~
3. ~~The dumb\_controller.FeederMechanism shall receive the schedule from the Schedule Manager.~~
4. The dumb\_controller.FeederMechanism shall signal the DC Motor object to activate and dispense the food to the animals.
5. The dumb\_controller.FeederMechanism shall ~~send the Schedule Manager object~~ save the completion status. (successful/unsuccessful)