CSE 4345 SEP : HW3

Premise

•A software system will be built to allow drones to autonomously herd cattle in farms. These drones can be remotely controlled by human operators.

•Explain how multiple architectural patterns can fit together to help build this kind of system

Explanation

To choose the best architecture, we must analyze what data flow and functionality the software has. First we can break down the objects we need to control with the software.

- Have two modes
  - Human Control
  - Auto Pilot : default mode
- Herd the cattles
  - For Operator Mode
    - Have camera control to see the cattles
    - Be able to perform action to herd the cattles (e.g., make sounds, shock cattles)
    - Be able to switch between operator mode and Auto mode
  - For Auto Mode
    - Be able to take pre-made instruction to perform herding actions at given condition
    - Camera control to be able to detect cattles and check for conditions on whether to perform herding actions.

  By Parts, it will be divided into the following.
  - Detector : Everything related to seeing / detecting the cattle and their action
  - Selector : Mode selection
  - Controller : Control logic for the software
  - Herder : Performance code to herd the cattle

Data Flow : Pipe Architecture

The basic data flow would be useful if we use Pipe style.

Detector	**->**	**Selector/Controller**	**->**	**Herder**
Collect Data		Choose Mode/Take Control		Perform

The detector will collect visual data and feed it to the selector. If you choose the auto mode, it will utilze that data to start up control logic and determine whether to enforce herding. If you choose operator mode, you will have the visual data and you will be able to input controls to perform herding.

Time Critical Task : Daemon

We don't won't the cattle running away due to delay in the system. This is a hard real-time system. For best performance on Automated system, we want the daemon listening to the detector for certain condition that is set. For instance, if one cattle strays from the herd and the camera finds it the detector will send signal to Daemon. Daemon will immediately make a process to send message to Herder, and go back to listening to the detector until the stray comes back. Once the stray comes back, detector will send the signal and Daemon will send stop signal to Herder.

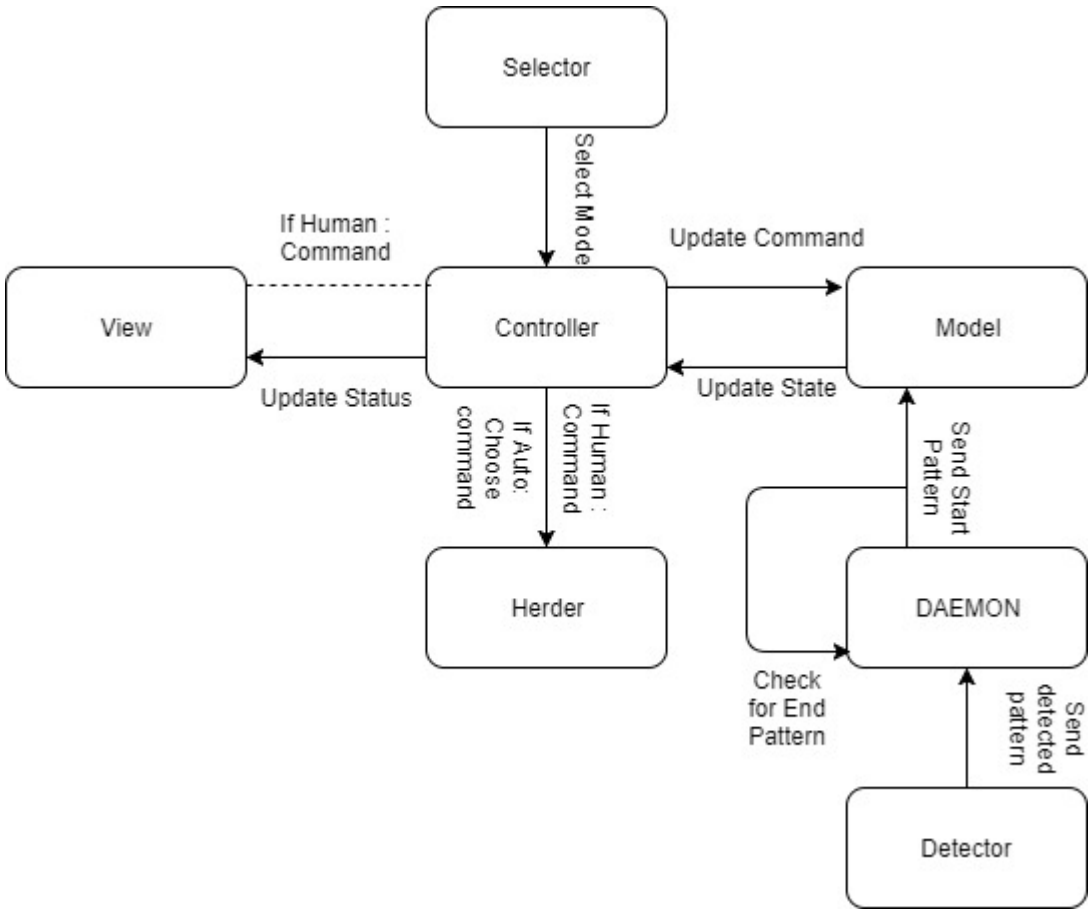
Final Architecture Integration : MVC

While having the data flow diagram and daemon, we want the MVC model for Operator Mode.

- View : Sends user actions (commands to control the drone, herding action) to Model. Receives state of the software from Model. This view will be
- Controller : Receive the update of commands from the view. Sends Model the commands for change of state. Receives update of state from Model. Sends View the update of state.
- Model : Sends the current state to the controller to display at the view. Receives update of commands from Controller to change state.

To avoid too long of a code in controller, we can divide the contoller's role up and out source them. While maintaining View <->Controller<->Model architecture, we will have division of labor from Controller.

The MVC architecture will be the part of the Selector in data flow. The model will get signal from the detector which contains the Daemon, and controller will receive choice from outside selector. With information being sent to views for display, the controller will also send the data to Herder. So if the auto pilot is selected, it will not take commands from views but only send out instructions to the Herder portion of the program. The diagram is below.



The detector interacts with the hardware (camera) to detect certain patterns. Once it sees a stray, or any state that requires herding it will send a signal to daemon to send current state (need_herding) to the model. The daemon will send the signal and go back to watching for end signal (no_need_herding) from detector. The model will update this state to the controller. Controller will take input from selector on which mode to operate on, with default mode being auto. In this mode, the controller will only send pass the visual state to views. Based on the state passed in from model, it will send herder commands to operate on. If the mode is selected to be human operator, it will receive command from View to Controller. Then the Controller will send the received command to the herder. If the end signal is sent from Daemon to Model, the Model will update the state to controller and it will kill all commands.