## Biased Economizer Sensor: Mixed Temperature

### Description

def description

return “When sensors drift and are not regularly calibrated, it causes a bias. Sensor readings often drift from their calibration with age, causing equipment control algorithms to produce outputs that deviate from their intended function. This fault is categorized as a fault that occur in the economizer system (sensor) during the operation stage. This fault measure is based on a physical model where certain parameter(s) is changed in EnergyPlus to mimic the faulted operation; thus simulates the biased economizer sensor (mixed air temperature) by modifying the SetpointManager object assigned to the economizer. The fault intensity (F) is defined as the biased temperature level (K). A positive number means that the sensor is reading a temperature higher than the true temperature."

end

### Modeler Description

def modeler\_description

return “Two user inputs are required and, based on these user inputs, the setpoint temperature at the mixed air temperature node will be replaced by the following equation, Tma\_setpoint,F, = Tma\_setpoint – F, where Tma\_setpoint,F is the mixed air temperature setpoint affected by the bias, Tma\_setpoint is the actual mixed air temperature setpoint, and F is the fault intensity. To use this Measure, choose the Controller:OutdoorAir object to be faulted. Set the level of temperature sensor bias that you want at the mixed air duct for the economizer during the simulation period. Positive value of F means sensor is reading higher value than the actual temperature. The algorithm checks if a real sensor exists in the mixed air chamber, and set up the bias at the sensor appropriately if it exists. For instance, SetpointManager:MixedAir does not model a real temperature sensor in the mixed air chamber, and will not be affected by this model."

end

### Measure Type

EnergyPlus Measure

**Taxonomy**

HVAC.HVAC Controls

### Arguments

def arguments(workspace)

args = OpenStudio::Ruleset::OSArgumentVector.new

#make choice arguments for economizers

controlleroutdoorairs = workspace.getObjectsByType("Controller:OutdoorAir".to\_IddObjectType)

chs = OpenStudio::StringVector.new

controlleroutdoorairs.each do |controlleroutdoorair|

chs << controlleroutdoorair.name.to\_s

end

econ\_choice = OpenStudio::Ruleset::OSArgument::makeChoiceArgument('econ\_choice', chs, true)

econ\_choice.setDisplayName("Choice of economizers.")

econ\_choice.setDefaultValue(chs[0].to\_s)

args << econ\_choice

#make a double argument for the temperature sensor bias

mix\_temp\_bias = OpenStudio::Ruleset::OSArgument::makeDoubleArgument("mix\_temp\_bias", false)

mix\_temp\_bias.setDisplayName("Enter the bias level of the mixed air temperature sensor. A positive number means that the sensor is reading a temperature higher than the true temperature. (K)")

mix\_temp\_bias.setDefaultValue(2) # default bias level at 2K

args << mix\_temp\_bias

return args

end

### Initial Condition

#Select economizer object that is being faulted.

runner.registerInitialCondition("Imposing Sensor Bias on "+econ\_choice+".")

### Final Condition

#Impose sensor bias on the economizer object.

runner.registerFinalCondition("Imposed Sensor Bias on "+econ\_choice+".")

### Not Applicable

#When fault measure is not applicable in the economizer model,

runner.registerAsNotApplicable("BiasedEconomizerSensorMixedT is not running for "+econ\_choice+" because of inapplicability. Skipping......")

### Warning

n/a

### Error

#When the node (mixed temperature) name is found in the nodelist instead of node,

runner.registerError("Nodelist is found instead of node. Exiting......")

#When selected economizer cannot be found in the model,

runner.registerError("Measure BiasedEconomizerSensorMixedT cannot find "+econ\_choice+". Exiting......")

### Information

* Works with,
  + SetpointManager:OutdoorAirReset
  + SetpointManager:SingleZone:Reheat
  + SetpointManager:SingleZone:Heating
  + SetpointManager:SingleZone:Cooling
  + SetpointManager:OutdoorAirPretreat
  + SetpointManager:MultiZone:Cooling:Average
  + SetpointManager:MultiZone:Heating:Average
  + SetpointManager:Warmest
  + SetpointManager:Coldest
  + SetpointManager:WarmestTemperatureFlow
  + SetpointManager:FollowOutdoorAirTemperature
  + SetpointManager:FollowGroundTemperature
  + SetpointManager:FollowSystemNodeTemperature
  + SetpointManager:SingleZone:OneStageCooling
  + SetpointManager:SingleZoneOneStageHeating SetpointManager:Scheduled
  + SetpointManager:Scheduled:DualSetpoint
  + SetpointManager:ReturnAirBypassFlow
  + SetpointManager:MixedAir
* Leakage at the downstream of zone terminal unit.

### Code Outline

* Define arguments (economizer where fault occurs, fault level in constant value).
* Find the economizer where the fault occurs.
* Find the node name of the mixed air chamber.
* Verify the type of SetpointManager object used at the mixed air chamber.
* Impose sensor bias according to the type of SetpointManager Object as shown below.

|  |  |
| --- | --- |
| SetpointManager:OutdoorAirReset  SetpointManager:SingleZone:Reheat  SetpointManager:SingleZone:Heating  SetpointManager:SingleZone:Cooling  SetpointManager:OutdoorAirPretreat  SetpointManager:MultiZone:Cooling:Average  SetpointManager:MultiZone:Heating:Average  SetpointManager:Warmest  SetpointManager:Coldest  SetpointManager:WarmestTemperatureFlow  SetpointManager:FollowOutdoorAirTemperature  SetpointManager:FollowGroundTemperature  SetpointManager:FollowSystemNodeTemperature  SetpointManager:SingleZone:OneStageCooling  SetpointManager:SingleZoneOneStageHeating | Reduce the setpoint in each object by the value of sensor bias to impose fault.  Setpointfault = Setpoint - bias |
| SetpointManager:Scheduled  SetpointManager:Scheduled:DualSetpoint  SetpointManager:ReturnAirBypassFlow | Use EMS to impose fault.   * Define sensor object (storing actual sensor values). * Define program object (calculate faulted sensor measurement). * Define ProgramCallingManager object (define EMS calling point). * Define Actuator object (apply sensor bias to economizer object(s)). |

### Tests

* Test model with several SetpointManager objects shown in above table.
* Test invalid user argument values to make sure measure fails gracefully.