## Biased Economizer Sensor: Outdoor Relative Humidity

### 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. A positive bias in the economizer outdoor relative humidity (RH) sensor leads to a higher estimate in the outdoor air enthalpy, which shifts the economizer switch-off point and could cause higher cooling or heating energy consumption. This measure simulates the biased economizer sensor (outdoor air RH) by modifying the Controller:OutdoorAir object in EnergyPlus assigned to the heating and cooling system. The fault intensity (F) for this fault is defined as the biased RH level (%)"

end

### Modeler Description

def modeler\_description

return "Two user inputs are required, based on these user inputs, the outdoor air RH reading in the economizer will be replaced by the equation below, where RHoaF is the biased outdoor air RH reading, RHoa is the actual outdoor air RH, and F is the fault intensity. RHoaF = RHoa + F. To use this measure, choose the Controller:OutdoorAir object to be faulted. Set the level of relative humidity sensor bias between 0 to 1 that you want at the outdoor air duct for the economizer during the simulation period. For example, setting F=3 means the sensor is reading 25% when the actual relative humidity is 22%. You can also impose a schedule of the presence of fault during the simulation period. If a schedule name is not given, the model assumes that the fault is present during the entire simulation period."

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 relative humidity sensor bias

oa\_rh\_bias = OpenStudio::Ruleset::OSArgument::makeDoubleArgument("oa\_rh\_bias", false)

oa\_rh\_bias.setDisplayName("Enter the bias level of the return air relative humidity sensor. A positive number means that the sensor is reading a relative humidity higher than the true relative humidity. [%]")

oa\_rh\_bias.setDefaultValue(0) #default fouling level to be 0%

args << oa\_rh\_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 level is defined as zero,

runner.registerAsNotApplicable("#{name} is not running with zero fault level. Skipping......")

#When MinimumFlowWithBypass is selected as economizer option,

runner.registerAsNotApplicable("MinimumFlowWithBypass in #{econ\_choice} is not an economizer and is not supported. Skipping......")

#When LockoutWithHeating or LockoutWithCompressor is selected as economizer option,

runner.registerAsNotApplicable(controlleroutdoorair.getString(14).to\_s+" in #{econ\_choice} is not supported. Skipping......")

#When BypassWhenOAFlowGreaterThanMinimum is selected as economizer option,

runner.registerAsNotApplicable(controlleroutdoorair.getString(25).to\_s+" in #{econ\_choice} is not supported. Skipping......")

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

runner.registerAsNotApplicable("#{name} is not running for #{econ\_choice} because of inapplicability. Skipping......")

### Warning

n/a

### Error

#When selected economizer cannot be found in the model,

runner.registerError("Measure #{name} cannot find #{econ\_choice}. Exiting......")

### Information

Measures below share the same resource codes.

* BiasedEconomizerSensorReturnRH
* BiasedEconomizerSensorOutdoorRH

### Code Outline

* Define arguments (economizer where fault occurs, fault level in constant value).
* Find the economizer where the fault occurs.
* Check applicability of the model to the economizer defined in the model.
* Write EMS program (appropriately according to economizer options that were already defined in the model) to impose sensor bias for each economizer object.
* Append EMS code to impose sensor bias level at the outdoor air measurement reading.
* Append EMS code to recalculate other thermophysical properties based on biased reading.
* Append appropriate EMS code if Minimum Outdoor Air Schedule option is defined.
* Append appropriate EMS code if Mechanical Ventilation Controller option is defined.
* Append appropriate EMS code if Economizer Control Type option is defined as NoEconomizer. If not,
  + Append appropriate EMS code if Lockout Type option is defined.
    - Append appropriate EMS code if Lockout Type option is defined as either LockoutWithHeating or LockoutWithCompressor.
  + Append appropriate EMS code if Economizer Control Type option is defined as DifferentialDryBulb.
  + Append appropriate EMS code if Economizer Control Type option is defined as either FixedDryBulb, FixedEnthalpy, FixedDewPointAndDryBulb or ElectronicEnthalpy.
  + Append appropriate EMS code if Economizer Control Type option is defined as DifferentialDryBulbAndEnthalpy.
  + Append appropriate EMS code if Economizer Control Type option is defined as DifferentialEnthalpy.
  + Append appropriate EMS code if Economizer Minimum Limit Dry-Bulb Temperature option is defined.
  + Append appropriate EMS code if High Humidity Control option is defined as yes.
    - Append appropriate EMS code if Control High Indoor Humidity Based on Outdoor Humidity Ratio option is defined as yes.
  + Append appropriate EMS code if Time of Day Economizer Control Schedule Name option is defined.
* Append appropriate EMS code if Economizer Control Action Type option is defined as MinimumFlowWithBypass.
* Append appropriate EMS code if High Humidity Control option is defined as yes.
* Append appropriate EMS code if Minimum Fraction of Outdoor Air Schedule Name option is defined.
* Append appropriate EMS code if Maximum Fraction of Outdoor Air Schedule Name option is defined.
* Append appropriate EMS code to calculate modified outdoor air flow rate.
* Append appropriate EMS code to check whether modified outdoor air flow rate exceeds maximum limit.
* Append appropriate EMS code and texts for defining objects in idf based on above options to complete the code.
  + Define EnergyManagementSystem:Subroutine
  + Define EnergyManagementSystem:ProgramCallingManager
  + Define EnergyManagementSystem:GlobalVariable
  + Define EnergyManagementSystem:Actuator
  + Define EnergyManagementSystem:InternalVariable
  + Define EnergyManagementSystem:Sensor
  + Define Output:EnergyManagementSystem

### Tests

* Test model with different Cotroller:OutdoorAir options.
* Test invalid user argument values to make sure measure fails gracefully.