1. Window (IOT 24)

|  |
| --- |
| var ENVIRONMENTS = ["Argon", "CO", "CO2", "Hydrogen", "Helium", "Methane", "Nitrogen", "O2", "Ozone", "Propane", "Smoke"];  var ENVIRONMENT\_MAX\_IMPACT = -0.01; // 2% max when door opens  var TEMPERATURE\_TRANSFERENCE\_MULTIPLIER = 1.20; // increase speed 25% when door open  var HUMIDITY\_TRANSFERENCE\_MULTIPLIER = 1.20;  var GASES\_TRANSFERENCE\_MULTIPLIER = 2;  var state = 0;  //set up client to talk and listen to IoE registration server  function setup() {  IoEClient.setup({  type: "Window",  states: [{  name: "On",  type: "bool",  controllable: true  }]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state ? 0 : 1);  }  //update carbon dioxide and carbon monoxide and send new data to registration server  function loop() {  updateEnvironment();  delay(1000);  }  //process data received from server  function processData(data, bIsRemote)  {  if ( data.length <= 0 )  return;  data = data.split(",");  setState(parseInt(data[0]));  }  //set state and update component image to reflect the current state  function setState(newState)  {  if ( newState === 0 )  digitalWrite(1, LOW);  else{  digitalWrite(1, HIGH);  }  state = newState;  customWrite(0, state);  IoEClient.reportStates(state);  setDeviceProperty(getName(), "state", state);  }  function updateEnvironment()  {  var rate,max;  if ( state == 1)  {  for(var i=0; i<ENVIRONMENTS.length; i++){  max = Environment.get(ENVIRONMENTS[i]) \* ENVIRONMENT\_MAX\_IMPACT;  // the max is reached in an hour, so we divide by 3600 to get seconds  // then this rate is also based on 100,000 cubic meters (approx. coporate office size)  rate = max / 3600 \* 100000 / Environment.getVolume();  Environment.setContribution(ENVIRONMENTS[i], rate, max);  Environment.setTransferenceMultiplier(ENVIRONMENTS[i], GASES\_TRANSFERENCE\_MULTIPLIER);  }    Environment.setTransferenceMultiplier("Ambient Temperature", TEMPERATURE\_TRANSFERENCE\_MULTIPLIER);  Environment.setTransferenceMultiplier("Humidity", HUMIDITY\_TRANSFERENCE\_MULTIPLIER);  }  else  {  for(var j=0; j<ENVIRONMENTS.length; j++){  Environment.setContribution(ENVIRONMENTS[j], 0, 0);  Environment.removeCumulativeContribution(ENVIRONMENTS[j]);  Environment.setTransferenceMultiplier(ENVIRONMENTS[j], 1);  }  Environment.setTransferenceMultiplier("Ambient Temperature", 1);  Environment.setTransferenceMultiplier("Humidity", 1);    }  } |

2. Air-Conditioner (IOT 34)

|  |
| --- |
| var HUMIDITY\_RATE = -2/3600; // -2% per hour  var TEMPERATURE\_RATE = -10/3600; // -10C per hour  var VOLUME\_AT\_RATE = 100000;  var input;  function setup() {  IoEClient.setup({  type: "AC",  states: [{  name: "On",  type: "bool",  controllable: true  }]  });    IoEClient.onInputReceive = function(data) {  if ( data.length <= 0 )  return;  data = data.split(",");  processData(parseInt(data[0]));  };  attachInterrupt(0, function() {  processData(digitalRead(0)/1023);  });  processData(digitalRead(0)/1023);    var VAR = getDeviceProperty(getName(), "VOLUME\_AT\_RATE");  if(!VAR)  setDeviceProperty(getName(), "VOLUME\_AT\_RATE", VOLUME\_AT\_RATE);  }  function processData(data) {  input = data;  if ( input > 0 )  digitalWrite(5, HIGH);  else  digitalWrite(5, LOW);  IoEClient.reportStates(input);  }  function loop()  {  updateEnvironment();  delay(1000);  }  function updateEnvironment()  {  var VAR = parseFloat(getDeviceProperty(getName(), "VOLUME\_AT\_RATE"));  if( VAR < 0 )  VAR = VOLUME\_AT\_RATE;  //Serial.println("VAR: " + VAR);  //Serial.println("input: " + input);  var humidity\_rate = input\*HUMIDITY\_RATE \* VAR / Environment.getVolume();  var temperature\_rate = input\*TEMPERATURE\_RATE \* VAR / Environment.getVolume();  Environment.setContribution("Humidity", humidity\_rate);  Environment.setContribution("Ambient Temperature", temperature\_rate);    //Serial.println("T rate: " + temperature\_rate);  } |

3. Ceiling-Fan (IOT 0): -

|  |
| --- |
| var FAN\_SPEED\_LOW = 0.4; // kph  var FAN\_SPEED\_HIGH = 0.8; // kph  var COOLING\_RATE = -1/3600; // -1C/hour  var HUMDITY\_REDUCTION\_RATE = -1/3600; // -1%/hour  var VOLUME\_AT\_RATE = 100000; // the given rates are based on this volume  var state = 0; // 0 off, 1 low, 2 high  var level = 0;  function setup() {    IoEClient.setup({  type: "Ceiling Fan",  states: [  {  name: "Status",  type: "options",  options: {  "0": "Off",  "1": "Low",  "2": "High"  },  controllable: true  }  ]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  toggleState();  }  function processData(data, bIsRemote)  {  if ( data.length <= 0 )  return;  data = data.split(",");  setState(parseInt(data[0]));  }  function sendReport()  {  var report = state; // comma seperated states  customWrite(0, report);  IoEClient.reportStates(report);  setDeviceProperty(getName(), "state", state);  }  function setState(newState)  {  analogWrite(A1, newState);  state = newState;    sendReport();  updateEnvironment();  }  function toggleState()  {  ++state;  if ( state >= 3 )  state = 0;    setState(state);  }  function updateEnvironment()  {  var volumeRatio = VOLUME\_AT\_RATE / Environment.getVolume();  if ( state === 0){  Environment.setContribution("Wind Speed", 0, 0);  Environment.setContribution("Ambient Temperature", 0, 0);  Environment.setContribution("Humidity", 0,0);  }  else if ( state == 1 )  {  Environment.setContribution("Wind Speed", FAN\_SPEED\_LOW, FAN\_SPEED\_LOW, false);    // everytime the fan restarts, it can do another -100C  Environment.setContribution("Ambient Temperature", COOLING\_RATE/2\*volumeRatio,  Environment.getCumulativeContribution("Ambient Temperature")-100);  Environment.setContribution("Humidity", HUMDITY\_REDUCTION\_RATE/2\*volumeRatio,  Environment.getCumulativeContribution("Humidity")-100);    }  else if ( state == 2)  {  Environment.setContribution("Wind Speed", FAN\_SPEED\_HIGH, FAN\_SPEED\_HIGH, false);    Environment.setContribution("Ambient Temperature", COOLING\_RATE/2\*volumeRatio,  Environment.getCumulativeContribution("Ambient Temperature")-100);    Environment.setContribution("Humidity", HUMDITY\_REDUCTION\_RATE\*volumeRatio,  Environment.getCumulativeContribution("Humidity")-100);  }        } |

4. Beacon (IOT 22):-

|  |
| --- |
| var DEFAULT\_BEACON\_UUID = "{00000000-0000-0000-0000-000000000001}";  var DEFAULT\_BEACON\_DATA = "Location 1";  function setup() {  Bluetooth.init();  Bluetooth.enableBroadcast(true);    var uuid = getDeviceProperty(getName(), "beaconUuid");  if (uuid == null)  setDeviceProperty(getName(), "beaconUuid", DEFAULT\_BEACON\_UUID);  var data = getDeviceProperty(getName(), "beaconData");  if (data == null)  setDeviceProperty(getName(), "beaconData", DEFAULT\_BEACON\_DATA);  }  function loop() {  var uuid = getDeviceProperty(getName(), "beaconUuid");  var data = getDeviceProperty(getName(), "beaconData");  Serial.println("Broadcasting to " + uuid);  Bluetooth.broadcastBeacon(uuid, data);    delay(5000);  } |

5. Light (IOT 10)

|  |
| --- |
| var ENVIRONMENT\_IMPACT\_DIM = 10;  var VOLUME\_AT\_RATE = 100000;  var state = 0; // 0 off, 1 low, 2 high  var lastTimeInSeconds = 0;  function setup() {  IoEClient.setup({  type: "Light",  states: [  {  name: "Status",  type: "options",  options: {  "0": "Off",  "1": "Dim",  "2": "On"  },  controllable: true  }  ]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });  state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state+1);  }  function loop() {  updateEnvironment();  delay(1000);  }  function processData(data, bIsRemote) {  if ( data.length <= 0 )  return;  setState(parseInt(data));  }  function setState(newState) {  if (newState >= 3)  newState = 0;  state = newState;    analogWrite(A1, state);  customWrite(0, state);  IoEClient.reportStates(state);  setDeviceProperty(getName(), "state", state);  }  function updateEnvironment()  {  var volumeRatio = VOLUME\_AT\_RATE / Environment.getVolume();  if ( state === 0 )  Environment.setContribution("Visible Light", 0,0);  else if ( state === 1)  Environment.setContribution("Visible Light", ENVIRONMENT\_IMPACT\_DIM\*volumeRatio, ENVIRONMENT\_IMPACT\_DIM\*volumeRatio, false);  else if ( state === 2 )  Environment.setContribution("Visible Light", ENVIRONMENT\_IMPACT\_DIM\*2\*volumeRatio, ENVIRONMENT\_IMPACT\_DIM\*2\*volumeRatio, false);  } |

6. Solar-Panel (IOT 32): -

|  |
| --- |
| //Solar Panel  //Read the sunlight levels  //Output electricity based on sunlight  //Panel will be 160Watts per square meter  //Features output to IoE Server:  // number of kWh of energy produced since turning on  // number of kWh per minute  // current production  var ENVIRONMENT\_NAME = "Sunlight";  var MULTIPLIER = 255/1023;  var MAX\_POWER = 1000;//1000 Watts of power based on one meter solar panel at noon at the equator  var EFFICIENCY = 0.16;//About a 16 percent efficiency per solar panel  var PANEL\_POWER = MAX\_POWER \* EFFICIENCY;  var LOG\_BASE = 1.0749111034571373359815489867558;  var state = 1;  var electricity = 0;  //var tick = 0;  function setup(){    IoEClient.setup({  type: "Solar",  states: [  {  name: "Status",  type: "number",  unit: 'Wh',  controllable: false  }  ]  });    IoEClient.onInputReceive = function(input) {  // Serial.println("input: " + input);  processData(input, true);  };      sendReport();  }  function loop(){  // if ( (tick++ % 10) === 0 ) // is tick consistent across devices?  // {  electricity = Math.round(getElectricityProduction());  //Serial.println(electricity);  displayElectricity();  sendReport();  outputElectricity();  delay(1000);  // }  }  function displayElectricity(){  setCustomText(70, 45, 1000, 1000, String(parseInt(electricity)) + '\tW');  }  function getElectricityProduction(){  return PANEL\_POWER \* Environment.get(ENVIRONMENT\_NAME) / 100;  }  function sendReport()  {  var report = state; // comma seperated states  IoEClient.reportStates(electricity);  setDeviceProperty(getName(), "level", electricity);  }  function outputElectricity(){  var el\_log = Math.floor(Math.log(electricity)/Math.log(LOG\_BASE));  if(el\_log < 0)  el\_log = 0;  else if (el\_log > 255)  el\_log = 255;  // Serial.println(el\_log);  analogWrite(0, el\_log);  } |

7. Wind-Detector (IOT 33): -

|  |
| --- |
| var ENVIRONMENT\_NAME = "Wind Speed";  var state = 0;  var level = 0;  var tick = 0;  //set up client to talk and listen to IoE registration server  function setup() {  IoEClient.setup({  type: "Wind Detector",  states: [{  name: "Wind",  type: "bool",  controllable: false  }]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    setState(state);  sendReport();  }  //continously checking if WIND exist and send report to registration server  function loop() {  if ( (tick++ % 10) === 0 ) // is tick consistent across devices?  {  detect();  sendReport();  }  }  //get WIND variable defined in Environment  function detect()  {  var value = Environment.get(ENVIRONMENT\_NAME);  if (value >= 1 )  setState(1);  else  setState(0);  }  //process data received from server  //not being called since controllable set to false in client setup  function processData(data, bIsRemote)  {  if ( data.length <= 0 )  return;  data = data.split(",");  setState(parseInt(data[0]));  }  //send wind state to the server  function sendReport()  {  var report = state; // comma seperated states  IoEClient.reportStates(report);  }  //set state and update component image to reflect the current state  function setState(newState)  {  if ( newState === 0 ){  digitalWrite(1, LOW);  }  else{  digitalWrite(1, HIGH);  }    state = newState;    sendReport();  }  //toggle wind state  function toggleState()  {  if ( state === 0)  setState(1);  else  setState(0);  } |

8. Door (IOT 5): -

|  |
| --- |
| var ENVIRONMENTS = ["Argon", "CO", "CO2", "Hydrogen", "Helium", "Methane", "Nitrogen", "O2", "Ozone", "Propane", "Smoke"];  var ENVIRONMENT\_MAX\_IMPACT = -0.02; // 2% max when door opens  var TEMPERATURE\_TRANSFERENCE\_MULTIPLIER = 1.25; // increase speed 25% when door open  var HUMIDITY\_TRANSFERENCE\_MULTIPLIER = 1.25;  var GASES\_TRANSFERENCE\_MULTIPLIER = 2;  var doorState = 0; // 0 is closed, 1 is opened  var lockState = 0; // 0 is unlocked, 1 is locked  function setup () {  IoEClient.setup({  type: "Door",  states: [{  name: "Open",  type: "bool"  }, {  name: "Lock",  type: "options",  options: {  "0": "Unlock",  "1": "Lock"  },  controllable: true  }]  });    IoEClient.onInputReceive = function (input) {  processData(input, true);  };    attachInterrupt(0, function () {  processData(customRead(0), false);  });    setDoorState(doorState);  setLockState(lockState);  }  function mouseEvent (pressed, x, y, firstPress) {  if (firstPress) {  if ( isPointInRectangle(x, y, 10,40,5,10) )  {  if ( lockState === 0 ) {  setLockState(1);  } else {  setLockState(0);  }  } else {  if ( doorState === 0 ) {  openDoor();  } else {  closeDoor();  }  }  }  }  function loop () {    }  function processData (data, bIsRemote) {  if ( data.length <= 0 ) {  return;  }  Serial.println(data);    data = data.split(",");  var doorStateData = parseInt(data[0]);  var lockStateData = parseInt(data[1]);  if ( lockStateData > -1 ) {  setLockState(lockStateData);  }    if ( doorStateData > -1 && !bIsRemote ) {  if ( doorStateData === 0 ) {  closeDoor();  } else {  openDoor();  }  }  }  function sendReport () {  var report = doorState+","+lockState; // comma seperated states  customWrite(0, report);  IoEClient.reportStates(report);  setDeviceProperty(getName(), "door state", doorState);  setDeviceProperty(getName(), "lock state", lockState);  }  function closeDoor () {  setDoorState(0);  updateEnvironment();  }  function openDoor () {  if ( lockState===0 ) {  setDoorState(1);  updateEnvironment();  } else {  Serial.println("can't open locked door");  }      }  function setDoorState (state) {  if ( state === 0) {  digitalWrite(1, LOW);  setComponentOpacity("led", 1); // show the led  } else {  digitalWrite(1, HIGH);  setComponentOpacity("led", 0); // hide the led  }  doorState = state;  sendReport();  }  function setLockState (state) {  if ( state === 0 ) {  digitalWrite(2, LOW);  } else {  digitalWrite(2, HIGH);  }    lockState = state;  sendReport();  }  function updateEnvironment () {  var rate,max;  if ( doorState == 1) {  for (var i=0; i<ENVIRONMENTS.length; i++) {  max = Environment.get(ENVIRONMENTS[i]) \* ENVIRONMENT\_MAX\_IMPACT;  // the max is reached in an hour, so we divide by 3600 to get seconds  // then this rate is also based on 100,000 cubic meters (approx. coporate office size)  rate = max / 3600 \* 100000 / Environment.getVolume();  Environment.setContribution(ENVIRONMENTS[i], rate, max);  Environment.setTransferenceMultiplier(ENVIRONMENTS[i], GASES\_TRANSFERENCE\_MULTIPLIER);  }    Environment.setTransferenceMultiplier("Ambient Temperature", TEMPERATURE\_TRANSFERENCE\_MULTIPLIER);  Environment.setTransferenceMultiplier("Humidity", HUMIDITY\_TRANSFERENCE\_MULTIPLIER);  } else {  for (var i=0; i<ENVIRONMENTS.length; i++) {  Environment.setContribution(ENVIRONMENTS[i], 0, 0);  Environment.removeCumulativeContribution(ENVIRONMENTS[i]);  Environment.setTransferenceMultiplier(ENVIRONMENTS[i], 1);  }  Environment.setTransferenceMultiplier("Ambient Temperature", 1);  Environment.setTransferenceMultiplier("Humidity", 1);  }  }  function isPointInRectangle (x,y, rx, ry, width, height) {  if (width <= 0 || height <= 0) {  return false;  }    return (x >= rx && x <= rx + width && y >= ry && y <= ry + height);  } |

9. Appliance:-

|  |
| --- |
| var state = 0;  function setup() {    IoEClient.setup({  type: "Appliance",  states: [{  name: "On",  type: "bool",  controllable: true  }]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });    setState(state);  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state ? 0 : 1);  }  function processData(data, bIsRemote) {  if ( data.length <= 0 )  return;  setState(parseInt(data));  }  function setState(newState) {  state = newState;    if ( state === 0 )  digitalWrite(1, LOW);  else  digitalWrite(1, HIGH);    customWrite(0, state);  IoEClient.reportStates(state);  setDeviceProperty(getName(), "state", state);  } |

10. Water Drain: -

|  |
| --- |
| var WATERLEVEL\_RATE = -0.5; // 0.5 cm per second  var VOLUME\_AT\_RATE = 100000;  var state = 0; // 0 off, 1 on  function setup() {    IoEClient.setup({  type: "Water Drain",  states: [  {  name: "Status",  type: "bool",  controllable: true  }  ]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state ? 0 : 1);  }  function processData(data, bIsRemote) {  if ( data.length <= 0 )  return;  setState(parseInt(data));  }  function setState(newState)  {  state = newState;  analogWrite(A1, state);  customWrite(0, state);  IoEClient.reportStates(state);  setDeviceProperty(getName(), "state", state);  updateEnvironment();  }  function updateEnvironment()  {  if ( state == 1){  var volumeRatio = VOLUME\_AT\_RATE / Environment.getVolume();  if ( Environment.get("Water Level") >= 0 )  Environment.setContribution("Water Level", WATERLEVEL\_RATE\*volumeRatio);  else  Environment.setContribution("Water Level", 0);  }  else  {  Environment.setContribution("Water Level", 0);  }  } |

11. Smoke-Detector: -

|  |
| --- |
| var ENVIRONMENT\_NAME = "Smoke";  var state = 0;  var level = 0;  var ALARM\_LEVEL = 40;  function setup() {  IoEClient.setup({  type: "Smoke Detector",  states: [{  name: "Alarm",  type: "bool",  controllable: false  },  {  name: "Level",  type: "number",  controllable: false  }]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  restoreProperty("Alarm Level", 40);    setState(state);    }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function loop() {  var value = Environment.get(ENVIRONMENT\_NAME);  //Serial.println(value);  if (value >= 0)  setLevel(Environment.get(ENVIRONMENT\_NAME));  delay(1000);  }  function processData(data, bIsRemote) {  if (data.length <= 0 )  return;  data = data.split(",");  setState(parseInt(data[0]));  }  function sendReport()  {  var report = state + "," + level; // comma seperated states  IoEClient.reportStates(report);  setDeviceProperty(getName(), "state", state);  setDeviceProperty(getName(), "level", level);  }  function setState(newState) {  state = newState;  if (newState === 0)  digitalWrite(1, LOW);  else  digitalWrite(1, HIGH);  sendReport();  }  function setLevel(newLevel) {  if (level == newLevel)  return;    level = newLevel;  if (level > ALARM\_LEVEL)  setState(1);  else  setState(0);    sendReport();  } |

12. Web- Cam: -

|  |
| --- |
| var state = 0;  //set up client to talk and listen to IoE registration server  function setup() {    IoEClient.setup({  type: "Webcam",  states: [{  name: "On",  type: "bool",  controllable: true  },  {  name: "Image",  type: "image"  }]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  sendReport();  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state ? 0 : 1);  }  //send captured image file path to registration server  function loop() {  sendReport();  delay(1000);  }  //process data received from server  function processData(data, bIsRemote)  {  if ( data.length <= 0 )  return;  data = data.split(",");  setState(parseInt(data[0]));  }  //send image path to server  var imageLoop=0;  function sendReport()  {  var report = state + ","; // comma seperated states    if (state === 0)  report += '../art/IoE/SmartDevices/camera\_off.png';  else{  report += '../art/IoE/SmartDevices/camera\_image'+imageLoop+'.png';  imageLoop++;  if ( imageLoop >= 3)  imageLoop =0;  }  customWrite(0, report);  IoEClient.reportStates(report);  setDeviceProperty(getName(), "state", state);    }  //set state and update component image to reflect the current state  function setState(newState)  {  if ( newState === 0 )  digitalWrite(1, LOW);  else  digitalWrite(1, HIGH);    state = newState;  } |

13. Lawn Sprinkler

|  |
| --- |
| var WATERLEVEL\_RATE = 0.1; // 0.1 cm per second  var HUMIDITY\_RATE = 5/3600; // 5% per hour  var VOLUME\_AT\_RATE = 100000;  var state = 0; // 0 off, 1 on  function setup() {    IoEClient.setup({  type: "Lawn Sprinkler",  states: [  {  name: "Status",  type: "bool",  controllable: true  }  ]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };    attachInterrupt(0, function() {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state ? 0 : 1);  }  function processData(data, bIsRemote) {  if ( data.length <= 0 )  return;  setState(parseInt(data));  }  function setState(newState)  {  state = newState;    digitalWrite(5, state);  customWrite(0, state);  IoEClient.reportStates(state);  setDeviceProperty(getName(), "state", state);  updateEnvironment();  }  function updateEnvironment()  {  if ( state == 1){  var volumeRatio = VOLUME\_AT\_RATE / Environment.getVolume();  Environment.setContribution("Water Level", WATERLEVEL\_RATE\*volumeRatio);  Environment.setContribution("Humidity", HUMIDITY\_RATE\*volumeRatio);  }  else  {  Environment.setContribution("Water Level", 0);  Environment.setContribution("Humidity", 0);  }  } |

14. Siren: -

|  |
| --- |
| var state = 0; // 0 off, 1 on  function setup() {  IoEClient.setup({  type: "Siren",  states: [{  name: "On",  type: "bool",  controllable: true  }]  });    IoEClient.onInputReceive = function(input) {  processData(input, true);  };  attachInterrupt(0, function() {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty( getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty( getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state ? 0 : 1);  }  function processData(data, bIsRemote) {  if ( data.length <= 0 )  return;  setState(parseInt(data));  }  function setState(newState) {    state = newState;  digitalWrite(1, state ? HIGH : LOW);  customWrite(0, state);  IoEClient.reportStates(state);  setDeviceProperty( getName(), "state", state);  } |

15. Home Speaker: -

|  |
| --- |
| var VOLUME\_AT\_RATE = 100000;  var SOUND\_VOLUME = 65;  var SOUND\_PITCH = 20;  var SOUND\_WHITE\_NOISE = 20;  // Purpose:  // Home Speaker that can play one of several predefined sounds.  var g\_sounds = [{soundID: 'sound1', soundPath: '/../Sounds/buzzLow.wav'},  {soundID: 'sound2', soundPath: '/../Sounds/buzzMedium.wav'},  {soundID: 'sound3', soundPath: '/../Sounds/buzzHigh.wav'}];    var g\_currSound = -1;  // Purpose:  // Setup the Home Speaker. Stop any old sounds and add the sounds to use.  function setup()  {  setDeviceProperty(getName(), 'SOUND\_dB', 0);  IoEClient.setup({  type: 'Home Speaker',  states: [{  name: 'Signal',  type: 'number',  controllable: false  }]  });    destroySounds();    for(var ind = 0; ind < g\_sounds.length; ++ind)  addSound(g\_sounds[ind].soundID, g\_sounds[ind].soundPath);    restoreProperty("SOUND\_dB", 0);  g\_currSound = restoreProperty("Signal", -1);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  // Purpose:  // Update function. Is called once each update.  function loop()  {  updateState();  delay(1000);  }  // Purpose:  // Update the sound state, reading from the slot and playing the approperiate sound.  function updateState()  {  var playValue = 255\*(analogRead(A0)/1023);  if(0 > playValue)  playValue = 0;  else if(playValue > 255)  playValue = 255;    setDeviceProperty(getName(), 'SOUND\_dB', playValue \* (60/255));  if(-1 != g\_currSound)  stopSound(g\_sounds[g\_currSound].soundID);    if(0 === playValue)  g\_currSound = -1;  else  g\_currSound = Math.floor((playValue-1) / (255/g\_sounds.length));    if(-1 !== g\_currSound)  playSound(g\_sounds[g\_currSound].soundID, 1);    IoEClient.reportStates(g\_currSound);  setDeviceProperty(getName(), "Signal", g\_currSound);  updateEnvironment();  }  function updateEnvironment()  {  var volumeRatio = VOLUME\_AT\_RATE / Environment.getVolume();  if (g\_currSound >= 0)  {  Environment.setContribution("Sound Volume", SOUND\_VOLUME\*volumeRatio, SOUND\_VOLUME\*volumeRatio, false);  Environment.setContribution("Sound Pitch", SOUND\_PITCH\*(g\_currSound+1), SOUND\_PITCH\*(g\_currSound+1), false);  Environment.setContribution("White Noise", SOUND\_WHITE\_NOISE, SOUND\_WHITE\_NOISE, false);  }  else  {  Environment.setContribution("Sound Volume", 0);  Environment.setContribution("Sound Pitch", 0);  Environment.setContribution("White Noise", 0);  }  } |

16. Street Lamp: -

* (main.js)

|  |
| --- |
| var DELAY = 200;  var SIZE = {width: 150, height: 150}; // largest component size  var SERVER\_PORT = 1234;  var SERVER\_IP = "192.168.0.100";  var socket = null;  var sensors = {};  var count = 0;  function setup() {  IoEClient.setup({  type: "Street Lamp",  states: [{  name: "Light",  type: "number",  controllable: false  }, {  name: "Light gradient",  type: "options",  options: {  "-1": "Decreasing",  "0": "No Change",  "1": "Increasing"  },  controllable: false  }, {  name: "Motion",  type: "number",  controllable: false  }, {  name: "Moton gradient",  type: "options",  options: {  "-1": "Decreasing",  "0": "No Change",  "1": "Increasing"  },  controllable: false  }]  });    sensors.light = new SensorLight();  sensors.motion = new SensorMotion();    socket = new UDPSocket();  socket.begin(SERVER\_PORT);  }  function loop() {  var t = new Timer();  var xpos = getX(),  ypos = getY();  var devices = devicesAt(xpos, ypos, SIZE.width, SIZE.height\*2);  t = new Timer();    sensors.light.update(sensors, devices);  sensors.motion.update(sensors, devices);  // send data to the server  var data = "";    data += "streetlamp," + getSerialNumber() + "^";  data += "light," + sensors.light.value() + "," + sensors.light.valueDirection() + "^";  data += "motion," + sensors.motion.value() + "," + sensors.motion.valueDirection();    socket.send(SERVER\_IP, SERVER\_PORT, data);  sendReport();  delay(DELAY);  }  function sendReport() {  var report = sensors.light.value() + "," + sensors.light.valueDirection() + ","  + sensors.motion.value() + "," + sensors.motion.valueDirection();  IoEClient.reportStates(report);  } |

Sensor-light.js

|  |
| --- |
| // cl-sensor-light.js  // Env.Sunlight is assumed to be from 0 to 100%  var SensorLight = function () {  this.eLightMin = 0;  this.eLightMax = 100; // this is %, per Environment semantics for sunlight  this.eLightMinValueOn = this.eLightMin+(this.eLightMax-this.eLightMin) / 3;  // init  setComponentOpacity("SensorLightOff", 0);  setComponentOpacity("SensorLightOn", 0);  setComponentOpacity("SensorLight", 0);  this.elight = Environment.get("Sunlight");  this.sync\_to\_env();  this.update\_visuals();  };  SensorLight.prototype.update = function (sensors, devices) {  this.sync\_to\_env();  this.update\_visuals();  };  SensorLight.prototype.valueDirection = function () {  if(this.elightPrev === this.elight)  return 0;  else if(this.elightPrev < this.elight)  return 1;  return -1;  };  SensorLight.prototype.value = function () {  return this.elight;  };  // private  SensorLight.prototype.update\_visuals = function () {  // value  var opacity = 0,  value = this.value();  if(value < this.eLightMinValueOn) {  opacity = 1 - (value - this.eLightMin)/(this.eLightMinValueOn - this.eLightMin);  setComponentOpacity("SensorLightOn", 1);  setComponentOpacity("SensorLightOff", 0);  }  else{  setComponentOpacity("SensorLightOn", 0);  setComponentOpacity("SensorLightOff", 1);  }  setComponentOpacity("SensorLightLevel", opacity);  };  // private  SensorLight.prototype.sync\_to\_env = function (){  this.elightPrev = this.elight;  this.elight = Environment.get("Sunlight");  if(this.elight < this.eLightMin)  this.elight = this.eLightMin;  if(this.elight > this.eLightMax)  this.elight = this.eLightMax;  }; |

Sensor motion.js

|  |
| --- |
| // cl-sensor-motion.js  var SensorMotion= function () {  this.motionPrev = 0;  this.motion = 0;  this.devices = {};  // init  this.update\_visuals();  };  SensorMotion.prototype.update = function (sensors, devices) {  this.update\_motion(sensors, devices);  this.update\_visuals();  };  SensorMotion.prototype.valueDirection = function () {  if(this.motionPrev === this.motion)  return 0;  else if(this.motionPrev < this.motion)  return 1;  return -1;  };  SensorMotion.prototype.value = function () {  if(this.motion)  return this.motion;  else if(this.valueDirection() !== 0)  return 1;  return 0;  };  // private  SensorMotion.prototype.update\_visuals = function () {  // value  var value = this.value();  if(value > 0) {  setComponentOpacity("SensorMotionOn", 1);  setComponentOpacity("SensorMotionOff", 0);  }  else{  setComponentOpacity("SensorMotionOn", 0);  setComponentOpacity("SensorMotionOff", 1);  }  };  // private  SensorMotion.prototype.update\_motion = function (sensors, devices) {  var myname = getName();  this.motionPrev = this.motion;  for(var n in this.devices) {  this.devices[n].current = false;  }  for(var i=0; i<devices.length; ++i) {  var name = devices[i],  mydev = this.devices[name];  if(name === myname)  continue;  if(mydev === undefined){  this.devices[name] = {current: true};  ++this.motion;  }  else {  mydev.current = true;  }  }  for(var nn in this.devices){  if(!this.devices[nn].current) {  delete this.devices[nn];  --this.motion;  }  }  }; |

Timer.js

|  |
| --- |
| var Timer = function() {  this.started = (new Date()).getTime();  }  Timer.prototype.elapsed = function(){  return (new Date()).getTime() - this.started;  } |

17. Old Car

|  |
| --- |
| var CO\_RATE = 1/3600; // 1% per hour  var CO2\_RATE = 2/3600;  var SMOKE\_RATE = 3/3600;  var TEMPERATURE\_RATE = 1/3600;  var VOLUME\_AT\_RATE = 100000;  var state = 0;  function updateEnvironment()  {  if ( state == 1 )  {  var volumeRatio = VOLUME\_AT\_RATE / Environment.getVolume();  Environment.setContribution("CO", CO\_RATE\*volumeRatio);  Environment.setContribution("CO2", CO2\_RATE\*volumeRatio);  Environment.setContribution("Smoke", SMOKE\_RATE\*volumeRatio);  Environment.setContribution("Ambient Temperature",TEMPERATURE\_RATE\*volumeRatio );      }  else  {  Environment.setContribution("CO", 0);  Environment.setContribution("CO2", 0);  Environment.setContribution("Smoke", 0);  Environment.setContribution("Ambient Temperature", 0 );  }    }  function setup() {  state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty(propertyName, defaultValue)  {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ){  if ( typeof(defaultValue) == "number" )  value = Number(value);    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent(pressed, x, y, firstPress) {  if (firstPress)  setState(state ? 0 : 1);  }  function setState(newState)  {  if ( newState === 0 )  digitalWrite(1, LOW);  else{  digitalWrite(1, HIGH);  }  state = newState;  setDeviceProperty(getName(), "state", state);  updateEnvironment();  } |

18. Garage Door

|  |
| --- |
| var ENVIRONMENTS = ["Argon", "CO", "CO2", "Hydrogen", "Helium", "Methane", "Nitrogen", "O2", "Ozone", "Propane", "Smoke"];  var ENVIRONMENT\_MAX\_IMPACT = -0.04; // 4% max when door opens  var TEMPERATURE\_TRANSFERENCE\_MULTIPLIER = 1.50; // increase speed 25% when door open  var HUMIDITY\_TRANSFERENCE\_MULTIPLIER = 1.50;  var GASES\_TRANSFERENCE\_MULTIPLIER = 2;  var state = 0; // 0 off, 1 on  function setup () {    IoEClient.setup ({  type: "Garage Door",  states: [{  name: "On",  type: "bool",  controllable: true  }]  });    IoEClient.onInputReceive = function (input) {  processData(input, true);  };    attachInterrupt(0, function () {  processData(customRead(0), false);  });    state = restoreProperty("state", 0);  setState(state);  }  function restoreProperty (propertyName, defaultValue) {  var value = getDeviceProperty(getName(), propertyName);  if ( !(value === "" || value == "undefined") ) {  if ( typeof(defaultValue) == "number" ) {  value = Number(value);  }    setDeviceProperty(getName(), propertyName, value);  return value;  }    return defaultValue;  }  function mouseEvent (pressed, x, y, firstPress) {  if ( firstPress ) {  setState(state ? 0 : 1);  }  }  function updateEnvironment () {  var rate,max;  if ( state == 1) {  for (var i=0; i<ENVIRONMENTS.length; i++) {  max = Environment.get(ENVIRONMENTS[i]) \* ENVIRONMENT\_MAX\_IMPACT;  // the max is reached in an hour, so we divide by 3600 to get seconds  // then this rate is also based on 100,000 cubic meters (approx. coporate office size)  rate = max / 3600 \* 100000 / Environment.getVolume();  Environment.setContribution(ENVIRONMENTS[i], rate, max);  Environment.setTransferenceMultiplier(ENVIRONMENTS[i], GASES\_TRANSFERENCE\_MULTIPLIER);  }    Environment.setTransferenceMultiplier("Ambient Temperature", TEMPERATURE\_TRANSFERENCE\_MULTIPLIER);  Environment.setTransferenceMultiplier("Humidity", HUMIDITY\_TRANSFERENCE\_MULTIPLIER);  } else {  for ( var j=0; j<ENVIRONMENTS.length; j++ ) {  Environment.setContribution(ENVIRONMENTS[j], 0, 0);  Environment.removeCumulativeContribution(ENVIRONMENTS[j]);  Environment.setTransferenceMultiplier(ENVIRONMENTS[j], 1);  }  Environment.setTransferenceMultiplier("Ambient Temperature", 1);  Environment.setTransferenceMultiplier("Humidity", 1);  }  }  function processData (data, bIsRemote) {  if ( data.length <= 0 ) {  return;  }  setState(parseInt(data));  }  function setState (newState) {  state = newState;  digitalWrite(1, state ? HIGH : LOW);  customWrite(0, state);  IoEClient.reportStates(state);  setDeviceProperty(getName(), "state", state);    updateEnvironment();    } |