```
var access_token = ''; // ←トークンを記入
var spreadsheetId = ''; // ←スプレッドシートの ID を記入
var channelToken = ''; // ← LINE のチャンネルアクセス・トークン
var onSignalId = ''; // 照明をオンにする信号 ID を指定
var offSignalId = ''; // 照明をオフにする信号 ID を指定

function remo() {
  var data = getNatureRemoData(); // API を叩いてデータ取得
  var lastData = getLastData(); // スプレッドシートの記載済最終行を取得
  // スプレッドシート末端に取得したデータを書き込む
  addToSpreadSheet(
    {
      te: data[0].newest_events.te.val, // 温度
      hu: data[0].newest_events.hu.val, // 湿度
      il: data[0].newest_events.il.val, // 照度
      mo_last: data[0].newest_events.mo.created_at, // 人感更新時刻
    },
    lastData.row + 1 // 記載済最終行+1 行目にデータを書きこむ
  );
  // スプレッドシートのデータを基に照明を制御する
  controlLightBasedOnSpreadsheet();
}

// Remo から Get（1/devices）でデータを取得するメソッド（Remo の api 使用）
function getNatureRemoData() {
  // アクセス先 URL（1/devices）
  var url = 'https://api.nature.global/1/devices';
  // ヘッダーに受取形式とトークン埋め込み
  var headers = {
    'Content-Type': 'application/json;',
    'Authorization': 'Bearer ' + access_token,
  };
  // オプションで GET メソッドであることと、ヘッダーを指定
  var options = {
    'method': 'get',
    'headers': headers,
  };
  // UrlFetchApp を使って GET(1/devices) を実行し、センサデータを取得
  var data = JSON.parse(UrlFetchApp.fetch(url, options));
  // 取得したデータをログに記載
  Logger.log(data[0].newest_events);
  // 取得したデータを出力
  return data;
}

// スプレッドシートの記載済最終行を取得するメソッド
function getLastData() {
  var datas = SpreadsheetApp.openById(spreadsheetId).getSheetByName('log').
  getDataRange().getValues(); // log シートをゲットする  +  長いので 1 行を 2 行に分けた
  var data = datas[datas.length - 1];
  return {
    totalpoint: data[1],
    coupon: data[2],
    row: datas.length,
    mo_value: data[4] // E 列（人感センサの値）を取得
```

```
  };
}

// データをスプレッドシートに書き込む
function addToSpreadSheet(data, row) {
  var sheet = SpreadsheetApp.openById(spreadsheetId).getSheetByName('log');
  sheet.getRange(row, 1).setValue(new Date()); // A 列：取得した日時
  sheet.getRange(row, 2).setValue(data.te); // B 列：温度追加
  sheet.getRange(row, 3).setValue(data.hu); // C 列：湿度追加
  sheet.getRange(row, 4).setValue(data.il); // D 列：照度追加
  sheet.getRange(row, 6).setValue(data.mo_last); // I 列：人感更新時刻追加
  // 前行の人感更新時刻を取得
  var previous_mo_last = sheet.getRange(row - 1, 6).getValue();
  // 人感更新時刻が前行と異なる（人感センサ更新ある）とき、E 列に「1」を記載
  if (row >= 2 && previous_mo_last != data.mo_last) {
    sheet.getRange(row, 5).setValue(1);
    sendMessage('人感センサーが反応しました！'); // ここでメッセージを送信
    resetTimer(); // タイマーリセット
  }
  // 人感更新時刻が前行と同じ（人感センサ更新ない）とき、E 列に「0」を記載
  else {
    sheet.getRange(row, 5).setValue(0);
  }
}

// スプレッドシートのデータを基に照明を制御する関数
function controlLightBasedOnSpreadsheet() {
  var lastData = getLastData();
  // スプレッドシートの最終行の人感センサの値が 1 の場合に照明をオンにする
  if (lastData.mo_value == 1) {
    controlLightOn();
  }
}

// 照明をオンにする関数
function controlLightOn() {
  var url = 'https://api.nature.global/1/signals/' + onSignalId + '/send';
  var headers = {
    'Authorization': 'Bearer ' + access_token,
    'Content-Type': 'application/json',
  };
  var options = {
    'method': 'post',
    'headers': headers,
  };
  // リクエストを送信
  var response = UrlFetchApp.fetch(url, options);
  var result = JSON.parse(response.getContentText());
  // レスポンスをログに出力（デバッグ用）
  Logger.log(result);
}

// 照明をオフにする関数
function controlLightOff() {
  var url = 'https://api.nature.global/1/signals/' + offSignalId + '/send';
```

```
  var headers = {
    'Authorization': 'Bearer ' + access_token,
    'Content-Type': 'application/json',
  };
  var options = {
    'method': 'post',
    'headers': headers,
  };
  // リクエストを送信
  var response = UrlFetchApp.fetch(url, options);
  var result = JSON.parse(response.getContentText());
  // レスポンスをログに出力（デバッグ用）
  Logger.log(result);
}

function doPost(e) {
  var replyToken = JSON.parse(e.postData.contents).events[0].replyToken;
  if (typeof replyToken === 'undefined') {
    return;
  }

  var input = JSON.parse(e.postData.contents).events[0].message.text;

  if (input.toLowerCase() === 'y') {
    // y が入力された場合、照明を消灯する
    controlLightOff();
  } else if (input.toLowerCase() === 'n') {
    // n が入力された場合、何もしない（照明をつけたまま）
    sendMessage('照明はつけたままにします。');
  } else {
    // それ以外の入力の場合、再度尋ねる
    sendMessage('電気を消灯しますか？ (y or n)');
  }

  return ContentService.createTextOutput(JSON.stringify({'content': 'post ok'})).
  setMimeType(ContentService.MimeType.JSON);//長いので１行を２行に分けた
}

// ^^e5^^8f^^91送消息函数
function sendMessage(message) {
  var url = 'https://api.line.me/v2/bot/message/push';
  var messages = [{
    'type': 'text',
    'text': message,
  }];

  UrlFetchApp.fetch(url, {
    'headers': {
      'Content-Type': 'application/json; charset=UTF-8',
      'Authorization': 'Bearer ' + channelToken,
    },
    'method': 'post',
    'payload': JSON.stringify({
      to: 'U30f6f507c00ae6625f5c2419b840c0c7',
      messages: messages
```

```
    })
  });
}

// 15 分タイマーのリセット関数
function resetTimer() {
  clearExistingTriggers();
  ScriptApp.newTrigger('sendConfirmationMessage')
           .timeBased()
           .after(15 * 60 * 1000) // 15 分後に実行
           .create();
}

// 既存のトリガーをクリアする関数
function clearExistingTriggers() {
  var allTriggers = ScriptApp.getProjectTriggers();
  for (var i = 0; i < allTriggers.length; i++) {
    if (allTriggers[i].getHandlerFunction() == 'sendConfirmationMessage') {
      ScriptApp.deleteTrigger(allTriggers[i]);
    }
  }
}

// 15 分後に実行される関数
function sendConfirmationMessage() {
  sendMessage('電気を消灯しますか？ (y or n)');
}
```