成果発表

4班
小沼 燦太　数住 幸之介
吉田 啓人　小椋 勇輝　西中 龍

目次

1. 要求仕様
2. 設計
3. プロジェクト計画
4. 実際の開発スケジュール
5. 実装したプログラム
6. 感想

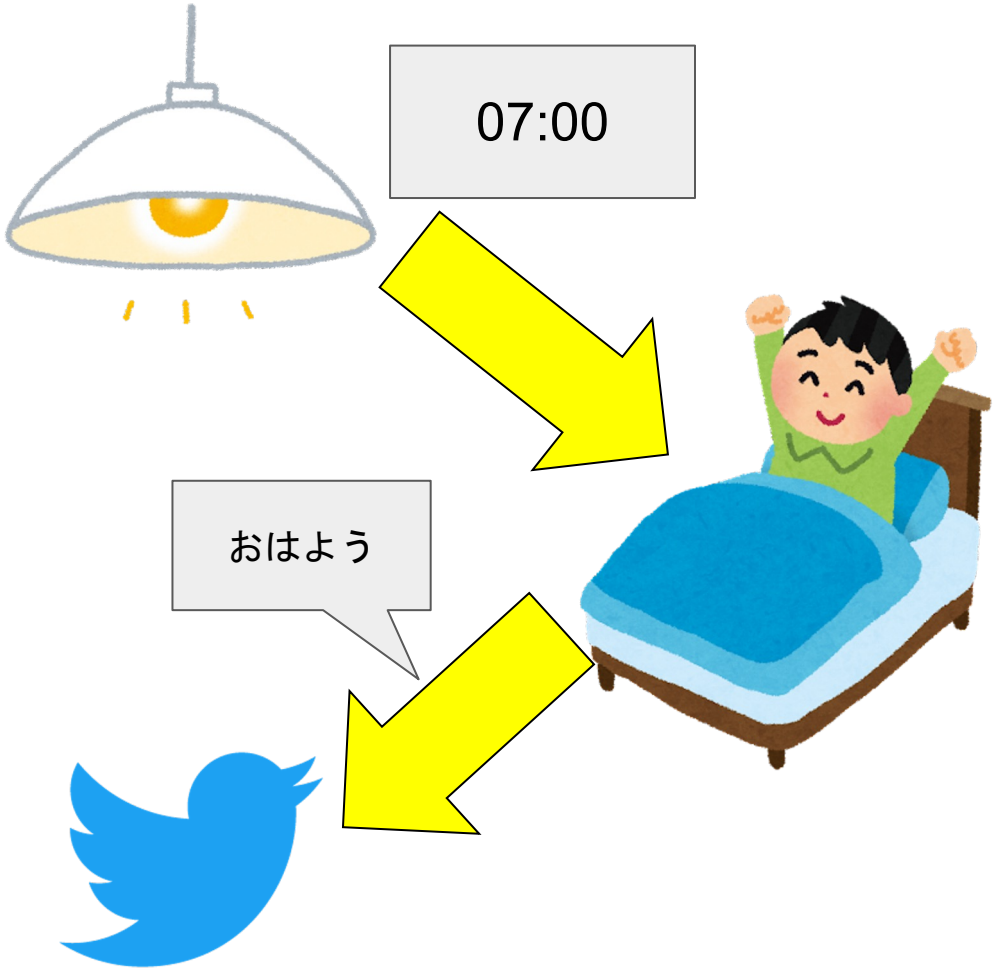
1. 要求仕様

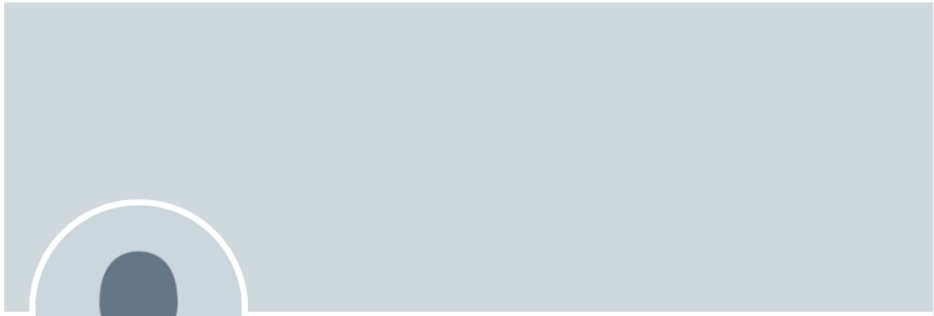
システム概要

・既定の時間に家電を操作し起床を手助けする

・睡眠・起床した時に、twitterに「おはよう」と投稿する

・人感センサーが反応しなければ「二度寝をしています」と投稿する





Set up profile

Remo大好き大学生
@remostudent

Joined May 2023

0 Following **0** Followers

Tweets | Replies | Media | Likes

Remo大好き大学生 @remostudent · 14m
2023年6月3日0:50
ohayo

1.　要求仕様

・ユーザが設定した就寝予定時間以降に照明が消えた際に記録される日付と時刻をスプレッドシート上で確認できること

・ユーザは照明が消えた時間にTwitter上で「おやすみ」と投稿されたことを確認できること

・ユーザが設定した起床予定時間になるとテレビを起動させることができること

・ユーザが設定した起床予定時間になると照明を起動させることができること

・ユーザが設定した時間から10分以内に人感センサーが反応すればTwitter上で「おはよう」と投稿し、反応しなければ「二度寝をしています。。。」と投稿されたことが確認できること

・ユーザが設定した時間以降に人感センサーが反応すればスプレッドシートにその時刻が書き込まれたことを確認できること

想定する利用者

- 全てのtwitterユーザ

特に

- 朝起きるのが苦手な学生や社会人
- 睡眠管理を徹底したい人
- Twitter上で自分の睡眠記録を共有したい人

2. 設計

システムの流れ

人感センサーの
データ渡す

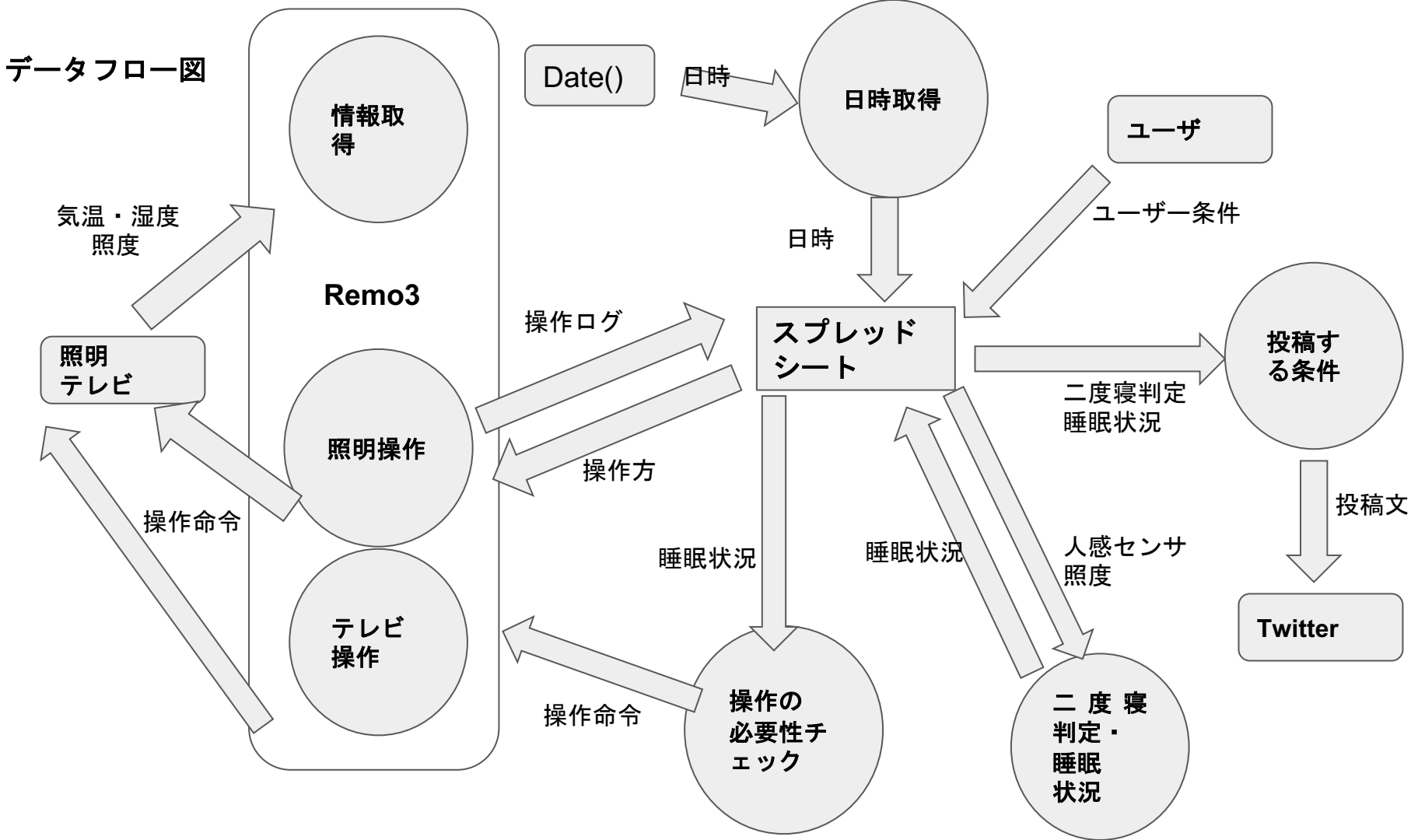
起床・就寝予定時間と
twitterアカウントを設定

Google Sheets

予定起床・就
寝時間になっ
たことを伝達

操作

おはよう、お休み、二度寝
しているツイート


データフロー図
情報取得
Remo3
気温・湿度
照度
照明
テレビ
照明操作
操作命令
テレビ
操作
Date()
日時
日時取得
日時
ユーザ
ユーザー条件
操作ログ
操作方
スプレッド
シート
投稿する条件
二度寝判定
睡眠状況
投稿文
Twitter
睡眠状況
睡眠状況
人感センサ
照度
操作命令
操作の
必要性チェック
二 度 寝
判定・
睡眠
状況

必要なモジュール

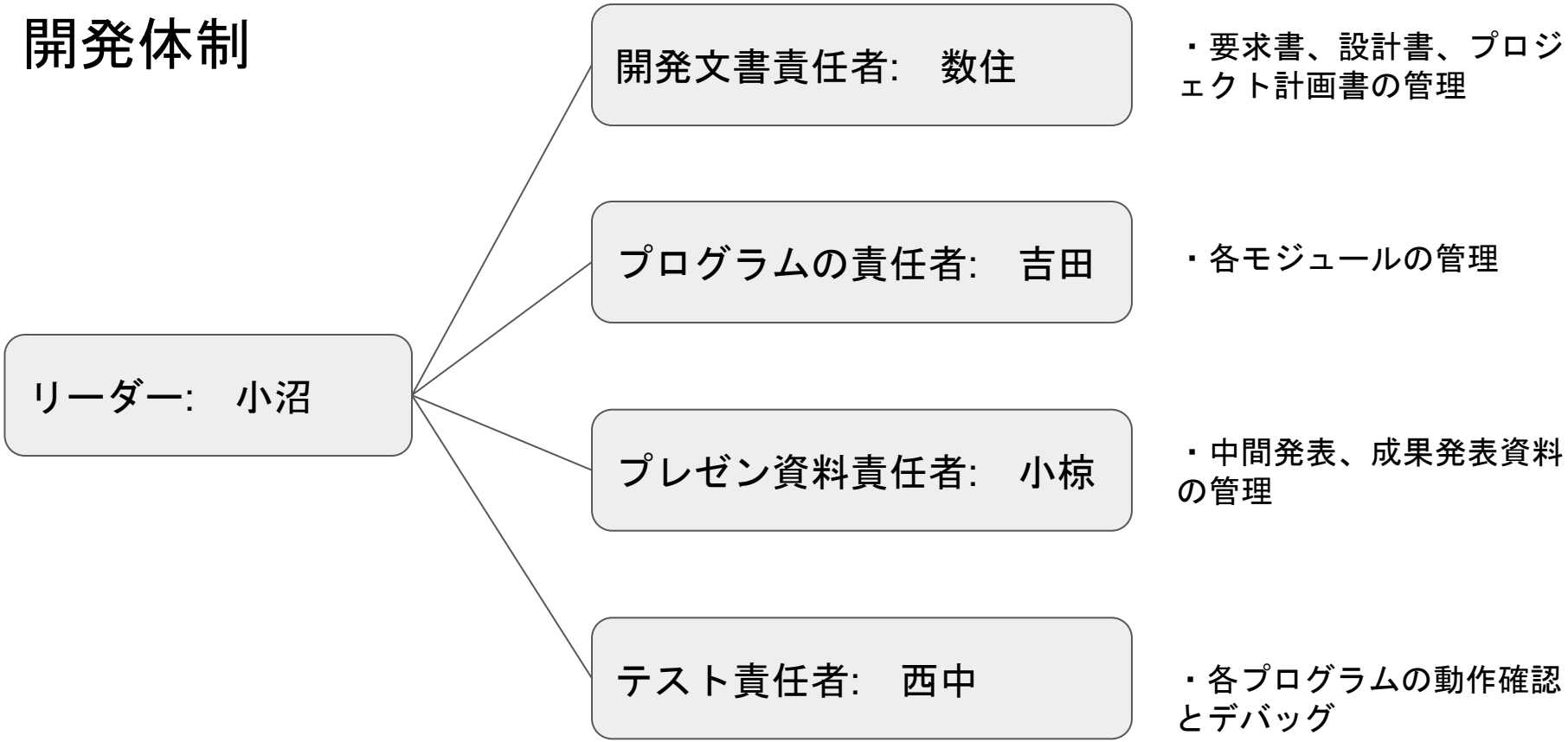
- スプレッドシート管理プログラム
- Remo3からのデータ取得用プログラム
- 人感センサデータ管理用プログラム

＋

- Twitter 用プログラム
- テレビ操作用プログラム
- 照明操作用プログラム
- 睡眠時間計算用プログラム
- 二度寝検知プログロム

3. プロジェクト計画

開発体制


リーダー:　小沼
開発文書責任者:　数住
・要求書、設計書、プロジェクト計画書の管理
プログラムの責任者:　吉田
・各モジュールの管理
プレゼン資料責任者:　小椋
・中間発表、成果発表資料の管理
テスト責任者:　西中
・各プログラムの動作確認とデバッグ

開発スケジュール

タスク	担当者	5/10４限	5/17３限	5/17 ４限	5/24 ３限	5/24 ４限	5/31 ３限	5/31 ４限
要求仕様決定の見直し	全員							
スプレッドシート管理プログラム	小沼							
Remo3からのデータ取得用プログラム	小沼							
人感センサデータ管理用プログラム	数住							
Twitter 用プログラム	数住							
テレビ操作用プログラム	吉田							
照明操作用プログラム	吉田							
睡眠時間計算用プログラム	小椋							
二度寝検知プログロム	小椋							

開発スケジュール

タスク	担当者	5/10４限	5/17３限	5/17 ４限	5/24 ３限	5/24 ４限	5/31 ３限	5/31 ４限
スプレッドシート管理プログラムテスト	小沼 西中							
Remo3からのデータ取得用プログラムテスト	小沼 西中							
人感センサデータ管理用プログラム	数住 西中							
Twitter 用プログラムテスト	数住 西中							
テレビ操作用テスト	吉田 西中							
照明操作用テスト	吉田 西中							
テレビ操作、照明操作のTwitterとの連動テスト	吉田 小椋 西中							
二度寝検知プログラムテスト	小椋 西中							
システムテスト	**全員**							
成果発表資料作成	**全員**							

4. 実際の開発スケジュール

実際の開発スケジュール　赤:遅れ　青:予定通り 緑:遅れて完成

タスク	担当者	5/10４限	5/17３限	5/17 ４限	5/24 ３限	5/24 ４限	5/31 ３限	5/31 ４限
要求仕様決定の見直し	全員							
スプレッドシート管理プログラム	小沼							
Remo3からのデータ取得用プログラム	小沼							
人感センサデータ管理用プログラム	数住							
Twitter 用プログラム	数住							
テレビ操作用プログラム	吉田							
照明操作用プログラム	吉田							
睡眠時間計算用プログラム	小椋							
二度寝検知プログロム	小椋							

タスク	担当者	5/10４限	5/17３限	5/17 ４限	5/24 ３限	5/24 ４限	5/31 ３限	5/31 ４限
スプレッドシート管理プログラムテスト	小沼 西中							
Remo3からのデータ取得用プログラムテスト	小沼 西中							
人感センサデータ管理用プログラムテスト	数住 西中							
Twitter 用プログラムテスト	数住 西中							
テレビ操作用テスト	吉田 西中							
照明操作用テスト	吉田 西中							
テレビ操作、照明操作のTwitterとの連動テスト	吉田 小椋 西中							
二度寝検知プログラムテスト	小椋 西中							
システムテスト	**全員**							
成果発表資料作成	**全員**							

5. プログラム

実装したプログラム

main.gs

CODE.gs（スプレッドシート管理用プログラム）

Tweet.gs（ツイート用プログラム）

POST.gs（テレビ・証明操作用プログラム）

Caluculator.gs（睡眠時間・起床時間等の計算用プログラム）

DoubleSleepPrevention.gs（二度寝防止アルゴリズム）

CODE.gs

（スプレッドシート管理用プログラム）

スプレッドシートidとアクセストークンを取得

スプレッドシートにRemo3から読み込んだデータを書き込む

```
var access_token = '0N7itvZ70SyUGjWB1SXpcUwmn5qKHDuWCdxXdlZ2gdE.v6YInmlb5Au6ALDF-a-fa3I_SSLGA8vVV-mNpx93MQo'//←トークンを記入
var spreadsheetId = '1UE-jKZ9l0riR2bfNO94U1E5bzt8adB80mWMXoUFgLsg'//←スプレッドシートのIDを記入
```

```
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
 );
}//RemoからGet（1/devices）でデータを取得するメソッド（Remoのapi使用）
```

CODE.gs

```
//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
```

Remo3からデータを取得

CODE.gs

Remo3から読み込んだデータを最新のものに更新

```
function getLastData() {
 var datas =
SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').getDataRa
nge().getValues()//logシートをゲットする
 var data = datas[datas.length - 1]
 return {
   date:data[0],
   temp:data[1],
   humi:data[2],
   ill:data[3],
   hu:data[4],
   huTime:data[5],
   row:datas.length,
   beforeDate:datas[datas.length - 2][0]
 }
}
```

CODE.gs

読み込んだデータをスプレッドシートに書き込む

```
function addToSpreadSheet(data, row) {
SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').getRange(row,
1).setValue(new Date())//A列：取得した日時
SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').getRange(row,
2).setValue(data.te)//B列：温度追加
SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').getRange(row,
3).setValue(data.hu)//C列：湿度追加
SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').getRange(row,
4).setValue(data.il)//D列：照度追加
SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').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 =
SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').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){  SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').getRange(row,
5).setValue(1)
 }
 //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 {   SpreadsheetApp.openById(spreadsheetId).getSheetByName('SA').getRange(row,
5).setValue(0)}}
```

	data	temp	humi	ill	hu	人感更新時刻
2	2023/06/02 23:27:43	22.6	80.0	111.0	1	2023-06-02T14:27:00Z
3	2023/06/02 23:28:03	22.6	80.0	111.0	0	2023-06-02T14:27:00Z
4	2023/06/02 23:31:22	22.9	80.0	112.0	1	2023-06-02T14:28:13Z
5	2023/06/02 23:36:22	23.1	80.0	11.0	0	2023-06-02T14:28:13Z
6	2023/06/02 23:41:22	23.3	80.0	11.0	0	2023-06-02T14:28:13Z
7	2023/06/02 23:46:24	23.4	80.0	61.0	0	2023-06-02T14:28:13Z
8	2023/06/02 23:49:06	23.4	80.0	61.0	0	2023-06-02T14:28:13Z
9	2023/06/02 23:51:22	23.5	80.0	49.0	0	2023-06-02T14:28:13Z
10	2023/06/02 23:56:22	23.5	80.0	61.0	0	2023-06-02T14:28:13Z
11	2023/06/02 23:58:02	23.5	80.0	61.0	0	2023-06-02T14:28:13Z
12	2023/06/02 23:59:59	23.5	80.0	48.0	0	2023-06-02T14:28:13Z
13	2023/06/03 0:00:06	23.5	80.0	48.0	0	2023-06-02T14:28:13Z
14	2023/06/03 0:01:22	23.5	80.0	61.0	0	2023-06-02T14:28:13Z
15	2023/06/03 0:02:26	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
16	2023/06/03 0:03:03	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
17	2023/06/03 0:06:23	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
18	2023/06/03 0:11:24	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
19	2023/06/03 0:15:39	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
20	2023/06/03 0:16:24	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
21	2023/06/03 0:19:25	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
22	2023/06/03 0:21:23	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
23	2023/06/03 0:23:03	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
24	2023/06/03 0:23:57	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
25	2023/06/03 0:24:49	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
26	2023/06/03 0:25:14	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
27	2023/06/03 0:25:21	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
28	2023/06/03 0:25:59	23.7	80.0	61.0	0	2023-06-02T14:28:13Z
29	2023/06/03 0:26:22	23.7	80.0	61.0	0	2023-06-02T14:28:13Z

Tweet.gs（ツイート用プログラム）

・Twitterの認証（Google App Script 及び Googleアカウントと連携）する関数

・起床、就寝、二度寝したことをツイートする関数

Tweet.gs

```
function GoodMorningTweet() {
  var year = new Date().getFullYear();
  var month = new Date().getMonth()+1;
  var date = new Date().getDate();
  var hour = new Date().getHours();
  var min = new Date().getMinutes();
  // 配列
var arr = [ "おはよう", "おきたぜ", "おきたよ", "GoodMorning","おい！お前ら朝だぞ！" ] ;

// 配列からランダムで値を選択
var ohayo = arr[ Math.floor( Math.random() * arr.length ) ] ;
  var payload = {
    text: year + "年" + month + "月" + date + "日" + hour + ":" + min + "¥n" + ohayo
  }
```

現在の時刻を取得

ランダムでツイートする
文の候補の配列

ツイートする文を候補から
ランダムで決定する

実際にツイートする文

Tweet.gs

```
  var service = getService();
  if (service.hasAccess()) {
    var url = `https://api.twitter.com/2/tweets`;
    var response = UrlFetchApp.fetch(url, {
      method: 'POST',
      'contentType': 'application/json',
      headers: {
        Authorization: 'Bearer ' + service.getAccessToken()
      },
      muteHttpExceptions: true,
      payload: JSON.stringify(payload)
    });
    var result = JSON.parse(response.getContentText());
    Logger.log(JSON.stringify(result, null, 2));
  } else {
    var authorizationUrl = service.getAuthorizationUrl();
    Logger.log('Open the following URL and re-run the script: %s',authorizationUrl);
  }
}
```

Tweet.gs

```
function GoodNightTweet() {
  var year = new Date().getFullYear();
  var month = new Date().getMonth()+1;
  var date = new Date().getDate();
  var hour = new Date().getHours();
  var min = new Date().getMinutes();
    // 配列
var arr = [ "おやすみ", "いい夢見ろよ！", "この時刻をもちまして誠に勝手ながら就寝させていただきます",
 "Good Night!","すーぱーおふとんすやすやたいむ突入！" ] ;

// 配列からランダムで値を選択
var oyasumi = arr[ Math.floor( Math.random() * arr.length ) ] ;
  var payload = {
    text: year + "年" + month + "月" + date + "日" + hour + ":" + min + "¥n" + oyasumi
  }
```

現在の時刻を取得

ランダムでツイートする
文の候補の配列

実際にツイートする文

Tweet.gs

```
function NidoneTweet() {
  var year = new Date().getFullYear();
  var month = new Date().getMonth()+1;
  var date = new Date().getDate();
  var hour = new Date().getHours();
  var min = new Date().getMinutes();
  var payload = {
    text: year + "年" + month + "月" + date + "日" + hour + ":" + min + "¥n" + '二度寝していま
す...zzz'
  }
```

POST.gs（テレビ・証明操作用プログラム）

```
function TV_On(){
   const REMO_ACCESS_TOKEN = "0N7itvZ70SyUGjWB1SXpcUwmn5qKHDuWCdxXdlZ2gdE.v6YInmlb5Au6ALDF-a-fa3I_SSLGA8vVV-mNpx93MQo"
   //取得したsignalIDをURL内に指定(signalsの後に書かれている)
    var url = "https://api.nature.global/1/signals/40774c63-6944-4eff-8cb0-293310837ee3/send"
    //headerの指定
  const headers = {
    "Content-Type" : "application/json;",
    'Authorization': 'Bearer ' + REMO_ACCESS_TOKEN,
  };
    //実行命令(今回はpost)を指定
  const options = {
    "method" : "post",
    "headers" : headers,
  };
    var reply = UrlFetchApp.fetch(url, options);
```

GetData.gs

```
}
```

```
function GetDeviceId(){
   const REMO_ACCESS_TOKEN =
"0N7itvZ70SyUGjWB1SXpcUwmn5qKHDuWCdxXdlZ2gdE.v6YInmlb5Au6ALDF-a-
fa3I_SSLGA8vVV-mNpx93MQo"
//apiのURLを設定
    var url = "https://api.nature.global/1/" + "appliances"
//optionで呼び出すheadersを設定
  const headers = {
    "Content-Type" : "application/json;",
    'Authorization': 'Bearer ' + REMO_ACCESS_TOKEN,
  };
//実行する命令の指定
  const options = {
    "method" : "get",
    "headers" : headers,
  };
    var reply = UrlFetchApp.fetch(url, options);
//remoに登録されているすべての機器情報を表示
    Logger.log(reply)
}
```

Caluculator.gs（睡眠時間・起床時間等の計算用プログラム）

・時、分、秒ごとに起床時間から就寝時間を引く

```
function sleepTimeCalculator(getUpTime,goBedTime)
//起きた時間、寝た時間
{
  sleepTime = (getUpTime - goBedTime) / 1000; // 差分をミリ秒から秒に変換

  //睡眠時間を時、分、秒に分割
  var sleepTimeHour = Math.floor(sleepTime/3600);
  var sleepTimeMinute = Math.floor((sleepTime-(sleepTimeHour*3600))/60);
  var sleepTimeSecound = sleepTime-(sleepTimeHour*3600)-(sleepTimeMinute*60);

  return sleepTimeHour+ ":" + sleepTimeMinute + ":" + sleepTimeSecound;
}
```

main.gs

・就寝の処理

1. スプレッドシートに起床時間と就寝時間が記入されている、就寝予定時間より後か確認
2. 人感センサーが反応していない、人感センサーの反応から13分たったか確認
3. 就寝時間記録
4. お休みツイート

main.gs

・起床の処理

1. スプレッドシートに起床時間が記入されていない、就寝時間が記入されている、起床予定時間より後か確認
2. テレビとライトをつける
3. 人感センサーが反応しているか確認
4. 起床時間記入
5. おはようツイート

main.gs

・二度寝防止の処理

1. 起床の分岐に入っているとき、
2. 人感センサーが反応していない、起床予定時間から5分以上たったか確認
3. テレビの音量を上げる
4. 二度寝しているツイート

main.gs

・計算結果表示

1. 起床時間と就寝時間が記録されているかつ睡眠時間が表示されていないか確認

2. 計算結果記録

main.gs

・予定時間を超えているか確認

1. 現在時(hour)が予定時(hour)を超えているならtrue
2. 現在時(hour)が予定時(hour)を超えていないか確認
3. 現在日(day)が予定日(day)を超えているならtrueそうでないなら　　false
4. 現在分(minute)が予定分(minute)以上の時true、そうでないならfalse

睡眠開始予定時刻（hour）	睡眠開始予定時刻（minute）	起床予定時刻（hour）	起床予定時刻（minute）
21	30	9	30
睡眠時刻	起床時刻	睡眠時間	
2023/05/30 23:57:00	2023/05/31 6:00:00	6:03:00	
2023/06/02 23:03:23	2023/06/02 23:46:25	0:43:1.6489999999998872	
2023/06/02 23:49:08	2023/06/02 23:51:24	0:2:16.298000000000002	
2023/06/02	2023/06/02	0:1:38.656000000000006	
2023/06/03	2023/06/03	0:00:07	
2023/06/03	2023/06/03	0:00:08	
2023/06/03	2023/06/03	0:00:06	
2023/06/03	2023/06/03	0:01:00	
2023/06/03	2023/06/03	0:00:31	
2023/06/03	2023/06/03	0:00:05	
2023/06/03	2023/06/03	0:00:11	
2023/06/03	2023/06/03	0:00:10	
2023/06/03	2023/06/03	0:00:09	
2023/06/03	2023/06/03	0:00:06	
2023/06/03	2023/06/03	0:25:49.02500000000009	
2023/06/03	2023/06/03	0:00:07	
2023/06/03	2023/06/03	2:38:30	
2023/06/03	2023/06/03	4:55:01	

main.gs

```
/*起きるときの処理*/
  if(logData[0]!="" && logData[1]=="" && CheckTime(parseInt(nowDate.getDate()), parseInt(nowHour),
parseInt(nowMinute), parseInt(plannedDate.getDate()), parseInt(plannedData[2]), parseInt(plannedData[3])) ==
true) //就寝時刻は書かれているが起床時刻が書かれていないかつ起床予定時間より後になったら
  {
    Light_On(); //ライトつく
    //TV_On(); //テレビつく
    if(nowTime - getUpTime > 5 && remoData.hu==0) //起きる時間から５分以上経過したかつ人感センサーの反応がないとき
    {
        Volume_up();
        NidoneTweet();
    }
    else if(remoData.hu==1)
    {
      SpreadsheetApp.openById(spreadsheetId).getSheetByName('SleepManager').getRange(writeDataRow-1,
2).setValue(new Date()); //現在日時記録
      GoodMorningTweet(); //おはようツイート
    }
  }
```

main.gs

```
/*寝るときの処理*/
  if(logData[0] == "睡眠時刻" || logData[0]!="" && logData[1]!="" && CheckTime(parseInt(nowDate.getDate()), parseInt(nowHour), parseInt(nowMinute), parseInt(new Date(logData[0]).getDate()), parseInt(plannedData[0]), parseInt(plannedData[1])) == true) //前回の起床時刻が書かれているかつ睡眠開始予定時間より後になったら
  {
    if(remoData.hu == 0 && nowDate.getTime()-huSensorTime > 780000) //人感センサーが反応したかつ反応してから13分経ったら
    {
      SpreadsheetApp.openById(spreadsheetId).getSheetByName('SleepManager').getRange(writeDataRow, 1).setValue(new Date()); //現在日時記録
      GoodNightTweet(); //おやすみツイート
    }
  }
```

main.gs

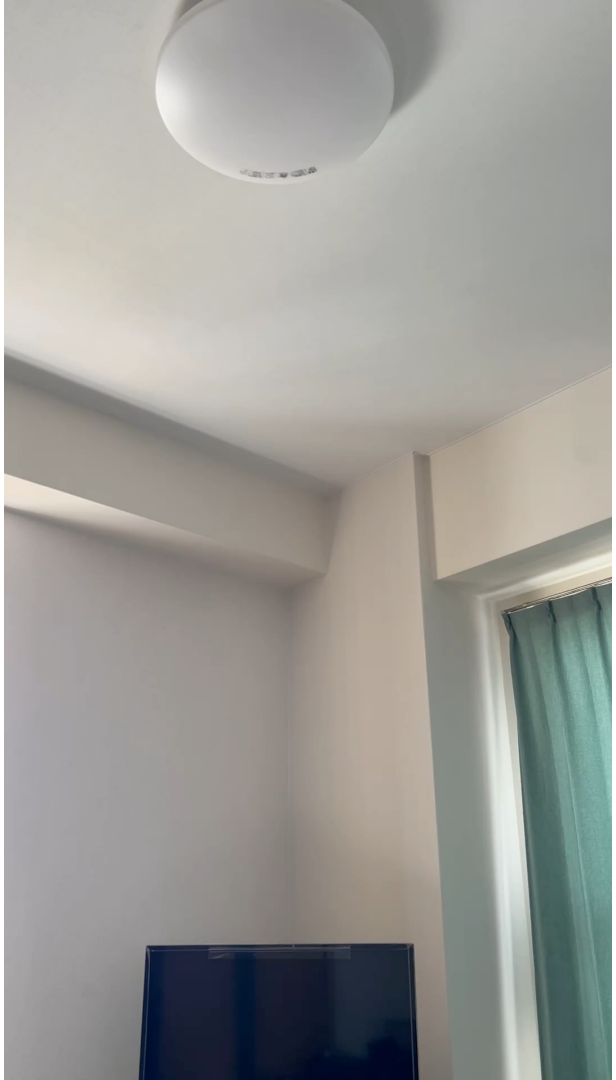
```
/*睡眠時間計算結果表示*/
  if(logData[0]!="" && logData[1]!="" && logData[2] == "")
  {
    var sleepingTime = sleepTimeCalculator(new Date(logData[1]).getTime(),new Date(logData[0]).getTime());
    SpreadsheetApp.openById(spreadsheetId).getSheetByName('SleepManager').getRange(writeDataRow-1,
3).setValue(sleepingTime); //睡眠時間記録
  }
```

main.gs

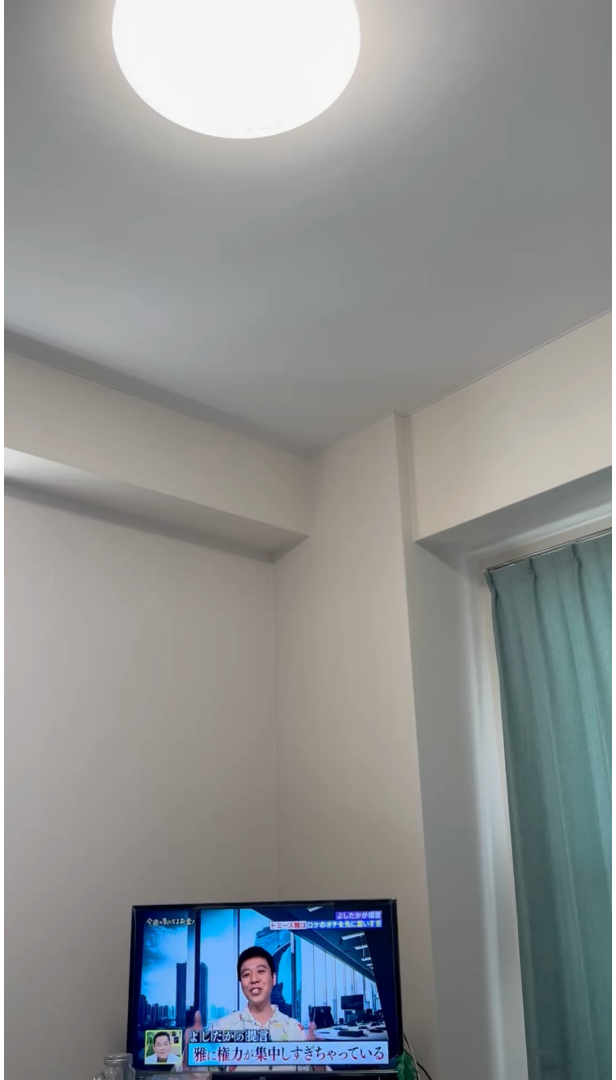
```
function CheckTime(nowDate, nowHour, nowMinute, plannedDate, plannedHour, plannedMinute) //予定時刻を超えたか確認
(現在日,現在時,現在分,前回プログラムが動いた日,予定時,予定分)
{
  if(nowHour>plannedHour) //予定時を超えた時
    return true;
  else if(nowHour<plannedHour) //予定時を超えていないとき
    if(nowDate>plannedDate) //日付が直前に記録された睡眠日を超えているとき
      return true;
    else
      return false;

  else if(nowMinute>=plannedMinute) //予定分以上の時
    return true;
  else
    return false;
}
```

実際に実行してみる

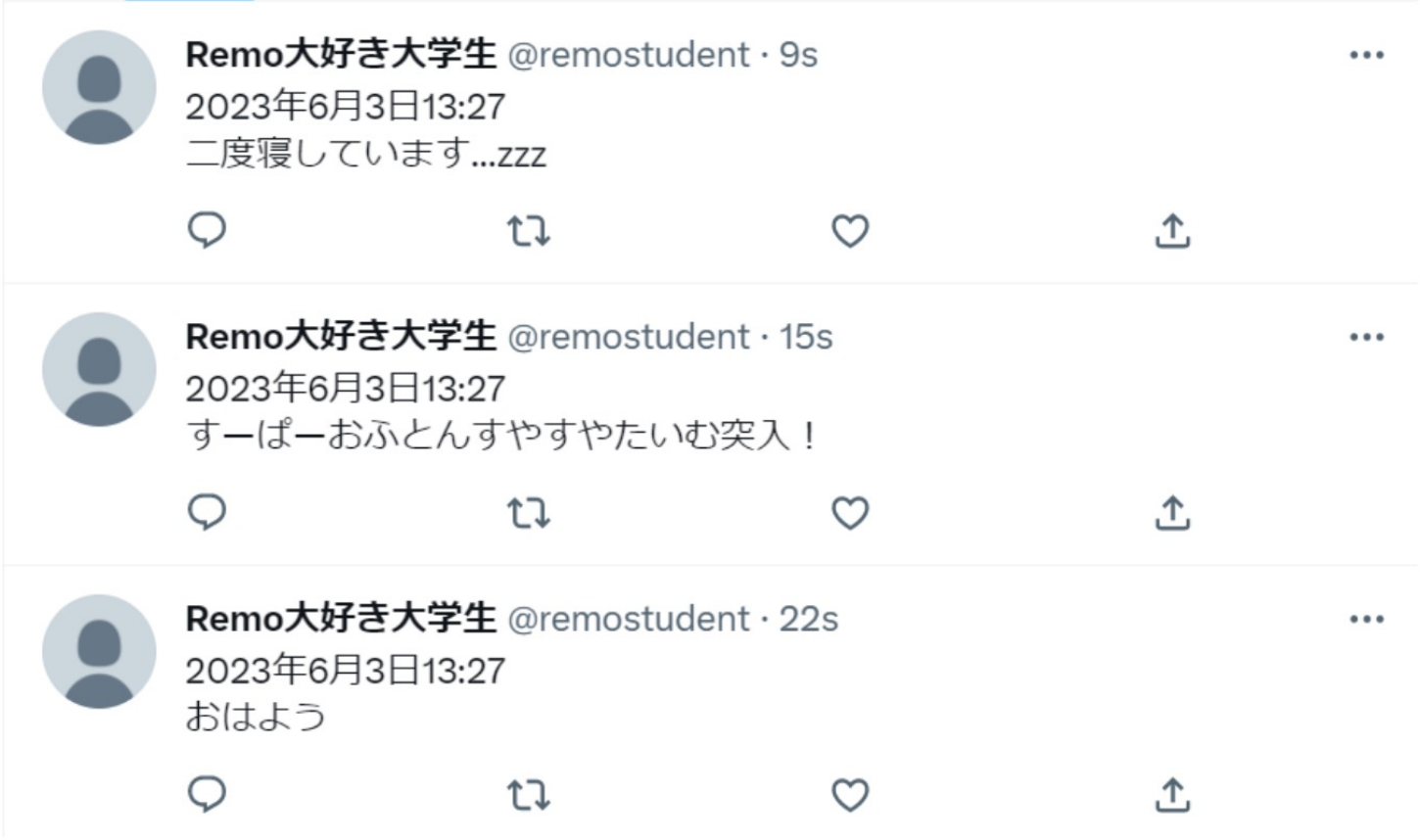
TV_On() Light_On()



Volume_up



実際につぶやかれたTwitterの様子



6. 感想

一つ一つのプログラムを作成するのはあまり難しいことではなかったが、

それらの依存関係を考慮してプログラムの実装を行ったり、

作ったすべてのプログラムが動くようにプログラムを書くのは難しかった。

参考文献

https://prtn-life.com/blog/gas-twitter-api

https://qiita.com/utsuki/items/19e7199bc46a76b4a68c

ご清聴ありがとうございました