Documentação do script de automatização do 7GHz

Edison Neto

4 de Fevereiro de 2019

Conteúdo

1 Introdução

O script está escrito em Javascript(ECMAScript), usando a biblioteca do TheSkyX. Características mais recentes do Javascript, como programação funcional, ou definição de classes, não estão presentes no SkyX.

2 Funcionamento geral

A rotina está dentro de um loop infinito que fica pegando o horário atual do computador e comparando com os horários pré-determinados para iniciar algum dos processo.

Antes do início do loop são definidos os horários (UT) para ligar, fazer o flip e desligar. Para o início, é verificada se a hora é exatamente a do horário de inicialização. Antes de fazer o Slew, é necessário usar a função FindHome. Como não há uma forma de saber se o telescópio já fez o home ou não, a função FindHome deve ser executada sempre na inicialização. O flip, como a inicialização, é realizado precisamente no horário determinado. O desligamento ocorre se o tracking estiver sendo realizado e se a hora atual for maior ou igual a hora de desligamento.

Se a conexão for perdida há a possibilidade dela ser recuperada e que o telescópio volte a sua rotina normal. Entretanto, o problema causado pela perda de conexão pode não ser resolvido, e há a possibilidade de que seja necessária um reconexão manual.

A documentação das funções usa o padrão JSDoc

3 Funções do script

Algumas funções foram escritas com tratamento de erro, e com finalidade mais relacionada à automatização do 7GHz.

3.1 Funções de utilidade geral

3.1.1 Documentando a atividade no log e no debugger

Para documentar a atividade do posicionador no log e no debugger com o horário de execução da atividade.

```
1  function WriteLog(text)
2  {
3    var filename = setFileName();
4    try {
5      TextFile.openForAppend(filename);
6      var horario = getHorario();
7      TextFile.write(text + " " + horario + "\n");
8      print(text + " " + horario);
9      TextFile.close();
10   } catch (texterr) {
```

```
11      print("Erro ao editar o log");
12    }
13  }
```

3.1.2 Escrevendo no debugger e no RunJavaScript

Para escrever no debugger e na janela RunJavaScript a mesma mensagem em tempo de execução.

```
1  function PrintAndOut(text)
2  {
3    print(text);
4    RunJavaScriptOutput.writeLine(text);
5  }
```

3.1.3 Pegando a ascensão reta e a declinação do objeto

Pega a ascensão reta e a declinação de algum objeto qualquer dentro do limite preestabelecido.

```
1  function GetRADec(object)
2  {
3    if (!Sky6IsConnected()) {
4      WriteLog("Erro de conexao tentando executar a funcao GetRADec "
           );
5      return false;
6    }
7
8    try {
9      sky6StarChart.Find(object);
10   } catch (finderr) {
11     WriteLog("Erro durante o find.\n" + finderr.message + " ");
12     return false;
13   }
14
15   sky6ObjectInformation.Property(54);
16   var targetRA = sky6ObjectInformation.ObjInfoPropOut;
17   sky6ObjectInformation.Property(55);
18   var targetDec = sky6ObjectInformation.ObjInfoPropOut;
19
20   return {"ra": targetRA, "dec": targetDec};
21 }
```

3.2 Funções de controle

3.2.1 Conexão

Inicia a conexão entre o SkyX e a montagem e cria o arquivo de log do dia. Essa função deve ser executada quando o SkyX não está conectado e for exatamente 11:00(UT), ou o horário escolhido para o início da execução. Esse processo é muito rápido comparado com outras operações de controle, demorando não mais de 1 segundo.

```
function Connect_c()
{
  var time = getTimeNow();
  var formattedTime = getFormattedTime();

  print("Conectado as " + formattedTime);
  ConnectTelescope();

  var filename = setFileName();
  TextFile.createNew(filename);
  TextFile.write(String(time.day) + "/" + String(time.month) + "/"
      + String(time.year) + "\n");
  TextFile.write("Conectado as " + formattedTime + "\n");
  TextFile.close();
}
```

3.2.2 Inicialização

Inicia o rastreamento do sol. Essa função dever ser executada quando o SkyX está conectado e for exatamente 11:00(UT). Pelo fato do processo de conexão ser muito rápido, não há necessidade de iniciar 1 ou 2 minuto(s) depois da conexão.

```
function Initialize_c()
{
  sky6RASCOMTele.FindHome();
  var propriedade = GetRADec("Sun");

  WriteLog("Iniciou o slew as")
  SlewTelescopeTo(propriedade.ra, propriedade.dec, "Sun");

  WriteLog("Iniciou o rastreamento as");
}
```

3.2.3 Flip

Faz o flip, basicamente reiniciando o rastreamento. A única diferença de código entre a função Initialize_c, é a inutilidade da função FindHome, visto que sua execução somente é necessária uma única vez por dia (desconsiderando problemas de conexão).

```
function Flip_c()
{
  var propriedade = GetRADec("Sun");
  WriteLog("Iniciou o slew as");
  SlewTelescopeTo(propriedade.ra, propriedade.dec, "Sun");

  WriteLog("Completou o flip as");
}
```

3.2.4 Desligamento

Desliga o rastreamento primeiro e depois vai para a posição de parking, desconectando logo em seguida. É executada quando o SkyX está conectado e já passou das 20:00(UT).

```
function TurnOff_c()
{
  SetTelescopeTracking(0, 1, 0, 0);
  WriteLog("Desligou o rastreamento as");

  ParkTelescope();
  WriteLog("Desconectado as");
}
```

3.2.5 Reconexão

Reconecta o telescópio e reinicia o rastreamento. Também executa a função FindHome, já que se o script for (re)iniciado depois das 11:00, este processo pode não ter sido realizado. É executada quando o SkyX não está conectado e a hora atual está entre o horário de execução.

```
function Reconnect_c()
{
  WriteLog("(Re)conectando as");
  ConnectTelescope();
  sky6RASCOMTele.FindHome();

  if (sky6RASCOMTele.IsTracking == 0) {
    RestartTracking_c();
  }
}
```

3.2.6 Reinicialização do tracking

Reinicia o rastreamento. É executada na função de reconexão e quando o SkyX está conectado, não está fazendo o tracking e a hora atual está entre o horário de execução.

```
function RestartTracking_c()
{
  var propriedade = GetRADec("Sun");

  WriteLog("Iniciou o slew as");
  SlewTelescopeTo(propriedade.ra, propriedade.dec, "Sun");

  WriteLog("Reiniciou o rastreamento as");
}
```

3.2.7 Calibração

Aponta para o céu somando 20 graus na altitude da observação atual. A calibração é feita duas vezes ao dia, uma hora antes do flip e uma hora depois.

```
function CalibrateTelescope_c()
{
    WriteLog("Calibracao iniciada as")
    var delta = 20;
    var props = GetAzAlt();
```

```
6       var newAlt = props.alt + delta;
7       SlewTelescopeToAzAlt(props.az, newAlt, "");
8   }
```

4 Guia de estilo do código

4.1 Indentação

O código usa espaços com 4 caracteres de largura.

4.2 Posicionamento das chaves

A forma correta é colocar a chave de abertura por último na linha, e colocar a chave de fechamento primeiro.

```
1  if (something === true) {
2    print(something);
3  }
```

Para funções coloque a chave embaixo da próxima linha.

```
1  function myFunction()
2  {
3    return 0;
4  }
```

4.3 Nomeando funções

As funções que usam alguma classe do SkyX são nomeadas usando UpperCamelCase, já as que não usam são nomeadas usando lowerCamelCase. As função principais de controle, são nomeadas usando UpperCamelCase e com o sufixo '_c'.

5 Biblioteca do SkyX

5.1 Principais classes

- sky6ObjectInformation - Informações dos objetos
- sky6StarChart - Acesso aos aspectos visuais do SkyX
- sky6RASCOMTele - Controle físico da montagem
- TextFile - Manipulação de arquivos

5.2 Exemplos

5.2.1 sky6ObjectInformation

Dá o acesso ao banco de dados do SkyX. Podendo pegar diversas informações sobre o objeto sendo observado.

Property()

Argumentos:

- *number* number - Representa um certa informação do objeto.

Há um total de 189 informações separadas nessa função. A função em si não retorna nada, o valor fica armazenado na propriedade *ObjInfoPropOut*, como escrito no exemplo.

Exemplo:

```
1 sky6ObjectInformation.Property(55);
2 print(sky6ObjectInformation.ObjInfoPropOut);
```

As duas propriedades que são usadas no script, são as seguintes:

- 54: Ascensão reta
- 55: Declinação

5.2.2 sky6StarChart

Controle da parte visual do SkyX. Basicamente com essa classe é possível fazer o que se faria clicando no TheSkyX.

Find()

Argumentos:

- *string* objectName - O nome do objeto a ser procurado.

Procura pelo objeto dado.

Exemplo:

```
1 sky6StarChart.Find("Sun");
```

Exemplo usando o sky6ObjectInformation.Property():

```
1 // Procura pelo sol.
2 sky6StarChart.Find("Sun");
3 // Prepara no ObjInfoPropOut o valor da declinacao.
4 sky6ObjectInformation.Property(55);
5
6 print(sky6ObjectInfomation.ObjInfoPropOut + "\n")
```

5.2.3 sky6RASCOMTele

Dá o controle físico da montagem. Para coisas como o slew ou parking.

Connect(void)

Inicia a comunicação entre a montagem e o SkyX.

Disconnect(void)

Termina a conexão entre a montagem e o SkyX.

Abort(void)

Aborta a operação que estiver sendo realizada.

SlewToRaDec()

Argumentos:

- *number* TargetRa - A ascensão reta;
- *number* TargetDec - A declinação;
- *string* targetObject - O nome do objeto.

Aponta o telescópio para a coordenada dada.

```
1  sky6RASCOMTele.Connect();
2
3  var targetObject = "Sun";
4  sky6StarChart.Find(targetObject);
5
6  sky6ObjectInformation.Property(54);
7  var targetRA = sky6ObjectInformation.ObjInfoPropOut;
8
9  sky6ObjectInformation.Property(55);
10 var targetDec = sky6ObjectInformation.ObjInfoPropOut;
11 sky6RASCOMTele.SlewToRaDec(targetRa, TargetDec, targetObject);
```

GetRaDec(void)

Pega a declinação e a ascensão reta atual, e prepara os valores nas variáveis dRa e dDec.

Exemplo:

```
1  sky6RASCOMTele.Connect();
2  sky6RASCOMTele.GetRaDec();
3
4  print(String(sky6RASCOMTele.dRa) + " | " + String(sky6RASCOMTele.
       dDec));
```

Park(void)

Faz o slew para a posição de parking, e finaliza a conexão com o TheSky6.

ParkAndDoNotDisconnect(void)

Tem quase o mesmo funcionamento que a função 'Park'. A diferença é que essa função não finaliza a conexão entre o telescópio e o TheSky6. Para fazer outro Slew depois de usar esta função é necessário utilizar a função 'Unpark' antes.

Unpark(void)

Tira o telescópio do estado de parked.

IsConnected

Tem o valor zero se o telescópio não estiver conectado.

IsParked

Tem o valor zero se o telescópio não estiver na posição de parking.

IsTracking

Tem o valor zero se o telescópio não estiver fazendo o tracking.

dRa

A ascensão reta atual.

dDec

A declinação atual.

5.2.4 TextFile

Classe usada para manipulação básica de arquivos.

createNew()

Cria um arquivo txt. Os arquivos são obrigatoriamente na pasta */Meus Documentos/Software Bisque/TheSkyX Professional Edition/ScriptFiles*, o local dos não pode ser modificado.O nome do arquivo só pode conter letras e números.

Argumentos:

- *string* filename - O nome do arquivo (sem a extensão).

write()

Escreve uma string no arquivo.
Argumentos:

- *string* text - String a ser escrita no arquivo.

openForAppend()

Abre o arquivo de forma a anexar novos conteúdos depois do que já está escrito. Caso o arquivo não tiver sido criado e essa função for usada, o arquivo será criado.
Argumentos:

- *string* filename - O nome do arquivo

close(void)

Fecha o arquivo salvando as modificações feitas.

5.2.5 Não relacionadas com as classes

String()

Argumentos:

- *number* int - Uma variável numérica qualquer.

Transforma um número em uma string. Essa função funciona como o método toString() do javascript(que curiosamente não funciona no SkyX).
Exemplo:

```
1 var int = 2;
2 print(String(int));
```

print()

Argumentos:

- *(string, number)* text - Uma variável qualquer.

Essa função escreve nos logs do debugger. Ela escreve enquanto o programa roda, diferentemente da variável Out.

A variável Out

Essa variável armazena tudo que será escrito na tela do "Run Java Script"no SkyX. Ela só é escrita quando o script acaba de rodar. É possível escrever nessa tela em tempo de execução com a classe RunJavaScriptOutput.

6 O código

6.1 Versão 1.4

```
1  /*
2   * Version: 1.4 12/07/18
3   */
4
5  /**
6   * Confirma se o script tem conexao com o telescopio.
7   *
8   * @returns {boolean} false se nao estiver conectado.
9   *                    true se estiver conectado.
10  */
11 function Sky6IsConnected()
12 {
13     if (sky6RASCOMTele.IsConnected == 0) {
14         return false;
15     }
16     return true;
17 }
18
19 /**
20  * Estabiliza a conexao com o telescopio.
21  *
22  * @returns {boolean} false caso algum erro aconteca.
23  */
24 function ConnectTelescope()
25 {
26     try {
27         sky6RASCOMTele.Connect();
28     } catch (connerr) {
29         WriteLog("Erro de conexao com a montagem\n" + connerr.
                message + " ");
30         return false;
31     }
32
33     return true;
34 }
35
36 /**
37  * Liga o tracking para um lugar especifico, ou desliga o tracking.
38  *
39  * @param {number} IOn - O numero que desliga ou liga o tracking.
40  *                               0 - desliga
41  *                               1 - liga
42  *
```

```
43  * @param {number} IIgnoreRates - O numero que especifica se e para
        o
44  *                                telescopio usar a taxa de
        tracking atual.
45  *                              0 - Ignora os valores de dRaRate e
        dDecRate
46  *                              1 - Usa os valores de dRaRate e
        dDecRate
47  *
48  * @param {number} dRaRate - Especifica a ascensao reta a ser usada
        .
49  *                           So e utilizada se IIgnoreRates for
        igual a 1.
50  *
51  * @param {number} dDecRate - Especifica a declinacao a ser usada.
52  *                           So e utilizada se IIgnoreRates for
        igual a 1.
53  *
54  * @returns {boolean} false se a montagem nao estiver conectada.
55  */
56 function SetTelescopeTracking(IOn, IIgnoreRates, dRaRate, dDecRate)
57 {
58     if (!Sky6IsConnected()) {
59         return false;
60     }
61
62     sky6RASCOMTele.SetTracking(IOn, IIgnoreRates, dRaRate, dDecRate
        );
63     var Out = "RA Rate = " + sky6RASCOMTele.dRaTrackingRate;
64     Out += " | Dec Rate = " + sky6RASCOMTele.dDecTrackingRate;
65     PrintAndOut(Out);
66     return true;
67 }
68
69 /**
70  * Faz o slew para um determinado objeto dados sua ascensao reta e
        declinacao.
71  *
72  * @param {number} dRa - ascensao reta.
73  * @param {number} dDec - declinacao.
74  * @param {string} targetObject - Objeto em questao.
75  *
76  * @returns {boolean} true se tudo tiver ocorrido corretamente.
77  */
78 function SlewTelescopeToRaDec(dRa, dDec, targetObject)
79 {
80     if (!Sky6IsConnected()) {
81         PrintAndOut("Telescopio nao conectado.");
82         return false;
83     }
84
85     try {
86         sky6RASCOMTele.SlewToRaDec(dRa, dDec, targetObject);
87         return true;
88     } catch (slewerr) {
89         WriteLog("Falha durante o slew\n" + slewerr.message);
90         return false;
```

```
91      }
92  }
93
94  /**
95   * Faz o slew para um determinado objeto dados azimute e altitude.
96   *
97   * @param {number} az - Azimute.
98   * @param {number} alt - Altitude.
99   * @param {string} targetObject - Objeto em questao.
100  *
101  * @returns {boolean} true se tudo tiver ocorrido corretamente.
102  */
103 function SlewTelescopeToAzAlt(az, alt, targetObject)
104 {
105     if (!Sky6IsConnected()) {
106         PrintAndOut("Telescopio nao conectado.");
107         return false;
108     }
109
110     try {
111         sky6RASCOMTele.SlewToAzAlt(az, alt, targetObject);
112         return true;
113     } catch (slewerr) {
114         WriteLog("Falha durante o slew\n" + slewerr.message);
115         return false;
116     }
117 }
118
119 /**
120  * Leva o telescopio para a posicao de parking.
121  *
122  * @returns {boolean} true se tudo tiver ocorrido corretamente.
123  */
124 function ParkTelescope()
125 {
126     if (!Sky6IsConnected()) {
127         return false;
128     }
129
130     if (sky6RASCOMTele.IsParked != 0) {
131         sky6RASCOMTele.Park();
132         WriteLog("Parking finalizado as");
133         return true;
134     }
135 }
136
137 /**
138  * Prcoura pelo objeto dado e pega a ascensao reta e a declinacao
        dele
139  * no momento.
140  *
141  * @param {string} object - Nome do objeto a ser encontrado.
142  * @returns {object} Um objeto com a ascensao reta e a declinacao.
143  */
144 function GetRADec(object)
145 {
146     if (!Sky6IsConnected()) {
```

```
147         WriteLog("Erro de conexao tentando executar a funcao
                GetRADec");
148         return false;
149     }
150
151     try {
152         sky6StarChart.Find(object);
153     } catch (finderr) {
154         WriteLog("Erro durante o find.\n" + finderr.message + " ");
155         return false;
156     }
157
158     sky6ObjectInformation.Property(54);
159     var targetRA = sky6ObjectInformation.ObjInfoPropOut;
160     sky6ObjectInformation.Property(55);
161     var targetDec = sky6ObjectInformation.ObjInfoPropOut;
162
163     return {"ra": targetRA, "dec": targetDec};
164 }
165
166 /**
167  * Pega o azimute e a altitude do objeto sendo observado no momento
        .
168  *
169  * @returns {object} Um objeto com o azimute e a altitude.
170  */
171 function GetAzAlt()
172 {
173     if (!Sky6IsConnected()) {
174         WriteLog("Erro de conexao tentando executar a funcao
                GetAzAlt");
175         return false;
176     }
177
178     sky6RASCOMTele.GetAzAlt();
179     var az = sky6RASCOMTele.dAz;
180     var alt = sky6RASCOMTele.dAlt;
181
182     return {"az": az, "alt": alt};
183 }
184
185 /**
186  * Verifica se o telescopio esta apontando para o sol.
187  *
188  * @returns {boolean}
189  */
190 function IsPointingSun()
191 {
192     var sun_props = GetRADec("Sun");
193     var current_props = sky6RASCOMTele.GetRaDec();
194     var current_ra = sky6RASCOMTele.dRa;
195     var current_dec = sky6RASCOMTele.dDec;
196
197     if (sun_props.ra == current_ra && sun_props.dec == current_dec)
            {
198         return true;
199     } else {
```

```
200             return false;
201         }
202     }
203
204     /**
205      * Pega a data e o horario do momento que a funcao e chamada.
206      *
207      * @returns {object} Um objeto com os dados.
208      */
209     function getTimeNow()
210     {
211         var time = new Date();
212         var day = time.getDate();
213         var month = time.getMonth();
214         var year = time.getFullYear();
215         var hour = time.getHours();
216         var minutes = time.getMinutes();
217         var seconds = time.getSeconds();
218
219         return {
220             "day": day,
221             "month": month,
222             "year": year,
223             "hour": hour,
224             "minutes": minutes,
225             "seconds": seconds
226         };
227     }
228
229     /**
230      * Cria o nome do arquivo para o dia atual.
231      *
232      * @returns {string} O nome do arquivo do dia atual.
233      */
234     function setFileName()
235     {
236         var time = getTimeNow();
237
238         if (time.day < 10) {
239             var day = "0" + String(time.day);
240         } else {
241             var day = String(time.day);
242         }
243
244         if (time.month < 10) {
245             var month = "0" + String(time.month);
246         } else {
247             var month = String(time.month);
248         }
249
250         var year = String(time.year);
251
252         var filename = year + month + day;
253
254         return filename;
255     }
256
```

```
257 /**
258  * Pega o horario atual do computador.
259  *
260  * @returns {string} O horario no formato %H:%M:%S.
261  */
262 function getFormattedTime()
263 {
264     var time = getTimeNow();
265     var formattedTime = String(time.hour) + ":" + String(time.
        minutes) + ":" + String(time.seconds);
266     return formattedTime;
267 }
268
269 /**
270  * Escreve no debugger e no log a mesma mensagem, junto com o
       horario
271  * do momento.
272  *
273  * @param {string} text -  A mensagem a ser escrita.
274  */
275 function WriteLog(text)
276 {
277     var filename = setFileName();
278     try {
279         TextFile.openForAppend(filename);
280         var formattedTime = getFormattedTime();
281         TextFile.write(text + " " + formattedTime + "\n");
282         print(text + " " + formattedTime);
283         TextFile.close();
284     } catch (texterr) {
285         PrintAndOut("Erro ao editar o log.\n" + texterr.message);
286     }
287 }
288
289 /**
290  * Escreve no debugger e na janela Run Java Script.
291  *
292  * @param {string} text - O conteudo a ser escrito.
293  */
294 function PrintAndOut(text)
295 {
296     print(text);
297     RunJavaScriptOutput.writeLine(text);
298 }
299
300 /**
301  * Conecta o telescopio e cria o arquivo de log diario.
302  */
303 function Connect_c()
304 {
305     var time = getTimeNow();
306     var formattedTime = getFormattedTime();
307
308     PrintAndOut("Conectado as " + formattedTime);
309     ConnectTelescope();
310
311     var filename = setFileName();
```

```
312     TextFile.createNew(filename);
313     TextFile.write(String(time.day) + "/" + String(time.month) + "/
            " + String(time.year) + "\n");
314     TextFile.write("Conectado as " + formattedTime + "\n");
315     TextFile.close();
316 }
317 
318 /**
319  * Processo de inicializacao.
320  */
321 function Initialize_c()
322 {
323     sky6RASCOMTele.FindHome();
324     var props = GetRADec("Sun");
325 
326     WriteLog("Iniciou o slew as");
327     SlewTelescopeToRaDec(props.ra, props.dec, "Sun");
328 
329     WriteLog("Iniciou o rastreamento as");
330 }
331 
332 /**
333  * Processo do Flip.
334  */
335 function Flip_c()
336 {
337     var props = GetRADec("Sun");
338     WriteLog("Iniciou o slew(flip) as");
339     SlewTelescopeToRaDec(props.ra, props.dec, "Sun");
340 
341     WriteLog("Completou o flip as");
342 }
343 
344 /**
345  * Desliga o tracking e faz o parking.
346  */
347 function TurnOff_c()
348 {
349     SetTelescopeTracking(0, 1, 0, 0);
350     WriteLog("Desligou o rastreamento as");
351 
352     ParkTelescope();
353     WriteLog("Desconectado as");
354 }
355 
356 /**
357  * Reconecta o telescopio e reinicia o tracking.
358  */
359 function Reconnect_c()
360 {
361     WriteLog("(Re)conectado as");
362     ConnectTelescope();
363     sky6RASCOMTele.FindHome();
364     if (sky6RASCOMTele.IsTracking == 0) {
365         RestartTracking_c();
366     }
367 }
```

```
368
369 /**
370  * Reinicia o rastreamento.
371  */
372 function RestartTracking_c()
373 {
374     var props = GetRADec("Sun");
375
376     WriteLog("Iniciou o slew as");
377     SlewTelescopeToRaDec(props.ra, props.dec, "Sun");
378
379     WriteLog("Reiniciou o rastreamento as");
380 }
381
382 /**
383  * Aponta para o ceu.
384  */
385 function CalibrateTelescope_c()
386 {
387     WriteLog("Calibracao iniciada as")
388     var delta = 20;
389     var props = GetAzAlt();
390     var newAlt = props.alt + delta;
391     SlewTelescopeToAzAlt(props.az, newAlt, "");
392 }
393
394 /*
395  * Configura os horario para inicializar, fazer o flip e desligar.
396  */
397 var work_time = {
398     "start_hour": 11,
399     "start_minutes": 00,
400     "flip_hour": 16,
401     "flip_minutes": 00,
402     "turn_off_hour": 20,
403     "turn_off_minutes": 00,
404     "first_calibration_hour": 15,
405     "first_calibration_minutes": 00,
406     "first_calibration_seconds": 00,
407     "second_calibration_hour": 17,
408     "second_calibration_minutes": 00,
409     "second_calibration_seconds": 00,
410     "finish_first_calibration_hour": 15,
411     "finish_first_calibration_minutes": 01,
412     "finish_first_calibration_seconds": 00,
413     "finish_second_calibration_hour": 17,
414     "finish_second_calibration_minutes": 01,
415     "finish_second_calibration_seconds": 00,
416 };
417
418 /**
419  * Verifica se e a hora da primeira calibracao.
420  *
421  * @param {object} time - Horario atual.
422  * @returns {boolean}
423  */
424 function timeToFirstCalibration(time)
```

```
425 {
426     return time.hour == work_time.first_calibration_hour &&
427                 time.minutes == work_time.first_calibration_minutes
                        &&
428                 time.seconds == work_time.first_calibration_seconds
                        ;
429 }
430
431 /**
432  * Verifica se e a hora de voltar para o sol.
433  *
434  * @param {object} time - Horario atual.
435  * @return {boolean}
436  */
437 function timeToFinishFirstCalibration(time)
438 {
439     return time.hour == work_time.finish_first_calibration_hour &&
440                 time.minutes == work_time.
                        finish_first_calibration_minutes &&
441                 time.seconds == work_time.
                        finish_first_calibration_seconds;
442 }
443
444 /**
445  * Verifica se e a hora da segunda calibracao.
446  *
447  * @param {object} time - Horario atual.
448  * @returns {boolean}
449  */
450 function timeToSecondCalibration(time)
451 {
452     return time.hour == work_time.second_calibration_hour &&
453                 time.minutes == work_time.
                        second_calibration_minutes &&
454                 time.seconds == work_time.
                        second_calibration_seconds;
455 }
456
457 /**
458  * Verifica se e a hora de voltar para o sol.
459  *
460  * @param {object} time - Horario atual.
461  * @return {boolean}
462  */
463 function timeToFinishSecondCalibration(time)
464 {
465     return time.hour == work_time.finish_second_calibration_hour &&
466                 time.minutes == work_time.
                        finish_second_calibration_minutes &&
467                 time.seconds == work_time.
                        finish_second_calibration_seconds;
468 }
469
470 /**
471  * Verifica se e a hora de inicializar.
472  *
473  * @param {object} time - Horario atual.
```

```
474  * @returns {boolean}
475  */
476 function timeToInitialize(time)
477 {
478     return time.hour == work_time.start_hour &&
479                 time.minutes == work_time.start_minutes;
480 }
481
482 /**
483  * Verifica se e a hora do flip.
484  *
485  * @param {object} time - Horario atual.
486  * @returns {boolean}
487  */
488 function timeToFlip(time)
489 {
490     return time.hour == work_time.flip_hour &&
491                 time.minutes == work_time.flip_minutes;
492 }
493
494 /**
495  * Verifica se e(ou ja passou) (d)a hora de desligar e se o
        tracking esta ocorrendo.
496  *
497  * @param {object} time - Horario atual.
498  * @returns {boolean}
499  */
500 function timeToTurnOff(time)
501 {
502     return time.hour >= work_time.turn_off_hour &&
503                 sky6RASCOMTele.IsTracking != 0;
504 }
505
506 /**
507  * Verifica se e a hora de iniciar a conexao.
508  *
509  * @param {object} time - Horario atual.
510  * @returns {boolean}
511  */
512 function timeToConnect(time)
513 {
514     return time.hour == work_time.start_hour &&
515                 time.minutes == work_time.start_minutes;
516 }
517
518 /**
519  * Verifica se o telescopio esta desconectado e se esta no horario
        de funcionamento.
520  * Procurar prever um eventual problema de simples desconexao do
        SkyX.
521  *
522  * @param {object} time - Horario atual.
523  * @returns {boolean}
524  */
525 function connectionProblem(time)
526 {
527     return time.hour >= work_time.start_hour &&
```

```
528                 time.hour < work_time.turn_off_hour;
529 }
530
531 /**
532  * Verifica se o telescopio esta no horario de funcionamento, mas
        nao esta
533  * fazendo o tracking.
534  *
535  * @param {object} time - Horario atual.
536  * @returns {boolean}
537  */
538 function checkTracking(time)
539 {
540     return sky6RASCOMTele.IsTracking == 0 && time.hour >= work_time
          .start_hour &&
541                 time.hour < work_time.turn_off_hour;
542 }
543
544 while (true)
545 {
546     var time = getTimeNow();
547
548     if (Sky6IsConnected()) {
549         if (timeToInitialize(time)) {
550             Initialize_c();
551         }
552         else if (timeToFirstCalibration(time)) {
553             CalibrateTelescope_c();
554         }
555         else if (timeToFinishFirstCalibration(time)) {
556             RestartTracking_c();
557         }
558         else if (timeToFlip(time)) {
559             Flip_c();
560         }
561         else if (timeToSecondCalibration(time)) {
562             CalibrateTelescope_c();
563         }
564         else if (timeToFinishSecondCalibration(time)) {
565             RestartTracking_c();
566         }
567         else if (timeToTurnOff(time)) {
568             TurnOff_c();
569         }
570     }
571     else if (timeToConnect(time)) {
572         Connect_c();
573     }
574     else if (connectionProblem(time)) {
575         Reconnect_c();
576     }
577 }
```

6.2 Versão mais recente

https://github.com/3ldr0n/Automatiza-o-7ghz/