Relatório Final

Título do projeto de pesquisa:	Desenvolvimento de um monitor de cintilação ionosférico baseado em GPS
Bolsista:	Igor Bragaia
Orientador(a):	Dr. Alison de Oliveira Moraes
Período a que se refere o relatório:	Agosto de 2017 a Agosto de 2018

Resumo

O monitor de cintilação é uma ferramenta fundamental para estudo da ionosfera, bem como para entender seus efeitos para os usuários de sistemas de navegação via satélite. Os monitores de cintilação profissionais disponíveis no mercado custam entre 3.000 e 20.000 dólares, sem considerar os impostos. O objetivo desta linha de ação é validar um protótipo de um monitor de cintilação ionosférico de baixo custo (menos de 100 dólares) e resolução adequada, capaz de captar dados que possam auxiliar nas pesquisas relacionadas ao tema. Através desse protótipo obteve-se o índice de cintilação S4 por meio de sensores GPS acoplados em conjunto com um computador pessoal. Após uma série de testes com dados de campo, decidiu-se adotar a taxa de amostragem 10 Hz para cada canal GPS. Desenvolveu-se uma rotina computacional que a partir dos dados recebidos por cada canal GPS, formata apropriadamente os dados, obtendo-se em seguida a distribuição temporal do índice de cintilação S4 e exibindo-se gráficos ilustrativos. Por fim, o método foi valido com base em comparações dos resultados com monitores profissionais já consolidados.

1. Introdução

Um monitor ionosférico é um dispositivo físico de medição que tem como finalidade obter dados sobre o estado da ionosfera ao longo de um prolongado período de tempo a uma alta taxa de amostragem, para posterior determinação de índices ionosféricos.

O monitor ionosférico desenvolvido neste realiza medições com o uso de um sensor GPS a uma taxa de amostragem de 10 Hz. Estas medições são no formato NMEA[1] (um conjunto de especificações de dados e elétricas para comunicação de dispositivos eletrônicos de

navegação) armazenadas como arquivo de texto e posteriormente processadas de modo a ser possível a determinação de índices por meio da execução de algoritmos.

Em relação à taxa de amostragem, um monitor ionosférico comercial tipicamente atua a uma taxa de amostragem em média de 50 Hz. No entanto, de acordo com testes preliminares com dados obtidos em monitores em operação do Instituto Nacional de Pesquisas Espaciais (INPE), verificou-se que uma taxa de amostragem de 10 Hz apresenta desempenho satisfatório na determinação de índices que descrevem as condições ionosféricas.

No contexto do sensor GPS, verificou-se a viabilidade do uso do sensor Adafruit (Companhia de Hardware Open-Source) Ultimate GPS Breakout. A escolha deste sensor foi devida ao seu relativo baixo custo, com o objetivo de manter cada protótipo barato; à sua taxa de 10 Hz, que cumpre o pré-requisito de determinação de índices ionosféricos; à possibilidade de configuração através de comunicação serial, visando obtenção de dados necessários para determinação de índices; e à fácil integração do sensor com entrada USB via adaptador USB-UART. Este projeto tem como objetivo o desenvolvimento de um monitor de cintilação que opere em rede e de forma autônoma, para que seja aplicado em larga escala no território brasileiro. Neste relatório serão apresentadas as atividades desenvolvidas para atingir estes objetivos.

2. Materiais e Métodos

Como o requisito operacional deste sistema é conexão em rede e autonomia, optou-se pela adoção da plataforma computacional do tipo Raspberry Pi. Sendo assim, o monitor de cintilação possui os componentes:

- Adaptador conversor USB para UART (R$ 15)
- Sensor GPS Adafruit Ultimate GPS Breakout ($40)
- Computador Raspberry Pi.

O sensor GPS conecta-se diretamente ao adaptador USB-UART de forma que o computador recebe dados seriais via porta USB. O software desenvolvido para recebimento de dados foi feito em Python[2] e encontra-se descrito abaixo.

```
import serial
from datetime import datetime

ser=serial.Serial("COM5",9600)
with open("nmea_data.txt", "a") as text_file:
```

```
while True:
        if ser.inWaiting():
                text_file.write(ser.readline(ser.inWaiting()))
```

O funcionamento do recebimento de dados se dá por meio do sensor GPS, o qual recebe dados dos satélites disponíveis naquele instante, enviando-os via porta serial para o buffer UART e sendo persistidos no computados, conforme sugere o código acima. Os dados recebidos, por sua vez, são formatados no padrão mundial NMEA. O sensor gera 10 quadros como esse abaixo por segundo.

```
$GPGGA,220020.300,2312.1863,S,04552.4492,W,1,06,1.49,574.2,M,-4.2,M,,*74
$GPGSA,A,3,31,22,23,11,01,03,,,,,,,1.77,1.49,0.95*02
$GPGSV,3,1,12,01,67,311,38,22,62,126,15,03,50,196,26,33,46,056,28*72
$GPGSV,3,2,12,23,43,249,30,11,42,326,22,31,29,138,15,26,20,077,18*78
$GPGSV,3,3,12,09,18,273,,16,16,048,,17,04,252,,08,02,354,*7F
$GPZDA,220020.300,10,04,2016,,*57
```

Figura 1 Dado bruto em padrão NMEA.

Os dados, por sua vez, podem ser interpretados da forma seguinte.

`$GPGGA,220020.300,2312.1863,S,04552.4492,W,1,06,1.49,574.2,M,-4.2,M,,*74`
Dados da localização obtida via satélite (*Fix)* Tempo: GMT: 22 horas 0 minutos 20 segundos 300 microssegundos Latitude: 23° 12,1863' S Longitude: 45º 52,4492' O 6 satélites em vista 574.2 metros acima do nível do mar
`$GPGSA,A,3,31,22,23,11,01,03,,,,,,,1.77,1.49,0.95*02`
Dados de erro do satélite (*Dilution of Precision*) e identificadores dos satélites visíveis

`$GPGSV,3,1,12,01,67,311,38,22,62,126,15,03,50,196,26,33,46,056,28*72`
`$GPGSV,3,2,12,23,43,249,30,11,42,326,22,31,29,138,15,26,20,077,18*78`
`$GPGSV,3,3,12,09,18,273,,16,16,048,,17,04,252,,08,02,354,*7F`

Dados dos satélites visíveis: elevação, azimute e intensidade do sinal para cada satélite
`$GPZDA,220020.300,10,04,2016,,*57`
Dados de tempo de aquisição. Tempo GMT: 22 horas 0 minutos 20 segundos 300 microssegundos Data: 10 de abril de 2016

Figura 2 Interpretação do dado bruto em padrão NMEA.

Em seguida, a partir dos dados brutos no padrão NMEA persistidos localmente no computador, pôde-se programar uma rotina em Python que a partir dos dados, extrai informações a respeito do comportamento ionosférico e da ocorrência do fenômeno de

cintilação ionosférica. Assim, inicialmente, formata-se o dado bruto de forma a isolar-se cada satélite e as condições de captura dos dados, tais como seu SNR e sua altitude relativa ao horizonte, por exemplo, obtendo-se um conjunto de dados da seguinte forma:

"HrMinSec(UTC);Day;Month;Year;Satellite PRN number;Elevation;Azimuth;SNR".

A partir desse conjunto de dados é obtido uma lista de dados na forma "*Date Object; S_4 Index*" para um *Satellite PRN* especificado e com o índice S_4 calculado por intervalos minutais utilizando-se, em média, 600 amostras de SNR para dado satélite por meio da expressão:

$$S_4 = \frac{std(I)}{mean(I)}, \text{ onde } I = 10^{\frac{SNR}{10}}$$

Assim, finalmente, a partir do NMEA bruto executa-se a rotina que toma todos os diferentes satélites capturados e traça suas respectivas curvas para índices S4, mapeando a cintilação ionosférica a partir de todos os satélites geoestacionários e não geoestacionários visíveis naquele instante. A rotina em Python está descrita abaixo.

```
import numpy as np, os, matplotlib.pyplot as plt
from datetime import datetime

def nmeaParser(array):
	newArray = []
	currentPackage = []
	for row in array:
		try:
			if row[1] == 'G' and row[2] == 'P' and row[3] == 'G' and row[4] == 'S' and row[5] ==
'V':
				gpgsv = row.split(',')
				gpgsv = gpgsv[4 : len(gpgsv)]
				gpgsv[len(gpgsv)-1] = gpgsv[len(gpgsv) - 1].split('*')[0]
				i = 0
				while i <= len(gpgsv) - 4:
					currentPackage.append(';'.join(gpgsv[i : i + 4]) + ';')
					i += 4
			elif row[1] == 'G' and row[2] == 'P' and row[3] == 'Z' and row[4] == 'D' and row[5] ==
'A':
				gpzda = ';'.join(row.split(',')[1 : 5]) + ';'
				for newRow in currentPackage:
					newArray.append(gpzda +  newRow)
				currentPackage = []
		except Exception as e:
			pass
	return newArray
```

```
def getTime(dateStr):
        dateStr = dateStr[:2] + ';' + dateStr[2:4] + ';' + dateStr[4:6] + ';' + dateStr[11:]
        date = datetime.strptime(dateStr, '%H;%M;%S;%d;%m;%Y')
        return date

def getSummary(array):
        minute = array[1].split(';')[0][2 : 4]
        returnArray = []
        summary = []
        summarize = []
        for i in range(1, len(array)):
                snr = array[i].split(';')[7]
                elevation = array[i].split(';')[7]
                if i == len(array) - 1 or minute != array[i + 1].split(';')[0][2 : 4]:
                        if i != len(array) - 1:
                                minute = array[i + 1].split(';')[0][2 : 4]
                                time = ';'.join(array[i].split(';')[0 : 4])
                        if len(summarize) > 0:
                                summary.append({
                                        'time': getTime(time),
                                        's4': 100.0 * S4Index(summarize),
                                        'elevation': float(elevation)
                                        })
                                summarize=[]
                if snr != "":
                        summarize.append(snr)
        return summary

def S4Index(array):
        intensity = []
        for snr in array:
                intensity.append((10.0 ** (float(snr) / 10.0)))
        s4 = round(1.0 * np.std(intensity) / np.mean(intensity), 3)
        return s4

def plot(x, y, yy, prn, directory):
        plt.plot(x, y, 'r--', x, yy, 'b--')
        plt.ylim([0,90])
        plt.title('PRN ' + prn)
        plt.legend(['100 * S4', 'Elevation'])
        plt.savefig(directory + '/' + prn + '.png', bbox_inches='tight')
        plt.close()
```

```
def getNmea():
        array = []
        with open('nmea_example.txt', 'rb') as f:
            for line in f:
                array.append(line[:len(line) - 1])
        return array

array = getNmea()
array = nmeaParser(array)
prns = set([ elem.split(';')[4] for elem in array])
now = datetime.now()
directory = 'images_nmea_parser_'+str(now.year)+"_"+str(now.month)+"_"+str(now.day)+"_"+str(now.second)
if not os.path.exists(directory):
    os.makedirs(directory)
print(prns)
for prn in prns:
        try:
            filtered_array = [elem for elem in array if elem.split(';')[4] == prn]
            array_to_plot = getSummary(filtered_array)
            x = [el['time'] for el in array_to_plot]
            y = [el['s4'] for el in array_to_plot]
            yy = [el['elevation'] for el in array_to_plot]
            plot(x, y, yy, prn, directory)
        except Exception as e:
            pass
```

Por fim, os códigos podem ser consultados integralmente no GitHub, no seguinte link:

https://github.com/igorbragaia/scintillation

3. Resultados

A partir das rotinas anteriormente descritas, foram geradas curvas relativas aos valores do índice S4 obtido a partir de diferentes satélites e os resultados estão exibidos abaixo.

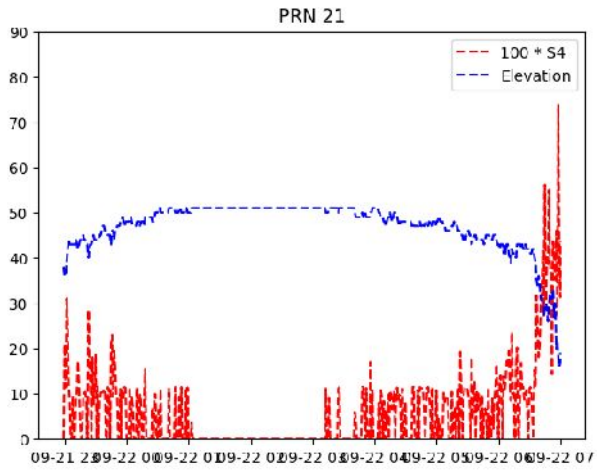

Figura 3 Comportamento do índice S4 de acordo com a elevação do satélite PRN21

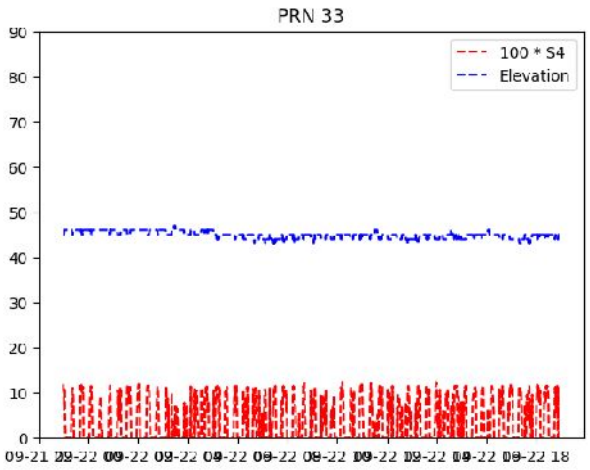

Figura 4 Comportamento do índice S4 de acordo com a elevação do satélite PRN33

Nota-se que na Figura 3, por exemplo, o índice oscila de modo a sugerir ocorrência de efeitos ionosféricos (deve-se levar em conta também distúrbios causados pela localização do sensor GPS ou por obstáculos naturais tais como elevações terrestres ou condições atmosféricas), enquanto que na Figura 4 o índice apresenta valores baixos considerados saudáveis, ou seja, que não sugerem ocorrência de cintilação.

Os resultados numéricos que geraram as figuras acima foram confrontados com dados para os mesmos períodos e mesmos satélites obtidos a partir dos sensores do projeto CIGALA (Concept for Ionospheric Scintillation Mitigation for Professional GNSS in Latin America) o qual consiste em uma base de dados integrada financiada pela European Commission (EC) e mantido, entre outros, pela FCT/UNESP. Os resultados confrontados foram obtidos via Javascript e PHP a partir dos mesmos dados processados do NMEA bruto usados nas Figuras 3 e 4 e os códigos podem ser consultados em:

https://github.com/igorbragaia/ionik

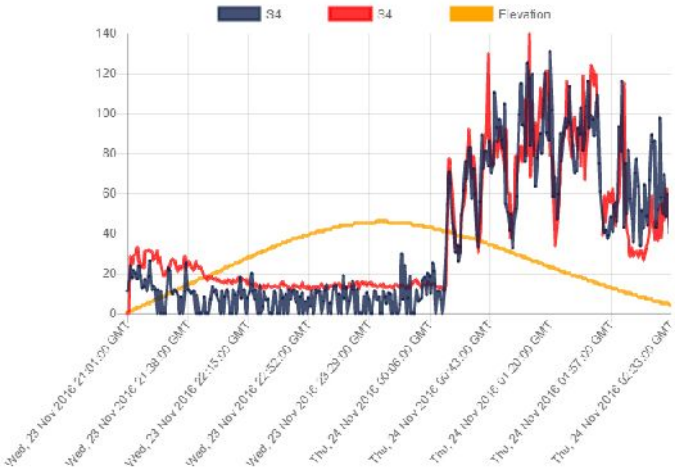

Figura 5 Comparativo entre resultados obtidos nesse experimento e no CIGALA. As linhas azul escuro e amarela são relativas ao índice S4 do protótipo desenvolvido e a elevação do satélite de PRN26, respectivamente, e a linha vermelha é relativa ao mesmo satélite, porém obtidos pelo CIGALA para o mesmo período.

Notou-se, por fim, que os resultados se mostraram satisfatórios dado o custo e a menor complexidade do aparamento experimental em questão se comparado com os monitores ionosféricos profissionais do CIGALA.

Inicialmente o projeto do monitor de cintilação foi projetado para operar utilizando a plataforma Arduino. Esta plataforma atendeu o objetivo inicial do projeto que era a prova de conceito sobre o uso de sensores de baixo custo. Entretanto os testes de validação mostraram que o Arduino não atenderia um projeto mais complexo envolvendo a transmissão de sinais pela rede e sua operação em longo prazo mostrou-se inviável. Em razão deste fato, este projeto foi revisto e optou-se pela adoção do Raspberry Pi como plataforma do monitor de cintilação. As atividades apresentadas nesta seção e desenvolvidas nos últimos 6 meses consiste na implementação do código e testes do mesmo para operar no Raspberry Pi.

4. Próximas Etapas

As próximas etapas consistem em validar o software em Python desenvolvido em um dispositivo do tipo Raspberry Pi e posteriormente aplicá-lo em escala para testes. Em seguida, dar-se-á início a montagem de uma rede de monitoramento ionosférico para atender um conjunto de monitores deste tipo.

5. Conclusões

Este relatório apresentou as atividades desenvolvidas neste projeto até o presente momento. Após a validação do protótipo, pôde-se concluir que é possível a obtenção de dados

satisfatórios relativos à estudos ionosféricos por meio de equipamentos de baixos custos baseados em tecnologias menos complexas do que equipamentos profissionais disponíveis no mercado. Também se nota a flexibilidade e aplicabilidade da linguagem Python no contexto do processamento de dados, de forma que se executa a captura, tratamento e visualização dos resultados em uma única rotina integrada.

Além disso, o baixo custo do protótipo e sua flexibilidade de uso permite com que futuramente sejam distribuídos monitores em locais diferentes e assim obtido um mapeamento da ionosfera em um território amplo permitindo estudos mais complexos e de maior impacto sobre o território brasileiro.

6. Divulgação dos Resultados

O projeto de desenvolvimento de um monitor ionosférico de baixo custo, o qual esse presente projeto dá continuação, teve um Abstract aceito na 32nd URSI Conference em Montreal, Canadá. Assim, foi apresentado um pôster na conferência pelo pesquisador Dr. Eurico Rodrigues de Paula, do INPE, e o pôster em inglês pode ser visualizado abaixo.

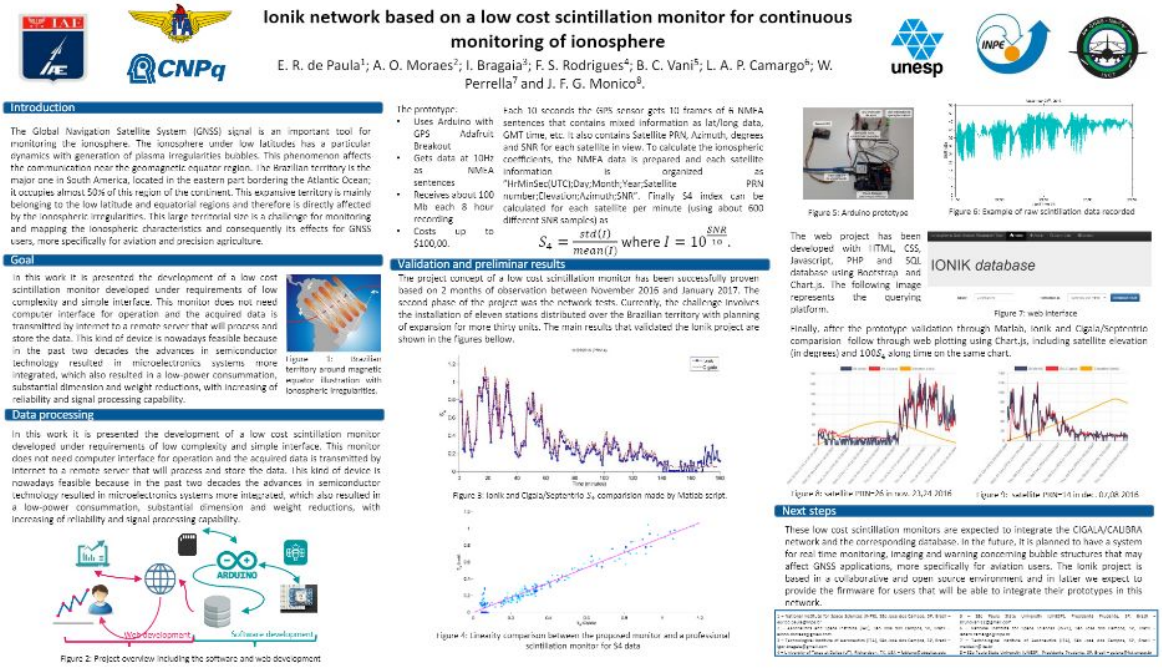

Figura 6 Pôster apresentado na 32nd URSI Conference. Pode-se consultar o poster em alta resolução em https://github.com/igorbragaia/ionik/blob/master/poster.pdf

Referências

[1] http://www.gpsinformation.org/dale/nmea.htm

[2] https://www.python.org/doc/