MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Desenvolvimento de um Emulador de Painéis Solares para Nanosatélites

RELATÓRIO FINAL DE PROJETO DE INICIAÇÃO CIENTÍFICA
(PIBIC/CNPq/INPE)

Igor Frassoni Guedes dos Santos (UNIFESP, Bolsista PIBIC/CNPq)
E-mail: igor.santos@inpe.br

Leandro Toss Hoffmann (DSS/ETE/INPE, Orientador)
E-mail: leandro.hoffmann@inpe.br

COLABORADORES

Cleber Toss Hoffmann (FUNCATE, LIT/INPE)

Julho de 2015

Sumário

1. RESUMO 3
2. INTRODUÇÃO 4
2.1. Células Solares 4
3.0. OBJETIVOS 6
3.1. Cronograma 6
3.2. Objetivos Concluídos 7
4.0. DESENVOLVIMENTO 8
4.1. Circuito Exemplificativo de uma Célula Solar 8
4.1.1. Modelo Utilizado no Projeto 8
4.2. Gráfico Corrente-Tensão 10
4.3. Implementação e o uso do Arduino 13
4.4. Melhora dos scripts em Matlab 13
4.5. Desenvolvimento dos Scripts em Linguagem C 14
4.6. Desenvolvimento da Primeira Versão do Emulador 14
4.6.1. A Placa 15
4.6.2. O Software 16
4.6.3. Testes Preliminares 16
4.6.4. Software Embarcado 17
4.6.5. Dificuldades 19
5. RESULTADOS 20
6. CONCLUSÃO 22
7. REFERÊNCIAS BIBLIOGRÁFICAS 23
ANEXO 1: Script iparams 24
ANEXO 2: Script vcell 25
ANEXO 3: Script icell 26
ANEXO 4: Script plotiv 27
ANEXO 5: Tabela e gráfico de valores de Tensão 29
ANEXO 6: Tabela e gráfico de valores de Corrente 31
ANEXO 7: Função final da versão 1 do emulador 32

1. RESUMO

Este projeto tem como principal objetivo construir um emulador de painéis solares, visando o suporte de subsistemas de suprimento de energia. De forma específica, o bolsista deverá modelar as características elétricas de células fotovoltaicas e seu comportamento com base nas diferentes cargas resistivas e variações do ambiente (e.g. temperatura e iluminação). Com base neste modelo, um software será codificado e embarcado em um microcontrolador ARM acoplado a uma placa eletrônica que reproduzirá as características físicas de um painel fotovoltaico real. Por fim, os parâmetros do painel solar serão lidos de um simulador computacional de satélite existente.

Inicialmente, foi realizado um vasto estudo a respeito de painéis fotovoltaicos, bem como o funcionamento das células solares, como suas características físicas e energéticas. Em seguida, modelos das curvas de Corrente x Tensão das células foram implementados em script Matlab.

No decorrer do projeto, o bolsista obteve contato com placas Arduino, visando o contato inicial com softwares embarcados. A partir do contato inicial, o bolsista foi capaz de iniciar o desenvolvimento no microcontrolador ARM, tendo finalizado a primeira versão do emulador.

Com a finalização da primeira versão, espera-se para trabalhos futuros a implementação da segunda e terceira versões do emulador, onde esta é aprimorada com a variação de parâmetros ambientais e por fim integrado ao simulador externo.

2. INTRODUÇÃO

A utilização de painéis fotovoltaicos em satélites é grande, pois esta é uma fonte muito viável de produção de energia em um ambiente tão extremo, como o espaço. Além do mais, sua complexidade operacional e custo podem ser considerados baixos, quando comparados a outras fontes d eenergia (e.g. energia nuclear).

Com o crescente aumento do uso desse sistema de suprimento de energia no meio espacial, técnicas de validação e simulação vêm sendo criadas para projetar melhores sistemas e minimizar os custos de missões espaciais.

Um dos principais fatores que influenciam o sistema de suprimento de energia é a necessidade de se determinar a quantidade de energia que o satélite vai demandar, o qual, por sua vez, determinará a dimensão dos painéis solares e o número de células solares utilizados.

2.1. Células Solares

Células solares se caracterizam basicamente por ser um conjunto de materiais (semicondutores) capaz de gerar energia elétrica a partir da "conversão" de fótons em corrente elétrica. Na *figura 1* pode-se ver um exemplo de célula solar.

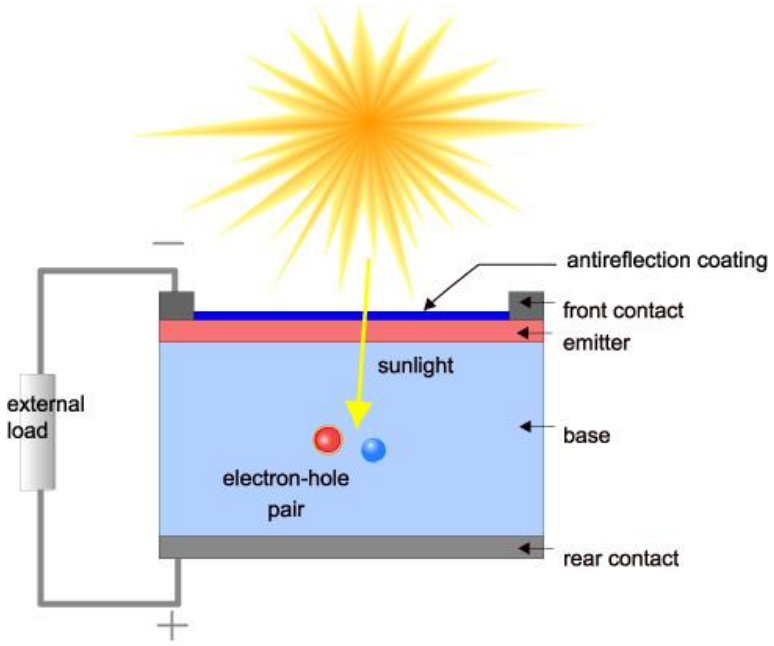


Figura 1: exemplo de estrutura de uma célula solar
(Imagem proveniente de: http://www.pveducation.org/pvcdrom/solar-cell-operation/solar-cell-structure)

Na figura 1, o funcionamento ocorre da seguinte maneira:

1. Os raios solares incidem sobre o painel solar (que possui uma placa de anti-reflexo).
2. Os fótons presentes nos raios solares são absorvidos pelo material semicondutor (em vermelho), este material por sua vez, tem características químicas com tendências a ceder elétrons. Essa absorção faz com que este material se torne carregado eletronegativamente.
3. Este processo de incidência de fótons "cria" um elétron e um "buraco", caracterizando-se por ser a falta deste elétron na estrutura atômica do material.
4. Em azul, encontra-se outro material semicondutor, porém com características químicas com tendências a receber elétrons.
5. Como o material está carregado eletronegativamente (em vermelho), e em azul, pode-se dizer que está com falta de elétrons com relação ao outro material, cria-se então uma diferença de potencial entre esses materiais.
6. Com esta diferença de potencial, um fluxo de elétrons (corrente elétrica) começa a aparecer a partir da condução deste por meio de uma ligação (carga externa, external load). No fim do ciclo, o elétron se encontra com o "buraco" gerado.

3.0. OBJETIVOS

O principal objetivo deste projeto consiste em desenvolver um emulador de painéis fotovoltaicos para apoiar o desenvolvimento de subsistemas de suprimento de energia de nanosatélites. Para tanto, o bolsista modelará as características elétricas de células fotovoltaicas e seu comportamento frente a diferentes cargas resistivas e variações do ambiente (e.g. temperatura e iluminação). Com base neste modelo, um software será codificado e embarcado em um microcontrolador acoplado a uma placa eletrônica que reproduzirá as características físicas de um painel fotovoltaico real. Por fim, os parâmetros do painel solar serão lidos de um simulador computacional de satélite existente.

3.1. Cronograma

Etapa	Mês (m1 = agosto)											
	1	2	3	4	5	6	7	8	9	10	11	12
A	■	■	■									
B			■	■								
C					■							
D						■	■					
E								■	■			
F										■	■	
G												■

O cronograma original do projeto previa as seguintes atividades a serem desenvolvidas pelo bolsista:

A. **Levantamento bibliográfico e embasamento teórico**: atividades de leitura sobre o funcionamento de células fotovoltaicas, suas principais características e arranjos; identificação de modelos matemáticos existentes na literatura.

B. **Reprodução de modelos de painéis fotovoltaicos**: geração de gráficos das curvas IV de células e painéis solares a partir da codificação em linguagem de alto nível dos modelos matemáticos levantados; análise do comportamento dos modelos mediante variação dos seus parâmetros.

C. **Familiarização com o ambiente de desenvolvimento**: inspeção da placa eletrônica a ser utilizada na construção do emulador e criação de cenários de uso simplificados a fim de se identificar a arquitetura do circuito e se familiarizar com o ambiente de desenvolvimento do microcontrolador.

D. **Desenvolvimento da versão 1 do emulador**: codificação de um componente de software no microcontrolador para operar o módulo PWM e emular uma tensão elétrica a partir de um modelo simplificado de painel fotovoltaico, na forma de uma curva IV fixa.

E. **Desenvolvimento da versão 2 do emulador**: o software da versão 1 é aprimorado com a substituição da curva IV fixa por um modelo parametrizável do painel fotovoltaico e ajustável em função da variação de parâmetros ambientais.

F. **Desenvolvimento da versão 3 do emulador**: o software da versão 2 é aprimorado para efetuar a leitura dinâmica dos parâmetros ambientais a partir de uma comunicação serial com um simulador externo.

G. Produção de relatório final e publicações

3.2. Objetivos Concluídos

De acordo com o cronograma apresentado, desde o início do projeto, as etapas de A a D foram concluídas, de forma que na primeira e segunda etapas, o bolsista efetuou diversos estudos a respeito do funcionamento de células solares e de simulações efetuadas (VILLALVA, 2009). Na segunda etapa foi feito também a implementação de scripts em Matlab baseados no autor citado anteriormente, de forma a desenvolver modelos e gerar dados necessários, como gráficos de corrente e tensão, potência, etc. Na terceira etapa, foi iniciado o estudo a respeito da linguagem de programação C, desenvolvendo pequenas rotinas para a familiarização com o ambiente, este por sua vez será utilizado posteriormente. Além do mais, foi feito um breve estudo a respeito do microcontrolador ARM Cortex M3, utilizando-se de seu guia para tanto. Na quarta etapa, o bolsista começou a desenvolver diretamente no software para a programação do microcontrolador ARM, o IAR Embedded Workbench. Nesta etapa, o bolsista finalizou a primeira versão do emulador proposto. Atualmente, encontra-se trabalhando na etapa E, com o aprimoramento da versão 1 do emulador.

4.0. DESENVOLVIMENTO

Com base no cronograma, os estudos foram iniciados a partir de leituras a respeito de células solares aplicadas no uso espacial, de forma a introduzir o bolsista ao projeto.

Como visto, a partir do estudo do funcionamento da célula solar, foi possível verificar os fatores que determinam as principais características das células e arranjos, apresentados sucintamente a seguir.

4.1. Circuito Exemplificativo de uma Célula Solar

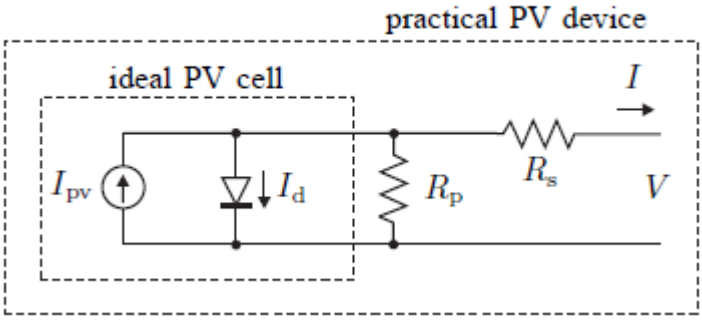


Figura 2: circuito exemplificativo de uma célula solar
Fonte: VILLALVA, 2009, p. 1200

Onde:

Ipv Representa a corrente fornecida por meio da incidência solar

I_D Representa a corrente do diodo (representação dos materiais semicondutores)

I Representa a corrente final gerada para a carga

Sendo assim, a corrente final (útil) pode ser definida como:

$$I = \text{Ipv} - I_D - \text{Ip}$$

4.1.1. Modelo Utilizado no Projeto

No projeto, o modelo utilizado foi o mesmo utilizado por (PERONDI, 1987) , sendo este baseado em (RAUSCHENBACK, 1980). As equações utilizadas no modelo podem ser vistas a seguir.

- Corrente de Curto-circuito

$$\text{Isc} = (\text{Iscn} + \text{aISC} * (\text{T} - \text{Tn})) * \text{Risc} * \text{Area}$$

Onde,

Isc: Corrente de curto-circuito (mA)

Area: é a área da célula solar (m²)

- Corrente de Máxima Potência

$$Imp = (Impn + aIMP * (T) * Rimp * Area$$

Onde,

Imp: corrente de máxima potência (mA)

- Tensão em Circuito Aberto

$$Voc = (Vocn + aVOC * (T - Tn)) * Rvoc$$

Onde,

Voc: Tensão de circuito aberto (V)

- Tensão de Máxima Potência

$$Vmp = (Vmpn + aVMP * dT) * Rvmp$$

Onde,

Vmp: Tensão de máxima potência (V)

Risc: Resistência de curto-circuito (Ohm)

Rimp: Resistência de máxima portência (Ohm)

Rvoc: Resistência de circuito aberto (Ohm)

- Cálculo da corrente

$$c2 = (Vmp / Voc - 1) / \log(1 - Imp / Isc),$$

$$c1 = (1 - Imp / Isc) * \exp(-Vmp / (c2 * Voc)),$$

$$i = Isc * (G / Gn - c1 * (\exp(V / (c2 * Voc)) - 1))$$

As referências às variáveis utilizadas podem ser encontradas na *Tabela 1*.

4.2. Gráfico Corrente-Tensão

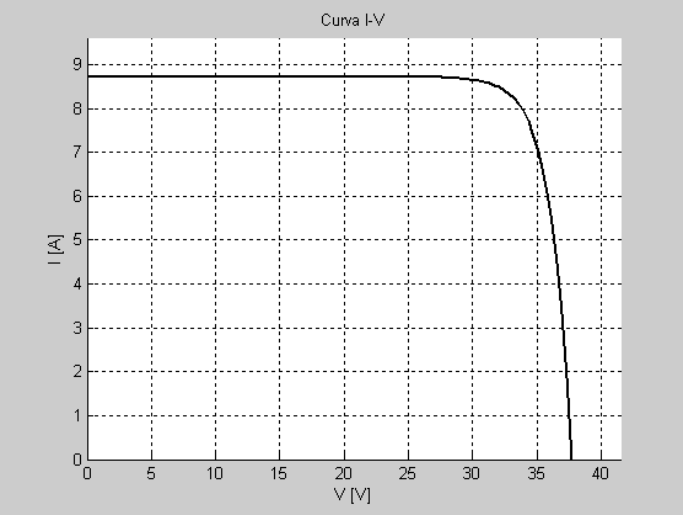


Figura 3: Curva I-V

Diversos fatores são considerados na elaboração de uma célula solar, dependendo da sua necessidade de uso. Por exemplo, devem-se considerar as necessidades do dispositivo que será alimentado por meio da célula solar, as características elétricas necessárias, como a tensão necessária, bem como a corrente. Tais fatores podem ser balanceados a partir do gráfico Corrente-Tensão deste dispositivo.

Na Figura 3, pode-se observar o gráfico característico da curva de Corrente-Tensão de uma célula Solar.

Diversos fatores influenciam na alteração dessa curva, de forma que esta sofre alterações tanto nas mudanças de tensão quanto de corrente, dependendo do fator.

Na figura 4, tem-se que a corrente varia drasticamente com a variação do número de células colocadas em paralelo. Na figura, conforme o número de células em paralelo aumenta, o valor da corrente aumenta também.

Na figura 5, pode-se observar a variação da curva IV, conforme se altera a temperatura e/ou a quantidade de luz recebida pela célula solar. Conforme os valores de irradiância aumentam, a área da curva tende a ficar maior, e o contrário ocorre com a variação de temperatura.

Na figura 6, conforme a temperatura aumenta, a potência diminui, sendo assim uma relação inversamente proporcional entre a temperatura e a potência.

As células solares podem ser arranjadas de diversas formas, de forma a possibilitar variadas características elétricas, podendo este conjunto de células serem dispostas de tal forma que criem uma grande diferença de potencial (tensão) ou então de forma a gerar um grande fluxo de corrente. Tais características podem ser criadas a partir do tipo do arranjo criado.

É importante ressaltar que os gráficos apresentados foram gerados por meio de scripts implementados no Matlab pelo bolsista. A seguir, pode-se verificar como esses fatores alteram a

curva IV.

Neste trabalho, as características das células utilizadas no modelo do emulador estão apresentadas na tabela 1.

Tabela 1: Valores nominais de operação da célula

	Spectrolab GaAs double junction	Optimum Mod SG2 GaAS - single junction
ISC0	30.50	15.05
IMP0	28.60	14.15
VOC0	1.025	2.360
VMP0	0.900	2.085
aISC	0.0200	0.012
aIMP	0.0200	0.013
aVOC	-0.00180	-0.0048
aVMP	-0.00190	-0.005
Risc	0.95	0.95
Rimp	0.95	1.00
Rvoc	0.95	0.95
Rvmp	0.94	0.96
Área	16	16
T0	28	
S0	1353	

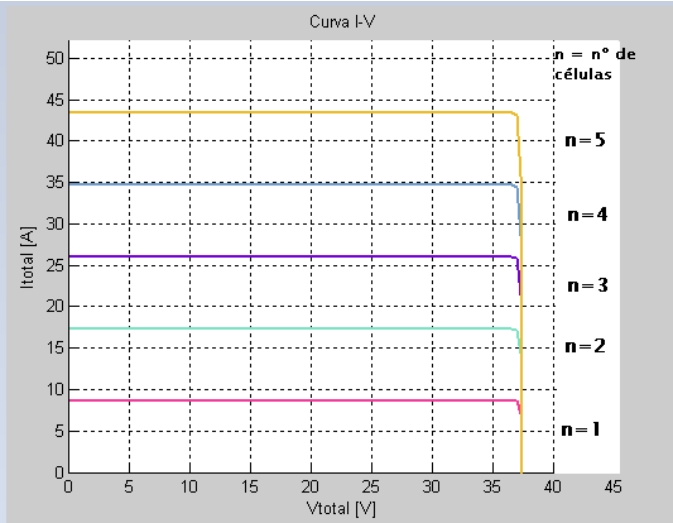


Figura 4: Demonstração da alteração da curva IV conforme número de células em paralelo

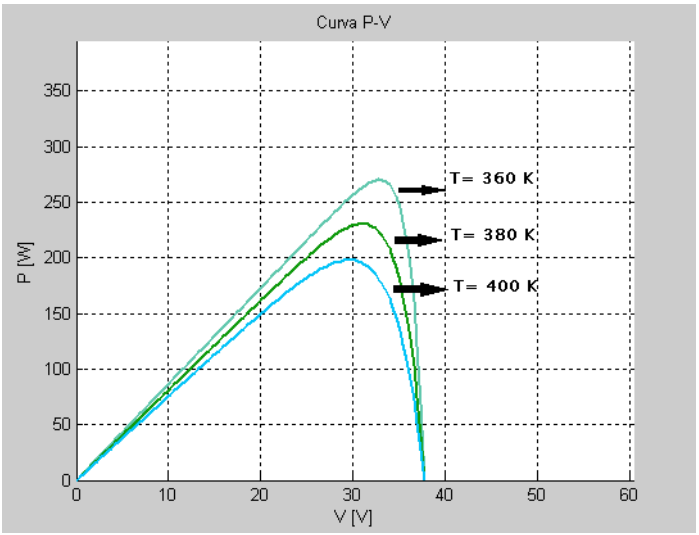


Figura 5: Exemplo de curva IV de acordo com a variação de temperatura e irradiância

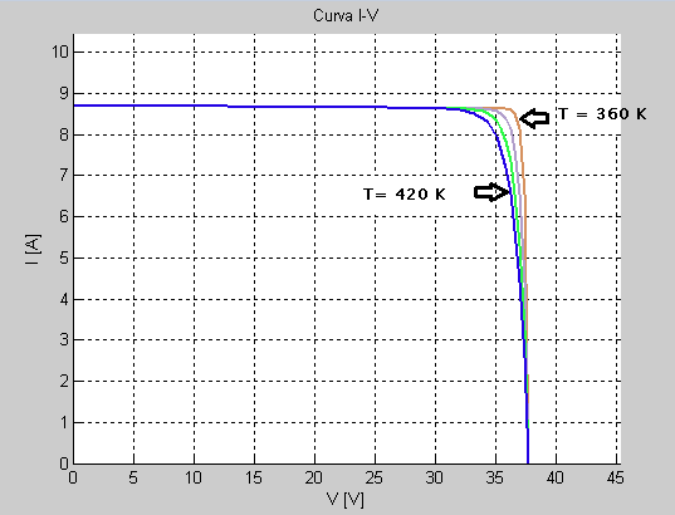


Figura 6: Exemplo de curva PV de acordo com a variação de temperatura e irradiância

No intuito de se criar um conjunto que gere uma tensão alta, as células dever ser dispostas em paralelo, visto, que assim como as características elétricas de um circuito básico, a tensão em seus componentes aumenta quando estes se encontram em série. Se o objetivo for criar um arranjo que possibilite uma corrente mais elevada, tal arranjo deve ser feito com células em paralelo, com o mesmo pensamento da tensão, a corrente em componentes que estão em paralelo, se soma ao fim do nó. Nas figuras a seguir, pode-se verificar como o processo ocorre. Dependendo da necessidade, há também a possibilidade de se criar arranjos mistos, de forma a criar uma boa relação entre Tensão x Corrente.

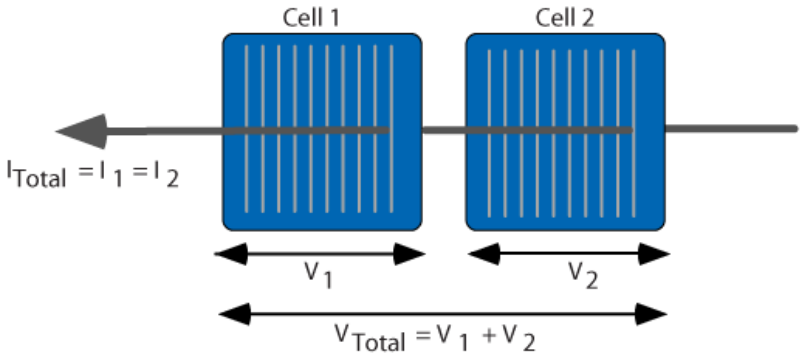


Figura 7: Arranjo de células em série, tensão total = V1+V2, corrente total é a mesma para as células, I1=I2
*(*Imagem proveniente de: http://www.pveducation.org/pvcdrom/modules/mismatch-for-cells-connected-in-series*)*

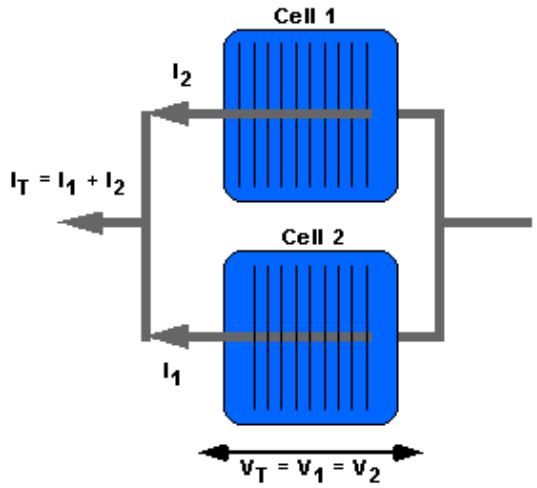


Figura 8: Arranjo de células em paralelo, tensão total = V1=V2, corrente total = I1+I2
(Imagem proveniente de: http://www.pveducation.org/pvcdrom/modules/mismatch-for-cells-connected-in-parallel)

4.3. Implementação e o uso do Arduino

A implementação dos scripts foi realizada por meio do Matlab, e esta foi baseada em artigos previamente estudados, entre eles pode-se destacar os trabalhos de Villalva, o qual apresenta uma métrica bastante detalhada para gerar a simulação, considerando diversos fatores que causam a alteração da corrente/tensão na célula solar.

A implementação realizada não levou em conta todos os fatores apresentados no trabalho estudado, visto que seriam fatores que na prática, não seriam possíveis de ser reproduzidos com tamanha definição.

A plataforma de prototipagem eletrônica de hardware livre e de placa única, também conhecida como Arduino, foi utilizada no projeto apresentado, pois esta mostrou-se útil com relação a familiarização com o ambiente de desenvolvimento integrado, com o uso da linguagem C e de seu uso em plataformas embarcadas, que futuramente, será utilizado no microcontrolador ARM.

4.4. Melhora dos scripts em Matlab

No período correspondente de março a abril o bolsista trabalhou na correção do relatório parcial, fazendo ajustes e acrescentando novos itens no relatório, bem como a correção e melhoria de scripts em Matlab com a ajuda de seu orientador. Mais especificamente foram criados scripts para a geração do gráfico Corrente x Tensão, com uma melhor organização frente aos anteriores. Um desses scripts pode ser encontrado no *anexo 1*. Ainda, os scripts estão dispostos de uma forma vista na *figura 9*.

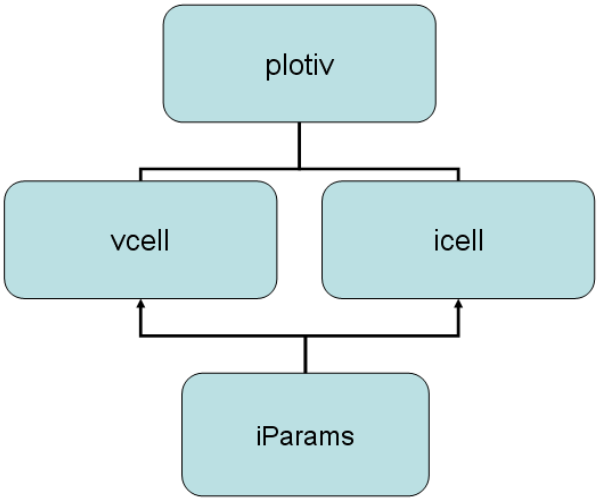


Figura 9: Diagrama exemplificativo dos scripts utilizados para geras os gráficos de corrente e tensão

O script *iParams* contém os parâmetros iniciais necessários para os cálculos realizados nos scripts *vcell* e *icell*, esses por sua vez, geram os dados necessários para criar os gráficos de corrente e tensão no script *plotiv.*

4.5. Desenvolvimento dos Scripts em Linguagem C

A partir dos scripts desenvolvidos em Matlab, o bolsista começou o desenvolvimento dos mesmos em C, de forma a posteriormente, ser possível efetuar a implementação no microcontrolador ARM.
Nesta etapa, o bolsista basicamente copiou os códigos desenvolvidos em Matlab e transferi-os para a linguagem C, com exceção de algumas funções específicas do Matlab, como a 'fzero', uma função que retorna as raízes de equações não-lineares. Outras rotinas para representar esse processo tiveram que ser desenvolvidas.

4.6. Desenvolvimento da Primeira Versão do Emulador

Com os scripts em Linguagem C finalizados, deu-se inicio a implementação diretamente na placa eletrônica que possui o microcontrolador ARM.
Primeiramente, para esta etapa, alguns procedimentos foram efetuados, tais como:

- Familiarização do ambiente de desenvolvimento – Neste ponto, o bolsista dirigiu-se para o laboratório, onde encontravam-se os componentes para o desenvolvimento. Por meio de um computador previamente configurado com o software IAR Embedded Workbench, o bolsista foi apresentado a plataforma.
- Testes preliminares com a placa – Após o contato inicial, o bolsista realizou alguns testes que futuramente viriam a serem úteis para a modelagem computacional. Os testes são apresentados na seção *4.6.2*.
- Implementação do software embarcado – Com os testes realizados, deu-se inicio ao

processo de implementação em si.

4.6.1. A Placa

A construção do emulador é baseada na placa eletrônica da Figura a seguir, a qual contém um microcontrolador ARM® Cortex-M3 modelo LM3S5G51.

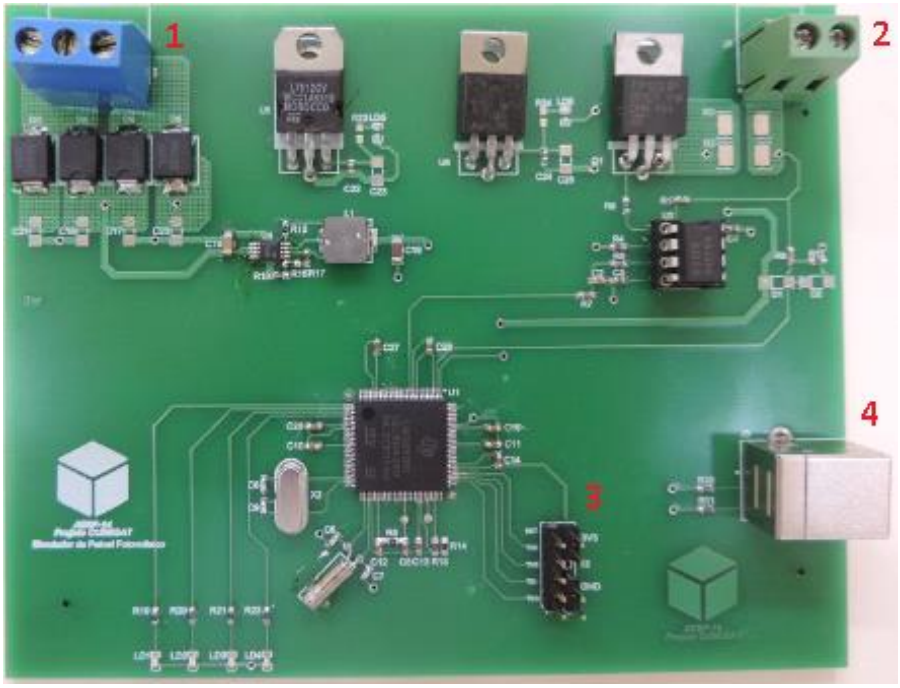


Figura 10: Placa eletrônica utilizada no projeto

Legenda:

1) Ponto onde a placa é alimentada
2) Ponto de Leitura (onde a corrente desejada é lida)
3) Ponto de Gravação, onde o software implementado é transferido para a placa (por meio de conexão serial)
4) Ponto de Comunicação, onde a placa e a plataforma se comunicam.

A placa tem como função gerar uma corrente no ponto 2, que pode ser programada a partir do software. Esta corrente pode ser programada de diversas formas, como por exemplo, por meio de software, definir um valor fixo, respeitando as limitações da placa.
No decorrer do desenvolvimento, notaram-se alguns detalhes a respeito da placa, como por exemplo, o fato de esta gerar um valor de tensão/corrente diferente (dentro de uma faixa correspondente ao valor real) quando, por meio de comando no software, se solicita o valor de corrente e/ou tensão no ponto 2.

4.6.2. O Software

O software de implementação utilizado no projeto foi o IAR Embedded Workbench, uma plataforma de desenvolvimento voltada para a área de software embarcado, com microcontroladores ARM. Nesse ambiente, é possível desenvolver rotinas computacionais de forma a ser possível embarcá-las no hardware, por meio de interface USB/serial. No projeto, o processo pode ser visto na *figura 11*.

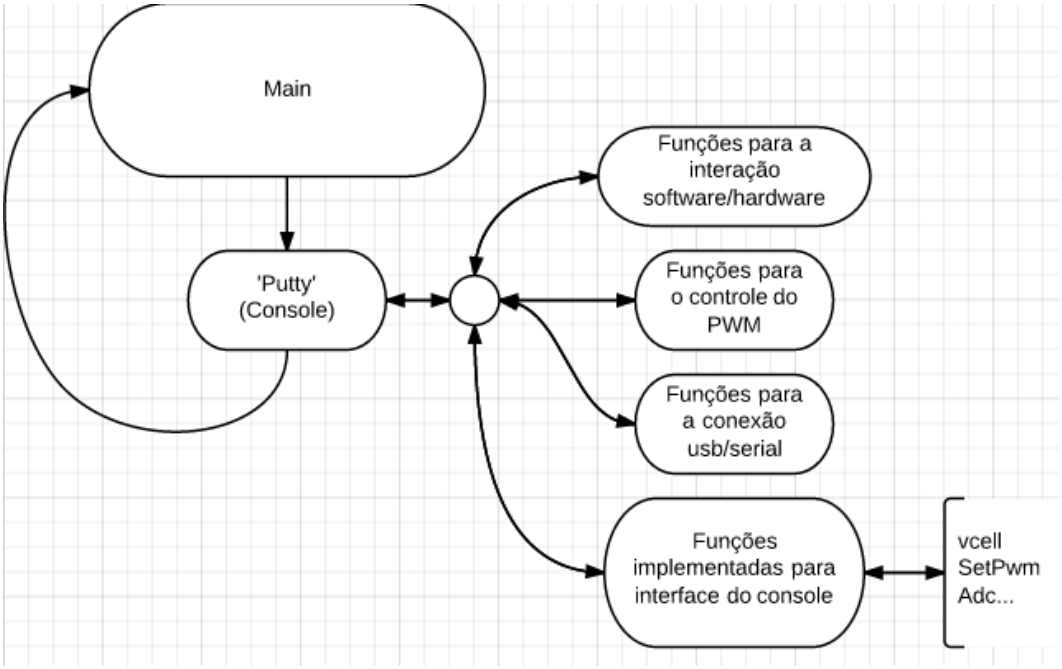


Figura 11: Esquemático da plataforma de desenvolvimento com relação ao projeto

Na *figura 11,* pode-se observar de uma forma simplificada o funcionamento do software. Um detalhe a ser destacado, é o console, referenciado como ‘putty’, este foi desenvolvido especialmente para a placa, sendo que possui algumas funções especiais para a conexão entre ambos[1]. É por meio deste também que ocorre a entrada e saída de dados.

4.6.3. Testes Preliminares

Como visto anteriormente, alguns testes foram realizados de forma a serem utilizados como parâmetros base para o desenvolvimento do emulador. Entre esses testes, estão:
A análise de variação de tensão da placa – a tensão na placa varia de forma analógica, sendo que no software esta variação é interpretada de forma digital. Sendo assim, o teste baseou-se em criar uma tabela de variação de tensão e caracterizar o valor correspondente encontrado por meio de medição direta (multímetro) com o valor interpretado pelo software.

[1] A placa e o console foram desenvolvidos pelo colaborador Cleber Toss Hoffmann

A análise de variação de corrente da placa – Da mesma forma que a tensão, a corrente é gerada por meio do PWM (*Pulse-Width Modulation*, do inglês, Modulação por Largura de Pulso), e no software é interpretada de forma digital. O teste foi feito de forma semelhante ao da tensão, com uma diferença na forma de medir, onde foi acrescentado um potenciômetro na saída (ponto 2). O esquema pode ser visto na figura 12.

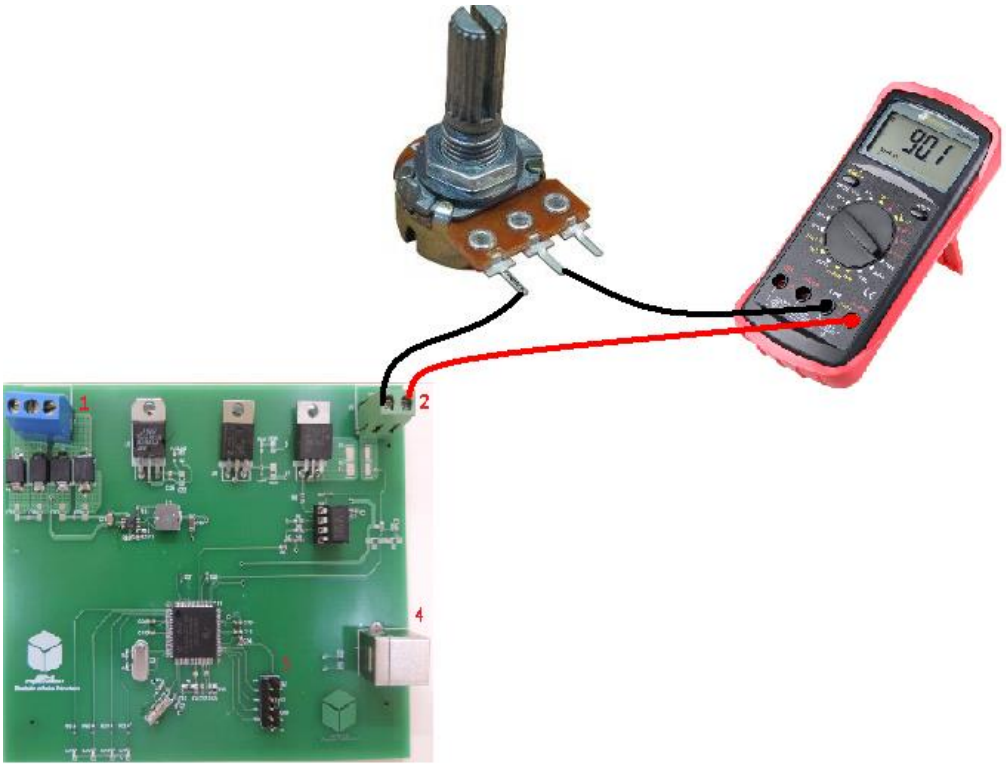


Figura 12: Esquema de leitura de corrente na placa

4.6.4. Software Embarcado

Com os dados obtidos por meio dos testes realizados, foi possível criar uma curva de variação para a corrente e tensão da placa, tornando-se mais fácil a modelagem computacional. Um fluxograma do processo pode ser visto na *figura 13*.

A seguir é apresentado o processo de desenvolvimento:

- A partir da plataforma IAR, por meio de um workspace (local de trabalho, em inglês) pré-configurado, o software é desenvolvido.
- O workspace possui diversas configurações, entre elas, comandos de leitura e impressão de dados.
- A partir de um script no workplace, funções são programadas conforme a necessidade, no caso da implementação proposta, foi criada uma função *vcell*.
- A função *vcell* funciona da seguinte maneira:

- Parâmetros espaciais são pré-definidos (i.e. temperatura, irradiação);
- A corrente no ponto de leitura é lida e convertida em mA;
- Com a corrente inicial, faz-se uso de técnicas para a busca do valor esperado de tensão. A técnica utilizada consiste em atribuir valores de limite superior e inferior e analisar o esse intervalo na busca do valor desejado;
- Feito o procedimento, tendo o valor de tensão, o algoritmo converte o valor encontrado em sinal digital, setando esse novo valor no local onde anteriormente havia-se feito a leitura de corrente;
- Com esse novo valor de tensão, a corrente no ponto muda, e caso o algoritmo seja executado novamente, o novo valor de tensão encontrado será baseado nesse novo valor de corrente, ou seja, de forma recursiva, até que este valor de corrente convirja para um ponto;

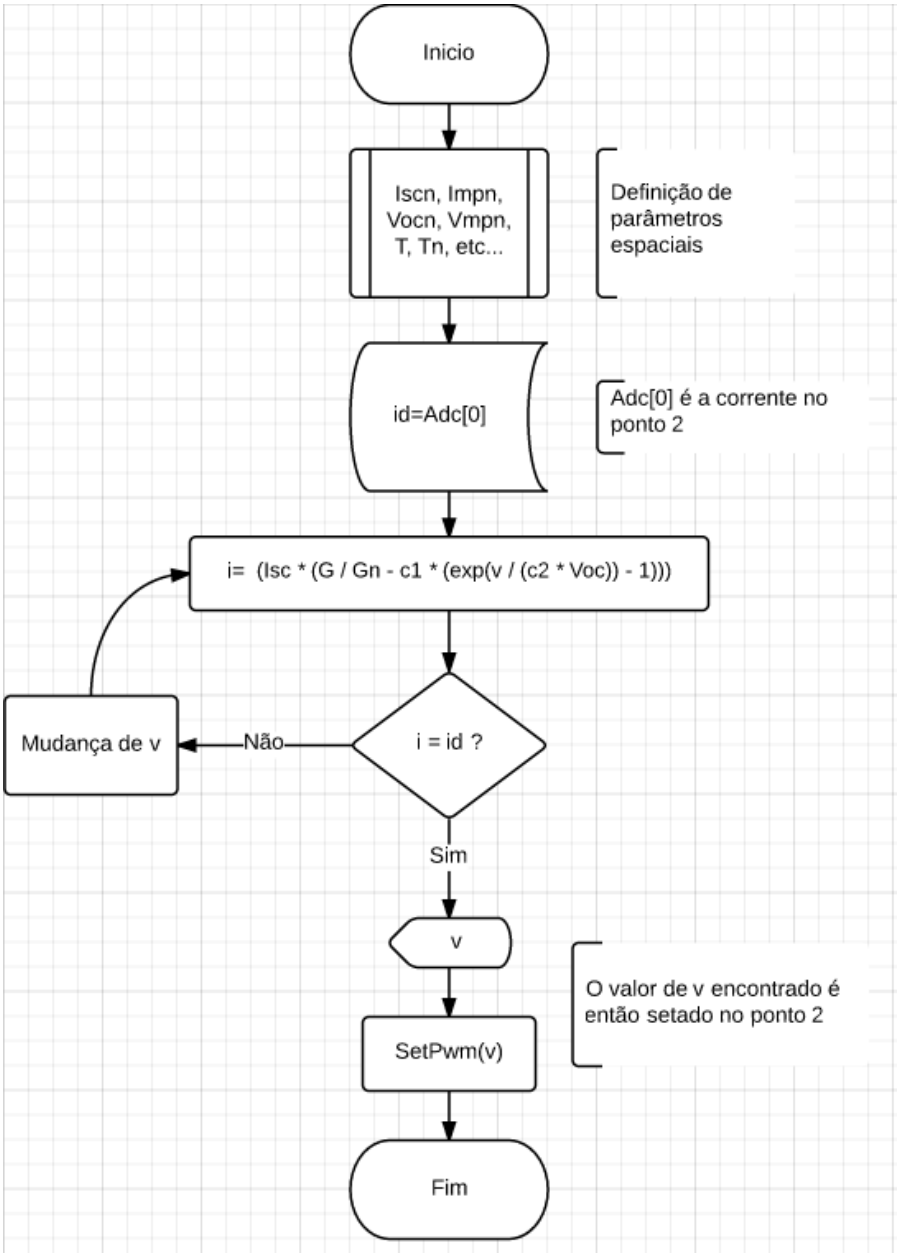


Figura 13: Fluxograma da função vcell

4.6.5. Dificuldades

Um dos maiores obstáculos encontrados pelo bolsista foi com relação a placa e o software de desenvolvimento, onde este por muitas vezes encontrou-se com dificuldades com relação a algumas funções e métodos de operação/execução da placa e do software.

Uma outra limitação, veio por parte da placa, onde esta não retorna resultados precisos, condizentes com o resultado obtido em software, ou seja, o software foi capaz de gerar valores de uma forma realística, porém o hardware não foi tão preciso ao ponto de refletir os resultados obtidos com o software. Este é um detalhe que futuramente pode ser corrigido com a melhora da placa.

5. RESULTADOS

Com o algoritmo pronto, alguns testes foram conduzidos, e são apresentados a seguir como forma de resultados para o projeto.

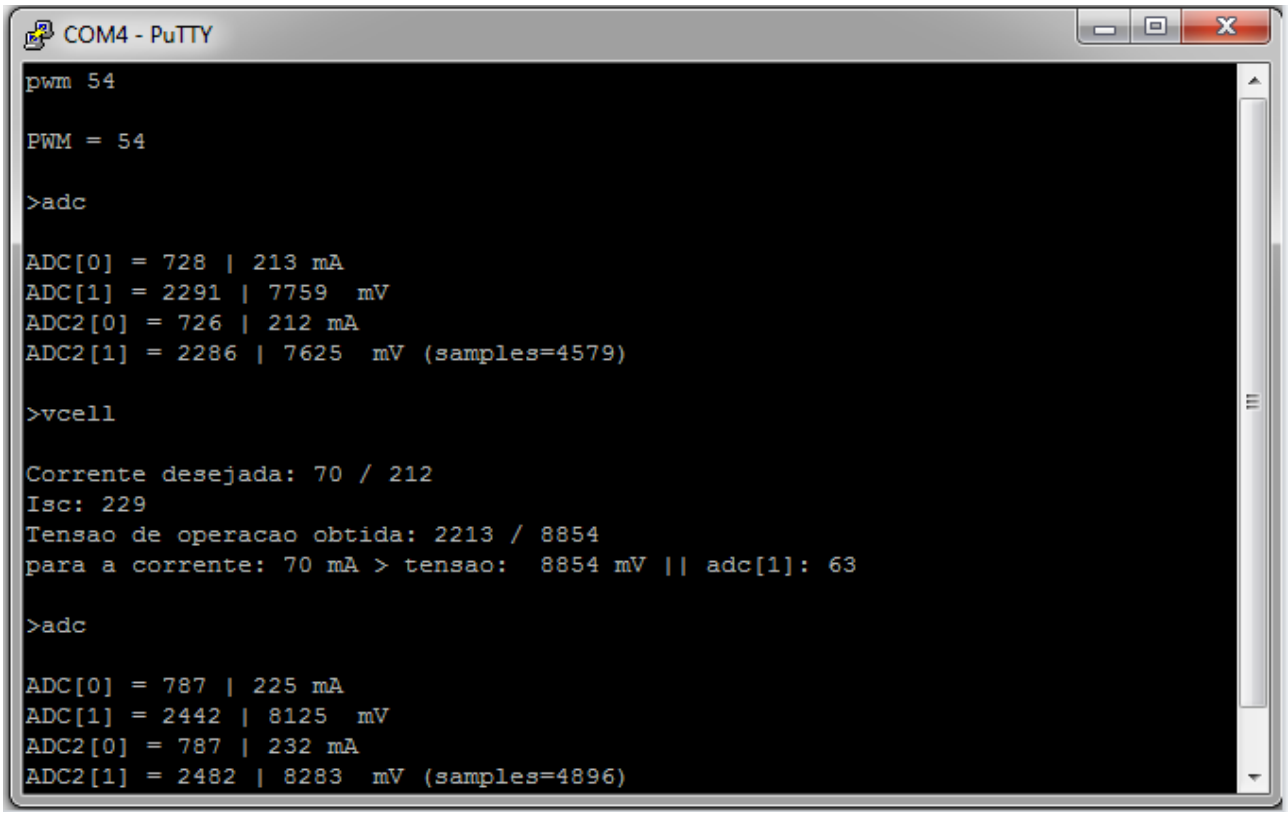


Figura 14: teste com o console 'putty'

Na *figura 14*, tem-se a execução do algoritmo, o qual funciona da seguinte maneira:

- Na primeira linha, tem-se o comando 'pwm 54', onde o pwm é setado em 54 (duty cycle de 54%);
- Na terceira linha, o comando 'adc' é inserido, gerando como resultado as quatro linhas subseqüentes, que representam os valores de corrente e tensão lidos no ponto 2. Adc[0] e Adc[1] são valores lidos diretamente, já Adc2[0] e Adc2[1] são valores filtrados, para obter um melhor resultado de corrente e tensão, pois como visto anteriormente, esses valores mudavam a medida que o comando era exacutado (sem o filtro);
- Logo após, o comando 'vcell' é executado, sendo este o algoritmo desenvolvido. Como visto anteriormente, ele utiliza-se do valor da corrente no ponto 2 (adc2[0]) para gerar uma nova tensão de operação, introduzindo esta nova tensão no mesmo ponto;
- Por fim, o comando 'adc' é executado novamente, para conferir se os valores de tensão e corrente se alteraram, conforme a função 'vcell' sugere. Como visto, os valores se alteram, comprovando a eficácia do algoritmo. É importante salientar que os valores mudam devido a uma execução interna da função SetPwm (equivalente ao comando 'pwm 54'), com base na

nova tensão obtida;

- O potenciômetro foi setado em 20 para este teste.

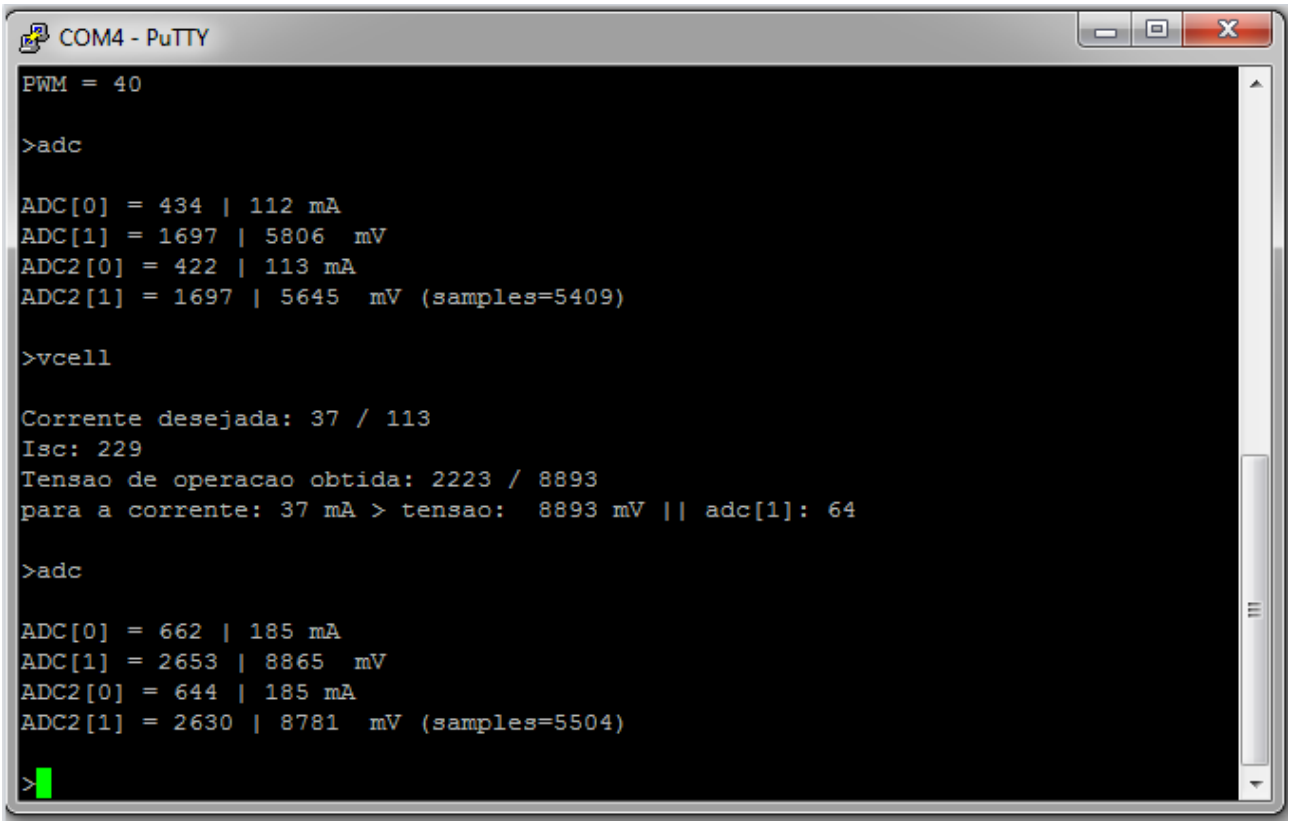


Figura 15: teste 2 com o console putty

A figura 15 representa um novo teste, para fins amostrais, a execução segue o mesmo processo descrito anteriormente, com a diferença agora do valor do pwm, setado em 40, e a resistência no potenciômetro, setado em 50.

6. CONCLUSÃO

Com base no que foi apresentado, pode-se perceber a importância do projeto com relação aos aspectos de suprimento de energia em satélites e nanossatélites. Portanto, torna-se extremamente necessário o desenvolvimento de mecanismos que possam simular e validar projetos desta natureza.

O projeto pode ser considerado importante, pois este servirá de apoio em projetos, construção, testes, verificação e validação de subsistemas de suprimento de energia bem como o balanço de potência de satélites com classe abaixo de 10 kg. Em decorrência disto, a arquitetura proposta poderá ser avaliada para futura adaptação e uso em satélites de maior escala, cujos subsistemas requerem faixas de potência mais elevadas, tendo como potenciais usuários as missões de satélites do INPE.

Além do mais, o projeto mostrou-se de grande utilidade para a vida acadêmica do bolsista, no qual este pôde aprimorar seus conhecimentos em diversas áreas da engenharia, como também na área de pesquisa.

Com relação aos prazos de entrega, o projeto não pôde ser finalizado por completo, porém ressalta-se que o ponto atingido já é extremamente significativo, pois se trata da parte principal do projeto, o desenvolvimento da primeira versão do emulador, onde o restante pode ser desenvolvido futuramente, com base nessa versão pronta, sem muita dificuldade. Os resultados obtidos serão apresentados no seminário de iniciação científica do INPE.

7. REFERÊNCIAS BIBLIOGRÁFICAS

- Borthakur, R.; S. Narkhede. Modeling of Photovoltaic Array. National Institute of Technology, Rourkea, Índia, Trabalho de Conclusão, 2010.
- Durago, J. Photovoltaic Emulator Adapatable to Irradiance, Temperature and Panel-Specific I-V Curves, Faculty of California Polytechnic State University, Estados Unidos, Dissertação de Mestrado, 2011.
- Marques, C. H. & de Moraes Cioffi, V. Sistema para captação, armazenamento e distribuição de energia solar para aplicações terrestres Universidade do Vale do Paraíba, 1994, 160
- Perondi, L. F.Análise de potência do satélite de coleta de dados 1 (SCD-1). Relatório Técnico, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 1987.
- PV Education. Solar Energy, on-line: http://www.pveducation.org/pvcdrom/instructions, acessado em agosto de 2014.
- Rauschenbach, H. S. Solar cell array design handbook. Van Nostrand Reinhold, 1980.
- Villalva, M. G.; Gazoli, J. R.; Ruppert F. E. Modeling And Circuit-Based Simulation of Photovoltaic Arrays. Brazilian Journal of Power Electronics, v. 14, n. 1, p. 35-45, 2009.

ANEXO 1: Script iparams

```
function [Isc, Imp, Voc, Vmp] = iParams(T,G,n, a, R, Area, Tn, Gn)
% Funcao para gerar parâmetros iniciais utilizados nos scripts icell e vcell
%
% Igor Frassoni  04/2015
%
% Input:
% T ..... Temperatura [°C]
% G ..... Irradiancia [W/m²]
% n[4] ... Vetor de parametro nominais (I e V)
% a[4] ... Vetor de parametro nominais (a's)
% R[4] ... Vetor de parametro nominais (Resistencias)
%
% Output:
% Isc .... Corrente de curto-circuito [mA]
% Imp .... Corrente de maxima potencia [mA]
% Voc .... Tensao de circuito aberto [V]
% Vmp .... Tensao de maxima potencia [V]
%
% Modelo baseado no trabalho do Dr. Leonel Perondi,
% "Analise de Potencia do Satelite de coleta de dados 1 (SCD-1)"
%


%%

Iscn = n(1);
Impn = n(2);
Vocn = n(3);
Vmpn = n(4);
aISC = a(1);
aIMP = a(2);
aVOC = a(3);
aVMP = a(4);
Risc = R(1);
Rimp = R(2);
Rvoc = R(3);
Rvmp = R(4);
%Area = Area;
%Tn = Tn;
%Gn = Gn;


dT = (T-Tn);

Isc = (Iscn + aISC * dT) * Risc * Area;
Imp = (Impn + aIMP * dT) * Rimp * Area;
Voc = (Vocn + aVOC * dT) * Rvoc;
Vmp = (Vmpn + aVMP * dT) * Rvmp;
```

ANEXO 2: Script vcell

```
function V = vcell(i,T,G,n, a, R, Area, Tn, Gn)

% Funcao para para calculo da tensAo na cElula
%
% Igor Frassoni  09/2014
%
% Input:
%
% i ..... corrente [mA]
% T ..... Temperatura [°C]
% G ..... Irradiância [W/m²]
% n[4] .. Vetor de parametro nominais (I e V)
% a[4] .. Vetor de parametro nominais (a's)
% R[4] .. Vetor de parametro nominais (Resistencias)
% Area .. Area do painel solar [m²]
% Tn .... Temperatura nominal [°C]
% Gn .... Irradiância nominal [W/m²]
%
% Output:
% v ..... tensao na celula [V]
%
% Modelo baseado no trabalho do Dr. Leonel Perondi,
% "Análise de Potência do Satélite de coleta de dados 1 (SCD-1)"
%
%v, 28, 1350, [15.05 14.15 2.36 2.085], [0.012 0.013 -0.0048 -0.005],[0.95 1
0.95 0.96], 16, 28, 1353
% utilizar o proprio script para encontrar o ponto maximo
%%

[Isc, Imp, Voc, Vmp] = iParams(T,G,n, a, R, Area, Tn, Gn);

Iscn = n(1);
Impn = n(2);
Vocn = n(3);
Vmpn = n(4);
aISC = a(1);
aIMP = a(2);
aVOC = a(3);
aVMP = a(4);
Risc = R(1);
Rimp = R(2);
Rvoc = R(3);
Rvmp = R(4);

c2 = (Vmp / Voc - 1) / log(1 - Imp / Isc);

c1 = (1 - Imp / Isc) * exp(-Vmp / (c2 * Voc));


%V = log ((((i * Gn)-(Isc * G)-(c1 * Isc * Gn))/(Isc * Gn * (-c1)))) * c2 * Voc;
%V = log (((((G/Gn)-(i/Isc)))/c1)+1) * c2 * Voc;

V = fzero(@(v) icell(v,T,G,[Iscn, Impn,Vocn, Vmpn], [aISC, aIMP, aVOC, aVMP],
[Risc, Rimp, Rvoc, Rvmp], Area, Tn, Gn)-i, 0.01);
```

ANEXO 3: Script icell

```
function i = icell(V,T,G,n, a, R, Area, Tn, Gn)
% Função para para cálculo da corrente na célula
%
% Igor Frassoni  09/2014
%
% Input:
%
% V ..... Tensão [V]
% T ..... Temperatura [°C]
% G ..... Irradiância [W/m²]
% n[4] .. Vetor de parametro nominais (I e V)
% a[4] .. Vetor de parametro nominais (a's)
% R[4] .. Vetor de parametro nominais (Resistencias)
% Area .. Area do painel solar [m²]
% Tn .... Temperatura nominal [°C]
% Gn .... Irradiância nominal [W/m²]
%
% Output:
% i ..... Corrente na célula [mA]
%
% Modelo baseado no trabalho do Dr. Leonel Perondi,
% "Análise de Potência do Satélite de coleta de dados 1 (SCD-1)"
%



%%


[Isc, Imp, Voc, Vmp] = iParams(T,G,n, a, R, Area, Tn, Gn);


c2 = (Vmp / Voc - 1) / log(1 - Imp / Isc);

c1 = (1- Imp / Isc) * exp(-Vmp / (c2 * Voc));

i = Isc * (G / Gn - c1 * (exp(V / (c2 * Voc)) - 1));
```

ANEXO 4: Script plotiv

```
function plotiv(c, plota, T, G)
% Funcao para para calculo da corrente na celula
%
% Igor Frassoni  04/2015
%
% Input:
% c ..... Características da celula ["single junction","double junction","none"]
% plota.. Plotar qual grafico [utilizando vcell ou icell]
% G ..... Irradiancia [W/m²]
% T ..... Temperatura [°C]
%
% Output:
% Grafico de corrente e tensao
%
% Modelo baseado no trabalho do Dr. Leonel Perondi,
% "Analise de Potencia do Satelite de coleta de dados 1 (SCD-1)"
%


%% Verificar o Tipo de Celula a ser usado

    switch c
        case 'double'
            DoubleJunction
        case 'single'
            SingleJunction
        case 'none'

        DoubleJunction   % Alterar aqui caso queira caracteristicas especificas
da celula

        Vocn = input('Entre com o VOC nominal');
        Iscn = input('Entre com o ISC nominal');
        Vmpn = input('Entre com o VMP nominal');
        Impn = input('Entre com o IMP nominal');
    end

%%

[Isc, Imp, Voc, Vmp] = iParams(T,G,[Iscn, Impn,Vocn, Vmpn], [aISC, aIMP, aVOC,
aVMP], [Risc, Rimp, Rvoc, Rvmp], Area, Tn, Gn);

disp(sprintf('variaveis de operacao:\n'));
disp(sprintf(' Temp = %f',T));
disp(sprintf(' Isc = %f',Isc));
disp(sprintf(' Imp = %f',Imp));
disp(sprintf(' Voc = %f',Voc));
disp(sprintf(' Vmp = %f',Vmp));
disp(sprintf('\n'));


%% Case para definir qual script utilizar para o calculo

    switch plota

        case 'vcell'
```

```
            % v(i)
            figure(1)
            iDt = 0.01;
%             maxI = fzero(@(corrente) vcell(corrente,T,G,[Iscn, Impn,Vocn,
Vmpn], [aISC, aIMP, aVOC, aVMP], [Risc, Rimp, Rvoc, Rvmp], Area, Tn, Gn), 0.01)
            maxI = icell(0,T,G,[Iscn, Impn,Vocn, Vmpn], [aISC, aIMP, aVOC,
aVMP], [Risc, Rimp, Rvoc, Rvmp], Area, Tn, Gn)
            i=[0:iDt:maxI];

            V=zeros(1, size(i,2));

            tic
            for k = 1:size(i,2)
                V(k) = vcell(i(k),T,G,[Iscn, Impn,Vocn, Vmpn], [aISC, aIMP,
aVOC, aVMP], [Risc, Rimp, Rvoc, Rvmp], Area, Tn, Gn);
            end
            toc

        case 'icell'

            % i(v)
            figure(2)

            vDt = 0.001;
            maxV = fzero(@(tensao) icell(tensao,T,G,[Iscn, Impn,Vocn, Vmpn],
[aISC, aIMP, aVOC, aVMP], [Risc, Rimp, Rvoc, Rvmp], Area, Tn, Gn), .01);
            V=[0:vDt:maxV];
            i= icell(V,T,G,[Iscn, Impn,Vocn, Vmpn], [aISC, aIMP, aVOC, aVMP],
[Risc, Rimp, Rvoc, Rvmp], Area, Tn, Gn);

    end



%% Plot

grid on
hold on
title('I-V curve');
xlabel('V [V]');
ylabel('I [mA]');

plot(V,i,'LineWidth',2,'Color','k');
plot(V,i);
```

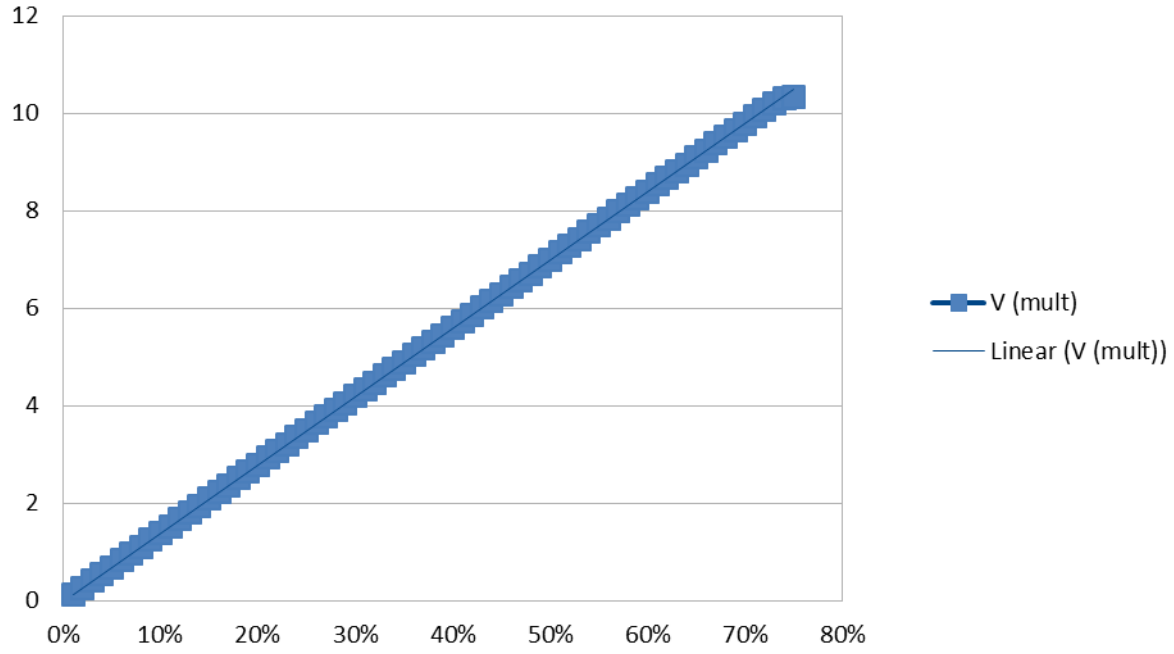
ANEXO 5: Tabela e gráfico de valores de Tensão

Duty Cycle	V (mult)	V (AD)	adc[0]
1%	0,12	57	20
2%	0,26	102	18
3%	0,4	147	18
4%	0,54	185	18
5%	0,68	228	20
6%	0,82	262	18
7%	0,96	302	18
8%	1,1	337	20
9%	1,24	384	19
10%	1,38	434	20
11%	1,52	480	18
12%	1,66	532	17
13%	1,8	552	17
14%	1,94	663	17
15%	2,09	636	20
16%	2,22	668	16
17%	2,37	750	22
18%	2,51	760	20
19%	2,65	796	20
20%	2,79	830	18
21%	2,93	919	22
22%	3,07	920	22
23%	3,21	975	20
24%	3,36	1013	21
25%	3,49	1083	20
26%	3,64	1109	20
27%	3,77	1130	20
28%	3,91	1149	20
29%	4,05	1269	21
30%	4,19	1231	17
31%	4,33	1337	21
32%	4,47	1228	20
33%	4,61	1367	18
34%	4,75	1401	20
35%	4,89	1449	21
36%	5,03	1556	21
37%	5,17	1568	22
38%	5,31	1554	20
39%	5,44	1632	20

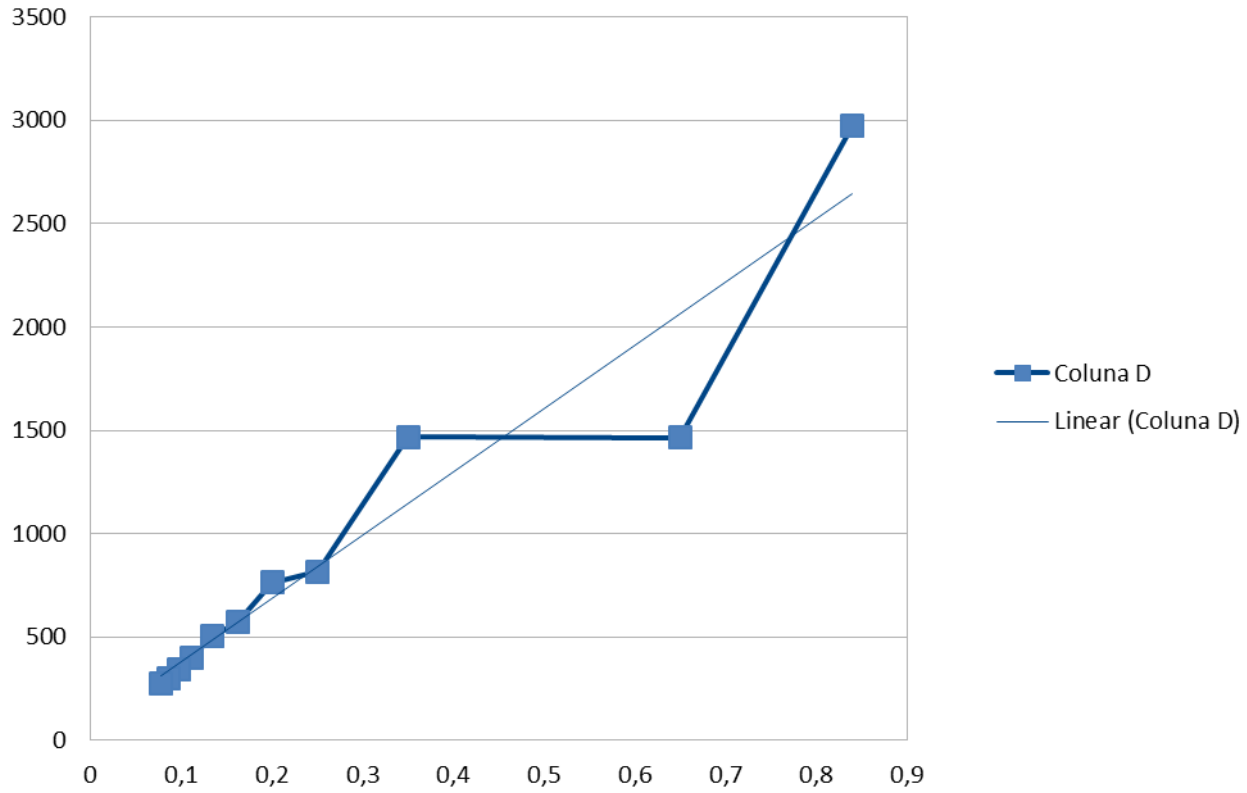
Duty Cycle	V (mult)	V (AD)	adc[0]
40%	5,59	1650	20
41%	5,73	1775	20
42%	5,87	1778	21
43%	6,01	1824	21
44%	6,15	1800	23
45%	6,29	1853	22
46%	6,44	1949	20
47%	6,58	1949	18
48%	6,71	1968	23
49%	6,87	2096	18
50%	7	2056	20
51%	7,14	2161	19
52%	7,28	2156	20
53%	7,42	2210	22
54%	7,56	2253	20
55%	7,7	2287	19
56%	7,84	2363	20
57%	7,98	2393	20
58%	8,12	2490	24
59%	8,25	2457	21
60%	8,4	2485	18
61%	8,54	2621	20
62%	8,68	2637	28
63%	8,82	2661	21
64%	8,95	2632	22
65%	9,1	2704	20
66%	9,23	2760	22
67%	9,38	2786	20
68%	9,52	2888	23
69%	9,65	2858	24
70%	9,79	2903	20
71%	9,93	3027	21
72%	10,07	2998	24
73%	10,21	3021	21
74%	10,3	3096	21
75%	10,33	3096	20
76%	10,33	3100	23
77%	10,33	3096	21
78%	10,33	3096	22

Duty Cycle	V (mult)	V (AD)	adc[0]
79%	10,33	3098	22
80%	10,33	3094	21
81%	10,33	3095	24
82%	10,33	3095	21
83%	10,33	3098	23
84%	10,33	3097	22
85%	10,33	3099	22
86%	10,33	3096	22
87%	10,34	3099	24
88%	10,34	3101	24
89%	10,34	3096	20

Duty Cycle	V (mult)	V (AD)	adc[0]
90%	10,34	3096	19
91%	10,34	3095	20
92%	10,34	3099	21
93%	10,34	3098	23
94%	10,34	3098	20
95%	10,34	3097	22
96%	10,34	3102	22
97%	10,34	3096	23
98%	10,34	3097	20
99%	10,34	3094	21



ANEXO 6: Tabela e gráfico de valores de Corrente



ANEXO 7: Função final da versão 1 do emulador

```
 int CMD_vcell (int argc, char **argv)
{
  float V,i,id,G,T,Iscn,Impn,Vocn, Vmpn, aISC, aIMP, aVOC, aVMP, Risc, Rimp, Rvoc, Rvmp,
Area, Tn, Gn;
  float dT,c1,c2;
  float Isc,Imp,Voc,Vmp;
  float teste,razao;
id=0;
  /* Parametros */


  G=1353;
  T=30;
  Iscn = 15.05;
  Impn = 14.15;
  Vocn = 2.360;
  Vmpn = 2.085;
  aISC = 0.012;
  aIMP = 0.013;
  aVOC = -0.0048;
  aVMP = -0.005;
  Risc = 0.95;
  Rimp = 1.00;
  Rvoc = 0.95;
  Rvmp = 0.96;
  Area = 16;
  Tn = 28;
  Gn = 1353;

  /**/

  //argc_1 = strtoul(argv[1], 0, 10);
  USBFlushTx(true); USBprintf("\n");



  dT = (T-Tn);

    //id = (((dAdc[0]-76.33577)/3058.97488))*1000./3.;   // Corrente em mA
    id = (((dAdc2[0]-76.33577)/3058.97488))*1000./3.;   // Corrente em mA

    USBprintf("Corrente desejada: %u / %u\n", (unsigned long) (id), (unsigned long) (id*3));

    Isc = (Iscn + aISC * dT) * Risc * Area;
    Imp = (Impn + aIMP * dT) * Rimp * Area;
    Voc = (Vocn + aVOC * dT) * Rvoc;
    Vmp = (Vmpn + aVMP * dT) * Rvmp;
```

```
c2 = (Vmp / Voc - 1) / log(1 - Imp / Isc);

c1 = (1- Imp / Isc) * exp(-Vmp / (c2 * Voc));

V = 0;
i = 0;
razao = 0;

USBprintf("Isc: %u\n", (unsigned long) Isc);

if(id > Isc)
{
  USBprintf("ERROR: id > Isc\n");
  SetPwm(5);
  return 0;
}

const float EPS = 1e-3;
const float NMAX = 100;
float ninter = 0;

float vInf = 0;
float vSup = Voc;
float vk = vSup / 2.;

float diff = (Isc * (G / Gn - c1 * (exp(vk / (c2 * Voc)) - 1)));
diff = diff - id;

while (fabs(diff) > EPS && ninter < NMAX)
{
  if(diff > 0)
  {
    vInf = vk;
    vk = (vSup + vk) / 2.;
  } else {
    vSup = vk;
    vk = (vk + vInf) / 2.;
  }

  if(vk == vSup || vk == vInf)
  {
    USBprintf("Nao convergiu (vk = %u)\n", (unsigned long) (vk*1000));
    break;
  }

  diff = (Isc * (G / Gn - c1 * (exp(vk / (c2 * Voc)) - 1)));
  diff = diff - id;

  ninter++;
}
i = (Isc * (G / Gn - c1 * (exp(vk / (c2 * Voc)) - 1)));
```

```
   if(ninter >= NMAX)
     USBprintf("Nao convergiu (diff=%u, i=%u)\n", (unsigned long)(diff*1000), (unsigned
long)(i*1000));



  //    unsigned long Vv = (unsigned long)(V*1000);
  //    USBprintf("V = %u",Vv);

   USBprintf("Tensao de operacao obtida: %u / %u\n", (unsigned long) (vk*1000), (unsigned long)
(vk*4000));

  V=vk*4;


       if (V>10.33)
       {
         teste = 100.0;
       }
       else
       {
         //teste = (V+0.00012)/0.14;
         teste = (V+0.0165)/0.14;
         teste = round(teste);
       }


       unsigned long testee = (unsigned long)teste;

       if (testee>2)
       {
          SetPwm(testee);
       }


       USBprintf("para a corrente: %u mA > tensao:  %u mV || adc[1]: %u\n",(unsigned long)
i,(unsigned long)(V*1000),testee);



  return (0);

}
```