

FUNDAÇÃO UNIVERSIDADE FEDERAL DO ABC
CENTRO DE ENGENHARIA, MODELAGEM E CIÊNCIAS SOCIAIS APLICADAS
TRABALHO DE GRADUAÇÃO II

Frederico Alves de Oliveira Silva

Fonte de Corrente Controlada por Microprocessador com 2 Estágios de Amplificação

São Bernardo do Campo - SP

2018

Frederico Alves de Oliveira Silva

Fonte de Corrente Controlada por Microprocessador com 2 Estágios de Amplificação

Trabalho de Graduação II em Engenharia Bi-
omédica realizado na Fundação Universidade
Federal do ABC.

Orientador: Prof. Dr. Olavo Luppi Silva

São Bernardo do Campo - SP
2018

Resumo

A Tomografia por Impedância Elétrica é uma técnica de imagem médica que necessita de uma fonte de corrente controlada com características próprias de uma fonte de corrente ideal. Assim, o objetivo deste trabalho é dar continuidade ao projeto de uma fonte de corrente controlada por microprocessador que foi iniciado em um trabalho de graduação anterior. Para isto, serão implementados: 2 estágios de amplificação, aumento da resolução do potenciômetro digital, a especificação do amplificador operacional será modificada, o cálculo do algoritmo de demodulação será otimizado e a impedância de entrada do circuito de medição será aumentada.

Sumário

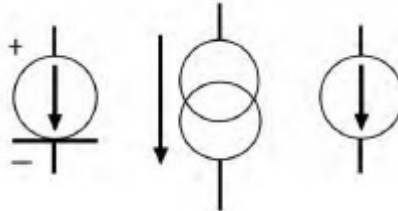
1	INTRODUÇÃO	4
2	OBJETIVOS	6
2.1	Objetivos Gerais	6
2.2	Objetivos Específicos	6
3	REVISÃO DE LITERATURA	7
3.1	Fonte de Corrente Microprocessada para Aplicação na TIE	7
4	FUNDAMENTAÇÃO TEÓRICA	10
4.1	Microprocessadores	10
4.2	<i>Beaglebone Black</i>	12
4.3	Controle de malha fechada	14
4.4	Demodulação	18
5	METODOLOGIA	20
5.1	Materiais	20
5.2	Procedimento	20
5.2.1	Montagem do circuito base	20
5.2.2	Montagem do circuito com 2 estágios de amplificação	20
5.2.3	Teste do circuito	23
5.2.4	Implementação do Algoritmo de Demodulação	23
5.2.5	Teste da fonte de corrente	24
6	RESULTADOS	25
6.1	Teste do potenciômetro digital	25
6.1.1	<i>Serial Peripheral Interface (SPI)</i>	25
6.1.2	Habilitação da <i>Serial Peripheral Interface</i>	26
6.1.3	Teste da <i>Serial Peripheral Interface</i>	29
	REFERÊNCIAS	31
7	ANEXOS	32

1 INTRODUÇÃO

Independentemente de onde estivermos estamos cercados por equipamentos elétricos. Estas máquinas geralmente são alimentadas por dispositivos chamados fontes. Existem fontes de tensão e fontes de corrente elétrica, que tem como objetivo fazer os equipamentos elétricos funcionarem por meio do fornecimento contínuo de tensão e corrente elétrica. Uma boa aproximação para uma fonte de tensão são as baterias, ou pilhas, que funcionam em conjunto com diversos aparelhos. No caso das fontes de corrente não há aproximações deste tipo. Apesar disto, elas têm grande importância para equipamentos eletromédicos.

Em um diagrama que representa um circuito elétrico, uma fonte de corrente pode ser representada por meio da utilização de um símbolo em que há uma seta dentro de uma circunferência ou, ainda, duas circunferências com as bordas que se cruzam, Figura 1

Figura 1 – Diversos símbolos de fontes de corrente ideal.



Fonte: *Current and Voltage Sources*.

Assim, as fontes de corrente reais podem ser definidas como bipolos ativos que possuem a capacidade de fornecer um determinado valor de corrente elétrica, que varia em função do tempo, para um outro circuito que está conectado a elas (1). As fontes de corrente reais são tidas como blocos constituintes de circuitos eletrônicos utilizados na arquitetura de circuitos integrados (CIs) e placas de circuitos originais de fabricantes (COF ou *Original Electronic Manufacturers - OEM*). Elas, geralmente, são constituídas de resistores, transistores e diodos (*Bipolar Junction Transistors - BJTs* e *Field Effect Transistors - FETs*). Entretanto, há também a possibilidade de construí-las com a utilização de amplificadores operacionais (AmpOps) com fontes de tensão precisas (1).

As fontes de corrente surgiram há pelo menos duas décadas antes dos CIs. Elas eram, inicialmente, aplicadas em circuitos de tubos à vácuo. Quando os transistores se tornaram disponíveis na década de 60, os projetistas foram capazes de desenvolver fontes de corrente que eram conectadas em trilhas de suprimento positivas ou negativas. Então, quando o CI bipolar de silicone foi criado no final dos anos 60, as fontes de corrente já eram parte integral da arquitetura interna de CIs e conseguiam realizar alimentação com estabilidade (1).

Na área de instrumentação biomédica as fontes de corrente possuem diversas aplicações. Elas podem, por exemplo, ser utilizadas na alimentação de amplificadores operacionais que atuam em conjunto com sensores no processo de recuperação de sinais de baixo

nível (1). Uma outra aplicação para as fontes de corrente seria em um sistema medidor de impedância elétrica aplicado no monitoramento de variáveis fisiológicas como, por exemplo, a taxa de respiração por meio da pneumografia por impedância (2). Elas ainda podem, de acordo com (3), ser aplicadas na técnica de Tomografia por Impedância Elétrica (TIE) onde cada pixel na imagem corresponde a um valor de impeditividade elétrica e, assim, para que seja possível obter uma imagem de boa qualidade se faz necessário o fornecimento de corrente elétrica aproximadamente constante.

Este projeto trata do desenvolvimento de uma fonte de corrente controlada por microprocessador com 2 estágios de amplificação e, ele está dividido nas seguintes seções: objetivos, revisão da literatura, fundamentação teórica, metodologia, resultados e discussões, conclusão e anexos. Na próxima seção deste documento podem ser vistos os objetivos geral e específicos do projeto. A revisão de literatura nos mostra um trabalho de graduação anterior em que foi desenvolvida uma fonte de corrente microprocessada para TIE. A fundamentação teórica consiste em apresentar temas sobre: microprocessadores, a plataforma de desenvolvimento *BeagleBone Black*, uma revisão a respeito de sistemas de controle em malha fechada e o conceito de demodulação. A parte seguinte aborda a metodologia que será adotada para a execução deste trabalho, assim, nesta seção há o procedimento que será utilizado. As próximas seções que a serem adicionadas neste documento nos próximos quadrimestres são: resultados e discussões, conclusão e a complementação da seção de anexos em que já consta o cronograma de trabalho e que no futuro haverá o algoritmo de demodulação.

2 OBJETIVOS

2.1 Objetivos Gerais

O propósito deste trabalho de graduação é realizar a otimização de uma fonte de corrente elétrica controlada por microprocessador por meio da utilização de 2 estágios de amplificação para que, assim, esta fonte possa fornecer corrente elétrica próxima a uma fonte de corrente ideal.

2.2 Objetivos Específicos

Para que seja possível alcançar os resultados esperados, os objetivos específicos deste trabalho são: implementar 2 estágios de amplificação, aumentar a resolução do potenciômetro digital, modificar a especificação do amplificador operacional, otimizar o cálculo de demodulação e aumentar a impedância de entrada do circuito de medição.

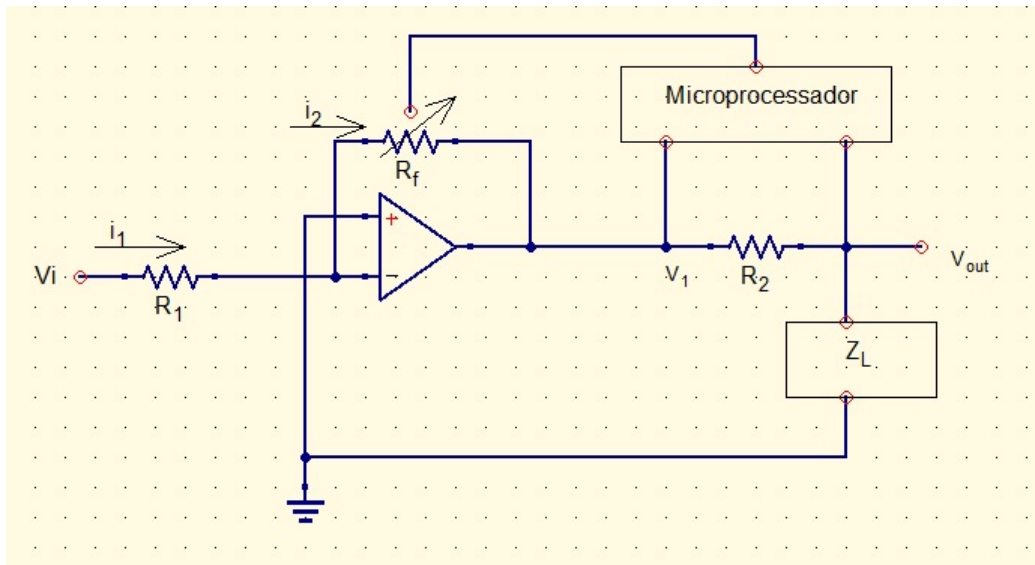
3 REVISÃO DE LITERATURA

3.1 Fonte de Corrente Microprocessada para Aplicação na TIE

O autor do trabalho (3), abordou a aplicação da fonte de corrente elétrica controlada por microprocessador na técnica de Tomografia por Impedância Elétrica, com o propósito de sanar a dificuldade de manter a corrente elétrica gerada a um valor próximo do ideal. A vantagem desta técnica em relação a tomografia convencional é que ela não utiliza radiação ionizante, isto é, esta técnica não tem potencial de ionizar átomos no tecido biológico e, assim, não há riscos à saúde do paciente. A técnica TIE obtém imagens da distribuição de impeditividade elétrica, através da solução de um problema inverso que tem como entrada medidas de potencial e de corrente elétrica que atravessam o corpo do paciente, feitas por eletrodos fixados sobre sua pele. Assim, cada pixel da imagem corresponde a um valor de impeditividade elétrica.

O circuito utilizado em (3), possuía, inicialmente, como componentes dois resistores R_1 e R_2 , um amplificador operacional (AmpOp) na configuração inversora, um potenciômetro digital (R_f), uma carga com impeditividade indutiva Z_L e um microcontrolador, Figura 2.

Figura 2 – Circuito básico para uma fonte de corrente.



Fonte: Desenvolvimento de Fonte de Corrente Controlada por Microprocessador para Aplicação na Tomografia por Impedância Elétrica.

Se utilizarmos a lei de Kirchhoff das correntes no nó da entrada inversora do AmpOp, considerando como condições de simplificação dos cálculos:

$$\begin{aligned} i_p &= i_n = 0; \\ V_p &= V_n. \end{aligned}$$

nós temos que:

$$i_1 = i_2 \tag{3.1}$$

$$\frac{V_i - V_n}{R_1} = \frac{V_n - V_1}{R_f}$$

como o terminal V_p está aterrado e tem potencial igual a 0 V, o valor de V_n também é $V_n = 0$. Assim, temos

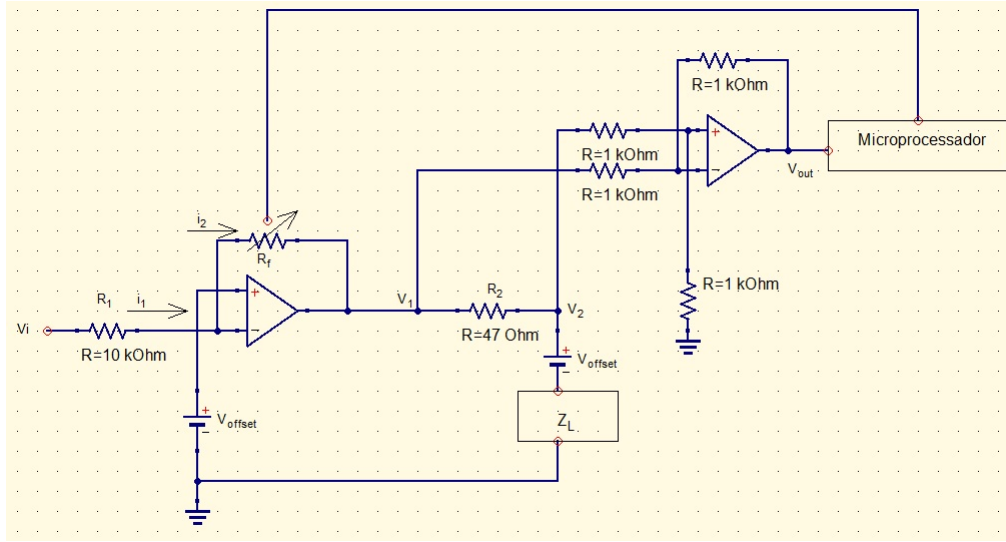
$$\frac{V_i}{R_1} = -\frac{V_1}{R_f}$$

$$V_1 = -\frac{R_f}{R_1} V_i \tag{3.2}$$

Basicamente, o que este circuito faz é receber uma tensão de entrada V_i de alta frequência e, então, a corrente i_1 entra no nó da entrada inversora e a corrente i_2 sai deste nó e vai para o nó com potencial V_{out} . O microprocessador calcula a corrente i_2 que passa pelo resistor R_2 e ajusta-a por meio do potenciômetro R_f com o intuito de fazê-la variar o mínimo possível para que o comportamento deste circuito gerador de corrente seja próximo a uma fonte de corrente ideal.

Entretanto, o circuito desenvolvido para atuar com o microprocessador teve de ser modificado devido a uma limitação da tensão da entrada analógica da *BeagleBone Black* ser no máximo 1.8 V. Para isto, foi acrescentado mais um amplificador operacional na configuração diferencial com o intuito de medir a tensão no resistor sentinela R_2 e, além disso, foram adicionadas mais duas fontes próximas da carga Z_L para adicionar uma tensão de $V_{offset} = 0.36$ V.

Figura 3 – Circuito desenvolvido para ser uma fonte de corrente controlado por microprocessador.



Fonte: Desenvolvimento de Fonte de Corrente Controlada por Microprocessador para Aplicação na Tomografia por Impedância Elétrica.

As tensões de saída V_1 e V_{out} foram calculadas conforme as equações a seguir:

$$V_1 = -\frac{R_f}{10 \text{ k}\Omega} V_i \quad (3.3)$$

$$V_{out} = V_2 - V_1 \quad (3.4)$$

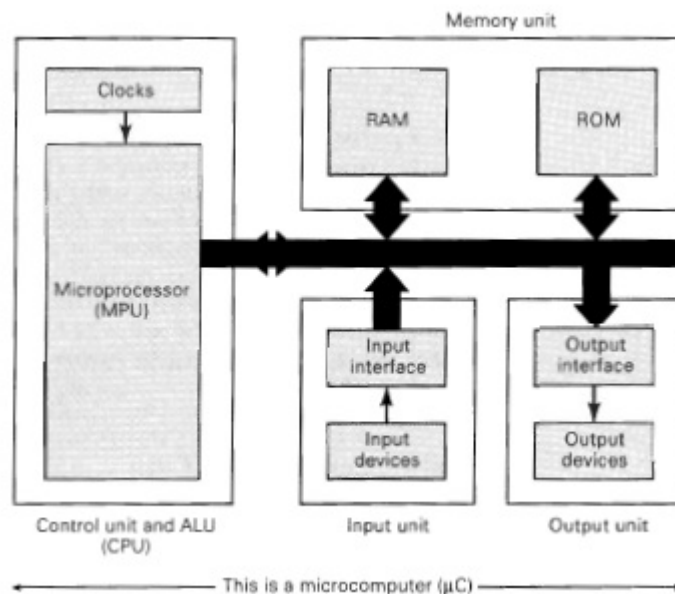
Os resultados obtidos pela fonte de corrente desenvolvida em (3) demonstram a viabilidade desta ideia para a sua aplicação na técnica de TIE. Dentre eles, destaca-se que dos dois microprocessadores utilizados, *Raspberry Pi Model B+* e o *Beaglebone Black*, o melhor microprocessador foi o *Beaglebone Black* devido a taxa de aquisição de dados ter sido realizada na frequência de, aproximadamente, 120 ksps (kilo *samples per seconds* - kilo amostras por segundo), que é um valor próximo ao esperado que foi estimado em 110 ksps. Este valor foi obtido após a utilização da expansão da memória e da mudança do padrão estipulado pelo modo *PRU* imposto pela plataforma. A placa *Raspberry Pi Model B+* foi preterida por causa da sua taxa de aquisição de dados ter sido abaixo do valor esperado. Contudo, apesar dos resultados obtidos com a plataforma *BeagleBone Black*, o autor de (3) apontou diversas melhorias que ainda podem ser aplicadas para otimizar o funcionamento desta fonte de corrente, estas ações sugeridas serão aplicadas neste projeto.

4 FUNDAMENTAÇÃO TEÓRICA

4.1 Microprocessadores

Os microprocessadores estão presentes em diversos equipamentos e aparelhos utilizados no dia-a-dia, dentre eles: computadores, telefones celulares, sistemas de alarme, carros e outros. Antes de definirmos o que é um microprocessador se faz necessário entender um pouco sobre arquitetura de computadores digitais para que assim, seja possível diferenciá-los de um microprocessador. Um computador digital é um sistema que consiste de cinco partes: unidade lógica aritmética (*Aritmetic Logic Unit - ALU*), unidade de controle, unidade de memória, unidade de entrada e unidade de saída (4), Figura 4.

Figura 4 – Estrutura de um microcomputador com suas cinco partes. À esquerda observa-se duas unidades juntas, a unidade lógica aritmética e a unidade de controle. À direita na parte superior está a unidade de memória. Na parte inferior e ao centro está a unidade de entrada. E, na parte inferior e à direita está a unidade de saída.



Fonte: *Digital Systems (4)*.

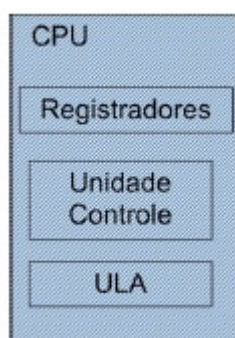
A ALU tem por função a realização de operações lógicas e aritméticas. A unidade de controle é responsável pelo direcionamento da operação de todas as outras unidades, ela fornece sinais de controle e de tempo, além disso, esta unidade também é responsável por transportar a informação da memória para o processador e de ler comandos da unidade de memória. O propósito da unidade de memória é armazenar dados que podem representar programas que o computador executa e também dados que são interpretados por ele. Uma outra função da unidade de memória é servir como um contêiner para resultados de operações aritméticas. A unidade de entrada é composta de dispositivos responsáveis pela

entrada de dados para o sistema, isto é, estes componentes levam a informação e os dados externos ao computador para a memória ou para a ALU. A última unidade, a unidade de saída, é constituída de dispositivos que levam os dados e a informação do computador para o mundo exterior (4).

Visto isto, agora é possível definirmos o que é um microprocessador. Um microprocessador é um chip que contém a junção da ALU com a unidade de controle, recebendo o nome de Unidade de Processamento Central (*Central Processing Unit - CPU*). A CPU é o resultado da junção da ALU com a unidade de controle. Assim, a diferença entre um computador digital e um microprocessador é que o microprocessador é uma parte do computador, isto é, o microprocessador é um subconjunto do sistema que forma um computador (4).

A unidade que compõe o microprocessador é feita de circuitos lógicos e possui três seções: a seção de tempo e controle (unidade de controle), a seção de registro, e a unidade lógica aritmética, Figura 5 . E cada uma destas três partes possui uma função específica. A seção de tempo e controle tem a função de trazer dados da memória e interpretar seus códigos e, então, gerar os sinais de controle requeridos por outra unidade de microprocessamento. A seção de registradores é responsável pelo transporte dos dados do microprocessador. Um microprocessador deve possuir uma grande quantidade de registradores, devido ao lento acesso à memória que é externa ao chip. A ALU realiza operações aritméticas e lógicas com os dados, dentre elas estão as operações de soma, multiplicação, divisão e, operações utilizando funções *AND*, *OR* ou *XOR* (4).

Figura 5 – Microprocessador ou unidade central de processamento. No esquema observa-se que a CPU é composta de registradores, unidade de controle e unidade lógica de aritmética.



Fonte: Microprocessadores e Microcontroladores.

As aplicações dos microprocessadores são variadas, dentre elas destacam-se:

- Fornecer sinais de tempo e controle para todos os elementos do computador;
- Fazer o transporte de instruções e dados da memória e para ela;
- Realizar operações lógicas e aritméticas;

- Decodificar instruções.

É importante observar que o aumento do desempenho dos microprocessadores depende de alguns fatores: incremento do número interno de bits, aumento do número externo de bits, redução do número de ciclos para executar cada instrução, aumento da capacidade e velocidade da memória cache e, aumento de *clock*. O *clock* é uma onda retangular que é distribuída para todas as partes do sistema e determina o momento exato de mudança na saída de sistemas digitais síncronos. Desta forma o *clock* fica responsável pelo sincronismo entre as unidades de processamento internas ao microprocessador e pelas unidades externas. Quanto maior a frequência de *clock* mais veloz é o processamento, entretanto, se o aumento for demasiadamente grande é possível perceber aumento de temperatura no sistema. Com relação ao número de bits, o seu incremento interno e externo possibilita um maior trânsito de bits por unidade de tempo. A ampliação da capacidade e velocidade da memória cache possibilita um aumento da velocidade de transferência de dados entre a CPU e a memória principal e, com isso, ocorre uma melhora no desempenho do microprocessador (4),(5).

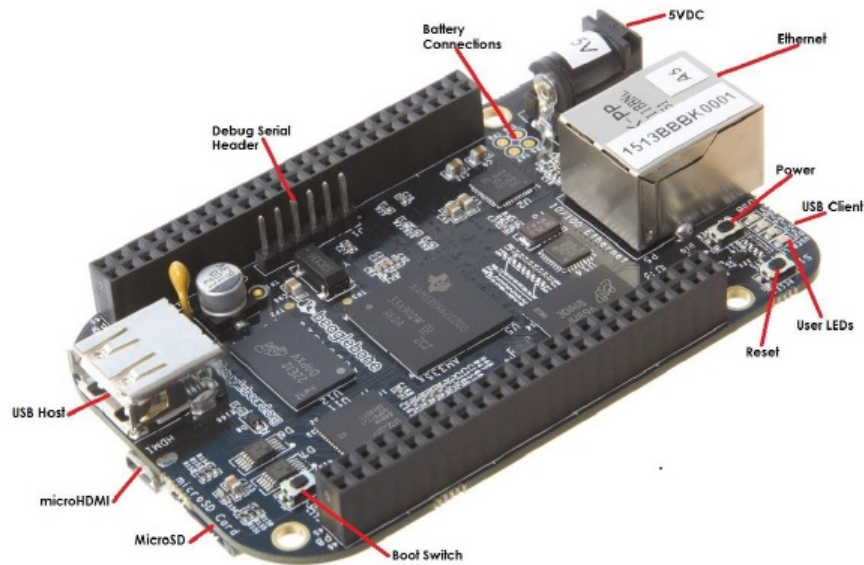
4.2 *Beaglebone Black*

A plataforma de desenvolvimento selecionada para ser aplicada neste trabalho de conclusão de curso é a *BeagleBone Black*. Ela foi selecionada devido ao seu bom desempenho e, por poder ser implementada em um modo chamado PRU (*Programmable Real-time Units*). Esta placa possui o processador Sitara AM3358BZCZ100 com 1 GHz, isto é, ela é capaz de realizar 1 bilhão de operações por segundo. A *BeagleBone Black* é uma plataforma de desenvolvimento de baixo custo que foi projetada para realizar as mesmas tarefas que um computador realiza.

Inicialmente, esta placa foi projetada pela *Circuitco LLC* em *Richardson Texas* pelo colaborador da empresa *Texas Instruments* chamado Gerald Coley que também é um membro da comunidade *BeagleBoard.org* (6).

Seu software é desenvolvido por pessoas da comunidade *Open Source* e colaboradores das empresas *Texas Instruments*, *DigiKey* e *Circuitco* (6). A plataforma *BeagleBone Black* é capaz de utilizar diversos sistemas operacionais, dentre eles o *Linux* e o *Android* (7).

Esta placa permite a adição de outras placas-filhas que são chamadas de "*capex*" com o propósito de combinar várias características destes dispositivos, ou ainda, ela permite ser conectada a outros circuitos. A Figura 6, a seguir mostra uma imagem desta plataforma de desenvolvimento e seus componentes.

Figura 6 – Esquema da plataforma de desenvolvimento *BeagleBone Black*.

Fonte: *BeagleBone Black System Reference Manual* (6).

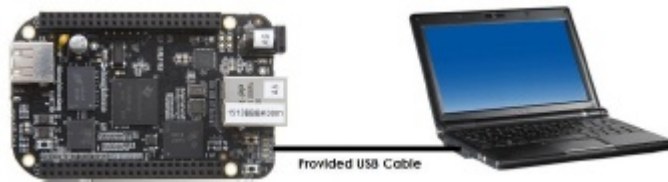
Por meio da Figura 6 que foi reproduzida de (6) é possível identificar que a *BeagleBone Black* possui os componentes a seguir:

- Fonte DC: trata-se de uma entrada para fonte DC e que possui tensão nominal de 5 V;
- *Power Button*: este é o botão que liga ou desliga a plataforma;
- *Ethernet*: é o componente que realiza a conexão com a rede LAN;
- *Serial Debug*: é uma porta que é responsável pela tarefa denominada como *debug*;
- *USB Client*: este componente é um conector USB para conectar a placa a um computador para que assim seja possível que o computador possa ser utilizado para alimentar esta plataforma;
- *BOOT Switch*: é uma chave que pode ser utilizada para forçar a inicialização do cartão microSD em casos de queda ou oscilação de energia. Assim, este componente é responsável por remover a energia elétrica da placa e reaplicá-la novamente na plataforma;
- *LEDs*: são pequenos *LEDs* que servem para sinalização;
- Botão *Reset*: é um botão que permite a operação de reinicialização do microprocessador;
- MicroSD: este componente é um conector específico para que seja possível colocar nele o cartão microSD;

- MicroHDMI: é um conector no qual algum *display* pode ser conectado;
- USB *Host*: trata-se de uma porta USB através da qual pode ser conectada por diferentes interfaces USB, sejam elas, *wi-fi*, teclado e etc.

A placa pode funcionar de duas maneiras: conectada a um computador por meio do cabo USB ou quando ela faz o papel de computador e nela são conectados o teclado, mouse, uma fonte de 5 V e um monitor. Neste trabalho, vamos usar a plataforma na primeira configuração para inserir o código de demodulação do computador para a placa, conforme a Figura a seguir:

Figura 7 – Ilustração de como a *BeagleBone Black* pode funcionar quando ela está conectada a um computador.



Fonte: *BeagleBone Black System Reference Manual*.

Os passos para conectar a *BeagleBone Black* a um computador, de acordo com (6), são :

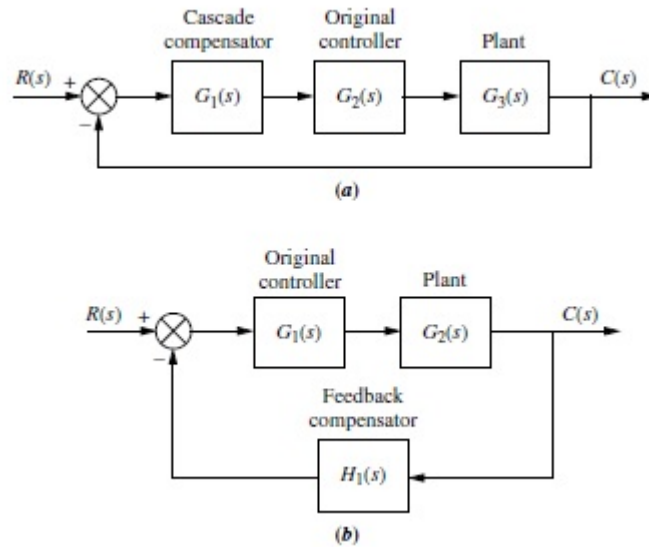
- Conectar o conector pequeno do cabo USB à porta USB da placa;
- Conectar o conector maior à porta USB do computador;
- Esperar até que os *LEDs* acendam;
- Acessar a pasta *USB Drive* e abrir o arquivo *start.html*.

4.3 Controle de malha fechada

Neste trabalho será aplicado o conceito de controle em malha fechada no sistema (circuito), para que se possa controlar a resposta transitória e o erro de regime permanente. Para isto o componente Z_L do circuito mostrado na Figura 15 será modificado. A melhoria da resposta transitória e do erro de regime permanente em um sistema é possível por meio da adição de pólos e zeros para que o sistema seja compensado. No caso da resposta transitória, a melhoria pode vir na forma de um acréscimo de derivação, enquanto que no caso do erro de regime permanente a mudança ocorre por meio da inserção de integração no sistema (8). Entretanto, estas implementações que foram citadas produzem estes resultados somente em controladores Proporcional, Integral e Derivativo (PID), estes controladores serão explicados adiante.

Para aplicarmos controle em nosso sistema podemos fazer uso de compensadores de forma que eles atuem junto dos controladores originais. Assim, há dois tipos de configuração para estes compensadores: a compensação em cascata e a configuração em realimentação, Figura 8.

Figura 8 – Configurações disponíveis para um sistema. Configuração em cascata aparece na parte superior e a configuração em realimentação é observada na parte inferior.



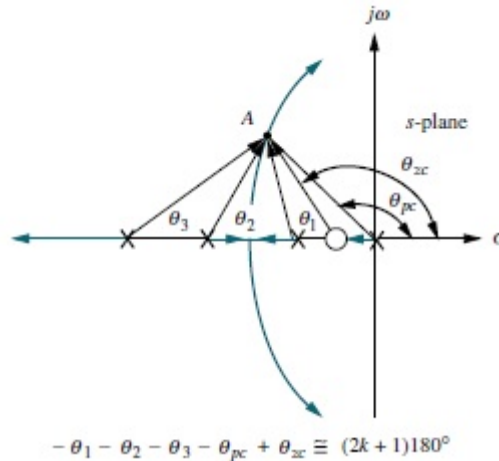
Fonte: *Control Systems Engineering* (8).

A Figura 8 nos mostra que na configuração em cascata, o compensador aparece antes da planta com uma função de transferência genérica $G_1(s)$ que é multiplicada pelo controlador original $G_2(s)$ e pela planta $G_3(s)$. O compensador em realimentação aparece no ramo da realimentação do diagrama de blocos da Figura, nela ele aparece em outro ramo do sistema como uma função de transferência genérica $H_1(s)$ em um ramo de *feedback* que realimentará o sistema, assim, $H_1(s)$ estará na função de transferência geral do sistema $T(s)$ em malha fechada multiplicando as funções de transferência do controlador original $G_1(s)$ e da planta $G_2(s)$ no denominador de $T(s)$. Ambos os métodos fazem os pólos e zeros adicionados em malha aberta passarem no lugar das raízes em uma malha fechada (8). Neste trabalho, abordaremos apenas a configuração em cascata.

De maneira mais precisa, para podermos reduzir o erro de regime permanente para zero (ou um valor muito próximo a ele) sem modificar de maneira significativa a resposta transitória do sistema, introduzimos um pólo na origem do plano s juntamente de um zero próximo a ele, Figura 9. Ao, acrescentarmos o pólo na origem do plano, nós conseguimos reduzir o erro de regime permanente, contudo, o lugar das raízes não passará pelo ponto que necessitamos, assim, com a adição de um zero próximo ao pólo, o lugar das raízes passa pelo ponto que queríamos e, com isso, obtemos a redução do erro de estado estacionário sem reduzir significativamente a resposta transitória. Este compensador que possui um

pólo na origem do sistema e um zero próximo a ele é chamado de compensador integral ideal ou controlador PI, porque há um ganho proporcional K_1 e o ganho do integrador $\frac{K_2 a}{s}$ (8).

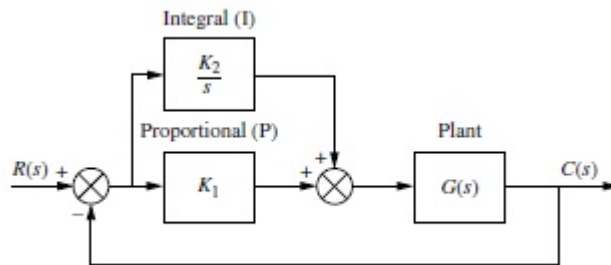
Figura 9 – Plano s contendo o pólo adicional na origem e o zero próximo a ele fazendo com que o lugar das raízes passe por um ponto A arbitrário.



Fonte: *Control Systems Engineering*.

O diagrama de blocos de um sistema genérico utilizando controlador PI pode ser visto na Figura 10.

Figura 10 – Diagrama de blocos de um sistema genérico que utiliza um controlador proporcional integral (PI).



Fonte: *Control Systems Engineering*.

De acordo com o diagrama de blocos da Figura 10, a função de transferência do sistema com o controlador PI é dada por:

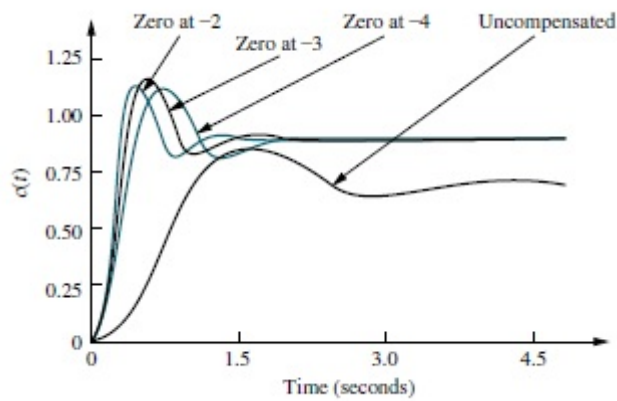
$$G_{PI}(s) = K_1 + \frac{K_2}{s} \quad (4.1)$$

Para que se obtenha um sistema com resposta transitória desejável, isto é, com a ultrapassagem percentual requerida e um menor tempo de assentamento do que o obtido com o sistema sem compensação, utiliza-se a chamada compensação derivativa ideal ou controlador PD (controlador Proporcional Derivativo). O controlador PD consiste em

um derivador puro que é colocado em paralelo com um controlador proporcional antes da planta. Um possível efeito negativo deste compensador é que ele pode introduzir ruído na resposta final do sistema, a frequência deste ruído adicional pode ser da mesma ordem de magnitude que o sinal desejado (8).

O aumento da velocidade do sistema se dá por meio da introdução de um zero antes da planta. O efeito das posições do zero na velocidade do sistema pode ser visto na Figura 11 a seguir:

Figura 11 – Efeito dos zeros na velocidade do sistema. Quanto mais próximos do eixo imaginário mais rápido é o sistema.

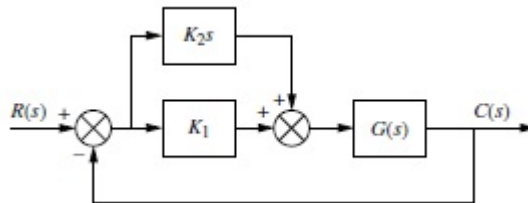


Fonte: *Control Systems Engineering*.

Ao analisarmos a Figura 11 podemos ver que para um sistema com ordem maior que 2 por exemplo, quanto mais perto do eixo imaginário, mais rápido o sistema fica. Apesar deste fato, nem sempre uma melhora na resposta transitória produz uma melhora no erro de regime permanente.

A seguir podemos observar o diagrama de blocos básico para a aplicação do controlador PD e, sua respectiva função de transferência:

Figura 12 – Diagrama de blocos para um sistema genérico que utiliza um controlador PD.



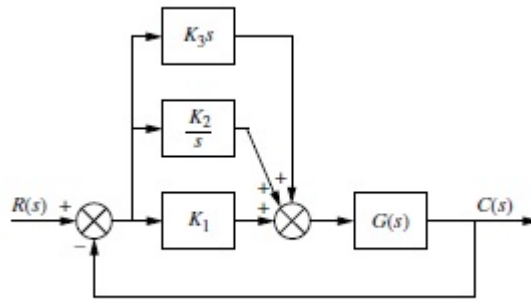
Fonte: *Control Systems Engineering*.

$$G_{PD}(s) = K_2s + K_1 \quad (4.2)$$

na equação 4.2 o termo K_1 representa o controle proporcional e o termo K_2 representa o ganho do controle derivativo.

Um controlador proporcional, integral e derivativo (PID) consiste na combinação entre um controlador PD e um PI, nesta ordem. Assim, primeiramente, obtemos a melhora na resposta transitória do sistema para então fazer a redução no erro de regime permanente. É importante ressaltar que ao fazer estas correções ocorre uma perda de velocidade durante a correção do erro de estado estacionário (8). O diagrama de blocos do controlador PID e a sua função de transferência podem ser observados na Figura 13 e na equação a seguir, respectivamente:

Figura 13 – Diagrama de blocos do controlador PID.



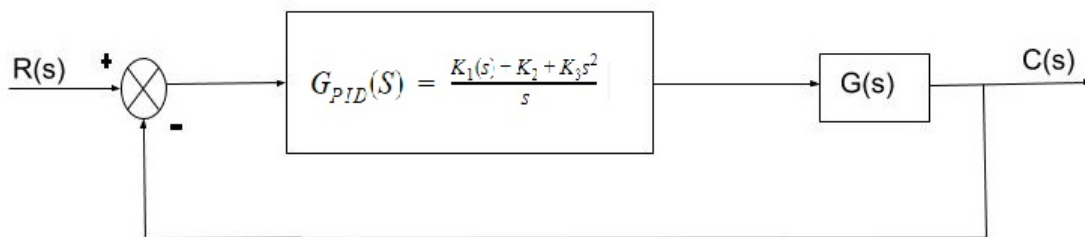
Fonte: *Control Systems Engineering*.

$$G_{PID}(s) = K_1 + \frac{K_2}{s} + K_3s = \frac{K_1s + K_2 + K_3s^2}{s} \quad (4.3)$$

onde: K_1 representa o ganho proporcional, K_2 representa o ganho integral e K_3 representa o ganho derivativo.

De forma simplificada o diagrama de blocos do controlador PID pode ser representado como na Figura 14 a seguir:

Figura 14 – Diagrama de blocos do controlador PID com a função de transferência $G_{PID}(s)$ incluída em um bloco de forma simplificada.



4.4 Demodulação

A modulação é necessária quando desejamos transmitir um sinal de baixa frequência e isto não é possível justamente pelo fato da onda ter esta baixa frequência, então,

multiplicamos esta onda de baixa frequência por uma onda de alta frequência (onda portadora) que carregará este sinal. Este processo de modulação tem por finalidade alterar a amplitude, ou a frequência ou a fase deste sinal para que ele possa ser transmitido (9).

Dados dois sinais senoidais I/Q (*In-phase* e *Quadrature*) deslocados 90° em fase, sendo o sinal I uma onda cossenoidal e o sinal Q uma onda senoidal, dizemos que estes sinais estão em quadratura. A modulação por quadratura é um processo que se baseia na soma de dois sinais que estão em quadratura, isto é, sinais I/Q portadores são gerados por um oscilador e multiplicados pelos sinais de I/Q de fluxo de dados (sinais modulantes), e logo após esta etapa, os sinais I/Q portadores que estão misturados com os sinais modulantes I/Q são somados. Isto permite que dois bits sejam transmitidos por uma onda (9).

Após o processo de modulação, para obtermos o sinal modulante da onda portadora, ela passa por um processo chamado demodulação que nada mais é do que o processo de obtenção da frequência, da fase e da amplitude do sinal modulante que estava misturado com a onda portadora.

No caso da demodulação por quadratura temos que ao recebermos uma onda, ela é multiplicada por um sinal de um oscilador contendo de um lado uma onda I e de outro lado uma onda Q . Estes sinais então passam por um filtro passa-baixas para que se possa então extrair os sinais modulantes I e Q . Este processo então, depende de ondas referência I e Q que saem do oscilador e da presença de dois filtros passa-baixas um para cada onda (9).

5 METODOLOGIA

5.1 Materiais

Foram utilizados os materiais listados a seguir:

Tabela 1 – Materiais utilizados.

Material	Quantidade
Jumpers	1 Pacote
BeagleBone Black	1
Cabo USB	1
Potenciômetro Digital Microchip MCP41010	1
Computador	1

5.2 Procedimento

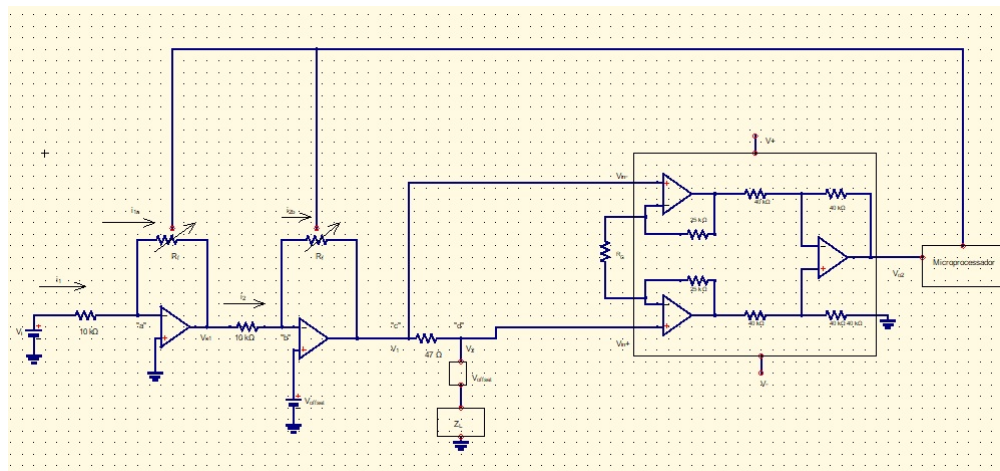
5.2.1 Montagem do circuito base

Em um primeiro momento será montado o circuito realizado no trabalho em (3), para entender o funcionamento do algoritmo de demodulação e reproduzir os resultados obtidos por (3).

5.2.2 Montagem do circuito com 2 estágios de amplificação

Nesta etapa será montado o circuito da Figura 15 que foi adaptado para ter dois estágios de amplificação. O modelo base é o circuito da Figura 3 utilizado em (3).

Figura 15 – Circuito da fonte de corrente com 2 estágios de amplificação adaptado do modelo realizado no trabalho (3).



Este circuito possui 2 amplificadores operacionais do tipo LM741 e um amplificador de instrumentação INA129. Os dois AmpOps (lado esquerdo e centro da Figura 15)

estão na configuração inversora e o último amplificador mais a direita funciona como um amplificador diferencial. O primeiro amplificador operacional está conectado a um resistor de $10\text{ k}\Omega$ na entrada inversora e a um potenciômetro digital, R_f , no ramo de *feedback* negativo e, a entrada não inversora está aterrada. O segundo amplificador está conectado a um resistor de $10\text{ k}\Omega$ na entrada inversora que está diretamente ligada na saída do primeiro amplificador e também, haverá um potenciômetro digital ligado a este amplificador no ramo de *feedback* negativo. Na entrada não-inversora do segundo amplificador há uma fonte de tensão com potencial V_{offset} que está com seu terminal negativo aterrado. O terceiro e último amplificador está com o terminal inversor conectado ao nó "c" que possui potencial V_1 e ele está ligado na saída do segundo amplificador. O terminal não-inversor do amplificador de instrumentação estará conectado a um resistor de 47Ω no nó "d" e este nó tem potencial V_2 , além disso, ele estará ligado a uma fonte de tensão com tensão com potencial V_{offset} e também a uma carga com impedância Z_L . Como dito antes, entre os nós com potenciais V_1 e V_2 haverá um resistor de 47Ω , que servirá para medir a diferença de potencial que estará na saída do INA129, V_{o2} . Na saída do amplificador de instrumentação cujo o potencial é V_{o2} haverá um microprocessador conectado. A outra entrada do microprocessador estará conectada aos potenciômetros cujos os valores de resistência são R_{f1} e R_{f2} .

O circuito foi projetado de tal maneira que, a corrente elétrica fluirá da fonte de tensão V_i , passando pelo resistor de $10\text{ k}\Omega$, em seguida, pelo potenciômetro R_{f1} , até chegar na saída V_{o1} do primeiro amplificador operacional. A partir desta etapa, a corrente elétrica seguirá um caminho similar para o segundo amplificador operacional, i.e., ela seguirá através do resistor de $10\text{ k}\Omega$, depois passará pelo potenciômetro digital R_{f2} e chegará na saída V_1 do segundo amplificador operacional. Então o INA129 utilizará as tensões dos nós "c" e "d" para produzir a tensão V_{o2} para o microprocessador que, exercerá a tarefa de controlar os valores das resistências dos potenciômetros de acordo com o valor de tensão V_{o2} .

Baseado no caminho determinado para a corrente elétrica, a tensão V_{o1} é determinada por meio da aplicação da Lei de Kirchhoff das Correntes (LKC) no nó "a", considerando como condições de simplificação

$$i_{an} = i_{ap} = 0$$

$$V_{an} = V_{ap}$$

temos, então pela LKC em "a"

$$i_1 = i_{an} + i_{1a} \quad (5.2)$$

$$i_1 = i_{1a} \quad (5.3)$$

sendo

$$i_1 = \frac{V_i - V_{an}}{10 \ k\Omega} \quad (5.4)$$

$$i_{1a} = \frac{V_{an} - V_{o1}}{R_{f1}} \quad (5.5)$$

aplicamos as equações 5.4 e 5.5 na equação 5.3 e obtemos

$$\begin{aligned} \frac{V_i - V_{an}}{10 \ k\Omega} &= \frac{V_{an} - V_{o1}}{R_{f1}} \\ \frac{V_i}{10 \ k\Omega} &= -\frac{V_{o1}}{R_{f1}}, \quad V_{an} = 0 \\ V_{o1} &= -\frac{R_{f1}}{10 \ k\Omega} V_i \end{aligned} \quad (5.6)$$

Para o nó "b", aplicamos novamente a LKC. Assim, considerando como condições para realizar o cálculo

$$\begin{aligned} i_{bn} &= i_{bp} = 0 \\ V_{bn} &= V_{bp} \end{aligned}$$

temos, assim pela LKC em "b"

$$i_2 = i_{bn} + i_{2b} \quad (5.7)$$

$$i_2 = i_{2b} \quad (5.8)$$

sendo

$$i_2 = \frac{V_{o1} - V_{bn}}{10 \ k\Omega} \quad (5.9)$$

$$i_{2b} = \frac{V_{bn} - V_1}{R_{f2}} \quad (5.10)$$

então, voltando a equação 5.8 e aplicando as equações 5.9, nós obtemos

$$\frac{V_{o1} - V_{bn}}{10 \text{ k}\Omega} = \frac{V_{bn} - V_1}{R_{f2}}$$

$$\frac{V_{o1} - V_{offset}}{10 \text{ k}\Omega} = \frac{V_{offset} - V_1}{R_{f2}}$$

rearranjando os termos da equação anterior, é possível chegar em

$$V_1 = V_{offset} \left(1 + \frac{R_{f2}}{10 \text{ k}\Omega}\right) \frac{R_{f1} R_{f2}}{10^2 \Omega} V_i \quad (5.11)$$

A tensão de saída V_{o2} do amplificador de instrumentação é calculada pela diferença entre as tensões V_2 e V_1 dos nós "c" e "d" multiplicada pelo ganho do INA129. Assim de acordo com (10), temos que o ganho é

$$G = \left(1 + \frac{50 \text{ k}\Omega}{R_G}\right) \quad (5.12)$$

e a tensão na saída do é representada pelas equações a seguir

$$V_{o2} = \left(1 + \frac{50 \text{ k}\Omega}{R_G}\right) (V_2 - V_1)$$

$$V_{o2} = \left(1 + \frac{50 \text{ k}\Omega}{R_G}\right) (V_{offset} - V_{offset} \left(1 + \frac{R_{f2}}{10 \text{ k}\Omega}\right) - \frac{R_{f1} R_{f2}}{100 \text{ k}\Omega} V_i) \quad (5.13)$$

5.2.3 Teste do circuito

Após a montagem do circuito em uma *protoboard*, será realizada a sua montagem em uma *PCB* e, seus componentes serão soldados na placa por meio da utilização de estanho e ferro de solda e, somente então será realizada a etapa de testes. A alimentação do circuito se dará por meio de uma fonte geradora de ondas ajustada em 5 V. Assim, esta fonte será o componente V_i do circuito que tem como propósito energizá-lo. A entrada analógica da placa *BeagleBone* será alimentada com a tensão V_{o3} que deverá estar situada entre 0 e 1.8 V. Então, com o auxílio do osciloscópio será verificada a passagem de tensão e corrente em nó por nó do circuito e, além disso, será realizada a visualização da forma de onda gerada pela fonte de corrente construída.

5.2.4 Implementação do Algoritmo de Demodulação

Após a etapa de testes do circuito, será testado um algoritmo de demodulação para que seja possível obter o valor da amplitude da tensão elétrica com a finalidade de controlar a corrente de saída da fonte de corrente. O algoritmo de demodulação será feito na linguagem *Python* em um computador e, após sua conclusão, ele será enviado para a *BeagleBone Black* para que seja possível verificar os resultados da corrente elétrica gerada

pela fonte. Esta linguagem foi escolhida pois de acordo com (3), há um modo de execução chamado *PRU* (*Programmable Real-time Unit*), que aproveita o fato de que a *BeagleBone* possui dois processadores *PRU* que, em combinação com este modo de funcionamento faz com que o desempenho na aquisição de dados seja melhor do que em outros tipos de microprocessadores.

Para o teste de demodulação serão utilizados sinais senoidais AC nas frequências de 1, 10, 20, 30 e 50 kHz, com amplitude de 0.5 V e V_{offset} de 0.6 V. O gerador de ondas será ligado à entrada inversora do AmpOp LM741 no nó "a" por meio de um resistor de 10 k Ω . E, a *BeagleBone Black* estará trabalhando em uma frequência de 110 kHz conforme realizado em (3). Então, os resultados serão analisados.

5.2.5 Teste da fonte de corrente

Uma outra etapa de testes é, de fato, a etapa de testes da fonte de corrente por meio da utilização de cargas Z_L distintas. Inicialmente, serão colocadas cargas de 1, 2, 5, 10, 100, 1000, 10000 e 200 000 Ω no circuito, caso a fonte ainda mantenha o valor ideal de corrente a carga Z_L será aumentada até a fonte falhar. Então, para cada carga, será realizada a etapa de coleta de dados da fonte de corrente por, aproximadamente, 10 minutos. Com os valores serão calculadas as suas médias e desvio-padrões e, estes valores serão utilizados para determinar a carga máxima na qual a fonte de corrente fornecerá 99 % do seu valor de corrente elétrica nominal.

Após a etapa anterior, será realizado um teste de resposta a uma entrada degrau unitário na fonte. Para isto, será aplicado, de forma abrupta, um degrau unitário de tensão em V_i . Então, serão observados os seguintes parâmetros: corrente elétrica, pico de sobressinal, tempo de subida e o erro de estado estacionário. Após a medição destes valores, será feita uma comparação desta fonte de corrente com valores da literatura.

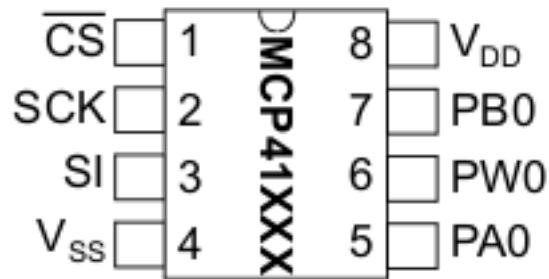
6 RESULTADOS

6.1 Teste do potenciômetro digital

Antes de montar o circuito da Figura 3, foi necessário realizar o teste do *digipot* MCP41010 de 10 k Ω com a finalidade de entender seu funcionamento já que ele apresenta entradas seriais do tipo *Serial Peripheral Interface (SPI)*. A Figura 16 mostra os pinos com as entradas seriais citadas e por meio da Figura 17 é possível visualizar o circuito utilizado para o teste do digipot. É importante ressaltar que este teste foi baseado no circuito da Figura 4-4 apresentado no *datasheet* do potenciômetro digital (11).

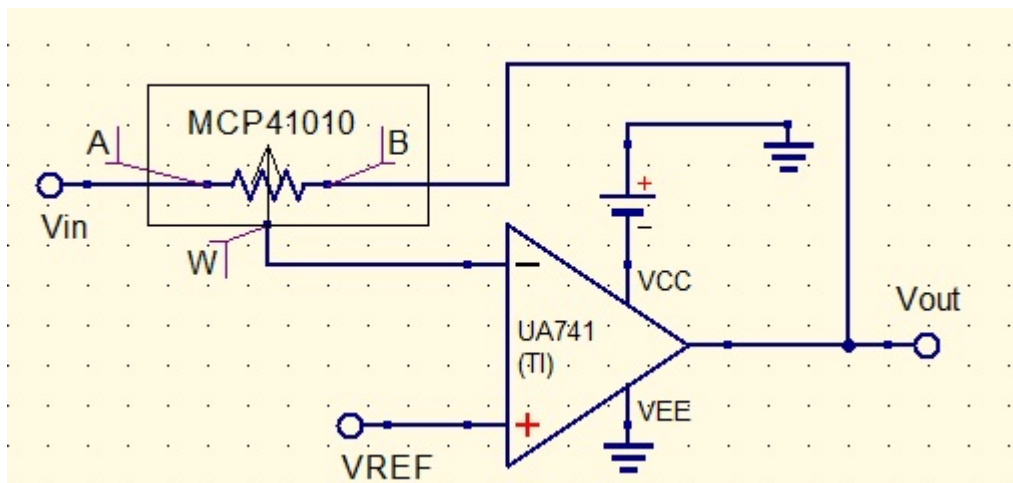
Figura 16 – Representação do potenciômetro digital de 8 bits com seus pinos.

PDIP/SOIC



Fonte: *Datasheet* MCP41010 (11).

Figura 17 – Circuito utilizado para o teste do potenciômetro digital.



6.1.1 Serial Peripheral Interface (SPI)

De acordo com (12) uma *SPI* é um barramento de dados seriais síncronos que habilita dispositivos como a *BeagleBone Black* comunicarem-se com outros em distancias curtas.

A *Serial Peripheral Interface* é *full-duplex*, isto significa que ele pode transmitir e receber dados ao mesmo tempo, por meio da utilização de linhas separadas para o envio e recebimento de dados.

A comunicação serial acontece entre um dispositivo mestre e outro escravo. Primeiramente, o circuito mestre define a frequência de *clock* na qual ocorre a sincronização dos canais de comunicação de dados. Assim, este bloco leva o *Chip Select (CS)* para o estado *LOW*, que faz com que o dispositivo seja habilitado. Esta linha também é conhecida como seleção de escravo, em inglês, *Slave Select (SS)*. Então, o *SPI* mestre envia ciclos de *clock* e dados para fora da linha *Master Out-Slave In (MOSI)* e recebe dados da linha *Master In-Slave Out (MISO)*. O dispositivo escravo lê os dados da linha *MOSI* e os transmite por meio da linha *MISO*. Nesta transmissão um bit é enviado e um bit é recebido em cada ciclo de *clock*. No final, o bloco mestre para de enviar o sinal de *clock* e então trás a linha *CS* para o estado *HIGH*, desativando o bloco escravo, (12). Como informação adicional, a Tabela 2 trás os modos de operação do barramento *SPI*.

Tabela 2 – Modos de operação do barramento *SPI*

Modo	Polaridade do <i>clock</i> (<i>CPOL</i>)	Fase do <i>clock</i> (<i>CPHA</i>)
0	0 (baixo em repouso)	0 (dados obtidos na borda de subida do sinal de <i>clock</i>)
1	0 (baixo em repouso)	1 (dados obtidos na borda de descida do sinal de <i>clock</i>)
2	1 (alto em repouso)	0 (dados obtidos na borda de descida do sinal de <i>clock</i>)
3	1 (alto em repouso)	1 (dados obtidos na borda de subida do sinal de <i>clock</i>)

6.1.2 Habilitação da *Serial Peripheral Interface*

Para que seja possível testar o potenciômetro digital, antes, é necessário que o barramento *SPI* seja testado. Para isso, é preciso habilitar a interface serial da *BeagleBone Black*. Uma etapa deste processo foi o carregamento do *Device Tree Overlay (DTO)* que tem como função descrever como o hardware deveria ser configurado, para isto, foram digitados os seguintes comandos:

```
~ $ export SLOTS = /sys/devices/bone_capemgr.9/slots
```

```
~ $ export PINS = /sys/kernel/debug/pinctrl/44e10800.pinmux/pins
```

Estes comandos declararam duas *environment variables*, chamadas *SLOTS* e *PINS*.

Em seguida, estas duas linhas foram adicionadas para o arquivo *.profile*, no diretório *home* da *BeagleBone Black*. Para isto, foi utilizado o editor de textos chamado *nano*. O comando de chamada foi: *nano ~/.profile* e, então, as duas linhas foram adicionadas no final do documento conforme a Figura 18.

Figura 18 – Declaração das variáveis *SLOTS* e *PINS*.

```

GNU nano 2.2.6 File: /root/.profile

# ~/.profile: executed by Bourne-compatible login shells.

if [ "$BASH" ]; then
  if [ -f ~/.bashrc ]; then
    . ~/.bashrc
  fi
fi

mesg n
export SLOTS=/sys/devices/bone_capemgr.9/slots
export PINS=/sys/kernel/debug/pinctrl/44e10800.pinctrl/pins

[ Read 12 lines ]
G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text    ^C Cur Pos
X Exit      ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
  
```

Fonte: Terminal do Ubuntu.

Para que fosse possível transferir estas *environment variables* para o *root user*, foi utilizado o comando "visudo" e, abaixo da linha *env_reset*, foi utilizado o comando *sudo* para reter as variáveis *SLOTS* e *PINS* em uma chamada *su*, Figura 19:

Figura 19 – Inclusão das *environment variables* para o *root user*.

```

GNU nano 2.2.6 File: /etc/sudoers.tmp

This file MUST be edited with the 'visudo' command as root.

Please consider adding local content in /etc/sudoers.d/ instead of
directly modifying this file.

See the man page for details on how to write a sudoers file.

defaults    env_reset
defaults    env_keep += "SLOTS"
defaults    env_keep += "PINS"
defaults    mail_badpass
defaults    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:$

Host alias specification

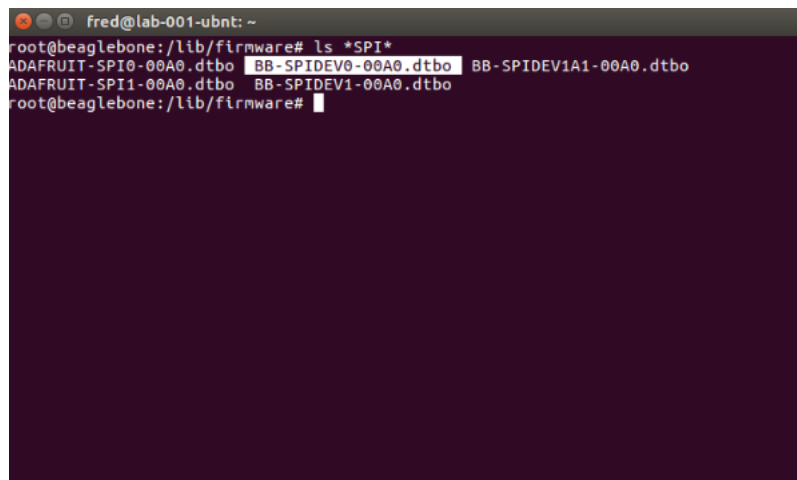
User alias specification

Cmnd alias specification

[ Read 31 lines ]
G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text    ^C Cur Pos
X Exit      ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
  
```

Fonte: Terminal do Ubuntu

Feito isto, utilizou-se o comando *sudo sh -c "echoBB - SPIDEV0 > \$SLOTS"* para carregar o *overlay*, Figuras 20 e 21.

Figura 20 – Habilitação do primeiro barramento *SPI* (SPI0).

```
fred@lab-001-ubnt: ~  
root@beaglebone:/lib/firmware# ls *SPI*  
ADAFRUIT-SPI0-00A0.dtbo BB-SPIDEV0-00A0.dtbo BB-SPIDEV1A1-00A0.dtbo  
ADAFRUIT-SPI1-00A0.dtbo BB-SPIDEV1-00A0.dtbo  
root@beaglebone:/lib/firmware#
```

Fonte: Terminal do Ubuntu

Figura 21 – Continuação da habilitação do primeiro barramento *SPI* (SPI0).

```
fred@lab-001-ubnt: ~  
root@beaglebone:/lib/firmware# ls *SPI*  
ADAFRUIT-SPI0-00A0.dtbo BB-SPIDEV0-00A0.dtbo BB-SPIDEV1A1-00A0.dtbo  
ADAFRUIT-SPI1-00A0.dtbo BB-SPIDEV1-00A0.dtbo  
root@beaglebone:/lib/firmware# sudo sh -c "echo BB-SPIDEV0 > $SLOTS"  
root@beaglebone:/lib/firmware# cat $SLOTS  
0: 54:PF---  
1: 55:PF---  
2: 56:PF---  
3: 57:PF---  
4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,BB-BONE-EMMC-2G  
5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,BB-BONELT-HDMI  
10: ff:P-O-L Override Board Name,00A0,Override Manuf,BB-SPIDEV0  
root@beaglebone:/lib/firmware#
```

Fonte: Terminal do Ubuntu

A certeza de que o carregamento do *overlay* foi bem sucedida foi obtida por meio da visualização de dois novos dispositivos criados no diretório */dev*, o *spidev1.0* e *spidev1.1*, onde, no primeiro o número 0 significa *Chip Select* 0 no barramento 1 e, no segundo, *Chip Select* 1 no barramento 1, Figura 22.

Figura 22 – Novos dispositivos em /dev.

```

fred@lab-001-ubnt: ~
root@beaglebone:/lib/firmware# ls *SPI*
ADAFRUIT-SPI0-00A0.dtbo  BB-SPIDEV0-00A0.dtbo  BB-SPIDEV1A1-00A0.dtbo
ADAFRUIT-SPI1-00A0.dtbo  BB-SPIDEV1-00A0.dtbo
root@beaglebone:/lib/firmware# sudo sh -c "echo BB-SPIDEV0 > $SLOTS"
root@beaglebone:/lib/firmware# cat $SLOTS
0: 54:PF---
1: 55:PF---
2: 56:PF---
3: 57:PF---
4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,BB-BONE-EMMC-2G
5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,BB-BONELT-HDMI
10: ff:P-O-L Override Board Name,00A0,Override Manuf,BB-SPIDEV0
root@beaglebone:/lib/firmware# cd /dev
root@beaglebone:/dev# ls sp*
spidev1.0  spidev1.1
root@beaglebone:/dev#

```

Fonte: Terminal do Ubuntu

6.1.3 Teste da *Serial Peripheral Interface*

Para testar a *SPI*, foi necessário utilizar o código antigo chamado *spidev_test.c*. O código pode ser visualizado no anexo deste documento, (13). A Figura 23 nos mostra o resultado do teste pela aplicação do algoritmo quando não há nenhum *jumper* conectado nas entradas seriais.

Figura 23 – Resultado do teste da *Serial Peripheral Interface* quando não havia nenhum pino conectado.

```

fred@lab-001-ubnt: ~
1: 55:PF---
2: 56:PF---
3: 57:PF---
4: ff:P-O-L Bone-LT-eMMC-2G,00A0,Texas Instrument,BB-BONE-EMMC-2G
5: ff:P-O-L Bone-Black-HDMI,00A0,Texas Instrument,BB-BONELT-HDMI
7: ff:P-O-L Override Board Name,00A0,Override Manuf,BB-SPIDEV0
root@beaglebone:/lib/firmware# cd /dev
root@beaglebone:/dev# ls sp*
spidev1.0  spidev1.1
root@beaglebone:/dev# nano
root@beaglebone:/dev# gcc spidev_test.c -o spidev_test
root@beaglebone:/dev# ./spidev_test
spi mode: 0
bits per word: 8
max speed: 500000 Hz (500 KHz)

FF FF FF FF FF FF
FF FF FF FF FF FF
FF FF FF FF FF FF
FF FF FF FF FF FF
FF FF FF FF FF FF
FF FF
root@beaglebone:/dev#

```

Fonte: Terminal do Ubuntu

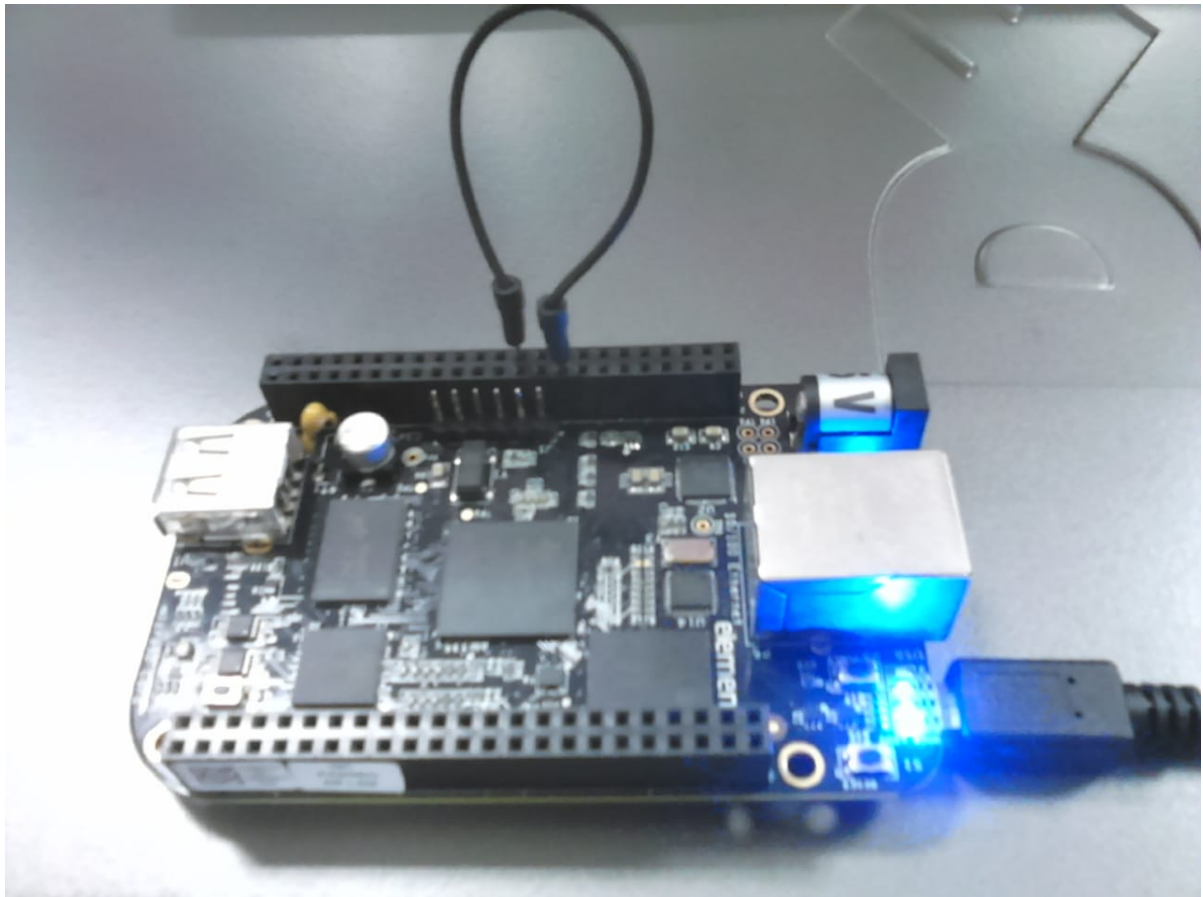
A saída do código testado está no formato hexadecimal, e é igual a 0xFF que, significa que nada está conectado ao barramento, i.e., não há fios conectando os pinos seriais. Após conectarmos um fio entre os pinos P_{918} ($spi0_d1$) e P_{921} ($spi0_d0$) na configuração *MOSI* ao *MISO*, Figuras 24 e 25, foi obtido o resultado a seguir:

Figura 24 – Janela de teste da *Serial Peripheral Interface* quando os pinos P_{18} e P_{21} foram conectados.

```
fred@lab-001-ubnt: ~  
root@beaglebone:/dev# ./spidev_test  
spi mode: 0  
bits per word: 8  
max speed: 500000 Hz (500 KHz)  
  
FF FF FF FF FF FF  
40 00 00 00 00 95  
FF FF FF FF FF FF  
FF FF FF FF FF FF  
FF FF FF FF FF FF  
DE AD BE EF BA AD  
F0 0D  
root@beaglebone:/dev#
```

Fonte: Terminal do Ubuntu

Figura 25 – *BeagleBone Black* com as entradas $spi0_d1$ e $spi0_d0$ conectadas por um *jumper*.



Esta nova sequência pode ser interpretada como uma transmissão de dados entre P_{918} (*MOSI*) e P_{921} (*MISO*).

Referências

- 1 L. T. Harrison, *Current Sources and Voltage References: A Design Reference for Electronics Engineers*. Elsevier, 2005.
- 2 F. Seoane, R. B. , and K. Lindecrantz, “Current source for multifrequency broadband electrical bioimpedance spectroscopy systems. a novel approach,” in *Engineering in Medicine and Biology Society, 2006. EMBS’06. 28th Annual International Conference of the IEEE*, pp. 5121–5125, IEEE, 2006.
- 3 S. F. Alkmin, “Desenvolvimento de fonte de corrente controlada por microprocessador para aplicação na tomografia por impedância elétrica,” 2016.
- 4 R. J. Tocci, *Digital Systems: Principles and Applications*. Pearson Education India, 1980.
- 5 J. W. L. Nerys, “Notas de aula microprocessadores e microcontroladores.”
- 6 G. Coley, *BeagleBone Black System Reference Manual*, 2013.
- 7 J. Kridner, “Beaglebone.org. beaglebone: open-hardware expandable computer.” <<http://beagleboard.org/Support/bone101>>, 2017. Acesso em: 26/04/2018.
- 8 N. S. Nise, *Engenharia de Sistemas de Controle*. LTC, 6 ed., 2012.
- 9 T. R. Kuphaldt, “Practical guide to radio-frequency analysis and design.” <<https://www.allaboutcircuits.com/textbook/>>, 1996. Acesso em: 15/04/2018.
- 10 T. Instruments, “Ina12x precision, low-power instrumentation amplifiers.” <<http://www.ti.com/lit/ds/symlink/ina129.pdf>>, 2018. Acesso em: 28/04/2018.
- 11 Microchip, “Mcp41xxx/42xxx: Single/dual digital potentiometer with spi interface.” <<http://ww1.microchip.com/downloads/en/DeviceDoc/11195c.pdf>>, 2018. Acesso em: 10/06/2018.
- 12 D. Molloy, *Exploring BeagleBone: tools and techniques for building with embedded Linux*. John Wiley & Sons, 2014.
- 13 A. Vorontsov, “Serial peripheral interface test.” <https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/plain/Documentation/spi/spidev_test.c?id=95b1ed2ac7ffe3205afc6f5a20320fbd984da92>, 2007. Acesso em: 13/08/2018.

Código para testar a Serial Peripheral Interface

```
/*
 * SPI testing utility (using spidev driver)
 *
 * Copyright (c) 2007 MontaVista Software, Inc.
 * Copyright (c) 2007 Anton Vorontsov <avorontsov@ru.mvista.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License.
 *
 * Cross-compile with cross-gcc -I/path/to/cross-kernel/include
 */

#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>

#define ARRAY_SIZE(a) (sizeof(a) / sizeof((a)[0]))

static void pabort(const char *s)
{
    perror(s);
    abort();
}

static const char *device = "/dev/spidev1.1";
static uint8_t mode;
static uint8_t bits = 8;
static uint32_t speed = 500000;
static uint16_t delay;
```

```

static void transfer(int fd)
{
    int ret;
    uint8_t tx[] = {
        0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
        0x40, 0x00, 0x00, 0x00, 0x00, 0x95,
        0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
        0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
        0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
        0xDE, 0xAD, 0xBE, 0xEF, 0xBA, 0xAD,
        0xF0, 0x0D,
    };
    uint8_t rx[ARRAY_SIZE(tx)] = {0, };
    struct spi_ioc_transfer tr = {
        .tx_buf = (unsigned long)tx,
        .rx_buf = (unsigned long)rx,
        .len = ARRAY_SIZE(tx),
        .delay_usecs = delay,
        .speed_hz = speed,
        .bits_per_word = bits,
    };

    ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
    if (ret == 1)
        perror("can't send spi message");

    for (ret = 0; ret < ARRAY_SIZE(tx); ret++) {
        if (!(ret % 6))
            puts("");
        printf("%.2X ", rx[ret]);
    }
    puts("");
}

void print_usage(const char *prog)
{
    printf("Usage: %s [-Dsbd1HOLC3]\n", prog);
    puts("  -D --device    device to use (default /dev/spidev1.1)\n"
        "  -s --speed    max speed (Hz)\n");
}

```

```

" -d --delay      delay (usec)\n"
" -b --bpw        bits per word \n"
" -l --loop       loopback\n"
" -H --cpha       clock phase\n"
" -O --cpol       clock polarity\n"
" -L --lsb        least significant bit first\n"
" -C --cs-high    chip select active high\n"
" -3 --3wire      SI/SO signals shared\n");
exit(1);
}

void parse_opts(int argc, char *argv[])
{
while (1) {
static const struct option lopts[] = {
{ "device", 1, 0, 'D' },
{ "speed", 1, 0, 's' },
{ "delay", 1, 0, 'd' },
{ "bpw", 1, 0, 'b' },
{ "loop", 0, 0, 'l' },
{ "cpha", 0, 0, 'H' },
{ "cpol", 0, 0, 'O' },
{ "lsb", 0, 0, 'L' },
{ "cs-high", 0, 0, 'C' },
{ "3wire", 0, 0, '3' },
{ "no-cs", 0, 0, 'N' },
{ "ready", 0, 0, 'R' },
{ NULL, 0, 0, 0 },
};
int c;

c = getopt_long(argc, argv, "D:s:d:b:lHOLC3NR", lopts, NULL);

if (c == -1)
break;

switch (c) {
case 'D':
device = optarg;

```

```
break;
case 's':
    speed = atoi(optarg);
    break;
case 'd':
    delay = atoi(optarg);
    break;
case 'b':
    bits = atoi(optarg);
    break;
case 'l':
    mode |= SPI_LOOP;
    break;
case 'H':
    mode |= SPI_CPHA;
    break;
case '0':
    mode |= SPI_CPOL;
    break;
case 'L':
    mode |= SPI_LSB_FIRST;
    break;
case 'C':
    mode |= SPI_CS_HIGH;
    break;
case '3':
    mode |= SPI_3WIRE;
    break;
case 'N':
    mode |= SPI_NO_CS;
    break;
case 'R':
    mode |= SPI_READY;
    break;
default:
    print_usage(argv[0]);
    break;
}
```

```
}

int main(int argc, char *argv[])
{
    int ret = 0;
    int fd;

    parse_opts(argc, argv);

    fd = open(device, O_RDWR);
    if (fd < 0)
        pabort("can't open device");

    /*
     * spi mode
     */
    ret = ioctl(fd, SPI_IOC_WR_MODE, &mode);
    if (ret == -1)
        pabort("can't set spi mode");

    ret = ioctl(fd, SPI_IOC_RD_MODE, &mode);
    if (ret == -1)
        pabort("can't get spi mode");

    /*
     * bits per word
     */
    ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
    if (ret == -1)
        pabort("can't set bits per word");

    ret = ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits);
    if (ret == -1)
        pabort("can't get bits per word");

    /*
     * max speed hz
     */
    ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
```

```
if (ret == -1)
pabort("can't set max speed hz");

ret = ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed);
if (ret == -1)
pabort("can't get max speed hz");

printf("spi mode: %d\n", mode);
printf("bits per word: %d\n", bits);
printf("max speed: %d Hz (%d KHz)\n", speed, speed/1000);

transfer(fd);

close(fd);

return ret;
}
```