**Lista de figuras**

Figura 1: Placas PCI e PCI-Express da National Instruments (NI, 2016).

Figura 2: Exemplo de anúncio de venda casada do Kit DI-149 .

Figura 3: Esquemático de um DAQ conectado a um computador (Próprio Autor).

Figura 4: Diagrama do Satira AM3358A (COLEY, 2013).

Figura 5: O BeagleBone Black e seus componentes (MOLLOY, 2014).

Figura 6: BeagleBone controlando uma CNC e rodando uma aplicação para CNC simultaneamente (PRACTICAL MICRO, 2014).

Figura 7: Gravando a imagem do Debian 7.5 de 14/05/2014 no cartão SD (Próprio Autor).

Figura 8: Verificando a atualização do Debian 7.5 foi instalada com sucesso (Próprio Autor).

Figura 9: Comparação entre o modelo OSI e a pilha de protocolos da internet (Próprio autor).

Figura 10: BeagleBone Black conectado a um roteador (Próprio autor).

Figura 11: Configurações para conectar o BBB com o computador através do PuTTy (Próprio Autor).

Figura 12: Comunicação serial.

**Lista de siglas**

*ADC* Analog-digital converter, 6

ARM Advanced RISC Machine, 8

BBB BeagleBone Black, 8

*CPU* Central Processing Unit, 5

CTAI Centro de Capacitação Tecnológica em Automação Industrial, 10

DAC Digital-Analog Converters, 6

DAQ Data acquisition system, 3

DC Direct Current, 12

DSI Display Serial Interface, 9

eMMC embedded MultiMediaCard, 12

FAT File Allocation Table, 16

FTP File Transfer Protocol, 20

GPU graphics processing unit, 8

GUI: Graphical User Interface, 13

HDMI High-Definition Multimedia Interface, 9

HTTP Hypertext Transfer Protocol, 20

I/O input & output, 9

I²C Inter-Integrated Circuit, 9

IP Internet Protocol, 19

IRC Internet Relay Chat, 20

ISO International Organization for Standardization, 19

JTAG Joint Test Action Group, 12

LDO Low Dropout Regulator, 12

*NI* National Instruments, 5

OSI Open Systems Interconnectons, 19

PCI Peripheral Component Interconnect, 3

PCI-e PCI-express, 3

PDA Personal digital assistant, 12

PRU Programmable Real-Time Unit, 8

PWM pulse width modulation, 9

RAM random access memory, 9

rev. revision, 11

SD Secure Digital Card, 9

SO Sistema Operacional, 19

SOC System on Chip, 6

SPI Serial Peripheral Interface, 9

SSH Secury Shell, 19

UART Universal asynchronous receiver/transmitter, 9

*USB* Universal Serial Bus, 4

voIP Voice over IP, 20

# Introdução

Um sistema de aquisição de dados, ou DAQ (Data acquisition system), é um dispositivo, ou um conjunto deles, capaz de coletar, armazenar e distribuir uma determinada informação de tal forma que esta, posteriormente possa ser manipulada ou utilizada para entender melhor um certo fenômeno. Na prática esses sistemas são utilizados para capturar dados de uma determinada variável física de um processo, geralmente vinda de um sensor.

## Um breve histórico

Inicialmente os sistemas de aquisição de dados eram dispositivos eletromecânicos que mostravam uma determinada grandeza física em um visor analógico e registravam as mesmas em papéis ou fitas magnéticas. Estes primeiros aparelhos que registravam os dados de maneira independente ficaram conhecidos como *data loggers*. Atualmente o uso de data loggers ainda é bastante comum, por serem mais simples e robustos, entretanto, com o avanço da eletrônica e com o advento dos computadores surgiram novas classes de equipamentos que capturam dados, dentre eles, os que podem ser conectados a PCs e trabalham em conjunto com *softwares* de engenharia e matemática, como MATLAB e LabVIEW, também conhecidos como PC-based data acquisition equipment (MEASUREMENT COMPUTING CORPORATION, 2012).

Os primeiros *PC-based DAQs* surgiram na década de 60, inicialmente utilizado pelas grandes indústrias e centros de pesquisa. Entretanto, a partir da década de 90, com a popularização dos computadores pessoais, surgiram os primeiros DAQs na forma de placas de extensão para computadores utilizando *slots* de extensão PCI[[1]](#footnote-1) (Peripheral Component Interconnect). Hoje em dia os *slots* PCI evoluíram para o padrão PCI-e (PCI-express), um barramento com elevadíssima taxa de transferência de dados, podendo chegar a 32Gbps para o padrão PCI-e 4.0 (PEREIRA, 2012). As placas PCI ainda existem, mas estão entrando em desuso, pois todas as placas-mãe atuais vem somente com o padrão mais moderno. Na figuraFigura 1 tem-se dois modelos de DAQs na forma de placas de extensão para computadores com o padrão PCI (Esquerda) e o padrão PCI-e (Direita).

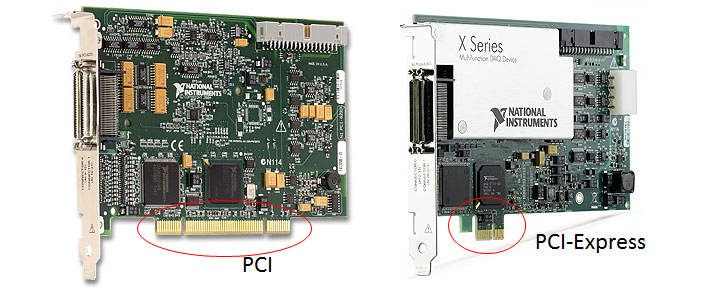


Figura 1: Placas PCI e PCI-Express da National Instruments (NI, 2016).

Os modelos de DAQs na forma de extensão de placas para PCs foram bastantes comuns na década de 90, porém estas classes de dispositivos podem sofrer interferência eletromagnética e eletroestática devido as máquinas rotativas dos computadores, como coolers, e a própria estrutura de barramentos do computador, necessitando de um sistema de proteção contra estes tipos de interferência. Atualmente os fabricantes vem disponibilizando sistemas de aquisição de dados em uma caixa separada do computador, garantindo um isolamento melhor em relação as interferências citadas. O fato de estar externo às máquinas de análise, permite, também, uma maior portabilidade e menos restrição de espaço, podendo ser utilizado em conjunto com notebooks, não se limitando apenas a uma máquina, e ainda, em alguns casos, permite a comunicação entre DAQs a metros ou quilômetros de distância do seu PC de análise, por exemplo, a distância entre o chão de fábrica e a sala de controle.

Estas caixas separadas do computador só foram possíveis com a evolução dos protocolos de comunicação, principalmente àqueles relativos ao PC e seus periféricos externos. Os primeiros dispositivos isolados dos computadores, ou stand-alone, utilizavam principalmente a comunicação serial RS-232 e suas variantes, como o RS-485; este último mais no meio industrial.

Com o tempo surgiram protocolos de comunicação com alta largura de banda, como a USB (Universal Serial Bus), a partir da versão 2.0, a Ethernet e redes sem fio de alta velocidade como Wi-Fi, permitindo o surgimento de data acquisitions stand-alone de alta velocidade e vários canais. Nas industrias há, ainda, muitos dispositivos que utilizam os padrões de conexão mais antigo, entretanto nos escritórios, onde a troca de equipamentos é bem mais frequente, aparelhos com protocolos antigos estão cada vez mais escassos. Portanto, para DAQs que funcionam em escritórios e centros de pesquisa utilizar padrões de comunicação modernos é uma necessidade devido, principalmente, a compatibilidade, pois não está cada vez mais raro encontrar computadores com porta serial, por exemplo.

## Situação atual dos DAQs

Historicamente os sistemas de aquisição de dados foram criados para atender a necessidade de grandes indústrias e centros de pesquisa. Estes tipos de equipamentos, por sua vez, nunca foram baratos, demanda de equipamentos de alta qualidade, com condições de operação mais severas e, principalmente, devido ao grande poder de compra dos clientes. Por isso, os maiores fabricantes destes dispositivos entregam soluções com preços pouco convidativos e, quase sempre, com software e hardware proprietários.

Para exemplificar, a NI (National Instruments), grande fabricante do ramo, vende DAQs que variam de à , segundo o site oficial da loja no Brasil (NI, 2016). Com o uso de software e hardware proprietário é comum muitos fabricantes praticarem venda casada, onde, para ter uma experiência completa o cliente deve comprar diversos outros produtos além do de interesse. É possível observar esta prática no anuncio da DAQ Instruments, onde o interessado deve comprar o pacote do DAQ e as licenças de softwares. Detalhe, sozinho o aparelho custa e o pacote , ou seja, mesmo com o desconto de do pacote, custa mais que o triplo do equipamento sozinho.

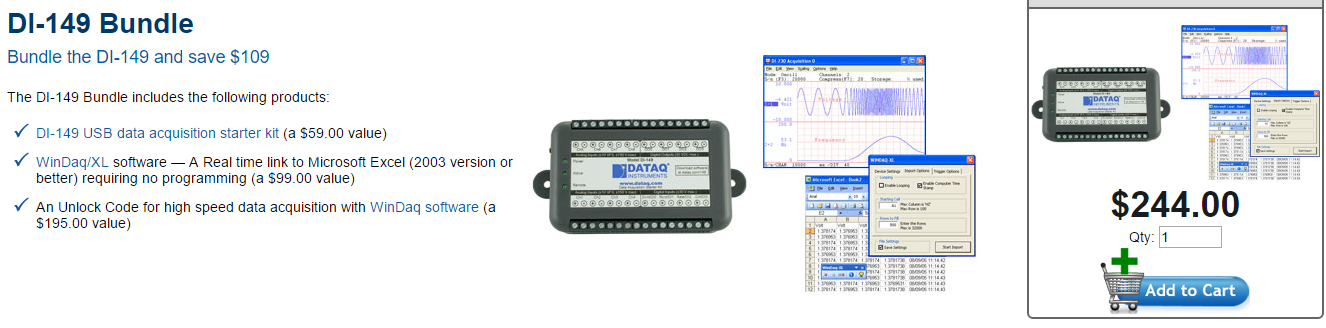


Figura 2: Exemplo de anúncio de venda casada do Kit DI-149 .

## Novas soluções

Na topologia padrão de um DAQ conectado a um computador é necessário converter o sinal analógico, geralmente vindo de um sensor, para dados digitais, de tal forma que possa ser reconhecido pelo computador. Esses dados posteriormente serão enviados através de uma interface de comunicação entre o computador e o DAQ. Para que tudo isso seja possível, existe uma CPU (Central Processing Unit) capaz de gerenciar os dados de entrada do ADC (Analog-digital converter) e a interface de comunicação. Um esquemático desta topologia pode ser visto na figuraFigura 3.

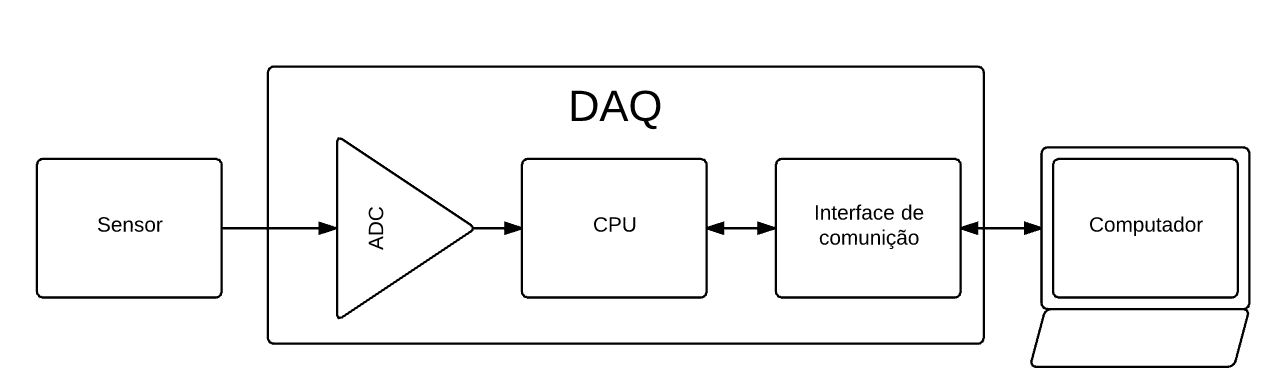


Figura 3: Esquemático de um DAQ conectado a um computador (Próprio Autor).

Na verdade, a topologia do DAQ representado na figuraFigura 3 é algo bastante comum na eletrônica embarcada, por isso, já vem embutida em alguns sistemas embarcados. Sistemas embarcados são dispositivos feitos para um propósito específico, desde o controle de um elevador ao gerenciamento de um instrumento de aquisição de dados. Nestes sistemas é comum o uso microcontroladores, que são computadores dedicados àquela tarefa exclusiva. Muito microcontroladores, além da CPU e memória, vem diversos periféricos como sistemas de comunicação com o meio externo, ADCs, DACs (Digital-Analog Converters) embutidos em um único encapsulamento. Essa integração de diversos periféricos em um único chip deixa o projeto mais simples e barato. Por isso, é cada vez mais comum os fabricantes disponibilizarem sistemas completos em um único encapsulamento. Este conceito é conhecido como System on Chip, os famosos SOCs, que estão embarcados nos mais diversos dispositivos, até mesmo em celulares.

Os primeiros microcontroladores surgiram no final do século passado. Inicialmente eles não eram muito acessíveis, tanto na questão de preço quanto na complexidade de programação e desenvolvimento. Muitos deles tinham que ser programados em assembly*[[2]](#footnote-2)* por causa da baixa memória disponível. Entretanto, com o tempo, os novos modelos vieram com mais memória, mais poder de processamento e periféricos mais avançados. Junto a isso, surgiram novas ferramentas e compiladores, facilitando bastante o desenvolvimento nestes dispositivos para uso específico.

Uma grande revolução neste sentido ocorreu em meados de 2005, quando um grupo de pesquisadores da Itália propuseram um sistema embarcado de baixo custo, fácil utilização e código aberto, conhecida como Arduino. O Arduino é uma plataforma de prototipagem completa incluindo o hardware e software para isto. Do lado do hardware, além do microcontrolador, ela vem com todos os circuitos externos para o seu funcionamento, incluindo fonte de alimentação, regulador de tensão oscilador externo, entrada USB e pinos de extensão (BANZ, 2012). Do lado do software o Arduino dispõe de uma linguagem de programação que abstrai os elementos de hardware do microcontrolador, portanto é possível programar na plataforma sem precisar alterar nenhum valor de registradores diretamente no código, pois operações de baixo nível como estas, são feitas em background pelo firmware pré-instalado na memória do microcontrolador.

O sucesso da placa de desenvolvimento veio principalmente do fato de ser código aberto e da facilidade de utilização, que atraiu diversos entusiastas, “hobbistas” e profissionais da área, se tornando uma das maiores comunidades, se não a maior, de sistemas embarcados no momento em que esta monografia está sendo escrita. De uma forma geral, o Arduino, além de aumentar o número de pessoas interessadas em eletrônica, aumentou a quantidade de projetos e a velocidade com que eles eram feitos, possibilitando a criação de diversas invenções como o tênis que amarra sozinho (BLAKEBEVIN, 2010) e impressoras 3D.

Hoje existem diversos fabricantes que fazem módulos de expansão para o Arduino conhecidos como shields. Existem shields para tudo: controle de motores relés, conexão Ethernet, Wi-fi, Bluetooth, dentre outros. Com os módulos e o Arduino em mãos o usuário só precisa de conectores, criatividade e programação para criar ideais fantásticas como as citadas no parágrafo anterior, sem a necessidade de fazer as placas de circuito impresso, que dão tanto trabalho, na fase de desenvolvimento.

Depois que o Arduino se tornou popular diversas organizações e empresas resolveram criar suas placas de desenvolvimento open source. Na mesma época em que o Arduino foi criado surgiram os primeiros smartphones baseado em telas touch-screen. Junto a eles, surgiram também, os SOCs com processadores ARM[[3]](#footnote-3) (Advanced RISC Machine) e GPUs[[4]](#footnote-4) (graphics processing unit) de baixo consumo que permitiu aliar o baixo consumo, baixa dissipação de calor, tamanho portátil e alto desempenho em um único chip. Em 2008 a Texas Instruments, uma empresa privada que desenvolve diversos produtos semicondutores como microcontroladores e SOCs com arquitetura ARM para celulares, resolveu criar a BeagleBoard em parceria com a Digikey e a Element14, duas empresas do ramo de varejo eletrônico nos EUA. A ideia era demonstrar o poder do SOC OMAP3530 em uma placa de desenvolvimento do tamanho de um cartão de crédito, capaz de rodar uma distribuição Linux portada para ARM, incluindo os aplicativos preparados para este sistema operacional. A empreitada era uma das pioneiras do ramo e poderia ser promissora se não fosse pelo preço de que não a fez decolar.

Os dispositivos baseados em e-Linux[[5]](#footnote-5) (Embedded Linux) só se tornaram popular quando a fundação Raspberry Pi resolveu criar um produto de mesmo nome por apenas 35 dólares, em fevereiro de 2012. A ideia da fundação era criar uma plataforma de baixíssimo custo que ajudassem as crianças a aprender a programar (MACHADO, 2012). Hoje o Raspberry Pi é de longe a e-Linux board mais vendida, e também, com a maior comunidade.

Junto ao sucesso do Raspberry, diversas empresas e fundações criaram suas plataformas open source com Linux embarcado, ou atualizaram suas antigas soluções, como foi o caso da empresa criadora do BeagleBoard. Assim aproveitando a onda dos computadores single board, a Texas Intruments criou o BeagleBone Black O grande diferencial deste computador em relação ao Raspberry Pi não é o poder de processamento, o BBB (BeagleBone Black) tem 65 pinos de extensão, um microcontrolador auxiliar para programação em tempo real, o PRU (Programmable Real-Time Unit), e muitos mais periféricos que o Raspberry Pi. Para efeito de comparação, na tabelaTabela 1, tem-se o comparativo entre BeagleBone Black, Arduino Yun, uma das versões do Arduino com Linux embarcado, e o Raspberry Pi.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Arduino Yun** | **BBB** | **Raspberry Pi** |
| CPU | MIPS32 | ARM Cortex-A8 | ARM1176 |
| CPU Freq. | 400Mhz | 1Ghz | 700Mhz |
| Microcontrolador | ATmega32U4 | PRU | Não tem |
| Controlador Freq | 16Mhz | 200Mhz | - |
| RAM | 64Mb | 512Mb | 512Mb |
| GPU | Não tem | PowerVR SGX530 | Broadcom VideoCore IV |
| Memória interna | 16Mb | 4GB | Não tem |
| Memória externa | Micro SD | Micro SD | SD |
| I/O | 20 | 65 | 17 |
| Ethernet | 10/100 Mbit | 10/100 Mbps | 10/100 Mbps |
| ADC | 12x 10bits, 0-5V | 7x 12bits 0-1,8V | Não tem |
| PWM |  |  |  |
| UART |  |  |  |
| SPI |  |  |  |
| I²C |  |  |  |
| USB host |  |  |  |
| USB client |  |  | Não tem |
| Video | Não tem | Micro HDMI | HDMI, RCA, DSI |
| Audio | Não tem | Micro HDMI | HDMI, P2 |
| Preço |  |  |  |

Tabela 1: Comparação entre os e-Linux board open source mais usados no mercado (DICOLA, 2014).

Pela tabela, percebe-se que o BBB é o único concorrente que avalia o alto poder de processamento com a grande quantidade de periféricos e I/O (input & output). A quantidade elevada de I/Os permitiu a fabricante do BeagleBone criar o conceito de placas de expansão para a e-Linux board conhecidas como capes, semelhante aos Shields disponíveis para Arduino.

O BeagleBone Black quando lançado fez um relativo sucesso, mais que o seu antecessor, principalmente por causa da nova política de preço, seu microcontrolador integrado, a quantidade generosa de I/Os e seus periféricos. Entretanto, isso não foi o suficiente para desbancar o Raspeberry Pi. Parte disso é devido a forma de como as e-Linux boards são utilizadas. A maioria das aplicações dos single board computers está na área da computação, como na criação de pequenos servidores, centrais de emulação, media centers, câmeras de vigilância, dentre outros. Para isso, as poucas portas do Raspberry Pi são o suficiente, quando há a necessidade de alguma aplicação em tempo real, é comum utilizá-lo em conjunto com algum dispositivo dedicado como um Arduino, por exemplo.

Outro motivo para o BeagleBone Black não ter desbancado o Raspberry, foi o fato do primeiro ter sido lançado tarde. Em 2013, quando foi lançado, a comunidade do Pi já estava grande e já existiam outros concorrentes como a Cubie Board. Além disso, as capes que poderiam ser um grande diferencial são caras e sem muita variedade. Mesmo assim, o BBB é um dos singles board computers mais utilizados, mas longe de ser o mais popular, e ainda, o conceito das capes não engrenou como a fabricante previa.

## Objetivo deste trabalho

O CTAI (Centro de Capacitação Tecnológica em Automação Industrial) é um centro da UFBA com laboratórios destinados ao desenvolvimento de atividades de pesquisa e academia. Neste centro houve a demanda de novos dispositivos de aquisição de dados. O método tradicional seria adquirir estes aparelhos através de uma fabricante tradicional como a National Intruments, mas com o baixo custo da eletrônica embarcada, optou-se por esta opção, pois além de economizar, poderá gerar novas tecnologias relativas aos DAQs, até então, carente no Brasil.

Embora o BeagleBone Black tenha os seus contras, ele ainda é uma das melhores opções para construir um protótipo de um DAQ moderno. Esta plaquinha contempla um ADC de alta qualidade em relação aos outros da categoria, uma CPU de alto desempenho e interfaces de comunicação modernas, acessíveis e com alta largura de banda.

Este trabalho, então, propõe provar que um BeagleBone Black pode ser utilizado como um dispositivo de aquisição de dados acessível aos computadores de escritórios e com taxa de aquisição e número de canais comparáveis a alguns modelos de DAQ dos fabricantes tradicionais. Para este trabalho será utilizada a porta USB disponível no BBB como interface de comunicação com o mundo externo. Em trabalhos futuros os conceitos apresentados neste trabalho podem servir como base para a construção de um dispositivo de aquisição de dados mais completo, capaz de trabalhar com os acelerômetros analógicos do CTAI da mesma forma que os que seriam adquiridos pela NI.

# O BeagleBone Black

Nesta seção será apresentado os fundamentos necessários, relativos ao BeagleBone Black, para a execução dos experimentos na seção X. No meio do percurso serão feitas as configurações necessárias para a execução dos mesmos experimentos.

## Componentes do BeagleBone Black

No capítulo 1 foi introduzido o BeagleBone Black como um minicomputador de arquitetura ARM. Esta seção irá mostrar os componentes da placa e os integrados ao SOC Sitara AM335x. Na figuraFigura 5 tem-se uma foto de frente e verso da placa de circuito impresso do BBB, identificando os componentes, e na tabelaTabela 2 identifica a função de cada componente na placa. Para complementar na figuraFigura 4 mostra os componentes integrados ao SOC Sitara AM3358A, este último disponível na rev. (revision) 3 deste produto.

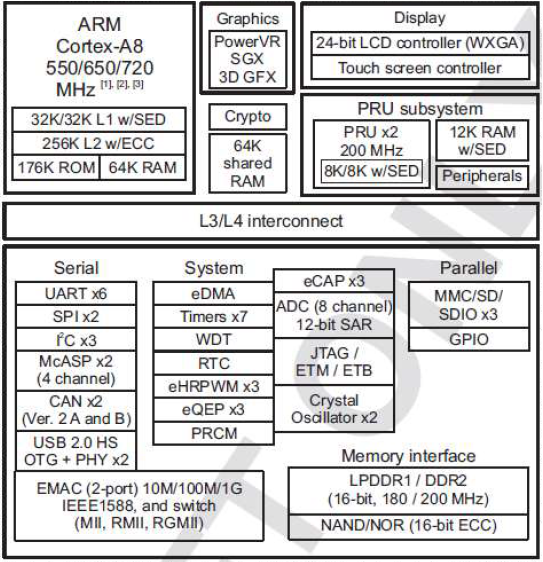


Figura 4: Diagrama do Satira AM3358A (COLEY, 2013).

| **No.** | **Componente** | **BBB** |
| --- | --- | --- |
| 1 | Satara AM335x | SOC do BBB contendo diversos componentes integrados incluindo CPU, GPU e PRU |
| 2 | HDMI framer | Converte o controlador de LCD do AM335x |
| 3 | Memória RAM | 512MB DDR3 |
| 4 | eMMC | 4GB de memória de armazenamento interno |
| 5 | TPS65217C | Regulador de potência sofisticado com 4 reguladores de tensão LDO[[6]](#footnote-6) controlado por I²C |
| 6 | Ethernet PHY | PowerVR SGX530 |
| 7 | LEDs | 4GB |
| 8 | Push buttons | Micro SD |
| 9 | Micro HDMI | 65 |
| 10 | Ethernet RJ45 | 10/100 Mbps |
| 11 | 5V DC | 7x 12bits 0-1,8V |
| 12 | Slot micro SD |  |
| 13 | Serial debug |  |
| 14 | USB 2.0 client |  |
| 15 | USB 2.0 host |  |
| 16&17 | Expansões P8 e P9 |  |
| 18 | JTAG |  |
| 19 | Conector de bateria |  |

Tabela 2: Relação dos componentes do BBB de acordo com a figuraFigura 5 (MOLLOY, 2014).

## As distribuições do BeagleBone Black

Como foi dito nos capítulos anteriores, o BeagleBone Black é um minicomputador completo, capaz de rodar sistemas operacionais portados para a arquitetura ARM. Atualmente a comunidade do BeagleBone desenvolveu o port*[[7]](#footnote-7)* para Android e as distribuições Linux, Debian, Ubuntu, Ångström e Arch Linux.

As primeiras versões do BBB, mais especificamente as revisões 1 e 2, vinham com Ångström instalado por padrão. Esta última uma distribuição criada exclusivamente para sistemas embarcados, tablets, PDAs, set top boxes, roteadores e outros dispositivos com SOCs mobile (KOEN, 2015). Nesta época era bastante comum os usuários remover o Ångström e instalar uma distribuição mais conhecida, como o Ubuntu. O Ångström não tem raízes comuns com as distribuições mais populares no desktop, isto dificultava o aprendizado e o desenvolvimento, principalmente se a pessoa não tem contato muito contato com o Linux, que representa boa parcela dos compradores do BBB.

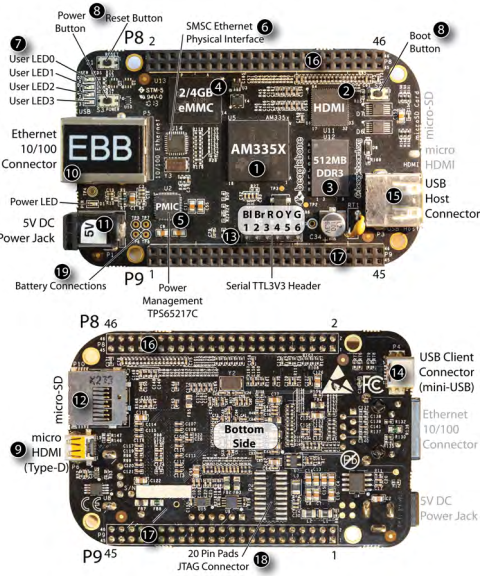


Figura 5: O BeagleBone Black e seus componentes (MOLLOY, 2014).

Percebendo a popularidade de tutoriais ensinando como trocar de distribuição, a Texas passou a incluir o Debian a partir da revisão 3 do BeagleBone Black, lançada em 2014 e até o momento a mais recente. Uma outra adição desta versão foi o aumento da memória eMMC de 2GB para 4GB. Estas duas adições foram fundamentais para tornar o BBB mais atraente para ser usado como o computador. O espaço extra permitiu a instalação de aplicativos adicionais, além dos que já viam com o sistema operacional, principalmente aqueles mais pesados, como Java, Open Office, reprodutores de vídeo e editores de imagem.

Com o Debian, o BeagleBone ficou muito mais amigável para os novatos, principalmente aqueles que tiveram pouco contato com o Linux. A maioria dos comandos seriam iguais àqueles utilizados no computador, por exemplo, para instalar ou atualizar programas pode-se utilizar o comando , no Ångström o comando equivalente é outro.

A nova distribuição facilitou, também, a execução de aplicativos com interface gráfica, ou GUI (Graphical User Interface), baseados nas toolchains mais populares para Linux Desktop, o Qt e GTK+. Toolchains são um conjunto de ferramentas que permitem o desenvolvimento de interface gráficas. Ao desenvolver um aplicativo em Qt ou GTK+, por uma questão memória, nem todas as bibliotecas são incluídas no empacotamento do programa. Assim é comum programas terem dependências relativo a toolchain que o criou. Portanto, com o Debian a criação de aplicativos fica muito parecida com o que já se fazia no desktop, além disso, o port de aplicativos x86 para ARM se torna facilitado. No Ångström é comum outras toolchains, como Bitbake e EmbToolkit (E-LINUX.ORG, 2016).

O uso de aplicativos com interface de desktop é bastante popular em e-Linux boards para servir como interface homem-máquina. Na figura mostra o exemplo de uma máquina CNC (Controle Numérico Computadorizado) controlada por um BeagleBone em conjunto com a cape K9 CNC I/O. Para controla-la a fabricante da cape criou um aplicativo com interface gráfica para Linux, onde o usuário é capaz de visualizar o processo e interagir com a CNC.

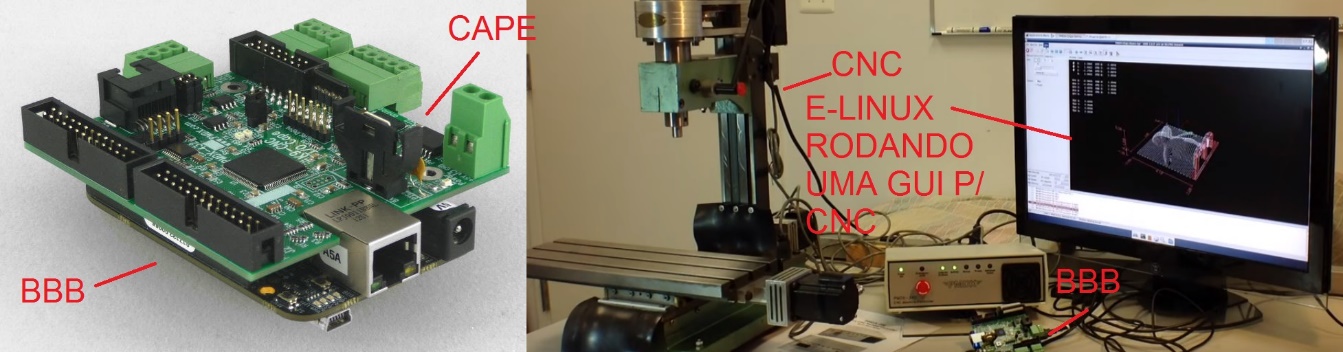


Figura 6: BeagleBone controlando uma CNC e rodando uma aplicação para CNC simultaneamente (PRACTICAL MICRO, 2014).

## Instalando uma nova distribuição no BeagleBone Black

Na seção 2.2 falou-se da importância de utilizar a distribuição Debian. Portanto, se o BeagleBone Black adquirido for anterior a rev. 3, deve-se instalar o Debian 7.5 Wheezy de 14/05/2014. Caso o BBB venha com uma versão superior a esta é recomendado fazer o downgrade para versão 7.5, principalmente, se for a versão 8 Jessie ou superior. A versão 7.5 foi projetada pensando, também, nas revisões anteriores a rev. 3, por isso, ocupa um pouco menos de 2GB, sobrando aproximadamente 2GB dos 4GB da eMMC (Para as rev. 3 ou superiores). Esse espaço extra será suficiente para instalar novos programas, módulos e gerar grandes arquivos de dados, se necessário.

Um outro motivo para fazer o downgrade é que neste trabalho e a maioria das bibliografias encontrada na literatura, faz-se o uso desta versão do Debian, portanto, se for utilizada uma versão diferente, pode haver casos em que determinados procedimentos não sejam os mesmos desta monografia, dificultando a reprodução dos experimentos.

A primeira coisa que deve ser feita para instalar uma nova distribuição é fazer o download do sistema operacional, que pode ser baixado na página que contém as últimas imagens do BeagleBone Black, disponível em (BEAGLEBOARD.ORG, 2016). Caso o leitor desta monografia esteja lendo-a muito tempo depois, existem uma alternativa no domínio oficial do Debian disponível em (DEBIAN, 2016), ou ainda, no site do e-Linux, disponível em (E-LINUX.ORG, 2016).

No site das últimas imagens para BBB existem duas versões do Debian 7.5 14/15/2014. Uma delas é para utilizar o sistema operacional por um cartão SD (without flashing the eMMC), enquanto a outra grava o sistema operacional na eMMC (eMMC flasher). É preferível que o sistema operacional seja instalado na memória interna do BeagleBone, pois, além desta ter uma taxa de leitura e escrita maior do que o cartão SD, fica-se com o slot SD livre para outras ocasiões.

O formato do arquivo baixado é ***.img.xz*** que é um formato de compactação bastante comum no Linux. Caso o usuário esteja utilizando o Windows talvez seja necessário fazer o download de alguma ferramenta capaz de descompactar este tipo de arquivo. Uma sugestão de descompactador capaz de trabalhar com este formato é o 7-Zip, disponível em (7-ZIP.ORG, 2016), gratuito, open source e sem propagandas.

Enquanto a imagem do sistema operacional está sendo baixada, deve-se fazer o download de outro programa para gravar a imagem no cartão de memória. Uma sugestão dada pela Wiki do e-Linux (E-LINUX.ORG, 2016) é o Win32 Disk Imager, disponível em (DAVIS e DAVIS, 2016). Quando ambos os arquivos estiverem baixados, descompacte a imagem do Debian que está no formato ***.img.xz*** e irá obter um arquivo ***.img***. Insira o cartão SD no leitor de cartão do seu computador, abra o Disk Imager e selecione a letra correspondente a partição do cartão de memória, selecione a imagem do Debian e clique em gravar como mostra na figuraFigura 7. Irá aparecer uma caixa de mensagem explicando que esta operação pode danificar os dados do cartão, clique em yes e espere o processo ser finalizado.

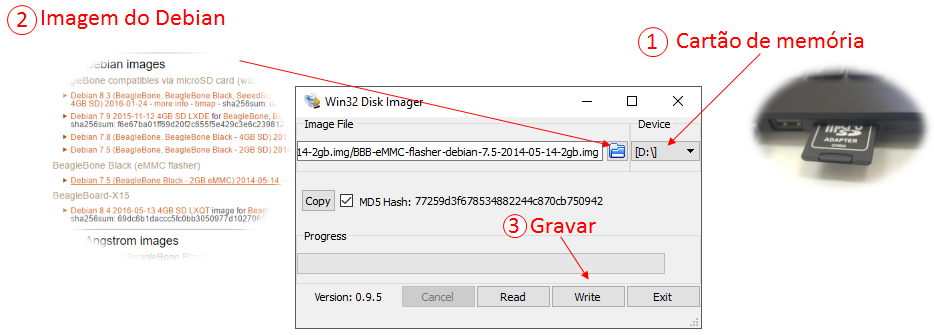


Figura 7: Gravando a imagem do Debian 7.5 de 14/05/2014 no cartão SD (Próprio Autor).

Depois do processo de gravação finalizar com sucesso, retire o cartão SD do leitor e com o BBB ainda desligado, insira o microSD no slot apropriado (Item 12 na figuraFigura 5). Antes de ligar a placa, mantenha pressionado o boot switch (Item 8 na figuraFigura 5), e ainda, com o botão pressionado, conecte à USB client do BeagleBone (Item 14 na figuraFigura 5) e ao computador ou alguma fonte de alimentação. Depois dos users LEDs (Item 7 na figuraFigura 5) começarem a piscar solte o boot switch, a partir daí o sistema operacional será instalado na eMMC. O processo dura cerca de 30 a 40 minutos, durante este tempo não desligue o BeagleBone. Quando a instalação estiver concluída os 4 user LEDs irão se apagar.

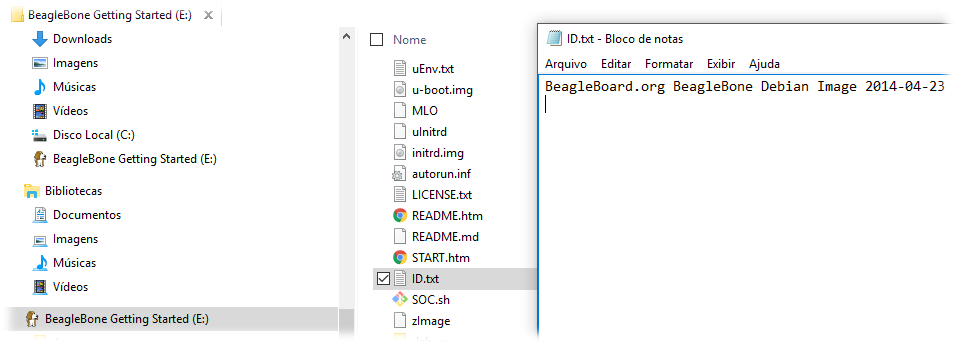


Figura 8: Verificando a atualização do Debian 7.5 foi instalada com sucesso (Próprio Autor).

Neste momento desconecte o cabo USB da fonte de alimentação e, posteriormente, retire o cartão SD, para, então, conectar novamente o BBB o computador. Será montada uma partição FAT com o nome BeagleBone Getting Started. Clique nela e abra o arquivo com o bloco de notas. Se aparecer o texto, “BeagleBoard.org BeagleBone Debian Image 2014-04-23”, a atualização da distribuição foi realizada com sucesso.

## Comunicando com o BeagleBone Black

O BeagleBone Black vanilla, ou seja, da forma como veio de fábrica, não tem nenhuma interface com usuário além de alguns push-buttons e LEDs indicadores (Itens 7 e 8 na figuraFigura 5), e isso não é o suficiente para programar ou adicionar alguma função à placa de desenvolvimento, a menos que usuário use o BeagleBone ligado a um monitor mouse e teclado.

Para interagir com o board computer sem a necessidade desses aparelhos é necessário se comunicar com o BBB através de um computador hospedeiro. Existem diversas formas de fazer esta comunicação, mas esta seção irá focar no protocolo SSH e no serial debug através dos pinos UART0 (Item 13 na figuraFigura 5). Primeiro será falado do protocolo SSH.

### O que é o terminal

Nos primórdios da informática não existia mouse, nem sistema operacional com interface gráfica, janelas e menus. Portanto a interação homem-máquina era feita por comandos que, por sua vez, executavam aplicativos e faziam operações, como entrar em diretórios, copiar arquivos, escrever documentos e até mesmo programar. Este tipo de interação ficou conhecido como linha de comando e era bastante popular em sistemas mais antigos como o DOS. Com o surgimento do Windows, a Microsoft passou a desestimular o uso da linha de comando em detrimento à interface de janelas em conjunto com o mouse. Foi a partir daí que a informática começou a se popularizar, pois a interface de janelas era mais intuitiva e, por isso, atraiu a atenção das massas.

Com o tempo as pessoas se acostumaram com o ambiente gráfico do Windows e a linha de comando ficou em abandono, não apenas pelos seus usuários, mas também pela própria Microsoft, cujo o principal objetivo era deixar o ambiente de janelas cada vez mais rico e moderno. Por motivos de legado, a empresa de Bill Gates disponibilizou o programa ou Prompt de Comando. Ele é um emulador de DOS onde é possível executar boa parte das operações do sistema operacional antigo. Contudo, como não houve atualizações do DOS durante décadas, o Prompt de Comando é ultrapassado, porque as novas tecnologias foram implementadas apenas na interface gráfica, tornando-o bastante pobre em relação ao ambiente de janelas. Entretanto, isto só é verdade no Windows, seus grandes concorrentes, Linux e Mac, não abandonaram a linha de comando.

O Linux e Mac são baseados em um ancestral comum, o Unix. Por isso, sua base de arquivos e a forma de como são organizados são parecidas Uma das principais semelhanças desses sistemas operacionais é o uso da mesma linguagem de linha de comando, o Shell. Esta linguagem, ao contrário do Prompt Comando, é bastante completa, sendo capaz de fazer quase todas as operações do sistema operacional, as vezes até mais operações que a interface gráfica. O programa que executa os comandos Shell é chamado de terminal.

Por muito tempo o Linux só permitia a instalação de programas e drivers através da linha de comando. Uma tarefa um tanto complicada para usuários comuns. Isso, teoricamente, afastou as pessoas comuns deste sistema operacional, chegando ao ponto de muitos atribuírem a baixa adoção do SO (Sistema Operacional) à ausência de uma interface gráfica tão completa quanto a do Windows.

Com o tempo isso mudou e hoje o Linux permite fazer quase tudo, incluindo a instalação de drivers e aplicativos pela interface gráfica. Porém, é possível fazer o mesmo pela linha de comando e isso é uma vantagem enorme do Linux e Mac em relação ao Windows, o freguês pode escolher a maneira de interagir com o sistema.

Existem algumas tarefas que são muito mais rápidas e práticas de serem feitas através do terminal, embora não sejam tão intuitivas quanto, um grande exemplo disso é a automação de tarefas. Com o Shell é possível criar rapidamente pequenos macros ou scripts capazes de automatizar tarefas chatas e repetitivas. Além disso, por utilizar apenas textos para enviar e mostrar informações, esta é uma forma de interação que consome poucos recursos da máquina. E ainda, o uso exclusivo de texto facilita o compartilhamento de artigos e tutoriais relacionados a interações com o sistema, principalmente se eles forem mais complexos como alterar parâmetros de configurações, pois basta o usuário copiar os comandos do tutorial, colar no terminal e apertar Enter. O mesmo tutorial utilizando interface gráfica necessitaria de alguns textos explicativos e capturas de tela, por isso que a maioria dos tutoriais para Linux, disponíveis em livros e na internet, são feitos através da linha de comando, embora seja perfeitamente possível fazer o mesmo pela interface gráfica. O lado ruim dessa prática é que o usuário leigo muitas vezes não tem noção dos processos necessários para a execução de tal tarefa e, por isso, não aprende o porquê ou como reproduz o mesmo sem copiar e colar a sequência de comandos. Por causa disso, muitos usuários recém-chegados no Linux pensam que está é a única forma de fazer determinada tarefa, e acabam abandonando a plataforma por pensar que o sistema operacional não foi feito para “seres humanos”.

### SSH e as pilhas de protocolo IP

A sigla SSH significa Secure Shell, ou seja, uma forma de enviar comandos com criptografia, ou forma segura, de uma máquina para outra. Este protocolo foi criado, inicialmente, para facilitar o acesso remoto de máquinas dentro de servidores (MOLLOY, 2014). Para exemplificar, imagine que um engenheiro necessita instalar um novo aplicativo de um servidor a quilômetros de distância. Com o SSH o engenheiro por de iniciar uma conexão, ou seção, SSH entre o seu computador e uma das máquinas do servidor para, posteriormente, escrever os comandos de instalação do aplicativo, tudo através do terminal da sua máquina.

O fato do SSH ser baseado em somente texto torna a comunicação leve, no sentido de transmitir e receber poucos dados, e fácil de ser implementada. Além disso, traz consigo todas as vantagens que o Shell oferece em relação ao ambiente de janelas. Tudo isso é evidente quando se compara o esforço necessário para fazer uma comunicação remota através de interface gráfica. Para que isso seja possível, é necessário, no mínimo, fazer a captura de tela, mouse e teclado do computador remoto, enquanto com SSH, só precisa os dados digitados no terminal sejam enviados no formato de texto criptografado.

O protocolo SSH é orientado a IP (Internet Protocol), ou seja, faz parte do mesmo padrão de comunicação utilizado pela internet, também conhecido como pilha de protocolos TCP/IP. Este modelo surgiu como um projeto de comunicação do exército americano. Com o tempo o projeto se expandiu e tornou-se a internet como conhecemos hoje.

Atualmente o TCP/IP é baseado num modelo simplificado do padrão de referência da ISO (International Organization for Standardization) para sistemas de comunicação, o OSI (Open Systems Interconnections). O modelo da ISO é baseado em 7 camadas hierárquicas. Cada camada é responsável por executar determinada função e é especificada através de protocolos de comunicação Estes, por sua vez, podem ser ou não compatíveis com os protocolos de outras camadas. Quando tomado em conjunto os protocolos de diferentes camadas, denominamos pilhas de protocolos. A pilha de protocolos da internet é chamada de TCP/IP. Ela é composta obrigatoriamente de 5 das 7 camadas do padrão OSI (Figura 9). As outras duas camadas, dependendo da aplicação, podem existir, mas são opcionais.

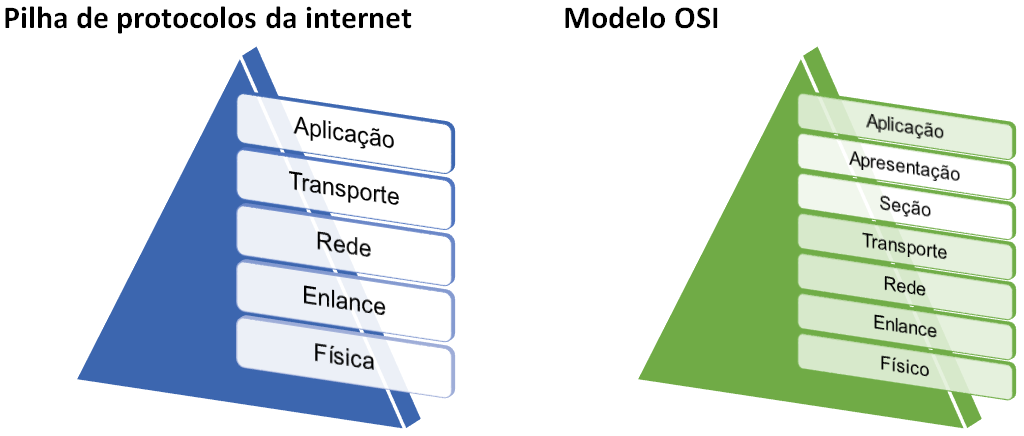


Figura 9: Comparação entre o modelo OSI e a pilha de protocolos da internet (Próprio autor).

A camada com maior grau de abstração do TCP/IP é a aplicação. Nela contém os protocolos responsáveis por oferecer os serviços às máquinas ou ao usuário final. O SSH é um exemplo de protocolo desta camada, ele oferece o serviço de acesso remoto aos computadores. Outros exemplos de serviços dessas camadas são: enviar arquivos (Protocolo FTP, BitTorrent), navegar na web (Protocolo HTTP) e comunicar-se através de chat (Protocolo IRC). Estes protocolos geralmente são implementados via software, junto com a camada de transporte.

A camada de transporte é responsável por transportar as mensagens da camada de aplicação. Existem dois protocolos possíveis para esta camada, o TCP (Transmission Control Protocol), que veio primeiro e também o responsável pelo acrônimo TCP/IP. E o UDP (User Datagram Protocol), que veio depois, devido a necessidade de aumentar a velocidade de transmissão em aplicações em tempo real. É importante ressaltar que alguns protocolos da camada de aplicação foram feitos para ser usado somente com o protocolo TCP, ou somente com o protocolo UDP, ou ambos. A maioria dos protocolos de aplicação são definidos sob o protocolo TCP. São exemplos de serviços baseados em TCP a navegação web (HTTP) e a transferência de arquivos (FTP). Serviços definidos sob o protocolo UDP podem ser protocolos de jogos online, vídeo conferência e voIP[[8]](#footnote-8) (Voice over IP).

A camada de rede é responsável por movimentar os pacotes de dados de uma máquina para outra. Esta camada tem dois componentes principais, o primeiro deles é o protocolo IP, que define o famoso conjunto de 4 números separados por pontos, os endereços de IP. O protocolo de IP é único e é ele quem caracteriza os sistemas que utilizam a pilha de dados TCP/IP. O outro componente da camada de rede são os protocolos de roteamento. Estes protocolos só fazem sentido em redes maiores, como as redes locais ou a rede mundial de computadores. Como, na nossa aplicação, estamos interessados apenas na comunicação ponto a ponto entre o computador e o BBB, utilizando a USB, esta segunda parte da camada de rede é irrelevante. Neste caso, o endereço de IP padrão do BeagleBone é . O componente roteamento só se torna relevante se o BBB for conectado à rede local pela porta Ethernet (Item 10 da figuraFigura 5), através de um roteador.

A camada de enlance de dados, que, por sua vez, significa, ligação ou *link* de dados, define as especificações de transmissão, recepção, controle de fluxo, opcionalmente, a correção de erros que venham a ocorrer na camada física. As especificações desta camada estão intimamente ligadas ao meio em que se propagam. A família de protocolo DSL, por exemplo, é muito comum nas bandas largas comerciais, pois foi projetado para utilizar as linhas telefônicas como meio de transmissão. Já o protocolo Wi-Fi foi projetado para ser utilizado em redes locais cujo o meio de transmissão é o ar, assim é tem suporte a senhas de acesso, correção de erro e tem um limite de usuários simultâneos menor do que protocolos para redes sem fio maiores, como o LTE, padrão utilizado na *internet* móvel dos celulares. Nas redes locais com fio é comum utilizar o protocolo Ethernet na camada de enlace.

Por fim, a camada física é o próprio meio de transmissão, ou seja, o ar, cabo coaxial, linhas telefônicas. Como dito anteriormente, é comum que os padrões da camada de enlance seja implementado em conjunto com a camada física, por isso, os circuitos integrados de rede geralmente já implementam as três camadas mais básicas (Rede, enlance e físico) via *hardware*. Por exemplo, as placas de rede dos computadores pessoais especificam que deve ser utilizado como camada física cabos UTP com conectores RJ45 em conjunto com o protocolo Ethernet na camada de enlance e o protocolo IP na camada de rede. O CI SMSC do (Item 6 da figuraFigura 5) segue esta mesma especificação e é utilizado em conjunto com o conector Ethernet 10/100 (Item 10 da figuraFigura 5). Neste caso o BeagleBone deve ser conectado a um roteador como mostra na figuraFigura 10. Como não estamos utilizando a porta USB, o endereço de IP da placa de desenvolvimento será diferente de .

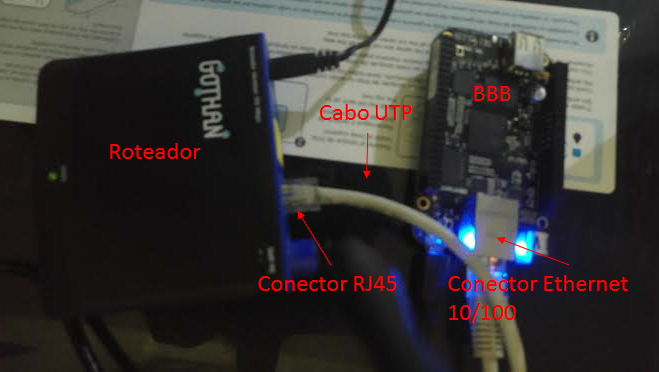


Figura 10: BeagleBone Black conectado a um roteador (Próprio autor).

Um outro cenário possível é a utilização do protocolo Ethernet sobre a porta USB (Ethernet over USB). A porta USB é bastante utilizada para conectar periféricos a computadores e muitas vezes estes periféricos deve utilizar protocolos definidos para funcionar exclusivamente sobre IP, por isso, é comum que impressoras, celulares, *smartphones* e dispositivos embarcados que utilizam a porta USB para se comunicar com o computador, implementem as camadas de enlance e rede via *software* para que seja possível utilizar os serviços disponibilizados por IP.

No caso de dispositivos embarcados com Linux isso é quase uma obrigação, visto que o próprio Linux já vem com driver USB-eth, que implementa os protocolos Ethernet e IP sobre a interface física da USB, criando uma rede local entre cliente e hospedeiro, no nosso caso computador e BeagleBone. A partir daí quase todos os serviços e protocolos disponibilizados por IP estão possíveis, incluindo aí a comunicação remota via SSH. No Windows é importante que os *drivers*, disponibilizados em (BEAGLEBOARD.ORG, 2016), estejam atualizados, pois ambos os lados (Computador e BeagleBone Black) devem estar preparados para este tipo de conexão.

Pelo fato do protocolo Ethernet não ser um padrão definido na comunicação USB, e ainda ser implementado via software, é normal que o desempenho teórico fique abaixo dos 480Mbps teórico da USB 2.0. Segundo o usuário *jons34yp* do site de perguntas e respostas, superuser.com (STACK OVERFLOW, 2013), foi possível atingir a média de 90Mbps em uma comunicação entre um *smartphone* com Android e um computador. Para este trabalho taxas de transferências próximas a 2Mbps são mais do que o suficiente.

### Conectando ao terminal através do SSH

O Linux, Mac e sistemas operacionais baseados no Unix, já vem com suporte nativo ao SSH, que pode ser acessado através do terminal. Entretanto, usuários do Windows devem baixar um cliente SSH como PuTTY, disponível em (TATHAM, 2016). Antes de iniciar o processo de comunicação é importante atualizar os drivers do BeagleBone, disponível em (BEAGLEBOARD.ORG, 2016).

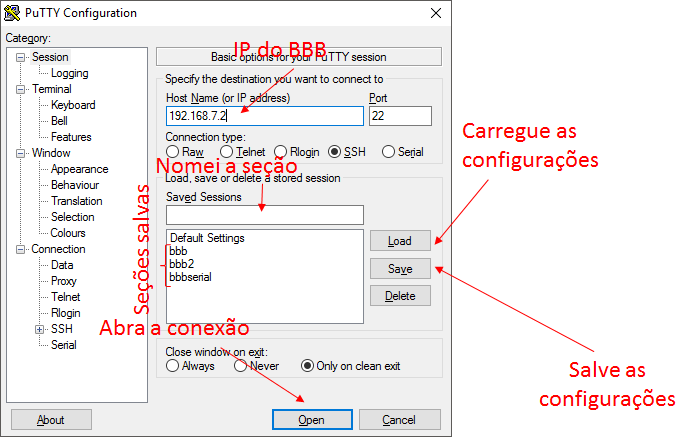


Figura 11 Figura 12: Configurações para conectar o BBB com o computador através do PuTTy (Próprio Autor).

Depois da atualização feita e o PuTTY baixado, abra este aplicativo e no campo Host Name (or IP address) escreva o IP do BeagleBone que por padrão é . Deixe as outras opções nos valores padrões como mostra a figuraFigura 12. Por fim, clique em Open para iniciar uma conexão SSH entre o computador e o BeagleBone Black. Uma janela de terminal se abrirá, inicialmente, pedindo para que o usuário entre com um login e senha. Mais detalhes de como operar o terminal do BeagleBone Black podem ser vistos na seção 2.5.

Existe a possibilidade de criar configurações pré-definidas neste aplicativo. Para isso, preencha os campos e selecione as opções de acordo como o desejado. Depois no campo Saved Sections escreva o nome dessa configuração e clique em Save. Caso o usuário, por algum motivo, feche o PuTTY e abra-o novamente, os campos do aplicativos serão restaurados para os seus valores padrões. Se houver alguma comunicação pré-definida é possível carrega-la rapidamente apertando o botão Load.

### UART e o serial debug

Os protocolos da internet são muito susceptíveis a ataques de maliciosos, geralmente por vírus. Para evitar possíveis ataques os firewalls bloqueiam a maior parte das portas de conexões TCP e UDP, podendo tornar a comunicação SSH indisponível em alguns casos. Em outros casos pode ser que os drivers não funcione corretamente e a conexão por IP entre o BBB e computador não seja estabelecida.

O acrônimo UART significa Universal asynchronous receiver/transmitter, é um tipo de conexão ponto a ponto assíncrona, ou seja, que não precisa do clock para sincronizar os dados. Para que a conexão seja estabelecida são necessários pelo menos dois fios, o RX responsável por receber os dados e o TX responsável por enviar os dados. Fios adicionais podem ser úteis, como um fio para igualar o referencial de terra (GND) entre os dois dispositivos de comunicação. Na figuraFigura 13 têm-se uma representação de uma comunicação serial entre dois dispositivos, incluído o GND. Observe que o RX de um dispositivo é conectado no TX do outro e vice-versa.

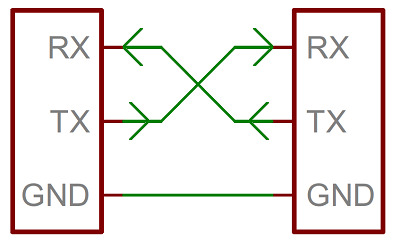


Figura 13: Ligação dos cabos em uma comunicação serial UART (SPARKFUN, 2015).

O protocolo UART tem pacotes pequenos que podem ser configurados de acordo com a aplicação (Figura 14). Para que a conexão funcione ambos os dispositivos devem ser configurados com os mesmos parâmetros. Estes parâmetros inclui as especificações dos pacotes e a quantidade de pacotes por segundo (baud rate), podendo variar o tamanho da palavra (Data bits), quantidade de bits de fim (Stop bits) e correção de erro por paridade (Parity bit), que podem ser vistos na tabelaTabela 3.

| **Parâmetro** | **Valores permitidos** | **UART0** | **Função** |
| --- | --- | --- | --- |
| Data bits | 5 a 9 | 8 | Determina a quantidade de bit de dados enviado em um pacote |
| Stop bits | 1 a 2 | 1 | Quantidade de bits de para determinar o fim do pacote |
| Parity | N/O/M/S | N | Bit de paridade (Correção de erro) |
| Speed (Baud rate) | Vários | 115200 | Velocidade de transmissão dada em pacotes por segundo |
| Flow Control | None | None | O controle de fluxo não é suportado |

Tabela 3: Parâmetros da UART0 e suas funções (Próprio autor).



Figura 14: Os pacotes de dados da comunicação serial podem ser configurados (SPARKFUN, 2015).

O BeagleBone tem 5 portas UART, sendo que a UART0 já vem pré-configurada pelo sistema operacional e tem os parâmetros definidos na tabelaTabela 3. Seu uso exclusivo para comunicação serial com proposito de *debug* e está disponível pelos pinos Serial TTL3V3 Header (Item 13 da figuraFigura 5). O protocolo USB especifica a conexão serial através da classe CDC. Existem cabos ou conversores que já fazem a conversão da UART para a conexão serial especificada pelo protocolo USB, estes cabos são conhecidos como USB Serial TTL. Eles têm de um lado um conector USB macho e do outro lado alguns fios coloridos que devem ser conectados diretamente aos pinos Serial TTL3V3 Header do BeagleBone Black. A coloração dos fios varia de fabricante para fabricante, por isso, antes de fazer as ligações é importante saber é a função de cada fio. Na Wiki do E-Linux, disponível em (E-LINUX.ORG, 2016), têm as principais colorações utilizadas.

### Conectando ao terminal utilizando o serial debug

Verificada a função de cada fio do conversor TTL, conecte os fios aos pinos do Serial TTL3V3 Header como mostra a figuraFigura 15. A ordem de conexão depende da marca do conversor, mas de uma maneira geral deve-se conectar o GND do cabo no pino 1, RX no pino 4 e o TX no pino 5.

Depois de conectado clique com o botão direito no menu iniciar e selecione gerenciador de dispositivos e procure por Portas (COM e LPT) e estará listado o cabo TTL, anote o número da porta COM. Com o número em mãos abra o PuTTY selecione Serial e preencha os campos de acordo com a figuraFigura 16 e clique em Open. A partir daí uma janela de terminal se abrirá e será possível controlar o BeagleBone remotamente semelhante ao protocolo SSH.

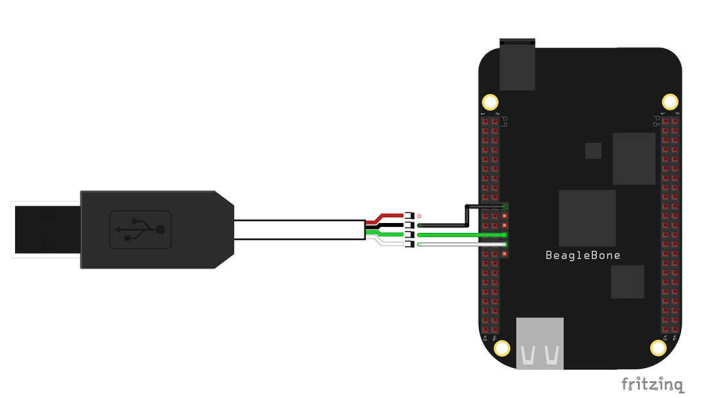


Figura 15: Conectando o cabo TTL ao BBB (SANTOS e PERESTRELO, 2015).

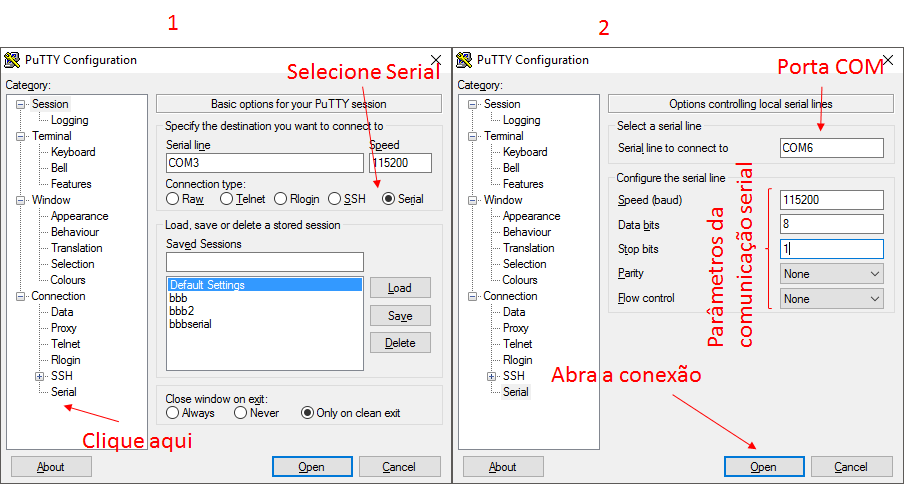


Figura 16: Conectando ao BBB através do serial debug.

## Terminal do BeagleBone Black

Na seção 2.4 foi explicado o que é o terminal e como conectá-lo através do PuTTY utilizando o protocolo SSH ou o serial debug. Nesta seção será explicado como funciona o terminal.

### Usuários no Linux

No Linux existem, basicamente, três tipos de usuários. O primeiro tipo são os usuários comuns. Esses usuários são equivalentes ao que temos no Windows. A função deles é permitir que diferentes pessoas compartilhe o mesmo computador, mas tenha arquivos e configurações pessoais diferentes. Sendo assim, cada usuário comum tem uma pasta ***/home/*<*nome\_do\_usuário*>**. Nela estão guardadas as configurações e arquivos pessoais de cada pessoa, muito parecido com que se tem no Windows.

No BeagleBone o usuário comum é o ***debian***, a senha é ***temppwd***. É possível criar mais usuários, mas, ao contrário do ambiente desktop, isso não é muito trivial em sistemas projetados para tarefas específicas, como este da monografia.

Os usuários comuns têm acesso limitado a arquivos fora da pasta sua ***/home***. Portanto este tipo de usuário não pode fazer operações que depende de manipulações fora desta pasta, como instalar arquivos através do comando ***apt-get***. Para isso, existe um outro tipo de usuário, o ***root***.

O , ou super usuário, é equivalente ao administrador do sistema. O é único, portanto não pode existir mais de um super usuário na mesma máquina. O seu login é sempre o próprio nome, ***root***, e a senha pode variar, mas no caso do BeagleBone Black é em branco.

Normalmente não é recomendado acessar o sistema operacional através do root. Pois, se por acaso este usuário executar algum programa malicioso, este pode ter acesso completo ao sistema e causar problemas. Por isso, alguns sistemas operacionais para desktop não permitem que as pessoas entrem diretamente com o root. Ao invés disso, o indivíduo deve acessar o sistema como um usuário comum e, se necessário, ter acesso temporário ao ***root*** através do prefixo ***sudo*** antes do comando que se deseja executar. Feito isso o terminal irá pedir a senha do ***root*** e ao apertar Enter o comando será executado como super usuário. Após a execução do comando o sistema devolve novamente o acesso ao usuário comum.

Uma outra maneira de acessar o ***root*** quando se está logado através de um usuário comum, é através do comando ***su*** A diferença deste comando para o pré-fixo é que, o primeiro dá acesso permanente ao super usuário ***root***, enquanto o outro, o acesso se dá apenas durante a execução do comando. Isso é útil quando precisa fazer várias operações como ***root***. Somente o super usuário pode executar este comando, portanto deve-se escrever ***sudo su***.

O terceiro tipo são os usuários do sistema. Diferente dos outros dois, essas contas não se conectam. São usadas apenas para propósitos específicos do sistema operacional, por exemplo, o usuário, ***lp*** (line printer), manipula as solicitações de impressão. Na prática, estes usuários trabalham de forma transparente e nada deve ser feito com eles.

### Comandos no terminal

Geralmente os comandos no Linux segue a sintaxe do algoritmoAlgoritmo 1. Assim além do comando em si, é possível adicionar opções e parâmetros. Os parâmetros podem ser obrigatórios ou não, dependendo do comando, enquanto as opções são opcionais, pois, caso não seja passado opções o comando, será executado com os valores padrões.

***$ comando [opções] [parâmetros]***

Algoritmo 1: Sintaxe básica dos comandos no terminal (Próprio autor).

Para ficar mais simples o entendimento será exemplificado um exemplo com o comando ***ls***. Este comando tem a função de exibir conteúdo de arquivos e diretórios. Como parâmetro, o comando recebe o diretório ou arquivo em que se deseja analisar e as opções mais diversas. Sendo alguma delas mostrada na Tabela 4.

| **Opção** | **Função** |
| --- | --- |
| **-a** | Exibe arquivos ocultos |
| **-A** | Não exibe diretórios |
| ***-l*** | Listagem detalhada |
| **-d** | Exibe o diretório especificado e não o conteúdo |
| **-i** | Exibe o número de índice (I-node) dos arquivos |

Tabela 4: Algumas opções do comando (PAULO, 2006)**.**

Observe que todos as opções começam com “**-**“. Este indicador é feito para diferenciar parâmetros de opções. Tudo que começa com “**-**“ é uma opção e o resto são parâmetros. É possível executar um comando com mais de uma opção, por exemplo, ao escrever ***-al*** será exibido os arquivos ocultos de uma determinada pasta de maneira detalhada.

***$ ls***

***ExploringBB***

***$ ls-l***

***total 4***

***drwxr-xr-x 15 root root 4096 Sep 4 18:43 ExploringBB***

***$ ls-al***

***total 12***

***drwxr-xr-x 3 root root 4096 Sep 4 18:42 .***

***drwxr-xr-x 34 root root 4096 Oct 25 12:08 ..***

***drwxr-xr-x 15 root root 4096 Sep 4 18:43 ExploringBB***

***$ ls ExploringBB***

***chp02 chp05 chp08 chp10 chp12 extras License.txt***

***chp03 chp06 chp09 chp11 chp13 library README.md***

Algoritmo 2: Executando o comando com diferentes opções e parâmetros (Próprio autor).

O algoritmoAlgoritmo 2 exemplifica o uso do comando***ls*** com diferente opções e parâmetros numa pasta qualquer do sistema. Veja que o uso de parâmetro não é obrigatório, neste caso o comando irá listar o conteúdo da pasta atual, entretanto, ao escrever o nome da pasta de interesse é exibido o conteúdo da pasta, como mostra no final do algoritmo. Além do comando ***ls*** existem muitos outros. A tabelaTabela 5 lista alguns dos mais utilizados.

| **Comando** | **Sintaxe** | **Função** |
| --- | --- | --- |
| **cd** | ***$ cd [diretório]*** | Seleciona diretório |
| ***more*** | ***$ more [opções] [arquivo]*** | Lista conteúdo de arquivo aos poucos, útil para arquivos grandes |
| ***echo*** | ***$ echo [opções] [texto]*** | Exibe um texto na tela do terminal |
| **apt-get** | ***$ echo [opções][operação]***  ***[pacote]*** | Instala, remove e faz outras funções de gerencialmente de arquivos |
| **shutdown** | ***$ shutdown [opções]***  ***[hora] [mensagem de alerta]*** | Desliga o sistema operacional |
| **clear** | ***$ clear*** | Limpa a tela do terminal |
| **reboot** | ***$ reboot*** | Reinicia a máquina |

Tabela 5: Alguns comandos do terminal (PAULO, 2006).

### Operadores

É possível utilizar alguns operadores para mudar algumas opções nos comandos, como redirecionar a entrada ou saída. Por exemplo, imagem que por algum motivo deseja-se escrever um determinado arquivo usando o comando . Por padrão este comando escreve no console de saída do terminal, mas é possível direcionar esta saída para o arquivo em que se deseja escrever, basta utilizar o operador de redirecionamento de saída, ***>***. Se por acaso utilize-se o operador de redirecionamento de saída para o mesmo arquivo, o comando irá apagar e reescrever com a nova mensagem. Para adicionar uma mensagem ao final do arquivo pode-se utilizar o operador ***>>***.

Além desses operadores existe um outro para redirecionamento de entrada, o ***<***, que pode ser utilizado para mudar a entrada padrão de um determinado comando. Um outro operador bastante útil é o de concatenamento de comandos, o ***|***, com ele pode-se executar mais de um comando de uma única vez. O algoritmoAlgoritmo 3 exemplifica o uso desses operadores.

***$ ls***

***Desktop bin***

***$ echo -e "Estou escrevendo em um arquivo com o comando echo" > arquivo.txt***

***$ ls***

***Desktop arquivo.txt bin***

***$ more arquivo.txt***

***Estou escrevendo em um arquivo com o comando echo***

***$ echo -e "Estou escrevendo de novo" > arquivo.txt***

***$ more arquivo.txt***

***Estou escrevendo de novo***

***$ echo -e "Crie uma nova linha" >> arquivo.txt***

***$ more arquivo.txt***

***Estou escrevendo de novo***

***Crie uma nova linha***

***$ echo "apague tudo" > arquivo.txt | more arquivo.txt***

***apague tudo***

Algoritmo 3: Utilizando alguns operadores para escrever em arquivos com o comando (Próprio autor).

### Arquivos no Linux

Ao contrário do senso comum, no Linux, diretórios e aplicativos são apenas um tipo de arquivo. Em outras palavras, tudo Linux são arquivos, incluindo aí diretórios, aplicativos, partições, pinos de entrada e saída do BeagleBone (GPIO), periféricos como PWM, UART e outros.

Para exemplificar o que isso significa, o mostra como acender o User LED3 (Item 7 na figuraFigura 5) através do terminal. Perceba que tudo que foi feito foi acessar a pasta onde está localizado o LED, entrar neste componente e rescrever a opção ***brightness*** para 1 com o comando ***echo***, neste caso com o pré-fixo ***sudo***, pois o arquivo em questão estava fora da pasta home. Ao entrar no componente há uma mudança de ***$*** para ***#***, isso indica que não se está mais num diretório, mas num componente. O comando ***ls***, neste caso, listou as opções disponíveis para esse LED. É possível alterar as configurações de outros componentes de maneira semelhante.

***$ cd /sys/class/leds***

***$ ls***

***beaglebone:green:usr0 beaglebone:green:usr2***

***beaglebone:green:usr1 beaglebone:green:usr3***

***$ cd beaglebone\:green\:usr3 | ls***

***brightness device max\_brightness power subsystem trigger uevent***

***# sudo echo 1 > brightness***

Algoritmo 4: Acendendo o User LED3 pelo terminal (Próprio autor).

## Programando no BeagleBone Black

O Linux permite que o BeagleBone Black seja programado em diversas linguagens, dentre elas, quatro serão abordadas nesse capítulo C++, Python, JavaScript e Bash.

### Linguagens de script, interpretadas e compiladas

As linguagens de programação de alto nível podem ser divididas em três categorias. A primeira delas são as linguagens compiladas. Compilar em ciência da computação significa transformar o código diretamente em linguagem de máquina, ou seja, em um arquivo binários contendo zeros e uns. O programa compilado, ao ser executado, passa a conversar diretamente com a máquina, sem intermédios. Isso faz com que este tipo de linguagem seja a mais rápida e eficiente, porém, pelo mesmo motivo, acaba perdendo em outros quesitos como portabilidade e facilidade de programação.

Um aplicativo compilado feito para Linux, por exemplo, não pode rodar no Windows. O mesmo vale para arquiteturas diferentes. Neste caso para rodar em outras plataformas o mesmo código deve ser adaptado. Um outro fator negativo desse tipo de linguagem é a dificuldade de se programar. A necessidade de gerar um binário que conversa diretamente com a máquina cria algumas limitações no sentido de abstração, assim o programador deve se preocupar com mecânicas como tipo de variável, endereçamentos e alocação de memória. São linguagens compiladas C++, Pascal e Fortran.

O segundo tipo se refere as linguagens interpretadas. Neste grupo o código também é compilado. Mas, ao contrário do primeiro, o código roda em cima de um programa que serve como intermediário entre a máquina e o arquivo. O programa interpretador, neste caso, é uma máquina virtual criada especificamente para rodar aplicativos criados em uma determinada linguagem interpretada. São exemplos de linguagem interpretada o Java e o JavaScript.

As linguagens interpretadas estão no meio termo quando se refere a performance. A grande vantagem delas é a portabilidade, pois elas são compiladas para rodar em cima de um programa interpretador e se esse programa existir nas mais diversas plataformas o código passa a ser praticamente universal.

O último tipo são as linguagens de script. Neste tipo o código é interpretado linha a linha por um outro programa. No momento de execução do script, o programa interpretador ler uma linha, executa a tarefa, passa para próxima linha e assim sucessivamente. O lado negativo dessa forma de execução é que, geralmente, tem baixa performance comparado a outros métodos.

A tabela Tabela 6 compara a execução de 5000 iterações um algoritmo de performance computacional, n-Body. O programa foi escrito em diferentes linguagens de programação e colocado para rodar em BBBs com diferentes frequências de clock da CPU. Pela tabela, percebe-se que a linguagem interpretada não tem tanta perda de performance em relação a compilada. O mesmo não ocorre com as linguagens de script, o tempo de execução pode chegar a ser 32,2 vezes mais lento que o código compilado em C++. Ainda na mesma tabela, observa-se que a frequência de clock é diretamente proporcional ao tempo de execução, portanto, para aumento de performance, é recomendado configurar para o clock máximo.

| **Linguagem** | **Tipo** | **BBB@1Ghz** | **BBB@800Mhz** | **BBB@300Mhz** |
| --- | --- | --- | --- | --- |
| **C++** | Compilada | (33s) | (42s) | (114s) |
| **Java** | Interpretada | (39s) | (49s) | (134s) |
| **JavaScript** | Interpretada | (78s) | (93s) | (248s) |
| **Perl** | Script | (917s) | (1170s) | (3244s) |
| **Python** | Script | (1063s) | (1294s) | (3586s) |

Tabela 6: Tempo de processamento para a execução de 5000 iterações do algoritmo n-Body em um BBB com Debian, em diferentes frequências (MOLLOY, 2014).

Se por um lado as linguagens de scripts são lentas computacionalmente, por outro lado, são conhecidas por alta versatilidade, facilidade de programação e portabilidade. O fato de não ser compilada permite altas abstrações em relação a máquina, e ainda, a execução do programa ao mesmo tempo em que está sendo escrito. Isso permite que os códigos sejam desenvolvidos e testados rapidamente, podendo ainda, rodar em diferentes arquiteturas sem qualquer modificação no código, tornando-a bastante atrativa para desenvolvimento de projetos. São exemplos de linguagens de script: MATLAB, Python, Lua, Bash e Perl.

### C++

C++ é uma linguagem de médio/alto nível desenvolvida pela Bell Labs durante a década de 80. Ela é uma evolução da linguagem C, desenvolvida pela AT&T na década de 70, sendo que foram incorporados vários conceitos, principalmente aqueles relacionados a programação orientada a objeto.

Diferente de outras linguagens, C++ não pertence a uma única empresa. Durante anos houve contribuições de várias as partes e, em 1998, a ISO passou a padronizar a linguagem C++ para remover as inconsistências entre os vários compiladores de C++ das diferentes organizações. O padrão ISO deu força a linguagem, e ainda hoje, é uma das mais utilizadas no mundo.

Para compilar um código em C++ pode-se utilizar o comando ***g++*** como mostra no exemplo do algoritmoAlgoritmo 5. Neste comando o primeiro parâmetro é o código fonte. Por padrão o nome de arquivo de saída é ***a.out***, para poder escolher o nome do arquivo de saída, pode-se utilizar a opção ***-o*** seguido do nome do arquivo de saída, neste caso ***olamundo***.

Para executar o arquivo compilado basta escrever ***./<nome\_do\_arquivo>***. Observe que, diferente do Windows, não é necessário que o arquivo binário tenha a extensão ***.exe*** para ser executado.

***$ g++ olamundo.cpp -o olamundo***

***$ ./olamundo***

***Ola mundo!***

Algoritmo 5: Compilando em C++ (Próprio autor).

A linguagem C++ é conhecida por criar programas com alta eficiência, entretanto, ela tem algumas desvantagens. A principal delas está relacionada ao suporte da linguagem no BeagleBone Black. Programar em C++ é uma tarefa um tanto complicada para usuários menos experientes, por isso a própria comunidade propõe o uso de linguagens mais simples e diretas de se programar como JavaScript, e principalmente, o Python. Isso implica que a maior parte do material para leitura, códigos e bibliotecas estão disponível nessas linguagens.

Além disso, por padrão o C++ não suporta interface gráfica, protocolos de comunicação e outras funções relacionadas aos periféricos do BeagleBone Black. Portanto, caso não exista alguma biblioteca desenvolvida para uma determinada função, o caminho se torna muito mais longo. Aliado a isso, existe a necessidade que estas bibliotecas sejam adaptadas para o BeagleBone, visto que o código criado em C++ não é portável, ou seja, boa parte das bibliotecas criadas para desktop não podem ser usada diretamente no BBB. Como a comunidade de C++ é menor é normal que se tenha menos coisas prontas para esta linguagem.

### Bash

A linguagem Shell é bastante genérica. Na verdade, a linguagem foi evoluindo e várias pessoas criaram seus próprios programas interpretadores que incluía novos comandos e mecânicas. Isso fez com que existissem várias versões da linguagem Shell, semelhante ao que ocorria com o C++ antes da padronização pela ISO.

Para resolver o problema de compatibilidade o usuário deve digitar qual interpretador será utilizado (Algoritmo 6), caso essa linha não seja escrita o sistema operacional irá escolher o interpretador padrão. No Linux, o interpretador de comandos mais utilizado é o Bash e também o padrão em muitas distribuições, inclusive do Debian, por isso, muitas vezes o programa funciona mesmo com a ausência da linha do algoritmoAlgoritmo 6. Todavia, é recomendado a inclusão dessa linha, pois nem sempre o interpretador padrão é o Bash.

***#!/bin/bash***

***LED3\_PATH=/sys/class/leds/beaglebone:green:usr3***

***echo 1 > $LED3\_PATH/brightness***

Algoritmo 6: Código Bash para acender o User LED3 (Próprio autor).

Depois de escolher o interpretador, a programação se torna parecida com a execução de comandos do terminal, com a vantagem de poder utilizar ifs, fors e variáveis. Muitas vezes, os comandos do terminal são aplicações criadas em Shell, muitas vezes utilizando o interpretador Bash. Essas aplicações, por sua vez, interagem com os arquivos da mesma forma que poderia ser feito através da linha de comando.

Shell não é uma linguagem muito poderosa e nem tem muitos recursos além daqueles relacionados ao sistema operacional. Por isso, sua utilização se resume a operações relacionadas a manipulação e execução de arquivos (Lembre-se que arquivos é qualquer coisa no Linux).

### JavaScript

JavaScript é uma linguagem originalmente desenvolvida para a web e, por isso, tem uma comunidade forte e bastante desenvolvida, incluindo grandes empresas como a Google. Embora o JavaScript seja uma linguagem interpretada, ela se assemelha muito com as linguagens de script, pois não é necessário compilar a linguagem antes de sua execução, neste caso, a compilação ocorre momentos antes da execução, de forma transparente para o usuário.

JavaScript é uma linguagem de excelente desempenho, pois existe um esforço muito grande das empresas de tecnologia para que ela se torne tão eficiente quanto um código compilado. Pode-se afirmar que o desempenho da linguagem aumenta a cada ano.

No BeagleBone Black esta linguagem é implementada através do . Uma plataforma de desenvolvimento de aplicações, cujo o programa interpretador é o V8 JavaScript Engine, a mesma componente utilizada pelo navegador Google Chrome para executar JavaScript, desenvolvida pela Google. Com o o BeagleBone passa a funcionar como um servidor, ou seja, caso haja uma conexão com o meio externo, por exemplo entre o computador e o BBB, é a placa que comanda os serviços fornecidos ao cliente, neste caso o computador.

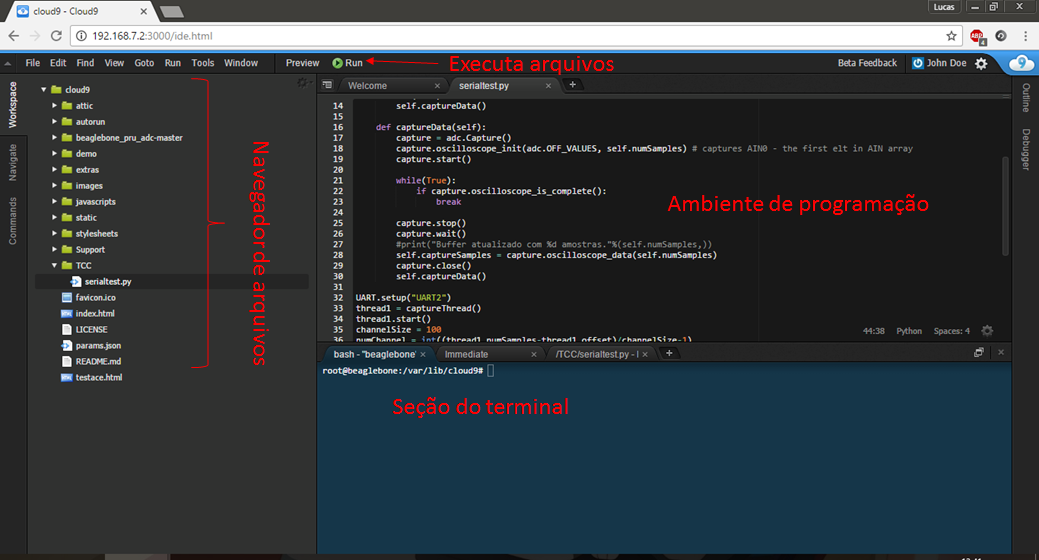
A grande vantagem dessa linguagem em relação ao C++ é que existe um suporte muito forte no BBB. O foi colocado no BeagleBone Black propositalmente, pela própria fabricante, inclusive os tutoriais iniciais da BeagleBone.org ensinam a programar nesta linguagem. Aliado a isso, a própria fabricante desenvolveu várias bibliotecas para acesso e configuração dos periféricos.

Por ser uma linguagem interpretada, as bibliotecas e frameworks desenvolvidas para JavaScript não precisam ser adaptadas como as C++. Além disso, a própria comunidade está continuamente desenvolvendo soluções para do BBB com esta linguagem.

### Python

### Cloud9 IDE

Cloud9 é uma IDE (Integrated Development Environment) que roda no navegador através do protocolo HTTP. Ela é uma ferramenta que aceita várias linguagens como C, C++, PHP, Ruby, Perl, Python, Bash e JavaScript. Além disso, permite acessar o terminal pelo próprio ambiente e tem um navegador de arquivos.



Algoritmo 7: Ambiente de desenvolvimento Cloud9 (Próprio autor).

Essa ferramenta também tem a possibilidade de executar diretamente aplicativos criados em Python, , Shell e Ruby através do botão *Run*, de tal forma que, para programar nestas linguagens não é necessário digitar nenhuma linha no terminal, tornando esta ferramenta uma mão na roda para usuários menos experientes e facilitando o trabalho dos mais avançados. O ambiente de desenvolvimento já vem integrado por padrão em todos os BeagleBone Black e para acessá-lo basta digitar o endereço do BeagleBone Black na porta 3000, neste caso .

## Device Tree Overlays

## BeagleBone Black e tarefas em tempo real

# Bibliografia

7-ZIP.ORG. **7-Zip**, 2016. Disponivel em: <http://www.7-zip.org/>. Acesso em: 24 de outubro 2016.

BANZ, M. **How Arduino is open-sourcing imagination**, 2012. Disponivel em: <https://www.ted.com/talks/massimo\_banzi\_how\_arduino\_is\_open\_sourcing\_imagination>. Acesso em: 04 de outubro 2016.

BEAGLEBOARD.ORG. **Latest Firmware Images**, 2016. Disponivel em: <https://beagleboard.org/latest-images>. Acesso em: 2016 de outubro 2016.

BEAGLEBOARD.ORG. **Getting Started**, 2016. Disponivel em: <http://beagleboard.org/static/beaglebone/latest/README.htm>. Acesso em: 24 de outubro 2016.

BLAKEBEVIN. **TPower Laces- the Auto lacing shoe**, 2010. Disponivel em: <http://www.instructables.com/id/Power-Laces-the-Auto-lacing-shoe/>. Acesso em: 30 de setembro 2016.

COLEY, G. **BeagleBone Black System Reference Manual**. A5.2. ed. Dallas: Texas Instruments, 2013. Disponivel em: <https://cdn-shop.adafruit.com/datasheets/BBB\_SRM.pdf>. Acesso em: 18 de outubro 2016.

DAVIS, T.; DAVIS, J. **Win32 Disk Imager**, 2016. Disponivel em: <https://sourceforge.net/projects/win32diskimager/>. Acesso em: 2016 de outubro 2016.

DEBIAN, 2016. Disponivel em: <https://debian.beagleboard.org/images/>. Acesso em: 24 de outubro 2016.

DICOLA, T. **Embedded Linux Board Comparison**, 2014. Disponivel em: <https://cdn-learn.adafruit.com/downloads/pdf/embedded-linux-board-comparison.pdf>. Acesso em: 04 de outubro 2016.

E-LINUX.ORG. **Toolchains**, 2016. Disponivel em: <http://elinux.org/Toolchains>. Acesso em: 24 de outubro 2016.

E-LINUX.ORG. **Beagleboard:** BeagleBoneBlack Debian, 2016. Disponivel em: <http://elinux.org/Beagleboard:BeagleBoneBlack\_Debian>. Acesso em: 24 de outubro 2016.

E-LINUX.ORG. **Beagleboard:** Updating The Software, 2016. Disponivel em: <http://elinux.org/Beagleboard:Updating\_The\_Software>. Acesso em: 24 de outubro 2016.

KOEN. **The Ångström Distribution**, 2015. Disponivel em: <http://wp.angstrom-distribution.org/introduction/>. Acesso em: 05 de outubro 2016.

LINUX, E. B. T. A. T. F. B. W. E. **Exploring BeagleBone:** Tools and Techniques for Building with Embedded Linux. [S.l.]: Wiley, 2014.

MACHADO, J. **Raspberry Pi:** computador de 50 reais pode revolucionar a informática, 2012. Disponivel em: <http://www.tecmundo.com.br/hardware/23175-raspberry-pi-como-um-computador-de-50-reais-pode-revolucionar-a-informatica.htm>. Acesso em: 04 de outubro 2016.

MEASUREMENT COMPUTING CORPORATION. Data Aquisition Handbook, Norton, 2012. Disponivel em: <http://www.mccdaq.com/pdfs/anpdf/Data-Acquisition-Handbook.pdf>. Acesso em: 26 de setembro 2016.

MOLLOY, D. **Exploring BeagleBone:** Tools and Techniques for Building with Embedded Linux. [S.l.]: Wiley, 2014. Disponivel em: <http://www.exploringbeaglebone.com/>.

NI. **Aquisição de dados**, 2016. Disponivel em: <http://www.ni.com/data-acquisition/pt/>. Acesso em: 25 de Setembro 2016.

NVIDIA. **O que é computação acelerada por placa de vídeo?**, 2016. Disponivel em: <http://www.nvidia.com.br/object/what-is-gpu-computing-br.html>. Acesso em: 17 de outubro 2016.

PAANKIEWICZ, I. **O que é um port?**, 2009. Disponivel em: <http://www.tecmundo.com.br/video-game-e-jogos/1910-o-que-e-um-port-.htm>. Acesso em: 21 de outubro 2016.

PEREIRA, A. L. **O que é PCI-Express?**, 2012. Disponivel em: <http://www.tecmundo.com.br/hardware/1130-o-que-e-pci-express-.htm>. Acesso em: 26 de setembro 2016.

PRACTICAL MICRO. **K9 CNC I/O Cape for the BeagleBone Black**, 2014. Disponivel em: <http://www.pmdx.com/PMDX-432>. Acesso em: 05 de outubro 2016.

STACK OVERFLOW. **The best way to do TCP/IP over USB on Linux**, 2013. Disponivel em: <http://superuser.com/questions/593757/the-best-way-to-do-tcp-ip-over-usb-on-linux>. Acesso em: 13 de outubro 2016.

TATHAM, S. **Download PuTTY - a free SSH and telnet client for Windows**, 2016. Disponivel em: <http://www.putty.org/>. Acesso em: 24 de outubro 2016.

1. Padrão de barramento interno de computadores para conectar periféricos. [↑](#footnote-ref-1)
2. Linguagem de programação de baixo nível onde cada palavra representa o código de alguma instrução de uma máquina que usa uma determinada arquitetura. [↑](#footnote-ref-2)
3. Arquitetura de processador 32 bits de alto poder de processamento, baixo consumo e baixa dissipação de calor que permitiu surgimento de smartphones como iPhone. [↑](#footnote-ref-3)
4. Tipo de processador especializado em processar gráficos. Hoje em dia as GPU estão cada vez mais versáteis permitindo processar outras tarefas como equações matemáticas, física e inteligência artificial (NVIDIA, 2016). [↑](#footnote-ref-4)
5. Embedded Linux, ou Linux embarcado, é o termo utilizado para distribuições Linux portadas para ARM com o intuito de funcionar em sistemas embarcados. [↑](#footnote-ref-5)
6. Regulador de tensão capaz de regular a voltagem de saída quando a tensão de estrada está próxima da tensão de saída [↑](#footnote-ref-6)
7. Port, no que tange os softwares, se trata do processo de adaptar um código criado para uma plataforma computacional de modo que o resultado final seja compatível com outras plataformas (PAANKIEWICZ, 2009). [↑](#footnote-ref-7)
8. Tecnologia de comunicação que utiliza o Internet Protocol para transferir chamadas telefônicas e serviços de voz de uma maneira geral. [↑](#footnote-ref-8)