Chapter
12

Sistemas embarcados: explore sua criatividade construindo Hardware e Software

Thiago Moratori Peixoto, Tiago Machado, Luciano Jerez Chaves,
Eduardo Pagani Julio

Abstract

This course aims to present embedded systems and its several applications on science and technologies fields. It will present some basic concepts about embedded systems and digital microelectronics. Moreover, this course will focus on the platforms Arduino and Mbed, both used for digital systems prototyping. It will also describe the hardware of these devices, as well as the basics principles for programming their middleware. The concepts covered will be exercised in applications developed during practical workshops.

Resumo

Este minicurso tem como objetivo geral apresentar sistemas embarcados e suas infinitas aplicações no campo da ciência e tecnologia. Serão apresentados conceitos básicos de sistemas embarcados e microeletrônica digital. Além disto, o minicurso dará ênfase no desenvolvimento para as plataformas de prototipagem Arduino e Mbed, utilizadas no projeto de sistemas digitais. O minicurso apresentará tanto o hardware destes dispositivos, quanto os fundamentos para a programação do middleware*. Os conceitos abordados serão exercitados em aplicações reais desenvolvidas durante oficinas práticas.*

12.1. Introdução

Os sistemas digitais tornaram-se parte do nosso dia-a-dia devido ao modo intenso pelo qual os circuitos digitais e as técnicas digitais passaram a ser utilizados em quase todas as áreas: computadores, automação, robôs, tecnologias e ciências médicas, transportes, telecomunicações, entretenimento, e assim por diante [Tocci et al. 2011].

Para chegarmos onde estamos hoje, a construção de sistemas digitais evoluíram desde o uso das grandes válvulas eletrônicas, passando pelos transistores até chegamos nos atuais circuitos integrados (CI) digitais. Essa evolução foi responsável por permitir a

implementação de sistemas digitais cada vez mais complexos em um menor espaço físico, utilizando técnicas precisas e sofisticadas durante a fabricação dos CIs [Moore 1998]. O baixo custo de fabricação até mesmo dos complexos circuitos integrados (como é o casos dos microprocessadores que usamos em nossos computadores, *notebooks* e *smartphones*), permite que os projetistas incluam cada vez mais funcionalidades nestes sistemas, tornando-os adequados para as mais diversas aplicações. Entretanto, esse baixo custo de fabricação está associado à produção em larga escala, o que inviabiliza o projeto e fabricação de alguns poucos CIs apenas para fins de teste, ou para utilização em aplicações muito específicas em pequena escala.

Neste contexto, surgiram os Dispositivos Lógicos Programáveis (PLDs em inglês), que são componentes eletrônicos usados para construir circuitos digitais reconfiguráveis. Diferente dos circuitos lógicos tradicionais, que possuem um funcionamento lógico fixo definido durante o projeto, os PLDs saem das fábricas sem nenhum comportamento fixo, podendo ser programados posteriormente pelos desenvolvedores de acordo com suas necessidades e se tornando propícios para o uso em casos de aprendizagem, prototipagem e testes em pequena escala.

O uso dos PLDs levou à criação de microcontroladores. Também conhecidos como MCU, são um computador num *chip*, contêm um processador, memória e periféricos de entrada e saída (E/S). O microcontrolador é como um microprocessador que pode ser programado para funções específicas, em contraste com outros microprocessadores de propósito geral, como os utilizados em computadores. Diferente dos PLDs, os microcontroladores não são apenas um circuito programável, mas um sistema completo que já oferece funcionalidades básicas como gerência de memória e acesso a dispositivos de E/S. Para facilitar ainda mais o desenvolvimento de sistemas utilizando microcontroladores, eles normalmente são acoplados à plataformas de prototipagem, como é o caso do Arduino [Arduino 2012a] e do Mbed [Holdings 2011]. Essas plataformas oferecem não somente o microcontrolador, mas também outros componentes para realizar a interface com computadores (como portas USB, Ethernet, RS-232, etc.), componentes de microeletrônica básicos (LEDs e botões), além de outros dispositivos de E/S. Toda a programação destas plataformas é feita através de *softwares* que são instalados em computadores. Através dos ambientes desenvolvidos pelos próprios fabricantes, é possível programar todo o funcionamento desses microcontroladores, além de enviar o código já compilado para o dispositivo, normalmente através de uma interface USB.

Neste contexto, este minicurso visa apresentar duas das várias plataformas de prototipagem disponíveis: o Arduino e o Mbed. Ambas são plataformas com bom poder de processamento e recursos, que permitem a implementação desde aplicações didáticas simples até projetos de pequeno e médio porte, criados a partir da imaginação dos desenvolvedores mais aventureiros. Uma análise com maior precisão classificaria o Arduino como uma plataforma voltada principalmente para uso educacional, pois é uma plataforma *OpenSource* própria para a integração com dispositivos eletrônicos básicos, além de apresentar uma interface de programação simplificada com um conjunto de bibliotecas básicas já disponíveis para uso. Por sua vez, o Mbed apresenta um *hardware* mais rápido e potente, incluindo também mais recursos embutidos (como controladores de rede) e bibliotecas de *software* ainda mais sofisticadas, viabilizando projetos maiores.

O restante deste minicurso apresenta ao leitor as plataformas de prototipagem Arduino e Mbed, incluindo algumas atividades práticas no uso destes dispositivos. Para isso, a Seção 12.2 apresenta alguns conceitos de microeletrônica básica, necessários para que possamos entender até mesmo as aplicações mais simples. A Seção 12.3 apresenta as plataformas de prototipagem eletrônicas Mbed e Arduino, destacando suas características técnicas. A Seção 12.4 apresenta os ambientes de desenvolvimento e as bibliotecas utilizadas para a programação de ambos dispositivos. A Seção 12.5 traz alguns exemplos de aplicações que podem ser construídas com estes equipamentos e descreve as atividades didáticas propostas como oficina prática. Por fim, a Seção 12.6 apresenta as considerações finais.

12.2. Microeletrônica básica

Para poder montar um sistema digital interativo é preciso algum tipo de interface de comunicação com os usuários. Esta interface é normalmente construída através de dispositivos eletrônicos, que são conectados à plataforma de prototipagem. Os componentes eletrônicos básicos mais utilizados serão apresentados nesta unidade.

12.3. Plataformas de prototipagem

Esta unidade visa apresentar em detalhes as plataformas de prototipagem abordadas: Mbed e Arduino. São discutidos as especificações técnicas e os recursos oferecidos por cada um dos dispositivos.

12.3.1. Mbed

Os microcontroladores Mbed consistem de uma série de microcontroladores elaborados para prototipagem rápida. São projetados com encapsulamento DIP (*dual in-line package*) para montagem em placas *protoboard* sem solda, *stripboards* e *trough-hole*. Existem duas variantes da plataforma Mbed: a NXP LPC11U24 e a NXP LPC1768, sendo esta última que será utilizada nesse mini-curso de tal forma que todas as referências são relacionadas a ela [Mbed 2012c]. A Figura 12.1 ilustra o Mbed modelo NXP LPC1768 [Mbed 2012b].

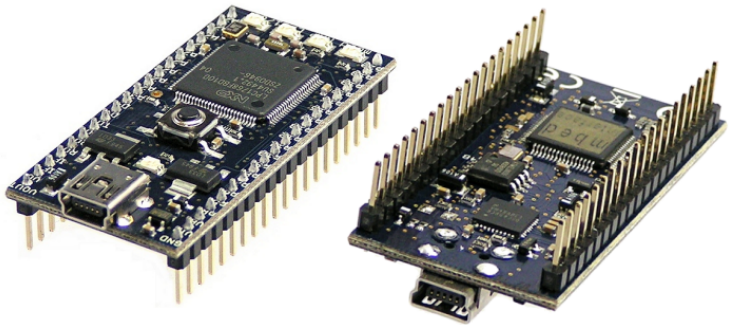

Figure 12.1. Visão frontal do Mbed NXP LPC1768.

Esses microcontroladores incluem uma interface USB de acesso para programação. Para uso, basta se conectar o dispositivo a uma porta USB do computador, copiar o programa ARM binário para a memória *flash* para que o processador seja capaz de executar o código. Essa interface é compatível com os sistemas Windows, Mac OS X e Linux,

desta forma a plataforma é indiferente ao sistema do Mbed. O programa binário pode ser obtido utilizando-se um compilador em "nuvem", disponível através do *site* do próprio fabricante [Holdings 2011].

O Mbed possui inúmeras portas de conexões. No total, são 25 pinos disponíveis para o LPC1768. Através destes pinos podem ser incorporados dispositivos e funcionalidades como interface de rede Ethernet, leitores e gravadores RFID, leitores biométricos, câmeras, teclados numéricos, telas LCD *touch screen*, entre outros. Além disto, esses microcontroladores podem ser alimentados através da interface USB de um computador ou através de fontes externas com voltagem operando entre 4.5V e 12V. A Figura 12.2 ressalta as principais características deste modelo de microcontrolador.

Característica	**Valor**
Aplicações alvo	Ethernet, USB e de alto desempenho
Núcleo de processamento	ARM Cortex-M3
Frequência de operação	96 MHz
Memória Flash	512 KB
Memória RAM	32 KB
Alimentação	60-120 mA

Figure 12.2. Principais características de hardware do NXP LPC1768.

Esse modelo também oferece as seguintes interfaces periférica: suporte para controlador Ethernet, USBHost e USBDevices, SPI, I2C, CAN, AnalogIn, PwmOut, AnalogOut, DigitalIn e DigitalOut. Cada uma delas são detalhadas logo abaixo.

- **Ethernet**: permite que os microcontroladores Mbed se conectem e se comuniquem com uma rede Ethernet, podendo ser utilizado para conversar com outros dispositivos na rede, incluindo comunicação com computadores através da web, servidores de email ou atuar com um *webserver* (utilizada nos pinos p33/34/p35/p36);

- **USB**: com essa interface é possível utilizar o Mbed para, por exemplo, emular mouse e teclado, emular uma porta serial, enviar e receber mensagens MIDI, tocar músicas ao ser reconhecido como um dispositivo de áudio e receber pacotes de áudio vindos do computador (utilizada nos pinos p31 e p32);

- **SPI** (*Serial Peripheral Interface*): fornece uma interface periférica serial mestre que permite a comunicação com dispositivos SPI escravos, como memória *flash*, telas LCD e outros módulos ou circuitos integrados (utilizada nos pinos p5/p6/p7 ou p11/p12/p13);

- **I2C** (*Inter-Integrated Circuit*): fornece funcionalidade mestre I2C, que por sua vez propicia a comunicação com dispositivos I2C escravos, como memória serial, sensores e outros módulos ou circuitos integrados (utilizada nos pinos p9/p10 ou p27/p28);

- **CAN** ou Controlador de Área de Rede: é um padrão de barramento projetado para permitir que os microcontroladores e dispositivos se comuniquem uns com os outros sem um computador *host* (utilizada nos pinos p9/p10 ou p29/p30);

- **AnalogIn**: usada para ler a tensão aplicada a um pino de entrada analógica (pode utilizar qualquer pino de p15 a p20);
- **PwmOut**: utilizada para controlar a frequência de uma sequência de pulsos digitais (pode utilizar qualquer pino de p21 a p26);
- **AnalogOut**: utilizada para ajustar a tensão de um pino de saída analógica (só pode utilizar o pino p18);
- **DigitalIn**: utilizada para ler o valor de um pino de entrada digital (pode utilizar qualquer pino de p5 a p30);
- **DigitalOut**: utilizada para configurar e controlar um pino de saída digital (pode utilizar qualquer pino de p5 a p30).

A Figura 12.3 pode ser utilizada como um guia, pois mostra a numeração dos pinos e suas funcionalidades.

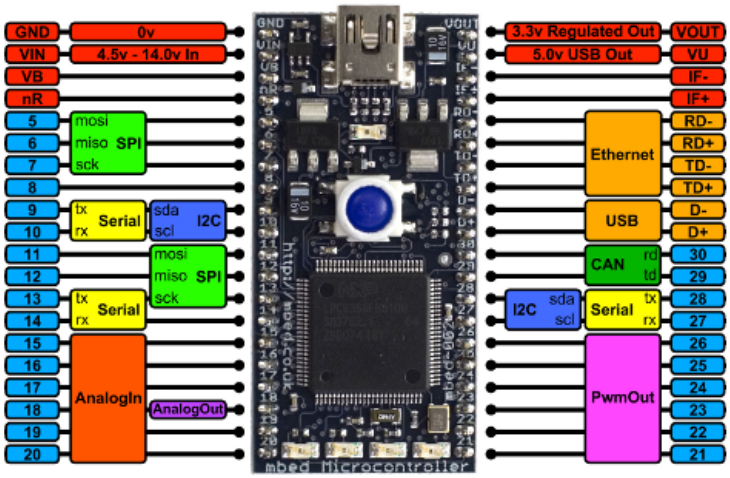

Figure 12.3. Diagrama de conexões do Mbed NXP LPC1768.

12.3.2. Arduino

O Arduino é uma plataforma *Open Source* de prototipagem eletrônica baseada em *hardware* e *software* flexíveis e fáceis de usar [Culkin 2012]. O Arduino é capaz de "interagir" com o ambiente através de sinais de entrada provenientes de uma variedade de sensores e modificá-lo através de sinais de saídas para atuadores, motores, luzes controladoras (acendendo ou apagando um LED por exemplo), etc. O microcontrolador na placa é programado usando a linguagem de programação do Arduino (baseado no *framework Wiring* [Barragán 2012], essencialmente C/C++) e o ambiente de desenvolvimento Arduino (baseada no ambiente de desenvolvimento *Processing* [Fry 2012]). Projetos Arduino podem ser independentes ou podem comunicar com outros *softwares* que estejam rodando em um computador. As placas podem ser compradas com facilidade, e o *software* de desenvolvimento pode ser baixado gratuitamente. Existem inúmeros projetos de referência do Arduino disponíveis na Web, o usuário é livre para adaptá-los às suas

necessidades. Ao contrário do Mbed, o Arduino possui vários modelos diferentes, que variam na capacidade de processamento e quantidade de pinos para conexões externas. Esse minicurso irá tratar apenas da versão UNO, de tal forma que referências posteriores devem ser relacionadas a essa versão. A Figura 12.4 ilustra a visão frontal do Arduino UNO, identificando ainda suas principais conexões.

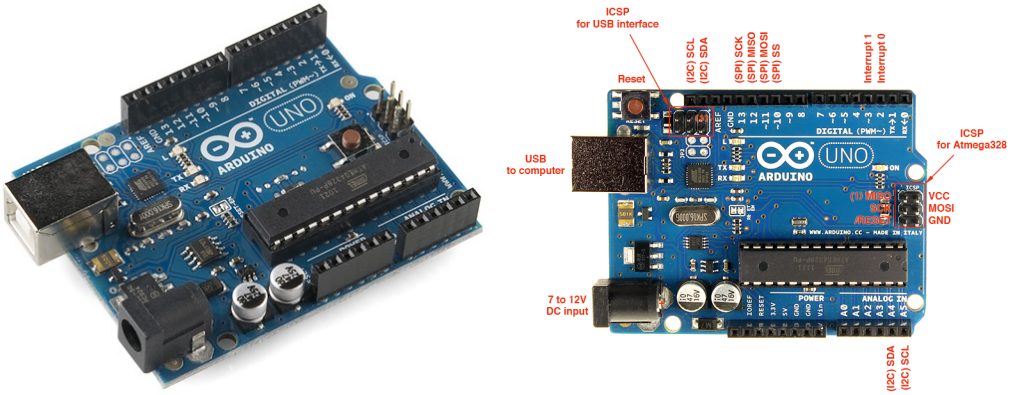

Figure 12.4. Visão frontal do Arduino UNO rev 3.

O Arduino Uno é uma plataforma de prototipagem baseada no microcontrolador ATmega328. A plataforma possui 14 pinos digitais que podem funcionar como pinos de entrada ou saída digital, 6 entradas analógicas, um cristal oscilador de 16 MHz, uma porta de conexão USB, um conector de alimentação, e um botão de *reset*. Esta plataforma contém tudo o que é necessário para suportar o microcontrolador. Basta conectá-la a um computador com um cabo USB ou ligá-la em uma fonte de energia para que a mesma comece a funcionar. A Figura 12.5 abaixo lista as principais características de *hardware* do Arduino.

Característica	**Valor**
Microcontrolador	ATmega328
Voltagem de Operação	5V
Voltagem recomendada	7-12V
Voltagem máxima	6-20V
Pinos Digitais de E/S	14
Pinos Analógicos	6
Corrente pino de E/S	40 mA
Corrente pino de 3.3V	50 mA
Memória Flash	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidade de Clock	16 MHz

Figure 12.5. Principais características de hardware do Arduino UNO.

Cada um dos 14 pinos digitais do Arduino Uno podem ser utilizados tanto como pinos de entrada ou de saída. Essa configuração é feita em *software*, durante o desenvolvimento do aplicativo. Todos os pinos operam com 5 Volts e uma corrente máxima de

40 mA. Cada pino tem um resistor *pull-up* interno (desconectado por padrão) de 20-50 kOhms. Além disso, alguns pinos têm funções especializadas, como os indicados abaixo:

- Pinos seriais 0 (RX) e 1 (TX) podem ser usados para receber (RX) e transmitir (TX) dados seriais;
- Pinos de interrupções externas 2 e 3, que podem ser configurados para disparar uma interrupção por um alguma variação específica em seus sinais de entrada;
- Pino LED número 13, onde há um LED *built-in* que acende quando o pino está com valor alto, e apaga caso contrário.

O Arduino Uno tem também 6 entradas analógicas, rotuladas de A0 a A5, que podem ser conectadas em sensores como termômetros, sensores de luminosidade, etc. Cada um provê 10 bits de resolução (1024 valores diferentes). Além disso há também um botão de *reset*, usado para reiniciar o microcontrolador reiniciando toda a placa quando acontece algum problema que bloqueia o funcionamento da mesma.

Uma das principais características do Arduino é que ele é um projeto livre, onde todas as especificações estão publicamente disponíveis na Web. Isso permite que desenvolvedores possam construir suas próprias plataformas, ou mesmo modificar as existentes para atender às suas necessidades. A Figura 12.6 ilustra o projeto de referência do Arduino Uno.

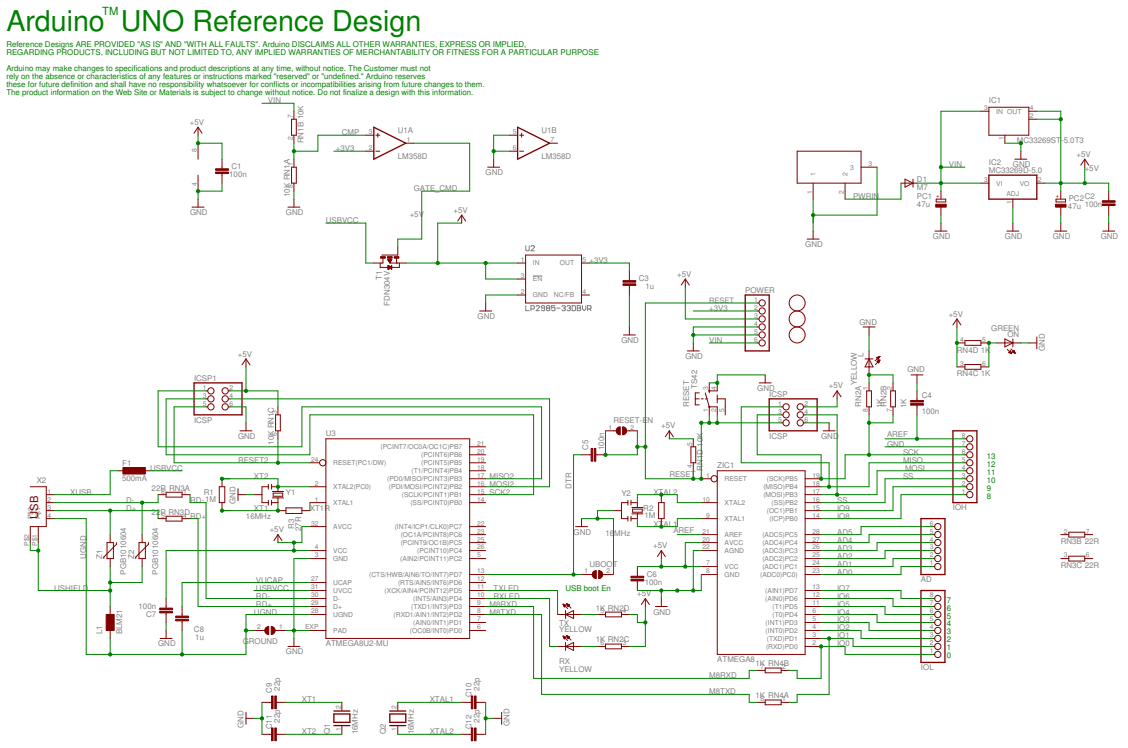

Figure 12.6. Projeto de referência para o Arduino Uno.

12.4. Ambientes de desenvolvimento

Para concluir a parte teórica, nesta unidade serão apresentados os ambientes de desenvolvimento integrado que são disponibilizados pelos fabricantes dos dispositivos. Estes ambientes são utilizados para programar o *middleware* que será executado pelo dispositivo. Além do aplicativo, esta unidade aborda também a utilização das bibliotecas já existentes, que facilitam o trabalho de implementação de aplicações maiores.

12.4.1. Mbed

O compilador Mbed fornece um IDE (ambiente de desenvolvimento integrado) *online* C/C++ que é pré-configurado para permitir que se escreva e compile programas rapidamente e seja capaz de transferi-los para executar em seu microcontrolador Mbed. Não é necessário que se instale ou configure nada para começar a programar com o Mbed. Por ser uma aplicação web, o ambiente de desenvolvimento fornecido permite que seja possível trabalhar em qualquer navegador no Windows, Mac ou Linux e que se tenha acesso em qualquer lugar e continue a partir de onde parou. Cada usuário Mbed possui uma conta que lhe da acesso a um compilador e a um *workspace* privado que contém os seus programas [Mbed 2012a].

Apesar de ser leve e *online*, o compilador também é poderoso. Ele utiliza a *engine* profissional do compilador ARMCC, e portanto produz código eficiente que pode ser usado gratuitamente, mesmo em aplicações comerciais. O IDE possui um editor completo que possui marcador de palavras reservadas, atalhos de teclado padrão do editor, função de copiar/colar, tabulação e até mesmo auto-formatação de código. A Figura 12.7 retrata o ambiente pessoal com múltiplos arquivos, pastas e programas em que o usuário irá desenvolver.

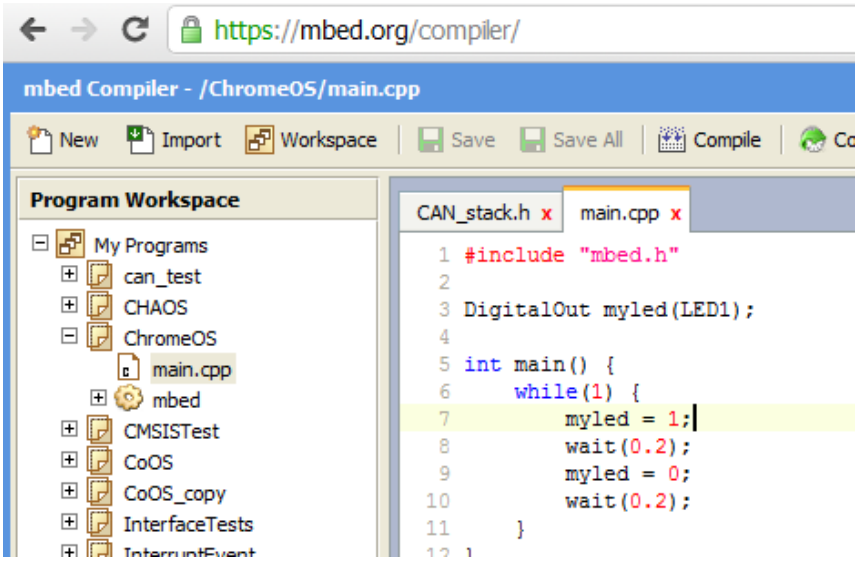

Figure 12.7. Ambiente de desenvolvimento para Mbed.

O sistema de pesquisa da IDE permite que o usuário procure por uma *string*, seja num programa inteiro ou apenas no módulo requisitado. No exemplo representado pela Figura 12.8 temos a pesquisa da *string* "*myled*" no programa ChromeOS em todos os

módulos e tipos de módulos (`.c`, `.cpp`, `.h`, `.hpp`, `.s`). Na mesma figura temos o resultado da pesquisa.

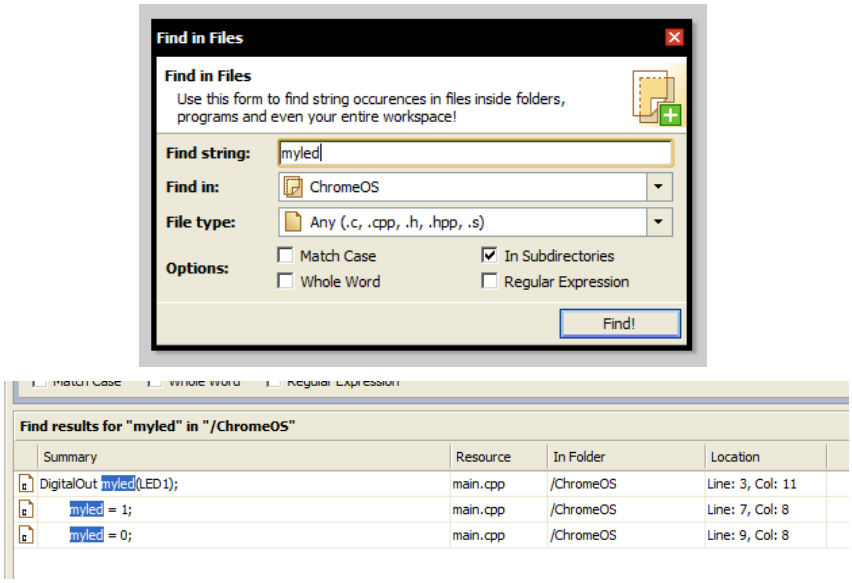

Figure 12.8. Sistema de buscas do ambiente de desenvolvimento Mbed.

O ambiente de desenvolvimento integrado do Mbed possui ainda um controle de versão embutido que permite o usuário fazer operações de *commit* de uma versão do projeto, ver o histórico de revisões, comparar modificações feitas entre versões, fazer operações de *update* ou *revert* para uma nova versão, e operações de *branch* e *merge* entre versões. A Figura 12.9 abaixo mostra um exemplo de operação de versionamento.

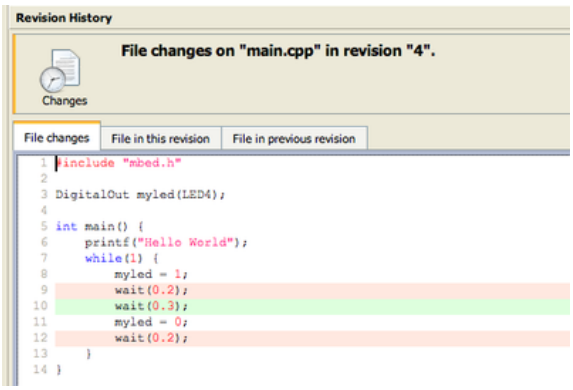

Figure 12.9. Sistema de controle de versão embutido do Mbed.

Outra funcionalidade é o assistente de importação, que permite a importação de programas a partir de uma URL Mbed. Isso é útil para importar rapidamente para dentro do projeto do usuário um código que tenha sido empacotado como uma biblioteca reutilizável (por exemplo, uma classe para um periférico). A Figura 12.10 abaixo mostra o sistema de importação de arquivos.

O IDE ainda permite que o usuário publique seus programas ou bibliotecas facilmente na seção de desenvolvimento do site do Mbed. Esse recurso facilita o processo

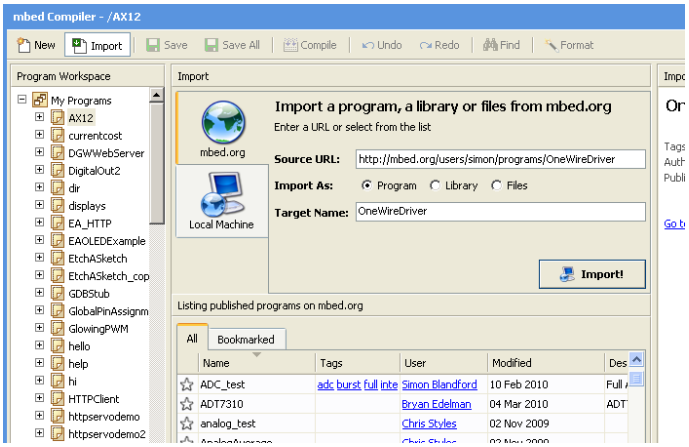

Figure 12.10. Sistema de importação de programas, bibliotecas do Mbed.

de compartilhamento tanto de códigos bem sucedidos quanto de eventuais problemas e consequente *feedback* para ambos os casos.

Para realizar a compilação, o compilador Mbed utiliza a *compiler engine ARM RVDS 4.1* na configuração padrão para dar excelentes tamanho do código e desempenho. Não há limitações no tamanho do código (salvo os limites do próprio dispositivo). O código gerado pode ser usado livremente para uso comercial e não-comercial. Quando você compila um programa, você vai ter uma exibição do uso de memória. Isso mostra o espaço que o código irá ocupar na memória *flash* do dispositivo, e as variáveis que acabam em memória principal. A Figura 12.11 mostra um exemplo de uso de memória.

O SDK usado com o compilador *online* do Mbed também é compatível com uma série de outras ferramentas ARM populares, de tal forma que seja possível exportar os programas diretamente para essas ferramentas. Por exemplo, se o usuário desejar migrar para um conjunto de ferramentas diferentes quando o projeto passa da fase de protótipo, pode-se optar por exportar um projeto Mbed clicando com o botão direito sobre ele conforme exemplificado na Figura 12.11.

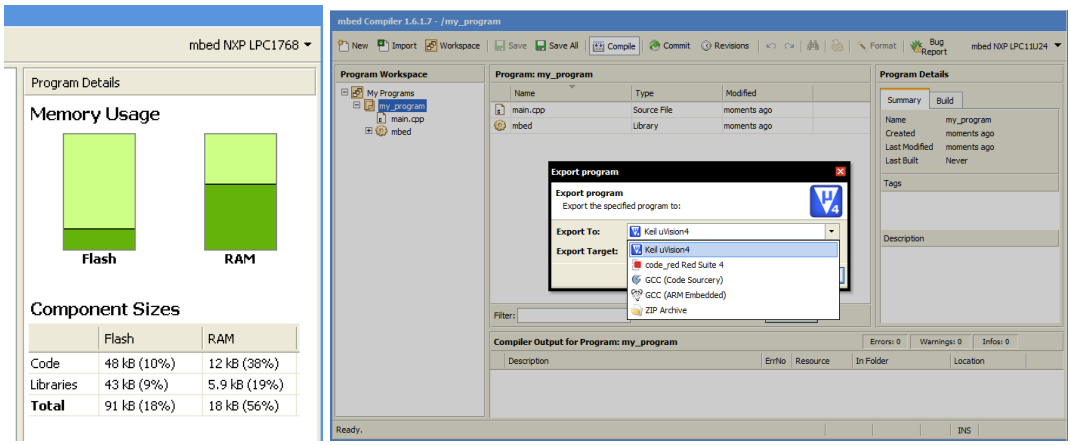

Figure 12.11. Uso de memória e opções de exportação de programa.

Para o leitor interessado em aumentar seus conhecimentos em relação ao desen-

volvimento para Mbed, sugerimos consultar uma referência mais completa sobre programação, como as que podem ser encontradas na página do projeto [Holdings 2011].

12.4.2. Arduino

Assim como o *hardware* do Arduino, o *software* também é *Open Source* e multiplataforma (roda em Windows, Linux e OS X), baseado no projeto *Processing* [Fry 2012]. Ele é implementado em linguagem Java, e todo o código está disponível publicamente na web. O ambiente é simplista e intuitivo, além de ser compatível com todas as versões do Arduino. Ele contém um editor de texto para escrita de código, uma área de mensagem, um console em texto, uma barra de ferramentas com botões para as funcionalidades mais comuns, e uma série de menus. Ele se conecta ao *hardware* do Arduino para fazer o *upload* de programas e se comunicar com o mesmo. A Figura 12.12 mostra este ambiente de desenvolvimento.

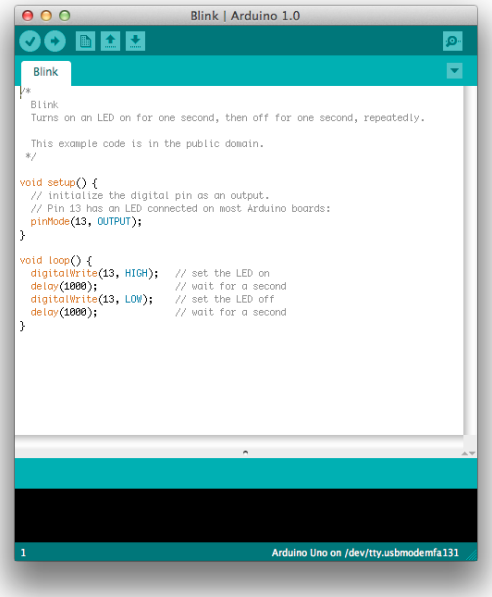

Figure 12.12. Ambiente de desenvolvimento para Arduino.

As principais funcionalidades oferecidas pelo ambiente e utilizadas constantemente pelos desenvolvedores contam com botões na barra de ferramentas. Na ordem em que eles aparecem na Figura 12.12, nós temos o botão de **Verificar**, responsável por analisar todo o código em busca de erros; e o botão de **Upload**, que compila o código e envia para o dispositivo através da conexão USB com o mesmo. Ainda na barra de ferramenta, há também os botões para **Novo**, **Abrir** e **Salvar**.

Os *softwares* escritos usando Arduino são chamados de *sketches*. Estes códigos são verificados quanto a erros ainda no ambiente de desenvolvimento, para somente depois serem compilados e enviados ao *hardware* através do processo de *upload*. O processo de atualização do Arduino é bastante facilitado graças ao *bootloader* previamente instalado no microcontrolador. Um cabo USB e alguns cliques de mouse são suficientes para ter um sistema inteiro rodando no *hardware* do Arduino.

A sintaxe da linguagem de desenvolvimento para o Arduino é baseada na linguagem de programação de alto nível C. Ela oferece praticamente as mesmas estruturas básicas da linguagem C: condicional usando `if`, `then` e `else`; `switch` e `case`; e estruturas de repetição `while`, `do while` e `for`; funções e procedimentos; `structs`; etc. Dessa forma, é transparente para um desenvolvedor já habituado com a linguagem C a começar implementar pequenos códigos neste ambiente.

O ambiente de desenvolvimento para Arduino conta também com série de bibliotecas que provêem funcionalidades extras que podem ser utilizadas nos *sketches*, como funções para manipular configurações avançadas do *hardware* ou manipular dados com maior precisão. Para utilizar um biblioteca é preciso carregar os arquivos através das funcionalidades já oferecidas pelo ambiente.

Para o desenvolvimento de um *sketch*, duas funções obrigatórias devem estar presentes em todos os códigos: `void setup()` e `void loop()`. A primeira função é executada apenas uma vez, no início da execução do programa. Suas principais aplicações são para inicialização de variáveis, inicialização da utilização de bibliotecas, definição das funcionalidades dos pinos e início da comunicação serial, quando aplicável. Por sua vez, a segunda função é executada em *loop* contínuo, se comportando como o programa principal (função `main()`). A partir desta função é permitido chamar outras funções, sejam elas implementadas pelo próprio desenvolvedor, ou já disponíveis em bibliotecas carregadas previamente. O código abaixo exemplifica uma aplicação simples para Arduino, que analisaremos na sequência.

```
 1. void setup () {
 2.   pinMode (12, OUTPUT);
 3. }
 4.
 5. void loop () {
 6.   digitalWrite(12, HIGH);
 7.   delay(1000);
 8.   digitalWrite(12, LOW);
 9.   delay(1000);
10. }
```

Este é o código chamado Blink. Seu objetivo é fazer com que um LED que esteja com seu terminal positivo (anodo) conectado no pino de saída 12 do Arduino pisque constantemente com a frequência de 1 Hz. Neste exemplo estamos assumindo que o terminal negativo (catodo) do LED esteja aterrado e que o resistor apropriado esteja sendo utilizado para limitar a corrente. Analisando o *sketch*, nele temos declarado no `setup()` que usaremos o pino 12 do Arduino como um pino de saída de dados (OUTPUT). Ou seja, é por ele que vai sair o sinal que vai controlar o LED. O caso alternativo a este é definir um pino como sinal digital de entrada (INPUT), como por exemplo, um pino conectado a um botão.

Já na função `loop()`, temos duas tarefas distintas sendo executadas:

1. A função `digitalWrite(12,HIGH);` diz para o Arduino que o pino 12 deve

ter como saída o valor lógico 1. Isso, na prática, significa que esse pino terá como saída +5V, que será utilizado para acender o LED.

2. A função `digitalWrite(12,LOW);` faz o processo inverso, dizendo para o Arduino que o pino 12 deveria se tornar agora o polo negativo, onde não há fluxo de energia para que o LED fique desligado.

3. Por último, a função `delay(1000);` é utilizada para inserir uma pausa (atraso) na execução das próximas funções. Neste caso, o programa fica parado por 1000 ms antes de continuar e reiniciar a execução da função `loop()`.

Neste exemplo, teremos o LED ligado por 1s e desligado por 1s. Como todo o código de ligar e desligar o LED está no `loop()`, que executa infinitas vezes, teremos o LED piscando o tempo todo. É importante observar que se não inserirmos o atraso, então essas operações seriam realizadas tão rapidamente que seria impossível percebermos o LED piscar. A Figura 12.13 mostra como deveriam ser conectados os componentes externos na placa para realmente visualizar o funcionamento deste *sketch* (não se esqueça do resistor). Uma variação deste código pode ser conseguida facilmente adicionando um botão para controlar o funcionamento do LED, onde somente quando o botão estivesse pressionado o LED permanece aceso (atuando como um interruptor).

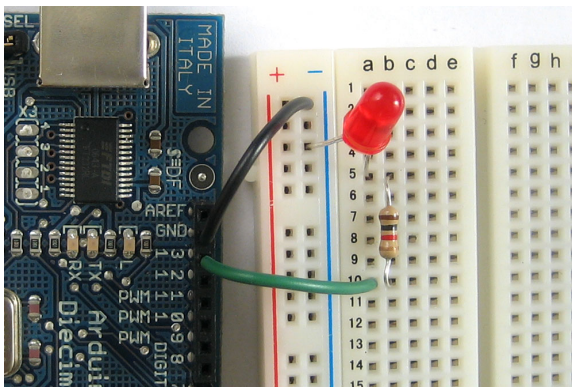

Figure 12.13. Protoboard com o projeto Blink em funcionamento.

Para o leitor interessado em aumentar seus conhecimentos em relação ao desenvolvimento para Arduino, sugerimos consultar uma referência mais completa sobre programação, como as que podem ser encontradas na página do projeto [Arduino 2012b] e no livro [McRoberts 2011].

12.5. Atividade prática

12.6. Considerações finais

References

[Arduino 2012a] Arduino (2012a). Arduino. Disponível em: `http://www.arduino.cc`.

[Arduino 2012b] Arduino (2012b). Learning arduino. Disponível em: http://www.arduino.cc/en/Tutorial/HomePage.

[Barragán 2012] Barragán, H. (2012). Wiring. Disponível em: http://wiring.org.co.

[Culkin 2012] Culkin, J. (2012). Arduino. Disponível em: http://www.jodyculkin.com/wp-content/uploads/2012/04/arduino-comic-latest.pdf.

[Fry 2012] Fry, B. (2012). Processing. Disponível em: http://processing.org.

[Holdings 2011] Holdings, A. (2011). Rapid prototyping for microcontrollers. *Cambridge, UK. Disponível em: http://www.mbed.org*.

[Mbed 2012a] Mbed (2012a). Mbed compiler. Disponível em: http://mbed.org/handbook/mbed-Compiler.

[Mbed 2012b] Mbed (2012b). Mbed NXP LCP11U24. Disponível em: http://mbed.org/handbook/mbed-NXP-LPC11U24.

[Mbed 2012c] Mbed (2012c). Microcontroladores. Disponível em: http://mbed.org/handbook/mbed-Microcontrollers.

[McRoberts 2011] McRoberts, M. (2011). *Arduino Básico*. Novatec.

[Moore 1998] Moore, G. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82 –85.

[Tocci et al. 2011] Tocci, R. J., Widmer, N. S., and Moss, G. L. (2011). *Sistemas Digitais: princípios e aplicações*. Pearson, 15th edition.

12.7. Minicurrículo dos autores

Thiago Moratori Peixoto é graduando em Ciência da Computação pela Universidade Federal de Juiz de Fora - UFJF. Atualmente atua como desenvolvedor em bolsa de treinamento profissional intitulada "Desenvolvimento de um Sistema de Segurança Utilizando Dispositivos Embarcados". As áreas de interesse em pesquisa e atuação são Redes de Computadores, Sistemas Distribuídos e Sistemas Embarcados.

Tiago Machado recebeu seu diploma de Bacharel em Ciência da Computação pela Universidade Federal de Juiz de Fora - UFJF (2009). Atualmente cursa pós-graduação *latu sensu* em Redes de Computadores nesta mesma instituição. Suas áreas de interesse em pesquisa e atuação são: Redes de Computadores, Sistemas Distribuídos e Sistemas Embarcados. Desde 2010 é Técnico Administrativo em Educação do Departamento de Ciência da Computação na UFJF. Atua ainda em projetos de Extensão e Pesquisa na área de gerência de redes, sistemas embarcados entre outros.

Luciano Jerez Chaves recebeu seu diploma de Bacharel em Ciência da Computação pela Universidade Federal de Juiz de Fora - UFJF (2007) e de Mestre em Ciência da Computação pela Universidade Estadual de Campinas - UNICAMP (2010). Atualmente é aluno de doutorado nesta mesma instituição, e seus interesses de pesquisa incluem Redes de Computadores, Algoritmos Cognitivos e Computação Autonômica. Desde 2011 é Professor Assistente no Departamento de Ciência da Computação da UFJF, lecionando disciplinas na área de Arquitetura de Computadores e Sistemas Digitais. Seus trabalhos recentes estão relacionados com implementações de algoritmos cognitivos para redes sem fio em dispositivos embarcados.

Eduardo Pagani Julio recebeu seu diploma de Bacharel em Informática pela Universidade Federal de Juiz de Fora - UFJF (2000) e de Mestre em Computação pela Universidade Federal Fluminense - UFF (2007). Atualmente é aluno de doutorado nesta mesma instituição, e seus interesses de pesquisa incluem Redes de Computadores e Redes de Rádios Cognitivos. Desde 2011 é Professor Assistente no Departamento de Ciência da Computação da UFJF, lecionando disciplinas na área de Redes de Computadores e Segurança de Sistemas Computacionais. Seus trabalhos recentes estão relacionados com implementações de sistemas de segurança em dispositivos embarcados.