TEEE I-Projeto de Robôs Móveis

Microcontrolador

Os avanços tecnológicos atingidos com o desenvolvimento dos circuitos Integrados demandam cada vez mais dispositivos eletrônicos e a cada dia são criados componentes mais versáteis e poderosos. Nesta categoria, estão os microcontroladores que apresentam uma excelente performance. Permitem o projeto relativamente rápido e fácil de novos equipamentos, devido a sua facilidade de uso em ampla faixa de aplicações.

Portanto, faz-se uma introdução das principais características de um microcontrolador, sua arquitetura e recursos importantes para o desenvolvimento de projetos de automação e controle. Em seguida apresenta-se as características do microcontrolador da família Atmel, o atmega 8.

1. Aspectos gerais

Um microcontrolador contém um processador, acesso a memória e periféricos de entrada/saída. Está comprovada sua eficiência para pequenos protótipos que necessitem de baixa velocidade de processamento e pouca memória.

Basicamente, o uso de um microcontrolador consiste no processamento de dados obtidos em um de seus periféricos, tendo como saída outro conjunto de dados. Por exemplo, envio de dados via porta serial, acender um LED (saída) como reação a uma informação de entrada , etc.

Os microcontroladores têm agregado inúmeras funcionalidades, tais como: gerador interno independente de *clock*; memória SRAM, EEPROM e FLASH; conversores A/D, D/A; vários temporizadores/contadores; comparadores analógicos; PWM; diferentes tipos de interface de comunicação,

incluindo USB, UART, I2C, CAN, SPI, JTAG; relógios de tempo real; circuitos para gerenciamento de energia no chip; circuitos para controle de *reset*, alguns tipos de sensores; interface para LCD; e outras funcionalidades de acordo com o fabricante.

Apresenta-se na Tab. 1, uma lista das várias famílias dos principais fabricantes. A coluna Núcleo indica o tipo de arquitetura ou unidade de processamento que constitui a base do microcontrolador, a coluna IDE lista o nome dos ambientes de desenvolvimento que podem ser baixados do site da internet de cada fabricante.

Tab. 1.– Principais fabricantes de microcontroladores .

Fabricante	Internet	Barramento	Família	Arquitetura	Núcleo	IDE
Analog Device	www.analog.com	8-bits	ADUC8xx	CISC	8051	-
		32-bits	ADUC7xx	RISC	ARM7	-
Atmel	www.atmel.com	8-bits	AT89xxx	CISC	8051	independent programs
		8-bits	AVR	RISC	-	AVR studio
		32-bits	AVR32	RISC	-	-
		32-bits	AT91xxx	RISC	ARM7/9	-
Cirrus Logic	www.cirrus.com	32-bits	EP73xxx	RISC	ARM7	-
		32-bits	EP93xxx	RISC	ARM9	-
Cygnal	www.silabs.com	8-bits	C8051F	CISC	8051	-
Freescale (ex. Motorola)	www.freescale.com	8-bits	HC05	CISC	6800	-
		8-bits	HC08	CISC	6809	Code Warrior
		8-bits	HC11	CISC	6809	-
		16-bits	HC12	CISC	-	-
		16-bits	HCS12	CISC	-	Code Warrior
		16-bits	HC16	CISC	-	-
		16-bits	56800	CISC	-	-
		32-bits	68K	CISC	68000	-
		32-bits	ColdFire	CISC	-	-
		32-bits	MAC7100	RISC	ARM7	-
Fujitsu	www.fujitsu.com	8-bits	F2MC-8	CISC	-	-
		16-bits	F2MC-16	CISC	-	-
		32-bits	FR	RISC	-	-
Infineon	www.infineon.com	8-bits	C5xxx, C8xxx	CISC	8051	-
		16-bits	C16xxx	CISC	-	-
		16-bits	XC16xxx	CISC	-	-
		32-bits	TCxxx	CISC	-	-
Intel	www.intel.com	8-bits	MCS251	CISC	8051	-
		16-bits	MCS96/296	CISC	-	-
Maxim (Dallas)	www.maxim-ic.com	8-bits	DS80Cxxx	CISC	8051	-
		8-bits	DS83Cxxx	CISC	8051	-
		8-bits	DS89Cxxx	CISC	8051	-
		16-bits	MAXQ	RISC	-	-
Microchip	www.microchip.com	8-bits	PIC-10,12,16,18	RISC	-	MPLAB
		16-bits	dsPIC	RISC	-	MPLAB

Fabricante	Internet	Barramento	Família	Arquitetura	Núcleo	IDE
NS	www.national.com	8-bits	COP8xxx	CISC	-	Webench
		16-bits	CR16Cxxx	CISC	-	-
		16-bits	CP3000	RISC	-	-
Philips	www.semiconductors.philips.com	8-bits	P8xxx	CISC	8051	-
		16-bits	Xaxxx	CISC	-	-
		32-bits	LPC2000	RISC	ARM7	-
Rabbit Semiconductor	www.rabbitsemiconductor.com	8-bits	Rabbit2000	CISC	-	-
		8-bits	Rabbit3000	CISC	-	-
Renesas	www.renesas.com	8-bits	740	CISC	-	-
		16-bits	H8	CISC	-	HEW
		16-bits	H8S	CISC	-	HEW
		16-bits	M16C	CISC	-	-
		16-bits	7700	CISC	-	-
		32-bits	H8SX	CISC	-	-
		32-bits	Super H	CISC	-	HEW
ST	www.stm.com	8-bits	ST5	CISC	-	Visual FIVE
		8-bits	ST6	CISC	-	-
		8-bits	ST7	CISC	-	STVD-7
		8-bits	ST9	CISC	-	STVD-9
		16-bits	ST9	CISC	-	STVD-9
		16-bits	ST10	CISC	-	-
		32-bits	ARM7	RISC	ARM7	-
Texas Instruments	www.ti.com	8-bits	MSC12xxx	CISC	8051	-
		16-bits	MSP430	CISC	-	Eclipse
		32-bits	TMS470	RISC	ARM7	-
Toshiba	chips.toshiba.com	8-bits	870	CISC	-	-
		16-bits	900/900H	CISC	-	-
		32-bits	900/900H	CISC	-	-
Ubicom (ex.Scenix)	www.ubicom.com	8-bits	SXxx	RISC	-	-
Zilog	www.zilog.com	8-bits	Z8xxx	CISC	Z80	-
		8-bits	Z8Encore-!	CISC	Z80	-
		8-bits	eZ80Acloim	CISC	Z80	-

(fonte: revista Elektor 02/2006).

A arquitetura de um microcontrolador em geral consiste em um núcleo de processamento, barramento e periféricos:

- **Núcleo de processamento** consiste no processador de dados (cálculos, controle de fluxo de programa, etc) e na administração dos periféricos;
- **Barramento** é dividido em dados e endereços, consiste nas linhas de comunicação entre o processador e os periféricos;

- **Periféricos** caracterizam o conjunto de funcionalidades disponíveis pelo microcontrolador e são controlados pelo processador. Por exemplo, memória, porta serial, porta paralela e conversor A/D.

Quando se fala em barramentos em processadores, existem dois tipos de arquitetura, a **Von-Neuman** e a arquitetura **Harvard**.

A **arquitetura Harvard** (figura 1-a direita) possui os barramentos separados para instruções e dados, permitem larguras diferentes, com isso o barramento de instruções é otimizado para uma palavra de comprimento único. O número de bits do barramento de instruções **depende de quantas instruções são implementadas e do número de registradores** disponíveis em cada família de microcontroladores.

A arquitetura **Von-Neumman** (tradicional) (figura1-a esquerda) utiliza o mesmo barramento para fazer a busca à instruções na memória de programa e para acessar (escrever ou ler) na memória de dados.

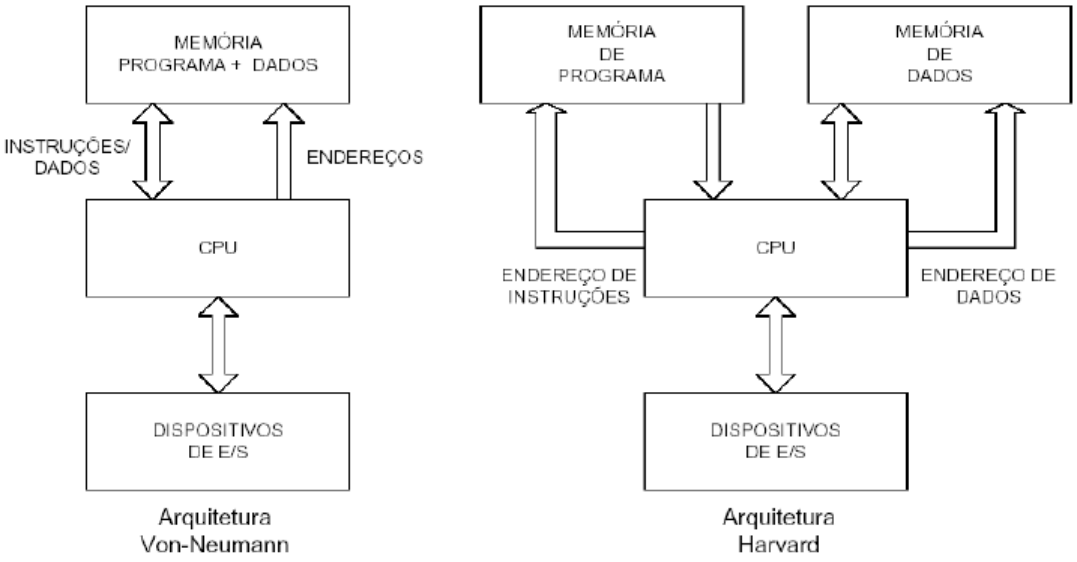
Pode-se dizer que a primeira é uma arquitetura paralela e a segunda serial .A arquitetura Von-Neumann permite produzir um conjunto complexo de código de instruções para o processador (CISC – *Complex Instructions Set Computer*), com um tempo de execução por instrução de vários ciclos de *clock;* é mais simples, com menor número de portas lógicas, entretanto, sua velocidade é menor que a Havard.

A arquitetura Havard produz um conjunto simples de códigos de instruções e, devido ao paralelismo de sua estrutura, é capaz de executar uma instrução por ciclo de *clock*. A arquitetura Havard necessita de mais linhas de código para executar a mesma tarefa que uma arquitetura Von-Neumann, a qual possui muito mais tipos de instruções.

Nos dias atuais, nas modernas arquiteturas de microcontroladores há um domínio da Harvard, a qual evoluiu para a chamada arquitetura **Harvard estendida** ou **avançada**. Sendo composta por um grande número de instruções e ainda com a redução da quantidade necessária de portas lógicas, produz um núcleo de processamento compacto, veloz e com programação eficiente (menor número de linhas de código). Devido às questões de desempenho, compatibilidade eletromagnética e economia de energia, hoje é praticamente inaceitável que um moicrocontrolador não execute a maioria das

instruções em poucos ciclos de *clock* (o que diminui o consumo e a dissipação de energia).

Figura 1 – Arquiteturas Clássicas de processadores: Von-Neuman x Harvard.



Fonte: Apostila atmega -SC

Microcontroladores são geralmente utilizados em automação e controle de produtos e periféricos, como sistemas de controle de motores automotivos, controles remotos, máquinas de escritório e residenciais, brinquedos, sistemas de supervisão, embarcados, robotica, etc.

2. Os Microcontroladores AVR

Os microcontroladores AVR foram desenvolvidos na Noruega em 1995 e são produzidos pela ATMEL. Apresentam ótima eficiência de processamento e núcleo compacto (poucos milhares de portas lógicas), possuem uma estrutura RISC avançada, com mais de uma centena de instruções e uma arquitetura voltada à programação C, a qual permite produzir códigos compactos. Por causa de sua arquitetura, o desempenho do seu núcleo de 8 bits é equivalente ao desenvolvido por microcontroladores de 16bits.

As principais características dos microcontroladores AVR são:

- Executam poderosas instruções em um simples ciclo de *clock* e operam com tensões entre 1,8 e 5,5 V, com velocidades de até 20 MHz. Estão disponíveis em diversos encapsulamentos (de 8 até 64 pinos);

- Alta integração e grande número de periféricos com efetiva compatibilidade entre toda a família AVR ;

- Possuem vários modos para redução do consumo de energia;

- Possuem 32 registradores de propósito geral, memória de acesso *load-store* e a maioria das instruções é de 16bits;

- Memória de programação FLASH programável *in-system*, SRAM e EEPROM, para desenvolvimentos rápidos e flexibilidade de projeto.

- Facilmente programados e com debug *in-system* via interface simples, ou com interfaces JTAG compatível com 6 ou 10 pinos;

- Preço acessível e um conjunto completo e gratuito de softwares;

Existem microcontroladores AVR específicos para diversas áreas, tais como: automotiva, controle de LCDs, redes de trabalho CAN, USB, controle de motores, controle de lâmpadas, monitoração de bateria, 802.15.4/ZigBee™ e controle por acesso remoto.

2.1 A família AVR

Os principais componentes da família AVR são:

- **tinyAVR® - ATtiny -** µcontroladores de propósito geral de até 8 kbytes de memória Flash, 512 bytes de SRAM e EEPROM.

- **megaAVR® - ATmega -** microcontroladores de alto desempenho com multiplicador por hardware, com até 256 kbytes de memória FLASH, 4 kbytes de EEPROM e 8 kbytes de SRAM.

- **picoPower™ AVR -** microcontroladores com características especiais para economia de energia.

- **XMEGA™ ATxmega-** microcontroladores XMEGA 8/16-bit dispõem de novos e avançados periféricos com aumento de desempenho, DMA (*Direct Memory Access*) e sistema de eventos.

- **AVR32** (não pertence às famílias acima) - microcontroladores de 32 bits com arquitetura RISC projetada para maior processamento por ciclos de *clock*, conjunto de instruções para DSP (*Digital Signal Processing*) com SIMD (*Single Instruction, Multiple Data*) com soluções SoC (*System-on-a-chip*) e completo suporte ao Linux.

As Tabelas 2 e 3 apresentam as principais características dos AVRs ATmega e ATtiny.

Tabela 2- Comparação entre os ATmega (04/2009)

8-bit Timer	Brown Out Detector	Ext Interrupts	Hardware Multiplier	Interrupts	ISP	On Chip Oscillator	PWM Channels	RTC	Self Program Memory	SPI	TWI	UART	Wat
2	Yes	8	Yes	34	Yes	Yes	8	Yes	Yes	1	Yes	2	
2	Yes	32	Yes	57	Yes	Yes	16	Yes	Yes	1+USART	Yes	4	
2	Yes	17	Yes	48	Yes	Yes	9	Yes	Yes	1+USART	Yes	2	
2	Yes	3	Yes	35	Yes	Yes	6	Yes	Yes	Yes	Yes	2	
2	Yes	3	Yes	28	Yes	Yes	6	Yes	Yes	1	--	2	
2	Yes	32	Yes	31	Yes	Yes	6	Yes	Yes	1+USART	Yes	2	
2	Yes	32	Yes	31	Yes	Yes	6	Yes	Yes	1+USART	Yes	2	
2	Yes	17	Yes	23	Yes	Yes	4	Yes	Yes	1+USI	USI	1	
2	Yes	26	Yes	26	Yes	Yes	6	Yes	Yes	1+USART	Yes	1	
2	Yes	26	Yes	26	Yes	Yes	6	Yes	Yes	1+USART	Yes	1	
2	Yes	3	Yes	20	Yes	Yes	4	Yes	Yes	1	Yes	1	
2	Yes	32	Yes	57	Yes	Yes	16	Yes	Yes	1+USART	Yes	4	
2	Yes	17	Yes	48	Yes	Yes	9	Yes	Yes	1+USART	Yes	2	
2	Yes	32	Yes	31	Yes	Yes	6	Yes	Yes	1+USART	Yes	2	
2	Yes	32	Yes	31	Yes	Yes	6	Yes	Yes	1+USART	Yes	2	
2	Yes	17	Yes	23	Yes	Yes	4	Yes	Yes	1+USI	USI	1	
2	Yes	17	Yes	32	Yes	Yes	4	Yes	Yes	1+USI	USI	1	
2	Yes	17	Yes	32	Yes	Yes	4	Yes	Yes	1+USI	USI	1	
2	Yes	17	Yes	23	Yes	Yes	4	Yes	Yes	1+USI	USI	1	
2	Yes	26	Yes	26	Yes	Yes	6	Yes	Yes	1+USART	Yes	1	
2	Yes	3	Yes	19	Yes	Yes	4	Yes	Yes	1	Yes	1	
2	Yes	26	Yes	26	Yes	Yes	6	Yes	Yes	1+USART	Yes	1	
2	Yes	8	Yes	34	Yes	Yes	8	Yes	Yes	1	Yes	2	
2	Yes	32	Yes	57	Yes	Yes	16	Yes	Yes	1+USART	Yes	4	
2	Yes	32	Yes	31	Yes	Yes	6	Yes	Yes	1+USART	Yes	1	
2	Yes	32	Yes	31	Yes	Yes	6	Yes	Yes	1+USART	Yes	2	
2	Yes	17	Yes	23	Yes	Yes	4	Yes	Yes	1+USI	USI	1	
2	Yes	17	Yes	32	Yes	Yes	4	Yes	Yes	1+USI	USI	1	
2	Yes	8	Yes	34	Yes	Yes	8	Yes	Yes	1	Yes	2	
2	Yes	2	Yes	18	Yes	Yes	3	Yes	Yes	1	Yes	1	
1	Yes	3	Yes	16	Yes	Yes	3	--	Yes	1	--	1	
2	Yes	3	Yes	20	Yes	Yes	4	--	Yes	1	Yes	1	
2	Yes	26	Yes	26	Yes	Yes	6	Yes	Yes	1+USART	Yes	1	

DevicesDevices	Flash (Kbytes)	EEPROM (Kbytes)	SRAM (Bytes)	Max I/O Pins	F.max (MHz)	Vcc (V)	10-bit A/D Channels	Analog Comparator	16-bit Timers
ATmega128	128	4	4096	53	16	2.7-5.5	8	Yes	2
ATmega1280	128	4	8192	86	16	1.8-5.5	16	Yes	4
ATmega1281	128	4	8192	54	16	1.8-5.5	8	Yes	4
ATmega1284P	128	4	16K	32	20	1.8 - 5.5	Yes	Yes	2
ATmega162	16	0.5	1024	35	16	1.8-5.5	--	Yes	2
ATmega164P	16	0.5	1024	32	20	1.8-5.5	8	Yes	1
ATmega164PA	16	0.5	1024	32	20	1.8-5.5	8	Yes	1
ATmega165P	16	0.5	1024	54	16	1.8-5.5	8	Yes	1
ATmega168	16	0.5	1024	23	20	1.8-5.5	6/8	Yes	1
ATmega168P	16	0.5	1024	23	20	1.8-5.5	8	Yes	1
ATmega16A	16	0.5	1024	32	16	2.7-5.5	8	Yes	1
ATmega2560	256	4	8192	86	16	1.8-5.5	16	Yes	4
ATmega2561	256	4	8192	54	16	1.8-5.5	8	Yes	4
ATmega324P	32	1	2048	32	20	1.8-5.5	8	Yes	1
ATmega324PA	32	1	2048	32	20	1.8-5.5	8	Yes	1
ATmega325	32	1	2048	54	16	1.8-5.5	8	Yes	1
ATmega3250	32	1	2048	69	16	1.8-5.5	8	Yes	1
ATmega3250P	32	1	2048	69	20	1.8-5.5	8	Yes	1
ATmega325P	32	1	2048	54	20	1.8-5.5	8	Yes	1
ATmega328P	32	1	2048	23	20	1.8-5.5	8	Yes	1
ATmega32A	32	1	2048	32	16	2.7-5.5	8	Yes	1
ATmega48PA	4	0.25	512	23	20	1.8-5.5	8	Yes	1
ATmega64	64	2	4096	54	16	2.7-5.5	8	Yes	2
ATmega640	64	4	8192	86	16	1.8-5.5	16	Yes	4
ATmega644	64	2	4096	32	20	1.8-5.5	8	Yes	1
ATmega644P	64	2	4096	32	20	1.8-5.5	8	Yes	1
ATmega645	64	2	4096	54	16	1.8-5.5	8	Yes	1
ATmega6450	64	2	4096	69	16	1.8-5.5	8	Yes	1
ATmega64A	64	2	4096	54	16	2.7-5.5	8	Yes	2
ATmega8	8	0.5	1024	23	16	2.7-5.5	6/8	Yes	1
ATmega8515	8	0.5	512	35	16	2.7-5.5	--	--	1
ATmega8535	8	0.5	512	32	16	2.7-5.5	8	Yes	1
ATmega88PA	8	0.5	1024	23	20	1.8-5.5	8	Yes	1

Tabela 3- Comparação entre os ATtiny (04/2009)

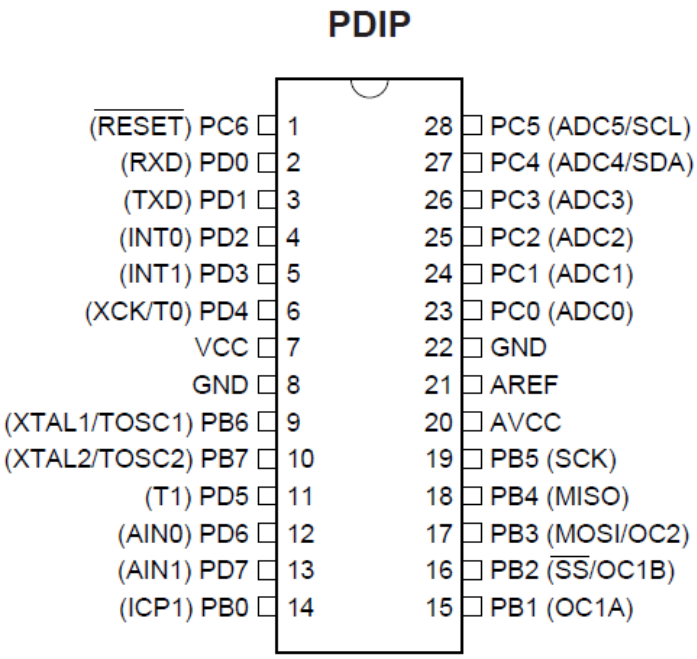
Devices Devices	Flash (Kbytes)	EEPROM (Kbytes)	SRAM (Bytes)	Max I/O Pins	F.max (MHz)	Vcc (V)	10-bit A/D Channels	Analog Comparator	16-bit Timers	8-bit Timer	Brown Out Detector	Ext Interrupts	Interrupts	ISP	On Chip Oscillator	PWM Channels	RTC	Self Program Memory
ATtiny12	1	0.0625	--	6	8	1.8-5.5	--	Yes	--	1	Yes	1	5	Yes	Yes	--	--	--
ATtiny13A	1	0.0625	64B + 32 reg	6	20	1.8-5.5	4	Yes	--	1	Yes	6	9	Yes	Yes	2	--	Yes
ATtiny2313	2	0.125	128	18	20	1.8-5.5	--	Yes	1	1	Yes	2	8	Yes	Yes	4	--	Yes
ATtiny24	2	0.125	128	12	20	1.8-5.5	8	Yes	1	1	Yes	12	17	Yes	Yes	4	--	Yes
ATtiny24A	2	0.125	128	12	20	1.8-5.5	8	Yes	1	1	Yes	12	17	Yes	Yes	4	--	Yes
ATtiny25	2	0.125	128	6	20	1.8-5.5	4	Yes	--	2	Yes	7	15	Yes	Yes	4	--	Yes
ATtiny261	2	0.125	128	16	20	1.8 - 5.5	11	Yes	1	2	Yes	2	19	Yes	Yes	2	--	Yes
ATtiny28L	2	--	32	11	4	1.8-5.5	--	Yes	--	1	--	2(+8)	5	--	Yes	--	--	--
ATtiny43U	4	0.0625	256	16	8	0.7 - 1.8	4	Y	--	2	Y	17	16	Y	--	4	--	--
ATtiny44A	4	0.25	256	12	20	1.8-5.5	8	Yes	1	1	Yes	12	17	Yes	Yes	4	--	Yes
ATtiny45	4	0.25	256	6	20	1.8-5.5	4	Yes	--	2	Yes	7	15	Yes	Yes	4	--	Yes
ATtiny461	4	0.25	256	16	20	1.8 - 5.5	11	Yes	1	2	Yes	2	19	Yes	Yes	2	--	Yes
ATtiny48	4	0.0625	256	24/28	12	1.8 - 5.5	6/8	Yes	1	1	Yes	30	20	Yes	Yes	1	--	Yes
ATtiny84	8	0.5	512	12	20	1.8-5.5	8	Yes	1	1	Yes	12	17	Yes	Yes	4	--	Yes
ATtiny85	8	0.5	512	6	20	1.8-5.5	4	Yes	--	2	Yes	7	15	Yes	Yes	4	--	Yes
ATtiny861	8	0.5	512	16	20	1.8 - 5.5	11	Yes	1	2	Yes	2	19	Yes	Yes	2	--	Yes
ATtiny88	8	0.0625	512	24/28	12	1.8 - 5.5	6/8	Yes	1	1	Yes	30	20	Yes	Yes	1	--	Yes

3. ATmega8

O ATmega 8 apresenta a maioria das características da família AVR.É um microcontrolador de 8-bits com baixo consumo, de tecnologia CMOS e arquitetura **RISC** (*Reduced Instruction Set Computer*), cuja pinagem é mostrada na figura 2. Apresenta a capacidade de executar uma instrução por ciclo de *clock* devido à conexão direta de seus 32 registradores gerais com a unidade lógica aritmética e uma frequência de funcionamento na faixa de **0 a 20MHz**. Além disso, apesar de ser RISC, possui um grande número de instruções, permitindo uma otimização do código de alto nível em linguagem C.

Ao programar este microcontrolador, devido as similaridades entre as famílias, os conceitos de programação são válidos para qualquer outro da família AVR. As pequenas mudanças, de hardware e software podem ser consultadas em seus respectivos *Datasheets.*

Figura 2 - Pinagem do Microcontrolador AVR ATmega8.



Fonte: Atmel (2004)

.

3.1.Características do ATmega8

- Apresenta baixa potência e alto desempenho, com arquitetura RISC avançada;
- 130 instruções, a maior parte executada em um único ciclo de *clock*;
- 32×8 registradores de trabalho de propósito geral;
- Operação de até 16 MIPS (milhões de instruções por segundo) a 16 MHz ;
- Multiplicação por hardware em 2 ciclos de *clock*;
- 8 kbytes de memória de programa FLASH de auto programação *In-System* (16K, 32K, 64K, 128K nos respectivos ATmega16, ATmega32, ATmega64 e ATmega128).
- 512 bytes de memória EEPROM;
- 1 kbyte de memória SRAM;
- Ciclos de escrita e apagamento: memória FLASH 10.000 vezes, EEPROM 100.000 vezes;
- Bits de bloqueio para proteção do software;

Periféricos embutidos:

- 23 entradas e saídas (I/Os) programáveis;
- 2 Temporizadores/Contadores de 8 bits com *Prescaler* separado, 1 modo de comparação;
- Um Temporizador/Contador de 16 bits com *Prescaler* (divisor)separado, modo de comparação e captura;
- contador de tempo real (com cristal externo de 32.768 Hz conta precisamente 1s);
- 3 canais PWM;
- 8 canais A/D com precisão de 10 bits na versão TQFP, 6 canais na versão PDIP;
- Interface serial para dois fios orientada a byte (TWI), compatível com o protocolo I2C;
- Interface serial USART (*Universal Synchronous Asynchronous Receiver Transmitter);*
- Interface serial SPI *Master/Slave;*
- *Watchdog Timer* com oscilador interno separado;
- Um comparador analógico.

Características especiais:

- *Power-on Reset* e detecção *Brown-out* programável;
- Oscilador interno RC (não há a necessidade do uso de cristal externo ou de outra fonte de *clock*);
- Fontes de interrupções internas e externas;
- 5 modos de *Sleep*: *Idle*, Redução de ruído do A/D, *Power-down*, *Power-save* e *Standby;*

Tensão de operação: 2,7-5,5 V (ATmega8L), 4,5-5,5 V (ATmega8).

Outra característica que permite a maximização do desempenho nestes microcontroladores é o paralelismo - a **arquitetura Harvard e a técnica do *pipeline*.**

A técnica em "*pipeline*" sobrepõe busca e execução, tornando a execução de instruções possível de se realizar em um único ciclo de máquina. Qualquer instrução de desvio (tais como GOTO, CALL, ou escrever no PC) leva dois ciclos de máquina.

A **memória de programa** é executada em *pipeline* de dois estágios- enquanto uma instrução está sendo executada, a próxima instrução é previamente buscada da memória de programa. Esse conceito habilita a execução de instruções em todo ciclo de *clock.* Esta memória é do tipo FLASH, e com as instruções relativas de "jump" e "call", todo o espaço de endereçamento de 8k é diretamente acessado. A maioria das instruções AVR tem um único formato de palavra de 16-bits. Todo endereço da memória de programa contém uma instrução de 32-bits.

Com a arquitetura RISC são necessários menos instruções que, por exemplo, a tradicional arquitetura CISC, permitindo que os sistemas nela baseados, possam rodar mais rápidos porque o processador tem funções limitadas, em benefício de seu desempenho.

3.2. Memórias EEPROM e FLASH

A EEPROM presente no AVR possui 512 bytes e está ligada ao barramento de dados de 8-bits interno permitindo que possa ser escrita diretamente sobre o microcontrolador durante o processo de gravação ou que o próprio microcontrolador escreva os dados nas posições desta memória. O tempo de acesso de gravação é em média de 2,5 a 4ms, dependendo da tensão a qual é submetida.

A memória FLASH ou FLASH ROM é uma EEPROM que utiliza baixas tensões de apagamento, e em um tempo reduzido. O apagamento da memória FLASH é extremamente rápido e, ao contrário da EEPROM, não é possível reprogramar apenas um único endereço, isto é, quando a memória é apagada,

todos os seus endereços são zerados. O AVR apresenta 8k bytes de memória FLASH Programável on-chip para armazenamento de programas.

3.3. Memória SRAM

As informações existentes em uma memória RAM (*Random Access Memory*) caso não sejam salvas fisicamente, são perdidas ao se desligar o computador. Os 608 endereços baixos pertencem às alocações da Memória de Dados do *Register File*, à Memória de I/O e a SRAM interna, desses os primeiros 96 endereços representam o Register File + Memória de I/O, e os 512 endereços restantes são da SRAM interna.

3.4. Registradores do AVR

Uma característica básica é a presença de um grupo de registradores internos. A arquitetura AVR, apresenta 32 registradores de 8 bits, que podem ser manipulados para leitura / escrita, como 16 palavras de 16 bits Há também os registradores de I/O, os quais são em número de 64 e podem ser endereçados diretamente em instruções de apenas um ciclo de *clock*.

3.5. Registradores e Comandos I/O

Os microcontroladores AVR possuem múltiplos registradores, mas a maior parte deles é utilizada para entrada/saída. Outros possuem funções especiais e alguns para acesso de dados na memória do microcontrolador.

Além disso, há registradores exclusivos em alguns modelos de AVR. Os nomes dos registradores são definidos nos *header files* para os tipos apropriados de AVR, por exemplo, *#include<avr/io.h>* incluído na programação.

Os registradores de I/O são bastante utilizados para as diversas funções de controle no AVR. Eles controlam os acessos às portas e às interfaces com o microcontrolador. Deve-se ressaltar que há diferenciação entre os registradores de 8-bits e os de 16-bits.

3.6. Acesso às portas do AVR

Há três endereços de memória (registradores) que são alocados para cada uma das portas do AVR: *PORTx (Data Register), DDRx (Data Direction Register) e PINx (Port Inputs Pins)*. Este último só é utilizado para leitura enquanto que os outros dois podem ser utilizados tanto para leitura como escrita. Estes estão detalhados na figura 3.

Figura 3 - Registradores para direcionar as portas do AVR

DDRx	Registradores de direção de dados para a porta x, onde x corresponde às portas A, B, C, etc. Caso esteja ativado, a porta é utilizada como saída. Caso contrário, são utilizados como entradas.
PINx	Refere-se à condição da porta quando definida como entrada, ou seja, se for igual a 1, temos que a porta está em nível lógico alto. Caso contrário, em nível lógico baixo.
PORTx	Quando a porta é definida como saída, PORTx é utilizada para obter o valor no pino. Quando a porta é definida como entrada, PORTx é utilizada para ativar ou desativar o pull-up dos resistores.

Fonte: Atmel (2004).

```
#include <avr/io.h>
#include <util/delay.h>
int main (void)
{
	PORTD = 0x00;/* ativa o resistor pull-up nos pinos PIND*/
	DDRD = 0x0C;/* seta os pinos PIND 2 e 3 como saida */
	unsigned int d;
	d = 10000;
	for (;;)//loop infinito
	{
			PORTD = 0x04;
			_delay_ms(d);
			PORTD = 0x08;
			_delay_ms(d);
	}
}
```

O núcleo AVR combina um conjunto de instruções com 32 registradores de trabalho, os quais estão diretamente conectados à Unidade Lógica e Aritmética (ALU), permitindo que dois registradores independentes sejam

acessados com uma simples instrução em um único ciclo de clock. Seis dos 32 registradores podem ser usados como registradores de endereçamento indireto de 16 bits (ponteiro para o acesso de dados).

A CPU garante a correta execução do programa, sendo capaz de acessar as memórias, executar cálculos, controlar os periféricos e tratar interrupções . Na Figura 5 mostra-se um diagrama de blocos da CPU do AVR. Seguindo a arquitetura Harvard percebe-se apresenta o barramento de dados para programa e para dados . O paralelismo permite que uma instrução seja executada enquanto a próxima é buscada na memória de programa, produzindo a execução de uma instrução por ciclo de clock.

Figura 5– Diagrama em Blocos da CPU do ATmega8

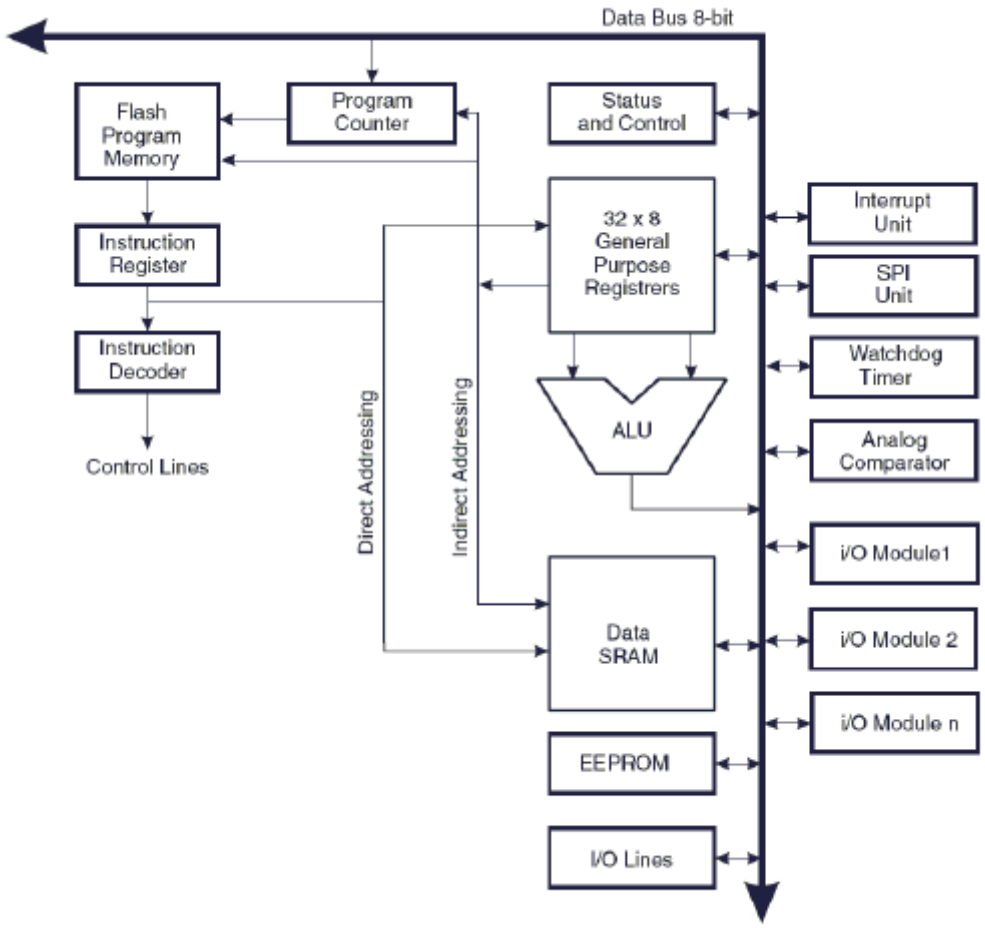
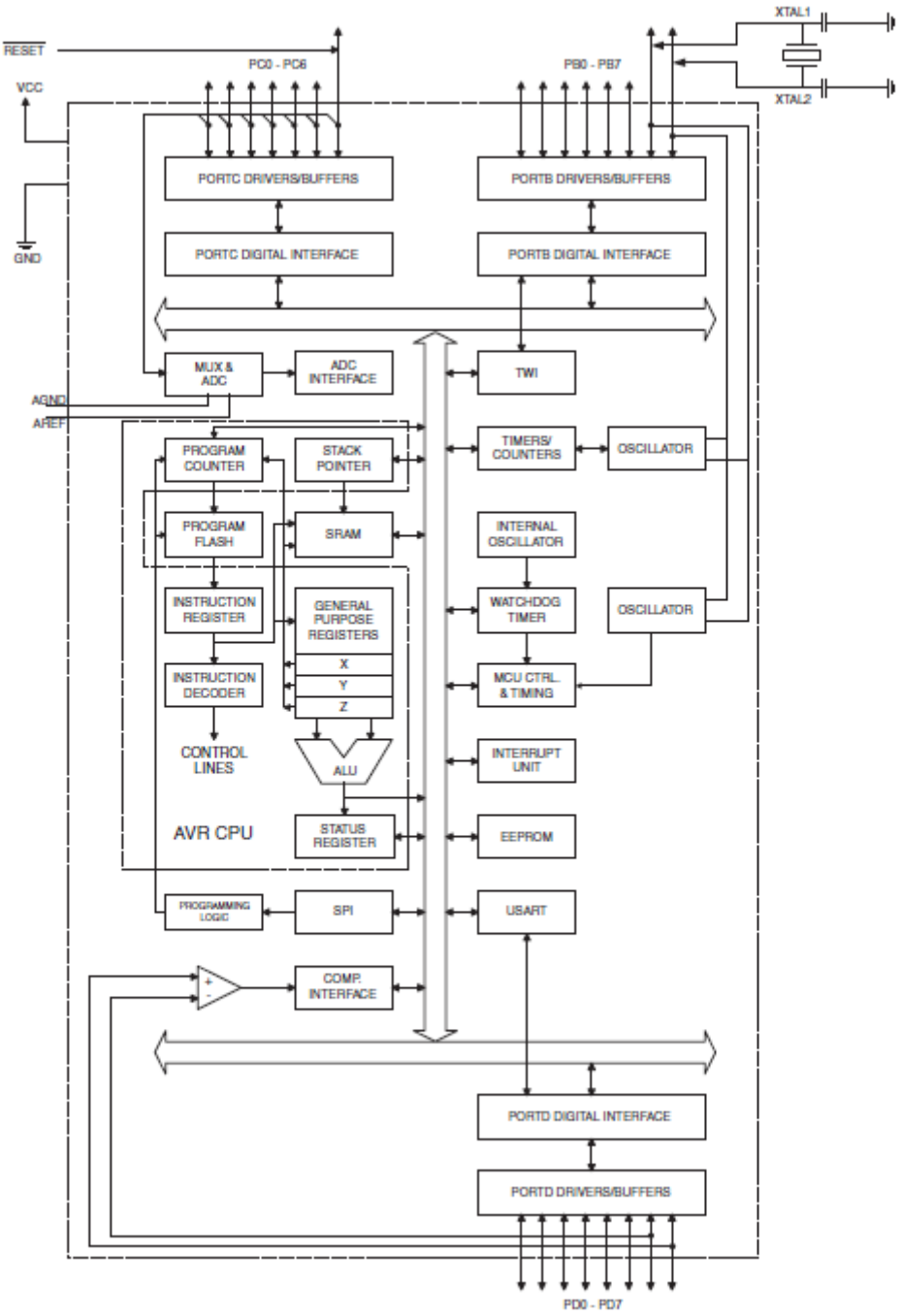


Figura 6– Diagrama em Blocos do ATmega8

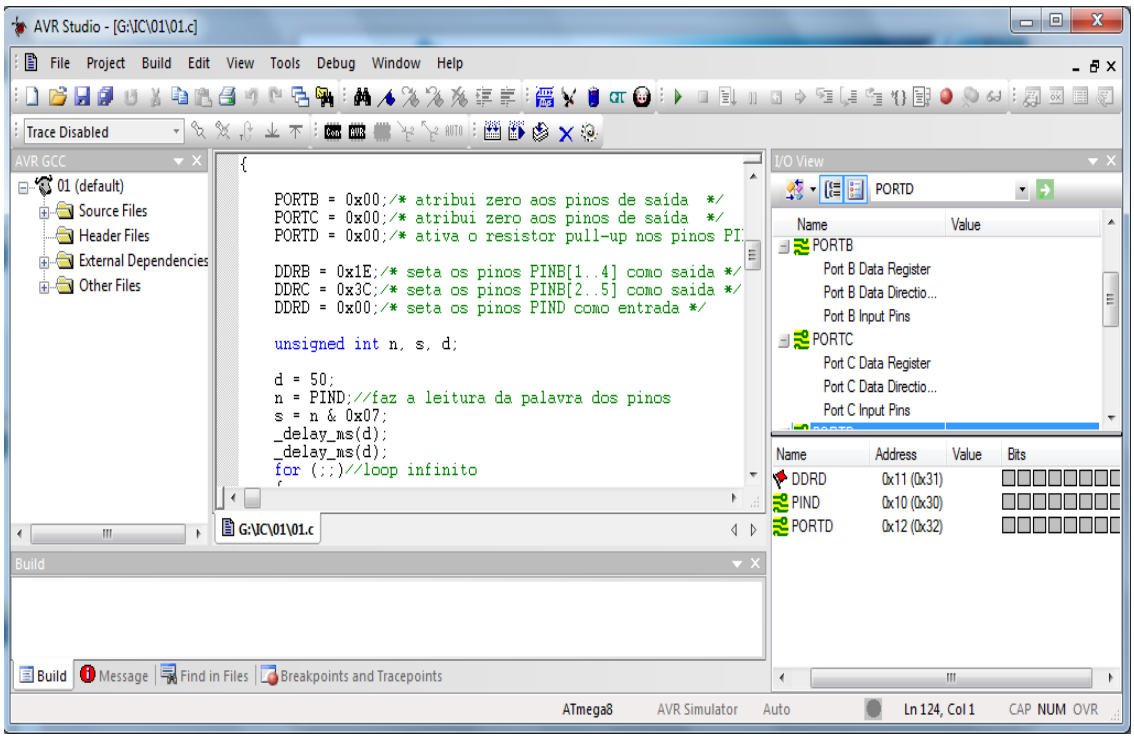


4 . Gravação do Programa

Na programação do microcontrolador ATmega8 da família AVR utiliza-se dois programas, são eles AVR Studio 4 e o PonyProg 2000.

A programação pode ser feita em linguagem C (...). Utiliza o software AVR Studio 4 (Figura 7), onde é possível fazer a compilação e depuração do de um programa Esta ferramenta é útil para a detecção de erros no programa. Ao compilar o programa, o software cria um arquivo em hexadecimal, que é salvo na pasta padrão do projeto e este será utilizado no software PonyProg para a gravação do microcontrolador.

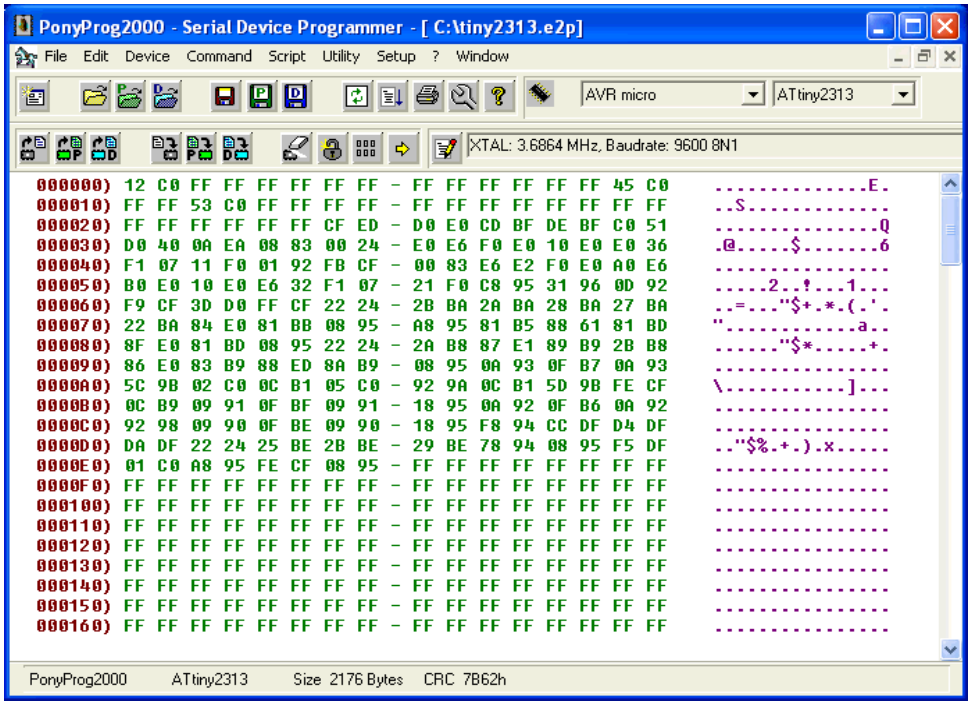
Figura 7 – Tela do AVR Studio



Fonte: Própria do autor.

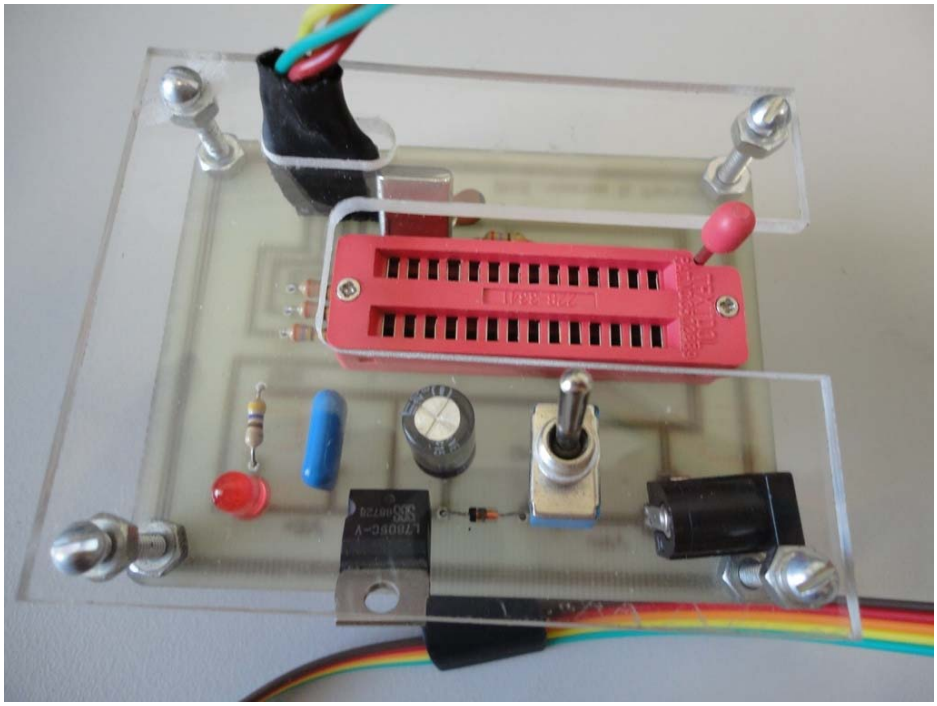
Para a efetiva “queima” do microcontrolador é necessário o software PonyProg (Figura 8) e um circuito de gravação, o gravador, Figura 9. O programa é salvo na memória FLASH do microcontrolador, opção de escolha presente no software.

Figura 8- Interface do programa PonyProg.



Fonte: Própria do autor

Figura 92 - Gravador do AVR ATmega8



Fonte: Própria do autor

A gravação dos dados e controles, originados de um computador e a comunicação com o microcontrolador é feita por meio de um *flat cable* ligados à porta serial do PC através do conector DB-9, através dos pinos MISO, MOSI , RESET, SCLK, presentes no microcontrolador, conforme especificados na tabela 4 .

Tabela 4 - Ligações SPI(*Serial Peripheral Interface Bus*) ou SSI (Synchronous Serial Interface) dos microcontroladores AVR

Nome da conexão		**Função**
MISO	Serial Out	Saída de dados
MOSI	Serial In	Entrada de dados
SCLK	Serial Clock	*Clock* de sincronismo
RESET		Controle do Reset

Fonte: Própria do autor

5. Criando um projeto no AVR Studio 4

5.1 Em assembly

1) Project-> new Project (colocar o nome , a pasta e o simulador

No AVR Simulator -> escolhe-se ATmega8

Digitar o projeto (não esquecer o include)

Projeto exemplo:

```
Include " 8515def.inc"
deftemp=16
Reset:
        ldi temp,0x0FF
        out DDRB,temp
        ldi temp, 0x01
Loop:
       out portb,temp
       rol temp
       rjmp Loop
```

2) Simular -> (debug)

Clicar em Build-> build -. (deve aparece na tela o arquivo anterior)

3) Simular –AVR Simulator para testar o projeto. Verificar os I/O view

Obs: degug (seleciona o chip e a plataforma –AVR Simulator .

4) Selecionar Debug\Start Debugging

6. Roteiro de Gravação com o ATmel-Pony Prog 2000.

Pony Program 2000.

1) Abrir o arq. .hex (teste) do aTmega 8 . Em seguida (build) deve-se compilar .
2) Por default-> tem o . hex
3) Levar para o Pony program 2000

Dicas para programação em C

Leitura de um registrador de I/O

Para leitura, podem-se acessar os registradores de I/O como simples variáveis. Em códigos fonte, o acesso de leitura é feito pela função *inp()*. Em versões mais atuais do compilador o acesso pode ser feito de forma direta e a função *inp()* não é mais necessária, como observa-se a seguir:

```
#include <arv/io.h>
Uint8_t foo;
...
        int main(void)
{
// Copia o status dos pinos da porta B na variavél foo
        foo = PINB
...
}
```

Leitura de um bit

A biblioteca do AVR possui funções para averiguação de um bit individual do registrador:

bit_is_set (<Register>, <Bitnumber>):

A função examina se um bit foi ativado, retornando um valor diferente de 0.

bit_is_clear (<Register>, <Bitnumber>):

A função examina se o bit foi desativado, retornando um valor diferente de 0.

Escrevendo em um registrador I/O

Para escrita, podem-se acessar os registradores de I/O como simples variáveis. Em códigos fonte o acesso a escrita é feito pela função *outp()*. Em versões mais atuais do compilador o acesso pode ser feito de forma direta e a função *outp()* não é mais necessária, como observa-se a seguir,

```
#include <arv/io.h>
...
int main(void)
{
DDRA = 0xff; // Define todos os pintos como saída
PORTA = 0x03 // Define o nível lógico de cada saída
.....
}
```