UFMG
UNIVERSIDADE FEDERAL DE MINAS GERAIS

Introdução à Robótica
Handy Board & Interactive C

Prof. Douglas G. Macharet
douglas.macharet@dcc.ufmg.br

VERLab Visão e Robótica
DCC DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

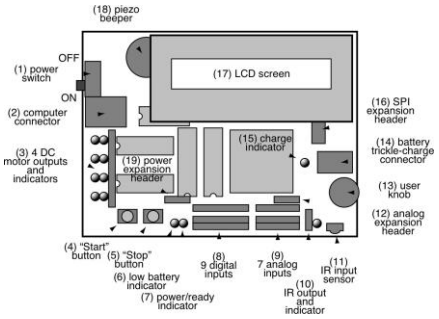
Handy Board & Interactive C

Leiam os manuais!
Sejam cuidadosos!

Handy Board
Especificações

- Microprocessador Motorola 68HC11 (8 bits)
- Clock de 2 MHz
- 32 Kb de RAM
- Saídas
  - 4 motores DC (9v, 1A)
- Entradas
  - 7 sensores analógicos
  - 9 sensores digitais
- 2 botões programáveis (*start*, *stop*), *knob*, *beeper*

Handy Board



Interactive C
Especificações

- Compilador de linguagem C
  - Randy Sargent
  - Desenvolvido para aplicações robóticas
  - Compilada em um pseudo-código (não nativo)
- Interatividade
  - Linha de comando
  - Digitar expressões e chamadas de funções

Interactive C
Especificações

- Estabilidade
  - Lança exceção ao invés de dar *crash* no sistema
- Multi-tarefa
  - Até 12 processos (funções) simultaneamente

Interactive C
Tipos de dados

- *Case sensitive*
- Tipos suportados
  - **int**, 16-bits (-32768 a +32767)
  - **long**, 32-bits (-2147483648 a +2147483647)
  - **float**, 32-bits (10^{-38} a 10^{38})
  - **char**, 8-bits

Interactive C
Tipos de dados

- Variáveis Locais
  - Declaradas no contexto de uma função
  - Inicializadas quando a função é executada
- Variáveis Globais
  - Declaradas fora de uma função específica
  - Inicialização
    - Um novo arquivo é copiado para a HB
    - A função *main()* é executada
    - Ocorre um *reset* no *hardware*

Interactive C
Tipos de dados

- Variáveis Globais Persistentes
  - Não inicializadas (valor inicial arbitrário)
  - Mantém o estado quando
    - A HB é desligada/ligada
    - A função *main()* é executada
    - Ocorre um *reset* no *hardware*

Interactive C
Tipos de dados

- Variáveis Globais Persistentes
  - Devem ser declaradas primeiramente
  - Principais exemplos de uso
    - Calibração e configuração
    - Aprendizado

Interactive C
Tipos de dados

```
persistent int i;    /* persistente global , ? */
float z = 3.0;       /* global, 3.0 */

void main()
{
   int x;            /* local , 0 */
   int y = 7;        /* local , 7 */
}
```


Interactive C
Função *main()*

- Automaticamente executada ao ligar a HB
- Ligando sem executar a função *main()*
  - Manter pressionado o botão *start* ao ligar a HB

Interactive C
Controle de fluxo

- **If-Else**
- **While**
- **For**
- **Break**
  - Sai de um **while** ou **for**
- NÃO suporta os comandos *case* e *switch*

Interactive C
Vetores

```
int retrieve_element(int index, int array[])
{
   return array[index];
}

void main ( )
{
   int foo[10];
   int array[] = {0, 4, 5, -8, 17, 301};
   char string[] = "Hello there";
   retrieve_element(3, array);
}
```


Interactive C
Ponteiros

```
void avg_sensor(int port, int result)
{
   int sum = 0;
   int i;
   for (i = 0, i < 10, i ++)
      sum += analog(port);

   *result = sum/10;
}

void main()
{
   int result;
   avg_sensor(0, &result);
}
```

*Aritmética de ponteiros não é suportada!

Interactive C
Motores

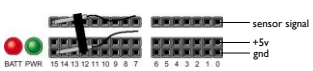
- Utiliza Pulse Width Modulation (PWM)
- Funções
  - **fd(int m)**: Aciona o motor **m** (p=100)
  - **bk(int m)**: Aciona o motor **m** na direção oposta (p=-100)
  - **motor(int m, int p)**: Aciona o motor **m** com potência **p**
  - **off(int m)**: Desliga o motor **m**
  - **alloff()** ou **ao()**: Desliga todos os motores simultaneamente
- Valores válidos
  - Motor: {0, 1, 2, 3}
  - Potência: [-100, ..., 100]

Interactive C
Motores

```
void main()
{
   fd(0);
   fd(1);
   motor(2, 50);
   sleep(1.0);
   off(2);
   sleep(1.0);
   ao();
}
```


Interactive C
Sensores

- Analógicos
  - Retorna um valor no intervalo [0, ..., 255]
  - **int analog(int p)**: Valor do sensor conectado à porta **p**
  - Portas 0-6
- Digitais
  - Retorna um valor 0/1
  - **int digital(int p)**: Valor do sensor conectado à porta **p**
  - Portas de 7-15



Interactive C
Multi-tarefa

- Todo processo inicializado
  - Executa por um determinado número de *ticks*
  - Possui sua própria pilha de execução
- Funções
  - **int start_process(function(...), [ticks], [stack-size])**
    - Valor padrão de *ticks* é 5 milisegundos
    - Valor padrão de *stack* é 256 bytes
  - **int kill process(int pid)**
  - **kill_all()**
- Processos se comunicam através de variáveis globais

Interactive C
Multi-tarefa

```
void check_sensor(int n)
{
   while(1)
      printf("Sensor %d is %d\n", n, digital(n));
}

void main()
{
   int pid;
   pid=start_process(check_sensor(2));
   sleep(1.0);
   kill_process(pid);
}
```


Interactive C
Escrita de mensagens no LCD

- Utilizar o comando **printf()**

```
printf(format-string, [arg-1] , ... , [arg-N] )
```

- Caracteres de formatação de mensagens

%d	Type: int Description: decimal number
%x	Type: int Description: hexadecimal number
%b	Type: int Description: low byte as binary number
%c	Type: int Description: low byte as ASCII character
%f	Type: float Description: floating point number
%s	Type: char array Description: char array (string)

Interactive C
Macros

```
#define RIGHT_MOTOR 0
#define LEFT_MOTOR  1

#define GO_RIGHT(power)  (motor(RIGHT_MOTOR,(power)))
#define GO_LEFT(power)   (motor(LEFT_MOTOR,(power)))

#define GO(left, right) {GO_LEFT(left); GO_RIGHT(right)}

void main()
{
   GO(25, 25);
}
```


Interactive C
Compilação condicional

```
#define DEBUG

void main()
{
   #ifdef DEBUG
   printf("Mensagem de Debug!");
   #endif
}
```


Interactive C
Demais funções disponíveis

- Botões
  - **int stop_button()**: Retorna o valor do botão *stop* (0/1)
  - **int start_button()**: Retorna o valor do botão *start* (0/1)
  - **stop_press()**: Espera o botão *stop* ser pressionado e liberado
  - **start_press()**: Espera o botão *start* ser pressionado e liberado
  - **int knob()**: Retorna o valor da posição do *knob* ([0, ..., 255])

```
while (!stop_button( ));
while(stop_button( ));
beep ( ) ;
```

Interactive C
Demais funções disponíveis

- Tempo
  - **sleep(float s)**: Espera por **s** segundos
  - **msleep(long ms)**: Espera por **ms** milisegundos
- Tom
  - **beep()**: Produz um tom de 500 Hz por 0,3 segundos
  - **tone(float f, float t)**:Tom de **f** Hz por **t** segundos

Interactive C
Arquivos

- Um programa pode ser definido em vários arquivos
- Carregar os arquivos com o comando de console **load**
  - É possível informar mais de um arquivo como parâmetro
- Definir um arquivo **.lis** relacionando todos os arquivos