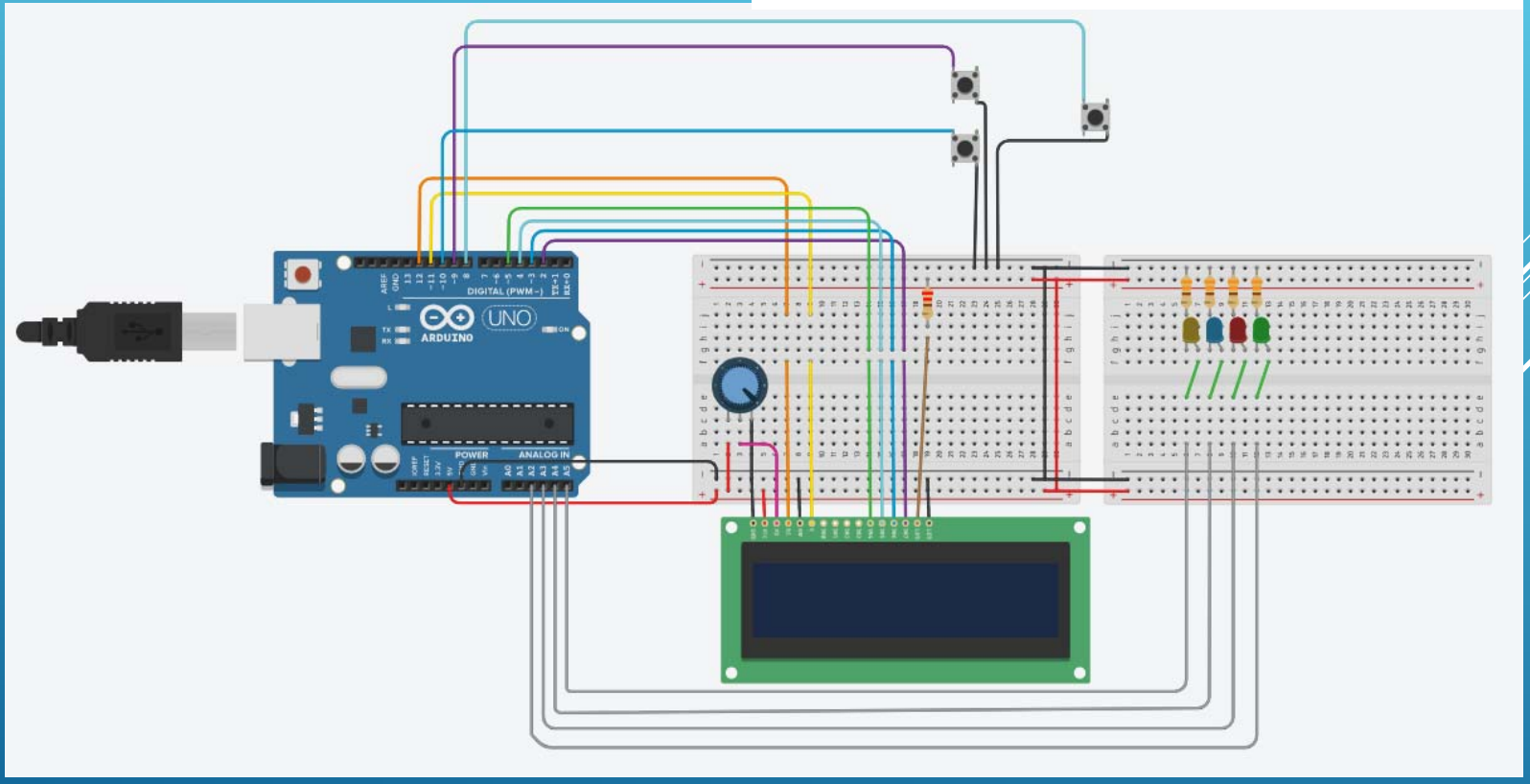
SISTEMAS EMBARCADOS APLICADOS I – MENU E FUNÇÕES

Prof. Dr. Dalton Vidor

MICROCONTROLADORES:

- Menu,
- IHM com poucas teclas e
- Evitar o repique(debounce) das teclas.

MENU e EVITAR DEBOUNCING



MICROCONTROLADORES:

- Menu,
- lHM com poucas teclas e
- Evitar o repique(debounce) das teclas.

- Bibliotecas, variáveis e definições dos pinos.

```
// incluir a biblioteca para display operar
#include <LiquidCrystal.h>
//definição dos pinos e nomes associados às teclas
#define botao_p_baixo        10
#define botao_p_cima         9
#define botao_seleciona      8

//definição dos pinos e nomes associados aos leds
#define L_AM      A5
#define L_AZ      A4
#define L_VM      A3
#define L_VD      A2

int menu = 1;              //Variável para selecção da página.
int LedVerdePisca = 0;     //Variável para liberar o led verde para piscar


// Definir pinos ligados ao display conforme LiquidCrystal(rs, enable, d4, d5,
d6, d7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // para a placa lcd keypad shield
```

MICROCONTROLADORES:

- Menu,
- IHM com poucas teclas e
- Evitar o repique(debounce) das teclas.

- SETUP e atualiza display (ou inicializa)
- Observar pinos com PULLUP para não gastar com resistors.

```
void setup() {
  // define a velocidade da comunicação serial
  Serial.begin(9600);
  // configurar o display com colunas (16) e linhas (2)
  lcd.begin(16, 2);
  // inicailizar os pinos digitais das teclas com PullUp
  pinMode(botao_p_baixo, INPUT_PULLUP);
  pinMode(botao_p_cima, INPUT_PULLUP);
  pinMode(botao_seleciona, INPUT_PULLUP);
  // inicailizar os pinos dos LEDs com função digital
  pinMode(L_AM, OUTPUT);
  pinMode(L_AZ, OUTPUT);
  pinMode(L_VM, OUTPUT);
  pinMode(L_VD, OUTPUT);
  // nos pinos do display a biblioteca se encarrega
de acertar
  AtualizaMenu();                    // chama função que
atualiza o Menu
}
```

MICROCONTROLADORES:

- LOOP infinito que testa as teclas e altera índice do menu.
- Espera soltar a Tecla para evitar o repique.
- Atualiza display
- E executa o pisca-pisca que é dependente de variável e é contínuo

```
void loop() {
 if (!digitalRead(botao_p_baixo)){ // testa se os botões vão para nível zero!!!
   menu++;                          // incrementa tela de menu
   AtualizaMenu();                  // chama função que atualiza o Menu
   delay(100);                      // aguarda um tempo para soltar a tecla
   while (!digitalRead(botao_p_baixo)); // espera soltar a tecla
 }
 if (!digitalRead(botao_p_cima)){ // testa se os botões vão para nível zero!!!
   menu--;                          // decrementa tela de menu
   if(menu>5) menu=5;
   AtualizaMenu();                  // chama função que atualiza o Menu
   delay(100);                      // aguarda um tempo para soltar a tecla
   while(!digitalRead(botao_p_cima)); // espera soltar a tecla
 }
 if (!digitalRead(botao_seleciona)){ // testa se os botões vão para nível zero!!!
   ExecutaAcao();                   // chama função que executa ação
   AtualizaMenu();                  // chama função que atualiza o Menu
   delay(100);                      // aguarda um tempo para soltar a tecla
   while (!digitalRead(botao_seleciona)); // espera soltar a tecla
 }
 if(LedVerdePisca){
   digitalWrite(L_VD, !digitalRead(L_VD));
   delay(100);
 }
}
```

MICROCONTROLADORES:

- FUNÇÕES:
  - 1) ATUALIZA MENU

```
void AtualizaMenu() {
 switch (menu) {
   case 0:                                   // se a seleção é menor do que a
primeira tela
     menu = 1;                               // então seleciona a primeira tela
     break;
   case 1:
     lcd.clear();
     lcd.print("1.LED AMARELO");
     lcd.setCursor(0, 1);
     lcd.print("Selec->muda");
     break;
   case 2:
     lcd.clear();
     lcd.print("2.LED AZUL");
     lcd.setCursor(0, 1);
     lcd.print("Selec->submenu");
     break;
   case 3:
     lcd.clear();
     lcd.print("3.LED VERMELHO");
     lcd.setCursor(0, 1);
     lcd.print("Selec->submenu");
     break;
   case 4:
     lcd.clear();
     lcd.print("4.LED VERDE");
     lcd.setCursor(0, 1);
     lcd.print("Selec->muda");
     break;
   case 5:
     menu = 4;
     break;
 }
}
```

MICROCONTROLADORES:

- FUNÇÕES:
  - 2) CHAMA A FUNÇÃO - AÇÃO
  - Conforme índice do menu

```
void ExecutaAcao() {
 switch (menu) {
   case 1:
    acao1();
    break;
   case 2:
    acao2();
    break;
   case 3:
    acao3();
    break;
   case 4:
    acao4();
    break;
   case 21:
    acao21();
    break;
   case 31:
    acao31();
    break;
 }
}
```

MICROCONTROLADORES:

- FUNÇÕES:
  - 3) EXECUTA AÇÃO
  - Funções sem a passage de informações ou retorno específico.
  - Apenas a variável global LedVerdePisca é que será alterada.
  - Esta variável e a variável menu efetivamente retornam um valores por serem variáveis globais (observada em todo o Código)

```
void acao1() {
 lcd.clear();
 lcd.print("Executando1");
 digitalWrite(L_AM, !digitalRead(L_AM));
 delay(1500);
}
void acao2() {
 lcd.clear();
 lcd.print("Executando2");
 menu=21;                    // seleciona submenu
 lcd.setCursor(0, 1);
 lcd.print("Vai p submenu21");
 delay(1500);
}
void acao3() {
 lcd.clear();
 lcd.print("Executando3");
 menu=31;                    // seleciona submenu
 lcd.setCursor(0, 1);
 lcd.print("Vai p submenu31");
 delay(1500);
}
void acao4() {
 lcd.clear();
 lcd.print("Executando4");
 if(LedVerdePisca){
  LedVerdePisca=0;
 } else{
  LedVerdePisca=1;
 }
 delay(1500);
}
```

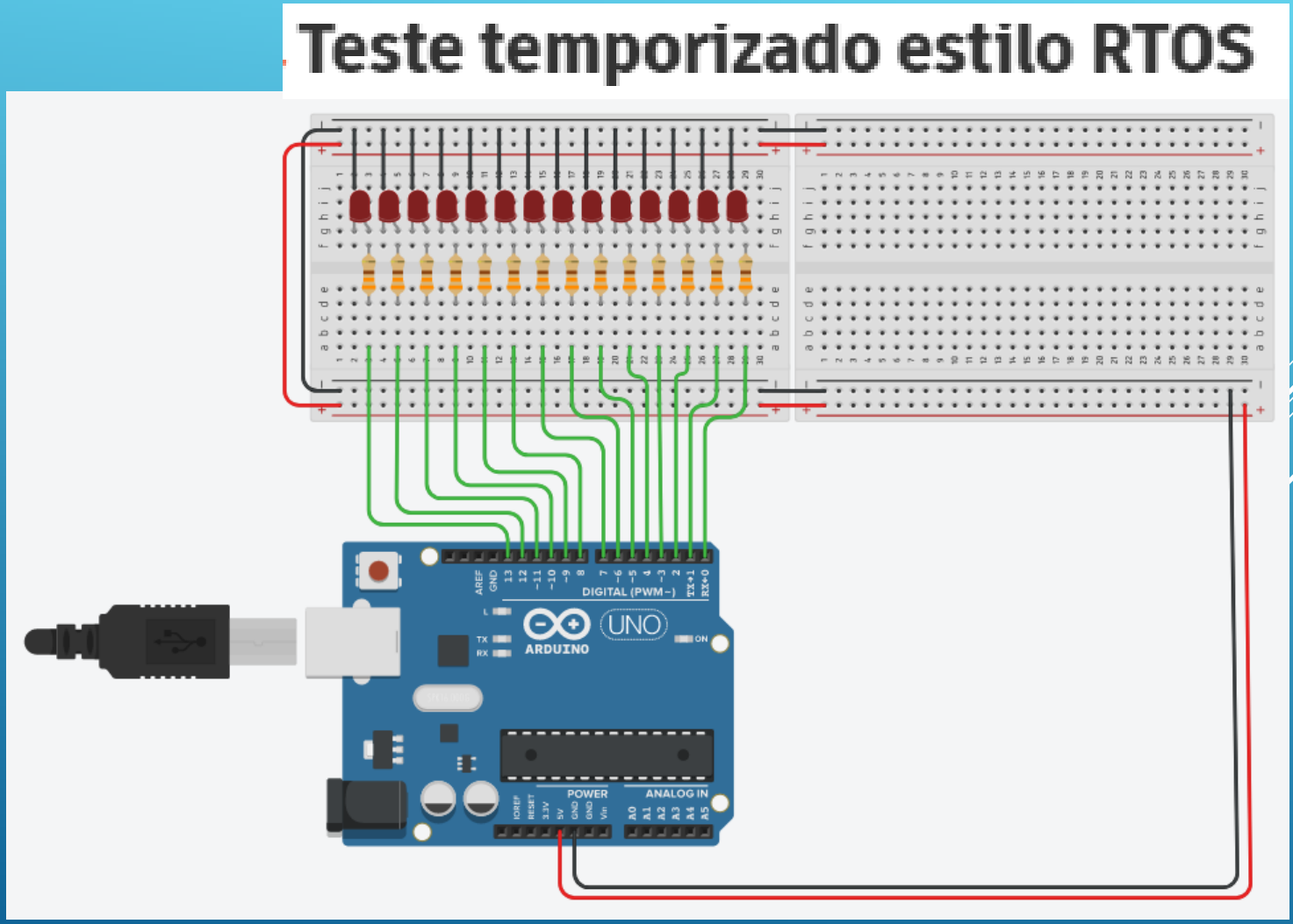
MICROCONTROLADORES:

- FUNÇÕES:
  - 3) EXECUTA AÇÃO
  - Observar os números utilzados e o incremento ou não da variável menu.

```
void acao21() {
  lcd.clear();
  lcd.print("Executando21");
  digitalWrite(L_AZ, !digitalRead(L_AZ));
  delay(1500);
}
void acao31() {
  lcd.clear();
  lcd.print("Executando31");
  digitalWrite(L_VM, !digitalRead(L_VM));
  delay(1500);
}
```

MICROCONTROLADORES:

- FUNÇÕES EXECUTADAS EM TEMPOS DISTINTOS
- Execução parecida com o RTOS (Real time Operational System)



MICROCONTROLADORES:

- Loop FOR para inicializar vários pinos ou executar funções em lotes...

- Variável para a função MILLIS unsigned long

```
unsigned long UltimoMillis;
int i, Tempo100, Tempo500,Tempo1000;


void setup() {
  for (i=0; i<14; i++){
    pinMode(i, OUTPUT);     // configura todos os pinos no loop
  }
  //pinMode(button, INPUT_PULLUP);
  UltimoMillis = millis();
}
```

MICROCONTROLADORES:

- Loop para coisas que rodam rápido e conta o tempo para 5 milisegundos...

- Função para coisas de 5 milisegundos (que rodam em) e conta tempo para 100 milisegundos...

```
void loop() {

  if ((millis() - UltimoMillis) >= 5) {
    UltimoMillis = millis();
    Coisas_5_mili();
  }
  digitalWrite(0, !digitalRead(0));
  digitalWrite(1, !digitalRead(1));
}


void Coisas_5_mili(){
    digitalWrite(2, !digitalRead(2));
    digitalWrite(3, !digitalRead(3));
    if (Tempo100<20) {
      Tempo100++;
    } else{
      Coisas_100_mili();
      Tempo100=0;
    }
}
```

MICROCONTROLADORES:

- Função para coisas de 100 milisegundos (que rodam em) e conta tempo para 500 milisegundos...
- Função para coisas de 500 milisegundos (que rodam em) e conta tempo para 1000 milisegundos...
- Função para coisas de 1000 milisegundos (neste caso sem nada)

```
void Coisas_100_mili(){
   digitalWrite(4, !digitalRead(4));
   digitalWrite(5, !digitalRead(5));
   if (Tempo1000<10) {
    Tempo1000++;
   } else{
    Coisas_500_mili();
    Tempo1000=0;
   }
}

void Coisas_500_mili(){
   digitalWrite(6, !digitalRead(6));
   digitalWrite(7, !digitalRead(7));
   if (Tempo500<5) {
    Tempo500++;
   } else{
    Coisas_1000_mili();
    Tempo500=0;
   }
}

void Coisas_1000_mili(){

}
```

MICROCONTROLADORES:

- Leituras de teclas – tempos de 100 a 500 milisegundos
- Atualização de display – 500 milisegundos ou 1 Segundo (exepcionalmente mais)
- Leituras térmicas de ambientes – 500 milisegundos ou 1 Segundo
- Leitura de grandezas da rede para calcular potência, valor eficaz ou outros valores detalhados – 100 microsegundos ou menos a integrar em um ciclo.
- Leituras específicas para posicionamento ou eventos – tartar como uma interrupção
- Leituras analógicas que precisam de filtros (media) – estabelecer um loop no tempo a ser tratada ou adicionar uma leitura em tempos menores até formar a tabela para a media.
- Divisores e multiplicadores por 2^n são rápidos pois são rotações de bit para a direita ou para a esquerda. Lembrar disso ao calcular medias – utilizar media de 2, 4, 8, 16, 32 leituras...
- Especializar com periféricos (hardware) coisas detalhadas e rápidas ou que usam muitos pinos (neste caso usar latches ou GPIO para liberar os pinos do microcontrolador).