Instituto Superior Técnico

Aplicações e Computação para a Internet das Coisas

20xx-20xx

1º Laboratório: Construir um sistema embebido

Grupo:		
Aluno 1:		
Aluno 2:		

Objetivo

O objetivo deste trabalho é colocar pela primeira vez os alunos em contacto com o ambiente do Arduino/ESP32 e controlar atuadores simples (neste caso LEDs).

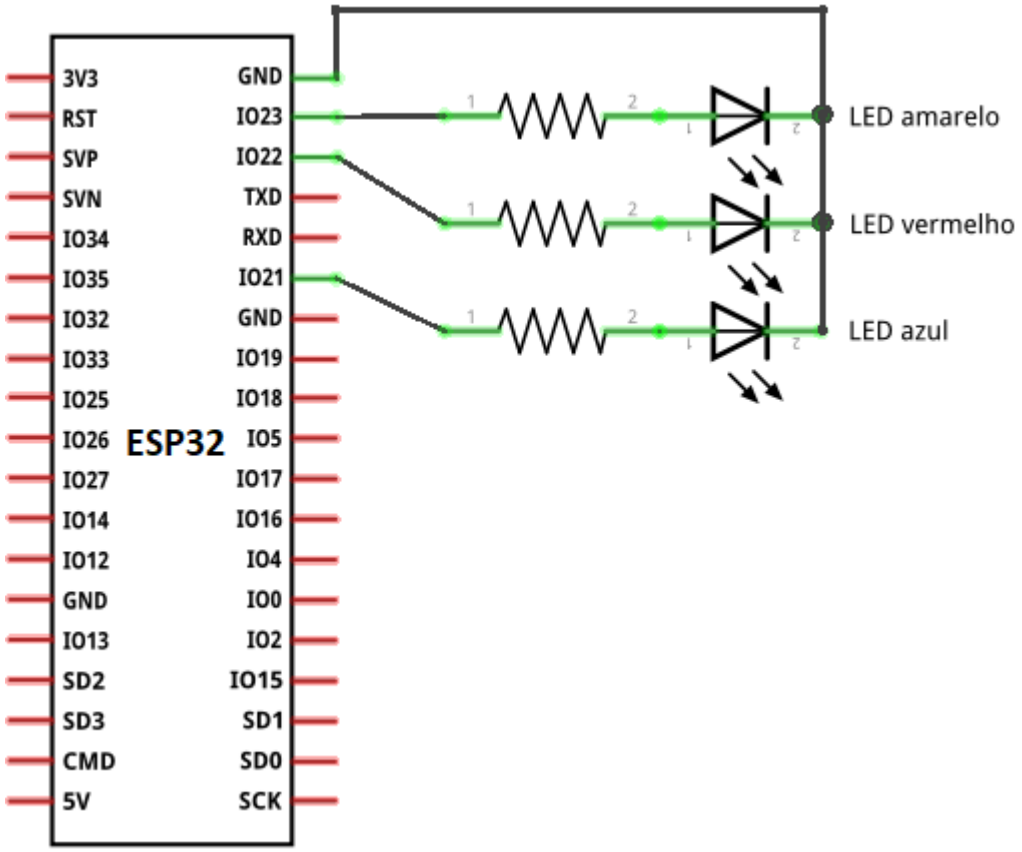
Descrição

Construa um sistema embebido usando um ESP32 para controlar 3 LEDs com cores diferentes. Durante a operação normal, a cada período de 4 segundos o sistema terá o seguinte comportamento, com, no máximo, apenas um LED ativo de cada vez (janelas de 1 segundo):

1. LED amarelo ligado
2. LED vermelho ligado
3. LED azul ligado
4. Todos os LEDs desligados

O comportamento é então repetido.

Um diagrama do circuito a montar está representado na seguinte figura.



Referências

1. Pinos ESP32: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/
2. Instalar o ESP32 no Arduino IDE: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/
3. Entrada/Saída digital: https://randomnerdtutorials.com/esp32-digital-inputs-outputs-arduino/
4. Delay: https://www.arduino.cc/en/Reference/Delay
5. Millis: https://www.arduino.cc/reference/en/language/functions/time/millis/
6. Leitora analógica: https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/

Recomendações

De modo a realizar o seu trabalho em segurança e não estragar nenhum equipamento usado, lembre-se de ter em conta as recomendações abaixo mencionadas. Enquanto trabalha, preencha as caixas para ter a certeza que cumpre todas as medidas de segurança.

Trabalhe sempre com os circuitos desconectados da sua fonte de alimentação.	
Chame o professor, ou o responsável pelo laboratório, antes de conectar os circuitos à sua fonte de alimentação.	
Verifique se o circuito está bem montado (sensores, resistências, capacitores, etc.) para prevenir curto-circuito ou estragos no equipamento. (Ex. nunca conecte um LED diretamente ao pino do controlador e ao GND, ou VCC.)	
Evite dobrar os terminais dos componentes o máximo possível. Se necessário, (ex. resistências) dobre o terminal a aproximadamente 5mm do corpo do componente.	

Este trabalho deverá ser implementado em 1 sessão de laboratório.

Desenhar a interface

1. Calcule o valor das resistências associadas aos LEDs.
    a. R_amarelo
    b. R_vermelho
    c. R_azul

2. Ligue um botão de pressão ao circuito.

    Sempre que o botão for pressionado (OFF → ON → OFF) o LED ativado nesse momento deve ficar aceso (Na etapa 4 o sistema pára com todos os LEDs desligados). Com o sistema parado é mais fácil ler a queda de voltagem de cada LED. Quando pressionar o botão outra vez o sistema deve continuar a sua operação normal.

    Projete e desenhe o circuito que liga o botão ao controlador.

3. Meça a queda de voltagem em cada LED. Pode usar um multímetro ou medir usando um pino analógico do ESP32 e transferindo o valor para o PC. Descreva o método usado.
    a. V_amarelo
    b. V_vermelho
    c. V_azul

4. Estime o consumo de energia da montagem feita (o circuito com as resistências e os LEDs na figura) em operação normal.

Programe a aplicação

Forneça o código criado, adequadamente estruturado e comentado.

```
void setup(){
  ...
}

void loop(){
  ...
}

...
```

Proposta de resolução

Código:

```
const int yellowLedPin = 23;
const int redLedPin = 22;
const int blueLedPin = 21;
const int buttonPin = 18;

int state = 0;

// Debounce
int buttonState;
int lastButtonState = HIGH;
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;

// Blink LED period
const int blinkPeriod = 1000;
unsigned long lastBlinkTime = 0;

void setup() {
  pinMode(yellowLedPin, OUTPUT);
  pinMode(redLedPin, OUTPUT);
  pinMode(blueLedPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP); // INPUT_PULLUP inverts HIGH and LOW when
reading
}

void loop() {

  // Debounce button and change state of the loop STOP/RUN
  int reading = digitalRead(buttonPin);
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {
      buttonState = reading;

      if (buttonState == LOW) {
        state = state * -1;
      }
    }
  }
  lastButtonState = reading;

  // Run the normal LED's sequence
```

```
  if ((millis() - lastBlinkTime) > blinkPeriod) {
    switch(state) {
      case 1:
        digitalWrite(yellowLedPin, HIGH);
        digitalWrite(redLedPin, LOW);
        digitalWrite(blueLedPin, LOW);
        break;
      case 2:
        digitalWrite(yellowLedPin, LOW);
        digitalWrite(redLedPin, HIGH);
        digitalWrite(blueLedPin, LOW);
        break;
      case 3:
        digitalWrite(yellowLedPin, LOW);
        digitalWrite(redLedPin, LOW);
        digitalWrite(blueLedPin, HIGH);
        break;
      case 4:
        digitalWrite(yellowLedPin, LOW);
        digitalWrite(redLedPin, LOW);
        digitalWrite(blueLedPin, LOW);
        break;
      default:
        return;
    }
    state = (state % 4) + 1;
    lastBlinkTime = millis();
  }
}
```