Instituto Superior Técnico

Aplicações e Computação para a Internet das Coisas

20xx-20xx

2º Laboratório: Sentir o Mundo Físico

Grupo:		
Aluno 1:		
Aluno 2:		

Objetivo

O objetivo deste trabalho é medir quantidades físicas e controlar atuadores de acordo com o ambiente medido.

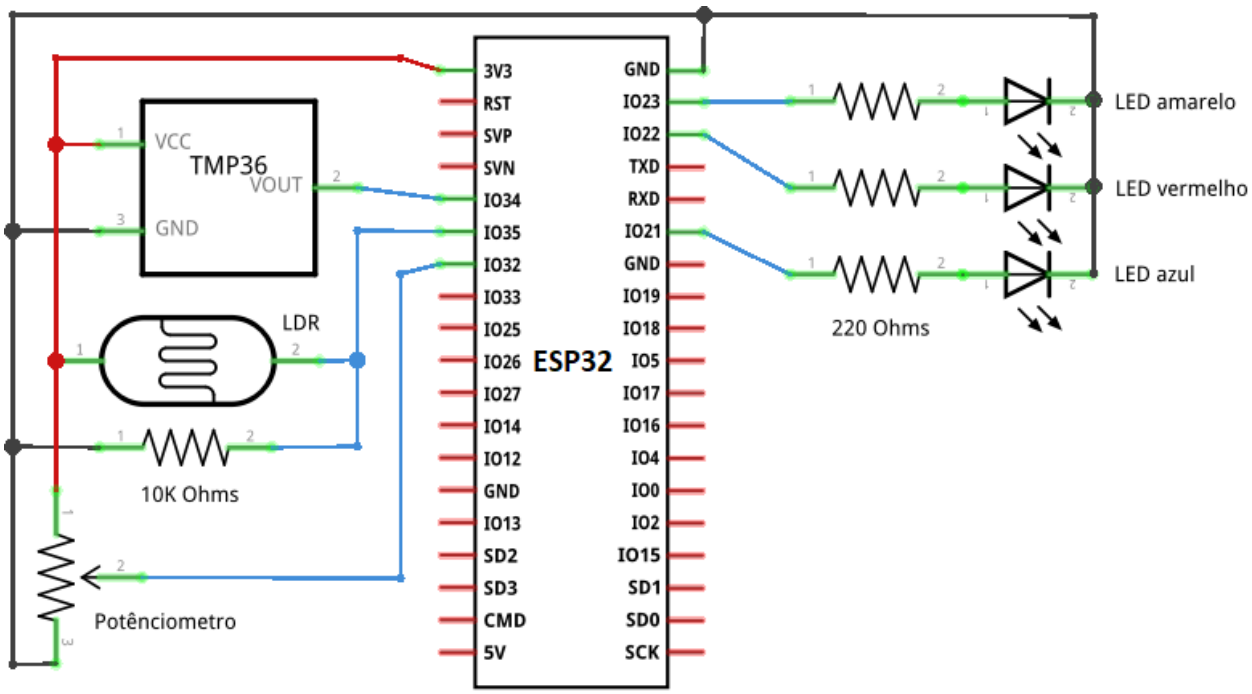
Descrição

Construa um sistema embebido usando um ESP32 para simultaneamente controlar 3 LEDs (os atuadores) dependendo do estado de 3 sensores - temperatura, ângulo de rotação (potenciômetro) e intensidade da luz.

- **Temperatura** - O LED amarelo associado à temperatura deve estar ligado quando a temperatura lida for maior que 26ºC (valor que eventualmente pode ser redefinido no laboratório de acordo com as condições ambientais).
- **Rotação** - O LED vermelho controlado pelo potenciômetro deve piscar com um período de tempo a variar continuamente entre 0.2 e 2 segundos, dependendo do ângulo de rotação do potenciômetro.
- **Luz** - O LED azul deve ajustar continuamente a sua própria intensidade com base na intensidade de luz medida no ambiente em que se encontra. O seu comportamento deve ser:
    - Escuridão - LED totalmente ligada
    - Luminoso - LED desligada

  (Ver referência sobre PWM)

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



Referências

1. Pinos ESP32: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/
2. Entrada/Saída digital: https://randomnerdtutorials.com/esp32-digital-inputs-outputs-arduino/
3. Leitora analógica: https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/
4. Calibração: https://www.arduino.cc/en/Tutorial/Calibration
5. Monitor Serial: https://esp32io.com/tutorials/esp32-serial-monitor
6. Delay: https://www.arduino.cc/en/Reference/Delay
7. PWM: https://randomnerdtutorials.com/esp32-pwm-arduino-ide/
8. Millis: https://www.arduino.cc/reference/en/language/functions/time/millis/

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.

Mapear medições analógicas

Normalmente as leituras digitais recolhidas dos sensores não correspondem diretamente ao valor de quantidade física, mas sim a valores entre 0 e um valor binário máximo (ex. o ADC do ESP32 usa palavras de 12 bits, 4095 = 2^{12} - 1). Assim, algum mapeamento é preciso.

Quando o valor apenas se encontra desviado, um mapeamento simples é adequado, mas em geral será necessário uma conversão mais complexa. A função `map`, disponível no Arduino IDE, simplifica este tipo de conversões. Por exemplo, ao rodar um motor Servo com a leitura de um potenciômetro, sabemos que quando o valor do potenciômetro é 0 o ângulo do Servo também deve ser 0º. Logicamente, se o valor lido no potenciômetro é 4095 (leitura máxima) o ângulo do Servo deve ser 180º (posição máxima):

```
map(valor, 0, 4095, 0, 180)
```

Além disso, alguns sensores requerem a linearização da sua função de transferência (quantidade física → quantidade elétrica, ex. a voltagem). Por exemplo, para converter a leitura de um circuito com um sensor de temperatura (no nosso caso a temperatura depende da resistência) para a temperatura real, várias transformações são necessárias:

- linearizar a função de transferência do sensor $R_T = f(T)$.
- linearizar a função de transferência do circuito em que o sensor está incluído.

Para o componente e configuração a usar (baseado no componente TMP36) verifica a seguinte função:

$$T = (((sensorvalue/4096.0) * 3.3) - 0.5) * 100$$

Programar com sensores analógicos

Para aceder a um sensor analógico externo deve ligá-lo a um pino analógico do ESP32. A correspondência entre o pino analógico e o software é feita com o seguinte código:

```
const int tempSensorPin = 34;
```

Onde `34` é o pino físico a que o sensor está ligado.

Para ler o valor de um pino específico use a função `analogRead(PIN)`, como se segue:

```
int temperatureValue = analogRead(tempSensorPin);
```

Debug

De modo a controlar algumas variáveis e depurar o programa é possível imprimi-las no monitor de Serial, disponível no Arduino IDE (“Serial Monitor”). Este tipo de ferramenta é útil, por exemplo, para observar a variação da temperatura ou para ajudar a calibrar o sistema.

Para usar esta funcionalidade, primeiro inicie a comunicação Serial com o PC:

```
void setup(){...; Serial.begin(115200); ...;}
```

Depois, use `Serial.println` para imprimir as suas variáveis ou texto:

```
Serial.println(temperatureValue);
```

Programe a aplicação

Forneça o código criado.

Estruture o programa modularmente segmentando o código em blocos separados, um para cada função a ser implementada (ex. ler o sensor de temperatura, controlar o alarme de temperatura). No próximo laboratório terá de distribuir algumas funções por dois controladores ESP32 independentes.

Evite, ou use com cuidado, a função `delay` pois esta pára a execução do programa durante o intervalo de tempo especificado.

- Código:

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

void loop(){
  ...
}

...
```

Resultados

1. Para cada um dos 3 pares de sensor-atuador descreva:
    a. O processo de mapeamento implementado.
    b. O processo de calibração.
    c. O processo, ou técnica usada para modular o comportamento do atuador.
    d. A configuração preparada para demonstrar a funcionalidade do sistema.
2. Qual o modelo de comportamento de software utilizado para desenvolver a aplicação?
3. Quais são as limitações de tempo do sistema? (tempo de execução)

Proposta de resolução

Código:

```
const int tempLedPin = 23;
const int potLedPin = 22;
const int lightLedPin = 21;

const int tempSensorPin = 34;
const int lightSensorPin = 35;
const int potSensorPin = 32;

// PWM setup
const int pwmChannel = 0;
const int pwmFreq = 5000;
const int pwmResolution = 8;

// Calibration
const float tempOffset = 0.0;
const int potMin = 0;
const int potMax = 4095;
const int lightSensorMin = 0;
const int lightSensorMax = 4095;

unsigned long prevBlinkCheck = 0;

void setup() {
  Serial.begin(115200);

  pinMode(tempLedPin, OUTPUT);
  pinMode(potLedPin, OUTPUT);
  pinMode(lightLedPin, OUTPUT);

  ledcSetup(pwmChannel, pwmFreq, pwmResolution);
  ledcAttachPin(lightLedPin, pwmChannel);
}

void loop() {
  setLedBlinkIntervalTask( readBlinkIntervalTask() );
  setLedIntensity( readLightTask() );
  setLedTemp( readTempTask() );
}

float readTempTask() {
  int reading = analogRead(tempSensorPin);
  float temperature = (((reading / 4096.0) * 3.3) - 0.5) *100;
  temperature = temperature + tempOffset;
  Serial.print("Temperature: "); Serial.println(temperature);
```

```
  return temperature;
}

void setLedTemp(float temperature) {
  if ( temperature > 23.0 ) digitalWrite(tempLedPin, HIGH);
  else digitalWrite(tempLedPin, LOW);
}

int readLightTask() {
  int reading = analogRead(lightSensorPin);
  int intensity = map(reading, lightSensorMin, lightSensorMax, 0, 255);
  Serial.print("Intensity: "); Serial.println(intensity);
  return intensity;
}

void setLedIntensity(int intensity) {
  // channel where the led is attached
  ledcWrite(pwmChannel, 255 - intensity);
}

int readBlinkIntervalTask() {
  int reading = analogRead(potSensorPin);
  int readBlinkInterval = map(reading, potMin, potMax, 200, 2000);
  Serial.print("Read Blink Interval (ms): "); Serial.println(readBlinkInterval);
  return readBlinkInterval;
}


void setLedBlinkIntervalTask(int blinkInterval) {
  if (millis() - prevBlinkCheck > blinkInterval) {
    prevBlinkCheck = millis();
    digitalWrite(potLedPin, LOW);
  } else if (millis() - prevBlinkCheck > blinkInterval/2) {
    digitalWrite(potLedPin, HIGH);
  }
}
```