Instituto Politécnico da Guarda Escola Superior de Tecnologia e Gestão

Trabalho Prático 2

Arquitetura de Computadores

Ana Vidal

Índice

Introdução

Este trabalho é dividido em duas partes, a primeira trata-se de fazer um estudo comparativo sobre o número de vezes que a função loop() é chamada enquanto usamos o sonar HC-SR04 para medir a distância com uma frequência de 15Hz, usando ou não interrupts. As distâncias a serem medidas para esta análise são 50cm, 100cm e 150cm.

Relativamente á segunda parte, optei por realizar o projeto do controlo do brilho do led usando o sonar. Este consiste em aumentar a luz do led à medida que o objeto se afasta, isto é, à medida que a distancia aumenta.

Projeto

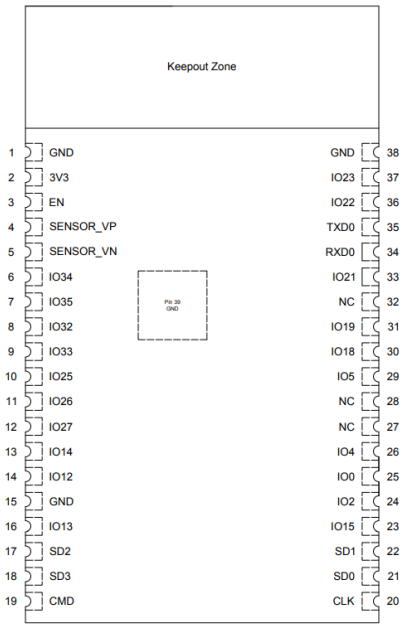

Figura1 – Pin Layout (Top View)

Sensor HC-SR04

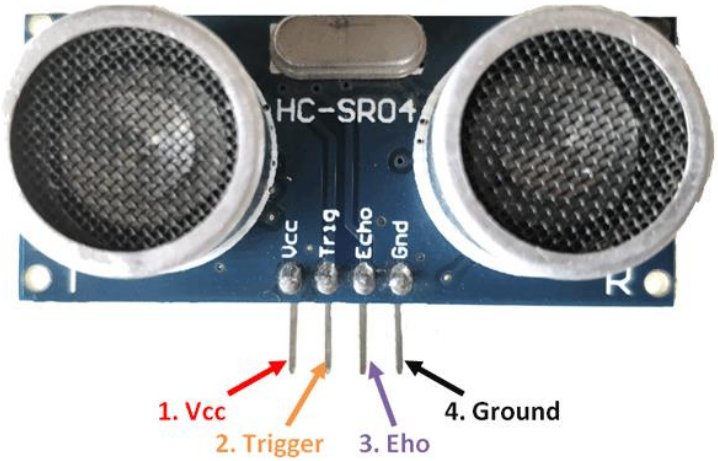

Figura2 – HC-SR04 Sensor pinout

Como exemplificado acima, o sensor HC-SR04 é um modulo de 4 pinos. Este sensor é usado em muitas aplicações onde a medição de distância ou objetos de deteção são necessário. A medição da distância é através de uma fórmula simples.

$Dis\tan c\,ia\ = \frac{SoundSpeed \times tempo}{2}$ m

$SoundSpeed\ =\ 340$ m/s

$$\frac{SoundSpeed}{2} = 170$$

Passar metros para centímetros:

$$170 \times 10^{-6} \times 10^{2} = 0.017$$

$Dis\tan c\,ia\ = tempo \times 0.17$ cm

Assim o trigger transmite uma onda ultrassônica, onde esta viaja até encontrar o obstáculo, que posteriormente é refletida de volta (echo) para o sensor, como é exemplificado na figura abaixo.

Figura3 – Sensor funcionando

Deste modo, calculamos a distância e analisamos os resultados obtidos.

Parte 1

Esquema do circuito

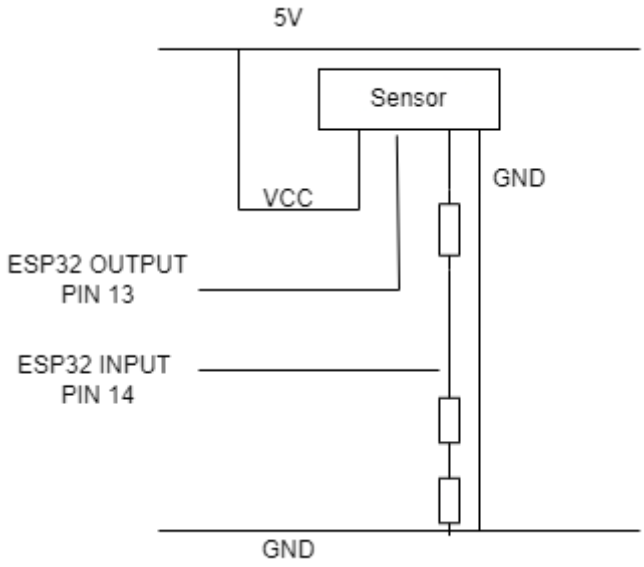

Figura4 – Esquema do circuito parte 1

Neste circuito uso um sensor HC-SR04, um ESP32 e três resistências de 1kΩ.

Sem interrupts

Algoritmo

Setup()

1. Configurar pin 13 como output
2. Configurar pin 14 como input
3. Iniciar conexão com o esp32
4. Ler tempo inicial em microssegundos para o sensor
5. Ler tempo inicial em milissegundos para o contador
6. Atribuir pin 13 a HIGH

Loop()

1. Ler tempo atual em microssegundos para o sensor
2. Ler tempo inicial em milissegundos para o contador
3. Adicionar um à variável contar
4. Se tempo atual menos tempo inicial for maior ou igual a 67000us então
    - 4.1. Atribuir pin 13 a LOW
    - 4.2. Ler sonar
    - 4.3. Atribuir pin 13 a HIGH
    - 4.4. Calcular distância
    - 4.5. Adicionar 67000us ao tempo inicial para o sensor
5. Fim se
6. Se tempo atual menos tempo inicial for maior ou igual a 1000ms então
    - 6.1. Definir zero à variável contar
    - 6.2. Adicionar 1000ms ao tempo inicial para o contador
7. Fim se

Código

```
#define TRIG_PIN 13
#define ECHO_PIN 14
unsigned long InicialTimeSonar;
unsigned long CurrentTimeSonar;
unsigned long InicialTimeContador;
unsigned long CurrentTimeContador;
unsigned long tempo;
int contar = 0;
long Distancia;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(115200);
  InicialTimeSonar = micros();
  InicialTimeContador = millis();
  digitalWrite(TRIG_PIN, HIGH);
}
void loop() {
  CurrentTimeSonar = micros();
  CurrentTimeContador = millis();
  contar++;
  if(CurrentTimeSonar - InicialTimeSonar >= 67000){
    digitalWrite(TRIG_PIN, LOW);
    tempo = pulseIn(ECHO_PIN, HIGH);
    digitalWrite(TRIG_PIN, HIGH);
    Distancia = tempo * 0.017;
    Serial.print("\nDistancia: \n");
    Serial.print(Distancia);
    InicialTimeSonar += 67000;
    Serial.print("\n\n");
  }
  if(CurrentTimeContador - InicialTimeContador >= 1000){
    Serial.print("\nContador: \n");
    Serial.print(contar);
    contar = 0;
    InicialTimeContador += 1000;
  }
}
```

Medidas

Contador/Distancia	50cm	100cm	150cm
NºLoop1	427613	404455	386456
NºLoop2	427641	405062	386397
NºLoop3	427681	406813	386418

Análise

De acordo, com os dados obtidos observei que à medida que a distancia aumenta, o número de vezes que função loop é chama diminui, isto deve se ao facto da função pulseIn(). Esta função captura a duração do pulso no pin echo quando está a HIGH até passar a LOW ou vice-versa, no meu caso começa a temporizar quando passa para HIGH, depois faz uma espera até que o pin echo passe para o estado LOW e termina de temporizar, portanto á medida que a distancia aumenta maior é a espera da função até ao pin echo mudar para o estado LOW.

Com interrupts

Algoritmo

Função TRIG

1. Escrever no pin 13 a variável disparar
2. Inverter valor da variável disparar

Função ECHO

1. Ler pin 14
2. Se pin 14 for igual a HIGH então
    2.1. Ler tempo inicial
3. Fim se
4. Se não
    4.1. Ler tempo final
    4.2. Calcular tempo
    4.3. Detetar nova medida
5. Fim se não

Setup()

1. Configurar pin 13 como output

2. Configurar pin 14 como input
3. Iniciar conexão com o esp32
4. Ler tempo inicial em milissegundos
5. Associar o timer 0, com o divisor 80 e com incremento
6. Associar a função tempo ao timer 0
7. Definir o período 67000 us do timer 0 e periódico
8. Inicializar o timer 0
9. Associar o pin 14 para ativar o interrupt e chamar a função echo no modo CHANGE
10. Atribuir pin 13 a HIGH

Loop()

1. Ler tempo atual em milissegundos
2. Adicionar um à variável contar
3. Se nova distância for igual a true então
    - 3.1. Calcular distância
    - 3.2. Se Distancia menor que 3
        - 3.2.1. Distancia é igual a 0
    - 3.3. Fim se
    - 3.4. Se Distancia maior 399
        - 3.4.1. Distancia é igual a 400
    - 3.5. Fim se
4. Fim se
5. Se tempo atual menos tempo inicial for maior ou igual a 1000 então
    - 5.1. Variável contar igual a zero
    - 5.2. Adicionar 1000 ao tempo inicial
6. Fim se

Código

```
#define TRIG_PIN 13
#define ECHO_PIN 14
unsigned long InicialTime;
unsigned long CurrentTime;
long tempo;
unsigned long tempoSegundos;
unsigned long InicialTimeSonar;
unsigned long FinalTimeSonar;
int contar = 0;
volatile long Distancia;
hw_timer_t*timer_n = NULL;
bool disparar;
bool novaDistancia;
```

```
void IRAM_ATTR trig(){
  digitalWrite(TRIG_PIN, disparar);
  disparar = !disparar;
}

void IRAM_ATTR echo(){
  //bool estado = digitalRead(ECHO_PIN);
  if(digitalRead(ECHO_PIN) == HIGH){
    InicialTimeSonar = micros();
  }
  else{
    FinalTimeSonar = micros();
    tempo = FinalTimeSonar - InicialTimeSonar;
    novaDistancia = true;
  }
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(115200);
  InicialTime = millis();
  timer_n = timerBegin(0, 80, true);
  timerAttachInterrupt(timer_n, trig, true);
  timerAlarmWrite(timer_n, 67000, 1);
  timerAlarmEnable(timer_n);
  attachInterrupt(ECHO_PIN, echo, CHANGE);
  digitalWrite(TRIG_PIN, HIGH);
}

void loop() {
  CurrentTime = millis();
  contar++;
  if(novaDistancia == true){
      Distancia = tempo * 0.017;
    if(Distancia < 3){
       Distancia = 0;
    }
    else if(Distancia > 399){
      Distancia = 400;
    }
  }
```

```
  if(CurrentTime - InicialTime >= 1000){
    Serial.print("\nDistancia: \n");
    Serial.print(Distancia);
    Serial.print("\nContador: \n");
    Serial.print(contar);
    Serial.print("\n\n");
    contar = 0;
    InicialTime += 1000;
  }
}
```

Medidas

Contador/Distancia	50cm	100cm	150cm
NºLoop1	457969	457618	457826
NºLoop2	457610	457899	456305
NºLoop3	457441	458037	457820

Análise

Com o uso de interrupts observo que a função loop é chamada praticamente o mesmo número de vezes nas três distâncias diferentes, pois quando aciono o interrupt(timer ou external), interrompe a tarefa atual do processador e assim processar o que eu indiquei na função(trig ou echo), não sendo necessário fazer todas as verificações do loop principal.

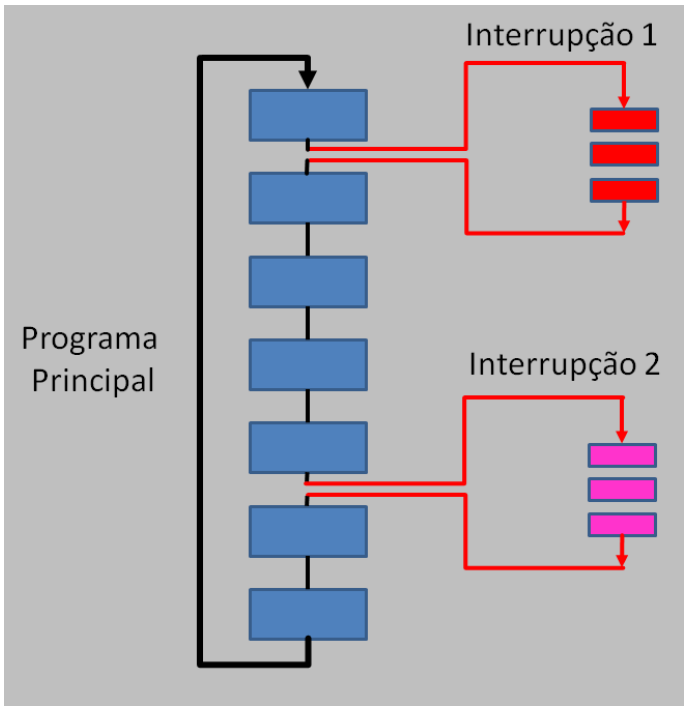

Figura5 – Esquema dos Interrupts

Parte 2

Esquema do circuito

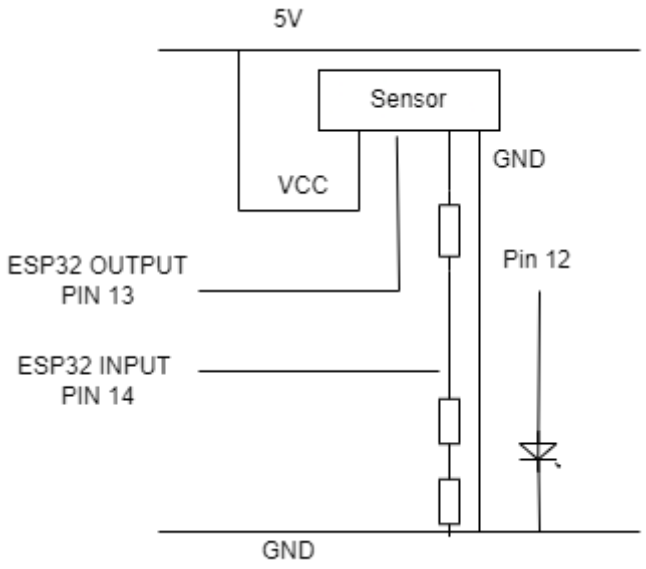

Figura6 – Esquema do circuito parte 2

Neste circuito uso um sensor HC-SR04, um ESP32, três resistências de 1kΩ e um Led.

Algoritmo

Função TRIG

3. Escrever no pin 13 a variável disparar
4. Inverter valor da variável disparar

Função ECHO

6. Ler pin 14
7. Se pin 14 for igual a HIGH então
    7.1. Ler tempo inicial
8. Fim se
9. Se não
    9.1. Ler tempo final
    9.2. Calcular tempo
    9.3. Detetar nova medida
10. Fim se não

Setup()

1. Configurar o pin 13 como output
2. Configurar pin 14 como input
3. Configurar pin 12 como output
4. Iniciar conexão com o esp32
5. Associar o timer 0, com o divisor 80 e com incremento
6. Associar a função tempo ao timer 0
7. Definir o período 67000 us do timer 0 e periódico
8. Inicializar o timer 0
9. Definir o canal zero do pwm com uma frequência de mil hz e uma resolução de 8 bits
10. Associar o pin 12 com o canal zero do pwm
11. Associar o pin 14 para ativar o interrupt e chamar a função echo no modo CHANGE
12. Atribuir pin 13 a HIGH

Loop()

1. Se nova distância for igual true então
    - 1.1. Calcular distancia
    - 1.2. Se Distancia menor que 3
        - 1.2.1. Distancia é igual a 0
    - 1.3. Fim se
    - 1.4. Se Distancia maior 399
        - 1.4.1. Distancia é igual a 400
    - 1.5. Fim se
2. Fim se
3. Usar uma função matemática para aumentar a luz do led à medida que a distancia aumenta
4. Atribuir a variável luz ao canal zero

Código

```
#define TRIG_PIN 13
#define ECHO_PIN 14
#define LED_PIN 12
long tempo;
volatile long Distancia;
int luz;
bool disparar;
hw_timer_t*timer_n = NULL;
unsigned long InicialTimeSonar;
unsigned long FinalTimeSonar;
```

```
bool novaDistancia;


void IRAM_ATTR trig(){
  digitalWrite(TRIG_PIN, disparar);
  disparar = !disparar; //inverter variável disparar
}

void IRAM_ATTR echo(){
  if(digitalRead(ECHO_PIN) == HIGH){
    InicialTimeSonar = micros();
  }
  else{
    FinalTimeSonar = micros();
    tempo = FinalTimeSonar - InicialTimeSonar;
    novaDistancia = true; //detetar que há uma nova medida
  }
}

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(115200);
  timer_n = timerBegin(0, 80, true);
  timerAttachInterrupt(timer_n, trig, true);
  timerAlarmWrite(timer_n, 67000, 1);
  timerAlarmEnable(timer_n);
  ledcSetup(0, 1000, 8);
  ledcAttachPin(LED_PIN, 0);
  attachInterrupt(ECHO_PIN, echo, CHANGE);
  digitalWrite(TRIG_PIN, HIGH);

}

void loop() {
  if(novaDistancia == true){
    Distancia = tempo * 0.017;
    if(Distancia < 3){
      Distancia = 0;
    }
    else if(Distancia > 399){
      Distancia = 400;
    }
```

```
    Serial.print("\nDistancia:");
    Serial.print(Distancia);
    Serial.print("\nLuz:");
    Serial.print(luz);
  }
  luz = (((Distancia - 3) * (255 - 0)) / ((400 - 3) + 0));
  ledcWrite(0, luz);
}
```

Medidas

Intensidade/Distancia	50cm	100cm	150cm
Luz	30	62	94
Luz	30	62	94
Luz	30	62	94

Análise

Nesta parte do trabalho foi pretendido a observação gradual da intensidade do brilho aumentar à medida que a distancia medida pelo sonar aumentada. Este este aumento da luz como da distância é diretamente proporcional. Desta forma, usei o PWN com uma resolução de 8 bits, definindo assim um valor máximo de controlo do ciclo. De seguida, usei a expressão matemática usada na função map(), para me definir uma nova escala para o aumento e diminuição da luz do brilho.

Conclusão

Começo por afirmar que alcancei o objetivo tanto da primeira parte do trabalho como na segunda. Podendo assim concluir que o uso de interrupts nas leituras do sonar é mais competente do que sem o uso dos mesmo para as três medidas pedidas, e assim optei por fazer a segunda parte do trabalho usando interrupts, pois permite uma forma mais fácil de lidar com tempos cíclicos, sendo também mais eficiente nos resultados.