POLI
TÉCNICO
GUARDA

Instituto Politécnico da Guarda
Escola Superior de Gestão e Tecnologia

Arquitetura de Computadores 2021/2022

Trabalho usando ESP32

Trabalho realizado por:
João Saraiva Aleixo
Nº 1704473

Índice

Introdução

No âmbito da disciplina de Arquitetura de Computadores foi-me proposto a realização de um trabalho utilizando o ESP32 que consiste em duas partes sendo a primeira parte comparar o número de vezes que a função loop é chamada quando fazemos leituras a 15Hz utilizando o sensor HC-SR04 com e sem interrupts, sendo este estudo feita para três distâncias já definidas 50cm, 100cm e 150cm. A segunda parte do trabalho consiste em controlar o brilho de um led de acordo com a distância medida pelo sonar.

ESP32

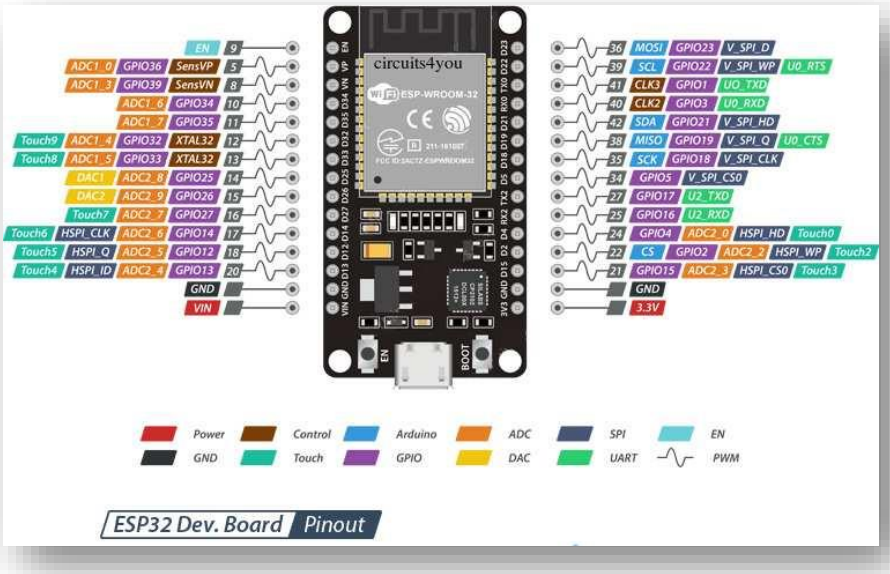


Figura 1. ESP32

Para o nosso trabalho utilizamos como base o ESP32, um microprocessador com uma grande variedade de recursos o qual se encaixa para a realização do trabalho proposto.

Sensor HC-SR04

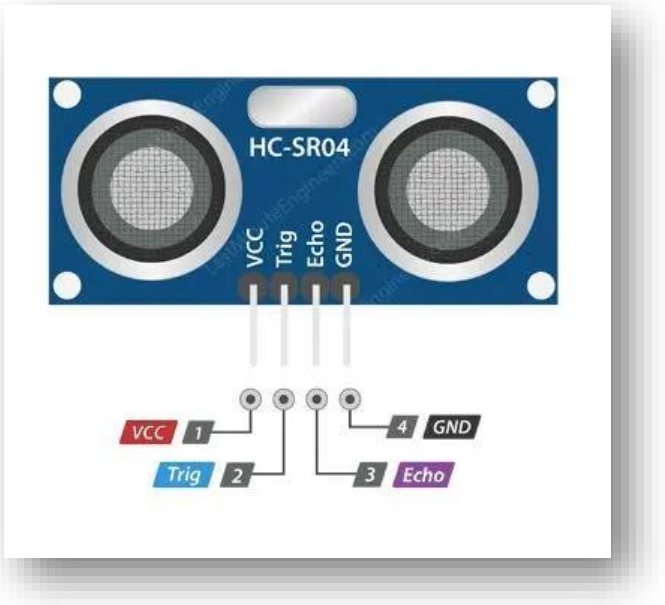


Figura 2. Sensor HC-SR04

O Sensor de Distância HC-SR04 foi desenvolvido para aperfeiçoar projetos de robótica e eletrónica, é ideal para calcular a distância com precisão de objetos. Este módulo possui um circuito pronto com emissor e receptor acoplados e 4 pinos (VCC, Trigger, ECHO, GND) para a medição. O HC-SR04 é um Sensor de Distância composto por um emissor e um receptor, com capacidade de medir distâncias de 2cm até 4m, com uma precisão de aproximadamente 3mm. Este sensor emite sinais ultrassónicos que refletem no objeto a ser atingido e retornam ao sensor, indicando a distância do alvo.

A velocidade do sinal ultrassónico emitida pelo Sensor HC-SR04 corresponde à velocidade do som, que é de aproximadamente 340 m/s, assim, se o sensor estiver a uma distância x do objeto, o sinal percorrerá uma distância equivalente a 2x, ou seja, a onda é enviada pelo sensor e rebatida no obstáculo, logo ela percorre 2 vezes a distância procurada.

Como calcular a distância com o sensor a partir do tempo:

2 × distancia = velocidadeSom × tempo

distancia = $\frac{velocidadeSom \times \text{tempo}}{2}$

velociadadeSom $\cong 340\text{m/s}$

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

<=> **distancia = tempo × 0.017**

Parte 1

Como já referi anteriormente para a primeira parte do trabalho vamos fazer uma comparação com o número de vezes que a função loop é chamada com e sem interrupts.

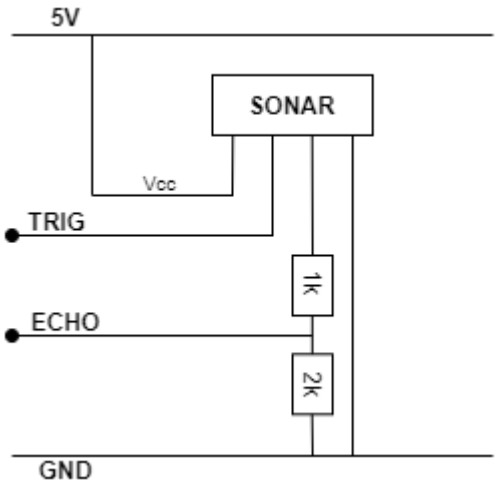


Figura 3. Circuito

Com Interrupts

Algoritmo

triggerFunction()

1. Escrever no pin 25 a variável turn
2. Inverter o valor da variável boleana turn

echoFunction()

1. Ler o pin 32
2. Se o pin 32 for igual a HIGH
   2.1. Ler tempo inicial
3. Se não
   3.1. Ler tempo corrente
4. Calcular a tempo da onda

Setup()

1. Conectar com o esp32
2. Configurar o pin 25 como OUTPUT
3. Configurar o pin 32 como INPUT
4. Ler o tempo inicial em milissegundos
5. Ler o tempo inicial distancia em microssegundos
6. Configurar o timer
7. Configurara interrupt externo

Loop()

1. Ler o tempo corrente em milissegundos
2. Ler tempo corrente distancia em microssegundos
3. Incrementar uma unidade á variável count
4. Se o tempo corrente distancia menos o tempo inicial distancia for maior ou igual a (1000000/15) então
   4.1. Calcular distancia
   4.2. Escrever distancia
   4.3. Adicionar ao tempo inicial distancia (1000000/15)
5. Se o tempo corrente menos o tempo inicial for maior ou igual a 1000 então
   5.1. Escrever count
   5.2. Igualar a variável count a 0
   5.3. Somar 1000 ao tempo inicial

Código

```
#define TRIGGER_PIN 25
#define ECHO_PIN 32

unsigned long startTime;
hw_timer_t * timer_n = NULL;
unsigned long currentTime;
int count;
bool turn; // Activar ou Desativar o trigger (Low,High)
bool state; // Estado do ECHO_PIN
unsigned long startTimeEcho;
unsigned long currentTimeEcho;
unsigned long startTimeDist;
unsigned long currentTimeDist;
// volatile é usado para variaveis que são chamadas fora das funções interrupt
volatile unsigned long dist;
```

```
volatile unsigned long waveTime;

void IRAM_ATTR triggerFunction(){ //  Emitir o sinal Trigger
  digitalWrite(TRIGGER_PIN,turn);
  turn = !turn;
}

void IRAM_ATTR echoFunction(){ // Receber o sinal da onda (medir o tempo)
  state = digitalRead(ECHO_PIN);
  if(state == HIGH){
    startTimeEcho = micros();
  }else{
    currentTimeEcho = micros();
    waveTime = currentTimeEcho - startTimeEcho ;
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN,INPUT);
  startTime = millis();
  startTimeDist = micros();
  timer_n = timerBegin(0,80,true);
  timerAttachInterrupt(timer_n,triggerFunction,true);
  timerAlarmWrite(timer_n,(1000000/15),true);
  timerAlarmEnable(timer_n);
  attachInterrupt(ECHO_PIN,echoFunction,CHANGE);
}

void loop() {
  currentTime = millis();
  currentTimeDist = micros();
  count++;

  if(currentTimeDist - startTimeDist >= (1000000/15)){
    dist = waveTime * 0.017;
    Serial.print("Distância: ");
    Serial.println(dist);
    startTimeDist += (1000000/15);
  }
  if(currentTime - startTime >= 1000){
    Serial.print("Número de loops: ");
    Serial.println(count);
    count = 0;
    startTime += 1000;

  }
```

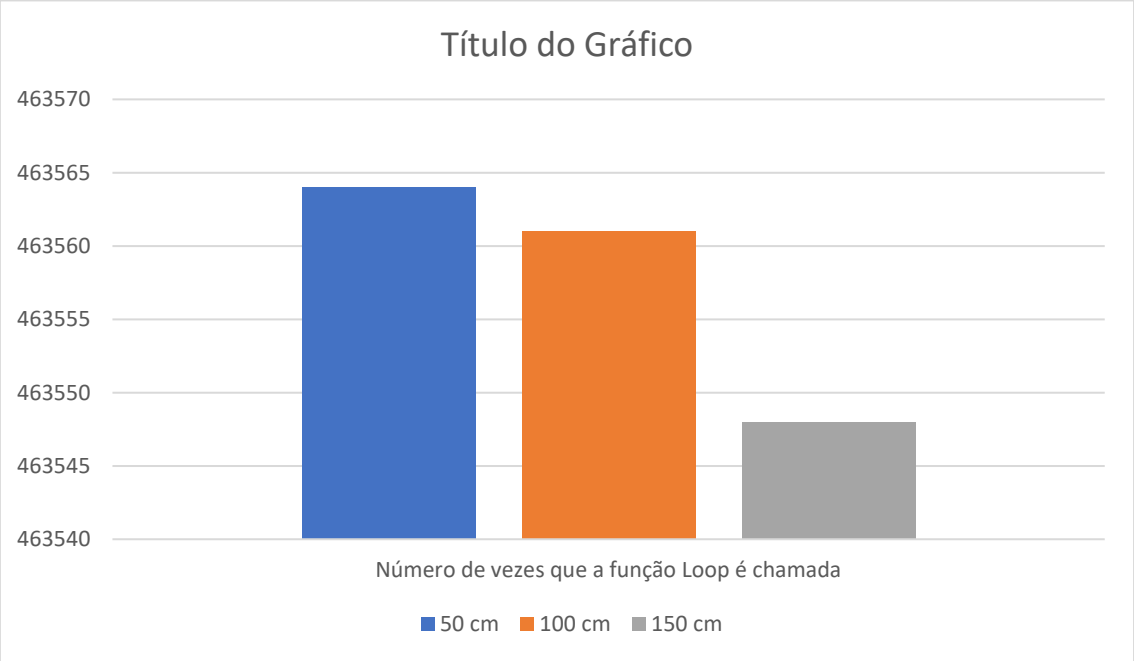
}

Medidas

Tabela

Distância	50 cm	100 cm	150 cm
Nº Loops	463570	463573	463538
Nº Loops	463573	463547	463559
Nº Loops	463549	463564	463547
Média	463564	463561	463548

Gráfico



Análise

Com base nos dados retirados podemos verificar que o número de vezes que a função loop é chamada são bastante similares nas três distâncias, isto deve-se ao facto de a distância ser calculada com base na diferença entre dois tempos (currentTimeEcho, startTimeEcho).

Sem Interrupts

Algoritmo

Setup()

1. Conectar com o esp32
2. Configurar o pin 32 como OUTPUT
3. Configurar o pin 25 como INPUT
4. Ler tempo inicial em milissegundos
5. Ler tempo inicial do sonar em microssegundos
6. Configurar o pin 25 como HIGH

Loop()

1. Ler tempo corrente do sonar em microssegundos
2. Ler tempo corrente em milissegundos
3. Incrementar uma unidade á variável count
4. Se tempo corrente do sonar menos o tempo inicial do sonar for maior ou igual a (1000000/15) então
   4.1. Configurar o pin 25 como LOW
   4.2. Calcular a distancia
   4.3. Configurar o pin 25 como HIGH
   4.4. Se distancia for menor que 3 então
      4.4.1. A variável distancia vai ser igual a 2
   4.5. Se a distancia for maior que 399 então
      4.5.1. A variável distancia vai ser igual a 400
   4.6. Escrever distancia
   4.7. Adicionar ao tempo inicial do sonar (1000000/15)
5. Se o tempo corrente menos o tempo inicial for maior ou igual a 1000 então
   5.1. Escrever count
   5.2. Igualar a variável count a 0
   5.3. Somar 1000 ao tempo inicial

Código

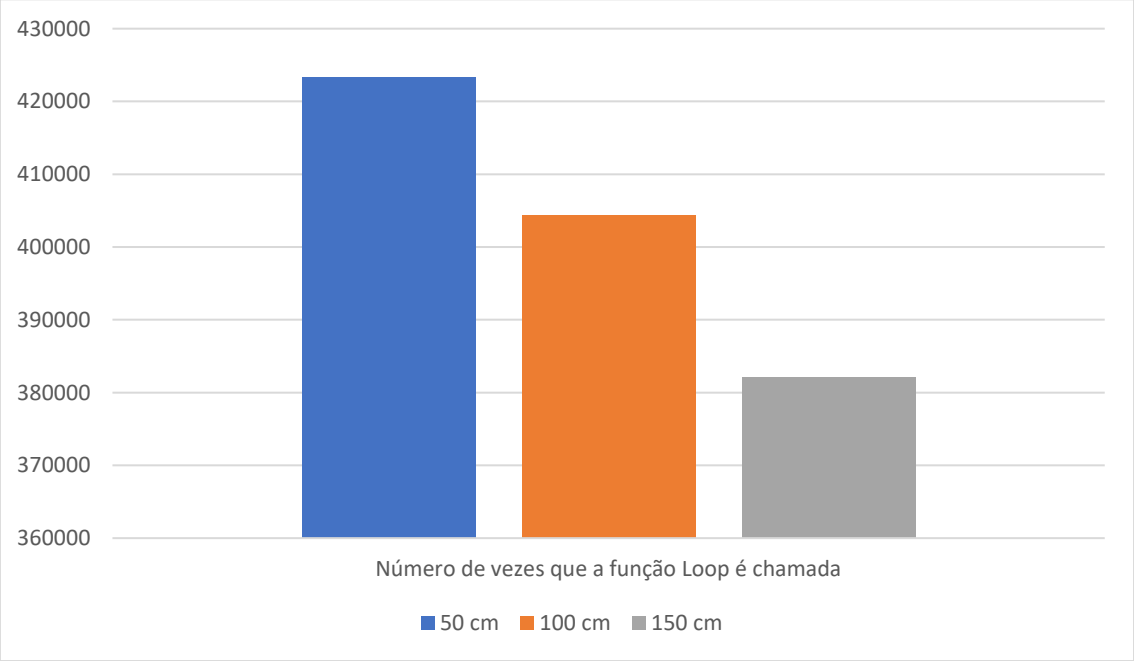
```
#define TRIGGER_PIN 25
#define ECHO_PIN 32

unsigned long startTime;
unsigned long currentTime;
unsigned long startTimeSonar;
unsigned long currentTimeSonar;
int count = 0;
int dist;

void setup() {
  Serial.begin(115200);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN,INPUT);
  startTime = millis();
  startTimeSonar = micros();
  digitalWrite(TRIGGER_PIN,HIGH);
}

void loop() {
  currentTimeSonar = micros();
  currentTime = millis();
  count ++;

  if(currentTimeSonar - startTimeSonar >= (1000000/15)){
    digitalWrite(TRIGGER_PIN,LOW);// verificar se o trigger está a 0
    int dist = pulseIn(ECHO_PIN,HIGH) * 0.017; // 170*10^-6*10^2
    digitalWrite(TRIGGER_PIN,HIGH);
    if(dist < 3){
      dist = 0;
     }
    if(dist > 399){
      dist = 400;
    }
    Serial.print("Distância: ");
    Serial.println(dist);
    startTimeSonar += (1000000/15);
  }
  if(currentTime - startTime >= 1000){
    Serial.print("Número de loops: ");
    Serial.println(count);
    count = 0;
    startTime += 1000;
   }
}
```

Medidas

Tabela

Distância	50 cm	100 cm	150 cm
Nº Loops	420022	404338	382472
Nº Loops	424872	404365	383744
Nº Loops	424884	404272	380269
Média	423259	404335	382161

Gráfico



Análise

Com base nos dados retirados podemos concluir que o valor vai diminuindo significativamente consoante o aumento da distância, isto deve-se ao facto de que para o cálculo da mesmo utilizamos a função pulseIn a qual faz um momento de espera até recebermos a onda cada vez que calculamos uma distância, seguindo essa lógica quanto maior for a distância maior será o tempo de espera, ou seja, menor será o número de loops.

Parte 2

Para a segunda parte do trabalho foi me proposto que fizesse variar o brilho do led de acordo com a distância medida pelo sonar. Para além do trabalho proposto consegui implementar também uma parte onde o led pisca num certo intervalo de espaço e consoante a distância este aumenta a velocidade com que pisca.

Em síntese programei o led para ter um brilho de 0 quando a distância medida pelo sensor é nula, por outro lado atinge o seu brilho máximo quando medimos uma distância de 49 cm além disso a passar pelas diferentes distâncias presentes neste intervalo [0cm,50cm [, notamos um aumento progressivo do brilho. Para a parte que adicionei ao trabalho proposto que está contido no intervalo [50cm,150cm], onde o começa por piscar com um maior período, o qual vai diminuindo consoante o aumento da distância e atingir o seu máximo à distância de 150 cm.

A razão utilizei destes intervalos para as distâncias foi pelo facto de que apesar de o sensor ter como capacidade máxima de distância 400cm, eu notei que este não apresentava muita precisão para distâncias superiores a $\cong 160\ \mathrm{cm}$, daí ter uma distância máxima para realização do meu trabalho de 150cm.

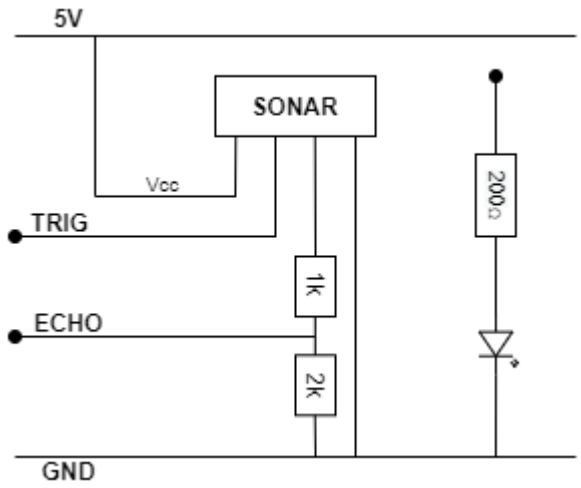


Figura 3. Circuito

Algoritmo

Setup()

1. Conectar com o esp32
2. Configurar o pin 25 como OUTPUT
3. Configurar o pin 32 como INPUT
4. Configurar o pin 33 como OUTPUT

5. Ler tempo inicial em microssegundos
6. Ler tempo inicial Blink em microssegundos
7. Associar o pin 33 ao canal 0
8. Atribuir ao canal 0 uma frequência de 1000 Hz com 8 bits
9. Escrever o pin 25 como HIGH

Loop()

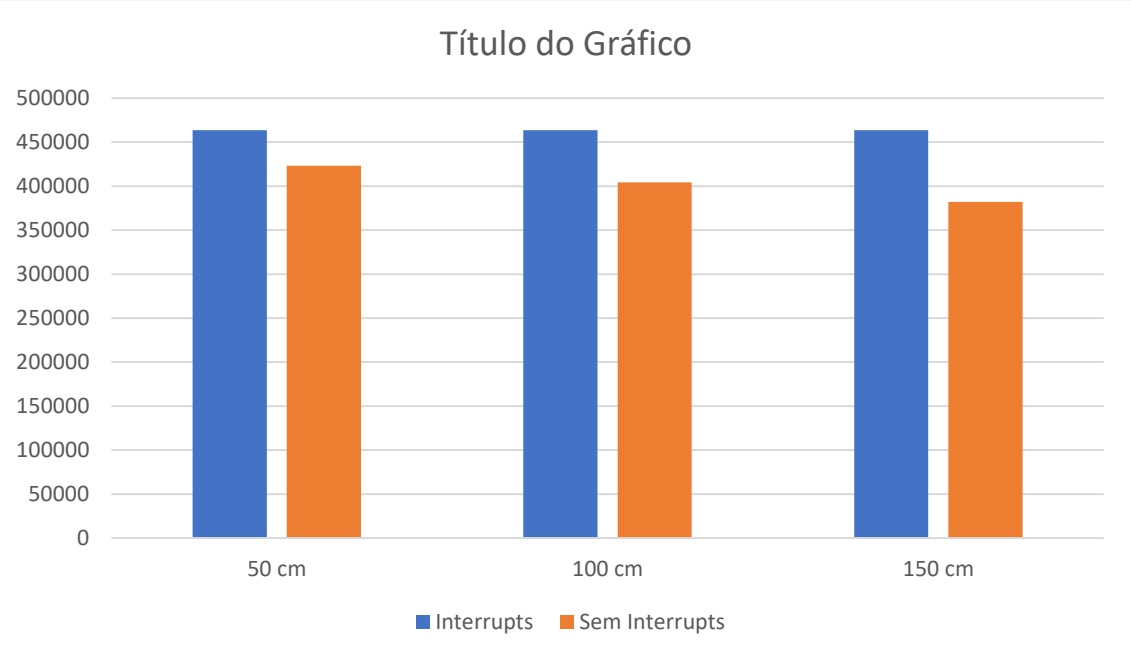
1. Ler tempo corrente
2. Ler tempo corrente Blink
3. Se o tempo corrente menos o tempo inicial for maior ou igual (1000000/15) então
   3.1. Escrever no pin 25 LOW
   3.2. Calcular a distancia
   3.3. Escrever no pin 25 HIGH
   3.4. Atribuir á nova variável distancia2 o valor da variável distancia
   3.5. Se distancia for maior ou igual do que 49 então
      3.5.1. A variável distancia2 vai ser igual a 49
   3.6. Se a distancia for menor ou igual do que 2 então
      3.6.1. A variável distancia vai ser igual a 0
      3.6.2. A variável distancia2 vai ser igual a 0
   3.7. Se a distancia for maior ou igual do que 400 então
      3.7.1. A variável distancia vai ser igual a 400
   3.8. Se a distancia for maior ou igual a 50 e menor ou igual do que 150 então
      3.8.1. Calcular a variável power
      3.8.2. Desassociar o pin 33
      3.8.3. Se o tempo corrente Blink menos o tempo inicial Blink for maior ou igual do que (1000000 menos a variável power) então
         3.8.3.1. Inverter o valor da variável booleana blink
         3.8.3.2. Escrever no pin 33 a variável blink
         3.8.3.3. Adicionar ao tempo inicial Blink (1000000 menos a variável power)
   3.9. Calcular a variável Bright
   3.10. Se a distancia for menor do que 50 ou maior do que 150
      3.10.1. Associar o pin 33 ao canal 0
      3.10.2. Escrever no pin 33 a variável Bright
   3.11. Escrever a variável distancia
   3.12. Somar a variável tempo inicial (1000000/15)

Código

```
#define TRIGGER_PIN 25
#define ECHO_PIN 32
#define LED_PIN 33

unsigned long currentTime;
unsigned long startTime;
unsigned long currentTimeBlink;
unsigned long startTimeBlink;
```

```
int dist;
int dist2;
double power;
double bright;
boolean blink;

void setup() {
  Serial.begin(115200);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  startTime = micros();
  startTimeBlink = micros();
  ledcAttachPin(LED_PIN, 0);//Atribuimos o pino 33 ao canal 0.
  ledcSetup(0, 1000, 8);//Atribuimos ao canal 0 a frequencia de 100Hz com resolucao de 8bits.
  digitalWrite(TRIGGER_PIN,HIGH);
}
void loop() {
  currentTime = micros();
  currentTimeBlink = micros();
  if (currentTime - startTime >= (1000000 / 15)) {
    digitalWrite(TRIGGER_PIN,LOW);// verificar se o trigger está a 0
    int dist = pulseIn(ECHO_PIN,HIGH) * 0.017; // 170*10^-6*10^2
    digitalWrite(TRIGGER_PIN,HIGH);
    dist2 = dist;
    if(dist >= 49){
      dist2 = 49;
    }
    if (dist <= 2) {
      dist = 0;
      dist2 = 0;
    }
    if (dist >= 400) {
      dist = 400;
    }
    if(dist >= 50 && dist <= 150){
      power = 6000 * dist;
      ledcDetachPin(LED_PIN);
      if(currentTimeBlink - startTimeBlink >= (1000000-power)){
        blink = !blink;
        digitalWrite(LED_PIN,blink);
        startTimeBlink += (1000000-power);
      }
    }
    bright = (dist2*255) / 49;
    if(dist < 50 || dist > 150){
      ledcAttachPin(LED_PIN, 0);//Atribuimos o pino 33 ao canal 0.
      ledcWrite(0, bright);
    }
    Serial.print("Distancia: ");
    Serial.println(dist);
    startTime += (1000000 / 15);
  }
}
```

Conclusão



Em relação á primeira parte do trabalho após retirar os dados para as três distâncias propostas (50cm,100cm,150cm), pude assim concluir que em relação á dimensão do valor de número de vezes que a função loop é chamada é visível que quando fazemos as leituras com interrupts estes valores são relativamente maiores, isto deve-se a como já referi anteriormente quando usamos interrupts o CPU vai parar de executar o programa principal e executar os interrupts quando acaba essa execução volta automaticamente para o programa principal onde este foi interrompido .

Outra conclusão que consegui retirar está relacionada com a variância do número de vezes que a função loop é chamada de acordo com a distância, ou seja, foi possível verificar que usando interrupts os valores eram bastante similares entre si, por outro lado, quando não usamos interrupts verificamos uma discrepância entre os valores observados sendo que estes vai diminuindo consoante a distância vai aumentando, isto é comprovado pelas diferentes maneiras que calculamos a distância, ou seja, quando usamos interrupts a distância é calculada com base na diferença de dois tempos, ou seja, não envolve esperar por um sinal que é exatamente o que acontece quando não usamos interrupts, por outras palavras para calcular a distância temos que esperar que após o Trigger emitir um sinal, este seja recebido consequentemente quanto maior for a distância maior será o tempo de espera convertendo-se assim em um menor número de vezes que a função Loop é chamada.

No que diz respeito á segunda parte do trabalho consegui introduzir uma nova funcionalidade ao programa e assim desenvolver um pouco as minhas capacidades.

Em suma foi um trabalho interessante e consegui cumprir com todos os objetivos propostos.