Practica A:

1. Informe de Funcionament i sortides

```
C/C++
struct Button {
const uint8_t PIN;
uint32_t numberKeyPresses;
bool pressed;
};
Button button1 = {18, 0, false};
```

Creació d'una estructura que actua com a botó per guardar els valors d'apretat o no apretat, el numero de vegades que s'ha apretat i el pin on està connectat. A continuació assigna els valors del pin a la sortida 18 i inicialitza els valors restants.

```
C/C++
void IRAM_ATTR isr() {
button1.numberKeyPresses += 1;
button1.pressed = true;
}

void setup() {
Serial.begin(115200);
pinMode(button1.PIN, INPUT_PULLUP);
attachInterrupt(button1.PIN, isr, FALLING);
}
```

Creació de les funcions de setup per inicialitzar el pin de la ESP32 i la creació de la funció per canviar l'estat del botó.

```
C/C++
void loop() {
      if (button1.pressed) {
            Serial.printf("Button 1 has been pressed %u
            times\n", button1.numberKeyPre
            button1.pressed = false;
      }
//Detach Interrupt after 1 Minute
      static uint32_t lastMillis = 0;

      if (millis() - lastMillis > 60000) {
            lastMillis = millis();
            detachInterrupt(button1.PIN);
```

```
            Serial.println("Interrupt Detached!");
            }
}
```

Inici del loop infinit de la ESP32:

a. Si el botó s'apreta apareix un missatge per pantalla especificant el número total de vegades que s'ha apretat el botó. Al finalitzar el missatge es torna a posar el valor del botó a no apretat;
b. Inicialització del valor lastMillis a 0;
c. Si el valor de millis() (el temps que ha passat des de la ultima vegada que s'ha interromput el servei) inicialment es superior a 60.000ms o 1 minut, s'interromp el funcionament del botó i s'imprimeix un missatge per pantalla informant de la interrupció.

Practica B:

1. Informe de Funcionament i sortides

```C/C++
volatile int interruptCounter;
int totalInterruptCounter;
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
```

Inicialització de valors de contador d'interrupcions i total de les mateixes, també s'inicialitza els valors dels contadors i es desbloqueja el port MUX per poder fer servir un dels contadors.

```C/C++
void IRAM_ATTR onTimer() {
portENTER_CRITICAL_ISR(&timerMux);
interruptCounter++;
portEXIT_CRITICAL_ISR(&timerMux);
}
```

Creació de la funció onTimer() que, bàsicament entra a un dels timers de la MUX que hem habilitat amb anterioritat, suma 1 al valor del contador d'interrupcions i surt del timer de la MUX.

```C/C++
void setup() {
Serial.begin(115200);
timer = timerBegin(0, 80, true);
timerAttachInterrupt(timer, &onTimer, true);
```

```
timerAlarmWrite(timer, 1000000, true);
timerAlarmEnable(timer);
}
```

A continuació creem la funció SetUp per inicialitzar el nostre timer. La nostra placa té 4 timers, en el nostre cas no hem especificat quin timer utilitzarem per tant utilitza el primer que trobi lliure.

a. timer begin (0, **80**, true)-> Aquesta funció s'utilitza per configurar el temporitzador. En el nostre cas dividim la senyal per 80)
b. timer AttachInterrupt ->Les interrupcions del temporitzador permeten realitzar una tasca a intervals de temps molt específics.
c. timer Alarm Write -> Aquesta funció s'utilitza per configurar el valor de l'alarma i la càrrega automàtica del temporitzador. (100000 polsos per generar l'esdeveniment.)
d. timer Alarm Enable -> ens marca quants polsos he de contar (100000 polsos)

```
C/C++
void loop() {
      if (interruptCounter > 0) {
      portENTER_CRITICAL(&timerMux);
      interruptCounter--;
      portEXIT_CRITICAL(&timerMux);
      totalInterruptCounter++;
      Serial.print("An interrupt as occurred. Total number: ");
      Serial.println(totalInterruptCounter);
      }
}
```

Inici del loop infinit de la ESP32:

a. Si el valor del contador d'interrupcions és positiu, primer es torna aquest valor a 0 i es suma 1 al valor del contador d'interrupcions totals.
b. S'imprimeix un missatge per pantalla informant de la interrupció i del número total d'interrupcions des de l'inici del programa.

En aquest cas podem veure que es fa servir el valor del interruptCounter com a un booleà

<u>Aplicacions reals:</u>

- Un sistema d'interrupcions en fàbriques o en màquines individuals assegura els sistemes de seguretat, i efectuar parades automàtiques quan això passi evita molts problemes. En una producció en cadena, és molt útil ja que es pot evitar que un defecte produit en una màquina concreta es propagui al llarg de tota la producció i pugui malmetre altres màquines o productes.
- Els sistemes d'interrupció en programació també son molt importants per poder fer una bona depuració del codi que s'està escrivint i evitar problemes i temps.

Utilitat de les interrupcions:

- Mesurar senyals a intervals amb la mateixa freqüència de mostreig
- Calcular temps entre esdeveniments
- Enviar senyals de freqüències determinades
- Comprovar periòdicament dades
- I moltes altres aplicacions relacionades.