Báo cáo thực hành

Chương 1: Lập trình nhúng

Học phần: Phát triển ứng dụng trên thiết bị di động (KC359), HK2 – NH 2025-2026

Mã nhóm học phần: KC359001

Nhóm 9, danh sách nhóm:

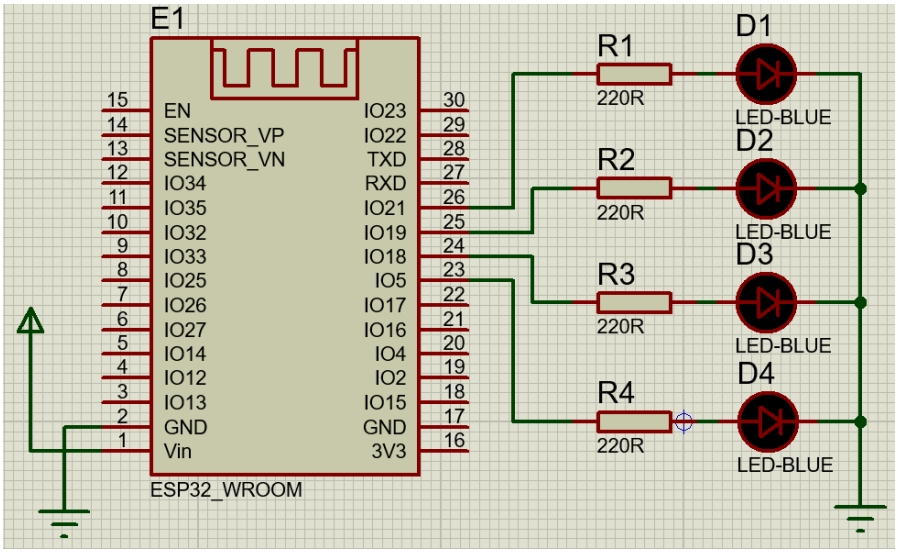
STT	Họ và Tên	MSSV	Mức độ tham gia
1	Trần Nguyên Hiền	B2305223	100%
2	Vưu Quang Nhân	B2305247	100%
3	Lê Thanh Toàn	B2305266	100%
4	Đặng Văn Til	B2305264	100%
5	Trương Hoàng Long	B2305238	100%

Nội dung thực hành:

Câu 1.1: Giải thích ngắn gọn hoạt động của hàm *loop*.

Trong hàm loop() Led sáng 500ms, tắt 500ms và lặp lại liên tục, tạo hiệu ứng chớp tắt với tần số 1Hz.

Câu 1.2: Kết nối 4 LED đơn vào 4 GPIO của ESP32 (*Sinh viên tự chọn GPIO kết nối LED*), vẽ sơ đồ mạch vào báo cáo (sử dụng phần mềm).



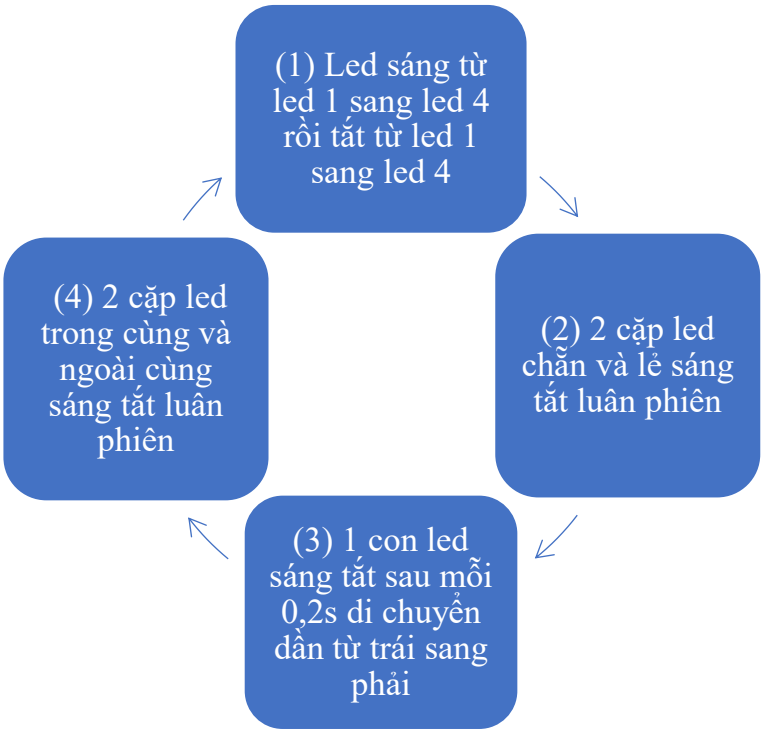
Câu 1.3: Viết chương trình chớp tắt tất cả các LED vừa kết nối với ESP32, tốc độ chớp tắt 5 Hz. (Lưu ý chỉ trình bày nội dung hàm loop vào báo cáo).

```
int led[] ={21,19,18,5};

void loop() {
for(int i = 0; i < 4; i++){
 digitalWrite(led[i], HIGH);
 delay(100);
 digitalWrite(led[i], LOW);
 delay(100);
}
}
```

Giải thích: Tần số 5Hz thì chu kỳ là 0.2s = 200ms nên mỗi trạng thái sáng hoặc tắt sẽ chiếm nửa chu kỳ là 100ms.

Câu 1.4: Người học tự đề xuất 4 hiệu ứng trên các LED vừa kết nối (ngoại trừ chớp tắt), viết chương trình và kiểm chứng kết quả. Các hiệu ứng phải được viết trên cùng một chương trình. Hãy trình bày nội dung hàm loop và một vài hình ảnh minh chứng kết quả vào báo cáo.

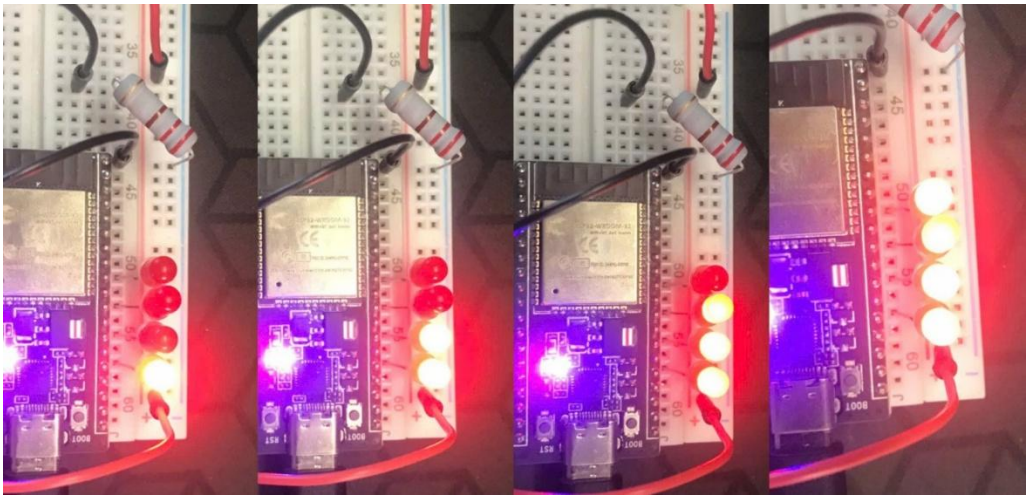


Viết chương trình:

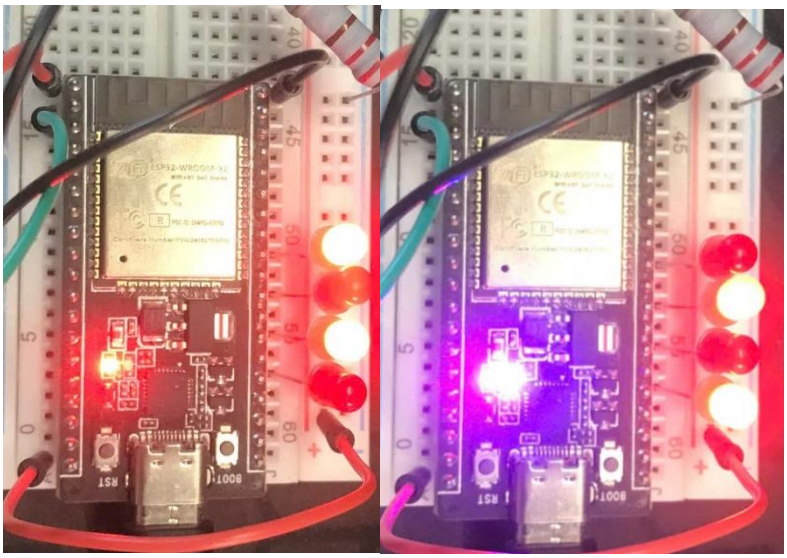
```
int led[] = {21,19,18,5};
void setup() {
for(int i = 0; i < 4; i++){
 pinMode(led[i],OUTPUT);
 }
}
void loop() {
//---------- Hieu Ung 1 ----------//
  //led sang tu led1 sang led4 roi tat tu led 1 sang led 4
for(int i = 0; i < 4; i++){
 digitalWrite(led[i],1);
 delay(500);
}
 //led tat tu led1 sang led4
for(int i = 0;i < 4; i++){
 digitalWrite(led[i],0);
 delay(500);
}
//---------- Hieu Ung 2 ----------//
 // 2 cap led chan va le sang tat luan phien
for(int i = 0; i < 5 ;i++){
  digitalWrite(led[0],1);
  digitalWrite(led[2],1);
  delay(200);
  digitalWrite(led[0],0);
  digitalWrite(led[2],0);
  digitalWrite(led[1],1);
  digitalWrite(led[3],1);
  delay(200);
  digitalWrite(led[1],0);
  digitalWrite(led[3],0);
  delay(200);
}
//---------- Hieu Ung 3 ----------//
 // 1 con led sang tat 0,2s di chuyen dan tu trai sang phai
for(int i = 0; i < 4; i++) {
 digitalWrite(led[i], 1);
 delay(200);
 digitalWrite(led[i],0);
 delay(200);
}
```

```
//---------- Hieu Ung 4 ----------//
 //2 cap led trong cung va ngoai cung sang tat luan phien
  for(int i = 0; i < 4; i++)
  {
    digitalWrite(led[i], 1);
    digitalWrite(led[3-i], 1);
    delay(100);
    digitalWrite(led[i], 0);
    digitalWrite(led[3-i], 0);
  }
}
```

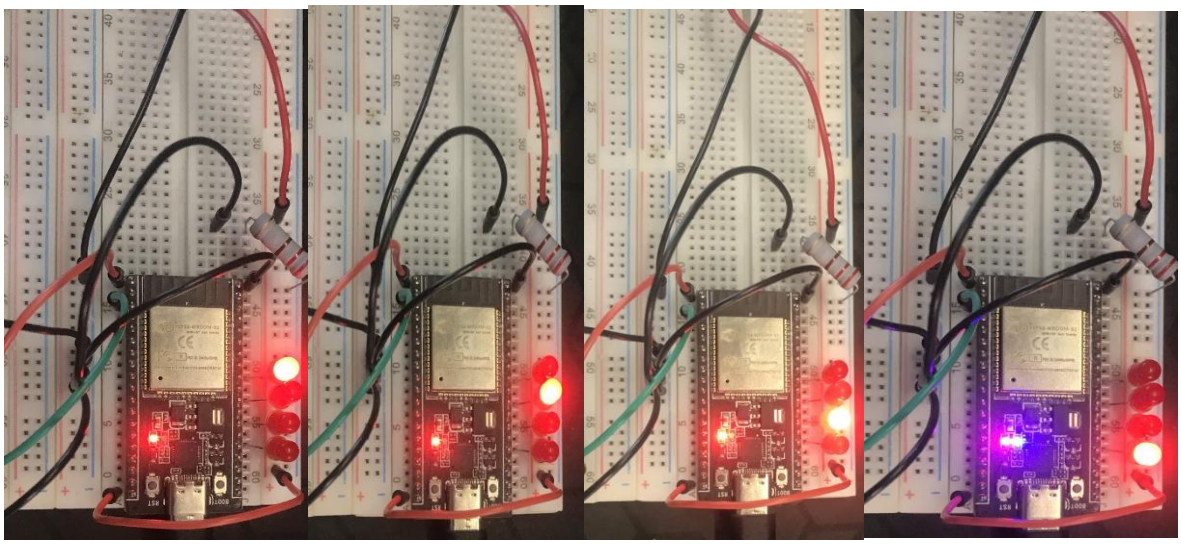
Hiệu ứng 1: Led sáng từ led 1 sang led 4 và tắt từ led 4 sang led 1.



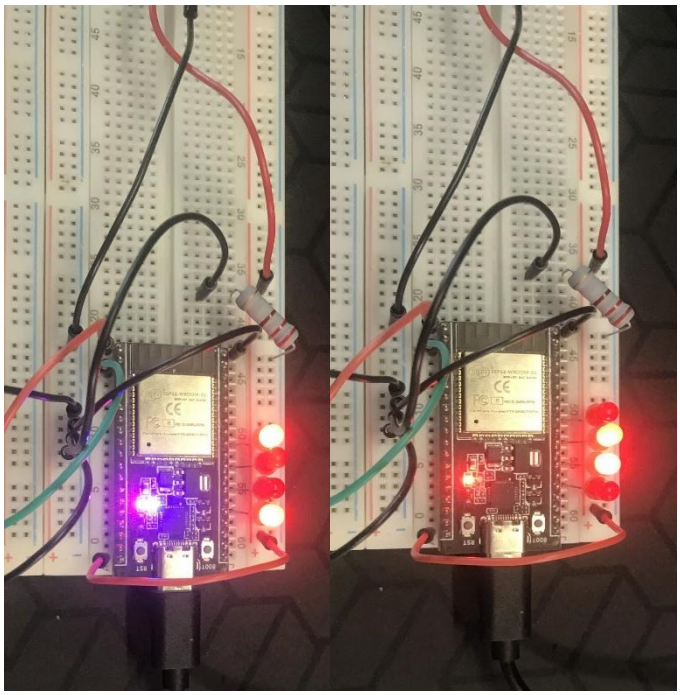
Hiệu ứng 2: 2 cặp led chẵn và lẻ sáng tắt luân phiên.



Hiệu ứng 3: 1 con led sáng tắt di chuyển từ trái sang phải.



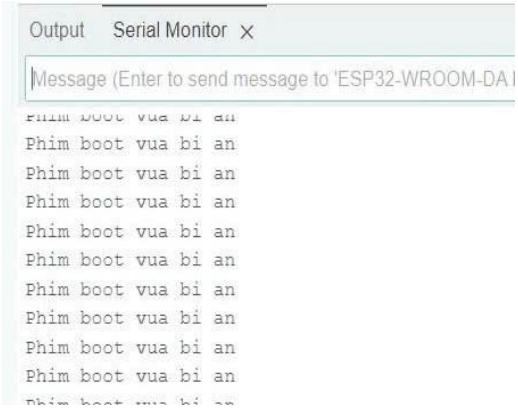
Hiệu ứng 4: 2 cặp led trong cùng và ngoài cùng sáng tắt luân phiên.



Câu 1.5: Viết chương trình liên tục in trạng thái của phím Boot lên cửa sổ Serial monitor. Nếu phím bị ấn hãy in ra "Phim boot vua bi an", bỏ qua nếu không bị ấn. Hãy trình bày nội dung hàm loop và kết quả in trong cửa sổ Serial Monitor vào báo cáo. Lưu ý, câu này không yêu cầu người học áp dụng kỹ thuật chống dội phím nên mỗi lần ấn phím số lượng chuỗi được in ra là không thể kiểm soát được.

- Hình bên dưới là nội dung hàm loop và thông báo trên cửa sổ Serial Monitor.

```
#define button 0
void setup() {
  pinMode(button,INPUT);
  Serial.begin(115200);
}
void loop() {
  if(!digitalRead(button)){
    Serial.println("Phim boot vua bi an");
  }
}
```



***Câu 1.6*:** Hiệu chỉnh chương trình trong Câu 5 để thêm chức năng đếm số lần ấn phím (mỗi lần ấn chỉ in ra Serial monitor một lần duy nhất, hay phải chống dội). Kết quả hiển thị trên cửa sổ Serial monitor được trình như sau:

Phim boot vua bi an lan 01

Phim boot vua bi an lan 02

...

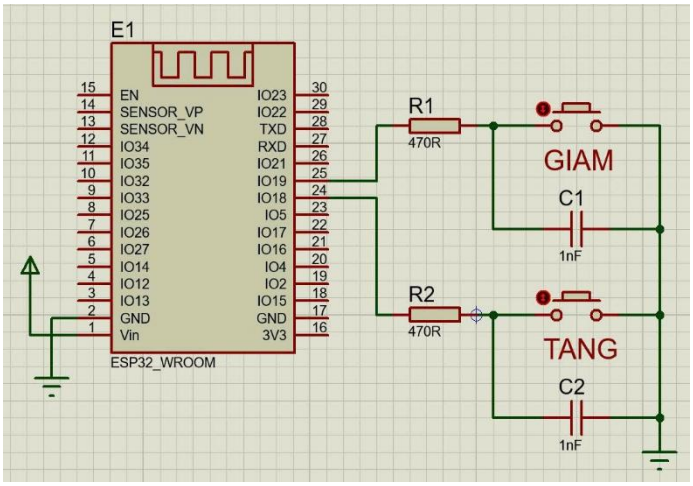
Người học hãy trình bày nội dung của hàm loop và kết quả in trong cửa sổ Serial Moniter vào báo cáo.

Chương trình đã áp dụng kỹ thuật chống dội đơn giản bằng cách dùng lệnh ***while (digitalRead(button)==0)*** chờ đến khi người dùng thả phím ra. Một biến đếm **a** được sử dụng để theo dõi số lần phím được nhấn và hiển thị thứ tự tương ứng. Mỗi lần nhấn phím Boot, một dòng thông báo được in lên Serial Monitor, thể hiện rõ số thứ tự lần nhấn phím.. Nội dung cụ thể của hàm **loop()** như sau:

```
#define button 0
int a=0;
 void setup() {
pinMode(button,INPUT);
Serial.begin(115200);
}
void loop() {
if(!digitalRead(button)){
a++;
 Serial.print("Phim boot vua bi an lan ");
Serial.println(a);
while (digitalRead(button)==0) {
delay(1); }
}
 }
```

```
Output  Serial Monitor ×
Message (Enter to send message to 'ESP32-WRO
Phim boot vua bi an lan 1
Phim boot vua bi an lan 2
Phim boot vua bi an lan 3
Phim boot vua bi an lan 4
Phim boot vua bi an lan 5
Phim boot vua bi an lan 6
Phim boot vua bi an lan 7
Phim boot vua bi an lan 8
Phim boot vua bi an lan 9
```

***Câu 1.7*:** Hãy lắp thêm hai phím ấn vào mạch, người học tự chọn GPIO kết nối, hãy vẽ sơ đồ nguyên lý (sử dụng phần mềm chuyên dụng) và ghi nhận vào báo cáo.



Câu 1.8*:** Viết chương trình hiển thị trạng thái của hai phím ra hai LED đơn (Sử dụng hai trong bốn LED ở ***Câu 1.2 phần 1.1, cần trình bày rõ sử dụng LED tại GPIO nào vào báo cáo). Người học hãy ghi nhận nội dung vòng loop và một số hình ảnh minh chứng kết quả vào báo cáo.

Gợi ý: Tương tự chương trình mẫu nhưng áp dụng hai lần cho cho hai phím vừa lắp ráp thêm.

Khi ta dùng tay nhấn một trong hai phím thì Led tương ứng với một trong hai phim đó sẽ bị tắt đi trong thời gian mà ta đang nhấn phím và khi thả ra led sẽ sáng trở lại như lúc đầu . Đồng thời, chương trình đếm số lần nhấn của từng phím và hiển thị thông báo trên cửa sổ Serial Monitor.

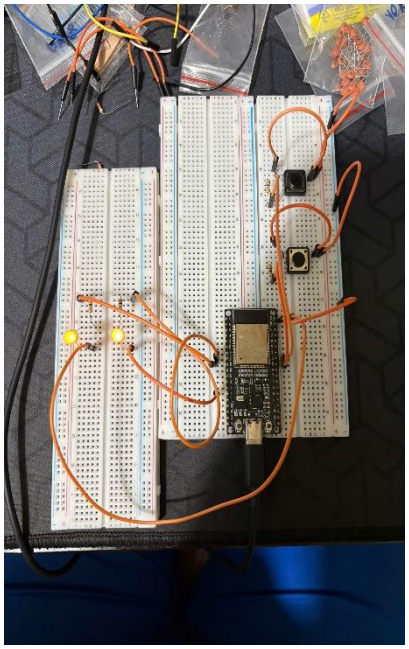
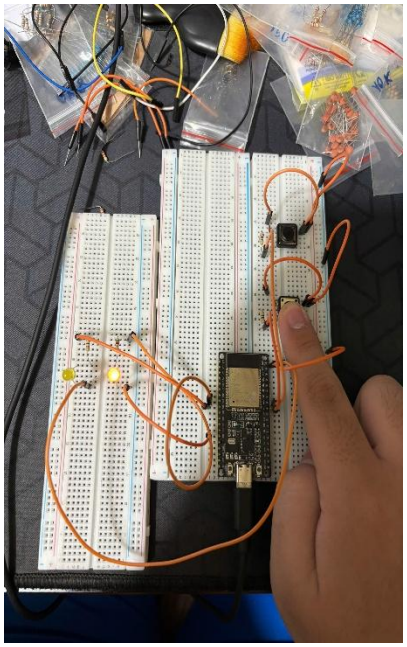
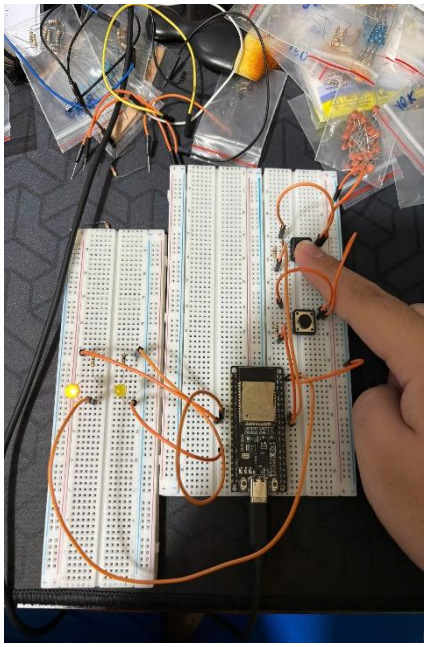
+ Phím một kết nối với chân G18 , điều khiển Led tại chân G25.

+ Phím hai kết nối với chân G19, điều khiển Led tại chân G33.

-Dưới đây là hình nội dụng hàm loop và thông báo trên cửa sổ Serial Monitor .

```
void loop() {
bool b1 = (digitalRead(BTN1) == LOW );
bool b2 = (digitalRead(BTN2) == LOW );
if (b1) {
  digitalWrite(LED1, LOW);
  count1++;
  Serial.print("Phim thu nhat vua duoc an lan");
  if ( count1 < 10 )
  Serial.print("0");
  Serial.println(count1);
  while ( digitalRead(BTN1) == LOW)
  delay(1); }
else{
  digitalWrite(LED1,HIGH);
  }
  if (b2) {
  digitalWrite(LED2, LOW);
  count2++;
  Serial.print("Phim thu hai vua duoc an lan");
  if ( count2 < 10 )
  Serial.print("0");
  Serial.println(count2);
  while ( digitalRead(BTN2) == LOW)
  delay(1); }
else{
  digitalWrite(LED2,HIGH);
  }
}
```

-Một số hình ảnh thực tế :

(đường dẫn video thực tế: https://youtube.com/shorts/ToFAK3GQpl0?si=RH_dyByPHWkVYsc6)

Câu 1.9: Viết chương trình thực hiện chức năng tăng/giảm giá trị của một biến đếm (ví dụ biến count) sử dụng hai phím vừa lắp ráp thêm, CHỈ in kết quả ra cửa sổ Serial monitor khi có bất kỳ một phím được ấn, mỗi lần ấn CHỈ tăng/giảm đúng một đơn vị. Các GPIO sử dụng giao tiếp phím do người học tự chọn. Hãy ghi nhận nội dung vòng loop và hình chụp kết quả trong cửa sổ Serial monitor vào báo cáo.

Gợi ý: Áp dụng thuật toán chống dội phím trong Câu 1.6 hai lần chc hai phím mới và viết lại mã lệnh công việc cho phù hợp yêu cầu. Nếu người học sử dụng thuật toán khác vui lòng diễn giải nó trước khi áp dụng.

```
void loop() {
  // Tang len 1 don vi
  if (digitalRead(18) == 0) {
    count++;
    Serial.print("Da an tang. Gia tri: ");
    Serial.println(count);

    while (digitalRead(18) == 0) {
      delay(1);
    }
  }
```

```
// Giam xuong 1 don vi (chi giam den count=0)
  if (digitalRead(19) == 0) {
    if(count>0){
      count--;
      Serial.print("Da nhan giam. Gia tri: ");
      Serial.println(count);
    }else Serial.println("Khong the giam duoc nua!");
    while (digitalRead(19) == 0) {
      delay(1);
    }
  }
}
```

```
Output  Serial Monitor ×
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM5')
23:17:46.039 -> Khong the giam duoc nua!
23:17:48.578 -> Da an tang. Gia tri: 1
23:17:48.871 -> Da an tang. Gia tri: 2
23:17:49.198 -> Da an tang. Gia tri: 3
23:17:49.591 -> Da an tang. Gia tri: 4
23:17:50.464 -> Da nhan giam. Gia tri: 3
23:17:50.791 -> Da nhan giam. Gia tri: 2
23:17:51.183 -> Da nhan giam. Gia tri: 1
23:17:51.681 -> Da nhan giam. Gia tri: 0
23:17:52.591 -> Khong the giam duoc nua!
```

Câu 1.10: Hai biến __cntR, __cntP có vai trò gì? Người đọc hay tăng giảm giá trị của hai biến này (mặc định bằng 5), đánh giá hiệu quả chống dội xác định giá trị phù hợp nhất cho hai biến này cho phần cứng hiện tại của nhóm và ghi nhận váo báo cáo. Nếu hai giá trị này có giá trị quá lớn thì điều gì xảy ra?

-Vai trò của hai biến __cntP và __cntR:

+__cntP : Là biến đếm số lần liên tiếp nút được nhả (mức 1).

+__cntR: Là biến đếm số lần liên tiếp nút được nhấn (mức 0).

+ Khi __cntP đạt đến giá trị DEBOUNCE_CONST 5 , nghĩa là nút nhấn đã được nhả ra ổn định (không có hiện tượng bị nhiễu). Ngược lại khi __cntR đạt đến giá trị DEBOUNCE_CONST 5 có nghĩa là nút nhấn đã được nhấn ổn định (không có hiện tượng bị nhiễu).

-Để đánh giá hiệu quả chống dội khi ta tăng giảm giá trị của hai biến __cntP và __cntR :

+Khi ta giảm giá trị của 2 biến này xuống về 1 hoặc 2 thì hệ thống phản hồi nhanh nhưng dễ bị xuất hiện nhiễu , đếm sai số lần nhấn.

+Khi ta tăng giá trị của 2 biến này lên 10 đến 15 thì hệ thống sẽ ổn định nhưng ta phải nhấn rồi giữ nút chứ không thể buôn ra ngay.

-Để có thể xác định được hiệu quả chống dội của chương trình này với phần cứng hiện tại của nhóm chúng em đang có thì giá trị phù hợp nhất là từ 5 đến khoảng 6, 7 vì với khoảng giá trị này có thể đạt được hiệu quả chống nhiễu cũng như đảm bảo tốc độ phản hồi không quá lâu.

-Nếu ta setup giá trị DEBOUNCE_CONST quá lớn thì nút nhấn sẽ phản hồi rất chậm, phải giữ lâu thì hệ thống mới nhận tín hiệu , hiệu quả chống nhiễu / chống dội tốt nhưng giá trị thực tế mang lại cho người dùng không cao.

Câu 1.11: Người học hãy hiệu chỉnh chương trình Lab1_03.ino đề chống dội cho 3 phím ấn (Phím trên mạch và hai phím lắp thêm ***Câu 1.7***. Lưu ý cần đọc kỹ hướng dẫn cách viết hàm key_read để lập trình trường hợp này cho phù hợp. Người học tự đề xuất chức năng của các phím ấn). Người học hãy trình bày nội dung của hàm key_read, loop và ghi nhận một số hình ảnh kết quả trong cửa số Serial monitor vào báo cáo.

-Thiết lập chức năng các nụt như sau:

+ Phím 1 : Khi nhấn phím 1 tăng giá trị biến count lên 1 đơn vị.

+ Phím 2 : Khi nhấn phím 2 giảm giá trị biến count đi 2 đơn vị.

+ Phím 3 : Khi nhấn phím 3 đặt giá trị biến count thành 36.

-Dưới đây là hình của số Serial Monitor và nội dung hàm key_read và hàm loop sau khi hiệu chỉnh:

```
void loop() {
  key_read_debounce();

  if (pr&&flag) {
    flag = 0;
    if (key == 1) {        // Phim 1
      count++;
      Serial.println("BUTTON1 pressed -> count = " + String(count));
    } else if (key == 2) { // Phim 2
      count=count-2;
      Serial.println("BUTTON2 pressed -> count = " + String(count));
    } else if (key == 3) { // Phim 3
      count = 36;
      Serial.println("BUTTON3 pressed -> reset count = 36");
    }
  }
}
```

```
int8_t key_read() {
  if (digitalRead(BUTTON1) == LOW) return 1;
  if (digitalRead(BUTTON2) == LOW) return 2;
  if (digitalRead(BUTTON3) == LOW) return 3;
  return 0;
}
```

```
Chuong trinh bat dau...
BUTTON1 pressed -> count = 1
BUTTON1 pressed -> count = 2
BUTTON1 pressed -> count = 3
BUTTON1 pressed -> count = 4
BUTTON2 pressed -> count = 2
BUTTON2 pressed -> count = 0
BUTTON2 pressed -> count = -2
BUTTON2 pressed -> count = -4
BUTTON2 pressed -> count = -6
BUTTON3 pressed -> reset count = 36
BUTTON3 pressed -> reset count = 36
BUTTON1 pressed -> count = 37
BUTTON1 pressed -> count = 38
BUTTON1 pressed -> count = 39
BUTTON1 pressed -> count = 40
BUTTON1 pressed -> count = 41
BUTTON2 pressed -> count = 39
BUTTON2 pressed -> count = 37
BUTTON3 pressed -> reset count = 36
```

Câu 1.12: Viết thêm lệnh sử dụng hàm analogRead để đọc giá trị chuyển đổi ADC tại VOUT, in đồng thời hai giá trị (Giá trị ADC và điện áp) ra sơ đồ sánh. Hiệu chỉnh điện áp từ 0 V đến 3,3 V, người học hãy lặp bằng một số cặp giá trị từ nhỏ đến lớn, nhận xét.

ADC	Vout
0	142mV
62mV	191mV
288mV	380mV
899mV	884mV
1233mV	1160mV
1607mV	1466mV
2029mV	1827mV
2722mV	2408mV
3152mV	2709mV
3829mV	3041mV
4095mV	3155mV

Nhận xét:

- Giá trị điện áp Vout tăng dần thì giá trị ADC cũng tăng tương ứng. Tại ADC = 0, Vout = 142mV.
- Khi đo ADC từ lệnh analogRead() thì không thể đọc giá trị khi vặn biến trở ở mức thấp nhất với điện áp khoảng <= 150mV. Nhưng khi đo ADC từ lệnh analogReadMilliVolts() thì khi vặn biến trở ở mức thấp nhất điện áp đo được Vout = 142mV.
- Khi dùng lệnh analogRead() thì chỉ đo số nguyên thô từ **0 đến 4095**. Còn khi dùng lệnh analogReadMilliVolts() giá trị điện áp được hiệu chỉnh có đơn vị đo là **mV**.

***Câu 1.13*:** Hãy hiệu chỉnh biến trở để V_{out} khoảng 0,5 V (đo bằng VOM - Volt-Ohm-Milliammeter), ghi nhận điện áp V_{out} (đo bằng VOM) và bằng ESP32 (đọc trên cửa sổ Serial monitor, sử dụng lại chương trình Lab1_04.ino) vào Bảng 1.5, lặp lại thí nghiệm 10 lần, mỗi lần cách nhau tối thiểu 30 giây.

Trong các bảng, e và MAE được tính bằng công thức (1), (2) tương ứng như sau:

$$|e| = |V_{out} - V_{ADC}|$$

$$MAE = \frac{\sum_{i=1}^{10} \sum |e_i|}{10}$$

Tiếp tục hiệu chỉnh Vout khoảng 1,5 V, hoàn thành Bảng 1.6.

Sau cùng, hãy hiệu chỉnh Vour khoảng 2,8 V và hoàn thành Bảng 1.7. Dựa vào các giá trị $|e|$ và MAE trong các bảng sau khi thí nghiệm, hãy nhận xét ngắn gọn về sai số đo của mô-đun ADC tích hợp trên ESP32 khi đo điện áp một chiều. Lập bảng tóm tắt kết quả thí nghiệm gồm giá trị trung bình, lớn nhất và nhỏ nhất của $|e|$ trong từng trường hợp thí nghiệm.

So sánh điện áp khi chỉnh biến trở khoảng 0.5V

STT	**V_{VOM}**	**V_{ADC}**	**|e|**
1	0.51V	0.46V	0.05
2	0.51V	0.43V	0.08
3	0.5V	0.45V	0.06
4	0.5V	0.47V	0.04
5	0.49V	0.5V	0.01
6	0.5V	0.49V	0.01
7	0.49V	0.5V	0.01
8	0.49V	0.51V	0.02
9	0.49V	0.45V	0.01
10	0.5V	0.51V	0.01
MAE			**0.3**

So sánh điện áp khi chỉnh biến trở khoảng 1.5V

STT	V_{VOM}	V_{ADC}	|e|
1	1.5V	1.5V	0
2	1.49V	1.5V	0.01
3	1.5V	1.5V	0
4	1.51V	1.51V	0
5	1.5V	1.51V	0.01
6	1.5V	1.5V	0
7	1.49V	1.5V	0.01
8	1.51V	1.5V	0.01
9	1.5V	1.51V	0.01
10	1.5V	1.5V	0
MAE			**0.05**

So sánh điện áp khi chỉnh biến trở khoảng 2.8V

STT	V_{VOM}	V_{ADC}	|e|
1	2.85V	2.83V	0.02
2	2.88V	2.82V	0.06
3	2.89V	2.82V	0.07
4	2.86V	2.82V	0.04
5	2.89V	2.83V	0.06
6	2.87V	2.82V	0.05
7	2.88V	2.81V	0.07
8	2.89V	2.82V	0.07
9	2.91V	2.82V	0.9
10	2.91V	2.81V	0.1
MAE			**0.63**

Nhận xét: Nếu điện áp cao (2.8V) hoặc thấp (0.5V) sẽ dẫn đến sai số của modun ADC trên ESP32. Khoảng điện áp hoạt động tốt nhất của modun ADC trên ESP32 là 1.5V. Sai số nhiều nhất với điện áp 2.8V

Bảng tóm tắt kết quả thí nghiệm:

Điện áp	Giá trị trung bình của |e|	Giá trị lớn nhất của |e|	Giá trị nhỏ nhất của |e|
V=0.5	0.03	0.08	0.01
V=1.5	0.097	0.01	0
V=2.8	0.144	0.1	0.02