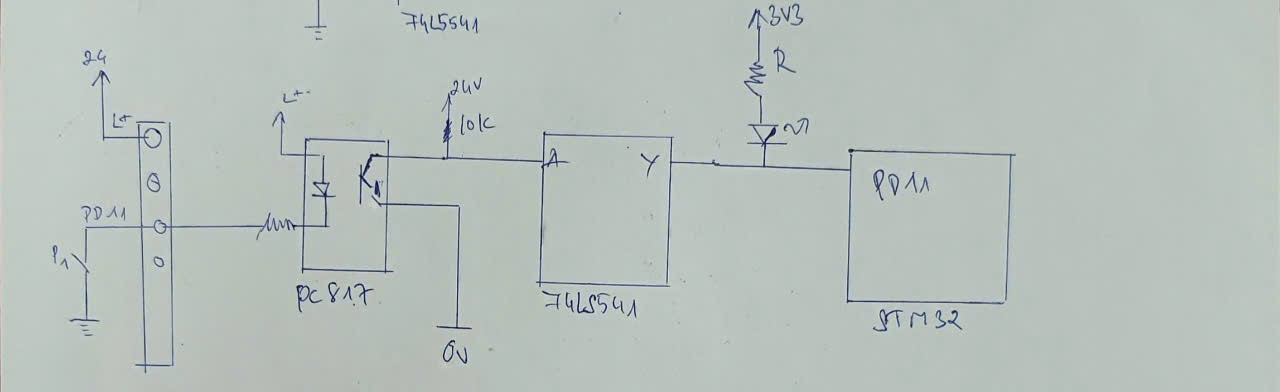
|  |  |
| --- | --- |
| TRƯỜNG ĐH SƯ PHẠM KỸ THUẬT TPHCM  **KHOA CƠ KHÍ CHẾ TẠO MÁY**  **BỘ MÔN CƠ ĐIỆN TỬ**  **Họ và tên:** Võ Hồng Quân  **Mã số SV:** 22134012 | **BÁO CÁO**  **THỰC TẬP LẬP TRÌNH VI XỬ LÝ**  Ngày:  Tuần học: 01  Nội dung: **GPIO và State Machine** |

1. **Khảo sát sơ đồ nguyên lý:**

* Sinh viên vẽ sơ đồ nguyên lý khối **Digital Input** và giải thích

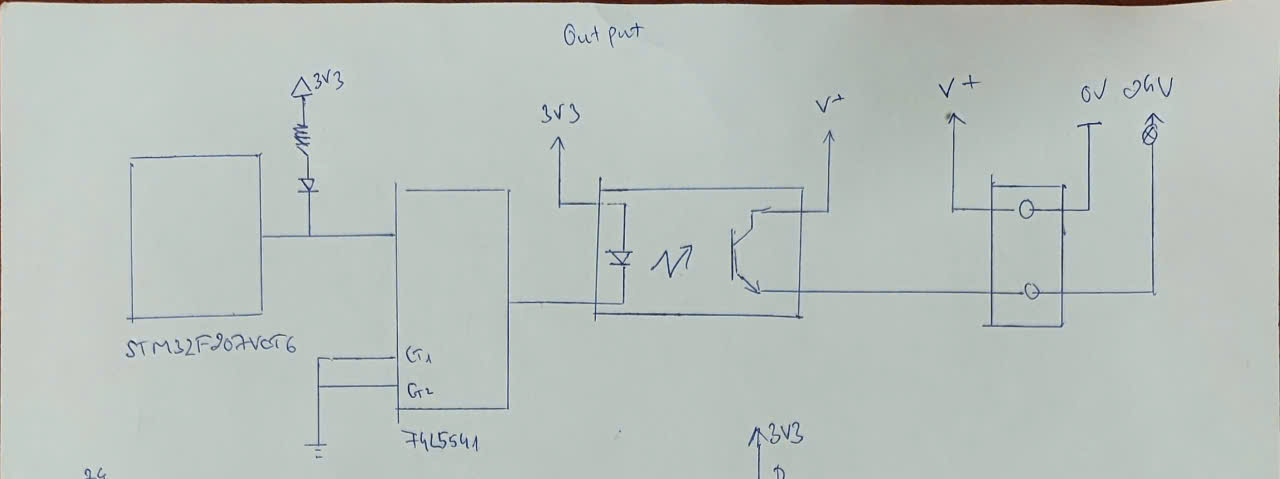


Sử dụng optocoupler: Để cách ly tín hiệu 24V với vi điều khiển STM32 (hoạt động ở mức 3.3V).

Mạch có bộ đệm (74LS541): Giúp ổn định tín hiệu đầu vào trước khi đưa vào vi điều khiển.

Điện trở pull-up: Được sử dụng để kéo tín hiệu lên mức cao khi optocoupler tắt.

* Sinh viên vẽ sơ đồ nguyên lý khối **Digital Output** và giải thích



Sử dụng optocoupler để cách ly: Giúp bảo vệ vi điều khiển khi điều khiển tải điện áp cao.

Dùng transistor để khuếch đại dòng: Cần thiết để điều khiển tải tiêu thụ dòng lớn hơn.

1. **Bảng kết nối input – output**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên phần cứng | Vị trí chân vi điều khiển tương ứng | Mức tín hiệu ở trạng thái tích cực |
| 1 | Đèn D1 | PA8 | Thấp |
| 2 | Đèn D2 | PC9 | Thấp |
| 3 | Nút nhấn P1 | PE13 | Thấp |
| 4 | Nút nhấn P2 | PE12 | Thấp |
| 5 | Nút nhấn P3 | PE10 | Thấp |
| 6 | Nút nhấn P4 | PE8 | Thấp |
| 7 | Công tắt gạt vị trí 1 | PE7 | Thấp |
| 8 | Công tắt gạt vị trí 2 | PB2 | Thấp |
| 9 | Công tắt gạt vị trí 3 | PC4 | Thấp |

1. **Lập trình**
   1. **Cài đặt phần mềm và thư viện:**

* Để lập trình STM32 thì người sử dụng cần cài đặt các phần mềm sau:

|  |  |  |
| --- | --- | --- |
| STT | Phần mềm | Chức năng |
| 1 | STM32 CubeMX | Tạo dự án lập trình bằng giao diện GUI  Thư viện HAL |
| 2 | Keil C for ARM | Môi trường soạn thảo mã nguồn chương trình và biên dịch C cho vi điều khiển ARM. |
| 3 | STM32 ST-LINK Utility | Tải chương trình đã biên dịch (.hex) từ máy tính xuống vi điều khiển |

* 1. **Tạo thư viện myGPIO**
* Viết thư. viện myGPIO.h và myGPIO c với các hàm và macro sau đây:
  + Định nghĩa các trạng thái đèn D\_ON, D\_OFF
  + Định nghĩa các trạng thái nút nhấn BUTTON\_ACTIVE, BUTTON\_INACTIVE
  + Định nghĩa các trạng thái công tắc gạt SW1\_ACTIVE, SW2\_ACTIVE, SW3\_ACTIVE
  + Hàm void **Light\_cmd**(uint8\_t **d**, uint8\_t **status**) để bật tắt đèn D1, D2:
    - Light\_cmd(D1, D\_ON) : Bật đèn D1 sáng
    - Light\_cmd(D1, D\_OFF) : Tắt đèn D1
    - Light\_cmd(D2, D\_ON) : Bật đèn D2 sáng
    - Light\_cmd(D2, D\_OFF) : Tắt đèn D2
  + Hàm GPIO\_PinState **PushButton\_read**(uint8\_t **p**) để đọc trạng thái của nút nhấn p. Ví dụ:
    - **PushButton\_read** (P1) : Đọc về trạng thái của nút nhấn P1. Nếu đang được nhấn thì trả về giá trị BUTTON\_ACTIVE; nếu chưa được nhấn thì trả về giá trị BUTTON\_INACTIVE.
  + Hàm uint8\_t **Switch\_read**(void) để đọc vị trí của switch. Ví dụ:
    - **Switch\_read** (void) : Đọc về trạng thái của Switch. Nếu SW đang gạt về vị trí 1 (bên trái) thì hàm trả về SW1\_ACTIVE. Nếu SW đang gạt về vị trí 2 (giữa) thì hàm trả về SW2\_ACTIVE. Nếu SW đang gạt về vị trí 3 (bên phải) thì hàm trả về SW3\_ACTIVE.

Mã nguồn chương trình (chỉ copy nội dung đã thêm hoặc chỉnh sửa từ code dự án)

* myGPIO.h:

|  |
| --- |
| #ifndef MYGPIO\_H  #define MYGPIO\_H  #include "main.h"  #include <stdint.h>  // Định nghĩa trạng thái LED  #define D\_ON GPIO\_PIN\_RESET // Đèn bật (0)  #define D\_OFF GPIO\_PIN\_SET // Đèn tắt (1)  // Định nghĩa trạng thái nút nhấn  #define P\_ACTIVE GPIO\_PIN\_RESET // Nút nhấn được nhấn  #define P\_INACTIVE GPIO\_PIN\_SET // Nút nhấn không được nhấn  // Định nghĩa LED  #define D1 1  #define D2 2  // Định nghĩa trạng thái công tắc gạt  #define SW\_ON GPIO\_PIN\_RESET  #define SW\_INACTIVE GPIO\_PIN\_SET  #define SW1\_ACTIVE 1  #define SW2\_ACTIVE 2  #define SW3\_ACTIVE 3  // Định nghĩa nút nhấn  #define P1 1  #define P2 2  #define P3 3  #define P4 4  // Biến trạng thái toàn cục (extern để tránh lỗi multiple definition)  extern uint8\_t stateSW;  extern uint8\_t stateP1;  extern uint8\_t stateP2;  extern uint16\_t DelayCnt;  extern GPIO\_PinState P1\_STATUS;  extern GPIO\_PinState P2\_STATUS;  extern GPIO\_PinState P3\_STATUS;  extern GPIO\_PinState P4\_STATUS;  // Nguyên mẫu hàm điều khiển GPIO  void Light\_cmd(uint8\_t d, uint8\_t status);  uint8\_t PushButton\_read(uint8\_t p);  uint8\_t Switch\_read(void);  void Baitap3\_3(uint8\_t P1\_STATUS, uint8\_t P2\_STATUS);  void Baitap3\_4(uint16\_t SW\_STATUS, uint8\_t P1\_Status);  void Baitap3\_5(uint8\_t P1\_Status, uint8\_t P2\_Status);  #endif // MYGPIO\_H |

* myGPIO.c:

|  |
| --- |
| #include "main.h"  #include "myGPIO.h"  // Biến trạng thái  GPIO\_PinState P1\_STATUS;  GPIO\_PinState P2\_STATUS;  GPIO\_PinState P3\_STATUS;  GPIO\_PinState P4\_STATUS;  //======== Light\_cmd (Bật/tắt đèn) ========//  void Light\_cmd(uint8\_t d, uint8\_t status) {  switch (d) {  case D1:  HAL\_GPIO\_WritePin(D1\_GPIO\_Port, D1\_Pin, status);  break;  case D2:  HAL\_GPIO\_WritePin(D2\_GPIO\_Port, D2\_Pin, status);  break;  default:  return; // Nếu giá trị không hợp lệ, không làm gì cả  }  }  //======== PushButton\_read (Đọc trạng thái nút nhấn) ========//  uint8\_t PushButton\_read(uint8\_t p) {  switch (p) {  case P1:  return HAL\_GPIO\_ReadPin(P1\_GPIO\_Port, P1\_Pin);  case P2:  return HAL\_GPIO\_ReadPin(P2\_GPIO\_Port, P2\_Pin);  case P3:  return HAL\_GPIO\_ReadPin(P3\_GPIO\_Port, P3\_Pin);  case P4:  return HAL\_GPIO\_ReadPin(P4\_GPIO\_Port, P4\_Pin);  default:  return P\_INACTIVE; // Trả về trạng thái không nhấn nếu không hợp lệ  }  }  //======== Switch\_read (Đọc trạng thái công tắc gạt) ========//  uint8\_t Switch\_read(void) {  if (HAL\_GPIO\_ReadPin(SW\_1\_GPIO\_Port, SW\_1\_Pin) == SW\_ON) {  return SW1\_ACTIVE;  } else if (HAL\_GPIO\_ReadPin(SW\_2\_GPIO\_Port, SW\_2\_Pin) == SW\_ON) {  return SW2\_ACTIVE;  } else if (HAL\_GPIO\_ReadPin(SW\_3\_GPIO\_Port, SW\_3\_Pin) == SW\_ON) {  return SW3\_ACTIVE;  } else {  return SW\_INACTIVE;  }  } |

* 1. **Lập trình ứng dụng**
* Khi mới mở điện: Đèn D1, D2 tắt;
* Mỗi khi nhấn nút **P1** đèn D1 đảo trạng thái (vừa nhấn vào đã tác động):
  + Trạng thái 1: Đèn sáng; Trạng thái 2: Đèn tắt. Ví dụ: Nhấn P1 lần 1 thì đèn D1 sáng; nhấn P1 lần 2 thì đèn D1 tắt; nhấn P1 lần 3 thì đèn D1 sáng.v.v…
* Mỗi khi nhấn nút **P2** thì đèn D2 đảo trạng thái (nhấn vào thả tay ra mới tác động):
  + Trạng thái 1: Đèn chớp tắt chu kỳ 2s; Trạng thái 2: Đèn tắt.

Mã nguồn chương trình (chỉ copy nội dung đã thêm hoặc chỉnh sửa từ code dự án)

|  |
| --- |
| /\* Includes ------------------------------------------------------------------\*/  #include "main.h"  #include "myGPIO.h"  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  /\* Biến trạng thái \*/  uint8\_t stateP1 = 0;  uint8\_t stateP2 = 0;  uint32\_t lastToggleTime = 0;  /\* Hàm xử lý logic P1 và P2 \*/  void Baitap3\_3(uint8\_t P1\_STATUS, uint8\_t P2\_STATUS) {  // === Xử lý P1: Đảo trạng thái D1 ===  if (P1\_STATUS == P\_ACTIVE) { // Nhấn nút P1  while (PushButton\_read(P1) == P\_ACTIVE); // Chờ nhả nút (tránh lặp)  stateP1 = !stateP1; // Đảo trạng thái  Light\_cmd(D1, stateP1 ? D\_ON : D\_OFF); // Cập nhật LED  }  // === Xử lý P2: Chớp tắt D2 chu kỳ 2s ===  if (P2\_STATUS == P\_ACTIVE) { // Nhấn nút P2 để bật/tắt chế độ chớp  while (PushButton\_read(P2) == P\_ACTIVE); // Chờ nhả nút  stateP2 = !stateP2; // Đảo trạng thái  if (!stateP2) Light\_cmd(D2, D\_OFF); // Nếu tắt chế độ, D2 tắt luôn  }  // Nếu đang bật chế độ chớp  if (stateP2) {  uint32\_t currentTime = HAL\_GetTick();  if (currentTime - lastToggleTime >= 1000) { // Mỗi 1s đổi trạng thái  lastToggleTime = currentTime;  HAL\_GPIO\_TogglePin(D2\_GPIO\_Port, D2\_Pin);  }  }  }  int main(void) {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  while (1)  {  P1\_STATUS = PushButton\_read(P1);  P2\_STATUS = PushButton\_read(P2);  Baitap3\_3(P1\_STATUS, P2\_STATUS);  }  } |

* 1. **Lập trình ứng dụng**
* Khi mới mở điện: Đèn D1, D2 tắt;
* Công tắt gạt chuyển trạng thái của đèn D1 như sau:
  + SW1\_ACTIVE: Trạng thái 1 đèn D1 tắt.
  + SW2\_ACTIVE: Trạng thái 2 đèn D1 sáng tắt chu kỳ 2s.
  + SW3\_ACTIVE: Trạng thái 3 đèn D1 sáng liên tục.
* Mỗi khi nhấn nút nhấn P1, đèn D2 chuyển 3 trạng thái (lặp lại):
  + Nhấn lần 1: Trạng thái 1 đèn D2 tắt.
  + Nhấn lần 2: Trạng thái 2 đèn D2 sáng tắt chu kỳ 2s.
  + Nhấn lần 3: Trạng thái 3 đèn D2 sáng liên tục.
* Mã nguồn chương trình (chỉ copy nội dung đã thêm hoặc chỉnh sửa từ code dự án)

|  |
| --- |
| void Baitap3\_4(uint16\_t SW\_STATUS, uint8\_t P1\_Status) {  // Kiểm tra trạng thái của công tắc gạt SW  switch (stateSW) {  case 0:  Light\_cmd(D1, D\_OFF);  Light\_cmd(D2, D\_OFF);  if (SW\_STATUS == SW1\_ACTIVE) stateSW = SW1\_ACTIVE;  else if (SW\_STATUS == SW2\_ACTIVE) stateSW = SW2\_ACTIVE;  else if (SW\_STATUS == SW3\_ACTIVE) stateSW = SW3\_ACTIVE;  break;  case SW1\_ACTIVE:  Light\_cmd(D1, D\_OFF);  if (SW\_STATUS == SW2\_ACTIVE) {  stateSW = SW2\_ACTIVE;  DelayCnt = 0;  }  break;  case SW2\_ACTIVE:  if (HAL\_GetTick() - DelayCnt >= 1000) {  Light\_cmd(D1, D\_ON);  } else if (HAL\_GetTick() - DelayCnt >= 2000) {  Light\_cmd(D1, D\_OFF);  DelayCnt = HAL\_GetTick(); // Reset lại bộ đếm  }  if (SW\_STATUS == SW3\_ACTIVE) stateSW = SW3\_ACTIVE;  else if (SW\_STATUS == SW1\_ACTIVE) stateSW = SW1\_ACTIVE;  break;  case SW3\_ACTIVE:  Light\_cmd(D1, D\_ON);  if (SW\_STATUS == SW2\_ACTIVE) stateSW = SW2\_ACTIVE;  break;  default:  stateSW = 0;  break;  }  // Kiểm tra trạng thái của nút nhấn P1  switch (stateP1) {  case 0:  Light\_cmd(D2, D\_OFF);  if (P1\_Status == P\_ACTIVE) {  stateP1 = 1;  while (PushButton\_read(P1) == P\_ACTIVE); // Chờ thả nút  }  break;  case 1:  Light\_cmd(D2, D\_OFF);  if (P1\_Status == P\_ACTIVE) {  stateP1 = 2;  DelayCnt = HAL\_GetTick();  while (PushButton\_read(P1) == P\_ACTIVE);  }  break;  case 2:  if (HAL\_GetTick() - DelayCnt >= 1000) {  Light\_cmd(D2, D\_ON);  } else if (HAL\_GetTick() - DelayCnt >= 2000) {  Light\_cmd(D2, D\_OFF);  DelayCnt = HAL\_GetTick();  }  if (P1\_Status == P\_ACTIVE) {  stateP1 = 3;  while (PushButton\_read(P1) == P\_ACTIVE);  }  break;  case 3:  Light\_cmd(D2, D\_ON);  if (P1\_Status == P\_ACTIVE) {  stateP1 = 0;  while (PushButton\_read(P1) == P\_ACTIVE);  }  break;  default:  stateP1 = 0;  break;  }  } |

|  |
| --- |
| int main(void)  {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  stateSW = 0;  DelayCnt = HAL\_GetTick(); // Sử dụng HAL\_GetTick để tránh sử dụng delay blocking    while (1)  {  P1\_STATUS = PushButton\_read(P1);  P2\_STATUS = PushButton\_read(P2);  uint8\_t SW\_STATUS = Switch\_read();  Baitap3\_4(SW\_STATUS, P1\_STATUS);  }  } |

* 1. **Lập trình ứng dụng**
* Khi mới mở điện: Đèn D1, D2 tắt;
* Mỗi khi nhấn nút nhấn P1 hoặc P2, đèn D1 chuyển 3 trạng thái (lặp lại):
  + Nhấn lần 1: Trạng thái 1 đèn D1 tắt.
  + Nhấn lần 2: Trạng thái 2 đèn D1 sáng tắt chu kỳ 2s.
  + Nhấn lần 3: Trạng thái 3 đèn D1 sáng liên tục.

Mã nguồn chương trình (chỉ copy nội dung đã thêm hoặc chỉnh sửa từ code dự án)

|  |
| --- |
| /\* Includes ------------------------------------------------------------------\*/  #include "main.h"  #include "myGPIO.h"  #include <stdbool.h> // Include for boolean type  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  /\* Global Variables \*/  uint8\_t stateP1 = 0;  uint8\_t stateP2 = 0;  uint32\_t LastTick = 0;  uint8\_t stateSW = 0;  /\*\*  \* @brief Handles button press behavior for P1 and P2  \*/  void Baitap3\_5(uint8\_t P1\_Status, uint8\_t P2\_Status) {  bool buttonPressed = (P1\_Status == P\_ACTIVE || P2\_Status == P\_ACTIVE);  switch (stateP1) {  case 0:  Light\_cmd(D1, D\_OFF);  if (buttonPressed) {  stateP1 = 1;  HAL\_Delay(50); // Debounce delay  while (PushButton\_read(P1) == P\_ACTIVE || PushButton\_read(P2) == P\_ACTIVE);  }  break;  case 1:  Light\_cmd(D1, D\_OFF);  if (buttonPressed) {  stateP1 = 2;  LastTick = HAL\_GetTick(); // Reset time for blinking  HAL\_Delay(50);  while (PushButton\_read(P1) == P\_ACTIVE || PushButton\_read(P2) == P\_ACTIVE);  }  break;  case 2:  if (HAL\_GetTick() - LastTick >= 1000) {  Light\_cmd(D1, D\_ON);  } else if (HAL\_GetTick() - LastTick >= 2000) {  Light\_cmd(D1, D\_OFF);  LastTick = HAL\_GetTick();  }  if (buttonPressed) {  stateP1 = 3;  HAL\_Delay(50);  while (PushButton\_read(P1) == P\_ACTIVE || PushButton\_read(P2) == P\_ACTIVE);  }  break;  case 3:  Light\_cmd(D1, D\_ON);  if (buttonPressed) {  stateP1 = 0;  HAL\_Delay(50);  while (PushButton\_read(P1) == P\_ACTIVE || PushButton\_read(P2) == P\_ACTIVE);  }  break;  default:  stateP1 = 0;  break;  }  }  /\*\*  \* @brief The main application loop.  \*/ |

|  |
| --- |
| int main(void)  {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  stateSW = 0;  LastTick = HAL\_GetTick();  while (1)  {  P1\_STATUS = PushButton\_read(P1);  P2\_STATUS = PushButton\_read(P2);  uint8\_t SW\_STATUS = Switch\_read();  Baitap3\_5(P1\_STATUS, P2\_STATUS);  }  } |

* 1. **Lập trình ứng dụng**
* Khi mới mở điện: Đèn D1, D2 tắt;
* Mỗi khi nhấn nút nhấn P1 đèn D1, D2 chuyển 2 trạng thái (lặp lại):
  + Trạng thái 1 đèn D1, D2 tắt.
  + Trạng thái 2 đèn D1 sáng tắt chu kỳ 2s. Đèn D2 tắt.
* Mỗi khi nhấn nút nhấn P2 đèn D1, D2 chuyển 2 trạng thái (lặp lại):
  + Trạng thái 1 đèn D1, D2 tắt.
  + Trạng thái 2 đèn D1 tắt. Đèn D2 sáng tắt chu kỳ 2s.
* Mỗi khi nhấn nút nhấn P3 đèn D1, D2 chuyển 2 trạng thái (lặp lại):
  + Trạng thái 1 đèn D1, D2 tắt.
  + Trạng thái 2 đèn D1, D2 sáng tắt so le chu kỳ 2s.
* Mỗi khi nhấn nút nhấn P4 đèn D1, D2 chuyển 2 trạng thái (lặp lại):
  + Trạng thái 1 đèn D1, D2 tắt.
  + Trạng thái 2 đèn D1, D2 sáng tắt cùng nhau chu kỳ 2s.

Mã nguồn chương trình (chỉ copy nội dung đã thêm hoặc chỉnh sửa từ code dự án)

|  |
| --- |
| /\* Includes ------------------------------------------------------------------\*/  #include "main.h"  #include "myGPIO.h"  #include <stdbool.h> // Include for boolean type  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  /\* Global Variables \*/  uint8\_t stateP1 = 0, stateP2 = 0, stateP3 = 0, stateP4 = 0;  uint32\_t lastTick\_P1 = 0, lastTick\_P2 = 0, lastTick\_P3 = 0, lastTick\_P4 = 0;  /\*\*  \* @brief Handles button press behavior for P1, P2, P3, and P4  \*/  void Baitap3\_6(uint8\_t P1\_Status, uint8\_t P2\_Status, uint8\_t P3\_Status, uint8\_t P4\_Status) {  uint32\_t currentTick = HAL\_GetTick();  // P1: D1 nhấp nháy 2s, D2 luôn tắt  if (P1\_Status == P\_ACTIVE) {  stateP1 = !stateP1;  HAL\_Delay(50);  while (PushButton\_read(P1) == P\_ACTIVE);  }  if (stateP1) {  if ((currentTick - lastTick\_P1) >= 1000) {  Light\_cmd(D1, !HAL\_GPIO\_ReadPin(D1\_GPIO\_Port, D1\_Pin));  lastTick\_P1 = currentTick;  }  } else {  Light\_cmd(D1, D\_OFF);  }  // P2: D2 nhấp nháy 2s, D1 luôn tắt  if (P2\_Status == P\_ACTIVE) {  stateP2 = !stateP2;  HAL\_Delay(50);  while (PushButton\_read(P2) == P\_ACTIVE);  }  if (stateP2) {  if ((currentTick - lastTick\_P2) >= 1000) {  Light\_cmd(D2, !HAL\_GPIO\_ReadPin(D2\_GPIO\_Port, D2\_Pin));  lastTick\_P2 = currentTick;  }  } else {  Light\_cmd(D2, D\_OFF);  }  // P3: D1 và D2 nhấp nháy so le  if (P3\_Status == P\_ACTIVE) {  stateP3 = !stateP3;  HAL\_Delay(50);  while (PushButton\_read(P3) == P\_ACTIVE);  }  if (stateP3) {  if ((currentTick - lastTick\_P3) >= 1000) {  Light\_cmd(D1, !HAL\_GPIO\_ReadPin(D1\_GPIO\_Port, D1\_Pin));  Light\_cmd(D2, !HAL\_GPIO\_ReadPin(D2\_GPIO\_Port, D2\_Pin));  lastTick\_P3 = currentTick;  }  } else {  Light\_cmd(D1, D\_OFF);  Light\_cmd(D2, D\_OFF);  }  // P4: D1 và D2 sáng tắt cùng nhau  if (P4\_Status == P\_ACTIVE) {  stateP4 = !stateP4;  HAL\_Delay(50);  while (PushButton\_read(P4) == P\_ACTIVE);  }  if (stateP4) {  if ((currentTick - lastTick\_P4) >= 1000) {  bool currentState = HAL\_GPIO\_ReadPin(D1\_GPIO\_Port, D1\_Pin);  Light\_cmd(D1, !currentState);  Light\_cmd(D2, !currentState);  lastTick\_P4 = currentTick;  }  } else {  Light\_cmd(D1, D\_OFF);  Light\_cmd(D2, D\_OFF);  }  }  /\*\*  \* @brief The main application loop.  \*/ |

|  |
| --- |
| int main(void)  {  HAL\_Init();  SystemClock\_Config();  MX\_GPIO\_Init();  while (1)  {  P1\_STATUS = PushButton\_read(P1);  P2\_STATUS = PushButton\_read(P2);  P3\_STATUS = PushButton\_read(P3);  P4\_STATUS = PushButton\_read(P4);  Baitap3\_6(P1\_STATUS, P2\_STATUS, P3\_STATUS, P4\_STATUS);  }  } |