Hướng dẫn lập trình bộ Trí Uẩn thông minh

1. Các khối Trí Uẩn

Các ký hiệu trong hình vẽ của các khối Trí Uẩn

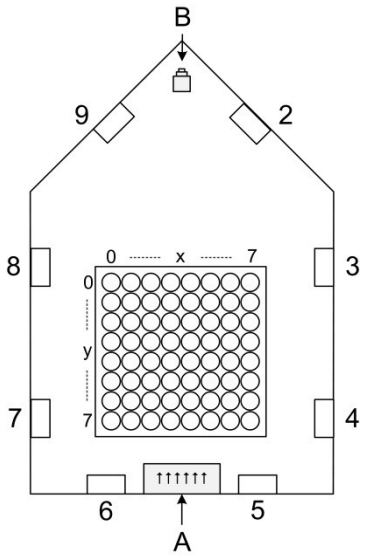
Ký hiệu	Mô tả
A	Cổng giao tiếp máy tính
B	Nút bấm bật / tắt nguồn pin
Số từ 2 tới 9	Các cổng giao tiếp VLC, nối với các chân 2-9 của Arduino (Visible Light Communication: giao tiếp dữ liệu bằng ánh sáng)
Số 10	Cảm biến ánh sáng, nối với chân số 10 của Arduino
Mặt trên	Nhìn từ mặt trên
Mặt dưới	Nhìn từ mặt dưới (mặt có cổng giao tiếp máy tính)

Hình vẽ "Mặt trên" thể hiện vị trí các cổng giao tiếp dữ liệu khi nhìn từ mặt trên.

Hình vẽ "Mặt dưới" thể hiện vị trí các cổng giao tiếp dữ liệu khi nhìn từ mặt dưới.

Chú ý: kiểm tra và tắt nguồn pin ngay khi không chạy nữa để dùng pin được lâu.

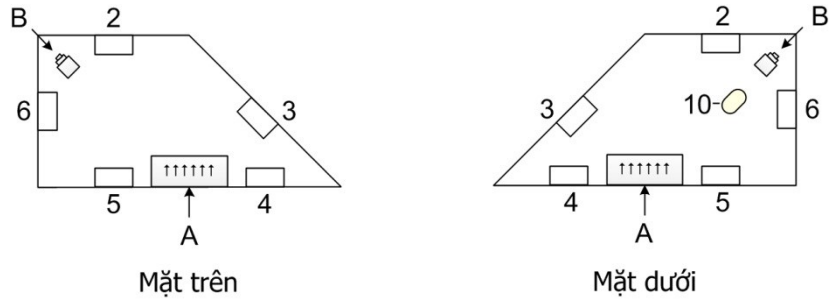
1.1. Khối Trí Uẩn 0



Khối Trí Uẩn 0 có tám cổng giao tiếp VLC nối với các chân từ 2 tới 9 của mạch Arduino. Trên khối này có một bảng LED ma trận kích thước 8×8 dùng để hiển thị hình ảnh.

Mạch Arduino trong khối Trí Uẩn 0 tương thích với **Arduino Uno**. Khi lập trình cho khối này bạn cần lựa chọn board **Arduino Uno**.

1.2. Khối Trí Uẩn 1

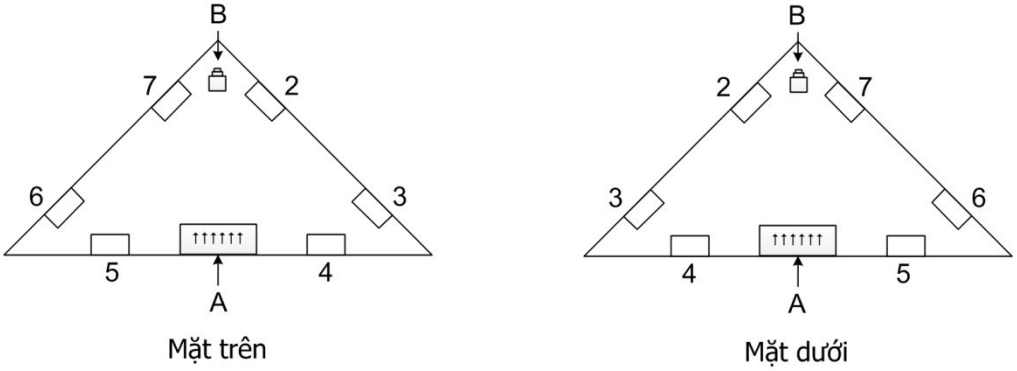


Khối Trí Uẩn 1 có năm cổng giao tiếp VLC nối với các chân từ 2 tới 6 của mạch Arduino.

Ở mặt dưới có một cảm biến ánh sáng nối với chân 10 của Arduino. Có thể sử dụng cảm biến này để xác định xem khối Trí Uẩn đang ngửa lên hay úp xuống.

Mạch Arduino trong khối Trí Uẩn 1 tương thích với **ATmega8**. Khi lập trình cho khối này bạn cần lựa chọn board **ATmega8** (ATmega8 @ 8 MHz / 38400).

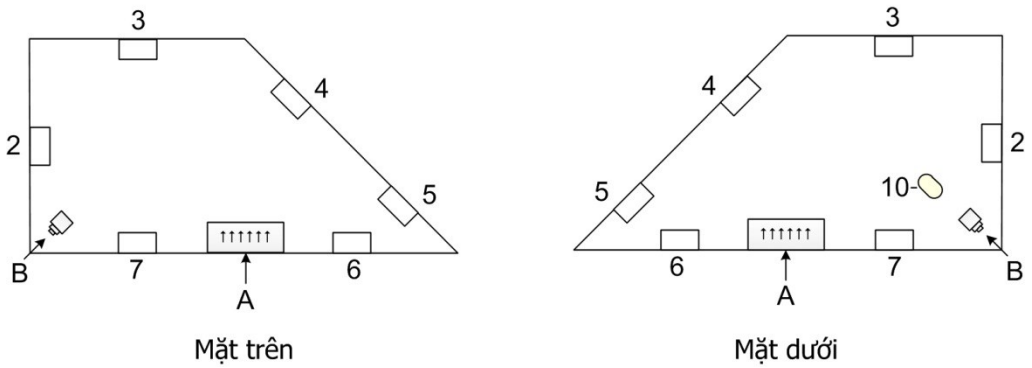
1.3. Khối Trí Uẩn 2



Khối Trí Uẩn 2 có sáu cổng giao tiếp VLC nối với các chân từ 2 tới 7 của mạch Arduino.

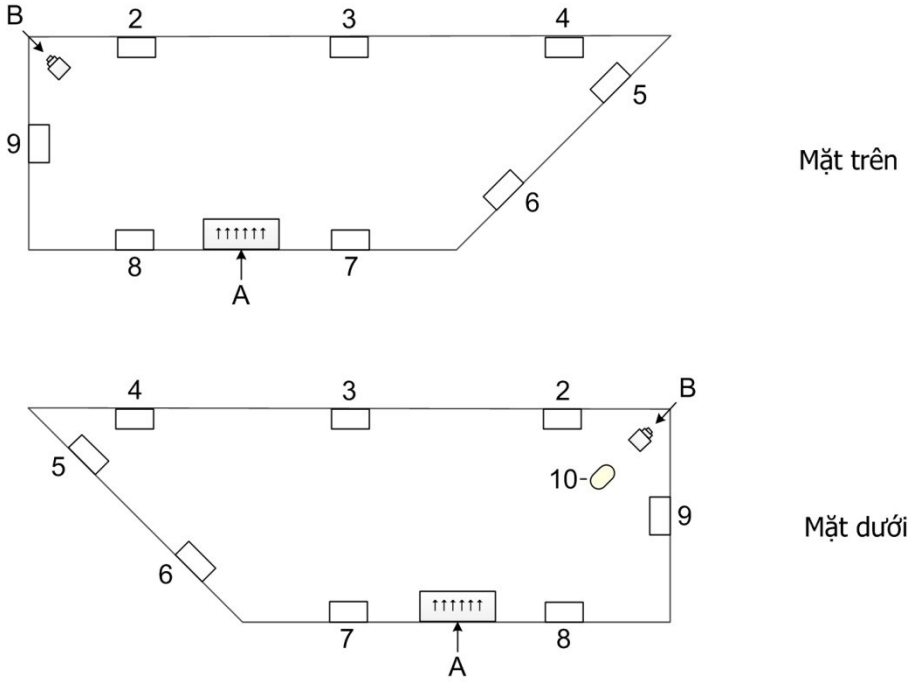
Chọn board **ATmega8** khi lập trình cho khối Trí Uẩn 2.

1.4. Khối Trí Uẩn 3



Khối Trí Uẩn 3 có sáu cổng giao tiếp VLC nối với các chân từ 2 tới 7 của mạch Arduino.
Ở mặt dưới có một cảm biến ánh sáng nối với chân 10 của Arduino.
Chọn board **ATmega8** khi lập trình cho khối Trí Uẩn 3.

1.5. Khối Trí Uẩn 4



Khối Trí Uẩn 4 có tám cổng giao tiếp VLC nối với các chân từ 2 tới 9 của mạch Arduino.
Ở mặt dưới có một cảm biến ánh sáng nối với chân 10 của Arduino.
Chọn board **ATmega8** khi lập trình cho khối Trí Uẩn 4.

2. Thư viện thVLC

Bộ ghép hình Trí Uẩn thông minh gồm bảy khối, bên trong mỗi khối có một mạch Arduino và các cổng giao tiếp VLC. Khi hai cổng VLC của hai khối được đặt đối diện và sát nhau, hai khối có thể trao đổi dữ liệu qua hai cổng VLC này. Như vậy các khối có thể nhận biết vị trí của các khối xung quanh, qua đó nhận biết được hình ghép tổng thể của cả bảy khối. Cổng VLC phát dữ liệu thông qua đèn LED và nhận về dữ liệu thông qua cảm biến ánh sáng (quang trở).

Chú ý: 2 cổng giao tiếp VLC chỉ trao đổi dữ liệu được khi 2 cổng này được đặt đối diện và sát nhau.

Thư viện thVLC cho phép lập trình gửi nhận dữ liệu qua các cổng VLC. Thư viện có các hàm cơ bản sau:

```
begin()
getID()
sendByte()
receiveReady()
receiveResult()
sensorRead()
```

Ngoài ra thư viện thVLC còn có một số hàm nâng cao cho phép lập trình linh hoạt hơn:

```
ledWrite()
txSetByte()
txSend()
```

2.1. begin()

Mô tả

Khởi tạo việc gửi nhận dữ liệu cho các cổng giao tiếp VLC.

Cú pháp

```
thVLC.begin()
```

Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>

void setup()
{
  thVLC.begin();
}

void loop()
{
}
```

2.2. getID()

Mô tả

Xác định mã số của khối Trí Uẩn (một trong năm loại khối như mô tả trong phần 1)

Cú pháp

```
thVLC.getID()
```

Giá trị trả về

Một trong năm giá trị sau:

```
TRI_UAN_0 (0)
TRI_UAN_1 (1)
TRI_UAN_2 (2)
TRI_UAN_3 (3)
TRI_UAN_4 (4)
```

Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>

void setup()
{
  Serial.begin(9600);
  thVLC.begin();

  int IDBlock = thVLC.getID();

  switch (IDBlock)
  {
    case TRI_UAN_0:
      Serial.println("Tri Uan 0");
      break;
    case TRI_UAN_1:
      Serial.println("Tri Uan 1");
      break;
    case TRI_UAN_4:
      Serial.println("Tri Uan 4");
      break;
    default:
      Serial.println("ID not set");
      break;
  }
}

void loop()
```

```
{
}
```

2.3. sendByte()

Mô tả

Gửi một dữ liệu 8 bit ra cổng giao tiếp VLC.

Cú pháp:

```
thVLC.sendByte(port, value)
```

Tham số

port: cổng giao tiếp VLC để gửi dữ liệu, giá trị từ 2 đến 9.

value: dữ liệu sẽ được gửi đi, giá trị từ 0 tới 255 (byte).

Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>

void setup()
{
  thVLC.begin();
}

void loop()
{
  int sendPort = 2;
  byte sendValue = 14;
  thVLC.sendByte(sendPort, sendValue);
  delay(1000);
}
```

2.4. receiveReady()

Mô tả

Kiểm tra xem cổng VLC có nhận được dữ liệu hay không.

Cú pháp

```
thVLC.receiveReady(port)
```

Tham số

port: cổng giao tiếp VLC cần kiểm tra, giá trị từ 2 đến 9.

Giá trị trả về

- `true` nếu cổng VLC đã nhận được dữ liệu.
- `false` nếu cổng VLC chưa nhận được dữ liệu.
- Khi cổng VLC nhận được dữ liệu, receiveReady() sẽ trả về giá trị `true` cho tới khi người dùng đọc dữ liệu nhận về (bằng hàm receiveResult()).

Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>

void setup()
{
  Serial.begin(9600);
  thVLC.begin();
}

void loop()
{
  int port = 2;
  if (thVLC.receiveReady(port))
  {
    Serial.println("VLC port 2 received data.");
    thVLC.receiveResult(port);
  }
}
```

2.5. receiveResult()

Mô tả

Đọc về dữ liệu vừa nhận về trên cổng giao tiếp VLC.

Mỗi khi cổng VLC nhận được một dữ liệu thì receiveReady() sẽ trả về giá trị true để thông báo đã nhận dữ liệu, khi này ta có thể dùng receiveResult() để đọc về dữ liệu. Nếu gọi receiveResult() khi receiveReady() có giá trị false thì giá trị trả về không chính xác.

Cú pháp:

```
thVLC.receiveResult(port)
```

Tham số

port: cổng giao tiếp VLC, giá trị từ 2 đến 9.

Giá trị trả về

Dữ liệu cổng VLC nhận được, nằm trong khoảng từ 0 tới 255 (byte).

Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>

void setup()
{
  Serial.begin(9600);
  thVLC.begin();
}

void loop()
{
  int port = 2;
  if (thVLC.receiveReady(port))
  {
    byte result = thVLC.receiveResult(port);
    Serial.print("VLC port 2 received data : ");
    Serial.println(result);
  }
}
```

2.6. sensorRead()

Mô tả

Đọc về giá trị của cảm biến ánh sáng được nối với chân `pin` của Arduino.

Trên mỗi cổng giao tiếp VLC có một bộ cảm biến ánh sáng để nhận dữ liệu. Ngoài ra, ở các khối Trí Uẩn 1, Trí Uẩn 3 và Trí Uẩn 4 còn có một cảm biến ánh sáng được nối với chân 10 của Arduino.

Việc đọc về giá trị của cảm biến ánh sáng trên các khối Trí Uẩn sẽ cung cấp cho bạn thông tin về mức độ sáng, tối của môi trường xung quanh mỗi cảm biến. Với cảm biến ánh sáng nối với chân 10, thông tin này sẽ cho biết khối Trí Uẩn đang úp xuống hay đang ngửa lên.

Cú pháp

```
thVLC.sensorRead(pin)
```

Tham số

pin: chân của Arduino nối với cảm biến ánh sáng, giá trị từ 2 đến 10.

Giá trị trả về

Kết quả đo của cảm biến ánh sáng, giá trị nằm trong khoảng từ 0 tới 1023.

Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>

void setup()
{
  Serial.begin(9600);
  thVLC.begin();
}

void loop()
{
  int sensorPin = 10;
  int sensorValue = thVLC.sensorRead(sensorPin);

  Serial.print("[sensor 10] : ");
  Serial.println(sensorValue);

  delay(500);
}
```

❖ Các hàm nâng cao

2.7. ledWrite()

Mô tả

Bật hoặc tắt đèn LED trên cổng giao tiếp VLC.

Cú pháp

```
thVLC.ledWrite(port, value)
```

Tham số

port: cổng giao tiếp VLC của khối Trí Uẩn, nhận giá trị từ 2 đến 9.

value: nhận giá trị `HIGH` hoặc `LOW`.

+ value = `HIGH` : bật sáng LED
+ value = `LOW` : tắt đèn LED

Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>

void setup()
```

```
{
  thVLC.begin();
}

void loop()
{
  thVLC.ledWrite(2, HIGH);
  delay(500);
  thVLC.ledWrite(2, LOW);
  delay(500);
}
```

2.8. txSetByte()

Mô tả

Kết hợp với txSend() để gửi nhiều dữ liệu đồng thời trên nhiều cổng giao tiếp VLC. txSetByte() thiết lập dữ liệu sẽ gửi đi trên từng cổng VLC.

Cú pháp

```
thVLC.txSetByte(port, value)
```

Tham số

port: cổng giao tiếp VLC sẽ gửi dữ liệu.

value: dữ liệu sẽ gửi đi, giá trị từ 0 tới 255 (byte).

Ví dụ

Xem ví dụ cho txSend().

2.9. txSend()

Mô tả

Kết hợp với txSetByte() để gửi nhiều dữ liệu đồng thời trên nhiều cổng giao tiếp VLC. txSend() gửi đi tất cả các dữ liệu qua các cổng VLC như đã được thiết lập bằng txSetByte().

Cú pháp

```
thVLC.txSend()
```

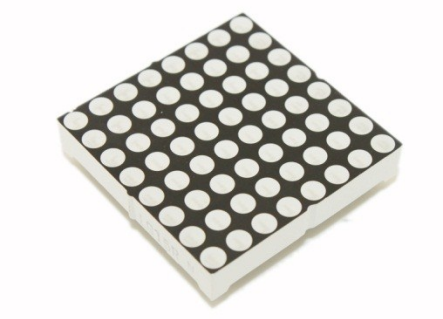
Ví dụ

```
#include <thAvr.h>
#include <thVLC.h>
```

```
void setup()
{
  thVLC.begin();
}

void loop()
{
  int port2 = 2;
  int port5 = 5;
  int port2Data = 20;
  int port5Data = 14;

  thVLC.txSetByte(port2, port2Data);    // VLC port 2 sẽ gửi dữ liệu 20
  thVLC.txSetByte(port5, port5Data);    // VLC port 5 sẽ gửi dữ liệu 14
  thVLC.txSend();                       // Gửi dữ liệu ra VLC port 2 và 5

  delay(1000);
}
```

3. Thư viện thLedMatrix

Khối Trí Uẩn 0 có một bảng LED ma trận kích thước 8×8 (với hai màu cơ bản xanh và đỏ) để hiển thị hình ảnh.



Thư viện thLedMatrix cho phép lập trình hiển thị hình ảnh lên bảng LED ma trận. Thư viện có các hàm sau:

```
begin()
clear()
setPixel()
setColumn()
setBitmap()
```

3.1. begin()

Mô tả

Khởi tạo hoạt động của bảng LED ma trận.

Cú pháp

```
thLedMatrix.begin()
```

Ví dụ

```
#include <thAvr.h>
#include <thLedMatrix.h>

void setup()
{
  thLedMatrix.begin();
}

void loop()
{
}
```

3.2. clear()

Mô tả

Xóa toàn bộ bảng LED ma trận.

Cú pháp

```
thLedMatrix.clear()
```

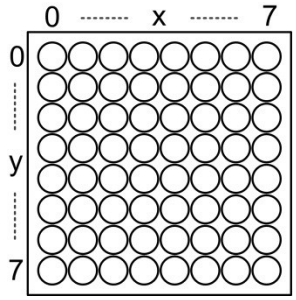
Ví dụ

```
#include <thAvr.h>
#include <thLedMatrix.h>

void setup()
{
  thLedMatrix.begin();
}

void loop()
{
  thLedMatrix.setPixel(1, 1, ORANGE);
  delay(1000);
  thLedMatrix.clear();
  delay(1000);
}
```

3.3. setPixel()

Mô tả

Thay đổi màu của một điểm ảnh (pixel). Mỗi điểm ảnh có hai màu cơ bản là xanh và đỏ, kết hợp lại thì hiển thị được 4 màu: xanh (green), đỏ (red), da cam (orange) và đen (black).



Hệ tọa độ của các pixel trên bảng LED ma trận được thể hiện như hình trên. Điểm (0, 0) nằm ở góc trên bên trái. Điểm (7, 7) nằm ở góc dưới bên phải.

Cú pháp

```
thLedMatrix.setPixel(x, y, color)
```

Tham số

x: tọa độ x của pixel, giá trị trong khoảng từ 0 tới 7

y: tọa độ y của pixel, giá trị trong khoảng từ 0 tới 7

color: màu sắc của pixel, một trong các giá trị `BLACK`, `RED`, `GREEN` hoặc `ORANGE`

Ví dụ

```
#include <thAvr.h>
#include <thLedMatrix.h>

void setup()
{
  thLedMatrix.begin();
}

void loop()
{
  thLedMatrix.setPixel(0, 7, ORANGE);
  thLedMatrix.setPixel(1, 6, RED);
  thLedMatrix.setPixel(2, 5, GREEN);
}
```

3.4. setColumn()

Mô tả

Thay đổi màu của một cột 8 pixel. Bảng LED ma trận có 8 cột từ 0 tới 7 theo tọa độ x.

Cú pháp

```
thLedMatrix.setColumn(x, redBitmap, greenBitmap)
```

Tham số

x: tọa độ x của cột, giá trị từ 0 tới 7

redBitmap: giá trị mã hóa màu đỏ (red) của 8 pixel trong cột.

greenBitmap: giá trị mã hóa màu xanh (green) của 8 pixel trong cột.

(bit 0 tương ứng với y = 0, bit 7 tương ứng với y = 7)

Ví dụ

```
#include <thAvr.h>
#include <thLedMatrix.h>

void setup()
```

```
{
  thLedMatrix.begin();
}

void loop()
{
  int columnX = 3;
  byte redBitmap   = 0x0F;
  byte greenBitmap = 0xF0;
  thLedMatrix.setColumn(columnX, redBitmap, greenBitmap);
}
```

3.5. setBitmap()

Mô tả

Hiển thị một hình ảnh (kích thước 8×8) lên bảng LED ma trận. Hình ảnh này được mã hóa thành 16 byte và lưu trong một mảng kiểu BITMAP có cấu trúc như sau:

```
BITMAP image[] =
{
  red0, grn0,
  red1, grn1,
  red2, grn2,
  red3, grn3,
  red4, grn4,
  red5, grn5,
  red6, grn6,
  red7, grn7
};
```

- red0, red1,… red7 tương ứng với các giá trị mã hóa màu đỏ của các cột từ 0 tới 7
- grn0, grn1,… grn7 tương ứng với các giá trị mã hóa màu xanh của các cột từ 0 tới 7

Cú pháp

```
thLedMatrix.setBitmap(image)
```

Tham số

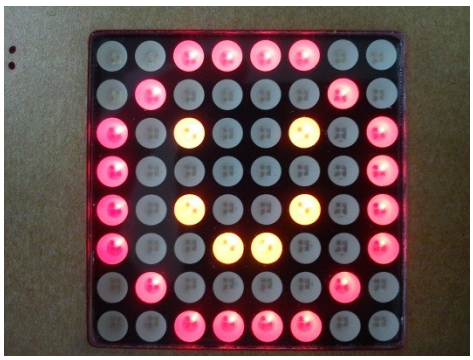
image: hình ảnh hiển thị lên bảng LED ma trận

Ví dụ

```
#include <thAvr.h>
#include <thLedMatrix.h>

BITMAP image[] =
{
  0x3C, 0x00,
```

```
  0x42, 0x00,
  0x95, 0x14,
  0xA1, 0x20,
  0xA1, 0x20,
  0x95, 0x14,
  0x42, 0x00,
  0x3C, 0x00
};

void setup()
{
  thLedMatrix.begin();
}

void loop()
{
  thLedMatrix.setBitmap(image);
}
```

Kết quả chạy ví dụ



4. Lập trình với loa báo

Trong khối Trí Uẩn 0 có một chiếc loa nối với chân A5 của mạch Arduino.

Để lập trình điều khiển loa, trước tiên phải khởi tạo hoạt động cho chân điều khiển:

```
int buzzerPin = A5;
pinMode(buzzerPin, OUTPUT);
```

Sau khi khởi tạo, sử dụng các lệnh sau để điều khiển loa phát âm thanh:

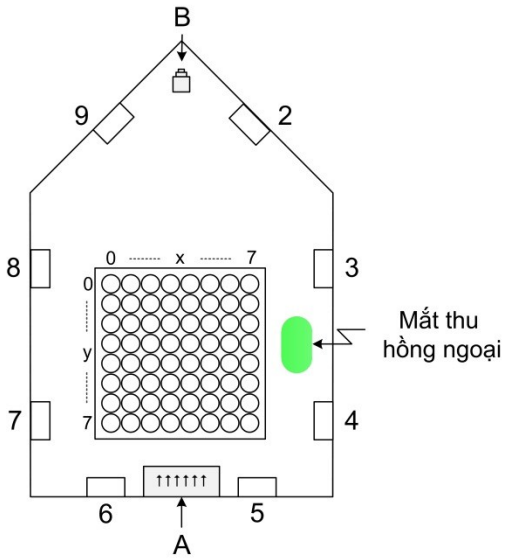
```
digitalWrite(buzzerPin, HIGH);    // Bật loa (kêu)
digitalWrite(buzzerPin, LOW);     // Tắt loa (không kêu)
```

Ví dụ

```
int buzzerPin = A5;

void setup()
{
  pinMode(buzzerPin, OUTPUT);       // Khởi tạo chân điều khiển loa
}

void loop()
{
  digitalWrite(buzzerPin, HIGH);    // Bật loa (kêu)
  delay(50);
  digitalWrite(buzzerPin, LOW);     // Tắt loa (không kêu)
  delay(950);
}
```

5. Lập trình với điều khiển hồng ngoại

Trong khối Trí Uẩn 0 có một mắt thu hồng ngoại để nhận tín hiệu điều khiển từ bộ điều khiển hồng ngoại 21 phím đi kèm (hoặc các bộ phát điều khiển hồng ngoại khác).



Vị trí mắt thu hồng ngoại

Mắt thu hồng ngoại nối với các chân 12 và chân A4 của mạch Arduino. Trong đó:

- Chân A4 của mạch Arduino dùng để cấp nguồn điện hoạt động cho mắt thu hồng ngoại. Cách điều khiển như sau:

```
int IRRxVcc = A4;
pinMode(IRRxVcc, OUTPUT);       // Khởi tạo chân cấp nguồn của mắt thu hông ngoại

digitalWrite(IRRxVcc, HIGH);    // Cấp nguồn điện cho mắt thu hồng ngoại hoạt động
digitalWrite(IRRxVcc, LOW);     // Không cấp điện cho mắt thu hồng ngoại, ngừng hoạt động
```

- Tín hiệu nhận về của mắt thu hồng ngoại được đưa tới chân 12 của mạch Arduino. Arduino sẽ đọc về tín hiệu này và giải mã tín hiệu bằng thư viện IRremote.

Tham khảo thư viện IRremote tại trang web https://github.com/shirriff/Arduino-IRremote

Chương trình ví dụ

```
#include <IRremote.h>

int IRRxVcc = A4;
int IRRxPin = 12;
```

```
IRrecv IRRx(IRRxPin);           // Khai báo module nhận tín hiệu hồng ngoại IRRx

decode_results results;

void setup()
{
  Serial.begin(9600);

  pinMode(IRRxVcc, OUTPUT);     // Khởi tạo chân cấp nguồn của mắt thu hông ngoại
  digitalWrite(IRRxVcc, HIGH);  // Cấp nguồn điện cho mắt thu hồng ngoại hoạt động
  IRRx.enableIRIn();            // Bắt đầu nhận tín hiệu hồng ngoại
}

void loop()
{
  if (IRRx.decode(&results))
  {
    int IRResult = results.value;
    IRRx.resume();              // Tiếp tục nhận tín hiệu hồng ngoại
    Serial.println(IRResult, HEX);
  }
}
```