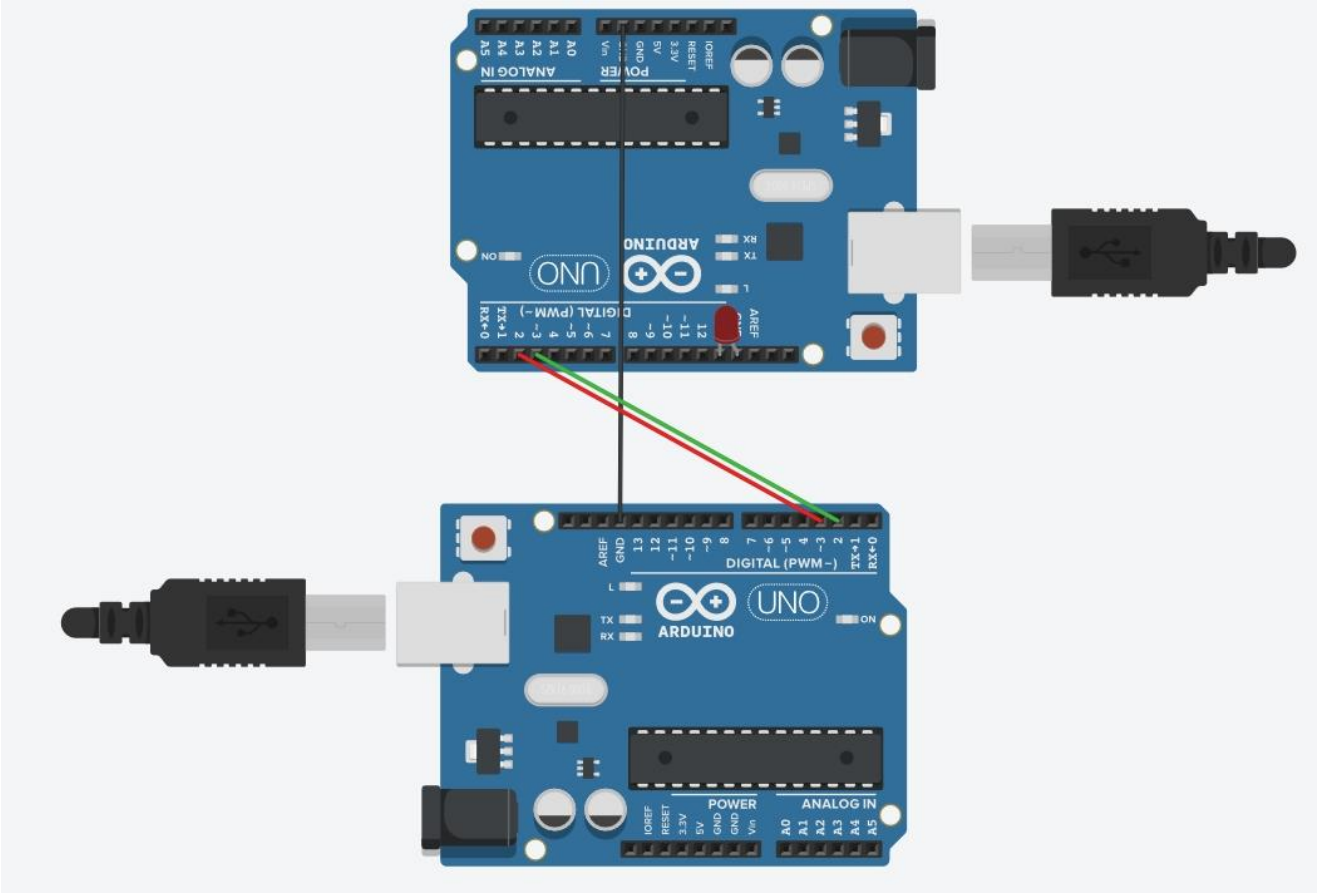
01076101
วิศวกรรมคอมพิวเตอร์เบื้องต้น
Introduction to Computer Engineering

Arduino #6

Serial Communication

การสื่อสารระหว่างบอร์ด

- การสื่อสารระหว่างบอร์ด สามารถทำได้หลายวิธี เช่น ถ้าต่อ Arduino 2 บอร์ด เข้าด้วยกันตามรูป



การสื่อสารระหว่างบอร์ด

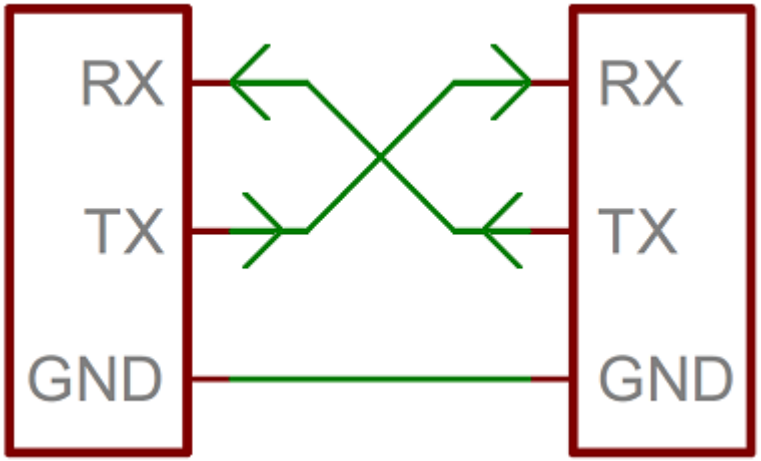
- ขา 2 เป็น Input ขา 3 เป็น Output โดยต่อครอสกันระหว่างขา 2 และ 3 ของทั้งสองบอร์ด
- ให้บอร์ดด้านบนรับข้อมูลจากบอร์ดด้านล่าง โดยรับเป็น 0 หรือ 1
- บอร์ดด้านล่างรับข้อมูลจากสวิตซ์ โดยถ้ากดสวิตซ์ให้ไฟที่ด้านบนติด

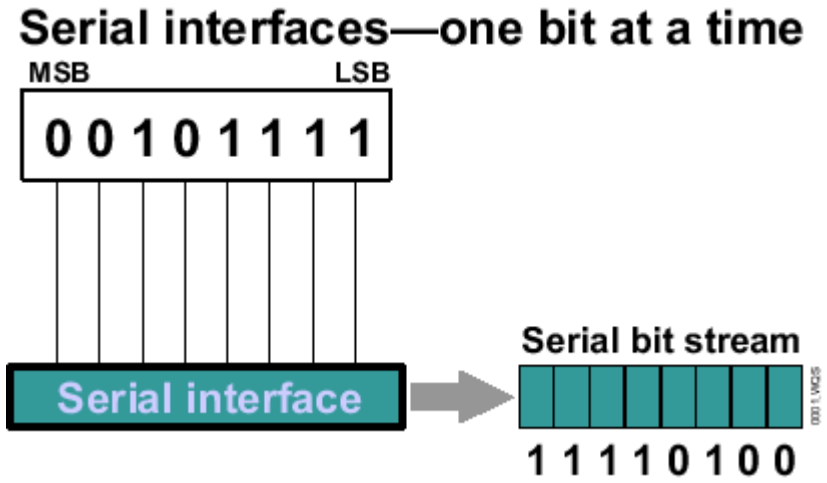
Discussion

- ถ้าบอร์ดด้านบนต้องการส่งคำว่า Hello มาแสดงใน Serial Monitor ของบอร์ดด้านล่าง จะทำได้หรือไม่ ผ่านวงจรข้างต้น
- มีปัญหา หรือ อุปสรรคอะไร หรือไม่
- หากมี จะมีแนวทางการแก้ปัญหาอย่างไร

Serial Communication

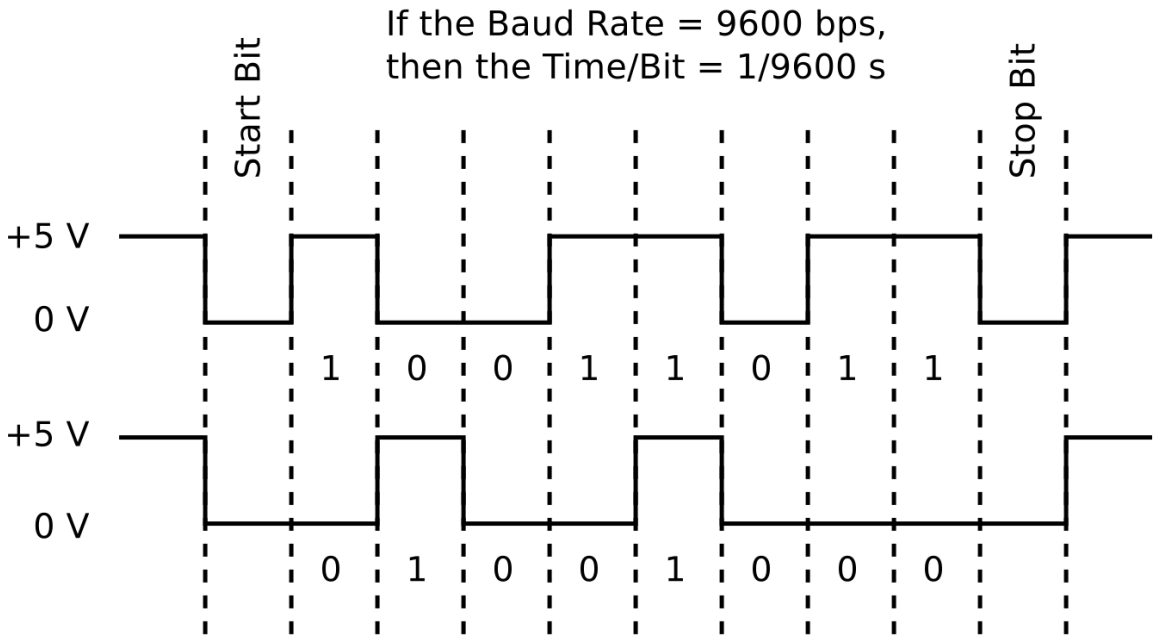
- เป็นการส่งข้อมูลครั้งละ 1 บิต โดย หากส่งด้านเดียวเรียก Simplex หากส่งได้ 2 ด้าน เรียก Duplex (หากรับส่งพร้อมกันเรียก Full Duplex หากรับส่งทีละข้าง เรียก Half Duplex)
- กรณีที่จะส่งข้อมูลหลายบิต จะต้องค่อยๆ เลื่อนมาทีละบิตเพื่อส่ง





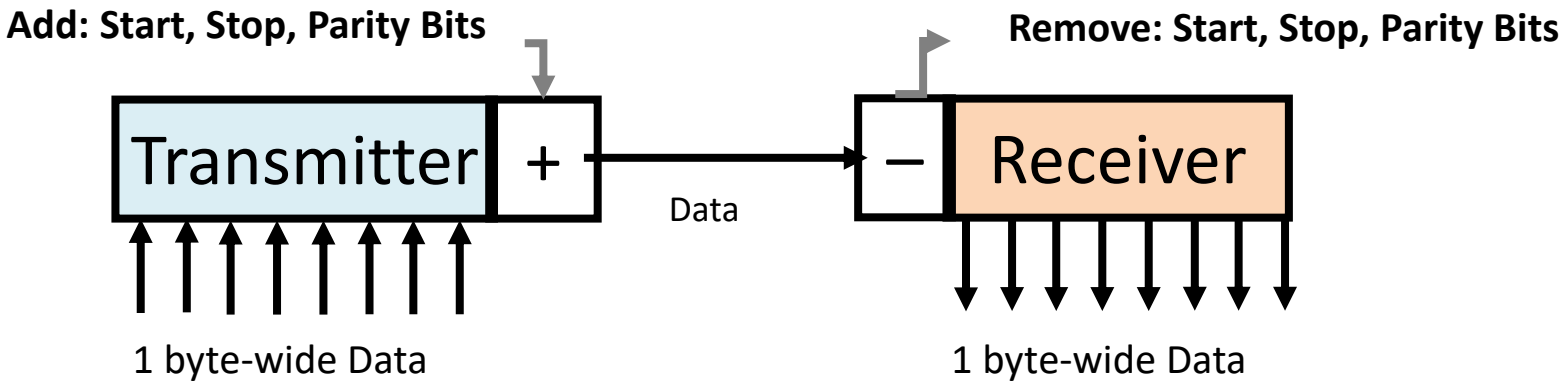
Serial Communication

- การส่งนิยมส่งเป็นชุด ชุดละ 1 ไบต์ (8 บิต)
- ในการส่งจะต้องมีการ Synchronize ระหว่าง 2 ข้างเสียก่อน เพื่อให้รู้ว่าจุดเริ่มของการส่งอยู่ที่ไหน (มักเรียกว่า start bit) (1->0) จากนั้นเป็นข้อมูล 8 บิต และตามด้วย stop bit (0->1)
- คาบเวลาของแต่ละบิตจะต้องเท่ากันทั้งสองด้าน เรียกว่า baud rate
- ถ้า baud rate = 9600 แต่ละบิตจะใช้เวลา 1/9600 s



Serial Communication

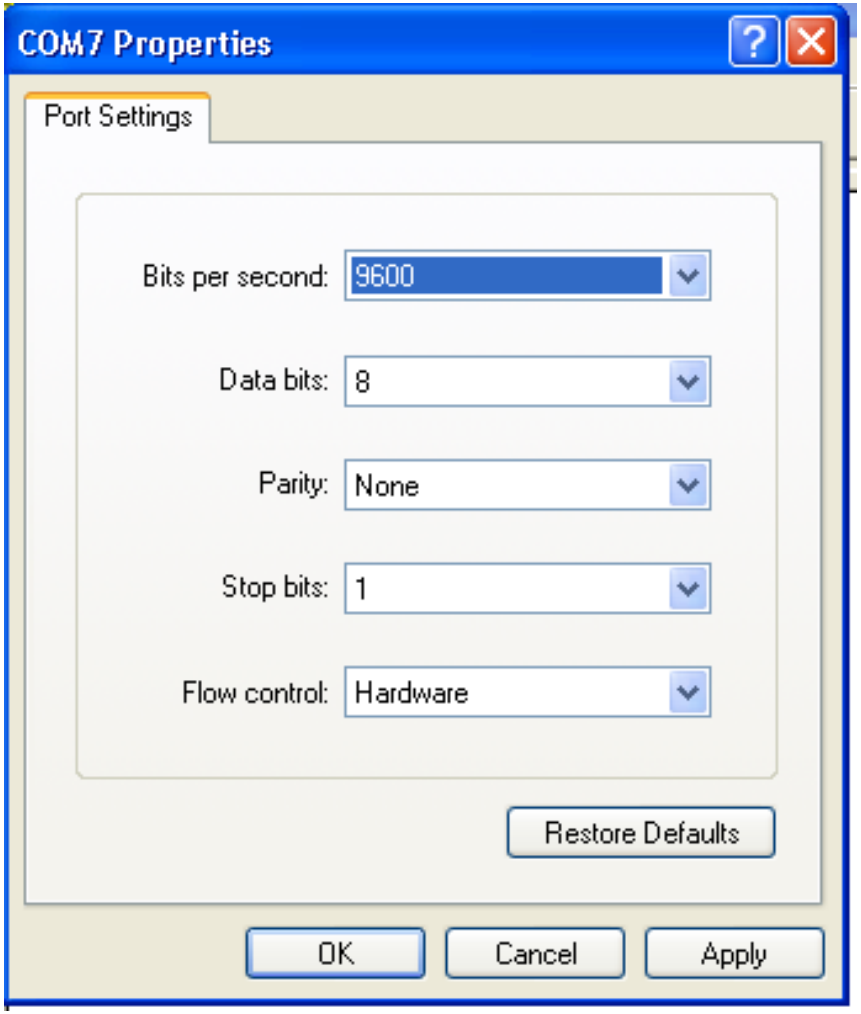
- ฝั่งรับและฝั่งส่ง จะมีเนื้อที่หน่วยความจำที่ใช้เพื่อการเลื่อนบิตรับและส่ง โดยจะเรียกหน่วยความจำส่วนนี้ว่า Transmit Buffer และ Receive Buffer (ขนาด Buffer อาจแตกต่างกันไปตามประเภทของ Hardware)
- Receive Buffer จะมีอีก 1 หน้าที่ คือ บอกว่า มีข้อมูลมาถึงหรือยัง
- สำหรับบอร์ด Arduino จะมี IC ที่ทำหน้าที่แปลง Serial <-> USB ทำให้ส่ง Serial ผ่าน USB ได้



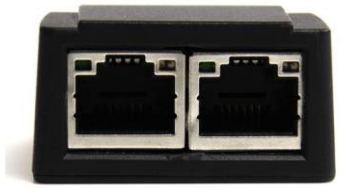
ข้อกำหนดเพื่อให้การส่งข้อมูลถูกต้อง

- Baud Rate : (Bit per second)
- Data bits
- Parity bit
- Stop bit

- 9600 – 8 – N - 1

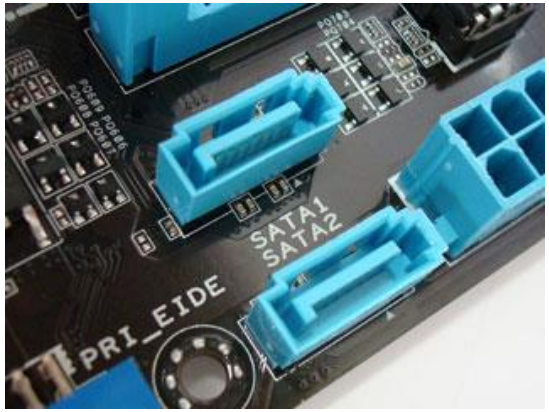


Serial Port

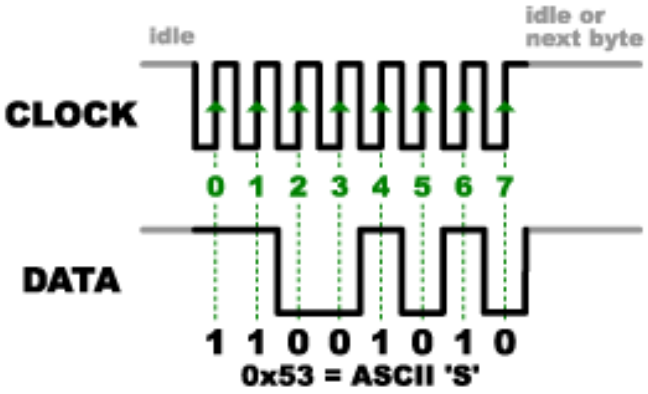


USB cable and port

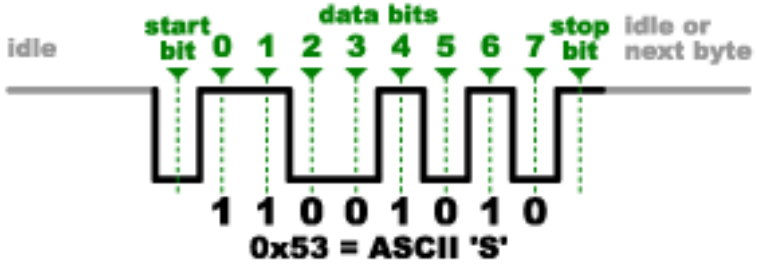
ComputerHope.com



Serial Communication



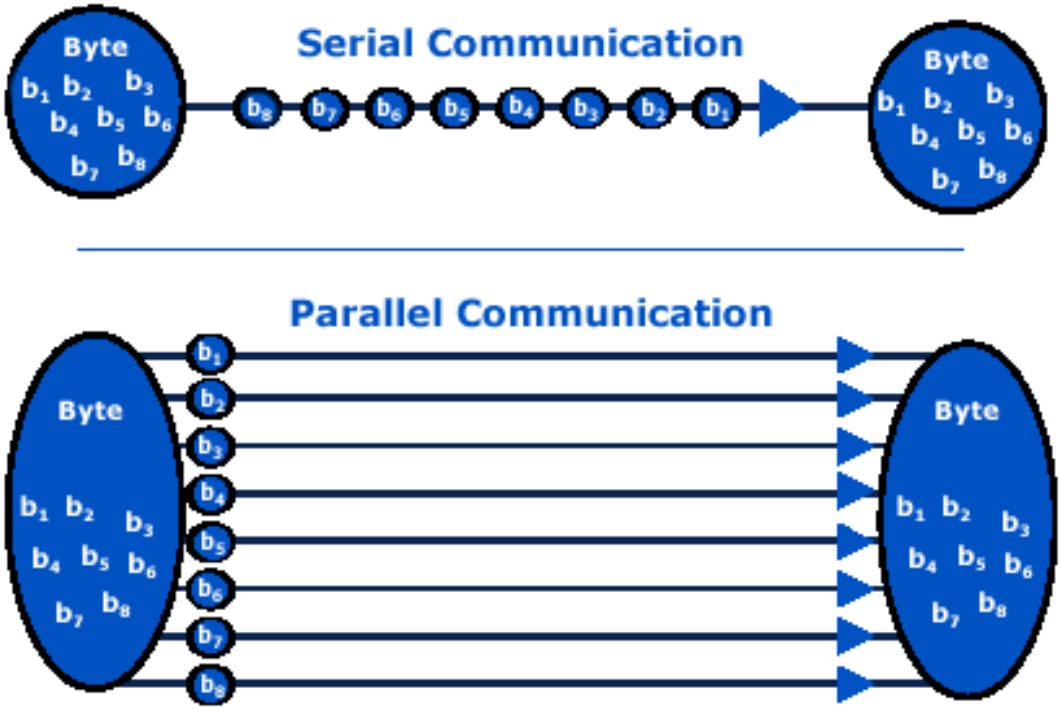
Synchronous



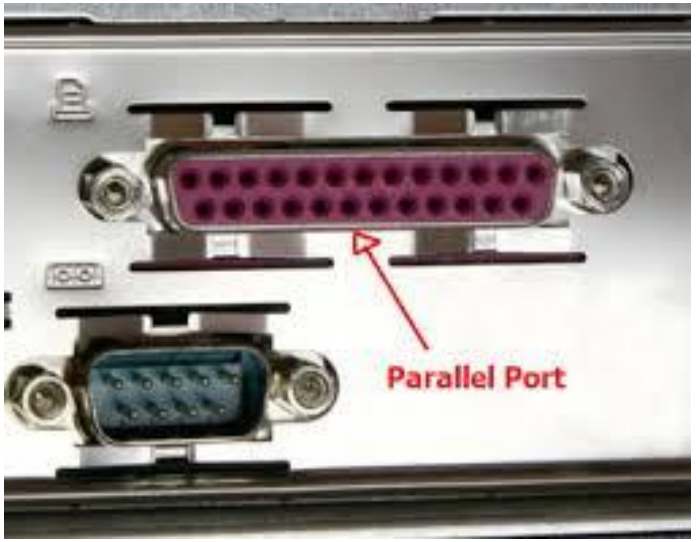
Asynchronous

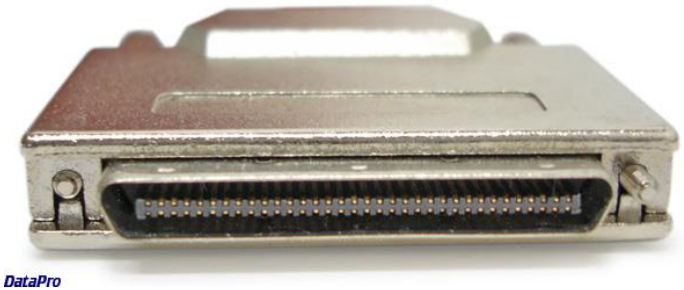
Parallel Communication

- การสื่อสารอีกรูปแบบหนึ่งที่มีการใช้กันมากในยุคก่อน คือ การสื่อสารแบบขนาน

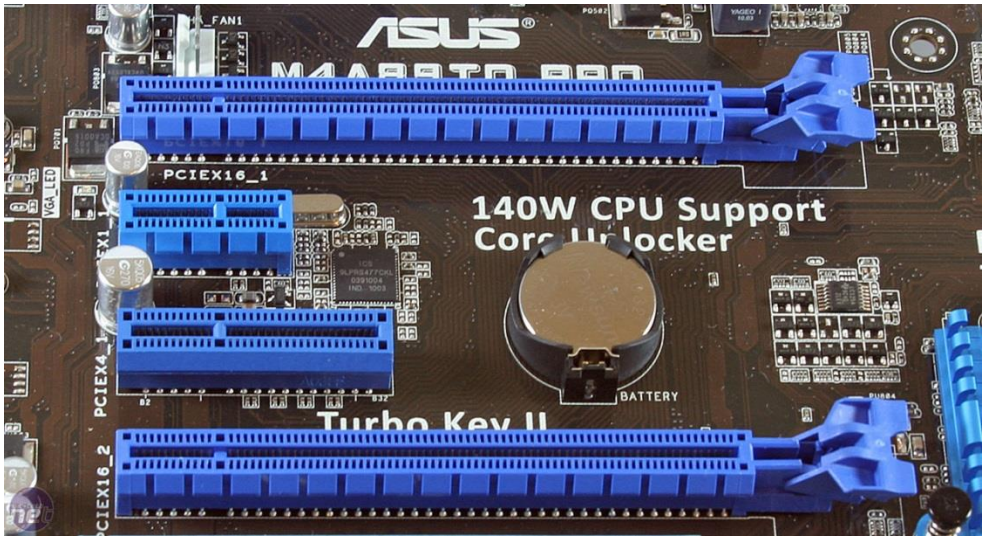


Parallel Port



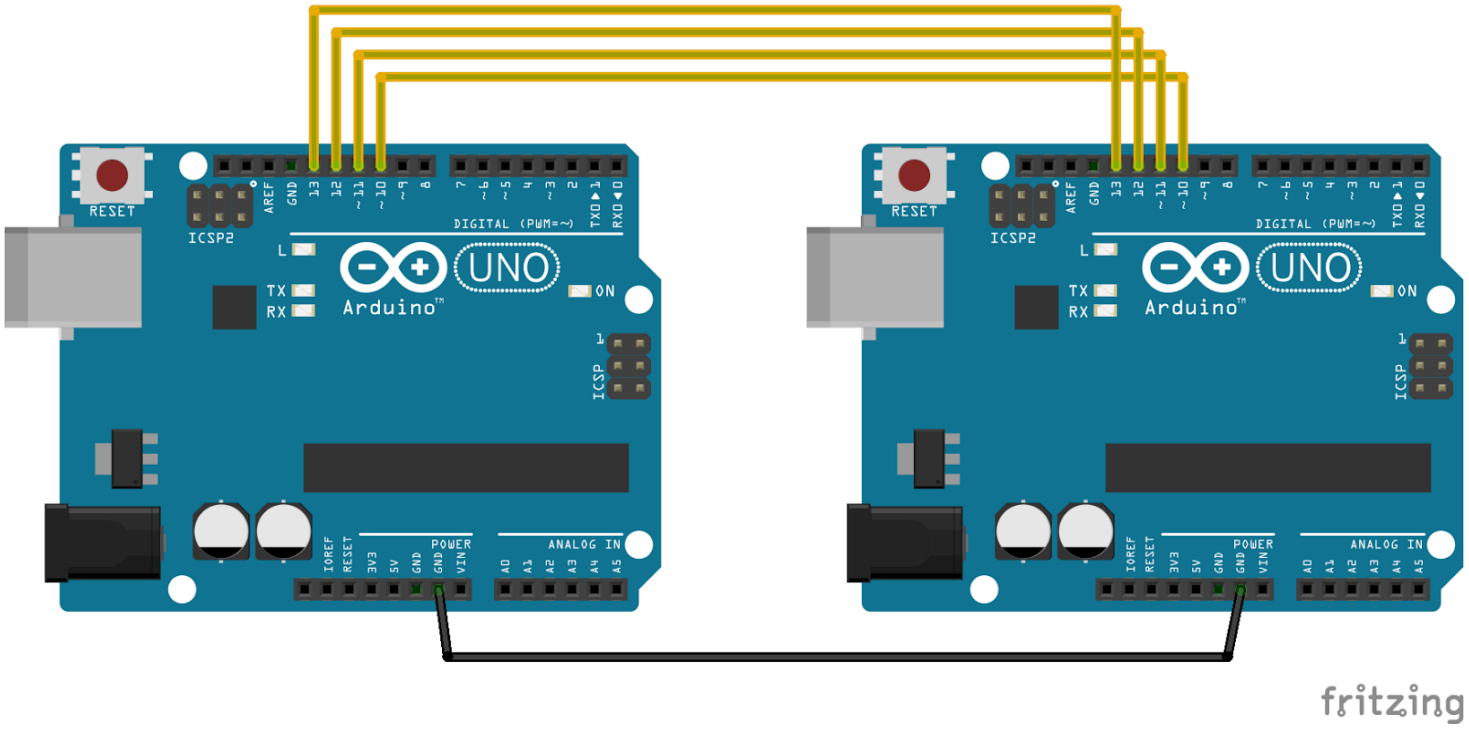


SCSI Port



Disscussion

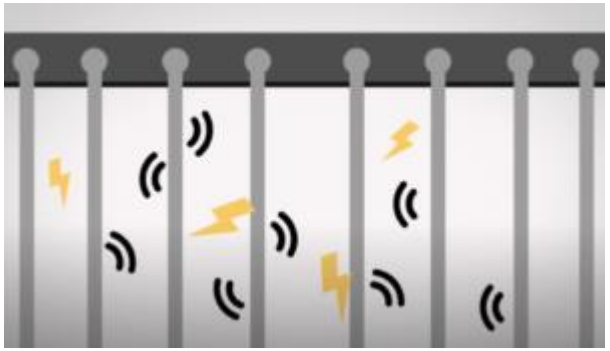
- จากโจทย์ก่อนหน้า หากต้องการส่งข้อมูลแบบขนานสามารถทำได้หรือไม่
- คิดว่าจะมีปัญหา อุปสรรคใดหรือไม่ และจะเลือกแบบไหนใช้งาน เพราะเหตุใด



Discussion

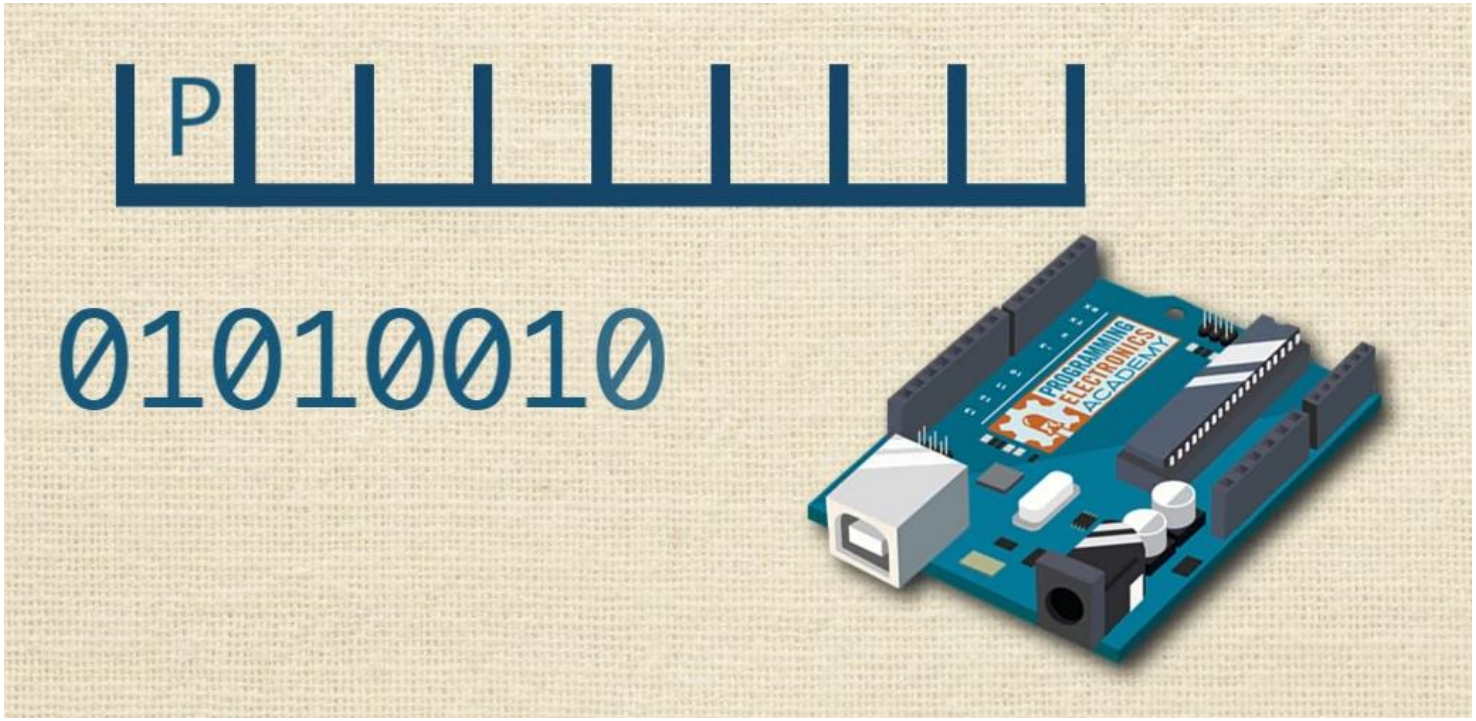
ระหว่าง Parallel กับ Serial อะไรส่งข้อมูลได้เร็วกว่ากัน

ทำไม Serial จึงนิยมมากกว่า



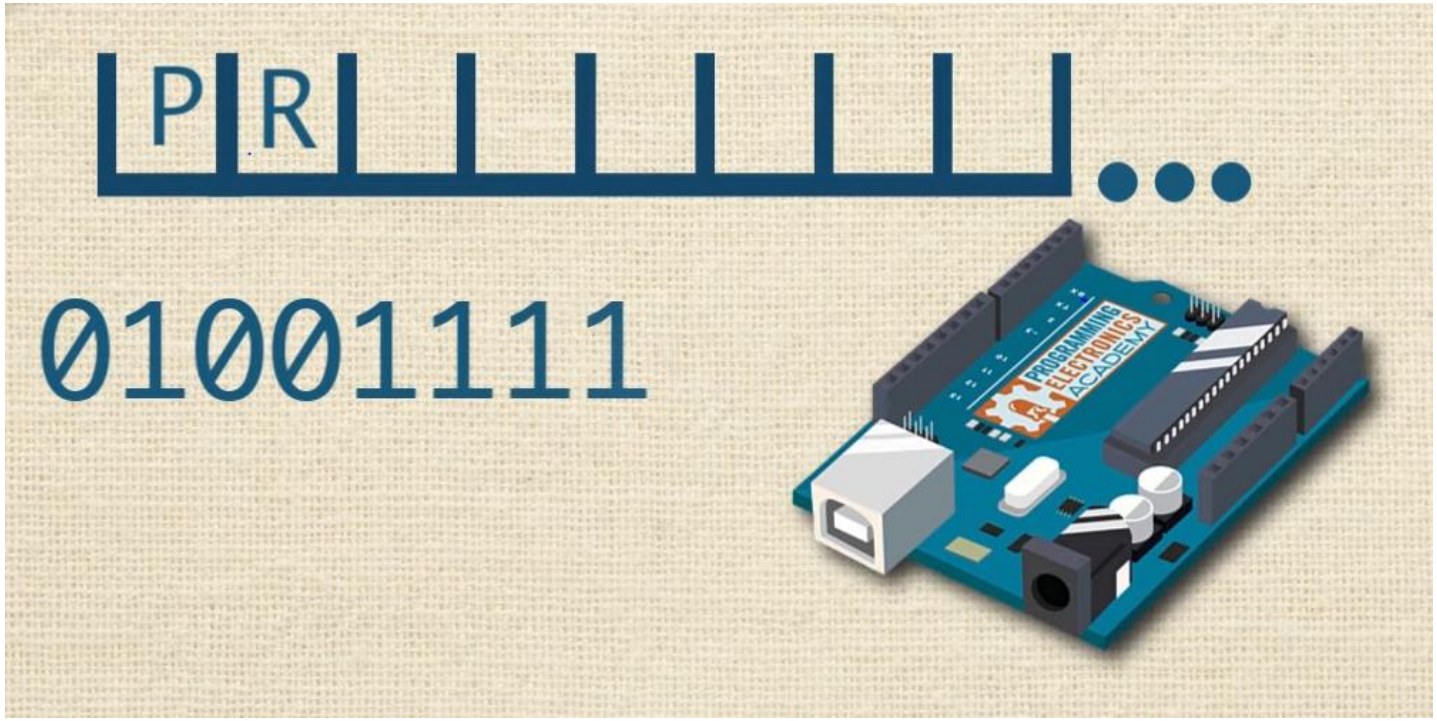
Serial Function

- Arduino จะรับข้อมูล ครั้งละ 1 บิต จนครบ 8 บิต จึงนำมาใส่ Receive Buffer
- จากรูป 01010010 = R (ก่อนหน้านี้รับตัว P มาแล้ว)



Serial Function

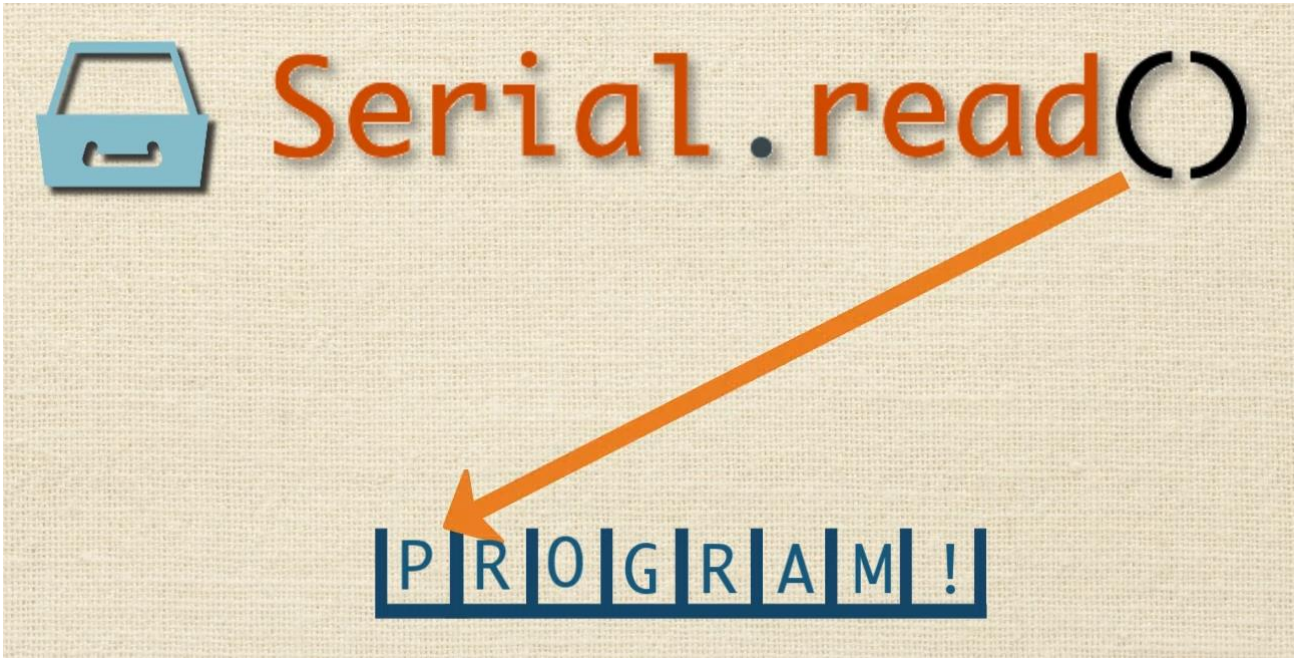
- ใน Arduino จะมี Receive Buffer ขนาด 64 ไบต์ แปลว่าฝั่งตรงข้ามสามารถจะส่งข้อมูลมาได้ถึง 64 ไบต์ จึงจะเต็ม Buffer และรับไม่ได้อีก



Serial Function

- การนำข้อมูลออกจาก Receive Buffer จะใช้ฟังก์ชัน ชื่อ Serial.read()
- เมื่อเรียกใช้ Serial.read() เป็นการนำข้อมูลไบต์แรกของ Receive Buffer ออกไป

```
char myFirstCharacter = Serial.read();
```



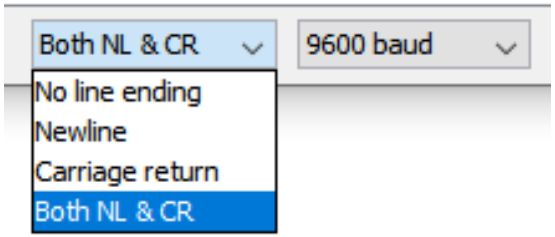
Serial Function

Serial.available()

- คืนค่าจำนวนไบต์ (ตัวอักษร) ที่อยู่ใน Receive Buffer ที่พร้อมจะให้อ่านออกมา จากตัวอย่าง จะใช้ร่วมกับ Serial.read() โดยฟังก์ชัน read() จะส่งค่าไบต์แรกที่สามารถอ่านออกมาได้ (-1 ถ้าไม่มีข้อมูลให้อ่าน)

```
while (Serial.available()) // recheck serial is available
{
  char inChar = (char)Serial.read(); // get the new byte:
}
```

- กรณีที่รับส่งข้อมูลเป็นประโยค จะปิดท้ายข้อความด้วย CR/LF ทั้งนี้ขึ้นกับการตั้งค่าใน Serial Monitor ด้วย



Serial Function

Example:

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

void loop()
{
     while (Serial.available()) // recheck serial is available
     {
          char inChar = (char)Serial.read();
          Serial.print(inChar);
     }
 }
```

Activity

- เขียนโปรแกรมรับข้อมูลจาก Serial Monitor โดยรับเป็นตัวเลข 1-9
- สั่งให้ไฟที่ขา 13 กระพริบตามจำนวนตัวเลขที่รับมา

Serial Function

- โปรแกรมรับข้อมูลครั้งละ 1 บรรทัด

```
static char message[MAX_MESSAGE_LENGTH];
static unsigned int message_pos = 0;

while (Serial.available() > 0)
{
    char inByte = Serial.read();

    if ( inByte != '\n' && (message_pos < MAX_MESSAGE_LENGTH - 1) )
    {
       message[message_pos] = inByte;
       message_pos++;
    }
    else
    {
      message[message_pos] = '\0';
      Serial.println(message);
      message_pos = 0;
    }
}
```

Serial Event

- นอกเหนือจากการวนลูปเพื่อตรวจสอบข้อมูลที่ส่งเข้ามาแล้ว ยังใช้กลไกการ Interrupt เพื่อตรวจสอบได้เช่นเดียวกัน

```
void setup()
{
    Serial.begin(9600); // initialize serial:
    pinMode(13,OUTPUT);
}
void loop()
{
    Serial.println("Wait for command");
    delay(500);
}
void serialEvent()
{
    while(Serial.available()) {
        char inChar=(char)Serial.read(); // get the new byte:
        if(inChar=='1') {                // check received 'enter' (0x0D)
            digitalWrite(13,HIGH);
        }
        else if(inChar=='0') {
            digitalWrite(13,LOW);
        }
    }
}
```

Software Serial

- ในบอร์ด Arduino จะมี Serial (ที่เป็น Hardware) อยู่เพียงชุดเดียว ทำให้ต้องเลือกว่า
  - ต่อกับ Computer เพื่อ Upload หรือใช้ Serial Monitor
  - ต่อกับอุปกรณ์อื่นๆ
- จะใช้กันพร้อมกัน 2 ฟังก์ชันไม่ได้
- จึงมีการพัฒนาซอฟต์แวร์ที่สามารถทำให้ขา GPIO ใดๆ ใช้เป็น Serial ได้
- การใช้งานโดยทั่วไปจะคล้ายกับ Hardware Serial แต่ความแน่นอนจะต่ำกว่าเล็กน้อย โดยเฉพาะที่ความเร็วสูง

Software Serial

```
#include <SoftwareSerial.h>
String inputString = ""; // a string to hold incoming data
SoftwareSerial mySerial(10,11); // SoftwareSerial(rxPin, txPin)

void setup()
{
     // Open serial communications and wait for port to open:
     Serial.begin(4800);
     Serial.println("Hello World");
     // set the data rate for the SoftwareSerialport
     mySerial.begin(4800); // recommend low speed
     mySerial.println("Software Serial->Hello, world?");
}
void loop() // run over and over
{
     if(mySerial.available())
     {
          Serial.print((char)mySerial.read());
     }
}
```

Software Serial

```
void serialEvent()
{
     while(Serial.available()) // recheck serial is available
     {
          char inChar=(char)Serial.read(); // get the new byte:
          inputString+=inChar; // add it to the inputString:
          if (inChar=='\r') // check received 'enter' (0x0D)
          {
                mySerial.print("TX from Software serial -> ");
                mySerial.println(inputString);
                inputString="";
          }
     }
}
```

Software Serial Interrupt

- เป็นการปรับปรุงให้การรับข้อมูลใช้วิธี Interrupt

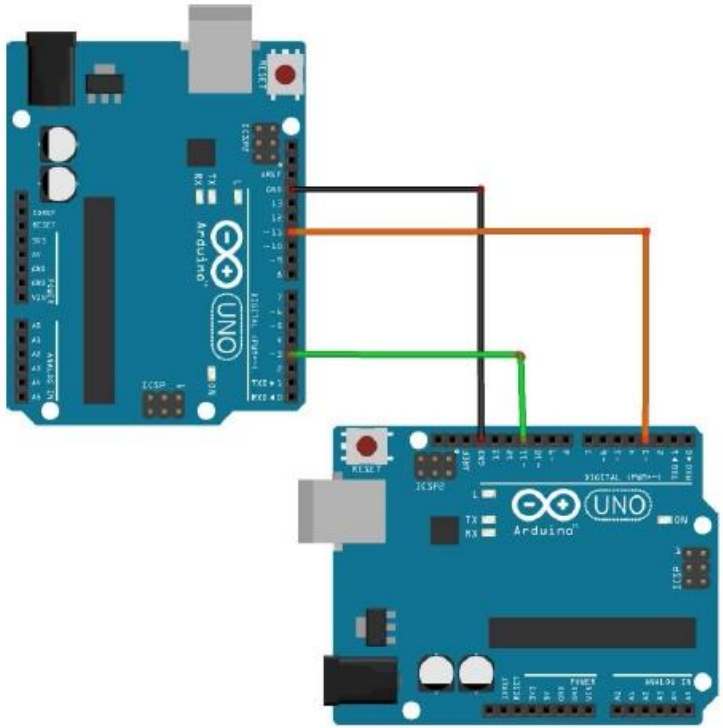
```
#include <SoftwareSerial.h>
String inputString = ""; // a string to hold incoming data
SoftwareSerial mySerial(3,11); // SoftwareSerial(rxPin, txPin)

void setup()
{
    // Open serial communications and wait for port to open:
    Serial.begin(4800);
    Serial.println("Hello World");
    // set the data rate for the SoftwareSerialport
    mySerial.begin(4800); // recommentlow speed
    mySerial.println("Software Serial->Hello, world?");

    // attachInterrupt(interrupt, ISR, mode) interrupt-> 1(pin3)
    attachInterrupt(1,SoftwareSerialEvent,FALLING);
}
void loop() // run over and over
{

}
```

Software Serial Interrupt

```
void serialEvent()
{
     while(Serial.available()) // recheck serial is available
     {
          char inChar=(char)Serial.read(); // get the new byte:
          inputString+=inChar; // add it to the inputString:
          if (inChar=='\r') // check received 'enter' (0x0D)
          {
                mySerial.print("TX from Software serial -> ");
                mySerial.println(inputString);
                inputString="";
          }
     }
}


void SoftwareSerialEvent()
{
      if(mySerial.available()) // test this condition by connecting pin rxsoftware with pin'0'(Rx)
      {
          Serial.print((char)mySerial.read());
      }
}
```

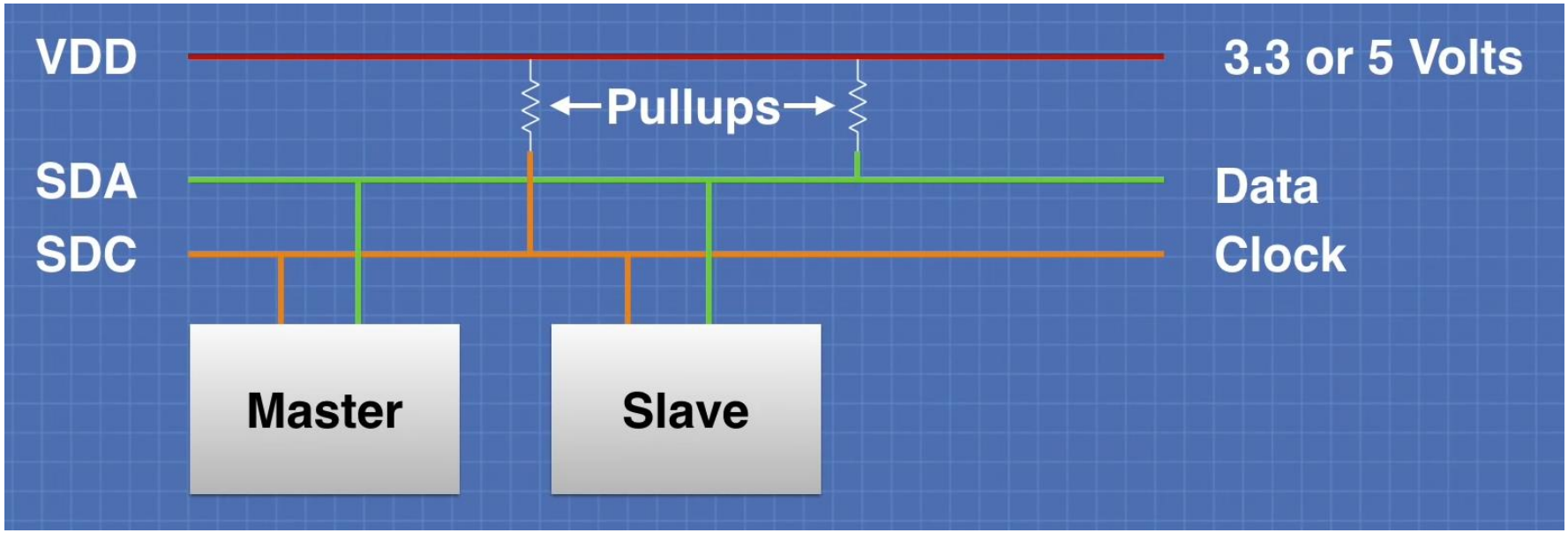
Activity Serial Chat

- ให้นำบอร์ด Arduino 2 บอร์ดต่อกันตามรูป ให้เขียนโปรแกรม Serial Chat โดยป้อนข้อมูลคุยกันระหว่างคอมพิวเตอร์ 2 เครื่อง (ใช้ขา D11,D3) ผ่าน Serial Monitor



การสื่อสารระหว่างบอร์ด โดยใช้ I^2C

- I^2C Bus Communication



การสื่อสารระหว่างบอร์ด โดยใช้ I^2C

- Master จะทำหน้าที่สร้าง clock และไม่มี address
- Master จะทำหน้าที่เริ่มการสื่อสาร
- Slave จะระบุโดย Address ซึ่งมีขนาด 7 บิต (128 devices)
- บางอุปกรณ์ จะสามารถกำหนด address เองได้ แต่โดยทั่วไปจะกำหนดมาแล้ว เช่น OLED = 3ch

การสื่อสารระหว่างบอร์ด โดยใช้ I^2C

- การเชื่อมต่อแบบ I2C ต้องเรียกใช้ Library ชื่อ Wire

```
#include <Wire.h>
```

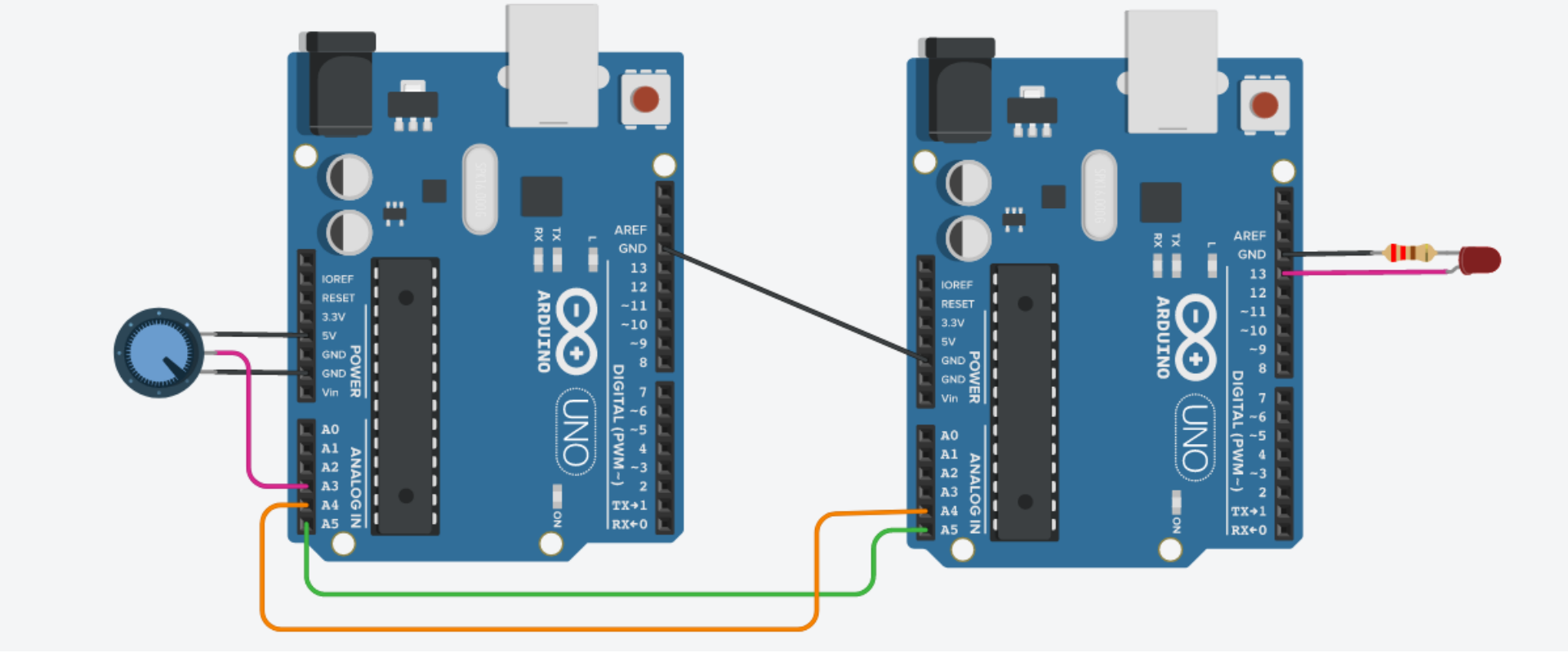
- ฟังก์ชันที่ใช้งานมีดังนี้
  - **Wire.begin()** ใช้ใน setup() เพื่อเริ่มการทำงาน
  - **Wire.beginTransmission(ADDR)** ใช้สำหรับ master เมื่อต้องการส่งข้อมูลให้ Slave
  - **Wire.endTransmission()** ใช้สำหรับ master เมื่อจบการส่ง
  - **Wire.write(value)** ใช้ในการส่งข้อมูลไปบน Bus

การสื่อสารระหว่างบอร์ด โดยใช้ I^2C

- ฟังก์ชันที่ใช้งานมีดังนี้
  - **`Wire.requestFrom(ADDR,size)`** ใช้ใน master เพื่อแจ้งให้ slave ส่งข้อมูล
  - **`Wire.available()`** ใช้ทั้ง master และ slave เพื่อตรวจสอบข้อมูลใน receive buffer
  - **`Wire.read()`** ใช้สำหรับอ่านข้อมูลจาก receive buffer
  - **`Wire.onReceive(handler)`** ใช้กำหนด function ที่จะเรียกขึ้นมาทำงานเมื่อฝั่ง slave ได้รับข้อมูลจาก master
  - **`Wire.onRequest(handler)`** ใช้กำหนด function ที่จะเรียกขึ้นมาทำงานเมื่อฝั่ง slave ได้รับการร้องขอข้อมูลจาก master

การสื่อสารระหว่างบอร์ด โดยใช้ I^2C

- ต่อวงจรตามรูป (A4 (SDA) <-> A4, A5 (SCL) <-> A5



การสื่อสารระหว่างบอร์ด โดยใช้ I²C

Master

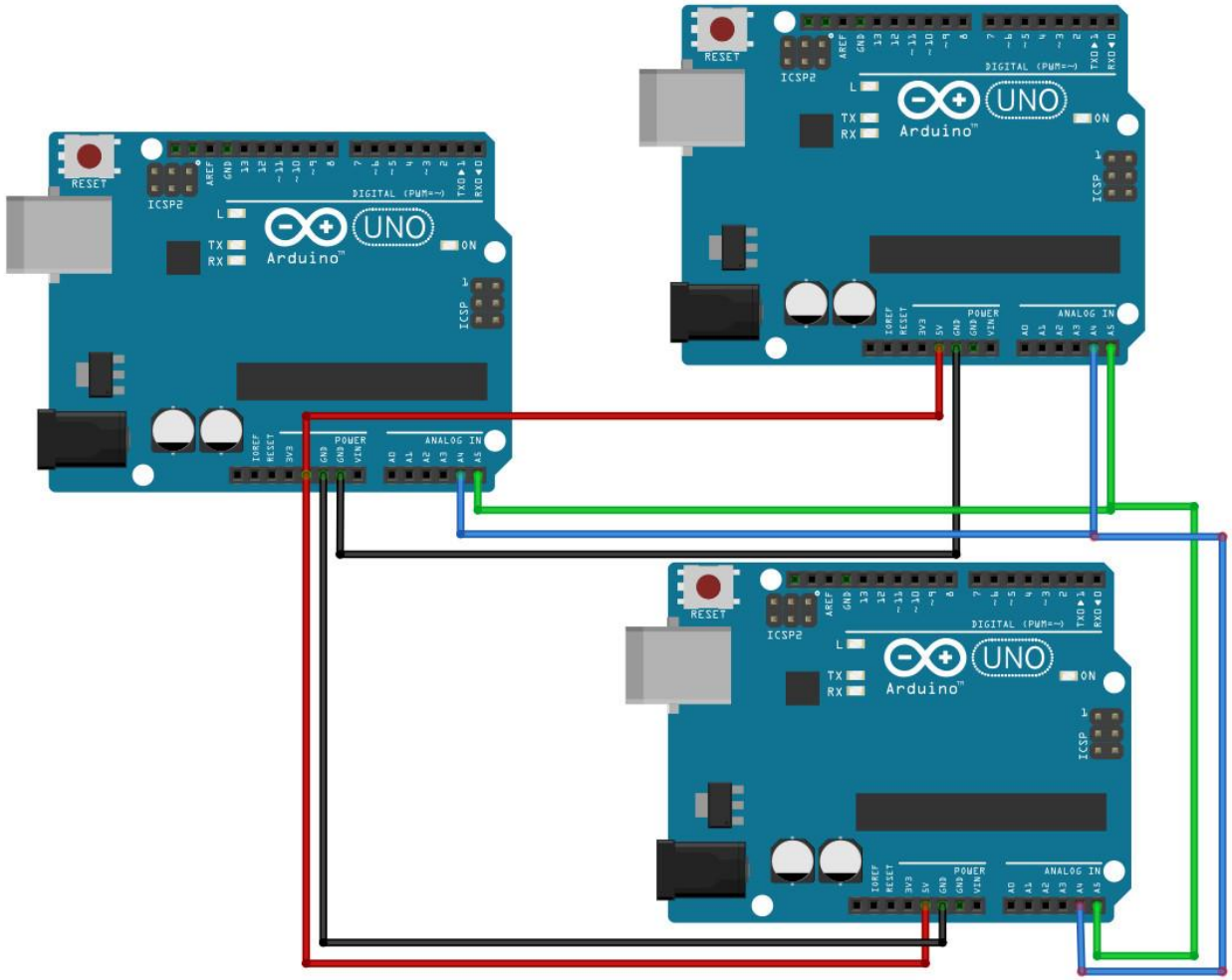
```
#include <Wire.h>
#define SLAVE_ADDR 9
int analogPin = A3;
int val = 0;

void setup() {
    // Initialize I2C communications as Master
    Wire.begin();

}

void loop() {
    delay(50);

    // Read pot value
    // Map to range of 1-255 for flash rate
    val = map(analogRead(analogPin), 0, 1023, 255, 1);

    // Write a charatre to the Slave
    Wire.beginTransmission(SLAVE_ADDR);
    Wire.write(val);
    Wire.endTransmission();

}
```

การสื่อสารระหว่างบอร์ด โดยใช้ I²C

Slave

```
#include <Wire.h>
#define SLAVE_ADDR 9

int LED = 13;
int rd;   // Variable for received data
int br;   // Variable for blink rate

void setup() {
  pinMode(LED, OUTPUT);
  Wire.begin(SLAVE_ADDR);
  Wire.onReceive(receiveEvent);  // Function to run when data received from master
  Serial.begin(9600);
}

void receiveEvent() {
  rd = Wire.read();   // read one character from the I2C
  Serial.println(rd); // Print value of incoming data
}

void loop() {
  delay(50);
   br = map(rd, 1, 255, 100, 2000);    // Calculate blink value
   digitalWrite(LED, HIGH);        delay(br);
   digitalWrite(LED, LOW);    delay(br);
}
```

Assignment #6 : Network chat



Assignment #6 : Network chat

- ให้ทำ 2 กลุ่มร่วมกัน หรือ ยืมบอร์ดจากกลุ่มอื่น
- ต่อวงจรใน Slide ที่แล้ว
- ให้เขียนโปรแกรมจำลองเครือข่าย โดยกำหนดให้มี Token ที่สร้างโดย Master (Arduino #0) โดยส่งข้อความ Token#0# โดย 0 แปลว่า token ว่าง
- จากนั้นให้ส่ง Token ต่อไปที่ Arduino #2 และ Arduino #2 ส่งต่อไปที่ Arduino #3 จากนั้น Arduino #3 ส่งกลับมาที่ Master โดยส่งต่อเป็นวงกลมไปเรื่อยๆ
- ให้มี 2-3 เครื่องที่ต่อกับ PC และรับข้อมูลจาก Serial Monitor โดยเมื่อมีการส่งข้อมูล ให้รอจนได้ Token แล้วส่งข้อความ Token#1#n#message โดย 1 หมายถึง ไม่ว่าง, n คือ เครื่องเป้าหมาย และ message คือ ข้อความที่ต้องการส่ง

Assignment #6 : Network chat

- เมื่อเครื่องเป้าหมายได้รับข้อความ ให้แสดงข้อความออกที่ serial monitor แล้ว reset token ให้เป็น Token#0# อีกครั้ง
- กรณีที่ใช้ 2 เครื่องในการส่ง สามารถ fix หมายเลข n ของคู่สนทนาได้
- กรณีที่ใช้ 3 เครื่องในการส่ง ให้รับหมายเลขของเครื่องเป้าหมายก่อน แล้วจึงรับข้อความ
- ให้ทำ debug mode โดยให้แสดง message ทั้งหมดที่แต่ละโหนดได้รับด้วย เพื่อให้เห็นการส่งวนรอบ เพื่อใช้ในการส่งงาน

For your attention