تطبيقات ميكاترونيك / 2 /

الجلسة : 4
2023 – 2022
I2C connection Protocol
a) Connecting two Arduino uno boards using I2C protocol

بروتوكول الإتصال المتزامن I2C

يعتبر الـ Inter-integrated Circuit - I2C نظام اتصال تسلسلي serial عالي السرعة وثنائي الاتجاه للربط مع الطرفيات كما هو الحال بالنسبة للـ SPI والـ UART. ويعتبر هذا البروتوكول أبطأ من SPI ولكنه يستخدم عدد أقل من الخطوط للإتصال حيث بالإمكان الإختيار بين أحد السرعات الأربع التالية:

- قياسي - 100 كيلوبت في الثانية
- سريع - 400 كيلوبت في الثانية
- سرعة مضاعفة - 1 ميجابت في الثانية
- سرعة عالية – 3.33 ميجابت في الثانية

الميزة الرئيسية التي تميز I2C عن UART و SPI هي أن ناقل I2C يمكنه دعم العديد من الأجهزة الرئيسية master والفرعية slave على نفس الناقل حيث تتيح لنا القدرة على دعم أجهزة متعددة في ناقل واحد تقليل عدد الأطراف pins الخارجية على المتحكم، مما يقلل من تكلفة وحجم الجهاز. وبإمكان جميع الأجهزة المتصلة أن تعمل إما بالوضع الرئيسي master أو الفرعي slave. بالإضافة إلى أنه بإمكان هذه الأجهزة التبديل بين هذين الوضعين.

كما أن لكل جهاز متصل معرَف أو عنوان فريد unique address خاص به ومعيّن له مسبقاً، ولذلك يمكن للجهاز الرئيسي تحديد الجهاز الفرعي الذي سيقوم بالاتصال به.

الهاردوير

كل ما نحتاج اليه في هذا البروتوكول هما خطان:

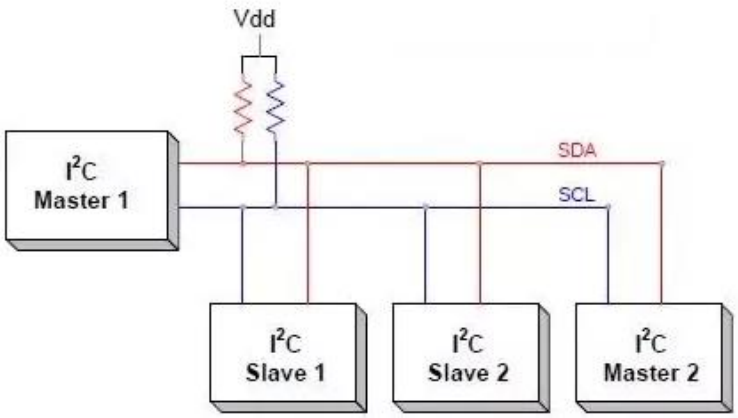
1. SCL - Serial Clock

وهي تنقل إشارة الساعة والتي تنظم وتجدول نقل البيانات بين الأجهزة ويتم توليدها من الجهاز الرئيسي. وبما أن لدينا خط مستقل للساعة فذلك يجعل الـ I2C بروتوكول نقل متزامن synchronous.

2. SDA - Serial Data

هذا الخط مسؤول عن نقل البيانات.

في هذا البروتوكول يتم توصيل جميع الأجهزة الفرعية بشكل متوازي بالخطوط السابقة. فلو كان لديك أكثر من جهاز فرعي slave متصل بجهاز رئيسي واحد master فكل ما عليك فعله هو ربط جميع خطوط الـ SDA معاً وجميع خطوط الـ SCL معاً مما يجعلنا نحتاج الى استخدام طرفين pins فقط لكل جهاز متصل، وهذه ميزة تحسب لصالح هذا البروتوكول.



الخطان من نوع "تصريف مفتوح" open drain مما يعني أنه يجب إيصالهما بمقاومات لأعلى pull up resistors بحيث تكون الخطوط مرتفعة high، وهي الحالة الغير نشطة idle للناقل bus بينما عندما تكون الأجهزة على هذا الناقل نشطة تكون الخطوط منخفضة low. وللإيضاح أكثر، فإن الخطوط من هذا النوع تكون الأطراف المتصله بها غير متصلة بشكل مباشر بمصدر الطاقة VCC وإنما تتصل مباشرة بـ GND.

البروتوكول

نذكر هنا سريعاً بعض الخصائص المتعلقة بهذا البروتوكول:

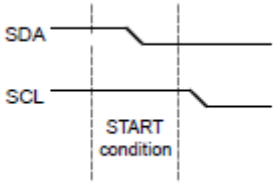
- لكل جهاز فرعي عنوان فريد معين له مسبقاً ويتكون من 7 بت
- خطي الإشارة SCL و SDA هما ثنائي الاتجاه

- يتم إرسال البيانات بشكل متسلسل وتحتوي كل عملية إرسال على 8 بت
- هنالك مجموعة من السرعات التي يمكن الإختيار بينها وتتراوح بين 100 كيلوبت في الثانية وصولًا إلى 3.33 ميجابت في الثانية
- يتم نقل البيانات عن طريق إرسال الـ most significant bit msb أولاً
- البيانات المرسلة تكون محصورة بين إشارتي البدء والتوقف
- يعتبر الناقل bus غير نشط عندما يكونا كلا الخطين مرتفعين high

وفيما يلي سنقوم بذكر تفاصيل الأجزاء المختلفة لهذا البروتوكول:

بت البدء Start

في البداية يكون كل من خطي SCL و SDA في الوضع المرتفع high بسبب مقاومات السحب لأعلى pull-up resistors، وهذا يعني أنه لا يوجد هنالك أي نشاط في الناقل bus. ومن أجل الإشارة إلى بدء عملية نقل البيانات، سيقوم الجهاز الرئيسي بخفض خط SDA بينما يكون خط SCL عالي. وتعتبر هذه إشارة البدء START حيث يمكن إعتبارها تنبيهاً لجميع الأجهزة المتصلة بالناقل حيث أنها تصبح مستعدة للاستماع للجهاز الرئيسي.



العنوان Address

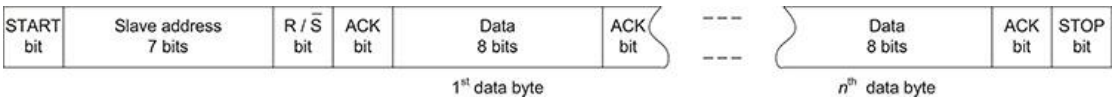
بعد بت البدء START، يقوم الجهاز الرئيسي بإرسال عنوان Address الجهاز الفرعي slave الذي يود التواصل معه وذلك عن طريق إرسال البت الأعلى most significant bit أولاً وإنتهاءاً بالبت الأدنى least significant bit.
ويتكون العنوان من سبعة بتات لأن البت الثامنة هي بت التحكم بالإرسال والإستقبال R/S وتستخدم للإشارة الى ما اذا كان الجهاز الرئيسي سيقوم بالكتابة الى الجهاز الفرعي (0) أم القراءة منه (1) أي هل على الجهاز الفرعي الاستعداد لاستقبال بيانات أم هل هو طلب بيانات معينة من هذا الجهاز الفرعي .
بعد استلام العنوان، ستقوم جميع الأجهزة الفرعية بمقارنة العنوان المستلم من الجهاز الرئيسي بعنوانها الخاص فإذا تطابقت أستمرت في التواصل وإذا لم تتطابق فإنها تستمر في الانتظار حتى يصبح الناقل bus غير نشط وذلك بعد إرسال بت التوقف.

التأكيد ACK

سوف يستجيب الجهاز الفرعي الذي يتطابق معه العنوان بإرسال إشارة تأكيد ACK للإشارة الى أنه قد استلم بنجاح التسلسل السابق للبتات، وبذلك يقوم الجهاز الرئيسي بتحويل مسؤولية خط البيانات SDA الى الجهاز الفرعي.
وإذا ما كان الجهاز الفرعي قد تلقى التسلسل السابق بنجاح فإنه سيقوم بخفض خط SDL وهذا هو التأكيد ACK. أما اذا لم يقم الجهاز الفرعي بذلك فهذه الحالة الغير مؤكدة NACK وتعني بأن الجهاز الفرعي لم يستلم البتات السابقة بشكل صحيح لأي سبب كان.

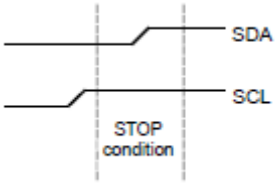
إرسال/استقبال البيانات

بعد تلقي إشارة ACK سيقوم الجهاز الرئيسي إما بنقل أو استقبال البيانات حسب إشارة التحكم R/S المرسلة بعد عنوان الجهاز الفرعي. في حالة الإرسال، سيقوم الجهاز الرئيسي بإرسال البيانات بايت واحد في كل مرة ويستجيب الجهاز الفرعي بإرسال إشارة ACK بعد استلام كل بايت من البيانات. وعندما ينتهي الجهاز الرئيسي من إرسال جميع البيانات سيقوم بإرسال إشارة التوقف STOP.
أما في حالة أن الجهاز الرئيسي سيستقبل بيانات، فإن الجهاز الفرعي سيقوم بإرسال بايت البيانات الأول بعد إرسال إشارة ACK وسيستجيب الجهاز الرئيسي بإرسال إشارة ACK بعد استلام كل بايت من البيانات.



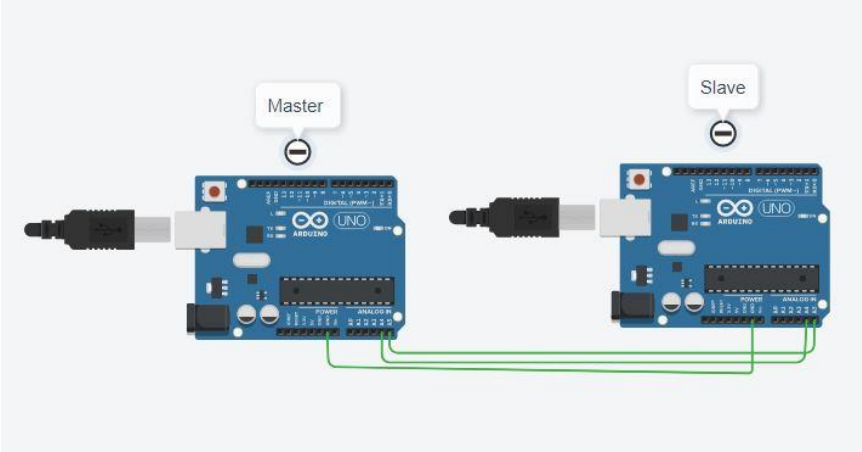
التوقف Stop

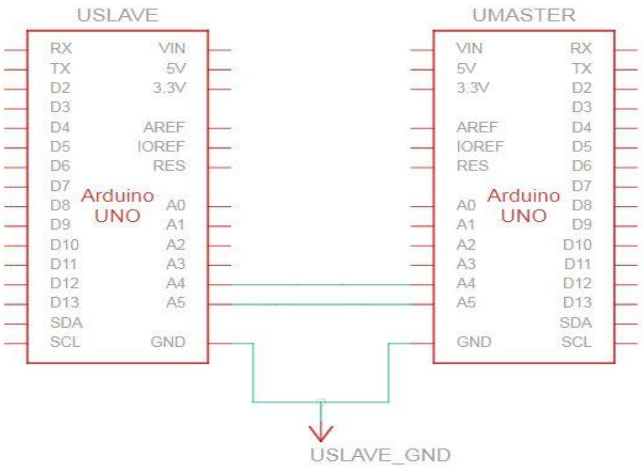
بمجرد أن يتلقى الجهاز الرئيسي جميع البيانات فإنه سيرسل إشارة NACK متبوعة بإشارة STOP لإخلاء الناقل لأي عملية نقل بيانات أخرى. وتحدث إشارة التوقف عندما ينتقل خط SDA من الوضع المنخفض ٠ الى الوضع المرتفع 1 بينما يكون خط الـ SCL مرتفع .



مثال عملي: ***Connecting two Arduino Uno boards using I2C protocol***

دارة المشروع:





الكود:

```
/* Master Code */
/*************/
#include <Wire.h>

void setup(){
       Serial.begin(9600);          /* begin serial comm/*.
       Wire.begin();                   /* join i2c bus as master/*
      Serial.println("I am I2C Master");
}

void loop(){
       Wire.beginTransmission(8);          /* begin with device address 8/*
       Wire.write("Hello Slave");            /* sends hello string/*
       Wire.endTransmission();               /* stop transmitting/*
```

```
        Wire.requestFrom(8, 9);             /* request & read data of size 9
                                                    from slave/*

        while( Wire.available() ) {
               char c = Wire.read();              /* read data received from
                                                        slave/*

              Serial.print(c);
        }
        Serial.println();
        delay(900);
       }
```

```
                                /* Slave Code */
                                /*************/

#include <Wire.h>

void setup(){
       Wire.begin(8);                                  /* join i2c bus with
address 8/*
        Wire.onReceive(receiveEvent);              /* register receive event/*
       Wire.onRequest(requestEvent);              /* register request event/*
        Serial.begin(9600);                             /* start serial comm/*.
        Serial.println("I am I2C Slave");
       }

void loop(){
        delay(100);
       }


//function that executes whenever data is received from master
void receiveEvent(int howMany){
       while ( 0 < Wire.available () ){
                char c = Wire.read();                   /* receive byte as a
character/*
                Serial.print(c);                        /* print the character/*
              }
       Serial.println();                                /* to newline/*
       }

//function that executes whenever data is requested from master
void requestEvent(){
Wire.write("Hi Master");    /*send string on request/*
}
```

إعداد :
م. علي علي

---------- (نهاية الجلسة) ----------