

به نام خدا



درس مدارهای واسط

استاد:

دکتر امین فصحتی

پروژه

نویسندهان:

یوسف سدیدی | 401171111

مهرشاد دهقانی | 401105912

امیرعلی شیخی | 401106158

کیهان مسعودی | 401101111

زمستان 1403

مقایسه عملکرد چند واسط ارتباطی در سیستم‌های نهفته

مقدمه

در این پژوهش، قصد داریم عملکرد چندین واسط ارتباطی را برای انتقال داده‌ها بین دو برد آردوینو مقایسه کنیم. این واسطه‌ها شامل One-Wire، I2C، SPI، UART، Wi-Fi، CAN و میکروکنترلرهای مانند سنسورها، EEPROM و نمایشگرهای LCD هستند. در ادامه، هر پروتکل به صورت جداگانه توضیح داده شده و نحوه اتصال دو برد آردوینو به یکدیگر شرح داده می‌شود. کدهای مربوط به تمام موارد به پیوست این فایل موجود است.

۱. پروتکل I2C

معرفی

پروتکل I2C (Inter-Integrated Circuit) یک روش ارتباطی سریال دو سیمه است که برای انتقال داده بین میکروکنترلرهای و قطعات جانبی مانند سنسورها، EEPROM و نمایشگرهای LCD مورد استفاده قرار می‌گیرد. این پروتکل دارای یک خط داده (SDA) و یک خط کلک (SCL) است که برای همگام‌سازی انتقال داده به کار می‌روند. هر دستگاه متصل به باس I2C دارای یک آدرس منحصر به فرد است که امکان ارتباط چندین دستگاه را فراهم می‌کند. I2C دارای حالت‌های Master-Slave است که در آن، دستگاه Master کنترل ارتباط را در اختیار دارد. این پروتکل می‌تواند در نرخ‌های انتقال ۱۰۰ کیلوبیت، ۴۰۰ کیلوبیت، ۱ مگابیت و ۳.۴ مگابیت بر ثانیه عمل کند. سادگی سیم‌کشی، هزینه کم و مصرف توان پایین از مزایای این پروتکل هستند.

ویژگی‌ها

- ارتباط چندگانه (Multi-Master, Multi-Slave)
- استفاده از فقط دو سیم
- قابلیت آدرس‌دهی ۱۲۷ دستگاه روی یک باس
- پشتیبانی از سرعات‌های ۱۰۰ کیلوبیت، ۴۰۰ کیلوبیت و ۳.۴ مگابیت بر ثانیه

نحوه اتصال دو برد آردوینو

SDA به SDA (A4) -

SCL به SCL (A5) -

GND به GND-

مراحل پیاده سازی

۱. تنظیم برد برای ارسال داده

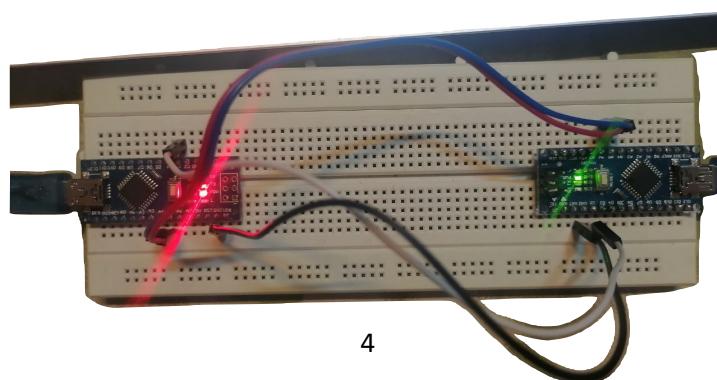
۲. تنظیم برد برای دریافت داده

۳. ارسال و دریافت داده با استفاده از توابع Wire.h

تصاویر خروجی و کدها

The image shows two side-by-side Arduino IDE windows. The left window is titled 'I2C-sender | Arduino IDE 2.1.0' and contains the code for the I2C sender. The right window is titled 'I2C-receiver | Arduino IDE 2.1.0' and contains the code for the I2C receiver. Both windows have their serial monitors open, showing the transmission and decryption of a 128-character message.

```
I2C-sender.ino
File Edit Sketch Tools Help
Arduino Nano
I2C-sender.ino
37
38 digitalWrite(PULSE_PIN, HIGH);
39 delayMicroseconds(100);
40 digitalWrite(PULSE_PIN, LOW);
41
42 int encryptedLength = encryptMessage((byte*)message, strlen(message), encrypted);
43
44 Serial.println("Sending encrypted message in 16-byte chunks...");
45
46 for (int i = 0; i < 128 ; i += 16) {
47     Wire.beginTransmission(8);
48     Wire.write(&encrypted[i], 16);
49     Wire.endTransmission();
50 }
51
52 Serial.println("Data sent via I2C.");
53 delay(4000);
54 }
55
56 int encryptMessage(byte* input, int inputLength, byte* output) {
57     byte local_iv[16];
I2C-receiver.ino
File Edit Sketch Tools Help
Arduino Nano
I2C-receiver.ino
47
48 void receiveEvent(int howMany) {
49
50     while (Wire.available()) {
51         char c = Wire.read();
52         receivedEncrypted[receivedIndex++] = c;
53     }
54
55     if (receivedIndex == 128) {
56
57         byte decrypted[156];
58         int decryptedLength = decryptMessage(receivedEncrypted, receivedIndex, decrypted);
59
60         unsigned long receiveTime = micros();
61         unsigned long transmissionTime = receiveTime - pulseTime;
62         Serial.print("Transmission Time: ");
63         Serial.println(transmissionTime);
64
65         if (decryptedLength > 0) {
66             Serial.print("Decrypted message: ");
67             Serial.println((char*)decrypted);
}
Output Serial Monitor x
Message (Enter to send message to 'Arduino Nano' on 'COM3') New Line 9600 baud
Transmission Time: 21736
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 21744
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 22768
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 21736
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 21740
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Output Serial Monitor x
Message (Enter to send message to 'Arduino Nano' on 'COM4') New Line 9600 baud
Ln 48, Col 33 Arduino Nano on COM3 428 AM 2/5/2025
```



۲. پروتکل SPI

معرفی

پروتکل SPI (Serial Peripheral Interface) یک روش ارتباطی سریال پرسرعت است که معمولاً برای ارتباط بین میکروکنترلرها و دستگاه‌های جانبی مانند کارت‌های حافظه، نمایشگرهای LCD و مبدل‌های ADC/DAC به کار می‌رود. این پروتکل از چهار خط MOSI (Master Out Slave In)، MISO (Master In Slave Out)، SCK (Serial Clock) و SS (Slave Select) برای انتقال داده استفاده می‌کند. SPI نسبت به I2C سرعت بالاتری دارد اما تعداد سیم‌های بیشتری نیاز دارد. این پروتکل معمولاً در مواردی که نیاز به انتقال سریع داده است، مانند پردازش تصویر و ذخیره‌سازی اطلاعات، استفاده می‌شود. ارتباط در SPI به صورت Full Duplex است، به این معنی که داده می‌تواند به طور همزمان ارسال و دریافت شود.

ویژگی‌ها

- استفاده از ۴ سیم (MOSI، MISO، SCK، SS)
- پشتیبانی از سرعت‌های بسیار بالا (تا چندین مگابیت بر ثانیه)
- ساختار Master-Slave ساده

نحوه اتصال دو برد آردوبینو

- (D11) MOSI به MOSI -
- (D12) MISO به MISO -
- (D13) SCK به SCK -
- (D10) SS به SS (CS) -
- GND به GND -

مراحل پیاده‌سازی

۱. تنظیم برای ارسال داده

۲. تنظیم برای دریافت داده

۳. تبادل داده با استفاده از کتابخانه SPI.h

تصاویر خروجی و کدها

The image shows two side-by-side Arduino IDE windows. The left window is titled 'SPI-master' and the right is 'SPI-slave'. Both are connected to an 'Arduino Nano' board. The master code initializes pins and starts an SPI transaction. The slave code handles the transaction and prints the decrypted message. Below each IDE window is a 'Serial Monitor' window showing the transmitted and received data.

```
/* SPI Master Example */
#include <SPI.h>
const int PULSE_PIN = 10;
const int SS_PIN = 11;

void setup() {
    SPI.begin();
    pinMode(PULSE_PIN, OUTPUT);
    digitalWrite(PULSE_PIN, HIGH);
    SPI.beginTransaction(SPISettings(1000000, MSBFIRST, SPI_MODE0));
}

void loop() {
    digitalWrite(PULSE_PIN, HIGH);
    delayMicroseconds(100);
    digitalWrite(PULSE_PIN, LOW);

    int encryptedLength = encryptMessage((byte*)message, strlen(message), encrypted);
    for (int i = 0; i < 128; i++) {
        Serial.print(encrypted[i], HEX);
        Serial.print(" ");
    }
    Serial.println();
}
```

```
/* SPI Slave Example */
#include <SPI.h>
byte decrypted[156];
int decryptedLength = 0;
bool dataReceived = false;

void setup() {
    if (byteIndex >= 128) {
        byteIndex = 0;
        dataReceived = true;
    }
}

void loop() {
    for (int i = 0; i < 128; i++) {
        while(!(SPSR | _BV(SPIF)));
        receivedEncrypted[i] = SPI.transfer(0x00);
    }

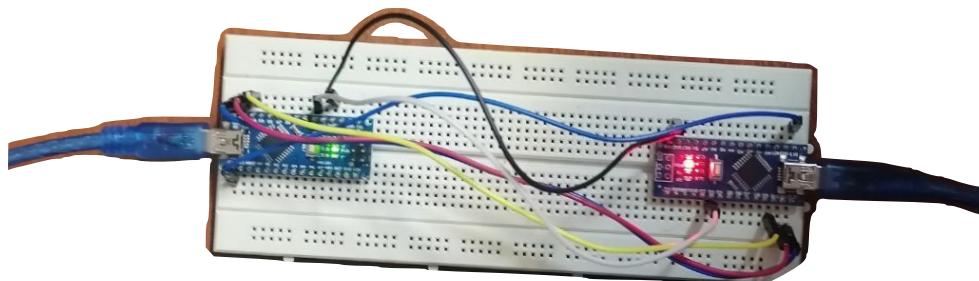
    byte decrypted[156];
    int decryptedLength = decryptMessage(receivedEncrypted, 128, decrypted);

    unsigned long receiveTime = micros();
    unsigned long transmissionTime = receiveTime - pulseTime;
    Serial.print("Transmission Time: ");
    Serial.println(transmissionTime);

    if (decryptedLength > 0) {
        // Process decrypted data
    }
}
```

Message (Enter to send message to 'Arduino Nano' on 'COM3')
New Line 9600 baud
Message sent.
CE 1B 41 E9 C 18 A 69 2D 49 C 2 4A 94 C3 EE 3D 44 1F A9 A2 41 47 6E B9 B3 39 15 20 32 9C 1
Sending encrypted message...
Message sent.
CE 1B 41 E9 C 18 A 69 2D 49 C 2 4A 94 C3 EE 3D 44 1F A9 A2 41 47 6E B9 B3 39 15 20 32 9C 1
Sending encrypted message...
Message sent.

Message (Enter to send message to 'Arduino Nano' on 'COM4')
New Line 9600 baud
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 370776
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 370760
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 370776
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters



۴. پروتکل UART معرفی

پروتکل UART (Universal Asynchronous Receiver-Transmitter) یکی از قدیمی‌ترین روش‌های ارتباطی سریال است که در آن داده‌ها بدون نیاز به کلاک مشترک بین دو دستگاه ارسال و دریافت می‌شوند. در این روش، دستگاه‌های فرستنده و گیرنده باید روی یک نرخ انتقال داده مشخص تنظیم شوند. UART در ارتباطات سریال پایه، ماژول‌های GPS، بلوتوث و ماژول‌های مخابراتی به کار می‌رود. این پروتکل دارای دو خط اصلی TX و RX (Receive) است که داده را ارسال و دریافت می‌کنند. در مقایسه با I2C و SPI، پروتکل UART ساده‌تر است اما نیاز به تنظیم دقیق نرخ انتقال داده دارد.

ویژگی‌ها

- استفاده از دو سیم (RX و TX)
- عدم نیاز به سیگنال کلاک
- نرخ تبادل داده قابل تنظیم (۱۱۵۲۰۰، ۹۶۰۰، ۵۷۶۰۰ بیت بر ثانیه)

نحوه‌ی اتصال دو برد آردوبینو

- از برد اول به RX برد دوم
- از برد اول به TX برد دوم
- GND به GND -

مراحل پیاده‌سازی

۱. تنظیم سرعت و مقدار داده در دو برد
۲. ارسال و دریافت داده با استفاده از `()Serial.begin`
۳. نمایش داده‌ها در Serial Monitor

تصاویر خروجی و کدها

```

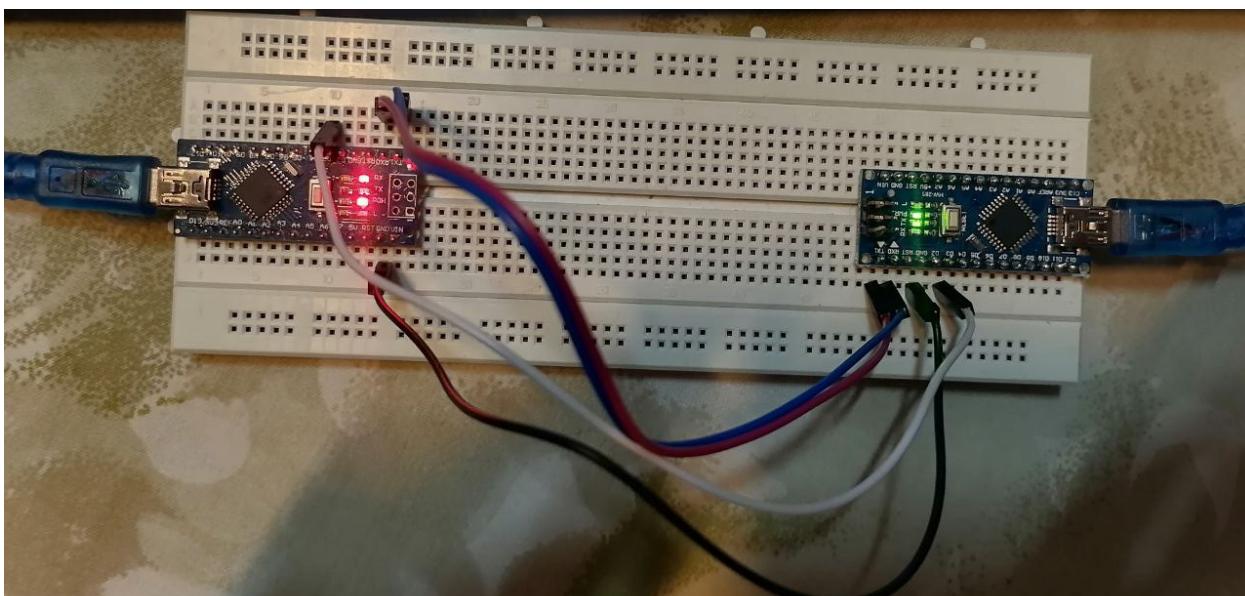
UART-sender.ino
33 }
34
35 void loop() {
36     // Send the entire encrypted message over UART
37     digitalWrite(PULSE_PIN, HIGH);
38     delayMicroseconds(100);
39     digitalWrite(PULSE_PIN, LOW);
40
41     // Encrypt the message
42     int encryptedLength = encryptMessage((byte*)message, strlen(message), encryptionKey);
43
44     // Serial.println("Sending encrypted message...");
45     Serial.write(encrypted, encryptedLength);
46     // Serial.println("Message sent.");
47
48     delay(1000);
49
50 }
51
52 int encryptMessage(byte* input, int inputLength, byte* output) {
53     byte local_iv[16];
54     memcopy(local_iv, iv, 16);

```

```

UART-receiver.ino
36 // Serial.println("UART Receiver with Decryption");
37
38 }
39
40 void loop() {
41     // Check if data is available on UART
42     if (Serial.available() > 0) {
43         Serial.readBytes(receivedEncrypted, 128);
44
45         for (int i = 0; i < 128; i++) {
46             Serial.print(receivedEncrypted[i], HEX);
47             Serial.print(" ");
48         }
49         Serial.println();
50
51         byte decrypted[128];
52         int decryptedLength = decryptMessage(receivedEncrypted, 128, decrypted);
53
54         unsigned long receiveTime = micros();
55         unsigned long transmissionTime = receiveTime - pulseTime;
56         Serial.print("Transmission Time: ");

```



۴. پروتکل Wi-Fi

معرفی

Wi-Fi یک روش ارتباط بی‌سیم است که از طریق امواج رادیویی داده‌ها را ارسال و دریافت می‌کند. در این پروژه، از مازول ESP8266 برای اتصال برد های آردوینو به شبکه استفاده می‌شود. این پروتکل در اینترنت اشیا (IoT) و سیستم‌های کنترل از راه دور کاربرد دارد. Wi-Fi امکان ارسال داده به برد های دیگر و همچنین اتصال به سرورهای اینترنتی را فراهم می‌کند. این پروتکل از استانداردهای ۲.۴ گیگاهرتز و ۵ گیگاهرتز پشتیبانی می‌کند و بسته به شرایط محیطی برد قابل توجهی دارد.

ویژگی‌ها

- انتقال داده بی‌سیم با برد زیاد
- نرخ انتقال داده بالا (تا چند صد مگابیت بر ثانیه)
- امکان ارتباط بردها با شبکه‌های محلی و اینترنت

نحوه‌ی اتصال دو برد آردوبینو

- استفاده از ماژول ESP8266
- برقراری ارتباط یک برد به عنوان سرور و دیگری به عنوان کلاینت

مراحل پیاده‌سازی

۱. تنظیم SSID و Password شبکه WiFi.h
۲. راه اندازی سرور و کلاینت با استفاده از TCP یا UDP
۳. ارسال و دریافت داده از طریق پروتکل CAN

تصاویر خروجی و کدها

۵. پروتکل CAN

معرفی

پروتکل CAN (Controller Area Network) یک استاندارد ارتباطی سریال است که در سیستم‌های صنعتی و خودرویی برای تبادل داده بین میکروکنترلرهای استفاده می‌شود. این پروتکل به دلیل پایداری بالا، مقاومت در برابر نویز و قابلیت کار در محیط‌های صنعتی پر نویز، کاربرد گسترده‌ای دارد.

ویژگی‌ها

- ارتباط چند دستگاه بدون نیاز به Master

- مقاوم در برابر نویز

- استفاده از دو سیم (CAN_L و CAN_H)

نحوه‌ی اتصال دو برد آردوینو

- استفاده از مازول MCP2515 CAN

برد دوم از برد اول به CAN_H -

برد دوم از برد اول به CAN_L -

- اتصال پایه‌های مازول به پین‌های SPI در آردوینو

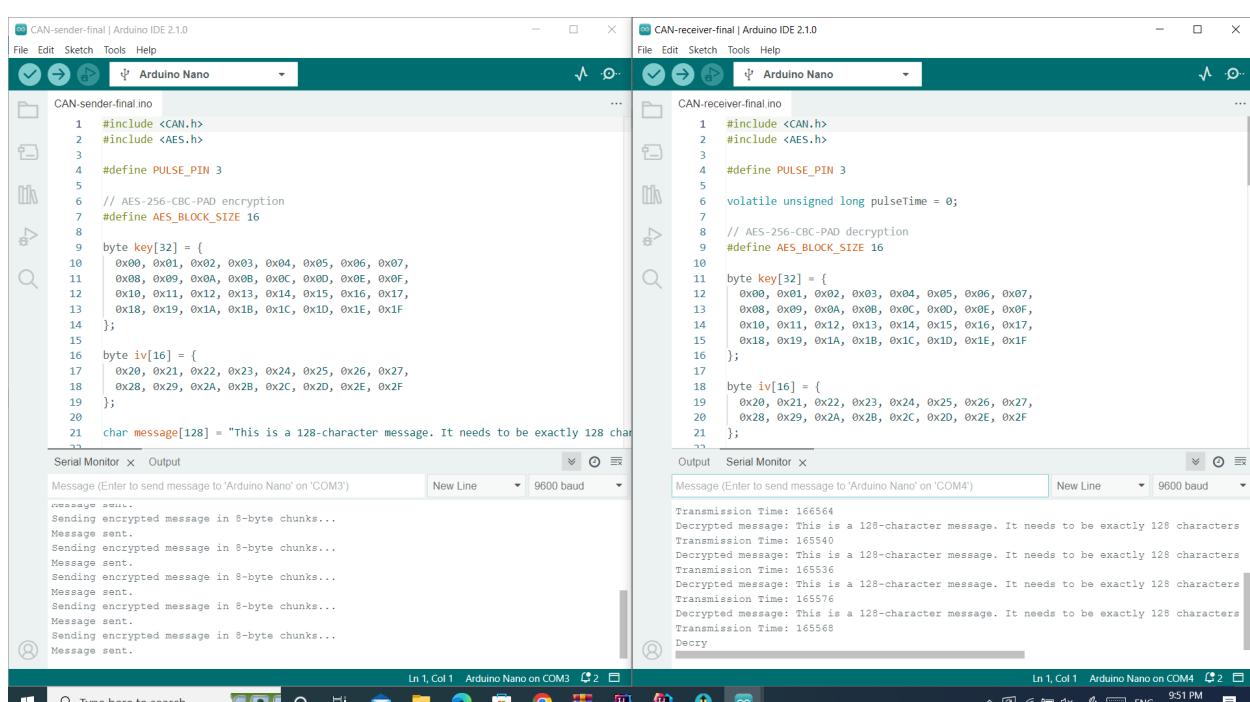
مراحل پیاده‌سازی

۱. مقداردهی اولیه مازول CAN

۲. ارسال پیام از یک برد و دریافت توسط دیگری

۳. پردازش داده‌ها

تصاویر خروجی و کدها



The image shows two side-by-side Arduino IDE windows. The left window is titled 'CAN-sender-final' and the right window is titled 'CAN-receiver-final'. Both windows have the 'Arduino Nano' board selected from the dropdown menu.

CAN-sender-final.ino:

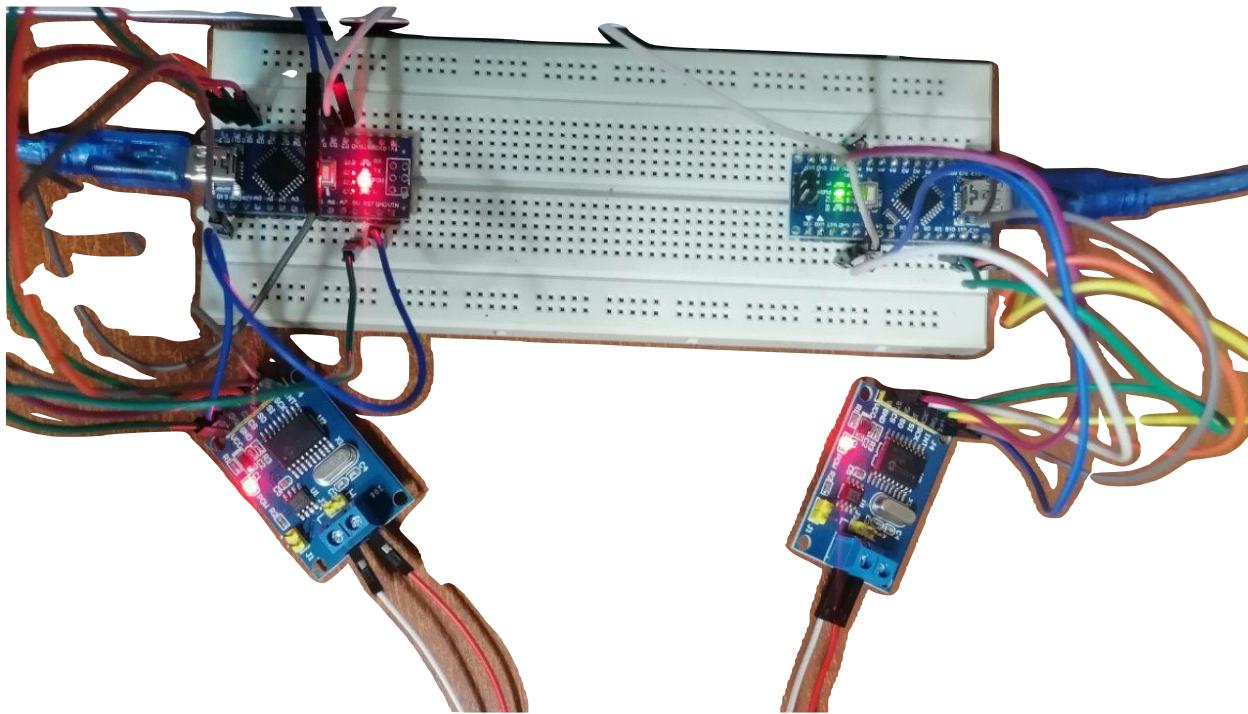
```
1 #include <CAN.h>
2 #include <AES.h>
3
4 #define PULSE_PIN 3
5
6 // AES-256-CBC-PAD encryption
7 #define AES_BLOCK_SIZE 16
8
9 byte key[32] = {
10    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
11    0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,
12    0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
13    0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F
14 };
15
16 byte iv[16] = {
17    0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
18    0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F
19 };
20
21 char message[128] = "This is a 128-character message. It needs to be exactly 128 characters long.";
```

CAN-receiver-final.ino:

```
1 #include <CAN.h>
2 #include <AES.h>
3
4 #define PULSE_PIN 3
5
6 volatile unsigned long pulseTime = 0;
7
8 // AES-256-CBC-PAD decryption
9 #define AES_BLOCK_SIZE 16
10
11 byte key[32] = {
12    0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
13    0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,
14    0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
15    0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F
16 };
17
18 byte iv[16] = {
19    0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
20    0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F
21 };
```

Serial Monitor:

Both serial monitors show the same output: 'Message (Enter to send message to 'Arduino Nano' on 'COM3')' and 'Transmission Time: 165544 Decrypted message: This is a 128-character message. It needs to be exactly 128 characters'. The receiver's baud rate is set to 9600 baud.



۶. پروتکل One-Wire

معرفی

پروتکل One-Wire یک روش ارتباطی سریال ساده و کم‌هزینه است که تنها از یک سیم دیتا به همراه GND استفاده می‌کند. این پروتکل معمولاً در سنسورهای دما مانند DS18B20 و سیستم‌های خواندن RFID به کار می‌رود. One-Wire نیازی به خطوط اضافی مانند کلاک ندارد و به دلیل طراحی ساده، در بسیاری از سنسورها و سیستم‌های نهفته کاربرد دارد.

ویژگی‌ها

- استفاده از یک سیم برای انتقال داده
- قابلیت اتصال چند دستگاه به یک خط
- سرعت پایین (در مقایسه با SPI و I2C)

نحوه اتصال دو برد آردوینو

- یک سیم دیتا بین دو برد متصل شود

- استفاده از مقاومت پول آپ ۴.۷ کیلو اهم

مراحل پیاده سازی

۱. مقداردهی اولیه پروتکل با کتابخانه OneWire.h

۲. ارسال و دریافت داده از طریق یک خط

۳. پردازش اطلاعات دریافت شده

تصاویر خروجی و کدها

The image shows two side-by-side Arduino IDE windows. The left window is titled 'One-sender' and the right is 'One-receiver'. Both are connected to an 'Arduino Nano' board. The 'One-sender' sketch contains code for generating a 128-character encrypted message and sending it via serial. The 'One-receiver' sketch contains code for receiving and decrypting the message. Below each IDE window is its respective 'Serial Monitor' window, which displays the transmitted and received data in hex format. The Windows taskbar at the bottom shows other open applications like File Explorer, Edge, and Google Chrome.

```
One-sender.ino
File Edit Sketch Tools Help
One-receiver.ino
File Edit Sketch Tools Help
One-sender.ino
1 #include <OneWire.h>
2 #include <AES.h>
3
4 const int ONE_WIRE_PIN = 2;
5
6 #define PULSE_PIN 3
7
8 volatile unsigned long pulseTime = 0;
9
10 #define AES_BLOCK_SIZE 16
11
12 char decrypted[128] = "This is a 128-character message. It needs to be exactly 128 characters long";
13
14 byte key[32] = {
15     0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
16     0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F,
17     0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
18     0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F
19 };
20
21 byte iv[16] = {
22     0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
23     0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F
24 };
25
26
27
28
29
30
31
32
33
34
35
36
37
38 digitalWrite(PULSE_PIN, HIGH);
39 delayMicroseconds(100);
40 digitalWrite(PULSE_PIN, LOW);
41
42 int encryptedLength = encryptMessage((byte*)message, strlen(message), encrypted);
43
44 for (int i = 0; i < 128; i++) {
45     Serial.print(encrypted[i], HEX);
46     Serial.print(" ");
47 }
48 Serial.println();
49
50 Serial.println("Sending encrypted message...");
51
52 oneWire.reset();
53 oneWire.write_bytes(encrypted, 128);
54
55 Serial.println("Message sent.");
56 delay(5000);
57 }
58
59 int encryptMessage(byte* input, int inputlength, byte* output) {
60     byte local_iv[16];
61
62     for (int i = 0; i < inputlength; i++) {
63         output[i] = input[i] ^ local_iv[i % 16];
64     }
65
66     return inputlength;
67 }
```

Output: Serial Monitor x

Message (Enter to send message to 'Arduino Nano' on 'COM3') New Line 9600 baud

CE 1B 41 E9 C 18 A 69 2D 49 C 2 4A 94 C3 EE 3D 44 1F A9 A2 41 47 6E B9 B3 39 15 20 32 9C 1
Sending encrypted message...
Message sent.

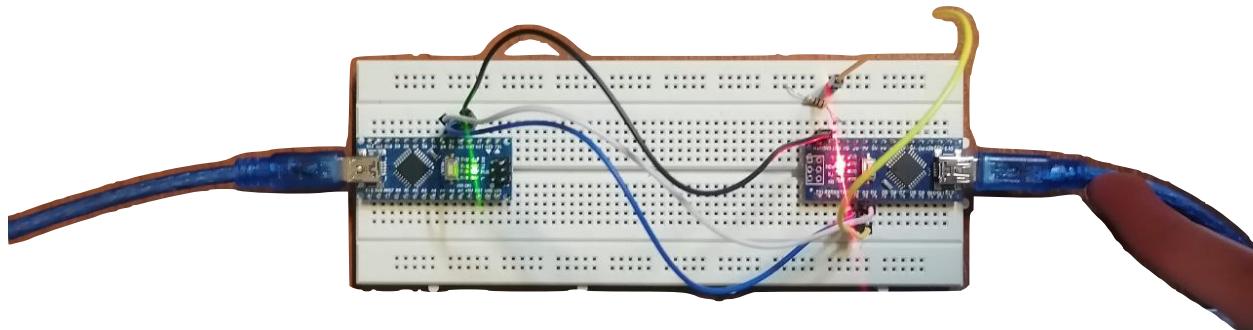
CE 1B 41 E9 C 18 A 69 2D 49 C 2 4A 94 C3 EE 3D 44 1F A9 A2 41 47 6E B9 B3 39 15 20 32 9C 1
Sending encrypted message...
Message sent.

CE 1B 41 E9 C 18 A 69 2D 49 C 2 4A 94 C3 EE 3D 44 1F A9 A2 41 47 6E B9 B3 39 15 20 32 9C 1
Sending encrypted message...
Message sent.

Output: Serial Monitor x

Message (Enter to send message to 'Arduino Nano' on 'COM4') New Line 9600 baud

Transmission Time: 437004
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 436996
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 436996
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 436976
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters
Transmission Time: 436976
Decrypted message: This is a 128-character message. It needs to be exactly 128 characters



Encryption & Decryption

در تمام از کدها از کتابخانه AESLib و توابع آن برای Encryption و Decryption به صورت مشابهی استفاده شده است. کلیدها نیز برای همه یکسان در نظر گرفته شده است.

اندازه گیری زمان

در تمام موارد برای اندازه گیری زمان ارسال از Digital Pin 3 استفاده شده است. به این صورت که فرستنده پیش از شروع کار یک پالس را به واسطه این پین به گیرنده ارسال می کند و در آن لحظه گیرنده زمان را ثبت می کند. سپس انتقال پیام انجام می شود و در نهایت گیرنده مجددا زمان را ثبت می کند. با محاسبه اختلاف این دو زمان transitionTime محاسبه می شود. (دریافت پالس در گیرنده به صورت event driven پیاده سازی شده است.)

نتیجه گیری

در این سند، پروتکل های مختلف ارتباطی مورد بررسی قرار گرفتند و نحوه اتصال دو برد آردوینو برای هر پروتکل شرح داده شد. در ادامه، مقایسه های عملکرد آن ها انجام خواهد شد تا مشخص شود کدام روش برای شرایط مختلف بهتر است.

مقایسه زمان ها:

با توجه به اعداد مشخص شده در تصاویر می توان نتیجه گرفت که پروتکل I2C در بین این پروتکل ها از سرعت بیشتری برخوردار است. پس از آن به ترتیب OneWire ، SPI ، CAN ، UART قرار دارند.

پروتکل WIFI از تمام موارد فوق سریعتر عمل می کند.

توجه کنید که با توجه به کتابخانه های مختلف و توابع و پارامترهای مختلف این زمان ها می توانند متفاوت باشد. برای مثال در UART با تغییر baudRate برای پورت Serial زمان انتقال پیام تغییر چشمگیری خواهد داشت.

