دانشگاه صنعتی شهدای هویزه

موضوع

سیستم آبیاری هوشمند با قابلیت ارسال پیامک

استاد راهنما

دکتر شاکوهی

نگارش

عباس آزاد

آنالوگ به دیجیتال در آردوینو

در این آموزش مفهوم ADC (تبدیل آنالوگ به دیجیتال) را در ARDUINO UNO معرفی می کنیم. همانطور که در شکل زیر نشان داده شده است، برد آردوینو شش کانال ADC دارد. یک یا همه آنها می تواند به عنوان ورودی برای ولتاژ آنالوگ استفاده شود. آنالوگ به دیجیتال در برد Arduini Uno با دقت 10 بیتی (یعنی مقادیر عدد صحیح بین ^ 2) -0) ((1023 10است. به این معنی که ولتاژ های ورودی بین 0 تا 5 ولت را به مقادیر عدد صحیح بین 0 تا 1023 تبدیل میکند. بنابراین برای هر واحد 1024/5 = 4.9 mVاست.

در این آموزش ما پتانسیومتر را به کانال A0 متصل میکنیم و با تغییر آن ، نتیجه تبدیل آنالوگ به دیجیتال را روی ال سی دی کاراکتری نمایش مدهیم.

دستورات آنالوگ به دیجیتال در آردوینو

کانال های Analog Digital Converter دارای مقدار مرجع پیش فرض 5 Vهستند. این بدان معنی است که ما می توانیم حداکثر ولتاژ ورودی 5 ولت را برای تبدیل ADC در هر کانال ورودی قرار بدهیم. از آنجا که برخی از سنسورها ولتاژ های 0-2.5 ولت را ارائه می دهند ، با استفاده از یک ولتاژ مرجع 5 ولت دقت کمتری می گیریم، بنابراین ما یک دستورالعمل داریم که به ما امکان تغییر این مقدار مرجع را می دهد. بنابراین برای تغییر مقدار مرجع ما از دستور زیر استفاده میکنیم. ولتاژ مرجع مورد نظر را در بین پرانتز قرار دهید.

```
("analogReference ()")
```

به صورت پیش فرض ما حداکثر دقت ADC را که در برد آردوینو یونو 10 بیت است دریافت می کنیم، این دقت را می توان با استفاده از دستورالعمل زیر تغییر داد. این تغییر دقت ممکن است برای برخی موارد مفید واقع شود. در دستور زیر بین پرانتز بیت مورد نظر را وارد کنید.

```
(analogReadResolution(bit))
```

حالا اگر نمیخواهید این دو دستور که در بالا گفتیم را تغییر دهید نیاز به وارد کردن آنها در کد نیست و میتوانید از همان مقدار پیشفرض آن استفاده کنید .حالا می توانیم مقدار را از کانال آنالوگ به دیجیتال ‘0″ به صورت مستقیم باتابع

”(analogRead(pin))“بخوانیم ، در اینجا ”pin“ نشانگر پینی است که به آن سیگنال آنالوگ را وصل کردیم، در آموزش ”A0“ است. مقدار ADC را می توان به یک عدد صحیح تبدیل کرد. به دستور العمل زیر دقت کنید، با استفاده از دستور العمل زیر ما مقدار آنالوگ را در متغیر ADCVALUE ذخیره میکنیم.

```
int ADCVALUE = analogRead(A0);
```

توضیحات مدار لازم برای تبدیل آنالوگ به دیجیتال

کاربرد ADC در آردوینو – مبدل آنالوگ به دیجیتال

بعنوان مثال یک سری از سنسور های دما مثل سری LM و به طور خاص LM35 که خیلی پر مصرفه توی پروژه ها ، خروجی ای با ولتاژ آنالوگ دارن و با تغییر دما ، ولتاژ پایه های خروجیشون تغییر میکنه و ما نیاز داریم برای خوندنشون خروجی رو تبدیل به دیجیتال کنیم.

استفاده و راه اندازی واحد ADC در آردوینو یا مبدل آنالوگ به دیجیتال

توی مبدل های آنالوگ به دیجیتال 2 مشخصه داریم که خیلی مهم

1. دقت
2. سرعت

دقت تراشه برای اندازه گیری کمترین مقدار سیگنال آنالوگ مهمه . این مقدار تناسب داره با تعداد بیت های خروجی داده های دیجیتال (زمانی که تبدیل انجام شده(

و سرعت که به فرکانس نمونه برداری مربوط میشه و هرچقدر فاصله زمانی بین نمونه برداری کمتر باشه(همون دوره تناوب) سرعت هم بیشتر میشه.مقدار داده ی دیجیتالی خروجی هم با فرمول زیر حساب میشه:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

VIN:ولتاژ ورودیمونه که قراره تبدیل بشه.

VREF : ولتاز رفرنسه که جلوتر توضیح میدم . ولی درحالت دیفالت و معمول 5 ولته.

طبق قانون نایکوئیست فرکانس نمونه برداری باید حداقل 2 برابر بیشترین فرکانس موجود سیگنال باشه. باید حواسمون باشه به اینکه بالا بودن فرکانس نمونه برداری باعث بالارفتن توان مصرفی میشه و این برای سیستم هایی که قراره با باتری کار کنن خیلی مهمه. مثلا وقتی داریم از یک سنسور دما یا رطوبت استفاده میکنیم زیاد سرعت نمونه برداری ی بالا مهم نیست چون دما و رطوبت توی زمان پایین تغییر وضعیت نمیدن.(برای هر سنسوری میشه زمان بین تغییرات خروجیش رو توی دیتاشیت نگاه کرد(

میکورکنترلر های avr حداکثر 10 بیت دقت دارن و بازه ی نمونه برداری از 0 تا ولتاژ مرجع (فرض میکنیم 5ولت) رو به 1024 قسمت تقسیم می کنند.بنابراین با اضافه شدن حدود 5 میلی ولت به سیگنال آنالوگ یک بیت به مقدار دیجیتال اضافه میشه!(توی فرمول بالا خودتون حساب کنید به این نتیجه میرسین)

ADCدر آردوینو UNO و بقیه ی آردوینو ها هم 10 بیتی ای هست چون هسته avr دارن ، در سری هایی که هسته ARMدارن هم ای مقدار 12 بیته.

فک کنم بهتره بقیه توضیح رو با یک مثال ادامه بدم:

مثلا میخوایم با یک LM35 کار کنیم که یک سنسور دماست، توی دیتاشیت این آی سی که نگاه میکنیم میبینیم که گفته 10 میلی ولت به ازای هر درجه سانتی گراد به خروجیش اضافه میشه.دقت آی سی هم تغریبا 0.5 درجه هست ، یعنی توانایی اندازه گیری 0.5 درجه سانتیگراد رو با دقت خوبی داره، پس به ازای هر 0.5 درجه 5 میلی ولت در خروجی افزایش داریم که برابر افزایش یک بیت در مقدار دیجیتال هست.بنابراین ، مبدل کاملا سیگنال آنالوگ رو ردیابی میکنه و دقت خوبی داره.البته اگه قرار باشه دما رو بدون اعشار نشون بدیم دقت حتی بیش از نیاز ما هم هست

برای انتخاب ولتاژ مرجع هم باید اینو بدونیم که ولتاژ مرجع به دامنه سیگنال آنالوگ ورودی وابسته هستش، آردوینو ، 2

ولتاژ مرجع 5 ولت و داخلی رو بدون نیاز به قطعه خارجی فراهم کرده(دمش گرم)

برای استفاده از ولتاژ های مرجع باید از تابع زیر استفاده کنیم:

تابعanalogReference()

که ورودی اون شامل متغیرهای زیره:

DEFAULT:ولتاژ 5(3.3)

INTERNAL:ولتاژ 1.1 ولت در تراشه هایATMEGA168 & ATMEGA328

INTERNAL1V1:ولتاژ 1.1 ولت تنها در بردهای مگا

INTERNAL3V56:ولتاژ 2.56 ولت تنها در بردهای مگا

EXTERNAL:ولتاژ مرجع دلخواه بین صفر تا 5 ولت در پایهAREF

تابعanalogRead()

برای تبدیل و خوندن داده، تابع analogRead رو استفاده میکنیم ومتغیر ورودی تابع ، شماره پایه ی ورودی هستش و خروجی تابع یک عدد ده بیتی صفر تا 1023هستش که مقدار دیجیتال و متناظر سیگنال آنالوگ در لحظه نمونه برداری هست.

)یک نکته هم اینکه هر تبدیل در مبدل آنالوگ به دیجیتال به جز تبدیل اول ، سیزده پالس ساعت طول میکشه(

تابعanalogReadResolution()

)مخصوص برد های DUE و(ZERO (M0 PRO

یک تابع هم مخصوص برد های DUE و(ZERO (M0 PRO وجود داره ، اول مطلب گفتم که میکروهای AVR دقت 10 بیتی دارن ، اما این برد ها چون هسته ARM دارن دارای دقت بالاتری و 12 بیتی هستند وخروجی بین 0 تا 4095 دارن.البته با این کد میشه دقتی 16 بیتی هم داشت و درکل وظیفه این تابع اینه که دقت خوندن ورودی آنالوگ رو تنظیم میکنه.

اینم یک مثال برای این تابع:

```
1  void setup() {
2
3    // open a serial connection
4
5    Serial.begin(9600);
6
7  }
8
9
10 void loop() {
11
12   // read the input on A0 at default resolution (10 bits)
13
14   // and send it out the serial connection
15
16   analogReadResolution(10);
17
18   Serial.print("ADC 10-bit (default) : ");
19
20   Serial.print(analogRead(A0));
21
22
```

```
23  // change the resolution to 12 bits and read A0
24
25  analogReadResolution(12);
26
27  Serial.print(", 12-bit : ");
28
29  Serial.print(analogRead(A0));
30
31
32  // change the resolution to 16 bits and read A0
33
34  analogReadResolution(16);
35
36  Serial.print(", 16-bit : ");
37
38  Serial.print(analogRead(A0));
39
40
41  // change the resolution to 8 bits and read A0
42
43  analogReadResolution(8);
44
45  Serial.print(", 8-bit : ");
46
47  Serial.println(analogRead(A0));
48
49
50  // a little delay to not hog serial monitor
```

```
51
52  delay(100);
53
54 }
```

کد آنالوگ به دیجیتال درArduino

در ابتدای کد ما کتابخانه ال سی دی کاراکتری را فراخوانی میکنیم. در خط دوم پین های LCD کاراکتری را پیکربندی میکنیم. یعنی پین های .Enable , d4 , d5 , d6 , d7 سپس مقدار دهی اولیه کاراکتر ADC را انجام میدهیم.

```
#include <LiquidCrystal.h> // فراخوانی کتابخانه ال سی دی کاراکتری
LiquidCrystal lcd(8, 9, 10, 11, 12, 13); // پین های ENABLE PIN,D4 PIN,D5 PIN,
D6 PIN, D7 PIN
char ADCSHOW[5]; //سایز کاراکتر آ.ب.د برای نمایش در ال سی دی
```

سپس به تابع void setup میرسیم. همانطور که در قسمت های قبل آموزش آردوینو گفتیم، در این قسمت پیکربندی پروژه را انجام میدهیم. در این پروژه و در این قسمت ما تعداد ستون ها و ردیف های LCD کاراکتری را تنظیم میکنیم.

```
void setup()
{
lcd.begin(16, 2); // تنظیم ستون ها و ردیف های LCD
}
```

سپس نوبت به تابع void loop میرسد. همانطور که قبلا گفتیم در این قسمت ما یک چرخه تکرار شونده را تنظیم میکنیم. دستورات این تابع به طور مداوم انجام میشوند.

همچنین اگر در مورد این مطلب سوالی داشتید در انتهای صفحه در قسمت نظرات بپرسید

در ابتدا ما در خط اول ال سی دی کاراکتری کلمه دلخواه خود را نشان میدهیم .این کلمه میتواند هرچیزی باشد ما از irenx.irاستفاده کرده ایم. در خط بعدی کد ما مکان متن irenx.ir را تنظیم میکنیم. و در خط سوم این کد نیز ، ردیف دوم ال سی دی کاراکتری را تنظیم میکنیم. تا بنویسد : **Natije ADC**

```
void loop()
{
lcd.print("IRENX.IR"); //نمایش نام اولیه
lcd.setCursor(0, 1); // تنظیم مکان نام
lcd.print("Natije ADC:"); //نمایش جمله نتیجه ADC
```

در قسمت بعدی ما در خط اول کد ، مقدار آنالوگ را از پین A0 دریافت میکنیم. در خط دوم با توجه به سایز کاراکتر که در قسمت اول کد انتخاب کردیم این مقدار آنالوگ را به دیجیتال تبدیل میکنیم. و در خط سوم مقدار ADC را در جلوی Natije ADC :نمایش میدهیم. در خط پنجم نیز مکان نمایش ADC را تنظیم میکنیم.

```
String ADCVALUE = String(analogRead(A0)); //دریافت مقدار اولیه آنالوگ
ADCVALUE.toCharArray(ADCSHOW, 5); // تبدیل مقدار خوانده شده به دیجیتال
```

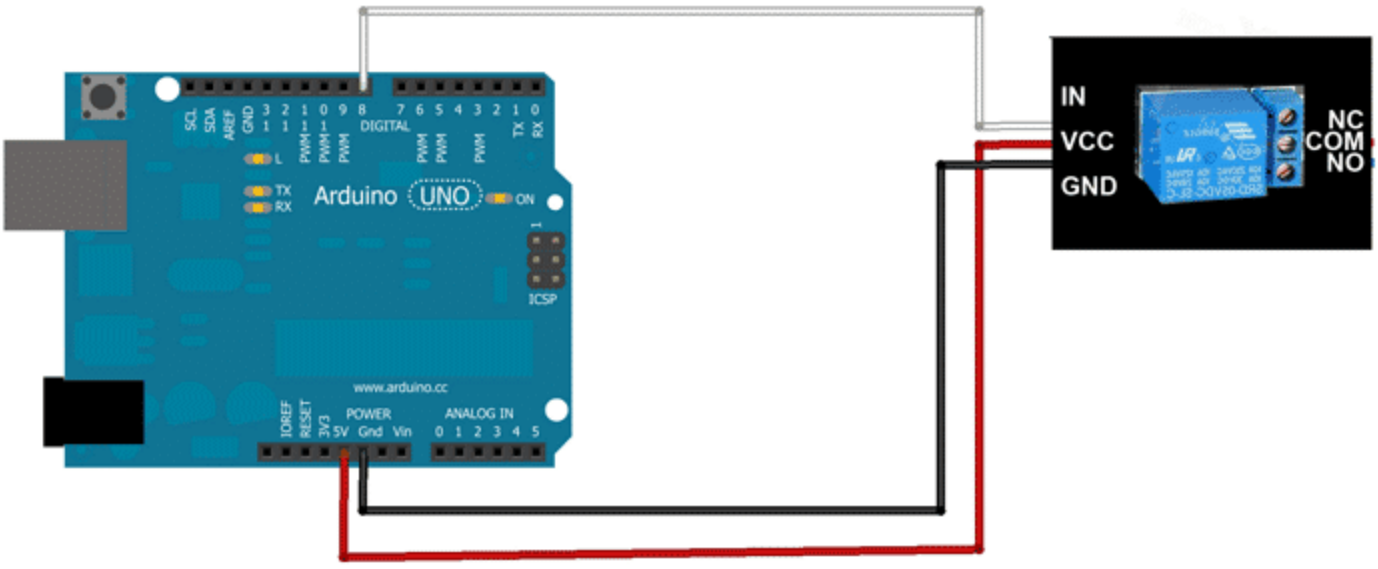
```
lcd.print(ADCSHOW); //نمایش مقدار ADC
lcd.print(" ");
lcd.setCursor(0, 0); // تنظیم مکان نمایش ADC
}
```

در باکس زیر کد کامل به همراه توضیح را میبینید.

```
#include <LiquidCrystal.h> // فراخوانی کتابخانه ال سی دی کاراکتری
LiquidCrystal lcd(8, 9, 10, 11, 12, 13); // پین های ENABLE PIN,D4 PIN,D5 PIN,
D6 PIN, D7 PIN
char ADCSHOW[5]; //سایز کاراکتر آ.ب.د برای نمایش در ال سی دی

void setup()
  {
lcd.begin(16, 2); // تنظیم ستون ها و ردیف های LCD
}
void loop()
  {
lcd.print(" IRENX.IR"); //نمایش نام اولیه
lcd.setCursor(0, 1); // تنظیم مکان نام
lcd.print("Natije ADC:"); //نمایش جمله نتیجه ADC
String ADCVALUE = String(analogRead(A0)); //دریافت مقدار اولیه آنالوگ
ADCVALUE.toCharArray(ADCSHOW, 5); // تبدیل مقدار خوانده شده به دیجیتال
lcd.print(ADCSHOW); //نمایش مقدار ADC
lcd.print("    ");
lcd.setCursor(0, 0); // تنظیم مکان نمایش ADC
{
```

راه اندازی رله با آردوینو

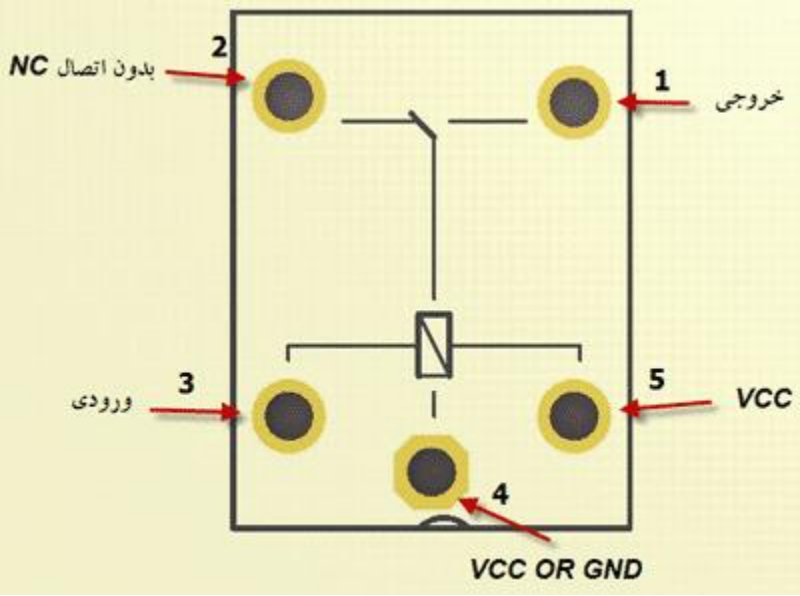


قبل از اینکه بریم سراغ راه اندازی رله با آردوینو بهتره که اول تعریف کنیم این قطعه چیه، عموما در بحث راه اندازی قطعات با آردوینو از ماژول ها یا شیلد ها استفاده میشه ، در این بخش هم ما ابتدا توضیحی درباره ی رله و خود این قطعه میدیم و بعد میریم سراغ ماژول رله 5 ولتی.

رله Relay چیست ؟

رله یک کلید یا سوئیچ الکترونیکیه ، مثل کلیدی که لامپ رو باهاش روشن میکنین! فقط این قطعه با اعمال ولتاژ این کار رو براتون انجام میده و مشخصه که خیلی قطعه پرکاربردی میتونه باشه. مثلا وقتی که طبق برنامه ی خاص و دقیقی قرار باشه موتوری روشن/خاموش بشه یا به وسیله پیامک چراغی روشن/خاموش بشه میشه از رله استفاده کرد و...

عموما رله ها 4 یا 5 پایه هستن ، که اینجا بیشتر هدف مون روی معرفی پرکاربرد ترین رله بازار ینی رله 5 ولتی (مدل 12 ولتی هم تفاوتی توی ساختار نداره) و 5 پایه هستش که به شکل زیره.



-1پایه یک خروجی ما هستش که اون رو به LED ، موتور یا هر وسیله ای که قراره روشن بشه وصل میکنیم. این پایه در حالت عادی خاموشه. یا اصطلاحا N.C (Normally closed) هستش.

-2این پایه مثل پایه یکه فقط در حالت عادی همیشه وصله و با اعمال فرمان خروجی قطع میشه.این پایه اصطلاحا N.O (Normally open)هستش.

اگر این پایه رو استفاده نکنیم بدون اتصال باقی میمونه

ممکنه بخوایم با وصل شدن یک وسیله به برق، همون لحظه وسیله روشن بشه، در این صورت از پایه 2 استفاده میکنیم. اما بیشتر مواقع یک وسیله رو به برق میزنیم و تا فرمان روشن شدن بهش ندیم (مثلا یک موتور) انتظار نداریم که روشن بشه! در نتیجه بیشتر از پایه یک استفاده میکنیم(ما هم در ادامه از پایه یک استفاده میکنیم!)

-3پایه 3 ، پایه ای هستش که قراره ما به اون فرمان بدیم، یعنی با صفر و یک شدن اون تغییر وضعیت در خروجی ایجاد میشه.

-4برای قطع و وصل کردن جریان led یا موتور ، کافیه از سمت سیم فاز یا نول سیم رو قطع کنیم و کلید رو در اون جا قرار بدیم ، این پایه هم همین کاربرد رو داره.

یعنی شما میتونین پایه VCC یا GND رو به این پایه وصل کنید و در صورت اعمال فرمان به رله، در خروجی رله، VCC یا GND رو دریافت کنید و به موتور یا LED بدین ! البته توصیه میشه که سعی کنید VCC رو به این پایه

بدین ...

-5پایه پنج هم VCC مدار وصل میشه و با پایه 3 یک میدان مغناطیسی برای اتصال داخلی رله فراهم می کنه تا فرمان اجرا بشه.

توجه:بین پایه 3 و 5 یک دیود قرار می گیره تا از جریان برگشتی جلوگیری بشه(پایه 3(کاتد) و پایه 5(آند))

راه اندازی ماژول رله با آردوینو

خب توضیحات تا اینجا درباره ی قطعه ی رله بود ، اما ماژول هایی هستنن که کار مدار بستن رو هم ساده تر میکنن! ماژول رله پایه ها به شرح زیره:

پایه ی :NC که همون پایه ی Normally closed هستش

پایه ی :NO که همون پایه ی Normally open هست.

پایه ی :C پایه ی مشترک) ولتاژ VCC یا GND ای که قراره به LED موتور یا .. بره رو به این وصل میکنیم(!

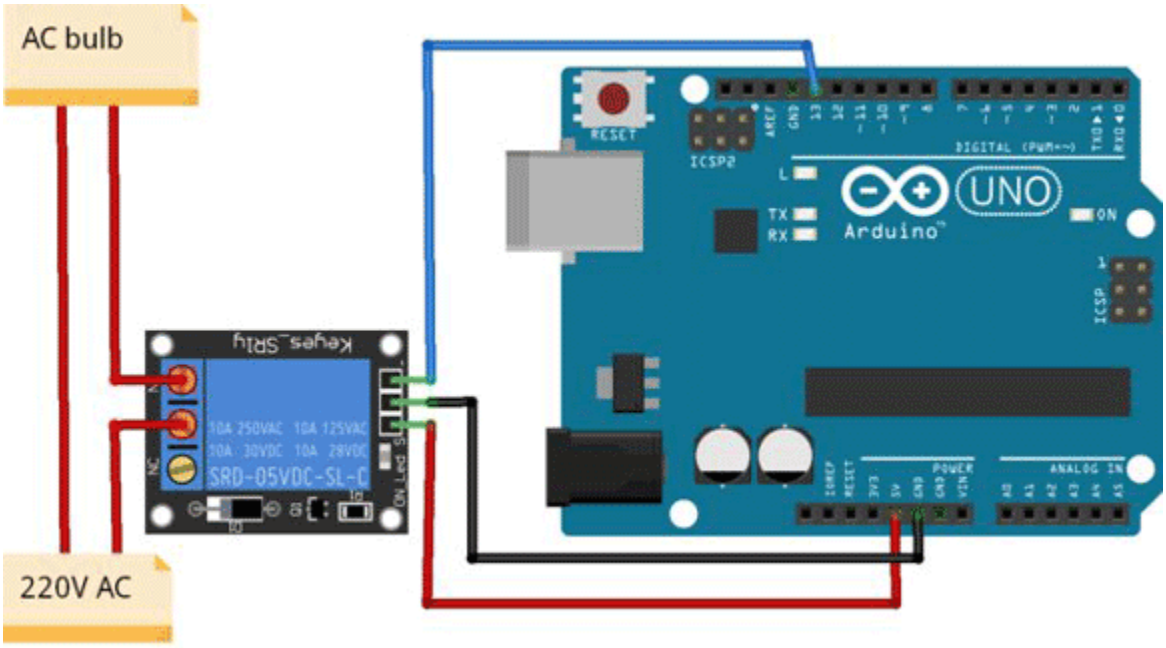
پایه ی :Ground به GND آردوینو وصل میشه.

پایه ی V Vcc: 5به 5 ولت آردوینو وصل میشه.

پایه ی :Signal با فرمان آردوینو به این پایه خروجی تغییر وضعیت میده.(به پایه های دیجتال وصل میشه)

بعضی از ماژول ها یک جامبر دارن که پایه های H و L داره ، اون پایه برای تعیین وضیعت نوع تحریک رله هستش! حالا این یعنی چی؟ یعنی اینکه اگر پایه ی Signalیک شد رله عمل کنه یا اگر صفر شد؟ که میشه با جامپر تعیین کرد روی کدوم حالت ، رله تغییر وضیعت بده.

پروژه ی آردوینو با رله




```
1  void setup()
2  {
3    pinMode(13, OUTPUT);
4  }
5
6  void loop() {
7    digitalWrite(13,LOW);
8    delay(4000);            // wait for 5 seconds
9    digitalWrite(13,HIGH);
10   delay(4000);            // wait for 5 seconds
11 }
```

و پروژه دوم رله با آردوینو:

```
1  int relay=13;
2
3  void setup() {
4    // put your setup code here, to run once:
5  pinMode(13, OUTPUT);
6  Serial.begin(9600);
7  }
8
9  void loop() {
10  if(Serial.available()>0){
11    int dat=Serial.read()-48;
12    Serial.println(dat);
13    digitalWrite(relay,dat);
14 }}
```

این پروژه از ارتباط سریال فرمان میگیره ، در ارتباط سریال (فرض میکنیم جامبر نداریم و اگر هم داریم روی ماژول روی حالت H یا HIGH تنظیم شده) اگر عدد یک فرستاده بشه خروجی وصل میشه و اگر عدد 0 ارسال بشه خروجی قطع میشه.

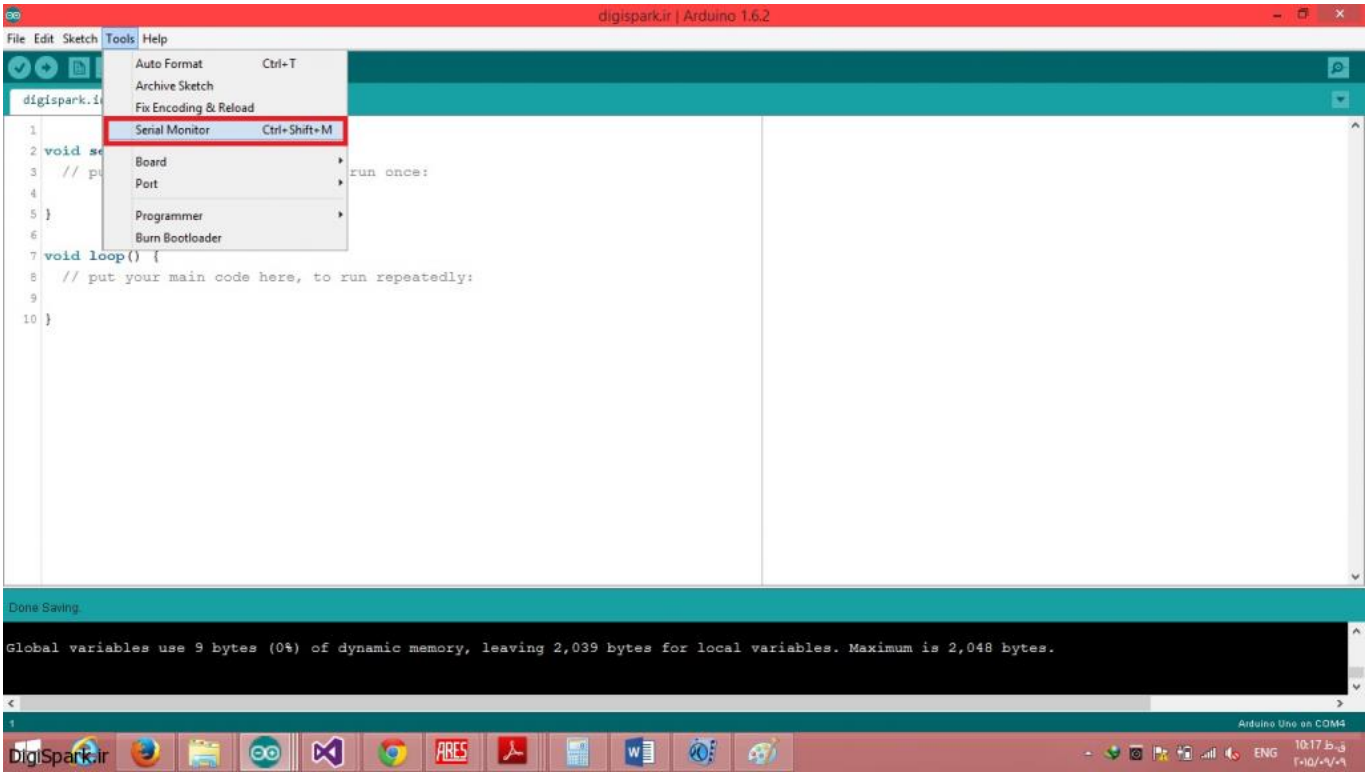
طبق جدول کدهای اسکی با ارسال عدد 0 میکرو عدد 48 رو برگشت میده ، که برای برابر صفر شدن وقرار دادن در دستور digitalWrite اون رو بعد از خوندن منهای مقدار 48 میکنیم.

این قضیه برای مقدار 1 هم صادقه و با ارسال عدد 1 در پورت سریال عدد 49 دیده میشه که او« هم از 48 کم میشه و خروجی برابر با 1 میشه.

آموزش ارتباط سریال آردوینو بخش اول

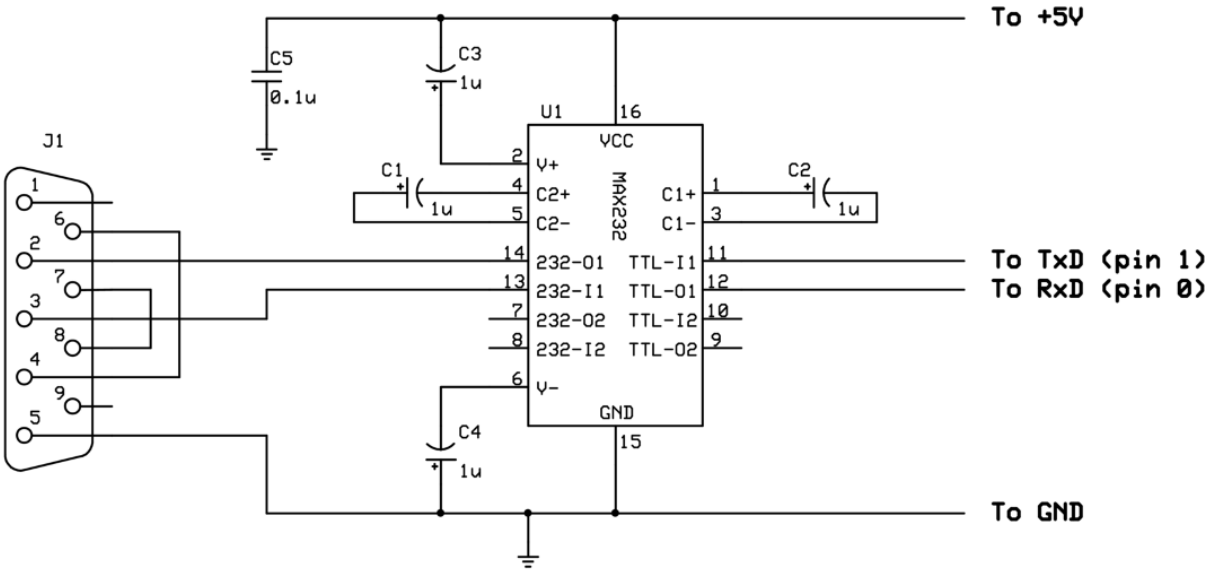
تمام برد های آردوینو دارای حداقل یک عدد پورت سریال می باشد که اختصارا به آن ها USART, UART گفته می‌شود. در آردوینو از پین‌های دیجیتال ۰ و ۱ برای راه اندازی پورت سریال استفاده می‌شود و میتوان به وسیله‌ی درگاه USBبر روی برد ،آردوینو را به کامپیوتر متصل نمود(پین ۰)RX گیرنده اطلاعات) و پین ۱) TX فرستنده اطلاعات)می باشد).هنگامی که از پین های ۰ و ۱ به عنوان پروت سریال استفاده شود ، دیگر نمی‌توان از این پین ها به عنوان ورودی خروجی دیجیتال استفاده نمود.

شما می‌توانید از سریال مانیتوری که در محیط نرم افزار آردوینو در نظر گرفته شده است برای انتقال دیتا میان کامپیوتر و آردوینو استفاده کنید.برای باز کردن سریال مانیتور همانند تصویر زیر عمل کنید.



ارتباط سریال آردوینو

پس از باز کردن این پنجره در قسمت boude rate ، مقداری که در هنگام استفاده از تابع begin وراد کرده اید را در این قسمت وارد کنید(این قسمت به طور کامل در طول آموزش توضیح داده می‌شود) .آردوینو Uno فقط دارای یک پورت ارتباط سریال می‌باشد که از پین های ۰ و ۱ برای این منظور استفاده شده است. آردوینو سری MEGA علاوه بر پورت سریالی که بر روی پین های ۰ و ۱ دارد ، دارای سه پورت سریال دیگر به نام‌های Serial1 , Serial2 , Serial3 می‌باشد.اگر شما قصد برقراری ارتباط آردوینو با کامپیوتر به وسیله‌ی این سه پورت (Serial1 , Serial2 , Serial3) را دارید باید از یک مبدل USB TO Serial استفاده کنید.همچنین می‌توانید برای متصل کردن پورت‌های سریال آردوینو به کابلDB9 ، مطابق مدار زیر از یک آی‌سی MAX232استفاده کنید.



DigiSpark.ir

```
1 Serial.begin(9600 , SERIAL_8N1 );
```

این کد به این معناست که از بادریت ۹۶۰۰ و از مقدار SERIAL_8N1 به عنوان config استفاده کرده ایم.

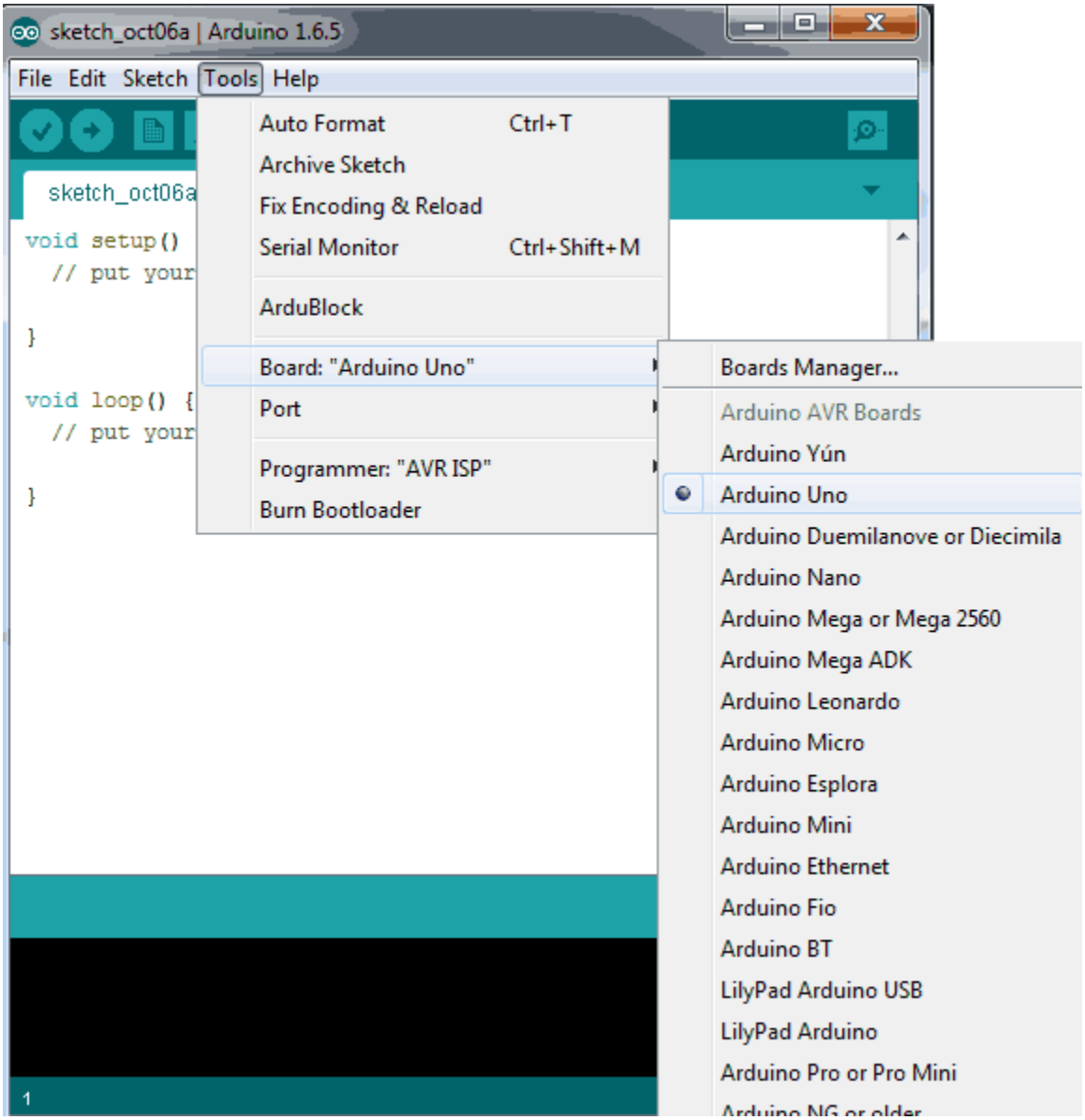
به این علت که کاراکتر O و K هر کدام ۸ بیت و همچنین در کانفیگ از SERIAL_8N1 که به معنای ۸ بیت داده ، بدون بیت توازن و ۱ عدد stop bit می‌باشد ، باید در ۲ بسته اطلاعات را ارسال کنیم . هم اکنون به شرح ارسال کاراکتر Oمی‌پردازیم .کاراکتر O دارای کد اسکی ۷۹ می باشد و در مبنای دو دارای مقدار ۰۱۰۰۱۱۱۱ می‌باشد.

بنابراین نقشه ی ارسال کاراکتر O به صورت زیر می‌باشد.

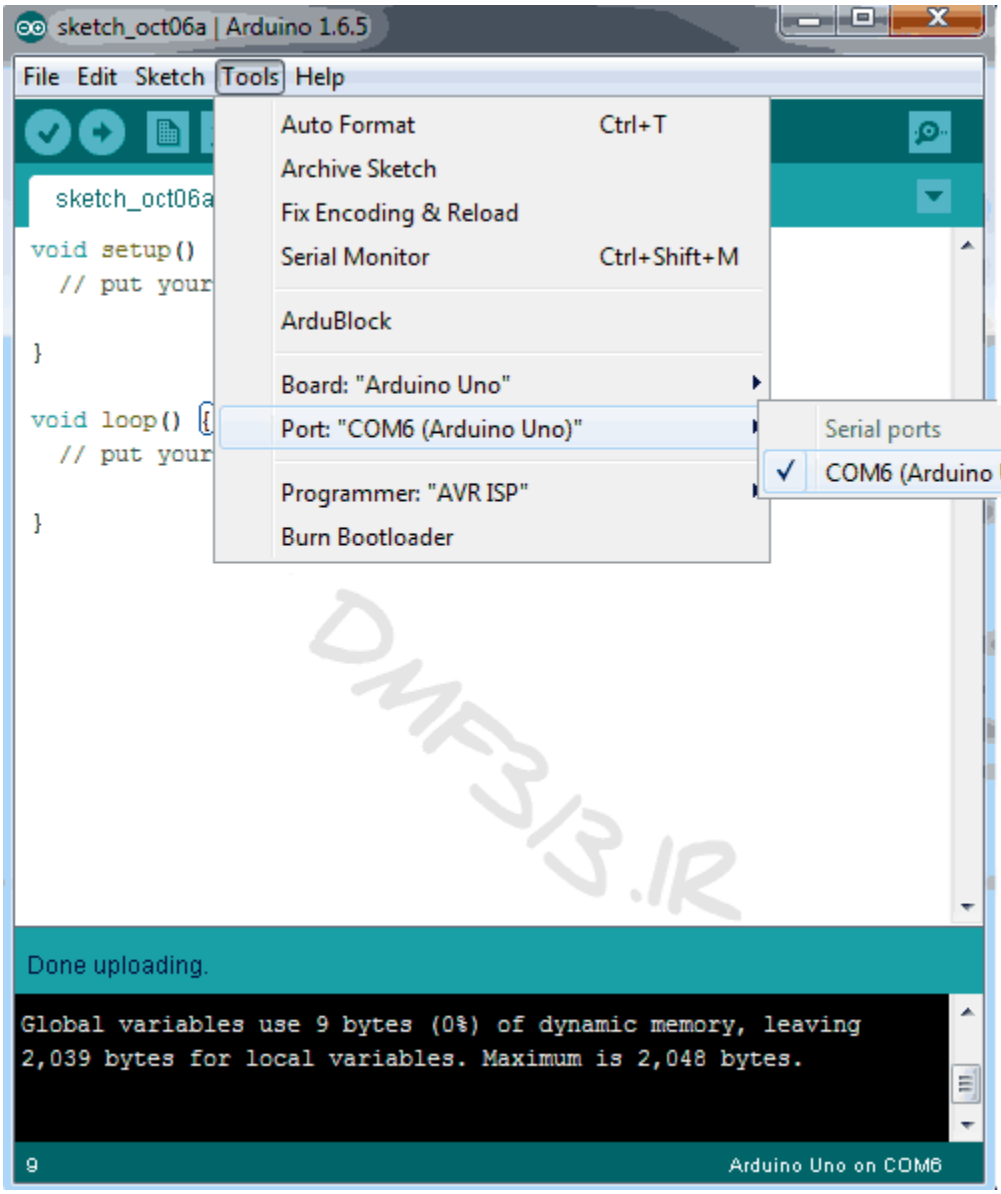
نقشه‌ی ارسال کاراکترO

(ریختن یه برنامه ساده تو آردوینو و تست کردن و نحوه ریختن برنامه داخل آردوینو

خب بعد این که **نرم افزار آردوینو** رو دانلود کردید، اجراش میکنید



۴,۲)بعد باید پورت مورد نظر رو انتخاب کنید(حالا ممکنه این پورت برا کامپیوتر من پورت ۵ باشه و برا شما پورت ۷ بشه و برا یکی دیگه متفاوت باشه، که زیاد مهم نیست(.



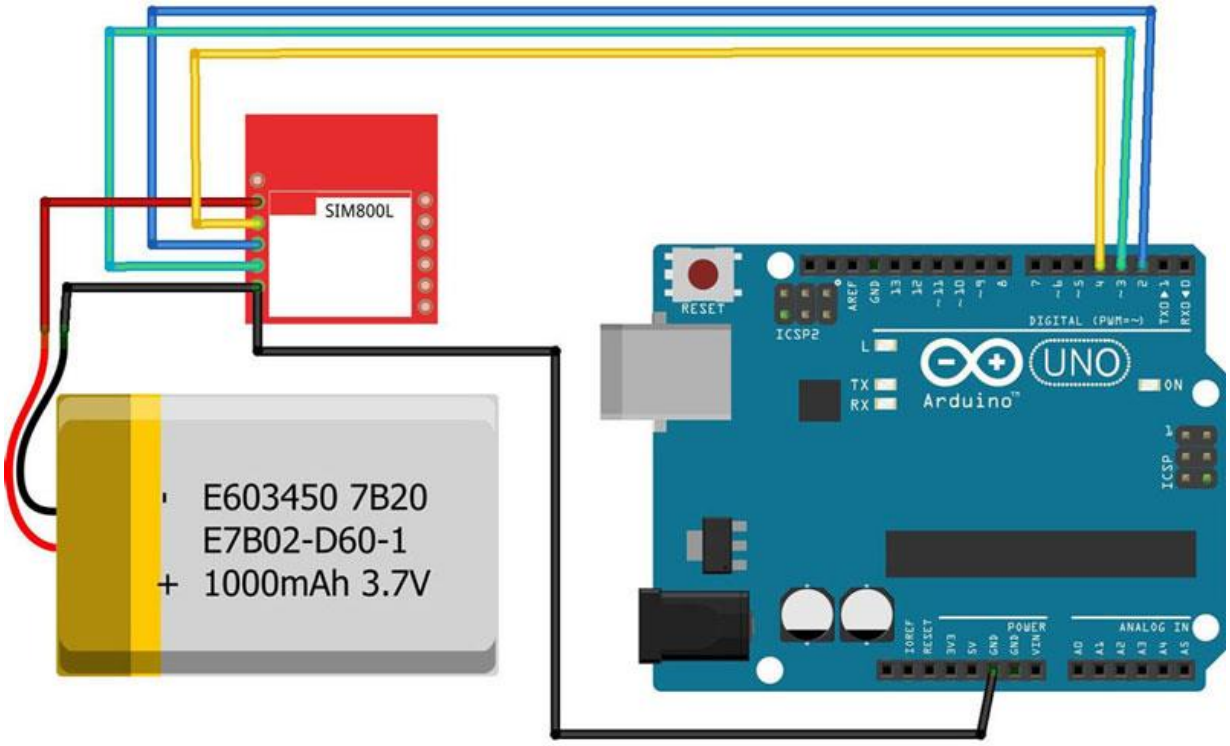
۴,۳ (حالا کدهای زیر رو به نرم افزار اضافه میکنیم(در محیط نرم افزار قرار میدیم.)، فعلا به این که این کد ها چی هستش و توضیحات و... گیر ندید در ادامه و مطلب بعد قشنگ متوجه میشید، تنها چیزی که باید بدونید و باهاش فعلا ور برید کد **delay(100);**هستش، که اون عدد داخل پرانتز رو تغییر بدی زمان روشن و خاموش شدن LED روی برد آردوینو تغییر میکنه، و از این طریق هم یه برنامه نمونه ریختید تو بردتون و از سالم بودنش مطمئن شدید و هم نحوه پروگرام کردن رو یاد گیرفتید و هم ...دیگه چیزی به فکرم نمیرسه ، فقط حواستون باشه که **delay(100);**در کد های زیر ۲ بار نوشته شده که اولی زمان خاموش بودن و دومی زمان روشن بودن LED هستش.

```
void setup()
{
  pinMode(13,OUTPUT);
}
void loop()
{
  digitalWrite(13,0);
  delay(100);
  digitalWrite(13,1);
  delay(100);
}
```

۴,۴)بعد از انجام کارهایی که در بالا گفتم، پروژتون رو در یه جایی ذخیره میکنید(وای بحالتون اگه بفهمم نحوه ذخیره کردن رو بلد نیستید) .حالا روی دکمه ای که به شکل فلش هستش کلیک میکنید(شکل زیر، اون دکمه ای که سفید رنگ هستش) تا برنامه کد هاتون رو بررسی کنه(و اگه مشکلی بود بهتون بگه) و بعد که نرمافزار دید کد هاتون مشکلی نداره میاد و اونو میریزه تو برد و شما اولین پروژتون رو میبینید و کیف میکنید(مثل من هنگام بستن اولین پروژه با آردوینو)

۴,۵)در این مرحله پروژتون رو میبینید ، و اگه کار کرد من رو دعا میکنید و اگه کار نکرد بازم منو دعا میکنید، فقط با این تفاوت که در قسمت نظرات مشکلتون رو هم میگید تا با هم، دو نفری مشکل رو حلش کنیم .

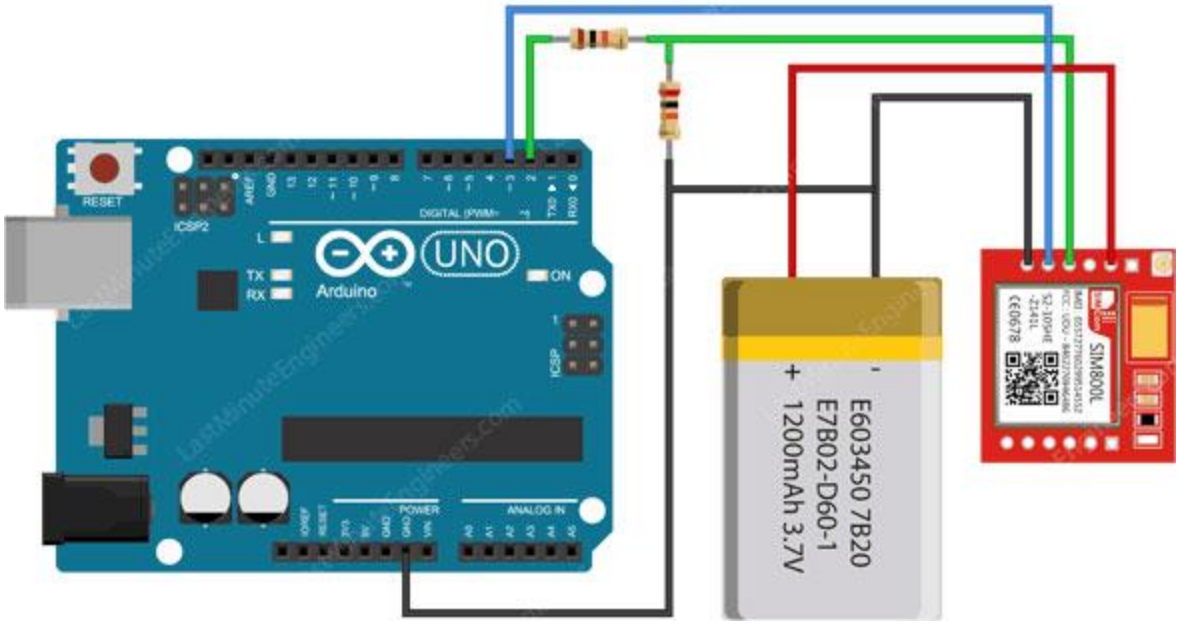
راه اندازی ماژول SIM800L با آردوینو UNO



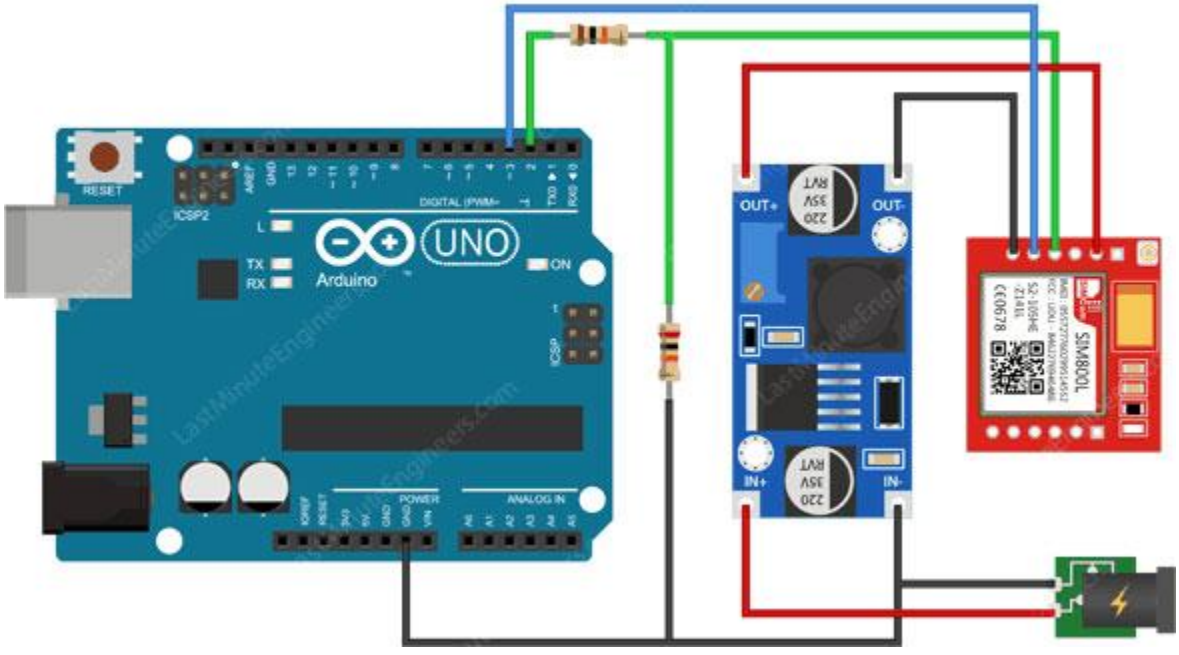
برای راه اندازی ماژول SIM800L با آردوینو ، ابتدا باید آنتن را به ماژول GSM وصل کرد. سپس سیم کارت میکرو فعال شده را در سوکت سیم کارت قرار داد. با اتصال پین TX ماژول به پین 3 آردوینو می توان از طریق مانیتور سریال آردوینو با ماژول ارتباط برقرار کرد .

باید به این نکته توجه شود که نمی توان پین Rx ماژول را به صورت مستقیم به پین دیجیتال آردوینو وصل کرد. زیرا ولتاژ روی پایه های GPIO در آردوینو Unoپنج ولت است، در حالیکه ماژول SIM800L با 3.3 ولت کار می کند و تحمل 5 ولت را ندارد. بنابراین سطح ولتاژی که از آردوینو به ماژول وارد می شود باید به 3.3 کاهش یابد تا به ماژول آسیبی نرسد. چندین راه حل برای کاهش سطح سیگنال وجود دارد که ساده ترین آن استفاده از یک تقسیم مقاومتی ساده است .می توان یک مقاومت 10 کیلو اهم بین پین Rx ماژول SIM800L GSM و پین 2 آردوینو Uno و همچنین یک مقاومت 20 کیلو اهم بین پین RX ماژول و زمین قرار داد.

حال باید پین های منبع تغذیه وصل شوند. همانطور که در مقاله معرفی ماژول SIM800L GSM توضیح داده شد، می توان برای تغذیه ماژول به دو روش عمل کرد که در شکل های زیر شماتیک مدار و نحوه اتصال آن آورده شده است. در شکل اول از باتری Li-Poبا ظرفیت 1200 mAhو در شکل دوم از مبدل کاهنده باک LM2596 برای تغذیه ماژول استفاده شده است.



راه اندازی ماژول SIM800l با آردوینو UNO و استفاده از باتری Li-Po 3.7 ولت برای تغذیه ماژول



راه اندازی ماژول SIM800l با آردوینو UNO و استفاده از مبدل کاهنده باک برای تغذیه ماژول

توجه :هنگامیکه از مبدل کاهنده LM2596 به عنوان منبع تغذیه ماژول استفاده می شود، زمین های مدار باید به یکدیگر متصل باشند.

هنگامیکه تمام اتصالات انجام شد، می توان آردوینو را برای ارسال و دریافت تماس و پیامک برنامه ریزی کرد.

بررسی دستورات ATبرای راه اندازی ماژول SIM800l با آردوینو

برای ارسال دستورات AT و ارتباط با ماژول SIM800L از مانیتور سریال نرم افزار آردوینو استفاده می شود. کد زیر برای ارتباط آردوینو با ماژول SIM800Lبر روی مانیتور سریال ، نوشته شده است. . برای برنامه ریزی آردوینو ، ابتدا باید آردوینو به کامپیوتر وصل شود و سپس کد زیر اجرا و در آردوینو آپلود شود.

لازم به ذکر است که در پنجره مانیتور سریال ، باید گزینه NL&CR انتخاب شود. حال به شرح قسمت های مختلف کد پرداخته می شود.

```
#include < SoftwareSerial.h >
//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 &
#2
void setup()
{
 //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
 Serial.begin(9600);

 //Begin serial communication with Arduino and SIM800L
  mySerial.begin(9600);
  Serial.println("Initializing...");
  delay(1000);
  mySerial.println("AT"); //Once the handshake test is successful, it will
back to OK
  updateSerial();
  mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31
is the best
  updateSerial();
  mySerial.println("AT+CCID"); //Read SIM information to confirm whether the
SIM is plugged
  updateSerial();
 mySerial.println("AT+CREG?"); //Check whether it has registered in the
network
  updateSerial();
}
void loop()
{
  updateSerial();
}
void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to Software
Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial received to
Serial Port
  }
}
```

در ابتدای کد کتابخانه SoftwareSerial.h برای ارتباط سریال میان پین های آردوینو و پورت های Rx /Tx ماژول SIM800Lافزوده شده است .

```
#include < SoftwareSerial.h >
//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 &
#2
```

در تابع setup ، تنظیمات اولیه ارتباط سریال بین آردوینو، محیط آردوینو IDE و ماژول SIM800L با نرخ ارسال 9600 آورده شده است.

```
//Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
 Serial.begin(9600);

 //Begin serial communication with Arduino and SIM800L
  mySerial.begin(9600);
```

پس از انجام تنظیمات اولیه ، با ارسال دستورات AT به ماژول، ارتباط برقرار می شود. همچنین ارسال بصورت خودکار است. در صورتیکه کاراکترهای AT دیده شوند به این معناست که کد به درستی اجرا شده است. سپس کد دستوری برای درخواست ارتباط با ماژول داده می شود که حاوی اطلاعات زیر است:

دستور : ATپایه ترین دستور AT است که با ارسال آن و دریافت OK از سمت ماژول می توان از برقراری ارتباط بین ماژول و آردوینو اطمینان یافت. سپس می توان دستوراتی را مانند دستورات زیر، برای ماژول ارسال کرد و اطلاعاتی را در مورد آن بدست آورد:

دستور : AT+CSQ شدت سیگنال را بررسی می کند. عدد اول، شدت سیگنال به دسی بل را بیان می کند که باید مقدارش از 5 بیشتر باشد. هر چه این مقدار بیشتر باشد، بهتر است. البته این مقدار به نوع آنتن مورد استفاده و موقعیت مکانی ماژول بستگی دارد .

دستور : AT+CCIDبا ارسال این دستور و دریافت OK می توان از فعال بودن سیم کارت اطمینان پیدا کرد. به علاوه با این دستور شماره سیم کارت قابل دریافت است .

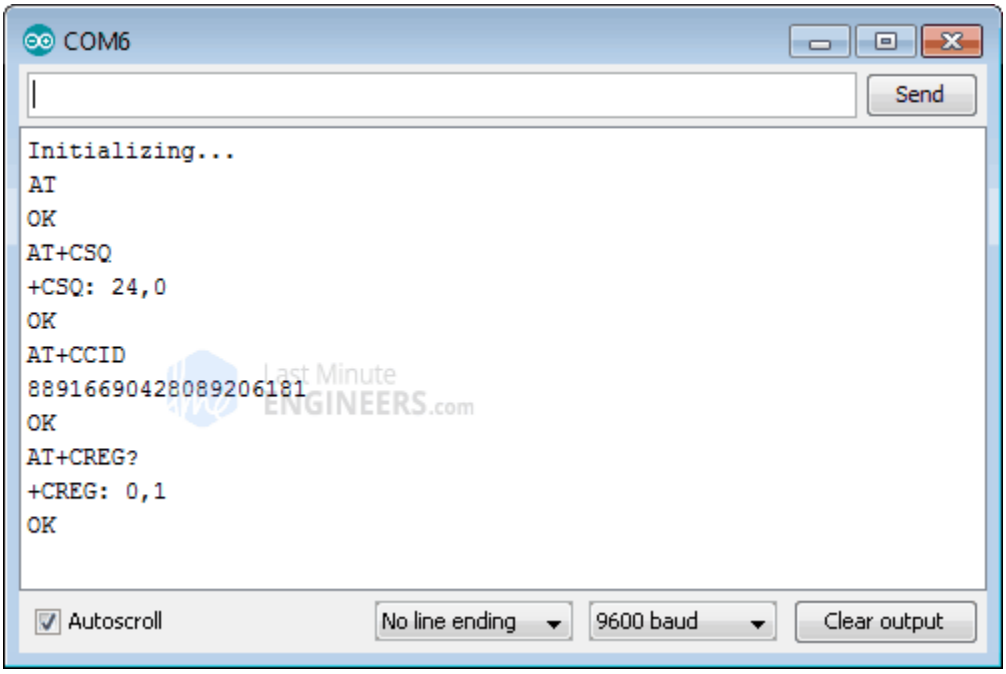
دستور : AT+CREG? بررسی می کند که آیا سیم کارت در شبکه ثبت شده یا خیر .عدد دوم که در پاسخ به این دستور دریافت می شود، نوع شبکه ای که سیم کارت در آن ثبت شده است را نشان می دهد. این عدد می تواند 1 یا 5 باشد. عدد 1 نشان دهنده شبکه خانگی و عدد 5 نشان دهنده شبکه رومینگ است. هر عدد دیگری به جز 1 و 5 ، نشان می دهد که سیم کارت در هیچ شبکه ای ثبت نشده است.

```
  delay(1000);
  mySerial.println("AT"); //Once the handshake test is successful, it will
back to OK
  updateSerial();
  mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-31 , 31
is the best
  updateSerial();
  mySerial.println("AT+CCID"); //Read SIM information to confirm whether the
SIM is plugged
  updateSerial();
 mySerial.println("AT+CREG?"); //Check whether it has registered in the
network
  updateSerial();
```

در تابع loop ، تابع updateSerial() قرار دارد که دائماً منتظر ورودی های مانیتور سریال است تا آن را از طریق پین 2) پین Rx ماژول (به ماژول SIM800Lارسال کند. همچنین دائماً پین) D3 پین Tx ماژول (را چک می کند تا اگر ماژول پاسخی به ورودی داشته باشد، آن را در مانیتور سریال نشان دهد.

```
void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to Software
Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial received to
Serial Port
  }
}
```

با اجرای کد، شکل زیر بر روی مانیتور سریال دیده می شود.



دستورات پایه AT برای ارتباط با ماژولSIM800

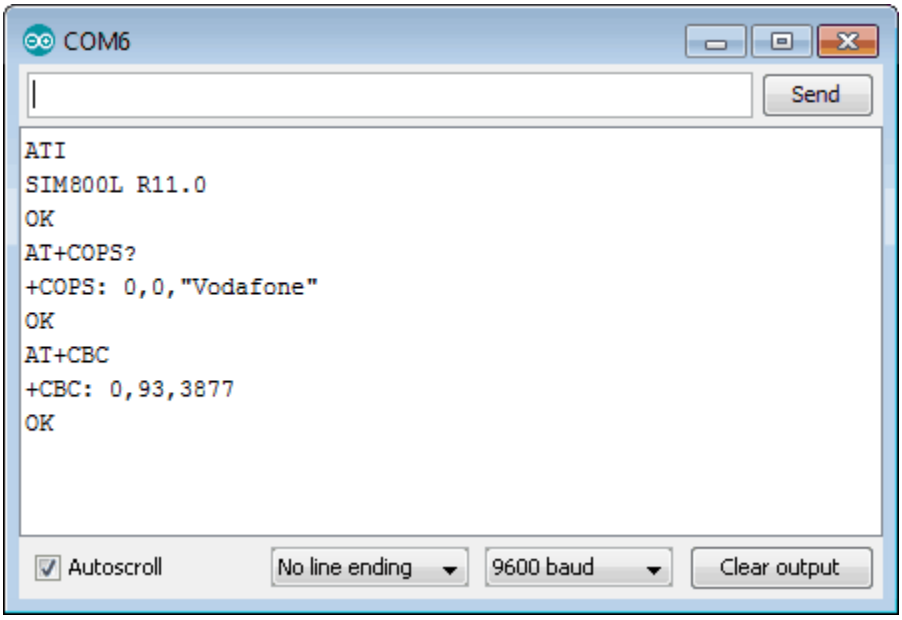
علاوه بر دستوراتی که بیان شد، دستورات دیگری نیز می توان برای دریافت اطلاعات بیشتری در مورد اتصال شبکه و وضعیت باتری از طریق مانیتور سریال به ماژول ارسال کرد. در ادامه چند دستور آورده شده است.

دستور : ATI نام و نسخه ماژول را می دهد.

دستور : AT+COPS? اتصال به شبکه را بررسی می کند. .

دستور : AT+COPS=? لیست اُپراتورهای موجود در شبکه را نشان می دهد.

دستور : AT+CBC وضعیت باتری Li-Po را نشان می دهد. عدد دوم میزان شارژ باتری (که در اینجا 93%) و عدد سوم ولتاژ واقعی به میلی ولت (در اینجا 3.877ولت) است.



دستورات AT برای اتصال شبکه در ماژولSIM800L GSM

ارسال پیام با ماژولSIM800L GSM

پس از آماده سازی آردوینو وSIM800 ، می توان ماژول آردوینو را برای ارسال پیام به شماره موبایل مورد نظر کد نویسی کرد. قبل از کد نویسی باید شماره موبایل را وارد کرد. در رشته ZZxxxxxxxxxx ، کد کشور باید جایگزین ZZ و شماره 10 رقمی به جای xxxxxxxxxx قرار گیرد.

```
#include < SoftwareSerial.h >

//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 &
#2

void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and SIM800L
  mySerial.begin(9600);

  Serial.println("Initializing...");
  delay(1000);

  mySerial.println("AT"); //Once the handshake test is successful, it will
back to OK
```

```
  updateSerial();

  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with country code
and xxxxxxxxxx with phone number to sms
  updateSerial();
  mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text
content
  updateSerial();
  mySerial.write(26);
}

void loop()
{
}

void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
        mySerial.write(Serial.read());//Forward what Serial received to
Software Serial Port
  }
  while(mySerial.available())
  {
        Serial.write(mySerial.read());//Forward what Software Serial received
to Serial Port
  }
}
```

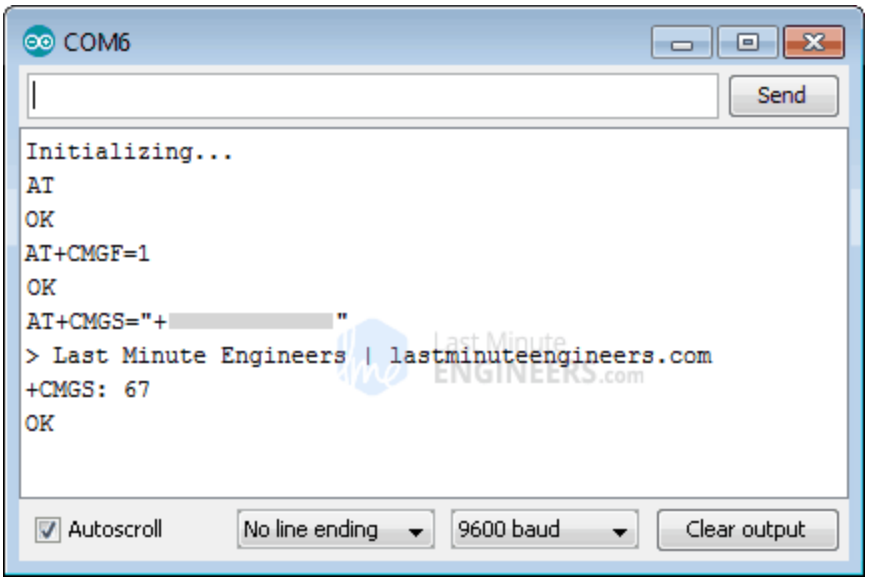
این کد تقریبا شبیه کد قسمت قبل است. در اینجا به محض اینکه اتصال برقرار شود، دستورات زیر ارسال می شوند:

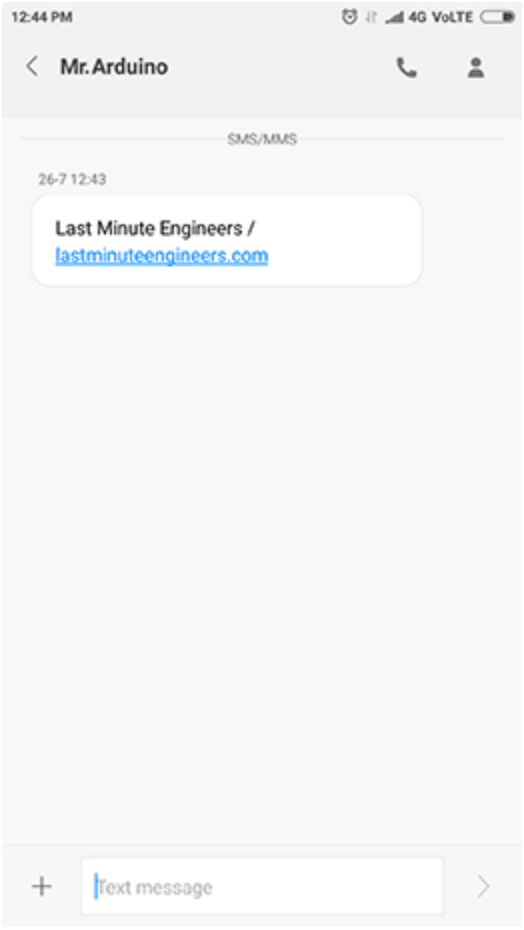
دستور : AT+CMGF=1 قالب پیام را از نوع "متن" قرار می دهد. قالب پیشفرض) PDU پروتکل انتقال داده (است.

دستور : AT+CMGS=+ZZxxxxxxxxxx پیام را به شماره مشخص شده ارسال می کند .پیامی که به دنبال آن کلید 'Ctrl+z'وارد شود به عنوان یک پیام متنی در نظر گرفته می شود. کلید 'Ctrl+z' بیست و ششمین کاراکتر جدول ASCIIاست .بنابراین، لازم است که هنگام ارسال پیام متنی 26 دسیمال یا 1 Aهگز ارسال شود .

```
println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with country code
and xxxxxxxxxx with phone number to sms
  updateSerial();
  mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text
content
  updateSerial();
  mySerial.write(26);
```

در صورتی که فقط یک بار پیام ارسال شود، تابع loop خالی می ماند. در صورتی که لازم باشد که یک بار دیگر نیز پیام ارسال شود، تنها کافیست که کلید RESET در آردوینو فشار داده شود. تصویر زیر پیام ارسالی به ماژول SIM800lرا نشان می دهد.





پیام ارسالی به ماژولSIM800

دریافت پیام باSIM800L

برای دریافت پیام می توان آردوینو را بصورت زیر کد نویسی کرد. کد زیر برای زمانی که لازم باشد هنگام دریافت پیام، عمل خاصی انجام گیرد نیز کاربرد دارد. به عنوان مثال، هنگامی که آردوینو پیامی را دریافت کند، رله خاموش یا روشن شود.

```
#include < SoftwareSerial.h >

//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 &
#2

void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and SIM800L
  mySerial.begin(9600);

  Serial.println("Initializing...");
  delay(1000);

  mySerial.println("AT"); //Once the handshake test is successful, it will
back to OK
  updateSerial();

  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly arrived SMS
messages should be handled
  updateSerial();
}

void loop()
{
  updateSerial();
}

void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
        mySerial.write(Serial.read());//Forward what Serial received to
Software Serial Port
  }
  while(mySerial.available())
  {
        Serial.write(mySerial.read());//Forward what Software Serial received
to Serial Port
  }
}
```

این کد نیز مشابه کدهای قبل است و تنها در چندین دستور AT متفاوت است. در این کد ، هنگامی که اتصال برقرار می شود، دستورات AT زیر اجرا می شوند :

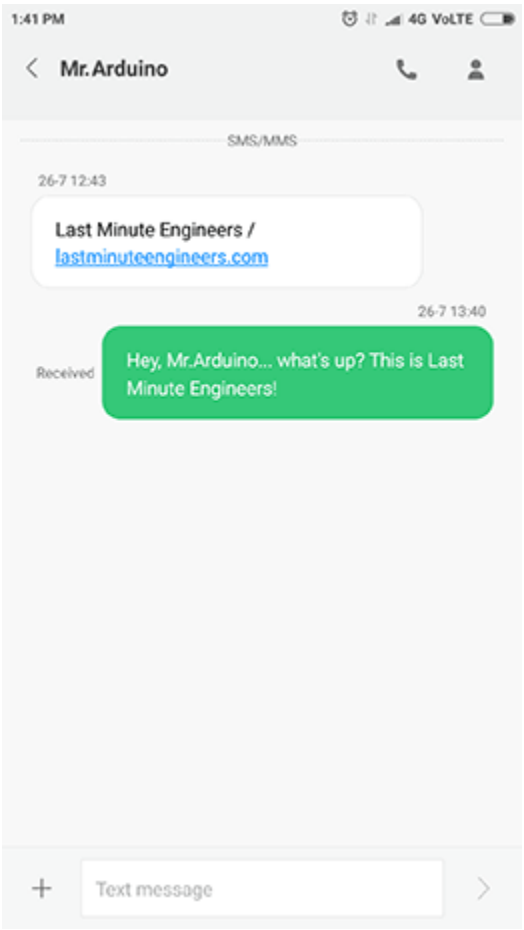
دستور : AT+CMGF=1 قالب پیام را از نوع ‘متنی’ قرار می دهد. قالب پیشفرض PDU است .

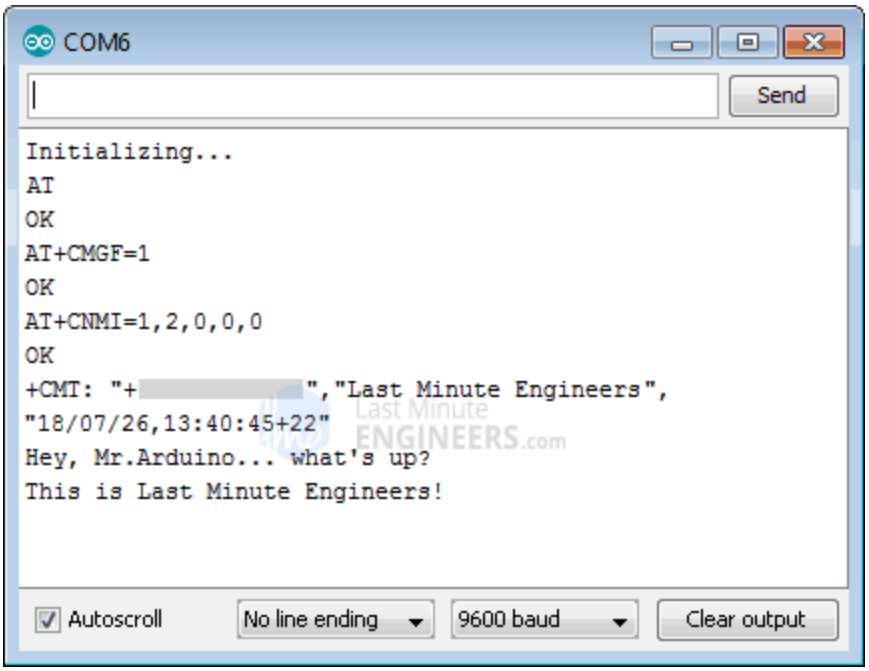
دستور : AT+CNMI=1,2,0,0,0چگونگی دستیابی کامپیوتر به پیام جدید را تعیین می کند. با استفاده از این دستور می توان تعیین کرد که ماژول پیام های ورودی جدید را مستقیما به کامپیوتر ارسال کند یا آنها را در حافظه خود ذخیره کند و سپس مکان ذخیره سازی پیام را به کامپیوتر اطلاع دهد.

پاسخ ماژول با کد دستوری +CMT شروع می شود: تمام مقادیر در آن با استفاده از کاما ‘,’ از یکدیگر جدا شده اند. اولین مقدار شماره موبایل، مقدار دوم نام شخصی است که پیام را فرستاده، مقدار سوم تاریخ و زمان دریافت پیام و مقدار چهارم متن پیام است .

```
println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly arrived SMS
messages should be handled
  updateSerial();
```

این بار تابع loop (حلقه) خالی نیست تا هر زمان که پیامی به ماژول ارسال شود، بتوان آن را خواند. به محض ارسال پیام به ماژولSIM800 ، می توان آن را بر روی مانیتور سریال مشاهده کرد.





دریافت پیام توسط ماژول SIM800 و نمایش خروجی بر روی مانیتور سریال

افزایش اندازه بافر SoftwareSerial آردوینو برای دریافت پیام

در صورتی که پیام طولانی باشد، ممکن است که برخی از کاراکترها حذف شوند. این مساله به دلیل خطای کد نویسی نمی باشد بلکه بافر دریافتی SoftwareSerial پر شده و به همین دلیل، برخی از کاراکترها حذف می شوند .

ساده ترین راه حل افزایش اندازه بافر SoftwareSerial است. اندازه پیشفرض بافر 64 بایت است که می توان آن را تا 256 بایت (این مقدار بسته به کاربرد میتواند کمتر انتخاب شود) افزایش داد. برای این منظور، در کامپیوتر در صورتی که دارای سیستم عامل ویندوز باشد، به آدرس زیر رفته:

C:\Program Files (x86) -> Arduino -> hardware -> Arduino -> avr -> libraries -> SoftwareSerial

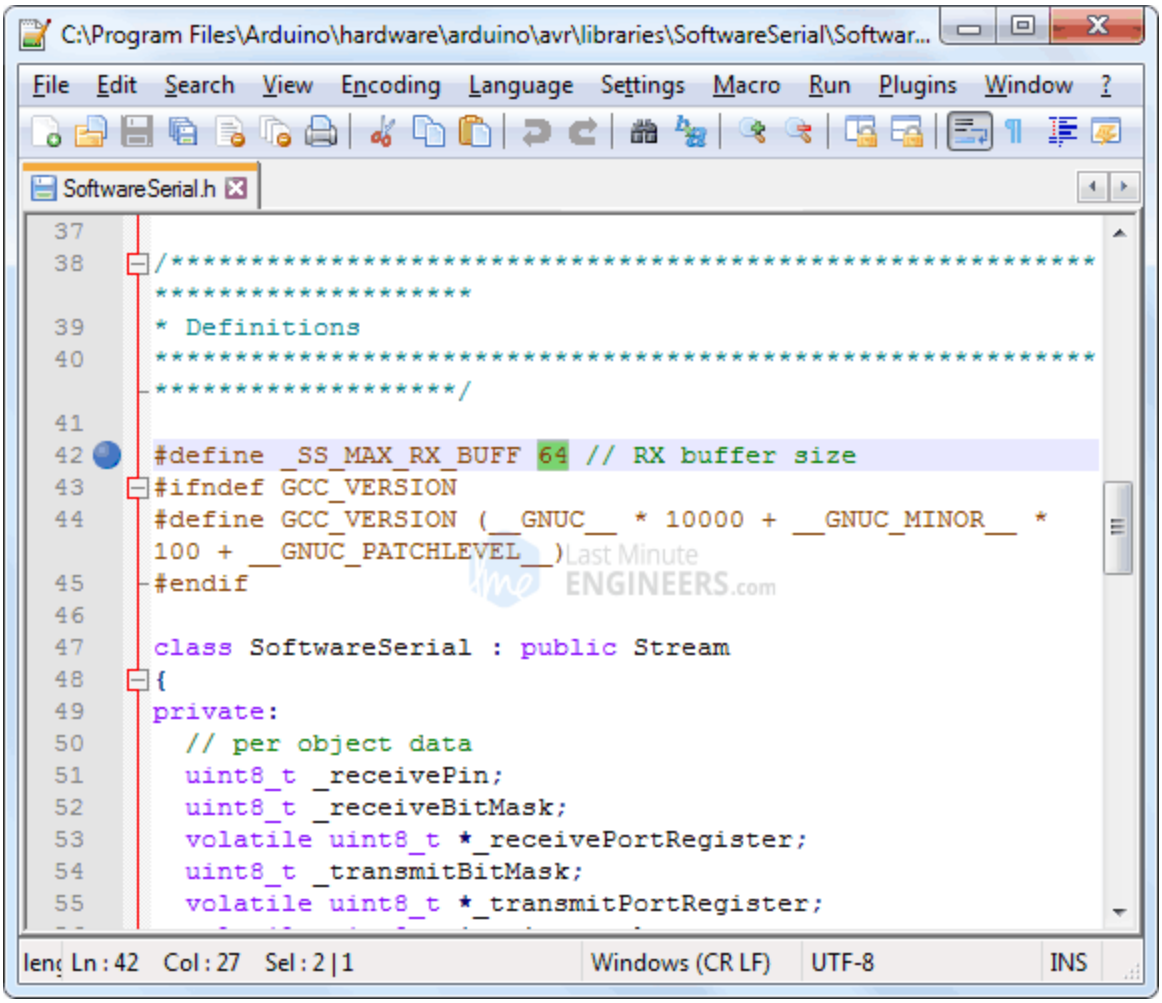
)در صورتی که از نسخه های جدیدتر آردوینو IDE استفاده می شود به پوشه srcرفته)، سپس کتابخانه SoftwareSerial.hرا باز کرده و دستور مربوط به اندازه بافر مانند زیر تغییر کند:

```
// RX buffer size
#define _SS_MAX_RX_BUFF 64
```

باید به صورت زیر تغییر کند:

```
// RX buffer size
#define _SS_MAX_RX_BUFF 256
```

پس از ذخیره کردن فایل، باید کد دوباره اجرا شود.



افزایش اندازه بافر SoftwareSerial آردوینو

برقراری تماس با ماژول SIM800L و آردوینو

این دستورات برای برقراری تماس کمکی (SoS) در موقعیت های خطرناک و اضطراری بطور مثال، زمانیکه آتش سوزی یا سرقت در منزل رخ بدهد، کاربرد دارد .

قبل از اجرای برنامه، ابتدا باید شماره موبایل مورد نظر وارد شود به این صورت که در رشته ZZxxxxxxxxxx کد کشور با ZZ و شماره موبایل 10 رقمی مورد نظر با xxxxxxxxxx جایگزین شوند.

```
#include < SoftwareSerial.h >
 //Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 &
#2
void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);
  //Begin serial communication with Arduino and SIM800L
  mySerial.begin(9600);
```

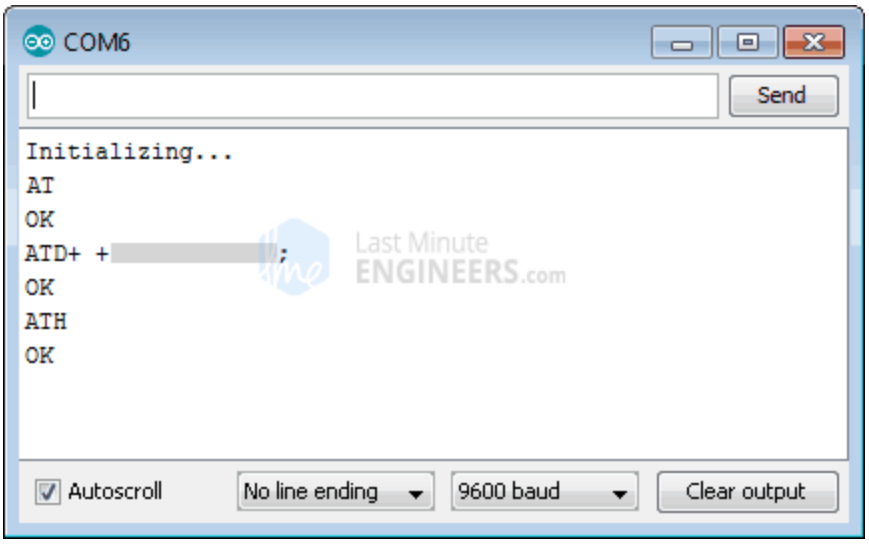
```
   Serial.println("Initializing...");
  delay(1000);
   mySerial.println("AT"); //Once the handshake test is successful, i t will
back to OK
  updateSerial();
  mySerial.println("ATD+ +ZZxxxxxxxxxx;"); //  change ZZ with country code
and xxxxxxxxxxx with phone number to dial
  updateSerial();
  delay(20000); // wait for 20 seconds...
  mySerial.println("ATH"); //hang up
  updateSerial();
}
 void loop()
{
}
 void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
        mySerial.write(Serial.read());//Forward what Serial received to
Software Serial Port
  }
  while(mySerial.available())
  {
        Serial.write(mySerial.read());//Forward what Software Serial received
to Serial Port
  }
}
```

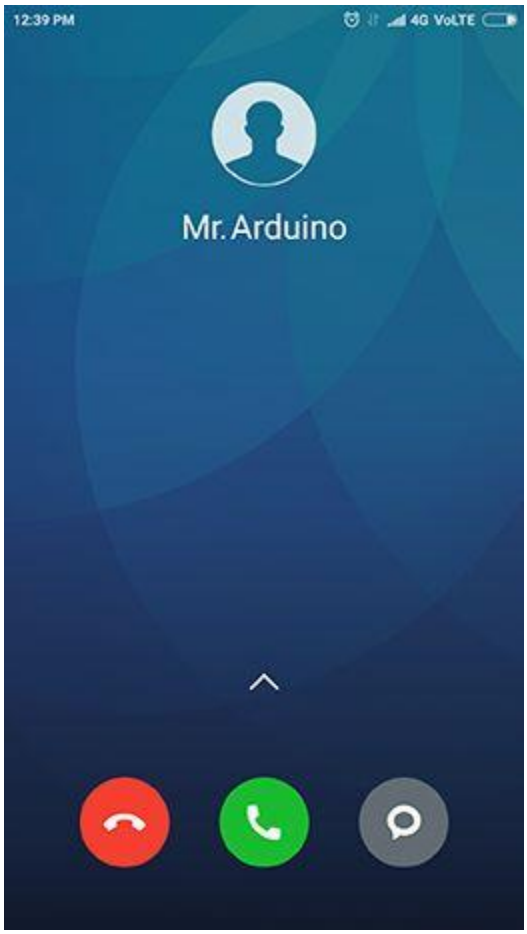
برای برقراری تماس باید دستورات AT زیر را به کار برد :

دستور : ATD+ +ZZxxxxxxxxxx; با شماره وارد شده تماس می گیرد. کاراکتر (;) نشان دهنده پایان دستور است و باید در انتهای تمام خط کدها قرار بگیرد .

دستور : ATH تماس را قطع می کند.

```
println("ATD+ +ZZxxxxxxxxxx;"); //  change ZZ with country code and
xxxxxxxxxxx with phone number to dial
  updateSerial();
  delay(20000); // wait for 20 seconds...
  mySerial.println("ATH"); //hang up
  updateSerial();
```





برقراری تماس با ماژول SIM800 و نمایش دستورات AT در مانیتور سریال