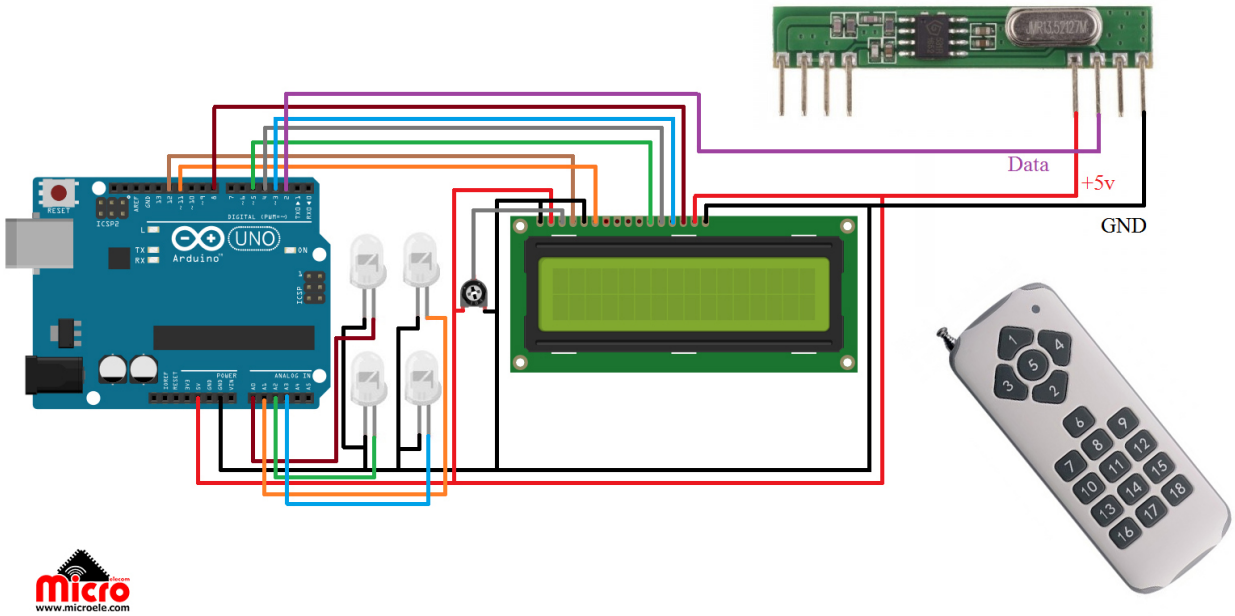
پروژه رایگان گیرنده 18 کانال ریموت کد لرن با آردوینو



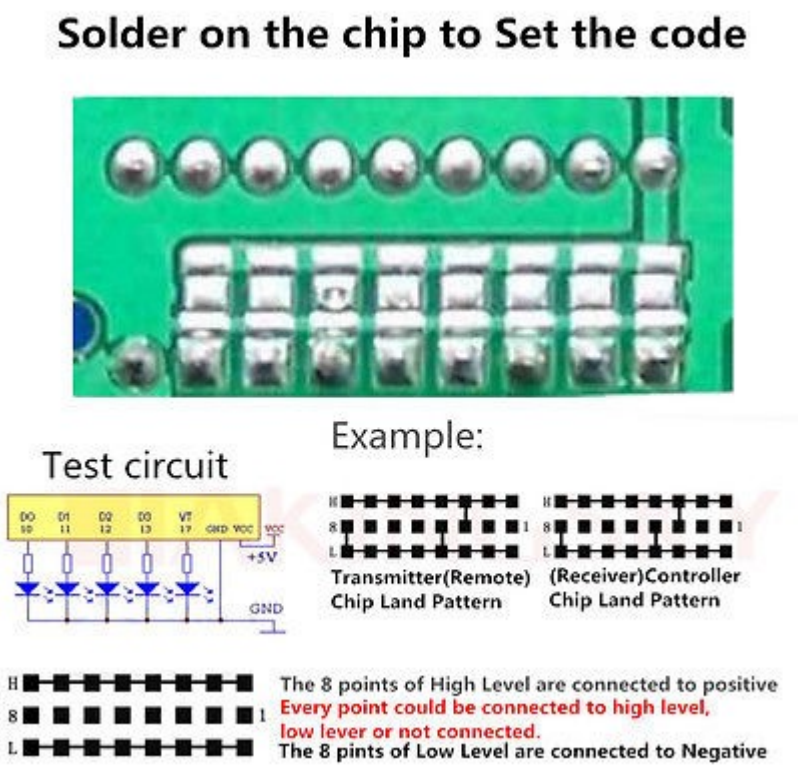
https://blog.microele.com

تاریخ انتشار۷ تیر, ۱۳۹۹ توسط سید حسین سلطانی

در این ریموت ها از بستر امواج رادیویی برای ارسال اطلاعات استفاده شده است. این ریموت ها در انواع مختلف و فرکانس های 315 یا 433 مگاهرتز موجود می باشند. بطور کلی، امنیت یکی از مهم ترین مسائل در ارسال و دریافت اطلاعات از طریق امواج رادیویی می باشد. از این رو، ریموت های رادیویی در انواع فیکس کد (کد ثابت)، کد لرن و نوع هاپینگ موجود می باشند و هر کدام دارای امنیت متفاوتی هستند.

ریموت های فیکس کد:

این نمونه از ریموت ها، دارای 8 پایه برای تنظیم آدرس ریموت می باشند. برای تنظیم آدرس این ریموت ها باید پایه های IC را به '1' یا '0' وصل کرد. و همین روند را دقیقا در سمت گیرنده پیاده سازی میکردیم تا گیرنده و فرستنده با هم بتوانند ارتباط برقرار کنند.



چرا به این نمونه ریموت ها کد فیکس گویند؟

همانطور که از نام آن مشخص است، این ریموت ها دارای یک کد ثابت برای هر ریموت است. در گیرنده نیز، دقیقا باید همین کد به عنوان آدرس آن تنظیم شده باشد. برای اضافه کردن یک ریموت دیگر به گیرنده باید کد ریموت جدید نیز مانند ریموت های دیگر باشد. که این مورد می تواند امنیت را دچار چالش کند. چرا که اگر کسی کد ریموت شما را پیدا کند یا در صورت مفقودی ریموت، می تواند به راحتی یک ریموت دیگر مانند آن، کد دهی کرده و دستگاه شما را کنترل کند.

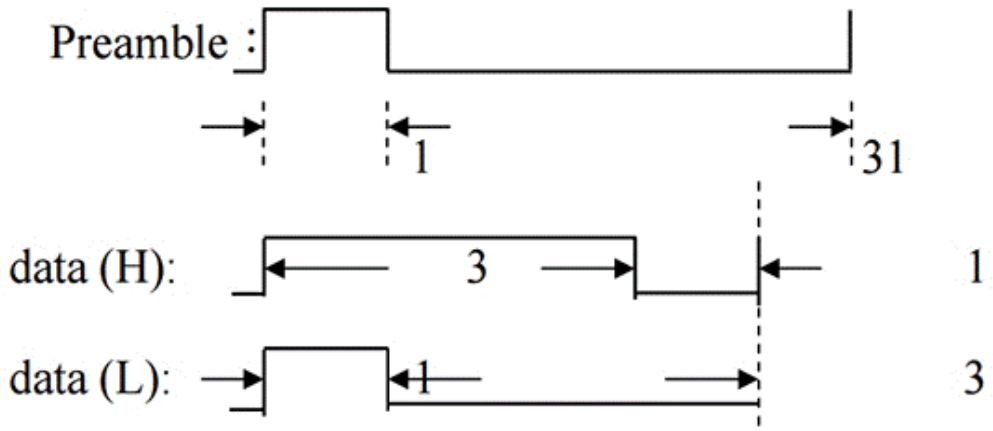
ریموت کد لرن:

نوعی دیگر از ریموت ها که دارای امنیت بیشتری نسبت به ریموت های کد فیکس می باشند، ریموت های کد لرن هستند. به این صورت که در دل این ریموت ها تراشه ای (HS1527 یا EV1527) وجود دارد که از قبل توسط کارخانه سازنده یک کد اختصاصی 20 بیتی تصادفی به آن اختصاص داده شده است و با سایر ریموت ها دارای کد متفاوتی می باشد. پس تا حد خیلی زیادی به لطف آدرس های متمایز با ریموت های دیگر امنیت آن تامین شده است.

پروتکل استفاده شده در ریموت های کد لرن چه از تراشه EV1527 یا HS1527 استفاده شده باشد، یک شکل بوده و از انکدر OTP استفاده می کنند. در این روش برای هر کلید موجود بر روی ریموت 24 بیت داده ارسال خواهد شد.

Output data frame:

Preamble	C0 ~ C19 (1 million codes)	D0	D1	D2	D3



همانطور که در تصویر بالا مشخص است، 20 بیت (C0-C19) کد اختصاصی هر ریموت است که بصورت اتفاقی برای هر ریموت از طرف کارخانه سازنده در نظر گرفته می شود. در این حالت دارای 1048576 آدرس متفاوت می باشد. در اطلاعات ارسالی از سمت ریموت، تعداد 20 بیت مربوط به کد آدرس ریموت و 4 بیت آخر (D0-D3) مربوط به کد هر کلید موجود روی ریموت می باشد. در ابتدای هر ارسال، یک بیت Preamble (همزمانی) ارسال میگردد. با دریافت این بیت در گیرنده، نسبت به دریافت 24 بیت بعدی اقدام خواهیم کرد.

برای رمزگشایی کد های ارسالی از ریموت به بهترین نحو ممکن، باید با دریافت لبه بالارونده، یک شمارنده شروع به شمارش کرده و با فرارسیدن لبه پایین رونده مقدار این شمارنده را خوانده و مقدار شمارنده را صفر کند. باید دقت شود که حتما در مرحله اول، بیت همزمانی را دریافت کرده و نهایتا 24 بیت بعدی را، این کار باعث می شود که

اطمینان بیشتری در صحت دریافت کد های ریموت داشته باشیم.

مطابق با دیتا شیت تراشه مورد استفاده ریموت، سرعت مورد نیاز برای شمارش تایمر کانتر را می توان فهمید تا بتوانیم به درستی طول پالس ها را اندازه گیری کرده و اطلاعات ارسالی را دریافت نماییم.

EV1527 Rosc& data cycle:(Compatible:EV1527,RT1527,FP1527)

Rosc& TD:	12V	11V	10V	9V	8V	7V	6V	5V	4V
300K	1.34ms	1.37ms	1.41ms	1.45ms	1.5ms	1.57ms	1.66ms	1.79ms	1.99ms
330K	1.48ms	1.51ms	1.55ms	1.59ms	1.64ms	1.71ms	1.81ms	1.94ms	2.19ms
360K	1.71ms	1.75ms	1.79ms	1.84ms	1.91ms	1.99ms	2.11ms	2.27ms	2.53ms
390K	1.82ms	1.86ms	1.91ms	1.96ms	2.03ms	2.11ms	2.24ms	2.401ms	2.69ms
430K	2.035ms	2.085ms	2.135ms	2.195ms	2.265ms	2.365ms	2.505ms	2.705ms	3.01ms

ریموت های هاپینگ:

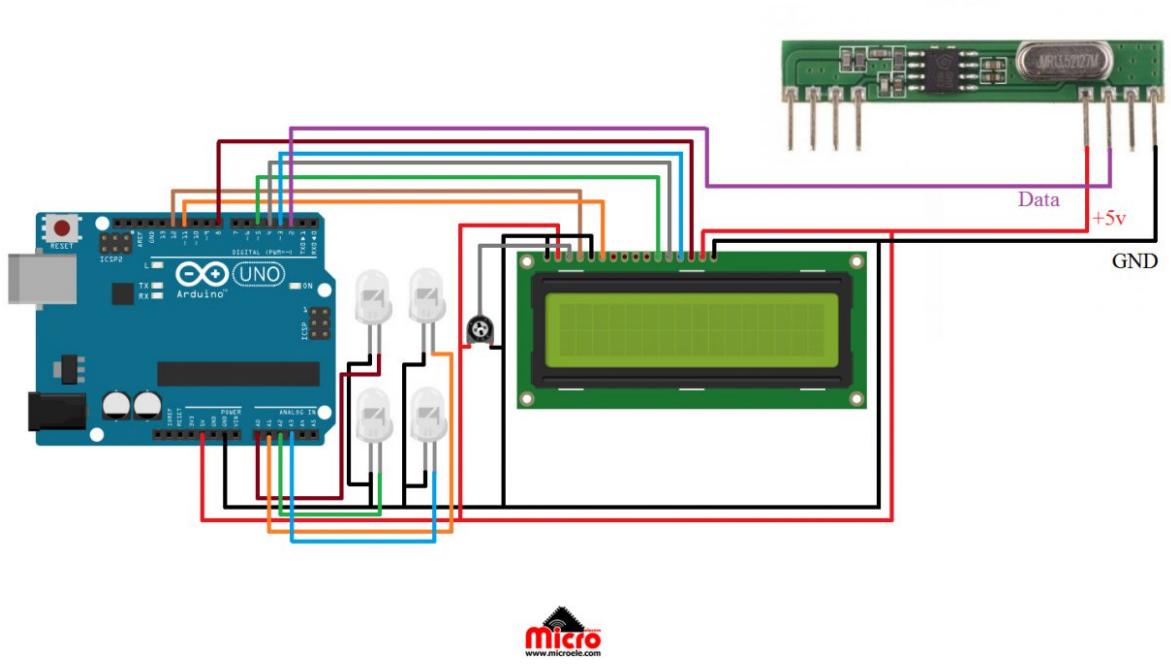
این نمونه از ریموت ها از الگوریتم پیشرفته و پیچیده تری برای رمزگذاری و ارسال اطلاعات استفاده می کنند که به شدت دارای امنیت بالاتری می باشند.

دیکد کردن ریموت کد لرن توسط آردوینو:

وسایل مورد نیاز:

- آردوینو UNO
- LCD 2*16
- ریموت کنترل 18 کاناله
- (RXB18 (433MHz
- LED
- برد بورد
- سیم مخصوص برد بورد

شماتیک پروژه:



مراحل کد نویسی:

برای کد نویسی این پروژه از محیط برنامه نویسی Arduino IDE استفاده کرده ایم. برای خواندن کد های ریموت، از کتابخانه RCSwitch استفاده شده است. این کتابخانه از تراشه های زیر پشتیبانی می کند:

- SC5262 / SC5272
- HX2262 / HX2272
- PT2262 / PT2272
- EV1527 / RT1527 / FP1527 / HS1527

- HT6P20X

برای دانلود کتابخانه مربوطه به این لینک مراجعه کنید.

معرفی کتابخانه ها:

```
#include <RCSwitch.h>
#include <LiquidCrystal.h>
```

برای خواندن کد ارسالی ریموت، از دستورات زیر استفاده می کنیم:

```
if (mySwitch.available() > 0)
{
    Serial.println("دریافت شد");
    output(mySwitch.getReceivedValue(), mySwitch.getReceivedBitlength());
    Serial.println(b);
    lcd.setCursor(0, 1);
    lcd.print("Recieve");
}
```

با استفاده از دستور زیر می توان کد ارسال شده و تعداد بیت های آن را متوجه شد:

```
output(mySwitch.getReceivedValue(), mySwitch.getReceivedBitlength());
```

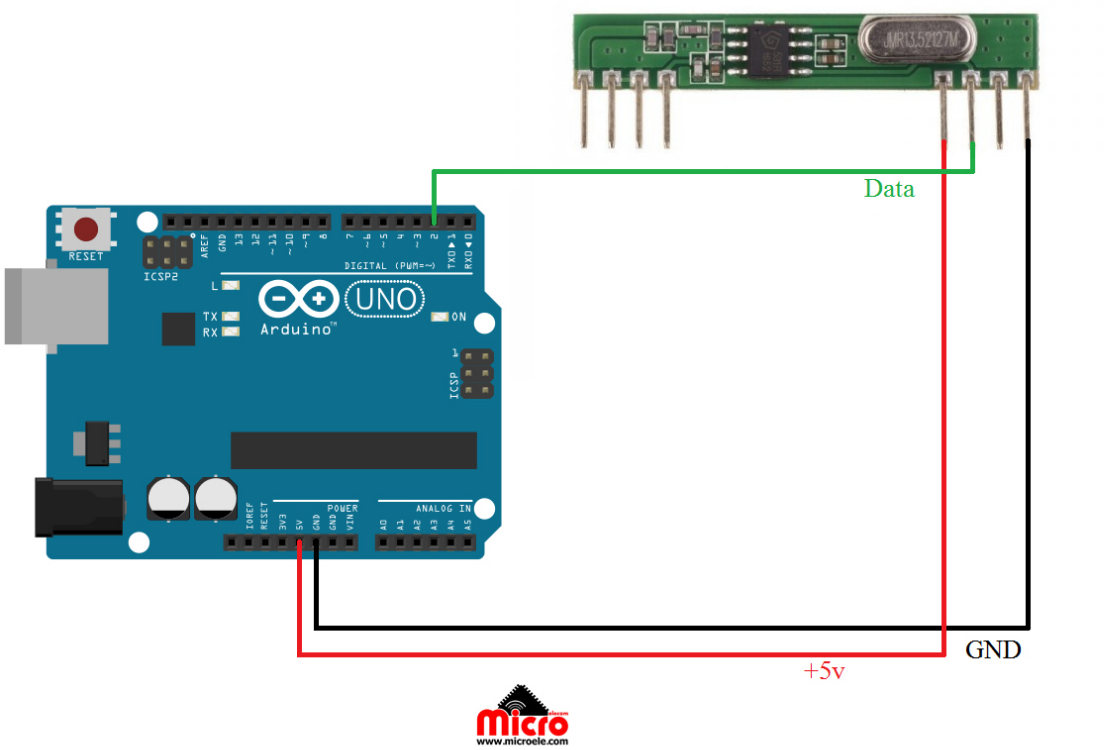
برای اینکه کد های ریموت را بصورت باینری 24 بیتی داشته باشیم، از توابع زیر استفاده می کنیم:

```
void output(unsigned long decimal, unsigned int length)
{
```

```
  b = dec2binWzerofill(decimal, length);
  b_length = b.length() + 1;
  b.toCharArray(input_array, b_length);
}
String dec2binWzerofill(unsigned long Dec, unsigned int bitLength)
{
  static char bin[64];
  unsigned int i = 0;
  while (Dec > 0)
  {
    bin[32 + i++] = ((Dec & 1) > 0) ? '1' : '0';
    Dec = Dec >> 1;
  }
  for (unsigned int j = 0; j < bitLength; j++) {
    if (j >= bitLength - i)
    {
      bin[j] = bin[ 31 + i - (j - (bitLength - i)) ];
    }
    else
    {
      bin[j] = '0';
    }
  }
  bin[bitLength] = '\0';
  return bin;
}
```

با استفاده از کد زیر که در مثال های کتابخانه نیز موجود می باشد، می توانیم علاوه بر نمایش کد های ارسالی ریموت، طول بیت ها را نیز نمایش دهیم.

شماتیک:



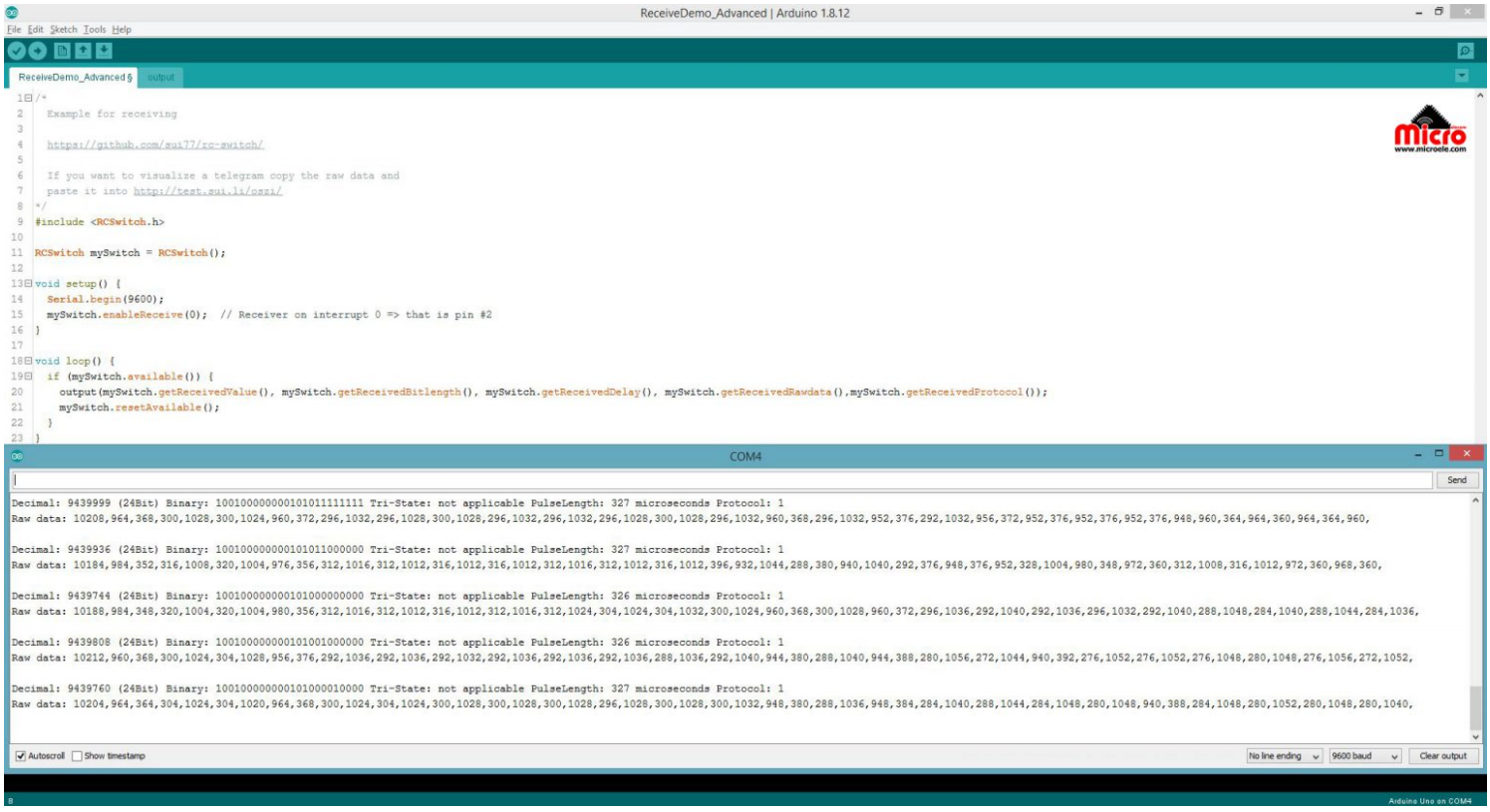
برای دسترسی به این سورس کد، مراحل زیر را دنبال می کنیم:

File --> Examples --> rc-switch-master --> RecieveDemo_Advanced

سورس کد این مثال:

```
#include <RCSwitch.h>
RCSwitch mySwitch = RCSwitch();
void setup() {
  Serial.begin(9600);
```

```
  mySwitch.enableReceive(0);  // Receiver on interrupt 0 => that is pin #2
}
void loop() {
if (mySwitch.available())
{
output(mySwitch.getReceivedValue(),mySwitch.getReceivedBitlength(),mySwitch.g
etReceivedDelay(),mySwitch.getReceivedRawdata(),mySwitch.getReceivedProtocol(
));
mySwitch.resetAvailable();
  }
}
```



برای نمایش کد کلید فشرده شده بر روی ریموت، نیاز است که ابتدا بدانیم چه کدی برای هر دکمه اختصاص داده شده است. از آنجا که در این آموزش از

ریموت 18 کاناله استفاده شده است، از 24 بیت ارسالی، تعداد 16 بیت بعنوان آدرس ریموت و 8 بیت دیگر بعنوان کد هر کلید اختصاص داده شده است.
جدول زیر مربوط به کد های ارسالی یک نمونه ریموت 18 کاناله می باشد.

ریموت کنترل 18 کاناله (مدل MEC17-LN4)

Key No.	Decimal	Code
1	9439747	10010000000101000000011
2	9439756	10010000000101000001100
3	9439792	10010000000101000110000
4	9439948	10010000000101011001100
5	9439951	10010000000101011001111
6	9439804	10010000000101000111100
7	9439807	10010000000101000111111
8	9439984	10010000000101011110000
9	9439939	10010000000101011000011
10	9439759	10010000000101000001111
11	9439996	10010000000101011111100
12	9439987	10010000000101011110011
13	9439999	10010000000101011111111
14	9439936	10010000000101011000000
15	9439795	10010000000101000110011
16	9439744	10010000000101000000000
17	9439808	10010000000101001000000
18	9439760	10010000000101000010000

برای اینکه 8 بیت کد هر کلید را بدست آوریم از دستور زیر استفاده می کنیم. در تابع "output" ابتدا محتوای دریافتی را توسط کد زیر به آرایه تبدیل کرده:

```
void output(unsigned long decimal, unsigned int length)
{
  b = dec2binWzerofill(decimal, length);
  b_length = b.length() + 1;
  b.toCharArray(input_array, b_length);
```

```
}
```

و توسط دستور زیر، 8 بیت آخر آن را جدا می کنیم:

```
code = b.substring(16, 24);
```

همانطور که در جدول فوق مشخص است، این ریموت دارای آدرس "1001000000001010" می باشد. پس ما باید با دستورات شرطی، 8 بیت کد هر کلید را مورد بررسی قرار داده و عملیات مد نظر خود مثلا روشن و خاموش کردن یک LED را انجام دهیم. همین کد را می توان برای سایر کلید های ریموت نیز نوشت و تا 18 عدد خروجی که در اینجا LED وصل شده است، این مرحله را پیش برد.

```
if (code == "00000011")
{
      state1 = state1 + 1;
      Serial.println("1");
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("Button Number:");
      lcd.setCursor(8, 1);
      lcd.print("1");
      if (state1 == 1)
      {
        digitalWrite(A0, HIGH);
      }
      if (state1 == 2)
      {
        digitalWrite(A0, LOW);
        state1 = 0;
      }
```

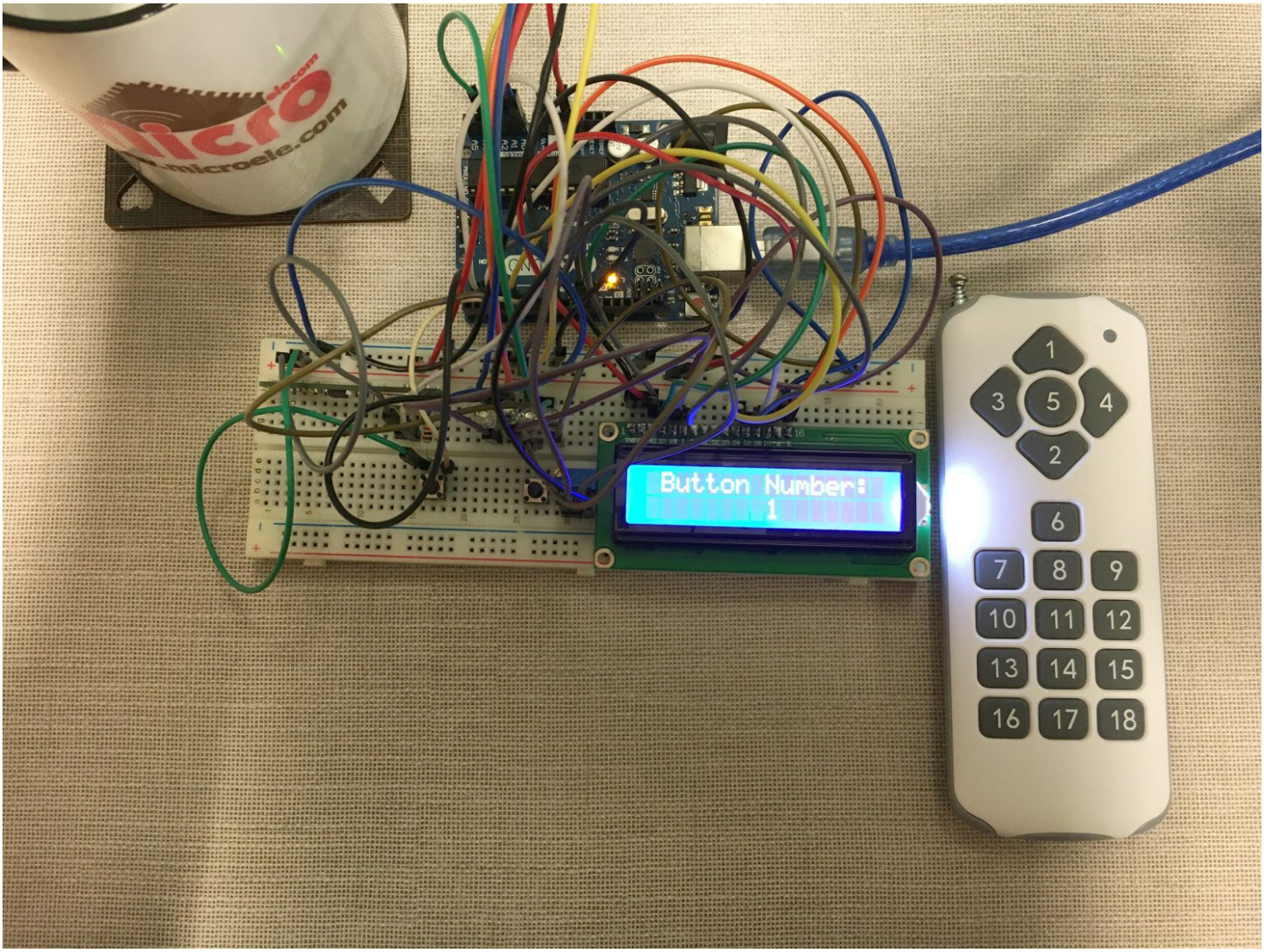
```
}
```

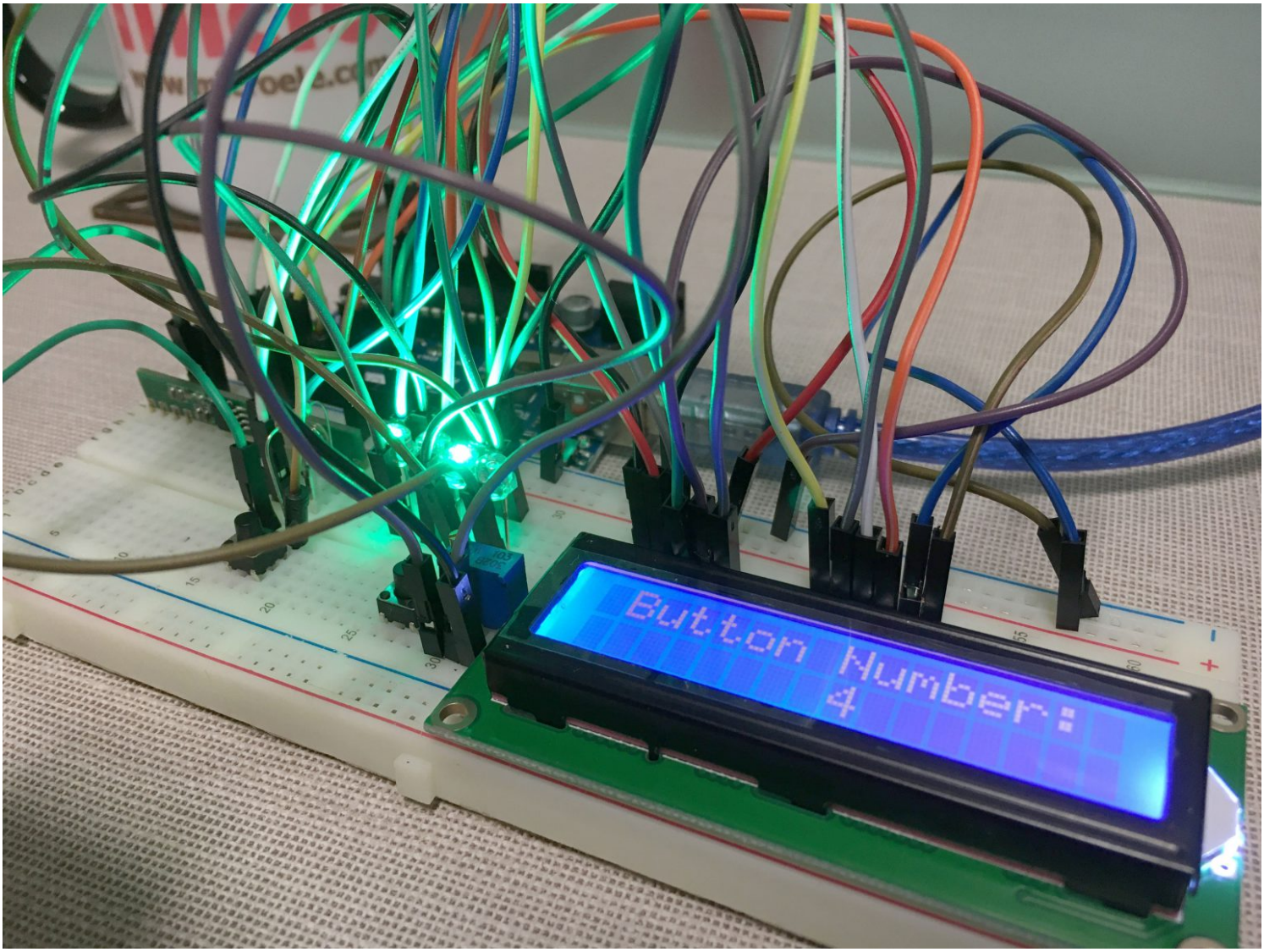
برای دانلود سورس کد این پروژه می توانید از این لینک استفاده نمایید.

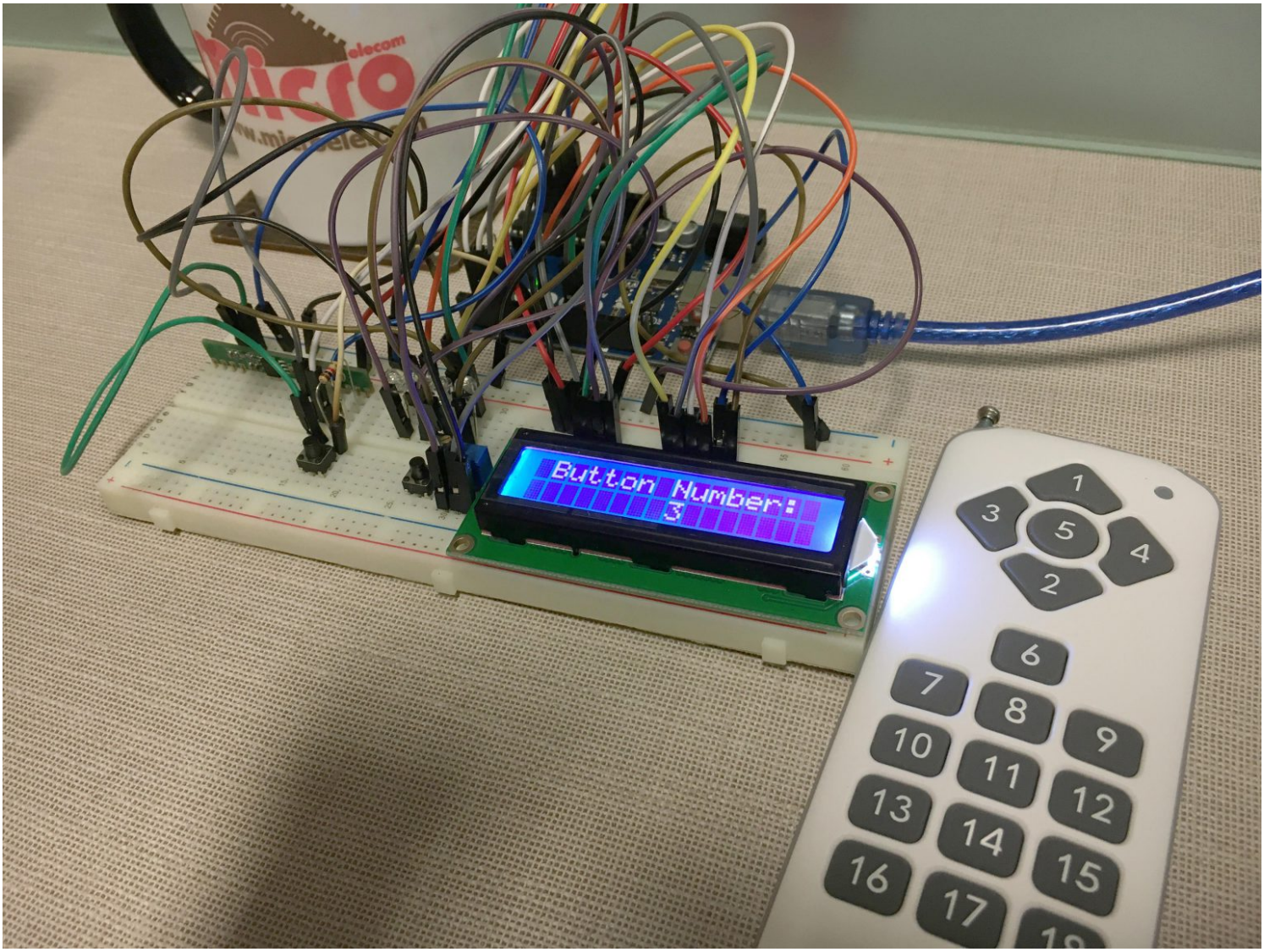
امیدوارم از این آموزش کمال استفاده را داشته باشید. چنانچه در اجرای پروژه در هر قسمت سوال یا نظری داشتید در انتهای همین صفحه در بخش نظرات اون را با ما در میان بگذارید تا سریع جوابش رو بهتون بگیم.

پروژه رو اجرا کنید و اون رو در پیج اینستاگرام خودتون با هشتگ (#microelecom) انتشار بدید و ما رو مطلع کنید تا انرژی بیشتری بگیریم و پروژه های خفن تری رو برای آموزش قرار بدیم.

تصاویر:


Button Number:
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18


Button Number:
4


Button Number:
3
1
3
5
4
2
6
7
8
9
10
11
12
13
14
15
16
17