



دانشکده مهندسی برق، کامپیوتر و فناوری‌های پیشرفته

دستور کار آزمایشگاه ریزپردازنده

تهیه کننده:

مجید نیک‌زاد

تابستان ۱۴۰۳

ویرایش ۱

۱	آشنایی با بردهای آزمایشگاه
۱	برد آردوینو UNO:
۱	بین های ورودی و خروجی
۳	نقشه اتصال بین های آردوینو به پورت های ATMEGA328
۳	خلاصه برد UNO
۴	آردوینو MEGA
۴	خلاصه برد MEGA
۵	نقشه اتصال بین های آردوینو به پورت های ATMEGA2560
۵	محیط برنامه نویسی آردوینو
۶	سریال مانیتور:
۶	دستورات سریال مانیتور:
۷	آزمایش ۱: آشنایی با پورت های برد آردوینو و پیاده سازی مدار LED چشمک زن
۷	دستورات آزمایش ۱:
۷	آزمایش ۱-۱
۷	آزمایش ۲-۱
۷	آزمایش ۳-۱
۷	آزمایش ۴-۱
۸	آزمایش ۲: آشنایی با نمایشگر 7SEGMENT و طراحی شمارنده دو رقمی
۸	طرز کار نمایشگر 7SEGMENT
۹	آزمایش ۱-۲
۹	آزمایش ۲-۲
۹	آزمایش ۳-۲
۱۰	آزمایش ۳: آشنایی با KEYPAD و LCD کاراکتری و طراحی ماشین حساب ساده
۱۰	آشنایی با KEYPAD و LCD کاراکتری
۱۲	دستورات LCD:
۱۲	آزمایش ۱-۳
۱۲	آزمایش ۲-۳
۱۳	آزمایش ۴: آشنایی با واحد ADC میکروکنترلر و طراحی دماسنج دیجیتال
۱۳	واحد ADC: (مبدل آنالوگ به دیجیتال : ANALOG TO DIGITAL CONVERTOR)
۱۴	دستورات ADC:
۱۴	سنسور دمای LM35:

۱۴	آزمایش ۱-۴
۱۴	آزمایش ۲-۴
۱۴	آزمایش ۳-۴
۱۵	آزمایش ۵: زمانبندی و وقفه ها در آردوینو
۱۵	زمانبندی:
۱۶	وقفه های خارجی:
۱۶	دستور ایجاد ISR برای یکی از پایه های وقفه دار:
۱۷	آزمایش ۱-۵
۱۷	آزمایش ۲-۵
۱۷	آزمایش ۳-۵
۱۸	آزمایش ۶: ماژول RFID و طراحی سیستم کنترل تردد
۱۸	طریقه راه اندازی ماژول RFID:
۱۹	آزمایش ۱-۶
۱۹	آزمایش ۲-۶
۲۰	آزمایش ۷: ارتباط سریال
۲۱	آزمایش ۱-۷
۲۱	آزمایش ۲-۷
۲۲	پروژه آزمایشگاه
۲۲	لیست پروژهها:
۲۲	۱- طراحی سیستم کنترل شرایط محیطی گلخانه با دو سنسور و دو دستگاه
۲۲	۲- طراحی مولد پالس با فرکانس و سیکل کاری قابل تنظیم با دو خروجی
۲۲	۳- طراحی سیستم کنترل تردد با امکان ثبت ورود و خروج و گزارش دهی

آشنایی با بردهای آزمایشگاه

برد آردوینو UNO:

برد آردوینو Uno یک میکروکنترلر بر پایه ATmega328 می باشد. این برد ۱۴ پین ورودی و خروجی دیجیتال (که ۶ تای آن می تواند به عنوان خروجی PWM استفاده گردد)، ۶ ورودی آنالوگ، یک تشدیدگر سرامیکی ۱۶مگاهرتز (Ceramic Resonator)، یک پورت USB، یک پاورجک (ورودی منبع تغذیه)، یک ICSP header و یک دکمه ریست دارد. برد Uno شامل کلیه امکانات مورد نیاز جهت بکارگیری میکروکنترلر موجود بر روی برد می باشد. برای شروع تنها با یک کابل USB، به سادگی برد را به کامپیوترتان متصل کنید و یا آن را با یک آداپتور AC-To-DC و یا باتری راه اندازی نمایید.



شکل (۱-۱) تصویر برد UNO

پین های ورودی و خروجی

هریک از ۱۴ پین دیجیتال Uno می تواند با استفاده از توابع

`pinMode()`, `digitalWrite()`, `digitalRead()`

به عنوان ورودی یا خروجی استفاده شود. ولتاژ پین ها ۵ ولت بوده و ظرفیت جریان جهت هر پین حداکثر ۴۰ میلی آمپر می باشد. همچنین هر یک از این پین ها دارای یک مقاومت داخلی (۲۰-۵۰ کیلو اهم) جهت

Pull-Up می باشد (که به صورت پیش فرض غیرفعال است). بعلاوه بعضی از پین ها دارای عملکردهای منحصر به فردی می باشند که شرح آن در ذیل آمده است:

Serial - 0 (RX) و 1 (TX) : پین RX برای دریافت و TX جهت انتقال اطلاعات به صورت سریال و با پروتکل TTL استفاده می شود. این پین ها به پین های مرتبط **ATmega8U2 USB-to-TTL** متصل هستند.

External interrupts (وقفه های خارجی) - ۲ و ۳: این پین ها می توانند طوری تنظیم شوند که یک وقفه را براساس اندکی افزایش یا کاهش لبه، و یا هر نوع تغییر در مقدار، ایجاد نمایند.

PWM - 3,5,6,9,10,11 : امکان دسترسی به یک خروجی **PWM** هشت بیتی را با استفاده از تابع **analogWrite()** فراهم می کنند.

SPI - 10(SS), 11(MOSI), 12(MISO), 13(SCK): با استفاده از توابع کتابخانه ای **SPI** این پین ها می توانند یک ارتباط **SPI library** ایجاد نمایند.

LED - 13: یک **LED** آماده، به پین دیجیتال ۱۳ متصل شده است. هنگامی که پین در حالت **HIGH** قرار دارد، **LED** روشن و زمانی که پین در حالت **LOW** قرار دارد، خاموش می شود.

برد **Uno** ۶ ورودی آنالوگ دارد که از **A0** تا **A5** نامگذاری شده اند. میزان تفکیک پذیری هر یک از پین ها تا ۱۰ بیت می باشد (به عنوان ۱۰۲۴ مقدار مختلف). به صورت پیش فرض این پین ها می توانند ولتاژی بین ولتاژ پایه (**Ground**) تا حداکثر ۵ ولت را اندازه گیری نمایند. ولی با استفاده از پین **AREF** و تابع **analogReference()** تغییر حد بالای میزان تفکیک پذیری امکان پذیر می باشد. همچنین بعضی از پین ها دارای عملکردهای منحصر به فردی می باشند که شرح آن در ذیل آمده است:

I2C: پین **A4** یا **SDA** و **A5** یا **SCL**: این پین ها امکان ایجاد یک ارتباط **I2C** را با استفاده از توابع کتابخانه ای **Wire** مقدور می سازند.

سایر پین ها:

AREF: ولتاژ مرجع برای ورودی های آنالوگ، از طریق این پین و با استفاده از تابع **analogReference()** تأمین می گردد.

Reset: وضعیت لاین مرتبط را برای ریست میکروکنترلر در حالت **LOW** قرار می دهد، معمولاً زمانی از این پین استفاده می شود که بخواهید بر روی شیلدتان دکمه ریست قرار دهید. زیرا استفاده از شیلدها از دسترسی به دکمه ریست موجود بر روی برد آردوینو جلوگیری می کند.

نقشه اتصال بین پین های آردوینو به پورتهای ATmega328

نقشه اتصال پین ها برای ATmega8، ATmega168 و ATmega328 یکسان بوده و در شکل ۱-۲ قابل مشاهده می باشد.

Arduino function				Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13) analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12) analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11) analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10) analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9) analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8) analog input 0
VCC	VCC	7	22	GND GND
GND	GND	8	21	AREF analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5) digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4) digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3) digital pin 11 (PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2) digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1) digital pin 9 (PWM)

شکل ۱-۲) نقشه اتصال بین پین های آردوینو به پورتهای ATmega328

خلاصه برد UNO

ATmega328

۵ ولت
 ۷-۱۲ ولت
 ۶-۲۰ ولت
 ۱۴ (۶ تا آن به عنوان خروجی PWM استفاده می شود).
 ۶
 ۴۰ میلی آمپر
 ۵۰ میلی آمپر
 ۳۲ کیلوبایت که ۰.۵ کیلوبایت از آن مورد استفاده BootLoader قرار می گیرد.
 ۲ کیلوبایت
 ۱ کیلوبایت
 ۱۶ مگاهرتز

میکروکنترلر

ولتاژ عملیاتی

ولتاژ ورودی (پیشنهادی)

ولتاژ ورودی (محدوده)

پین های دیجیتال ورودی/خروجی

پین های ورودی آنالوگ

جریان DC هر پین ورودی و خروجی

جریان DC جهت پین ۳.۳V

حافظه فلش

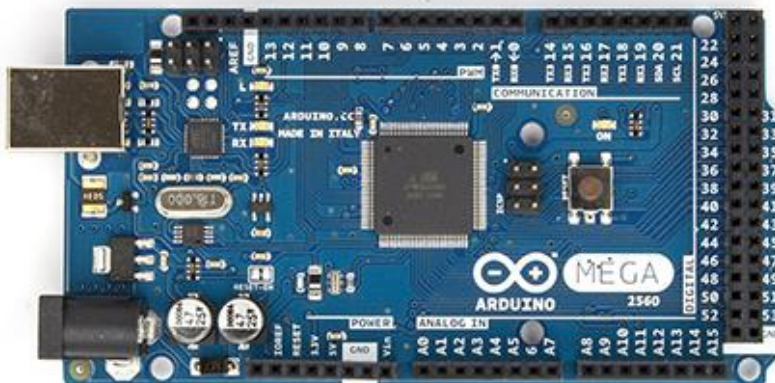
SRAM

EEPROM

سرعت ساعت

آردوینو Mega

برد آردوینو Mega2560 یک برد میکروکنترلر بر پایه ATmega2560 می باشد. این برد مجموعاً ۵۴ پین ورودی/خروجی دیجیتال (که ۱۵ تای آن می تواند به عنوان خروجی PWM استفاده گردد)، ۱۶ ورودی آنالوگ، ۴ پورت UART (پورت های سریال سخت افزاری)، به مانند برد UNO یک نوسان ساز کریستال ۱۶ مگاهرتز، یک پورت USB، یک پاورجک، یک ICSP Header و یک دکمه ریست دارد.



شکل ۱-۳) تصویر برد Mega

خلاصه برد Mega

ATmega2560

۵ ولت

۷-۱۲ ولت

۶-۲۰ ولت

۵۴ (۱۵ تای آن به عنوان خروجی PWM استفاده می شود).

۱۶

۴۰ میلی آمپر

۵۰ میلی آمپر

۲۵۶ کیلوبایت (۸ کیلوبایت از آن توسط BootLoader استفاده می شود).

۸ کیلوبایت

۴ کیلوبایت

۱۶ مگاهرتز

میکروکنترلر

ولتاژ عملیاتی

ولتاژ ورودی (پیشنهادی)

ولتاژ ورودی (محدوده)

پین های دیجیتال ورودی/خروجی

پین های آنالوگ ورودی

جریان DC هر پین ورودی/خروجی

جریان DC پین ۳.۳ ولتی

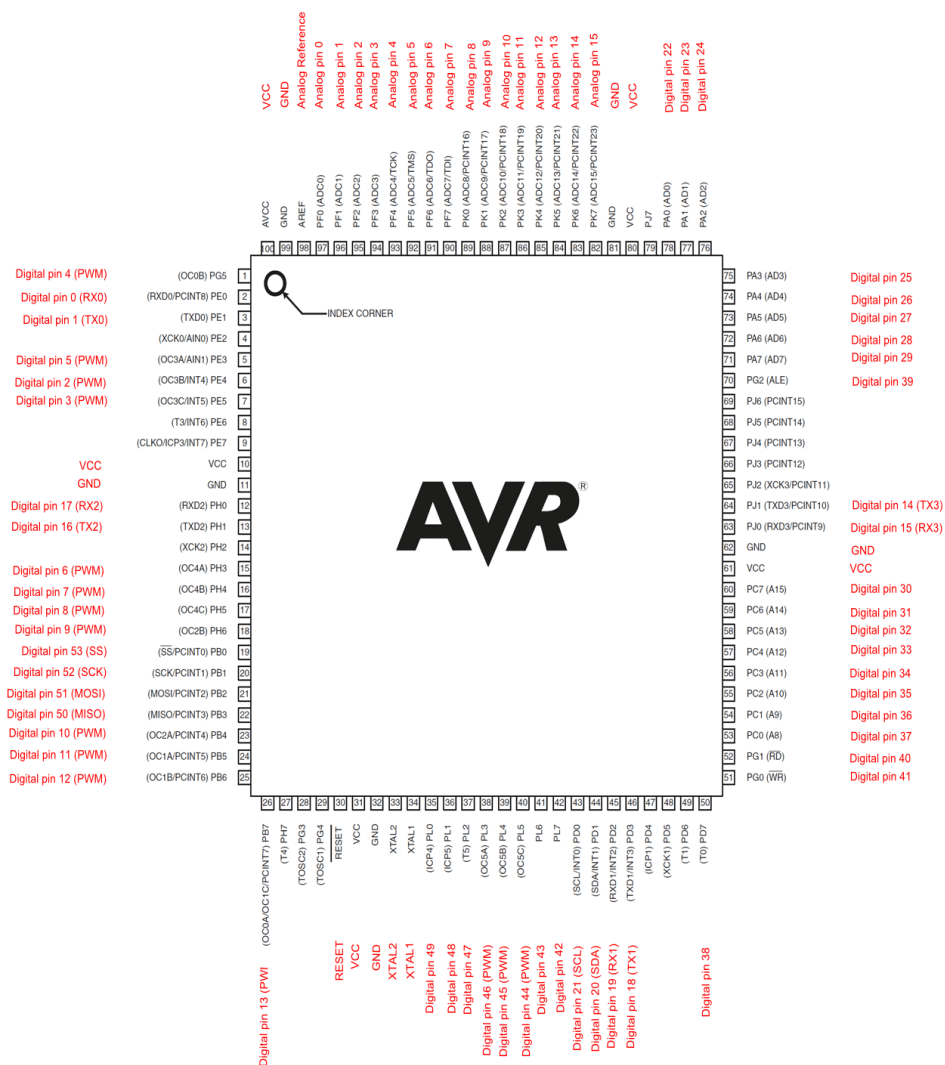
حافظه فلش

SRAM

EEPROM

سرعت ساعت

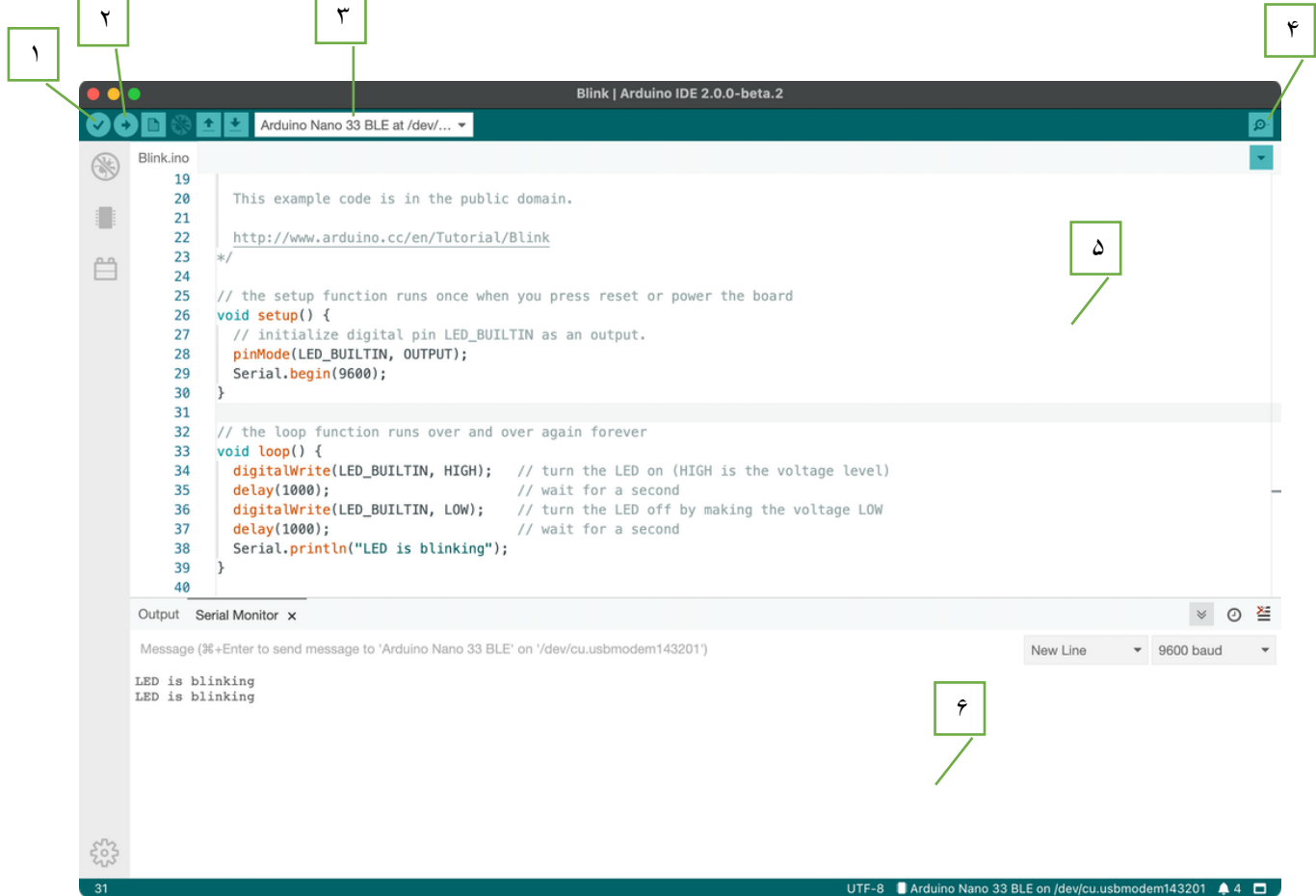
نقشه اتصال بین پین های آردوینو به پورت های ATmega2560



شکل (۱-۴) نقشه اتصال بین پین های آردوینو به پورت های Mega

محیط برنامه نویسی آردوینو

برای برنامه نویسی برد آردوینو از Arduino IDE استفاده خواهد شد. در شکل ۱-۵ نمای محیط برنامه نویسی را مشاهده می فرمائید. توسط کلید ۱ کدهای نوشته شده متناسب با برد انتخاب شده کامپایل می شود. کلید ۲ برنامه نوشته شده را بعد از کامپایل در برد آردوینو متصل شده به کامپیوتر آپلود می کند. توسط پنجره ۳ می توان نوع بردی که با آن کار می کنیم را انتخاب کرد. کلید ۴ پنجره سریال مانیتور را باز و بسته می کند. در پنجره ۵ کدهای برنامه به زبان C/C++ نوشته می شود و در پنجره ۶ می توان داده های ارسال شده توسط برد آردوینو را مشاهده کرد.



شکل (۵-۱) Arduino IDE

سریال مانیتور:

برای ارتباط PC با برد آردوینو می‌توان از سریال مانیتور استفاده نمود. از طریق سریال مانیتور هم می‌توان داده‌های مورد نیاز را از برد آردوینو به PC و برعکس ارسال نمود.

دستورات سریال مانیتور:

دستور راه‌اندازی سریال برد آردوینو UNO:

```
Serial.begin(speed);
speed: 4800, 9600, ... , 2Mbps
```

دستور ارسال داده از برد به PC:

```
Serial.println(val);
val: متن یا هر داده‌ای
```

دستور بررسی تعداد بایت موجود در بافر ورودی سریال:

```
Serial.available();
```

دستور خواندن یک بایت از بافر ورودی سریال:

```
Serial.read();
```

دستور خواندن string بصورت کامل از بافر ورودی:

```
Serial.readString();
```

مقدار برگشتی یک متغیر string می‌باشد.

آزمایش ۱: آشنایی با پورت‌های برد آردوینو و پیاده‌سازی مدار LED چشمک زن

دستورات آزمایش ۱:

دستور تعیین وضعیت پایه (ورودی یا خروجی):

```
pinMode (pin, mode)
```

pin: شماره پایه آردوینو

mode: INPUT, OUTPUT, یا INPUT_PULLUP

دستور خواندن وضعیت پایه: (0 یا 1 منطقی)

```
digitalRead (pin)
```

مقدار برگشتی تابع: 0 (LOW) یا 1 (HIGH) منطقی

دستور نوشتن وضعیت پایه: (0 یا 1 منطقی)

```
digitalWrite (pin, value)
```

value: 0 یا LOW , 1 یا HIGH

دستور تاخیر:

```
delay (ms)
```

تاخیر به میلی‌ثانیه: ms

آزمایش ۱-۱

LED روی برد را به صورت متناوب یک ثانیه روشن و یک ثانیه خاموش کنید.

آزمایش ۲-۱

توسط یک pushButton بصورت لحظه‌ای LED روی برد را روشن کنید.

آزمایش ۳-۱

توسط یک pushButton بصورت دائمی LED روی برد را روشن و خاموش کنید.

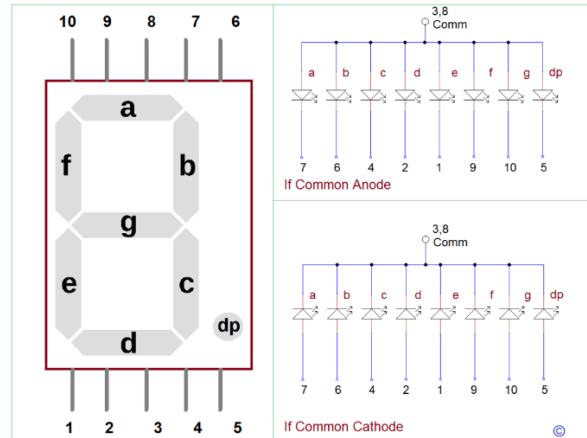
آزمایش ۴-۱

توسط یک pushButton یک شمارنده بسازید و مقدار شمارش شده را توسط چهار LED بصورت باینری نمایش دهید. (راهنمایی: برای سهولت در کدنویسی می‌توانید از رجیسترهای PORTX، DDRX استفاده کنید.)

آزمایش ۲: آشنایی با نمایشگر 7segment و طراحی شمارنده دو رقمی

طرز کار نمایشگر 7Segment

نمایشگر هفت قسمتی از هفت LED که متناسب با نوع نمایشگر (آند مشترک یا کاتد مشترک) بصورت شکل ۲-۱ به هم متصل شده‌اند. برای روشن شدن هر قسمت باید ولتاژ مثبت به آند و ولتاژ منفی به کاتد LED وصل گردد. همیشه برای محدود کردن جریان LED ها یک مقاومت حدود چندصد اهم با آن سری شود.



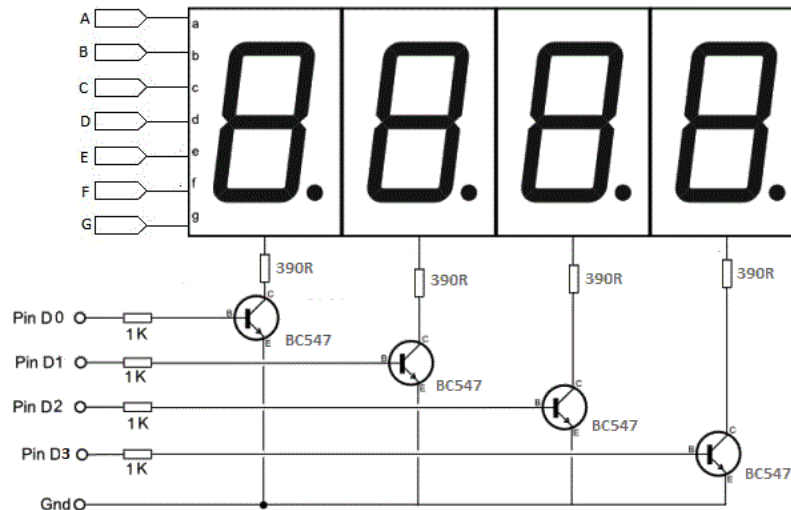
شکل ۲-۱) ساختار نمایشگر هفت قسمتی (7Seg)

برای نمایش ارقام روی نوع کاتد مشترک می‌توان از جدول زیر کمک گرفت و یکی از پورت‌ها را به ترتیب به A تا G وصل نمود.

Decimal	ABCDEFG	A	B	C	D	E	F	G
0	0x7E	1	1	1	1	1	1	0
1	0x30	0	1	1	0	0	0	0
2	0x6D	1	1	0	1	1	0	1
3	0x79	1	1	1	1	0	0	1
4	0x33	0	1	1	0	0	1	1
5	0x5B	1	0	1	1	0	1	1
6	0x5F	1	0	1	1	1	1	1
7	0x70	1	1	1	0	0	0	0
8	0x7F	1	1	1	1	1	1	1
9	0x7B	1	1	1	1	0	1	1

جدول ۲-۱) جدول مقدار هشت بیتی مورد نیاز برای نمایش هر رقم در نمایشگر نوع کاتد مشترک

اگر تعداد بیشتر از یک رقم را بخواهیم نمایش دهیم برای هر رقم یک پورت نیاز داریم. این ساختار بدلیل اشغال تعداد زیاد پایه‌های میکروکنترلر اصلاً توصیه نمی‌شود. برای رفع این مشکل می‌توان از تقسیم زمانی برای نمایش ارقام استفاده کرد. بدین شکل که با انتخاب فقط یک نمایشگر در هر لحظه و اعمال رقم متناظر با آن یک عدد چند رقمی را فقط با یک پورت نمایش داد. البته به تعداد رقم‌ها پایه از یک پورت دیگر نیاز داریم.



شکل ۲-۲) ساختار نمایشگر هفت قسمتی ۴ رقمی

آزمایش ۱-۲

توسط رجیسترهای DDRX و PORTX یک رقم به دلخواه روی یک نمایشگر 7Seg نمایش دهید.

آزمایش ۲-۲

یک ثانیه شمار دو رقمی با قابلیت شروع و توقف و بازنشانی طراحی و پیاده‌سازی نمایید.

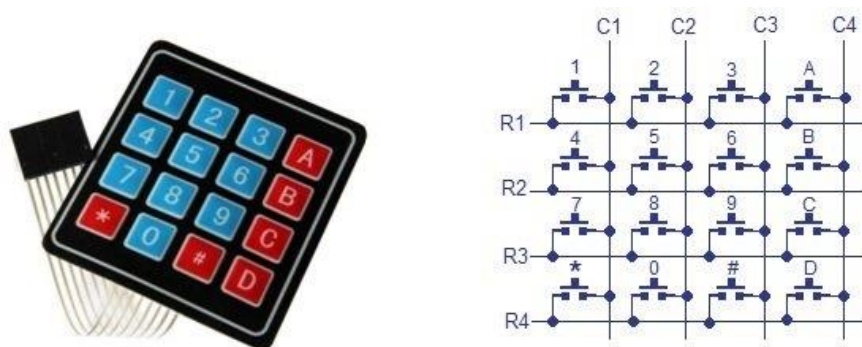
آزمایش ۳-۲

ثانیه شمار بخش قبل را بدون تابع delay طراحی کنید. (راهنمایی: از تابع millis() استفاده نمایید).

آزمایش ۳: آشنایی با keypad و LCD کاراکتری و طراحی ماشین حساب ساده

آشنایی با keypad و LCD کاراکتری

برای انجام تنظیمات و وارد کردن اعداد در سیستم‌های دیجیتال می‌توان از ترکیب کی پد و LCD با ابعاد مختلف استفاده کرد. در شکل ۱-۳ نمونه یک Keypad 4*4 را مشاهده می‌کنید. در این کی پد چهار سطر و چهار ستون داریم. سطرها توسط کلید به ستون‌ها وصل می‌باشند. این ساختار باعث می‌شود که به سادگی محل کلید فشرده شده را تشخیص دهیم. به عنوان مثال اگر کلید شماره ۶ فشرده شود سطر ۲ به ستون ۳ وصل خواهد شد. یک روش برای تشخیص کلید فشرده شده توسط میکروکنترلر اعمال مقادیر طبق جدول ۱-۳ به سطرها و پایش مقادیر ستون‌ها می‌باشد. حال اگر مقدار صفر به سطر دوم و یک به سطرهای دیگر اعمال شود و فقط ستون ۳ مقدار صفر به خود بگیرد نشان دهنده فشرده شدن کلید شماره ۶ می‌باشد.



شکل ۱-۳ کی پد ۴*۴

R1	R2	R3	R4	C1	C2	C3	C4	نتیجه
1	1	1	0	1	1	1	1	NO KEY
				1	1	1	0	D
				1	1	0	1	#
				1	0	1	1	0
				0	1	1	1	*
1	1	0	1	1	1	1	1	NO KEY
				1	1	1	0	C
				1	1	0	1	9
				1	0	1	1	8
				0	1	1	1	7
1	0	1	1	1	1	1	1	NO KEY
				1	1	1	0	B
				1	1	0	1	6
				1	0	1	1	5
				0	1	1	1	4
0	1	1	1	1	1	1	1	NO KEY
				1	1	1	0	A
				1	1	0	1	3
				1	0	1	1	2
				0	1	1	1	1

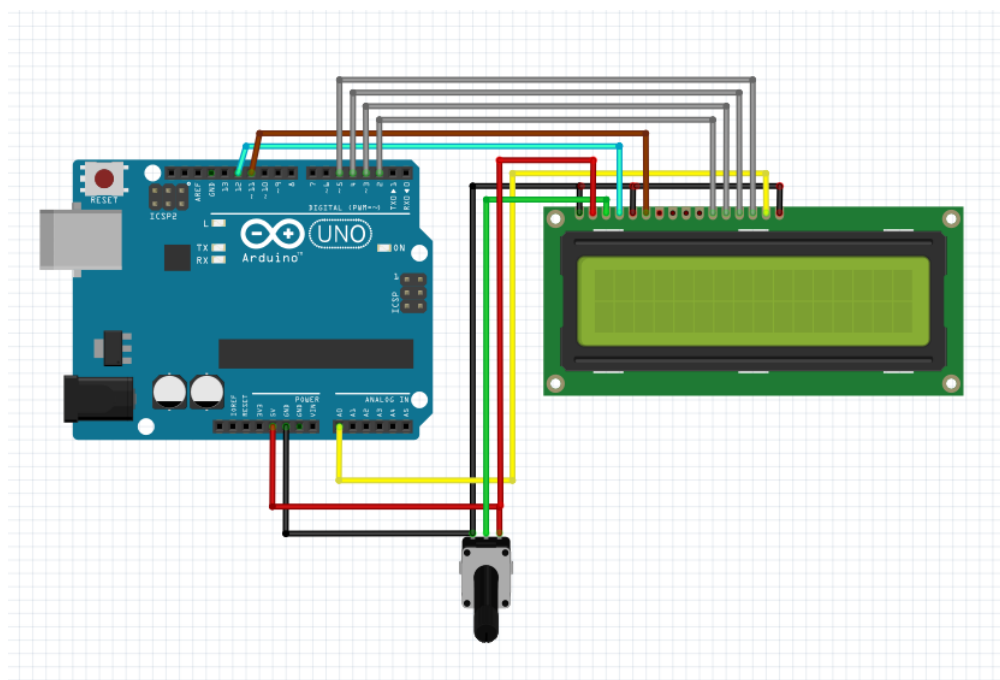
جدول ۱-۳ حالت‌های مختلف برای پایش کلید فشرده شده برای کی پد ۴*۴

LCDهای کاراکتری برای نمایش حروف و اعداد طراحی شده‌اند (هرچند با دسترسی به رجیسترهای آنها می‌توان اشکال را نیز در آنها نمایش داد). در شکل ۲-۳ یک LCD کاراکتری ۱۶*۲ را مشاهده می‌کنید. این LCD شامل ۱۶ ستون و ۲ سطر می‌باشد.



شکل ۲-۳ تصویر LCD کاراکتری ۱۶*۲

طریقه اتصال LCD و آردوینو:



شکل ۳-۳ طریقه اتصال LCD و آردوینو UNO

راه اندازی LCD با ۱۶ ستون و ۲ سطر:

```
lcd.begin(16, 2);
```

چاپ متن یا عدد روی LCD:

```
lcd.print("متن");
```

پاک کردن LCD:

```
lcd.clear();
```

انتقال مکان نما به سطر و ستون دلخواه:

```
lcd.setCursor(سطر, ستون);
```

آزمایش ۱-۳

توسط یک کی پد کلید فشرده شده را روی LCD نمایش دهید.

آزمایش ۲-۳

یک ماشین حساب ساده طراحی کنید که :

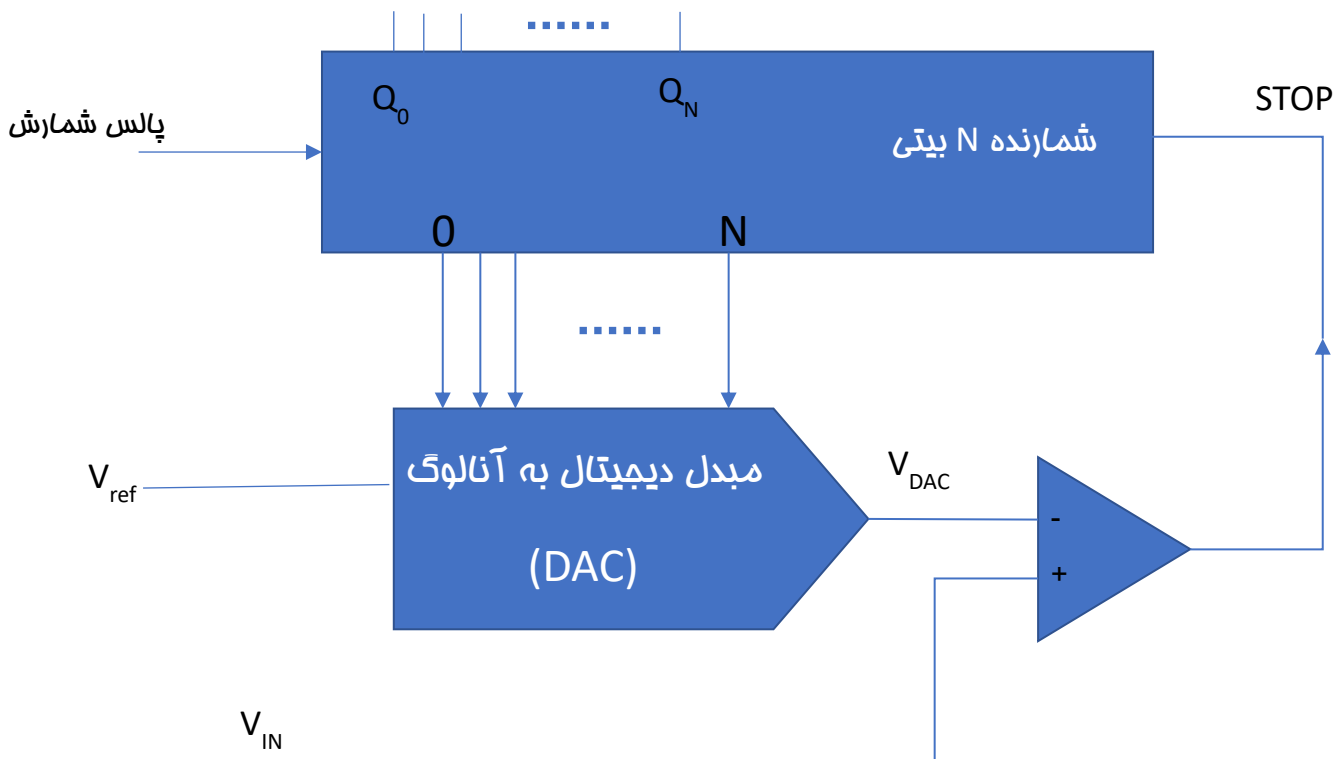
- چهار عمل اصلی را انجام دهد.
- قابلیت پاک کردن صفحه داشته باشد.
- در صورت ورودی اشتباه پیام خطا نمایش دهد.

آزمایش ۴: آشنایی با واحد ADC میکروکنترلر و طراحی دماسنج دیجیتال

واحد ADC: (مبدل آنالوگ به دیجیتال : Analog to digital convertor)

برخی مواقع نیاز داریم تا به جای تشخیص سطح ولتاژ مربوط به صفر یا یک منطقی مقدار دقیق ولتاژ وصل شده به پایه میکروکنترلر را اندازه گیری کنیم. به عنوان مثال خروجی سنسور دمای LM35 متناسب با دما در محدوده بین ۰ تا ۱.۵ ولت تغییر می کند. برای اندازه گیری دما باید ابتدا ولتاژ خروجی سنسور را بوسیله واحد ADC میکروکنترلر اندازه گیری کرد و سپس با قرار دادن مقدار ولتاژ در رابطه بین دما و ولتاژ، مقدار دما را محاسبه کرد.

پیکربندی پیش فرض در آردوینو ولتاژ مرجع ۵ ولت و دقت اندازه گیری ۱۰ بیت می باشد. بلوک دیاگرام واحد ADC در شکل ۴-۱ قابل مشاهده می باشد.



شکل ۴-۱) بلوک دیاگرام واحد ADC میکروکنترلر ATmega328

مقدار خوانده شده توسط تابع `analogRead` را متناسب با ولتاژ مرجع انتخاب شده باید طبق فرمول زیر به مقدار ولتاژ تبدیل کرد:

$$v_{in} = \frac{digitalValue * v_{ref}}{2^n - 1} ; n = \text{تعداد بیت}$$


```
analogRead(pin);
```

pin: A0 ~ A5 (UNO در آردوینو)

مقدار برگشتی: 0 ~ 1023 (اگر در حالت ۱۰ بیت انتخاب شده باشد).

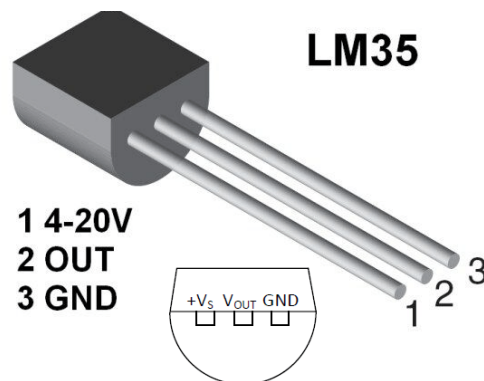
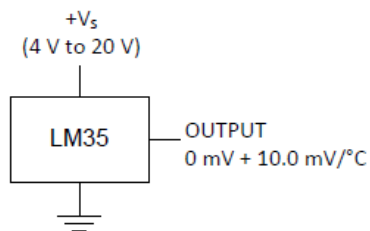
```
analogReference(type);
```

ولتاژ وصل شده به پایه **type: DEFAULT: 5v, INTERNAL: 1.1v, EXTERNAL: AREF**

سنسور دمای LM35:

سنسور LM35 یک سنسور دما با خروجی ولتاژ می‌باشد. طبق شکل ۲-۴ پایه خروجی **OUT** به ازای هر درجه افزایش یا کاهش دما ۱۰ میلی‌ولت افزایش یا کاهش خواهد داشت. در دمای ۲۵ درجه خروجی سنسور حدود ۲۵۰ میلی‌ولت می‌باشد. با توجه به این اطلاعات می‌توان رابطه‌ای بین ولتاژ خروجی سنسور و دمای محیط پیدا کرد. (این رابطه را بدست آورید).

Basic Centigrade Temperature Sensor (2°C to 150°C)



شکل ۲-۴ سنسور دمای LM35

آزمایش ۱-۴

توسط یک پتانسیومتر ولتاژ متغیر ایجاد کرده و مقدار آن را روی LCD نمایش دهید.

آزمایش ۲-۴

با استفاده از سنسور دمای آنالوگ LM35 یک دماسنج دیجیتال طراحی نمایید.

آزمایش ۳-۴

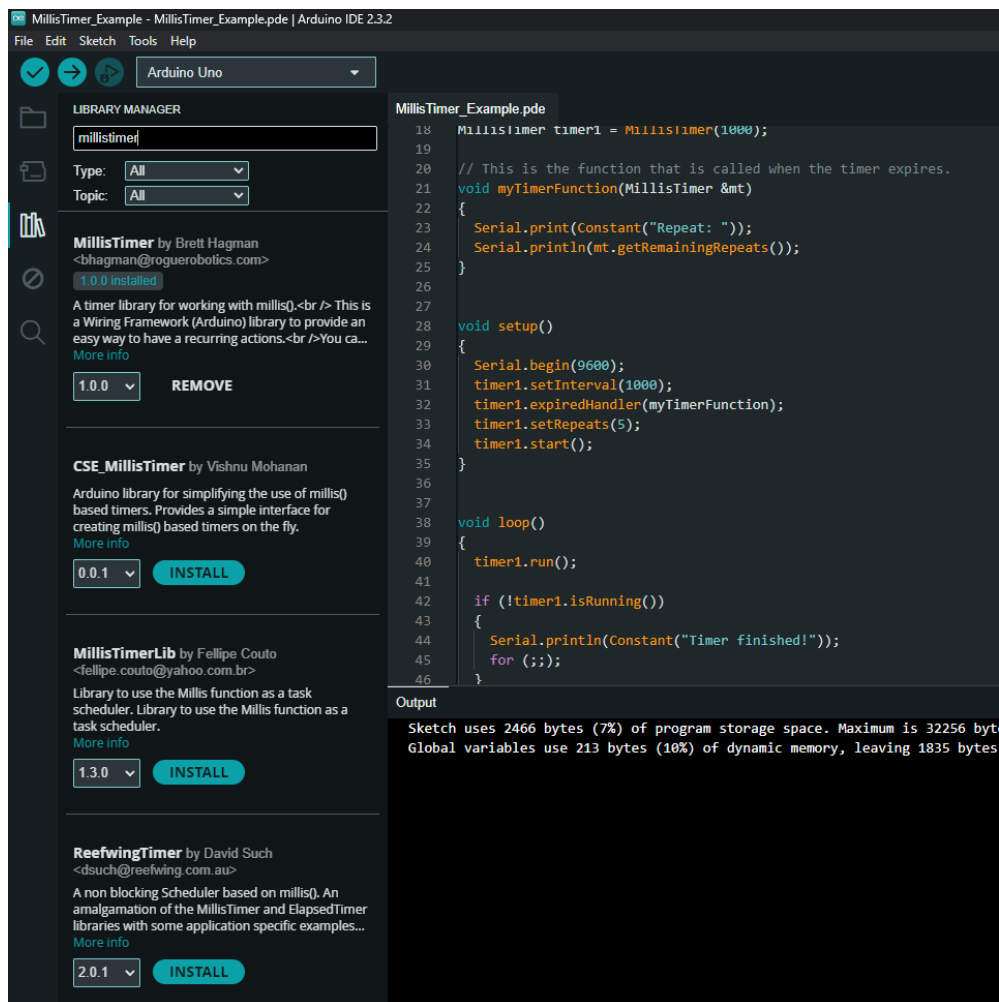
یک کنترل کننده دمای محیط با قابلیت تنظیم دمای مطلوب به روش قطع و وصل طراحی نمایید. (می‌توانید دمای مطلوب را توسط پتانسیومتر تنظیم نمایید).

آزمایش ۵: زمانبندی و وقفه ها در آردوینو

یک سری از مجموعه دستورات مهم و حساس باید در زمان مشخص و در حین اجرای مجموعه دستورات با اولویت پائین تر اجرا شوند. برای رسیدن به این مهم می توان از وقفه ها استفاده نمود. این وقفه ها می توانند هم بصورت متناوب و زمانبندی شده صورت بگیرند و هم همزمان با برخی از پدیده های خارجی همچون فشردن یک کلید یا رسیدن بازوی ربات به یک زاویه بحرانی.

زمانبندی:

برای زمانبندی می توان از کتابخانه `millisTimer` آردوینو استفاده کرد. در بخش `library manager` و قسمت جستجو `millisTimer` را تایپ کنید و بعد از مشاهده در لیست آن را نصب کنید. اگر کتابخانه مذکور به درستی نصب شود مانند شکل ۵-۱ باید به جای `install` کلمه `remove` نوشته شود. در این شکل مثال مربوط به چگونگی استفاده از این کتابخانه آمده است. یکبار این مثال را آپلود کرده و آن را امتحان کنید.



The screenshot shows the Arduino IDE interface. On the left, the Library Manager is open, displaying search results for 'millisTimer'. The first result, 'MillisTimer' by Brett Hagman, is selected and shows a 'REMOVE' button, indicating it is already installed. The code editor on the right shows the following code:

```
18 MillisTimer timer1 = MillisTimer(1000);
19
20 // This is the function that is called when the timer expires.
21 void myTimerFunction(MillisTimer &mt)
22 {
23   Serial.print(constant("Repeat: "));
24   Serial.println(mt.getRemainingRepeats());
25 }
26
27
28 void setup()
29 {
30   Serial.begin(9600);
31   timer1.setInterval(1000);
32   timer1.expiredHandler(myTimerFunction);
33   timer1.setRepeats(5);
34   timer1.start();
35 }
36
37
38 void loop()
39 {
40   timer1.run();
41
42   if (!timer1.isRunning())
43   {
44     Serial.println(constant("Timer finished!"));
45     for (;;);
46   }
```

The Output window at the bottom shows the following text:

```
Sketch uses 2466 bytes (7%) of program storage space. Maximum is 32256 bytes.
Global variables use 213 bytes (10%) of dynamic memory, leaving 1835 bytes
```

شکل ۵-۱) نصب کتابخانه `millisTimer`

در برد UNO فقط پایه های ۲ و ۳ را می توان به عنوان وقفه خارجی استفاده کرد. درحالیکه در برد Mega علاوه بر این پایه ها، پایه های ۱۸، ۱۹، ۲۰ و ۲۱ نیز دارای این قابلیت می باشند.

دستور ایجاد ISR برای یکی از پایه های وقفه دار:

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)
```

ISR: Interrupt Service Routine

mode: **LOW:** حساس به لبه پائین: **RISING:** حساس به لبه بالارونده **FALLING:** حساس به لبه پائین رونده ،

CHANGE: حساس به هر دو لبه

مثال: برنامه ای که در آن تعداد پالس های ورودی را شمرده و هر یک ثانیه در سریال مانیتور چاپ می کند.

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte count = 0;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), counter, RISING);
}

void loop() {
  Serial.println(count);
  delay(1000);
}

void counter() {
  count++;
}
```

دقت فرمائید که در این برنامه متغیر **count** بصورت **volatile** تعریف شده است. **volatile** مشخصاً به کامپایلر دستور می دهد که متغیر را از **RAM** بخواند (لود کند)، نه از یک ثبات ذخیره (انباره). ثبات انباره مکانی برای حافظه ی موقت است که متغیرهای برنامه در آن ذخیره و دست کاری می شوند. تحت شرایطی خاص، مقدار یک متغیری که در ثبات ذخیره شده می تواند غلط باشد.

یک متغیر باید هنگامی به عنوان **volatile** اعلان شود که توسط چیزی فراتر از کنترل قسمتی از کدی که در آن آمده است، مقدارش قابل تغییر باشد. در آردوینو تنها جایی که احتمال رخ دادن این قضیه وجود دارد، در قسمتی از کد است که به وقفه ها (**interrupts**) مرتبط است.

نکته دیگری که باید توجه داشته باشیم این است که در ISR تابع delay() و millis() به درستی کار نمی کنند و احتمال دارد داده های دریافتی از سریال را از دست دهیم.

آزمایش ۵-۱

توسط millisTimer یک LED را با فرکانس ۱ هرتز روشن و خاموش کنید و توسط دو کلید فرکانس را کم و زیاد نمایید.

آزمایش ۵-۲

فرض کنید دو پایه مربوط به وقفه آردوینو به خروجی دو سنسور شمارش وصل شده است. برنامه ای بنویسید که در سریال مانیتور تعداد شمارش شده هر سنسور را بصورت جداگانه بنویسد. (هر موقع شئی روبروی سنسور قرار گیرد خروجی آن ۱ و در غیر اینصورت خروجی آن ۰ خواهد بود).

آزمایش ۵-۳

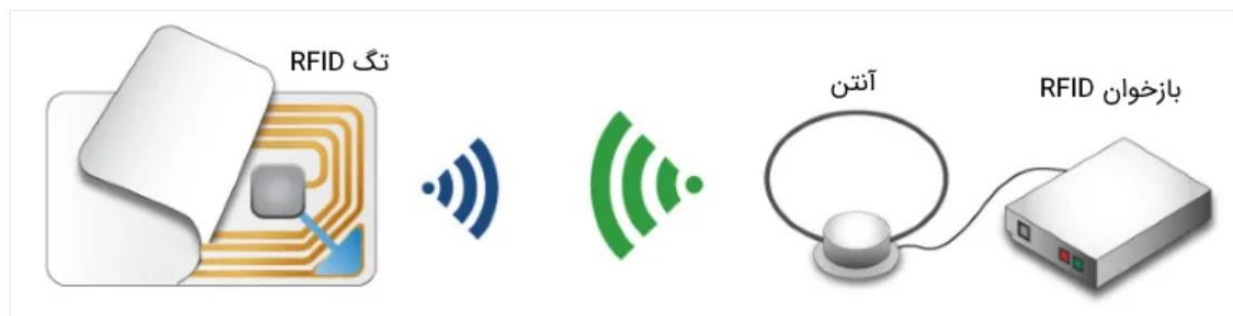
یک فرکانس متر در محدوده فرکانسی ۱۰ تا ۱۰۰۰ هرتز طراحی نمایید.

آزمایش ۶: ماژول RFID و طراحی سیستم کنترل تردد

سامانه بازشناسی با امواج رادیویی (Radio-frequency Identification) یا به اختصار RFID ، روشی برای استفاده بدون تماس از امواج فرکانس رادیویی برای انتقال داده است که به کاربران این امکان را می‌دهد موجودی و دارایی‌ها را به طور خودکار و منحصر به فرد شناسایی و ردیابی کنند.

RFID از اولین کاربرد خود در شناسایی هواپیماهای خودی و دشمن تا کاربردهایی که امروزه پیدا کرده است، نه تنها سال به سال به پیشرفت خود ادامه داده، بلکه هزینه پیاده‌سازی و استفاده از سیستم‌های مرتبط به آن نیز همواره رو به کاهش بوده است این امر باعث شده RFID مقرون به صرفه و کارآمدتر شود.

یک سیستم RFID شامل دو قسمت اصلی است؛ یک فرستنده یا تگ (Tag) که روی جسمی قرار داده شده که می‌خواهیم شناسایی شود و یک ترنسیور (فرستنده-گیرنده) یا یک بازخوان.



شکل ۶-۱) سیستم دو بخشی RFID

طریقه راه اندازی ماژول RFID:

ابتدا باید کتابخانه مربوط به MFRC522 را از بخش Library Manager نصب نمود و سپس با استفاده از یکی از مثال‌های موجود که راهنمایی کامل برای اتصالات در آن نوشته شده است ماژول را به آردوینو وصل نمائیم.

از آنجا که این ماژول از ارتباط SPI استفاده می‌کند باید SPI آردوینو راه‌اندازی شود.

```
SPI.begin(); // Init SPI bus
```

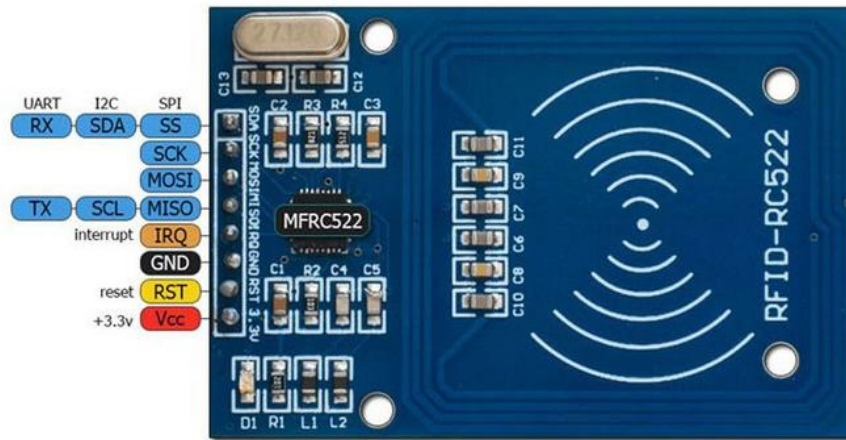
سپس خود ماژول راه‌اندازی می‌شود.

```
rfid.PCD_Init(); // Init MFRC522
```

می‌توان با دستور زیر از تکمیل فرآیند خواند تگ RFID مطمئن شد. در صورت تکمیل فرآیند خواندن این تابع مقدار True را بازمی‌گرداند.

```
rfid.PICC_ReadCardSerial();
```

پس از خواندن ID مربوط به تگ مقدار آن در متغیر آرایه‌ای `rfid.uid.uidByte` قرار می‌گیرد. از آنجائیکه شناسه‌های RFID دارای چهاربایت می‌باشد این آرایه چهار درایه خواهد داشت.



شکل ۶-۲) ماژول خواندن و نوشتن RFID

آزمایش ۶-۱

یکی از مثال‌های موجود در کتابخانه MFRC522 را آپلود کرده و از صحت ماژول و درستی سیم‌بندی خود مطمئن شوید.

آزمایش ۶-۲

یک سیستم کنترل تردد با قابلیت‌های زیر طراحی نمائید:

- اضافه و حذف کارت RFID توسط کارت master
- اجازه عبور به کارت تعریف شده (یک رله به مدت ۵ ثانیه وصل و سپس قطع گردد).
- اعلام هشدار در صورت تشخیص کارت نامعتبر

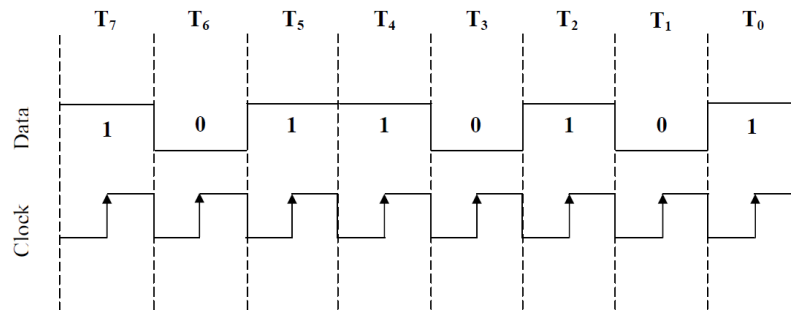
(برای اضافه کردن کارت به سیستم نیاز به استفاده از EEPROM آردوینو می‌باشد. با استفاده از مثال مربوط به EEPROM دستورات نوشتن و خواندن در آن را آموخته و به کار برید.)

آزمایش ۷: ارتباط سریال

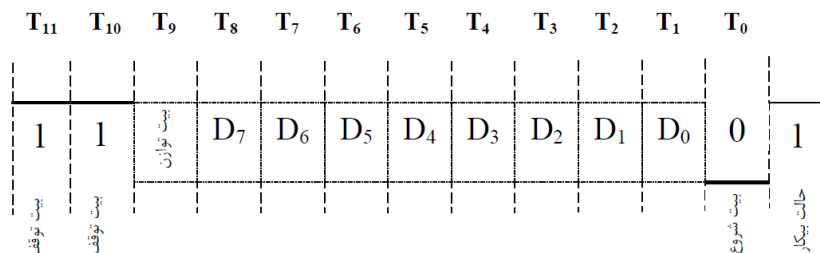
برای ارتباط بین دو سخت‌افزار می‌توان داده‌ها را هم بصورت موازی و هم بصورت سریال ارسال کرد. در حالت موازی نیاز به خطوط بیشتر می‌تواند در اغلب مواقع بدلیل اشغال پایه‌های بیشتر از تراشه مشکل‌زا باشد. بخاطر همین تاجائیکه سرعت اجازه می‌دهد تمایل به سمت استفاده از ارتباط سریال می‌باشد.

یکی از مرسوم‌ترین روش‌های ارتباط سریال $USART^1$ می‌باشد.

این ارتباط می‌تواند هم به روش همزمان (شکل ۷-۱) و هم به روش غیرهمزمان (شکل ۷-۲) صورت گیرد. روش غیرهمزمان بدلیل نیاز به خطوط کمتر بیشتر مورد استفاده قرار می‌گیرد. در این نوع ارتباط باید قالب و نرخ انتقال داده در هر دو طرف (فرستنده و گیرنده) یکسان باشد.



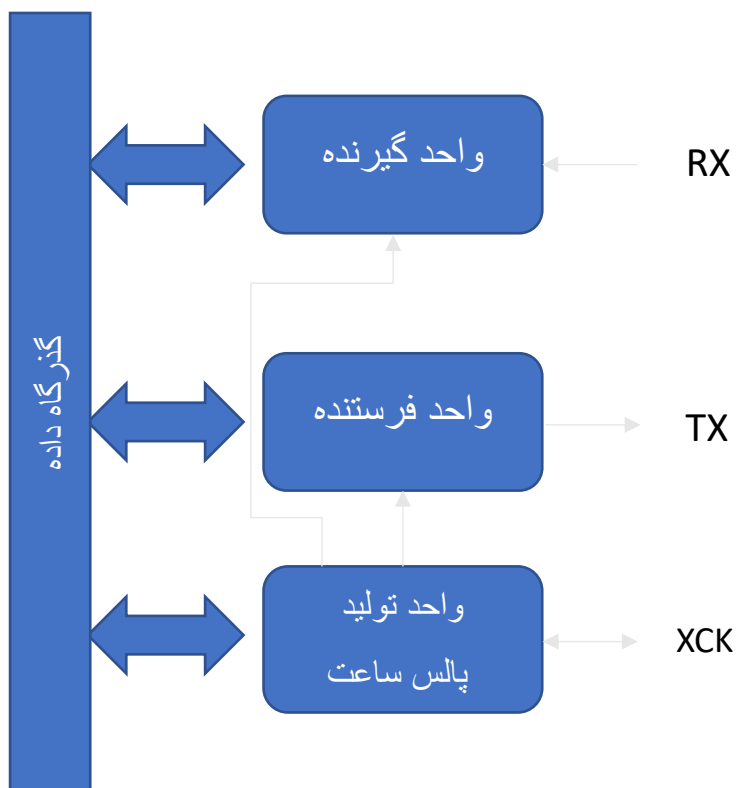
شکل ۷-۱) ارتباط سریال بصورت Synchronous



شکل ۷-۲) ارتباط سریال بصورت Asynchronous

در حالت غیرهمزمان به سه خط TX، RX و GND نیاز داریم. RX تراشه اول باید به TX تراشه دوم و بالعکس وصل گردد. پرواضح است که GND دو تراشه باید به‌همدیگر متصل شود تا مرجع مقایسه سطوح ولتاژ برای تشخیص 0 و 1 منطقی در دو مدار یکی شود.

¹ Universal Synchronous and Asynchronous serial Receiver and Transmitter



شکل ۷-۳) بلوک دیاگرام واحد سریال

آزمایش ۷-۱

توسط سریال مانیتور LED روی برد آردوینو را روشن و خاموش نمائید.

آزمایش ۷-۲

این آزمایش توسط دو گروه باید صورت گیرد. بدین صورت که هر گروهی باید بتواند LED برد گروه دیگر را توسط سریال مانیتور خود به صورت زیر کنترل نماید.

- روشن شدن دائم.
- خاموش شدن دائم.
- روشن شدن به مدت معین
- از CRC^2 برای اطمینان از صحت دستور دریافتی استفاده شود. (می توانید از کتابخانه CRC استفاده کنید).
- برد دستور دهند باید از صحت اعمال دستور در برد مقصد مطمئن گردد.

² Cyclic Redundancy Check

پروژه آزمایشگاه

پس از انجام کامل ۷ آزمایش دانشجو باید یکی از پروژه های زیر را به اختیار انتخاب کرده و به صورت گروهی انجام دهد. جلسات باقیمانده از ترم را به صورت حضوری در آزمایشگاه به انجام پروژه خواهید پرداخت تا از راهنمایی های استاد بهره مند گردید. تحویل پروژه به صورت تکی خواهد بود تا ارزیابی دقیقی تری از دانشجویان صورت گیرد. یک طراحی اولیه توسط هر گروه باید صورت بگیرد تا بعد از تایید (و در صورت نیاز بازنگری) توسط استاد توسعه داده شده و تکمیل گردد.

لیست پروژه ها:

- ۱- طراحی سیستم کنترل شرایط محیطی گلخانه با دو سنسور و دو دستگاه
- ۲- طراحی مولد پالس با فرکانس و سیکل کاری قابل تنظیم با دو خروجی
- ۳- طراحی سیستم کنترل تردد با امکان ثبت ورود و خروج و گزارش دهی