استفاده از کدهای بدون نرخ در پیاده‌سازی شبکه‌های حسگر بی‌سیم LoRa

مقدادی
hamid.meghdadi@gmail.com

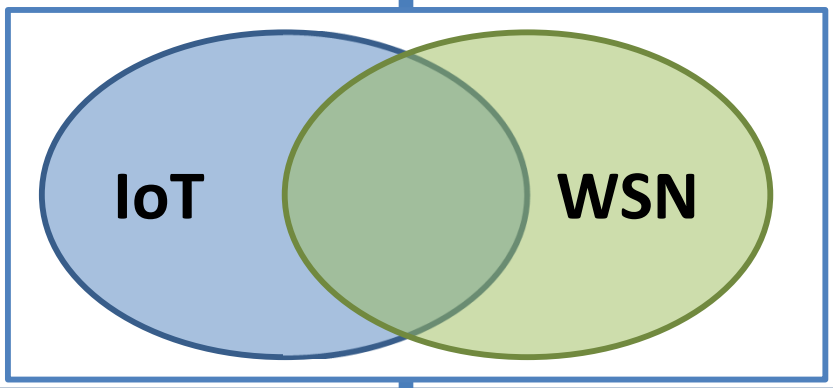
WSN vs IoT

شبکهٔ حسگر بی‌سیم

- شبکه:
  - اتصال اطلاعاتی
  - توزیع جغرافیایی
  - ایستگاه مرکزی
- حسگر:
  - پایش
  - پارامتر فیزیکی
- بی‌سیم
  - بستر مخابراتی بی‌سیم

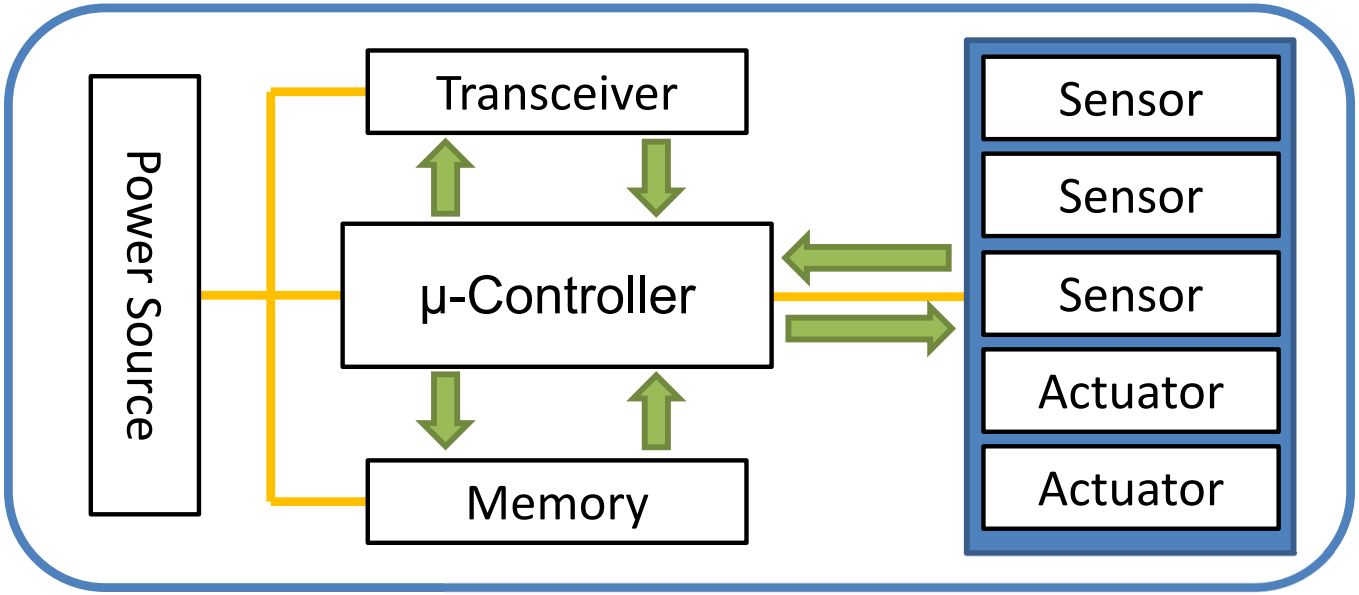
اینترنت اشیاء

- اینترنت:
  - Inter + Net(work)
  - شبکهٔ شبکه‌ها
  - حالت خاص: شبکهٔ جهانی اینترنت
- اشیاء:
  - فیزیکی: دنیای واقعی
    - ارتباط با شبکه و دنیای بیرون
    - حسگر (sensor)، عملگر (actuator)، ...
  - مجازی: دنیای اطلاعات
    - فقط ارتباط با شبکه
    - نرم‌افزار، صوت و تصویر، ...

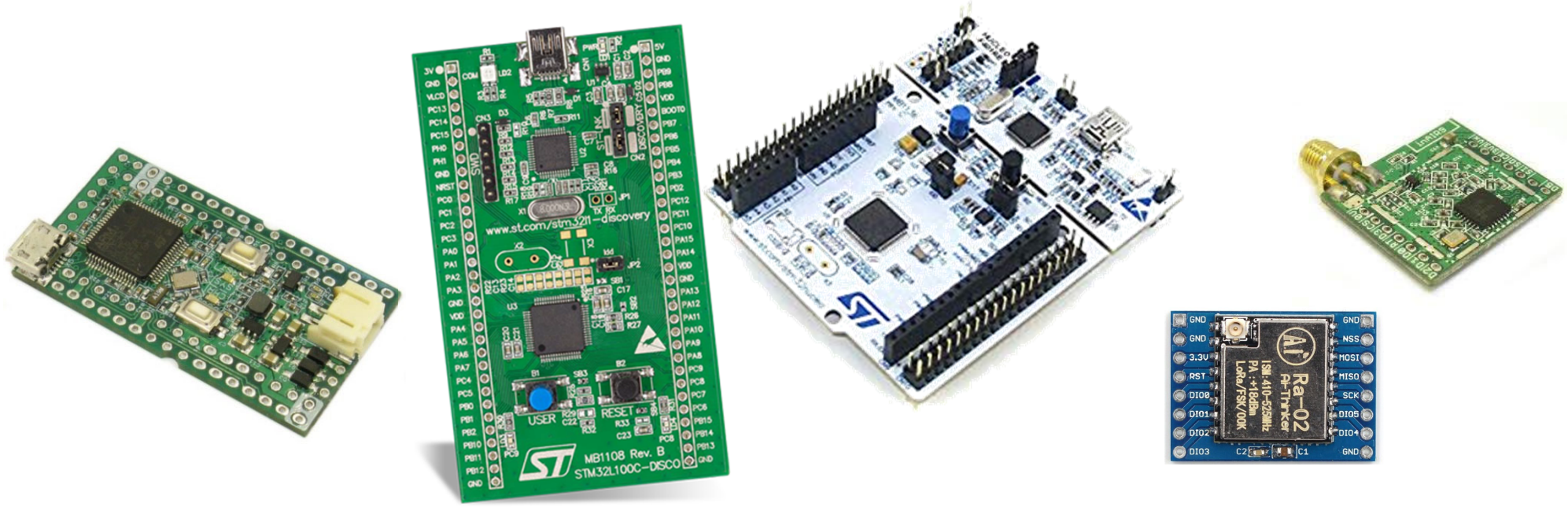


App
تجربهٔ کاربری متفاوت
08:16
19
°C
AUTO
ECO
libelium

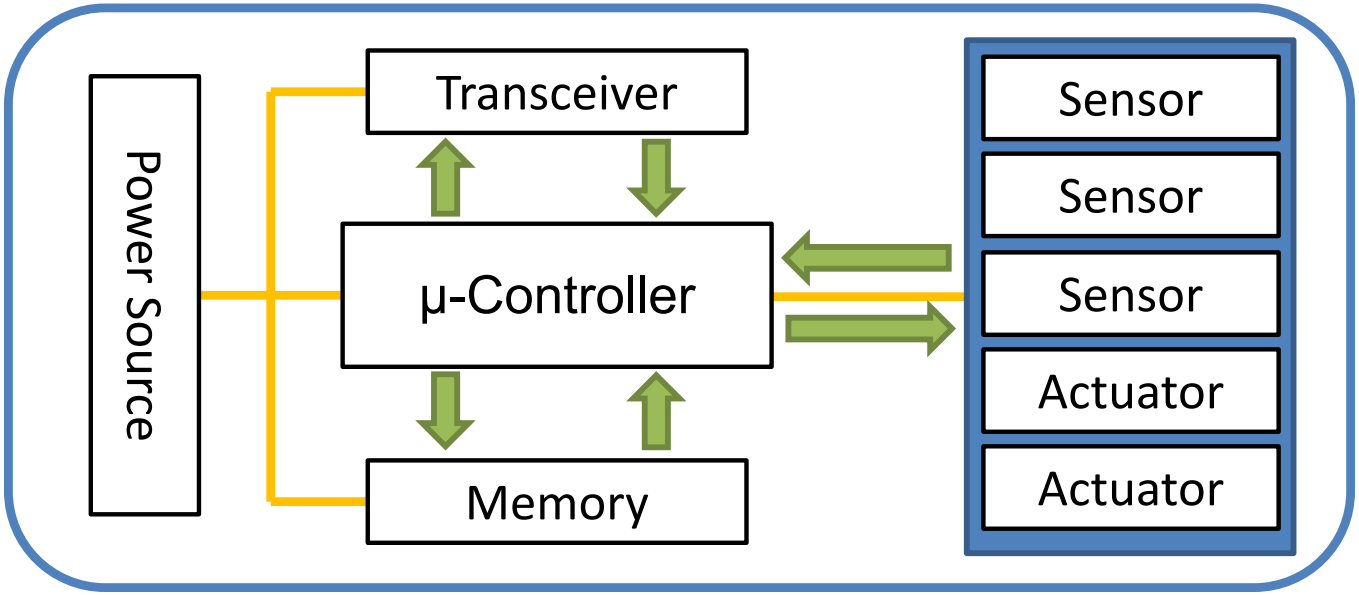
گره‌های حسگر



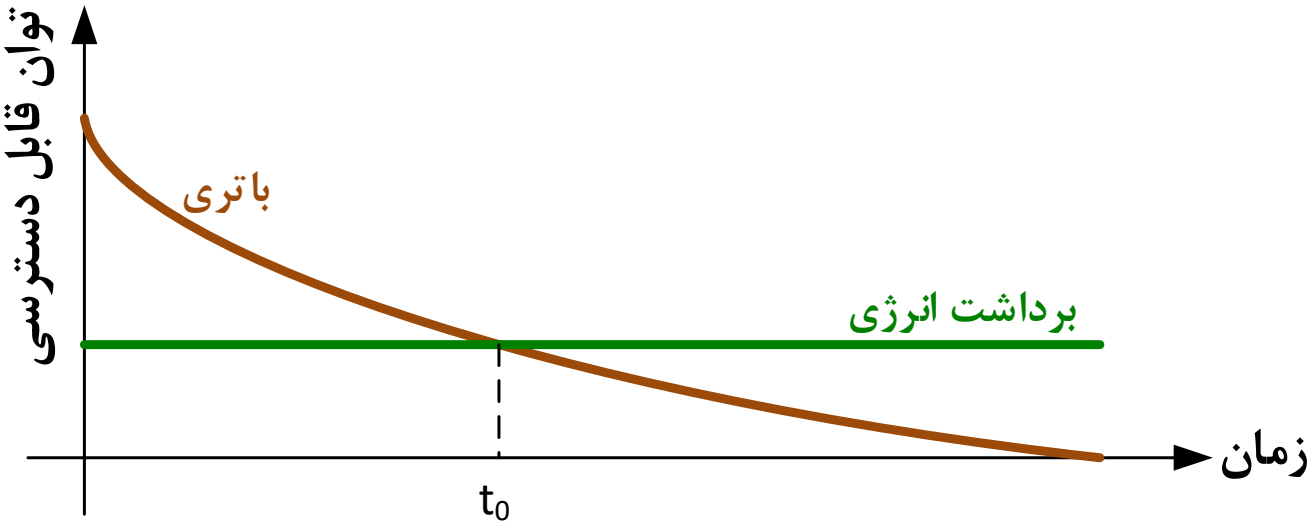
- فرستنده/گیرنده
- کنترل کننده
- تعدادی حسگر یا sensor
- (تعدادی عملگر یا actuator)
- (حافظه)
- منبع انرژی



گره‌های حسگر



- فرستنده/گیرنده
- کنترل کننده
- تعدادی حسگر یا sensor
- (تعدادی عملگر یا actuator)
- (حافظه)
- منبع انرژی
  - باتری
  - برداشت انرژی
  - ترکیبی

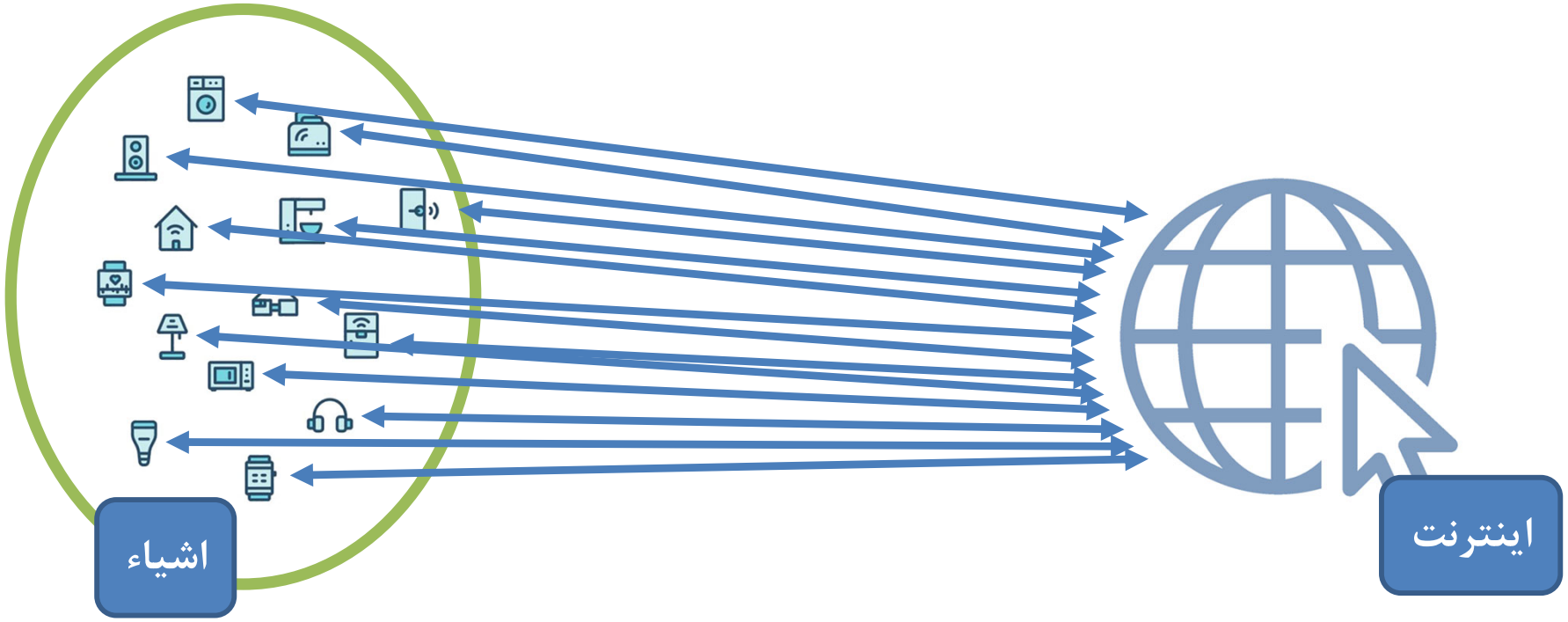


دروازه

- اتصال اشیاء به اینترنت
- اتصال حسگرها به پایگاه مرکزی

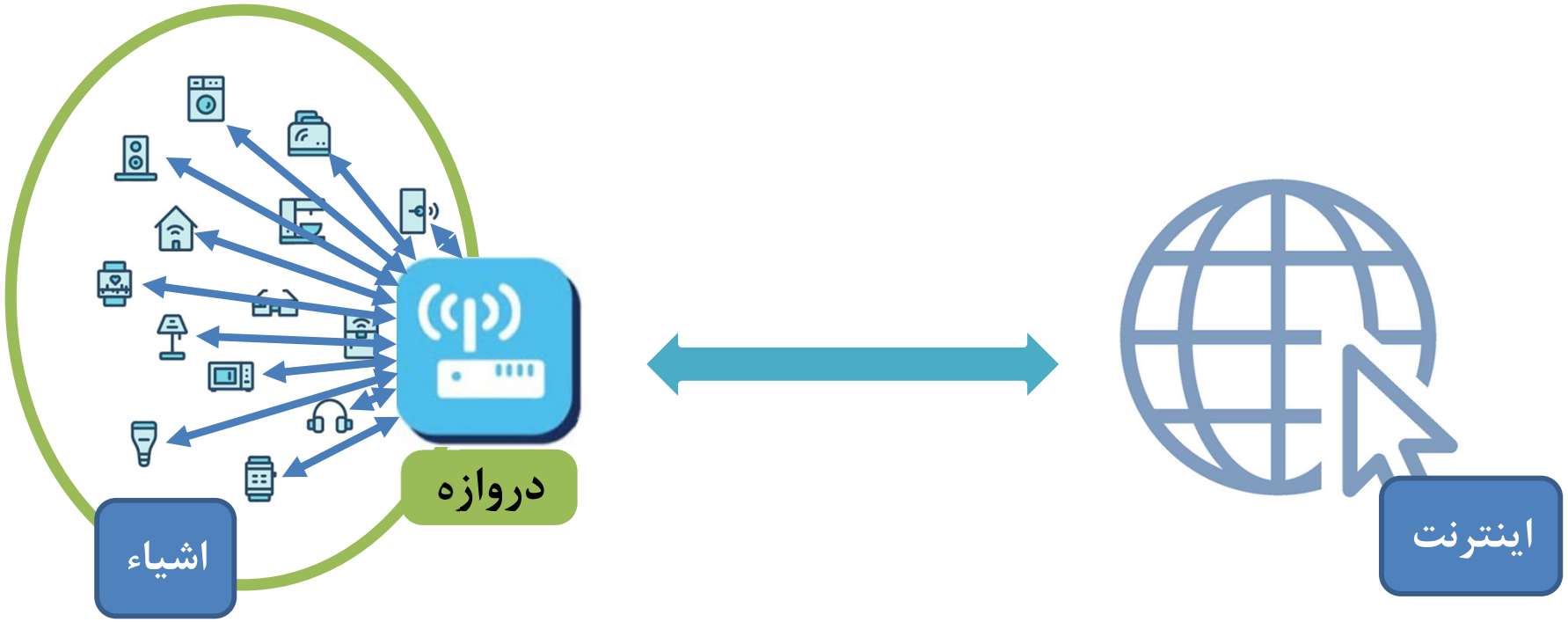
دروازه

- اتصال اشیاء به اینترنت
- اتصال حسگرها به پایگاه مرکزی



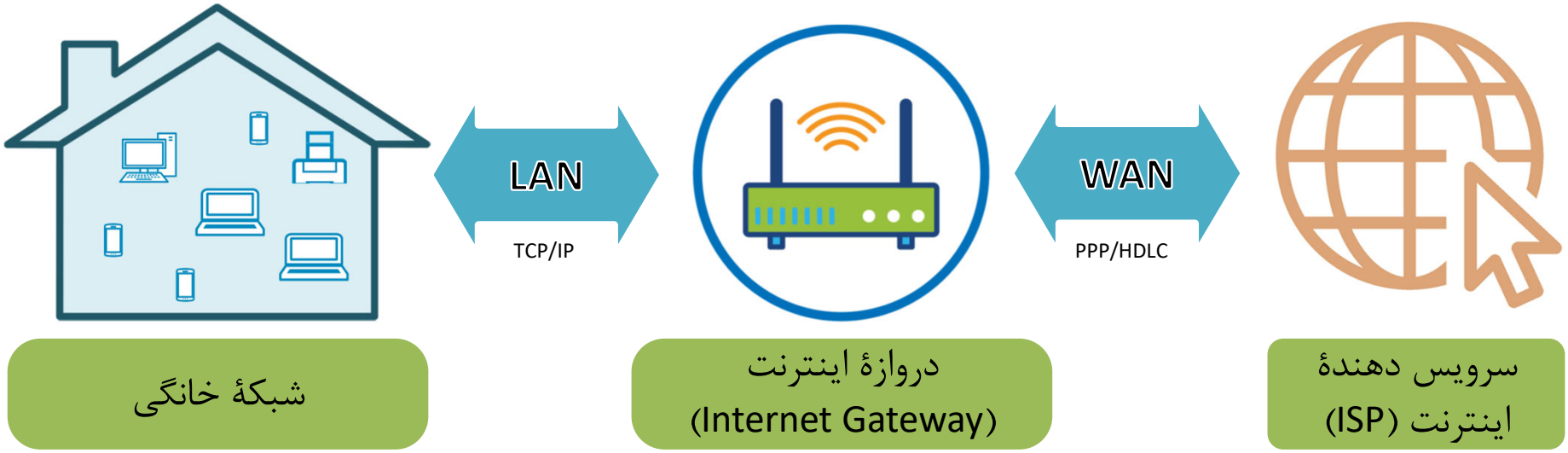
دروازه

- اتصال اشیاء به اینترنت
- اتصال حسگرها به پایگاه مرکزی



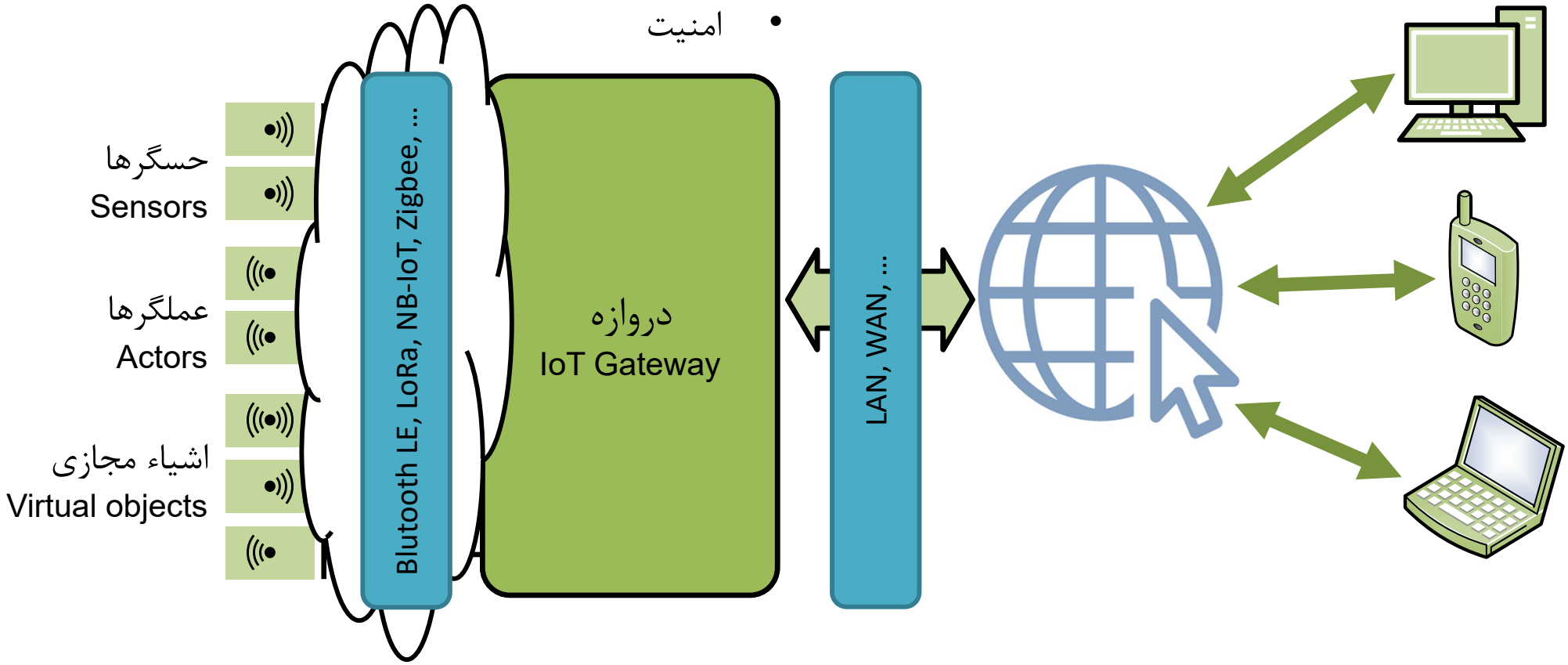
دروازه

- ارتباط بین دو تکنولوژی ارتباطی مختلف

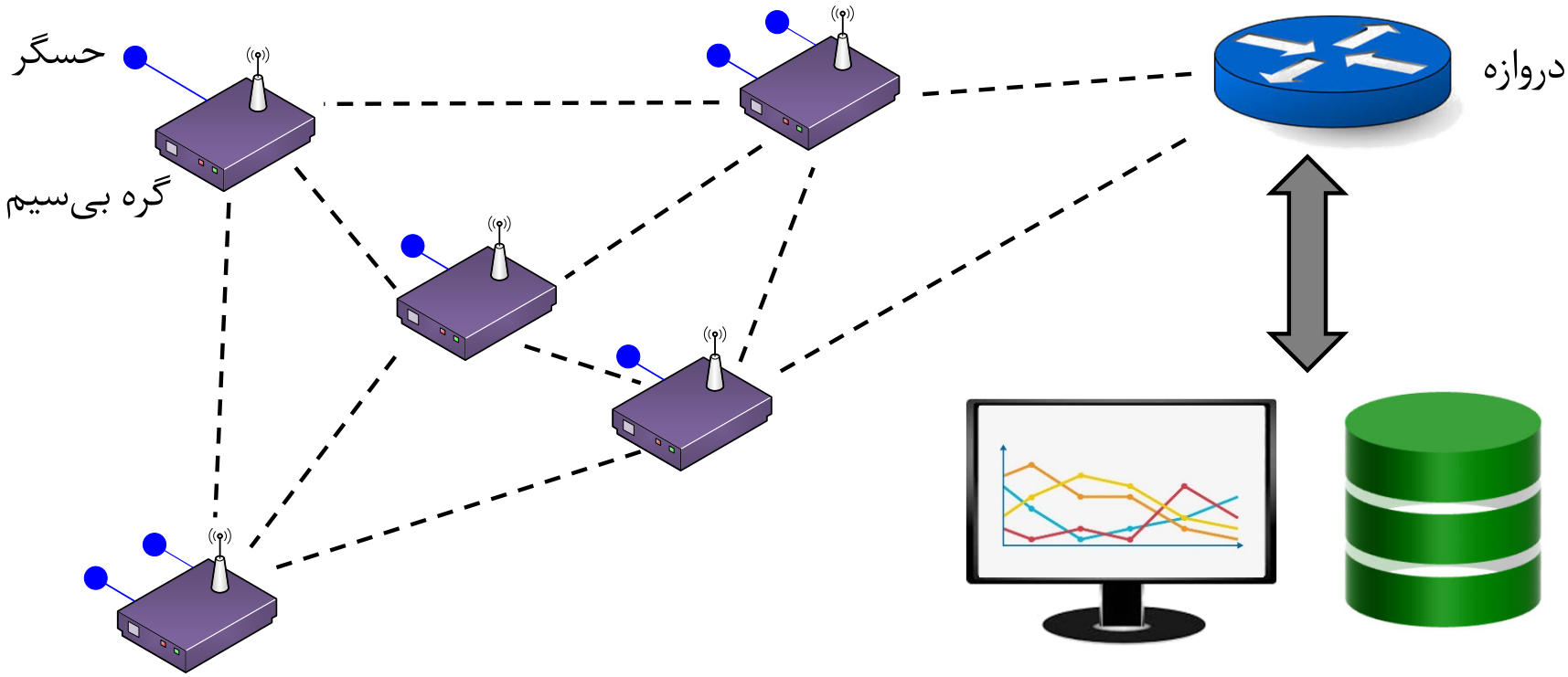


اینترنت اشیاء مبتنی بر شبکهٔ حسگری

- پل ارتباطی
- پردازش اطلاعات
- فشرده سازی اطلاعات
- امنیت



شبکهٔ حسگر بی‌سیم

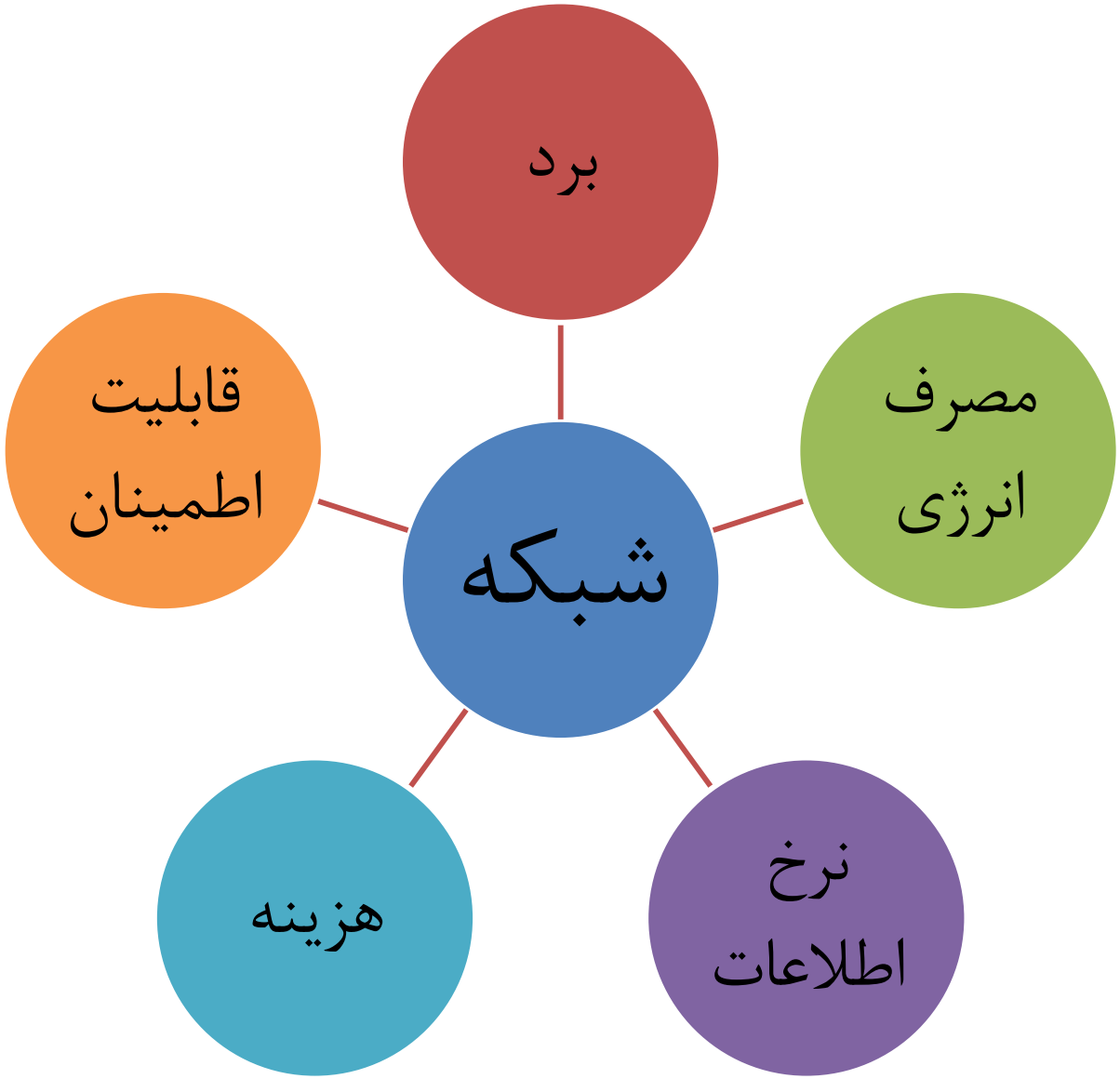


ایستگاه مرکزی

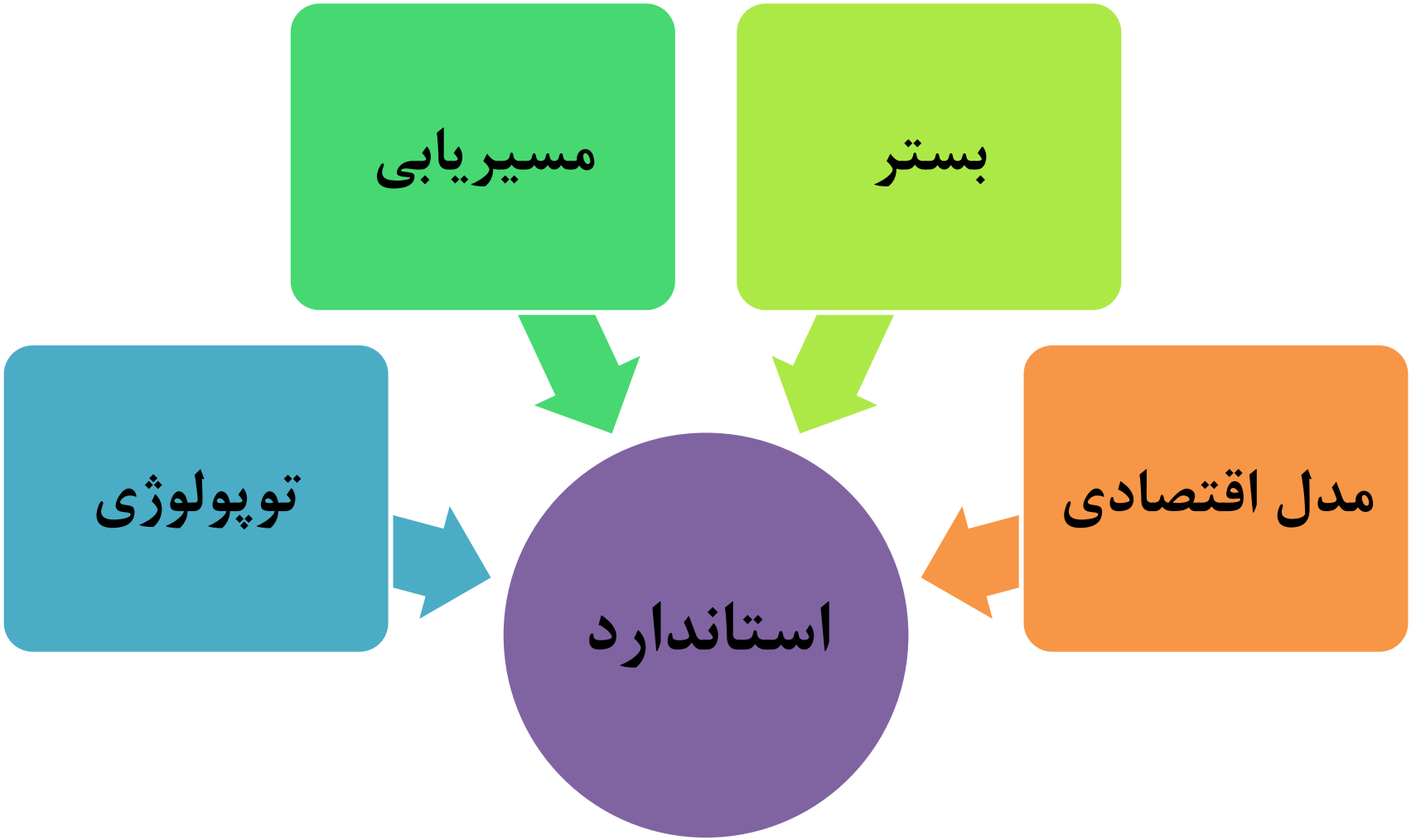
پایگاه داده
سرور اینترنتی
تنظیمات

گزارش‌گیری
هشدارها
کنترل مرکزی

انتخاب شبکهٔ مخابراتی (معیارها)

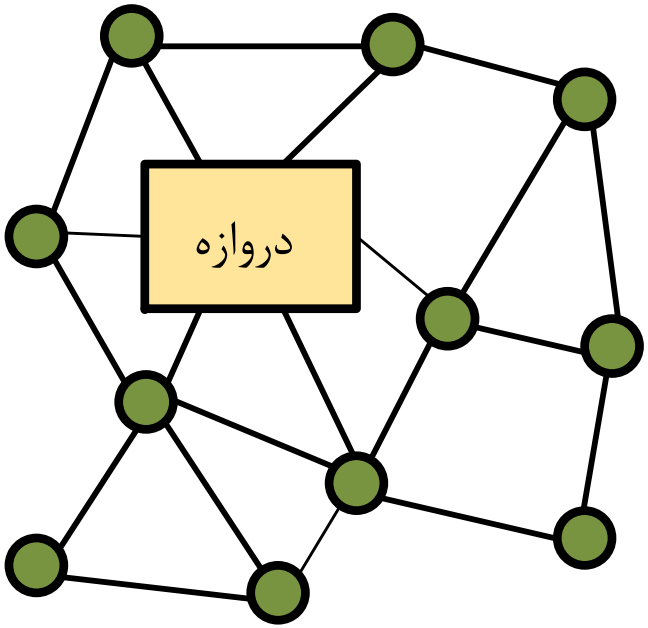


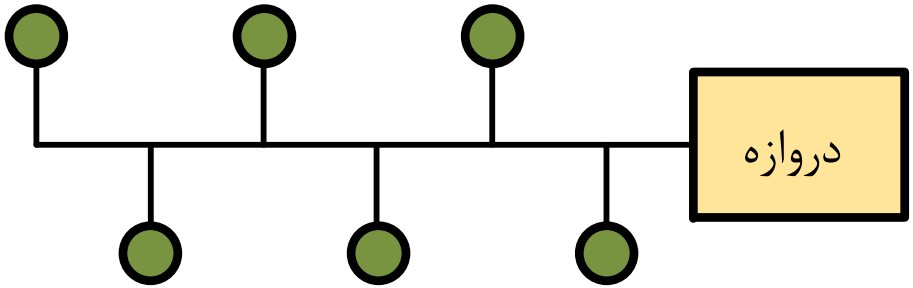
انتخاب شبکهٔ مخابراتی (پارامترها)

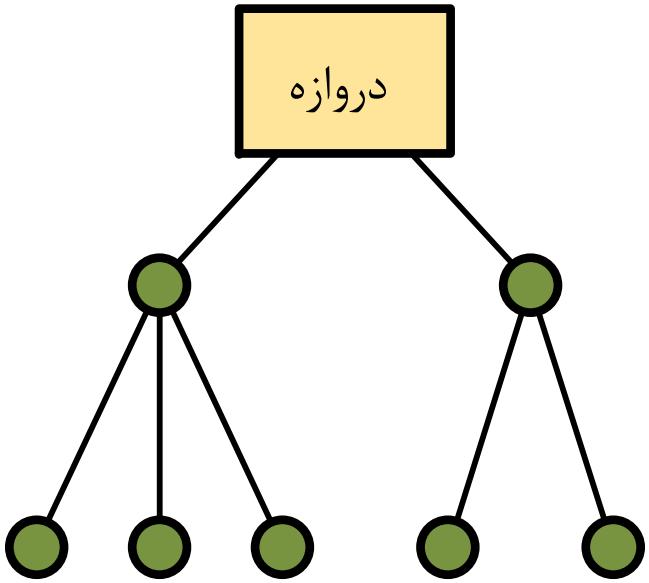


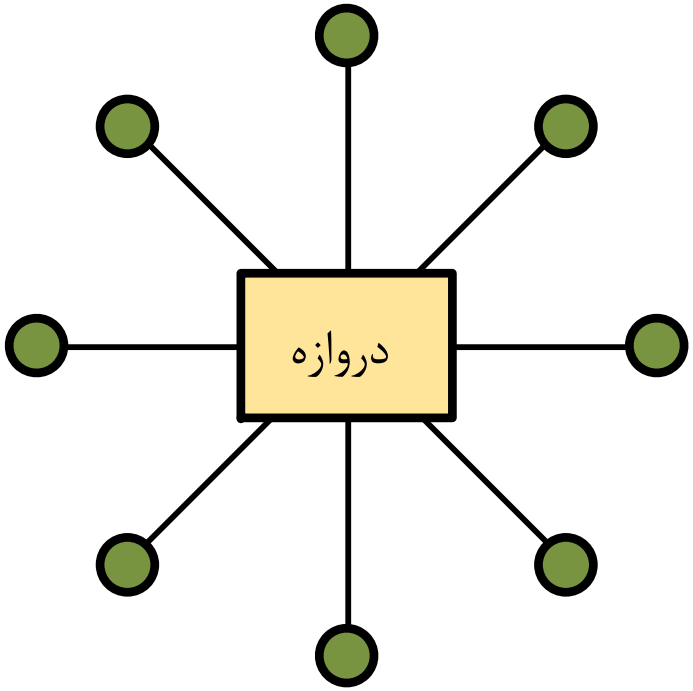
توپولوژی

- خطی
- درختی
- ستاره
- مش

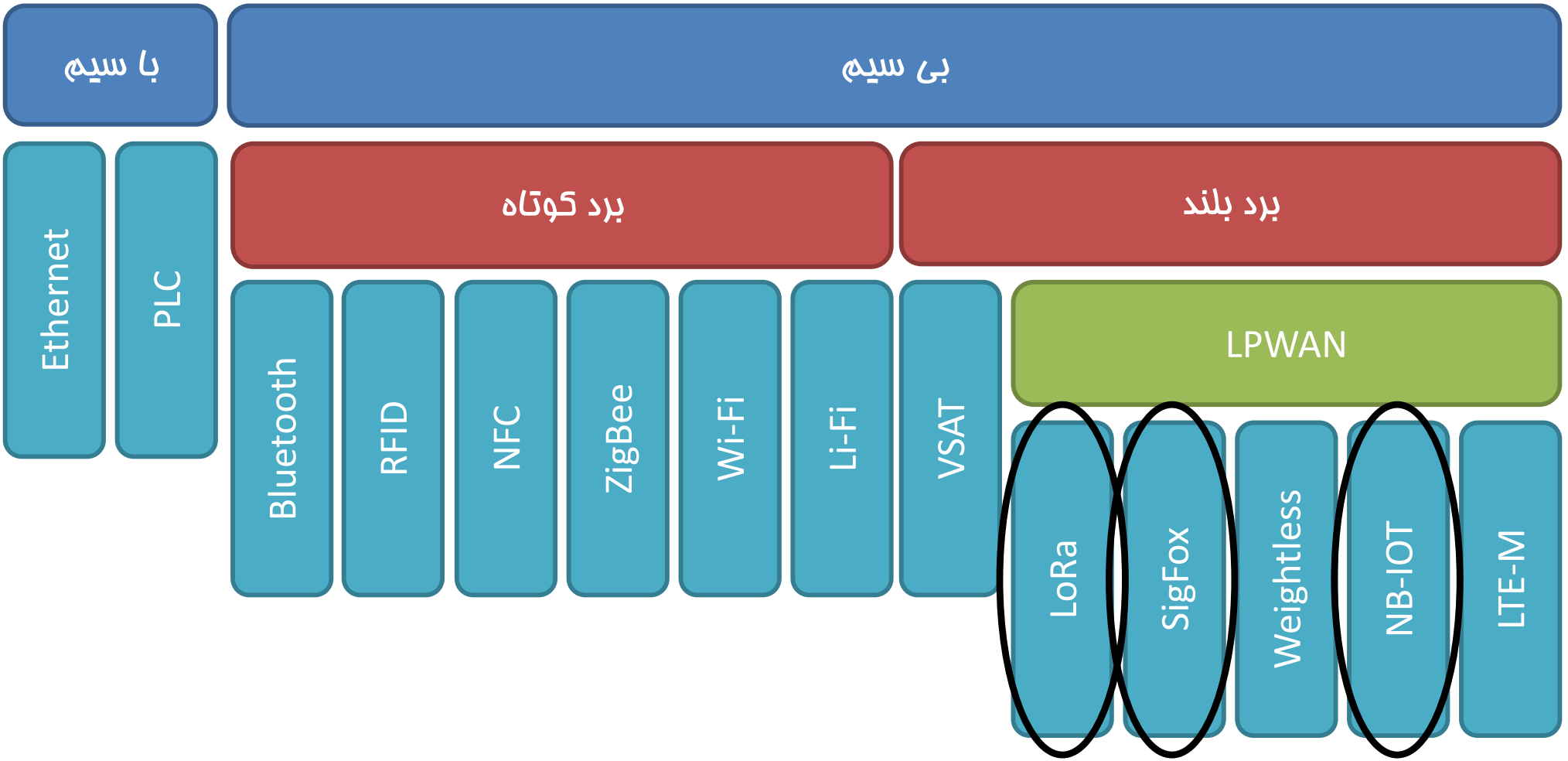


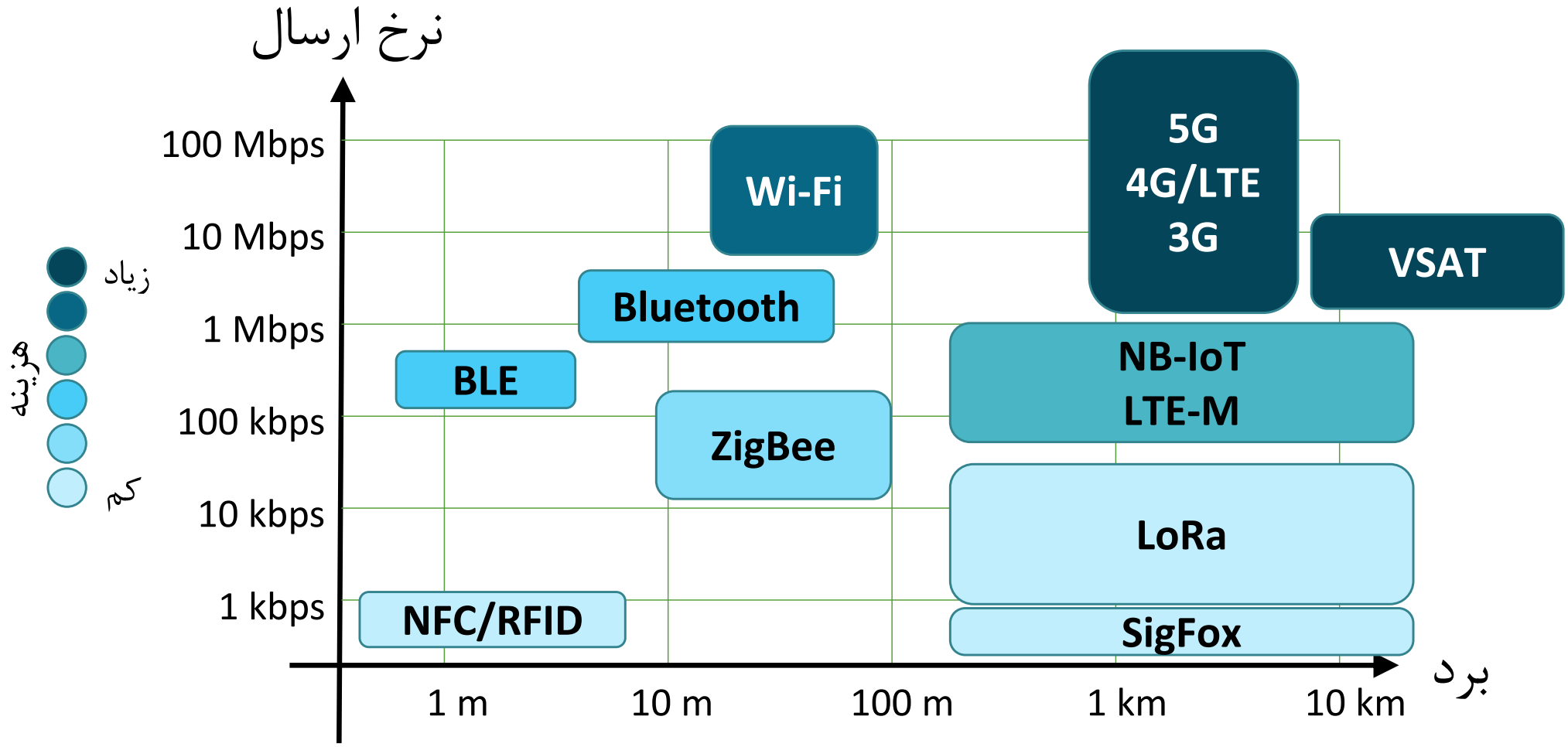


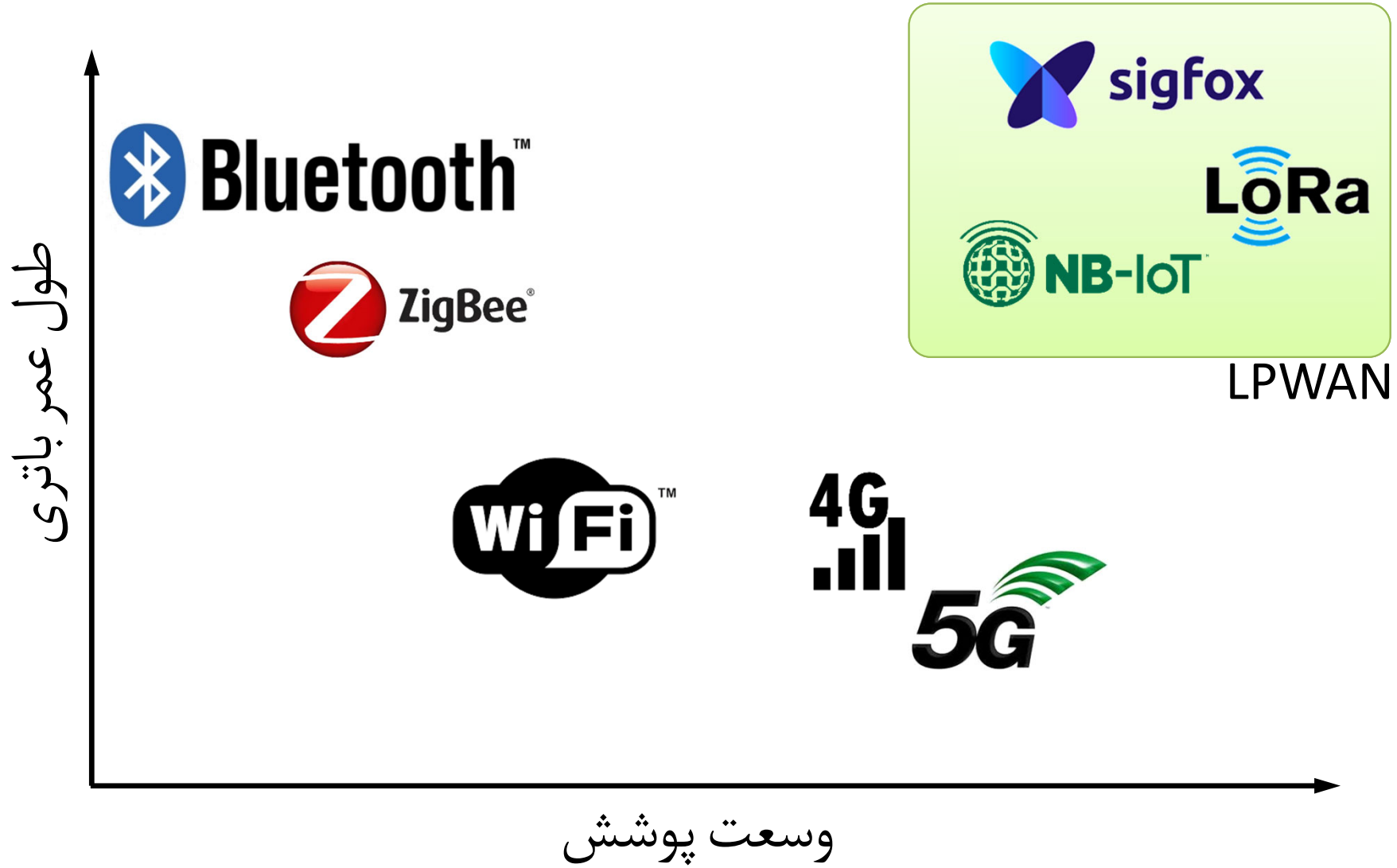




بستر ارتباطی

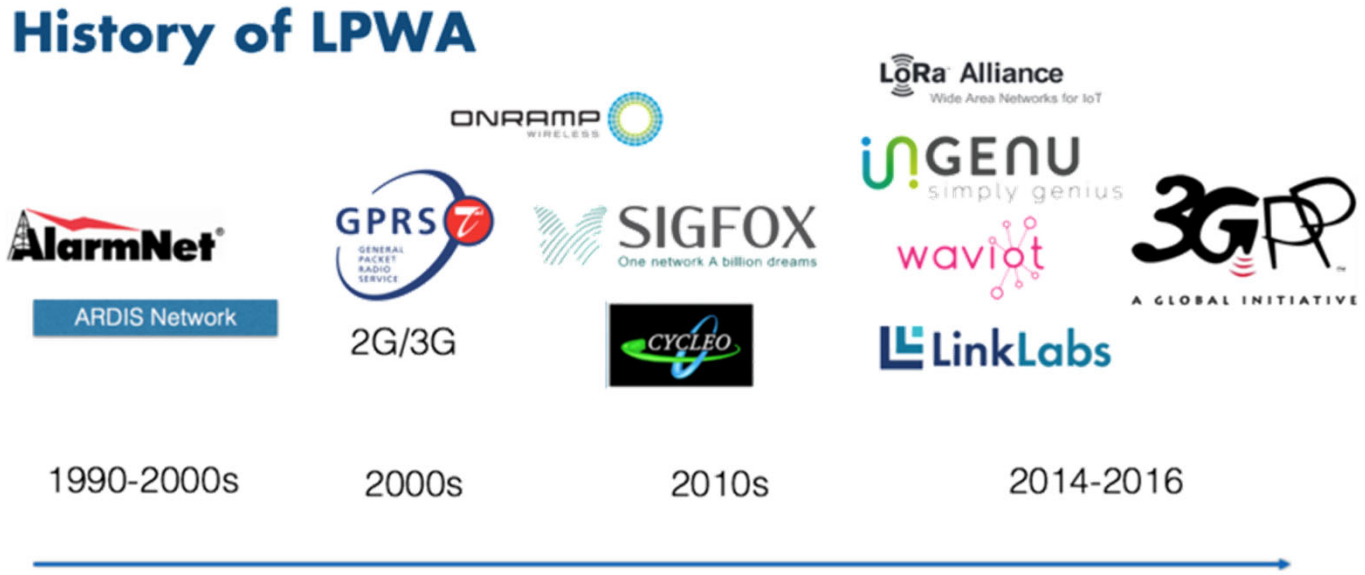



نرخ ارسال
100 Mbps
10 Mbps
1 Mbps
100 kbps
10 kbps
1 kbps
Wi-Fi
5G
4G/LTE
3G
VSAT
Bluetooth
BLE
NB-IoT
LTE-M
ZigBee
LoRa
NFC/RFID
SigFox
1 m
10 m
100 m
1 km
10 km
برد
هزینه
زیاد
کم


sigfox
LoRa
NB-IoT
LPWAN
Bluetooth™
ZigBee®
Wi Fi™
4G
5G
طول عمر باتری
وسعت پوشش

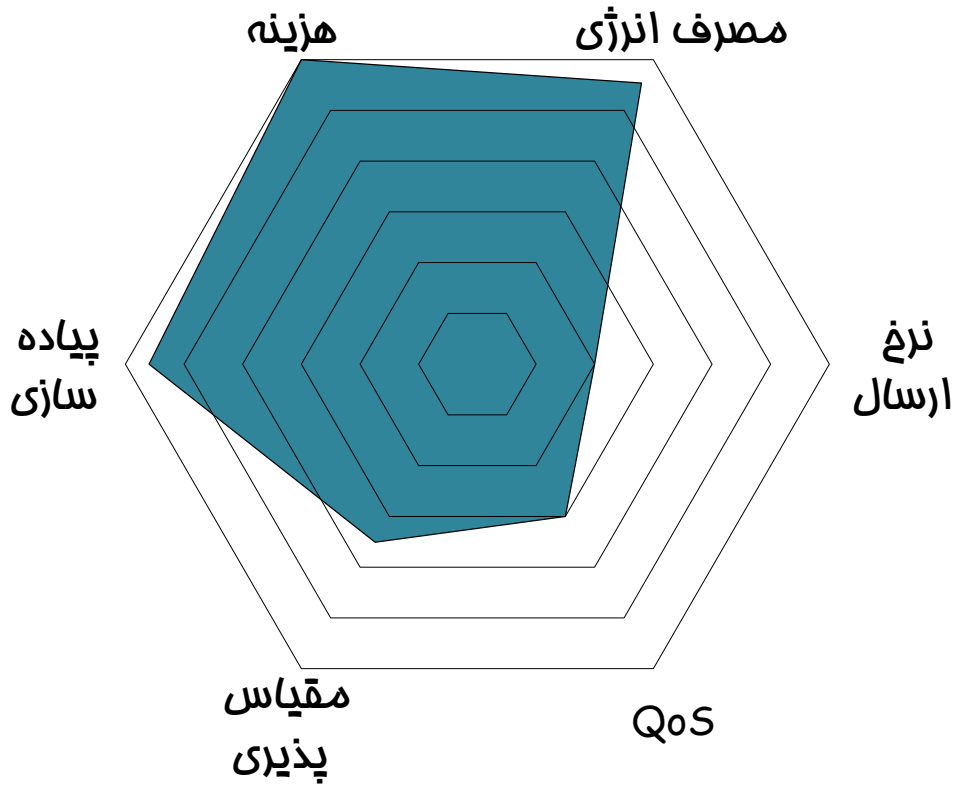
LPWAN

- Low-Power Wide-Area Network
- تکنولوژی:
  - استانداردهای 3GPP:
    - LTE-M
    - NB-IOT
    - EC-GSM
  - گروه‌های صنعتی:
    - LoRa
    - Weightless
  - راه‌حل‌های اختصاصی:
    - Sigfox
    - Ingenu



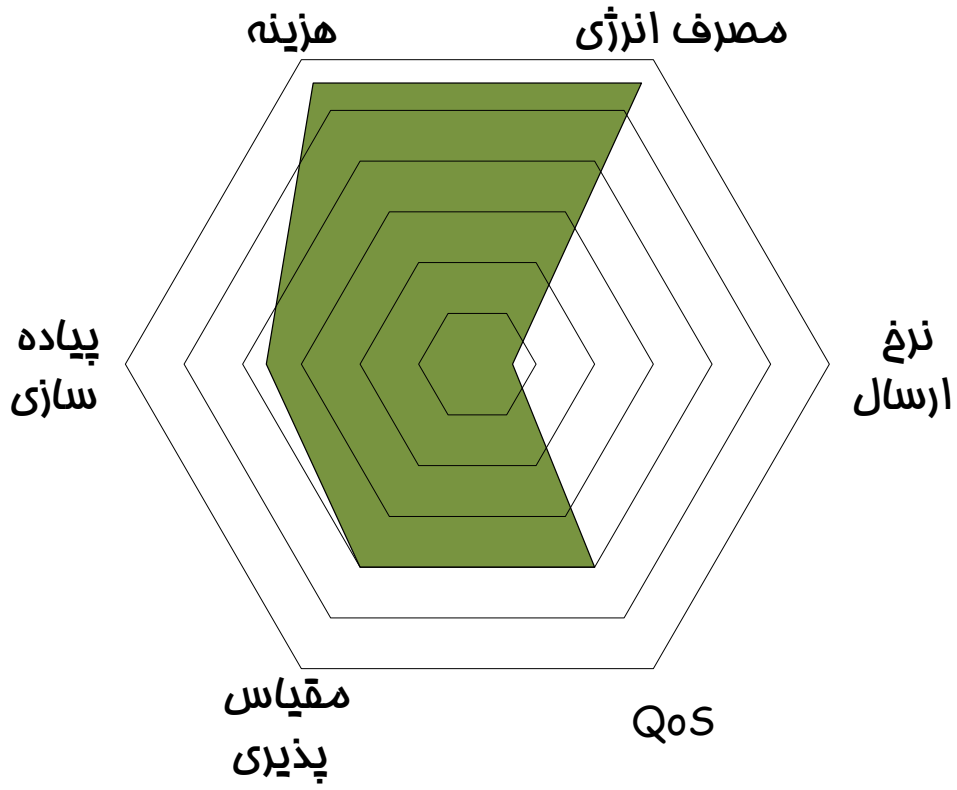
مقایسه

- LoRa



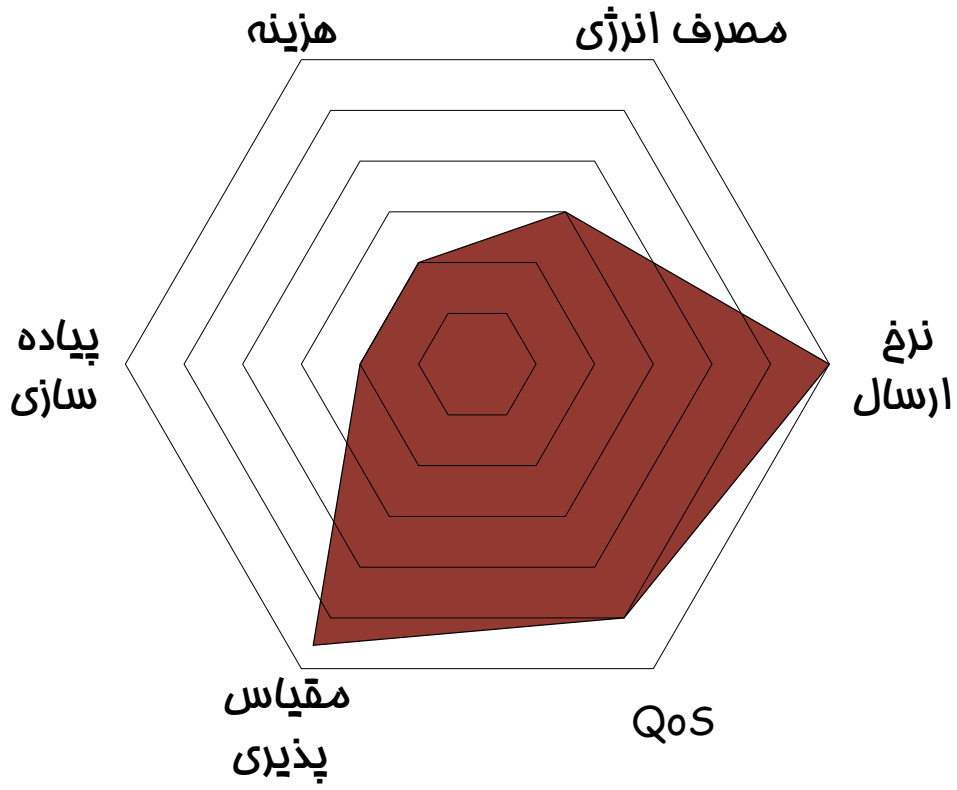
مقایسه

- SigFox



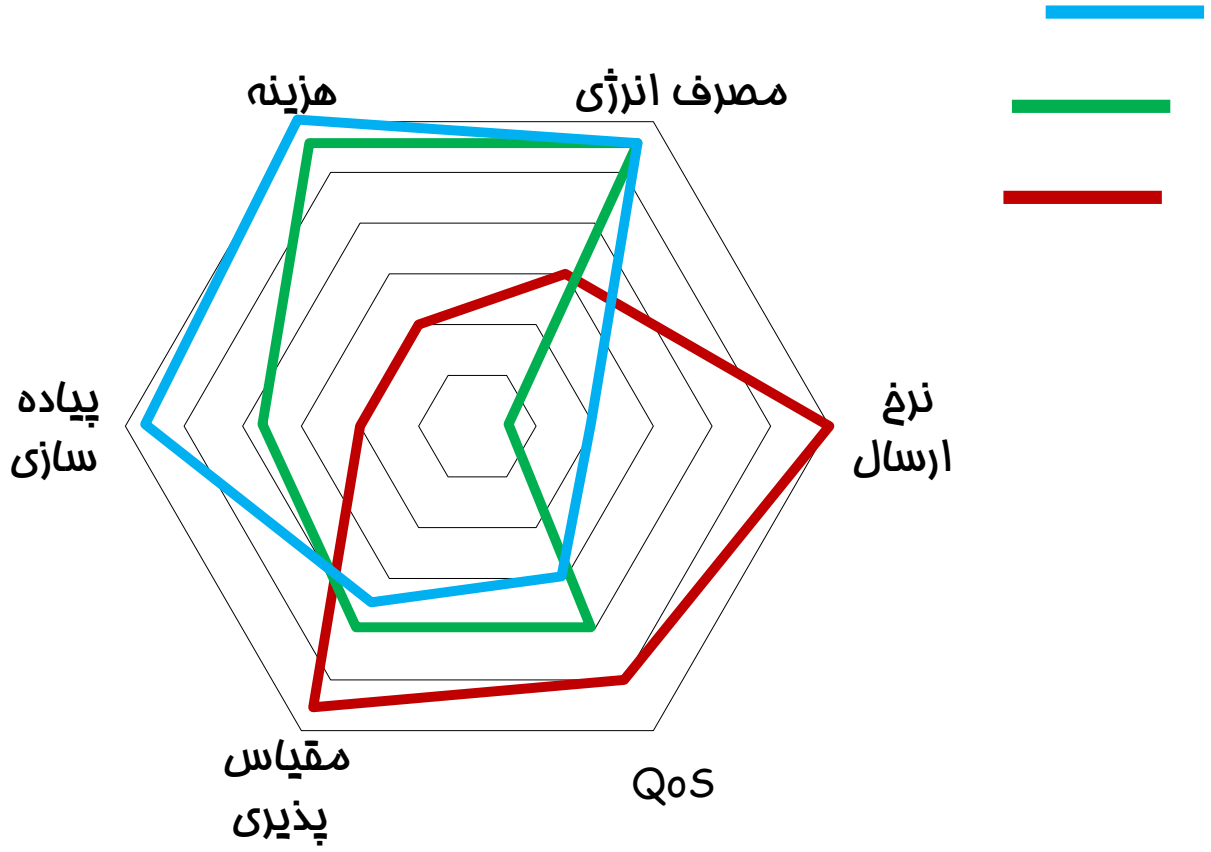
مقایسه

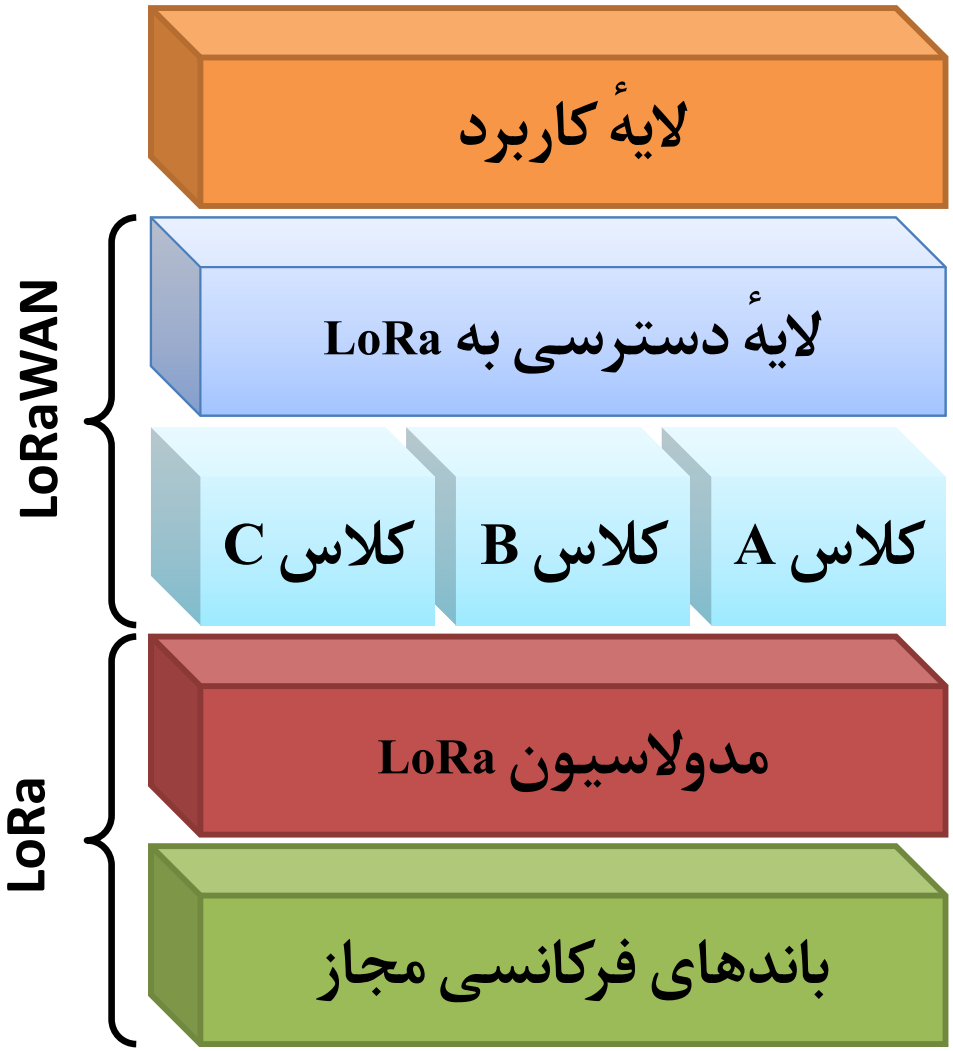
- NB-IoT



مقایسه

- LoRa
- SigFox
- NB-IoT



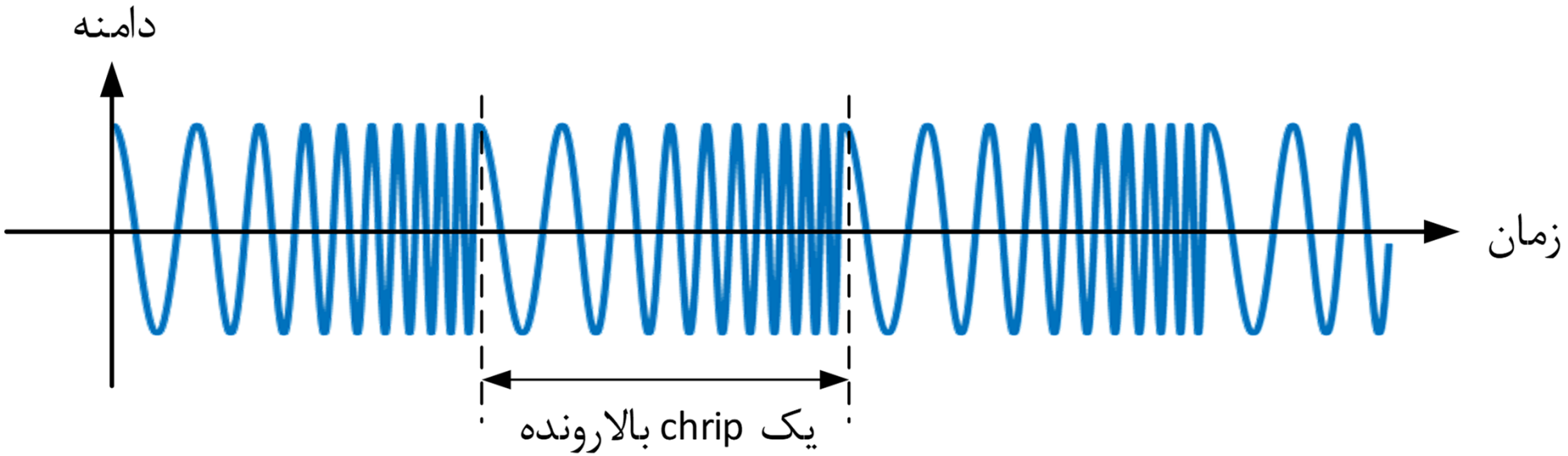

لایهٔ کاربرد
LoRaWAN
لایهٔ دسترسی به LoRa
کلاس A
کلاس B
کلاس C
LoRa
مدولاسیون LoRa
باندهای فرکانسی مجاز

معرفی LoRa

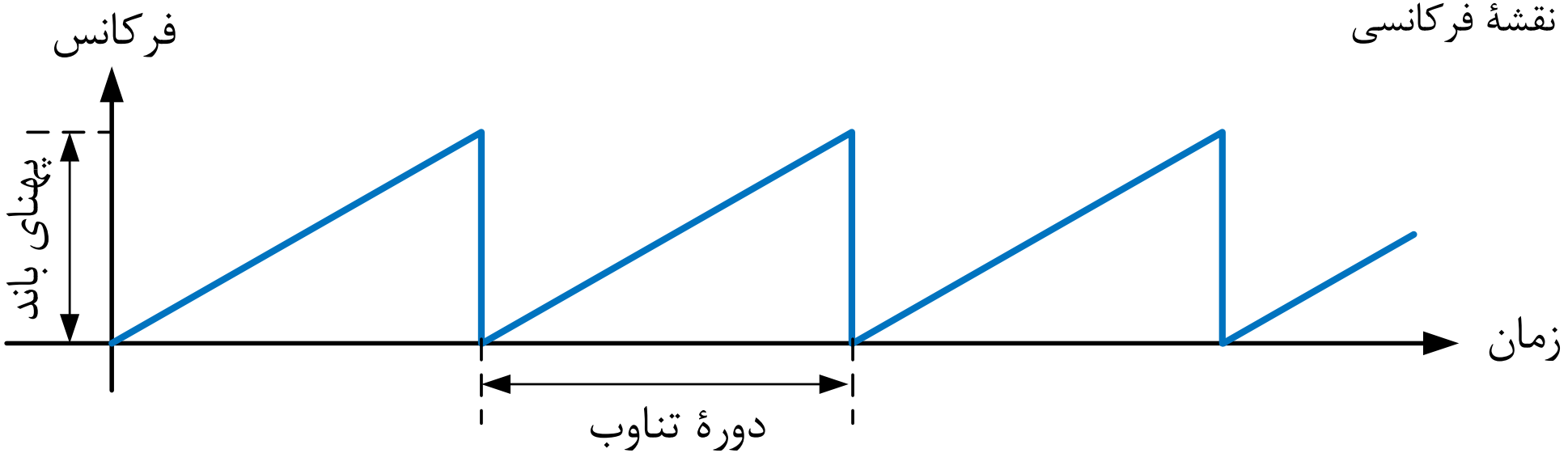
- Long Range
- توسط شرکت cycleo (بعداً Semtech) در سال ۲۰۱۰
- اکنون در حدود ۳۰۰ میلیون دستگاه به کار رفته (مارس ۲۰۲۳)
- مدولاسیون طیف گسترده
- استفاده از باندهای فرکانسی ISM
- مصرف انرژی پایین
  - در حالت ارسال: حدود 300 mW
  - در حالت دریافت: حدود 30 mW
  - در حالت انتظار: حدود 5µW
- برد بسیار بلند: حدود ۱۰ کیلومتر
- نرخ ارسال پایین: حداکثر تا 30 kbps
- کم هزینه

مدولاسیون LoRa

- سیگنال chirp یا چهچهه

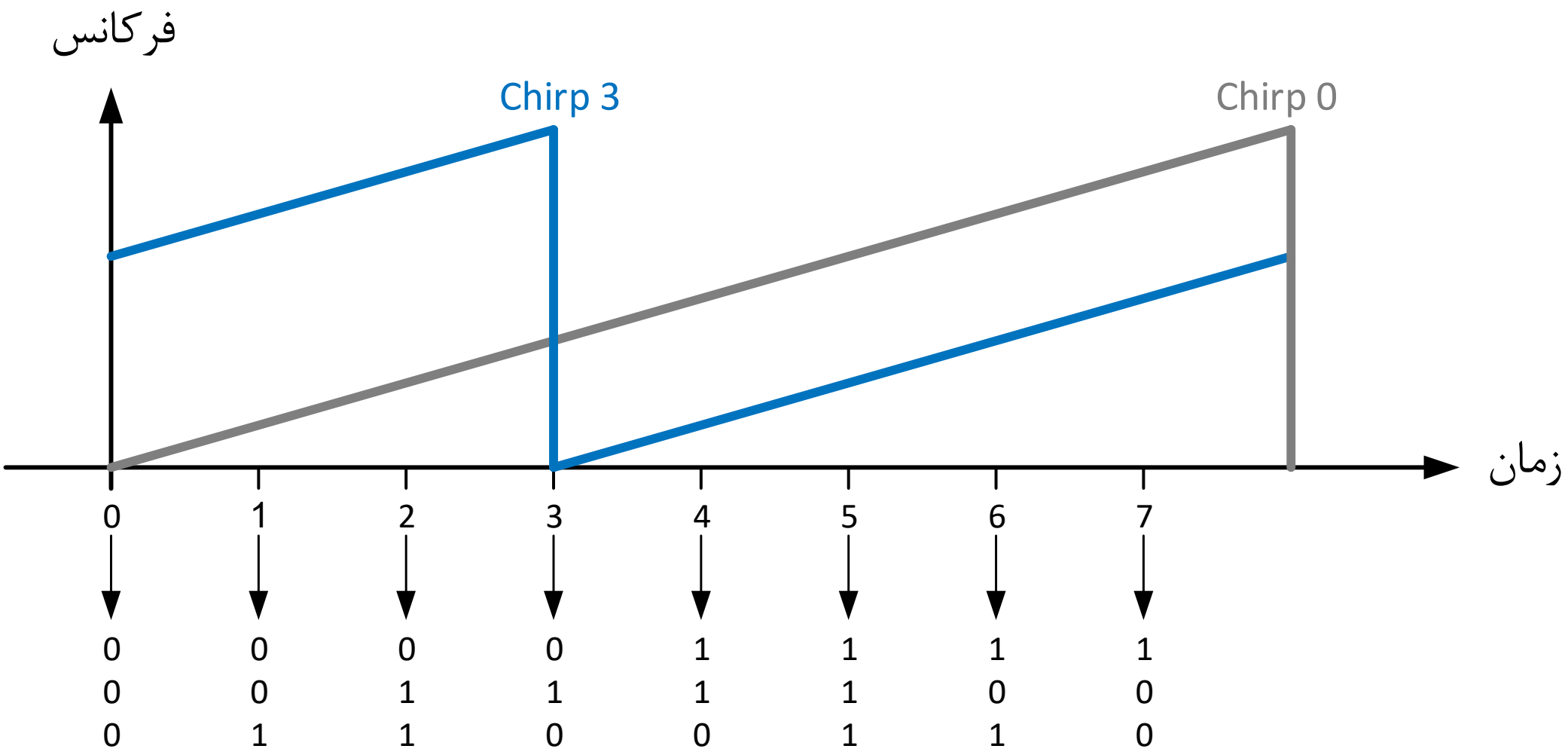


- نقشهٔ فرکانسی



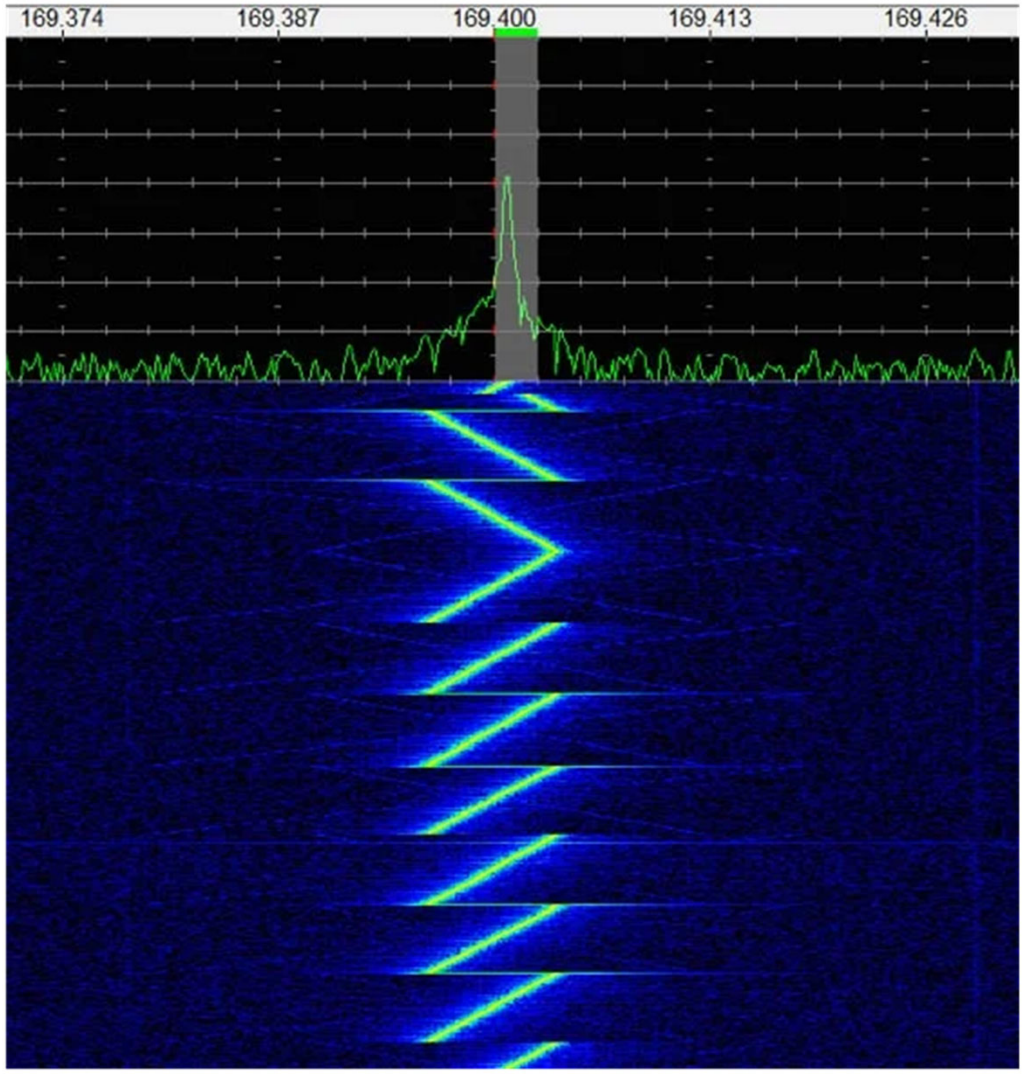
مدولاسیون LoRa

- فرستادن اطلاعات ⟸ chirp جابه‌جا شده

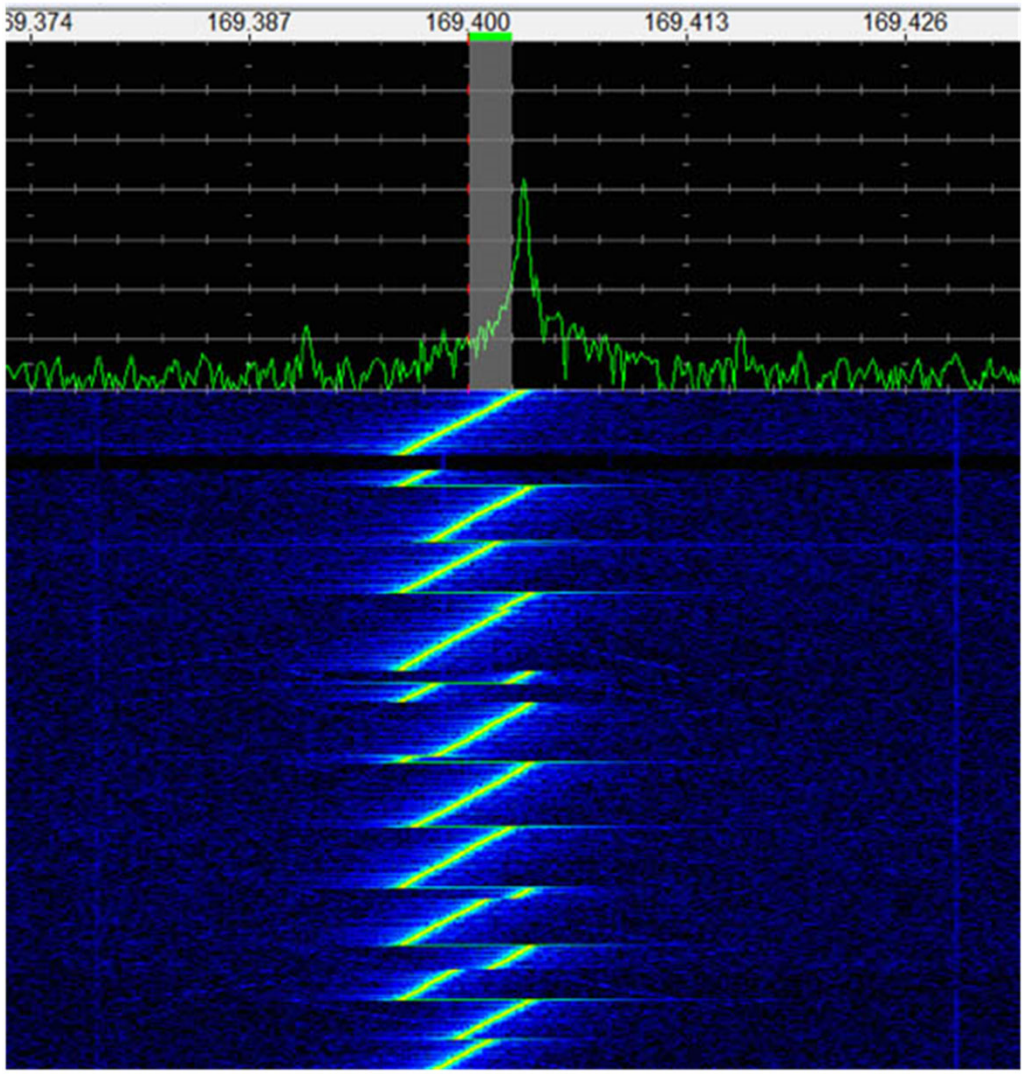


مدولاسیون LoRa

- آغاز بسته (مدوله نشده)

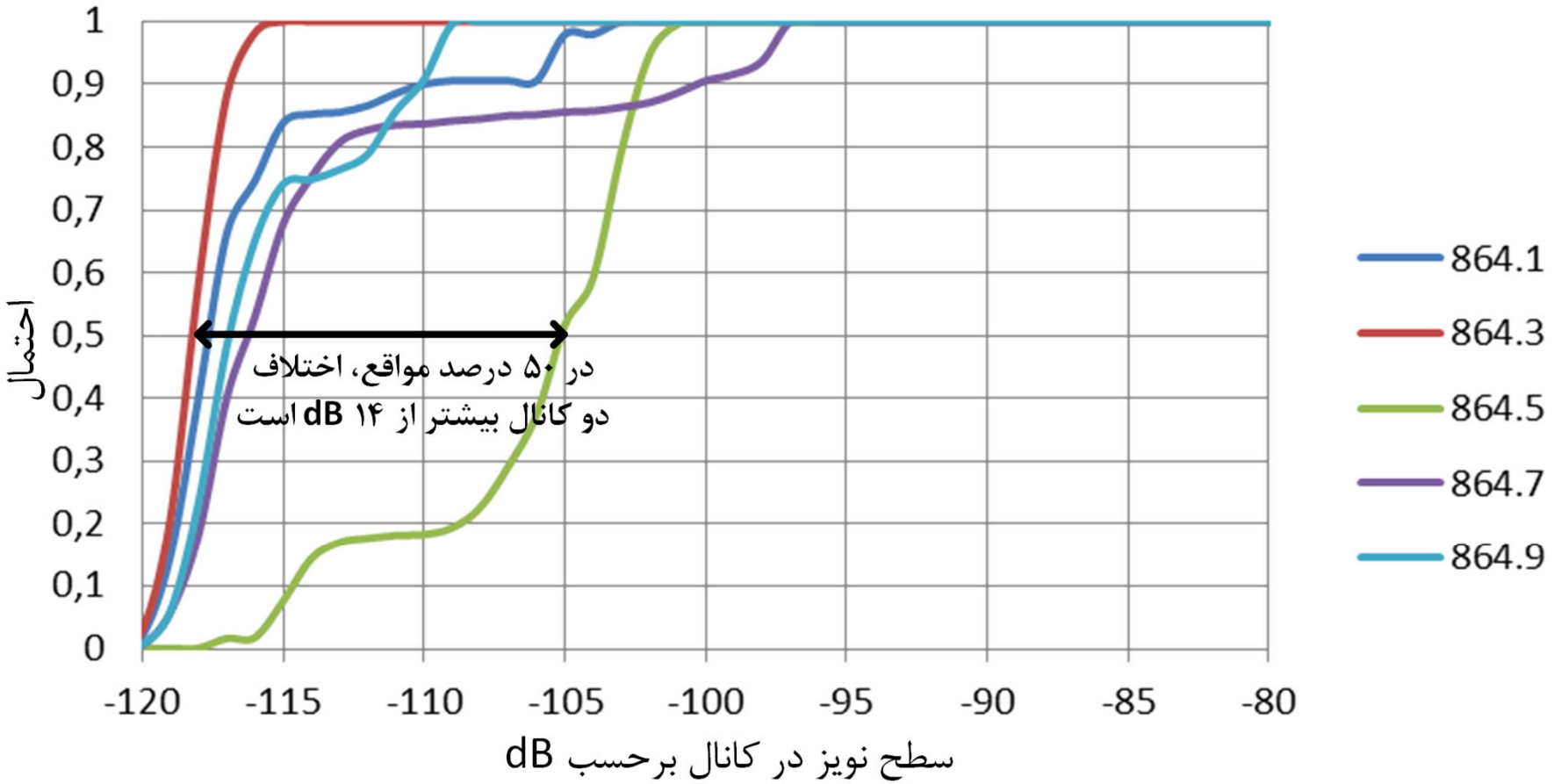


- اطلاعات مدوله شده



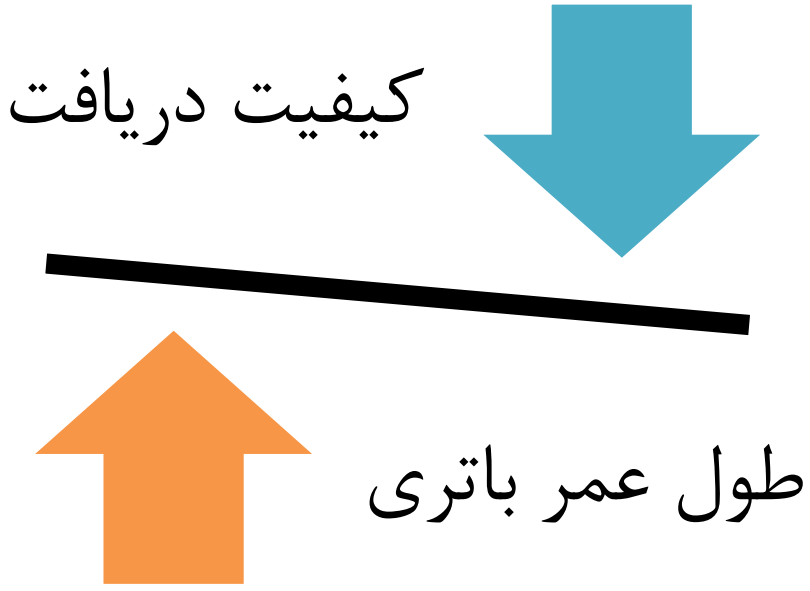
پارمترها

- فرکانس مرکزی
  - باند ISM
- توان فرستنده
  - در حالت عادی حداکثر 14dBm



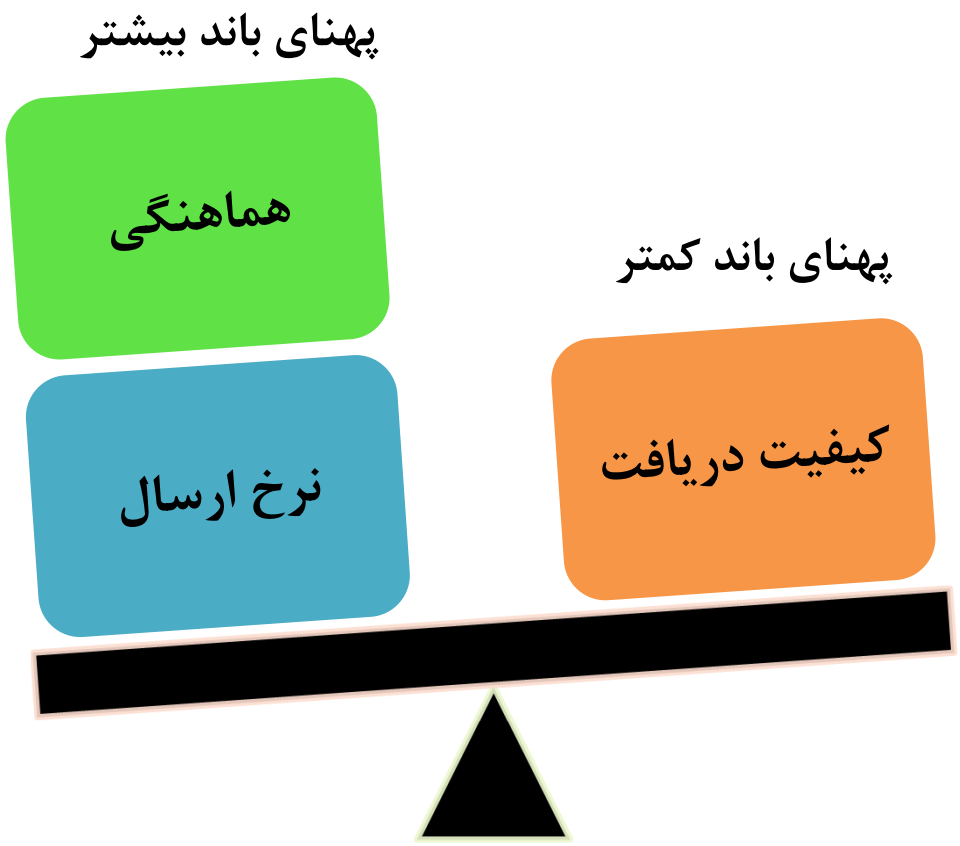
پارمترها

- فرکانس مرکزی
  - باند ISM
- توان فرستنده
  - در حالت عادی حداکثر 14dBm



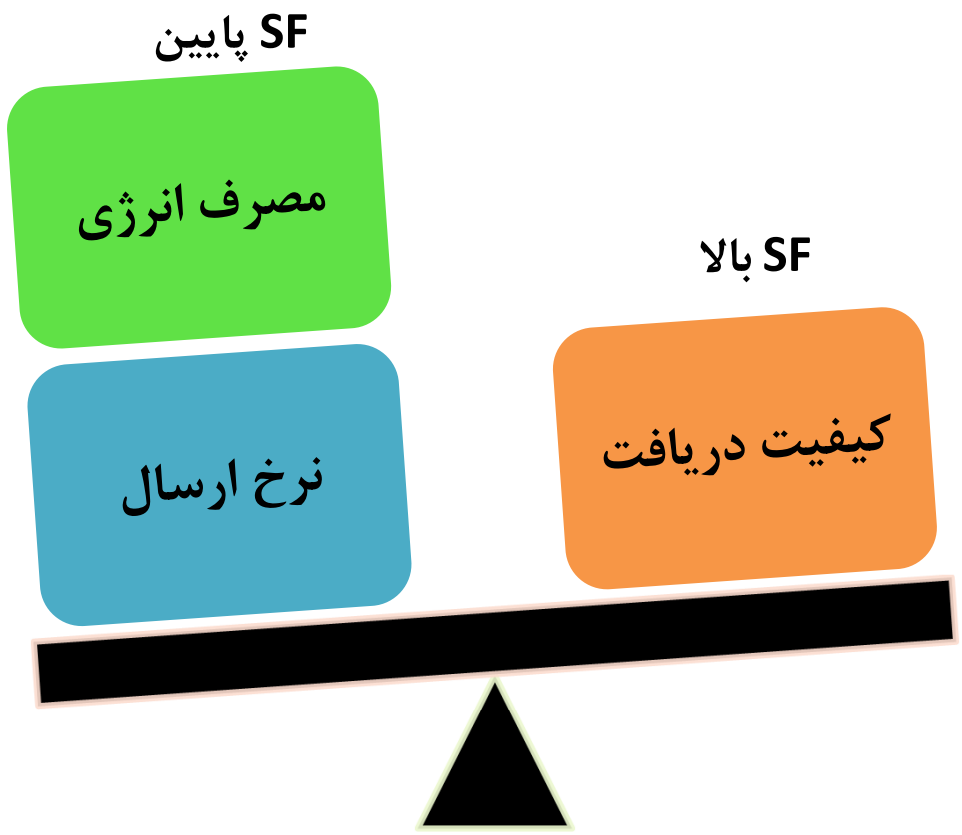
پارمترها

- فرکانس مرکزی
  - باند ISM
- توان فرستنده
  - در حالت عادی حداکثر 14dBm
- پهنای باند
  - از 7.8 kHz تا 500 kHz



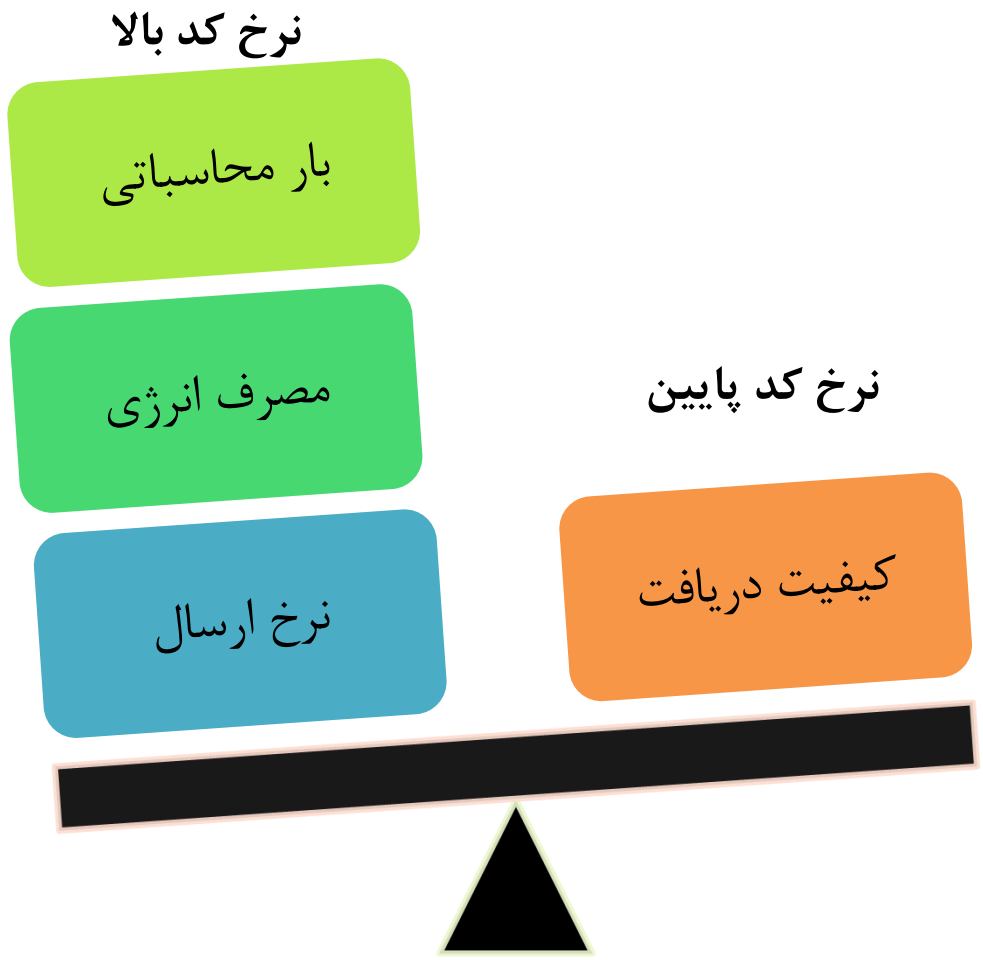
پارمترها

- فرکانس مرکزی
  - باند ISM
- توان فرستنده
  - در حالت عادی حداکثر 14dBm
- پهنای باند
  - از 7.8 kHz تا 500 kHz
- ضریب گسترش
  - از 7 تا 12

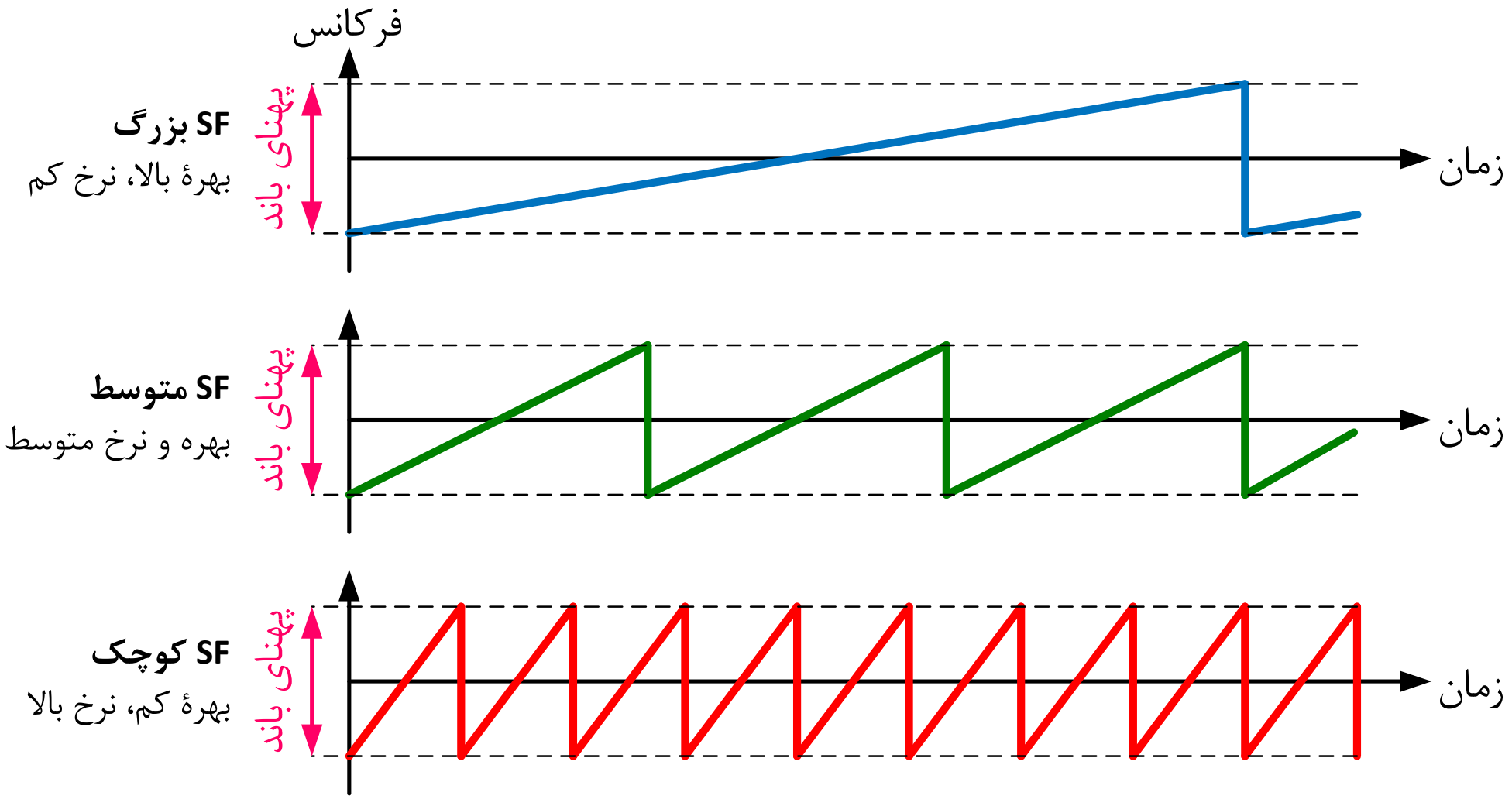


پارمترها

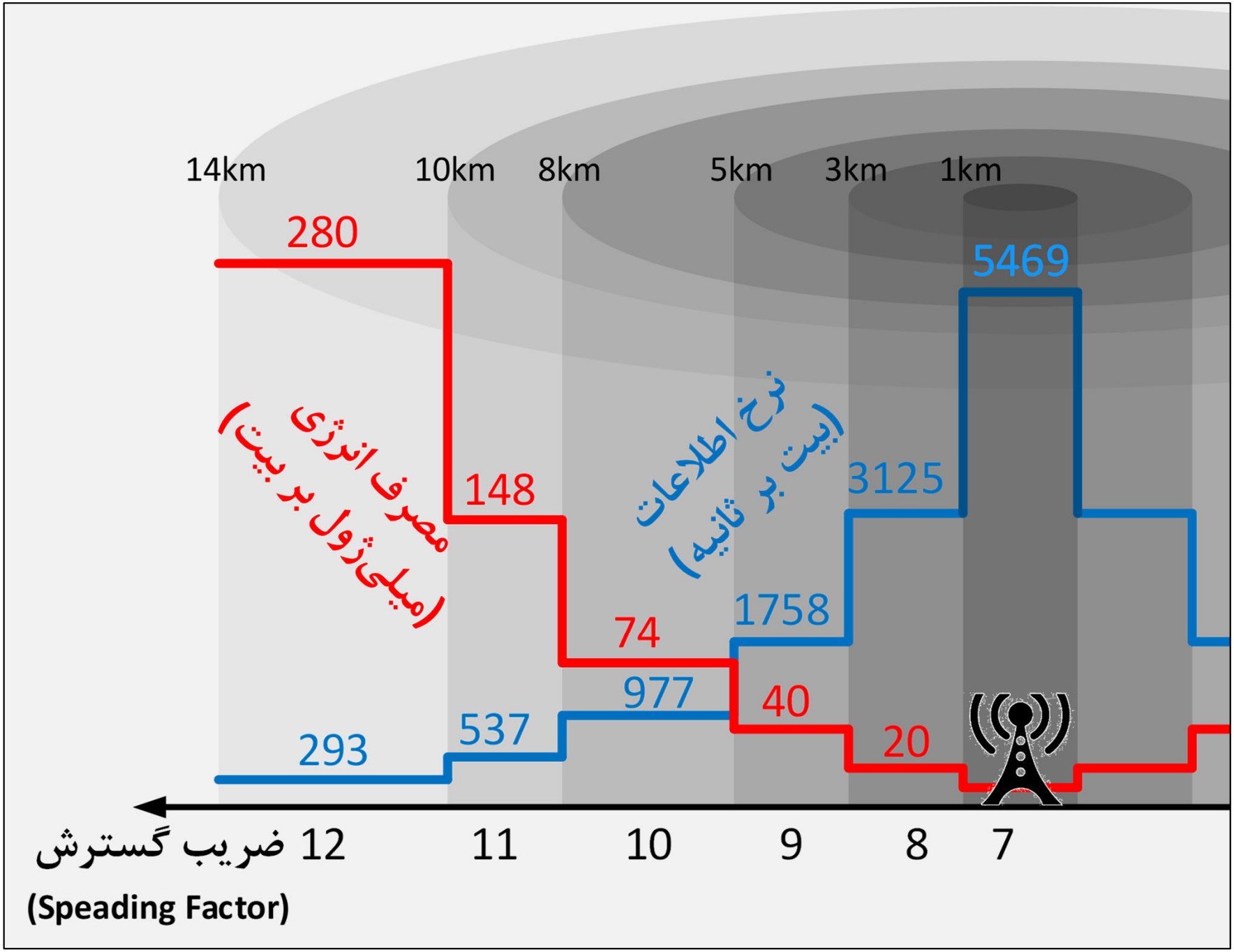
- فرکانس مرکزی
  - باند ISM
- توان فرستنده
  - در حالت عادی حداکثر 14dBm
- پهنای باند
  - از 7.8 kHz تا 500 kHz
- ضریب گسترش
  - از 7 تا 12
- نرخ کد
  - از $\frac{1}{2}$ تا 1 (بدون کدینگ)



ضریب گسترش (Spreading Factor)



ضریب گسترش (Spreading Factor)



میزان مصرف انرژی

- مشخصات SX127x (فقط خود IC)

Description	Conditions	Min	Typ	Max	Unit
Supply current in Sleep mode		-	0.2	1	uA
Supply current in Idle mode	RC oscillator enabled	-	1.5	-	uA
Supply current in Standby mode	Crystal oscillator enabled	-	1.6	1.8	mA
Supply current in Synthesizer mode	FSRx	-	5.8	-	mA
Supply current in Receive mode	*LnaBoost* Off, band 1 *LnaBoost* On, band 1 Bands 2&3	- - -	10.8 11.5 12.0	- - -	mA
Supply current in Transmit mode with impedance matching	RFOP = +20 dBm, on PA_BOOST RFOP = +17 dBm, on PA_BOOST RFOP = +13 dBm, on RFO_LF/HF pin RFOP = + 7 dBm, on RFO_LF/HF pin	- - - -	120 87 29 20	- - - -	mA mA mA mA

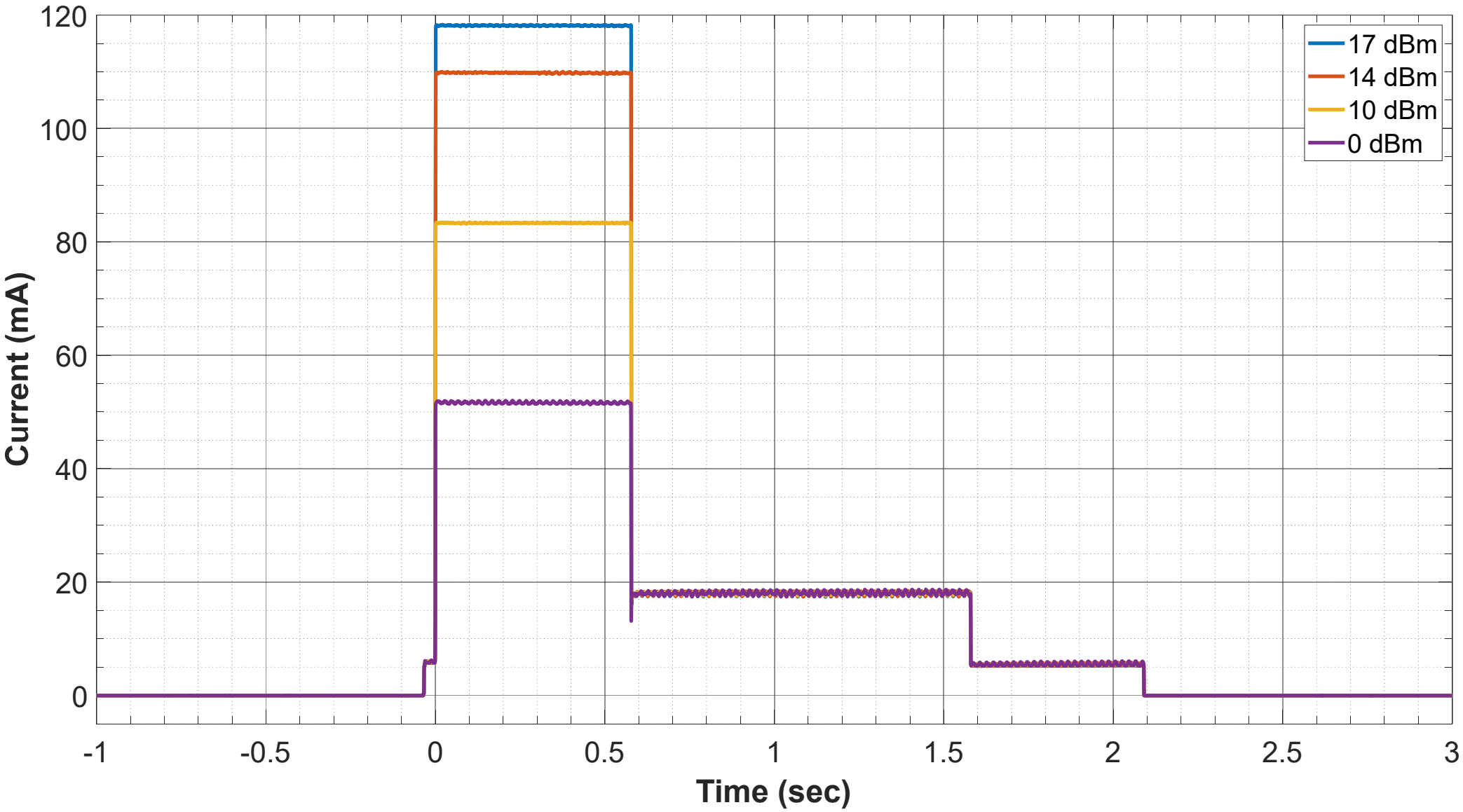
میزان مصرف انرژی

- مشخصات LoRa Ra-02 (مدار کامل)

Max Transmit Power	18±1 dBm
Power (Typical Values)	433MHz: TX:93mA RX:12.15mA Standby:1.6mA 470MHZ: TX:97mA RX:12.15mA Standby:1.5mA
Power Supply	2.5~3.7V， Typical 3.3V

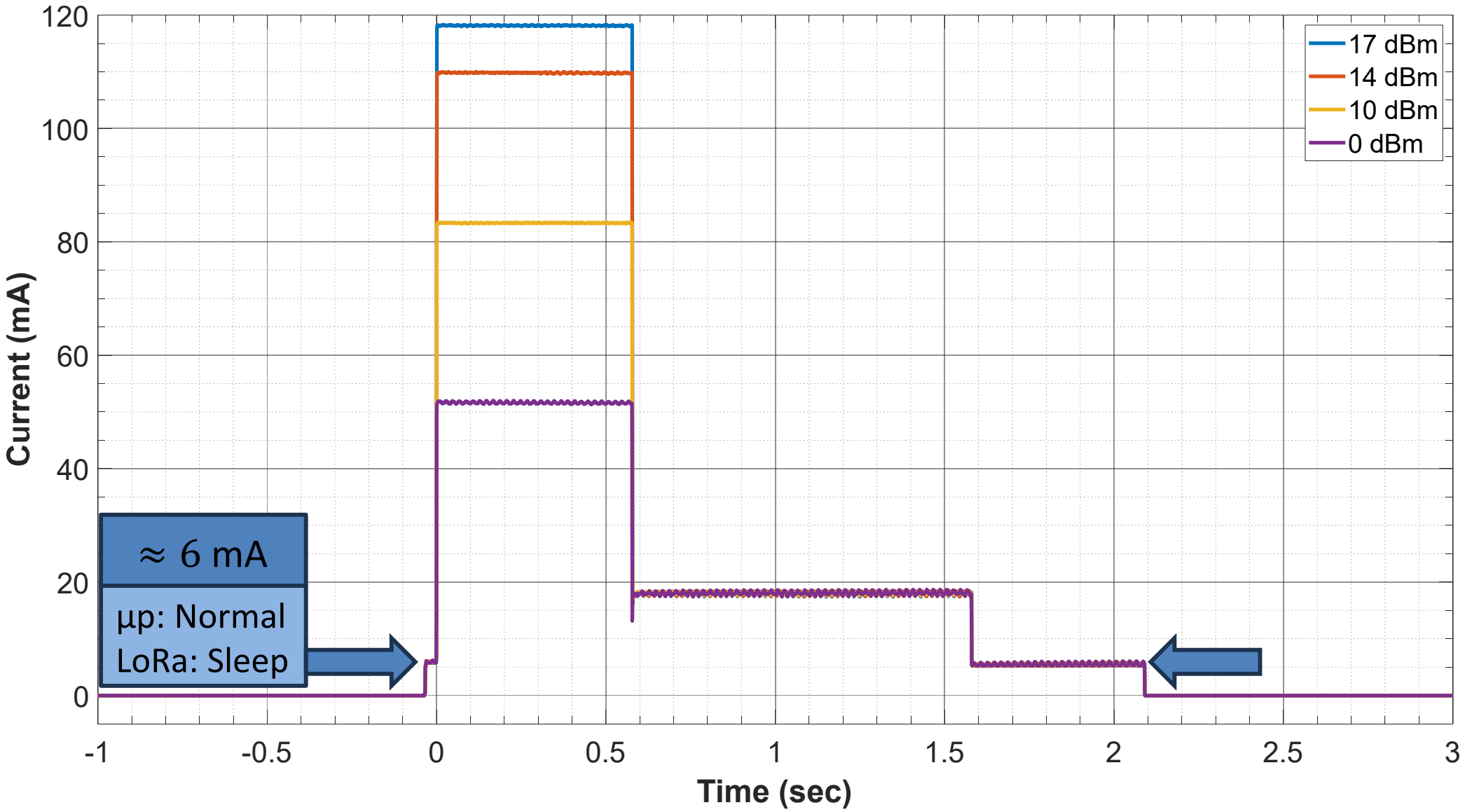
میزان مصرف انرژی

- اندازه‌گیری جریان (3.3 V)



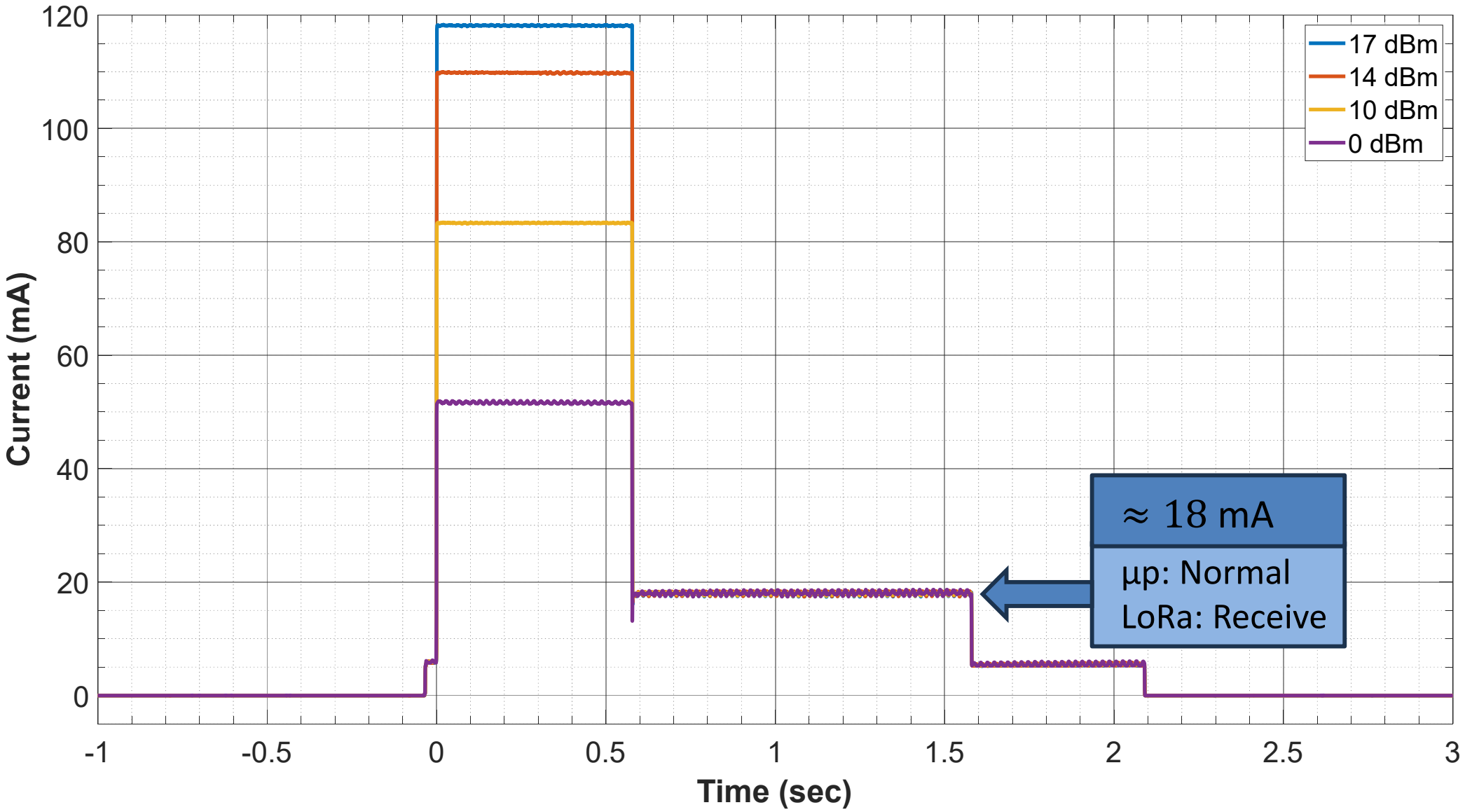
میزان مصرف انرژی

- اندازه‌گیری جریان (3.3 V)



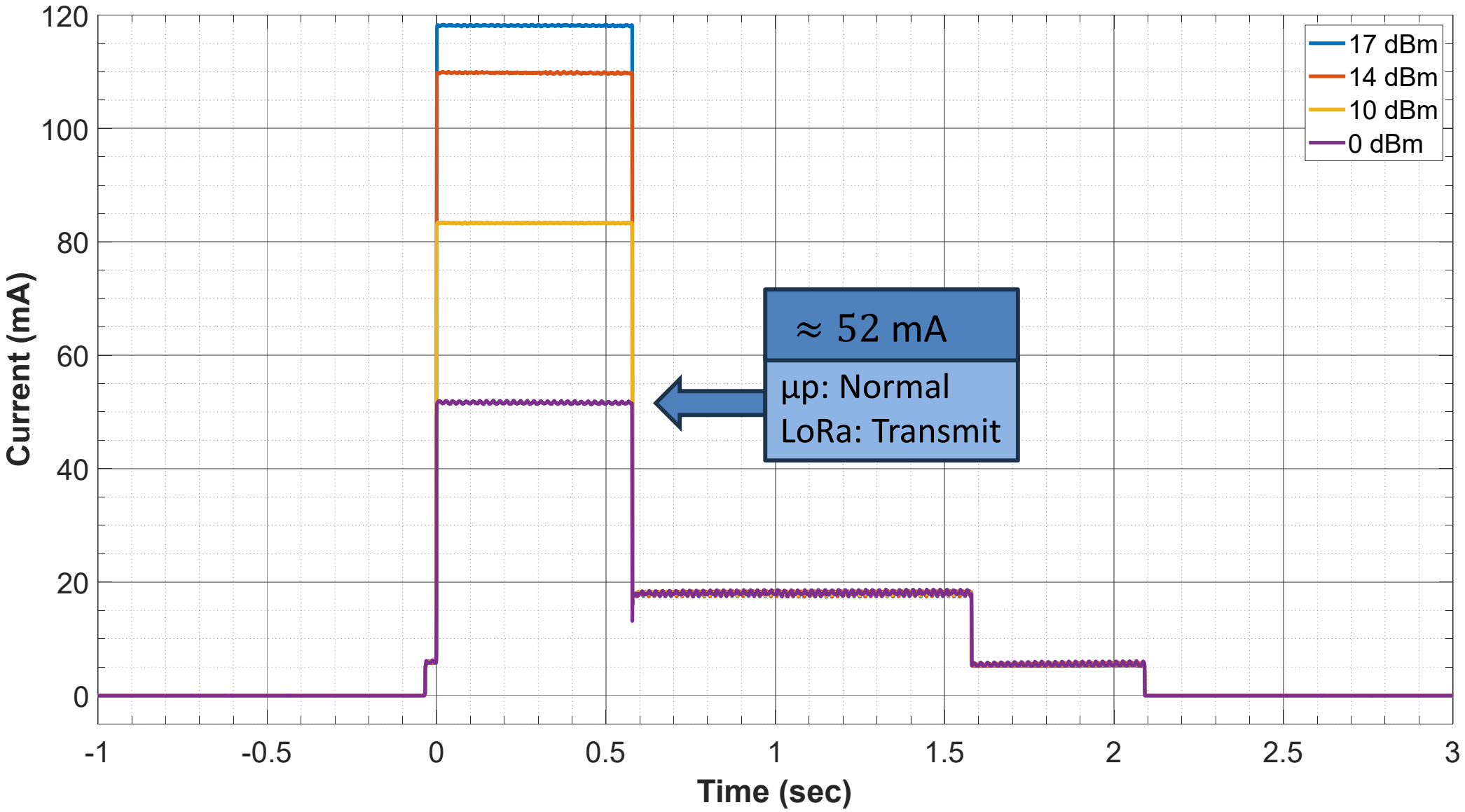
میزان مصرف انرژی

- اندازه‌گیری جریان (3.3 V)



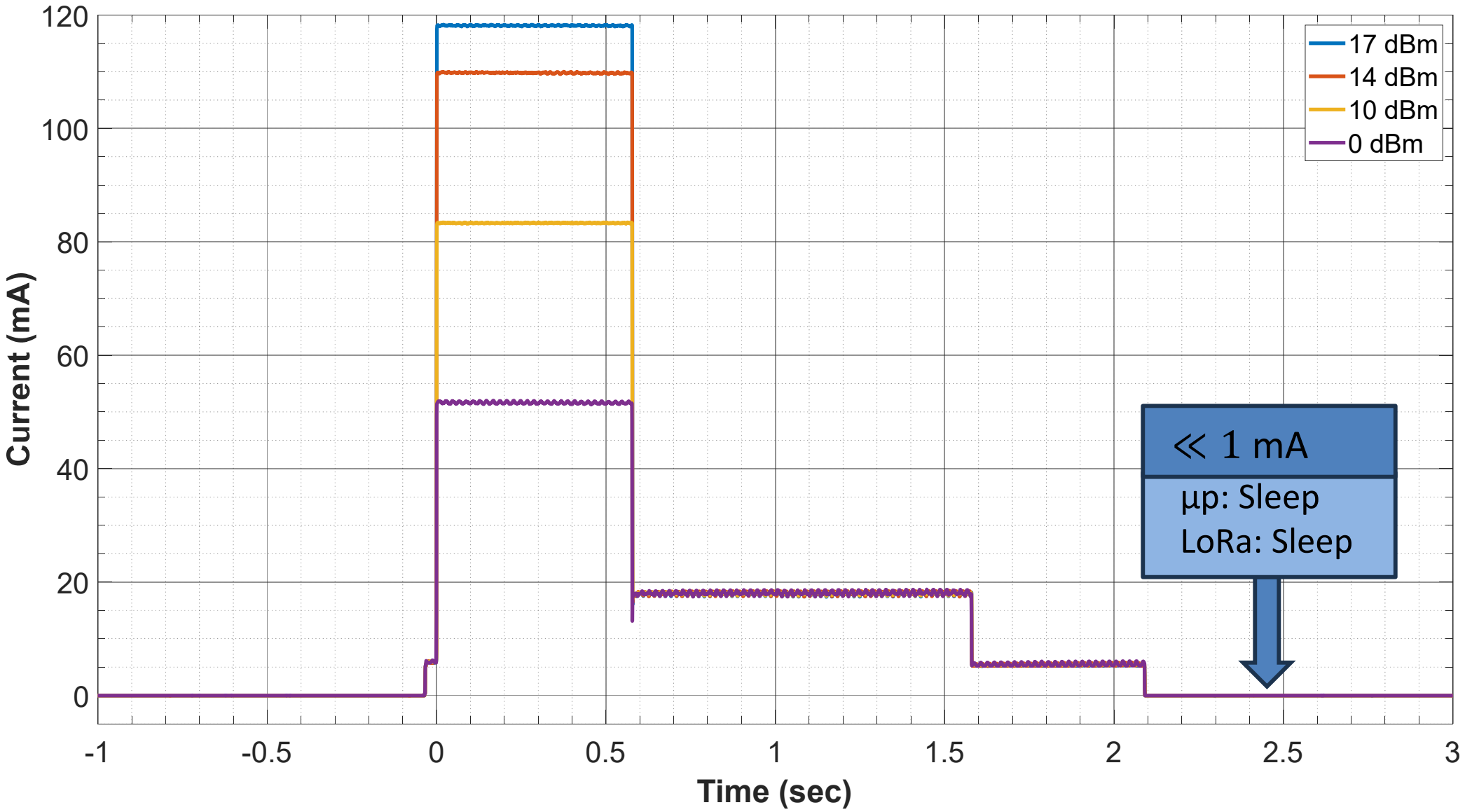
میزان مصرف انرژی

- اندازه‌گیری جریان (3.3 V)



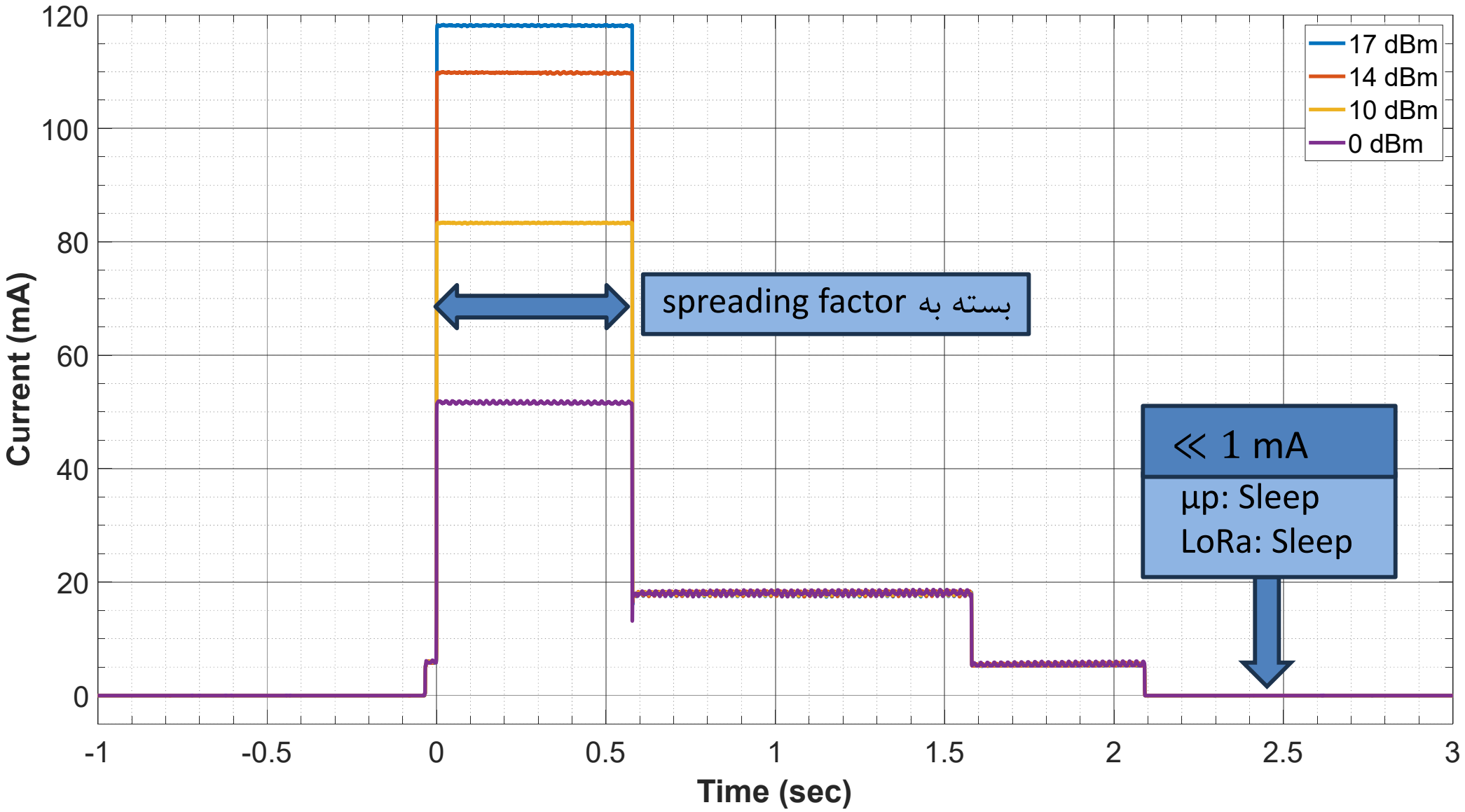
میزان مصرف انرژی

- اندازه‌گیری جریان (3.3 V)



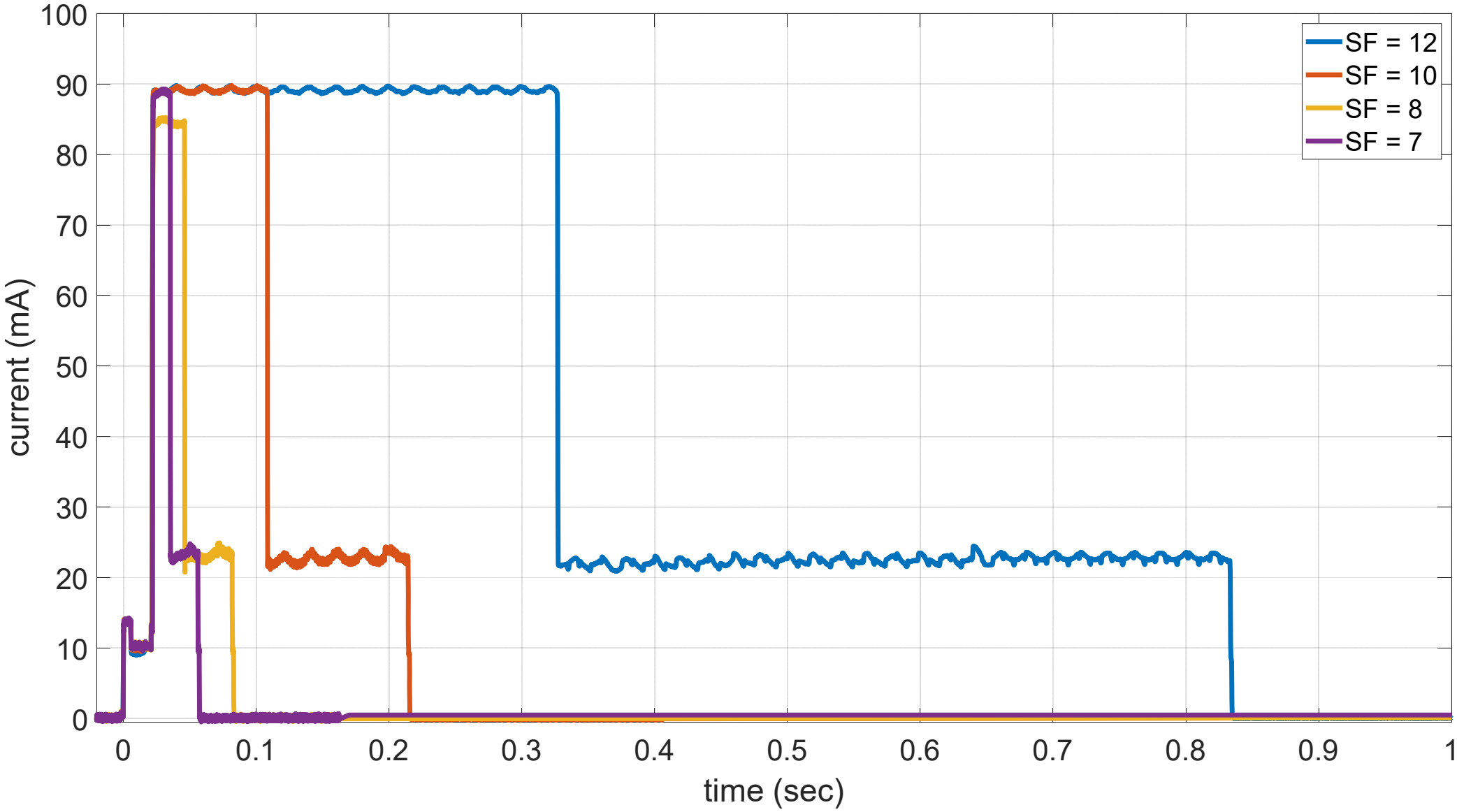
میزان مصرف انرژی

- اندازه‌گیری جریان (3.3 V)



میزان مصرف انرژی

- اندازه‌گیری جریان (3.3 V)

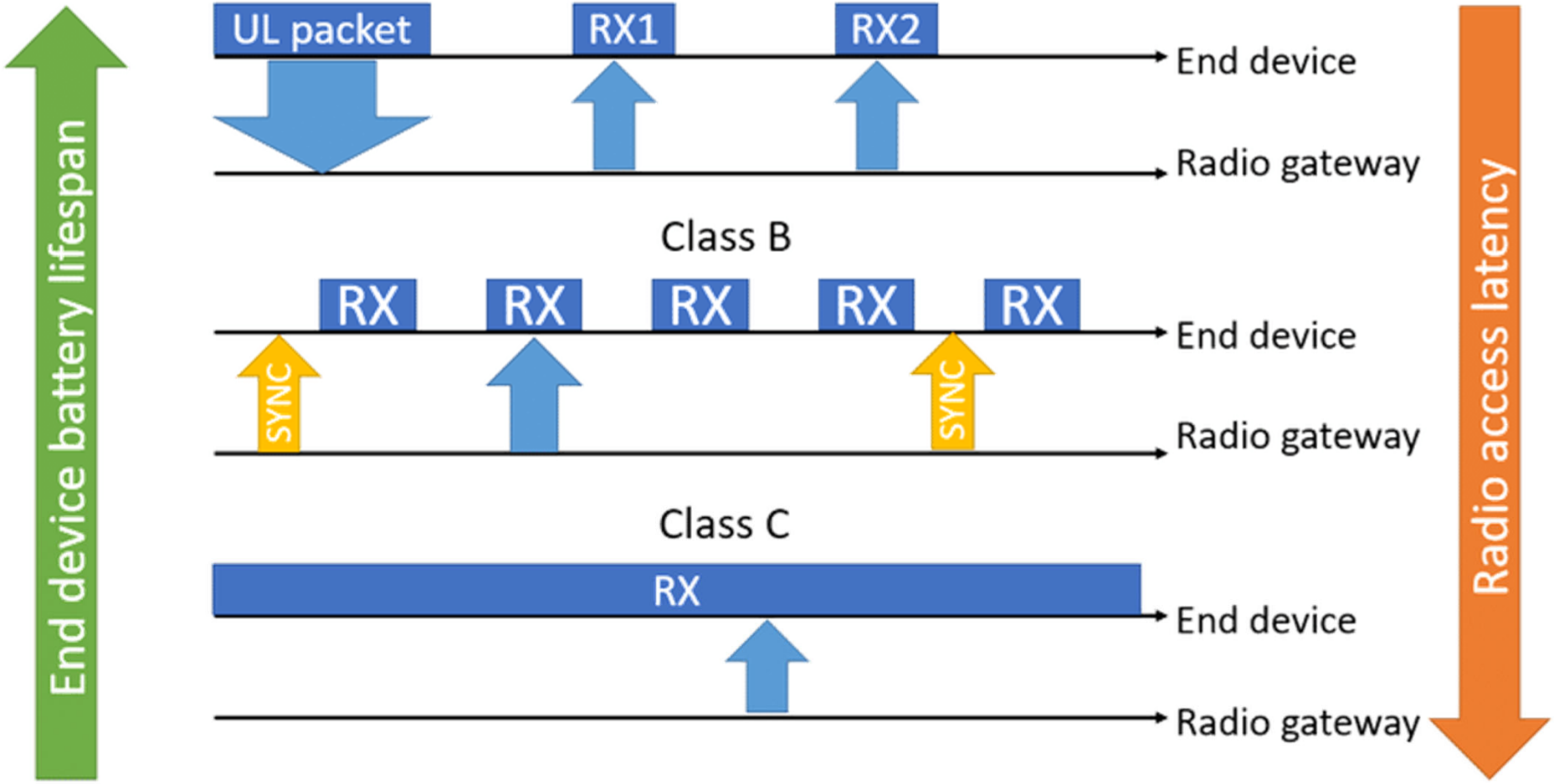


طول عمر باتری (مثلاً 600 mAh)

- بستگی به duty cycle دارد
  - کلاً خواب: چند ده سال
  - ارسال دائم: چند ساعت !!
  - دریافت دائم: چند روز
  - ارسال هر نیم ساعت یک‌بار: چند سال

طول عمر باتری (مثلاً 600 mAh)

- بستگی به duty cycle دارد
  - کلاً خواب: چند ده سال
  - ارسال دائم: چند ساعت !!
  - دریافت دائم: چند روز
  - ارسال هر نیم ساعت یک‌بار: چند سال



طول عمر باتری (مثلاً 600 mAh)

- بستگی به duty cycle دارد
  - کلاً خواب: چند ده سال
  - ارسال دائم: چند ساعت !!
  - دریافت دائم: چند روز
  - ارسال هر نیم ساعت یک‌بار: چند سال
  - ارسال هر نیم‌ساعت، دریافت پس از هر ارسال

$$I_{Tx} = 51\,mA, \quad T_{Tx} = 2 \times 0.4\,sec = 0.8\,sec$$
$$I_{Rx} = 18\,mA, \quad T_{Rx} = 2 \times 2\,sec = 4\,sec$$
$$I_{sleep} = 2\,\mu A, \quad T_{sleep} = 3600\,sec - (0.8 + 4)\,sec = 3595.2\,sec$$

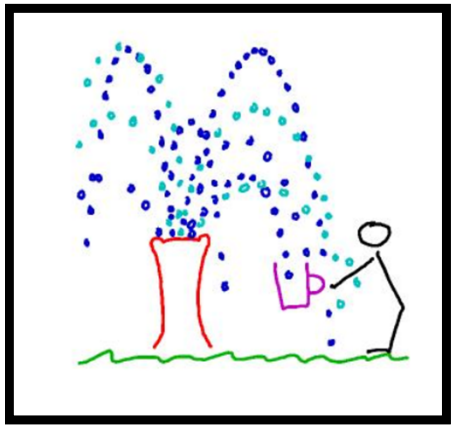
$$\begin{aligned} E_{1hour} &= V_{cc} \times (I_{Tx}T_{Tx} + I_{Rx}T_{Rx} + I_{sleep}T_{sleep}) \\ &= 3.3 \times (52 \times 0.8 + 18 \times 4 + 0.002 \times 3595.2) \\ &= 3.3 \times 120.79\,mVA\,sec \\ &= 398.6\ mVAsec \end{aligned}$$

$$\begin{aligned} E_{battery} &= 3.3 \times 600\ mVAh \\ &= 1980\ mVAh \\ &= 7128000\ mVA\,sec \end{aligned}$$

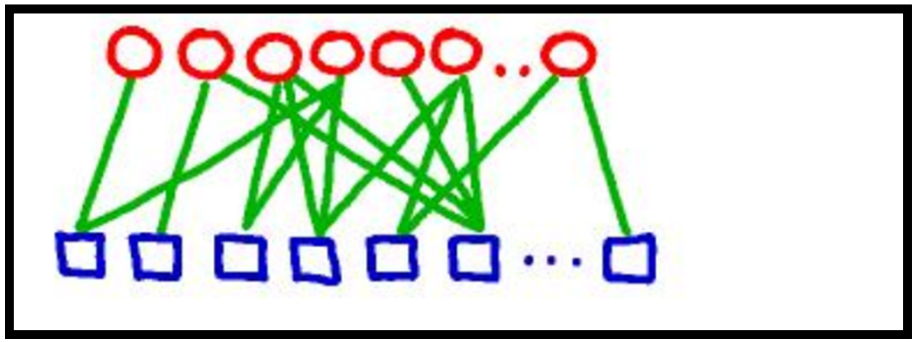
$$\text{Battery Life} = \frac{E_{battery}}{E_{1hour}} = \frac{7128000}{398.6}\,hour = 17882\,hour = 2\,year$$

کدهای آبشاری

- کدهای آبشاری (fountain) یا بدون نرخ (rateless)
- کدگذاری شبکه (Network coding)
- مانند پر کردن یک سطل در یک چشمه: مهم نیست کدام قطره درون سطل می‌ریزد، مهم این است به تعداد کافی قطره داخل سطل باشد تا سطل پر شود.

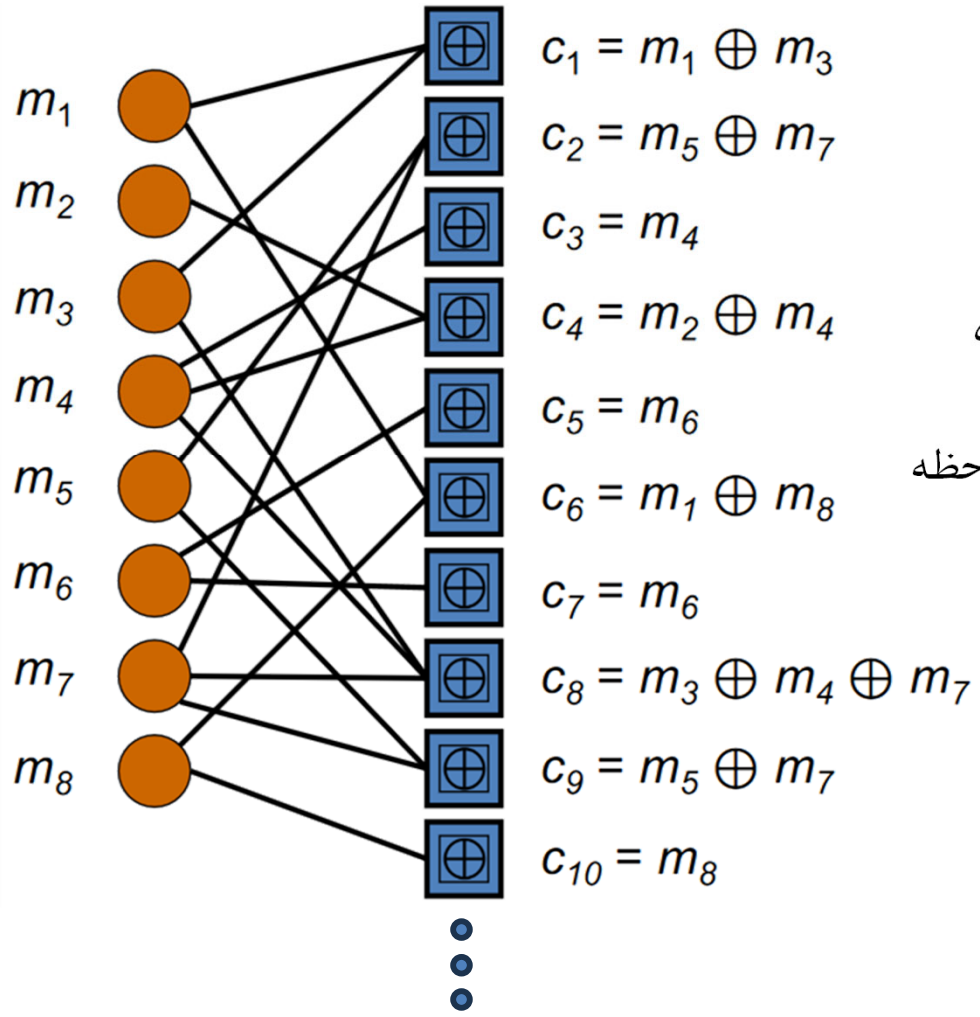


- در این‌جا هم، مهم نیست کدام بسته‌ها به مقصد برسند، فقط کافی است به تعداد کافی بسته به مقصد برسد تا اطلاعات قابل بازیابی باشد.



کدهای آبشاری

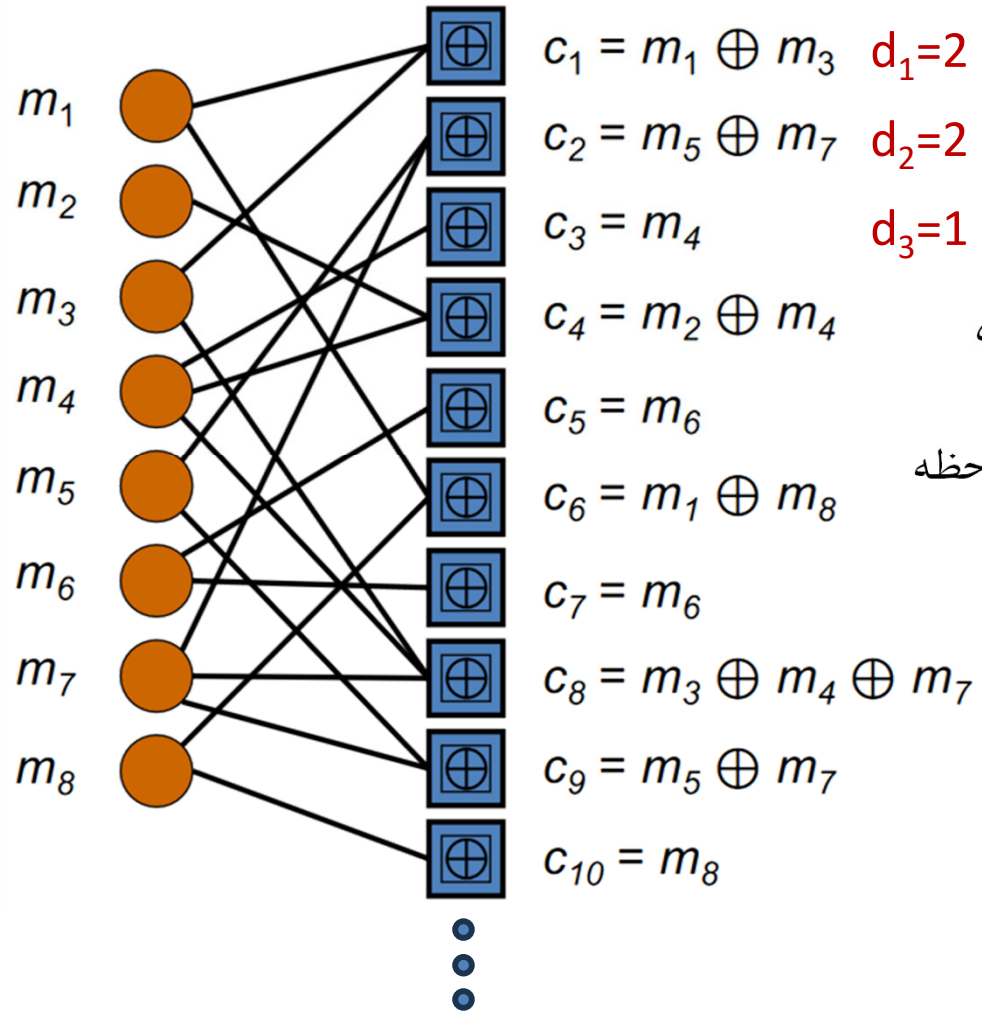
- فرض کنیم k بستهٔ اطلاعات را قرار است ارسال کنیم.
  - استفاده از کدهای با نرخ ثابت: تطبیق نرخ با شرایط کانال
  - استفاده از کدهای بدون نرخ



- کدهای بدون نرخ:
  - فرستنده مدام بسته‌های جدید تولید می‌کند و به گیرنده ارسال می‌کند.
  - گیرنده بررسی می‌کند که آیا بسته‌های دریافتی تا این لحظه برای فهمیدن بسته‌های ارسالی کافی است؟
    - حداقل k بسته باید دریافت شود.
    - حداکثر؟ معلوم نیست!
  - اگر بلی، ارسال پیام برای توقف ارسال به فرستنده.

کدهای آبشاری

- فرض کنیم k بستهٔ اطلاعات را قرار است ارسال کنیم.
  - استفاده از کدهای با نرخ ثابت: تطبیق نرخ با شرایط کانال
  - استفاده از کدهای بدون نرخ



- کدهای بدون نرخ:
  - فرستنده مدام بسته‌های جدید تولید می‌کند و به گیرنده ارسال می‌کند.
  - گیرنده بررسی می‌کند که آیا بسته‌های دریافتی تا این لحظه برای فهمیدن بسته‌های ارسالی کافی است؟
    - حداقل k بسته باید دریافت شود.
    - حداکثر؟ معلوم نیست!
  - اگر بلی، ارسال پیام برای توقف ارسال به فرستنده.

تعریف:
درجه: تعداد پیام‌های موجود در بستهٔ ارسالی

روش decode کردن

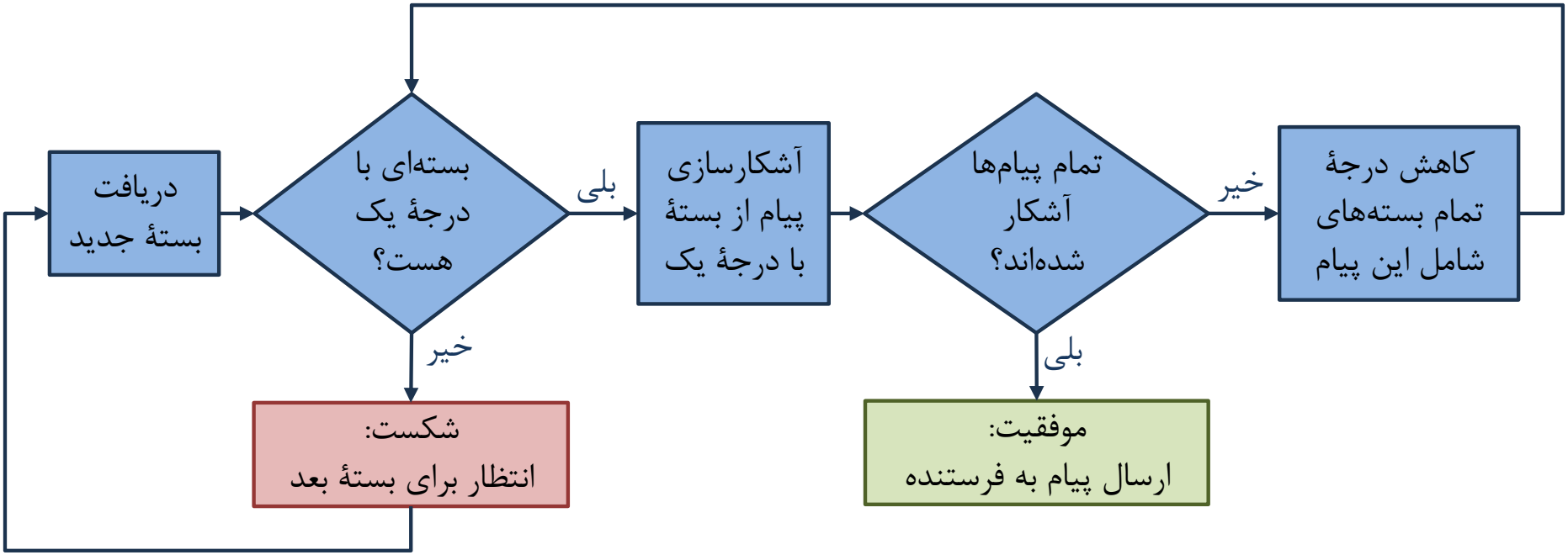
- روش ایده‌آل: آیا دستگاه معادلات برای حل معادله و به دست آوردن k مجهول کافی است؟
  - باید k بستهٔ مستقل خطی دریافت شده باشد.
  - حل دستگاه ساده نیست.

- روش عملی:
  - ابتدا بسته‌های با درجهٔ «یک» پیدا می‌شوند.
  - درجهٔ تمام بسته‌های شامل این پیام یکی کم می‌شود.
  - اگر تمام بسته‌ها پیدا شدند: success

روش decode کردن

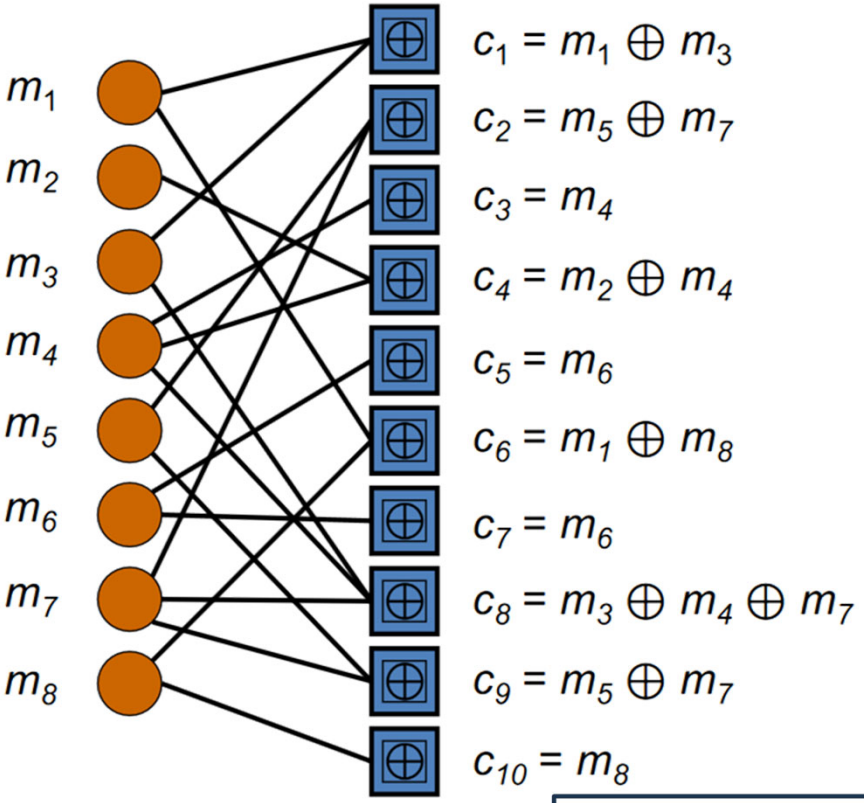
- روش ایده‌آل: آیا دستگاه معادلات برای حل معادله و به دست آوردن k مجهول کافی است؟
  - باید k بستهٔ مستقل خطی دریافت شده باشد.
  - حل دستگاه ساده نیست.

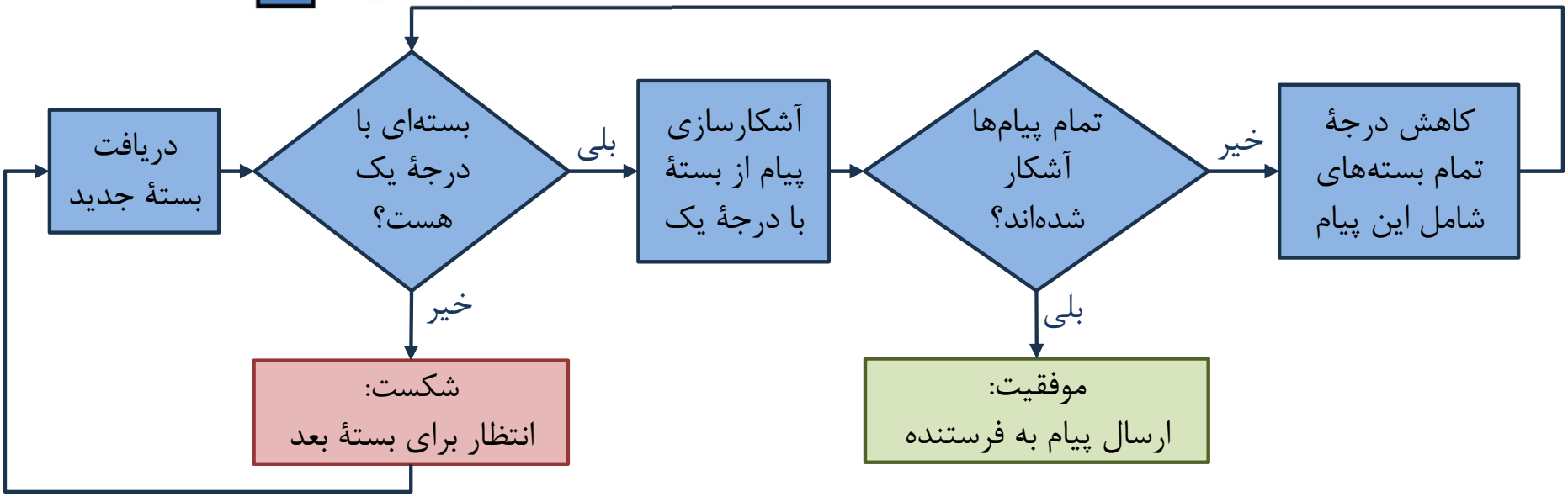
- روش عملی:
  - ابتدا بسته‌های با درجهٔ «یک» پیدا می‌شوند.
  - درجهٔ تمام بسته‌های شامل این پیام یکی کم می‌شود.
  - اگر تمام بسته‌ها پیدا شدند: success



روش decode کردن

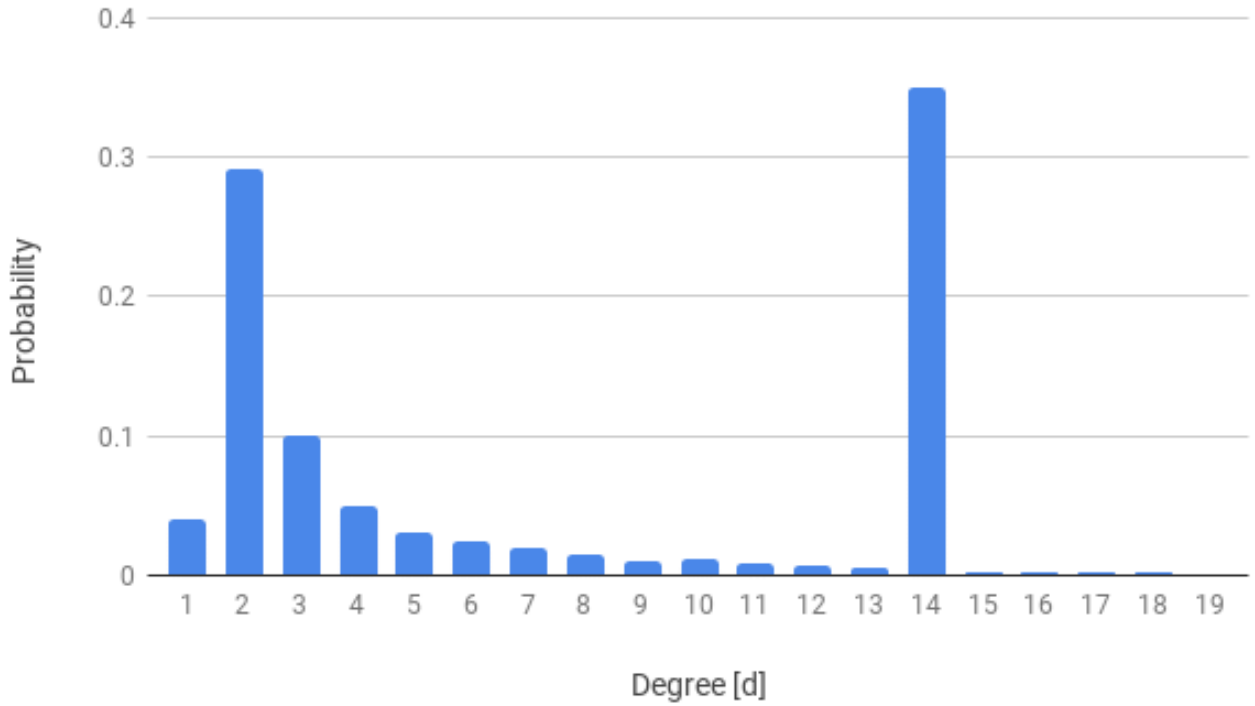
- اجرای الگوریتم:



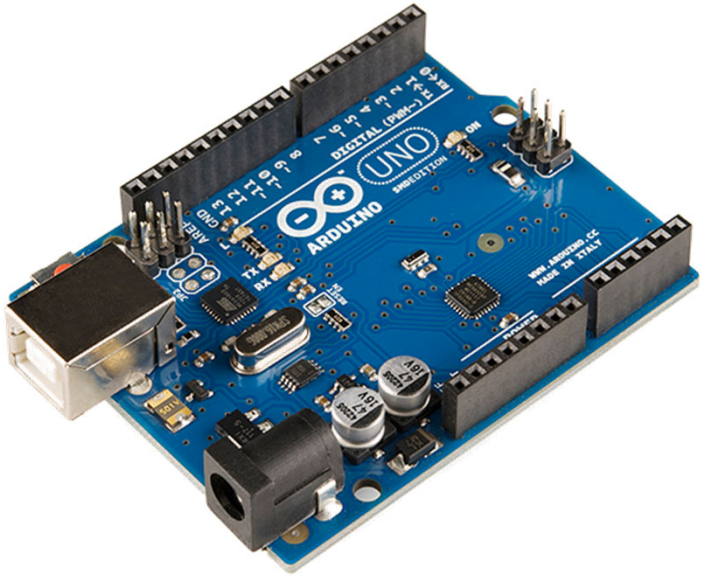


درجه

- تعداد مناسبی بسته با درجهٔ یک برای شروع الگوریتم
- تعداد زیادی بسته با درجهٔ پایین برای ادامهٔ الگوریتم
- تعدادی بسته با درجهٔ بالا برای اطمینان از وجود تمام پیام‌ها در بسته‌های دریافتی



معرفی Arduino



- اولین مدل: Arduino UNO
  - مدل برنامه‌نویسی بسیار ساده و سطح بالا
  - پردازشگر ساده
  - ارتباطات پایه
  - حجم حافظهٔ پایین
  - **قیمت ارزان**

- اولین مدل: Arduino UNO
  - مدل برنامه‌نویسی بسیار ساده و سطح بالا
  - پردازشگر ساده
  - ارتباطات پایه
  - حجم حافظهٔ پایین
  - **قیمت ارزان**

28.5.10 SPI register map

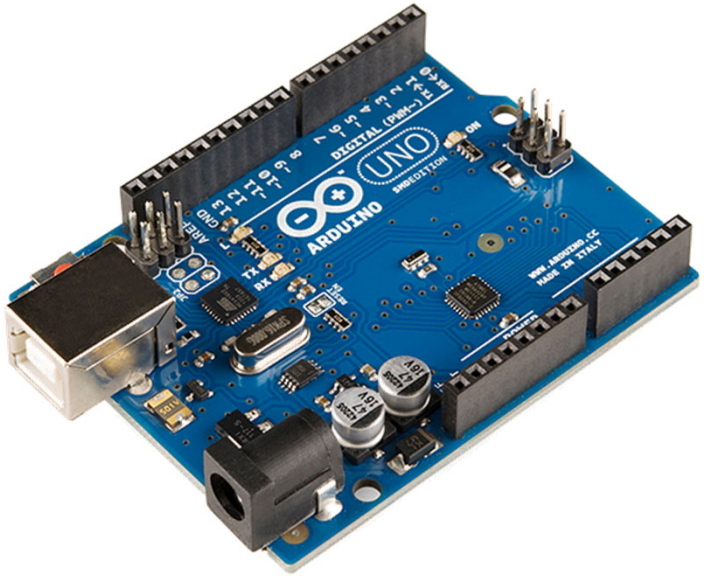
The table provides shows the SPI register map and reset values.

Table 129. SPI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	Reserved																								TXEIE	RXNEIE	ERRIE	FRF	Reserved	SSOE	TXDMAEN	RXDMAEN
	Reset value																									0	0	0	0		0	0	0
0x08	SPI_SR	Reserved																							FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																								0	0	0	0	0	0	0	1	0
0x0C	SPI_DR	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPI_RXCRCR	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPI_I2SCFGR	Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN	CHLEN
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x20	SPI_I2SPR	Reserved																						MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

Refer to *Section 2.3: Memory map* for the register boundary addresses.

معرفی Arduino



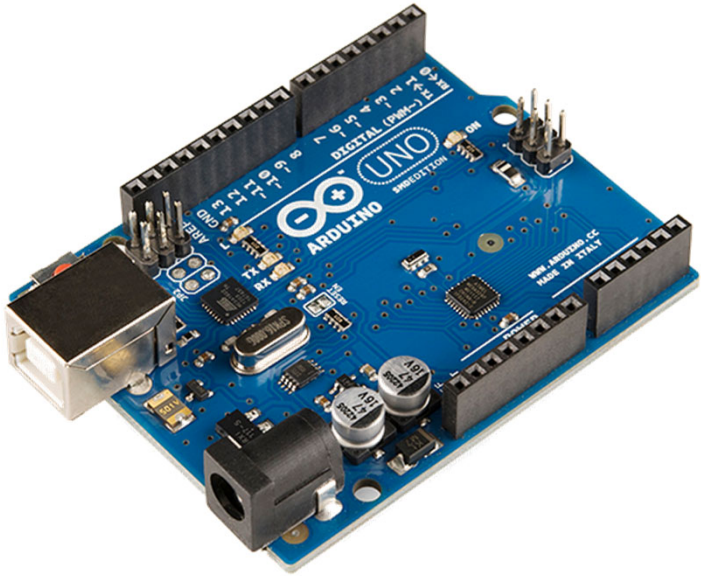
- اولین مدل: Arduino UNO
  - مدل برنامه‌نویسی بسیار ساده و سطح بالا
  - پردازشگر ساده
  - ارتباطات پایه
  - حجم حافظهٔ پایین
  - **قیمت ارزان**

```
// SPI Config
  SpiaRegs.SPICCR.all =0x0047;
  SpiaRegs.SPICTL.all =0x0006;
  SpiaRegs.SPIBRR =19;
  SpiaRegs.SPICCR.all =0x00C7;
  SpiaRegs.SPIPRI.bit.FREE = 1;
```

```
// Transmit start
SpiaRegs.SPITXBUF=0xFA00;
while ( ! SpiaRegs.SPISTS.bit.INT_FLAG ) {}
SpiaRegs.SPIRXBUF=SpiaRegs.SPIRXBUF;
SpiaRegs.SPITXBUF=low<<8;
while ( ! SpiaRegs.SPISTS.bit.INT_FLAG ) {}
SpiaRegs.SPIRXBUF=SpiaRegs.SPIRXBUF;

SpiaRegs.SPITXBUF=hi<<8;
while ( ! SpiaRegs.SPISTS.bit.INT_FLAG ) {}
SpiaRegs.SPIRXBUF=SpiaRegs.SPIRXBUF;
```

معرفی Arduino

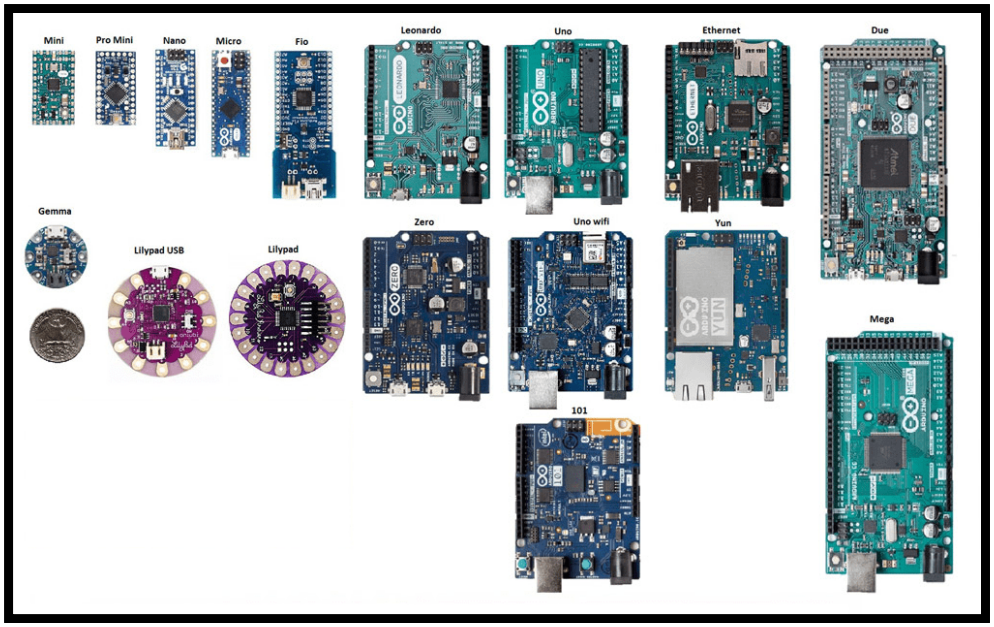


- اولین مدل: Arduino UNO
  - مدل برنامه‌نویسی بسیار ساده و سطح بالا
  - پردازشگر ساده
  - ارتباطات پایه
  - حجم حافظهٔ پایین
  - **قیمت ارزان**

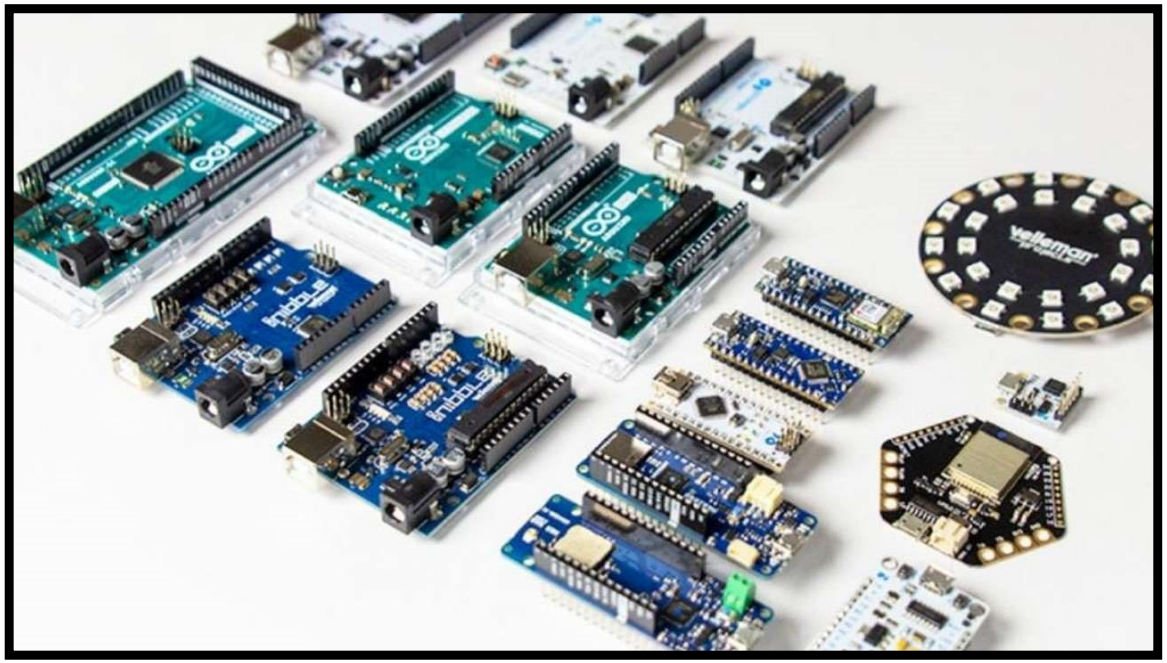
```
// SPI Config
 SPI.begin();
```

```
// Transmit start
SPI.transfer(0xA5);
```

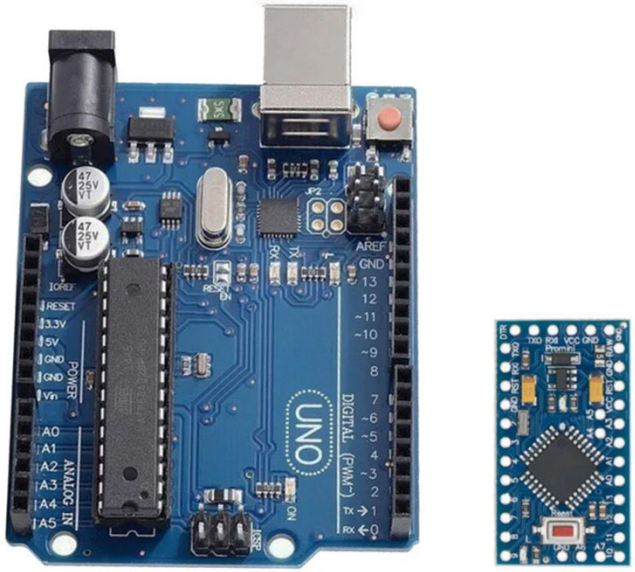
معرفی Arduino



- اولین مدل: Arduino UNO
  - مدل برنامه‌نویسی بسیار ساده و سطح بالا
  - پردازشگر ساده
  - ارتباطات پایه
  - حجم حافظهٔ پایین
  - **قیمت ارزان**
- گسترش فوق‌العاده سریع
  - جامعهٔ کاربری فعال
  - ایجاد کتابخانه تقریباً برای همهٔ سخت‌افزارها
- مدل‌های بعدی:
  - پردازنده‌های قوی‌تر
  - حافظه‌های بزرگتر
  - ابعاد کوچک‌تر
  - ورودی و خروجی‌های متنوع
  - ادوات جانبی گسترده



Arduino Pro Mini



- دقیقاً مشابه UNO
- ابعاد کوچک‌تر
- بدون امکان اتصال به USB

- بخش‌های برنامه:
  - `setup()`
  - `loop()`

- اجرای برنامهٔ ساده
  - فایل start.ino
  - اتصال بورد به رابط سریال و رابط سریال به کامپیوتر
  - انتخاب بورد
  - اجرای برنامه
  - باز کردن رابط سریال

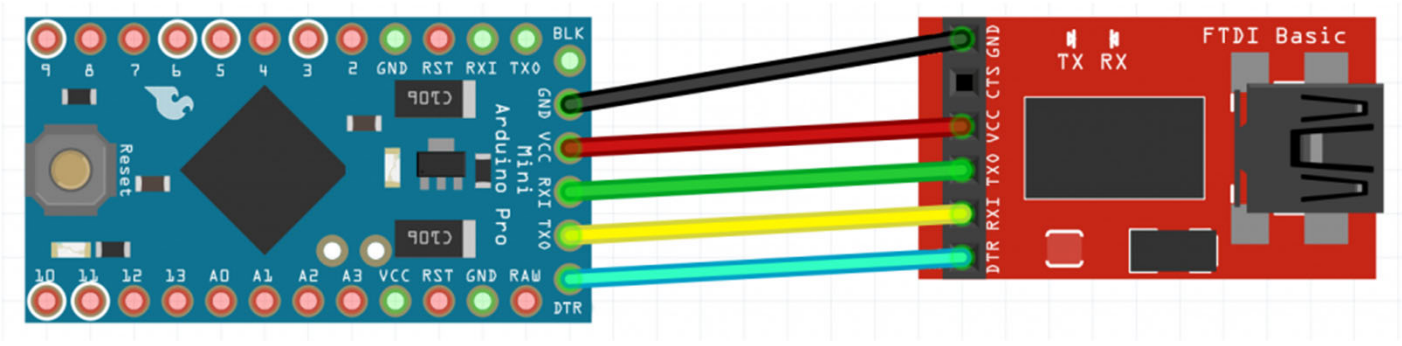
Arduino Pro Mini

- دقیقاً مشابه UNO
- ابعاد کوچک‌تر
- بدون امکان اتصال به USB

Arduino Pro Mini	UART to USB
GND	GND
VCC	+3.3V
RX	TX
TX	RX
DTR	DTR

- بخش‌های برنامه:
  - `setup()`
  - `loop()`

- اجرای برنامهٔ ساده
  - فایل start.ino
  - **اتصال بورد به رابط سریال و رابط سریال به کامپیوتر**
  - انتخاب بورد
  - اجرای برنامه
  - باز کردن رابط سریال

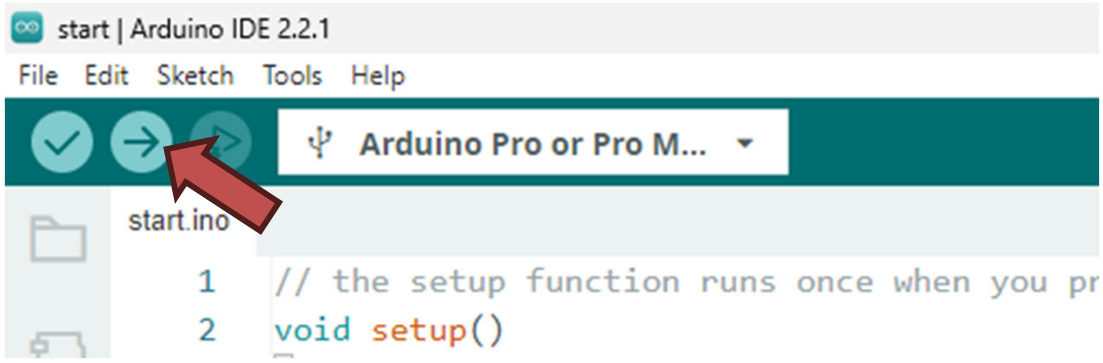


Arduino Pro Mini

- دقیقاً مشابه UNO
- ابعاد کوچک‌تر
- بدون امکان اتصال به USB

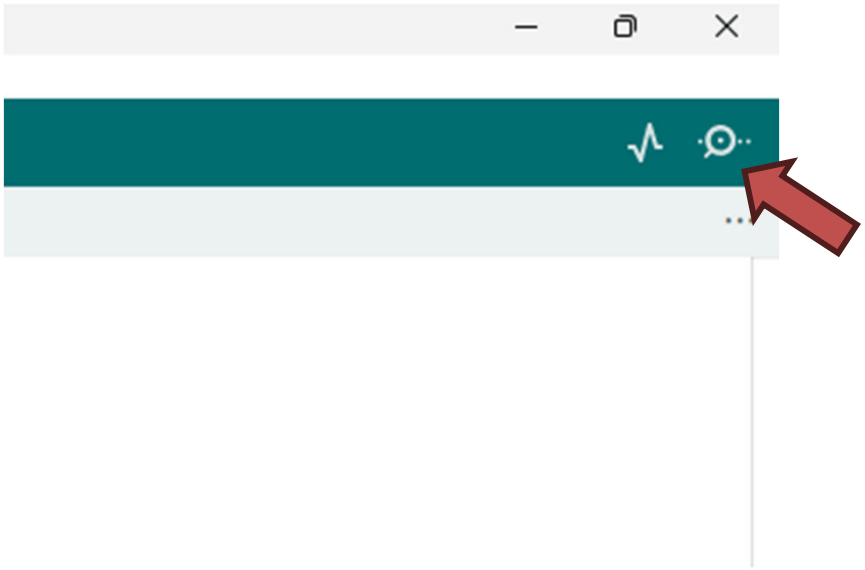
- بخش‌های برنامه:
  - `setup()`
  - `loop()`

- اجرای برنامهٔ ساده
  - فایل start.ino
  - انتخاب بورد
  - **اجرای برنامه**
  - باز کردن رابط سریال



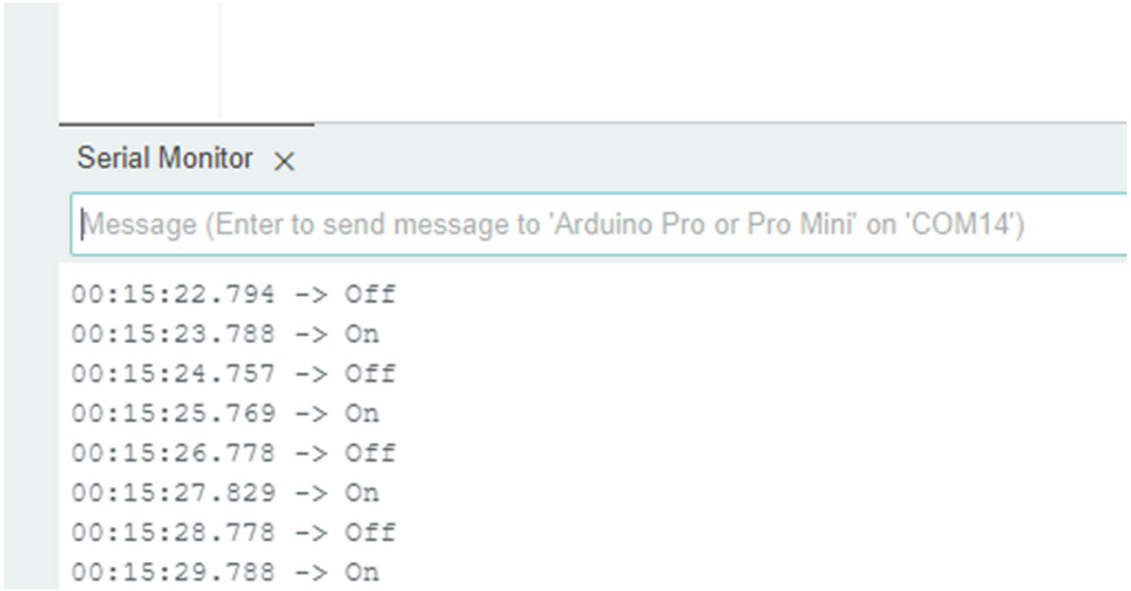
Arduino Pro Mini

- دقیقاً مشابه UNO
- ابعاد کوچک‌تر
- بدون امکان اتصال به USB



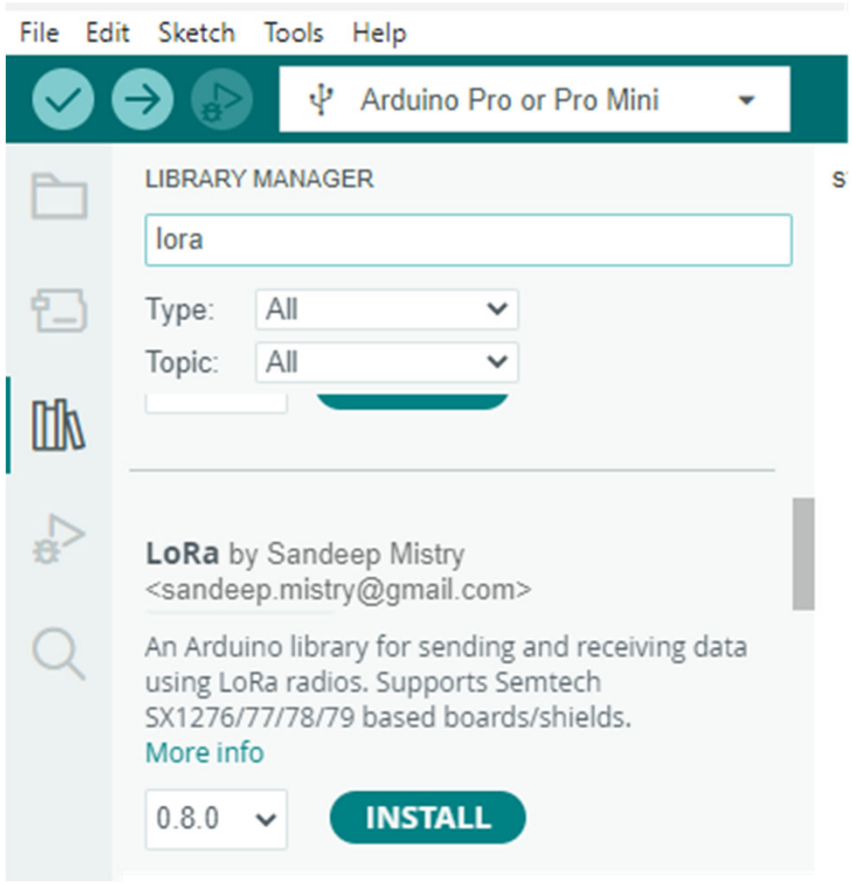
- بخش‌های برنامه:
  - `setup()`
  - `loop()`

- اجرای برنامهٔ ساده
  - فایل start.ino
  - انتخاب بورد
  - اجرای برنامه
  - **باز کردن رابط سریال**



برنامه‌نویسی برای LoRa

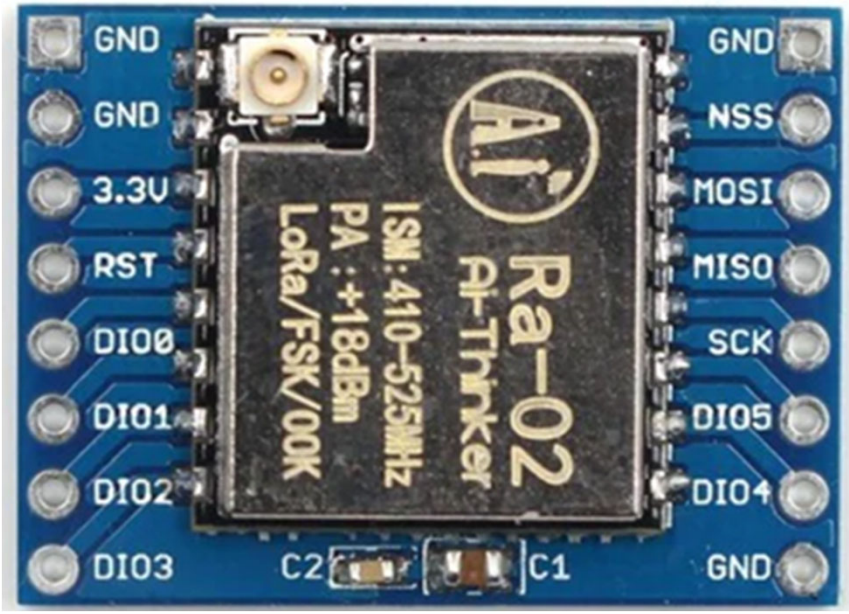
- نصب کتابخانه
  - Tools >> Manage Libraries ...
  - جستجوی lora
  - انتخاب LoRa by Sandeep Mistry
  - کلیک بر روی Install



برنامه‌نویسی برای LoRa

- نصب کتابخانه
- برقراری اتصالات

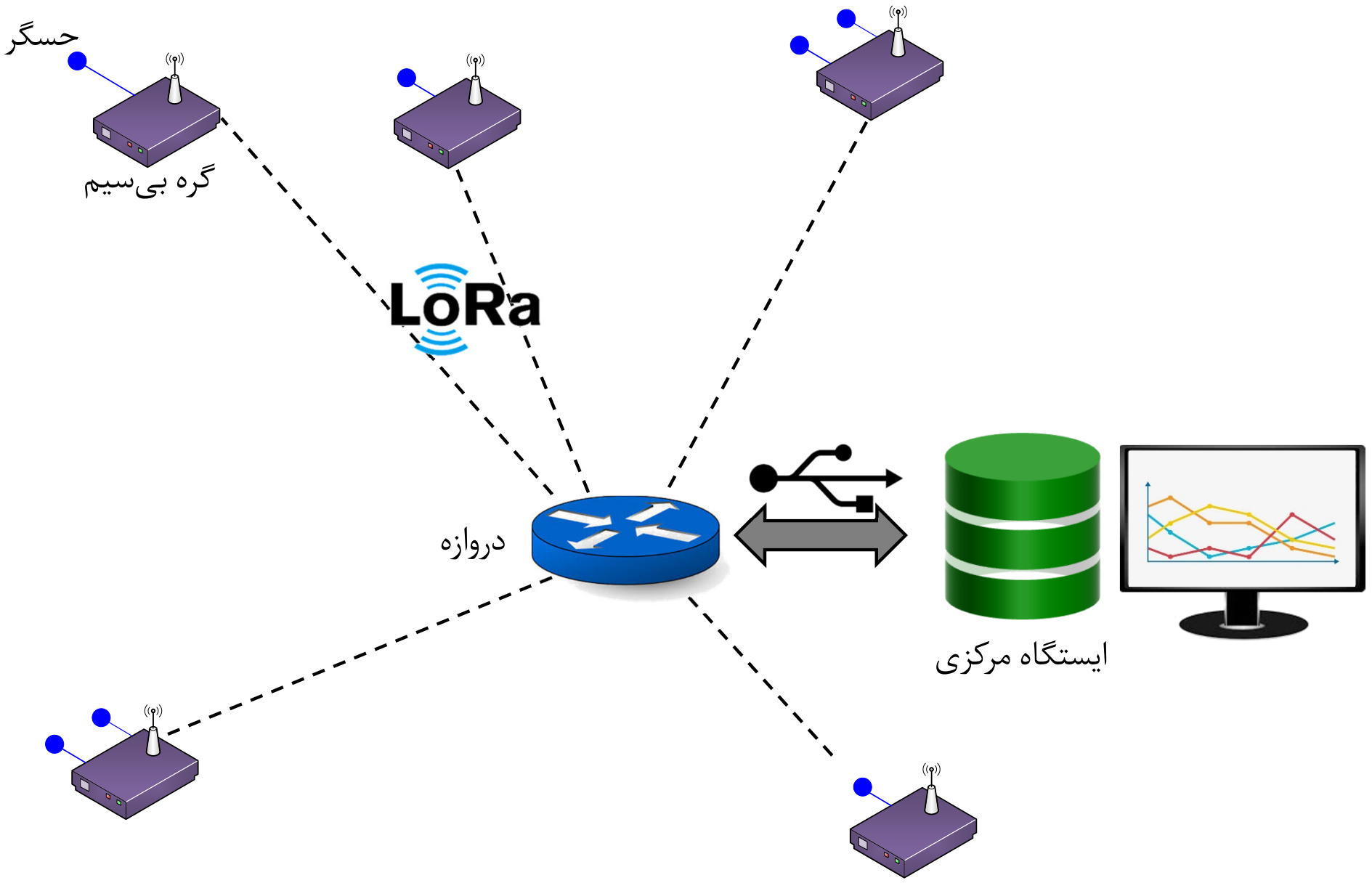
LoRa	Arduino
GND	GND
VCC	VCC
RST	9
NSS	10
MOSI	11
MISO	12
SCK	13
DIO0	2



برنامه‌نویسی برای LoRa

- نصب کتابخانه
- برقراری اتصالات
- اجرای برنامه
  - فایل lora_test1.ino
  - تغییر خط 9 برنامه و قرار دادن نام خودتان.
  - اجرای برنامه و مشاهدهٔ خروجی سریال.

مدل شبکه

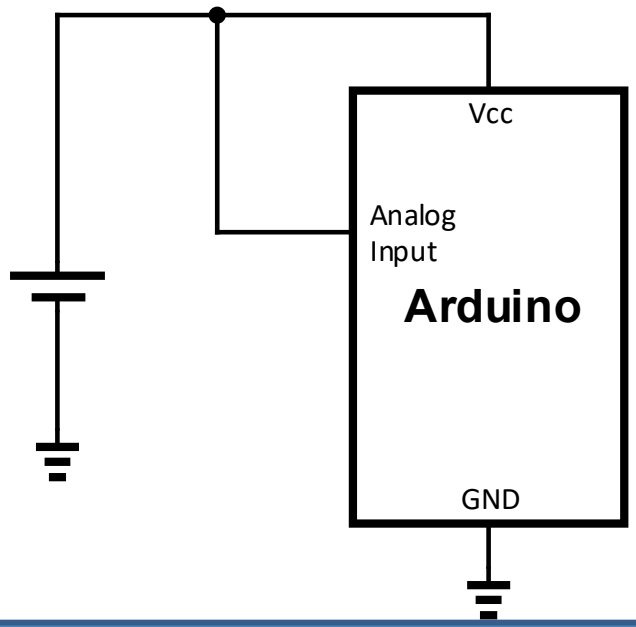


سخت‌افزار

- حسگرهای دما و رطوبت (نداریم!)
- LED
- کلید فشاری
- کاهش مصرف توان
  - تغذیه LED
  - رگولاتور ولتاژ

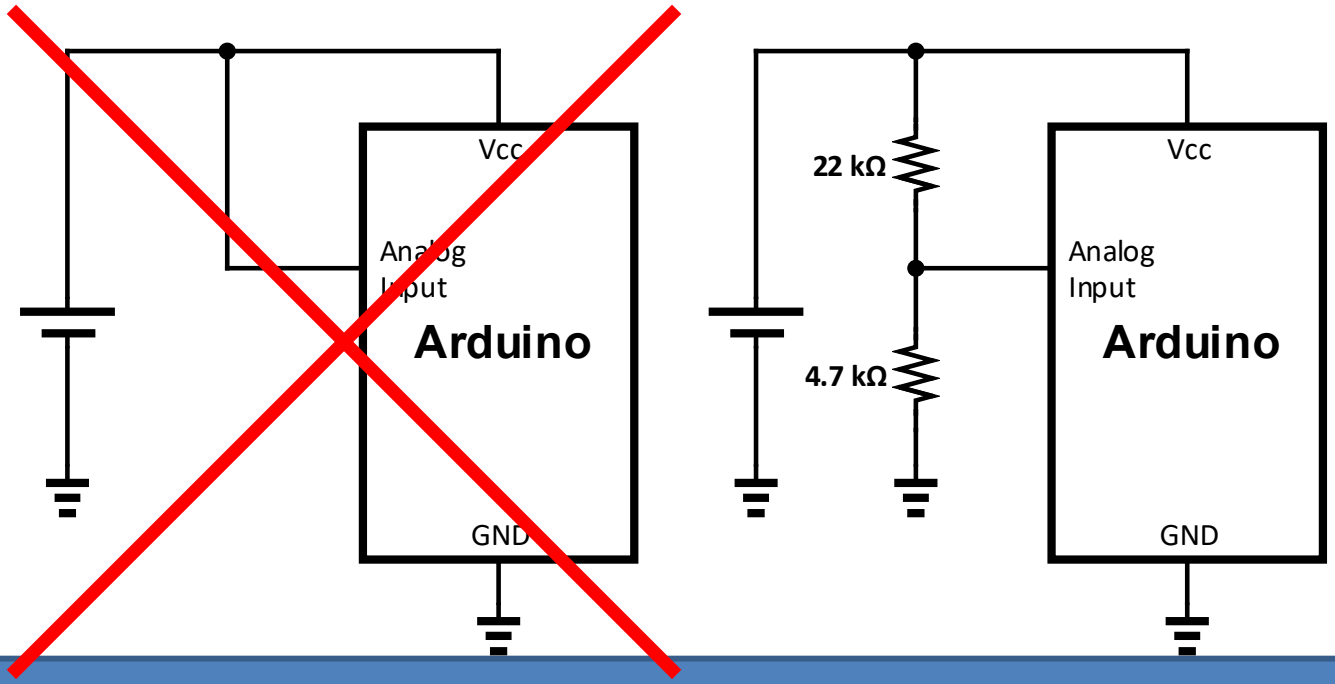
سخت‌افزار

- حسگرهای دما و رطوبت (نداریم!)
- LED
- کلید فشاری
- کاهش مصرف توان
  - تغذیه LED
  - رگولاتور ولتاژ
- اندازه‌گیری ولتاژ تغذیه
  - Arduino پایه‌هایی برای اندازه‌گیری ولتاژ آنالوگ دارد.



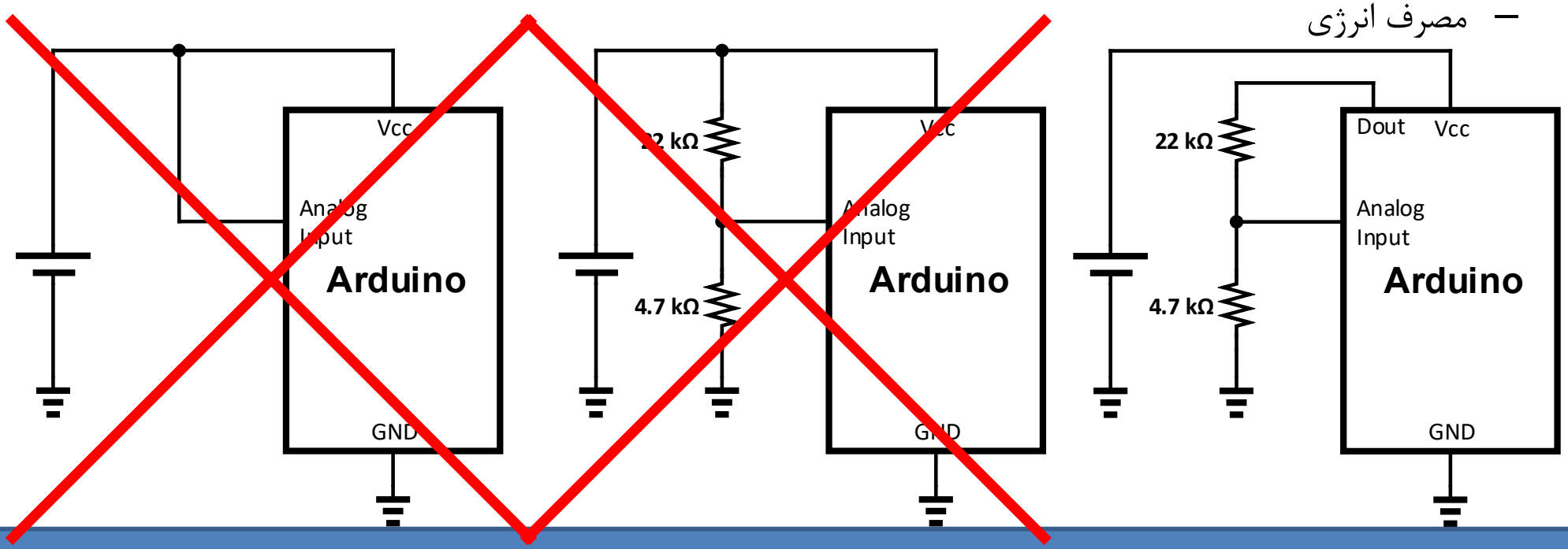
سخت‌افزار

- حسگرهای دما و رطوبت (نداریم!)
- LED
- کلید فشاری
- کاهش مصرف توان
  - تغذیه LED
  - رگولاتور ولتاژ
- اندازه‌گیری ولتاژ تغذیه
  - Arduino پایه‌هایی برای اندازه‌گیری ولتاژ آنالوگ دارد.
  - ولتاژ مرجع 1.1 V



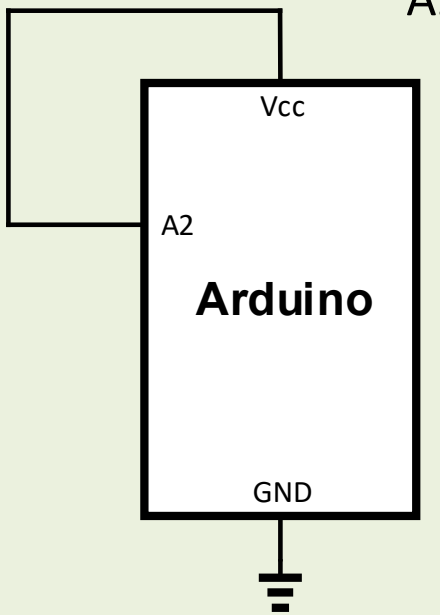
سخت‌افزار

- حسگرهای دما و رطوبت (نداریم!)
- LED
- کلید فشاری
- کاهش مصرف توان
  - تغذیه LED
  - رگولاتور ولتاژ
- اندازه‌گیری ولتاژ تغذیه
  - Arduino پایه‌هایی برای اندازه‌گیری ولتاژ آنالوگ دارد.
  - ولتاژ مرجع 1.1 V
  - مصرف انرژی

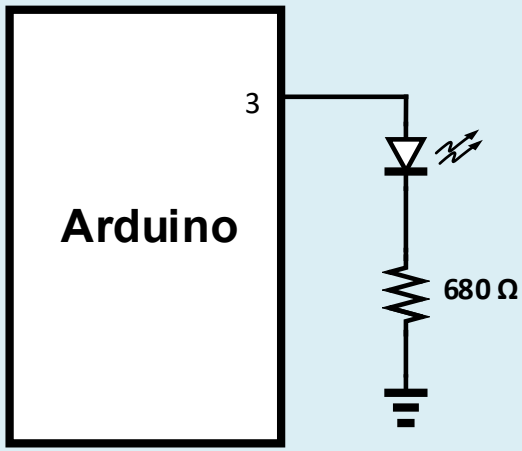


- موارد زیر را به مدار قبلی اضافه می‌کنیم (اختیاری):

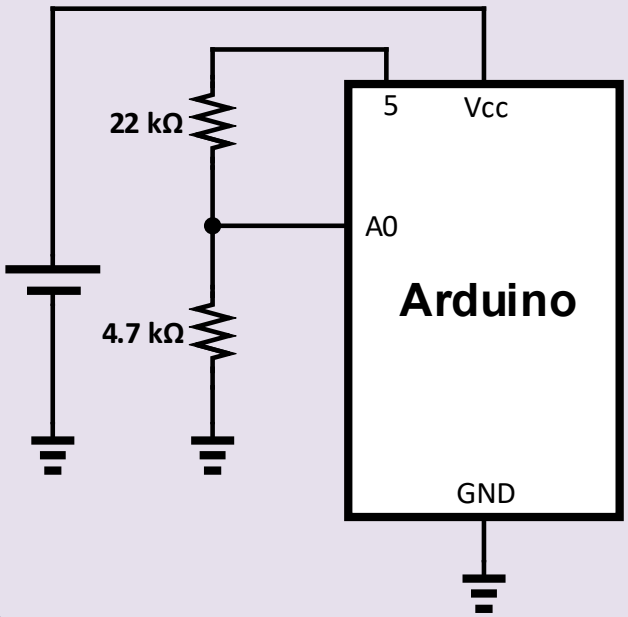
Button: از پایهٔ A2



LED: از پایهٔ 3

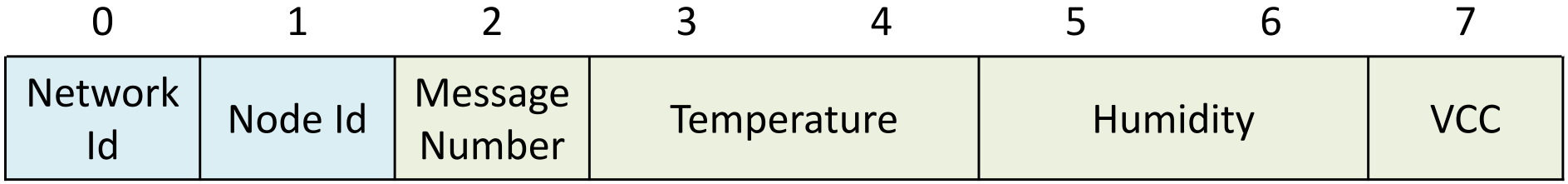


اندازه‌گیری تغذیه: از پایهٔ 5 و A0

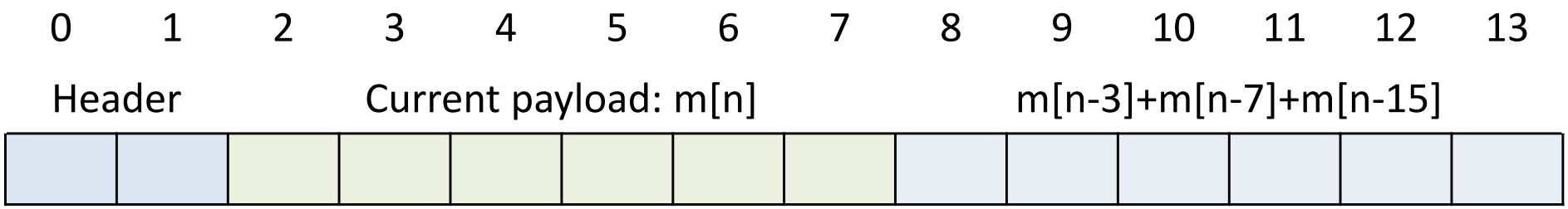


نرم‌افزار

- قالب بسته:



- بازیابی بسته‌های گم شده:



- اجرای برنامه:
  - فایل lora_network_node.ino
  - تغییر خط 22 در فایل utilities.h و قرار دادن یک عدد یکتا
  - اجرای برنامه