المتحكمات الصغرية والنظم المضمنة

محاضرة 5

USB and Serial Communication

د. فادي متوج

- لعل أهم جزء في أي لوحة أردوينو هو قدرتها على البرمجة مباشرة عبر منفذ USB التسلسلي.
- تتيح لنا هذه الميزة برمجة الأردوينو دون الحاجة إلى أي أجهزة خاصة، مثل المبرمجات.
- عادةً، تعتمد المتحكمات الصغرية على قطعة إلكترونية خارجية مخصصة تعمل كمبرمجة تربط بين حاسوبنا والمتحكم الصغري الذي نحاول برمجته. في حالة الأردوينو، هذاه المبرمجة مدمجة بشكل أساسي في اللوحة، بدلاً من أن تكون قطعة إلكترونية خارجية.
- تحتوي شريحة ATMega328P الموجودة في لوحة Arduino Uno على منفذ تسلسلي واحد مدمج. يتضمن هذا المنفذ رجل إرسال (TX) ورجل استقبال (RX) يمكن الوصول إليهما عبر الأرجل الرقمية 0 و 1.

- من أفضل مزايا تصميم الأنظمة المدمجة إمكانية عملها بشكل مستقل عن الحاسوب.
- حتى الآن، كنا معتمدين على الحاسوب لعرض أي نوع من المعلومات من خلال ميزة **التراسل التسلسلي (Serial Monitor)**.
- بإضافة شاشة عرض LCD إلى أردوينو، يُمكننا عرض المعلومات المعقدة (مثل قيم الحساسات، ومعلومات الزمن، والإعدادات، وغيرها) بسهولة أكبر مباشرةً على مشروع أردوينو دون الحاجة إلى الاتصال التسلسلي عبر الحاسوب.
- الأرجل 0 و 1 هي أرجل **متعددة المهام (multiplexed)** مما يعني أنها مكلفة بأكثر من وظيفة واحدة ؛ فهي تتصل بشكل غير مباشر بخطوط الإرسال والاستقبال لكابل USB الخاص بنا بالإضافة لوظيفتها الأساسية كأرجل دخل/خرج رقمي. ومع ذلك، فإن الاتصال التسلسلي و USB ليسا متوافقين بشكل مباشر، لذا يتم استخدام دارة متكاملة (IC) ثانوية موجودة لتسهيل التحويل بينهما.

الاستماع إلى الأردوينو

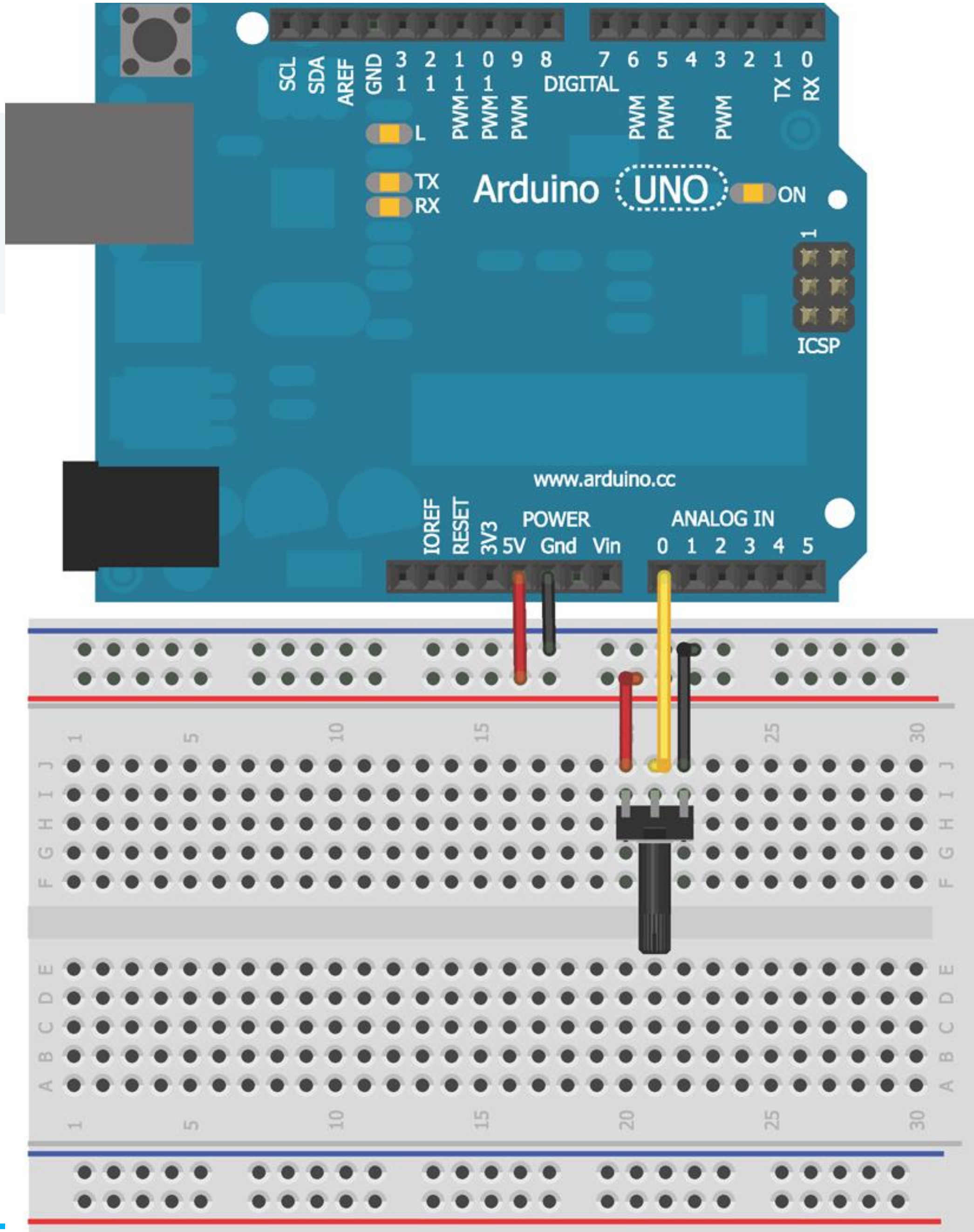
- أبسط وظيفة تسلسلية يمكننا القيام بها مع الأردوينو هي الطباعة إلى الطرفية التسلسلية (serial terminal) للحاسب.
- لطباعة البيانات إلى الطرفية، نحتاج فقط إلى استخدام ثلاث توابع:

➢ `Serial.begin(baud_rate)`

➢ `Serial.print("Message")`

➢ `Serial.println("Message")`

اختبار الطباعة التسلسلية باستخدام مقاومة متغيرة (Potentiometer)

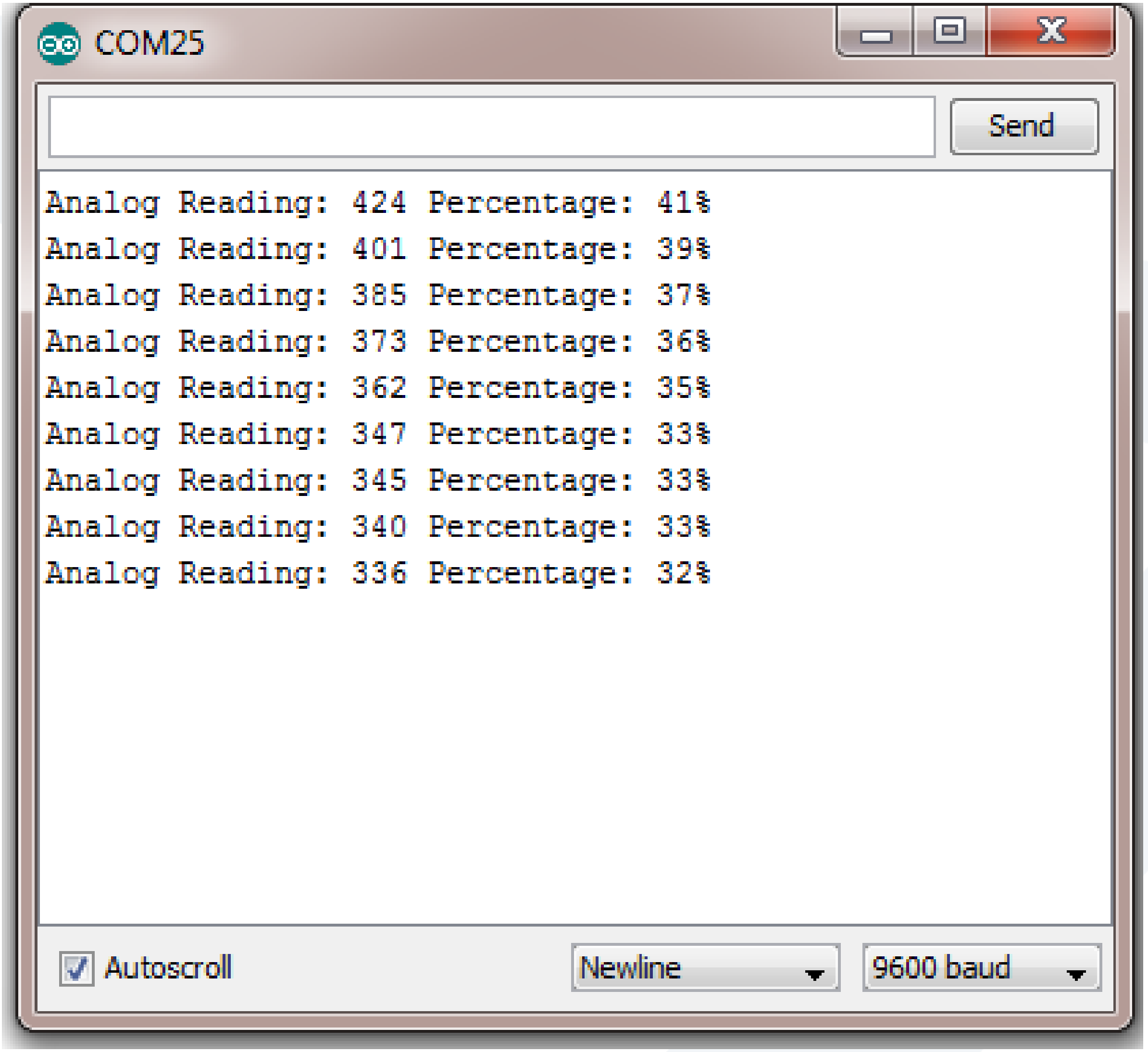


- لتجربة هذه الوظيفة، سنقوم بتوصيل دارة بسيطة تحتوي على مقاومة متغيرة متصلة بالرجل A0 في الأردوينو. سنقرأ قيمة المقاومة المتغيرة ونطبعها كقيمة خام وكقيمة مئوية.

الكود البرمجي للمثال

```
//Simple Serial Printing Test with a Potentiometer
const int POT=0;          //Pot on analog pin 0
void setup()
{
Serial.begin(9600);        //Start serial port with baud = 9600
}
void loop() {
int val = analogRead(POT);                   //Read potentiometer
int per = map(val, 0, 1023, 0, 100);         //Convert to percentage
Serial.print("Analog Reading: ");
Serial.print(val);                           //Print raw analog value
Serial.print(" Percentage: ");
Serial.print(per);                  //Print percentage analog value
Serial.println("%");              //Print % sign and newline
delay(1000);                   //Wait 1 second, then repeat
}
```

خرج المثال


COM25
Send
Analog Reading: 424 Percentage: 41%
Analog Reading: 401 Percentage: 39%
Analog Reading: 385 Percentage: 37%
Analog Reading: 373 Percentage: 36%
Analog Reading: 362 Percentage: 35%
Analog Reading: 347 Percentage: 33%
Analog Reading: 345 Percentage: 33%
Analog Reading: 340 Percentage: 33%
Analog Reading: 336 Percentage: 32%
Autoscroll
Newline
9600 baud

- يجب استدعاء التابع `Serial.begin()` مرة واحدة في بداية البرنامج ضمن تابع setup() لإعداد المنفذ التسلسلي للاتصال.
- بعد القيام بذلك، يمكننا استخدام تابعي `Serial.print()` و `Serial.println()` لكتابة البيانات إلى المنفذ التسلسلي.
- الفرق الوحيد بينهما هو أن `Serial.println()` تجعل النص التالي المطبوع يظهرعلى السطر التالي.

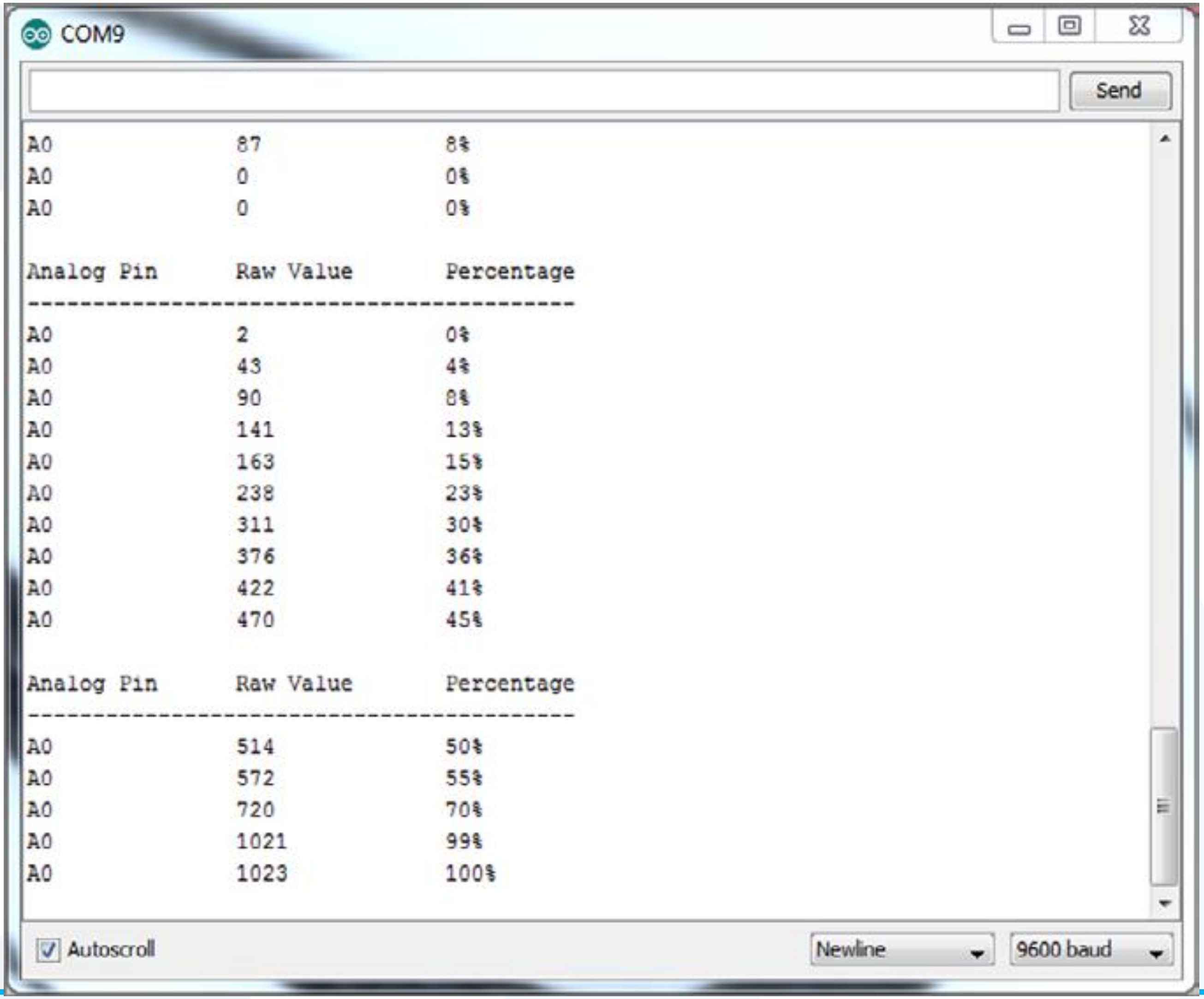
استخدام المحارف الخاصة

- يمكننا إرسال مجموعة متنوعة من "المحارف الخاصة" عبر الاتصال التسلسلي، والتي تسمح لنا بتغيير تنسيق البيانات التسلسلية التي نطبعها.
- نشير إلى هذه الأحرف الخاصة بالشرطة المائلة (\ ) متبوعًا بحرف أمر.
- هناك مجموعة متنوعة من هذه المحارف الخاصة، ولكن أهم اثنين بالنسبة لنا هما محرف الجدولة (tab) ومحرف السطر الجديد (newline) .
- لإدراج محرف جدولة، نضيف **\t** إلى السلسلة النصية.
- لإدراج محرف سطر جديد، نضيف **\n** إلى السلسلة النصية. هذا مفيد بشكل خاص إذا أردنا إدراج سطر جديد في بداية السلسلة النصية، بدلاً من نهايتها كما يفعل تابع Serial.println().
- إذا أردنا، لسبب ما، طباعة **\n** أو **\t** فعليًا في السلسلة النصية، يمكننا القيام بذلك عن طريق طباعة \\n أو \\t على التوالي.

اختبار الطباعة التسلسلية المجدولة باستخدام مقاومة متغيرة

```
const int POT=0;  //Pot on analog pin 0
void setup()
{
Serial.begin(9600);      //Start Serial Port with Baud = 9600
void loop() {
Serial.println("\nAnalog Pin\tRaw Value\tPercentage");
Serial.println("------------------------------------------");
for (int i = 0; i < 10; i++)
{
int val = analogRead(POT);      //Read potentiometer
int per = map(val, 0, 1023, 0, 100);   //Convert to percentage
Serial.print("A0\t\t");
Serial.print(val);
Serial.print("\t\t");
Serial.print(per);          //Print percentage analog value
Serial.println("%");        //Print % sign and newline
delay(1000);                //Wait 1 second, then repeat
} }
```

مخرجات الطرفية التسلسلية



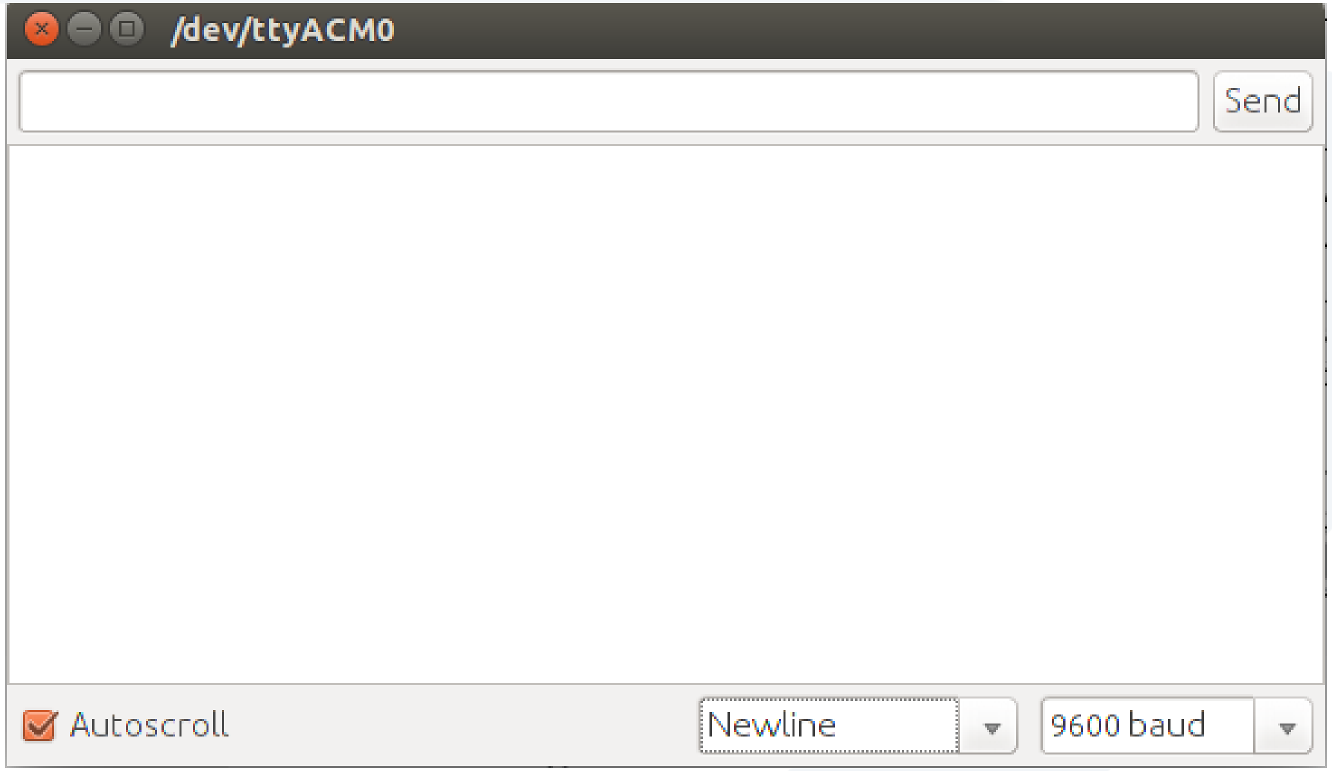
خيارات أنواع البيانات التسلسلية

DATA TYPE	EXAMPLE CODE	SERIAL OUTPUT
Decimal	Serial.println(23);	23
Hexadecimal	Serial.println(23, HEX);	17
Octal	Serial.println(23, OCT)	27
Binary	Serial.println(23, BIN)	00010111

التحدث إلى الأردوينو

- ما فائدة المحادثة مع الأردوينو إذا كانت تسير في اتجاه واحد فقط؟ الآن بعد أن فهمنا كيف يرسل الأردوينو البيانات إلى حاسوبنا، سوف نناقش كيفية إرسال الأوامر من حاسوبنا إلى الأردوينو.

- يحتوي مراقب المنفذ التسلسلي (serial monitor) في بيئة تطوير الأردوينو (Arduino IDE) على حقل إدخال نصي في الأعلى، وقائمة منسدلة في الأسفل.
- تحدد القائمة المنسدلة ما إذا كان سيتم إلحاق أي شيء بنهاية أوامرنا عند إرسالها إلى الأردوينو.
- عند تحديد Newline، يتم فقط إلحاق \n بنهاية أي شيء نرسله من حقل إدخال النص في أعلى نافذة مراقب المنفذ التسلسلي.
- على عكس بعض برامج الطرفية الأخرى، يرسل مراقب المنفذ التسلسلي في Arduino IDE سلسلة الأوامر بأكملها مرة واحدة عند الضغط على مفتاح Enter أو زر الإرسال. هذا على النقيض من طرفيات تسلسلية أخرى مثل PuTTy التي ترسل المحارف أثناء كتابتها.



قراءة المعلومات من الحاسب أو جهاز تسلسلي آخر

- كيف نستخدم مراقب المنفذ التسلسلي في Arduino IDE لإرسال الأوامر يدويًا إلى الأردوينو؟
- كيف نرسل قيم أوامر متعددة مرة واحدة؟
- يحتوي منفذ الأردوينو التسلسلي على **مخزن مؤقت (buffer)** بعبارة أخرى، يمكننا إرسال عدة بايتات من البيانات مرة واحدة وسيقوم الأردوينو بوضعها في قائمة انتظار ومعالجتها بالترتيب بناءً على محتوى البرنامج الخاص بنا.
- لا داعي للقلق بشأن إرسال البيانات بشكل أسرع من زمن دورة ()loop، ولكن يجب أن نقلق بشأن إرسال كمية كبيرة من البيانات تؤدي إلى تجاوز سعة المخزن المؤقت وفقدان المعلومات.

إخبار الأردوينو بترديد البيانات الواردة (Echo)

- أبسط ما يمكننا فعله هو جعل الأردوينو يردد كل ما نرسله إليه. لتحقيق ذلك، يحتاج الأردوينو بشكل أساسي فقط إلى مراقبة مخزنه المؤقت للإدخال التسلسلي وطباعة أي حرف يتلقاه.
- للقيام بذلك، نحتاج إلى استخدام أمرين:
- **Serial.available()** : تُرجع عدد الأحرف (أو البايتات) المخزنة حاليًا في المخزن المؤقت للإدخال التسلسلي للأردوينو. كلما كان هذا العدد أكبر من صفر، سنقرأ المحارف ونرددها مرة أخرى إلى الحاسب.
- **Serial.read()** : تقرأ وتُرجع المحرف التالي المتاح في المخزن المؤقت.
- كل استدعاء لتابع Serial.read() سيعيد بايت واحد فقط، لذا نحتاج إلى تنفيذها طالما أن Serial.available() تُرجع قيمة أكبر من صفر.
- في كل مرة يلتقط فيها تابع Serial.read() بايتًا، تتم إزالة هذا البايت من المخزن المؤقت، بحيث يكون البايت التالي جاهزًا للقراءة.

```
//Echo every character
char data;     //Holds incoming character
void setup()
{
Serial.begin(9600);     //Serial Port at 9600 baud
}
void loop()
{
        //Only print when data is received
if (Serial.available() > 0)
    {
       data = Serial.read();   //Read byte of data
       Serial.print(data);     //Print byte of data
    }
}
```

- عندما نرسل محرفًا أبجديًا رقميًا عبر مراقب المنفذ التسلسلي، فإننا لا نمرر فعليًا "5" أو "A"نحن نرسل **بايتًا** يفسره الحاسب كمحرف.
- في حالة الاتصال التسلسلي، يتم استخدام مجموعة محارف ASCII لتمثيل جميع المحارف والأرقام والرموز والأوامر الخاصة التي قد نرغب في إرسالها.

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

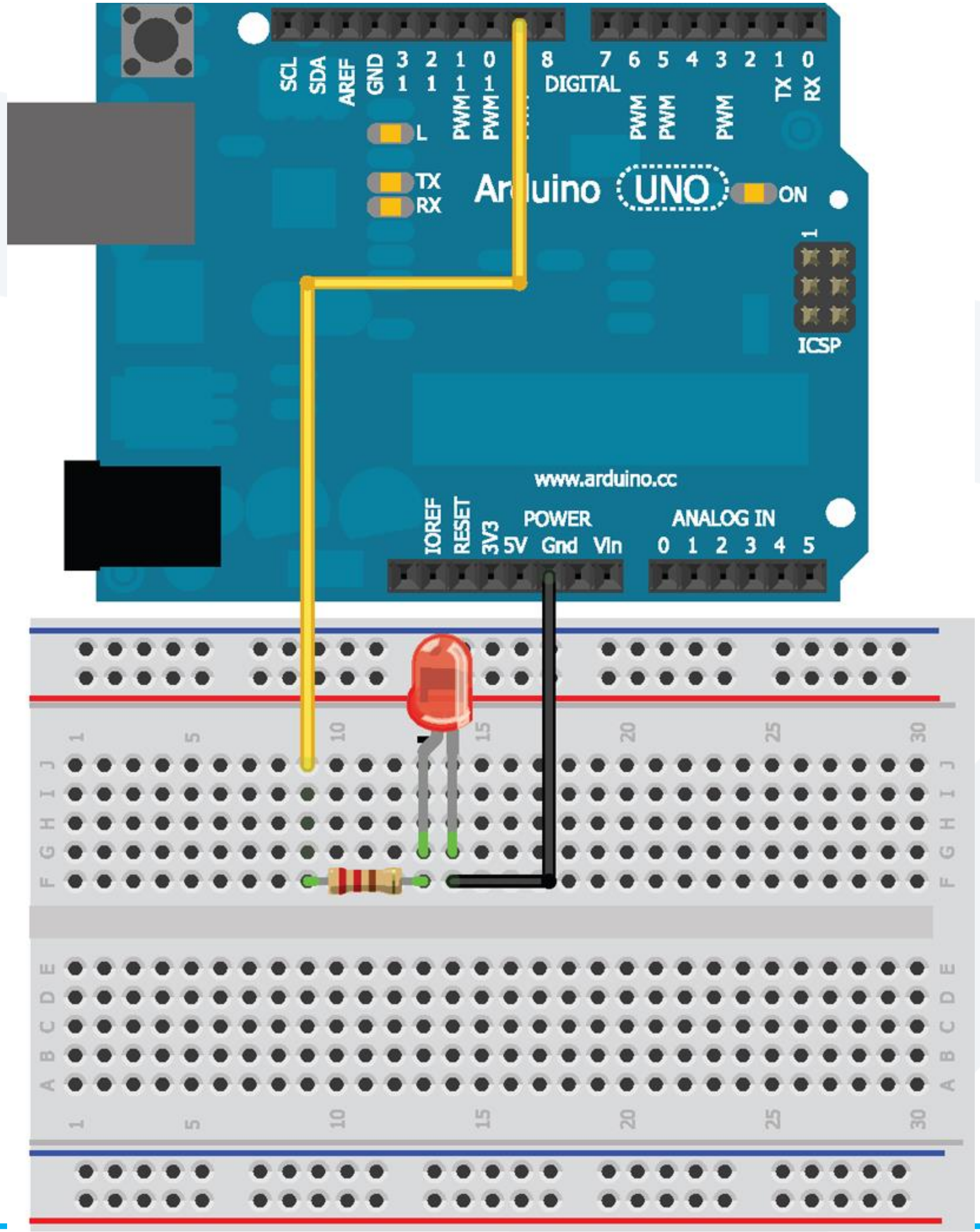
ASCII table

- عند قراءة قيمة أرسلناها من الحاسب، يجب قراءة البيانات كنوع **char**. حتى لو كنا نتوقع إرسال أرقام فقط من الطرفية التسلسلية، فنحن بحاجة إلى قراءة القيم كمحرف أولاً، ثم تحويلها حسب الضرورة.
- على سبيل المثال، إذا قمنا بتعديل الكود لتعريف data كنوع **int**، فإن إرسال قيمة 5 سيعيد 53 إلى مراقب المنفذ التسلسلي لأن التمثيل العشري للمحرف '5' هو الرقم 53.

- غالبًا ما نرغب في إرسال قيم رقمية إلى الأردوينو. فكيف نفعل ذلك؟
- يمكننا القيام بذلك بعدة طرق:
  - أولاً، يمكننا ببساطة مقارنة الأحرف مباشرة. إذا أردنا تشغيل LED عند إرسال '1'، يمكننا مقارنة قيم الأحرف هكذا: if (Serial.read() == '1').
  - الخيار الثاني هو تحويل كل بايت وارد إلى عدد صحيح عن طريق طرح قيمة الحرف الصفري، هكذا: int val = Serial.read() - '0'. ومع ذلك، هذا لا يعمل بشكل جيد إذا كنا نعتزم إرسال أرقام أكبر من 9، لأنها ستكون متعددة الأرقام. للتعامل مع هذا، تتضمن بيئة تطوير الأردوينو تابع مفيد يسمى parseInt() يحاول استخراج الأعداد الصحيحة من تدفق البيانات التسلسلية.

مثال 1: إرسال محارف مفردة للتحكم في LED

- في هذا المثال، نكتب برنامجًا يستخدم مقارنة محارف بسيطة للتحكم في LED عند إرسال محرف واحد فقط، يكون الشيء الأسهل هو إجراء مقارنة محارف بسيطة في جملة if.
- سنرسل '1' لإضاءة الـ LED، و '0' لإطفائه.
- نقوم بتوصيل الـ LED بالرجل 9 في الأردوينو.
- في كل مرة يتم فيها إضافة محرف إلى المخزن المؤقت، تتم مقارنته بـ '0' أو '1'، ويتم اتخاذ الإجراء المناسب.



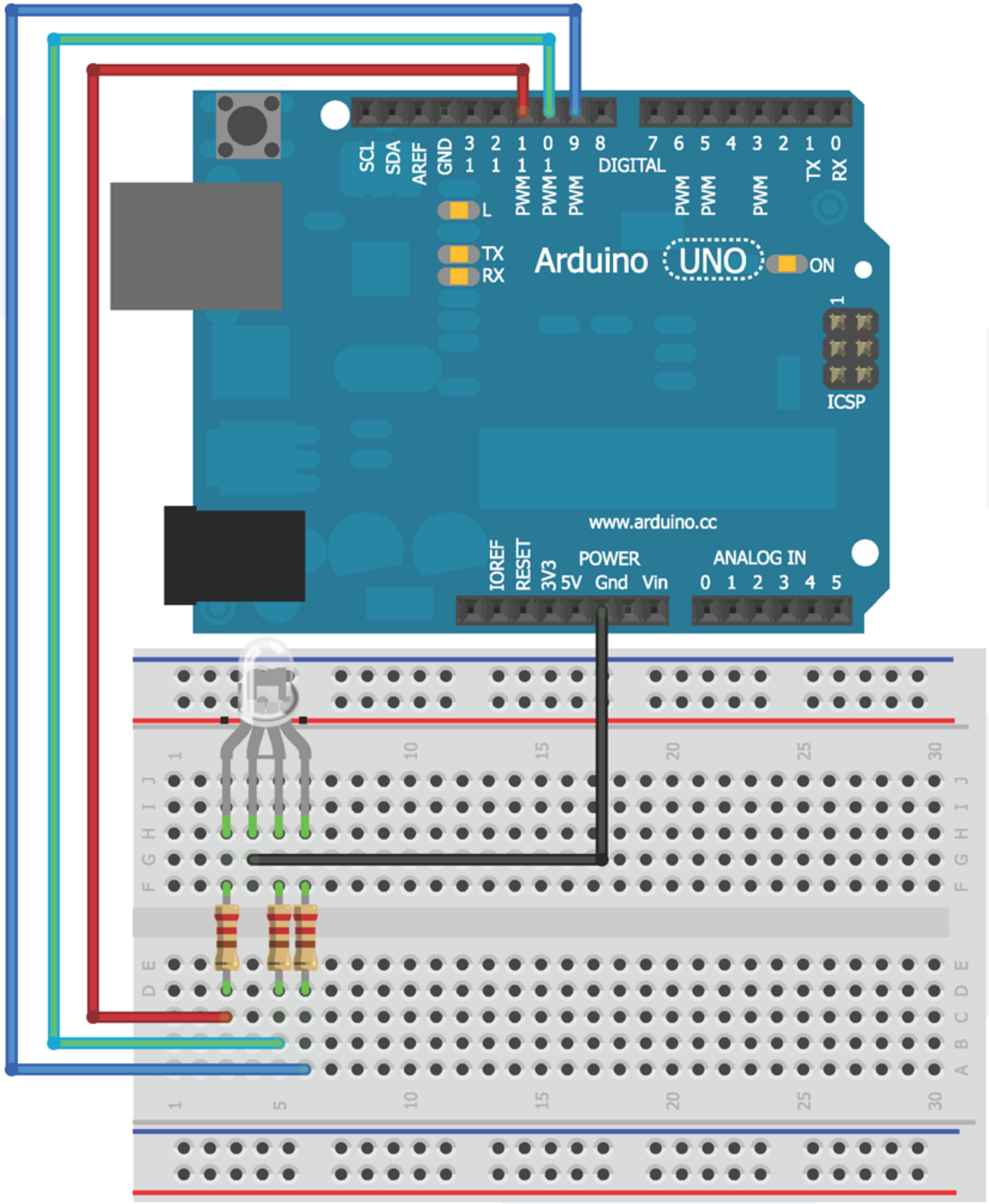
جَـامعة
المَـنارة
MANARA UNIVERSITY

```
//Single Character Control of an LED
const int LED=9;
char data; //Holds incoming character
void setup()
{
Serial.begin(9600); //Serial Port at 9600 baud
pinMode(LED, OUTPUT);
}
void loop()
{
//Only act when data is available in the buffer
if (Serial.available() > 0)
{
data = Serial.read(); //Read byte of data
//Turn LED on
if (data == '1')
{
digitalWrite(LED, HIGH);
Serial.println("LED ON");
}
//Turn LED off
else if (data == '0')
{
digitalWrite(LED, LOW);
Serial.println("LED OFF");
} } }
```

- في هذا المثال، يتم استخدام جملة else if بدلاً من جملة else بسيطة.
- نظرًا لأن الطرفية الخاصة بنا مضبوطة أيضًا لإرسال محرف سطر جديد(\n) مع كل إرسال، فمن الضروري قراءة هذه المحارف من المخزن المؤقت.
- سيقرأ تابع Serial.read() محرف السطر الجديد الذي لا يساوي '0' أو '1'، وسيتم الكتابة فوقه في المرة التالية التي يتم فيها استدعاء Serial.read()
- إذا تم استخدام جملة else فقط، فإن كلاً من '0' و \n سيؤديان إلى إطفاء الـ LED. حتى عند إرسال '1'، سينطفئ الـ LED فورًا مرة أخرى عند تلقي \n

مثال 2: إرسال قوائم من القيم للتحكم في LED ثلاثي الألوان (RGB)

- إرسال محرف أمر واحد مناسب للتحكم في رجل رقمية واحدة، ولكن ماذا لو أردنا التحكم في أجهزة متعددة؟
- يتضمن هذا المثال إرسال قيم متعددة مفصولة بفواصل للتحكم في أجهزة متعددة في وقت واحد.
- لتسهيل اختبار ذلك، نقوم بتوصيل LED ثلاثي الألوان ذي مهبط مشترك (common cathode RGB LED).


Arduino UNO
DIGITAL
PWM
TX
RX
L
ON
ICSP
www.arduino.cc
POWER
ANALOG IN
IOREF
RESET
3V3
5V
Gnd
Vin

- للتحكم في LED ثلاثي الألوان هذا، نرسل ثلاث قيم منفصلة بحجم 8 بت (0-255) لضبط سطوع كل لون من ألوان الـ LED
- على سبيل المثال، لضبط جميع الألوان على السطوع الكامل، نرسل "255,255,255".
- هذا يطرح بعض التحديات:
  - نحتاج إلى التمييز بين الأرقام والفواصل.
  - نحتاج إلى تحويل هذه السلسلة من المحارف إلى أعداد صحيحة يمكننا تمريرها إلى تابع analogWrite()
  - نحتاج إلى أن نكون قادرين على التعامل مع حقيقة أن القيم يمكن أن تكون مكونة من رقم واحد أو اثنين أو ثلاثة أرقام.

- تستخدم بيئة تطوير الأردوينو تابع مفيد جدًا لتحديد واستخراج الأعداد الصحيحة: Serial.parseInt()
- يرجع تابع Serial.parseInt() أول رقم صحيح صالح من المخزن المؤقت التسلسلي. يتم تخطي الأحرف التي ليست أعدادًا صحيحة.
- يتم إنهاء تابع Serial.parseInt() بأول محرف ليس رقمًا.
- ينتظر كل استدعاء لهذا التابع حتى يدخل قيمة غير رقمية إلى المخزن المؤقت التسلسلي، ويحول الأرقام السابقة إلى عدد صحيح.
- تتم قراءة أول قيمتين عند اكتشاف الفواصل، وتتم قراءة القيمة الأخيرة عند اكتشاف محرف السطر الجديد.

```
//Define LED pins
const int RED =11;
const int GREEN =10;
const int BLUE =9;
//Variables for RGB levels
int rval = 0;
int gval = 0;
int bval = 0;
void setup()
{
Serial.begin(9600); //Serial Port at 9600 baud
//Set pins as outputs
pinMode(RED, OUTPUT);
pinMode(GREEN, OUTPUT);
pinMode(BLUE, OUTPUT);
}
```

```
void loop()
{
//Keep working as long as data is in the buffer
while (Serial.available() > 0)
{
rval = Serial.parseInt();        //First valid integer
gval = Serial.parseInt();       //Second valid integer
bval = Serial.parseInt();       //Third valid integer

if (Serial.read() == '\n') //Done transmitting
{
//set LED
analogWrite(RED, rval);
analogWrite(GREEN, gval);
analogWrite(BLUE, bval);
}
}
}
```

- يستمر البرنامج في البحث عن ثلاث قيم عددية صحيحة حتى يتم اكتشاف سطر جديد. بمجرد حدوث ذلك، تُستخدم القيم التي تمت قراءتها لضبط درجة سطوع مصابيح الـ LED.
- لاستخدام هذا، نفتح شاشة المراقبة التسلسلية (serial monitor) وندخل ثلاث قيم بين 0 و 255 مفصولة بفاصلة، مثل "200,30,180".
- يمكننا تجربة مزج جميع أنواع الألوان بهذه الطريقة.