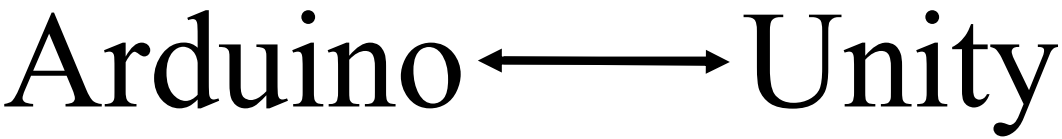
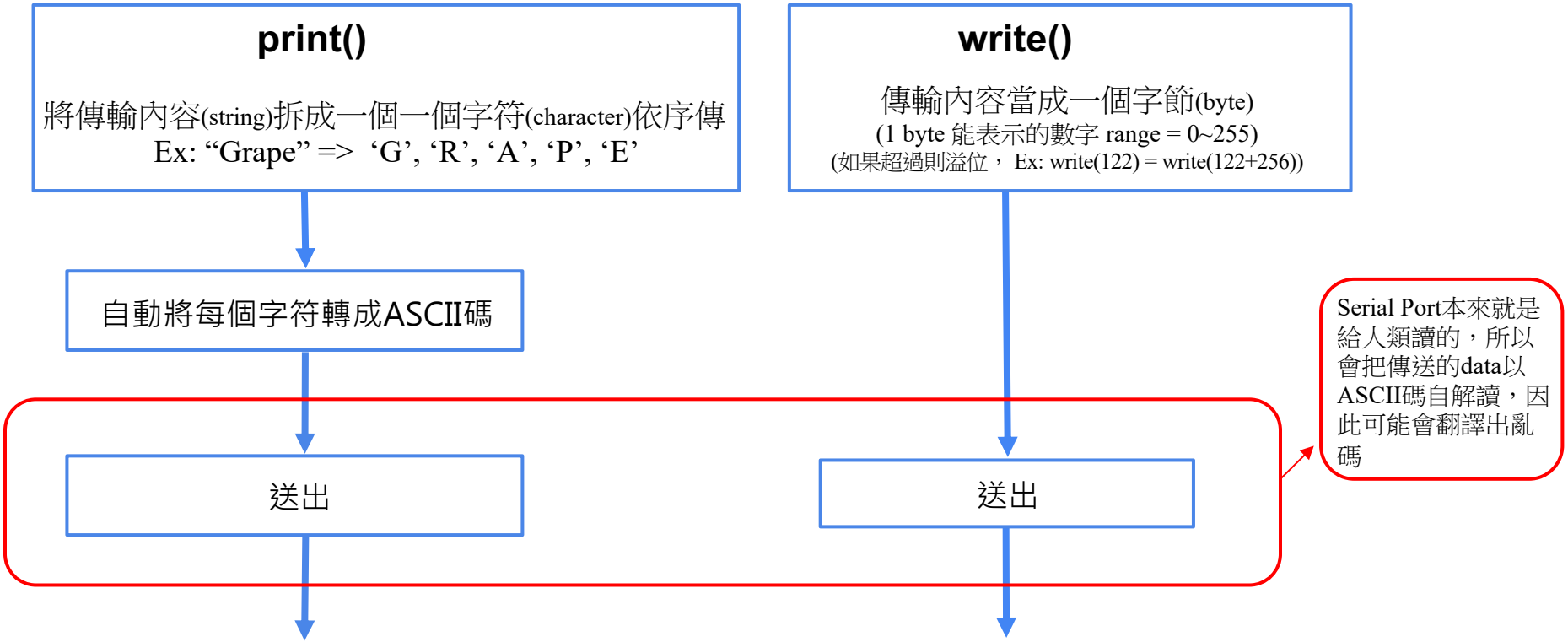
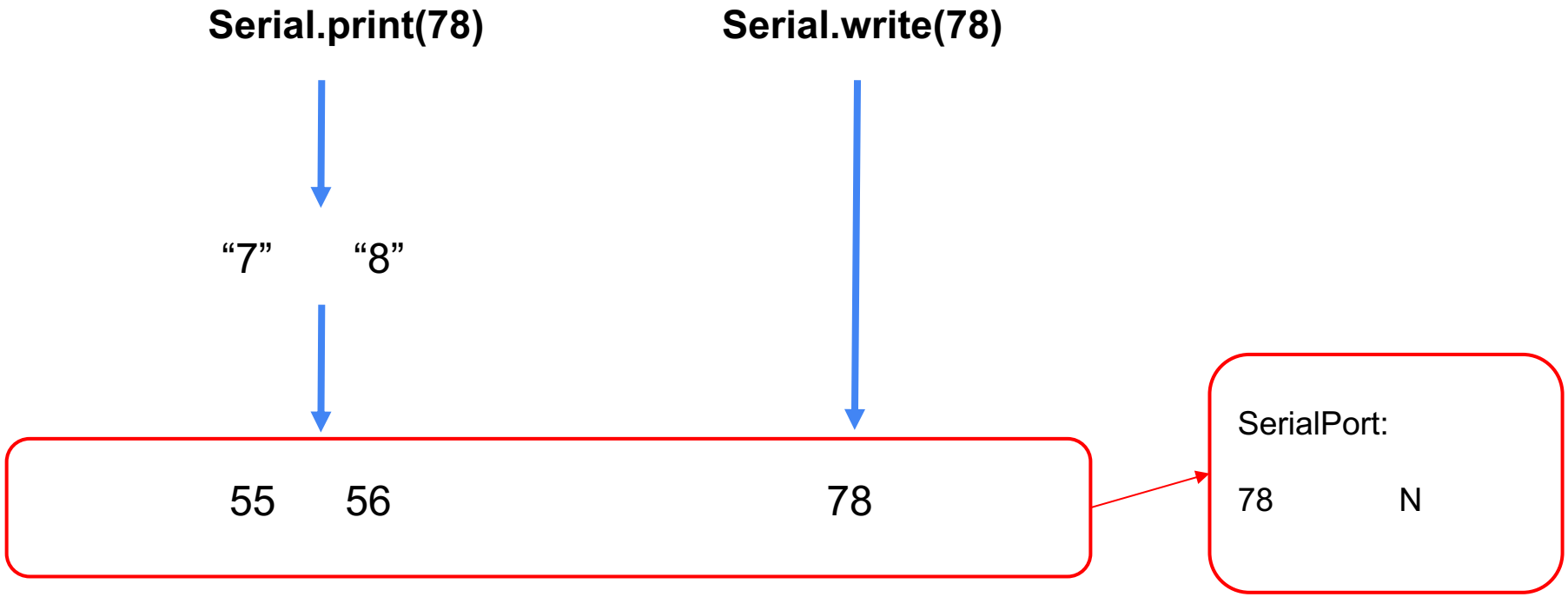

Arduino
Unity

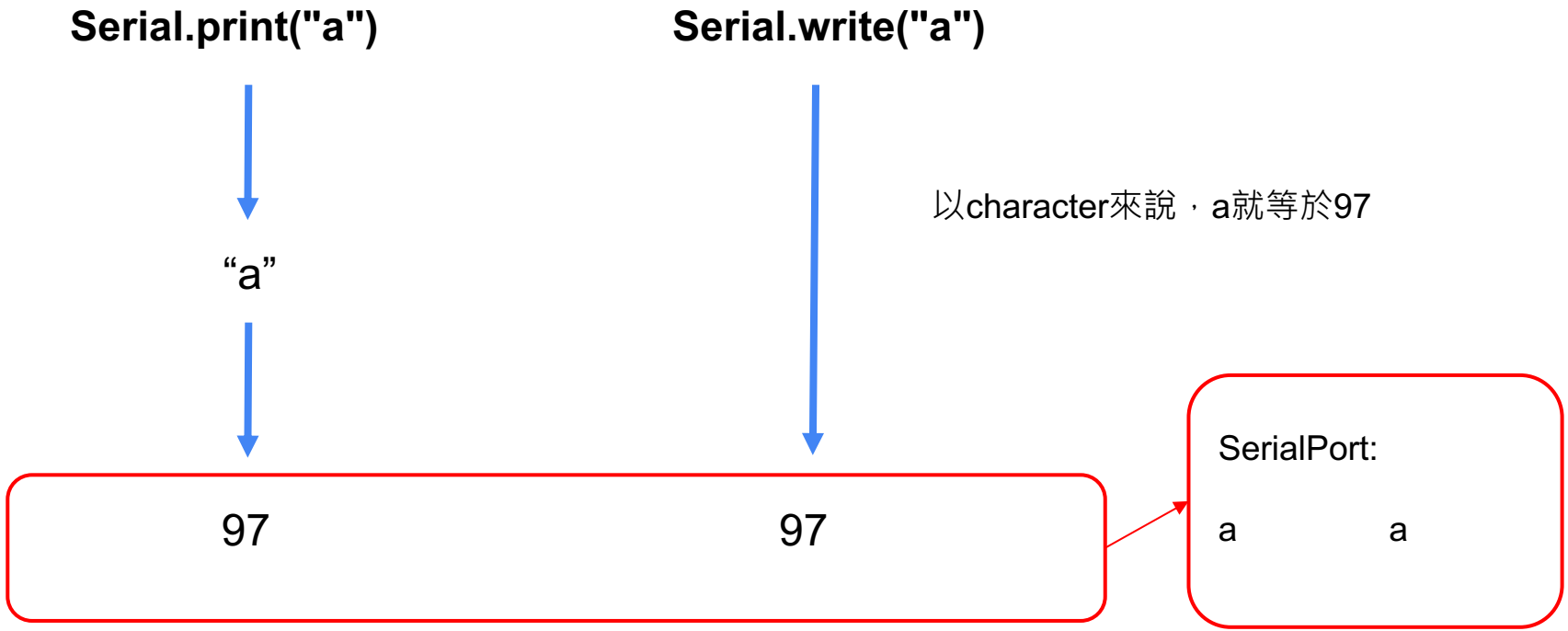


優點: 可供人類讀取，自動轉換成ASCII碼
缺點: 步驟較多，較慢，可能會丟包

優點: 傳送data穩定也較快
缺點: 傳送一些數值形態時不能監控


Serial.print(78)
Serial.write(78)
“7”
“8”
55
56
78
SerialPort:
78
N

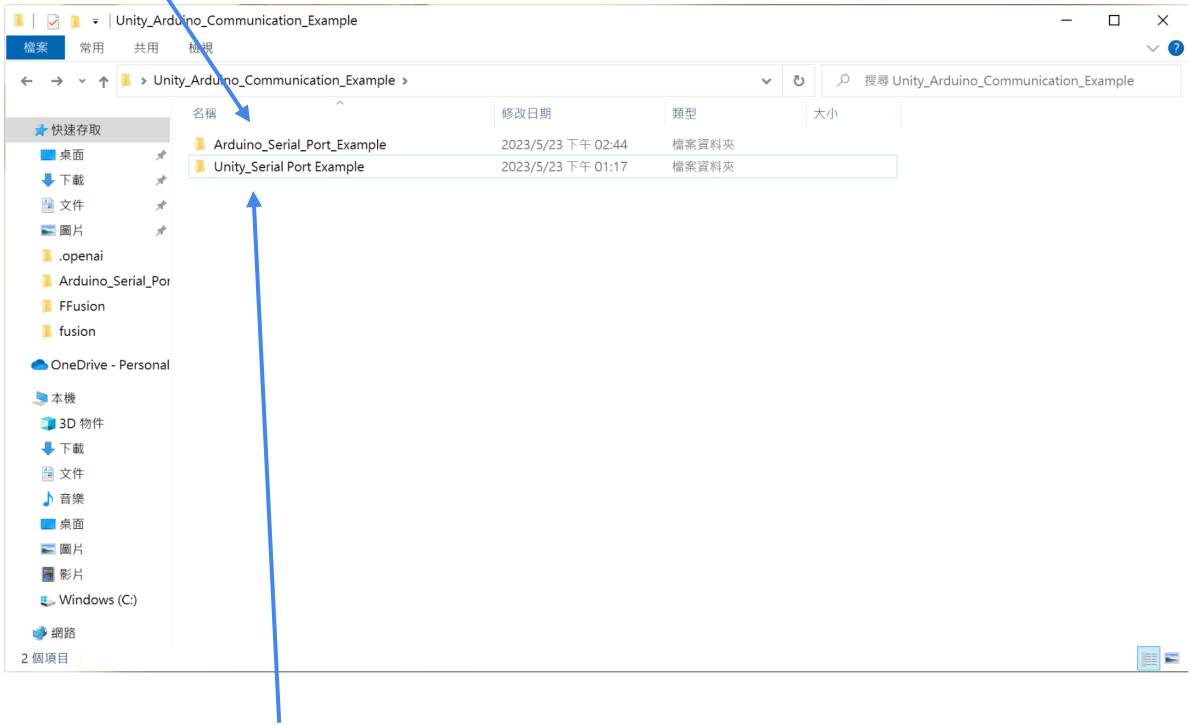
Dec	Hx	Oct	Char	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr
0	0	000	NUL (null)	32	20	040	Space	64	40	100	@	96	60	140	`
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT (end of transmission)	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL


Serial.print("a")
Serial.write("a")
以character來說，a就等於97
“a”
97
97
SerialPort:
a
a

1. Arduino ⟶ Unity
2. Arduino ⟵ Unity
3. Arduino ⟷ Unity

Arduino ⟶ Unity

1. Arduino燒錄草稿



2. 用Unity Hub 開啟此專案

Arduino Send

ArduinoUnityCommunication

```
* Arduino的print() 是把傳輸內容拆成一個一個字符依序傳( 會先把字符轉ascii碼，再
* Arduino的write() 是把傳輸內容當成位元組( 一次傳1byte的range=0~255 ) ( 內容

* 因為write range只有1 byte = 8 bit range=0~255 若給超過會溢位 例如write(1

*/


// 用來 Read data
const int receiveBufferSize = 200;
char receive[receiveBufferSize];


// 用來 Send data
char temp[] = "23C";

void setup() {

  Serial.begin( 9600 );

}

void loop() {

  sendData(temp);
  // readData();

}

void sendData(char dataToSend[]){

  Serial.write(dataToSend, strlen (dataToSend));
  delay(100);

}

void readData(){

  if (Serial.available() > 0) {

    int rlen = Serial.readBytes(receive, receiveBufferSize);

    sendData(receive);
  }

}
```

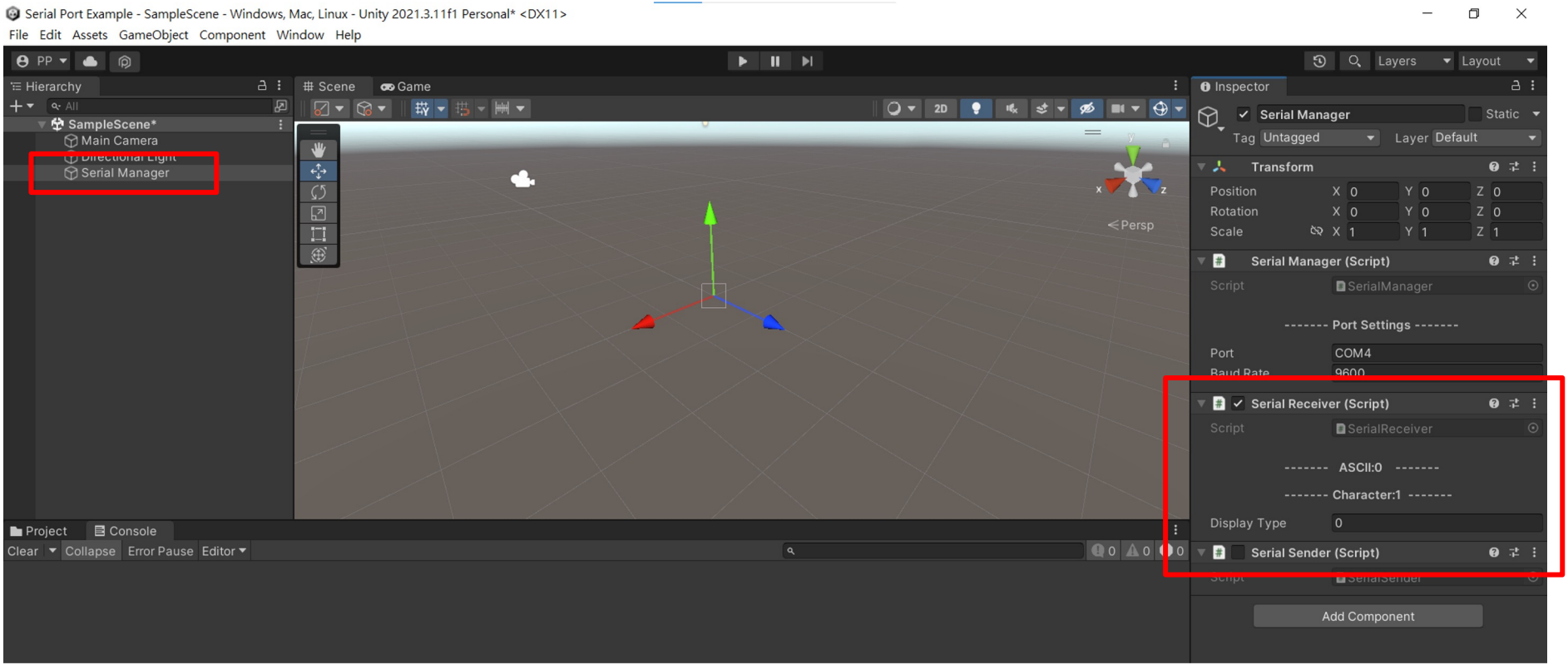
以char array的形式宣告要傳給Unity的data

只要sendData()

可以直接給write fuction一個char array跟要傳送的character數目，在這邊就是整個array的大小，就會依序傳

Unity Receive

在Serial Manager 物件看到有三個script，下面兩個只要activate Serial Receiver就好



```
using System;
using UnityEngine;

0 references
public class SerialReceiver : MonoBehaviour {

    [SpaceAttribute(10)]
    [HeaderAttribute("                    -------   ASCII:0    ------- ")]
    [HeaderAttribute("                    ------- Character:1  ------- ")]
    [SpaceAttribute(10)]
    1 reference
    public int DisplayType;


    2 references
    private SerialManager serialmanager;
    2 references
    private int data;

    0 references
    private void Awake () {

        serialmanager = GetComponent<SerialManager>();

    }

    0 references
    void Update() {

        serialReceive();

    }

    1 reference
    private void serialReceive() {

        try {

            readOneByteAtATime();

        }
        catch {

        }

    }
    1 reference
    private void readOneByteAtATime() {

        data = serialmanager.serialPort.ReadByte();
        displayStyle(DisplayType, data);

    }

    1 reference
    private void displayStyle(int displayType, int data) {

        if (displayType == 0) {

            Debug.Log(data);

        }
        else if (displayType == 1) {

            char c = Convert.ToChar(data);
            Debug.Log(c);

        }

    }
}
```

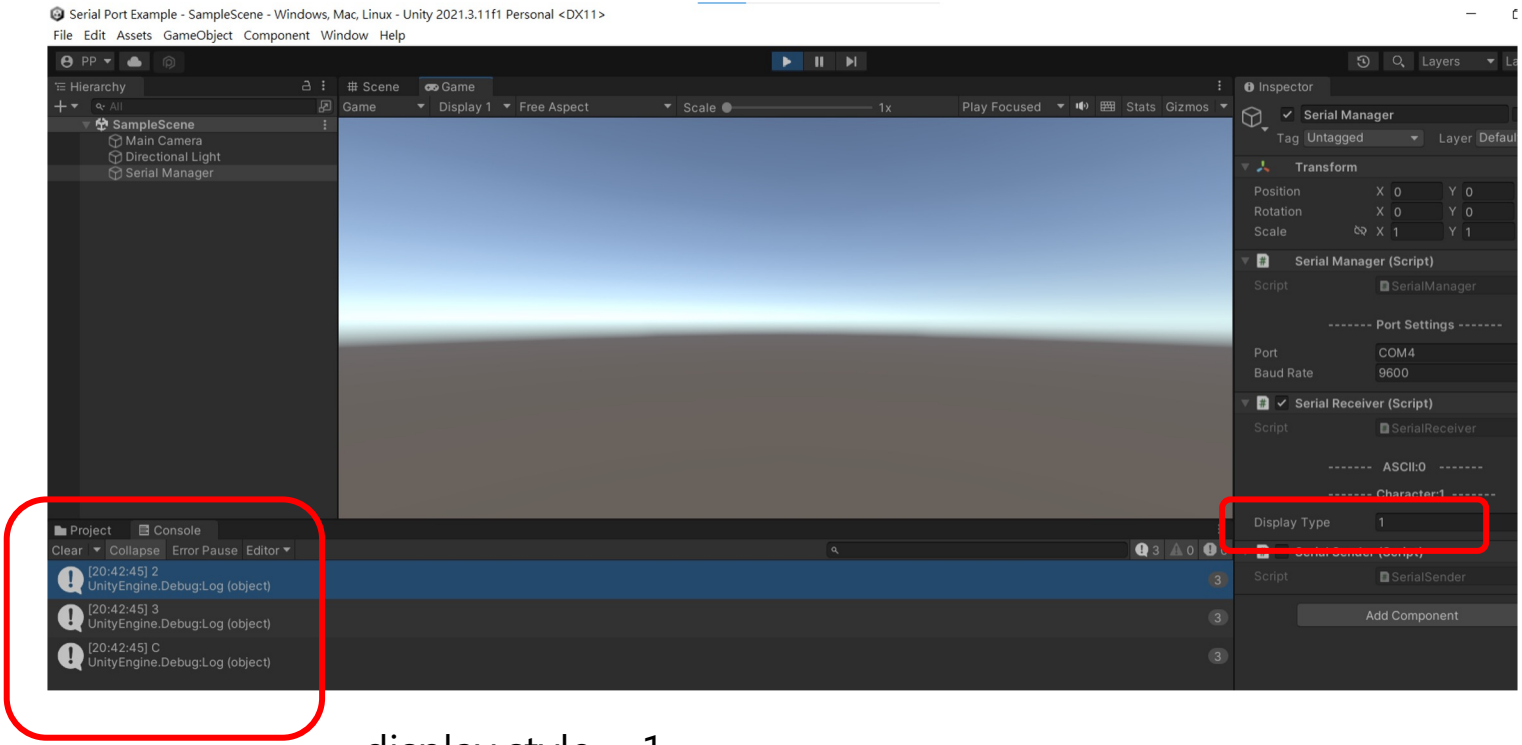
讀一個byte的data

displayStyle() 這個function只是展示真的傳來的是ASCII碼

displayType = 0 就不翻譯，顯示原始ASCII碼

displayType = 1 就翻譯成可讀的 character

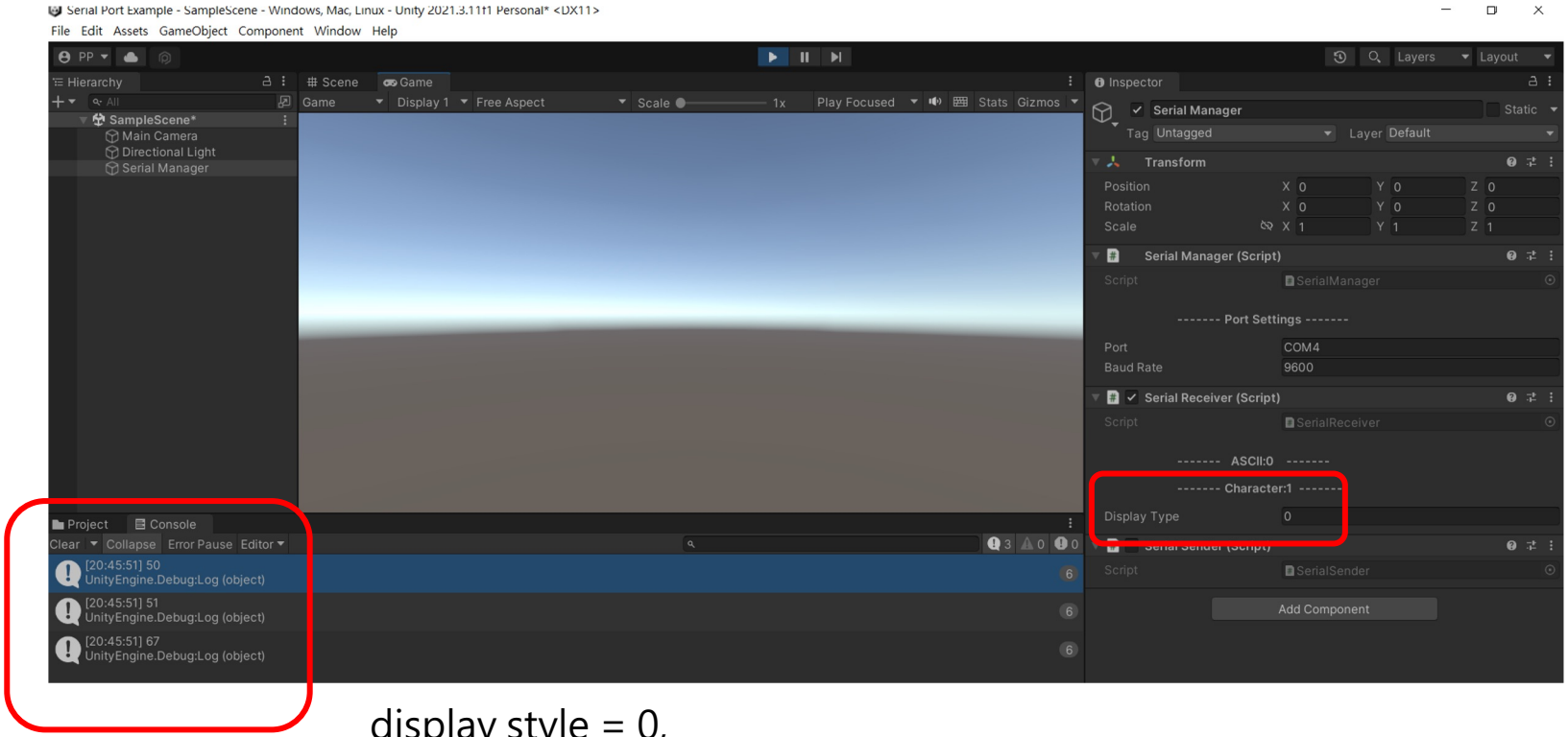
Unity部分: Unity Serial Receiver



display style = 1,

Console可以看到Arduino傳來的23c

Unity部分: Unity Serial Receiver

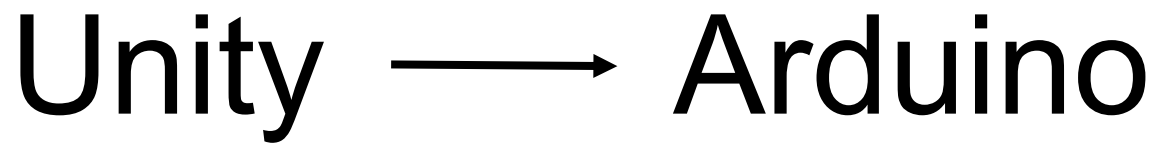


display style = 0,

Console可以看到Arduino傳來的50 51 67

Unity部分: Unity Serial Receiver

Dec	Hx	Oct	Char	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr	Dec	Hx	Oct	Chr
0	0	000	NUL (null)	32	20	040	Space	64	40	100	@	96	60	140	`
1	1	001	SOH (start of heading)	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX (start of text)	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX (end of text)	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT (end of transmission)	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ (enquiry)	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK (acknowledge)	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL (bell)	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS (backspace)	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB (horizontal tab)	41	29	051	)	73	49	111	I	105	69	151	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT (vertical tab)	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR (carriage return)	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	E	016	SO (shift out)	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI (shift in)	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE (data link escape)	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1 (device control 1)	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2 (device control 2)	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3 (device control 3)	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4 (device control 4)	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN (synchronous idle)	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB (end of trans. block)	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN (cancel)	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM (end of medium)	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB (substitute)	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC (escape)	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS (file separator)	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS (group separator)	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS (record separator)	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US (unit separator)	63	3F	077	?	95	5F	137	_	127	7F	177	DEL


Unity
Arduino

Arduino Receive

ArduinoUnityCommunication

```
* Arduino的write() 是把傳輸內容當成位元組( 一次傳1byte的ran

* 因為write range只有1 byte = 8 bit range=0~255 若給超過(

*/


// 用來 Read data
const int receiveBufferSize = 200;
char receive[receiveBufferSize];


// 用來 Send data
char temp[] = "23C";

void setup() {

  Serial.begin( 9600 );

}

void loop() {

  // sendData(temp);
  readData();

}

void sendData(char dataToSend[]){

  Serial.write(dataToSend, strlen (dataToSend));
  // delay(100);

}

void readData(){

  if (Serial.available() > 0) {

    Serial.readBytes(receive, receiveBufferSize);

    sendData(receive);

  }

}
```

宣告一個size為200的char array作為buffer

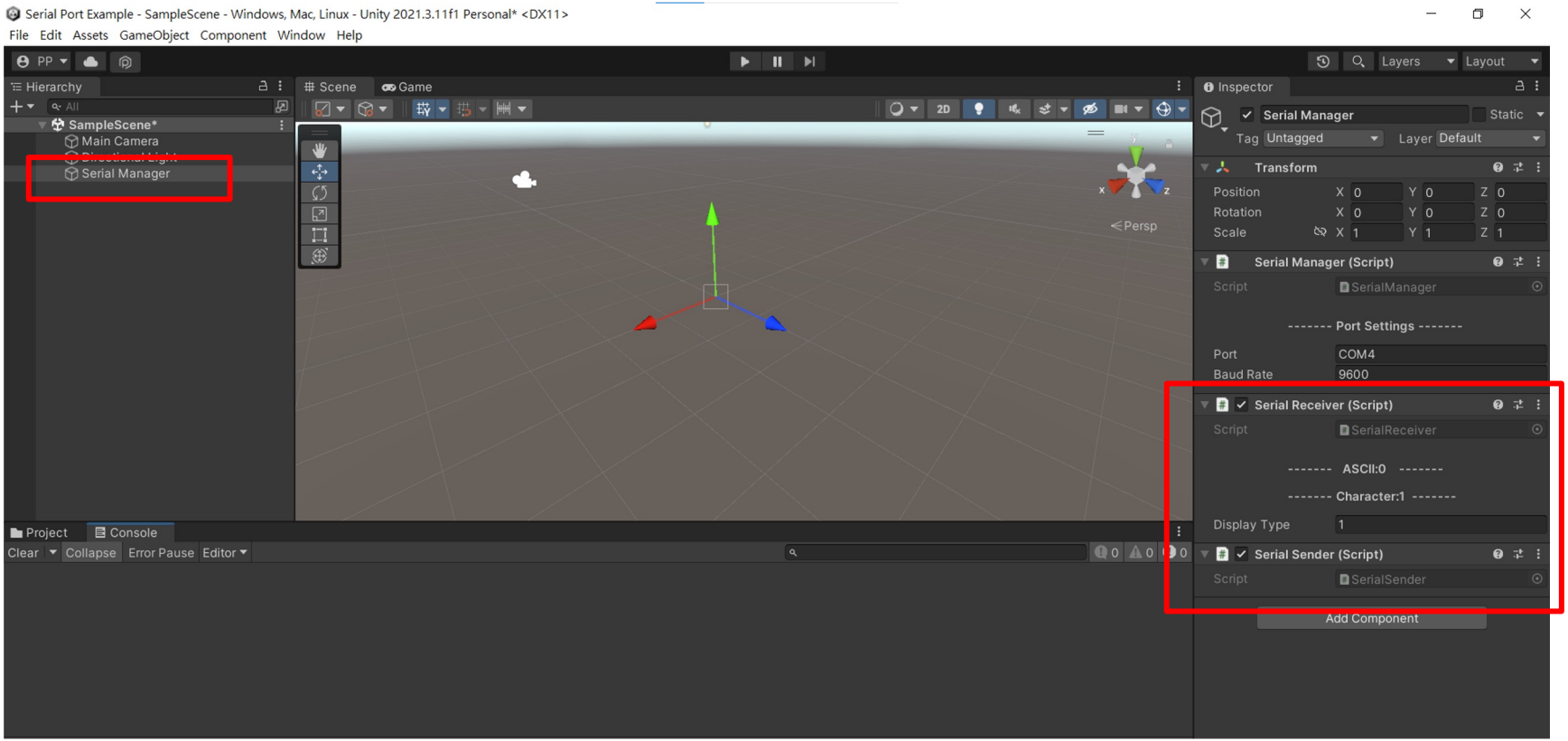
只要readData()

把 delay 註解掉，收到後可以及時回傳

read the incoming bytes，並存進buffer，然後用前面的send function 回傳給Unity檢查

Unity Send and Receive

在Serial Manager 物件的三個script，下面兩個都要activate，因為要接收Arduino的回傳來檢查



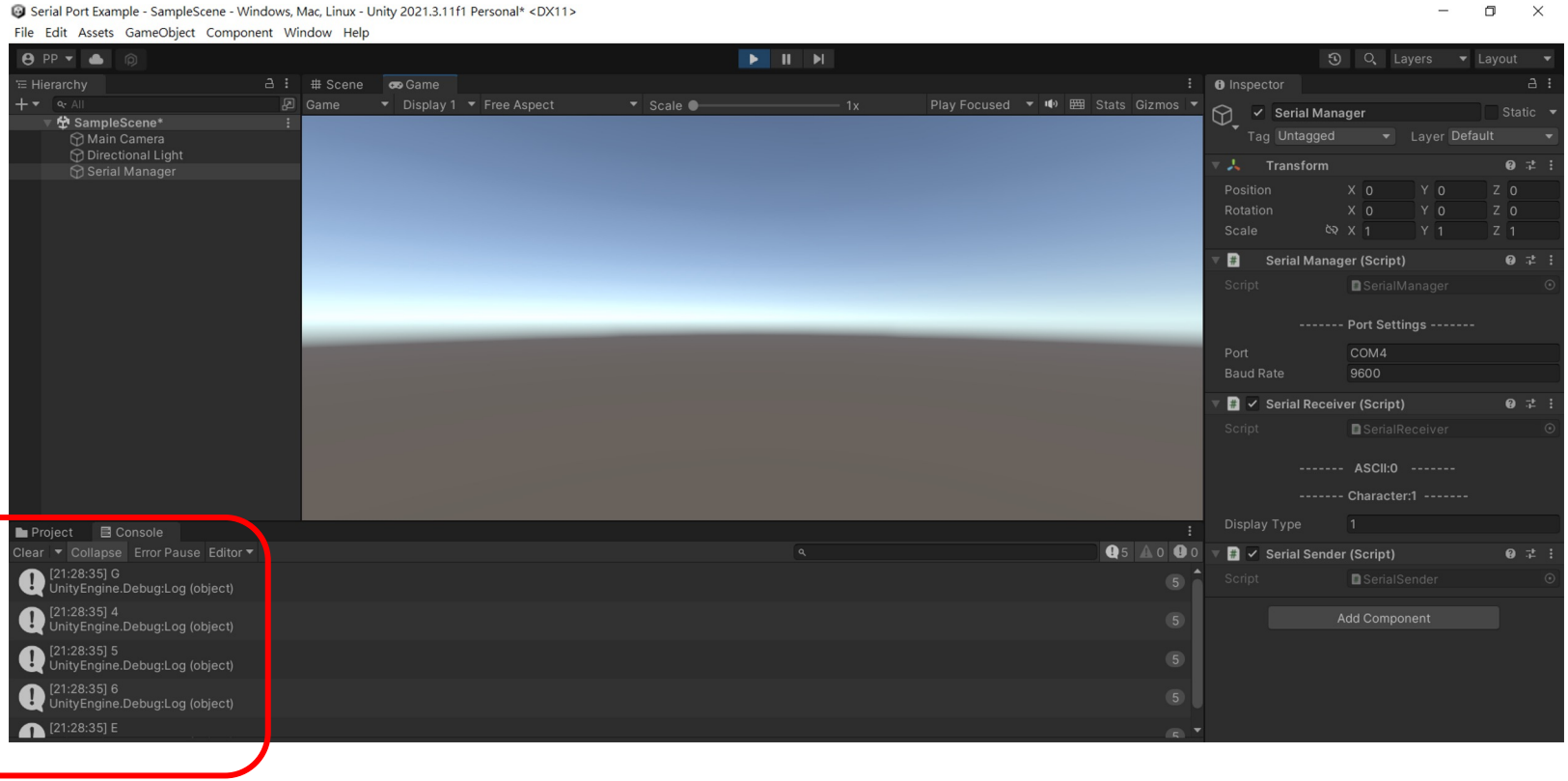
Unity Send

```
/*
Unity send + Arduino read example

* 因為Unity send時不能開Arduino serial port monitor檢查，所以這邊會回傳收到的data給unity供檢查

*/
using System.Text;
using UnityEngine;
using System.Collections;

0 references
public class SerialSender : MonoBehaviour {

    2 references
    private SerialManager serialmanager;
    1 reference
    private string TextToSend = "G456E";

    0 references
    private void Awake () {

        serialmanager = GetComponent<SerialManager>();

    }

    0 references
    private void Start () {

        StartCoroutine(ExampleCoroutine(TextToSend));

    }

    1 reference
    IEnumerator ExampleCoroutine(string text) {

        while(true) {

            serialmanager.serialPort.Write(Encoding.ASCII.GetBytes(text), 0, Encoding.ASCII.GetBytes(text).Length);

            // yield return null;
            yield return new WaitForSeconds(1);         // delay 1秒

        }

    }

}
```

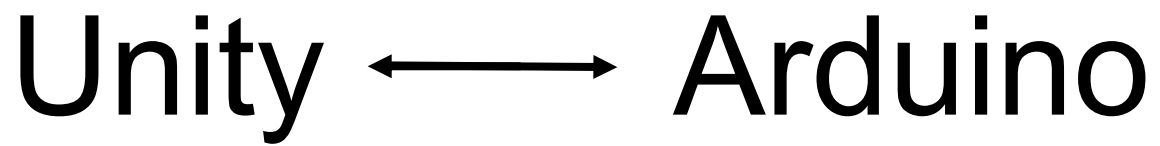
宣告要傳送給Arduino的string

把string parse成bytes 後送出

Unity Send



可以看到Arduino收到後的回傳順序是對的


Unity
Arduino

Arduino Send and Receive

ArduinoUnityCommunication

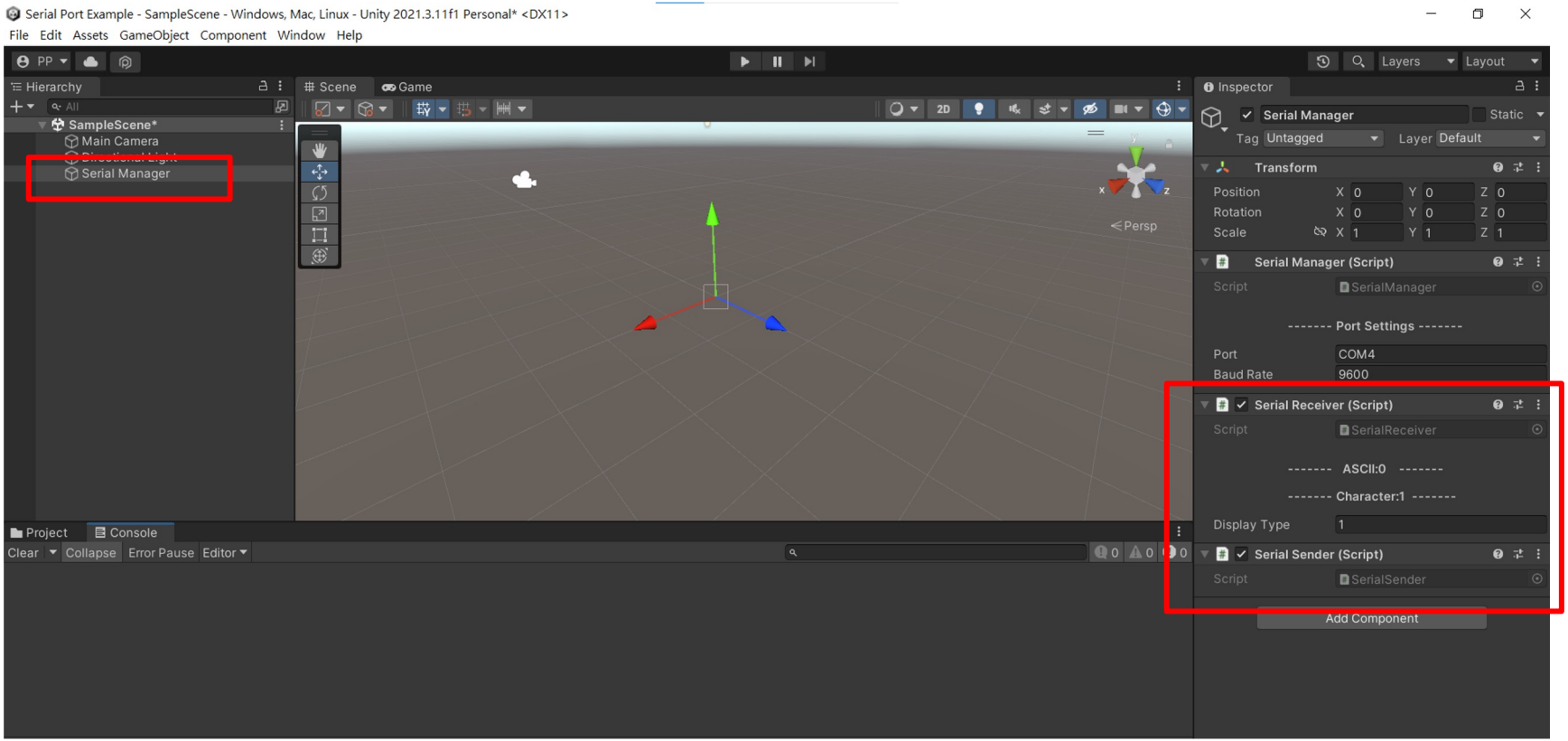
```
* Arduino的write() 是把傳輸內容當成位元組( 一次傳1byte的rang

* 因為write range只有1 byte = 8 bit range=0~255 若給超過會

*/


// 用來 Read data
const int receiveBufferSize = 200;
char receive[receiveBufferSize];


// 用來 Send data
char temp[] = "23C";

void setup() {

  Serial.begin( 9600 );

}

void loop() {

  sendData(temp);
  readData();

}

void sendData(char dataToSend[]){

  Serial.write(dataToSend, strlen (dataToSend));
  // delay(100);

}

void readData(){

  if (Serial.available() > 0) {

    Serial.readBytes(receive, receiveBufferSize);

    sendData(receive);

  }

}
```

readData() 跟 sendData() 都要

一樣把 delay 註解掉才能及時回傳

Unity Send and Receive

跟Unity send的時候一樣，下面兩個都要activate，因為要接收Arduino的傳送跟Arduino接收後的回傳



G456E是Unity給Arduino後回傳的，23C是Arduino直接傳給Unity的