PROGRAMMING ARDUINO WITH AUTONOMOUS CARS

RobotLAB

SMART. USEFUL. ROBOT.

Table of Contents

1. ALTINO's System

1.1 Overview of ALTINO

The ALTINO System is a robot platform designed for education and development of MSRDS (Microsoft Robotics Developer Studio) Visual Programming Language, C Language and Android Application Programming, and can be used as an education/ development platform for elementary, middle, high school and university students.

1.2 Features of ALTINO

- Nice design based on robot engineering design
- Graphics-based programming (Visual Programming Language) environment with the MSRDS
- Comprehensive education program with various application services including voice recognition provided in VPL
- At least 9 service activities to control the robot in VPL
- Steering structure based on rear wheel drive like vehicles
- Structure that can be assembled and disassembled to allow users to understand the mechanical principle of a robot
- Structure in which 6 ports are protruded on top, and composed of power and communication ports
- Robot that can run at maximum speed of 0.5m/s
- Structure with a compass sensor to know the direction of magnetic field
- Acceleration sensor equipped to measure changes in slopes and instantaneous speed
- Light sensor equipped to sense outside lightness
- Temperature sensor to know the internal temperature of the robot
- Remote sensor to control the robot
- Parallel use of lithium ion batteries with general primary cells, and driving for at least 6 hours

1.3 ALTINO Specifications

- Dimension: 98mm x 180mm x 63mm (LxWxH) / Standing height: 12mm
- Material: ABS,
- Processor: Atmega 128, Atmega 88 Dual Processor
- Communication between dual processors: Digital communication error detection algorithm by means of serial communication
- Motor: DC Geared motor / 20:1 / 250RPM / 3.5~8VDC
- Steering control motor: DC Geared motor / 220:1 / 2.5~6VDC
- Top speed: 0.5m/s (Max)
- Sensor:
  ① Infrared sensor for detecting obstacles: 6 (Front: 3 / Rear: one / Side: one for each side)
  ② Distance measurement: [Min] 2cm ~ [Max] 10cm
  ③ Steering control sensor: One
  ④ Magnetic field sensor [3 axes]: One
  ⑤ Battery voltage sensing: Output as ADC
  ⑥ Acceleration sensor [3 axes]: One
  ⑦ Temperature sensor: One
  ⑧ Light sensor: One
  ⑨ IR receiver: One
  ⑩ Gyro sensor [3 axes]: One

- Display
  - ① Dot-matrix 8x8,
  - ② State LED indicator: 13
  - ③ Buzzer: One
- I/O device
  - ① Power switch input: 1ea
  - ② PC communication connection port: RS232 1ea
  - ③ Wireless communication: Short-range wireless communication network Bluetooth
  - ④ Program downloading: ISP (In System Program)
- Battery:
  - ① Battery: Li-ion cell
  - ② Nominal Voltage: 7.4V 2600mAh
  - ③ Charging Voltage: 8.4V 1200mAh
  - ④ Runtime: At least 3 hours
- Charger:
  - ① Input: 100-240VAC / 47-63Hz
  - ② Output: 8.4V 1200mAh
  - ③ LED indicators - Red: Charging, - Green: Fully charge
  - ④ Charge time: 2 hours and 30 minutes
  - ⑤ Certification: Rohus, CE certified

1.4 Communication

1.4.1 Types of Communication Supported

	Type	Description
1	Bluetooth	Wireless environment. ex) wireless communication with smart phone, wireless communication with laptop/host PC, etc.
2	UART	Wired environment; for communication with other devices ex.) laser sensor, image recognition camera, sensors, etc.
3	ISP	used for downloading ATmega 128 program ex.) used for changing the data code created in the host PC to Hex and then downloading them to ALTINO

1.4.2 Protocol structure

- Transmission of 28-byte data from a computer (PC) to a mobile platform (ALTINO)

PC▸ ALTINO(28byte) COMMAND2 #1					
Array Num	Command code	Description of command	Array Num	Command code	Description of command
0	STX	Start (0x02)	14	LED ARRAY 1st	Dot-matrix right column 2
1	LENGTH	Transmission length (28)	15	LED ARRAY 2st	Dot-matrix right column 3

2	CHECKSUM	Data change check	16	LED ARRAY 3st	Dot-matrix right column 4
3	COMMAND1	Device No.	17	LED ARRAY 4st	Dot-matrix right column 5
4	COMMAND2	Transmission method code (1)	18	LED ARRAY 5st	Dot-matrix right column 6
5	FRONT STEERING VALUE	Front steering value	19	LED ARRAY 6st	Dot-matrix right column 7
6	RIGHT M OP MODE	Right motor mode setup	20	LED ARRAY 7st	Dot-matrix right column 8
7	RIGHT M PWM H	Right motor PWM higher value	21	IR SETUP	IR sensor operation setup
8	RIGHT M PWM L	Right motor PWM lower value	22	BUZZER	Buzzer output value
9	LEFT M OP MODE	Left motor mode setup	23	LED SETUP	Robot's front/rear indicator control value
10	LEFT M PWM H	Left motor PWM higher value	24	STEERING MODE	Steering Motor mode setup
11	LEFT M PWM L	Left motor PWM lower value	25	STEERING FINE VALUE	Steering Motor fine tuning value setup
12	LED COMMAND	Dot-matrix ASCII code value	26	SP	Temporary buffer
13	LED ARRAY 0st	Dot-matrix right column 1	27	ETX	End (0x03)

Array Num	Command code	Description of command
0	STX	Start (0x02): ALTINO checks whether it is the start of communication.
1	LENTH	Transmission length (28): ALTINO checks length.

2	CHECKSUM	Check data change: Adds 17 arrays except Array No.2 out of 28 arrays to check the remainder divided by 256.
3	COMMAND1	Device No.: Connects the robot with other devices to give them ID, respectively. **current command ID -> 1**
4	COMMAND2	Transmission method code: Gives ID to set up functions. **current command ID -> 1** **current command ID -> 2** **current command ID -> 3**
5	FRONT STEERING VALUE	Front steering value: Controls ALTINO steering direction. **Array Num 24 is 0:** **Left -> 1 Center -> 2 Right -> 3** **Array Num 24 is 1:** **Left -> 128 + (0~127) Right -> 0~127**
6	RIGHT M OP MODE	Set up right motor mode: Sets up whether to use the PID controller. **Do not use PID: 0** **Use PID: 1** **Motor Stop: 255**
7	RIGHT M PWM H	Higher value of right motor PWM: Receives an integer to have a quotient obtained by dividing it by 256. **Forward: 0~1000 (input value/256)** **Backward: 32768 + 0~1000 (input value/256)**
8	RIGHT M PWM L	Lower value of right motor PWM: Receives an integer to have a remainder obtained by dividing it by 256. **Forward: 0~1000 (input value%256)** **Backward: 32768 + 0~1000 (input value%256)**
9	LEFT M OP MODE	Set up left motor mode: Sets up whether to use PID controller. **Do not use PID: 0** **Use PID: 1** **Motor Stop: 255**
10	LEFT M PWM H	Higher value of left motor PWM: Receives an integer to have a quotient obtained by dividing it by 256. **Forward: 0~1000 (input value/256)** **Backward: 32768 + 0~1000 (input value/256)**
11	LEFT M PWM L	Lower value of left motor PWM: Receives an integer to have a remainder obtained by dividing it by 256. **Forward: 0~1000 (input value%256)** **Backward: 32768 + 0~1000 (input value%256)**

Array Num	Command code	Description of command

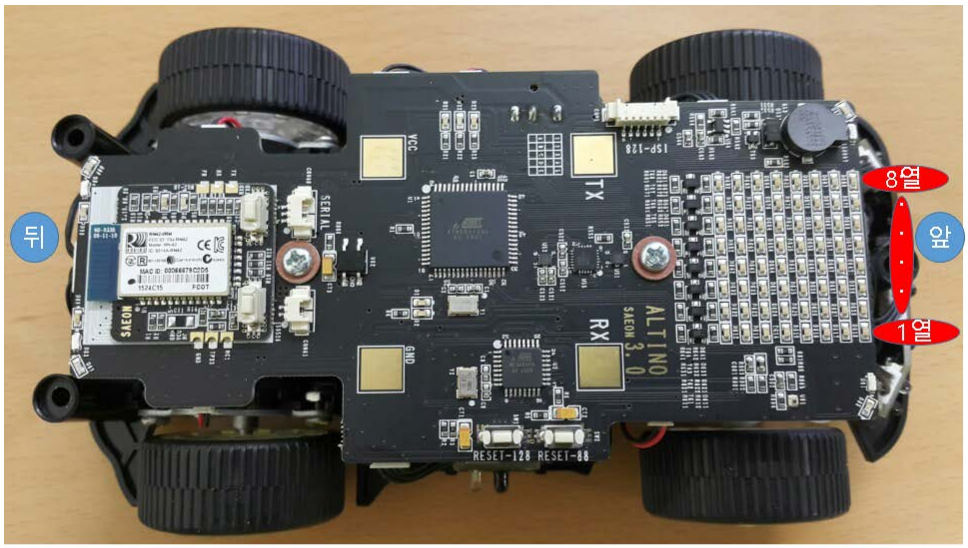
12	LED COMMAND	Dot-matrix ASCII Code value **Input value: ASCII Code value (see the following table)**

DEC	HEX	Char	DEC	HEX	Char	DEC	HEX	Char
0	00	NUL	43	2B	+	86	56	V
1	01	SOH	44	2C	,	87	57	W
2	02	STX	45	2D	-	88	58	X
3	03	ETX	46	2E	.	89	59	Y
4	04	EOT	47	2F	/	90	5A	Z
5	05	ENQ	48	30	0	91	5B	[
6	06	ACK	49	31	1	92	5C	\
7	07	BEL	50	32	2	93	5D	]
8	08	BS	51	33	3	94	5E	^
9	09	HT	52	34	4	95	5F	_
10	0A	LF	53	35	5	96	60	`
11	0B	VT	54	36	6	97	61	a
12	0C	FF	55	37	7	98	62	b
13	0D	CR	56	38	8	99	63	c
14	0E	SO	57	39	9	100	64	d
15	0F	SI	58	3A	:	101	65	e
16	10	DLE	59	3B	;	102	66	f
17	11	DCI	60	3C	<	103	67	g
18	12	DC2	61	3D	=	104	68	h
19	13	DC3	62	3E	>	105	69	I
20	14	DC4	63	3F	?	106	6A	j
21	15	NAK	64	40	@	107	6B	k
22	16	SYN	65	41	A	108	6C	l
23	17	ETB	66	42	B	109	6D	m
24	18	CAN	67	43	C	110	6E	n
25	19	EM	68	44	D	111	6F	o
26	1A	SUB	69	45	E	112	70	p
27	1B	ESC	70	46	F	113	71	q
28	1C	FS	71	47	G	114	72	r
29	1D	GS	72	48	H	115	73	s
30	1E	RS	73	49	I	116	74	t
31	1F	US	74	4A	J	117	75	u
32	20	Space	75	4B	K	118	76	v
33	21	!	76	4C	L	119	77	w
34	22	"	77	4D	M	120	78	x
35	23	#	78	4E	N	121	79	y
36	24	$	79	4F	O	122	7A	z

37	25	%		80	50	P		123	7B	{
38	26	&		81	51	Q		124	7C	|
39	27	'		82	52	R		125	7D	}
40	28	(		83	53	S		126	7E	~
41	29	)		84	54	T		127	7F	DEL
42	2A	*		85	55	U				

※ 128 + 0 to 127 are inverted in the dot matrix.

Array Num	Command code	Description of command
13	LED ARRAY 0st	Dot matrix (Right column 1) **Input: 0x00 ~ 0xff**
14	LED ARRAY 1st	Dot matrix (Right column 2) **Input: 0x00 ~ 0xff**
15	LED ARRAY 2st	Dot matrix (Right column 3) **Input: 0x00 ~ 0xff**
16	LED ARRAY 3st	Dot matrix (Right column 4) **Input: 0x00 ~ 0xff**
17	LED ARRAY 4st	Dot matrix (Right column 5) **input: 0x00 ~ 0xff**
18	LED ARRAY 5st	Dot matrix (Right column 6) **Input: 0x00 ~ 0xff**
19	LED ARRAY 6st	Dot matrix (Right column 7) **Input: 0x00 ~ 0xff**
20	LED ARRAY 7st	Dot matrix (Right column 8) **Input: 0x00 ~ 0xff**



8	4	2	1	8	4	2	1	
1	0	0	0	0	0	0	1	0x81
1	1	0	0	1	0	0	1	0xC9
1	1	1	0	1	1	0	1	0xED
0	0	0	0	1	1	1	1	0x0F

Array Num	Command code	Description of command
21	IR SETUP	Sets up whether to operate the IR sensor (To extend battery life) IR OUT: 0 (light emitting IR, not operating) IR OUT: 10 (light emitting IR, operating)
22	BUZZER	Buzzer output value Input: 0 ~ 255 (frequency)

- Setting up Buzzer Out. scale frequencies are printed out when data is entered from 0 to 255 at 4 Octave La(A) 440Hz criteria

1 Octave	Frequency (Hz)	Value	2 Octave	Frequency (Hz)	Value
Do(C)	32.7032	1	Do(C)	65.4064	13
C#	34.6478	2	C#	69.2957	14
Re(D)	36.7081	3	Re(D)	73.4162	15
D#	38.8909	4	D#	77.7817	16
Mi(E)	41.2034	5	Mi(E)	82.4069	17
Fa(F)	43.6535	6	Fa(F)	87.3071	18
F#	46.2493	7	F#	92.9989	19
So(G)	48.9994	8	So(G)	97.9989	20

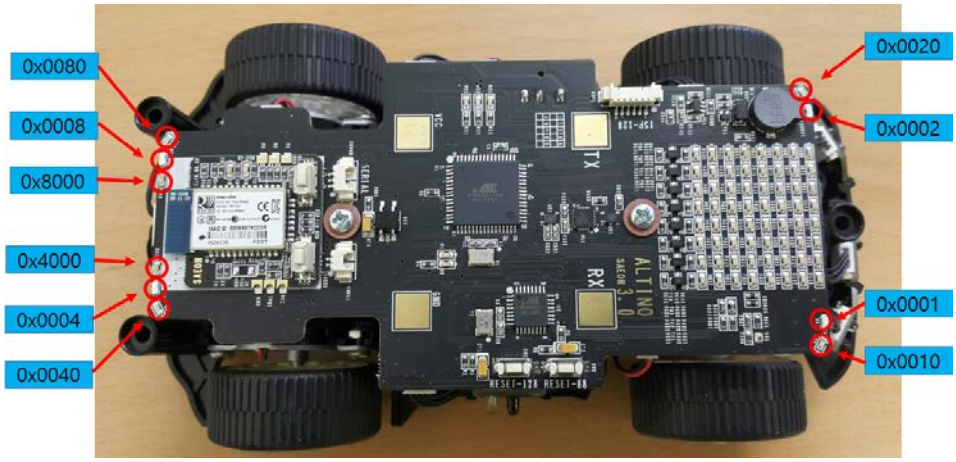
G#	51.9130	9	G#	103.8262	21
La(A)	55.0000	10	La(A)	110.0000	22
A#	58.2705	11	A#	116.5409	23
It(B)	61.7354	12	It(B)	123.4708	24

3 Octave	Frequency (Hz)	Value	4 Octave	Frequency (Hz)	Value
Do(C)	130.8128	25	Do(C)	261.6256	37
C#	138.5913	26	C#	277.1826	38
Re(D)	146.8324	27	Re(D)	293.6648	39
D#	155.5635	28	D#	311.1270	40
Mi(E)	164.8138	29	Mi(E)	329.6276	41
Fa(F)	174.6141	30	Fa(F)	349.2282	42
F#	184.9972	31	F#	369.9944	43
So(G)	195.9977	32	So(G)	391.9954	44
G#	207.6523	33	G#	415.3047	45
La(A)	220.0000	34	La(A)	440.0000	46
A#	233.0819	35	A#	466.1638	47
Ti(B)	246.9417	36	Ti(B)	493.8833	48

5 Octave	Frequency (Hz)	Value	6 Octave	Frequency (Hz)	Value
Do(C)	523.2511	49	Do(C)	1046.5020	61
C#	554.3653	50	C#	1108.7310	62
Re(D)	587.3295	51	Re(D)	1174.6590	63
D#	622.2540	52	D#	1244.5080	64
Mi(E)	659.2551	53	Mi(E)	1318.5100	65
Fa(F)	698.4565	54	Fa(F)	1396.9130	66
F#	739.9888	55	F#	1479.9780	67
So(G)	783.9909	56	So(G)	1567.9820	68
G#	830.6094	57	G#	1661.2190	69
La(A)	880.0000	58	La(A)	1760.0000	70
A#	932.3275	59	A#	1864.6550	71
Ti(B)	987.7666	60	Ti(B)	1975.5330	72

7 Octave	Frequency (Hz)	Value	8 Octave	Frequency (Hz)	Value
Do(C)	2093.0050	73	Do(C)	4186.0090	85
C#	2217.4610	74	C#	4434.9220	86
Re(D)	2349.3180	75	Re(D)	4698.6360	87
D#	2489.0160	76	D#	4978.0320	88
Mi(E)	2637.0200	77	Mi(E)	5274.0410	89
Fa(F)	2793.8260	78	Fa(F)	5587.6520	90
F#	2959.9550	79	F#	5919.9110	91
So(G)	3135.9630	80	So(G)	6271.9270	92
G#	3322.4380	81	G#	6644.8750	93
La(A)	3520.0000	82	La(A)	7040.0000	94
A#	3729.3100	83	A#	7458.6200	95
Ti(B)	3951.0660	84	Ti(B)	7902.1330	96

Array Num	Command code	Command code
23	LED OUT	Robot front and rear light control value Input Value: See the figure below.



※ Array Num 21 High 2bit and Array Num 23

Array Num	Command code	Description of command

24	STEERING MODE	Steering Motor mode setup **0 -> 3 step control (Left, right, center)** **1 -> absolute positioning control, proportional movement Do** **2 -> Relative position control, Angle increment of time**
25	STEERING FINE VALUE	Steering Motor fine-tuning value setup **If the MSB = 0, the fine tuning is on the left side. If the MSB = 1, the fine tuning is on the right side. The most significant bit except the remaining value for fine tuning settings.**
26	SP	Temporary buffer
27	ETX	End (0x03)

PC ▸ ALTINO(22byte) COMMAND #2

Array Num	Command code	Description of command	Array Num	Command code	Description of command
0	STX	Start(0x02)	11	LEFT M I gain	Left motor I gain
1	LENTH	Transmission length (22)	12	LEFT M D gain	Left motor D gain
2	CHECKSUM	Check data change	13	RIGHT M T Torque	Right motor T torque
3	COMMAND1	Device No.	14	RIGHT M P gain	Right motor P gain
4	COMMAND2	Transmission method code (2)	15	RIGHT M I gain	Right motor I gain
5	STEERING M T Torque	Steering motor T torque	16	RIGHT M D gain	Right motor D gain
6	STEERING P gain	Steering motor P gain	17	SP	Temporary buffer
7	STEERING I gain	Steering motor I gain	18	SP	Temporary buffer
8	STEERING D gain	Steering motor D gain	19	SP	Temporary buffer

9	LEFT M T Torque	Left motor T torque	20	SP	Temporary buffer
10	LEFT M P gain	Left motor P gain	21	ETX	End (0x03)

PC ▸ ALTINO(22byte) COMMAND #3					
Array Num	Description of command	Description of command	Array Num	Command code	Description of command
0	STX	Start(0x02)	11	G-sensor CTR 4	Acceleration sensor CTR 4
1	LENTH	Transmission length (22)	12	G-sensor CTR 5	Acceleration sensor CTR 5
2	CHECKSUM	Check data change	13	G-sensor CTR 6	Acceleration sensor CTR 6
3	COMMAND1	Device No.	14	G-sensor CTR 7	Acceleration sensor CTR 7
4	COMMAND2	Transmission method code (2)	15	G-sensor CTR 8	Acceleration sensor CTR 8
5	G-sensor X offset	Acceleration sensor X offset	16	G-sensor CTR 9	Acceleration sensor CTR 9
6	G-sensor Y offset	Acceleration sensor Y offset	17	SP	Temporary buffer
7	G-sensor Y offset	Acceleration sensor Z offset	18	SP	Temporary buffer
8	G-sensor CTR 1	Acceleration sensor CTR 1	19	SP	Temporary buffer
9	G-sensor CTR 2	Acceleration sensor CTR 2	20	SP	Temporary buffer
10	G-sensor CTR 3	Acceleration sensor CTR 3	21	ETX	End (0x03)

PC ▸ ALTINO(22byte) COMMAND #4					
Array Num	Command code	Description of command	Array Num	Command code	Description of command
0	STX	Start(0x02)	11	M-sensor CTR 4	Magnetic field sensor CTR4
1	LENTH	Transmission length (22)	12	M-sensor CTR 5	Magnetic field sensor CTR5
2	CHECKSUM	Check data change	13	M-sensor CTR 6	Magnetic field sensor CTR6
3	COMMAND1	Device No.	14	M-sensor CTR 7	Magnetic field sensor CTR7
4	COMMAND2	Transmission method code (4)	15	M-sensor CTR 8	Magnetic field sensor CTR8
5	M-sensor X H offset	Higher offset of magnetic field sensor X	16	M-sensor CTR 9	Magnetic field sensor CTR9
6	M-sensor X L offset	Lower offset of magnetic field sensor X	17	SP	Temporary buffer
7	M-sensor Y H offset	Higher offset of magnetic field sensor Y	18	SP	Temporary buffer
8	M-sensor Y L offset	Lower offset of magnetic field sensor Y	19	SP	Temporary buffer
9	M-sensor Z H offset	Higher offset of magnetic field sensor Z	20	SP	Temporary buffer
10	M-sensor Z L offset	Lower offset of magnetic field sensor Z	21	ETX	End (0x03)

Sending 31 bytes from mobile robot platform (ALTINO) to the computer (PC)

ALTINO ▸ PC(31byte) COMMAND #1					
Array Num	Command code	Description of command	Array Num	Command code	Description of command
0	STX	Start(0x02)	16	IR sensor 5 data L	Lower value of infrared sensor 5 data

1	LENTH	Transmission length (31)	17	IR sensor 6 data H	Higher value of infrared sensor 6 data
2	CHECKSUM	Check data change	18	IR sensor 6 data L	Lower value of infrared sensor 6 data
3	COMMAND1	Device No.	19	Right Mortar Torque H	Higher value of right motor torque data
4	COMMAND2	Transmission method code (1)	20	Right Mortar Torque L	Lower value of right motor torque data
5	System code H		21	Left Mortar Torque H	Higher value of left motor torque data
6	System code L		22	Left Mortar Torque L	Lower value of left motor torque data
7	IR sensor 1 data H	Higher value of infrared sensor 1 data	23	NTC temperature sensor H	Higher value of temperature sensor data
8	IR sensor 1 data L	Lower value of infrared sensor 1 data	24	NTC temperature sensor L	Lower value of temperature sensor data
9	IR sensor 2 data H	Higher value of infrared sensor 2 data	25	CDS H	Higher value of light sensor data
10	IR sensor 2 data L	Lower value of infrared sensor 2 data	26	CDS L	Lower value of light sensor data
11	IR sensor 3 data H	Higher value of infrared sensor 3 data	27	Remote control data	Remote control data value
12	IR sensor 3 data L	Lower value of infrared sensor 3 data	28	SP	Temporary buffer
13	IR sensor 4 data H	Higher value of infrared sensor 4 data	29	SP	Temporary buffer
14	IR sensor 4 data L	Lower value of infrared sensor 4 data	30	ETX	End (0x03)
15	IR sensor 5 data H	Higher value of infrared sensor 5 data			

ALTINO ▸ PC(31byte) COMMAND #2					
Array Num	Command code	Description of command	Array Num	Command code	Description of command
0	STX	Start(0x02)	16	M sensor Y Data L	Lower value of magnetic field sensor Y data
1	LENTH	Transmission length (31)	17	M sensor Z Data H	Higher value of magnetic field sensor Z data
2	CHECKSUM	Check data change	18	M sensor Z Data L	Lower value of magnetic field sensor Z data
3	COMMAND1	Device No.	19	Steering Var H	Higher value of steering variable resistance data
4	COMMAND2	Transmission method code (2)	20	Steering Var L	Lower value of steering variable resistance data
5	System code H		21	steering motor toque H	Higher value of steering motor torque data
6	System code L		22	steering motor toque L	Lower value of steering motor torque data
7	G sensor X Data H	Higher value of acceleration sensor X data	23	Battery Value H	Higher value of battery voltage data
8	G sensor X Data L	Lower value of acceleration sensor X data	24	Battery Value L	Lower value of battery voltage data
9	G sensor Y Data H	Higher value of acceleration sensor Y data	25	M sensor temperature	Magnetic field sensor temperature data value
10	G sensor Y Data L	Lower value of acceleration sensor Y data	26	SP	Temporary buffer
11	G sensor Z Data H	Higher value of acceleration sensor Z data	27	SP	Temporary buffer
12	G sensor Z Data L	Lower value of acceleration sensor Z data	28	SP	Temporary buffer

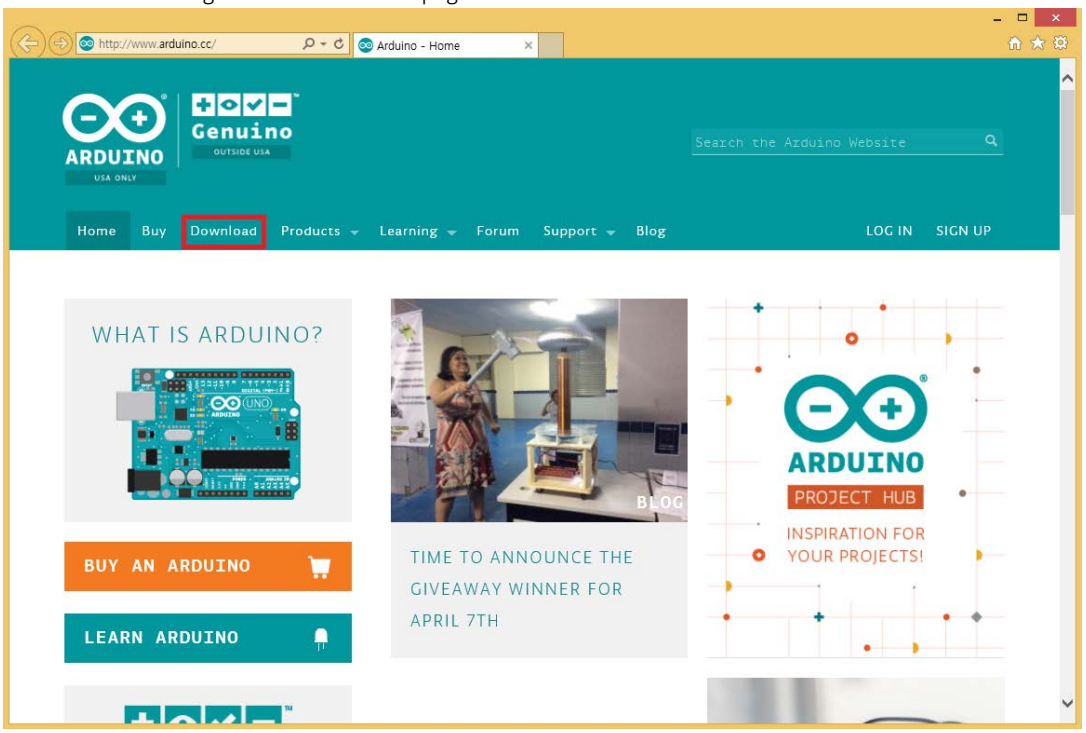
13	M sensor X Data H	Higher value of magnetic field sensor X data	29	SP	Temporary buffer
14	M sensor X Data L	Lower value of magnetic field sensor X data	30	ETX	End (0x03)
15	M sensor Y Data H	Higher value of magnetic field sensor Y data			

2. Installing a Sketch

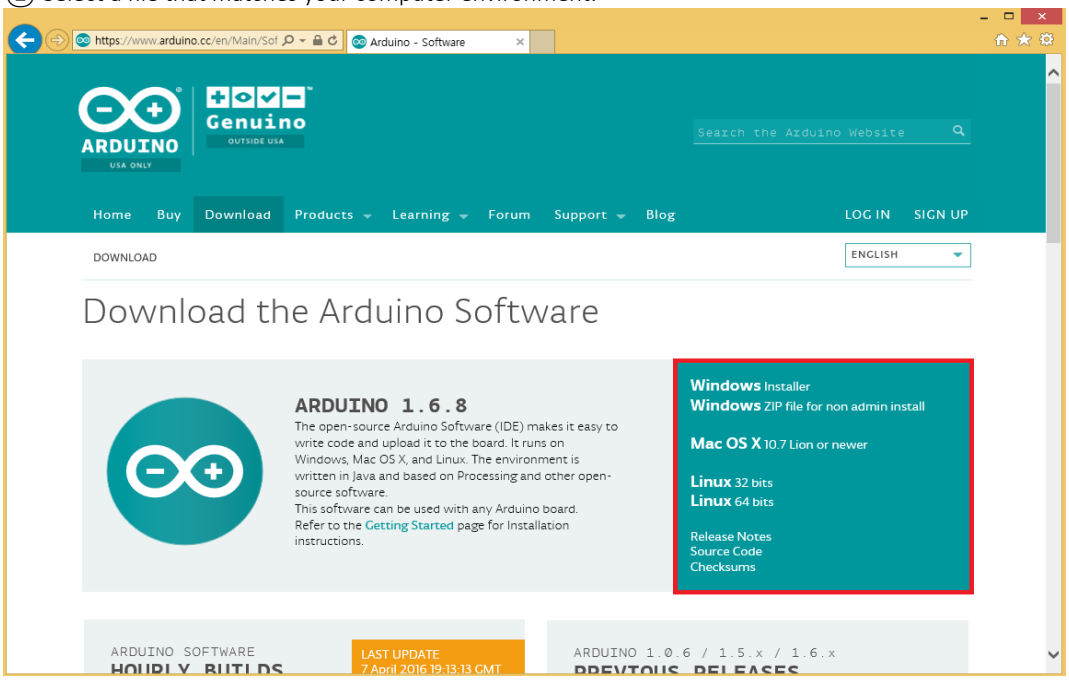
2.1 Arduino Installation

① http://www.arduino.cc/

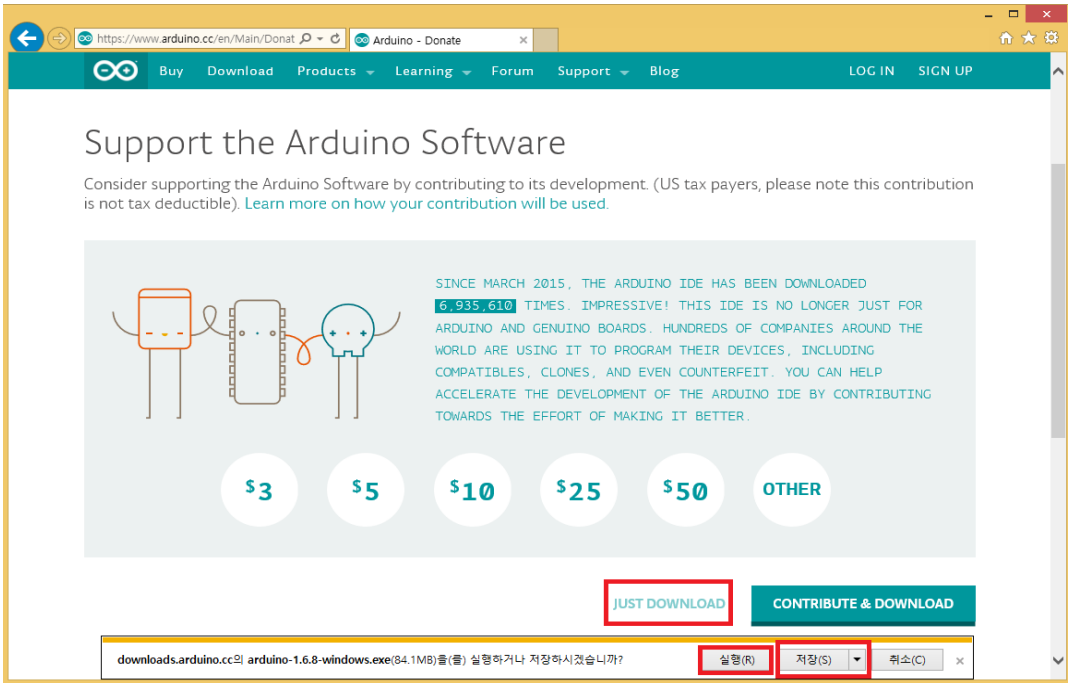
Access the site and go to the 'Download' page.



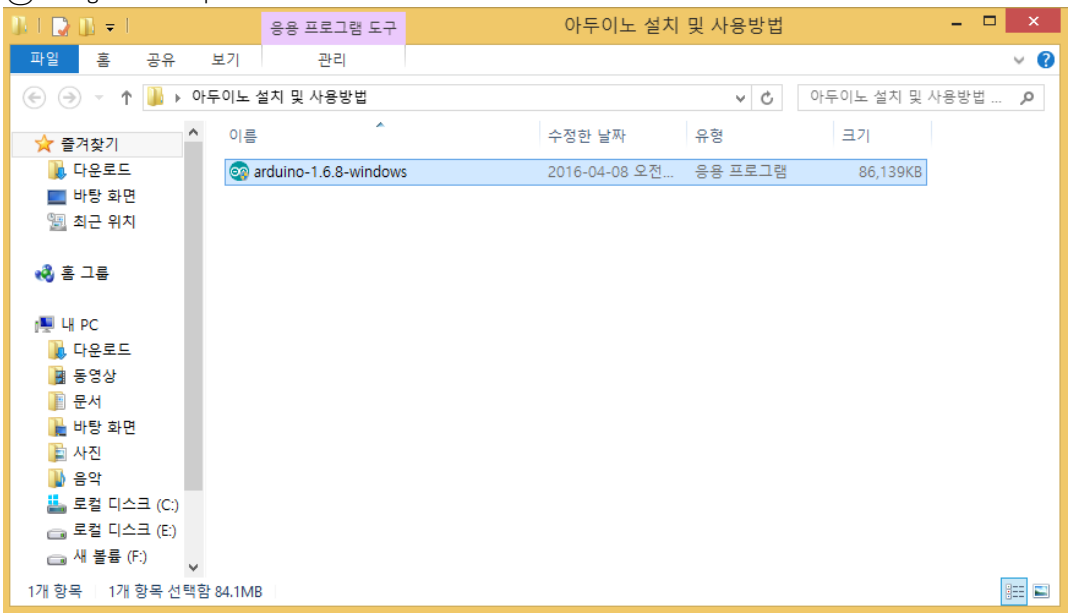
② Select a file that matches your computer environment.



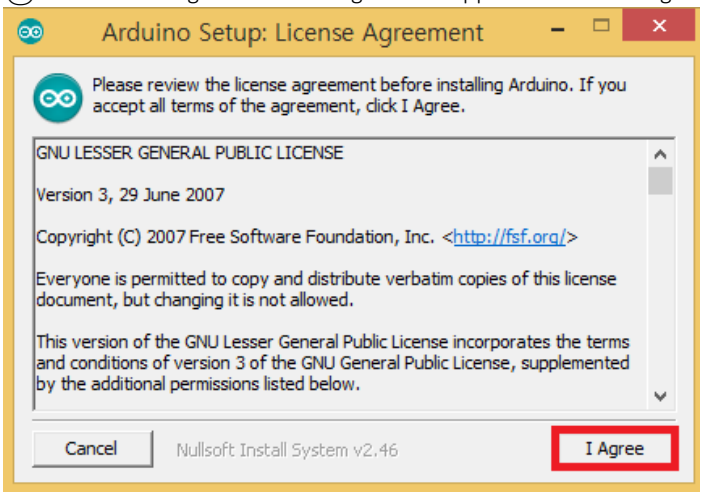
③ When you click "JUST DOWNLOAD", a download popup window will appear and you can select whether to execute or save the file.



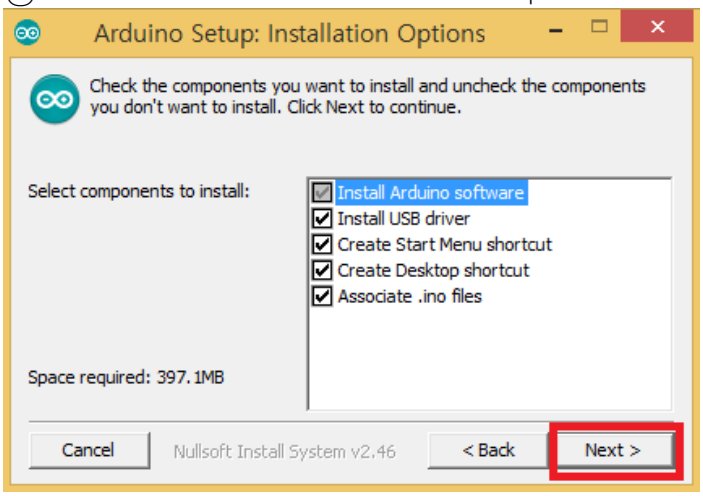
④ Navigate to the path of the downloaded Arduino installation file and run the Arduino installation file.



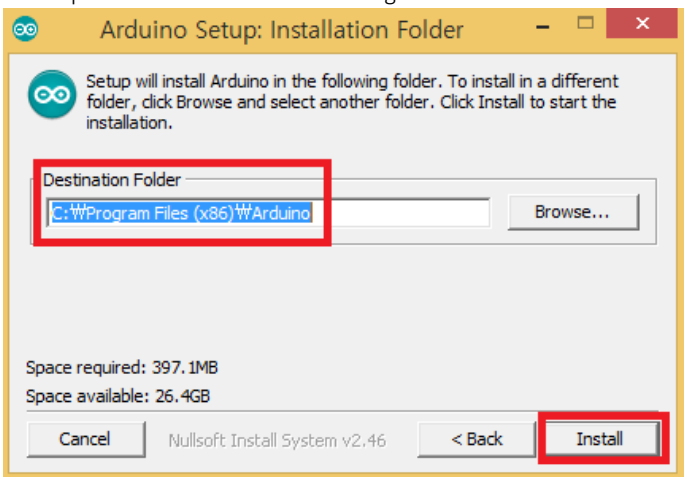
⑤ the "License Agreement" dialog box will appear. Click the "I Agree" button to proceed.



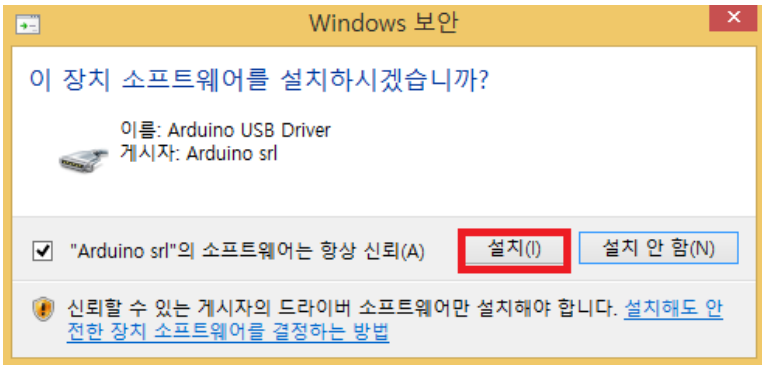
⑥ Check all boxes and click the "Next >" button to proceed.



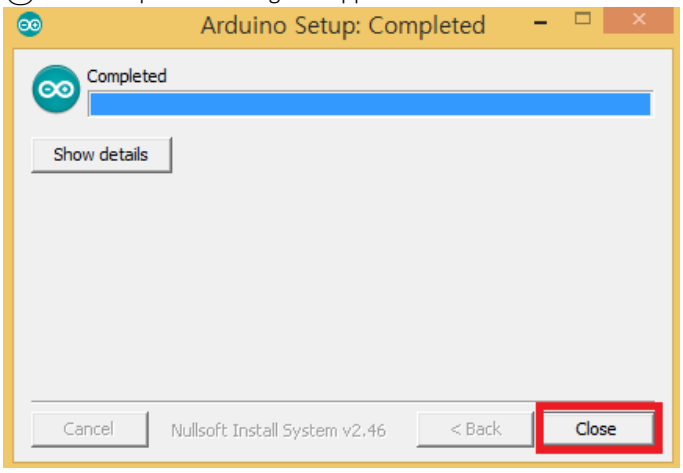
⑦ the "Installation Folder" dialog box for deciding on the configuration path appears. Click the "Install" button to proceed with the default settings.



⑧ Click "Install (I)" button to proceed.

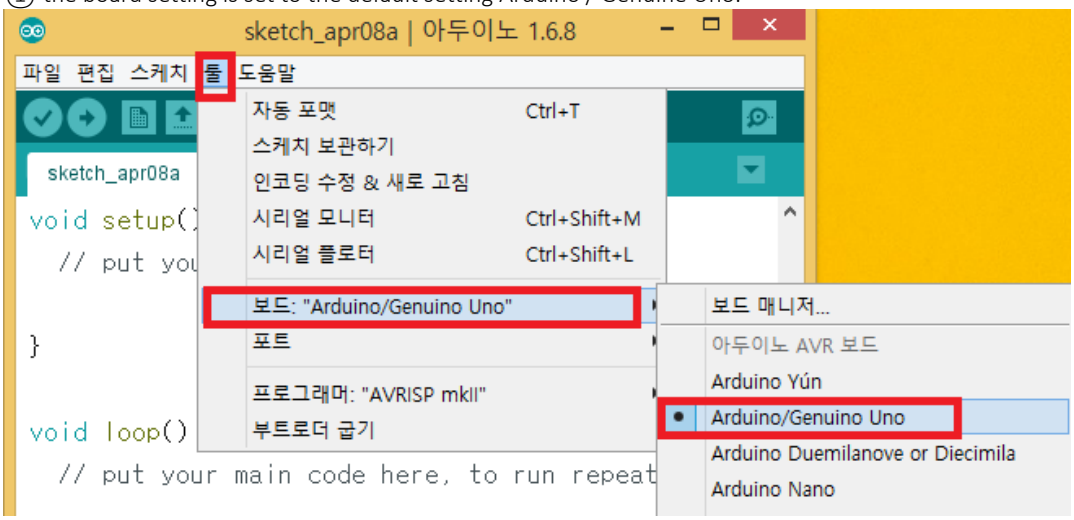


⑨ the "Completed" dialog box appears. Click the "Close" button to finish Arduino Setup and launch Arduino.

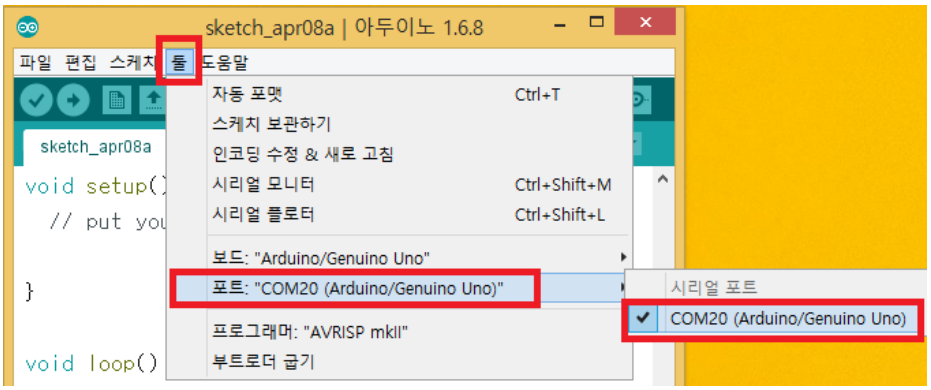


2.2 Arduino Preferences

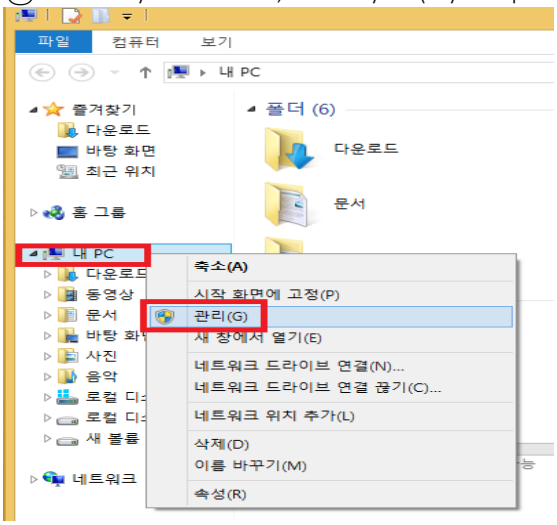
① the board setting is set to the default setting Arduino / Genuino Uno.



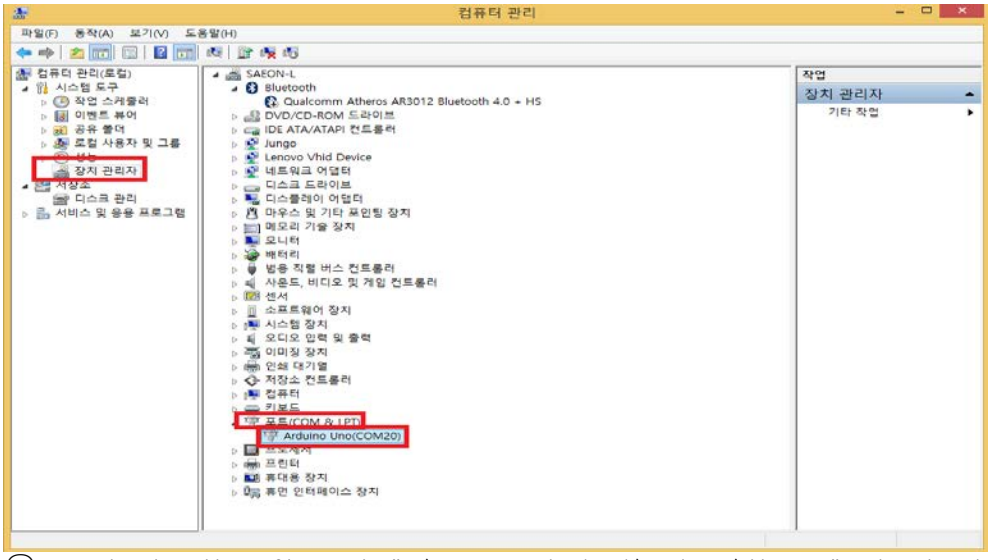
② the port settings are set to ComPort, which is loaded on your computer.



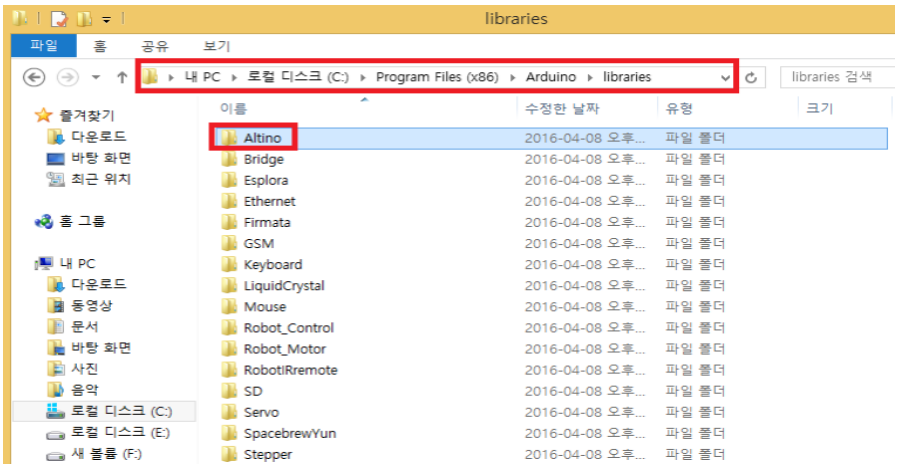
③ To check your ComPort, click "My PC (My Computer)- Right Mouse Button- Manage".



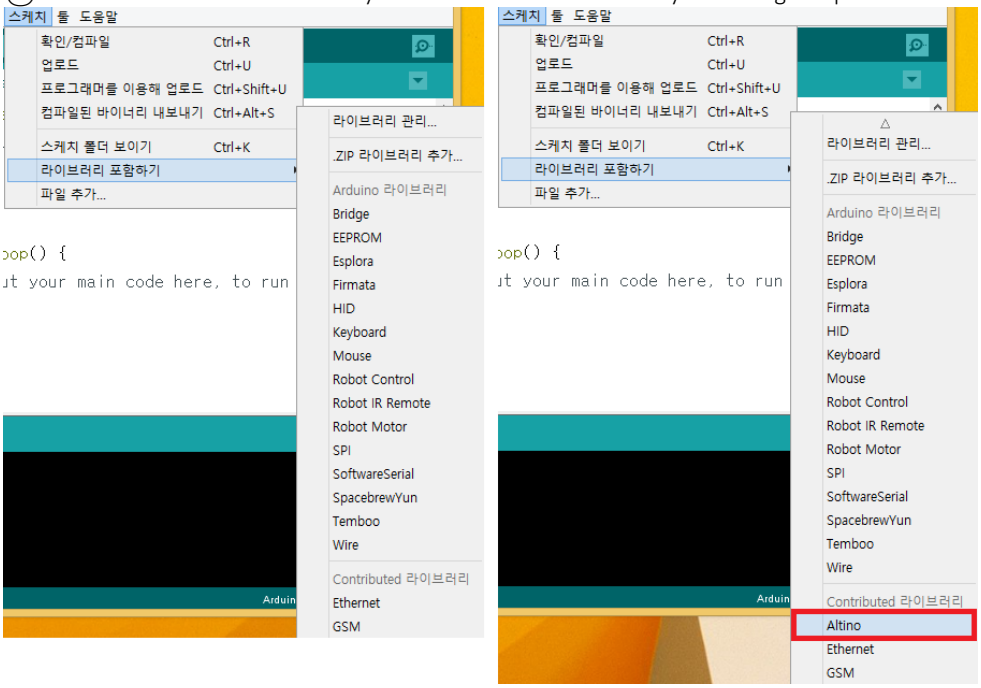
④ In the Computer Management window, click "Device Manager- Port (COM & LPT)" to see the currently created ComPort.



⑤ Paste the Altino library file into the "C: \ Program Files (x86) \ Arduino \ libraries" path with Arduino- libraries.



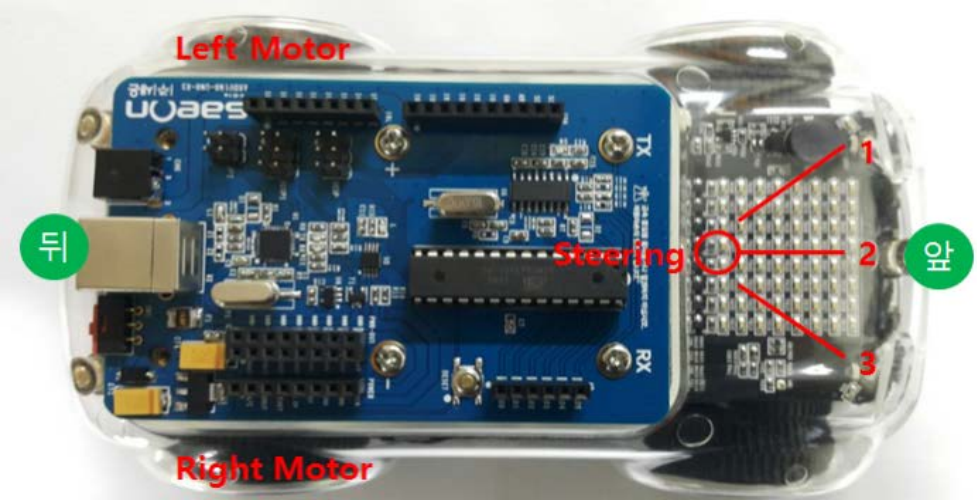
⑥ You can see that the Altino library has been created in the library list through step 5.



3. Control ALTINO Motor

3.1 Understanding the Altino DC motor

- Motor: DC Geared motor / 20:1 / 250RPM / 3.5~8VDC
- Steering control motor: DC Geared motor / 220:1 / 2.5~6VDC



3.2 ALTINO DC motor operation

- Go Function

Altino is a rear-wheel drive. Two motors are attached to the rear wheel of the Altino.

void Go (int left, int right)

The left argument value indicates the speed of the left motor and can be set to a value between-1000 and 1000. 0 means stop, 1000 means go at the fastest speed. If- is attached, the direction of rotation of the motor is reversed.

The right factor value indicates the speed of the right motor and can be set to a value between-1000 and 1000. 0 means stop, 1000 means go at the fastest speed. If- is attached, the direction of rotation of the motor is reversed.

- Steering Function

Altino's front wheels are the steering structure of the car.

void Steering (int steering value)

The value of the steering argument is 1 when rotated left, 2 when positioned in the center and 3 when rotated right.

1: Left direction
2: Center direction
3: Right direction

void Steering2(byte value1, byte value2)

value1 인자 값은 왼쪽 방향으로 회전할 때 128 + 0에서 127까지의 범위 값을 입력In the center, the direction is 0, and the right direction is 0 to 127.

Left direction: 128 + (0 ~ 127)
Center direction: 0
Right direction: 0 to 127

Value2 is a value to set the median value of Altino. If Altino is shifted to the right, enter the range value

from 128 + 0 to 127. If Altino is shifted to the left, enter 0 to 127, Can be adjusted.
When Altino shifts to the right: 128 + (0 ~ 127)
When Altino shifts to the left: 0 to 127

3.3 Altino DC motor operation training

3.3.1 Control Altino rear wheel motor

① Adding the source as shown below will cause Altino to advance at speed 300 and then stop after 1 second.

```
1   #include <Altino.h>
2
3   void setup ()
4   {
5     Serial. Begin (115200);
6     Go (300,300);
7     delay (1000);
8     Go (0,0);
9   }
10
11  void loop ()
12  {
13  }
```

1 : Function header file to control Altino
2 :
3 : Start the setup function
4 : Serial communication baud rate setting
5 : Altino advances at 300 speeds
6 : 1 second delay
7 : Altino stop
8 : End the setup function
9 :
10 :
11 : Start the loop statement
12 : end of loop statement

3.4 Control Altino Steering

① Adding the source as shown below will cause Altino to turn to the left and after 3 seconds change the steering direction to center.

```
1   #include <Altino.h>
2
3   void setup ()
4   {
5     Serial. Begin (115200);
6     Steering (1);
7     delay (3000);
8     Steering (2);
9   }
10
11  void loop ()
12  {
13  }
```

1: Function header file to control Altino
2:
3:
4: Start the setup function
5: Serial communication baud rate setting
6: Altino steering left direction
7: Delay for 3 seconds
8: Altino steering center direction
9: End the setup function
10:
11:
12: Start the loop statement
13: end of loop statement

3.4.1 Altino rear wheel motor and steering control

① Adding the source as shown below will cause Altino to turn to the right and back at 300, then after 3 seconds the steering will turn to the center and stop.

```
1   #include <Altino.h>
2
3   void setup ()
4   {
5     Serial. Begin (115200);
6     Steering (3);
7     Go (-300, -300);
8     delay (3000);
9     Steering (2);
10    Go (0,0);
11  }
12
13  void loop ()
14  {
15  }
```

```
1: Function header file to control Altino
2:
3:
4: Start the setup function
5: Serial communication baud rate setting
6: Altino steering right direction
7: Altino retracts at 300 speeds
8: Delay for 3 seconds
9: Altino steering center direction
10: Altino stop
11: End of setup function
12:
13:
14: Start the loop statement
15: end of loop statement
```

3.5 Practice of Altino DC motor operation application

3.5.1 Altino to operate as follows.

[Rear wheel motor 300 forward, steering left]
[3 second delay]
[rear wheel 300 rear right, steering right]
[2 second delay]
[stop, steering center]

3.5.2 Altino to operate as follows.

[1 second delay]
[rear wheel motor 400 forward, steering right]
[1 second delay]
[stop, center of steering]

3.5.3 Altino to operate as follows.

[Rear wheel motor 500 forward, steering right]
[1 second delay]
[rear wheel motor 500 forward, steering left]
[2 second delay]

4 . Control ALTINO buzzer and 10 LEDs

4.1 Understanding Altino Buzzer and LEDs

- It is possible to control the pitch of the buzzer by changing the frequency value.
When inputting data from 0 to 255, the lower scale frequency is output.
4 Octave (A) Based on 440 Hz.

1 Octave	Frequency (Hz)	BZO value	2 Octave	Frequency (Hz)	BZO value
Do(C)	32.7032	1	Do(C)	65.4064	13
C#	34.6478	2	C#	69.2957	14
Re(D)	36.7081	3	Re(D)	73.4162	15
D#	38.8909	4	D#	77.7817	16
Mi(E)	41.2034	5	Mi(E)	82.4069	17
Pa(F)	43.6535	6	Pa(F)	87.3071	18
F#	46.2493	7	F#	92.9989	19
Sol(G)	48.9994	8	Sol(G)	97.9989	20
G#	51.9130	9	G#	103.8262	21
La(A)	55.0000	10	La(A)	110.0000	22
A#	58.2705	11	A#	116.5409	23
Ti(B)	61.7354	12	Ti(B)	123.4708	24

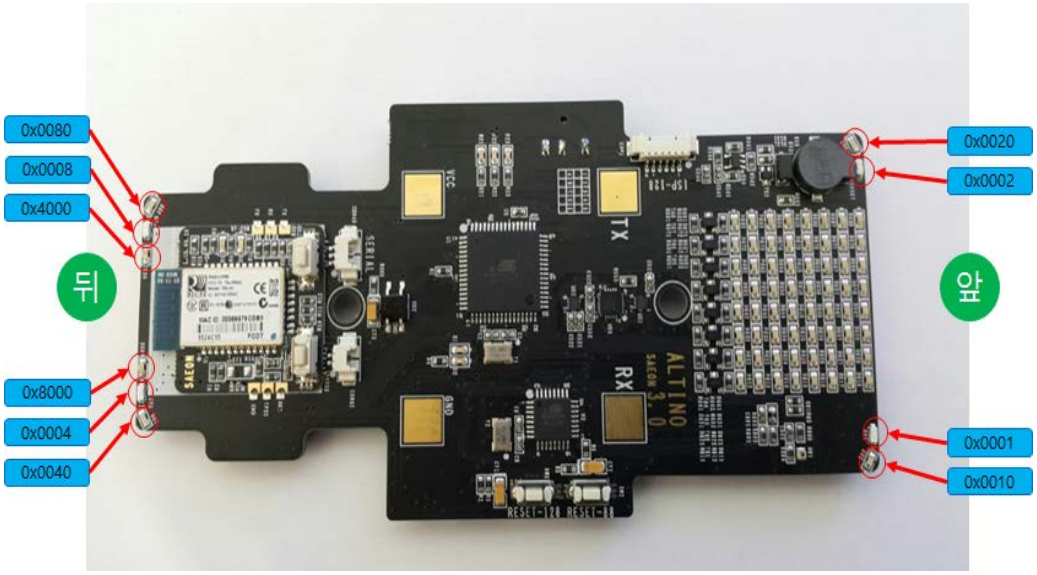
3 Octave	Frequency (Hz)	BZO value	4 Octave	Frequency (Hz)	BZO value
Do(C)	130.8128	25	Do(C)	261.6256	37
C#	138.5913	26	C#	277.1826	38
Re(D)	146.8324	27	Re(D)	293.6648	39
D#	155.5635	28	D#	311.1270	40
Mi(E)	164.8138	29	Mi(E)	329.6276	41
Pa(F)	174.6141	30	Pa(F)	349.2282	42
F#	184.9972	31	F#	369.9944	43
Sol(G)	195.9977	32	Sol(G)	391.9954	44
G#	207.6523	33	G#	415.3047	45
La(A)	220.0000	34	La(A)	440.0000	46

A#	233.0819	35	A#	466.1638	47
Ti(B)	246.9417	36	Ti(B)	493.8833	48

5 Octave	Frequency (Hz)	BZO value	6 Octave	Frequency (Hz)	BZO value
Do(C)	523.2511	49	Do(C)	1046.5020	61
C#	554.3653	50	C#	1108.7310	62
Re(D)	587.3295	51	Re(D)	1174.6590	63
D#	622.2540	52	D#	1244.5080	64
Mi(E)	659.2551	53	Mi(E)	1318.5100	65
Pa(F)	698.4565	54	Pa(F)	1396.9130	66
F#	739.9888	55	F#	1479.9780	67
Sol(G)	783.9909	56	Sol(G)	1567.9820	68
G#	830.6094	57	G#	1661.2190	69
La(A)	880.0000	58	La(A)	1760.0000	70
A#	932.3275	59	A#	1864.6550	71
Ti(B)	987.7666	60	Ti(B)	1975.5330	72

7 Octave	Frequency (Hz)	BZO value	8 Octave	Frequency (Hz)	BZO value
Do(C)	2093.0050	73	Do(C)	4186.0090	85
C#	2217.4610	74	C#	4434.9220	86
Re(D)	2349.3180	75	Re(D)	4698.6360	87
D#	2489.0160	76	D#	4978.0320	88
Mi(E)	2637.0200	77	Mi(E)	5274.0410	89
Pa(F)	2793.8260	78	Pa(F)	5587.6520	90
F#	2959.9550	79	F#	5919.9110	91
Sol(G)	3135.9630	80	Sol(G)	6271.9270	92
G#	3322.4380	81	G#	6644.8750	93
La(A)	3520.0000	82	La(A)	7040.0000	94
A#	3729.3100	83	A#	7458.6200	95
Ti(B)	3951.0660	84	Ti(B)	7902.1330	96

- Eight LEDs can be controlled as shown below.



4.2 Altino to 10 LED and buzzer operation

- Sound function

It outputs the buzzer sound on the Altino.

void Sound (unsigned char buzzer)

Buzzer The buzzer sound value is 0 ~ 255.

- Led function

It outputs the buzzer sound on the Altino.

void Led (unsigned char led)

The LED argument value controls 10 LEDs from 0x0000 to 0xc0ff.

4.3 Altino buzzer and 10 LED operation exercises

4.3.1 Control Altino buzzer

① When you add the source as shown below, Altino will play the buzzer sound for 3 seconds with sound of level 37 (Do).

```
1  #include <Altino.h>
2
3  void setup ()
4  {
5    Serial. Begin (115200);
6    Sound (37);
7    delay (3000);
8    Sound (0);
9  }
10
11 void loop ()
12 {
13 }
```

```
1: function header file to control Altino
2:
3:
4: Start the setup function
5: Serial communication baud rate setting
6: Altino buzzer sound 37 levels output
7: Delay for 3 seconds
8: Altino buzzer stops sounding
9: End the setup function
10:
11:
12: Start the loop statement
13: end of loop statement
```

4.3.2 Control Altino 10 LEDs

① Adding the source as shown below will cause Altino to turn on all 10 LEDs and then turn them off after 5 seconds.

```
1   #include <Altino.h>
2
3   void setup ()
4   {
5     Serial. Begin (115200);
6     Led(0xC0FF);
7     delay (5000);
8     Led(0x0000);
9   }
10
11  void loop ()
12  {
13  }
```

```
1: function header file to control Altino
2:
3:
4: Start the setup function
5: Serial communication baud rate setting
6: Turn on all 10 Altino LEDs
7: 5 second delay
8: Turn off all 10 Altino LEDs
9: End the setup function
10:
11:
12: Start the loop statement
13: end of loop statement
```

4.3.3 Control Altino buzzer and 10 LEDs

① When adding the source as below, Altino will turn on only 2 front lights among 10 LEDs, operate with buzzer sound level 39 (LE), and after 10 seconds all 10 LEDs will be turned off and the buzzer sound will stop.

```
1   #include <Altino.h>
2
3   void setup ()
4   {
5     Serial. Begin (115200);
6     Led(0x0003);
7     delay (1000);
8     Sound (39);
9     delay (5000);
10    Led (0);
11    delay (1000);
12    Sound (0);
13    delay (1000);
14  }
15
16  void loop ()
17  {
18  }
```

1: function header file to control Altino
2:
3:
4: Start the setup function
5: Serial communication baud rate setting
6: Turn on two lights such as Altino front lights
7:
8: Altino buzzer sound 39 (Le) level output
9: 5 second delay
10: Turn off all 10 Altino LEDs
11: 1 second delay
12: Stopping Altino buzzer sound
13: 1 second delay
14: end of setup function
15:
16:
17: Start the loop statement
18: end of loop statement

4.3.4 Playing "Do Re Mi Fa So La Ti Do" with the Altino Buzzer

① Add the source as shown below and Altino plays "Do Re Mi Pa Sol La Ti Do", then the buzzer stops.

```
1   #include <Altino.h>
```

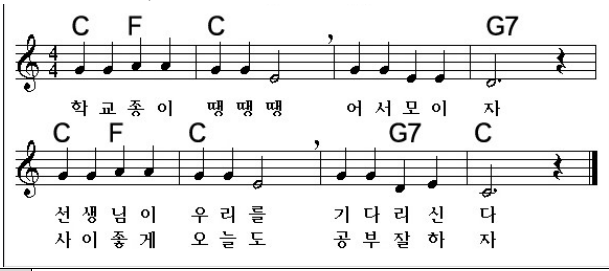
```
2
3   void setup ()
4   {
5     Serial. Begin (115200);
6     Sound (37);
7     delay (1000);
8     Sound (39);
9     delay (1000);
10    Sound (41);
11    delay (1000);
12    Sound (42);
13    delay (1000);
14    Sound (44);
15    delay (1000);
16    Sound (46);
17    delay (1000);
18    Sound (48);
19    delay (1000);
20    Sound (49);
21    delay (1000);
22    Sound (0);
23  }
24
25  void loop ()
26  {
27  }
28
```

13 : 1: function header file to control Altino
14 : 2:
15 : 3:
16 : 4: Start the setup function
17 : 5: Serial communication baud rate setting
18 : 6: Altino buzzer sound 37 levels (Do) Output
19 : 7: 1 second delay
20 : 8: Altino Buzzer Sound Level 39 (Le) Output
21 : 9: 1 second delay
22 : 10: Sound of Altino buzzer 41 level (Mi) Output
23 : 11: 1 second delay
24 : 12: Altino buzzer sound level 42 (Pa) Output
25 : 13: 1 second delay
26 : 14: Altino Buzzer Sound Level 44 (Sol) Output
27 : 15: 1 second delay
28 : 16: Altino buzzer sound Level 46 (La) Output
29 : 17: 1 second delay
30 : 18: Altino buzzer sound Level 48 (Ti) Output
31 : 19: 1 second delay
32 : 20: Altino buzzer sound Level 49 (Do) output
33 : 21: 1 second delay
34 : 22: Stop Altino buzzer sound
35 : 23: end of setup function
36 : 24:
37 : 25:

38 : 26: Starting the loop statement 39 : 27: end of loop statement

4.3.5 Playing "school paper ding-ding" with the Altino Buzzer

① When you add the source as shown below, Altino uses a buzzer to play "school paper ding ding" then the buzzer stops.




```
#include <Altino.h>

void setup ()
{
 Serial. Begin (115200);
 //Sol Sol La La Sol Sol Mi Sol Sol Mi Mi Re
 Sound (44);
 delay (500);
 Sound (0);
 delay (100);
 Sound (44);
 delay (500);
 Sound (0);
 delay (100);
 Sound (46);
 delay (500);
 Sound (0);
 delay (100);
 Sound (46);
 delay (500);
 Sound (0);
 delay (100);

 Sound (44);
 delay (500);
 Sound (0);
 delay (100);
 Sound (44);
 delay (500);
 Sound (0);
 delay (100);

 Sound (41);
 delay (1000);
 Sound (0);
 delay (100);

```

```
38  Sound (44);
39  delay (500);
40  Sound (0);
41  delay (100);
42  Sound (44);
43  delay (500);
44  Sound (0);
45  delay (100);
46
47  Sound (41);
48  delay (500);
49  Sound (0);
50  delay (100);
51  Sound (41);
52  delay (500);
53  Sound (0);
54  delay (100);
55  Sound (39);
56  delay (1000);
57  Sound (0);
58  delay (100);
59
60  //Sol Sol La La Sol Sol Mi Sol Mi Re Mi Do
61  Sound (44);
62  delay (500);
63  Sound (0);
64  delay (100);
65  Sound (44);
66  delay (500);
67  Sound (0);
68  delay (100);
69  Sound (46);
70  delay (500);
71  Sound (0);
72  delay (100);
73  Sound (46);
74  delay (500);
75  Sound (0);
76  delay (100);
77
78  Sound (44);
79  delay (500);
80  Sound (0);
81  delay (100);
82  Sound (44);
83  delay (500);
84  Sound (0);
85  delay (100);
86
87  Sound (41);
88  delay (1000);
89  Sound (0);
```

```
90    delay (100);
91
92    Sound (44);
93    delay (500);
94    Sound (0);
95    delay (100);
96    Sound (41);
97    delay (500);
98    Sound (0);
99    delay (100);
100
101   Sound (39);
102   delay (500);
103   Sound (0);
104   delay (100);
105   Sound (41);
106   delay (500);
107   Sound (0);
108   delay (100);
109   Sound (37);
110   delay (1000);
111   Sound (0);
112   delay (100);
113 }
114
115 void loop ()
116 {
117 }
```

4.4 Practice Altino buzzer and LED operation

4.4.1 Altino to operate as follows.

[10 LEDs left direction light 2 on, steering left]
[3 seconds delay]
[10 LEDs right direction light 2 on, steering right]
[2 second delay]
[10 LEDs off, steering middle]

4.4.2 Altino to operate as follows.

[Advance to 300 speeds, turn on 2 lights of 10 LED lights]

[2 seconds delay]
[10 LEDs turn on two brake lights, back to 300 speeds, buzzer sound 41 level Delay]
[10 LEDs off, buzzer stop, rear wheel motor stop]

4.4.3 Altino to operate as follows.

[3 Delay]

[3 speed delay]
[Leftward, steering direction left, 10 LEDs on the left side. (10 degrees)]
[10 LEDs off, buzzer sound stops, rear wheel motor stop, steering direction center]

5 . Control ALTINO Dot Matrix

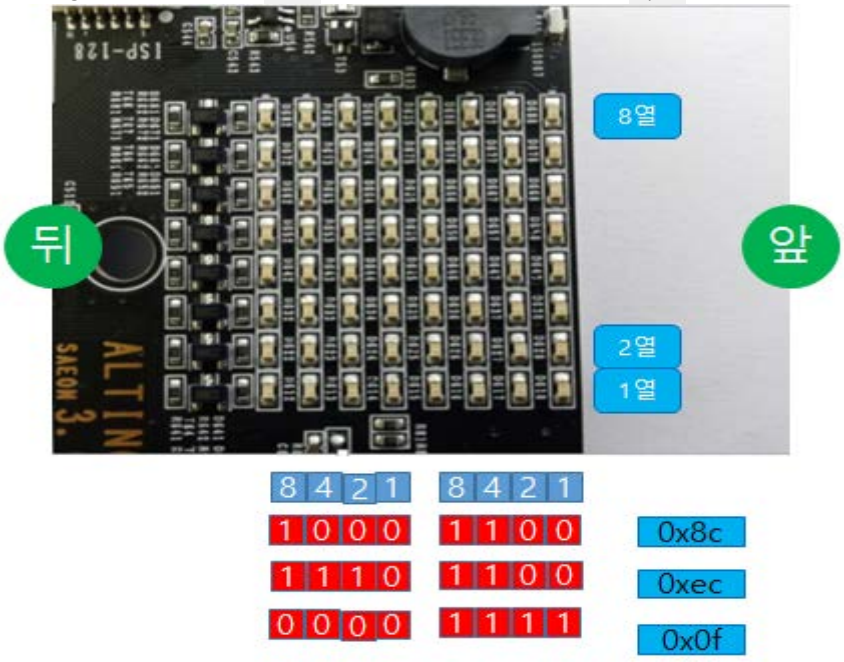
5.1 Understanding the Altino Dot Matrix

- The table below shows the ASCII code values.

DEC	HEX	Char	DEC	HEX	Char	DEC	HEX	Char
0	00	NUL	43	2B	+	86	56	V
1	01	SOH	44	2C	,	87	57	W
2	02	STX	45	2D	-	88	58	X
3	03	ETX	46	2E	.	89	59	Y
4	04	EOT	47	2F	/	90	5A	Z
5	05	ENQ	48	30	0	91	5B	[
6	06	ACK	49	31	1	92	5C	\
7	07	BEL	50	32	2	93	5D	]
8	08	BS	51	33	3	94	5E	^
9	09	HT	52	34	4	95	5F	_
10	0A	LF	53	35	5	96	60	`
11	0B	VT	54	36	6	97	61	a
12	0C	FF	55	37	7	98	62	b
13	0D	CR	56	38	8	99	63	c
14	0E	SO	57	39	9	100	64	d
15	0F	SI	58	3A	:	101	65	e
16	10	DLE	59	3B	;	102	66	f
17	11	DCI	60	3C	<	103	67	g
18	12	DC2	61	3D	=	104	68	h
19	13	DC3	62	3E	>	105	69	I
20	14	DC4	63	3F	?	106	6A	j
21	15	NAK	64	40	@	107	6B	k
22	16	SYN	65	41	A	108	6C	l
23	17	ETB	66	42	B	109	6D	m
24	18	CAN	67	43	C	110	6E	n
25	19	EM	68	44	D	111	6F	o
26	1A	SUB	69	45	E	112	70	p
27	1B	ESC	70	46	F	113	71	q
28	1C	FS	71	47	G	114	72	r
29	1D	GS	72	48	H	115	73	s
30	1E	RS	73	49	I	116	74	t
31	1F	US	74	4A	J	117	75	u
32	20	Space	75	4B	K	118	76	v
33	21	!	76	4C	L	119	77	w
34	22	"	77	4D	M	120	78	x

35	23	#		78	4E	N		121	79	y
36	24	$		79	4F	O		122	7A	z
37	25	%		80	50	P		123	7B	{
38	26	&		81	51	Q		124	7C	|
39	27	'		82	52	R		125	7D	}
40	28	(		83	53	S		126	7E	~
41	29	)		84	54	T		127	7F	DEL
42	2A	*		85	55	U				

- The figure below shows the dot matrix can be controlled line by line.



5.2 Running the Altino Dot Matrix

- Display function

The dot matrix of Altino can be controlled. When an ASCII code value is input, the character corresponding to the ASCII code value is output to the dot matrix.

void Display (unsigned char ASCII)

- Display Line function

The ASCII factor value outputs the ASCII code value to the dot matrix.

void Display Line (unsigned char dot0, unsigned char dot1, unsigned char dot2, unsigned char dot3, unsigned char dot4, unsigned char dot5, unsigned char dot6, unsigned char dot7)

dot0 factor value is Right of 1 column Dot-matrix. Input value is 0x00 ~ 0xff
dot1 factor value is Right of 2 columns Dot-matrix. Input value is 0x00 ~ 0xff
dot3 factor value is Right of 4 columns Dot-matrix. Input value is 0x00 ~ 0xff
dot4 factor value is Right of 5 columns Dot-matrix. Input value is 0x00 ~ 0xff

dot5 factor value is Right of 6 columns Dot-matrix. Input value is 0x00 ~ 0xff
dot6 factor value is Right of 7 columns Dot-matrix. Input value is 0x00 ~ 0xff
dot7 factor value is Right of 8 columns Dot-matrix. Input value is 0x00 ~ 0xff

5.3 Practice of Altino dot matrix operation

5.3.1 Control the Altino dot matrix ASCII code

① Adding the source as shown below will cause Altino to display the 'A' character in the dot matrix and disappear after a 2 second delay.

```
1   #include <Altino.h>
2
3   void setup ()
4   {
5     Serial. Begin (115200);
6     Display('A');
7     delay (2000);
8     Display (0);
9   }
10
11  void loop ()
12  {
13  }
```

1 : function header file to control Altino
2 :
3 :
4 : Start the setup function
5 : Serial communication baud rate setting
6 : Output 'A' character to Altino dot matrix
7 : 2 second delay
8 : Turn off Altino dot matrix
9 : End the setup function
10 :
11 :
12 : Start the loop statement
13 : end of loop statement

5.3.2 Control the Altino dot matrix line

① Adding the source as shown below will cause Altino to turn on all rows of the dot matrix and then turn them off after a 3 second delay.

```
1  #include <Altino.h>
2
3  void setup ()
4  {
5    Serial. Begin (115200);
6    Display Line(0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff);
7    delay (3000);
8    Display Line (0,0,0,0,0,0,0,0);
9  }
10
11 void loop ()
12 {
13 }
```

14 : Function header file to control Altino
15 :
16 :
17 : Start the setup function
18 : Serial communication baud rate setting
19 : Turn on all the Altino dot matrices
20 : 3 second delay
21 : Turn off all heat in the Altino dot matrix
22 : End setup function
23 :
24 :
25 : Start loop statement
26 : End of loop statement

5.3.3 Altino dot matrix ASCII code and line control

① Adding the source as shown below will cause Altino to output '5' to the dot matrix, delay for 3 seconds, turn on only the first column of the dot matrix, delay for three seconds, and turn off the dot matrix.

```
1  #include <Altino.h>
2
3  void setup ()
4  {
5    Serial. Begin (115200);
6    Display ('5');
7    delay (3000);
8    Display Line(0xff,0,0,0,0,0,0,0);
9    delay (3000);
10   Display (0);
11 }
12
13 void loop ()
14 {
15 }
```

```
1: function header file to control Altino
2:
3:
4: Start the setup function
5: Serial communication baud rate setting
6: Output '5' character to the Altino dot matrix
7: Delay for 3 seconds
8: Turn on the first row of the Altino Dot Matrix
9: Delay for 3 seconds
10: Turn off the Altino dot matrix
11: End the setup function
12:
13:
14: Start the loop statement
15: end of loop statement
```

5.4 Practice of application of Altino dot matrix

5.4.1 Altino to operate as follows.

['A' character output]
[2-second delay - Output 'b' character]
[2-second delay]
['c' character output]
[2-second delay]
[Dot Matrix Off]

5.4.2 Altino to operate as follows.

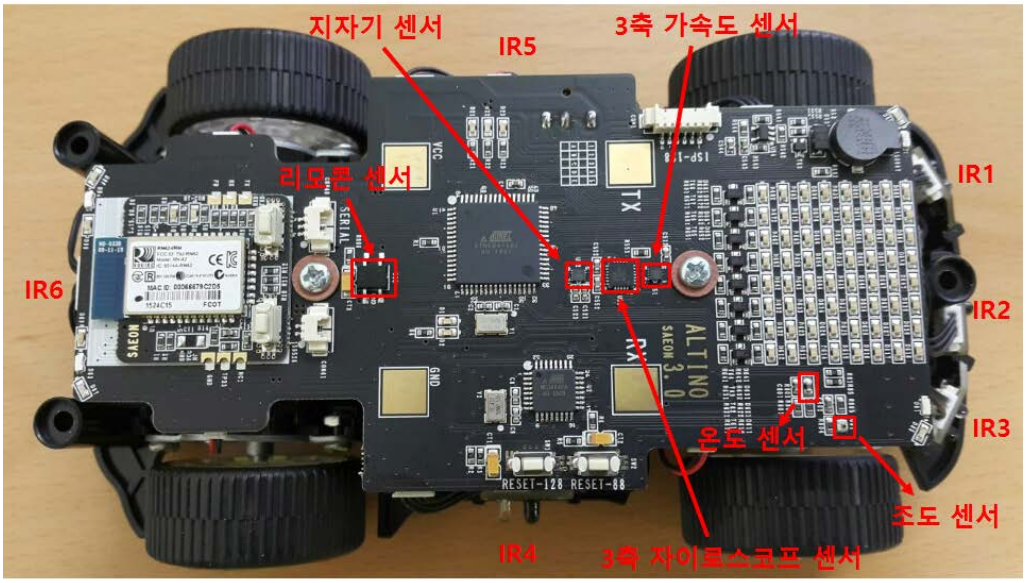
[':' Character output]
[2-second delay]
['?' Output character]
[2-second delay]
['^' character output]
[2-second delay]
[Dot Matrix Off]

5.4.3 Make a heart shape on the Altino dot matrix.

6 . Control ALTINO Sensors

6.1 Understanding Altino sensors

- Altino has 6 IR sensors, the illuminance sensor, 3-axis acceleration sensor, magnetic sensor, temperature sensor and remote sensor.



6.2 Operating Altino Sensors

- Altino Sensors

1. IR sensor
2. Illuminance sensor
3. Temperature sensor
4. Torque sensor
5. 3-axis acceleration sensor
6. Magnetic sensor
7. Rear wheel motor torque
8. Steering angle
9. Steering torque
10. Battery remaining indicator.

- Sensor Data Sensor ()

The Sensor Data remover returns the sensor value to the ADC. There are 6 IR sensors, 2 torque sensors, 1 temperature sensor, 1 illuminance sensor, 3 axes acceleration sensor, 3 magnetic field sensors, 1 steering angle, 1 steering torque and 1 remaining battery indication.

6.3 Practice of Altino sensor operation

6.3.1 Control Altino illumination sensor

① Add the source as below to declare the structure variable and receive the sensor value. Altino will get the sensor value and output it to the dot matrix.

```
1  #include <Altino.h>
2
```

```
3   Sensor Data data;
4
5   void setup ()
6   {
7     Serial. Begin (115200);
8   }
9
10  void loop ()
11  {
12    data=Sensor (1);
13    Display (48+sdata.CDSSensor/100);
14  }
```

1: function header file to control Altino
2:
3: Declaring a structure variable to receive the sensor value
4:
5:
6: Start the setup function
7: End the setup function
8:
9:
10: Start the loop statement
11: Accepts the value of the Altino sensor. (When inputting the parameter value 1, 6 IR sensor values, rear motor torque value, temperature sensor, and illuminance sensor value is obtained. Sensor XYZ axis, steering value, steering torque value).
12: Output of illuminance sensor value
13: Start the loop statement

6.3.2 Control Altino battery power

① Add the source as below to declare the structure variable and receive the sensor value. Altino will receive the sensor value and outputs the battery residual value to the dot matrix.

```
1   #include <Altino.h>
2
3   Sensor Data data;
4
5   void setup ()
6   {
7     Serial. Begin (115200);
8   }
9
10  void loop ()
11  {
12    data=Sensor (2);
13    Display(48+(antibattery/100));
14    delay (1000);
15    Display (48+ (sdata. Battery%100)/10);
16    delay (1000);
```

```
17    Display (48+ (sdata. Battery%100) %10);
18    delay (1000);
19  }
```

1: function header file to control Altino
2:
3: Declaring a structure variable to receive the sensor value
4:
5:
6: Start the setup function
7: Serial communication transmission speed setting
8: End of setup function
9:
10:
11: Start the loop statement
12: Accepts the value of the Altino sensor (If you input 1, it obtains six IR sensors, a rear motor torque value, a temperature sensor, and an illuminance sensor value. Sensor XYZ axis, steering value, steering torque value).
13: Remaining battery capacity 100-digit output
14: 1 second delay
15: Battery power remaining 10 outputs
16: 1 second delay
17: Battery power remaining 1 output
18: 1 second delay
19: end of loop statement

6.4 Practice of application of Altino sensor

6.4.1 Output the value of Altino 3-axis acceleration sensor with Sound function.

6.4.2 Output the Altino temperature sensor value to the display function as many digits.

6.4.3 Output the Altino geomagnetic sensor value as Sound function.

7 . Understanding data and variables using ALTINO

Each program defines data types that can be used in the language. Specifying data types is important because they determine the purpose of use and the size of the memory space needed.

7.1 Understanding Data

7.1.1 Constant

- Name for a storage device of which the value first declared does not change
- Numerical constant: Numbers including integers or real numbers
- Character constant: Constant represented for a number by using symbols

7.1.2 Data Types and Usage

Type	Byte Count	Expression Range
char	1	-128 - 127
signed char	1	-128 - 127
unsigned char	1	0 - 255
int	4	-2147483648 - 2147483647
signed int	4	-2147483648 - 2147483647
unsigned int	4	0 - 4294967295
short int	2	-32768 - 32767
signed short int	2	-32768 - 32767
unsigned short int	2	0 - 65535
long int	4	-2147483648 - 2147483647
signed long int	4	-2147483648 - 2147483647
unsigned long int	4	0 - 4294967295
float	4	3.4E-38 - 3.4E+38
double	8	1.7E-308 - 1.7E+308
long double	80(bit)	3.4E-4932 - 3.4E4932

7.2 Understanding variables

7.2.1 Variable

- Value to be used during program execution
- The name given to a temporary storage location
- Variable names must all be unique
- Uppercase and lowercase letters, numbers, and underscores are the only characters allowed.
- The first character of a variable name must be an alphabetic or an underscore
- The maximum variable name length is 31 characters
- Certain reserved words can not be used as variable names

7.2.2 Reserved words

- char, int, float, double, long, short, unsigned, struct, union, enum, void, typedef, switch, case, default, break, continue, return, goto, for, do, while, if, else, auto, static, register, extern

7.2.3 Specifying types

Output type	Description
%d	Output as a decimal integer
%o	Output as an octal integer
%x	Output as a hexadecimal integer
%u	Output as an unsigned decimal integer
%c	Output one character
%s	Output character string
%f	Output as floating point like 12.345
%e	Output as floating point like 1,2345E12
%g	Output as a type which needs smaller digits between %e and %f

7.3 Exercise C++ programming with ALTINO

7.3.1 Using character variable to output characters in ALTINO dotmatrix

① Add the following source

```
1   #include <Altino.h>
2
3   void setup () {
4
5    Serial.begin(115200);
6
7    char a;
8    a='K';
9    Display(a);
10   delay(2000);
11   a='9';
12   Display(a);
13   delay(2000);
14   a=75;
15   Display(a);
16   delay(2000);
17  }
18
    void loop() {

    }
```

```
1: function to control Altino
2 :
3: Start the setup function
4 :
5: Serial communication baud rate setting
6:
7: Character variable declaration
8: Save 'K' in character variable a
9: Text output using the character variable a in the Altino dot matrix
10: Delay for 2 seconds
11: Store '9' in character variable a
12: Text output using the character variable a in the Altino dot matrix
13: Delay for 2 seconds
14: Store integer 75 in character variable a
15: Text output using the character variable a in the Altino dot matrix
16: 2 second delay
17:
18: Start the loop statement
19: end of loop statement
```

7.3.2 Controlling ALTINO speed by using integer variable

① Add the following source

```
1   #include <Altino.h>
2
3   int left;
4   int right;
5   void setup() {
6
7     Serial.begin(115200);
8
9     left = 200;
10    right = 200;
11    Go(left, right);
12
13    delay(2000);
14
15    left = -200;
16    right = -200;
17    Go(left, right);
18
19    delay(2000);
20
21    left = 0;
22    right = 0;
23    Go(left, right);
24    delay(2000);
25  }
26
27  void loop() {
```

```
28
}
```

```
27 : 1: function to control Altino
28 : 2 :
29 : 3: Declaration of integer variable left
30 : 4: Declare integer variable right
31 : 5: Start the setup function
32 : 6:
33 : 7: Serial communication transmission speed setting
34 : 8 :
35 : 9: Store 200 in integer variable left
36 : 10: Store 200 in integer variable right
37 : 11: Input the left variable value to the left motor of Altino rear wheel motor and the right variable value to the right motor.
38 : 12:
39 : 13: Delay for 2 seconds
40 : 14:
41 : 15: -200 stored in the left variable of the integer variable
42 : 16: Store -200 in the right variable of the integer variable
43 : 17: Input the left variable value to the left motor of Altino rear wheel motor and the right variable value to the right motor.
44 : 18:
45 : 19: 2 second delay
46 : 20:
47 : 21: Store -200 in the left variable of the integer variable
48 : 22: Store -200 in integer variable right
49 : 23: Input the left variable value to the left motor of Altino rear wheel motor and the right variable value to the right motor.
50 : 24: 2 second delay
51 : 25: end of setup function
52 : 26:
53 : 27: Start the loop statement
54 : 28: end of loop statement
```

7.3.3 Control the Altino dot matrix in hexadecimal

① Add the source as shown below.

```
1  #include <Altino.h>
2
3  void setup() {
4
5   Serial.begin(115200);
6
7   byte  dot0, dot1, dot2, dot3, dot4, dot5, dot6, dot7;
8
9   dot0 = 0xff;
10  dot1 = 0xff;
11  dot2 = 0xff;
12  dot3 = 0xff;
13  dot4 = 0xff;
14  dot5 = 0xff;
15  dot6 = 0xff;
16  dot7 = 0xff;
17
18  Display Line(dot0, dot1, dot2, dot3, dot4, dot5, dot6, dot7);
19
20  delay(2000);
21
22  dot0 = 0x0f;
23  dot1 = 0x0f;
24  dot2 = 0x0f;
25  dot3 = 0x0f;
26  dot4 = 0xf0;
27  dot5 = 0xf0;
28  dot6 = 0xf0;
29  dot7 = 0xf0;
30
31  Display Line(dot0, dot1, dot2, dot3, dot4, dot5, dot6, dot7);
32
33  delay(2000);
34 }
35
36 void loop() {
37
    }
```

1: function to control Altino
2 :
3: Start setup function
4 :
5: Serial communication baud rate setting
6:
7: Declare eight byte variables
8 :
9: Store hex 0xff in byte variable dot0
10: Store hexadecimal 0xff in byte variable dot1
11: Store hexadecimal 0xff in byte variable dot2
12: Hexadecimal 0xff stored in byte variable dot3

```
13: Store hexadecimal 0xff in byte variable dot4
14: Store hexadecimal 0xff in byte variable dot5
15: Store hexadecimal 0xff in byte variable dot6
16: Store hexadecimal 0xff in the byte variable dot7
17:
18: Altino dot matrix control is performed by inputting eight hexadecimal byte variables into a
function controlled by an Altino dot matrix line.
19:
20: 2 second delay
21:
22: Store hexadecimal 0xff in byte variable dot0
23: Store hexadecimal 0xff in byte variable dot1
24: Store hexadecimal 0xff in byte variable dot2
25: Store hexadecimal 0xff in byte variable dot3
26: Store hexadecimal 0xff in byte variable dot4
27: Store hexadecimal 0xff in byte variable dot5
28: Store hexadecimal 0xff in byte variable dot6
29: Store hexadecimal 0xff in byte variable dot7
30:
31: Altino dot matrix control by inputting 8 hexadecimal byte variables into a function controlled by
an Altino dot matrix line.
32:
33: 2 second delay
34: end of setup function
35:
36: Start loop statement
37: end of loop statement
```

7.4 Practice data and variable application using Altino

7.4.1 Output 0~9 in the ALTINO by using character variables.

7.4.2 Output your first initial by using character variables.

7.4.3 Output “Altino” by entering an integer constant in the character variable.

7.4.4 Move the ALTINO forward at its maximum speed for 3 seconds and then stop it using integer variables.

7.4.5 Move the ALTINO backward at its maximum speed for 3 seconds and then stop it using integer variables.

7.4.6 Make the ALTINO carry out the following task using integer variables.

[Forward for 1 second]
[Backward for 1 second]
[Forward for 2 seconds]
[Backward for 2 seconds – Stop]

7.4.7 Make a heart shape using hexadecimal numbers to output on the ALTINO.

7.4.8 Output each line from the left to the right of the dot matrix in the ALTINO

7.4.9 Output a heart shape from the left to the right in the ALTINO using hexadecimal numbers.

8 Learning Operator by Using ALTINO

C programming uses a variety of operators. Exemplary operators include the four rules of arithmetic and operators for calculating true or false. They are the four rules of arithmetic, that is, +, -, *, /, and % (arithmetic operator for finding remainders).

Another exemplary operator are the relational operator and logical operator. They are described below in more detail.

8.1 Understanding operators

8.1.1 Arithmetic operator

Operator	Function	Expression	Priorities
-	Subtraction	a-b	2
+	Plus	abs	2
*	Multiply	a*b	1
/	Dividing	a/b	1
	실수나누기(실수)		
%	Remainder (only integer)	abs	1

8.1.2 Relational operator

Operator	Function	Expression	Priorities
>	Big	a>b	1
>=	Bigger or Equal	a>=b	1
<	Small	a<b	1
<=	Smaller or Equal	a<=b	1
==	Same	a==b	2
!=	Different	a!=b	2

8.1.3 Logical operator

Operator	Function	Expression	Priorities
!	NOT	!(Relation)	1
&&	AND	(Relation)&&(Relation)	2

||	OR	(Relation)||(Relation)	3

Type	Use	Description
&&	a &&b	True if both a and b are true, false if either of the two is false
||	a || b	True if either a or b is true or both are true, false if both are false
!	!a	False if a is true, true if a is false

8.1.4 Bitwise operator

Operator	Function	Expression	Priorities
~	NOT	!(bit)	1
&	AND	(bit)&(bit)	2
^	XOR	(bit)^(bit)	3
|	OR	(bit)|(bit)	4

Type	Use	Description
&	a &b	1 if both bits are 1, otherwise 0
|	a | b	1 if at least one bit is 1, 0 if both are 0
^	a ^ b	1 if bits of variable a and variable b are different, 0 if they are the same
~	~a	0 if the bit of variable a is 1, 1 if it is 0

8.1.5 Move operator

Operator	Function	Expression	Priorities
<<	Move the bit to the left	x <<(number) move x value to the left as much as number	1

>>	Move the bit to the right	move x value to the right as much as number	2

8.1.6 Assignment operator

Operator	Function	Expression
=	Substitution	x=10 (substitute 10 for x) y=x (substitute 10 for y) z=x+y (substitute 20 for z)

8.1.7 Compound operator

Operator	Function	Expression of Compound assignment operators	
+=	add to present	x+=c	x=exec
-=	subtract to present	x-=c	x=exec
*=	multiply to present	x*=c	x=x*c
/=	dividing to present	x/=c	x=x/c
%=	remainder to present	x%=c	x=x%c
&=	OR to present	x&=c	x=x&c
|=	AND to present	x|=c	x=x|c
>>=	left shift	x>>=c	x=x>>c
<<=	right shift	x<<=c	x=x<<c

8.1.8 Increment and decrement operator

Operator	Function	Expression	Equation	Priority
++	increment by 1	a++, ++a	a+=1(a=a+1)	1
--	decrement by 1	a--, --a	a-=1(a=a-1)	1

8.1.9 Conditional operator

Operator	Function	Expression
?:	select by condition	[variable]=(condition)?(sentense) : (sentense)

8.1.10 Casting operator

Operator	Function	Expression
Data Type	Data Type conversion	(Data Type) variable

8.2 Exercise C++ programming with ALTINO

8.2.1 Output 234 in the dotmatrix.

① Display 234 in regular sequence at Dot-matrix by using the dividing (/) and remainder (%) operators after inputting Integer variable 234.

```
1   #include <Altino.h>
2
3   void setup () {
4
5     Serial.begin(115200);
6
7     int data=234;
8
9     Display(48+data/100);
10    delay (200);
11
12    Display(48+data%100/10);
13    delay (200);
14
15    Display(48+data%10);
16    delay (200);
17  }
18
19  void loop () {
20  }
```

```
1: function to control Altino
2:
3: Start setup function
4:
5: Serial communication baud rate setting
6:
7: Declare the integer variable data and enter 234 as the initial value.
8:
9: Output the value of hundreds of digits in the Altino dot matrix. 48 is an ASCII
   code value of 0, so 48 + numbers are used to output 0 to 9 digits. So, to compute
   the hundreds digit of 248 stored in the data variable to print the value of
   hundreds, you can calculate it as 48 + data / 100.
10: 0.2 second delay
11:
12: Outputs a value of ten digits to the Altino dot matrix. 48 is an ASCII code value
   of 0, so 48 + numbers are used to output 0 to 9 digits. Therefore, to compute
   the 24 digits of the tens stored in the data variable to print the value
   of the decimal place, you can calculate it as 48 + data% 100/10.
13: 0.2 second delay
14:
15: Outputs the value of one digit in the Altino dot matrix. 48 is an ASCII code
   value is 0, so 48 + numbers are used to output 0 to 9 digits. Therefore, to
   compute the 248-digit number stored in the data variable to print the value of
   the day of the week, you can calculate it as 48 + data% 10.
16: 0.2 second delay
17: end of setup function
18:
19: Starting the loop statement
20: end of loop statement
```

8.2.2 Controlling dot matrix by using shift operator

① After inputting 0x01 in byte variable, 1byte shift to the left and display on the dot-matrix.

```
1   #include <Altino.h>
2
3   void setup () {
4
5    Serial.begin(115200);
6
7    byte data=0x01;
8
9    Display Line(data,data,data,data,data,data,data,data);
10   delay(1000);
11
12   data=data<<1;
13   display Line(data,data,data,data,data,data,data,data);
14   delay(1000);
```

```
15
16    data=data<<1;
17    DisplayLine(data,data,data,data,data,data,data,data);
18    delay(1000);
19
20    data=data<<1;
21    DisplayLine(data,data,data,data,data,data,data,data);
22    delay(1000);
23
24    data=data<<1;
25    DisplayLine(data,data,data,data,data,data,data,data);
26    delay(1000);
27
28    data=data<<1;
29    DisplayLine(data,data,data,data,data,data,data,data);
30    delay(1000);
31
32    data=data<<1;
33    DisplayLine(data,data,data,data,data,data,data,data);
34    delay(1000);
35
36    data=data<<1;
37    DisplayLine(data,data,data,data,data,data,data,data);
38    delay(1000);
39  }
40
41  void loop() {
42  }
```

1: function to control Altino
2 :
3: Start setup function
4 :
5: Serial communication baud rate setting
6:
7: Declare the byte variable data and input hexadecimal number 0x01 as the initial value.
8 :
9: Turn on the first line of the top line in the Altino dot matrix.
10: 1 second delay
11:
12: Shifts the data variable one bit to the left and stores it. 0x02 is stored in data.
13: Turn on the second line from the top line to the Altino dot matrix.
14: 1 second delay
15:
16: Shifts the data variable one bit to the left and stores it. 0x04 is stored in the data.
17: Turn on the third line from the top row to the Altino dot matrix.
18: 1 second delay
19:
20: Shifts the data variable one bit to the left and stores it. 0x08 is stored in the data.
21: Turn on the fourth line from the top line to the Altino dot matrix.
22: 1 second delay
23:
24: Shifts the data variable one bit to the left and stores it. 0x10 is stored in data.

```
25: Turn on the fifth line from the top line to the Altino dot matrix.
26 1 second delay
27
28 Shift the data variable one bit to the left and save it. 0x20 is stored in data.
29 Turn on the sixth line from the top line to the Altino dot matrix.
30 1 second delay
31
Shift the data variable one bit to the left and save it. 0x40 is stored in data.
33 Turn on the 7th line from the top row to the Altino dot matrix.
34 1 second delay
35
36 Shift the data variable one bit to the left and save it. 0x80 is stored in data.
37 Turn on the 8th line from the top row to the Altino dot matrix.
38 1 second delay
39 end setup
40
41 Starting the loop statement
42 End of loop statement
```

8.2.3 Display on Altino's dot-matrix by using Shift and byte operator

① After assigning 0x01 to each byte variable each byte will shift to left. then after calculating the OR operator it will display on the dot-matrix.

```
1   #include <Altino.h>
2
3   void setup() {
4
5    Serial.begin(115200);
6
7    byte data = 0x01;
8
9    DisplayLine(data, data, data, data, data, data, data, data);
10   delay(1000);
11
12   data = data << 1 | 0x01;
13   DisplayLine(data, data, data, data, data, data, data, data);
14   delay(1000);
15
16   data = data << 1 | 0x01;
17   DisplayLine(data, data, data, data, data, data, data, data);
18   delay(1000);
19
20   data = data << 1 | 0x01;
21   DisplayLine(data, data, data, data, data, data, data, data);
22   delay(1000);
23
24   data = data << 1 | 0x01;
25   DisplayLine(data, data, data, data, data, data, data, data);
26   delay(1000);
27
28   data = data << 1 | 0x01;
29   DisplayLine(data, data, data, data, data, data, data, data);
```

```
30    delay(1000);
31
32    data = data << 1 | 0x01;
33    DisplayLine(data, data, data, data, data, data, data, data);
34    delay(1000);
35
36    data = data << 1 | 0x01;
37    DisplayLine(data, data, data, data, data, data, data, data);
38    delay(1000);
39  }
40
41  void loop() {
42  }
```

1 : function to control Altino
2 :
3 : Start setup function
4 :
5 : Serial communication baud rate setting
6 :
7 : Declare the byte variable data and input hexadecimal number 0x01 as the initial value.
8:
9: Turn on the first line of the top line in the Altino dot matrix.
10: 1 second delay
11:
12: Shifts the data variable one bit to the left, stores it after OR operation with 0x01. 0x03 is stored in data.
13: Turn on the second line from the top line to the Altino dot matrix.
14: 1 second delay
15:
16: Shifts the data variable one bit to the left, stores it after OR operation with 0x01. 0x07 is stored in data.
17: Turn on the Altino dot matrix from the top row to the third row.
18: 1 second delay
19:
20: Shifts the data variable one bit to the left, stores it after OR operation with 0x01. 0x0f is stored in the data.
21: Turn on the top line to the fourth line on the Altino dot matrix.
22: 1 second delay
23:
24: Shifts the data variable one bit to the left, stores it after OR operation with 0x01. 0x1f is stored in the data.
25: Turn on the Altino dot matrix from the top row to the fifth row.
26: 1 second delay
27:
28: Shift the data variable one bit to the left and save it. 0x20 is stored in data.
29: Turn on the sixth line from the top line to the Altino dot matrix.
30: 1 second delay
31:
32: Shift the data variable one bit to the left and save it. 0x40 is stored in data.

```
33: Turn on the 7th line from the top row to the Altino dot matrix.
34: 1 second delay
35:
36: Shift the data variable one bit to the left and save it. 0x80 is stored
    in data.
37: Turn on the 8th line from the top row to the Altino dot matrix.
38: 1 second delay
39: end setup
40:
41: Starting the loop statement
42: End of loop statement
```

8.2.4 Output using Altino's Buzzer increment and decrement operator.

① Sound/output Altino's buzzer by using Increment and decrement operator after inputting 5 in integer variable.

```
1   #include <Altino.h>
2
3   void setup () {
4
5    Serial.begin(115200);
6
7    int biodata=5;
8
9    Sound(buzdata);
10   delay(500);
11
12   buzdata++;
13   Sound(buzdata);
14   delay(500);
15
16   buzdata++;
17   Sound(buzdata);
18   delay(500);
19
20   buzdata++;
21   Sound(buzdata);
22   delay(500);
23
24   buzdata--;
25   Sound(buzdata);
26   delay(500);
27
28   buzdata--;
29   Sound(buzdata);
30   delay(500);
31
32   buzdata--;
```

```
33    Sound(buzdata);
34    delay(500);
35
36    Sound(0);
37    delay(500);
38  }
39
40  void loop() {
41
42  }
```

1: function to control Altino
2 :
3: Start the setup function
4 :
5: Serial communication baud rate setting
6:
7: Declare the integer variable buzdata and input 5 as the initial value.
8 :
9: Output 5 values stored in buzdata to Altino buzzer.
10: 0.5 second delay
11: 10:
12: Increase buzdata variable by 1 and store 6 in buzdata.
13: Output 6 values stored in inedita to Altino buzzer.
14: 0.5 second delay
15:
16: Increase buzdata variable by 1 and store 7 in buzdata.
17: Output 7 values stored in ice data to Altino buzzer.
18: 0.5 second delay
19:
20: Increase buzdata variable by 1 and store 8 in buzdata.
21: Outputs 8 values stored in inedita to Altino buzzer.
22: 0.5 second delay
23:
24: Decrement 1 in buzdata variable and store 7 in buzdata.
25: Output 7 values stored in buzdata to Altino buzzer.
26: 0.5 second delay
27:
28: Decrement 1 in buzdata variable and store 6 in buzdata.
29: Altino Buzzer outputs 6 values stored in buzdata.
30: 0.5 second delay
31:
32: Decrement 1 in buzdata variable and store 5 in buzdata.
33: Output 5 values stored in buzdata to Altino buzzer.
34: 0.5 second delay
35:
36: Enter 0 from Altino to make no sound.
37: 0.5 second delay
38: end of setupfunction
39:
40: Start the loop statement
41:
42: end of loop statement

8.2.5 Move Altino using variation operator in rear motor

① Move Altino's rear motor using a conditional operator after inputting 5 in integer variable.

```
1  #include <Altino.h>
2
3  void setup() {
4
5    Serial.begin(115200);
6
7    int data=5;
8
9    data = data>3 ? 100 : 0;
10   Go(data,data);
11   delay(1000);
12
13   data = data<7 ? 100 : 0;
14   Go(data,data);
15 }
16
17 void loop() {
18
   }
```

1: function to control Altino
2 :
3: Start the setup function
4 :
5: Serial communication baud rate setting
6:
7: Declare the integer variable data and input 5 as the initial value.
8 :
9: conditional operator is a conditional operator that stores 100 in data if the value stored in data is greater than 3, otherwise stores 0 in data. 100 is stored in data.
10: The Altino rear-wheel motor operates at 100 speeds stored in the data using the Altino motor function.
11: 1 second delay
12:
13: conditional operator is a conditional operator that stores 100 in data if the value stored in data is less than 7, otherwise stores 0 in data. 0 is stored in data.
14: The Altino rear-wheel motor operates at zero speed stored in the data using the Altino motor function.
15: End of setup function
16:
17: Start the loop statement
18: end of loop statement

8.2.6 Control speed using the complex operator of Altino's rear motor.

① Control the rear motor after setting an integer variable to 100 and use a complex operator to increase the

speed up to 400 by incrementing by 100, then slow down.

```
1  #include <Altino.h>
2
3  void setup() {
4
5    Serial.begin(115200);
6
7    int data=100;
8
9    Go(data,data);
10   delay(500);
11
12   data+=100;
13   Go(data,data);
14   delay(500);
15
16   data+=100;
17   Go(data,data);
18   delay(500);
19
20   data+=100;
21   Go(data,data);
22   delay(500);
23
24   data-=100;
25   Go(data,data);
26   delay(500);
27
28   data-=100;
29   Go(data,data);
30   delay(500);
31
32   data-=100;
33   Go(data,data);
34   delay(500);
35
36   data-=100;
37   Go(data,data);
38 }
39
40 void loop() {

   }
```

1 : action function to control.
2 :
3 : setup function the start.
4 :
5 : serial communication rates settings.
6 :
7 : Enter the initial 100 and an integer variable data.
8 :

```
9 : albino motor is 100 action at a speed rear motor movements stored in the data using.
10 : half second delay.
11 :
To save the 100 in 12 : data data the 200 to save.
13: altino motor is 200 altino at a speed rear motor movement stored in the data using.
14: a half second delay.
15:
16: To save 100 16: data data to 300 stores.
17: altino motor is 300 altino at some speed rear motor movements stored in the data using.
18: a half second delay.
19:
20 :To save the one in 20 : data data to 400 stores.
21: altino motor is 400 altino at a speed rear motor movement stored in the data using.
22: a half second delay.
23:
24: To save the 100 in 24: data data to 300 stores.
25: altino motor is 300 altino at a speed rear motor movement stored in the data using.
26: half second delay.
27:
To save the 100 in the 28: data data the 200 to save.
29: altino motor is 200 altino at a speed rear motor movement stored in the data using.
30: a half second delay.
31:
32: To save the 100 in 32: data data to 100 to save.
33: altino motor is 100 altino at a speed rear motor movement stored in the data using.
34: a half second delay.
35:
36: To save a hundred in 36: data data to 0 to save.
37: altino motor is altino at zero speed rear motor behavior is stored in the data using.
38: the end of the setupfunction
39: Starting the 39: loop
40: loop the end of the door.
```

8.2.7 Display by using Altino's dot matrix's relational operators

① Using conditional operators, check whether the relational operator is true or false. Display 'O' if it's true and 'X' if it's wrong.

```
1  #include <Altino.h>
2
3  void setup () {
4
5    Serial.begin(115200);
6
7    int data1=3;
8    int data2=8;
9
10   data1 = (data1>1 && data1<5)? 79: 88;
11   Display(data1);
12   delay (2000);
13
14   data2 = (data2<1 || data2>9)? 79: 88;
15   Display(data2);
16 }
17
18 void loop () {
19
20 }
```

1: function to control Altino
2:
3: start set function
4:
5: Serial communication baud rate setting
6:
7: Declare the integer variable data1 and input 3 as an initial value.
8: Declare the integer variable data2 and enter 8 as an initial value.
9:
10: Connect Bluetooth with Altino (check COM port)
11: conditional operator is used. If the value stored in data1 is greater than 1 and less than 5, store 79 (English character 'O') in data1; otherwise store 88 (English character 'X'). Because it is true, 79 is stored in data1.
12: Output 79 (English character 'O') stored in data1 to the Altino dot matrix.
13: 2 second delay
14:
15: conditional operator. If the value stored in data2 is less than 1 or greater than 8, store 79 (English character 'O') in data2, otherwise store 88 (English character 'X'). Because it is false, it stores 88 in data2.
16: Output 88 (English character 'X') stored in data1 to the Altino dot matrix.
17: end of set function
18:
19: Starting the loop statement
20: end of loop statement

8.3 Practice the operator application using Altino

Try these:

8.3.1 Let us use a shift operator to move the dot matrix from bottom to top.

8.3.2 Use shift operations to turn on all dot matrixes and turn off one line at a time.

8.3.3 Turn on the dot matrix one line at a time.

8.3.4 Write a program that uses the increment operator on the dot matrix to increment from 0 to 9 and then decrement back to 0.

8.3.5 Decrease the rear-wheel motor by 200 (0 ~ 1000 ~ 0).

9 . Learning Conditional Statements Using ALTINO

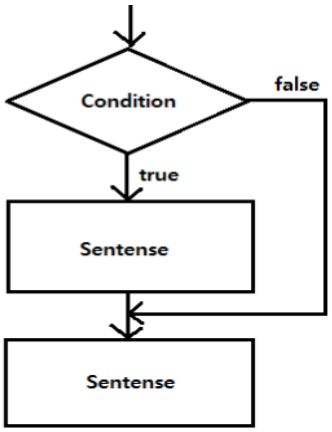
The conditional statement is an essential element in every programming language. Various conditional statements enable more complex control to be implemented. The condition given in the conditional statement determines how the program works.

9.1 Understanding if statements

9.1.1 if statement

General format

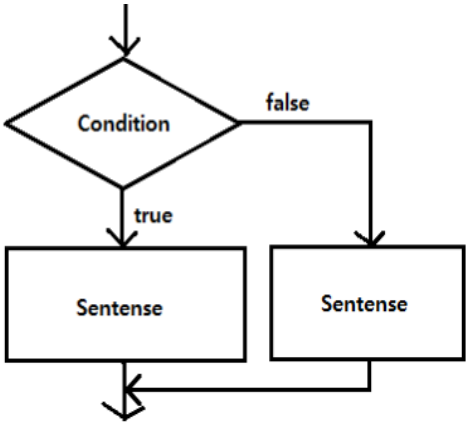
```
if (conditional equation) {
        statement 1;
        statement 2;
}
```



9.2 if else statement

General format

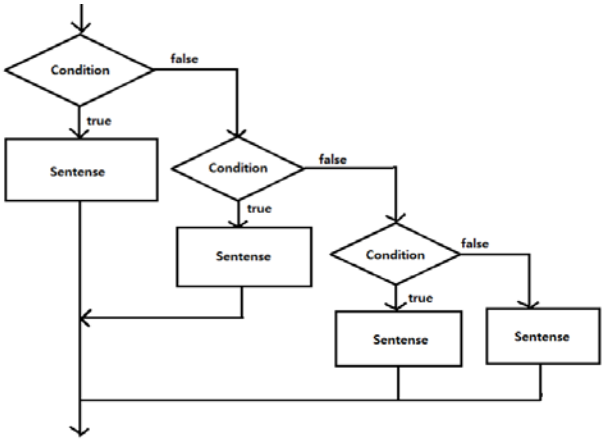
```
if (conditional equation) {
        statement 1;
}
else {
        statement 2;
}
```



9.3 Understanding compound if else statement

General format

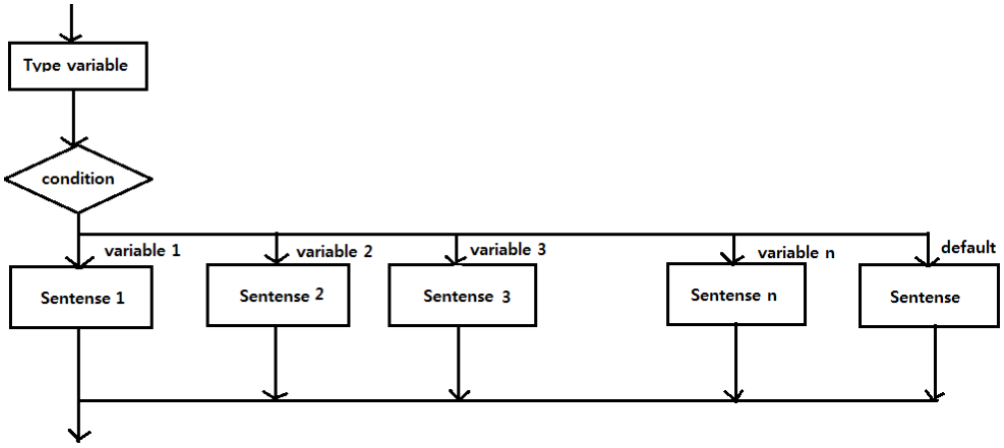
```
if (conditional equation) {
        statement 1;
}
else if (conditional equation) {
        statement 2;
}
else if (conditional equation) {
        statement 3;
}
else {
        statement 4;
}
```



9.4 Understanding switch case statement

General format

```
switch (variable) {
case variable 1:    statement 1;
            break;
case variable 2:    statement 2;
            break;
case variable n:    statement n;
            break;
default:  statement;
}
```



9.5 Exercise C++ programming with ALTINO

9.5.1 Receiving integer, Output 'A' in the dot matrix if 1 is entered

① Add the following source code.

```
1   #include "Altino.h"
2   Sensor Data sdata;
3
4   setup () {
5           Serial.begin(115200);
6   }
7
8   loop () {
9           sdata=Sensor (1);
10          if (sdata. CDS Sensor < 200)
11          {
12                  Display('A');
13          }
14  }
```

1: function to control Altino
2: Declaring a structure variable to receive the sensor value
3:
4: Start the setup function
5: Serial communication baud rate setting
6: end of setupfunction
7:
8: Start the loop statement
9: Accepts the value of the Altino sensor. (When the parameter value is set to 1, 6 IR sensor values, rear motor torque value, temperature sensor, and illuminance sensor value is obtained. Sensor XYZ axis, steering value, steering torque value).
10: If the illuminance sensor value is less than 200
11: Start the if statement
12: Output 'A' character to the Altino dot matrix
13: end of if statement
14: end of loop statement

① If the illuminance sensor value is greater than 200, no sound is produced. Otherwise, the Altino sound level is output to 37.

```
1   #include "Altino.h"
2   Sensor Data sdata;
3
4   setup () {
5           Serial.begin(115200);
6   }
7
8   loop () {
9           sdata=Sensor (1);
10          if (sdata. CDS Sensor > 200)
11          {
12                  Sound (0);
13          }
14          else
15          {
16                  Sound (37);
17          }
18  }
```

1: function to control Altino
2: Declaring a structure variable to receive the sensor value
3: Start the setup function
4: Serial communication baud rate setting
5: end of setupfunction
6:
7: Start the loop statement
8: Accepts the value of the Altino sensor (If you input 1 as the parameter value, 6 IR sensor values, rear motor torque value, temperature sensor, and illuminance sensor value are obtained. Sensor XYZ axis, steering value, steering torque value).
9: If the illuminance sensor value is greater than 200
10: Start the if statement
11: Does not output sound to Altino buzzer.
12: end of if statement
13: Otherwise
14: Start the else statement
15: Output sound level 37 to the Altino buzzer.
16: end of else statement
17: end of loop statement

9.5.2 Print the following in a dot matrix (repeated conditional).

- If IR sensor 1 is greater than 300, output '1' to dot matrix
- If IR sensor 2 is greater than 300, output '2' to the dot matrix
- If IR sensor 3 is greater than 300, output '3' to dot matrix
- Otherwise, output 'X' to the dot matrix

① If IR sensor #1 is greater than 300, '1' is output to the dot matrix. If IR sensor # 2 is greater than 300, '2' is output to the dot matrix. If IR sensor # 3 is greater than 300, '3' is output to the dot matrix.

```
1   #include "Altino.h"
2   Sensor Data sdata;
3
4   setup () {
5           Serial.begin(115200);
6   }
7
8   loop () {
9           sdata=Sensor (1);
10          if (sdata. IRSensor [0] > 300)
11          {
12                  Display ('1');
13          }
14          else if (sdata. IRSensor [1] > 300)
15          {
16                  Display ('2');
17          }
18          else if (sdata. IRSensor [2] > 300)
19          {
20                  Display ('3);
21          }
22          else
23          {
24                  Display('X');
25          }
26  }
```

1: function to control Altino
2: Declaring a structure variable to receive the sensor value
3:
4: Start the setup function
5: Serial communication baud rate setting
6: end of setupfunction
7:
8: Start the loop statement
9: Accepts the value of the Altino sensor. (When the parameter value is set to 1, 6 IR sensor values, rear motor torque value, temperature sensor, and illuminance sensor value is obtained. Sensor XYZ axis, steering value, steering torque value).
10: If the IR sensor value of 1 is greater than 300
11: Start the if statement
12: Output '1' character to the Altino dot matrix
13: end of if statement
14: If the IR sensor value of No. 2 is greater than 300
15: Start the else if statement
16: Output '2' character to the Altino dot matrix
17: else if statement end
18: If the IR sensor number 3 is greater than 300
19: Start the else if statement
20: Output '3' character to the Altino dot matrix
21: else if statement end
22: Otherwise
23: Start the else statement
24: Output 'X' character to the Altino dot matrix

```
25: end of else statement
26: end of loop statement
```

9.5.3 Output at the sound level according to the value of the number given by the ambient light sensor.

- 100 < sdata 200, indicates sound level 37
- 200 < sdata 300, sound level 39
- 300 < sdata 400, sound level 41
- 400 < sdata 500, sound level 42
- 500 < sdata 600, sound level 44
- 600 < sdata 700, sound level 46
- 700 < sdata 800, sound level 48
- 800 < sdata 900,, sound level 49
- Other numeric sound level 0

Output the sound level according to the digit number level of the ambient light sensor 100. (switch case)

```
1  #include "Altino.h"
2  Sensor Data sdata;
3
4  setup () {
5          Serial.begin(115200);
6  }
7
8  loop () {
9          sdata=Sensor (1);
10         switch (sdata. CDS Sensor/100)
11         {
12         case '1':
13                 Sound (37);
14                 break;
15         case '2':
16                 Sound (39);
17                 break;
18         case '3':
19                 Sound (41);
20                 break;
21         case '4':
22                 Sound (42);
23                 break;
24         case '5':
25                 Sound (44);
26                 break;
27         case '6':
28                 Sound (46);
29                 break;
30         case '7':
31                 Sound (48);
32                 break;
33         case '8':
34                 Sound (49);
```

```
35                      break;
36              default:
37                      Sound (0);
38                      break;
39              }
40  }
```

1: altino function to control.
2: for the sensor value over a structure like variable declaration.
3:
4: setup function the start.
5: serial communication rates settings.
6: the end of the setupfunction.
7:
8: Starting the loop
9: Enter the value received. (factor value of 1, Sensor altino ir six sensors, rear motor torque value, temperature sensor, light sensor value is obtained and the Son of value to 2. Triaxial acceleration sensor xyz, terrestrial magnetism sensor, steering torque value is obtained and the value of the steering shaft xyz.
10: switch case statements, and switch light sensor in the door 100 digits based upon the level of outputs, the sound.
11: switch case the door started.
12: If the number 100 = 1.
13: altino 37 levels of output in the buzzer.
14: switch case out the door.
15: In the case of two digits of the 100.
16: altino level 39, output in the buzzer.
17: switch case out the door.
18: 100 digits of the three cases.
19: altino 41 level of output in the buzzer.
20: switch case out the door.
21: if the 4 digits of the 100.
22: altino 42 level of output in the buzzer.
23: switch case out the door.
24: if five digits of the 100.
25: altino 44 level of output in the buzzer.
26: switch case out the door.
27: 100 digits of the case of six.
28: altino 46 level of output in the buzzer.
29: switch case out the door.
30: 7, 100 digits.
31: altino 48 level of output in the buzzer.
32: switch case out the door.
33: If 8, number 100.
34: altino 49 level of output in the buzzer.
35: switch case out the door.
36: If the value of the number of other value of the 100.
37: altino buzzer sound not output.
38: switch case out the door.
39: switch case the end of the door.
40: loop the end of the door.

9.6 Application of C++ programming with ALTINO

9.6.1 If it is bight. Display "●" at Dot-matrix or If it is dark, Display "o" at Dot-matrix

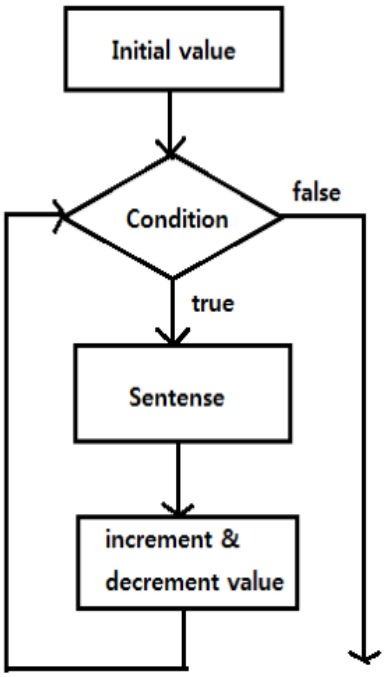
9.6.2 Try to move Altino slowly in bright light and stop in the dark.

9.6.3 Make the sound change according to brightness in the room.

10 . Learning Iteration Statement by Using ALTINO

An iteration statement makes the code to repeat a specified number of times. This is to import a command by means of an iteration command without stating a command a plurality of times to execute the same command a plurality of times. It is important because it allows you to repeat the function of a block of code without needing to repeat the actual code over and over.

10.1 Understanding while statement

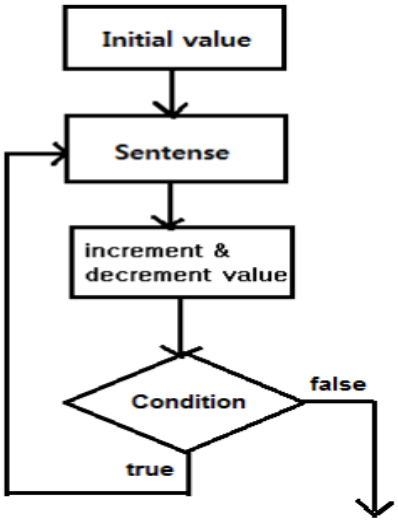


General format

```
variable= initial value;
while (conditional equation) {
        statement;

        variable=increment value;
}
```

10.2 Understanding do while statement



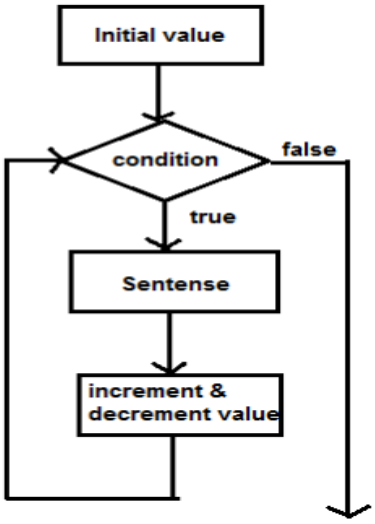
General format

```
variable= initial value
do {
        statement;
        variable=increment value;
}
while (conditional equation);
```

10.3 Understanding for statement

General format

```
for (variable= initial value; conditional equation;
variable = increment value)
{
        statement;
}
```



- Usability of the for statement

It is not necessary to use more than one equation.
The first equation of for statement is not necessary to be an initialization equation.
The empty for statement is always thought as true and infinitely repeats.

10.4 Break statement

* This statement is used to stop the control statement in execution regardless of the number of iterations originally established.
* It can even stop the execution of the iteration statement currently in progress

10.4.1 Continue statement

- Stop the execution of the current sequence in the loop and execute the next sequence
- means to skip one number of repetitions and to execute the next number of repetitions

10.5 Pre-processor

1) #define
- swap macro name for a real value, which is called a macro swap.
- define a macro constant (character constant) or macro function.
- Use capital letters for the macro name to avoid confusion with variables or function name.
- Characters rather than numbers are clear in meaning and easy to read.
- The same constant is frequently used and easy to change.
- Define and use very simple macro functions.

2) #typedef
- This enables you to change the data type to one you made.
- This is used to change difficult declarations to simple ones.
- Use capital letters to avoid confusion with other variables.

3) Purpose of pre-processor
- Pre-processing before actual compiling
- A compiler pre-processes a specific process.
- Possibility of joint development
- Separation of code for debugging from code for release
- Not essential but necessary for efficiency

4) Declaration of a global variable

- One header file can be referenced from a plurality of source files.
- Errors occur because one global variable declaration is executed in several locations.
- Better to declare it in the source file than the header file.

5) Tips for using #define statement
- Do not use semicolons (“; ") at the end of statement.
- Using values not currently defined is not allowed.
- Use "(" and ")" for processing it for operation.
- Use capital letters for the name newly defined and represented.
- Example #define MAX 100
  #define MIN (MAX - 90)
  #define AUTHOR_NAME "Kang Suk Bum"

10.6 Practice loops using Altino

10.6.1 Repeat SAEON to the dot matrix.

① You can always put 1 in the while statement to make it repeat (over and over forever) and print "SAEON" characters to the dot matrix.

```
1   #include <Altino.h>
2
3   void setup () {
4
5    Serial.begin(115200);
6
7    while (1)
8    {
9      Display('S');
10     delay (500);
11     Display('A');
12     delay (500);
13     Display('E');
14     delay (500);
15     Display('O');
16     delay (500);
17     Display('N');
18     delay (500);
19   }
20  }
21
22  void loop () {
23
    }
```

```
1: function to control Altino
2:
3:  Start the setup function
4:
5: Serial communication baud rate setting
6:
7: Since the condition of the while statement is 1, it is always true and repeats infinitely.
8: Start the while statement
9: The letter 'S' is output to the Altino dot matrix.
10: Delay time 0.5 seconds
11: The letter 'A' is output to the Altino dot matrix.
12: Delay time 0.5 seconds
13: The letter 'E' is output to the Altino dot matrix.
14: Delay time 0.5 sec
15: The letter 'O' is output to the Altino dot matrix.
16: delay time 0.5 seconds
17: The letter 'N' is output to the Altino dot matrix.
18: delay time 0.5 seconds
19: end of while statement
20: End of setup function
21:
22: Starting the loop statement
23: End of loop statement
```

10.6.2 Using FOR statement 0~9 to output them in dot matrix

① Add the following source:

```
1   #include <Altino.h>
2
3   void setup () {
4
5    Serial.begin(115200);
6
7    int i;
8
9    for (i = 0; i < 10; i++)
10   {
11     Display (48 + i);
12     delay (500);
13   }
14  }
15
16  void loop () {
17
    }
```

```
1: function to control Altino
2:
3: Start the setup function
4:
5: Declare Serial Communication Baud Rate
6:
7: Declaration of integer variable i
8:
9: for statement, I increment the value of i from 0 to 1, repeat the contents of the
   braces, and if it becomes smaller than 10, stop the iteration and leave the braces.
10: Start the for statement
11: Output according to the value of i in the matrix of the aluminum dot.
12: Delay time 0.5 seconds
13: end of for statement
14: Start the setup function
15:
16: Start the loop statement
17: end of loop statement
```

10.7 Practice loop statements using Altino

10.7.1 Display a capital letter on dotmatrix.

10.7.2 Display inputting character (Q) on dotmatrix.

10.7.3 Slowly light up dotmatrix by lighting one line each loop.

10.7.4 Move right on dotmatrix by lighting one line each loop.

10.7.5 Write and repeat your own name on dotmatrix.

11 Understanding the array using ALTINO

11.1 Understanding arrays

11.1.1 One-dimensional array

- Array must be applied only to variables with the same data type
- made in consecutive locations (addresses) within the main memory location
- Declaration format
  Data type Array Name [Array size]

11.1.2 Two-dimensional array

- Declaration format
Data type Array Name [one-dimensional size] [two-dimensional size];
- total number of arrays = one dimensional dimension X two-dimensional dimension

11.1.3 N-dimensional array

- Declaration format
Data type Array Name [1D size] [2D size] [3D size] ..... [nD size];
- total number of arrays = one dimensional dimension X two-dimensional dimension X three-dimensional dimension X ..... [n dimensional size]

If the array is n in size, the range is from 0 to n-1.

11.2 Practice arrays using Altino

11.2.1 Program the speed change in the array and accelerate and decelerate the speed.

① Put an array in motor speed to put the value and use the for loop to speed up and slow down.

```
1  #include <Altino.h>
2
3  void setup () {
4
5   Serial.begin(115200);
6
7   int motor speed [11] = {0, 100, 200, 300, 400, 500, 400, 300, 200, 100, 0};
8   int i;
9
10  for (i = 0; i < 11; i++)
11  {
12    Go (motor speed[i], motor speed[i]);
13    delay (1000);
14  }
15 }
16
17  void loop () {
18  }
```

1: function to control Altino
2:
3: Start the setup function
4:
5: Declare Serial Communication Baud Rate
6:
7: Create an integer array in which you can enter 11, and enter the initial value.
8: Create an integer variable i to perform the loop.

9:
10: Repeat from 0 to 10 using the for statement.
11: Start the for statement
12: Accelerates / decelerates the rear wheel motor to the value in the array.
13: Delay time 1 second
14: end of for statement
15: End of setup function
16:
17: Start the loop statement
18: end of loop statement

11.2.2 Type a character in the array and print the input character.

① Outputs a string to an Altino dot matrix using an iteration with a character as the initial value in the character array.

```
1   #include <Altino.h>
2
3   void setup () {
4
5    Serial.begin(115200);
6
7    char name[]="kangsukbum";
8    int i;
9
10   for(i=0; i<10; i++)
11   {
12    Display(name[i]);
13    delay(500);
14   }
15  }
16
17   void loop() {
18   }
```

1: function to control Altino
2 :
3: Start the setup function
4 :
5: Serial communication baud rate setting
6:
7: Make a character variable into an array and enter a string.
8: Create an integer variable i to perform the loop.
9:
10: Repeat from 0 to 9 using the for statement.
11: Start the for statement
12: The output is output to the Altino dot matrix using a loop.
13: delay time 0.5 sec
14: end of for statement
15: End of setup function
16:
17: Start the loop statement
18: end of loop statement

11.3 Practice of array application using Altino

11.3.1 Control the steering by putting the steering values into an array.

11.3.2 Store and output 10 LEDs in the array.

11.3.3 Save your initials in a character array and print them.

11.3.4 Create two arrays to move the robot in a rectangle.

12 . Understanding the function using ALTINO

12.1 Understanding function

12.1.1 Special feature

- Once created, the function can be used multiple times when needed.
- Because functions are independent, they can be broken down into functions to solve problems.
- Because the function is independent, it is easy to fix errors.
- The function can return the result to the calling program.

12.1.2 Advantages

- Make the program easier to read.
- Maintenance / repair becomes easy.
- Increase program efficiency.
- Easy to code.

12.1.3 Using function

- Create a specific function and call it on demand
- use library function

12.1.4 Local vs global variables

- Local variables (local)

Variables declared within a function or block

● Variable value is retained only in declared function or block,

- global variable (global)

● declaration outside of function
● Variable values apply to one function or several functions
● Maintain variable values even outside function
• The value of the global variable is retained until the execution of the program is completed (memory is occupied)

12.1.5 Apply to multiple functions

- Maintain variable values even outside functions
- The value of the global variable is maintained until the end of the program execution (memory occupation)

Place of declaration	Memory specifier	Memory location	Life span	Scope
External of function	No	Static scope	Invariable	Global area of all modules
	Static	Static scope	Invariable	Global area of declared modules
	Extern	External variables defined in other locations can be used in global area of the module after this declaration		
Internal of function	Auto or no	Stack	Disappear	In declared blocks
	Register	Register or stack	Disappear	In declared blocks

	Static	Static scope	Invariable	In declared blocks
	Extern	External variables defined in other locations can be used in global area of the module after this declaration		

12.2 ALTINO control function

void Open (char *szport)	Bluetooth connection function
void Close ()	Bluetooth disconnection function
void Go (int left, int right)	rear motor control function
void Sound (unsigned char buzzer)	buzzer control function
void Display (unsigned char ASCII)	dot-matrix control function
void Steering (int steering value)	steering control function
void DisplayLine (unsigned char dot0, unsigned char dot1, unsigned char dot2, unsigned char dot3, unsigned char dot4, unsigned char dot5, unsigned char dot6, unsigned char dot7)	dot-matrix line control function
void Led (unsigned char led)	8 LED control function
Sensor Data Sensor (int command)	Sensor control function

12.3 Function exercises using Altino

12.3.1 Use the Go function to create and use Go2(int sp) function.

① Use Go2 function and make your own function to operate the left motor of the rear wheel motor and the right motor at the same speed.

```
1  #include <Altino.h>
2
3  void Go2(int sp)
4  {
5          Go (sp, sp);
6  }
7
8  void setup () {
```

```
9
10    Serial.begin(115200);
11
12    Go2(300);
13    delay (1000);
14    Go2(0);
15  }
16
17    void loop () {
18    }
```

1: function to control Altino
2:
3: It is a function to control the Altino robot rear left motor and right motor by inputting the integer. It can rotate forward and backward, reverse backward, and adjust in the range of 0 ~ 1000
4: Start Go2function
5: Go function
6: Go2function end
7:
8: Start setupfunction
9:
10: Serial communication transmission speed setting
11:
12: Advance to rear-wheel motor speed 300
13: 1 second delay
14: Rear motor stop
15: end of setupfunction
16:
17: Start the loop statement
18: end of loop statement

12.3.2 Create a function that creates a heart shape.

① Neurofunction to make the heart shape and Neurofunction to eliminate the heart shape so it blinks repeatedly.

```
1   #include <Altino.h>
2
3   void Heart On ()
4   {
5           DisplayLine(0x18,0x3c,0x7e,0xfc,0xfc,0x7e,0x3c,0x18);
6   }
7
8   void Heart Off ()
9   {
10          DisplayLine (0,0,0,0,0,0,0,0);
11  }
12
```

```
13  void setup () {
14
15    Serial.begin(115200);
16
17  }
18
19  void loop () {
20    HeartOn();
21    delay(500);
22    Heart Off ();
23    delay (500);
24  }
```

1: altino function to control.
2:
3: altino dote dot-matrix each of the row by setting the heart-shaped function to make.
4: heartonfunction Starting.
5: DisplayLine function
6: the end of the heartonfunction
7:
8: altino doteu of dot-matrix each column by setting the heart shape to get rid of the function.
9: Starting heartofffunction
10: DisplayLine function
11: the end of the heartofffunction
12:
13: Starting setupfunction
14:
15: Serial communication transfer rate settings.
16:
17: the end of the setupfunction
18:
19: Starting the Gate loop
20: Hearts on function the call.
21: a half second delay.
22: Hearts off function the call.
23: a half second delay.
24: loop the end of the door.

12.3.3 Obtain a function that accepts a buzzer sound and a dot matrix and outputs them at the same time.

```
1   #include <Altino.h>
2
3   void BuzzerDotmatrix (unsigned char buz, unsigned char dot)
4   {
5           Sound(buz);
6           Display(dot);
7   }
8
9   void setup () {
10
11    Serial.begin(115200);
```

```
12
13  }
14
15  void loop () {
16    BuzzerDotmatrix (5, 'O');
17    delay (500);
18    BuzzerDotmatrix (0, 'X');
19    delay (500);
20  }
```

```
13  Function to control Altino
14
15  Create a BuzzerDotmatrixfunction that uses a buzzer and a dot matrix together.
16  Start BuzzerDotmatrixfunction
17  Sound function
18  Display function
19  BuzzerDotmatrixfunction end
20
21  Start setupfunction
22
23  Serial communication baud rate setting
24
25  End setup
26
27  Start loop statement
28  Using BuzzerDotmatrixfunction, the buzzer will print 5 and the dot matrix will print 'O'.
29  Delay time 0.5 seconds
30  Using BuzzerDotmatrixfunction, the buzzer will print 0 and the dot matrix will print 'X'.
31  Delay time 0.5 seconds
32  End of loop statement
```

12.4 Function application exercise using Altino

12.4.1 Make GO (), Back (), Left (), Right () functions.

12.4.2 Make a function to control dotmatrix and rear motor together.

12.4.3 Make a function to display on dotmatrix from 0 to 9.

12.4.4 Make a function to display 8 LEDs and control steering together.

12.4.5 Make a function to display 8 LEDs and control rear motor together.