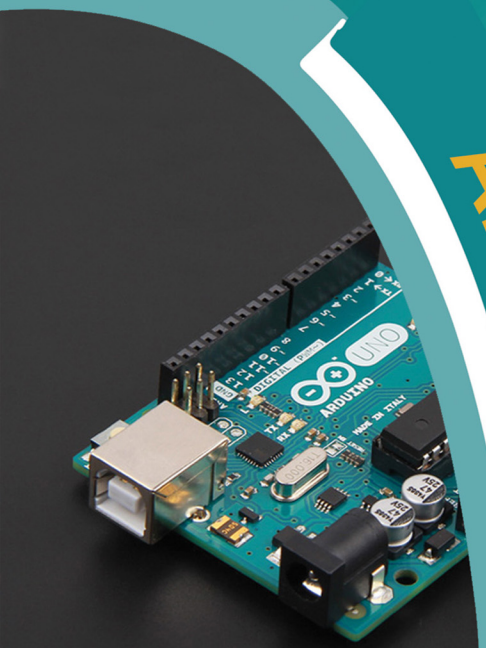
الآردوينو

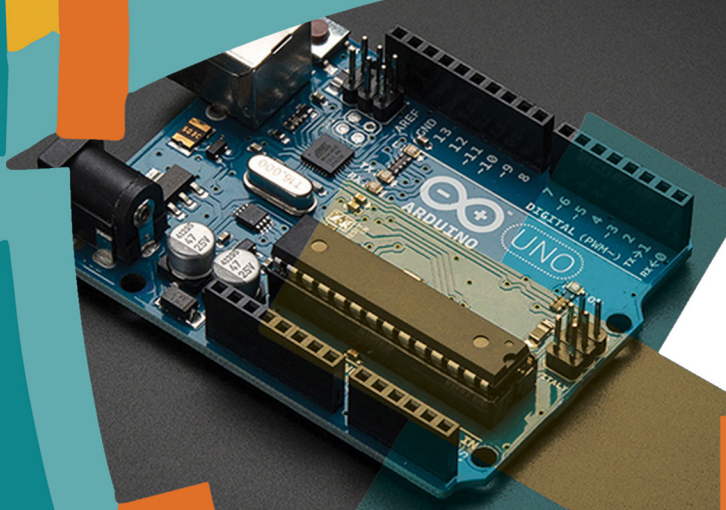
كما لم تعرفه من قبل!!

حـــزمة تـــعليمية في:
عالم الآردوينو مع أمثلة
وشروحات متقدمة

Arduino

م. محمود مسلماني





الجزء الأول

الآردوينو
ARDUINO

كما لم تعرفه من قبل!!

م. محمود مسلماني

مقدمة المؤلف:

بفضل من الله وكرمه وبعونه تعالى أتممت الجزء الأول من كتابي " **الآردوينو .... كما لم تعرفه من قبل**" والذي جمعت فيه خلاصة تجربتي في مجال البرمجة مستفيدا من خبرتي الطويلة في التعليم البرمجي، فأدركت مكامن الضعف وأصبت كبد الخلل فعملت على سد الفجوة وردم البؤرة الموجودة لدى الطالب المقبل على تعلم لغة العصر لغة البرمجة ونسف الصورة المسبقة عن صعوبة لغة البرمجة.

كتابي هذا لن يكون كتابا تقليديا ولا سردا مملا - فكلنا يعرف مدى الشهرة الواسعة التي اكتسبتها لوحات الآردوينو منذ انطلاقتها والدروس والمحاضرات التي سجلت عنها- لكن سيكون أكاديميا في سرده وترتيبه للأفكارعمليا ملامسا للواقع في طرح الأمثلة وحلها، ففي الجزء الأول من الكتاب سعيت لجعله تأسيسيا للطالب فركزت فيه على لغة البرمجة C++ كونها اللغة التي سوف نتعامل بها وقدمت أمثلة وفيرة تجمع في طياتها بساطة الدارات وعبقرية البرمجة مستشفا ذلك من الملاحظات والتوجيهات والنتائج العملية للطلاب الذين تم إعطاؤهم هذا الكتاب قبل نشره، وفي النهاية أسأل الله تعالى القبول والسداد.....

أخوكم المحب

م. محمود مسلماني

الطبعة الأولى: ٢٠١٧

الشكر موصول لـ .....:

- شركة ابتكار للحلول الهندسية لدعمها اللكبير على إنهاء هذا العمل وتزويدنا بأفضل المنتجات لإتمام هذا العمل.

https://fb.me/EptekarEP18

https://t.me/eptekar

https://www.youtube.com/channel/UCkUvPLRxt3jrqxfSuLW39Tg?view_as=sub

مكتبة أمجاد بما قدمته من خدمات في مجال الطباعة والتصميم لتقديم هذا المحتوى بأجمل صيغة وأفضل عرض فجزاهم الله كل خير.

https://t.me/Amjad_books

ألا يا لقومٍ هلْ لما حمّ دافعُ؟ وهلْ ما مضى من صالح العيش راجعُ

تذكّرْتُ عصْراً قد مَضَى فتهافَتَتْ بناتُ الحشا، وانهلّ مني المدامعُ

صَبَابَةُ وَجْدٍ ذكّرَتني أحِبّةً وقَتلى مَضَوْا فيهِمْ نُفَيْعٌ وَرَافِعُ

وسعدٌ فأضحوا في الجنانِ وأوحشتْ منازِلُهُمْ والأرْضُ منهُمْ بَلاقِعُ

وَفَوْا يَوْمَ بَدْرٍ للرّسولِ، وفَوْقَهُمْ ظِلالُ المَنَايَا والسّيوفُ اللّوَامِعُ

دعا فأجابوهُ بحقٍّ، وكلهمْ مُطِيعٌ لهُ في كلّ أمْرٍ وسامِعُ

فَما بَدّلوا حتّى تَوَافَوْا جَماعَةً ولا يَقْطَعُ الآجَالَ إلاّ المَصَارِعُ

لأنهمُ يرجونَ منهُ شفاعة إذا لمْ يكنْ إلا النبيينَ شافعُ

وذلكَ، يا خيرَ العبادِ، بلاؤنا وَمَشْهَدُنا في اللَّهِ، والمَوْتُ ناقِعُ

لَنَا القَدَمُ الأولى إليكَ، وَخَلْفُنا لأوّلِنَا، في طَاعَةِ اللَّهِ تَابِعُ

ونعلمُ أنّ الملكَ للهِ وحدهُ وَإنّ قَضَاءَ اللَّهِ لا بُدّ وَاقِعُ

الفصل الأول:

نظرة عامة عن الآردوينو

لوحات الآردوينو .... النشأة والتطور

ما هو الآردوينو Arduino؟؟؟

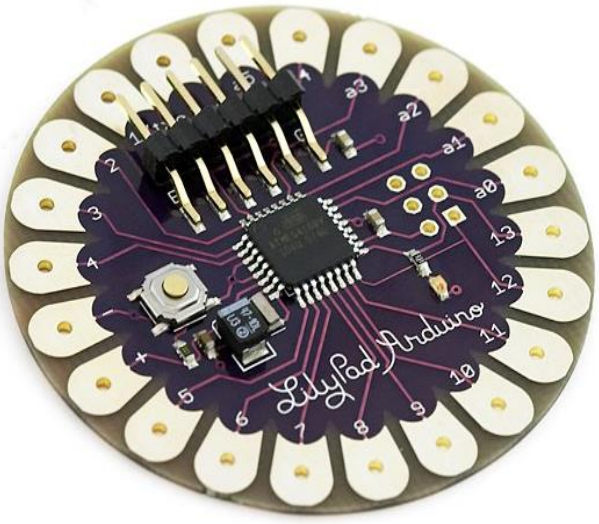
Arduino: هو كومبيوتر صغير الحجم بإمكانه التفاعل و التحكم في الوسط المحيط به بشكل أفضل من الكومبيوتر المكتبي Desktop أما تقنيا فهو منصة Platform برمجية مفتوحة المصدر تتكون من متحكم إلكتروني **Micro-Controller** وبيئة تطويرية تكاملية لكتابة البرمجيات**IDE** .

قوة الآردوينو **Arduino** تتجلى في قدرته الكبيرة على التواصل مع القطع الإلكترونية الأخرى كالمحولات Switches أو الحساسات Sensors والاستفادة منها في الحصول على مختلف البيانات كدرجة الحرارة والرطوبة أو شدة الإضاءة وكذلك فاعليته الكبيرة في التحكم في المحركات Motors وديودات الإصدار الضوئي LED والكثير من القطع الإلكترونية.

يمكن تشغيل مشاريع الآردوينو **Arduino** عن طريق وصله بالكومبيوتر وجعله يتعامل مع أحد البرامج الموجودة على الجهاز أو بالإمكان تشغيله باستقلالية تامة.

تعد لوحات الآردوينو **Arduino** الأشهر عالميا لما تقدمه من سهولة في التعامل وتنوع الأفكار والمشاريع والبساطة في التطبيق والتكلفة المادية المنخفضة نسبيا لكن كيف ظهرت فكرت هذه اللوحات **وكيف عرفت طريقها للنجاح .... ؟؟؟**

تبدأ القصة عام 2005 في مدينة ايفريا الإيطالية حيث قام كل من ماسيمو بانزي بالتعاون مع دايفيد كوارتيليس و جاينلوكا مارتينو بإطلاق مشروع **Arduin of Ivrea** وتم تسمية المشروع باسم أشهر شخصية تاريخية في المدينة وكان الهدف الأساسي للمشروع هو عمل بيئة تطوير للمتحكمات دقيقه بصوره مفتوحه المصدر 100 في المئة وتضمن هذا المشروع عمل بيئة تطوير برمجيه للمتحكمات الدقيقة **Integrated Development Environment(IDE)** وتكون مجانيه في ذات الوقت كما تضمن عمل لوحات تطوير Development Boards صغيره الحجم بتكلفه بسيطة تبلغ حالياً قرابة 27 دولار ليتمكن الطلاب والهواة التقنين تحمل سعرها، وحتى عام 2013 تم شحن أكثر من 700 ألف لوحة آردوينو.



لماذا الآردوينو .... ؟؟؟

في الحقيقة يوجد الكثير من المتحكمات الإلكترونية **Micro-Controllers** المتوفرة في السوق مثل **Parallax** و **Basic Stamp** و **Raspberry Pi** و كلها تتميز بإمكانيات قوية و لها القدرة على التحكم في مختلف القطع الإلكترونية و البرمجيات Software و ذلك طبعا بنسبة أفضلية متفاوتة لكن ما يميز الآردوينو **Arduino** هو مجموعة من الأمور التي تصنع الفارق بينه و بين غيره أهمها:

- **البساطة:** قطعة الآردوينو **Arduino** مصممة لتناسب احتياجات الجميع، محترفين، أساتذة، طلاب و هواة الإلكترونيات التفاعلية.
- **الثمن:** لوح الآردوينو **Arduino** أقل ثمناً مقارنةً مع الألواح الأخرى من نفس النوع فثمن أغلى **Arduino** لا يتجاوز 50 $
- **التركيب الذاتي(Self-Assembly):** يمكنك تحميل ورقة البيانات Datasheet الخاصة بالآردوينو **Arduino** مجاناً من الموقع الرسمي وشراء القطع وتركيبه بنفسك.
- **متعدد المنصات:** برنامج الآردوينو له القدرة على العمل على الويندوز **windows** ، **Mac OS** و اللينكس **Linux** بينما أغلب المتحكمات الإلكترونية الأخرى تعمل فقط على الويندوز.

- **بيئة برمجية سـهلة وبسـيطة:** البيئة البرمجية Programming Environment مصممة لتكون سهلة للمبتدئين وثابتة وقوية للمحترفين.

- **Open Source Software:** مكتوب بلغة C++ و متاح للجميع لتحميله و بإمكان المبرمجين التعديل عليه وفق احتياجاتهم.

- **Open Source Hardware :** الآردوينو **Arduino** مصنوع أساساً من متحكمات ATMEGA328 و ATMEGA168 و المخططات منشورة تحت ترخيص Creative Commons مما يتيح إلى مصممي الدارات الإلكترونية Electronic Circuits تصميم داراتهم الخاصة.

ماذا نقصد بمفتوح المصدر Open Source ؟؟؟

مخططات تصميم العتاد **Hardware Schema** الخاصة بالآردوينو **Arduino** متاحة للجميع لتحميلها ودراستها لفهم مبدأ عمل القطعة والتعديل عليها وكذلك إمكانية الاستفادة منها تجارياً كذلك الكود المصدري الخاص ببرنامج **Arduino** مفتوح المصدر ومتوفر بترخيص GPL.

لوحات الآردوينو .... التعدد والميزات

منذ انطلاق مشروع الآردوينو وحتى وقتنا الحالي تم إنتاج العديد من التصاميم واللوحات والهدف من ذلك تأمين احتياجات المشاريع المتعددة وبناء لوحات تتناسب مع التطبيق المطلوب من حيث السرعة وعدد المداخل والمخارج والسعر وأمور أخرى لها علاقة بالبنية الداخلية للمتحكم Microcontroller الموجود على لوحة الآردوينو حتى بلغ عدد لوحات الآردوينو على ما يزيد عن 40 نوع بعضها يتشابه في المتحكم ويختلف في الحجم والملحقات المتوفرة على لوحة الآردوينو نفسها، لكن ما يجمع كل لوحات الآردوينو أنها تقسم في بنيتها إلى أربعة أقسام عامة:

- **قسم المتحكم**: والذي يضم المتحكم وكل ما يلزم لعمل المتحكم بالشكل الصحيح (الكرستالة الخارجية مع المكثفات الخاصة بها، كباس إعادة الضبط RESET، مكثفات الاستقرار الكهربائي على مداخل التغذية، أرجل أقطاب الدخل والخرج العامة).
- **قسم الاتصال مع الحاسب:** وهي شريحة لتأمين الاتصال بين الحاسب ولوحة الآردوينو أي بين البروتوكولين USB < > UART (غالبا تكون هذه الشريحة إما IC: FTDI أو متحكم ATmega8).
- **قسم التغذية**: لتأمين وضبط تغذية المتحكم ويتم ذلك عبر منظمات جهد لتعطي على لوحة الآردوينو جهد بقية **3.3v** & **5v** أو أحد القيم السابقة.
- **ملحقات على لوحة الآردوينو**: كالكباسات اللحظية ومداخل قراءة كرت الذاكرة أو حساس أو أي نوع آخر وتكون هذه الملحقات لتسهيل التوصيل مع الآردوينو، وتتوفر هذه الملحقات على لوحات دون غيرها.

استعراض سريع لأشهر لوحات الآردوينو:

عند التعامل مع لوحات الآردوينو سنواجه الكثير من الأنواع منها والسؤال الذي سوف يتبادر للذهن أي لوحة هي الأفضل...؟؟

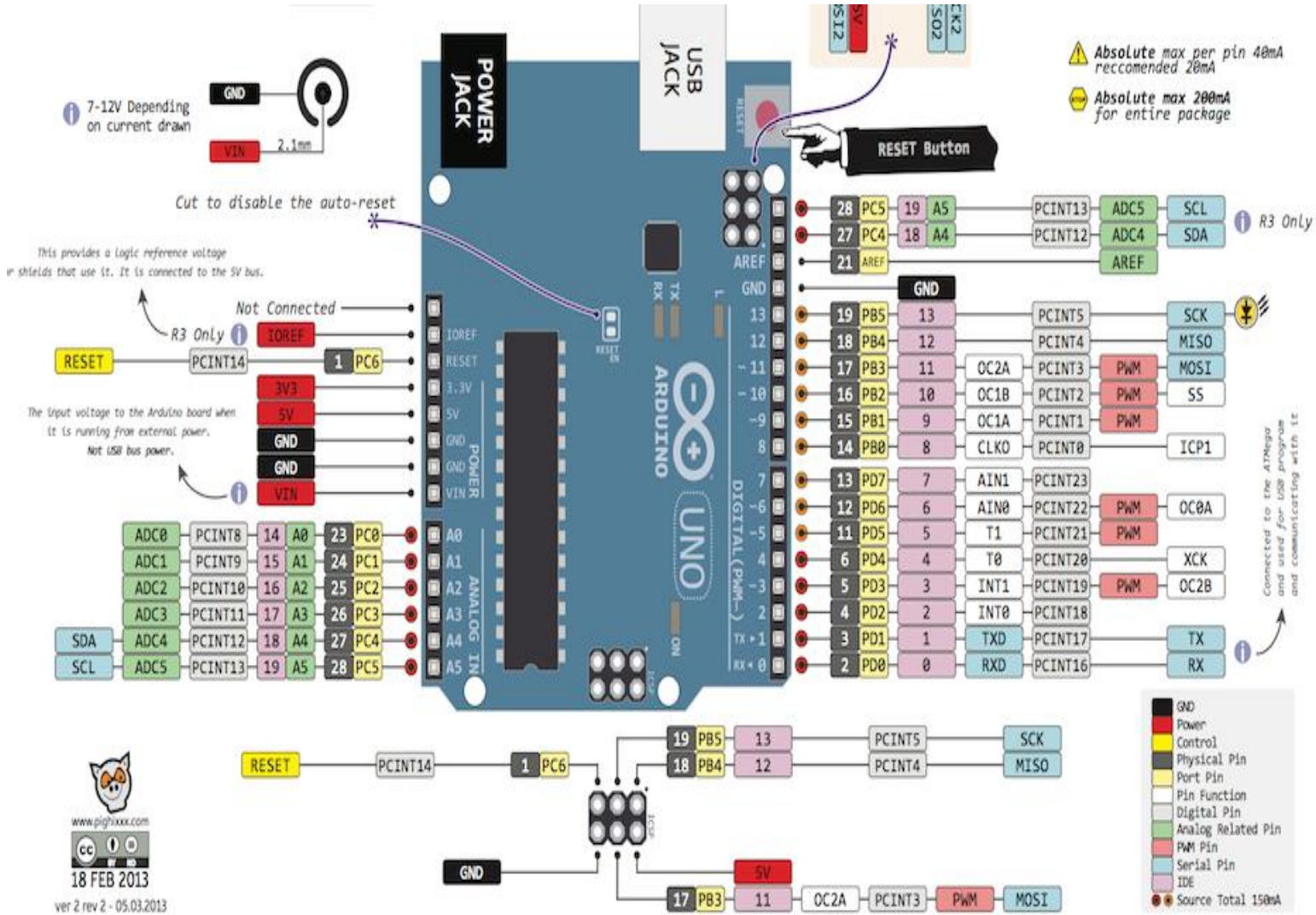
في الحقيقة لا يمكن اختصار الإجابة بنوع واحد معين وتعميمه على أنه الأفضل، فلكل مشروع متطلباته ولكل لوحة ميزات تجعلها هي الأنسب لحالات دون غيرها، ففي حال الحاجة لأعداد كثيرة من الأرجل الرقمية أو التشابهية نختار لوحات مثل لوحة MEGA أما في حالة المشاريع البسيطة الإنتاجية نختار لوحة Pro mini وفيحال المشاريع الإنتاجية مع ضرورة قراءة بارامترات على الحاسب نختار لوحة NANO وهكذا لكن بالمجمل يجب أن يكون لدى المبرمج إطلاع على البارامترات الأساسية للوحات الآردوينووهي:

- جهد التغذية الخاص بلوحة الآردوينو الحدي والأعظمي.
- سرعة المعالج والتي تقاس بـ **MHz**.
- عدد الأقطاب الرقمية **GPIOs** وكذلك الأقطاب الخاصة بالمبدل التشابهي.
- عدد قنوات **PWM**.
- عدد قنوات المبدل التشابهي الرقمي **ADC**.
- حجم ذواكر المعالج **SRAM**,**EEPROM** , **Flash**.
- الميزات الإضافية والتي تشمل الملحقات الإضافية على اللوحة وأبعاد اللوحة وكذلك المميزات الخاصة ببنية المعالج وغير ذلك من الأمور.

لوحة Arduino UNO:

Technical specs

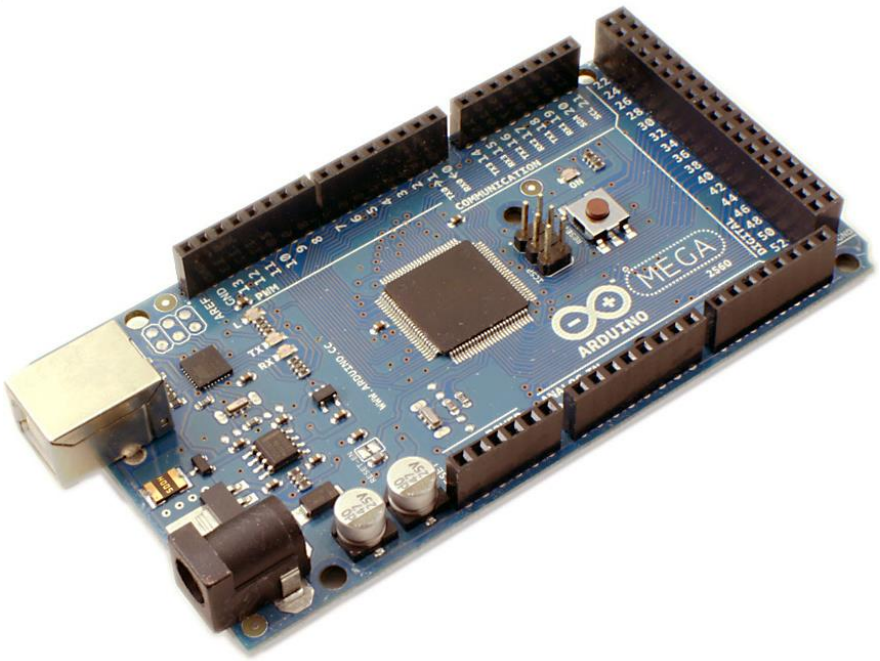
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

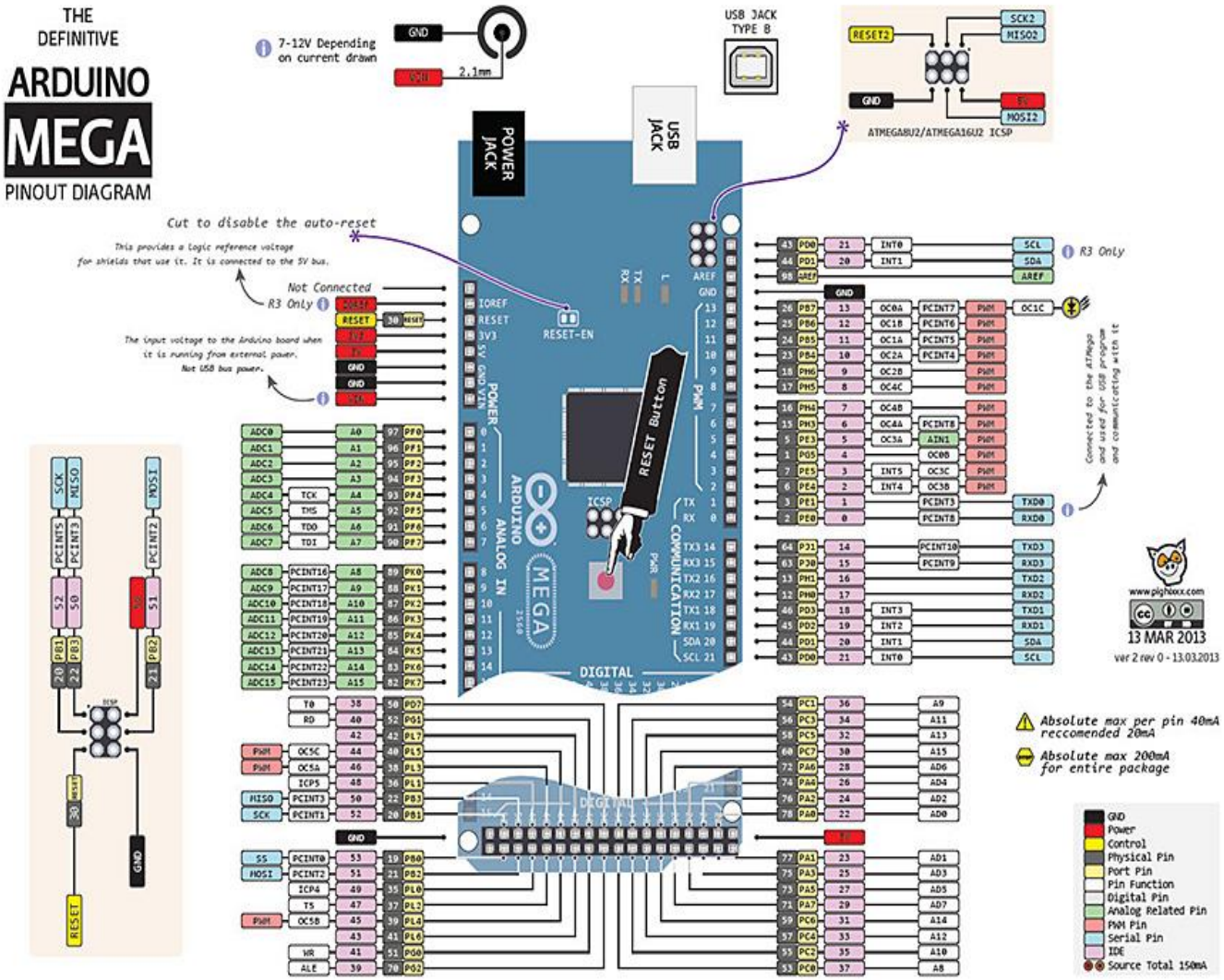

POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
GND
VIN
2.1mm
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
Cut to disable the auto-reset
This provides a logic reference voltage
r shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
IOREF
RESET
PCINT14
1 PC6
3V3
5V
GND
GND
VIN
The input voltage to the Arduino board when
it is running from external power.
Not USB bus power.
ADC0 PCINT8 14 A0 23 PC0
ADC1 PCINT9 15 A1 24 PC1
ADC2 PCINT10 16 A2 25 PC2
ADC3 PCINT11 17 A3 26 PC3
SDA ADC4 PCINT12 18 A4 27 PC4
SCL ADC5 PCINT13 19 A5 28 PC5
28 PC5 19 A5 PCINT13 ADC5 SCL
27 PC4 18 A4 PCINT12 ADC4 SDA
21 AREF AREF
R3 Only
GND
19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
17 PB3 11 OC2A PCINT3 PWM MOSI
16 PB2 10 OC1B PCINT2 PWM SS
15 PB1 9 OC1A PCINT1 PWM
14 PB0 8 CLKO PCINT0 ICP1
13 PD7 7 AIN1 PCINT23
12 PD6 6 AIN0 PCINT22 PWM OC0A
11 PD5 5 T1 PCINT21 PWM
6 PD4 4 T0 PCINT20 XCK
5 PD3 3 INT1 PCINT19 PWM OC2B
4 PD2 2 INT0 PCINT18
3 PD1 1 TXD PCINT17 TX
2 PD0 0 RXD PCINT16 RX
Connected to the ATMega
and used for USB program
and communicating with it
RESET PCINT14 1 PC6 19 PB5 13 PCINT5 SCK
18 PB4 12 PCINT4 MISO
GND
5V
17 PB3 11 OC2A PCINT3 PWM MOSI
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

لوحة Arduino Mega:

Technical specs

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

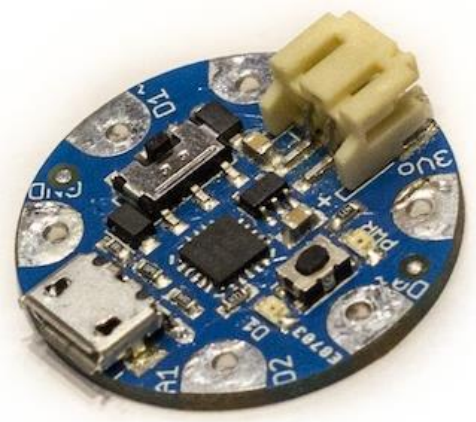



THE
DEFINITIVE
ARDUINO
MEGA
PINOUT DIAGRAM
7-12V Depending on current drawn
GND
2.1mm
USB JACK TYPE B
POWER JACK
USB JACK
ATMEGA8U2/ATMEGA16U2 ICSP
RESET2
SCK2
MISO2
GND
5V
MOSI2
Cut to disable the auto-reset
This provides a logic reference voltage for shields that use it. It is connected to the 5V bus.
Not Connected
R3 Only
IOREF
RESET
3V3
5V
GND
VIN
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
RESET-EN
RESET Button
ICSP
ARDUINO MEGA
ANALOG IN
POWER
DIGITAL
PWM
COMMUNICATION
AREF
GND
SCL
SDA
R3 Only
Connected to the ATMega and used for USB program and communicating with it
TXD0
RXD0
TXD3
RXD3
TXD2
RXD2
TXD1
RXD1
SDA
SCL
www.pighixxx.com
13 MAR 2013
ver 2 rev 0 - 13.03.2013
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

لوحة Arduino Gemma:

Technical specs

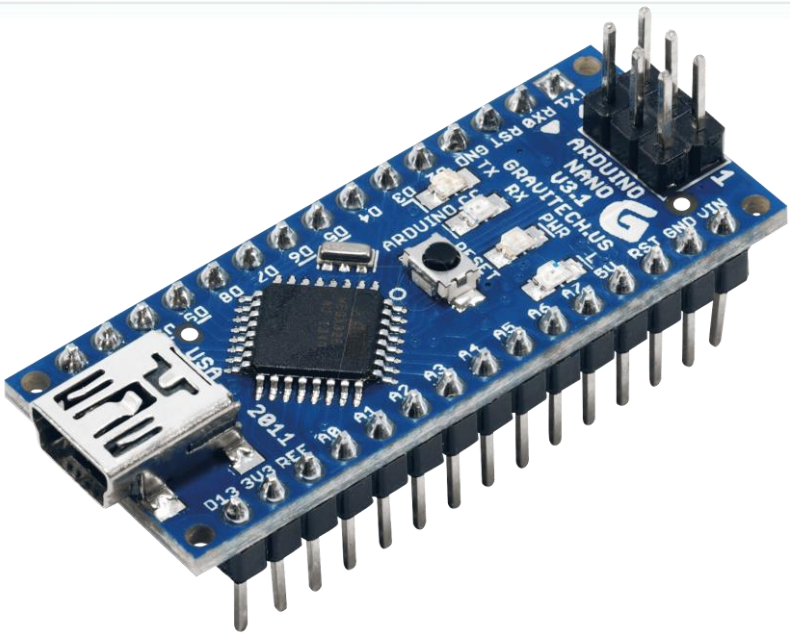
Microcontroller	ATtiny85
Operating Voltage	3.3V
Input Voltage	4V-16V
Digital I/O Pins	3
PWM Channels	2
Analog Input Channels	1
DC Current per I/O Pin	20 mA
Absorption	9 mA while running
Flash Memory	8 kB (ATtiny85) of which 2.75 kB used by bootloader
SRAM	512 Bytes (ATtiny85)
EEPROM	512 Bytes (ATtiny85)
Clock Speed	8 MHz
LED_BUILTIN	1
Diameter	27.94 mm



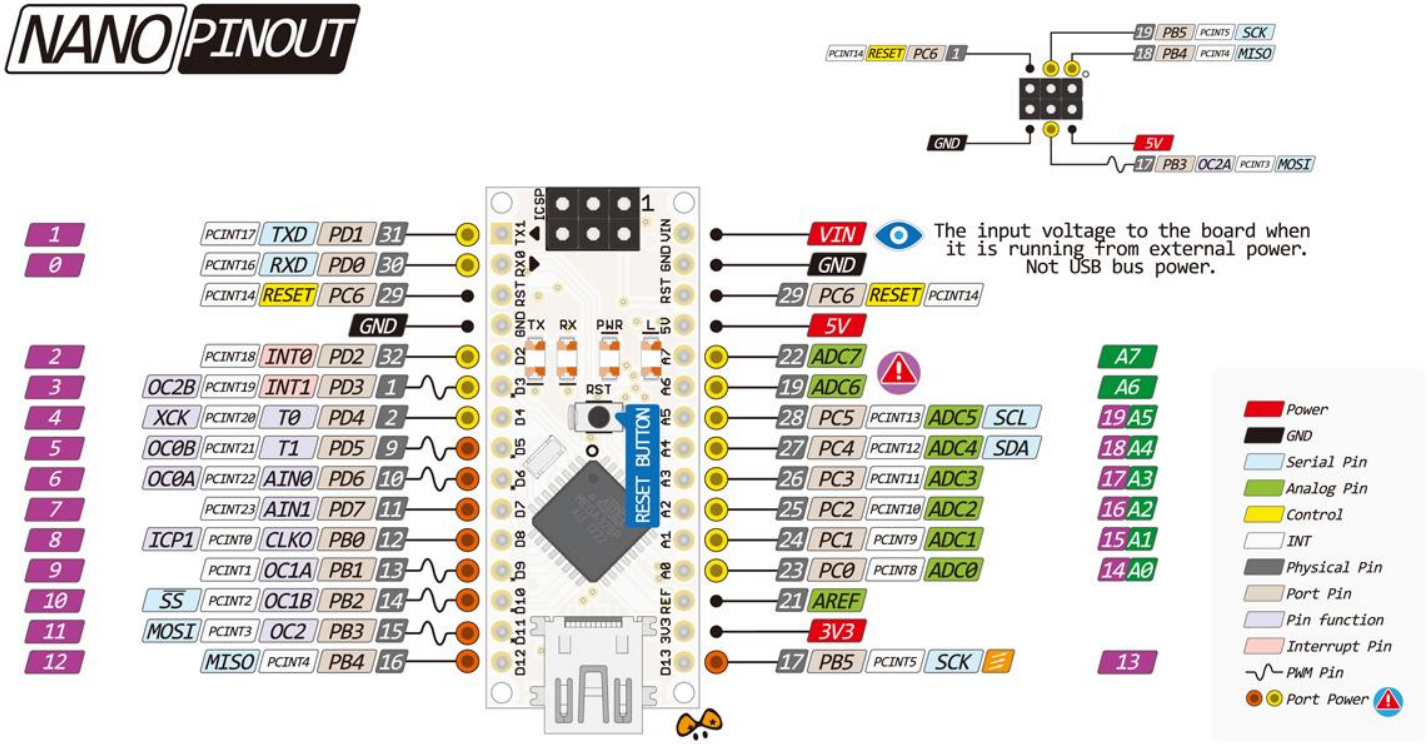
لوحة Arduino Nano:

Technical specs

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog I/O Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12 V
Digital I/O Pins	22
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005



NANO PINOUT



The input voltage to the board when it is running from external power. Not USB bus power.

- Power
- GND
- Serial Pin
- Analog Pin
- Control
- INT
- Physical Pin
- Port Pin
- Pin function
- Interrupt Pin
- PWM Pin
- Port Power

The power sum for each pin's group should not exceed 100mA

Absolute MAX per pin 40mA
recommended 20mA

Absolute MAX 200mA
for entire package

USB JACK
Mini Type B

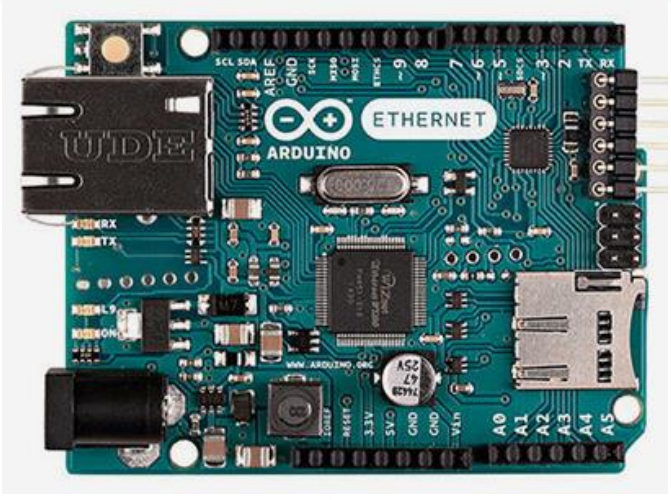
Analog exclusively Pins

19 AUG 2014
ver 3 rev 1

لوحة Arduino Ethernet:

Technical Specs

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage Plug (recommended)	7-12V
Input Voltage Plug (limits)	6-20V
Input Voltage PoE (limits)	36-57V
Digital I/O Pins	14 (of which 4 provide PWM output)
Arduino Pins reserved:	
	10 to 13 used for SPI
	4 used for SD card
	2 W5100 interrupt (when bridged)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
W5100 TCP/IP Embedded Ethernet Controller	
Power Over Ethernet ready Magnetic Jack	
Micro SD card, with active voltage translators	
Length	68.6 mm
Width	53.3 mm
Weight	28 g

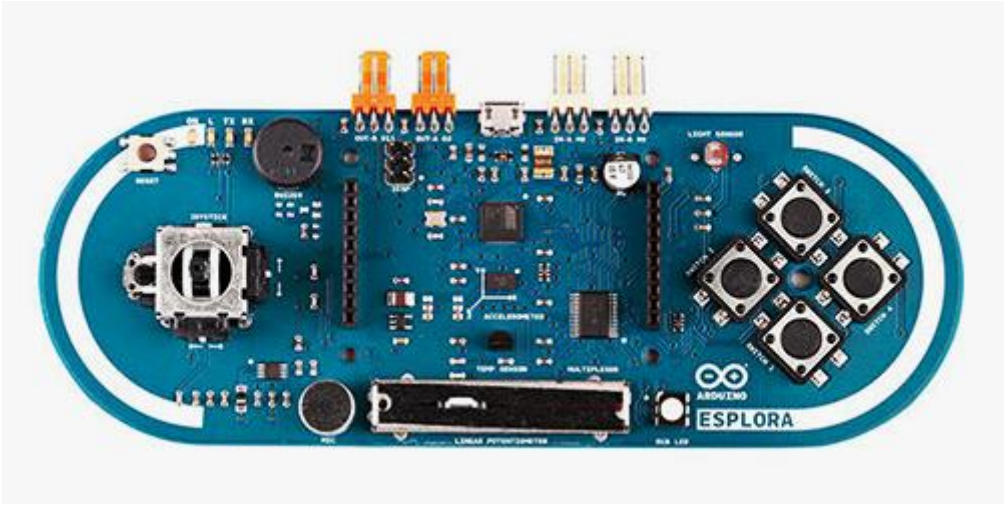


لوحة Arduino Esplora:

Technical specs

Summary

Microcontroller	ATmega32u4
Operating Voltage	5V
Flash Memory	32 KB of which 4 KB used by bootloader
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz
Lenght	164.04 mm
Width	60 mm
Weight	53 g



لوحة Arduino AG101:

Technical specs

Microcontroller	Intel Curie
Operating Voltage	3.3V (5V tolerant I/O)
Input Voltage (recommended)	7-12V
Input Voltage (limit)	7-17V
Digital I/O Pins	14 (of which 4 provide PWM output)
PWM Digital I/O Pins	4
Analog Input Pins	6
DC Current per I/O Pin	20 mA
Flash Memory	196 kB
SRAM	24 kB
Clock Speed	32MHz
LED_BUILTIN	13
Features	Bluetooth LE, 6-axis accelerometer/gyro
Length	68.6 mm
Width	53.4 mm
Weight	34 gr.

لوحة Arduino M0:

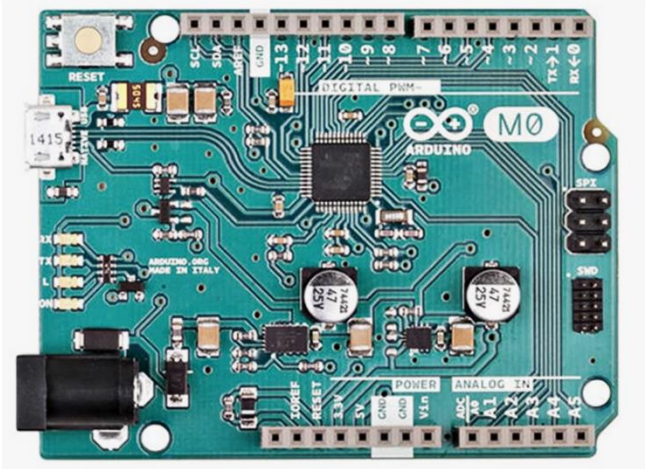
Technical specs

Arduino Microcontroller

Microcontroller	ATSAMD21G18, 48pins LQFP
Architecture	ARM Cortex-M0+
Operating Voltage	3.3V
Flash memory	256 KB
SRAM	32Kb
Clock Speed	48 MHz
Analog I/O Pins	6 +1 DAC
DC Current per I/O Pins	7 mA (I/O Pins)

General

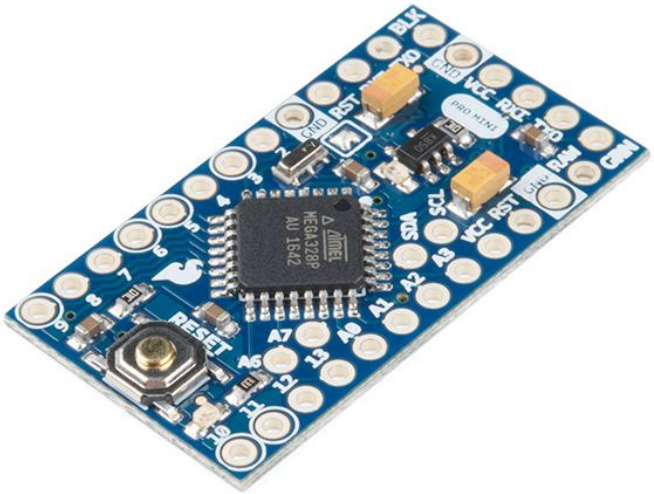
Input Voltage	5-15 V
Digital I/O Pins	20, with 12 PWM and UART
PWM Output	12
Power Consumption	29 mA
PCB Size	53 x 68.5 mm
Weight	21g
Product Code	A000103

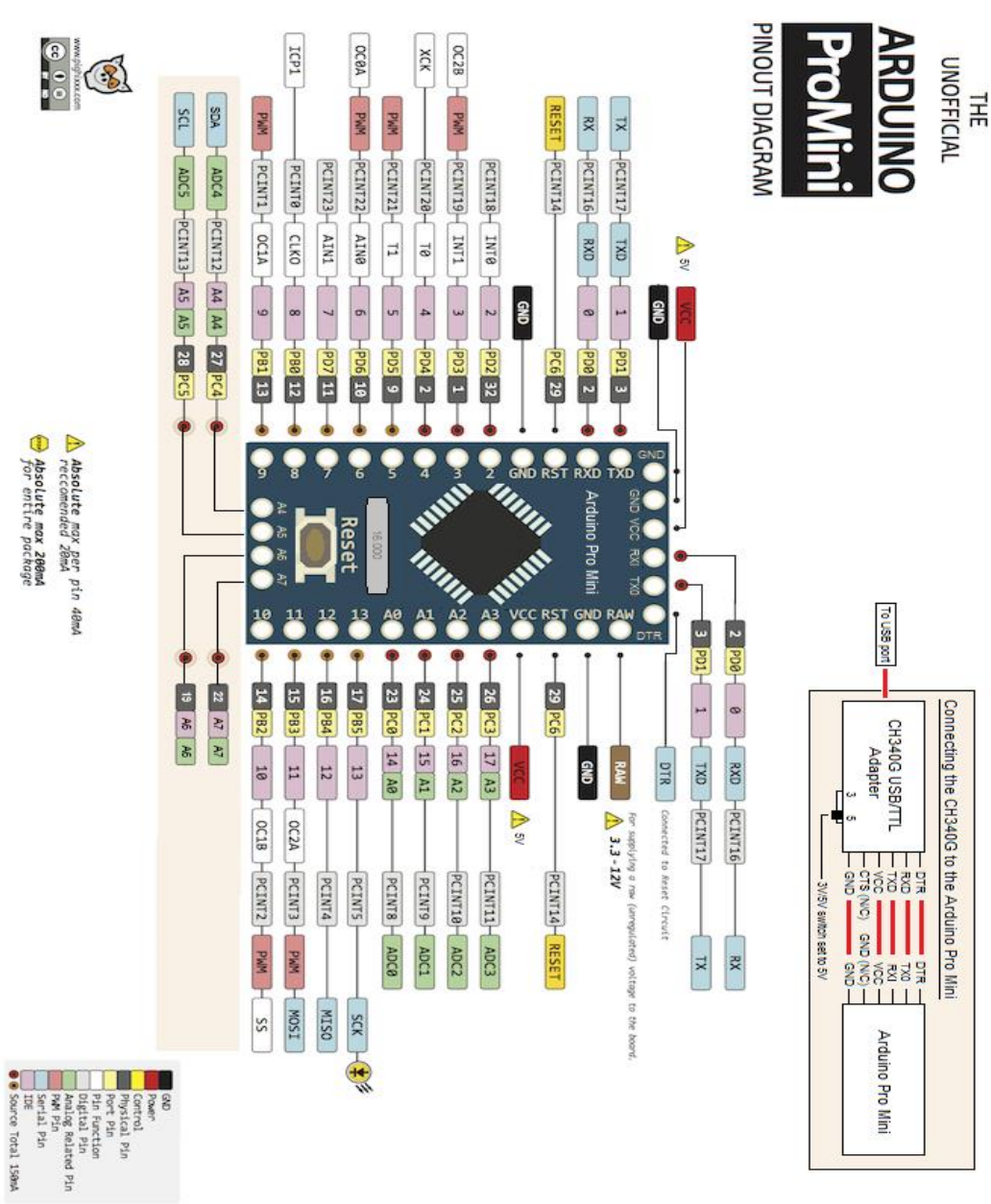


لوحة Arduino Pro mini:

Technical specs

Microcontroller	ATmega328
Board Power Supply	3.35 -12 V (3.3V model) or 5 - 12 V (5V model)
Circuit Operating Voltage	3.3V or 5V (depending on model)
Digital I/O Pins	14
PWM Pins	6
UART	1
SPI	1
I2C	1
Analog Input Pins	6
External Interrupts	2
DC Current per I/O Pin	40mA
Flash Memory	32KB of which 2KB used by bootloader
SRAM	2KB
EEPROM	1KB
Clock Speed	8MHz (3.3V versions) or 16MHz (5V versions)
LED_BUILTIN	13





<u>ملاحظة</u>: تحتوي لوحة الآردوينو Arduino Pro mini على منظم جهد داخلي للحصول على 5v، وبالتالي يمكننا الاستفادة من هذا الجهد لكن يجب **الانتباه** إلا أن **أكبر تيار** يمكن **استجراره** من هذا المنظم هو **200mA**، عدم الانتباه لهذا الأمر يسبب **تلف** لوحة الآردوينو.

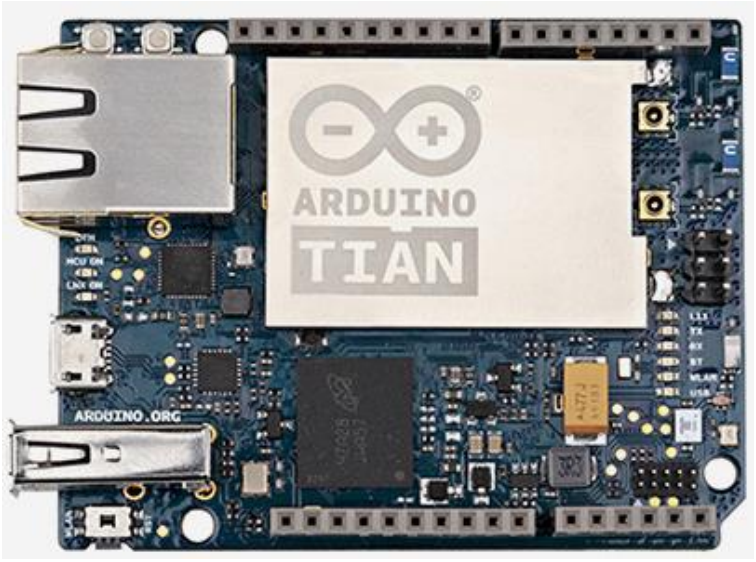
لوحة Arduino Tian:

Arduino Microprocessor

Processor	Atheros AR9342
Architecture	MIPS
Operating Voltage	3.3V
Flash Memory	16MB + 4GB eMMC
Ram	64MB DDR2
Clock Speed	560 MHz
WIFI	802.11 b/g/n 2.4 GHz dual-band
Ethernet	802.3 10/100/1000 Mbit/s
USB	2.0 Host

Arduino Microcontroller

Microcontroller	SAMD21G18
Architecture	ARM Cortex-M0+
Operating Voltage	3.3V
Flash Memory	256 KB
SRAM	32 KB
Clock Speed	48 MHz
Analog I/O Pins	6
DC Current per I/O Pins	7mA (I/O Pins)



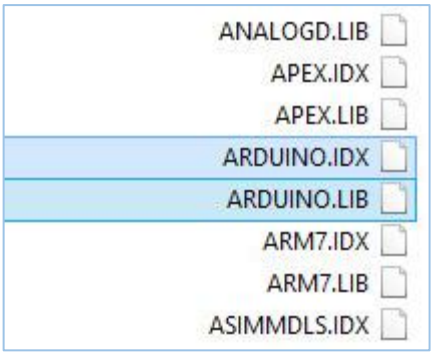
محاكاة الآردوينو على برنامج Proteus 8.5

السؤال الذي سوف يتبادر لنا عند التعامل مع لوحات الآردوينو هو كيف سنقوم بتجريب مشاريعنا وبنائها وتقديمها كمشاريع خدمية متكاملة...؟؟؟ وهل نحن مضطرون لشراء جميع القطع المراد تعلمها وتعلم آلية عملها ...؟؟؟

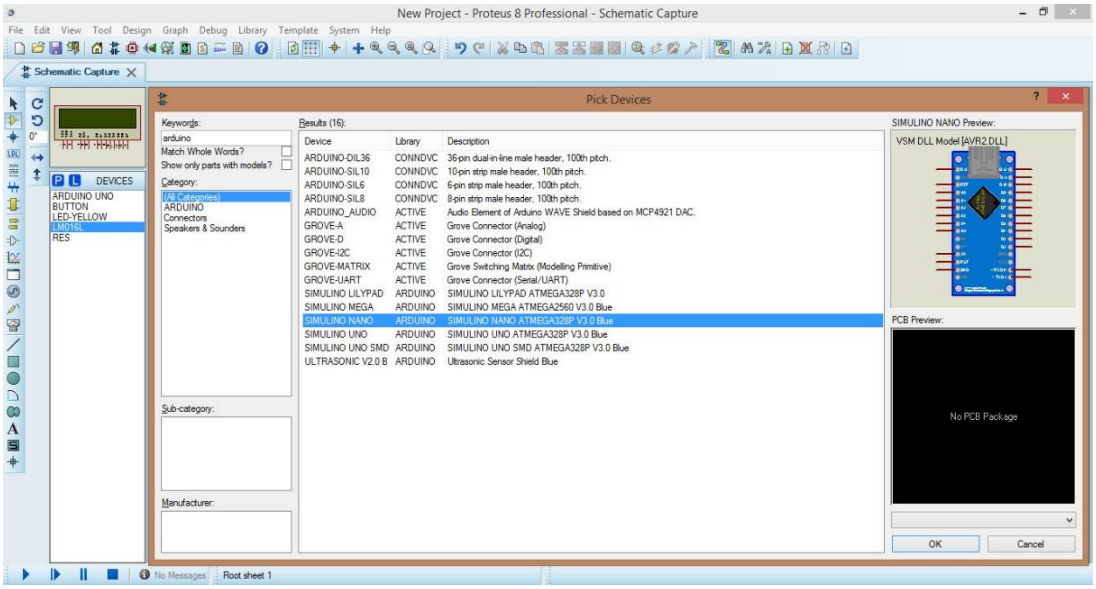
تتوفر العديد من البرامج التي تقوم بمحاكاة لوحات الآردوينو ومحاكاة القطع المختلفة التي يتم توصيلها مع لوحات الآردوينو من شاشات بأنواعها المختلفة إلى المحركات فالحساسات وبروتوكولات الاتصال المختلفة وأبرز هذه البرامج هو برنامج **Proteus 8** والذي قدم بنسخته الأخيرة العديد من الميزات التي تسهل بناء مشاريع عملية متقدمة ومتكاملة، فهو يؤمن من خلال المكتبيات الخاصة بلوحات الآردوينو محاكاة رائعة لعمل هذه اللوحات كما يؤمن قدرة كبيرة على التعامل مع القطع المختلفة من شاشات بأنواعها بدأ من شاشة الكرستالية LCD إلى الشاشة المبنية أساسا من الليد (شاشات القطع السبع 7 segment – الشاشات النقطية Dot Matrix) انتهاء بشاشات اللمس الملونة والمعروفة اختصارا بشاشات TFT، والمحركات بأنواعها المستمرة ومحركات الخطوة ومحركات السيرفو، فضلا عن الكثير من الحساسات التشابهية وآخرها حساس المسافات باستخدام الأمواج فوق الصوتية والذي يأتي مرفقا مع مكتبيات الآردوينو , وليس هذا فقط بل يدعم أيضا تحويل مشاريع المحاكاة إلى مشاريع تطبيقية عبر تحويلها لدارت مطبوعة جاهزة **PCB** عبر قسم رسم الدارات المطبوعة الموجود في البرنامج وأدواته الرائعة ومكتبياته الضخمة للعناصر المختلفة والسهولة الكبيرة التي يقدمها في تصميم الدارات وتجنب الأخطاء التصميمية.

إضافة المكتبيات الخاصة بالآردوينو.

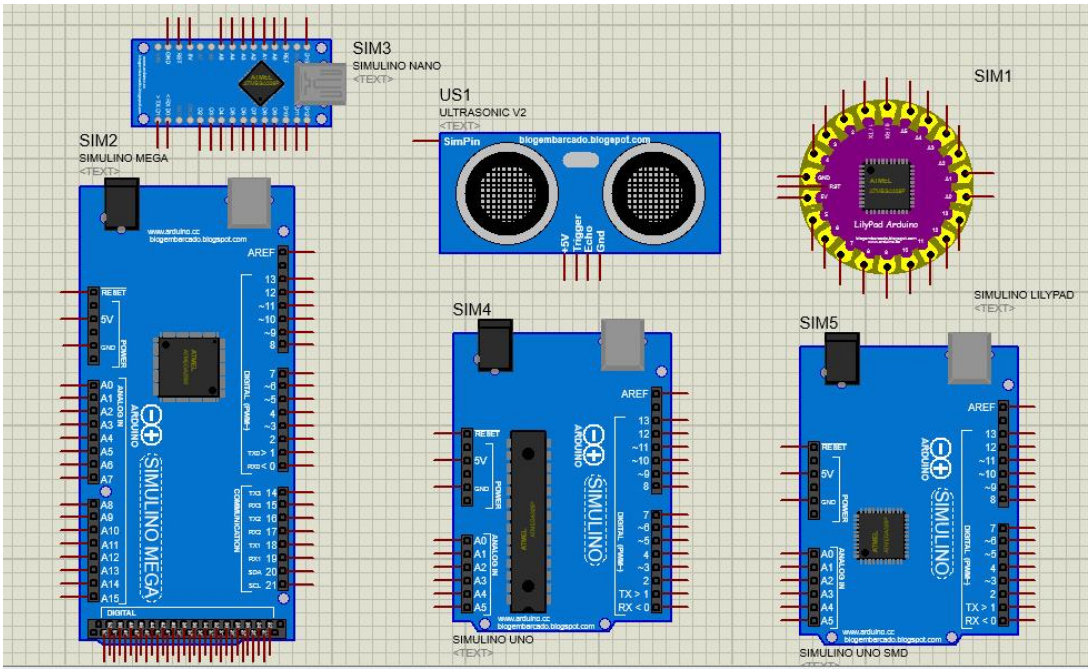
يتوفر العديد من المكتبيات التي تقدم نماذج محاكاة جاهزة للوحات الآردوينو المختلفة وهذه المكتبيات متواجدة على العديد من المواقع، وبعد الحصول على هذه المكتبيات نقوم بنسخ هذه المكتبيات إلى مجلد المكتبيات **Library** الخاص ببرنامج **Proteus 8.5** ثم نعيد تشغيل البرنامج فيكون بذلك قد تمت إضافة هذه المكتبيات للبرنامج وأصبحت جاهزة للعمل.



بالدخول إلى برنامج Proteus 8 نكتب في مكان البحث Arduino فتظهر لدينا لوحات الأردوينو التي تم إضافتها من المكتبية الخاصة.

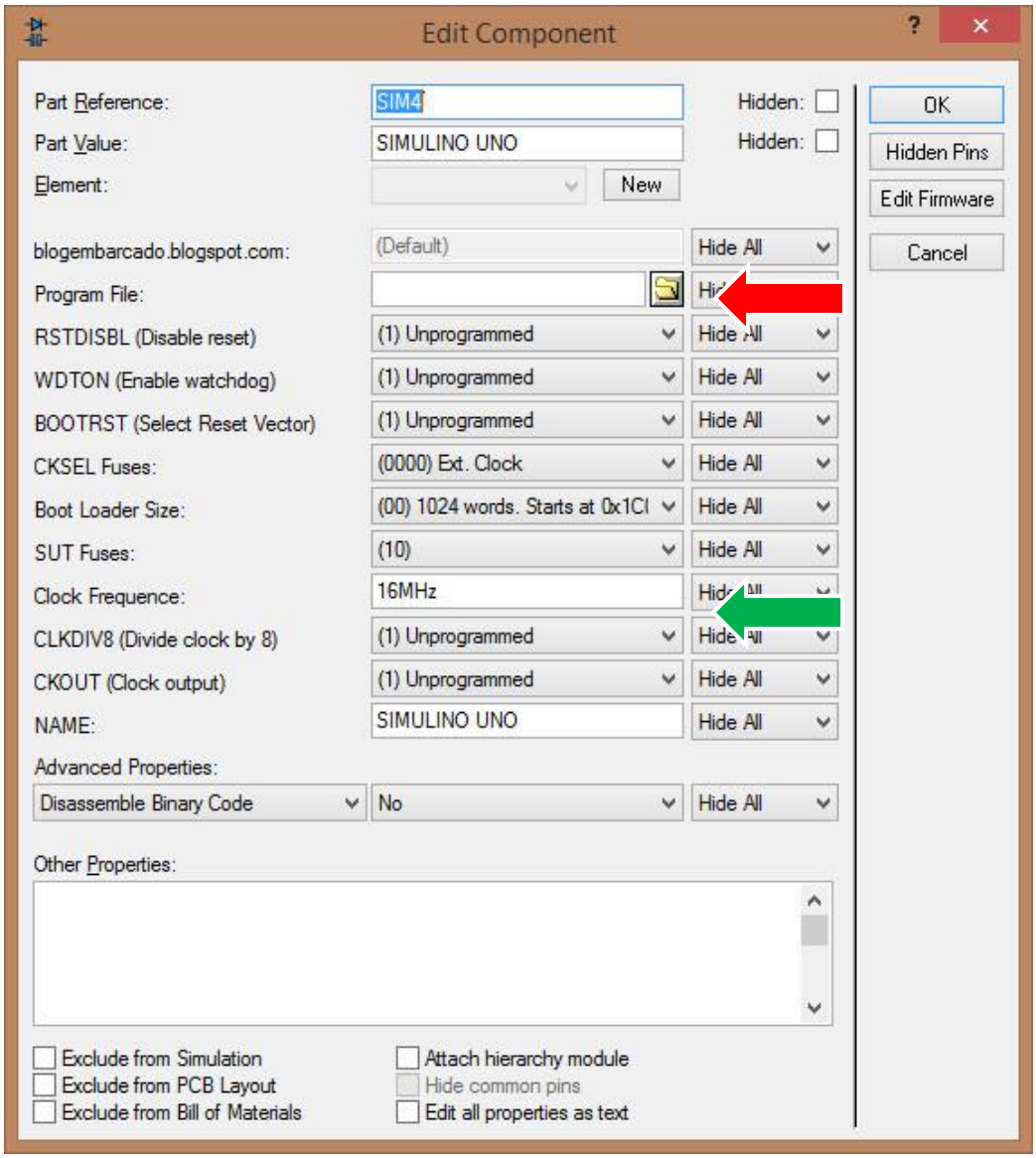


أشهر اللوحات التي يتم إضافتها هي لوحة Arduino UNO ولوحة Arduino Mega ولوحة Arduino Nano بالإضافة لحساس الأمواج فوق الصوتية Ultrasonic لحساب المسافات.



بعد إضافة مكتبيات الآردوينو بقي لدينا خطوة مهمة وهي كيف نحصل على الكود البرمجي الذي تم كتابته في بيئة التطوير **Arduino IDE** وإضافته للوحة الآردوينو على البروتيس Proteus لمحاكاته؟؟؟

يتعامل برنامج البروتيس مع الملفات ذات اللاحقة الثنائية "code.bin" أو اللاحقة الست عشرية "code.hex" وهذه اللواحق تمثل اللغة التي تفهمها الآلة، وبالتالي فإننا نحول الكود الذي نكتبه في بيئة التطوير **Arduino IDE** إلى الصيغة الثنائية (من القائمة Sketch نختار الأمر Export compiled Binary أي تصدير الكود بالصيغة الثنائية والذي سوف يظهر في نفس المجلد الذي تم حفظ ورقة العمل فيه بنفس الاسم لكن بامتداد الثنائي Binary) وبعد تحويله للصيغة الثنائية يتم جلب هذه الصيغة للوحة الآردوينو في بيئة المحاكاة Proteus 8 عبر الضغط على لوحة الآردوينو في برنامج Proteus 8 فتظهر نافذة يتم فيها تحديد مكان تواجد الكود البرمجي ومن ثم تشغيله .



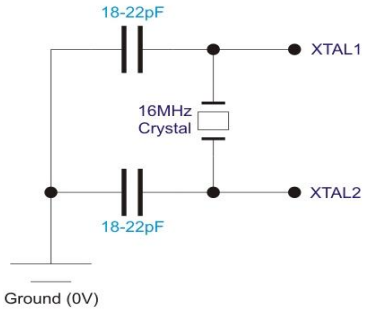
نلاحظ من واجهة خصائص لوحة الآردوينو أن تردد عمل المتحكم 16MHz وهو نفس التردد الذي تعمل عليه لوحة الآردوينو نوع **UNO**، فموضوع **ضبط التردد** في المشاريع التي تتم فيها المحاكاة هو أمر **مهم جدا جدا** لأن اختلاف التردد سيؤدي إلا نتائج غير متوقعة أو غير مضبوطة بالشكل المطلوب وخاصة في المشاريع التي تعتمد على المؤقتات أو على بعض بروتوكولات الاتصال.

الهزاز الكريستالي:

الكريستال أو **الكوارتز** وهو عنصر يتألف من ذرتي أوكسجين مع ذرة سيليكون **SiO2**، أهم ما يميز هذا العنصر أنه ذو مفعول **ارتدادي** فإذا ما تعرض لصدمة ميكانيكية تولد على سطحه جهد كهربائي والعكس صحيح فإذا ما تعرض الكوارتز لجهد كهربائي ينتج عنه اهتزاز للكوارتز ينتج عنه رنين أو تذبذب على قطبي الكرستالة، هذا الاهتزاز يمكن التحكم به وذلك حسب أبعاد شريحة الكوارتز والجهد المطبق عليها.

توصيل الكرستالة مع المتحكم:

إن توصيل الكرستالة مع المتحكم يتضمن عناصر إضافية وهي مكثفتين من قيمة **18 ~ 22 pF** ويتم التوصيل بالشكل التالي:

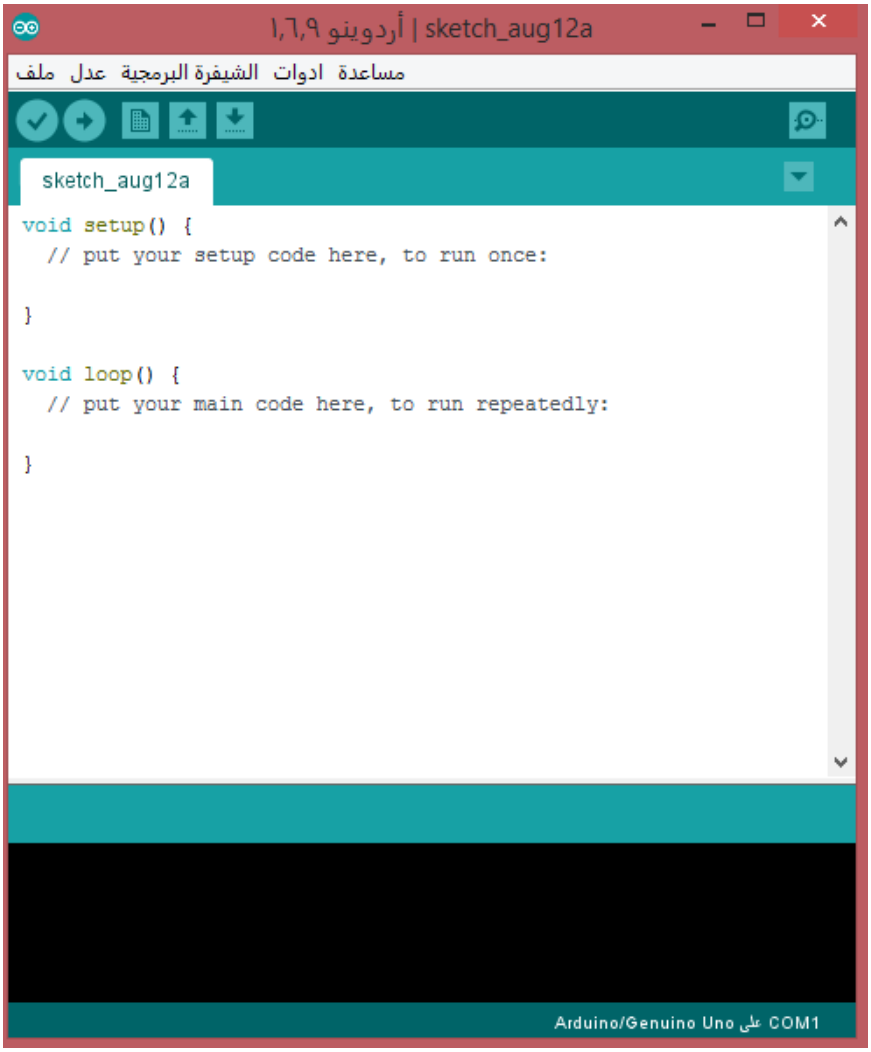


تتميز الهزازات الكرستالية عن نظيراتها من هزازات **RC** أو **LC** بالدقة العالية واستقراريه تتراوح بين **0.01 ~ 0.001 %** ، إضافة للجودة العالية لها.

بيئة التطوير المتكاملة Arduino IDE

واجهة البرنامج الرئيسية:

بيئة التطوير **IDE** وهي اختصار **I**ntegrated **D**evelopment **E**nvironment أي بيئة التطوير المتكاملة، وهي عبارة عن بيئة برمجية تقدم للمبرمج الكثير من الأدوات البرمجية المجمعة كلها في مكان واحد ويتم من خلالها كتابة وتحرير الكود البرمجي وتفحصه ومن ثم تحويله إلى لغة يفهمها الآردوينو وتمريرها له.



تعتمد بيئة **Arduino IDE** في أساسها على لغة البرمجة **C/C++** وتحتوي في بنيتها على العديد من المكتبيات الجاهزة والتعليمات التي تختصر الكثير من الوقت والتي تعتبر من أسهل لغات البرمجة التي تستخدم في برمجة المتحكمات المصغرة **Microcontroller**.

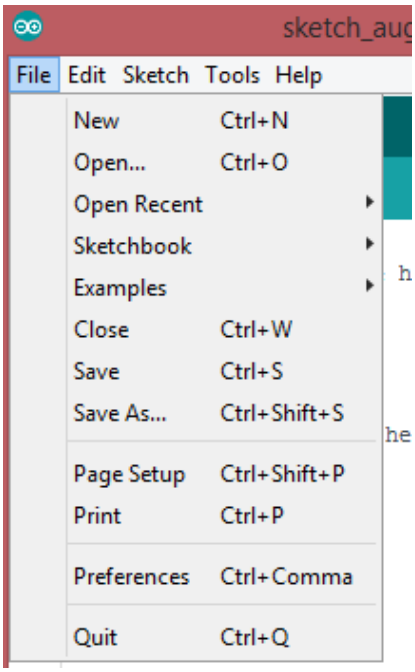
تسمى البيئة التطويرية **Arduino IDE** التي يتم فيها برمجة لوحات الآردوينو وكتابة الكود البرمجي بـ **Sketch** (التصميم أو المخطط)، وكأي بيئة برمجة تحتوي واجهة البرنامج على قوائم وأوامر واختصارات على الواجهة الرئيسية مهمتها تسهيل كتابة الكود البرمجي وتقديم الميزات الكبيرة الموجودة في بيئة التطوير **Arduino IDE**، سنستعرض هذه القوائم بشكل متسلسل مع شرح مبسط عن محتوى كل قائمة:

الاختصارات على الواجهة الرئيسية:

- **فحص Verify**: تفحص الكود من الأخطاء البرمجية.
- **رفع Upload**: تحويل الكود للغة يفهمها المتحكم ومن ثم رفعها للوحة الآردوينو.
- **جديد New**: إنشاء صفحة عمل جديدة.
- **فتح Open**: فتح ملفات موجودة مسبقا أو مشاريع جاهزة للتعديل عليها.
- **حفظ Save**: حفظ صفحة العمل.
- **مراقب المنفذ التسلسلي Serial monitor**: لمراقبة الأحداث عبر المنفذ التسلسلي.

قائمة ملف File:

جديد New: فتح صفحة عمل جديدة.



فتح Open: فتح صفحة عمل موجودة سابقا.

المفتوحة حديثا Open recent: عرض صفحات العمل الأكثر استخداما.

كتاب شيفرة البرمجة Sketchbook: يدل على الرسومات المضمنة في مجلد صفحة العمل.

أمثلة Example: يقدم هذا الخيار العديد من الأمثلة الجاهزة في بيئة **Arduino IDE** وكذلك المكتبيات الجاهزة.

إغلاق Close: إغلاق نافذة العمل الحالية.

حفظ Save: حفظ صفحة العمل الحالية بالاسم الحالي.

حفظ باسم Save as: حفظ صفحة العمل باسم جديد.

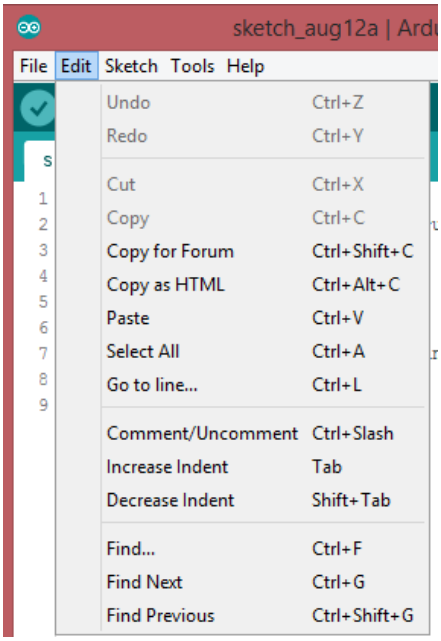
إعدادات الصفحة Page Setup: تغيير إعدادات الصفحة للطباعة.

طباعة Print: طباعة صفحة العمل.

تفضيلات Preference: يتم فتح نافذة يتم من خلالها تحرير إعدادات ورقة العمل منها إعدادات اللغة وترقيم أسطر البرنامج وغير ذلك.

خروج Quite: إغلاق جميع نوافذ البرنامج.

قائمة تحرير Edit:



تراجع /إعادة Undo/Redo : يتم من خلال هذين الأمرين التراجع للخلف خطوة أو التقدم للأمام خطوة أثناء تحرير الكود البرمجي.

قص Cut: إزالة قسم محدد من المحرر ووضعه في الحافظة.

نسخ Copy: نسخ قسم محدد من المحرر ووضعه في الحافظة.

نسخ للمنتدى Copy for Forum: نسخ الكود البرمجي لورقة العمل الخاصة بالمبرمج لموقع المنتدى.

نسخ كـ Copy as HTML: نسخ ورقة العمل بامتداد HTML المتناسب مع صفحات مواقع النت.

لصق Paste: وضع محتوى الحافظة في موضع المؤشر في محرر الكود.

تحديد الكل Select All: يختار ويحدد جميع محتوى محرر الكود.

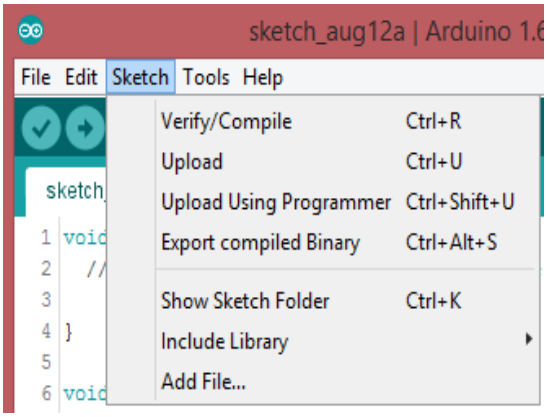
الذهاب للسطر Go to line: الذهاب للسطر المطلوب والذي له رقم.

ملاحظة/إلغاء الملاحظة Comment/Uncomment: وضع أو إزالة علامة التعليق " // " في بداية كل سطر محدد.

زيادة /إنقاص المسافة البادئة Increase/Decrease Indent: يتم عبر مفتاح Tab.

إيجاد Find: فتح نافذة يتم من خلالها البحث عن جملة مطلوبة واستبدالها.

قائمة الشيفرة البرمجية Sketch:



تحقق/ترجمة Verify/Compile:
لفحص الكود البرمجي وترجمته للغة يفهمها المتحكم.

رفع Upload: رفع الكود للوحة الآردوينو بعد تحويله للغة يفهمها المتحكم.

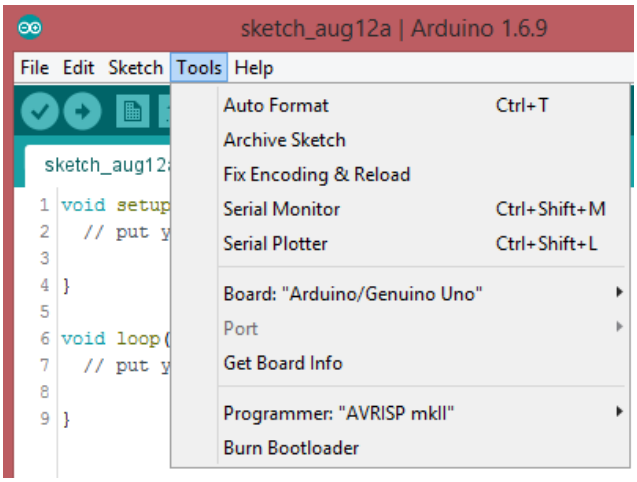
رفع باستخدام المبرمجة Upload Using Programmer: نسخ الكود البرمجي عبر مبرمجة للمتحكم، لكن استخدام هذه الآلية سوف تأدي للكتابة فوق محمل الإقلاع Bootloader.

تصدير الترجمة باللغة الثنائية Export compiled Binary: يتم تصدير الكود بصيغة الست عشري " **hex**. " وبالصيغة الثنائية "**bin**. " لاستخدامها في تطبيقات أخرى كالمحاكاة في برنامج **Proteus**.

عرض مجلد ورقة العمل Show Sketch Folder: يتم فتح المجلد الذي تم حفظ ورقة العمل فيه.

إضافة مكتبة Include Library: إضافة مكتبية لورقة العمل وتتم عملية الإضافة بعد إشارة التضمي" # "، كما يتم من خلال هذا البند الوصول لإدارة المكتبات واستيرادها.

قائمة الأدوات Tools:



تنسيق تلقائي Auto Format: يتم من خلال هذا الأمر تنسيق النص البرمجي بشكل تلقائي فيتم ضبط النص وإغلاق الأقواس المفتوحة.

أرشفة الكود البرمجي Archive Sketch: يتم أرشفة ورقة العمل بنسخة ذات امتداد zip وحفظها في نفس مكان حفظ الكود البرمجي.

تشغيل مراقب المنفذ التسلسلي Serial Monitor: لإرسال واستقبال البيانات من لوحة الآردوينو عبر الحاسب.

Serial Plotter: فتح نافذة التخاطب مع المنفذ التسلسلي.

اللوحة Board: تحديد لوحة الآردوينو التي يتم التعامل معها.

المنفذ Port: تحديد المنافذ التسلسلية الحقيقة والافتراضية الموجودة على الجهاز.

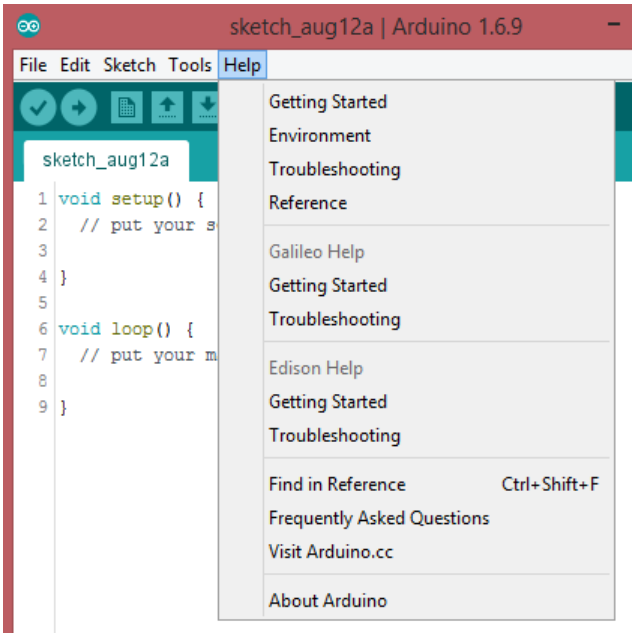
المبرمجة Programmer: يتم من خلال هذا الأمر اختيار الطريقة التي سوف يتم من خلالها برمجة المتحكم الموجود على اللوحة وفي حال عدم وجود محمل الإقلاع **Bootloader** سوف تحتاج لتحديد نوع المبرمجة.

تحميل محمل الإقلاع Burn Bootloader: نستخدم هذا الأمر من هذه القائمة عند وضع متحكم جديد على لوحة الآردوينو ونريد تزويد هذا المتحكم بمحمل الإقلاع حيث أن المتحكم يأتي بشكل افتراضي خالي من أي برنامج، لكن علينا الانتباه لـ Fuse Bit وضبطها بالشكل الصحيح.

قائمة الأدوات Tools:

في هذه القائمة نجد الأوامر المتعلقة بتقديم المساعدة للمبرمج وخاصة الذي يستخدم بيئة برمجة الآردوينو للمرة الأولى فنجد الأمر **Getting Started** لتقديم المساعدة عند الشروع في تعلم برمجة لوحات الآردوينو، أما الأمر **Environment** فيستعرض بيئة البرمجة والأوامر الموجودة في القوائم وما إلى ذلك، بينما يقدم الأمر **Troubleshooting** حلول للمشاكل التي تواجه المبرمج عبر طرح أجوبة للأسئلة الأكثر شيوعا، أما الأمر **Reference** فيقدم مرجعا لجميع الأوامر والتعليمات والتوابع الموجودة في بيئة التطوير **Arduino IDE**.

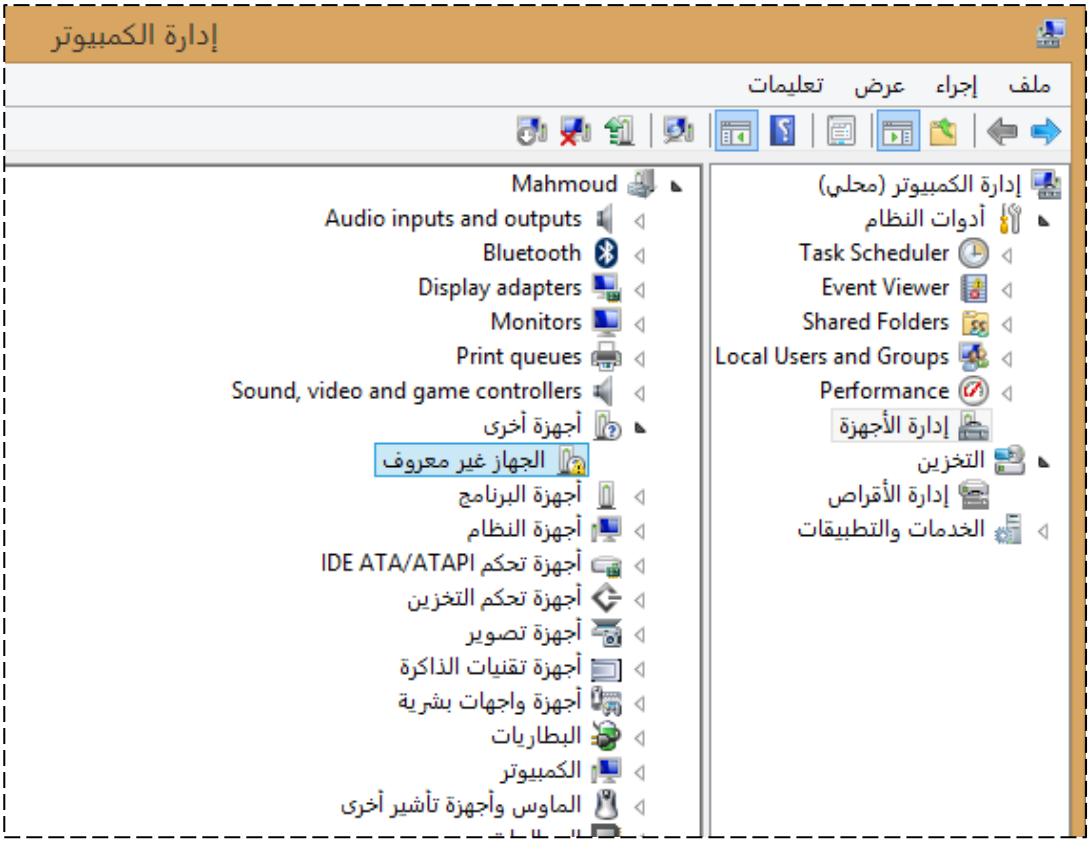
وفي حال الحاجة لمعرفة آلية عمل أي تعليمة أو تابع ما علينا سوى وضع المؤشر عند التعليمة المراد معرفة آلية عملها ومن ثم ضغط الأمر **Find in Reference** لإيجاد وشرح التعليمة المطلوبة.



تعريف لوحة الآردوينو على الحاسب

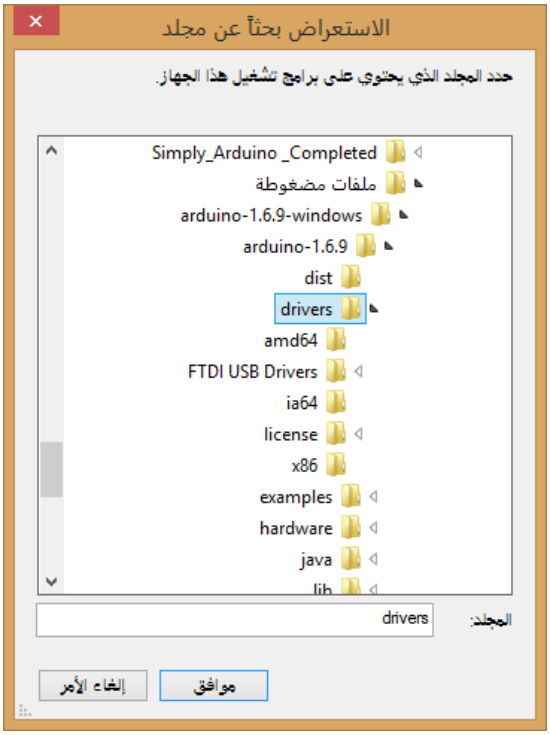
ماذا علينا أن نفعل عند تشغيل لوحة الآردوينو للمرة الأولى .... ؟؟؟؟؟؟

عند توصيل لوحة الآردوينو للمرة الأولى للحاسب علينا تعريف اللوحة وذلك ليتمكن برنامج **Arduino IDE** من نقل البرامج للوحة والتعامل معها، فعند وصل لوحة الآردوينو للمرة الأولى للحاسب سوف التكون غير معرفة ويظهر ذلك في قسم إدارة الأجهزة كما في الصورة:



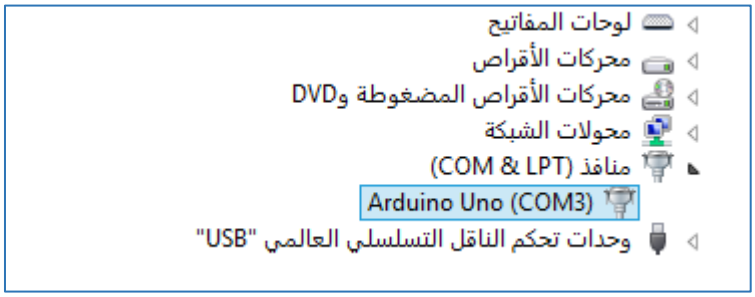
نلاحظ من الصورة أن لوحة الآردوينو غير معرفة في الحاسب ولا يمكن التعامل معها في هذه الحالة لذلك يجب علينا أن نعرفها.

نقوم الآن بتعريف اللوحة وذلك باختيار برنامج التشغيل المطلوب وتحديده من مجلد Driver الموجود مع برنامج **Arduino IDE** كما في الصورة:

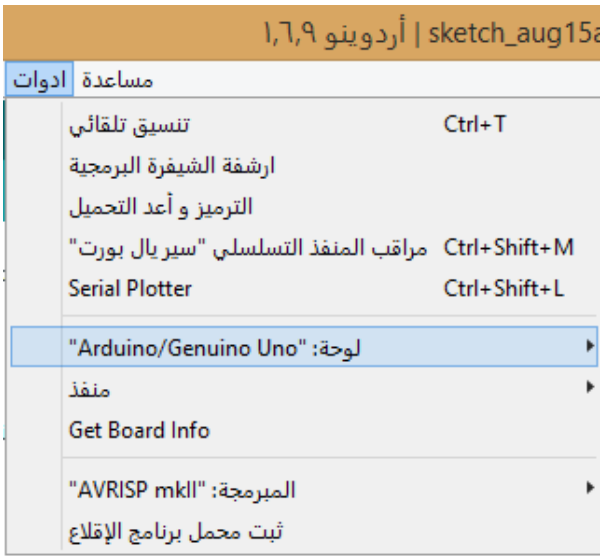


بعد اختيار المجلد driver سوف تظهر لنا النافذة التالية:

نختار " تثبيت " فيتم بذلك تثبيت لوحة الآردوينو التي تم وصلها مع الحاسب وهذا ما يظهر في قسم إدارة الأجهزة كما أنه يتم إعطاء رقم منفذ USB الذي تم توصيل اللوحة إليه على أنه رقم منفذ ال COM وهذا مهم لاحقا:



الآن بقي تحديد لوحة الآردوينو في برنامج **Arduino IDE** ليتم التعامل معها ويتم ذلك من قائمة الأدوات Tools ثم نختار اللوحة التي تم وصلها مع الحاسب (وهنا تم وصل لوحة آردوينو Arduino UNO) كما يتم من هذه القائمة أيضا تحديد منفذ COM الذي تم توصيل لوحة الآردوينو إليه (في حالتنا هذه رقم المنفذ COM3) وتحديد المنفذ مهم لعملية نقل الكود للوحة وعملية التخاطب مع اللوحة أثناء عمل الكود, وبذلك تتم عملية التهيئة وتجهيز لوحة الآردوينو للبرمجة والعمل عليها.

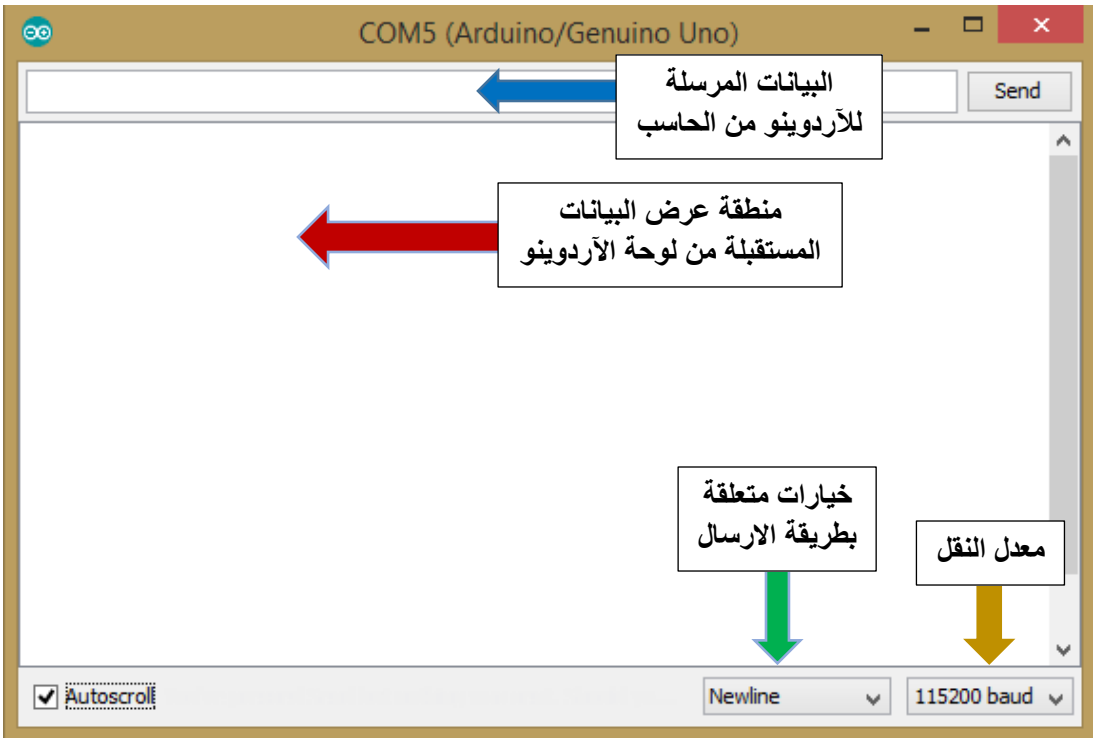


واجهة الاتصال التسلسلي UART

سنتطرق في هذا القسم لأمر مهم وهو واجهة الاتصال التسلسلية الموجودة في بيئة التطوير **Arduino IDE**، هذه الواجهة تؤمن آلية للتواصل مع المتحكم أثناء عمله وبالتالي يمكن أن نزود المتحكم أثناء عمله بأوامر لتوجيهه أو الحصول على معلومات وحسابات معينة مطلوبة من المتحكم أثناء عمله وبالتالي يتوفر لنا آلية جيدة للتأكد من أن الكود يعمل بالشكل الصحيح.

نقوم بتوصيل لوحة الآردوينو للحاسب ثم نشغل بيئة التطوير **Arduino IDE**، بعد ذلك نضبط في القائمة Tools نوع اللوحة التي تم توصيلها ورقم منفذ COM المعرف عليه اللوحة.

نشغل واجهة الاتصال التسلسلي عبر القائمة Tools نختار الأمر Serial Monitor أو من مفتاح الاختصار الخاص بواجهة الاتصال التسلسلي الذي سبق وتحدثنا عنه، فتظهر لنا الواجهة التالية:



أهم أمر يجب ضـــبطه عند تشغيل واجهة الاتصـــال التســـلسلي هو مـــعدل النقل والمعروف بـ **Baud Rate** ويقاس بواحدة **bps** أي معدل بايتات المرسلة في الثانية، تستخدم هذه الواجهة نافذة الاتصال التسلسلي الغير متزامن **UART** والتي لها قطـــبين على لوحـــــة الآردوينو همـــا **TX** - **RX** ، لن نخوض كثيرا في تفاصيل بروتوكول الاتصال هذا بل سنتطرق له في فصل لاحق لكن حاليا ما يهمنا هنا هو معرفة التعليمات الأساسية التي سوف نستخدمها لتفعيل دور هذه الواجهة في مشاريعنا , أهم التعليمات المستخدمة هي:

شرح التعليمة	التعليمة
تحديد معدل نقل البيانات بواحدة bps، في قسم الإعداد بعد الدالة void setup() { .	`Serial.begin(115200);`
طباعة (ارسال) بيانات Val من الآردوينو إلى الحاسب بالصيغة التي نختارها format، ويتم طباعة كل قيمة جديدة في سطر جديد.	`Serial.println(Val,format);`
طباعة (ارسال) بيانات Val من الآردوينو إلى الحاسب بالصيغة التي نختارها format، ويتم طباعة كل القيم في نفس السطر.	`Serial.print(Val,format);`
وهي تعليمة تستخدم للتأكد من وصول البيانات على المنفذ التسلسلي، ففي حال ورود أي بيانات على المنفذ التسلسلي تصبح نتيجة هذه التعليمة True.	`Serial.available();`

`Val = Serial.read( );`	**قراءة القيمة المرسلة إلى لوحة الآردوينو واسنادها للمتحول Val، والقيم التي يتم قراءتها من نوع محرفي حصرا.**
`Val = Serial.Stringread( );`	**قراءة سلاسل محرفية مدخلة على المنفذ التسلسلي، المتحول val من نوع String**

يجب الانتباه أن الصيغ المتاحة format هي (**BIN** , **HEX** , **DEC** , **OCT** , **ASCII**) وفي الحالة الافتراضية يتم طباعة الأرقام بالقيم الصحيحة والعبارات كمحارف، يمكن أن نضع بدل format رقم نختاره وفي هذه الحالة فإن الرقم يعبر عن عدد الخانات التي نريد طباعتها بعد الفاصلة العشرية .

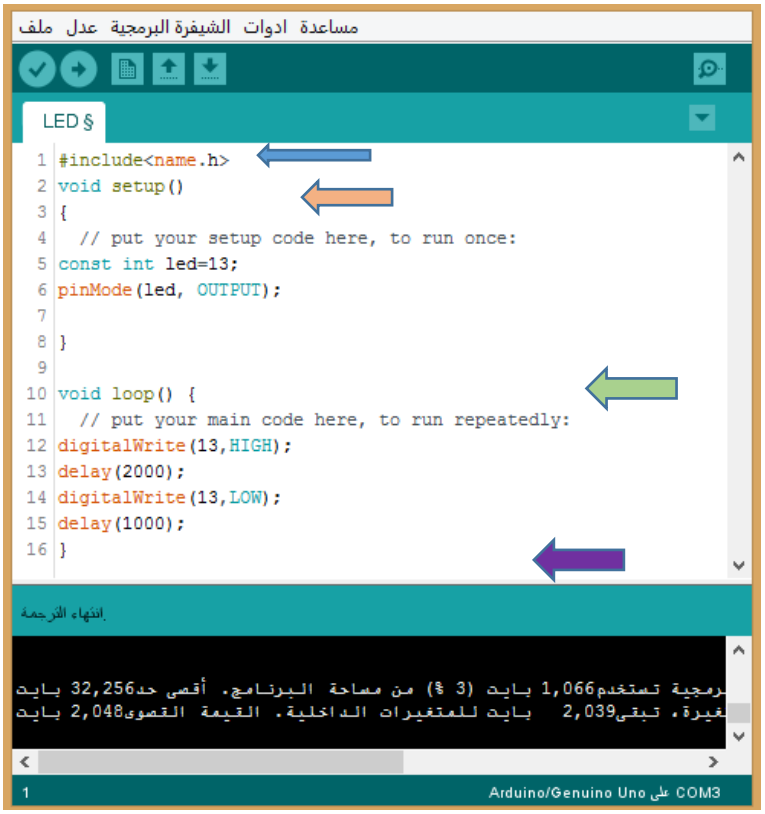
هذه جملة من التعليمات يتم استخدامها مع واجهة الاتصال التسلسلي وهي فقط استعراض سريع لكي نتعلم التعامل مع هذه الواجهة مبدئيا وسنتطرق لبروتوكول الاتصال **UART** بشكل كافي في قسم لاحق من الكتاب.

لغة البرمجة C/C++ في بيئة Arduino IDE

كما ذكرنا سابقا فإن بيئة برمجة لوحات الآردوينو تعتمد على لغة البرمجة C++، لذلك ولكي نكون متقنين لبرمجة لوحات الآردوينو يجب أن يكون لدينا أسس جيدة في هذه اللغة، وبناء عليه فإننا سنولي أهمية تعريفية بهذه اللغة مع الأخذ بعين الاعتبار أن هذه اللغة هنا موجهة لبرمجة الأنظمة المدمجة وليس لكتابة برامج بلغة الـ C++ في الحواسيب، ففهم هذه اللغة وأساسياتها يوفر علينا الكثير من الوقت في فهم المكتبيات الجاهزة والتي كتبت أساسا في بيئة C++.

تقسيم ورقة العمل:

من أجل كتابة كود برمجي متوازن وقابل للتعديل والتطوير يجب علينا تقسيم ورقة العمل أربعة أقسام أساسية هي:



- **القسم الأول:** يتم فيه تضمين المكتبيات المطلوبة للمشروع وتعريف الأسماء المستعارة للأقطاب وتعريف المتحولات المطلوبة والتي تعرف باسم **Global Varible** ويبدأ هذا القسم من السطر الأول لورقة العمل.
- **القسم الثاني:** يكون ضمن الدالة void setup ( ) {.....} وفي هذا القسم يتم تهيئة المداخل والمخارج، وتهيئة المكتبيات التي سيتم توصيلها للوحة الآردوينو.
- **القسم الثالث:** وهو ضمن الدالة void loop ( ) {.....} وهي حلقة البرنامج اللانهائية والتي يتم فيها تكرار العمليات والتعليمات المطلوبة بعدد لانهائي من المرات.
- **القسم الرابع:** وهو القسم الذي يضم التوابع الفرعية التي يتم فيها تنفيذ جزء محدد من الكود البرمجي وذلك كلما تم استدعاءه من الحلقة الرئيسية.

ملاحظات

- ✓ لإضافة تعليقات **Comments** على الكود البرمجي يجب وضع الإشارة " // " ومن ثم كتابة التعليق، أما إذا كان التعليق المطلوب وضعه طويل فيتم وضع الإشارة " /* " في بداية التعليق ومن ثم إنهاء التعليق بالإشارة " */ " .
- ✓ المنطقة السوداء الموجودة أسفل منطقة العمل هي منطقة تصحيح الأخطاء، فعند ضغط مفتاح التأكد من الكود فإنه سوف يتولد في هذه المنطقة رسائل توضح مكان الأخطاء الموجودة في الكود البرمجي.
- ✓ تعتبر بيئة التطوير **Arduino IDE** مشتقة من لغة البرمجة **C/C++** لذلك فهي لغة حساسة لحالة الأحرف (كبيرة أو صغيرة) كما أنها تعطي تنبأ تلقائي للأقواس وتغيير لون الكلمات المحجوزة.
- ✓ كل تعليمة في لغة **Arduino IDE** يجب أن تنتهي بـ " ; " وإلا فسوف يتولد لدينا رسالة خطأ عند رفع الكود.

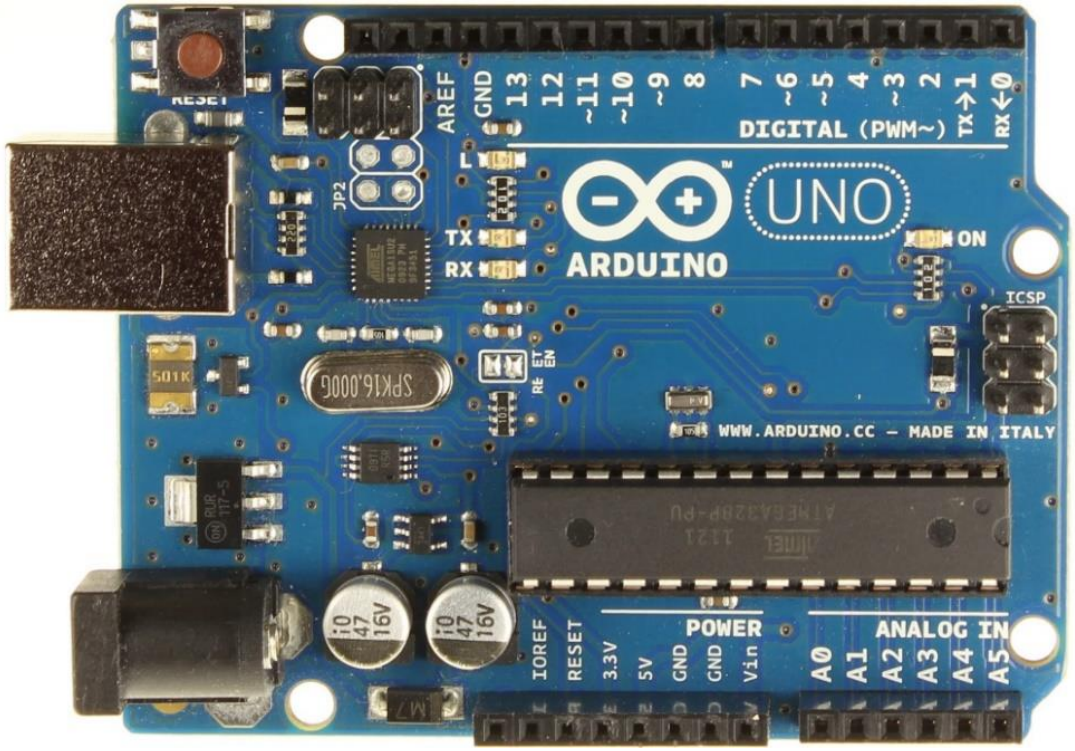
تعليمات تضمين المكتبيات والتعريفات:

التعليمة	شرح التعليمة
#include < …….. >	تعليمة إضافة المكتبيات.
#define new_name real_name	تعريف أسماء مستعارة لأقطاب أو غير ذلك ولا تأخذ أي حجم في الذاكرة.

تعليمات تعريف المداخل والمخارج:

التعليمة	شرح التعليمة
`pinMode (pin, Mode);`	تحديد القطب المطلوب " Pin " كقطب دخل INPUT أو كخرج OUTPUT أو دخل مع تفعيل مقاومة الرفع INPUT_PULLUP.
`digitalWrite(pin ,Mode );`	إعطاء قيمة للقطب المعرف كخرج إما HIGH أو LOW.
`digitalRead (pin);`	قراءة قيمة قطب " Pin" معرف كدخل.
`analogRead(pin)`	قراءة القيم التشابهية على قطب الدخل pin التشابهي (A0 ~ A5) في لوحات Arduino UNO .
`analogWrite(pin , value)`	التحكم بعرض النبضة على الأقطاب المخصصة لتوليد نبضات PWM على لوحات الآردوينو والتي يكون بجانب القطب الإشارة ~، أما القيمة التي يمكن اسنادها فضمن المجال 0 ~ 255.

كيف نحدد القطب المراد التعامل معه من أقطاب لوحة الآردوينو؟؟



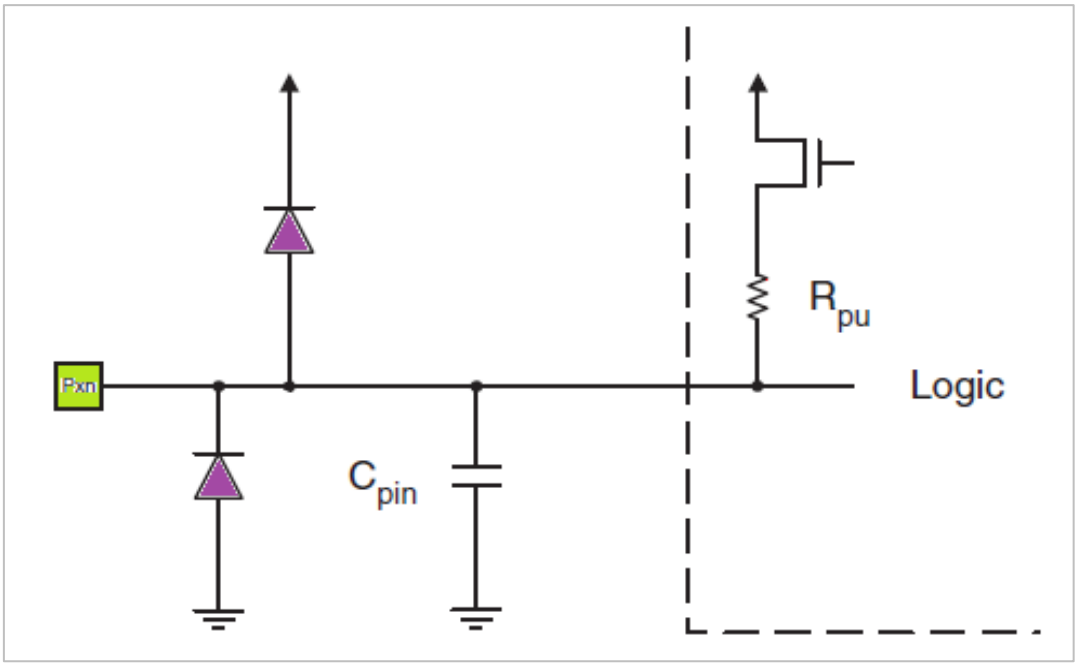
في السابق وعند برمجة متحكمات AVR أو أي نوع من المتحكمات كنا نعود للوثيقة الفنية DataSheet لاختيار الأقطاب المطلوب برمجتها وذلك لمعرفة الرمز الخاص بها، لكن في بيئة برمجة لوحات الآردوينو سيكون الوضع مختلف تمام، فلبرمجة أقطاب الدخل والخرج العامة GPIO يجب أن نكتب رمز القطب الموجود على لوحة الآردوينو نفسها ، هذه الرموز عادة تكون أرقام تبدأ من الصفر وهي الأقطاب العامة وبعض منها يوجد بجانب الرقم إشارة " ~ " للدلالة على أنه يمكن أن يكون خرج لنبضات PWM ، كما يوجد **أقطاب الدخل التشابهية** والتي ترمز بـ **Ax** ويمكن أيضا برمجتها كأقطاب دخل خرج رقمية.

ماذا يقصد بالخيار INPUT_PULLUP.....؟؟

كل قطب من أقطاب المتحكم له مقاومة رفع خاصة به قيمتها 10kR ولهذه المقاومة مسجل خاص لتفعيلها، وظيفة هذه المقاومة هي تأمين قيمة بدائية للدخل قيمتها واحد منطقي بدل أن يبقى قطب المتحكم بدون حالة بدائية لأنه سيكون عرضة للتأثر بالضجيج بشكل كبير.

نلاحظ وجود **ترانزستور** خاص وظيفته تفعيل مقاومة الرفع أو إلغائها، كما يوجد **ديودين** عكسيين أحدهما مع التغذية الموجبة والثاني مع التغذية السالبة لحماية المدخل من الجهد الزائد، و**مكثف** موصل على القطب الأرضي للحماية من الشحنات الكهربائية الساكنة.

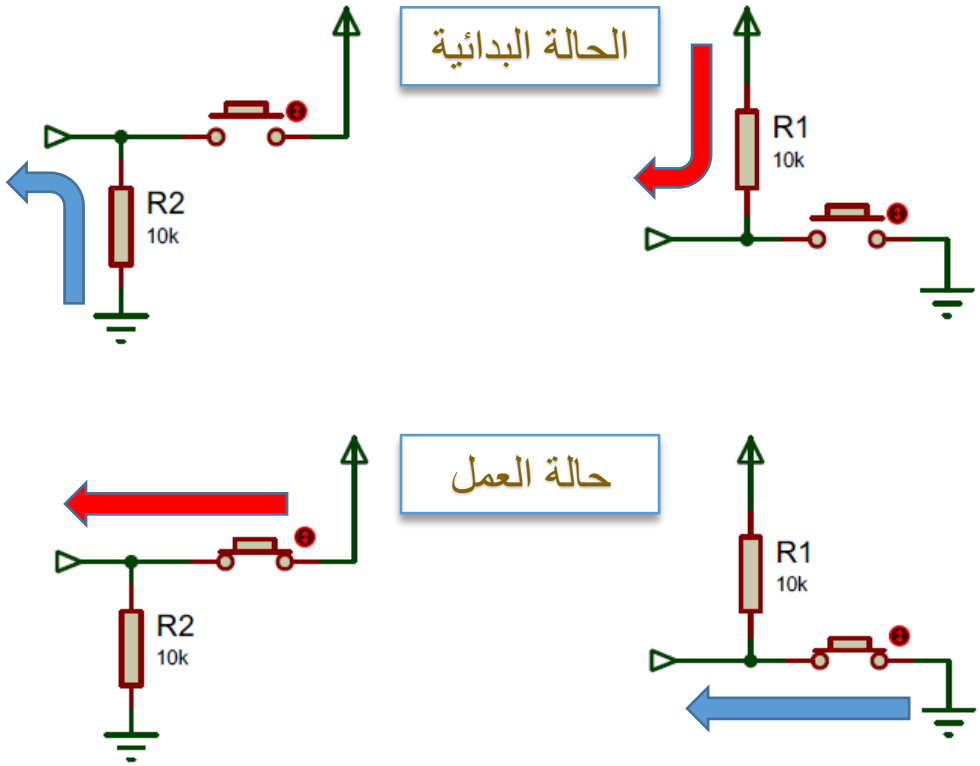
تختلف قيمة مقاومة الرفع بين أنواع المتحكمات لكن بالمجمل فإن قيمة هذه المقاومة تتراوح بين **10 ~ 50 kR** وفي حالات أخرى قد تصل قيمتها حتى **150 kR**.



مقاومة الرفع ومقاومة الخفض:

أو ما يعرف باسم Resister Pull down & Pull up وهي عبارة عن مقاومة ذات قيمة كبيرة تتراوح بين 10 ~ 150 kR وظيفتها تأمين حالة بدائية للقطب المتصلة معه ولها أهمية كبيرة في عملية الاستقرار الكهربائي وحماية أقطاب الدخل من الضجيج الحاصل حول الدارة الكهربائية.

تعتبر عملية تحديد الحالة البدائية لأقطاب المتحكم ضرورية جدا وذلك لمعرفة الحالة التي سوف يتغير إليها القطب والعمل الذي سوف يقوم به المتحكم بناء على التغيرات الحاصلة وهذا ما تؤمنه مقاومات الرفع والخفض، فمقاومة **الرفع** تعطي قيمة بدائية هي **1** منطقي أما مقاومة **الخفض** فتعطي قيمة بدائية هي **0** منطقي، أما آلية توصيل مقاومة الرفع والخفض فتتم كما:



تعليمات التأخير الزمني:

التعليمة	شرح التعليمة
delay(ms);	إيقاف عمل البرنامج لمدة محددة بواحدة الميلي ثانية (1s = 1000 ms)، تأخذ قيم ضمن المجال 0 ~ 4,294,967,295.
delayMicroseconds(us) ;	إيقاف عمل البرنامج لمدة محددة بواحدة الميكرو ثانية (1s = 1000000 us)، تأخذ قيم ضمن المجال 0 ~ 65535 .
millis () ;	تعيد هذه التعليمة قيم بالميلي ثانية حيث تبدأ العد لحظة تشغيل لوحة الآردوينو، وتصل للطفحان (أي لقمة العد ثم تعود للصفر) بعد حوالي 50 يوم، أما نوع المتحول المستخدم فهو من نوع `unsigned long` ، كما أن هذه التعليمة لقراءة قيمة العد فقط ولا يمكن تصفير هذا العداد برمجيا.
micros ();	تعيد قيم بالميكرو ثانية بدأ من لحظة تشغيل البورد وتصل لحالة الطفحان بعد حوالي 70 دقيقة من لحظة تشغيل لوحة الآردوينو (يختلف هذا الرقم من لوحة لأخرى)

المتحولات وأنواعها:

التعليمة	شرح التعليمة
`Boolean var ;`	متحول منطقي له حالتين فقط true or false.
`char var ;` `char x = 'a';`	متحول محرفي واحد بطول بايت واحد 127 ~ 128-.
`byte var ;`	متحول رقمي بطول 8Bit وقيمته تنتمي 255 ~ 0.
`int var ;`	متحول رقمي طبيعي بطول 16Bit قيمته ~ 32768 -32767
`unsigned int var;`	متحول رقمي طبيعي بطول 16Bit غير مؤشر ~ 0 65535.
`long var;`	متحول رقمي بطول 32Bit ينتمي لـ - 2,147,483,648. ~ 2,147,483,647.
`unsigned long var;`	متحول رقمي غير مؤشر بطول 32Bit ينتمي للمجال 0 ~ 4,294,967,295
`short var;`	متحول رقمي بطول 16Bit.
`float var;`	متحول رقمي ذو فاصلة عشرية بطول 32Bit، لكن عدد الأرقام بعد الفاصلة يجب ألا يتجاوز السبع خانات.
`double var;`	متحول رقمي عشري بطول 64Bit.
`String var;` `String a="Arduino";`	تعريف متحول محرفي عدد محارفه يحددها المبرمج.
`integer const var = value;`	تعريف ثابت رقمي محدد القيمة.
`float const var = value;`	تعريف ثابت رقمي بفاصلة عشرية محدد القيمة.

ملاحظات

- ✓ بالنسبة للمتحولات المعرفة كقيم عشرية float يجب وعند التعامل معها أن تكون هناك آلية للتأكد من الجواب إذ أن العمليات الحسابية وعند التعامل مع الأرقام العشرية تكون النتائج غير دقيقة.
- ✓ **عند تعريف أي متحول يجب مراعاة النقاط التالية:**
  - **لا** يجب أن يبدأ اسم المتحول برقم.
  - **اختيار** نوع المتحول بما يتناسب مع حجم البيانات المسندة إليه.
  - **يفضل** أن تكون أسماء المتحولات معبرة عن الوظيفة المسندة إليها حتى ولو كان الاسم طويل.
  - في حال كان اسم المتحول مؤلف من كلمتين **يفضل** كتابة الحرف الأول من الكلمة الثانية بحرف كبير دون ترك مسافة أو الفصل بين الاسمين بـ " _ ".
- ✓ يوجد نقطة مهمة يجب الانتباه إليها عند التعامل مع عدة متحولات من أنواع مختلفة وهو الترقية، فلو كان لدينا عملية حسابية على متحولين من نوعين مختلفين فإن المترجم compiler يقوم بترقية هذه المتحولات نحو النوع الأكبر بينهما ومن ثم يجري العملية الحسابية ويقدم الناتج بما يتوافق مع المتحول الأكبر، وهذا قد يسبب مشكلة في النواتج وخاصة إذا كان أحد المتحولات من نوع float وتم ترقيته لنوع أكبر فسيتم حذف الأرقام التي بعد الفاصلة.
- ✓ المتحول المنطقي من نوع **Boolean** له قيمتين فقط **False** تقابل القيمة **0** و **True** تقابل كل القيم عدا الصفر.

الحلقات الشرطية وحلقات الدوران:

شرح التعليمة	التعليمة
الحلقة الشرطية if والتي تنفذ جملة التعليمات بعد تحقق الشرط أو عدة شروط. **وللتعليمة if توابع هي else if و else وكل منها تنفذ جملة من التعليمات بعد تحقق الشرط.**	`if (conditional_1), (conditional_2) { // Action A } else if (conditional) { //Action B } else { //Action C }`

حلقة for والتي تستخدم لتكرار جملة من التعليمات عدد من المرات يحددها المبرمج. لهذه الحلقة ثلاثة بارامترات، الأول تهيئة initialization يتم فيه وضع قيمة بدائية والتي سوف يبدأ منها العد، البارامتر الثاني هو الشرط condition الذي سيتوقف عنده العد فما دام الشرط محقق فالحلقة سوف تستمر بالعمل، أما البارامتر الثالث فهو للزيادة Increment أو النقصان decrement.	``` for (initialization; condition; increment) { //Statement(s) } ```
الحلقة الشرطية switch وهي حلقة شرطية مشابهة للحلقة if ، فهي تقوم على اختبار حالة المتحول var وتنفيذ جملة من التعليمات تبعا للحالة case التي عليها المتحول وإلا فإن حالة أساسية default سوف تنفذ. كل حالة case يجب إنهاؤها بالتعليمة break وإلا فإنه سوف ينتقل للحالة التالية لتنفيذها ولن يتوقف حتى يجد التعليمة break. وفي حال عدم وجود التعليمة break يتم تنفيذ أيضا الحالة الأساسية default ثم الخروج من الحلقة ومتابعة بقية الكود البرمجي.	``` switch(var) { case 1: // statement break; case 2: //statement breake; default : } ```

حلقة الدوران **while** والتي سوف تدور بشكل مستمر مادام الشرط **expression** الموجود في الحلقة محقق فإن انتفى الشرط فسوف يتم الخروج من الحلقة.	`while (expression)` `{` `  //statement` `}`
حلقة الدوران الشرطية **do ... while** والتي تشابه للحلقة الشرطية **while** لكن الخلاف بينهما أنه في هذه الحلقة يتم تنفيذ التعليمات ثم يتم اختبار الشرط فإن انتفى الشرط يتم عندئذ الخروج من الحلقة.	`do` `{` `  //statement` `}` `while (test condition);`

ملاحظات

- ✓ في الحلقة for ممكن زيادة القيمة المطلوبة بالخطوة التي نحتاجها إذ أن الحالة الأساسية لهذه الحلقة هو الزيادة بمقدار واحد، يتم وضع الخطوة المطلوبة في قسم الزيادة أو النقصان بالعملية الحسابية المطلوبة.
- ✓ يمكن تعقيد الحلقة for وذلك بتعريف أكثر من متحول فيها وكذلك الشرط يمكن ربطه بأكثر من متحول كما في المثال التالي:

```
for (int i = 0, j = 0 ; i + j < 20 ; i++,j+=2)
```

- ✓ عند وجود أكثر من شرط في الحلقة for نفصل بين هذه الشروط بالفاصلة " , " .
- ✓ للخروج من الحلقة (do , for , while , and switch) نضع التعليمة break وعند الوصول لهذه التعليمة يتم الخروج من الحلقة وهو خروج قسري ينتقل فيه المترجم لتنفيذ التعليمات التي خلف الحلقة.
- ✓ نستخدم التعليمة continue في الحلقات (do , for , while , and switch) لإعادة بدء الحلقة من جديد دون متابعة بقية التعليمات الموجودة بعد هذه التعليمة، وبالتالي فهو تكرار لجملة محددة من تعليمات الحلقة دون غيرها لعدد مرات أكثر يرتقيها المبرمج.
- ✓ عند التعامل مع الجملة الشرطية if سوف يكون لدينا حالتين لكتابة هذه الحلقة:

- الأولى: جعل الشرط الأول يبدأ بـ if والشرط الثاني بـ else if والثالث.... ثم نختم بـ العبارة else.

- الثانية: يكون الشرط الأول بـ if والثاني كذلك بـ if و..... وننهي بـ else.

فما هو الفرق بين هاتين الحالتين...؟؟؟؟

في الحالة الأولى: سيتم اختبار العبارات الشرطية ومن ثم سينفذ البرنامج الشروط المحققة من الشروط الموجودة، وفي حال تحقق ولو شرط واحد فلن ينفذ عندها التعليمات الموجودة داخل الجملة `else`.

في الحالة الثانية: سيتم تنفيذ الجمل الشرطية المتحققة مهما كان عددها كون البرنامج سيعتبر كل جملة شرطية مستقلة عن الثانية، مع فارق هنا أن `else` متعلقة فقط بالجملة الشرطية `if` الأخيرة دون سابقاتها.

```
if(digitalRead(B1) == 0 )
  {
   digitalWrite(12,1);
  }
if(digitalRead(B2) == 0 )
  {
   digitalWrite(13,1);
  }
else
  {
   digitalWrite(12,0);
   digitalWrite(13,0);
  }
```

```
if(digitalRead(B1) == 0 )
  {
   digitalWrite(12,1);
  }
else if(digitalRead(B2)== 0)
  {
   digitalWrite(13,1);
  }
else
  {
   digitalWrite(12,0);
   digitalWrite(13,0);
  }
```

✓ في الحلقة الشرطية If إذا كان ما بعد تحقق الشرط **Action** عبارة عن تعليمة واحدة فعندئذ يمكن الاستغناء عن الأقواس " {.....} ".

العمليات الحسابية Arithmetic Operator:

شرح العملية	العملية الحسابية
الجمع addition	+
الطرح subtraction	-
الضرب multiplication	*
القسمة division	/
تعيد باقي القسمة modulo (باقي قسمة عدد صحيح على عدد صحيح)	%
عامل التعيين assignment operator	=

عمليات المقارنة Comparison Operator:

شرح العملية	العملية
مساوي لـ equal to	==
لا يساوي not equal	!=
أقل من less than	<
أكثر من greater than	>
أقل أو يساوي less than or equal to	<=
أكثر من أو يساوي greater than or equal to	>=

تتم هذه التعليمات على أي نوع من المتحولات لكن النتيجة تكون من نوع Boolean أي قيم منطقية فقط False or True .

العمليات الخاصة بالبت Bitwise Operator:

شرح التعليمة	التعليمة
Bitwise and Bit 1 Bit 2 Bit 1 and Bit 2 0 0 0 0 1 0 1 0 0 1 1 1	&
Bitwise or Bit 1 Bit 2 Bit 1 or Bit 2 0 0 0 0 1 1 1 0 1 1 1 1	|
Bitwise xor Bit 1 Bit 2 Bit 1 ^ Bit 2 0 0 0 0 1 1 1 0 1 1 1 0	^
Bitwise not تعكس حالة البت.	~
Bitshift left الإزاحة لليسار: هي عملية حذف خانة من اليسار والاستعاضة عنها بخانة صفرية في اليمين.	<<
Bitshift right الإزاحة لليمين: هي عملية حذف خانة من اليمين والاستعاضة عنها بخانة صفرية في اليسار.	>>

التعليمة	شرح التعليمة
`bit(bitPosition)`	**حساب قيمة البت المحدد رقمه (فالبت الذي رقمه 0 فقيمته 1 والبت الذي رقمه 1 قيمته 2 وهكذا).**
`bitRead(x,bitPosition)`	**قراءة موقع البت من المتحول x.**
`bitSet(x, bitPosition)`	**ضبط قيمة البت المحدد من المتحول بالقيمة 1.**
`bitClear(x, bitPosition)`	**ضبط قيمة البت المحدد من المتحول بالقيمة 0.**
`bitWrite(x, bitPosition,value)`	**ضبط القيمة المحددة (value = 0 or 1) للبت المحدد من المتحول x.**

تستخدم هذه التعليمات في عملية قراءة قيمة بت أو تعديل قيمته ولها استخدام كبير في عملية تعديل قيم أقطاب الخرج وفقا لحالات محددة والأمثلة الموجودة في آخر الفصل سوف توضح لنا كيفية تسخير هذه التعليمات بالشكل الأمثلي.

العمليات المركبة Compound Operators:

شرح التعليمة	التعليمة
زيادة	++
إنقاص	--
إنقاص من	-= x -= y //x=x-y
إضافة إلى	+= x += y //x=x+y
ضرب إلى	*= x *= y //x=x*y
قسمة على	/= x /= y //x=x/y
باقي قسمة	%= x %= y //x=x%y

ملاحظات

x++ سيزيد قيمة x بمقدار واحد ويقدم القيمة القديمة للمتحول x ، بينما ++x سيزيد x بمقدار واحد ومن ثم يقدم القيمة الجديدة للمتحول x , وكذلك الأمر بالنسبة لعملية الإنقاص.

العمليات المنطقية Boolean Operator:

شرح التعليمة	التعليمة
and	&&
or	||
Not	!

تعليمات خاصة بأقطاب الدخل/الخرج العامة GPIOs:

شرح التعليمة	التعليمة
توليد أمواج مربعة بترددات محددة على القطب pin، شرط ألا يكون تردد الموجة أقل من 31Hz ولا يزيد عن القيمة 65535. Frequency: يمكن أن يعرف كمتحول من نوع unsigned int. Duration: المدة الزمنية لصدور النغمة.	`tone(pin , frequency , duration);`
إيقاف توليد النغمات على القطب المحدد.	`noTone(pin);`
تستخدم هذه التعليمة لقراءة عرض النبضة المطبقة على القطب المحدد pin، أما القيمة Value فتأخذ إحدى القيمتين: - HIGH: حيث ينتظر حتى تصبح الموجة الداخلة على القطب في حالة HIGH فيبدأ العد حتى تعود للقيمة LOW وعندها يتوقف عن العد.	`pulseIn(pin , Value);`

- **LOW: ينتظر حتى تصبح النبضة المطبقة على القطب في حالة LOW عندئذ يبدأ بالعد حتى تعود لحالة HIGH وعندها يتوقف العد.** **القيمة التي تعيدها هذه التعليمة بواحدة uSec، وأعلى قيمة للنبضة 65535 uS ،ولكي تعمل هذه التعليمة مع القطب المحدد يجب أن يعرف القطب كقطب دخل INPUT.**	

وهي تعليمات لها نتائج خاصة ولا تتبع لمكتبيات خاصة ولهذه التعليمات أهمية كبيرة في بعض التطبيقات التي سنتطرق لها لاحقا.

التعليمات الحسابية على القيم:

وهي جملة من التعليمات الخاصة بالعمليات الحسابية كرفع العدد للقوة وحساب الجذر وغير ذلك من العمليات التي توفر على المبرمج الكثير من الوقت والجهد:

شرح التعليمة	التعليمة
نستخدم هذه التعليمة للحصول على مجال تغير جديد (toLow , toHigh) للقيمة value من المجال القديم (fromLow , fromHigh) الذي تتغير فيه هذه القيمة.	`newValue = map(value, fromLow, fromHigh, toLow, toHigh);`
تعليمة جبر القيمة، حيث تقوم هذه التعليمة بجبر قيمة x حسب الحالات التالية: • **x < a : تصبح قيمة x = a.** • **x > b: تصبح قيمة x = b.**	`val = constrain(x,a,b);`

• a < x < b : تبقى قيمة x كما هي.	
يعيد القيمة الأصغر من بين قيمتين فقط.	`val = min(x, y);`
يعيد القيمة الأكبر من بين قيمتين فقط.	`val = max(x, y);`
يعيد القيمة المطلقة للقيمة x.	`val = abs(x);`
يرفع العدد (float) base للقوة (float) exponent ويعيد الناتج val (double).	`val = pow(base, exponent);`
حساب الجذر التربيعي للعدد x.	`val = sqrt(x);`
حساب مربع العدد x.	`val = sq(x);`
تقريب العدد العشري لأصغر عدد صحيح.	`floor(val) ;`
تقريب العدد العشري لأكبر عدد صحيح.	`ceil(val) ;`

التعليمات الهندسية للزوايا Trigonometry:

شرح التعليمة	التعليمة
حساب جيب الزاوية.	`val = sin(rad);`
حساب تجيب الزاوية.	`val = cos(rad);`
حساب ظل الزاوية.	`val = tan(rad);`

عند التعامل مع هذه التعليمات يجب أن تكون قيمة الزاوية المراد حسابها بالراديان radians (float) أما الناتج فسوف يكون ضمن المجال 1 ~ 1-.

المصفوفات Arrays:

عند التعامل مع حجم كبير من البيانات كنا نستخدم الكثير من المتغيرات لتخزين ومعالجة تلك البيانات، وذلك لأن كل متحول يحمل قيمة واحدة فقط وهذا ما يجعل البرنامج طويل ومعقد، لكن وباستخدام المصفوفة يستطيع المبرمج استعمال متغيرات قليلة وذلك بتقسيم كل متغير إلى عدد من العناصر المتسلسلة، كما تسهل المصفوفة التعامل مع جملة من المتحولات بنفس الوقت دون تعقيد وباختصار من الوقت..... صحيح أن الكثير من المقبلين على تعلم لغة الآردوينو لديهم الكثير من التخوف من استخدام المصفوفة لكن نبشرهم بأن الموضوع سهل بإذن الله ويحتاج فقط للقليل من التركيزمع الكثير من التجريب.

المصفوفة: عبارة عن منطقة محددة من **الذاكرة** تتكون من عدد **محدد** (رقم صحيح يستخدم لتحديد عدد المواقع المطلوبة من الذاكرة ويسمى هذا الرقم **بالدليل Index**) **ومتجانس** (أي أن جميع عناصر المصفوفة من نفس نوع البيانات **Data Type**) من المواقع المتجاورة، ويمكن تعريف مصفوفات بعدة أبعاد فمنها ما هو ببعد واحد (سطر واحد بعدة أعمدة أو العكس) ومنها ما هو ببعدين فتكون بعدة أسطر وعدة أعمدة ، وهذين النوعين هما نقطة اهتمامنا كما يمكن تعريف مصفوفة بثلاثة أبعاد وهذا النوع ليس مجال عملنا (يتم عادة استخدام هذه المصفوفات للتعامل مع الصور مثلا فلكل نقطة من الشاشة ثلاثة بارامترات هي الألوان الأساسية R G B وبالتالي لكل بيكسل ثلاثة قيم هي قيم هذه الألوان وهنا نحن بحاجة لمصفوة بثلاثة أبعاد وهكذا).

المصفوفة ذات البعد الواحد:

وهي مصفوفة ذات سطر واحد تعرف كما يلي:

```
Type Name[size];
```

Type: نوع متحولات المصفوفة.

Name: اسم المصفوفة.

size: عدد عناصر المصفوفة ،كل عنصر يأخذ رقم يدل عليه بدء من الرقم 0 يسمى Index.

لتعريف مصفوفة بقيمة بدائية هي الصفر نكتبها بالشكل التالي:

```
Type Name[size] = { } ;
```

يمكن كذلك تحديد قيم بدائية لبعض عناصر المصفوفة وترك بقية العناصر دون قيم لتأخذ بذلك القيمة صفر كقيمة بدائية.

القراءة والكتابة في المصفوفة أحادية البعد:

الكتابة: وتتم بذكرمؤشر العنصر من المصفوفة المطلوبة وإعطاء هذا العنصر القيمة التي نريدها كقيمة جديدة له، فهنا قمنا بإسناد القيمة `value` للعنصر الذي ترتيبه `x` من المصفوفة `ArrayName`.

```
ArrayName [x] = value;
```

القراءة: وهي عملية معاكسة للكتابة حيث نقوم بإسناد قيمة العنصر الذي ترتيبه `x` من المصفوفة `ArrayName` للمتحول `value`.

```
value = ArrayName [x];
```

مثـــال:

ليكن لدينا المصفوفة التي تحتوي 10عناصر:

{2 , 8 , 14 , 5 , 3 ,7 , 10 , 12 , 0 , 5 }

نريد زيادة القيمة 5 لكل عنصر من عناصر المصفوفة ومن ثم طباعة قيم المصفوفة الجديدة على المنفذ التسلسلي، ونريد تكرار هذه العملية لثلاثة مرات.

```
int MyArray[10]={2,8,14,5,3,7,10,12,0,5};
int x = 0 ;
void setup()
{
Serial.begin(9600);
}

void loop()
{
if (x < 3 )
{
for(int i=0 ; i <10 ; i++)
  {
   MyArray[i] = MyArray[i] + 5 ;
   Serial.println(MyArray[i]);
   delay(200);
  }
  x++;
  Serial.print("Numper =");
  Serial.println(x);
  }}
```

المصفوفة ذات البعدين:

وهي مصفوفة لها عدة أسطر row وأعمدة column وتعرف بالشكل التالي:

```
Type Name[row size] [column size] ;
```

Type: نوع متحولات المصفوفة.

Name: اسم المصفوفة.

row size: عدد أسطر المصفوفة.

column size: عدد أعمدة المصفوفة.

تكتب عناصر السطر الواحد بي قوسين"{ , , }" ويفصل بينها بفاصلة ، ويفصل بين الأسطر كذلك بفاصلة.

```
byte array1[2][3] = {{5,3,70},{8,4,0}};
```

القراءة والكتابة في المصفوفة ذات البعدين:

الكتابة: وهنا علينا ذكر مكان العنصر في المصفوفة فنذكر مؤشر العنصر بالنسبة للسطر أولا ثم نذكر مؤشر العنصر بالنسبة للأعمدة، ثم نسند للعنصر القيمة التي نريدها كما هو موضح.

```
.ArrayName [x][y] = value;
```

القراءة: بعملية معاكسة لما سبق إذ علينا إسناد قيمة العنصر الذي مؤشر الأسطر له يحمل القيمة x ، بينما مؤشر الأعمدة يحمل القيمة y من المصفوفة ArrayName للمتحول value.

```
value = ArrayName [x][y];
```

مثـــال:

ليكن لدينا المصفوفة التالية:

$$\text{array} = \begin{pmatrix} 85 & 66 & -3 \\ -2 & 10 & 5 \\ 9 & 72 & 13 \end{pmatrix}$$

والمطلوب طباعة عناصر المصفوفة على نافذة الاتصال التسلسلي وذلك بعد إضافة القيمة 5 لكل عنصر من عناصر هذه المصفوفة.

```
int a[3][3]={{85,66,-3},{-2,10,5},{9,72,13}};

void setup()
{
Serial.begin(9600);
for(int i=0 ; i<3 ; i++)
 {
  for(int j=0 ; j<3 ; j++)
  {
    a[i][j] = a[i][j] + 5;
    Serial.print(a[i][j]);
    Serial.print("  ");
  }
  Serial.println();
 }
}

void loop()
{}
```

التوابع (الدوال) Functions:

وهو مجموعة من الأوامر والبيانات المرتبة تحت اسم واحد والتي يمكن استدعاؤها من أماكن مختلفة من الكود البرمجي، كما تعرف أيضا بالروتين الثانوي **subroutine**.

إن استخدام التوابع يقدم العديد من الميزات والخصائص للمبرمج أهمها:

- اختصار الكود البرمجي، إذ يكتفى باستدعائه باسمه فقط لينفذ العمل المطلوب.
- تلافي خطوات التكرار في الكود البرمجي.
- تساعد التوابع في عملية البرمجة نفسها.
- توفر مساحة من الذاكرة المطلوبة للبرنامج.
- اختصار عملية البرمجة وتنفيذ البرنامج بأسرع وقت ممكن.
- تسهيل عملية مراجعة الكود وتصحيح الأخطاء والتعديل على الكود.

شكل التابع في بيئة الآردوينو:

يكتب التابع ويعرف في بيئة **Arduino IDE** كما يعرف بلغة **C/C++** بالشكل التالي:

```
Function_Type  name ( Parameters )
{
    statement ;
    return result ;
}
```

- `Function_Type`: نوع القيمة التي سوف يرجعها التابع (int , float , ...) وفي حال لم يكن هناك قيمة سيتم إرجاعها عندها يكون التابع من النوع void.
- `name`: اسم التابع.
- `Parameters`: القيم المراد تمريرها للتابع، ويمكن ألا يكون هناك قيم.
- `statements`: البنية البرمجية للتابع.
- `return result`: القيمة التي سوف يتم إعادتها، حيث أن المتحول result من نفس نوع التابع وإذا لم يكن هناك قيمة معادة فلا حاجة لهذه التعليمة.

خطوات تعريف التابع في لغة C/C++:

لبناء أي تابع أو برنامج فرعي نقوم بتحديد بنيته في القسم النهائي من الكود البرمجي (وهذا أفضل مكان لكتابة البرنامج الفرعي، لكن في حالات معينة يكون مكان التابع التفرعي في بداية الكود البرمجي نذكرها في وقتها إن شاء الله)، أما استدعاء التابع فيمكن أن تتم في أي نقطة من الكود البرمجي وتتم العملية بذكر اسم التابع ; ( ) `name` وتحديد قيم الدخل له إن وجدت، كل ذا ستوضحه الأمثلة القادمة.

أنواع التوابع في لغة C/C++:

للتوابع أشكال عديدة وتختلف فيما بينها في قيم الدخل للتابع والناتج الذي سيقدمه التابع فتقسم بذلك التوابع لأربعة أقسام هي:

- **تابع ليس له دخل وليس له خرج:**

لنكتب **تابع برمجي** يقوم بإطفاء **الليدات** المتصلة مع لوحة الآردوينو على الأقطاب بين 3 ~ 8 وذلك كلما تم استدعاء التابع.

```
void setup()
{
for(int i = 3 ; i<9 ; i++)
{
  pinMode(i , OUTPUT);
}
}

void loop()
{
  led_off();
}

void led_off()
{
  for(int i = 3 ; i<9 ; i++)
{
  digitalWrite(i , 0);
}
```

في هذ التابع نلاحظ كيف يتم استدعاء التابع الخاص بعملية إطفاء الليدات `led_off();` في الحلقة الأساسية للكود البرمجي، كما نلاحظ أن التابع لا يعيد أي قيمة رقمية للمستخدم وليس له أي قيمة دخل.

- **تابع له دخل وليس له خرج.**

لنكتب تابع برمجي يتم من خلاله اختيار رقم قناة PWM و**القيمة** المراد إخراجها على القناة.

```
void setup()
{


}


void loop()
{
  PWM_control(3,150);
}


void PWM_control(byte PWM_channal ,byte PWM_value)
{
 analogWrite(PWM_channal , PWM_value);
}
```

نلاحظ أن لهذا التابع مدخلين من نوع `byte` أحدهما يحدد رقم القناة والثاني يحدد القيمة المطلوب إخراجها على القناة، أما التابع فنجد أنه لا يعيد أي قيمة رقمية للمستخدم.

- **تابع ليس له دخل لكن له خرج.**

لنكتب تابع برمجي يتم من خلاله قراءة قناة مبدل تشابهي رقمي **ADC** وتحويل القيمة المقروءة لقيمة تقابلها من المجال **0 ~ 255** وإعادة القيمة للمستخدم.

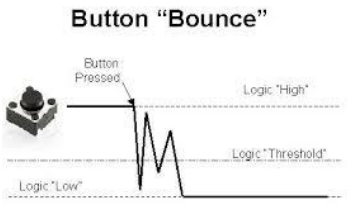
```
void setup()
{

}

void loop()
{
int x ;
x = read_value();
Serial.println(x);
delay(500);
}

int read_value()
{
int a ;
a = analogRead(A0);
a = map(a , 0,1023,0,255);
return a;
}
```

التابع السابق عبارة عن تابع يتم استدعاءه لمعرفة النتيجة التي هي عبارة عن قراءة المدخل التشابهي A0 وتحويل الناتج ضمن عملية map ومن ثم يعيد الناتج للبرنامج.

- **تابع له دخل وله خرج.**

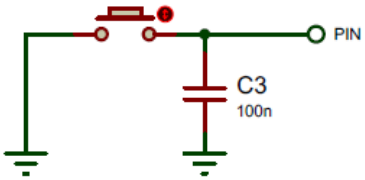
لنعرف تابع مدخله متحولين هما x , y من النوع integer ، اما العملية الحسابية التي تتم داخل التابع فهي من الشكل التالي: **3*x + 9*y** ، أما خرج التابع فهو من النوع long.

```
void setup()
{
 long myFunction;
}

void loop()
{
int x = 3;
int y = 9;
long k ;
k = myFunction(x , y);
}

long myFunction(int i , int j)
{
  long v;
  v = 3*i + 9*j;
  return v;
}
```



Bounce: وهي حالة فيزيائية ميكانيكية تنتج في الكباسات اللحظية عند الضغط عليها مما يولد عندنا حالة غير مستقرة لفترة بسيطة من الزمن وهي حالة غير مرغوبة حيث ينتج عنها فهم خاطئ تدل على أن المفتاح قد تم ضغطه عدة مرات، لكنه في الواقع لم يضغط إلا مرة واحدة.

عملية التخلص من هذا الارتداد تسمى **De bounce** ولها طريقتين:



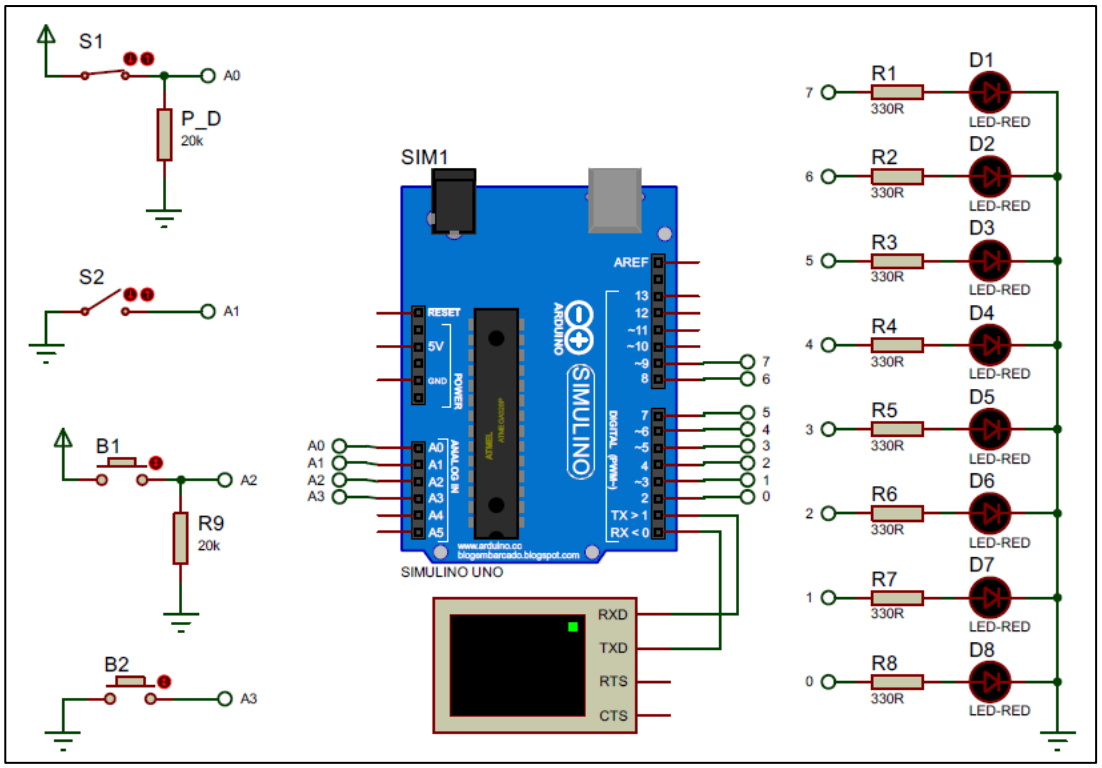
الأولى فيزيائية حيث يتم توصيل مكثف بقيمة **100nF** على التفرع مع الكباس اللحظي إلى الأرضي وبالتالي يتم التخلص من حالة الارتداد الفيزيائية كما في الشكل.

الثانية برمجية وهنا يتم وضع تأخير زمني لعملية قراءة حالة المفتاح تقدر هذه المدة بما يزيد عن 50ms، لكن لهذه الطريقة مشكلة أخرى وهي عملية تأخير الكود البرمجي وإدخاله في حلقة لانهائية لمدة 50ms ويتم التخلص من هذه المشكلة في الآردوينو باستخدام تعليمة millis()، وهو قراءة حالة المفتاح في المرة الأولى وبعد مدة محددة يتم قراءة حالة المفتاح مرة ثانية فإن كانت النتيجة واحدة فالمفتاح قد تم بالفعل ضغطه وهنا يتم تنفيذ الأوامر المتعلقة بعملية ضغط المفتاح.

أمـثلة عـمليـة

بعد استعراض مجموعة من تعليمات بيئة التطوير **Arduino IDE** سنقدم الآن عدد من الأمثلة العملية للتعليمات السابقة، ومن خلال هذه التعليمات سوف نبدأ رحلة التعامل مع لوحات الآردوينو.

لتكن لدينا لوحة الآردوينو **Arduino UNO** وتم توصيل عدد من الليدات **LEDs** والمفاتيح **switch** والكباسات اللحظية **Button** إليها كما في الشكل:



آلية التوصيل:

- **D0 ~ D7**: موصلة مع الأقطاب **2 ~ 9** من لوحة الآردوينو.
- المفاتيح (S1 = A0 , S2 = A1) والكباسات اللحظية (B1 = A2 , B2 = A3).

مثـــال 1 :

لنكتب برنامج يقوم بتشغيل الليد D0 لمدة ثانية ثم يطفئ الليد لمدة ثانية مع الاستمرار هكذا في الحلقة اللانهائية.

```
const int led = 2;
void setup() {
 pinMode(led,OUTPUT);}
void loop() {
digitalWrite(led,HIGH);
 delay(1000);
 digitalWrite(led,LOW);
 delay(1000);
}
```

```
#define led_0  2
boolean i ;
void setup(){
  pinMode(led_0 , OUTPUT);}
void loop()
{
 i = !i ;
 digitalWrite(led_0 , i);
 delay(1000);
}
```

في بداية الكود البرمجي تم التعبير عن القطب الذي رقمه 2 على لوحة الآردوينو بالاسم المستعار led وبالتالي فإن أي تعليمة تحتوي على الاسم led فهي تدل على القطب 2، ثم تحديد طبيعة القطب led على أنه قطب خرج OUTPUT، ثـم وضمن الحـلقة اللانهـائية للبرنـامـج void loop() سيتم تفعيل القطب led لمدة ثانية ومن ثم إطفائه لمدة ثانية وتستمر الحلقة اللانهائية بالدوران.

مثـــال 2 :

لنكتب كود برمجي يتم فيه تشغيل الليد D0 عند ضغط الكباس اللحظي B1 ويطفئ الليد D0 في حال عدم ضغط المفتاح B1.

بالعودة لمخطط المحاكاة السابق نجد أن المفتاح B1 له قيمة بدائية محددة هي القيمة صفر وذلك عبر مقاومة الخفض Pull Down المتصلة مع الأرضي والتي قيمتها 20kR.

```
const int led =2;
const int button=A2;
boolean state;

void setup() {
 pinMode(led,OUTPUT);
 pinMode(button,INPUT);
}
void loop() {
  state = digitalRead(button);
  if(state==1){
    digitalWrite(led,HIGH);
  }
  else{
    digitalWrite(led,LOW);
  }
}
```

يتم قراءة قيمة المفتاح واسنادها للمتحول المنطقي state فإن كان قيمة المتحول 1 فيتم عندئذ تفعيل الليد D0 وغير ذلك يكون الليد في حالة إطفاء.

مثـــال 3 :

اكتب كود برمجي يقوم بتشغيل الليدات D0 ~ D7 وإطفاؤها بشكل متتالي بفاصل زمني قدره واحد ثانية.

```
int x ;
void setup() {
  pinMode(0,OUTPUT);
  pinMode(1,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
}

void loop() {
for(x=0;x<=7;x++){
  digitalWrite(x,HIGH);
  delay(1000);
  digitalWrite(x,LOW);
  delay(1000);
}
}
```

مثـــال 4 :

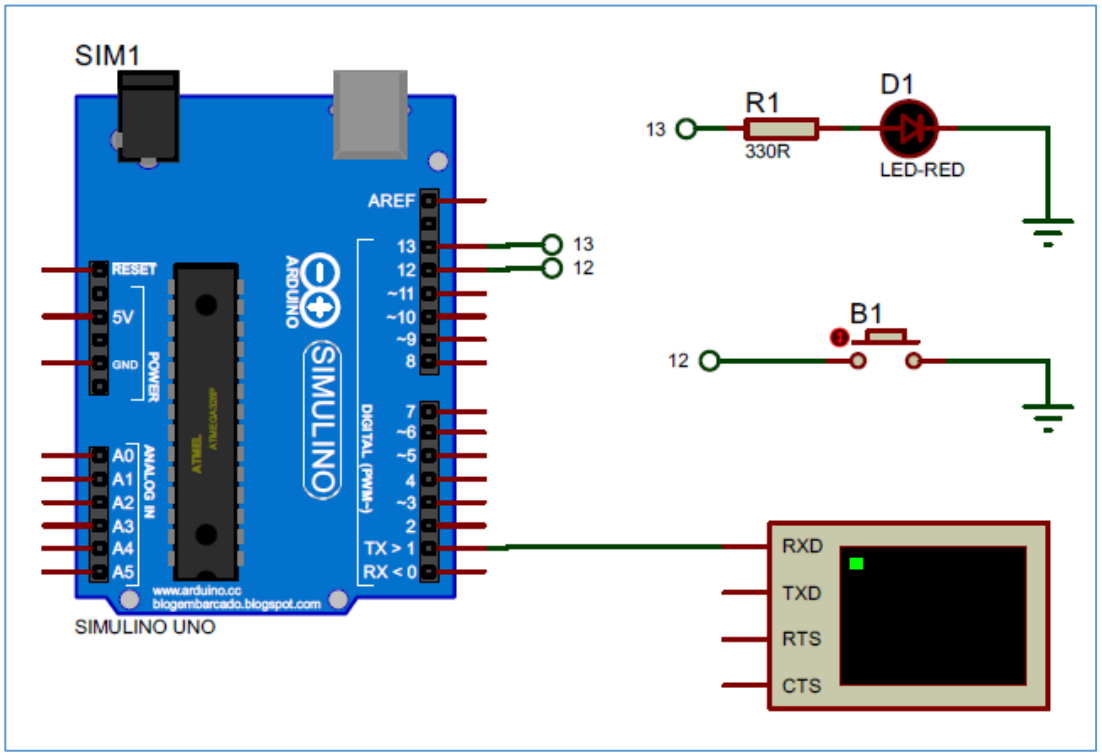
لنكتب كود برمجي يشغل الليدات D0 ~ D7 بشكل متتالي وبفاصل زمني 1sec عند عمل المفتاح S1 , أما عند تشغيل المفتاح S2 فيتم تشغيل كل الليدات.

```
const int S1=A0;
const int S2=A1;
int x ;
int state_1;
int state_2;

void setup() {
  pinMode(S1,INPUT);
  pinMode(S2,INPUT_PULLUP);

  pinMode(0,OUTPUT);
  pinMode(1,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
}

void loop() {
  state_1=digitalRead(S1);
  state_2=digitalRead(S2);
if(state_1==1){
```

```
  for(x=0;x<=7;x++){
    digitalWrite(x,HIGH);
    delay(1000);
    digitalWrite(x,LOW);
  }
}
if(state_2==0){
  digitalWrite(0,HIGH);
  digitalWrite(1,HIGH);
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
}
else{
  digitalWrite(0,LOW);
  digitalWrite(1,LOW);
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
}
}
```

مثـــال 5 :

اكتب كود برمجي يطبع حالة المفتاح B1 الموصل مع القطب 12 كما يطبع حالة الليد المتصل مع القطب 13 الذي يعمل مع عمل المفتاح B1 كما في المخطط التالي:



الكود البرمجي:

```
int led=13;
int button = 12;
int state;

void setup() {
  Serial.begin(115200);
  pinMode(led,OUTPUT);
```

```
  pinMode(button,INPUT_PULLUP);
}

void loop() {
 state=digitalRead(button);
 if (state==0){
  Serial.println("LED ON");
  digitalWrite(led,HIGH);
  delay(300);
 }
 else if(state==1){
  Serial.println("LED OFF");
  digitalWrite(led,LOW);
  delay(500);
 }
}
```

في بداية الكود قمنا بإسناد الأسماء المستعارة للمداخل و المخارج التي نريد التعامل معها حيث قمنا بتسمية القطب المتصل معه الليد باسم led والقطب الذي قمنا وصل الكباس اللحظي معه باسم button ومن ثم حددنا طبيعة هذه الأقطاب مع إعطاء قيمة بدائية للمدخل بتفعيل مقاومة الرفع فأصبحت الحالة البدائية للدخل button هي 1 منطقي ، وحددنا معدل سرعة النقل بـ 115200 bps .

في حلقة الكود الرئيسية يتم اختبار حالة المفتاح فإن كان مضغوط فإن القيمة الناتجة هي 0 وبالتالي سيطبع العبارة LED ON وسيفعل الليد وغير ذلك سيطبع LED OFF وسيطفئ الليد.

مثـــال 6 :

اكتب كود برمجي برنامجه الرئيسي يقوم بتشغيل الليد D0 لمدة ثانية ويطفئ لمدة ثانية، بينما هناك برنامج ثانوي مستقل وظيفته تشغيل الليد D1 بعد مرور زمن قيمته 500000 من بدء عمل التعليمة millis.

```
unsigned long time_work = 50000;
unsigned long time_mome ;
//#define led1  2

int led1 = 2 ;  // work normally
int led2 = 3 ;  // work at time

void setup()
{
Serial.begin(9600);
pinMode(led1 , OUTPUT);
pinMode(led2 , OUTPUT);
}

void loop()
{
  digitalWrite(led1 , HIGH);
  delay(1000);
  digitalWrite(led1 , LOW);
  delay(1000);
```

```
  time_mome = millis();
  Serial.println(time_mome);


  if(time_mome >= time_work )
  {
    digitalWrite(led2 , HIGH);
  }
}
```

مثـــال 7 :

اكتب كود برمجي يقوم فيه الكباس الأول بتشغيل كل الليدات بينما يقوم الآخر بإيقاف جميع الليدات.

```
#define B1   A2
#define B2   A3

void setup()
{
 for(int i = 2; i < 10 ; i++)
 {
  pinMode(i , OUTPUT);
 }
 pinMode(B1 ,INPUT);
 pinMode(B2 ,INPUT_PULLUP);
}

void loop()
{
if(digitalRead(B1) == 1)
{
  LED_ON();
}
if(digitalRead(B2) == 0)
{
  LED_OFF();
}

}
```

```
void LED_ON()
{
  for(int i = 2 ; i<10 ; i++)
  {
    digitalWrite(i , HIGH);
  }
  return;
}

void LED_OFF()
{
  for(int i = 2 ; i<10 ; i++)
  {
    digitalWrite(i , LOW);
  }
 return;
}
```

مثـــال 8 :

لنكتب كود برمجي يقوم بالتخلص من حالة الاهتزاز الميكانيكي Debounce بالاعتماد على تعليمة millis().

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;   / the number of the pushbutton
pin
const int ledPin = 13;      // the number of the LED pin

// Variables will change:
int ledState = HIGH;         // the current state of the
output pin
int buttonState;             // the current reading from
the input pin
int lastButtonState = LOW;    // the previous reading
from the input pin

// the following variables are unsigned long's because
the time, measured in miliseconds,
// will quickly become a bigger number than can be stored
in an int.
unsigned long lastDebounceTime = 0;   // the last time
the output pin was toggled
unsigned long debounceDelay = 50;     // the debounce
time; increase if the output flickers

void setup() {
  pinMode(buttonPin, INPUT);
```

```
  pinMode(ledPin, OUTPUT);

  // set initial LED state
  digitalWrite(ledPin, ledState);
}

void loop() {
  // read the state of the switch into a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed the button
  // (i.e. the input went from LOW to HIGH),  and you've
waited
  // long enough since the last press to ignore any
noise:

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for
longer
    // than the debounce delay, so take it as the actual
current state:
```

```
    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;

      // only toggle the LED if the new button state is
HIGH
      if (buttonState == HIGH) {
        ledState = !ledState;
      }
    }
  }

  // set the LED:
  digitalWrite(ledPin, ledState);

  // save the reading.  Next time through the loop,
  // it'll be the lastButtonState:
  lastButtonState = reading;
}
```

مثـــال 9 :

لنكتب كود يقوم بتوليد نغمات على ثلاثة Buzzers موصولة على الأقطاب 8 , 7 , 6 من لوحة Arduino UNO.

```
void setup()
{

}

void loop()
{
  // turn off tone function for pin 8:
  noTone(3);
  // play a note on pin 6 for 200 ms:
  tone(3, 440, 200);
  delay(200);

  // play a note on pin 7 for 500 ms:
  tone(3, 494, 500);
  delay(500);


  // play a note on pin 8 for 500 ms:
  tone(3, 523, 300);
  delay(300);

}
```

مثـــال 10

في هذا الكود سنستخدم التعليمة pulseIn() لحساب عرض المطبقة على القطب 3 من لوحة Arduino UNO وطباعة قيمة عرض النبضة على واجهة الاتصال التسلسلية UART.

```
long  durtion ;
void setup()
{
pinMode(3 ,INPUT);
Serial.begin(9600);
}

void loop()
{
 durtion = 0 ;
 durtion = pulseIn(3 , HIGH);
 Serial.print(durtion);
 Serial.println("uSec");
 delay(200);
}
```

مثـــال 11

لنكتب كود برمجي يقوم بقراءة قيمة المقاومة المتغيرة التي يمثل مقسم للجهد والمتصلة مع القطب A0 وطباعة الناتج على نافذة الاتصال التسلسلي.

```
int value ;
void setup()
{
  Serial.begin(115200);

}

void loop()
{
value = analogRead(A0);
Serial.println(value);
delay(200);
}
```

مثـــال 12

لنكتب كود برمجي يقوم بتوليد نبضات PWM للتحكم بشدة إضاءة LED المتصل مع القطب 9 من لوحة الآردوينو.

```
void setup()
{
  Serial.begin(9600);
  pinMode(9 , OUTPUT);
}

void loop()
{
for(int i = 0 ; i < 255 ; i++)
  {
  analogWrite(9 , i);
  Serial.println(i);
  delay(60);
  }
}
```

مثـــال 13

ليكن لدينا المصفوفة التالية:

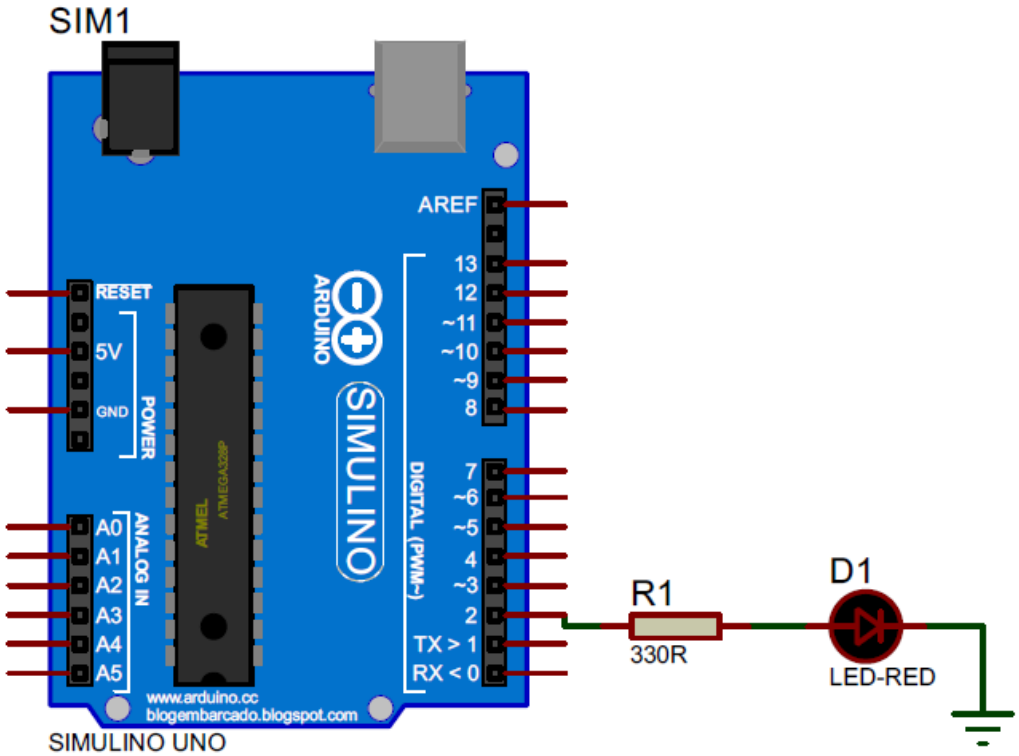
myA [10] = {8 , 33 , 65 , 66 , 89 , 15 , 7 , 85 , 98 , 10}

والمطلوب كتابة كود برمجي يتم من خلاله استخلاص القيمة الكبرى من عناصر المصفوفة وكذلك القيمة الصغرى وطباعتها على واجهة الاتصال التسلسلي UART.

```
byte myA[10] =  {8,33 ,65 ,66 ,89 ,15 ,7 ,85 ,98 ,10};
int max_v , min_v ;

void setup()
{
 Serial.begin(9600);
  min_v = myA[0];
  max_v = myA[0];
  for (int i = 0 ; i<10 ;i++)
  {
    max_v = max(max_v , myA[i]);
    min_v = min(min_v , myA[i]);
  }
Serial.print("Maximum Value = ");
Serial.println(max_v);
Serial.print("Minimum Value = ");
Serial.println(min_v);
}
void loop(){}
```

مثـــال 14

اكتب كود برمجي يتم من خلاله التحكم بإنارة LED متصل مع لوحة الآردوينو على القطب 3 عبر الحاسب من خلال النافذة التسلسلية UART.




```
char a;


void setup()
{
Serial.begin(9600);
pinMode(3 , OUTPUT);
}
```

```
void loop()
{
 if(Serial.available())
 {
  a = Serial.read();
 }
 if (a == '1')
 {
  digitalWrite(3 , 1);
  Serial.println("LED on");
 }
 else if(a == '0')
 {
  digitalWrite(3 , 0);
  Serial.println("LED off");
 }
 delay(200);
}
```

مثـال 15

ليكن لدينا القيم التالية:

A = 5 , B = 7 , C = 11 , D = 12 , E = 8

نريد تطبيق العمليات التالية (المخصصة للبت) على القيم السابقة وطباعة النواتج على نافذة الاتصال التسلسلي بالصيغة الثنائية:

E ^ A	**~ D**	**A & B**
D << 4	**(C & D) | E**	**A & C & D**

```
int const A = 5 , B = 7 , C = 11 , D = 12 , E = 8;
void setup()
{
 Serial.begin(9600);
}

void loop()
{
  Serial.println("A & B");
  Serial.print(A);
  Serial.print(" = ");
  Serial.println(A ,BIN);
  Serial.print(B);
  Serial.print(" = ");
  Serial.println(B ,BIN);
  Serial.println(A & B ,BIN);
  Serial.println("====================");
  delay(2000);
```

```
Serial.println("E ^ A");
Serial.print(E);
Serial.print(" = ");
Serial.println(E ,BIN);
Serial.print(A);
Serial.print(" = ");
Serial.println(A ,BIN);
Serial.println(E ^ A ,BIN);
Serial.println("====================");
delay(2000);

Serial.println("~D");
Serial.print(D);
Serial.print(" = ");
Serial.println(D ,BIN);
Serial.println(~D ,BIN);
Serial.println("====================");
delay(2000);

Serial.println("A & C & D");
Serial.print(A);
Serial.print(" = ");
Serial.println(A ,BIN);
Serial.print(C);
Serial.print(" = ");
Serial.println(C ,BIN);
Serial.print(D);
```

```
  Serial.print(" = ");
  Serial.println(D ,BIN);
  Serial.println(A & C & D ,BIN);
  Serial.println("====================");
  delay(2000);

  Serial.println("(C & D) | E");
  Serial.print(C);
  Serial.print(" = ");
  Serial.println(C ,BIN);
  Serial.print(D);
  Serial.print(" = ");
  Serial.println(D ,BIN);
  Serial.print(E);
  Serial.print(" = ");
  Serial.println(E ,BIN);
  Serial.println((C & D)| E ,BIN);
  Serial.println("====================");
  delay(2000);

  Serial.println("D << 4");
  Serial.print(D);
  Serial.print(" = ");
  Serial.println(D ,BIN);
  Serial.println(D << 4 ,BIN);
  Serial.println("====================");
  delay(2000);
}
```

مثـال 16

ليكن لدينا خمسة ليدات LEDs متصلة مع أقطاب لوحة الآردوينو ذات الأرقام (7, 6 , 5 , 4 , 3) والمطلوب تشغيل الليدات وفق الأنماط التالية:

(B01011 , B11010 , B00100 , B01010 , B11000 , B00001 , B11111)

في هذا المثال سوف نستفيد من تعليمات قراءة البت وتحديد حالته وتغيير حالته للحالة المطلوبة:

```
// Value on Output Pin
const byte outputValue[] = {
  B00001011 ,
  B00011010 ,
  B00000100 ,
  B00001010 ,
  B00011000 ,
  B00000001 ,
  B00011111
};

const byte pinOut[5] = {7,6,5,4,3};

byte value ;
boolean valueBit;

void setup()
{
  pinMode(3 ,OUTPUT);
  pinMode(4 ,OUTPUT);
  pinMode(5 ,OUTPUT);
```

```
  pinMode(6 ,OUTPUT);
  pinMode(7 ,OUTPUT);
  Serial.begin(9600);
}

void loop()
{
//first loop for get value
for(int i = 0 ; i < 7 ;i++)
  {
    value =outputValue[i];
    Serial.print(i);
    Serial.print(" = ");
    Serial.println(value ,BIN);
    for(int j = 0 ; j < 5 ; j++)
      {
        valueBit = bitRead(value,j);
        digitalWrite(pinOut[j],valueBit);
      }
    delay(3000);
  }
}
```

مثـال 17

ليكن لدينا مجموعتين من LEDs، المجموعة الأولى متصلة مع الأقطاب {3 , 8 , 12 , A0}، أما المجـــموعة الثانية فمتصـلة مع الأقطـــاب {2 , 7 , 13 , A1}، والمطلوب كتابة كود برمجي يكون خرج المجموعة الأولى القيم التالية: {B10001 , B00111 , B11001 , B01011} أما المجموعة الثانية فتكون نفس القيم السابقة مع عكس حالة البتين الأول والأخير.

```
byte valueOut[]={B01011 , B11001 , B00111 ,
B10001};
int group_1[]={3 , 8 , 12 , 14};
int group_2[]={2 , 7 , 13 , 15};
int value1 ;
boolean b1 ;

void setup()
{
  pinMode(3  ,OUTPUT);
  pinMode(8  ,OUTPUT);
  pinMode(12 ,OUTPUT);
  pinMode(14 ,OUTPUT);

  pinMode(2  ,OUTPUT);
  pinMode(7  ,OUTPUT);
  pinMode(13 ,OUTPUT);
  pinMode(15 ,OUTPUT);

  Serial.begin(9600);
```

```
}

void loop()
{
 for(int i=0 ; i < 4 ; i++)
 {
  value1 = valueOut[i];

     for(int j = 3 ; j >= 0 ; j--)
     {
      b1 = bitRead(value1 , j);
      digitalWrite(group_1[j],b1);
      if(j == 0 || j == 3)
      {
        b1 = !b1;
      }
      digitalWrite(group_2[j],b1);
     }
 }
}
```

مثـال 18

لدينا مجموعة من المفاتيح المتصلة مع الأقطاب {A0 , A3 , 3 , 9} وهذه الأقطاب معرفة كأقطاب دخل دون تفعيل مقاومة الرفع بل يتم توصيل مقاومة خفض للمفاتيح (وبالتالي فعند ضغط المفتاح تكون حالة الدخل 1 منطقي وعند عدم ضغط المفتاح تكون حالة الدخل 0 منطقي)، المطلوب قراءة حالة المفاتيح وتحويل القيمة الثنائية التي تمثلها هذه المداخل إلى قيمة عشرية وطباعة النتيجة على نافذة الاتصال التسلسلي UART.

```
int pinInput[4]={14 , 17 , 3 , 9};
int value_IN , bit_value;

void setup()
{
pinMode(A0 , INPUT);
pinMode(A3 , INPUT);
pinMode(3  , INPUT);
pinMode(9  , INPUT);
Serial.begin(9600);
}

void loop()
{
for(int i = 3 ; i>=0 ; i--)
  {
   bit_value = digitalRead(pinInput[i]);
   bitWrite(value_IN , i ,bit_value);
  }
```

```
  Serial.print(" Value in binary = ");
  Serial.print(value_IN , BIN);
  Serial.print(" ,  DEC = ");
  Serial.println(value_IN);
  Serial.println("============================");
  delay(200);
}
```

تمارين الفصل الأول

1

ليكن لدينا المصفوفة التالية: myArray [10] = {9 , -8, 52 , 87 , 12, 7 , -6}
المطلوب كتابة كود برمجي يقوم بإضافة القيمة 5 لكل عنصر من عناصر المجموعة لثلاث مرات وفي كل مرة نقوم بحساب متوسط عناصر المصفوفة وطباعة الناتج على النافذة التسلسلية وتشغيل زمور متصل مع أحد أقطاب الآردوينو لمدة نصف ثانية.

2

لنكتب كود برمجي لتشغيل ليد LED متصل مع القطب 13 من لوحة الآردوينو بحيث يتم تشغيل وإطفاء الليد من نفس الكباس اللحظي المتصل مع القطب 3 على ألا تزيد عدد مرات تشغيل الليد العشر مرات فقط، مع وجود كباس لحظي آخر متصل مع القطب 2 لتصفير عدد المرات التي تم ضغطها.

3

لدينا كباسين لحظيين متصلين مع القطبين A0 , A1 الأول لزيادة قيمة المتحول val بمقدار 2 والثاني لإنقاص قيمة المتحول val بمقدار 1، مع المحافظة على قيمة المتحول val ضمن المجال (8 , 3-)،

4

باستخدام التعليمة milis() اكتب كود برمجي لتشغيل LED متصل مع أحد أقطاب الآردوينو بحيث يعمل فقط بعد استمرار ضغط كباس لحظي Push Button متصل مع القطب 2 لمدة 3sec وعند ترك الكباس يتم إطفاء LED.

5

لدينا مقاومة متغيرة موصولة مع القطب التشابهي A0 من لوحة Arduino UNO كما يوجد لدينا 10 ليدات متصلة مع الأقطاب 2 ~ 11 والمطلوب: كتابة كود برمجي يقوم بتشغيل الليدات بالاعتماد على النسبة المئوية لقيمة المقاومة (أي كل 10 % من قيمة المقاومة يقابلها تشغيل ليد من الليدات بحيث يكون مجموع الليدات التي تعمل متناسب مع النسبة المئوية للمقاومة).

6

ليكن لدينا خمسة مقاومات متغيرة (تؤدي هذه المقاومات دور حساسات تشابهية) هذه الحساسات متصلة مع الأقطاب التشابهية A4 ~ A0 ، كما يوجد كباس لحظي متصل مع القطب 2 من أقطاب لوحة الآردوينو والمطلوب:

كتابة كود برمجي يحدد في كل مرة يتم فيها ضغط المفتاح من تحديد رقم الحساس الذي يعطي أكبر قراءة وطباعة رقم الحساس والقيمة التي يعطيها.

7

ليكن لدينا ثلاثة كباسات لحظية متصلة مع الأقطاب 4 , 3 , 2 كما يوجد لدينا ثلاثة LEDs متصلة مع الأقطاب A2 , A1 , A0 ، أيضا يوجد زمور Buzzer متصل مع القطب 13 ، والمطلوب وباستخدام التعليمة milis() كتابة كود برمجي لتشغيل الزمور لمدة 1sec وإطفائه بنفس المدة ، وكذلك وعند الضغط على أي كباس لحظي يتم تشغيل الليد المقابل له طالما الكباس مضغوط .

الفصل الثاني:
المحيطيات (I)

مـقـدمـــــة:

بعد أن استعرضنا في الفصل السابق في شرح مبسط وواضح بيئة التطوير **Arduino IDE** وكيفية محاكاة المشاريع في برنامج المحاكاة ورسم الدارات المطبوعة **Proteus 8** سننتقل لفصل جديد وممتع وهو قسم الطرفيات والمحيطيات التي يمكن وصلها مع لوحة الآردوينو ونخصص هذا الفصل للحديث عن جملة محددة من المحيطيات وهي:

- لوحة المفاتيح الست عشرية Keypad.
- شاشة القطع السبع بأنواعها 7 Segment.
- شاشة العرض الكرستالية المحرفية LCD.
- شاشة العرض الكرستالية الرسومية GLCD.
- ليد الألوان الثلاثة **RGB** LED

وهذا غيض من فيض من القطع التي يمكن وصلها مع الآردوينو وسنتطرق للكثير الكثير من القطع والمحيطيات والحساسات التي يمكن وصلها في الفصول القادمة.

ستكون آلية العمل في هذه الفصول هو طرح الطرفية أو المحيطية التي نريد التعامل معها وتقديم شرح مبسط وواضح لهذه الطرفية ثم ننتقل لبيئة الآردوينو لنرى كيف تتم برمجة هذه الطرفية وكيف يمكن تشغيلها والاستفادة منها في مشاريع لاحقة وسيكون هناك مشروع بسيط عن هذه الطرفية، وسيكون لنا في نهاية كل فصل جملة من المشاريع التي تربط الأفكار مع بعضها وتكون مشاريع مفيدة وقابلة للتطوير وهذه المشاريع ستكون جامعة لعدة أفكار.

إضافة مكتبة عمل لبيئة التطوير Arduino IDE:

قبل الشروع في الحديث عن الطرفيات الجديدة التي سندرسها في هذا الفصل سنستعرض أمر مهم جدا في بيئة التطوير Arduino IDE وهو المكتبيات الجاهزة وأهميتها عند الشـــروع في كتابة الكود البرمجي.

تعتمد بيئة التطوير Arduino IDE في غالب عملها على المكتبيات الجاهزة التي يقوم مبرمجون متخصصون بتصميمها وتطويرها ومن ثم رفعها إلى موقع البرنامج Arduino IDE , فهو كما ذكرنا سابقا لغة تطوير مفتوحة المصدر Open Source،وكونها لغة تطوير مفتوحة المصدر فإن الاعتماد على هذه المكتبيات يوفر علينا الكثير من الجهد والوقت كون هذه المكتبيات تكون مخصصة للموديولات المختلفة (كموديولات تحديد الموقع GPS وموديولات الارســـال والاســـتقبال عبر الأمواج الراديوية **RF** وموديولات **GSM** وغيرها الكثير) أو محيطيات والطرفيات (كالشاشات بأنواعها المختلفة والمحركات والحساسات المتنوعة) أو دارات تكاملية خاصـــة (كالذواكر الدائمة والمؤقتة ودارات الوقت الحقيقي ودارات التســـارع و الجيرسكوب) وغيرها الكثير الكثير من القطع.

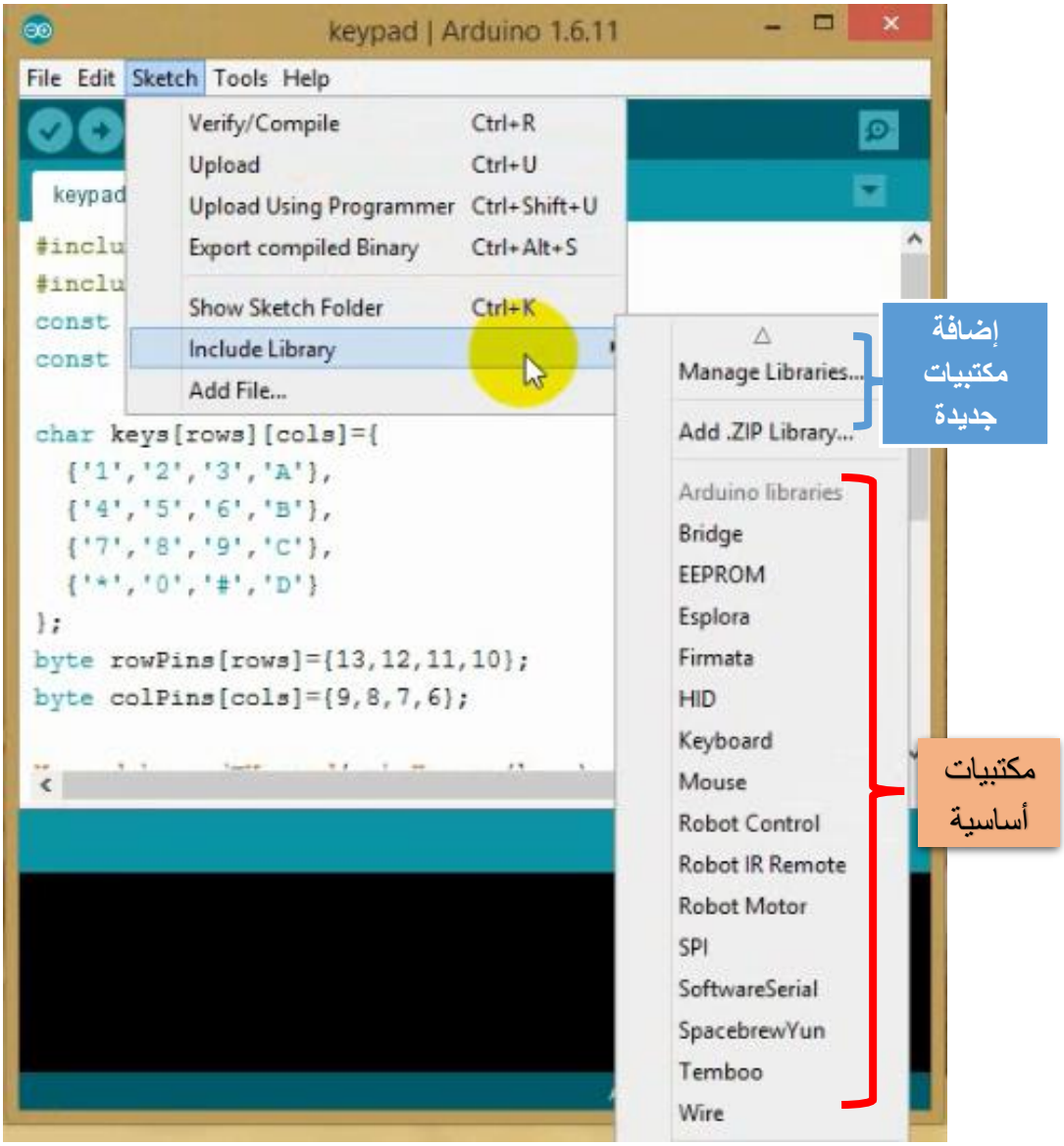
لكن كيف يتم إضافة المكتبية المطلوبة وكيف يتم الاستفادة منها في المشاريع وهل يمكن إضافة أكثر من مكتبة لنفس الكود البرمجي.....؟؟؟؟؟

إن ما يميز المكتبيات الجاهزة هو إمكانية إضافة العديد من المكاتب لنفس الكود البرمجي ولا يمكن التوقف عند عدد محدد من المكاتب الجاهزة بل وهناك حالات من الترابط والتكامل بين

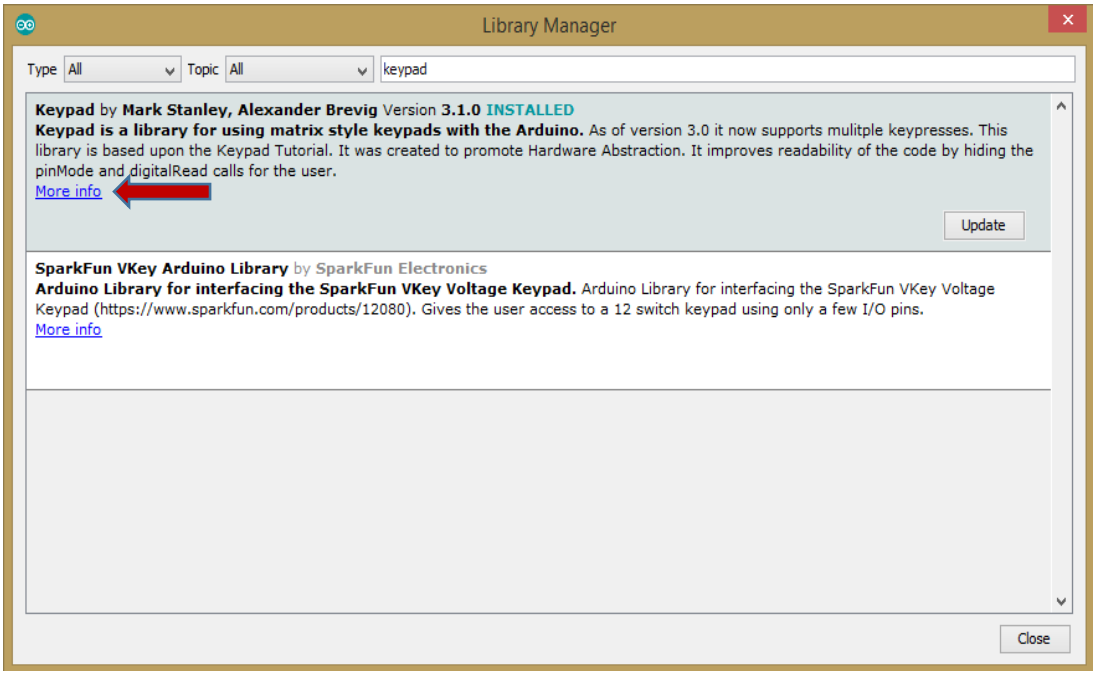
بعض المكتبيات للحصول على كود برمجي متكامل ومتطور، أما خطوات إضافة مكتبية لأي كود برمجي فهي وفق الخطوات التالية:

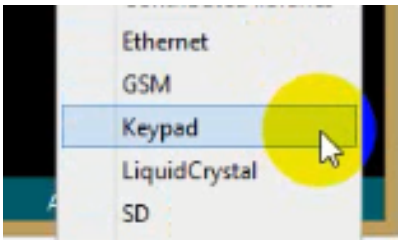
١- من القائمة sketch نختار إضافة مكتبة Include Library.

٢- تظهر قائمة تضم العديد من المكتبيات الموجودة أساسا في البرنامج، نختار منها المكتبة المطلوبة.



٣- أما إذا كانت المكتبية المطلوبة غير موجودة فممكن إضافتها عبر خيار إدارة المكتبيات Manage Library فتظهر لنا النافذة التالية:





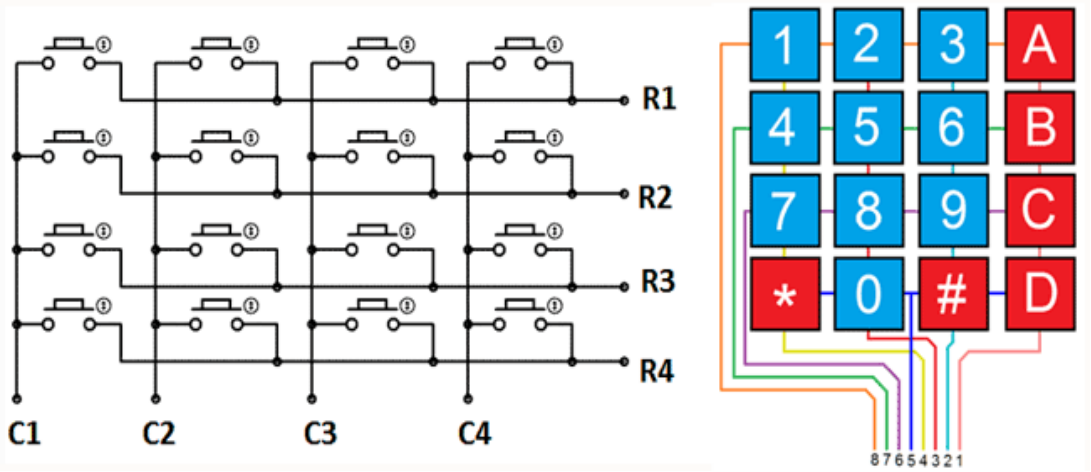
يتم من خلال هذه النافذة البحث عن المكتبية المطلوبة (مثلا في النافذة السابقة يتم البحث عن المكتبية الخاصة بلوحة المفاتيح الست عشرية keypad) وإضافتها لقائمة المكتبيات الموجودة أساسا في البرنامج، فيتم الضغط عليها لإضافتها للكود البرمجي، ولعرض المزيد من المعلومات حول المكتبية المطلوبة نختار الأمر More info فيتم عرض صفحة تحتوي على شرح عن المكتبة ومثال عن استخدامها.

٤- يتم إضافة المكتبيات أيضا التي تكون على شكل ملفات مضغوطة " zip " عبر الخيار Add ZIP Library فيتم استعراض الجهاز لتحديد مكان المكتبية ومن ثم إضافتها.

لوحة المفاتيح الست عشرية
Key Pad

تتألف لوحة المفاتيح الست عشرية من ستة عشر كباس لحظي , تكون هذه الكباسات متصلة مع بعضها كما في الشكل السابق (المفاتيح التي بجانب بعضها تكون أحد أطرافها متصلة مع بعضها مشكلة بذلك صف , أما العامود فيتشكل من المفاتيح التي فوق بعضها و بذلك تتشكل هذه اللوحة) و يكون لهذه اللوحة ثمانية أقطاب أربع أسطر و أربع أعمدة وصل عليها مقاومات قيمتها **R=470Ω** و لهذه المقاومات أهمية كبيرة لحماية المتحكم , يقوم المتحكم بعملية مسح للوحة المفاتيح الست عشرية و المفتاح المضغوط يتم معرفته من خلال معرفة نقطة تقاطع السطر مع العمود, الهدف من لوحة المفاتيح هو استخدام عدد كبير من الكباسات اللحظية بعدد من المداخل أقل وذلك لتوفيرها في استخدامات أخرى .

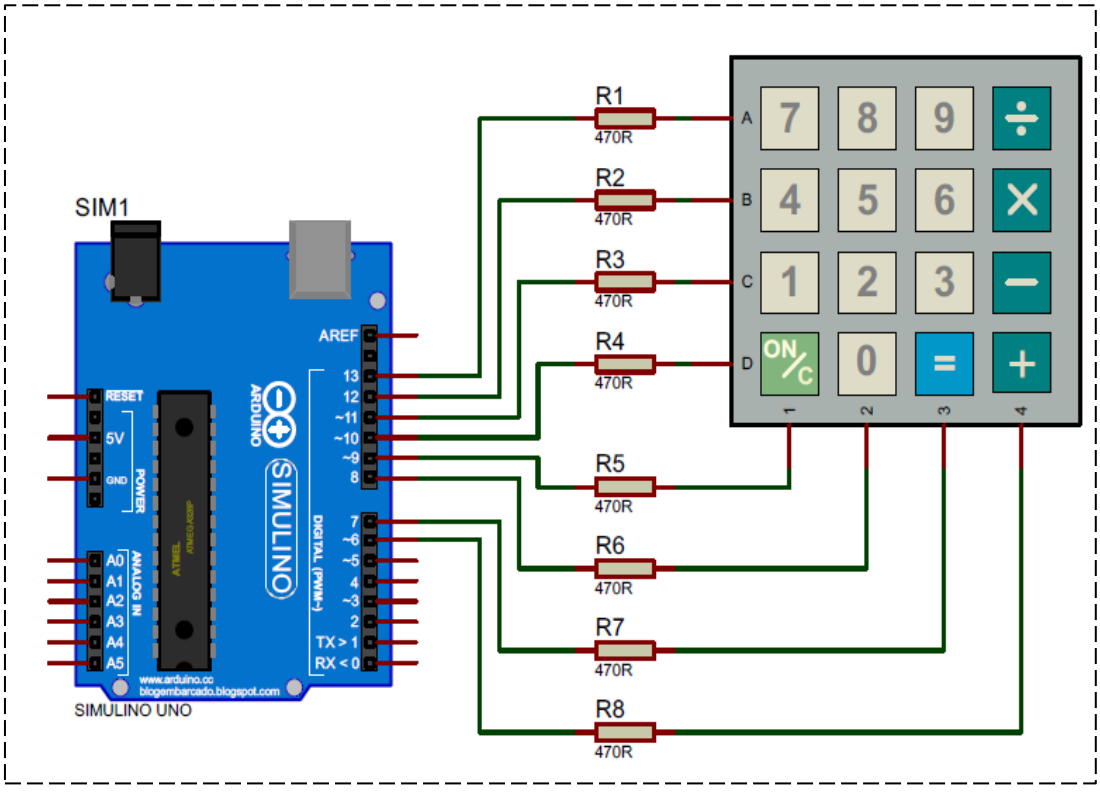
تتوفر لوحة المفاتيح والتي تعرف باسم **Keypad** بعدة أحجام وقياسات فقد تكون مؤلفة من أربع أسطر بـ ثلاثة أعمدة وقد تكون ثلاثة أسطر بـ ثلاثة أعمدة وهكذا لكن بالمجمل فإن طريقة البرمجة لا تختلف فالنتيجة واحدة.



التعليمات الخاصة بلوحة المفاتيح الست عشرية:

بالعودة لبيئة التطوير **Arduino IDE** نجد أن هناك تعليمات وتوابع خاصة بالتعامل مع لوحة المفاتيح الست عشرية وهذه التعليمات موجودة في مكتبية خاصة keypad، لكن قبل الدخول في التعليمات سنطرح بالمخطط التالي آلية توصيل لوحة المفاتيح الست عشرية مع لوحة الآردوينو نوع Arduino UNO:

المكتبية كما ذكرنا سابقا تكون عن جملة من التعليمات والتوابع المبنية بهدف تقديم آلية معينة للتعامل مع قطعة معينة أو أي شيء آخر، وهنا سنستعرض المكتبية الخاصة بتشغيل لوحة المفاتيح الست عشرية وكيف يتم الاستفادة من المكتبية في تبسيط التعامل مع القطعة المطلوبة.



شرح التعليمة	التعليمة
تعليمة تضمين المكتبة.	`#include <Keypad.h>`
تحضير تابع تعريف لوحة المفاتيح • تعريف أبعاد لوحة المفاتيح (عدد الأسطر والأعمدة) والأرقام الموجودة عليها (الأرقام والرموز والأحرف). • تحديد الأقطاب التي تم توصيل أسطر وأعمدة لوحة المفاتيح إلى لوحة الآردوينو. • في النهاية يتشكل لنا التابع FKeypad الذي يضم اسم لوحة المفاتيح keys توصيل الأسطر rowPin وتوصيل الأعمدة colPin وعدد الأسطر rows وعدد الأعمدة cols.	`char keys[rows][cols]={` `{'1','2','3','A'},` `{'4','5','6','B'},` `{'7','8','9','C'},` `{'*','0','#','D'}` `};` `byte rowPins[rows]={13,12,11,10};` `byte colPins[cols]={9,8,7,6};` `Keypad FKeypad = Keypad( makeKeymap(keys), rowPins, colPins, rows, cols);`
قراءة المفتاح المضغوط من لوحة المفاتيح المعرفة بالتابع FKeypad واسناد القيمة المضغوطة للمتحول الحرفي B_Key.	`char B_Key = FKeypad.getKey();`

مثـــال 1 :

لنكتب كود برمجي نستخدم فيه لوحة المفاتيح الست عشرية ونعرض القيم المضغوطة على نافذة الاتصال التسلسلي.

```
#include <Key.h>
#include <Keypad.h>
const byte rows=4;
const byte cols=4;

char keys[rows][cols]={
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[rows]={13,12,11,10};
byte colPins[cols]={9,8,7,6};
Keypad FKeypad = Keypad( makeKeymap(keys), rowPins,
colPins, rows, cols);


void setup(){
  Serial.begin(9600);
}
void loop(){
  char B_Key = FKeypad.getKey();
    if (B_Key){
    Serial.println(B_Key);
  }
}
```

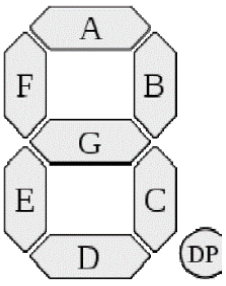
شاشة القطع السبع 7 Segment

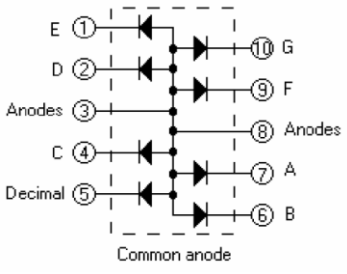
تعتبر شـــاشـــات القطع الســـبع إحدى أبرز الشـــاشـــات المســـتخدمة مع لوحات الآردوينو وذلك لرخص ثمنها بالنسـبة لشـاشـات أخرى لكن يعتبر التحكم بهـا أصـــعب مقـارنـة مع شاشات الـ LCD كما أنها تقتصر على إظهار أشـــيـاء معينـة كـالأرقـام وبعض الأحرف الإنكليزية وعدد من الرموز فقط.

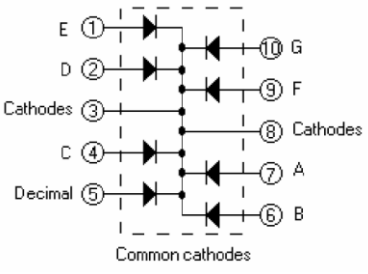
بما أن البنية الأساسية لشاشات القطع السبع هي الليدات لذلك نجد نوعين لهذه الشاشات هما:

- **مهبط مشترك:** تكون **مهابط** الليدات موصولة مع بعضها ويتم التحكم عبر **المصاعد**.
- **مصعد مشترك:** تكون **مصاعد** الليدات موصولة مع بعضها ويتم التحكم عبر **المهابط**.

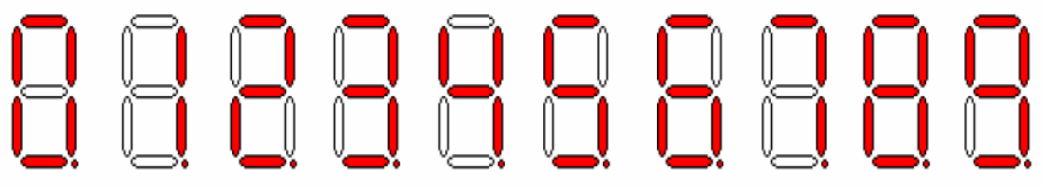
ومعرفة نوعية الشـاشـة المسـتخدمة شـيء ضـروري وذلك لكتابة الكود البرمجي، فعندما تكون الشاشة مهبط مشترك لا تعمل الشاشة إلا إذا تم تغذية القطب المشترك بالصفر منطقي وإعطاء الواحد منطقي للقطعة المطلوب تشـــغيلها، أما إذا كانت مصـــعد مشـــترك فيجب تغذية القطب المشترك بالواحد منطقي حتى تعمل الشاشة وإعطاء الصفر المنطقي للقطعة المراد تشغيلها ويتم الكشف عن ذلك عبر مقياس الفولط واختبار الشاشة أو بالعودة للوثيقة الفنية للشاشة عبر الرقم المطبوع عليها.







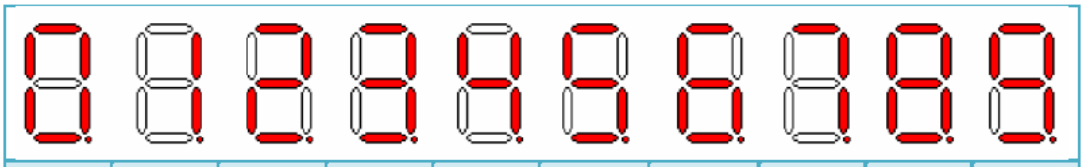
لدينا شاشة القطع السبع **7Segmant** ذات المهبط المشترك ونريد أن نعرض عليها الأرقام التالية:



نلاحظ أنه عند إظهار العدد المطلوب نقوم **بتشغيل** الليدات المطلوبة، و**إطفاء** الليدات الغير مطلوبة وفقا للجدول التالي:

الرقم المطلوب إظهاره	القيمة على البوابة PortC								القطع التي هي في حالة عمل:
	0-A	1-B	2-C	3-D	4-E	5-F	6-G	7-H	
0	1	1	1	1	1	1	0	0	A.B.C.D.E.F
1	0	1	1	0	0	0	0	0	B.C
2	1	1	0	1	1	0	1	0	A.B.D.E.G
3	1	1	1	1	0	0	1	0	A.B.C.D.G
4	0	1	1	0	0	1	1	0	B.C.F.G
5	1	0	1	1	0	1	1	0	A.C.D.F.G
6	1	0	1	1	1	1	1	0	A.C.D.E.F.G
7	1	1	1	0	0	0	0	0	A.B.C
8	1	1	1	1	1	1	1	0	A.B.C.D.E.F.G
9	1	1	1	1	0	1	1	0	A.B.C.D.F.G

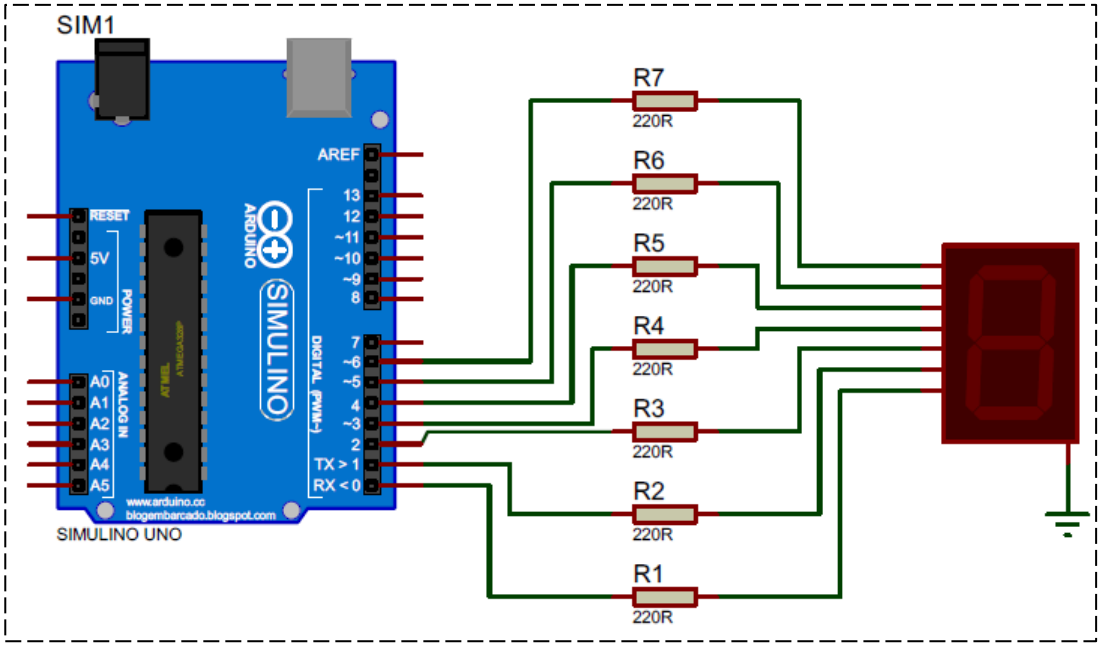
في حال كانت الشاشة من نوع الموجب المشترك فإننا نأخذ المتمم للجدول السابق ومن ثم نحول القيم للصيغة الست عشرية أو الثنائية.



&H3F	&H06	&H5B	&H4F	&H66	&H6D	&H7D	&H07	&H7F	&H6F

تشغيل شاشة القطع السبع مع لوحة الآردوينو:

لا يوجد تعليمات خاصة في بيئة التطوير **Arduino IDE** بشاشة القطع السبع لكن سنكتب مثال بسيط عن تشغيل شاشة القطع السبع ذات المهبط المشترك بخانة واحدة فقط متصلة مع لوحة الآردوينو **Arduino UNO** كما في الشكل التالي وسنوضح كيف يتم عرض الأرقام عليها أما الشاشة التي تتألف من أكثر من خانة فهناك العديد من الطرق لتشغيلها أفضلها استخدام الدارات التكاملية الخاصة بتشغيل الشاشات منها الدارة التكاملية IC: MAX7219 والتي تعمل وفق البروتوكول SPI التسلسلي أو غير ذلك من الطرق التي سنتطرق لها لاحقا.



الشكل السابق يوضح كيف يتم توصيل شاشة ذات المهبط المشترك مع لوحة **الآردوينو UNO** وفي هذه الحالة تقوم لوحة الآردوينو بتقديم التغذية الموجبة لكل قطعة مطلوب تشغيلها أي أن لوحة الآردوينو تعمل دور المنبع، وإذا أردنا استخدام لوحة من نوع موجب مشترك فعندئذ ستأدي لوحة الآردوينو دور المصرف للتيار الكهربائي.

مثـــال 2:

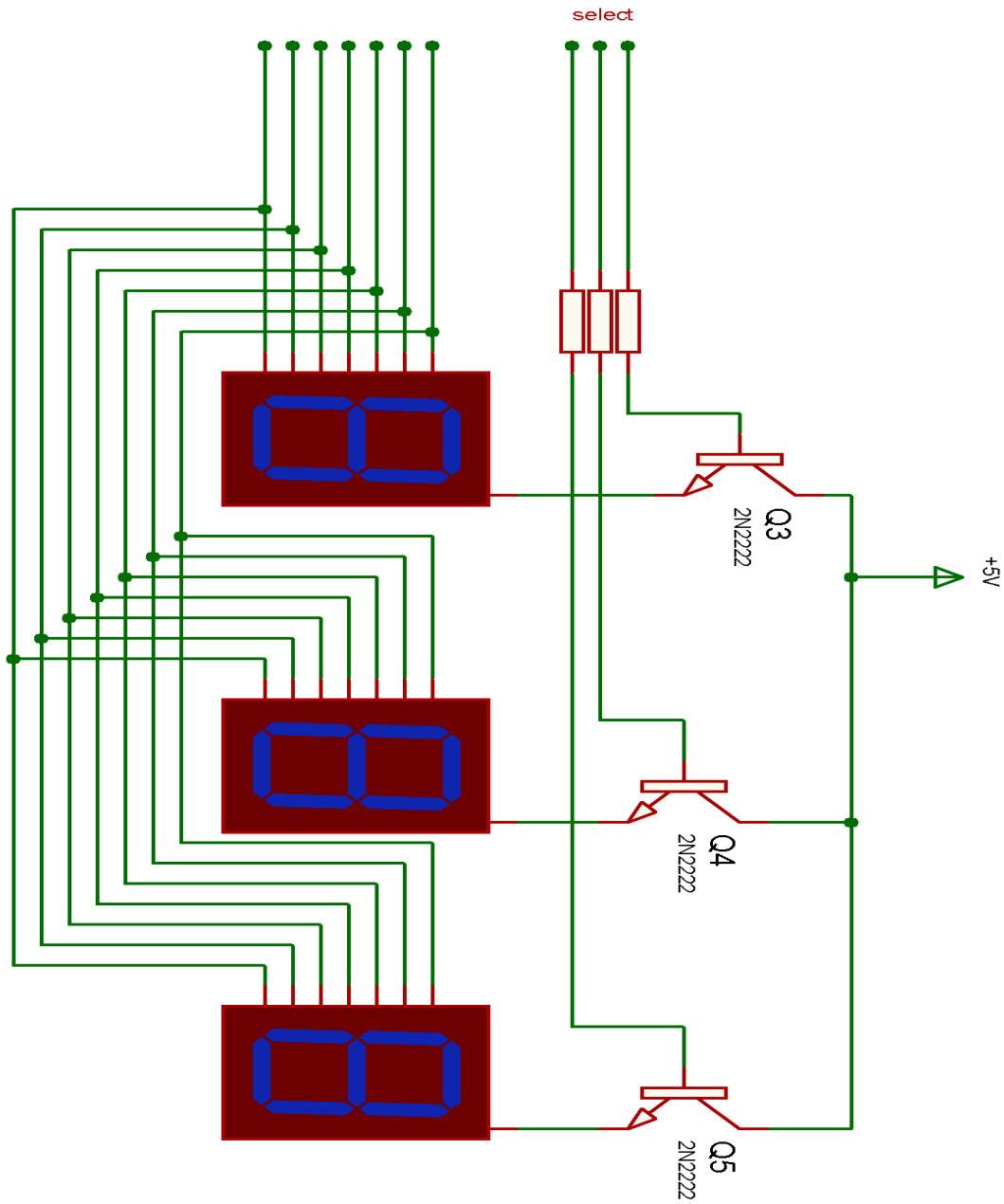
لنكتب الآن كود برمجي لتشغيل شاشة القطع السبع 7Segment للعد التدريجي من الصفر وحتى التسعة.

```
byte seg_val[10]={
  //ABCDEFG/dp
  B11111100, // 0
  B01100000, // 1
  B11011010, // 2
  B11110010, // 3
  B01100110, // 4
  B10110110, // 5
  B00111110, // 6
  B11100000, // 7
  B11111110, // 8
  B11100110, // 9
};
//             dp,G,F,E,D,C,B,A
int segPIN[8]={2,3,4,5,6,7,8,9};
boolean bit_val ;

void setup()
{

pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
```

```
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);

}
void loop()
{
for(int i = 0; i < 10 ; i++)
  {
     for(int j=0 ; j<8 ; j++)
     {
      bit_val = bitRead(seg_val[i],j);
      digitalWrite(segPIN[i] , bit_val);
     }
  }
delay(1000);
}
```

فكرة الكود تقوم على عمل حلقتي for متداخلتين، حيث تقوم الأولى باستدعاء العنصر المطلوب عرضه على الشاشة من مصفوفة القيم `seg_val[i]` ، بينما تقوم المصفوفة الثانية بتوزيع القيم على الليدات فتقوم بتشغيل الليد المطلوب وتشغيل الليدات الغير مطلوبة وهكذا حتى يتم عرض باقي الأرقام.

من خلال الكود السابق تعلمنا كيف نتعامل مع شاشة 7 segment أحادية الخانة بنوعيها موجب مشترك أو سالب مشترك، لكن السؤال كيف يتم التعامل مع شاشة من نفس النوع لكن بعدة خانات وما هي بنيتها الداخلية وكيف يتم استخدامها....؟؟؟

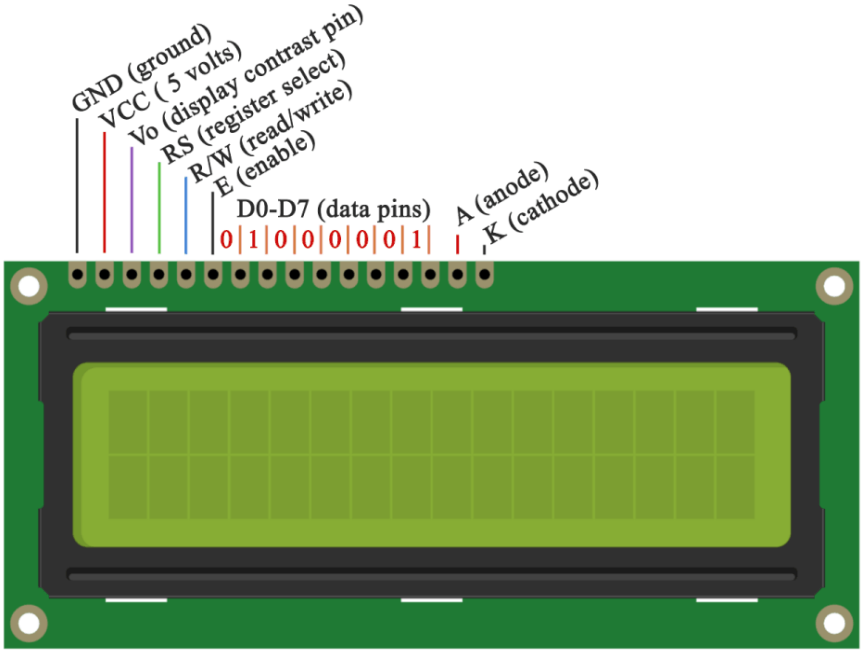


أولا ولكي يتم التعامل معها بشكل صحيح يجب علينا فهم البنية الداخلية للشاشات التي تأتي بأكثر من خانة، حيث يتم وصل الأقطاب الخاصة بكل قطعة من كل شاشة مع بعضها البعض(أي كل القطع A من الشاشات يكون لها نفس القطب وهكذا الأمر بالنسبة لبقية الأحرف وهذا التوصيل يكون داخلي، بينما الشكل السابق هو للفهم وتبسيط الفكرة)، ويكون هناك القطب المشترك الخاص بكل شاشة (فإن كانت الشاشة من نوع موجب مشترك يكون هناك قطب تفعيل لكل خانة من خانات الشاشة وكذلك الأمر في حالة السالب المشترك)، فمثلا لتشغيل الحرف C من الخانة الثانية (بفرض الشاشة ذات مصعد مشترك) يتم إعطاء **GND** من لوحة الآردوينو للقطب الخاص بالقطع **C** بينما يتم وصل **الموجب المشترك** بالخانة المطلوبة مع التغذية المناسبة للشاشة فيعمل الليد للخانة المطلوبة دون بقية الخانات، إلى الآن الأمور سهلة لكن لو أردنا تشغيل كل الخانات مع بعضا البعض في نفس الوقت فكيف سنقوم بذلك ؟؟؟

بالطبع لا يمكن تشغيل كل الخانات في نفس الوقت، لكن سوف يتم الاعتماد على الخداع البصري للعين البشرية، فالعين البشرية غير قادرة على ملاحظة التغير للترددات فوق ال **50Hz** لذلك سوف يتم تشغيل الخانات بشكل متتالي وبفواصل زمنية **20ms** وبالتالي لن تلاحظ العين حالة انتقال العرض بين الخانات، كيف يمكننا تحقيق هذا الأمرستكون الإجابة عبر مشروع بسيط عبارة عن مؤقت زمني بأربع خانات في نهاية الفصل الثاني بإذن الله تعالى.

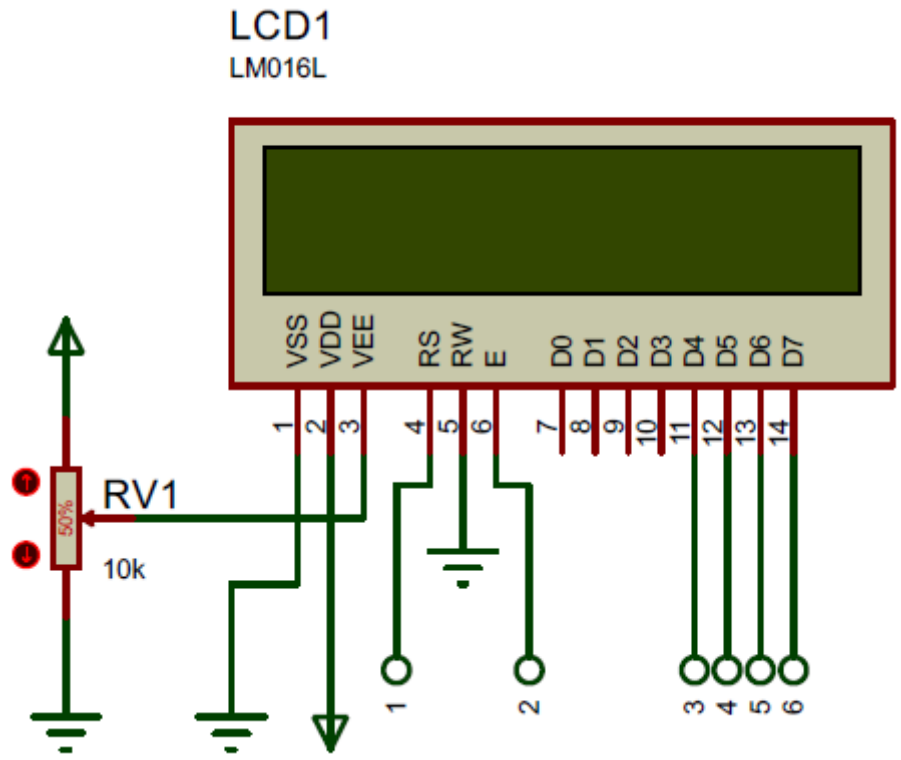
شاشة العرض الكرستالية المحرفية LCD

شاشة العرض البلوري السائل أو بالاختصار **LCD** Liquid Cristal Display، تعتبر شاشة العرض الكرستالية من أهم وسائل عرض البيانات المرسلة إليها من لوحة الآردوينو وتتميز بسهولة التعامل وانخفاض التكلفة المادية وإمكانية عرض البيانات وبعض الرموز والأشكال، كما تحوي بداخلها معالج خاص لتسهيل التعامل معها حيث يكون هو المسؤول عن العمليات المعقدة للإظهار على الشاشة كما تحتوي على ذاكرة لحفظ المحارف المطلوب عرضها على الشاشة , تتوفر الشاشة بعدة قياسات و أحجام و تقاس عادة هذه الشاشة بعدد الأسطر و الأعمدة التي تحتويها الشاشة، لكن و مهما اختلفت أبعاد الشاشة يبقى لها نفس الأقطاب الخارجية للتحكم :



- أقطاب البيانات **D0 ~ D7**.
- قطب التغذية **VDD = 5v**.
- قطب الأرضي **Vss = GND**.
- قطب التحكم بتباين الشاشة **Vee** .
- قطب التفعيل **E**.
- قطب الضبط **RS**.
- **A** & **K** موجب وسالب ليد إضاءة الشاشة.

وعند توصيل شاشة العرض الكرستالية مع لوحة الآردوينو يجب توصيلها كما في الشكل التالي:



يتم من خلال المقاومة المتغيرة التحكم بنصوعية الشاشة وتكون قيمتها **10kR**، كما يجب وضع مكثفين عدسيين قيمة كل منهما **100nF** بحيث يكونان قريبين جدا من أقطاب التغذية لحماية الشاشة من الضجيج الكهربائي أما الأقطاب المرقمة من **1 ~ 6** (RS , E , D4 , D5 ,D6,D7) فيتم توصيلها مع لوحة الآردوينو بالترتيب الذي نختاره , كما يجب توصيل القطب RW مع الأرضي وبالتالي تكون الشاشة جاهزة للعمل والكتابة عليها.

تشغيل شاشة LCD مع لوحة الآردوينو:

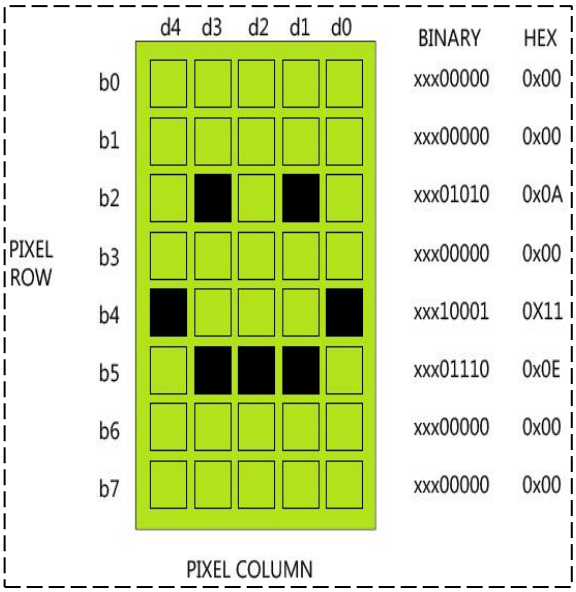
تضمن بيئة التطوير Arduino IDE مكتبية خاصة بتشغيل شاشة العرض الكرستالية LCD نجدها في قسم المكتبيات بالاسم LiquidCrystal ولهذه المكتبية عدة بارامترات يجب ضبطها قبل الشروع في العمل على الشاشة كما توفر هذه المكتبية تعليمات العرض المختلفة ومسح البيانات والتحكم بموقع المؤشر على الشاشة وإمكانية إنشاء محرف جديد كل هذا سنجده من خلال أدوات هذه المكتبية.

شرح التعليمة	التعليمة
تضمين مكتبية تشغيل شاشة العرض الكرستالية للكود.	`#include <LiquidCrystal.h>`
تحديد الأقطاب التي تم توصيل الشاشة إليها على لوحة الآردوينو مع المحافظة على هذا الترتيب للأقطاب.	`LiquidCrystal lcd(RS,E,D4,D5,D6,D7);`
تحديد أبعاد الشاشة في قسم التهيئة، وذلك بتحديد عدد الأعمدة cols والأسطر rows.	`lcd.bgin(cols,rows);`

lcd.print( );	**طباعة البيانات على الشاشة، متحولات وعبارات.**
lcd.write( );	**طباعة العبارات فقط على الشاشة.**
lcd.setCursor(col,row);	**وضع مؤشر الكتابة في الموضع المحدد بالسطر والعمود على اعتبار أن السطر الأول والعامود الأول يحملان الرقم 0.**
lcd.clear( );	**مسح الشاشة.**
lcd.blink( );	**جعل المؤشر في حالة خفقان.**
lcd.noBlink( );	**إيقاف خفقان مؤشر الكتابة.**
lcd.cursor( );	**تشغيل المؤشر.**
lcd.display( );	**تشغيل الشاشة.**
lcd.noDisplay( );	**إطفاء الشاشة.**
lcd.rightToLeft( );	**تحديد جهة الكتابة من اليمين إلى اليسار.**
lcd.leftToRight( );	**تحديد جهة الكتابة من اليسار إلى اليمين.**
lcd.autoscroll( );	**تمرير النص خانة نحو اليمين.**
lcd.write(Serial.read());	**طباعة البيانات التي تصل من النافذة التسلسلية على شاشة العرض الكرستالية.**

بعد عرض جملة التعليمات السابقة نكون قد وصلنا لدرجة كافية من معرفة عمل الشاشة وتشغيلها والاستفادة الكاملة من كل خصائصها وتسخيرها لاحقا في مشاريعنا، وسندعم هذه التعليمات بجملة من الأمثلة التي توضح آلية عمل هذه التعليمات.

تصميم محرف بشكل خاص:



في حالات معينة نريد تصميم محرف معين خاص بنا وعرضه على الشاشة بحيث يكون لهذا المحرف وظيفة معينة، وبالعودة لبيئة التطوير Arduino IDE نجد أنها تدعم تصميم محرف خاص، لكن يجب ألا يتجاوز عدد المحارف الإضافية 7 محارف، وقبل الخوض في آلية تصميم المحرف علينا معرفة أنا أبعاد كل محرف في شاشة العرض الكرستالية هو ثمانية أسطر بخمسة أعمدة كما يبينه الشكل، **أما خطوات تصميم محرف جديد فهي:**

- لدينا المحرف الجديد والذي اسمه **name** نحدد النقاط التي سوف تعمل من المحرف من التي لن تعمل عبر التعليمة:

```
byte name[8] = { 0b00000,0b00000,0b01010 ,0b00000
,0b10001,0b01110,0b00000,0b00000 };
```

- نعطي للمحرف الجديد رقم خاص به بحيث يكون الرقم أحد الأرقام من 0 ~ 7 بحيث لا نكرره مع غيره من المحارف المصممة ويتم ذلك بالتعليمة:

```
lcd.createChar(n , name);
```

- نستدعي المحرف المطلوب بالتعليمة:

```
lcd.write(n);
```

مثـــال 3 :

لنكتب كود برمجي نستعرض من خلاله جميع التعليمات الخاصة بشاشة العرض الكرستالية LCD.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(13,12,11,10,9,8);
int x;
byte smile[8]={
  0b00000,
  0b00000,
  0b01010,
  0b00000,
  0b10001,
  0b01110,
  0b00000
};


void setup() {
 lcd.begin(16,2);
lcd.createChar(1,smile);
}

void loop() {
  x=0;
 lcd.noCursor();
 lcd.setCursor(0,0);
 lcd.write("Arduino");
```

```
delay(2000);
lcd.blink();

for(x=0;x<=10;x++){
   lcd.clear();
   lcd.setCursor(x,0);
   lcd.print(x);
   delay(500);
   lcd.clear();
}
lcd.cursor();
lcd.write("My Love");
lcd.setCursor(10,0);
lcd.write(1);
delay(2000);
lcd.clear();
}
```

في بداية الكود البرمجي قمنا بتعريف مكتبية شاشة العرض الكرستالية LCD ومن ثم قمنا بإنشاء محرف جديد يحمل الاسم smile وحددنا النقاط التي سوف تعمل من المحرف من التي لن تعمل، بعد ذلك وفي حلقة الإعداد void setup قمنا بتحديد أبعاد الشاشة التي يتم التعامل معها وتعريف المحرف الجديد بمنحه رقم خاص به.

في حلقة البرنامج الرئيسية قمنا بضبط مكان توضع مؤشر الكتابة وتحديد القيمة البدائية للمحول x والتي سوف يبدأ منها بعد كل عملية بدأ جديدة للحلقة اللانهائية، ثم يتم طباعة العبارة

Arduino والانتظار لمدة ثانيتين وبعد ذلك نغير من خصائص مؤشر الكتابة فنجعله في حالة خفقان , بعد ذلك تبدأ حلقة العد **for** والتي سوف تطبع الأرقام من 10 ~ 0 بشكل متتالي وبفاصل نصف ثانية وكل رقم في خانة جديدة .

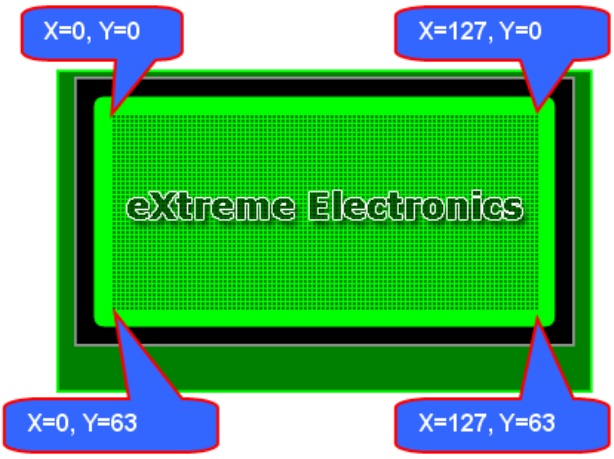
بعد ذلك نطبع العبارة **My love** ومن ثم ننقل المؤشر لموقع الكتابة في السطر الأول العامود العاشر لطباعة المحرف الجديد ومن ثم ننتظر لمدة ثانيتين، وبعد ذلك يتم مسح الشاشة والعودة والبدء من جديد.

يتوفر في السوق نوع جديد من شاشة العرض الكرستالية يكون على الشاشة قطعة يتم من خلالها مخاطبة الشاشة عبر البروتوكول I2C (والذي سوف نتحدث عنه لاحقا في فصل بروتوكولات الاتصال) والذي يختزل عدد الأقطاب الواجب وصلها مع لوحة الآردوينو من 6 أقطاب في الطريقة التقليدية إلى قطبين فقط (SDL & SCL وهما أقطاب البروتوكول I2C) وهذا التوصيل الجديد أعطى للشاشة ميزة جديدة وهي تقليل عدد أقطاب التوصيل.

شاشة العرض الكرستالية الرسومية GLCD

تعتبر شاشة الـ **Graphic LCD** نسخة مطورة عن شاشة العرض المحرفية الكرستالية البلورية، وتستخدم لعرض الرسوم والمخططات البيانية بالأبيض والأسود فقط، وبخلفية للشاشة مختلفة حسب الشركة المصنعة بين الأخضر و الأزرق و الأحمر وغيرها من الألوان

بالإضافة لعرض الأحرف بعدة خطوط وأحجام، الاسم التجاري لهذه الشاشة **GDM LCD 128*64**، وتختلف هذه الشاشات عن بعضها من حيث شريحة التحكم بكل منها، سنستخدم في دراستنا الشاشة التي شريحة القيادة الخاصة بها تحمل الرقم **KS0108** , كما يتوفر منها عدة قياسات حيث يقاس حجمها بعدد البيكسل الموجود في السطر الواحد والعامود الواحد وسنستخدم الشاشة التي أبعادها **128*64**.

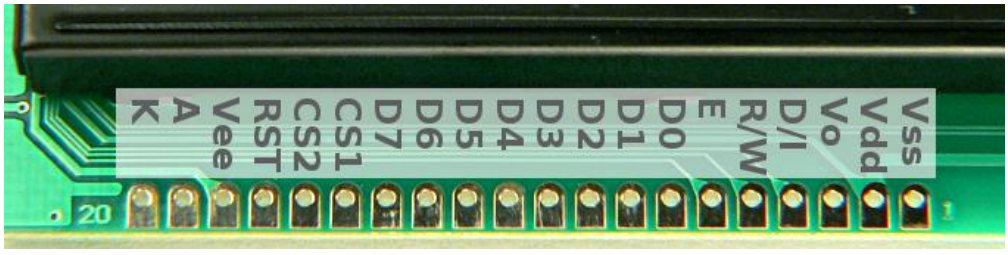


الشكل المجاور يبين إحداثيات الشاشة والنقطة الصفرية لها وهذا مهم لمعرفة توجيه مؤشر العرض على الشاشة.

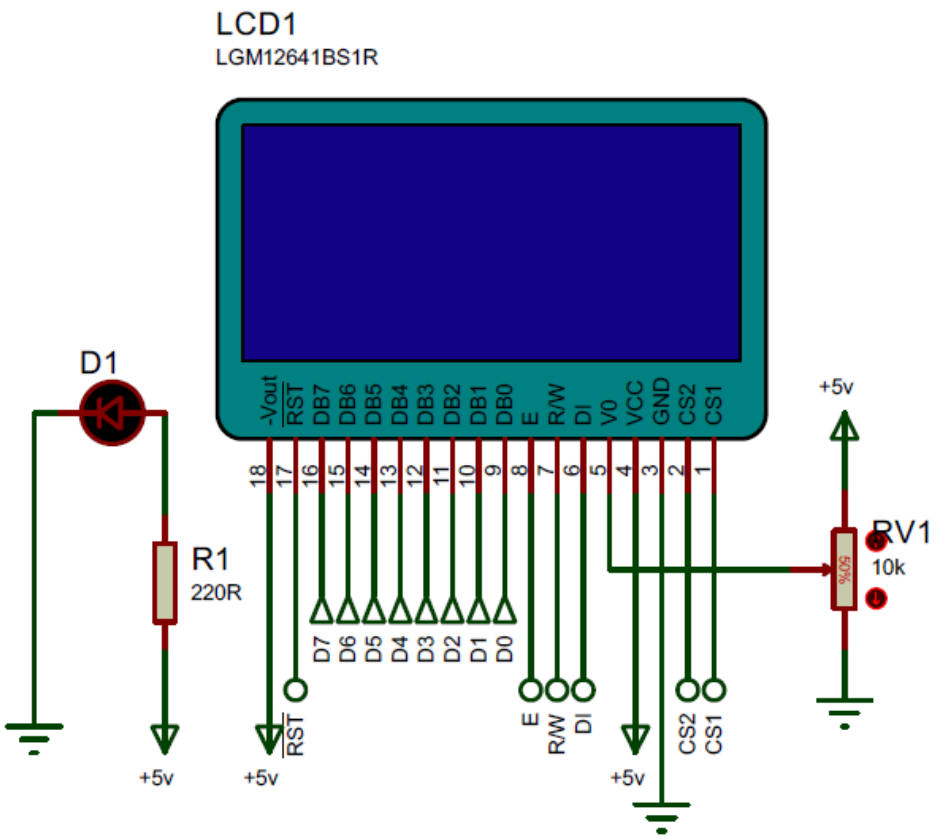
أقطاب شاشة العرض الكرستالية المحرفية GLCD:

عند شراء شاشة العرض الرسومية يجب العودة للوثيقة الفنية الخاصة بالشاشة وذلك بسبب وجود عدة أنواع من هذه الشاشة بنفس الأقطاب لكن بتوزع مختلف لها.

رقم القطب	القطب من الشاشة	شرح القطب
1	Vss	قطب التغذية السالبة (الأرضي GND).
2	Vdd	قطب التغذية الموجبة volt 5+.
3	Vo	جهد قيادة الشاشة.
4	D/I	قطب بيانات الدخل/الخرج لمسجل الإزاحة الداخلي.
5	R/W	قطب الاختيار بين القراءة والكتابة.
6	E	قطب التفعيل.
7 ~ 14	D0 ~ D7	أقطاب البيانات.
15	CS1	اختيار الشريحة 1.
16	CS2	اختيار الشريحة 2.
17	RST	قطب تصفير الاتصال.
18	Vee	خرج تغذية السالبة.
19	A	القطب الموجب لليد الإضاءة الخلفية.
20	K	القطب السالب لليد الإضاءة الخلفية.



طريقة توصيل شاشة العرض الرسومية مع الآردوينو:



- المقاومة المتغيرة والتي قيمتها **10k** وظيفتها التحكم بشدة إضاءة الشاشة.
- من المهم وضع مقاومة ثابتة قيمتها **220R** على التسلسل مع تغذية الليد المسؤول عن الإضاءة الخلفية للشاشة وذلك لحمايته.
- يجب ألا تتجاوز تغذية الشاشة **5.5 volt** حتى تعمل بشكل جيد.

نصل الآن لقضية مهمة جدا وهي كيف يتم توصيل أقطاب شاشة العرض الرسومية مع لوحة الآردوينو، تقدم المكتبية الخاصة بتشغيل شاشة العرض الكرستالية الرسومية توصيل ثابت للشاشة مع لوحة الآردوينو وهذا التوصيل لا يمكن تغييره وإلا فإن الشاشة لن تعمل، لذلك

سنستعرض في الجدول التالي مكان توضع أقطاب الشاشة مع لوحة آردوينو من نوع **UNO** ومن نوع **MEGA**:

قطب شاشة GLCD	لوحة UNO	لوحة MEGA
D0	**8**	**22**
D1	**9**	**23**
D2	**10**	**24**
D3	**11**	**25**
D4	**4**	**26**
D5	**5**	**27**
D6	**6**	**28**
D7	**7**	**29**
CS1	**14(A0)**	**33**
CS2	**15(A1)**	**34**
RESET	**RESET**	**RESET**
R/W	**16(A2)**	**35**
DI	**17(A3)**	**36**
EN	**18(A4)**	**37**

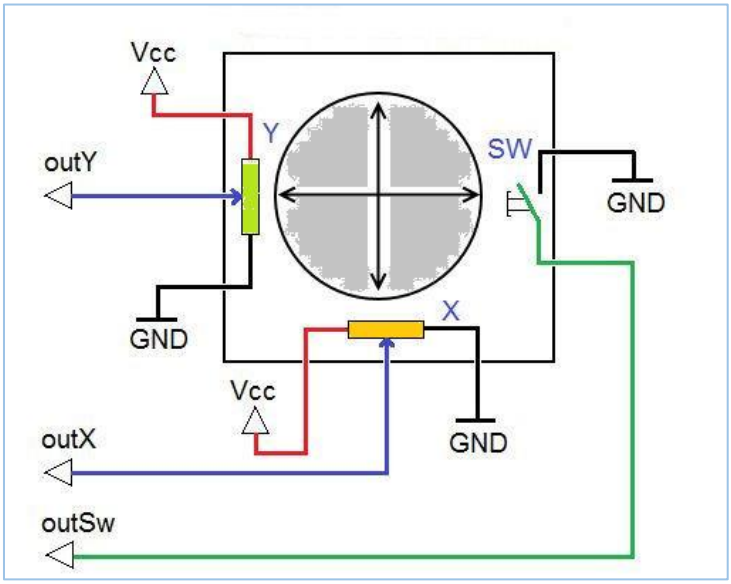
وهكذا نكون قد استعرضنا آلية التوصيل الصحيحة بين شاشة العرض الرسومية ولوحة الآردوينو وبهذا نكون قد انتهينا من قسم التوصيلات ووصلنا لبحث آلية تشغيل الشاشة في بيئة التطوير Arduino IDE.

للتعامل مع شاشة العرض الرسومية الكرستالية GLCD يجب علينا تحميل وإضافة المكتبية الخاصة بالشاشة ويتم الحصول على المكتبية المطلوبة من خلال الموقع التالي:

http://code.google.com/p/glcd-arduino/downloads/list

القبضة التشابهية ذات المحورين X & Y

وهي عبارة عن مقاومتين متغيرتين متوضعتين بشكل متصالب يتم توصيل كل واحدة منهما على شكل مقسم للجهد (للمقاومة المتغيرة ثلاثة أقطاب أحد هذه الأقطاب يوصل مع التغذية والآخر مع الأرضي أما المتغير منهما مع أحد أقطاب التشابهية من لوحة الآردوينو) بحيث تعبر إحداهما عن المحور X بينما تعبر الثانية عن المحور Y، كما تحتوي كباس لحظي يضغط نحو الأسفل من القبضة، الشكل التالي يبين البنية الداخلية للقبضة التشابهية:



كيف نستفيد من هذه القبضة التشابهية .... ؟؟؟

لهذه القبضة الكثير من التطبيقات وخاصة عند التعامل مع المحركات، فيتم استخدام هذه القبضة لتحديد جهة دوران المحركات وكذلك سرعتها وغير ذلك من التطبيقات الكثيرة، أما عن استخدامها في بيئة Arduino IDE فيمكن التعامل معها على أنها مقاومة متغيرة وبالتالي نصلها بأي مدخل تشابهي ونقوم بقراءة قيمتها، أو إضافة مكتبية خاصة لهذه القبضة والتي تسهل التعامل مع القبضة التشابهية.

هذا الكود البسيط سنطبع من خلاله قيمة المقاومة على المحورين X & Y :

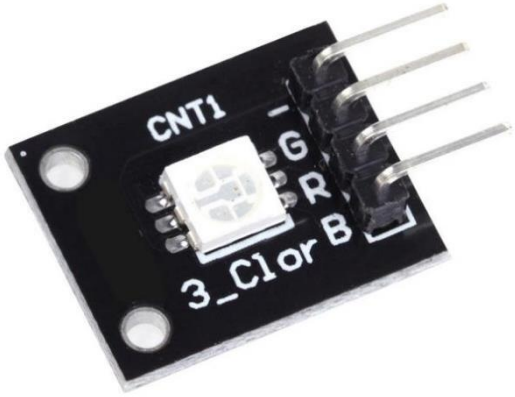
```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int x_axis = analogRead(A0);
  int y_axis = analogRead(A1);

  Serial.print(" X axis = ");
  Serial.print(x_axis , 4);

  Serial.print(" || ");
```

```
  Serial.print(" Y axis = ");
  Serial.println(y_axis , 4);


  delay(100);
}
```

الكود كما نلاحظ وهو عبارة عن قراءة القيم التشابهية على الأقطاب A1 & A0 وعرض القيم على واجهة الاتصال التسلسلي UART، أما الرقم 4 مع تعليمة الطباعة الخاصة بطباعة القيم الخاصة بالمحورين Y & X فهي لتحديد عدد الخانات المراد طباعتهابشكل دائم.

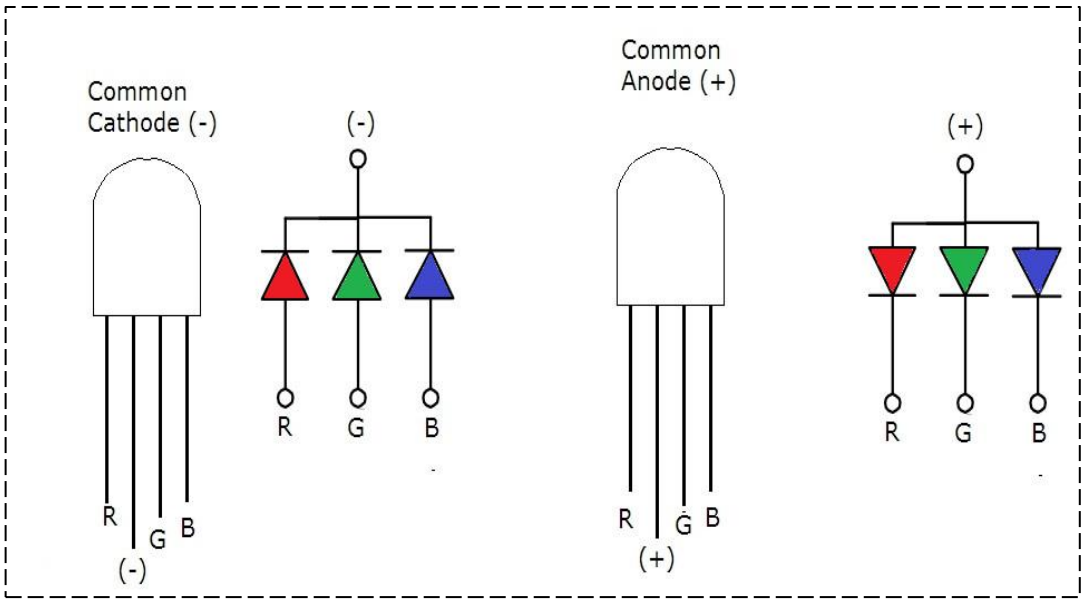
في الفصول القادمة سوف يكون هناك العديد من التطبيقات التي سنستخدم فيها هذه القبضة.

ليد الألوان الثلاثة RGB LED

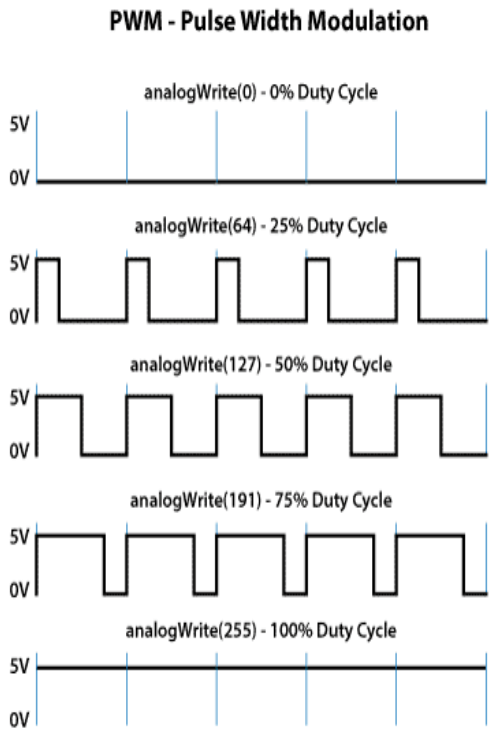


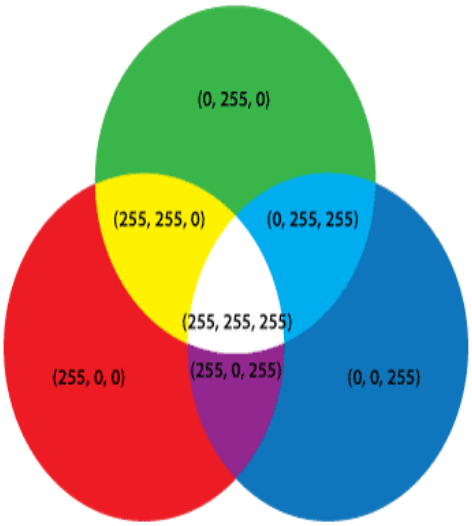
ليد الألوان الثلاثة أو ما يعرف بالاختصار **RGB** LED: **R**ed **G**reen **B**lue LED وهو عبارة عن مجموع ثلاثة ليدات في ليد واحد، ألوان هذه الليدات هي الأحمر و الأخضر و الأزرق ويكون لهذه الليدات نوعين حسب طريقة تجميع الليدات، فقد يتم تجميع مهابط (**Cathodes**) الليدات مع بعضها والتحكم بالليدات يتم عبر المصعد (**Anode**) الخاص بكل ليد أو يتم تجميع المصاعد(**Anodes**) مع بعضها والتحكم بالليدات يتم عبر المهبط (**Cathode**) الخاص بكل ليد، الشكل التالي يبين طريقة تجميع هذه الليدات:

ماهي الفائدة التي توفرها ليدات RGB...؟؟؟؟



إن طريقة بناء هذه الليدات يوفر العديد من الميزات ففي حالات معينة قد نضطر لاستخدام عدة ليدات بعدة ألوان فتقوم هذه الليدات بتوفير مساحة على البورد مع تقديم نفس الوظيفة المطلوبة، وليس هذا فقط بل يمكن توليد العديد من الألوان وذلك بتشغيل ليدين معا فنحصل على ألوان إضافية أو الحصول على اللون الأبيض عند تشغيل الليدات الثلاثة، كما يمكن الحصول على ألوان أخرى متعددة من خلال استخدام خاصية التحكم بعرض النبضة أو ما يعرف بالاختصار **PWM**: **P**ulse **W**idth **M**odulation حيث يتم توليد نبضات بنسب متفاوتة على أقطاب الليدات والذي ينتج عنه ألوان بتدرجات مختلفة تبعا لعرض النبضة المطبقة.



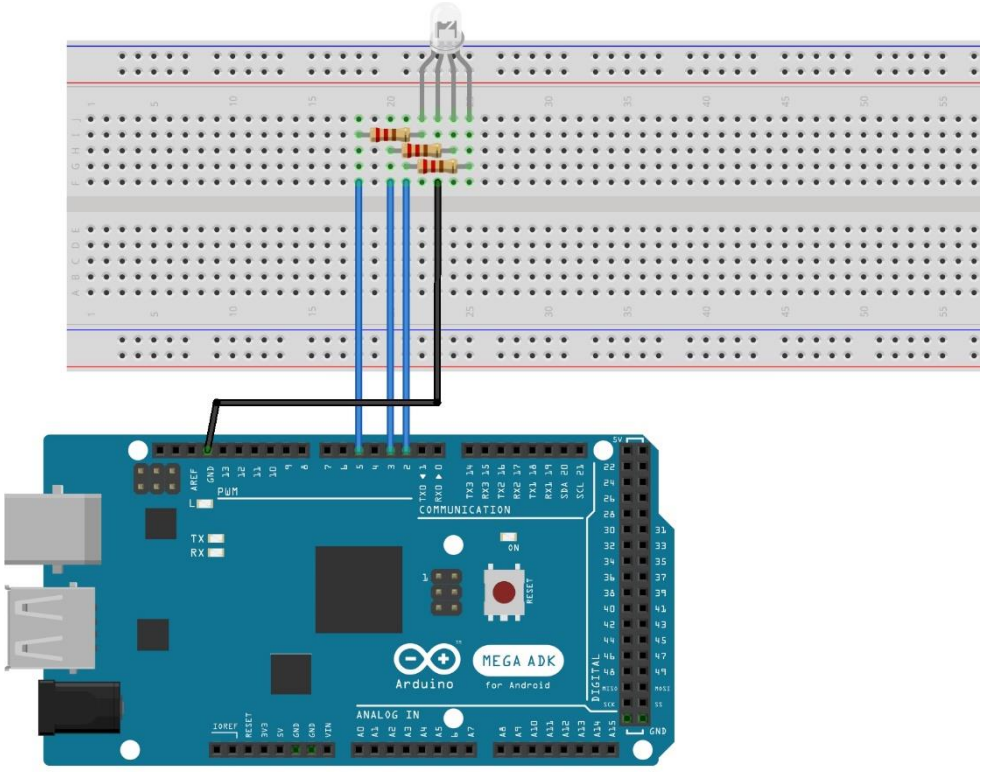


لتشغيل الليد لا ننسى توصيل **مقاومة حماية** على أقطاب الليدات تكون قيمتها **220R** وذلك عند توصيلها بشكل مباشر مع لوحات الآردوينو.

يقصد بتغيير عرض النبضة التحكم بنسبة القيمة الفعالة من النبضة مقارنة بنسبة القيمة الغير فعالة منها للحصول على قيمة جهد وسطية تعبر عن النسب المختارة وسنتطرق لمفهوم عرض النبضة بالتفصيل لاحقا.

مثـــال 4 :

ليكن لدينا RGB LED متصل مع الأقطاب 7 , 6 , 5 من لوحة آردوينو نوع MEGA، لنكتب كود برمجي يتحكم بعرض النبضة للتحكم بالألوان على الليد.



نقوم بتوصيل أرجل الليد مع الأقطاب التي تعطي على خرجها نبضة يمكن التحكم بعرضها، وفي لوحة الآردوينو **MEGA** تكون الأقطاب التي تحمل الأرقام من **2** وحتى **13** كلها أقطاب يمكن أن تعطي على خرجها نبضة متحكم بعرضها **PWM**، نقوم بوصل أقطاب الألوان بحيث يكون **الأحمر** مع **5** و**الأزرق** مع القطب **3** و **الأخضر** مع القطب **2** ، ونضع **مقاومات حماية** على التسلسل مع كل قطب من أقطاب الليد قيمة كل مقاومة **220R** .

```
    int redPin   = 7;
    int greenPin = 6;
    int bluePin  = 5;

    void setup() {
      pinMode(redPin   , OUTPUT);
      pinMode(greenPin, OUTPUT);
      pinMode(bluePin , OUTPUT);
    }
    void loop() {
      setColor(255, 0, 0); // Red Color
      delay(1000);
      setColor(0, 255, 0); // Green Color
      delay(1000);
      setColor(0, 0, 255); // Blue Color
      delay(1000);
      setColor(255, 255, 255); // White Color
      delay(1000);
      setColor(170, 0, 255); // Purple Color
      delay(1000);
    }
    void setColor(int redValue,int greenValue,int
blueValue) {
      analogWrite(redPin   , redValue  );
      analogWrite(greenPin, greenValue);
      analogWrite(bluePin , blueValue );
    }
```

لغة C++ من جديد

كنا قد درسنا في الفصل الأول المتحولات وأنواعها المختلفة لكن هناك الكثير من العمليات والتحويلات البرمجية التي لابد من الاطلاع عليها لكي يكون للمبرمج أريحية كبيرة في تعديل الكود البرمجي والتعامل مع كل الحالات التي ستواجهه أثناء عملية تحضير الكود البرمجي، لذلك سنتطرق هنا لبعد العمليات مع شرح بسيط عنها.

التحويلات Conversion:

تستخدم هذه التعليمات لتحويل نوع المتحول المطلوب لنوع جديد بما يتناسب مع الاحتياجات البرمجية.

الشرح	التعليمة
تحويل المتحول x إلى النوع **byte**	`byte(x)`
تحويل المتحول x إلى النوع المحرفي **char**	`char(x)`
تحويل المتحول x إلى النوع العشري **float**	`float(x)`
تحويل المتحول x إلى النوع الصحيح **int**	`int(x)`
تحويل المتحول x إلى النوع الصحيح **long**	`long(x)`
تحويل المتحول x إلى النوع الصحيح **word**	`word(x)`

النوع القديم للمتحول x غير مهم في عمليات التحويل السابقة، فلن تواجهنا أي مشكلة عند التحويل بين الأنماط الرقمية المختلفة، أما وعند التحويل من قيمة محرفية لقيمة رقمية فسوف يكون الناتج هو رقم يدل على قيمة ASCII للقيمة المحرفية التي تم تحويلها للقيمة الرقمية وهنا نحن بحاجة لتعليمات تحويل جديدة.

تعليمات خاصة بالتحويل المحرفي للرقمي وبالعكس:

شرح التعليمة	التعليمة
لتحويل السلسلة لرقم صحيح بشرط أن تكون قيم السلسلة أرقام حصرا، أما ناتج التحويل فهو من نوع long. أما إذا كانت السلسة تبدأ بأرقام وتنتهي بأحرف فإن هذه التعليمة تعيد فقط الأرقام التي في بداية السلسلة.	stringName.toInt() //exp String val_in = "125"; long v; v = val_in.toInt(); v = v + 205; Serial.print(v); // 330
للحصول على محتوى العنصر n من السلسة stringName، علما أن n يبدأ من القيمة 0، أما الفراغ space فيأخذ بعين الاعتبار أيضا.	stringName.charAt(n) //exp String val_in = "12ML"; Serial.println(val_in.charAt(2)); // M
لإضافة عنصر ما للسلسلة stringName ولا يشترط نوع محدد لهذا العنصر إذ أنه يتم تحويله لنوع محرفي وإضافته لنهاية السلسلة.	stringName.concat(parameter); //exp String val_in = "12ML"; char a = 'A'; int x = 12; val_in.concat(a); val_in.concat(x); Serial.print(val_in); //12LMA12
للمقارنة بين قيمة سلسلتين، والناتج يكون منطقي أي محقق أو غير محقق.	String1.equals(string2)
للبحث عن القيمة val (والتي قد تكون محرف أو سلسلة) ضمن السلسلة stringName بحيث يتم البحث من	stringName.indexOf(val,from) //exp String val_in = "12ML" ; String val = "2M";

نقطة محددة from أو من أول السلسلة في حال عدم تحديد نقطة بدء البحث، ويعيد رقم يدل على مكان القيمة val ضمن السلسلة أو القيمة 1- في حال عدم وجود هذا العنصر ضمن السلسلة.	``` int index ; index = val_in.indexOf(val); Serial.print(index); //1 //exp2 String val_in = "12ML" ; String val = "M2"; int index ; index = val_in.indexOf(val); Serial.print(index); // -1```
للحصول على طول السلسلة (أي عدد عناصر السلسلة) كما يتضمن الفراغات الموجودة ضمن السلسلة.	```stringName.length() //exp String val_in = " 12 M L"; int n ; n = val_in.length(); Serial.print(n); // 7```
لحذف قيم معينة من السلسلة بدء من الموقع index وبعدد محدد من العناصر count وفي حال عدم تحديد قيمة count يتم الحذف حتى نهاية السلسلة.	```stringName.remove(index, count) //exp String val = "12358MLsfr113"; Serial.println(val); val.remove(5,4); Serial.println(val); //12358r113```
تستخدم التعليمة replace من أجل استبدال قسم محدد substring1 من السلسلة الأساسية stringN بسلسلة أخرى substring2.	```stringN.replace(substring1, substring2) //exp String val = "12358ML"; Serial.println(val); //12358ML val.replace("358M" , "MS"); Serial.println(val); // 12MSL```

التعريفات والنطاق للمتحولات Variable Scope & Qaulifiers:

وهي عبارة عن تعليمات تكتب قبل تعريف نوع المتحول وتستخدم لتحدد طبيعة المتحول في الكود البرمجي أهم هذه العمليات:

الشرح	التعليمة
تستخدم التعليمة const لإعطاء قيمة ثابتة للمتحول بحيث لا يمكن تغييرها لاحقا داخل الكود، أي يصبح هذا المتحول للقراءة فقط.	`const type name = value;` `const float pi = 3.14;`
تستخدم التعليمة static لجعل المتحول خاص بتابع معين وبالتالي لايمكن استدعائه أو إجراء أي عملية عليه خارج التابع الموجود فيه.	`static type name;` `static int place;`
تستخدم التعليمة volatile في حالات خاصة مع المتحولات التي تحتاج دقة عالية في النتيجة حيث يقوم المترجم بتخزين المتحول في الذاكرة RAM مباشرة وليس في الذاكرة Flash وذلك للحصول على أكبر سرعة في عملية التغير على المتحول.	`volatile type name;` `volatile byte state;`

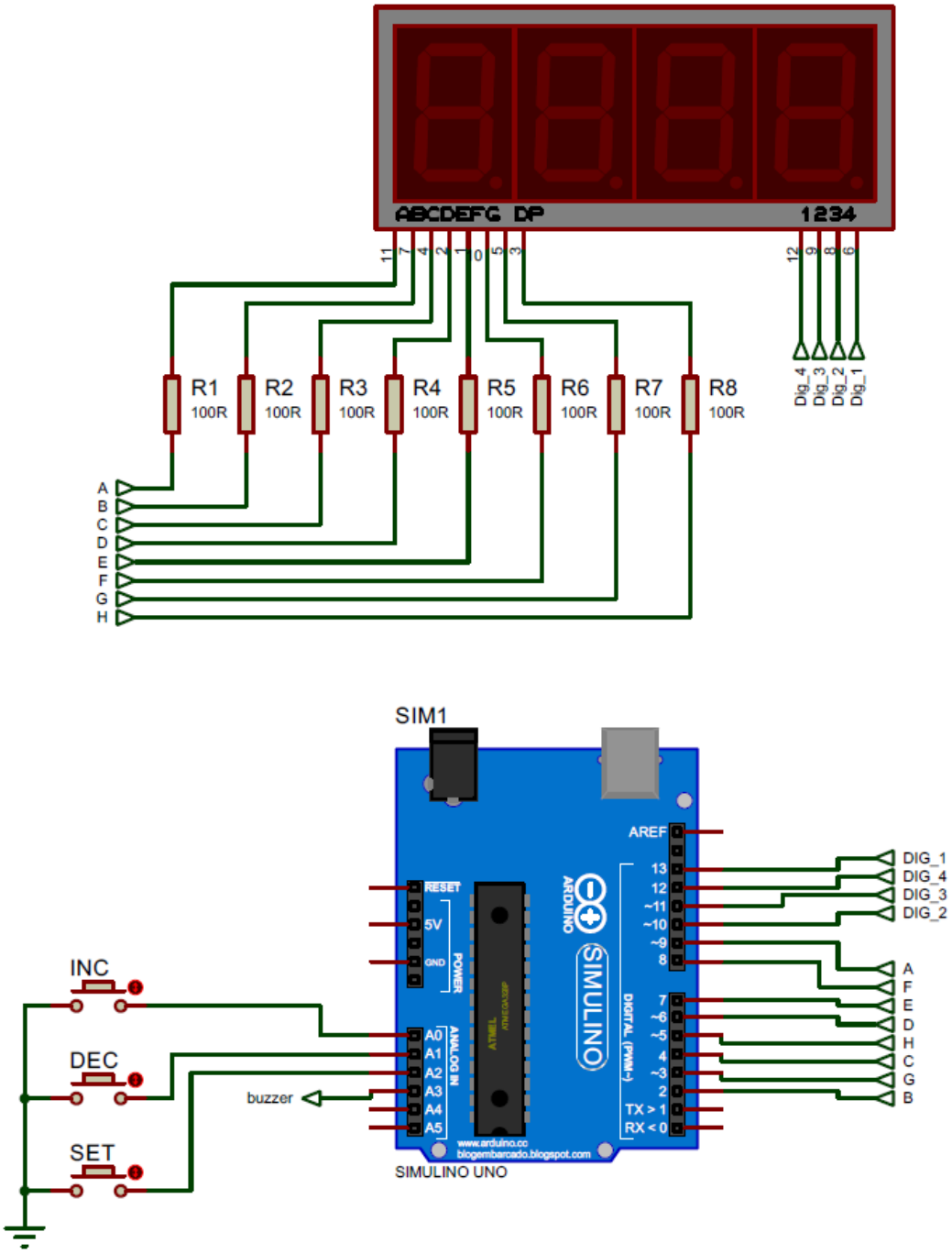
مشروع الفصل: مؤقت تنازلي بأربع خانات باستخدام شاشة 7segment

تقوم فكرة المشروع على بناء مؤقت تنازلي يبدأ العد عند الضغط على مفتاح SET / START وعند الوصول لنهاية العد يقوم بتشغيل زمور لمدة محددة ثم يتوقف، الهدف من هذا المشروع هو تعلم كيفية التعامل مع شاشات القطع السبع ذات الخانات المتعددة وكيفية العرض عليها، فالتعامل مع هذه الشاشات يتطلب بالدرجة الأولى فهم البنية الداخلية للشاشات المراد تشغيلها، ثم فهم كيفية ربطها مع المتحكم للحصول على أفضل أداء.

الشكل التالي يبين كيفية توصيل الشاشة مع الآردوينو (يمكن فصل الأقطاب لقسمين الأول تحكم والذي من خلاله نحدد الشاشة التي تعمل من الشاشة التي لا تعمل والثاني للعرض والذي من خلاله نحدد الرقم الذي نريد عرضه، أما ترتيب الأقطاب فليس مشكلة لأن أمر العرض والتحكم يتم برمجيا وبالتالي لا داعي لأن تكون التوصيلات متسلسة)، كما يوجد ثلاثة كباسات لحظية للتحكم حيث سيكون أحدها للضبط ولبدء التشغيل بينما الكباسين المتبقيين فلزيادة القيمة أو إنقاصها عند الضبط.

سنعرض على أول خانتين من الشاشة (الأقل أهمية) الثواني بينما سيتم عرض الدقائق على الخانتين المتبقيتين وبالتالي سيكون أقل وقت هو دقيقة واحدة وأكبر وقت هو 99 دقيقة.

سنقوم بوصل زمور صغير يعطي صوت عند كل ثانية وعند إنتهاء الوقت يعطي صوت لمدة مستمرة محددة ثم يطفئ ويتوقف العمل.


ABCDEFG DP
1234
11 7 4 2 1 10 5 3
12 9 8 6
Dig_4
Dig_3
Dig_2
Dig_1
R1 100R
R2 100R
R3 100R
R4 100R
R5 100R
R6 100R
R7 100R
R8 100R
A
B
C
D
E
F
G
H
SIM1
AREF
RESET
5V
GND
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
DIGITAL (PWM~)
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX > 1
RX < 0
ARDUINO
SIMULINO
ATMEL
www.arduino.cc
blogembarcado.blogspot.com
SIMULINO UNO
DIG_1
DIG_4
DIG_3
DIG_2
A
F
E
D
H
C
G
B
INC
DEC
SET
buzzer

الـكـود الـبـرمـجي:

كما تعودنا عند كتابة أي كود برمجي نقوم في بداية الكود بالتصريح عن الأقطاب التي سوف نستخدمها وأسمائها المستعارة التي سوف نتعامل بها خلال الكود وهذا الأمر مريح جدا جدا عند كتابة الكود البرمجي ويعطي أريحية كبيرة في التعديل. نلاحظ عدم وجود أي مكتبية في الكود البرمجي لأننا سنكتب كود برمجي بالاعتماد على أنفسنا فالهدف هو زيادة المهارات ورفع المستوى دون الاعتماد الكلي على المكتبيات الجاهزة.	`#define A_display 9` `#define B_display 2` `#define C_display 4` `#define D_display 6` `#define E_display 7` `#define F_display 8` `#define G_display 3` `#define H_display 5` `#define sec_dig_1 10` `#define sec_dig_2 13` `#define mom_dig_3 12` `#define mom_dig_4 11` `#define incr_pin A0` `#define decr_pin A1` `#define set_pin A2` `#define buzzer_pin A3` `#define OUT1_pin A4` `#define OUT2_pin A5`
نعرف بعد ذلك **ثلاثة مصفوفات**: الأولى: مخصصة لأقطاب العرض للشاشات وهي	`byte display_pin[8] = {A_display , B_display , C_display , D_display , E_display , F_display , G_display , H_display};` `byte seg_val[10]={` `//HGFEDCBA/dp`

الأقطاب a , b , c , d ,e ,f ,g ,h. الثانية والثالثة: القيم المراد عرضها على الشاشات [0,1 ,2,3 , 4 , 5 , 6 , 7 ,8 ,9] مع فارق بينها حيث أن المصفوفة الأولى تعرض الأرقام بدون تشغيل الفاصلة العشرية بينما الثانية فتشغل الأرقام مع الفاصلة العشرية.	0b00111111, // 0 0b00000110, // 1 0b01011011, // 2 0b01001111, // 3 0b01100110, // 4 0b01101101, // 5 0b01111101, // 6 0b00000111, // 7 0b01111111, // 8 0b01101111, // 9 }; byte seg_val_point[10]={ //HGFEDCBA/dp 0b10111111, // 0 0b10000110, // 1 0b11011011, // 2 0b11001111, // 3 0b11100110, // 4 0b11101101, // 5 0b11111101, // 6 0b10000111, // 7 0b11111111, // 8 0b11101111, // 9 };
	bool bit_val ; long duration,duration_old; int second = 60 , moment = 1 , moment_dis; int set_timer ; byte digit_1_val ,// second _1

digit_2_val ,// second _2 digit_3_val ,// moment _1 digit_4_val ;// moment _2 byte flage_set; byte second_old, second_now ; byte x ;	
void setup() { flage_set = 1 ; Serial.begin(9600); pinMode(A_display , OUTPUT); pinMode(B_display , OUTPUT); pinMode(C_display , OUTPUT); pinMode(D_display , OUTPUT); pinMode(E_display , OUTPUT); pinMode(F_display , OUTPUT); pinMode(G_display , OUTPUT); pinMode(H_display, OUTPUT); pinMode(sec_dig_1, OUTPUT); pinMode(sec_dig_2, OUTPUT); pinMode(mom_dig_3, OUTPUT); pinMode(mom_dig_4, OUTPUT); pinMode(buzzer_pin, OUTPUT); pinMode(incr_pin,INPUT_PULLUP); pinMode(decr_pin,INPUT_PULLUP); pinMode(set_pin ,INPUT_PULLUP);	في دالة التهيئة يتم فيها تهيئة أقطاب الدخل والخرج وتهيئة نافذة الاتصال التلسلي. كما يتم تشغيل الزمور لمدة نصف ثانية. أيضا يتم إسناد قيمة بدائية للمتحول duration_old وهي قراءة قيمة المؤقت millis.

tone(buzzer_pin , 200 ,500); set_time(); duration_old = millis(); }	
void loop() { set_time(); timer_work(); display_work(); digit_1_val = second%10; digit_2_val = second/10; digit_3_val = moment_dis%10; digit_4_val = moment_dis/10; }	الحلقة الدورانية اللانهائية تضم عدة توابع لكل منها وظيفة خاصة سنشرحها لاحقا. كما يتم تقسيم خانات الثواني والدقائق وتوزيعها على متحولات خاصة من أجل عرضها لاحقا على الشاشات.
void timer_work() { duration = millis(); if(duration - duration_old > 1000) // 1000 = 1sec { duration_old = millis(); second--; second_old = second ; moment_dis = moment - 1; tone(buzzer_pin , 700 ,300); //tone(pin,frequency,uration); //Serial.print(moment_dis);	التوابع الأول وظيفته إنقاص قيمة الثواني بدأ من القيمة 59 وحتى الصفر وعند كل وصول للصفر يتم إنقاص قيمة الدقائق أيضا بمقدار واحد. عند الوصول للقيمة صفر لكل من الدقائق والثواني يدخل المعالج في حلقة لانهائية يعمل ضمنها الزمور فقط.

//Serial.print(":"); //Serial.println(second); if(second == 0) { second = 60 ; moment--; if(moment == 0) { while (moment == 0) { moment = 0 ; second = 0 ; tone(buzzer_pin , 1000 ,100); //tone(pin,frequency,duration); } } } } }	
void set_time() { if(digitalRead(set_pin) == 0) { delay(10); x++; if(x == 100)	تابع ضبط الوقت يتم من خلاله ضبط قيمة الدقائق المطلوبة عبر كباسات الزيادة والإنقاص، مع حصر قيمة الدقائق ضمن المجال [1 - 99] .

```
    {
      flage_set = 1 ;
      x = 0 ;
    }
  }
  else
  {
   x = 0 ;
  }


 while(flage_set == 1)
 {
  second = 60 ;
  if(digitalRead(incr_pin)==0)
  {

while(digitalRead(incr_pin)==0){}
    moment++;
  }
  else if(digitalRead(decr_pin)==0)
  {

while(digitalRead(decr_pin)==0){}
    moment--;
  }
  moment =constrain(moment,1,99);
  Serial.println(x);
```

digit_1_val = 0; digit_2_val = 0; digit_3_val = moment%10; digit_4_val = moment/10; display_work(); if(digitalRead(set_pin) == 0) { delay(10); x++; if(x == 200) { flage_set = 0 ; x = 0 ; } } else { x = 0 ; } } }	
void display_work() { /*Serial.print(digit_4_val); Serial.print(digit_3_val); Serial.print(":");	في هذا التابع يتم تشغيل الخانات بالتتالي بفارق زمني قدره 5 ms والهدف من هذا الفاصل الزمني القصير هو تأمين خداع بصري لخداع

العين البشرية ولكي نعطي إنطباع بصري بأن الشاشات كلها تعمل بنفس الوقت.	Serial.print(digit_2_val); Serial.println(digit_1_val);*/ digitalWrite(sec_dig_1 ,0); digitalWrite(sec_dig_2 ,1); digitalWrite(mom_dig_3 ,1); digitalWrite(mom_dig_4 ,1); select_number(digit_1_val); delay(5); digitalWrite(sec_dig_1 ,1); digitalWrite(sec_dig_2 ,0); digitalWrite(mom_dig_3 ,1); digitalWrite(mom_dig_4 ,1); select_number(digit_2_val); delay(5); digitalWrite(sec_dig_1 ,1); digitalWrite(sec_dig_2 ,1); digitalWrite(mom_dig_3 ,0); digitalWrite(mom_dig_4 ,1); select_number_point(digit_3_val); delay(5); digitalWrite(sec_dig_1 ,1); digitalWrite(sec_dig_2 ,1); digitalWrite(mom_dig_3 ,1); digitalWrite(mom_dig_4 ,0);

select_number(digit_4_val); delay(5); }	
void select_number(int i) { for(int j=0 ; j<8 ; j++) { bit_val = bitRead(seg_val[i],j); digitalWrite(display_pin[j], bit_val); } } //========================== void select_number_point(int i) { for(int j=0 ; j<8 ; j++) { bit_val=bitRead(seg_val_point[i],j); digitalWrite(display_pin[j], bit_val); } }	التابعان الأخيران وظيفتهما طباعة الرقم المطلوب على الشاشة، فالأول يطبع الرقم بدون فاصلة عشرية بينما يقوم الثاني بطباعة الرقم مع الفاصلة العشرية.

تمارين الفصل الثاني

1

اكتب كود برمجي يتم من خلاله تغيير تدرج الألوان لكل لون من ألوان RGB LED وكتابة النسبة المئوية لكل لون على شاشة LCD، بحيث يكون لكل لون من الألوان مقاومة متغيرة للتحكم بزيادة وإنقاص نسبت اللون.

2

لدينا 8 حساسات نهاية شوط متصلة مع {16, 10 , 9 , 15 , 6 , 12 , 8 , 5} من لوحة الآردوينو ومعرفة كأقطاب دخل مع مقاومة رفع، كما يوجد لدينا LED متصل مع القطب 3 والذي يعتبر مخرج PWM والمطلوب:

كتابة كود يقرأ حالة الحساسات ويحول قيمتها لقيمة عشرية (8 bit = 255) ويسند القيمة لخرج نبضة PWM التي بدورها سوف تتحكم بقوة إضاءة LED وطباعة القيمة المقروءة من الحساسات وشدة الإضاءة على نافذة UART.

3

اكتب كود برمجي يطلب من المستخدم إدخال قيمة المتحول val عبر واجهة الاتصال التسلسلي UART ثم يتحقق البرنامج من القيمة فإن كانت القيمة ليست رقمية يعيد ويطلب إدخال القيمة، أما إن كانت القيمة رقمية فيقوم البرنامج بإدخالها على المعادلة الرياضية: x = val * 105 + val^2

4

أكتب كود برمجي يتم من خلاله إدخال أسماء الطلاب و علاماتهم في مادة معينة (العلامة ولتكن من عشرة) فيقوم البرنامج بتحديد مستوى كل طالب (العلامة بين 9 ~ 10 التقييم A ، أما إن كانت بين 8 ~ 7 التقييم B ، وغير ذلك التقييم C) ويطبع جدول فيه أسماء الطلاب والعلامات والتقييم.

الفصل الثالث:

المحركات الكهربائية والتحكم بها

مقـــدمـــة:

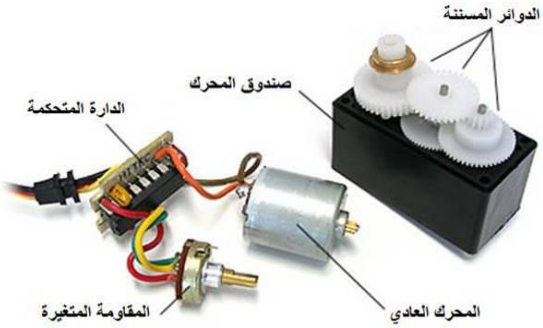
في هذا الفصل سيكون التركيز على أنواع المحركات المختلفة والتي يمكن التعامل معها عبر لوحة الآردوينو، تقوم فكرة العرض في هذا الفصل على استعراض البنية الداخلية لكل محرك سيتم دراسته ومعرفة كيف تم تصميمه، ثم بعد ذلك ننتقل لفهم دارة القيادة الخاصة به في حال وجدت، وفي النهاية يتم كتابة كود برمجي لتشغيل المحرك.

سنستعرض في هذا الفصل محركات Servo ذات الأهمية الكبيرة مع ذكر أهم الملاحظات عند التعامل معها، بعد ذلك ننتقل لمحركات **Brushless** كون طريقة كتابة الكود البرمجي متشابهة، ثم محركات الخطوة **Stepper Motor** باستعراض سريع لها ومثال لدارة قيادة خاصة بالمحركات الخطوية، ولا ننسى المشفرات البصرية **Encoder** التي تستخدم مع المحركات لمعرفة جهة الدوران والسرعة والتي سنستعرض نوعيها وطريقة استخدام كل نوع.

نختم الفصل بتمارين للتدريب والأهم من ذلك هو مشروع الفصل والذي سيكون **تحويل محرك تيار مستمر لمحرك سيرفو** يمكن التحكم بموقعه والذي سيكسب المتدرب خبرة جيدة جدا بإذن الله.

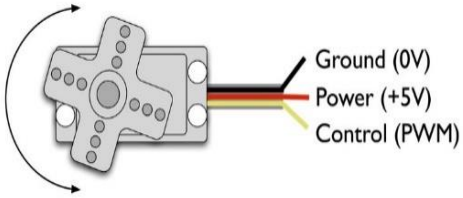
محركات السيرفو Servo Motor

محركات السيرفو Servo motor: عبارة عن محركات تيار مستمر **DC** مجهز بدارة الكترونية بداخله للتحكم بدقة في اتجاه الدوران وتوضعه، يأتي مع علبة تروس **Gear** وناقل حركة Shaft والذي يعطي عزما أكبر ودقة كبيرة، يتوفر منه نوعين أحدهما يستطيع الدوران بزاوية (**0 ~ 180º**) في كلا الاتجاهين وهو الأكثر شيوعا والآخر يدور بزاوية دوران كاملة في كلا الاتجاهين (**0 ~360º**).



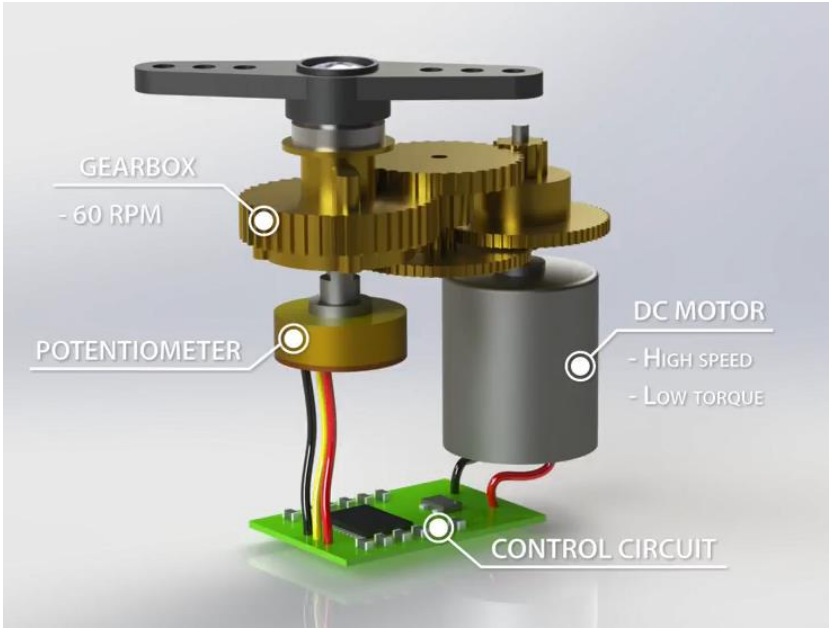
يستخدم محرك السيرفو في التطبيقات التي تحتاج التحكم بالموضع كالروبوتات وأجنحة الطائرات والعديد من التطبيقات التي تحتاج لحركة دقيقة وبطيئة، ويتوفر في السوق بأحجام وعزوم مختلفة.

يمتلك محرك السيرفو ثلاثة خطوط خارجة من المحرك هي خطي التغذية (**Vcc** & **GND**) وخط للتحكم بجهة الدوران والتموضع **Control**.



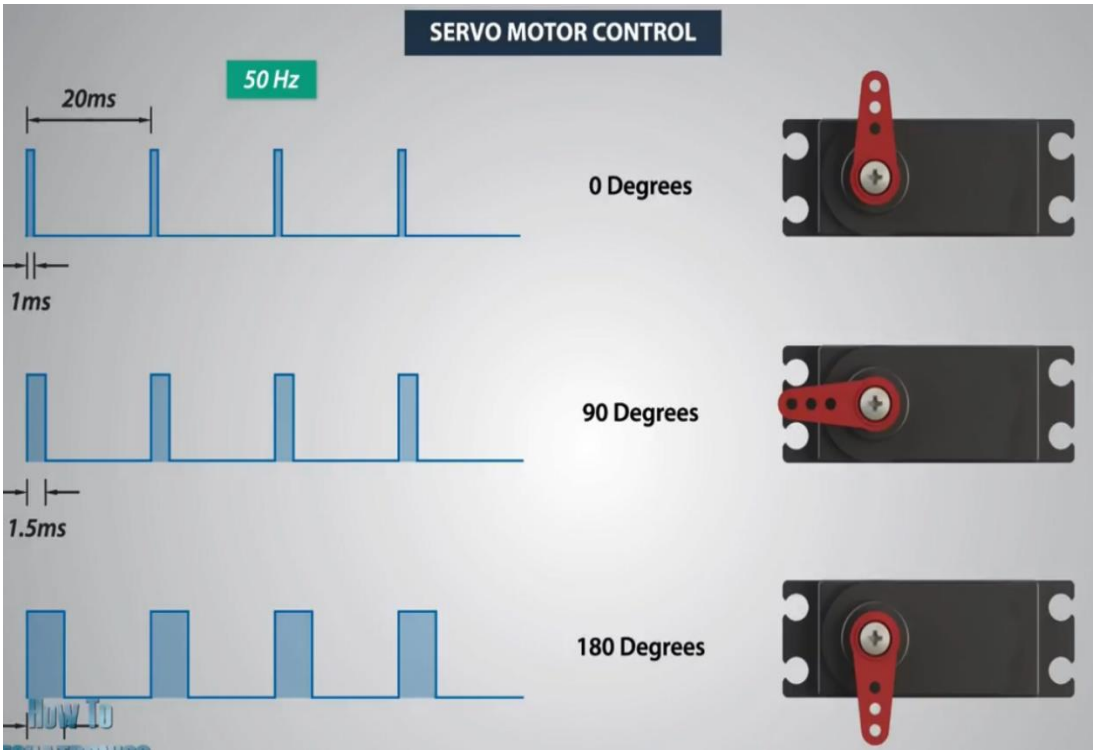
كيف يتم التحكم بمحركات السيرفو .... ؟؟؟

يتم التحكم بجهة دوران وتوضع محرك السيرفو عبر قطب التحكم **Control** الذي يكون غالبا باللون الأصفر , و تختلف محركات السيرفو فيما بينها في طريقة التحكم لكن غالب هذه المحركات لها نفس الطريقة التي سنطرحها الآن لذلك و عند شراء أي محرك يجب العودة للوثيقة الفنية لمعرفة آلية عمله , بالعودة لطريقة التشغيل يجب تطبيق إشارة على قطب التحكم بتردد **50Hz** أي ما يعادل **20 mS** أما قسم التحكم بالمحرك فيكون بعرض **2 mS** من مجمل النبضة ، ومن خلال التحكم بعرض النبضة يتم التحكم بجهة دوران المحرك أما آلية تحديد تموضع محور المحرك فتتم عبر مقاومة متغيرة متصلة مع محور محرك الـ **DC** الموجود داخل الغلاف الكلي بتغير زاوية الدوران تتغير قيمة المقاومة المتغيرة و بالتالي تغير قيمة الجهد المرجعي هذا التغير يتم تحليليه من قبل دارة تكاملية موجودة داخل الغلاف و عليه يتم توقيف المحرك عن الدوران ,كما توفر التروس **Gears** الموجودة داخل الغلاف تحويل السرعة لمحرك التيار المستمر **DC** إلى عزم و هذا هو سبب بطئ و دقة دوران المحرك.



تتوفر محركات السيرفو في السوق بعدة أحجام واختلاف الأحجام هو بسبب اختلاف العزوم فيما بينها حيث تقاس قوة هذه المحركات بواحدة **kg.cm** وقد تصل قوتها لـ **40 kg.cm**.

الشكل التالي يوضح القيم التي تأخذها دارة القيادة الداخلية الخاصة بمحرك السيرفو حيث تتراوح كما ذكرنا السابقا القيمة بين القيمة 1ms والتي يقابل الزاوية 0 درجة، مرورا بالقيمة 1.5ms والتي يقابلها الموقع 90 درجة وحتى القيمة 2ms المقابلة للقيمة 180 درجة.

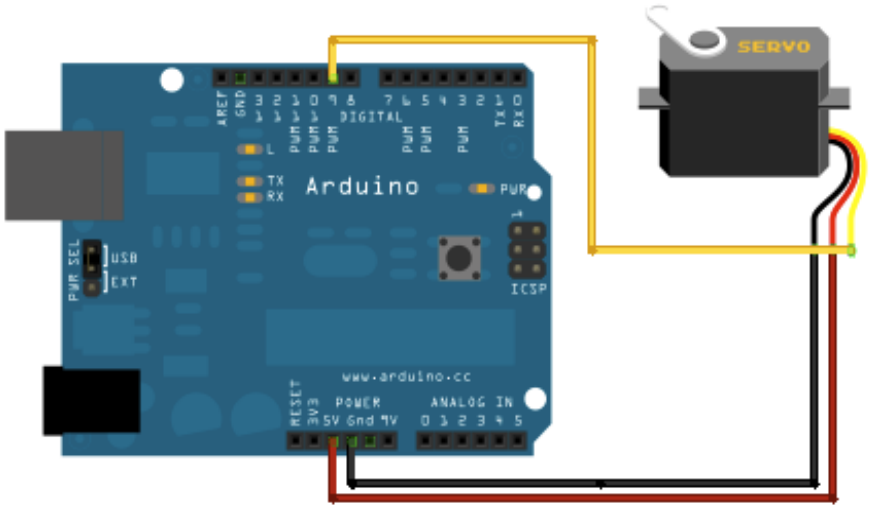


بعد إضافة المكتبية الخاصة بتشغيل محرك السيرفو (نوع hobby) علينا معرفة ضبط المحرك وأين يجب أن يوصل على لوحة الآردوينو وكيف يتم ضبطه ضمن بيئة التطوير Arduino IDE.

تتميز المكتبية الخاصة بتشغيل محركات السيرفو بقدرتها على تشغيل **12 محرك** سيرفو في لوحة الآردوينو **UNO** ويصل العدد حتى **60 محرك** في لوحة الآردوينو نوع **DUE**، وتحتوي على تعليمات خاصة لتحديد القطب الذي سيتم توصيل قطب الإشارة للمحرك مع لوحة الآردوينو وتحديد الزاوية التي سيتم تحريك المحرك إليها وهي محصورة بين القيمة **180+ ~ 0**.

توصيل محرك السيرفو مع لوحة الآردوينو

يبين الشكل التالي كيفية توصيل محرك سيرفو مع لوحة آردوينو UNO:



أما المكتبية والتعليمات المستخدمة مع محرك السيرفو فالجدول التالي يبين تفاصيل هذه المكتبية:

التعليمة	شرح التعليمة
`#include <Servo.h>`	**المكتبية الخاصة بتشغيل محركات السيرفو.**
`Servo myservo;`	**توليد متحول لتسمية كل محرك سيرفو سيتم توصيله.**
`myservo.attach(n);`	**تحديد رقم القطب n الذي سيتم توصيل المحرك إليه.**
`myservo.write(pos);`	**تحديد مكان تموضع محرك السيرفو عبر إعطائه قيمة الزاوية المطلوبة ضمن المجال المحدد لدوران المحرك.**

مثـــال 1 :

لنكتب كود برمجي لتدوير محرك السيرفو المبين في الشكل السابق المتصل مع لوحة UNO في الاتجاهين بكامل مجال العمل.

```
#include <Servo.h>
Servo servo1;
int pos = 0;

void setup()
{
   servo1.attach(8);
}

void loop()
{
   for (pos = 0; pos <= 180; pos += 1)
  {
   // goes from 0 degrees to 180 degrees
   // in steps of 1 degree
     servo1.write(pos);
   // servo2.write(pos);
   // tell servo to go to position in variable 'pos'
     delay(150);
   // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1)
  {
   // goes from 180 degrees to 0 degrees
     servo1.write(pos);
```

```
    //servo2.write(pos);
    // tell servo to go to position in variable 'pos'
      delay(15);
    // waits 15ms for the servo to reach the position
   }
}
```

في الكود السابق قمنا بعد تعريف المكتبية الخاصة بالمحرك وتعريف مكان توصيل المحرك بعد ذلك وفي حلقة البرنامج اللانهائية قمنا بتحريك المحرك بمقدار درجة بفاصل زمني بين كل درجة وأخرى بمقدار **150mS**، ويتم التحريك بزوايا متزايدة ومن ثم متناقصة وذلك عبر حلقتي الدوران for المتعاكستين في جهة الدوران.

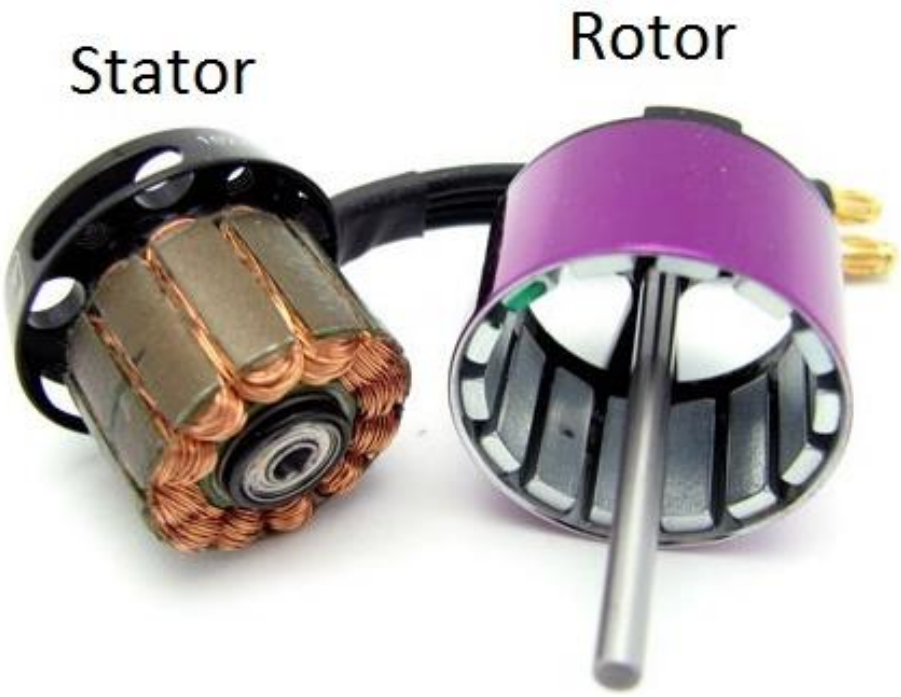
- ✓ يمكن تشغيل كما ذكرنا سابقا أكثر من محرك مع نفس لوحة الآردوينو فقط علينا تحديد اسم المحركات وتحديد أماكن توصيل كل محرك مع لوحة الآردوينو مع مراعاة كمية الكهرباء المستجرة من لوحة الآردوينو حتى لا تتعطل لوحة الآردوينو.
- ✓ يجب إضافة مكثف بقيمة **47uF / 25volt** على طرفي تغذية المحرك وذلك لضمان عمل المحرك بالشكل الصحيح.

محركات Brushless motor

وهي عبارة عن **محركات تيار مستمر** لا تحتوي في بنيتها على **المسفرات** أو ما يعرف باسم **Brush** ومنه أتت التسمية **Brushless** أي عديمة المسفرات.

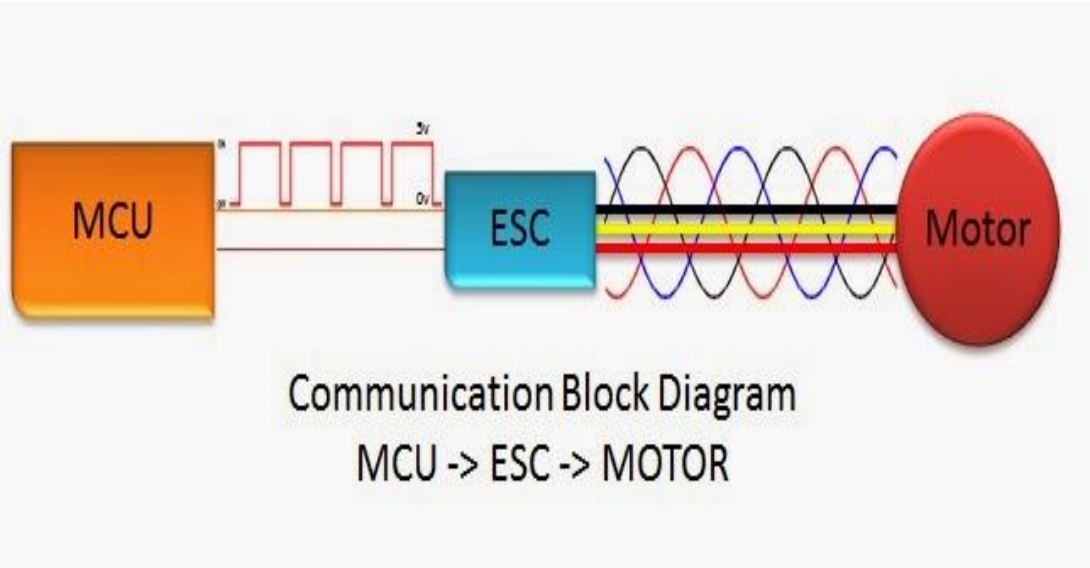
عادة **يسبب** وجود المسفرات (بعض المراجع تسميها بالفحمات) احتكاك وضوضاء للمحرك وارتفاع حرارته بزيادة سرعته كما أن المسفرات تحتاج للتبديل بشكل مستمر، لكن وبالاستغناء عن المسفرات نجد أن المحرك أصبحت سرعته كبيرة جدا لكن على حساب العزم إذ أن العزم يكاد يكون معدوم في هذه المحركات.

بالنسبة للبنية الداخلية لمحرك **Brushless** نجد أنه يتألف من قسمين أساسين هما **القسم الثابت Stator** (وهو القسم الداخلي الذي يضم ملفات المحرك وعددها ثلاثة ملفات لكل منها سلك خارج جسم المحرك والتي سوف يمر فيها التيار الكهربائي المقطع إلى موجة جيبية للحكم بالسرعة)، **والقسم الدوار Rotor** أو المتحرك (وهو القسم الخارجي من المحرك والذي تتوضع عليه الأجزاء المغناطيسية بشكل متتالي ومتعاكس ... **S , N , S , N**)، أما الأسلاك الخارجة من المحرك فتكون عادة ثلاثة أسلاك إما أن تأخذ كلها اللون **الأسود** أو تأخذ الألوان **الأحمر** و**الأسود** و**الأصفر**، ومن هذه الأقطاب يتم التحكم بسرعة المحرك وجهة الدوران.



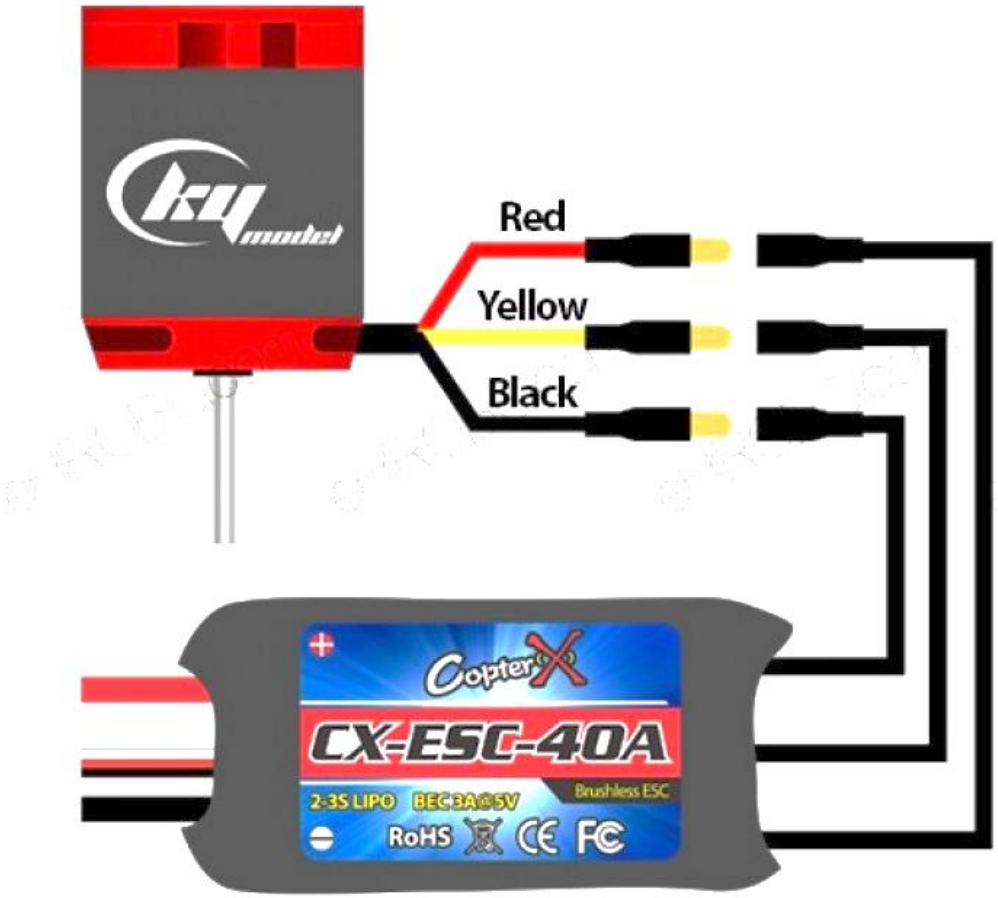
كيف يتم قيادة محركات Brushless

يعتبر التحكم بهذه المحركات صعب جدا كون أن الإشارة الداخلة للمحرك يجب أن تكون موجة sine wave بثلاثة أطوار كما يجب تقطيع هذه الموجة لكي نتمكن من تحديد سرعة المحرك، لذلك يتوفر دارات قيادة خاصة بهذه المحركات والتي تعرف باسم Electrical Speed Control أو اختصارا بـ **ESC**، ويتوفر من لوحات القيادة العديد من الأشكال والأحجام وأهم بارامتر يميز هذه اللوحات عن بعضها البعض هو شدة التيار الكهربائي الذي يمكن أن تتحمله هذه اللوحات.



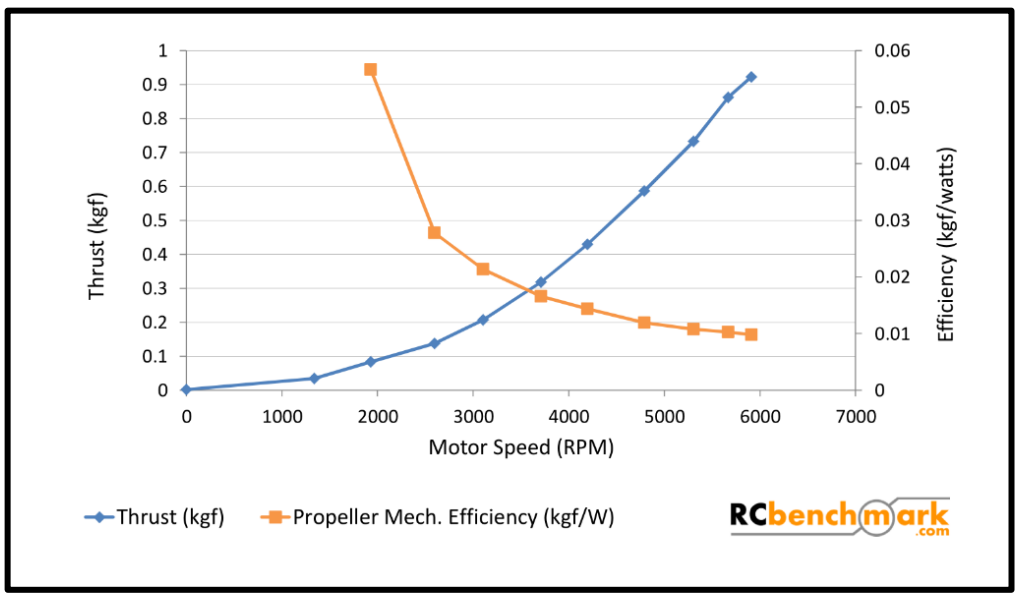
أما **الأسلاك** التي تكون عليها ففي **الدخل** تحتوي على سلكين ثخينين (**أحمر** + **أسود**) يتم وصلهما لمنبع التغذية (عادة يكون منبع التغذية عبارة عن بطارية ليثيوم ذات أمبير كبير)، وأسلاك **تحكم** تكون متصلة مع بعضها، قد تكون ثلاثة أسلاك تحمل الألوان **الأسود** و**الأحمر** و**البرتقالي**، أو سلكين فقط تحمل اللونين **الأسود** ولون آخر، وفي كلتا الحالتين يكون إشارة التحكم المدخلة هي نفس الإشارة المدخلة لمحركات السيرفو (*تم دراسة محركات السيرفو في فقرة سابقة من الفصل*).

أما آلية توصيل لوحة التحكم **ESC** مع **المحرك** فيتم توصيل أسلاك الخرج من لوحة التحكم مع الألوان التي تقابلها من المحرك (**الأسود** مع **الأسود**، **الأصفر** مع **الأصفر**، **الأحمر** مع **الأحمر**) وفي حال كانت الأسلاك تحمل نفس اللون فيتم توصيلها بالترتيب مع لوحة التحكم، مع الأخذ بعين الاعتبار أن توصيل **الأحمر** مكان **الأسود** بالنسبة للمحرك سوف يؤدي **لعكس** جهة دورانه.



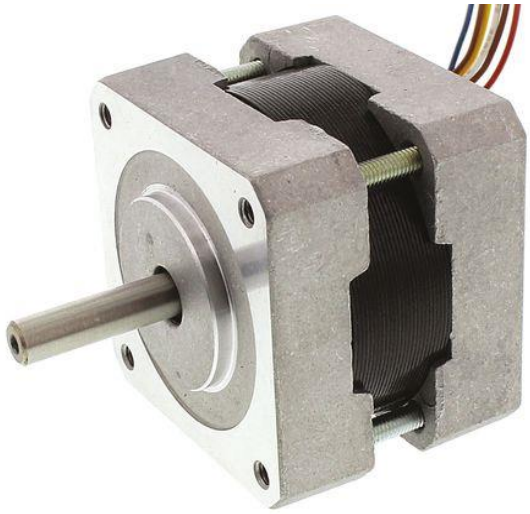
ملاحظات عامة:

- ✓ تقاس قوة هذه المحركات بواحدة **KV** والتي تدل على عدد **الدورات** التي يدورها المحرك عند تغذيته بواحد **فولط**، وبالتالي فلو تم تغذية المحرك بمنبع تغذية وليكن 8v فإن سرعة المحرك سوف تكون القيمة المكتوبة عليه مضروبة بالعدد 8.
- ✓ العلاقة بين **سرعة speed** المحرك **وقوة الرفع** thrust للمحرك هي علاقة **خطية عكسية**، فكلما زادت السرعة نقصت قوة الرفع للمحرك والعكس بالعكس، فقد نجد نفس المحرك من نفس الشركة ونفس الأبعاد لكن بسرعتين الأول سرعته عالية وعزم منخفض والثاني قوة الرفع له كبيرة مع انخفاض السرعة هذا الاختلاف مفيد لتحديد طبيعة العمل الذي سيقوم به المحرك فللطائرات المجنحة يجب أن تكون قوة الرفع كبيرة ولا تهم سرعة المحرك عندئذ أما في الطائرات المروحية فيجب مراعات السرعة للمحركات ولا يهم عندئذ موضوع الرفع للمحرك..



- ✓ عند التحكم بالمحرك من لوحة الآردوينو عبر **مكتبية** التحكم بمحرك السيرفو فإن محرك **Brushless** لن يبدأ بالدوران حتى تصبح القيمة المسندة للتعليمة servo1.write(pos) أكبر من القيمة **100**، ويجب الانتباه نوع الدرايفر **ESC** فالأنواع التجارية تتصف بضعف التجاوب فلا يمكن إعطاؤها قيم مختلفة بسرعة كبيرة بل نحتاج إعطاء قيم متدرجة حتى تعمل بشكل سليم ولا نسبب تلفها، على العكس من الأنواع ذات الجودة العالية التي تتميز بقدرة عالية على تغيير السرعة لكن بالمقابل سيكون الفرق بشكل أساسي في الكلفة.

المحركات الخطوية Stepper Motor



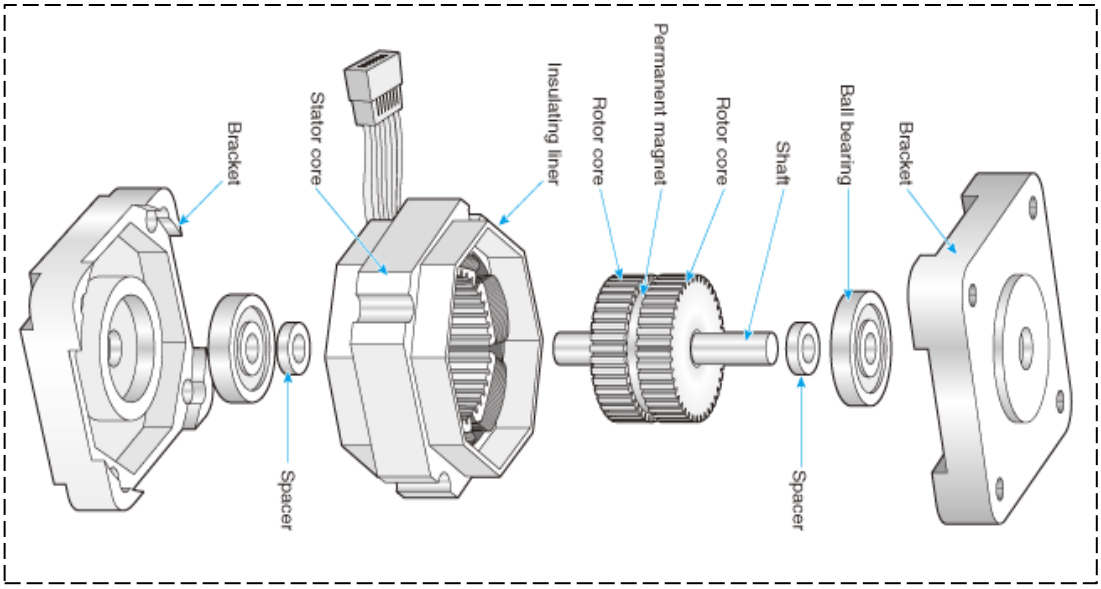
وهي عبارة عن محركات تيار مستمر لكنها تتميز بعزوم كبيرة ودقة في الحركة ويستخدم للتحكم في الآلات التي تحتاج للتحكم بدقة في الحركة كالطابعات ومقصات الليزر ومكنات الحفر CNC والعديد من التطبيقات الأخرى، وسميت بالمحركات الخطوية لأنها قادرة على الدوران خطوة خطوة حتى تتم دورة كاملة وبالتالي لهذه الخطوات قيم زاوية تقابلها.

يختلف عدد الخطوات بين هذه المحركات فمنها من يكون مقدار كل خطوة 1.8^o درجة أي حتى تتم دورة كاملة 360^o فإنها تحتاج لـ 200 خطوة، ومنها ما تكون قيمة الخطوة 3.6^o أي تحتاج لـ 100 خطوة لإتمام دورة كاملة .... وهكذا، وكلما صغرت قيمة الخطوة ازدادت دقة المحرك وتكون قيمة الخطوة وجهد وتيار أطوار المحرك مدونة عليه.

لو أردنا تقسيم محركات الخطوة تبعا لمغناطيسيتها الداخلية فسوف نجد لها نوعان هما محركات خطويه أحادية القطبية **Unipolar** ومحركات خطويه ثنائية القطبية **Bipolar** والخلاف بين النوعين السابقين فهو في طريقة القيادة وفي العزم الذي يقدمه كل نوع من هذه المحركات وهذين النوعين هما اللذان سيتم دراستهما في بحثنا هذا ولن نتطرق لباقي أنواع المحركات الخطوية.

تتألف المحركات الخطوية ذات المغناطيسية الدائمة (أحادي القطبية وثنائي القطبية) من الأقسام التالية:

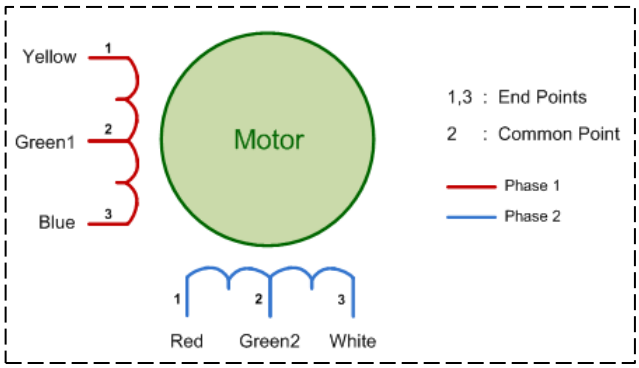
- **القسم الدوار** والذي يخرج منه محور المحرك.
- **القسم الثابت** ويضم ملفات النحاس التي تقدم المغناطيسية الدائمة ويختلف عدد هذه الملفات وطريقة توصيلها تبعا لنوع المحرك.
- **ملحقات** كأسلاك خرج الملفات وبيليات على طرفي المحور لتسهيل حركة الدوران والغلاف المعدني الخارجي.



تقاس قوة محركات الخطوة بواحدة الـ **kg.cm** أي **الثقل** المحدد بواحدة الكيلو غرام الذي يستطيع محور المحرك ذو **الطول** المحدد بواحدة السنتيمتر تحمله.

سنعتمد في طريقة الشرح على أخذ كل نوع على حدا وذكر البنية والخصائص ثم عرض آلية القيادة وبعد ذلك قيادة المحرك باستخدام الآردوينو.

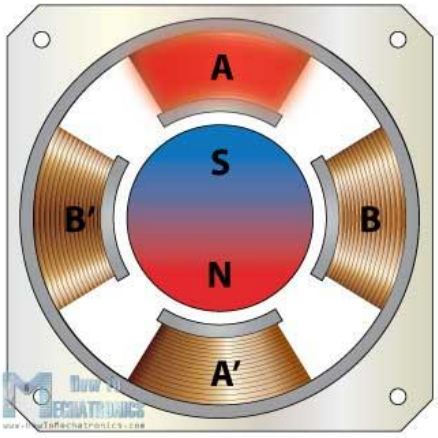
المحركات الخطوية أحادية القطبية Unipolar Stepper Motor:

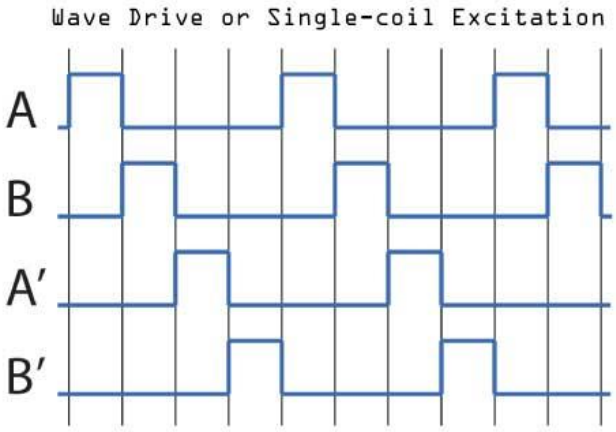


أغلب هذه المحركات تحتوي على ستة أقطاب على الخرج و التي تتوزع كما في الشكل الجانبي يصطلح عادة تسمية الملف الأول أو **Phase 1** بالرمز **A** وطرفا هذا الملف تحمل الرمز A1 & A2 ، بينما يرمز للملف الثاني أو **Phase 2** بالرمز **B** و كذلك لطرفاه فيحملان الترميز B1 & B2، و للترميز السابق أهمية في تسهيل شرح كيفية تدوير هذا النوع من المحركات ، و كما نلاحظ فإن كل من هذين الملفين لهما نقطة منتصف و يكون لهما نفس اللون في الخرج، و في حالات معينة فقد تكون ألوان الأسلاك مختلفة عن الترميز الشائع فعندها يتم تحديد أطراف الملفات و نقط المنتصف للملفات عبر جهاز الأفومتر من خلال قياس قيم مقاومات ملفات المحرك.

أما طريقة قيادة هذا المحرك فتتم بوصل نقطتي منتصف الملفين مع بعضها إلى قطب التغذية **Vcc** أو إلى قطب الأرضي وتغذية باقي أطراف الملف بالتغذية المتممة كما في التسلسل التالي المبين في الجدول:

A1	A2	B1	B2
0	0	0	1
0	1	0	0
0	0	1	0
1	0	0	0

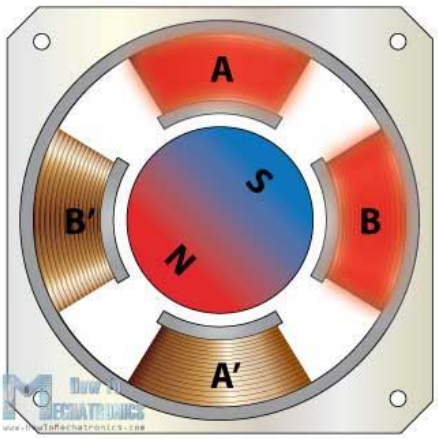


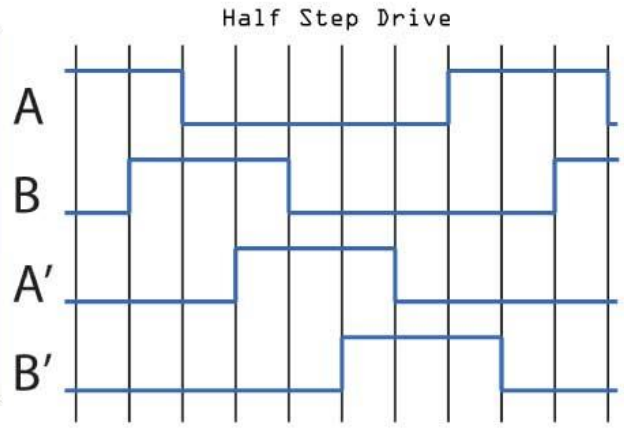


A1	A2	B1	B2
0	1	0	1
0	1	1	0
1	0	1	0
1	0	0	1

جدول قيادة محرك أحادي القطبية بمقدار خطوة

في نمط أحادي الفاز، في هذا النمط يتم تغذية طرف واحد من أحد أطراف ملفي المحرك فنحصل على دوران بمقدار خطوة واستطاعة منخفضة وبالتالي استهلاك أقل للتيار.

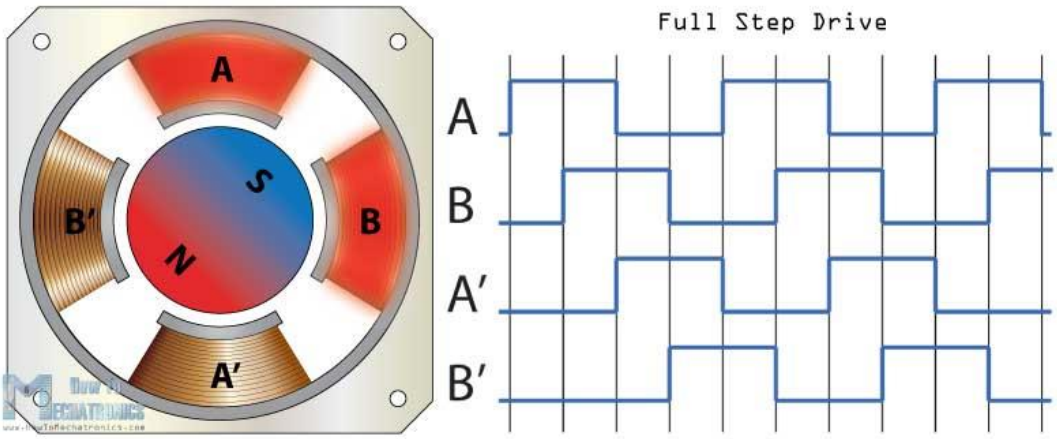




أما لقيادة المحرك في نمط النصف خطوة فيتم دمج الجدولين السابقين مع بعضهما وبالتالي إعطاء تسلسل أوامر لشغيل المحرك مرة في نمط الفاز ومرة في نمط ثنائي الفاز كما يبينه الجدول.

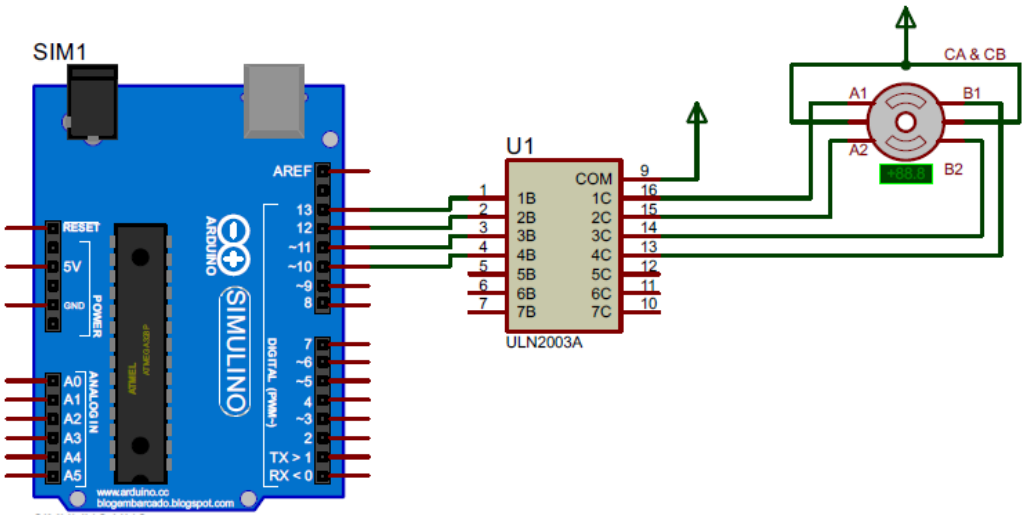
A1	A2	B1	B2
0	0	0	1
0	1	0	1
0	1	0	0
0	1	1	0
0	0	1	0
1	0	1	0
1	0	0	0
1	0	0	1

هذا النمط يؤمن لنا إمكانية قيادة المحرك وتحديد تموضعه بدقة أكبر ومضاعفة عدد الخطوات التي صنع عليها المحرك، فلو كانت خطوة المحرك 1.8درجة ففي هذا النمط ستصبح النصف خطوة 0.9 درجة.



دارة قيادة محرك خطوي أحادي القطبية Unipolar Stepper Motor:

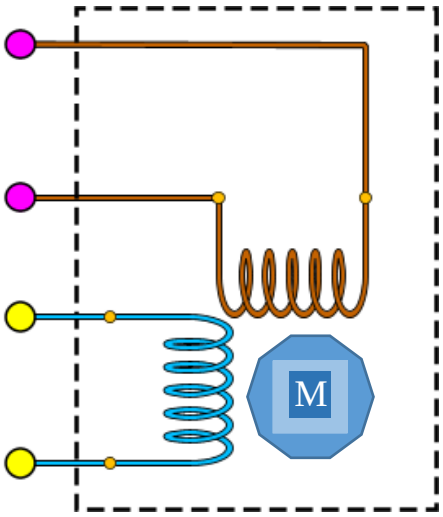
تعتبر عملية قيادة المحرك الخطوي أحادي الطور **Unipolar** عملية سهلة مقارنة مع عملية قيادة النوع الآخر من هذه المحركات وهي المحركات ثنائية القطبية **Bipolar**، فيكفي لتشغيل المحرك الخطوي أحادي الطور تأمين تغذية لخطي منتصف الملف من كل طور وتأمين التغذية المتممة لكل طرف من أطراف الملفات بالترتيب الذي ذكرناه في الترتيب السابق لاختيار نمط العمل الذي نريده، سنقدم الآن دارة قيادة لمحرك خطوي أحادي الطور لفهم آلية عمل الدارة:



إذا من الدارة السابقة نجد أنه ولتحريك المحرك علينا التحكم بأربع خطوط هي خطوط أطراف كل فاز من فازات المحرك، وبالتالي فإنه لدينا ثلاث بارامترات يجب تحديدها لتحريك المحرك بالشكل الصحيح وهي:

- عدد خطوات المحرك الذي لدينا Number of Step لإتمام دورة كاملة.
- سرعة الدوران المطلوبة rpm (عدد الدورات التي يقضيها المحرك خلال دقيقة).
- الأقطاب لوحة الآردوينو التي تم توصيل أطراف الفازات إليها.

المحركات الخطوية ثنائية القطبية Bipolar Stepper Motor:



يعتبر هذا النوع من المحركات الخطوية أكثر شهرة من سابقة لكن يتطلب آلية تحكم أكثر تعقيدا إذ يجب تغيير جهة التيار المار في الملفات بعد كل خطوة دورانية ، تمتلك هذه المحركات في خرجها على **أربع أسلاك** فقط يمثل كل خطين منها طرفي ملف و لا يوجد لهذه الملفات نقطة منتصف كما في المحركـــــات الخطويـــة أحادية القطبية ، سـيتم تسـمية أطراف الملفين بالترميز **A , B** للملف الأول و **C , D** للملف الثاني ، تتميز هذه المحركات بعزم أكبر من المحركات أحادية القطبية و ذلك لأننا لتدوير هذه المحركات علينا التحكم بطرفي الملف في نفس الوقت ، أما أنماط التشغيل فهي مشابهة للأنماط التي درسناها في النوع السابق وبنفس ترتيب الأوامر لكن الاختلاف فقط هو دارة القيادة.

أحادي الفاز	ثنائي الفاز	نمط النصف خطوة
A B	**A B - C D**	A B
C D	**B A - C D**	**A B - C D**
B A	**B A - D C**	C D
D C	**A B - D C**	**B A - C D**
-	-	B A
-	-	**B A - D C**
-	-	D C
-	-	**A B - D C**

تشغيل المحرك الخطوي مع لوحة الآردوينو:

تتوفر في بيئة الآردوينو مكتبية جاهزة للتحكم بالمحرك الخطوي بنوعيه أحادي القطبية وثنائي القطبية إذ أن ترتيب الأوامر المرسلة هو نفسه لكن الخلاف في دارة القدرة فقط المسؤولة عن تدوير المحرك لذلك سندرس المكتبية المخصصة للمحركات الخطوية دون التطرق لنوع المحرك الذي نتعامل معه:

التعليمة	شرح التعليمة
`#include <Stepper.h>`	**تعريف المكتبية الخاصة بالمحرك الخطوي.**
`#define STEPS 100`	**تعريف ثابت يحدد عدد خطوات المحرك الذي عندنا.**
`Stepper name(STEPS, 8, 9, 10, 11);`	**تسمية المحرك الموصول مع لوحة الآردوينو name ومن ثم تحديد الأقطاب التي تم توصيل أقطاب المحرك إليها , وعدد خطوات المحرك ليتم دورة واحدة.**
`stepper.setSpeed(30);`	**ضبط سرعة المحرك ليتم دورة عدد محدد من الدورات خلال دقيقة واحدة.**
`stepper.step(x);`	**تحديد عدد الخطوات x المطلوب تحريك المحرك بها.**

تتالي أوامر القيادة يتم في نمط الخطوة الكاملة ثنائي الفاز، أي يتم التحكم بفازين معا لتحريك المحرك والذي يعطي عزم أكبر عند الدوران.

مثـــال 2 :

لنكتب كود برمجي لتشغيل محرك خطوي أحادي القطبية عدد خطواته 200 مع لوحة الآردوينو نوع UNO.

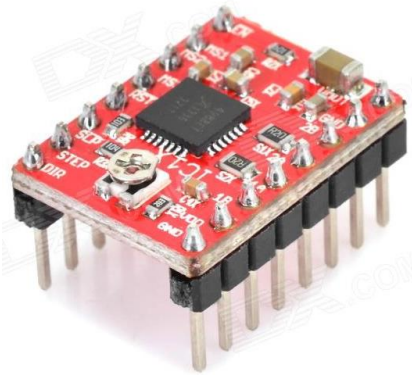
```
#include <Stepper.h>
// change this to the number of steps on your motor
#define STEPS 200

Stepper stepper(STEPS, 10, 11, 12, 13);

void setup()
{

  stepper.setSpeed(1);
}

void loop()
 {
  // put your main code here, to run repeatedly:
stepper.step(3);
delay(1000);
}
```

بعد تعريف المكتبية الخاصة بالمحرك الخطوي نقوم بتعريف متحول خاص بعدد خطوات المحرك التي ينجزها في دورة واحدة كما نحدد الأقطاب التي سيتم توصيل أطراف أطوار المحرك، بعد ذلك وفي قسم الإعداد نقوم بتحديد سرعة المحرك الخطوي حيث نضع عدد اللفات في دقيقة واحدة وفي الحلقة اللانهائية نقوم بتحريك المحرك ثلاث خطوات كل واحد ثانية.

تشغيل المحرك الخطوي ثنائي الطور مع لوحة الآردوينو عبر موديولات خاصة:

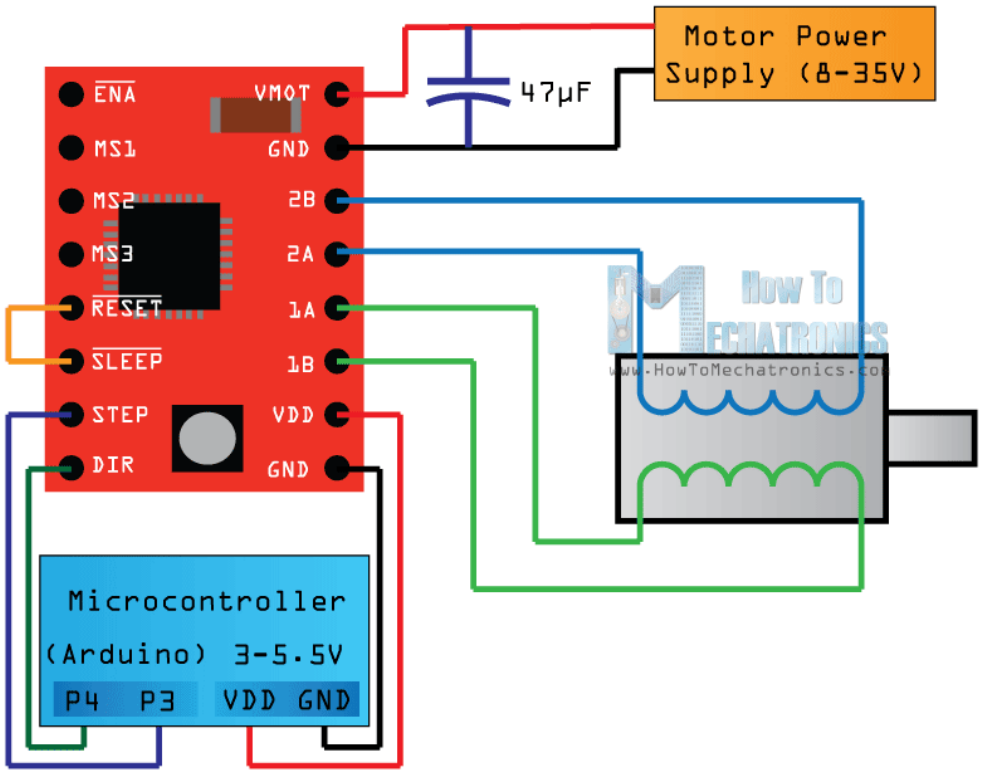
كما ذكرنا سابقا فإن دارات قيادة المحركات الخطوية ثنائية الطور تعتبر عملية معقدة أكثر بكثير من قيادة المحركات الخطوية أحادية الطور، حيث أن دارات قيادة المحركات الخطوية ثنائية الطور تحتاج إلى عناصر إلكترونية كثيرة فهي تحتاج جسر قيادة كامل **Full H-bridge** لقيادة كل طور من أطوار المحرك بالإضافة لعناصر أخرى لتحسس التيار وغير ذلك.

يتوفر في السوق العديد من الموديولات الجاهزة لقيادة المحركات الخطوية ثنائية الطور وتتفق هذه الموديولات بالمجمل بمعظم الخصائص وتختلف فيما بينها في بعض التفاصيل، سنستعرض أحد هذه الموديولات وليكن الموديول الذي يضم الدارة التكاملية **IC: A4988** ، وآلية التحكم بهذا الموديول لتدوير المحرك وغير ذلك من الخصائص ، **يتميز موديول التحكم A4988 بالخصائص العامة التالية:**



- تيار تشغيل مستمر 1A.
- حماية من قصر في الحمل.
- حماية من الحرارة الزائدة.
- جهد تحكم منطقي ضمن المجال 3.3 ~ 5 volt.
- جهد تشغيل للمحرك ضمن المجال 8 ~ 35 volt.
- تيار تشغيل لكل طور من أطوار المحرك يصل حتى 2Amp.
- جدول اختيار للخطوة (**1 , 1/2 , 1/4 , 1/8 , 1/16**).
- ضبط تيار عمل المحرك.

أما طريقة توصيل واستخدام الموديول **A4988** فهي موضحة في الشكل التالي:



وظيفة الأقطاب	أقطاب الموديول
التحكم بأطوار المحرك وتدويرها.	1A , 1B , 2A , 2B
أقطاب تغذية المحرك، يجب توصيل مكثف كيميائي بقيمة 47uF بين هذين القطبين.	VMOT & GND
تغذية القسم المنطقي من الموديول	VDD & GND
تحديد الاتجاه	DIR
تردد الخطوة	STEP
تحديد قيمة مقسم الخطوة	MS1 , MS2 , MS3

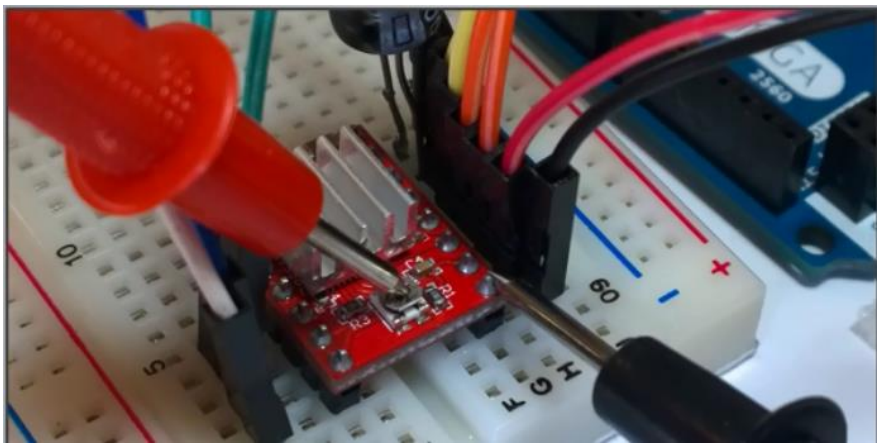
الأقطاب MS1 , MS2 , MS3 وظيفتها تقسيم الخطوة الأساسية لأجزاء من الخطوة والهدف من ذلك زيادة الدقة في الحركة لكن سيتم ذلك على حساب العزم , فكلما زاد تقسيم الخطوة نقص العزم , أما آلية التقسيم فتتم باستخدام تقنية **PWM** ويعرف هذا التقسيم بـــ **Micro Step**، الجدول التالي يبين تقسيم الخطوة تبعا لحالة الأقطاب المسؤولة عن ذلك:

MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

كما يتوفر على الموديول **مقاومة متغيرة** وظيفتها تحديد التيار الحدي المستجر من الموديول حيث يعطى قانون التيار المسحوب من الموديول بالعلاقة:

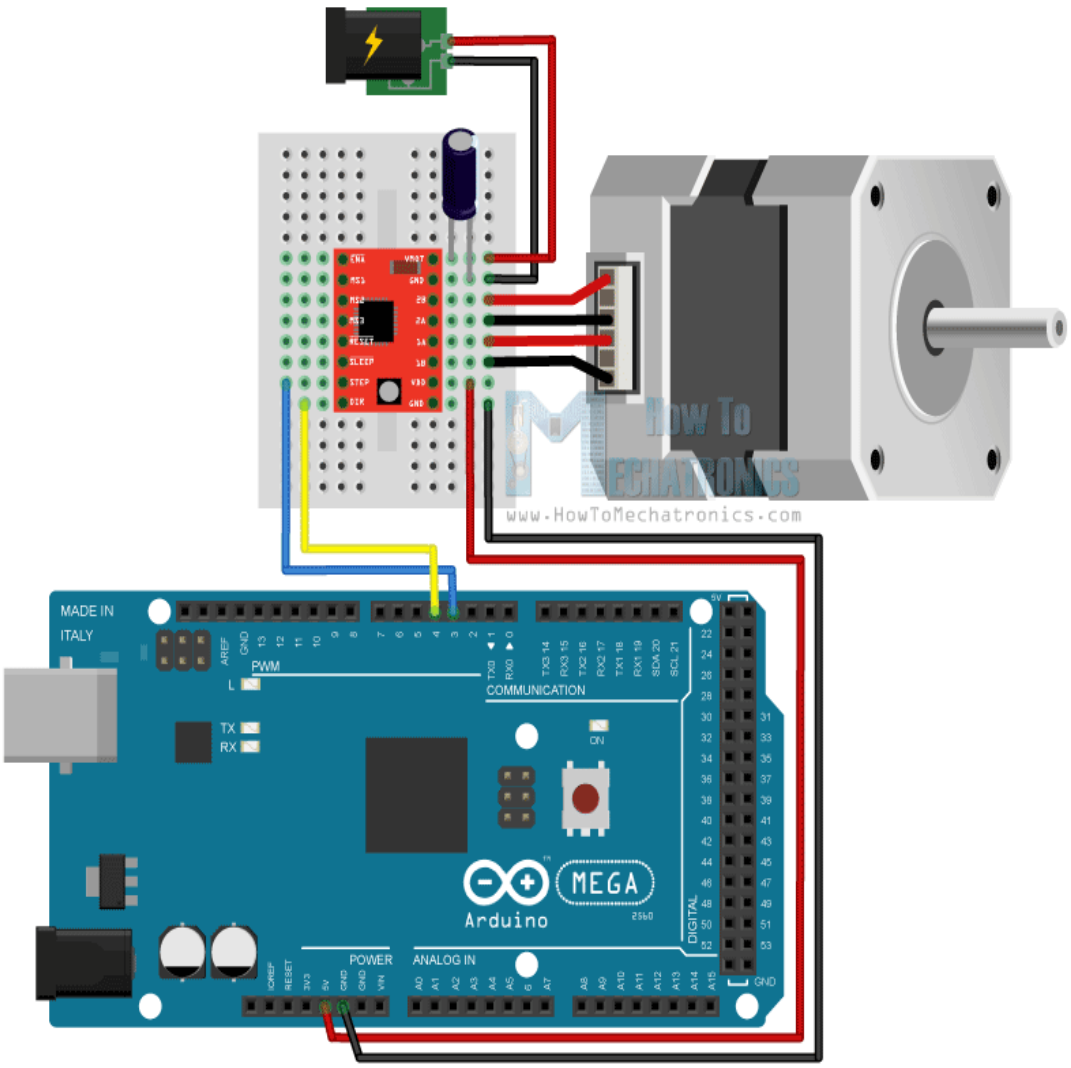
$$\textbf{Current limit} = \textbf{Vref} * 2$$

أما الجهد المرجعي فيضبط عبر المقاومة المتغيرة ويقاس مباشرة من المقاومة المتغيرة كما في الشكل التالي، وبعد قياس قيمة الجهد نعوضه في العلاقة السابقة فنحصل على قيمة التيار الحدي.



مثـــال 3 : **في هذا المثال سنقوم بتشغيل المحرك الخطوي ثنائي الطور مع موديول A4988 مع لوحة آردوينو نوع Mega.**

يتم توصيل قطب التحكم بالاتجاه **DIR** من الموديول مع القطب الذي رقمه **4**، أما القطب المسؤول عن تقديم النبضة **STEP** من الموديول فيتم وصله مع القطب **3** من لوحة الآردوينو.



```
const int stepPin = 3;
const int dirPin = 4;


void setup() {
  // Sets the two pins as Outputs
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
}
void loop() {
  digitalWrite(dirPin,HIGH);
// Makes 200 pulses for making one full cycle rotation
  for(int x = 0; x < 200; x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(500);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(500);
  }
  delay(1000);
 //Changes the rotations direction
  digitalWrite(dirPin,LOW);
```

```
// Makes 400 pulses for making two full cycle rotation
      for(int x = 0; x < 400; x++) {
        digitalWrite(stepPin,HIGH);
        delayMicroseconds(500);
        digitalWrite(stepPin,LOW);
        delayMicroseconds(500);
      }
      delay(1000);
    }
```

الكود السابق لا يعتمد على أي مكتبية فهو عبارة عن قطبين للتحكم أحدهما يولد نبضات متتالية وهو `stepPin` والثاني يحدد الاتجاه `dirPin` وبالتالي يتم تحدد سرعة الدوران من الأول وجهة الحركة من الثاني.

عند التعامل مع المحركات الخطوية نجد أنها تقاس وفق الترميز العالمي NEMA-xx فما المقصود بهذا الترميز .... ؟

تعتبر جمعية أو اتحاد مصنعي الكهرباء الوطنية والتي يعبر عنها بالاختصار **NEMA**:

NEMA-**N**ational **E**lectric **M**anufactures **A**ssociation

أكبر جمعية تجارية لمصنعي المعدات الكهربائية في الولايات المتحدة الأمريكية، تأسست في عام ١٩٢٦ وتحتفظ بمقرها في روسلين- فيرجينيا، وتضم حوالي 450 شركة عضو في التجمع وكلها شركات لإنتاج المعدات الكهربائية.

كما يوجد تحالف من الشركات المنتجة للمعدات الطبية والتابعة لهذا الاتحاد والتي تنتج أجهزة الرنين المغناطيسي والتصوير المقطعي والتصوير بالأشعة السينية وغير ذلك.

ولهذه الرابطة واحداتها الخاصة في القياس فهي تعتمد واحدة الإنش في القياس والنيوتن في العزوم وغير ذلك من الواحدات الخاصة بهذه الرابطة، وفي المحركات الخطوية نجد مثلا محرك بقياس **NEMA 17** أو غير ذلك من الأرقام لكن في النتيجة فإن الرقم يدل على قياس قطر المسقط الأمامي للمحرك بواحدة الإنش مقسوما على 10 أي أن 1.7inch = 17 وهكذا ... ولها أرقام محددة وليست عشوائية فنجد ... **NEMA23 , 34 ,42**.

المشفرات البصرية The Encoder

في حالات معينة وفي بعض التطبيقات التي نستخدم بها المحركات يتطلب هذا التطبيق تحديد جهة دوران المحرك ومعرفة سرعته أو معرفة موضع محور المحرك بالنسبة للدائرة (تحديد الزاوية بالنسبة لنقطة صفرية على محور المحرك) لذلك نستخدم الانكودر(Encoder) أو المشفر والذي يكون متصل مع محور المحرك.

إذا: الانكودر جهاز له خرج **رقمي** متغير تبعا للإزاحة الدورانية أو الخطية.

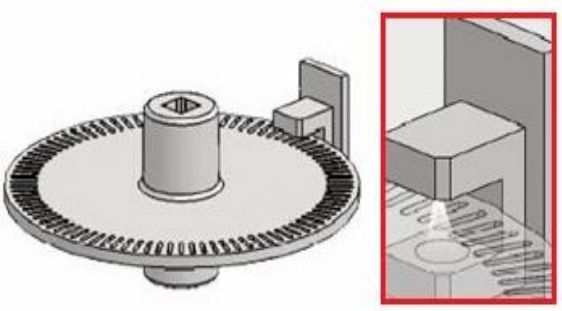
أنواع المشفر البصري Encoder:

يوجد نوعين من المشفر البصري هما:

- **المشفر المتزايد**: يكشف الإزاحة الدورانية أو الخطية بالنسبة إلى وضع استناد محدد.
- **المشفر المطلق**: يعطي التغير الزاوي أو الخطي الفعلي.

بنية المشفر البصري الـ Encoder:

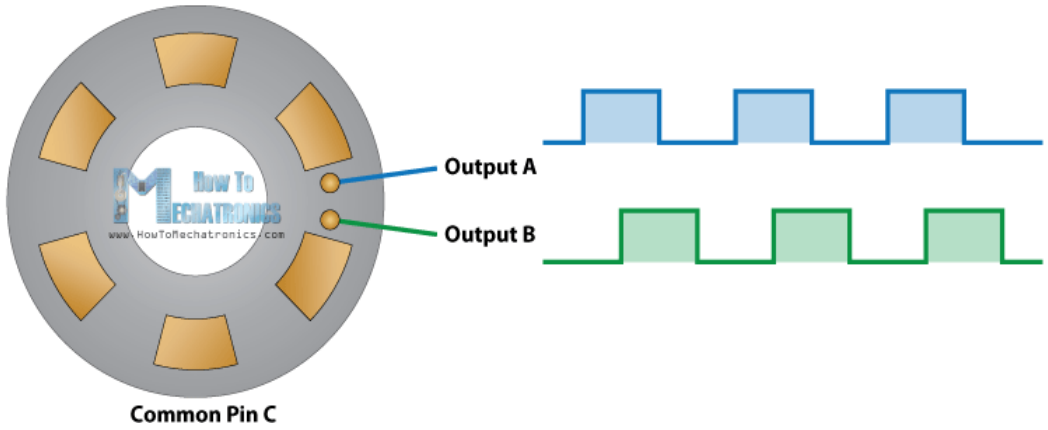
المشفر المتزايد:

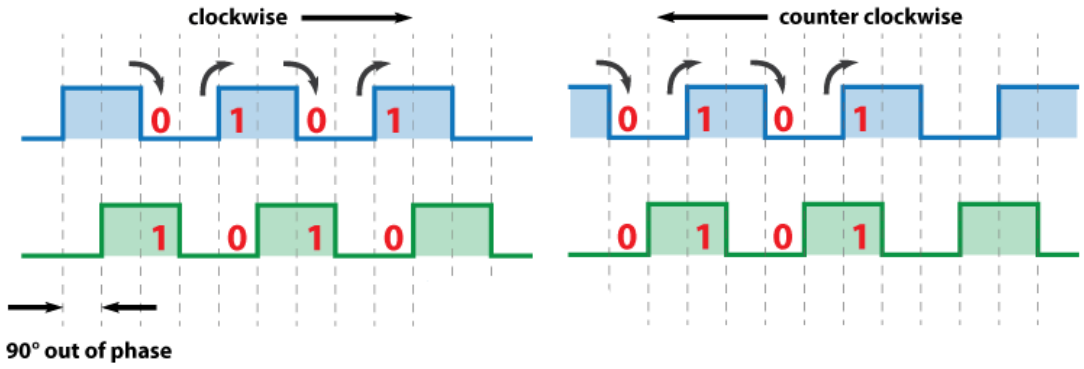


كما ذكرنا سابقا فإن هذا النوع يستخدم لتحديد جهة الدوران و السرعة حيث يحتوي على ثلاثة أقطاب للخرج هي **A** , **B** , INDEX يكون الخرج **B** متأخرا عن الخرج **A** بمقدار نصف نبضة , هذا الفارق يستخدم لتحديد جهة الدوران، فعندما يتحرك محور القرص باتجاه عقارب الساعة فإن النبضة على الخرج **A** سترد أولا ثم وبعد نصف

نبضة (أي ربع دور) تصل النبضة على الخرج **B** هذا التأخر في ورود النبضة ينتج عنه معرفة جهة دوران محور المحرك باتجاه عقارب الساعة **CW**.

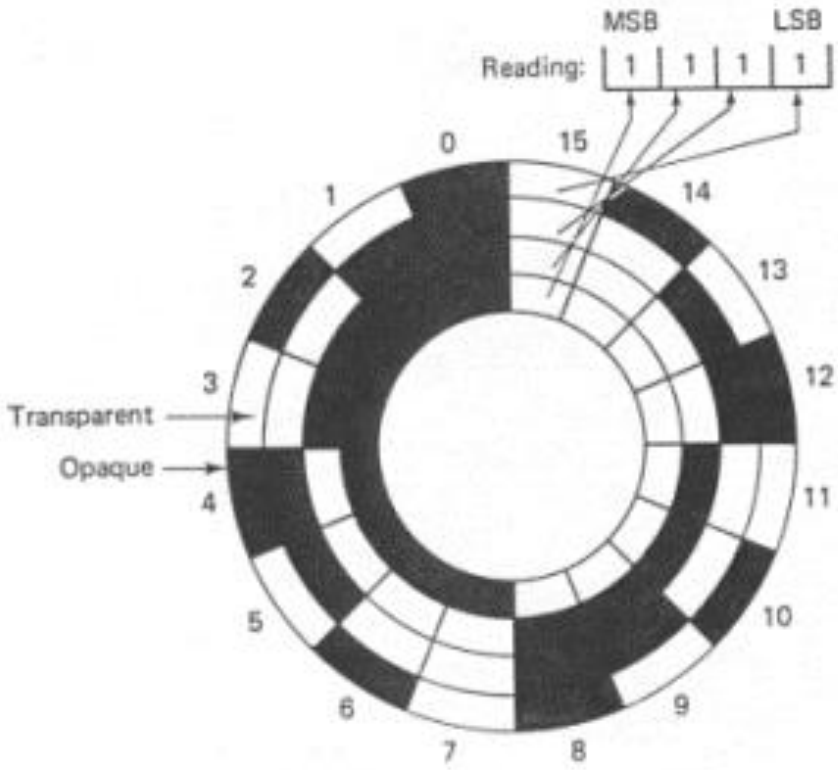
أما لو كانت جهة دوران محور المحرك بعكس عقارب الساعة فينتج عن ذلك أن جهة ورود النبضة على الخرج **B** ستسبق النبضة على الخرج **A** بمقدار نصف نبضة وبالتالي فدوران المحرك بعكس عقارب الساعة **CCW**.





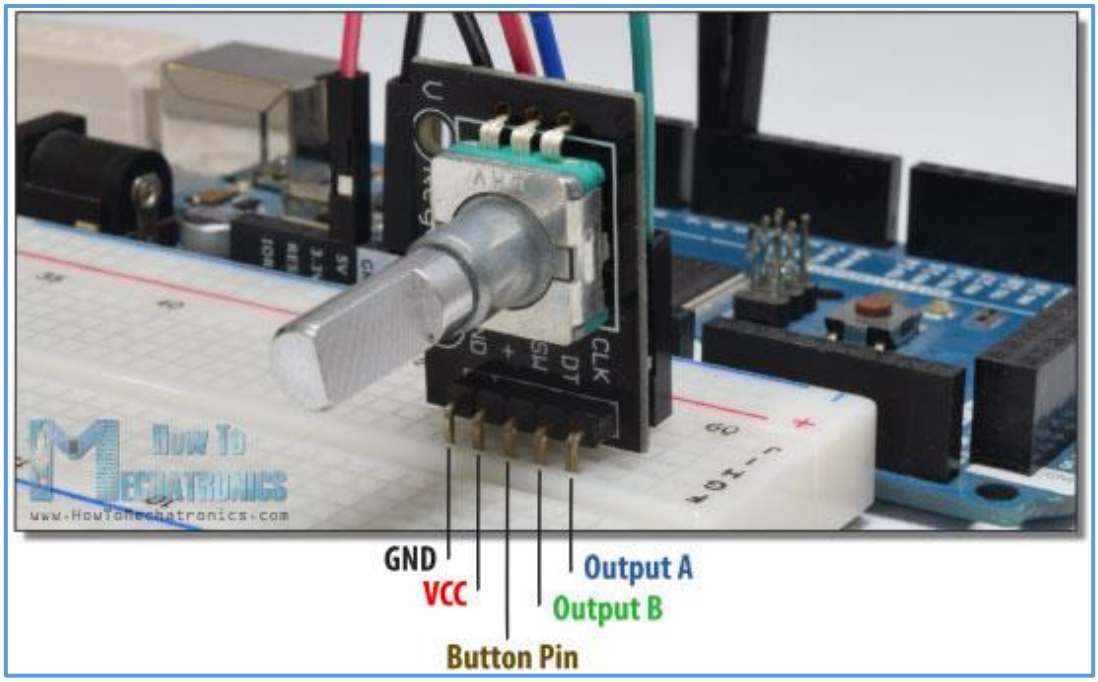
المشفر المطلق:

في هذا النوع يتم تقسيم قرص المشفر البصري إلى قطاعات وعند كل قطاع يكون الخرج عبارة عن قيمة رقمية تمثل موقع محور المحرك بالنسبة لنقطة مرجعية وهذا ما يوضحه الشكل التالي، حيث يظهر لدينا في الشكل التالي مشفر مطلق بأربع بتات أي أن لدينا 16 حالة للخرج وكل حالة تمثل موضع معين، وكلما زادت بتات الخرج حصلنا على دقة أكبر لتموضع محور المحرك المتصل مع قرص المشفر البصري.



التعامل مع المشفر المتزايد عبر لوحة الآردوينو:

ليكن لدينا المشفر البصري المتزايد والذي يحمل الرقم **EC11E** والذي سوف نربطه مع لوحة الآردوينو، والذي يمكن ربطه مع محور محرك لمعرفة عدد دورات المحرك أو سرعته أو تموضع المحرك بالنسبة لنقطة صفرية أو تدويره يدويا للتحكم بمحرك بناء على حركة المشفر**ولهذا المشفر خمسة خطوط هي:**



ما يلزمنا مع لوحة الآردوينو هو قطبي الخرج **B** & **A** والذين سنستفيد منهما في تحديد الجهة والسرعة والموضع وهذا ما سنعرفه عند التعامل مع المكتبية الخاصة بالمشفر الدوراني، أما التغذية فقيمتها 5v.

التعليمة	شرح التعليمة
`#include <Encoder.h>`	**المكتبية الخاصة بتشغيل المشفر المتزايد**
`Encoder myEnc(pin1, pin2);`	**تسمية المشفر وتحديد الأقطاب من لوحة الآردوينو التي سيتم توصيل القطبين B & A من المشفر إليها.**
`myEnc.read();`	**قراءة قيمة المشفر الدوراني.**
`myEnc.write(newPosition);`	**كتابة قيمة للمشفر الدوراني.**

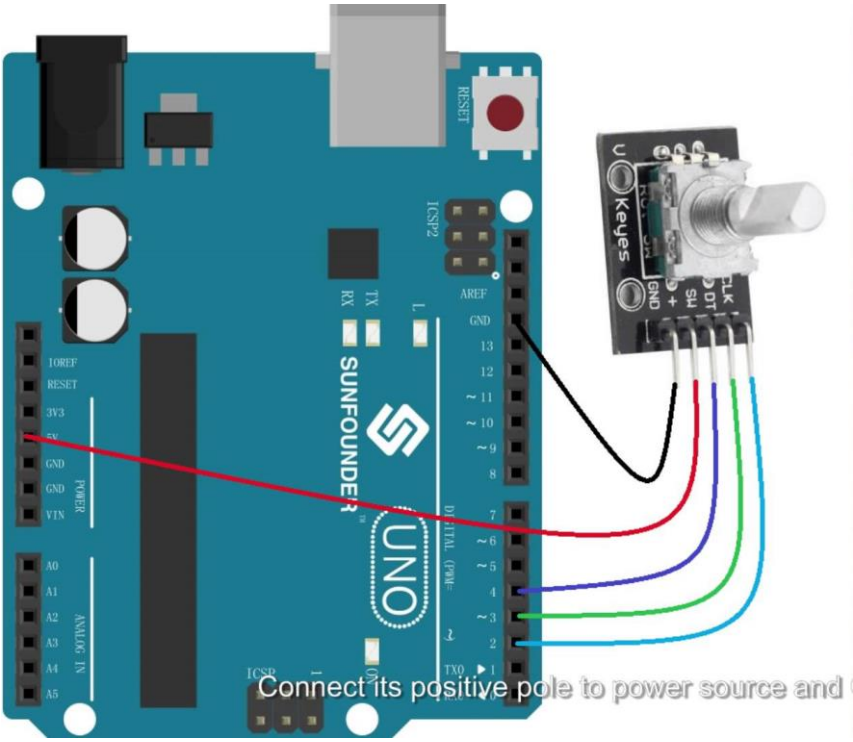
تعتمد المكتبية الخاصة بتشغيل المشفر الدوراني في بنيتها على المقاطعات الخارجية الخاصة بالمتحكم لذلك وعند توصيل القطبين **B** & **A** للوحة الآردوينو يجب توصيلهما لأقطاب المقاطعات الخارجية **Ex. Interrupts**، الجدول التالي يحدد الأقطاب التي يجب استخدامها.

Board	Interrupt Pins	LED Pin (do not use)
Teensy 3.0	All Digital Pins	13
Teensy 2.0	5, 6, 7, 8	11
Teensy 1.0	0, 1, 2, 3, 4, 6, 7, 16	
Teensy++ 2.0	0, 1, 2, 3, 18, 19, 36, 37	6
Teensy++ 1.0	0, 1, 2, 3, 18, 19, 36, 37	
Arduino Due	All Digital Pins	13
Arduino Uno	2, 3	13
Arduino Leonardo	0, 1, 2, 3	13
Arduino Mega	2, 3, 18, 19, 20, 21	13
Sanguino	2, 10, 11	0

يمكن التعامل مع المشفر البصري دون استخدام المكتبية الخاصة به والتي تعتمد على المقاطعات بل استخدام أي قطبين ومعرفة أي قطب يتحسس للإشارة أولا وهذا ما سوف نبينه في المثال التالي.

مثـــال 4 :

ليكن لدينا المشفر الدوراني المتصل مع لوحة الآردوينو نوع UNO كما في الشكل التالي ونريد قراءة القيم الناتجة عنه وتحديد قيم إليه.




```
#define outputA 2
 #define outputB 3
 int counter = 0;
 int aState;
 int aLastState;
 void setup() {
   pinMode (outputA,INPUT);
   pinMode (outputB,INPUT);

   Serial.begin (9600);
   // Reads the initial state of the outputA
   aLastState = digitalRead(outputA);
```

```
 }
 void loop() {
   aState = digitalRead(outputA); // Reads the "current"
state of the outputA
// If the previous and the current state of the outputA are
different, that means a Pulse has occured
   if (aState != aLastState){
// If the outputB state is different to the outputA state,
that means the encoder is rotating clockwise
     if (digitalRead(outputB) != aState) {
       counter ++;
     } else {
       counter --;
     }
     Serial.print("Position: ");
     Serial.println(counter);
   }
   aLastState = aState; // Updates the previous state of
the outputA with the current state
 }
```

في الحلقة اللانهائية يتم فيها ما يلي:

قراءة حالة الدخل A الحالية وإسنادها للمتحول الجديد aState ومقارنتها مع الحالة القديمة aLastState فإن كانت الحالة الجديدة مختلفة عن الحالة القديمة فمعنا ذلك أن هناك تغير في حالة المشفر البصري، بعد ذلك يتم قراءة حالة الدخل B فإن كان مختلف عن حالة الدخل A فمعنى ذلك أن المشفر البصري يدور مع عقارب الساعة وغير ذلك يعني أن المشفر يدور بعكس عقارب الساعة، بعد ذلك يتم إسناد القيمة الجديدة للدخل A لمتحول الحالة القديمة.

PID Control: هو حلقة تحكم شامل بتغذية رجعية وهو شائع الاستخدام في نظم التحكم الصناعي، ويضم هذا النظام ثلاثة عناصر تحكم هي:

Proportional - **I**ntegral - **D**erivative

التفاضل - التكامل - التناسب

هذه العناصر هي المسؤولة عن تصحيح الخطأ الناتج عن الفرق بين القيمة المطلوبة والقيمة المقاسة الحالية.

تحتوي حلقة PID على ثلاثة بارامترات أساسية هي:

- **PV**- **P**rocess **V**ariable: وهي القيمة العملية الحالية.
- **SP**-**S**et **P**oint: وهي القيمة المضبوطة المطلوبة.
- **MV**-**M**anipulated **V**ariable: عملية التحكم بالمتحول المتأثر.

تتلخص حلقة التحكم بثلاث وظائف أساسية هي:

- **وظيفة القياس:** والتي تتم عادة بالحساسات أو أجهزة القياس.
- **وظيفة المقارنة والحساب**: يتم فيها مقارنة القيمة المقاسة **PV** مع القيمة المطلوبة **SP** والتي ينتج عنها قيمة تدعى بقيمة الخطأ **Error**، والذي بدوره يمرر لأحد عناصر مجموعة التحكم(**PID**) ونتيجة المعالجة ترسل إلى قسم التحكم النهائي للعمل به.
- **التحكم النهائي**: تشغيل عناصر تحكم تعمل على تصحيح الخطأ.

تمارين الفصل الثالث

1

ليكن لدينا ثلاثة محركات سيرفو متصلة مع لوحة الآردوينو على الأقطاب 2 , 3 , 4 على التوالي من لوحة الآردوينو، كما يوجد كباس لحظي متصل مع القطب 5 من لوحة الآردوينو، والمطلوب: كتابة كود برمجي يسمح للمستخدم إدخال رقم المحرك المراد التحكم به وموقعه وذلك عند الضغط على الكباس اللحظي.

2

لدينا محركين نوع Brushless متصلين مع الأقطاب 13 , 12 كما يوجد لدينا قبضة تشابهية محور X منها متصل مع القطب A0 من لوحة الآردوينو، والمطلوب كتابة كود برمجي للتحكم بسرعة المحركين بحيث أن التحريك من المنتصف ونحو اليمين للتحكم بسرعة المحرك الأول بينما من المنتصف ونحو اليسار للتحكم بسرعة المحرك الثاني، أما المنتصف فهو حالة إيقاف للمحركين.

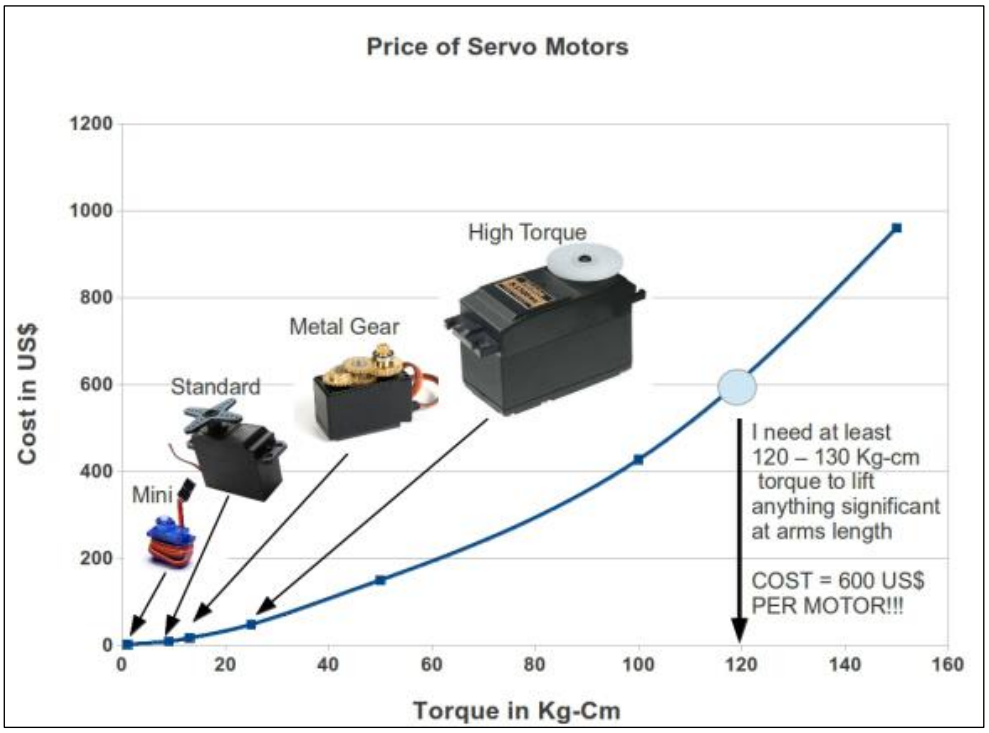
3

لدينا محرك خطوي يتحكم به عبر دارة القيادة A4988 والتي بدورها تتصل مع لوحة الآردوينو عبر الأقطاب 10 للأمر STEP والقطب 11 للأمر DIR ،كما يوجد كباسين لحظيين متصلين مع الأقطاب 3 , 2 والمطلوب كتابة كود للتحكم بجهة دوران المحرك يمينا أو يسار حسب ضغط المفاتيح وبخطوة ثابته.

المشروع الفصل: تصميم محرك Servo من محرك DC

فكرة المشروع:

تقوم فكرة المشروع على بناء محرك **Servo** انطلاقا من محرك تيار مستمر **DC**، إذ أن محركات **Servo** ذات العزوم العالية تعتبر غالية الثمن فالمحركات التي تتراوح عزومها بين 20 ~ 30 Kg-cm تكون أسعارها قريبة من **30$** أما لو أردنا محرك بعزم قريب من القيمة 120 Kg-cm فالسعر سوف يرتفع لما يقارب **600$** وبالتالي فالتكلفة مرتفعة جدا، لذلك قمنا بطرح هذا المشروع للحصول على محرك بعزم قوي وأداء قريب جدا من أداء محركات **Servo** المتوفرة في الأسواق.



- ✓ يقاس العزم **Torque** بواحدة **Kg-cm** أي الوزن المعطى بواحدة **Kg** الذي يستطيع المحرك بذراع تبعد عن محور الحركة مقدار **1cm** تحريك هذا الوزن، أو بواحدة N/cm.

بالعودة لآلية بناء محرك **servo** وجدنا خلال دراسة هذا المحرك أنه يحتوي في بنيته على محرك DC مع مسننات لتحويل السرعة لعزم، كما يتضمن دارة قيادة لهذا المحرك من حيث السرعة والاتجاه والتي تتم بدارة **H-Bridge** ، أما المهمة الأهم فهي وجود تغذية عكسية لمعرفة مكان المحرك وجهة دورانه لضمان وصوله للمكان المحدد وتصحيح الخطأ، أما مصدر هذه التغذية المرجعية فيمكن أن يكون إما **مقاومة متغيرة** مربوطة مع محور المحرك النهائي وكل قيمة لهذه المقاومة تعبر عن موضع محدد، أو **مشفر بصري** الدوار **Rotary Encoder** متصل مع محور المحرك لمعرفة زاوية الدوران من دقة المشفر نفسه، ولكل طريقة من الطريقتين السابقتين مساوئها ومحاسنها ، سنستعرض الطريقتين بإذن الله بتطبيق عملي لكلتا الطريقتين، أما القاسم المشترك بين الطريقتين:

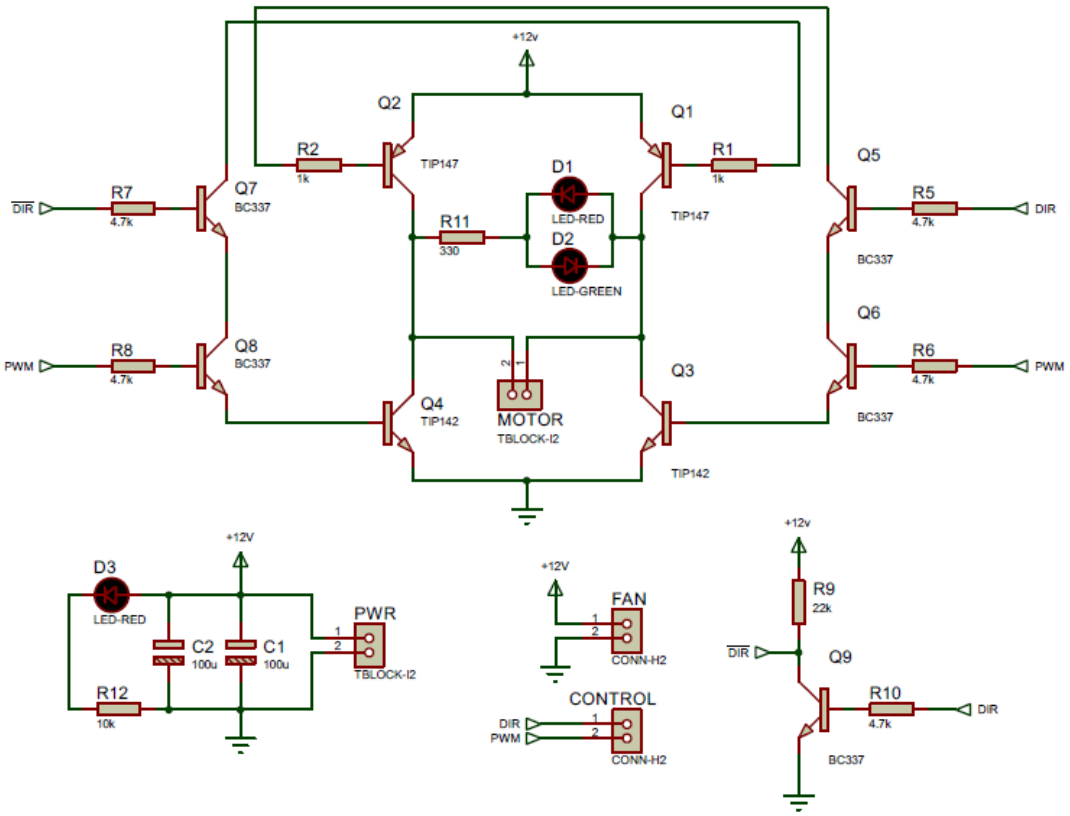
- » محرك تيار مستمر DC مع علبة سرعة (في حال الحاجة لتحويل السرعة لعزم).
- » دارة قيادة للمحرك **H-Bridge** للتحكم بجهة دوران المحرك وسرعته (يتوفر العديد من موديولات **H-Bridge**).
- » مصدر **التحكم بأوامر** حركة المحرك واتجاهه (قد يكون عبارة عن كباسات لحظية أو قبضة تشابهية للتحكم باتجاهه وسرعته).

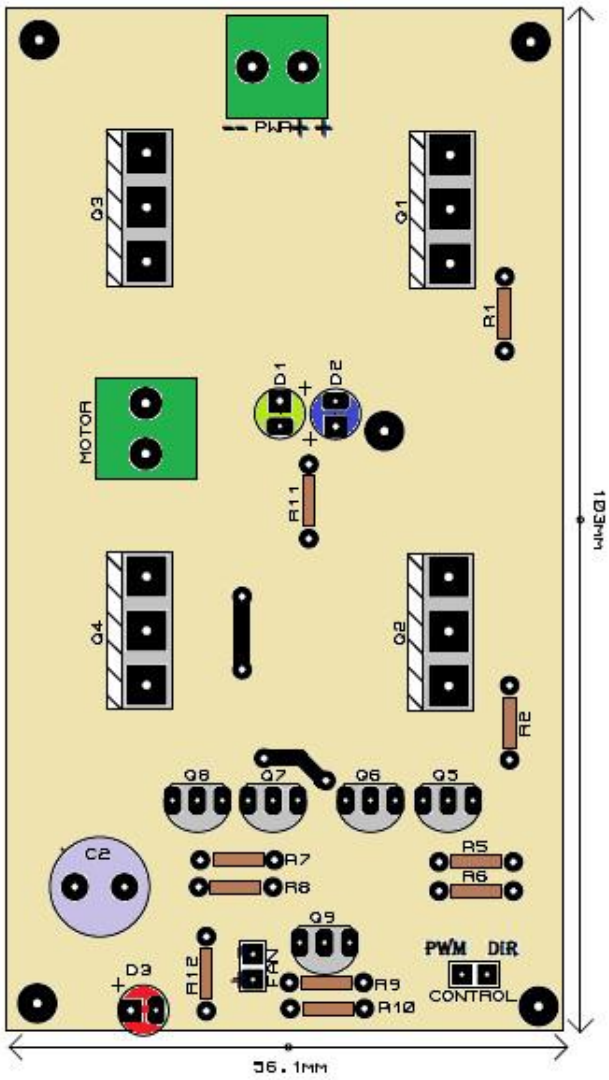
بعد استعراض القاسم المشترك بين طريقتي التحكم بقي أن نتحدث قليلا عن دارة القيادة الخاصة بمحرك التيار المستمر والتي تعرف بـ **H-Bridge** أو جسر H للتحكم بجهة دوران المحرك وسرعته (التحكم بالسرعة يتم عبر التحكم بعرض النبضة **PWM**).

دارة القيادة لمحرك التيار المستمر H-Bridge:

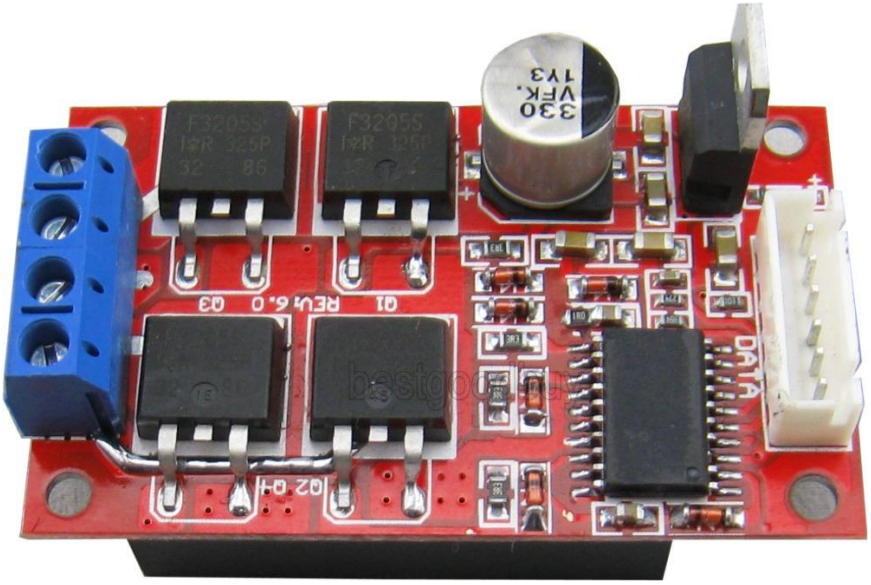
تتميز دارة القيادة هذه بالميزات التالية:

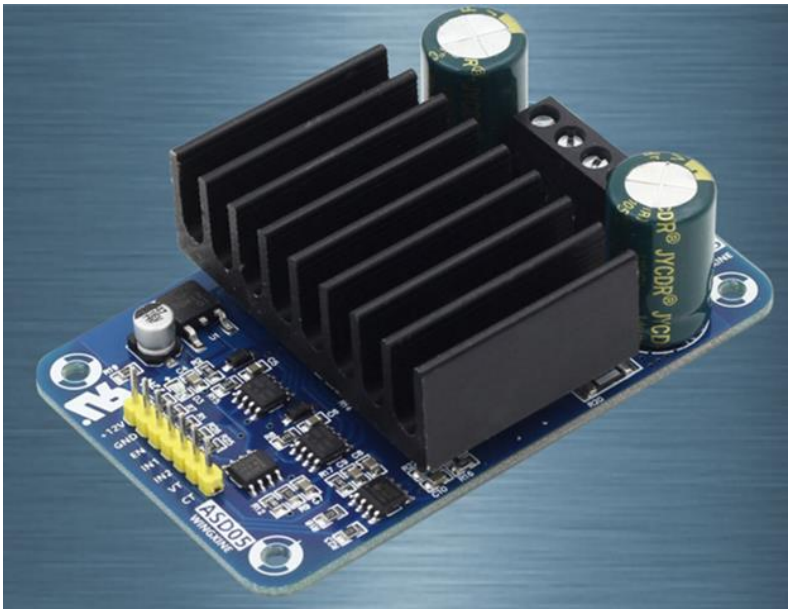
- استطاعة تشغيل تصل حتى **125W** بأمبير أعظمي 10A وجهد أعظمي 100V.
- سهلة البناء والتجميع وسهلة التحكم والقيادة.
- تتألف مداخل التحكم من مدخلين مدخل للتحكم بجهة الدوران **DIR** ومدخل للتحكم بالسرعة (**PWM**).
- تحتوي على مروحة تبريد لضمان عمل عناصر الدارة في الحرارة المناسبة.





يتوفر في الأسواق العديد من دارات قيادة محركات التيار المستمر وتتميز بعزوم كبيرة وأمبيرات مرتفعة وتختلف فيما بينها في عدد الأقطاب التي تحتويها للتحكم فمنها من يحتوي قطبين لتحديد الاتجاه وقطب ثالث لتحديد السرعة بينما أنواع أخرى تحتوي على قطبين فقط لتحديد الاتجاه وفي نفس الوقت لتحديد السرعة على كل إتجاه وهكذا .... نستعرض أشهرها.

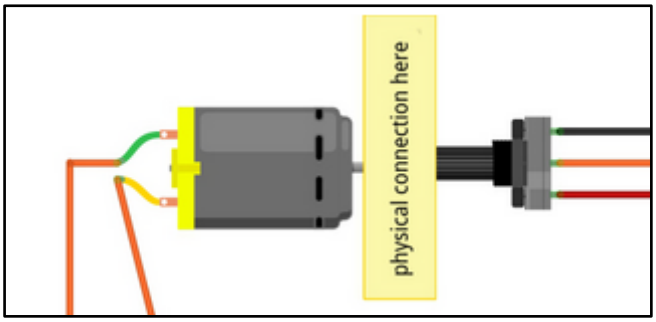




بعد الانتهاء من بناء دارة القيادة نكون قد أنهينا أغلب العمل المتعلق ببناء المشروع وبقي علينا ربط المقاومة المتغيرة أو المشفر البصري مع محور المحرك وهنا يبقى عندنا القسم البرمجي الذي سيكون عنوان فقرتنا القادمة.

أولا: التغذية العكسية المعتمدة على المقاومة المتغيرة:

وفي هذا النمط نقوم بوصل مقاومة متغيرة دورانية قيمتها 10kR مع محور الحركة الدوار النهائي للمحرك وبالتالي فإن أي قيمة لهذه المقاومة يقابلها موقع مختلف لمحور المحرك.



أما الكود البرمجي للتحكم فيكون بالشكل التالي على اعتبار أن أقطاب التحكم تأخذ الأرقام التالية:

مصدر نبضات **PWM** تأخذ الرقم **6**، أما أقطاب التحكم **بجهة دوران** المحرك فتأخذ الأرقام **4 & 5** ويمكن استخدام قطب تحكم واحد للاتجاه فقط، كما يوجد مدخلين للقيم التشابهية الأول **A0** لقراءة قيم **المقاومة** المربوطة على **محور المحرك** والثاني **A1** فيأخذ قيم **التحكم بموقع المحرك** وتحركه نحو الموقع المطلوب.

- تعريف أقطاب التحكم بالاتجاه وعرض النبضة.
- مقاومة متصلة مع محور المحرك.
- مقاومة للتحكم بالمحرك.

```
#define motor_p1          4
#define motor_p2          5
#define pwmPin            6
#define resis_motor       A0
#define resis_control     A1

int currentAngle;
int requiredAngle;
int errorAmount;
```

- currentAngle: متحول لقراءة قيمة المقاومة على محور المحرك.
- requiredAngle: قيمة المقاومة المطلوبة.
- errorAmount: الفارق بين القيمة المطلوبة والقيمة الحالية.
- remappedErrorAmount: المتحول الذي على أساسه سوف تكون سرعة الانتقال للموقع المطلوب فهو المتحكم بعرض النبضة.
- acceptableError: نعومة الانتقال .

تهيئة المخارج ونافذة الاتصال التسلسلي UART.

```
int remappedErrorAmount;
byte acceptableError = 4;




void setup()
{
  Serial.begin(9600);
  pinMode(motor_p1, OUTPUT);
  pinMode(motor_p2, OUTPUT);
  pinMode(pwmPin, OUTPUT);
}

void loop()
{
  readAndConditionAngle();

readAndConditionRequiredAngle();
  if (currentAngle<requiredAngle)
  {
    digitalWrite(motor_p1, HIGH);
    digitalWrite(motor_p2, LOW);
  }
  if (currentAngle>requiredAngle)
  {
    digitalWrite(motor_p1,LOW);
```

حلقة البرنامج الأساسية: استدعاء تابع قراءة قيمة المقاومة المطلوبة. استدعاء تابع قراءة قيمة المقاومة الحالية.	digitalWrite(motor_p2,HIGH); } calculateErrorAmount (currentAngle,requiredAngle); analogWrite (pwmPin,remappedErrorAmount); Serial.println(currentAngle); }
تحريك المحرك تبعا لحالة الفارق بين المقاومتين.	void calculateErrorAmount(int currentAngle,int requiredAngle) { errorAmount = abs(currentAngle-requiredAngle); remappedErrorAmount = map(errorAmount,0,1000,0, 255); if (remappedErrorAmount < acceptableError) { remappedErrorAmount=0; } }
	void readAndConditionAngle() { currentAngle = analogRead(resis_motor); if (currentAngle<200)

استدعاء تابع تحديد سرعة انتقال المحرك للموقع المطلوب. طباعة قيمة الزاوية الحالية.	{ currentAngle=200; } if (currentAngle>800) { currentAngle=800; } } void readAndConditionRequiredAngle() { requiredAngle = analogRead(resis_control); if (requiredAngle<200) { requiredAngle=200; } if (requiredAngle>800) { requiredAngle=800; } }

لكي يعمل الكود بشكل صحيح يجب علينا أولا أن نستخدم محرك DC بعلبة سرعة بحيث تكون سرعة المحرك قابلة للتحكم بها (يجب ألا تزيد سرعة المحرك عن 100 rpm)، كما يجب ضبط المقاومة المتغيرة المتعلقة بحركة محور المحرك بين قيمتين هما مجال عمل المحرك، كذلك ضبط الموجب والسالب للمحرك للتحريك نحو الجهة المطلوبة.

ثانيا: التغذية العكسية المعتمدة على مشفر دوراني متزايد:

وفي هذا النمط من التحكم يتم ربط مشفر دوراني مع محور المحرك ويتم معرفة اتجاه دوران المحرك وموقع المحرك من خلال النبضات القادمة من المشفر وبالتالي التحكم الكامل بالمحرك.

في هذا النمط يتطلب توفر الأقطاب التالية على لوحة الآردوينو:

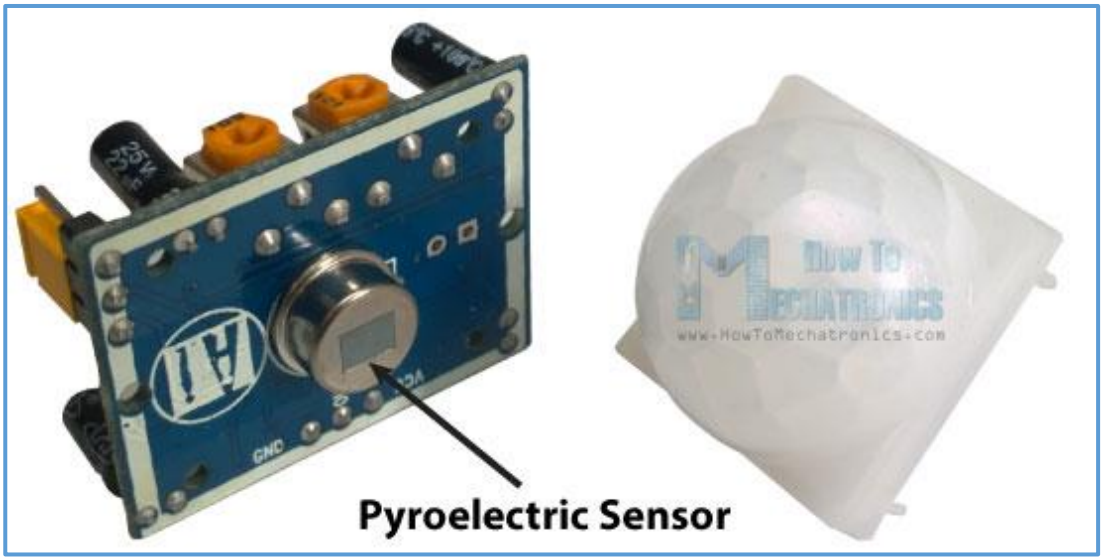
- أقطاب قيادة المحرك المتصل مع دارة القيادة الجسرية وهي أقطاب التحكم في الاتجاه **DIR** وقطب التحكم بالسرعة عبر التحكم بعرض النبضة **PWM**.
- أقطاب قراءة النبضات الواردة من المشفر البصري وهي قطبين **A & B** وتوصل مع أقطاب المقاطعة الخارجية **INT1 & INT2**.

وظيفة الكود البرمجي قراءة عدد النبضات القادمة من المشفر البصري وحساب عددها زيادة أو نقصان حسب جهة دوران المحرك، ومن خلال معرفة عدد النبضات التي تم حسابها يتم معرفة الموقع الحالي للمحرك حيث أن كل نبضة يقابلها زاوية ميل لمحور المحرك، أما الكود البرمجي فسنتركه لإصدارات لاحقة أو ننشره على قناتنا على التيليغرام EPTEKAR

الفصل الرابع:
حساسات الحركة والمسافة

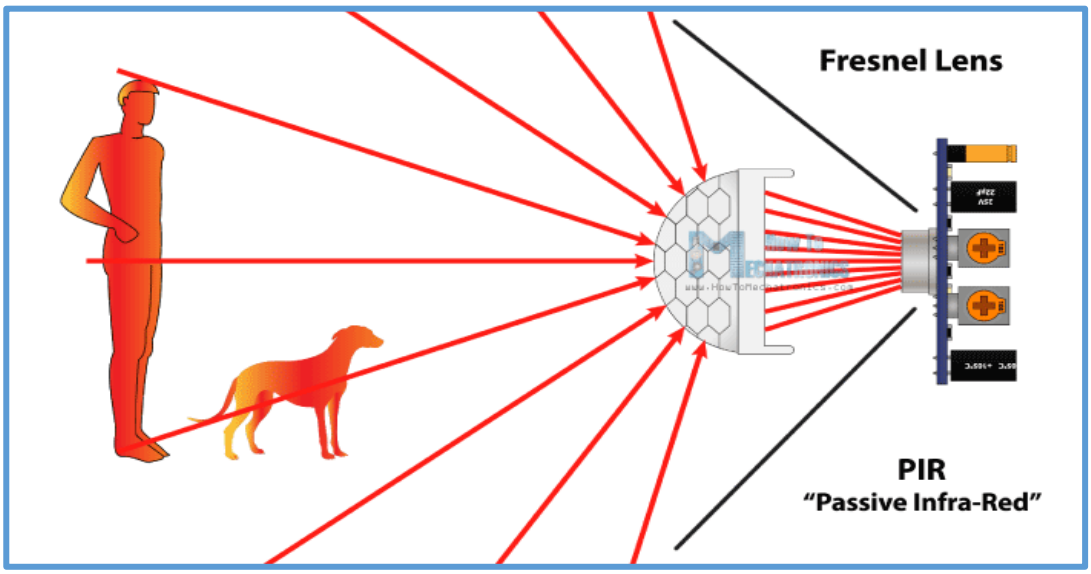
حساس الحركة PIR.

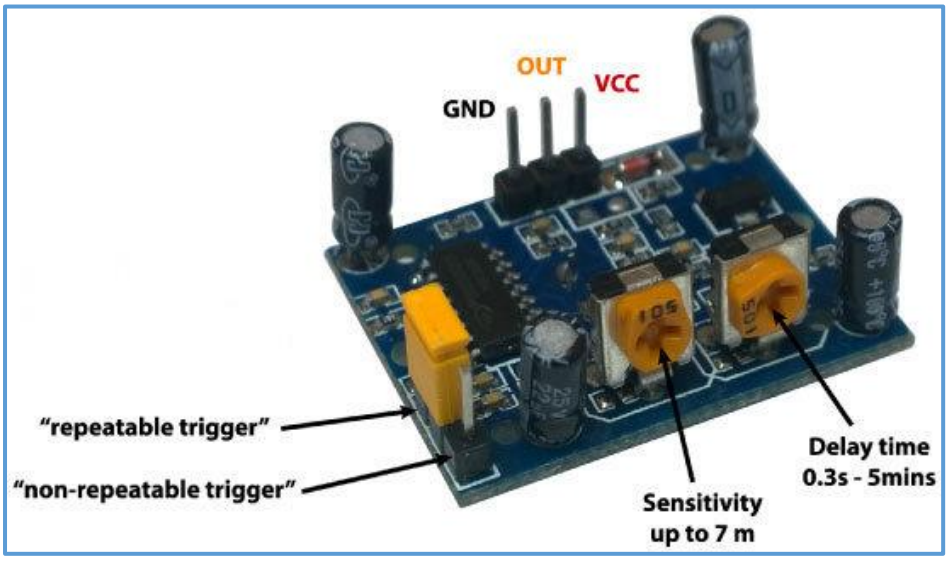
في هذه الفقرة من الكتاب سنتحدث عن حساس الحركة PIR: Passive Infra-Red حيث يستخدم هذا الحساس لاستشعار الحركة في البيئة المحيطة ضمن مجال عمل الحساس ويتميز بتكلفته المنخفضة، استهلاك قليل للطاقة الكهربائية، سهل الاستخدام.



يحتوي هذا الحساس في بنيته على خلية حساسية للتغير الحراري **Pyroelectric** والتي تقوم بتحويل التغير الحراري الذي يحدث ضمن مجال الحساس لجهد كهربائي، حيث يحتوي الحساس على قطاعات عمل فعند تحرك جسم أمام الحساس سيولد اختلاف بين قطاعات العدسة الحساسة والذي بدوره يولد خرج رقمي، أما القبة البلاستيكية التي لها شكلها الخاص حيث تحتوي على تقسيمات على سطحها الخارجي بشكل متساوي تتوضع أمام عدسة الحساس وظيفتها تقسيم المجال أمام الحساس لقطاعات متعددة والتي تساعد على تركيز الأشعة تحت الحمراء الناتجة عن الأجسام

التي أمام الحساس في بؤر مخصصة على عدسة الاستشعار الحراري للحساس، **والشكل التالي يبين آلية العمل:**





بالعودة للموديول الخاص بحساس الحركة PIR نجد أنه يضم في بنيته:

- ثلاثة أقطاب إثنين منها هي أقطاب التغذية (GND - **VCC**) والقطب الثالث هو قطب الخرج (**OUT**) والذي يعطي واحد منطقي عند استشعار الحساس لأي جسم يمر أمامه.

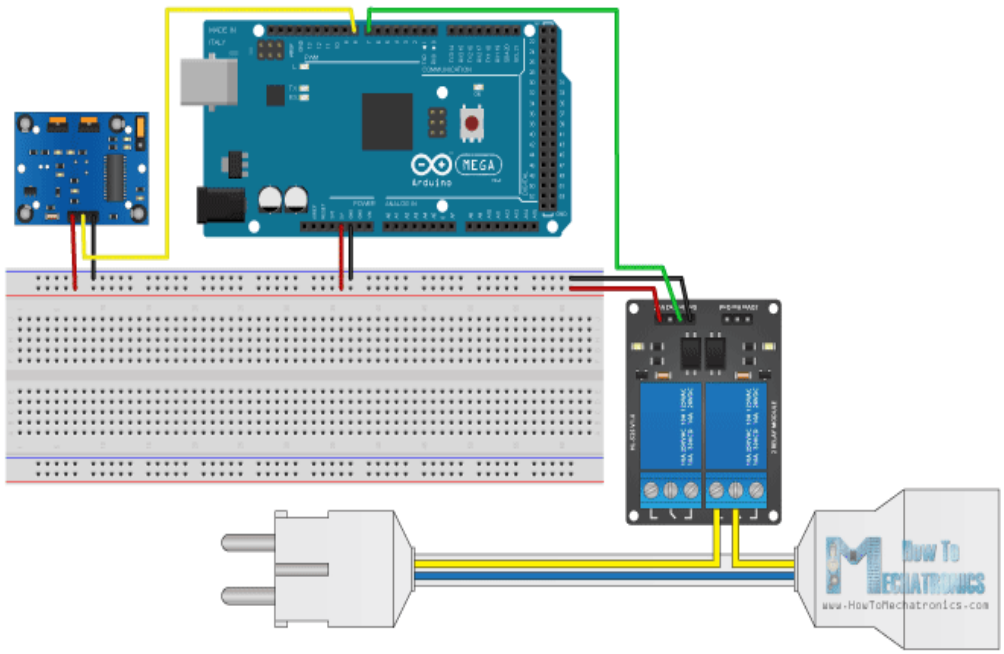
- **مقاومة متغيرة أولى** والتي يتم من خلالها تحديد وقت التأخير الزمني (Delay Time) للخرج عند استشعار الحساس ومجال التأخير هو ضمن المجال **0.3s ~ 5min**.
- **مقاومة متغيرة ثانية** لتحديد مدى الحساسية ويصل حتى **7m**.
- ثلاثة أقطاب يتم من خلالها تحديد طبيعة عمل الخرج عند استشعار الحساس ففي **النمط الأول** (non-repeatable trigger) يعكس الحالة بعد انتهاء وقت التأخير الزمني أي يعود للصفر منطقي، بينما **النمط الثاني** (repeatable trigger) فيستمر بإعطاء خرج مادام هناك جسم يتحرك أمام الحساس.

من خلال التجريب وجدنا أن حساس الحركة **PIR** يتصف بعدة عوامل تجعل استخدامه يتطلب عدة ضوابط وذلك للحصول على أفضل النتائج، فمثلا يجب أن يستخدم في أماكن لا تحتوي مصادر حرارية أمام الحساس مثل المدفئة أو الأضواء التي تعطي حرارة وغير ذلك من الأجسام، لأن ذلك سيؤثر على عملية ضبط الحساس كما يجب اختيار التأخير الزمني بما يتناسب مع بيئة العمل كذلك الأمر بالنسبة لضبط مجال استشعار الحساس.

بالمجمل فإن الحساس يعتبر رخيص السعر وسهل الاستخدام والبرمجة لكن عملية الضبط للحصول على أفضل النتائج تعتبر صعبة نسبيا فلكل مكان عمل مساوئ يجب مراعاتها وخاصة المصادر الحرارية.

مثـــال 1 :

لنكتب كود برمجي للاستفادة من حساس الحركة PIR لتشغيل ضوء متحكم به عبر ريليه خارجية عند مرور جسم أمام الحساس.

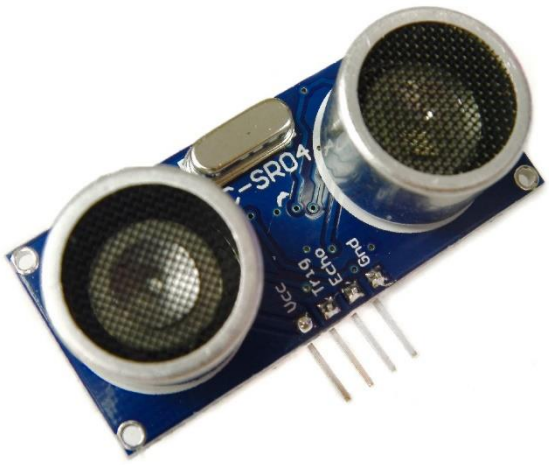



```
int pirSensor = 8;
int relay_out= 13;
int sensorVal;


void setup() {
  pinMode(pirSensor,INPUT);
  pinMode(relay_out,OUTPUT);
```

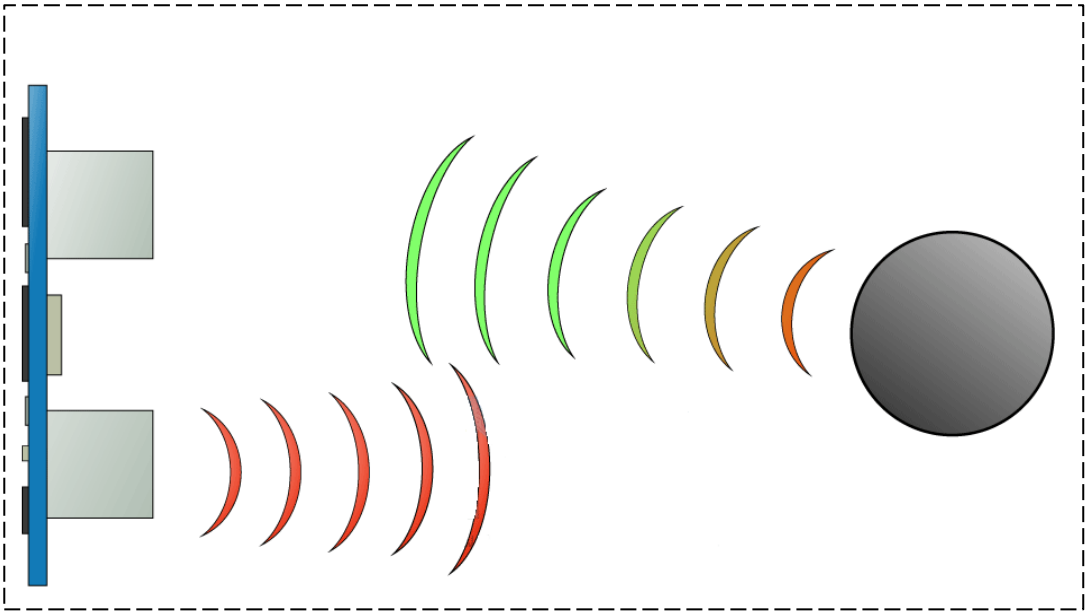
```
}

void loop() {
  sensorVal=digitalRead(pirSensor);

  if(sensorVal==HIGH){
    digitalWrite(relay_out,LOW);



  }
  else{
  digitalWrite(relay_out,HIGH);
  }

}
```

الكود بسيط جدا حيث يتم التعامل مع الحساس كأنه مفتاح ويتم قراءته إعتمادا على ذلك.

حساس الأمواج فوق الصوتية Ultrasonic Sensor



في هذا البحث سوف نستخدم حساس قياس المسافات بالأمواج فوق الصوتية والذي يأخذ الرمز **HC-SR04**، يعتمد هذا النوع من الحساسات على ارسال أمواج فوق صوتية عبر الهواء وعند اصطدام هذه الأمواج بأي جسم أو عائق تنعكس وتعود للحساس ومن خلال حساب الزمن الذي تستغرقه الموجة في عملية الذهاب والإياب يتم معرفة المسافة التي يبعدها الجسم عن الحساس وذلك على اعتبار أن سرعة الأمواج الصوتية في الهواء ثابتة.



بالعودة لحســـاس الأمواج فوق الصوتيـــة **HC-SR04** فقبل التعامـــل معه يجب أن يـــكون لدينا صـــورة واضحة عن البنية والخصـــائص التي يتمتع بها هذا الحساس حيث يضم هذا الحســـاس **أربعة أقطاب هي :**

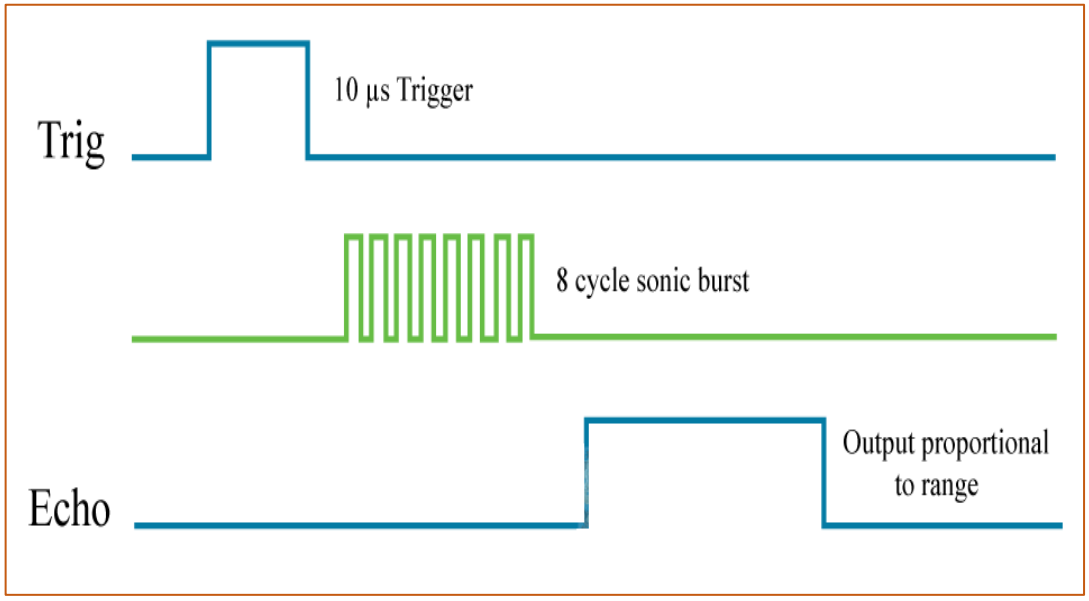
- قطب التغذية الموجبة **Vcc**.
- قطب التغذية السالبة **GND**.
- قطب القدح **Trigger Pin** الذي يعطي أمر للحساس بإرسال الأمواج فوق الصوتية عبر جهاز الإصدار الصوتي المثبت على سطح الحساس.
- قطب استقبال الموجة المنعكسة **Echo pin**.

أما الميزات التصميمة لهذا الحساس فهي:

- جهد تغذية **Vcc = 5 volt** واستهلاك منخفض للتيار يصل حتى **15 mA**.
- تردد عمل الحساس **40 kHz**.
- مجال حساب المسافة للحساس تنتمي للمجال **2cm ~ 4m**.
- زاوية قياس الحساس **15 درجة**.
- نبضة قدح الحساس **10uS** بمنطق TTL.
- خرج نبضة الاستقبال متناسبة مع المسافة المدى.

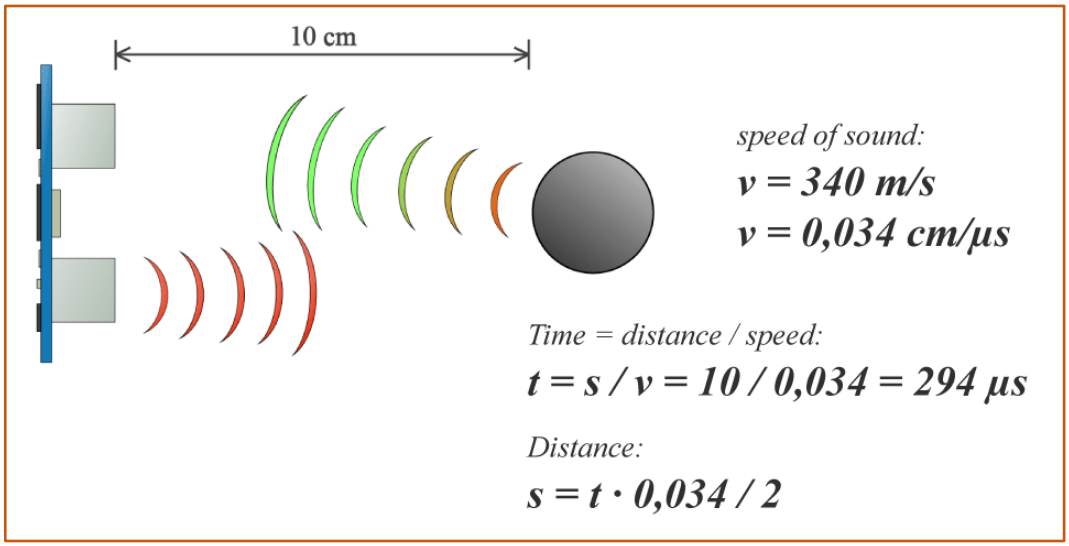
آلية عمل حساس الأمواج فوق الصوتية:

يتم ارسال **نبضة القدح** عبر الآردوينو إلى قطب القدح للحساس **Trigger Pin** بشرط أن تكون عرض النبضة بقيمة **10us**، فيرسل الحساس **ثمانية نبضات** متتالية من الأمواج فوق الصوتية بتردد 40kHz عبر مجهار صوتي مثبت على الحساس، وبعد انعكاس الأمواج وعودتها للحساس يتولد على قطب الخرج **Echo Pin** نبضة بطول معين هذا الطول يتناسب مع المسافة التي قطعتها الموجة في الهواء منذ خروجها من الحساس إلى أن اصطدمت بالجسم المقابل للحساس وعادت للحساس.



ليكن لدينا جسم يبعد عن الحساس مسافة 10cm كيف يتم حساب المسافة التي يبعدها الجسم عن الحساس...؟؟؟؟

تعتبر سرعة الأمواج الصوتية في الهواء ثابتة **340 m/s** (تتغير هذه السرعة بتغير العوامل الجوية كالحرارة وسرعة الرياح الارتفاع عن سطح البحر لكن سيتم إهمال هذه العوامل لصغر مدى عمل الحساس) هذه السرعة تكون وبتغيير واحدة قياس الزمن للميكرو ثانية والمسافة للسنتيمتر 0.034 cm/us.



يعطى قانون السرعة بالشكل (المسافة * الزمن = السرعة) ومن هذا القانون فإن السرعة معلومة لدينا وهي سرعة الصوت في الهواء وكذلك الزمن نحصل عليه من خرج الحساس (مسافة الذهاب والغياب لذلك نقسم القيمة التي نحصل عليها من خرج الحساس على **2** للحصول على مسافة الذهاب فقط) ويكون بواحدة **us** بقي مجهول واحد في العلاقة وهو المسافة وبالتالي نكون قد حسبنا المسافة التي يبعدها الجسم عن الحساس.

Distance = Time * speed /2

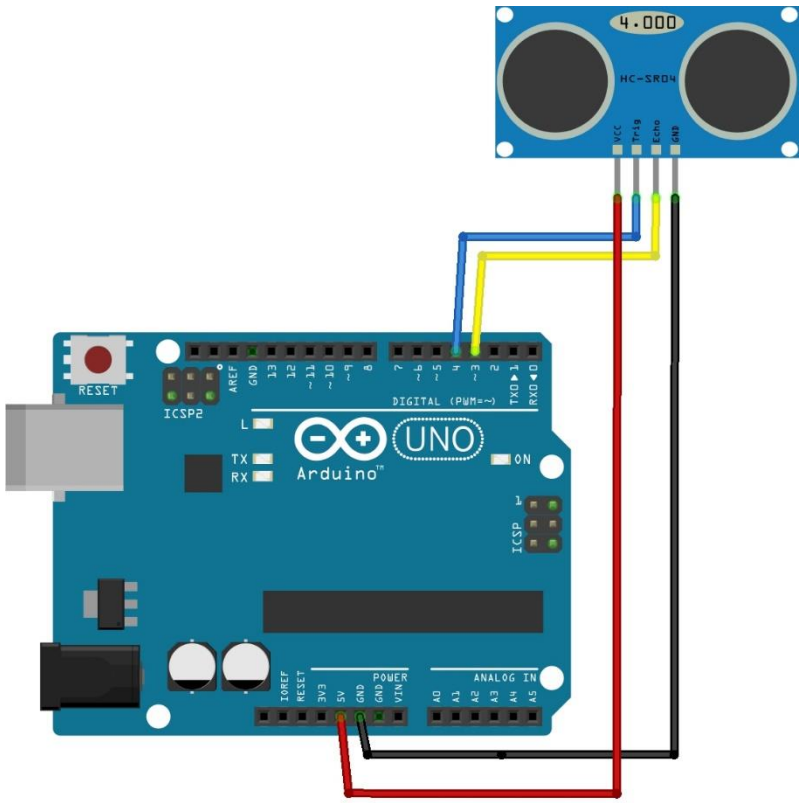
بالعودة لبيئة التطوير **Arduino IDE** ما الذي نحتاجه من بيئة التطوير هذه للتعامل مع حساس الأمواج فوق الصوتية **HC-SR** واستثماره بالشكل الأمثل، بالعودة لطريقة عمل هذا الحساس نجد أننا بحاجة لتابع يقوم بقراءة عرض النبضة الناتجة عن القطب **Echo** من الحساس والتي يعبر عرضها عن المسافة التي اجتازتها الموجة من الحساس وحتى عادت إليه، التعليمة التي سوف نستخدمها في قياس عرض النبضة الداخلة للوحة الآردوينو من الحساس هي تعليمة `pulseIn` والتي سبق وأن تطرقنا لها في الفصل الأول من الكتاب.

شرح التعليمة	التعليمة
قراءة النبضة الداخلة للقطب pin وتحديد الحالة Mode التي سيتم قراءتها على القطب (HIGHT - LOW) وإسناد الوقت الذي استغرقته النبضة إلى المتحول V بواحدة us	`v = pulseIn(pin, Mode);`

لا ننسى أن التابع يتميز `pulseIn` بالقدرة على قياس عرض النبضة المطلوبة من أي قطب في الحالتين (**HIGH - LOW**) وعرض نبضة ينتمي للمجال **10uS ~ 3min**.

مثـــال 2 :

لنكتب كود برمجي نقيس من خلاله المسافة التي يبعدها جسم ما عن حساس المسافة HC-SR04 وطباعة المسافة إلى المنفذ التسلسلي.




```
int TrigPin = 4;
int EchoPin = 3;

long duration;
int distance;

void setup() {
pinMode(TrigPin,OUTPUT);
pinMode(EchoPin,INPUT);
```

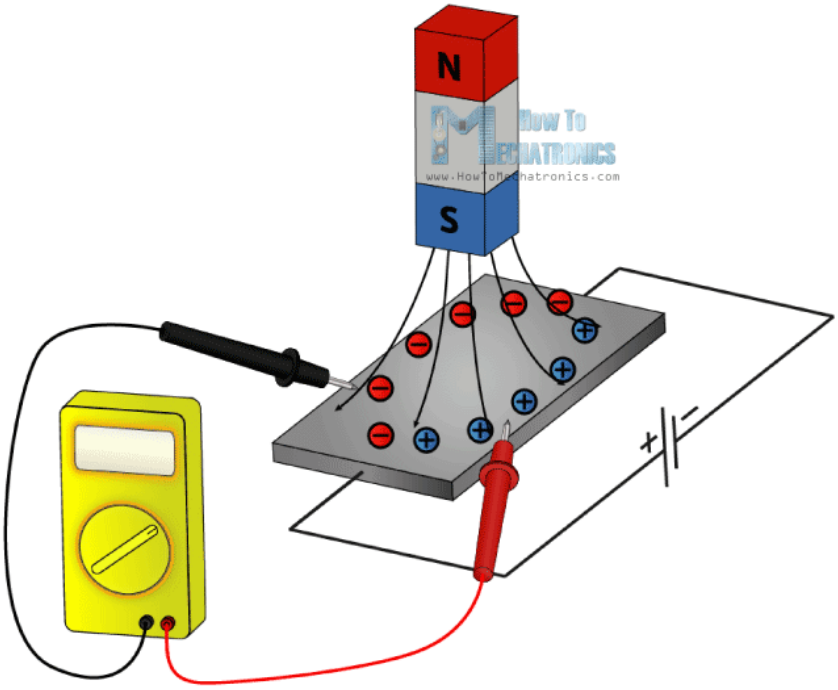
```
Serial.begin(9600);
}

void loop() {
  //Clears the TrigPin
  digitalWrite(TrigPin,LOW);
  delayMicroseconds(2);

  digitalWrite(TrigPin,HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin,LOW);

  duration = pulseIn(EchoPin,HIGH);

  distance=duration*0.034/2;
  Serial.print("Distance");
  Serial.println(distance);

}
```

بعد تعريف الأقطاب المطلوبة لعمل الحساس وهي قطب القدح TrigPin وقطب قراءة الصدى EchoPin يتم الدخول في حلقة البرنامج الرئيسي والتي يتم فيها تفعيل نبضة القدح لمدة 10us ثم إطفاء نبضة القدح، بعدها يتم قراءة قيمة النبضة الواردة من قطب الصدى وإدخال القيمة المقروءة في عملية تحويل رياضي للتوافق بين الواحدات ومن ثم التعويض في قانون حساب المسافة مع قسمة الناتج على 2 نحصل على القيمة المطلوبة.

حساس أثر هول Hall Effect sensor.

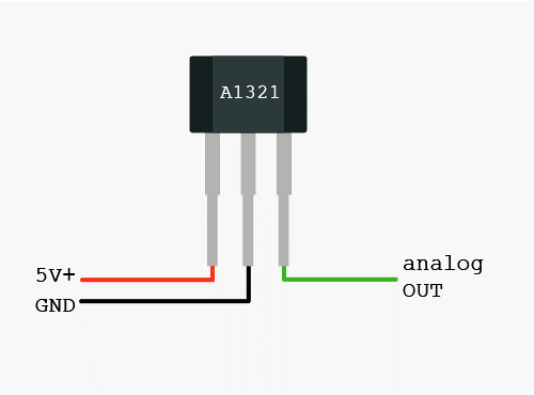
ما هو أثر هول Hall Effect ...؟؟؟

هو ميل حاملات الشحنة بنوعيها سواء كانت موجبة أو سالبة للانزياح نحو الأطراف في الموصلات الكهربائية بسبب **المجال المغناطيسي** المطبق أو المتعرض له، كما ينشأ عن ذلك **فرق جهد** (يسمى جهد هول) بين الأقطاب المتعاكسة في موصل كهربائي تعتمد قطبيته على إشارة هذه الحاملات.



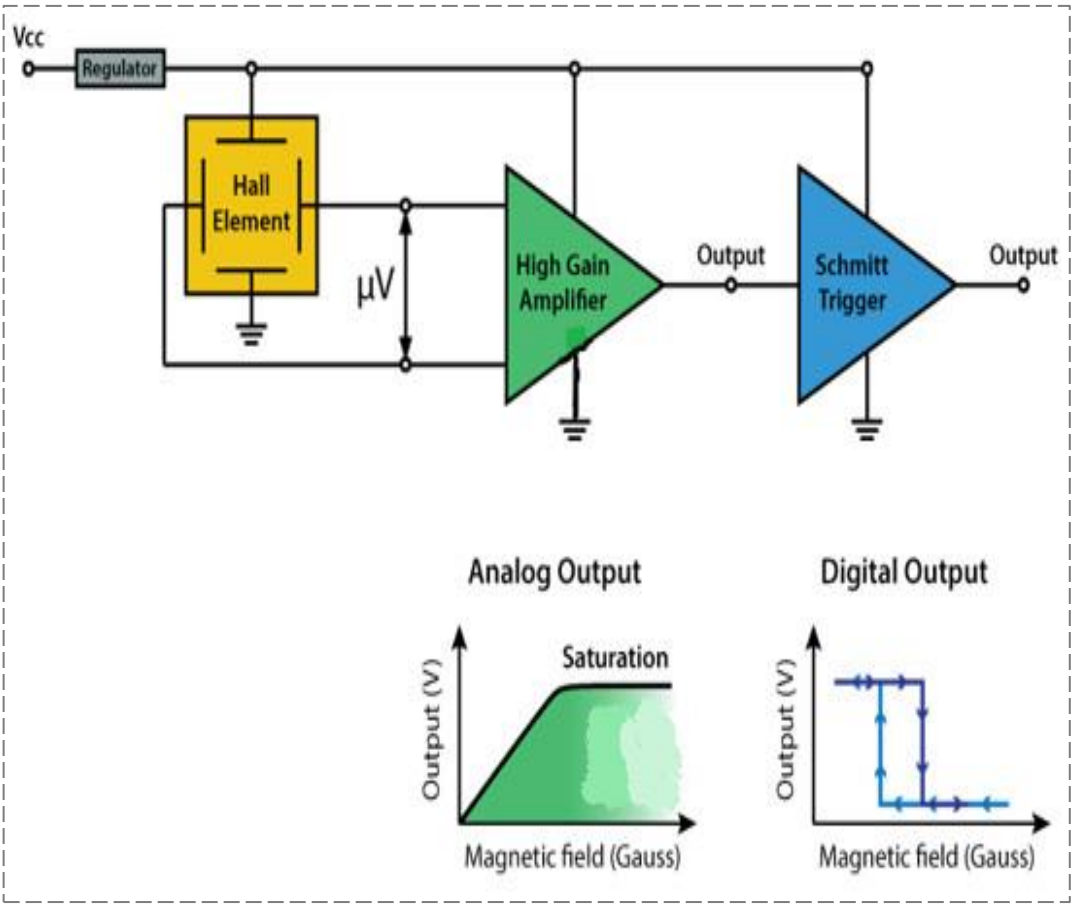
تم الاستفادة من ظاهرة أثر هول للحصول على العديد من التطبيقات والحساسات ومن ضمن الاستخدامات الشهيرة لأثر هول هو الحصول على الحساس التحريضي أو حساس أثر هول، الحساس التقاربي، البوصلات وغير ذلك من التطبيقات المتعددة.

الحساس التقاربي ..... البنية والاستخدام:

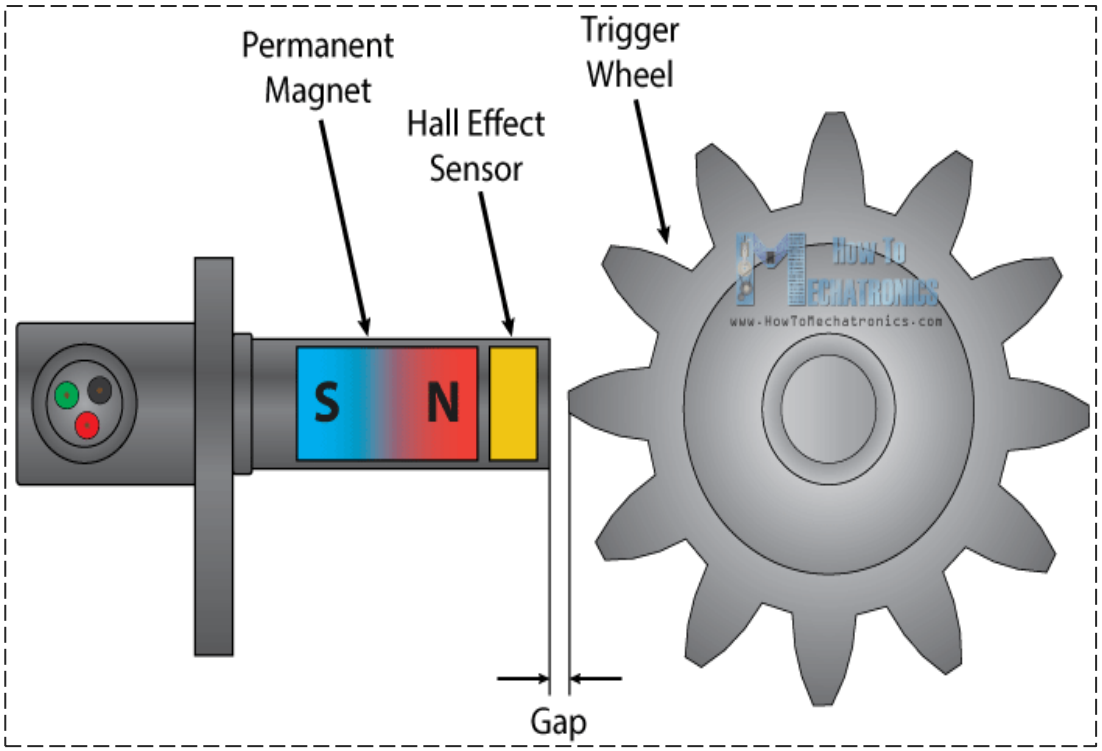


بالعودة لأثر هول سندرس استخدام هذه الظاهرة في الحساس التقاربي والذي يستخدم في قياس بعد جسم معدني عن الحساس (ضمن مجال عمل صغير جدا لا يتجاوز البضع مليمترات فقط) ويتوفر منه نوعان من حيث طبيعة الخرج فمنها ما هو خرجه رقمي ومنها ما خرجه تشابهي ولكل استخدامه، ما يهمنا هو الخرج الرقمي حيث سنستخدم الحساس كمفتاح رقمي للتحكم بأوامر معينة تبعا لحالة الحساس

بالمجمل فإن هذه الحساسات تحوي في بنيتها على منظم جهد خاص بالحساس كما يضم مواد حساسة **لأثر هول** ينتج عنها جهد كهربائي صغير جدا (من واحدة مايكرو فولط) يتم بعد ذلك إدخال هذا الجهد إلى **مضخم جهد** ذو عامل ربح كبير يعطي على خرجه جهد تشابهي يعتمد هذا الجهد على الحقل المغناطيسي المطبق على الحساس، أما في الحساسات التقاربية التي خرجها رقمي فيتم إضافة مرحلة إضافية بعد الخرج التشابهي لتحويل الجهد التشابهي لرقمي وهي مرحلة قادح شميت **Schmitt Trigger** والذي يحول الدخل التشابهي لنبضات رقمية ،هذا النوع من الخرج يستخدم في مفاتيح نهاية الشوط أو حساسات الموضع والتي تعرف بمفاتيح أثر هول .



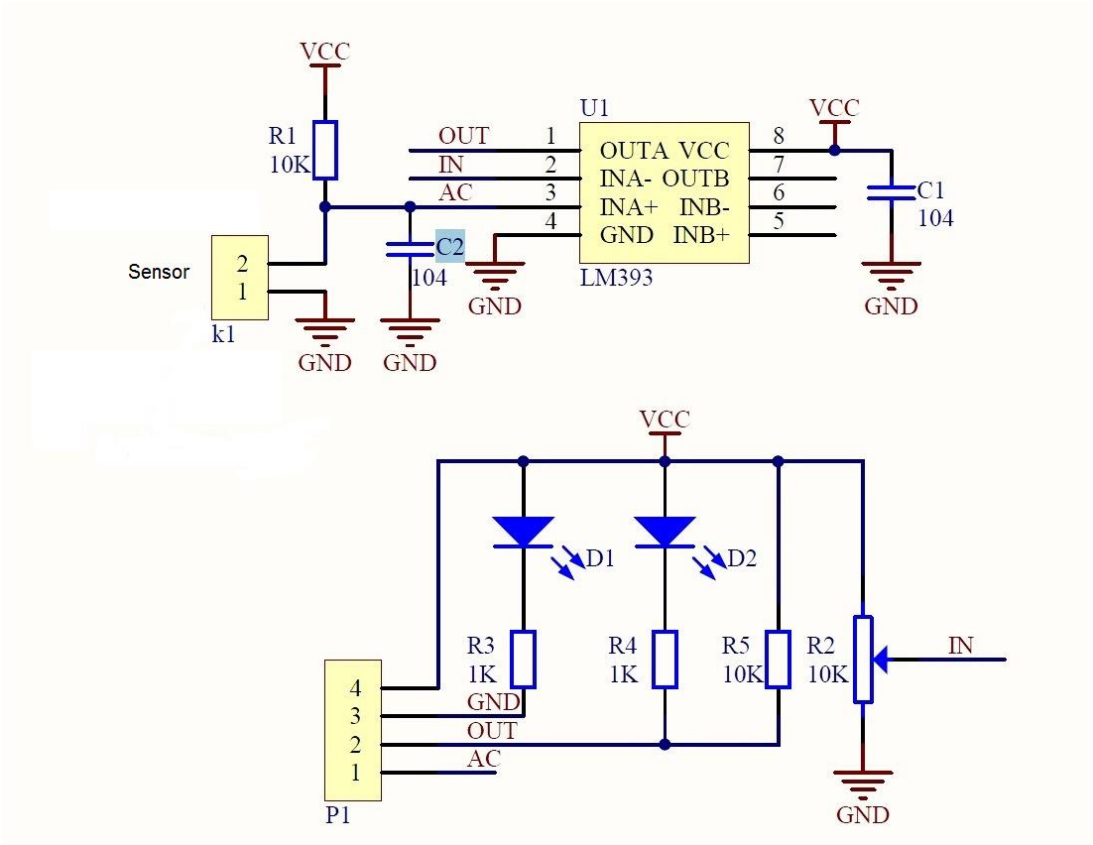
أحد أشهر استخدام الحساس التقاربي هو قياس سرعة المحرك حيث يوضع الحساس على مقربة من محور الحساس ويتم حساب سرعة دوران المحرك **RPM** من خلال حساب عدد نبضات الحساس خلال دقيقة واحدة:



حيث يتم وضع قطعة معدنية على المحور الدوار المراد حساب سرعته ثم وضع الحساس بموقع يتيح له التحسس للجسم المعدني عند الاقتراب منه ويتم عد الدورات مع كل عملية قراءة جديدة للحساس، يتوفر هذا الحساس بنوعين فمنها ما يكون تغيره من الأرضي للموجب عند عمله، بينما يكون تغير النوع الآخر من الموجب للأرضي.

الموديولات التي تحوي في بنيتها IC:LM393

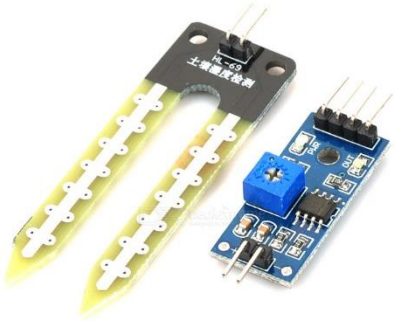
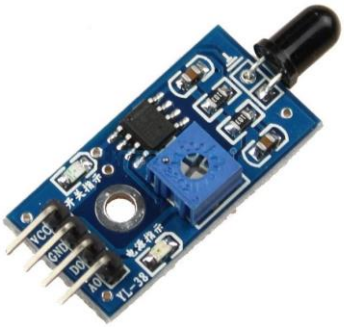
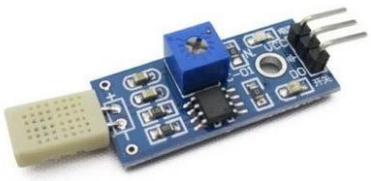
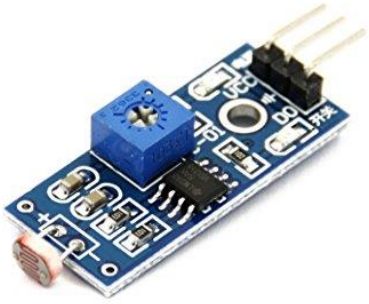
في هذا القسم سنتطرق للعديد من الموديولات والحساسات التي بنية بالاعتماد على دارة **مضخم العملياتي** ذو الرقم **LM393**، حيث تم الاستفادة من هذا المضخم وبناء دارة **مقارن Comparator** بين الجهد الناتج عن الطرفية الأساسية (مقاومة ضوئية ، مقاومة متغيرة ، مستقبل أشعة تحت الحمراء ، حساس صوت ، حساس رطوبة ....) ودرجة الحساسية التي سوف يتم ضبطها عبر المقاومة المتغيرة الموجودة على الموديول والتي يتم من خلالها تحديد الدرجة التي سوف يبدأ الحساس عندها بالعمل.



لو استعرضنا المخطط النظري لبنية هذا الموديول لوجدنا أنها تتألف من الأقسام التالية:

- **القسم الأول:** وتتعلق **بالحساس** الموجود على الموديول **وملحقاته**، وغالبا ما يشكل الحساس مع مقاومة أخرى (**R1 = 10k**) مقسم جهد خرجه متصل مع القطب **INA+** من المقارن.
- **القسم الثاني:** ويضم **المقارن** الذي يحتوي على المدخلين هما المدخل +INA من الحساس والمدخل -IN من المقاومة المتغيرة لضبط نقطة العمل للحساس، أما الخرج **OUT** فهو خرج رقمي يعطي القيمة 0 منطقي عندما يكون القيمة على القطب -INA أكبر من القيمة الناتجة من الحساس، بينما يعطي 1 منطقي عندما يكون القيمة الناتجة عن الحساس أكبر من نقطة الضبط على القطب -INA.
- **القسم الثالث:** ويضم **أقطاب التوصيل** وهي أقطاب التغذية (Vcc & GND) وقطب الخرج الرقمي **OUTA** وقطب الخرج التشابهي **AC** والذي يكون متصل مباشرة مع نقطة الخرج لمقسم الجهد للحساس مع المقاومة 10k، أو قد يكون غير متصل وذلك حسب نوع الحساس الموجود على الموديول.
- **ملحقات إضافية:** كما يوجد على الموديول **ليدات** للدلالة على التغذية والخرج الرقمي وكذلك بعض **المكثفات** للاستقرار والتي تكون بقيمة **100nF**.

لهذا الموديول العديد من الأنواع التي تختلف فيما بينها فقط في الحساس الموجود عليها لكن في النهاية فإن طريقة العمل واحدة وهي خرج رقمي على الرجل OUT من الموديول، الجدول التالي سيبين أشهر الموديولات التي تحتوي في بنيتها على الدارة التكاملية IC:LM393:

حساس رطوبة التربة: 	**حساس الحرائق:** 
حساس صوت: 	**حساس المسافة باستخدام مرسل مستقبل أشعة تحت الحمراء** 
حساس رطوبة الجو 	**حساس ضوء باستخدام LDR** 

تمارين الفصل الرابع

1

اكتب كود برمجي لحساس أمواج فوق صوتية متصل مع لوحة الآردوينو، يقوم الحساس وعند تشغيل الكود البرمجي للمرة الأولى بحساب المسافة البدائية بين الحساس وبين أقرب حاجز يقع ضمن مجال الحساس، ثم يقوم بالتحسس لأي جسم جديد سوف يقع بين الحساس والحاجز البدائي فيشغل زمور باستخدام التعليمة `.tone()`

2

لدينا ثلاثة حساسات PIR متوضعة بحيث يغطي كل منها قطاع زاوي قدره 60 درجة، كما يوجد محرك سيرفو وعليه كاميرا، والمطلوب كتابة كود برمجي يقوم بتحريك الكاميرا بالاتجاه الذي يتحسس عنده أحد الحساسات.

3

لدينا حساس لمعرفة درجة رطوبة التربة متصل مع لوحة الآردوينو، كما يوجد لدينا ريليه للتحكم بتدفق مياه الري للتربة، والمطلوب كتابة كود برمجي يشغل الريليه عند انخفاض درجة رطوبة التربة دون المستوى المطلوب لمدة ثلاثة دقائق أو حتى يعطي الحساس أمر بأن الرطوبة للتربة وصلت للحد المطلوب، مع عرض الوقت المتبقي على خانة واحدة من شاشة 7Segment .

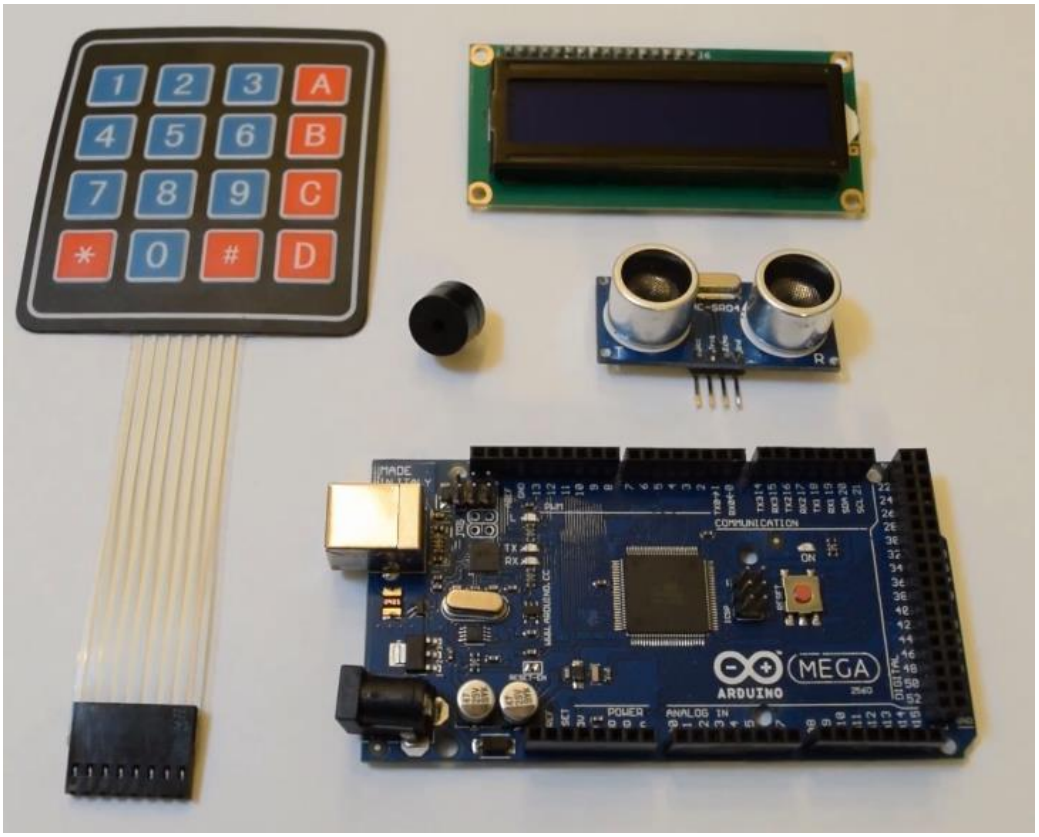
المشروع الأول: جهاز إنذار بكلمة سر وحساس مسافة

تقوم فكرة المشروع على بناء جهاز إنذار يعمل بالتنسيق مع حساس المسافة، فعند بدء التشغيل يقدم الجهاز للمستخدم خيارين:

- **الأول**: تشغيل الجهاز **Actviate** ويكون متعلق بالمفتاح **A** من لوحة المفاتيح، وفي حال اختياره يبدأ عداد ثواني تنازلي بالعمل ليتم تفعيل الحماية بعد 10sec مع إظهار العد على الشاشة.
- **الثاني**: تغيير كلمة السر **Change Pass** ويكون متعلق بالمفتاح **B** من لوحة المفاتيح، وعند اختياره سيطلب منك إدخال كلمة السر القديمة فإن كانت صحيحة سيطلب منك إدخال كلمة السر الجديدة، ثم يعود لينتقل للشاشة الأساسية التي تعرض على المستخدم اختيار أحد النمطين الأساسيين.

في حال الدخول في نمط الحماية ومر جسم ما أمام الحساس فإن جهاز الإنذار سوف يعمل ويشغل الزمور ويطلب إدخال كلمة السر حتى يتوقف، وفي حال إدخال كلمة خاطئة يعيد ويطلبك منك إدخال الكلمة الصحيحة، إن فهم خطوات عمل المشروع تساعد على فهم الكود البرمجي وسبب ترتيبه بشكله الحالي.

المكونات الأساسية للمشروع:

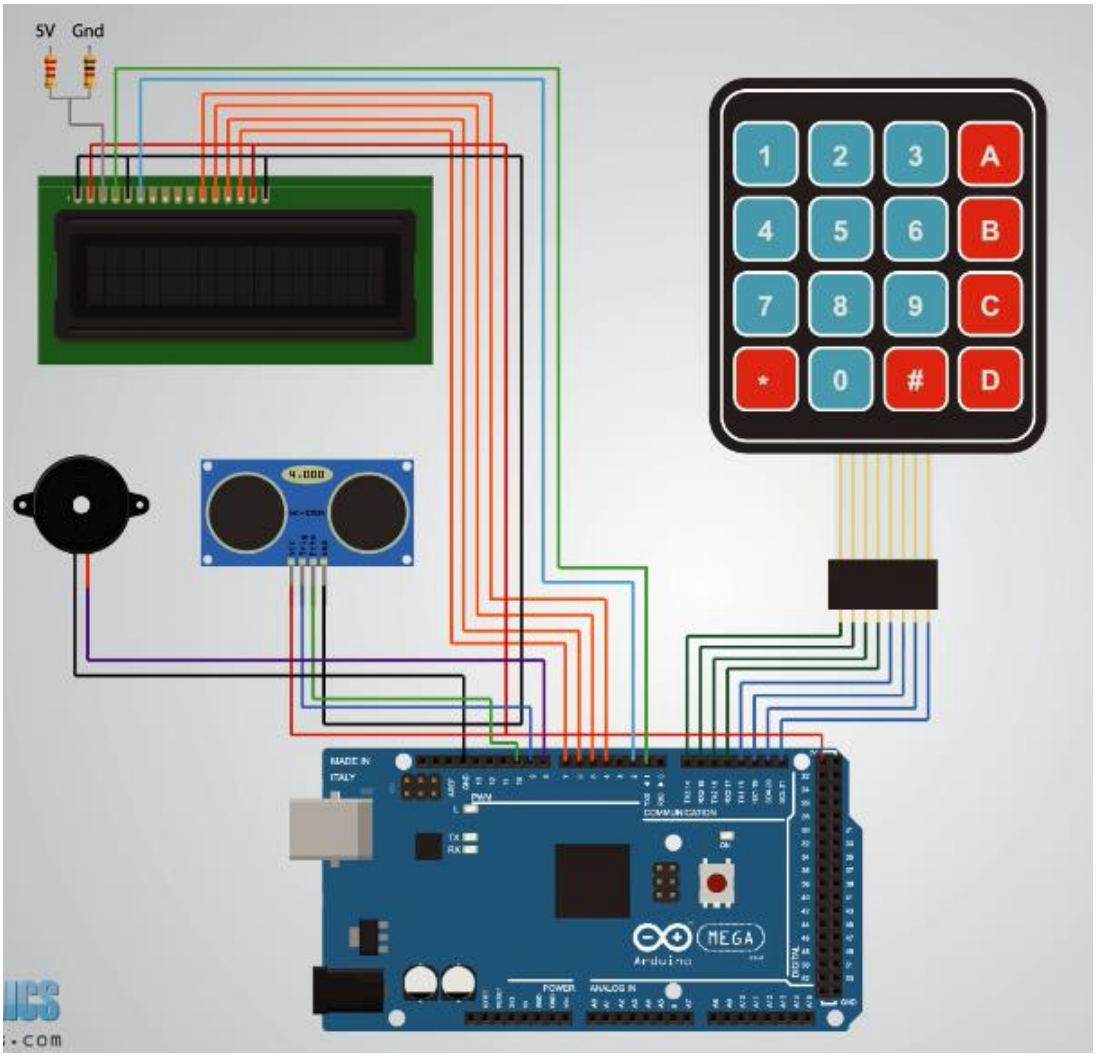


يضم المشروع العناصر الأساسية التالية:

- لوحة آردوينو من نوع **Arduino MEGA** أو **Arduino UNO**.
- لوحة مفاتيح ست عشرية hex keypad.
- حساس مسافة بالأمواج فوق الصوتية Ultra-Sonic.
- شاشة عرض محرفية كرستالية **LCD** أبعادها **16*2**.
- زمور إنذار **buzzer**.

المخطط النظري للمشروع:

بعد التعرف على العناصر المطلوبة بقي لدينا تحديد الأقطاب المخصصة لكل طرفية متصلة مع لوحة الآردوينو وذلك بفرض أن لوحة الآردوينو **Arduino MEGA** هي التي سوف يتم التعامل معها، أما لوحة المفاتيح الست عشرية فسيتم توصيلها مع الأقطاب 21 ~ 14، أما شاشة العرض الكرستالية **LCD** فيتم توصيلها مع الأقطاب (1,2,4,5,6,7)، بقي لدينا حساس الأمواج فوق الصوتية والذي يتم توصيله مع الأقطاب (10 , 9)، وأخيرا الزمور يتم توصيله للقطب (8).



الكود البرمجي للمشروع:

في القسم الأول من الكود يتم تعريف المكتبيات والمتحولات المطلوبة لعمل الكود البرمجي، فتم تعريف مكتبية شاشة العرض الكرستالية وكذلك مكتبية لوحة المفاتيح الست عشرية، وتعريف أقطاب بأسماء ثانية للدلالة عليها وهي أقطاب الزمور وقطب القدح لحساس الأمواج فوق الصوتية وقطب قراءة النبضة الواردة من حساس الأمواج فوق الصوتية.

أما المتحولات التي عرفنها في البداية فمنها قسم خاص بحساس الأمواج فوق الصوتية (لحساب المسافة distance وتحديد المسافة البدائية initialDistance عند أول التشغيل وكذلك تحديد المسافة الحالية عند كل قراءة currentDistance ، بالإضافة لمتحولات أخرى لضبط كلمة السر البدائية (معرف كسلسلة محرفية) ومتحولات منطقية لضبط حالة جهاز الإنذاروغير ذلك من المتحولات التي سنتعرف على وظيفتها لاحقا.

```
#include <LiquidCrystal.h>
#include <Keypad.h>
#define buzzer  8
#define trigPin 9
#define echoPin 10
long duration;
int distance, initialDistance, currentDistance, i;
int screenOffMsg =0;
String password="1234";
String tempPassword;
boolean activated = false; // State of the alarm
boolean isActivated;
boolean activateAlarm = false;
```

```
boolean alarmActivated = false;
boolean enteredPassword;  //State of the entered password to stop the alarm
boolean passChangeMode = false;
boolean passChanged = false;
```

ثم نقوم بضبط لوحة المفاتيح من حيث عدد الأسطر والأعمدة ومكان اتصالها على لوحة الآردوينو والقيم التي سوف تعطيها لوحة المفاتيح والخاصة بكل مفتاح، مع الانتباه أن القيم التي سوف تعيدها لوحة المفاتيح تم ضبطها كقيم محرفية char، وهذا الأمر مهم عند التعامل مع كلمة السر والتي هي بالأساس سلسلة محرفية.

ثم نقوم بتحديد أقطاب التوصيل للشاشة على لوحة الآردوينو.

```
    const byte ROWS = 4; //four rows
    const byte COLS = 4; //four columns
    char keypressed;
    //define the cymbols on the buttons of the keypads
    char keyMap[ROWS][COLS] = {
      {'1','2','3','A'},
      {'4','5','6','B'},
      {'7','8','9','C'},
      {'*','0','#','D'}
    };
//Row pinouts of the keypad
byte rowPins[ROWS]= {14, 15, 16, 17};
//Column pinouts of the keypad
byte colPins[COLS]= {18, 19, 20, 21};
```

```
Keypad   myKeypad=Keypad(makeKeymap(keyMap),   rowPins,
colPins, ROWS, COLS);
//:(rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(1, 2, 4, 5, 6, 7);
```

ضمن حلقة التهيئة `void setup()` يتم تهيئة شاشة العرض الكرستالية وكذلك الأقطاب المطلوبة للمشروع وهي قطب الزمور وقطب القدح لحساس المسافة وقطب قراءة خرج حساس المسافة.

```
void setup()
{
 lcd.begin(16,2);
 pinMode(buzzer, OUTPUT);// Set buzzer as an output
 pinMode(trigPin,OUTPUT);//Sets the trigPin as an Output
 pinMode(echoPin, INPUT); // Sets the echoPin as an Input
 }
```

بداية البرنامج الرئيسي والذي سوف نقم بتقسيمه لعدة أقسام.

```
void loop()
{
```

في بداية الكود يختبر المتحول **activateAlarm** (والذي تصبح قيمته **true** عند إختيار تفعيل الجهاز في بداية الكود عبر ضغط المفتاح A من لوحة المفاتيح والذي سنصل له لاحقا من الكود البرمجي) فإن كان قد تم اختيار الدخول في نمط عمل الجهاز يقوم بما يلي:

- يمسح الشاشة ويطبع عليها العبارة **Alarm will** be في السطر الأول بينما يطبع في السطر الثاني العبارة **activated in**.

- يحدد موقع المؤشر في آخر خانة على الشاشة بعد نهاية العبارة السابقة ثم يدخل في حلقة عداد لمدة 10sec عبر الحلقة الشرطية `while(countdown != 0)`، وفي كل مرة يعد ويشغل الزمورللتنبيه على الدخول في نمط العد وهكذا.
- بعد الانتهاء من العد يمسح الشاشة ويطبع العبارة الجديدة `Alarm Activated`، أي تم تفعيل الإنذار ثم يتم إستدعاء التابع `getDistance()` ومعرفة المسافة البدائية **`initialDistance`** التي تفصل الجهاز عن أقرب جسم أمامه لكي يتم اعتبارها المسافة البدائية.
- وفي النهاية يعيد قيمة المتحول **`activateAlarm`** للقيمة صفر، بينما يغير قيمة المتحول **`alarmActivated`** للقيمة **`true`** لكيلا يدخل الكود البرمجي في نفس الحلقة مرة ثانية ويتم الانتقال لحلقة جديدة متعلقة بالمتحول **`alarmActivated`**.


```
if (activateAlarm)
{
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print( "Alarm will be" );
```

```
        lcd.setCursor(0,1);
        lcd.print( "activated in" );
      //9 seconds count down before activating the alarm
        int countdown = 9;
        while (countdown != 0)
        {
          lcd.setCursor(13,1);
          lcd.print(countdown);
          countdown--;
          tone(buzzer, 700, 100);
          delay(1000);
        }
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print( "Alarm Activated!" );
        initialDistance = getDistance();
        activateAlarm = false;
        alarmActivated = true;
    }
```

بعد اختيار نمط تشغيل جهاز الإنذار (إعطاء المتحول `alarmActivated == true`) يتم الدخول في الجملة الشرطية المتعلقة بهذا النمط والتي يتم فيها إستدعاء التابع الخاص بحساب المسافة للأجسام امام الحساس فإن تغيرت المسافة عن القيمة البدائية **initialDistance** التي تم اعتمادها في الجملة الشرطية السابقة (مع وجود هامش خطأ قيمته 10cm للتخلص من تفاوت القراءات التي يعطيها الحساس)، وفي حال تواجد

جسم جديد سوف يعمل الزمور ويتم مسح الشاشة واستدعاء التابع enterPassword(); والذي سيطلب من المستخدم إدخال كلمة السر.

```
if (alarmActivated == true)
   {
     currentDistance = getDistance() + 10;
     if (currentDistance < initialDistance)
        {
         tone(buzzer,1000); //Send 1KHz sound signal
         lcd.clear();
         enterPassword();
        }
   }
```

هذه الحلقة الشرطية هي أول حلقة شرطية يتم الدخول إليها والتي ستعرض على الشاشة اختيار أي نمط مطلوب، إما نمط الإنذار **A – Activate** أو النمط **B - Change Pass** ومن ثم يتم تغيير قيمة المتحول screenOffMsg والذي يلعب دور العلم أو flage في هذا القسم.


```
  if (!alarmActivated)
      {
        if (screenOffMsg == 0 )
        {
```

```
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("A - Activate");
            lcd.setCursor(0,1);
            lcd.print("B - Change Pass");
            screenOffMsg = 1;
        }
```

يتم الآن قراءة قيمة المفتاح المضغوط من لوحة المفاتيح وبعد ذلك يتم اختبار القيمة المقروءة فإن كانت A فيتم إعطاء القيمة `activateAlarm = true` ليتم الدخول للنمط الأول.

```
keypressed = myKeypad.getKey();

 if (keypressed =='A') //If A is pressed, activate the alarm
 {
  tone(buzzer, 1000, 200);
  activateAlarm = true;
 }
```

الحالة الشرطية الثانية هي اختيار المفتاح B من لوحة المفاتيح فيتم الدخول في النمط الثاني والذي يتضمن تغيير كلمة السر، في البداية يطلب من المستخدم إدخال كلمة السر القديمة ويتأكد من صحتها:

ثم يطلب من المستخدم إدخال كلمة السر الجديدة ليتم تخزينها واعتبارها كلمة السر الجديدة:

أما الخطوات التي يتم فيها هذا التغيير فتتم كما يلي:

- يتم مسح الشاشة وتعريف متحول i (والذي سوف يكون كمؤشر لكي يتحرك بين خانات الشاشة مع كتابة الأرقام) ثم يتم طباعة العبارة Current Password في السطر الأول ثم يتم طباعة " > " في السطر الثاني من الشاشة مع نقل حالة المتحولين الخاصين بكلمة السر (`passChangeMode` ، `passChanged`) للحالة الفعالة **`true`** .
- بما أن المتحول `passChanged = true` يتم الآن الدخول في حلقة شرطية لا نهائية while لا يتم الخروج منها حتى يتم إدخال كلمة السر، يتم فيها قراءة حالة لوحة المفاتيح فإن كان هناك مفتاح مضغوط وهذا المفتاح ينتمي لأحد القيم (0 ~ 9) يتم تسجيل

القيمة المدخلة على المتحول (tempPassword += keypressed) وفي نفس الوقت يتم طباعة الإشارة " * " على الشاشة وتشغيل الزمور لفترة قصيرة للدلالة على إدخال القيمة.

- إن تم إدخال أكثر من خمس أرقام أو تم ضغط المفتاح " # " يمسح القيم المدخلة ويعيد عملية طلب إدخال كلمة المرور.
- أما إن تم ضغط المفتاح " * " فيتم مقارنة القيمة المدخلة مع القيمة السابقة فإن كانت متساوية فيتم مسح القيمة القديمة والطلب من المستخدم إدخال كلمة المرور الجديدة وهنا ندخل في حلقة while جديدة لكن هذه المرة من أجل إدخال كلمة المرور الجديدة.
- كما في حلقة while السابقة ،يتم اختبار المفاتيح المضغوطة فإ تم ضغط المفتاح " * " فيتم تثبيت الكلمة الجديدة ، وإعطاء المتحولات (passChangeMode ، passChanged) القيمة false وبهذا يتم الخروج من نمط تعديل كلمة السر والعودة للواجهة الرئيسية.

```
else if (keypressed =='B')
        {
          lcd.clear();
          int i=1;
          tone(buzzer, 2000, 100);
          tempPassword = "";
          lcd.setCursor(0,0);
          lcd.print("Current Password");
          lcd.setCursor(0,1);
```

```
            lcd.print(">");
            passChangeMode = true;
            passChanged = true;
            while(passChanged) {
            keypressed = myKeypad.getKey();
            if (keypressed != NO_KEY)
            {
if (keypressed == '0' || keypressed == '1'
 || keypressed == '2' || keypressed == '3'
 || keypressed == '4' || keypressed == '5'
 || keypressed == '6' || keypressed == '7'
 || keypressed == '8' || keypressed == '9')
 {
   tempPassword += keypressed;
   lcd.setCursor(i,1);
   lcd.print("*");
   i++;
   tone(buzzer, 2000, 100);
  }
  }
  if (i > 5 || keypressed == '#')
     {
       tempPassword = "";
         i=1;
         lcd.clear();
         lcd.setCursor(0,0);
         lcd.print("Current Password");
         lcd.setCursor(0,1);
```

```
          lcd.print(">");
        }
        if (keypressed == '*')
         {
          i=1;
          tone(buzzer, 2000, 100);
            if (password == tempPassword)
              {
                tempPassword="";
                lcd.clear();
                lcd.setCursor(0,0);
                lcd.print("Set New Password");
                lcd.setCursor(0,1);
                lcd.print(">");
                 while(passChangeMode)
                  {
                    keypressed = myKeypad.getKey();
                    if (keypressed != NO_KEY)
                 {
if (keypressed == '0' || keypressed == '1'
 || keypressed == '2' || keypressed == '3'
 || keypressed == '4' || keypressed == '5'
 || keypressed == '6' || keypressed == '7'
 || keypressed == '8' || keypressed == '9')
 {
                          tempPassword += keypressed;
                          lcd.setCursor(i,1);
                          lcd.print("*");
```

```
                    i++;
                    tone(buzzer, 2000, 100);
                  }
              }

              if (i > 5 || keypressed == '#')
              {
                tempPassword = "";
                i=1;
                tone(buzzer, 2000, 100);
                lcd.clear();
                lcd.setCursor(0,0);
                lcd.print("Set New Password");
                lcd.setCursor(0,1);
                lcd.print(">");
              }
              if ( keypressed == '*') {
                i=1;
                tone(buzzer, 2000, 100);
                password = tempPassword;
                passChangeMode = false;
                passChanged = false;
                screenOffMsg = 0;
              }
            }
          }
        }
      }
```

```
        }
    }
```

تابع إدخال كلمة السر enterPassword() ، كما رأينا أن هذا التابع يستدعى في حال تواجد جسم جديد أمام الجهاز حيث يقوم بتشغيل الزمور والطلب من المستخدم إدخال كلمة السر، ماهي وظيفة هذا التابع.... يقوم هذا التابع بما يلي:

- تشغيل المتحول الخاص بالتفعيل (activated = true) يطبع على الشاشة العبارة " ***Alarm*** " في السطر الأول، أما السطر الثاني فيطبع العبارة " Pass>".
- بعد ذلك يدخل في حلقة while متعلقة بالمتحول activated ويتم فيها كما في الحلقات المشابهة السابقة إذ يتم قراءة القيم المدخلة من لوحة المفاتيح والتأكد من عدم تجاوزه 5 خانات أو عدم ضغط المفتاح " # "، أما إذا ضغط المفتاح " * " يتم إختبار القيمة المدخلة مع القيمة الأساسية ففي تحقق المساواة يتم إيقاف الزمور وإلغاء متحولات الخاصة بالتفعيل، أما في حال عدم تحقق المساواة فيتم طلب المحاولة مرة ثانية.

ننتبه إلى أن القيمة البدائية للمتحول k = 5 والسبب متعلق في توافق موقع المؤشر على شاشة العرض الكرستالية.

```
void enterPassword()
{
int k=5;
tempPassword = "";
activated = true;
lcd.clear();
```

```
lcd.setCursor(0,0);
lcd.print(" *** ALARM *** ");
lcd.setCursor(0,1);
lcd.print("Pass>");
    while(activated)
    {
    keypressed = myKeypad.getKey();
    if (keypressed != NO_KEY)
    {
if (keypressed == '0' || keypressed == '1'
 || keypressed == '2' || keypressed == '3'
 || keypressed == '4' || keypressed == '5'
 || keypressed == '6' || keypressed == '7'
 || keypressed == '8' || keypressed == '9')
          {
            tempPassword += keypressed;
            lcd.setCursor(k,1);
            lcd.print("*");
            k++;
          }
    }

    if (k > 9 || keypressed == '#')
    {
      tempPassword = "";
      k=5;
      lcd.clear();
      lcd.setCursor(0,0);
```

```
      lcd.print(" *** ALARM *** ");
      lcd.setCursor(0,1);
      lcd.print("Pass>");
    }

    if ( keypressed == '*')
    {
      if (tempPassword == password )
          {
            activated = false;
            alarmActivated = false;
            noTone(buzzer);
            screenOffMsg = 0;
          }
      else if (tempPassword != password)
          {
            lcd.setCursor(0,1);
            lcd.print("Wrong! Try Again");
            delay(2000);
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print(" *** ALARM *** ");
            lcd.setCursor(0,1);
            lcd.print("Pass>");
          }
    }
  }
}
```

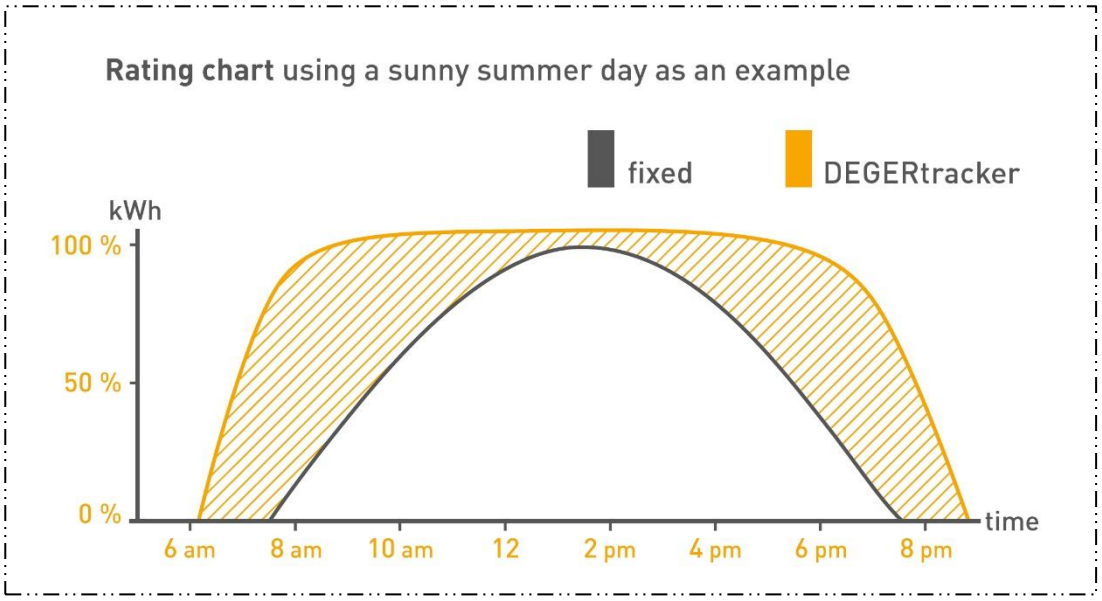
أما تابع حساب المسافة بين الحساس والأجسام التي أمامه فيتم عبر التابع getDistance() ، أما آلية حساب المسافة فهي نفس المثال الذي تم ذكره في الأمثلة السابقة (راجع حساس الأمواج فوق الصوتية)، وفي كل مرة يتم فيها استدعاء التابع يعطينا المسافة التي تفصل أقرب جسم عن الحساس وتكون القيمة بواحدة cm.

```
void  long getDistance()
    {
      digitalWrite(trigPin, LOW);
      delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
      duration = pulseIn(echoPin, HIGH);
// Calculating the distance
      distance = duration*0.034/2;
      return distance;
    }
```

المشروع السابق يعتبر تطبيق عملي جيد جدا للأفكار البرمجية التي تطرقنا لها خلال الجزء الأول وهو مشروع مأخوذ من موقع How to Mechatronics والذي يحتوي على العديد من الأفكار البرمجية الجيدة.

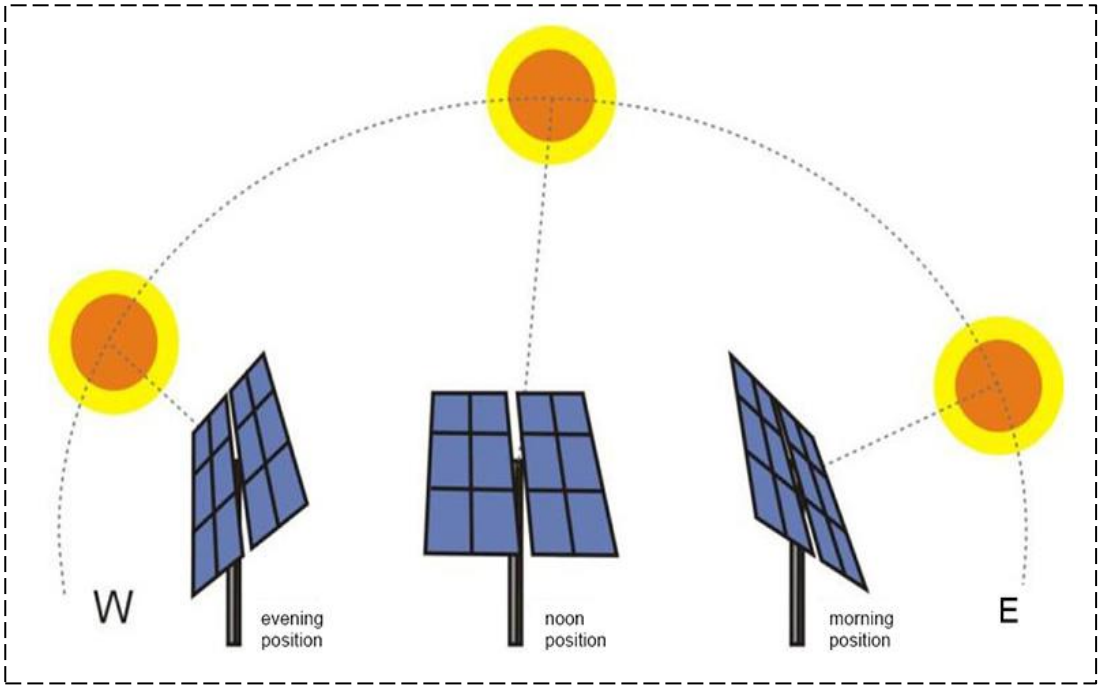
المشروع الثاني: جهاز ملاحقة شمسية

تقوم فكرة المشروع على التحكم بتوجيه ألواح الطاقة الشمسية باتجاه الشمس طوال فترة النهار مستخدمين بذلك الآردوينو مع عدة ملحقات من محركات ومقاومات ضوئية وغير ذلك من القطع المطلوبة للمشروع، والهدف من تحريك الألواح الشمسية هو تحصيل أكبر قدر ممكن من الطاقة كون أن الألواح تقدم أعظم استطاعة لها عندما تكون الأشعة الشمسية متعامدة مع الألواح وهذا ما يبينه المخطط البياني التالي:



فلو تتبعنا المخطط السابق نجد أن ألواح الطاقة الشمسية تقد أعظم استطاعة في ساعات محددة لا تتجاوز الأربع ساعات بينما نجد الألواح المتحركة تعطي طاقة عظمى في أغلب ساعات اليوم وبالتالي تحسين المردود بما يزيد عن 70% من المردود في الحالة الثابتة.

بالعودة لمشروعنا فنحن أمام خيارين إما أن نجعل حركة الألواح وفق محور واحد فقط من شروق الشمس وحتى غروبها على نفس المستوي وبالتالي يجب علينا تعديل الميول بالنسبة للشمس أربع مرات في السنة كون الشمس لها أربع زوايا ميول على مدار العام والتي تحدد بداية كل فصل، أو وفق محورين وهو الحالة الأفضل فيصبح الجهاز قادر على ملاحقة الشمس في كافة الاتجاهات على مدار العام ودون الحاجة لأي تعديل فقط عندنا الضبط لأول مرة عند التركيب فقط.



بفرض أن الجسم الميكانيكي قابل للحركة في كافة الاتجاهات وفق المحورين **X & Y** يبقى لدينا تحديد **أنواع المحركات** التي سوف تحرك الألواح وفي حالة الألواح الكبيرة يفضل استخدام محركات تيار مستمر **خطية** الحركة وهي الأفضل للمشروع، وفي حال كانت الألواح صغيرة يمكن استخدام محركات خطوية أو محركات سيرفو.

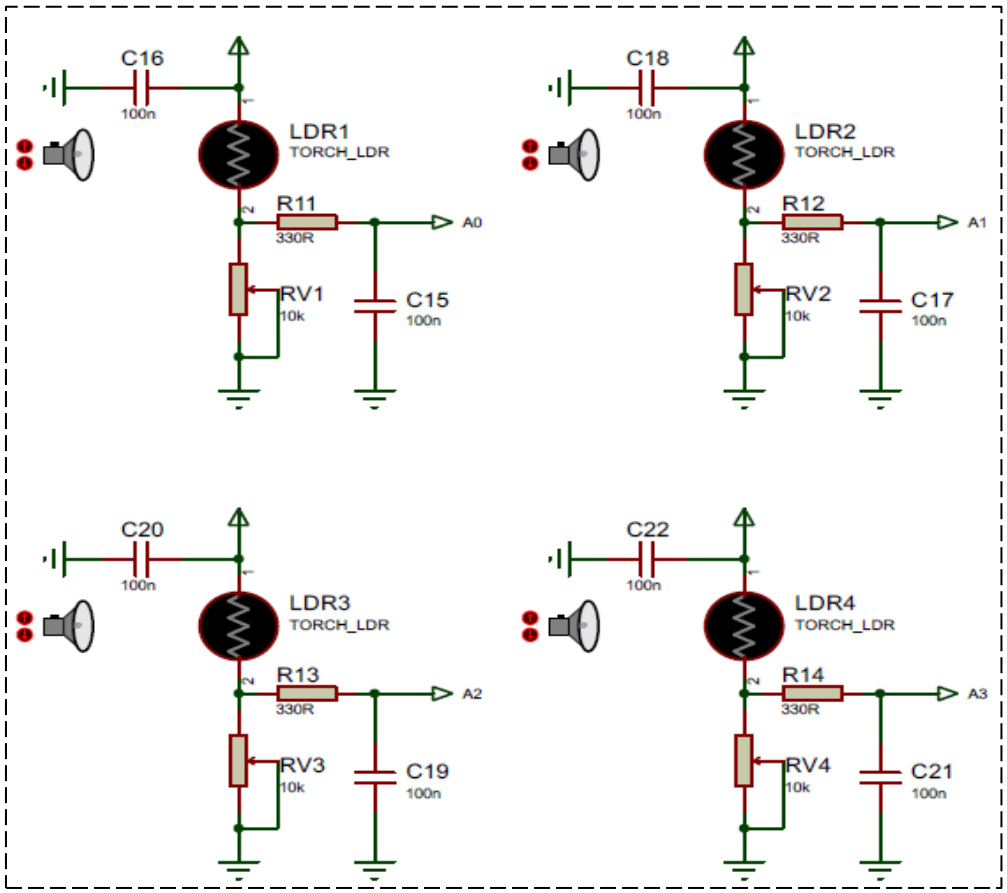
المكونات الأساسية للمشروع:

- محرك تيار مستمر(عدد 2) خطي **Linear DC Motor** للـحركة وفق المحـورين **X & Y**.
- لوحة آردوينو **Arduino NANO**.
- دارة قيادة للمحركين والتي سوف تكون عبارة عن جسر **H-Bridge** من تماسات الريليهات.
- مقاومات ضوئية **LDR** والتي سوف تلعب دور المحدد للموقع الأفضل لاستقبال أكثر الأشعة الشمسية.

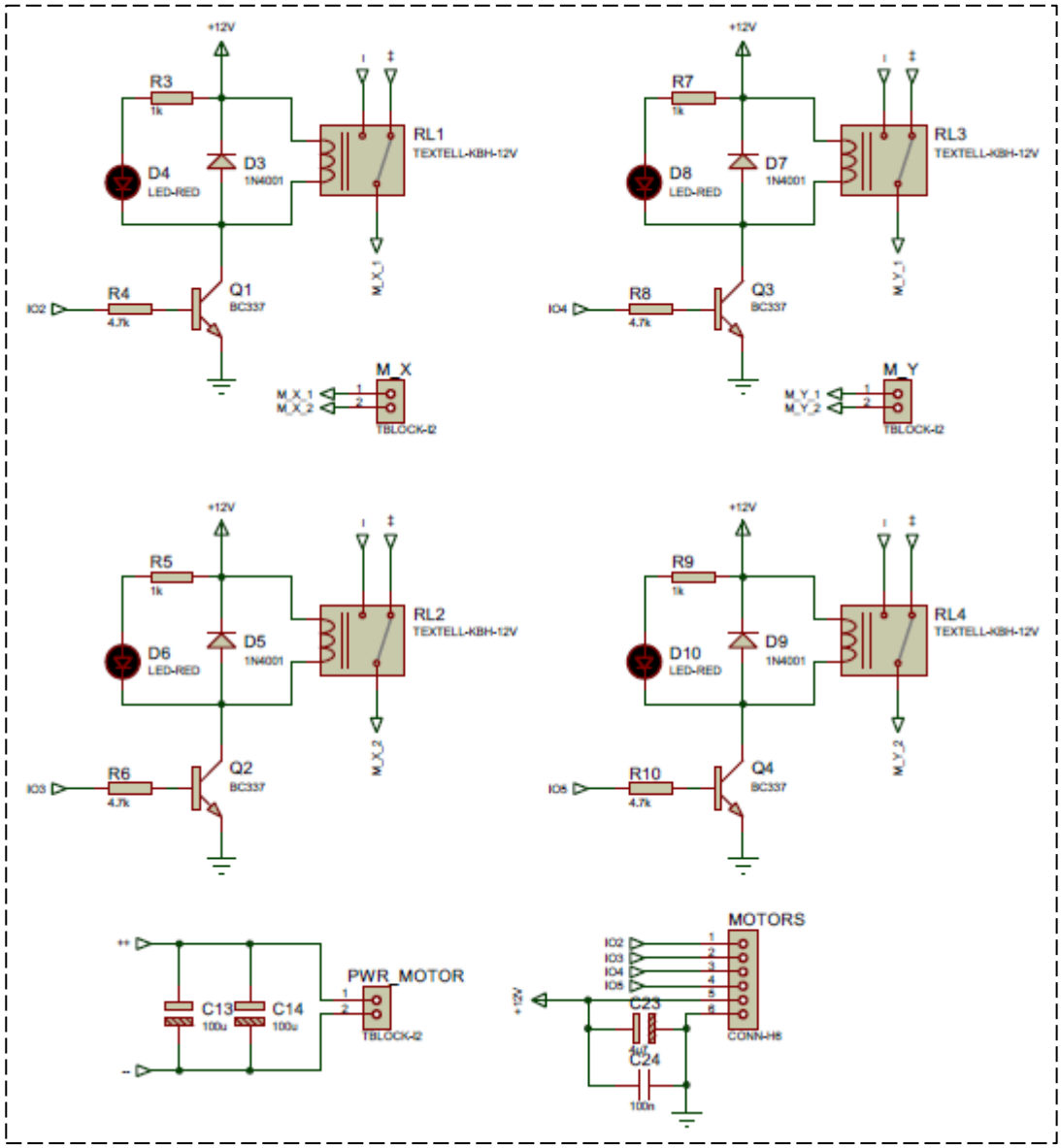
المخطط النظري للمشروع:

فيما يلي سنبين آلية توصيل الحساسات مع لوحة الآردوينو وكذلك الريليهات التي سوف تتحكم بحركة المحرك وسيكون كل قسم منها في لوح منفصل.

الشكل التالي يبين آلية توصيل المقاومات الضوئية وكيفية ربطها مع لوحة الآردوينو، ففي طريقة التوصيل هذه سيزداد جهد الخرج كلما زادت شدة الإضاءة المطبقة على المقاومة الضوئية، أما المقاومة المتغيرة المتصلة على التسلسل مع المقاومة الضوئية فهي لتأمين عملية الضبط الدقيقة كون المقاومات تعطي قيم خرج مختلفة عند نفس الإضاءة وهذا بسبب التصنيع، أما المكثفات فهي لزيادة الاستقرار والتخلص من الإشارات العشوائية والحصول على قراءة مستقرة.



وبالنسبة للمحركات فنحتج كما ذكرنا لـ **H-Bridge** سنصممه من **تماسات الريليهات** مع الأخذ بعين الاعتبار أن جهد العمل للمحرك يتراوح بين **12 ~ 36 volt** بما لا يتجاوز **5A** فيكون التصميم الخاص بإدارة المحركات كما في الشكل التالي:

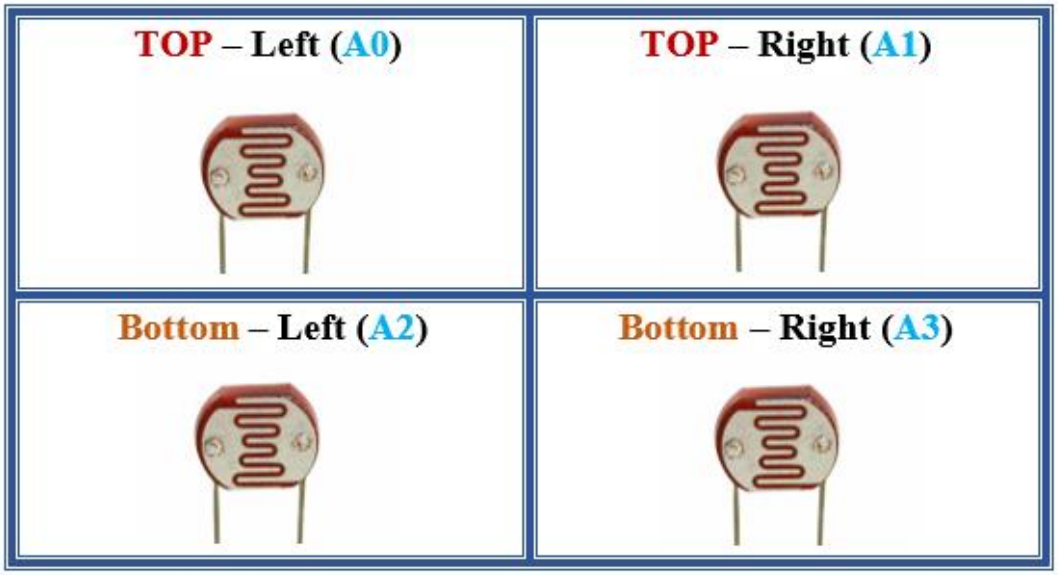


في حالتنا هذه افترضنا أن جهد التشغيل للملفات هو **12v** وهذا الأكثر انتشارا، من الضروري وضع **مكثفات** على منبع **التغذية** الخاص **بالمحرك** وذلك للاستقرار والتخلص من الضجيج، بعض المحركات الخطية تحتوي في بنيتها على مفاتيح نهاية شوط والتي يمكن ضبطها يدويا لتحديد المواضع الحدية المسموح للمحرك الوصول إليها ولا نحتاج هنا لتدخل الآردوينو برمجيا، وإذا لم تتوفر هذه المحركات عندئذ نضيف هذه الحساسات للمشروع وعددها أربعة مفاتيح نضيف لها مكثفات استقرار للتخلص من حالة الاهتزاز الميكانيكي مع تفعيل مقاومة الرفع الداخلية على أقطاب الآردوينو لتعمل عند وصول الصفر منطقي على القطب، سنوزع المفاتيح بحيث يكون نهاية المحور X من اليمين متصل مع 6 بينما الطرف اليساري للمحور X متصل مع 7، أما المحور Y فمن الأعلى مع القطب 8 ومن الأسفل مع القطب 9.

وهكذا تكون الدارة الكهربائية جاهزة للعمل ويبقى لدينا كتابة الكود البرمجي للمشروع، مع ترك حرية التعديل للمبرمج وإضافة ميزات للمشروع وتعديل الكود البرمجي.

الكود البرمجي للمشروع:

تقوم الفكرة الأساسية للكود البرمجي على قراءة حالة المقاومات الضوئية المتوزعة بشكل متصالب وتحديد حركة المحاور وفق القيم المقروءة وذلك لجعل **ألواح الطاقة** الشمسية **متعامدة** طوال اليوم مع **أشعة الشمس**، أما باقي الملحقات فعندنا قسم قراءة قيمة الجهد الناتج عن الطاقة الشمسية عبر القطب A4، وقسم عرض البيانات على شاشة LCD يتضمن حركة الألواح وقيمة الجهد الآتي من ألواح الطاقة الشمسية وكذلك قيمة التيار الكهربائي الذي تقدمه الألواح، علينا عند كتابة الكود مراعاة التوزيع التالي للمقاومات الضوئية LDR.



مع مراعات توضع المقاومات بشكل **مستوي موازي للوح الطاقة الشمسية** ومعايرة المقاومات للحصـــول على نفس الخرج عند نفس الشـــدة الضـــوئية ويتم الضـــبط عبر المقاومة المتغيرة الموصولة مع التسلسل مع كل مقاومة ضوئية.

في القسم الأول من الكود نكتب التعريفات الخاصة بتوزيع أقطاب التحكم بجهة دوران المحركات وكذلك الأقطاب الخاصة بمفاتيح نهاية الشوط

```
#define   LDR_TopRight_pin        A1
#define   LDR_TopLeft_pin         A0
#define   LDR_BottomRight_pin     A3
#define   LDR_BottomLeft_pin      A2
#define   Motor_X_left_pin        3
#define   Motor_X_right_pin       2
#define   Motor_Y_down_pin        5
#define   Motor_Y_up_pin          4
#define   limit_X_R_pin           6
#define   limit_X_L_pin           7
#define   limit_Y_U_pin           8
#define   limit_Y_D_pin           9
```

بعد ذلك نقوم بتعريف المتحولات المطلوبة للكود البرمجي، فنعرف المتحول `ErrorAmount` لتحديد هامش الخطأ عند مقارنة قراءات المقاومات المتغيرة، أما المتحول `night_value` فسوف نستخدمه لتحديد القيمة التي تنبؤنا بحلول الليل لكي نقوم بإعادة تحريك الألواح نحو المشرق (لكي تكون الألواح جاهزة للعمل مباشرة في اليوم التالي).

أما باقي المتحولات فهي لتخزين قيم القراءات التشابهية المأخوذة من المقاومات الضوئية ولتخزين حالة المفاتيح الحدية للمحورين X & Y .

```
int ErrorAmount = 25;
int night_value = 150;
```

```
int val_TR ,val_TL , val_BR ,val_BL ;
boolean state_X_R ,state_X_L , state_Y_U, state_Y_D;
```

في حلقة التهيئة يتم تفعيل الأقطاب المطلوبة بحسب طبيعة عملها مع تفعيل **مقاومة الرفع** لأقطاب الدخل الخاصة بمفاتيح نهاية الشوط، كما نفعل واجهة الاتصال التسلسلي لاستخدامها إن لزم الأمر.

```
void setup()
{
  Serial.begin(9600);
  pinMode(Motor_X_left_pin  , OUTPUT);
  pinMode(Motor_X_right_pin , OUTPUT);
  pinMode(Motor_Y_down_pin  , OUTPUT);
  pinMode(Motor_Y_up_pin    , OUTPUT);
  pinMode(limit_X_R_pin ,INPUT_PULLUP);
  pinMode(limit_X_L_pin ,INPUT_PULLUP);
  pinMode(limit_Y_U_pin ,INPUT_PULLUP);
  pinMode(limit_Y_D_pin ,INPUT_PULLUP);
}
```

حلقة البرنامج الرئيسية يتم فيها أولا قراءة قيم المقاومات الضوئية وقراءة حالة مفاتيح نهاية الشوط ثم يتم الدخول في حلقات شرطية متعلقة بالقيم المقروءة (سنعتبر أن الحركة لليمين للمحور X باتجاه الغروب بينما الحركة لليسار ستكون باتجاه الشروق) فيتم تحريك المحركات تبعا للقراءات السابقة وتبعا لحالة مفاتيح نهاية الشوط لكيلا يدخل المحرك في حالة كبح كهربائي ويسبب ذلك تلفه.

كما يوجد شرط يختبر قراءة جميع المقاومات الضوئية فإن كانت قيمة القراءة أقل من القيمة الحدية (المتحول night_value الذي يحدد القيمة التي يكون عندها الليل ويتم اختيار قيمته بعد التجريب) يتم تحريك الألواح نحو اليسارأي نحو المشرق، اما الحالة العامة وفي حال تساوي القراءات (يتم ضبط التساوي من خلال هامش الخطأ ErrorAmount) يتم إيقاف جميع الأوامر الخاصة بتحريك المحركات.

```
void loop()
{
  val_TR   = analogRead(LDR_TopRight_pin);
  val_TL   = analogRead(LDR_TopLeft_pin);
  val_BR   = analogRead(LDR_BottomRight_pin);
  val_BL    = analogRead(LDR_BottomLeft_pin);
  state_X_R = digitalRead(limit_X_R_pin);
  state_X_L = digitalRead(limit_X_L_pin);
  state_Y_U = digitalRead(limit_Y_U_pin);
  state_Y_D = digitalRead(limit_Y_D_pin);

  if((val_TR   >   val_TL+ErrorAmount   ||   val_BR   >
val_BL+ErrorAmount)&& state_X_R==1)
  {
    digitalWrite(Motor_X_right_pin , HIGH);
    digitalWrite(Motor_X_left_pin  , LOW );
  }

   else  if((val_TL  >  val_TR+ErrorAmount||  val_BL  >
val_BR+ErrorAmount)&& state_X_L==1)
```

```
  {
    digitalWrite(Motor_X_right_pin , LOW );
    digitalWrite(Motor_X_left_pin  , HIGH);
  }

  else  if((val_BR  >  val_TR+ErrorAmount||  val_BL  >
val_TL+ErrorAmount)&& state_Y_D==1)
  {
    digitalWrite(Motor_Y_up_pin    , LOW );
    digitalWrite(Motor_Y_down_pin  , HIGH);
  }

  else  if((val_TR  >  val_BR+ErrorAmount||  val_TL  >
val_BL+ErrorAmount)&& state_Y_U==1)
  {
    digitalWrite(Motor_Y_up_pin    , HIGH);
    digitalWrite(Motor_Y_down_pin  , LOW );
  }

  else if(val_TR < night_value && val_TL < night_value
&&  val_BR  <  night_value  &&  val_BL  <  night_value  &&
state_X_L == 1)
  {
    digitalWrite(Motor_X_right_pin , LOW );
    digitalWrite(Motor_X_left_pin  , HIGH);
  }
  else
  {
```

```
    digitalWrite(Motor_X_right_pin , LOW);
    digitalWrite(Motor_X_left_pin  , LOW);
    digitalWrite(Motor_Y_up_pin    , LOW);
    digitalWrite(Motor_Y_down_pin  , LOW);
  }
  delay(100);
}
```

يمكن للمستخدم أن يطور هذا المشروع بإضافة شاشة عرض كرستالية LCD ووضع بعض الحساسات الإضافية كحساس قياس التيار الكهربائي لقراءة التيار المقدم من الألواح أو إضافة مقسم جهد وقراءة قيم الجهد على البطاريات وغير ذلك من الأفكار التي يمكن من خلالها تقديم منتج متكامل للمستخدم.

حل تمارين الكتاب

حل تمارين الفصل الأول

1

ليكن لدينا المصفوفة التالية: myArray [10] = {9 , -8, 52 , 87 , 12, 7 , -6}

المطلوب كتابة كود برمجي يقوم بإضافة القيمة 5 لكل عنصر من عناصر المجموعة لثلاث مرات وفي كل مرة نقوم بحساب متوسط عناصر المصفوفة وطباعة الناتج على النافذة التسلسلية وتشغيل زمور متصل مع أحد أقطاب الآردوينو لمدة نصف ثانية.

```
int myArray[10]={9 , -8, 52 , 87 , 12, 7 , -6};
int x, Average ,sum ;

void setup()
{
pinMode(2 ,OUTPUT);
Serial.begin(9600);
}

void loop()
{
if( x < 3)
  {

    for(x ; x<3 ;x++)
    {
      Average = 0;
      sum =0 ;
      for(int i = 0 ; i<10 ;i++)
```

```
      {
        myArray[i] = myArray[i]+5;
        Serial.println(myArray[i]);
        delay(100);
      }
      for(int k=0 ;k<10 ;k++)
      {
        sum = sum + myArray[k] ;
      }
      Average = sum / 10;
      Serial.print("Average =");
      Serial.println(Average);

    }
  }
else if(x == 3)
  {
    tone(2 , 315 , 500);
  }
}
```

2

لنكتب كود برمجي لتشغيل ليد LED متصل مع القطب 13 من لوحة الآردوينو بحيث يتم تشغيل وإطفاء الليد من نفس الكباس اللحظي المتصل مع القطب 3 على ألا تزيد عدد مرات تشغيل الليد العشر مرات فقط، مع وجود كباس لحظي آخر متصل مع القطب 2 لتصفير عدد المرات التي تم ضغطها.

```
#define LED  13
#define B1   3
#define RST  2
int x ;

void setup()
{
  pinMode(LED ,OUTPUT);
  pinMode(B1  ,INPUT_PULLUP);
  pinMode(RST ,INPUT_PULLUP);
  Serial.begin(9600);
}

void loop()
{
  if(digitalRead(B1) == 0 &&  x < 20)
  {
    delay(10);
    while(digitalRead(B1) == 0);
    x+=1;
    digitalWrite(LED ,digitalRead(LED)^1);
    Serial.println(x);
```

```
  }

  else if(digitalRead(RST)==0)
  {
    delay(10);
    x = 0 ;
    Serial.println(x);
  }
}
```

3

لدينا كباسين لحظيين متصلين مع القطبين A0 , A1 الأول لزيادة قيمة المتحول val بمقدار 2 والثاني لإنقاص قيمة المتحول val بمقدار 1، مع المحافظة على قيمة المتحول val ضمن المجال (8 , 3-).

```
#define INCR  A0
#define DECR  A1

int val ;
boolean ON = 0   , OFF = 1 ;

void setup()
{
  pinMode(INCR  ,INPUT_PULLUP);
  pinMode(DECR ,INPUT_PULLUP);
  Serial.begin(9600);
}

void loop()
{
  if(digitalRead(INCR) == ON)
  {
    delay(10);
    while(digitalRead(INCR) == ON);
    val = val + 2;
    val = constrain(val, -3 ,8);
    Serial.println(val);
  }
```

```
 else if(digitalRead(DECR) == ON)
  {
    delay(10);
    while(digitalRead(DECR) == ON);
    val = val - 1;
    val = constrain(val, -3 ,8);
    Serial.println(val);
  }
}
```

4

اكتب كود برمجي يتم من خلاله بناء مؤقت يعد مع بدء تشغيل اللوحة وعندما يصبح الوقت ساعة كاملة يشغل زمور متصل مع أحد أقطاب لوحة الآردوينو لمدة ثانية ويطبع العبارة Done على النافذة التسلسلية ثم يعود ويبدء العد من جديد.

```
byte s , m ,h ;
void setup()
{
Serial.begin(9600);
}

void loop()
{
s++;
if(s > 59)
  {
    s = 0;
    m++;
      if(m > 59)
      {
        m = 0;
        h++ ;
          if( h == 1)
          {
           h = 0 ;
           tone(2 ,500 ,1000);
           Serial.print(" Done ");
           delay(2000);
```

```
            }
        }
    }
Serial.print(h);
Serial.print(":");
Serial.print(m);
Serial.print(":");
Serial.println(s);
delay(1000);
}
```

5

باستخدام التعليمة millis() اكتب كود برمجي لتشغيل LED متصل مع أحد أقطاب الآردوينو بحيث يعمل فقط بعد استمرار ضغط كباس لحظي Push Button متصل مع القطب 2 لمدة 3sec وعند ترك الكباس يتم إطفاء LED.

```
long timet,duration;
void setup()
{
  pinMode(3,INPUT_PULLUP);
  pinMode(13,OUTPUT);
}
void loop()
{
  timet=millis();
  while(digitalRead(3)==LOW)
  {
    duration=millis()-timet;
    if (duration>=3000)
    {
      digitalWrite(13,HIGH);
    }
    else digitalWrite(13,LOW);
  }
  digitalWrite(13,LOW);
}
```

6

لدينا مقاومة متغيرة موصولة مع القطب التشابهي A0 من لوحة Arduino UNO كما يوجد لدينا 10 ليدات متصلة مع الأقطاب 2 ~ 11 والمطلوب: كتابة كود برمجي يقوم بتشغيل الليدات بالاعتماد على النسبة المئوية لقيمة المقاومة (أي كل 10 % من قيمة المقاومة يقابلها تشغيل ليد من الليدات بحيث يكون مجموع الليدات التي تعمل متناسب مع النسبة المئوية للمقاومة).

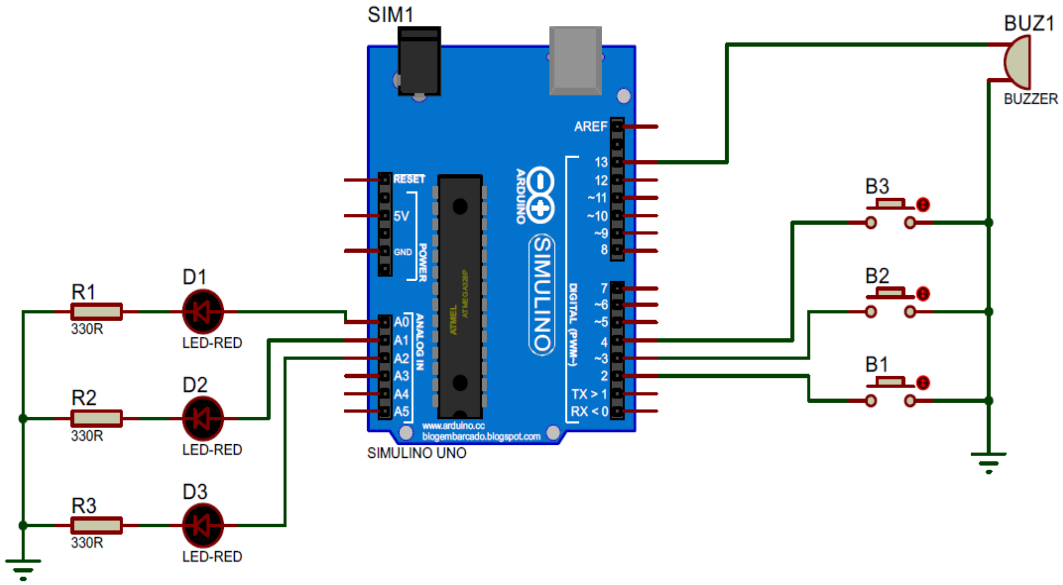
```
void setup()
{
  forint i=2 ; i<12 ;i++)
  {
    pinModei , OUTPUT);
  }
Serial.begin(9600);

}

void loop()
{
int x ;
 x = sensoer_read();
 led_off();
 led_on(x);
}

int sensoer_read()
{
  int val ;
  val = analogRead(A0);
```

```
  val = map(val,0 , 1023 ,2 , 11);
  return val;
}

void led_off()
{
  for(int j=2 ;j<12 ;j++)
  {
    digitalWrite(j , 0);
  }
}

void led_on(int v)
{
  for(int i =2 ; i<12 ;i++)
  {
    digitalWrite( i ,1);
    if(i == v)
    {
      break;
    }
  }
}
```

8

ليكن لدينا ثلاثة كباسات لحظية متصلة مع الأقطاب 2 , 3 , 4 كما يوجد لدينا ثلاثة LEDs متصلة مع الأقطاب A2 , A1 , A0 ، أيضا يوجد زمور Buzzer متصل مع القطب 13 ، والمطلوب وباستخدام التعليمة millis() كتابة كود برمجي لتشغيل الزمور لمدة 1sec وإطفائه بنفس المدة ، وكذلك وعند الضغط على أي كباس لحظي يتم تشغيل الليد المقابل له طالما الكباس مضغوط .

المخطط التالي يبين التوصيل الكهربائي للمشروع:




```
#define   B1     2
#define   B2     3
#define   B3     4
```

```
#define  led1  A0
#define  led2  A1
#define  led3  A2
#define  buz   13

unsigned long time1;

void setup()
{
 pinMode(led1 , OUTPUT);
 pinMode(led2 , OUTPUT);
 pinMode(led3 , OUTPUT);
 pinMode(buz , OUTPUT);
 pinMode(B1 , INPUT_PULLUP);
 pinMode(B2 , INPUT_PULLUP);
 pinMode(B3 , INPUT_PULLUP);

 time1 = millis();

}

void loop()
{
  if(millis() - time1 > 1000)
  {
    digitalWrite(buz ,digitalRead(buz)^1);
```

```
    time1 = millis();
  }

  if(digitalRead(B1) == 0)
  {
    delay(20);
    digitalWrite(led1 ,HIGH);
  }
  else
  {
    digitalWrite(led1 ,LOW);
  }

  if(digitalRead(B2) == 0)
  {
    delay(20);
    digitalWrite(led2 ,HIGH);
  }
  else
  {
    digitalWrite(led2 ,LOW);
  }

  if(digitalRead(B3) == 0)
  {
    delay(20);
```

```
    digitalWrite(led3 ,HIGH);
  }
  else
  {
    digitalWrite(led3 ,LOW);
  }
}
```

لا جديد في الكود سوى الفكرة المتعلقة بالتعليمة **millis()** حيث يتم إدخال إختبار الوقت المنقضـــي في كل مرة ومن ثم عكس حالة القطب الخاص بالزمور عبر إدخال قراءة حالة القطب 13 مع بوابة XOR والتي تعكس حالة القطب في كل مرة يتحقق فيها الشرط.

حل تمارين الفصل الثاني:

1

اكتب كود برمجي يتم من خلاله تغيير تدرج الألوان لكل لون من ألوان RGB LED وكتابة النسبة المئوية لكل لون على شاشة LCD، بحيث يكون لكل لون من الألوان مقاومة متغيرة للتحكم بزيادة وإنقاص نسبت اللون.

المقاومات المتغيرة التي سوف تتحكم بالألوان سوف يتم توصيلها مع الأقطاب التشابهية A0 ~ A2، بحيث تكون كل مقاومة مسؤولة عن التحكم بنسبة لون معين بدءا من اللون الأحمر وحتى اللون الأزرق.

أما الليدات فسوف يتم توصيلها مع الأقطاب ذات مخارج PWM وهي الأقطاب 11 ~ 9 ، أما بالنسبة لشاشة LCD فسوف يتم توصيل أقطابها الستة مع الأقطاب من الأرقام 8 ~ 3.

```
#include <LiquidCrystal.h>
#define redPin    9
#define greenPin  10
#define bluePin   11
LiquidCrystal lcd(3, 4, 5, 6, 7, 8);
int redValue   , greenValue   , blueValue   ;
int redValueR , greenValueR , blueValueR ;
```

```
int redPre    , greenPre    , bluePre    ;


void setup()
{
  lcd.begin(16, 2);
  pinMode(redPin  , OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin , OUTPUT);
}


void loop()
{
 DisplayLCD();
 calculat();
 setColor (redValue , greenValueR , blueValueR);
}


void DisplayLCD()
{
 lcd.print("Red=");
```

```
 lcd.print(redPre);


 lcd.setCursor(8, 0);
 lcd.print("Gre=");
 lcd.print(greenPre);


 lcd.setCursor(0, 1);
 lcd.print("Blu=");
 lcd.print(bluePre);
 lcd.clear();


}


void calculat()
{
//Read value from Resistor
redValueR   = analogRead(A0);
greenValueR = analogRead(A1);
blueValueR  = analogRead(A2);
```

```
// For PWM Output
redValue   = map(redValueR   ,0 , 1024 , 0 ,255);
greenValue = map(greenValueR ,0 , 1024 , 0 ,255);
blueValue  = map(blueValueR  ,0 , 1024 , 0 ,255);

//For print on LCD
redPre    = map(redValueR   ,0 , 1024 ,0 ,100);
greenPre  = map(greenValueR ,0 , 1024 ,0 ,100);
bluePre   = map(blueValueR  ,0 , 1024 ,0 ,100);
}

void setColor(int R ,int G ,int B)
{
  analogWrite(redPin   , redValue);
  analogWrite(greenPin, greenValue);
  analogWrite(bluePin , blueValue);
}
```

2

لدينا 8 حساسات نهاية شوط متصلة مع {16, 10 , 9 , 15 , 6 , 12 , 8 , 5} من لوحة الآردوينو ومعرفة كأقطاب دخل مع مقاومة رفع، كما يوجد لدينا LED متصل مع القطب 3 والذي يعتبر مخرج PWM والمطلوب:

كتابة كود يقرأ حالة الحساسات ويحول قيمتها لقيمة عشرية (8 bit = 255) ويسند القيمة لخرج نبضة PWM التي بدورها سوف تتحكم بقوة إضاءة LED وطباعة القيمة المقروءة من الحساسات وشدة الإضاءة على نافذة UART.

```
byte pinInput[8] = {5,8 ,12 ,6 ,15 ,9 ,10 ,16};
byte b , value ;

void setup()
{
  for(int i=0 ; i<8 ;i++)
  {pinMode(pinInput[i] , INPUT);}
  Serial.begin(9600);
  pinMode(3 , OUTPUT);
}

void loop()
{
  for(int i = 0 ; i < 8 ; i++)
  {
   b = digitalRead(pinInput[i]);
   bitWrite(value , i , b);
```

```
  }
analogWrite(3 , value);
Serial.println(value) ;
delay(200);
}
```

الهدف من هذا التمرين هو استذكار التعليمات المتعلقة بالبت من القراءة والكتابة وهذا الكود يعتبر آلية معاكسة لحالة شاشة العرض 7Segment فهنا سوف يتم قراءة حالة المفاتيح ووضع كلقيمة مقروءة في بت من بتات المتحول الذي يعبر عن الرقم الذي تشكله هذه المفاتيح بالقيمة الثنائية.

3

اكتب كود برمجي يطلب من المستخدم إدخال قيمة المتحول val عبر واجهة الاتصال التسلسلي UART ثم يتحقق البرنامج من القيمة فإن كانت القيمة ليست رقمية يعيد ويطلب إدخال القيمة، أما إن كانت القيمة رقمية فيقوم البرنامج بإدخالها على المعادلة الرياضية:

$$x = val * 105 + val^2$$

سنعتمد في هذا التمرين على التعليمات التي تعلمناها في نهاية الفصل الثاني والمتعلقة بالسلاسل والتعليمات الخاصة بها.

```
String val ;
void setup()
{  Serial.begin(9600);}
void loop()
{
if(Serial.available())
{
  val = Serial.readString();
  long x ;
  x = val.toInt();
    if(x > 0 )
     x = x*105 + sq(x);
     Serial.println(x);
}
delay(50);
}
```

4

أكتب كود برمجي يتم من خلاله إدخال أســـماء الطلاب و علاماتهم في مادة معينة (العلامة من عشرة) فيقوم البرنامج بتحديد مستوى كل طالب (العلامة بين 10 ~ 9 التقييم A، أما إن كانت بين 8 ~ 7 التقييم B ، وغير ذلك التقييم C) ويطبع جدول فيه أسماء الطلاب والعلامات والتقييم.

أيضا يأتي هذا التمرين لترسيخ الأفكار المتعلقة بعملية القراءة من المنفذ التسلسلي والعمليات المتعلقة بالسلاسل المحرفية وكيفية التعامل معها.

```
String student_name[10] ;
String student_mark[10] ;
char student_level[10];
int s_index = 0 ;
void setup()
{
  Serial.begin(9600);
}

void loop()
{
if(s_index < 10)
{
  Serial.println("Please enter the student name ");
  while(!Serial.available())
  {}
  student_name[s_index] = Serial.readString();
```

```
restart:
Serial.println("Please enter the student Mark ");
while(!Serial.available())
{}
  student_mark[s_index] = Serial.readString();
  long val =student_mark[s_index].toInt();
  if (val > 10 || val < 0)
  {
    Serial.println("Error");
    goto restart ;
  }
  else if ( val <= 10  && val >= 9 )
  {
    student_level[s_index] = 'A';
  }
  else if(val<9 && val >6)
  {
    student_level[s_index] = 'B';
  }
  else
  {
    student_level[s_index] = 'C';
  }
Serial.print(student_name[s_index]);
Serial.print("   ");
Serial.print(student_mark[s_index]);
```

```
  Serial.print("    ");
  Serial.println(student_level[s_index]);
  s_index++;
  }}
```

الكود كما هو واضح لا يضم الكثير من التفاصيل التي يمكن أن يطورها المستخدم حيث حاولنا تبسيط الفكرة لتكون بداية التعامل مع السلاسل ولكسر حاجز الخوف لدى الكثير من المبرمجين المبتدئين، ففي أول الكود عرفنا ثلاثة مصفوفات الأولى لأسماء الطلاب، الثانية فلتسجيل علامات الطلاب أما الثالثة والأخيرة فلوضع تقييم الطالب فيها، ولا ننسى المتحول `s_index` والذي سيكون المؤشر الأساسي للإنتقال بين عناصر المصفوفات ولتخزين القيم في مكانها الصحيح.

في حلقة البرنامج الأساسية يتم اختبار قيمة المؤشر قبل الدخول في البرنامج والذي سيطلب في البداية إدخال اسم الطالب ليتم تخزينه في السلسلة الخاصة بالأسماء، مع وجود حالة توقف إجباري للكود عند حلقة `while` والتي ينتظر عندها حتى يتم إدخال القيمة، فيقوم بتخزينها في مصفوفة الأسماء.

ثم نقوم بطلب علامة الطالب مع وضع شرط على القيمة بأن تكون ضمن المجال 10 ~ 0 وغير هذه القيم يتم طلب القيمة مرة ثانية من المستخدم، وبعد إدخال القيمة الصحيحة للعلامة يتم تحويلها لقيمة رقيمية ثم تقييم مستوى الطالب حسب العلامة المدخلة ثم يتم طبع اسم الطالب ودرجته وتقييمه والانتقال لإدخال قيمة جديدة وهكذا حتى نتم إدخال قيم العشر طلاب.

طبعا يمكن إضافة الكثير من الأمور لتطوير الكود منها مثلا ألا يزيد اسم الطالب عن 10 محارف أو التأكد من أن الاسم المدخل لا يحتوي أرقام، أو إضافة مفتاح لإعادة إدخال الاسم مرة أخرى وهكذا.... كل هذه الأفكار نتركها للطالب وقدرته على التطوير.

حل تمارين الفصل الثالث

1

ليكن لدينا ثلاثة محركات سيرفو متصلة مع لوحة الآردوينو على الأقطاب 2 , 3 , 4 على التوالي من لوحة الآردوينو، كما يوجد كباس لحظي متصل مع القطب 5 من لوحة الآردوينو، والمطلوب: كتابة كود برمجي يسمح للمستخدم إدخال رقم المحرك المراد التحكم به وموقعه وذلك عند الضغط على الكباس اللحظي.

```
#include <Servo.h>
#define servo1_pin   2
#define servo2_pin   3
#define servo3_pin   4
#define B_control    5

Servo servo1;
Servo servo2;
Servo servo3;
String motor_number , motor_position;
int  m_num , m_position;

void setup()
{
  Serial.begin(9600);
  servo1.attach(servo1_pin);
  servo2.attach(servo2_pin);
  servo3.attach(servo3_pin);
```

```
  pinMode(B_control ,INPUT_PULLUP);
}

void loop()
{
 if(digitalRead(B_control) == 0)
 {
  restart1:
  Serial.println("Please enter number of motor");
  Serial.println("Number between  1 ~ 3");
  while(Serial.available())
  motor_number = Serial.readString();
  m_num = motor_number.toInt();
  if (m_num > 3 || m_num <1)
  {
    Serial.println("Error number .... retry ");
    goto restart1;
  }

  restart2:
  Serial.println("   Please   enter   value      of
position");
  Serial.println("Number between  1 ~ 180");
  while(Serial.available())
  motor_position= Serial.readString();
  m_position    = motor_number.toInt();
```

```
  if (m_position > 180 ||  m_position <1)
  {
    Serial.println("value is error .... retry ");
    goto restart2;
  }
  control(m_num ,m_position);
 }
}

void control(int x , int y)
{
  Serial.print("Servo number = ");
  Serial.println(x);

  Serial.print("Servo position = ");
  Serial.println(y);

  switch (x)
  {
  case 1:
  servo1.write(y);
  break;
  case 2:
  servo2.write(y);
  break;
  case 3:
```

```
   servo3.write(y);
   break;
   }
}
```

2

لدينا محركين نوع Brushless متصلين مع الأقطاب 13 , 12 كما يوجد لدينا قبضة تشابهية محور X منها متصل مع القطب A0 من لوحة الآردوينو، والمطلوب كتابة كود برمجي للتحكم بسرعة المحركين بحيث أن التحريك من المنتصف ونحو اليمين للتحكم بسرعة المحرك الأول بينما من المنتصف ونحو اليسار للتحكم بسرعة المحرك الثاني، أما المنتصف فهو حالة إيقاف للمحركين.

```
#include <Servo.h>

#define brushless1_pin   12

#define brushless2_pin   13

#define x_joystick       A0


Servo brushless1;

Servo brushless2;


void setup()

{

  Serial.begin(9600);

  brushless1.attach(brushless1_pin);

  brushless2.attach(brushless2_pin);
```

```
  pinMode(x_joystick ,INPUT);
}

void loop()
{
int X_val = analogRead(x_joystick);

if(X_val > 535)
{
  X_val = map(X_val ,530 , 1023 , 100 , 180);
  brushless1.write(X_val);
  Serial.print("Brushless 1 , speed = ");
  Serial.print(map(X_val ,100 , 180 , 0 , 100));
  Serial.println(" % ");
}
else if (X_val < 490)
{
  X_val = map(X_val ,490 , 0 , 100 , 180);
  brushless2.write(X_val);
  Serial.print("Brushless 2 , speed = ");
```

```
  Serial.print(map(X_val ,100 , 180 , 0 , 100));
  Serial.println(" % ");
}
else
{
 brushless1.write(0);
 brushless2.write(0);
 Serial.println("Motors are stop");
}
 delay(50);
}
```

ننتبه فقط في الكود أننا نبدأ قيمة الخرج باتجاه ESC من القيمة 100 لأنه قبل هذه القيمة لن يتحرك المحرك.

3

لدينا محرك خطوي يتحكم به عبر دارة القيادة A4988 والتي بدورها تتصل مع لوحة الآردوينو عبر الأقطاب 10 للأمر STEP والقطب 11 للأمر DIR ،كما يوجد كباسين لحظيين متصلين مع الأقطاب 2 , 3 والمطلوب كتابة كود للتحكم بجهة دوران المحرك يمينا أو يسار حسب ضغط المفاتيح وبخطوة ثابته.

يتوفر الكثير من المكتبيات الخاصة بالتعامل مع دارة القيادة **A4988** الخاصة بالمحرك الخطوي سنأخذ إحداها ونطبق عليها المشروع الذي نريد.

```
#include <Arduino.h>
//#include "DRV8834.h"
//#include "DRV8825.h"
#include "A4988.h"

#define  MOTOR_STEPS 200
#define  DIR   11
#define  STEP  10
#define  MS1   9
#define  MS2   8
#define  MS3   7
#define  B1    2
#define  B2    3

A4988  stepper(MOTOR_STEPS,  DIR,  STEP,  MS1,  MS2,
MS3);
```

```
void setup()
{
  Serial.begin(9600);
  pinMode(B1 , INPUT_PULLUP);
  pinMode(B2 , INPUT_PULLUP);

  stepper.setRPM(1);
  stepper.setMicrostep(1);
}

void loop()
{
  if(digitalRead(B1)==0)
  {
    delay(15);
    stepper.move(1);     // CW
   // stepper.rotate(180);   Move to the angle
  }

  if(digitalRead(B2)==0)
  {
    delay(15);
    stepper.move(-1);    // CCW
   // stepper.rotate(-180);   Move to the angle
  }
}
```

حل تمارين الفصل الرابع

1 **اكتب كود برمجي لحساس أمواج فوق صوتية متصل مع لوحة الآردوينو، يقوم الحساس وعند تشغيل الكود البرمجي للمرة الأولى بحساب المسافة البدائية بين الحساس وبين أقرب حاجز يقع ضمن مجال الحساس، ثم يقوم بالتحسس لأي جسم جديد سوف يقع بين الحساس والحاجز البدائي فيشغل زمور باستخدام التعليمة `tone()` .**

بفرض أن أقطاب الحساس متصلة مع لوحة الآردوينو بحيث أن القطب Trig = 3 والقطب Echo = 4، بينما يتم توصيل الزمور مع القطب buzzer = 5 وبالتالي يكون الكود بالشكل التالي:

```
#define  Echo     4
#define  Trig     3
#define  buzzer   5
int old_distance , new_distance , distance ,flag ;

void setup()
{
pinMode(Trig    , OUTPUT);
pinMode(buzzer , OUTPUT);
}
void loop()
{
if(flag == 0 )
```

```
  {
   old_distance = Distance_cal();
   flag = 1 ;
  }

new_distance = Distance_cal();
if(new_distance > (old_distance + 5))
  {
    tone(buzzer , 400 , 1000);
  }
}

int Distance_cal()
{
  int duration ;
  digitalWrite(Trig,LOW);
  delayMicroseconds(2);

  digitalWrite(Trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig,LOW);
  duration = pulseIn(Echo,HIGH);

  distance = duration*0.034/2 ;
  return distance;
}
```

2

لدينا ثلاثة حساسات PIR متوضعة بحيث يغطي كل منها قطاع زاوي قدره 60 درجة، كما يوجد محرك سيرفو وعليه كاميرا، والمطلوب كتابة كود برمجي يقوم بتحريك الكاميرا بالاتجاه الذي يتحسس عنده أحد الحساسات.

في هذا المشروع سيكون لدينا **ثلاثة حساسات** تعمل بالأشعة تحت الحمراء سيتم توصيلها مع الأقطاب **3 , 4 , 5** من لوحة الآردوينو ، أما **محرك السيرفو** فسيتم توصيله مع القطب **6** من لوحة الآردوينو، يجب ضبط الحساسات لكي تعطي نبضة فقط عندما يمر أمامها جسم ويستمر إعطاء الحساس للنبضة ما دام هناك جسم أمام الحساس، فيكون الكود البرمجي على الشكل التالي:

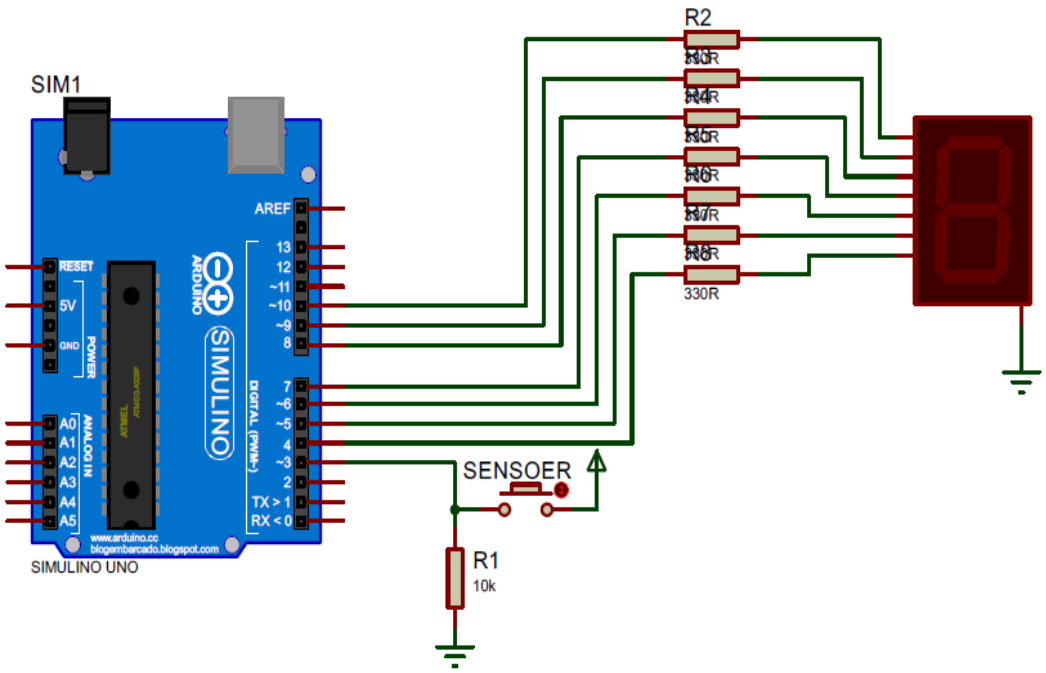
```
#include <Servo.h>
#define  sensor1   3
#define  sensor2   4
#define  sensor3   5
#define  servoPin  6
Servo myservo;
int s1 , s2 , s3 ;
void setup(){
myservo.attach(servoPin);
pinMode(sensor1 , INPUT);
pinMode(sensor2 , INPUT);
pinMode(sensor3 , INPUT);}
void loop()
{
s1 = digitalRead(sensor1);
```

```
s2 = digitalRead(sensor2);
s3 = digitalRead(sensor3);
if(s1 == 1)
  {
   myservo.write(30);
  }
else if(s1 == 1 && s2 == 1)
  {
   myservo.write(60);
  }
else if(s2 == 1)
  {
   myservo.write(90);
  }
else if(s2 == 1 && s3 == 1)
  {
   myservo.write(120);
  }
else if(s3 == 1)
  {
   myservo.write(150);
  }
else
  {
   myservo.write(90);
  }}
```

3

لدينا حساس لمعرفة درجة رطوبة التربة متصل مع لوحة الآردوينو، كما يوجد لدينا ريليه للتحكم بتدفق مياه الري للتربة، والمطلوب كتابة كود برمجي يشغل الريليه عند انخفاض درجة رطوبة التربة دون المستوى المطلوب لمدة ثلاثة دقائق أو حتى يعطي الحساس أمر بأن الرطوبة للتربة وصلت للحد المطلوب، مع عرض الوقت المتبقي على خانة واحدة من شاشة 7Segment .

سنقوم بتوصيل الحساس مع القطب 3 من لوحة الآردوينو والذي سوف يلعب دور مفتاح يعمل عند وصول رطوبة التربة للدرجة المطلوبة، أما شاشة العرض 7Segment فهي من نوع سالب مشترك وسوف يتم وصلها مع الأقطاب 4 ~10 وهي أقطاب خرج فيكون المخطط النظري للمشروع على الشكل التالي:



أما الكود البرمجي:

```
#define  sensor   3
#define  Faucet   11
// Array for print value on 7 segment { 1 , 2 , 3 , A }
byte segment_val [] =
{0b0000110,0b1011011,0b1001111,0b1110111};

// Array for pin 7 segment with Arduino
byte segment_pin [7] = {10,9,8,7,6,5,4};
int sensorVal , RealTime ,bitVal ,flag1,segmentPlay;

void setup()
{
// Configeration pins
for(int i=0 ;i<7 ;i++)
  {
    pinMode(segment_pin[i] , OUTPUT);
  }
pinMode(sensor , INPUT );
pinMode(Faucet , OUTPUT);
Serial.begin(9600);
RealTime =180;
}

void loop()
{
sensorVal = digitalRead(sensor);
if((sensorVal == 1)  && (RealTime > 0))
```

```
{
 digitalWrite(Faucet , HIGH);
 RealTime--;
 RealTime = constrain(RealTime , 0 , 180);
  if(RealTime >= 120 && RealTime > 0)
  {
    segmentPlay = 2 ; // 3 on Display
  }
  else if ((RealTime < 120)&&(RealTime>= 60))
  {
    segmentPlay = 1 ; // 2 on Display
  }
  else if ((RealTime < 60) && (RealTime>0))
  {
   segmentPlay = 0 ;  // 1 on Display
  }
delay(1000);          // Delay one secound evry one loop
Serial.println(RealTime);
}

else
{
 RealTime = 180;
 segmentPlay = 3;        //choose A to print on Segment
 digitalWrite(Faucet , LOW);
}
Segment(segmentPlay);   // Call Function what I want to
print on Display
```

```
}

void Segment(int x)
{
int a = segment_val[x];
for(int i = 0 ; i<7 ; i++)
{
  digitalWrite(segment_pin[i] , bitRead(a ,i));
}
}
```

المحتويات

الآردوينو ... كما لم تعرفه من قبل

م.محمود مسلماني

للتواصل AMJAD_BOOK
قناة المكتبة AMJAD_BOOKS