Erasmus+ KA-202

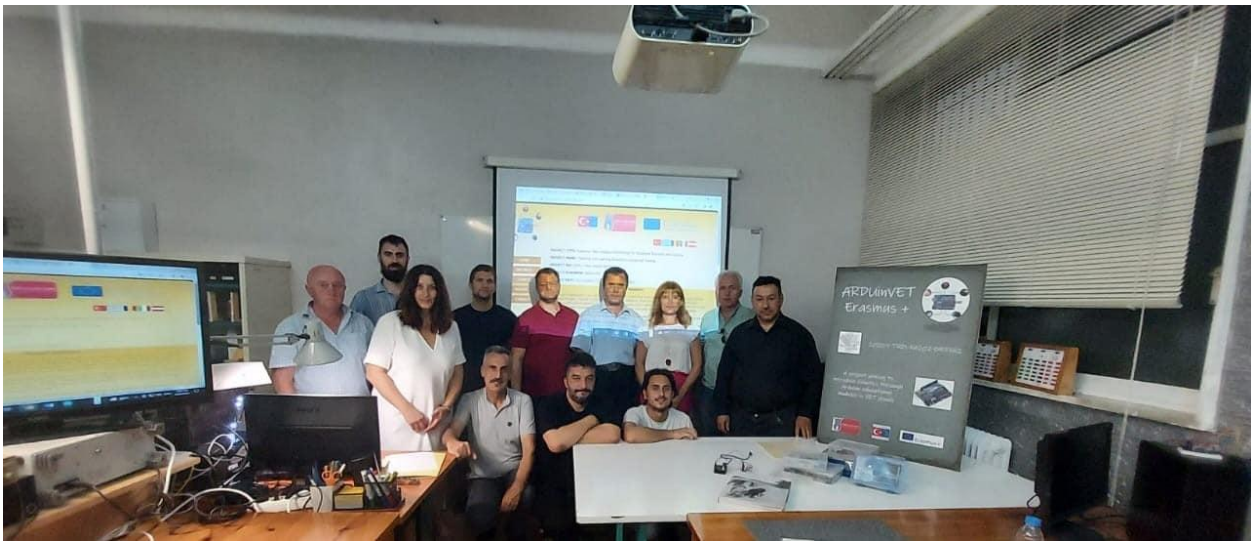
Strategic Partnerships Project for Vocational Education and Training

Project Title: “Teaching and Learning Arduinos in Vocational Training”

Project Acronym: “ ARDUinVET ”

Project No: “2020-1-TR01-KA202-093762”

ARDUinVET GUIDEBOOK



"Bu proje Erasmus+ Programı kapsamında Avrupa Komisyonu tarafından desteklenmektedir. Ancak burada yer alan görüşlerden Avrupa Komisyonu ve Türkiye Ulusal Ajansı sorumlu tutulamaz."

"This project is Funded by the Erasmus+ Program of the European Union. However, European Commission and Turkish National Agency cannot be held responsible for any use which may be made of the information contained therein"

COORDINATOR:

Gölbaşı Mesleki ve Teknik Anadolu Lisesi (Ankara / TURKEY)

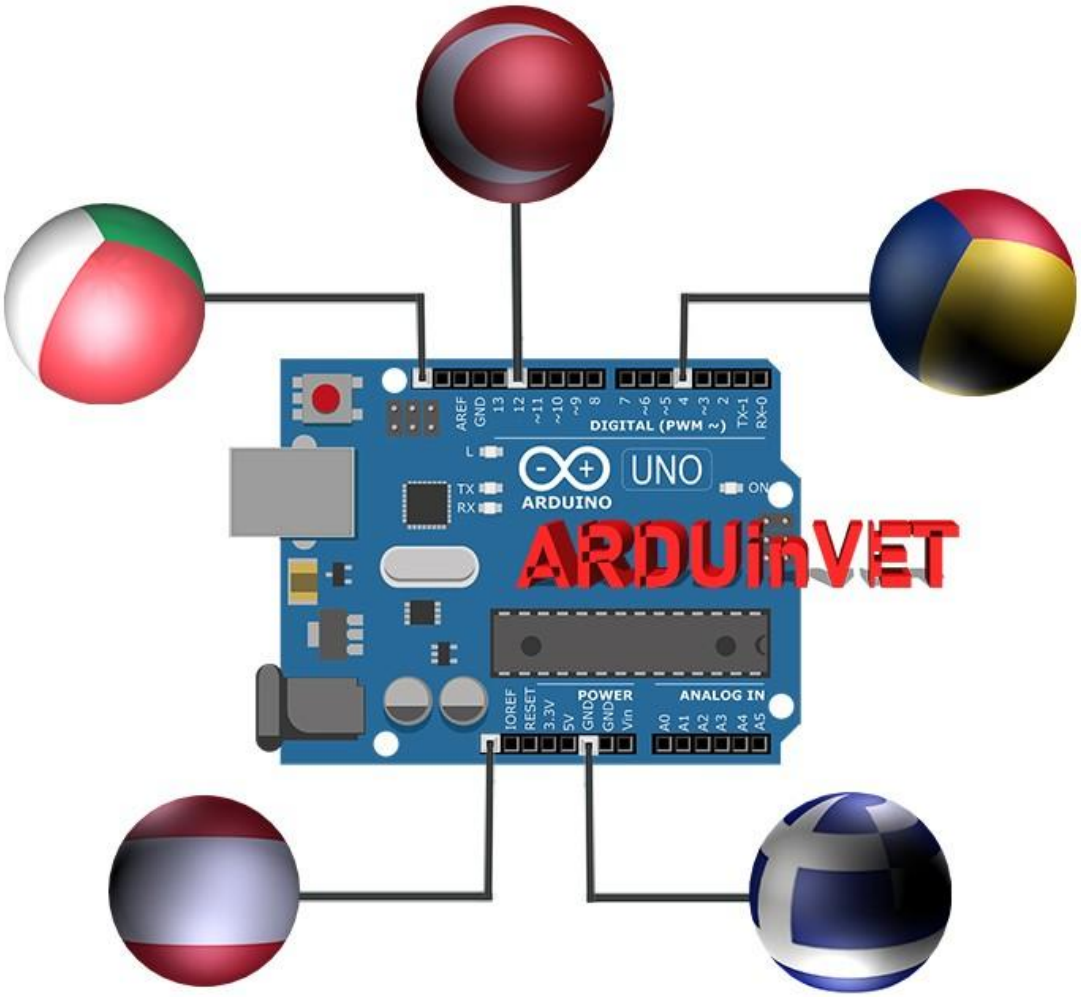
PARTNERS:

2 EK Peiraia (Piraeus / Greece)

Liceul Tehnologic Grigore Moisil Braila (Braila / Romania)

Istituto di Istruzione Superiore Einstein De Lorenzo Potenza (Potenza / Italy)

HTBLA Wolfsberg / (College for Engineering Wolfsberg) (Wolfsberg / Austria)



Project Summary
"Teaching and Learning Arduinos in Vocational Training"

Technology is developing very rapidly day by day. It is a fact that technological developments force a human to develop himself / herself continuously in his/her profession. More than any other field, electronic and information and communication technology are developing more rapidly. These developments have led to the production of more complex control systems. Arduinos are now used to control these complex systems.

Since we are educational institutions that teach technology, we have to follow the developments in the world. "Teaching and Learning Arduinos in Vocational Training" Project aims to adapt arduino applications to vocational training and to develop a more efficient training set and a guidebook for the laboratories and the workshops of vocational & technical education students.

The main objective of this project is to develop a good practice guide book and a Arduino training set, to introduce arduino training models to other participants during their visits to the host country, to compare different educational systems and training methods with other participating schools and to share best practices. Participants of the projects; electrical, electronic, ICT, automation VET teachers. Participants are teaching arduino in VET schools. There are one coordinator and a total of 4 partners in the project and partners are vocational schools from Turkey, Italy, Romania, Austria, Greece. The total number of mobilities in the project is 40. A total number of 5 TPMs will be realized throughout the project and each partner will participate in each TPM with 2 participants. One TPM will be held in each country.

With the project activities, it will be ensured that the best practices will be shared for arduino teaching and the partners will adapt them to their educational environments. The project partners will produce Arduino experimental kits and a best practice Guidebook. Our project activities aim to provide a successful learning and teaching environment for all VET teachers and students. As a condition of successful learning, teachers need to strengthen their role in facilitating learning. Teachers need new experimental kits and modules for innovation, teamwork, feedback and evaluation. Teachers should be given this opportunity for continuing professional development.

Main outputs of the project are; an Arduino Training Set for arduino lessons in vocational education and a GuideBook for this set, a project website, a project DVD. The methodology to be used in the execution of the project is "Make-Develop-Share". In our project, good practices will be made first, then developed, and finally shared. Everything is open and transparent in our Project. Each task and responsibility will be recorded or written in the project. The products of the project are not only written or documentary products, but also a practical Arduino training set and kits.

In the long term, we aim to disseminate the project to teachers, students and vocational education schools, local educational institutions, electronic and ICT labor market. We believe that the results and products of our projects will have short and long-term impacts on vocational education and labor market by the dissemination activities. The total budget of the project with 5 partners is 59,000 Euros.

CONTENTS:

Co-funded by the Erasmus+ Programme of the European Union

PROJECT MODULES and KITS of ARDUinVET

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Project Title: “Teaching and Learning Arduinos in Vocational Training”

Project Acronym: “ ARDUinVET ”

Project No: “2020-1-TR01-KA202-093762”

INTRODUCTION to ARDUINOS

(BASICS of ARDUINOS)

INTRODUCING the ARDUINO

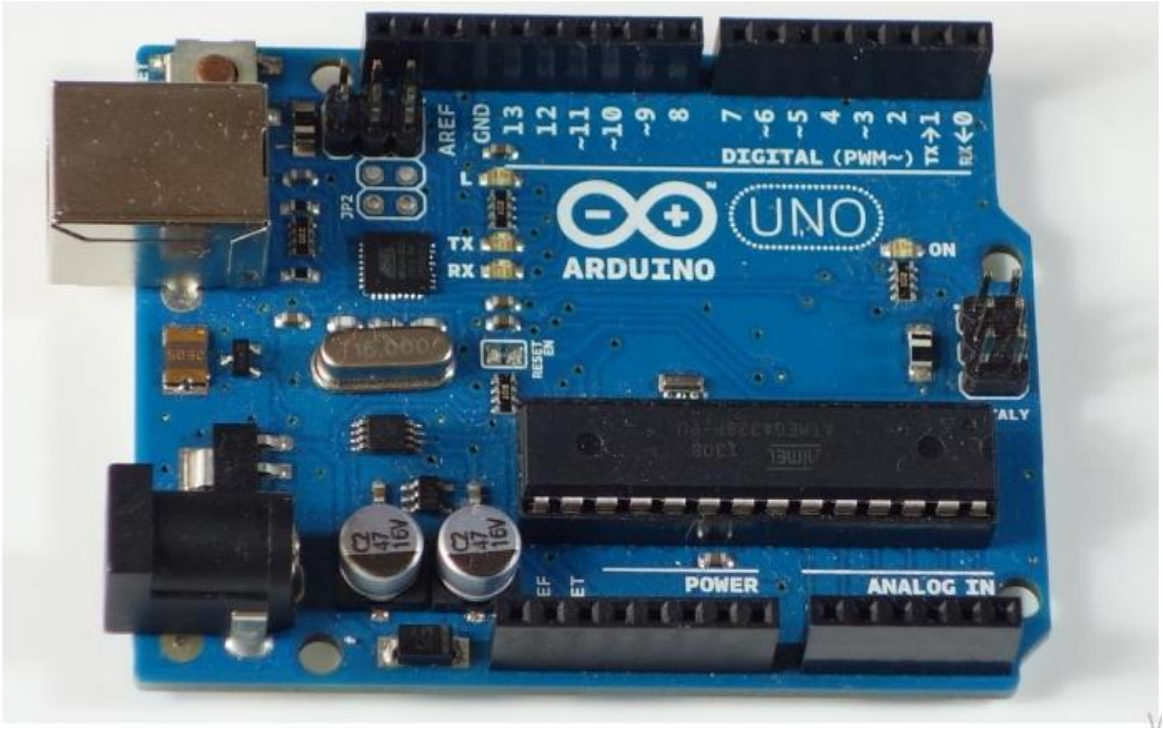
Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

On te other words, the Arduino is a small computer that you can program to read information from the world around you and send commands to the outside world. All of this is possible because you can connect several devices and components to the Arduino to do what you want. You can do amazing projects with it, there is no limit for what you can do, and using your imagination everything is possible!

In simple terms, the Arduino is a tiny computer system that can be programmed with your instructions to interact with various forms of input and output. The current Arduino board model, the Uno, is quite small in size compared to the average human hand.

What is an Arduino?

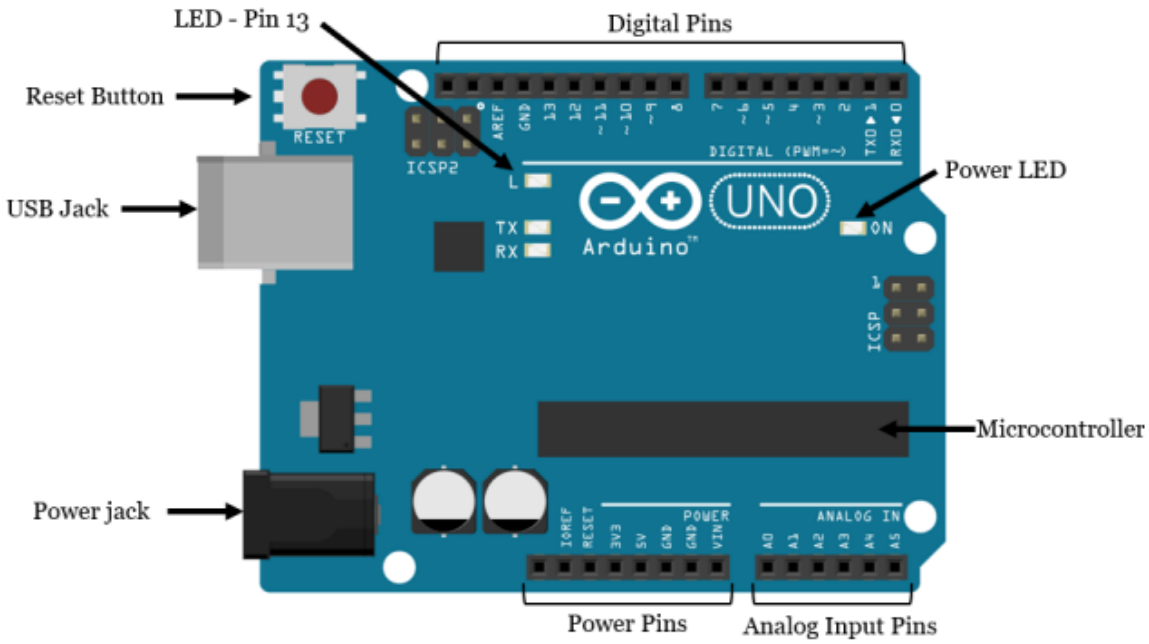
The Arduino is the board shown in the figure below.



Basically, it is a small development board with a brain (also known as a microcontroller) that you can connect to electrical circuits. This makes it easy to read inputs – read data from the outside – and control outputs - send a command to the outside. The brain of this board (Arduino Uno) is an ATmega328p chip where you can store your programs that will tell your Arduino what to do.

Exploring the Arduino Uno Board

In the figure below, you can see an Arduino board labeled. Let's see what each part does.



• **Microcontroller:** the ATmega328p is the Arduino brain. Everything on the Arduino board is meant to support this microcontroller. This is where you store your programs to tell the Arduino what to do.

• **Digital pins:** Arduino has 14 digital pins, labeled from 0 to 13 that can act as inputs or outputs.
o When set as inputs, these pins can read voltage. They can only read two states: HIGH or LOW.
o When set as outputs, these pins can apply voltage. They can only apply 5V (HIGH) or 0V (LOW).

• **PWM pins:** These are digital pins marked with a ~ (pins 11, 10, 9, 6, 5 and 3). PWM stands for "pulse width modulation" and allows the digital pins output "fake" varying amounts of voltage. You'll learn more about PWM later.

• **TX and RX pins:** digital pins 0 and 1. The T stands for "transmit" and the R for "receive". The Arduino uses these pins to communicate with other electronics via Serial. Arduino also uses these pins to communicate with your computer when uploading new code. Avoid using these pins for other tasks other than serial communication, unless you're running out of pins.

• **LED attached to digital pin 13:** This is useful for an easy debugging of the Arduino sketches.

• **TX and RX LEDs:** these leds blink when there are information being sent between the computer and the Arduino.

• **Analog pins:** the analog pins are labeled from A0 to A5 and are often used to read analog sensors. They can read different amounts of voltage between 0 and 5V. Additionally, they can also be used as digital output/input pins like the digital pins.

• **Power pins:** the Arduino provides 3.3V or 5V through these pins. This is really useful since most components require 3.3V or 5V to operate. The pins labelled as "GND" are the ground pins.

• **Reset button:** when you press that button, the program that is currently being run in your Arduino restarts. You also have a Reset pin next to the power pins that acts as reset button. When you apply a small voltage to that pin, it will reset the Arduino.

• **Power ON LED:** will be on since power is applied to the Arduino.

• **USB jack:** you need a male USB A to male USB B cable (shown in figure below) to upload programs from your computer to your Arduino board. This cable also powers your Arduino.

• **Power jack:** you can power the Arduino through the power jack. The recommended input voltage is 7V to 12V. There are several ways to power up your Arduino: for example; rechargeable batteries, disposable batteries, wall-warts and solar panel.

Arduino Features

Arduino Uno; has Atmel Atmega 328P microcontroller and also has USB connection input, power jack input, reset buton.. Arduino has everything that a microcontroller should have.

Microcontroller	Atmega328P
Working voltage	5V
Input voltage (recommended)	7-12V
Input voltage (limit)	6-20V
Digital input / output pins	14

PWM input / output pins	6
Analog input pin	6
Dc current per input / output pin	20mA
DC current for 3.3V	50mA
Flash memory	32 KB
Sram	2KB
EEPROM	1 KB
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figure: Arduino Uno Features

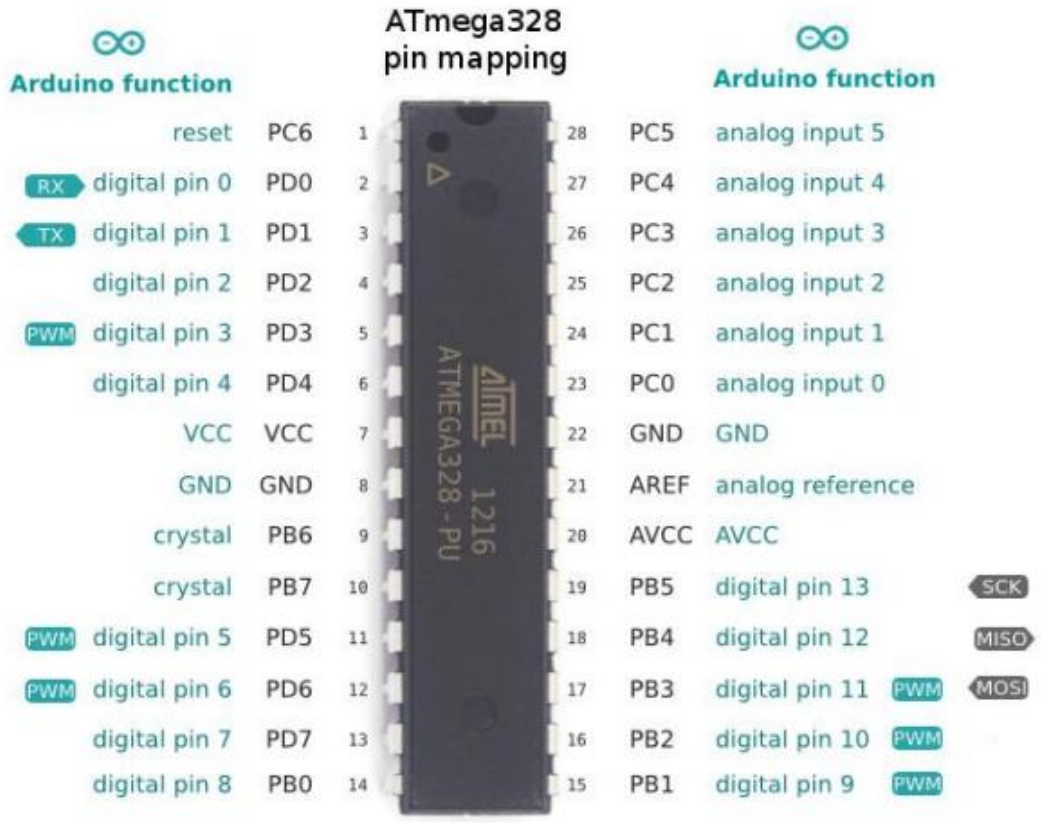


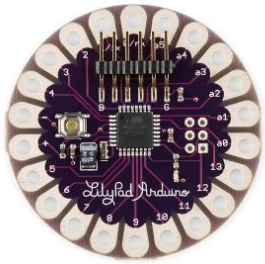
Figure: Atmega 328P Pins

OTHER TYPES of ARDUINOS

ARDUINO MEGA

It has the Atmega 2560 microcontroller on it. It has 54 digital input-output pins, 16 analog inputs, 4 hardware serial ports, and a 16 mhz crystal oscillator. It is powered by both USB and DC adapter. Generally, the card, which has the same features as the Arduno UNO, is preferred in larger projects because it has more pins.	

ARDUINO LILYPAD

Lilypad is designed to be sewn on dresses and fabrics. In this way, it can be used in interesting projects that can be designed to be wearable. It has an Atmega 168V microcontroller on it.	

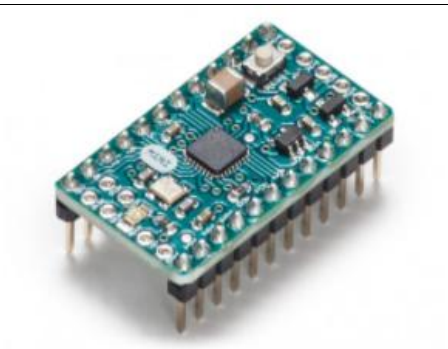
ARDUINO ETHERNET

It has an Ethernet chip and an Ethernet port for making internet-connected projects. There is also an SD-Card slot on the card, which has the Atmega 328 model as a microcontroller.	

ARDUINO BLUETOOTH

There is a Bluetooth module on Arduino BT, ideal for making applications communicating with the Bluetooth protocol. This module can also be used to program Arduino via Bluetooth.	

ARDUINO MİNİ

It is an Arduino model designed to be operated on a breadboard or integrated into another design. There is Atmega 168 or Atmega 328 model microcontroller on it. It is ideal for applications where small size is particularly important.	

ARDUINO NANO

It is a very small and designed model suitable for applications on the circuit board, and has an Atmega 328 or Atmega 168 microcontroller, voltage regulator, serial to USB converter chip, DC voltage input port and mini USB port.	

ARDUINO LEONARDO

It is one of the Arduino boards, which contains an Atmega 32u4 microcontroller on the Arduino Leonardo and does not require an additional chip for USB connection. With 20 digital inputs / outputs and 12 analog inputs, the microcontroller on the board has a surface mount cover. Thanks to its USB connection capabilities, Leonardo can be connected to the	

computer as a mouse or keyboard.	

ARDUINO ESPLORA

Esplora is an Arduino board that contains various sensors, unlike the others. Thanks to the sensors on the card, it is possible to perform many applications without the need for other additions and excessive electronic knowledge. Esplora is equipped with a slide potentiometer, light and sound sensor, temperature sensor, sound generator, 2-axis mini analog joystick, 3-color LED and an accelerometer. Esplora is also equipped with Atmega 32U4 AVR microcontroller like Leonarda. Applications that can act as a mouse or keyboard can be developed when connected to a computer with its micro USB connection.	

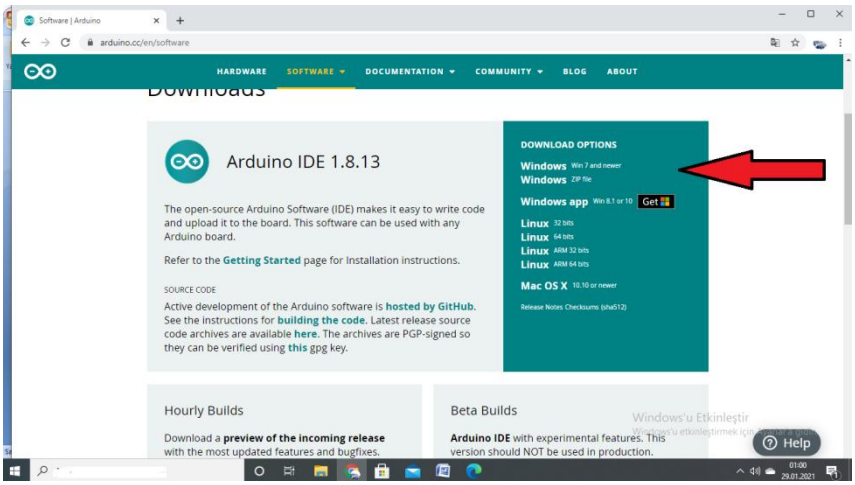
Downloading the Arduino IDE

The Arduino IDE (Integrated Development Environment) is where you develop your programs that will tell the Arduino what to do.

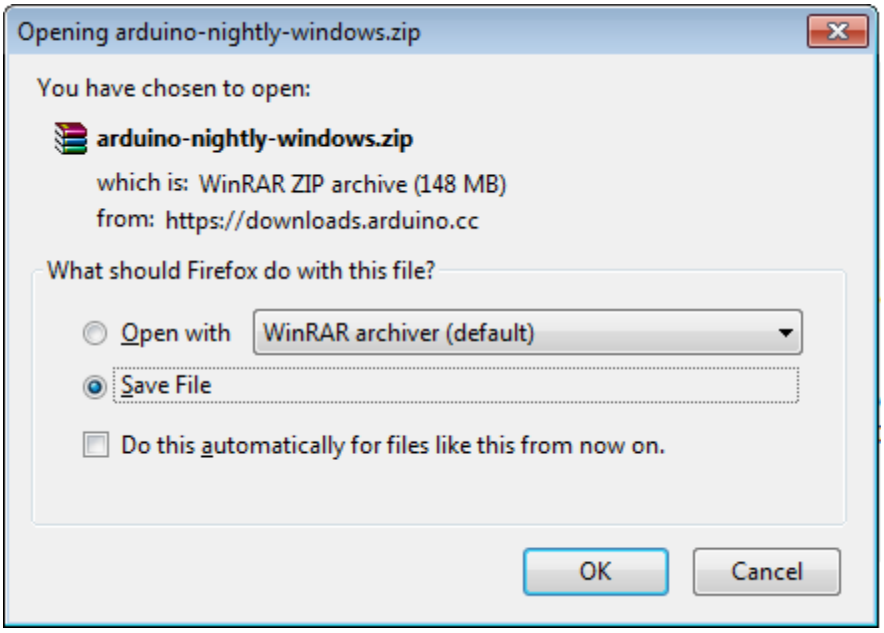
To install the Arduino IDE for Windows, we have to follow instructions.

You can load new programs onto the main chip, the ATmega328p, via USB using the Arduino IDE.

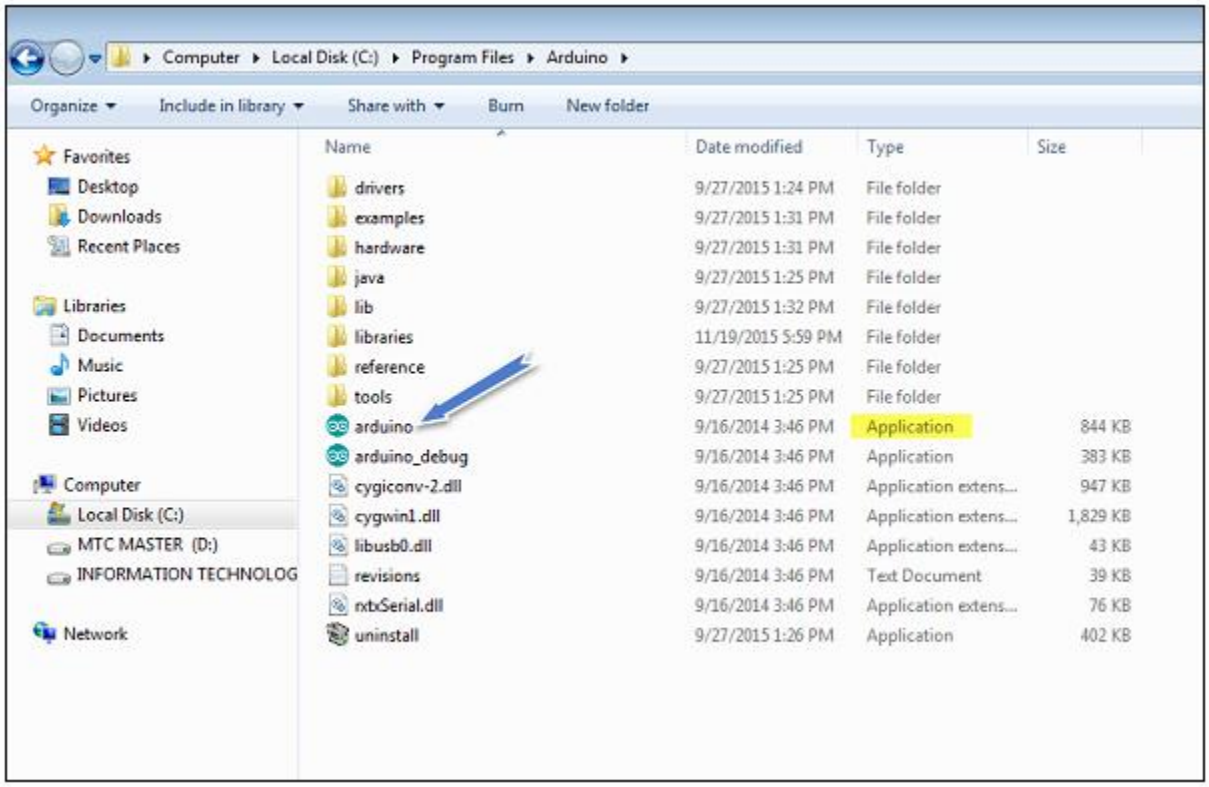
To download the Arduino IDE, please click on the following link: https://www.arduino.cc/en/Main/Software.



Select which Operating System you're using and download it. After our Arduino IDE software is downloaded, we need to unzip the folder.



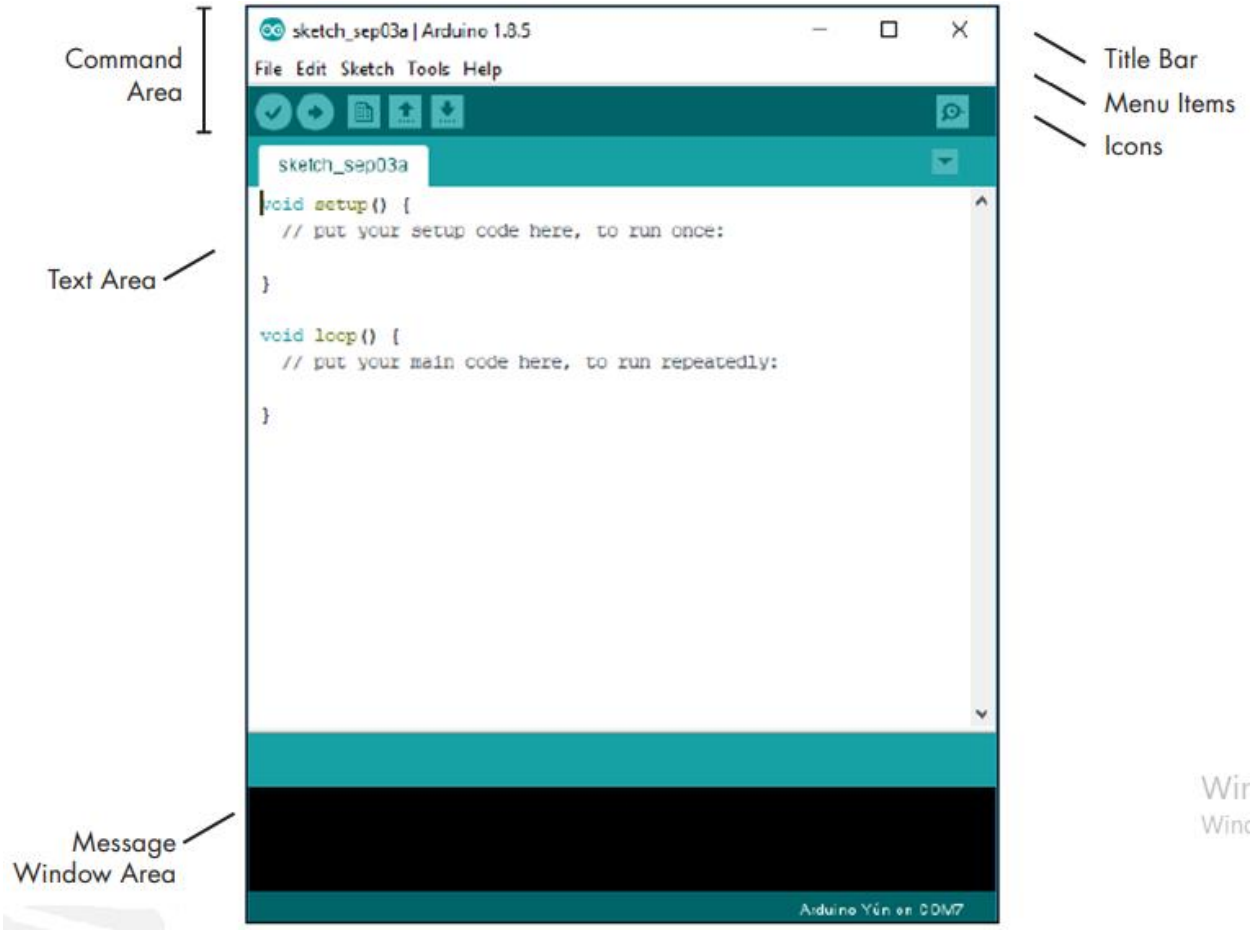
Inside the folder, we can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE. Then, simply follow the installation wizard to install the Arduino IDE.



Arduino IDE Window to Write Programs

When you first open the Arduino IDE, you should see something similar to the figure below.

As shown in Figure below, the Arduino IDE resembles a simple word processor. The IDE is divided into three main areas: the command area, the text area, and the message window area.



Menu Items

As with any word processor or text editor, you can click one of the menu items to display its various options.

File: Contains options to save, load, and print sketches; a thorough set of example sketches to open; as well as the Preferences submenu.

Edit: Contains the usual copy, paste, and search functions common to any word processor

Sketch: Contains the function to verify your sketch before uploading to a board, and some sketch folder and import options.

Tools: Contains a variety of functions as well as the commands to select the Arduino board type and USB port.

Help: Contains links to various topics of interest and the version of the IDE.

What the Sketch is

An Arduino sketch is a set of instructions that you create to accomplish a particular task; in other words, a sketch is a program.

The sketch is nothing more than a set of instructions for the Arduino to carry out. Sketches created using the Arduino IDE are saved as .pde files. To create a sketch, you need to make the three main parts: Variable declaration, the Setup function, and the main Loop function.

Arduino IDE Toolbar Buttons

Below the menu toolbar are six icons. Mouse over each icon to display its name. The icons, from left to right, are as follows:

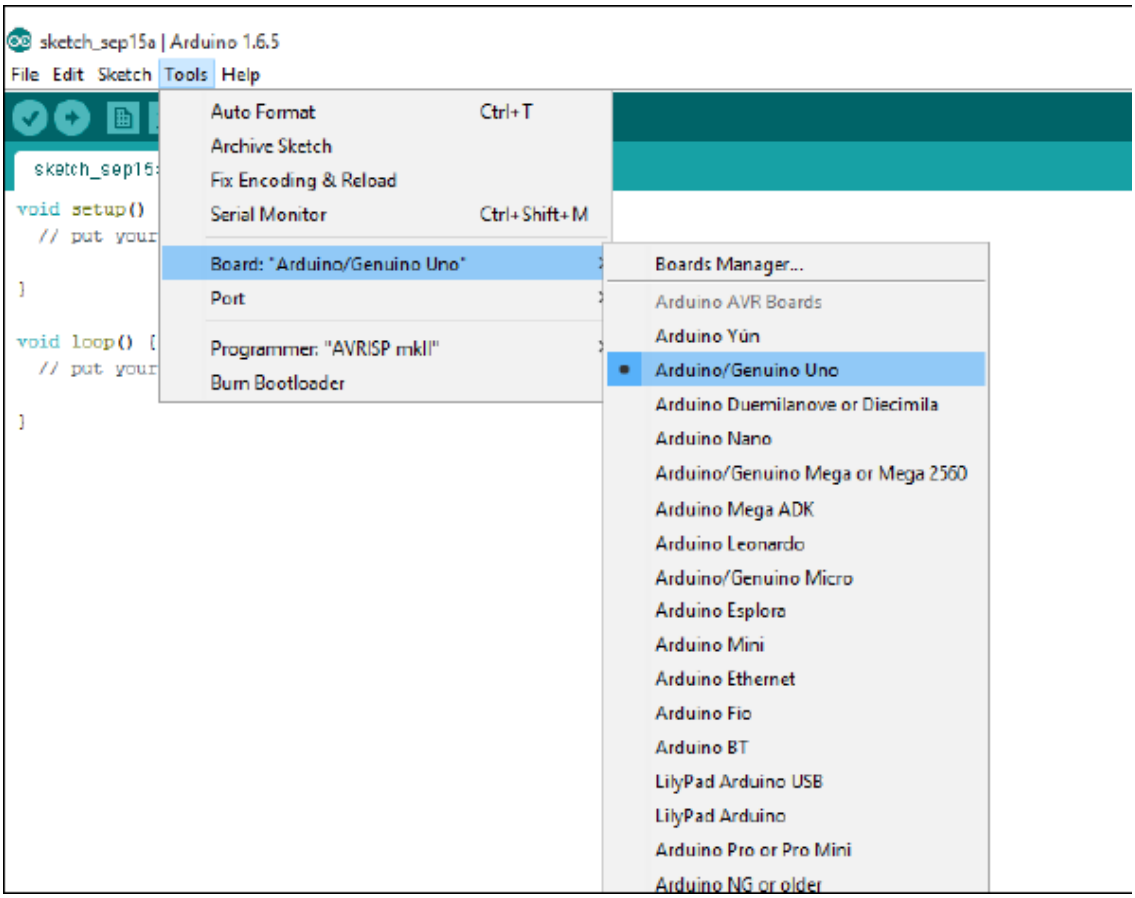
	**Verify (Compile):** Click this to check that the Arduino sketch is valid and doesn't contain any programming mistakes.
	**New:** Click this to open a new blank sketch in a new window.
	**Open:** Open Click this to open a saved sketch.
	**Save:** Click this to save the open sketch. If the sketch doesn't have a name, you will be prompted to create one.
	**Upload:** Click this to verify and then upload your sketch to the Arduino board.
	**Serial Monitor:** Click this to open a new window for use in sending and receiving data between your Arduino and the IDE.

Connecting your Arduino

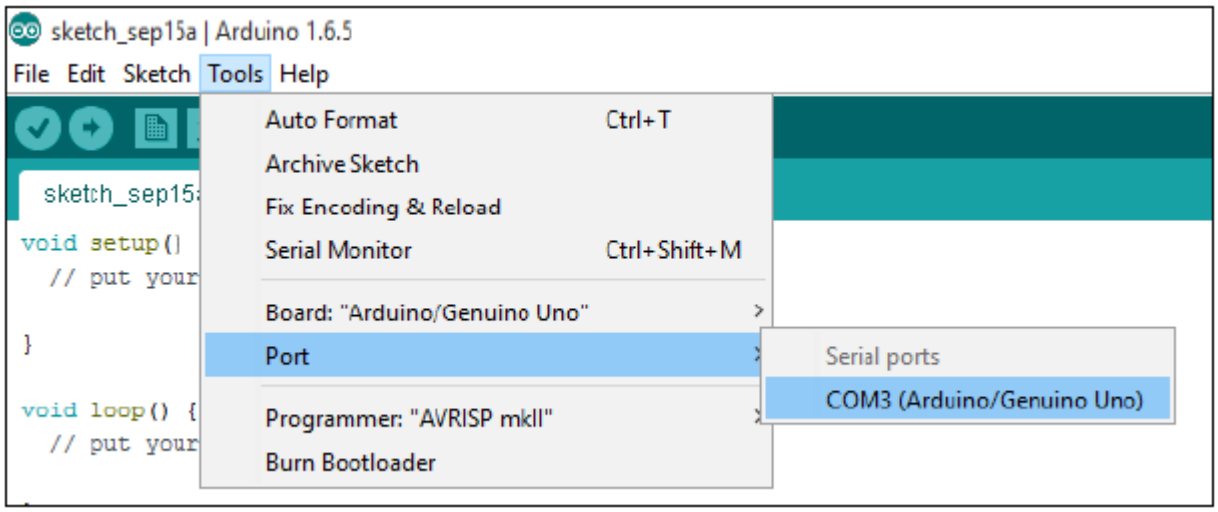
Connect your Arduino UNO to your computer via USB.

After connecting your Arduino with a USB cable, you need to make sure that the Arduino IDE has selected the right board.

In our case, we're using Arduino Uno, so we should go to **Tools ▸ Board: ▸ Arduino/Genuino Uno.**



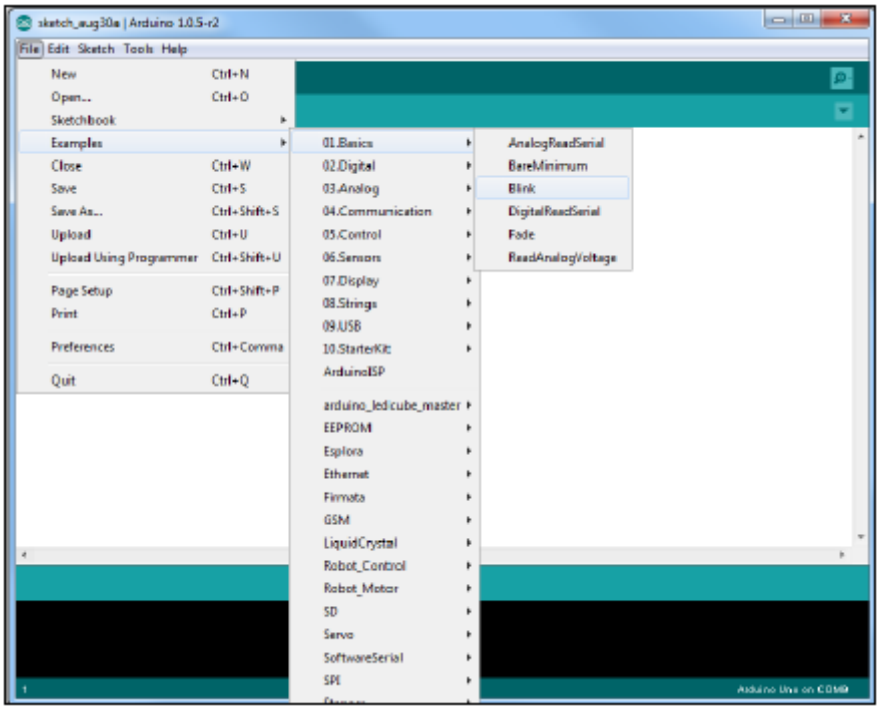
Then, you should select the serial port where your Arduino is connected to. Go to **Tools ▸ Port** and select the right port.



Uploading an Arduino Sketch

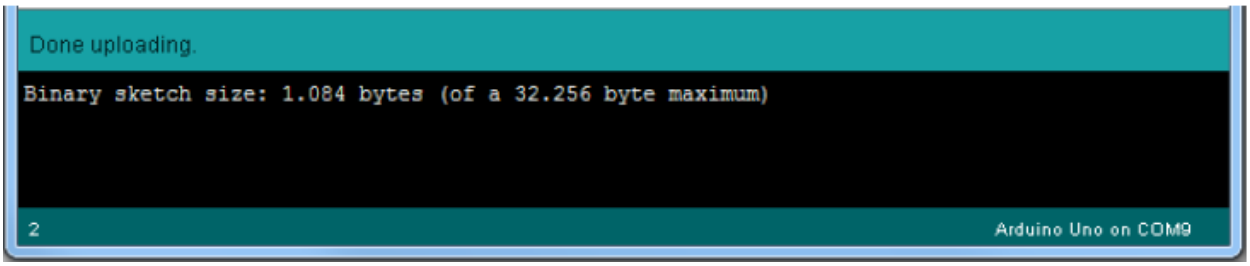
To show you how to upload code to your Arduino board, we'll show you a simple example. This is one of the most basic examples – it consists in blinking the on-board LED or digital pin 13 every second.

1. Open your Arduino IDE.
2. Go to **File ▸ Examples ▸ 01.Basics ▸ Blink**



By default, the Arduino IDE comes pre-configured for the Arduino UNO. Click the **Upload** button and wait a few seconds.

After a few seconds, you should see a **Done uploading** message.



This code simply blinks the on-board LED on your Arduino UNO (highlighted with red color). You should see the little LED turn on for one second, and turn off for another second repeatedly.

Control an Output and Read an Input

An Arduino board contains digital pins, analog pins and PWM pins.

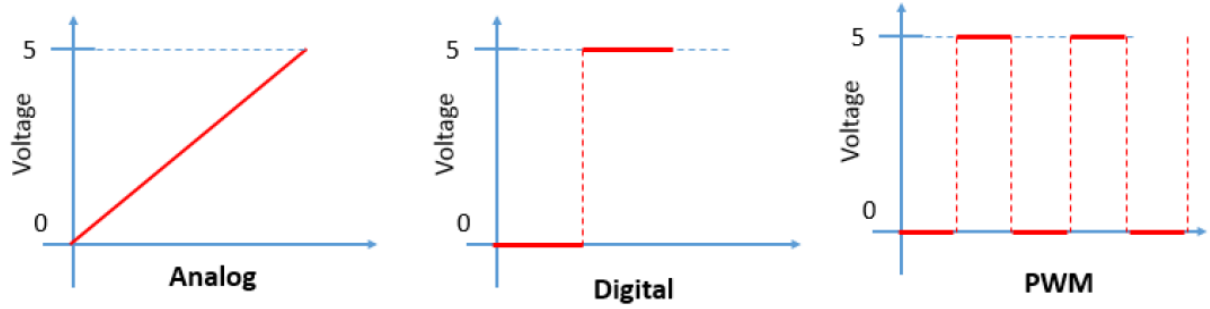
Difference between digital, analog and PWM

In **digital pins**, you have just two possible states, which are on or off. These can also be referred as High or Low, 1 or 0 and 5V or 0V.

For example, if an LED is on, then, its state is High or 1 or 5V. If it is off, you'll have Low, or 0 or 0V.

In **analog pins**, you have unlimited possible states between 0 and 1023. This allows you to read sensor values. For example, with a light sensor, if it is very dark, you'll read 1023, if it is very bright you'll read 0 If there is a brightness between dark and very bright you'll read a value between 0 and 1023.

PWM pins are digital pins, so they output either 0 or 5V. However these pins can output "fake" intermediate voltage values between 0 and 5V, because they can perform "Pulse Width Modulation" (PWM). PWM allows to "simulate" varying levels of power by oscillating the output voltage of the Arduino.



Controlling an output

To control a digital output you use the digitalWrite() function and between brackets you write, the pin you want to control, and then HIGH or LOW.

To control a PWM pin you use the analogWrite() function and between brackets you write the pin you want to control and a number between 0 and 255.

Reading an input

To read an analog input you use the function analogRead() and for a digital input you use digitalRead().

Note: The best way for you to learn Arduino is practising. So, make many projects and start building something.

Erasmus+ KA-202

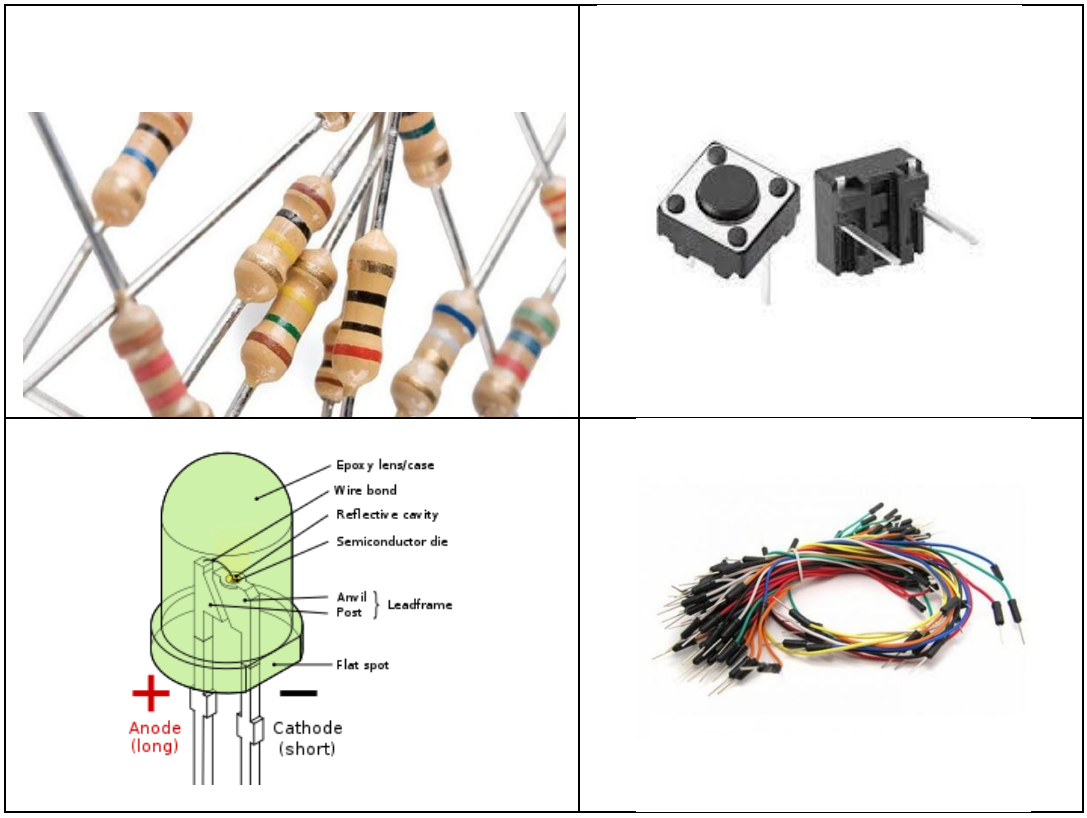
Strategic Partnerships Project for Vocational Education and Training

Project Title: "Teaching and Learning Arduinos in Vocational Training"

Project Acronym: " ARDUinVET "

Project No: "2020-1-TR01-KA202-093762"

Arduino Input/Output Module and Kit Module



Planning Our Projects

When starting our first projects, you might be tempted to write your sketch immediately after you've come up with a new idea. But before you start writing, a few basic preparatory steps are in order. After all, your Arduino board isn't a mind-reader; it needs precise instructions, and even if these instructions can be executed by the Arduino, the results may not be what you expected if you overlooked even a minor detail.

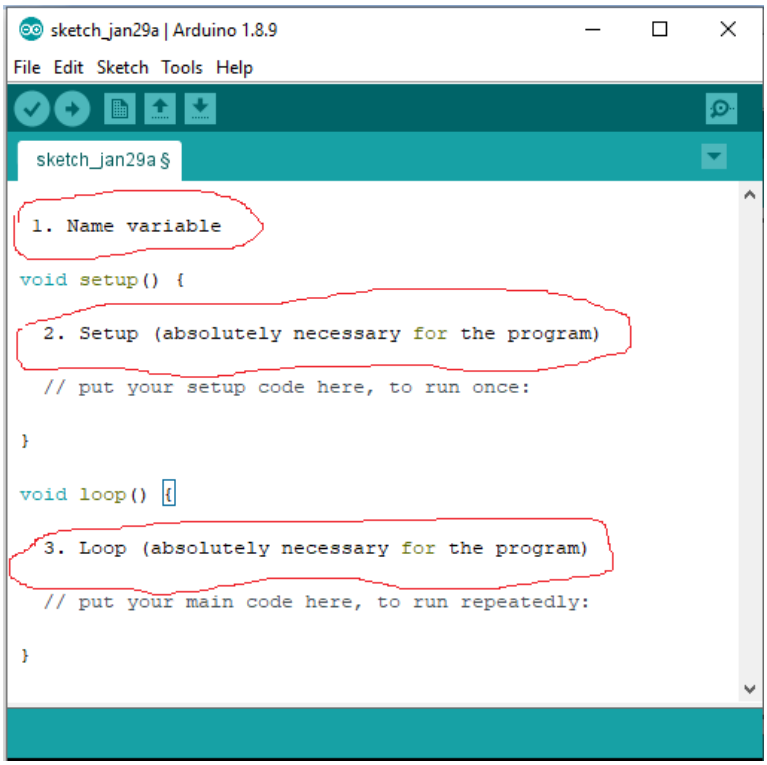
Whether you are creating a project that simply blinks a light or an automated model railway signal, a detailed plan is the foundation of success. When designing your Arduino projects, follow these basic steps:

1. Define your objective. Determine what you want to achieve.
2. Write your algorithm. An algorithm is a set of instructions that describes how to accomplish your project. Your algorithm will list the steps necessary for you to achieve your project's objective.
3. Select your hardware. Determine how it will connect to the Arduino.
4. Write your sketch. Create your initial program that tells the Arduino what to do.
5. Wire it up. Connect your hardware, circuitry, and other items to the Arduino board.
6. Test and debug. Does it work? During this stage, you identify errors and find their causes, whether in the sketch, hardware, or algorithm.

The more time you spend planning your project, the easier time you'll have during the testing and debugging stage.

Basic structure of a sketch

The Arduino program is called as "sketch". A sketch can be divided in three parts.



1. Name variable:

In the first part elements of the program are named. This part is not absolutely necessary.

2. Setup (absolutely necessary for the program):
The setup will be performed only once. Here you are telling the program for example what Pin (slot for cables) should be an input and what should be an output on the boards.
Defined as Output: The pin should put out a voltage. For example: With this pin a LED is meant to light up.
Defined as an Input: The board should read out a voltage. For example: A switch is actuated. The board recognized this, because it gets a voltage on the Input pin.

3. Loop (absolutely necessary for the program):
This loop part will be continuously repeated by the board. It assimilates the sketch from beginning to end and starts again from the beginning and so on.

Further Syntax Rules

It is necessary to pay attention to these rules while writing Arduino programs. Otherwise, our program will fail.

; (Semicolon):

; (Semicolon) is used to end a statement. Forgetting to end a line in a semicolon will result in a compiler error.

Example: int a=13;

{} (Curly Braces):

Curly braces are a major part of the Arduino programming language. They are used in several different constructs, and this can sometimes be confusing for beginners. An opening curly brace "{" must always be followed by a closing curly brace "}".

The main uses of curly braces: Functions, Loops, Conditional statements

Example:

```
void myfunction(datatype argument)
{       statements(s)                }
```

// (Single line comment) and /* */ (Multi-line comment):

These are lines in the program that are used to inform yourself or others about the way the program works. They are ignored by the compiler, and not exported to the processor, so they don't take up any space on the Atmega chip.
Comments only purpose are to help you understand (or remember) how your program works or

to inform others how your program works. There are two different ways of marking a line as a comment:

Example:

```
x = 5;      // This is a single line comment. Anything after the slashes is a comment
//  to the end of the line

x = 5;    /*This is a multi-line comment.    ……
This is the end of  a multi-line comment.    */
```

#define:

#define allows the programmer to give a name to a constant value before the program is compiled. Defined constants in arduino don't take up any program memory space on the chip. In general, the const keyword is preferred for defining constants and should be used instead of #define.

Example:

```
#define ledPin 3     // The compiler will replace any mention of ledPin with the value 3 at compile time.
```

#include:

#include is used to include outside libraries in your sketch. This gives the programmer access to a large group of standard C libraries (groups of pre-made functions), and also libraries written especially for Arduino.

Example:

```
#include <servo.h>
```

Arduino - Data Types

Data types are used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in the storage and how the bit pattern stored is interpreted.

Expressions that are used to store any information in memory and can change the value during program flow are called variables. Variables can be numbers, characters, or logical expressions. The appropriate data type should be selected according to the variable type. A certain area is allocated according to the type of data, used in defining the variable in memory.

The following table provides all the data types that you will use during Arduino programming.

Type	Contains
boolean	can contain either true or false
char	-128 to 127
byte	0 to 255
unsigned char	0 to 255
int	-32,768 to 32,767
unsigned int	0 to 65,535
word	(same as unsigned int)
long (or long int)	-2,147,483,648 to 2,147,483,647
unsigned long	0 to 4,294,967,295
float	-3.4028235E+38 to 3.4028235E+38
double	(same as float)

Arduino - Variables & Constants

While defining the variable, the name of the variable, the value of the variable and the appropriate data type for the variable must be determined.

The definition is made as seen in the example below:

int LED =12;

Here: **int**=data type **LED**=variable name **12**=variable value

Variables, which Arduino uses, have a property called scope. A scope is a region of the program and there are three places where variables can be declared. They are:

- Inside a function or a block, which is called local variables.
- In the definition of function parameters, which is called formal parameters.
- Outside of all functions, which is called global variables.

A constant is a variable *qualifier* that modifies the behavior of the variable, making a variable "*read-only*". This means that the variable can be used just as any other variable of its type, but its value cannot be changed. You will get a compiler error if you try to assign a value to a const variable.

Constants defined with the const keyword obey the rules of variable scoping that govern other variables. This, and the pitfalls of using #define, makes the const keyword a superior method for defining constants and is preferred over using #define.

Example:

```
const float pi = 3.14;
```

Notes:

#define **or** const: You can use either const or #define for creating numeric or string constants. For arrays, you will need to use const. In general const is preferred over #define for defining constants.

Arduino – Operators

An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. In Arduinos, the following types of operators may be used.

Arithmetic Operators:

Assume variable A holds 10 and variable B holds 20 then −

Operator name	Operator simple	Example
assignment operator	=	A = B
addition	+	A + B will give 30
subtraction	-	A - B will give -10
multiplication	*	A * B will give 200
division	/	B / A will give 2
modulo	%	B % A will give 0

Comparison Operators:

Assume variable A holds 10 and variable B holds 20 then −

Operator name	Operator symble	Example
equal to	==	(A == B) is not true
not equal to	!=	(A != B) is true
less than	<	(A < B) is true
greater than	>	(A > B) is not true
less than or equal to	<=	(A <= B) is true
greater than or equal to	>=	(A >= B) is not true

Boolean Operators

Assume variable A holds 10 and variable B holds 20 then −

Operator name	Operator simple	Example
and	&&	(A && B) is true
or	||	(A || B) is true
not	!	!(A && B) is false

Bitwise Operators

Assume variable A holds 60 and variable B holds 13 then −

Operator name	Operator simple	Example
and	&	(A & B) will give 12 which is 0000 1100
or	|	(A | B) will give 61 which is 0011 1101

xor	^	(A ^ B) will give 49 which is 0011 0001
not	~	(~A) will give -60 which is 1100 0011
shift left	<<	A << 2 will give 240 which is 1111 0000
shift right	>>	A >> 2 will give 15 which is 0000 1111

Arduino - I/O Functions (Commands)

Functions allow structuring the programs to perform individual tasks. The typical case for creating a function is when one needs to perform the same action multiple times in a program. They will become clearer when we show actual program examples in circuits and Arduino programmes below. To help explain the various command functions, we've broken them down into separate Commands

The pins on the Arduino board can be configured as either inputs or outputs. We will explain the functioning of the pins in those modes. It is important to note that a majority of Arduino analog pins, may be configured, and used, in exactly the same manner as digital pins.

pinMode() Function:

The pinMode() function is used to configure a specific pin to behave either as an input or an output. It is possible to enable the internal pull-up resistors with the mode INPUT_PULLUP.

Syntax: pinMode(pin, mode)

```
Void setup () {
 pinMode (pin , mode);
                }
```

Parameters:
pin: the Arduino pin: number to set the mode of.
mode: INPUT, OUTPUT, or INPUT_PULLUP.

Examples:

```
pinMode(13, OUTPUT);      // sets the digital pin 13 as output
pinMode(5, INPUT);        // sets the digital pin 5 as input
```

digitalWrite() Function:

The digitalWrite() function is used to write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim. Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor, which acts like a large current-limiting resistor.

Syntax:

```
digitalWrite (pin ,value);
```

pin: the number of the pin whose mode you wish to set

value:HIGH (1), or LOW(0).

Example:

```
digitalWrite(LED, HIGH);          // turn on led
digitalWrite(LED, LOW);           // turn off led
```

digitalRead()Function:

The digitalRead() function reads the value from a specified digital pin, either HIGH or LOW.

Syntax: digitalRead(pin)
pin: the Arduino pin number you want to read.

Examples:

```
val = digitalRead(inPin);   // read the input pin
```

Note: The analog input pins can be used as digital pins, referred to as A0, A1, etc.

delay() Function:
The dalay() function pauses the program for the amount of time (in milliseconds) specified as parameter. (There are 1000 milliseconds in a second.)

Syntax: delay(ms):

ms: the number of milliseconds to pause. Allowed data types: unsigned long.

Examples:

```
delay(1000);                    // waits for 1 second
delay(2000);                    // waits for 2 seconds
```

analogWrite() Function:

The analogWrite() function writes an analog value (PWM wave) to a pin. It can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady rectangular wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite()) on the same pin.

Examples:

Sets the output to the LED proportional to the value read from the potentiometer.

```
val = analogRead(analogPin);   // read the input pin
analogWrite(ledPin, val / 4);   // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
```

analogRead( ) Function

Arduino is able to detect whether there is a voltage applied to one of its pins and report it through the digitalRead() function. There is a difference between an on/off sensor (which detects the presence of an object) and an analog sensor, whose value continuously changes. In order to read this type of sensor, we need a different type of pin.

In the lower-right part of the Arduino board, you will see six pins marked “Analog In”. These special pins not only tell whether there is a voltage applied to them, but also its value. By using the analogRead() function, we can read the voltage applied to one of the pins.

This function returns a number between 0 and 1023, which represents voltages between 0 and 5 volts. For example, if there is a voltage of 2.5 V applied to pin number 0, analogRead(0) returns 512.

Syntax: analogRead(pin);

pin: the number of the analog input pin to read from 0 to 5.

Example:

```
val = analogRead(analogPin);        // read the input pin
Serial.println(val);                // debug value
```

if Function:

The if statement checks for a condition and executes the following statement or set of statements if the condition is 'true'.

Syntax:
if (condition) {
//statement(s)
}

condition: a boolean expression (i.e., can be true or false).

Examples: (The brackets may be omitted after an if statement. If this is done, the next line (defined by the semicolon) becomes the only conditional statement.

```
if (x > 120) digitalWrite(LEDpin, HIGH);

if (x > 120)
digitalWrite(LEDpin, HIGH);

if (x > 120) {digitalWrite(LEDpin, HIGH);}

if (x > 120) {
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}
            // all are correct
```

if-else Command:

The if…else allows greater control over the flow of code than the basic if statement, by allowing multiple tests to be grouped. An else clause (if at all exists) will be executed if the condition in the if statement results in false. The else can proceed another if test, so that multiple, mutually exclusive tests can be run at the same time.

Each test will proceed to the next one until a true test is encountered. When a true test is found, its associated block of code is run, and the program then skips to the line following the entire if/else construction. If no test proves to be true, the default else block is executed, if one is present, and sets the default behavior. An unlimited number of such else if branches are allowed.

Syntax:

```
if (condition is TRUE) {
  // do Thing A
}
else
  //if NOT,  do Thing B
}
```

```
if (condition1) {
  // do Thing A
}
else if (condition2) {
  // do Thing B
}
else {
  // do Thing C
}
```

Examples: (Below is an extract from a code for temperature sensor system.)

```
if (temperature >= 70) {
  // Danger! Shut down the system.
}
else if (temperature >= 60) { // 60 <= temperature < 70
  // Warning! User attention required.
}
else { // temperature < 60
  // Safe! Continue usual tasks.
}
```

for Command:

The for statement is used to repeat a block of statements enclosed in curly braces. An increment counter is usually used to increment and terminate the loop. The for statement is useful for any repetitive operation, and is often used in combination with arrays to operate on collections of data/pins.

Syntax:

```
for (initialization; condition; increment) {
  // statement(s);
}
```

Parameters:

initialization: happens first and exactly once.
condition: each time through the loop, condition is tested; if it's true, the statement block, and the increment is executed, then the condition is tested again. When the condition becomes false, the loop ends.
increment: executed each time through the loop when condition is true.

Examples:

```
for (int i = 0; i <= 255; i++)            {
    analogWrite(PWMpin, i);               }

for (int x = 2; x < 100; x = x * 1.5) {
        println(x);                       }

for (int i = 0; i > -1; i = i + x)   {
  analogWrite(PWMpin, i);        }
```

switch...case Command

Like **if** statements, switch/ case controls the flow of programs by allowing programmers to specify different code that should be executed in various conditions. In particular, a switch statement compares the value of a variable to the values specified in case statements. When a case statement is found whose value matches that of the variable, the code in that case statement is run.

The **break** keyword exits the switch statement, and is typically used at the end of each case. Without a break statement, the switch statement will continue executing the following expressions ("falling-through") until a break, or the end of the switch statement is reached.

Syntax:

```
switch (var) {
  case label1:
    // statements
    break;
  case label2:
    // statements
    break;
  default:
    // statements
    break;
}
```

Example Code:

```
switch (var) {
  case 1:
    //do something when var equals 1
    break;
  case 2:
    //do something when var equals 2
    break;
  default:
    // if nothing else matches, do the default
    // default is optional
    break;
}
```

var: a variable whose value to compare with various cases. Allowed data types: int, char.

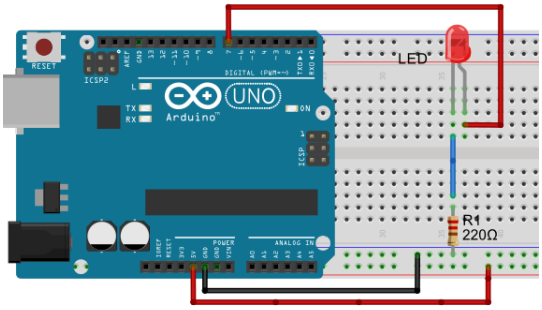
label1, label2: constants. Allowed data types: int, char.

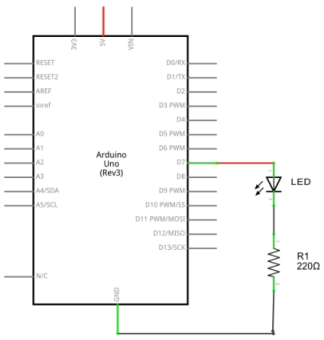
Note:

Arduino circuits are shown in two ways;

1-) Breadboard view

2-) Schematic view





1-) Breadboard view

2-) Schematic view

We will use Breadboard view which is used more.

Enough talking! Let's make something!

Circuits of Arduino Button and LED Module Kit

Most pins on the Arduino can be configured as input or output. The Button and LED Module Kit is first training KITof the ARDUinVET to make students learn I/O systems of Arduinos. So, it can be named as I/O Arduino training KIT. In this Training Kit, as shown as below, 6 buttons are connected to Arduino as inputs. And a buzzer, a 2-pin connector and 6 LEDs are connected as outputs.

As students can use this training kit to learn I/O systems of Arduinos, this training kit can maket them test Arduino circuit more easy. Also, they can use a breadboard to test Arduino circuits and experiments.

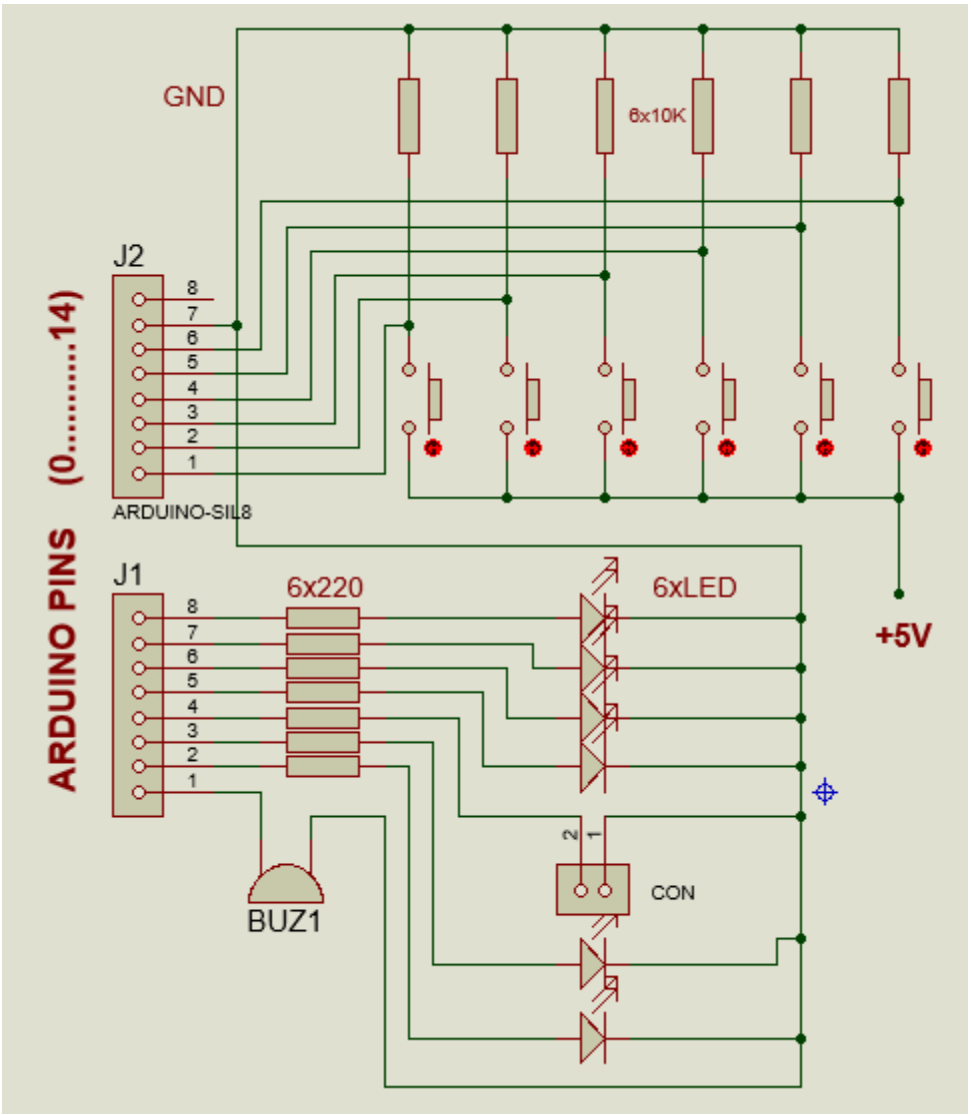


Figure: Open Circuit of Button and LED Module Kit

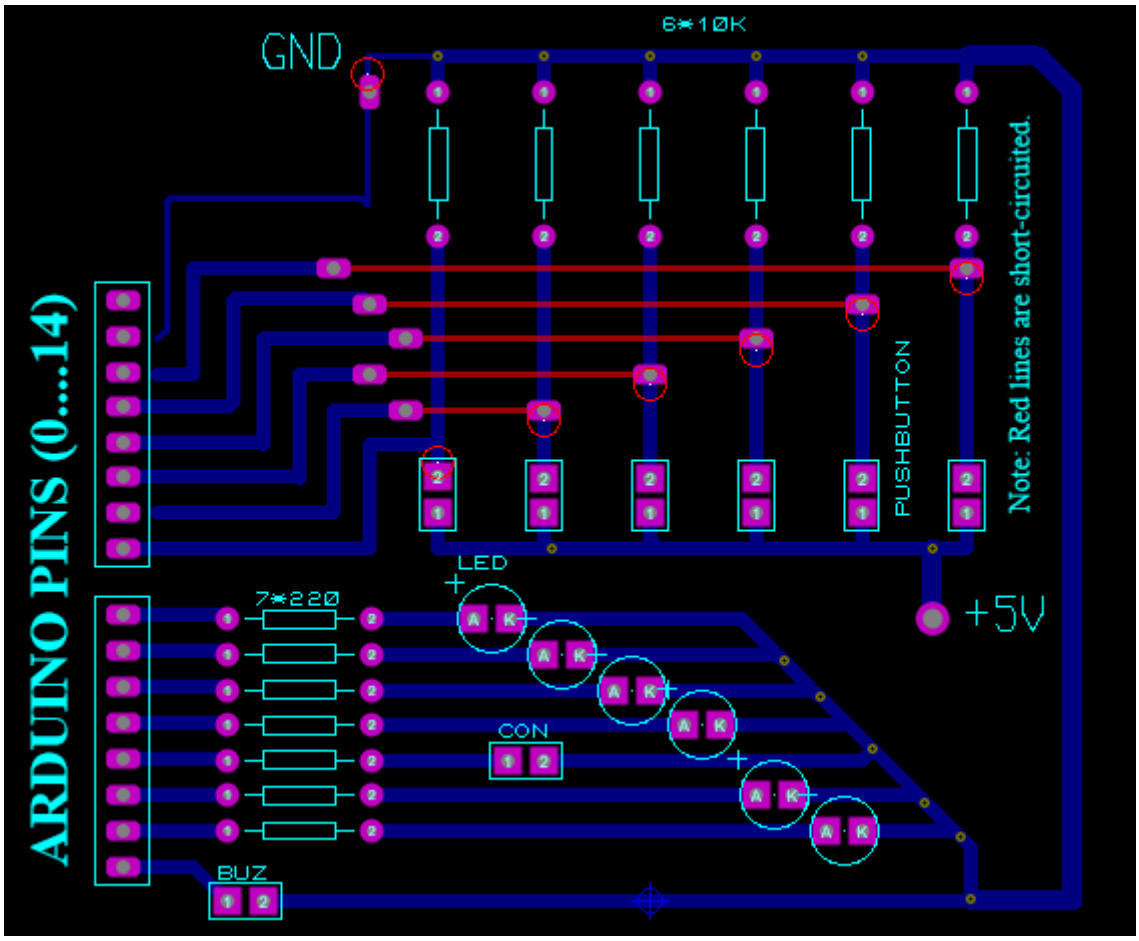


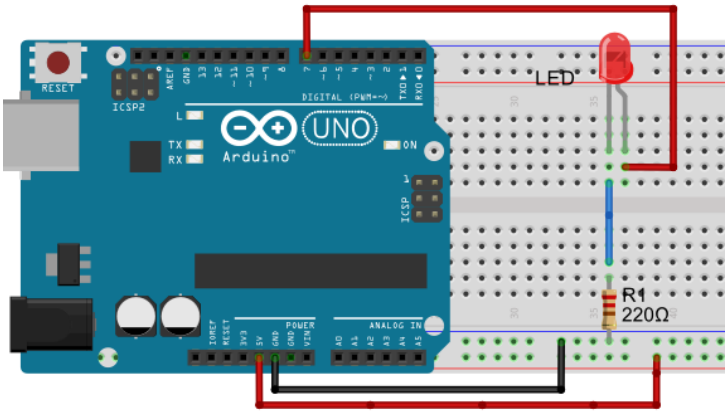
Figure: PCB Schema of Button and LED Module Kit

Sample Arduio Circuits and Programs

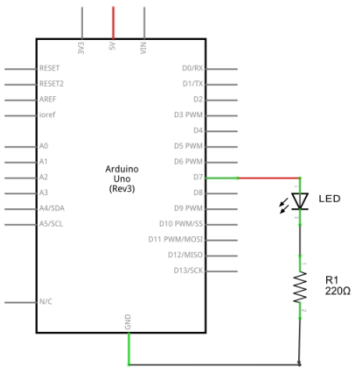
Circuit 1:

Circuit title: LED flashing Program

Circuit description: An LED is connected to 13th pin of the Arduino. The LED is flashed continuously with 1second interval.



1-) Breadboard view

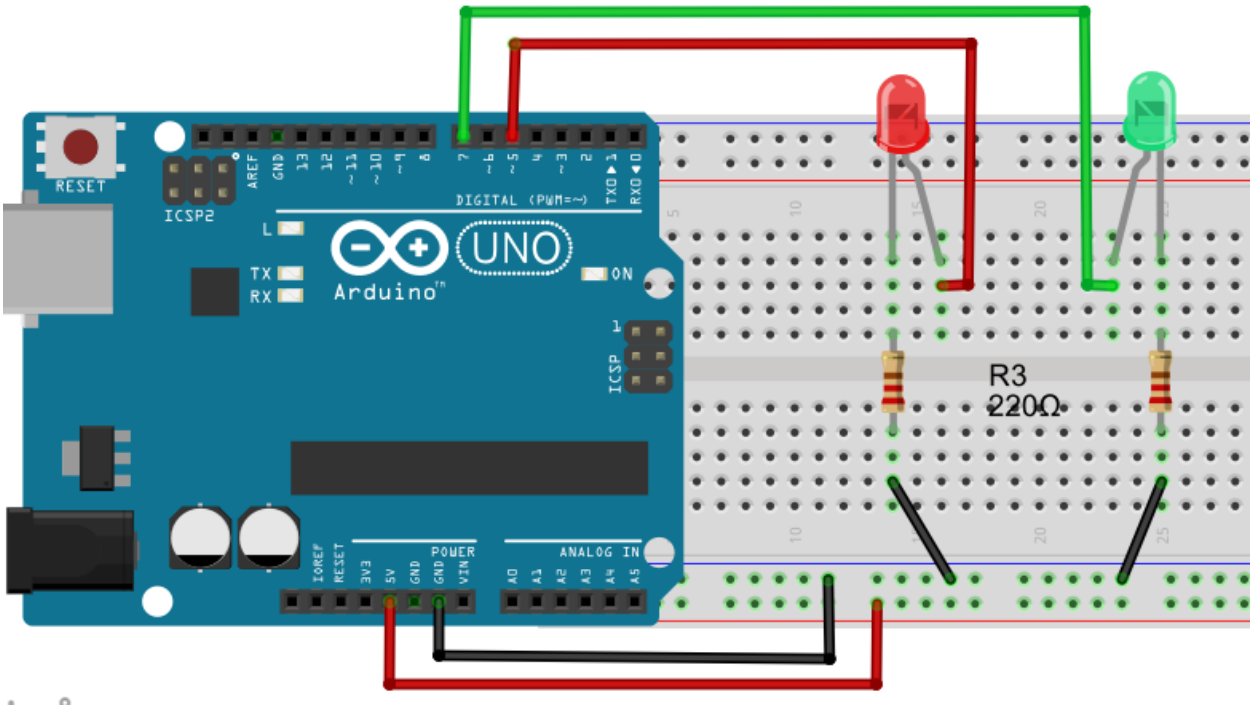


2-) Schematic view

```
/*    LED flashing Program, Switching a LED on and off    */

int led = 7;                    // integer variable led is declared

void setup() {                  // the setup() method is executed only once
pinMode(led, OUTPUT);           // the led PIN is declared as digital output
}

void loop() {                   // the loop() method is repeated
digitalWrite(led, HIGH);        // switching on the led
delay(1000);                    // stopping the program for 1000 milliseconds
digitalWrite(led, LOW);         // switching off the led
delay(1000);                    // stopping the program for 1000 milliseconds
 }
```

Circuit 2:

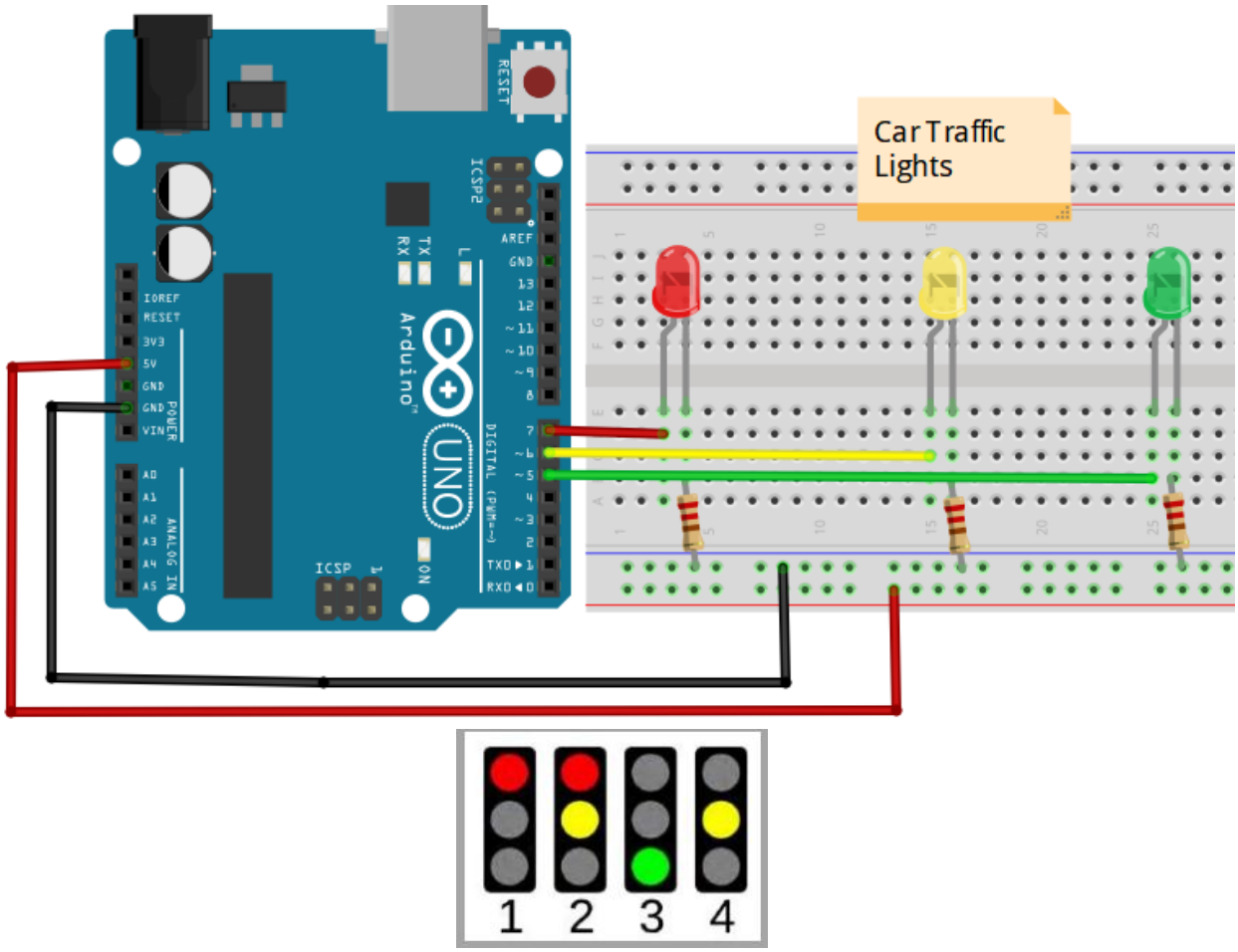
Circuit title: Flip-Flop , 2 LEDs flashing Program
Circuit description: 2 LEDs blink sequentially with 2 second intervals.




```
/*    Flip Flop     */

int greenLED=5;                    // Pin where the green LED is attached
int redLED=7;                      // Pin where the red LED is attached

void setup()  {
pinMode(greenLED, OUTPUT);         // green LED pin is initialised as OUTPUT
pinMode(redLED, OUTPUT);           // red LED pin is initialised as OUTPUT
                  }

void loop(){
digitalWrite(greenLED,  HIGH);          // switch on green LED
digitalWrite(redLED,  LOW);             // switch off red LED
pause(2000);

digitalWrite(greenLED,  LOW);           // switch off green LED
digitalWrite(redLED,  HIGH);            // switch on red LED
pause(2000);
                  }
```

Circuit 3:

Circuit title: Traffic Lights
Circuit description: In this Project, We are going to build a traffic lights system. There are 3 LEDs with different colors (green, yellow and red).



/* Traffic Lights */

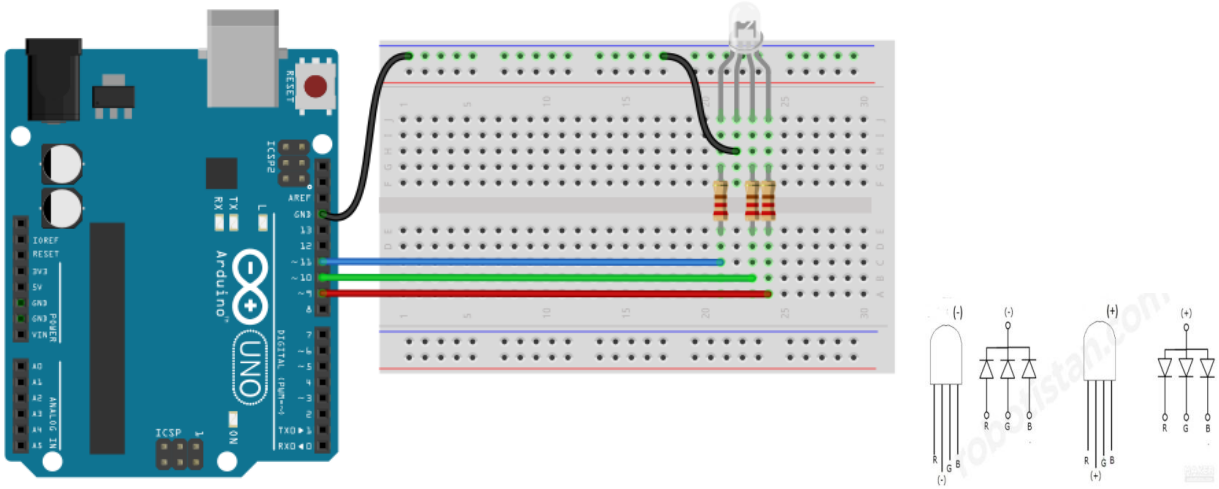
```
int redLED = 7;
int yellowLED =6;
int greenLED = 5;

void setup() {                          //  here, we are initializing our pins as outputs
pinMode(redLED,  OUTPUT);
pinMode(yellowLED,  OUTPUT);
pinMode(greenLED,  OUTPUT);
}

void loop() {

digitalWrite(redLED,  HIGH);            // redLED is ON for 9 seconds
```

```
digitalWrite(yellowLED,  LOW);
digitalWrite(greenLED,  LOW);
delay(9000);

digitalWrite(redLED,  HIGH);          // redLED and  yellowLED are  ON for 2seconds
digitalWrite(yellowLED,  HIGH);
digitalWrite(greenLED,  LOW);
delay(2000);

digitalWrite(redLED,  LOW);           // greenLED is ON for 9 seconds
digitalWrite(yellowLED,  LOW);
digitalWrite(greenLED,  HIGH);
delay(9000);

digitalWrite(redLED,  LOW);           // Again, yellowLED is ON for 2seconds
digitalWrite(yellowLED,  HIGH);
digitalWrite(greenLED,  LOW);
delay(2000);

/* The loop starts again    */
 }
```

Circuit 4:

Circuit title: RGB LED APPLICATION

Circuit Explanation: The RGB LED is the 3 LEDs, connected in common, placed in a single case. It is possible to control the light intensity of three colors digitally. In addition, desired colors can be obtained by using the PWM technique.



```
/*  We will flash each color of RGB LED in 1 second intervals. If we want to display white light,
we need to turn on all the LEDs..*/
```

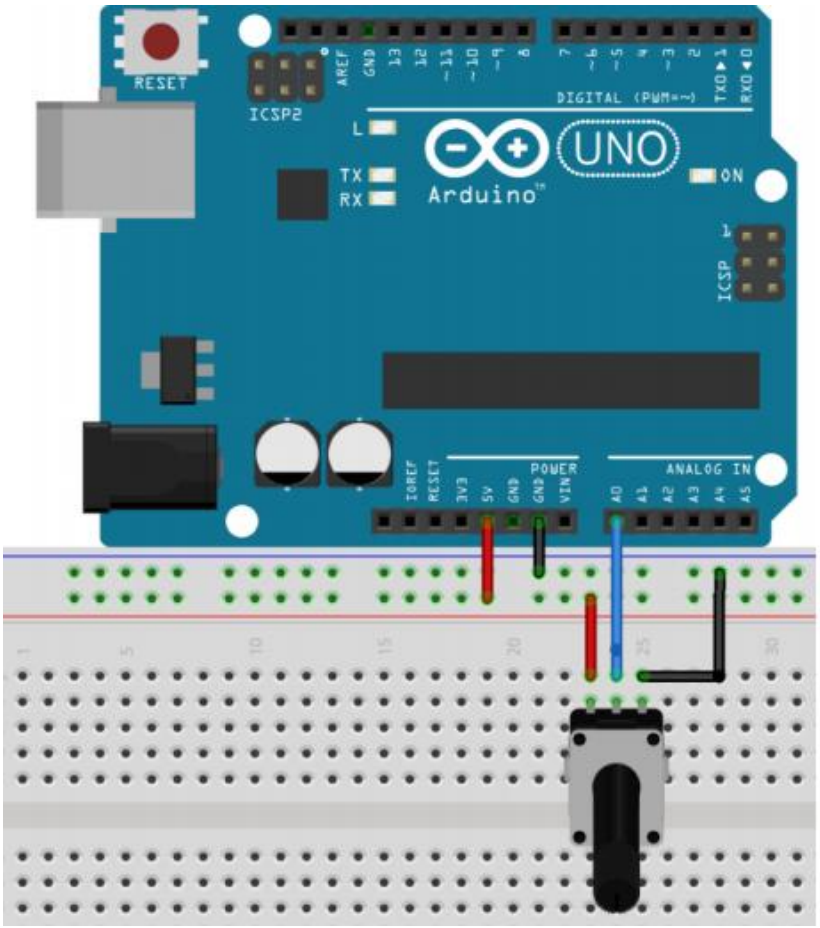
```
const int BlueLed=11;          // we connect the blue led to pin-11
const int GreenLed=10;         //  we connect the green led to pin-10
const int RedLed=9;            // we connect the red led to pin-11

        // We assign the pins to which the LEDs are connected as outputs.
void setup()  {
pinMode(BlueLed,OUTPUT);
pinMode(GreenLed,OUTPUT);
pinMode(RedLed,OUTPUT);      }

        //  The loop  starts here.

void loop() {
digitalWrite(BlueLed, LOW);         // RedLed is ON.
digitalWrite(GreenLed, LOW);
digitalWrite(RedLed, HIGH);
delay(1000);

 digitalWrite(BlueLed,  LOW );     // GreenLed is ON.
digitalWrite(GreenLed, HIGH);
digitalWrite(RedLed, LOW );
delay(1000);

 digitalWrite(BlueLed,  HIGH);          // BlueLed is ON.
digitalWrite(GreenLed,  LOW);
digitalWrite(RedLed, LOW);
delay(1000);

        // We display the white color by activating all the leds.
digitalWrite(BlueLed,  HIGH);
digitalWrite(GreenLed,  HIGH);
digitalWrite(RedLed,  HIGH);
delay(1000);
}
```

Circuit 5:

Circuit title: Reading Analog Voltage, Reading the Value From Potentiometer

Circuit Explanation: Here, we learn how to read an analog input on analog pin-0. The input is converted from analogRead() into voltage, and printed out to the serial monitor of the Arduino IDE.

Potentiometer: A potentiometer (or pot) is a simple electro-mechanical transducer. It converts rotary or linear motion from the input operator into a change of resistance. This change is (or can be) used to control anything from the volume of a hi-fi system to the direction of a huge container ship.	



/* ReadAnalogVoltage : Reads an analog input on pin 0, converts it to voltage, and prints the result to the serial monitor.
Graphical representation is available using serial plotter (Tools > Serial Plotter menu).
Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground. */

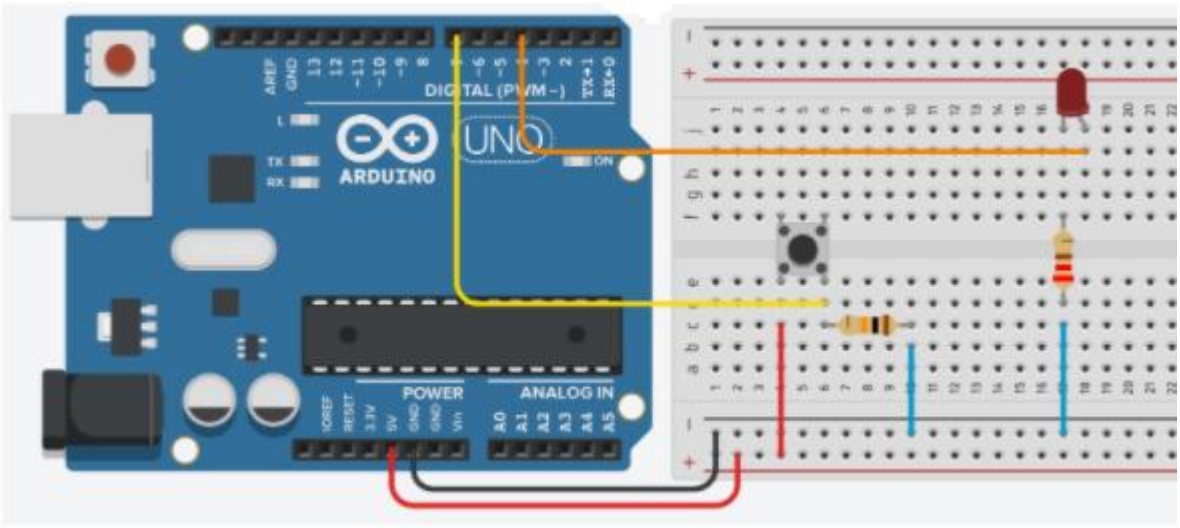
// the setup routine runs once when you press reset:

```
void setup()        {
          // initialize serial communication at 9600 bits per second:
  Serial.begin(9600); }
         // the loop routine runs over and over again forever:
void loop() {
          // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
          // print out the value you read:
  Serial.println(sensorValue);
  delay(1000);        // delay between reads for stability
}
```

Circuit 6:

Circuit title: Using buttons on Arduinos

Circuit Explanation: When pressing a pushbutton attached to pin 7, the button turns on and off a (LED), connected to digital pin 4,




```
/*     Button Turns on and off a light emitting diode(LED) connected to digital  pin 4, when
pressing a pushbutton attached to pin 7.       */

void setup()
{
  pinMode(4, OUTPUT);  // pin-4 is output
  pinMode(7, INPUT);     //  pin7 is input. 7
}
void loop()
{
   if (digitalRead(7) == HIGH)                 //  If pin7 is high,
    digitalWrite(4, HIGH);                     // Led is ON,
    if (digitalRead(7) == LOW)                 // If pin7 is low,
     digitalWrite(4, LOW);                     // Led is OFF.
}
```

NOTE: IF-THEN command also can be used for connecting buttons to Inputs pins of Arduinos. These connection can be done in two differents ways. Pull_Up connection and Pull-down connection.

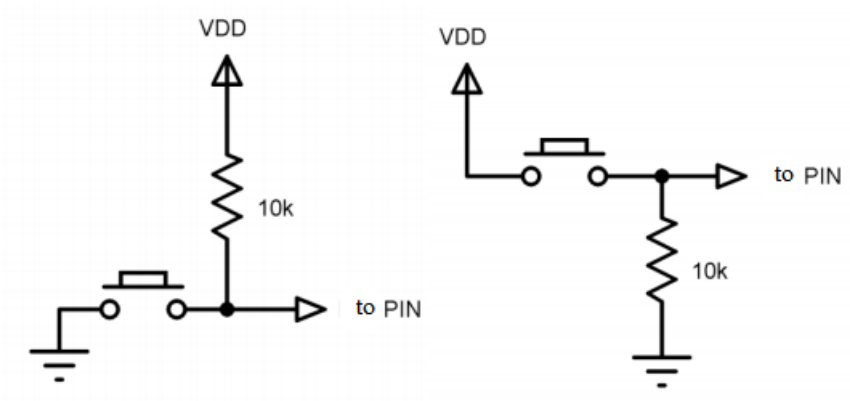
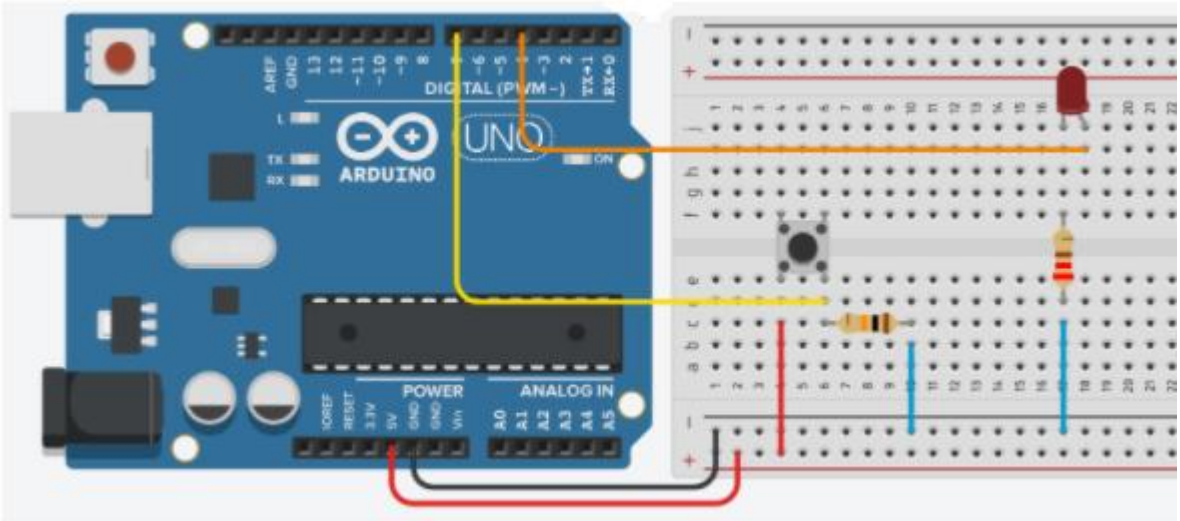


Figure: Switches or Butons that can be used for the IF…THEN command

Circuit 7:

Circuit title: Using ELSE command on Arduinos

Circuit Explanation: When pressing a pushbutton attached to pin 7, the button turns on and off a (LED), connected to digital pin 4. Also, We will learn to determine the pins with the "int" variable.



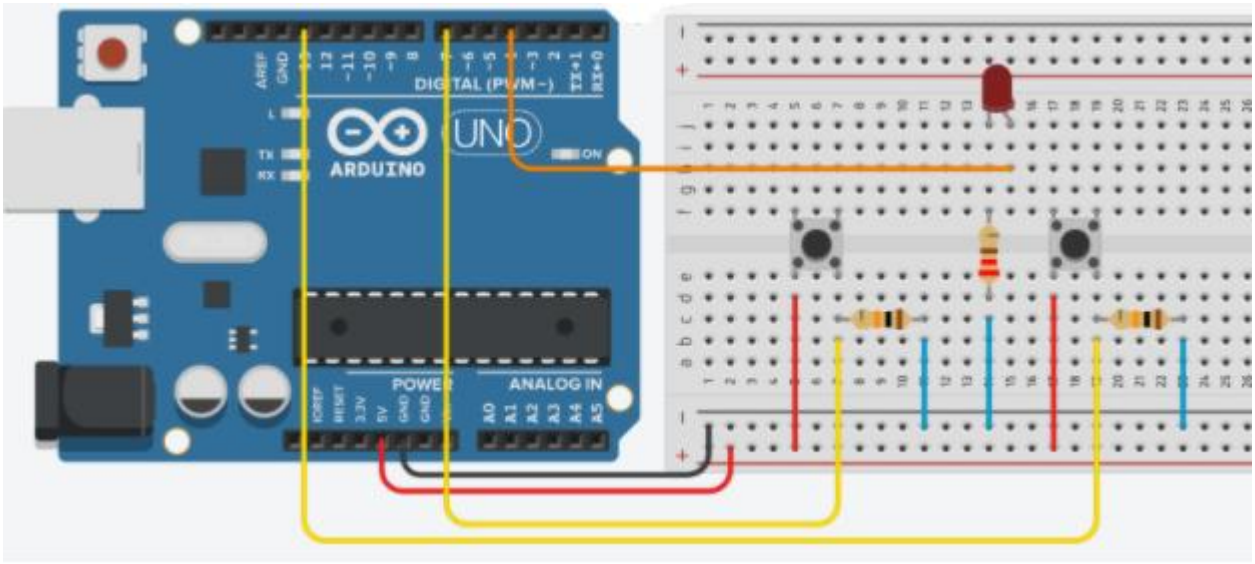

```
int led = 4;
int buton =7;
void setup()
```

```
{
  pinMode(led, OUTPUT);
  pinMode(buton, INPUT);
}
void loop()
{
    if (digitalRead(buton) == HIGH)          // The button is ON,
       digitalWrite(led, HIGH);              // Led is ON
     else                                    // If  not,
       digitalWrite(led, LOW);               // Led is OFF.
}
```

Circuit 8:

Circuit title: 2 Buttons versus 1 LED

Circuit Explanation: One buton turns on the Led, another buton turn off the Led.




```
/*      2 buttons versus 1 LED
Buttons are connected  with 10K resistors in series.      */

int led = 4;                    //  The  pin-4  is assigned  as "led"
int button1 = 7;                // The  pin-7  is assigned  as   "button1"
int button2 = 13;               // The  pin-13  is assigned  as   "button2"
void setup()
{
  pinMode(led, OUTPUT);                 // led=Output
  pinMode(button1, INPUT);               //  button1=Input
  pinMode(button2, INPUT);               // button2=Input
}
void loop()
{
  if (digitalRead(button1) == HIGH)         // IF  "button1" is ON (active),
     digitalWrite(led, HIGH);                // Led is ON.
    if (digitalRead(button2) == HIGH)       // IF  "button2"  is ON (active),
     digitalWrite(led, LOW);                // Led is OFF.
}
```

HOW-TO Use the ARDUINO SERIAL MONITOR

The Arduino IDE has a feature that can be a great help in debugging sketches or controlling Arduino from your computer's keyboard.

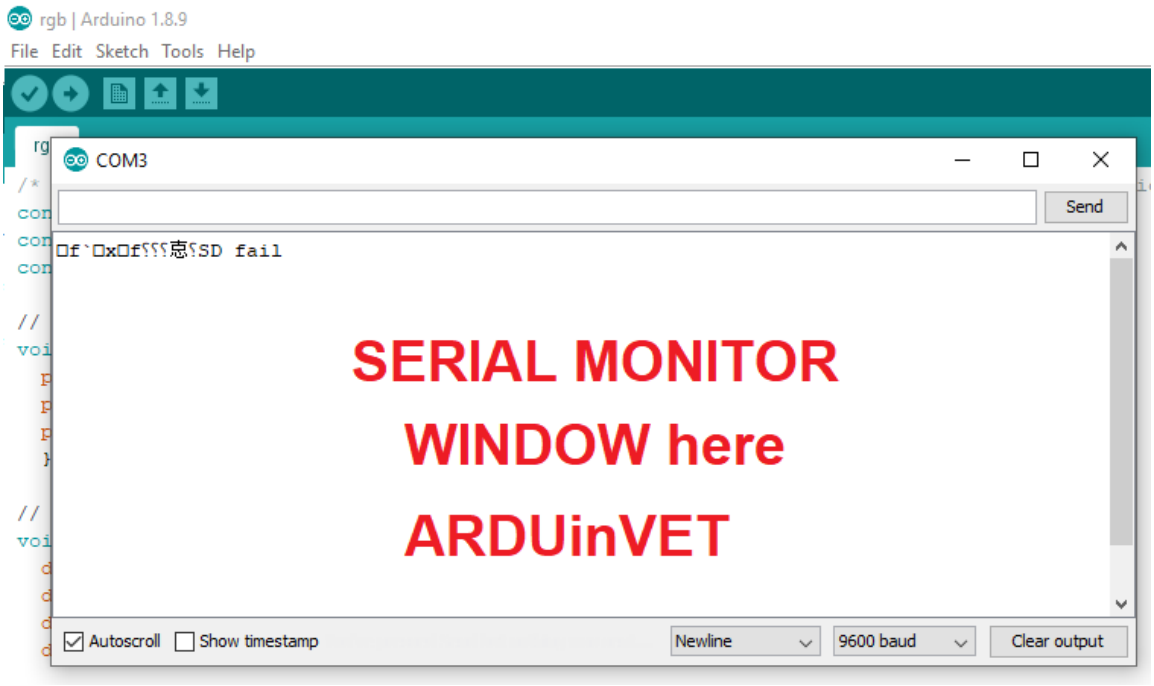
The Serial Monitor is a separate pop-up window that acts as a separate terminal that communicates by receiving and sending Serial Data. See the icon on the far right of the image here 

.

Serial Data is sent over a single wire (but usually travels over USB in our case) and consists of a series of 1's and 0's sent over the wire. Data can be sent in both directions (In our case on two wires).

You will use the Serial Monitor to debug Arduino Software Sketches or to view data sent by a working Sketch. You must have an Arduino connected by USB to your computer to be able to activate the Serial Monitor.

Open the Serial Monitor by clicking on the Serial Monitor box in the IDE. It should look like the screenshot below. Make SURE the baud (speed) is set to 9600. It is located in the bottom right corner. (The important thing is that it is set the same in our program and here. Since the default here is 9600, we set our



Main Commands to Use Serial Monitor

Serial.begin(); Sets the data rate in bits per second (baud) for serial data transmission. For communicating with Serial Monitor, make sure to use one of the baud rates listed in the menu at the bottom right corner of its screen. You can, however, specify other rates - for example, to communicate over pins 0 and 1 with a component that requires a particular baud rate.

Syntax: Serial.begin(speed);
Example: Serial.begin(9600);

Serial.print(); Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. For example:

Serial.print(78;) gives "78"

Serial.print('N'); gives "N"

Serial.print("Hello ArduinVet."); gives "Hello ArduinVet."

Serial.println(); Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). This command takes the same forms as Serial.print().

Serial.println(val);
Serial.println(val, format);
Serial.printle(13, BIN); gives "1101", binary value of 15 on erial Monitor.

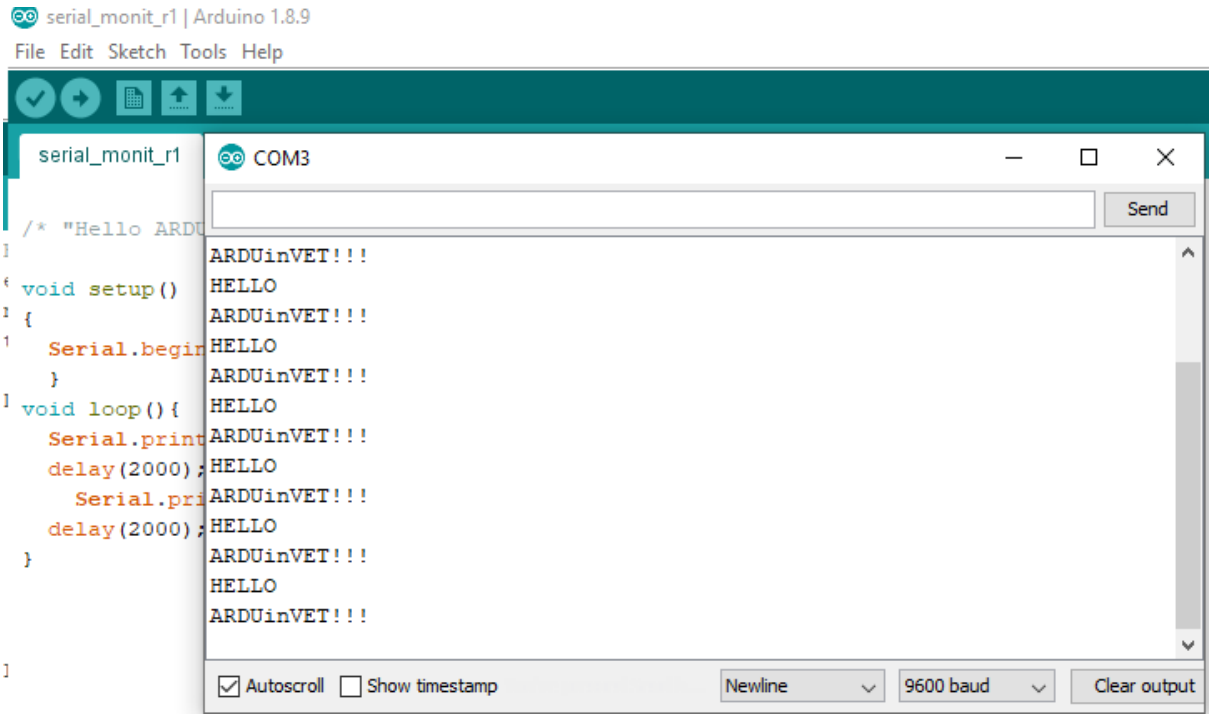
NOTE: The only difference between Serial.print and Serial.println is that Serial.println means that the next thing sent out the serial port after this one will start on the next line. There is a third new thing you may have noticed. There is something in quotes (“). This is called a string.

Circuit 9:

Circuit title: "Hello ARDUinVET" Application in Serial Monitor"

Circuit Explanation: Hello ARDUinVET will be seen on the serial Monitor.

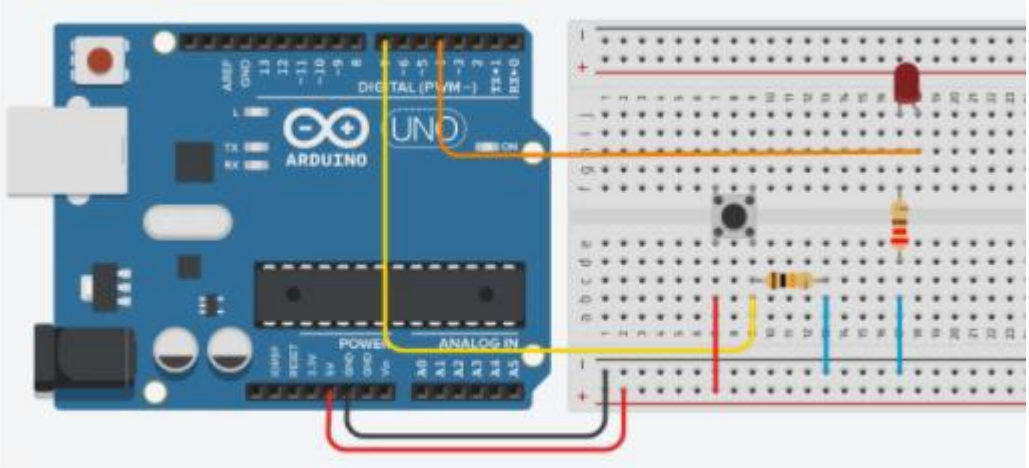
```
/*       "Hello ARDUinVET" Application in Serial Monitor  */
void setup()
{
  Serial.begin(9600);     // Serial communication speed
}
void loop()
{
  Serial.println("  HELLO ");
  delay(2000);
    Serial.println("ARDUinVET!!!");
  delay(2000);
}
```



Circuit 10:

Circuit title: " Button Status Analysis on Serial Monitor"

Circuit Explanation: If the buton is pressed, "Led is ligthing, LED=ON" message is on the serial monitor and the Led is lighting. If If the buton is not pressed, ("Button is not pressed") message is on the serial monitor and the the Led is lighting.




```
/*      Button Status Analysis on Serial Monitor   */

void setup()                  {
 pinMode(4, OUTPUT);
 pinMode(7, INPUT);
   Serial.begin(9600);        }
void loop()                    {
 if (digitalRead(7) == HIGH)        // Read the digital signal on Pin-7
   {
   digitalWrite(4, HIGH);
   Serial.println("Led is ligthing, LED=ON");
   delay(250);
   }
  else                        // If the previous condition is not  met.
   {
   digitalWrite(4, LOW);
   Serial.println("Button is not pressed");
```

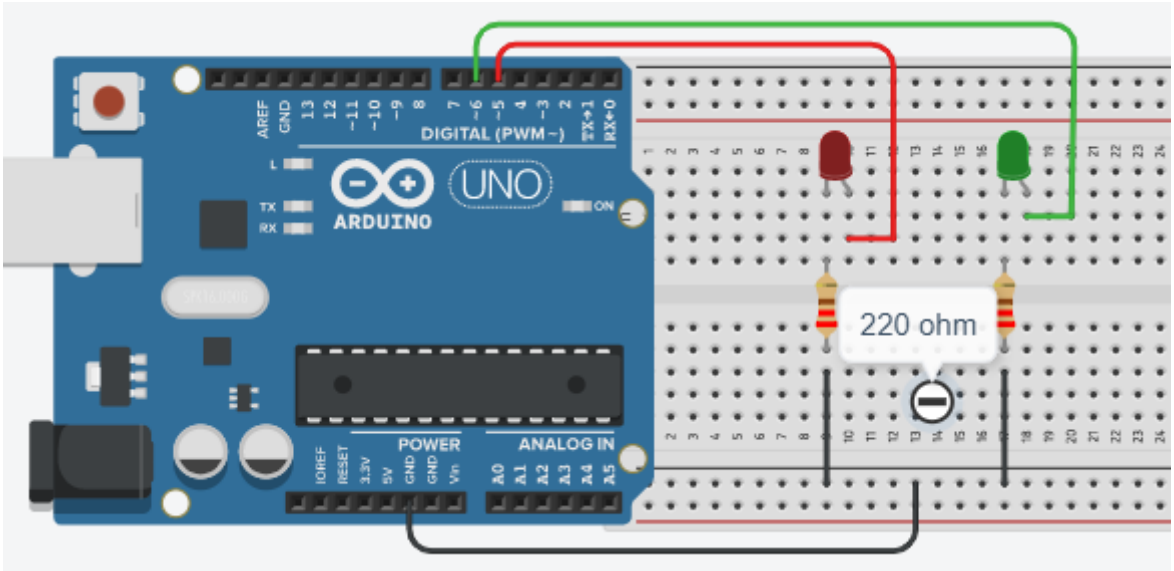
```
  delay(1000);
  }
}                          //The view of the serial monitor is seen below.
```

```
Button is not pressed
Button is not pressed
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Led is ligthing, LED=ON
Button is not pressed
```

Circuit 11:

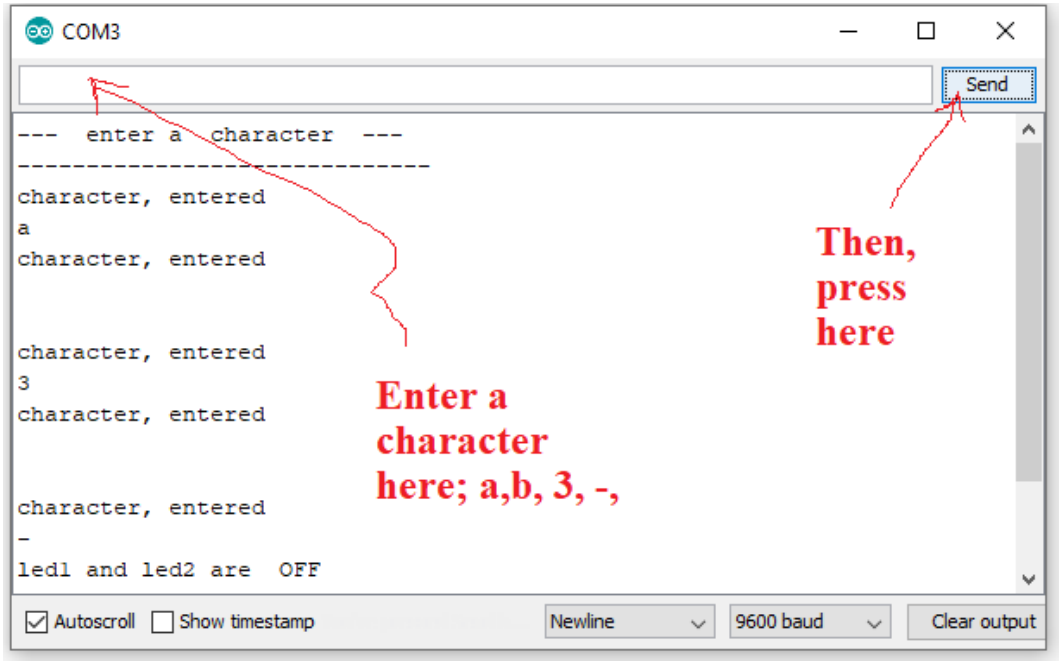
Circuit title: " Sending data from the serial monitor."

Circuit Explanation: If the "A" character on the keyboard is pressed, led1 is ON.
If the "3" character on the keyboard is pressed, led2 is ON.
If the "-" character on the keyboard is pressed, led1 nd led2 is OFF.



/* Sending data from the serial monitor */

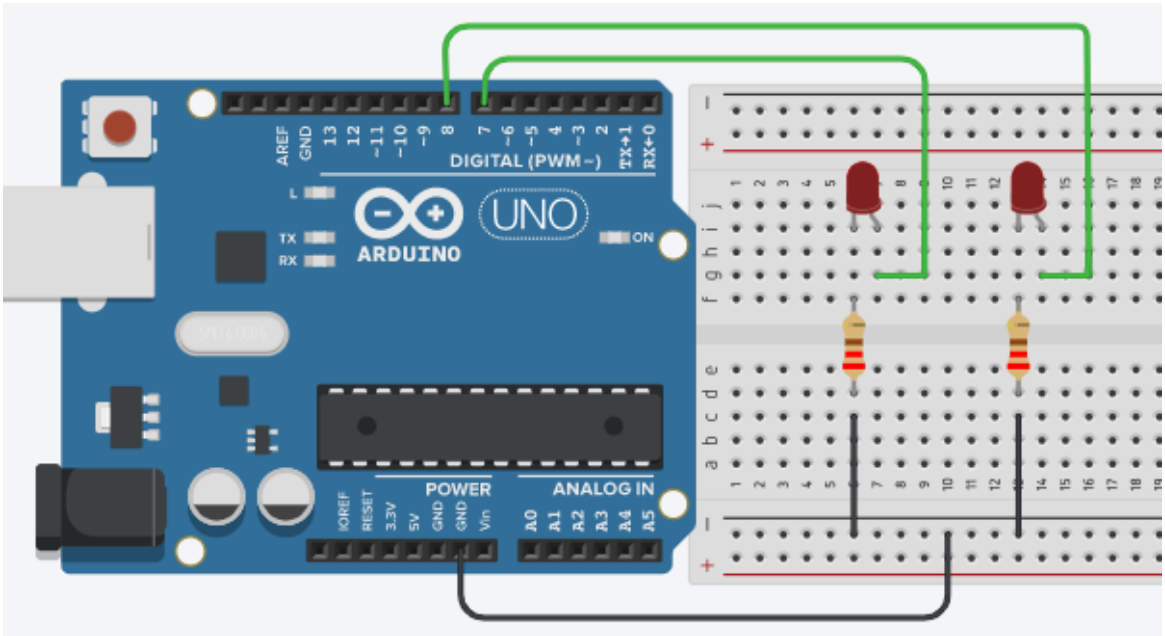
```
char character;
#define  led1 5
#define  led2 6

 void setup()   {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
Serial.begin(9600);
Serial.println ("---  enter a  character  --- ");
Serial.println ("-----------------------------");
}

void loop()
{                                                                    “
if (Serial.available()>0)
{
character=Serial.read();
Serial.println ("character, entered ");
Serial.println (character);

 if (character=='A') digitalWrite (led1, 1);

if (character=='a') digitalWrite (led1, 1);

if (character=='3') digitalWrite (led2, 1);

if (character=='-')
{
digitalWrite(led1,0);
digitalWrite(led2,0);
Serial.println ("led1 and led2 are  OFF ");
}
}
}
```



Circuit 12:

Circuit title: "Learning the FOR Command"

Circuit Explanation:



/* "Learning the FOR Command" */

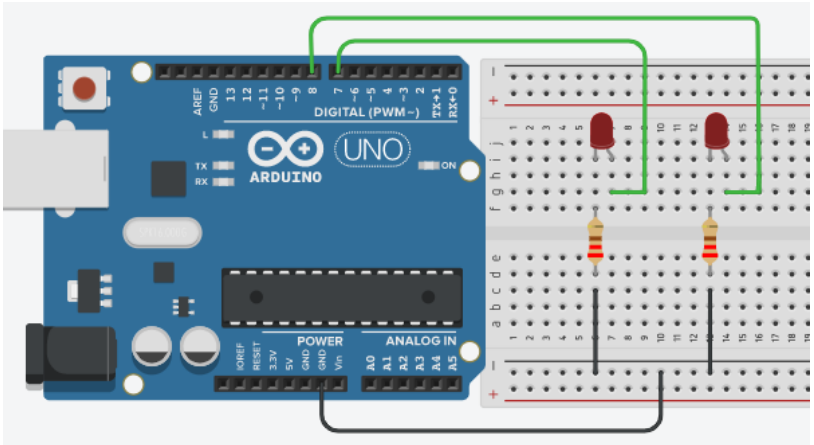
```
int   led1 = 7;
int   led2 = 8;

void setup() {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
}

void loop()
{
for(int i=1; i<6; i++)
{
Serial.println(i);
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);

 digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
}
```

Circuit 13:

Circuit title: " FOR Command versus Serial Monitor"

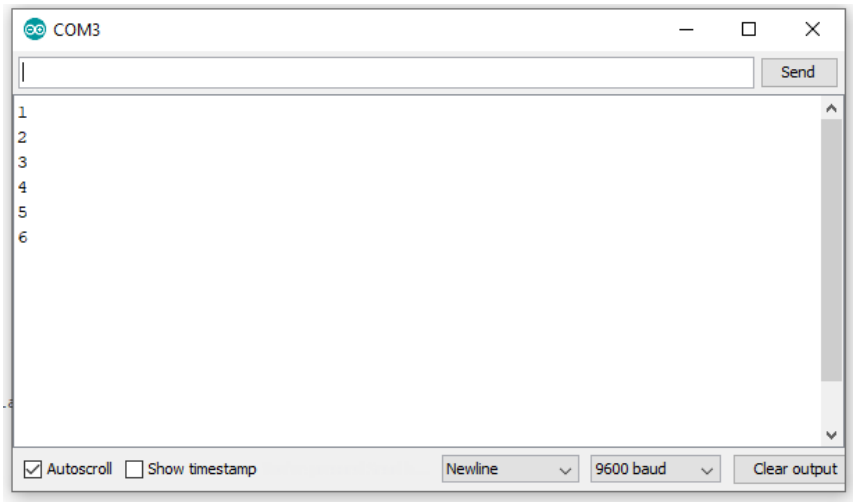
Circuit Explanation: In this application, the LEDs will blink 6 times. Numbers 1, 2, 3, 4, 5,6 will appear on the serial monitor. Then it will be entered in the while loop by exiting the for loop. And the program will stop here. To restart, the reset button must be pressed.

Here, we are using the same circuit as the previous one.



```
/*    "FOR  Command versus Serial Monitor"   */

int   led1 = 7;
int   led2 = 8;

void setup()                  {
Serial.begin(9600);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);      }

void loop() {

  for(int i=1; i<=6; i++)              // the value of i = 1, 2,3, 4,5, 6

  {

 Serial.println(i);                    // Write the values of i, on the serial monitor
digitalWrite(led1, 1);
digitalWrite(led2, 1);
delay(1000);
digitalWrite(led1, 0);
digitalWrite(led2, 0);
delay(1000);
}
while(1);                  //  the for loop doesn't get into an infinite loop.

}
```

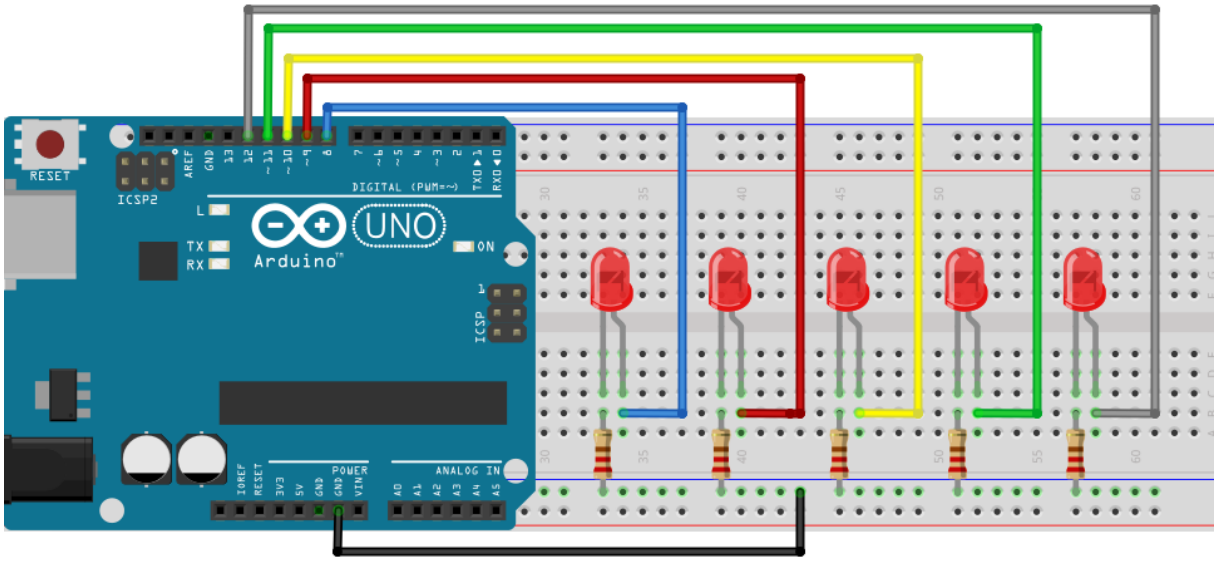
NOTE: To restart, the reset button must be pressed.



Circuit 14:

Circuit title: " Knight Rider with 5 Leds by using the FOR Command "

Circuit Explanation:




```
/*     Knight Rider with 5 Leds  by using the  FOR  Command  */

void setup() {
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
pinMode(11, OUTPUT);
pinMode(12, OUTPUT);     }

void loop() {

 for (int b=8; b<=12; b++)                    // Upcounter  starts here

  {

 digitalWrite(b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);

  }

  for (int b=12; b>=8; b--)                    // Downcounter  starts here

  {
```

```
 digitalWrite (b, HIGH);
delay(150);
digitalWrite (b, LOW);
delay(150);

  }
}
```

Circuit 15:

Circuit title: " Producing random colors with RGB led, Using the switch/case command"

Circuit Explanation: Here, Random command and Switch/Case command are used in the application. In this circuit, red, green, blue colors will be obtained randomly.

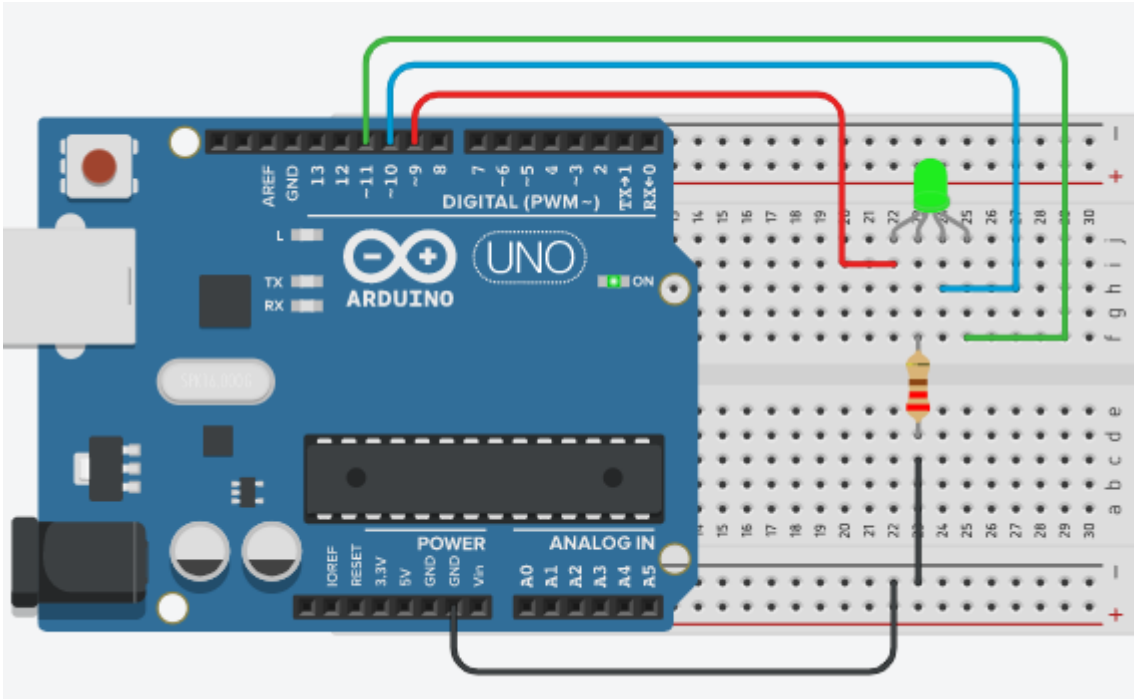
NOTE:

random() : The random function generates random numbers.

Syntax: random(max), random(min, max)

min: lower bound of the random value, (optional).

max: upper bound of the random value.



```
/*     Producing random colors with RGB led,  Using  the switch/case  command */

#define R  9
#define G 10
#define B 11
int colour;
int dly=3000;

void setup() {
pinMode(R, OUTPUT);
pinMode(G, OUTPUT);
pinMode(B, OUTPUT);
Serial.begin(9600);
}

void loop()
{
colour=random(4);
Serial.println(colour);

switch(colour)  {

 case 0:
digitalWrite (R, 1);              //RedLED=ON
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;

 case 1:
digitalWrite (R, 0);            //GreenLED=ON
digitalWrite (G, 1);
digitalWrite (B, 0);
delay(dly);
break;

 case 2:                          //BlueLED=ON
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 1);
delay(dly);
break;

 case 3:                           //No colour
digitalWrite (R, 0);
digitalWrite (G, 0);
digitalWrite (B, 0);
delay(dly);
break;
```

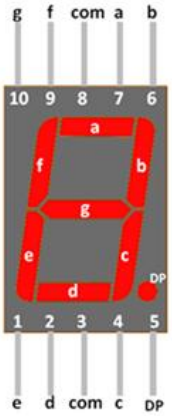
```
}
}
```

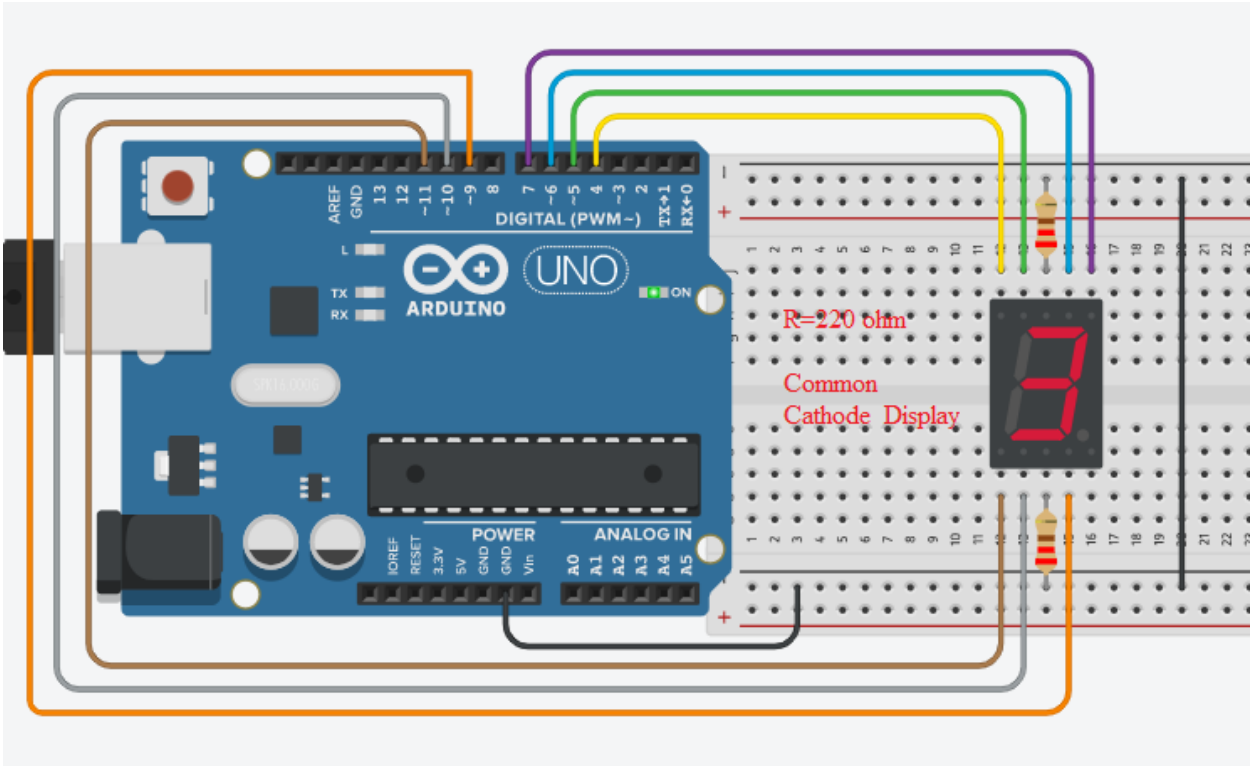
Circuit 16:

Circuit title: " 0-9 UpCounter with 7segment Common-Anode Display "

Circuit Explanation: In order to see a number on the Display, the corresponding LED is lit from the 7 LEDs, represented by the letters a, b, c, d, e, f, g.

NOTE: A seven-segment display is a form of electronic display device for displaying decimal numerals.





```
/*    " 0-9 UpCounter  with 7segment Common-Cathoode Display "    */

int       a=6,   b=7,   c=9,    d=10,     e=11,    f=5,    g=4;
int  number;

void setup() {
pinMode(a,  OUTPUT);
pinMode(b,  OUTPUT);
pinMode(c,  OUTPUT);
pinMode(d,  OUTPUT);
pinMode(e,  OUTPUT);
pinMode(f,  OUTPUT);
pinMode(g,  OUTPUT);
}

void loop()  {
for(number=0;  number< =9;  number++)    {
delay(1000);
switch(number)  {

 case 0:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, LOW);
break;

case 1:
digitalWrite (a, LOW);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;

case 2:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, LOW);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, LOW);
digitalWrite (g, HIGH);
break;

case 3:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, HIGH);
break;

case 4:
digitalWrite (a,LOW );
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

case 5:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
```

```
case 6:
digitalWrite (a, HIGH);
digitalWrite (b, LOW);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

case 7:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, LOW);
digitalWrite (e, LOW);
digitalWrite (f, LOW);
digitalWrite (g, LOW);
break;

case 8:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, HIGH);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;

case 9:
digitalWrite (a, HIGH);
digitalWrite (b, HIGH);
digitalWrite (c, HIGH);
digitalWrite (d, HIGH);
digitalWrite (e, LOW);
digitalWrite (f, HIGH);
digitalWrite (g, HIGH);
break;
}
}
}
```

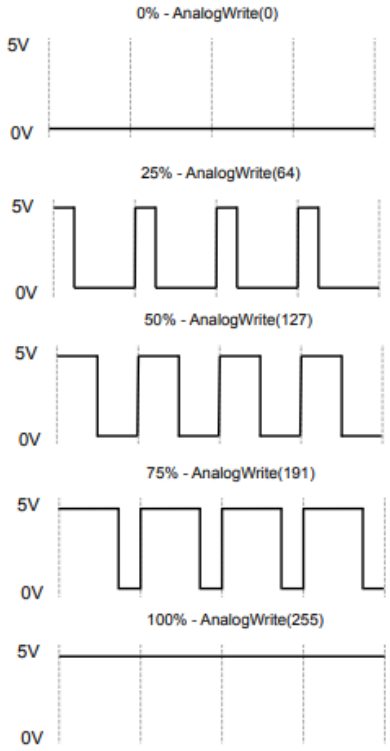
Circuit 17:

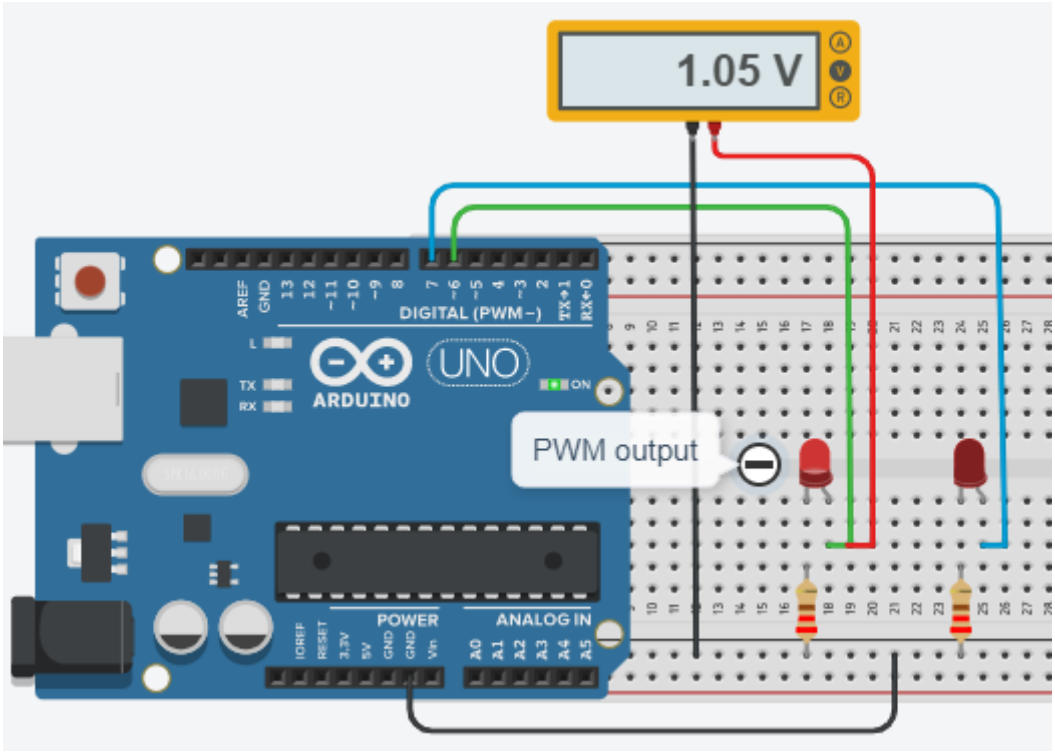
Circuit title: " Using the PWN Technique"

Circuit Explanation: In practice, pin-6 as a PWM output and pin-7 as a digital output are used. Thus, it is aimed to observe the difference between them. Applications such as led brightness adjustment, motor speed control can be performed with the PWM method.

NOTE: The Arduino supports PWM (on certain pins marked with a tilde(~) on your Arduino board - pins 3, 4,5,9,10 and 11) at 500Hz. (500 times a second.) You can give it a value between 0 and 255. 0 means that it is never 5V. 255 means it is always 5V. To do this you make a call to analogWrite() with the value. The ratio of "ON" time to total time is called the "duty cycle". A PWM output that is ON half the time is said to have a duty cycle of 50%.

You can think of PWM as being on for x/255 where x is the value you send with analogWrite(). Below is an example showing what the pulses look like:


0% - AnalogWrite(0)
5V
0V
25% - AnalogWrite(64)
5V
0V
50% - AnalogWrite(127)
5V
0V
75% - AnalogWrite(191)
5V
0V
100% - AnalogWrite(255)
5V
0V

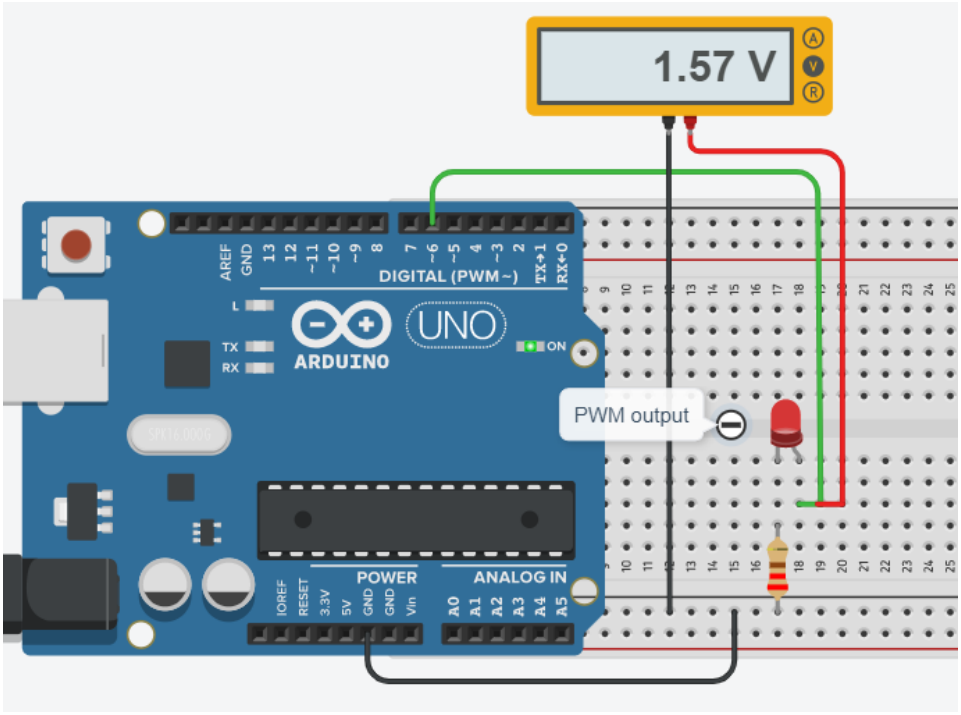

1.05 V
DIGITAL (PWM~)
UNO
ARDUINO
PWM output
POWER
ANALOG IN

```
/* Learnning the PWN Technique by using  analogWrite()   */

#define    led1  6
#define    led2  7

void setup()   {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);       }

void loop()                {

 analogWrite(led1, 60);        //  The value of 60 is sent to Led1 pin, i.e 1.05 V.

 analogWrite(led2,60);         //  The value of 60 is sent to Led2 pin, i.e 1.05 V.

delay (100);                   // only, the led1   lights.
}
```

Circuit 18:

Circuit title: " To change the brightness of an LED from minimum to maximum."

Circuit Explanation: By using the PWM technique, the brightness of an LED from minimum to maximum is changed. Here, analogWrite () and the for commands will be used together.



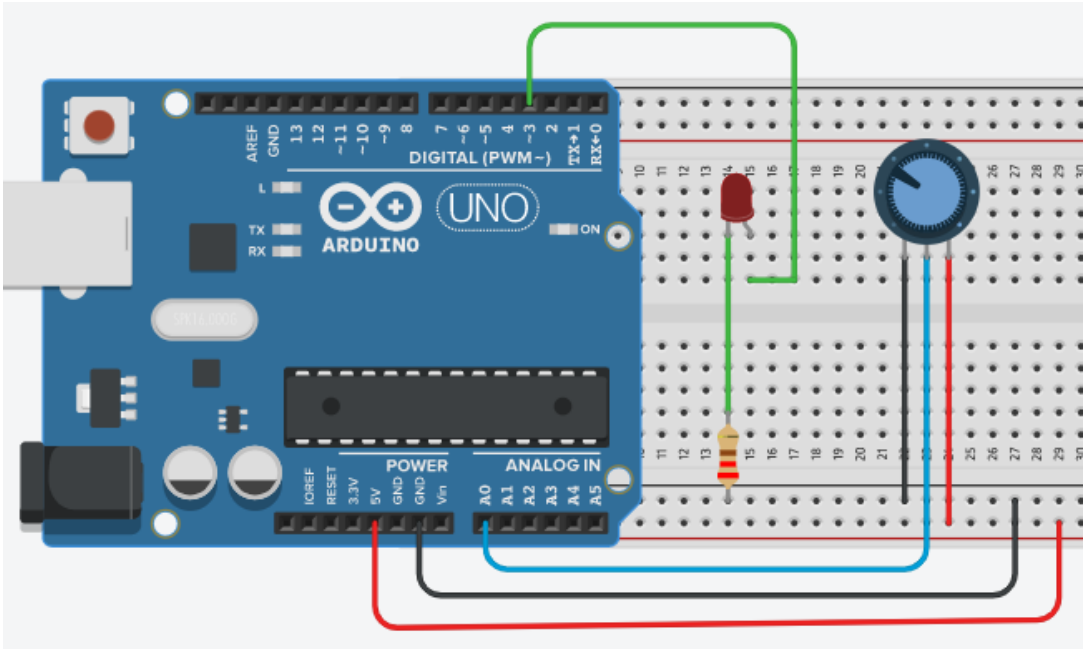
/* To change the brightness of an LED from minimum to maximum */

```
#define led1    6
int a;

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

void loop() {

 for (a=0; a<=255; a++) {

 Serial.println(a);

analogWrite(led1, a);

 delay (100);
}                }
```

NOT: Values ranging from 0 to 5 Volts are displayed on the voltmeter. Counting numbers from 0 to 255 are displayed on the serial monitor.

Circuit 19:

Circuit title: " Fotentiometer fades led."
Circuit Explanation: By using the potentiometer (10K), the brightness of an LED from minimum to maximum is changed. Here, the function of map() is used.



Note:

map() Function:

Re-maps a number from one range to another. That is, a value of **fromLow** would get mapped to **toLow**, a value of **fromHigh** to **toHigh**, values in-between to values in-between, etc.

Does not constrain values to within the range, because out-of-range values are sometimes intended and useful. The constrain() function may be used either before or after this function, if limits to the ranges are desired.

The map() function uses integer math so will not generate fractions, when the math might indicate that it should do so. Fractional remainders are truncated, and are not rounded or averaged.

Syntax: map(value, fromLow, fromHigh, toLow, toHigh);

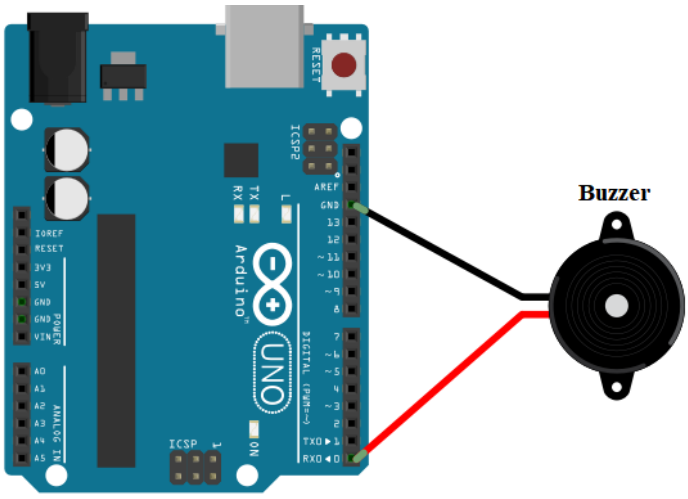
Example: val = map(val, 0, 1023, 0, 255);

```
/*     Fotentiometer fades led      */

int LED_PIN = 3;

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

void loop() {

  // reads the input on analog pin A0 (value between 0 and 1023)

 int analogValue = analogRead(A0);

  // scales it to brightness (value between 0 and 255)

 int brightness = map(analogValue, 0, 1023, 0, 255);

  // sets the brightness LED that connects to  pin 3

 analogWrite(LED_PIN, brightness);

  // print out the value

  Serial.print("Analog: ");

  Serial.print(analogValue);

  Serial.print(" Brightness: ");

  Serial.println(brightness);

 delay(100);
// Serial Monitor screen i below
}
```

Circuit 20:

Circuit title: " To use a buzzer"

Circuit Explanation: A buzzer is an audio signal device, which may be mechanical, electromechanical, or piezoelectric (piezo for short). Typical uses of buzzers in the industry is as an alarm devices, which makes a buzzing or beeping noise while need buzzing.

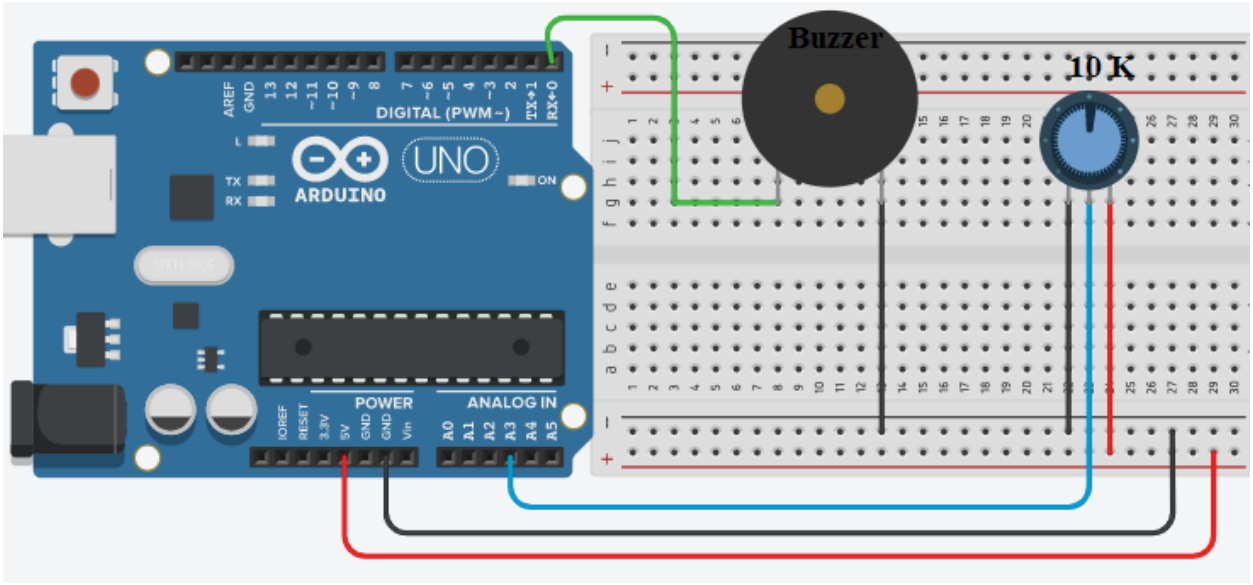



```
/* To use a buzzer  in Arduino circuits*/

#define buzzer_pin  0

 void setup()  {

 pinMode(buzzer_pin, OUTPUT);        }

void loop()       {

 digitalWrite(buzzer_pin, HIGH);

 delay(500);

 digitalWrite(buzzer_pin, LOW);

 delay(500);      }
```

Circuit 21:

Circuit title: " To control a buzzer with Potentiometer "

Circuit Explanation: " When the the value of potentiometer is higher than 500, the buzzer sounds."




```
/* To control a  buzzer with Potentiometer  */

const int POT_PIN = A3;              // Arduino pin connected to Pot pin
const int BUZZER_PIN  = 0;           // Arduino pin connected to Buzzer's pin
const int ANALOG_THRESHOLD  = 500;
int analogValue;

void setup() {

  pinMode(BUZZER_PIN, OUTPUT);       // set arduino pin to output mode

}

void loop() {

 analogValue = analogRead(POT_PIN);            // read the input on analog pin

if(analogValue > ANALOG_THRESHOLD)
digitalWrite(BUZZER_PIN, HIGH);                // turn on Buzzer

else
digitalWrite(BUZZER_PIN, LOW);                 // turn off Buzzer

}
```

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Project Title: "Teaching and Learning Arduinos in Vocational Training"

Project Acronym: "ARDUinVET"

Project No: "2020-1-TR01-KA202-093762"

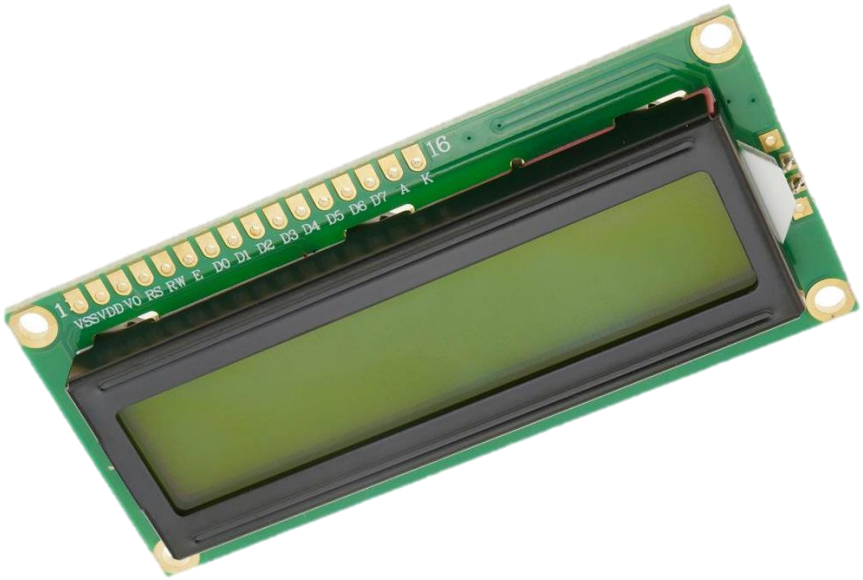
LCD Module and Training KIT

LCD Module and Training KIT

In this module we want to explain how to display status messages or sensor readings of Arduino on LCD displays . They are extremely common and a fast way to add a readable interface to your project.
An LCD is short for Liquid Crystal Display. It is a display unit which uses liquid crystals to produce a visible image. When current is applied to this special kind of crystal, it turns opaque blocking the backlight that lives behind the screen. As a result that particular area will become dark compared to other. And that's how characters are displayed on the screen.

Interfacing 16×2 Character LCD Module with Arduino

There are different kind of LCD displays that you can connect to your Arduino. The most common one is based on parallel interface LCD controller chip from Hitachi called the HD44780.
These LCDs are ideal for displaying text/characters only, hence the name 'Character LCD'. The display has an LED backlight and can display 32 ASCII characters in two rows with 16 characters on each row.
There is a little rectangles for each character on the display, each of these rectangles is a grid of 5×8 pixels.
Although they display only text, they do come in many sizes and colors: for example, 16×1, 16×4, 20×4, with white text on blue background, with black text on green and many more.
The Arduino community has already developed a library to handle HD44780 LCDs (**LiquidCrystal Library)**; so we'll have them interfaced in a few time.



Below is the LCD pinout:

- **VSS:** is a ground pin and should be connected to the ground of Arduino;
-
- **VDD:** connected to 5 V;
- **VD:** for contrast adjustment (connected to the central pin of the potentiometer) ;
- **RS:** controls in which lcd memory zone the sent data are stored;
- **R/W:** pin to select read/write mode;
- **E:** if enabled, allows the LCD module to perform special instructions;
- **D0 to D7:** data transmission;
- **A and K:** anode and cathode to provide backlight to the LCD module.

Arduino - LCD Functions (Commands)

LiquidCrystal lcd() - Creates a variable of type LiquidCrystal. The display can be controlled using 4 or 8 data lines. If the former, omit the pin numbers for d0 to d3 and leave those lines unconnected. The RW pin can be tied to ground instead of connected to a pin on the Arduino; if so, omit it from this function's parameters. Syntax:

LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)
LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)
LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

Parameters:
rs: the number of the Arduino pin that is connected to the RS pin on the LCD

rw: the number of the Arduino pin that is connected to the RW pin on the LCD (optional)

enable: the number of the Arduino pin that is connected to the enable pin on the LCD

d0, d1, d2, d3, d4, d5, d6, d7: the numbers of the Arduino pins that are connected to the corresponding data pins on the LCD. d0, d1, d2, and d3 are optional; if omitted, the LCD will be controlled using only the four data lines (d4, d5, d6, d7).

lcd.begin() - Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. begin() needs to be called before any other LCD library commands. Syntax:
lcd.begin(cols, rows)

Parameters:
lcd: a variable of type LiquidCrystal

cols: the number of columns that the display has

rows: the number of rows that the display has

lcd.print() - Prints text to the LCD. Syntax:

lcd.print(data)
lcd.print(data, BASE)

Parameters:
lcd: a variable of type LiquidCrystal

data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

Returns
byte print() will return the number of bytes written, though reading that number is optional

lcd.setCursor() - Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed. Syntax:

lcd.setCursor(col, row)

Parameters:
lcd: a variable of type LiquidCrystal

col: the column at which to position the cursor (with 0 being the first column)

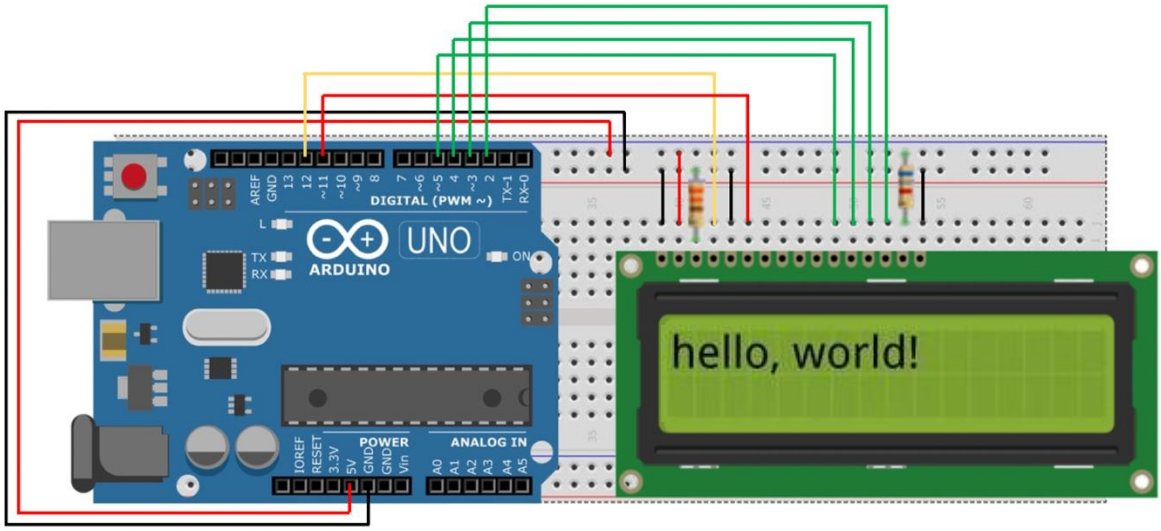
row: the row at which to position the cursor (with 0 being the first row)

lcd.clear(); - Clears the LCD screen and positions the cursor in the upper-left corner.

Syntax: lcd.clear()

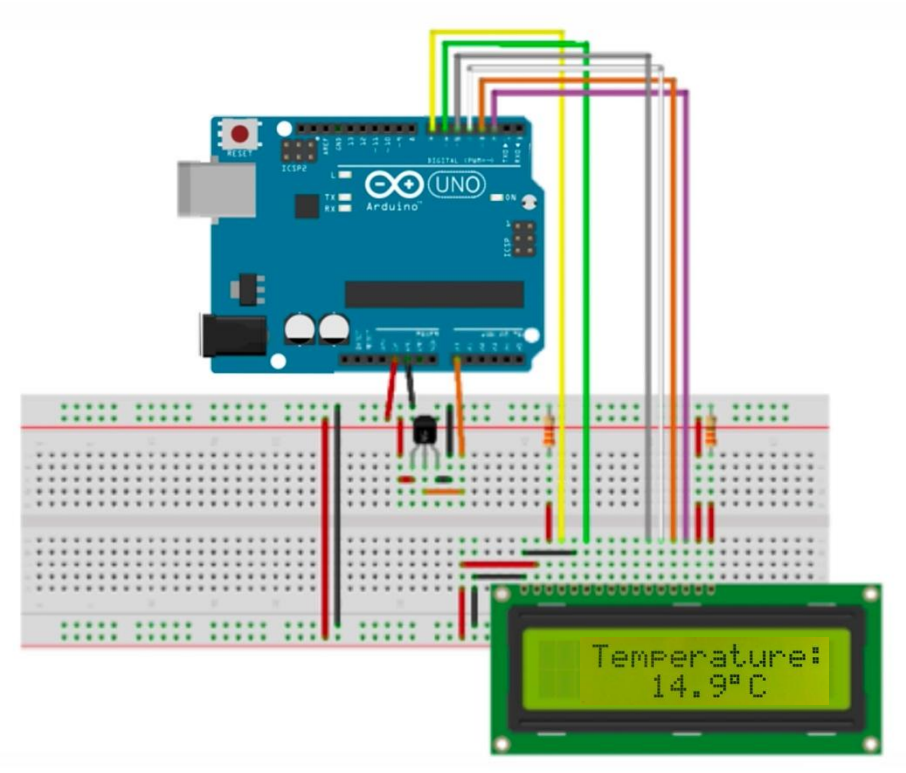
Parameters: lcd: a variable of type LiquidCrystal

Circuit 1:
Circuit title: "Print a message to the LCD"
Circuit Explanation: it is possible to connect an LCD display to the microcontroller and print the desired messages on the screen.
Note: you have to include the library LiquidCrystal.h to use the LCD functions descripted below.




```
/*  Print a message  */

#include <LiquidCrystal.h> // include the library code

//constants for the number of rows and columns in the LCD

const int numRows = 2;

const int numCols = 16;

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()

{

lcd.begin(numCols, numRows);

lcd.print("hello, world!"); // Print a message to the LCD.

}
```

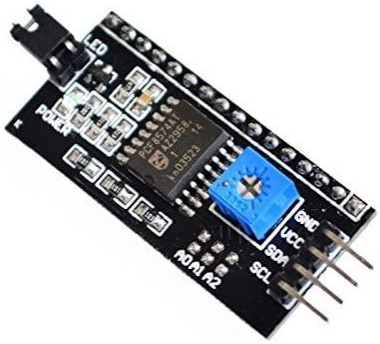
Circuit 2:
Circuit title: "Digital Thermometer"
Circuit Explanation: create a digital thermometer with Arduino. The temperature will be taken with the LM35 temperature sensor and must be displayed on an LCD display and updated every second.
The Vo pin of the LCD display adjusts the contrast of the characters displayed in the display. As mentioned above, it must be connected to a 3.3 kΩ resistance connected to GND. However, in some displays you may need to connect the Vo pin directly to GND
Note: the LM35 sensor has 3 terminals: one for the power supply, one for mass and one for the output of the voltage proportional to the detected temperature, which is equal to 10 mV for each degree centigrade, and is calibrated in degrees Celsius.



```
/*  Digital Thermometer  */

#include <LiquidCrystal.h> //Library to drive LCD display

#define pin_temp A0 //Temperature sensor Vout foot connection pin

float temp = 0; //Variable in which the detected temperature will be stored

LiquidCrystal lcd(7, 6, 5, 4, 3, 2); //Initializing the library with LCD display pins

void setup()

{
  lcd.begin(16, 2); //Setting the number of columns and rows in the display
  LCD  lcd.setCursor(0, 0); //Move the cursor over the first row (row 0) and the first
  column lcd.print ("Temperature:"); Print the message 'Temperature:' on the first line

 //*Imposed ADC Vref at 1.1V
 (for greater accuracy in temperature calculation)

  IMPORTANT: If you use Arduino Mega replace INTERNAL with INTERNAL1V1 */
  analogReference(INTERNAL);
}


void loop()
{
  /*calculate the temperature ==============================================*/
  temp = 0;

  for (int i = 0; i < 5; i++) { //It executes the next statement 5 times
    temp += (analogRead(pin_temp) / 9.31); // It calculates temperature and sum at variable
    'temp'

  }
  temp /= 5; //It calculates the mathematical average of temperature values
 /*=======================================================================*/

  /*I see the temperature on the LCD display
  ========================================*/
  lcd.setCursor(0, 1); //lcd.print(temp); Move the cursor over the first column
  and the second row lcd.print(temp);
  // Mold on LCD display temperature

  lcd.print(" C"); // Mold a space and the 'C' font on the display
 /*=======================================*/

  delay(1000); //Delay by one second (it can be changed)
}
```

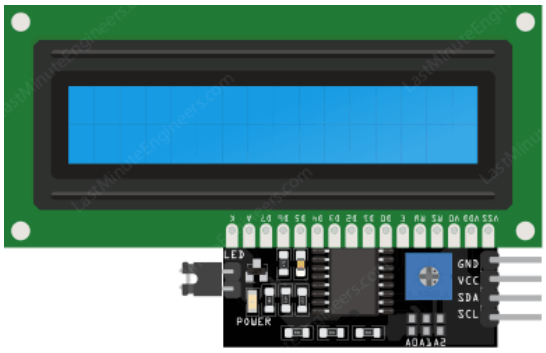
Interfacing I2C LCD with Arduino

If you want to connect an LCD display with Arduino, you have to consume a lot of pins on the Arduino. Even in 4-bit mode, the Arduino still requires a total of seven connections, which is half of the available digital I/O pins.
The solution is to use an LCD display that interfaces with I2C protocol. It only consumes two I/O pins which can be not even part of a digital I/O pin set and can be shared with other I2C devices as well.
The most diffuse I2C LCD display consists of a HD44780 based character LCD display and an I2C LCD adapter.



The most important part of the adapter is the 8-Bit I/O Expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required by the LCD display. The board also comes with a small potentiometer to make fine adjustments to the contrast of the display. If you are using more than a device on the same I2C bus, you may need to set a different I2C address for the board, so that it does not conflict with another I2C device. For this reason, the board has three solder jumpers (A0, A1 and A2).

An I2C LCD has only 4 pins that interface it to the Arduino.



The pinout is as follows:

- **GND:** is a ground pin and should be connected to the ground of Arduino;

- **VCC:** supplies power to the module and the LCD. Connect it to the 5V output of the Arduino or a separate power supply;

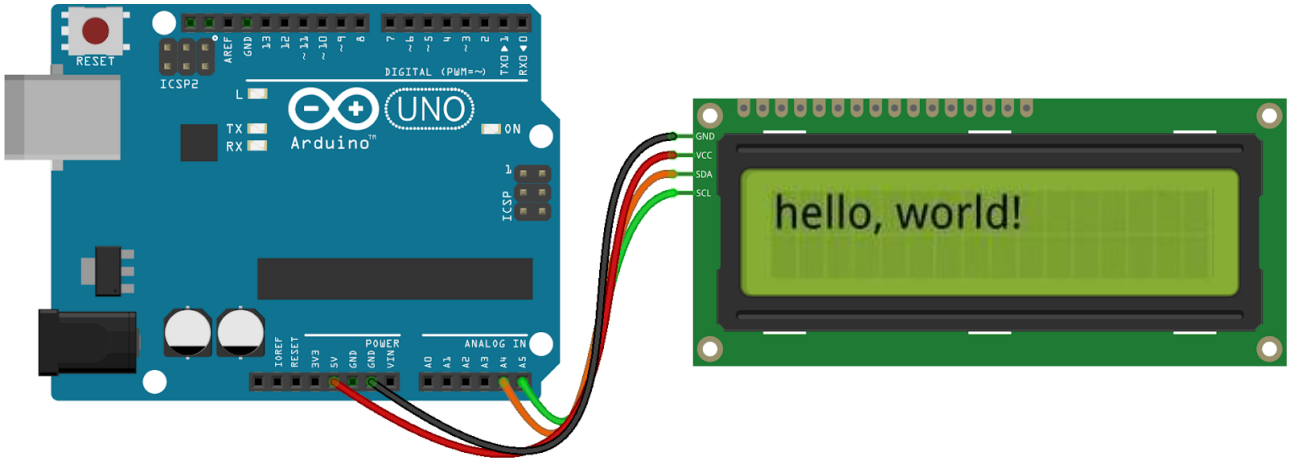
- **SDA:** is a Serial Data pin. This line is used for both transmit and receive. Connect to the SDA pin on the Arduino;

- **SCL:** is a Serial Clock pin. This is a timing signal supplied by the Bus Master device. Connect to the SCL pin on the Arduino.

On the Arduino boards with the R3 layout, the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. They are also known as A5 (SCL) and A4 (SDA). Refer the below table to identify the correct pins depending on the Arduino model.

	SCL	SDA
Arduino Uno	A5	A4
Arduino Nano	A5	A4
Arduino Mega	21	20
Leonardo/Micro	3	2

Circuit 3:
Circuit title: "Print a message to the I2C LCD"
Circuit Explanation: it is possible to connect an LCD display to the microcontroller with only 4 pins and print the desired messages on the screen.
Note: you have to install a library called LiquidCrystal_I2C. This library is an improved version of the LiquidCrystal library that comes packaged with your Arduino IDE.



```
/*  Print a message on a I2C Display */

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,16,2);  // set the LCD address to 0x3F for a 16 chars and 2 line display

void setup() {

  lcd.init();

  lcd.clear();

  lcd.backlight();      // Make sure backlight is on

  // Print a message on both lines of the LCD.

  lcd.setCursor(2,0);   //Set cursor to character 2 on line 0

  lcd.print("Hello world!");

  lcd.setCursor(2,1);   //Move cursor to character 2 on line 1

  lcd.print("with I2C protocol!");

}

void loop() {

}
```

Interfacing OLED Graphic Display Module with Arduino

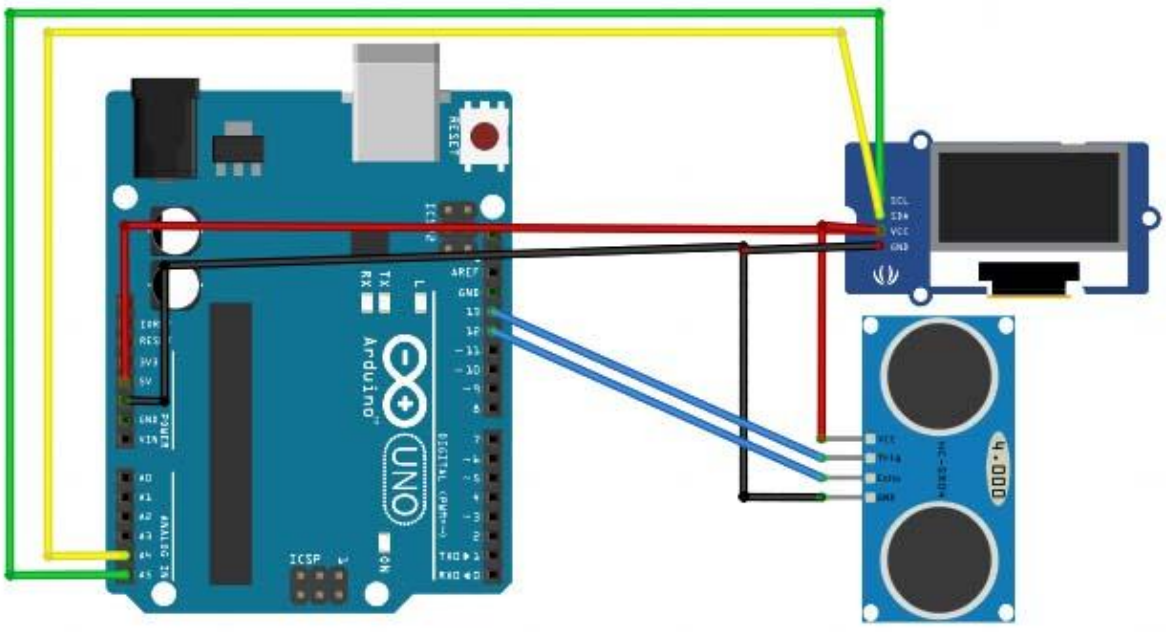
Another possibility to use the I2C protocol is choosing an OLED (Organic Light-Emitting Diode) display. They're super-light and thin, and produce a brighter and crisper picture.

An OLED display works without a backlight. This is why the display has such high contrast, extremely wide viewing angle and can display deep black levels. Absence of backlight significantly reduces the power required to run the OLED.
As we can see from the figure the pinout is the classic four-pin I2C interface already seen above.
The Arduino community has already developed a few libraries to handle these OLED displays, such as Adafruit's SSD1306 library. To install the library navigate to the Sketch > Include Library > Manage Libraries… Wait for Library Manager to download libraries index and update list of installed libraries.

Arduino - OLED Graphic Display Module Functions (Commands)

pinMode();
display.begin()
display.clearDisplay();

Circuit 4:
Circuit title: "Distance sensor"
Circuit Explanation: The distance will be taken with the HC-SR04 ultrasonic sensor and be displayed on a OLED display.
Note: There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground).



```
/*  Distance sensor  */

#include <SPI.h>        // this library allows you to communicate with SPI devices, with the
                        Arduino as the master device.

#include <Wire.h>      // this library allows you to communicate with I2C / TWI devices

#include <Adafruit_GFX.h>            // the OLED display's libraries
#include <Adafruit_SSD1306.h>

#define CommonSenseMetricSystem
#define trigPin 13      // define the pins of the sensor
#define echoPin 12

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); //initialize with the I2C addr 0x3C
(128x64)
  display.clearDisplay();

}

void loop() {
  long duration, distance;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;
  display.setCursor(22,20);    //oled display setting cursor
  display.setTextSize(3);        //size of the text
  display.setTextColor(WHITE);       //if you write black it erases things
  display.println(distance);     //print our variable
  display.setCursor(85,20);
  display.setTextSize(3);

    display.println("cm");

  display.display();

  delay(500);
  display.clearDisplay();
   Serial.println(distance);//debug
```

}

Interfacing Nokia 5110 Graphic LCD Display with Arduino

You can interface Arduino with little LCDs similar to that Nokia used in their 3310 and 5110 cell phones. these displays are small (only about 1.5″), inexpensive, easy to use, fairly low power (as low as 6 to 7mA only) and can display text as well as bitmaps.

These are graphic display of 84×48 pixels. They interfaces to microcontrollers through a serial bus interface similar to SPI. The LCD also comes with a backlight in different colors such as red, green, blue and white. The backlight is nothing but four LEDs spaced around the edges of the display.
It has 8 pins that interface it to the Arduino, the pinout is as follows:

- **RST:** resets the display. It's an active low pin meaning; you can reset the display by pulling it low. You can also connect this pin to the Arduino reset so that it will reset the screen automatically;

- **CE(Chip Enable**): is used to select one of many connected devices sharing same SPI bus;

- **D/C(Data/Command):** pin tells the display whether the data it's receiving is a command or displayable data;

- **DIN:** is a serial data pin for SPI interface;

- **CLK:** is a serial clock pin for SPI interface;

- **VCC:** supplies power for the LCD which we connect to the 3.3V volts pin on the Arduino;

- **BL(Backlight):** controls the backlight of the display. To control its brightness, you can add a potentiometer or connect this pin to any PWM-capable Arduino pin;

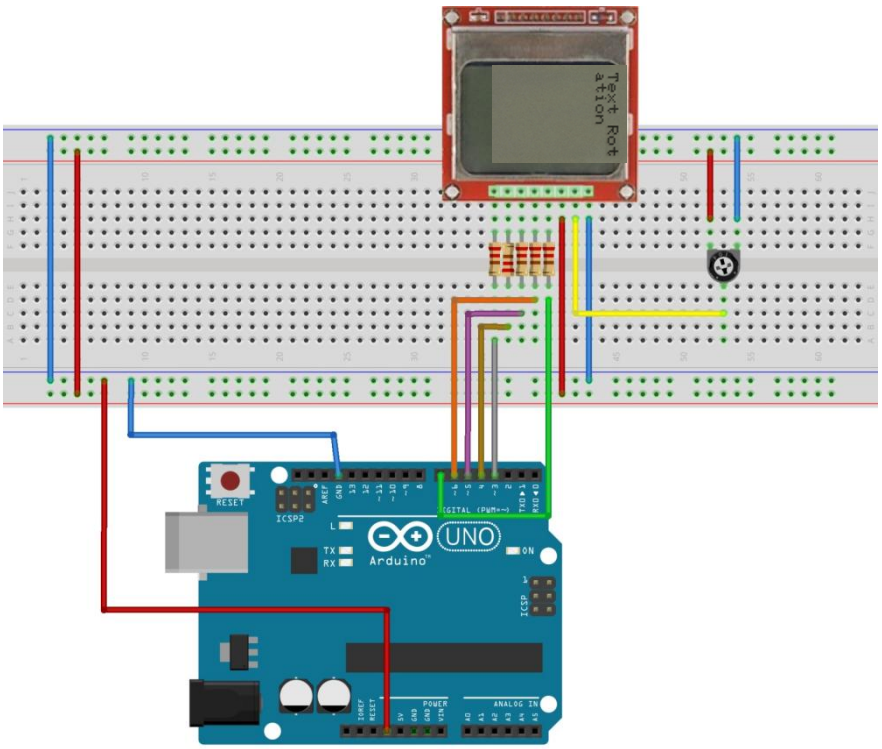
- **GND:** is a ground pin and should be connected to the ground of Arduino.

You can connect data transmission pins to any digital I/O pin. The LCD has 3v communication levels, so we cannot directly connect these pins to the Arduino. One way is to add resistors inline

with each data transmission pin. Just add 10kΩ resistors between the CLK, DIN, D/C, and RST pins and a 1kΩ resistor between CE. The backlight(BL) pin is connected to 3.3V via 330Ω current limiting resistor. You can add a potentiometer or connect this pin to any PWM-capable Arduino pin, if you wish to control its brightness.
The Arduino community has already developed a few libraries to handle these NOKIA displays, such as Adafruit's PCD8544 Nokia 5110 LCD library. To install the library navigate to the Sketch > Include Library > Manage Libraries… Wait for Library Manager to download libraries index and update list of installed libraries.

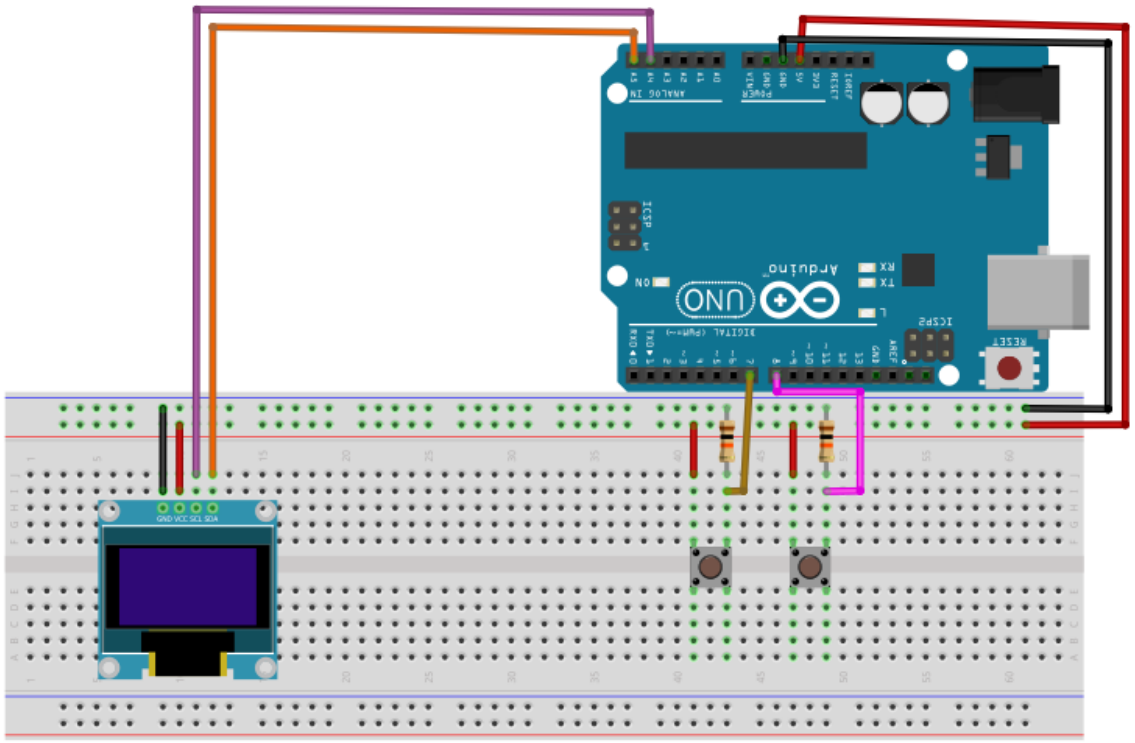
Circuit 5:
Circuit title: "Text Rotation"
Circuit Explanation: You can rotate the contents of the display by calling setRotation() function. It allows you to view your display in portrait mode, or flip it upside down.
Note: The function accepts only one parameter that corresponds to 4 cardinal rotations. This value can be any non-negative integer starting from 0. Each time you increase the value, the contents of the display are rotated 90 degrees counter clockwise. For example:

- 0 – Keeps the screen to the standard landscape orientation.
- 1 – Rotates the screen 90° to the right.
- 2 – Flips the screen upside down.
- 3 – Rotates the screen 90° to the left.



```
/*  Text Rotation  */

#include <SPI.h>        // this library allows you to communicate with SPI devices, with the
                        Arduino as the master device.

#include <Adafruit_GFX.h>           // the OLED display's libraries
#include <Adafruit_PCD8544.h>

// Declare LCD object for software SPIAdafruit_PCD8544(CLK,DIN,D/C,CE,RST);
Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

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

        //Initialize Display
        display.begin();

        display.setContrast(57);        // you can change the contrast around to adapt the display

        display.clearDisplay();         // clear the buffer.

// Text Rotation
while(1)
{
        display.clearDisplay();
        display.setRotation(rotatetext);
        display.setTextSize(1);
        display.setTextColor(BLACK);
        display.setCursor(0,0);
        display.println("Text Rotation");
        display.display();
        delay(1000);
        display.clearDisplay();
        rotatetext++;
}
}

void loop() {}
```

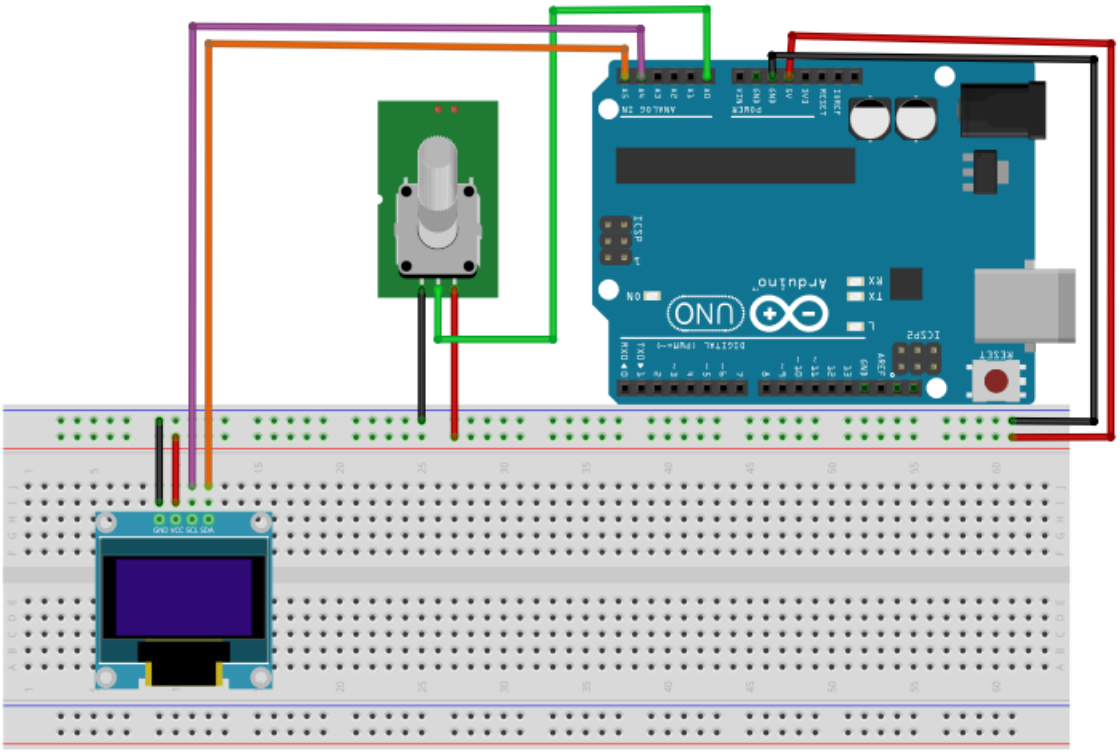
Circuit 6:
Circuit title: "Button counter"
Circuit Explanation: an OLED display showing a number, which can be incremented and decremented at the click of two different buttons.



```
#include <U8glib.h> //lcd libraries
#include "U8glib.h"
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);  //display model

int button_plus = 8, button_minus = 7; //declaration pin buttons
int state_plus, state_minus, number=0;

void setup() {

  pinMode(button_plus,INPUT);
  pinMode(button_minus,INPUT);

  Serial.begin(9600);

  u8g.setFont(u8g_font_fub25n); //font to be used for the write of the number
}
```

```
void write_number(void) {
  u8g.setPrintPos(10,50);
  u8g.print(number);
}

void loop() {

  //buttons
  state_plus = digitalRead(button_plus);
  state_minus = digitalRead(button_minus);

  if(state_plus==1)
  {
     number++;
     delay(50);
  }

  if(state_minus==1)
  {
    number--;
    delay(50);
  }

  //write de number sul display

  u8g.firstPage();
  do {
    write_number();
  } while  ( u8g.nextPage() );

    u8g.firstPage();
}
```

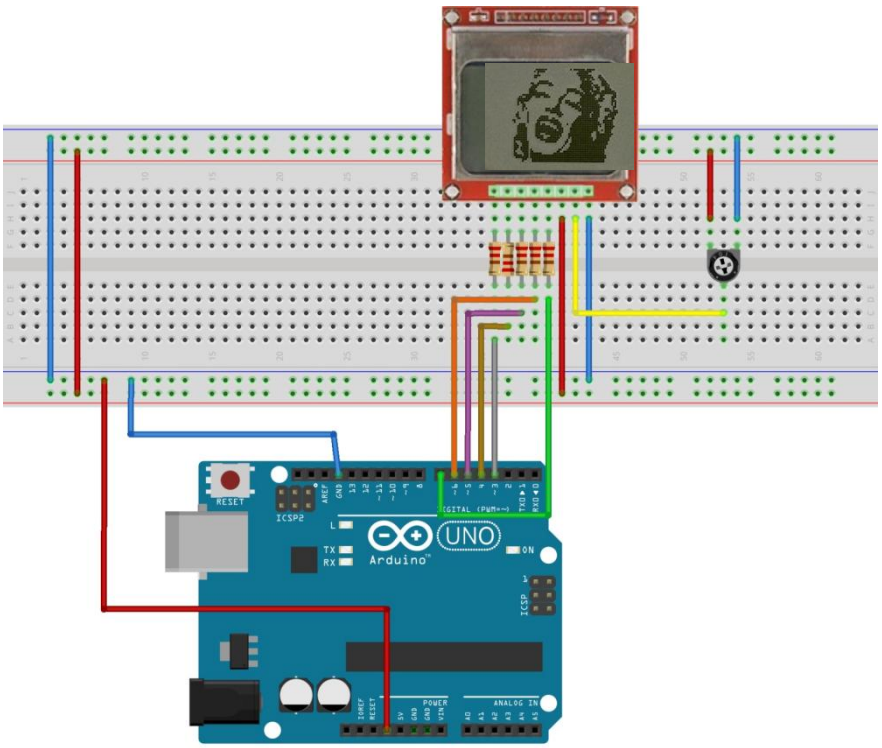
Circuit 7:
Circuit title: "Counter with potentiometer"
Circuit Explanation: an OLED display that shows a number, incrementing and decrementing as a potentiometer is turned.



```
#include <U8glib.h> //lcd libraries
#include "U8glib.h"
U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0);  //display
model

int potentiometer = 0; //declaration and potentiometer
int state_pot, stop_pot, difference, number=0;

void setup() {
  Serial.begin(9600);
  u8g.setFont(u8g_font_fub25n);//font to be used for the write of the number
}

void write_number(void) {
  u8g.setPrintPos(10,50);
  u8g.print(number);
}
```

```
void loop() {
  state_pot = analogRead(potentiometer);

  //potentiometer
  stop_pot = state_pot;//save the value when it is stop to see if the potentiometer turns clockwise
or counterclockwise
  delay(100);
  state_pot = analogRead(potentiometer);
  difference = stop_pot-state_pot;
  if((difference>2)||(difference<2))//is used to avoid making trades if the potentiometer is stop
  {
    number=number-difference/5;//if difference is greater than 0 then it turns clockwise and adds
otherwise it subtracts
    delay(100);
  }

  //write the number on display
  u8g.firstPage();
  do {
    write_number();
  } while
  ( u8g.nextPage() );
    u8g.firstPage();
}
```

Circuit 8:
Circuit title: "Marilyn Bmp Image"
Circuit Explanation: How to draw bitmap images to the Nokia 5110 LCD Display. In this example there is a portrait of Marilyn Monroe.
Note: the screen resolution of Nokia 5110 LCD display is 84×48 pixels, so images larger than that will not display correctly. To show bitmap image on the Nokia 5110 LCD display we need to call drawBitmap() function. It takes six parameters:. top left corner X coordinate, top left corner Y coordinate, byte array of monochrome bitmap, width of bitmap in pixels, height of bitmap in pixels and Color. In our example, the bitmap image is 84×48 in size. So, X & Y coordinates are set to 0 while width & height is set to 84 and 48.




```
/*  Marilyn Bmp Image */

#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>

Adafruit_PCD8544 display = Adafruit_PCD8544(7, 6, 5, 4, 3);

// 'Marilyn Monroe 84x48', 84x48px
const unsigned char MarilynMonroe [] PROGMEM = {
```

0x00, 0x00, 0x00, 0x7f, 0x00, 0x02, 0xfe, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xbe, 0x00,
0x00, 0x1f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x3f, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x00, 0x1f, 0xe1, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0,
0x00, 0x00, 0x0f, 0xf1, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0e, 0xd8, 0xe0,
0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0x80, 0x00, 0x07, 0xe0, 0x70, 0x00, 0x00, 0x00, 0x00, 0x03,
0x3f, 0xe0, 0x00, 0x07, 0xf0, 0x78, 0x00, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x70, 0x00, 0x0f, 0xee,
0x7c, 0x00, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x0f, 0xf7, 0x1c, 0x00, 0x00, 0x00, 0x00,
0x07, 0x80, 0x00, 0x0f, 0xc7, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x00, 0x0f, 0xf3,
0xdf, 0x7f, 0x80, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00, 0x08, 0x7d, 0xef, 0xff, 0xc0, 0x00, 0x00,
0x00, 0x7f, 0xff, 0x80, 0x30, 0x0f, 0xfc, 0xe0, 0xc0, 0x00, 0x00, 0x01, 0x9e, 0x73, 0xc0, 0xe0,
0x07, 0xf8, 0xc1, 0xc0, 0x00, 0x00, 0x03, 0xfc, 0x00, 0x01, 0xc0, 0x0f, 0xfd, 0xe1, 0x80, 0x00,
0x00, 0x03, 0xf8, 0x00, 0x01, 0x9c, 0x0f, 0xff, 0xc1, 0xc0, 0x00, 0x00, 0x02, 0xc0, 0x00, 0x01,
0x9f, 0xbf, 0xfe, 0x01, 0x40, 0x00, 0x00, 0x02, 0x60, 0x00, 0x03, 0x07, 0xef, 0xff, 0x01, 0x40,
0x00, 0x00, 0x00, 0x60, 0x00, 0x07, 0x01, 0xf7, 0xff, 0x80, 0xc0, 0x00, 0x00, 0x00, 0x50, 0x01,
0xdf, 0x00, 0x7f, 0xff, 0x1c, 0x80, 0x00, 0x00, 0x00, 0x40, 0x01, 0xff, 0x00, 0x1f, 0xff, 0x1e,
0xe0, 0x00, 0x00, 0x02, 0x08, 0x00, 0x3f, 0x80, 0x07, 0xef, 0x03, 0xe0, 0x00, 0x00, 0x06, 0x08,
0x00, 0x03, 0xc0, 0x07, 0xdf, 0x07, 0xc0, 0x00, 0x00, 0x06, 0x08, 0x0f, 0x81, 0x80, 0x1f, 0xdf,
0x1f, 0x80, 0x00, 0x00, 0x03, 0x08, 0x1f, 0x98, 0x00, 0x3f, 0xfe, 0x19, 0x80, 0x00, 0x00, 0x18,
0x08, 0x3f, 0xfe, 0x00, 0x7f, 0xfe, 0x3f, 0x00, 0x00, 0x00, 0x08, 0x08, 0x30, 0x3f, 0x00, 0xff,
0xff, 0x3f, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x76, 0x0f, 0x89, 0xff, 0xff, 0x9f, 0x00, 0x00, 0x00,
0x03, 0xe0, 0x7f, 0xc3, 0x81, 0xff, 0xfe, 0x9f, 0x80, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xf3, 0xc3,
0xff, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x7f, 0xfd, 0xc3, 0xff, 0xfe, 0x5e, 0x00, 0x00,
0x00, 0x03, 0xf0, 0x7f, 0xff, 0xc3, 0xff, 0xf3, 0x1e, 0x00, 0x00, 0x00, 0x03, 0xf0, 0x71, 0xff,
0x87, 0xff, 0xe3, 0xff, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x7c, 0x3f, 0x87, 0xff, 0xe3, 0xfe, 0x00,

```
        0x00, 0x00, 0x0f, 0xf0, 0x3c, 0xff, 0x05, 0xff, 0xf3, 0xfc, 0x00, 0x00, 0x00, 0x0f, 0xf0,
0x0f,
        0xfe, 0x09, 0xff, 0xf7, 0xfc, 0x00, 0x00, 0x00, 0x08, 0xf8, 0x01, 0xfc, 0x19, 0xff, 0xff,
0xf8,
        0x00, 0x00, 0x00, 0x0c, 0x78, 0x00, 0x00, 0x13, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00,
0x0e, 0x78,
        0x00, 0x00, 0x23, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x0e, 0xf8, 0x00, 0x00, 0x47, 0xff,
0xff,
        0xf0, 0x00, 0x00, 0x00, 0x0c, 0xfa, 0x00, 0x01, 0x8f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00,
0x08,
        0x7b, 0x00, 0x03, 0x3f, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x0c, 0x3b, 0xf8, 0x0f, 0xff,
0xff,
        0xff, 0xe0, 0x00, 0x00, 0x00, 0x0f, 0xbb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00,
0x00,
        0x07, 0xfb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x71, 0xff, 0xff,
0xff,
        0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x41, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00,
0x00
};

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

        display.begin();

        display.setContrast(57);

        display.clearDisplay();

        // Display bitmap
        display.drawBitmap(0, 0,  MarilynMonroe, 84, 48, BLACK);
        display.display();

        // Invert Display
        //display.invertDisplay(1);
}

void loop() {}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

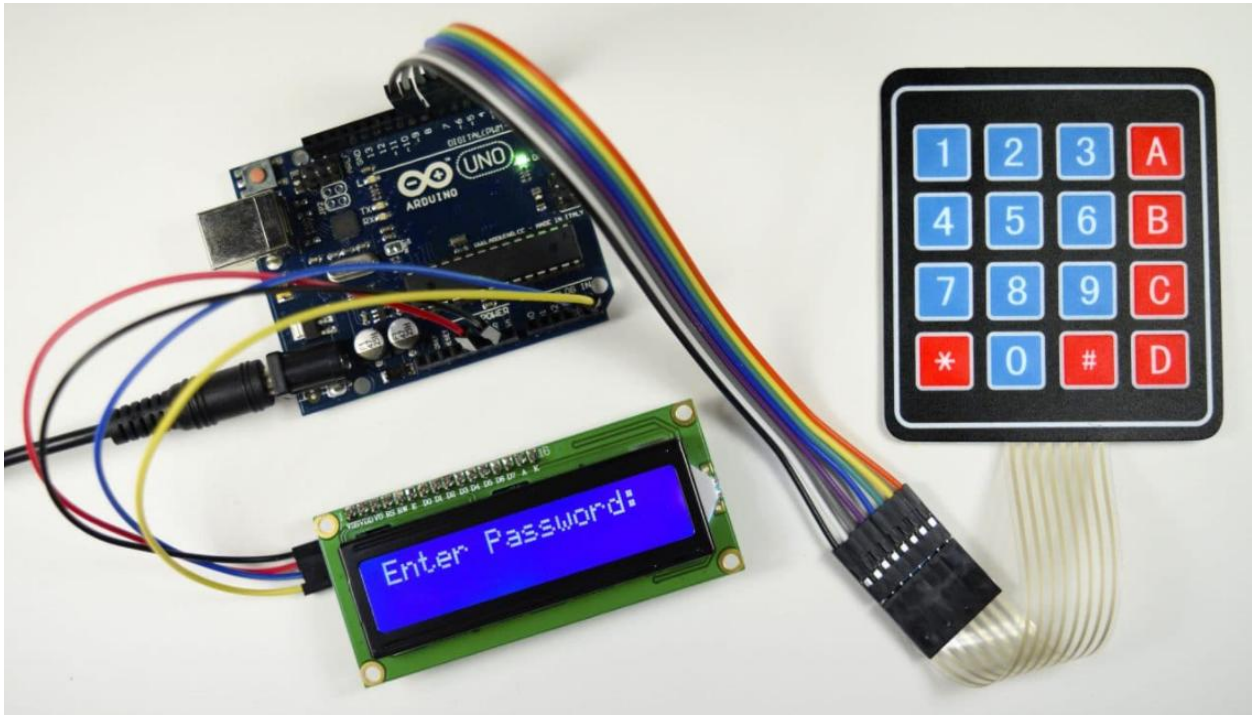
Strategic Partnerships Project for Vocational Education and Training

Project Title: "Teaching and Learning Arduinos in Vocational Training"

Project Acronym: " ARDUinVET "

Project No: "2020-1-TR01-KA202-093762"

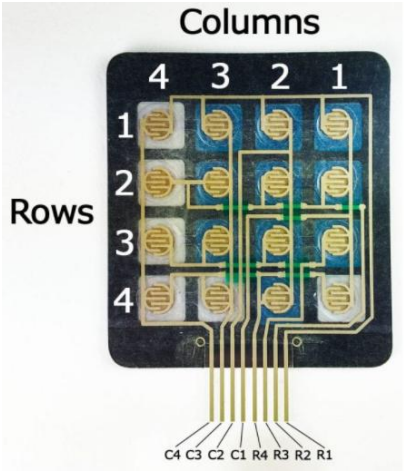
Keypad Module and Training Kit



What a keypad is?

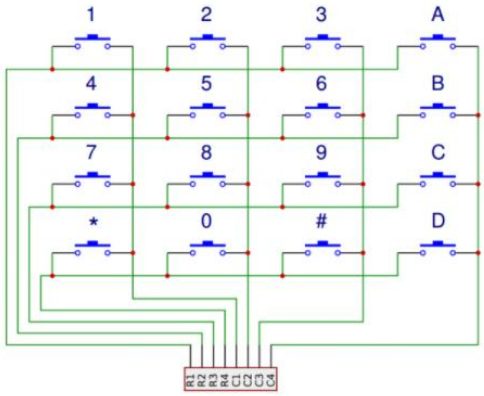
A Keypad is a system of buttons arranged in a matrix that works as a switching device to provide a connection between a line and a column.

We will use a membrane keyboard with 4X4 matrix, because it is thin and has adhesive support, so that it can be glued on most flat surfaces. Beneath each key is a membrane switch. Each switch in a row is connected to the other switches in the row by a conductive trace underneath the pad. Each switch in a column is connected the same way – one side of the switch is connected to all of the other switches in that column by a conductive trace. Each row and column is brought out to a single pin, for a total of 8 pins on a 4X4 keypad.	



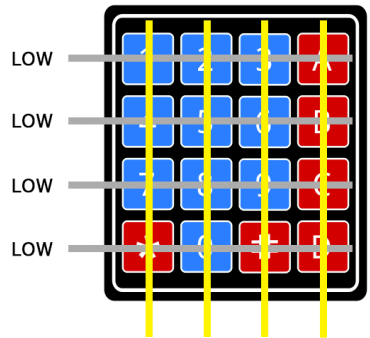
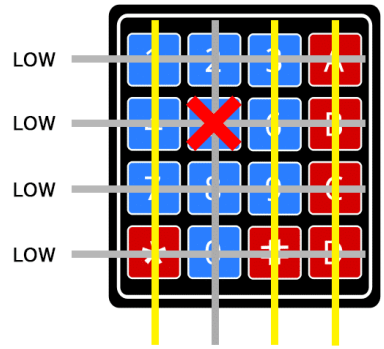
Pressing a button closes the switch between a column and a row trace, allowing current to flow between a column pin and a row pin.

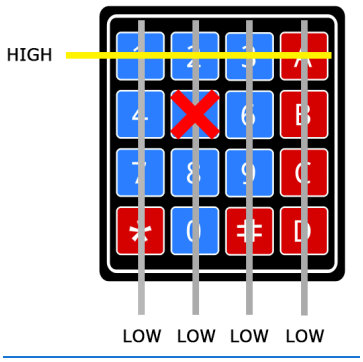
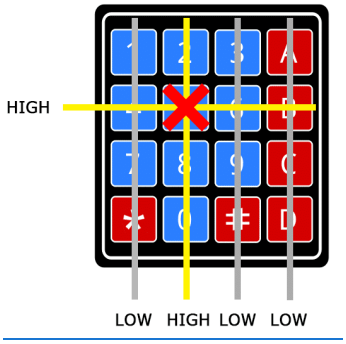
The schematic for a 4X4 keypad shows how the rows and columns are connected:



The Arduino detects which button is pressed by detecting the row and column pin that's connected to the button.

This happens in four steps:

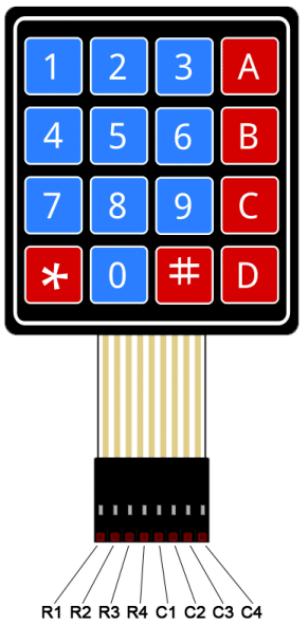
1. First, when no buttons are pressed, all of the column pins are held HIGH, and all of the row pins are held LOW	
2. When a button is pressed, the column pin is pulled LOW since the current from the HIGH column flows to the LOW row pin:	

3. The Arduino now knows which column the button is in, so now it just needs to find the row the button is in. It does this by switching each one of the row pins HIGH, and at the same time reading all of the column pins to detect which column pin returns to HIGH	
4. When the column pin goes HIGH again, the Arduino has found the row pin that is connected to the button:	

From the diagram above, you can see that the combination of row 2 and column 2 could only mean that the number 5 button was pressed.

CONNECT THE KEYPAD TO THE ARDUINO

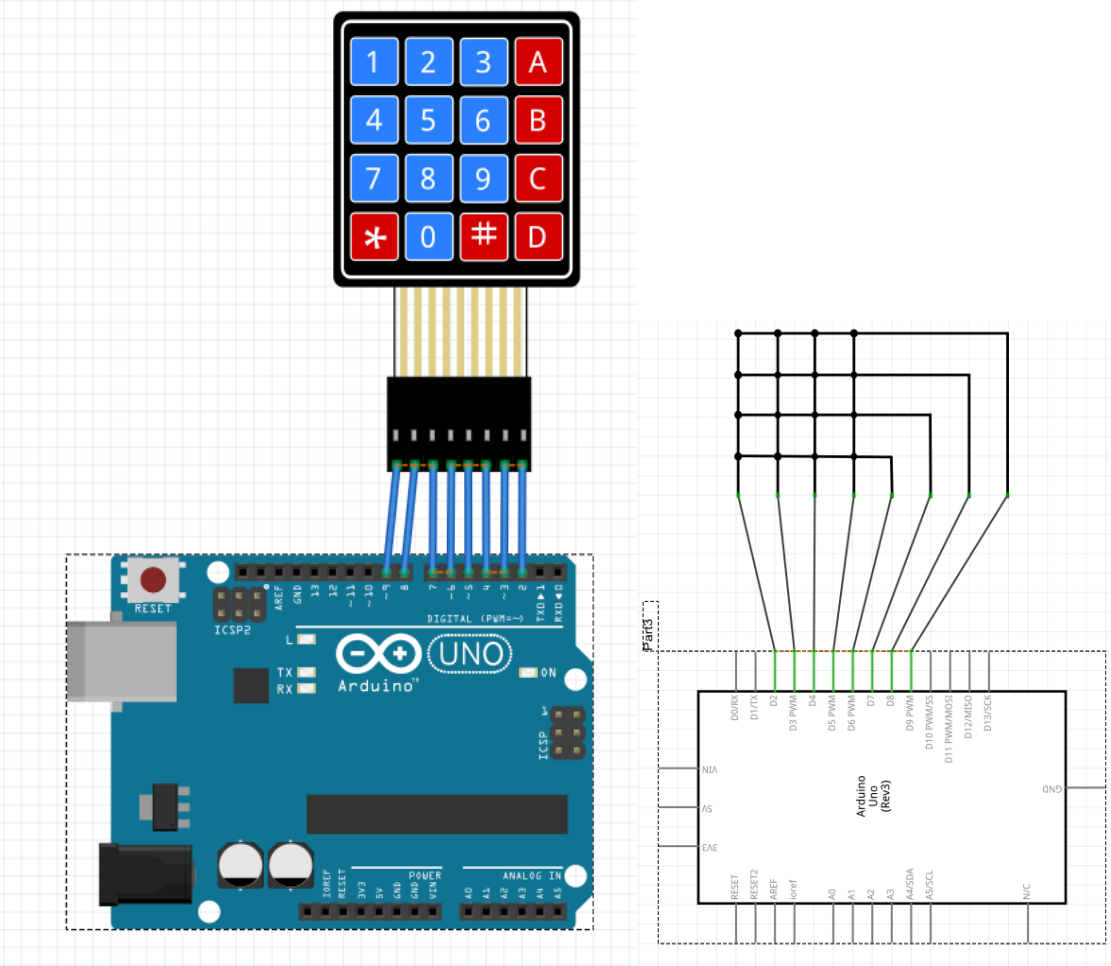
The pin layout for most membrane keypads will look like this:



Circuit 1

Circuit title: Serial monitor display the pressed key

Circuit description: Program will show us how to print each key press to the serial monitor.

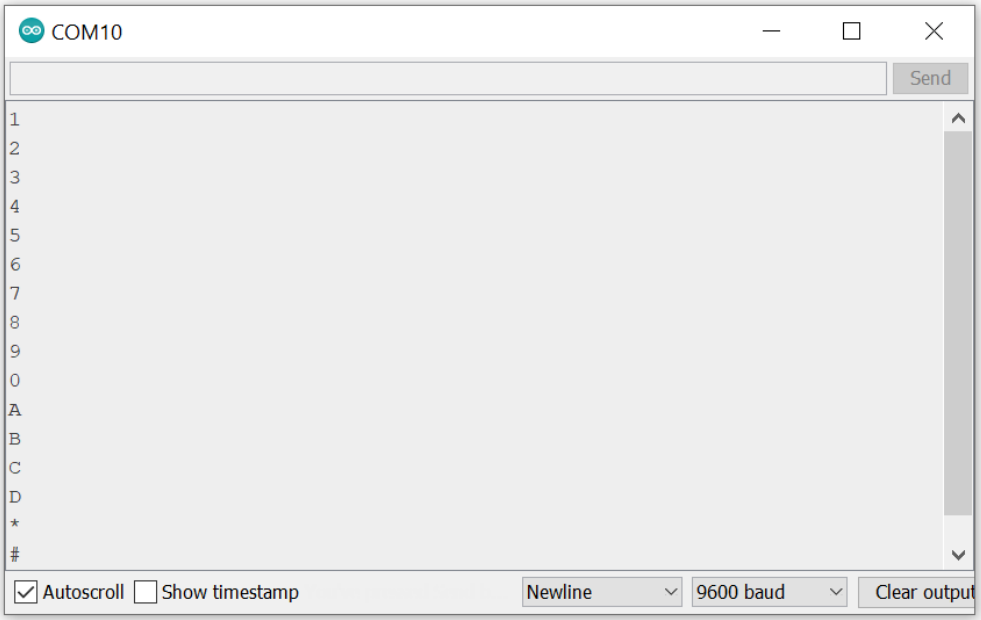


1-) Breadboard view 2-) Schematic view

/* "Serial monitor display the pressed key" */

```
#include <Keypad.h>
const byte ROWS = 4;
const byte COLS = 4;
char hexaKeys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
```

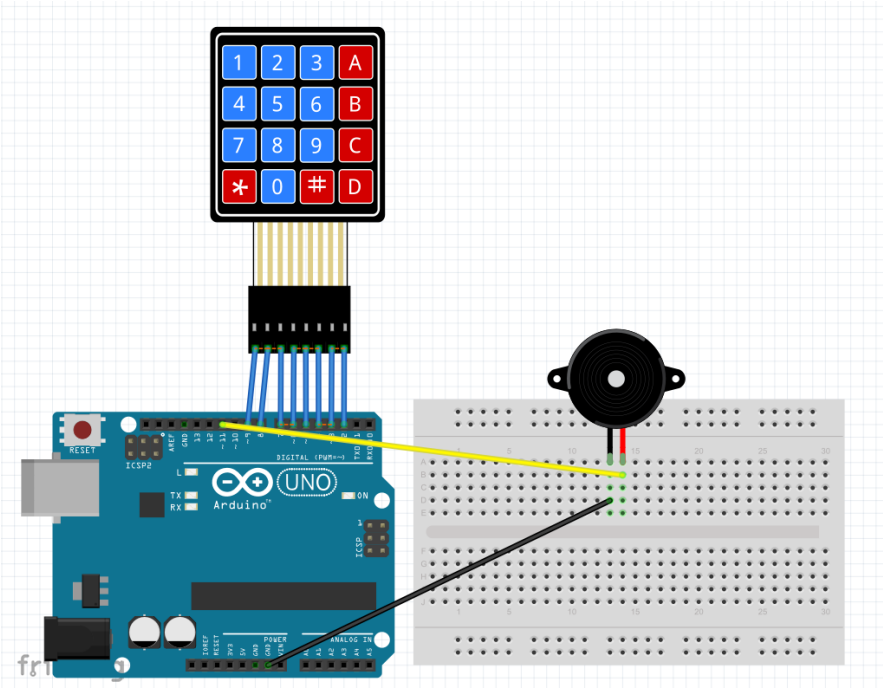
```
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS,
COLS);
void setup(){
  Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();
  if (customKey){
    Serial.println(customKey);
  }
}
```



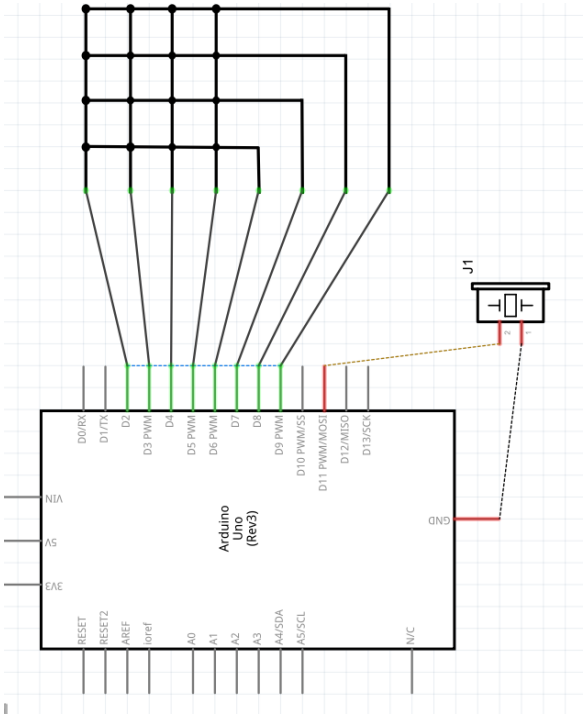
Circuit 2

Circuit title: Arduino Keypad Beep

Circuit description: When a key on the keypad is pressed, the piezo buzzer beeps



1-) Breadboard view



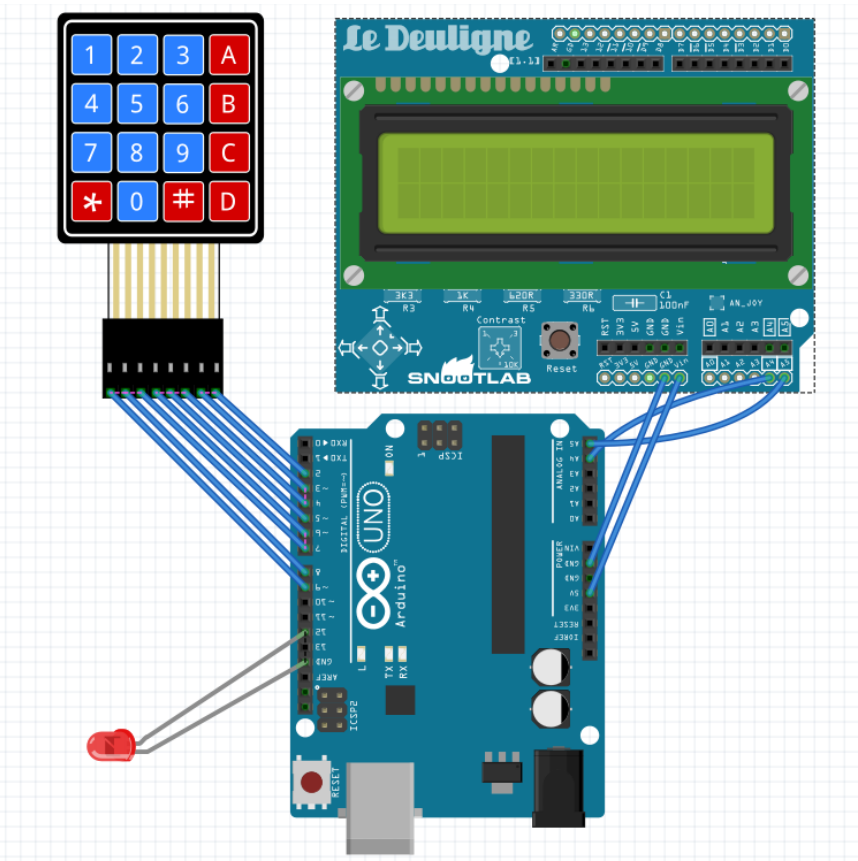
2-) Schematic view

```
/*    "Arduino Keypad Beep"   */

#include <Keypad.h>
#include <ezBuzzer.h>
const int BUZZER_PIN = 11;
const int ROW_NUM    = 4; // four rows
const int COLUMN_NUM = 4; // four columns
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
byte pin_rows[ROW_NUM] = {9, 8, 7, 6};      // connect to the row pinouts of the keypad
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; // connect to the column pinouts of the
keypad
Keypad keypad = Keypad(makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );
ezBuzzer buzzer(BUZZER_PIN); // create ezBuzzer object that attach to a pin;
void setup() {
  Serial.begin(9600);
}
void loop() {
  buzzer.loop(); // MUST call the buzzer.loop() function in loop()
  char key = keypad.getKey();
  if (key) {
    Serial.print(key); // prints key to serial monitor
    buzzer.beep(100);  // generates a 100ms beep
  }
}
```

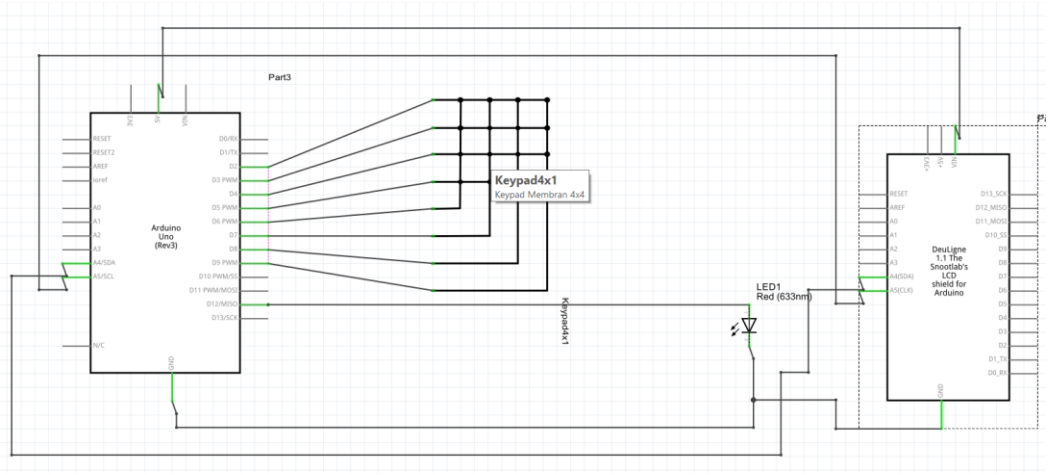
Circuit 3

Circuit title: Arduino Unlocking code

Circuit description: Program that sets a keypad on the Arduino. An LED will light up when you type the correct code



1-) Breadboard view



2-) Schematic view

```
/* Arduino Unlocking code */
#include <Keypad.h> //Libraries you can download them via Arduino IDE
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define Solenoid 12          //Actually the Gate of the transistor that controls the solenoid
                             //in my case I use a simple LED
#include <liquidcrystal_i2c.h></liquidcrystal_i2c.h></lcd.h></wire.h></keypad.h>
#define I2C_ADDR 0x27 // LCD i2c Adress and pins
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
LiquidCrystal_I2C
lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
const byte numRows= 4; //number of rows on the keypad
const byte numCols= 4; //number of columns on the keypad
int code = 1234;  //here is the code
int tot,i1,i2,i3,i4;
char c1,c2,c3,c4;
//keymap defines the key pressed according to the row and columns just as appears on the
keypad char keymap[numRows][numCols]=
{
{'1', '2', '3', 'A'},
{'4', '5', '6', 'B'},
{'7', '8', '9', 'C'},
{'*', '0', '#', 'D'}
};
//Code that shows the keypad connections to the arduino terminals
```

```
byte rowPins[numRows] = {9,8,7,6}; //Rows 0 to 3
byte colPins[numCols]= {5,4,3,2}; //Columns 0 to 3
//initializes an instance of the Keypad class
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
numCols);
void setup()
        {
         lcd.begin (16,2);
         lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
         lcd.setBacklight(HIGH);
         lcd.home ();
         lcd.print("ROMANIA DoorLock");
         lcd.setCursor(9, 1);
         lcd.print("Standby");
          pinMode(Solenoid,OUTPUT);
         delay(2000);
        }
void loop()
{
    char keypressed = myKeypad.getKey();  //The getKey fucntion keeps the program
runing, as long you didn't press "*" the whole thing bellow wouldn't be triggered
      if (keypressed == '*')          // and you can use the rest of you're code simply
         {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Enter Code");              //when the "*" key is pressed you can enter
the passcode
              keypressed = myKeypad.waitForKey();  // here all programs are stopped until
you enter the four digits then it gets compared to the code above
              if (keypressed != NO_KEY)
               {
                c1 = keypressed;
                lcd.setCursor(0, 1);
```

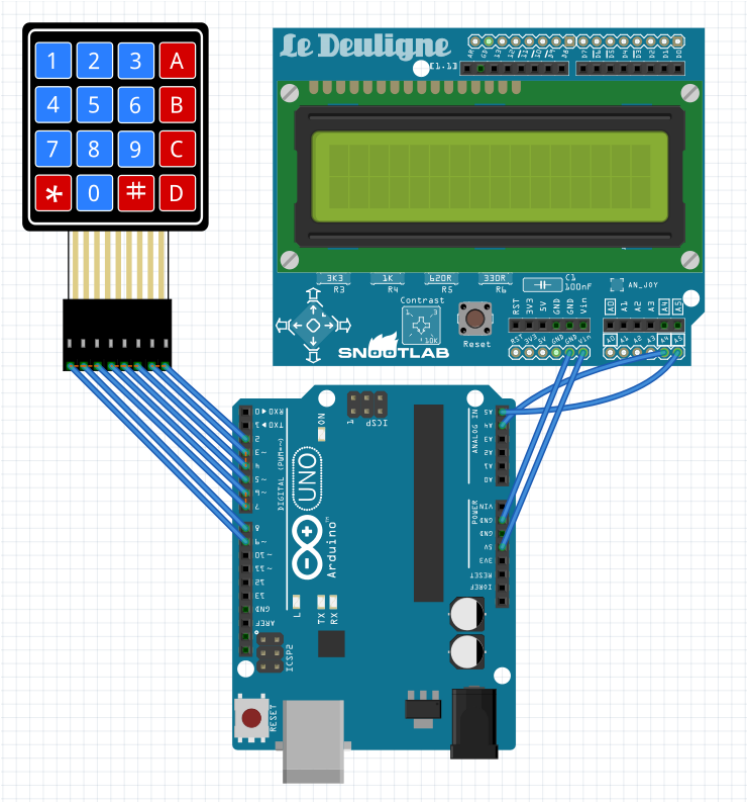
```
                lcd.print("*");
                }
              keypressed = myKeypad.waitForKey();
              if (keypressed != NO_KEY)
               {
                c2 = keypressed;
                lcd.setCursor(1, 1);
                lcd.print("*");
                }
               keypressed = myKeypad.waitForKey();
             if (keypressed != NO_KEY)
               {
                c3 = keypressed;
                lcd.setCursor(2, 1);
                lcd.print("*");
                }
                keypressed = myKeypad.waitForKey();
             if (keypressed != NO_KEY)
               {
                c4 = keypressed;
                lcd.setCursor(3, 1);
                lcd.print("*");
                }
               i1=(c1-48)*1000;       //the keys pressed are stored into chars I convert them to int then i did some multiplication to get the code as an int of xxxx
                i2=(c2-48)*100;
                i3=(c3-48)*10;
                i4=c4-48;
                tot=i1+i2+i3+i4;
            if (tot == code) //if the code is correct you trigger whatever you want here it just print a message on the screen
       {
        lcd.clear();
```

```
        lcd.setCursor(0, 0);
        lcd.print("Welcome");
        digitalWrite(Solenoid,HIGH);
 delay(3000);
 digitalWrite(Solenoid,LOW);
        lcd.setCursor(7, 1);
        lcd.print("ROMANIA DoorLock");

        delay(3000);
        lcd.clear();
        lcd.print("ROMANIA DoorLock");
        lcd.setCursor(9, 1);
        lcd.print("Standby");

       }
       else //if the code is wrong you get another thing
       {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("WRONG CODE");
        delay(3000);
        lcd.clear();
        lcd.print("ROMANIA DoorLock");
        lcd.setCursor(9, 1);
        lcd.print("Standby");
        }
          }
}
```

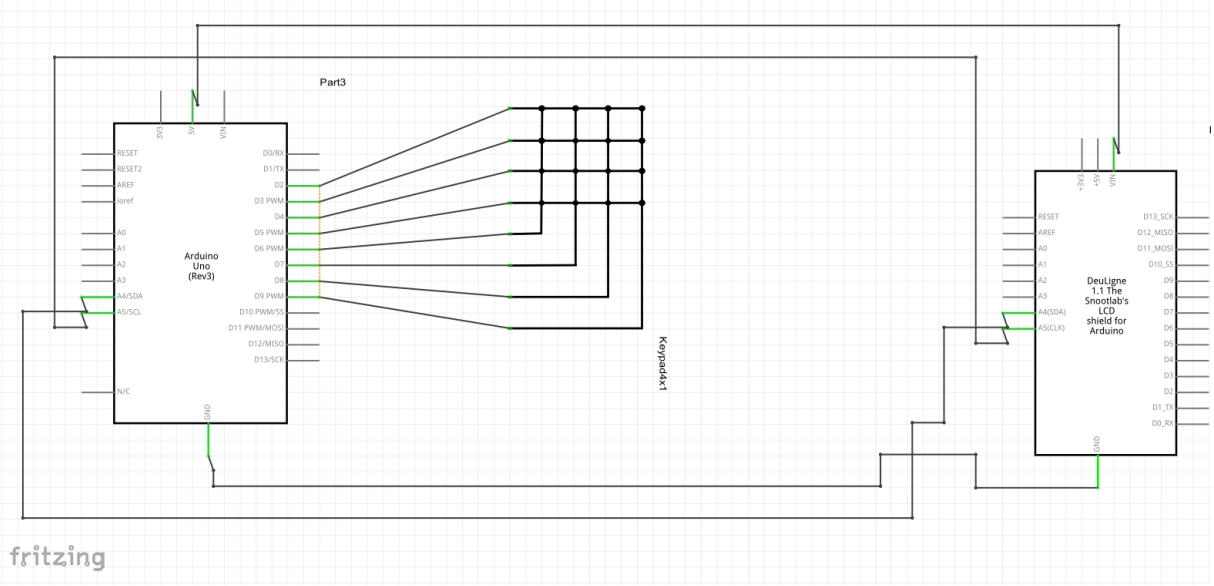
Circuit 4

Circuit title: Arduino calculator

Circuit description: Program does basic mathematical calculations



1-) Breadboard view



2-) Schematic view

```
/* Arduino calculator */
#include <Keypad.h>
#include <EEPROM.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR 0x27          //I2C adress
#define BACKLIGHT_PIN 3        // Declaring LCD Pins
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
const byte ROWS = 4; // Four rows
const byte COLS = 4; // Four columns
// Define the Keymap
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {9,8,7,6}; //Rows 0 to 3
byte colPins[COLS]= {5,4,3,2}; //Columns 0 to 3
LiquidCrystal_I2C
lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); //  Create
the Keypad
long Num1,Num2,Number;
 char key,action;
 boolean result = false;
 void setup()
```

```
        {
         lcd.begin (16,2);
         lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
           lcd.setBacklight(HIGH); //Lighting backlight
        lcd.print("Calculator Ready"); //Display a intro message
  lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
  lcd.print("A=+ B=- C=* D=/"); //Display a intro message
   delay(6000); //Wait for display to show info
    lcd.clear(); //Then clean it
                    //          for(i=0 ; i<sizeof(code);i++){       //When you upload the code the first
time keep it commented
//           EEPROM.get(i, code[i]);           //Upload the code and change it to store it in the
EEPROM
//            }       //Then uncomment this for loop and reupload the code (It's done only once)
        }
void loop() {
  key = kpd.getKey(); //storing pressed key value in a char
if (key!=NO_KEY)
DetectButtons();
if (result==true)
CalculateResult();
DisplayResult();
}
void DetectButtons()
{
     lcd.clear(); //Then clean it
    if (key=='*') //If cancel Button is pressed
    {Serial.println ("Button Cancel"); Number=Num1=Num2=0; result=false;}
        if (key == '1') //If Button 1 is pressed
    {Serial.println ("Button 1");
    if (Number==0)
    Number=1;
    else
```

```
Number = (Number*10) + 1; //Pressed twice
}
    if (key == '4') //If Button 4 is pressed
{Serial.println ("Button 4");
if (Number==0)
Number=4;
else
Number = (Number*10) + 4; //Pressed twice
}
    if (key == '7') //If Button 7 is pressed
{Serial.println ("Button 7");
if (Number==0)
Number=7;
else
Number = (Number*10) + 7; //Pressed twice
}
  if (key == '0')
{Serial.println ("Button 0"); //Button 0 is Pressed
if (Number==0)
Number=0;
else
Number = (Number*10) + 0; //Pressed twice
}
    if (key == '2') //Button 2 is Pressed
{Serial.println ("Button 2");
 if (Number==0)
Number=2;
else
Number = (Number*10) + 2; //Pressed twice
}
    if (key == '5')
{Serial.println ("Button 5");
 if (Number==0)
```

```
Number=5;
else
Number = (Number*10) + 5; //Pressed twice
}
    if (key == '8')
{Serial.println ("Button 8");
 if (Number==0)
Number=8;
else
Number = (Number*10) + 8; //Pressed twice
}
  if (key == '#')
{Serial.println ("Button Equal");
Num2=Number;
result = true;
}
    if (key == '3')
{Serial.println ("Button 3");
 if (Number==0)
Number=3;
else
Number = (Number*10) + 3; //Pressed twice
}
    if (key == '6')
{Serial.println ("Button 6");
if (Number==0)
Number=6;
else
Number = (Number*10) + 6; //Pressed twice
}
    if (key == '9')
{Serial.println ("Button 9");
if (Number==0)
```

```
   Number=9;
   else
   Number = (Number*10) + 9; //Pressed twice
   }
    if (key == 'A' || key == 'B' || key == 'C' || key == 'D') //Detecting Buttons on Column 4
  {
   Num1 = Number;
   Number =0;
   if (key == 'A')
   {Serial.println ("Addition"); action = '+';}
    if (key == 'B')
   {Serial.println ("Subtraction"); action = '-'; }
    if (key == 'C')
   {Serial.println ("Multiplication"); action = '*';}
    if (key == 'D')
   {Serial.println ("Division"); action = '/';}
   delay(100);
  }
}
void CalculateResult()
{
  if (action=='+')
    Number = Num1+Num2;
  if (action=='-')
    Number = Num1-Num2;
  if (action=='*')
    Number = Num1*Num2;
  if (action=='/')
    Number = Num1/Num2;
}
void DisplayResult()
{
  lcd.setCursor(0, 0);   // set the cursor to column 0, line 1
```

```
 lcd.print(Num1); lcd.print(action); lcd.print(Num2);
   if (result==true)
 {lcd.print(" ="); lcd.print(Number);} //Display the result
   lcd.setCursor(0, 1);   // set the cursor to column 0, line 1
 lcd.print(Number); //Display the result
}
```

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Project Title: “Teaching and Learning Arduinos in Vocational Training”

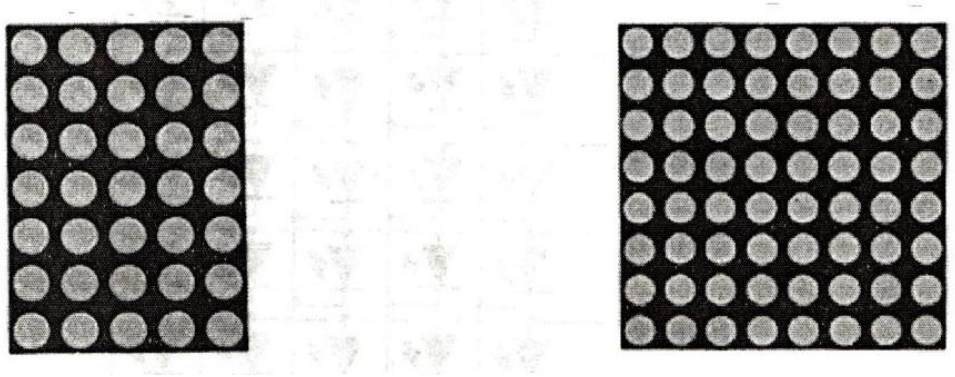
Project Acronym: “ ARDUinVET ”

Project No: “2020-1-TR01-KA202-093762”

Dot Matrix Module and Training Kit

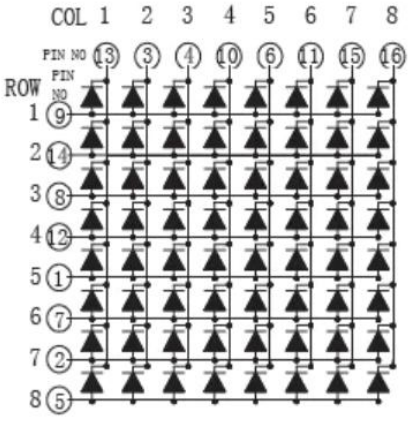
What is Dot Matrix Display?

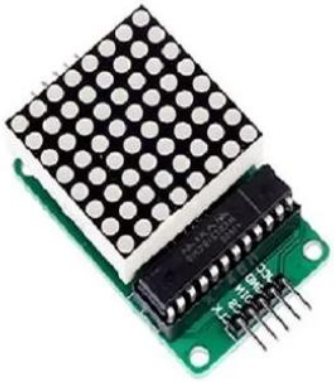
Dot Matrix display is a group of LEDs arranged in a specific system. A standard LED array is made up of 5x7 or 8x8 LEDs arranged in rows and columns as shown in the figure below. A 5*7 DOT matrix have 5 rows and 7 columns of LEDs and an 8*8 DOT matrix have 8 rows and 8 columns of LEDs which are connecting together.

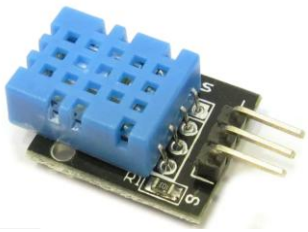


5*7 DOT Matrix Display

8*8 DOT Matrix Display

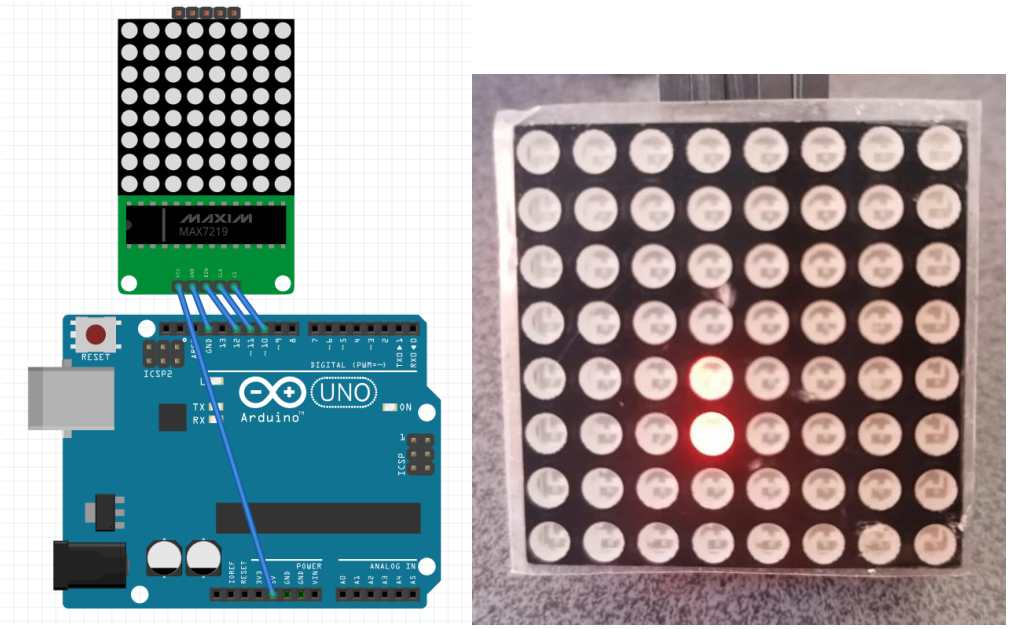
If we look at a piece of the 8x8 dot matrix, it contains 16 pins in which 8 pins used for rows and 8 for columns. That's mean in rows and column a total of 64 LEDs.We start from Pin # 1 to pin # 8. Pin number 1 is R5 (Row-5) and Pin number 8 is R3(Row-3) at the downside. At the upper side From Pin 9 (Row-1) to Pin 16 (column-1) located. But a newbie always confuses and starts from zero, because we know the picture/diagram. often we get from some source, also we have to sort out which one +VE and -VE. might be an expert can understand from common cathode/anode type.	

MAX7219 LED Dot matrix display is one of the popular ones available in the market and used by students, electronics hobbyists, and especially in industrial display applications. There are two types of modules generally available. These are the generic module. Each module consists of two units. One is the 8X8 LED dot matrix and the other is the MAX7219 IC. **MAX7219** is a common-cathode display driver IC with serial inputs and output pins. It has an adjustable current capability which can be set using only one external resistor. In addition to that, it has a four-wire serial interface that can be easily connected to all microprocessors. It can drive 64 individual LEDs connected at its output pins using only 4 wires by using Arduino. Furthermore, it can drive dot matrix displays, 7-Segments displays and bar graphs. On top of that, MAX7219 has a built-in BCD decoder that makes it easy to use with seven segment numeric displays. Additionally, it has an 8×8 static RAM that we can use to store numbers. It is one of the most popular display driver IC.	

	The DHT11 is a digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).

Circuit 1:

Circuit title: Display all LEDs one by one
Circuit description: The Dot Matrix display all LEDs one by one



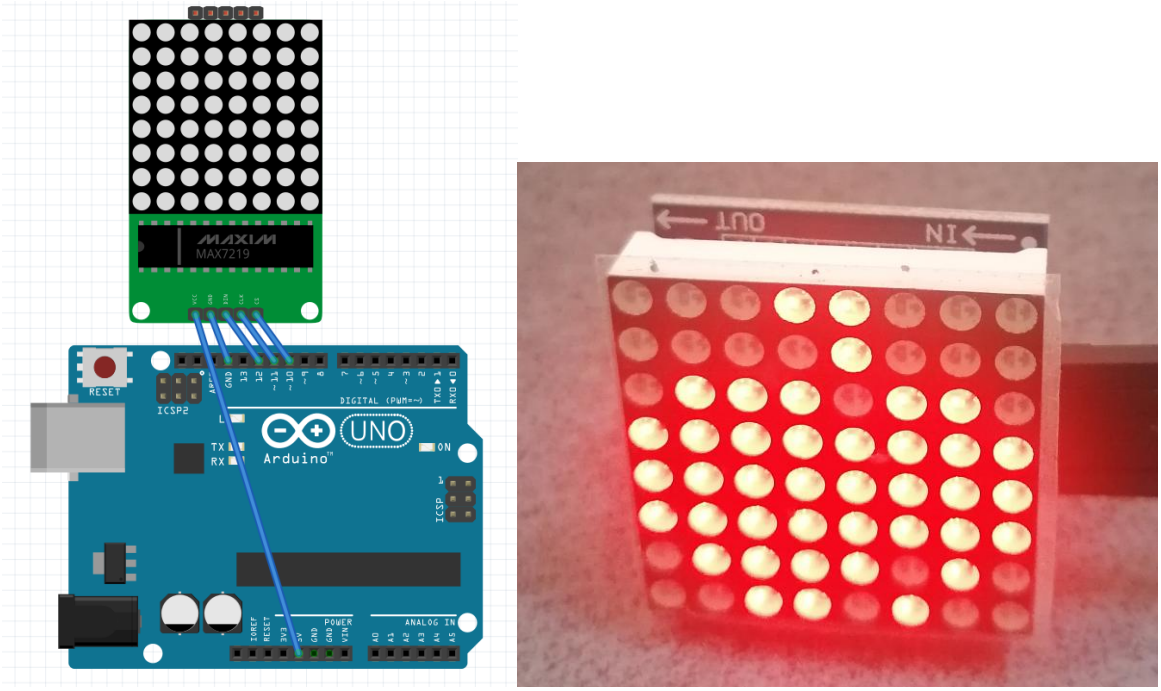
1-) Breadboard view

```
/*    Display all LEDs one by one  */

#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);}
void loop() {
  for(int j=0;j<8;j++){
     for(int i=0;i<8;i++){
       lc.setLed(0,j,i,true);
       delay(100);
       lc.setLed(0,j,i,false); } }}
```

Circuit 2:

Circuit title: Display the Apple icon

Circuit description: The Dot Matrix interface displays the Apple icon



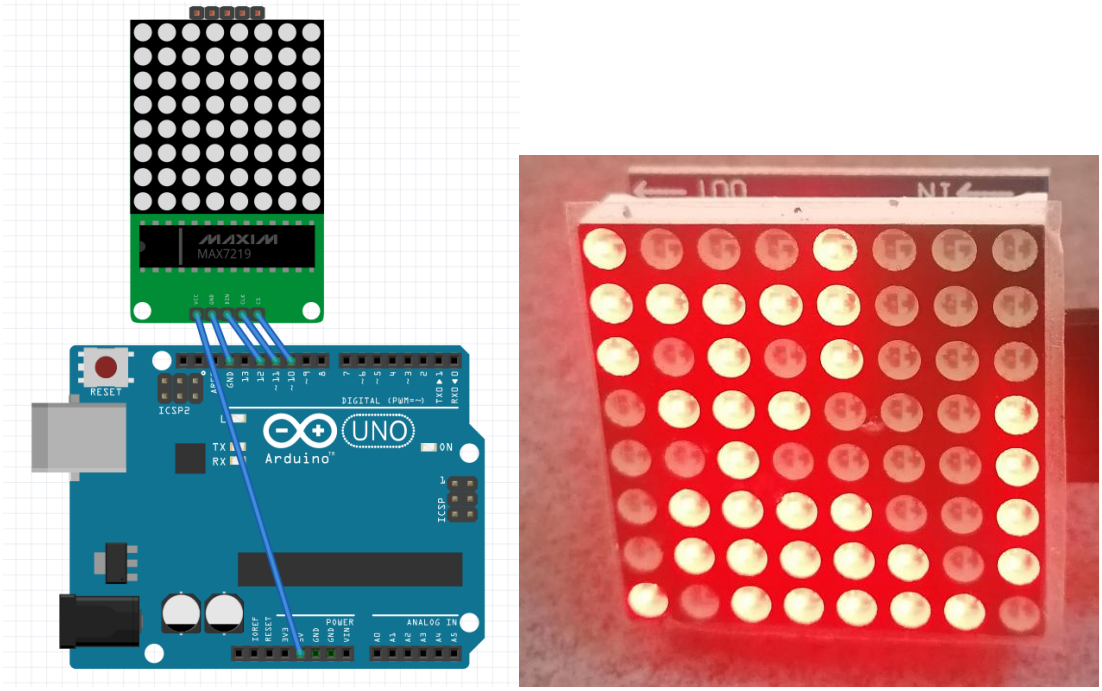
1-) Breadboard view

```
/*  Display the Apple icon*/
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
byte Apple
[8]={B00011000,B00001000,B01110110,B11111111,B11111111,B11111111,B01111010,B001
10100};
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);  }
  void loop(){
  for(int i=0;i<8;i++) lc.setRow(0,i,Apple[i]);   }
```

Circuit 3:

Circuit title: Display a cat icon

Circuit description: The Dot Matrix interface displays a cat icon



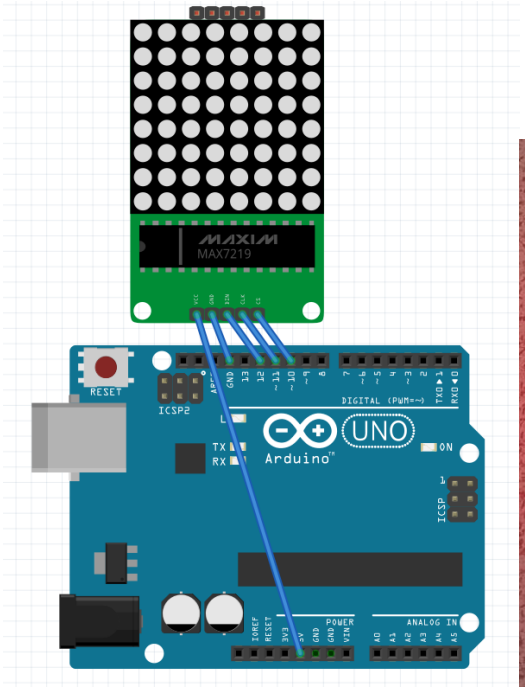
1-) Breadboard view

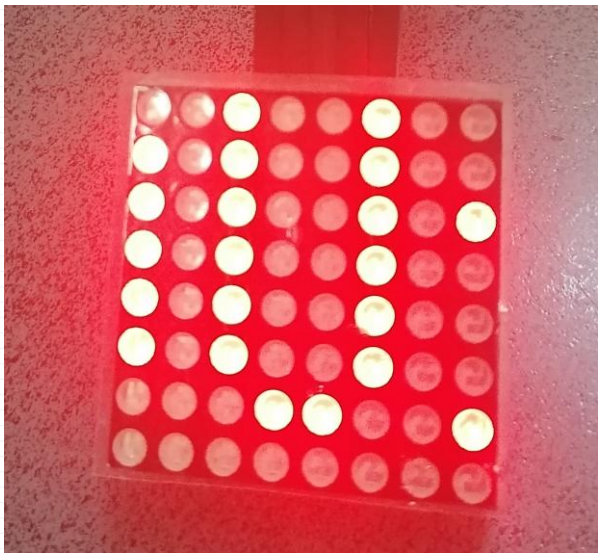
```
/*  Display a cat icon*/
#include <LedControl.h>
int DIN = 12;
int CS = 10;
int CLK = 11;
LedControl lc=LedControl(DIN, CLK, CS,0);
int Cat[8]
={B10001000,B11111000,B10101000,B01110001,B00100001,B01111001,B01111101,B10111
110 };
void setup() {
  lc.shutdown(0,false);
  lc.setIntensity(0,0);
  lc.clearDisplay(0);  }
  void loop(){
    for(int i=0;i<8;i++) lc.setRow(0,i,Cat[i]); }
```

Circuit 4:

Circuit title: Display flowing text ARDUINVET

Circuit description: The Dot Matrix interface displays flowing text ARDUINVET





1-) Breadboard view

```
/*  Display flowing text ARDUINVET*/
//D10=CS
//D11=CLK
//D12=DIN
#include <MaxMatrix.h> //include matrix library
#include <avr/pgmspace.h>
#include <stdlib.h>
PROGMEM const unsigned char CH[] = {
3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space
1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !
3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // "
5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #
4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $
5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %
5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &
```

```
1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // '
3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (
3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )
5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // *
5, 8, B00001000, B00001000, B00111110, B00001000, B00001000, // +
2, 8, B10110000, B01110000, B00000000, B00000000, B00000000, // ,
4, 8, B00001000, B00001000, B00001000, B00001000, B00000000, // -
2, 8, B01100000, B01100000, B00000000, B00000000, B00000000, // .
4, 8, B01100000, B00011000, B00000110, B00000001, B00000000, // /
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // 0
3, 8, B01000010, B01111111, B01000000, B00000000, B00000000, // 1
4, 8, B01100010, B01010001, B01001001, B01000110, B00000000, // 2
4, 8, B00100010, B01000001, B01001001, B00110110, B00000000, // 3
4, 8, B00011000, B00010100, B00010010, B01111111, B00000000, // 4
4, 8, B00100111, B01000101, B01000101, B00111001, B00000000, // 5
4, 8, B00111110, B01001001, B01001001, B00110000, B00000000, // 6
4, 8, B01100001, B00010001, B00001001, B00000111, B00000000, // 7
4, 8, B00110110, B01001001, B01001001, B00110110, B00000000, // 8
4, 8, B00000110, B01001001, B01001001, B00111110, B00000000, // 9
2, 8, B01010000, B00000000, B00000000, B00000000, B00000000, // :
2, 8, B10000000, B01010000, B00000000, B00000000, B00000000, // ;
3, 8, B00010000, B00101000, B01000100, B00000000, B00000000, // <
3, 8, B00010100, B00010100, B00010100, B00000000, B00000000, // =
3, 8, B01000100, B00101000, B00010000, B00000000, B00000000, // >
4, 8, B00000010, B01011001, B00001001, B00000110, B00000000, // ?
5, 8, B00111110, B01001001, B01010101, B01011101, B00001110, // @
4, 8, B01111110, B00010001, B00010001, B01111110, B00000000, // A
4, 8, B01111111, B01001001, B01001001, B00110110, B00000000, // B
4, 8, B00111110, B01000001, B01000001, B00100010, B00000000, // C
4, 8, B01111111, B01000001, B01000001, B00111110, B00000000, // D
4, 8, B01111111, B01001001, B01001001, B01000001, B00000000, // E
4, 8, B01111111, B00001001, B00001001, B00000001, B00000000, // F
4, 8, B00111110, B01000001, B01001001, B01111010, B00000000, // G
```

4, 8, B01111111, B00001000, B00001000, B01111111, B00000000, // H

3, 8, B01000001, B01111111, B01000001, B00000000, B00000000, // I

4, 8, B00110000, B01000000, B01000001, B00111111, B00000000, // J

4, 8, B01111111, B00001000, B00010100, B01100011, B00000000, // K

4, 8, B01111111, B01000000, B01000000, B01000000, B00000000, // L

5, 8, B01111111, B00000010, B00001100, B00000010, B01111111, // M

5, 8, B01111111, B00000100, B00001000, B00010000, B01111111, // N

4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // O

4, 8, B01111111, B00001001, B00001001, B00000110, B00000000, // P

4, 8, B00111110, B01000001, B01000001, B10111110, B00000000, // Q

4, 8, B01111111, B00001001, B00001001, B01110110, B00000000, // R

4, 8, B01000110, B01001001, B01001001, B00110010, B00000000, // S

5, 8, B00000001, B00000001, B01111111, B00000001, B00000001, // T

4, 8, B00111111, B01000000, B01000000, B00111111, B00000000, // U

5, 8, B00001111, B00110000, B01000000, B00110000, B00001111, // V

5, 8, B00111111, B01000000, B00111000, B01000000, B00111111, // W

5, 8, B01100011, B00010100, B00001000, B00010100, B01100011, // X

5, 8, B00000111, B00001000, B01110000, B00001000, B00000111, // Y

4, 8, B01100001, B01010001, B01001001, B01000111, B00000000, // Z

2, 8, B01111111, B01000001, B00000000, B00000000, B00000000, // [

4, 8, B00000001, B00000110, B00011000, B01100000, B00000000, // \ backsplash

2, 8, B01000001, B01111111, B00000000, B00000000, B00000000, //]

3, 8, B00000010, B00000001, B00000010, B00000000, B00000000, // hat

4, 8, B01000000, B01000000, B01000000, B01000000, B00000000, // _

2, 8, B00000001, B00000010, B00000000, B00000000, B00000000, // `

4, 8, B00100000, B01010100, B01010100, B01111000, B00000000, // a

4, 8, B01111111, B01000100, B01000100, B00111000, B00000000, // b

4, 8, B00111000, B01000100, B01000100, B00101000, B00000000, // c

4, 8, B00111000, B01000100, B01000100, B01111111, B00000000, // d

4, 8, B00111000, B01010100, B01010100, B00011000, B00000000, // e

3, 8, B00000100, B01111110, B00000101, B00000000, B00000000, // f

4, 8, B10011000, B10100100, B10100100, B01111000, B00000000, // g

4, 8, B01111111, B00000100, B00000100, B01111000, B00000000, // h

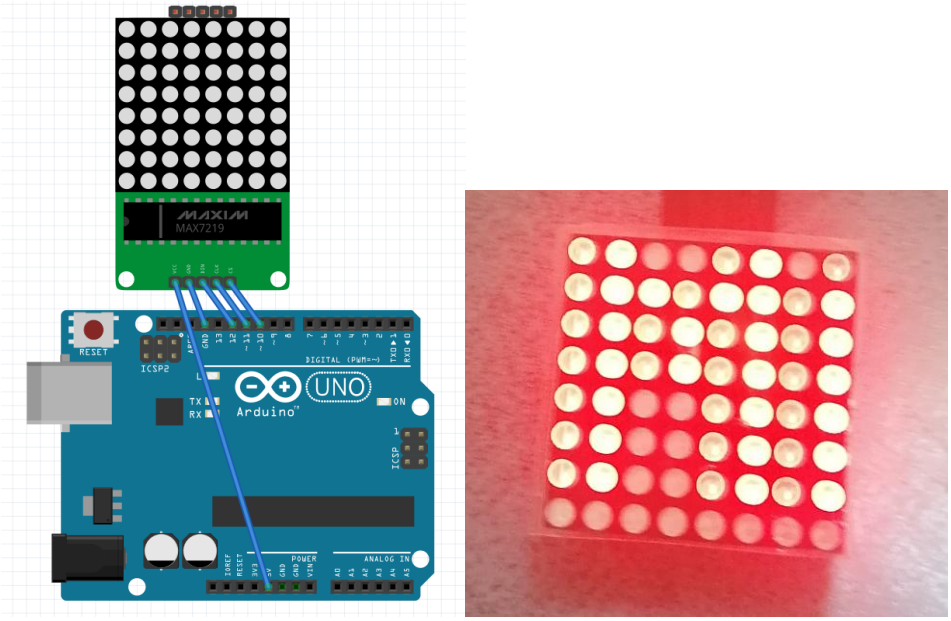
```
3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i
4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j
4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k
3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l
5, 8, B01111100, B00000100, B01111100, B00000100, B01111000, // m
4, 8, B01111100, B00000100, B00000100, B01111000, B00000000, // n
4, 8, B00111000, B01000100, B01000100, B00111000, B00000000, // o
4, 8, B11111100, B00100100, B00100100, B00011000, B00000000, // p
4, 8, B00011000, B00100100, B00100100, B11111100, B00000000, // q
4, 8, B01111100, B00001000, B00000100, B00000100, B00000000, // r
4, 8, B01001000, B01010100, B01010100, B00100100, B00000000, // s
3, 8, B00000100, B00111111, B01000100, B00000000, B00000000, // t
4, 8, B00111100, B01000000, B01000000, B01111100, B00000000, // u
5, 8, B00011100, B00100000, B01000000, B00100000, B00011100, // v
5, 8, B00111100, B01000000, B00111100, B01000000, B00111100, // w
5, 8, B01000100, B00101000, B00010000, B00101000, B01000100, // x
4, 8, B10011100, B10100000, B10100000, B01111100, B00000000, // y
3, 8, B01100100, B01010100, B01001100, B00000000, B00000000, // z
3, 8, B00001000, B00110110, B01000001, B00000000, B00000000, // {
1, 8, B01111111, B00000000, B00000000, B00000000, B00000000, // |
3, 8, B01000001, B00110110, B00001000, B00000000, B00000000, // }
4, 8, B00001000, B00000100, B00001000, B00000100, B00000000, // ~
};
int data = 12;    // DIN pin of MAX7219 module
int load = 10;    // CS pin of MAX7219 module
int clock = 11;  // CLK pin of MAX7219 module
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll use
MaxMatrix m(data, load, clock, maxInUse); // define module
byte buffer[10];
void setup(){
  m.init(); // module initialize
  m.setIntensity(2); // dot matix intensity 0-15
  Serial.begin(9600); // serial communication initialize
```

```
 Serial.println("ARDUinVET");  }
void loop(){
    printStringWithShift("ARDUinVET   ", 100);
  m.shiftLeft(false, true);  }
void printCharWithShift(char c, int shift_speed){
 if (c < 32) return;
 c -= 32;
 memcpy_P(buffer, CH + 7*c, 7);
 m.writeSprite(32, 0, buffer);
 m.setColumn(32 + buffer[0], 0);
 for (int i=0; i<buffer[0]+1; i++)
 {   delay(shift_speed);
   m.shiftLeft(false, false);  }
}
void printStringWithShift(char* s, int shift_speed){
 while (*s != 0){
   printCharWithShift(*s, shift_speed);
   s++;  }
}
void printString(char* s)
{  int col = 0;
 while (*s != 0)
 {   if (*s < 32) continue;
   char c = *s - 32;
   memcpy_P(buffer, CH + 7*c, 7);
   m.writeSprite(col, 0, buffer);
   m.setColumn(col + buffer[0], 0);
   col += buffer[0] + 1;
   s++;  }       }
```

Circuit 5:

Circuit title: Display every letter of alphabet

Circuit description: The Dot Matrix interface displays every letter of alphabet after 1 second



1-) Breadboard view

```
/*  Display every letter of alphabet*/
#include <MaxMatrix.h>//download from https://code.google.com/archive/p/arudino-maxmatrix-library/downloads
int DIN = 12;   // DIN pin of MAX7219 module
int CLK = 11;   // CLK pin of MAX7219 module
int CS = 10;    // CS pin of MAX7219 module
int maxInUse = 1;
MaxMatrix m(DIN, CS, CLK, maxInUse);
char A[] = {8, 8,
B00111000,B01000100,B01000100,B01000100,B01111100,B01000100,B01000100,B01000100};
char B[] = {8, 8,
B01111000,B01000100,B01000100,B01111000,B01000100,B01000100,B01111000,B00000000};
char c[] = {8, 8,
```

```
B00000000,B00111100,B01000000,B01000000,B01000000,B01000000,B00111100,B00000000};
char d[] = {8, 8,
B00000000,B01111000,B01000100,B01000100,B01000100,B01000100,B01111000,B00000000};
char e[] = {8, 8,
B00000000,B01111100,B01000000,B01000000,B01111100,B01000000,B01000000,B01111100};
char f[] = {8, 8,
B00000000,B01111100,B01000000,B01000000,B01111100,B01000000,B01000000,B01000000};
char g[] = {8, 8,
B00000000,B00111100,B01000000,B01000000,B01001110,B01000010,B00111110};
char h[] = {8, 8,
B00000000,B01000100,B01000100,B01111100,B01000100,B01000100,B01000100,B00000000};
char i[] = {8, 8,
B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B01111100,B00000000};
char j[] = {8, 8,
B00000000,B00000100,B00000100,B00000100,B00000100,B01000100,B00111000,B00000000};
char k[] = {8, 8,
B00000000,B00100100,B00101000,B00110000,B00101000,B00100100,B00100010,B00000000};
char l[] = {8, 8,
B00000000,B01000000,B01000000,B01000000,B01000000,B01000000,B01111100,B00000000};
char n[] = {8, 8,
B00000000,B01000010,B01100010,B01010010,B01001010,B01000110,B01000010,B00000000};
char o[] = {8, 8,
```

```
B00111100,B01000010,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100};
char p[] = {8, 8,
B00000000,B00111000,B00100100,B00100100,B00111000,B00100000,B00100000,B00000000};
char q[] = {8, 8,
B00111100,B01000010,B01000010,B01000010,B01001010,B01000110,B00111110,B00000001};
char r[] = {8, 8,
B01111000,B01000100,B01000100,B01111000,B01100000,B01010000,B01001000,B01000100};
char s[] = {8, 8,
B00111100,B01000000,B01000000,B00111000,B00000100,B00000100,B01111000,B00000000};
char t[] = {8, 8,
B00000000,B01111100,B00010000,B00010000,B00010000,B00010000,B00010000,B00000000};
char u[] = {8, 8,
B00000000,B01000010,B01000010,B01000010,B01000010,B01000010,B00111100,B00000000};
char v[] = {8, 8,
B00000000,B00100100,B00100100,B00100100,B00100100,B00100100,B00011000,B00000000};
char w[] = {8, 8,
B00000000,B01010100,B01010100,B01010100,B01010100,B01010100,B00111000,B00000000};
char x[] = {8, 8,
B00000000,B01000100,B00101000,B00010000,B00101000,B01000100,B00000000,B00000000};
char y[] = {8, 8,
B10000001,B01000010,B00100100,B00011000,B00011000,B00011000,B00011000,B00011000};
char z[] = {8, 8,
```

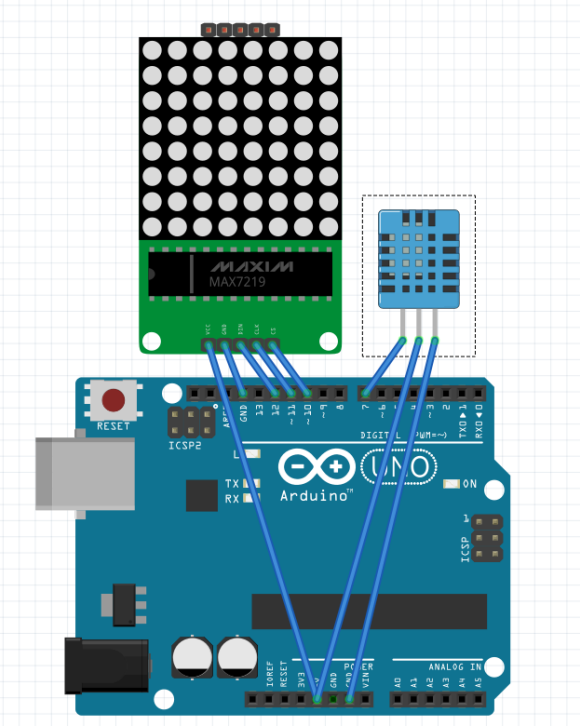
```
B00000000,B01111100,B00001000,B00010000,B00100000,B01111100,B00000000,B0000000
0};void setup() {
  m.init(); // MAX7219 initialization
  m.setIntensity(5); // initial led matrix intensity, 0-15     }
void loop() {
  // Seting the LEDs On or Off at x,y or row,column position
  m.setDot(6,2,true);
  delay(1000);
  m.setDot(6,3,true);
  delay(1000);
  m.clear(); // Clears the display
  for (int i=0; i<8; i++){
    m.setDot(i,i,true);
    delay(300);}
  m.clear();
  // Displaying the character at x,y (upper left corner of the character)
  m.writeSprite(0, 0, A);
  delay(1000);
  m.writeSprite(0, 0, B);
  delay(1000);
  m.writeSprite(0, 0, c);
  delay(1000);
  m.writeSprite(0, 0, d);
  delay(1000);
  m.writeSprite(0, 0, e);
  delay(1000);
  m.writeSprite(0, 0, f);
  delay(1000);
m.writeSprite(0, 0, g);
  delay(1000);
m.writeSprite(0, 0, h);
 delay(1000);
m.writeSprite(0, 0, i);
```

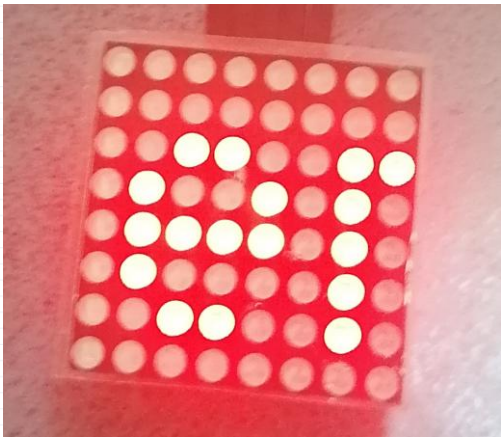
```
 delay(1000);
m.writeSprite(0, 0, j);
 delay(1000);
m.writeSprite(0, 0, k);
 delay(1000);
 m.writeSprite(0, 0, l);
 delay(1000);
m.writeSprite(0, 0, n);
 delay(1000);
m.writeSprite(0, 0, o);
 delay(1000);
  m.writeSprite(0, 0, p);
 delay(1000);
  m.writeSprite(0, 0, q);
 delay(1000);
  m.writeSprite(0, 0, r);
 delay(1000);
  m.writeSprite(0, 0, s);
 delay(1000);
  m.writeSprite(0, 0, t);
 delay(1000);
  m.writeSprite(0, 0, u);
 delay(1000);
  m.writeSprite(0, 0, v);
 delay(1000);
  m.writeSprite(0, 0, w);
 delay(1000);
  m.writeSprite(0, 0, x);
 delay(1000);
m.writeSprite(0, 0, y);
 delay(1000);
m.writeSprite(0, 0, z);
 delay(1000);}
```

Circuit 6:

Circuit title: Interface Dot Matrix with Arduino

Circuit description: The Dot Matrix interface displays the temperature measured using the temperature sensor DHT11





1-) Breadboard view

/* Interface Dot Matrix with Arduino */

```
//D7= temp
//D10=CS
//D11=CLK
//D12=DIN
#include <MaxMatrix.h> //include matrix library
#include <avr/pgmspace.h>
#include <stdlib.h>
#include "DHT.h" //include the temp sensor library
#define DHTPIN 7     // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11 temp&humid sensor
DHT dht(DHTPIN, DHTTYPE);
PROGMEM const unsigned char CH[] = {
```

```
3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space
1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !
3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // "
5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #
4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $
5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %
5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &
1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // '
3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (
3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )
5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // *
5, 8, B00001000, B00001000, B00111110, B00001000, B00001000, // +
2, 8, B10110000, B01110000, B00000000, B00000000, B00000000, // ,
4, 8, B00001000, B00001000, B00001000, B00001000, B00000000, // -
2, 8, B01100000, B01100000, B00000000, B00000000, B00000000, // .
4, 8, B01100000, B00011000, B00000110, B00000001, B00000000, // /
4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // 0
3, 8, B01000010, B01111111, B01000000, B00000000, B00000000, // 1
4, 8, B01100010, B01010001, B01001001, B01000110, B00000000, // 2
4, 8, B00100010, B01000001, B01001001, B00110110, B00000000, // 3
4, 8, B00011000, B00010100, B00010010, B01111111, B00000000, // 4
4, 8, B00100111, B01000101, B01000101, B00111001, B00000000, // 5
4, 8, B00111110, B01001001, B01001001, B00110000, B00000000, // 6
4, 8, B01100001, B00010001, B00001001, B00000111, B00000000, // 7
4, 8, B00110110, B01001001, B01001001, B00110110, B00000000, // 8
4, 8, B00000110, B01001001, B01001001, B00111110, B00000000, // 9
2, 8, B01010000, B00000000, B00000000, B00000000, B00000000, // :
2, 8, B10000000, B01010000, B00000000, B00000000, B00000000, // ;
3, 8, B00010000, B00101000, B01000100, B00000000, B00000000, // <
3, 8, B00010100, B00010100, B00010100, B00000000, B00000000, // =
3, 8, B01000100, B00101000, B00010000, B00000000, B00000000, // >
4, 8, B00000010, B01011001, B00001001, B00000110, B00000000, // ?
5, 8, B00111110, B01001001, B01010101, B01011101, B00001110, // @
```

4, 8, B01111110, B00010001, B00010001, B01111110, B00000000, // A

4, 8, B01111111, B01001001, B01001001, B00110110, B00000000, // B

4, 8, B00111110, B01000001, B01000001, B00100010, B00000000, // C

4, 8, B01111111, B01000001, B01000001, B00111110, B00000000, // D

4, 8, B01111111, B01001001, B01001001, B01000001, B00000000, // E

4, 8, B01111111, B00001001, B00001001, B00000001, B00000000, // F

4, 8, B00111110, B01000001, B01001001, B01111010, B00000000, // G

4, 8, B01111111, B00001000, B00001000, B01111111, B00000000, // H

3, 8, B01000001, B01111111, B01000001, B00000000, B00000000, // I

4, 8, B00110000, B01000000, B01000001, B00111111, B00000000, // J

4, 8, B01111111, B00001000, B00010100, B01100011, B00000000, // K

4, 8, B01111111, B01000000, B01000000, B01000000, B00000000, // L

5, 8, B01111111, B00000010, B00001100, B00000010, B01111111, // M

5, 8, B01111111, B00000100, B00001000, B00010000, B01111111, // N

4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // O

4, 8, B01111111, B00001001, B00001001, B00000110, B00000000, // P

4, 8, B00111110, B01000001, B01000001, B10111110, B00000000, // Q

4, 8, B01111111, B00001001, B00001001, B01110110, B00000000, // R

4, 8, B01000110, B01001001, B01001001, B00110010, B00000000, // S

5, 8, B00000001, B00000001, B01111111, B00000001, B00000001, // T

4, 8, B00111111, B01000000, B01000000, B00111111, B00000000, // U

5, 8, B00001111, B00110000, B01000000, B00110000, B00001111, // V

5, 8, B00111111, B01000000, B00111000, B01000000, B00111111, // W

5, 8, B01100011, B00010100, B00001000, B00010100, B01100011, // X

5, 8, B00000111, B00001000, B01110000, B00001000, B00000111, // Y

4, 8, B01100001, B01010001, B01001001, B01000111, B00000000, // Z

2, 8, B01111111, B01000001, B00000000, B00000000, B00000000, // [

4, 8, B00000001, B00000110, B00011000, B01100000, B00000000, // \ backsplash

2, 8, B01000001, B01111111, B00000000, B00000000, B00000000, //]

3, 8, B00000010, B00000001, B00000010, B00000000, B00000000, // hat

4, 8, B01000000, B01000000, B01000000, B01000000, B00000000, // _

2, 8, B00000001, B00000010, B00000000, B00000000, B00000000, // `

4, 8, B00100000, B01010100, B01010100, B01111000, B00000000, // a

```
4, 8, B01111111, B01000100, B01000100, B00111000, B00000000, // b
4, 8, B00111000, B01000100, B01000100, B00101000, B00000000, // c
4, 8, B00111000, B01000100, B01000100, B01111111, B00000000, // d
4, 8, B00111000, B01010100, B01010100, B00011000, B00000000, // e
3, 8, B00000100, B01111110, B00000101, B00000000, B00000000, // f
4, 8, B10011000, B10100100, B10100100, B01111000, B00000000, // g
4, 8, B01111111, B00000100, B00000100, B01111000, B00000000, // h
3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i
4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j
4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k
3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l
5, 8, B01111100, B00000100, B01111100, B00000100, B01111000, // m
4, 8, B01111100, B00000100, B00000100, B01111000, B00000000, // n
4, 8, B00111000, B01000100, B01000100, B00111000, B00000000, // o
4, 8, B11111100, B00100100, B00100100, B00011000, B00000000, // p
4, 8, B00011000, B00100100, B00100100, B11111100, B00000000, // q
4, 8, B01111100, B00001000, B00000100, B00000100, B00000000, // r
4, 8, B01001000, B01010100, B01010100, B00100100, B00000000, // s
3, 8, B00000100, B00111111, B01000100, B00000000, B00000000, // t
4, 8, B00111100, B01000000, B01000000, B01111100, B00000000, // u
5, 8, B00011100, B00100000, B01000000, B00100000, B00011100, // v
5, 8, B00111100, B01000000, B00111100, B01000000, B00111100, // w
5, 8, B01000100, B00101000, B00010000, B00101000, B01000100, // x
4, 8, B10011100, B10100000, B10100000, B01111100, B00000000, // y
3, 8, B01100100, B01010100, B01001100, B00000000, B00000000, // z
3, 8, B00001000, B00110110, B01000001, B00000000, B00000000, // {
1, 8, B01111111, B00000000, B00000000, B00000000, B00000000, // |
3, 8, B01000001, B00110110, B00001000, B00000000, B00000000, // }
4, 8, B00001000, B00000100, B00001000, B00000100, B00000000, // ~
};
int data = 12;    // DIN pin of MAX7219 module
int load = 10;    // CS pin of MAX7219 module
int clock = 11;  // CLK pin of MAX7219 module
```

```
int maxInUse = 1;    //change this variable to set how many MAX7219's you'll use
MaxMatrix m(data, load, clock, maxInUse); // define module
byte buffer[10];
void setup(){
  m.init(); // module initialize
  m.setIntensity(2); // dot matix intensity 0-15
  Serial.begin(9600); // serial communication initialize
  Serial.println("DHTxx test!");
  dht.begin();
}
void loop(){
  int t = dht.readTemperature();
  char temp[4];
  itoa(t,temp,10); //convert int to char!!!!
  Serial.println(temp);
   printStringWithShift("BRAILA ROMANIA", 100);
  printStringWithShift("  temp: ", 100);
  printStringWithShift(temp, 100);
  printStringWithShift(" C   ", 100);
  m.shiftLeft(false, true);
  }
void printCharWithShift(char c, int shift_speed){
  if (c < 32) return;
  c -= 32;
  memcpy_P(buffer, CH + 7*c, 7);
  m.writeSprite(32, 0, buffer);
  m.setColumn(32 + buffer[0], 0);
  for (int i=0; i<buffer[0]+1; i++)
  {
    delay(shift_speed);
    m.shiftLeft(false, false);
  }
}
```

```
void printStringWithShift(char* s, int shift_speed){
  while (*s != 0){
    printCharWithShift(*s, shift_speed);
    s++;
  }
}
void printString(char* s)
{
  int col = 0;
  while (*s != 0)
  {
    if (*s < 32) continue;
    char c = *s - 32;
    memcpy_P(buffer, CH + 7*c, 7);
    m.writeSprite(col, 0, buffer);
    m.setColumn(col + buffer[0], 0);
    col += buffer[0] + 1;
    s++;
  }
}
```

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Project Title: “Teaching and Learning Arduinos in Vocational Training”

Project Acronym: “ ARDUinVET ”

Project No: “2020-1-TR01-KA202-093762”

Arduino Motor Module and Training KIT

(DC, Step, Servo)

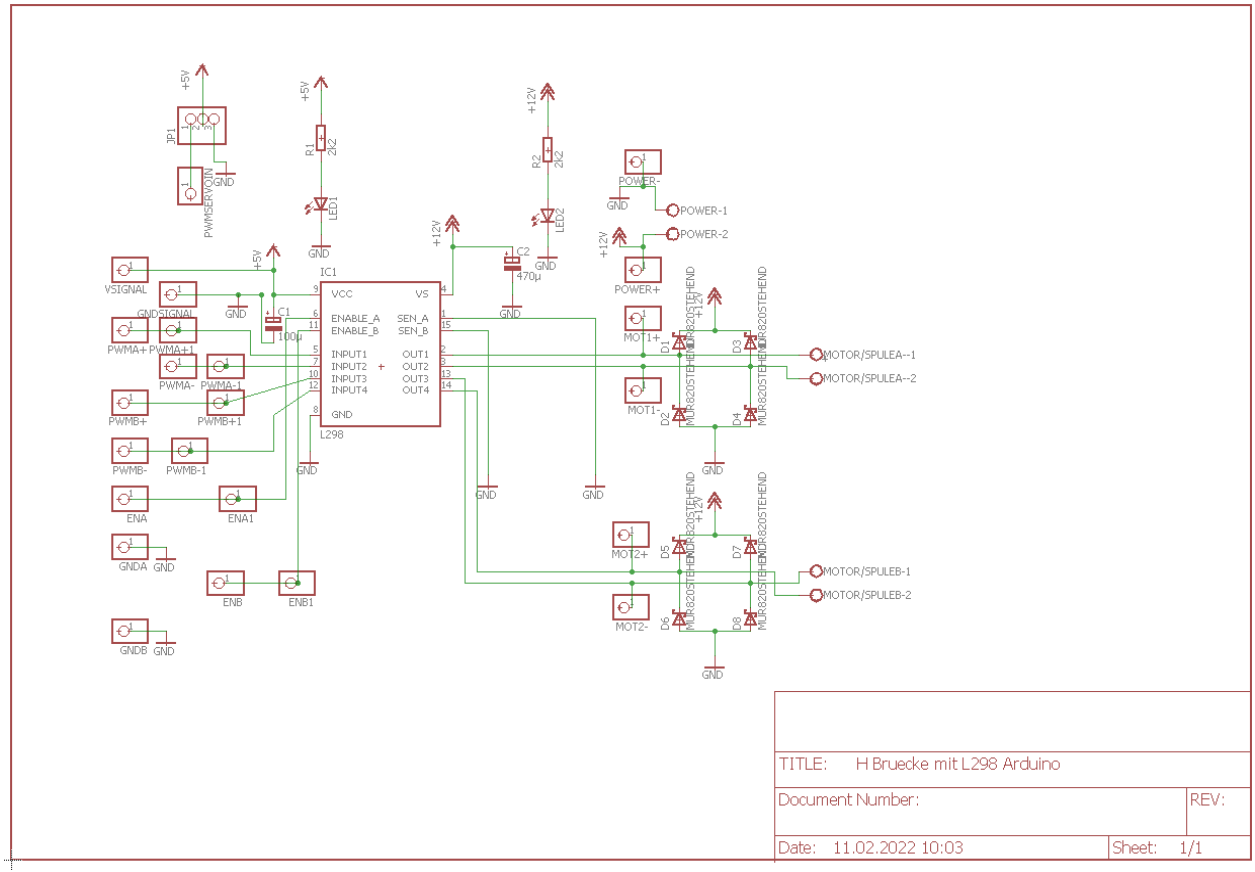
Arduino Motor Module and Training KIT

The motor module training kit is a didactic tool developed within the ARDUinVET project, with the aim of explaining in a simple way the functioning of different kinds of motors (Servo, Step and DC), controlled bu Arduino..

Using the programming platform pf Arduino, we intend to approach in a simple manner the programming of the Arduinos to put the different motors working, with the use of the hardware developed.

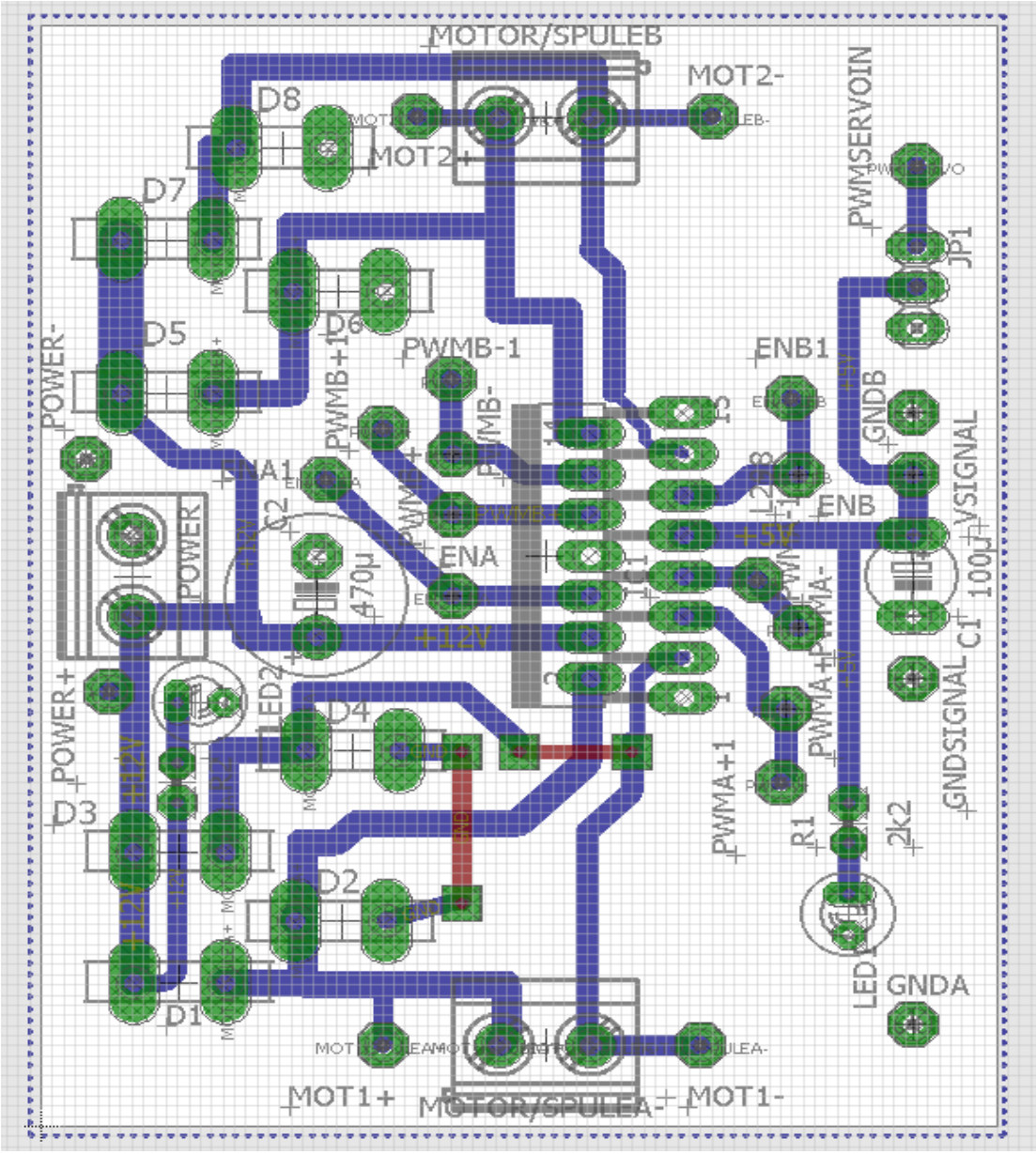
The goal is to show the students how to build and programme a printed circuit board (picture 1), as well as its applicability, making the motors work and execute different tasks, in this specific case.

Circuit Diagram Motor Shield



Picture: 1

Layout Motor Shield



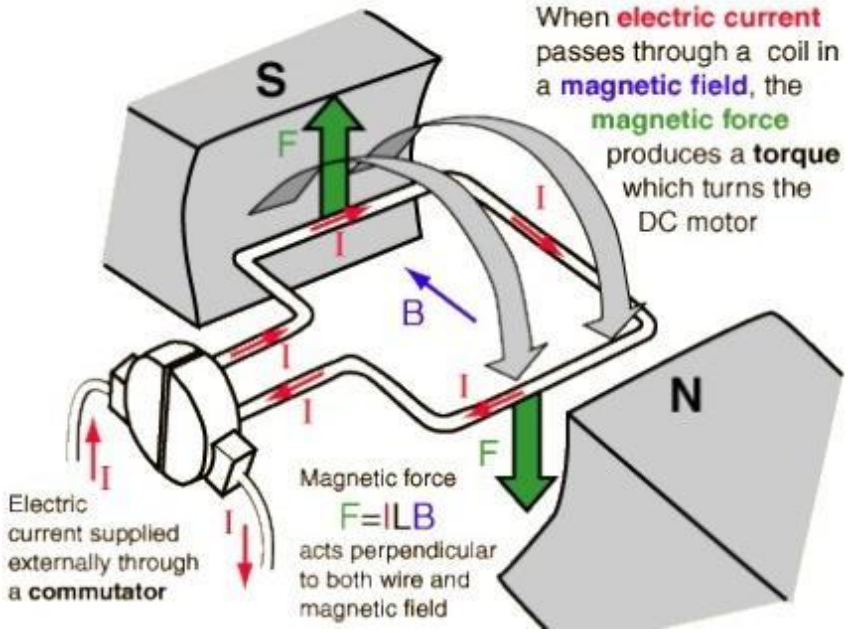
Picture: 2

Motors

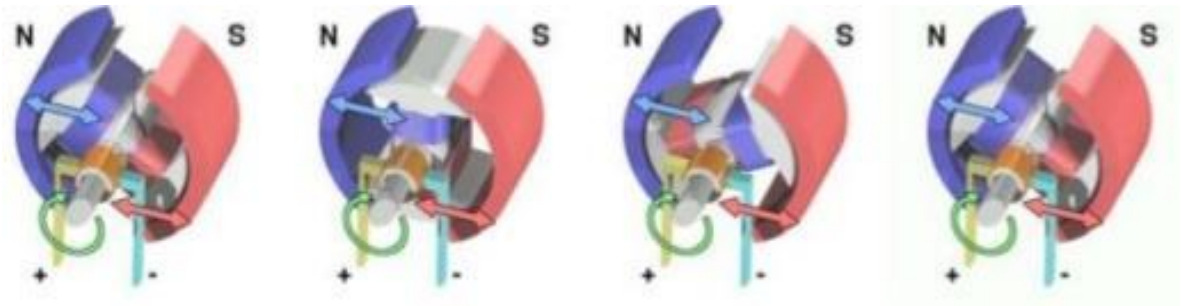
DC Motors

DC Motors, (Direct Current motors, picture 3) are electronic devices that operate through the attraction and repulsion forces generated by magnets.

Example:



Picture 3: DC motors functioning



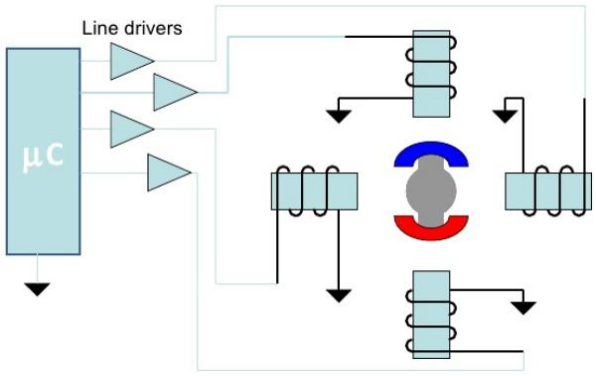
1 **2** **3** **4**

1. When the coil is fed, a magnetic field is generated around the rotor, causing a repulsion between the magnets, making the motor spin clockwise;
2. The rotor keeps spinning;

3. When the rotor becomes aligned horizontally the magnetic field is inverted, continuing the movement clockwise;
4. The process repeats itself continuously while the motor is being fed.

Step Motors

The step motor is an electric motor which moves according to a pulse received through a controller. The number of steps that the motor gives is exactly the same as the number of pulses received and the motor speed is the same as the pulses frequency. Therefore it is a simple device (brushless, switchless and with no encoders). They are motors used in several electronics and automation areas, such as robotics, Cartesian axes movement, among other.



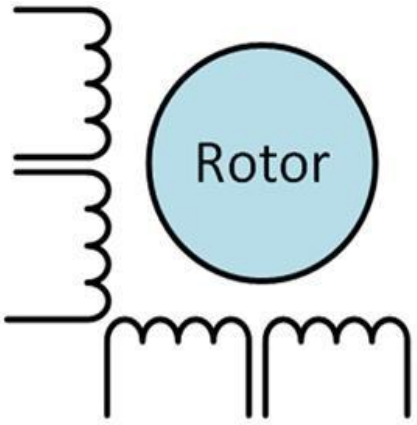
Picture 4: Simple diagram of a Step Motor

Picture 5: Step Motor

Examples of step motors

Unipolar motor

A unipolar motor has two coils by phase, one for each electric current direction. In this configuration a magnetic pole can be inverted without switching the electric current direction, the circuit of the switch can be done very simply for each coil.



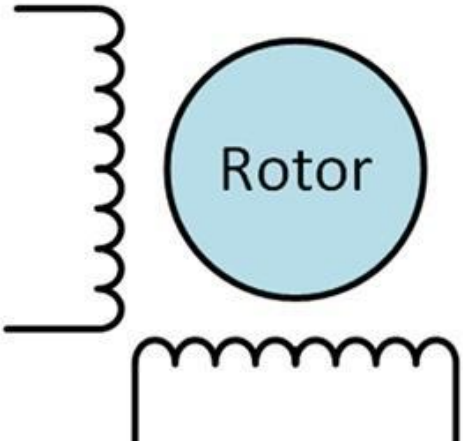
Picture 6: Motor unipolar

Cycle	C1	C2	C3	C4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Table 1: State table.

Motor Bipolar

The bipolar motors have a unique coil by phase. The electrical current in a coil needs to be inverted to invert a magnetic pole. So the electrical circuit is a little more complicated, requesting the need for an H bridge. There are two connections by phase and none is in common.

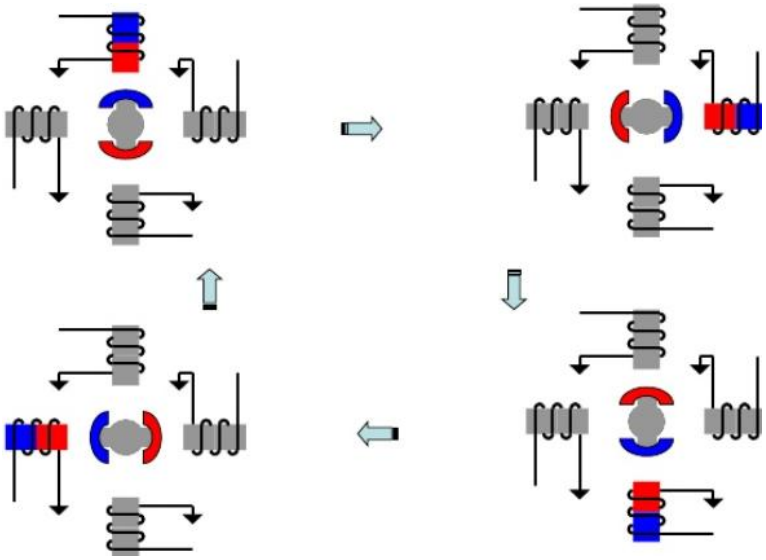


Picture 7: Motor Bipolar

Cycle	C1	C2	C3	C4
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0
D	0	0	0	1

Table 2: State Table

Step motor functioning



Picture 8: Step motor scheme

Servo Motors

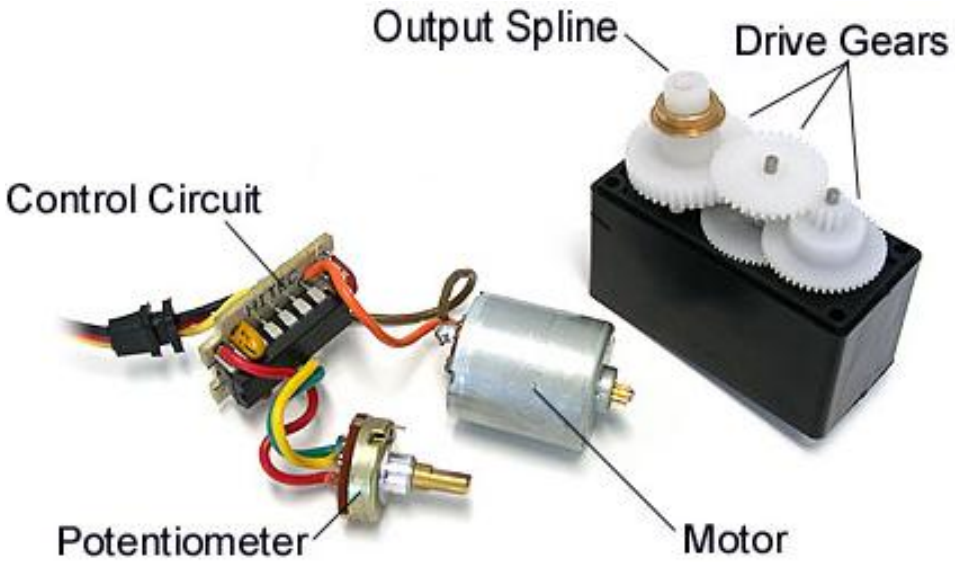
A Servo motor is an electromechanical device that through an electric signal, places the axis in an angular position. Normally this motor type is compact, allowing a precise position of its own axis. Currently the servo motors in used in robotics and modelling.

Picture 1: Servo Motor

Operation:

A servo motor can be divided into four parts:

- **Control Circuit** – Responsible for receiving the PWM signals and energy. Controls the position of the potentiometer and controls the motor de according to the PWM signal received.
- **Potentiometer** – Is connected to the servo output axis, positioning it.
- **Motor** – Moves the gear and the main axis of the servo.
- **Drive gears** – Reduces the motor rotation, decreasing the strength so that it applies a superior torque in the main axis. They move the potentiometer together with the axis.

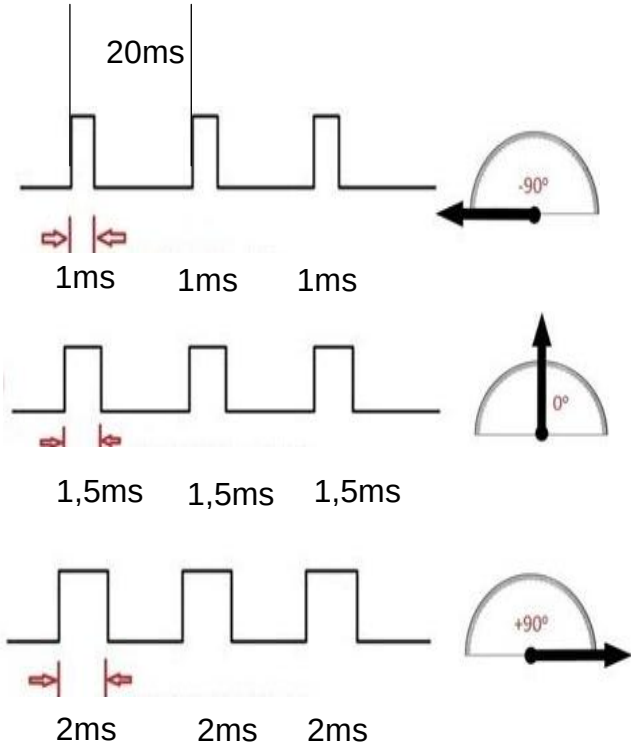


Picture 10: Servo motor parts

Servo Motor Positions

The control of a servo motor is obtained by an input signal that presents levels of TTL voltage that specifies its position. This signal format follows the PWM (Pulse Width Modulation) modulation as it is shown in picture 11. The codification in PWM is done through high level pulse in relation to the total period of oscillation, which in the servo motors case is 20ms.

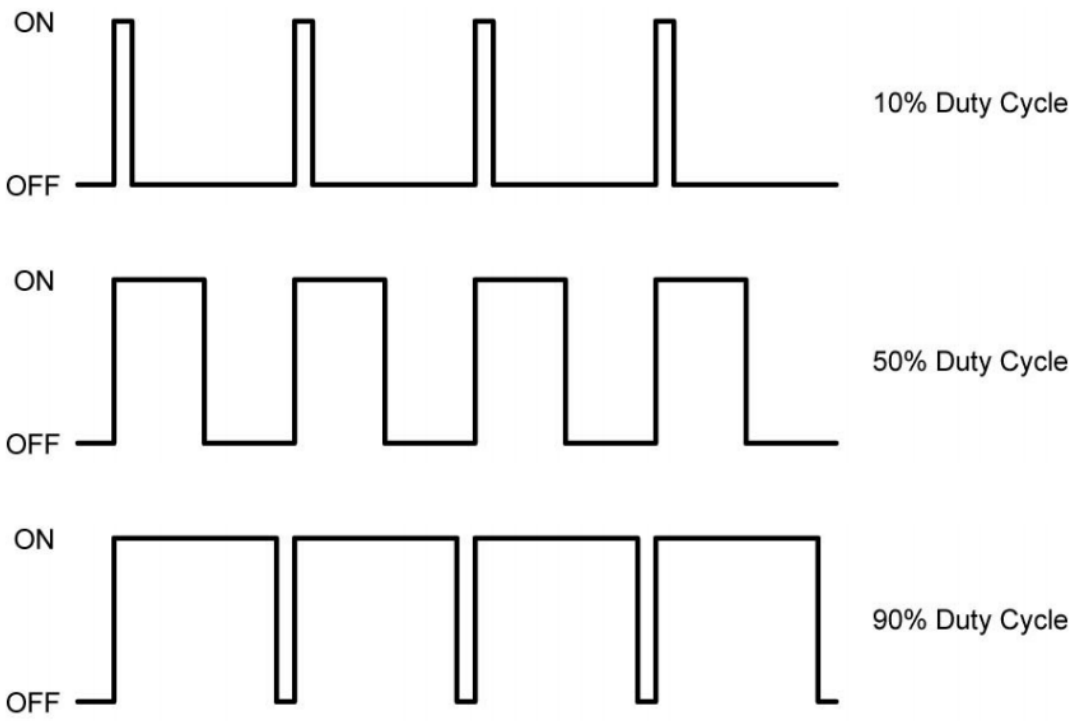
In this specific case, if in 20 milliseconds 1 millisecond is in high level the servo motor should be in the minimum position, as shown in picture 11.



Picture 11: PWM servo motor

PWM (Pulse Width Modulation)

The PWM can be implemented in several areas of electronics. One of its uses is in power supply, DC motors speed control, light control, servo motors control and several other applications. Through PWM we can control speed and power.



Picture 12: PWM

PWM operation

Considering a square wave, to obtain the correct operation of the PWM we must vary the pulse width of the wave. To calculate we need the period and the pulse width and its result is called duty-cycle, as it is defined by the equation:

$$Duty\ Cycle = 100\frac{Pulse\ Width}{Period}$$

Duty cycle: percentage value;

Pulse Width: sequence of time in which the signal is in high level;

Period: duration of a wave cycle.

NOTE: The following experiments will be done by using the Arduino Motor Kit, which the students will make by themselves.

Circuit title_1 (with DC Motors): "One DC-Motor simple"

Circuit Explanation: This program demonstrates on and off switching sequences of a dc motor

Program:

// One DC-Motor simple

```
/*
     Connection:
Arduino   | MotorShielt
PIN       | PIN
-------------------------
+5V       | VSIGNAL
GND       | GNDSIGNAL
4         | ENA
5         | PWMA+
6         | PWMA-
*/
//-------------------------------------------------------------
//  variables
//-------------------------------------------------------------
int ena = 4;    // Arduino pin 4 connected to ENA
int right = 5;  // Arduino pin 5 connected to PWMA+
int left = 6;   // Arduino pin 5 connected to PWMA-

//-------------------------------------------------------------
//  setup function
//-------------------------------------------------------------
void setup()
{
  pinMode(ena, OUTPUT);
  pinMode(right, OUTPUT);
  pinMode(left, OUTPUT);
  digitalWrite(ena, HIGH);  //Enable ENA
  delay(900);
}
//-------------------------------------------------------------
// loop Function
//-------------------------------------------------------------
void loop()
{
  analogWrite(right, 255);  // turn right with maximum speed
```

```
 delay(1000);
 analogWrite(right, 0);    // turn off
 delay(1000);
 analogWrite(left, 255);   // turn left with maximum speed
 delay(1000);
 analogWrite(left, 0);     // turn off
 delay(1000);
 analogWrite(right, 127);  // turn right with half speed
 delay(1000);
 analogWrite(right, 0);    // turn off
 delay(1000);
 analogWrite(left, 127);   // turn left with half speed
 delay(1000);
 analogWrite(left, 0);     // turn off
 delay(1000);
}
```

Circuit title_2: "One DC-Motor advance"

Circuit Explanation: This program control one dc motor. The control commands are issued via the serial monitor.

The control commands:

Write into the Serial Monitor	*Action*
stop	The motor stops
go right	Motor turn right
go left	Motor turn left
+	Increase speed
-	Decrease speed

Program:

// One DC-Motor advance

```
/*
     Connection:
Arduino   | MotorShielt
PIN       | PIN
-------------------------
+5V       | VSIGNAL
GND       | GNDSIGNAL
4         | ENA
5         | PWMA+
6         | PWMA-
```

```
*/
//-----------------------------------------------------------
//  variables
//-----------------------------------------------------------
int en_a = 4;              //ArduPin4 connectet to enable Pin
int right_a = 5;           //ArduPin5 connectet to PWMA+ Pin
int left_a = 6;            //ArduPin6 connectet to PWMA- Pin
String message = "";       //save the message of the seriel interface
int val = 80;              //defalt value for the speed
int del = 10;              //value in- and decrease by command "+" and "-"
int minimumVal = 70;       //minimum speed / value
int maximumVal = 250;      //maximum speed / value
bool flag_go_right = false; //becomes true when the command "go right" is issued
bool flag_go_left = false;  //becomes true when the command "go left" is issued

//-----------------------------------------------------------
//  setup function
//-----------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  pinMode(en_a, OUTPUT);
  pinMode(right_a, OUTPUT);
  pinMode(left_a, OUTPUT);
  digitalWrite(en_a, LOW);
  analogWrite(right_a, 0);
  analogWrite(left_a, 0);
  delay(900);
}
//-----------------------------------------------------------
// loop Function
//-----------------------------------------------------------
void loop()
{
  if(Serial.available()>0)
  {
    message = Serial.readString();
    Serial.println(" --> Incoming message: " + message ); //You see the incoming message

    if(message.equals("stop\n"))
    {
      digitalWrite(en_a, LOW);
            analogWrite(right_a, 0);
      analogWrite(left_a, 0);
      flag_go_right = false;
      flag_go_left = false;
      Serial.println(" <-- Outgoing message: stop \n");
    }
    else if(message.equals("go right\n"))  //check message if "go right"
```

```
  {
   digitalWrite(en_a, HIGH);          //set enable Pin of the Modul to HIGH
         analogWrite(right_a, val);         //write the value to the PWM Pin of turn right
   analogWrite(left_a, 0);            //write zero to the PWM Pin of turn left
   flag_go_left = false;              //set the flags
   flag_go_right = true;
   Serial.println(" <-- Outgoing message: go right\n");  //feedback
  }
  else if(message.equals("go left\n"))
  {
   digitalWrite(en_a, HIGH);
         analogWrite(right_a, 0);
   analogWrite(left_a, val);
   flag_go_right = false;
   flag_go_left = true;
   Serial.println(" <-- Outgoing message: go left\n");
  }
  else if(message.equals("+\n"))      //check message if "+"
  {
   if(val >= maximumVal)           // reach maximum value
   {
    Serial.println(" <-- Outgoing message: Maximum value! \n");
   }
   else if (val < maximumVal)
   {
    val = val + del;    //increase the value
    if(flag_go_right)   // check witch direction should be increase
    {
     analogWrite(right_a, val);
    }
    else if(flag_go_left)
    {
     analogWrite(left_a, val);
    }
    Serial.print(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
//feedback
   }
  }
  else if(message.equals("-\n"))
  {
   if(val <= minimumVal)                        // reach Minimum Speed
   {
    Serial.println(" <-- Outgoing message: Minimum value! \n");
   }
   else if (val > minimumVal)
   {
    val = val - del;
    if(flag_go_right)
    {
```

```
        analogWrite(right_a, val);
       }
       else if(flag_go_left)
       {
        analogWrite(left_a, val);
       }
       Serial.print(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
     }
    }
    else //generate message if the message is unknown
    {
     Serial.println("<<-- Incoming message " + message + "      IS UNKNOWN \n\n");
    }
   }
}
```

Circuit title_3: "Two DC-Motors advance"

Circuit Explanation: This program control two dc motors. The control commands are issued via the serial monitor.

The control command:

Write into the Serial Monitor	*Action*
stop	The motor stops
stop a	Stop Motor a
stop b	Stop Motor b
go right a	Motor a turn right
go left a	Motor a turn left
go right b	Motor b turn right
go left b	Motor b turn left
+	Increase speed
-	Decrease speed

Program:

// two DC-Motors advance

```
/*
 Connection:
Arduino  | MotorShielt
PIN      | PIN
-------------------------
+5V      | VSIGNAL
GND      | GND
4        | ENA
5        | PWMA+
6        | PWMA-
8        | ENB
9        | PWMB+
10       | PWMB-
*/

//-------------------------------------------------------------
// variables
//-------------------------------------------------------------
String message = "";

int en_a = 4;
int right_a = 5;
int left_a = 6;
bool flag_go_right_a = false;
bool flag_go_left_a = false;

int en_b = 8;
int right_b = 9;
int left_b = 10;
bool flag_go_right_b = false;
bool flag_go_left_b = false;

int val = 255;
int del = 10;
int minimumVal = 70;
int maximumVal = 250;

//-------------------------------------------------------------
// setup Function
//-------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
```

```
  pinMode(en_a, OUTPUT);
  pinMode(right_a, OUTPUT);
  pinMode(left_a, OUTPUT);

  pinMode(en_b, OUTPUT);
  pinMode(right_b, OUTPUT);
  pinMode(left_b, OUTPUT);

  digitalWrite(en_a, LOW);
  analogWrite(right_a, 0);
  analogWrite(left_a, 0);

  digitalWrite(en_b, LOW);
  analogWrite(right_b, 0);
  analogWrite(left_b, 0);

  delay(900);
}

//-------------------------------------------------------------
// loop Function
//-------------------------------------------------------------
void loop()
{
  if(Serial.available()>0)
  {
    message = Serial.readString();
    Serial.println(" --> Incoming message: " + message);

    if(message.equals("stop\n"))                    // both motors stop
    {
      digitalWrite(en_a, LOW);
            analogWrite(right_a, 0);
      analogWrite(left_a, 0);
      flag_go_right_a = false;
      flag_go_left_a = false;

      digitalWrite(en_b, LOW);
      analogWrite(right_b, 0);
      analogWrite(left_b, 0);
      flag_go_right_b = false;
      flag_go_left_b = false;

      Serial.println(" <-- Outgoing message: stop all\n");
    }
    else if(message.equals("stop a\n"))             // motor a stop
    {
      digitalWrite(en_a, LOW);
      analogWrite(right_a, 0);
```

```
  analogWrite(left_a, 0);
  flag_go_right_a = false;
  flag_go_left_a = false;
  Serial.println(" <-- Outgoing message: stop a\n");
 }
else if(message.equals("stop b\n"))                    // motor b stop
{
  digitalWrite(en_b, LOW);
  analogWrite(right_b, 0);
  analogWrite(left_b, 0);
  flag_go_right_b = false;
  flag_go_left_b = false;
  Serial.println(" <-- Outgoing message: stop b\n");
 }
else if(message.equals("go right a\n"))
{
  digitalWrite(en_a, HIGH);
        analogWrite(right_a, val);
  analogWrite(left_a, 0);
  flag_go_left_a = false;
  flag_go_right_a = true;
  Serial.println(" <-- Outgoing message: go right a\n");
}
else if(message.equals("go right b\n"))
{
  digitalWrite(en_b, HIGH);
  analogWrite(left_b, 0);
  analogWrite(right_b, val);
  flag_go_left_b = false;
  flag_go_right_b = true;
  Serial.println(" <-- Outgoing message: go right b\n");
}
else if(message.equals("go left a\n"))
{
  digitalWrite(en_a, HIGH);
        analogWrite(right_a, 0);
  analogWrite(left_a, val);
  flag_go_right_a = false;
  flag_go_left_a = true;
  Serial.println(" <-- Outgoing message: go left a\n");
}
else if(message.equals("go left b\n"))
{
  digitalWrite(en_b, HIGH);
  analogWrite(right_b, 0);
  analogWrite(left_b, val);
  flag_go_right_b = false;
  flag_go_left_b = true;
  Serial.println(" <-- Outgoing message: go left b \n");
```

```
  }
  else if(message.equals("+\n"))
  {
   if(val >= maximumVal)                    // reach maximum value
   {
    Serial.println(" <-- Outgoing message: Maximum value! \n");
   }
   else if (val < maximumVal)
   {
    val = val + del;
    if(flag_go_right_a) analogWrite(right_a, val);
    if(flag_go_right_b) analogWrite(right_b, val);
    if(flag_go_left_a) analogWrite(left_a, val);
    if(flag_go_left_b) analogWrite(left_b, val);
    Serial.print(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
   }
  }
  else if(message.equals("-\n"))
  {
   if(val <= minimumVal)                    // reach Minimum Value
   {
    Serial.println(" <-- Outgoing message: Minimum value! \n");
   }
   else if (val > minimumVal)
   {
    val = val - del;
    if(flag_go_right_a) analogWrite(right_a, val);
    if(flag_go_right_b) analogWrite(right_b, val);
    if(flag_go_left_a) analogWrite(left_a, val);
    if(flag_go_left_b) analogWrite(left_b, val);

    Serial.print(" <-- Outgoing message: value = "); Serial.println(val); Serial.println("");
   }
  }
  else
  {
   Serial.println("  <<-- Incoming message " + message + "      IS UNKNOWN \n\n");
  }
 }
}
```

Circuit title_4: "Simple step Motor program"

Circuit Explanation: " A step Motor turns clockwise and counterclockwise

Program:

```
// Simple step Motor program
/*
   Connection:
Arduino  | MotorShielt
PIN      | PIN
------------------------
+5V      | VSIGNAL
GND      | GNDSIGNAL
4        | ENA
5        | PWMA+
6        | PWMA-
8        | ENB
9        | PWMB+
10       | PWMB-
*/

//-------------------------------------------------------------
//  librarys
//-------------------------------------------------------------
#include <Stepper.h>
// libraty for the stepper
//-------------------------------------------------------------
//  constants
//-------------------------------------------------------------
#define STEPS 200
// Step Angle of used stepmotor = 1,8 degrees
// 360 / 1,8° = 200

//-------------------------------------------------------------
//  class
//-------------------------------------------------------------
Stepper Motor(STEPS, 5,6,9,10);
  //create a new instance of the Stepper class
  //Parameter:
  //  STEPS -> the number fo steps in one revolution of the Motor
  //  5,6,9,10 -> used Pins

//-------------------------------------------------------------
// variables
//-------------------------------------------------------------
int spe = 50;
```

```
//-------------------------------------------------------------
// setup function
//-------------------------------------------------------------
void setup()
{
  Serial.begin(9600);
  pinMode(4, INPUT);
  pinMode(8, INPUT);
  digitalWrite(4 ,HIGH);    //activate ENA Pin of the MotorShielt
  digitalWrite(8 ,HIGH);    //activate ENB Pin of the MotorShielt

  Motor.setSpeed(spe);  //Object Motor get the speed in "rotation per minute"
}

void loop()
{
  Motor.step(100);    //Turns the motor a specific number of steps,
                      //at a speed determined by the most recent call to setSpeed();
                      //in this case 100 steps clockwise;
  delay(200);

  Motor.step(-50);    //50 steps counterclockwise;
  delay(200);
}
```

Circuit title_5: "Advanced step Motor program"

Circuit Explanation: This program control a step-motors. The control commands are issued via the serial monitor.

Example: Send 100 over the Serial Monitor to the Arduino ☹The Step-Motor makes 100 Steps. If you send -100 ☹The Step-Motor makes 100 Steps in the ohter direction.

Program:

```
// Advanced step Motor program
/*
   Connection:
Arduino  | MotorShielt
PIN      | PIN
-------------------------
+5V      | VSIGNAL
GND      | GNDSIGNAL
4        | ENA
5        | PWMA+
6        | PWMA-
8        | ENB
9        | PWMB+
10       | PWMB-
*/

//-------------------------------------------------------------
//  librarys
//-------------------------------------------------------------
#include <Stepper.h>
//-------------------------------------------------------------
//  constants
//-------------------------------------------------------------
#define STEPS 200
//-------------------------------------------------------------
//  class
//-------------------------------------------------------------
Stepper Motor(STEPS, 5,6,9,10);
//-------------------------------------------------------------
// variables
//-------------------------------------------------------------

int spe = 100;
String message = "";

//-------------------------------------------------------------
// setup function
```

```
//-------------------------------------------------------------

void setup()
{
  Serial.begin(9600);
  pinMode(4, INPUT);
  pinMode(8, INPUT);
  digitalWrite(4 ,HIGH);
  digitalWrite(8 ,HIGH);
  Motor.setSpeed(spe);
}

void loop()
{
  if(Serial.available()>0)
  {
    message = Serial.readString();
    Serial.println("\n\n --> Incoming message: " + message );

    if(message.toInt())
    {
      Serial.print(" <-- Outgoing message:  Steps -> " + message);
      Motor.step(message.toInt());
    }
    else
    {
      Serial.println("  <<-- Incoming message " + message + "  !!! -->  is not a number <-- !!!
\n\n");
    }
  }

}
```

Circuit title_6 (with Servo Motor): ” Simple servo motor program ”

Circuit Explanation: In this program a servo motor is controlled by a pwm signal. The for loop is used

// Simple servo Motor programm

```
/*
   Connection:
Arduino  | MotorShielt
PIN      | PIN
-------------------------
+5V      | VSIGNAL
GND      | GNDSIGNAL
3        | PWMSERVOIN
*/
//-----------------------------------------------------------------
// variables
//-----------------------------------------------------------------
int del = 100;  // time for delay

//-----------------------------------------------------------------
// setup function
//-----------------------------------------------------------------
void setup()
{
 Serial.begin(9600);
 pinMode(3, OUTPUT);
}

void loop()
{
 for(int i = 0; i<255; i++)
   {
    analogWrite(3,i);
    Serial.println(i);
    delay(del);
  }

  delay(1000);
}
```

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Project Title: "Teaching and Learning Arduinos in Vocational Training"

Project Acronym: " ARDUinVET "

Project No: "2020-1-TR01-KA202-093762"

Sensors Module and Training Kit

Sensors

A sensor is a device, module, or subsystem of Electronics, whose purpose is to create an interface between a circuit and its environment. It is the system which receives information from the environment, the psysical world, and sends this information as a value to the circuit. Actually it detects events or changes which happen in the environment. To detect these events or changes, the sensor must measure certain values in a physical scale. It measures a property, such as pressure, position, temperature, or acceleration, and responds with feedback. Then it converts these values in an electric –usually- signal.

So, a sensor is always used with other electronics. Sensors can be found in many everyday objects, such as touch - or movement sensitive devices, devices operated by light or sound etc. Tthe use of sensors have expanded beyond the traditional fields of temperature, pressure or flow measurement, e.g. GPS sensors. Analog sensors are still widely used. Applications include manufacturing and machinery, airplanes and aerospace, cars, medicine, robotics and many other aspects of our day-to-day life.

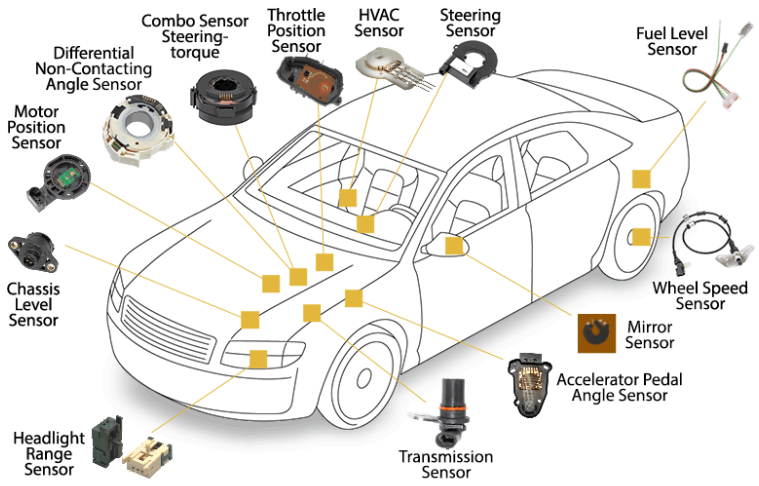


Figure 1. Automobile Sensors

Sensor Characteristics

A good sensor obeys the following rules:

- it must be sensitive to the measured property
- it must insensitive to any other property likely to be encountered in its application
- it must not influence the measured property.

The main basic characteristics of the sensors are:

- Linearity. The sensor has a property or feature, whose value changes. Whenthe physical quantity that measures is also changed. It is desirable that its variations in the physical quantity measurement cause appreciable changes in the property of the sensor. This property is called linearity and is of main importance.
- Accuracy. The proximity of the output value to the input value.
- Error: The difference between the measured value and the actual value.
- Tolerance: The maximum error the sensor can generate.
- Full - Scale Input (FSI): Specifies in which frames of the measured physical size can the sensor be used
- Full - Scale Output (FSO): Sets the values that voltage or current can receive at the output of the sensor
- Sensitivity: It expresses how high the output signal outputs the sensor for each unit of the measured physical size
- Resolution: Expresses the smallest change in the physical size that the sensor can detect and accordingly alter its output value
- Hysteresis: A hysteresis error causes that the output value varies depending on previous input values. If a sensor's output is different, depending on whether a specific input value was reached by increasing vs. decreasing the input, then the sensor has a hysteresis error
- Delay: It is the delay of the change of output value after an input change.
- Dead Zone: The maximum amount of input value change that does not affect the output value.

Sensor Categories

There are several ways of categorizing sensors, some of which are listed below.

The first categorizes the sensors according to the form of the output value, dividing the analog and digital sensors.

The second categorization is about what can a sensor measure, with a more significant distinction between natural and chemical sensors. Natural sensors control physical sizes such as location, mass, current, time and their relative sizes, while chemical sensors control the presence of different gases in a particular atmosphere.

One other way relates to materials in the physical properties of which sensor function, with main categories sensors with conductive, semiconductor, dielectric, magnetic and superconductive materials.

Finally, one more classification method refers to the field ofuse of the sensor, with major categories such as industrial, medical, military, environmental sensors, as well as sensors for transport and automation applications.

Figure 2: Various Sensors

Basic Sensors

Here are some basic sensors for common applications:

Photo Resistor LDR: The resistor is a variable resistor whose value changes depending on the light that falls on it.

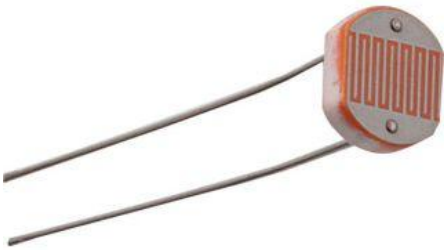


Figure 3: Photo Resistor

Digital Luminosity / Lux / Light sensor: The TSL2561 luminosity sensor is an advanced digital light sensor, ideal for use in a wide range of light situations. This sensor is more precise, allowing for exact lux calculations and can be configured for different gain/timing ranges to detect light ranges from up to 0.1 - 40,000+ Lux on the fly. It contains both infrared and full spectrum diodes! That means you can separately measure infrared, full-spectrum or human-visible light.

Figure 3: Digital Luminosity / Lux / Light sensor

Temperature sensor (with analogue output): The temperature range it measures is usually -40 ° C to + 100 ° C with an accuracy of ± 1 ° C.

Figure 4: Temperature sensor

Humidity and Temperature Sensor: highly accurate, digital temperature and humidity sensor. It features a 0-100% RH measurement range with a temperature accuracy of +/- 0.3°C @ 25°C.

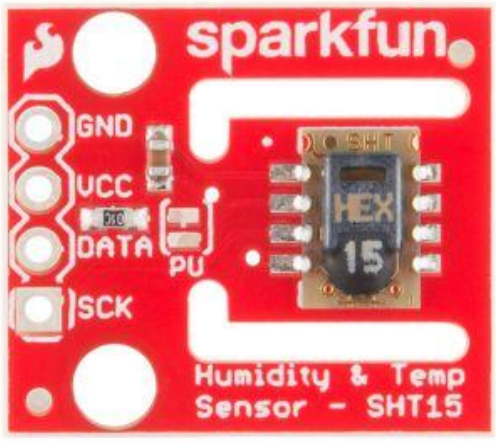


Figure 5: Humidity and Temperature sensor

Infrared Sensor: Used for distance calculation. The distance you can calculate is from 2cm. up to 400cm. with an accuracy of one centimeter.

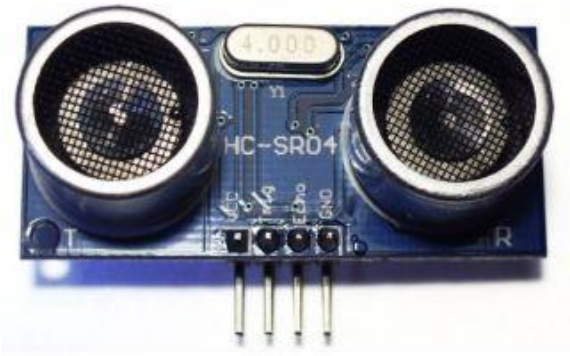


Figure 6: Ultrasonic sensor

Motion sensor: It has the ability to detect movement of a human or a pet within a room within six meters. The sensor has two trimmer resistors where the sensitivity and activation time from the moment it detects motioncan be adjusted.

Figure 7: Motion sensor

Vibration Sensor: When the sensor detects vibration, a voltage is generated, using a 1Mohm resistance

Figure 8: Vibration sensor

Triple Axis Accelerometer and Gyro Breakout Sensor: By combining a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor (DMP) capable of processing complex 9-axis Motion Fusion algorithms, the sensor can return all cross-axis alignment values.

Figure 9: Triple Axis Accelerometer and Gyro Breakout sensor

Gas Sensor: Sensitive for LPG, natural gas, coal gas. The output voltage boosts along with the concentration of the measured gases increases.

Figure 11: Gas sensor

Sound Detector: This sensor provides an audio output, but also a binary indication of the presence of sound and an analog representation of its amplitude.

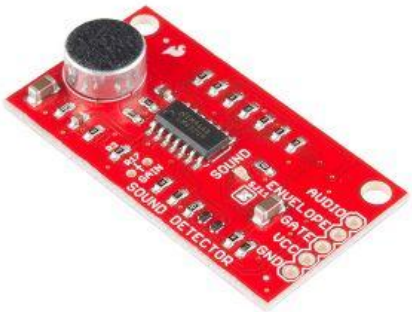


Figure 12: Sound Detector

PROJECT 1- NAME: ULTRASONIC SENSOR IN TRAFFIC SIGN

The Aim of the Project: In this project students will see how we can use an ultrasonic sensor in traffic signal using an external pull-up resistor.

Through this project, students will be able to experiment with an ultrasonic sensor, a device that measures the distance to an object using sound waves.

Methodology

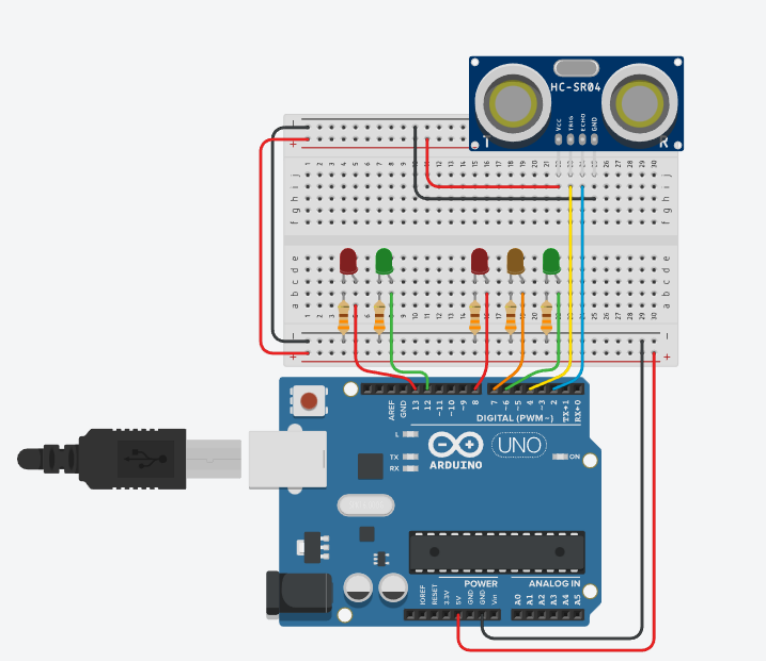
Initially, we describe the desired workflow of the project. Then we ask the students to select the needed components and use Tinkercad to design and draw the circuit. They will use the Tinkercad Code Tool to write the code.

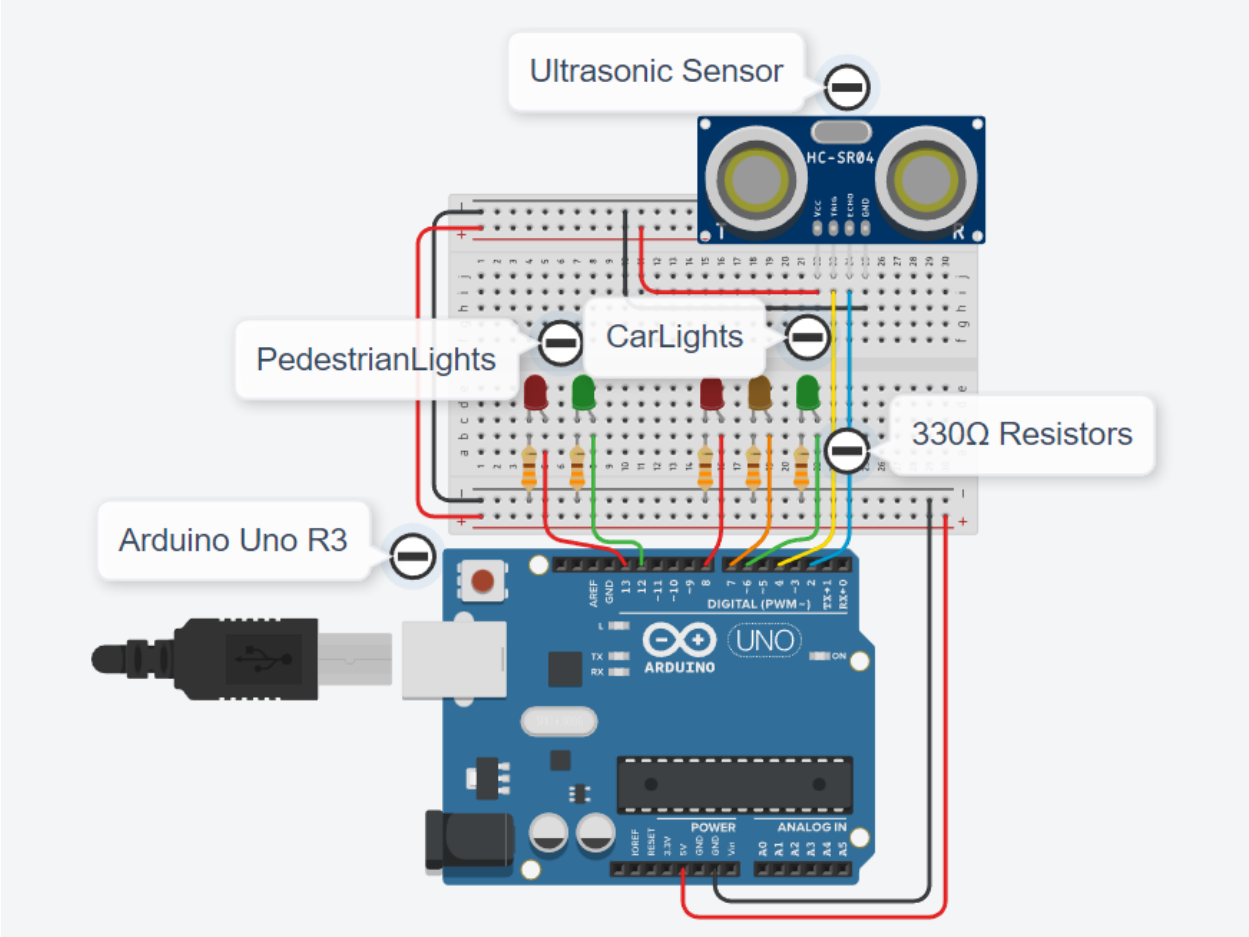
Then they will create the circuit, write the code on the Arduino software tool and uploaded it on the Arduino board.

Simulation on Tinkercad will verify that the circuit was built correctly.

This project is for basic level students, who are starting to benefit from the results and outcomes from our Erasmus+ KA-202 Strategic Partnerships in VET project, called “ArduinVET”.

Project Circuit.





How It Works:

The ultrasonic sensor works by sending a sound wave at an ultrasonic frequency and waiting for it to bounce back off the object. The time delay between the transmission of the sound and the reception of the sound is then used to calculate the distance.

Students can activate the sensor by placing their hand in front of the sensor causing the time delay between sound transmission and sound reception to decrease. The distance will be displayed on the serial screen. Students can see the activation of the traffic lights by observing the lights.

In the circuit, together with the ultrasonic sensor we use an external pull-up resistor.

- The pull-up resistor keeps Pin 3 permanently in HIGH (+5V) state.
- When the ultrasonic sensor is activated, Pin 3 is grounded (LOW state) momentarily.

Timing Table

Phases	Vehicles	pedestrians	Times	Description
0				The sensor has not been activated
1			3sec	Once the sensor is activated, phases 1 to 4 are activated and then the device returns to its previous state (Phase 0)
2			3sec	
3			4sec	
4			3sec	
0				Until the sensor is activated again

Components List:

Our components are:

- Arduino Board
- Ultrasonic Sensor HC-SR04
- Breadboard and Jump Wires
- Wire USB
- Arduino UNO R3
- 5XLED
- 5X330 ohm

Arduino Code of Project:

```
//Project Name: ULTRASOVIC SENSOR IN TRAFFIC LIGHTS
// Traffic signal with pedestrian light and ultrasonic sensor
// Declaration of constants
  const byte greenCar = 6;
  const byte orangeCar = 7;
  const byte redCar = 8;
  const byte greenPedestrian = 12;
  const byte redPedestrian = 13;
```

```
// Parameter declaration
  boolean buttonOn=false;

// Parameter declaration Ultrasonic Sensor HC-SR04 * ******
// defines pins numbers
  const int trigPin = 4;  //D4
  const int echoPin = 2;  //D3


void setup() {
 // Initialize constants Ultrasonic Sensor HC-SR04 * ******
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
 // parameter initialization
  pinMode(greenCar,OUTPUT);
  pinMode(orangeCar,OUTPUT);
  pinMode(redCar,OUTPUT);

  pinMode(greenPedestrian,OUTPUT);
  pinMode(redPedestrian,OUTPUT);

  // Initialization of prices for pedestrian signaling
  digitalWrite(greenPedestrian,LOW);
  digitalWrite(redPedestrian,HIGH);

  // Initialization of prices for vehicle signaling
  digitalWrite(greenCar,HIGH);
  digitalWrite(orangeCar,LOW);
  digitalWrite(redCar,LOW);
  pinMode (2, INPUT_PULLUP);
}

void loop() {
 //****** * 3. Code Ultrasonic Sensor HC-SR04 * ******
 // Clears the trigPin
  delay(500);
  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
  const long duration = pulseIn(echoPin, HIGH);
  Serial.print("Duration: ");
  Serial.println(duration);
```

```
// Calculating the distance
  const long distance= duration/58.2;
//Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);

  delay(2000);
  if (distance<20){
     buttonOn=true;
  }
  if(buttonOn == true)
    {
     buttonOn = false;

     delay(3000);
     digitalWrite(greenCar,LOW);
     digitalWrite(orangeCar,HIGH);

     delay(3000);
     digitalWrite(orangeCar,LOW);
     digitalWrite(redCar,HIGH);

     digitalWrite(greenPedestrian,HIGH);
     digitalWrite(redPedestrian,LOW);

     delay(4000);
     digitalWrite(greenPedestrian,LOW);
     digitalWrite(redPedestrian,HIGH);

     delay(3000);
     digitalWrite(greenCar,HIGH);
     digitalWrite(redCar,LOW);

     delay(5000);
    }else{
     digitalWrite(greenPedestrian,LOW);
     digitalWrite(redPedestrian,HIGH);

     digitalWrite(greenCar,HIGH);
     digitalWrite(orangeCar,LOW);
     digitalWrite(redCar,LOW);
     }
}
```

PROJECT 2 - NAME: ALARM SYSTEM

The Aim of the Project: In this project, students will see how an alarm system works that can detect liquid gas, movement in an enclosed space as well as unwarranted temperature rise. The alarm system will be controlled by a button and a photoresistor sensor. By pressing the button students can activate the alarm system. When the buzzer sounds by pressing the button they can stop it. When they associate an activated sensor with the push of the button they can disable the alarm system. The photoresistor activates the alarm system in case the alarm system is not activated and the lighting is low.

Through this project, students could experiment with:

- ✓ an active buzzer sensor module that has a built-in oscillator circuit that produces a sound frequency that is constant. It turns on and off with an Arduino pin
- ✓ an ultrasonic sensor, a device that measures the distance to an object using sound waves
- ✓ a gas sensor and
- ✓ two indicator lights and a button.
- ✓ a photoresistor sensor
- ✓ a temperature sensor

Methodology

First, we describe the desired workflow of the project to be developed in steps. Students are then asked to select the necessary components to design and draw the circuit. They will use the "Arduino" application to write the code.

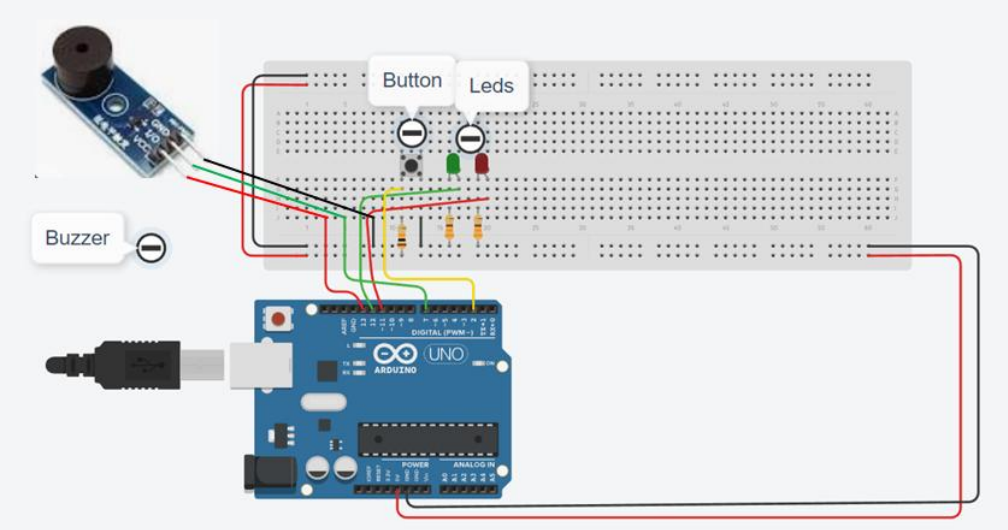
Then they will create the circuit, write the code on the Arduino software tool and uploaded it on the Arduino board.

Students will verify that the circuit was constructed correctly.

This project is for basic level students, who are starting to benefit from the results and outcomes from our Erasmus+ KA-202 Strategic Partnerships in VET project, called “ArduinVET”.

Step 1 (Buzzer with different sound tone in Alarm System)

Project Circuit.



How It Works:

An active buzzer sensor unit has a built-in oscillating circuit, so the frequency of the sound is constant. It is able to produce the sound itself.
Within this work the tone of sound ceases to have the same sound through code. As soon as the button is pressed it starts beeping with an Arduino pin, the red Led that is waiting turns off and the green Led turns on

In the circuit, together with the ultrasonic sensor we use an external pull-up resistor.

- The pull-up resistor keeps Pin 3 permanently in HIGH (+5V) state.
- When the button is pressed, Pin 3 is grounded (LOW state) momentarily.

Components List:

Our components are:

- Buzzer
- Breadboard and Jump Wires
- Wire USB
- Arduino UNO R3
- 2XLED
- 2X330 ohm
- Button
- 1X10Kohm

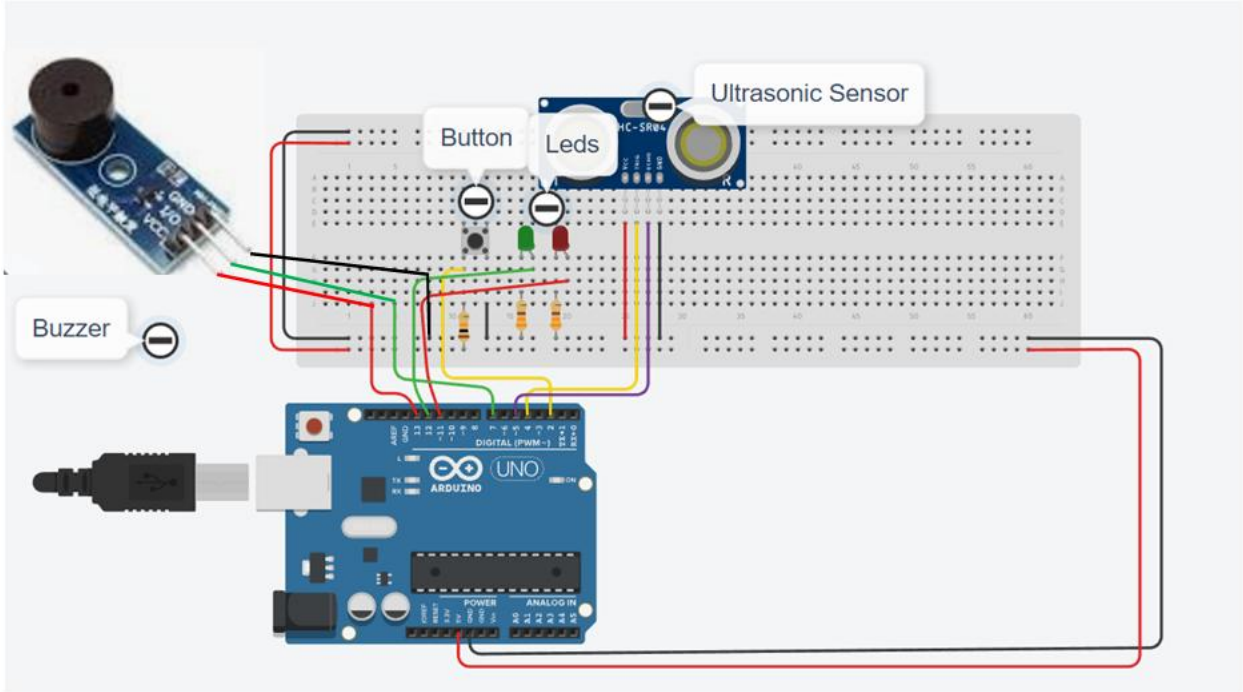
Arduino Code of Step 1:

```
//Project Name: Alarm System

// Step 1  (Buzzer with different sound tone)

//Declaration-define of variables - constants
int buzz= 7; // I/O-pin from buzzer Module connects here
int buzzVcc= 13; //Vcc-pin from buzzer Module connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

int red_led=12;
int green_led=11;

boolean buttonOn=false;
void setup()
{
  // Initialization of variables - constants
  pinMode(buzz,OUTPUT);
  pinMode(buzzVcc,OUTPUT);

  pinMode(red_led,OUTPUT);
  pinMode(green_led,OUTPUT);

  pinMode (2, INPUT_PULLUP);
}

void loop(){
  if (digitalRead(2)== LOW){
     buttonOn=true;
     }

    //Buzzer
    if(buttonOn==true){

      digitalWrite(red_led, HIGH);
      digitalWrite(green_led, HIGH);

      digitalWrite(buzz, LOW); // Tone ON
      delay(dashL); // Tone length
      digitalWrite(buzz, HIGH); // Tone OFF
      delay(sPause); // Symbol pause
      digitalWrite(buzzVcc, HIGH);
      digitalWrite(buzz, LOW); // Tone ON
```

```
    delay(dotL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dotL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    delay(lPause-sPause); // Subtracts pause already taken

    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause
    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dotL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause
    delay(wPause-sPause); // Subtracts pause already taken
    delay(5000);
    //Variable return to their original status
    buttonOn=false;
    digitalWrite(buzz, LOW); // Tone ON
    digitalWrite(buzzVcc, LOW);
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, LOW);
  }
}
```

Step 2 (Added an ultrasonic sensor HC-SR04 in Alarm System)

Connect the Ultrasonic Sensor HC-SR04



How It Works:

The ultrasonic sensor works by sending a sound wave at a frequency ultrasonic and waits to bounce off the object. The time delay between the transmission of the sound and the reception of the sound is then used to calculate the distance. Students can activate the alarm system by pressing the button, then the green led will turn on, and if they place their hand in front of the sensor, the time delay between sound transmission and sound reception will decrease and the green led will turn off, turn on red led and the audio buzzer will start to sound. The distance will be displayed on the serial screen. Students can also disable the alarm system by pressing the button again.

Arduino Code of Step 2:

```
// code of Step 2 - Added Ultrasonic Sensor HC-SR04
int red_led=12;
int green_led=11;
int sensorThres=400;

//buzzer
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
```

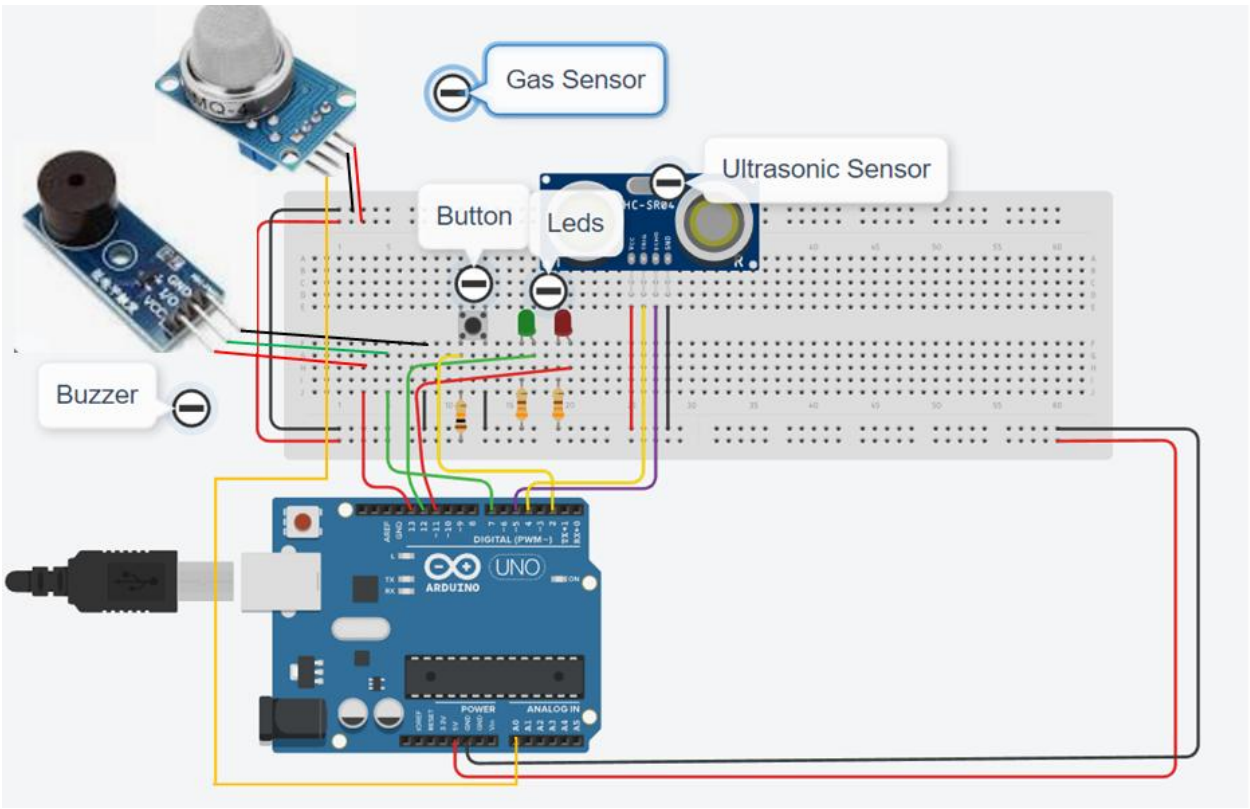
```
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

//Ultrasonic Sensor
#define trigPin 4
#define echoPin 5

//Button
boolean buttonOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(buzz,OUTPUT);
pinMode(green_led,OUTPUT);

pinMode(buzz,OUTPUT); // Set buzzer-pin as output
pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

//Ultrasonic Sensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}



void loop()
{
//Button
 if (digitalRead(2)== LOW){
     buttonOn=true;
     digitalWrite(green_led, HIGH);
     }

//Ultrasoc Sensor code
long duration, distance;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
```

```
distance = (duration/2) / 29.1;

digitalWrite(buzzVcc, LOW);

if (distance < 20)
 {
 while(buttonOn==true){

  if (digitalRead(2)== LOW){
    buttonOn=false;
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, LOW);
    digitalWrite( buzz, LOW);
 }else{
 digitalWrite(red_led, HIGH);
 digitalWrite(green_led, LOW);
 digitalWrite( buzz, HIGH);



 //buzzer
 digitalWrite(buzzVcc, HIGH);
 digitalWrite(buzz, LOW); // Tone ON
 delay(dashL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause

 digitalWrite(buzz, LOW); // Tone ON
 delay(dotL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause

 digitalWrite(buzz, LOW); // Tone ON
 delay(dashL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause

 digitalWrite(buzz, LOW); // Tone ON
 delay(dotL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause

 delay(lPause-sPause); // Subtracts pause already taken

 digitalWrite(buzz, LOW); // Tone ON
 delay(dashL); // Tone length
 digitalWrite(buzz, HIGH); // Tone OFF
 delay(sPause); // Symbol pause
 digitalWrite(buzz, LOW); // Tone ON
```

```
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
  delay(wPause-sPause); // Subtracts pause already taken

  }  //end else
  } //end while
} //end if

delay(100);
}
```

Step 3 (Added an Gas Sensor in Alarm System)

Connect the Gas Sensor



How It Works:

When the gas detector detects gas and the alarm system is already activated, the green light turns off, the red light turns on, and the buzzer sounds.

Arduino Code of Step 3:

```
//WORK : ALARM SYSTEM
// ADDED AN GAS SENSOR IN SYSTEM
int red_led=11;
int green_led=12;
int gas_value;
int sensorThres=400;

//buzzer
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
```

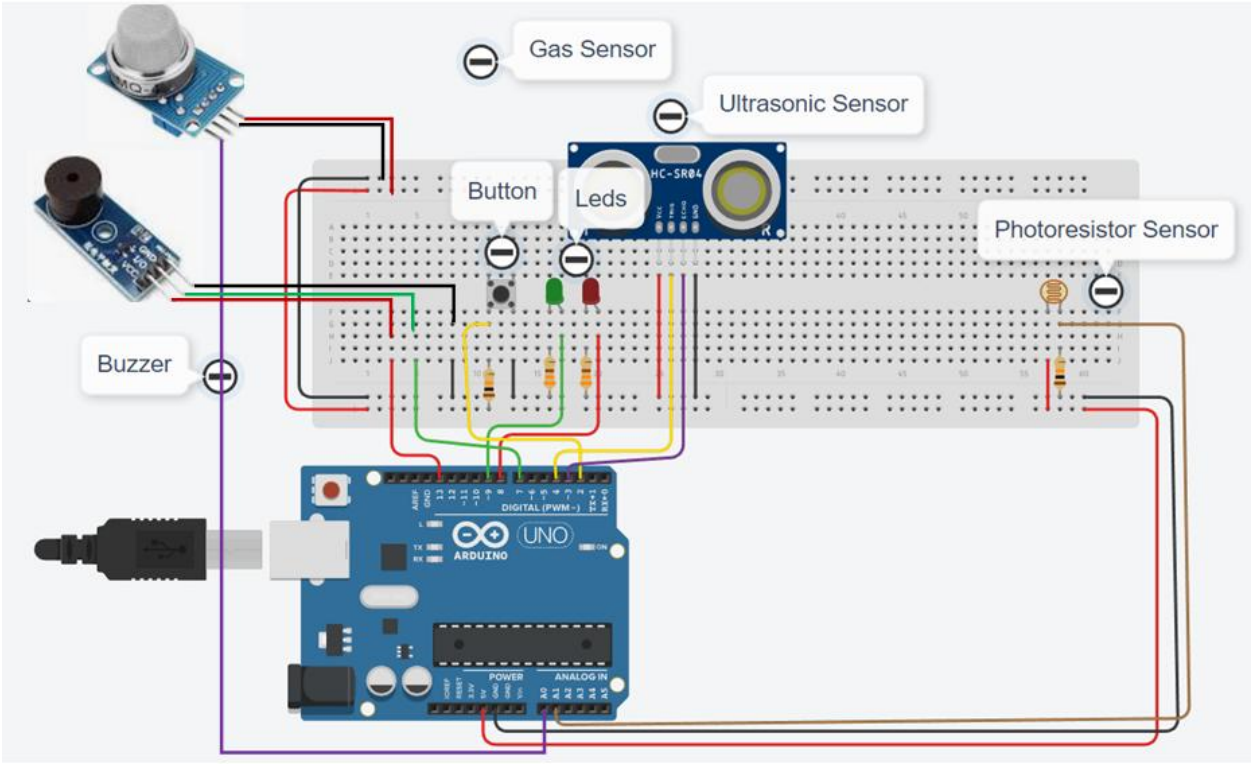
```
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

//Ultrasonic Sensor
#define trigPin 4
#define echoPin 5

//Button
boolean buttonOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(green_led,OUTPUT);
pinMode(A0,INPUT); //A0 Gass

pinMode(buzz,OUTPUT); // Set buzzer-pin as output
pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

//Ultrasonic Sensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}

void loop()
{
//Button
 if (digitalRead(2)== LOW){
     buttonOn=true;
     digitalWrite(green_led, HIGH);
     Serial.println("The alarm system is enabled!");
     }

    //Ultrasoc Sensor code
long duration, distance;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
```

```
digitalWrite(buzzVcc, LOW);

gas_value=analogRead(A0);

if (gas_value > sensorThres or distance < 20)
  {
   while(buttonOn==true){
   if (digitalRead(2)== LOW){
      buttonOn=false;
      digitalWrite(red_led, LOW);
      digitalWrite(green_led, LOW);
      digitalWrite( buzz, LOW);
      Serial.println("The alarm system is disabled!");
  }else{
  digitalWrite(red_led, HIGH);
  digitalWrite(green_led, LOW);
  digitalWrite( buzz, HIGH);
  Serial.println("DANGER!!!!");
  Serial.println(gas_value);

  //buzzer
  digitalWrite(buzzVcc, HIGH);
  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  delay(lPause-sPause); // Subtracts pause already taken

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
```

```
  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dotL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause

  digitalWrite(buzz, LOW); // Tone ON
  delay(dashL); // Tone length
  digitalWrite(buzz, HIGH); // Tone OFF
  delay(sPause); // Symbol pause
  delay(wPause-sPause); // Subtracts pause already taken
        } //end else
    }//end while
  } //end if

delay(100);
}
```

Step 4 (Added a Photoresistor Sensor in Alarm System)

Connect the Photoresistor Sensor



How It Works:

The alarm system is activated when the brightness decreases.

Arduino Code of Step 4:

```
//WORK : ALARM SYSTEM
//ADDED A PHOTORESISTOR SENSOR IN SYSTEM
int red_led=8;
int green_led=9;
int gas_value;
int sensorThres=400;

//------------------buzzer------------------------
int buzz= 13; // I/O-pin from buzzer connects here
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

//------------Ultrasonic Sensor--------------------
```

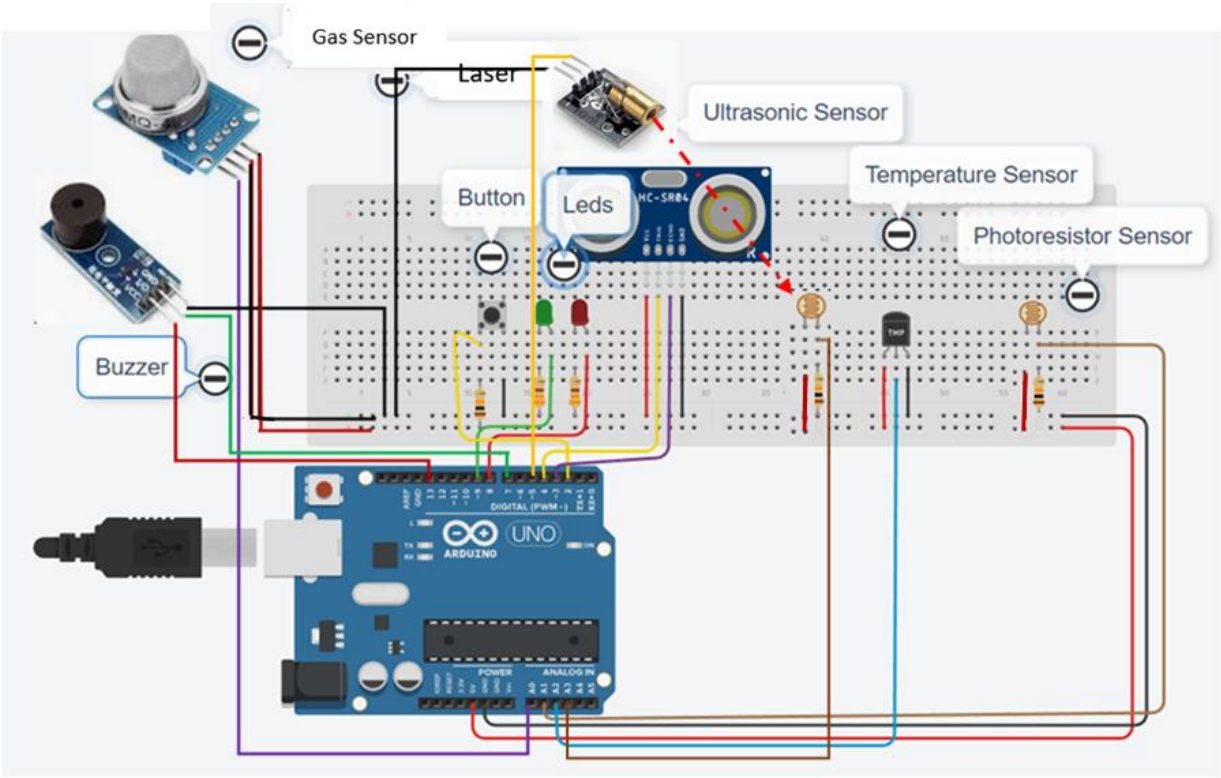
```
#define trigPin 4
#define echoPin 5

//------------------Button-------------------------
boolean buttonOn=false;

void setup()
{
pinMode(red_led,OUTPUT);
pinMode(green_led,OUTPUT);
digitalWrite(red_led, LOW);
digitalWrite(green_led, LOW);
pinMode(A0,INPUT); //A0 Gass

pinMode(buzz,OUTPUT); // Set buzzer-pin as output
pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

//Ultrasonic Sensor
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);

pinMode (2, INPUT_PULLUP);

Serial.begin(9600);
}
void loop()
{
  //-------------photoresistor---------------
  int valuePhotor = analogRead(A1);
   Serial.println("Analog valuePhotor : ");
   Serial.println(valuePhotor);
   delay(250);

  //------------------Button------------------
   if (digitalRead(2)== LOW or valuePhotor<110){
       buttonOn=true;
       digitalWrite(red_led, LOW);
       digitalWrite(green_led, HIGH);
       }

  //-------------Ultrasoc Sensor--------------
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;
```

```
Serial.println("Distance");
Serial.println(distance);

digitalWrite(buzzVcc, LOW);

gas_value=analogRead(A0);

if (gas_value > sensorThres or distance < 20){
  while(buttonOn==true){
    if (digitalRead(2)== LOW){
      buttonOn=false;
      digitalWrite(red_led, LOW);
      digitalWrite(green_led, LOW);
      digitalWrite( buzz, LOW);
      Serial.println("NO LEAKAGE");
      Serial.println(gas_value);
   }else{
   digitalWrite(red_led, HIGH);
   digitalWrite(green_led, LOW);
   digitalWrite( buzz, HIGH);
   Serial.println("DANGER!!!!");
   Serial.print("Gas Value: ");
   Serial.println(gas_value);
   Serial.print("Distance: ");
   Serial.println(distance);

   //buzzer
   digitalWrite(buzzVcc, HIGH);
   digitalWrite(buzz, LOW); // Tone ON
   delay(dashL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   digitalWrite(buzz, LOW); // Tone ON
   delay(dotL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   digitalWrite(buzz, LOW); // Tone ON
   delay(dashL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   digitalWrite(buzz, LOW); // Tone ON
   delay(dotL); // Tone length
   digitalWrite(buzz, HIGH); // Tone OFF
   delay(sPause); // Symbol pause

   delay(lPause-sPause); // Subtracts pause already taken
```

```
    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause
    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dotL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause

    digitalWrite(buzz, LOW); // Tone ON
    delay(dashL); // Tone length
    digitalWrite(buzz, HIGH); // Tone OFF
    delay(sPause); // Symbol pause
    delay(wPause-sPause); // Subtracts pause already taken
    } //end else
  } //end while
 } //end if
 delay(100);
}
```

Step 5 (Added a Temperature Sensor and a Laser Transmitter Module in Alarm System)

Connect the Temperature Sensor



In this step We will need a laser transmitter, a photoresistor for the receiver and a 1KOhm resistor to protect it.

How It Works:

The laser transmitter along with the photoresistor sensor which works as a recieper instructs the Arduino to turn on the system when for some reason the reception is interrupted. Also, the Alarm System is activated when the value of the temperature sensor exceeds a certain value.

Arduino Code of Step 5:

```
//WORK : ALARM SYSTEM
//ADDED A TEMPERATURE SENSOR IN SYSTEM
int red_led=8;
int green_led=9;
int gas_value;
int sensorThres=400;

//----------------buzzer--------------------------
int buzz= 13; // I/O-pin from buzzer connects here
```

```
int buzzVcc= 7; //Vcc-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

//-------------Ultrasonic Sensor------------------
#define trigPin 3
#define echoPin 4

//----------------Button-------------------------
boolean buttonOn=false;

//-----------temperature sensor-------------------
float tempf;
int tempPin = A2;

//-----------Laser-------------------
int Laser = 5; // creating a variable named Laser which is assigned to digital pin 5
int valueLaserPh=0;// creating a variable named valueLaserPh and setting is value to zero
float valueLaserPhF=0;// creating a variable named valueLaserPhF and setting is value to
zero

// room temperature in Fa
const float baselineTemp = 100.0;

void setup()
{
  pinMode(red_led,OUTPUT);
  pinMode(buzz,OUTPUT);
  pinMode(green_led,OUTPUT);
  pinMode(A0,INPUT); //A0 Gas

  pinMode(buzz,OUTPUT); // Set buzzer-pin as output
  pinMode(buzzVcc,OUTPUT); //Vcc Buzzer

  //Ultrasonic Sensor
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
```

```
//----------Laser-----------------
  pinMode (Laser,OUTPUT); // designating digital pin 5 for output
  digitalWrite(Laser,LOW); // just making sure the laser is off at startup or reset


  pinMode (2, INPUT_PULLUP);

  //temperature sensor
  pinMode(tempPin,INPUT);

  Serial.begin(9600);
}


void loop()
{
  digitalWrite(red_led, LOW);
  digitalWrite(Laser,HIGH);
  //-----------------Laser-------------------------------
  valueLaserPh=analogRead(A4); //reading the voltage on A4 and storing the value
received in "voltage"
  valueLaserPhF = valueLaserPh * (5.0 / 1023.0); // transforming the value stored in
"voltage" to readable information

  //----------temperature sensor-------------------------
  tempf = analogRead(tempPin);
  tempf=(tempf*5)/10;
  // convert the analog volt to its temperature equivalent
  Serial.print("TEMPERATURE = ");
  Serial.print(tempf); // display temperature value
  Serial.print("*F");
  Serial.println();
  delay(1000); // update sensor reading each one second

 //------------photoresistor-----------------
  int valuePhotor = analogRead(A1);
  Serial.print("Photoresistor value : ");
  Serial.println(valuePhotor);
  delay(250);

//---------------Button---------------------
 if (digitalRead(2)== LOW or valuePhotor<20){
```

```
    buttonOn=true;
    digitalWrite(red_led, LOW);
    digitalWrite(green_led, HIGH);
    digitalWrite(Laser,HIGH); // turning the laser on
    Serial.println("The alarm system is enabled!");
    }

//------------Ultrasoc Sensor -----------
long duration, distance;
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
Serial.print("distance : ");
Serial.println(distance);
digitalWrite(buzzVcc, LOW);
gas_value=analogRead(A0);
Serial.print("Gas Value : ");
Serial.println(gas_value);
Serial.print("Laser value : ");
Serial.println(valueLaserPhF);

if (gas_value > sensorThres or distance < 10 or tempf>baselineTemp or valueLaserPhF<1)
 {
  while(buttonOn==true){
   if (digitalRead(2)== LOW){
     buttonOn=false;
     //digitalWrite(red_led, LOW);
     digitalWrite(green_led, LOW);
     digitalWrite( buzz, LOW);
     Serial.println("The alarm system is disabled!");
     distance=100;
     tempf=0.0;
  }else{
       digitalWrite(red_led, HIGH);
       digitalWrite(green_led, LOW);
       digitalWrite( buzz, HIGH);
       Serial.println("DANGER!!!!");
       Serial.print("Gas Value: ");
```

```
Serial.println(gas_value);
Serial.print("Distance: ");
Serial.println(distance);
Serial.print("Temperature: ");
Serial.println(tempf);
Serial.print("Photoresistor value : ");
Serial.println(valuePhotor);
Serial.print("Laser value : ");
Serial.println(valueLaserPhF);

//--------------buzzer--------------
digitalWrite(buzzVcc, HIGH);
digitalWrite(buzz, LOW); // Tone ON
delay(dashL); // Tone length
digitalWrite(buzz, HIGH); // Tone OFF
delay(sPause); // Symbol pause

digitalWrite(buzz, LOW); // Tone ON
delay(dotL); // Tone length
digitalWrite(buzz, HIGH); // Tone OFF
delay(sPause); // Symbol pause

digitalWrite(buzz, LOW); // Tone ON
delay(dashL); // Tone length
digitalWrite(buzz, HIGH); // Tone OFF
delay(sPause); // Symbol pause

digitalWrite(buzz, LOW); // Tone ON
delay(dotL); // Tone length
digitalWrite(buzz, HIGH); // Tone OFF
delay(sPause); // Symbol pause

delay(lPause-sPause); // Subtracts pause already taken

digitalWrite(buzz, LOW); // Tone ON
delay(dashL); // Tone length
digitalWrite(buzz, HIGH); // Tone OFF
delay(sPause); // Symbol pause
digitalWrite(buzz, LOW); // Tone ON
delay(dashL); // Tone length
digitalWrite(buzz, HIGH); // Tone OFF
delay(sPause); // Symbol pause
```

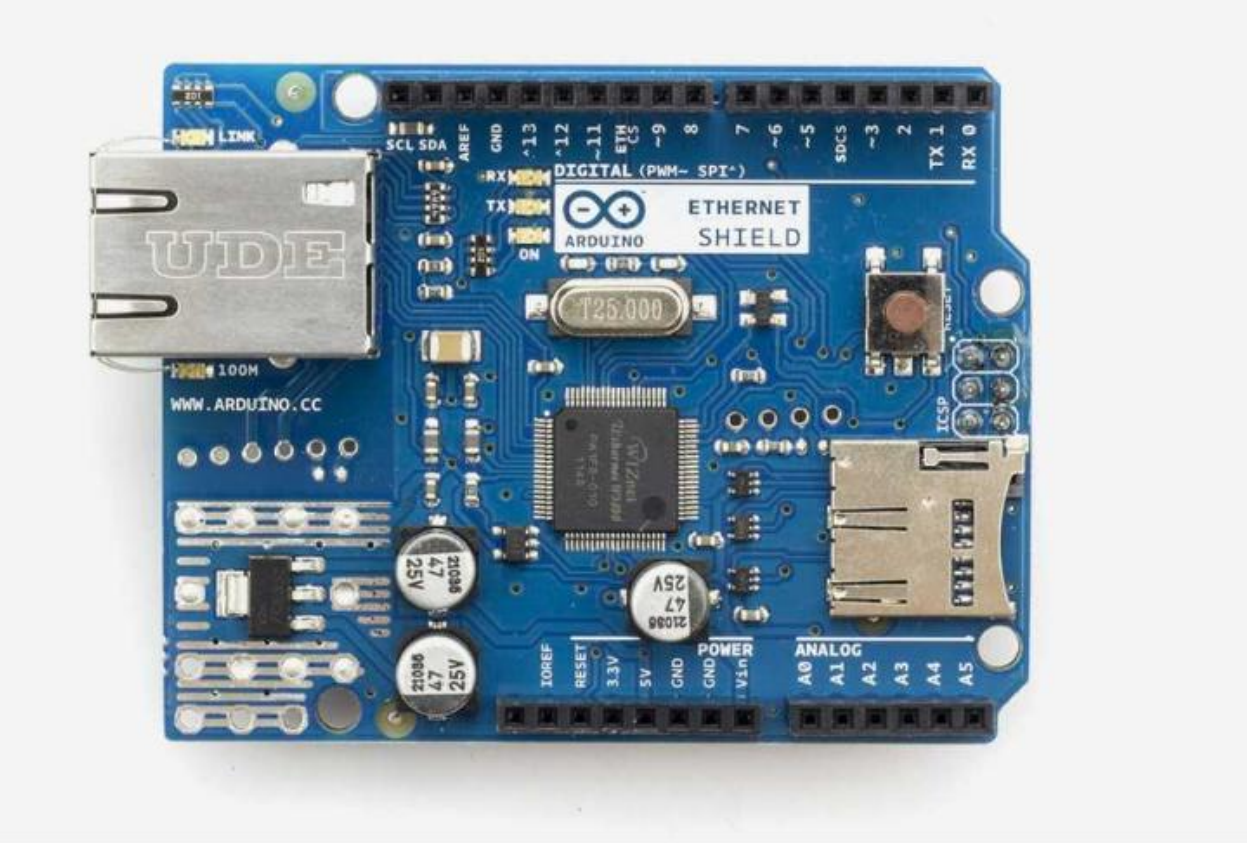
```
        digitalWrite(buzz, LOW); // Tone ON
        delay(dotL); // Tone length
        digitalWrite(buzz, HIGH); // Tone OFF
        delay(sPause); // Symbol pause

        digitalWrite(buzz, LOW); // Tone ON
        delay(dashL); // Tone length
        digitalWrite(buzz, HIGH); // Tone OFF
        delay(sPause); // Symbol pause
        delay(wPause-sPause); // Subtracts pause already taken
      }//end else
   }//end while
  } //end if
delay(100);
}
```

PROJECT 3- NAME: WebServer

The Aim of the Project: In this project, students will see how we connect an arduino to a local network using an ethernet shield. They will experiment with sensors and be able to see their received values on their computer screen via the local network.

Through this project, students could experiment with:

- ✓ a gas sensor and
- ✓ a photoresistor sensor



- ✓ a temperature sensor

This Ethernet shield allows an Arduino board to connect to the internet.

Components List:

- Ethernet Shield
- Ethernet cable
- Breadboard and Jump Wires
- Wire USB
- Arduino UNO R3
- a photoresistor sensor
- a gas sensor

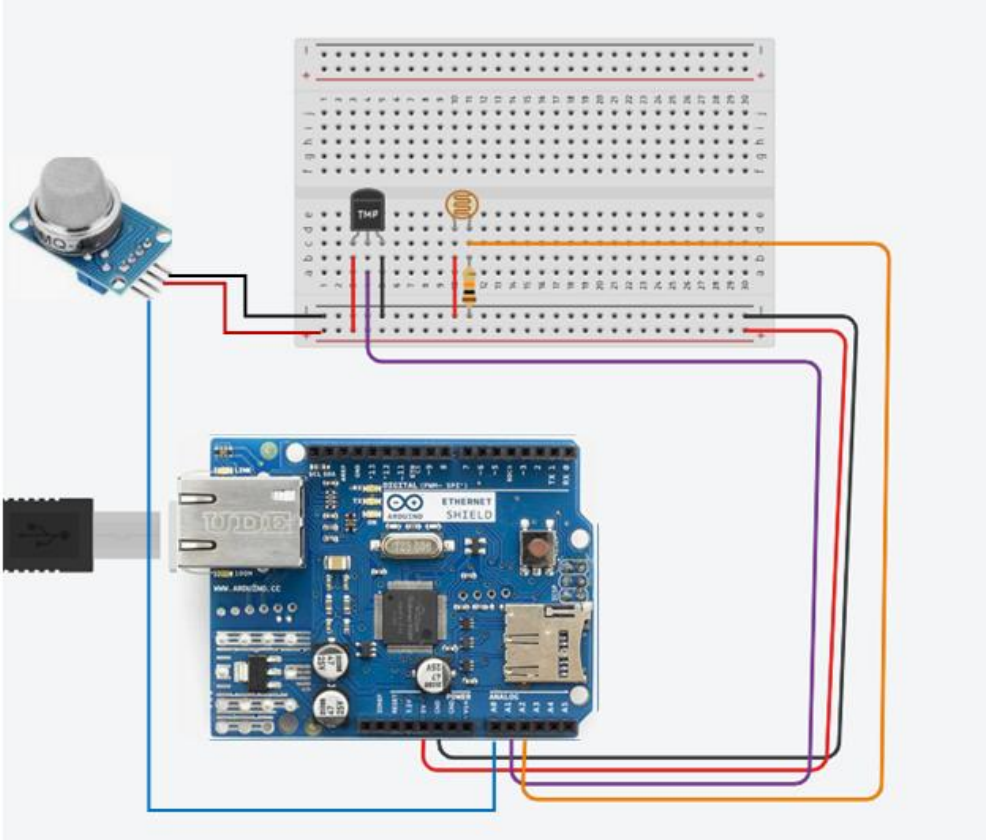
- a temperature sensor
- a resistor 10Kohm

How It Works:

The Arduino Ethernet Shield 2 allows an Arduino Board to connect to the internet. It is based on the (Wiznet W5500 Ethernet chip). The Wiznet W5500 provides a network (IP) stack capable of both TCP and UDP. It supports up to eight simultaneous socket connections.

In this example students will look up the IP of the ethernet shield in a local network browser and they will be able to see the values of the sensors they connected to the ethernet shield in a friendly page. The page refresh automatically every 5 sec.

Project Circuit.



Methodology

First, we describe the desired workflow of the project to be developed in steps. Students are then asked to select the necessary components to design and draw the circuit. They will use the "Arduino" application to write the code.

Then they will create the circuit, write the code on the Arduino software tool and uploaded it on the Arduino board.

Students will verify that the circuit was constructed correctly.

This project is for basic level students, who are starting to benefit from the results and outcomes from our Erasmus+ KA-202 Strategic Partnerships in VET project, called “ArduInVet”.

Arduino Code of Project:

```
/*
  Web Server

 A simple web server that shows the value of the analog input pins.
 using an Arduino Wiznet Ethernet shield.

 Circuit:
 * Ethernet shield attached to pins 10, 11, 12, 13
 * Analog inputs attached to pins A0 through A5 (optional)

 */

#include <SPI.h>
#include <Ethernet.h>
unsigned long refreshCounter  = 0;
// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
  0xA8, 0x61, 0x0A, 0xAE, 0x63, 0xA8
};
IPAddress ip(169, 254, 122, 108);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

EthernetClient client;
void setup() {
  // You can use Ethernet.init(pin) to configure the CS pin
  //Ethernet.init(10);  // Most Arduino shields
  //Ethernet.init(5);   // MKR ETH shield
  //Ethernet.init(0);   // Teensy 2.0
  //Ethernet.init(20);  // Teensy++ 2.0
  //Ethernet.init(15);  // ESP8266 with Adafruit Featherwing Ethernet
  //Ethernet.init(33);  // ESP32 with Adafruit Featherwing Ethernet
```

```
 // Open serial communications and wait for port to open:
 Serial.begin(9600);
 while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
 }
 Serial.println("ARDUinVET in Ethernet WebServer ");

 // start the Ethernet connection and the server:
 Ethernet.begin(mac, ip);

 // Check for Ethernet hardware present
 if (Ethernet.hardwareStatus() == EthernetNoHardware) {
  Serial.println("Ethernet shield was not found.  Sorry, can't run without hardware. :(");
        while (true) {
                delay(1); // do nothing, no point running without Ethernet hardware
        }
 }
 if (Ethernet.linkStatus() == LinkOFF) {
        Serial.println("Ethernet cable is not connected.");
 }
 // start the server
 server.begin();
 Serial.print("server is at ");
 Serial.println(Ethernet.localIP());
}

void loop() {
    // listen for incoming clients
   client = server.available();
   if (client) {
        Serial.println("ARDUinVET in client");
       // an http request ends with a blank line
       boolean currentLineIsBlank = true;
       while (client.connected()) {
           if (client.available()) {
              char c = client.read();
             Serial.write(c);
             // if you've gotten to the end of the line (received a newline
            // character) and the line is blank, the http request has ended,
            // so you can send a reply
               if (c == '\n' && currentLineIsBlank) {
                    // send a standard http response header
                    client.println("HTTP/1.1 200 OK");
                    client.println("ARDinVET - HTTP");
                    client.println("Content-Type: text/html");
                    client.println("Connection: close");  // the connection will be closed after
//completion of the response
                    client.println("Refresh: 5");  // refresh the page automatically every 5 sec
```

```
                client.println();
                webpage(client);
                break;
                }
                if (c == '\n') {
                        // you're starting a new line
                        currentLineIsBlank = true;
                        } else if (c != '\r') {
                                // you've gotten a character on the current line
                                currentLineIsBlank = false;
                          }
                }
            }
            // give the web browser time to receive the data
            delay(1);
            // close the connection:
            client.stop();
            Serial.println("client disconnected");
        }
}

void webpage(EthernetClient client) { /* function webpage */
////Send wbepage to client
client.println("ARDUinVET - HTTP");
//client.println("Content-Type: text/html");
client.println(""); //  do not forget this one
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<head>");
client.println("<title>ARDUinVET</title>");
//client.println("<script src='https://smtpjs.com/v3/smtp.js'></script>");
//client.println("<script>");
//client.println("function sendEmail() {");
/*client.println("Email.send({");
client.println("  Host: 'smtp.gmail.com',");
client.println("  Username : '**********@gmail.com',");
client.println("  Password : '******************',");
client.println("  To : '**********@gmail.com',");
client.println("  From : '**********@gmail.com',");
client.println("  Subject : 'Data from Arduino',");
client.println("  Body : 'Sensors value....',");
client.println("  }).then(");
client.println("    message => alert('mail sent successfully')");
client.println("  );");*/
//client.println("}");
//client.println("</script>");
client.println("</head>");
client.println("<body bgcolor = '#cccc00'>");
client.println("<hr/><hr>");
```

```
client.println("<h1 style='color : #0000cc;'><center>  Alarm System </center></h1>");
client.println("<hr/><hr>");
client.println("<br><br>");
client.println("<br><br>");
client.println("<center>");
client.println("<br>Sensors Output<br>");
client.println("Gas.....:");
client.println("  <input value=" + String(analogRead(A0)) + " readonly></input>");
client.println("<br>");
client.println("  Photoresistor..:");
client.println("  <input value=" + String(analogRead(A1)) + " readonly></input>");
client.println("<br>");
client.println("  Temperature....:");
client.println("  <input value=" + String(analogRead(A2)) + " readonly></input>");
client.println("<br>");
client.println("  </center>");
client.println("<br><br>");
client.println("<center>");
client.println("<a style='color : #0000cc;'
href='https://www.arduinvet.com/'>www.arduinvet.com</a>");
client.println("</center>");
client.println("<br><br>");
client.println("</body></html>");
client.println();
delay(1);
}
```

a screen sot of project-page

Co-funded by the Erasmus+ Programme of the European Union

Erasmus+ KA-202

Strategic Partnerships Project for Vocational Education and Training

Project Title: "Teaching and Learning Arduinos in Vocational Training"

Project Acronym: " ARDUinVET "

Project No: "2020-1-TR01-KA202-093762"

Free Arduino Projects

(These projects were made by students of partner schools in the ARDUinVET partnership.)

NO:	Project NAME	Country
1	SOCIAL DISTANCING HAT	TURKIYE
2	TEMPERATURE METER (GREECE)	GREECE
3	EMERGENCY BRAKE BLINKER - ADAPTIVE BRAKE LIGHT	ROMANIA
4	LINEFOLLOWER	AUSTRIA
5	AUTOMATED GREENHHOUSE (ITALY)	ITALY

1_PROJECT NAME: SOCIAL DISTANCING HAT (TURKIYE)

The Aim of the Project: In these days, when the Covid-19 pandemic is effective, following the "SOCIAL DISTANCE" rule is one of the basic conditions of not getting the disease.

In order to draw attention to the "social distaning" issue, we decide to design an Arduino project on the subject.

The rule is reminded with the “Social Distancing Hat”, which we will prepare. It will give an audible buzzer sound and at the same time a light warning.

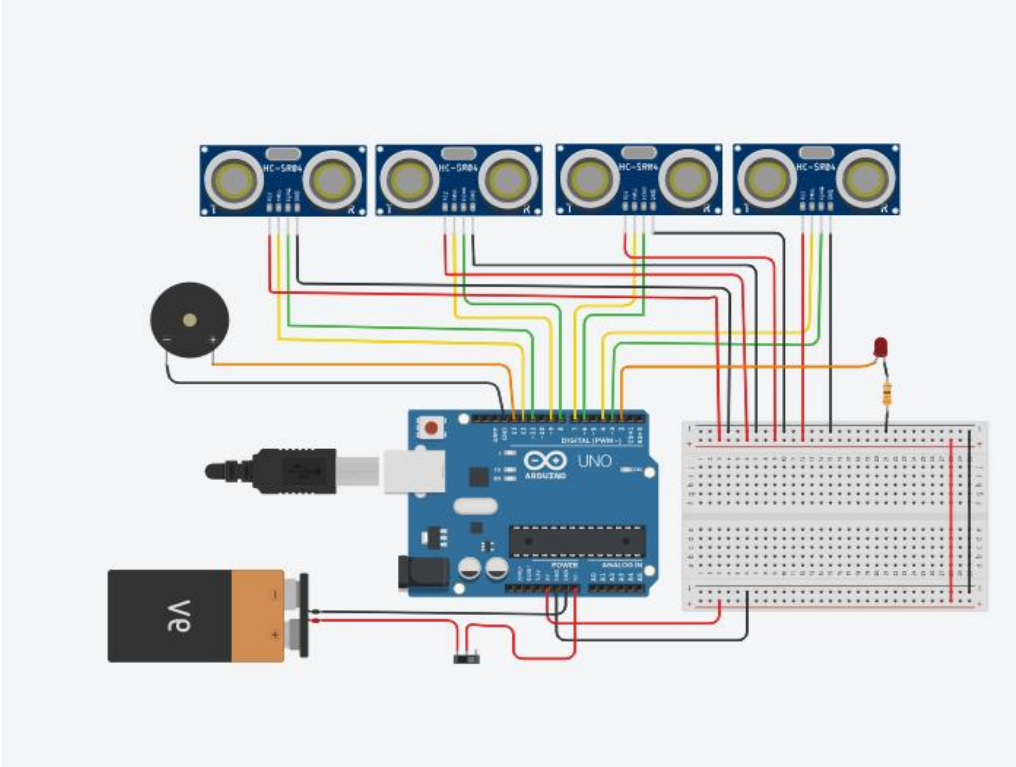
Our Project Method:

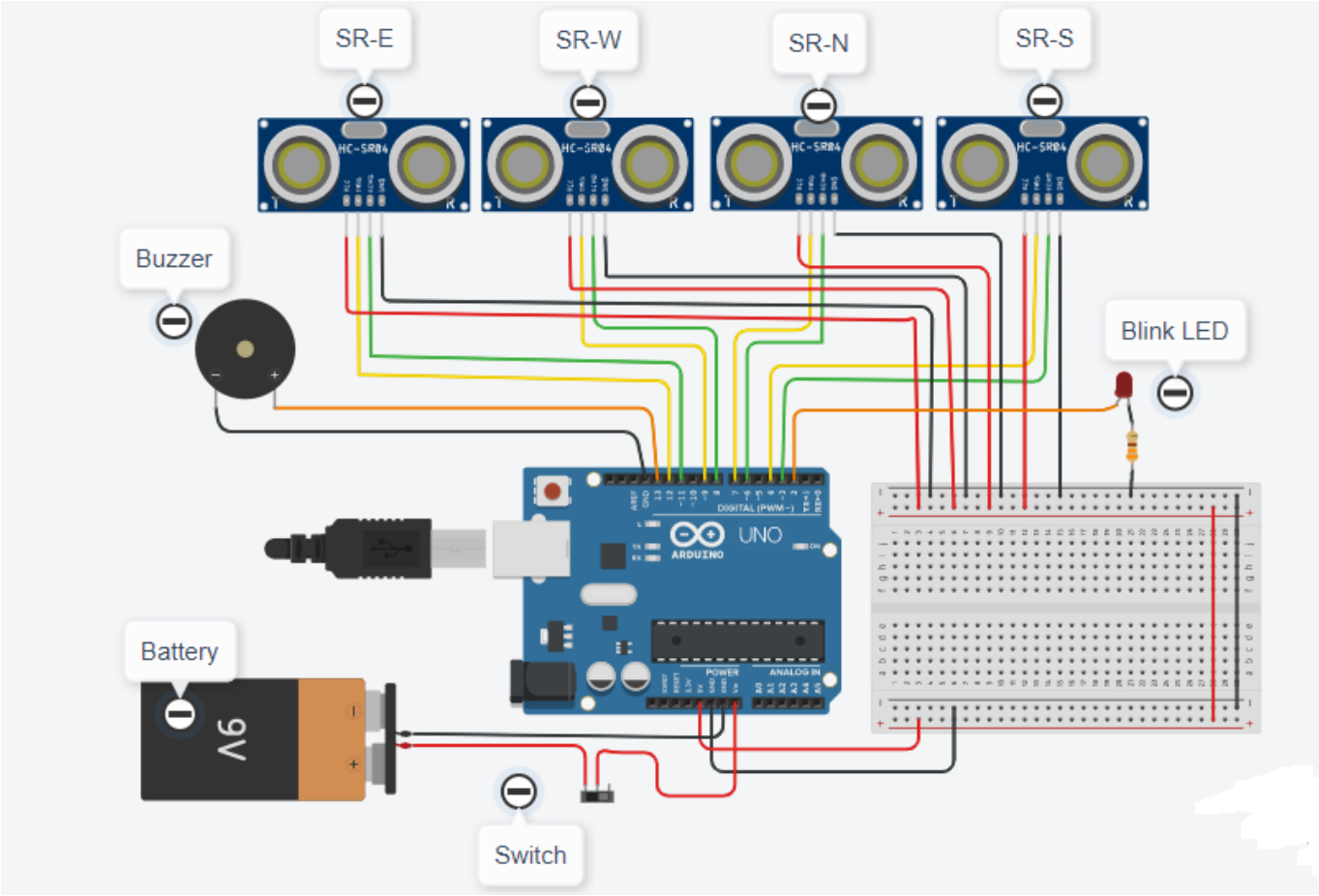
First, the algorithm of the project is prepared. According to this prepared algorithm, the logical design is made and the Arduino program is written. Arduino is used in the design of the circuit. While making these designs, our students benefit from all the results and products of the Erasmus+ KA-202 VET project, named as “ArduinVET”.

Our finalized design circuit is set up after providing the necessary equipment and programming the Arduino. Our final design circuit is visually placed on the cap.

This project is exhibited at fairs and reminds the visitors of the "Social Distancing" rule and its importance.

Project Circuit.





How It Works:

4 ultrasonic distance sensors are placed in 4 directions of the hat. When the sensors detect someone, approaching or close to 100 cm or below, the buzzer will sound and the led will flash. The circuit will be powered by a 9V battery.

List of Materials:

A Hat, an Arduino Uno R3, 4 Ultrasonic Distance Sensors, a Buzzer, a 330 Ω Resistor, a Blink-Red LED, a 9V Battery, a Slide Switch.

Arduino Program of Project:

//Project Name: SOCIAL DISTANCING HAT:

```
int trigPin=12;
int echoPin=11;
int trigPin2=9;
int echoPin2=8;
int trigPin3=7;
int echoPin3=6;
int trigPin4=4;
int echoPin4=3;
void setup()
{
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
 pinMode(trigPin2, OUTPUT);
pinMode(echoPin2, INPUT);
pinMode(trigPin3, OUTPUT);
pinMode(echoPin3, INPUT);
 pinMode(trigPin4, OUTPUT);
pinMode(echoPin4, INPUT);
pinMode(13,OUTPUT);
pinMode(2,OUTPUT);
 Serial.begin(9600);
}
void loop() {
 digitalWrite(trigPin, LOW);
```

```
delay(3);
digitalWrite(trigPin, HIGH);
delay(3);
digitalWrite(trigPin, LOW);
int time1 = pulseIn(echoPin, HIGH);
int distance1 = (time1/2) / 28.97;         //Social distance is equal to and less than 100cm.
//-----------------------------------------------------
 digitalWrite(trigPin2, LOW);
delay(3);
digitalWrite(trigPin2, HIGH);
delay(3);
digitalWrite(trigPin2, LOW);

int time2 = pulseIn(echoPin2, HIGH);
int distance2 = (time2/2)  / 28.97;    //Social distance is equal to and less than 100cm.
//-----------------------------------------------------
 digitalWrite(trigPin3, LOW);
delay(3);
digitalWrite(trigPin3, HIGH);
delay(3);
digitalWrite(trigPin3, LOW);
int time3 = pulseIn(echoPin3, HIGH);
int distance3 = (time3/2)  / 28.97;       //Social distance is equal to and less than 100cm.
//-----------------------------------------------------
digitalWrite(trigPin4, LOW);
delay(3);
```

```
digitalWrite(trigPin4, HIGH);
delay(3);
digitalWrite(trigPin4, LOW);
int time4 = pulseIn(echoPin4, HIGH);
int distance4 = (time4/2)  / 28.97;         //Social distance is equal to and less than 100cm.
//---------------------------------------------------------
if(distance1<75)
 {
  digitalWrite(13,HIGH);
  digitalWrite(2,HIGH);
  }
 else if(distance2<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 else if(distance3<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
 else if(distance4<75)
 {
  digitalWrite(13,HIGH);
   digitalWrite(2,HIGH);
  }
```

```
else
{
digitalWrite(13,LOW);
digitalWrite(2,LOW);
}
 Serial.println(distance1);      // To see which ultrosonc sensor is activated on the Serial monitor
 Serial.println(distance2);
 Serial.println(distance3);
 Serial.println(distance4);
 delay(500);
}
```

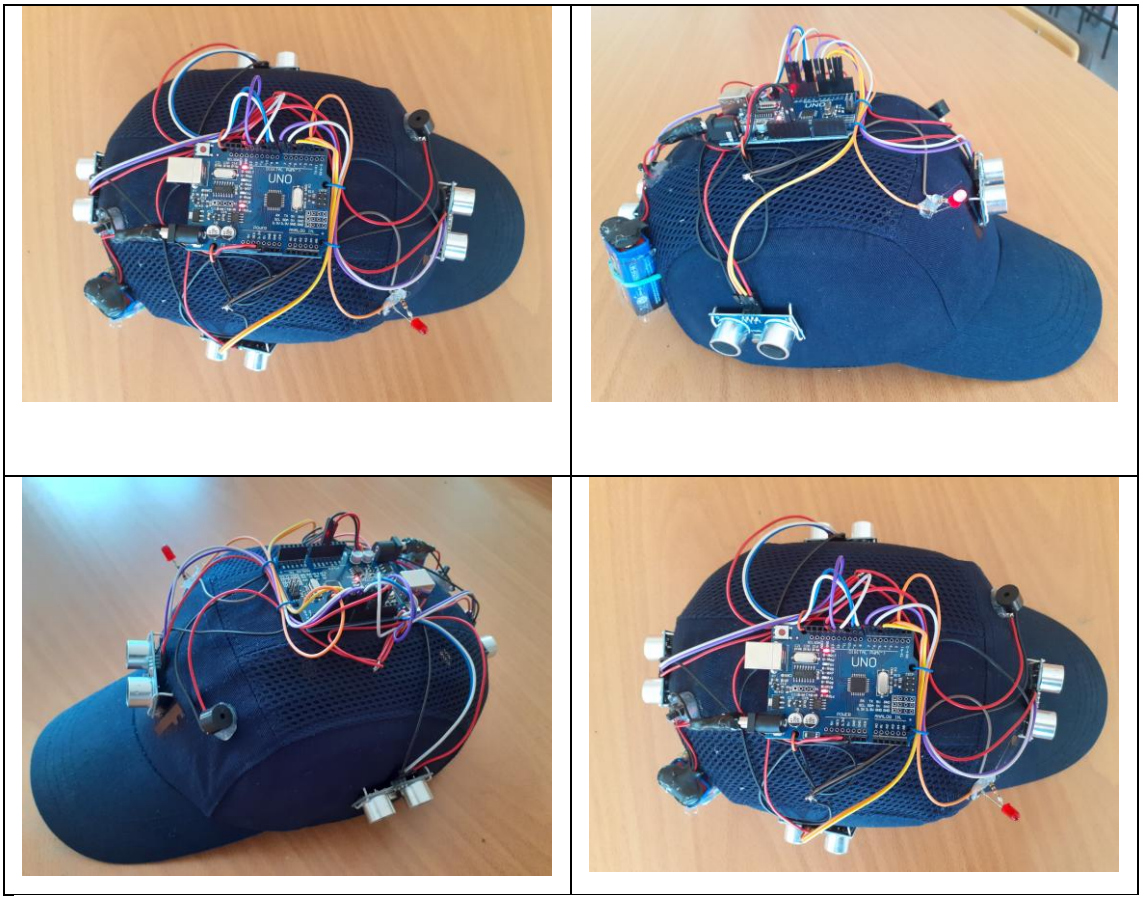


Photo: Final design circuit on the cap

2_PROJECT NAME: TEMPERATURE METER (GREECE)

The Aim of the Project: This project is an extended version of the simple basic Love Meter project someone can find in the original Arduino handbook. In this extended version, students will be able to deal with a sensor (temperature sensor), experiment with the potentiometer, light an RGB LED and get to know the LCD display (if not available, otherwise the display will be just the Serial Monitor of the Arduino software tool).

It's a project which allows the students to experiment with many components of different types and get familiar with the Input values and Display methods.

Methodology

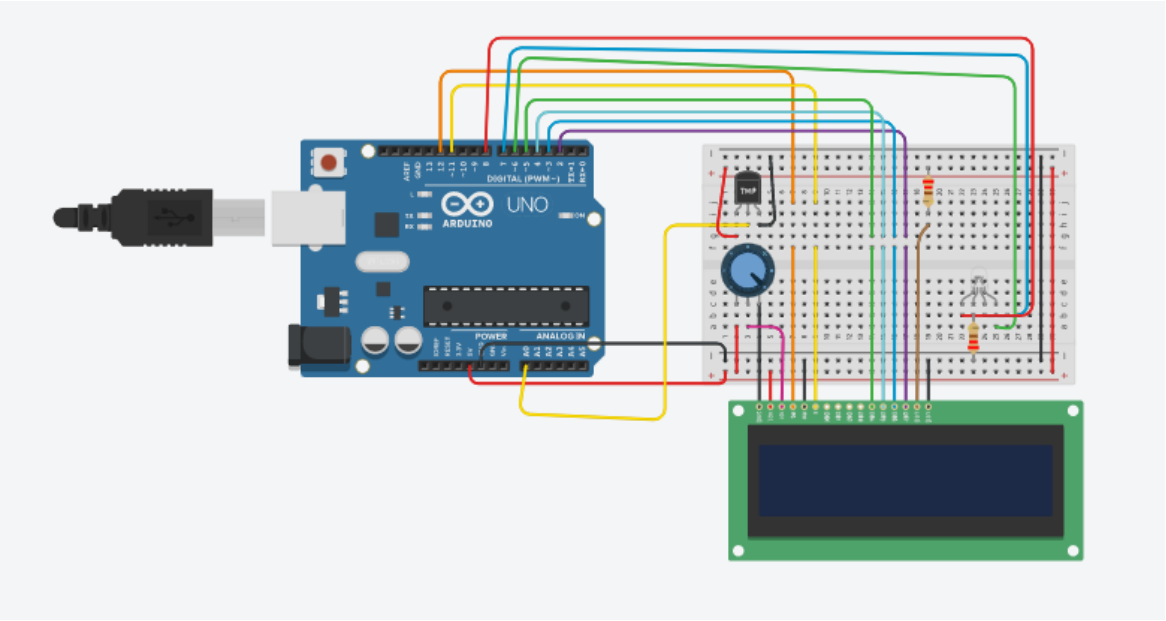
Initially, we describe the desired workflow of the project. Then we ask the students to select the needed components and use Tinkercad to design and draw the circuit. They will use the Tinkercad Code Tool to write the code.

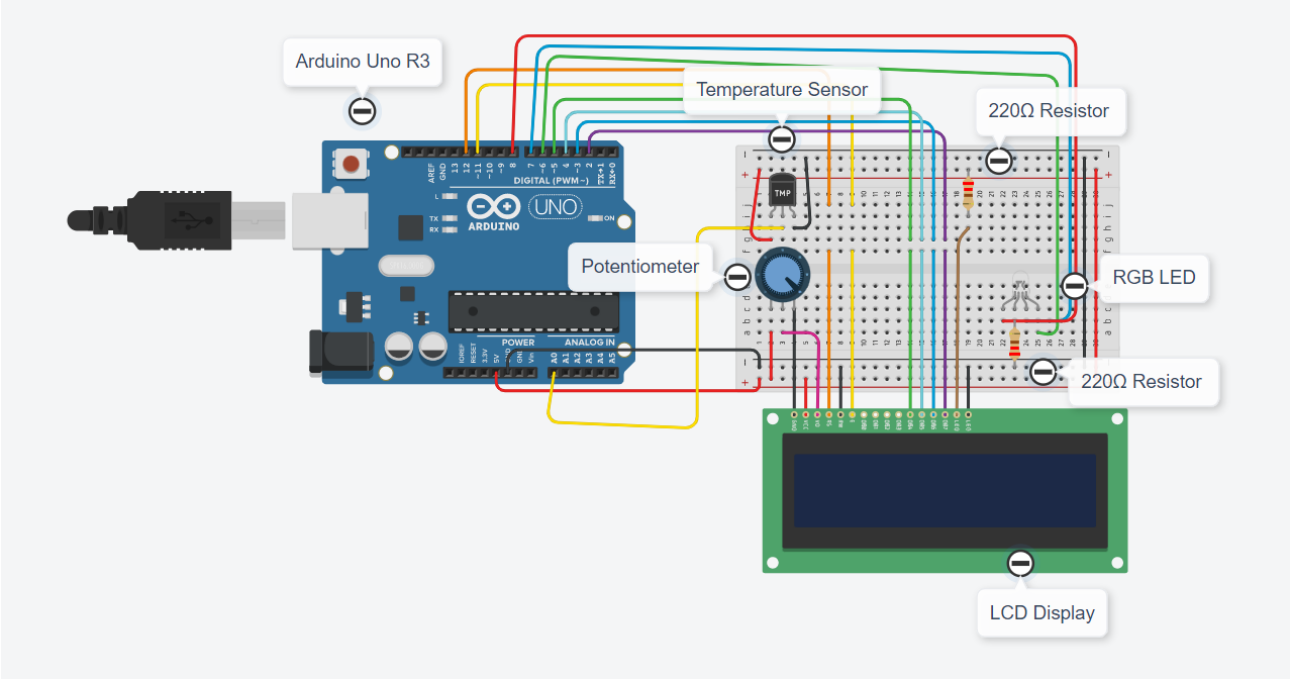
Then they will create the circuit, write the code on the Arduino software tool and uploaded it on the Arduino board.

Simulation on Tinkercad will verify that the circuit was built correctly.

This project is for basic level students, who are starting to benefit from the results and outcomes from our Erasmus+ KA-202 Strategic Partnerships in VET project, called "ArduinVET".

Project Circuit.





How It Works:

The temperature sensor inputs the values of the room temperature. Off course, students can change the temperature using their fingers. The temperature will be displayed in the LCD Display (if available) and the Serial Monitor. The RGB LED will have different colors (off, green, blue, red) according to the temperature value. The potentiometer will be used as a switch for the LCD Display. The circuit will be powered from the USB Cable (if needed, you can use a battery box instead).

Components List:

Our components are: an Arduino Uno R3, a Temperature Sensor, two (2) 220Ω Resistors, an RGB LED and (optional) a Potentiometer and a LCD Display (16*2).

Arduino Code of Project:

//Project Name: TEMPERATURE METER

// include the library code:

```
#include <LiquidCrystal.h>
int t=0;
int sensor = A0;
float temp;
float tempc;
float tempf;
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//************RGB**********
const int sensorPin = A0;
// room temperature in Celsius
const float baselineTemp = 20.0;
//**************************
void setup() {
  // set up the LCD's number of columns and rows:
pinMode(sensor,INPUT);
lcd.setCursor (0,0);
lcd.begin(16, 2);
  // Print a message to the LCD.
lcd.print ("    ARDUINO     ");
lcd.setCursor (0,1);
lcd.print ("TEMPERATURE METER");
```

```
delay (3000);
Serial.begin(9600);
for(int pinNumber = 6; pinNumber<8; pinNumber++){
    pinMode(pinNumber,OUTPUT); // pin6:Blue pin7:Green  pin8:Red
    digitalWrite(pinNumber, LOW);
   }
}
void loop() {
delay(2000);
t=t+2;
temp=analogRead(sensor);
//tempc=(temp*5)/10;
tempc=map(((temp-20)*3.04), 0, 1023, -40, 125);
tempf=(tempc*1.8)+32;
Serial.println("_______");
Serial.println("Temperature Logger");
Serial.print("Time in Seconds= ");
Serial.println(t);
Serial.print("Temp in deg Celcius = ");
Serial.println(tempc);
Serial.print("Temp in deg Fahrenheit = ");
Serial.println(tempf);
lcd.setCursor(0,0);
lcd.print("Temp in C = ");
lcd.println(tempc);
lcd.setCursor(0,1);
```

```
lcd.print("Temp in F = ");
lcd.println(tempf);


//*************code RGB***************
  // read the value on AnalogIn pin 0
  // and store it in a variable
  int sensorVal = analogRead(sensorPin);
  // send the 10-bit sensor value out the serial port
  Serial.println("sensor Value: ");
  Serial.println(sensorVal);

  // convert the ADC reading to voltage
  float voltage = (sensorVal/1024.0) * 5.0;
  // Send the voltage level out the Serial port
  Serial.println(", Volts: ");
  Serial.println(voltage);
   if(tempc < 20){
   digitalWrite(6, LOW);
   digitalWrite(7, LOW);
   digitalWrite(8, LOW);
   }
   else if(tempc >= 20 && tempc < 80){
         digitalWrite(6, HIGH);
   digitalWrite(7, LOW);
   digitalWrite(8, LOW);
  }
```

```
 else if(tempc >= 80 && tempc < 140){
        digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
  digitalWrite(8, LOW);
}
 else if(tempc >= 140){
        digitalWrite(6, LOW);
  digitalWrite(7, LOW);
  digitalWrite(8, HIGH);
}
delay(100);
}
```

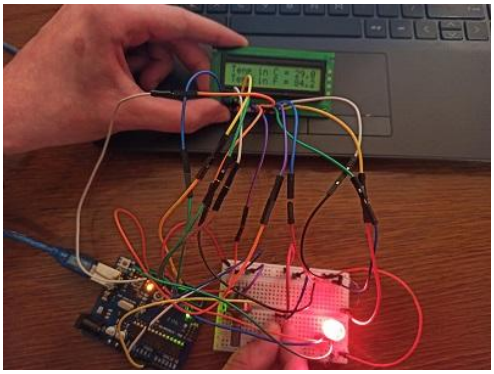
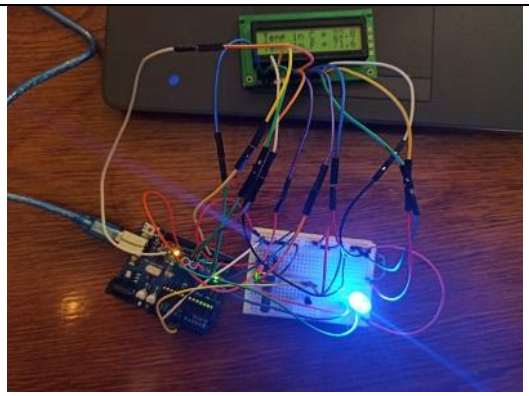





Photo: Project training kit and Students working on the project.

3_PROJECT NAME: EMERGENCY BRAKE BLINKER - ADAPTIVE BRAKE LIGHT

(ROMANIA)

The Aim of the Project: In the event of heavy braking, the adaptive brake light can warn oncoming traffic and thus help prevent rear-end collisions. They are activated if, during braking, and depending on the braking pressure and speed, if a critical braking situation is identified.

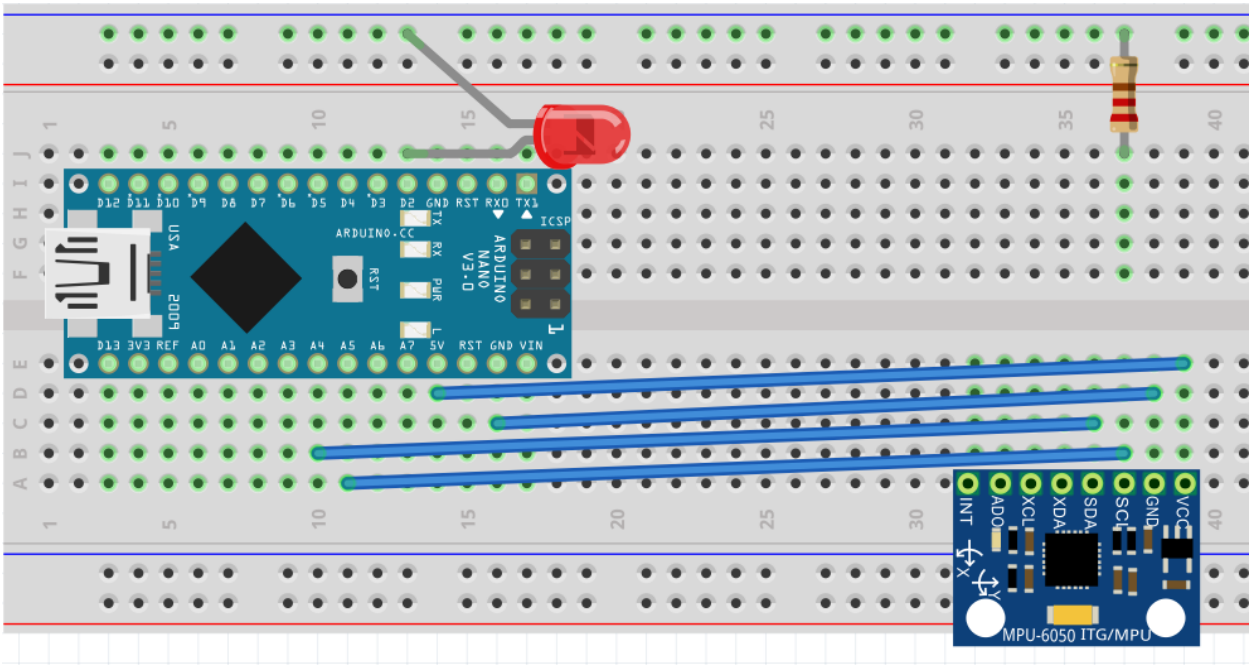
Our Project Method:

First, the algorithm of the project is prepared. According to this prepared algorithm, the logical design is made and the Arduino program is written. Arduino is used in the design of the circuit. While making these designs, our students benefit from all the results and products of the Erasmus+ KA-202 VET project, named as "ArduinVET".

Our finalized design circuit is set up after providing the necessary equipment and programming the Arduino. Our final design circuit is placed in a car.

This project is exhibited at fairs and reminds the visitors how important is traffic safety.

Project Circuit.



How It Works:

Emergency brake lights are **activated to alert vehicles behind about heavy braking**. The function means that the brake light flashes instead of - as in normal braking - shining with a constant glow. Emergency brake lights are activated at speeds above 50 km/h in the event of heavy braking.

It is based on an electronic accelerometer (MPU6050) assisted by a microcontroller and detects sudden braking based on the force of inertia and quickly flashes the 3rd brake stop, making the car much easier to notice than the rear.

MPU6050 has a 3-axis gyroscope, 3- axis Accelerometer and a Digital motion processor integrated on a single chip. It works on the power supply of 3V-5V. MPU6050 **uses the I2C protocol for communication and transfer of data**. This module has a built-in 16-bit ADC which provides great accuracy.

The Kalman filter is an efficient recursive filter that evaluates the state of a dynamic system based on a series of noise-sensitive measurements. Due to its intrinsic characteristics, it is an excellent filter for noise and disturbances that act on zero-average Gaussian systems.

List of Materials:

An Arduino Nano, an MPU6050, an Resistor 180 Ohm, an red LED.

Arduino Program of Project:

//Project Name: EMERGENCY BRAKE BLINKER - ADAPTIVE BRAKE LIGHT

```
#include<Wire.h>;
//https://www.codetd.com/en/article/11749279 this is an articol who explain very well how to
comunicate more fast with MPU6050
//in automotive conditions is many noise from the road conditions, vibrations etc. and I'll use a
very good software filter (Kalman)
const int MPU_addr = 0x68;
float kalman_old = 0;
float cov_old = 1;
float val1, val2, val3, val4, val5;
int med1_sort, med2_sort, med3_sort, med4_sort;
float avg;
int med;
```

```
int m;
int med_sort[5];
int c_avg = 1;
int c_med = 1;
int d = 0;
float old_x = 0;
float real_angle = 0;
float prev_angle = 0;
unsigned long previousMillis = 0;
const long interval = 50;
void setup() {
  Serial.begin(9600); // for USB monitor
  Serial.println ("Hit a key to start");     // signal initalization done
  while (Serial.available() == 1);
  //Connect to G-Sensor
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0x00); // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  delay(50);
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x1C);
  Wire.write(0x00);
  Wire.endTransmission(true);
  pinMode(2, OUTPUT); //output signal for FIRST and SECOND stage.
digitalWrite (2, LOW);
//pinMode(7, OUTPUT);//output signal for SECOND stage. Enable this and next line if you want more expressive functionality.
//digitalWrite (7, LOW); If enable this option, you need enable digital output pin 7 in all sketch!!
//this signal might be use with one PNP transistor or P-Gate MOSFET for complete the circuit
//in real circuit I use a 5v blue led. In fritzing circuit use a regular 1.8 red led with aditional resistor - 180 ohm}
```

```
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3D);
// 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
// 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
// 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 2, true);
    int16_t YAxisFull =  (Wire.read() << 8 | Wire.read());
    float YAxisFinal = (float) YAxisFull / 16384.0;
    int YAxis_Filtered = general_filter(c_avg, c_med, YAxisFull);
    int YAxis_kalman = kalman_filter(YAxisFull);
    int YAxis_movavg = mov_avg(c_avg, YAxisFull);
    int YAxis_median = median(c_med, YAxisFull);
    Serial.print("YAxis_Filtered");Serial.print(YAxis_Filtered/16384.0);
    Serial.print("\t");
   // Serial.print("YAxis_kalman");Serial.print(YAxis_kalman);
  //  Serial.print("\t");
  //  Serial.print("YAxis_movavg");Serial.print(YAxis_movavg);
   // Serial.print("\t");
  //  Serial.print("YAxis_median");Serial.print(YAxis_median);
  //  Serial.print("\t");
    Serial.print("YAxisFull");Serial.println(YAxisFull/16384.0);
   // Serial.print("\t");
   // Serial.print("YAxis_Final");Serial.println(YAxis_Final);
//FIRST stage
if (YAxis_Filtered/16384.0 > 0.4 and YAxis_Filtered/16384.0 <= 0.7){
  for (int i = 1; i <= 4; i++) {
      digitalWrite (2, HIGH);
     // digitalWrite (7, HIGH);
```

```
      delay (75);
      digitalWrite (2, LOW);
     // digitalWrite (7, LOW);
      delay (50);}}
else {
  //SECOND stage - very hard brake
if (YAxis_Filtered/16384.0 > 0.7){
   for (int i = 1; i <= 7; i++) {
      digitalWrite (2, HIGH);
     // digitalWrite (7, HIGH);
      delay (75);
      digitalWrite (2, LOW);
      //digitalWrite (7, LOW);
      delay (50);}}}
    c_avg = c_avg + 1;
    c_med = c_med + 1;
    if (c_avg > 5 && c_med > 5)
    {      c_avg = 6;
      c_med = 6;    }  }}
float kalman_filter (float input)
{  float kalman_new = kalman_old;
  float cov_new = cov_old + 0.50;
  float kalman_gain = cov_new / (cov_new + 0.9);
  float kalman_calculated = kalman_new + (kalman_gain * (input - kalman_new));
  cov_new = (1 - kalman_gain) * cov_old;
  cov_old = cov_new;
  kalman_old = kalman_calculated;
  return kalman_calculated;}
float mov_avg (int counter, float input)
{  avg = 0;
  m = 0;
  if (counter == 1) {
    val1 = input;
```

```
  avg = val1;  }
else if (counter == 2) {
  val2 = input;
  avg = val2;  }
else if (counter == 3) {
  val3 = input;
  avg = val3;  }
else if (counter == 4) {
  val4 = input;
  avg = val4;  }
else if (counter == 5) {
  val5 = input;
  avg = val5;  }
else if (counter > 5 ) {
  counter = 6;
  if (val1 == 0) {
    m = m + 1;    }
  if (val2 == 0) {
    m = m + 1;    }
  if (val3 == 0) {
    m = m + 1;    }
  if (val4 == 0) {
    m = m + 1;    }
  if (val5 == 0) {
    m = m + 1;    }
  if (input == 0) {
    m = m + 1;    }
  d = 6 - m;
  if (d == 0)
  {      avg = input;
    counter = 1;    }
  else
  {      avg = (val1 + val2 + val3 + val4 + val5 + input) / d;    }
```

```
    val1 = val2;
    val2 = val3;
    val3 = val4;
    val4 = val5;
    val5 = input;  }
  return avg;}
float median (int counter, int input)
{  if (counter == 1) {
    med1_sort = input;
    med_sort[0] = med1_sort;  }
  else if (counter == 2) {
    med2_sort = input;
    med_sort[1] = med2_sort;  }
  else if (counter == 3) {
    med3_sort = input;
    med_sort[2] = med3_sort;  }
  else if (counter == 4) {
    med4_sort = input;
    med_sort[3] = med4_sort;  }
  else if (counter >= 5) {
    counter = 6;
    med_sort[4] = input;
    sort(med_sort, 5);
    med = med_sort[2];
    med1_sort = med2_sort;
    med2_sort = med3_sort;
    med3_sort = med4_sort;
    med4_sort = input;
    med_sort[0] = med1_sort;
    med_sort[1] = med2_sort;
    med_sort[2] = med3_sort;
    med_sort[3] = med4_sort;  }
  return med;}
```

```
void sort(int a[], int size) {
  for (int i = 0; i < (size - 1); i++) {
    for (int o = 0; o < (size - (i + 1)); o++) {
      if (a[o] > a[o + 1]) {
        int t = a[o];
        a[o] = a[o + 1];
        a[o + 1] = t;      }    }  }   }
//function with all filters applied
float general_filter (int counter_avg, int counter_med, float input) {
float input_movavg = mov_avg(counter_avg, input);    //Moving Avg application
float input_med = median(counter_med, input_movavg); //Median application
float input_filtered = kalman_filter(input_med);     //Kalman application
  return input_filtered;}
```

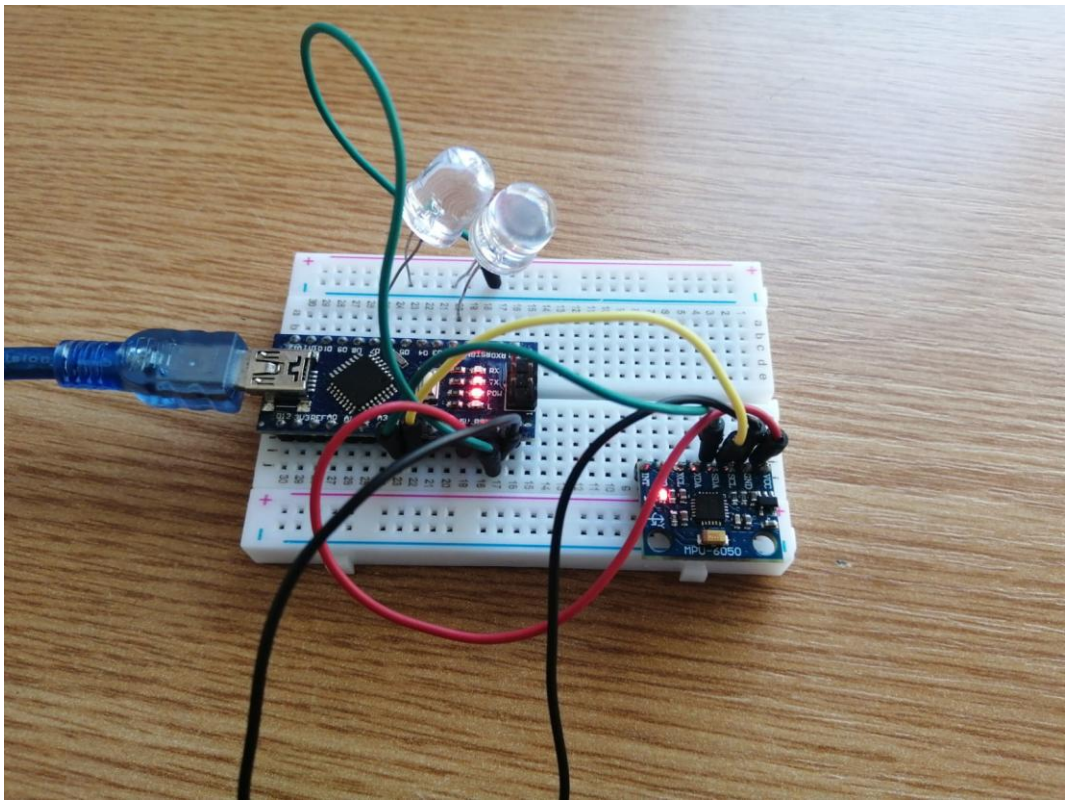


Photo: Final design circuit

4_PROJECT NAME: LINEFOLLOWER (AUSTRIA)

What a Linefollower is:

The line follower robot is a mobile machine that can detect and follow the line drawn on the floor. Generally, the path is predefined and can be either visible like a black line on a white surface with a high contrasted color. Definitely, this kind of robot should sense the line with its infrared ray (IR) sensors that installed under the robot. After that, the data is transmitted to the processor. Hence, the processor is going to decide the proper commends and then it sends them to the driver and thus the path will be followed by the line follower robot.

The important point of building a line follower robot is a good control that is sufficient to follow the path as fast as possible.

Working of Line Follower Robot

The concept of the line follower robot is related to light. Here, we use the behaviour of light on the black and white surface. The white colour reflects all the light that falls on it, whereas the black colour absorbs the light.

In this line follower robot, we use IR transmitters and receivers (photodiodes). They are used to send and receive the lights. When IR rays fall on a white surface, it is reflected towards IR receiver, generating some voltage changes.

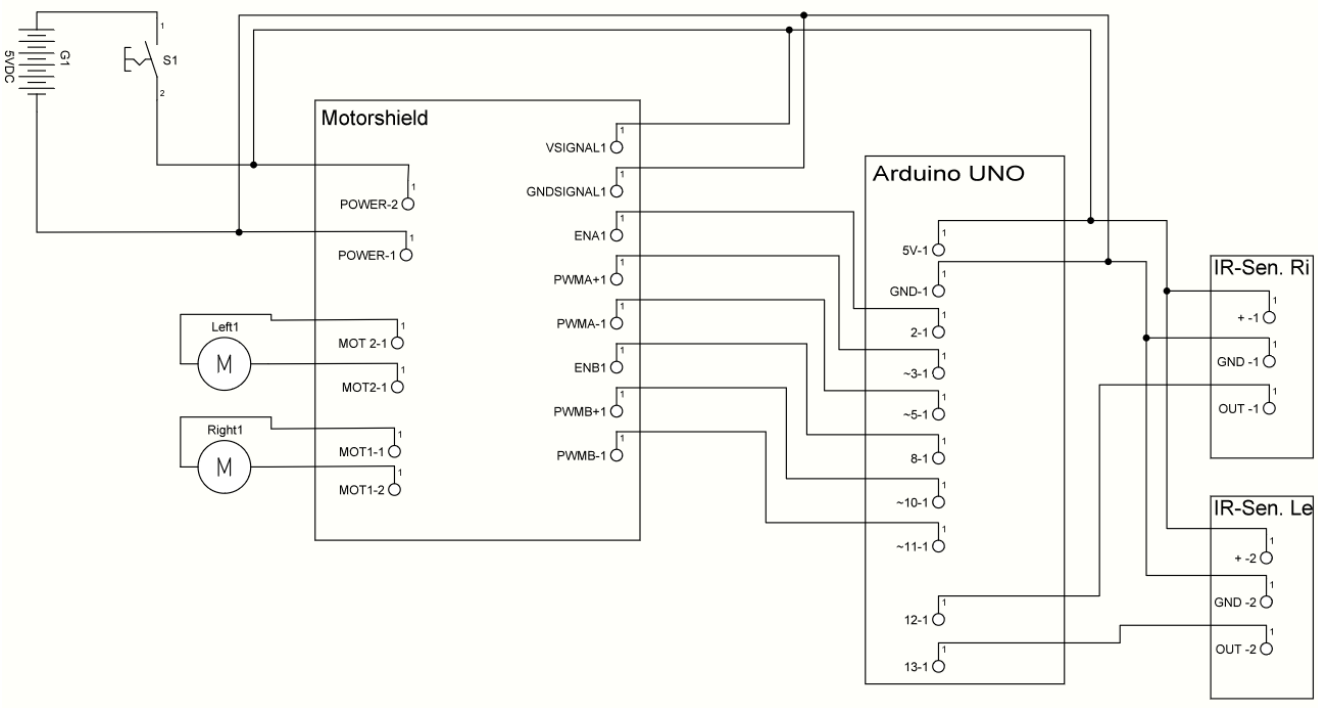
When IR rays fall on a black surface, it is absorbed by the black surface, and no rays are reflected; thus, the IR receiver doesn't receive any rays.

In this project, when the IR sensor senses a white surface, an Arduino gets 1 (HIGH) as input, and when it senses a black line, an Arduino gets 0 (LOW) as input. Based on these inputs, an Arduino Uno provides the proper output to control the bot.

Partlist:

parts	type	pieces
Arduino	Uno	1
Motorshield	by HTL Wolfsberg	1
DC-Motor	COM-Motor01, 3-9Volt DC, joy-it	2
PowerBank	Flip12 PhonePowerBank, 3350mAh	1
IR-Sensors	SEN-KY032IR, joy-it	2
Mainswitch		1
Jumperwire	RB-CB3-025, joy-it	1
USB Cable	Digitus,AK-300102-010-S, Typ A-B	s

Schematic circuit diagram:



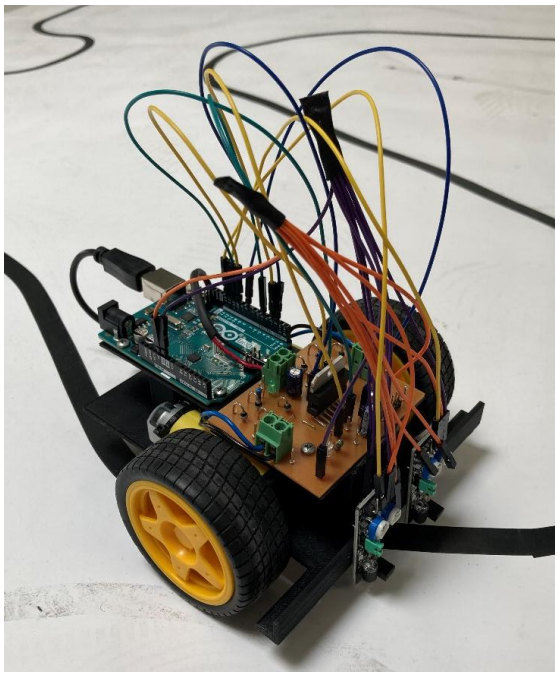


Photo: Final design circuit

Program:

```
// Linefollower
const byte sr = 12;
const byte sl = 13;
const byte enr = 2;
const byte enl = 8;
const byte mrf = 3;
const byte mrb = 5;
const byte mlf = 10;
const byte mlb = 11;

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

  pinMode(sr,INPUT);
  pinMode(sl,INPUT);

  pinMode(enr,OUTPUT);
  pinMode(enl,OUTPUT);
  digitalWrite(enr,HIGH);
  digitalWrite(enl,HIGH);

  pinMode(mrf,OUTPUT);
  pinMode(mrb,OUTPUT);
  pinMode(mlf,OUTPUT);
  pinMode(mlb,OUTPUT);

  analogWrite(mrf,0);
  analogWrite(mrb,0);
  analogWrite(mlf,0);
  analogWrite(mlb,0);
}
  bool senR;
  bool senL;

void loop()
{
  senR = digitalRead(sr);
  senL = digitalRead(sl);

  if(!senR && !senL) //staight
  {
    analogWrite(mrf,220);
    analogWrite(mrb,0);
    analogWrite(mlf,200);
    analogWrite(mlb,0);
```

```
  }

  //right sensor on line
  //drive to right
  else if(senR && !senL)
  {
    analogWrite(mrf,0);
    analogWrite(mrb,100);
    analogWrite(mlf,255);//255
    analogWrite(mlb,0);
  }

  //left sensor on line
  //drive to left
  else if(!senR && senL)
  {
    analogWrite(mrf,255);//255
    analogWrite(mrb,0);
    analogWrite(mlf,0);
    analogWrite(mlb,100);
  }

  // both sensors on black
  // stop
  else if(senR && senL)
  {
    analogWrite(mrf,0);
    analogWrite(mrb,0);
    analogWrite(mlf,0);
    analogWrite(mlb,0);
    delay(3000);
  }
}
```

5_PROJECT NAME: AUTOMATED GREEN HOUSE (ITALY)

The Aim of the Project: Why an automated greenhouse project at school?

Students at school, followed various paths on active citizenship and on the UN Agenda 2030 for Sustainable Development, about the energy sustainability, the importance of avoiding water wastage, and the need to develop innovative and environmentally friendly cultivation methods, considering the profound climate changes taking place. The young people's ecological awareness is expressed in the search for technical solutions and innovative ideas, starting from the theoretical and laboratory knowledge acquired at school inherent in the application opportunities of Arduino.

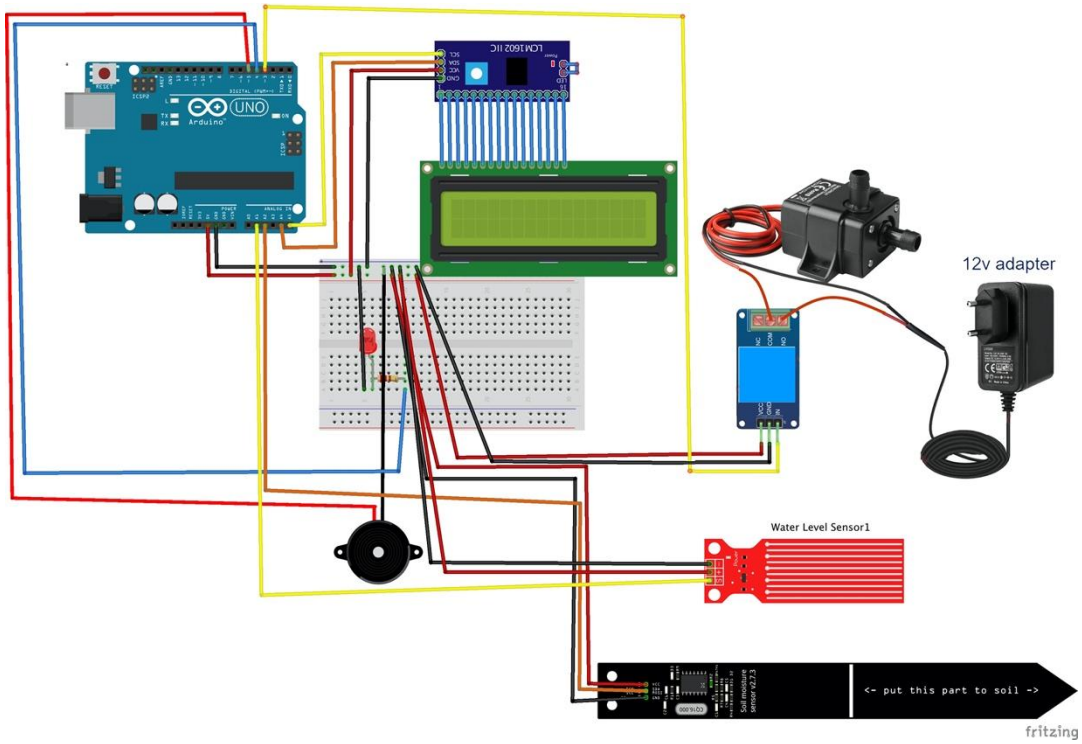
Some students supported by the teachers, designed a system for sensing soil and air temperature, soil moisture, to manage the irrigation and ventilation of a prototype greenhouse suitable for the cultivation of local fruits and vegetables.

The experimentation yielded the hoped-for results and will contribute to the actual implementation in the area in the vicinity of the school intended to host the future school vegetable garden.

Our Project Method

First, a list of all the parameters that we want to use to operate the greenhouse has been prepared. Starting from this a first version of the project algorithm was prepared. According to this algorithm, the necessary sensors were purchased and the logic project was realized. Arduino is used in circuit design. Finally, the final script was implemented. After checking the correct setting of the threshold values of the various sensors, they were positioned inside the greenhouse.

Project Circuit



How It Works:

An LCD display provides the values of temperature and humidity of the air, soil humidity and quantity of water present in the irrigation container. when the values are below a certain threshold, a motorized pump controlled by a relay starts irrigation of the soil. when the water level is below a certain threshold, the display shows a refill warning accompanied by an audible alarm.

List of Materials:

A greenhouse, an Arduino Uno R3, a temperature/humidity DHT11 sensor, a buzzer, a soil humidity sensor, a water level sensor, a red led, a 12V power adapter , a 12V pump, a 220 ohm resistor, a relay, an LCD display.

Arduino Program of Project:

//Project Name: AUTOMATED GREENHOUSE:

```
#include <LiquidCrystal_I2C.h>

#include <Wire.h>

#include <DHT.h>

#define DHTPIN 11

#define DHTTYPE DHT11 // DHT 11

DHT dht(DHTPIN, DHTTYPE);

LiquidCrystal_I2C lcd(0x27,20,4);  // set the LCD address to 0x27 for a 16 chars and 2 line display

int t, h, Lumen, lumin, water, igro, umdtr, wlevel;


const int pin_water = A1;

const int pin_igro = A2;

const int pin_pump = 3;

const int pin_led = 4;

const int pin_buzzer = 5;


void setup() {

   Serial.begin(9600);


   dht.begin();

   lcd.init();

   lcd.backlight();

   lcd.setCursor(5,0);

   lcd.print("School");

   lcd.setCursor(3,1);

   lcd.print("Greenhouse");

   delay (5000);
```

```
    lcd.clear();
    lcd.setCursor(2,0);
    lcd.print("IIS Einstein");
    lcd.setCursor(3,1);
    lcd.print("De Lorenzo");
    delay (5000);
    lcd.clear();

    pinMode(pin_pump,OUTPUT);
    pinMode(pin_led, OUTPUT);
    pinMode(pin_buzzer, OUTPUT);
    digitalWrite(pin_pump, HIGH);
}

void loop() {
    // water level
    water = analogRead (pin_water);
    wlevel = map (water,0, 1023, 0, 100);
    if(wlevel < 35){
        lcd.clear();
        tone(pin_buzzer, 800);
        digitalWrite(pin_led, HIGH);
        delay(300);
        noTone(pin_buzzer);
        digitalWrite(pin_led, LOW);
        delay(300);
        lcd.clear();
        //lcd.begin(16, 2);
```

```
    lcd.setCursor(2,0);
    lcd.print("Refill Water");
    delay (1000);
    lcd.clear();
} else {
    noTone(pin_buzzer);
    digitalWrite(pin_led, LOW);
}

// Igrometer
igro = analogRead(pin_igro);
umdtr = map (igro, 0, 1023, 0, 100);

if(umdtr < 45){
    digitalWrite(pin_pump, LOW);
    delay(10000);
    digitalWrite(pin_pump, HIGH);
}

// Lettura umidità e temperatura del sensore DHT11
h = dht.readHumidity();
t = dht.readTemperature();

// posiziona il cursore in colonna 0 e linea 1
// (nota: la linea 1 e la seconda linea, poichè si conta incominciando da 0):
lcd.setCursor(0, 0);
lcd.print("T:");
lcd.print(t);
```

```
    lcd.print( (char) 223 );
    lcd.print("C");

    lcd.setCursor(10, 0);
    lcd.print("H:");
    lcd.print(h);
    lcd.print("%");

    lcd.setCursor(0, 1);
    lcd.print("SH:");
    lcd.print(umdtr);
    lcd.print("%");

    lcd.setCursor(10, 1);
    lcd.print("WL:");
    lcd.print(wlevel);
    lcd.print("%");
}
```

Photo: Final design

Annexex

Arduino's mission is to enable anyone to enhance their lives through accessible electronics and digital technologies. There was once a barrier between the electronics, design, and programming world and the rest of the world. Arduino has broken down that barrier. For more information about the Arduinos, you can visit the website, below…

www.arduino.cc/

For more information about our ARDinVET project, you can visit the website, below…

www.arduinvet.com

“This project is Funded by the Erasmus+ Program of the European Union. However, European Commission and Turkish National Agency cannot be held responsible for any use which may be made of the information contained therein”