A
VOCATIONAL TRAINING REPORT
ON
EMBEDDED SYSTEM WITH ARDUINO
Submitted to
CHHATTISGARH SWAMI VIVEKANAND TECHNICAL UNIVERSITY
BHILAI

In partial fullfillment of requirement for the award of degree
Of
Bachelor of Technology
In
Electronics & Telecommunication Engineering
BY
Abhishek Chatterjee

Department of Electronics& Telecommunication Engineering
GOVERNMENT ENGINEERING COLLEGE
Old Dhamtari Road, Sejbahar ,Raipur
SESSION 2022-23

DECLARATION

I am undersigned solemnly declare that the report of the project work entitled "Embedded system with Arduino" is based my own work carried out during the course of my study under the supervision of Dr. Saji.
I assert that the statements made and conclusions drawn are an outcome of the project work. I further declare that to the best of my knowledge and belief that the report does not contain any part of my work which has been submitted for the award of any other degree certificate in this University or any other University.

SHUBUHAN AMINA
301602821304
Date:

Department of Electronics& Telecommunication Engineering
GOVERNMENT ENGINEERING COLLEGE

CERTIFICATE

This is to certify that the report of the project submitted is an outcome of the project work title "**Alcohol Detector Working"** carried out by **Shubuhan Amina** bearing **301602821304 & BF6720** carried out under my guidance and supervision for award of Degree in bachelor of Technology in **Electronics and Telecommunication** of Chhattisgarh Swami Vivekanand Technical University, Bhilai C.G.

(Signature Of The Guide)

Project work as mentioned above is hereby being recommended and forwarded for examination & evaluation.

(Signature of Head of the Department with seal)

Department of Electronics& Telecommunication Engineering
GOVERNMENT ENGINEERING COLLEGE

CERTIFICATE

Registration No. ESA-070222-005

Chhattisgarh Swami Vivekanand Technical University (CSVTU), Bhilai

(Centre for Skill Development and Informal Education)

Online Training Certificate

This is to certify that
Shubuhan Amina
D/O Mr. Shamsher Ahmed
has successfully completed the
Online Training in *Embedded System with Arduino* organized from
11.07.2022 to 10.08.2022
with B+ Grade.

Coordinator CSDIE
CSVTU, Bhilai (CG)

Bhilai (C.G.), September, 2022

Registrar
CSVTU, Bhilai (CG)

Department of Electronics& Telecommunication Engineering
GOVERNMENT ENGINEERING COLLEGE

CERTIFICATE BY THE EXAMINER

This is to certify that the project work entitled
(Title of the project)
Submitted by
<Name: Shubuhan Amina> Roll No.: 301602821304..Enrollment No. BF6720
<Name: Abhishek Chatterjee> Roll No.: 301602820004.Enrollment No. BK0154

Has been examined by the undersigned as a part of the examination for the award of Bachelor of Technology Degree in ET& T of Chhatisgarh Swami Vivekanand Technical University, Bhilai.

Internal Examiner
Date:

External Examiner
Date:

Department of Electronics& Telecommunication Engineering
GOVERNMENT ENGINEERING COLLEGE

ACKNOWLEDGEMENT

I would like to express my truthful gratitude to the Functional Coordinator, **Dr. Saji** who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path. Next, my deepest gratitude and special thanks to professor for guiding and keeping me on the correct path and allowing me to carry out my internship at their esteemed organization. Thank you for her motivation, support and kindly share his huge knowledge for developing my internship. I am very glad to have a superior advisor for my Industrial Training at tinker Circuit .I express my deepest thanks to Dr. saji and Mr. Vishal for giving me necessary advices and guidance and to make life easier throughout my internship. Last but not least, I would like to thanks to Faculty **CSDIE** staff and my lecturer on their contribution in providing help to completed the industrial training.

Department of Electronics& Telecommunication Engineering
GOVERNMENT ENGINEERING COLLEGE

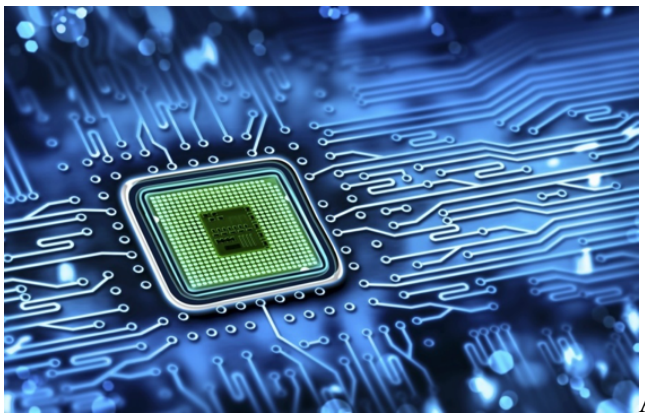
TABLE OF CONTENTS PAGE NO.

Chapter 1

INTRODUCTION

MICROCONTROLLER

What is Microcontrollers?



A **microcontroller** is a small and low-cost microcomputer, which is designed to perform the specific tasks of embedded systems like displaying microwave's information, receiving remote signals, etc.

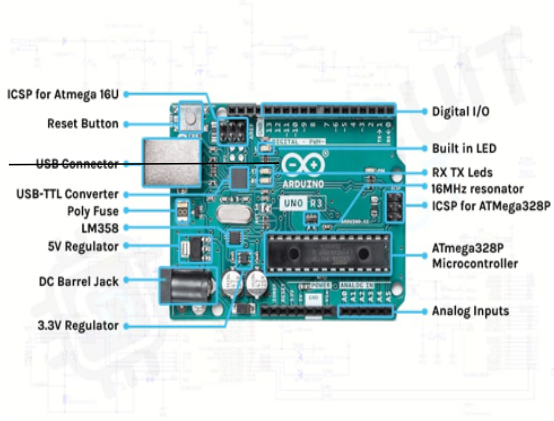
- Microcontrollers are used to execute a single task within an application.

- Its designing and hardware cost is low. Easy to replace.
- It is built with CMOS technology, which requires less power to operate.
- It consists of CPU, RAM, ROM, I/O ports.
- Peripherals such as clock , timer ,PWM ,etc
- Microprocessors are used for general purpose applications,while microcontroller are self sufficient and are used for specific task.
- Microcontrollers are an example of embedded systems.

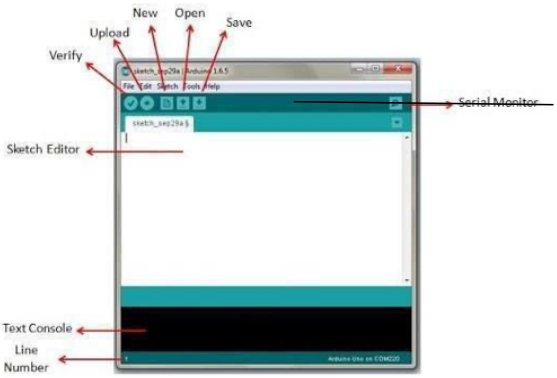
ARDUINO

ARDUINO IS AN OPEN SOURCE ELECTRONICS PLATFORM BASED ON EASY-TO-USE HARDWARE AND SOFTWARE.

HARDWARE



SOFTWARE



What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

Who Develop Arduino?

(back) Dave Mellis, Tom Igoe; (front) Gianluca Martino, David Cuartielles, Massimo Banzi

David Mellis developed the Arduino software, which was based on Wiring. Before long, Gianluca Martino and Tom Igoe joined the project, and the five are known as the original founders of Arduino.

Why Arduino Board?

Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux.

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms.
- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems.
- **Simple, clear programming environment** - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well.
- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers.
- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it.

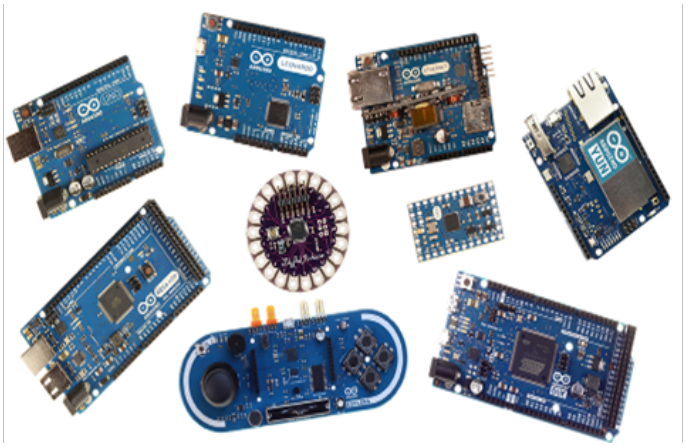
CHAPTER 2

ARDUINO AND ITS TYPES

Different Types Of Arduino Boards

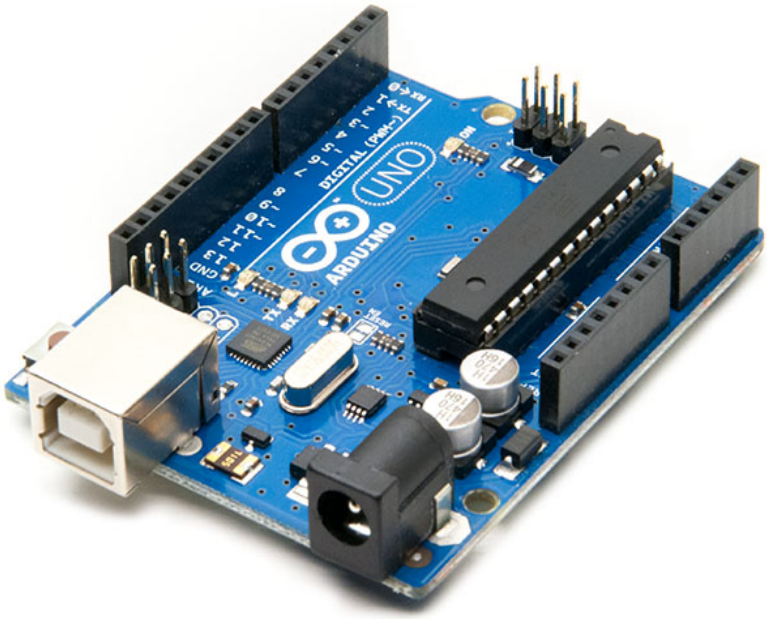
The list of Arduino boards includes the following such as

- Arduino Uno (R3)
- Arduino Nano
- Arduino Micro
- Arduino Due
- LilyPad Arduino Board
- Arduino Bluetooth
- Arduino Diecimila
- RedBoard Arduino Board
- Arduino Mega (R3) Board
- Arduino Leonardo Board
- Arduino Robot
- Arduino Esplora
- Arduino Pro Mic
- Arduino Ethernet
- Arduino Zero
- Fastest Arduino Board



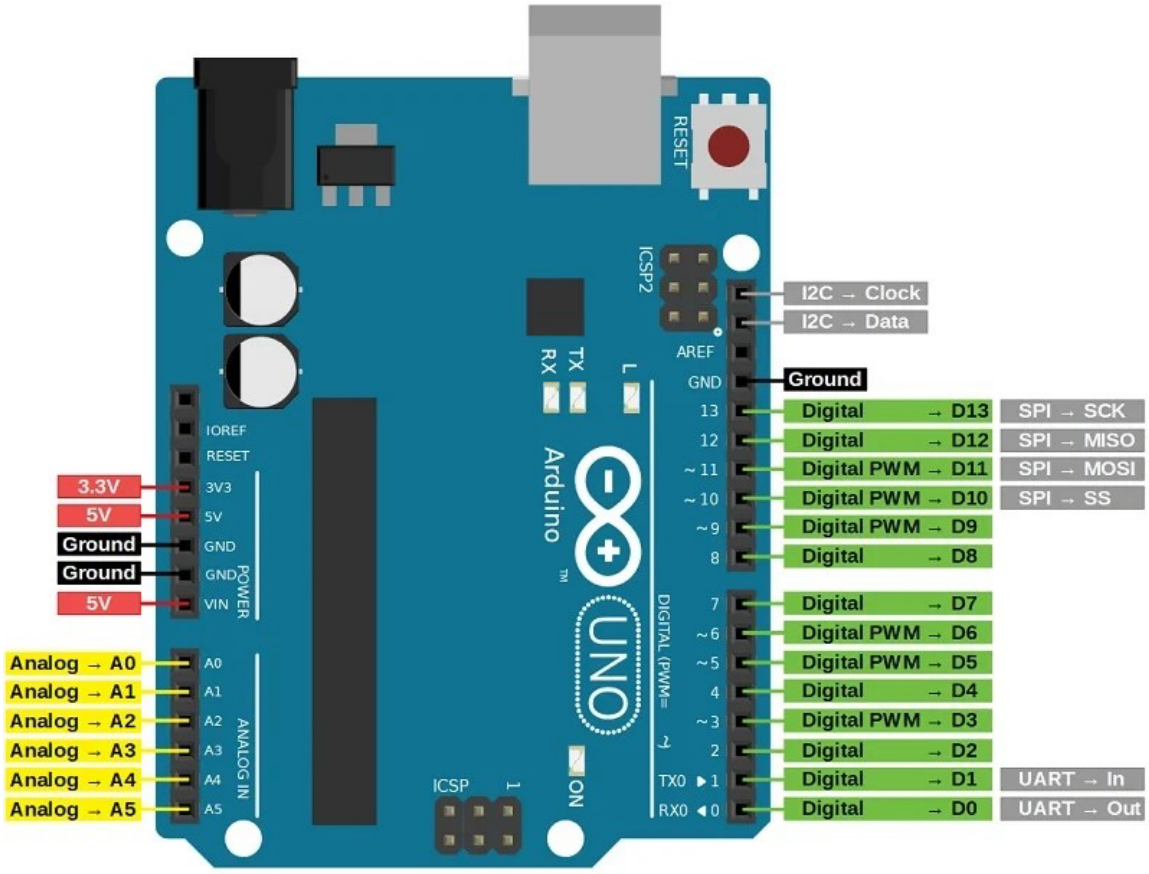
ARDUINO UNO (R3)

Definition



The Uno is a huge option for your initial Arduino. This Arduino board depends on an ATmega328P based microcontroller. As compared with other types of arduino boards, it is very simple to use like the Arduino Mega type board. .It consists of 14-digital I/O pins, where 6-pins can be used as PWM(pulse width modulation outputs), 6-analog inputs, a reset button, a power jack, a USB connection, an In-Circuit Serial Programming header (ICSP), etc. It includes everything required to hold up the microcontroller; simply attach it to a PC with the help of a USB cable and give the supply to get started with an AC-to-DC adapter or battery.

Arduino Uno is the most frequently used board and it is the standard form apart from all the existing Arduino Boards. This board is very useful for beginners. Please refer to this link to know more about Arduino Uno Board.We can directly connect the board to the computer via a USB Cable which performs the function of supplying the power as well as acting as a serial port.



Explain The Pins Of Arduino

Vin: This is the input voltage pin of the Arduino board used to provide input supply from an external power source.

5V: This pin of the Arduino board is used as a regulated power supply voltage and it is used to give supply to the board as well as onboard components.

3.3V: This pin of the board is used to provide a supply of 3.3V which is generated from a voltage regulator on the board

GND: This pin of the board is used to ground the Arduino board.

Reset: This pin of the board is used to reset the microcontroller. It is used to Resets the microcontroller.

Analog Pins: The pins A0 to A5 are used as an analog input and it is in the range of 0-5V.

Digital Pins: The pins 0 to 13 are used as a digital input or output for the Arduino board.

Serial Pins: These pins are also known as a UART pin. It is used for communication between the Arduino board and a computer or other devices. The transmitter pin number 1 and receiver pin number 0 is used to transmit and receive the data resp.

External Interrupt Pins: This pin of the Arduino board is used to produce the External interrupt and it is done by pin numbers 2 and 3.

PWM Pins: This pins of the board is used to convert the digital signal into an analog by varying the width of the Pulse. The pin numbers 3,5,6,9,10 and 11 are used as a PWM pin.

SPI Pins: This is the Serial Peripheral Interface pin, it is used to maintain SPI communication with the help of the SPI library. SPI pins include:

1. SS: Pin number 10 is used as a Slave Select
2. MOSI: Pin number 11 is used as a Master Out Slave In
3. MISO: Pin number 12 is used as a Master In Slave Out
4. SCK: Pin number 13 is used as a Serial Clock

LED Pin: The board has an inbuilt LED using digital pin-13. The LED glows only when the digital pin becomes high.

AREF Pin: This is an analog reference pin of the Arduino board. It is used to provide a reference voltage from an external power supply.

Programming

- The arduino programming language (based on wiring) ,and the arduino software (IDE) based on processing language running on the computer .
- The IDE enables to write sketches (a little program to tell the board what to do)
- The sketch is uploaded to the board through the USB connection.

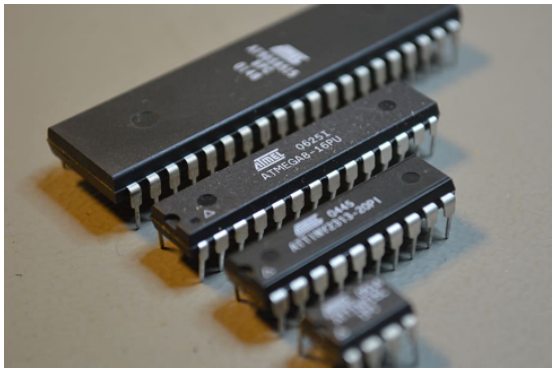
DATA SHEET OF ATMEGA328 MICROCONTROLLER

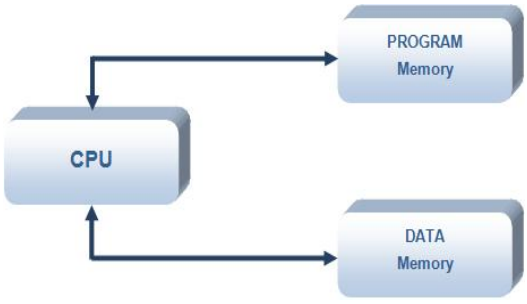
MICROCONTROLLER	ATMEGA328P
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	14(OF WHICH 6 PROVIDE PWM OUTPUT)
PWM DIGITAL I/O PINS	6
ANALOG INPUT PINS	6
DC CURRENT PER I/O PIN	20MA
DC CURRENT FOR 3.3 V PIN	50MA
FLASH MEMORY	32KB(ATMEGA32P)OF WHICH 0.5KB USED BY BOOTLOADER
SRAM	2KB(ATMEGA328P)
EEPROM	1KB
CLOCK SPEED LED_BULITIN	16MHZ 13
LENGTH	68.6MM
WIDTH	53.4MM
WEIGHT	25G

AVR FAMILY

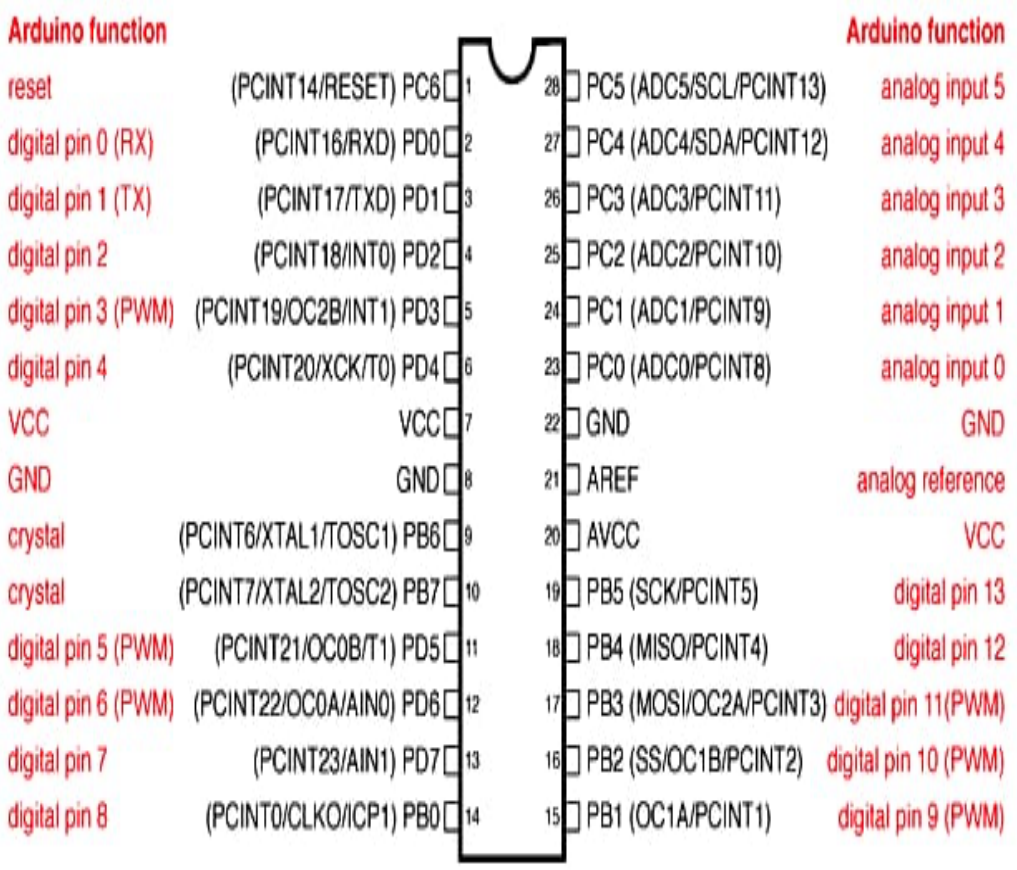
ATMEL (MICROCHIPS)

SERIES NAME	PINS	FLASH MEMORY	SPECIAL FEATURE
TINY AVR SERIES	6-32	0.5-8 KB	Small in size
MEGA AVR SERIES	28-100	4-256 KB	Extended peripherals
X MEGA AVR SERIES	44-100	16-384 KB	DMA , Event system included





ATMega328P and Arduino Uno Pin Mapping



Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

CHAPTER 3

ADVANTAGE AND DISADVANTAGES OF ARDUINO

Advantage Of Arduino

- Not much knowledge required to get started.
- Fairly low cost, depending on shields you need.
- Lots of sketches and shields available.
- No external programmer or power supply needed.

Disadvantage Of Arduino

- Cannot run more than one program at the same time.
- No memory safety checks.
- Expensive for the CPU power and memory.
- Lack of built-in communications.
- Lack of built-in peherals.
- Limited number of programming languages.
- Limited number of IDEs.

Application Of Arduino

- Weighing Machines.
- Traffic Light Count Down Timer.
- Parking Lot Counter.
- Embedded systems.
- Home Automation.
- Industrial Automation.
- Medical Instrument.
- Emergency Light for Railways.

CHAPTER 4

ARDUINO PROGRAMMING

INTRODUCTION TO THE ARDUINO PROGRAMMING LANGUAGE

The Arduino Programming Language is basically a framework built on top of C++. You can argue that it's not a real programming language in the traditional term, but I think this helps avoiding confusion for beginners.
A program written in the Arduino Programming Language is called **sketch** A sketch is normally saved with the .ino extension (from Arduino).
The main difference from "normal" C or C++ is that you wrap all your code into 2 main

functions. You can have more than 2, of course, but any Arduino program must provide at least those 2.
One is called setup(), the other is called loop(). The first is called once, when the program starts, the second is repeatedly called while your program is running.
We don't have a main() function like you are used to in C/C++ as the entry point for a program. Once you compile your sketch, the IDE will make sure the end result is a correct C++ program and will basically add the missing glue by preprocessing it.
Everything else is normal C++ code, and as C++ is a superset of C, any valid C is also valid Arduino code.
One difference that might cause you troubles is that while you can spawn your program over multiple files, those files must all be in the same folder. Might be a deal breaking limitation if your program will grow very large, but at that point it will be easy to move to a native C++ setup, which is possible.
Part of the Arduino Programming Language is the built-in libraries that allow you to easily integrate with the functionality provided by the Arduino board.
Your first Arduino program will surely involve making a led turn on the light, and then turn off. To do so, you will use the pinMode(), delay() and digitalWrite() functions, along with some constants like HIGH, LOW, OUTPUT.
Like this, the canonical first Arduino project (the "Hello, World!"):

```
#define LED_PIN 13

void setup() {
// Configure pin 13 to be a digital output
pinMode(LED_PIN, OUTPUT);
}

void loop() {
// Turn on the LED
digitalWrite(LED_PIN, HIGH);
// Wait 1 second (1000 milliseconds)
delay(1000);
// Turn off the LED
digitalWrite(LED_PIN, LOW);
// Wait 1 second
delay(1000);
}
```

This is all part of the Arduino Programming Language, or we'd better call it *suite* or *library*.

SUPPORT FOR OTHER LANGUAGE

As a reminder, I want to note that you are not limited to using this language and IDE to program an Arduino. Projects exist, among others, to let you run Node.js code on it using the Johnny Five project, Python code using pyserial and Go code with Gobot, but the Arduino Programming Language is definitely the one you'll see most tutorials based upon, since it's the native and canonical way to work with these devices.

The Arduino Programming Language Built-in constants
Arduino sets two constants we can use toHIGH equates to a high level of voltage, which can

differ depending on the hardware (>2V on 3.3V boards like Arduino Nano, >3V on 5V boards like Arduino Uno) LOW equates to a low level of voltage. Again, the exact value depends on the board usedThen we have 3 constants we can use in combination with the pinMode() function:

- INPUT sets the pin as an input pin
- OUTPUT sets the pin as an output pin
- INPUT_PULLUP sets the pin as an internal pull-up resistor
- The other constant we have is LED_BUILTIN, which points to the number of the on-board pin, which usually equates to the number 13.
- In addition to this, we have the C/C++ constants true and false.

Arduino Math Constants

- M_PI the constant pi (3.14159265358979323846)
- M_E the constant e
- M_LN10 the natural logarithm of the number 10.
- M_LN2 the natural logarithm of the number 2.
- M_LOG10E the logarithm of the e to base 10.
- M_LOG2E the logarithm of the e to base 2.

The Arduino Programming Language Built-in Functions

In this section I am going to make a reference for the built-in functions provided by the Arduino Programming Language.

Program lifecycle

- setup() this function is called once, when the program starts, and when the Arduino is shut down and restarted.
- loop() this function is repeatedly called while the Arduino program is running.

Handling I/O

• The following functions help with handling input and output from your Arduino device.

Digital I/O

- digitalRead() reads the value from a digital pin. Accepts a pin number as a parameter, and returns the HIGH or LOW constant.
- digitalWrite() writes a HIGH or LOW value to a digital output pin. You pass the pin number and HIGH or LOW as parameters.
- pinMode() sets a pin to be an input, or an output. You pass the pin number and the INPUT or OUTPUT value as parameters.

- pulseIn() reads a digital pulse from LOW to HIGH and then to LOW again, or from HIGH to LOW and to HIGH again on a pin. The program will block until the pulse is detected. You specify the pin number and the kind of pulse you want to detect (LHL or HLH). You can specify an optional timeout to stop waiting for that pulse.

- pulseInLong() is same as pulseIn(), except it is implemented differently and it can't be used if interrupts are turned off. Interrupts are commonly turned off to get a more accurate result.

- shiftIn() reads a byte of data one bit at a time from a pin.

- shiftOut() writes a byte of data one bit at a time to a pin.

- tone() sends a square wave on a pin, used for buzzers/speakers to play tones. You can specify the pin, and the frequency. It works on both digital and analog pins.

Analog I/O

- analogRead() reads the value from an analog pin.

- analogReference() configures the value used for the top input range in the analog input, by default 5V in 5V boards and 3.3V in 3.3V boards.

- analogWrite() writes an analog value to a pin

- analogReadResolution() lets you change the default analog bits resolution for analogRead(), by default 10 bits. Only works on specific devices (Arduino Due, Zero and MKR)

- analogWriteResolution() lets you change the default analog bits resolution for analogWrite(), by default 10 bits. Only works on specific devices (Arduino Due, Zero and MKR)

Time functions

- delay() pauses the program for a number of milliseconds specified as parameter.
- delayMicroseconds() pauses the program for a number of microseconds specified as parameter.

- micros() the number of microseconds since the start of the program. Resets after ~70 minutes due to overflow.

- millis() the number of milliseconds since the start of the program. Resets after ~50 days due to overflow.

Math functions

- abs() the absolute value of a number.

- constrain() constrains a number to be within a range, see usage.

- map() re-maps a number from one range to another, see usage
- max() the maximum of two numbers
- min() the minimum of two numbers
- pow() the value of a number raised to a power
- sq() the square of a number
- sqrt() the square root of a number
- cos() the cosine of an angle
- sin() the sine of an angle
- tan() the tangent of an angle

Working With Alphanumeric Characters

- isAlpha() checks if a char is alpha (a letter)
- isAlphaNumeric() checks if a char is alphanumeric (a letter or number)
- isAscii() checks if a char is an ASCII character
- isControl() checks if a char is a control character
- isDigit() checks if a char is a number
- isGraph() checks if a char is a printable ASCII character, and contains content (it is not a space, for example)
- isHexadecimalDigit() checks if a char is an hexadecimal digit (A-F 0-9)
- isLowerCase() checks if a char is a letter in lower case
- isPrintable() checks if a char is a printable ASCII character
- isPunct() checks if a char is a punctuation (a comma, a semicolon, an exclamation mark etc)
- isSpace() checks if a char is a space, form feed \f, newline \n, carriage return \r, horizontal tab \t, or vertical tab \v.
- isUpperCase() checks if a char is a letter in upper case

- isWhitespace() checks if a char is a space character or an horizontal tab \t

Random numbers generation

- random() generate a pseudo-random number

- randomSeed() initialize the pseudo-random number generator with an arbitrary initial number.

In Arduino, like in most languages, it's impossible to get really random numbers, and the sequence is always the same, so you seed it with the current time or (in the case of Arduino) you can read the input from an analog port.

Working with bits and bytes

- bit() computes the value of a bit (0 = 1, 1 = 2, 2 = 4, 3 = 8…)

- bitClear() clear (sets to 0) a bit of a numeric variable. Accepts a number, and the number of the bit starting from the right

- bitRead() read a bit of a number. Accepts a number, and the number of the bit starting from the right

- bitSet() sets to 1 a bit of a number. Accepts a number, and the number of the bit starting from the right

- bitWrite() write 1 or 0 to a specific bit of a number Accepts a number, the number of the bit starting from the right, and the value to write (0 or 1)

- highByte() get the high-order (leftmost) byte of a word variable (which has 2 bytes)

- lowByte() get the low-order (rightmost) byte of a word variable (which has 2 bytes)

Interrupts

- noInterrupts() disables interrupts

- interrupts() re-enables interrupts after they've been disabled

- attachInterrupt() allow a digital input pin to be an interrupt. Different boards have different allowed pins, check the official docs.

- detachInterrupt() disables an interrupt enabled using attachInterrupt()

Bottom of Form

Blink An Led With Arduino In Tinkercad

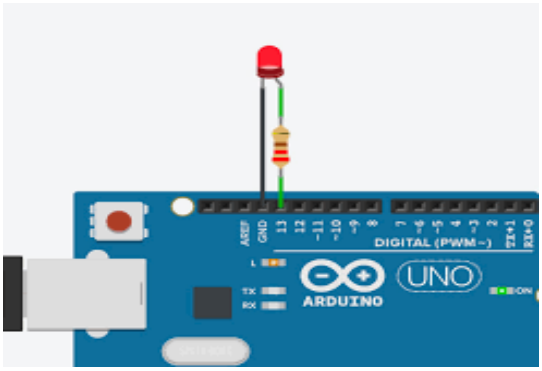
by circuit

Let's learn how to blink an LED (light emitting diode)using Arduino's digital output. If you're new to Arduino, this is a great place to start. We'll connect anLED to the Arduino Uno and compose a simple
program to turn the LED on and off.You can follow along virtually using Tinker cadCircuits.

You can even view this lesson from withinTinker cad if you like! Explore the sample circuit andbuild your own right next to it! Click "Start Simulation"to watch the LED blink. You can use the simulatorany time to test your circuits. Tinker cad Circuits is afree browser-based program that lets you build andsimulate circuits. It's perfect for learning, teaching,and prototyping.Find this circuit onTinker cadIf you want to follow along with your physical ArduinoUno (or compatible) board, you'll also need a USBcable and a computerwith the free Arduino software(it or plugin for the web editor) installed, and optionally single LED.

Step 1: LED Resistor Circuit

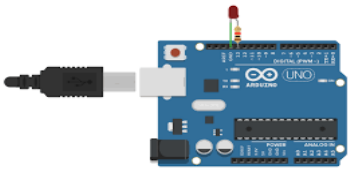
The LED's legs are connected to two pins on theArduino: ground and
pin 13. The component betweenthe LED and pin 13 is a resistor, which helps limit thecurrent to prevent the LED from burning itself out.
Without it, you'll get a warning that the LED mightburn out soon. It doesn't matter whether the resistorcomes before or after the LED inthe circuit, or whichway around it goes. The colored stripes identifytheresistor's value, and for this circuit, anywhere from100 ohms to 1000 ohms will work great.The LED, on the other hand, is polarized, whichmeans it only works when the legs are connected acertain way. The positive leg, called the anode,usually has a longer leg, and gets wired to power, inthis case coming from your Arduino's output pin. Thenegative leg, called the cathode, with its shorter leg,connects to ground.In the Tinkercad Circuits components panel, drag aresistor and LED onto the workplane. Edit theresistor's value by adjusting it to 220 ohms in thecomponent inspector which appears when theresistor is selected. Back in the components panel,find and bring over an ArduinoUno board. Click onceto connect a wire to a component or pin, and clickagain to connect the other end. Connect your resistorto either side of the LED. If you connected yourresistor to the LED's anode (positive, longer), connectthe resistor's other leg to Arduino's digital pin 13. Ifyou connected your resistor to the LED's cathode(negative, shorter leg), connect the resistor's other legto Arduino's ground pin(GND). Create another wirebetween the unconnected LED leg and pin 13 orground, whichever is still not connected.Extra credit: you can learn more about LEDs in thefree Instructables LEDs and Lighting class.
If you have a physical Arduino Uno (or compatible)board, you may plug an LED directly into pin 13(positive, longer leg anode) and ground (negative,shorter cathode), because pin 13 actually has a built-inresistor for exactly this testing purpose.



Step 2: Simple Code With Blocks

In Tinkercad Circuits, you can easily code up yourprojects using blocks. Let's go through the

simplecode controlling the blink by opening the code editor(button labeled "Code"). You can resize the codeeditor by clicking and dragging the left edge.The code starts out with two gray comment blocks,which are just notes for us humans to read. The first
blue output block sets the built-in LED HIGH, which isArduino's way of describing "on." This outputcommand will activate a 5V signal toanythingconnected to the specified pin. Next up is a a yellowcommand block that waits for one second, simpleenough. So theprogram will pause while the LED ison for one second. Next after another comment is ablue output block to set the LED back to LOW, or
"off," followed by another second-long pause.Try customizing this code by changing the wait times,and clicking "Start Simulation". You can even addmore output and wait blocks to create longer flashing
patterns.Did you notice the small LED flashing on the boarditself? This built in LED is also connected to pin 13,and is meant to be used for testing purposes withoutthe need to connect any externalcomponents. It evenhas its own tiny resistor, soldered directly to the
surface of the board. Ready to create your own? Click to select the Arduinoyou added to the workplane (or select it from thedropdown menu in the code editor) and start draggingcode blocks to create your own blinking program.



Step 3: Blink Arduino Code Explained

When the code editor is open, you can click the dropdown menu on the left and select "Blocks + Text" to revealthe Arduino codegenerated by the code blocks. All the extra symbols are part ofArduino's syntax, but don't beintimidated! It takes time to learn to write proper code from scratch. We'll go through each piece here, and you canalways use the blocks for comparison as you level up.

```
/*
This program blinks pin 13 of the Arduino (the
built-in LED)
*/
```

This first section is title block comment, describing what the program does. Block comments are bookended by anBlink an LED With Arduino in Tinkercad: Page 4
opening /* and closing */ .

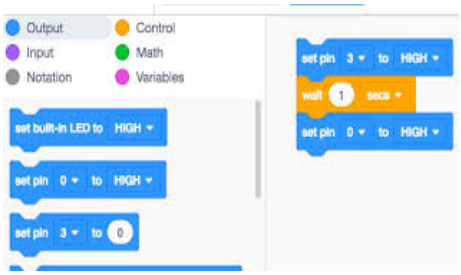
```
void setup()
{
pinMode(13, OUTPUT);
}
```

Next is the code's setup, which helps set up things your program will need later. It runs once when the programstarts up, and contains everything within its curly braces { } . Our blink sketch's setupconfigures pin 13 as anoutput, which prepares the board to send signals to it, rather than listen.

```
void loop()
{
// turn the LED on (HIGH is the voltage level)
digitalWrite(13, HIGH);
delay(1000); // Wait for 1000 millisecond(s)
// turn the LED off by making the voltage LOW
digitalWrite(13, LOW);
delay(1000); // Wait for 1000 millisecond(s)
}
```

The main body of the program is inside the loop, indicated by another set of curly braces { }

. This part of the codewill execute on repeat, so long as the board has power. The colored text following double slashes are alsocomments to help make the program easier to understand. The output command we're using is called digitalWrite() ,which is a function that sets a pin HIGH or LOW, on or off. To pause the program we'll use delay() , which takes anumber of milliseconds (1000ms = 1s)

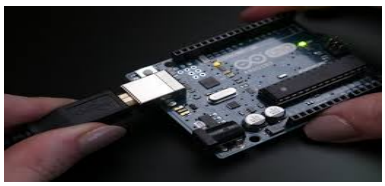


Step 4: Use the Blink Circuit Starter

This is a circuit we think you'll want to makefrequently, so it's saved as a circuit starter!Grab this circuit and code combo any time using the
starter available in the components panel (dropdownmenu -> Starters -> Arduino).For a more advanced version of this Arduino code,also check out the Blink Without Delay starter, whichuses the current time to keep track of blink intervalsinstead of delay();

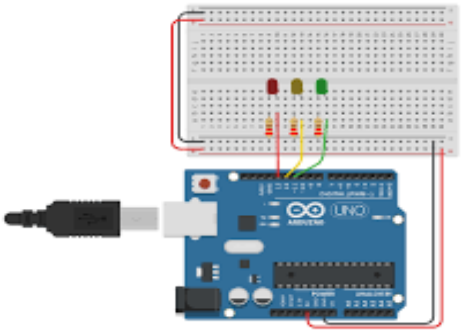
Step 5: Programming a Physical Arduino

To program your physical Arduino Uno, copy thecode from the window and paste it into an emptyArduino sketch, or click the download button andopen the resulting file using your Arduino software.This beginner example is also available directly withinthe Arduino software under File-> Examples->01.Basics-> Blink.Plug in your USB cable and select your board andport in the software's Tools menu.Upload the code and watch your onboard LED flashwith the custom blink you created earlier! For a morein-depth walk-through on setting up and programmingyour physical Arduino Uno board, checkout the freeInstructible Arduino class (first lesson).



Step 6: Next, Try...

Experiment with this simulation by adding moreblocks to createflashing patterns. Can you create aprogram that flashes out a message using MorseCode?Now that you know how to blink an LED using
Arduino's digital output, you're ready to try otherArduino exercises that utilize the digitalWrite() function.Try using a breadboard to add more LEDs and codeto control them in the next Tinkercad Circuits lesson:Multiple LEDs & Breadboards (Tinkercad lessonversion).Youcan also learn more electronics skills with thefree Instructables classes on Arduino, BasicElectronics, LEDs & Lighting, 3D Printing, and more.



Our project

What is MQ4 Methane Gas Sensor?

MQ4 methane gas sensor is a MOS (metal oxide semiconductor) type sensor, used to detect the methane gas concentration within the air at either home or industries & generates output like analog voltage by reading it. Here, the range of concentration for sensing ranges from 300 pm – 10,000 ppm which is appropriate for the detection of a leak.This gas sensor mainly includes a detecting element like ceramic based on aluminum-oxide (Al_2O_3), coated with Tin dioxide (SnO2) and arranged within a stainless-steel mesh.

MQ4 Methane Gas Sensor

When methane gas and detecting elements get in contact with each other then the resistivity of the detecting element will be changed. After that, the change is measured to get the methane gas concentration. The ignition of Methane gas is extremely exothermal which means it generates a huge amount of heat once ignited.

Pin Configuration:

The **pin configuration of the MQ4 methane gas sensor** is shown below. This sensor includes three pins which are discussed below.

MQ4 Methane Gas Sensor Pin Configuration

- **Pin1 (H Pins):** These pins are two where one of them is used to connect supply and remaining pin is connected to ground
- **Pin2 (A Pins):** Both the pins like A & B are interchangeable which will be connected to supply voltage.

- **Pin3 (B Pins):** A & B pins are exchangeable where one pin acts like output and another pin will be pulled to the GND terminal.

The **pin configuration of the MQ-4 methane gas sensor module** includes four pins which are discussed below.

- **VCC Pin:** This pin provides voltage to the module and the typical operating voltage is +5V
- **GND Pin:** This pin is used to connect the sensor module to the GND terminal of the system
- **DO (Digital Out) Pin:** This pin provides digital output by setting a threshold value with the help of the potentiometer
- **AO (Analog Out):** This pin provides output analog voltage which ranges from 0 to 5V depending on the intensity of gas.

Features & Specifications

The **features and specifications of the MQ4 methane gas sensor** include the following.

- Sensitivity is good for combustible gas in an extensive range
- Sigh sensitivity for natural gas, methane gas
- Small sensitivity for smoke and alcohol
- Quick responsive, long life and stable
- The drive circuit is simple
- Load resistance is 20KΩ
- Detecting resistance ranges from 10KΩ to 60KΩ

what is an Alcohol detector using Arduino

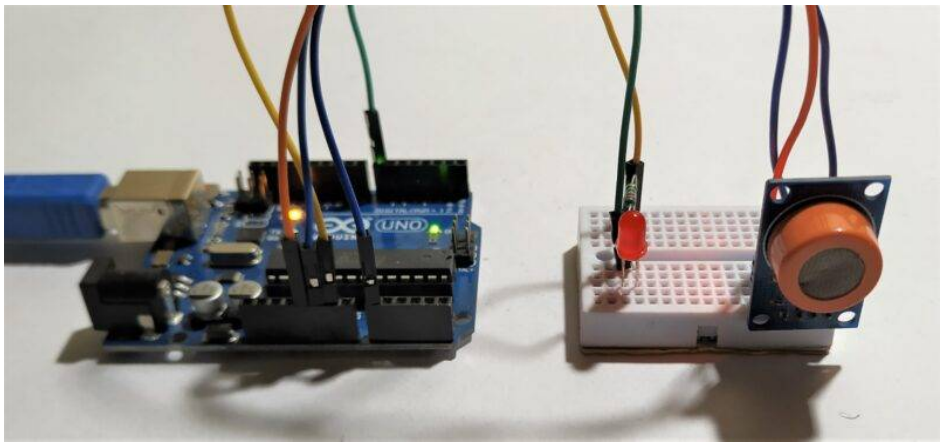
The stuff we are going to used today to build an alcohol indicator is Arduino, LEDs, and an MQ-3 Alcohol sensor. There are many MQ-X sensors is available in the market for different uses but we are here going to use MQ-3 because it is best for detecting alcohol. The working of most of the MQ sensors is the same. They all contain a heating element that heats up a layer of conducting material whose resistance is continuously measured. Its resistance changes when fumes or smell from alcohol comes in contact with the MQ-3 sensor. **Arduino based alcohol detector** having the mq3 sensor the specification given above. which can be monitor in our IoT based alcohol detector project.

The sensor also has an onboard power led and also an onboard Status led which will blink whenever the sensor detects alcohol(fumes).

ALCOHOL DETECTOR WORKING: –

As the code starts it initializes the pin to which the sensor sends its data. Then is setup we define the type of function we want on the initialized pins like input or output. Then in the loop, we read the data available on pin 3 and then print that value on our monitor. We also check for the condition of sensor data to be either high or low and according to the condition led glows and turns off.

In the buzzer code, we also initialize the buzzer pin to pin 13 and set pin mode to OUTPUT in the setup section. In the loop section along with the led we switch Buzzer also high and low. By this a simple DIY alcohol detector is complete and it can be used in various places.



Programming of alcohol detector :

```
const int A0_in = 0;
const int D0 = 8;
const int led = 13;
int threshold ;
int a0_output ;

void setup(){
Serial.begin(9600);
pinMode(D0 , INPUT);
pinMode(led , OUTPUT);
 }

void loop()
{
a0_output = analogRead(A0_in);
threshold = digitalRead(D0);
Serial.print(" Methane : ");
Serial.println(a0_output);
Serial.print(" Threshold : ");
Serial.print(threshold);
delay(200);

if (threshold == HIGH){
digitalWrite( led ,HIGH);
 }
else {
digitalWrite(led , LOW);
 }
}
```

□

Future scope

The Arduino is going to reduce the minimum volume necessary to include a control and sensing system with a product.
Instead of spending large amounts of money to build hundreds of inflexible circuit boards, the Arduino will allow businesses to bring many more unique devices to market at lower breakeven volumes.
We'll see a lot more lower-volume customized products.
The Arduino is going to allow businesses to develop products that are more easily upgradeable.
Right now if you buy a product, such as a microwave, there's no way to change the functionality.
If the microwave used an Arduino board, you would be able to change the interface or the way that the microwave cooked food to suit your desires.

References

- WWW.tinkercad.com
- www.arduino.ccprojects
  - Book - Simon Monk,Ryan Turner
  - Google