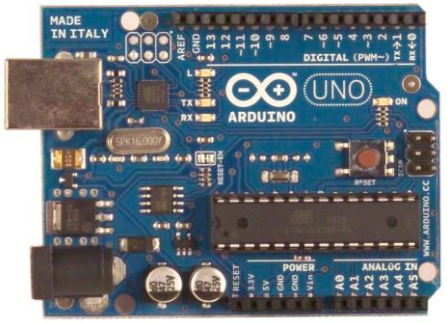
Embedded System Design & Application

Programming with Arduino

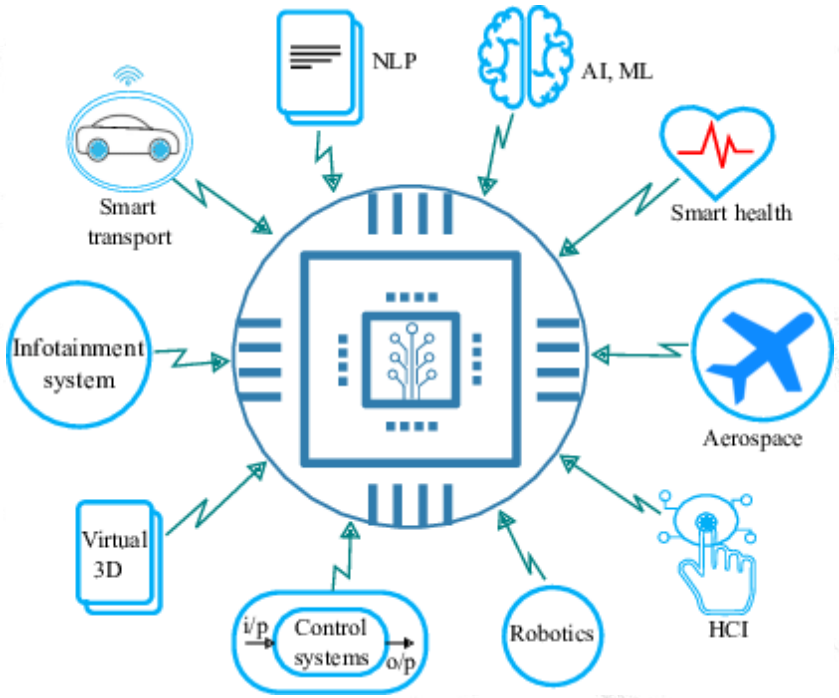
Dr. Sarwan Singh

NIELIT Ropar



agenda

- Embedded System Programming
  - Best embedded programming languages
- Arduino – Introduction, history
- Programming in C language.



Embedded System Programming

- ***Embedded systems programming*** is the process of developing software for embedded systems.
- This type of programming is different from traditional programming because it requires a deep understanding of the hardware that the software will be running on.
- Applications for embedded systems programming range from consumer electronics to industrial automation.

Source : linkedin.com/pulse/top-5-best-embedded-systems-programming/

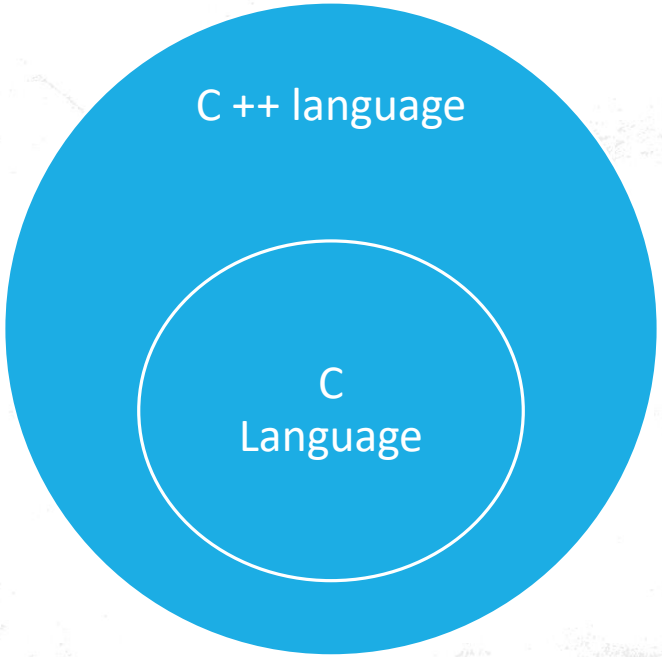
Top five best *embedded systems programming* languages.

C Language

- The most widely used *programming language* for embedded devices is C. It is a low-level language made to be quick and effective.
- C is used to develop software for microcontrollers, which are small computers that are embedded in a wide range of devices.
- C is a powerful language that allows developers to write code that is close to the hardware, which makes it ideal for embedded systems programming.
- C is a procedural language that provides low-level access to memory and hardware. It is a language that is widely used in the development of operating systems, *embedded systems*, and real-time systems.
- C is a language that is highly portable and can be used on a wide range of hardware platforms.

C++ Language

- C++ is a high-level programming language that is used in a wide range of applications, including *embedded systems programming*.
- C++ is an extension of the C programming language, and it includes features such as object-oriented programming and templates.
- C++ is a popular choice for embedded systems programming because it is a powerful language that allows developers to write efficient code.



Assembly Language

- Assembly language is a low-level programming language that is used to develop software for microcontrollers. Assembly language is used to write code that is specific to the hardware that it is running on. Assembly language is a powerful language that allows developers to write code that is highly efficient and fast.
- Assembly language is a language that provides low-level access to memory and hardware. It is a language that is widely used in the development of operating systems, embedded systems, and real-time systems. Assembly language is a language that is highly specific to the hardware platform that it is running on.

Python

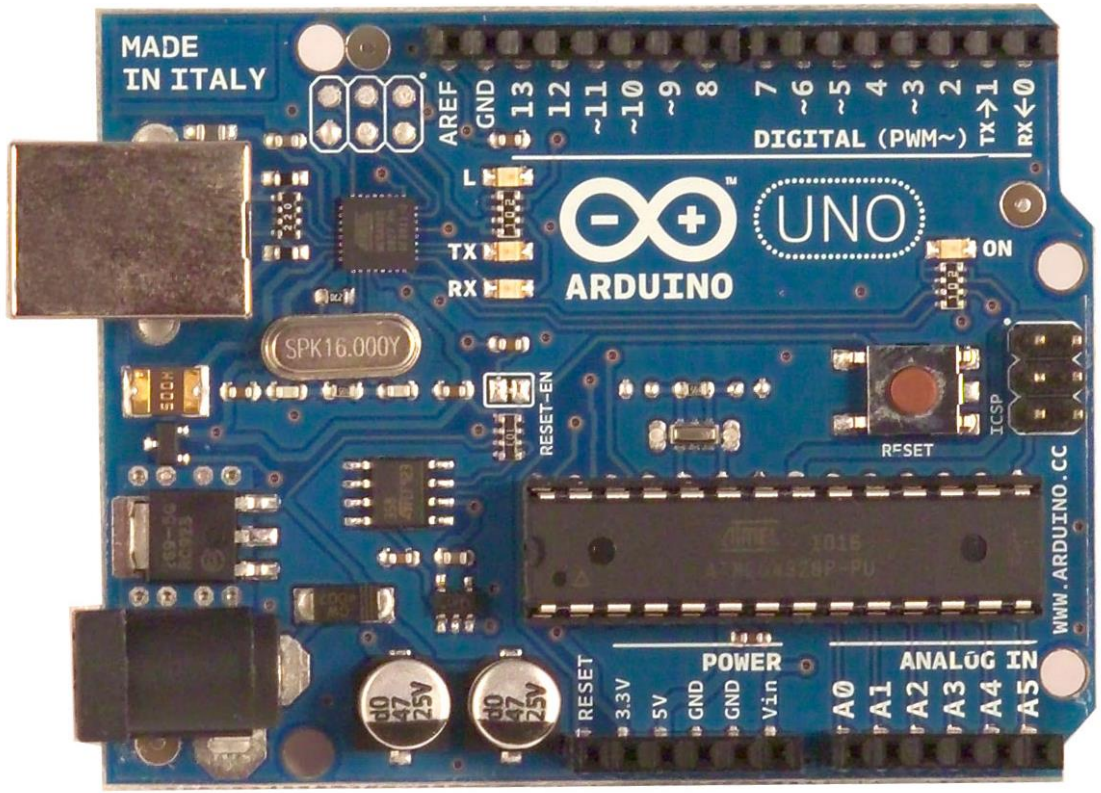
- Python is a high-level programming language that is used in a wide range of applications, including embedded systems programming. Python is a popular choice for embedded systems programming because it is easy to learn and use. Python is a powerful language that allows developers to write code that is efficient and fast.
- Python is a language that provides high-level abstractions that make it easier to write complex software. It is a language that is widely used in the development of web applications, scientific computing, and embedded systems. Python is a language that is highly portable and can be used on a wide range of hardware platforms.

Java

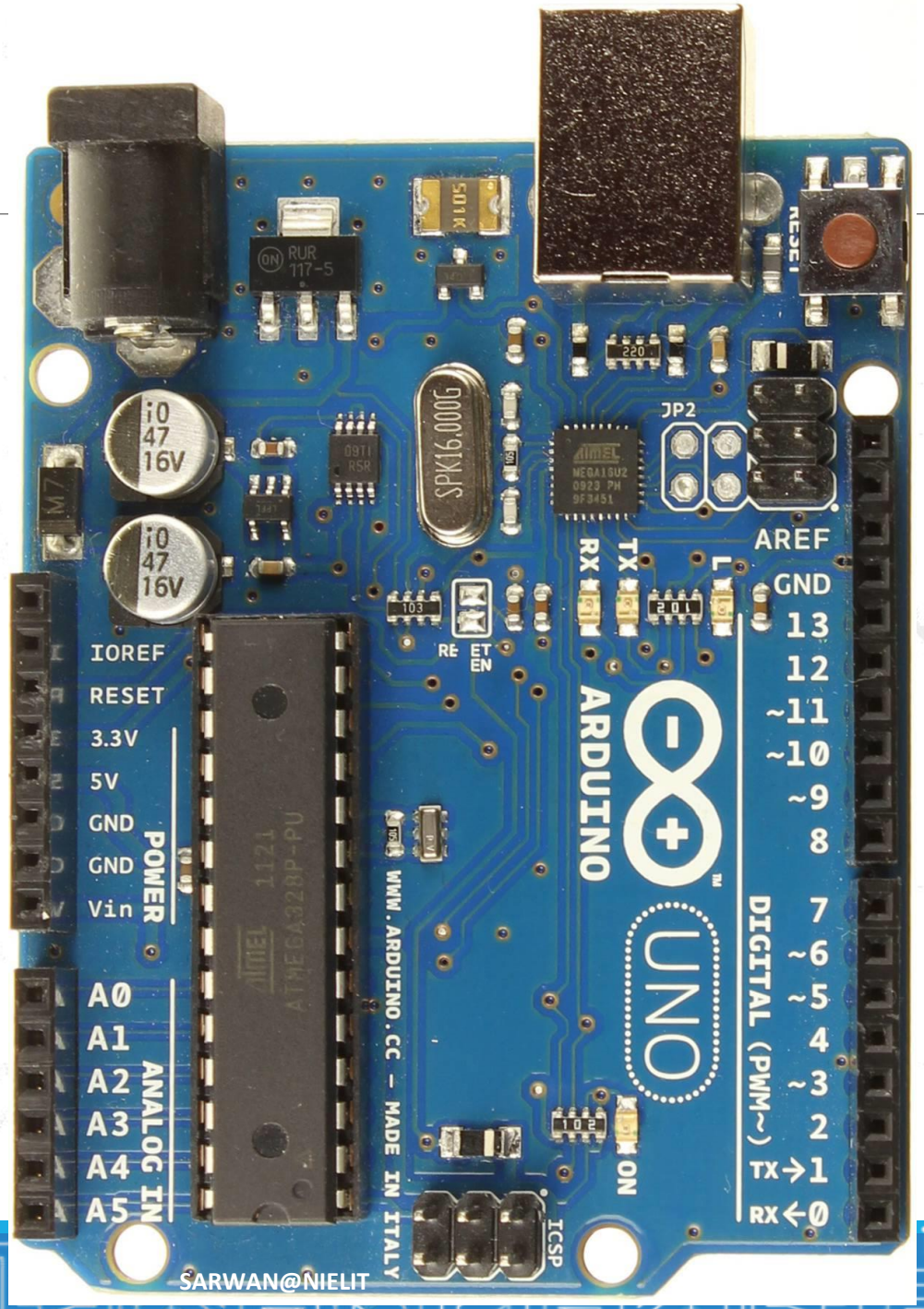
- Java is a high-level programming language that is used in a wide range of applications, including *embedded systems programming*.
- Java is a popular choice for embedded systems programming because it is a powerful language that allows developers to write code that is efficient and fast.
- Java is also a portable language, which means that it can run on a wide range of hardware platforms.
- Java is a language that provides high-level abstractions that make it easier to write complex software. It is a language that is widely used in the development of web applications, enterprise applications, and embedded systems.
- Java is a language that is highly portable and can be used on a wide range of hardware platforms.

What is an Arduino ?

- **Open Source** electronic prototyping **platform** based on flexible **easy to use** hardware and software.



Arduino UNO R3



Arduino

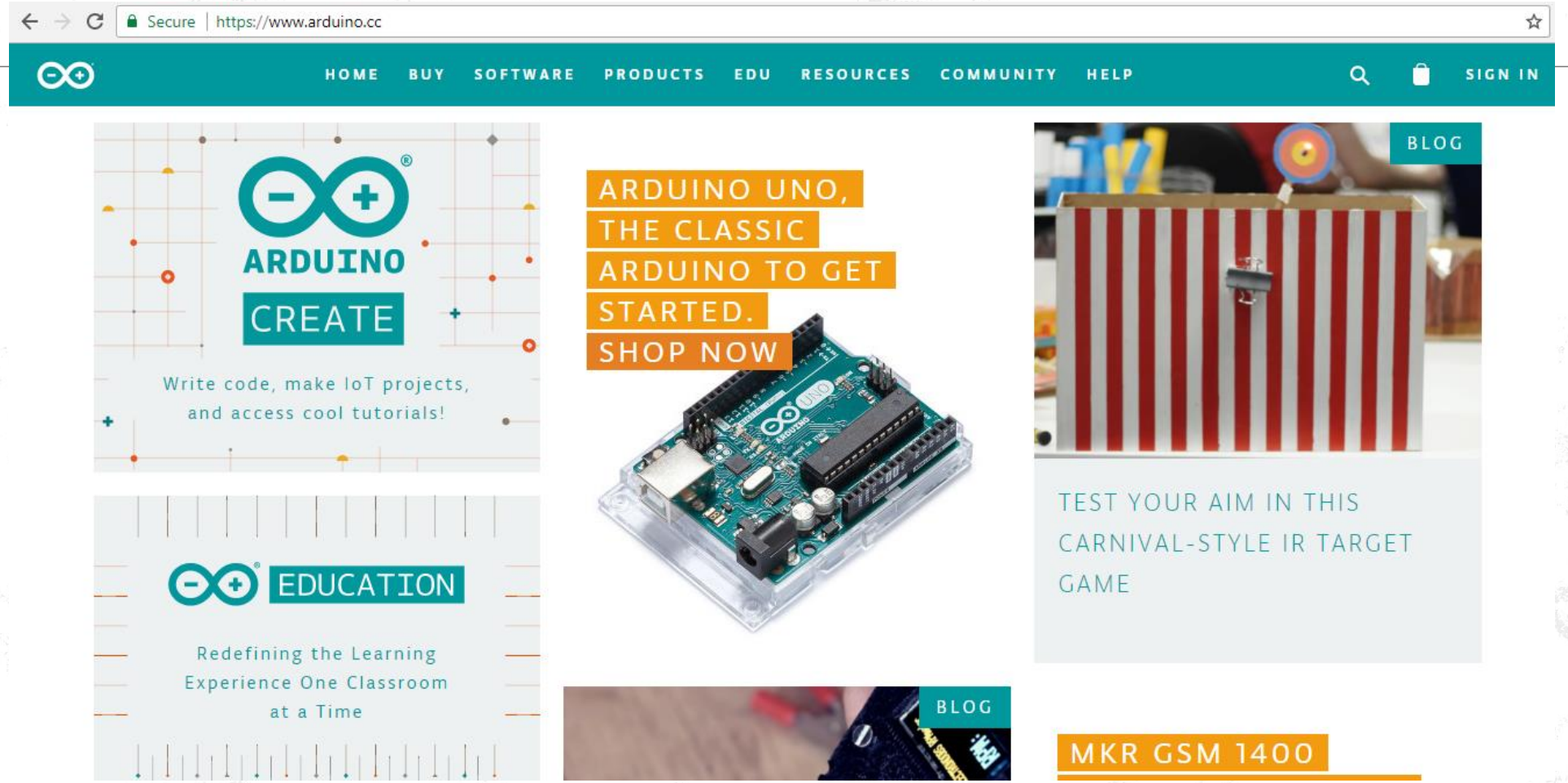
- Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.
- Founded by Massimo Banzi and David Cuartielles in 2005.
- An open-source hardware platform based on Atmel AVR microcontroller and a C++ based IDE.
- In-Expensive, Simple and easy to learn programing.
- Controller independent programming language.
- One language compatibility with all boards.
- Single software for programming, compiling and burning the code.

Advantages of Arduino for Learning Embedded Systems

- Real Embedded Hardware: Arduino boards host a microcontroller, making them genuine embedded systems. Learning on Arduino allows beginners to work with actual hardware while still being beginner-friendly.
- Low-Level Programming: Arduino's programming language, based on C/C++, helps learners grasp fundamental embedded programming concepts.
- Hardware Interfacing: Arduino's simplicity makes it easy for beginners to connect sensors, actuators, and other peripherals. Learning hardware interfacing is essential for understanding how embedded systems control external devices.
- Community Support: Arduino boasts a large and active community that offers tutorials, projects, and forums. Beginners can seek guidance, share ideas, and collaborate with others, fostering a supportive learning environment.
- Prototyping: Arduino's breadboard-friendly design and plug-and-play compatibility allow quick prototyping, enabling learners to test and iterate their ideas rapidly.

Downsides and Limitations of Arduino

- Limited Processing Power: Arduino boards utilize relatively simple microcontrollers with restricted processing capabilities, limiting their use in computationally intensive applications.
- Memory Constraints: Arduino's limited RAM and Flash memory might hinder projects requiring extensive code or handling large data sets.
- Lack of Real-Time Capabilities: While Arduino can handle simple real-time tasks, it lacks a full-fledged real-time operating system, making precise and predictable timing challenging.
- Not Suitable for High-Performance Applications: Arduino's modest processing power makes it less suitable for high-performance applications like complex image processing or machine learning.
- Inefficient Power Usage: Arduino's design and components may lead to higher power consumption, affecting battery life in portable applications.
- Less Control over Low-Level Hardware: Arduino abstracts hardware complexities for ease of use, which may limit users seeking more direct control over low-level hardware features.
- Limited Debugging and Profiling Support: Arduino's debugging and profiling capabilities are less robust compared to professional development environments, making troubleshooting complex issues more challenging.

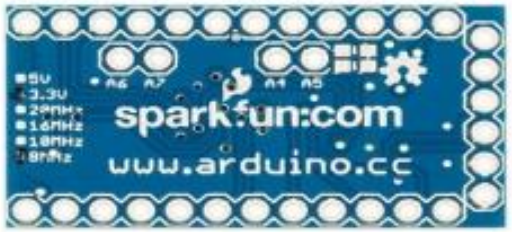

Secure | https://www.arduino.cc
HOME BUY SOFTWARE PRODUCTS EDU RESOURCES COMMUNITY HELP SIGN IN
ARDUINO
CREATE
Write code, make IoT projects, and access cool tutorials!
ARDUINO UNO, THE CLASSIC ARDUINO TO GET STARTED. SHOP NOW
BLOG
TEST YOUR AIM IN THIS CARNIVAL-STYLE IR TARGET GAME
EDUCATION
Redefining the Learning Experience One Classroom at a Time
BLOG
MKR GSM 1400

Different types of Arduinos

Arduino Mega 2560

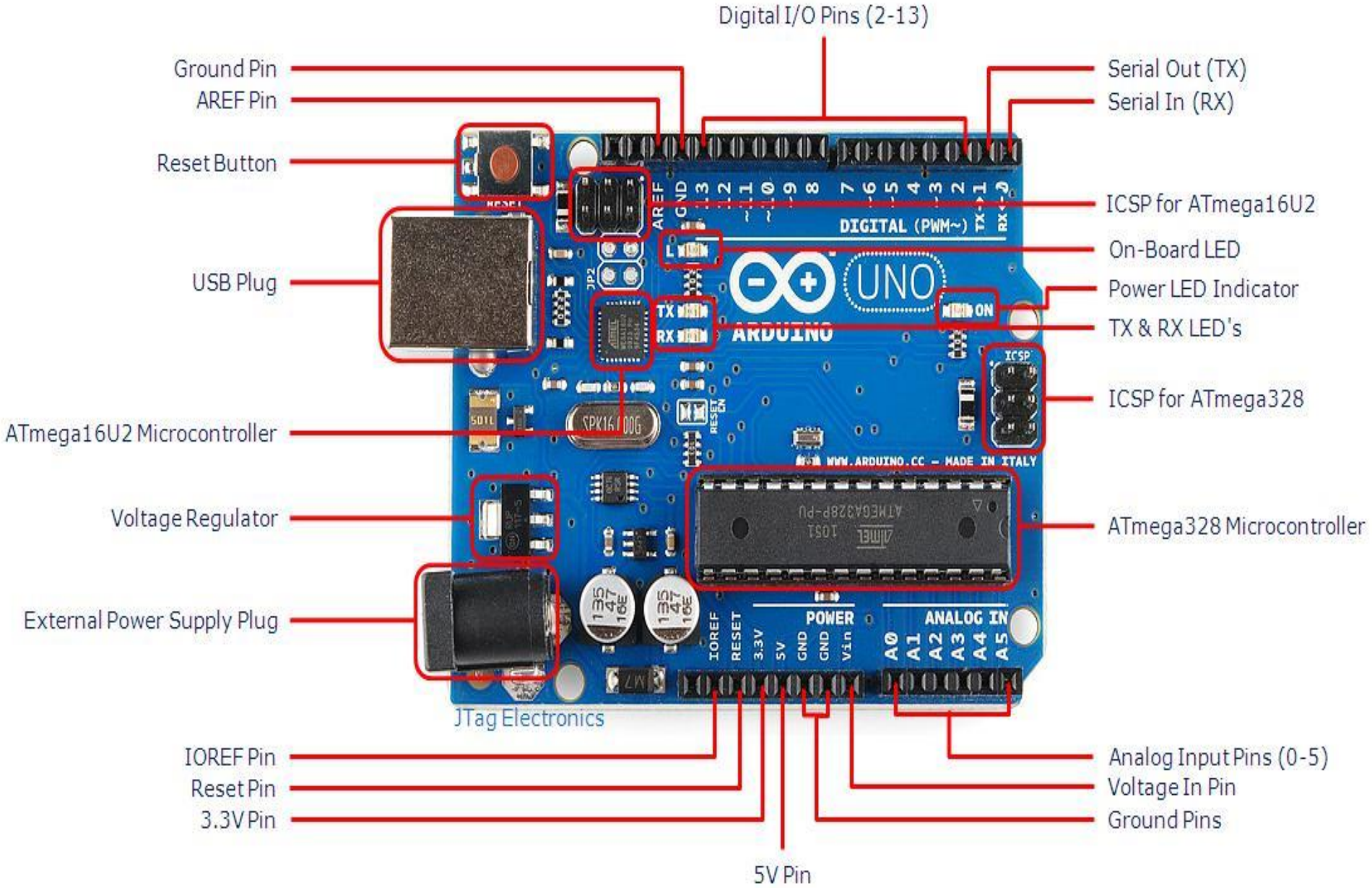
Arduino Uno

Arduino YUN

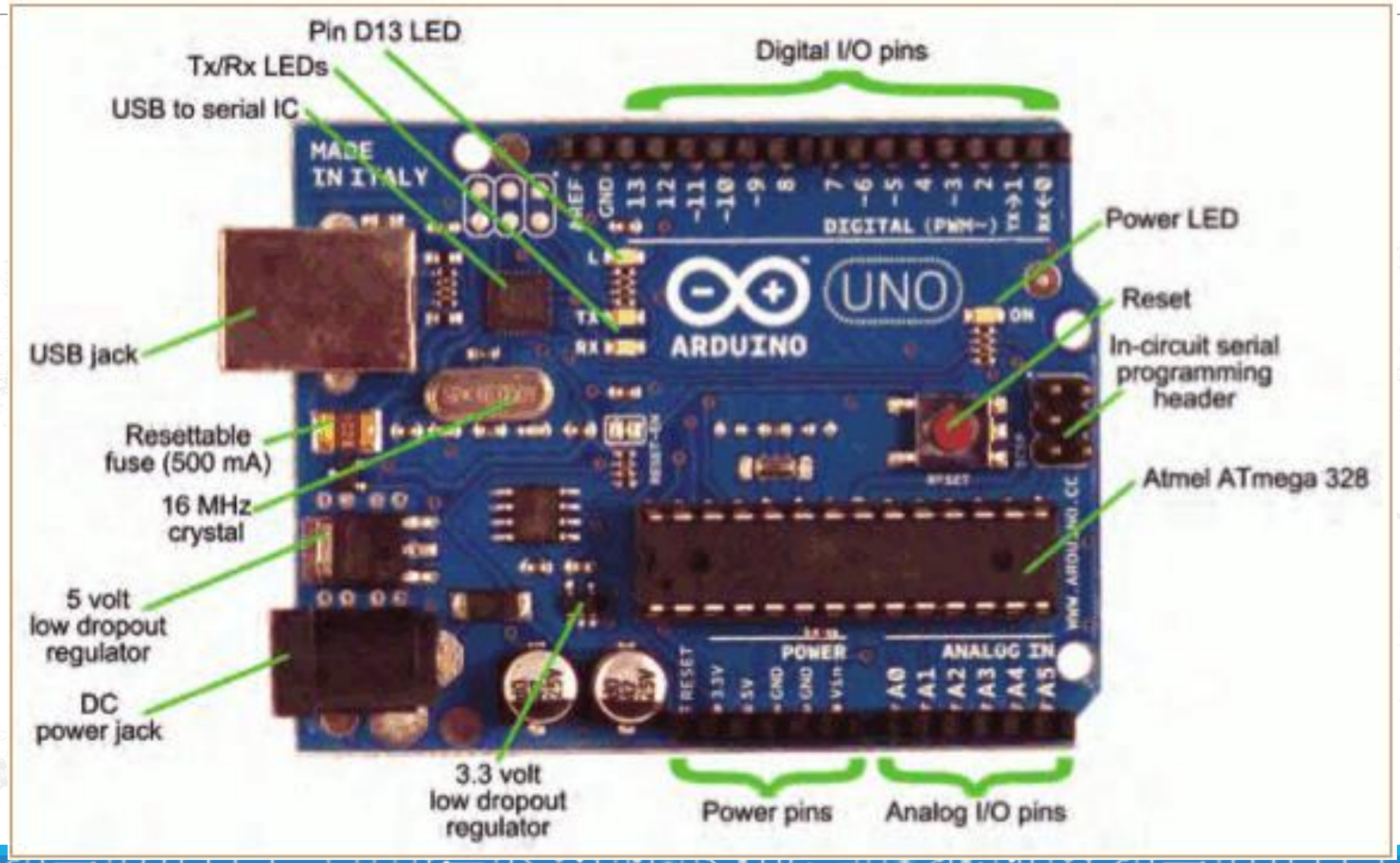


http://www.arduino.cc/

- Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.
  - Processor: 16 Mhz ATmega328
  - Flash memory: 32 KB
  - Ram: 2kb
  - Operating Voltage: 5V
  - Input Voltage: 7-12 V
  - Number of analog inputs: 6
  - Number of digital I/O: 14 (6 of them PWM)


Digital I/O Pins (2-13)
Ground Pin
AREF Pin
Reset Button
USB Plug
ATmega16U2 Microcontroller
Voltage Regulator
External Power Supply Plug
JTag Electronics
IOREF Pin
Reset Pin
3.3V Pin
5V Pin
Serial Out (TX)
Serial In (RX)
ICSP for ATmega16U2
On-Board LED
Power LED Indicator
TX & RX LED's
ICSP for ATmega328
ATmega328 Microcontroller
Analog Input Pins (0-5)
Voltage In Pin
Ground Pins
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
WWW.ARDUINO.CC – MADE IN ITALY

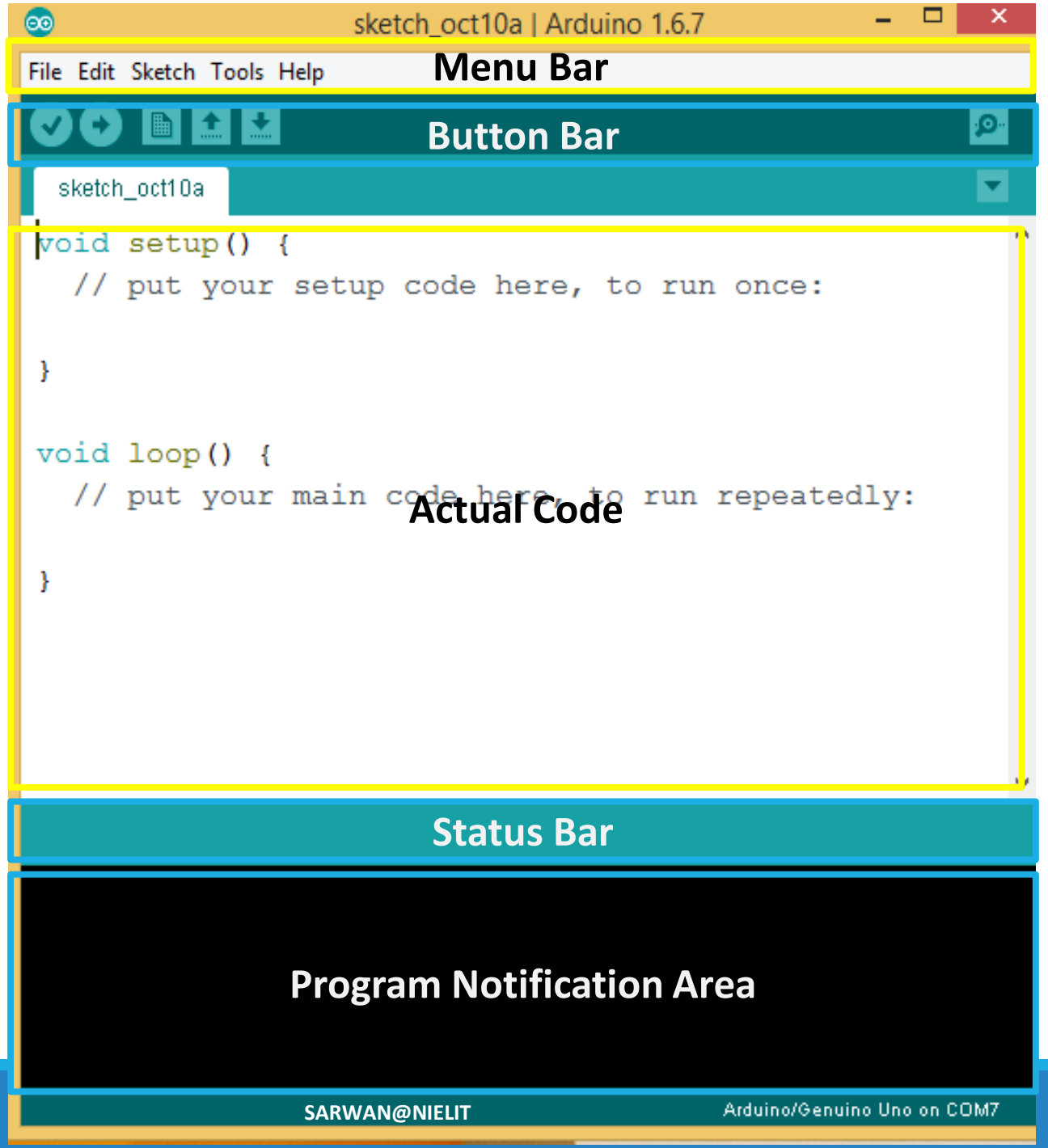
Meet Arduino Uno

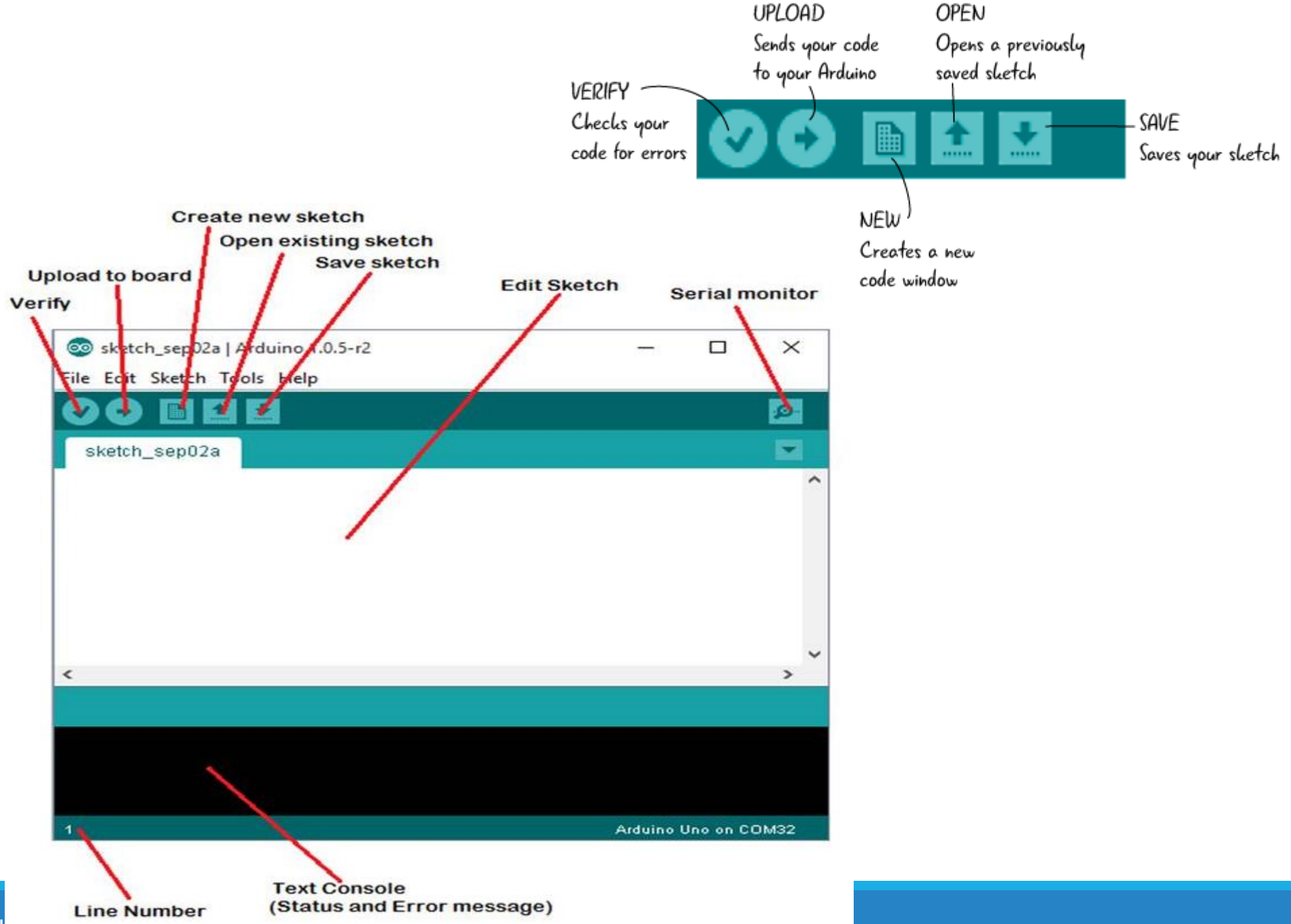


Getting Started

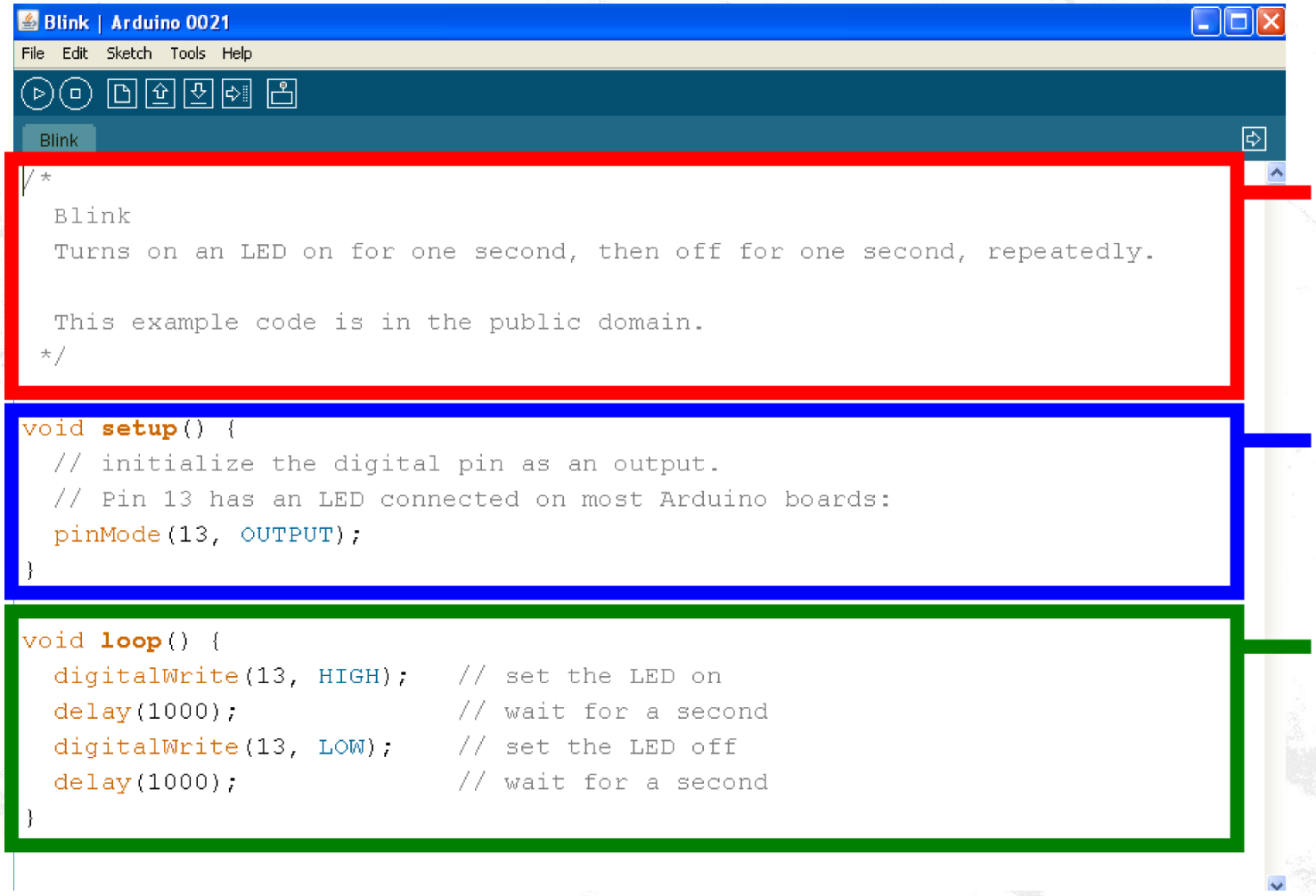
- Check out: http://arduino.cc/en/Guide/HomePage
  1. Download & install the Arduino environment (IDE)
  2. Connect the board to your computer via the USB cable. If needed, install the drivers
  3. Launch the Arduino IDE
  4. Select your board
  5. Select your serial port
  6. Open the blink example
  7. Upload the program

Arduino IDE


sketch_oct10a | Arduino 1.6.7
File Edit Sketch Tools Help
Menu Bar
Button Bar
sketch_oct10a
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
Actual Code
Status Bar
Program Notification Area
SARWAN@NIELIT
Arduino/Genuino Uno on COM7


UPLOAD
Sends your code to your Arduino
OPEN
Opens a previously saved sketch
VERIFY
Checks your code for errors
SAVE
Saves your sketch
NEW
Creates a new code window
Create new sketch
Open existing sketch
Save sketch
Upload to board
Verify
Edit Sketch
Serial monitor
File Edit Sketch Tools Help
sketch_sep02a
1
Arduino Uno on COM32
Line Number
Text Console
(Status and Error message)

Parts of the Sketch

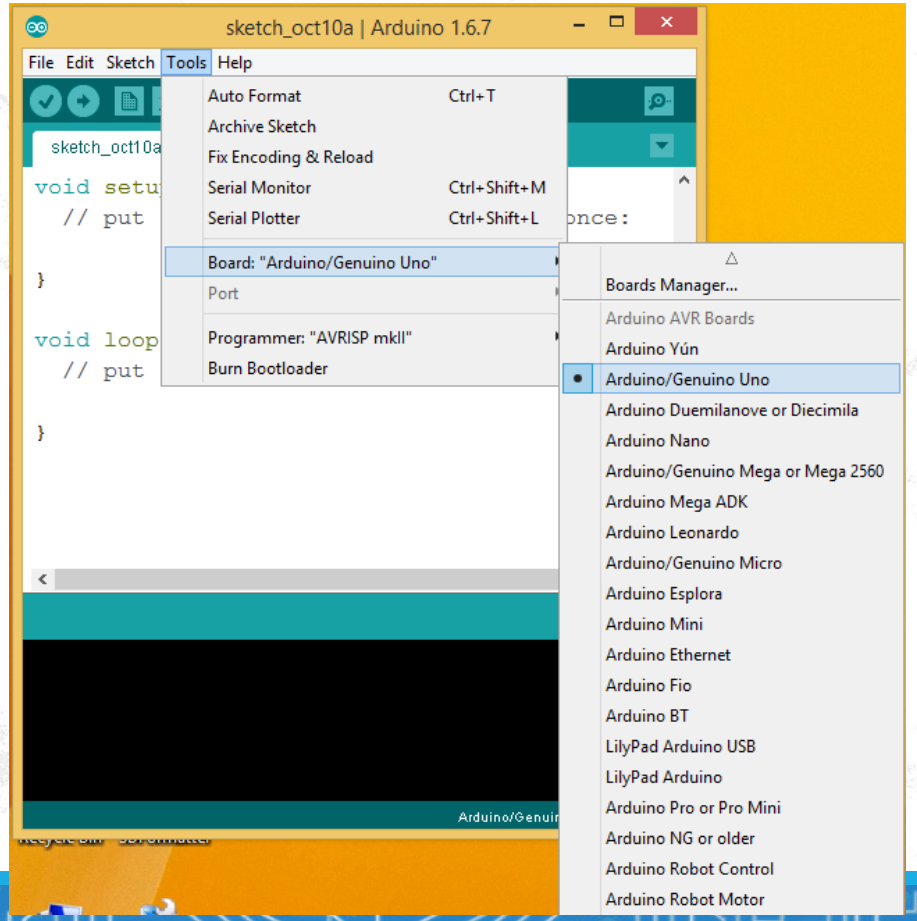


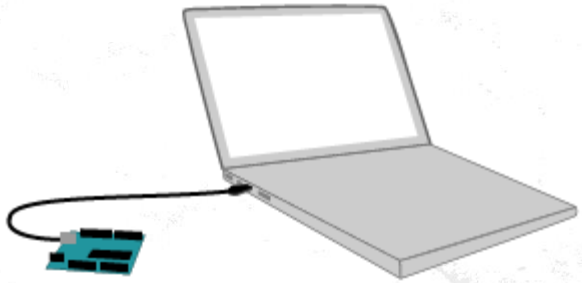
Comments / Explaining the game

Setup / Stretching or tying shoes

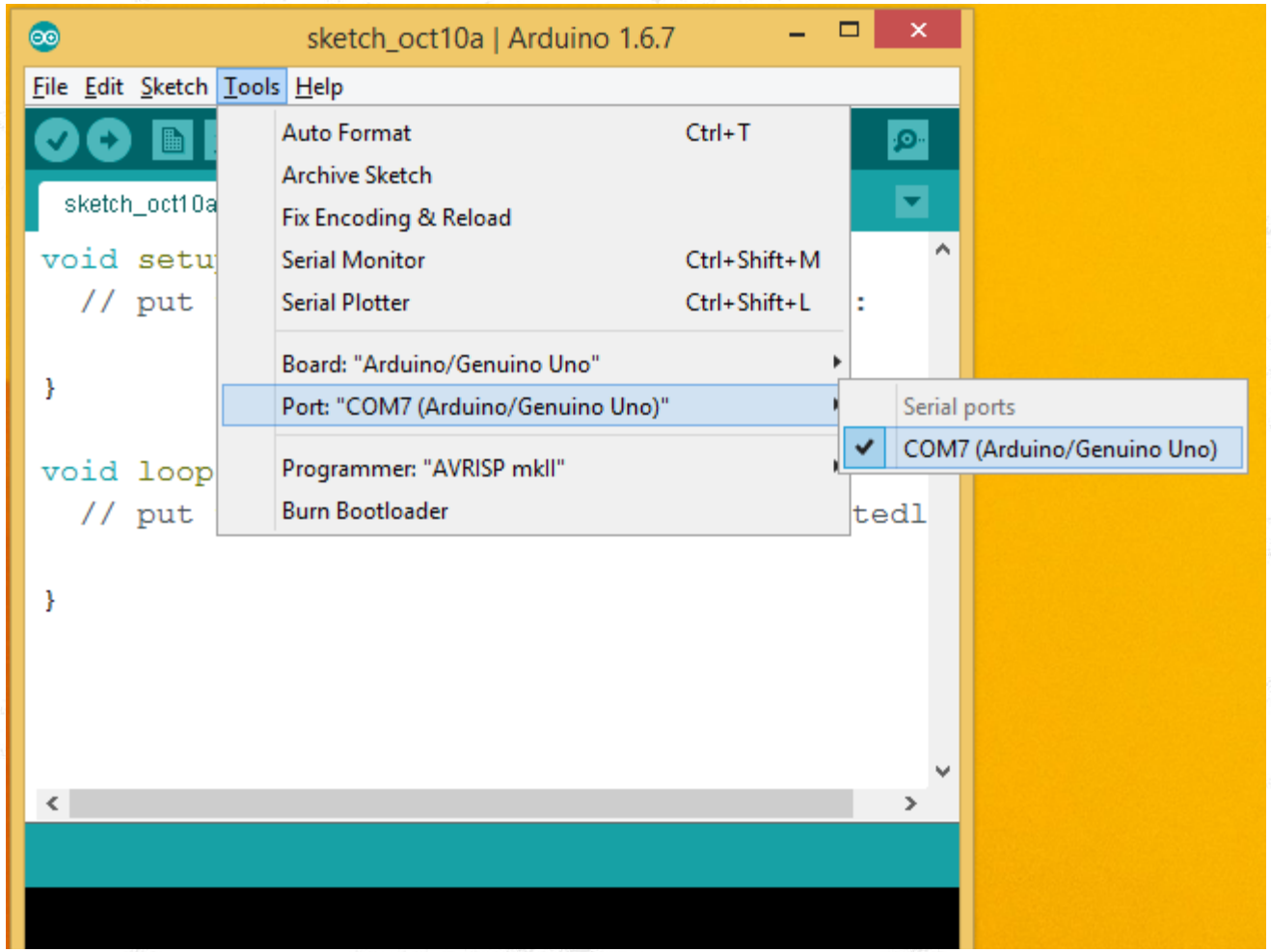
Loop / Playing the game

Select Board



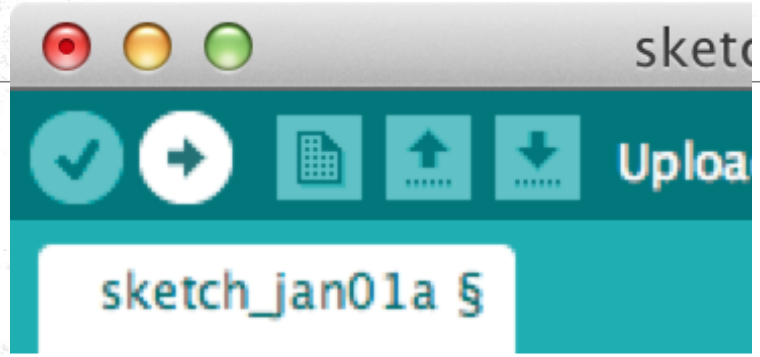


Select com port



First program

```
int ledPin = 13;
void setup()
{
   pinMode(ledPin, OUTPUT);
}
void loop()
{
   digitalWrite(ledPin, HIGH);
   delay(2000);
   digitalWrite(ledPin, LOW);
   delay(2000);
}
```




```
int ledPin = 13;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, LOW);
}
```

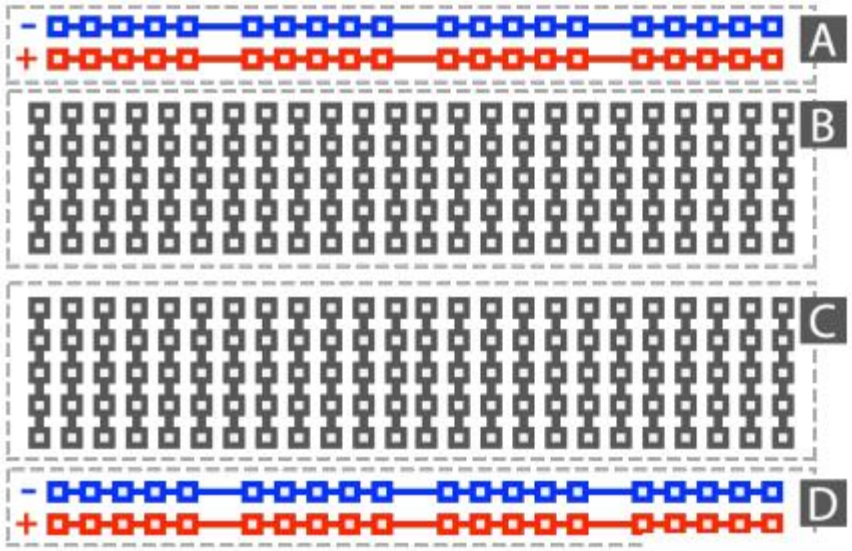
```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

Outputs are declare in setup, this is done by using the pinMode function

This particular example declares digital pin # 13 as an output, remember to use CAPS

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
  Serial.begin(9600);
  digitalWrite(12, HIGH);
}
```

Bread Board



Github Repo

https://github.com/sarwansingh/IoT/tree/master/ClassExamples/CEPTAM_ESDANov23

Happy Coding

JOURNEY BEGINS FROM HERE……