VOL. 1

LIMITED EDITION

CARTOOINO COMICS

GreenLab Cartooino

Acknowledgement

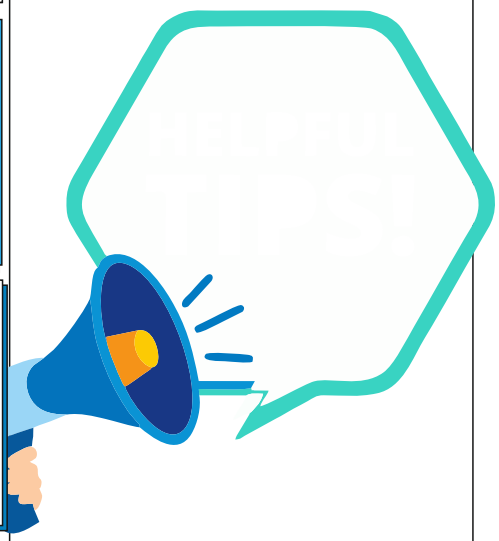
The cartooino Book is a comical adaptation of the Arduino Projects Book. The Book was developed by Greenlab Microfactory for use for her "One student, One Arduino" project. The essence of which is to create a more relatable learning approach for the participants. Therefore, the Microfactory would like to acknowledge Arduino LLC for their selflessness and contribution to the body of knowledge by fortifying the open source community with their products and projects.

In addition, Greenlab Microfactory will also like to acknowledge that some of the content of this book were derived and adapted from the online platforms of Tutorials Point etc.
.

Disclaimer

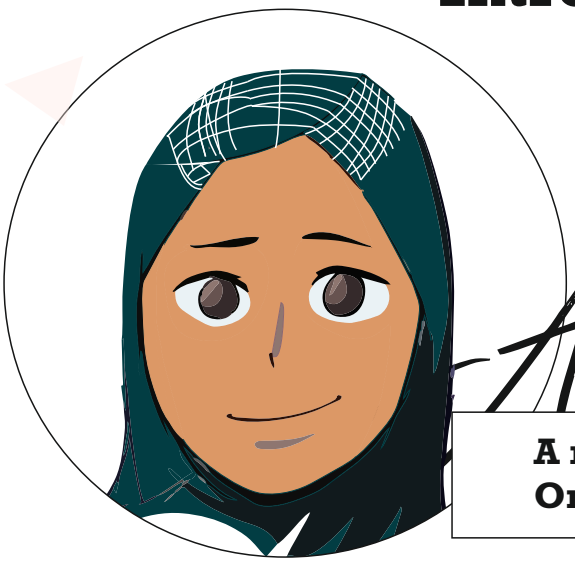
Stack of Content

Getting to know your tools	01
Building a simple circuit	02
Understanding the program interface	03
Spaceship interface	04
YouNiversity	05
Colour mixing lamp	06



GreenLab Cartoonno

Introduction

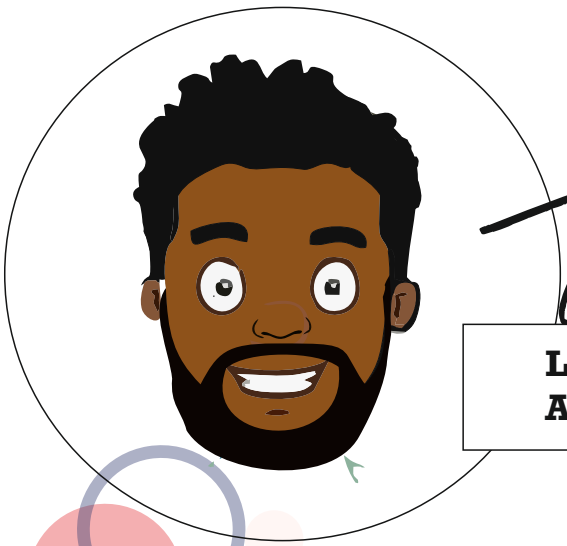


Aliya

A regular pupil and a participant of the One Student One Arduino Project

divine

A regular pupil and a participant of the One Student One Arduino Project



Uncle green

Lead facilitator of theOne Student One Arduino Project

GreenLab Cartoono

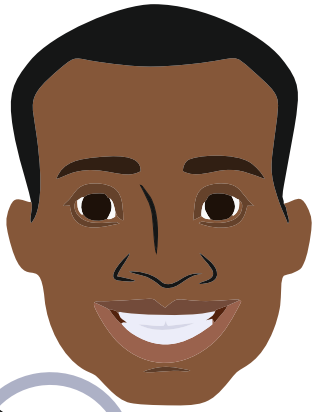
Introduction

Kunle

A regular pupil and a participant of the One Student One Arduino Project

Faith

A regular pupil and a participant of the One Student One Arduino Project



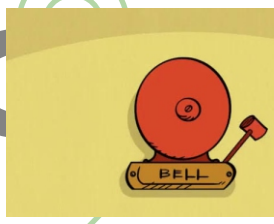
Emmanuel

A 400 level Student of computer science at UNIBEN and a co-facilitator of the One Student One Arduino Project

GreenLab Cartooino

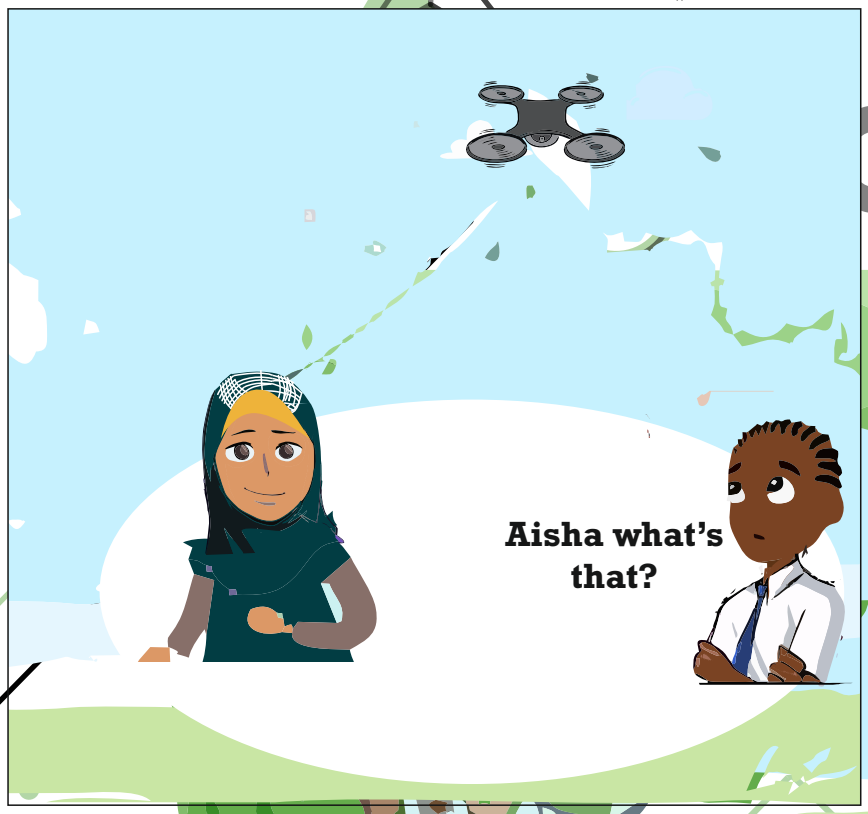
Maths teacher: X + Y= COS 30

Maths class was interesting but not as much as the kite competition

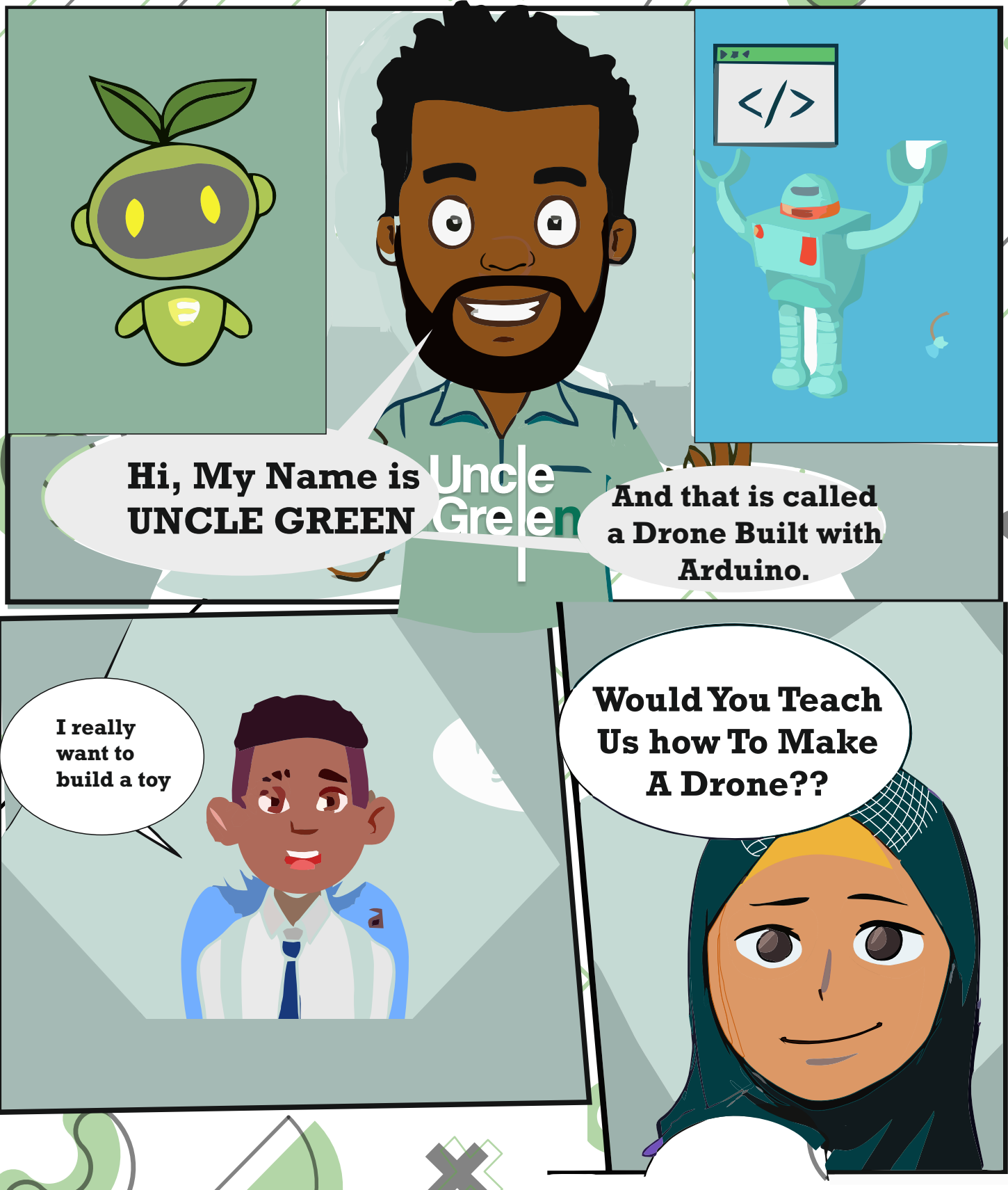


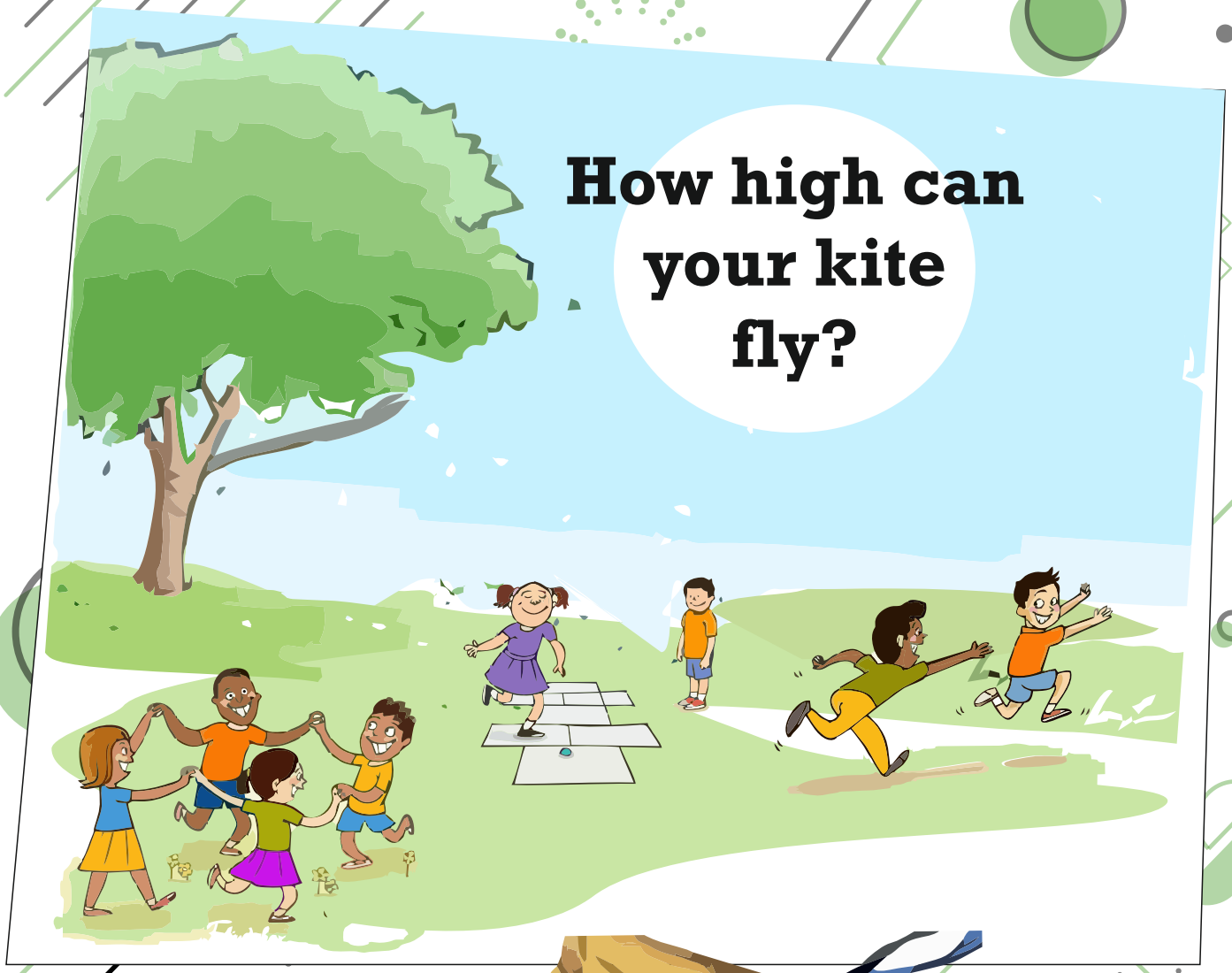
school over
school over
school over

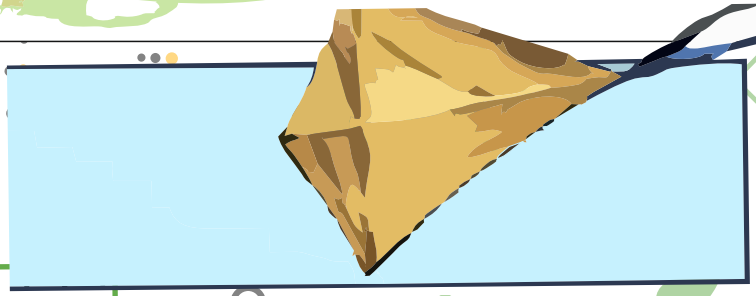
Stop Running Ayomide


Aisha what's that?


Look Over there!! I think the plane is been controlled by that uncle


Hi, My Name is UNCLE GREEN
Uncle Green
And that is called a Drone Built with Arduino.
I really want to build a toy
Would You Teach Us how To Make A Drone??


How high can your kite fly?




Nice

It will be stressful

Should we go join the Arduino project with Ayomide

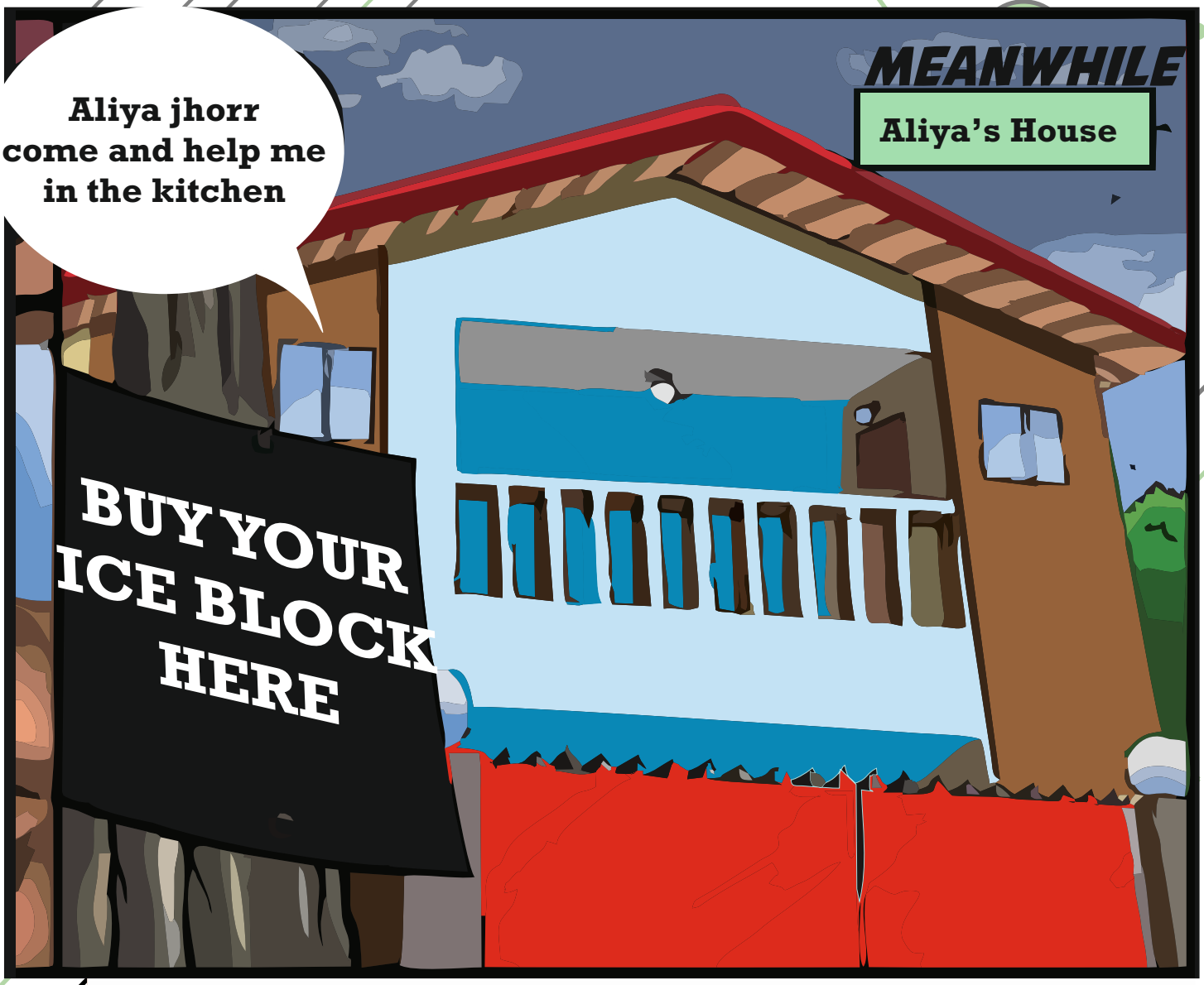
Aliya remember you have to tell your mom

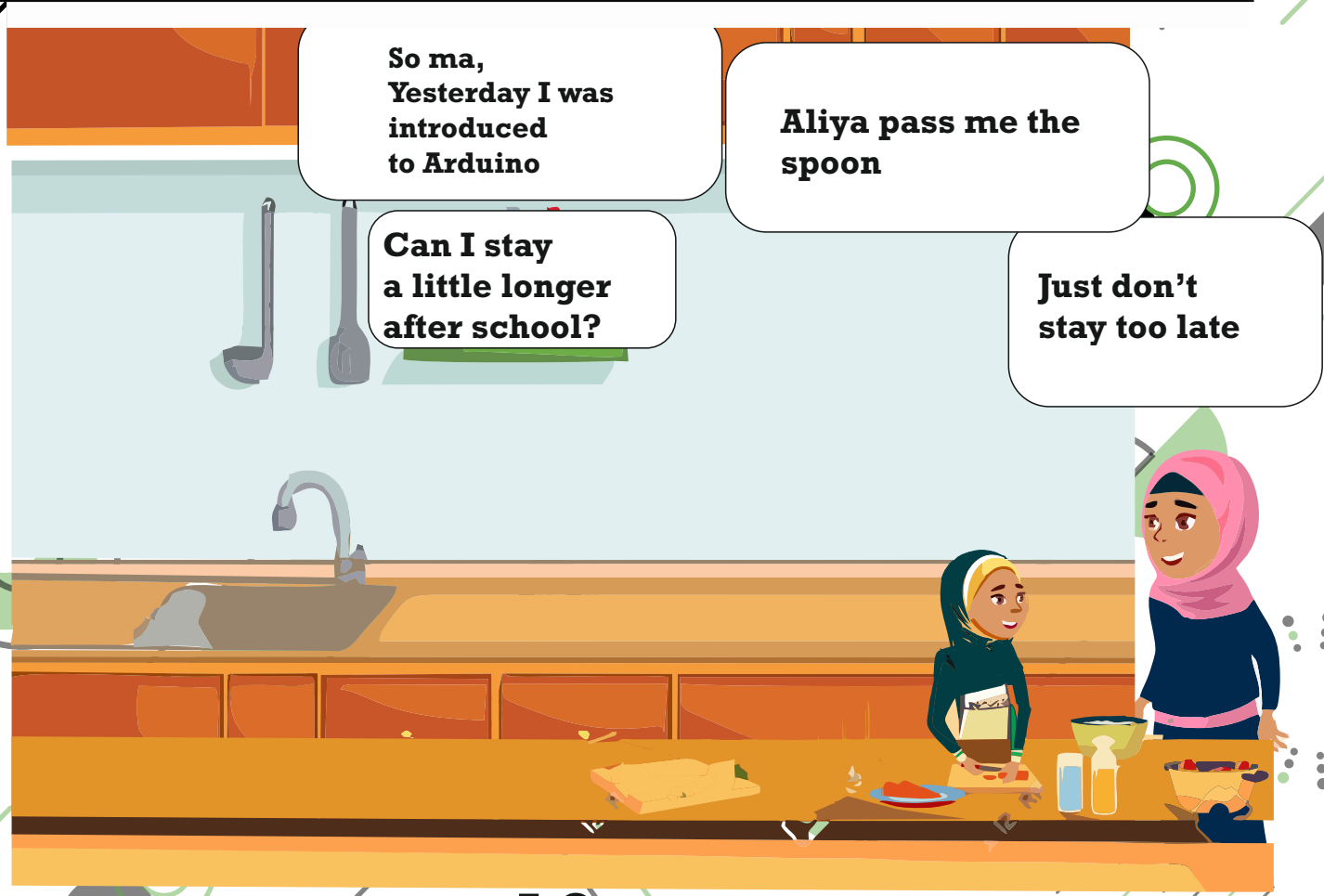
And don't forget to do your assignment

This place is awesome

WOW

At Uncle Green house


Aliya jhorr come and help me in the kitchen
MEANWHILE
Aliya's House
BUY YOUR ICE BLOCK HERE


So ma, Yesterday I was introduced to Arduino
Aliya pass me the spoon
Can I stay a little longer after school?
Just don't stay too late

NEXT DAY

Episode 01

Learn

Discover

Create

Getting to know your tool

Next Day

Hello kids, I am glad that you could come today, i officially welcome you to the "One student One Arduino" Project

Electricity is a type of energy, much like heat, gravity or light. Electrical energy flows through conductors like wire. You can convert electrical energy into other forms of energy.

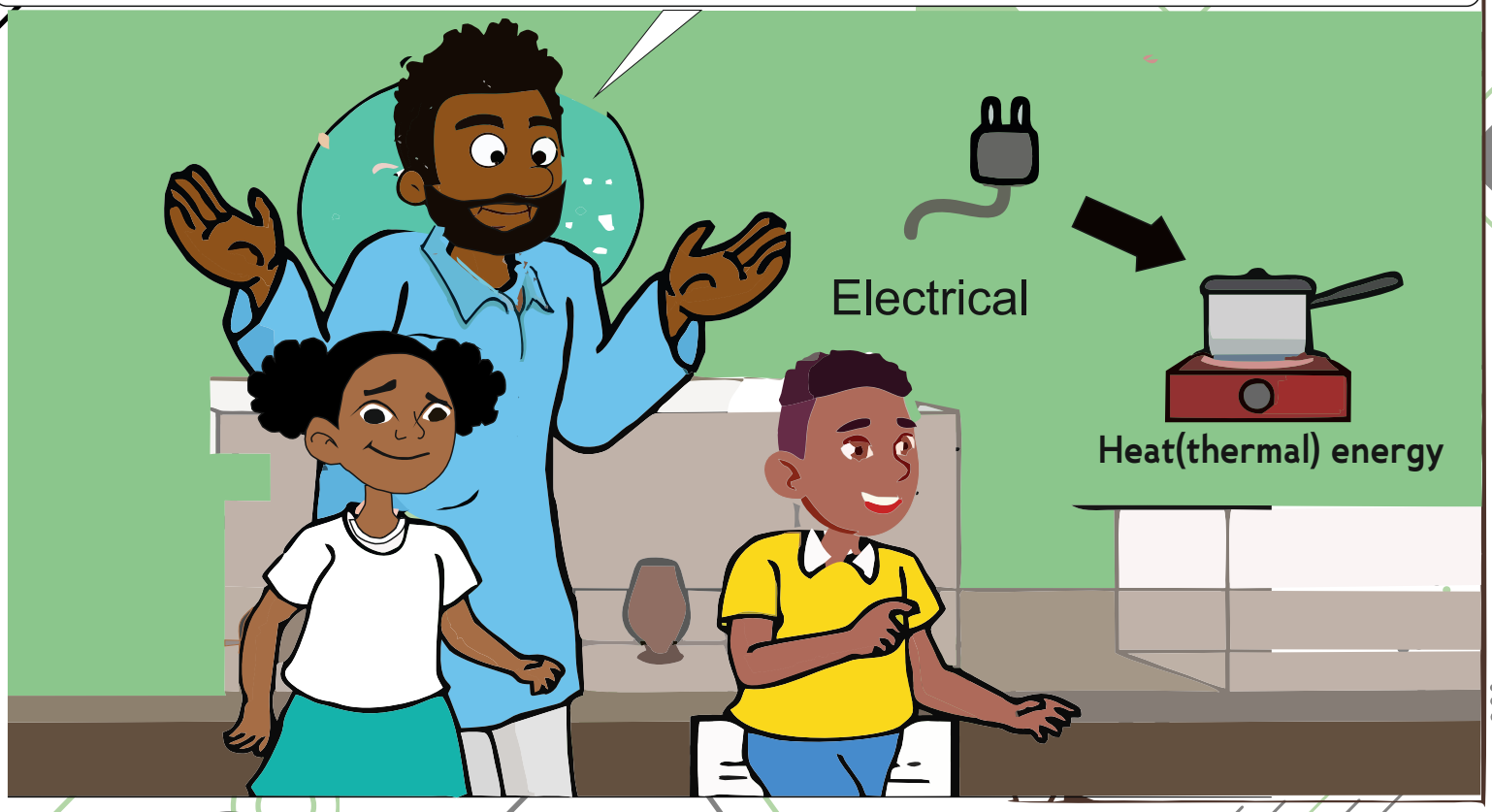
Do you know that one essential need is electricity

What can electricity be used for?

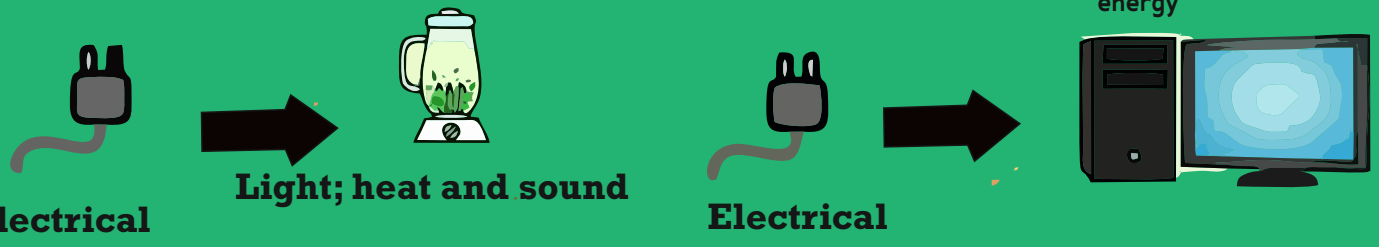
To watch TV

Like In Our Electric Bulb

These are examples of electric energy conversions

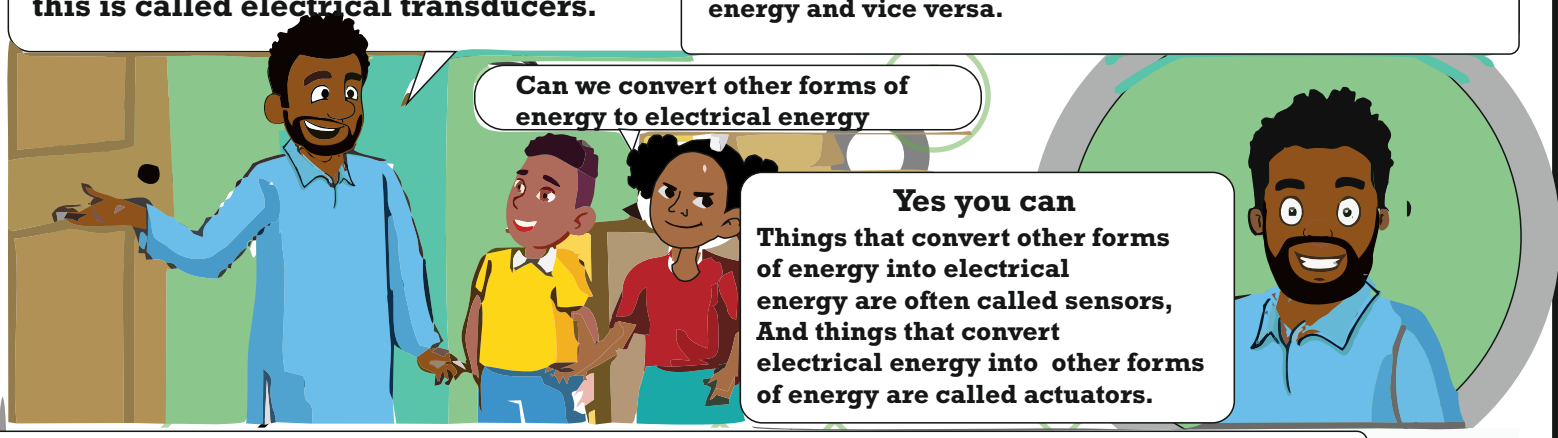


More Examples of Electrical energy Conversion

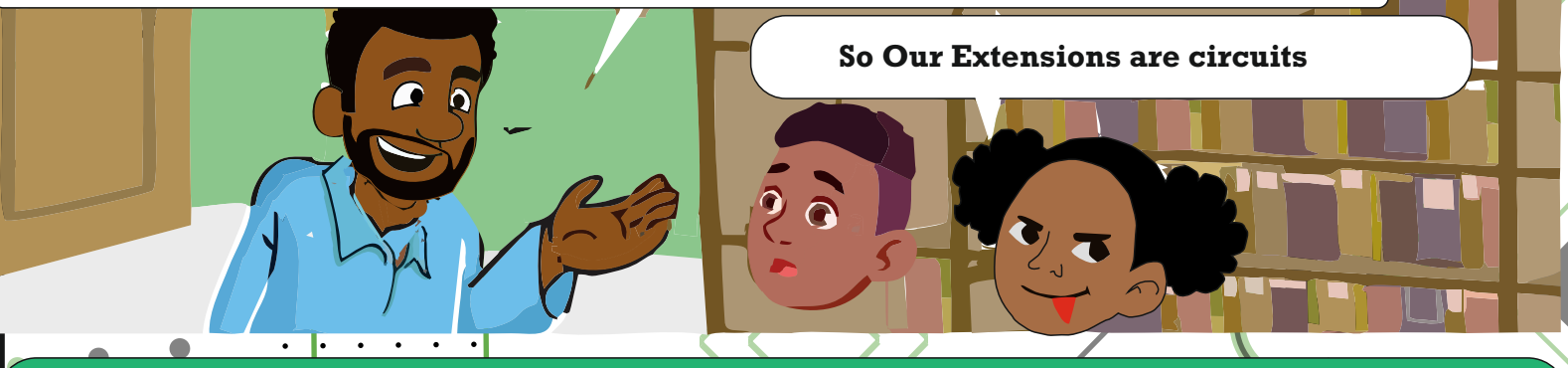


The components you might use to do this is called electrical transducers.

Transducers change other types of energy into electrical energy and vice versa.

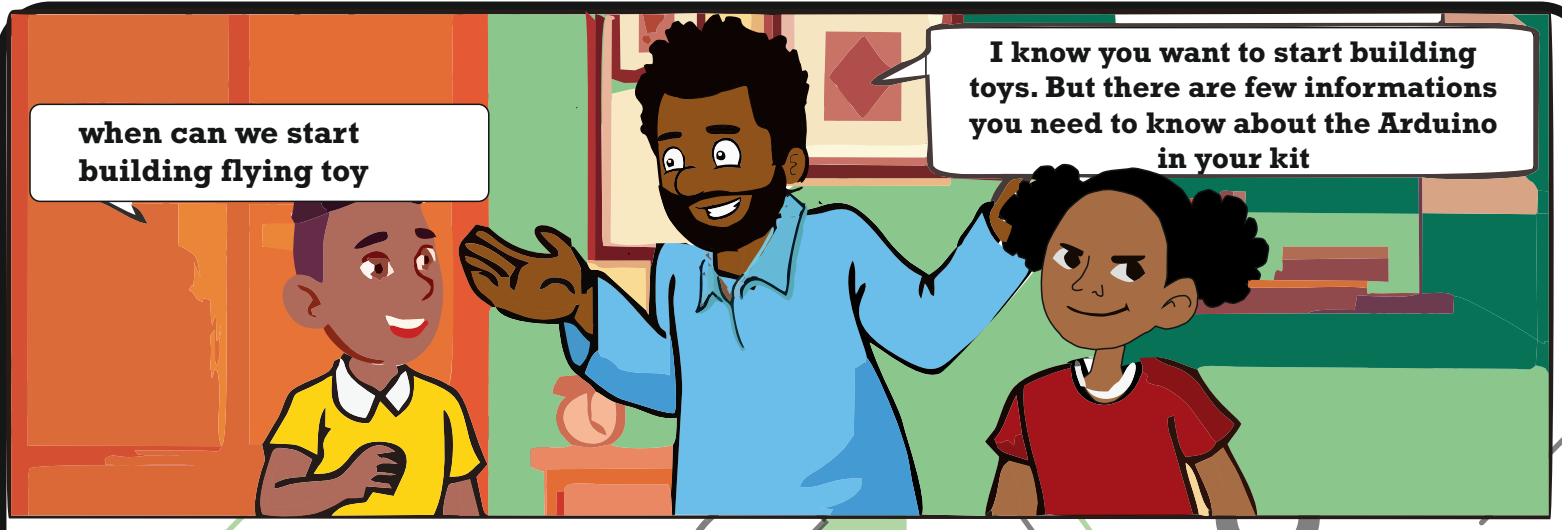


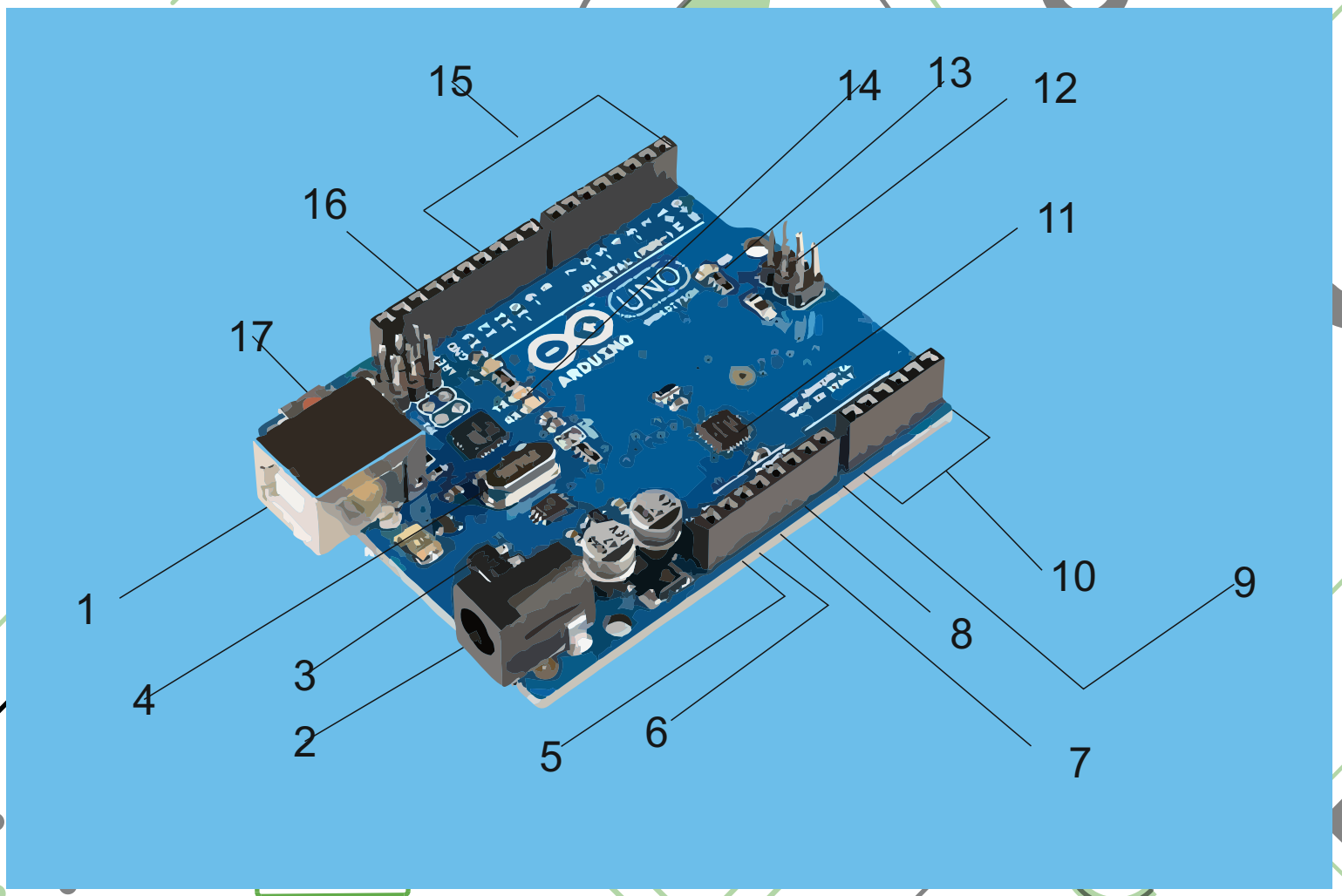
Circuits are closed loops of wires with a power source (like a battery) and something to do something useful with the energy, called a load.



In a circuit, electricity from a point of higher potential energy (usually referred to as power or +) to a point of lower potential energy. Ground(often represented with. a - or GND) is generally the point of least potential energy in a circuit.
In the circuit you are building, electricity only flows in one direction.
This type of circuit is called direct current, or DC. In alternating current(AC) the circuit changes its direction 50-60 times a second(depending on where you live).
This type of electricity that comes from your wall socket.

I will give you guys a little assignment.
look up these terms
Current, Voltage
and finally resistance





1. Power USB- it gives power to the Arduino board power when you connect the USB cable to the USB connection

2.Power(Barrel jack)

you can also power the board from AC mains power by connecting it to the barrel jack

3.Voltage Regulator-The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltage used by the processor and other elements.

4.Crstal Oscillator -The crystal Oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is by using the oscillator. The number printed on top of the Arduino crystal is 16,000H9H. It tells us the frequency is 16,000,000 Hertz or 16Mhz.

11. Main microcontroller

Each Arduino board has its own microcontroller

The Arduino UNO board has five analog input A0 through A5. These pins can read the signal from an analog sensor like the humility sensor or temperature sensor and convert it into digital value that can be read by the micro-processopr

12. ICSP pin

Mostly, ICSP(12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC and GND. It Is often referred to as an SPI (Serial Peripherial Interface), which could be considered as an "expression" of the output. Actually, you are slaving the output device to the master of the SPI bus.

13. Power LED indicator

This LED should light up when you plug your arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

15. Digital I/O

The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output.) These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc.
The pins labelled " " can be used to generate PWM.

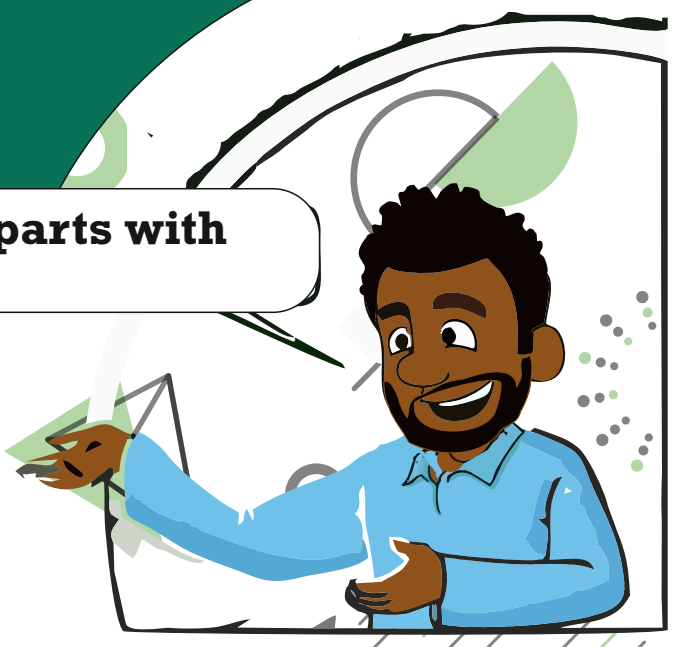
16. AREF

AREF stands for analog Reference. It is sometimes used to set an external volatge (between 0 and 5 volts) as the upper limit for the analog input pins.

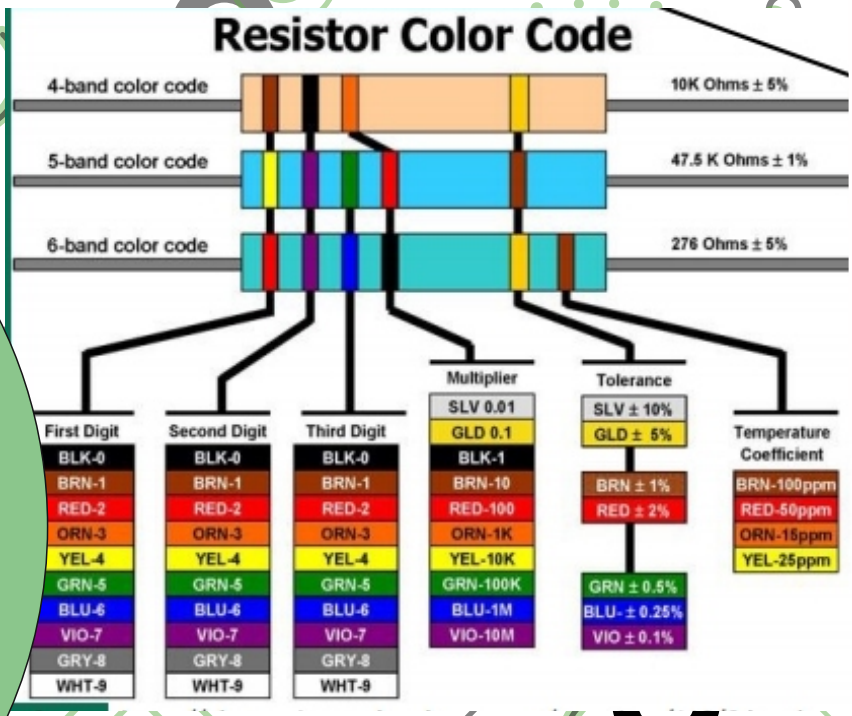
Arduino Reset -You can reset your Arduino board i.e, start your program from the beginning. You can reset the UNO board in two ways. The first, by using the reset buton (17) on the board.
Second you can connect an extenal
reset buttom to the arduino pin labelled RESET(5)

Pins(3.3,5, GND, Vin)

3.3V(6)-supply 3.3 output volt
5V(7)- Supply 5 output volt
Most of the components used with Arduino board works fine 3.3 volts and 5 volts.

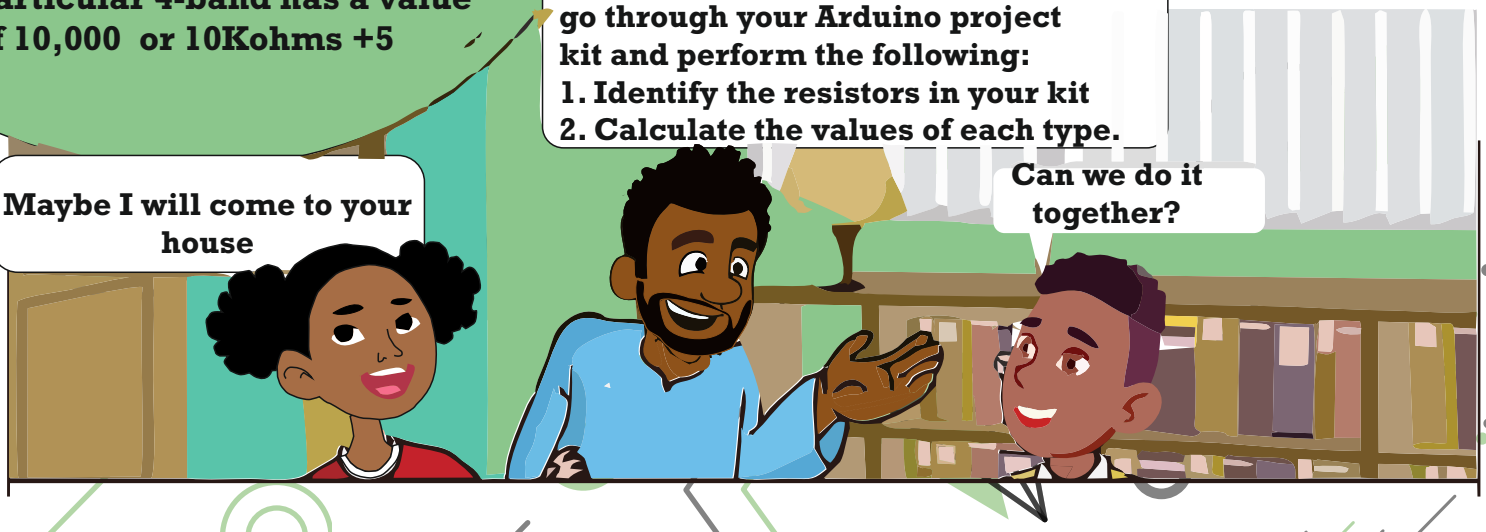


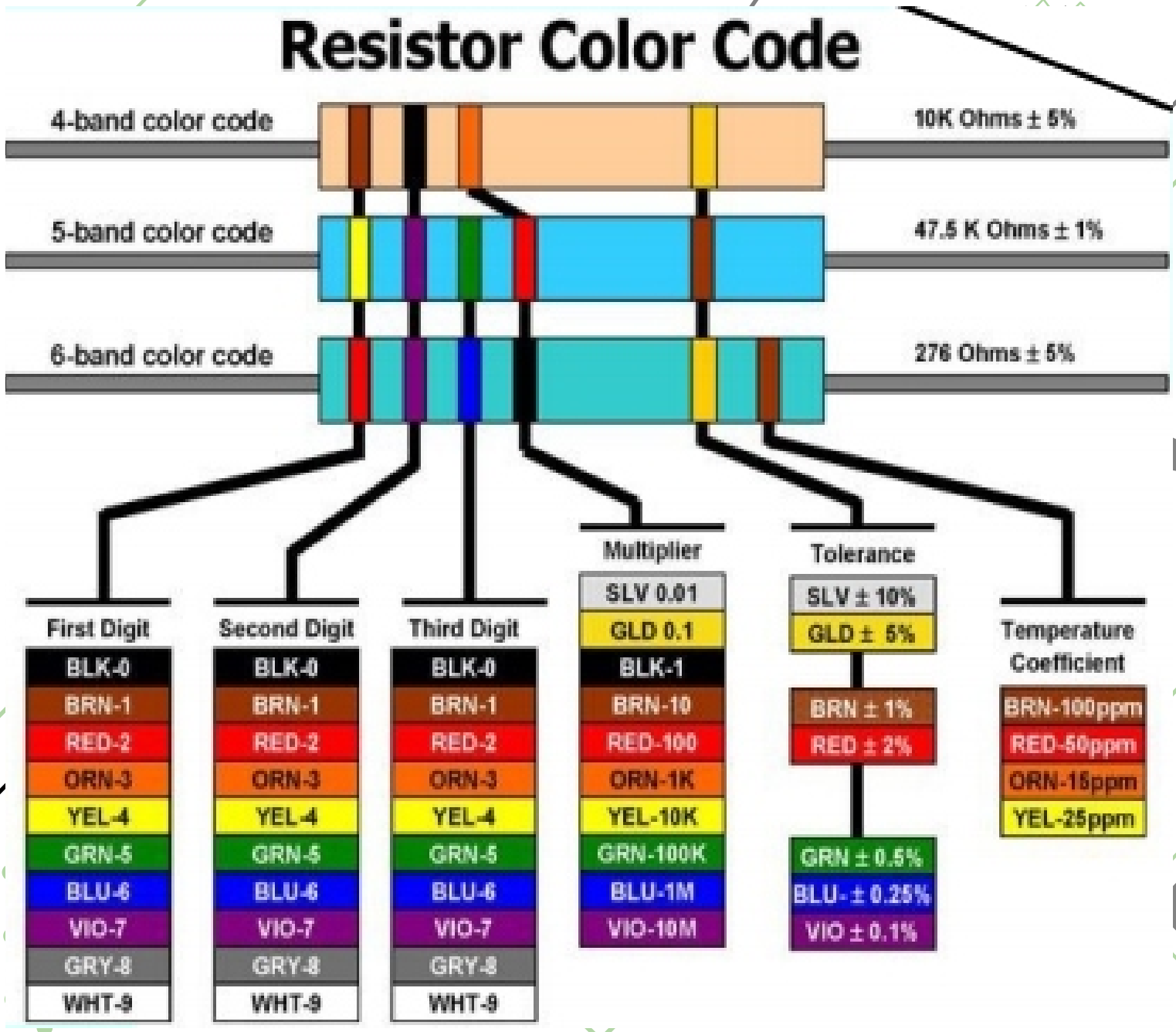
Before we proceed further, it is worthwhile to read a resistor



Resistors are measured in ohms. The values of resistors are represented by colour codes.As Presented as follows The resistors are of 4-band (colour) , 5-band or 6-band. In the 4-band resistor, the first two colours indicate the first two values, while the third band indicates the numbers of Zeros. The last band represents tolerance in the picture here, Gold has a value of 5% which means that the resistor can tolerate a load of plus or minus 5%, Therefore, this particular 4-band has a value of 10,000 or 10Kohms +5

For your assignment, go through your Arduino project kit and perform the following:
1. Identify the resistors in your kit
2. Calculate the values of each type.





Source:https://electronics.stackexchange.com/questions/119/6-brand-resistors-which-way-should-the-bands-be-read.

Getting to know your tools

We are a step closer to real Fun. In order to build your toys, you will need to download, install and setup the ARDUINO IDE on your computer.

We are going to use your computer

Oh I know how to use one

But Uncle! What is an IDE

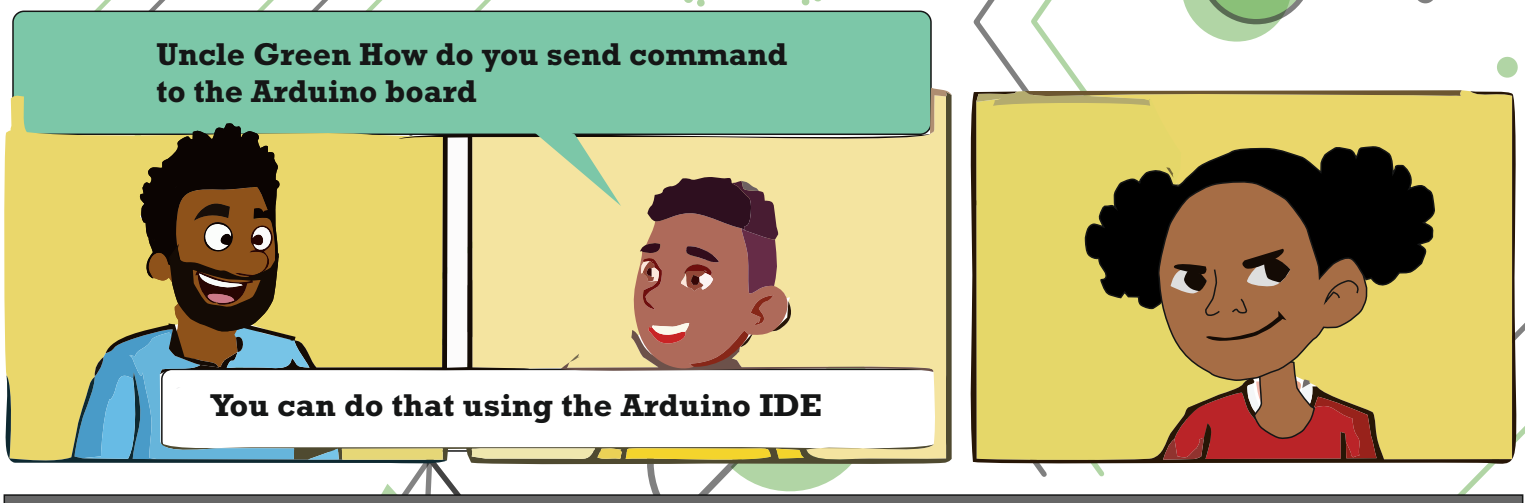
Good Question
IDE means Integrated Development Environment, and it is a software that tells the Arduino board what you want to do. As a developer you will be writing and testing software. So the IDE allows you to write and test your code

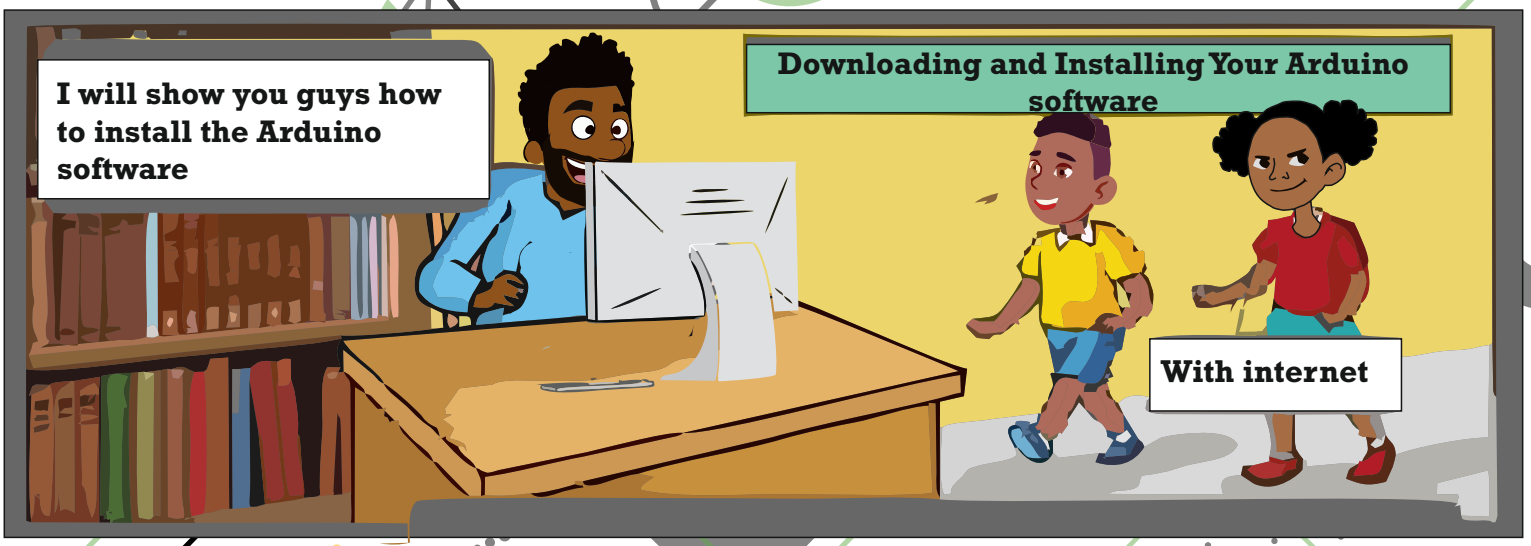
I know you would also like to know what a code is ?
A code is any collection of computer instructions, possibly with comments, written using human-readable programming language.

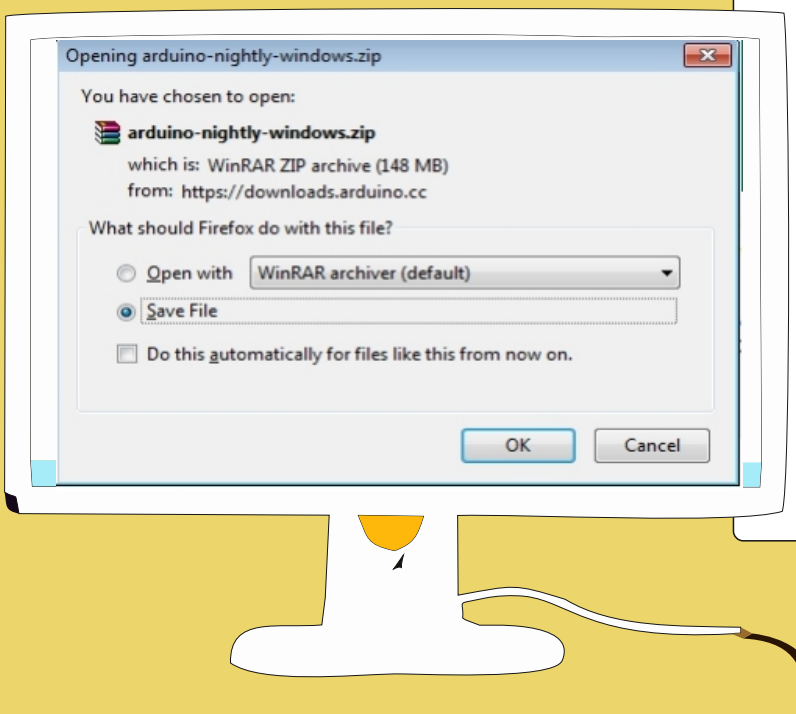
Episode 1.2

Getting to know your tool- How to Download, Install and set up the Arduino IDE software

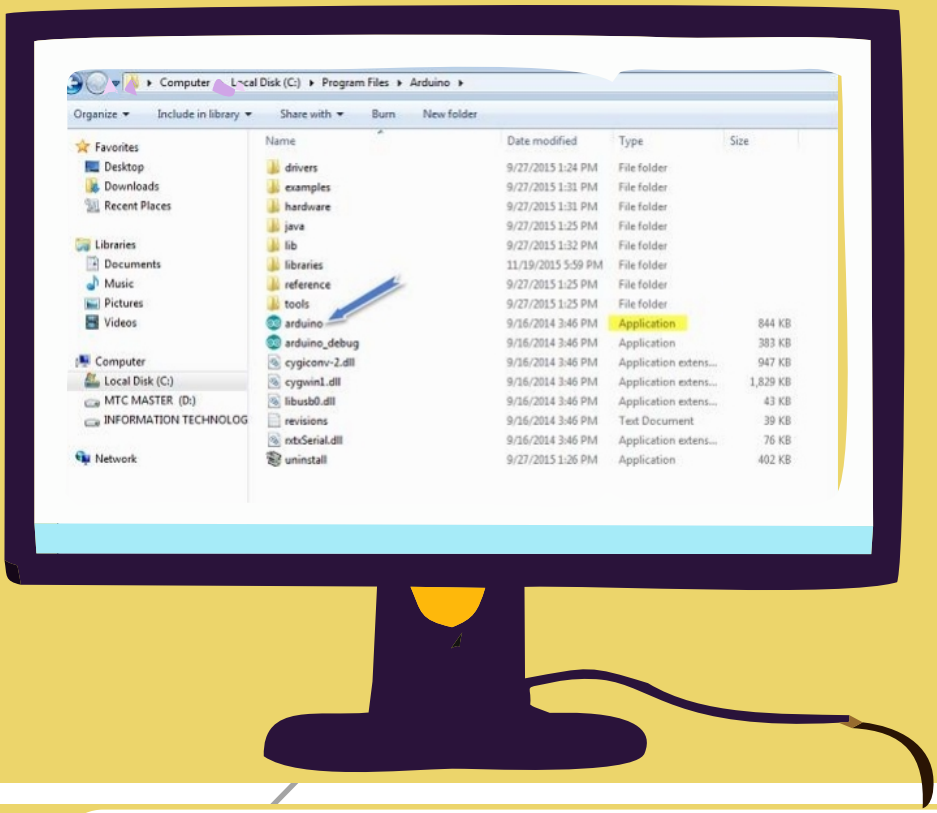
How to Download, Install and set up the Arduino IDE software







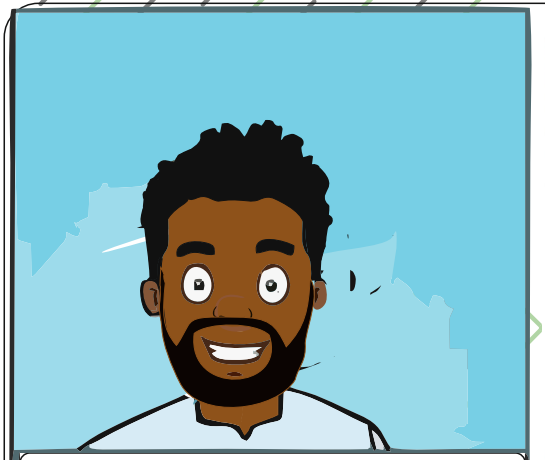
STEP 1.
Now we want to download the IDE, click the link- https://www.arduino.cc/en/Main/Software



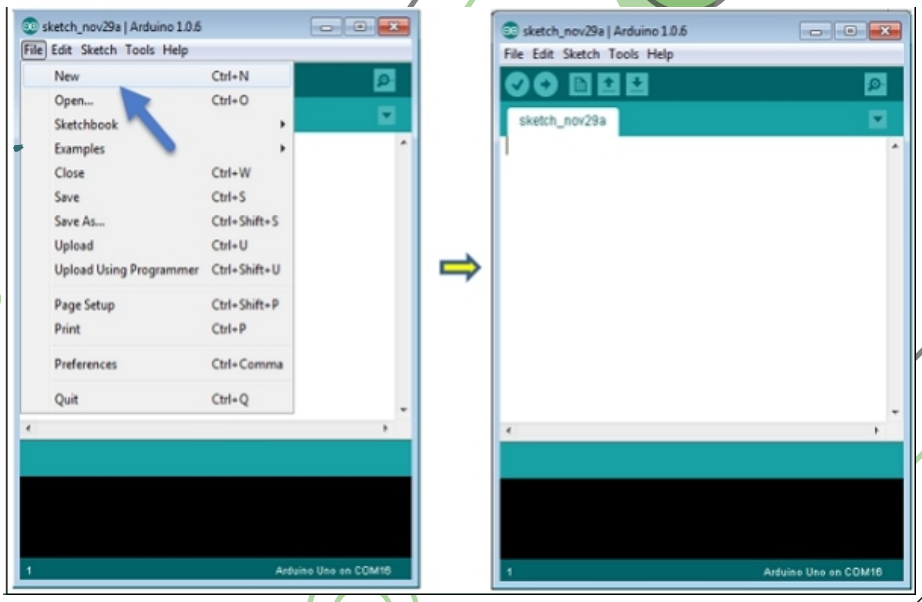
Step 2: After the download the next step is to power your Arduino Board.
The Arduino Uno automatically draw power from either, the USB extension to the computer or an external power supply.To power up your Arduino, connect the Arduino board to your computer using the USB cable. The green power LED(Labelled PWR)should glow.

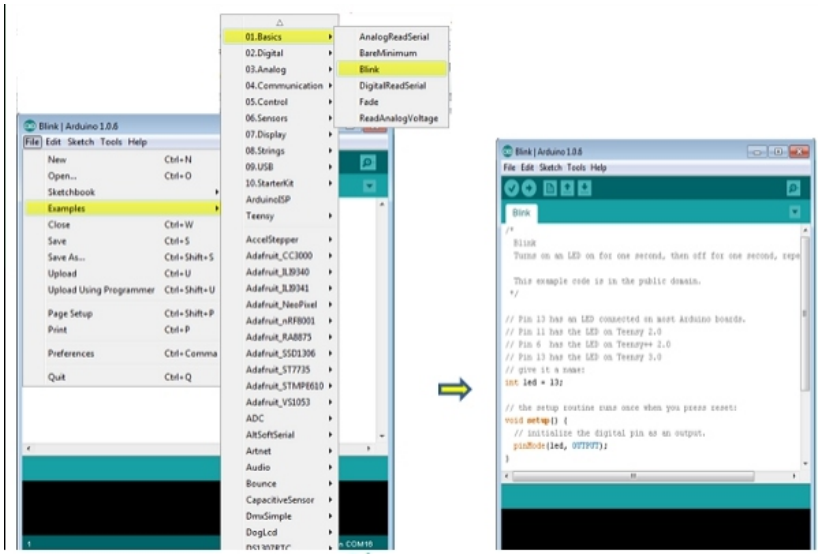
Step 3: Now launch the Arduino IDE
After you download your IDE, you need to unzip the folder. Inside the folder, you would find the application icon with and infinity( )label(application.exe).Double-click the icon to start the IDE. As shown

Now it is time to open your first project.
Once the software starts, you have two options,
(1)Create a new project,
or
(2) Open an existing project
example.
The next picture explains how to do these.
But before you can create your project, you need to do step 5 and 6. So do these first then create your project.

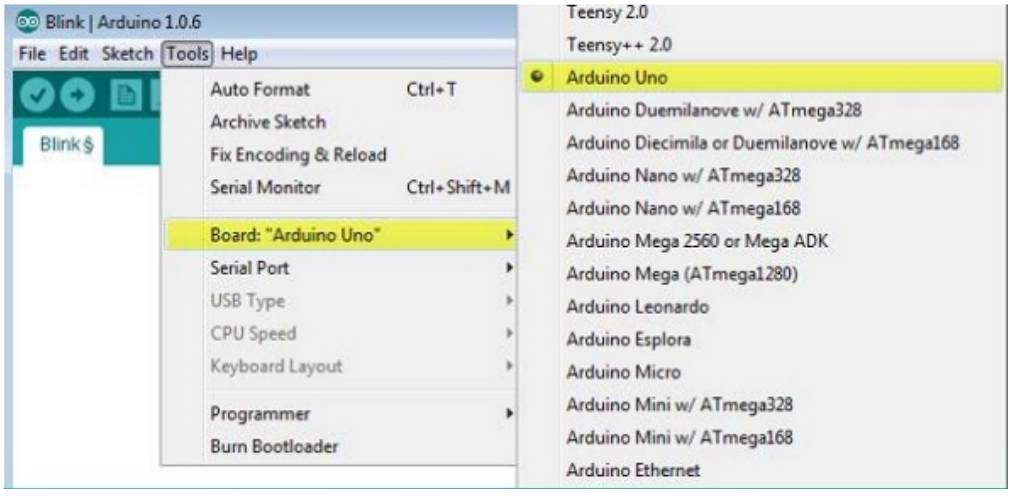


Step 4:1 To create a new project,
Select File New.
As shown



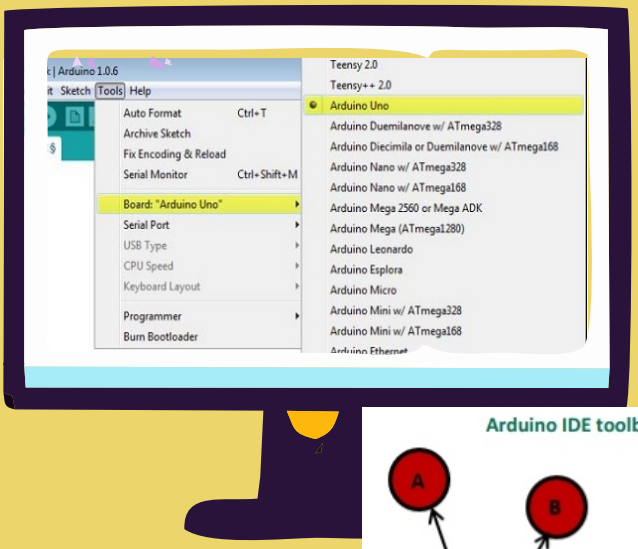


Step 4.2: To open an existing project example,
select File Example Basics Blink. Here, you are selecting just one of the examples with the name Blink. This turns the LED on and off with some time delay. If you like, you can select any other example from the list.



Go to Tools ➡
Board and select your board

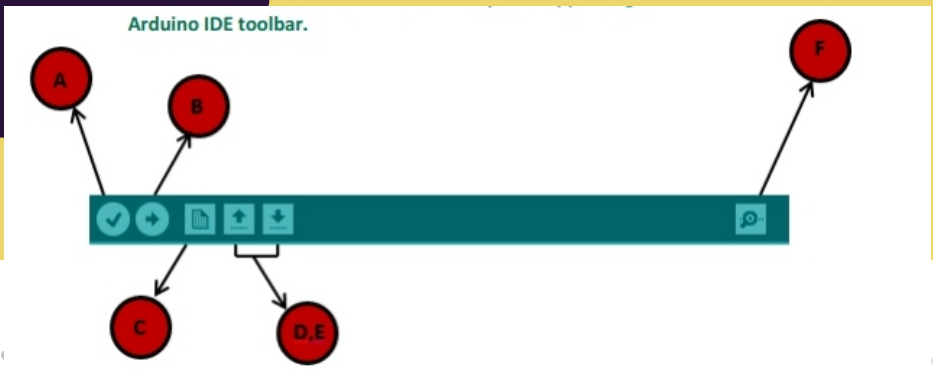
Step 5: Select your Arduino board. To avoid any error when uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.



Step 5: Select your Arduino board. To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools ➔ Board and select your board.

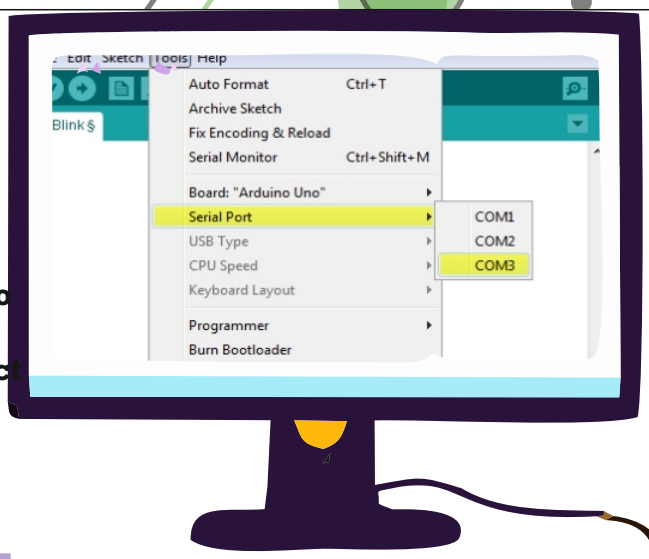
In the picture below, we have selected Arduino Uno board according to our tutorial, but if you are using another Arduino board you must select the name that matches the board you are using.



A- Used to check if there is any compilation error.
B- Used to upload a program to the Arduino board.
C- Shortcut used to create a new sketch.
D- Used to directly open one of the example sketch.
E- Used to save your sketch.
F- Serial monitor used to receive serial data to the the board.
Now, simply click the "Upload" button in the environment. Wait a few seconds you will see the RX and TX LEDs on the board, Flashing if the upload is successful, the message "Done uploading" will appear in the status bar.

Step 6: Select your serial port

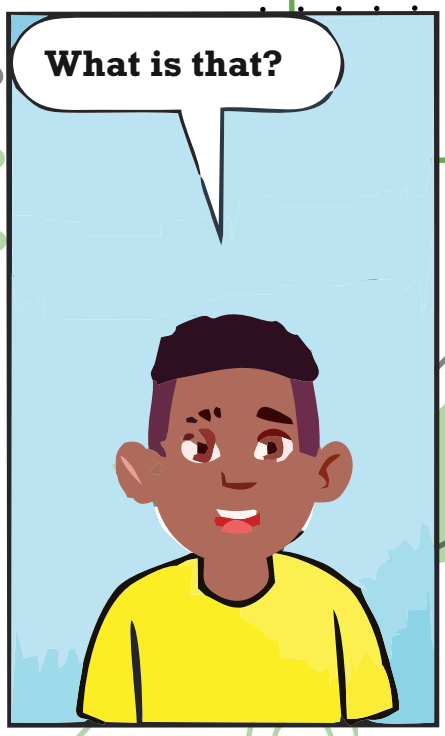
**Select the serial device of the Arduino board
Go to Tools--- Serial Port Menu.
This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be the Arduino board. Reconnect the board and select the serial port.**



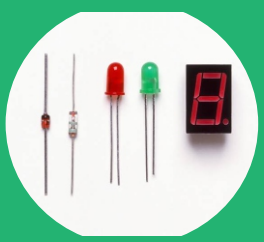
Piezo
This is an electrical component that can be used to detect vibrations and create noises.

Resistors
This resist flow of electrical in a circuit, changing the voltage and current as a result. Resistor values are measured in ohms (represented by the greek Omega character: Ω) The coloured stripes on the sides of the resistors indicate their value(see resistor colour table)

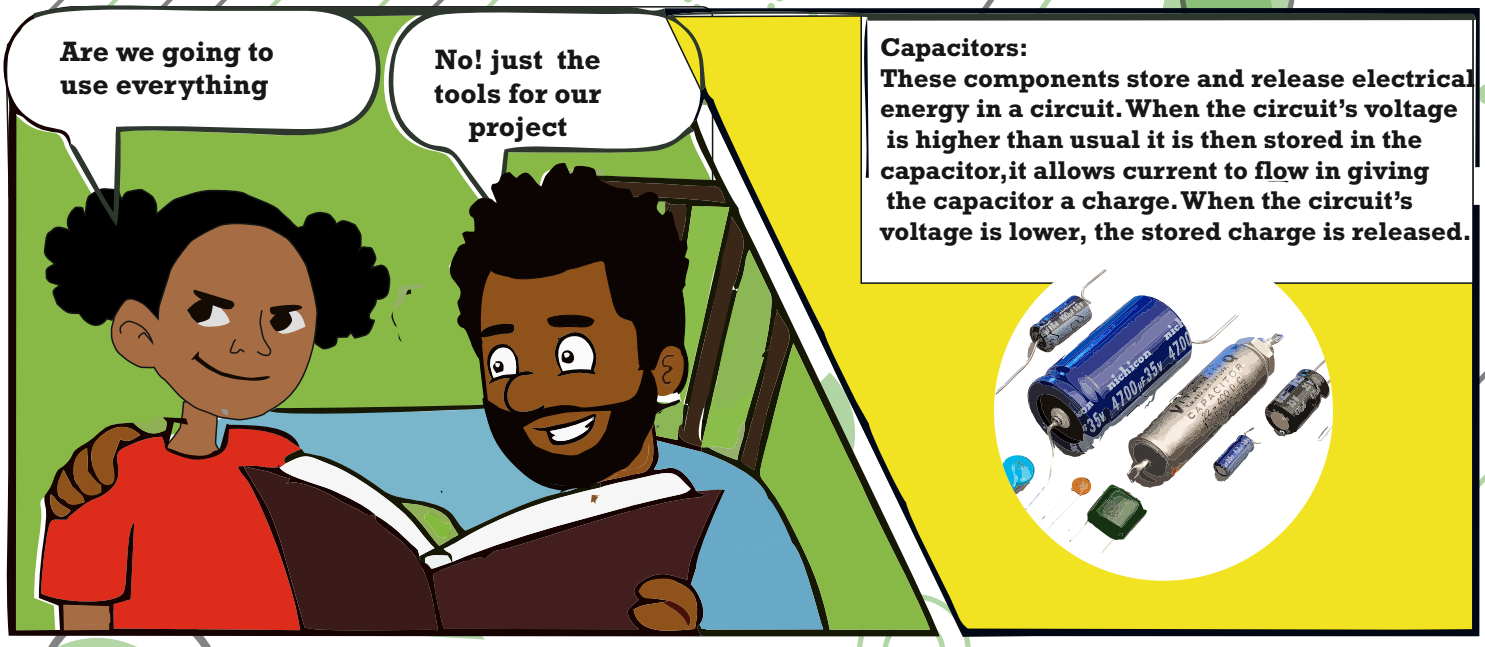
Arduino Uno
This picture shows the Arduino Uno microcontroller development board that will be at the centre of your projects. It's a simple computer, which you will use to build circuits and interfaces for interaction, and to tell the micro controller how to interface with other components.



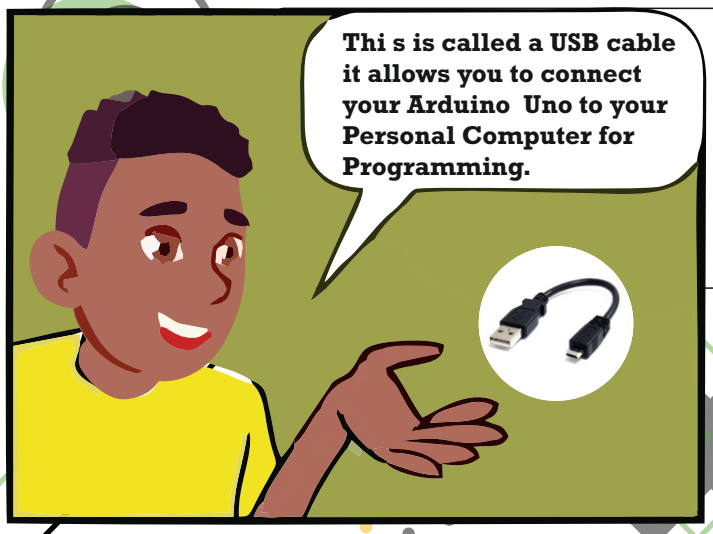
DC motor:
This converts electrical energy to mechanical energy when electricity is applied to its leads. Coils of wire inside the motor become magnetized when current flows through them

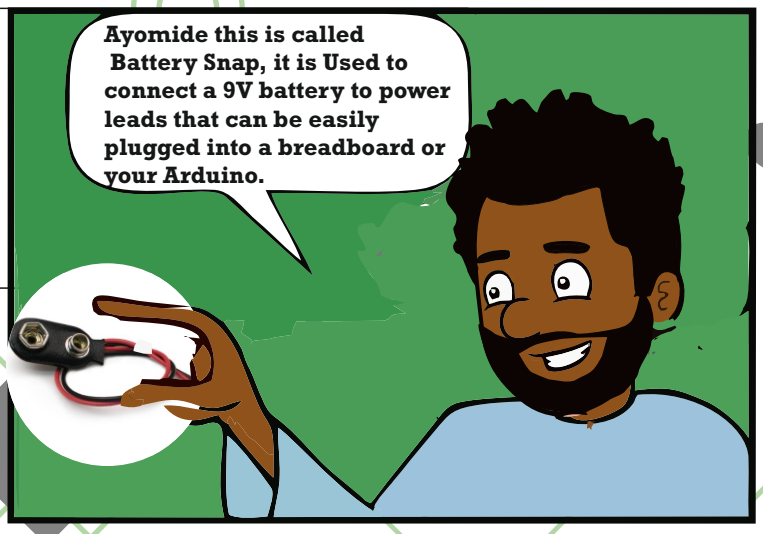


Light Emitting Diodes(LEDs)
A type of diode that illuminates when electricity passes through it.

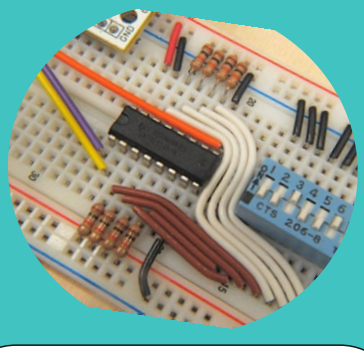


Capacitors:
These components store and release electrical energy in a circuit. When the circuit's voltage is higher than usual it is then stored in the capacitor, it allows current to flow in giving the capacitor a charge. When the circuit's voltage is lower, the stored charge is released.





Servo Motor
A type of geared motor that can only rotate 180 degrees. It is controlled by sending electrical pulses from your Arduino. These pulses tell the motor what position it should move to power to the Arduino for most of the projects in the kit.

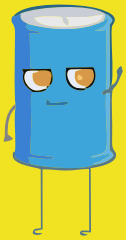


Breadboard:
Though the name sounds funny but it is a kind of bread that cannot be chewed. A board on which you can build electronic circuits. It's like a patch panel

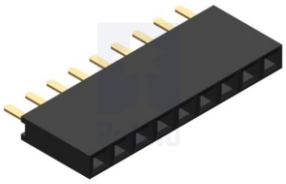
Tilt sensor
It's a type of switch that will open or close depending on its orientation. typically, they are hollow cylinders with a metal ball in-side that will make a connection across two leads when tilted in the particular direction.

TIP

A resistor is a device that restrict the flow of electric current in a system



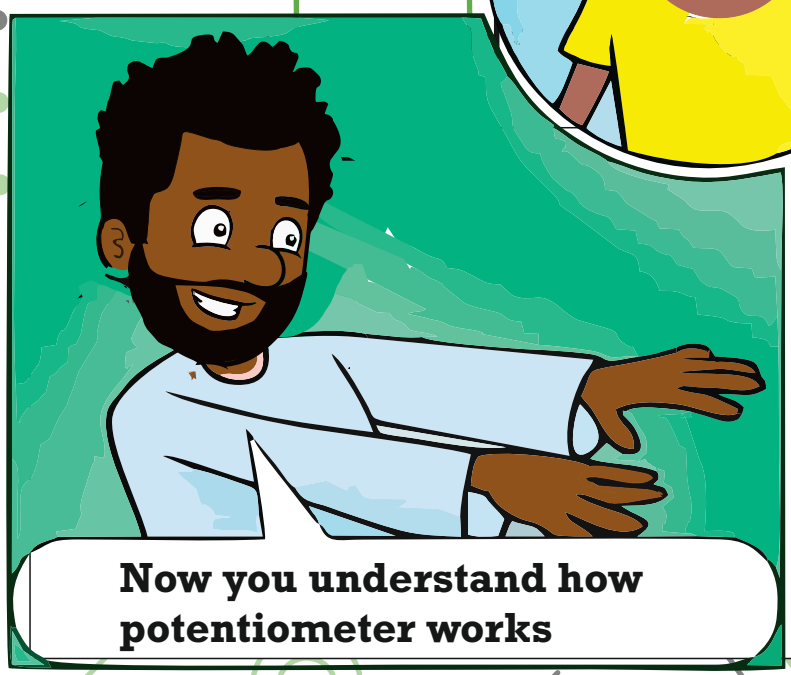
Push Button
It momentary switches that close a circuit when pressed. They snap into bread-boards easily. These are good for detecting on/off signals

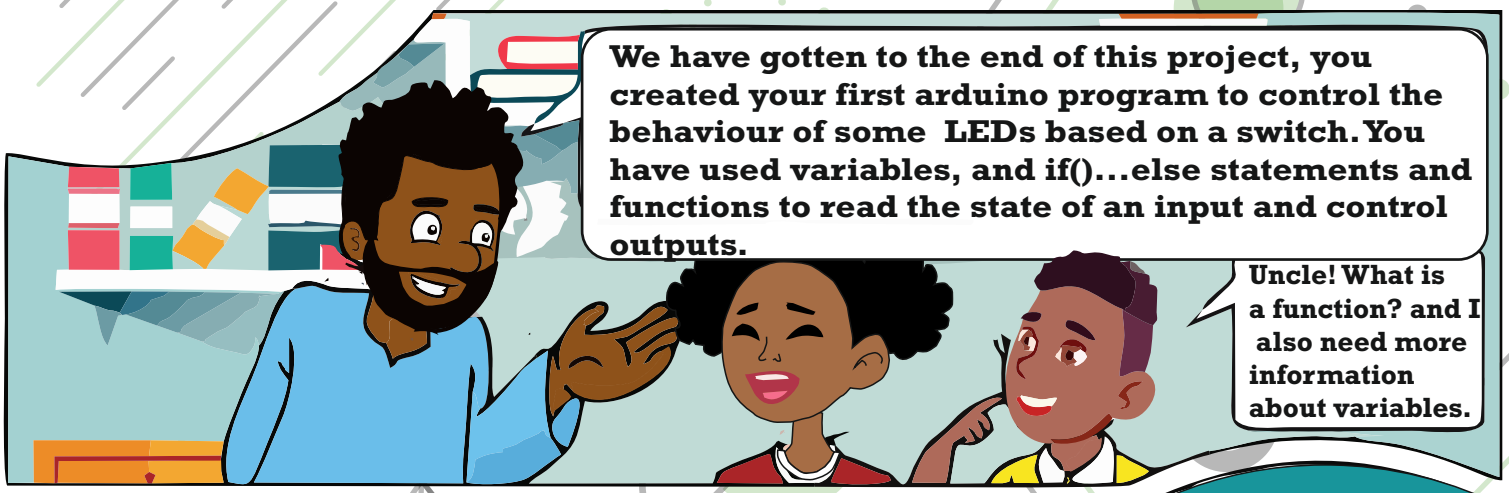


Male header pins

Jumper Wires are used to connect components to each other on the breadboard, and to the Arduino.

Potentiometer
This is a variable resistor with three pins, Two of the pins are connected to the ends of a fixed# resistor. The middle pin, or wiper, moves across the resistor, dividing it into two halves. When the external sides of the potentiometer are connected to voltage and ground, the middle leg will give the difference in voltage as you turn the knob.





#TIP

OHM'S LAW

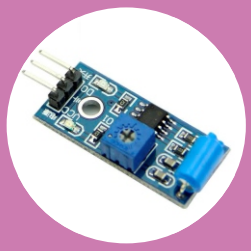
CURRENT = VOLTAGE/RESISTANCE
(I = V/R)

OR

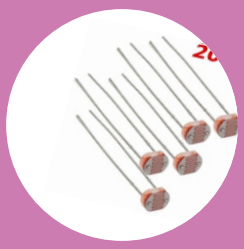
RESISTANCE = VOLTAGE/CURRENT
(R = V/I)

OR

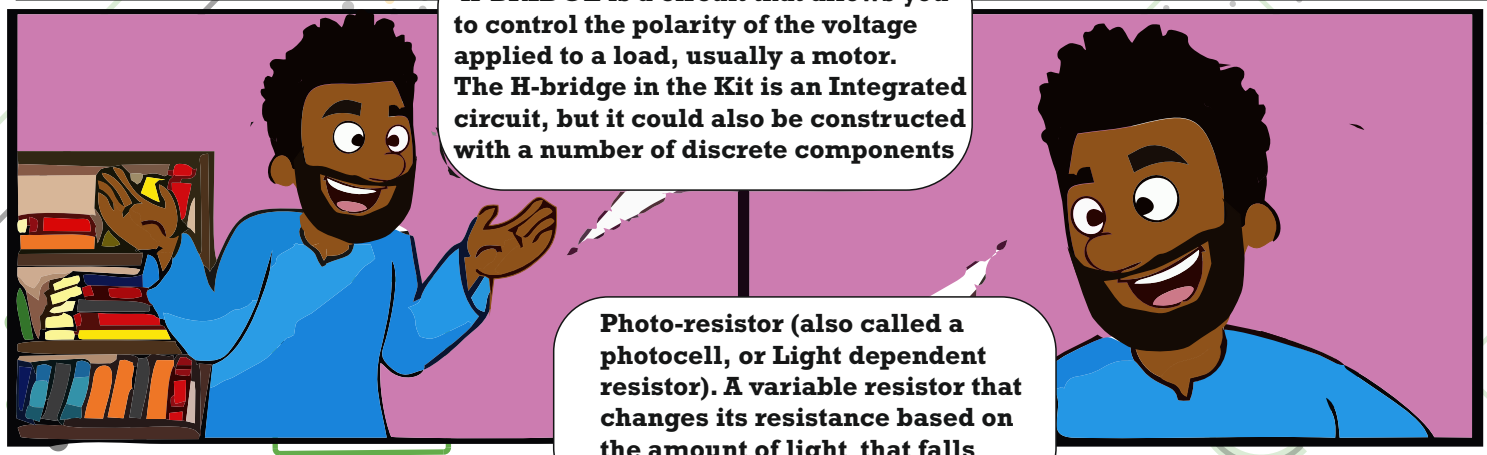
VOLTAGE = RESISTANCE * CURRENT
(V = R*I)

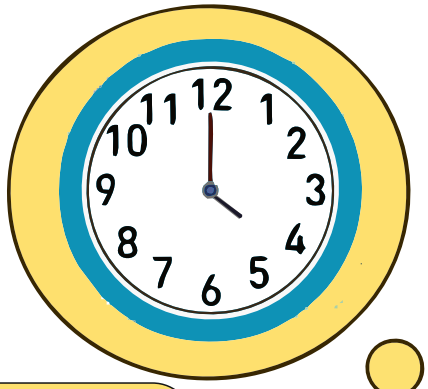


H-bridge



Photoresistors





It is 4^{o} Clock already

Practice Question

What does IDE stand for?

answer choices

- [ ] In Deep Environment
- [ ] Internal Deep Escape
- [ ] Integrated Development Environment
- [ ] Integrated Device Environment

Episode 2

Building a simple circuit

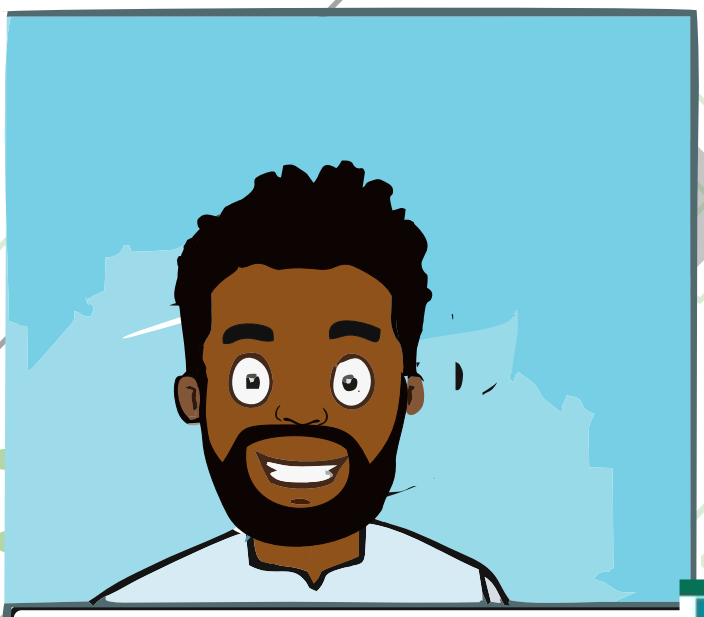
The Basic Electrical Theory

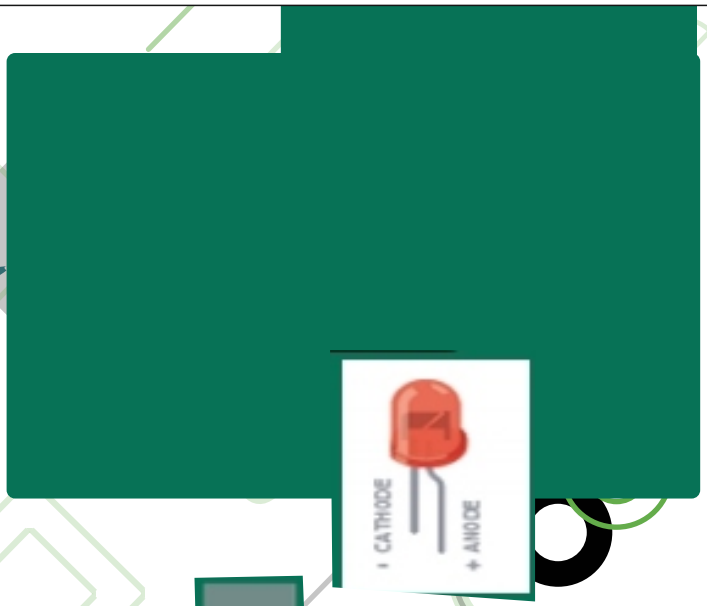
Electricity is a type of energy, much like heat, gravity or light. Electrical energy flows through conductors, like wire. You can convert electrical energy into other forms of energy to do something interesting, like turn on a light or make some noise out of a speaker. The components you might use to do this, like speakers or light bulbs, are electrical TRANSDUCERS. Transducers change other types of energy into electrical energy and vice versa. Things that convert other forms of energy into electrical energy are often called SENSORS, and things that convert electrical energy into other forms of energy are sometimes called ACTUATORS. You will be building circuits to move electricity through different components. Circuits are closed loops of wire with a power source (like a battery) and something to do something useful with energy, called a load.

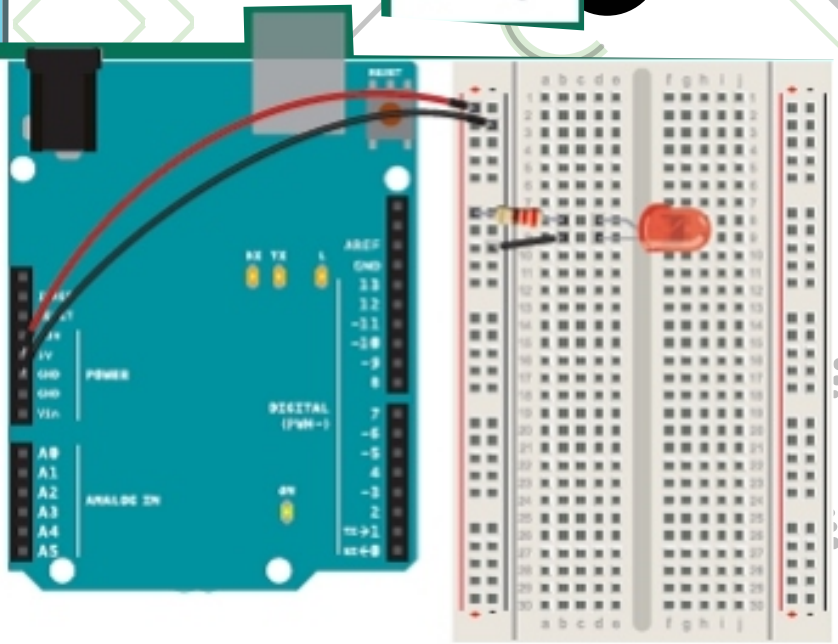
In a circuit, electricity flows from a point of higher potential energy(usuallyreferred to as power or +) to a point of lower potential energy in a circuit. In the circuits you are building, electricity only flows in one direction. This type of circuit is called direct current, or DC. In alternating current (AC) circuit electricity changes its direction 50 or 60 times a second (depending on where you live). This is the type or electricity that comes from a wall socket.

There are a few terms you should be familiar with when working with electrical circuits. Current(measured in amperes, or amps: with the A symbol) is the amount of electrical charge flowing past a specific point in your circuit. Voltage (measured in volts; with the V symbol) is the difference in energy between one point in a circuit and another finally, Resistance (measured in ohms, with the Ω symbol) is how much a component resist the flow of electrical energy.

-There needs to be a complete path from the energy source(power) to the point of least energy (ground) to make a circuit. If there's no path for the energy to travel, the circuit won't work.
-All the electrical energy gets used up in a circuit by the components in it. Each component converts some of the energy into another form energy. In any circuit, all of the voltage is converted to another form of energy (light, heat, sound, etc.)
-The flow of current at a specific point in a circuit will always be same coming in and going out.
-Electrical current will seek the path of least resistance to ground. Given two possible paths, more of the electrical current will go down the path with less resistance. If you have a connection that connects power and ground together with no resistance, you will cause a short circuit, and the current will try to follow that path. In a short circuit, the power source and wires converts the electrical energy into light and heat, usually as sparks for an explosion. If you've ever shorted a battery and seen sparks, you will know how dangerous a short circuit can be.

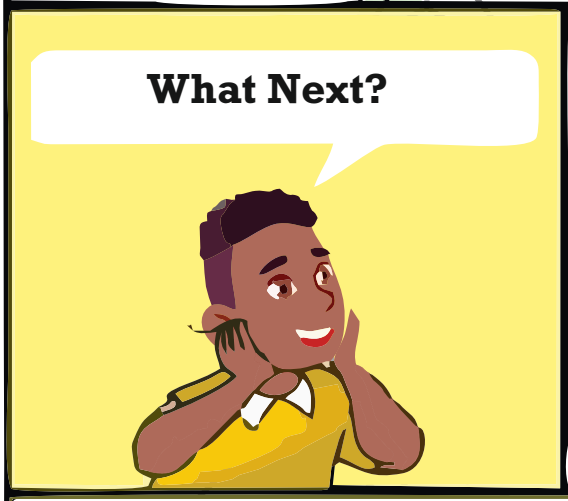






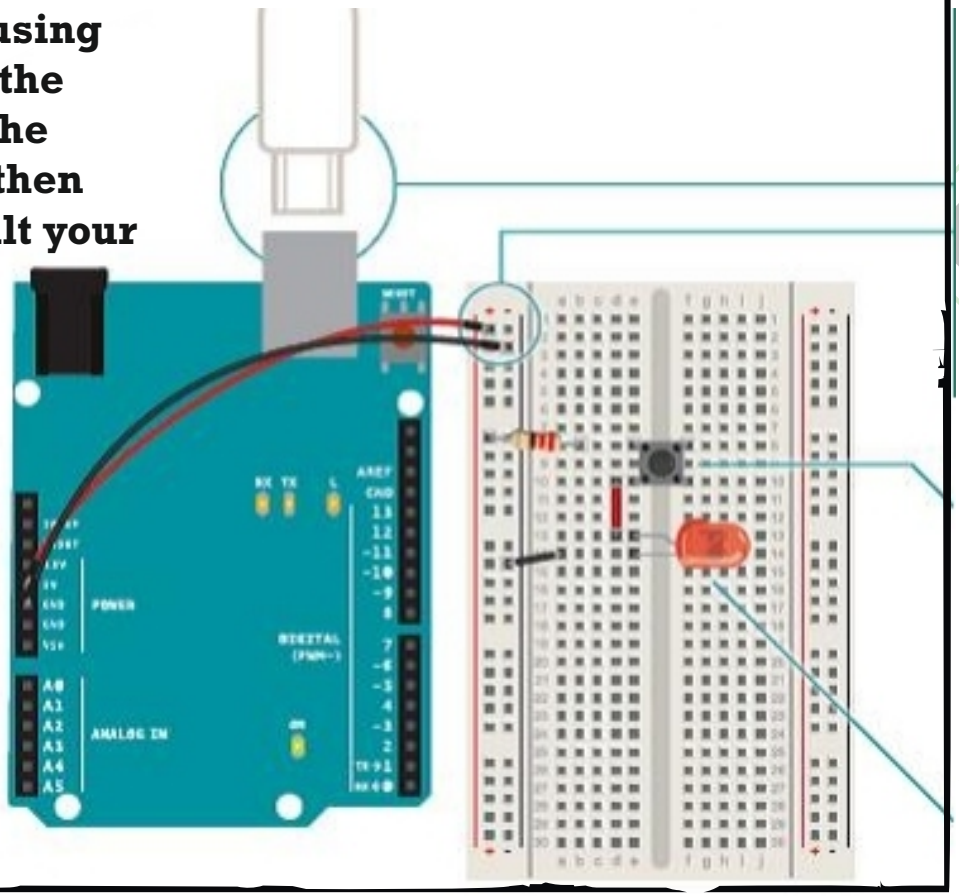
The diagram shows a simple circuit made from three jumper cables, a 220 ohm resistor and a LED. Please go to the information about each of these components. Don't forget that the long leg of the LED is the positive side(+) called anode, while the short leg is the negative (-) side call cathode

One major warning for you to note. Always unplug your Arduino from the power source before building the circuit



Connect the jumper cables, push button, resistor and LED (with a jumper wire connect the long leg of the LED to the anode, and the short leg of the LED to the ground) from the Arduino board to the breadboard as shown in this picture.

Once you are done connecting them, now connect your Arduino to the computer using the USB cable. Now press the push button to switch on the LED . I the LED lights up then you have successfully built your first circuit. Congratulations

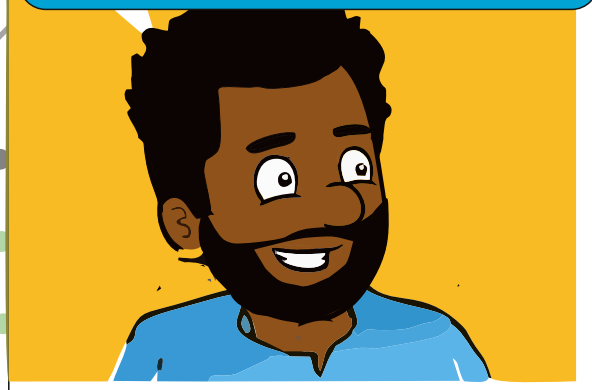


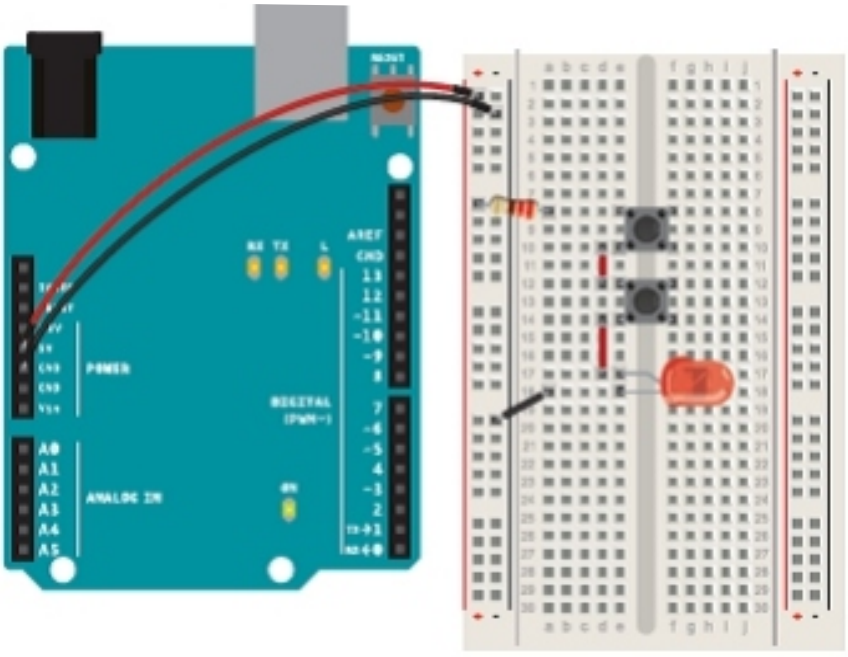
Once again do not forget to unplug the Arduino board from the computer before making any changes to the breadboard

Series circuit
- Components in SERIES come one after another

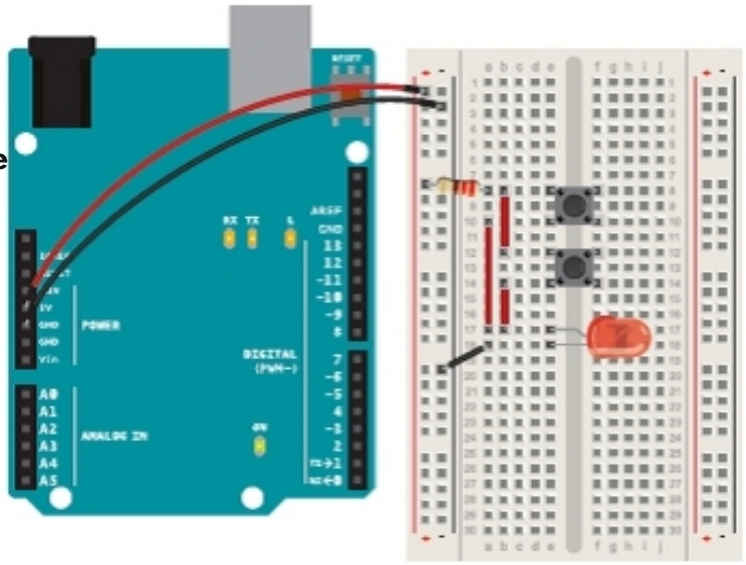
To build this remove your power source, add a with next to the one already on your breadboard. Wire them together in series as shown below. Connect the LED cathode to ground, Power up the Arduino again. Now turn on the LED, you will need to press both switches. Since these are in series, they both need to be closed for circuit to be completed.

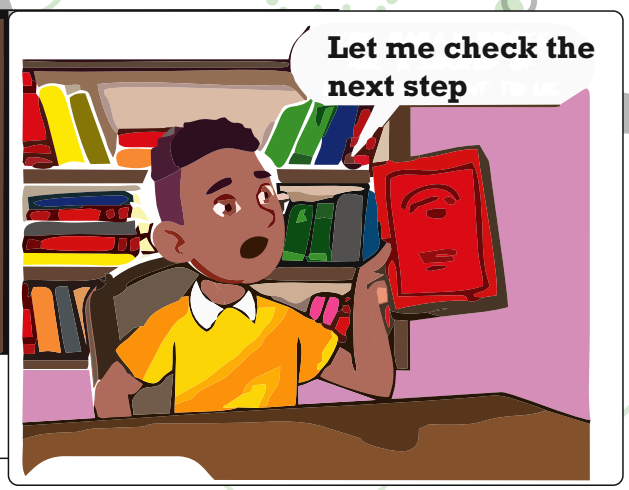
After successfully building your first circuit. Now you will learn the differences between the two types of circuit you can build. Which are series circuit and a parallel circuit.





Parallel Circuit
- Component in PARALLEL run side by side
To wire up switches in parallel, keep the switches and LED where they are, but remove the connection between the two switches. Wire both switches to resistor. Attach the other end of both switches to the LED, as shown in the diagram on the right hand side. Now when you press any of the buttons, the circuit is completed and the light turns on.





You have learned about the basic electrical theory, that is, properties of voltage, current and resistance while building a very simple circuit on a breadboard. With basic components like led, resistor and switches, you have created the simplest interactive system where the user presses the button in order to switch on the light. these fundamental knowledge of working with electronics will be built upon in the following project. in the next project will be building a spaceship interface so you can explore and discover the galaxy. see you then.

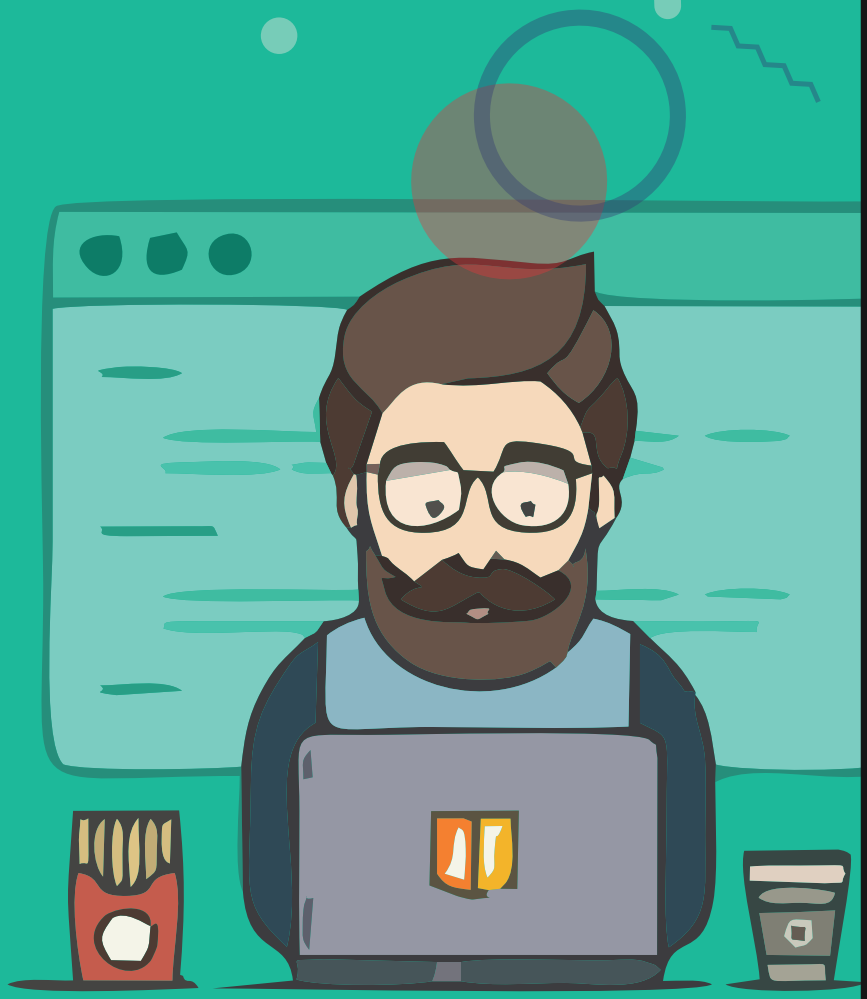
Practice Question

How do you give instructions on what the Arduino should do in an IDE?

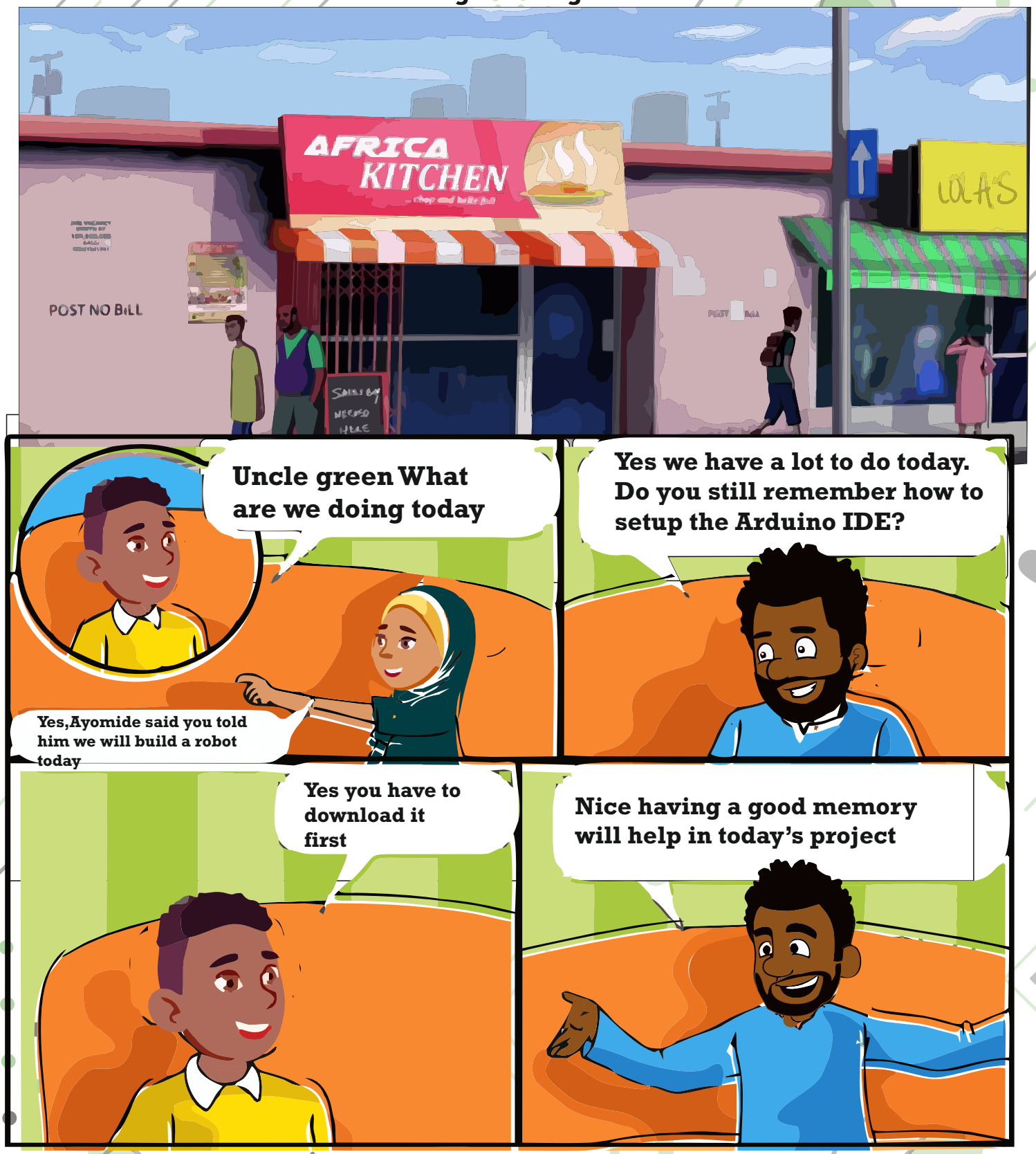
answer choices

- ☐ Editor
- ☐ Sketches
- ☐ Console
- ☐ IDE error

GreenLab Cartoonino



Episode 3 Understanding the program interface



#TIP

An electronic circuit is composed of individual elctronic components, suchas resistors, transistors, capacitors inductors and diodes connected by conductive wires or traces through

Understanding the Program structure

Today I would like you understand the program interface and structure

Is it how to use the Arduino IDE

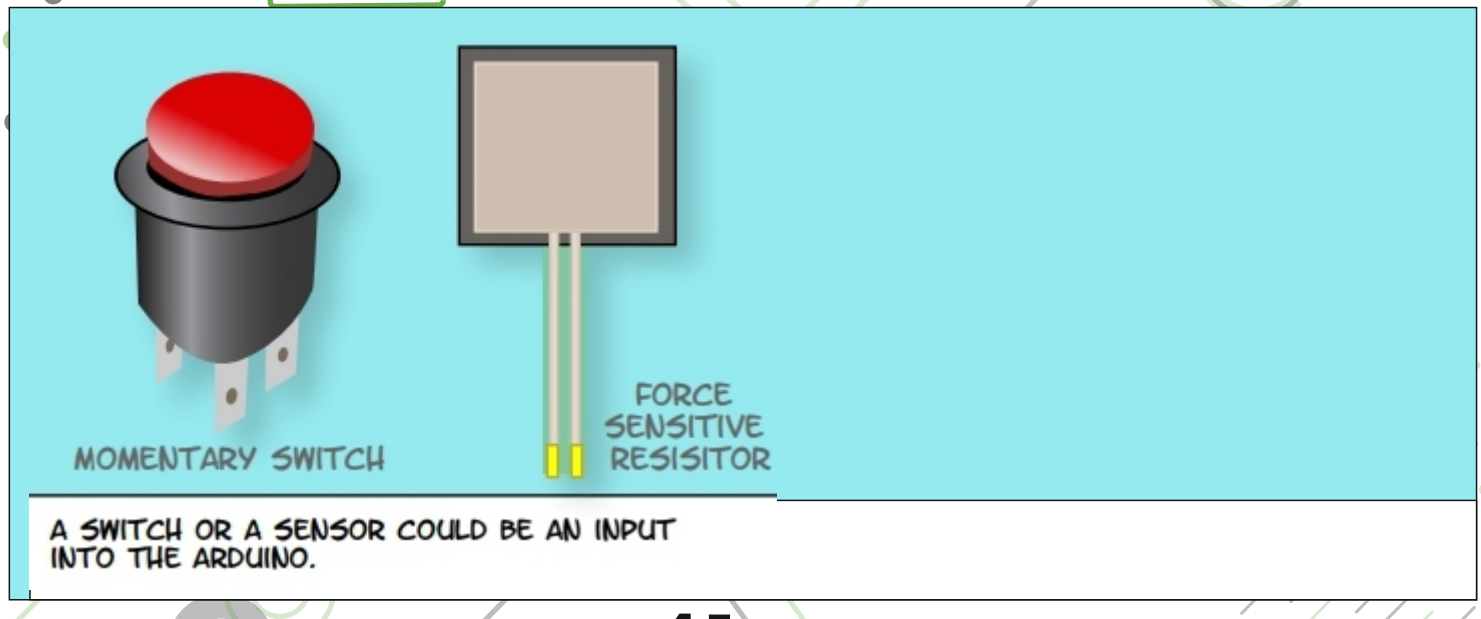
Uncle you would also teach us how to enter the code.

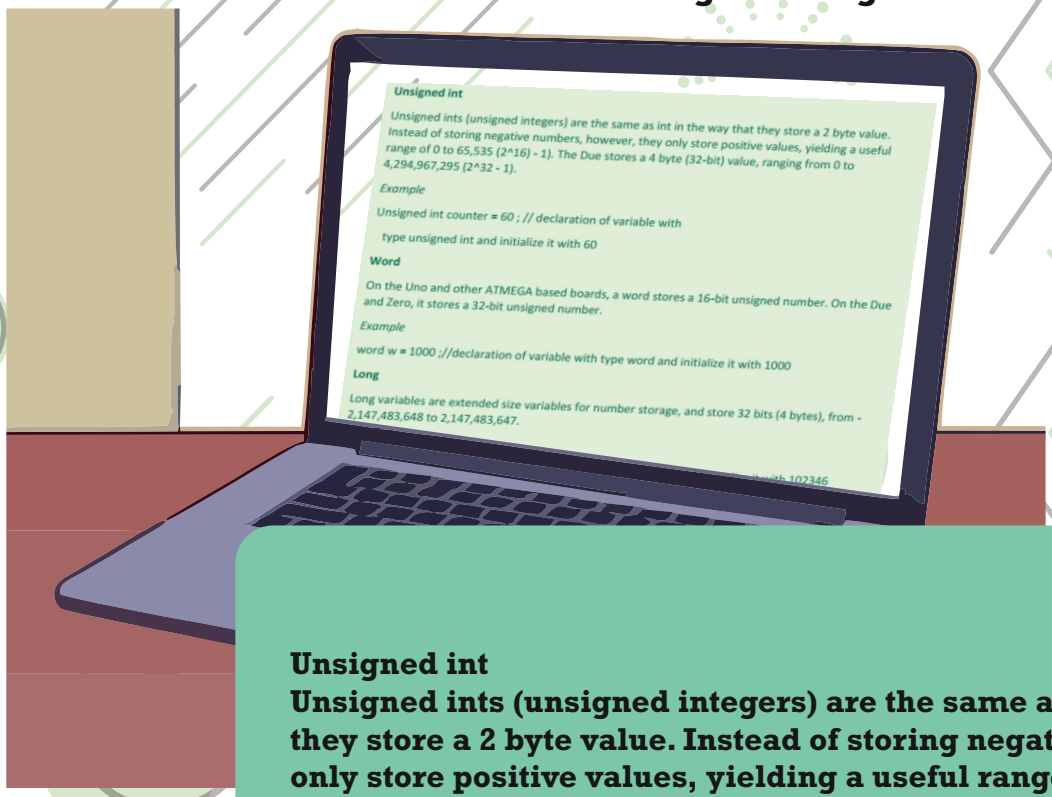
Now kids very soon you will visit the YouNiversity and so you need a laptop or phone with internet access.

My Mother has a laptop at home so I can practice

You see, without all these information, you will not be able to build your awesome toys and other prototypes properly.

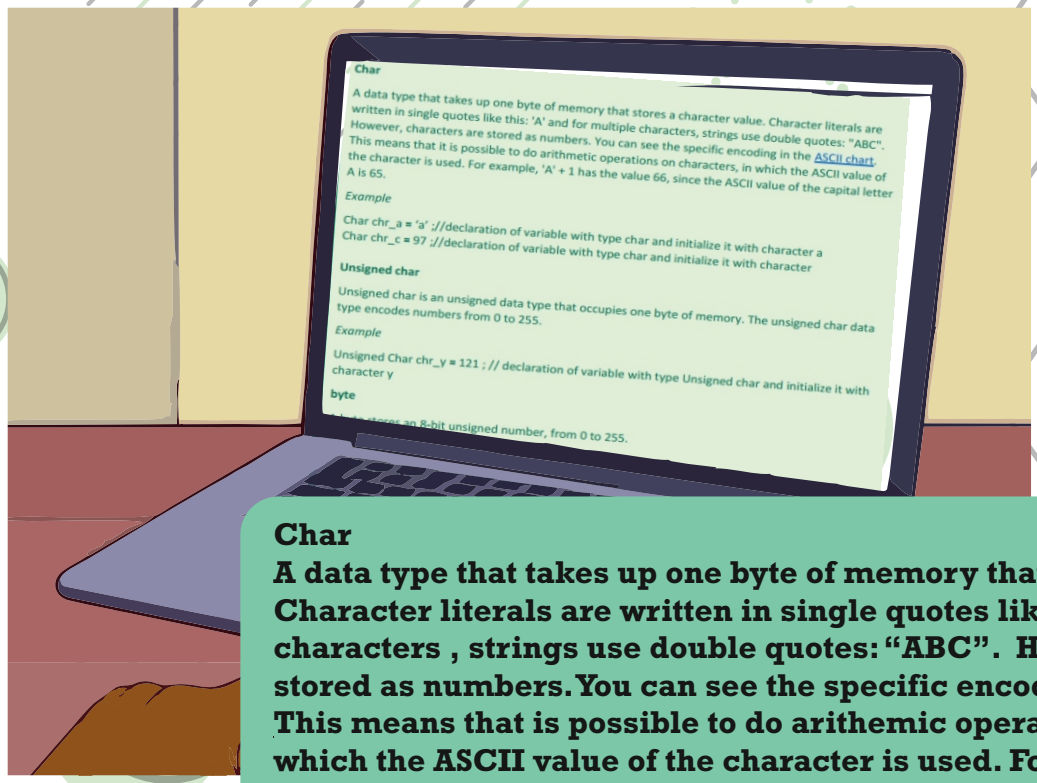
hmmmm!





Unsigned int
Unsigned ints (unsigned integers) are the same as int in the way that they store a 2 byte value. Instead of storing negative numbers, however, they only store positive values, yielding a useful range of 0 to 65,535(2^16)-1). The Due stores a 4 byte (32 bit) value, ranging from 0 to 4,294,967,295(2^32-1).
example

Unsigned int counter = 60 ; // declaration of variable with type unsigned int and initialize it with 60
Word
On the Uno and other ATMEGA based boards, a word stores a 16-bit unsigned number. On the Due and Zero, it stores a 32-bit unsigned number.
Example
word w = 1000 ;//declaration of variable with type word and initialize it with 1000
Long
Long variables are extended size variables for number storage, and store 32 bit (4 bytes), from - 2,147,483,647.
Example
Long Velocity = 102346 ;//declaration of variable with type long and initialize it with 102346
Unsigned long
Unsigned long variables are extended size variables for number storage and store 32 bits (4 bytes). Unlike standard longs, unsigned longs will not store negative numbers, making their range from 0 to 4,294,967,295 (2^32-1).
Example
Unsigned long Velocity = 101006 ;// declaration of variable with type Unsigned Long and intialize it with 101006
short
A short is a 16-bit data-type. On all Arduinos (ATMega and ARM based), a short stores a 16-bit(2-byte) value. This yields a range 0f -32,768 to 32,767(minimum value of -2^15 and a maximum value of (2^15)-1).
Example
short val = 13 ;//declaration of variable with type short and initialize it with 13



Char
A data type that takes up one byte of memory that stores a character value. Character literals are written in single quotes like this 'A' and for multiple characters , strings use double quotes: "ABC". However, character are stored as numbers. You can see the specific encoding in the ASCII chart. This means that is possible to do arithemic operations on characters, in which the ASCII value of the character is used. For example, 'A' +1 has the value 66, since the ASCII value of the capital letter A is 65.
Example
Char chr _a = 'a' ://declaration of variable with type char and initialize it with character a
Char chr_c = 97://declaration of variable with type char and initialize it with characecter

Unsigned char
Unsigned char is an unsigned data type that occupies one byte of memory. The unsigned char data type encodes numbers from 0 to 255.
Example
Unisigned Char chr _y = 121://decalaration of values with type Unsigned char and initialize it with character y

Byte
A byte stores an 8-bit unsigned number from 0 to 255.
Example
Byte m=25;//declaration of variab;e with type byte and inialize it with 25

int
Intergers are the primary data-type for number storage int stores a 16-bit (2-byte) value. This yields a range of -32,768 to 3,767(minimum value of -2^15 and a minimum. value of (2^15)-1):
The int size varies from board to board. On the Arduino Due, for example, an int stores a 32-bit(4-byte)value. This yields a range of -2,147,483,648 to 2,147,483,647 (minimum value of -2^31 and a maximum value of (2^31)-1).
Example
int counter =32 ;//declaration of variable with type int and initialize it with 32.

Operators

An Operator is symbol that tells the compiler to perform specific mathematical or logical functions. The following are the types of operators.

Arithmetic Operators

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

Example

```
void loop () {
  int a = 9,b = 4,c;
  c = a + b;
  c = a - b;
  c = a * b;
  c = a / b;
  c = a % b;
}
```

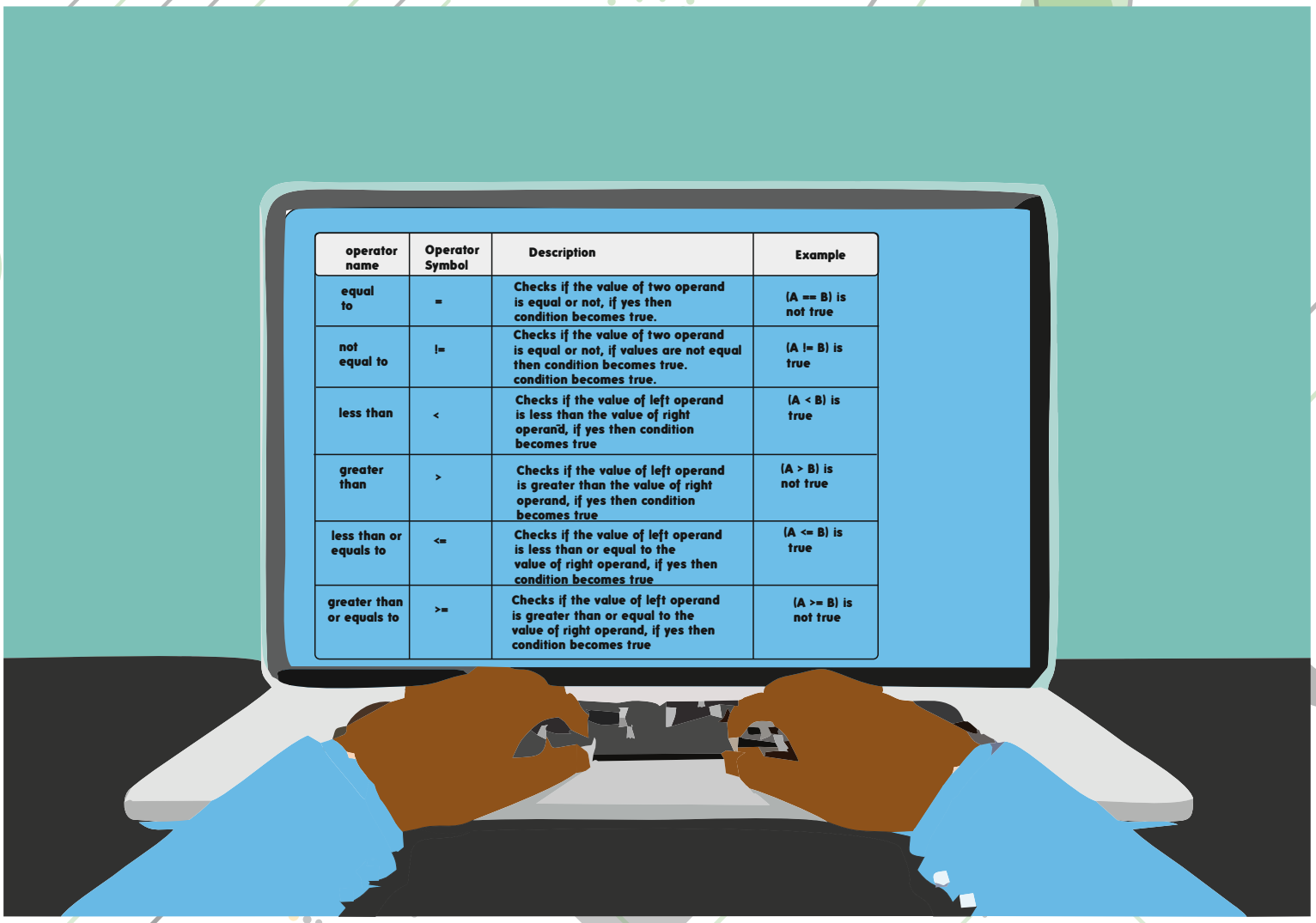
Result

```
a + b = 13
a - b= 5
a * b = 36
a / b = 2
Remainder when a divided by = 1
```

operator name	Operator Symbol	Description	Example
assignment operator	=	Stores the value to the right of the sign in the variable to the left of the equal sign.	A = B
addition	+	Adds two operands	A + B will give 30
subtraction	-	Subtracts second operand from the first	A - B will give -10
multiplication	%	Multiply both operands	A * B will give 200
division	/	Divide numerator by denominator an integer division	B / A will give 2
modulo	%	Modulus Operator and remainder of after an integer division	B % A will give 0

Comparison Operators

Operator table



operator name	Operator Symbol	Description	Example
equal to	=	Checks if the value of two operand is equal or not, if yes then condition becomes true.	(A == B) is not true
not equal to	!=	Checks if the value of two operand is equal or not, if values are not equal then condition becomes true. condition becomes true.	(A != B) is true
less than	<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true	(A < B) is true
greater than	>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true	(A > B) is not true
less than or equals to	<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true	(A <= B) is true
greater than or equals to	>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true	(A >= B) is not true

Bitwise Operators

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

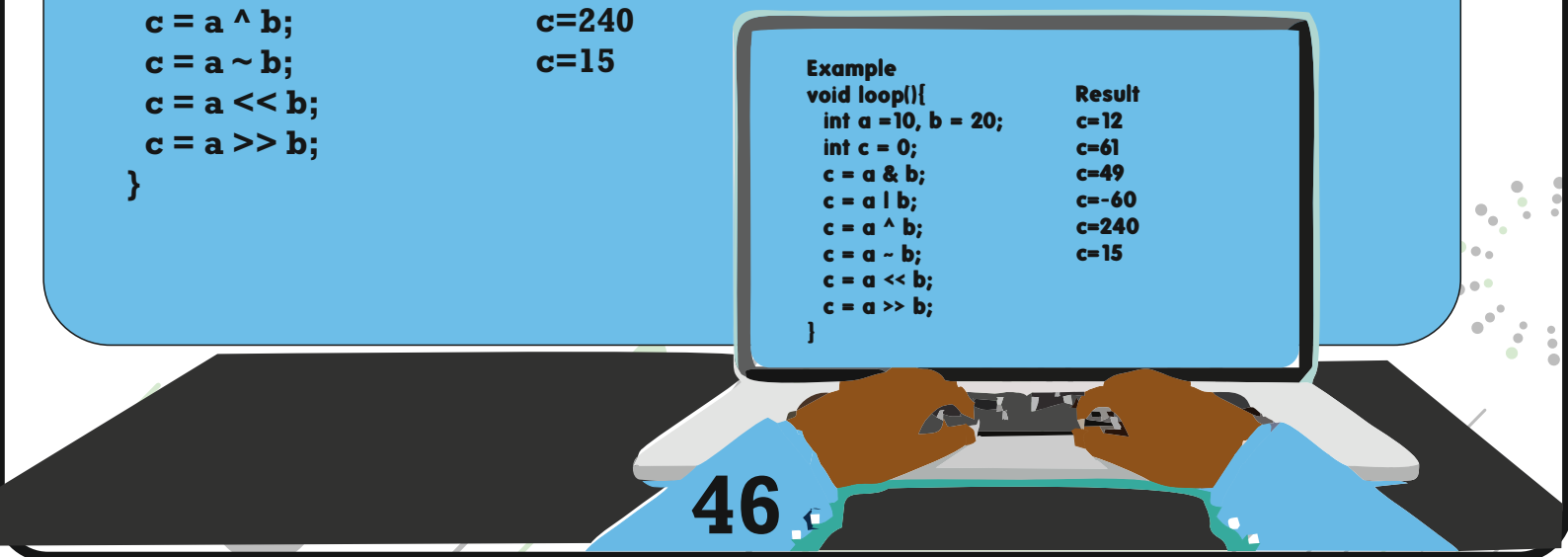
operator name	Operator Symbol	Description	Example
and	&	Binary AND operator copies a bit to the result if it exists in both operands	(A & B) will give 12 which is 0000 1100
or	l	Binary OR Operator copies a bit if it exists in either operand	(A l B) will give 61 which is 0011 1101
xor	^	Binary XOR Operator copies the bit if is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
not	~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -60 which is 1100 0011
shift left	<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
shift left	>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Example

```
void loop(){
  int a =10, b = 20;
  int c = 0;
  c = a & b;
  c = a l b;
  c = a ^ b;
  c = a ~ b;
  c = a << b;
  c = a >> b;
}
```

Result

```
c=12
c=61
c=49
c=-60
c=240
c=15
```



What is a variable a scope?

Variable in Arduino has 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, whixh is called LOCAL VARIABLES.
 # Inside the definition of function parameters, which is called FORMAL PARAMETERS.
 # Outside of all functions, which is called GLOBAL VARIABLES.

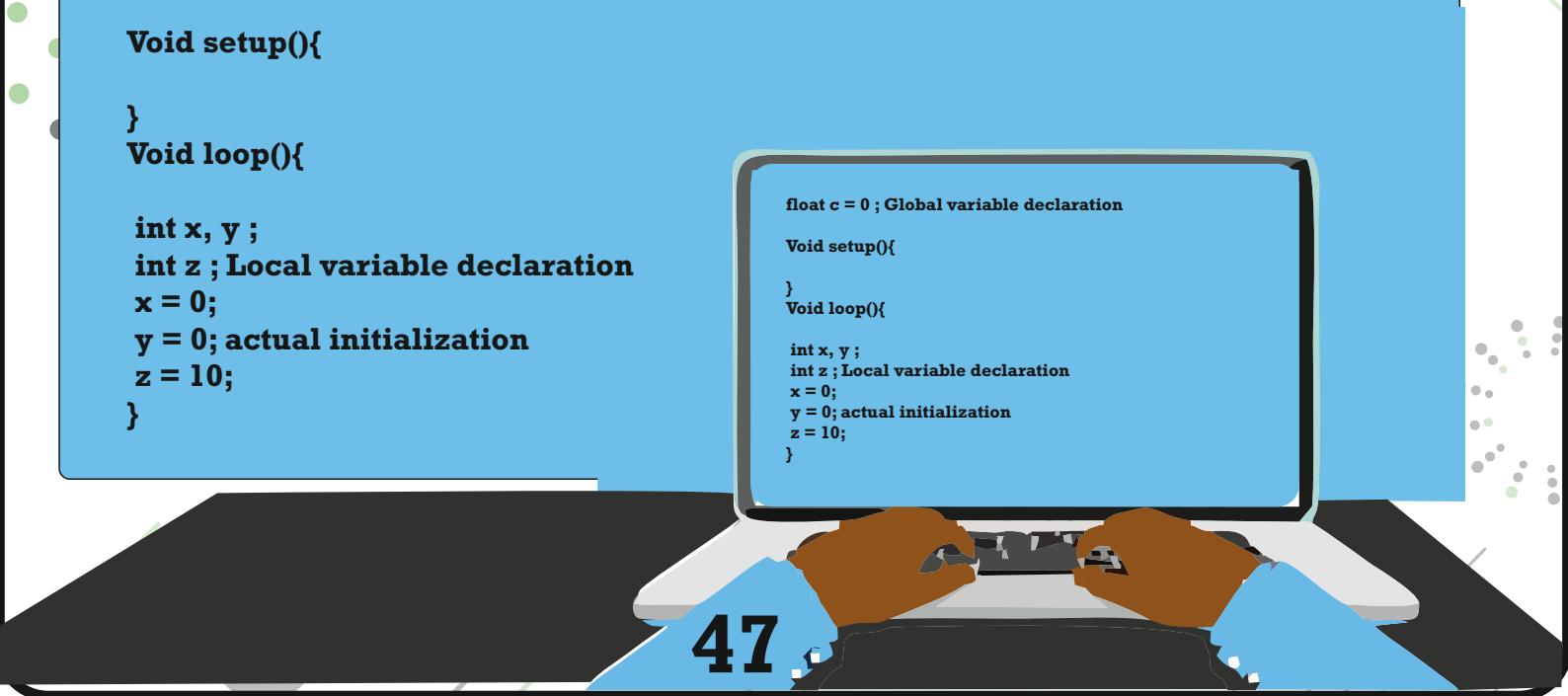
Local Variables
These are variables that are declared inside a function or block. They can be used only by the statement that are inside that function or block of code. Local variable are known to function outside their own. Following is the example using local variables -

```
Void setup(){
}
Void loop(){
 int x, y ;
 int z ; Local variable declaration
 x = 0;
y = 0; actual initialization
z = 10;
}
```

Global varibles
These variabls are defined outside of all the functions, usually at the top of the program. The global variable will hold their value throughout life-time of your program. A global variable can be accessed by any information. That is, a global variable is available for use throughout your entire program after its declaration.

The following examples uses global and local variables-

```
int T , S ;

float c = 0 ; Global variable declaration

Void setup(){

}
Void loop(){

 int x, y ;
 int z ; Local variable declaration
 x = 0;
 y = 0; actual initialization
 z = 10;
}
```



Comparison Operators

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

Example

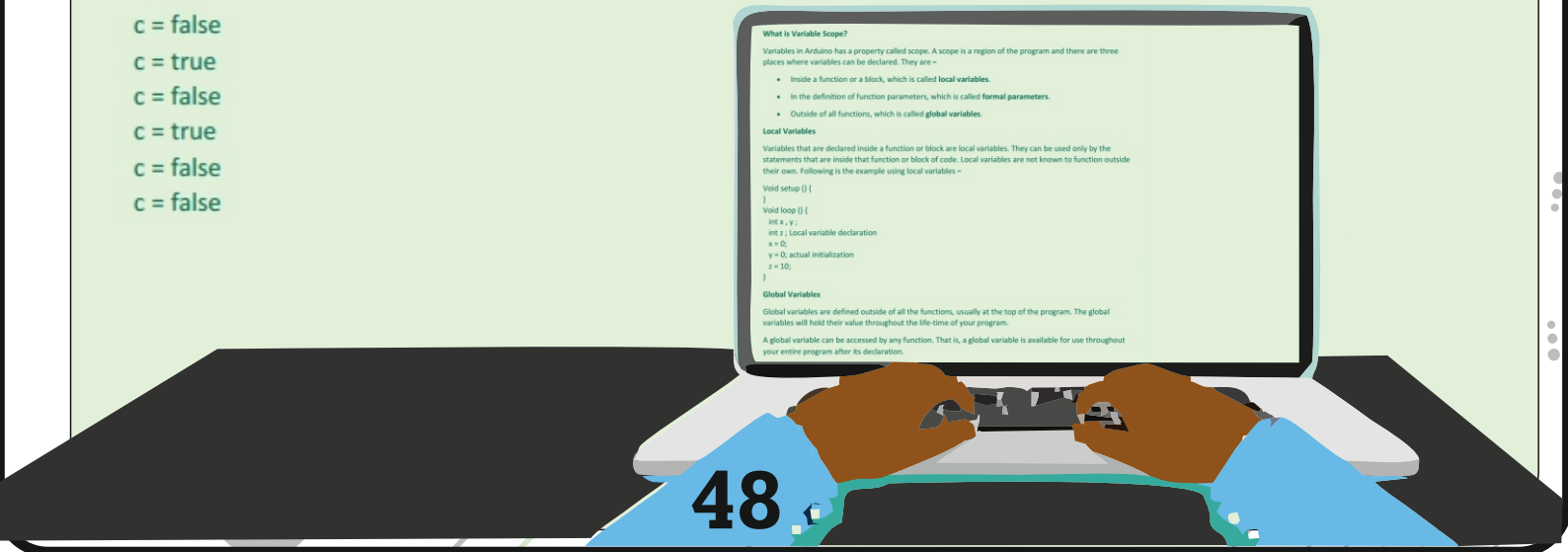
```
void loop () {
  int a = 9,b = 4
  bool c = false;
  if(a == b)
    c = true;
  else
    c = false;

  if(a != b)
    c = true;
  else
    c = false;

  if(a < b)
    c = true;
  else
    c = false;

  if(a > b)
    c = true;
  else
    c = false;

  if(a <= b)
    c = true;
  else
    c = false;

  if(a >= b)
    c = true;
  else
    c = false;
}
```

When input the comparison operator take note the of curly and normal brackets as they affect you code.

{ } ( )

Result

```
c = false
c = true
c = false
c = true
c = false
c = false
```

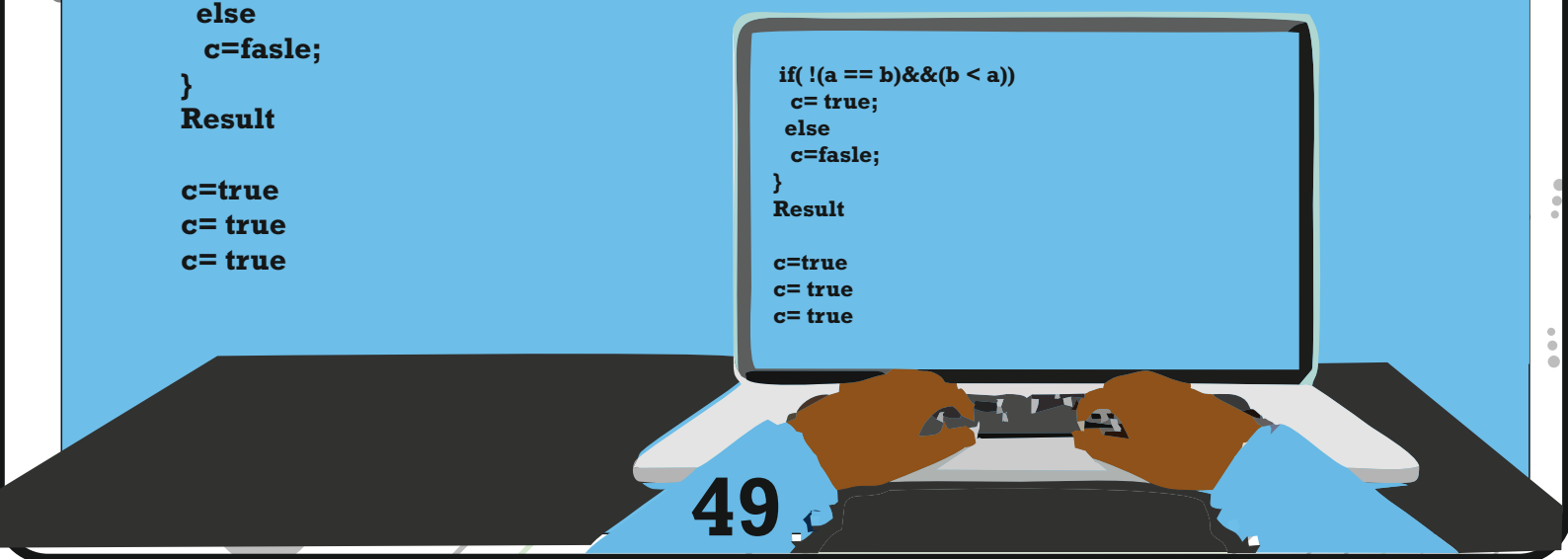


Boolean Operators

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

Operator name	Operator symbol	Description	Example
and	&&	Called Logical AND operator. If both the operands are non-zero then then condition becomes true	(A-&& B) is true
or	ll	Called Logical OR operator .If any of the two operands is non-zero then then condition becomes true.	(A ll B) is true
not	!	Called Logical NOT Operator. Use to reverses the logical state of operand. If a condition is true then Logical NOT operator will make false.	!(A ll B) is false

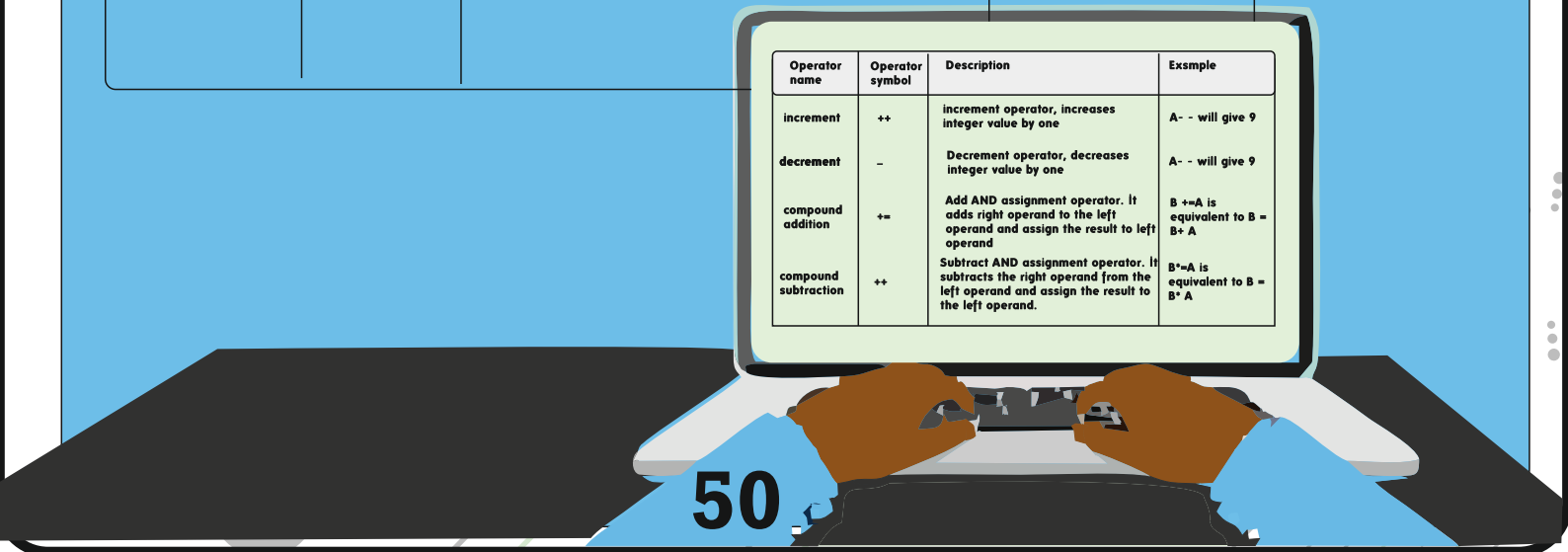
```
Example
void loop(){
  int a=9,b=4
  bool c=false,
  if((a > b)&& (b > a))

    c= true;
  else
    c= false;

if((a== b)l l (b < a))
   c= true;
  else
   c= false;

 if( !(a == b)&&(b < a))
   c= true;
  else
   c=fasle;
}
Result

c=true
c= true
c= true
```



Compound Operators

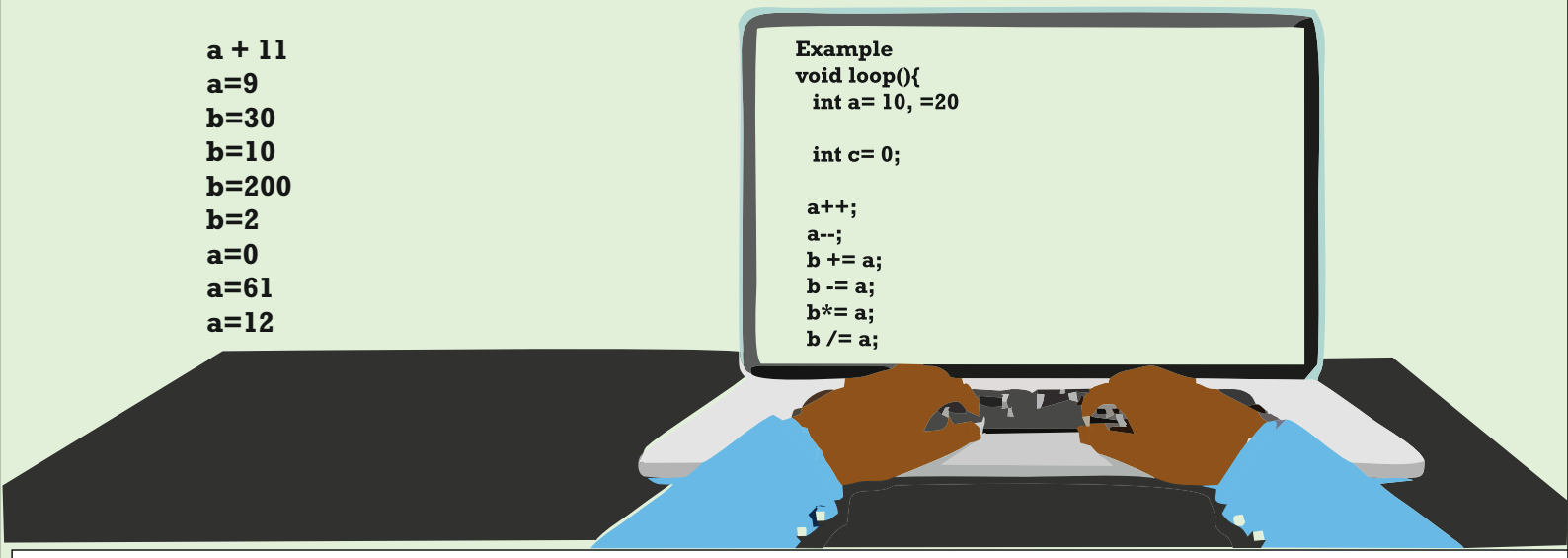
Assume variable A holds 10 and variable B holds 20 then-

Operator name	Operator symbol	Description	Example
increment	++	increment operator, increases integer value by one	A- - will give 9
decrement	–	Decrement operator, decreases integer value by one	A- - will give 9
compound addition	+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand	B +=A is equivalent to B = B+ A
compound subtraction	++	Subtract AND assignment operator. It subtracts the right operand from the left operand and assign the result to the left operand.	B*=A is equivalent to B = B* A
compound multiplication	*=	Multiply AND assignment operator. It multiples right operand with the left operand and assign the result to the left operand.	B*=A is equivalent to B = B* A
compound division	/=	Divide AND assignment operator. it divides left operand with the right operand and assign the result to the left operand	B/=A is equivalent to B = B/ A
compound modulo	%=	Modulus AND assignment operator. it takes modulus using two operands and assign the result to the left operand	B %=A is equivalent to B = B %A
compound bitwise or	I=	bitwise inclusive OR and assignment operator	A I=2 is same as A = A I 2
compound bitwise and	&=	Bitwise AND assignment operator	A &=2 is same as A = A & 2

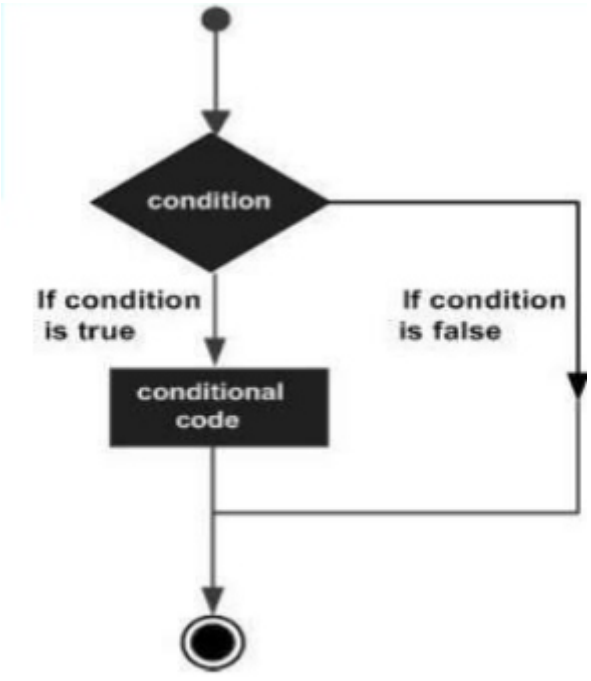


```
Example
void loop(){
  int a= 10, =20

  int c= 0;

 a++;
 a--;
 b += a;
 b -= a;
 b*= a;
 b /= a;
 a %= b;
 a l= b;
 a &= b;
}
Result

a + 11
a=9
b=30
b=10
b=200
b=2
a=0
a=61
a=12
```

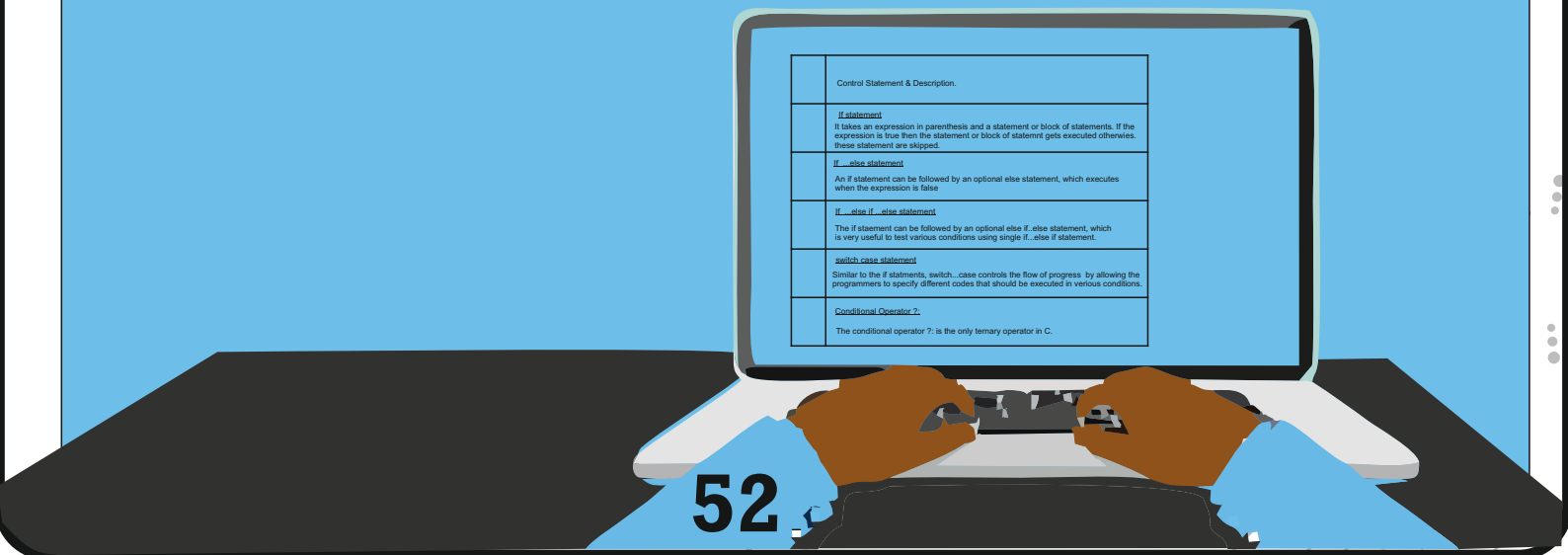
Booleans



Next, you will be learning more about the control statements which are 'if-else' and switch case statements. As shown in the picture here, if the condition is true then the conditional code is executed, else it is not executed.

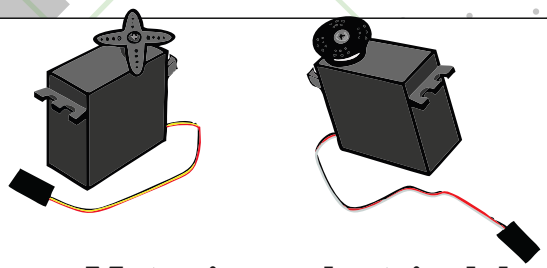


S/NO.	Control Statement & Description.
1.	<u>If statement</u> It takes an expression in parenthesis and a statement or block of statements. If the expression is true then the statement or block of statemnt gets executed otherwies. these statement are skipped.
2.	<u>If ...else statement</u> An if statement can be followed by an optional else statement, which executes when the expression is false
3.	<u>If ...else if ...else statement</u> The if staement can be followed by an optional else if..else statement, which is very useful to test various conditions using single if...else if statement.
4.	<u>switch case statement</u> Similar to the if statments, switch...case controls the flow of progress by allowing the programmers to specify different codes that should be executed in verious conditions.
5.	<u>Conditional Operator ?:</u> The conditional operator ?: is the only ternary operator in C.

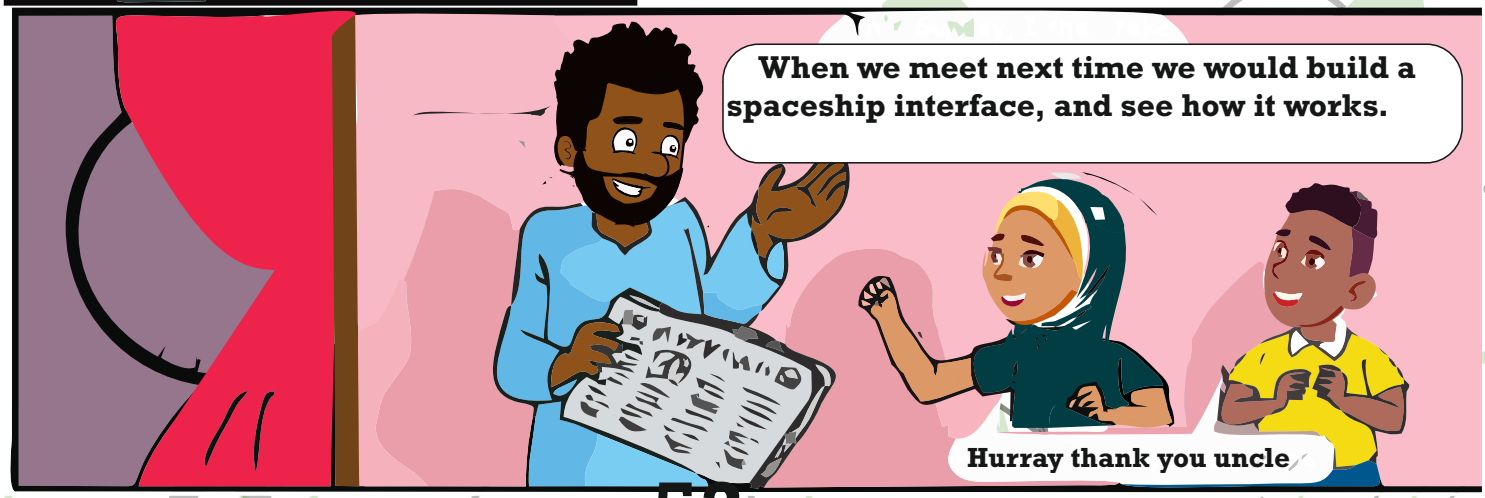


#TAG

RESISTANCE (R)
IS A MATERIAL'S OPPOSITION TO THE FLOW OF ELECTRIC CURRENT. IT IS MEASURED IN OHMS.

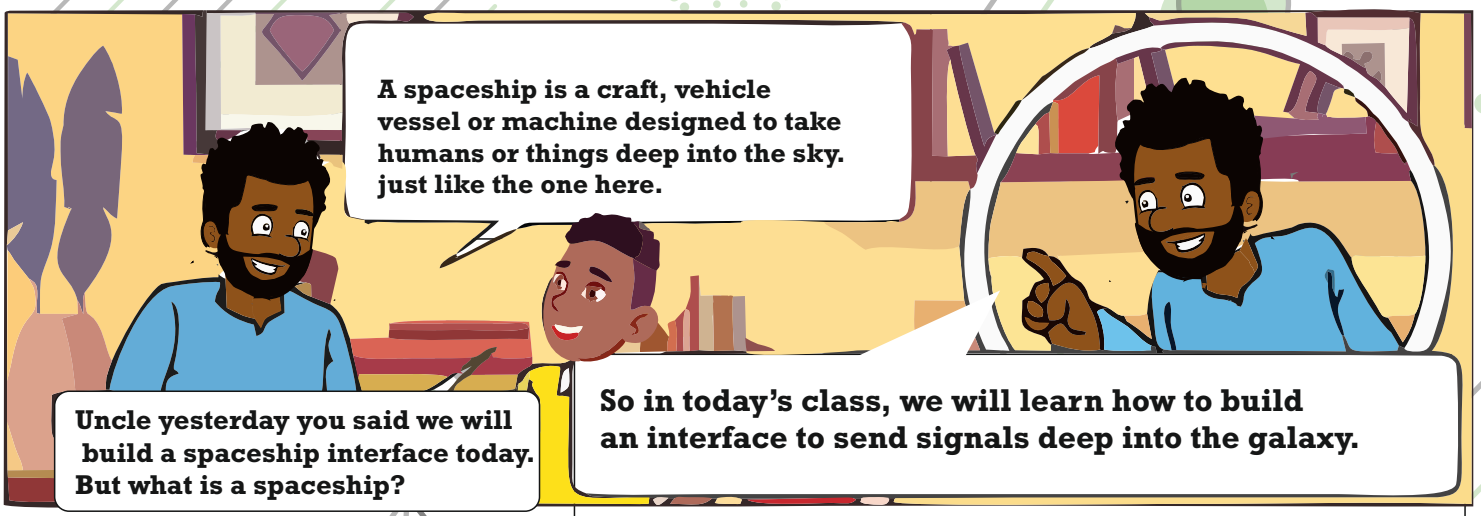


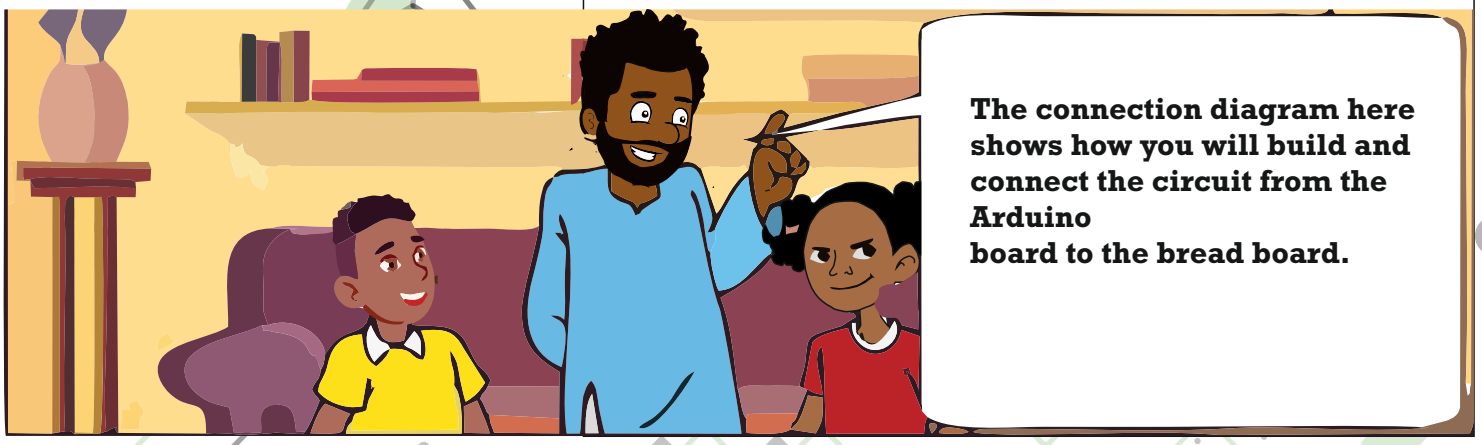
Servo Motor is an electrical device that can push or rotate an object with great precision.



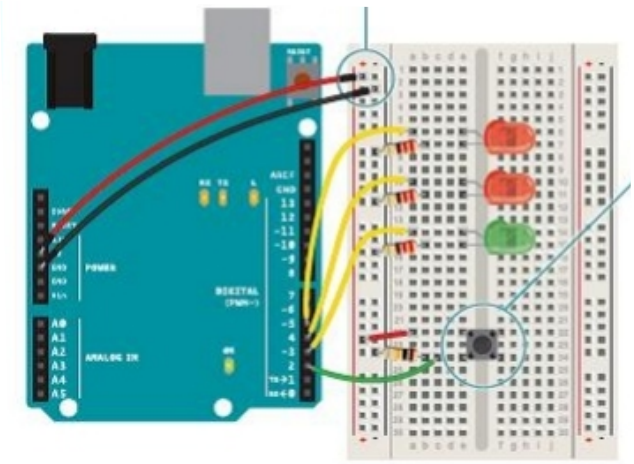
Episode 4
Spaceship interface

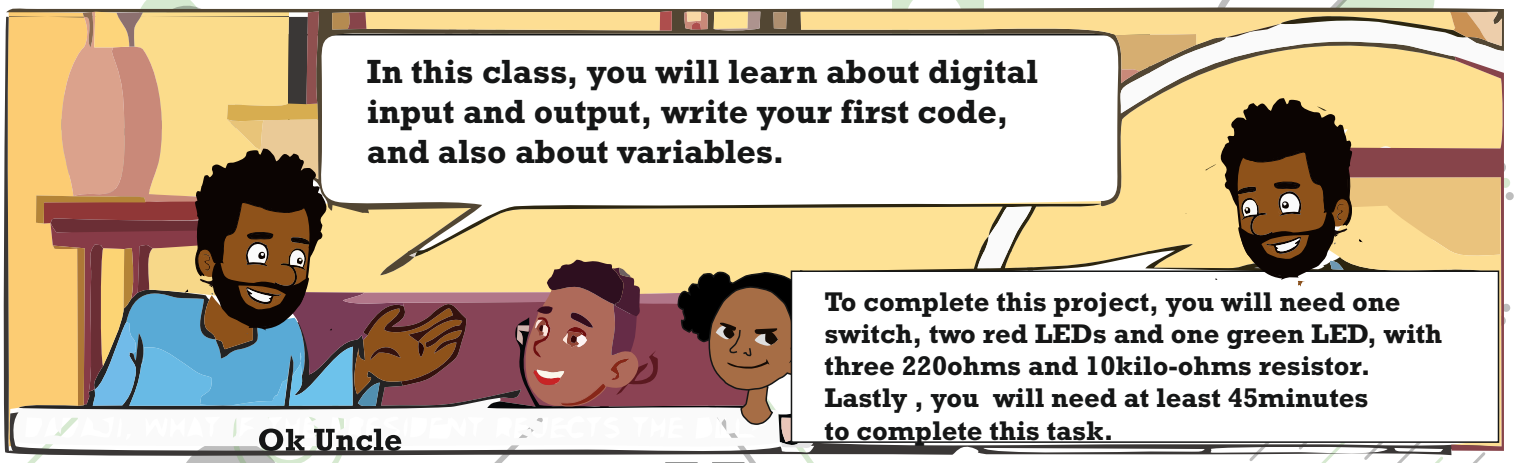
Spaceship Interface





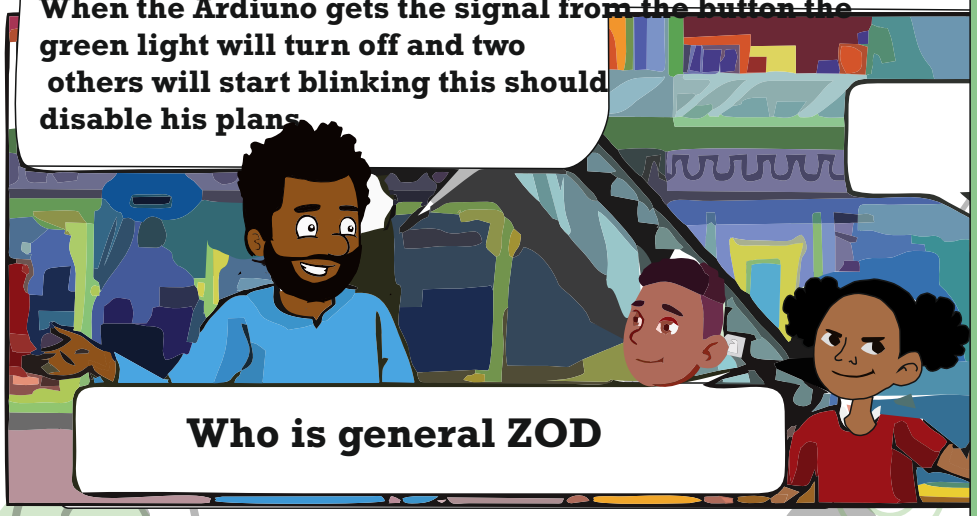
Arduino Uno is an open source micro-controller board based on the microchip ATmega328P micro-controller and developed by Arduino.cc.
It has digital and analog input & output pins that may be linked with various other circuits.






GreenLab Microfactory
This is going to be our virtual lab where major project will be carried out
Arduino Toolkits Here
This place looks like Dexter lab in Cartoon network
Uncle please show us round
Some of our activities will involve us stopping evil villians with Arduino
We will make use of a candance
what is a candence
It is a watch like device that allows change into your superhero outfits
What is a candence
Oh

kids you will learn in this project how to to control things with your Arduino. You will be making a control panel with a universal switch and light that would be used to stop General ZOD With this interface a green LED will be on until you press a button. When the Ardiuno gets the signal from the button the green light will turn off and two others will start blinking this should disable his plans.

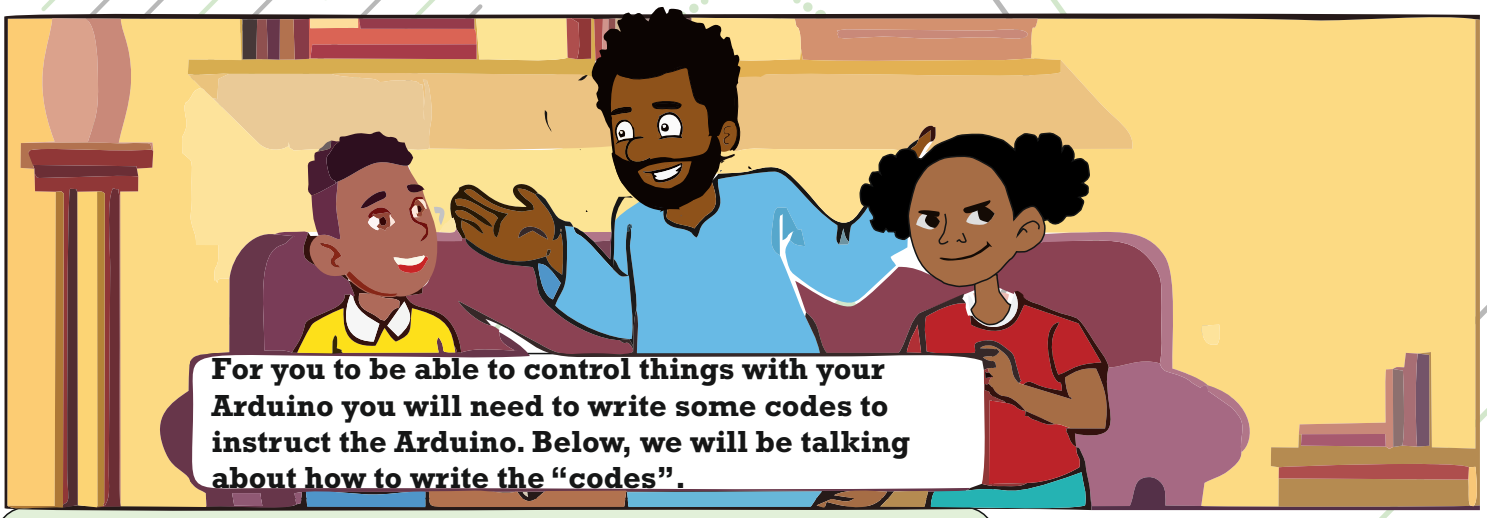


General ZOD is evil inventor that creates mind freak robots. His latest invention I s set to hit lagos. we have to stop his plan.

The Arduino digital pins can read only two states: when there is voltage on an input pin and when there's not. This kind of input is normally called digital(or sometimes binary, for two-stats). These states are commonly referred to as HIGH and LOW. HIGH is the same as saying "there is voltage here!" and LOW means "there is no voltage on this pin!". When you turn an OUTPUT pin HIGH using a command called digitalWrite(). you are turning it on. Measure the voltage between the pin and the ground, you will get 5 volts. When you turn an OUTPUT pin LOW, you are turning it off. The Arduino's digital pins can act as both inputs and outputs. In your code, you will configure them depending on what you want their function to be. When the pins are output, you can turn on components like LEDS, if you configure the pins as inputs, you can check if a switch is being pressed or not. Since pins 0 and 1 are used for communicating with the computer, it is best to start with pin2.

To proceed, follow the connection diagram above, wire up your breadboard to the Arduino's 5V and ground (GND) connections, just like the previous project. Place the two red LEDs and one green LED on the breadboard. Attach the cathode (short leg) of each LED to ground through a 220-ohm resistor. Connect the annode (long leg) of the green LED to pin 3. Connect the red LEDs' anodes to pin 4 and 5, respectively.

Place the switch on the breadboard just as you did in the previous project. Attach one side to power, and the other side to digital pin 2 on the Arduino. You will also need to add a 10k-ohm resistor from ground to the switch pin that connect to the Arduino. That pull down resistor connects the pin to ground when the switch is open, so it reads LOW when there is no voltage coming in through the switch.



Understanding the code

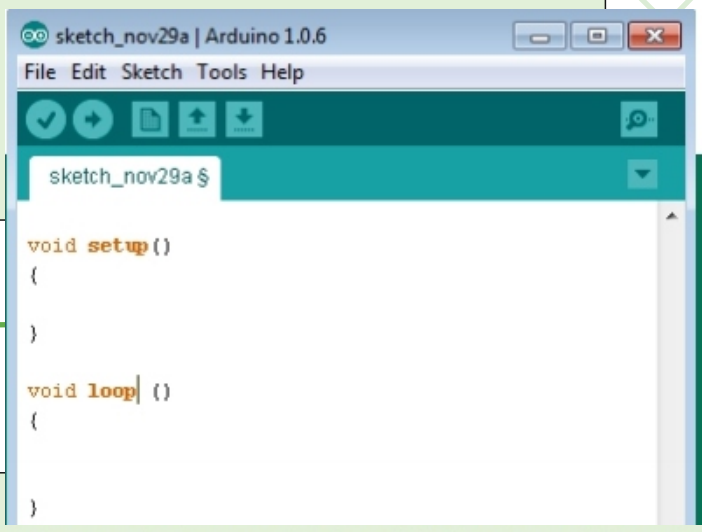
The source code in Arduino is called "Sketch"

Arduino programs can be divided in three main part: Structure, Values (variables and constants) and Functions. In tutorial, we will learn about the Arduino software program, step by step and how we can write the program without any syntax or compilation error.

Lets start with the Structure. Software structure consist of two main functions:-

-Setup( ) function

-Loop( ) function



{ Curly brackets }

Any code you write inside the curly brackets will be executed when the function is called.

The setup() function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The setup function will only run once, after each power up or reset of the Arduino board. This is where you configure the digital pins to be either inputs or outputs using a function named pinMode(). The pins connected to LEDs will be OUTPUTs and the switch pin will be an INPUT.

The loop() function does precisely what its name suggests, and loops continuously, allowing your program to change and respond. Use it to actively control the Arduino board. The loop() is where you'll check for voltage on the inputs, and turn outputs on and off. To check the voltage level on a digital input, you use the function digitalRead() that checks the chosen pin for voltage. To know what pin to check, digitalRead() expects an argument.

Arguments are information that you pass to functions, telling them how they should do their job. For example, digitalRead() needs one argument: what pin to check. In your program, digitalRead() is going to check the state of pin 2 and store the value in the switchState variable.

If there's voltage on the pin when digitalRead() is called, the switchState variable will get the value HIGH (or 1). If there is no voltage on the pin, switchState will get the value LOW (or 0).

Spaceship Interface


```
int switchState = 0; // this is a comment. Comments are used to explain a code
void setup() {
 // put your setup code here, to run once:
 pinMode(3,OUTPUT);
 pinMode(4,OUTPUT);
 pinMode(5,OUTPUT);
 pinMode(2,INPUT);
}
void loop() {
 // put your main code here, to run repeatedly:
 switchState = digitalRead(2);
 if (switchState == LOW) { // the button is not pressed
  digitalWrite(3, HIGH); // green light
  digitalWrite(4, LOW); // red light
  digitalWrite(5, LOW); // red light
 }
 else { // the button is pressed
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  delay(250); // waits for a quarter second
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  delay(250);
  }
}
```

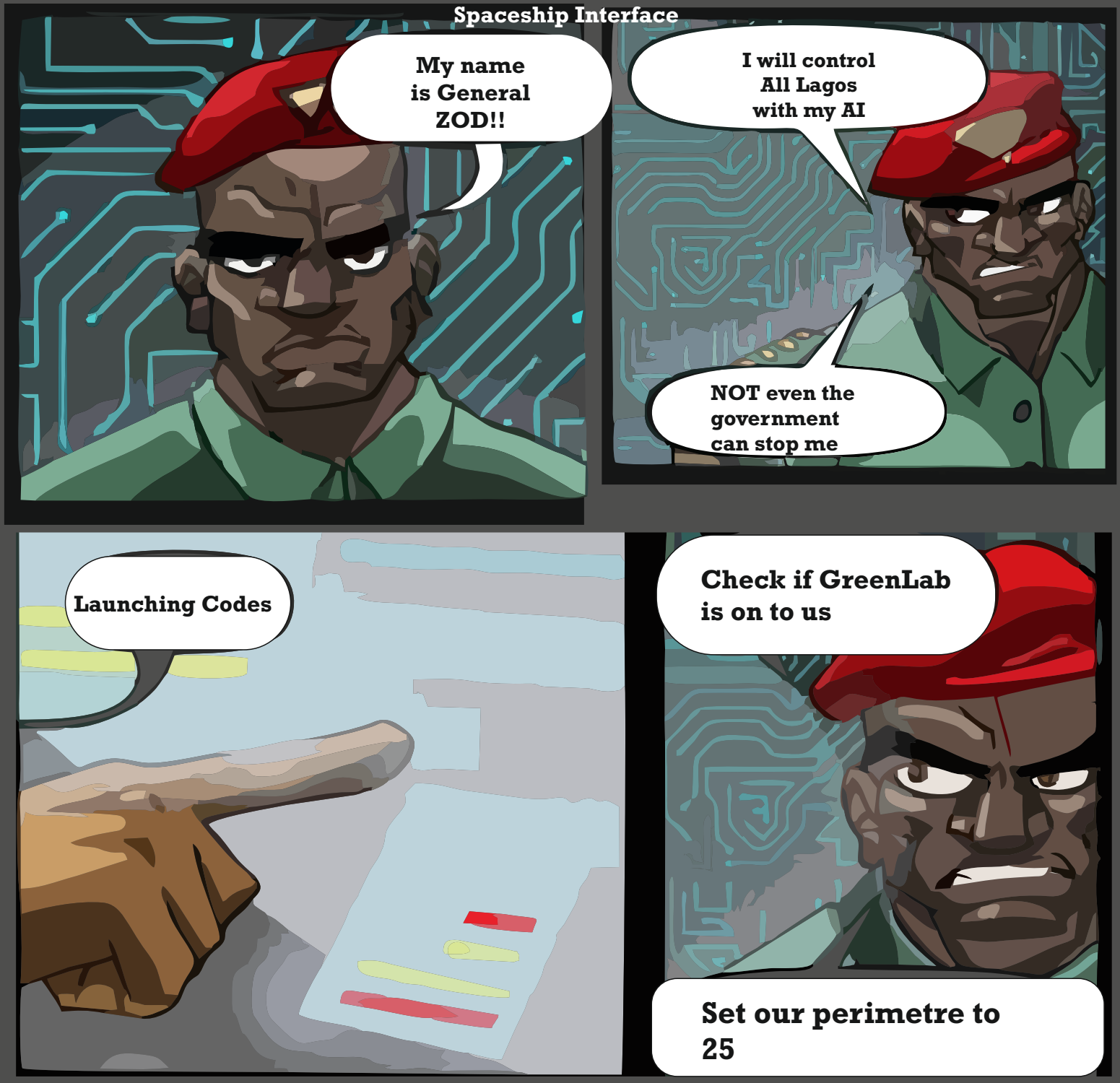
Case sensitivity

Pay attention to the case sensitivity in your code. For example, pinMode is the name of a command, but pinmode will produce an error.

Comments

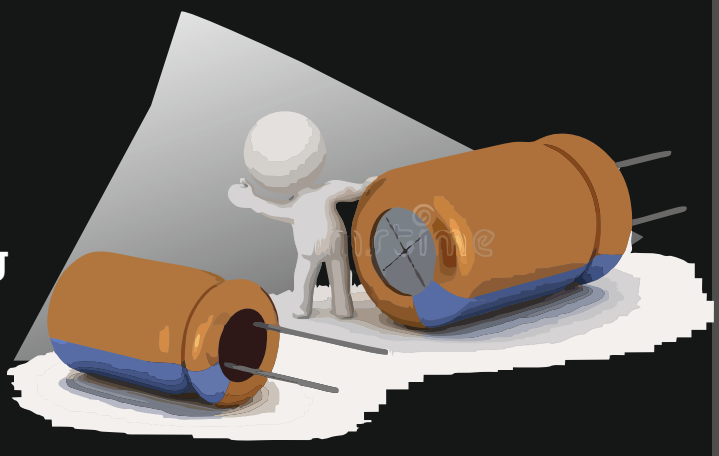
If you ever want to include natural language in your program, you can leave a comment. Comments are notes you leave for yourself that the computer ignores. To write a comment, add two slashes //

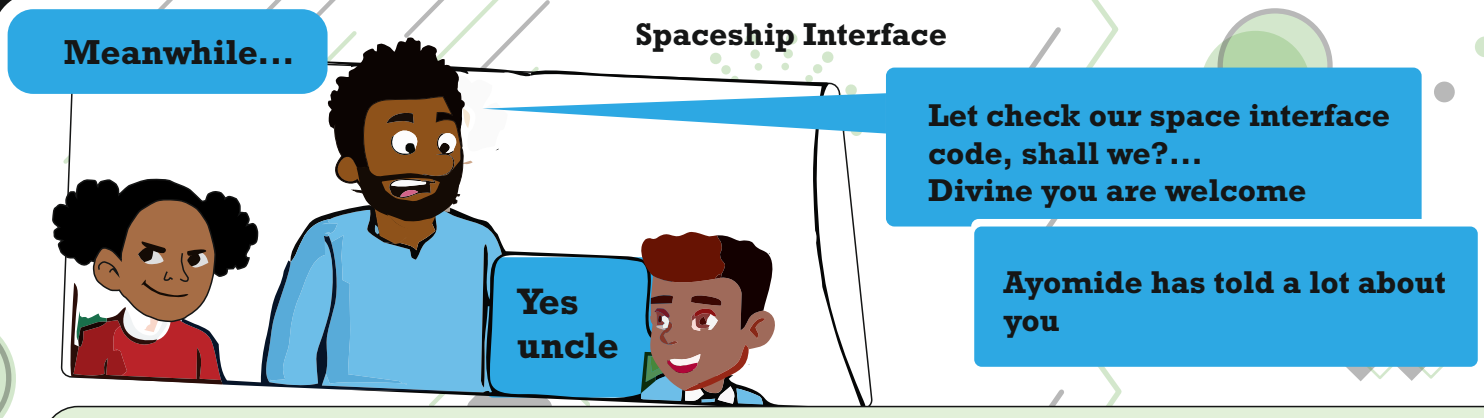
The computer will ignore anything on the line after those slashes.



#TIP

Capacitor is a device used to store electric charge, consisting of one or more pairs of conductors separated by an insulator





If Statement

In the code above, you used the word if to check the state of something (namely, the switch position). An if() statement in programming compares two things, and determines whether the comparison is true or false. Then it performs actions you tell it to do. When comparing two things in programming, you use two equal signs **==**. If you use only one sign, you will be setting a value instead of comparing it.

Spaceship Interface Code

digitalWrite() is the command that allows you to send 5V or 0V to an output pin. digitalWrite() takes two arguments: what pin to control, and what value to set that pin, HIGH or LOW. If you want to turn the red LEDs on and the green LED off inside your if() statement, your code would look like this .

You've told the Arduino what to do when the switch is open. Now define what happens when the switch is closed. The if() statement has an optional else component that allows for something to happen if the original condition is not met. In this case, since you checked to see if the switch was LOW, write code for the HIGH condition after the else statement.

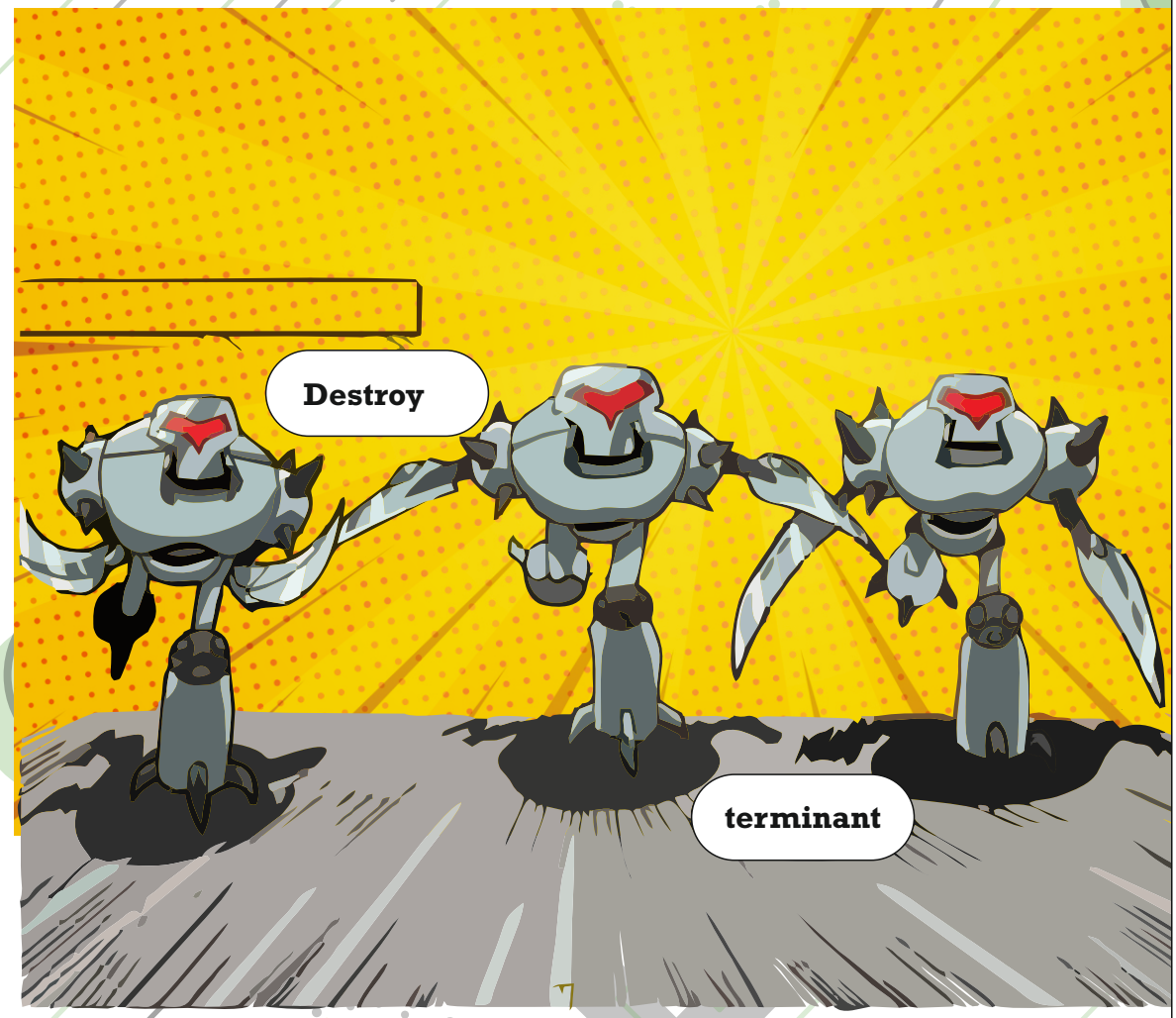
To get the red LEDs to blink when the button is pressed, you'll need to turn the lights off and on in the else statement you just wrote. To do this, change the code to look like this.

After setting the LEDs to a certain state, you'll want the Arduino to pause for a moment before changing them back. If you don't wait, the lights will go back and forth so fast that it will appear as if they are just a little dim, not on and off. This is because the Arduino goes through its loop() thousands of times each second, and the LED will be turned on and off quicker than we can perceive. The delay() function lets you stop the Arduino from executing anything for a period of time. delay() takes an argument that determines the number of milliseconds before it executes the next set of code. There are 1000 milliseconds in one second. delay(250) will pause for a quarter second.

Once your Arduino is programmed, you should see the green light turn on. When you press the switch, the red lights will start flashing, and the green light will turn off. Try changing the time of the two delay() functions; notice what happens to the lights and how the response of the system changes depending on the speed of the flashing. When you call a delay() in your program, it stops all other functionality. No sensor readings will happen until that time period has passed. While delays are often useful, when designing your own projects make sure they are not unnecessarily interfering with your interface.

its almost
time kids

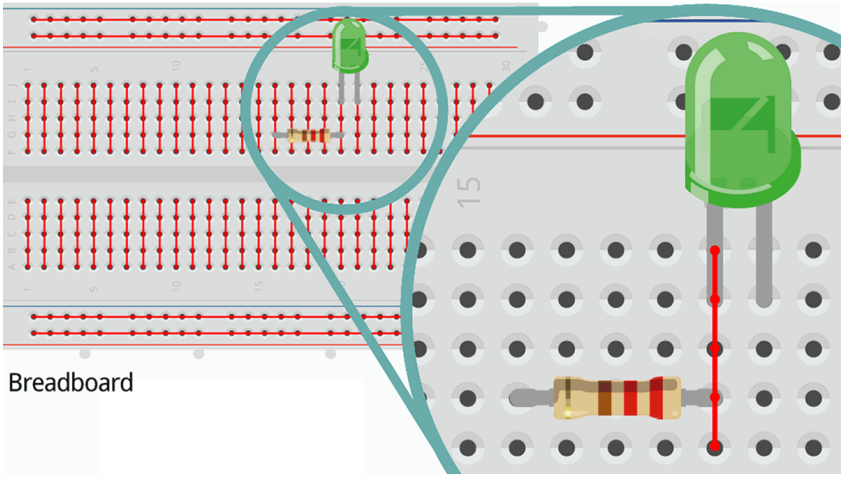
Kids change into your hero outfits


Destroy
terminant


Lets show
them who's
boss

Spaceship Interface

#TIP



How to use a Breadboard

You already used the breadboard but we totally forgot to talk about it. We can use a breadboard (protoboard) to connect electrical components. Inside the breadboard are metal clamps. Since metal conducts electricity, components in the same clamps are connected. In that illustration you can see which holes of a breadboard are connected internally.
Do you see that the resistor sticks with one leg in the same column like the green LED? That is the point where they are connected.
Expert Knowledge: Conductors and Insulators

Why does the metal clamps make a connection, but not the plastic of the breadboard? There are electrical conducting materials. They are called conductors. Metalls, for instance silver, copper or aluminium are very good conductors. They conduct, because they've got so called freely moving electrons. This is actually what current consists of.
Materials without the ability of conducting electricity are called insulators. Plastic, ceramic and glas belonging to the group of insulators. Maybe you've already seen the large insulators on high-tension power lines? They hold the cables to the high-voltage poles. Usually they are made of ceramic.

Your Rights to Innovate

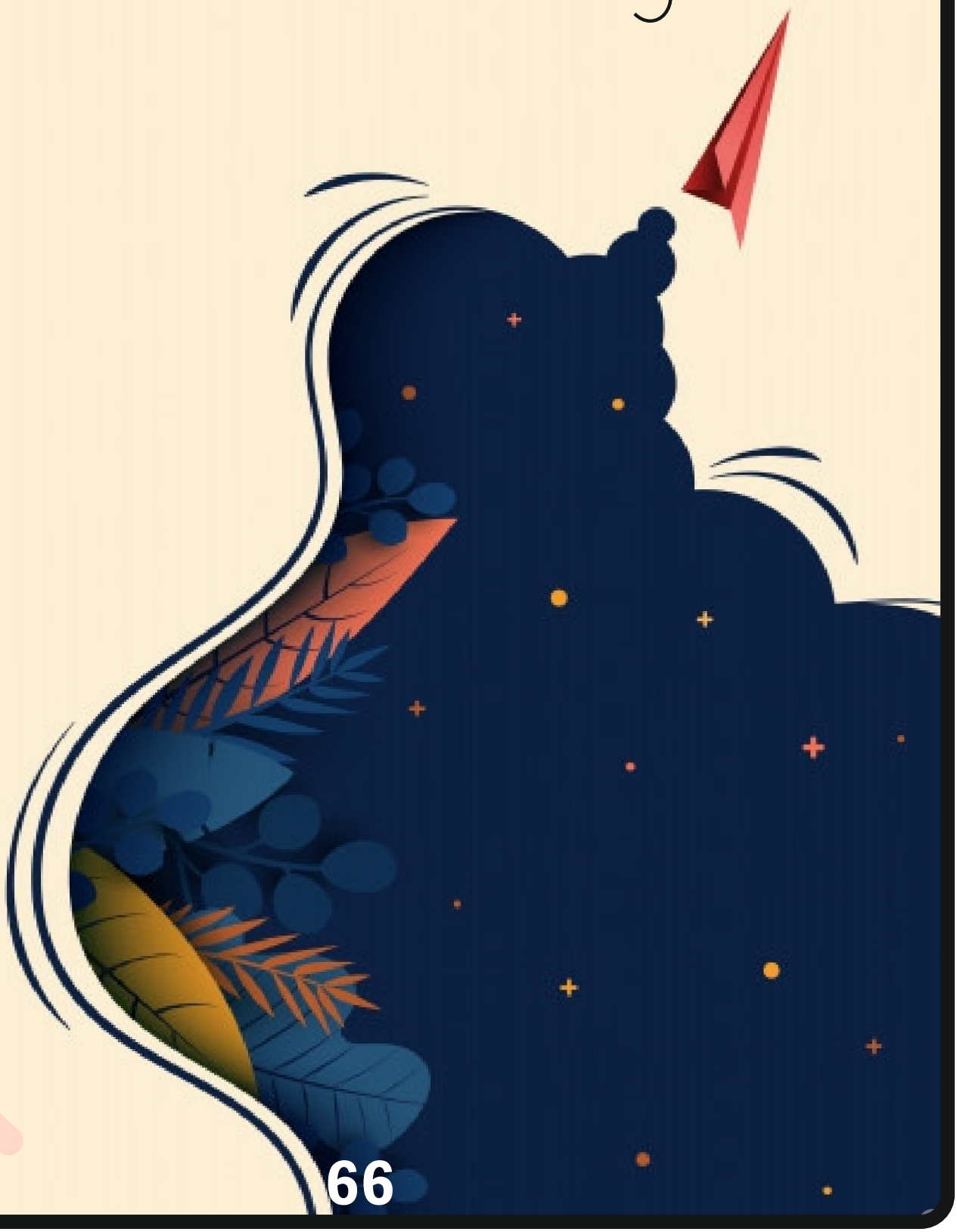
Episode 05- Welcome to YouNiversity

GreenLab Cartoons

GreenLab MICRO-FACTORY

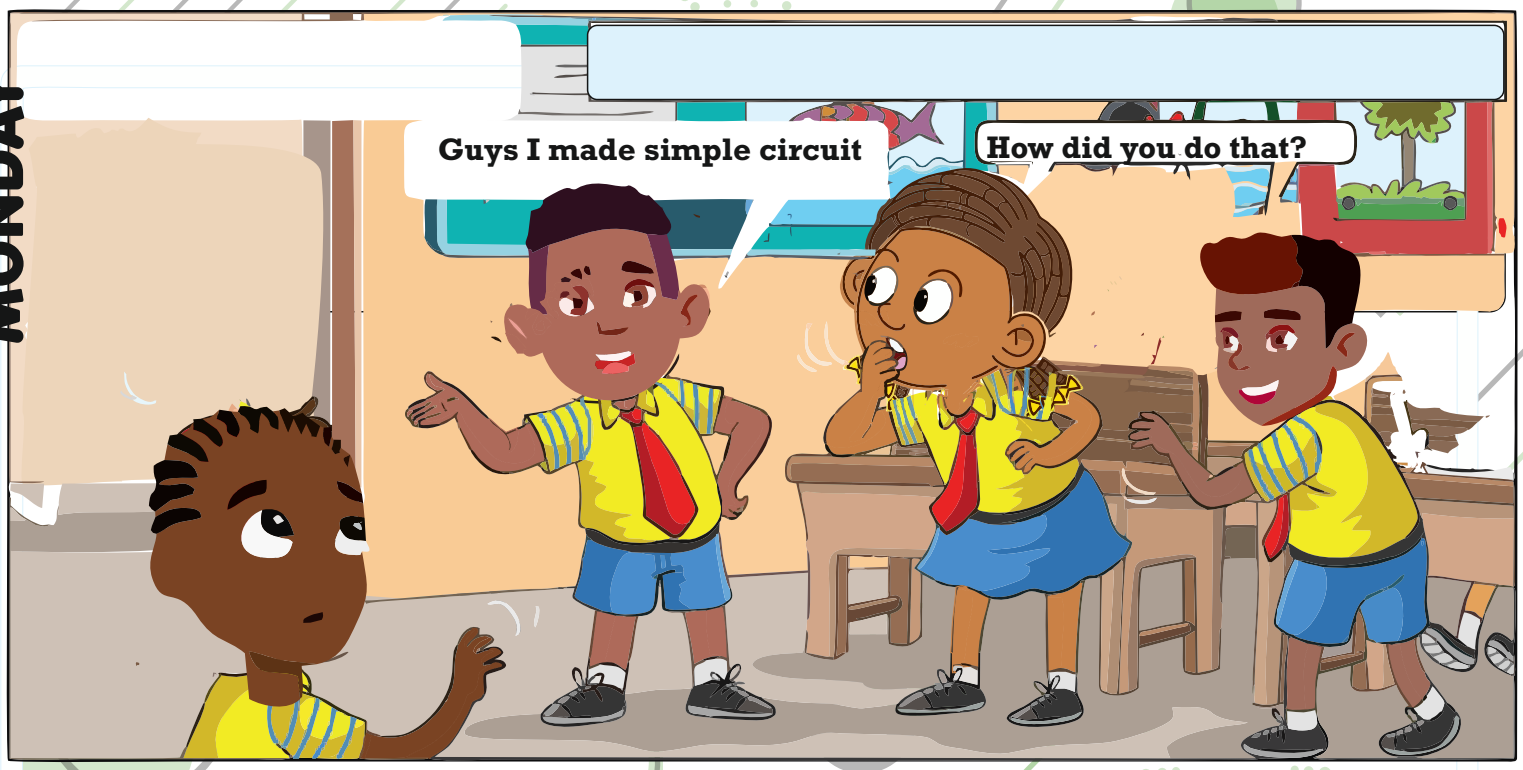
Your Rights to Innovate

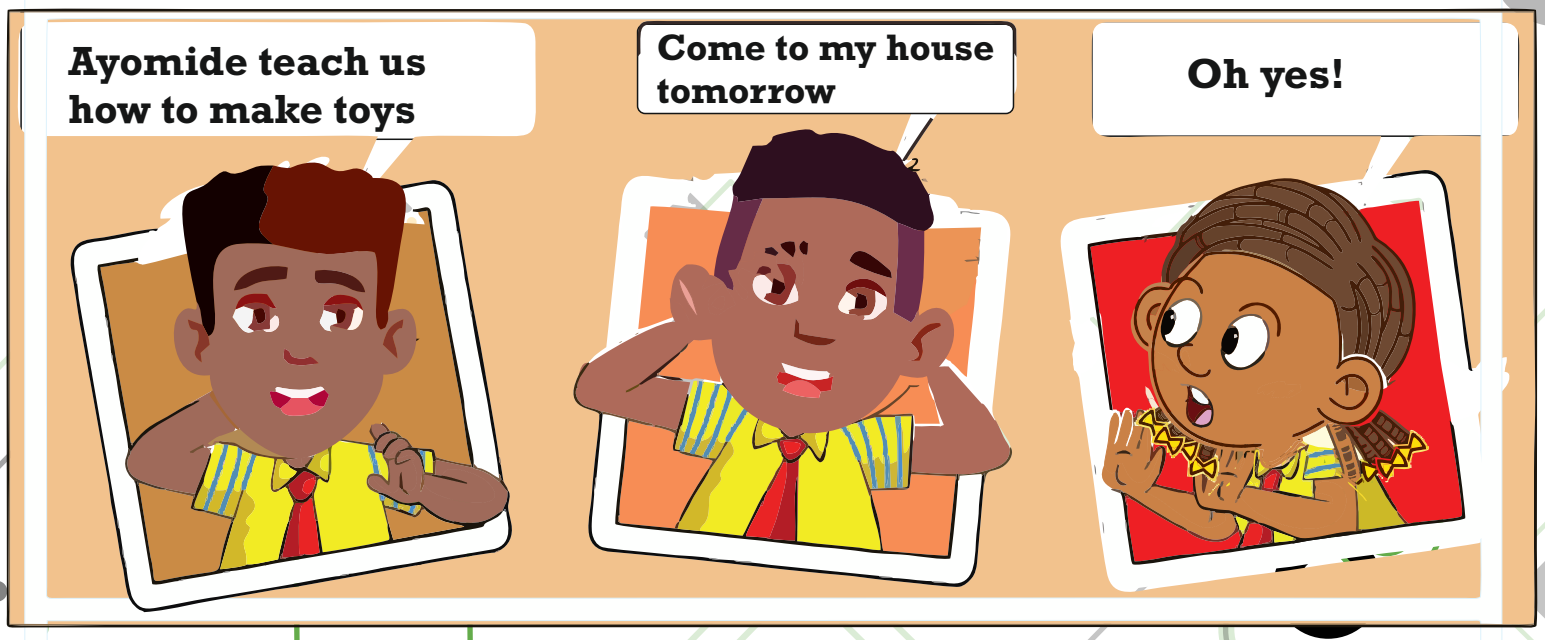
YouNiversity



Guys welcome to YouNiversity,
You have the following task
- Goto You tube
-Find an Arduino project
-Follow how it was built and build it
-Then present your project to your friends explaning to them why you selected the project ,what you have learnt, and how others can learn from the project.

MONDAY



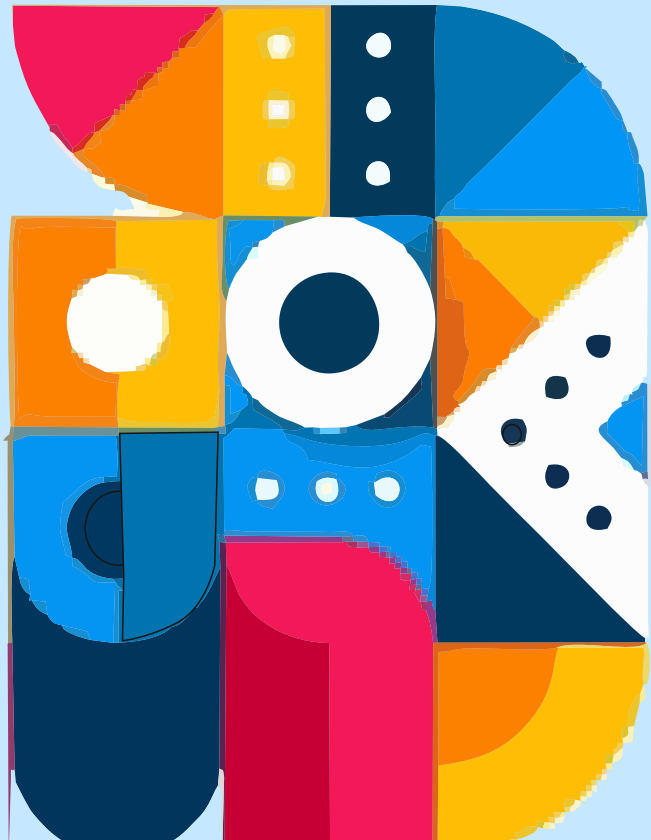


TIP

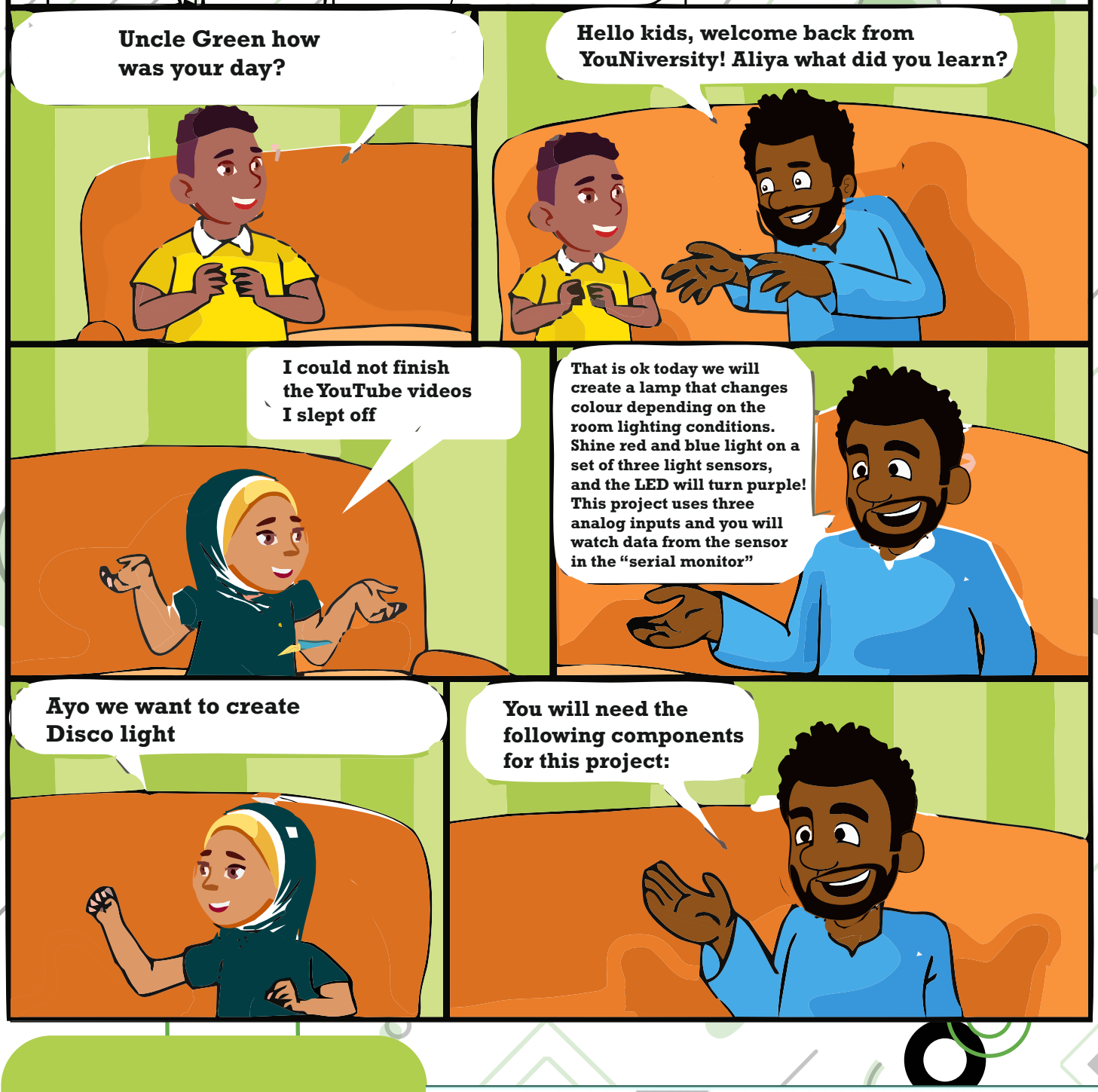
What is AI

The full meaning of AI is Artificial Intelligence

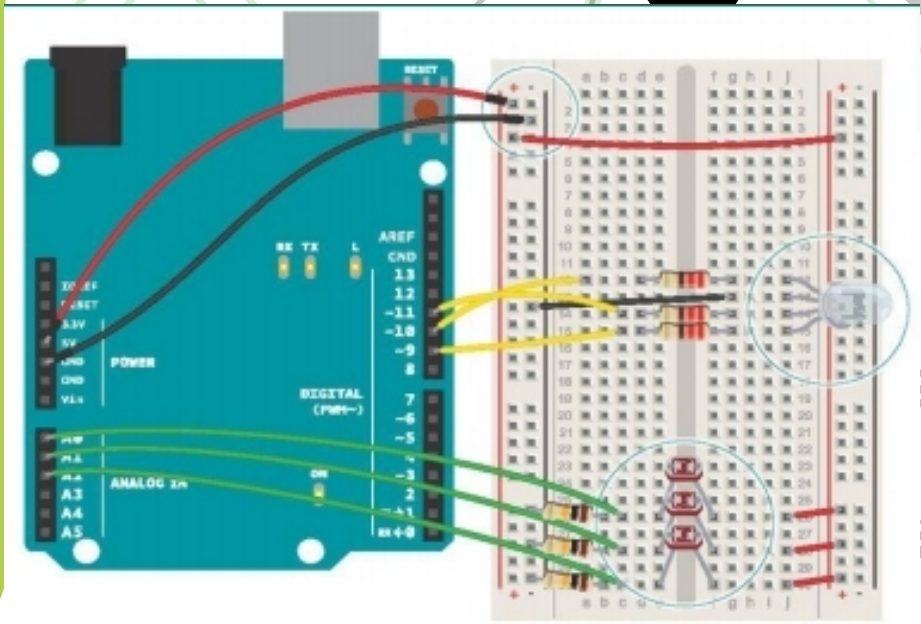
EPISODE 6 -
Colour mixing lamp



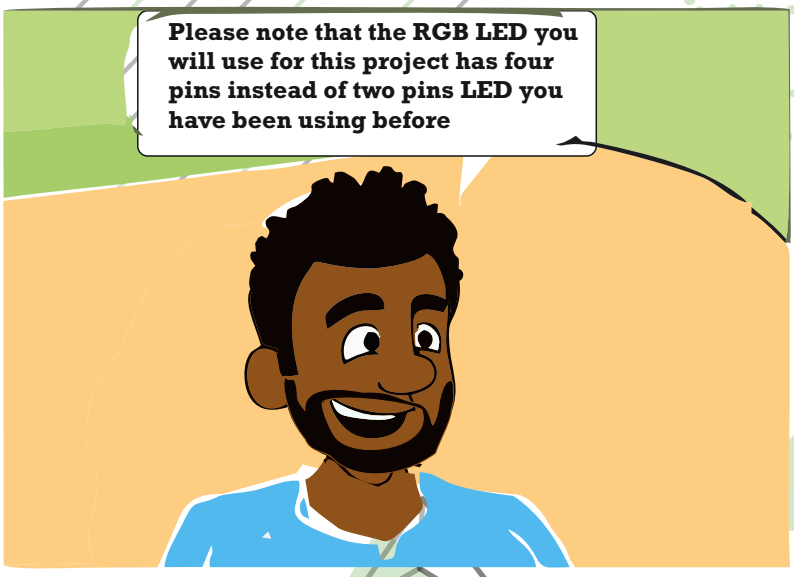
Colour mixing lamp

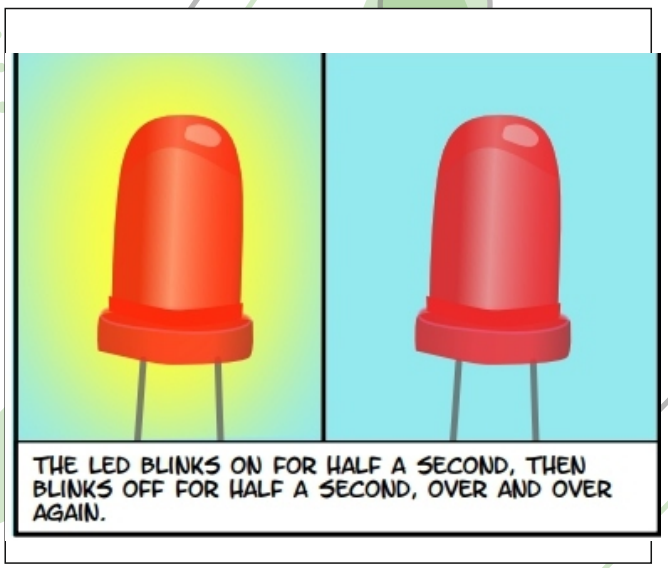


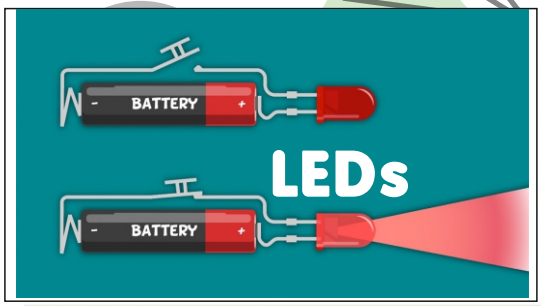
1.One RBG LED
2.three 220ohms resistors
3.three 10kilo-ohms resistors
4.three photoresistors
and at least 45minutes of work time. Using a tri-color LED and three photo-resistors, you will be creating a lamp that smoothly changes colors depending on external lighting conditions. So if you are ready, connect your components as shown in the image



Colour mixing lamp







In this project, the photoresistors (sensors that change their resistance depending on the amount of light that hits them, also known as photocells or light-dependent resistors) will be used as inputs. If you connect one end of the resistor to your Arduino, you can measure the change in resistance by checking the voltage on the pin.

How to connect

1. Place the three photoresistors on the breadboard so they cross the center divide from one side to the other, as shown in Fig. 1. Attach one end of each photoresistor to power. On the other side, attach a 10-kilohm resistor to ground. This resistor is in series with the photoresistor, and together they form a voltage divider. The voltage at the point where they meet is proportional to the ratio of their resistances, according to Ohm's Law (see Project 1 for more on Ohm's Law). As the resistance of the photoresistor changes when light hits it, the voltage at this junction changes as well. On the same side as the resistor, connect the photoresistors to Analog In pins 0, 1, and 2 with hookup wire.
2. Take the three colored gels and place one over each of the photoresistors. Place the red gel over the photoresistor connected to A0, the green over the one connected to A1, and the blue over the one connected to A2. Each of these filters lets only light of a specific wavelength through to the sensor it's covering. The red filter passes only red light, the green filter passes only green light, and the blue filter passes only blue light. This allows you to detect the relative colour levels in the light that hits your sensors.
3. The LED with 4 legs is a common cathode RGB LED. The LED has separate red, green, and blue elements inside, and one common ground (the cathode). By creating a voltage difference between the cathode and the voltage coming out of the Arduino's PWM pins (which are connected to the anodes through 220-ohm resistors), you'll cause the LED to fade between its three colours. Make note of what the longest pin is on the LED, place it in your breadboard, and connect that pin to ground. Connect the other three pins to digital pins 9, 10 and 11 in series with 220-ohm resistors. Be sure to connect each LED lead to the correct PWM pin, according to the figure on the left.

#TIP

LED(Light emitting diode) is a semi conductor device that emits light when an electric current is passed through it. Light is produced when the particles that carry the current known as electrons and holes combine together within the semiconductor material.

Colour mixing lamp

The Code

```
const int greenLEDPin=9;
const int blueLEDPin=10;
const int redLEDPin=11;

const int redSensorPin=A0;
const int greenSensorPin=A1;
const int blueSensorPin=A2;

int redValue=0;
int greenValue=0;
int blueValue=0;

int redSensorValue=0;
int greenSensorValue=0;
int blueSensorValue=0;

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

pinMode(greenLEDPin,OUTPUT);
pinMode(blueLEDPin,OUTPUT);
pinMode(redLEDPin,OUTPUT);
}

void loop() {
redSensorValue=analogRead(redSensorPin);
delay(5);
greenSensorValue=analogRead(greenSensorPin);
delay(5);
blueSensorValue=analogRead(blueSensorPin);

Serial.print("Raw Sensor Value \t red: ");
Serial.print(redSensorValue);
Serial.print(" \t green: ");
Serial.print(greenSensorValue);
Serial.print(" \t blue: ");
Serial.println(blueSensorValue);
```

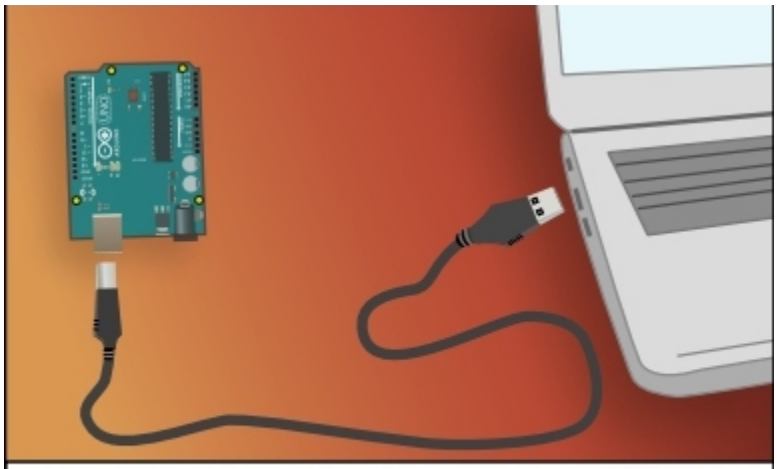
The Code - *continued*

```
//report the calculated LED light levels
redValue=redSensorValue/4;
greenValue=greenSensorValue/4;
blueValue=blueSensorValue/4;

Serial.print("Maped Sensor Value \t red: ");
Serial.print(redValue);
Serial.print(" \t green: ");
Serial.print(greenValue);
Serial.print(" \t blue: ");
Serial.println(blueValue);

analogWrite(redLEDPin,redValue);
analogWrite(greenLEDPin,greenValue);
analogWrite(blueLEDPin,blueValue);

}
```

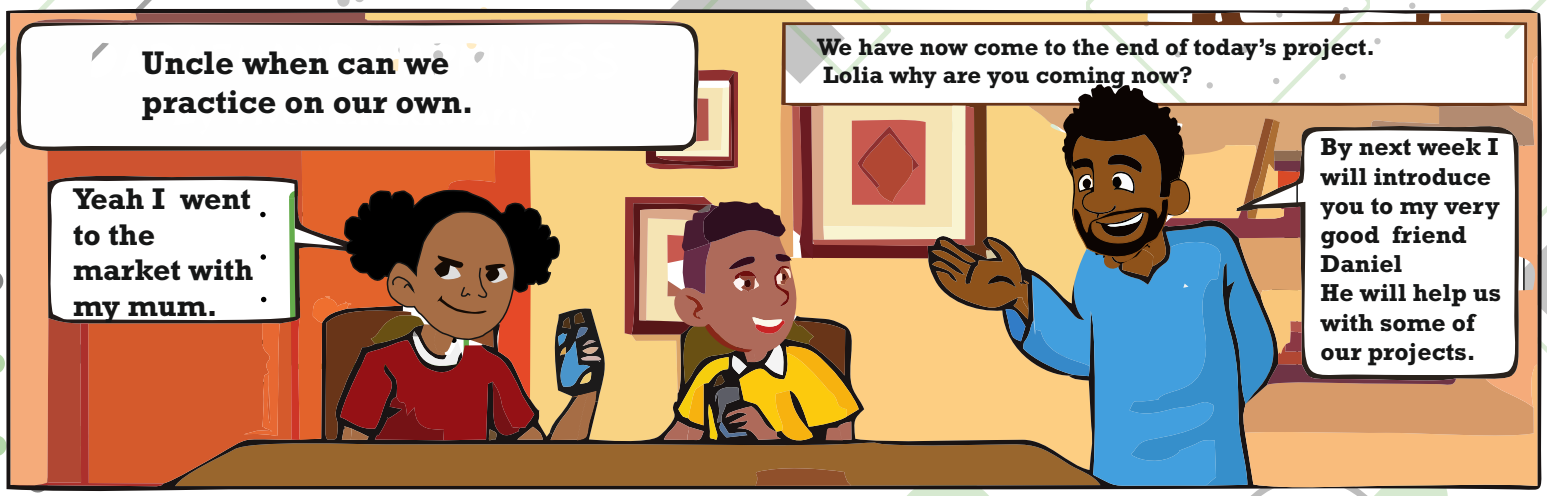
Use it

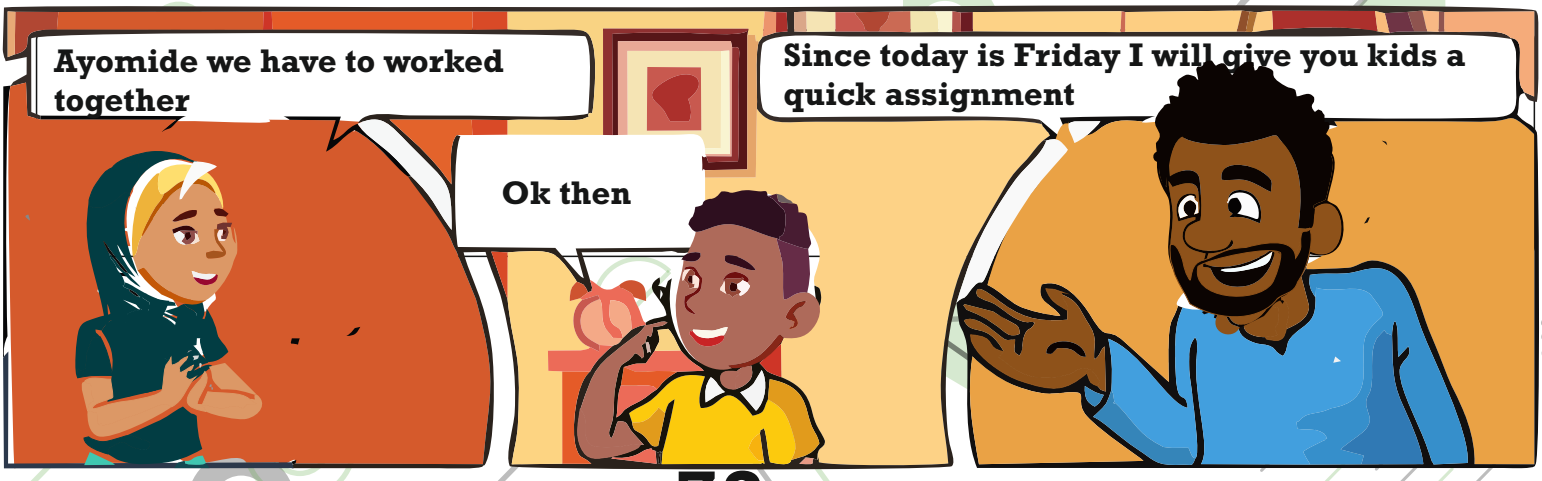
Once you have your Arduino programmed and wired up, open the serial monitor. The LED will probably be an off-white colour, depending on the predominant colour of the light in your room. Look at the values coming from the sensors in the serial monitor, if you're in an environment with stable lighting, the number should probably be fairly consistent.

Turn off the light in the room you're in and see what happens to the values of the sensors. With a flashlight, illuminate each of the sensors individually and notice how the values change in the serial monitor, and notice how the LED's colour changes. When the photoresistors are covered with a gel, they only re-act to light of a certain wavelength. This will give you the opportunity to change each of the colours independently.



Connecting your Arduino Board to your IDE





A rear picture of kids doing their project

UP NEXT

Cartooino Book 2

Cartooino Project Book 2020

GreenLab Cartooino