ISSUE:
GreenLab
Microfactory

Cartooino

COMICS

GreenLab

Hands-on

ONE STUDENT → ONE ARDUINO

Interactive

123 456 789 012

WIN

EXCITING PRIZES

VISIT OUR WEBSITE FOR

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.

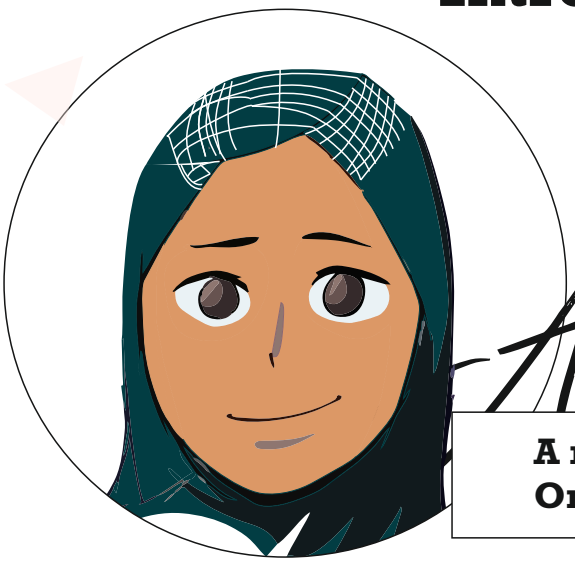
.

Disclaimer

This book was not created for commercial purposes, and the contents of the cartooino projects Book are licensed under a Creative Commons Attribution -ShareAlike (CC BY-SA) by Greenlab Microfactory. This means that you can copy, reuse, adapt and build upon the text of this book non-commercially while attributing the original work(but not in any way that suggests that we endorse you or your use of the work) and only if the result are transmitted under the same Creative Common license.

Full license terms: creativecommons. org/license/bu-nc-sa/3. 0/

Introduction

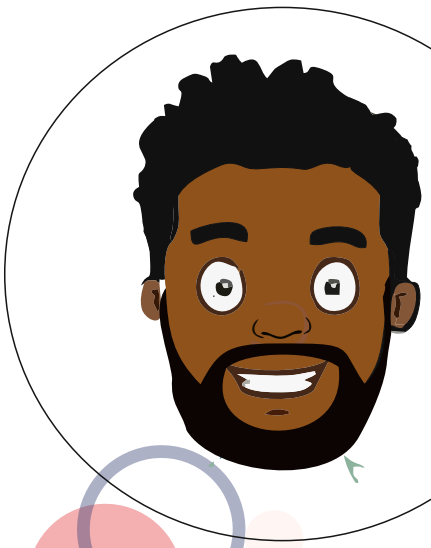


Aliya

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 Cartooino

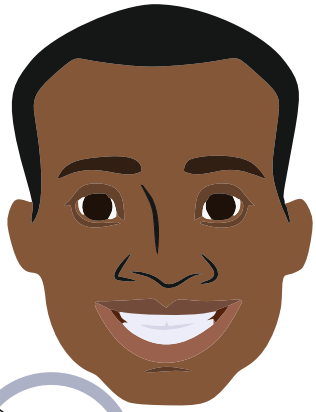
Introduction

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.



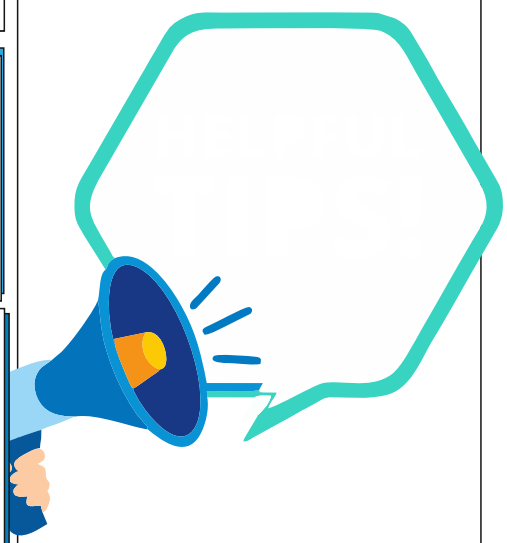
Emmanuel

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

GreenLab Cartooino

Stack of Content

Next Einstein	07
Mood Cue	08
YouNiversity	09
Little Symphony	10
Mini Piano	11
Crystal Ball	12



GreenLab Cartoono

EPISODE 7

Next EINSTIEN

Hello Uncle green.

Hello kids! welcome back to our seventh Arduino class. In today's class you will building a device to measure how hot you truly are.

Uncle is it going to be like the themometer?

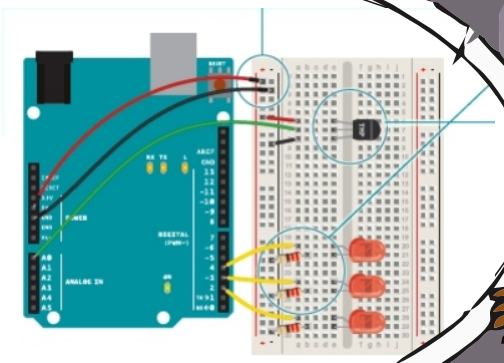
Yes it would, To complete this project we need three LED three 220 Ohms resistors, a temperature sensor, and some jumper wires. we are going to use 46minutes to complete this

I remember last when I wasn't feeling too well they used a thermometer to check my temperature .

Hope it wasn't serious oh well let's get started.

ooh.

The temperature sensor is the tiny black component in your kit with three legs labelled TMP. with this sensor will be measuring how warm your skin is. Once you are ready, look through your kit and connect the component as shown.



Ok Uncle.


MUSHIN, LAGOS
Dear let me check on Lolia.
Lolia have you washed the plates ?
Yes ma.
After that , make sure you go to bed.

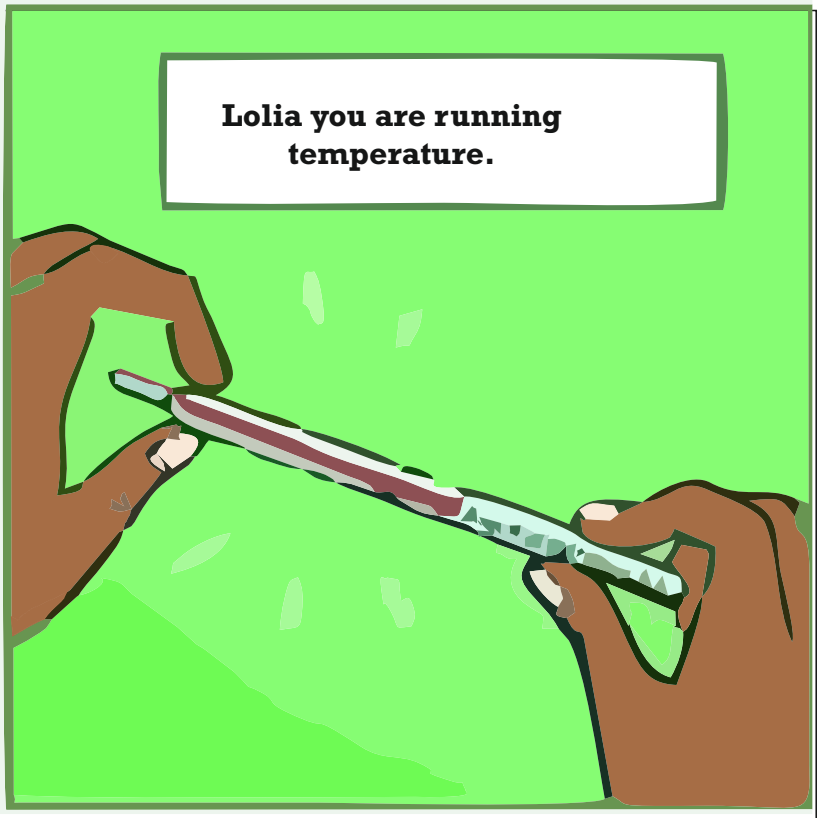
I woke up feeling pains in my tummy, Had to call my mummy because I couldn't sleep.

Lolia it's okay . In the morning I would check on you.

I threw up throughout the night.

Temperature

Temperature is the measure of hotness or coldness expressed in terms of any scale indicating the direction of heat energy will spontaneously flow in a body.



Thermometer

Thermometer is a device that measures temperature
A thermometer has two important elements.
1. A temperature sensor (which differs with the different types).
2. Visible scale .

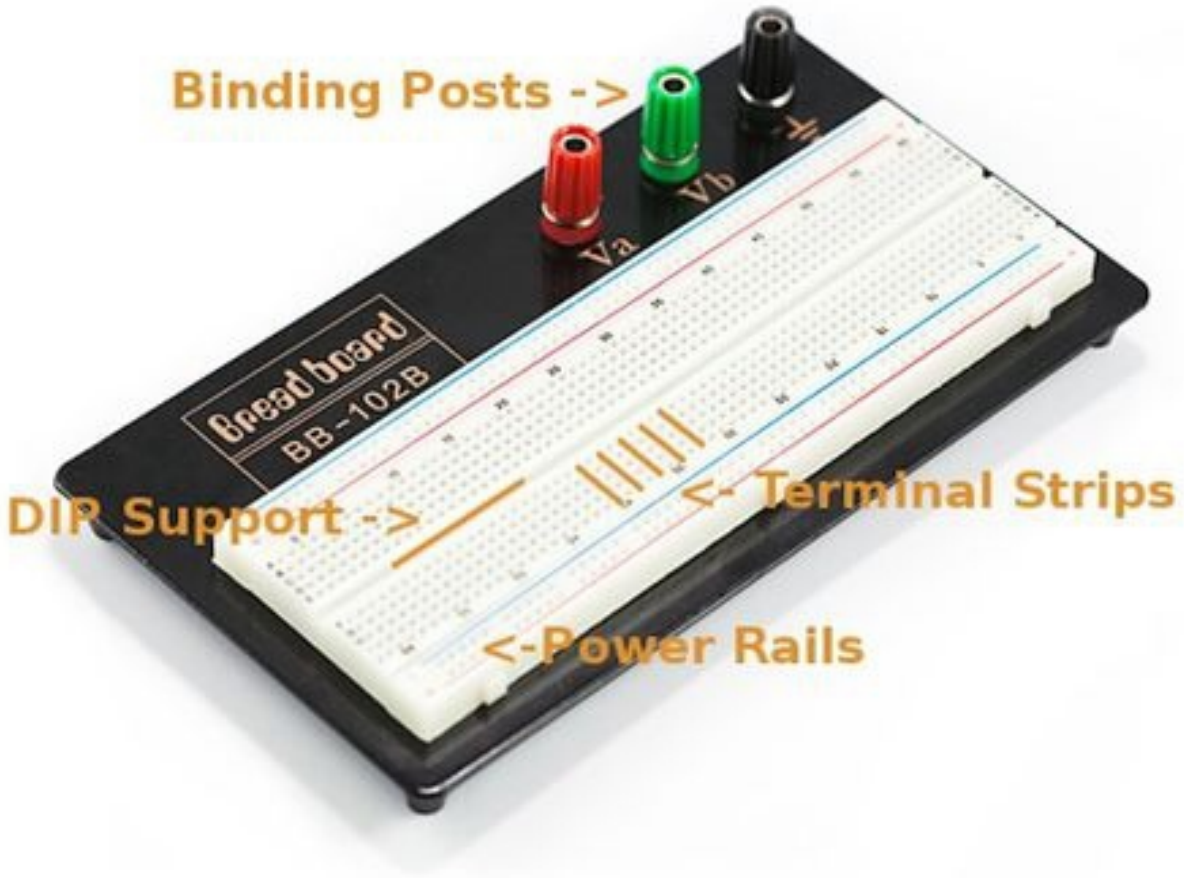
```
}// if the temperature rises 2-4 degrees, turn an LED on
else if(temperature >= baselineTemp+2 && temperature<baselineTemp+4){
  digitalWrite{2, HIGH};
  digitalWrite{3, LOW};
  digitalWrite{4, LOW};
} // if the temperature rises 4-6 degrees, turn a second LED on
else if(temperature >=baselineTemp+4 && temperature < baselineTemp+6}{
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
} // if the temperature rises more than 6 degrees, turn all LEDs on
  else if(temperature >= baselineTemp+6){
  digitalWrite(2. HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
 }
  delay(100);
}
```

With the code uploaded to the Arduino, click the serial monitor icon just as shown in the picture above. You should see a stream of values coming out, formatted like this: Sensor: 200, Volts: .70, degrees C: 17

Try putting your fingers around the sensor while it is plugged into the breadboard and see what happens to the values in the serial monitor. Make a note of what the temperature is when the sensor is left in the open air.

Close the serial monitor and change the baselineTemp constant in your program to the value you observed the temperature to be. Upload your code again, and try holding the sensor in your fingers. As the temperature rises, you should see the LEDs turn on one by one.

Practice Question



Identify the above image?

answer choices

- ☐ **Terminal Strips**
- ☐ **Power Rails**
- ☐ **Breadboard**
- ☐ **IDE**

“Episode 8 Mood Cue”

Mood Cue

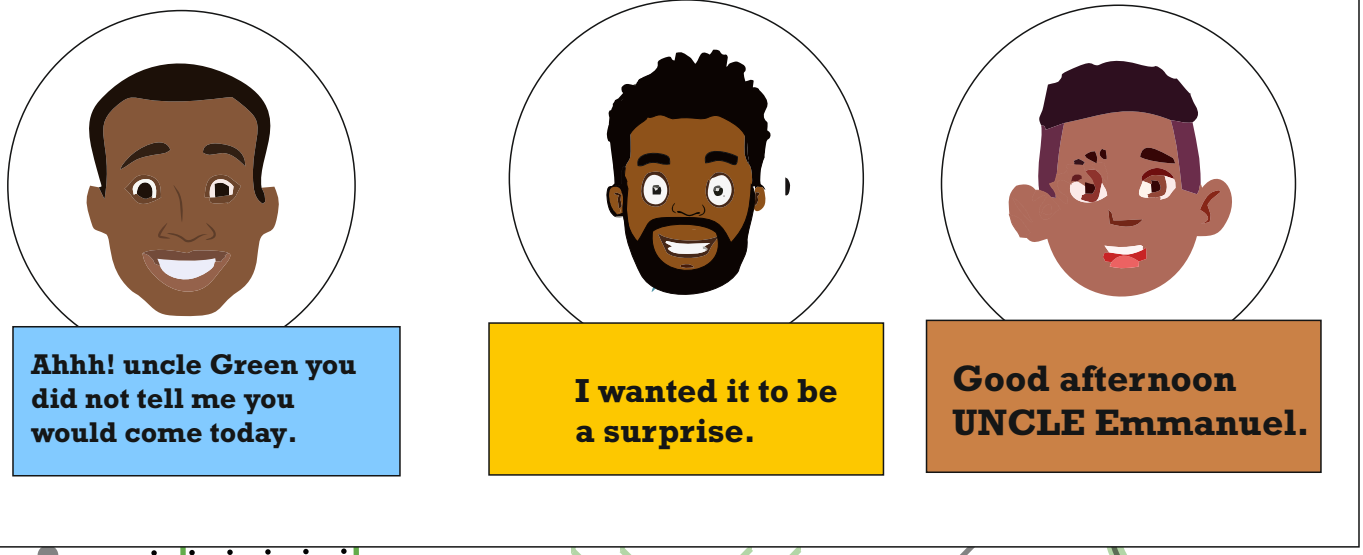
Uncle Emmanuel is a 400 Level Computer student at Federal University of Technology Akure. He however lives here in Lagos He would facilitate the remaining projects.

It will be fun I assure you . Uncle Emmanuel like kids.

WORD OF THE DAY

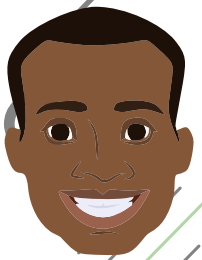
Current

An electric current is the rate of flow of an electric charge passed a point.



Hey Kids, arriving here today means not only that you are starting to familiarize with Arduino, but also that you are determined to be able to create amazing solutions with it. so first of all, we would like to congratulate you.

How many of you have been on a plane before?



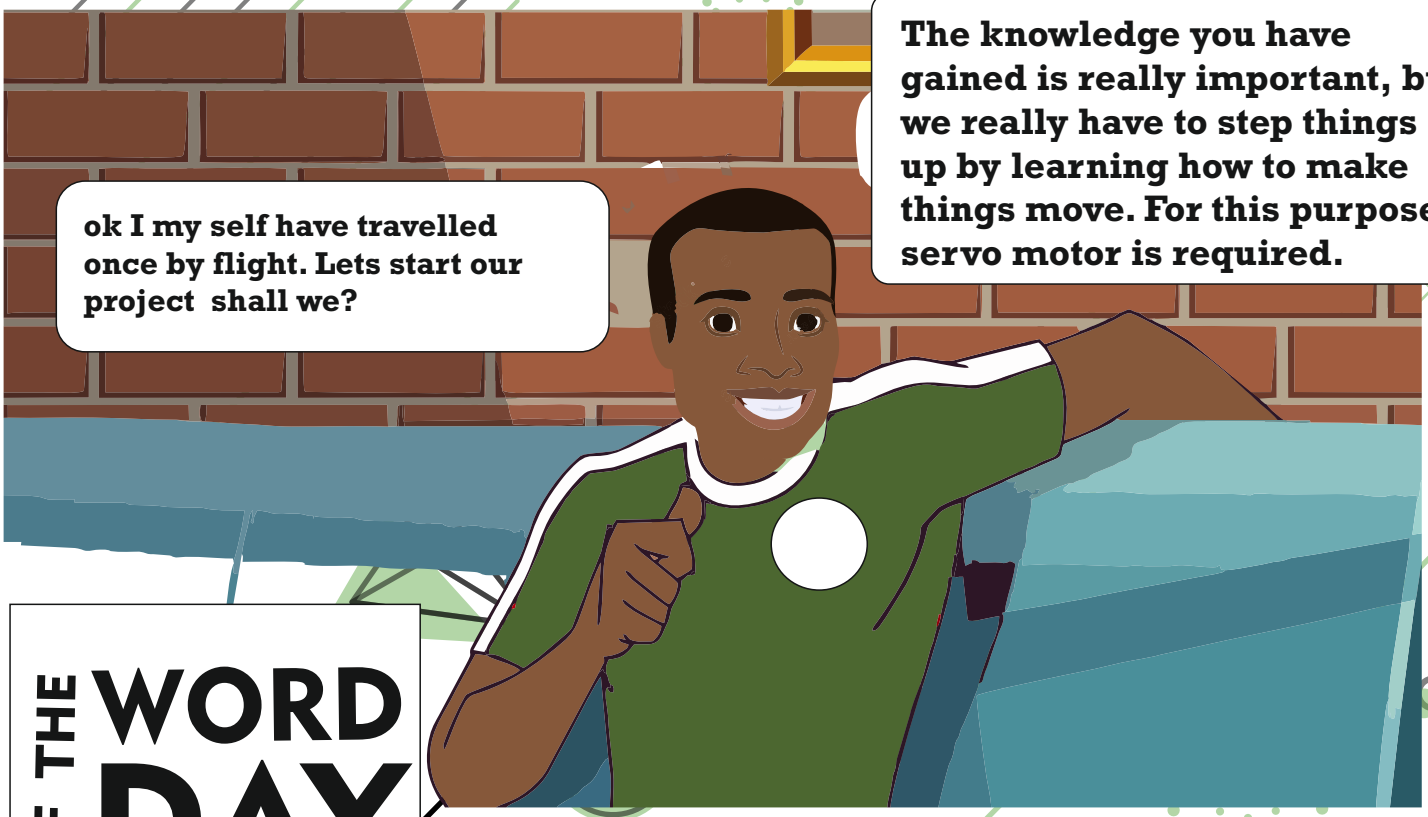
Me... sir I remember coming back from PortHarcourt, where I went to spend my holidays with my aunt.

The airport was very busy and had a lot of planes.

Port Harcourt International Airport

AIR INDUS

I played video games through out the flight.



WORD OF THE DAY

Jumper wires

These are wires or cables that have connector pins at the end, allowing them to be used to connect two point without soldering.

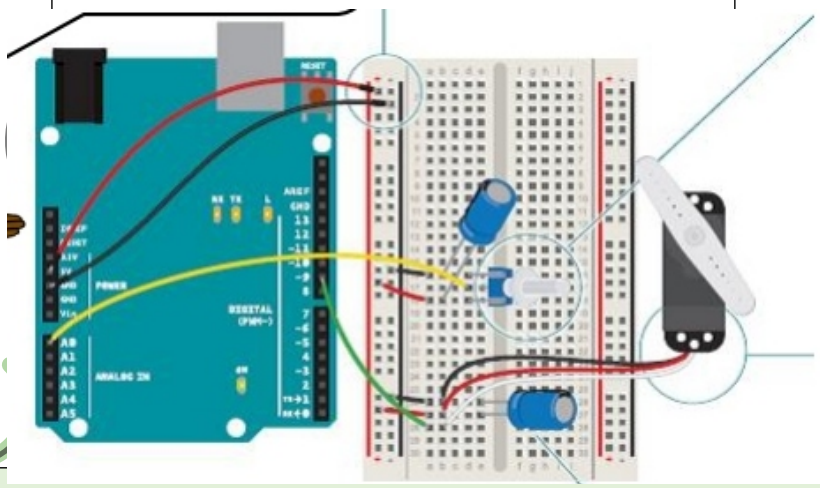
1x Potentiometer
1x Servomotor
2x 100UF Capacitors,
three male header pins
Some jumper wires
Lastly this project requires 60 minutes of our time. But before we proceed I would just explain briefly what a servomotor is.

In today's lesson, we will be making use of the following:
-one potentiometer,
-one servomotor
-two 100uF capacitors,
-three male header pins,
-and as usual some jumper wires.
Lastly, this project will require at least 60 min of your time. But before we proceed further, we will be giving a short explanation of the servomotor below.

The picture shows what the servomotor looks like. Servo motor are a special type of motor that don't spin around in a circle, but move to a specific position and stay there until you tell them to move again. Servos usually only rotate 180 degrees (one half of a circlr).

Similar to the way you used pulses to PWM on LED in the Colour Mixing Lamp Project, servo motors expect a number of pulses that tell them what angle to move to. The pulses always come at the same time intervals, but the width varies between 1000 and 2000 microseconds.
While it's possible to write code to generate these pulses, the Arduino software comes with a library that allows you to easily control the motor.
Because the servo only rotates 180 degrees, and your analog input goes from 0-1023, you'll need to use a function called map() to change the values coming from the potentiometer. One of the great things about the Arduino community are the talented people who extend its functionality through additional software. It's possible for anyone to write libraries to extend the Arduino's functionality. There are libraries for a wide range of sensors and actuators and other devices that users have contributed to the community. A software library expands the functionality of a programming environment. The Arduino software comes with a number of libraries that are useful for working with hardware or data. One of the included libraries is designed to use with servo motors. In your code, you'll import library, and all of its functionality will be available to you.
Before reading the code, not that a servomotor library is being used. After importing it, you can use all the functions that contains making the code much easier. As it is shown in the picture, the potentiometer (must be connected to an analog input as long as to the 5V entry and to the ground GND). Otherwise, the servo must be to a digital input as well as to the 5V and the GND. Make sure you connect the capacitors properly as they have polarity.

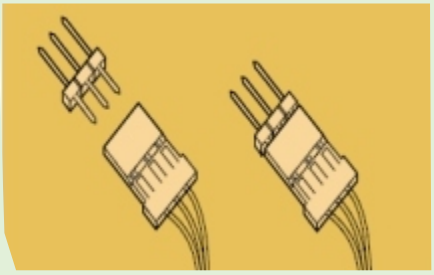
Once you are ready, connect the arduino board and breadboard as shown in the picture here. After this you have to enter the code on the arduino IDE.



How to connect

1. Attach 5V and ground to one side of your breadboard from the Arduino.
2. Place a potentiometer on the breadboard, and connect one side to 5V, and the other to ground. A potentiometer is a type of volt-age divider. As you turn the knob, you change the ratio of the voltage between the middle pin and power. You can read this change on an analog input. Connect the middle pin to analog pin 0. This will control the position of your servo motor.
3. The servo has three wires coming out of it. One is power (red), one is ground (black), and the third (white) is the control line that will receive information from the Arduino. Plug three male headers into the female ends of the servo wires (see Fig. 3). Connect the headers to your breadboard so that each pin is in a different row. Connect 5V to the red wire, ground to the black wire, and the white wire to pin 9.
4. When a servo motor starts to move, it draws more current than if it were already in motion. This will cause a dip in the voltage on your board. By placing a 100uf capacitor across power and ground right next to the male headers as shown in Fig. 1, you can smooth out any voltage changes that may occur. You can also place a capacitor across the power and ground going into your potentiometer. These are called decoupling capacitors because they reduce, or decouple, changes caused by the components from the rest of the circuit. Be very careful to make sure you are connecting the cathode to ground (that's the side with a black stripe down the side) and the anode to power. If you put the capacitors in backwards, they can explode.

Your servo motor should come with female connectors, so you'll need to add header pins to connect it to the breadboard as shown in this picture



The Code

```
#include <Servo.h>
Servo TestServo;
int const potPin=A0;
int potValue;
int angle;
void setup() {
TestServo.attach(9);
Serial.begin(9600);
}
void loop() {
potValue=analogRead(potPin);
Serial.print("potValue: ");
Serial.print(potValue);
angle=map(potValue, 0,1023,0,179);
Serial.print(",angle: ");
Serial.println(angle);
TestServo.write(angle);
delay(15);
}
```

How it works

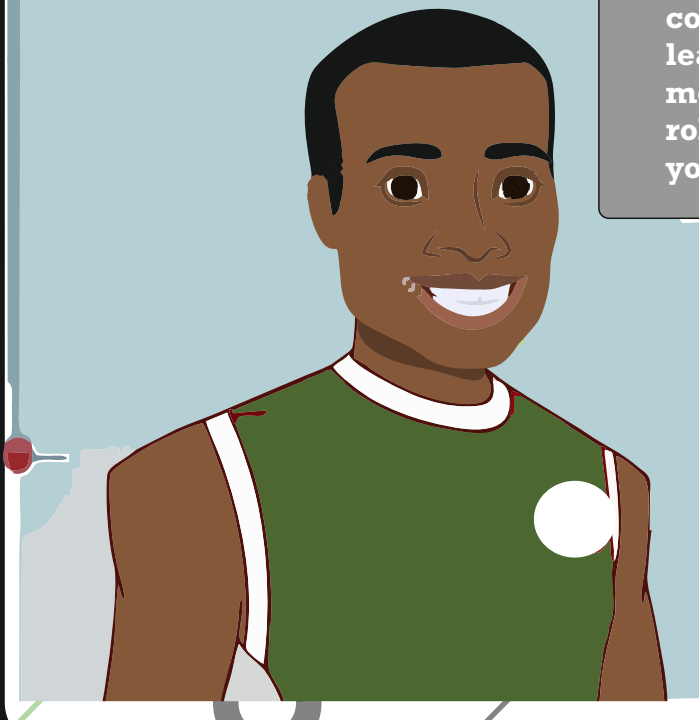
- To use the servo library, you'll first need to import it. This makes the additions from the library available to your sketch.
- To refer to the servo, you're going to need to create a named instance of the servo library in a variable. This is called an object. When you do this, you're making a unique name that will have all the functions and capabilities that the servo library offers. From this point on in the program, every time you refer to myServo, you'll be talking to the servo object.
- Set up a named constant for the pin the potentiometer is attached to, and variables to hold the analog input value and angle you want the servo to move to.
- In the setup(), you're going to need to tell the Arduino what pin your servo is attached to.
- Include a serial connection so you can check the values from the potentiometer and see how they map to angles on the servo motor.
- In the loop(), read the analog input and print out the value to the serial monitor.
- To create a usable value for the servo motor from your analog input, it's easiest to use the map() function. This handy function scales numbers for you. In this case it will change values between 0-1023 to values between 0-179. It takes five arguments : the number to be scaled (here it's potVal), the minimum value of the input (0), the maximum value of the input (1023), the minimum value of the output (0), and the maximum value of the output (179). Store this new value in the angle variable.
- Then, print out the mapped value to the serial monitor.
- Finally, it's time to move the servo. The command servo. write() moves the motor to the angle you specify. At the end of the loop() put a delay so the servo has time to move to its new position.

Once your Arduino has been programmed and powered up, open the serial monitor. You should see a stream of values similar to the one here

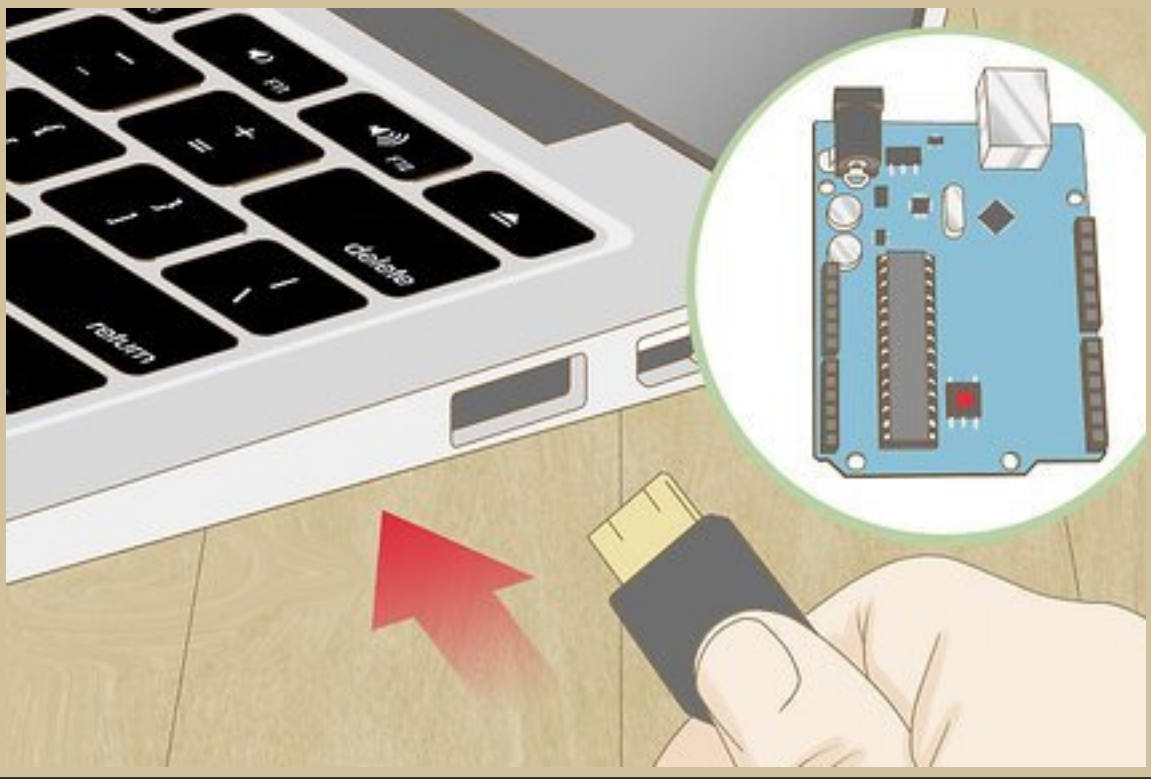
```
potVal : 1023, angle : 179
potVal : 1023, angle : 179
```

When you turn the potentiometer, you should see the numbers change. More importantly, you should see your servo motor move to a new position. Notice the relationship between the value of potVal and angle in the serial monitor and the position of the servo. You should see consistent results as you turn the pot.
One nice thing about using potentiometer as analog inputs is that they will give you a full range of values between 0 and 1023. This makes them helpful in testing projects that use analog input.

Congratulations! You have successfully completed the project. In this project you learnt how to make things move using a servo motor. Now your dream of making your toy robot id 70% accomplished. See you when you come around.



Practice Question



Identify the space on the Laptop that connect the Arduino Board

- ☐ **HDMI**
- ☐ **USB**
- ☐ **BLUETOOTH**
- ☐ **VGA**

Ayomide where are you going to...

Can I come along?

Kunle I am going to Uncle Emmanuel house; He is our Arduino facilitator.

Yes you can.

This is Mr Emmanuel's house.

Ayomide, are we going to write? Because I would need a pen.

Wait Ayomide I forgot my pen.

I think I have a spare.

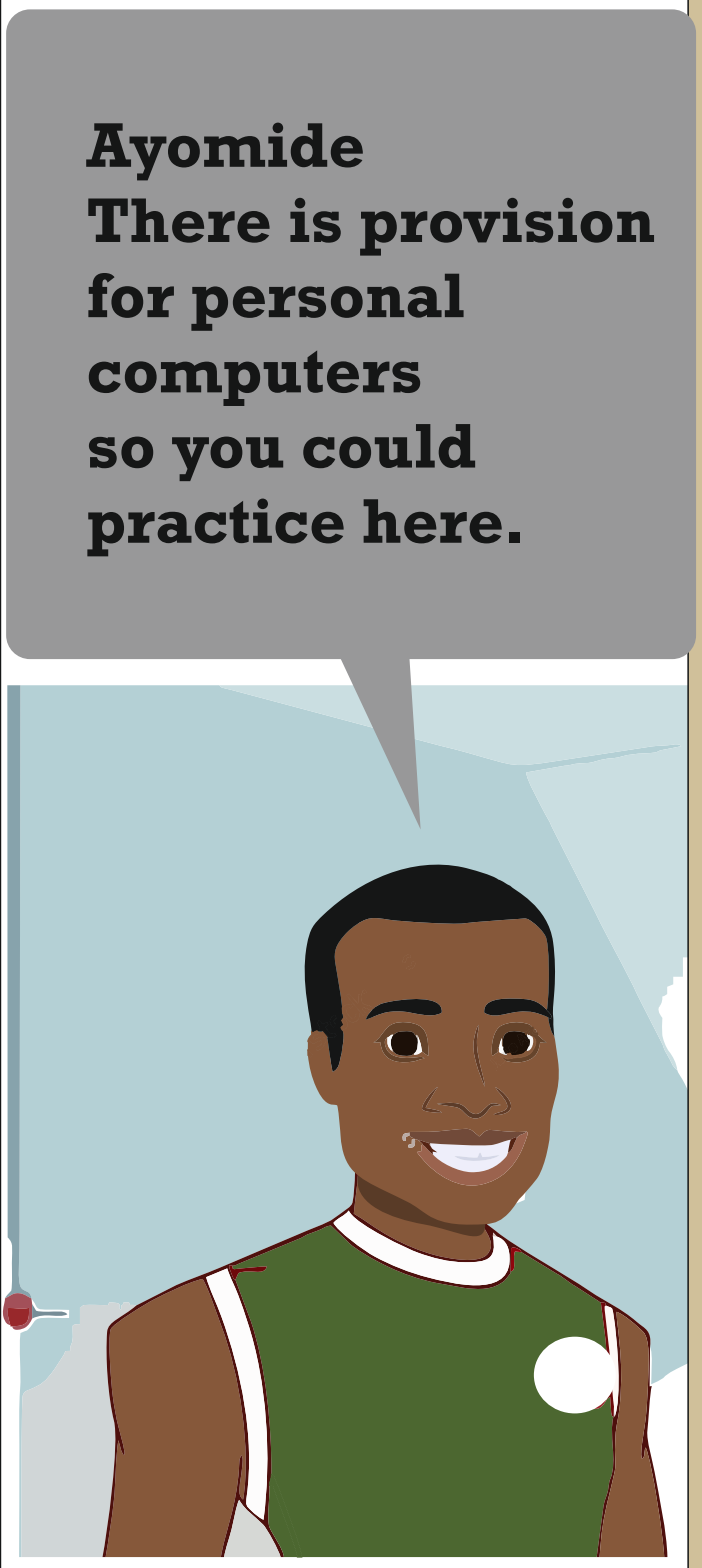
Welcome to Green Lab Microfactory

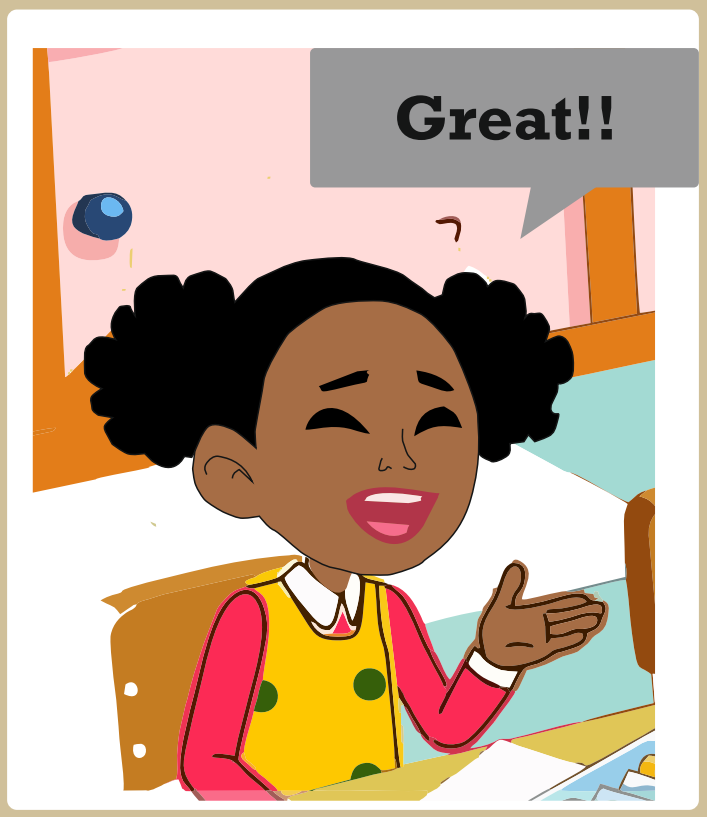
Hello kids you can call me uncle EMMANUEL, so how are you all doing today and how was school, I guess it went well.
So one of the most basic and fun thing you can do with Arduino micro-controller is to wire a DC motor. You can make a simple table fan or go all in and build a remote controlled car.

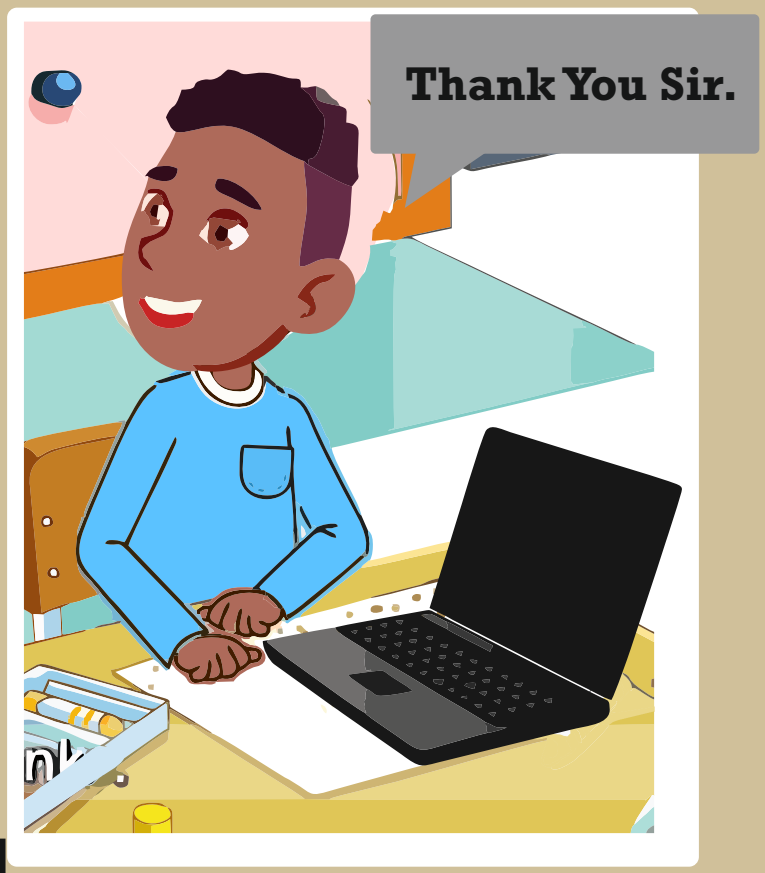
What is a Cathode ?
The negative end of a diode.

Tip

What are Datasheet - A classiification system that describes the functionality of a component. Typical information in a datasheet includes the maximum voltage and current a component requires, as well as an explanation of the functionality of the pins.

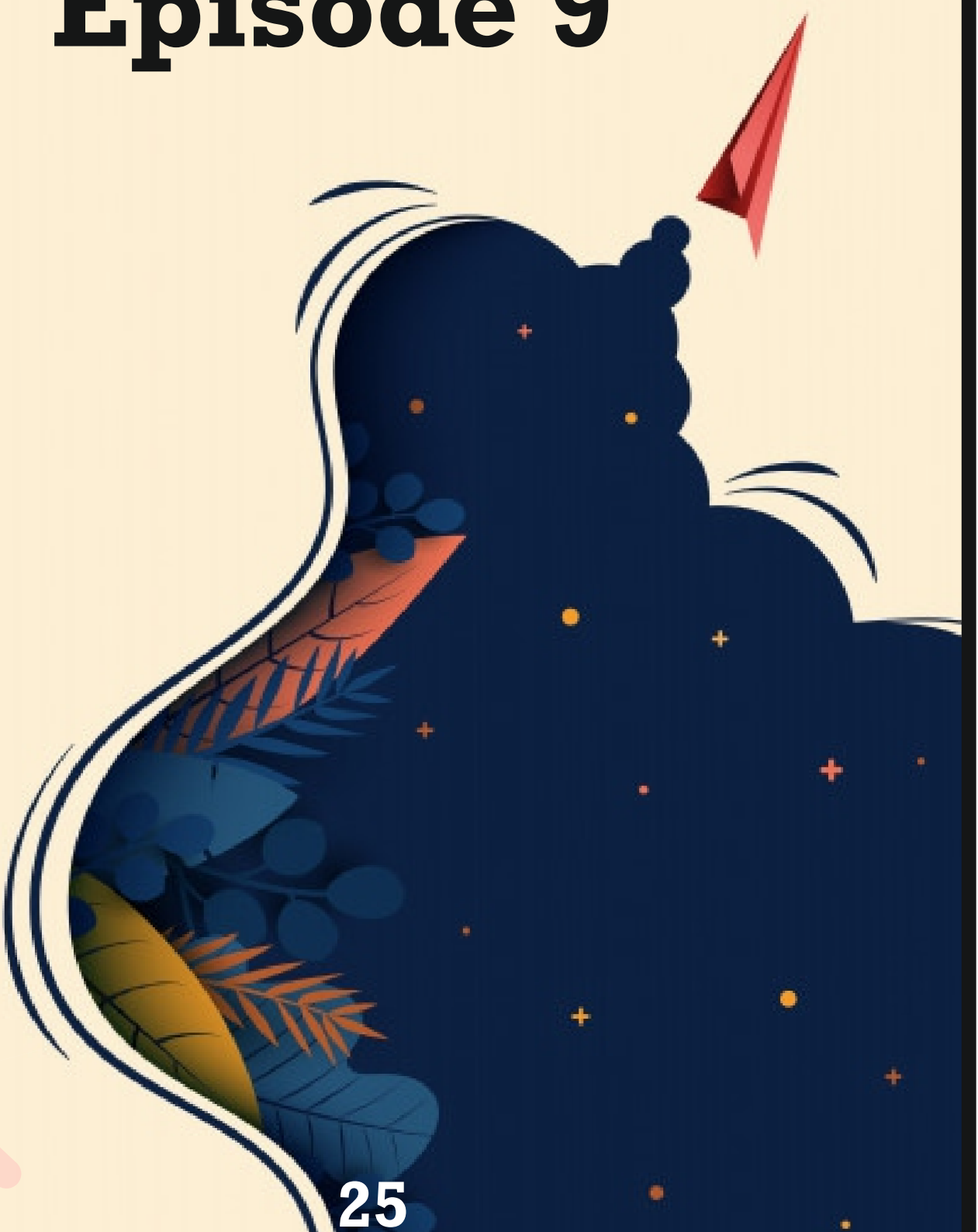


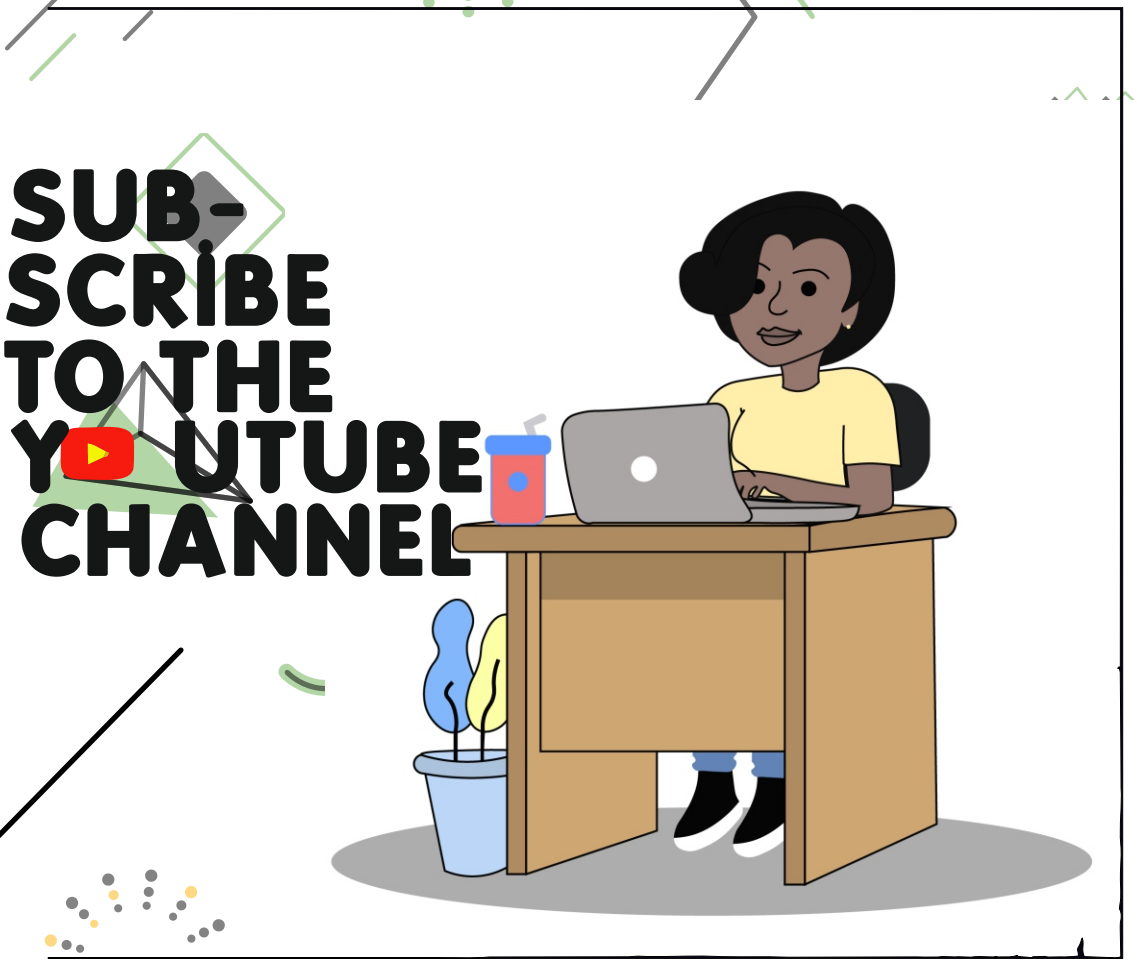




Your Rights to Innovat

Episode 9





Hi Guys
Welcome back to YouNiversity!
Just like you did in the last course.

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

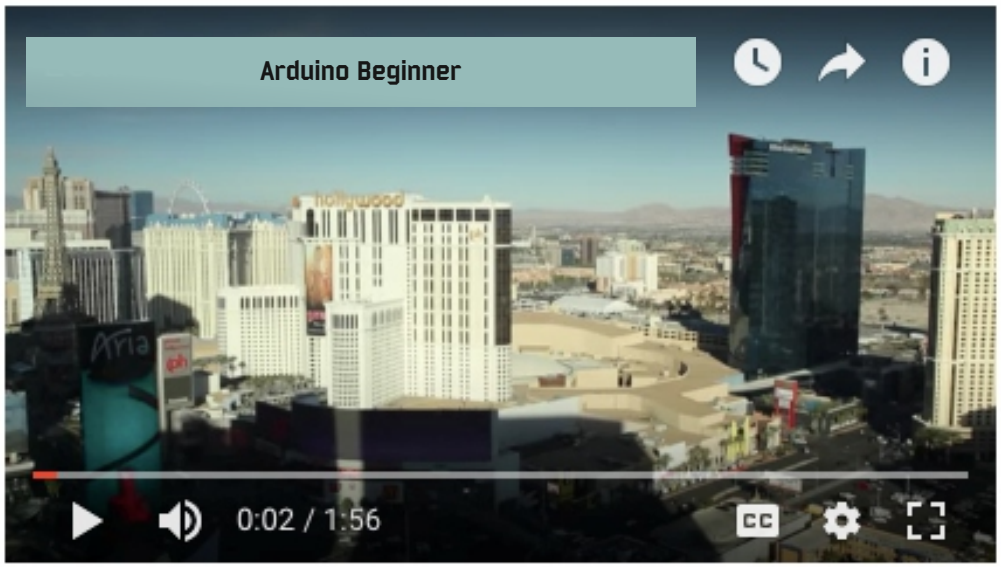
GreenLab
MICRO-FACTORY

Mr GREEN

105,345 subscribers

SUBSCRIBE 106K

HOME VIDEOS PLAYLISTS COMMUNITY



Practice Question

Q. A function is a series of program statements that can be called by name. Which command is repeated over and over as long as the Arduino has power.

answer choices----------

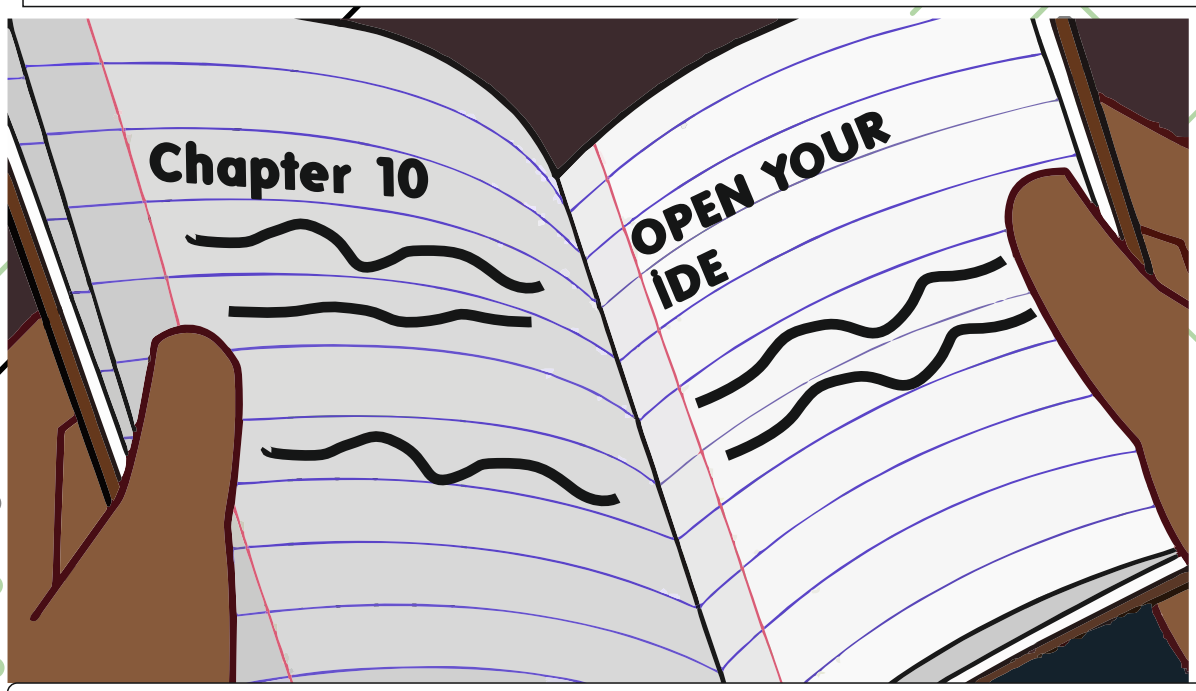
- ☐ **loop()**
- ☐ **setup()**
- ☐ **(output)**
- ☐ **(input)**

Arduino Fun Facts

As of last year, there were more than 1.2 million Arduino boards in the world, not to mention probably just as many counterfiets.

There was over 217 kickstater project and counting built on Arduino platform in the last 12 months alone.

Even facebook likes Arduino! During the F8 Developers Conference, Parse announced a lineup of SDKs for the internet of things.



The next project is Little Symphony
Prepare yourselves kids and come with your note books.

Uncle Emmanuel How do we enroll for YouNIversity.

Uncle Emmanuel do we need data to go online.

Uncle after we are done I would love to present my project first.

First of all open your browser and sign in to youtube.Search for GreenLab Micro factory and click Arduino.

Yes Kunle you need mobile data to access Youtube. If you don't have much data then you could come to the garage.

Tip

What are actuators -
A type of component that changes an electrical energy into motion. Motors are a type of electrical actuator..

#Tip

BAUD - Shorthand for ”bits per second”, signifying the speed at which two devices are communicating.

Kids you have to take note while spending time on YouNIversity Projects, so you can ask questions later.

Arduino has inspired countless young makers to pursue a career in engineering, some of whom have gone on to launch their own businesses and become crowfunding successes .

Practice Question

Q. The Object in the diagram is called_____ and it is needed when carrying out movement.

answer choices

- ☐ Arduino Uno Board
- ☐ Breadboard
- ☐ Jumper Cables
- ☐ White Plastic

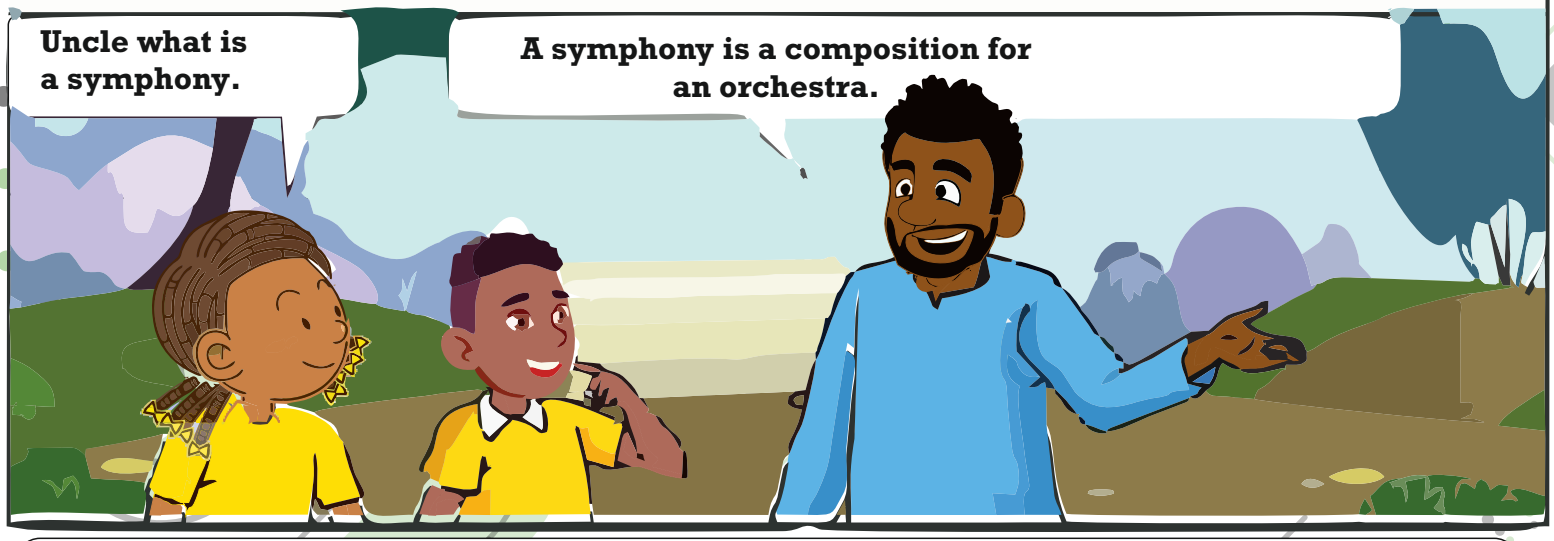
Episode 10

Little Sympony

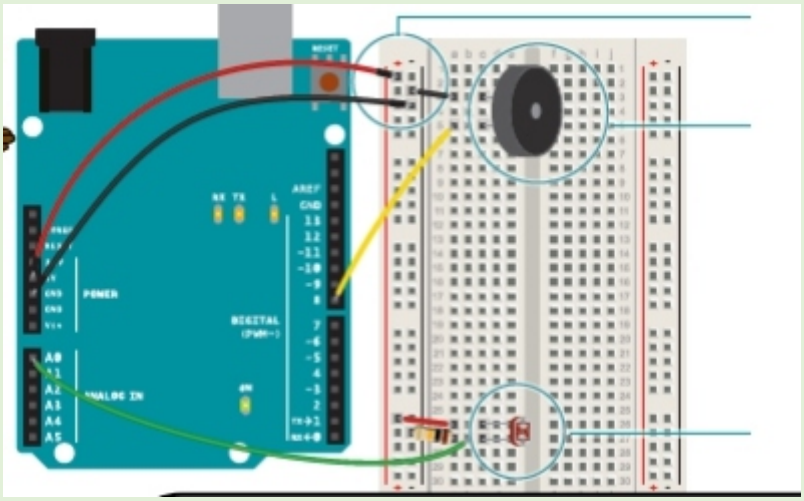
Hy kids I see, you are playing football. Hope you are still in good mood after the last project.

Episode 10 Little sympony

Today we would be doing yet another project, we would be building a little symphony using a theremin.



A theremin is an instrument that makes sounds based on the movement of a musician's hands around the instrument. A piezo is a small element that vibrates when it receives electricity. When it moves, it displaces air around it, creating sound waves. To build project, a piezo element, a Photoresistor and a 10kiloohm resistor. Lastly, you we need 45minutes to complete the project.



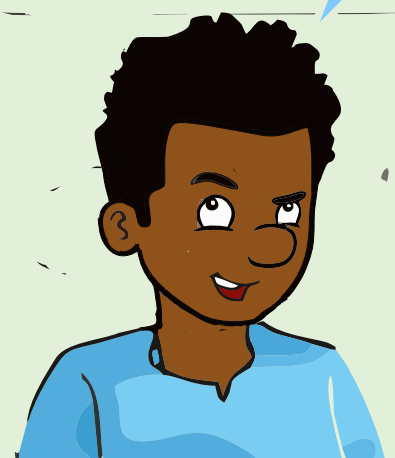
Once you are ready please make the connection as shown on this picture. In case you need more information on the components, please consult the introduction.

Take your piezo, and connect one end to ground, and the other to digital pin 8 on the Arduino.
Place your photoresistor on the breadboard, connecting one end to 5V. Connect the other end to the Arduino's analogIn pin 0, and to ground through a 10-kilohm resistor.

Instead of sensing capacitance with the Arduino, you'll be using a photoresistor to detect the amount of light. By moving your hands over the sensor, you'll change the amount of light that falls on the photoresistor's face, as you did in Project 4. The change in the voltage on the analog pin will determine what frequency note to play.

You'll connect the photoresistors to the Arduino using a voltage divider circuit like you did in Project 4. You probably noticed in the earlier project that when you read this circuit using analogRead(), your readings didn't range all the way from 0 to 1023. The fixed resistor connecting to ground limits the low end of the range, and the brightness of your light limits the high end. Instead of settling for a limited range, you'll calibrate the sensor readings getting the high and low values, mapping them to sound frequencies using the map() function to get as much range out of your theremin as possible. This will have the added benefit of adjusting the sensor readings whenever you move your circuit to a new environment, like a room with different light conditions.

The theremin detects where a performer's hands are in relation to two antennas. These antennas are connected to analog circuitry that creates the sound. One antenna controls the frequency of the sound and the other controls volume. While the Arduino can't exactly replicate the mysterious sound from this instrument, it is possible to emulate them using the tone() function.



About the Code

Create a variable to hold the analogRead() value from the photoresistor. Next, create variables for the high and low values. You're going to set the initial value in the sensorLow variable to 1023, and set the value of the sensorHigh variable to 0. When you first run the program, you'll compare these numbers to the sensor's readings to find the real maximum and minimum values.
Create a constant named ledPin. You'll use this as an indicator that your sensor has finished calibrating. For this project, use the on-board LED connected to pin 13.
In the setup(), change the pinMode() of ledPin to OUTPUT, and turn the light on.
The next steps will calibrate the sensor's maximum and minimum values. You'll use a while() statement to run a loop for 5 seconds. while() loops run until a certain condition is met. In this case you're going to use the millis() function to check the current time. millis() reports how long the Arduino has been running since it was last powered on or reset.
In the loop, you'll read the value of the sensor; if the value is less than sensorLow (initially 1023), you'll update that variable. If it is greater than sensorHigh (initially 0), that gets updated.
When 5 seconds have passed, the while() loop will end. Turn off the LED attached to pin 13. You'll use the sensor high and low values just recorded to scale the frequency in the main part of your program.
In the loop(), read the value on A0 and store it in sensorValue.
Create a variable named pitch. The value of pitch is going to be mapped from sensorValue. Use sensorLow and sensorHigh as the bounds for the incoming values. For starting values for output, try 50 to 4000. These numbers set the range of frequencies the Arduino will generate.
Next, call the tone() function to play a sound. It takes three arguments : what pin to play the sound on (in this case pin 8), what frequency to play (determined by the pitch variable), and how long to play the note (try 20 milliseconds to start).
Then, call a delay() for 10 milliseconds to give the sound some time to play.

NOTE:

OPEN SOURCE- Resource That can be used, redistributed or rewritten free of charge.

ELECTRONICS- Technology which makes use of the controlled motion of electronics through different media.

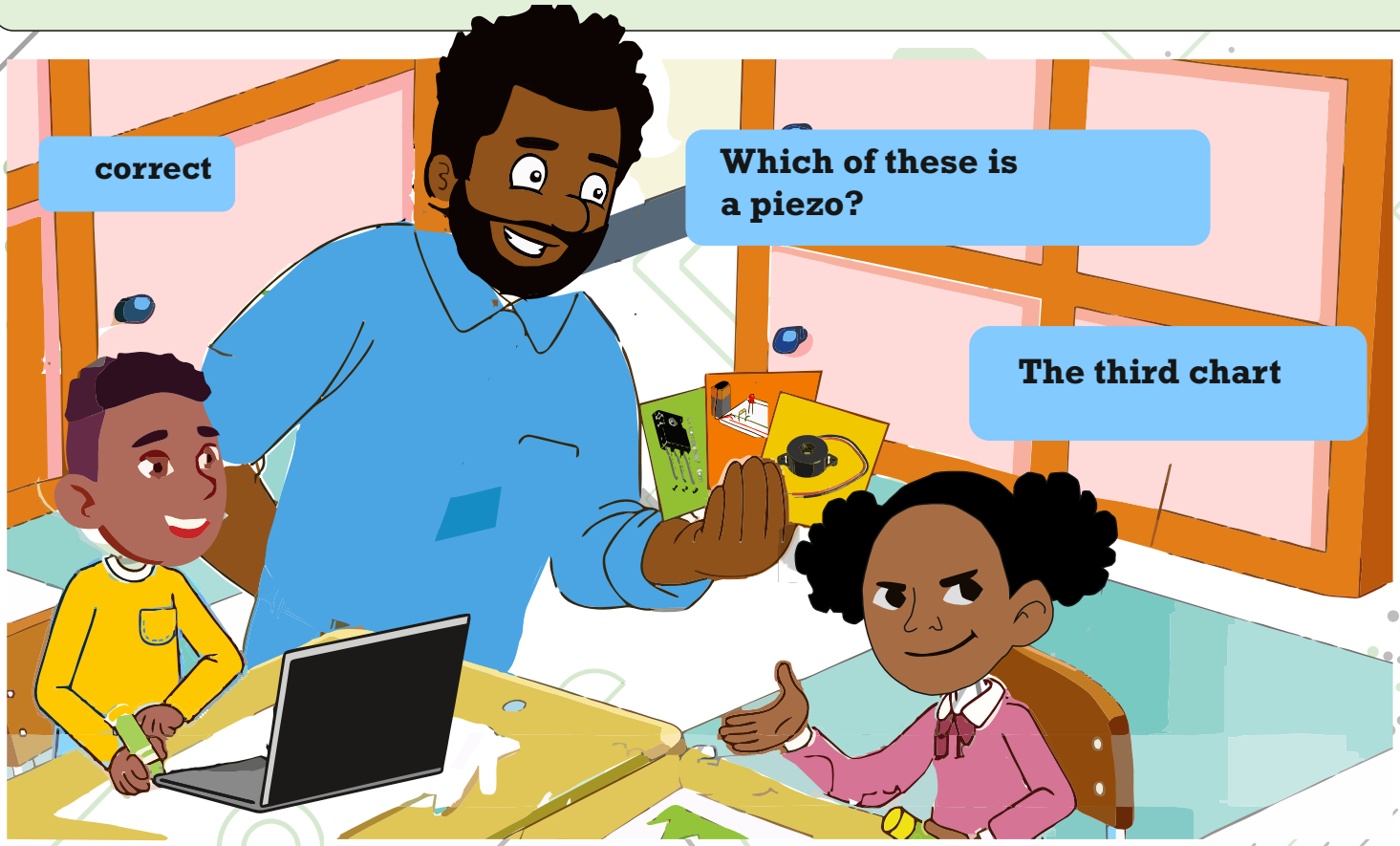
PROTOTYPE- An Original form that can serve as a basis or standard for other things.

PLATFORM: Hardware Architecture with software framework on which other software can run.

How it works

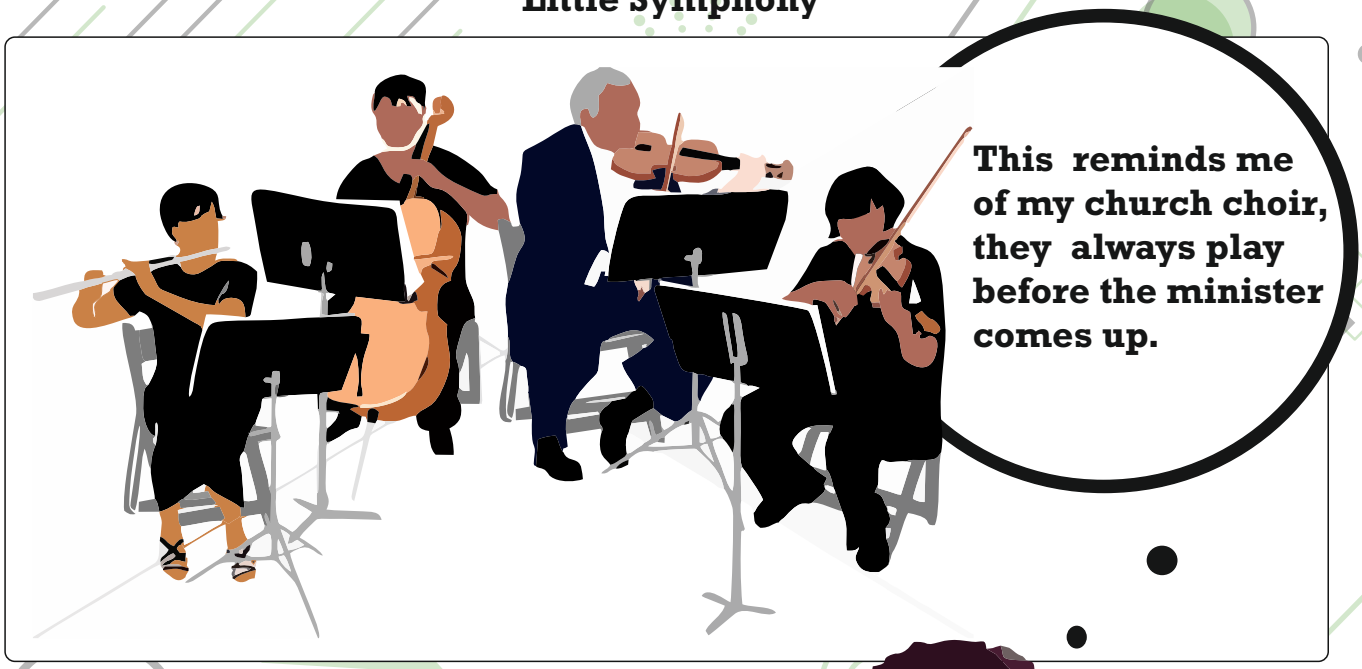
When you first power the Arduino on, there is a 5 second window for you to calibrate the sensor. To do this, move your hand up and down over the photoresistor, changing the amount of light that reaches it. The closer you replicate the motions you expect to use while playing the instrument, the better the calibration will be.

After 5 seconds, the calibration will be complete, and the LED on the Arduino will turn off. When this happens, you should hear some noise coming from the piezo! As the amount of light that falls on the sensor changes, so should the frequency that the piezo plays.



The Code

```
int sensorValue;
 // variable to calibrate low value
 int sensorLow = 1023;
 // variable to calibrate high value
 int sensorHigh = 0;
 // LED pin
 const int ledPin = 13;
void setup() {
 // Make the LED pin an output and turn it on
 pinMode(ledPin, OUTPUT);
 digitalWrite(ledPin, HIGH);
 // calibrate for the first five seconds after program runs
 while (millis() < 5000) {
 // save the maximum sensor value
 sensorValue = analogRead(A0);
 if (sensorValue > sensorHigh) {
 sensorHigh = sensorValue;
 }
 // save the minimum sensor value
 if (sensorValue < sensorLow) {
 sensorLow = sensorValue;
 }
 }
 //turn the LED off, signaling the end of the calibration
 digitalWrite(ledPin, LOW);
 }
void loop() {
 //read the input from A0 and store it in a variable
 sensorValue=analogRead(A0);
 // map the sensor values to a wide range of pitches
 int pitch=map(sensorValue, sensorLow, sensorHigh, 50, 4000);
 // play the tone for 20 ms on pin 8
 tone(8, pitch, 20);
 // wait for 10ms
 delay(10);
 }
```


This reminds me of my church choir, they always play before the minister comes up.


Oh

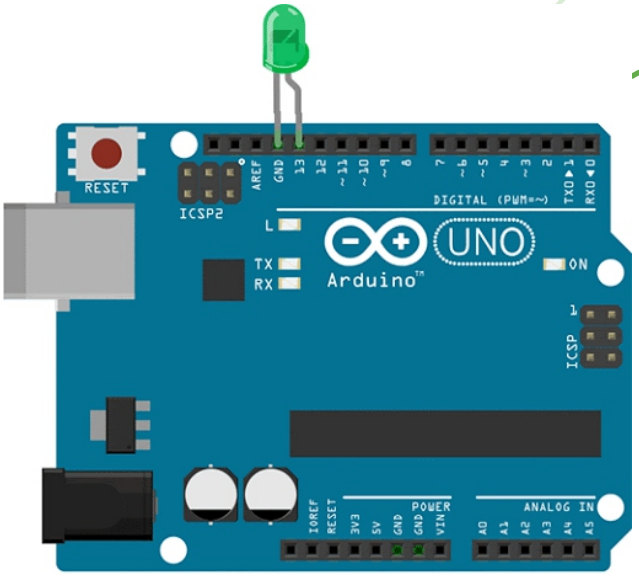

Let me read the instructions out.
I think we should follow through step by step.
Let check our manuals, for further details about this project.
Arduino 20
Arduino 20
Arduino 20

Practice Question

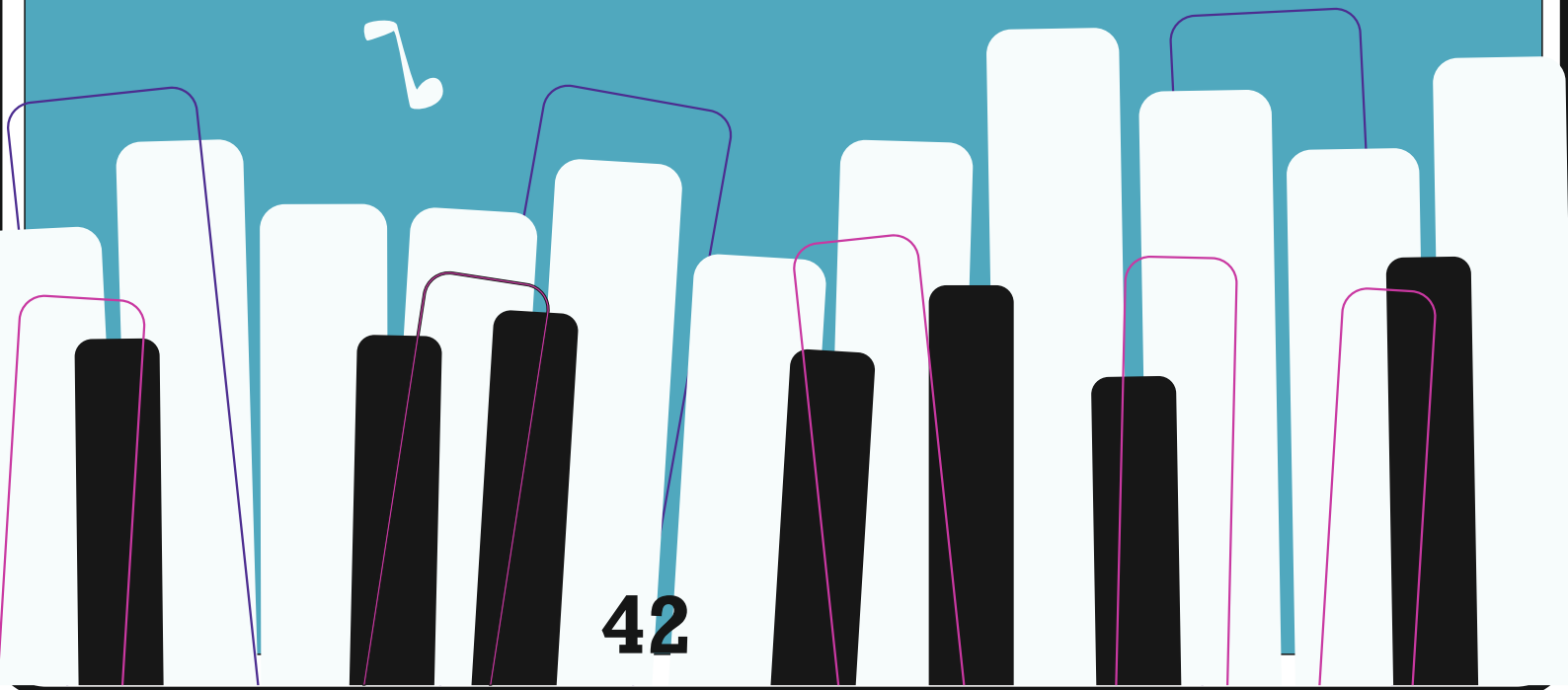
The Arduino needs power to run and then we need to attach it to the computer?

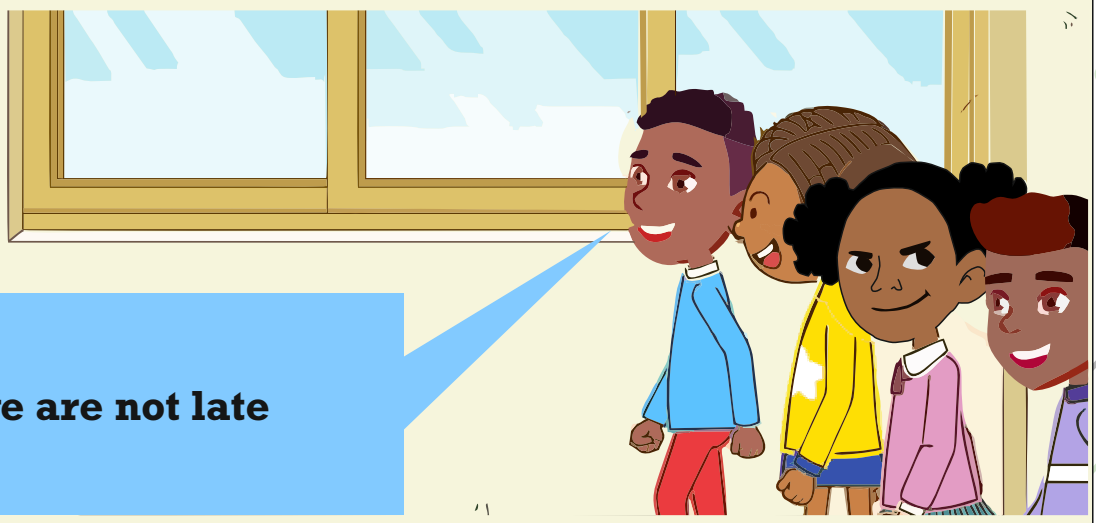
answer choices

- True
- False
- Not sure
- Undefinied



Episode 11
mini piano

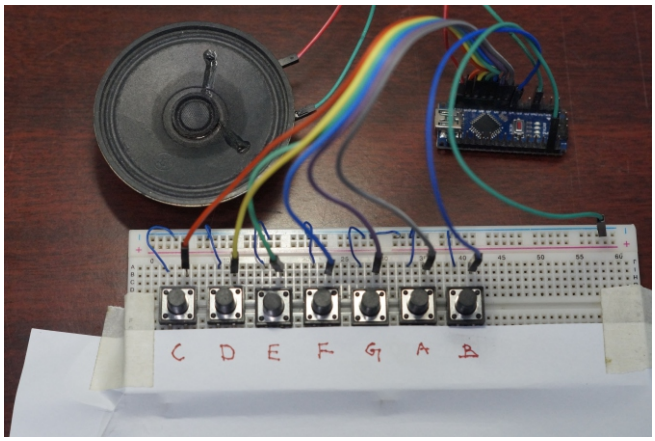




I love the piano.

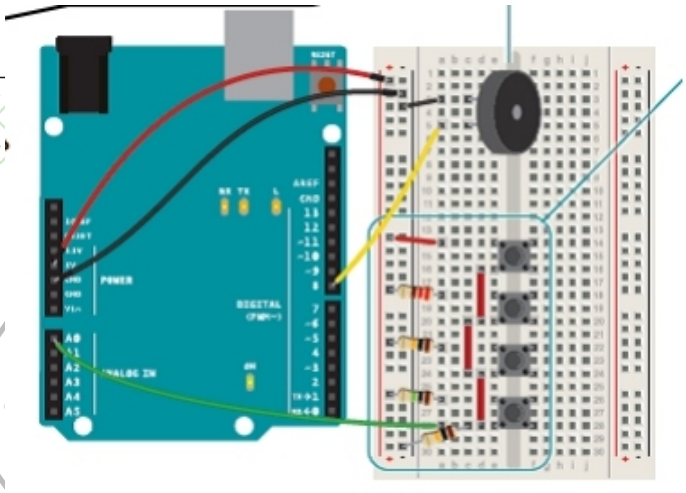
Me.. I can play, I have played severally in church.

So today kids, I will show you
how to make an Arduino based Piano.
It is a simple project made using Arduino UNO, few push buttons and a Piezo Buzzer. A special feature of this project is that Arduino will record the last played set of tones and repeat those tones like a one time record and repeat feature.



Understanding the code
We know that Arduino is capable of producing PWM signals. I will be using this feature of Arduino to generate tones. The other way to generate tones using Arduino is to use the function tone () function.
Using tone () function, you can generate square waves of different frequencies but with fixed duty cycle (50%). Internally, the tone () function relies on the Timers of Arduino (or the ATmega328P Microcontroller, to be more accurate).

Circuit Diagram
The following image shows the circuit diagram for piano application using Arduino. As you can see from the circuit diagram, it is a fairly simple circuit.



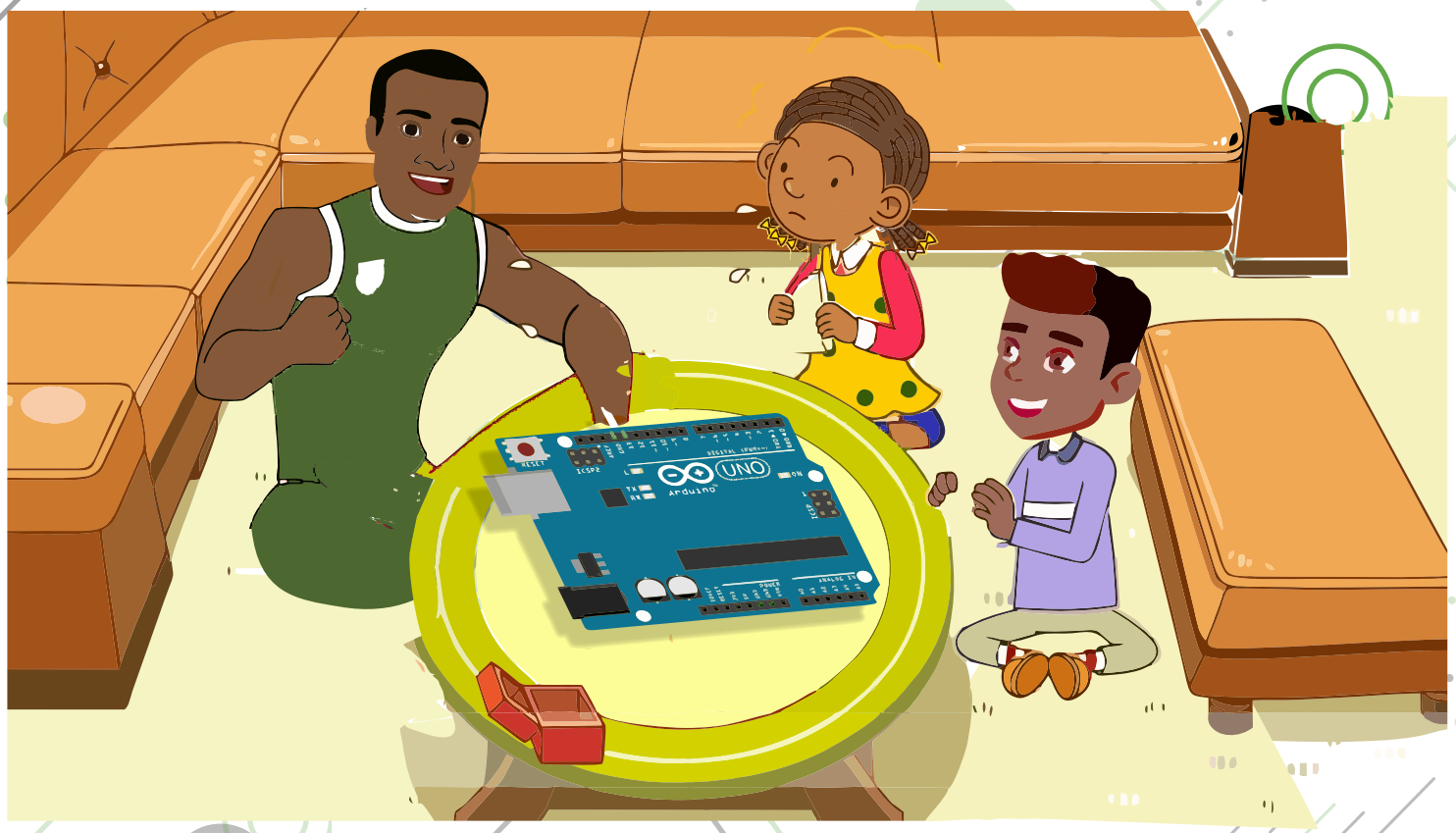
Components Required
-Arduino UNO
-Push Buttons X 8
-Small Piezo Buzzer (or a small speaker)
-Connecting Wires
-Breadboard
-Power Supply
-Circuit Design

The design of Arduino Piano circuit is very easy. First, connect a 5V Piezo Buzzer i.e. its positive terminal to Pin 10 of Arduino. It is necessary that you connect the Piezo Buzzer to one of the PWM capable pins of Arduino. The other end of the Piezo Buzzer is connected to GND.

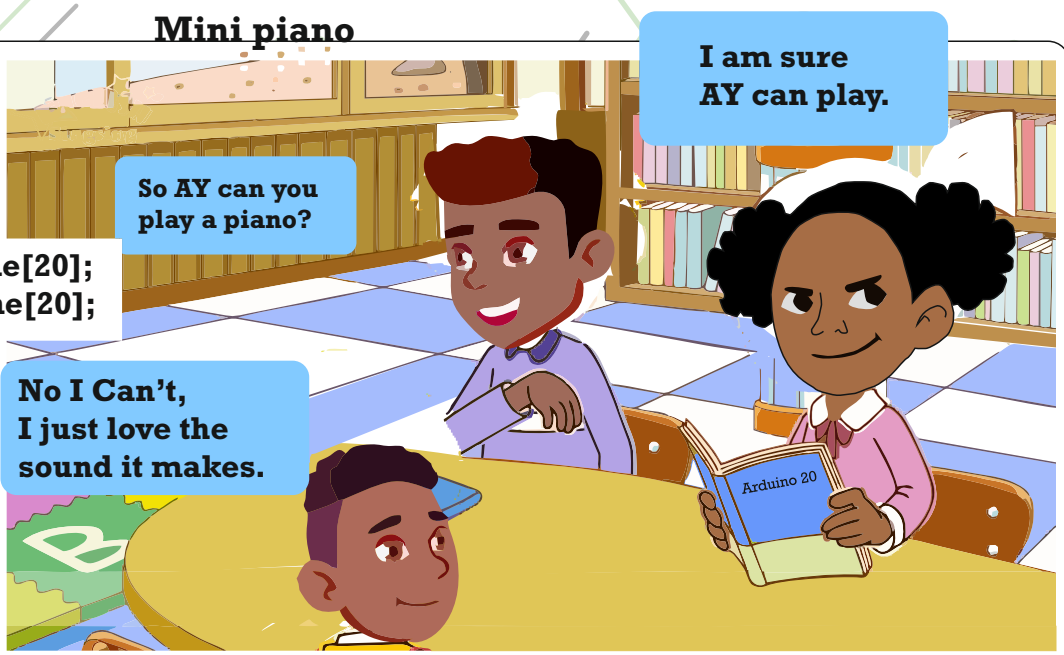
Now, connect 7 Push Buttons to digital I/O pins 3 through 9 of Arduino. These pins act as the tone input pins. I have used the INTERNAL PULL UP feature of Arduino and hence I haven't connected any external pull-up resistors to these pins.

All the other terminals of these push buttons are connected to GND. Finally, another push button is connected to Pin 2 of Arduino to act as an Interrupt pin. The other end of this button is also connected to GND.

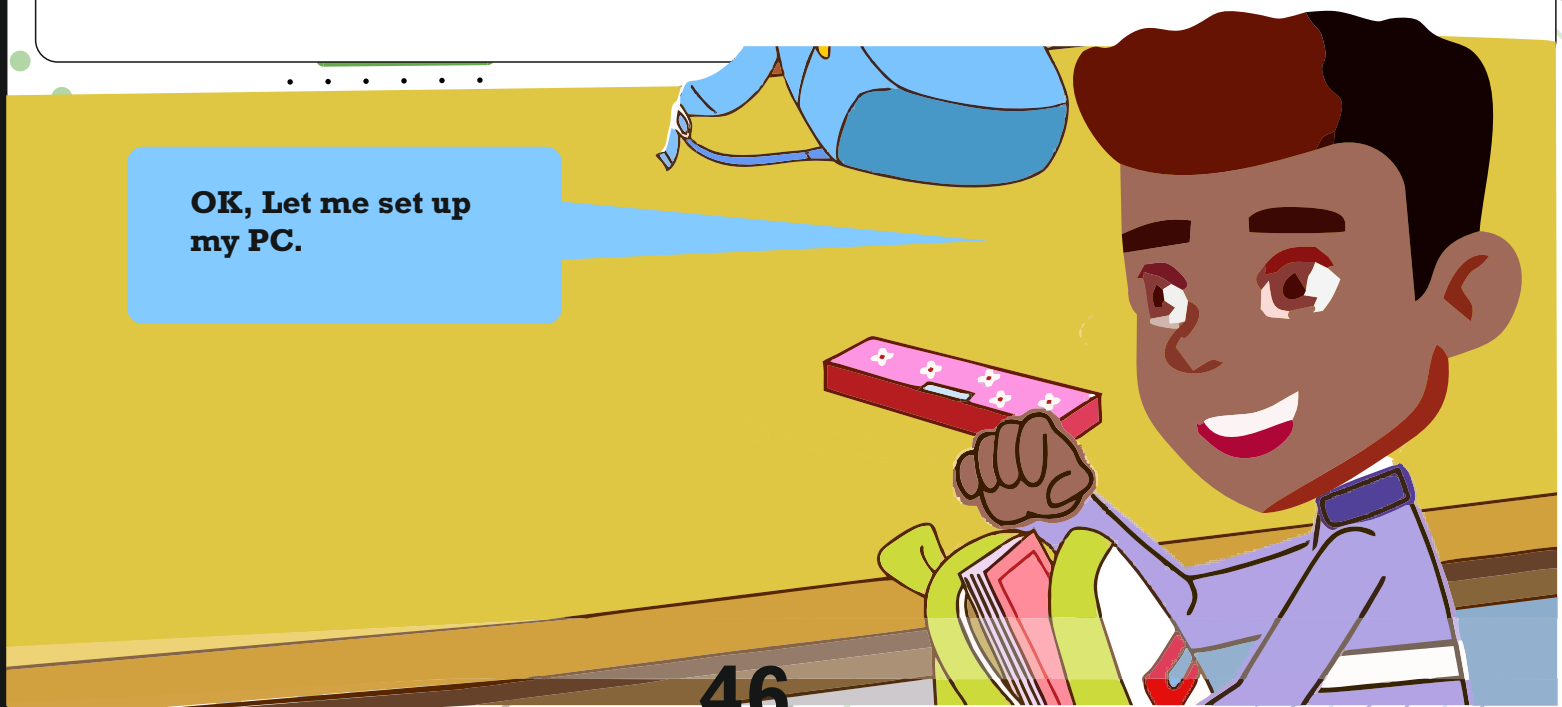
I have used the on-board LED (LED connected to Pin 13) to indicate between regular tone play and recorded tone play.



```
nt val=0;
int buzzer = 10;
unsigned long on_time=0;
unsigned long off_time=0;
unsigned long button_ontime[20];
unsigned long button_offtime[20];
int button_seq[20];
int button1=3;
int button2=4;
int button3=5;
int button4=6;
int button5=7;
int button6=8;
int button7=9;
int button8=10;
int frequency[] = {262, 294, 330, 349, 392, 440, 494};
```



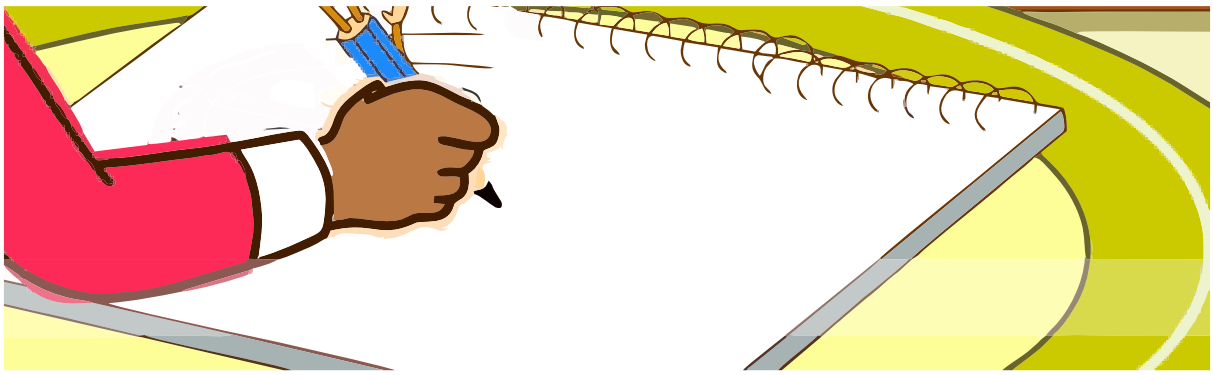

```
int buttonPin = 2;
int previousState = HIGH;
unsigned int previousPress;
volatile int buttonFlag;
int buttonDebounce = 20;


int path=1;
int i=0;
int led=13;
void playback (void);

void setup()
{
Serial.begin(9600);
pinMode(buzzer,OUTPUT);
pinMode(led,OUTPUT);

///////////////
pinMode(button1,INPUT_PULLUP);
pinMode(button2,INPUT_PULLUP);
pinMode(button3,INPUT_PULLUP);
pinMode(button4,INPUT_PULLUP);
pinMode(button5,INPUT_PULLUP);
pinMode(button6,INPUT_PULLUP);
pinMode(button7,INPUT_PULLUP);
pinMode(button8,INPUT_PULLUP);
pinMode(buttonPin,INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(2), button_ISR, CHANGE);
analogWrite(buzzer,0);
digitalWrite(led,HIGH);
}
```



```
I
void loop()
{
if(path==0)
{
 Serial.println("playback");
 playback();
}
if((millis() - previousPress) > buttonDebounce && buttonFlag)
 {
  previousPress = millis();
  if(digitalRead(buttonPin) == LOW && previousState == HIGH)
  {
   path =! path;
   previousState = LOW;
  }

  else if(digitalRead(buttonPin) == HIGH && previousState == LOW)
  {
   previousState = HIGH;
  }
  buttonFlag = 0;
 }

if(digitalRead(button1)==LOW)
{
 analogWrite(buzzer,frequency[0]);
 on_time=millis();
 if(i>0)
 {
  button_offtime[i-1]=on_time-off_time;
 }
 while(digitalRead(button1)==LOW);
 if(path==1)
 {
  off_time=millis();
  button_ontime[i]=(off_time-on_time);
  button_seq[i]=0;
  i++;
  Serial.println("button 1 stored");
 }
}
```

```
else if(digitalRead(button2)==LOW)
{
analogWrite(buzzer,frequency[1]);
on_time=millis();
if(i!=0)
button_offtime[i-1]=on_time-off_time;
while(digitalRead(button2)==LOW);
if(path==1)
 {
  off_time=millis();
  button_ontime[i]=(off_time-on_time);
  button_seq[i]=1;
   i++;
   Serial.println("button 2 stored");
 }
}

else if(digitalRead(button3)==LOW)
{
analogWrite(buzzer,frequency[2]);
on_time=millis();
if(i!=0)
button_offtime[i-1]=on_time-off_time;
while(digitalRead(button3)==LOW);
if(path==1)
 {
  off_time=millis();
  button_ontime[i]=(off_time-on_time);
  button_seq[i]=2;
   i++;
   Serial.println("button 3 stored");
 }
}
```

```
else if(digitalRead(button4)==LOW)
{
analogWrite(buzzer,frequency[3]);
on_time=millis();
if(i!=0)
button_offtime[i-1]=on_time-off_time;
while(digitalRead(button4)==LOW);
if(path==1)
 {
  off_time=millis();
  button_ontime[i]=(off_time-on_time);
  button_seq[i]=3;
   i++;
   Serial.println("button 4 stored");
 }
}

else if(digitalRead(button5)==LOW)
{
analogWrite(buzzer,frequency[4]);
on_time=millis();
if(i!=0)
button_offtime[i-1]=on_time-off_time;
while(digitalRead(button5)==LOW);
if(path==1)
 {
  off_time=millis();
  button_ontime[i]=(off_time-on_time);
  button_seq[i]=4;
   i++;
   Serial.println("button 5 stored");
 }
}
```



```
else if(digitalRead(button6)==LOW)
{
analogWrite(buzzer,frequency[5]);
on_time=millis();
if(i!=0)
button_offtime[i-1]=on_time-off_time;
while(digitalRead(button6)==LOW);
if(path==1)
  {
    off_time=millis();
    button_ontime[i]=(off_time-on_time);
    button_seq[i]=5;
     i++;
     Serial.println("button 6 stored");
  }
}

else if(digitalRead(button7)==LOW)
{
analogWrite(buzzer,frequency[6]);
on_time=millis();
if(i!=0)
button_offtime[i-1]=on_time-off_time;
while(digitalRead(button7)==LOW);
if(path==1)
  {
    off_time=millis();
    button_ontime[i]=(off_time-on_time);
    button_seq[i]=6;
     i++;
     Serial.println("button 7 stored");
 }
}
analogWrite(buzzer,0);
}


void playback (void)
{
 digitalWrite(led,LOW);
 for(int j=0;j<i;j++)
 {
  analogWrite(buzzer,frequency[button_seq[j]]);
  delay(button_ontime[j]);
  analogWrite(buzzer,0);
  delay(button_offtime[j]);
 }
 i=0;
 off_time=0;
 on_time=0;
 path=1;
 digitalWrite(led,HIGH);
}


void button_ISR()
{
  buttonFlag = 1;
```

Working

Make the connections as per the circuit diagram and upload the code to Arduino.
Once the power to the circuit is turned on, Arduino is ready to accept the input from the buttons. Each button is associated with a PWM signal in the code. When a button is pushed, that particular PWM signal is generated through the Piezo Electric Buzzer.
Now for the record and repeat mode, play a few tones using different buttons.
With each button pressed, Arduino starts recording i.e. makes note of the sequence of the buttons, its on time and off time.
Once you are done with the tone, you can push the Interrupt Button . As soon as the Arduino enters Interrupt Mode, all the previously pressed tones are played back through the Piezo Buzzer.
During normal tone playback i.e. when the buttons are being pressed, the LED on pin 13 stays ON. During repeat mode, it stays OFF.
An important point to understand here is that I did not use the tone() function of Arduino. You can try to generate sounds of different frequencies using that function.

Conclusion

This project can be helpful in understanding the sound capabilities of Arduino. Although I haven't used the tone () function, you can implement the same using that function for more accurate results.

Thank you fot your time and attention. We are now at the end of today's project.

Practice Question

Q. Before your program “code” can be sent to the board, it needs to be converted into instructions that the board understands. This process is called….

answer choices

- ☐ **Stop**
- ☐ **Compile**
- ☐ **Create**
- ☐ **Serial**


Episode 12 Crystal Ball


Hello kids welcome back to our Arduino class. In today's class we will be building a crystal ball to predict the future.


Am so excited.
That will be nice.

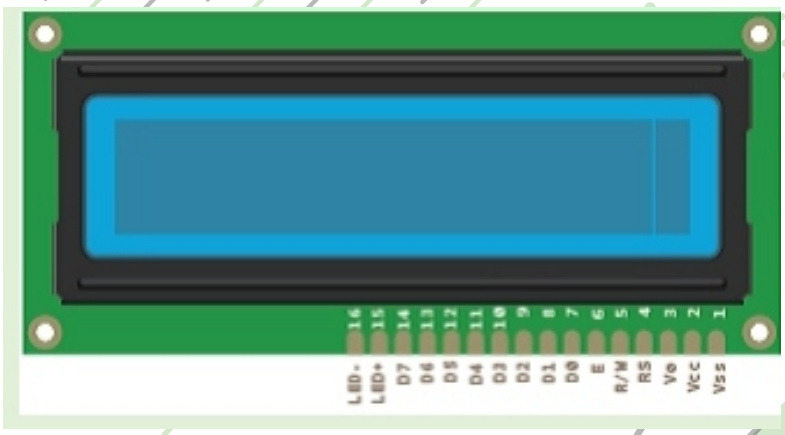

Uncle Emmanuel You mean we can predict what we want to be in future using a crystal ball.

What do you want to be in future?

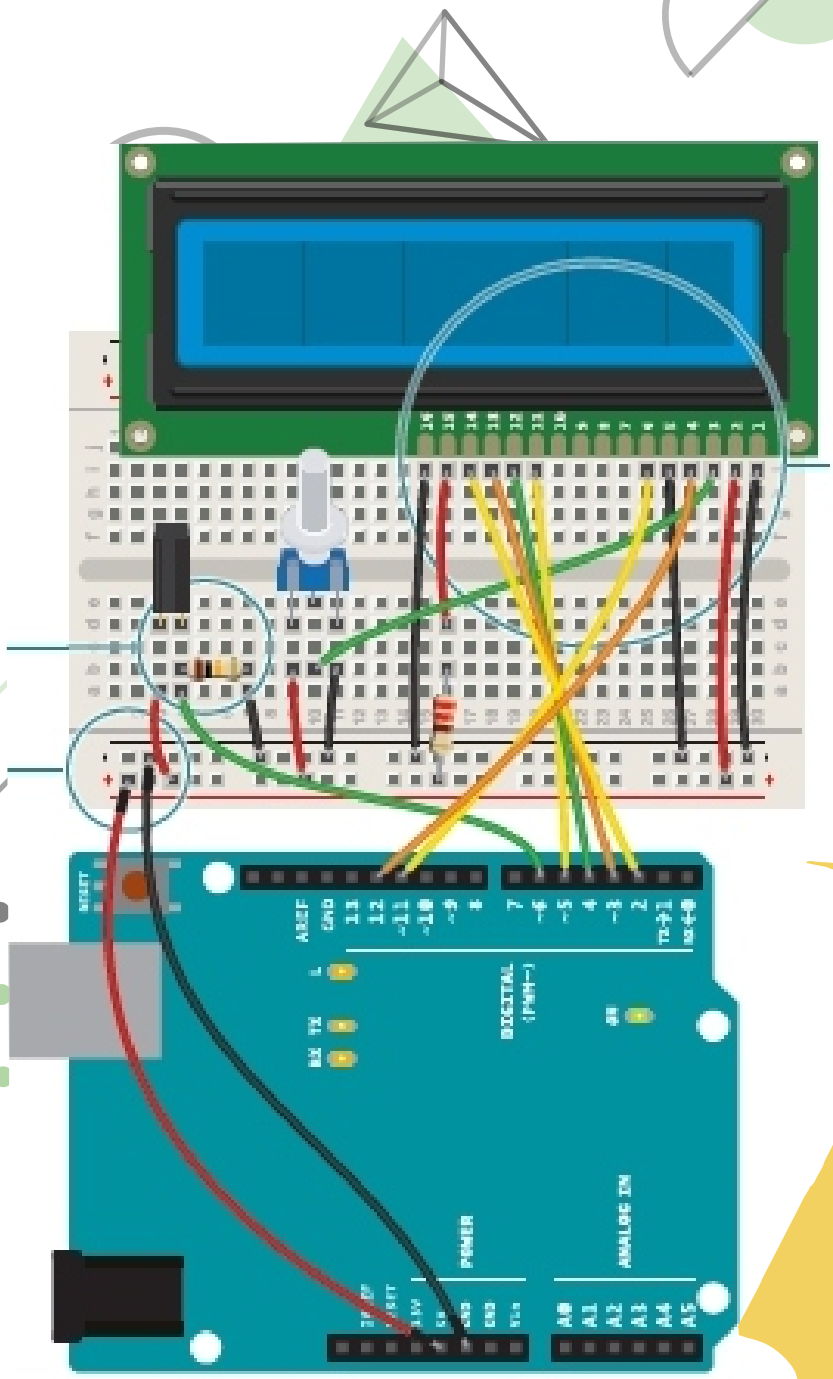
To complete this project you will need 220ohms resistor, one 10kiloohms resistor, the tilt switch, one potentiometer, one LCD screen and lots of jumper wires. This project also requires at least 60minutes of your time.

Crystal Balls can help predict the future. You ask a question to the all-knowing ball, and turn it over to reveal an answer. The answer will be predetermined but you can write in anything you like. You'll use your Arduino to choose from a total of 8 responses. The tilt switch in your kit will help replicate the motion of shaking the ball for answer.

The LCD (Liquid Crystal Display) can be display alphanumeric characters. The one in your kit has 16 columns and 2 rows, for a total of 32 characters. There are a large number of connections on the board. These pins are used for power and communication, so it knows what to write on screen, but you won't need to connect all of them. see Fig for the pins you need to connect.


LED-
LED+
D7
D6
D5
D4
D3
D2
D1
D0
E
R/W
RS
Vo
Vcc
Vss


Uncle can we start connecting it?




Once you are ready please make the connection as shown in the picture.

How to Connect

The circuit is not overly complex, but there are a lot of wires. Pay attention when wiring everything up to make sure it's correct.

1. Connect power and ground to one side of your breadboard.

2. Place the tilt switch on the breadboard and attach one lead to 5V. Attach the other side to digital input, just as you've done in several other projects.

3. The register select (RS) pin controls where the characters will appear on screen .The read/write (R/W) puts the screen in read or write mode. You'll be using the write mode in this project. The enable (EN) tells the LCD that it will be receiving a command. The data pins (D0-D7) are used to send character data to the screen. You'll only be using 4of these (D4-D7). Finally, there's a connectionfor adjusting the contrast of the display. You'll use a potentiometer to control this.

4. The LiquidCrystal library that comes with the Arduino software handles all the writing to these pins, and simplifies the process of writing software to display characters.

5. The two outside pins of the LCD (Vss and LED-) need to be connected to ground. Also, connect the R/W pin to ground. This places the screen in write mode. The LCD power supply (Vcc) should connect directly to 5V. The LED+ pin on the screen connects to power through a 220- ohm resistor.

6. Connect: Arduino Digital pin 2 to LCD D7, Arduino Digital pin 3 to LCD D6, Arduino Digital pin 4 to LCD D5, Arduino Digital pin 5 to LCD D4. These are the data pins that tell the screen what character to display.

7. Connect EN on the screen to pin 11 on your Arduino RS on the LCD connect to pin enable writing to the LCD.

8. Place the potentiometer on the breadboard, connecting one end pin to power and the other to ground. The centre pin should connect to V0 on the LCD. This will allow you to change the contrast of the screen.

Kids you have to follow the steps carefully when connecting your components to the board.

Understanding the code

First, you'll need to import the liquidCrystal library.

Next, you'll initialize the library, somewhat to the way you the servo library it what pins it will be using to communicate.

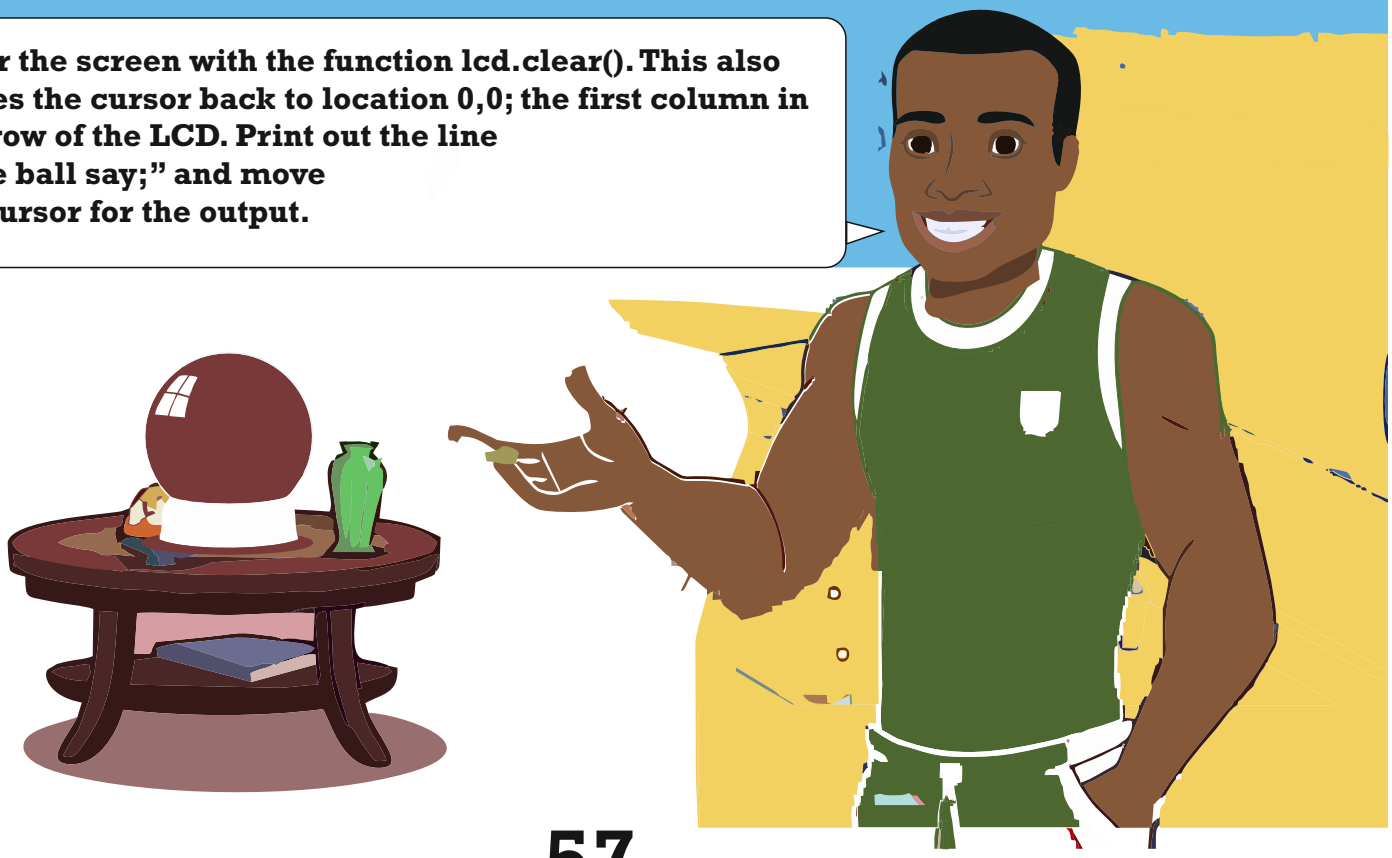
Now that you've set up the library. it's time to create some variable and constants. Create a constant to hold the pin of switch pin, a variable for the current state of the switch, a variable for the previous state of the switch, and one more to choose which reply the screen will show.
Set up the switch pin as input withMode() in your setup(). Start the LCD library, and tell it how large the screen is.
Now it's time to write a small introductory screen welcoming you to the 8-ball. The print() function writes to the words "Ask the" on the top line of the screen. The cursor is automatically at the beginning of the top line.

In order to write to next line, You'll have to tell the screen where to move the cursor. The coordinates of the first column on the second line are 0,1 (recall that computers are zero indexed. 0,0 is the first column of the first row). Use the function lcd. setCursor() to move the cursor to the proper place, and tell it to write "Crytal ball!" .

Now, when you start the program, it will say "Ask the Crystal ball!" on your screen.
In the loop(), you're going to check the switch first, and put the value in the value in the switchState variable.

Use an if() statement to determine if the switch is in a different position than it was previously. If it is different than it was before, and it is currently LOW, then it's time to choose a random reply. The random() function returns a number based on the argument you provide it. to start, you'll have a total number of 8 different responses for the ball. Whenever the statement random(8) is called, it will give a number between 0-7. Store that number in your reply variable.

Clear the screen with the function lcd.clear(). This also moves the cursor back to location 0,0; the first column in the row of the LCD. Print out the line "The ball say;" and move the cursor for the output.



The switch() statement executes different pieces of code depending on the value you give it. Each of these different pieces of code is called a case. switch() checks the value of the variable reply; whatever value reply holds will determine what named case statement is executed.

Inside the case statements, the code will be the same, but the messages will be different. For example, in case 0 the code says lcd.print ("Yes"). After the lcd.print() function, there's another command: break. It tells the Arduino where the end of the case is. When it hits break, it skips to the end of the switch statement. You'll be creating a total of 8 case statements to start out. Four of the responses will be positive, 2 will be negative, and the final 2 will ask you to try again.

The last thing to do in your loop() is to assign switchState's value to the variable prevSwitchState. This enables you to track changes in the switch the next time the loop runs.

The Code

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2); //generates an instance
in the lcd

cont int switchPin = 6;
int switchState = 0;
int prevSwitchState = 0;
int reply;

void setup() {
  lcd.begin(16,2),


pinMode(switchPin INPUT);
lcd.print("Ask the");
 lcd.setCursor(0,1);// changes the cursor to continue
writing in the second row
  lcd.print(Crstal Ball!");
}
void loo() {
  switchState=digitalRead(switchPin);

if (switchState != prevSwitchState) {
  if (switchState ==LOW){
     reply = random(8);
     lcd.clear(); // clears the writing
     lcd.setCursor(0,0);
     lcd.print("The ball says:");
     lcd.setcursor(0.1);
```

```
 switch(reply){ // the program will enter the case
assigned to the switch
      case 0:
      lcd.print("Yes");
      break;
      case 1:
      lcd.print("Most Likely");
      break 2;
      case 2:
      lcd.print("Certainly");
      break;
      case 3:
      lcd.print("Outlook Good");
      break;
      case 4:
      lcd.print("Unsure");
      break;
```

The code

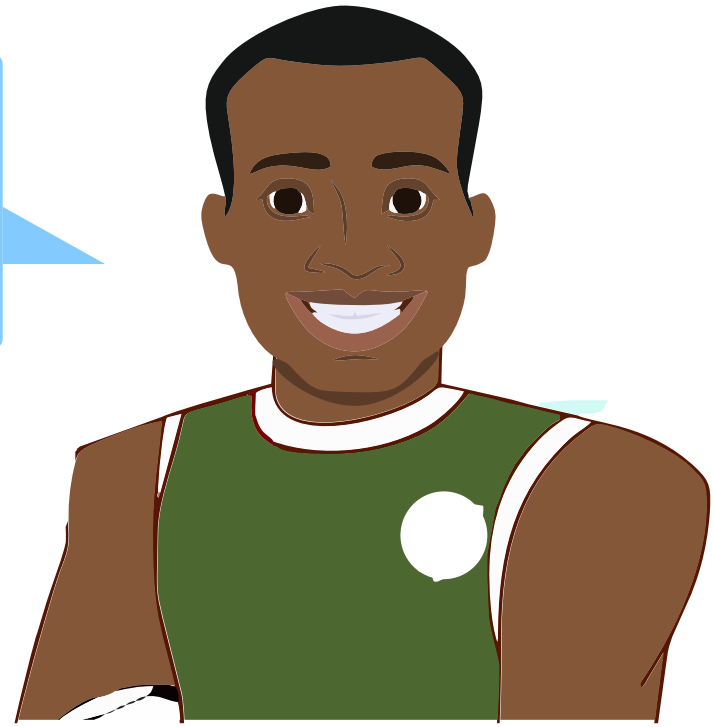
```
    case 5:
    lcd.print("Ask again");

    break;

    case6:

    lcd.print("Doubtful")

   break:

   case 7:

   lcd.print("No");

   break;

    }

  {

  prevSwitchState = switchState

  }
```

The Code

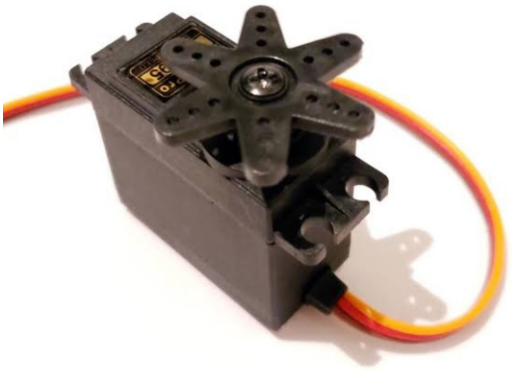
To use the magic ball, power the Arduino. Check the screen to make sure it says "Ask the Crystal ball!" If you can't see the characters, try turning the potentiometer. It will adjust the contrast of the screen.
Ask a question of your crystal ball, and try tilting the switch up-side down and back again. You should get an answer to your question. If the answer doesn't suit you, ask again.

Try adding your own sayings to the print() statements, but be mindful of the fact that there are only 16 characters to use per line. You can also try adding more responses. Make sure when you add additional switch cases, you adjust the number of options that will randomly populate the reply variable.
LCDs work by changing the electrical properties of a liquid sandwiched between polarized glass. The glass only allows certain kinds of light to pass through. When the liquid between the glass is charged, it starts to form into a semi-solid state. This new state runs in a different direction than the polarized glass, blocking light from passing through, thus creating the characters you see on the screen.

We are now at the end of today's class. Please do not hesitate to ask Uncle Green or me for help or check online for more information. See you in next class.



Practice Question



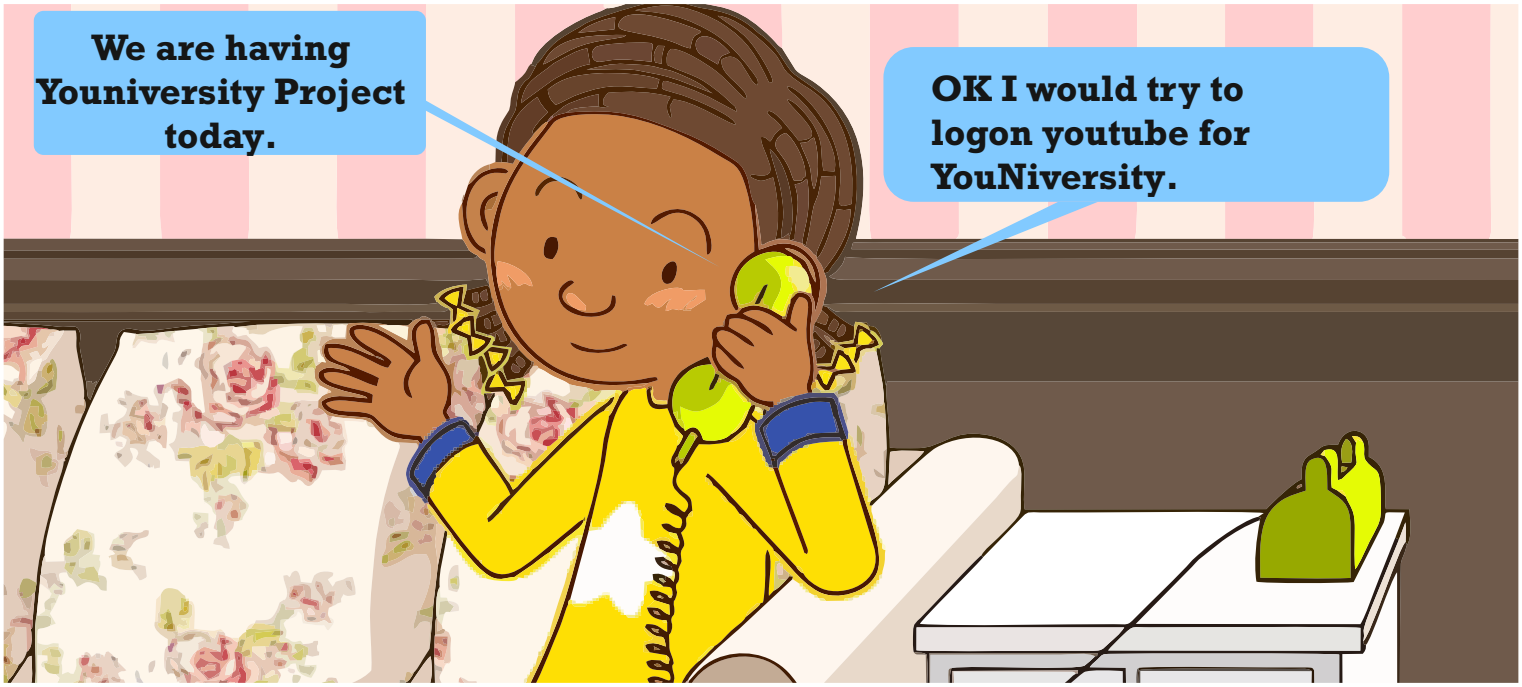
Q. The object in the diagram is called _____ and it is need when carrying out movement.

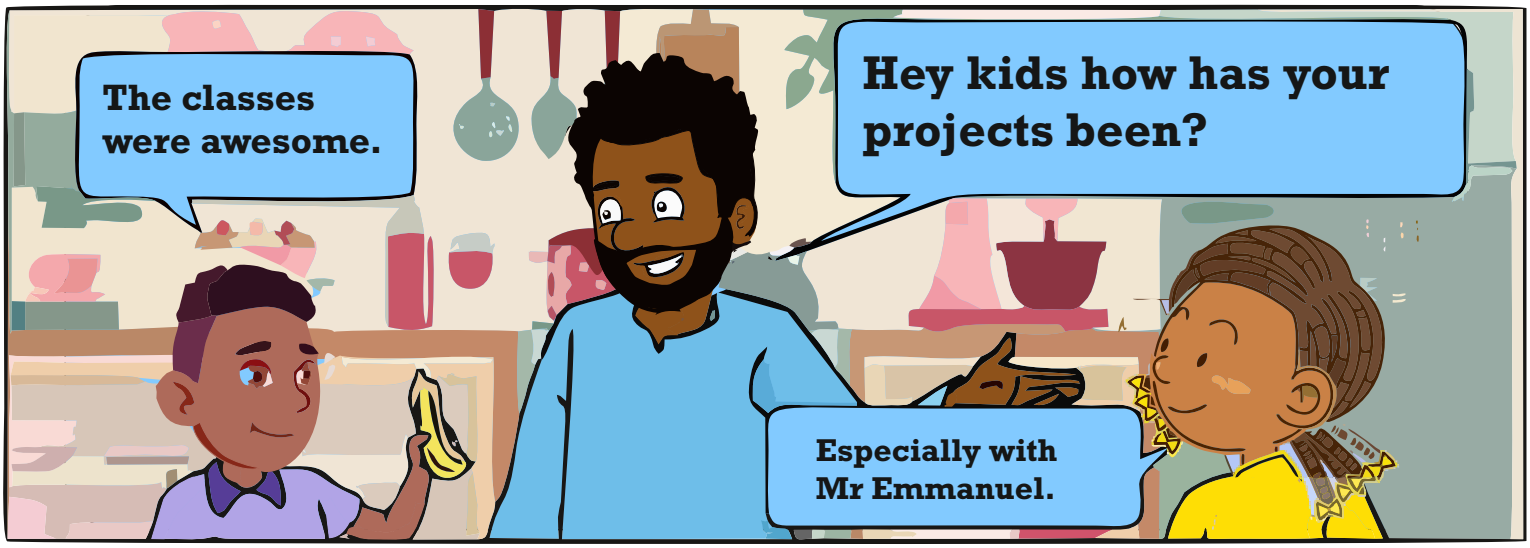
answer choices _____

- ☐ **Servo Motor**
- ☐ **Rotary Motor**
- ☐ **Fan**
- ☐ **Serial Record**


Let Me call Faith.


Faith how are you?
Ay I'm fine. Do we have any Project today?


We are having Youniversity Project today.
OK I would try to logon youtube for YouNiversity.



Arduino Stories

GreenLab MICRO-FACTORY

If you need further assistance, do not hesitate to ask your uncle green or Mr Emmanuel for help or check online for more information, I wish you a bright, prosperous and productive future.

A

API - Application programming Interface the interface used to interact programmically with a piece of software, The API of an Arduino library is the public functions exposed to the user. The library may contain many functions that are only used internally, but it is only necessary to understand the API in order to use the library.

AREF - Analog REFerence- the reference max voltage for the Analog to Digital converter Changing the AREF allows to use the resolution of the ADC at its best, especially for low voltage usually AREF can't be greater than the supply voltage of the ADC circuit.

ACCELEROMETER - A sensor that measures acceleration. Sometimes they are used to detect orientation, or tilt.

ACCESS POINT - A WiFi device that connects to the physical network and allows to access the LAN through the wireless connection. Usually it takes care of the authentication and the assignment of an IP address.

ACTUATOR - A type of component that changes an energy into motion. Motors are a type of electrical actuator.

ALTERNATING CURRENT - A type of current where electricity changes its direction periodically. This is the sort of electricity that comes out of a wall socket.

AMPERAGE (Amps or Amperes) - The amount of electrical charge flowing past a specific point in your circuit. Describes the current as it flows through a conductor, like a wire.

ANALOG - Something that can continuously vary overtime.

ANALOG-TO-DIGITAL CONVERTER (ADC) - A circuit number that converts an analog voltage into a digital number representing that voltage. This circuit is built-in to the micorcontroller, and is connected to the analog input pins of the Arduino board.

Glossary

ANODE - The positive end of a diode (remember that an LED is a type of diode).

ARGUMENT - A type of data supplied to a function as an input.For example, for digitalRead() to know what pin to check. It takes an argument in the form of a pin number

B

BACK-VOLTAGE - Voltage that pushes back against the current that created it. It can be created by motors spinning down. This can damage circuits,which is why diodes are often used in conjunction with motors (Free-wheel diode).

BAUD - Shorthand for "bits per second", signifying the speed at which two devices are communicating.

BINARY - Only two states are possible, like true/false or off/on.

BIT -The smallest piece of information a digital device can manage.

BOOLEAN - A datatype that indicates something binary.

BOOTLOADER - The Bootloader is a special piece of code that the microcontroller executes at power-up or under specific conditions and take care of the loading of the code into Flash memory. If the Bootloader doesn't receive an upload request by the host, it passes the execution to the user sketch.

C

CALIBRATION - The process of making adjustments to certain numbers or component to get results from a circuit or program. In Arduino project, this is often used when sensors in the real world may give different values in different circumstances, for instance the amount of light on a photoresistor. Calibration can be automatic or manual.

CAPACITANCE - The ability of a material to hold an electrical charge.

CATHODE- The negative end of a diode.

Glossary

Circuit - A circuti path from a power supply, through a load, and then back again to thee other end of the power supply. Current flows in a circuit only if it is closed that is, if the outgoing and return path are both uninterrupted (or open) then current will not flow through the circuit.

Common cathode LED - types of LEDs that have multiple colours in one fixture, with one cathode and multiple anodes.

CONDUCTOR - A material that allows electricity to flow, like a copper wire.

CONSTANT - A named identifier that cannot change its valuein a program.

CRYTOCHIP - Cryptochip it's an hardware chip that takes care of all the calculations required by the modern crytopgraphic standards. They work with any MCU, are extremely cost-effective, require only a single GPIO, and use every little power. Advanced protocols like ECDSA sign-verify (asymmetric authentication) and ECDH (key agreement in encryption/decrption settings) are built-in which makes adding sophisticated security easy.

CURRENT - The flow of electrical charge through a closed circuit. Measured in Amps.

D

DMA- Direct Memory Access is a technology that allows peripherals to access directly areas of the memory without CPU intervention. This increase the throughput some transfers on interfaces like SPI or DAC.

.

DATASHEET - A technical document that describes the functionality of a component. Typical information in a datasheet includes the maximum voltage and current a component requires, as well as an explanation of the functionality of the pins.

DATATYPE - A classification system that determines what values a particular constant, variable, or array will hold. Int, float, long and boolean are all types that can be used in the Arduino Software (IDE).

DEBUGGING - The process of going through a circuit or code, and finding errors (also referred as "bugs"), until the expected behaviour is achieved.

DECOUPLING CAPACITORs - Capacitors that are used to regulate spikes and dips in voltage and current, often placed close to the circuit they are referred.

DIGITAL - A system that deals with discrete values.

DIRECT CURRENT - A type of current which always flows in the same direction. Drain The pin of a Field Effect Transistor connected to the higher (n channel) or lower (p channel) voltage of i.e load to be controlled.

DUAL IN-LINE PACKAGE (DIP) - A type of packaging for integrated circuits Duty cycle. A ratio indicating the amount of time over a certain period that a component is turned on. When using a PWM value of 127 (out of a total of 256), you have a 50%

E

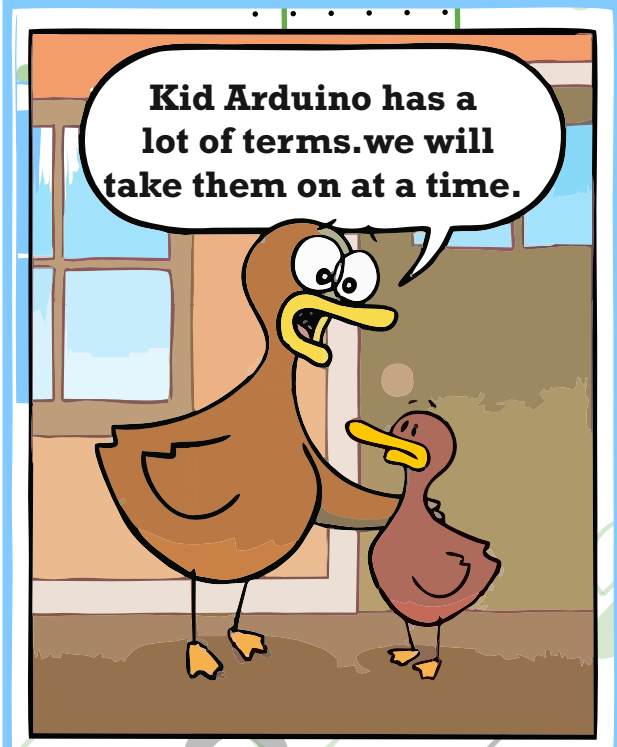
EDBG - The Atmel Embedded Debugger (EDBG) it's a chip that implements a composite USB device. Consisting of three interfaces.

EEPROM - An Electrically Erasable Programmable Reading Only Memory is a type of memory that retains its data without power, like a Read Only Memory, and that can be erased and written. The amount of memory of an EEPROM is usually of thousands of bytes and it is used to retain some data on microcontrollers between power cycles. Electricity A form of energy which we use to power machines and electrical devices.

F

FIRMWARE - The Firmware is similar to Software, but it is stored on non volatile memory and can be executed in place - without being copied in RAM - by a microcontroller. It is used in embedded systems and it is made of machine code instructions.Firmware is usually update able.

FLASH MEMORY - This type of memory is non volatile and it is an evolution of EEPROM. It is electrically erasable. Made with different technologies that offer different speeds and capacities (ML, SL, NAND, NOR), Flash Memory is used inside SD and microSD cards, inside mobile phones and also as program memory in microcontrollers.



FLOAT - A datatype used to represent a fraction. This entails the use of decimal points for floating point numbers. Function A block of code that executes a specific task.

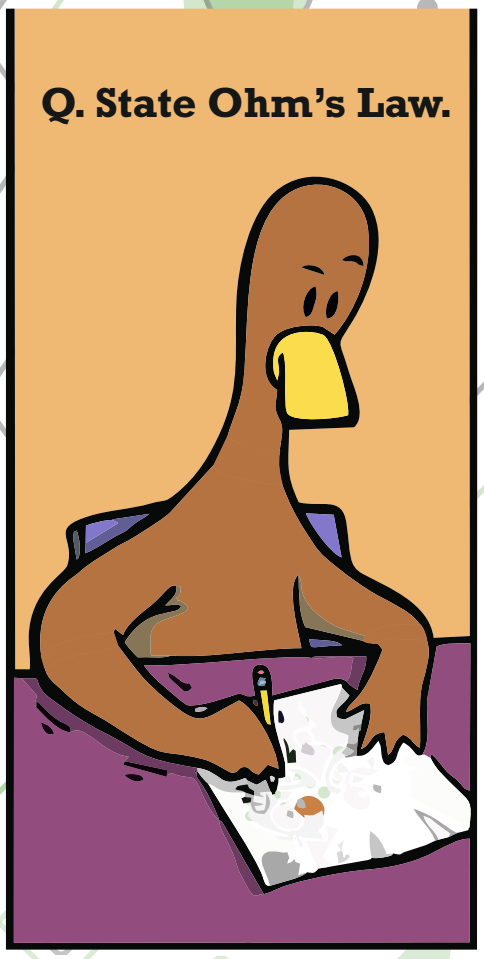
G

GATE - The pin of a Field Effect Transistor that allows to create a conducting channel between Drain and Source if correctly driven.

Glossary

GLOBAL VARIABLE - A named variable that can be accessed anywhere inside your program. It is declared before the setup() function.

GROUND - The point of a circuit where there is 0 potential electrical energy. Without a ground, electricity will not have a place to flow in a circuit.



I

I2C I2C - Inter Integrated Circuit - is two wires serial synchronous bus that provides communication between two Integrated Circuits., introduced in 1982 by Philips Semiconductor. The I2C can run at different speed: 100 Kbps, 400Kbps and 3.4 Mbps. It's used to connect also different boards with the simplicity of wiring only two cables, called SDA (Data line), SCL (Clock line). Can support up to 1008 slave devices.

ICSP In-Circuit Serial Programming is the ability of a microcontroller to be programmed directly on the board on which it is mounted. It is also a set of signals and pins used for the programming.

IDE Stands for "Integrated Development Environment". The Arduino IDE for example, is the place where you write software to upload to an Arduino board. Index The number supplied to an array that indicates which element you're referring to. Computers are zero-indexed, which means they start counting at 0 instead of 1. To access the third element in an array named tones, for example, you would write tones[2].

INDUCTION - The production of an electromotive force across a conductor when it is exposed to a time varying magnetic field.

INSTANCE - A copy of a software object. Insulator A material that prevents electricity to flow. Int A datatype that holds a whole number between -32,768 and 32,767. Integrated Circuit (IC) A complex circuit of nano or micro scale with a given package used for a specific purpose.gy which we use to power machines and electrical devices.

L

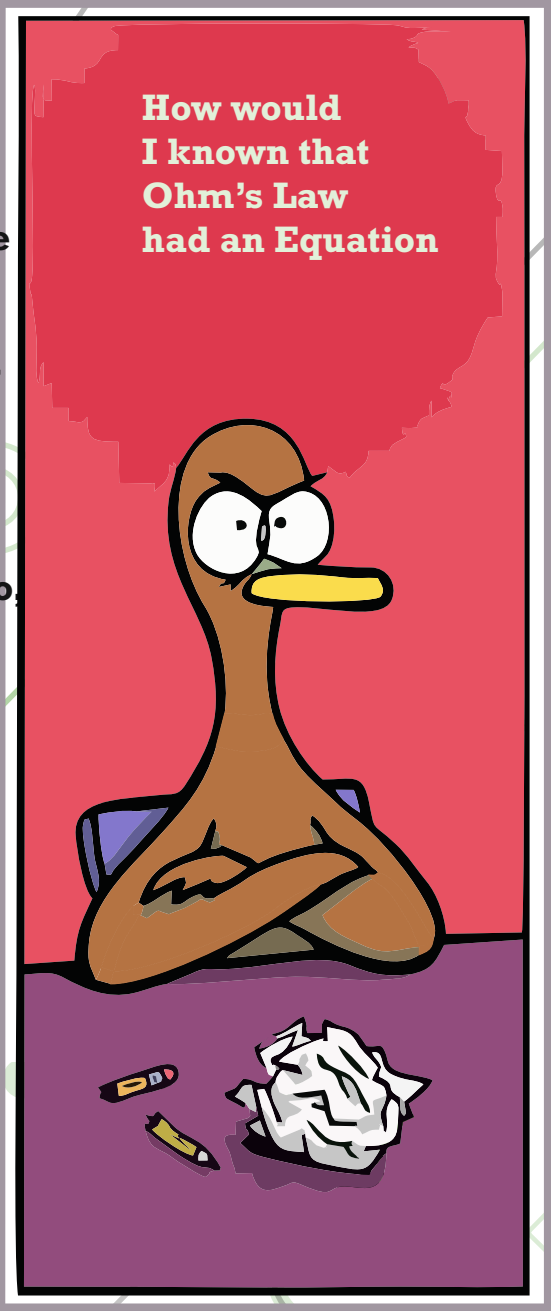
Li-Po Battery - One of the most recent technologies for rechargeable batteries. This Lithium - Polimer technology offers a very good ratio between capacity and weight, no memory effect and low self discharge rate. Li-Po batteries are charged at constant current.

Glossary

LIBRARY - Its a software extension of the Arduino API that expands the functionality of a program. Load An electrical component or portion of a circuit that consumes electric power to turn it into something else.

LOCAL VARIABLE - A type of variable that lives only in the scope in which it is declared and then forgotten. A variable declared inside the setup() of a program would be local: after the setup() finished running, the Galileo would forget that the variable ever existed.

LONG - A datatype that can hold a number, from -2,147,483,648 to 2,147,483,647.



M

MICROCONTROLLER - The brains of the Arduino, this is a small computer that you will program to listen for, process, and display information. Millisecond 1/1,00 of a second.

N

NTP- A time syncronization protocol, based on a client-server architecture, available on Internet on UDP port 123. It uses the UTC format and it is unrelated to time zones. When used it allows to synchronise clocks within 10 milliseconds of margin and around 200 microseconds of accuracy inside a LAN.

O

OBJECT -An instance of a library; e.g. when using the Servo library, were you to create an instance named myServo, myServo would be the object. Ohm Unit of measurement of resistance. Represented by the omega symbol.

OHM'S LAW - A mathematical equation that demonstrates the relationship between resistance, current and voltage. Usually stated as V (voltage) = I (current) x R (resistance).

OPTOCOUPLER - Also known as an opto-isolator, photo-coupler, photo-isolator, photo-switch, and opto-switch. An LED is combined in a sealed case with a phototransistor. The LED is positioned to illuminate the phototransistor, so that when the LED is turned on, the phototransistor will conduct. Used to provide a high degree of isolation as there is no electrical connection common to the input and the output.

Glossary

P

PARALLEL - Components connected across the same two points in a circuit are in parallel. Parallel components always have the same voltage drop across them.

PARAMETER - When declaring a function, a named parameter serves as the bridge between the local variables in the function, and the arguments it receives when the function is called.

PERIOD - A specific span of time in which something happens. When the period changes, you're adjusting the frequency at which something will occur.

PHOTOCELL - A device for converting light to voltage or current.

PHOTORESISTOR - A resistive device whose resistance varies according to how much light hits it.

PHOTOTRANSISTOR - A transistor whose conduction state is controlled by light.

POLARIZED - The leads of polarized components (e.g. LEDs or capacitors) have different functions, and thus must be connected the right way. Polarized components connected the wrong way might not work, might be damaged, or might damage other parts of your circuit. Non-polarized components (e.g. resistors) can be connected either way.

POWER - supply A source of energy, usually a battery, transformer, or even the USB port of your computer. Comes in many varieties such as regulated or unregulated, AC or DC. Usually the voltage is specified, along with the maximum current the supply can deliver before failing.

PROCESSING - A programming environment based on the Java language. Used as a tool to introduce people to the concepts of programming, and in production environments. The Arduino IDE is written in Processing, and so will look very familiar. In addition, Processing, Arduino and Galileo share a similar philosophy and motive, of creating free open source tools allowing non-technical people to work with hardware and software.

PSUEDOCODE - A bridge between writing in a computer programming language and using natural speech. When creating pseudocode, it's helpful to write in short declarative statements.

PULSE - Width Modulation (PWM) A way to simulate a varying static voltage.

R

RTC - The Real Time Clock is a circuit that uses either a quartz crystal or a laser trimmed oscillator to keep the time inside a computer or in an embedded system. The circuit has its own power source - usually a lithium battery - and has a very low power consumption. Once set, the RTC can be queried to get date and time through serial interfaces like I2C.

RESISTANCE - A measure of how efficiently a material will conduct electricity. In particular, resistance can be calculated by Ohm's Law as: R = V/I.

S

SAMD21 - A microcontroller of the Atmel's SAM D family. A rich set of peripherals, flexibility and ease-of-use combined with low power consumption make this microcontroller, based on Cortex-M0+, ideal for a wide range of home automation, consumer, metering and industrial applications.

SERCOM - This term is the combination of SERial and COMmunication. It is used in microcontrollers jargon to indicate a specific type of functional module inside the device. A SERCOM module can be programmed to become a USART, a SPI or an I2C serial port. Each microcontroller may have more tha one SERCOM.

SHA-256 - An algorythm developed to check the integrity of data. SHA-256 is Secure Hash Algorithm with 256 bit digest. It is part of the SHA-2 family of hashing algorithms. Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the output from execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity.

SPI - Serial Peripheral Interface is a synchronous serial communication technology that uses four wires and a master/slave architecture. A master device can communicate in full duplex with several slaves using a specific Slave Select signal for each slave. The communication is suitable for short distance, like the inside of the cabinet of an electronic device.

SRAM - A specific type of memory, Static RAM, that doesn't require the refresh mechanism, typical of the more common DRAM - Dynamic RAM - used in PCs. SRAM keeps the stored data as long as the power supply is kept, offers good access time and low power consumption.

Glossary

SSID - The Service Set IDentifier is the name with which a wireless network identifies itself. Several Access Point may share the same SSID if they are connected to the same LAN. An Access Point may broadcast more than one SSID or none at all. Sensor A component that measures one form of energy (like light or heat or mechanical energy) and converts it to voltage or current.

SERIAL BUFFER - A place in your computer's and microcontroller's memory where information received in serial communication is stored until it is read by a program. Serial communication A type of serial protocol between two devices.

SERIAL MOTOR - A tool built in to the Arduino IDE allowing sending and receiving serial data to and from a connected board. Series Components are in series when current flows from the first into the next. The current flowing through both is the same, and the voltage drops across each component. Short circuit A short circuit between power and ground will make your circuit stop working and thus should be avoided. In some cases this might damage your power supply or parts of your circuit, and rare cases might start a fire.

SKETCH - The term given to programs written in the Arduino IDE.

SoC - System on Chip is an evolution of embedded systems, where in a single package and substrate are hosted all the parts necessary to create a fully functional computer. The package contains a microprocessor, the various types of memory, interfaces for communication and peripheral management, converters and so on. With a SoC is possible to build a very cost effective computer that runs an operating system like Linux and connects to the usual set of I/O peripherals (mouse, keyboard, monitor…).

SOLDERING - The process of making an electrical connection by melting solder over electrical components or wires that are to be connected. This provides a solid connection between components. Source (transistor) The pin of a Field Effect Transistor connected to the lower (n channel) or higher (p channel) voltage of i.e load to be controlled.

SQUARE WAVE - A type of waveform that is identified by having only two states, on and off. When used to generate tones, they can sound "buzzy".

SWITCH - A component that can open or close an electrical circuit.

T

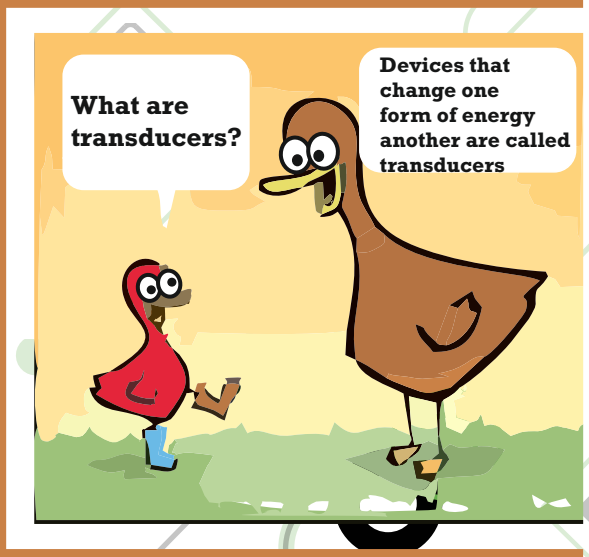
TTL A type of electronic logic circuitry that works with 5V/0V logic levels.

TRANSISTOR - Transistor Logic was developed during the sixties of the last century and became a worldwide standard. Low power and low voltage applications are now replacing 5V TTL with new technologies.

TOOLCHAIN (IDE) - A toolchain is a set of programming tools that is used to perform a complex set of operations. In the Arduino Software (IDE) the toolchain is hidden from the user, but it is used to compile and upload the user Sketch. It includes compiler, assembler, linker and Standard C & math libraries.

TRANSDUCERS - Something that changes one form of energy into another, Sensors or actuators can be called tranducers. Transducers are device that can perform an input (Sensors) or an output (Actuators) function. For example a force sensor converts the physical change into an electrical signal, an actuator transforms an electrical signal into a physical change like movement or sound. Another example of transducers are the microphone that converts sound waves into electrical signal, or a loudspeaker that converts this electrical signal back into sound waves.

TRANSISTOR - A 3 terminal (usually) electronic device which can act as either an amplifier or a switch. A control voltage or current between two leads controls a (usually) higher voltage or current between a different pair of leads. Common types of transistors include the Bipolar Junction Transistor (BJT) and the Metal Oxide Semiconductor Field Effect Transistor (MOSFET).



U

UART - An Universal Asynchronous Receiver Transmitter is an hardware device or module inside a microcontroller that implements serial communication with speeds and data format fully configurable. USB Stands for Universal Serial Bus. It's a generic port that is standard on most computers today. With a USB cable, it's possible to program and power an Arduino over a USB connection.

UNSIGNED - A term used to describe a datatypes, indicating that they cannot be a negative number. It's helpful to have an unsigned number if you only need to count in one direction. For instance, when keeping track of time with millis(), it's advisable to use the unsigned long datatype.

V

VCC - This label marks the pin used to supply the positive voltage to a circuit, a device or a board. The other pin, for ground, is usually labelled VSS or GND.

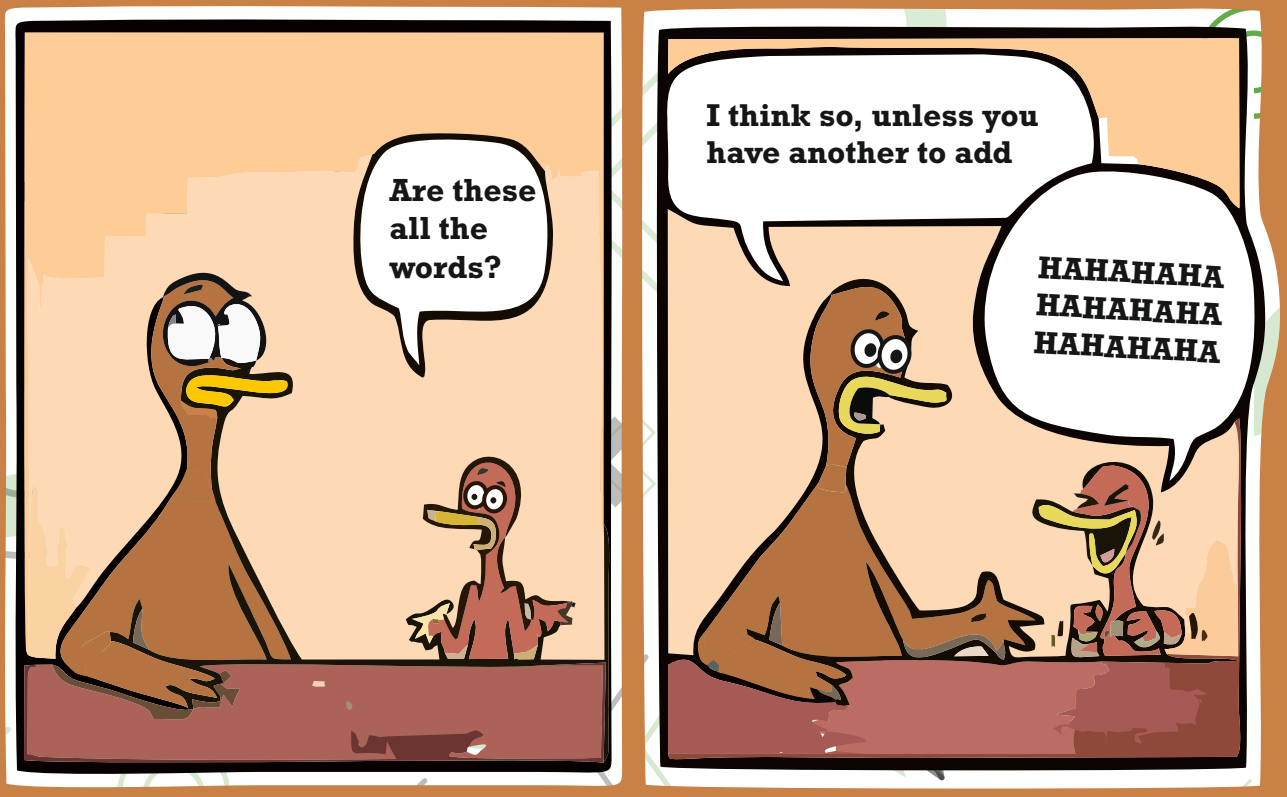
VIN - A label used to indicate the input pin for the voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

VARIABLE - A datatype that stores values which are likely to change as your program runs. A variable's type depends on the type of information you want to store, and the maximum size of the information; for example, a byte can store up to 256 different values, but an int can store up t 65,536 different values. Variables can be local to a particular block of code, or global to an entire program.

VOLTAGE DIVIDER - A type of circuit that provides an output that is a fraction of its input voltage. You are building a voltage divider when you combine a photoresistor with a fixed resistor to provide an analog input. A potentiometer can also be used as a voltage divider.

W

WPA The WiFi Protected Access is a cyphering protocol to keep the wireless transfer of data safe from easy eavesdropping. WPA and WPA2 are also certification programs that grant the compliance of the devices with the standards.



Cartooino Project Book 2020

GreenLab Cartooino