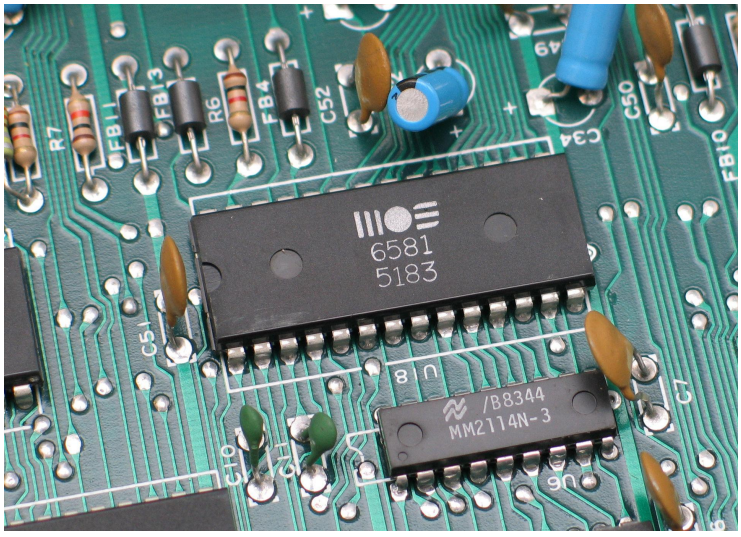
Electronic Circuit Assembly

The power of technology is that there are many different tools that can be used together to create awesome results. In the context of electronic circuit assembly, the concepts of computer science, programming, electrical engineering, design, hardware, and software all come together to create incredible projects that can do powerful tasks. There are many layers that can be added to these outputs via many sensors and other external devices to increase efficiency, accuracy, and scope of the output. These sensors and other electronic parts take in different inputs and give us valuable output we can use in our computer programs, which will change the output of the physical hardware.

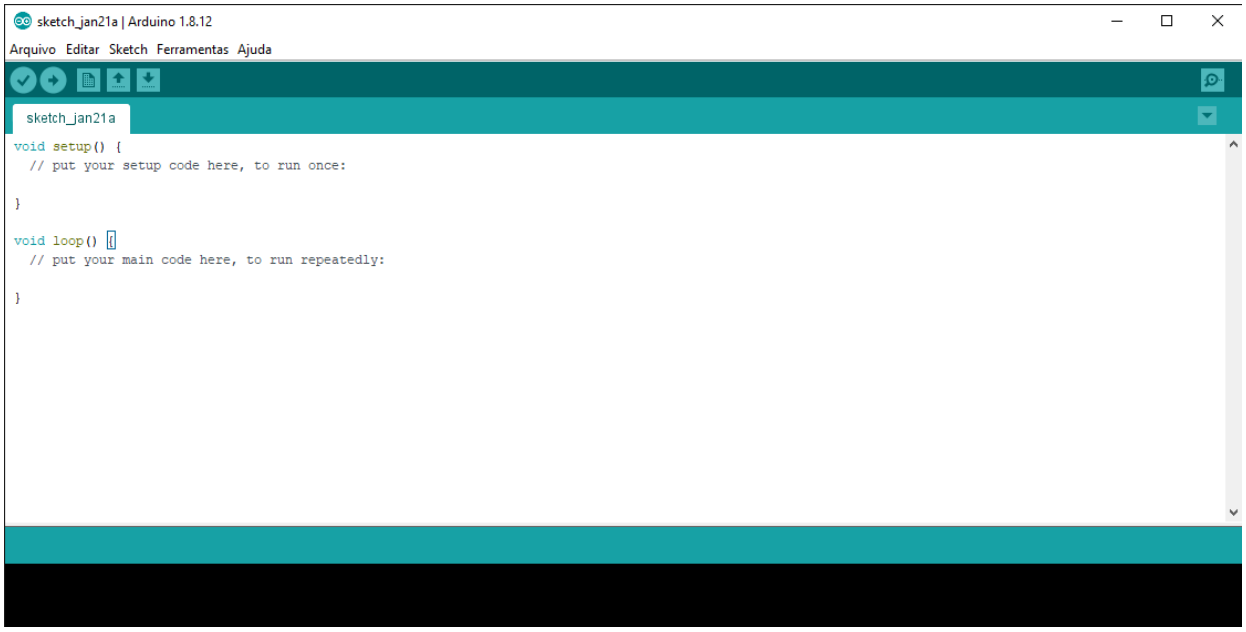


Reference: https://en.wikipedia.org/wiki/Printed_circuit_board

A photoresistor is a resistor whose resistance varies according to the intensity of the light that falls on it. It is a passive component, so we need to feed it to provide the measures of the quantity of interest. This sensor may have other names, for example LDR (light dependent resistor).

Arduino IDE, Serial Plotter, and Serial monitor: The communication between PC and Arduino can be done through the serial monitor. Thus, we can send data and commands from the PC to the Arduino (ex: LED intensity, number of motor turns, etc.) as well as receive data from the Arduino through the PC (ex: light intensity, temperature, humidity).

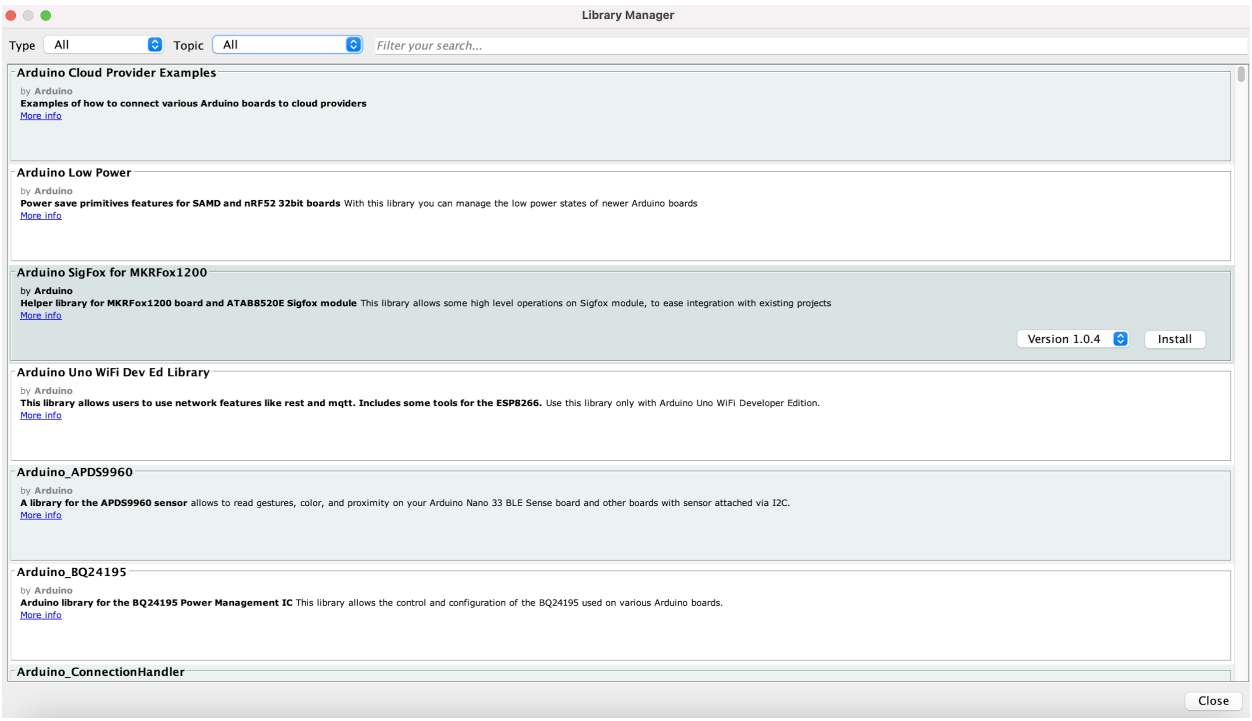
The Arduino IDE is the place where we write computer programs that we can upload to our physical Arduino device and the many sensors attached to it. This allows the sensors to output and work as intended by our computer program.



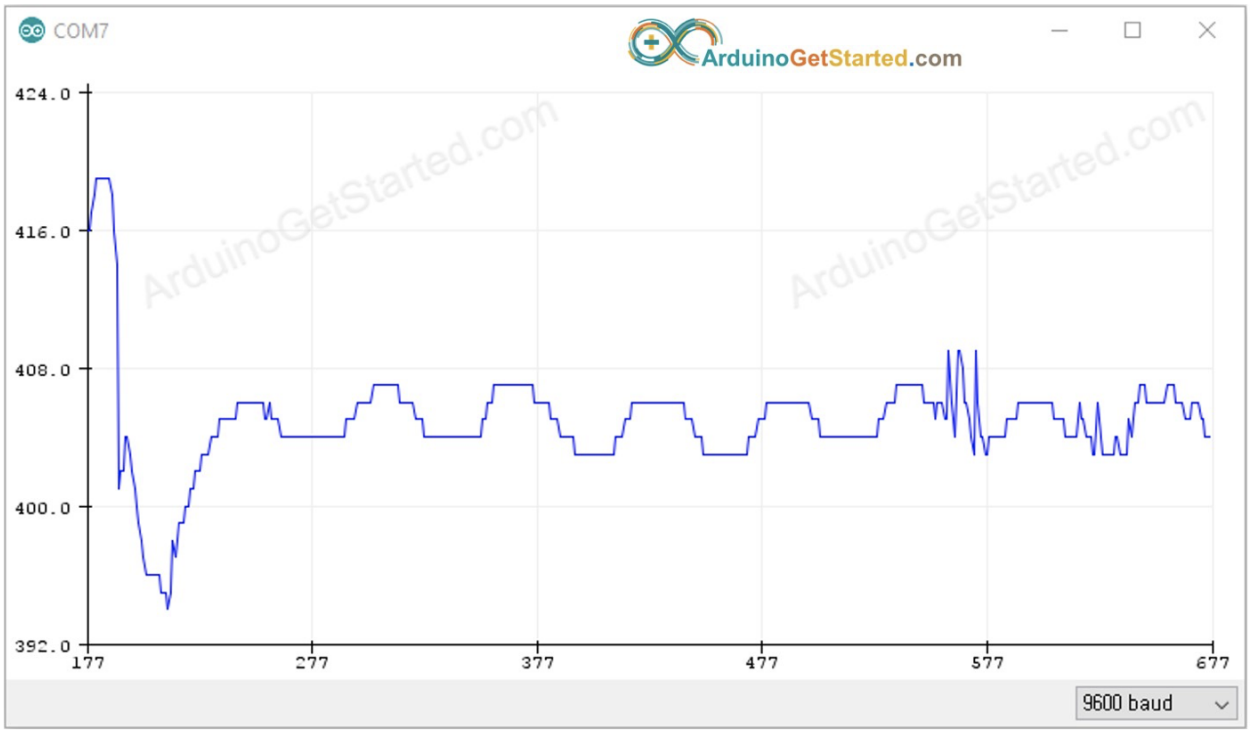
The Arduino IDE has many powerful tools. For example, there are built-in open source examples of many Arduino programs for different protoboards involving analog, digital, communication, sensors, display, Wifi, and much more. Open source is a powerful idea in the

technology world where everyone can contribute to these projects for the betterment of everyone.

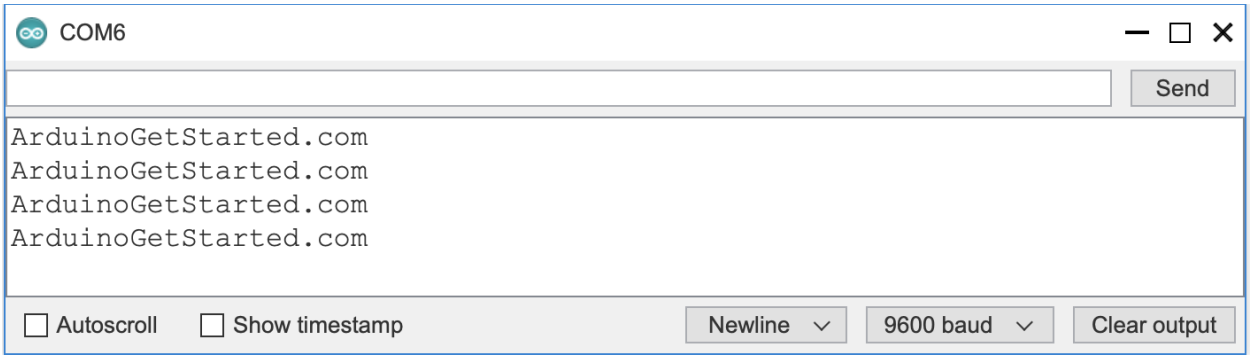
In addition, the Arduino IDE also can help provide many libraries for you to use. These involve many different functionalities and uses, including debugging, machine learning, and many other complicated concepts. These libraries are authentic code written by many people that allow other people to use it as a black box. This is powerful since libraries are used to increase complexity of our programs and projects in an easy manner.



The serial plotter is a tool in the Arduino IDE that is beneficial for displaying data, such as temperature, humidity, and voltage, from different sensors in a visual manner of a graph with waveforms. The power of the serial plotter is that it allows you to see different relationships between data in a fast way.



The serial monitor is another powerful tool in the Arduino IDE that is very helpful for monitoring the progress of our programs and debugging our programs. This tool displays data and other content that we tell it to display from our program code on the screen. The serial monitor is similar to the function in many programming languages that allow the programmer to print out test results and other content for debugging. This is used heavily in order to produce efficient and accurate programs.

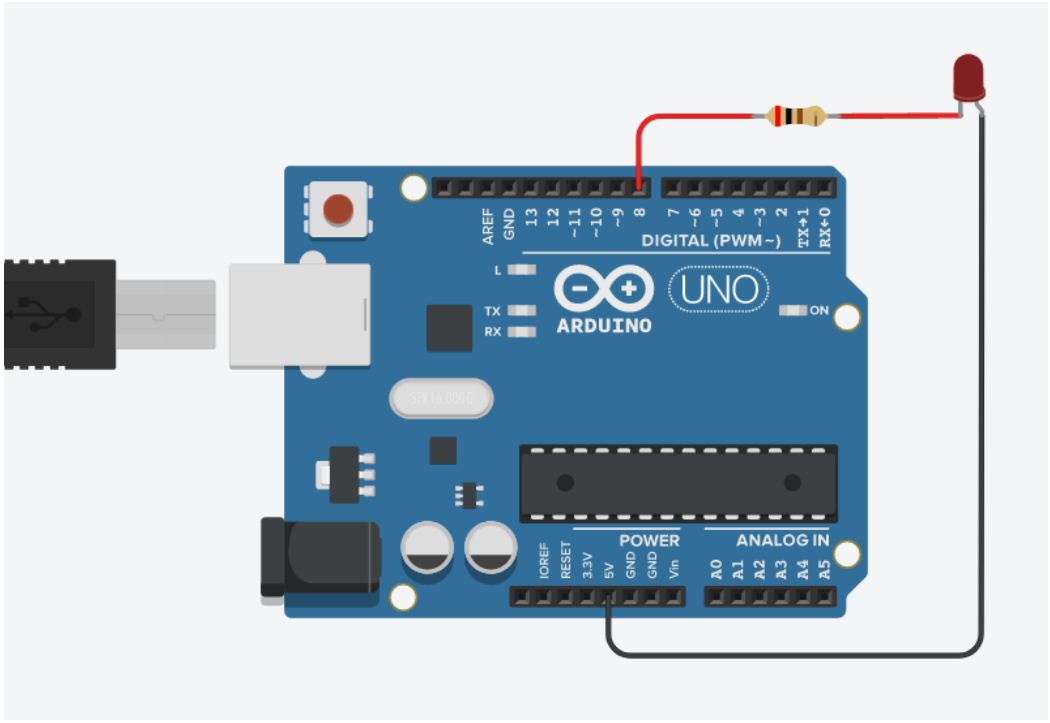


Exercise 1

We will use the serial monitor to receive information from the Arduino and print on the PC screen (built-in LED).

The serial monitor can be accessed from the Arduino IDE. The information received from the sensor can be displayed on the serial monitor through several commands. First, we have to set up the serial monitor by adding the proper communication channel in the setup function of the Arduino program. This can be done by writing Serial.begin(#) where the # represents the speed of communication, such as 9600 bits per second. Bits are the basic representation of information in computer science. A bit often can be defined by 1s and 0s.

Once we have set up the correct communication to the serial monitor, we can then use Serial.print(parameter) to print to the screen any parameter we want. This is powerful to see the outputs of the computer program and verify its accuracy.



Texto

1 (Arduino Uno R3)

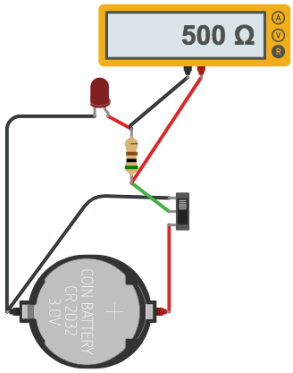
```
//O LED está conectado ao Arduino no pino 8

void setup(){ //Ligar o LED
  pinMode(8, OUTPUT);
}
void loop(){
  digitalWrite(8, HIGH); // Ligar o LED (Colocar pino ALTO)
  delay(1000); // Esperar por 1000 millisegundos ~ 1 segundo
  digitalWrite(8, LOW);  // Apagar o LED (Colocar pino BAIXO)
  delay(1000);// Esperar por 1000 millisegundos ~ 1 segundo
}
```

Exercise 2

We will use the serial monitor to receive the information from the photoresist sensor and print the read values.

1. In Tinkercad, assemble the circuit with Arduino, protoboard, photoresistor and a resistor (2K Ohm). Remember that a photoresistor is a resistor whose resistance varies according to the intensity of the light that falls on it. Also remember that a major purpose of a resistor is to reduce the flow of current throughout the circuit.

2. Connect a multimeter between the resistance terminals to check the resistance variation (in the circuit). Remember that a multimeter is powerful because it can read and display many different data points for the user.



Reference: https://www.tinkercad.com/things/k8eBP5VeKZt-multimeter

3. Print on the serial monitor the voltage values “read” by the Arduino on the assembled circuit. Remember that we need to initiate the proper communication channel in the setup function of the Arduino program using Serial.begin and then use Serial.print to print the desired values.

4. Assemble the physical circuit and print the values read by the Arduino on the serial monitor.

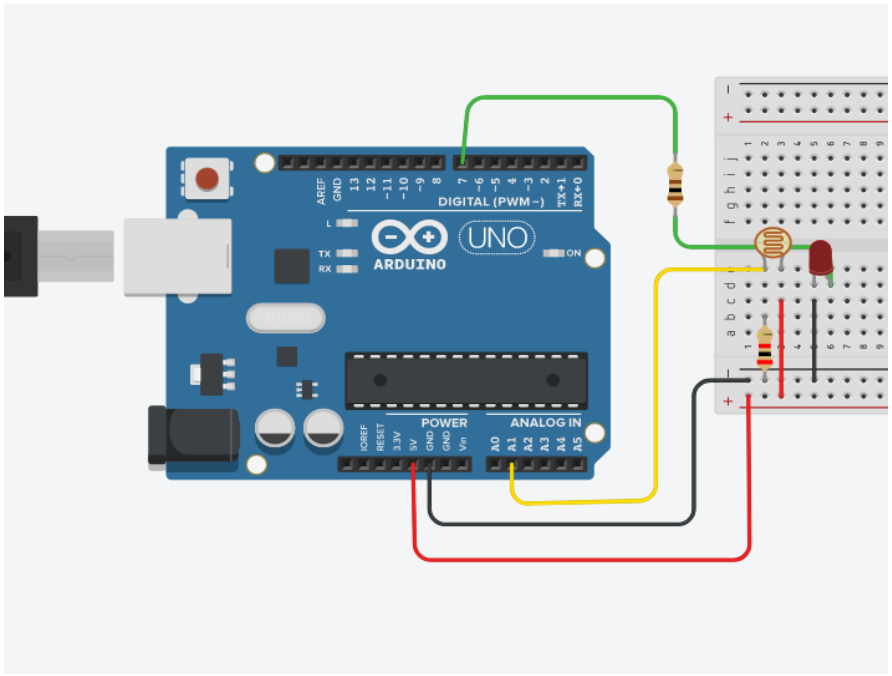
Exercise 3

We will use the values read on the sensor (photoresistor) to decide when to turn on the LED light.

1. Assemble an Arduino circuit, protoboard, resistor and LED. Remember that an LED is a light-emitting diode that emits light when electron particles flow through it.

2. On the protoboard, mount the resistance circuit, photoresistor and connection with Arduino.

3. Make a block program, where we turn on the LED from a certain brightness (value that can be defined from exercise 2).



Blocos + texto

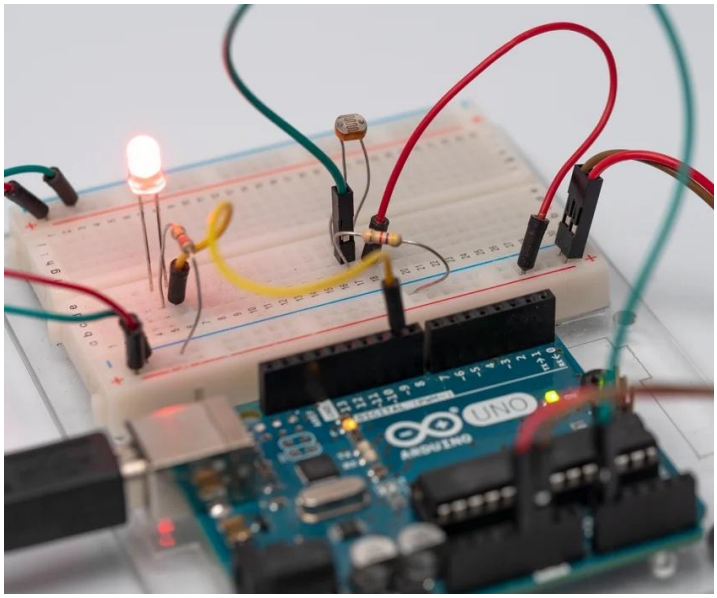
1 (Arduino Uno R3)

Saída
Entrada
Notação
Controlar
Matemática
Variáveis

ler pino digital 0
ler pino analógico A0
ler graus de servo no pino 0
número de caracteres seriais disponíveis
ler do serial
ler sensor de distância ultrassônico no pi
ler sensor de temperatura no pino A0

definir Luz como ler pino analógico A1
imprimir no monitor serial Luz com nova linha
se Luz < 500 , então
definir pino 7 como BAIXO
outro
definir pino 7 como ALTO

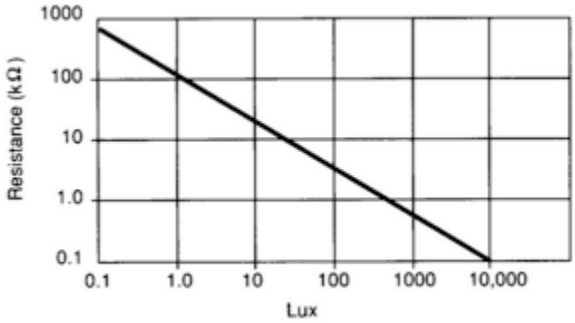
```
int Luz = 0;

void setup()
{
  pinMode(A1, INPUT);
  Serial.begin(9600);

  pinMode(7, OUTPUT);
}

void loop()
{
  Luz = analogRead(A1);
  Serial.println(Luz);
  if (Luz < 500) {
    digitalWrite(7, LOW);
  } else {
    digitalWrite(7, HIGH);
  }
  delay(10); // Delay a little bit to improve simulation
}
```



Reference: https://www.instructables.com/Light-Sensor-Photoresistor-Arduino-Tinkercad/

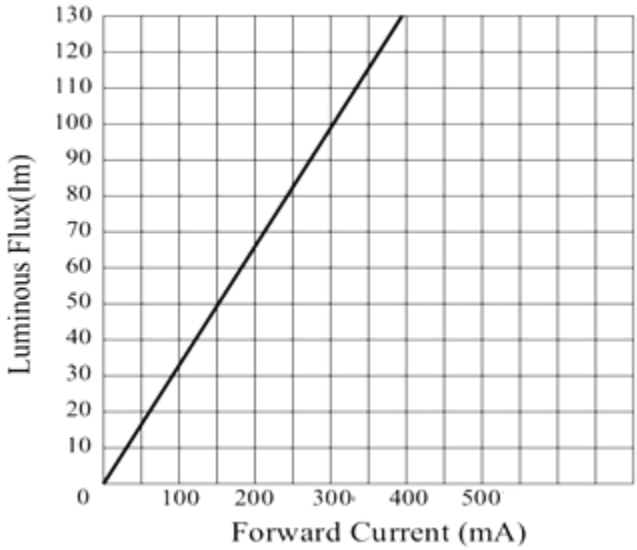
Note: operation of sensors

Resistance variation as a function of luminosity:



This relationship is called an indirect relationship. This means that as luminosity increases, the resistance decreases. Note that lux is the unit of illuminance. In many graphical relationships, we refer to the variable on the x axis(horizontal) to be the independent variable and the variable on the y axis(vertical) to be the dependent variable.

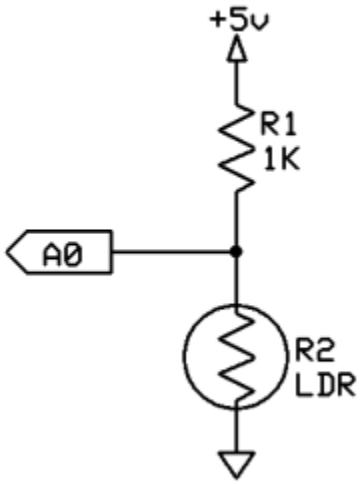
Luminosity variation as a function of current:



Reference: https://components101.com/diodes/1-watt-led

The opposite of an indirect relationship is a direct relationship. This means that as the independent variable increases, the dependent variable also increases.

We use this behavior of the photo resistive element to insert in the following circuit:

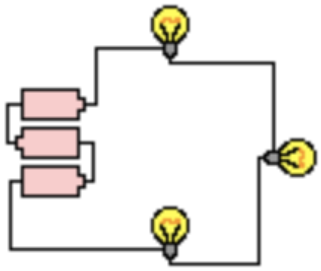


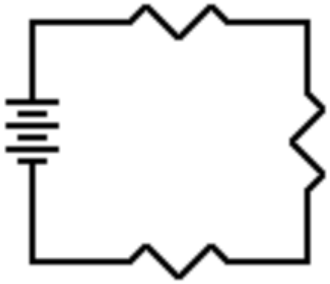
VA0 = (5V / (1K + LDR)) * LDR

- The lower the brightness, the greater the resistance of the LDR;

- The greater the resistance of the LDR, the greater the voltage at A0;
- So: the lower the brightness the higher the voltage in A0

In electrical engineering, many of these symbols are standard and used many times to note different sensors and parts of the circuit.



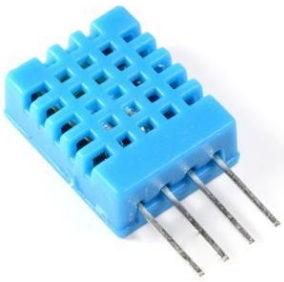


Reference:
https://www.physicsclassroom.com/class/circuits/Lesson-4/Circuit-Symbols-and-Circuit-Diagrams

Other sensors

- Temperature and humidity sensor:

  The DHT11 Humidity and Temperature Sensor is a temperature and humidity sensor that allows you to take temperature readings between 0 to 50 Celsius and humidity between 20 to 90%, widely used for projects with Arduino



- Ultrasonic sensor:

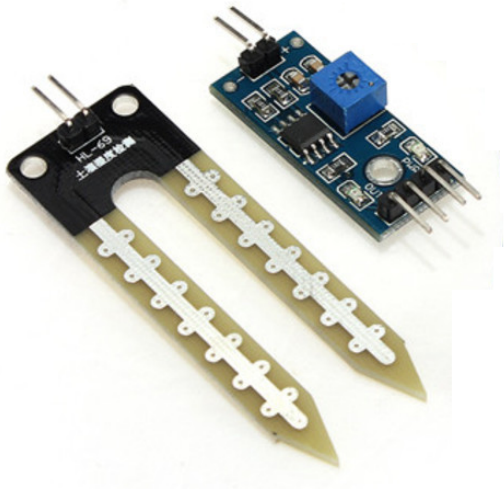
  The Ultrasonic Sensor HC-SR04 is a very common component in Arduino projects, and allows you to take readings of distances between 2 cm and 4 meters, with an accuracy of 3 mm. It can be used simply to measure the distance between the sensor and an object, such as to activate microcontroller doors, to divert a robot from obstacles, to trigger alarms, etc. In this tutorial we will teach you how to connect the HC-SR04 to the Arduino.

- Temperature sensor:

  Perform temperature measurements accurately not only in a dry environment but also in damp and wet environments with the waterproof DS18B20 temperature sensor. The DS18B20 temperature sensor can read accurately up to ± 0.5 ºC, and send the information to the microcontroller using only 1 wire.

- Soil moisture sensor:

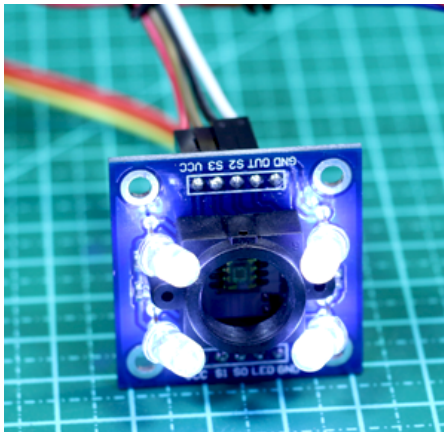
  The soil moisture sensor has a visual indicator of the soil moisture level and the health of your plant. It consists of 2 parts: a probe that comes in contact with the ground, and a small module containing a comparator chip LM393 (datasheet), which will read the data coming from the sensor and send it to the microcontroller, in our case, an Arduino Uno. As an output, we have a D0 pin, which is at level 0 or 1 depending on the humidity, and an analog output pin (A0), which makes it possible to monitor more accurately using an analog port of the microcontroller.



- Color sensor:

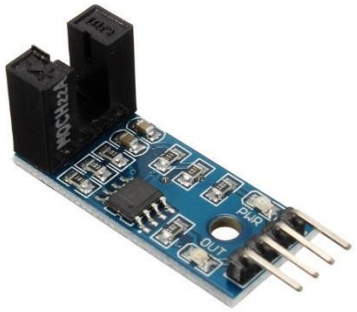
  Use the TCS3200 color sensor and an RGB led to easily assemble a color recognition system with Arduino. The TCS3200 color sensor (datasheet) uses the TCS3200 chip to detect the RGB color level (Red, Green and Blue, or Red, Green and Blue) of the object that is placed in front of the sensor.

  The TCS3200 chip has 64 photodiodes: 16 with filters for red, 16 for green, 16 for blue and 16 without filters. These photodiodes capture the intensity of the light, filtering the colors and generating the corresponding information on the OUT pin, which will send the data to the microcontroller.



- Speed sensor:

  The speed sensor is used to measure engine speed, pulse count and as a positioning controller. It can be used with the most diverse controllers and boards, such as Arduino, Raspberry Pi and PIC.



As you will continue to observe, the power of software is limited by the hardware. Software means the instructions that a computer understands, such a program in a programming language. There is new software being developed every single day with thousands of thousands of lines of code. However, without hardware, much of this software cannot work or solve a problem. Imagine you want to test a person's heart rate in a faster and more efficient manner. You can write all the code that you want, but the problem cannot be solved without a physical device or hardware. There are many other examples of the important relationship between hardware and software.

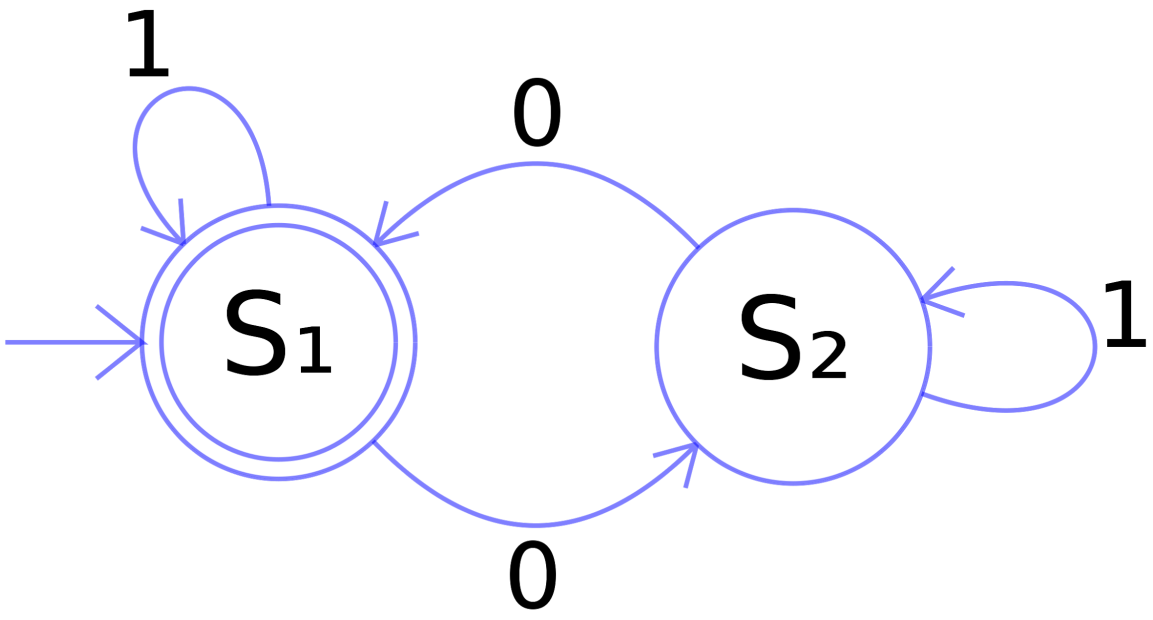
We will take a look at the relationship between an electronic circuit and the programming to make this electronic circuit useful. Electrical Engineering is this idea of circuits, electronics, and other such topics. Computer Science is the idea of programming and fixing errors or bugs. The combination of Electrical Engineering and Computer Science is what we will explore.

An audible alarm (buzzer) is a device that produces sound when we supply its positive (+) terminal with a voltage value of 5 V (HIGH). Its activation is done using an electronic circuit identical to the one used in the case of an LED. The buzzer is known as the hardware component and we will assemble the component by wiring correctly. This will allow the electricity and power to flow in a circuit that will make the buzzer or the LED work. We will then be able to program using code the specific way that the buzzer or the LED should work. Without both components working correctly, we will not get the desired output. Wiring is done on a physical board based on this idea of a completed circuit in order to allow the current to flow. Programming is done on a computer using an Integrated Development Environment(IDE), which we can upload to the physical device for it to work.

Exercise

In this activity, we will use the Arduino to monitor the status of two touch buttons and, based on this information, determine whether the external LED lights flashes or remains off, as well as whether or not the audible alarm should be activated.

Many of these principles and the way of thinking can be summarized by a state diagram and the idea of "states." These are useful when we have many physical components and they do different actions based on different physical inputs based on the set of instructions that a programmer writes. A "state" means the current condition of the program of interest. States are powerful because we can transition between different states in order to make a program. These states and transitions based on specific inputs can create powerful programs for us using both software and hardware.



Reference: https://en.wikipedia.org/wiki/State_diagram#/media/File:DFAexample.svg

In this exercise, we will explore these states and transitions further, seeing that they resemble a flow of actions, inputs, and changes, similar to a computer program. We want to achieve the following effects on devices depending on the states of the buttons:

Left button	Right button	Effect
Rest	Rest	LED off and silent alarm
Rest	Pressed	LED on and alarm silent
Pressed	Rest	LED on and alarm silent
Pressed	Pressed	Flashing LED and alarm activated

The entire electronic circuit is already mounted on your protoboard, so the remaining challenge is to prepare the program on Tinkercad. Remember the code structures seen in the previous activities and use the blocks available in the "Control" and "Mathematics" tabs.

Tip: each row in the table above can be “translated” into IF-THEN statements.

Explanation of States/Actions into IF-THEN statements

Some example pseudocode for the IF-THEN statements is the following:

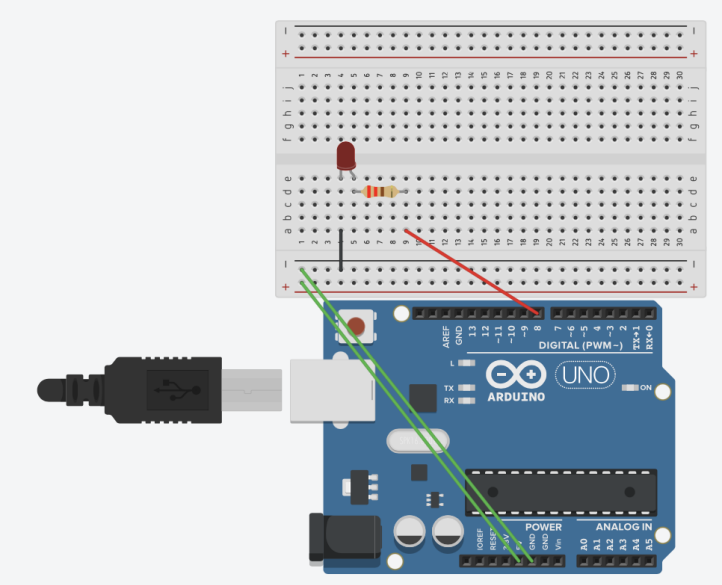
```
1
2       if LEFT_BUTTON == rest && RIGHT_BUTTON == rest:        //Row 1
3           LED = off
4           ALARM = off
5
6       if LEFT_BUTTON == rest && RIGHT_BUTTON == pressed:     //Row 2
7           LED = on
8           ALARM = off
9
10      if LEFT_BUTTON == pressed && RIGHT_BUTTON == rest:     //Row 3
11          LED = on
12          ALARM = off
13
14      if LEFT_BUTTON == pressed && RIGHT_BUTTON == pressed:  //Row 4
15          LED = on
16          ALARM = on
```

Pseudocode means code that is not directly understood by the computer. It helps human beings to get an idea of the flow of the program. In the example above, the details of the syntax or the grammar are not important. The flow and logic of the statements are more important. Each row of the table can turn into an IF-THEN statement by converting the status of the right and the left buttons being pressed or not as the conditionals in our if statements. Then, the output for each of these potential states is whether or not the LED light is on or off. There are some other conditions that we can write in our pseudocode, such as the light flashing instead of just being on. The important aspect is to understand that this set of instructions is going to affect the final output of the physical devices of the LED and the alarm.

There are helpful steps for doing the activity. It is important to understand the concepts of an LED, resistor, basic setup of the code, and the basic setup of the physical circuit. In these steps that follow, we will go through the important topics relating to each step.

Roadmap for reviewing topics

1. One LED circuit



The assembly is already illustrated in Tinkercad with the protoboard. Even though the class has the assembly done already, it is important to note the concepts of a 220 ohm resistor, the LED, and the Arduino itself. Arduino is a tool that allows people to utilize both hardware and software components to make a working system. The physical Arduino board consists of many sensors, wires, pins, and ports. This allows anyone to customize the setup of the circuit in order to create more powerful programs. In this case, the board has an LED that will help complete the circuit in order for the LED to be powered. An LED stands for “Light-Emitting Diode,” which is a way of saying that an LED is a light source that emits light when current flows through it. In order to complete the circuit, we also have a 220 ohm resistor, which is necessary to limit the current and prevent too much or too little current.

2. One button circuit

After everyone has put a button on the protoboard, ask them to add one more button. In many programs, it is important to start simple and then add complexity with more time. Buttons are physical sensors that are either pressed or not. This is similar to a Boolean value, where a button that is pressed has a value of True, which is equivalent to a “1” in many Arduino programs, while a button that is not pressed has a value of False, which is equivalent to a “0” in many Arduino programs.

3. Tinkercad: Prepare a program that makes the LED flash and check if the circuit works

A sample working program for a flashing LED light is the following:

```
int LED_PIN = YOUR_DIGITAL_PIN; //This is equal to a pin of the Arduino board in order to control the LED


//This function runs once as a setup.

void setup()
{

// This sets up the pinMode function that will allow us to write to or
// communicate to the pin of either turning on of off the LED.

 pinMode(LED_PIN,OUTPUT);

}

//This function runs forever

void loop()
{
    digitalWrite(LED_PIN,HIGH); //HIGH voltage would mean the LED turns on
    delay(1000);               //This delays moving forward by 1000 ms(1 sec)
    digitalWrite(LED_PIN,LOW);  //LOW voltage would mean the LED turns off
    delay(1000);               //This delays moving forward by 1000 ms(1 sec)
}
```

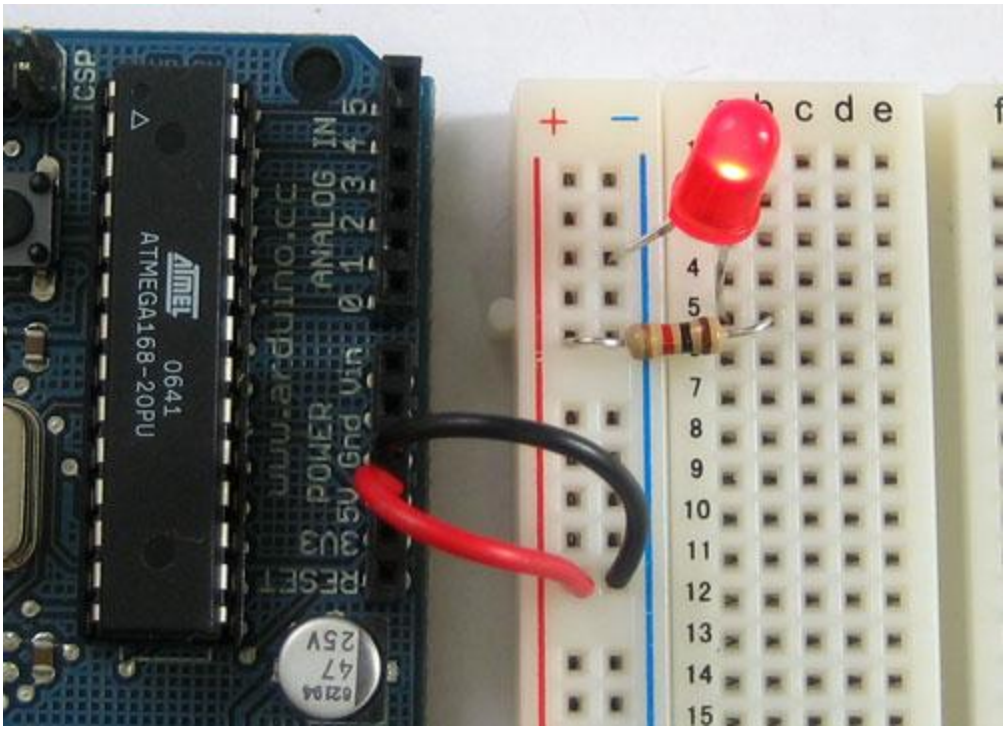
This program is broken down into a simple algorithm. We first turn the LED on and wait one second. Afterwards, we turn the LED off and wait one second. We then continue doing this alternating forever to create the effect of the LED flashing. The Arduino program is broken down into 2 main functions, which are a way of defining custom behaviors in a simple manner that can be done again and again easily.

In the program, we first initialize a variable to hold the digital pin number of the LED that is on the protoboard. This is important in order to actually control the LED in the later parts of the program.

The first function we define is the setup function, which only runs one time and sets the basic infrastructure of the program. In this case, it utilizes a built-in function called pinMode that will now do actions that will behave as an output instead of an input. This is what we want since we want the output of the LED to be either on or off.

The second function is where we will write the main logic of the program. This function runs forever unless stopped by external factors. We first turn our LED light on by communicating to the system that we want a HIGH voltage. At this point, the LED light is turned on. We now wait 1000 milliseconds, which is equal to 1 second. Afterwards, we now turn our LED light off by communicating to the system that we want a LOW voltage. We then wait 1 more second before repeating this behavior forever in a loop. These sequences of steps will make it seem like the LED is flashing.

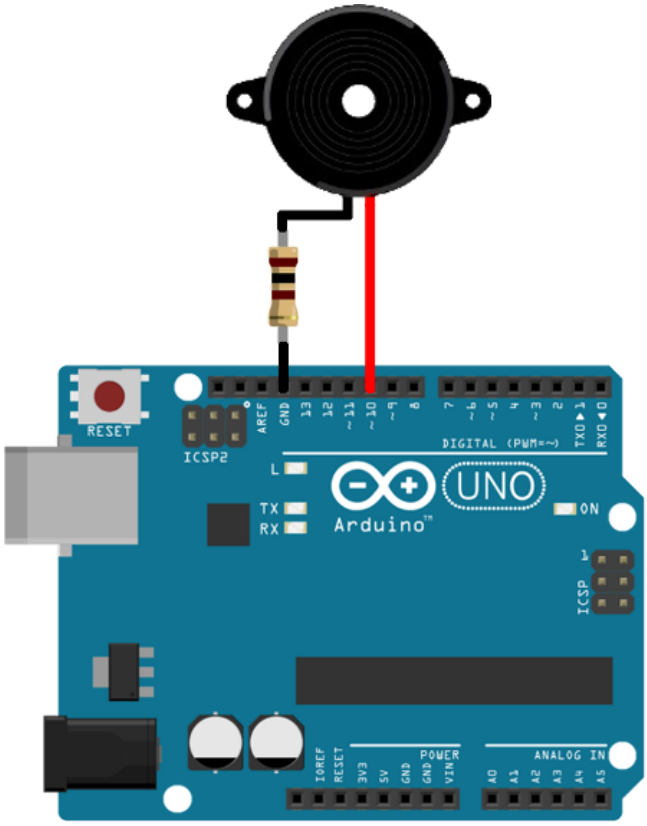
After this point, we remember how to upload a Tinkercad program onto Arduino, passing through the IDE. This will ensure the physical device will do the set of instructions we passed to the device.



Reference: https://www.ladyada.net/learn/arduino/lesson3.html

4. Swap the LED for the buzzer

Quickly discuss that the operation of the buzzer is identical, exchanging the visual (luminous) effect for a sound. There are many times where similar procedures can define many different programs through many similarities. In this case, the behavior of the buzzer is similar to the behavior of the LED since they are getting similar set of instructions for different devices. In computer science, there is a very famous idea of “reductions,” which simplify and relate a known problem to a different problem that can be solved via similar methods. In other words, knowing how to solve the problem of the LED via the same program we wrote gives us a way to solve the problem of the buzzer since the LED problem reduces to the buzzer problem. Some reductions cannot be solved easily and efficiently. There is extensive computer science research on some problems that are just impossible.



Reference: http://www.ozeki.hu/p_2977-how-to-use-a-buzzer-in-arduino.html

5. Think about how to adapt the program so that the LED lights (or flashes) only when one of the buttons is pressed.

   The variation of the program to only allow the LED to flash when one button is pressed can be done through the sequence of observations that follow:

   - Highlight the blocks:
     * Read digital pin.
     * If, then ... another / repeat while. (If there is time, show the two ways to implement the program, drawing attention to the fact that there is more than one way to solve the problem).
     * Equality test (=).
     * For more elaborate tests, with multiple conditions, there is the AND / OR function block.

6. Challenge of making the program with two buttons, LED and buzzer

To make it easier, we will take a complete assembly of the circuit in Tinkercad (publicly open) so that the class can copy and test the program. This idea of using code that is already written is actually popular in larger programs. The reason for this practice is that it is not efficient and easy

to code each specific part of a large program. Therefore, there are many “libraries” that people use to build complicated programs based on certain features that are common and open for everyone to use freely. “Libraries” are pieces of freely available code that people can take advantage of in order to build more complicated code for different systems and programs.