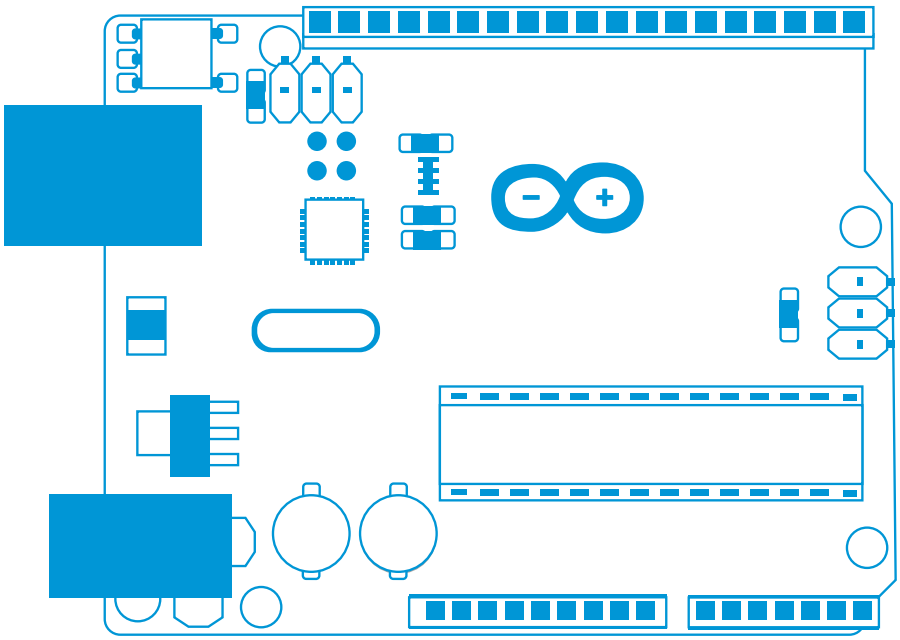
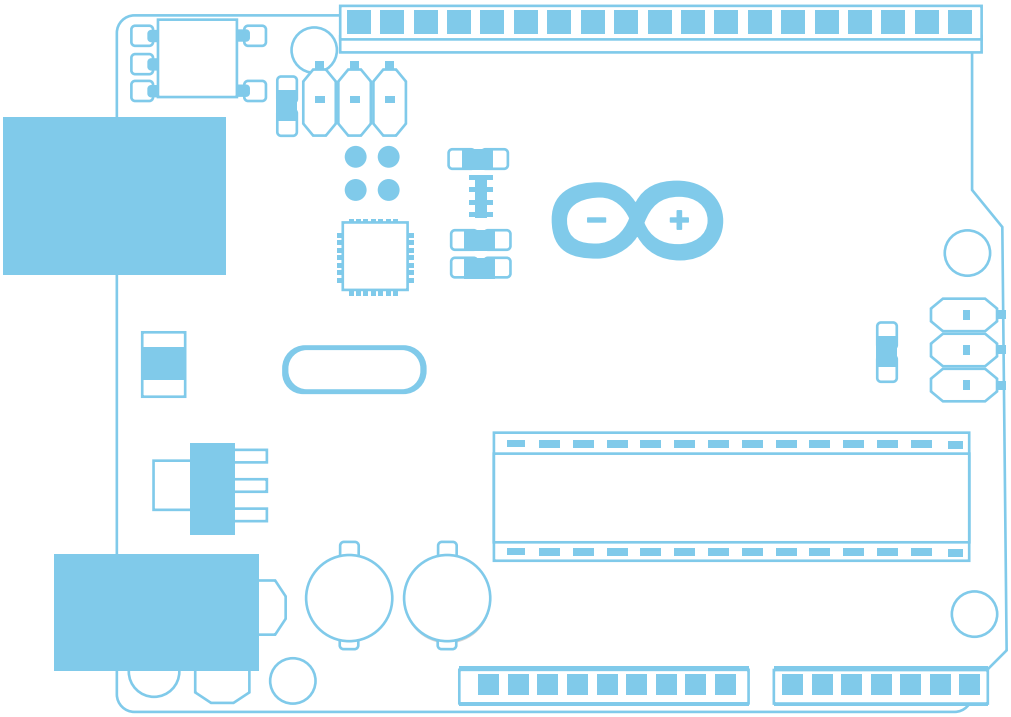
teach with space

→ MEET ARDUINO!

Introduction to Arduino computing using C++





Fast facts page 3
Summary of activities page 4

Activity 0: Getting started page 5
Activity 1: Make me blink! page 8
Activity 2: Blink an S.O.S.! page 11
Activity 3: Measuring temperature page 12
Activity 4: Measuring pressure page 16
Activity 5: Measuring altitude page 18

Student worksheets page 19

Links page 34

teach with space – meet arduino | T04.1
www.esa.int/education

The ESA Education Office welcomes feedback and comments
education@esa.int

An ESA Education production

→ MEET ARDUINO!

Introduction to Arduino computing using C++

Fast facts

Age range: 14-20 years old
Curriculum links: programming, electronics
Complexity: Medium
Lesson time required: 90-120 minutes
Location: Indoor
Includes the use of: Arduino software and hardware
Keywords: Arduino, Sensor, Code

Outline

Students will explore technology used in space through the Arduino tool. They will build circuits to blink an LED and to measure temperature, pressure and altitude. The basics of programming in C++ will be introduced using the Arduino IDE (Integrated Development Environment) software. This set of activities could be the starting point for future participation in the CanSat competition.

Learning objectives

- Improve analytical skills
- Develop understanding of a programming language
- Develop understanding of building circuits
- Understand how sensors can be used to collect data
- Improve teamwork skills

Activities - Content and outcome					
	Title	**Description**	**Outcome**	**Requirements**	**Time**
1	Getting started	An introduction to the components used throughout the activity	Students will become familiar with required components and their functions	Basic understanding of electronic components	10 minutes
1	Make me blink!	Students will build their first Arduino circuit	Students will be able to control an LED with code they have written	Completion of previous activity	15 minutes
2	Blink an S.O.S	Students will learn how to send more complex messages using an LED	Students will be able to write the code and program the LED to send an S.O.S	Completion of previous activities	15 minutes
3	Measuring temperature	Students will use a thermistor to measure the temperature of their surroundings	Students will be able to build more complex circuits and use them to take measurements of the environment	Completion of previous activities	20 minutes
4	Measuring pressure	Students are shown how to use a pressure sensor to measure the local air pressure	Students will be able to build more complex circuits and use them to take measurements of the environment	Completion of previous activities	20 minutes
5	Measuring altitude	Students are shown how to determine altitude from local air pressure measurements	Students will be able to confidently build circuits and manipulate code to take measurements of their surroundings	Completion of previous activities	20 minutes

→ Activity 0: Getting started

Introduction

Arduino is an open-source platform used to build a wide variety of electronic projects, including the possibility to simulate real space missions like ExoMars (http://exploration.esa.int/mars/). Arduino incorporates both a physical, programmable circuit board (hardware) and a piece of software or IDE (Integrated Development Environment) that runs on a computer.

The Arduino hardware and software was designed for artists, designers, hobbyists, hackers and people interested in creating interactive objects and environments. Arduino can be used to interact with buttons, LEDs, sensors, motors, speakers, GPS units, cameras, the internet, and even your smartphone or TV! This flexibility, combined with the facts that the Arduino software is free, the hardware boards and sensors are relatively cheap, and both the software and hardware are intuitive, has led to a large community of users (www.arduino.cc).

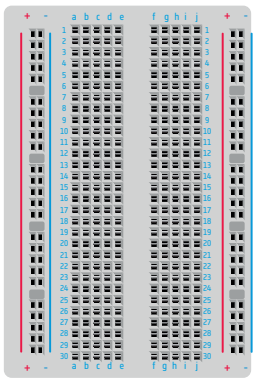
Hardware

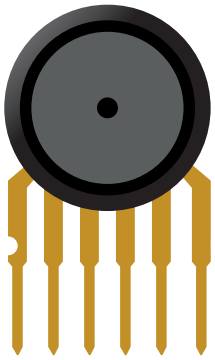
For this activity we are going to use the Arduino Uno (Figure A1, box number h). The Arduino can provide voltage to its pins (metallic legs) and read the voltage from them. Notice the number printed next to each pin, with **GND** (ground) acting as the negative pole and **Vin** or **5V** acting as positive poles (like the positive and negative sides of a battery).

We are going to use the following components shown in Figure A1:

a. **Breadboard:** A support structure to easily connect electronic components.
b. **Pressure sensor MPX4115A:** A component that measures the absolute ambient pressure.
c. **Resistors of 220Ω and 10kΩ:** To control the current flow. Resistance is measured in ohms (Ω). Resistors have coloured bands on them to indicate their value.
d. **USB cable:** Used to connect the Arduino to the computer and/or power source.
e. **Cables:** Used to electrically connect components.
f. **LEDs (Light Emitting Diode):** Electronic components that generate light when electricity is passed through them.
g. **Temperature sensors:** An analog sensor (such as a LM35) and a thermistor (a resistor that is especially sensitive to temperature).
h. **Arduino board:** Works like a simple computer that contains a processor, memory and some input/output pins.

Figure A1

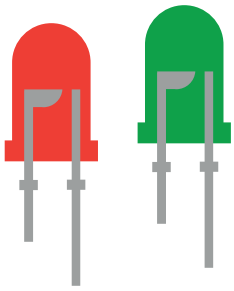

a.Breadboard

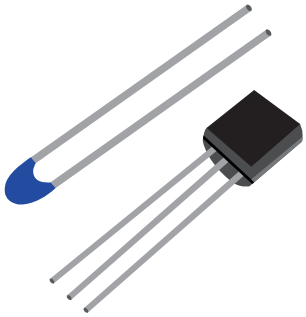

b.Pressure sensor

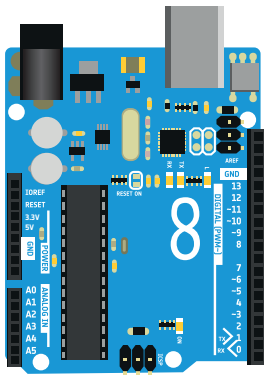

c.Resistors


d.USB cable


e.Cables

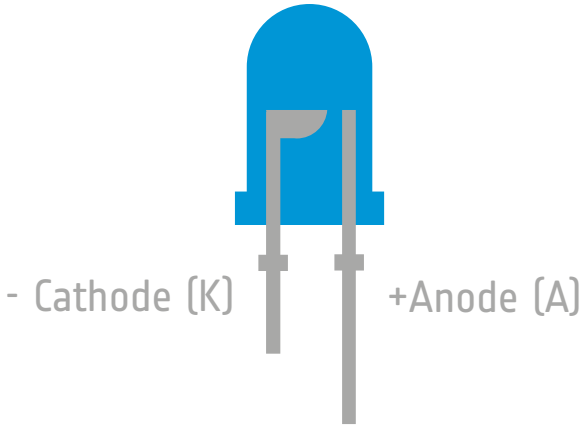

f.LEDs (Light Emitting Diode)


g.Temperature sensors


h.Arduino board

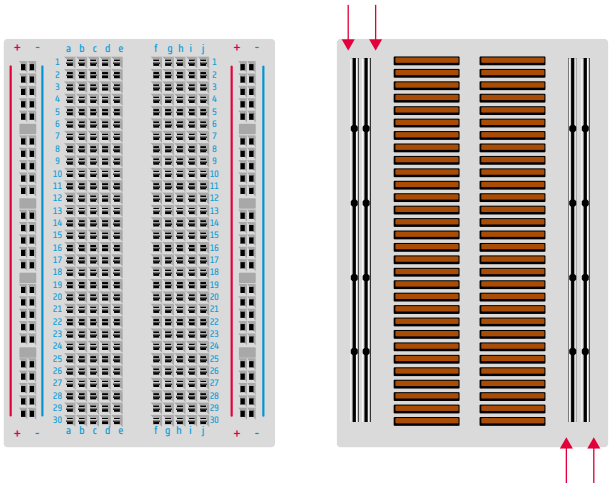
↑ Arduino basic kit components

Figure A2



↑ LED: Note that the LED has a long and a short leg. The long leg is called the anode and the short the cathode.

Figure A3



↑ Breadboard: Note that all the pins of each row are connected.

When you connect the Arduino, your computer should recognise it and initiate the driver installation process. If you have problems installing drivers you can find further instructions here: www.arduino.cc/en/Guide/ArduinoUno

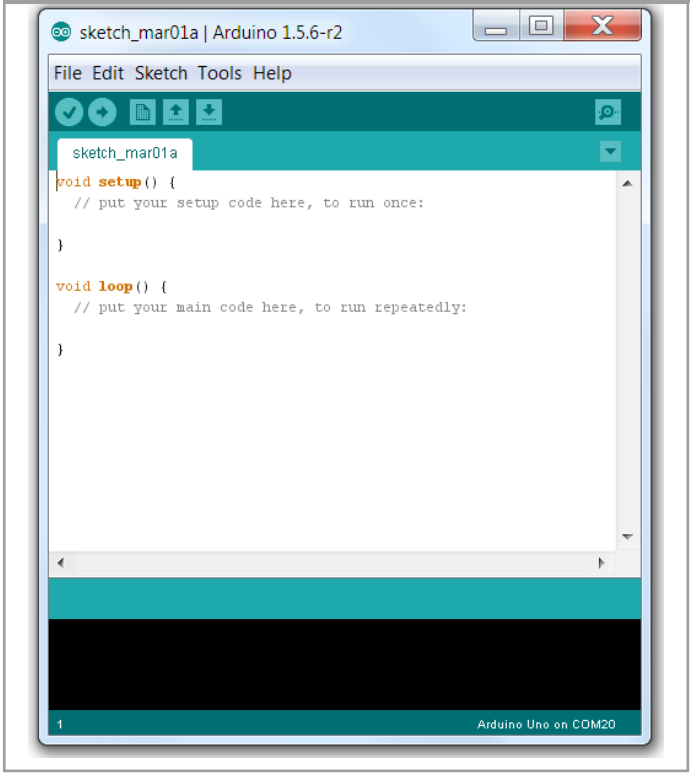
*Please be sure to select the right board (try Genuino Uno) and the correct serial port (COM) in the Tools menu!

Software

The Arduino software can be easily downloaded and installed here:
www.arduino.cc/en/Main/Software

Programs written using the Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the .ino file extension. The editor has features for cutting/pasting and for searching/replacing text. The message area at the bottom (Figure A4) gives feedback while saving and exporting and also displays errors. The selected board and serial port are displayed at the bottom right corner of the window. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches and open the serial monitor (to view data collected from sensors). Figure A4 is a screenshot of the IDE Arduino graphical interface.

Figure A4



↑ Arduino IDE interface

Arduino commands - Table A1	
	**Verify:** Checks your code for errors when compiling it.
	**Upload:** Compiles your code and uploads it to the configured board.
	**New:** Creates a new sketch.
	**Open:** Presents a menu of all the sketches in your sketchbook.
	**Save:** Saves your sketch.
	**Serial Monitor:** Opens the serial monitor, allowing you to see data collected from sensors.

Additional commands are found within the five menus: File, Edit, Sketch, Tools and Help.

→ Activity 1: Make me blink!

In this activity students will control an LED with the Arduino. The purpose is to control the blinking of the LED and to understand how the code is written/used to control each LED. More information can be found through a quick Google search: Arduino LED blink.

Equipment

- 1 Arduino Uno
- 1 breadboard
- 1 LED (green)
- 1 220Ω resistor (red/red/black/black/brown)
- 2 wires

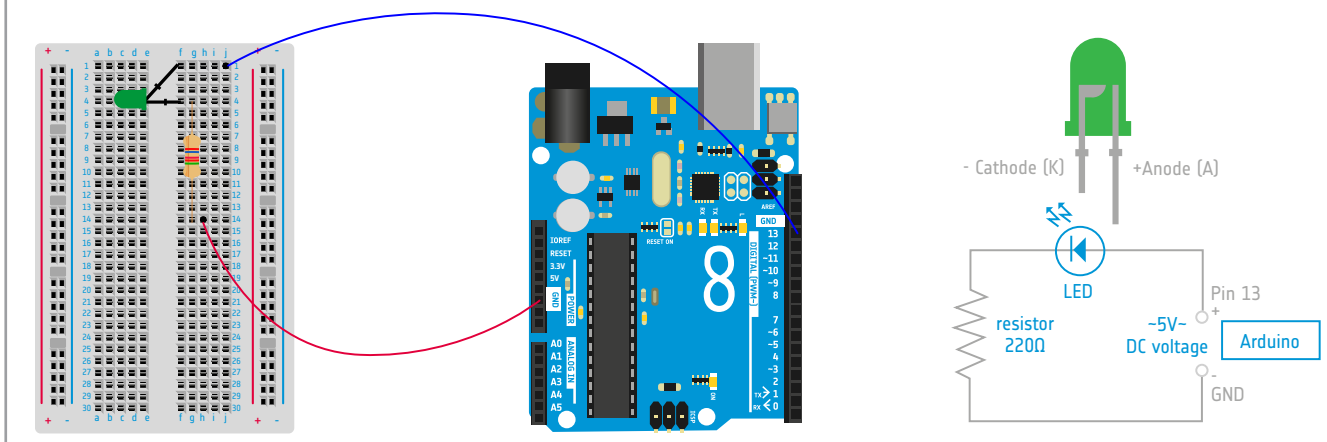
Exercise

Setup

The circuit is set up as follows:

a. Connect the green LED on the breadboard as shown in Figure A5.
b. Connect the 220**Ω** resistor (red/red/black/black/brown) to the short leg of the LED as shown in Figure 5.
c. Using a wire, connect the long leg of the LED to pin number **13** of the Arduino Uno
d. Using a wire, connect the resistor to the **GND** pin of the Arduino Uno.
e. Finally, connect the Arduino UNO to the computer using the USB cable .

Figure A5



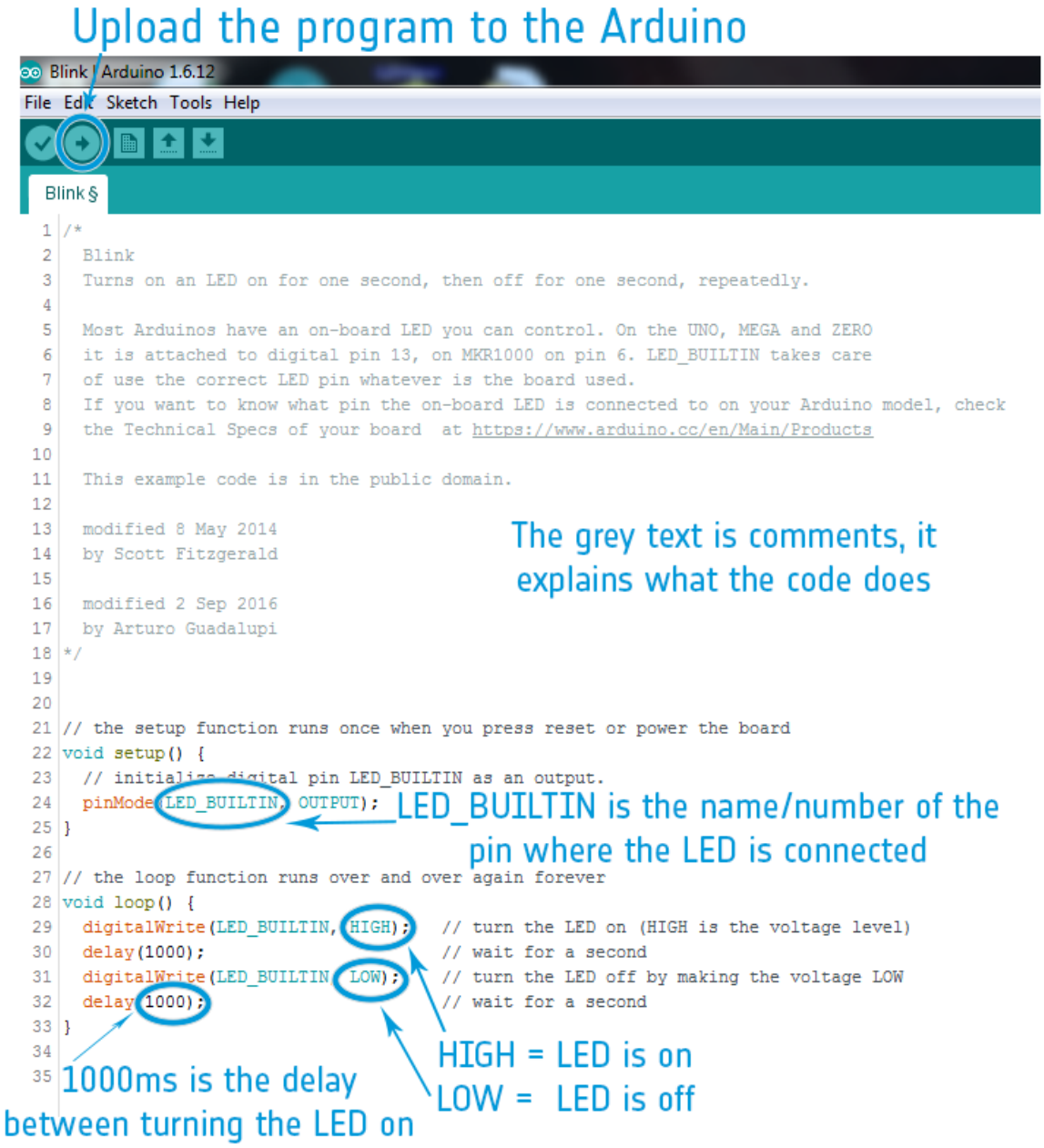
↑ Arduino circuit ready to blink the green LED

Code

Students will now upload a program to control the LED. They will use an example code that can be found by opening the Arduino Software (IDE) and clicking on:

File → Examples → Basics → Blink

The Blink code will appear on the screen and is shown in the image below. Comments are displayed in grey after the slashes and explain each line of the code.




```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN takes care
  of use the correct LED pin whatever is the board used.
  If you want to know what pin the on-board LED is connected to on your Arduino model, check
  the Technical Specs of your board  at https://www.arduino.cc/en/Main/Products

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

The main elements of the above code are explained below:

```
void setup()
```

This command is always written at the beginning when a sketch starts; it initializes the variables that are going to be used in the rest of the sketch. It will only run once, after each power up or reset of the Arduino board.

```
pinMode(LED_BUILTIN,OUTPUT)-> pinMode(13,OUTPUT)
```

Configures the specified pin to behave either as an input or an output.
In the blink example, replace the **LED_BUILTIN** by the number of the pin where the LED is connected, i.e. **13**.

```
void loop()
```

This command does precisely what its name suggests and consecutively loops all the commands that it contains, allowing your program to do multiple things successively.

```
digitalWrite(13,HIGH)
```

Turns the LED on. HIGH sets the voltage to around 5V to pin 13.

```
delay(1000)
```

Pauses the program for the amount of time indicated (in milliseconds).

```
digitalWrite(13,LOW)
```

Turn the LED off by making the voltage LOW. Low means giving a 0V voltage to pin 13.

A complete list of all possible commands can be found through a quick Google search: Arduino commands.

→ Activity 2: Blink an S.O.S.!

In this activity students use the Arduino circuit built in activity 1. They will control the green LED to send a message in Morse to demonstrate how they could communicate with a rover on Mars. To send an O, students should change the delay in the code to increase the time of the lighted green LED. In this case we changed 1000 ms to 5000 ms, but it could be any number greater than 1000 ms. The code is shown in Figure A6:

Figure A6

```
void loop() {
  digitalWrite(13, HIGH);
  delay(5000);
  digitalWrite(13, LOW);
  delay(1000);

  digitalWrite(13, HIGH);
  delay(5000);
  digitalWrite(13, LOW);
  delay(1000);

  digitalWrite(13, HIGH);
  delay(5000);
  digitalWrite(13, LOW);
  delay(1000);

}
```

↑ Code to send an O in Morse

→ Activity 3: Measuring temperature

In this activity, students measure temperature using the Arduino. The purpose is to read temperature from the temperature sensor (thermistor*) and to understand how the code is written/used to control the sensor. More information can be found through a quick Google search: Arduino thermistor sensor.

Equipment

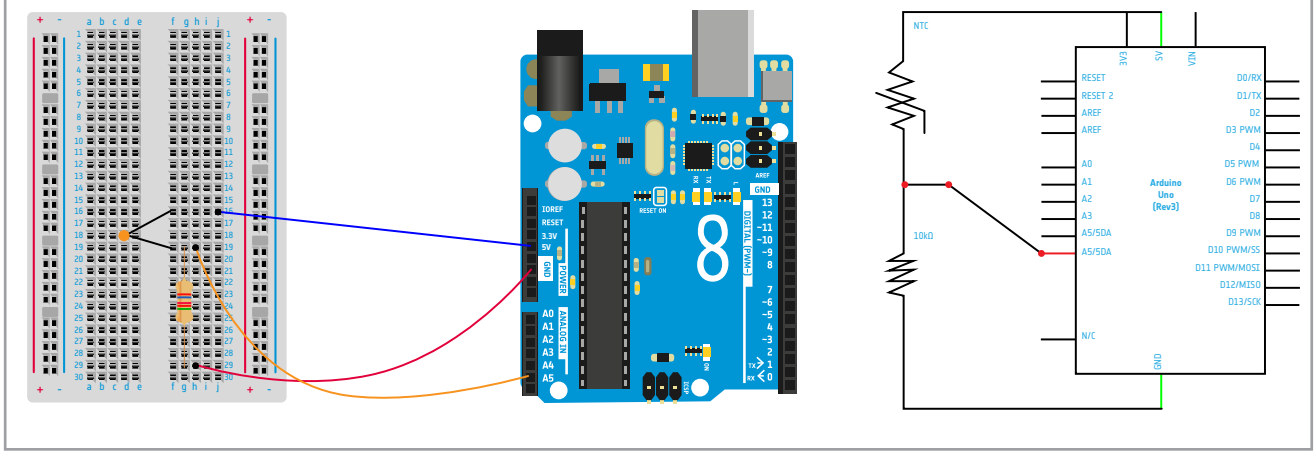
- 1 Arduino Uno
- 1 breadboard
- 1 temperature sensor (thermistor*)
- 1 10kΩ resistor (brown/black/black/red/brown)
- 3 wires

Exercise

Setup

a. Connect the thermistor on the breadboard as shown in Figure A7 .
b. Connect the 10kΩ resistor (brown/black/black/red/brown) to the thermistor as shown in Figure A7.
c. Using a wire, connect the thermistor to the **5V** pin of the Arduino Uno
d. Using a wire, connect the resistor to the **GND** pin of the Arduino Uno.
e. Using a wire, connect the thermistor and the resistor to the **A5** pin of the Arduino.
f. Finally connect the Arduino UNO to the computer using the USB cable .

Figure A7



↑ Arduino circuit ready to measure the temperature

* thermistor: an electrical resistor whose resistance is greatly reduced by heating

Code

Students will now write their own code. A complete list of all possible commands can be found through a quick Google search: Arduino commands. Figure A8 illustrates an example of a code that measures the temperature. In this example, the temperature that is read from the sensor in the circuit is called **AnalogT**, and is in units of volts because it is an electric signal; it does not have a physical meaning of temperature. The temperature that is converted to degrees Celsius (the unit of temperature used in Europe) is called **AnalogTf**. The name of the pin on the Arduino Uno is A5. Once the code is uploaded and begins running, open the Serial Monitor* window in order to see the data being collected.

Figure A8

```
1  void setup() {
2    // put your setup code here, to run once:
3  Serial.begin(9600);
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8  float AnalogT;
9  float AnalogTf;
10 AnalogT= float(analogRead(A5));
11 AnalogTf = (-40000/AnalogT) + 100;
12 Serial.println();
13 Serial.print("Temperature: ");
14 Serial.print(AnalogTf);
15 Serial.print("C");
16 delay(1000);
17
18 }
```

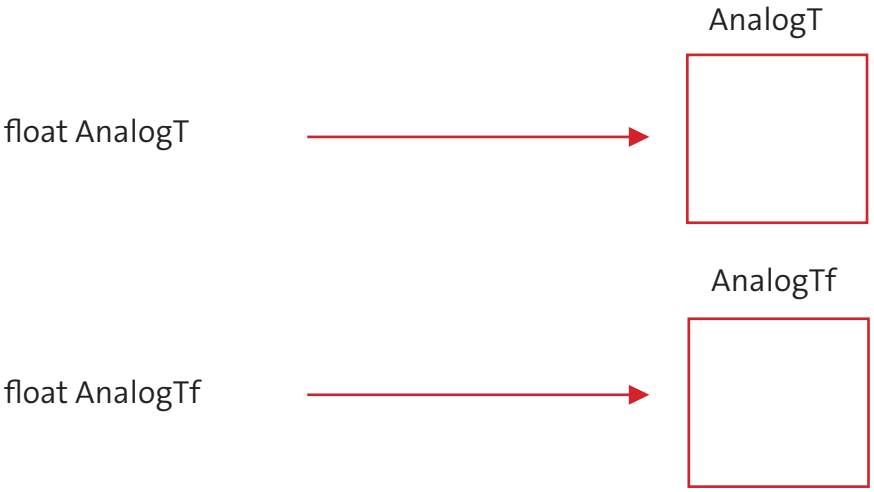
↑ Arduino code for measuring the temperature

*see Table A1: Arduino commands.

For explanations of the void setup and void loop, please refer to Activity 1 Exercise 3.

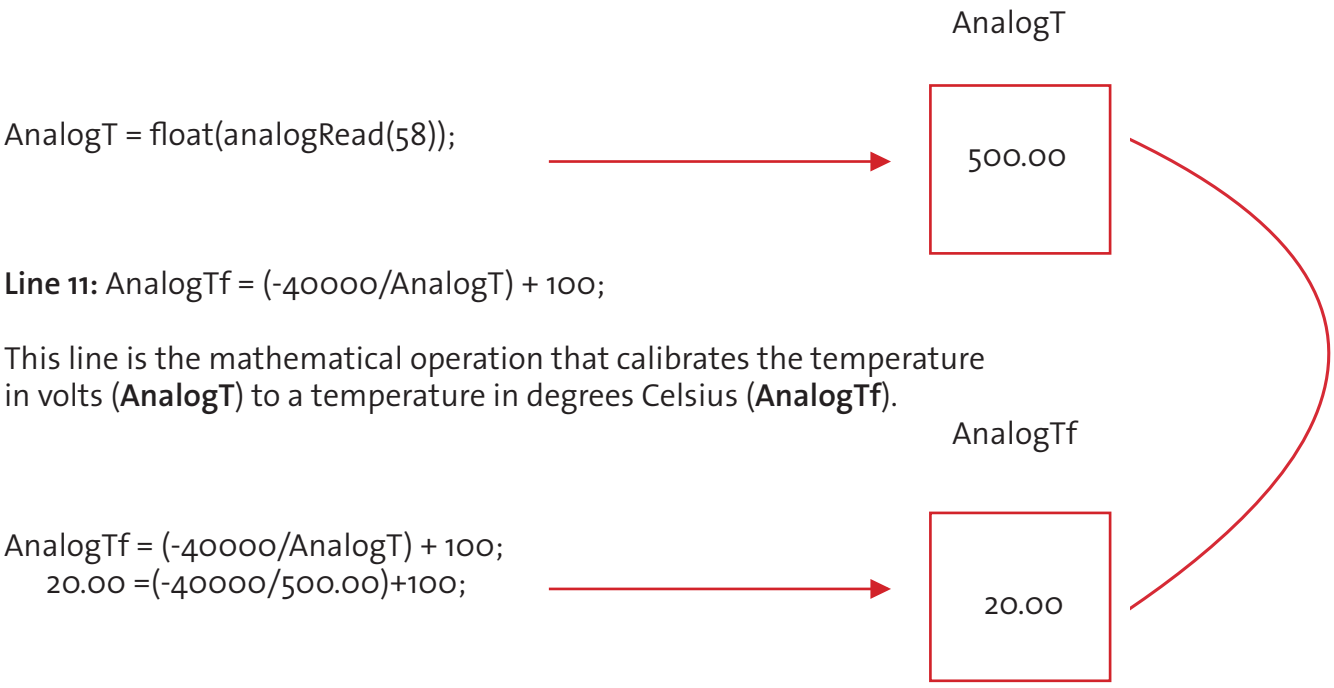
Line 3: As the temperature will be printed on the screen of the computer, it is necessary to predefine the data rate (9600 bits/second) for the data transmission between the Arduino and the computer.

Line 8-9: These two lines of code are used to define variables called AnalogT and AnalogTf. These names are chosen by the programmer. To model this process, we will use a graphical explanation. When you write float AnalogT, inside the memory of the computer you create a box or an empty space called AnalogT that has a specific size indicated by 'float'. Float means that the number inside the box will have a decimal point.



Line 10: AnalogT = float(analogRead(A5));

The computer will write the number read from the A5 pin on the Arduino Uno (e.g. 500.00) inside the AnalogT box. We use the function **analogRead()** to read any pin from the Arduino.



Line 11: AnalogTf = (-40000/AnalogT) + 100;

This line is the mathematical operation that calibrates the temperature in volts (**AnalogT**) to a temperature in degrees Celsius (**AnalogTf**).

Lines 12-15: To display the measure of the temperature on the screen, you need to use the print function. This function will recover the number inside any boxes and print it in the Serial monitor. In this example, the function will print the number inside the box called AnalogTf. All the words inside the "..." are directly printed on the serial monitor.

Line 16: The delay function is used to wait 1000 milliseconds to display the temperature on the serial monitor.

Discussion

Students will probably have to calibrate their own temperature sensor. They can use another thermometer to measure the real temperature as a reference for calibrating their sensor. To roughly calibrate, they run the code as currently written and if their data output is not the same as the real reference temperature, they try changing the last number (+ 100) of line 11 (AnalogTf) of the code in Figure A8.

Practice

Students are asked to calibrate the temperature sensor as if they were on the surface of Mars. As the average temperature of the Martian surface is considered to be around -60°C, and +14°C for Earth, students should calculate a shift of -74. The code should be adapted like this:

AnalogTf = (-40000/AnalogT) + 100 – 74;

→ Activity 4: Measuring pressure

In this activity, students measure the ambient pressure with the Arduino. The purpose is to read the pressure from the sensor (MPX4115A) and to understand how the code is written/used to control the sensor. More information can be found through a quick Google search: Arduino MPX4115A.

Equipment

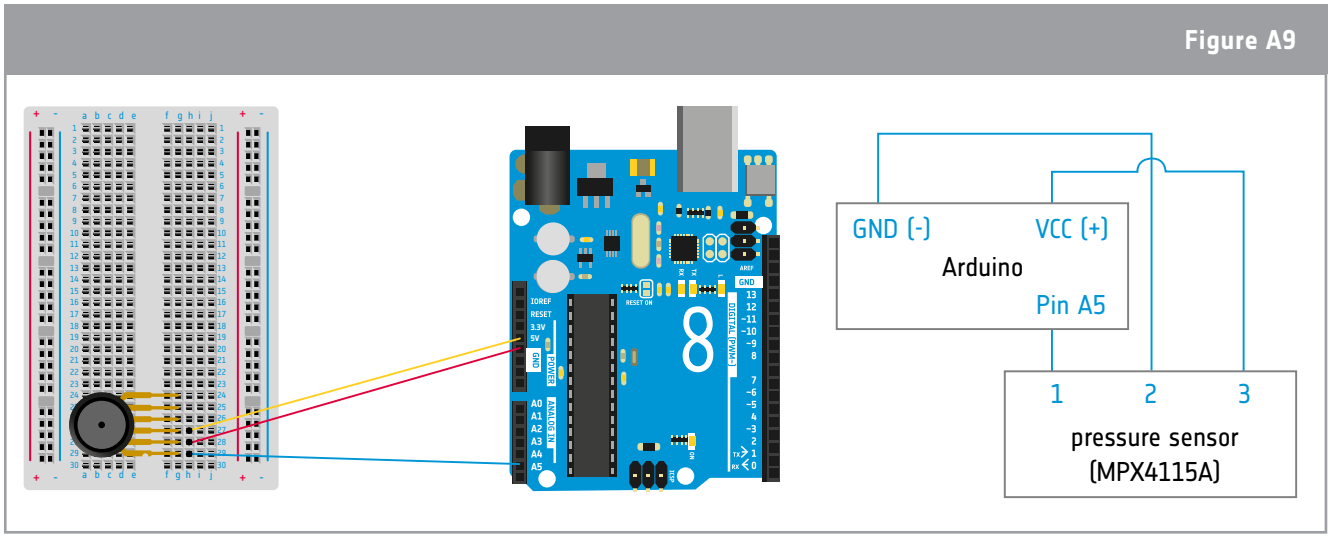
- 1 Arduino Uno
- 1 breadboard
- 1 pressure sensor
- 3 wires

Exercise

Setup

a. Connect the pressure sensor to the breadboard.
b. Using a wire, connect leg 1 of the pressure sensor (marked with a notch) to pin **A5.**
c. Using a wire, connect leg 2 of the pressure sensor in the **GND** row of the Arduino.
d. Using a wire, connect leg 3 of the pressure sensor to pin **5V** of the Arduino.
e. Finally connect the Arduino UNO to the computer using the USB cable .

Figure A9



↑ Arduino circuit ready to measure the pressure

Code

In this case, students use the same commands as in the previous temperature activity; again they will focus on calibrating the sensor. An important and highly useful step in using sensors is to look at their datasheets, which contain detailed information about how the sensor works. In this case, we can easily find useful information with a quick Google search: MPX4115A datasheet.

The pressure sensor converts the measured pressure into a voltage. This is an analogue signal that will be translated to a digital signal. To translate the measured voltages to ambient pressure values (in units of Pa or pascals in the International System of Units), the transfer function of the sensor is needed. This function describes the mathematical relation between the voltage output of the sensor and the equivalent pressure. This function can be found in the datasheets of the sensor. This function can vary from sensor to sensor.

If we look at the datasheet, we will find the following transfer function: $\mathbf{V_{out} = V_S\ (P\ x\ 0.009 - 0.095)}$. We only need to isolate the value of P (Pressure) and take into account that V_S = 1024 (the voltage read is divided in 1024 steps). $\mathbf{V_{out}}$ = Analogue value that we will read from pin **A5**.

Below is an example of the code that prints the values of pressure. Use it as a reference.

Figure A10

```
1  void setup() {
2    // put your setup code here, to run once:
3     Serial.begin(9600);
4  }
5
6  void loop() {
7    // put your main code here, to run repeatedly:
8     float AnalogP;
9     float AnalogPf;
10    AnalogP= float(analogRead(A5));
11    AnalogPf = ((AnalogP/1024)+0.095)/0.009;
12    Serial.println();
13    Serial.print("Pressure: ");
14    Serial.print(AnalogPf);
15    Serial.print("kPa");
16    delay(1000);
17
18 }
```

↑ Arduino code for measuring the pressure

The name given for the pressure in volts is AnalogP and for the pressure in pascal is AnalogPf. The name of the pin where the pressure sensor is connected is A9.

To calibrate their pressure sensor students can check the ambient pressure of their location on the internet. To roughly calibrate, they run the code as currently written and if their data output is not the same as the real reference pressure, they can adapt the line 11 (**AnalogPf**) of the code.

→ Activity 5: Measuring altitude

Pressure varies with altitude, and so pressure can be used to measure altitude. To measure altitude, students need to add this part of the code inside the loop of the code measuring the pressure.

Below is an example of the code that prints the altitude in metres. Use it as a reference.

Figure A11

```
float Altitude;
float Altitudef;
  Altitude = pow(AnalogPf/101.325,0.1903);
  Altitudef = (1-Altitude)*44300 + 100;
Serial.print("Altitude: ");
Serial.print(Altitudef); Serial.print ("meters");
```

↑ Arduino code for measuring the pressure

The name given to the altitude in meters is Altitudef. The pressure in units of kPa is then given by AnalogPf, in this example.

From the hints given in the 'Did you know' box, the students should realise that as altitude increases, the temperature of the atmosphere decreases, and this relationship should be seen in their graph of altitude and temperature.

How to use Arduino in the classroom

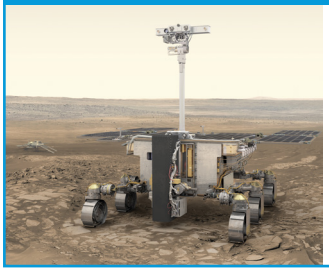
Arduino is an excellent tool to introduce students to basic programming and the wide range of applications it has. It can also be used to teach and learn STEM related subjects. Some examples are:

- **Introduction to circuit basics** - Activity 1 can be used to introduce students to a simple circuit.
- **Introduction to data analysis** - Students collect data from the real world (temperature and/or pressure) and analyse data using graphs. Students read information and draw conclusions.
- **Introduction to weather elements and how data is collected** - Students can develop their own weather station and predict sunny/cloudy days.

→ MEET ARDUINO!

Introduction to Arduino computing using C++

Did you know?



The 2020 mission of the ExoMars programme will deliver a European rover and a Russian science payload to the surface of Mars to establish if life ever existed there!

→ Activity 0: Getting started

Everyone, every day, uses technology! In space missions, engineers use technology to command robots and communicate with them to extend our scientific knowledge. In this activity, you will become the engineer to investigate Mars, the Red Planet, using a technology tool called Arduino. Let's find out how to identify components used in Arduino!

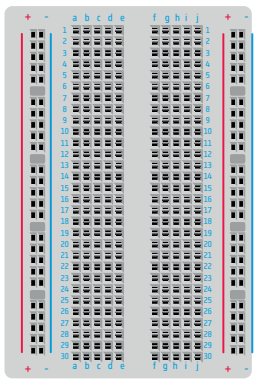
Equipment

- The Arduino basic kit

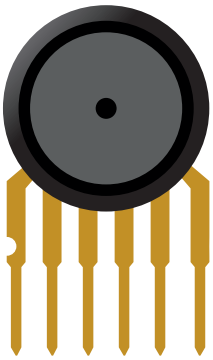
Ground Station components:

1. Work in pairs. Open your Arduino kit and match the components with the descriptions on the next page.

Figure A1



a.Breadboard

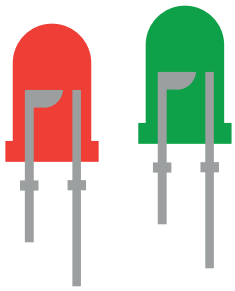


b.Pressure sensor

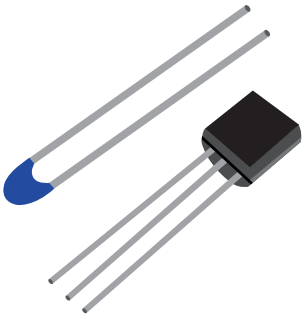
c.Resistors

d.USB cable

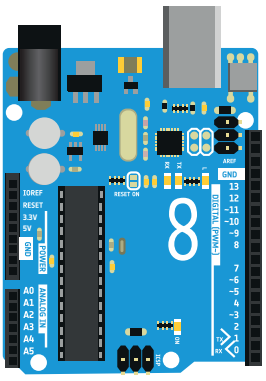
e.Cables



f.LEDs (Light Emitting Diode)



g.Temperature sensors



h.Arduino board

↑ Arduino basic kit components

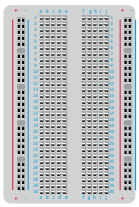
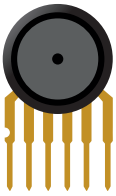
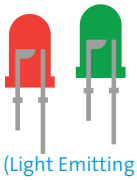
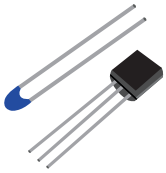
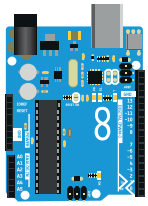
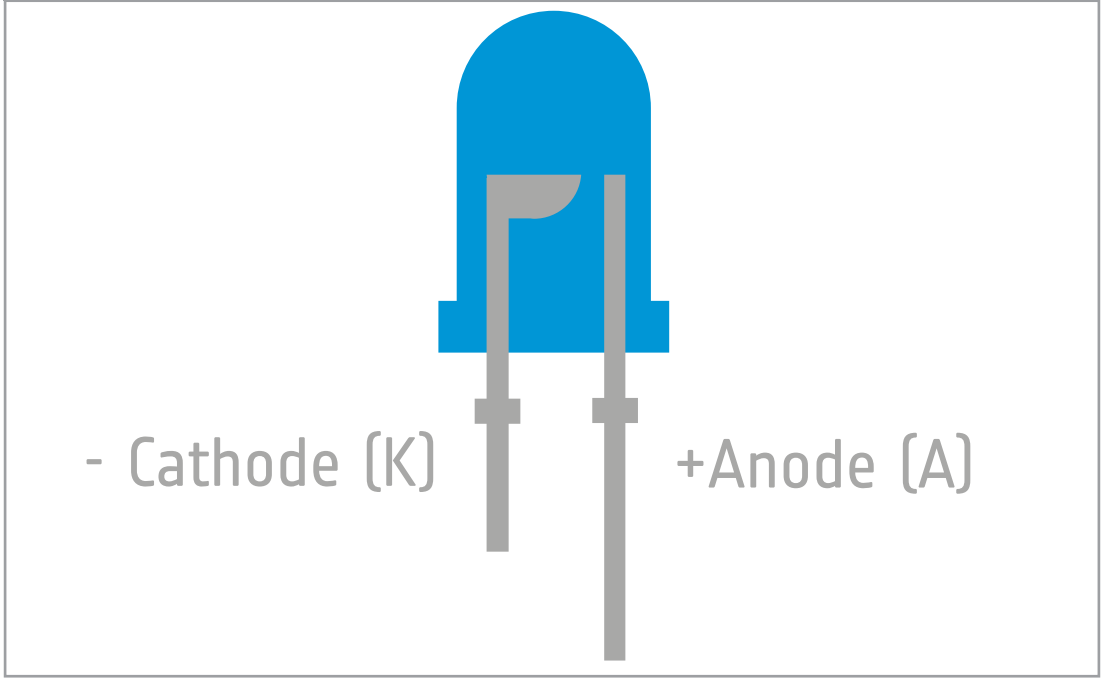
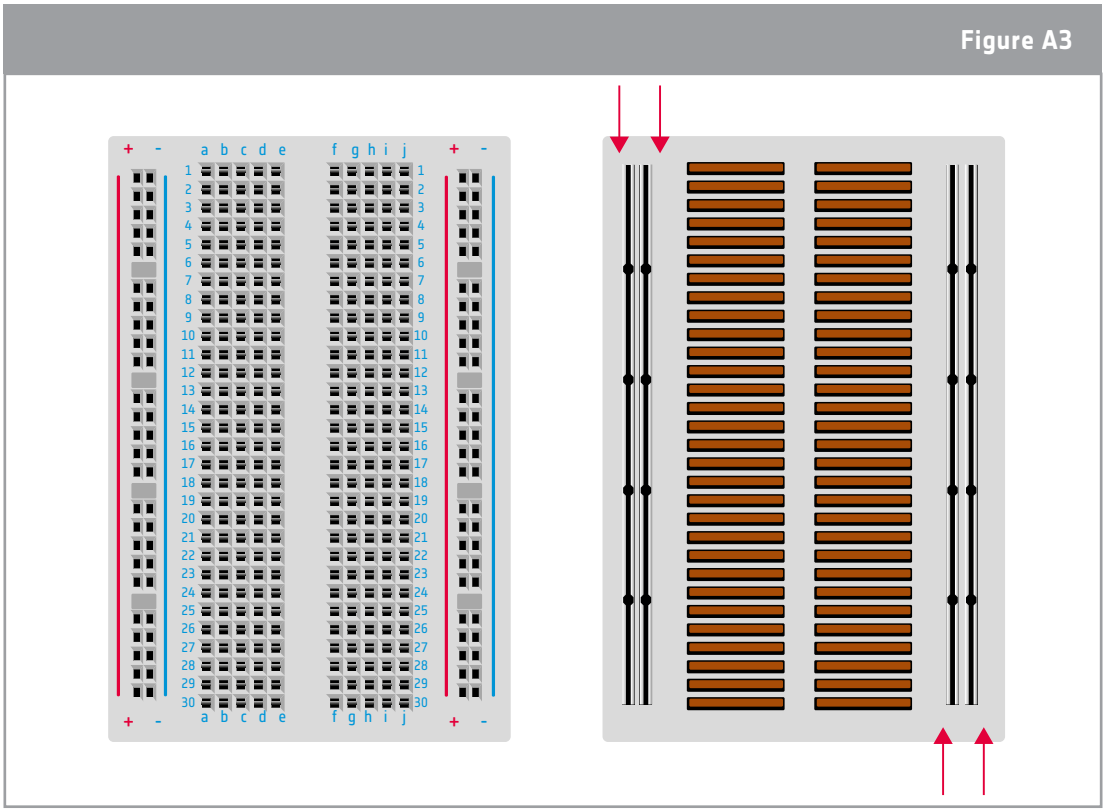
COMPONENT	DESCRIPTIONS
 Breadboard	has a circular shape and measures ambient pressure
 Pressure sensor	has input and output pins and works like a simple computer
 Resistors	is a white support used to easily connect electronic components
 USB cable	has two long legs connected to a blue head and measures the temperature
 Cables	cylinder shape with coloured bands, they reduce the current flowing in a circuit
 LEDs (Light Emitting Diode)	are red and green and emit light when electricity passes through them
 Temperature sensors	have various colours and are used to connect components to conduct electricity
 Arduino board	a long black cable used to connect the Arduino Uno to the computer

Figure A2



↑ LED: Note that the LED has a long and a short leg. The long leg is called the anode and the short the cathode.

Figure A3



↑ Breadboard: Note that all the pins of each row are connected.

→ Activity 1: Make me blink!

To command interactive objects, you need to use a tool such as an Arduino, made up of several electronic components. As you are the engineer in charge of studying the Red Planet, you need to communicate with the rover to command its experiments and collect its data. In this activity you will learn the C++ programming language in order to communicate with your Arduino Uno, and you will learn how to control light!

Equipment

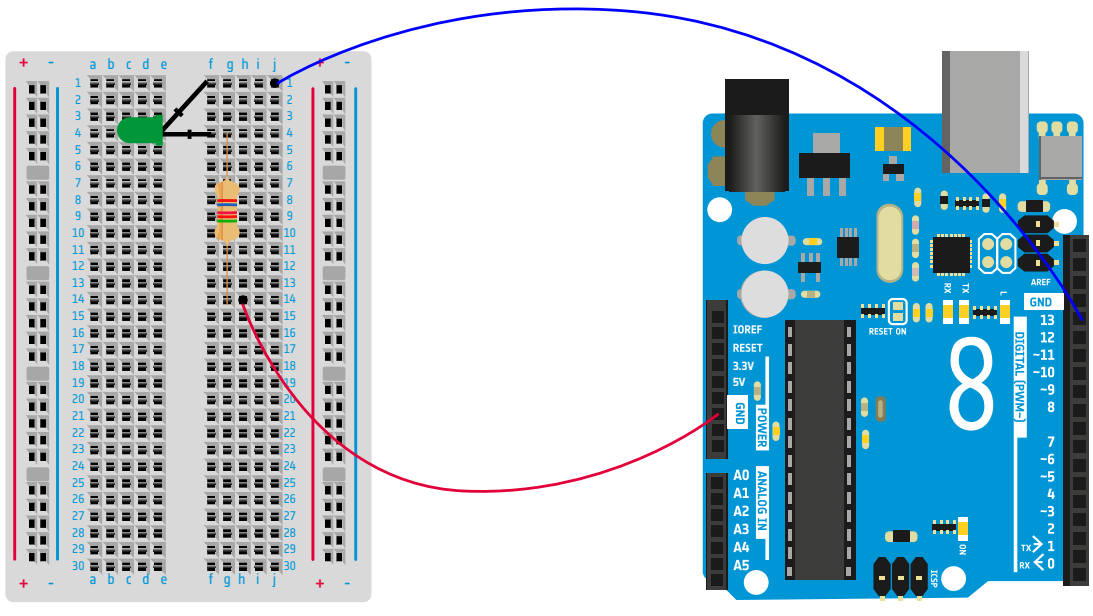
- 1 Arduino Uno
- 1 breadboard
- 1 green LED
- 1 resistor 220Ω(red/red/black/black/brown)
- 2 wires

Exercise

1. Carefully follow these instructions to set up the breadboard and control the green LED:

   a. Connect the green LED (clear plastic) on the breadboard as shown in Figure A4.
   b. Connect the 220Ω resistor (red/red/black/black/brown) to the short leg of the LED as shown in Figure A4.
   c. Using a wire, connect the long leg of the LED to pin number **13** of the Arduino Uno.
   d. Using a wire, connect the resistor to the **GND**(ground) pin of the Arduino Uno.
   e. Connect the Arduino Uno to the computer using the USB cable.

Figure A4



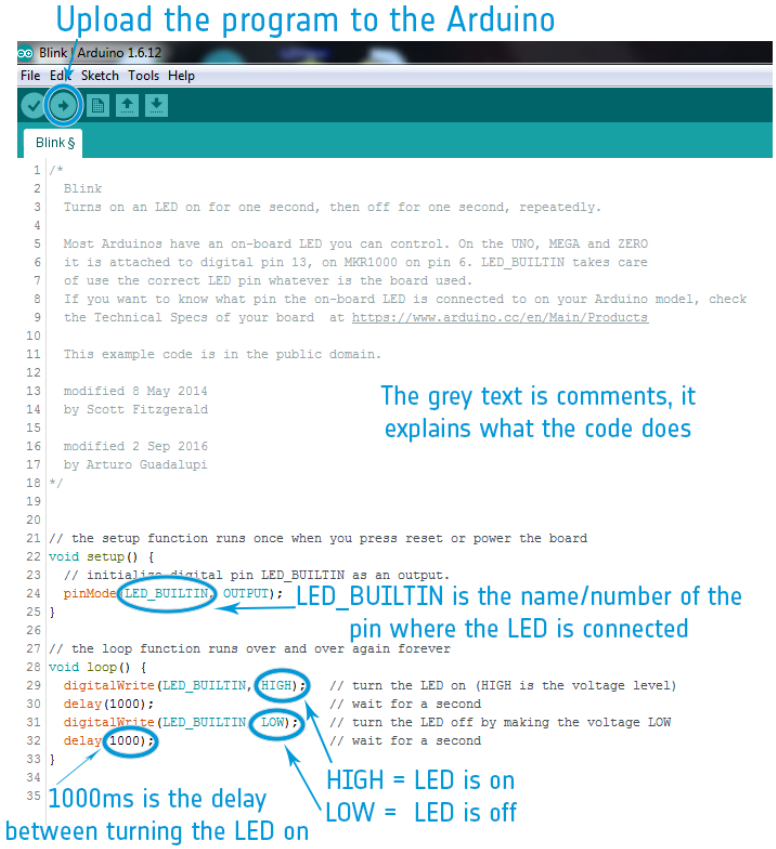
↑ Arduino circuit ready to blink the green LED

2. In the box below sketch the electrical circuit corresponding to the electrical connexions of Figure A4. Use these symbols:

3. Now that the breadboard is ready, you need to send instructions to the Arduino Uno to blink the LED.

   Open the Arduino software (IDE) on the computer and click on File → Examples → Basics → Blink.

   The Blink code will appear on the screen and is shown in the figure below. Carefully read the explanation to fully understand:

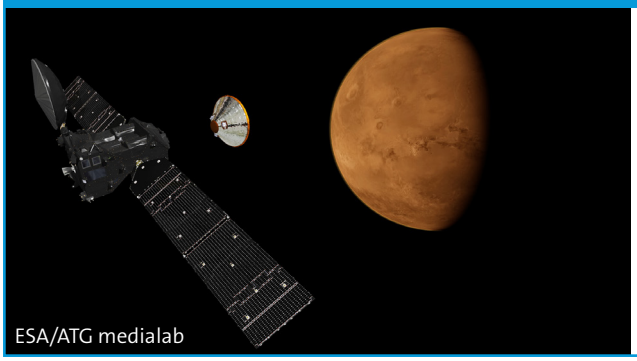


a. Before uploading the code to the Arduino Uno, you need to match the name of the pin in the code with the number of the pin where the green LED is connected to the Arduino Uno. Explain what you changed in the code to control the green LED:

b. Upload the code to the Arduino Uno by clicking on the  at the top left of the screen.

Congratulations, the green LED is blinking!

Did you know?



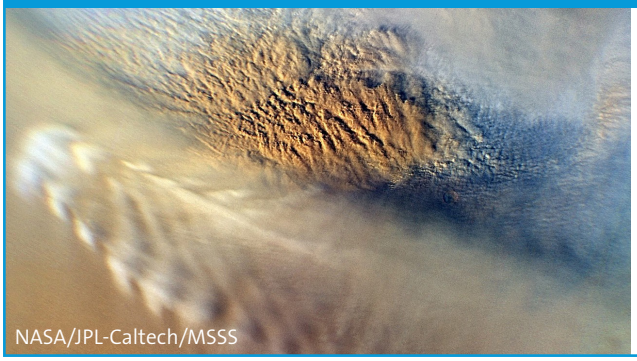
ESA/ATG medialab

Mars is so far away that signals take quite a long time to travel from the spacecraft or rover back to Earth. This delay in communication makes it challenging to have a conversation with the rover, or react quickly if anything unexpected occurs.

→ Activity 2: Blink a S.O.S.!

Now that you know how to blink an LED using Arduino code, this activity will teach you how to communicate using an LED and to send a message to a rover on Mars. In fact, a very strong sandy storm is planning to pass the rover! To prevent damage to the rover, make it send an S.O.S.!

Did you know?



This image of a dust storm on Mars shows the clouds that result from changes in atmospheric pressure, temperature and height because of vertical displacement, such as when wind blows over a mountain or crater wall.

Equipment

- Circuit built in Activity 1

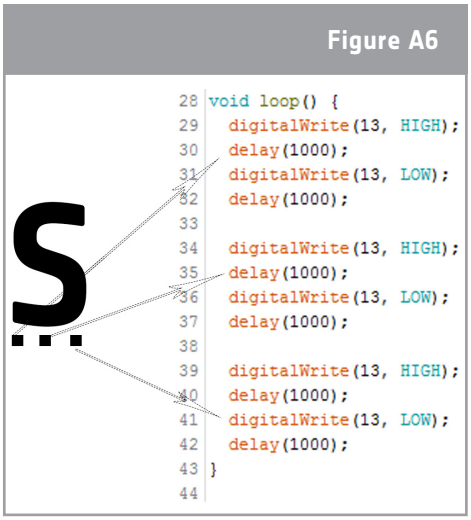
Exercise

Use Morse code to blink the green LED. In Figure A5, a point means a short signal, which will light up the LED for a short time, and a dash means a long signal, which will light up the LED for a longer time. The code to send an S is given as an example in Figure A6.

Figure A5

↑ SOS in Morse code

Figure A6



↑ Code to send a S in Morse

Explain what you need to change in the code to send an O:

Adapt the code for sending an S.O.S., then upload it to the Arduino and test it!

→ Activity 3: Measuring temperature

Arduino technology allows you to control an LED connected to an Arduino Uno using a code written on a laptop. When you upload the code, it sends the instructions to the Arduino Uno and activates the electronic components. In the ExoMars mission, technology will be used to investigate the environment of Mars thanks to some sensors. In this activity, you will discover how to measure the temperature of the classroom and simulate the temperature on Mars!

Equipment

- 1 Arduino Uno
- 1 breadboard
- 1 temperature sensor (thermistor)
- 1 10kΩ resistor (brown/black/black/red/brown)
- 3 wires

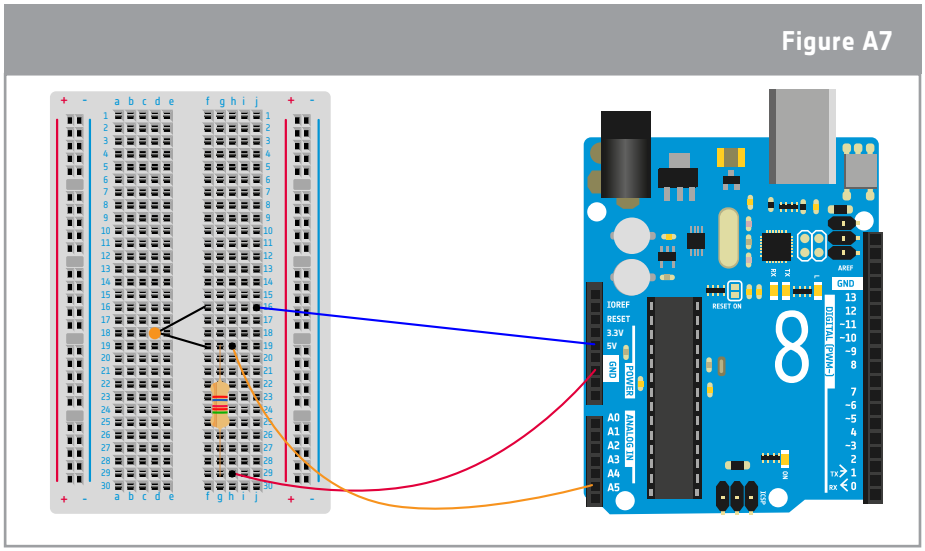
Exercise

1. To measure the temperature in the classroom, you need to build the circuit on the breadboard as shown below.

   Carefully follow these instructions:

   a. Connect the thermistor on the breadboard as shown in Figure A7.
   b. Connect the 10kΩ resistor (brown/black/black/red/brown) to the thermistor as shown in Figure A7.
   c. Using a wire, connect the thermistor to the **5V** pin of the Arduino Uno.
   d. Using a wire, connect the resistor to the **GND** pin of the Arduino Uno.
   e. Using a wire, connect the thermistor and the resistor to the pin **A5** of the Arduino.
   f. Connect the Arduino Uno to the computer using the USB cable.

Figure A7



↑ Arduino circuit ready to measure the temperature

2. Figure A8 illustrates a partly missing code to measure the temperature. Three values, shown as boxes (red, green and blue) need to be filled in to give correct instructions to the Arduino Uno in order to measure the temperature. Let's find out how to fill in this code.

Figure A8

```
1
2  void setup() {
3    Serial.begin(9600);
4  }
5
6  void loop() {
7
8    float [                    ];
9    float [                    ];
10
11       [                    ] = float(analogRead([  ]));
12       [                    ] = (-40000/[                    ])+100;
13
14   Serial.println();
15   Serial.print("Temperature in classroom: ");
16   Serial.print([                    ]);
17   Serial.print("C");
18   delay(1000);
19
20 }
```

↑ Arduino code for measuring the temperature

a. What unit is used to determine temperature in Europe?

The temperature sensor connected to the Arduino Uno will first read the temperature in another unit, the volt. To transform this number from volts to degrees Celsius, you will use a formula written on line 12 in Figure A8. Finally, you will display the temperature on the screen using the Serial.print function in the code.

b. Define the variable containing the temperature in volts: [ ]

Define the variable containing the temperature in degrees Celsius: [ ]

Look for the pin of the Arduino where the temperature sensor is connected: [ ]

c. You are ready to write the code on your laptop. Open the Arduino software and write the code in Figure A8, replacing the boxes with the variables you have defined for the red, green and blue boxes. Be careful to replace every colour box with its respective variable!

Upload the code to the Arduino Uno by clicking on the 

d. To see the temperature measurements, you need to open a specific window in the Arduino software called Serial monitor. Look for this  symbol on the toolbar to access it.

Discussion

1. Discuss the relevance of the temperature measurements with your peers. How can you verify that the temperature sensor accurately measures the temperature of the classroom?

2. Which line of the code in Figure A8 can you adapt to calibrate the sensor?

3. To practice a little more with the Arduino, calibrate the temperature sensor as if it were on the surface of Mars. Consider the average temperature of Mars' surface to be around -60°C.

   a. Average temperature of Earth = _____

   b. Write the new calibration formula of the temperature sensor here:

→ Activity 4: Measuring pressure

In this activity, you will learn how to measure the atmospheric pressure using the Arduino and a pressure sensor.

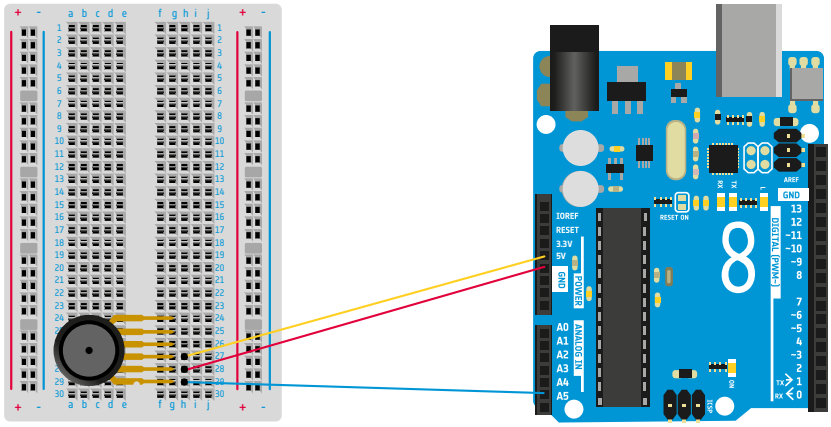
Equipment

- 1 Arduino Uno
- 1 breadboard
- 1 pressure sensor
- 3 wires

Exercise

1. To measure the pressure in the classroom, you first need to build the circuit on the breadboard (Figure A9). Carefully follow these instructions (Note: Each pin in the pressure sensor behaves differently, so make sure it is in the same orientation as in the figure):

Figure A9



↑ Arduino circuit ready to measure the pressure

2. Figure A10 illustrates a partly missing code to measure the pressure. Three variables, shown as boxes (red, green and blue), need to be filled in to give correct instructions to the Arduino Uno in order for it to measure the pressure. Let's find out how to fill in this code.

Figure A10

```
1  void setup() {
2
3    Serial.begin(9600);
4  }
5  void loop() {
6
7    float [          ] ;
8    float [          ] ;
9
10     [          ] = float(analogRead([ ]));
11     [          ] =(([          ] /1024)+0.095)/0.009;
12   Serial.println();
13   Serial.print("Pressure in classroom: ");
14   Serial.print([          ]);
15   Serial.print("kPa");
16   delay(1000);
17
18 }
```

↑ Arduino code for measuring the pressure

a. What is the physical unit used to determine the pressure in International System of Units?

---

b. Define the variable containing the pressure in volts:

Define the variable containing the pressure in kilopascal:

Look for the pin of the Arduino where the pressure sensor is connected:

The pressure sensor connected to the Arduino Uno will first read the pressure in another unit, the volt, which is the unit for electricity. To transform this number from volts to pascals, we use this formula from the datasheet of the sensor $V_{out} = V_S (P \times 0.009 - 0.095)$.

Finally, display the measurement of the pressure on the screen using the Serial.print function in the code.

c. Using the formula $V_{out} = V_S (P \times 0.009 - 0.095)$; calculate the calibration formula to complete line 12 in Figure A10. Because the voltage read is divided in 1024 steps, consider $V_S = 1024$.

---

d. You are ready to write the code on your laptop. Open the Arduino software and write the code in Figure A10, complete with the variables you have defined for the red, green and blue boxes. Be careful to replace every coloured box with its respective values!

Upload the code to the Arduino Uno by clicking on the 

e. To see the pressure measurements, you need to open a specific window in the Arduino software called Serial monitor. Look for this symbol on the toolbar to access it: 

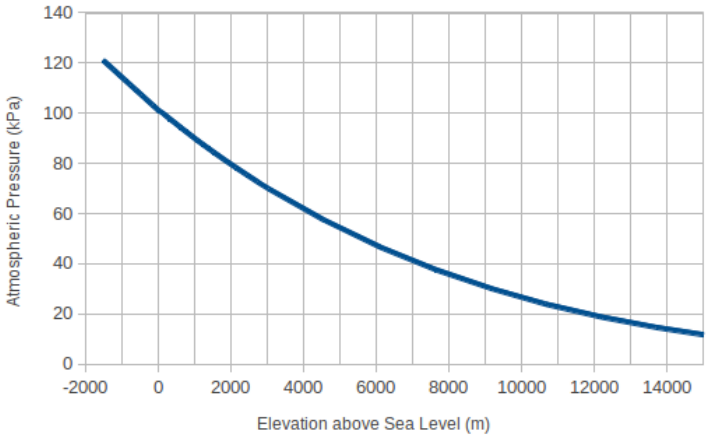
Discussion

Discuss the relevance of the pressure measurements with your peers. How can you verify that the pressure sensor accurately measures the pressure in the classroom?

Which line of the code in Figure A10 should you adapt to calibrate the sensor?

→ Activity 5: Measuring altitude

The pressure of the air varies with altitude. For example, the air pressure at sea is higher than the air pressure in the mountains. You can see how it varies in the graph. With a bit of maths and some understanding of the physics underpinning gas laws, we can calculate exactly what the relationship between pressure and altitude is. Right now, we don't need to worry about this. In this activity you will use your pressure measurements to determine your altitude using a simplified version of the pressure-altitude relationship formula known as the barometric formula.



Equipment

- Arduino circuit built in Activity 4
- Code to measure pressure in Activity 4

Exercise

1. Open the pressure code you have written in Activity 4 in the Arduino software. Add this partly missing code inside the void loop:

Figure A11

```
1  float                    ;
2  float                    ;
3
4                  = pow(                  /101.325,0.1903);
5                  =(1-                    )*44300+100;
6
7  Serial.println();
8  Serial.print("Altitude in classroom: ");
9  Serial.print(                );
10 Serial.print("meters");
11 delay(1000);
```

↑ Arduino code for measuring the altitude

2. To determine the altitude, you will use the measurements of the pressure sensor in a formula written on line 4 in Figure A11. As you discovered in the previous exercises, a sensor connected to the Arduino Uno will first read the measurement in volts. To transform this number from volts to metres, you will use a formula written on line 5 in Figure A11. Finally, altitude data can be displayed using the Serial.print function in the code. If the reading is not calibrated, it can be adjusted by changing line 5 of the code.

3. a. Define the variable containing the altitude in volts:

Define the variable containing the altitude in metres:

Look for the name you gave to the green box containing the pressure in pascal (see Activity 4):

b. Complete the code with the variables you defined for the yellow, purple and green boxes. Be careful to replace every coloured box with its respective value! Then upload the code.

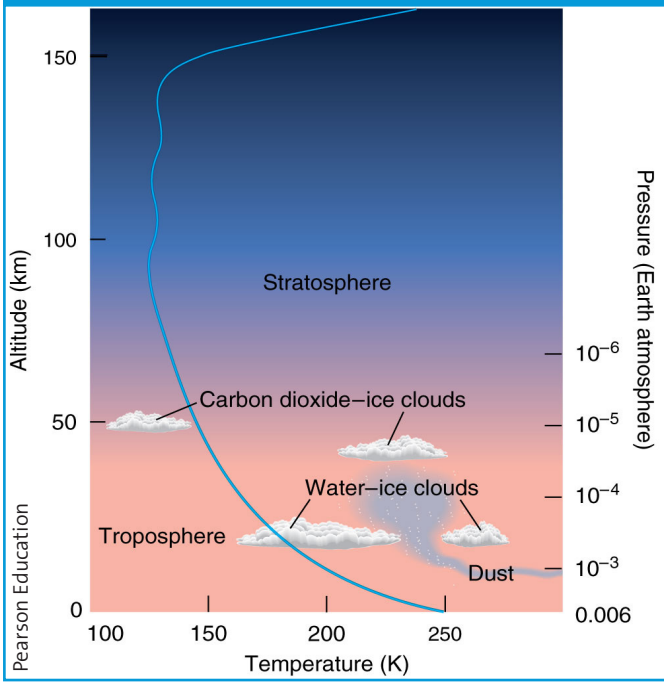
c. To display the altitude data, you need to open a specific window in the Arduino software called Serial monitor.

4. Combine both temperature and altitude code into a single sketch and upload it to the Arduino Uno.

Collect the temperature and altitude data from the floor to the ceiling of your classroom and then draw a graph displaying the results.

Can you explain the shape of the graph?

Did you know?



The Martian atmosphere is an extremely thin sheet of gas, principally carbon dioxide, that extends from the surface of Mars to the edge of space. The sun heats the surface of Mars and some of this heat goes into warming the gas near the surface. The heated gas then diffuses or convects up through the atmosphere. Thus, the gas temperature is highest near the surface and decreases as we increase altitude.

→ LINKS

Arduino blog, useful for finding the latest Arduino information:
https://blog.arduino.cc/

Guides to using various Arduino components:
https://quarkstream.wordpress.com/

Many Arduino based projects you can try at home can be found on Instructables:
http://www.instructables.com/howto/arduino/

HackADay hosts many other Arduino projects:
https://hackaday.io/list/3611-arduino-projects