Green STEAM Incubator

Handbook on Environmental Projects using Arduino

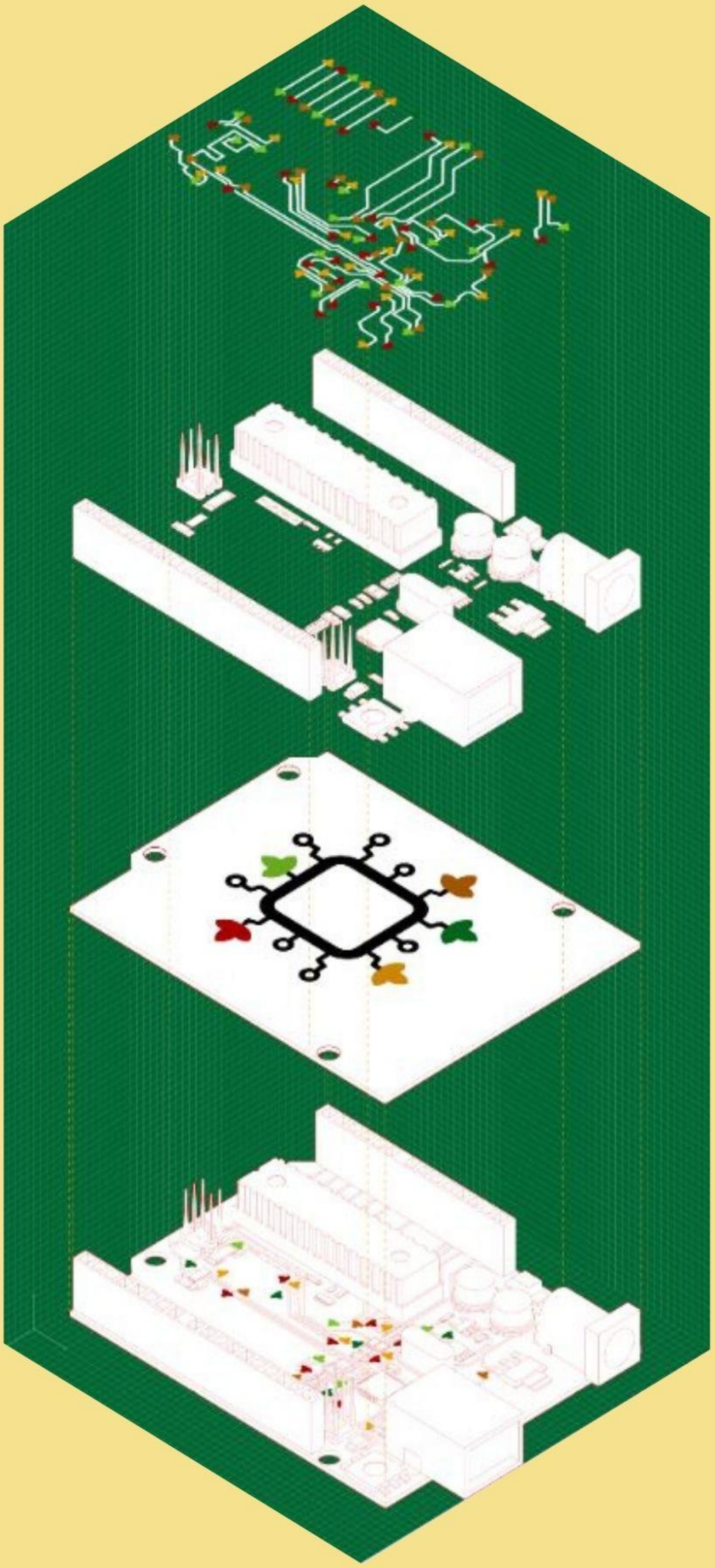


Table of Contents

INTRODUCTION

This handbook was created within the scope of Intellectual Output 3 of the Green STEAM Incubator project, which is related to Microcontrollers. The objective was to create a document with ideas for environmental projects that could be applied resorting to microcontrollers.

Every partner of the consortium of this project came up with different project ideas that were gathered in a single document. Each project has its own different objectives and learning results, but one thing is common to all of them: to explain the potential that microcontrollers have and the various ways they are useful in the field of agriculture.

Together, these project ideas serve as a 20-hour workshop to promote the aforementioned objectives. All the essential information to each session is given at the beginning of each project idea.

PROJECT: MOTION DETECTOR

- STEM field: Technology, Engineering
- Indicative calendar: Any time of the year
- Activity duration: 2h30 min
- Type of activity: Workshop
- Educational objectives: 1. Describe the usefulness of PIR Sensor; 2. Configure the PIR Sensor for motion detector; 3. Build the motion detector with PIR sensor
- Learning outcomes and acquired competences: By the end of the course the learners are expected to: 1. Characterize a PIR sensor, 2. Know how to connect a PIR sensor with Arduino, 3. Know how to program a simple code in Arduino IDE for motion detector use, 3. Upload their program in the Arduino Board, 5. Run the program to detect any moment in the room or around the motion sensor, 6. recognize the hardware needed to build a motion detector, 7. Determine the hardware to be used in different configurations with the PIR sensor,
- Required material and resources: Arduino UNO, Arduino IDE software, Computer, Internet connection, Passive Infrared Sensors (PIR) Motion Sensor (generic), LED (generic), Jumper wires (generic), USB-A to Micro-USB Cable, Solderless Breadboard Half Size.
- Description and/or step-by-step instructions

  The aim of this activity is to program a motion detector with the use of PIR motion sensor and Arduino. In this activity, the participants will learn how to connect PIR sensor with Arduino and program the Arduino to detect any moment in the room or around the motion sensor.

Motion detector

The aim of this activity is to program a motion detector with the use of PIR motion sensor and Arduino. In this activity, the participants will learn how to connect PIR sensor with Arduino and program the Arduino to detect any moment in the room or around the motion sensor.

Arduino UNO

As already introduced in the IO3 module, Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button etc - and turn it into an output - activating a motor, turning on an LED etc. With the use of the Arduino programming language and the Arduino Software (IDE) you can develop a program which in fact provides a set of instructions to the microcontroller on the board.

PIR Sensor

PIR sensor (see figure 1) detects a human being moving around within approximately 10m from the sensor. This is an average value, as the actual detection range is between 5m and 12m. PIR are fundamentally made of a pyro electric sensor, which can detect levels of infrared radiation. This sensor can be used in several projects when there is a need to discover when an individual has left or entered a certain area.

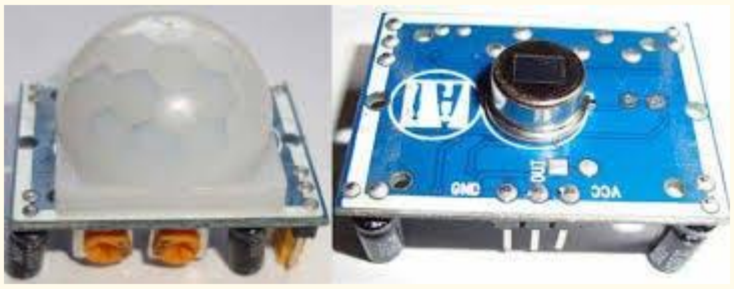


Figure 1 PIR Sensor, source: https://create.arduino.cc/projecthub/biharilifehacker/arduino-with-pir-motion-sensor-fd540a

Most PIR sensors have a 3-pin connection at the side or bottom. One pin will be ground, another will be signal, and the last pin will be the power. Power is usually up to 5V. Interfacing PIR with a microcontroller is very easy and simple. The PIR acts as a digital output so all you need to do is listening for the pin to flip high or low. The motion can be detected by checking for a high signal on a single I/O pin. Once the sensor warms up the output will remain low until there is motion, at which time the output will swing high for a couple of seconds, then return low. If motion continues the output will cycle in this manner until the sensors line of sight of still again. The PIR sensor needs a warm-up time with a specific end goal to capacity fittingly. This is because of the settling time included in studying nature's domain. This could be anyplace from 10-60 seconds.

Description of the project

The project consists of carrying out, in a simple way, a motion detector, with the use of a PIR motion sensor and its visualization occurs in a LED.

Circuit:

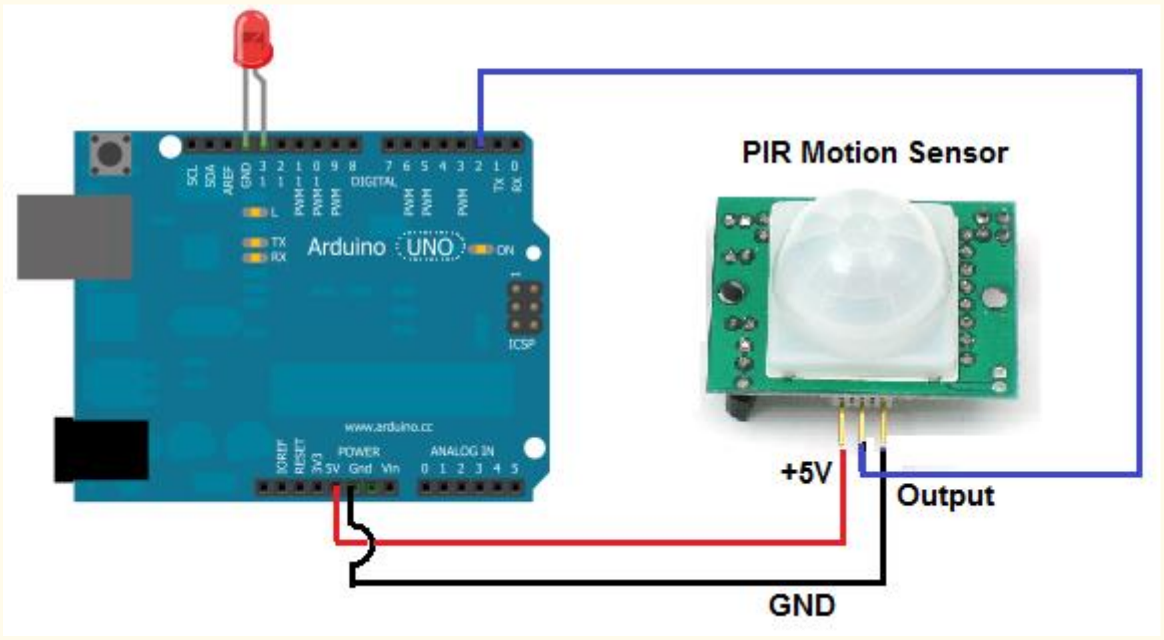


Figure 2 PIR Motion sensor with Led, source:
https://create.arduino.cc/projecthub/biharilifehacker/arduino-with-pir-motion-sensor-fd540a

Connections between pins:

- GND Arduino –> GND PIR Motion Sensor
- GND Arduino –> GND of the LED
- 5V Arduino –> 5V PIR Motion Sensor
- D2 Arduino –> Outpute PIR Motion Sensor

Programming/Coding:

```
int led = 13;                // the pin that the LED is atteched to
int sensor = 2;              // the pin that the sensor is atteched
to
int state = LOW;             // by default, no motion detected
int val = 0;                 // variable to store the sensor status
(value)

void setup() {
  pinMode(led, OUTPUT);      // initalize LED as an output
  pinMode(sensor, INPUT);    // initialize sensor as an input
  Serial.begin(9600);        // initialize serial
}

void loop(){
  val = digitalRead(sensor);   // read sensor value
  if (val == HIGH) {            // check if the sensor is HIGH
    digitalWrite(led, HIGH);    // turn LED ON
    delay(500);                 // delay 100 milliseconds

    if (state == LOW) {
      Serial.println("Motion detected!");
      state = HIGH;       // update variable state to HIGH
    }
  }
  else {
      digitalWrite(led, LOW); // turn LED OFF
      delay(500);             // delay 200 milliseconds

      if (state == HIGH){
        Serial.println("Motion stopped!");
        state = LOW;       // update variable state to LOW
    }
  }
}
```

- References:

Arduino (July 23, 2020). *Arduino with PIR Motion Sensor*. Retrieved January 2021, from: https://create.arduino.cc/projecthub/biharilifehacker/arduino-with-pir-motion-sensor-fd540a

PROJECT: ALARM SYSTEM FOR THE HENHOUSE DOOR AT NIGHT

- STEM field: Science, technology, and electronics.

- Indicative calendar: Any time of the year.

- Activity duration: 2h30.

- Type of activity: Develop installations using microcontrollers.

- Educational objectives: 1. To learn how to build useful tools for the farm using the Arduino model and electronic component systems, 2. Describe the usefulness of Ultrasinic Sensor, 3. Configure the Ultrasonic Sensor to work with Arduino, 4. Buils the system with Ultrasonic Sensor.

- Learning outcomes and acquired competences:
  - Learn how to use Arduino UNO to install an utra sound alarm system to prevent predators from entering the henhouse;
  - Characterize Ultrasonic Sensor;
  - Recognize the hardware needed to build a system with Ultrasonic Sensor and Buzzer;
  - Determine the hardware to be used for reading information and characteristics to Ultrasonic Sensor;
  - Program and configure the Ultrasonic Sensor with a Buzzer;
  - Plan and structure tasks;
  - Act with initiative and demonstrate analytical capacity.

- Required material and resources:
  - Arduino UNO R3
  - Small Active Buzzer B10
  - Ultrasonic Sensor - HC-SR04 (Generic)
  - Jumper wires (generic) Female-Female and Male-Female (Pack of 10)
  - USB Cable Arduino (A to B) and A 9 Volt battery
  - Mini Breadboard 170 pins
  - Computer and Arduino IDE software
  - Internet connection
  - Cable USB to Micro USB

- Description and/or step-by-step instructions
  In this project we will use a motion detector that will trigger the buzzer: an alarm of the type that is supposed to deter and frighten predators.

Green STEAM Incubator

The European Commission support for the production of this document does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

According to the code stored in Artduino Uno, the motion detector will activate the buzzer for a few seconds.

System with Ultrasonic Sensor

The main objective of this project is to install an alarm in a poultry house to prevent predators from entering the house.
In this project we will use a motion detector that will trigger the buzzer: an alarm of the type that is supposed to deter and frighten predators

Introduction

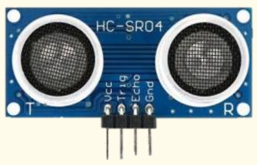
What we will need in this project:

A buzzer :

Source : Image

It is basically a tiny speaker that you can connect directly to an Arduino. You can make it sound a tone at a frequency you set. The buzzer produces sound based on reverse of the piezoelectric effect.

Ultrasonic Sensor - HC-SR04 (Generic)



Source: Image

Ultrasonic Sensor is a sensor that can measure distance. It emits
an ultrasound at 40 000 Hz (40kHz) which travels through the air and if there is an object or obstacle on its path It will bounce back to the module.

Step 1 Connect place your elements on the assembly breadboard and connect them with wires as it is shown in Figure 1:

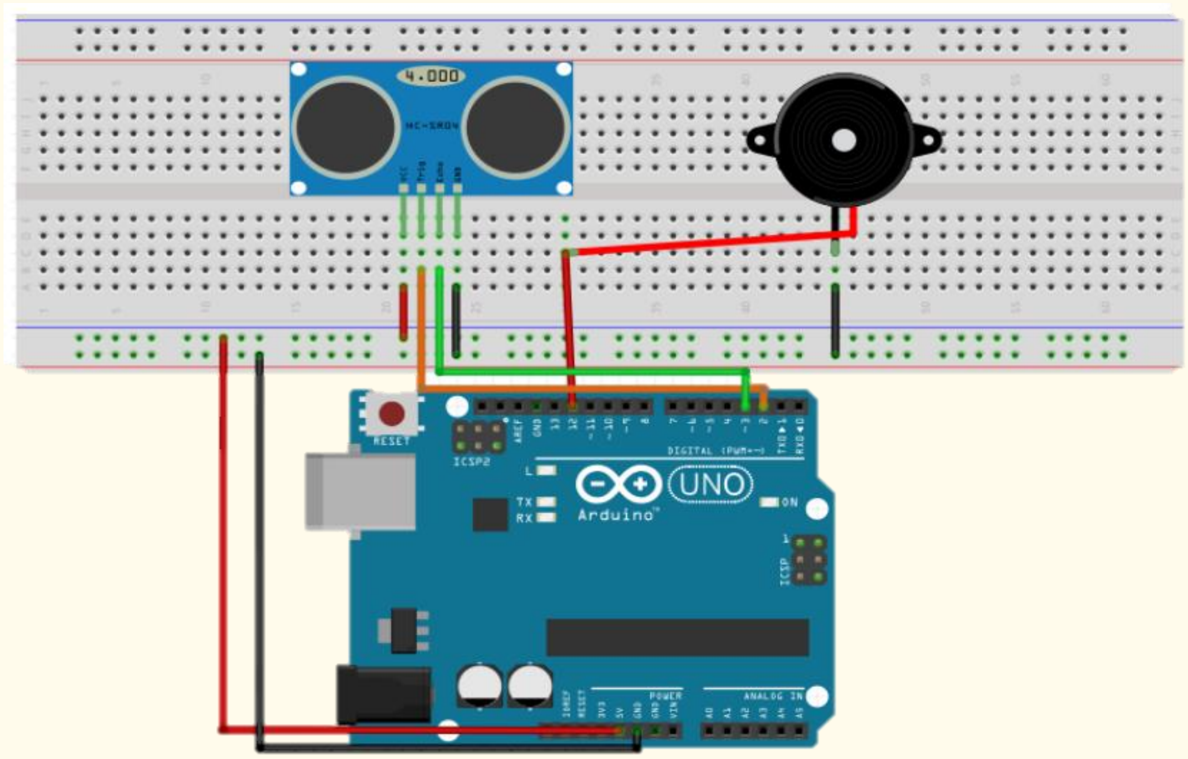


Figure 2 Connecting Diagram

Buzzer

Place the buzzer on the assembly breadboard and then use the wires to connect the positive terminal with PIN 12 and negative terminal to pin GROUND (GDN).

Ultrasonic Sensor

It is recommended to place the ultrasonic sensor as far right to the breadboard as possible and make sure it is facing out. Remember that the sensor is supposed to detect the opening and closing movements of the henhouse door.

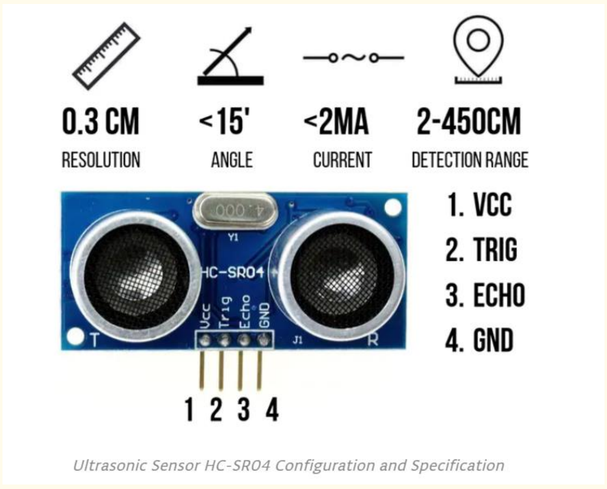


Image: Source

The ultrasonic sensor has 4 terminals:

The GND pin has to be connected to the negative channel of the breadboard.
The Trig pin to the digital output pin 2 on the Arduino.
The Echo Pin has to be connected to pin 3 on the Arduino.
The VCC pin needs to be plugged to the positive channel of the assembly breadboard and then connected to the Arduino board pin 5Volt.

Step 2

Plugin your Arduino. Connect your Arduino to the USB port of your computer:

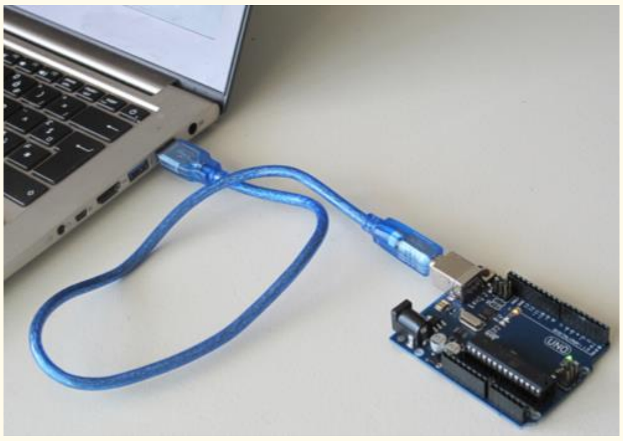


Figure 2 Connect your Arduino to the USB port of your computer. Source: www.getready.io/arduino

Step 3

Open Arduino IDE and select the Arduino Uno board:

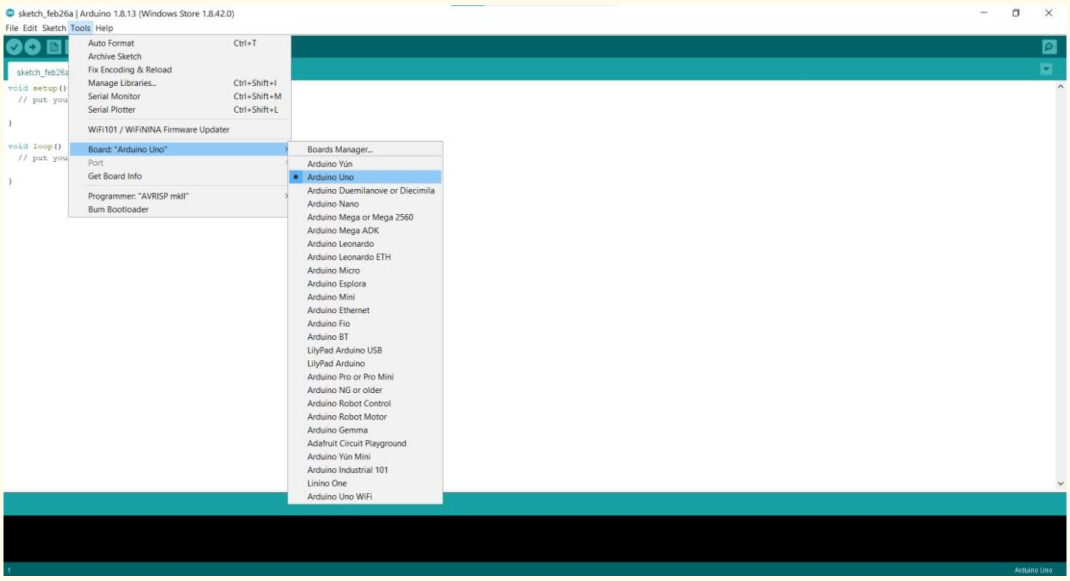


Figure 3 Choose the right board

Step 4 Open a new file:

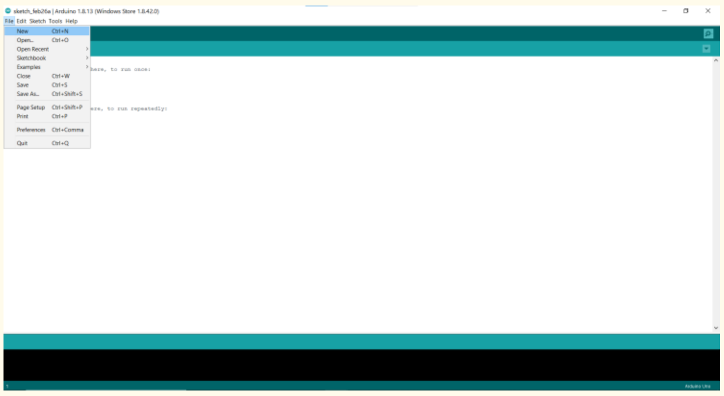
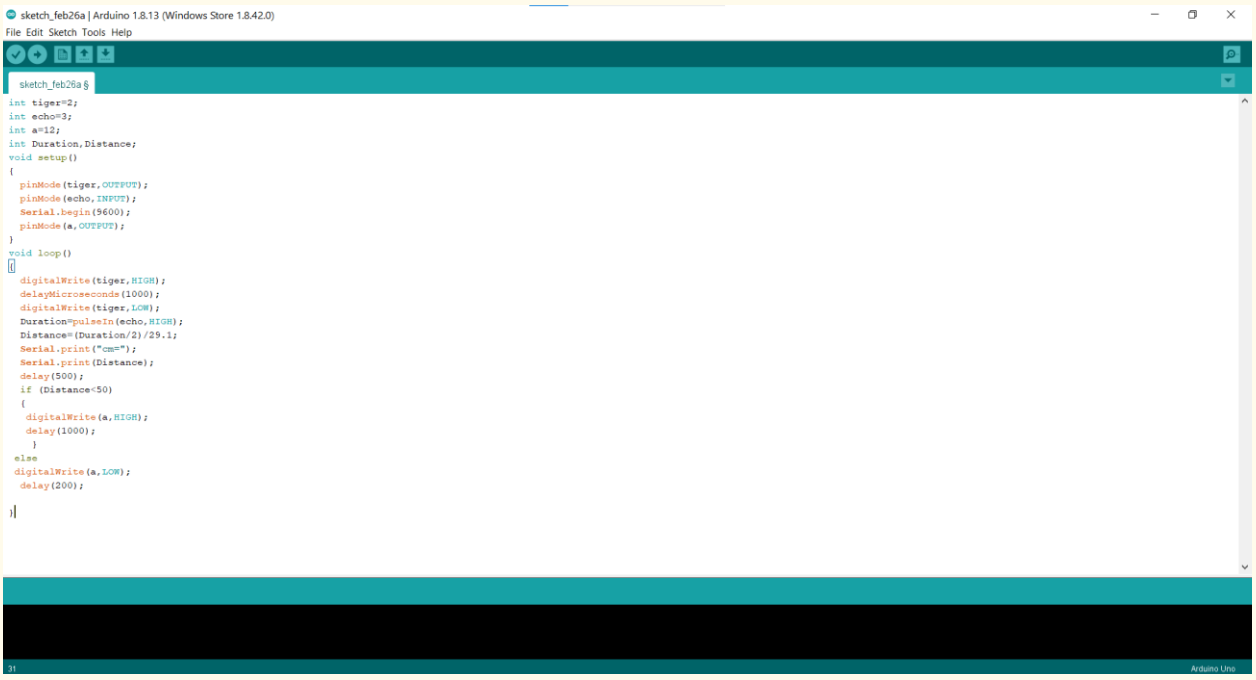


Figure 4 Open a new example file

Step 5 Copy and paste the following code to your project:

```
int tiger=2;
int echo=3;
int a=12;
int Duration,Distance;
void setup()
{
  pinMode(tiger,OUTPUT);
  pinMode(echo,INPUT);
  Serial.begin(9600);
  pinMode(a,OUTPUT);
}
void loop()
{
  digitalWrite(tiger,HIGH);
  delayMicroseconds(1000);
  digitalWrite(tiger,LOW);
  Duration=pulseIn(echo,HIGH);
  Distance=(Duration/2)/29.1;
  Serial.print("cm=");
  Serial.print(Distance);
  delay(500);
  if (Distance<50)
  {
   digitalWrite(a,HIGH);
   delay(1000);
    }
 else
 digitalWrite(a,LOW);
  delay(200);

}
```


sketch_feb26a | Arduino 1.8.13 (Windows Store 1.8.42.0)

File Edit Sketch Tools Help

sketch_feb26a §

```
int tiger=2;
int echo=3;
int a=12;
int Duration,Distance;
void setup()
{
  pinMode(tiger,OUTPUT);
  pinMode(echo,INPUT);
  Serial.begin(9600);
  pinMode(a,OUTPUT);
}
void loop()
{
  digitalWrite(tiger,HIGH);
  delayMicroseconds(1000);
  digitalWrite(tiger,LOW);
  Duration=pulseIn(echo,HIGH);
  Distance=(Duration/2)/29.1;
  Serial.print("cm=");
  Serial.print(Distance);
  delay(500);
  if (Distance<50)
  {
   digitalWrite(a,HIGH);
   delay(1000);
    }
 else
 digitalWrite(a,LOW);
  delay(200);

}
```

Figure 5 Copy and paste the code in your project

Step 6 Select the serial device of the board from the Tools |

Step 7 Upload the program

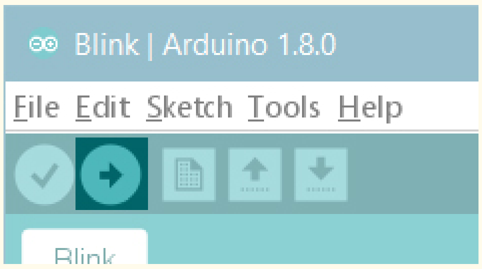


Figure 6 Upload the program

- **Bibliographic references:**

(Source: https://create.arduino.cc/projecthub/robodia-technology-solutions/entry-level-door-monitoring-alarm-system-3474e3

PROJECT: ANIMAL’S WATER CONTAINER ALARM

- STEM field: Science, technology, and electronics.
- Indicative calendar: Any time of the year.
- Activity duration: 2.5 hours.
- Type of activity: Use Arduino Uno to build an alarm system that detects when the water tank is empty.
- **Educational objectives:**
  - Describe the usefulness of Water Level Sensor;
  - Configure the Water Level Sensor to work with Arduino;
  - Build the system with Water Level Sensor;
  - Characterize water level sensor;
  - Recognize the hardware needed to build a system with Water Level Sensor and Buzzer;
  - Determine the hardware to be used for reading information and characteristics to Water Level Sensor;
  - Program and configure the Water Level Sensor with a Buzzer;
  - Plan and structure tasks;
  - Act with initiative and demonstrate analytical capacity.
- Required material and resources:
  - Arduino UNO;
  - USB Cable Arduino (A to B) and a 9 Volt battery;
  - A Water level sensor;
  - Jumper wires (generic) Female-Female and Male-Female (Pack of 10);
  - A small active buzzer;
  - Computer and Arduino IDE;
  - Internet connection;
  - Cable USB to microUSB;
  - Breadboard.
- Description and/or step-by-step instructions
  For this project we want to install a water level sensor so that we can know when the water tank for the animals reaches a critical level. When the water

level reaches a certain predefined level, an alarm will be triggered to inform about the need to refill the water tank for the animals.

System with water level Sensor

For this project we want to install a water level sensor so that we can know when the water tank for the animals reaches a critical level. When the water level reaches a certain predefined level, an alarm will be triggered to inform about the need to refill the water tank for the animals.

What we will need in this project:

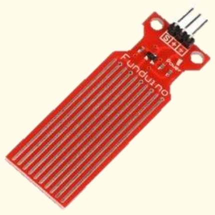
Introduction

A buzzer :

Source : Image

It is basically a tiny speaker that you can connect directly to an Arduino. You can make it sound a tone at a frequency you set. The buzzer produces sound based on reverse of the piezoelectric effect.

A water level sensor



Source: Image

This is a conductive-type water level sensor, where the change in resistance of parallel wires over varying depths of water is converted to voltage. This is not ideal for high precision water level monitoring and is only suitable for hobby projects.

The module has three pins: + (5V), - (GND), and S (Signal). The S pin outputs voltage corresponding to water level.

Step 1 Place your elements on the Arduino Uno board and connect them with wires as it is shown in Figure 1:

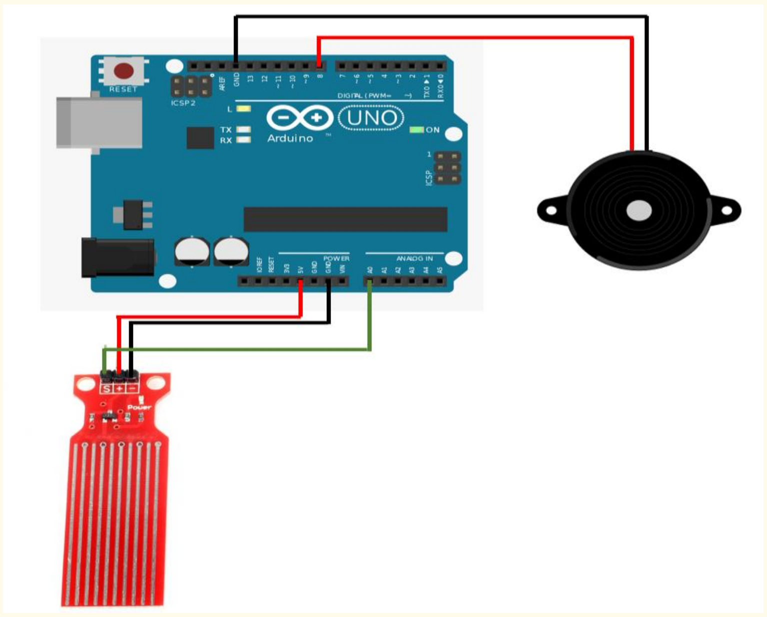


Figure 3 Connecting Diagram

Connect the water level sensor

Use the red wires to connect the positive terminal (+) with 5V
Use the black wire to connect the negative terminal (-) to pin GROUND (GDN)
Use the green wire to connect the sensor (S) to pin A0

Connect the buzzer

Use the red wires to connect the positive terminal (+) with pin 8
Use the black wire to connect the negative terminal (-) to pin GROUND (GDN)

Co-funded by the Erasmus+ Programme of the European Union

Step 2

Plugin your Arduino. Connect your Arduino to the USB port of your computer:

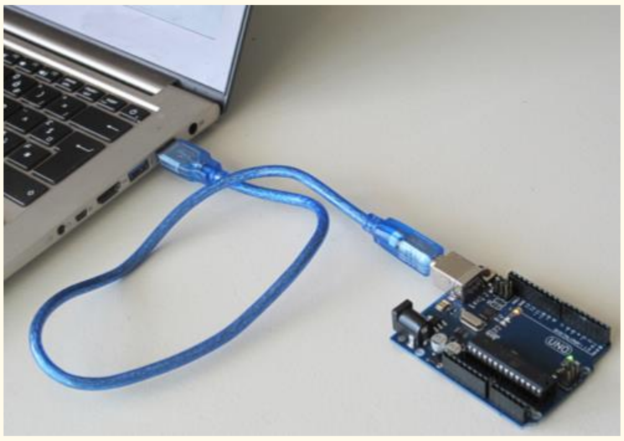


Figure 2 Connect your Arduino to the USB port of your computer. Source: www.getready.io/arduino

Step 3

Open Arduino IDE and select the Arduino Uno board:

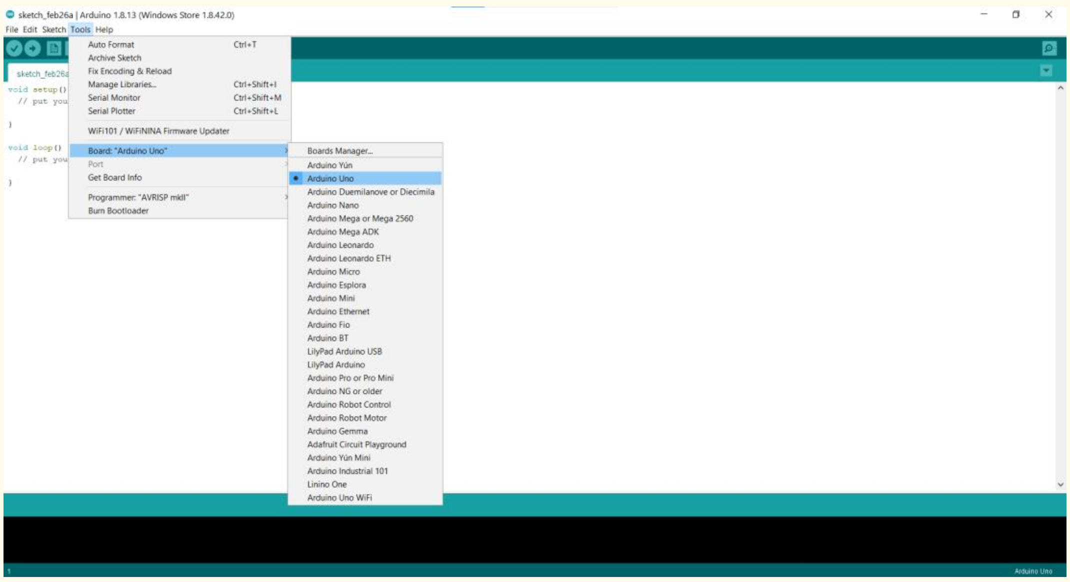
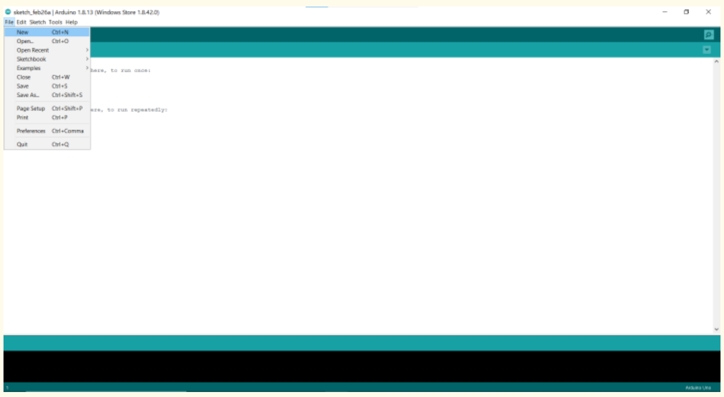


Figure 3 Choose the right board

Step 4 Open a new file:

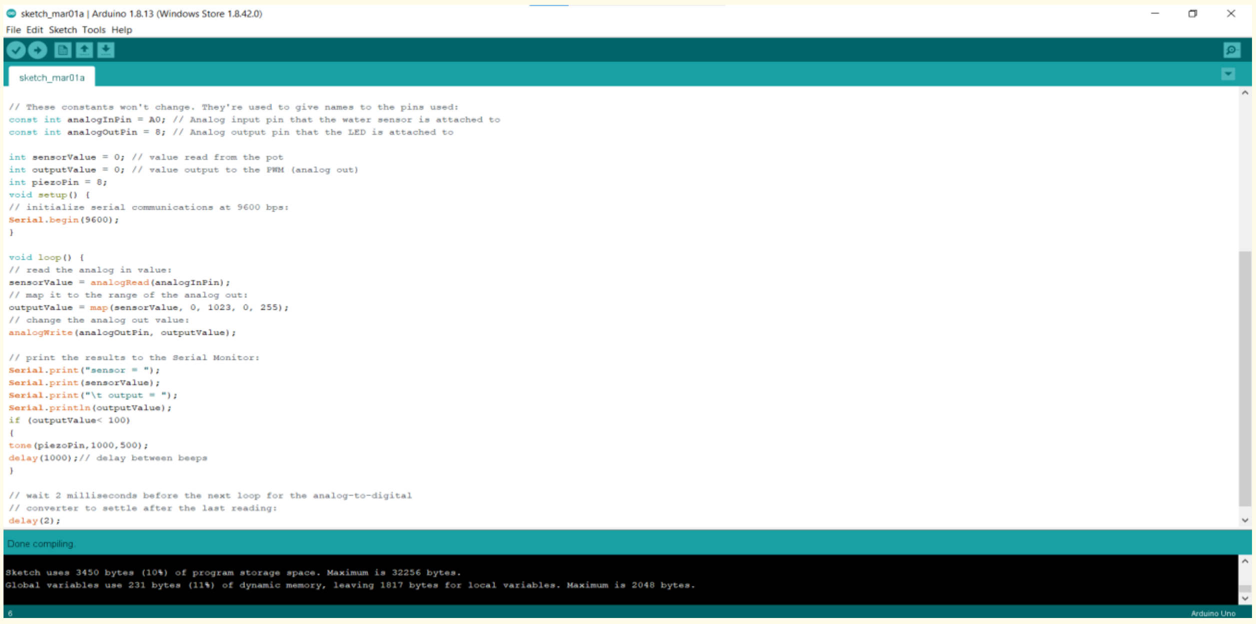


```
here, to run once:

ere, to run repeatedly:
```

Figure 4 Open a new example file

Step 5 Copy and paste the following code to your project:

```
// These constants won't change. They're used to give names to the
pins used:
const int analogInPin = A0; // Analog input pin that the water
sensor is attached to
const int analogOutPin = 8; // Analog output pin that the LED is
attached to

int sensorValue = 0; // value read from the pot
int outputValue = 0; // value output to the PWM (analog out)
int piezoPin = 8;
void setup() {
// initialize serial communications at 9600 bps:
Serial.begin(9600);
}

void loop() {
// read the analog in value:
sensorValue = analogRead(analogInPin);
// map it to the range of the analog out:
outputValue = map(sensorValue, 0, 1023, 0, 255);
// change the analog out value:
analogWrite(analogOutPin, outputValue);

// print the results to the Serial Monitor:
Serial.print("sensor = ");
Serial.print(sensorValue);
Serial.print("\t output = ");
Serial.println(outputValue);
if (outputValue< 100)
{
tone(piezoPin,1000,500);
delay(1000);// delay between beeps
}

// wait 2 milliseconds before the next loop for the analog-to-
digital
// converter to settle after the last reading:
delay(2);
}

/*
Analog input, analog output, serial output

Reads an analog input pin, maps the result to a range from 0 to 255
and uses
the result to set the pulse width modulation (PWM) of an output pin.
Also prints the results to the Serial Monitor.

The circuit:
- water level sensor connected to analog pin 0.
Center pin of the water sensor goes to the analog pin.
side pins of the water sensor go to +5V and ground
- Buzzer connected from digital pin 8 to ground

created 29 Dec. 2008
modified 9 Apr 2012
by Tom Igoe

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/AnalogInOutSerial
*/
```


sketch_mar01a | Arduino 1.8.13 (Windows Store 1.8.42.0)
File Edit Sketch Tools Help
sketch_mar01a

```
// These constants won't change. They're used to give names to the pins used:
const int analogInPin = A0; // Analog input pin that the water sensor is attached to
const int analogOutPin = 8; // Analog output pin that the LED is attached to

int sensorValue = 0; // value read from the pot
int outputValue = 0; // value output to the PWM (analog out)
int piezoPin = 8;
void setup() {
// initialize serial communications at 9600 bps:
Serial.begin(9600);
}

void loop() {
// read the analog in value:
sensorValue = analogRead(analogInPin);
// map it to the range of the analog out:
outputValue = map(sensorValue, 0, 1023, 0, 255);
// change the analog out value:
analogWrite(analogOutPin, outputValue);

// print the results to the Serial Monitor:
Serial.print("sensor = ");
Serial.print(sensorValue);
Serial.print("\t output = ");
Serial.println(outputValue);
if (outputValue< 100)
{
tone(piezoPin,1000,500);
delay(1000);// delay between beeps
}

// wait 2 milliseconds before the next loop for the analog-to-digital
// converter to settle after the last reading:
delay(2);
```

Done compiling.

```
Sketch uses 3450 bytes (10%) of program storage space. Maximum is 32256 bytes.
Global variables use 231 bytes (11%) of dynamic memory, leaving 1817 bytes for local variables. Maximum is 2048 bytes.
```

Figure 5 Copy and paste the code in your project

Step 6 Select the serial device of the board from the Tools |

Step 7 Upload the program

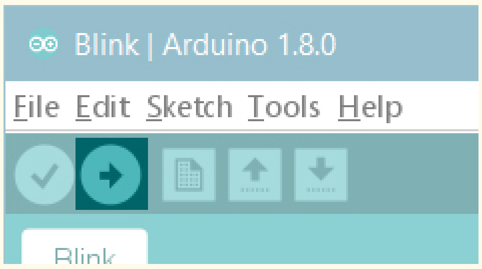


Figure 6 Upload the program

- **Bibliographic references:**

Source: https://talkingstuff.net/arduino-water-level-code/

PROJECT: OPERATING A HUMIDITY SENSOR

- STEM field: Science, technology and electronics.
- Indicative calendar: Any time of the year.
- Activity duration: 2h
- Type of activity: Use Arduino to operate a humidity sensor.
- Educational objectives
  - Know how to operate a humidity sensor;
  - Be able to work with Arduino;
- Learning outcomes and acquired competences
  - Have a good understanding of how a humidity sensor works;
  - Be able to operate a humidity sensor without help;
  - Have a god knowledge of Arduino.
- Required material and resources:
  - 1 Arduino Nano or Uno
  - 1 soil moisture sensor
  - 1 LED VM 3mm or 5mm
  - 1 LED VD 3mm or 5mm
  - 1 LED AM 3mm or 5mm
  - 1 Protoboard (contact matrix)
  - 3 carbon film resistors 1k Ohms 1/4W +/-5%
  - Cables
- Description and/or step-by-step instructions

Monitoring the amount of water in the soil can cause a huge difference in the agricultural productivity, due to the fact that it makes possible, in a more precise way, to know the amount of water present in the soil, enabling the regulation of an irrigation system. This way, it is possible to develop an intelligent system, making it more assertive when it comes to control and resorting to the information coming from the sensor.

We can also cross the system information with the climate characteristics of the region, so we can know beforehand the information about rain, and predict dates for maintenance, for the use of pesticides and even for crops.

According to information from the United Nations, more than 70% of drinking water in the world is used in irrigation systems, which is why there should be awareness of how to use it.

Co-funded by the Erasmus+ Programme of the European Union

A moisture sensor is an equipment that can help controlling the amount of water in the soil. The system that makes use of the moisture sensor will enable the measuring of the humidity, even if the soil is apparently dry, allowing for a more precise information and, that way, it can regulate water consumption, decrease the consumption of light coming originated from the activation of pumps system for irrigation purposes and regulate types of cultivation according to the moisture needs of a certain plantation.

Operation of the moisture sensor

Usually, the moisture sensor features two probes that measure the water volume in the soil. The probes originate an electric current that makes it possible to measure the resistance. The soil moisture value will be calculated from the resistance value, that varies from 0 to 1.023 (scale used in the microcontroller). It is important to point out that the greater the resistance detected, the less the electricity and the less the amount of water in the soil.

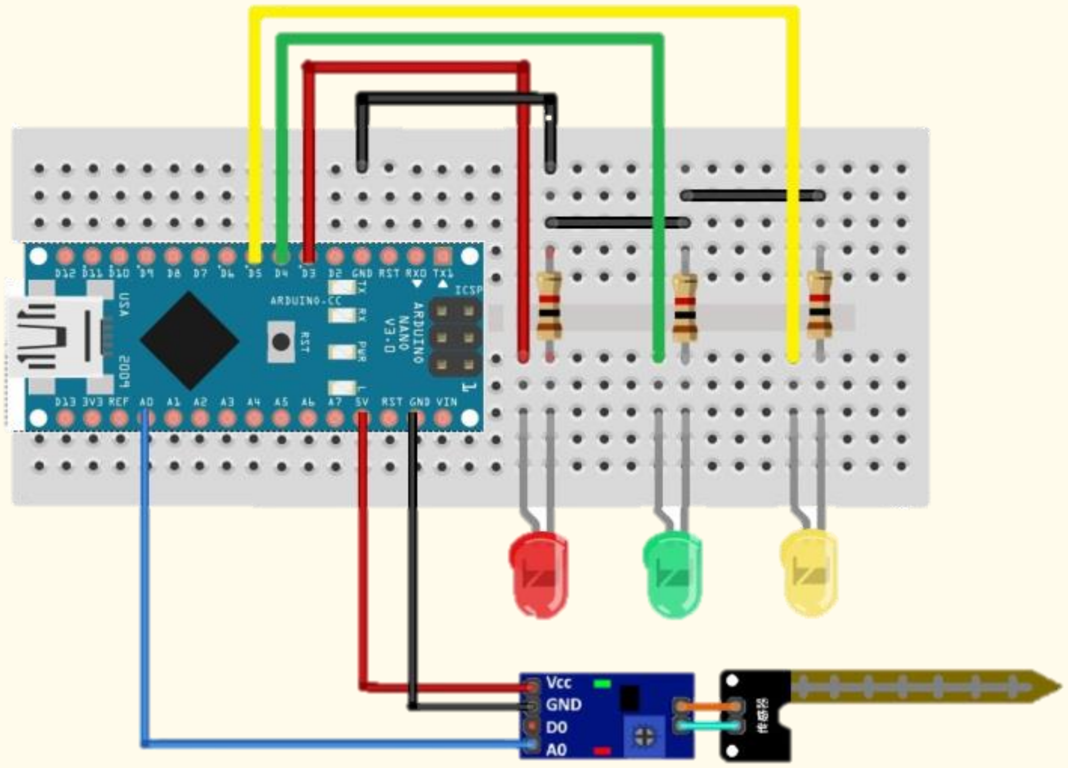
The use of this type of moisture sensor with the connection to a microcontrolled intelligent system will allow the obtainment of good results within the context of water control and, consequently, of agricultural production control, as well as energy saving.

Description of the project

The project consists of carrying out, in a simple way, a monitoring of the soil humidity, and its visualization occurs in three LEDs:

- Red LED indicates the soil is dry and needs to be moistened;
- Green LED indicates the soil has good moisture levels for most plants;
- Yellow LED indicates the soil is moist and that it can be harmful to some plants.

Circuit:



Connections between pins:

- GND Arduino –> GND moisture sensor –> GND of the 3 LEDs
- 5V Arduino –> VCC moisture sensor
- A0 Arduino –> A0 moisture sensor (analog input)
- D3 Arduino –> anode of the LED VM (Red LED)
- D4 Arduino –> anode of the LED VD (Green LED)
- D5 Arduino –> anode of the LED AM (Yellow LED)

Programming/Coding:

```
#define sensor A0                    // definition of the name of the variable "sensor" in the analog pin A0 of Arduino
#define LED_VM 3                     // definition of the name of the variable "LED_VM" in the digital pin 3 of Arduino
#define LED_VD 4                     // definition of the name of the variable "LED_VD" in the digital pin 4 of Arduino
#define LED_AM 5                     // definition of the name of the variable "LED_AM" in the digital pin 5 do Arduino

int sinal;                           // definition of the name of the internal variable "sinal" (signal) of the type whole

void setup()                         // Calls the setup() function, which is one of the main functions of Arduino
{                                    // Opening of the setup() function
  Serial.begin(9600);                // Initializes the serial monitor and defines the data transmission rate
  pinMode(sensor, INPUT);            // Definition of the pin of the variable sensor as input
  pinMode(LED_VM, OUTPUT);           // Definition of the pin of the variable LED_VM as output
  pinMode(LED_VD, OUTPUT);           // Definition of the pin of the variable LED_VD as output
  pinMode(LED_AM, OUTPUT);           // Definition of the pin of the variable LED_AM as output
}                                    // Closing of the setup() function

void loop()                         // Calls the loop() function, which is one of the main functions of Arduino (repetition
loop)
{                                   // Opening of the loop() function
  sinal = analogRead(sensor);       // Variable signal receives the value of the analog reading of the analog variable sensor
connected to pin A0
  Serial.print("Sinal: ");          // Requests that the text "Sinal:" is printed in the serial monitor
  Serial.print(sinal);              // Requests that the stored value in variable "sinal" is printed in the serial monitor

                                               // Humid soil condition --> enables the variable LED_AM
  if (sinal > 0 && sinal < 500)               // Calls conditional function if and checks if the variable signal is more than
0 and less than 500
```

```
  {                                            // Opening of the actions that must be carried out if the conditional function
if is true
    Serial.println(" Status: Solo humido");    // Requests that the text ”Status: Solo humido” (humid soil) is printed in the
serial monitor
    apagar();                                  // Calls the delete function to erase all LEDs
    digitalWrite(LED_AM, HIGH);                // Puts the digital variable LED_AM in high logic level, meaning, lights up
green LED
  }                                            // Closing of the conditional function if after the actions are carried out

                                               // Soil condition with moderate humidity --> enables the variable LED-VD
  if (sinal >= 500 && sinal < 800)             // Calls conditional function if and checks if the variable signal is equal or
more than 500 and less than 800
  {                                            // Opening of the actions that must be carried out if the conditional function
if is true
    Serial.println(" Status: humidade ok");    // Requests that the text “Status: humidade ok” (humidity ok)is printed in the
serial monitor
    apagar();                                  // Calls the delete function to erase all LEDs
    digitalWrite(LED_VD, HIGH);                // Puts the digital variable LED_VD in high logic level, meaning, lights up
yellow LED
  }                                           // Closing of the conditional function if after the actions are carried out

                                             // Soil condition with dry --> enables the variable LED-VM
  if (sinal >= 800 && sinal < 1024)          // Calls the conditional function if and checks if the variable signal is equal or
more than 800 and less than 1024
  {                                          // Opening of the actions that must be carried out if the conditional function if
is true
    Serial.println(" Status: Solo seco");    // Solicita que seja impresso no monitor serial o texto " Status: Solo seco"
    apagar();                                // Calls the delete function to erase all LEDs
    digitalWrite(LED_VM, HIGH);              // Puts the digital variable LED_VM in high logic level, meaning, lights up red
LED
  }                                          // Closing of the conditional function if after the actions are carried out
  delay(1000);                               // Implements a delay of 1000 ms so the reading does not stay continuous
unnecessarily
}                                            // Closing of the loop function

void apagar()                                // Calls the delete function
  {                                          // Opening of the actions that must be carried out when the delete function is
performed
  digitalWrite(LED_VM, LOW);                 // Puts the digital variable LED_VM in low logic level, meaning, lights up red LED
  digitalWrite(LED_VD, LOW);                 // Puts the digital variable LED_VD in low logic level, meaning, lights up green
LED
  digitalWrite(LED_AM, LOW);                 // Puts the digital variable LED_AM in low logic level, meaning, lights up yellow
LED
    }                                        // Closing of the delete function
```

Co-funded by the Erasmus+ Programme of the European Union

PROJECT: OPERATING A WEATHER STATION

- STEM field: Science, technology and electronics.
- Indicative calendar: Any time of the year.
- Activity duration: 2h30.
- Type of activity: Operate a weather station using Arduino.
- Educational objectives
  - Work with Arduino equipment;
  - Operate a weather station.
- Learning outcomes and acquired competences
  - Be able to operate a weather station without help;
  - Have a good knowledge of how Arduino functions.
- Required material and resources:
  - 1 Arduino Nano or Uno
  - 1 weather station
  - 1 Protoboard (contact matrix)
  - 1 resistance 10k Ohms
  - Cables
  - https://www.electrofun.pt/domotica/estacao-metereologica?fbclid=IwAR0teA_XllQk9CIQfYpV9MEAwyE2O1VRgjIJZ43Uu9438lHiMJE4QdpQgG4
  - http://cta.if.ufrgs.br/projects/estacao-meteorologica-modular/wiki/Anem%C3%B4metro
- Description and/or step-by-step instructions

The objective of developing a weather station is to provide a measuring of characteristics of the surrounding environment, namely, the measuring of the wind speed and direction through and anemometer in conjunction with a microcontroller. This equipment has in its interior a reed-switch (a kind of switch), which contains two small iron plates separated, and when they are together they send a signal to the microcontroller, and, that way, it can measure the wind speed. With this system, it is also possible to determine the direction of the wind through a voltage divider. The wind direction sensor has 8 switches, of which 4 are pointing to the cardinal points and 4 are pointing to the collateral points of the wind rose. Each of the 8 switches has an exact value resistance for each direction and the microcontroller will read values between 0 and 1023 in the analog pin, which means, each direction has a value between 0 and 1023 without repetition.

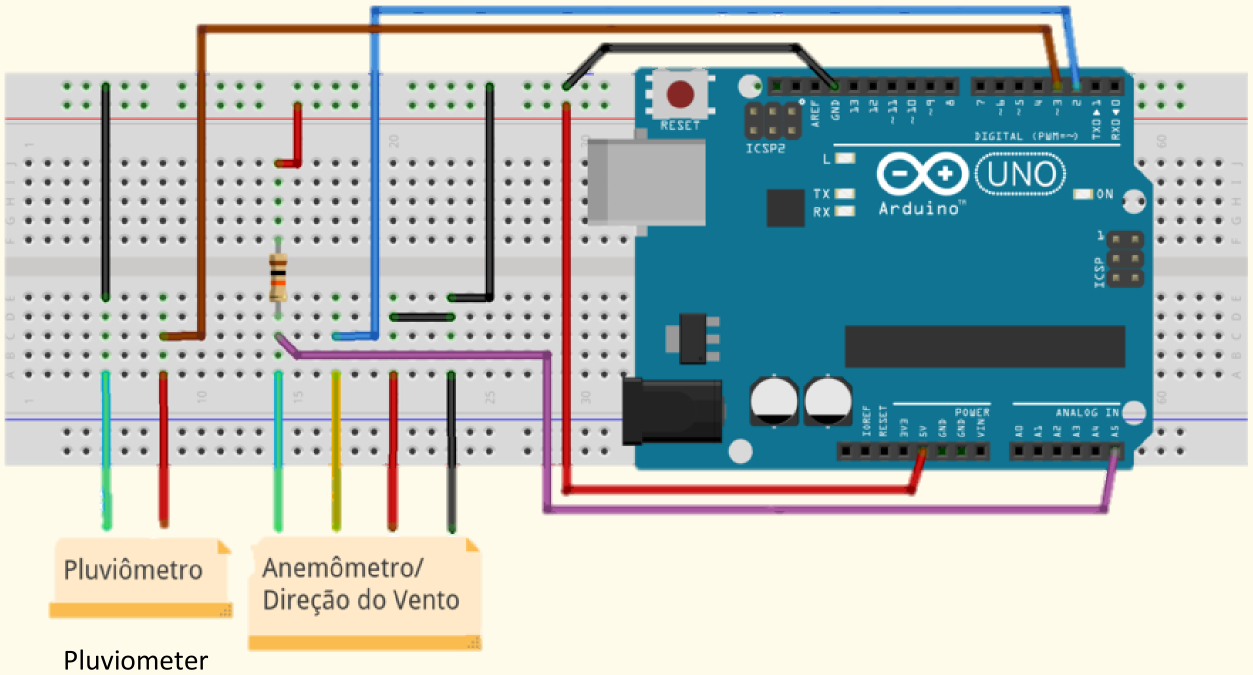
The weather station will also display a rainfall system, thus enabling the determination of the volume of rainfall, which occurs in a certain area and in a given period of time.

This data will be sent to the computer and the user can conduct a real time monitoring.

Description of the project:

The project consists of measuring the wind speed and direction with arduino, through an anemometer and a pluviometer that will measure rainfall intensity, if there is any.

Circuit:



Pluviometer

Anemometer/Wind direction

Programming:

```
// CALIBRATION CONSTANTS
// 1 rev/second = 1.492 mph = 2.40114125 kph
#define CTE_CAL_ANEMOMETER 2.4011
// 1 beat = 0.2794 mm
#define CTE_CAL_PLUVIOMETER 0.2794


// Period between measures in milliseconds
#define PERIOD_ANEMOMETER  5000
#define PERIOD_DIR_WIND   5000
#define PERIODO_PLUVIOMETER  5000

// Connection pins with Arduino
#define ANEMOMETER_PIN   2     // Digital 2
```

```
#define PLUVIOMETER_PIN  3      // Digital 3
#define DIR_WIND_PIN     5      // Analog 5

// Variables for incrementation
volatile int numRevsAnemometer = 0;
volatile int numBeatsBascule = 0;

// Variables to conduct polling
unsigned long nextMeasureAnemometer = 0;
unsigned long nextMeasurePluviometer = 0;
unsigned long nextMeasureDirWind = 0;
unsigned long time = 0;

// Wind direction, reading values to differentiate each direction:
// int adc[8] = {26, 45, 77, 118, 161, 196, 220, 256};
int adc[8] = {104, 180, 308, 472, 644, 784, 880, 1024};
// Relationship between the read analog values and what they represent
// To facilitate, you can use String library
char *directions[8] = {"W","NW","N","SW","NE","S","SE","E"};
int directionInitial = 0;

void setup() {
   Serial.begin(9600);
   pinMode(ANEMOMETER_PIN, INPUT);
   pinMode(PLUVIOMETER_PIN, INPUT);
   digitalWrite(ANEMOMETER_PIN, HIGH);
   digitalWrite(PLUVIOMETER_PIN, HIGH);
   attachInterrupt(0, counterAnemometer, FALLING);
   attachInterrupt(1, counterPluviometer, FALLING);
}

void loop() {
    // Conducting polling
   time = millis();

   if (time >= nextMeasureAnemometer) {
      Serial.print("Wind (km/h): ");Serial.println(calcSpeedWind(), 2);
      nextMeasureAnemometer = time + PERIOD_ANEMOMETER;
   }
   if (time >= nextMeasureDirWind) {
      Serial.print("Direction: ");Serial.println(calcDirectionWind());
      nextMeasureDirWind = time + PERIOD_DIR_WIND;
   }
   if (time >= nextMeasurePluviometer) {
      Serial.print("Rain (mm): ");Serial.println(calcAmountRain(), 3);
      nextMeasurePluviometer = time + PERIOD_PLUVIOMETER;
   }
}

/*
    Interrupt callback functions
*/
void counterAnemometer() {
   numRevsAnemometer++;
}

void counterPluviometer() {
   numBeatsBascule++;
}

double calcSpeedWind(){
   double speedAverage;

   speedAverage = numRevsAnemometer;
   speedAverage *= 1000.0*CTE_CAL_ANEMOMETER;
   speedAverage /= PERIOD_ANEMOMETER;

   // Resetting anemometer pulse counter
   numRevsAnemometer = 0;
   return speedAverage;
}

char* calcDirectionWind() {
   int value, x;
   value = analogRead(DIR_WIND_PIN);

   for (x = 0; x < 8; x++) {
      if (adc[x] >= value)
         break;
   }

   // Ajusting initial direction
   x = (x + directionInitial) % 8;
   return directions[x];
}

double calcAmountRain(){
    double volumeAverage;

    volumeAverage = numBeatsBascule;
    volumeAverage *= 1000.0*CTE_CAL_PLUVIOMETER;
    volumeAverage /= PERIOD_PLUVIOMETER;

    numBeatsBascule = 0;
    return volumeAverage;
}
```

PROJECT: SOIL PH MEASUREMENT

- STEM field: Science, technology and electronics.
- Indicative calendar: Any time of the year.
- Activity duration: 3h.
- Type of activity: Operate a humidity sensor with Arduino software.
- Educational objectives
    - Learn how to operate a humidity sensor;
    - Get familiar with Arduino equipment.
- Learning outcomes and acquired competences
    - Be able to operate a humidity sensor without help;
    - Have a good knowledge of Arduino equipment.
- Required material and resources:
    - 1 Arduino Nano or Uno
    - 1 Soil ph measurement sensor
    - Cables
    - https://wiki.dfrobot.com/Gravity__Analog_Spear_Tip_pH_Sensor___Meter_Kit__For_Soil_And_Food_Applications__SKU__SEN0249#Documents
- Description and/or step-by-step instructions

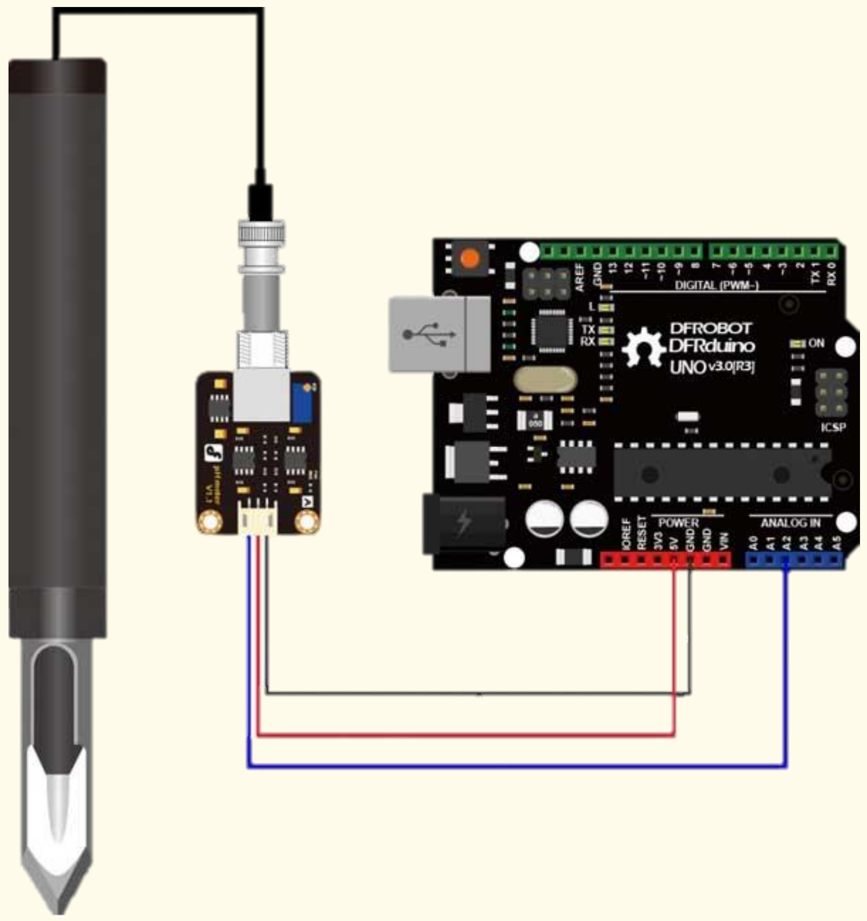
The measurement of pH (the hydrogen ion concentration measure in a certain solution) falls between the values 0 and 14, with 0 representing the most acidic solution and 14 the most alkaline. The pH of the water used for irrigation systems in agriculture plays an important role on crop health and influences the effectiveness of the pesticides and growth regulators. When the pH is too acid, it can turn the leaves yellow and it prevents a natural absorption of Iron and Nitrogen. The alkaline pH makes the micronutrients unavailable to the plant, causing a higher incidence of diseases. Usually, the plants grow better between a slightly acidic and a neutral pH (entre 5,5 e 7).

For this, it will be important for water treatment to take place and that it consists of the acidification or alkalinisation of its components, thus changing the proportion of the hydrogen ions. A system that could be helpful for this analysis is the pH medication factor in the water, so we will use a pH sensor that will send the information to the microcontroller, where it will be converted between the values 0 and 14 and the user will have, in real time, the water pH information. The water pH is a practical way of interfering directly in the soil pH, and in addition to regulating the water pH, it proceeds to fertirrigation, since the regulation of the pH is about fertilization system and organic nutrition in sustainable way.

Description of the project:

The pH probe in this kit is secured with a protective shield similar to a spear made of stainless steel with a sharp end. It can be pierced directly in the soft semi-solid materials in order to measure the pH value, like humid soil or food. But that cannot be done with a common pH probe, or it will be damaged.

Circuit:



Coding/programming:

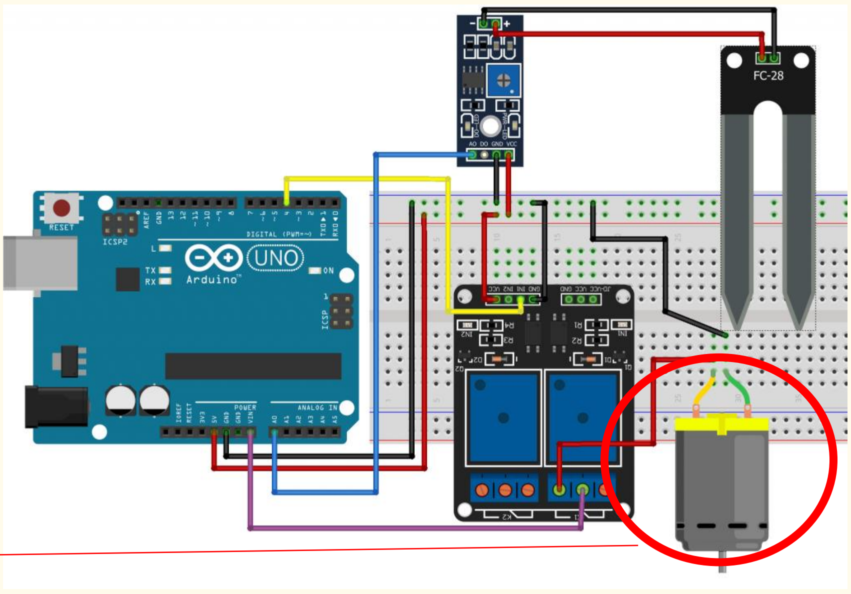
```
#define PHSensorPin  A2    //dissolved oxygen sensor analog output pin to arduino mainboard
    #define VREF 5.0    //for arduino uno, the ADC reference is the AVCC, that is 5.0V(TYP)
    #define OFFSET 0.00  //zero drift compensation

    #define SCOUNT  30           // sum of sample point
    int analogBuffer[SCOUNT];    //store the analog value in the array, readed from ADC
    int analogBufferTemp[SCOUNT];
    int analogBufferIndex = 0,copyIndex = 0;

    float averageVoltage,phValue;

    void setup()
    {
        Serial.begin(115200);
        pinMode(PHSensorPin,INPUT);
    }

    void loop()
    {
       static unsigned long analogSampleTimepoint = millis();
       if(millis()-analogSampleTimepoint > 30U)      //every 30 milliseconds,read the analog value from the
ADC
       {
         analogSampleTimepoint = millis();
         analogBuffer[analogBufferIndex] = analogRead(PHSensorPin);    //read the analog value and store
into the buffer
         analogBufferIndex  ;
         if(analogBufferIndex == SCOUNT)
             analogBufferIndex = 0;
       }
       static unsigned long printTimepoint = millis();
       if(millis()-printTimepoint > 1000U)
       {
          printTimepoint = millis();
          for(copyIndex=0;copyIndex<SCOUNT;copyIndex  )
          {
            analogBufferTemp[copyIndex]= analogBuffer[copyIndex];
          }
          averageVoltage = getMedianNum(analogBufferTemp,SCOUNT) * (float)VREF / 1024.0; // read the value
more stable by the median filtering algorithm
          phValue = 3.5 * averageVoltage+OFFSET;
          Serial.print("Voltage:");
          Serial.print(averageVoltage,2);
          Serial.print("   pH value:");
          Serial.println(phValue,2);
       }
    }

    int getMedianNum(int bArray[], int iFilterLen)
    {
          int bTab[iFilterLen];
          for (byte i = 0; i<iFilterLen; i  )
          {
          bTab[i] = bArray[i];
          }
          int i, j, bTemp;
          for (j = 0; j < iFilterLen - 1; j  )
          {
          for (i = 0; i < iFilterLen - j - 1; i  )
              {
            if (bTab[i] > bTab[i+1])
                {
            bTemp = bTab[i];
                bTab[i] = bTab[i+1];
            bTab[i+1] = bTemp;
             }
          }
          }
          if ((iFilterLen & 1) > 0)
        bTemp = bTab[(iFilterLen - 1) / 2];
          else
        bTemp = (bTab[iFilterLen / 2]+bTab[iFilterLen / 2 - 1]) / 2;
          return bTemp;
    }
```

PROJECT: AUTOMATED IRRIGATION SYSTEM

- STEM field: Science, technology, and electronics
- Indicative calendar: 1 day
- Activity duration: 2h30
- Type of activity: Make an automated irrigation system.
- Educational objectives
    - Describe the usefulness of moisture sensor;
    - Configure the moisture sensor to send the information about soil to the microcontroller;
    - Build the soil system analysis with moisture sensor.
- Learning outcomes and acquired competences:
    - Learn how to use Arduino UNO to make an automated irrigation system that can be used in agriculture;
    - Understand how humidity sensors work;
    - Recognize the hardware needed to build a soil moisture analysis system using a microcontroller;
    - Determine the hardware to be used for reading soil characteristics through the humidity sensor;
    - Program and configure the humidity sensor;
    - Plan and structure tasks;
    - Act with initiative and demonstrate analytical capacity.
- Required material and resources:
    - Arduino UNO R3
    - Small Active Buzzer B10
    - Power Adapter 9 Volt 1 amp
    - Jumper Cable Female-Female (Pack of 10)
    - Double Channel Relay Module
    - Soil Moisture Sensor
    - USB Cable Arduino (A to B)
    - Motor Water Pump Module 12V
    - Jumper Cable Male-Male (Pack of 10)
    - Jumper Cable Male-Female (Pack of 10)
    - Mini Breadboard 170 pins
    - Computer and Arduino IDE
    - Internet connection
    - Cable USB to microUSB
    - BreadBoard

- Description and/or step-by-step instructions
  According to the stored code in Arduino Uno, the water pump will get started automatically to provide water to the plant whenever the soil is dry. As the soil is wet, the soil moisture sensor senses enough moisture level in the ground, and the water pump will automatically stop.
  (Source: 1) https://www.makershala.com/DIY/project/Automatic-Plant-Watering-System-using-Arduino
  2) https://create.arduino.cc/projecthub/MisterBotBreak/how-to-use-a-soil-moisture-sensor-ce769b)

Step 1 Connect/wire all the hardware as it is shown in Figure 1:



The motor shown has the same connection as the water pump.

Figure 1 Connecting Diagram

Step 2 Plugin your Arduino. Connect your Arduino to the USB port of your computer:

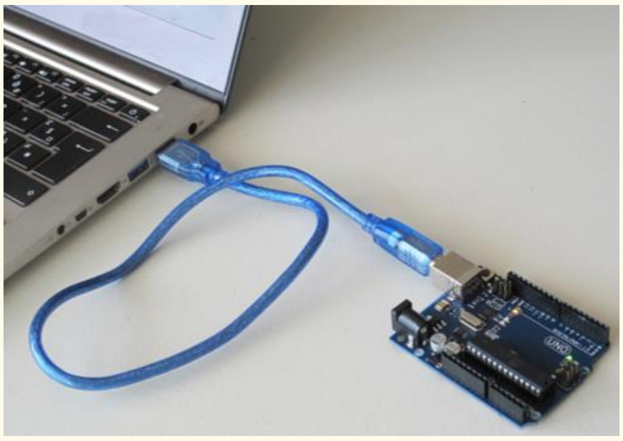
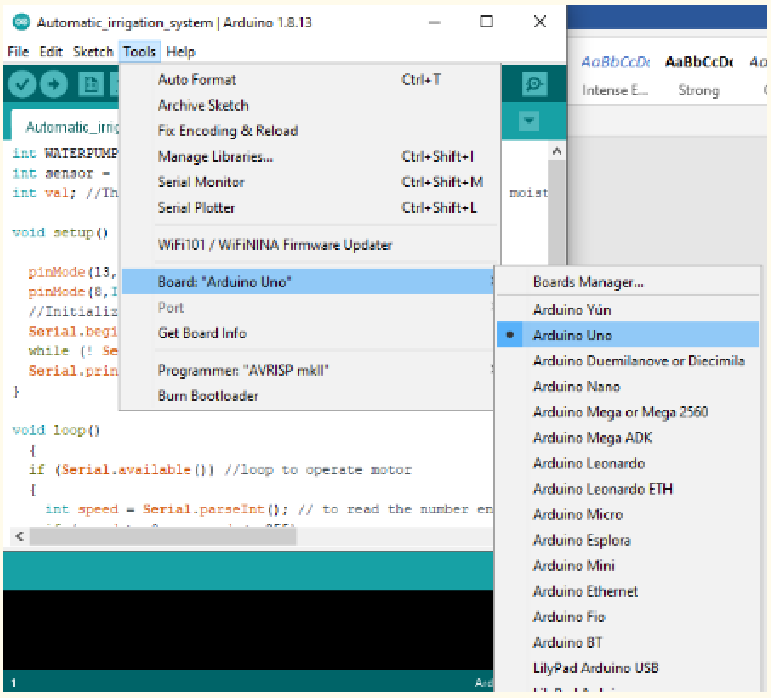


Figure 2 Connect your Arduino to the USB port of your computer. Source: www.getready.io/arduino

Step 3 Open Arduino IDE and select the Arduino Uno board:



```
int WATERPUMP
int sensor =
int val; //Th

void setup()

  pinMode(13,
  pinMode(8,I
  //Initializ
  Serial.begi
  while (! Se
  Serial.prin
}

void loop()
  {
  if (Serial.available()) //loop to operate motor
  {
    int speed = Serial.parseInt(); // to read the number en
```

Figure 3 Choose the right board

Co-funded by the Erasmus+ Programme of the European Union

Step 4 Open a new file:

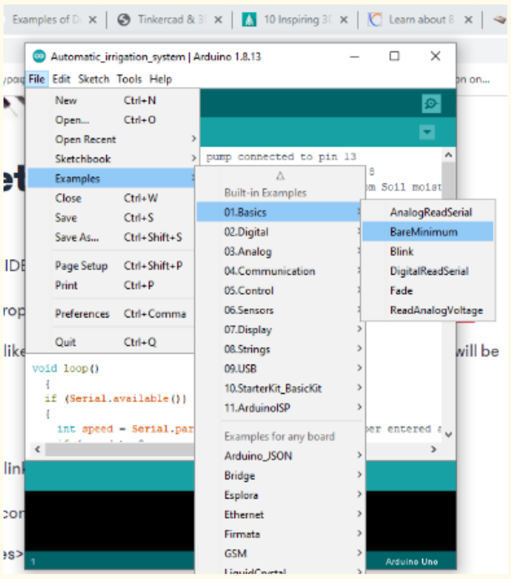
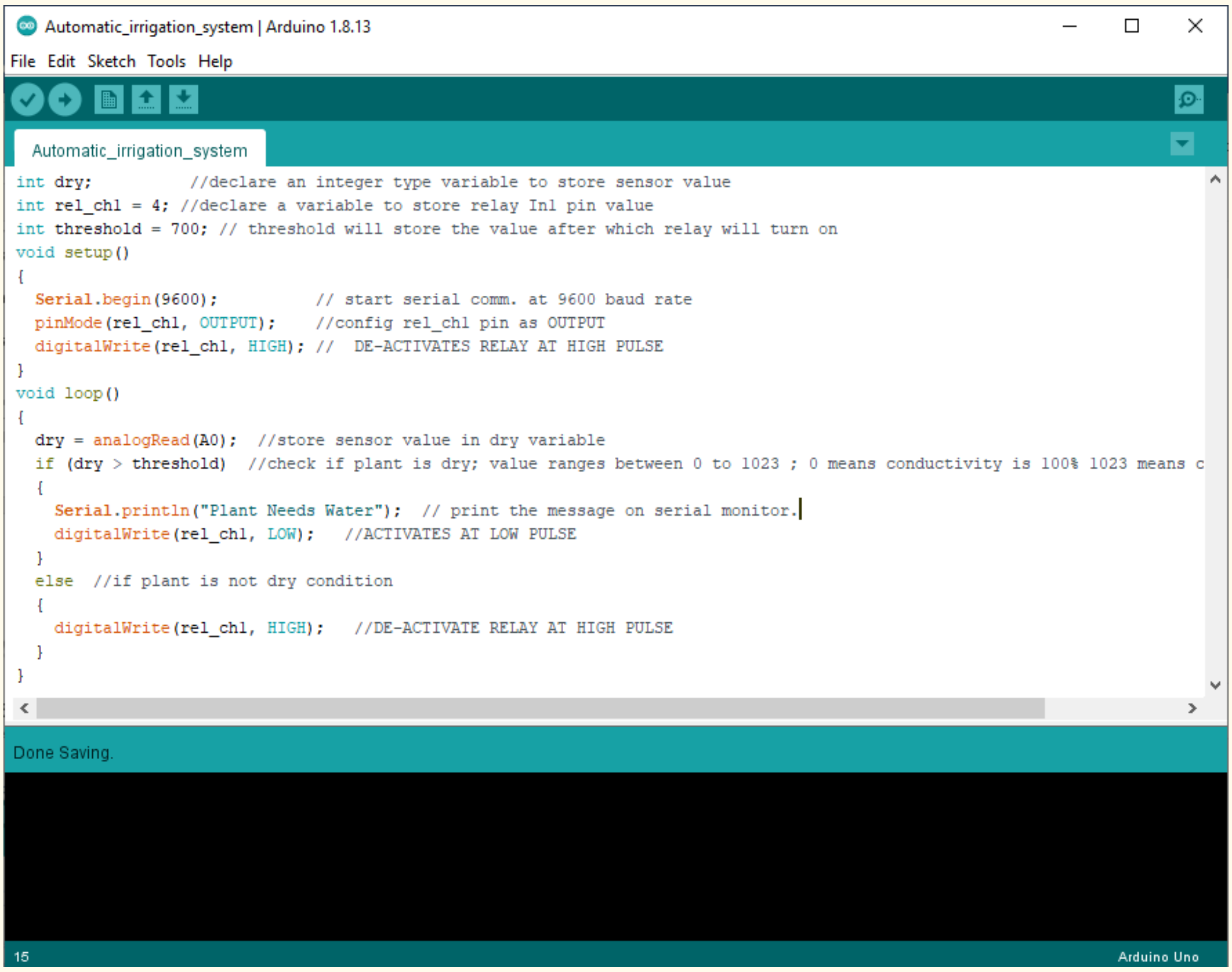


Figure 4 Open a new example file

Step 5 Copy and paste the following code to your project:

```
int dry;          //declare an integer type variable to store sensor
value
int rel_ch1 = 4; //declare a variable to store relay In1 pin value
int threshold = 700; // threshold will store the value after which
relay will turn on
void setup()
{
  Serial.begin(9600);          // start serial comm. at 9600 baud
rate
  pinMode(rel_ch1, OUTPUT);     //config rel_ch1 pin as OUTPUT
  digitalWrite(rel_ch1, HIGH); //  DE-ACTIVATES RELAY AT HIGH PULSE
}
void loop()
{
  dry = analogRead(A0);  //store sensor value in dry variable
  if (dry > threshold)  //check if plant is dry; value ranges
between 0 to 1023 ; 0 means conductivity is 100% 1023 means
conductivity is 0
  {
    Serial.println("Plant Needs Water");  // print the message on
serial monitor.


    digitalWrite(rel_ch1, LOW);    //ACTIVATES AT LOW PULSE
  }
  else  //if plant is not dry condition
  {
    digitalWrite(rel_ch1, HIGH);    //DE-ACTIVATE RELAY AT HIGH PULSE
  }
}
```



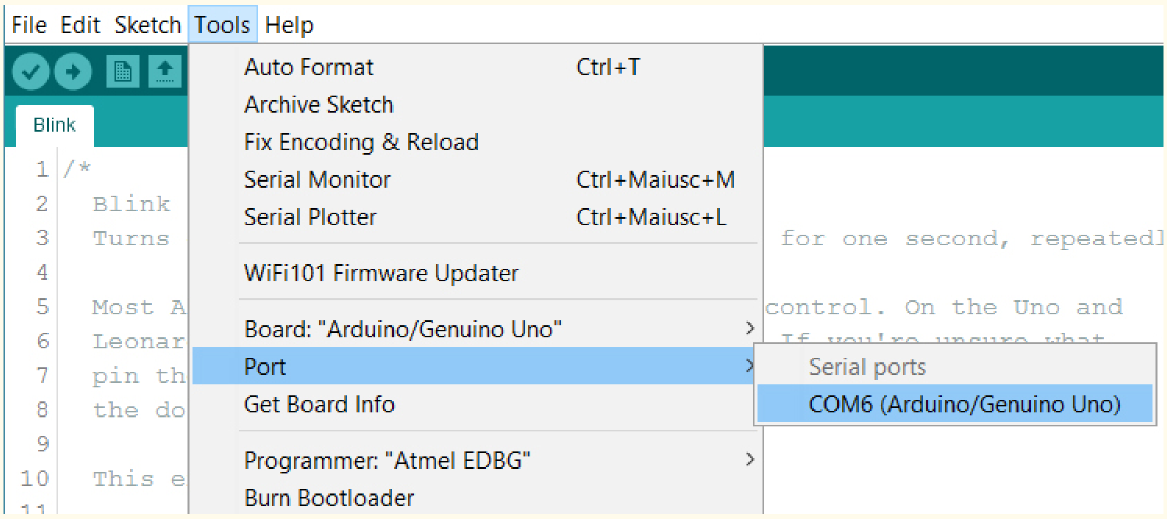
Automatic_irrigation_system | Arduino 1.8.13

```
int dry;          //declare an integer type variable to store sensor value
int rel_chl = 4; //declare a variable to store relay In1 pin value
int threshold = 700; // threshold will store the value after which relay will turn on
void setup()
{
  Serial.begin(9600);          // start serial comm. at 9600 baud rate
  pinMode(rel_chl, OUTPUT);    //config rel_chl pin as OUTPUT
  digitalWrite(rel_chl, HIGH); //  DE-ACTIVATES RELAY AT HIGH PULSE
}
void loop()
{
  dry = analogRead(A0);  //store sensor value in dry variable
  if (dry > threshold)  //check if plant is dry; value ranges between 0 to 1023 ; 0 means conductivity is 100% 1023 means c
  {
    Serial.println("Plant Needs Water");  // print the message on serial monitor.
    digitalWrite(rel_chl, LOW);   //ACTIVATES AT LOW PULSE
  }
  else  //if plant is not dry condition
  {
    digitalWrite(rel_chl, HIGH);   //DE-ACTIVATE RELAY AT HIGH PULSE
  }
}
```

```
Done Saving.
```

Figure 5 Copy and paste the code in your project

Step 6 Select the serial device of the board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports).



```
/*
  Blink
  Turns                for one second, repeatedl

  Most A             control. On the Uno and
  Leonar
  pin th
  the do

  This e
```

Figure 6 Select the right port

Step 7 Upload the program

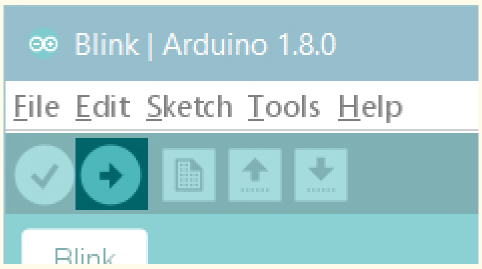


Figure 7 Upload the program

PROJECT: CO2 ANALYZER

- STEM field: Science, technology, and electronics.

- Indicative calendar: Any time of the year.

- Activity duration: 2h30.

- Type of activity: Use Arduino Uno to build a CO2 Analyzer that will be used in agriculture.

- **Educational objectives:**
    - Describe the usefulness of CO2 sensor;
    - Configure the CO2 sensor to send the information for microcontroller;
    - Build the system CO2 Analyzer;

- **Learning outcomes and acquired competences:**
    - Understand how the CO2 sensor works;
    - Recognize the hardware needed to build a CO2 analyzer;
    - Determine the hardware to be used for reading information and characteristics to sensor CO2;
    - Program and configure the CO2 sensor;
    - Plan and structure tasks;
    - Act with initiative and demonstrate analytical capacity;

- Required material and resources:
    - Arduino UNO;
    - USB cable to microUSB;
    - Computer;
    - CO2 Meter K30;
    - Four-wire jumpers;
    - Four 2-pin Vertical PC tail pin headers (Digikey part number 952-2262-ND);

- Description and/or step-by-step instructions

  A CO2 Analyser helps the storage of the crops in agriculture. It provides rapid and precise measurements of CO2 for Quality Assurance staff to increase shelf-life and prevent spoilage by monitoring ripening gases throughout the storage process. The project aims to develop a CO2 device that logs the data from the CO2 sensor onto an SD card for further analysis. This device is composed of four principal elements (Figure 1):

(A) The first component is an Arduino board.

(B) Next is the CO2 sensor for measuring the CO2 concentration in the air.

CO2 Analyser

A CO2 Analyser helps the storage of the crops in agriculture. It provides rapid and precise measurements of CO2 for Quality Assurance staff to increase shelf-life and prevent spoilage by monitoring ripening gases throughout the storage process. The project aims to develop a CO2 device that logs the data from the CO2 sensor onto an SD card for further analysis. This device is composed of four principal elements (Figure 1):

(A) The first component is an Arduino board.

(B) Next is the CO2 sensor for measuring the CO2 concentration in the air.

Step 1 Connect the CO2 Analyser to the Arduino as show in the following picture:

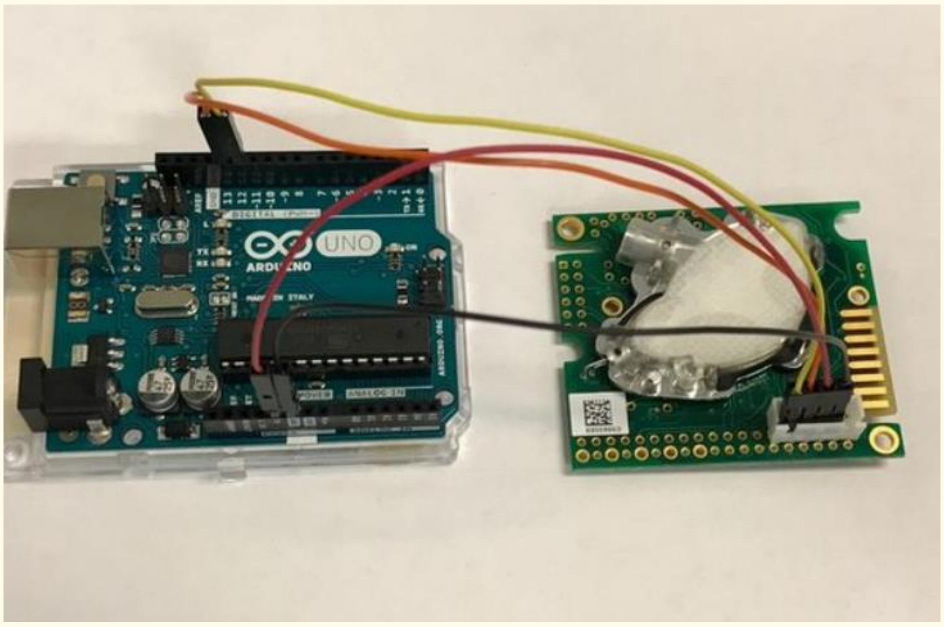


Figure 1 CO2 Analyser connection diagram

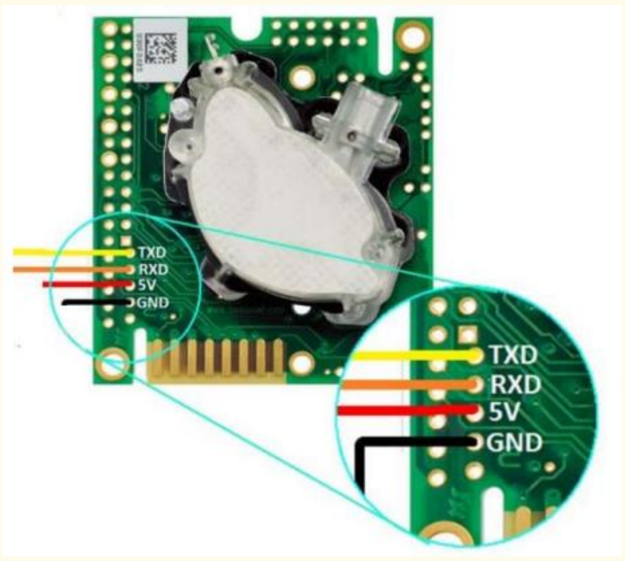
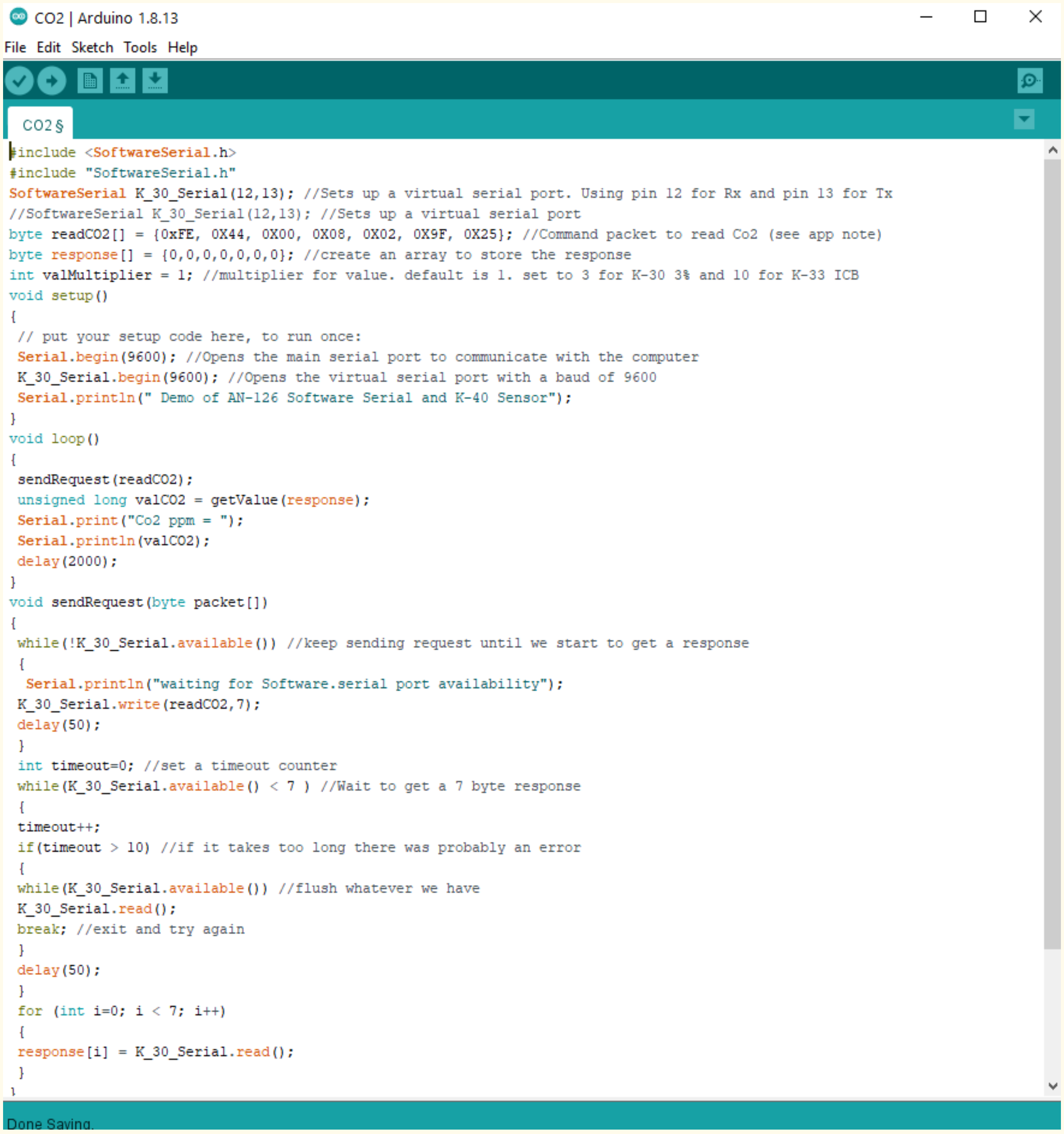


Figure 2 CO2 meter K30

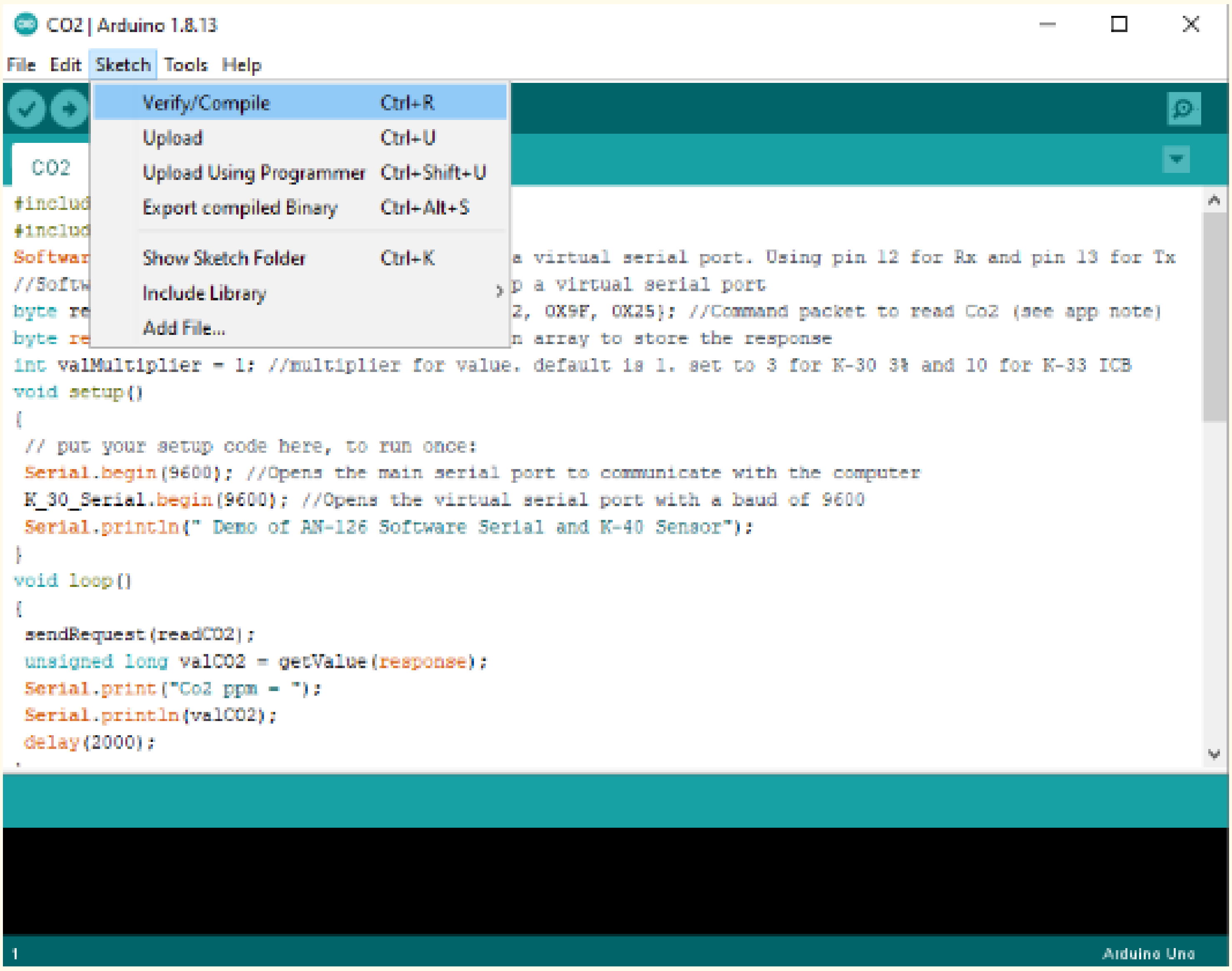
Step 2 Open up a new Android blank sketch and copy the code that you can find in Figure 3



```
#include <SoftwareSerial.h>
#include "SoftwareSerial.h"
SoftwareSerial K_30_Serial(12,13); //Sets up a virtual serial port. Using pin 12 for Rx and pin 13 for Tx
//SoftwareSerial K_30_Serial(12,13); //Sets up a virtual serial port
byte readCO2[] = {0xFE, 0X44, 0X00, 0X08, 0X02, 0X9F, 0X25}; //Command packet to read Co2 (see app note)
byte response[] = {0,0,0,0,0,0,0}; //create an array to store the response
int valMultiplier = 1; //multiplier for value. default is 1. set to 3 for K-30 3% and 10 for K-33 ICB
void setup()
{
 // put your setup code here, to run once:
 Serial.begin(9600); //Opens the main serial port to communicate with the computer
 K_30_Serial.begin(9600); //Opens the virtual serial port with a baud of 9600
 Serial.println(" Demo of AN-126 Software Serial and K-40 Sensor");
}
void loop()
{
 sendRequest(readCO2);
 unsigned long valCO2 = getValue(response);
 Serial.print("Co2 ppm = ");
 Serial.println(valCO2);
 delay(2000);
}
void sendRequest(byte packet[])
{
 while(!K_30_Serial.available()) //keep sending request until we start to get a response
 {
  Serial.println("waiting for Software.serial port availability");
 K_30_Serial.write(readCO2,7);
 delay(50);
 }
 int timeout=0; //set a timeout counter
 while(K_30_Serial.available() < 7 ) //Wait to get a 7 byte response
 {
 timeout++;
 if(timeout > 10) //if it takes too long there was probably an error
 {
 while(K_30_Serial.available()) //flush whatever we have
 K_30_Serial.read();
 break; //exit and try again
 }
 delay(50);
 }
 for (int i=0; i < 7; i++)
 {
 response[i] = K_30_Serial.read();
 }
}
```

Figure 3 CO2 meter code

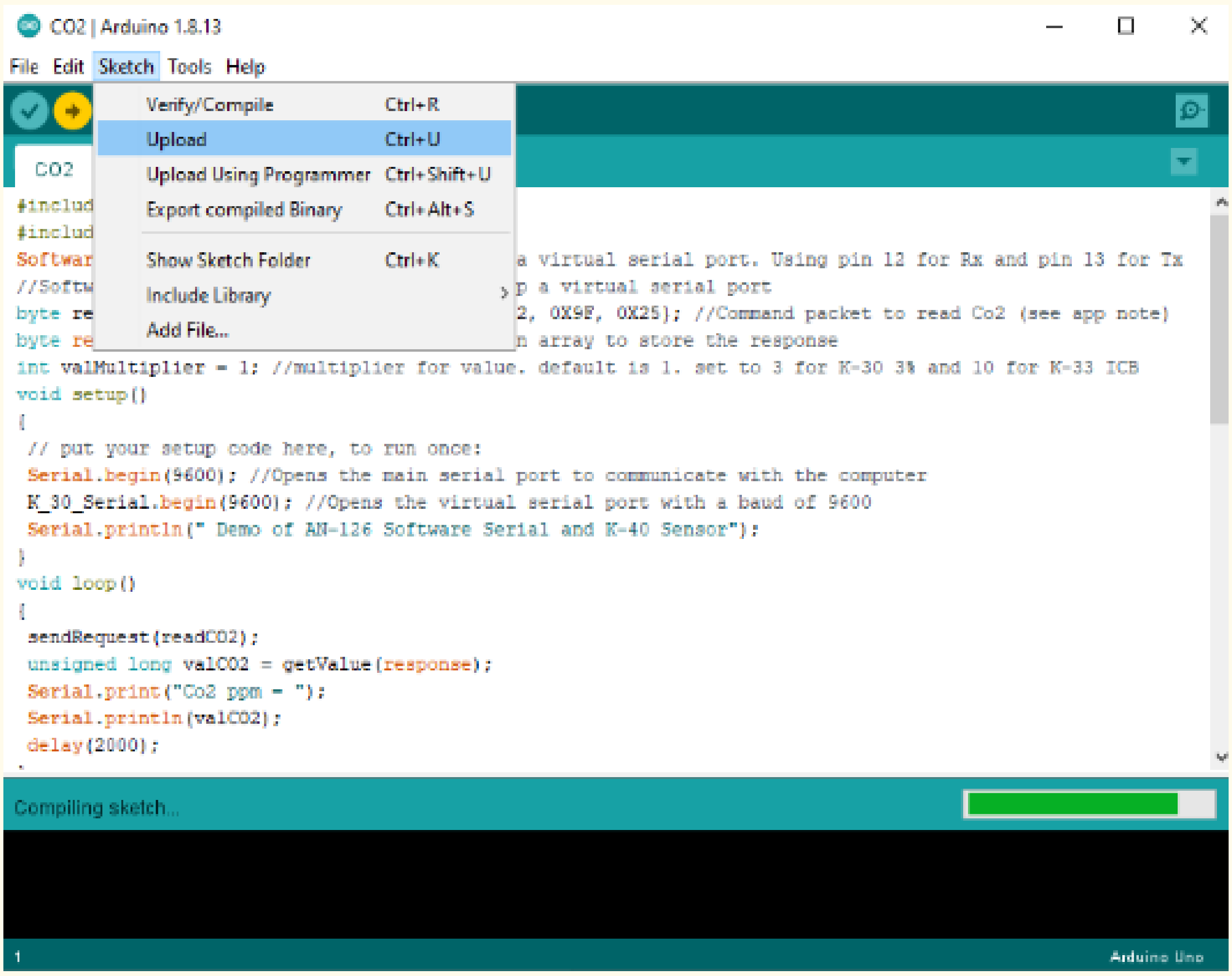
Step 3 In Arduino 1.8.13, click Verify/Compile 7


CO2 | Arduino 1.8.13
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
CO2
1
Arduino Uno

```
#includ
#includ
Softwar                                       a virtual serial port. Using pin 12 for Rx and pin 13 for Tx
//Softw                                       p a virtual serial port
byte re                                       2, 0X9F, 0X25}; //Command packet to read Co2 (see app note)
byte re                                       n array to store the response
int valMultiplier = 1; //multiplier for value. default is 1. set to 3 for K-30 3% and 10 for K-33 ICB
void setup()
{
 // put your setup code here, to run once:
 Serial.begin(9600); //Opens the main serial port to communicate with the computer
 K_30_Serial.begin(9600); //Opens the virtual serial port with a baud of 9600
 Serial.println(" Demo of AN-126 Software Serial and K-40 Sensor");
}
void loop()
{
 sendRequest(readCO2);
 unsigned long valCO2 = getValue(response);
 Serial.print("Co2 ppm = ");
 Serial.println(valCO2);
 delay(2000);
```

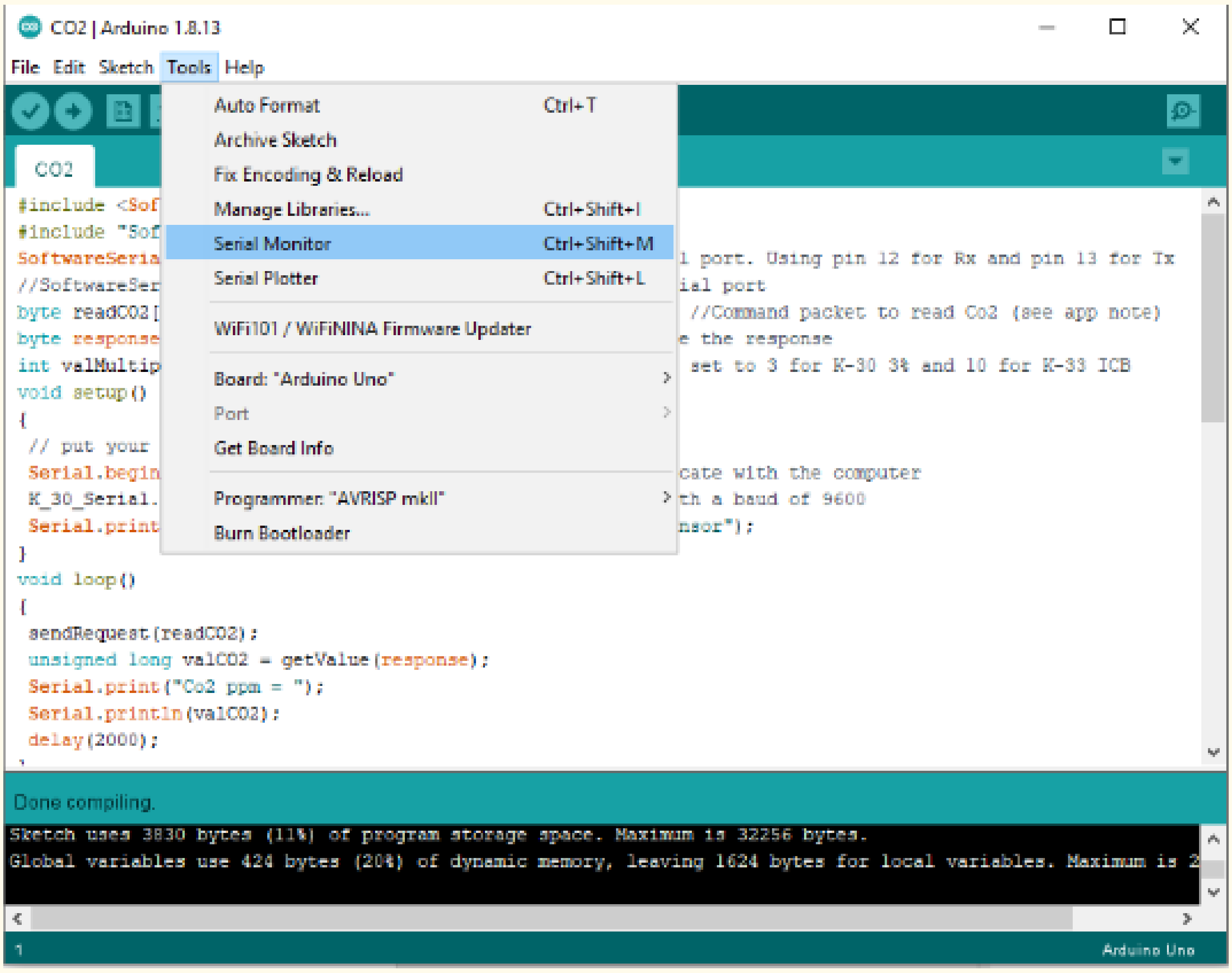
Step 4

To run, click on Sketch > Upload


CO2 | Arduino 1.8.13
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
CO2
Compiling sketch...
Arduino Uno

```
#includ
#includ
Softwar                a virtual serial port. Using pin 12 for Rx and pin 13 for Tx
//Softw                p a virtual serial port
byte re                2, 0X9F, 0X25}; //Command packet to read Co2 (see app note)
byte re                n array to store the response
int valMultiplier = 1; //multiplier for value. default is 1. set to 3 for K-30 3% and 10 for K-33 ICB
void setup()
{
 // put your setup code here, to run once:
 Serial.begin(9600); //Opens the main serial port to communicate with the computer
 K_30_Serial.begin(9600); //Opens the virtual serial port with a baud of 9600
 Serial.println(" Demo of AN-126 Software Serial and K-40 Sensor");
}
void loop()
{
 sendRequest(readCO2);
 unsigned long valCO2 = getValue(response);
 Serial.print("Co2 ppm = ");
 Serial.println(valCO2);
 delay(2000);
```

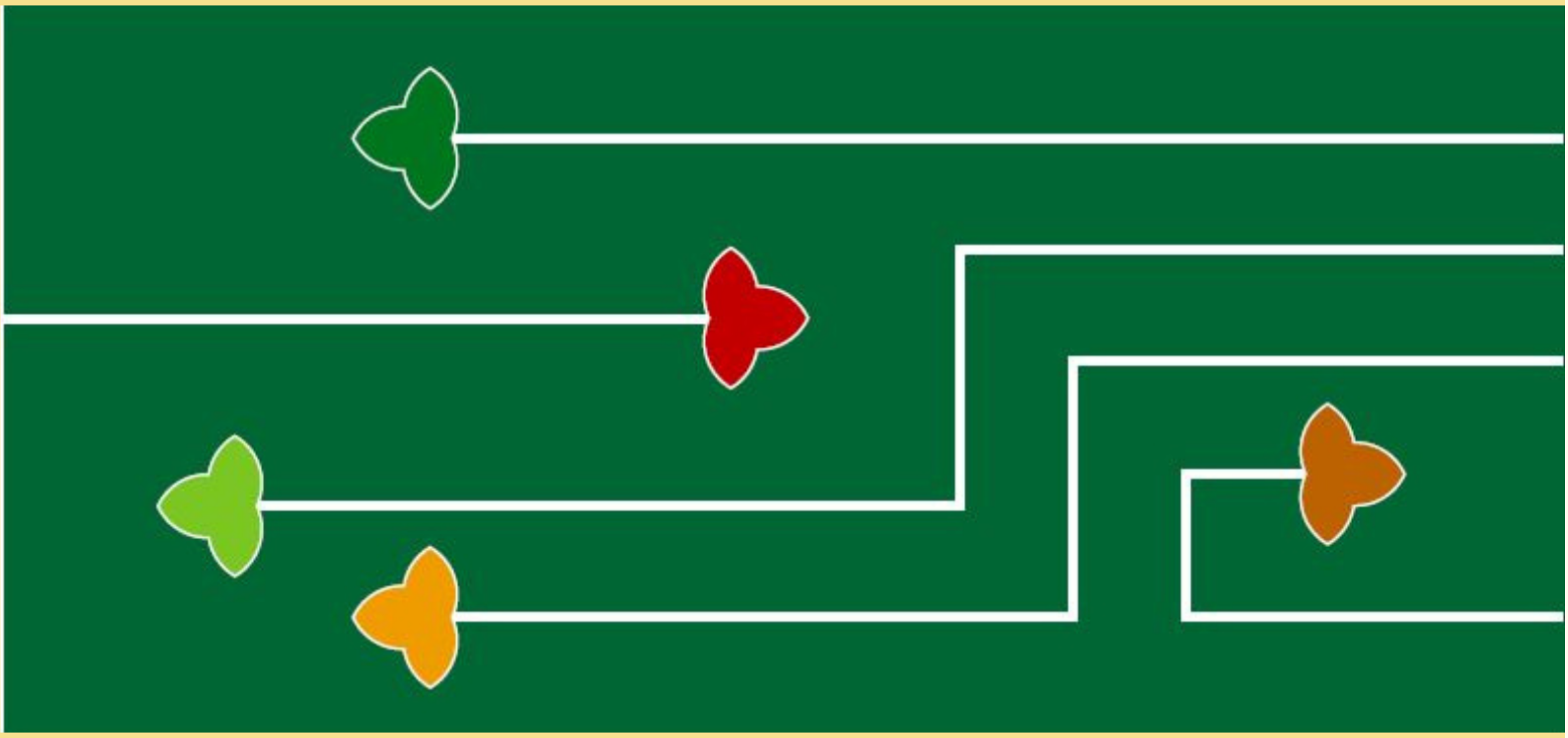
Step 5 To view the program operating, click on Tools > serial Monitor


CO2 | Arduino 1.8.13
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
CO2
Arduino Uno

```
#include <Sof
#include "Sof
SoftwareSeria                                  l port. Using pin 12 for Rx and pin 13 for Tx
//SoftwareSer                                  ial port
byte readCO2[                                   //Command packet to read Co2 (see app note)
byte response                                  e the response
int valMultip                                   set to 3 for K-30 3% and 10 for K-33 ICB
void setup()
{
 // put your
 Serial.begin                                  cate with the computer
 K_30_Serial.                                  th a baud of 9600
 Serial.print                                  nsor");
}
void loop()
{
 sendRequest(readCO2);
 unsigned long valCO2 = getValue(response);
 Serial.print("Co2 ppm = ");
 Serial.println(valCO2);
 delay(2000);
```

Done compiling.

```
Sketch uses 3830 bytes (11%) of program storage space. Maximum is 32256 bytes.
Global variables use 424 bytes (20%) of dynamic memory, leaving 1624 bytes for local variables. Maximum is 2
```



Erasmus+

The Green Steam Incubator project has been funded with support from the European Commission. The publication reflect the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Project number: 2019-3-CY02-KA205-001692