Table of Contents

senseBox:edu

The senseBox:edu is a toolbox which helps teaching programming to students and junior-researchers in a playful manner.

The Arduino-based set of electronic components is used to build increasingly complex circuits, which are controlled via a microcontroller.

With the learning-resources on these pages, the young students can gather experience in (Arduino-) programming and electronic circuits with hands-on training.

About this book

On these pages all resources regarding the senseBox:edu may be found. This includes tutorials, exercises, example projects, as well as download links.

Contributions (improvements to existing content, or entirely new content!) are appreciated! We're happy to include documentation you made for projects with the senseBox in this book. To do so, have a look on our contribution guide.

The source code of this book is available on GitHub, where you also may leave feedback about this book.

You may download these pages as PDF document for printing, have a look in the download section.

Contributing Projects

If you developed a senseBox project on your own and did document it, you may add it to this book!

Such a documentation should provide an overview over the project as well as a step-by-step tutorial on how to build / reproduce the project. Photos and code may be included, too!

For this book's management we use github.com and the tool GitBook, all content is written in markdown.

Besides new project tutorials we also appreciate any improvements or corrections to the existing content!

Writing the documentation

The documentation should be written in markdown, so the content can be directly included in the book. If you are not familiar with markdown, have a look at an explanation and syntax-explanation here.

To streamline the writing of markdown we recommend dedicated editors, such as the web-editor stackedit.

Content

Should it be applicable to your project, you may use our **project template** as a starting point.

Make shure you provide all information required to reproduce your project with a senseBox:edu kit!

File structure

Place your documentation file in the directory `edu/en/community_projects/` .

If you want to include additional resources, place them in a subfolder with the same name:

```
mobile-weatherstation.md
mobile-weatherstation/overview.jpg
mobile-weatherstation/mobile-weatherstation.ino
```

Filenames may not include any spaces!

Content license

Your contribution will be added under the same CC BY-SA 4.0 license as our content. This means, that the content may be freely adopted by others, as long as the authors name is provided with the content.

Uploading the documentation

To provide us your documentation, you can insert it in the book yourself by submitting a pull request on GitHub. The source code of this book is hosted on GitHub. **Fork this repository, add your content there, and create a new pull request.**

In case you are unfamiliar with the GitHub process, have a look at a GitHub contribution guide.

Alternatively to working directly on GitHub, there is also a Gitbook.com editor, wich might simplify the process. However we didn't try that one.

If none of that works, just send us your contribution written in markdown via mail.

> **Having issues? Mail our support!**

Thank you for your contribution!

Community Projects

In this section community-contributed projects are documented.

You may also have a look on german project tutorials in the german section of this book.

If you want to contribute a project documentation yourself, have a look on our contribution guide.

Project Title	Author	Date	additional material
ArduinoWeatherStation	Michelle Gybel & Johannes Schöning (Hasselt University)	07.07.2016	no

DIGITAL CITIZENS, CONNECTED COMMUNITIES:

THE ROLE OF SPATIAL COMPUTING AND VOLUNTEERED GEOGRAPHIC INFORMATION

TUTORIAL: "MOBILE SENSOR LOGGER"

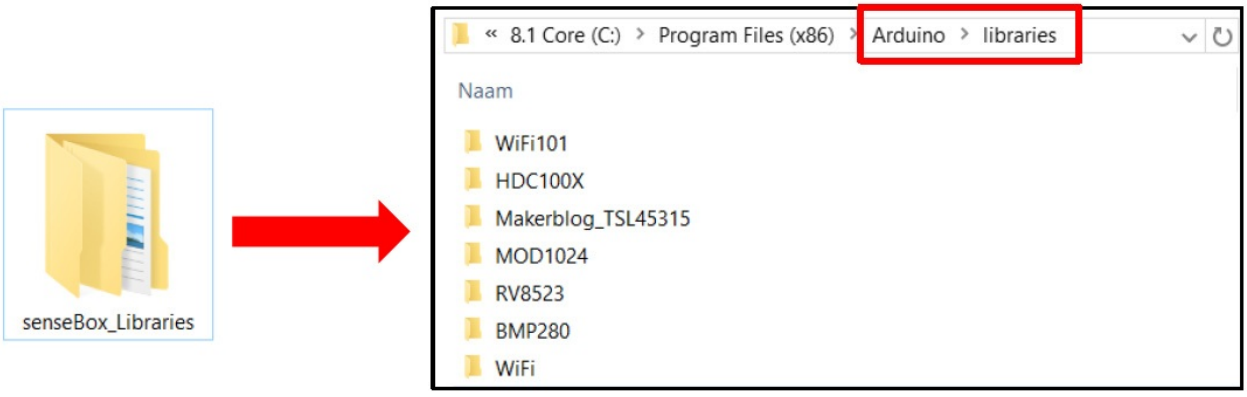
Authors: Michelle Gybel and Johannes Schöning of Hasselt University

Edited by Felix Erdmann using *pandoc*

This tutorial is a step-by-step guide which will teach you how to build mobile sensor logging device using the Sensebox Edu kit.

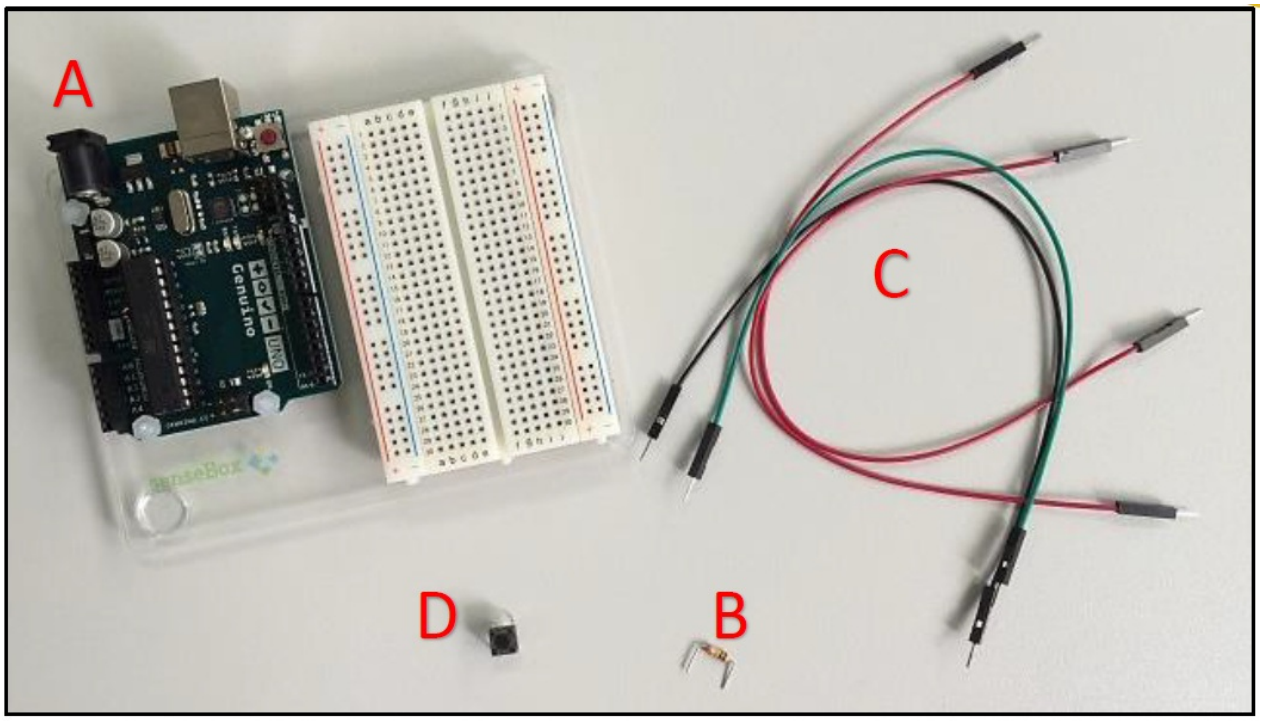
Preparation

1. Install Arduino
    - https://www.arduino.cc/en/Main/Software
    - (Extra:) **Tutorials** and **Reference Guides** can be found here: https://www.arduino.cc/en/Guide/HomePage
1. Install SenseBox Plugins
    - Download from link:
    - https://github.com/senseBox/OER/blob/master/Libraries/senseBox_Libraries.zip?raw=true
    - Extract the folder and copy the content to the **libraries** folder in Arduino's installation files.

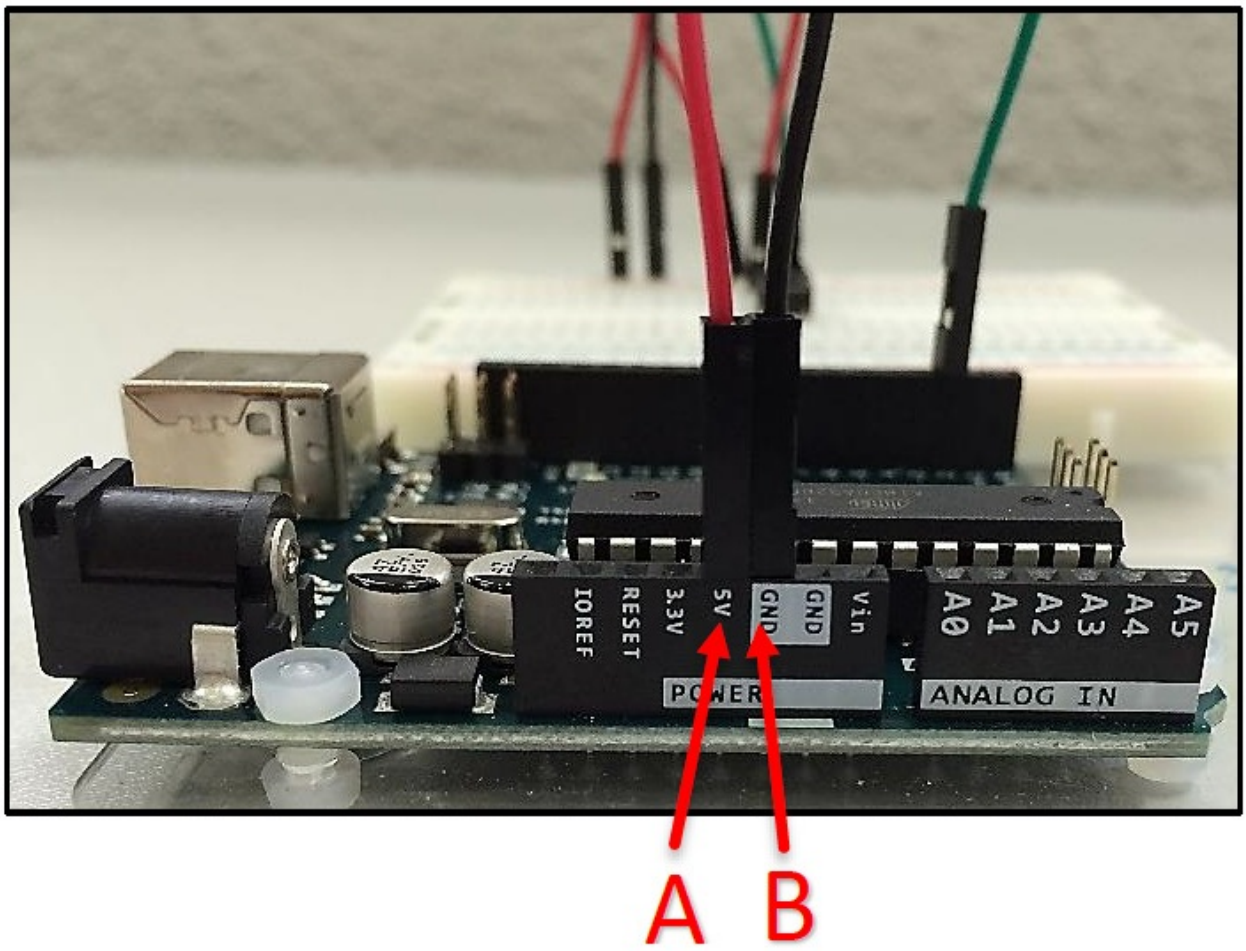


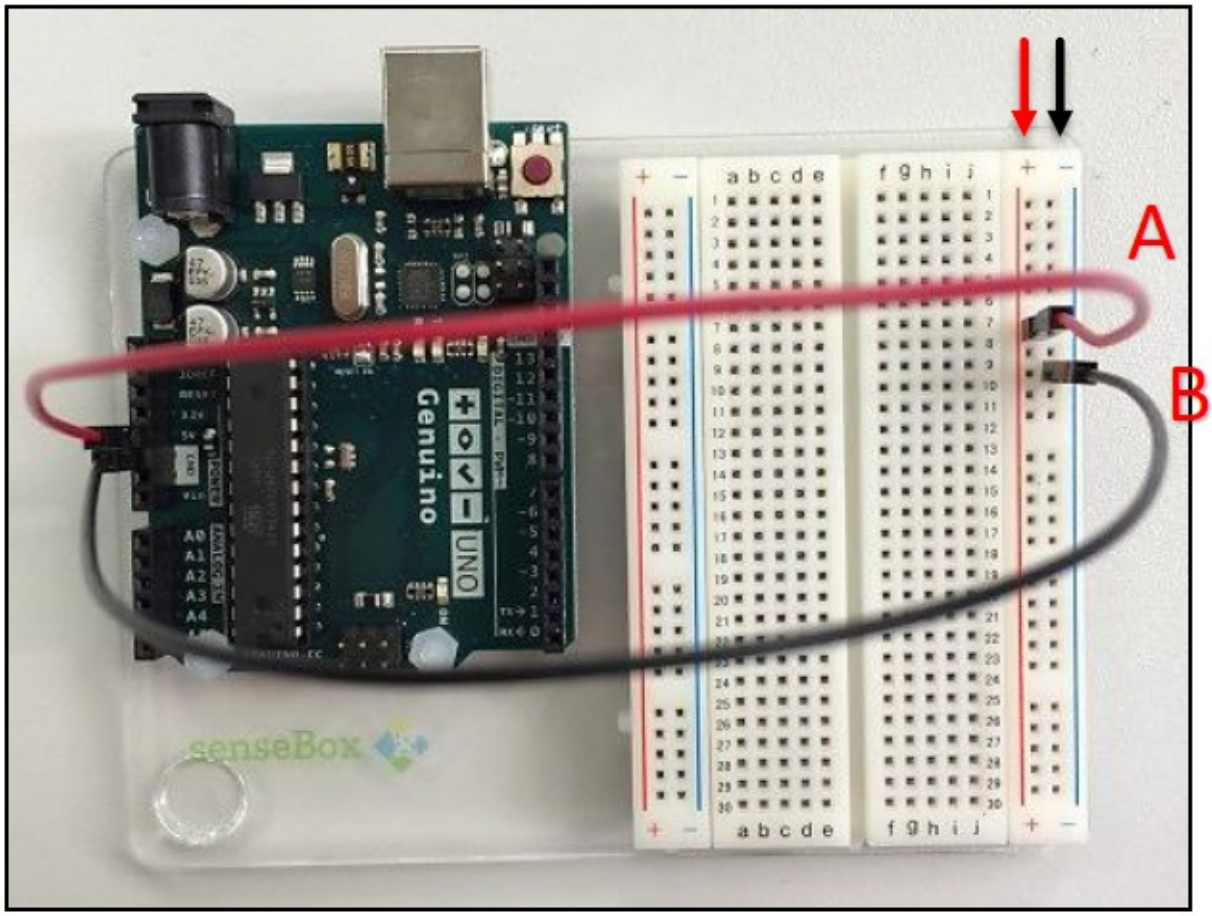
Start with the basics… How to Push a button!

1. What do you need?
    - Genuino Board with breadboard **(A)**
    - 10K ohm resistor **(B)**
    - Wires **(C)**
    - Pushbutton (switch) **(D)**

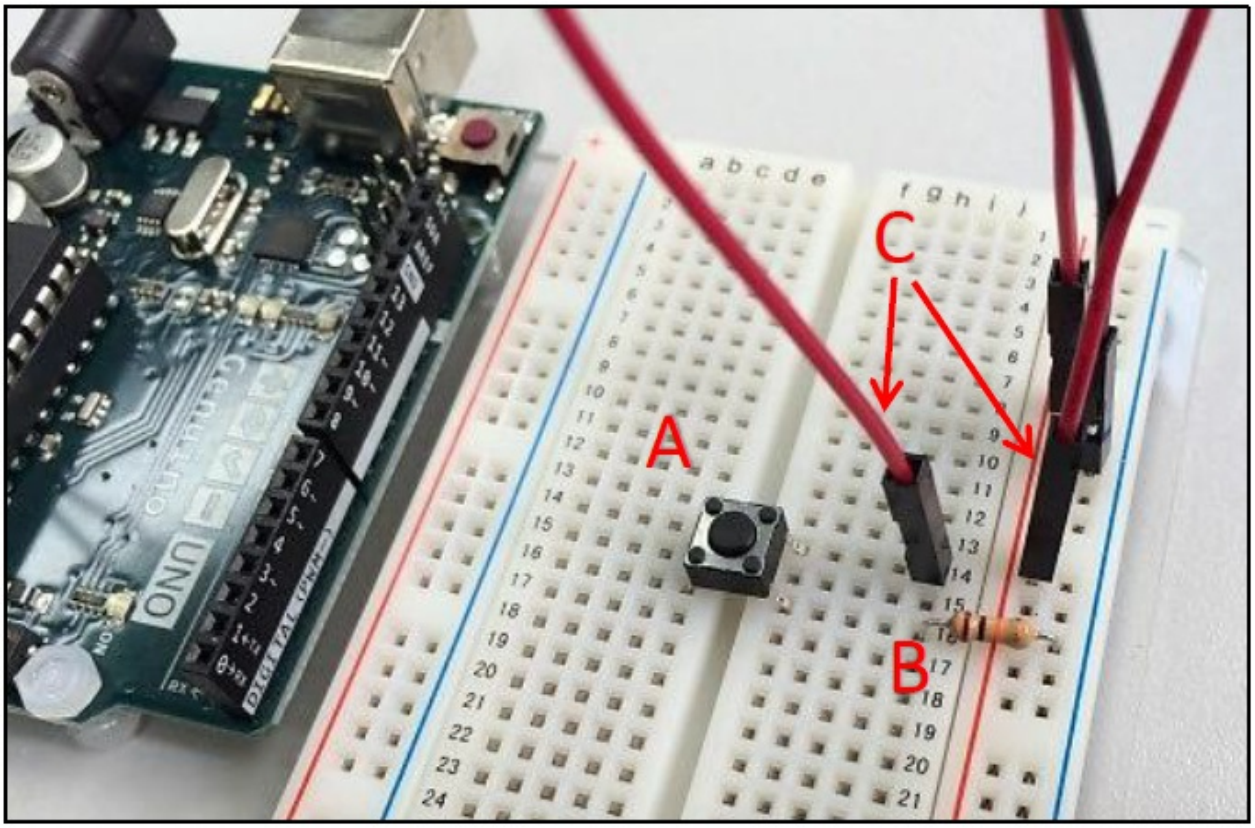


1. Provide voltage and grounding to the breadboard
    - Connect a red wire to the positive vertical row of the breadboard and to the 5 volt supply on the Genuino Board **(A)**.
    - Connect a black wire to the negative vertical row of the breadboard and to the ground (GND) on the Genuino Board **(B)**.

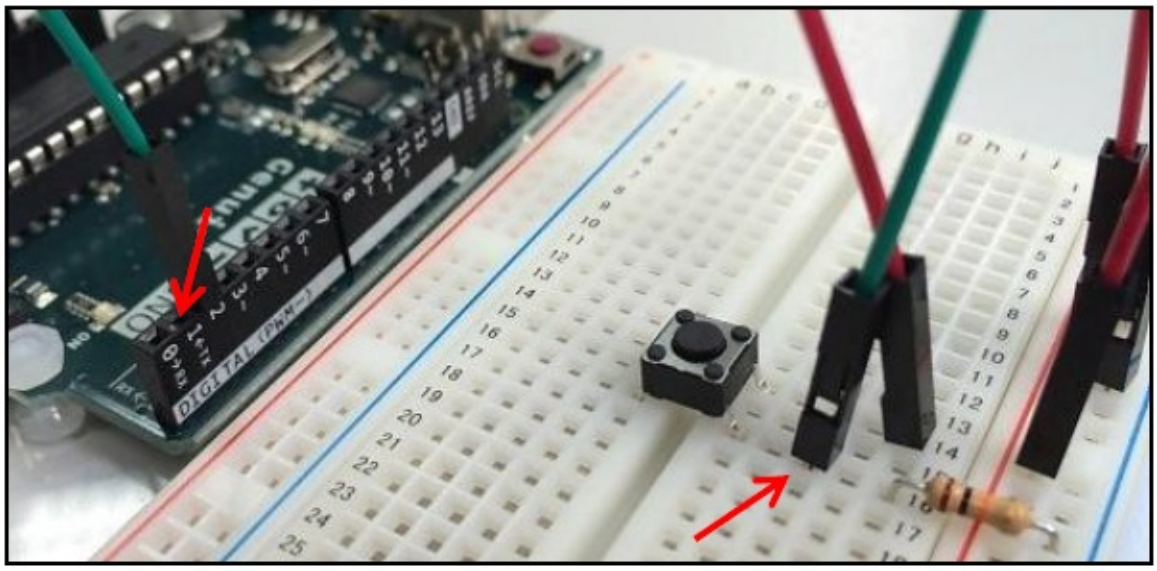

IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
POWER
ANALOG IN
A
B


Genuino
UNO
senseBox
A
B

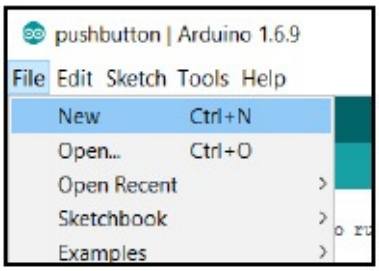
1. Add the pushbutton and provide it with power and grounding
    - Attach the **pushbutton** to the *breadboard* as presented below **(A)**.
    - Create a connection between one of the legs of the pushbutton and the ground with the **10K ohm resistor (B)**.
    - Connect a red wire to the **positive** vertical row of the *breadboard* **(C)**.



- Provide with a green wire a connection between digital pin 2 on the Genuino Board and the leg of pushbutton connected to the ground.

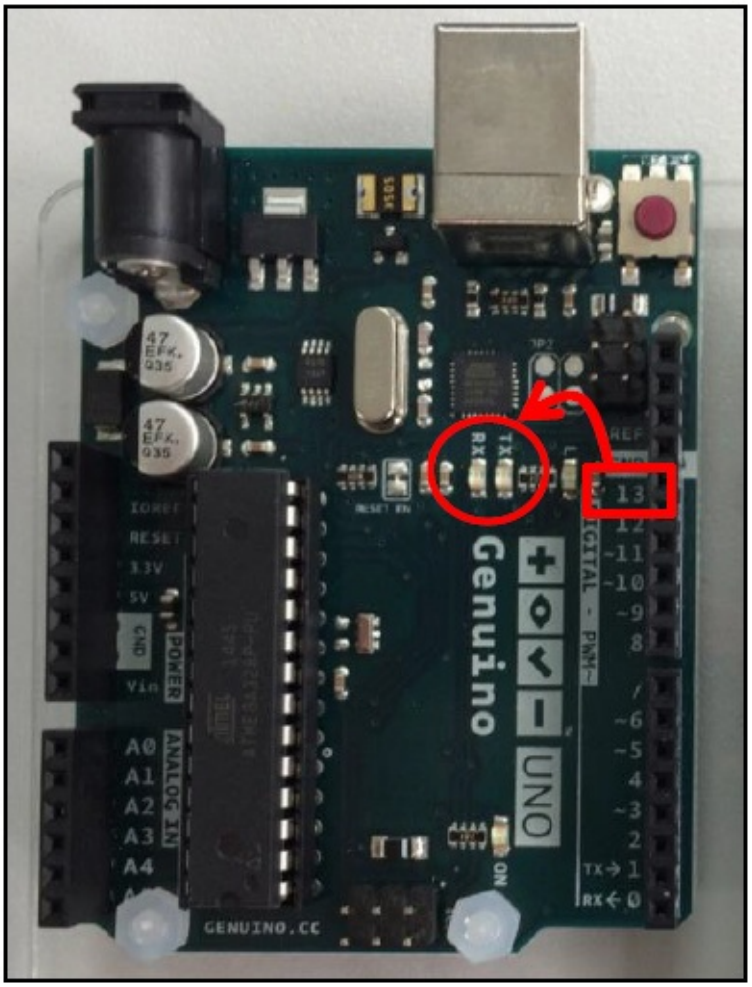


1. Turn on the light, ähm button… by writing some code
    - Start the Arduino IDE.
    - Programs written in the IDE are called sketches. Go to File New and name the file Pushbutton.ino or similar to create your first sketch.



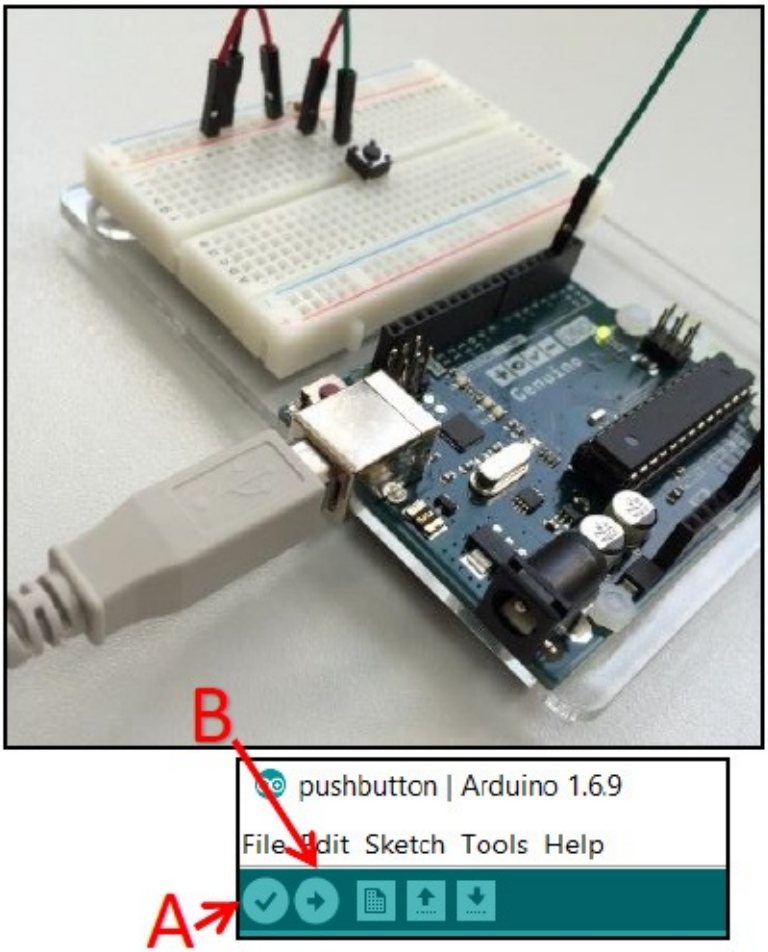
- Each sketch must contain a setup() and a loop() function:

- As a start you can may place the following code in your sketch. It initializes the which pins are connected to the LED light on the Genuino Board and the push button. Next, it provides the functionality to turn the LED light on when the button is pressed in. This LED is built-in in the Genuino board and driven by pin 13.




```
const int buttonPin = 2; // the number of the pin to which the pushbutton is connected
const int ledPin = 13; // the number of the pin connected to the LED
int buttonState = 0; // variable for reading the status of the pushbutton
void setup() {
  pinMode(buttonPin, INPUT); // initialize the pushbutton pin as an input
  pinMode(ledPin, OUTPUT); // initialize the LED pin as an output
}
void loop() {
  buttonState = digitalRead(buttonPin); // get the value of the pushbutton
  //check if the pushbutton is pressed in
  if (buttonState == HIGH){
  digitalWrite(ledPin, HIGH); // if the button is pushed, turn LED on
  } else {
  digitalWrite(ledPin, LOW); // if the button is released, turn LED off
  }
}
```

1. Compile and upload the sketch to the board
    - Connect the board to your computer with a USB cable.



    - Compile the code by clicking on the Verify button **(A)**.
    - Click on the Upload button **(B)**.

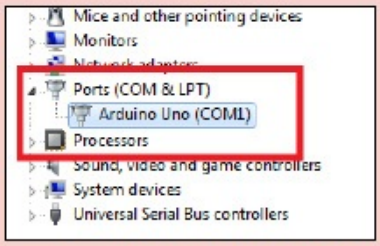
Help! I'm on Windows and I got an error!

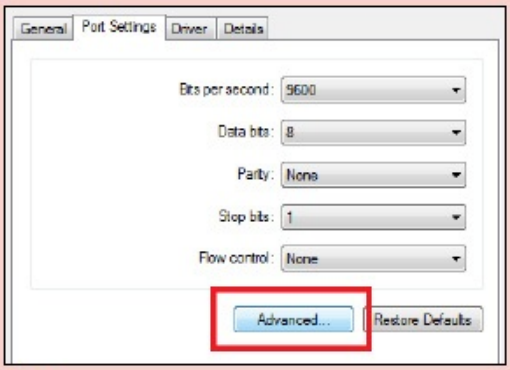
Windows users might get the following error while trying to upload their code:

AVRDUDE: SER_OPEN():SYSTEM CAN'T OPEN DEVICE "\.\COM1"...

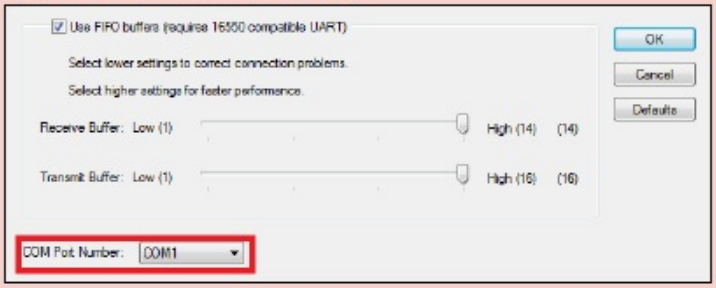
This can be solved by setting the Genuino Uno port to COM1:

- Open the **Device Manager**.
- Go to **Ports → Arduino Uno / Genuino Uno .**
- Open the **Advanced properties.**

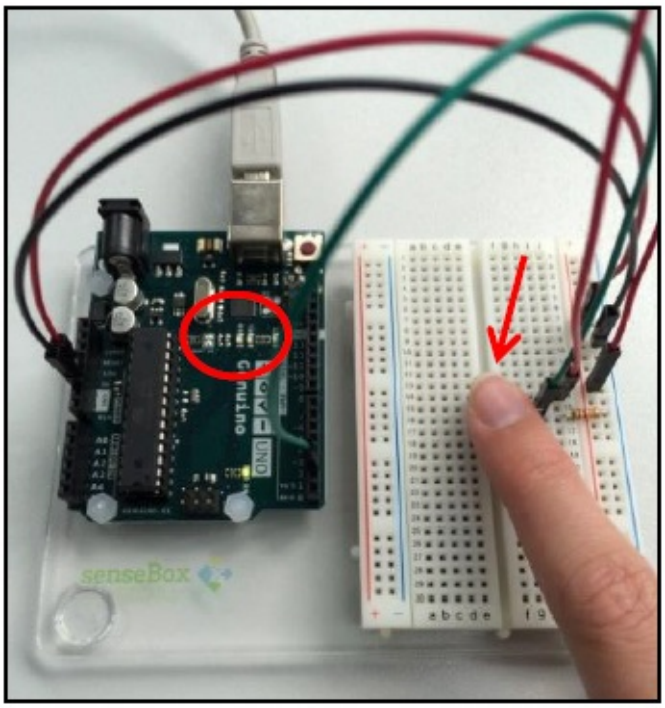




- Set the **COM Port Number** to **COM1.**

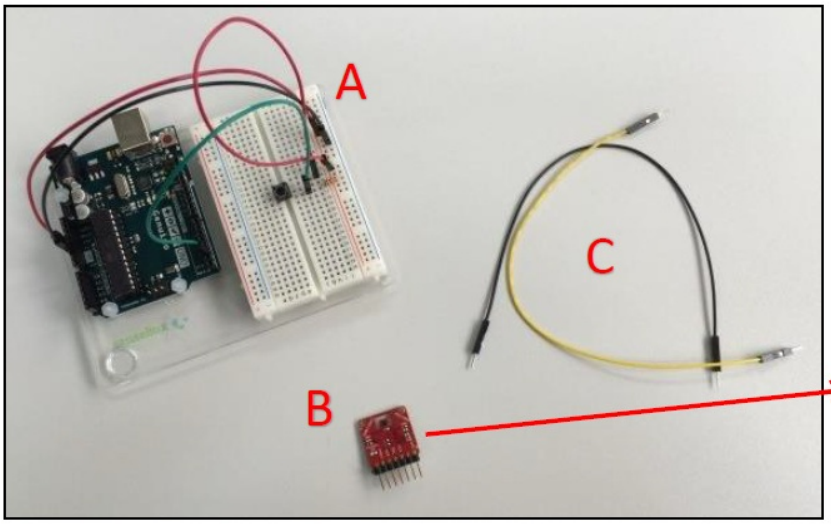


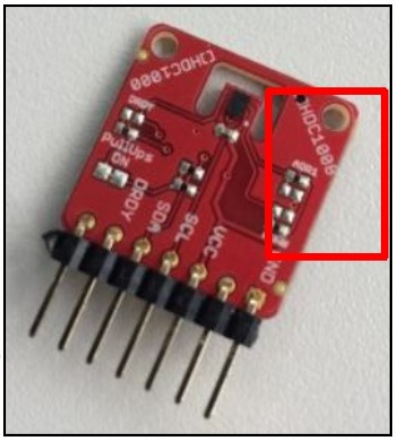
1. Test your program
    - Push the button on the board.
    - A LED light on the Genuino Board should light up.



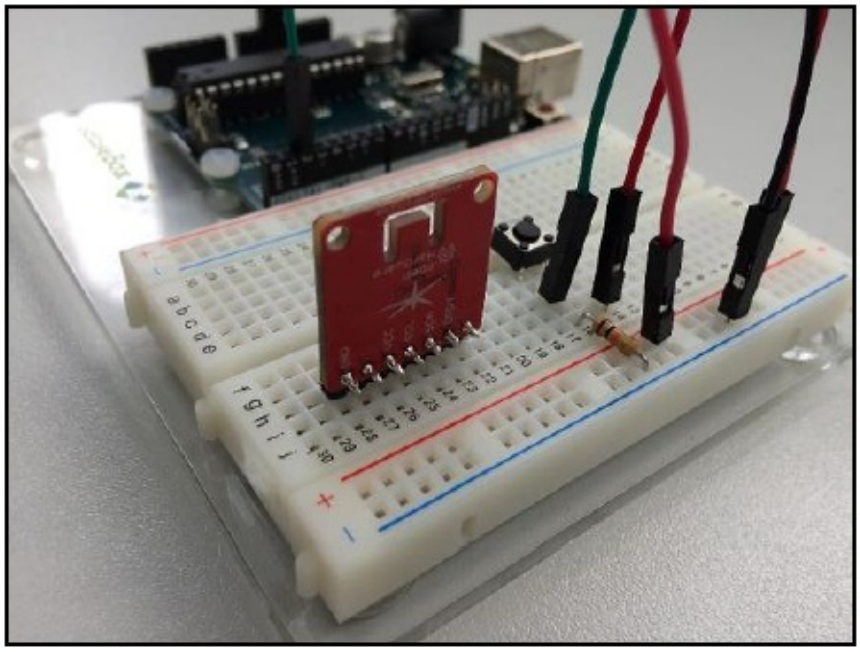
Let’s build the Sensor Data logger

1. What do you need?
    - Your “Push-a-button” construction (A)
    - HDC1008 Temperature and Humidity Sensor (B)
    - Extra wires (C)

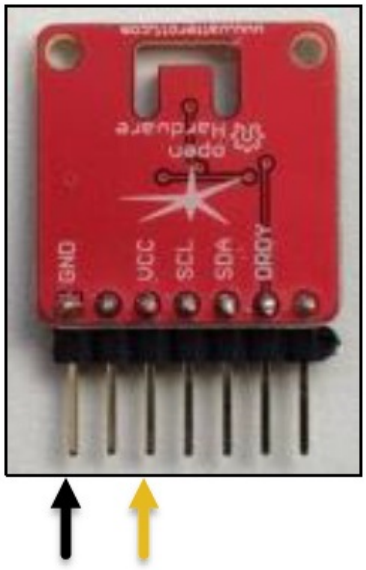


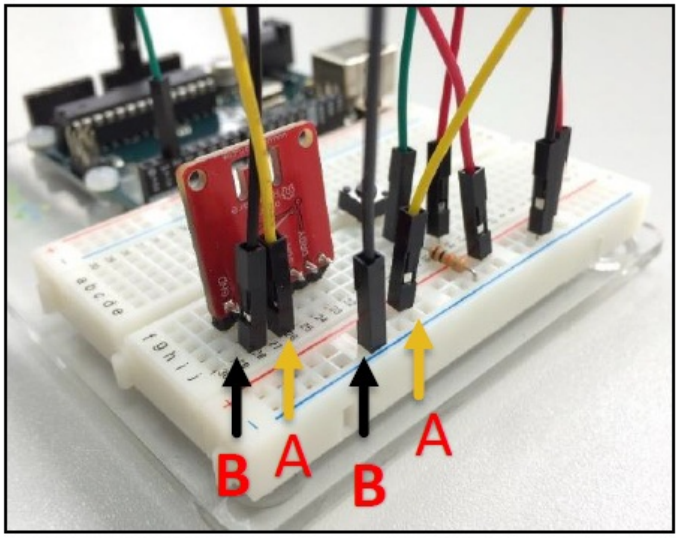


1. Connect the sensor and provide it with power and grounding
    - Connect the **sensor** to the *breadboard*.

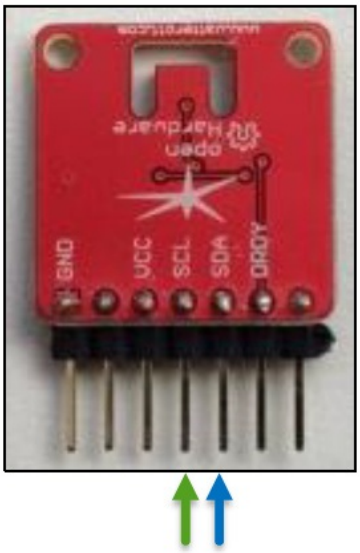


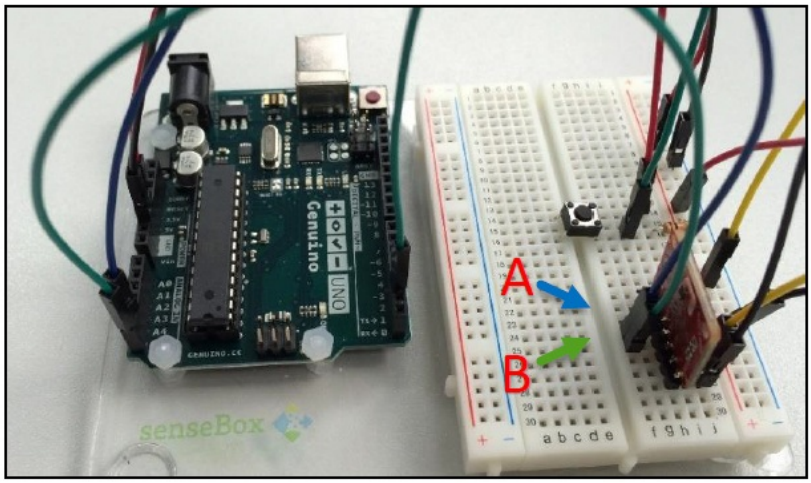
- The labels on the sensor mark tell which pins need to be connected to the power and ground. If you are unsure, please consult the data sheet for your sensor.
- Connect a yellow wire to the positive vertical row and to the VCC pin of the sensor (A).
- Connect a black wire to the negative vertical row and to the GND pin of the sensor (B).

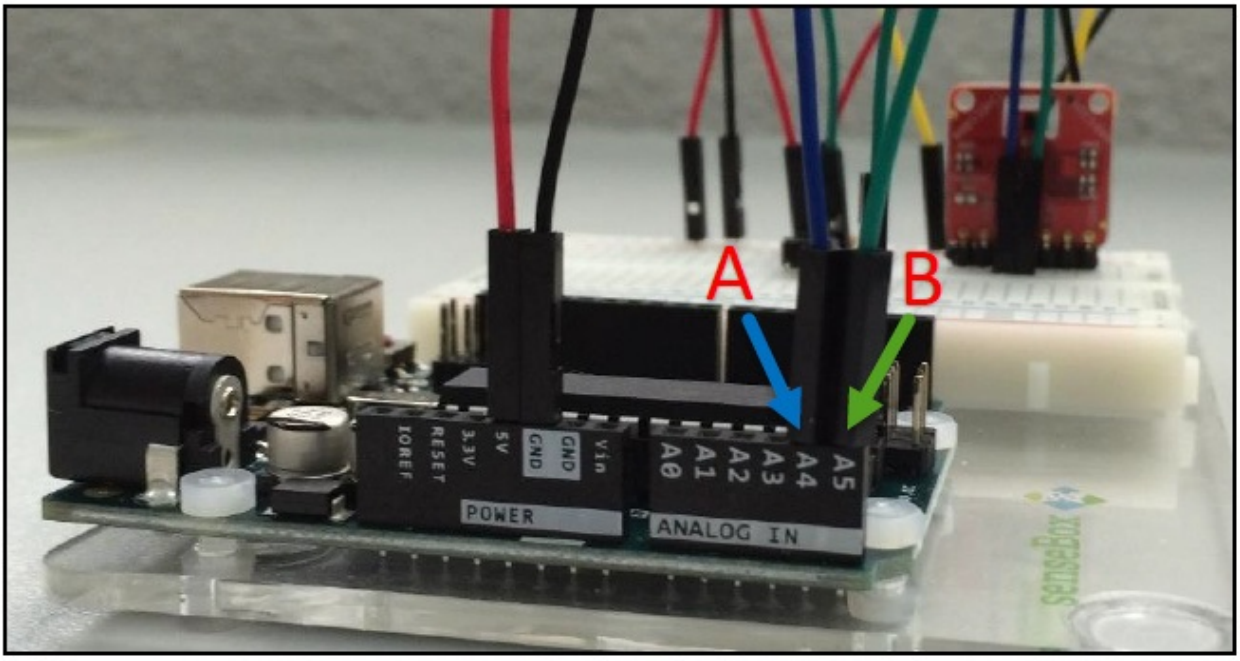




1. Create a connection between the sensor and the microcontroller to transmit measurements
   - Provide with a blue wire a connection between the **analog pin AV4** (you may also connect to another free analog pin; then check if you address the sensor in your code correctly) on the *Genuino Board* and the **SDA pin** of the sensor.
   - Provide with a green wire a connection between the **analog pin AV5** (you may also connect to another free analog pin; then check if you address the sensor in your code correctly) on the *Genuino Board* and the **SCL pin** of the sensor.







Write the program for the Weather Station

1. Create a new sketch
    - Start the Arduino IDE.
    - Go to File New and name the file PushbuttonTemperature.ino to create your own sketch.
1. Include the required libraries
    - Add the following code to your program to add two libraries:
        - Wire.h provides functions to communicate with the data line (SDA) pin and clock line (SCL) pin.
        - HDC100X.h makes communication and actions with the SD card possible.

        ```
        #include <Wire.h>;
        ```

        ```
        #include <HDC100X.h>;
        ```
1. Define global variables
    - Create a connection with the sensor and pushbutton.

- Define variables for the button state and id.

```
HDC100X HDC1(0x43); // create a connection to the sensor on address 0x43

int buttonPin = 2; // the number of the pin to which the pushbutton
// is connected, change this if you
// connected the button to a different port

int buttonState = 0; // variable for reading the status of the pushbutton

int lastButtonState = 0; // previous state of the pushbutton

int id = 0; // count for how many times the button has been pressed
// and set this as ID
```

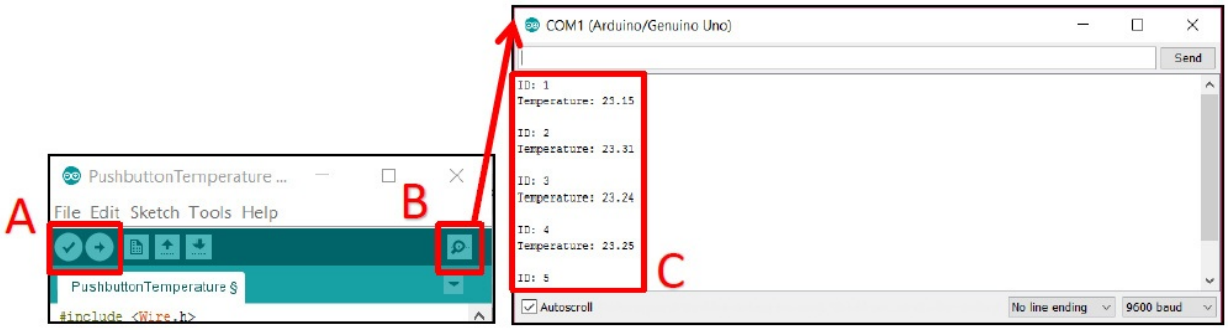
1. Write the setup() function
   - Start the sensor.
   - Initialize the pushbutton as an input.

```
void setup() {

Serial.begin(9600); // provide communication with the
// computer with data rate 9600 bits per second

HDC1.begin(HDC100X\_TEMP\_HUMI, HDC100X\_14BIT, HDC100X\_14BIT,
DISABLE); // start the sensor

pinMode(buttonPin,INPUT); // initialize the pushbutton pin
// as an input

}
```

1. Write the loop() function
   - Read the value from the push button.
   - If the button is pressed, measure the temperature and print it to the console.

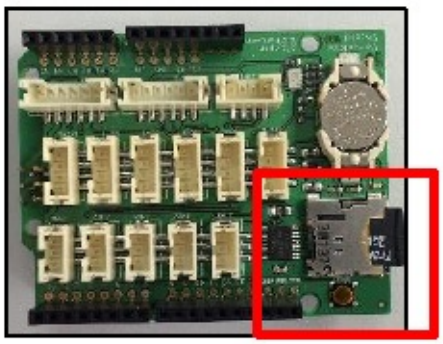
```
void loop() {

  buttonState = digitalRead(buttonPin); // read value from push button

  // if button is pressed in

  if (buttonState != lastButtonState && buttonState == HIGH) {

  id++; // update id

  Serial.print("ID: "); // print the id to the console

  Serial.println(id);

  Serial.print("Temperature: "); // get the temperature and
  // print it to the console

  Serial.println(HDC1.getTemp());

  Serial.print("\\n");

  }

  lastButtonState = buttonState; // set current state as last button state

}
```

1. Run your program
2. Compile your program and upload it to the board (A).
3. Click on the Serial Monitor button. The COM1 dialog should pop up (B).
4. Push the button on your board. A new result should appear on your screen (C).

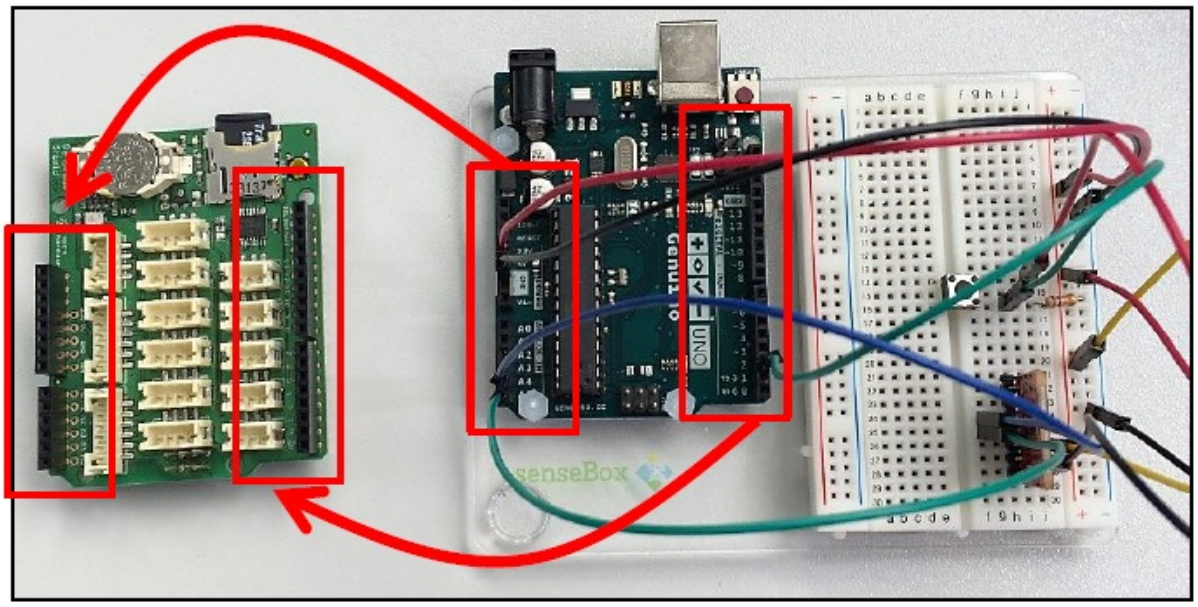


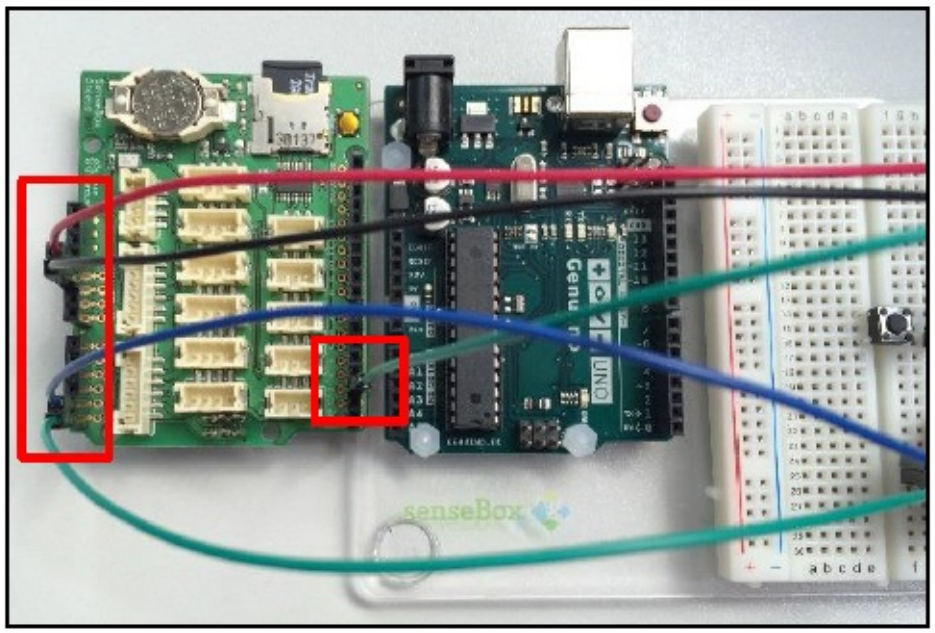
Write the results to a file on a SD card

1. What do you need?
    - Your Arduino construction from the previous steps.
    - A SenseBox Shield with SD card slot.

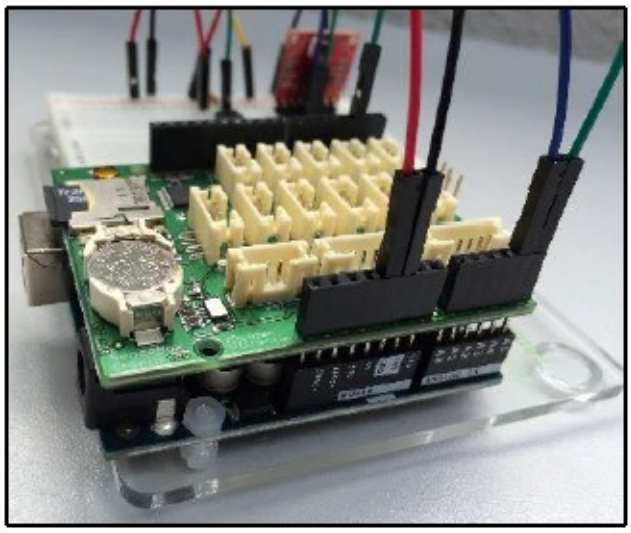


1. Connect the SenseBox Shield to the Genuino Board
    - Since the SenseBox Shield will be placed on top of the Genuino Board, the wires need to be connected to the Sensebox Shield.





2. Place the **SenseBox Shield** on the **Genuino Board**.



3. Adjust your program

- Open the file PushbuttonTemperature.ino in the Arduino IDE.
- Add a library for SD card functionalities:

```
#include <SD.h>
```

- Define a global variable which contains the identifier of the pin connected to the SD card.

```
int chipPin = 4; // pin connected to the SD card
```

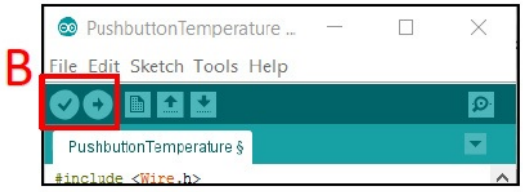
- Initialize the SD card in the **setup()** function.

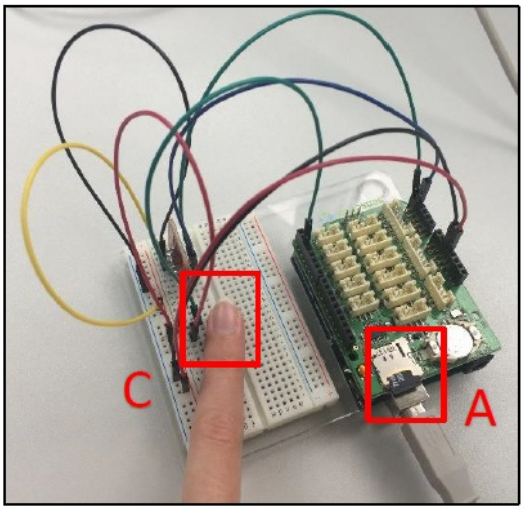
```
SD.begin(chipPin); // initialize the SD card.
```

- Adjust the loop() function.

```
void loop() {

  buttonState = digitalRead(buttonPin); // read value from push button

  // if button is pressed in

  if (buttonState != lastButtonState && buttonState == HIGH) {

    id++; // update id

    File file = SD.open("temp.csv", FILE\_WRITE); // open a file named temp.csv

    file.print(id); // print the id to the file

    file.print(";");

    file.print(HDC1.getTemp()); // get the temperature and print it to the file

    file.print("\\n");

    file.close(); // close the file

  }

  lastButtonState = buttonState; // set current state as last button state

}
```

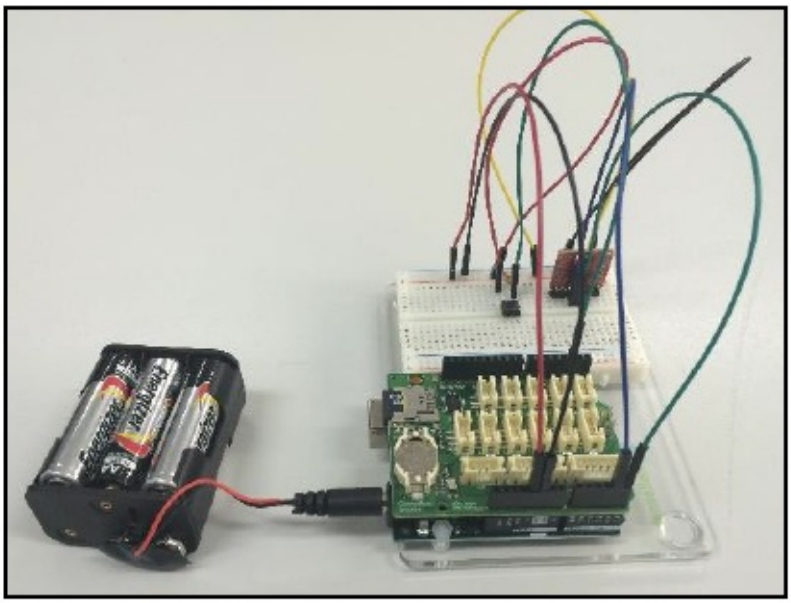
- Run your program
- Place the SD card in the slot on the SenseBox Board (A).
- Compile your program and upload it to the board (B).
- Push the button on the breadboard multiple times (C).





- View the results
  - Place the SD card in the card reader slot of your computer.
  - Open the file TEMP.CSV.
  - You can now see the resulting temperature measurements.

Make it mobile



Time to go on a field study and measure the temperatures throughout the building and surroundings. You can make your Arduino weather station more mobile by attaching a battery to the *Genuino Board*. Good luck during your expedition!

Downloads

In this area various downloads and helpful resources regarding the senseBox are listed.

Libraries

A package of all the required libraries for the various sensors we use is provided here. Note that we use a customized `Ethernet.h` library, as the stock-library is not compatible with our ethernet shield.

Datasheets

Most manufacturers provide datasheets for their sensors and other components, which provide specifications and further insights into the inner workings of the devices:

Sensor	Description	Manufacturer	Download
BMP280	air-pressure sensor	Bosch	datasheet
HC-SR04	supersonic distance-sensor	KT-Electronic	datasheet
HDC1008	temperature- & humidity-sensor	Texas Instruments	datasheet
TSL4531	digital lightintensity sensor	TAOS Texas Advanced Optoelectronics Solutions	datasheet
VEML6070	UV-light sensor	VISHAY	datasheet
GP2Y0A21YK0F	IR distance sensor	Sharp	datasheet

Documentation as PDF

This book is also available as printable PDF!