

Lab 1: Arduino Basics

Rodrigo Carbajales and Marco Zennaro
ICTP Trieste-Italy

Step Zero

Clean up your desks! :)

Goals of this Lab

- Learn how the programming takes place
- Exercises about:
 - installing the IDE
 - setting the clock
 - measuring the temperature
 - timestamping the temperature reading
 - saving the data on the SD card

Getting started

We will:

- 1) Download and install the Arduino IDE
- 2) Install the SODAQ Mbili files and libraries
- 3) Select the SODAQ Mbili hardware profile
- 4) Configure the serial port

Installing the IDE

- IDE= Integrated development environment
- Arduino IDE is Open Source

Download the Arduino IDE (Integrated Development Environment)



Access the Internet

In order to get your Arduino up and running, you'll need to download some software first from www.arduino.cc (it's free!). This software, known as the Arduino IDE, will allow you to program the Arduino to do exactly what you want. It's like a word processor for writing programs. With an internet-capable computer, open up your favorite browser and type in the following URL into the address bar:

< case sensitive >

Installing the IDE



Windows Installation Process

Go to the web address below to access the instructions for installations on a Windows-based computer.

[*http://arduino.cc/en/Guide/Windows*](http://arduino.cc/en/Guide/Windows)



Macintosh OS X Installation Process

Macs do not require you to install drivers. Enter the following URL if you have questions. Otherwise proceed to next page.

[*http://arduino.cc/en/Guide/MacOSX*](http://arduino.cc/en/Guide/MacOSX)



Linux: 32 bit / 64 bit, Installation Process

Go to the web address below to access the instructions for installations on a Linux-based computer.

[*http://www.arduino.cc/playground/Learning/Linux*](http://www.arduino.cc/playground/Learning/Linux)

Installing the Arduino IDE

After downloading and installing the software, you need to run the Arduino IDE once for it to create a the sketchbook directory called Arduino inside your **documents** folder. You can then close the program.

Installing the SODAQ Mbili files

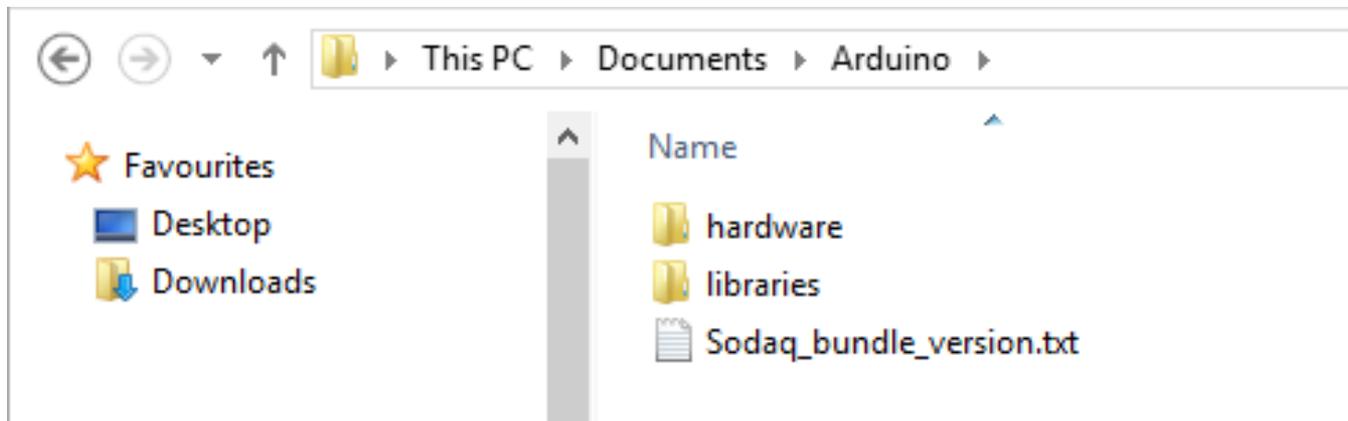
The next step is to install the SODAQ Mbili files (in the lab1.zip file)

You must unpack this zip file and place the contents in the Arduino sub-folder of your documents folder (the folder that was created by the Arduino IDE the first time it was run).

In Windows it is located in: C:
\Users\yourusername\Documents\Arduino\

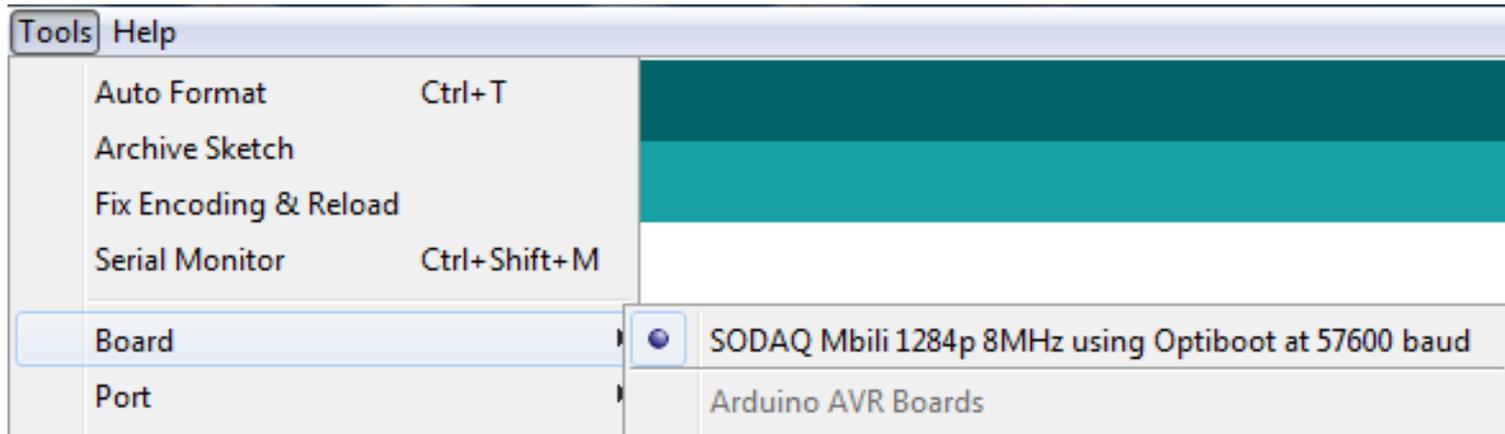
Installing the SODAQ Mbili files

The contents of the Arduino folder should now look like this:



Selecting the SODAQ hw profile

Select the SODAQ board in Tools → Board



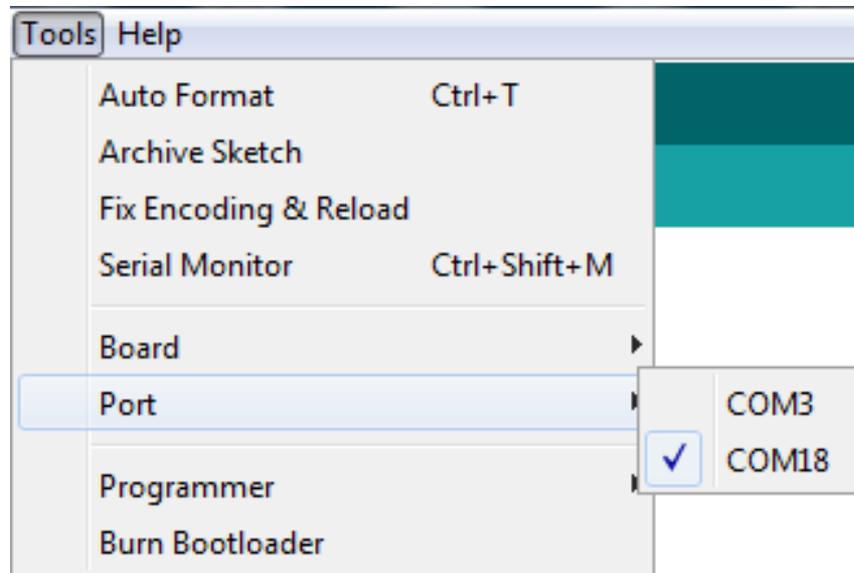
Configuring the serial port

Windows versions 7 and 8 will normally find the right USB driver when you plug in the SODAQ Mbili for the first time. The same is also true for Mac and Linux. If your system doesn't find the driver you will have to download the FTDI drivers.

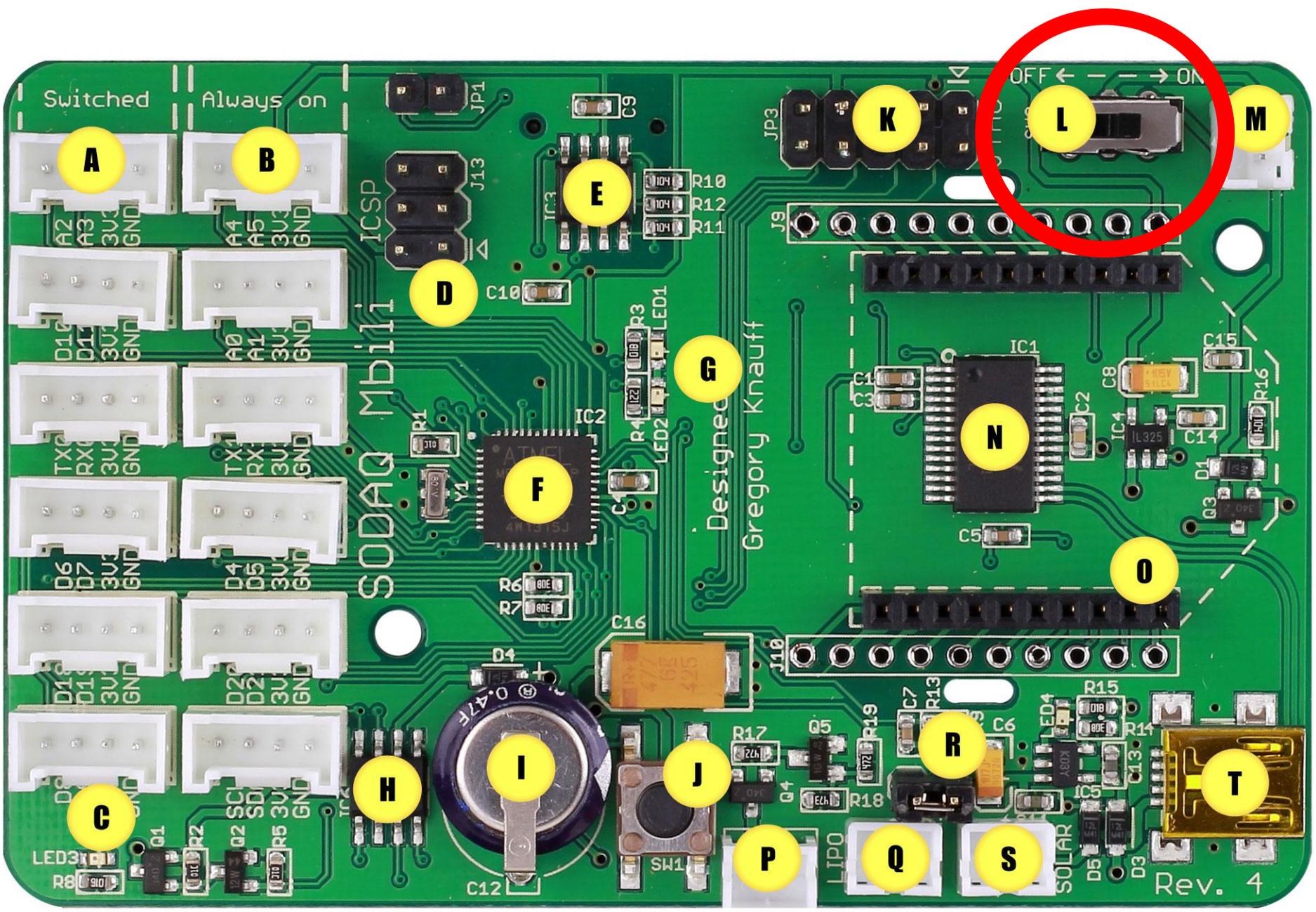
The FTDI driver adds a virtual serial port. In Windows this is `_COMx_` (so `_COM1`, `COM8`, `_` etc.).

Configuring the serial port

Select the serial port in Tools→ Port



The associated serial port is only visible in that list when the SODAQ Mbili board is **connected** and **switched on**.



A

B

D

E

K

L

M

G

F

N

O

C

H

I

J

P

R

Q

S

T

Switched

Always on

ICSP

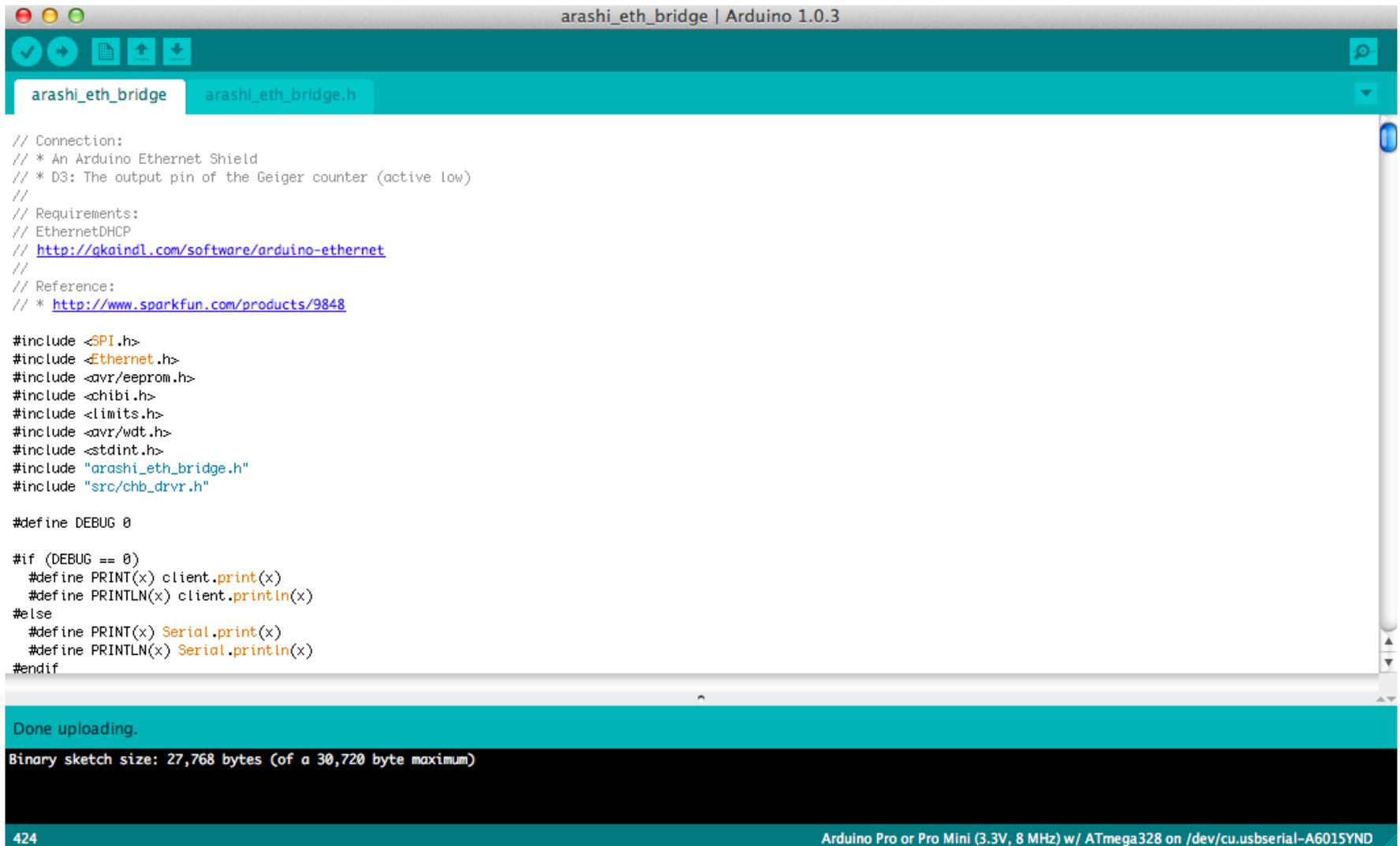
SODAQ Mbit11

Designed by
Gregory Knauft

OFF ← → ON

Rev. 4

The Arduino IDE



```
arashi_eth_bridge | Arduino 1.0.3
arashi_eth_bridge arashi_eth_bridge.h

// Connection:
// * An Arduino Ethernet Shield
// * D3: The output pin of the Geiger counter (active low)
//
// Requirements:
// EthernetDHCP
// http://gkaindl.com/software/arduino-ethernet
//
// Reference:
// * http://www.sparkfun.com/products/9848

#include <SPI.h>
#include <Ethernet.h>
#include <avr/eeprom.h>
#include <chibi.h>
#include <limits.h>
#include <avr/wdt.h>
#include <stdint.h>
#include "arashi_eth_bridge.h"
#include "src/chb_drvr.h"

#define DEBUG 0

#if (DEBUG == 0)
  #define PRINT(x) client.print(x)
  #define PRINTLN(x) client.println(x)
#else
  #define PRINT(x) Serial.print(x)
  #define PRINTLN(x) Serial.println(x)
#endif
```

Done uploading.

Binary sketch size: 27,768 bytes (of a 30,720 byte maximum)

424 Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328 on /dev/cu.usbserial-A6015YND

Programming an Arduino

From the File menu, choose Open and select the code you want to open.

The source code will appear in the IDE window.

Lab Examples

From the Workshop's webpage,
download the zip file with all the examples
for this Lab 1 Session.

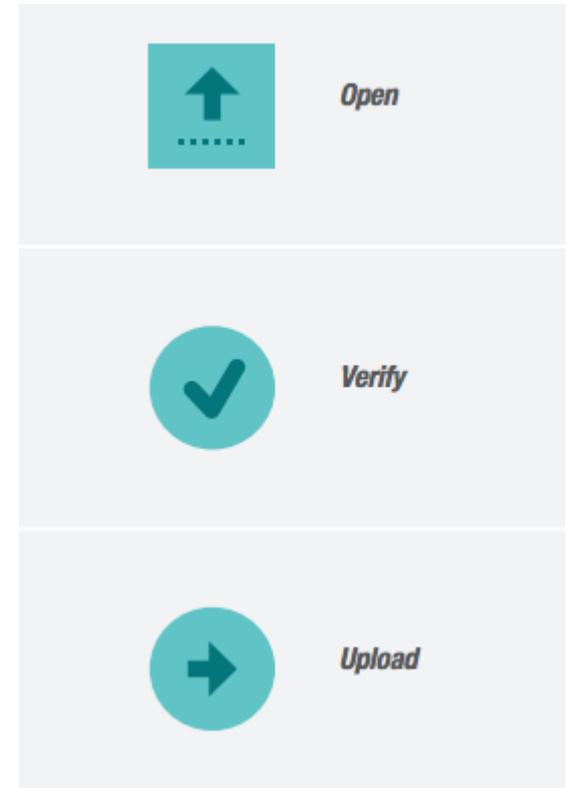
Open the folder called Lab1.1
and open the **Hello_world.ino** file

Programming workflow

1. Opening

2. Verifying

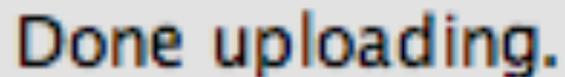
3. Uploading



Programming an Arduino

Click on the upload button and wait until the code has been compiled and uploaded.

At the end you will see in the bottom right corner:



Done uploading.

Programming an Arduino

This is the template of a basic Arduino program:

```
void setup()  
{  
  Initialize variables, open USB, open WiFi, etc  
}
```

SETUP
(once)

```
void loop()  
{  
  Perform some action  
  
  Wait for a certain number of msecs or wait for an alarm  
}
```

LOOP
(forever)

Lab session

This lab session will be like this:

```
For (i=1;i<=3;i++) {  
    Simple example (me) /* 2 min */  
    Extended example (you) /* 20 min */  
}
```

```
Real-world exercise /* 1 hour */
```

Start!



Example 1

Hello_world.ino will write Hello World on the serial port.

```
void setup()
{
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Starting...");
}

void loop()
{
  // put your main code here, to run repeatedly:
  Serial.println("Hello World");
  delay(1000);
}
```

Example 1

How do you see the output of your code?

Select Serial Monitor (via Tools → Serial):

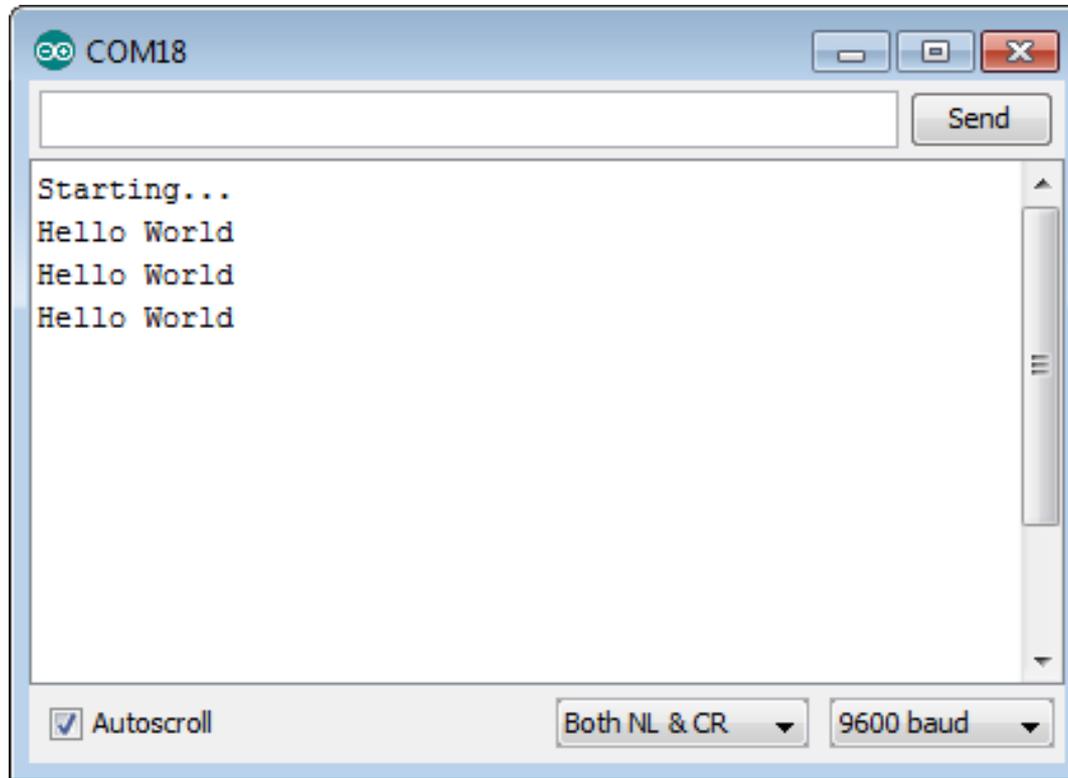


Note the Serial Baud Rate that is set in the code!

Your Serial Monitor needs to match that!

Example 1

You will see the following:



Example 1 - extended

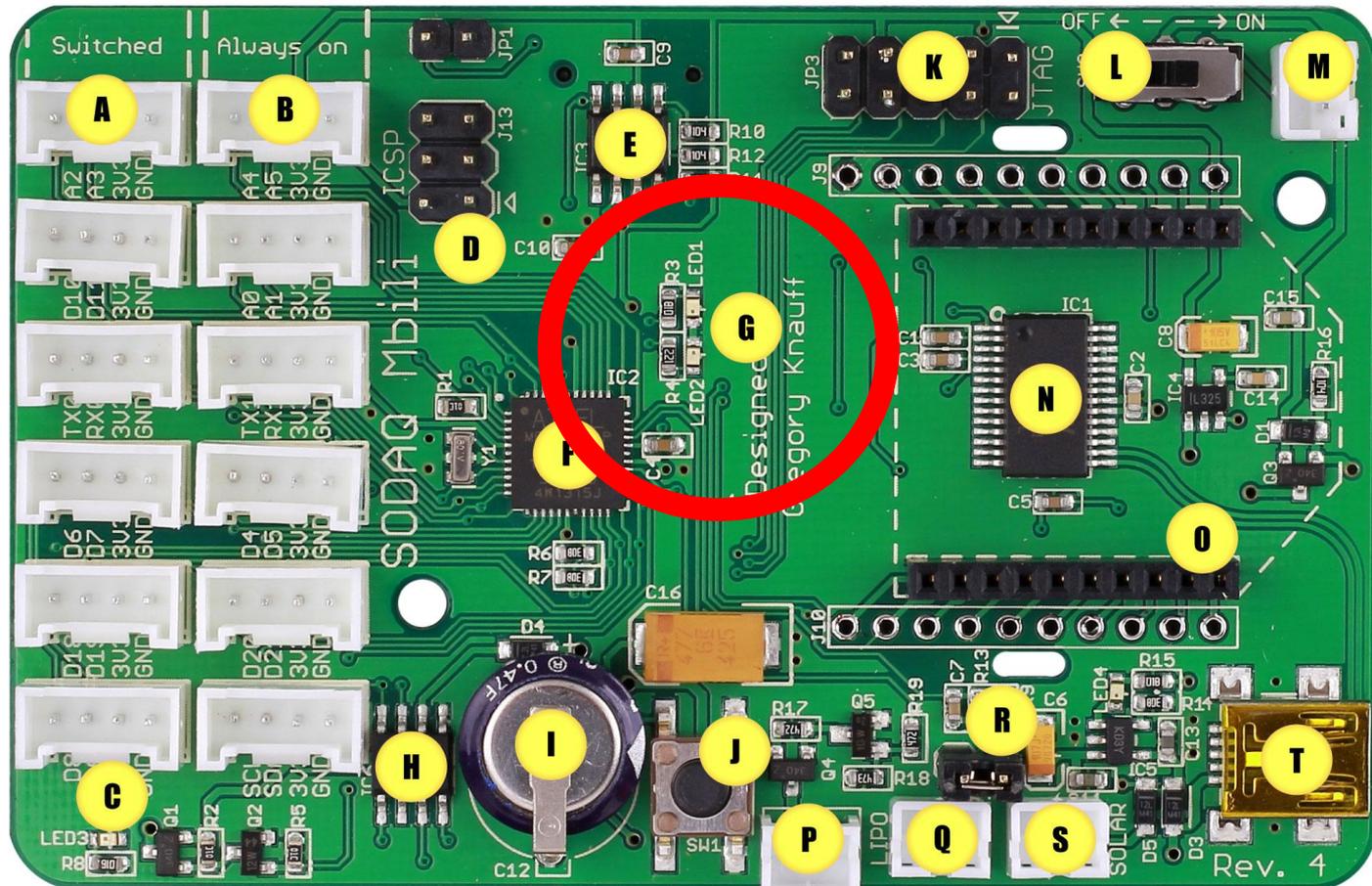
Get acquainted with the IDE.

Try to write something else.

Change the delay.

Example 2

Leds1_2.ino will blink LED1(green) and LED2(red).



Example 2

Leds1_2.ino will blink LED1(green) and LED2(red).

```
//How long to activate each LED
#define DELAY_TIME 1000

void setup()
{
  //LED1
  pinMode(LED1, OUTPUT);
  digitalWrite(LED1, LOW);

  //LED2
  pinMode(LED2, OUTPUT);
  digitalWrite(LED2, LOW);
}
```

Example 2

Leds1_2.ino will blink LED1(green) and LED2(red).

```
void loop()
{
  //Switch LED1 on then off again after DELAY_TIME (ms)
  digitalWrite(LED1, HIGH);
  delay(DELAY_TIME);
  digitalWrite(LED1, LOW);

  //Repeat for LED2
  digitalWrite(LED2, HIGH);
  delay(DELAY_TIME);
  digitalWrite(LED2, LOW);
}
```

Example 2 - extended

Make the LEDs blink as fast as possible.

Make the LEDs blink at the same time.

Write SOS in Morse Code (...---...).

Real Time Clock (RTC)

Having a RTC is useful for two reasons:

1. to time stamp the collected data (for example: temperature is 27.4C at 10:02:30 of 6/7/2020)
2. to be able to set alarms to wake up the mote from sleeping mode (for example: wake up on Tuesday 15th of August at 10:30:00).

SODAQ has a RTC!

Example 3

RTC_date_time_update.ino will set the time of the SODAQ using the RTC.

To program the RTC you need to download some libraries first. In the zip file you will find the DS3231 and Wire libraries.

Make sure the files in folder labelled “Sodaq_DS3231” and “Wire” are into your Arduino libraries folder.

Restart the IDE!

Example 3

RTC_date_time_update.ino will set the time of the Seeduino using the RTC.

Change the line:

DateTime dt(2011, 11, 10, 15, 18, 0, 5);

to adjust to today's date and time.

Format is: year, month, date, hour, min, sec and week-day (starts from 0 (Sunday) and goes to 6 (Saturday))

Example 3

RTC_date_time_update.ino will set the RTC and then show it.

```
/Include the necessary libraries  
#include <Wire.h>  
#include <Sodaq_DS3231.h>
```

```
Sodaq_DS3231 RTC; //Create RTC object for DS3231  
DateTime dt(2014, 06, 05, 11, 5, 00, 4);
```

Example 3

```
void setup()
{
  //Start serial
  Serial.begin(9600);
  Serial.println("Date,   Time");

  //Start the I2C protocol
  Wire.begin();

  //initialize the DS3231
  RTC.begin();
  RTC.setDateTime(dt); //Adjust date-time as defined 'dt' above
}
```

Example 3

```
void loop()
{

    String data = getDateTime();
    Serial.println(data);
}

String getDateTime()
{
    String dateTimeStr;

    //Create a DateTime object from the current time
    DateTime dt(RTC.makeDateTime(RTC.now().getEpoch()));

    //Convert it to a String
    dt.addToString(dateTimeStr);

    return dateTimeStr;
}
```

Example 3

Important:

Once the RTC has been set, line

```
RTC.setDateTime(dt);
```

has to be commented in order not to update every reboot of the board.

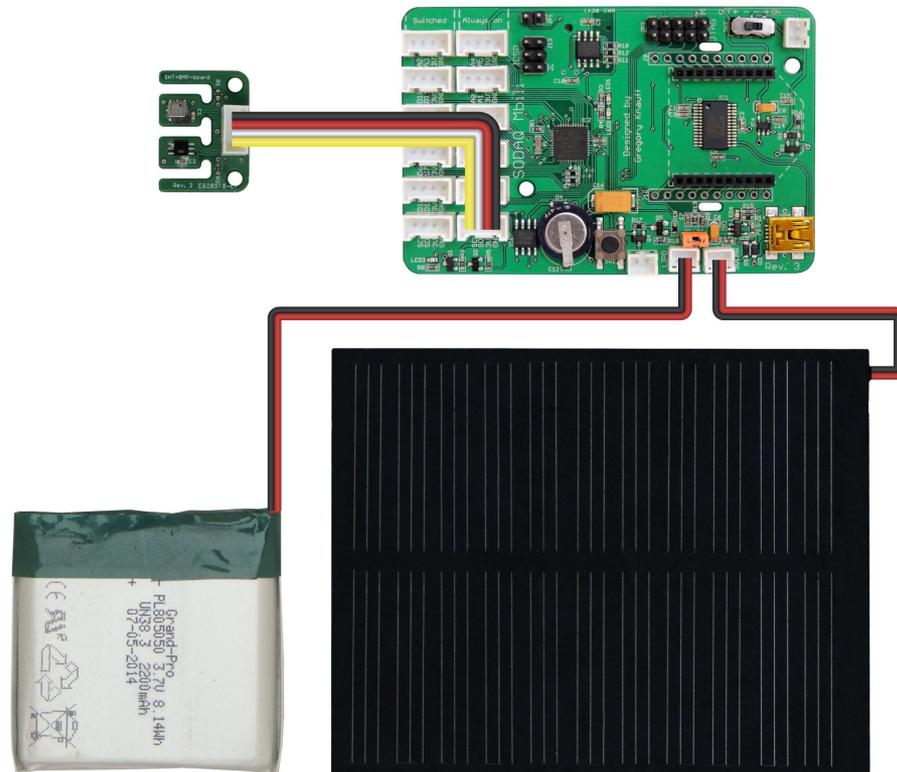
Example 3 - extended

Comment the line where you set the time. Is the time OK?

Disconnect the SSODAQ from the USB. Wait some minutes. Connect it again. Are the date and time OK?

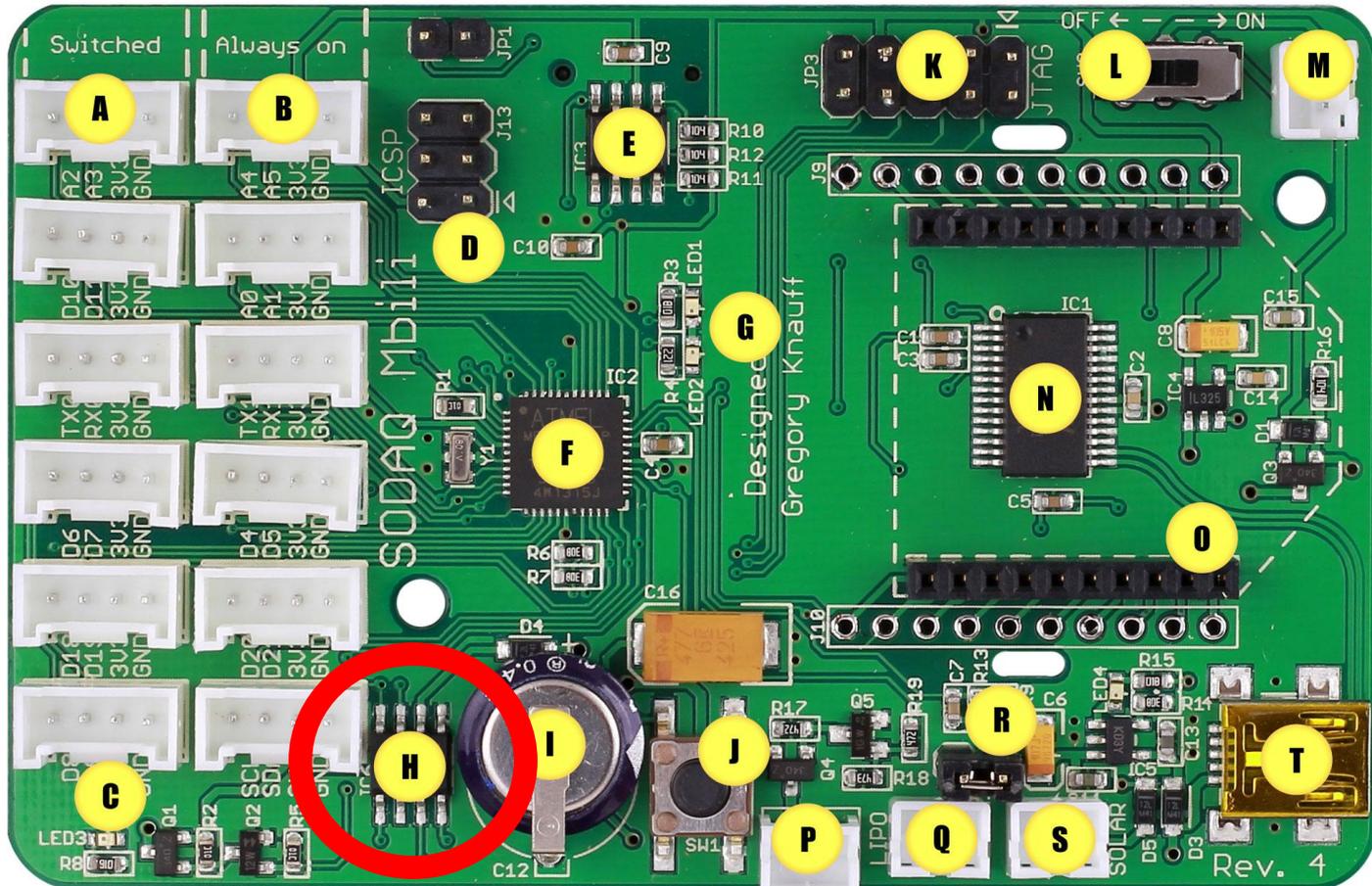
Example 4

RTC_date_Volt_Temp.ino will read the temperature of the RTC. It will also read the voltage level of the external battery. You don't need the TPH sensor yet.



Example 4

The RTC is shown here:



Example 4

RTC_date_Volt_Temp.ino will read the RTC temperature and the battery voltage.

```
void loop()
{
  ///Read the temperature
  RTC.convertTemperature();
  float temp = RTC.getTemperature();

  // Convert temperature voltage to string
  char buffer[14]; //make buffer large enough for 7 digits
  String temperatureS = dtostrf(temp, 7,2,buffer);
  temperatureS.trim();

  //Read the voltage
  int mv = getRealBatteryVoltage() * 1000.0;
```

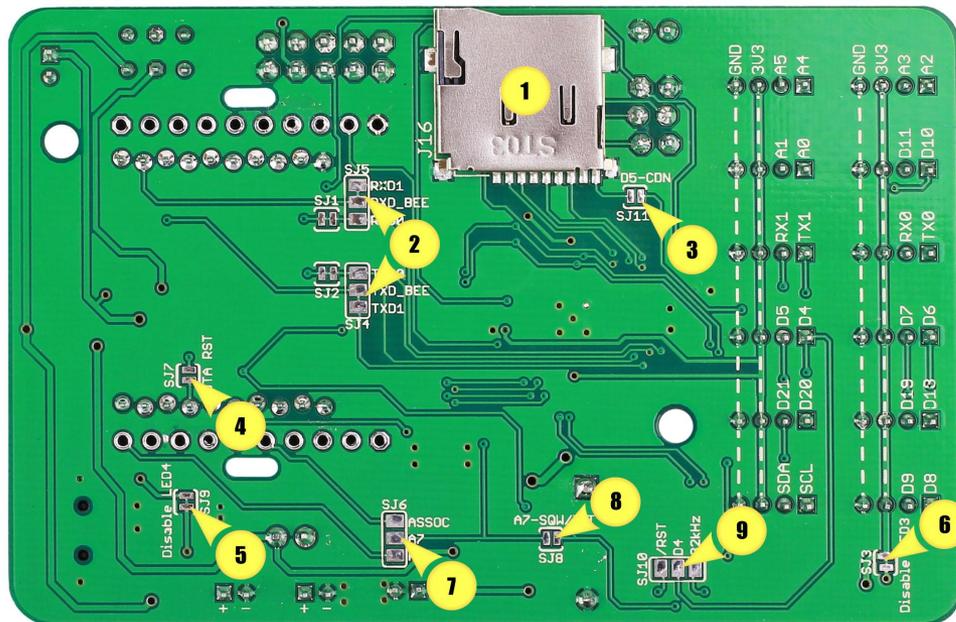
Example 4 - extended

Touch the RTC and check if the temperature goes up or down.

Convert the temperature to Fahrenheit and show values in both C and F.

Example 5

SD_write.ino will write a string to the μ SD card.
The SPI and SD Libraries come pre-installed with the Arduino IDE, so there is no need to install them.
Insert a μ SD card in the slot 1.



Example 5

SD_write.ino will write a string to the μ SD card.

```
//Digital pin 11 is the MicroSD slave select pin on the Mbili
#define SD_SS_PIN 11

//The data log file
#define FILE_NAME "DataLog.txt"

//Data header
#define DATA_HEADER "Hello world"
```

Example 5

SD_write.ino will write a string to the μ SD card.

```
void loop()
{
  //Create the data record
  String dataRec = createDataRecord();

  //Save the data record to the log file
  logData(dataRec);

  //Echo the data to the serial connection
  Serial.println(dataRec);

  //Wait before taking the next reading
  delay(READ_DELAY);
}
```

Example 5

SD_write.ino will write a string to the μ SD card.

```
String createDataRecord()  
{  
  //Create a String type data record in csv format  
  
  String data = "1st East-African Workshop on the Internet Of Things";  
  return data;  
}
```

Example 5 - extended

Try to write something else.

Remove the μ SD card and read it with a card reader. Are the data formatted OK?

Example 6

RTC_date_Volt_Temp_SD.ino will write a string to the μ SD card with date, temperature and voltage readings.

The log file will consist of Comma Separated Values (CSV) in ASCII format.

This type of application is called **data logger**.

Example 6

```
String createDataRecord()
{
    //Create a String type data record in csv format
    ///Read the temperature
    RTC.convertTemperature();
    float temp = RTC.getTemperature();
    // Convert temperature voltage to string
    char buffer[14]; //make buffer large enough for 7 digits
    String temperatureS = dtostrf(temp, 7,2,buffer);
    temperatureS.trim();
    //Read the voltage
    int mv = getRealBatteryVoltage() * 1000.0;

    String data = getDateTIme()+ ", ";
    data += String(temperatureS)+ "C, ";
    data += String(mv)+ "mV";
    return data;
}
```

Example 7

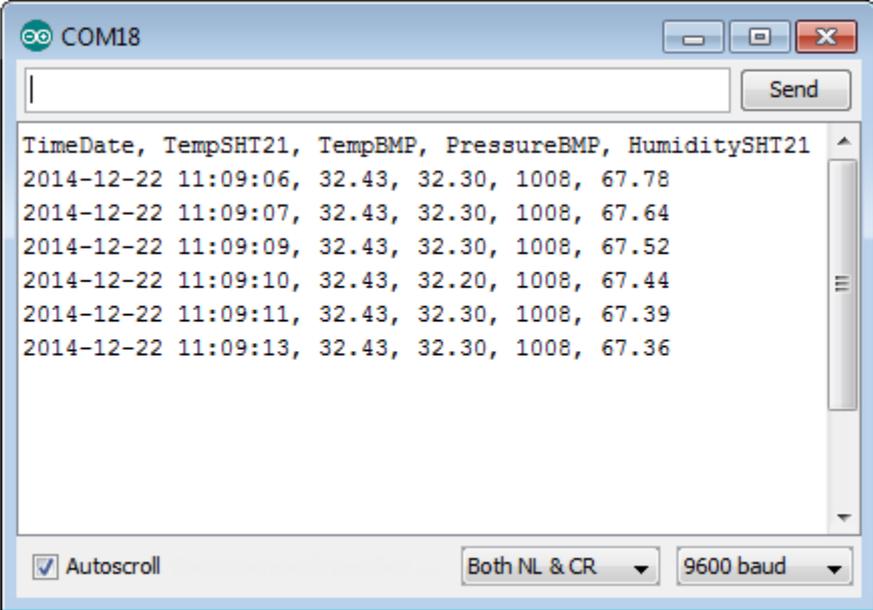
RTC_TPH_SD_Tiner.ino will demonstrate the use of a **RTCTimer** to schedule regular events.

This example builds on the previous example, but instead of using the `delay()` method, it uses a scheduling timer to control the frequency of the readings.

The required RTCTimer library is included with the SODAQ Mbili files that you have already installed.

Example 7

The `delay()` method pauses the execution for a specified number of msecs → delays due to sum of execution time and delay.



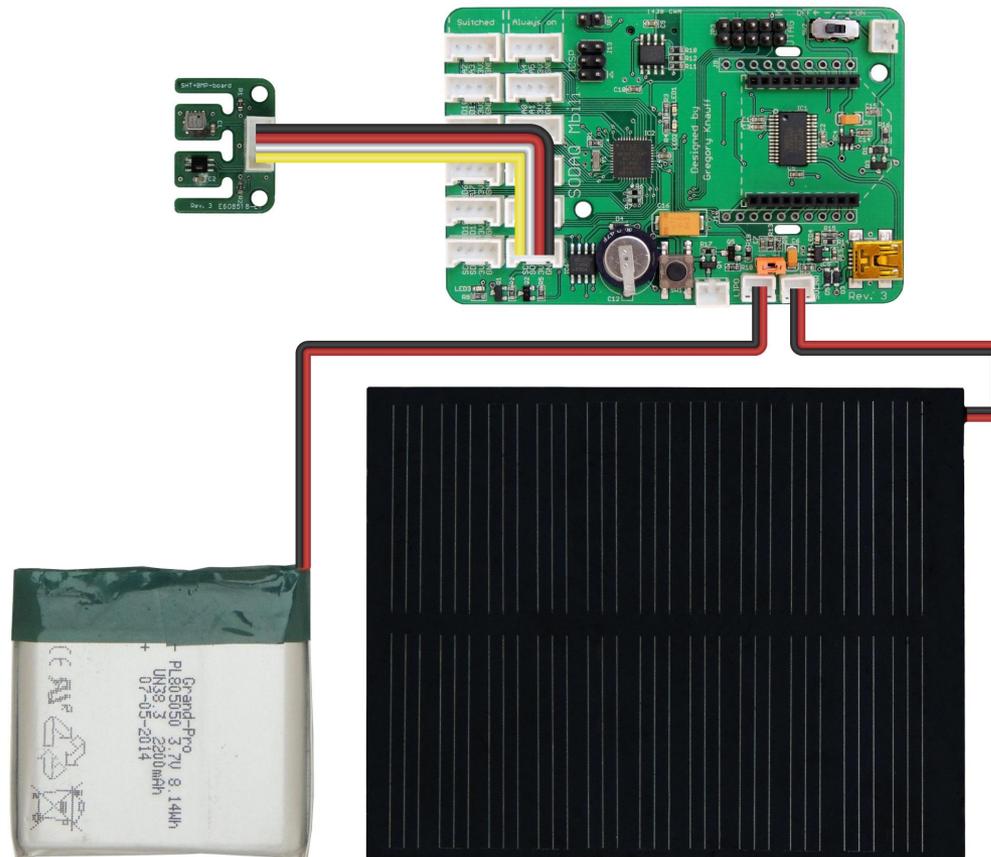
The screenshot shows a serial monitor window titled "COM18". It features a text input field at the top with a "Send" button to its right. Below the input field, the monitor displays a series of sensor readings in a tabular format. The data is as follows:

TimeDate	TempSHT21	TempBMP	PressureBMP	HumiditySHT21
2014-12-22 11:09:06,	32.43,	32.30,	1008,	67.78
2014-12-22 11:09:07,	32.43,	32.30,	1008,	67.64
2014-12-22 11:09:09,	32.43,	32.30,	1008,	67.52
2014-12-22 11:09:10,	32.43,	32.20,	1008,	67.44
2014-12-22 11:09:11,	32.43,	32.30,	1008,	67.39
2014-12-22 11:09:13,	32.43,	32.30,	1008,	67.36

At the bottom of the window, there are three controls: a checked "Autoscroll" checkbox, a dropdown menu set to "Both NL & CR", and another dropdown menu set to "9600 baud".

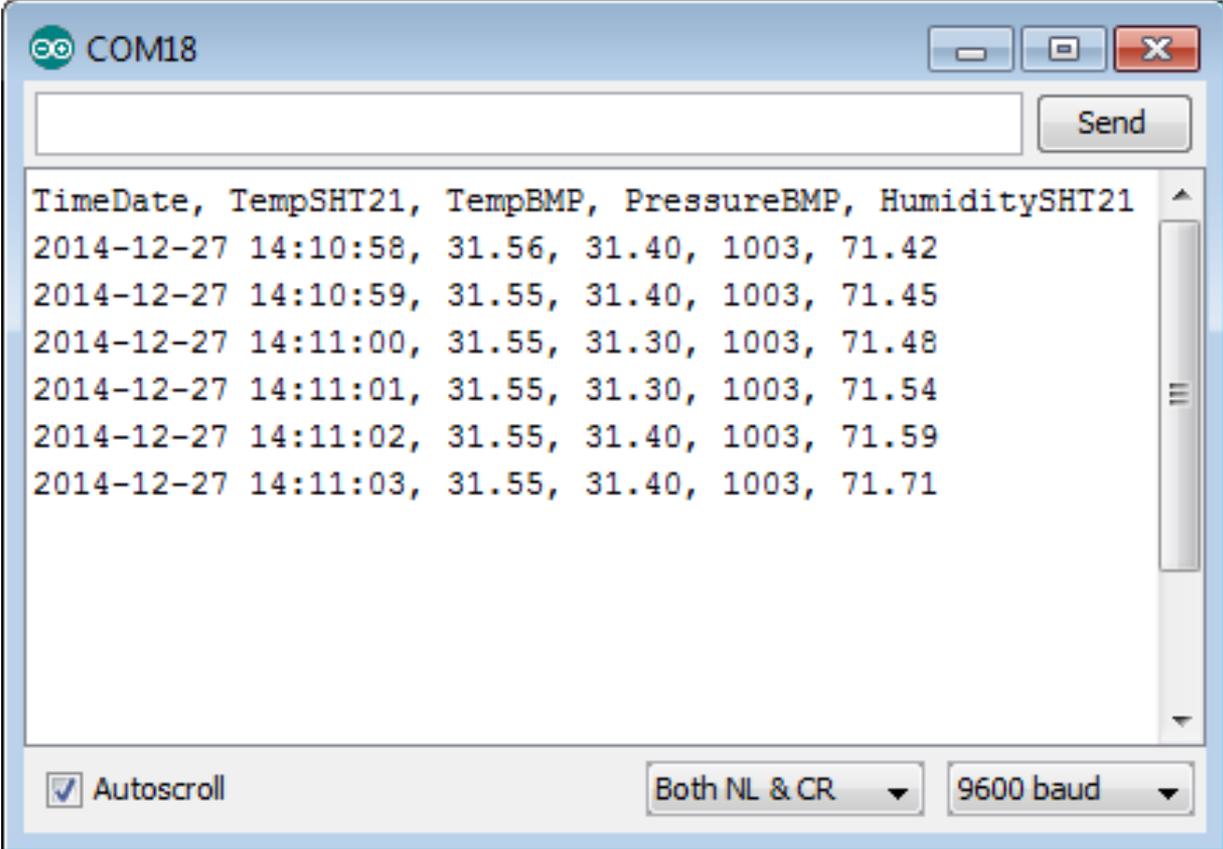
Example 7

We will read data from the TPH sensor, which is more precise than the internal RTC temperature sensor.



Example 7

Readings are now every second! (no delays)



The screenshot shows a serial terminal window titled "COM18". The window contains a list of sensor readings, each on a new line, separated by commas. The readings are as follows:

TimeDate	TempSHT21	TempBMP	PressureBMP	HumiditySHT21
2014-12-27 14:10:58	31.56	31.40	1003	71.42
2014-12-27 14:10:59	31.55	31.40	1003	71.45
2014-12-27 14:11:00	31.55	31.30	1003	71.48
2014-12-27 14:11:01	31.55	31.30	1003	71.54
2014-12-27 14:11:02	31.55	31.40	1003	71.59
2014-12-27 14:11:03	31.55	31.40	1003	71.71

The window also features a "Send" button at the top right, a scroll bar on the right side, and a status bar at the bottom with the following settings: Autoscroll, Both NL & CR, and 9600 baud.

Example 7 - extended

Compare the RTC temperature readings and the ones given by the TPH sensor.

Convert the temperature to Fahrenheit and save values in both C and F.

Thanks

Marco Zennaro and Rodrigo Carbajales

mzennaro@ictp.it

<http://wireless.ictp.it>