ArduinoExcel 3.0

rev 1.1 November 2021

by Roberto Valgolio

Introduction

Arduino Excel (former Arduino Excel Commander) is a powerful interface between Arduino and MS Excel that supports data exchanging in both directions.

Excel can represent real time data from sensors or it can be used as an extern database to overcome Arduino memory limitations. The main purposes are:

- data harvesting and consolidation
- monitoring activities with email alerts
- support for experiments or advanced applications (eg: robotic devices driven by Arduino)

Arduino Excel is typically used in prototypes but even in some professional applications for scientific experiments or industrial data harvesting accomplished with cheap hardware.

The main features are:

- data writing to any worksheet / cell
- data retrieving from any worksheet / cell
- email sending for alarms or notifications
- CSV files writing

Up to four Arduino can be connected at the same time thru USB ports.

The logic is built in the Arduino sketch with simple instructions like:

```
// write the x variable value to worksheet 'Example' range 'B5' with two digits as decimals
myExcel.write("Example", "B5", x, 2);
or
// get the value from worksheet 'Test' range 'A3' and put it in y variable
ret = myExcel.get("Test", "A3", y);
```

Find more documentation in the sketch supplied as example.

History: the project started in 2015, at beginning 2020 about 5000 users have worked with it especially in education but even in scientific or industrial environments. Top user countries are USA, Brazil, EU.

Coming soon: a new pro version with TCP an MQTT protocols is under study as an interface to SQL databases, stay in touch.

Source code: see on GitHub https://github.com/rvalgolio/ArduinoExcel_30

Contacts: if you have questions or you are interested on professional developments based on customized hardware and software please contact roberto.valgolio@gmail.com.
Students, makers, hobbyists are welcome and supported provided they have a basic knowledge on Arduino and C++.

Installing

As prerequisites ArduinoExcel needs:

- Arduino IDE from https://www.arduino.cc/
- Microsoft Excel
- Microsoft Outlook (option)

The system works with any Windows or Excel version but on some old Windows you must to follow the instructions in Appendix C for setup.

First step

- download the setup procedure from https://github.com/rvalgolio/ArduinoExcel_30/tree/main/Setup
- the regular installation tool is *Arduino_Excel_Setup.exe* but if this doesn't work (maybe you have an old Windows version) please see Appendix C
- IMPORTANT: launch *Arduino_Excel_Setup.exe* as Admnistrator (click the exe with the right mouse button and choose 'Run as Administrator')

Second step (Arduino side)

- reopen the Arduino IDE if it were already opened
- if you have Arduino Due or Intel Edison comment the line 17 in *Arduino\libraries\rExcel.h*: //#define ATMEL_COMPATIBLE
- open *Documents\Arduino\Arduino_Excel_30* and compile the sketch *Arduino_Excel_30.ino* in your Arduino projects (you can see output strings thru the serial monitor set at 115200)

Third step (Excel side)

- open *Documents\Arduino_Excel\Arduino_Excel_30.xls* (the file can be located where you like)
- allow macro execution (needed only at first run, look Appendix A for pictures)
    - Office 2003
        - on menu Tools, Macro, Protection, select Low as protection level (this allows VBA code to work)
        - on menu Tools, Macro, Trusted Sources tab, set 'Trust access to Visual Basic Project'
    - Office 2013 or following
        - on File, Options, Trust Center, Trust Center Settings, Macro settings
        - Allow all Macro and check 'Trust access to the VBA project object model'
- close Excel app and reopen it
- check the VBA environment
    - press the keys CTRL and 'a', if you get an error please refer to Appendix B of this document

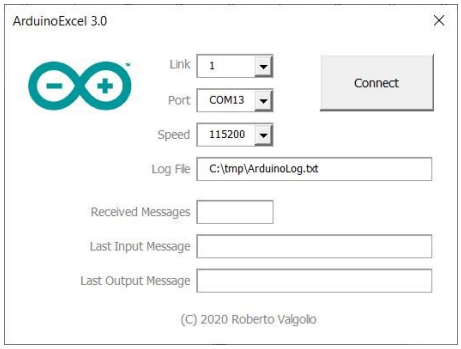
Now you're ready to go!

The installation procedure set the following folders and contents:

Documents\Arduino_Excel	*Arduino_Excel_30.xls*
Documents\Arduino\Arduino_Excel_30	*Arduino_Excel_30.ino*
Documents\Arduino\libraries\rExcel	*library*

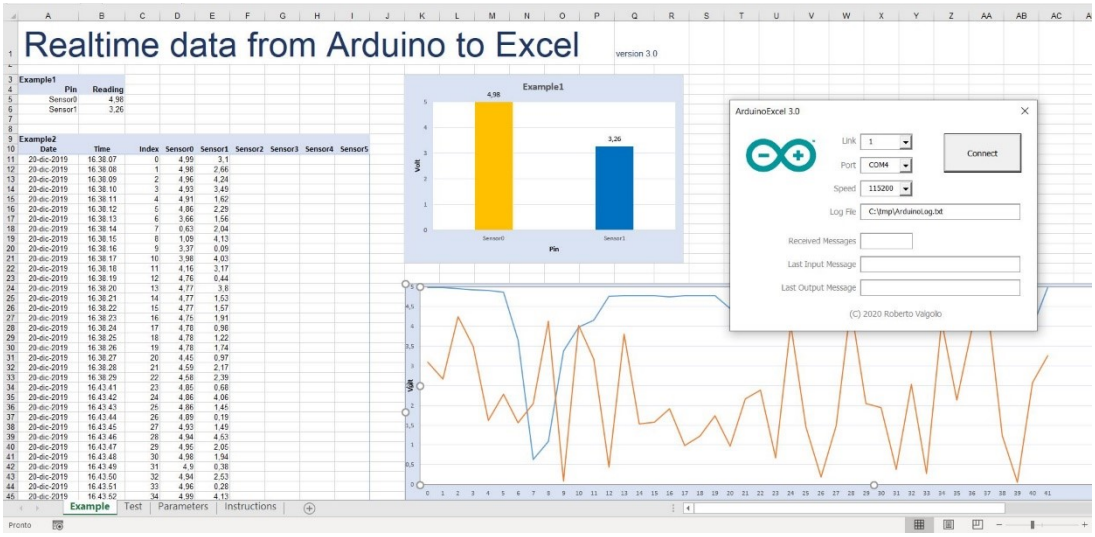
Getting started

- connect an Arduino board to a USB port of your PC (up to four Arduino can be connected at the same time)
- open *Arduino_Excel_30.xls*
- press the keys CTRL and 'a', you should see the launch form [(1)]



- select a link (a connection with an Arduino if more are present)
- select the port name
- select the port speed
- press Connect button

after some seconds you should see data from Arduino.

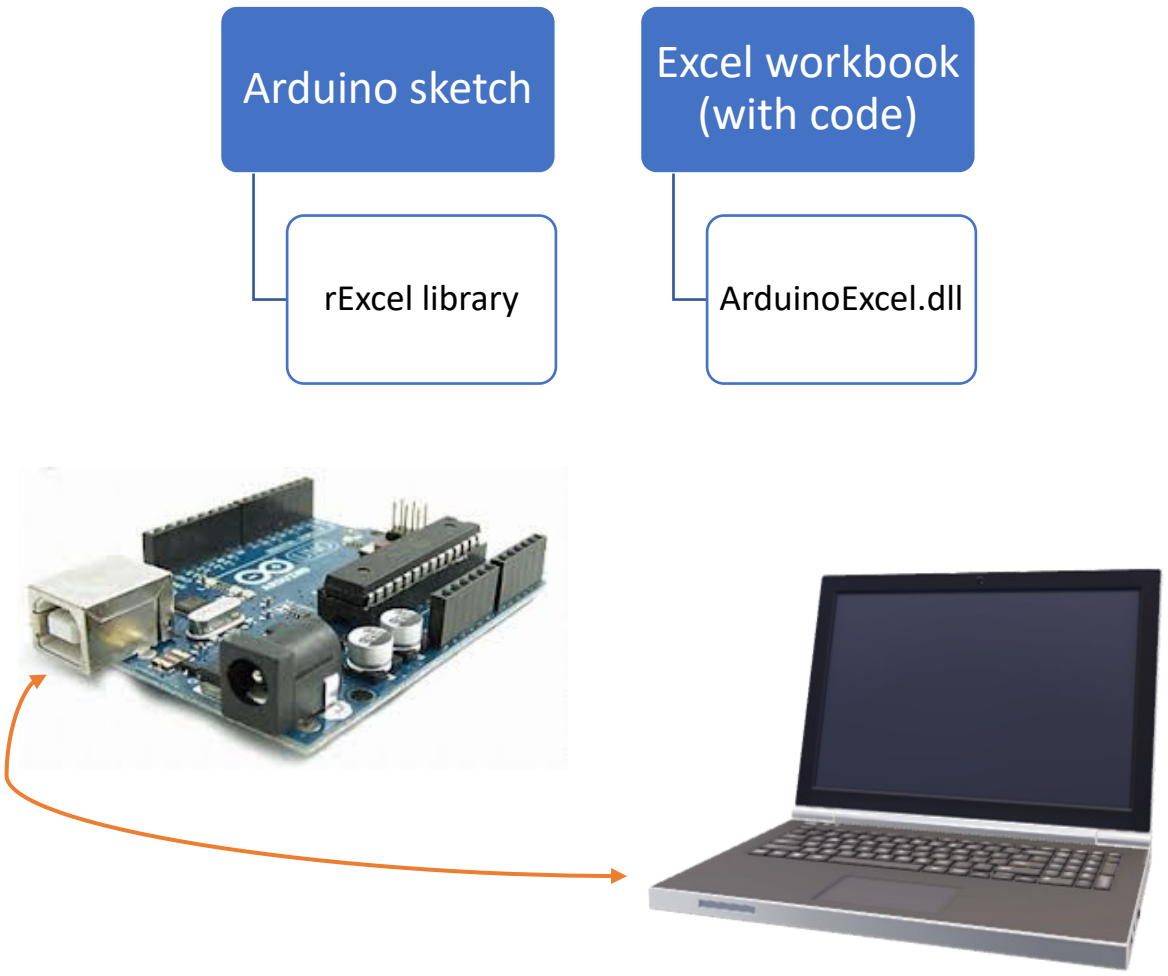


See the sketch for examples an modify them for your purposes. You can use a modified copy of *Arduino_Excel_30.xls* but note that it isn't a generic Excel file because contains the code (macro) to interact with Arduino.

[(1)] if CTRL 'a' key pressing doesn't work open menu Show, click on Macro icon, Show macro, select CommanderRunA and press Run button, press Options button to define a launch key as you like

Software Architecture

The standard interaction between systems is based on Excel acting as server and Arduino as client. Note that for Excel we mean a workbook with special code (macro) running.



The rExcel library defines the API for data interchange, ArduinoExcel.dll instead is to manage COMs and message queues since Excel VBA hasn't specific instructions for them.

Arduino Excel API

In order to make easy and clean the Arduino programming an API was implemented in rExcel library. See the following reference and the sketch supplied as example for practical uses.

```
// class for Excel data exchange
#include <rExcel.h>
rExcel myExcel;
```

up	test if Excel is connected
syntax	bool up();
parameters	none
return	true if Excel connected
example	if (myExcel.up()) { ...

write	write a value in a cell (Excel range form)

syntax	bool write(char* worksheet, char* range, int value);
	bool write(char* worksheet, char* range, long value);
	bool write(char* worksheet, char* range, float value, int decimals);
	bool write(char* worksheet, char* range, char* value);

parameters	worksheet destination worksheet range cell reference (range form) value value to write decimals number of decimals
return	true if successful
example	float x = 12.34; myExcel.write("Example", "B5",x, 2); // write the value from x variable to worksheet 'Example' cell 'B5' with two digits as decimals

writeIndexed	write a value in a cell (Excel cells form)

syntax	bool writeIndexed(char* worksheet, unsigned int row, unsigned int column, int value);
	bool writeIndexed(char* worksheet, unsigned int row, unsigned int column, long value);
	bool writeIndexed(char* worksheet, unsigned int row, unsigned int column, float value, int decimals);
	bool writeIndexed(char* worksheet, unsigned int row, unsigned int column, char* value);

parameters	worksheet destination worksheet row row number column column number value value to write, special keywords %date% and %time% decimals number of decimals
return	true if successful

example	int idx;
	float a0 = 12.34;
	myExcel.writeIndexed("Example", 11, 1,"%date%"); // write %date% (that will be converted in current date) to worksheet 'Example' row '11' column '1'
	myExcel.writeIndexed("Example", 11, 2,"%time%"); // write %time% (that will be converted in current time) to worksheet 'Example' row '11' column '2'
	myExcel.writeIndexed("Example", 11, 3, idx); // write the content of idx to worksheet 'Example' row '11' column '3'

myExcel.writeIndexed("Example", 11, 4, a0, 2); // write the content of a0 to worksheet 'Example' row '11' column '4' with two digits as decimals

send

send a free string

syntax	bool send(char* data);
parameters	data a string \0 terminated
return	true if successful
example	myExcel.send("hello\0");
note	the specific function Excel side must be customized

get

read a value from a cell (Excel range form)

syntax	bool get(char* worksheet, char* range, char* buffer);
parameters	worksheet source worksheet range cell reference (range form) buffer buffer for received info
return	true if successful

example	char value[32];
	myExcel.get("Test", "B5", value); // get a value from worksheet 'Test' cell 'B5'
	int x = atoi(value); // convert value and put it in a int variable

getIndexed

read a value from a cell (Excel cells form)

syntax	bool getIndexed(char* worksheet, unsigned int row, unsigned int column, char* buffer);
parameters	worksheet source worksheet row row number column column number buffer buffer for received info
return	true if successful
example	char value[32]; myExcel.getIndexed("Test", 5, 2, value); // get a value from worksheet 'Test' row '5' column '2'

clear

clear cell(s) (Excel range form)

syntax	bool clear(char* worksheet, char* range);
parameters	worksheet source worksheet range cell reference (range form)
return	true if successful
example	myExcel.clear("Example","A11:E70"); // clear cells in A11:F70 area of 'Example' worksheet

save

save the workbook

syntax	bool save();
parameters	none
return	true if successful
example	myExcel.save();// save the workbook

mail

send a mail

syntax	bool mail(char* recipient, char* recipientCc, char* subject, char* body, char* attach);
parameters	recipient recipientCc subject body attach only keyword %log% (no path or filename)
return	true if successful
example	myExcel.mail("x.y@gmail.com","","Test","Hello!","%log%"); // send a mail with the configured logfile as attach

log

write info in the configured log file (see Parameters worksheet)

syntax

```
bool    log(char* info);
```

parameters

info string with arguments semicolon (;) separated

return

true if successful

example

```
// write on log the values of idx and rnd
char floatValue[8];
char info[16];
dtostrf(rnd, 3, 2, floatValue); // trick because in Arduino sprintf() doesn't represent floats
sprintf(info,"%d;%s", idx, floatValue); // info must separated by semicolon, be careful on info lenght
myExcel.log(info);
```

note

date and time are automatically added for each writing

read

waits for data from Excel (Arduino server mode)

syntax

```
int     read(char* worksheet, char* range, unsigned int* row, unsigned int* column, char* buffer, char mode);
```

parameters

worksheet source worksheet
range cell reference (range form)
row
column
buffer
mode

return

number of read chars

example

```
// input task
// Arduino acts as server waiting data from Excel
if (Serial.available() > 0) {
  if ((ret = myExcel.read(worksheet, range, &row, &column, value, 'R')) > 0) {
    Serial.print(XLS_ACK);
    Serial.print("\n");
    // write here your code
    range[0] = 'H';
    myExcel.write(worksheet, range, value); // only as proof that works
  }
  else {
    // input clearing
    Serial.print(XLS_NAK);
    Serial.print("\n");
    myExcel.clearInput();
  }
```

note

this function is for advanced users

clearInput

clear chars on serial port

syntax

```
void    clearInput();
```

parameters

none

return

none

example

```
myExcel.clearInput();
```

Protocol

When an API function is called, the rExcel library builds a specific message and send it to Excel that process the message and respond with acknowledgement or not. If not acknowledgement the function does some retries before to give up.

The data exchange is based on readable CSV strings terminated with NewLine char.

The type of messages and their formats are:

- XLS,command,[arg1],[arg2],[arg3],[arg4],[arg5]
- LOG,infoSemicolonSeparated
- EML, recipient,recipientCc,subject,body,attach

For details about the message management see rExcel library and the code in Arduino_Excel_30.xls (open the file and press ALT F11 to get the programming environment).

At present no checksum or other protocol controls are performed anyway the code in Excel verifies the formal structure of the messages and refuse if something is wrong.

Programming tips

Arduino sketch and rExcel library

The supplied sketch Arduino_Excel_30.ino gives examples and comments for all API function. For more info refer to Arduino API chapter.
It works with all Arduinos.
Important: never insert delay() in your sketch because it can alter the communication between Arduino and Excel.

About rExcel library some things are highlighted:

#define ATMEL_COMPATIBLE	comment this define if your Arduino isn't Atmel compatible like Arduino Due or Intel Edison
XLS_BUFFER_SIZE = 32	buffer size for input, may be small if your app reads strings from Excel
XLS_TIMEOUT = 500	reading timeout ms
XLS_MAX_TRIES = 1	protocol tries, set 1 for up() function

The above values are tested for most applications, change them only for special requirements.

Excel VBA code (macro)

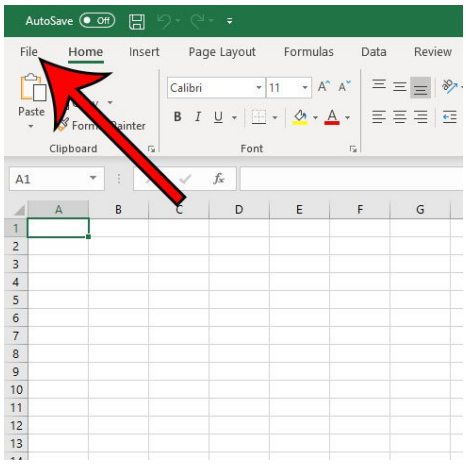
Said that the most programming is at Arduino level, here some tips about the Excel side.

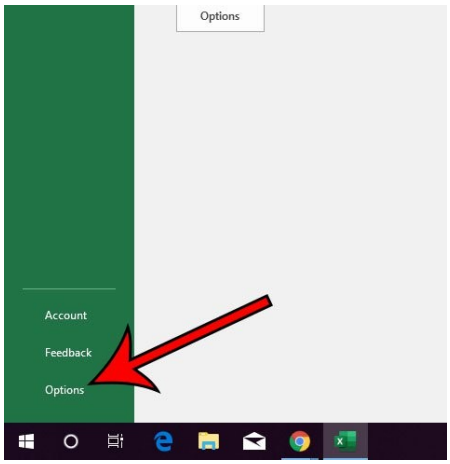
VBA hasn't two important features for this app: COMs management and timed events.
In order to overcome the first lack, a specific DLL was developed for COMs management within a separate process plus a message queue to store incoming messages. This software is protocol independent except for message ending that must be NewLine.
Without timed events a continuous loop looks for queued messages, check and performs all requests. This is the heart of the app, for more details get access to the programming environment and see the ReadingLoop() function in Module1.
The loop ends when all connections are closed.

All software has a lot of comments so they should be self-explaining for customizations.

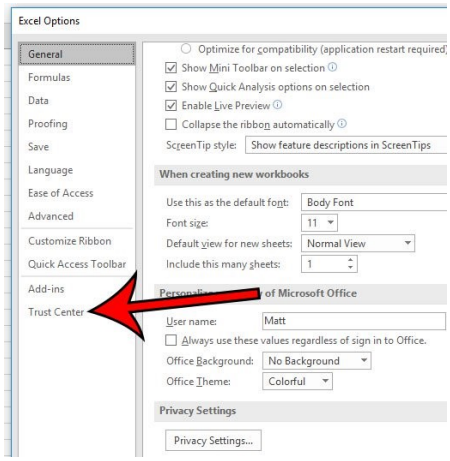
Appendix A – macro enabling

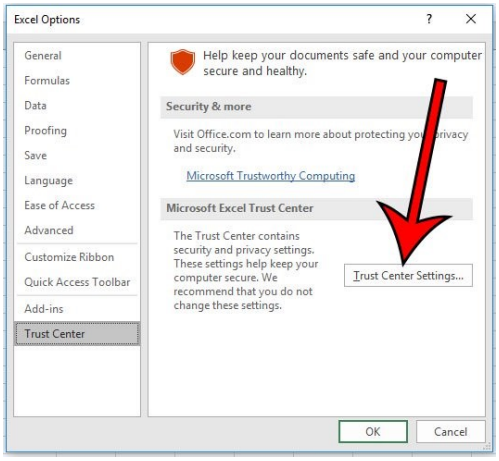
Click on File, Options



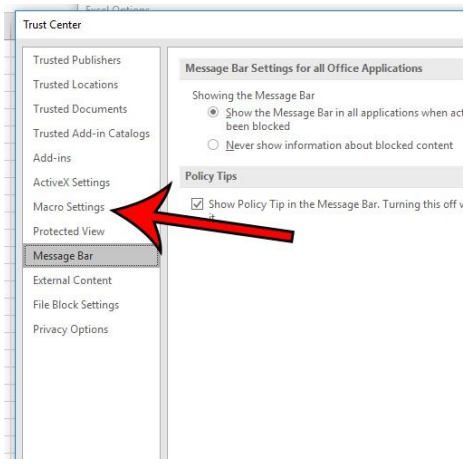


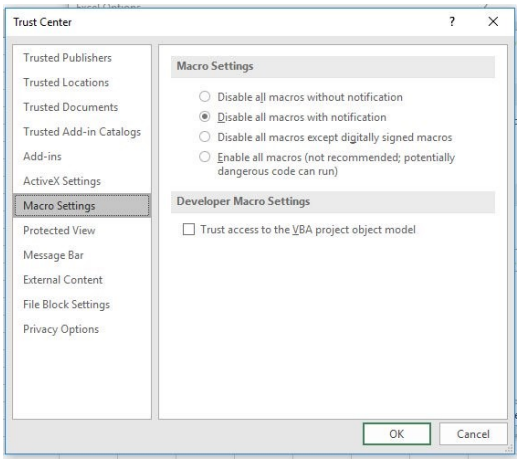
click on Trust Center, Trust Center Settings





click on Macro Settings



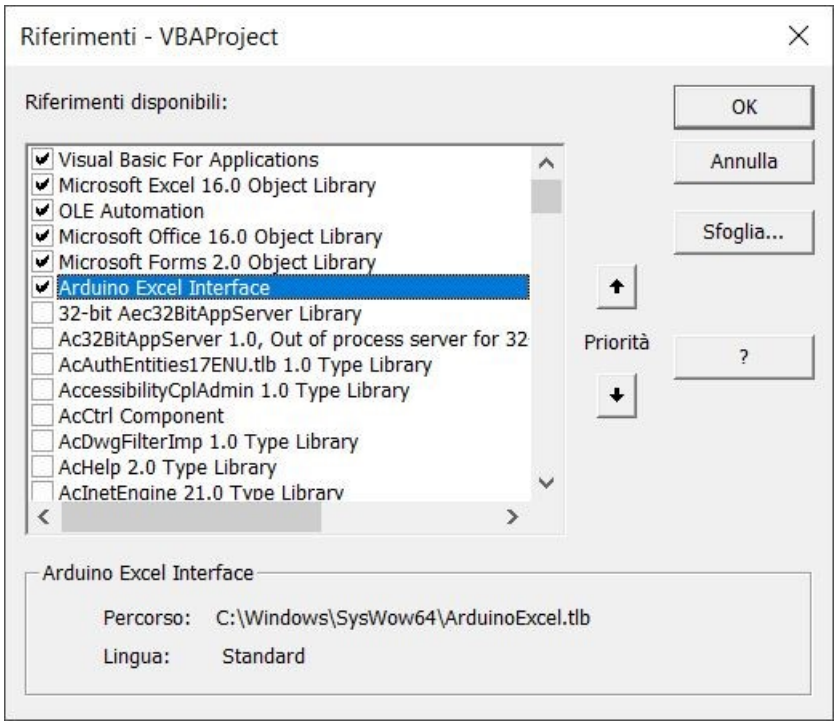


click 'Enable all Macros' and check 'Trust access to the VBA project object model'

Appendix B

May happen that your Excel VBA (Visual Basic Application) hasn't configured some component called "reference" so check as follows:

- press ALT F11
- click on Run menu and then Restore (this reset the error status)
- click on Tool menu and then on Reference, should appear this form



example from Office 365

- compare the checked items with yours
- If missing some reference find it across the list (if you have an old Office may be you find "Microsoft Office 11.0 Object Library" instead of "Microsoft Office 16.0 Object Library")
- check the references as needed
- press OK button
- press now CTRL 'a' to check if all is ok, an application form should appear
- go back to Getting Started chapter

Appendix C

ArduinoExcel works on old Windows versions but not it's regular setup so a manual procedure is required:

- download *ArduinoExcel_30.zip* from https://github.com/rvalgolio/ArduinoExcel_30/tree/main/Setup
- extract the files from the zip creating a folder on your PC (not simply open the zip)
- launch the batch *Setup.cmd* as administrator (right click on the file)
- come back to the Installing chapter (p.3) and continue from the "Second step"