bot'n roll ONE A⁺





Arduino[®] software user manual



©Copyright, SAR - Soluções de Automação e Robótica, Lda.



Software Manual for the Bot'n Roll ONE A+

CONTENT

1. Int	roduction2
1	L. Programming the Bot'n Roll ONE A+ 2
2	2. BnrOneAPlus Library for Arduino
	3. Arduino IDE4
Z	ł. C Language5
Anex	o A: VCP Driver Installation for USB converter (RS232)6
Anex	o B: Programming IDE 2.06
Anex E	o B: Programming IDE 2.0
Anex E	o B: Programming IDE 2.0
Anex E E	o B: Programming IDE 2.0
Anex E E	o B: Programming IDE 2.0

Revisão do Documento: 2 de Março de 2025



1. INTRODUCTION

The Bot'n Roll ONE A+ is programmed using the C language with the Arduino IDE programming environment. The ATmega328 microcontroller in the robot has the Arduino Uno bootloader, so the robot is programmed as if it were an Arduino Uno.

The robot has a second microcontroller, a PIC18F45K22, which comes pre-programmed with software developed by botnroll.com. In the Bot'n Roll ONE A+, it functions as a slave device that executes the command orders from the ATmega328 master.

The two microcontrollers in the Bot'n Roll ONE A+ communicate with each other via the SPI (Serial Peripheral Interface) bus. They exchange information in a coordinated and well-defined manner. For this purpose, a data transfer protocol was developed between the master and the slave. The master uses a list of commands that correspond to control instructions, and each command generates a response from the slave. The list of commands and the way data is transmitted between the master and slave are defined in the "BnrOneAPlus" library.

The "BnrOneAPlus" Arduino library allows the user to control the robot easily—just by correctly using the library's commands in the Arduino IDE. These commands are listed and explained in this manual.

Although both microcontrollers can be programmed in C by the user, only the ATmega328 with the Arduino bootloader is programmed on a daily basis using the "BnrOneAPlus" library.

The PIC18F45K22 can be programmed in C using the MPLABX IDE programming environment and the Microchip XC8 compiler or other compatible software. However, this should only be done by advanced users, as programming the PIC18F45K22 to include a new feature also requires updating the "BnrOneAPlus" library so that the Arduino can use the new functionality.

Contact bothroll.com if you'd like to see a new feature implemented in your Bot'n Roll ONE A+!

1. PROGRAMMING THE BOT'N ROLL ONE A+

To program the Bot'n Roll ONE A+, you need to have your computer set up with all the necessary tools, namely:

- VCP driver installed, the USB port driver for the Bot'n Roll ONE A+ (Annex A);
- Arduino IDE installed (Annex B1);
- BnrOneAPlus library installed in the Arduino IDE (Annex B2).

For detailed information on installing the items mentioned above, refer to Annexes A and B at the end of this manual.

C++ is also a necessary tool for programming the Bot'n Roll ONE A+. If you're not yet very comfortable with C++, the library examples are a great guide to help you get started in the world of programming. You can also check out the RoboParty presentations on C programming, and of course, there are thousands of web pages explaining C++ online.



2. BNRONEAPLUS LIBRARY FOR ARDUINO

A library is a set of "pre-made" code that you can insert and use in your program. To use the "BnrOneAPlus" library, simply include it in your code:

#include<BnrOneAPlus.h>

and create an instance of the BnrOneAPlus class:

BnrOneAPlus one;

From this point on, you have access to all the library functions, preceded by the instance you defined, meaning: **one**.*function_of_library()*;

A library is usually created for data or hardware manipulation and always consists of at least two files. However, in the case of Arduino, there is an additional file with the "**.txt**" extension:

- A file with the ".h" extension (header), which contains a list of all the functions, commands, and definitions of the library.
- A file with the ".cpp" extension (C++ source), which contains the implementation of all the functions declared in the header file.
- A **keywords.txt** file, which allows the Arduino IDE to recognise the library functions and display them in a different colour from the rest of the code.

The **"BnrOneAPlus"** library was created to control the hardware associated with the **PIC18F45K22** and allows the Arduino to interact with it via the SPI communication bus. The Arduino has access to all the hardware and functionalities defined in the library and in the **PIC18F45K22** software. The **"BnrOneAPlus"** library and the **PIC18F45K22** software were designed to work together, meaning that any change to one requires an adjustment to the other.

The library is available online at:

- https://github.com/botnroll/BnrOneAPlus

Its corresponding documentation and description of the available functions can be found at:

- https://botnroll.github.io/BnrOneAPlus/class_bnr_one_a_plus.html.



3. ARDUINO IDE

The Arduino development environment includes a text editor for writing code, a message area, a text console, a toolbar with the most important functions, and several menus. It connects to the Arduino hardware of the Bot'n Roll ONE A+ to upload code and communicate with the robot.

An Arduino program is called a "sketch", written in the text editor and saved on your computer with the ".ino" extension.

The message area displays information about saving and exporting programs, as well as any errors.

The console provides detailed text messages about errors and other information.

In the bottom right corner of the window, you can see information about the board being programmed and the serial port in use.

The buttons on the toolbar and their function:



Verify: Check for errors in the code

Upload: Compile the code and upload it to the Arduino.



New: Create a new sketch.



Open: Open a sketch saved on the computer.



Save: Save the sketch.

Serial Monitor: Open the serial monitor.

The serial monitor allows you to view data sent from the Arduino to the computer and also send data from the computer to the Arduino. It is very useful in programming, as you can print text and variable values here, making it easier to debug (find errors) in your program. When you open the serial monitor, your program on the Arduino restarts.

ar data Na. Mag	<pre>twitter like in the first</pre>	<pre>inter two two interventions and the second sec</pre>
A Addine Unit and Adding and A	Control Control Inform Information Information Information	<pre></pre>
LUDer i fic.lok demonselver. // Serie will det a library i fic.lok demonselver. // Serie will det a library i fic.lok demonselver. // Serie will det a library fic.lok demonselver. // Serie will be library fic.lok demonselvere	<pre>Unit for the development of the set is ball of a library for the development of the set is ball of a library for the development of the set is ball of a library for the development of the set is ball of a library for the set i</pre>	<pre>Construction of conversion of the second section of the second section of the second sec</pre>
<pre>if choluse composed/instance // if our mail loss As liberry if effolds composed/instance // if our communication instance // register what is control the set a nell out A if contrast setfailand if effolds contras</pre>	<pre>first.inde composition_shot // soft mail loss 44 liberry if sicklede composition_shot // soft communication in duriting if sicklede composition // soft communication liberry regained by involved luss reg if contains efficients if contains efficients if software composition is and the contains of the contains of the contains of the contains of the contains if software composition is an efficient of the communication if software composition is an efficient of the communication if software composition is an efficient of the communication if software composition is an efficient of the contains of the communication if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains if software composition is an efficient of the contains of the contains of the contains if software composition is an efficient of the contains of the contains of the contains if software composition is an efficient of the contains of the contains</pre>	<pre>include concostInable.b // strin soll one As liberry include cortains.b // strin soll one As include cortains // strin soll one As include cortains if include cortains // strinorization liberry regiment withing if include cortains // strinorization liberry regiment withing if include cortains // strinorization liberry regiment withing if include cortains if i</pre>
<pre>inicial cerefords // Execut reading and writing inicials cerefords // Execution of adject writable to control the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the control of the bot's woll one a inicial set of the bot's woll one a inicial set</pre>	<pre>include cerements. // creation reading and writing #include cerements. // creation of deject variable to control the Ket's Kell one A #include cerements. // scientific deject variable to control the Ket's Kell one A #include cerements. // creation of deject variable to control the Ket's Kell one A #include cerements. // creation of deject variable to control the Ket's Kell one A #include cerements. // creation of deject variable to control the Ket's Kell one A #include cerements. // creation of deject variable to control the Ket's Kell one A #include cerements. // creation of deject variable to control the Ket's Kell one A #include cerements. // creation of the Ket's Kell one A</pre>	<pre>include cettoreLLD // instruct reading and writing include cettorLD // instruct reading and instruct in Chatter include cettorLD // instruct reading and instruct in include cettorLD // instruct reading and instruct in instruction (instruction) // instruction (instruction) // instruction instruction (instruction) // instruction (instruction) // instruction instruction (instruction) // instruction (instruction) // instruction (instruction) // instruction instruction (instruction) // instruction (instruction) // instructing (instruction) // instruction (</pre>
<pre>is ficilule (GPLab // UF) communication liberary readined by thereables.cpg ficility (contrasts officiation ficility (con</pre>	<pre>info:due coff.bs // SFI communication liberary reached by increase/lat.reg(for discussion of // SFI communication liberary reached by increase/lat.reg(for discussion of the liberary reached by increase libera</pre>	<pre>information of the second second</pre>
<pre>in servowe/lus one;</pre>	<pre>service/bis one; // declaration of object variable to control the Bot'n Holl ONE A // (constants definition // setains definition /</pre>	<pre>if groups distance of adject satisfies to control the for's full One A</pre>
<pre>// (contraits definition // (contraits de</pre>	<pre>// (constants definition // (constants de</pre>	<pre>int loss of loss</pre>
<pre>interview of the second s</pre>	<pre>// (notacits definition) // (notacits def</pre>	<pre>iii // (constant set of the form of the set of the form of the construction iii // (constant set of the form of the set of</pre>
<pre>intering Spate 1 // State State();) Juit tor Sri Consultation if (); if (</pre>	<pre>intering Suff 1 // Laws Safe(c) juit to 51 commutation if it is an intering it</pre>	<pre>inf in series series if i / issue sate(:(s) juint of Sri Commutation if inf issue) if (find), segin((f)600); // start the commutation module</pre>
<pre> for sink struct() for sink segs(CSYMMS); // set hand rate to CSYMMMS for printing values at serial monitor.</pre>	<pre>init start() for init() ((i) (i) (i) (i) (i) (i) (i) (i) (i)</pre>	<pre>ind store() fortial.store() forti</pre>
<pre>provide the set of the set o</pre>	<pre>provide the set of the set o</pre>	<pre>provide the set of the set o</pre>
<pre>2 Serial.agg(cf050); // set hud rate to ToABDe for printing values at verial monitor. 3</pre>	<pre>containing: [foreling: [foreling: // set that draft to Childing for prioring values at serial mention;</pre>	<pre>intermediate in the intermediate of the intermediate values at serial monitor</pre>
<pre>// dec.up(comerci(comerci) // start the communication module // dec.up(comerci(comercic) // start start on the communication module // transition // tr</pre>	<pre>// generation module // state interest // state // s</pre>	immediation immediation immediation // sturt the communication medule immediation // sturt the communication medule immediation immediation
/* (dew.tog(); // stop matters /* (dew.tog(); // turn tip on /* (dew.tog(); // turn tip on /* (dew.tog(); // turn tip on /* dew.tog(); // turn tog(); // turn tip on /* dew.tog(); // turn tog(); // turn tip on /* dew.tog(); // turn tog(); // turn tog(f de	important important important
<pre>pic truth 1000() pic truth 100 () p</pre>	prist loor() // turn LLD OT prist loor() // valit issue Own @ Own @ bibdai variables use Job types (IN) of program storage space. Nuslam is J220 bytes. Deal variables notion bibdai variables use Job types (IN) of gramma storage space. Nuslam is J220 bytes for local variables. Notion bibdai variables use Job types (IN) of gramma storage space. Nuslam is J220 bytes for local variables. Notion bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (IN) of gramma storage space. bibdai variables use Job types (interference // turns LED ON interference // turns LED ON interference interference
/* mid 100() ?> <{ 	<pre>interface () for ()</pre>	inf lance() // torn: LEO 01 inf lance() // torn: LEO 01 inf lance() inf lance() inf lance() // torn: LEO 01 inf lance() inf lance() inf lanc
20 (Open Serial(UCOD); // turn LD OF Serial(sprint(CUD OF); // still second Serial monthme Own Serial(sprint(CUD OF); // still second Serial sprint(CUD OF); // still second Serial monthme Serial sprint(Serial second Serial monthme Serial monthme Serial variables use 260 bytes (HK) of Symmic memory, lassing 101 bytes for local variables. Monthme Serial monthme Serial variables use 260 bytes (HK) of Symmic memory, lassing 101 bytes for local variables. Monthme Serial monthme Serial serial monthme Serial serial monthme Serial monthme Serial serial serial serial serial monthme Serial serial monthme Serial serial monthme Serial seria	Image: Section (Section (S
<pre>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	<pre>36 ee.26(0000); // (un LD On 31 ee.26(0000); // unit LD On 32 ee.26(0000); // unit Lacone 32 ee.36(0000); // unit Lacone 32 ee.36(00000); // unit Lacone 32 ee.36(0000); // unit Laco</pre>	<pre>genc (effective); // turn LD 0% genc (effective); // turn LD 0% genc (effective); // exit is scored Out Setorh uses 2546 bytes (x3) of pregress storage space. Nucleum 15 32250 bytes. (ibbal variables use 387 bytes (idb) of dynamic energy, lawing 1s41 bytes for local variables. Nucleu (ibbal variables use 387 bytes (idb) of dynamic energy, lawing 1s41 bytes for local variables. Nucleu (ibbal variables use 387 bytes (idb) of dynamic energy, lawing 1s41 bytes for local variables. Nucleu (ibbal variables use 387 bytes (idb) of dynamic energy, lawing 1s41 bytes for local variables. Nucleu (ibbal variables use 387 bytes (idb) of dynamic energy, lawing 1s41 bytes for local variables. Nucleu (ibbal variables use average) (ibbal variables use ibbal variables use ibbal variables energy) (ibbal variables use ibbal variables use ibbal variables energy) (ibbal variables use ibbal variables use ibbal variables energy) (ibbal variables use ibbal variables use ibbal variables energy) (ibbal variables use ibbal variables use ibbal variables energy) (ibbal variables use ibbal variables use ibbal variables energy) (ibbal variables e</pre>
31 Serial, periodic (100 mC); // print i secon serial monitor. 20 Serial (2000); Onjur Image: Comparison of the second serial monitor. Sector uses 24th bytes (74) of program storage space. Nuclear is 32200 bytes. Image: Comparison of the second serial monitor. Sector uses 24th bytes (74) of program storage space. Nuclear is 32200 bytes. Image: Comparison of the second serial monitor. Sector uses 24th bytes (140) of opmatic memory. leaving 14th bytes for local variables. Nuclear is 11(Coll Andrews Uncern COMMS: C2 Image: Comparison of Comparison	31 Serial, periodic (100 mC); // print i secon serial monitor. 20 Serial (2000); Onjur ■ C Sector uses 24th bytes (74) of program storage space. Nuclean is 12230 bytes. ■ C Global variables use 247 bytes (180) of dynamic memory, leaving 1841 bytes for local variables. Maximum is 1000 bytes for local variables. Maximum is 11(Cold Anderso Uneum COMME C2 ■ Cold Dame upbedding.	11 (Set1).prfchr/CHO GY); // print fers on set1al montror. 22 (Set0): // print rescond Obget Setch uses 2016 bytes (73) of program storage space. Nuclima is 32220 bytes. 6/ball variables use 2017 bytes (185) of dynamic memory, leaving 184) bytes for local variables. Navian init(Gel Antamoleu on COMA) C2 C O Dame upfoading
		22 mellor(low); // wit: 1 scored Obpi Image: Comparison of the state of the s
Ouput IIII Store (AT) of program storage space. Hustman is 32256 bytes. Lidebal variables use 207 bytes (UHS) of dynamic memory, leaving 1841 bytes for local variables. Hustman Lidebal variables use 207 bytes (UHS) of dynamic memory, leaving 1841 bytes for local variables. Hustman Linit(Gal AntenerUmererGOMMS 072 IIII) Dame upboading	Ouput Congram Storage space. Hustman is 32206 bytes. Lidebal variables use 207 bytes (UHS) of Oynamic memory, leaving 1841 bytes for local variables. Hustman En11Coli Antonoclone of Coll Variables. Packing En11Coli Antonoclone on COLAN. C2 Coll O Done spikading.	Dague Setch uses 2446 bytes (243) of program storage space. Nuclima 15 32230 bytes. Biobal variables use 287 bytes (145) of dynamic memory, leaving 1441 bytes for local variables. Nuclea in11,Gol3 AntomoleurorCOMAB 07 C Dame upfooding
skerch uns 554 bytes (76) of program storage space. Austam (s 22% bytes: Clobal variables use 207 bytes (106) of dynamic memory, leading 1841 bytes for local variables. Huchmu in 1(Gel 3 Animulture COMAR (22)	skerch uns 554 bytes (76) of program storage space. Austam (s 22% bytes: Clobal variables use 207 bytes (106) of dynamic memory, leading 1841 bytes for local variables. Huchmu in 11,Ge3 AndoneuMou en COMAR (\$2 🗆 () Done upbeading.	sterch uns 2448 bytes (76) of program storage space. Nuclear 15 9226 bytes. Global variables use 207 bytes (166) of dynatic energy, leaving 1841 bytes for local variables. Nuclear In TLGGR Anternationan COMPS (72 C
in 11, Car3 Adama Una an COMAS 🖉 2 🗖	La 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una an OSMAS - OL 2 - OL 11, Gal 3 - Antaria Una antaria	te 11, Gol 3. Andraw line or COMAS C2. C O Done upleading.
O Dana upleading.	🕐 Dana upkading.	🕐 Done upleading.
🕐 Done upleading.	🚫 Dane spleading	💮 Done upleading.
		utput 🚍 d
Output III Control	Output 🔤 🛱	Sketch uses 2548 bytes (7%) of program storage space. Maximum is 32256 bytes.
Output = G Sketch uses 2548 bytes (78) of program storage space. Maximum is 32250 bytes.	Output 🚍 🛆 Sketch uses 2548 bytes (7%) of program storage space. Maximum is 32256 bytes.	Global variables use 207 bytes (10%) of dynamic memory, leaving 1841 bytes for local variables. Maximu
Oulput Sketch uses 2548 bytes (7A) of program storage space. Raximum is 32256 bytes. Global variables use 207 bytes (1087) of dynamic mewry, lewing 1941 bytes for Jocal variables. Maximum	Output == G Sketch uses 2548 bytes (7%) of program storage space. Maximum is 32256 bytes. Global variables use 207 bytes (10%) of dynamic memory, leaving 1841 bytes for local variables. Maximum	
\overline{Output} Statch uses 2648 bytes (7A) of program storage space. Muximum is 12256 bytes. Global variables use 207 bytes (100) of dynamic memory, leaving 1841 bytes for Jocal variables. Muximum	Oxput III Sketch uses 2548 bytes (74) of program storage space. Maximum is 12256 bytes. Global variables use 207 bytes (100) of dynamic memory, leaving 1841 bytes for local variables. Maximum	
Ouput Sketch uses 2548 bytes (7A) of program storage space. Maximum is 32256 bytes. Global variables use 207 bytes (105) of dynamic memory, lawing 1841 bytes for local variables. Maximum	Odapd Sketch uses 2548 bytes (78) of program storage space. Maximum is 32256 bytes. Global variables use 207 bytes (108) of dynamic memory, leaving 1841 bytes for local variables. Maximum	
Debut Setch uses 2548 bytes (70) of program storage space. Maximum is 32256 bytes. Global variables use 207 bytes (100) of dynamic memory, leaving 1841 bytes for local variables. Mucimum	Odqui ≕ A Sketch uses 3548 bytes (78) of program storage space. Roxime is 32256 bytes. Global variables use 207 bytes (188) of dynamic memory, leaving 1841 bytes for local variables. Nucleum	
Ougue Statch uses 2548 bytes (7A) of program storage space. Maximum is 12256 bytes. Global variables use 207 bytes (105) of dynamic memory, leaving 1841 bytes for local variables. Maximum	Odopd Saketch uses 2548 bytes (74) of program storage space. Raximum is 32206 bytes. Glabal variables use 207 bytes (108) of dynamic memory, leaving 1841 bytes for local variables. Maximum	
Deput Sketch uses 2548 bytes (70) of program storage space. Muximum is 33256 bytes. Glabal variables use 207 bytes (100) of dynamic memory, leaving 1841 bytes for local variables. Muximum	্যল্ল Satch uses 2548 bytes (7%) of program storage space. Maximum is 32256 bytes. Global variables use 207 bytes (10%) of dynamic memory, leaving 1841 bytes for local variables. Maximum	
Ougut Sadetch uses 2648 bytes (7Å) of program storage space. Maximum is 32256 bytes. Global variables use 207 bytes (10%) of dynamic memory, leaving 1841 bytes for local variables. Maximum	Oupud Setch uses 2648 bytes (7Å) of program storage space. Rusinum is 32256 bytes. αlobal variables use 207 bytes (10%) of dynamic memory, leaving 3841 bytes for local variables. Muximum	
Odgad Setch uns: 2548 bytes (70) of program storage space. Maximum is 32266 bytes. Global variables use 207 bytes (100) of dynamic memory, leaving 1841 bytes for local variables. Mucimum	$\equiv \hat{\Theta}$ Setch uses 2548 bytes (70) of groupum storage space. Raximm is 32256 bytes. Global variables use 207 bytes (105) of dynamic memory, leaving 1841 bytes for local variables. Maximum	





4. C LANGUAGE

The C language was developed in 1972 by Dennis Ritchie at Bell Labs in New Jersey. It was created to be a powerful and fast language for use in the Unix operating system that was being developed at the time. Over the years, it has been improved and updated, proving to be robust and reliable, and eventually being adopted by other operating systems such as Windows, macOS, and Linux.

C has been evolving continuously since the first version, known as "K&R C". In 1989, the first official standard specification was published by the American National Standards Institute (ANSI C). In 1990, the International Organization for Standardization (ISO) introduced ISO C. In 1999, the C99 standard was released, followed by periodic revisions aimed at making the language more powerful and better suited to modern needs.

All Arduino programs have two mandatory routines, or functions:

- The **setup()** routine, which is executed **only once** at the start of the program. Here, you should place all the necessary code to initialize variables, configure **input and output pins**, set up **SPI**, **Serial**, **and I2C communication**, and perform any other required configurations.

- After the setup, your program enters the **loop()** routine, where it remains indefinitely. The term **loop** means cycle, and in this case, it is an **infinite loop**, meaning that when the program reaches the end of the cycle, it starts over again! This is where you write your program and create the **intelligence** for your robot!

This manual does not explain **C++ programming** itself. Instead, we recommend using the **"BnrOneAPlus" library examples** and general **Arduino examples**. All the code is well-commented, and you'll need to experiment and test it to understand how it works.

Here are some tips:

- Create new programs based on basic examples. Try combining 3 or 4 functionalities of the robot in the same program!
- Programs rarely work perfectly the first time! Don't get discouraged—find the issue and fix it!
- Add code gradually and test frequently to check if everything works as expected.
- Testing takes longer than writing the program! A good program requires thorough testing!
- Use debugging tools like the LED, serial monitor, or LCD to print variable values and check if the program reaches certain parts of the code.

Programming is like learning a new sport!

At first, it feels **difficult and frustrating** because you don't have the necessary skills, don't know the rules, and feel a bit lost. But with practice and training, you **learn**, **improve**, **and get better** in every way. With effort, you'll eventually become a pro!

The **Bot'n Roll ONE A+** supports **a wide range of hardware**. There are additional components, commonly known as **Arduino shields**, that allow you to do **almost anything** you can imagine—and they are compatible with the Bot'n Roll ONE A+! **All shields come with libraries** to help you integrate them, and **your imagination is the only limit!**



ANEXO A: VCP DRIVER INSTALLATION FOR USB CONVERTER (RS232)

The driver allows your computer's operating system to communicate with the Bot'n Roll ONE A+.

To install the driver, visit the Bot'n Roll ONE A+ support page (<u>http://botnroll.com/one-a-plus/</u>) and download it by clicking on "VCP Driver - Windows" or "VCP Driver - Mac OS X", according to your operating system. Once the download is complete, extract the file with the ".zip" extension and run the application.

Whenever you connect the robot to the computer using the USB cable, a virtual COM port (VCP) is created, through which communication between the Bot'n Roll ONE A+ and the PC takes place. The application for programming the robot uses this port to communicate with the Bot'n Roll ONE A+ and transfer programs to the robot.

The USB-Serial converter used in the Bot'n Roll ONE A+ is a PoUSB12 from PoLabs, which uses the CP2102 Bridge device from Silicon Labs.

ANEXO B: PROGRAMMING IDE 2.0

The software used for programming the robot is Arduino IDE 2.0. This application is necessary for editing programs in C++ language. It is also used to transfer your programs to the Bot'n Roll ONE A+.

B.1 ARDUINO IDE INSTALATION

To install the Arduino IDE, visit the Bot'n Roll ONE A+ support page at (<u>http://botnroll.com/one-a-plus/</u>). In the "Software | Drivers" section, click on "Arduino IDE" to install it according to your operating system.

Once the download is complete, extract the "**.zip**" file and place the extracted folder in a directory of your choice on your computer.

This folder contains several subfolders and files, including the **"arduino.exe"** application, the executable that launches the Arduino IDE.

B.2 BNRONEAPLUS LIBRARY INSTALLATION ON ARDUINO IDE 2

Libraries are your working tools in programming. The "BnrOneAPlus" library, developed by botnroll.com for the Arduino IDE, contains all the necessary commands for controlling the robot. This library must be installed in the Arduino IDE.

To install the library in ArduinoTM IDE 2, click on the "Sketch" tab \rightarrow "Include Library" \rightarrow "Manage Libraries...", search for "BnrOneAPlus" and install the library by clicking the "Install" button.

To install the library in ArduinoTM IDE 1, click on the "Sketch" tab \rightarrow "Include Library" \rightarrow "Add .ZIP Library...", select the BnrOneAPlus.zip file, and the library will be installed automatically. Close and reopen the ArduinoTM IDE to ensure the library is functional!

The procedures for installing the VCP driver and the "BnrOneAPlus" library are also described in the Assembly Manual.



B.3 Configuring the communication with the robot

Before performing this step, make sure you have installed the VCP driver correctly (see ANNEX A). Connect the Bot'n Roll ONE A+ to the computer using the supplied USB cable. At this point, a COM port will be automatically assigned for communication with the robot.

Open the Arduino IDE, and in the top bar of the window, find a menu where you can choose the Board and the Port. Select the board "**Arduino Uno**". The Bot'n Roll ONE A+ will be programmed as if it were an Arduino Uno.

Fig. Selecionar a placa a programar

PORTS

Select both a Board and a Port if you want to upload a sketch. If you only select a Board you will be able to compile, but not to upload your sketch

If it does not appear instantly, you can click on "Select other board and port..." and the following menu will appear. Choose the board "Arduino Uno" and the corresponding COM port to program the Bot'n Roll ONE A+.

If no COM port is available, it is most likely that you have not correctly installed the VCP driver for the USB-to-Serial converter.

Open the Windows Device Manager and look for the item labelled "**Ports (COM & LPT)**". Expanding this item, you will see all the assigned COM ports.

"Silicon Labs CP210x USB to UART Bridge" is the designation that identifies the connection port to the Bot'n Roll ONE A+. (In the example from the figure, COM21 was assigned.)

Audio inputs and outputs Batteries Bluetooth 💻 Compute Disk drives Display adaptors lmaging device Keyboards Mice and other pointing devices Monitors 🚽 Network adapters Ports (COM & LPT) Find (com Labs CP210x USB to UART Bridge (COM21))
 Print queues
 Printers Processors E Sensors oftware components Software composed Software devices Sound, video and game controllers Storage controllers System devices Universal Serial Bus controllers

If the item "Silicon Labs CP210x USB to UART Bridge" does not appear, you will need to correctly install the VCP driver.

Fig. Portas COM no Gestor de Dispositivos

and the arduino uno q the board Arduino UNO R4 Minima Arduino UNO R4 MIRI Arduino UNO R4 WIFI Arduino UNO WFI Rev2 Arduino UNO WFI Rev2 Arduino Uno Mini Arduino Uno Mini Arduino Uno WiFi

Select Other Board and Port

BOARDS

Fig. Selecionar a Porta Série

🗄 Device Manager

File Action View Help

Show all port

-

 \times



B.4 LOADING A PROGRAM TO THE BOT'N ROLL ONE A+

In the Arduino programming environment, you will find several example programs that you can load onto the robot.

Click on "File -> Examples -> 01.Basics -> Blink" and a new window will appear with the code for this example.

Click on **"File -> Upload"** or press the arrow symbol pointing to the right to upload the program to the robot. Once the upload is complete, you should see the yellow L LED blinking every second!

File Edit Sketch	Tools Help					
New Sketch		Jno 🔻			,	∧ .o.
New Cloud Sketch						
Open						
Open Recent				ONE A+ library ing and writing		
Sketchbook		01.Basics		AnalogReadSerial		
Examples		02.Digital		BareMinimum		IE A
Close		03.Analog				
Save		04.Communication		DigitalReadSerial		_
Save As		05.Control		Fade		
Preferences	Ctrl+Comma	06.Sensors	•	ReadAnalogVoltage		

Fig. Carregar um programa de exemplo

	🐸 LED					
Fi	ile Edit	Sketch	Tools Help			
	0	€ €	🜵 Arduino Uno	Upload	∿	۰Q۰
F		LEC Uplo	ad			
			<pre>#include <bnroneaplus.< pre=""></bnroneaplus.<></pre>			
			<pre>#include <eeprom.h></eeprom.h></pre>			
1	<u>-</u>)					
			BnrOneAPlus one;			
	n-0	17				



Clicking on **"File -> Examples -> BnrOneAPlus -> …"** you will find all the example programs provided by botnroll.com specifically for the Bot'n Roll ONE A+.

In "File -> Examples -> BnrOneAPlus -> Basic ->...", you will find the basic programs designed to test all the robot's hardware. You should study and understand these small programs well!

In **"File -> Examples -> BnrOneAPlus -> Advanced -> ..."**, you will find more advanced programs that you should only study after understanding the simpler ones.

In **"File -> Examples -> BnrOneAPlus -> Extra -> ..."**, you will find programs related to the extra components that expand your Bot'n Roll ONE A+.

In **"File -> Examples -> BnrOneAPlus -> Fun Challenge -> ..."**, you will find programs related to the Fun Challenge.

In **"File -> Examples -> BnrOneAPlus -> Calibrate -> ..."**, you will find programs related to calibrating your Bot'n Roll ONE A+.

In **"File -> Examples -> BnrOneAPlus -> CrazyRace -> ..."**, you will find programs related to the CrazyRace challenge.





In "File -> Examples -> BnrOneAPlus -> Obstacles Sensor -> ...", you will find programs related to the obstacle sensors.

In "File -> Examples -> BnrOneAPlus -> Line Sensor -> ...", you will find programs related to the Line Following Sensor.