



Custom Primo Version

Name: Ivan Seidel Gomes

Repository: [GitHub.com/ivanseidel/Primo.IO](https://github.com/ivanseidel/Primo.IO)

Company: SESI-SP, Brazil

September 13, 2014

INTRODUCTION

I work at SESI-SP (SESI São Paulo) in Brazil, and we wanted to spread the knowledge that Primo could provide to kids inside SESI Schools, at the same time that older kids could also learn-by-doing it.

Goals

The idea is to produce Primo for a large number of schools in the state of São Paulo, Brazil, because of that, we required a prototype version. This version aims to be a prototype of the next version, that will be more replicable and scalable.

Problem

Electronics parts in Brazil are really expensive (actually, everything imported). Also, some of them are not easily accessible in the market, meaning that would require import of parts and components.

Solution

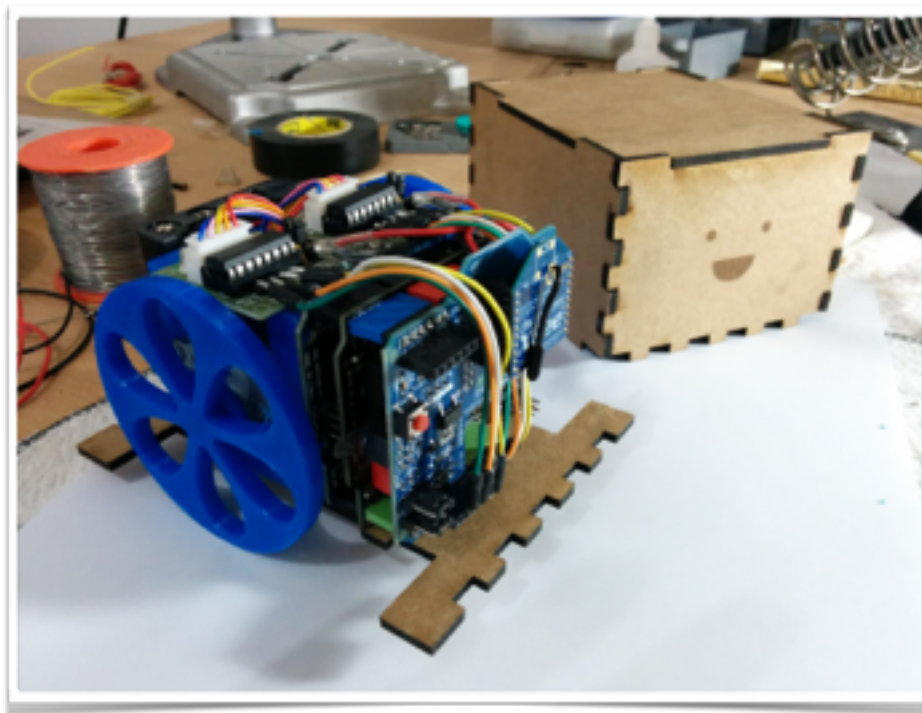
Because of that, some of the parts were modified and even improved, so that the next version could be produced with a lower material cost, and be easily built by kids with the given material, just like a “Primo Kit”.

Project Outline

The prototype was mainly focused on putting it to work. Some of the solutions found could also help others on building it. A few major changes in the Prototype:

- Use of Stepper motors. They require less hardware (only an small H-bridge), they are smaller, and also provide a precise control over (real) speed of wheels. One other great thing, is that if the motor are stucked (maybe a finger on the whell?), they nicely stops running without hurting body parts.
 - Use of 3D Printed parts for Cubetto. A few parts were designed to support motors, and also the Whells. But, they can be arranged to be Laser cutted as well.
 - A hole new Software. With this new Software, it can easily setup, and even change Communication, add more Brick types, add more commands, add more function...
 - Command Bricks were NOT build with Copper tape. Finding it in Brazil is hard, so the bricks were done using a small PCB with two contacts.
 - Main board uses only a few Copper tape, and more wires (that will be replaced by a PCB board on the next version).
 - Used only 4mm MDF boards. It's difficult to find in brazil, and it worked just great.
-

CUBETTO



BILL OF MATERIALS

Description	URL	Quantity	Unit Price	Cost
Arduino Shield - Prototipando	RoboCore	1	R\$ 100	R\$ 100
Stepper Motor + H-Bridge	DealExtreme	2	R\$ 10	R\$ 20
Arduino UNO (or Leonardo)	Proesi	1	R\$ 67	R\$ 67
Arduino xBee Shield	Proesi	1	R\$ 66	R\$ 66
xBee Series 1 - 1mw	LabDeGaragem	1	R\$ 117	R\$ 117
Esfera deslizante	RoboCore	1	R\$ 13	R\$ 13
3D Printed Parts				R\$ 0
Jumpers, Dual face tape, Super glue...				R\$ 0
Total				R\$ 383

BUILD FLOW (SIMPLE)

1. Print the Motor support in *Mecanica/Suporte_motor_passo.STL*
2. Print the wheels (or find one that fits the Stepper motor). It should have about 7cm of Diameter and 9mm of tickness (maximum). I used the wheel from another project of mine, that fitted just well. You need to resize it in tickness by about 50% (half of the height).
3. Glue the outside (Only the walls of Cubetto). Add some small pieces of [any-material-here] on the sides of the Walls (See picture *Photos-Cubetto/Cubetto_08.jpg*).
4. Place some double-sided tape on the base, and glue the Motor Support (it should be as shown on picture *Photos-Cubetto/Cubetto_03.jpg*. Leave enough space for your Battery behind it (measure it by adding the walls).
5. Put the ball roller on the bottom of Cubetto.
6. Attach the Arduino to the Prototype shield. Then Add the xBee Shield. Add some double-sided tape or even dual-lock on the side of the Motor Support, and add the Arduino there. See *Photos-Cubetto/Cubetto_07.jpg*.
7. Add the H Bridges on top of the motor support. Either glue, add a screw... Fix it there as shown on *Photos-Cubetto/Cubetto_03.jpg*
8. Solder the + and - of each H Bridge to the Prototype bridge and it's data pins as shown on *Photos-Cubetto/Cubetto_11.jpg*
9. Zip it all. See picture

TIPS AND TRICKS

- + Do not glue the walls to the base. You will need to remove them all the time...
 - + Do not glue the top as well. You might need to see inside, turn off/on, reset Arduino...
 - + Make a role on the wheel, so that you can plug the cable to the Arduino. Since the wheels I designed already had some holes, I just had to "open" it a bit more.
 - + If the stepper motors are not working really well... Well... Check the wiring order. They DO matter.
 - + I started with 4 AAA. Then changed to 8 AAA. Then changed to a 9V battery. Result: 9V battery are not so good. 4 AAA either. Use 8 AAA or any kind of 2S Lipo/LiFe Battery of about 7.4v.
 - + The height of my build was exceeding by about 1mm. It might be good to increase it a bit... But not a big worry for now.
-

PRIMO MAIN BOARD

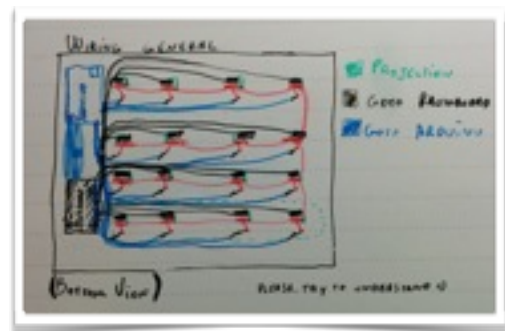
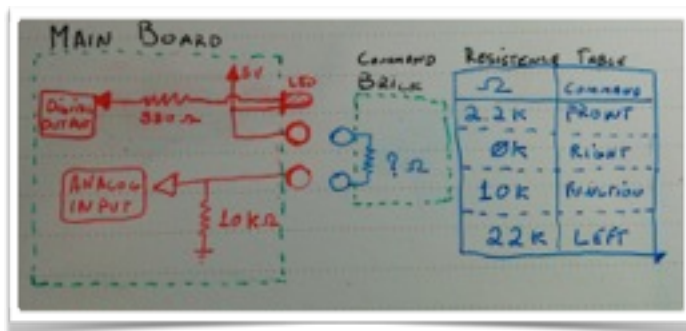


BILL OF MATERIALS

Description	URL	Quantity	Unit Price	Cost
Electrical Wire Tape	Leroy Merlin	1	R\$ 28	R\$ 28
or.. Cooper Tape	MercadoLivre	2	R\$ 15	R\$ 30
Arduino Mega	Proesi	1	R\$ 67	R\$ 67
Arduino xBee Shield	Proesi	1	R\$ 66	R\$ 66
xBee Series 1 - 1mw	LabDeGaragem	1	R\$ 117	R\$ 117
Small Protoboard (100 dots)	Proesi	1	R\$ 15	R\$ 15
Screws, nuts...				R\$ 0
Jumpers, Dual face tape, Super glue...				R\$ 0
Total				R\$ 323

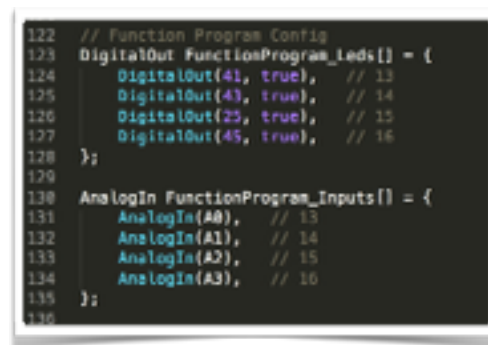
BUILD FLOW (SIMPLE)

1. You will only need 4 MDF Boards for the top. One for the bottom, and one for each side (find them). They will be grouped into two groups:
 1. The exterior with two layers. Top one with the drawing (arrows...), and bottom one with the block holes. See photo *Photos-MainBoard/MainBoard_01.jpg*.
 2. The Hardware layer. Here is where the fun (mess) will be. All electronics will be placed here, including Arduinos, Protoboards, wires... See *Photos-MainBoard/MainBoard_03.jpg*.
2. Add double-sided tape in the middle of each layer from above. NOTE: DO NOT GLUE BOTH GROUPS. See *Photos-MainBoard/MainBoard_21.jpg*. The two layers from top, are separated from the Bottom with a Screw nut. That's because the Cooper Tape will add a height to that layer.
3. Glue the walls. Same tips as the Cubetto. Add some reinforcements to the edge.
4. If you are using Cooper Tape: GREAT. Just cut it in parts of about 5cm. You might need to 'increase' the hole to fit. See *Photos-MainBoard/MainBoard_05.jpg* and *Photos-MainBoard/MainBoard_06.jpg*.
5. Place the Arduino and the Protoboard as shown on *Photos-MainBoard/MainBoard_09.jpg*. It's a good place to place it.
6. Because I don't like Cooper tape pretty much, I used wires. This is a major point, where will be simplified with multiple PCB boards for Command Brick contacts. I made some small changes to the Schematic of connection. Here is how each Block Terminal is wired and the big picture:



7. Solder All Positive wires, the Red ones in the photo: *Photos-MainBoard/MainBoard_14.jpg*. (NOTE: They will go to the Breadboard. You can group them all together...)
 8. Solder the POSITIVE terminal of all the Led's to the Positive terminal of the Command Brick contact. See *Photos-MainBoard/MainBoard_11.jpg*. (NOTE: The side of the Command Brick terminal doesn't matter, but since there is an LED on the side, try setting the closest side as the Positive, otherwise, you will have some trouble soldering the LED's to the terminal pads).
 9. Solder a resistor (330Ω) to the led's NEGATIVE terminal. Then solder an Jumper and wire it to the Arduino (it doesn't need to go to the Breadboard). See *Photos-MainBoard/MainBoard_15.jpg* and *Photos-MainBoard/MainBoard_16.jpg*.
-

-
10. Solder all Command Brick signal contacts, the Yellow ones in the photo: *Photos-MainBoard/MainBoard_14.jpg*. (NOTE: They will also go to the Breadboard).
 11. Hook each yellow wire to the Breadboard (place them in order, please...), and add at each one, a 10KΩ Resistor. See *Photos-MainBoard/MainBoard_18.jpg*.
 12. Now it becomes a (inevitable) mess. Hook each yellow brick terminal to the proper Analog Input pin on the Arduino with some Jumpers. See mess in *Photos-MainBoard/MainBoard_19.jpg*. You can configure it really easy. Go to *PrimoConfig.h* in *Softwares/Primo_Board*. You can change any pin you want. Note that they are in order, and grouped into **MainProgram** and **FunctionProgram**, that are the Normal Flow (MainProgram) and the Function Flow (FunctionProgram).



```
122 // Function Program Config
123 DigitalOut FunctionProgram_Leds[] = {
124     DigitalOut(41, true), // 13
125     DigitalOut(43, true), // 14
126     DigitalOut(25, true), // 15
127     DigitalOut(45, true), // 16
128 };
129
130 AnalogIn FunctionProgram_Inputs[] = {
131     AnalogIn(A0), // 13
132     AnalogIn(A1), // 14
133     AnalogIn(A2), // 15
134     AnalogIn(A3), // 16
135 };
136
```



```
61 // Main Program Config
62 DigitalOut MainProgram_Leds[] = {
63     DigitalOut(19, true), // 1
64     DigitalOut(29, true), // 2
65     DigitalOut(39, true), // 3
66     DigitalOut(31, true), // 4
67
68     DigitalOut(34, true), // 5
69     DigitalOut(23, true), // 6
70     DigitalOut(33, true), // 7
71     DigitalOut(32, true), // 8
72
73     DigitalOut(21, true), // 9
74     DigitalOut(24, true), // 10
75     DigitalOut(19, true), // 11
76     DigitalOut(30, true), // 12
77 };
78
79 AnalogIn MainProgram_Inputs[] = {
80     AnalogIn(A12), // 1
81     AnalogIn(A14), // 2
82     AnalogIn(A13), // 3
83     AnalogIn(A15), // 4
84
85     AnalogIn(A4), // 5
86     AnalogIn(A5), // 6
87     AnalogIn(A6), // 7
88     AnalogIn(A7), // 8
89
90     AnalogIn(A8), // 9
91     AnalogIn(A9), // 10
92     AnalogIn(A10), // 11
93     AnalogIn(A11), // 12
94 };

```

13. Next, you should really tight things up. Use something to make things look not-so-crazy.
 14. Invent your button. I used this push button: *Photos-MainBoard/MainBoard_23.jpg*, hot glued it on the hole, and super-glued it to the button cover, then soldered it and hooked it to the GND, and a Digital pin. (NOTE: The pin is being Pulled-Up in software, so you MUST set the other side to GND, not 5v).
 15. Workaround for xBee. Arduino Mega has many Serial ports. And both xBee and USB were creating lot's of problems. So, I decided to Hack things a little bit more, and changed the Serial pin from the Shield to the Serial1 from the Arduino (RX1 and TX1). See *Photos-MainBoard/MainBoard_25.jpg*. I have jumped the Serial TX and RX to TX1 and RX1.
-

COMMAND BRICKS



I guess that the only change in here, was that I made a small PCB to serve as the bottom pad. However, you can use the original Primo Brick instructions, they will work just well if you have the appropriate Copper Tape.

For the Bricks, I have used the following resistors with a 10K Pull-Down Resistor (added to the Breadboard):

- RIGHT: 0Ω (Just a wire jumper)
- FRONT: $2.2K\Omega$
- FUNCTION: $10K\Omega$
- LEFT: $22K\Omega$

If you need, you can change the resistors as well. To do so, make sure that the ADC value will vary at least 60 units from one another.

Take R2 as the Pull-Down Resistor (in my case, $10K\Omega$), and V as the Voltage (in the general case, 5v)

The ADC read by the Arduino, will be:

$$ADC = (V \cdot R2 / (R2 + YourResistance)) / V * 1023$$

Then go to *PrimoConfigs.h* inside Software/Primo_Board and alter the corresponding line value in the *BLOCK_STATES[]*:

```
34  int BLOCK_STATES[] = {
35      0,    // State 0: EMPTY
36      839,  // State 1: FRONT
37      320,  // State 2: LEFT
38      1023, // State 3: RIGHT
39      512,  // State 4: FUNCTION
40  };
```

Thats it.

SOFTWARE

There are two main programs inside the *Software* folder:

- **Primo_Cubetto:** Controls the Cubbeto by listening to the Serial and doing commands.
- **Primo_Board:** Controls the Board, flashes light, Compile, Generate commands, and execute them by sending it over Serial interface.

DEPENDENCIES

I used a few libraries to make the code Reliable, Generic, and easy to change if anyone needs it. It is based on my other Library called **ArduinoThreads** (<http://GitHub.com/ivanseidel/ArduinoThread>).

Also, it makes use of **ArduinoSensors** (<http://GitHub.com/ivanseidel/ArduinoSensors>) to read from IO, and debounce sensors.

As well as **LinkedList** (<http://GitHub.com/ivanseidel/LinkedList>), so that the program can be compiled into a generic *PrimoProgram*, and latter be sent over Serial.

And for Cubetto, I used **AccelStepper** (<http://www.airspayce.com/mikem/arduino/AccelStepper/>) to control the Stepper motor.

Make sure to have them installed.

It might look like a lot of things for get this going, but the goal is do do even more complex things with time, including: If's, Whiles, Wait blocks (Wait for Sound, Wait for touch...).

GET THINGS GOING

1. First, download those Dependencies (ArduinoThreads, ArduinoSensors, LinkedList), install them (copy to the *Libraries* folder of the Arduino), and Restart the Arduino Software.
 2. Load the Primo_Cubetto, and upload to Cubetto.
 3. Load the Primo_Board, and upload to the Board.
 4. One extra step, is configure the xBee. If you xBee is not configured yet, read this: <https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu>
 5. All ok =)
-