Metabot: a low-cost legged robotics platform for education

Grégoire Passault, Quentin Rouxel, François Petit, Olivier Ly

***Abstract*— This paper introduces an open-source 3D printed low-cost legged robot environment designed for educational purposes. The platform, called Metabot, is already operational and is currently experienced by teachers and pupils. The robot as a teaching aid, has several advantages: at first, it is stimulating for pupils, especially legged robots. Second, robotics is multidisciplinary and centralizes a broad spectrum of knowledge for the pupils; in particular, it allows to introduce programming in a concrete environment, which is now an important need in school. Finally, legged robots can illustrate concretely several matters of geometry.**

I. INTRODUCTION

Education is currently changing in secondary and high school. In France, the official national high school courses have introduced programming skills in teaching objectives. Actually, it mentions that pupils may learn *"Programming moves on a robot or a character on the screen"*[1]. Let us add that cross-disciplinary projects are encouraged, which further supports the use of robots.

Robots are indeed in all these scopes: they are cross-disciplinary, involving mechatronics, physics, mathematics, but also a lot of programming, for both low-level components and high-level behaviours. Let us also emphasize that robots have a motivating and stimulating role in such a context.

We presents the Metabot environment. It is centered around a quadruped robot (see Figure 1). The fact that the robot is legged increases the motivating character for the pupils. But more than that, this implies that the robot is holonomic. This fact opens new possibilities for dealing with the concept of trajectory for pupils. Also, legs offer a good context for illustrating several matters of geometry in a concrete way.

In the other hand, more and more on-the-shelf DIY components are available. An example is low-cost digital robotics servomotors that were released recently. Metabot uses XL-320 from Robotis[1]. One of the most interesting feature is a clutch-like mechanism that allows to avoid breaking the gearbox or the shaft when the motor can't provide enough torque. This makes low-cost plastic geared safe to use.

Today, the environment has been tested in several schools, in technology courses and in mathematics for programming. This is the second year of test. We also set up a robotics competition around the Metabot Environment: the Metabot League. At the moment, the Metabot is ready to be used at low cost.

Several other solutions exist for using robotics in education. Let us mention the Thymio robot together with its programming environment Aseba ([2], [3]). Let us mention also the robot Dash which uses a scratch-like environment for programming. These robots use wheels for locomotion and are not holonomic. We explore a solution with legs. Let us also mention le project Poppy[4] which proposes a humanoid robot for educational purposes, in particular to illustrate interaction with users. In the Metabot project, we investigate a low-cost solution (a few hundreds euros). Let us mention that we used Blockly as, which is a version of Scratch ready to be used in third party software.

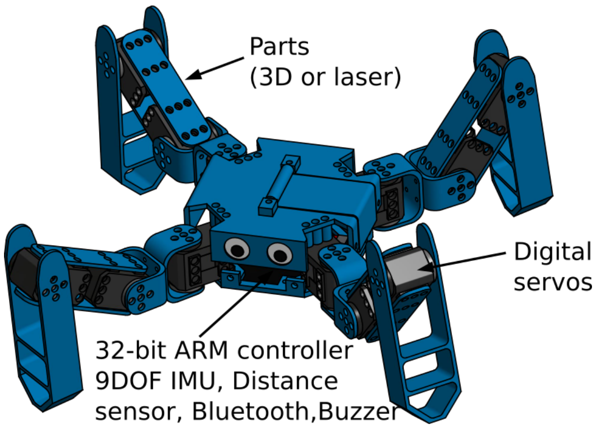


Fig. 1: Overview of the Metabot 12-DOF legged robot

The paper is organized as follows: we will discuss the design of a legged robotics platform for education. We will first describe the (hardware and software) robot architecture and then explain user experiences with it.

II. ARCHITECTURE

The robot design is open-source[2], which means that parts, electronics and software are available. People can build their own, buying on-the-shelf components and 3D-printing parts. It is also sold as a kit[3], that contains all the required parts for more convenience.

Selling a robot as a kit lowers the price, and mounting it is an interesting experience for the end-users. Metabot is mainly based on plastic rivets that are fast to assemble and disassemble. Having a robot open-source with documented

Three authors are with the *Rhoban team*, LaBRI, University of Bordeaux, France. Emails: {gregoire.passault, quentin.rouxel, olivier.ly}@labri.fr, François Petit is a mathematics teacher in secondary school.

[1]http://www.education.gouv.fr/pid285/bulletin_officiel.html?cid_bo=94753

[2]https://github.com/Rhoban/Metabot/

[3]http://metabot.cc/

parts and assembled by end-user is comforting because it allows later repair and maintenance, like replacing a motor.

A. *Hardware considerations*

The first design was made up with 3D printed parts. This is a clearly good choice to bootstrap such a project, since it quickly leads to a viable proof of concept. It is also an interesting asset in the educational world, since more and more schools are getting equipped with 3D printers. The robot was sold with and without 3D parts, letting the end user decides if he wants to make it himself or buy it. The next version of the platform will be cut from PMMA (using laser) or ABS (using milling) sheets. This is a good trade-off to avoid affording the cost of mould and injection.

As controller, a Cortex-M3 is used, which is one of the most widespread arm-based microcontroller, running at 72mhz, 120KB flash and 20KB SRAM. There is no operating system.

The digital servos are based on half-duplex serial daisy chaining. All the servos are addressed by an Id on the bus.

We decided to use bluetooth as communication because it is present on most modern computers and phones, with a simple pairing procedure. On-the-shelf bluetooth to serial chips are now available at reasonably low cost.

To secure the LiPo battery power supply, a fuse is added. If the input voltage reaches a too low level, an alarm is triggered to signal it to the user. If the battery isn't changed, the fuse is destroyed, making the electronics circuits opened and avoiding any problems related to over-discharging the battery.

B. *Motion*

The robot motion is similar to the one described in [5], the inverse kinematics was solved analytically to be able to control the leg position in the Cartesian body of reference. Figure 2 is a representation of the degrees of freedom of each leg. Kinematic is a direct application of trigonometry that is interesting to mention. This is a simple example of real use case of trigonometry to be presented to pupils.

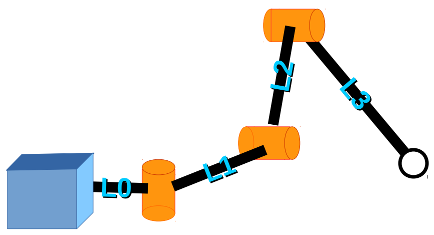


Fig. 2: Representation of the kinematic chain of a leg of the robot.

Legs are following splines, and their phase is trot pattern, which means that diagonally opposed are moving simultaneously. This pattern has experimentally proven to be efficient and stable in much situations.

The control parameters of the robot are $(\dot{x}, \dot{y}, \dot{\theta})$, respectively speed along x, y axis and rotation speed. This results in an holonomous walk. The robot can reach speeds as high as 40cm/s.

Other parameters can be tuned, making the robot more or less efficient regarding the floor frictions and possible obstacles. They are footstep height, walk frequency (the number of step per second), the height of the robot etc.

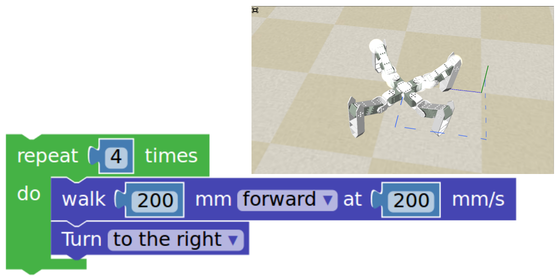
The robot is designed symmetrically, the front is virtual and can be remapped on-the-fly. The current virtual front is display using color LEDs embedded in servo-motors.

C. *Virtual machine*

In order to get user behaviours running on the robot, we designed an application-specific virtual machine, that is able to run on small micro-controllers (with no operating system). This way, user code can be compiled to small binaries that can be embedded and executed with a good control on programs. This design was also proposed by [6], and used in production on the Thymio robot[2].

Virtual machines also feature sand-boxing execution from the robot firmware itself, avoiding errors such as illegal memory access or locks. Access to hardware is done with native functions, which can be called in the virtual machine bytecode.

D. *Programing environment*



```
repeat 4 times
do  walk 200 mm forward at 200 mm/s
    Turn to the right
```

Fig. 3: An example of program and 3D view in the web application.

A first obstacle in teaching programming is dealing with theoretical (what is an algorithm?) and syntax (how do I write my algorithm?) on the same time. A common way to avoid this is using visual programming language, which are similar to textual ones, with no syntax errors possible. This field was investigated by the MIT, with StarLogo and Scratch[7], which is today one of the most widespread. This inspired App Inventor, an application designed by the MIT and Google to design mobile applications. Google then proposed Blockly[8], a library to design visual programming languages. All these technologies are web-based, involving Flash (Scratch) or JavaScript (blockly).

At this time, Blockly is the best choice for interfacing with custom logics (i.e customizing blocks and code production). It was designed with developers as first target, making it more versatile and customizable, with a clear documentation.

Scratch is in first place an on-line education application, which can be customized by writing extensions that can add blocks which communicates through, for instance HTTP requests with custom applications. However, it is not designed to generate custom code, and for instance produce an embeddable program for a robot. Blockly let access to the "parse tree", which make any generation possible on the top of the blocks scene. Moreover, Blockly have a great and active community.

Having the application web-based allows a simplified deployment, better portability and prone to upgrades. Curious users can give a try to the environment just browsing the site.

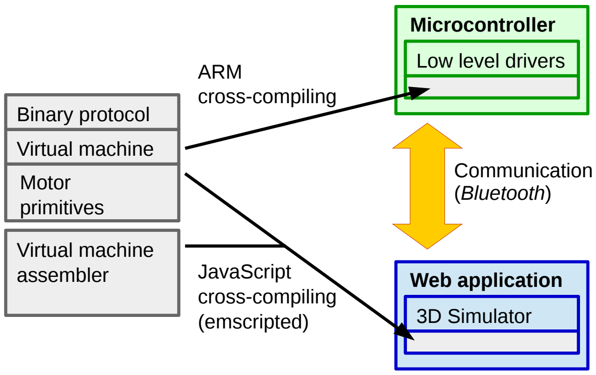


Fig. 4: The architecture of the robot software programming environment.

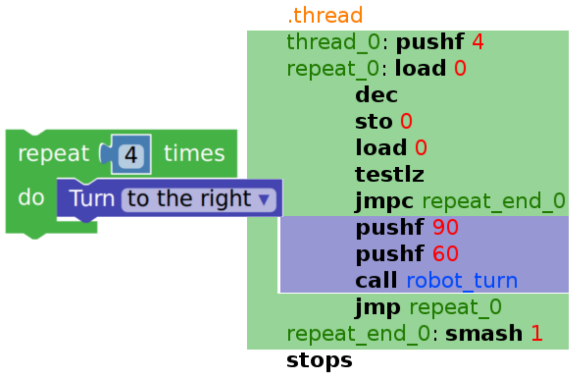
An interesting point in the architecture is that code is actually shared between the robot and the web simulator (see figure 4). This is possible thanks to emscripten[9], an LLVM-to-JavaScript compiler which allows to compile native code (for example C/C++) to JavaScript. Thus, parts of the robot firmware, virtual machine and bytecode assembler can be embedded directly in the simulator available on the web application. The 3D simulator takes advantage of WebGL and three.js[10][11]. This allow consistent behaviour between the simulated behaviour and reality.

Blockly allows to define custom blocks and code generation logics. In this case, we generate assembler code for our custom virtual machine, which is then assembled to bytecode.

The machine uses extensively stack to deal with variables. An example is shown on Figure 5. Native methods are used to do all the platform-specific operations, such as controlling robot motion and retrieving information from sensors.

When these methods are called in the bytecode, the virtual machine calls an user-defined method that can access the stack, and then exchange data between the virtual machine and the native world.

Multiple programs can be loaded in the virtual machine, interacting with each others using global variables, and they can contains multiple tasks. This makes concurrent executions possible, we introduced task blocks that can pause, start and stop each others. An example, shown in figure 6, is making the robot doing hexagonal path, and simultaneously



```
.thread
thread_0: pushf 4
repeat_0: load 0
          dec
          sto 0
          load 0
          testlz
          jmpc repeat_end_0
          pushf 90
          pushf 60
          call robot_turn
          jmp repeat_0
repeat_end_0: smash 1
stops
```

Fig. 5: An example of program and its assembler instructions. The robot_turn method is a native method in the robot firmware, that takes as arguments the degrees (here 90) and the speed (here 60 deg/s)

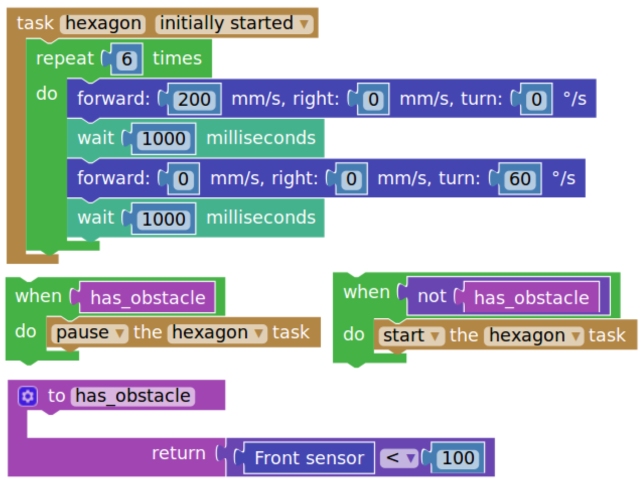

task hexagon initially started
repeat 6 times
do forward: 200 mm/s, right: 0 mm/s, turn: 0 °/s
wait 1000 milliseconds
forward: 0 mm/s, right: 0 mm/s, turn: 60 °/s
wait 1000 milliseconds
when has_obstacle
do pause the hexagon task
when not has_obstacle
do start the hexagon task
to has_obstacle
return Front sensor < 100

Fig. 6: A complete example containing loop, test, function, threads and tasks control

detects obstacles, which will pause the main behaviour, and resume it if the obstacle is no longer present. The blocks that are not connected to the other ones are considered as simultaneous threads that are run in the same time as the others. This the case for the *when* block in the example. Threads can also be named tasks, like *hexagon* from the example, that can then be controlled.

To control the robot, we decided to provide both distance and speed methods. Which mean that turning to the right can be achieved with the "turn to the right" block, but also using "turn at 90deg/s" and "wait 1s" blocks. This allows blocking and non-blocking calls, which are both useful in different situations.

The programming environment offers the possibility to load produced bytecode either on the simulator or on the actual robot.

The simulator executes the bytecode on the virtual machine that is cross-compiled with emscripten, and reads motor target angles to synchronize the 3D view. Robot moves are simulated using dynamics parameters.

E. *Mobile application*

In order to control the robot, a mobile application was developed. It allows to control the robot, but also to tune the parameters and thus behave as a remote control taking advantage of the bluetooth stack. The robot parameters can also be changed to try different walks (see figure 7).

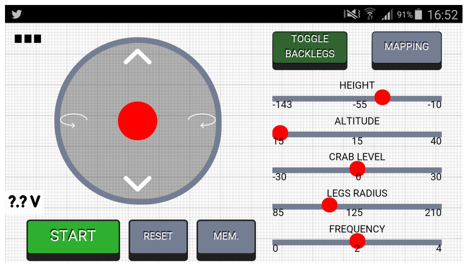


Fig. 7: A screenshot of the mobile application screen that can be used to tune the robot parameters

Another interesting feature of such an application is to start and stop behaviours that were previously designed in the web environment and loaded on the robot memory as virtual machine bytecode. This makes the robot and its programs usable without having a computer.

III. EDUCATION EXPERIENCE

A. *Learning programming*

Using a robot to learn programming is explained by the concepts of tangible interfaces[12]. The robot helps the transition from the material to the abstract world, but it is also itself a motivating object.

The approach of the pupils toward the programming environment is trial-and-error. This is encouraged by the visual environment, because blocks can be found in a lateral panel that acts as a library and is more intuitive than a documentation.

Let's explain a pedagogical example with a concrete sequence of incremental questions.

First, we can ask the pupils to make the robot walking following a square path, without giving them any advice or documentation. Most of them will eventually use the robot control blocks to make it do the path using a simple sequence (go forward and turn right, copied/pasted four times).

We then explain them that they could use loops, making them understand that they can do the same thing with less blocks and in an easier way to change the program.

We then ask them to use speeds instead of distances. This is a direct application of both cross multiplication and angles (like turning at d degree per seconds during t seconds). To insist on this point, we ask them to follow an hexagonal path.

Another exercise, on the top of this, is to follow the hexagon and pause when there is an obstacle, which can introduce tasks and events, like depicted on figure 6.

Within this example, we introduced a few programing principles like loops, functions and events.

This has been experimented in french secondary school in Math and Technology courses for 4ème and 3ème levels (French cursus), which corresponds to 13 and 14 years old pupils. In technology, pupils design the Metabot structure and in Mathematics, they program it: At first, the teacher introduces the Metabot programing interface; The initial session consists in having the Metabot cover a square route on the computer screen, after the relevant speed formula has been explained. This first session enables the students to grasp both the programing environment and the direct practical utility of loops.

The next session rounds up the project, by requesting the students to build up a full infinite loop, with the additional challenge of having to keep two circles from overlapping. As a rule, all the students get through this step successfully.

The following steps make it possible to master the various functionalities of the robot (height, lighting of the leds…) , the ultimate goal being to create tools that will eventually intervene in the building up of more complex programs, as required by the contest the students enter.

A specific session is devoted to a web search for the interface commanding the robot via a mobile phone. The technical challenge generates high motivation among groups.

B. *Importance of competition*

An important way to stimulate pupils on learning robotics and programming is through competition[13]. One of the most famous is indisputably the First LEGO League[14], a competition mostly focused on constructions made with LEGO blocks.

In the same scope, we started the Metabot League initiative, a competition involving Metabot robots (see figure 8).

Fig. 8: Poster announcing the Metabot League 2015

We proposed a dancing competition where the pupils had to create a dance using the programing environment, they could choose the music and customize the robot. This is an interesting challenge, because it is not technically restrictive, and resulted in an entertaining show (see figure 9).

Team sizes ranged from four to six people, aged from 12 to 16 years old.

IV. CONCLUSION

We experimented a legged robotics platform that suits education purposes with good perspectives in that domain.

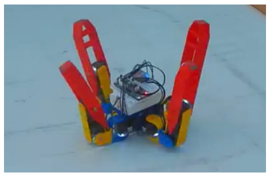


Fig. 9: A customized Metabot dancing during Metabot League 2015.

Fig. 10: The Metabot League 2015 dance competition taking place in the amphitheatre of an engineering school.

We noticed that the trial and error approach is a good way to let pupils start using programming tool, and that a visual programming environment is quickly handled and offers a comprehensive view of what programming is to bootstrap the lessons. We also encouraged them to participate to a competition, which clearly motivated them to learn.

The platform is still under development, a new version will be released, adding distance sensor and inertial measurement unit (mainly for yaw estimation).

The hardware of such robots could evolve toward a more powerful controllers, with promises of low-cost embedded systems (like Raspberry Pi Zero[15]). This would allow using actual programming language instead of custom virtual machine, but also having more computation capabilities.

In future work, teaching material will be produced in order to be able to teach with out-of-the box lessons. The programing environment will also be enhanced, adding granularity in the control of legs and motors, like for instance being able to control the leg tips using (x, y, z) position.

References

[1] C. N. Thai, "Actuator position control basics," in *Exploring Robotics with ROBOTIS Systems*. Springer, 2015, pp. 87–102.

[2] "Thymio, an educational robot with programming environment," http://www.thymio.org/.

[3] F. Riedo, M. Chevalier, S. Magnenat, and F. Mondada, "Thymio ii, a robot that grows wiser with children," in *Advanced Robotics and its Social Impacts (ARSO), 2013 IEEE Workshop on*. IEEE, 2013, pp. 187–193.

[4] M. Lapeyre, S. N'Guyen, A. Le Falher, and O. P.-Y., "Rapid morphological exploration with the poppy humanoid platform," in *Proceedings of the 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2014, pp. 959–966.

[5] B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut, "Omnidirectional locomotion for quadruped robots," in *RoboCup 2001: Robot Soccer World Cup V*. Springer, 2001, pp. 368–373.

[6] S. Magnenat, P. Rétornaz, M. Bonani, V. Longchamp, and F. Mondada, "Aseba: A modular architecture for event-based control of complex robots," *Mechatronics, IEEE/ASME Transactions on*, vol. 16, no. 2, pp. 321–329, 2011.

[7] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, *et al.*, "Scratch: programming for all," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[8] Google, "Blockly: a library for building visual programming editors," https://developers.google.com/blockly/.

[9] A. Zakai, "Emscripten: an llvm-to-javascript compiler," in *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. ACM, 2011, pp. 301–312.

[10] R. Cabello, "Javascript 3d library," https://github.com/mrdoob/three.js.

[11] B. Danchilla, "Three. js framework," in *Beginning WebGL for HTML5*. Springer, 2012, pp. 173–203.

[12] P. Marshall, "Do tangible interfaces enhance learning?" in *Proceedings of the 1st international conference on Tangible and embedded interaction*. ACM, 2007, pp. 163–170.

[13] A. Bredenfeld, A. Hofmann, and G. Steinbauer, "Robotics in education initiatives in europe-status, shortcomings and open questions," in *Proceedings of International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2010) Workshops*, 2010, pp. 568–574.

[14] D. Oppliger, "Using first lego league to enhance engineering education and to increase the pool of future engineering students (work in progress)," in *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, vol. 3. IEEE, 2002, pp. S4D–11.

[15] "Raspberry pi zero: the $5 computer," https://www.raspberrypi.org/blog/raspberry-pi-zero/.