Development of a multi-purpose, modular, low cost robotic arm for education and unmanned platforms

Gregorio Juliana - *Universidad Politécnica de Madrid*

gregojquiros@ieee.org

***Abstract*— Why use it, how was it designed and how does it work. Advantages vs. other solutions and potential applications.**

I. Introduction

When building robots and unmanned systems, it is usual to design all-in-one solutions meant to accomplish a certain range of missions. Depending on the situation, these solutions have to be adapted and reequipped with different payloads and actuators and the latter are usually very specialized and expensive.

Recognizing the need of a multi-purpose actuator which can be used in a wide variety of situations, we have designed a modular robot which can be easily built, attached and re-imagined for different situations at a cost that makes prototyping and testing fast and cheap. By using 3D printing and cheap electronics, this robotic arm can be built in less than ten minutes, using only two screws per joint and adding up to six degrees of freedom with the most basic setup, while keeping the hardware cost as low as 50~60$ depending on the configuration. Its modular nature means it can be adapted to many situations with the minimal effort, as the swappable parts are also the 3D printed ones.

II. Design considerations and key features

A. Cheap

As a prototyping tool, the arm must be cheap. Its main focus is not to offer surgeon-like precision or extreme weight lifting. This solution provides a fast alternative to test a specialized actuator that can be much more expensive on the final product. This is one of the reasons why the parts are 3D printed, and servos are used instead of steppers.

B. Easy to build

Its price and purpose make mandatory the ease of building. Also, it has to be reparable or even replaceable in no time. The joint design has to be effective, reducing friction and wear without the use of metal components like bearings.

C. Easy to program

All of the above would be of no use if this device were a nightmare to program. As the price is kept low and can be a very attractive option for educators intending to introduce students into robotics, an Arduino compatible custom board is used. Basically, the main board is an Arduino Leonardo modified to suit the needs of the system.

D. Easy to use

Due to its high (and always expandable) number of degrees of freedom, the software loaded in the Arduino board has to offer an easy way to interface with the arm and design intuitive control systems. With this in mind, a custom protocol was written in order to be able to control the arm via UART, meaning almost any computer application can access it. This includes voice control, gesture recognition and machine vision applications.

III. Mechanical design

Taking into account the rules mentioned before, the parts were designed and printed using a basic 3D RepRap style machine (Prusa i3):

A. Servo choice and attachment design

Two kinds of servos are used. For the vertical joints and the gripper, small 1.2kg micro servos while bigger 6.9kg standard size servos are used for the two base horizontal joints. Following the 'easy-to-build' rule, the attachment of the servos to the plastic was the most challenging part. The joints had to be strongly and securely attached, but kept simple at the same time. The final design only requires one screw and the use of the included plastic cross almost every commercial servo comes with.

B. Base design

Due to the adaptable nature of the arm, the base must meet only a few requirements, so any kind of attachment can be developed for different purposes. Because of this, the base only requires to be designed to have a servo pointing upwards so it connects with the rest of the device. The basic design built for the prototype attaches itself to an inexpensive methacrylate square thanks to three suction cups.

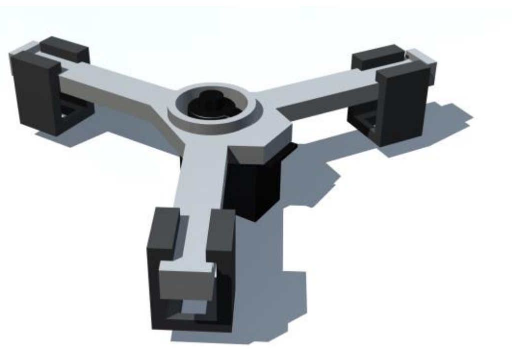


Figure 1

C. Modular joint design

Each segment has two plastic parts, which add two degrees of freedom. Virtually any number of segments can be added to a given arm, as long as the electronics have enough pins to control them (thirteen servos supported with our board design) and the servos enough power to lift them.

Each segment requires two servos, a micro one and a standard one.

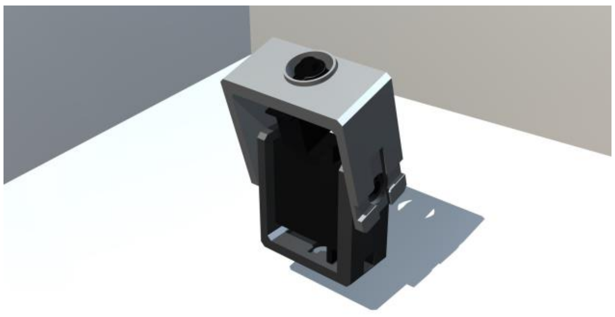


Figure 2

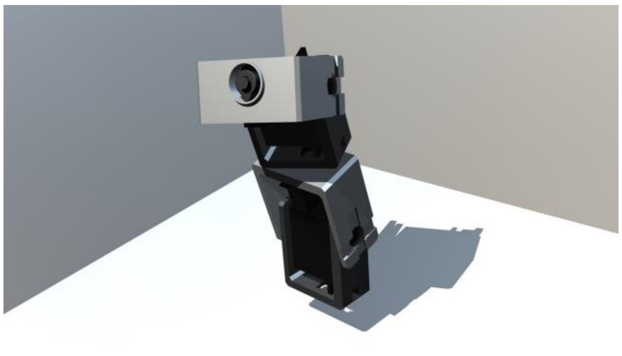


Figure 3

D. Gripper design

Due to the modular nature of the project, the top actuator can be swapped to fit the needs of the user, but for our basic prototype we included a simple two-piece gripper. It requires two micro servos, and can hold and lift light objects.

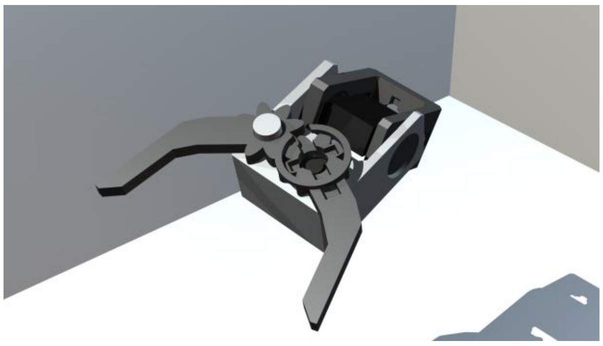


Figure 4

IV. Electronics design

Following the 'easy-to-use' principle, the popular and well-documented Arduino platform has been used. During the first testing phase a Duemilanove board was chosen as main board, but this device is incapable of delivering all the current the servos needed. On the other hand, the price and size of the standard Arduino boards made them unsuitable for the project, so a custom PCB was made. It is essentially an Arduino Leonardo with the power section redesigned and a different form factor. On the other hand, changes were made to the pin layout so standard servo connectors can be directly plugged into the board, from which they receive power and are directly connected to the signal pins. This allows us to use a well-tested, stable and documented platform and at the same time is a solution that perfectly fits our needs. Schematics can be found in appendix 3, and design considerations after the electrical tests are described in appendix 1.

V. Software design

The software for the robot is divided into two sections, the firmware inside the board and the client application. This dual design is what truly reveals the power or the platform, because the board only manages the servo positioning and communication, while a computer makes the heavy calculations based on whatever control system is required.

A. Dual control system and communication protocol

The approach used on the board code was asynchronous, meaning that any servo can be repositioned at any given time, independently from the rest and regardless of any previous order. Using a scheduler-tasks programming model inside the microcontroller, this is achieved seamlessly and transparently to the user, who only needs to send UART messages to the board following our protocol.

This protocol is based on raw bytes transmission following the pattern shown in Fig.5:

The control bytes can be either "200", meaning the next bytes are movement data, or "201", meaning we are disconnecting from the arm, so it puts itself into a standby state. It can be argued that those numbers (and the servo positions, because they range between 0 and 180 degrees) could be sent using less bytes, but the protocol was designed so it could send arbitrary large numbers in case it was required for the future. The baud rate (115200) is more than enough to send the bigger packets.

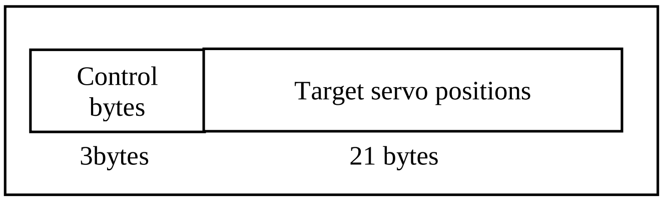


Figure 5

B. *Control application and Kinect integration*

Currently two versions of the control application are being developed, one written in Java due to the multiplatform support and another written in C# using Microsoft's .NET framework.

Following the 'easy-to-use' rule, three different example control systems were developed, in order to provide a good preview of different interfaces the arm can be used with. The first one is based on sliders, which set the position of the servos. This method is slow, but simple and precise.

During the coding phase, the need of pre-programmed movement sequences appeared, so it was built into the app. Sequences are triggered via voice commands or an in-app selector. In order to make easier to the user to design their own commands, there is a configuration file which can be edited to add new movement sequences. The syntax can be found here:

```
COMMAND; CommandName; VoiceResponse;
KEYFRAME; N0; N1; N2; N3; N4; N5; N6;
KEYFRAME; N0; N1; N2; N3; N4; N5; N6;
KEYFRAME; N0; N1; N2; N3; N4; N5; N6;
ENDCOMMAND
```

Where NX are numbers between zero and 180 that indicate the objective angle each servo must reach. Each keyframe is a fixed point that the arm must achieve within 500ms after the last one, does not matter where it was before. The firmware will interpolate the intermediate positions.

Finally, and in order to achieve a natural user interface, a Kinect sensor from Microsoft was added. The sensor maps the arm joints to the operator's, and then the data is sent to the robot. This is done in two parts. The right arm controls the horizontal joints angles, and the left arm control the vertical ones mapping them as follows:

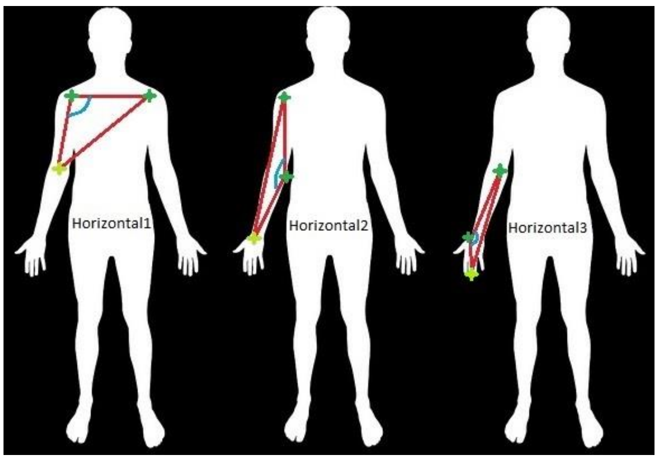


Figure 6

For horizontal movement, the dark green crosses are the reference joints, while the lighter ones mark the one responsible for the movement of the corresponding segment. The mapped angles are shown in blue.

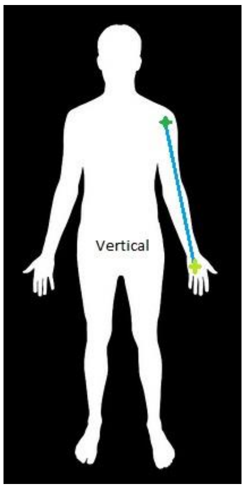


Figure 7

Vertical movement works differently, as the sensor maps the difference between the Z coordinate of the operator's shoulder and hand, making all the vertical segments rotate the same amount based on this.

The gripper is controlled using the operator's left hand. Making a fist with it will close the clamp, while opening it will make the robotic counterpart do the same.

This makes control much more intuitive, and allows different kinds of interactions with the arm. Combining voice and gesture controls is also possible. This last features are only available in the C# version of the application, due to the APIs used.

VI. Further work

A. *Integration in a simple rover*

As a future educational work, the arm will be integrated into a simple Bluetooth-enabled rover, which will be used as demonstration platform for the device. This side project will allow us to experiment with new control possibilities and test it in different environments and missions.

B. *Rational mechanical capacity analysis*

As seen in appendix 1, the arm is going to be tested under different scenarios in order to determine its capabilities. A theoretical analysis will also be performed in order to contrast the experimental results.

VII. Conclusions

This project, which started as the design of a didactic robot, soon revealed itself as a powerful prototyping tool that could be used in many other fields. The potential applications on unmanned vehicles and autonomous platforms that need specific actuators are uncountable, and its low price and adaptability allows to quickly test different configurations and later substitute it with specific tools. On the other hand, it can be used as it was initially designed, since it is very user-friendly and easy to build. Depending on the target audience, the project can be handed pre-built, with the firmware pre-programmed or unassembled, allowing even children to use it and learn robotics.

APPENDIX 1

Mechanical an electrical characteristics:

When designing the electronics, several tests were performed in order to exactly determine the power consumption of the device. This was necessary to correctly choose a power supply and avoid damaging the servos or the control electronics.

Both the control and the motor supply parts need 5V in order to function properly. This simplified the design, allowing the servos to be directly fed from the AC/DC power supply, while the control electronics have a 5V/800mA regulator beforehand. Considering that the control electronics could be used as a regular Arduino board in another context, the USB power was decoupled from the main lines that feed the servos, allowing to power only the board if necessary by just plugging it to a computer. The final tests determined that the overall current consumption was between 700mA (servos resting in full vertical position only holding the arm's own weight) and 2.7A when all of them were powered and trying to reach an extreme position while blocked. This is why a 5V@4A max power supply is used in the project, as it was the cheapest commercial one that filled our requirements.

As the project is still under development, the mechanical testing phase is still to be fulfilled. This includes, among others, measuring the maximum payload lifting and mechanical resistance.

A rational mechanic analyze could be done in order to theoretically get its load capacities and will be performed as further work for this paper. It is important to remember that the arm is composed by 3D printed parts and low cost hobby servos, so the load capacity is conditioned to the ones used, the structural stability of the modular parts geometry and the material used for printing the parts.

Here we will describe a simple test that will be performed to a slightly modified version of the arm (mostly related to the base fixation mechanism, which needs to be tougher for the tests). The aim is to experimentally determine the maximum load this arm is safely able to operate with and to quantify the deviation from the theoretical rational mechanic analysis of the forces the arm is able to manage.

Proposed experimental setting and tests

The arm will be placed and fixed in a perfectly flat table exactly at the (0, 0, 0) point of a reference inside a trihedral net conformed by three PVC panels of 0.25m2 each and a video camera is going to be placed for the later video analysis.

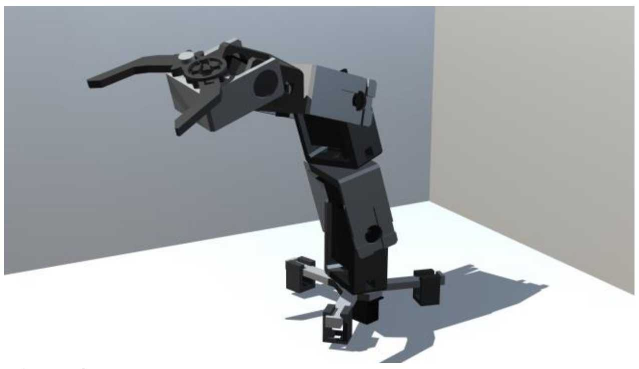


Figure 8

A set of small lab weights (between 50gr and 2Kg) will be used as load for the arm during this tests.

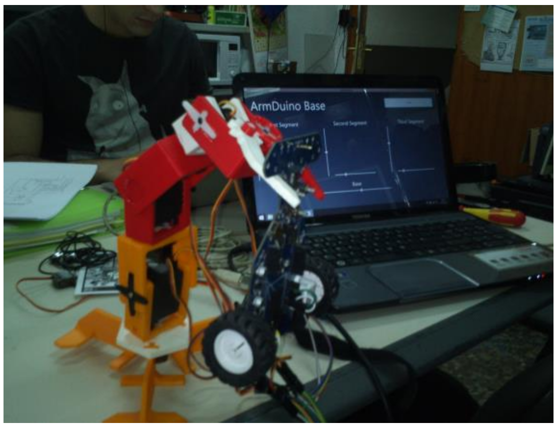
Every test will begin with the arm vertically extended and the weight in a different position (X, Y, Z).

Four different tests have been planned:

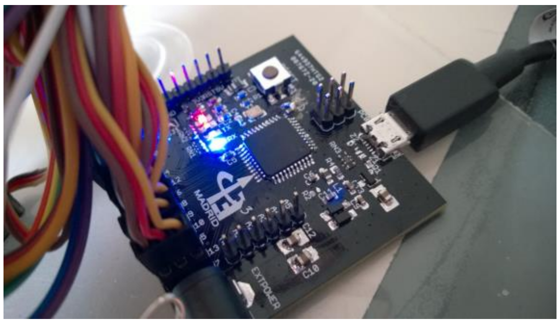
1. Simple elevation: Taking a weight from a (X, Y, Z) position with the clamp and lifting it to the 90º position of the middle section of the arm.
2. Elevation plus rotation: Taking a weight form a (X, Y, Z) position with the clamp and lifting it to the 90º degree position previously achieved. Once there, rotate the base from 0º to 90º.
3. Transversal elevation: rotating the base servo at the same time the arm is lifting the weigh until reaching the 90º position of the middle section of the arm.
4. Turning against a torque: the clamp will turn a metal arm braked by a hanging weighs.

Appendix 2
Developed prototypes:

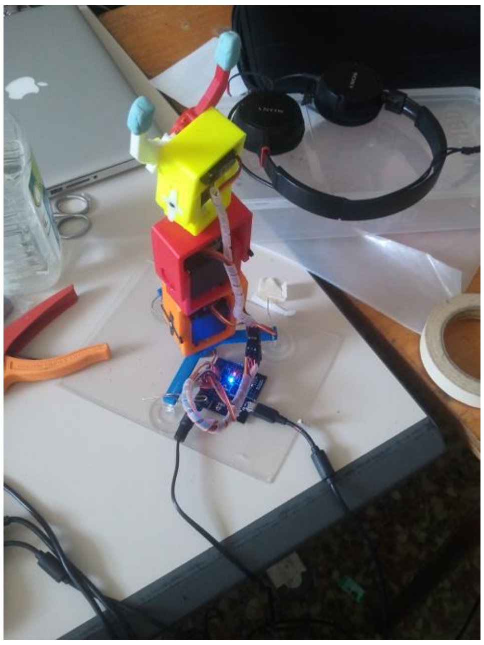
1 First working prototype lifting a line-following robot



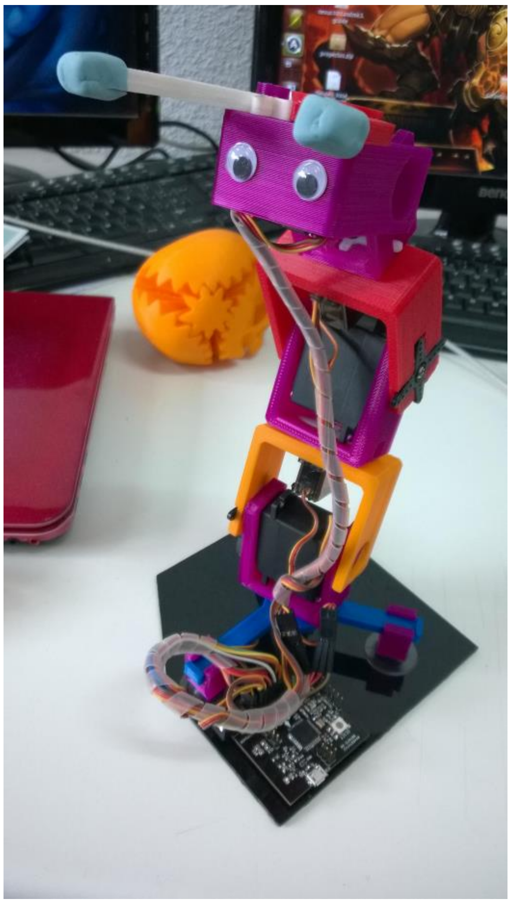
3 Main control board



2 Second prototype with redesigned base and gripper

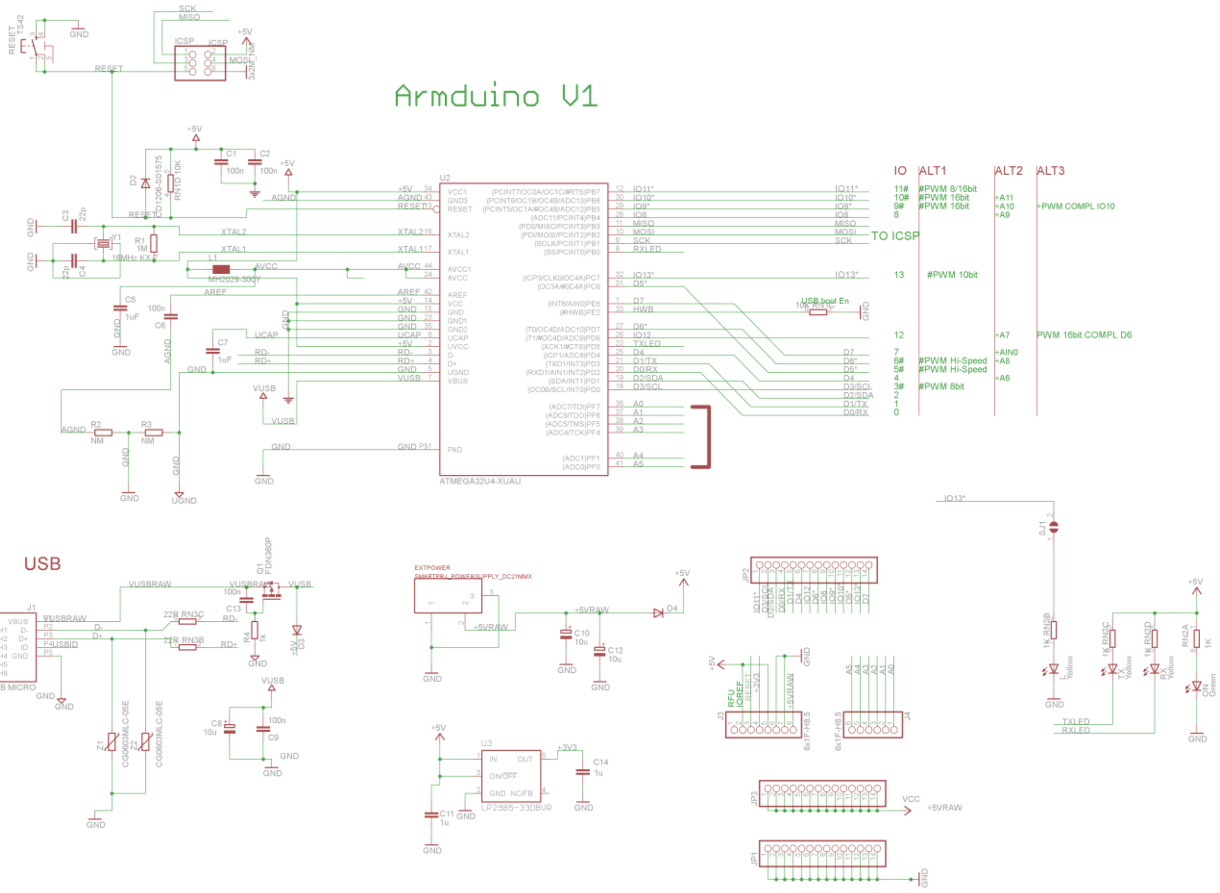


4 Latest model used in a presentation for children



APPENDIX 3

Main board schematic



ACKNOWLEDGMENTS

Thanks to the UPM's IEEE student branch for the materials, 3D printer, time, understanding and support. This project would not have been finished without them. Special thanks to David M. for his invaluable help during the electronics design part.

REFERENCES

[1] Arduino foundation website, Arduino Leonardo board reference and schematics: http://arduino.cc/en/Main/arduinoBoardLeonardo
[2] Microsoft's Kinect website, Kinect SDK description and human interface guidelines: http://www.microsoft.com/en-us/kinectforwindows/
[3] Project's log and documentation in Trello: https://trello.com/c/Cj7JInNH
[4] Software repository hosted in GitHub: https://github.com/Thunkar/ArmDuino
[5] Java RXTX fork website. Used for serial communication on UNIX systems: http://mfizz.com/oss/rxtx-for-java