INITIAL EFFORTS OVER A FREE, DISTRIBUTED AUTOMATION SYSTEM
As an alternative to Cloud Centric AI API based Automation

—

***** PRELIMINARY RESEARCH *****
SIMPLEMACHINES PIautos (Lambda Cancri)
(Sagittarius A* Decentralized Automation)

Sergio Sorrenti[†,*], Andrea Perrone[†],
Raman Gopalan[†], George Orais[†], Löic Maury[†]
Federico Briata[†], Dado Sutter[†], Nunzio Raciti[†], Marcus Jansson[†]

Abstract. In this document we aim to release many of the ideas that will forge the development of an innovative form of distributed automation as a valid and free alternative to what is consolidated as Cloud Automation associated and not associated with the various APIs of the Internet Centric AI.

Keywords: automation, robots, home, industry, building, cities, internet-wide

Media Summary

Simplemachines joins in the conceptual and implementation effort to go further in automation in the resolution of known and lesser known problems (probable future problems), to overcome the known ways of Distributed Automation and to consolidate old and new methods making them simple and usable by all the more experienced and less experienced programmers. Experimenting on known and current hardware like sub-dollar Microcontrollers such as the ESP32 (and future ESP64), our Demo board Mizar32, general purpose gateways like the Raspberry Pi, Orange Pi and similar. The use of all the existing Free Software has been plumbed and screened, making the implementation surgical and filling the necessary straightforward. The State of the Art as RTOS(Apache NuttX forked as Alcor OS), Interpreters(Python, Lua, Javascript, Forth, Lisp, F, OpenPLC) Remote-Controlled IDE (Nodered forked as General-Node) as well as GNU Operating System (Debian with Mira meta-package) to be adapted and configured in a documented, simple, innovative and functional way. Used in concert to deploy a single node (a simple machine), to multi-local-nodes (more complex machines) to networks of nodes (home, building and industrial automation) to entire cities (smart-cities) or in creating distributed automations over the Internet(rotate a PWM handle in Tokio and close the Niagara falls). From all this complex needs we aim to trace a simple way in order to fulfill all them in an incremental, scalable, flexible aproach. The alternative use of Secure DNS database, in making the localization or purpose of the nodes intuitive, the use of IPv6 where and how it easily happens, the disengagement of the gateways where not necessary, the non-centralization of automations, redundancy and fault-tolerance are also peculiar characteristics of our research in **Simplemachines PiAutos (Sagittarius A* Decentralized Automation).**

— [*]sergio@simplemachines.it — [†]volunteers in Simplemachines Italy —

1. Manifest of the automaton

Each automaton will have a manifest that will be exposed to the host that controls it, inside this manifest there will be all the information relating to the automaton such as the relative URL to which the OTA server is connected and its public key to authenticate the updates .

In a more detailed way, in the manifest we will also find a list of the permissions of the automaton, in fact the automatons will not have direct access to the low-level APIs of the operating system but as explained above everything will happen through IPC. Thanks to this we will also have an additional level of protection in addition to the possibility of implementing a firewall to prevent / allow the call of certain methods.

The list of permits will be organized by activity, in fact each activity of the automaton can be satisfied with certain actions. If an activity has not declared permission to access the filesystem, it will not work for that. In fact, by iterating the action to access it an error code will be returned instead.

Among the other information contained in the manifest we will have the scheme of public services accessible from other hosts or automata to any other automaton. The scheme is defined in the following subsection.

1.1. **Definition of Methods and Types.**

Each automaton will have a public and a standardized private scheme in which the various methods (i.e. functions), that can be called by other automata and services, will be defined. The scheme will have by definition some primitive types that must be implemented by each implementation, they will allow to create all the other types.

Among them we will have:

```
int, bool, biginteger, array, float, double, char (utf-8 / ascii), object.
```

In the scheme:

```
<name>:  <id>  <argument_name>:  <type>...  = <supertype>;
```

Code Example:

```
status:  0 # on:  bool latest_error_timestamp:  int = object;
```

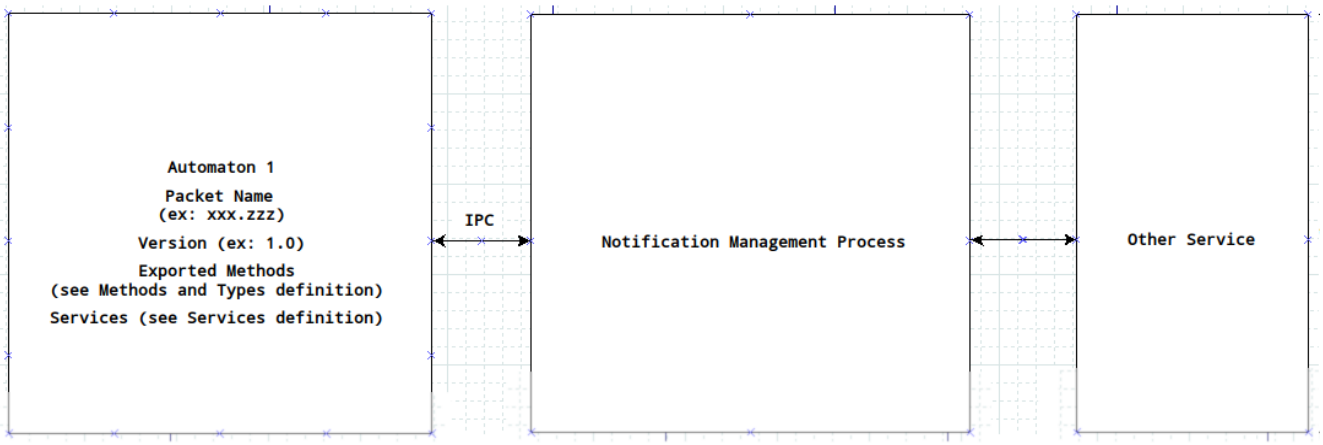

Figure 1. Internals Automaton

2. RPC and IPC

The automata will communicate with the other automata using the list of methods exposed by a service, each service will have a unique local and remote id. The local one is a unique progressive number (the first automaton will have number 0), while the remote one is a complex type which contains the information to communicate with it and the service id.

The address of the remote automaton will be an abstract type that will be correctly implemented by any network access implementation, both at the physical level and at the network level, addressing will be implementation-dependent. Some examples are: ipv4, ipv6 and cdbus, they will be used transparently by the HAL to access the network, the automaton instead must work independently from the addressing system used.

Ipc will have a main process where all the services of the automaton will connect, then through a polling system the data will be received and forwarded between the various services of the automaton.

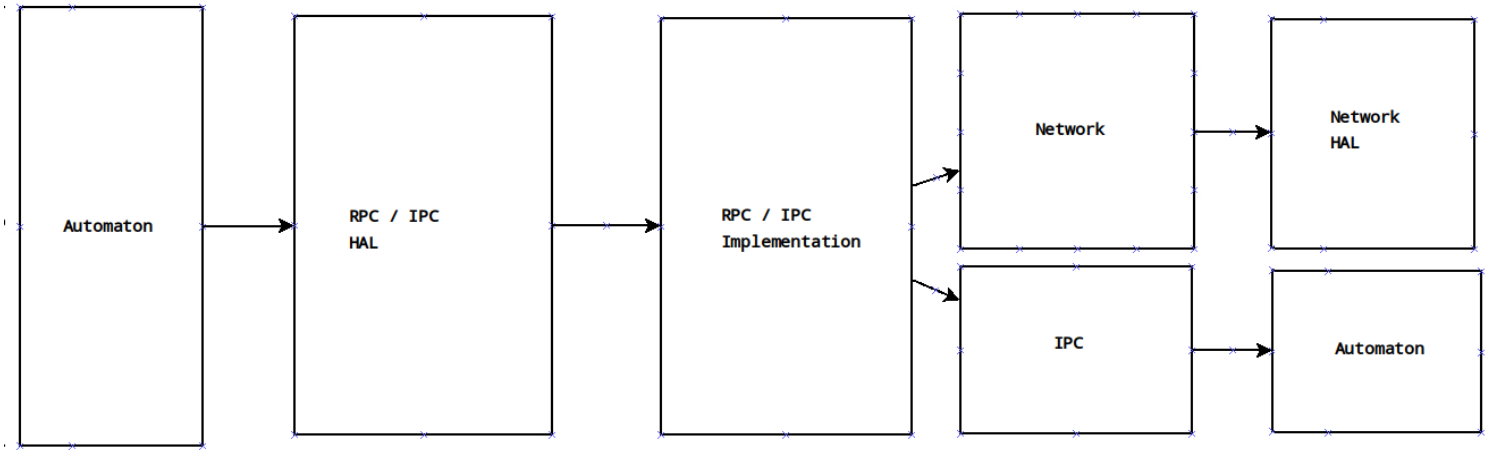

Figure 2. RPC — IPC

3. Event Loop

The event loop can be implemented in two different ways, the first is to have one event loop per process and it will be accessible by multiple threads, the second implementation instead will be per thread.

With the implementation by process we will have the ability to manage multiple blocking tasks that synchronize between them without occupying the thread even while waiting for synchronization

With the implementation per thread, on the other hand, we will have the possibility to create tasks that manage network resources without creating multiple threads.

Asynchronous Synchronization. Exploiting the event loop that allows to manage concurrency in an asynchronous way, we can implement non-blocking queues, sefamori and asynchronous locks, which will not always be possible in fact for realtime applications of these primitives for concurrency it will not be possible, since the event loop if clogged it could increase the delay of the unlock of the primitives, but instead it will be possible to use it for all those tasks that do not need an immediate reactivity.

Asynchronous Network Tasks. To make the most of the MCU scheduler The tasks that use the network will be made asynchronous which will allow to temporarily free the thread and assign it to a new job until all the information is received from the network, to do this an event must be implemented. asynchronous loop and various kernel interfaces for network devices.

4. Polling

The operating system will expose standard APIs that can be used by all the entire automata and services of the OS for synchronization operations between multiple processes.

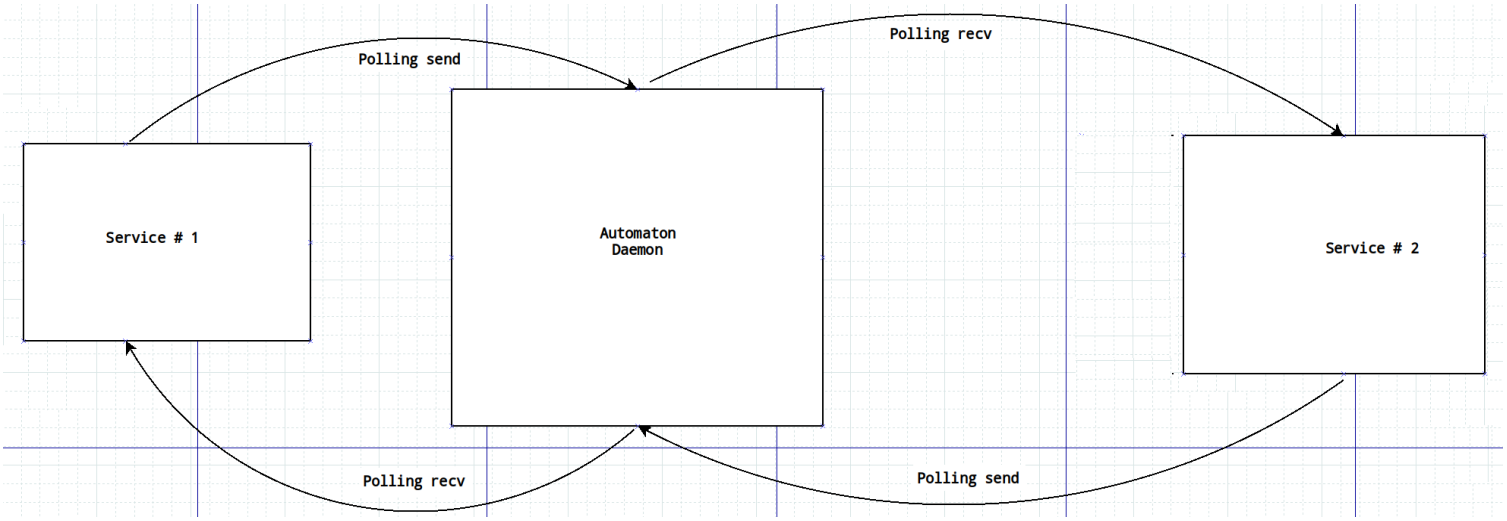

Figure 3. Polling

It will allow to have a polling system without overhead for the process thanks to the use of a locking system, and the os thanks to a green threading system will allow to temporarily de-allocate that thread from the poll and come to be used by another process if necessary.

5. Symmetric Cryptography

We will use symmetric cryptography after exchanging the secret key in a secure channel (see following asymmetric cryptography), the key will be temporary and will not have to be stored in a physical medium, however the resuming of a session already established but dead due to problems is allowed connection between 2 hosts that have not been restarted, in fact if a host loses the connection with another host temporarily for a certain period of time the other host will keep the session to allow you to quickly reconnect with it without having to re-perform the key exchange procedure

5.1. **Asymmetric Cryptography.** We will use an asymmetric key cryptographic algorithm to exchange a symmetric key in an insecure channel.

We will not use asymmetric cryptography consistently for performance problems, in fact it will not be used continuously but only for the purpose of key exchange

6. Secure Storage

All nodes in the rede will use a standard partitioning layout to allow data recovery in critical situations, the operating system and configuration files will be in a standard filesystem across all OSes and will not have to be written at runtime unless necessary.

The filesystem containing the data of the automaton can be changed as needed, importantly, each automaton must allow in a standard and documented way the export and re-import of data as well on a new instance of it without creating problems of inconsistency and inconsistency.

The backups will be encrypted using a public key shared with all the host automata, the private key must not be saved for any reason in the host where the automaton runs for security reasons, in fact the backups can also be shared on the network to the node master to have a redundant backup, without compromising their confidentiality.

7. Cache Management and Energy Blackout

The servers will be powered with redundancy but in any case in case of failure of the main energy source an emergency ups can be used, the ups will be able to display bees on the network to see the energy statistics and the remaining efficiency, in case of entry in ups action since we are in an unstable environment all write caches are temporarily disabled to avoid having data unsynchronized on the disk

8. Remote Update

The automata must be equipped with an OTA (over the air) update system that will allow the same to be remotely updated.

For security reasons, updates must be signed with an asymmetric key, all automata will have the public key that will be checked after downloading the update.

The automata for performance reasons will not download the update directly but it will be entrusted to the gateway node, it will have the power to decide how much to install the update, in fact a time must be determined in which the update does not compromise usability. of the automaton, then calculate the time when the automaton is not used on average and notify the user that an update can be installed at that time.

If you do not want to compromise the uptime of the automaton, you can opt for the temporary migration of the automaton to another node in the network in order to have the automaton always available.

A "reduced" mode of the automaton may also be defined which will allow the user to use the basic functionality of the automaton, because perhaps the host to which it migrates is not sufficiently performing to be able to support this additional automaton.

9. Reinventing the RS-485 in 2020

There is no need to archive what works well, even if a technology has been around for decades. But fill the gaps in a simple way if there are any. This is how we include an approach to use the old RS-485 at the hardware level by translating the differential voltage signal in optical. "+ and -" in voltage become "on / off" over two fibers and then we have a differential line in optics technology with old tranceiver and minimal support electronics (sub 1$ interface including copper to optical connectors). That would allow greater cable lengths or real 20 megabit speeds even over medium

distances, not anymore few meters at full speed but higher lenghts. Therefore, in the common use of copper or optical fiber, we consequently set ourselves the problem of giving a reliable protocol to this copper or optic RS-485 network. Our choice then went to CDBUS.

9.1. **CDBUS over RS-485.** CDBUS offers RS485 peer-to-peer communication, which is very simple, and solves the most common and most headache problem that RS485 has been facing for a long time.

Flexible:

Support multi-host, concurrent read and write, multicast, broadcast, free reporting and etc;

Hot-swappable, for example, arm end tools can be replaced automatically.

Simultaneous reading and writing: The host continuously issues commands to multiple devices, and then continuously receives responses, avoid that the delay of the command is amplified by N times the number of devices in traditional RS485;

High synchronization: Synchronizes the operation of each node by broadcasting at the speed of light (There is no forwarding delay like EtherCAT and it avoids the complicated synchronization problems).

High real-time performance: Because of the guarantee of priority, you can even use the rest of your bandwidth to listen to music and watch videos while in industrial control;

Support multiple hosts: Host failure can be seamlessly switched to standby host, without delaying production or even avoiding accidents.

Main applications: Petroleum Exploration, Coal Mining, Robotics, Robotic Arms, Autopilot, Medical Instruments, Industrial Automation, Smart Home ...

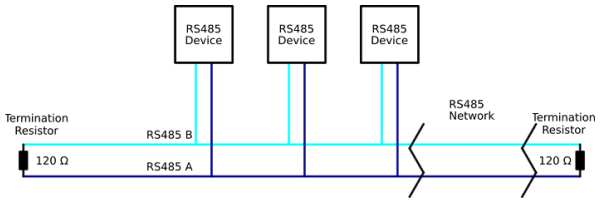

Figure 4. RS-485

10. A Wireless Technology is Needed

10.1. **Bluetooth Mesh.** is a computer mesh networking standard based on Bluetooth Low Energy that allows for many-to-many communication over Bluetooth radio. The Bluetooth Mesh specifications were defined in the Mesh Profile[1] and Mesh Model[2] specifications by the Bluetooth Special

Interest Group (Bluetooth SIG). Bluetooth Mesh was conceived in 2014[3] and adopted on July 13, 2017.

Our efforts goes in the direction to combinate this Mesh technology with **Bluetooth Long Range**.

10.2. **Bluetooth Long Range - 1Km to 5Km.** Bluetooth Low Energy was designed to provide considerably reduced power consumption and cost while maintaining communication ranges similar to Bluetooth Classic.

However, that is no longer the case. With Bluetooth Version 5.0, a new "long-range" mode was introduced. You can now achieve ranges of over 1 kilometer using Bluetooth Low Energy! Long-range mode is not only useful for extending the range of a Bluetooth connection or discovery of advertisements, but it also helps in achieving more robust communication in noisy RF environments and in areas with many obstacles. Examples of applications include:

Remote control and remote identification system for drones [1]
Monitoring sensors deployed on large-area farms
Making connections more robust in areas with many obstacles such as in industrial environments.

Appendix A. Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque id massa vulputate, tristique mi id, imperdiet mi. Mauris id ante ac lacus mollis sagittis. Sed imperdiet nibh id eros malesuada, at fermentum urna mollis. Sed id elit eu arcu varius tempor tincidunt in orci.