A Domotic Ecosystem Driven by a Networked Intelligence

Luca Ferrari, Matteo Gioia, Gian Luca Galliani and Bruno Apolloni
Department of Computer Science, University of Milano, Milan, Italy

Keywords: Advanced Domotic, Internet of Things, MQTT Protocols, Learning Algorithms.

Abstract: We describe a diffuse control system for household appliances rooted in an Internet of Thing network empowered by a cognitive system. The key idea is that these appliances constitute an ecosystem populated by a plenty of devices with common features, yet called to satisfy in an almost repetitive way needs that may be very diversified, depending on the user preferences. This calls for a network putting them in connection and a cognitive system that is capable to interpret the user requests and translate them into instructions to be transmitted to the appliances. This in turn requires a proper architecture and efficient protocols for connecting the appliances to the network, as well as robust algorithms that concretely challenge cognitive and connectionist theories to produce the instructions ruling the appliances. We discuss both aspects from a design perspective and exhibit a mockup where connections and algorithms are implemented.

1 INTRODUCTION

Control theory, as an offspring of cybernetics, is deeply rooted in the concept of feedback (Wiener, 1948) since the forties of the previous century. In the same period of time a different notion of control in terms of self adapting systems emerged with the first modern hypotheses of the brain computing facilities within the connectionist framework (Rosenblatt, 1958). These two ways of controlling a dynamic system ran in parallel up until the nineties, when hybrid control systems began exploiting synergies by conventional controls and by either recurrent neural networks (Ku and Lee, 1995) or reinforcement learning algorithms (Sutton and Barto, 1998). The Internet of Things paradigm suggests a renewed synergy between the two approaches. Namely, the network connecting things on the Web enjoys many properties of a neural network in terms of both connectivity and elementariness of the messages normally exchanged between the devices and their huge number. However, rather than distributed as in the connectionist paradigm, computations are expected to be more efficient if they are performed in a centralized way, yet exploiting the capillarity of the information the network may bring them – hence we call it *networked intelligence*. *Per se*, the machine where computations are done is immaterial, so that we may assume them to be carried out (and possibly distributed too) everywhere in the cloud. However, as with the multilayer perceptron (Haykin, 1994), the information management is dealt with better through a hierarchical architecture than through a distributed one.

This is the idea we pursue in our ecosystem. Namely, we are devising a system constituted by an ensemble of household appliances ruled by a social network which is from time to time committed by a user to operate them optimally with respect to a given task. For instance, the user asks the network to have trousers perfectly washed by his washing machine. In reply, the network sends directly to the machine a sequence of instructions, call it *recipe*, such as “charge water, heat water to 35 degrees”, etc., which drive the machine to carry out a perfect washing. On the one hand this a typical procedure we are used to expect from our personal devices. On the other one, the way of implementing the procedure is rather hard. In a extreme synthesis we need:

1. electronics allowing the network to communicate with the machine, possibly overriding its microcontroller logic;
2. a logic able to produce recipes that are optimal in respect to many criteria, from user preferences all the way to ecological goals such as water or electricity saving. The logic must learn how to reach these objectives from the feedback coming both from devices and from users. Hence it is a cognitive system.
3. an architecture and protocols vehiculating signals between devices and the networked intelligence in a safe and efficient way.

The good news is that the proposed ecosystem has some appeal, so that it got funded by the European Community with a horizon of 30 months (project Social&Smart *(SandS)*-http://www.sands-project.eu/). In this paper we report some of the project's progress as to both the architecture-and-protocols and logics. In addition, we describe an early mockup where instruction and signal dispatchings are enabled by proper circuits. Specifically, in Section 2 we introduce the architecture as the backbone of the project. In Section 3 we briefly discuss protocols, while in Section 4 we show the mockup in detail. In the last section we conclude the paper by saying where we are now and what we expect at the end of the project.

2 SandS ARCHITECTURE

Diffuse control may be viewed as an evolution of WEB 2.0 in terms of a social network whose goal is the dispatching of (optimally controlled) activities rather than the providing of information services. This fits well both with the paradigm of Internet of Things, as for architecture, and with the modern reword expectation from the interaction between members within an evolved society. In this new framework a member is spared from doing boring (because repetitive) activities and is enabled to enjoy better ways of life thanks to the automatic contribution of other members. The supporting infrastructure is a community of personal appliances realized through their connection in Internet. Let us explore these aspects in depth.

2.1 Social Network

Social networks can be seen as a repository of information and knowledge that can be queried when needed to solve problems or to learn procedures. The following definition has been proposed by Vannoy and Palvia [1] in the study of social models for technology adoption of social computing: "an ensemble of intra-group social and business actions practiced through group consensus, group cooperation, and group authority, where such actions are made possible through the mediation of information technologies, and where group interaction causes members to conform and influences others to join the group". We intend to update this definition through the novel idea of building social computing systems where part of the computations is hidden to the social network player, thus representing a form of subconscious computing.

Namely, in our social network we will distinguish between conscious and subconscious computing. The former is defined by the decisions and actions performed by the players (i.e. the users) on the basis of the information provided by the social service. The latter is realized by the data processing performed automatically and autonomously by the web service in order to search for or produce the information offered to the social players, or to fulfill other purposes, such as data mining for the advertising industry. It is an intelligent subconscious computing when new solutions to new or old problems are generated on demand.

We instantiate this new social network in the SandS project in the realm of household appliances and domestic services. The term *eahouker* – meaning easy household worker – is introduced in this context to denote the household appliance user empowered by the social network and social intelligence. In an extreme synthesis the project deals with a social network aimed at producing recipes with tools of computational intelligence, to be dispatched to household appliances grouped in the homes through a domestic wifi network. A *recipe* is a set of scheduled, possibly conditional, instructions (hence a sequence of parameters such as water temperature or soak duration) which completely define the running of an appliance. They are managed by a home middleware – called domestic infrastructure (DI) – in order to be properly transmitted to the appliance through suitable protocols. The entire contrivance is devised to optimally carry out ordinary housekeeping tasks through a proper function of house appliances with a minimal intervention on the part of the user. Feedbacks are sent by users and appliances themselves to the network intelligence to close the permanent recipe optimization loop, with offline tips and advices on the part of the appliance manufacturers. An electronic board will interface each single appliance to the DI (see Fig. 1 (Apolloni et al., 2013))

2.2 The Architecture

This architecture has head in the cloud and feet on the appliances. The general scheme is the following (See Fig. 2):

- user and appliances located in a house;
- both of them are interfaced to Internet through a home router: the latter as machines endowed with some transmission device, the former comfortably sitting on a recliner, sending short orders to DI from time to time;
- many houses refer to the same web-services provider. The connection of the router to the

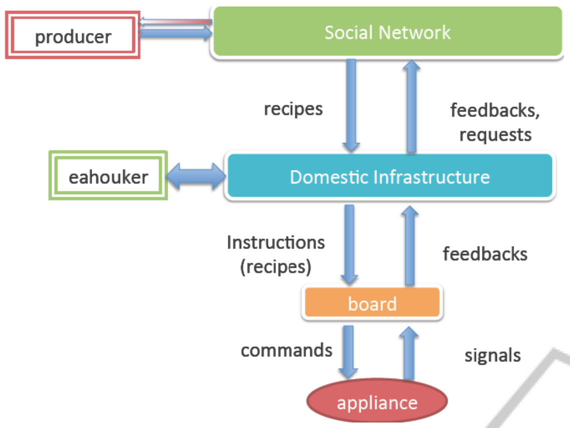


Figure 1: A SandS project synopsis. *eahouker* is a combination of the words easierly and houseworker.

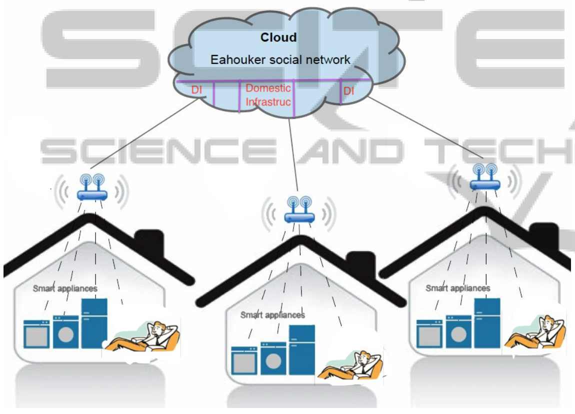


Figure 2: SandS domotic system.

provider is per usual, the internal networking of the communication is committed to a special protocol and a concentrator board if necessary;

- services are delivered in the frame of a social network.

With respect to this consolidated scenario, the peculiarities of our system are:

1. We look for a system as uninvasive as possible. This entails that extra-hardware must be reduced to the minimum; hence: the appliances to be as usual supplied by the stores plus the interfacing chip (Arduino board in the provisional solution we propose); no need for having a computer on, no set-top box.
2. We look for a system as undemanding as possible. This calls for plug&play procedures when an appliance is installed in or removed from the home. Analogously, the need for user input of data must be kept to a minimum; likewise the basic commands for activating the appliances. The way of inputting data is immaterial; for instance by smartphone, tablet, pc or other specific gadget.
3. we look for extremely adaptive services/recipes. This passes through two functional blocks: a social network which issues recipes on demand to the single user and a DI which administers the recipe according to rules on a home-by-home basis.
4. We look for a system that is transparent to the user. Hence, by definition, the DI is on the cloud, as is the social network. Transparency binds the entire chain from user to social network and back.
5. We look for an intelligent system. Here the intelligence of social network members is involved first when a service is initialized and then regularly when the user sends fuzzy feedbacks. While this is insufficient for creating a true intelligent system, we can fill up this task with a series of algorithms on a proper eahouker database (EDB) that constitute the Networked Intelligence (NI) in the core of the social network that is assigned to issuing recipes.

2.3 A Functional Layout

Fig. 3 gives an intuitive representation of the interactions between the system elements (Grana et al., 2013). The SandS Social Network mediates the interaction between a population of users (the eahoukers – hence ESN the name of the social network), each one with his/her own set of appliances. ESN has a repository of tasks that have been posed by the eahoukers and a repository of recipes for the use of appliances. These two repositories are related by a map between (to and from) tasks and recipes. This map does not need to be one-to-one. Blue dashed arrows correspond to the path followed by the eahouker queries, which are used to interrogate the database of known/solved tasks. If the task is already known, then the corresponding recipe can be returned to the eahouker appliance (solid black arrows). The eahouker can express his/her satisfaction with the results (dashed blue arrows). When the queried task is unknown and unsolved then the social network will request a solution from the SandS Networked Intelligence that will consists in a new recipe deduced from past knowledge stored in the recipe repository. This new solution will be generated by intelligent system reasoning. The repository of recipes solving specific tasks can be loaded by the eahoukers while commenting among themselves on specific cases, or by the appliance manufacturing companies as an example of use to foster sales by offering customer additional appealing services. These situations correspond to the

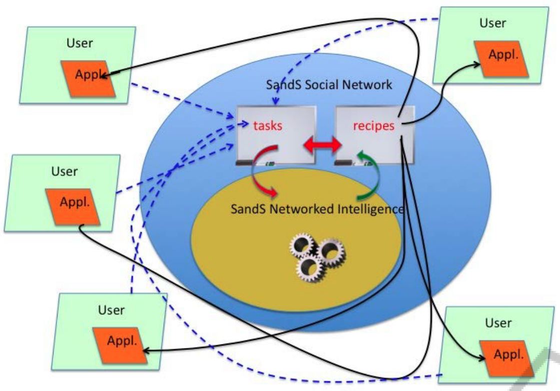


Figure 3: Social and Smart system functional layout.

conscious computing done on the social web service by human agents. The role of the Networked Intelligence is to provide the subconscious computing that generates innovation without eahouker involvement.

2.4 A Conceptual Map of a SandS Session

Figure 4 contains the conceptual map graph (Graña et al., 2013). The main elements are highlighted in red: the eahouker and the appliance, in fact all SandS, is designed to mediate between them. Green boxes contain explicitly active computational modules such as the natural language processing module, the task and recipe managers, the networked intelligence and the domestic middleware. The blue circle highlights the instrumental key of the system: the appliance recipe. The magenta box denotes a hidden reinforcement learning module which the eahouker is not aware of. The SandS session is started by the eahouker stating a task in natural language. The natural language processing module analyzes this expression obtaining a task description which is suitable for a formal search in databases. The task manager explores a task database looking for the best match to the proposed task. If there is an exact match life will be easy for the recipe manager which needs only to retrieve the corresponding recipe. In general, the task manager will select a collection of best matching task descriptions to be presented (or not) to the eahouker to assess the accuracy of the interpretation of his/her intentions by the system. The eahouker may agree to the best matching task descriptions. The recipe manager reads them and continues to explore the recipe database looking for best matches, or proceeds to ask the networked intelligence for the enrichment of the recipe database with new solutions that may better fulfill the task posed by the user. The recipe manager produces a selection of recipes and a best matching

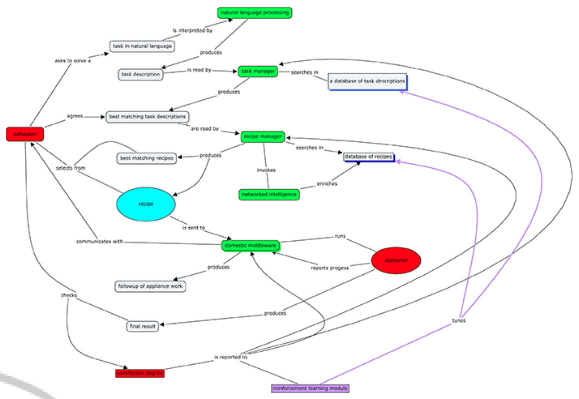


Figure 4: Description of a SandS session by a conceptual map.

recipe. For the engaged user, the selection of recipes may allow him/her to either ponder them and influence the recipe choice or simply trust on the first manager selection by default. This may even be an additional source of feedback to the networked intelligence.

When the recipe is selected, it is downloaded to the appliance via the domestic middleware, which controls its execution. This includes any communication with the user to operate the appliance (i.e. opening the appliance door). The domestic middleware produces a monitoring followup of appliance function that may be shown to the user to keep him/her informed of progress, expected time to completion, etc. The appliance produces a final result, which is returned to the eahouker. Then the eahouker expresses his/her satisfaction, which is the main feedback for all processes.

3 THE TELECOMMUNICATION INFRASTRUCTURE

Appliances and DI are permanently conneced in order to send each other status information and commands. A persistent connection is the best solution for an event oriented project like this. Events can be triggered by either DI side or by the appliance side and information can be sent immediately using the already established connection.

Information is secured by cryptographic functions and encapsulated in MQTT (http://mqtt.org/) frames at application layer. The connections, ciphering and MQTT encapsulation are managed by a connection manager module on the DI as shown on the figure 5.

Communication between DI and appliances should be encrypted to prevent that an attacker can get information or control the appliance. The wireless

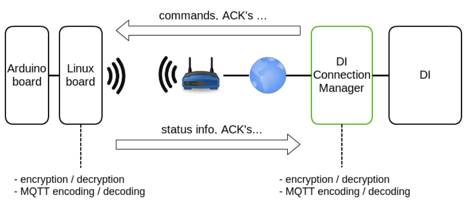

commands, ACK's ...
Arduino board
Linux board
DI Connection Manager
DI
status info, ACK's...
- encryption / decryption
- MQTT encoding / decoding
- encryption / decryption
- MQTT encoding / decoding

Figure 5: Information exchange between DI and appliance.

network of the user would be secured with a WPA2 system, and so an attack from someone not authenticated on the network will be prevented. However, a possible attack could come from someone who has illegitimately accessed and authenticated on the network. To prevent this and also to prevent possible attacks on the channel between the users router and the DI an additional security layer is intorduced (see Fig. 6) by adding a Transport Layer Security (TLS–http://www.ietf.org/rfc/rfc2246.txt) to the communication loop.

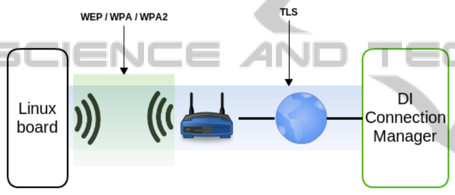


Figure 6: TLS and Wifi encryption.

TLS also allows the use of a certificate to confirm that DI is really DI (and not an attacker supplanting the DI). Even encrypting the communication an attacker could just resend some previously recorded frames in order to send orders to the appliance.

4 AN ENABLING MOCKUP

We have set up a first mockup in the Computer Science Department labs of University of Milano. From a purely operational perspective, it consists of two - three white goods wifi connected to Internet, each with its own Arduino board, in order to execute recipes and send back status signals. For the moment only a washing machine (*wm*) is being tested (see Fig. 7). We use an Arduino MEGA ADK board, an Arduino WIFI SHIELD (see Fig. 8) and a Sitecom wifi router connected to the university network. On this hardware we implement a communication protocol solving, albeit in a preliminary way, problems both of security and of exact message addressing from middleware to appliance and back, within a MQTT-like service (MQTTstandard,) that is sketched in a next section. The protocol is event

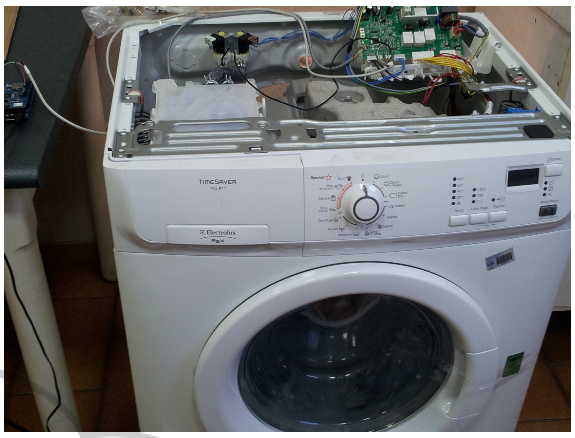


Figure 7: A early SandS mockup.

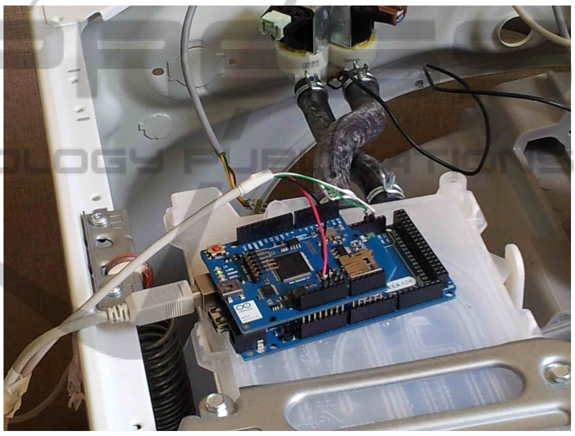
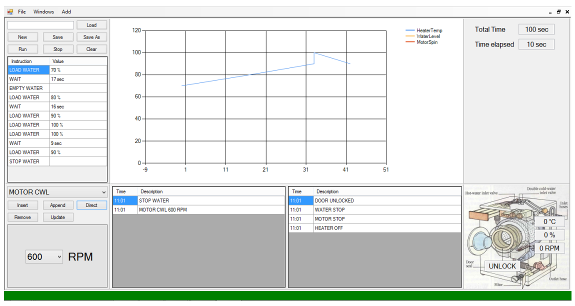


Figure 8: The Arduino interface.

driven, so messages are sent only if state variables change value. The appliance detection is realized in plug&play mode. The management of the appliance, in terms of specification and recipe requests, is done via a browser on any mobile or fixed device. The appliance is located on the ground floor of the Computer Science Department, while the consolle is on the third floor of the same building. A distance of around 300 m is covered by ethernet wiring, while the last 10 m (including two concrete walls) are gapped via wifi connection. A sketch of the parameters managed by the browser is reported in fig. 9. In the following we propose early considerations on the various parts of the mockup.

4.1 Computational Requirements to the Interfacing Board

Quite simply, we ask to the Arduino board to act as a transducer: in input an instruction to the wm actuators, in output the corresponding signal to be trans-



Total Time 100 sec
Time elapsed 10 sec

Instruction	Value
LOAD WATER	70 %
WAIT	17 sec
EMPTY WATER	
LOAD WATER	80 %
WAIT	16 sec
LOAD WATER	90 %
LOAD WATER	100 %
LOAD WATER	100 %
WAIT	9 sec
LOAD WATER	90 %
STOP WATER	

Time	Description
11:01	STOP WATER
11:01	MOTOR CWL 600 RPM

Time	Description
11:01	DOOR UNLOCKED
11:01	WATER STOP
11:01	MOTOR STOP
11:01	HEATER OFF

Figure 9: A screenshot of the washing machine monitor.

mitted to the wm microcontroller. From this perspective, the signals overwrite the native microcontroller firmware with a set of commands that change over time. Thus the microcontroller receives one command at a time consisting of the actuator ID and the value of the parameter to be fixed, where the time is decided in principle by the DI. On the one hand this sort of unitary code (one parameter per actuator) requires common tricks to manage multiparameter actuators. On the other hand, we distinguish two time scales, i.e. a time granularity triggering the shift between scales. The general philosophy is: a sequence of instructions, each lasting more than the time granularity, is dispatched at the proper time by the DI. Vice-versa, sequences of instructions lasting less than the time granularity are encoded into a single instruction that is interpreted and sequenced directly by the Arduino board. This requires a finite automaton to reside in the board, both to carry out the above sequencing and to manage a series of security checks to avoid plant damages and logical inconsistencies. As a matter of fact, even one shot instructions such as “heat the water” requires a set of controls – e.g. the water level to prevent the system from burning out in case of an empty drum – which are managed locally by Arduino. However, as to granularity, this instruction falls in the first category. One example of the second category is the alternating running of the wm motor during the pre-wash phase.

This is the basic situation with regard to normal running. In addition, the board is called upon to supervise an anomalous running at two levels – warning and alarm – to which different standby or shutdown sequences may follow as a consequence of the recognized drawback.

Thus the interfacing board is required for computational power to store the instructions to be decoded locally and to manage local time during the implementation of these instructions. For these purposes, the Arduino capacities (RAM 256 KB, processor ATMEGA2570) prove to be sufficient.

4.2 Communication Requirements to the Interfacing Board

The communication bandwith necessary for the mockup is normally very low. It increases relatively when the appliance is woken-up by a recipe execution request. The communication is stroked by the *keep alive* message sent by Arduino to the DI. Thus, every 5 seconds, the board waits for instructions by the DI. In the current debugging phase the board sends the appliance status (water temperature and level, spin rpm) as a more informative payload. In a greater detail, the message from board to DI is structured as follows, as a further simplification of MQTT scheme:

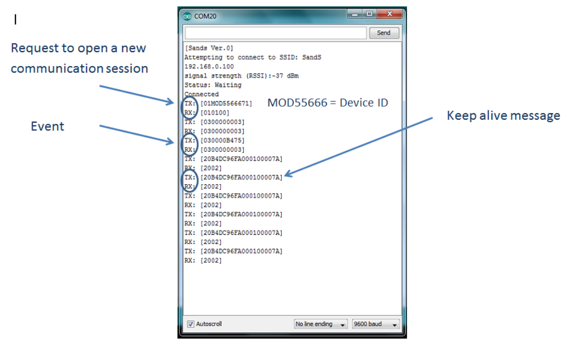
[CMD,EVN,VAN,VALUE,CHK]
[= start of message
CMD = command.
EVN = (progressive) Event number.
VAN = Var number (indexing the boling up variables).
VALUE = Var value (type:Byte, Int, Long o String).
CHK = Checksum23
] = end of message

Conversely, the DI sends instructions to the board. We index each instruction with a sequential number within a recipe ID. This indexing my prove suitable for both debugging and appliance quality control purposes. The message format is analogous, with internal event number mating with the one of the sent event for a correct reckoning of the queues.

[CMD,EVN,VAN,VALUE,CHK]
[= start of message
CMD = command.
EVN = Event number (the same of the message which is the answer to).
VAN = Var number.
VALUE = Var value.
CHK = Checksum
] = end of message

Once the connection is stated between DI and board, it proceeds in full duplex mode. The connection is open and maintained by the appliance through the *keep alive* messages. In Fig. 10 we can see a segment of the conversation between appliance and DI, as collected by the Arduino debugger.

The time granularity of the recipe transmission takes into account various idle times. Besides normal messages related to the recipe execution, both communication addressees may send overriding messages concerning alarm status and various kinds of shutdown/standby commands.



```
[Sands Ver.0]
Attempting to connect to SSID: SandS
192.168.0.100
signal strength (RSSI):-37 dBm
Status: Waiting
Connected
TX: [01MOD5566671]
RX: [010100]
TX: [0300000003]
RX: [0300000003]
TX: [030000B475]
RX: [0300000003]
TX: [20B4DC96FA000100007A]
RX: [2002]
TX: [20B4DC96FA000100007A]
RX: [2002]
TX: [20B4DC96FA000100007A]
RX: [2002]
TX: [20B4DC96FA000100007A]
RX: [2002]
TX: [20B4DC96FA000100007A]
RX: [2002]
TX: [20B4DC96FA000100007A]
RX: [2002]
```

Figure 10: A screenshot of the Arduino debugger.

4.3 Degrees of Freedom in Overwriting the Microcontroller Software

In principle the Arduino Interface is an open-hardware/open software device, so that any person may install it on his/her white good without requiring any license. However, its implementation requires a set of weld junctions and software for overwriting the appliance microcontroller that are specific to the appliance in hand. A further step in the mockup implementation will be to extend the software standardization to a maximum. But overwriting the microcontroller is not an easy task. Rather, we may expect the emergence of small business third parties – for instance, free lance professionals or white good stores with modern selling strategies – which are specialized in this task. For instance, Fig. 11 reports a typical calibration curve relating the water temperature level in the drum and the electrical signal transmitted to the board. Nevertheless, the obvious expectation is that appliances' manufacturers will pursue their own business interests and support this new paradigm of appliance usage by embedding their products with the necessary electronics.

In this early implementation, we played the role of networked intelligence substitute by featuring a few recipes by ourselves. This gave us the opportunity to appreciate the degrees of freedom of this operation and the optimality criteria we may pursue as a counterpart. Namely we jointly considered the following goals: 1) washing efficacy, 2) energy consumption, 3) water consumption and 4) environmental pollution.

5 CONCLUSIONS

While the project is still at an initial stage, we have come to see the high value it offers in terms both of user convenience and of environmental advantage. For instance, questions re the benefit and ecological

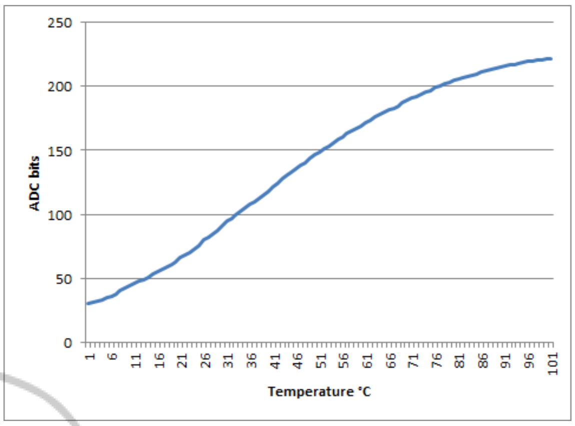


Figure 11: A calibration curve.

profitability of using a bio soap product or reducing the amount of wash water become theoretical opinions no longer. They may be experimented by the single user and many similar users as well within the eahouker social network, so that these questions may find a concretely *scientific* answer.

Though the concrete operations necessary for realizing the mockup delineate a scenario where each single user may implement her/his SandS terminal on home appliances with some technical help, the main way to implement our paradigm passes through a both moral and economical suasion to convince the manufacturers that social appliances are more efficient and rewarding. To achieve this goal SandS project will realize the above discussed architecture in order to set up an initial social network of eahoukers where the benefits of the paradigm will be tossed in concrete by a thousand members. These benefits and the members enjoying them will be the authentic promoters of the SandS ecosystem, again all on the spirit of exploiting actual facts besides sentences.

REFERENCES

Apolloni, B., Fiasch, M., Galliani, G., Zizzo, C., Caridakis, G., Siolas, G., Kollias, S. D., Romay, M. G., Barriento, F., and Jose, S. S. (2013). Social things - the sands instantiation. In *WOWMOM*, pages 1–6. IEEE.

Grana, M., Apolloni, B., Fiasche, M., Galliani, G., Zizzo, C., Caridakis, G., and et al. (2013). Social and smart: towards an instance of subconscious social intelligence. In *1st Workshop on Innovative European Policies and Applied Measures for Developing Smart Cities, 14th EANN*.

Graña, M., Nuñez-Gonzalez, J. D., and Apolloni, B. (2013). A discussion on trust requirements for a social network of eahoukers. In *HAIS*, pages 540–547.

Haykin, S. (1994). *Neural Networks: A Comprehensive*

Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Ku, C.-C. and Lee, K. Y. (1995). Diagonal recurrent neural networks for dynamic systems control. *IEEE Trans. Neural Netw. Learning Syst.*, 6(1):144–156.

MQTTstandard. http://mqtt.org/.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Wiener, N. (1948). *Cybernetics, Or Control and Communication in the Animal and the Machine*. Wiley, New York.