An Extendable Python Library To Manipulate Sensors Coupled To The Raspberry Pi

Edivaldo M. F. de Jesus Jr*
Instituto Federal da Bahia
Rua Emídio dos Santos, S/N
Barbalho, Salvador Bahia
juniorug@gmail.com

Manoel C. M. Neto†
Instituto Federal da Bahia
Rua Emídio dos Santos, S/N
Barbalho, Salvador Bahia
manoelnetom@ifba.edu.br

ABSTRACT

La convergence des technologies de radio, microprocesseurs et appareils numériques personnels conduit à la notion de l'informatique ubiquitaire où les dispositifs intelligents, mobiles et fixes, de coordonner les uns avec les autres pour fournir aux utilisateurs un accès immédiat et universel aux nouveaux services de manière transparente, visant à accroître humaine capacités. Ce travail vise à définir, mettre en œuvre et valider la conception et la mise en œuvre d'une bibliothèque Python extensible pour manipuler des capteurs / actionneurs couplés à la Raspberry Pi en utilisant le module de framboise-GPIO-python. La bibliothèque utilise le modèle de fabrique abstraite pour se assurer que les capteurs / actionneurs et les événements de la même famille étant utilisés en conjonction avec moyen garanti. Sur les autres plateformes, comme Arduino, les API fournissent bibliothèques qui encapsulent la complexité de mise en œuvre et ne offrent que l'interface à utiliser. Ces bibliothèques ne existent pas encore officiellement pour ceux qui veulent utiliser Pyton comme langage de développement appliqué à la Raspberry Pi. Cet article encourage l'utilisation des technologies open source, en raison de la croissance de la libre circulation du matériel, la déduction de la possibilité d'une alternative pour les ingénieurs et les professionnels à développer leurs projets et fournir à la diffusion des connaissances. Le projet présente également les résultats obtenus en utilisant certains des capteurs mis en œuvre, la modélisation du système et les résultats décrits et

Keywords

Ubiquitous Computing,Internet of Things, Raspberry Pi, Sensor, Python

*Aluno do curso de Análise e Desenvolvimento de Sistemas(ADS)

†Doutor em Ciência da Computação e Professor do Curso de Análise e Desenvolvimento de Sistemas

1 Introduction

The UbiComp in its various ramifications and applications, is considered by many as the twenty-first century's new paradigm of computing. It is the area of computing that studies the coupling of the physical world to the world of information and provides an abundance of services and applications, allowing users, machines, data and objects of physical space interact with each other seamlessly. The theme is considered one of the great challenges of research in Computer Science by the National Science Foundation (NSF) [1] and is also present in Computing search Grand Challenges report in Brazil from 2006 to 2016 [2], published by the Brazilian Computer Society (SBC).

Researches on ubiquitous computing are being held on topics such as: basic access to any wireless device, mobility support within the network transparently, safety, context treatment, efficient use of energy, presentation of multimedia content and so on. This work is focused on building intelligent interactive environments. In these environments, the fundamental idea is to create ways to avoid that the user needs to go to the computer/device, allowing many of these working at a distance. The use of platforms to integrate the devices that make up these environments is one of the key points to create it. Currently there are some options to fill this gap. This text emphasizes one: the project Raspberry Pi [3,4].

Raspberry Pi is a widely used platform for professionals who are interested in the field of ubiquitous applications. It provides basic interfaces for creating small projects and / or for those who must be fed by battery. The platform allows you to use high-level programming languages that are quite widespread as C / C ++, Python and Java. The Raspberry allows the development of a range of projects. For example, home automation (turn on and off electrical devices, remote control for TV, Air-conditioners, etc.), eBook and Audiobook readers and so on.

If we seek a way to access the Raspberry Pi bus we shall find a lot of papers in Python. Which incidentally, brings the acronym of the device name (Pi from Python). However, we cannot find any library that abstracts the details of wiring a sensor or actuator to the Raspberry allowing the user takes care just in read the data already processed and/or converted.

The developer must have a degree of experience that can be

considered basic to a computer professional, but advanced for those who are not.
This paper is structured as follow:

- Section 2 present several technologies involved in the preparation of this monograph, aimed to introduce important concepts of Computer area in which the context of the project is inserted
- Section 3 shows correlated works
- Section 4 presents the details of the executed implementation
- Sections 5 and 6 address the testing methodologies used and the results obtained during system validation.
- The last section presents the conclusions.

2 THEORETICAL BACKGROUND

2.1 General Context

The term Ubiquitous Computing was first defined by Mark Weiser [33] in the late 80. At this time, Weiser predicted an increasing in functionality and availability of computing services to end users and, on the other hand, he predicted a decreased visibility of these services. For Weiser, the computing would not be exclusive of a computer. He believed that in the future there would be several different devices connected to each other. At a time when users were using PCs (desktops) and that knowledge needed to operate a computer, Weiser bet on a future where the focus of the users would be the task itself, and not the tool used. In this way, they would use the computer without realizing or require specific technical knowledge. [34]

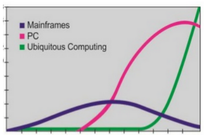

Figure 1: Ages of Computing

The passage of time has shown that betting Weiser was right. For Weiser, [35] evolution of computing has gone through two ages to reach to ubiquitous computing. The first was called the age of mainframe, where many people shared the same computer. The second era was the PC, where each computer was used by one person. Currently, the evolution of distributed information systems, the network connections type options' expansion, mobile computing and the various types of applications on non-conventional computing devices, are just some of the examples that can confirm: the Ubiquitous Computing (the third age) is already a reality. Figure 1 shows a chart with the ages of computing.

Terms such as ubiquitous computing, pervasive computing, nomadic computing, invisible computing, mobile computing and many others, have been used often interchangeably, although they differ conceptually and employing different organization of ideas and management of computer services. Insofar as each area progresses, these concepts will be better understood and its definitions will become clearer. This section presents the key concepts needed to understand the UbiComp besides presenting some project examples in the literature.

2.2 Mobile Computing

Mobile computing is based on the ability of a user to load or move (physically) computer services wherever it moves. In this context, the computer becomes an ever-present device that expands the ability of a user to use the services it offers, regardless of their location. Combined with the ability to access the network, mobile computing has transformed the computing an activity that can be taken almost anywhere.

An important conceptual limitation of mobile computing is that the computational model used in most applications does not change while users are moving. It means that a device is not able to obtain information on the physical context in which computation occurs, and consequently also can not adapt to the new context correctly. A solution to accommodate the changing context would pass to users the responsibility to monitor and manually configure an application / device to the extent that it moves. However, this solution is not well accepted by most users. This limitation was one of the inspirations for pervasive computing .

2.3 Pervasive Computing

The concept of pervasive computing implies that the computer is embedded invisibly in the environment to the user [24]. In this conception, the computer has the ability to: i) obtain environmental information in which it is embedded and ii) use it to build dynamically computational models that allow you to control, configure and tune the application to better suit the needs of a device or user. For this to be possible, the key point is the ability of computers be able to act as "smart" in the environment where users move. This environment is usually populated by computational sensors and services.

2.4 Ubiquitous Computing

As can be seen in Figure 2, the UbiComp can be defined as a computer area positioned between the Mobile Computing and Pervasive Computing [14, 24]. Ubiquitous is an adjective originated from Latin (ubiquu) which means "that is at the same time everywhere". Ubiquitous computing benefits from the advances in mobile and pervasive computing and arises from the need to integrate mobility with the functionality of pervasive computing. The term ubiquitous computing will be used here as a junction of pervasive computing and mobile computing. The justification to perform a distinction of these terms is a device that is embedded in an environment, not necessarily is mobile.

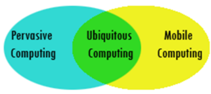

Figure 2: Ubiquitous Computing: intersection between pervasive and mobile computing

Research in Ubiquitous Computing approach about the technologies and infrastructure that enable the deployment of ubiquitous applications through a number of issues including the following:

- how to design hardware and operating systems for sensor platforms?
- how to allow devices to find each other and to use your services?
- how to allow systems involving limited processing resources and energy, to work well?

Generally, ubiquitous applications receive sensor data from other service providers devices, manage user actions, provide support mobility and use context information to perform tasks [6]. A ubiquitous system itself has a set of requirements, peculiarities and challenges that influence the design, implementation, deployment and evaluation of its project. [24] These are cornerstones of UbiComp and quite different from those used in the development of systems for PC's. Among these points, we can cite as an example.:

1. Resource-Constrained Devices;
2. Volatile Execution Environments;
3. Heterogeneous Execution Environments;
4. Fluctuating Usage Environments;
5. Invisible Computing;
6. Security and Privacy;

2.5 Internet of Things

The Internet of Things (IoT) is a multidisciplinary field, covering a wide range of subjects, from purely technical issues (eg, routing protocols, semantic queries) to a mixture of technical and social problems (security, privacy, usability) as well as social and business topics. The existing internet of things' applications are potentially diverse. Monitoring of environmental and personal health, monitoring and control of industrial processes, including agriculture, smart spaces and smart cities are just some examples of the IoT applications [5].

The connection of physical things to the Internet makes it possible to access remote sensor data and to control the physical world from a distance. The mash-up of captured data with data retrieved from other sources, e.g., with data that is contained in the Web, gives rise to new synergistic services that go beyond the services that can be provided by an isolated embedded system. The Internet of Things is based on this vision. A smart object, which is the building block of the Internet of Things, is just another name for an embedded system that is connected to the Internet[H. Kopetz, Real-Time Systems. Design Principles for Distributed Embedded Applications. Kluwer Academic Publications, 1997].

Everyday physical things that are enhanced by a small electronic device to provide local intelligence and connectivity to the cyberspace established by the internet. The small electronic device, a *computational component* that is attached to a *physical thing*, bridges the gap between the physical world and the information world. A *smart object* is thus a *cyber-physical system* or an *embedded system* consisting of a *thing* (the physical entity) and a *component* (the computer) that processes the sensor data and supports a wireless communication link to the Internet[H. Kopetz, Real-Time Systems. Design Principles for Distributed Embedded Applications. Kluwer Academic Publications, 1997].

2.6 Hardware

One of the fundamental requirements for developing ubiquitous systems is the use of hardware such as sensors, microcontrollers, communication devices (network cards, Bluetooth, etc.) and storage, among others. For example, sensors allow transform use of interactive environments from more transparent interfaces. Currently, there are some platforms that allow insert and control various types of sensors, using of communication interfaces and storage units. This section presents and details the main hardware devices available for the development of ubiquitous systems.

2.6.1 Sensors and Actuators

Sensors are devices that allow us to capture information from the environment in which they are inserted, such as temperature, pressure, presence, humidity, smoke detector, light intensity, among others. In general, the sensors work transforming parts of a physical quantity into an electrical signal, which in turn can be interpreted by electronic devices [7]. In other words, sensors are components that allow an electronic device to interact with the real world.

According to [7], when the sensors operate directly, transforming one form of energy into another are called transducers. The sensors where operations occur indirectly alter their properties, such as resistance, capacitance or inductance, under the action of physical grandeur so that this change is roughly proportional. For example, the light sensor LDR (Light-dependent resistors) vary inversely its resistance the amount of light falling on it. Thus, when there is a large amount of light falling on the sensor, they have a very low resistance and this allows the flow of electric current increases, whereas when there is little light, they have a high resistance and prevent current flow.

An actuator as well as a sensor is a transducer that converts one form of energy into another, and can also do the opposite [7]. In other words, rather than just transform parts of a physical quantity into an electrical signal, it can transform an electrical signal into a physical quantity such as motion, magnetism, heat, among others. For example, the relays are electromechanical devices that work with small power, but are able to control external circuits that involve high currents. They are basically composed of a coil and a set of contacts. When a current flows through the coil it creates a magnetic field that attracts and closes the contacts, remaining as long as power supply in the coil. As a result, it allows the passage of energy through the relay.

2.6.2 Arduino

Arduino was created in 2005 by Massimo Banzi and David Mellis in Italy with the goal of use as an electronic learning tool and programming for design students, so that they would use in art projects, interactivity and robotics. Electronic learning was expensive: a microcontroller was costing 100 euros. So they decided to make their own board. Sought employees and thus created an efficient technology, accessible and compatible with Windows, Mac and Linux.

Arduino is a platform that popularizes the concept of free hardware. In the book Getting Started with Arduino, Massimo Banzi describes the Arduino as a physical open-source computing platform based on a simple board with input/output pins that implements the Processing language. It is a small but powerful board constituted by a microcontroller that can be easily programmed via a Universal Serial Bus interface (USB) and able to build electronic devices and interesting systems.

Working with the construction of a hardware requires much time and effort, because is necessary to create new circuits, use various components such as resistors and capacitors and many welds. Using Arduino board abstracts much of this construction process, making it simpler, allowing people from different fields of knowledge being able to build their projects. The platform is widely used worldwide for offering advantages such as:

- A multiplatform environment that can run all major operating systems such as Windows, Linux and MacOS.
- Is an open-source hardware, ie, the circuit design is available so that if someone is interested in creating your own card, just buy the necessary components.
- Hardware cost is low.
- Is possible program it via a USB cable instead of a serial port. Remember that today's computers do not have serial ports.
- It has a development environment with intuitive interface for easy use.

The fact that both the hardware and the Arduino software be developed in an open, patent-free, allows its projects to be recreated in different ways. The adoption of open hardware concept motivates those who create projects to contribute with functions and libraries for the Arduino. Is knowledge about knowledge, the same principle of free software.

Note that there is not necessary advanced knowledge in electronics to use the platform. But for those who are interested in deepening the knowledge in this area, there are several available materials that can help, for example, in [19].

2.6.3 BeagleBone

The BeagleBoard is a low-power open-source hardware single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The BeagleBoard was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip.[7] The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and software capabilities. It is also sold to the public under the Creative Commons share-alike license. The board was designed using Cadence OrCAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used.

BeagleBone is an $89 Manufacturer suggested retail price (MSRP), credit-card-sized Linux computer that connects to the Internet and runs software such as Android 4.0 and Ubuntu. With plenty of I/O and processing power for real-time analysis provided by an AM335x 720MHz ARM® processor, BeagleBone can be complemented with cape plug-in boards to augment functionality.

2.6.4 Raspberry Pi

Raspberry Pi is a small computer, approximately with the size of a credit card. It can be used to do many things that are done by a common personal computer (RASPBERY PI FUNDATION, 2014a). In addition, it can also be used in electronic projects because it has a hardware interface: The general purpose input/output port (GPIO). The Raspberry Pi Foundation, creator of the project, is an educational charity headquartered in the UK and aims to help and encourage the teaching of computer science in schools.

This computer came up with the intention of reconnecting children and youth in computer programming and stimulate the creation of new projects so there is not only the consumption of the technology created by the market. While in the 1990s there was a growth in the number of children and youth who developed programming skills, starting in the 2000s, this number started to decline. (Raspbery PI Foundation, 2014b).

With the realization that, year by year, the students were moving away from programming and reducing the development of skills in computer science. The researchers Eben Upton, Rob Mullins, Jack Lang and Alan Mycrof, from the Computer Laboratory of University of Cambridge had the idea of creating a platform that would allow to reconcile the programming students and handling computers. (DENNIS, 2013) From 2006 to 2008, early versions of what is now the Raspberry Pi was developed. From 2008, with the expansion and emphasis on mobile devices, the devices are becoming more efficient and cheaper which enabled the launch of the Raspberry Pi model B to the public in February 2012 for $ 35.

The proposal of the Raspberry Pi is therefore be a low-cost computer, with the ability to interact with the outside world through the sensor coupling. As seen previously, it was thought to be used in the educational environment, in order to assist and encourage the teaching of programming and help understand the operation of computers. However, the Raspberry Pi has been used by people of all ages and interests in various projects, for example, projects involving automation, sensing and robotics, games, multimedia, networks and servers.

Currently there are three Raspberry Pi models: The model A, which costs $ 25 and the B and B+ models, which costs $ 35. As for the price, there is not much difference between the hardware models. The main differences are the amount of USB ports (1, 2 and 4 in the models A, B and B +, respectively), the Ethernet port (model A does not have) and ram (256MB on the model A against 512 for the others).

2.6.4.1 Features

The original Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC),[1] which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU,[8] and was originally shipped with 256 megabytes of RAM, later upgraded (models B and B+) to 512 MB.[9] The system has Secure Digital (SD) (models A and B) or MicroSD (models A+ and B+) sockets for boot media and persistent storage.[10]

In 2014, the Raspberry Pi Foundation launched the Compute Module, which packages a BCM2835 with 512 MB RAM and an eMMC flash chip into a module for use as a part of embedded systems.[11] The Foundation provides Debian and Arch Linux ARM distributions for download.[12] Tools are available for Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux),[14] C, C++, Java, Perl and Ruby.

Some other features presents on RPI:

- Display Serial Interface Connector (DSI): This connector receives a flat-ribbon cable of 15 pins that can be used to communicate with an LCD or organic light-emitting diode (OLED) display screen;
- Camera Serial Interface(CSI) Connector: This port allows a camera module be directly coupled to the card;
- P2 and P3 connectors: These two rows of connectors are JTAG connectors for test to Broadcom chip (P2) and LAN9512 network (P3). Due to the proprietary nature of Broadcom chipset, these connectors are unlikely to be of much used;
- Pin Protection: Most of the pins in the header go directly to the Broadcom chip. It is important to carefully design the components you attach to them as there is a risk you will permanently damage your Pi. Short circuits and wiring mistakes could also ruin your day so double check everything. A multimeter is probably going to help a lot here as you can double check wiring before you connect to the Pi.

2.6.4.2 Power Supply

The device is powered by 5v micro USB. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you hook up to it. We have found that purchasing a 1.2A (1200mA) power supply from a reputable retailer will provide you with ample power to run your Raspberry Pi for most applications, though you may want to get a 2.5A (2500mA) if you want to use all 4 USB ports on the Model B without using an external powered USB hub.

The power requirements of the Raspberry Pi increase as you make use of the various interfaces on the Raspberry Pi. The GPIO pins can draw 50mA safely (that is 50mA distributed across all the pins! An individual GPIO pin can only safely draw 16mA), the HDMI port uses 50mA, the camera module requires 250mA, and keyboards and mice can take as little as 100mA or over 1000mA! Check the power rating of the devices you plan to connect to the Pi and purchase a power supply accordingly. If you're not sure, buy a powered hub.

2.6.4.3 Processor

The System on a chip(SoC) used in the first generation Raspberry Pi is somewhat equivalent to the chip used in older smartphones (such as iPhone / 3G / 3GS). The Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC),[1] which includes an 700 MHz ARM1176JZF-S processor, VideoCore IV GPU,[8] and RAM. It has a Level 2 cache of 128 KB, used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

2.6.4.4 Inputs and Outputs

The number and options of input and output from a Raspberry Pi depend on the model used. Models RPi A+, B+ and 2B GPIO J8 have 40-pin as pinout. Models A and B have only the first 26 pins.In this paper we will present only the pins availlable in the model B.The pins are divided into:

- Digital pin to input or output (programmable) - 17 pins;
- Analog input pins or digital input / output (programmable) - 6 pins;
- Power pins (gnd, 5V, 3.3V) - 9 pins;

The first item on the list are the useful pins. They are the ones that are available for a programmer use. Is through these pins that the Raspberry Pi is coupled to the sensors to capture environmental information. Among the 17 digital input / output pins there are 2 pins that match the serial communication module UART (Universal asynchronous receiver/transmitter). This module allows communication between a computer (for example) and the Raspberry Pi (see Special Pins section). All pins have more than one function, pins can be input or output, and the roles of each are defined in the code of possible programs that can be run on the RPi.

2.6.4.5 Digital Inputs

All 17 programable pins can be used as digital inputs. When a pin is programmed to function as digital input, you use a command that, when executed, performs the "reading" of the voltage applied to it. Then, after running this command, you can know if the pin is in a "high" or "low" state (on or off).

From the electrical point of view, the program can tell if a pin is fed with 0 (zero) or 5 Volts. This function is usually used to identify whether a button is pressed or a sensor is capturing some environmental information. Note that the digital input function delivers only the values 0 or 1 (no voltage or with voltage). You can not know how much voltage is being applied to the pin.

2.6.4.6 Analog Inputs

The Raspberry Pi does not have analog pins as Arduino, but is possible to use a traditional analog-to-digital converter(ADC) chip connecting to the R-Pi via SPI or I2C. For example the Microchip MCP3424 (I2C interface) which has 4 differential inputs. All ADC should work on the R-Pi, except that the timing is not as good as a microcontroller due to the available Linux versions not being true real-time.

2.6.4.7 Digital outputs

With a digital output is possible only two types of values (0 or 5 volts, 1 or 0, etc.). From a pin programmed as digital output is possible for example light up an LED, connecting a relay, trigger a motor, etc. You can program the RPi for all 17 digital outputs.

2.6.4.8 Special Pins

RPi pins have some special characteristics which can be used from the software functions encoded by a software programmer. Are they:

- PWM - Pulse Width Modulation: Treated as analog output, it is actually a digital output that generates an alternating signal (between 0 and 1) where the time that the pin is in level 1 (on) is controlled. It is used, for example, for engine speed control, or generate voltages with values controlled by the program. For the PWM communication may use pin 12;

- UART - Universal asynchronous receiver/transmitter: One pin(TxD) can be used to transmit and another(RxD) to receive data in serial asynchronous format. For example, you can connect a data transmission module via bluetooth and allow communication with the Arduino remotely. Are reserved for UART pins 15 (RXD receives data) and 14 (TXD sends data);

- SPI Port - Serial Peripheral Interface: These are pins that allow synchronous serial communication faster than UART. They allow for example to connect memory cards (SD) and many other things. Are used for this purpose the pins 19 as Master Output, Slave Inpu(MOSI), 21 as Master Input, Slave Output(MISO), 23 as Serial Clock(SCLK), 24 as Chip Select0(CE0) e 26 as Chip Select1(CE1).

- I2C Bus - Inter-Integrated Circuit: The I2C bus allows multiple devices to be connected to the Raspberry Pi, each with a unique address, that can often be set by changing jumper settings on the module. It is very useful to be able to see which devices are connected to the RPi as a way of making sure everything is working.Are used for this purpose the pins 3 for Serial Data Line(SDA) and pin 5 for Serial Clock Line(SCL)

2.6.4.9 GPIO in Python

The easiest way to control the GPIO pins is using the RPi.GPIO Python library. Installing the library is easy if you follow my RPi.GPIO Installation Guide. Once installed using the pins is as easy as in figure 2:

```python
import RPi.GPIO as GPIO

# Use GPIO numbers not pin numbers
GPIO.setmode(GPIO.BCM)

# set up the GPIO channels - one input and one output
GPIO.setup(7, GPIO.IN)
GPIO.setup(8, GPIO.OUT)

# input from GPIO7
input_value = GPIO.input(7)

# output to GPIO8
GPIO.output(8, True)
```

Figure 3: controlling gpio with Python

2.6.4.10 General Purpose Input/Output

Another important aspect of hardware from Raspberry is the group of GPIO pins. These pins are programmable ports to input and output data used to provide an interface between the board and peripherals, microcontrollers/microprocessors, sensors, actuators, etc. The GPIO interface is fundamental for building intelligent interactive environments. It is the interface between the Raspberry and the real world. In simple terms, you can consider the GPIO pins as switches that can be turned on/off.

Similarly to the Arduino, besides the GPIO, the Raspberry also supports PWM, UART and SPI. Of the 26 pins available, 17 are reserved for GPIO and 8 are used for power and ground. Figure 2.6 shows the GPIO interface. The pins can be used via code written in a programming compatible language like Python, Scratch, Java, among others [8]. The GPIO pins are available on the PCB via a header and allow you to interface the Pi to the real world.

2.6.4.11 GPIO.BOARD and GPIO.BCM

The GPIO.BOARD option specifies that you are referring to the pins by the number of the pin the the plug - i.e the numbers printed on the board (e.g. P1) and in the middle of the diagrams below.

The GPIO.BCM option means that you are referring to the pins by the "Broadcom SOC channel" number, these are the numbers after "GPIO" in the green rectangles around the outside of the below diagrams:

Unfortunately the BCM numbers changed between versions of the Model B, and you'll need to work out which one you have guide here. So it may be safer to use the BOARD numbers if you are going to use more than one pi in a project.

2.6.4.12 WiringPi

WiringPi is a library for access to GPIO interface written in C. Its use can be performed with C, C ++, or other programming languages through wrappers (WIRING PI, 2014). Wrapper is an outer layer that extends WiringPi and can be implemented in different programming languages. This will allow the programmer to carry out projects not only in C or C ++, but also in language implemented by the wrapper. There are wrappers being developed in various languages such as Java, Ruby and Pyton. This last will be used in this paper from the wrapper raspberry-gpio-python.

2.6.4.13 raspberry-gpio-python

3 JUSTIFICATION

As with other platforms, Raspberry Pi allows coupling several sensors whose handling can be made from raspberry-gpio-python or any other API available. On other platforms, such as the Arduino, the APIs provide libraries that encapsulate the complexity of implementation and offer only the interface to use. These libraries do not yet exist formally for those who want to use Python as a development language for Raspberry Pi.

This may be a consequence of run under the Linux kernel which is not suitable for real time applications - it is multitasking O/S and another process may be given priority over the CPU, causing jitter in the program[??].

4 RELATED WORK

5 APPLICATION DEVELOPMENT

5.1 Method/Methodology

5.2 Functional Requirements

5.3 Non-functional Requirements

5.4 Why not use Java

5.5 Architecture

5.6 tests

6 CONCLUSION

7 FUTURE WORK

8 REFERENCES

8.1 Citations

8.2 Tables

8.3 Figures

8.4 Theorem-like Constructs

APPENDIX

A Headings in Appendices

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

A.2.2.1 Inline (In-text) Equations

A.2.2.2 Display Equations

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX *Expert*

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

This section is inserted by LATEX; you do not insert it. You just add the names and information in the `\additionalauthors` command at the start of the document.

A.6 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B More Help for the Hardy

The acm_proc_article-sp document class file itself is chock-full of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LaTeX, you may find reading it useful but please remember not to change it.