2014 1st International Congress on Computer, Electronics, Electrical, and Communication Engineering (ICCEECE2014)
IPCSIT vol. 59 (2014) © (2014) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2014.V59.9

Low-Cost and Resource-Aware Intelligent Device: A Core of Thing

Onur Akdemir [1], Turgay Altilar [2]

[1] Istanbul Technical University

[2] Istanbul Technical University

Abstract. The Internet of things is the concept of Internet-enabling physical objects. These devices with the help of sensors which are built on the devices do some specific tasks which controls environment and send sensor data to a master device. Lack of intelligence of these devices are eliminated with a master device which alerts humans who give decisions or to a powerfull computer which is able to give decisions. This paper describes an intelligent device core that is intelligent and continuous learner which need interaction only on emergency with Center application which thing objects are connected. A fully-autonomous device both energy efficient and intelligent device proposed. It is demonstrated that intelligence is possible on TI Stellearis LM4F120XL platform which can run FreeRTOS, a Learning Algorithm and other helper tasks together. Proposed core can also give several other decisions by supplying appropriate data.

Keywords: Internet of Things; Intelligence; Full-autonomous; Efficient k-means; FreeRTOS; Low power.

1. Introduction

The Internet of Things (IoT), or "embedded Internet" is widely seen as the next logical evolution of the Internet in a proximate fully-interconnected world. The basic idea of this concept is the pervasive presence around us of a variety of Things or Objects - such as RFID tags, sensors, actuators, mobile phones, etc. which, through unique addressing schemes (IPv6), are able to interact with each other and cooperate with their neighbors to reach common goals [1]. Nowadays, the main communication form on the Internet is human-human, but communication forms will expand to human-human, human-thing and thing-thing[2]. Especially, human-thing and thing-thing communication have meaning only with enabled intelligent thing devices. The Things composing the IoT will be characterized by low resources in terms of both computation and energy capacity. Accordingly, the proposed solutions need to pay special attention to resource efficiency besides the obvious scalability problems[1]. Much of the current work in this field is focused on Wireless Sensor Networks and the use of the IPv6 over low-power Wireless Personal Area Networks (6loWPAN) scheme, but there is significantly less research on Intelligence of things. This paper discusses some of the challenges that must be addressed for Intelligent and Low power devices. In particular, it has been discussed that the development of an intelligent, fully autonomous, low power, internet-enabled Forest Fire Thing which can detect fire early and inform the concerned ends(Human or thing). Such a device has the potential to support the development of Intelligent IoT applications for all smartness needs.

2. Motivation And Background

2.1. Motivation

The emerging internet of things devices are expected to be very small sized, energy efficient and powerfull enough to run targeted applications. Reduced power consumption prolongs the battery life time of embedded systems,and other systems running on battery[3]. This is especially important on certain embedded devices which need expensive recharging/replacement of batteries. It can be costly, or even impossible to access the device when the battery runs out. For some devices the life time is over when the battery is exhausted. Examples of this can be motion sensors embedded into the concrete of buildings, medical equipment(implants), equipment on satellites. Portable energy sources such as kinetic energy, or

solar panels produce little power and are expensive. Low power allows the use of smaller, lower cost solar panels. Reducing the power consumption also reduces the need for larger solar panels which is cheap and small in size. Other reasons to choose right microprocessor is that weight important. Some devices like small sized quadcopters' weight is only 10-15 gram which makes impossible to use high power microprocessors as mission computer. High power microprocessors which targeted embedded devices can run Linux OS, libraries and algorithm that does the job for this device. Running Linux and some libraries restricts the choise on microprocessor that developers must use at least 400-500 mhz microprocessor which can run a kmeans algorithm with the help of Linux and weka library. At first, rapid development seems reasonable but when energy consumption is considered, it is obvious that these devices coul not be the future's Internet of Thing devices. Using low power microprocessor for an Internet of Things device hardens to create an intellingent device. At first even it seems impossible to implement a device which is both energy efficient and intelligent enough for future IoT devices, choosing right platform and using efficient algorithm allows to build a core that would be a basis for these concept. Our platfrom only runs at 50 mhz on Cortex-M4 mcu, FreeRTOS and a kmeans algorithm. This study also shows that there is extra power to run other tasks on this core which allows future improvements.

2.2. The Internet of Things (IoT)

The IoT is the concept of networking real-world objects and is regarded as the next logical generation of the Internet. It is predicted that hundred billion devices to be connected in near future, so research on both hardware and connection methods are so popular[1]. These connected devices need some behaviour to act as useful ends. Many challenges must be faced to make this devices intelligent. Low resources and limited energy that is supplied by small capacity batteries contradict with off-the-shelf learning algorithms. Algorithms for IoT must be implemented without vasting of any resource. IoT cover all digital devices that can connect to internet and interact with humans. This communication can be between a human and IoT or IoT-IoT. IoT enables collective intelligence which eliminates human factor for decision making on multiple sensors. IoT devices give final decision and just result is sent to humans. Collective intelligence is not possible without intelligent IoT. It is obvious that management of this devices could be impossible for humans in near future when IPv6 enabled. Some intelligence can help both for management effort and automating the system.

2.3. Intelligent Software

Artifial Intelligence software is an active research for several decades. Very useful libraries that implemented several AI and pattern recognition algorithms have widespread usage[9]. Weka, matlab toolbox and other small scale libraries exist. Main disadvantage of these libraries is that they are too big for a device which has low-resources. Besides this, resource-aware intelligent software development must be done carefully to allow space for other tasks. Intelligent algorithms are based on the prior knowledge on specific topics. Starting with a known data on a special topic and further create new knowledge or extract knowledge from this data can be implemented with pattern recognition techniques. It is obvious that making machines as smart as humans not possible. So simulating human behaviour is the only way to think like a human. Alan Turing's proposal that the question " Can machine think?" can be replaced with the question "Can machines do what we (as thinking entities) can do?"[10]. This idea lead us to learning algorithms which starts with a prior knowledge and predicts some rules from new observations with the help of this prior knowledge. Learning from scratch for machines need special efforts. An IoT without prior knowledge has to built some base knowledge with the help of observations that are collected from sensors. Some trust level must be determined until that IoT could make trusted decisions. It maybe impossible for some knowledge to find a sutiable trust level , some decisions must be done on this subject. Beacuse of open issues exist about learning from scratch, in this paper this method ignored.

2.4. FreeRTOS

FreeRTOS is a free and open source real time operating system designed to have small footprint and targeted to embedded systems [8]. It is written in C language and does not contain any driver model. Also support for complex memory management is not possible for this small kernel. Scheduling algorithm is

simple round-robin with priorities, also co-operative, preemptive or hybrid scheduling is configurable. Both tasks and coroutines are supported, but in this work only tasks are considered. Tasks can be blocked for a specified time is usually used for creating periodic tasks. If tasks are blocked indefinitely, they are called suspended. Tasks can be unsuspended by calling an appropriate function from the interrupt service routine or by some other tasks.FreeRTOS keeps the track of time by counting periodically generated interrupts as ticks. In the official FreeRTOS ports for microcontrollers based on the Cortex-M4, the Systick timer is used as a tick source. Each time the Systick interrupt routine executes, an internal FreeRTOS variable called xTickCount is incremented and a check is performed whether any delayed task has to be deblocked. All internal time-based calculations, such as task delay time, depend on the xTickCount value. For the correct functioning of FreeRTOS, no Systick interrupt should be neglected, because deadlines of some tasks could be missed. The Cortex-M4 has a powerful possibility of disabling all interrupts that are below or equal to a priority determined by the BASEPRI register. Those interrupts are called low priority interrupts. Interrupts whose priority is greater than the BASEPRI register value are called high priority interrupts. As FreeRTOS uses this mechanism for disabling interrupts, they are never all disabled. By the system's-kernel function, only low priority interrupts can be disabled. In the interrupt service routines of high priority interrupts, the usage of the kernel system functions is not allowed. Therefore, the kernel function for disabling interrupts is used for keeping critical sections safe from other tasks preemption. This mechanism's advantage is that handling high priority interrupts is never delayed by the kernel system code.

2.5. TI Stellaris Driver Library

TI Stellaris series microcontrollers have drivers for peripherals on ROM [7]. This allows efficient use of limited resources, without using any flash and sram resources. While they are not drivers in the pure operating system sense (that is, they do not have a common interface and do not connect into a global device driver infrastructure), they do provide a mechanism that makes it easy to use the device's peripherals. FreeRTOS does not force any driver infrastructure so using this driverlib essential.These drivers written entirely in C except where absolutely not possible. All functions are reasonably efficient in terms of memory and processor usage. For many applications, the drivers can be used as is. But in some cases , the drivers have to be enhanced or rewritten in order to meet the functionality, memory, or processing requirements of the application. If so, the existing driver can be used as a reference on how to operate the peripheral. The peripheral driver library provides support for two programming models: the direct register access model and the software driver model. Each model can be used independently, or combined, based on the needs of the application or the programming environment desired by the developer. Each programming model has advantages and disadvantages. Use of the direct register access model will generally result in smaller and more efficient code than using the software driver model. However, the direct register access model does require detailed knowledge of the operation of each register, bit field, their interactions, and any sequencing required for proper operation of the peripheral; the developer is insulated from these details by the software driver model, generally requiring less time to develop applications.

3. System design and implementation

3.1. System Hardware Structure

FILID core designed in this paper consist of TI LM4F120XL series MCU (based on ARM Cortex-M4 Core) and corresponding peripheral modules. LM4F120XL is the 4th generation Stellaris MCU designed by TI. Peripheral modules connection avaliable by boosterpacks. There are plenty of boosterpack boards that allows new designs with LM4F120XL[5]. Real-world FILID things must have some kind of connection with others. Wi-fi, bluetooth, Zigbee connection types are available with boosterpacks. It is known that Wi-fi and other connection types' energy consumption are high, solutions to this problem is an active research called 6loWPAN. Low-energy usage goal is only available by creating custom hardware using the previously mentioned LM4F120XL mcu. Howerer, the outcome of this project is targeted towards showing intelligent thing devices with very limited resources, new hardware not proposed. Therefore, it was decided that the software development for the project would be carried out on the already available ek-lm4f120xl development board. The use of this development board also aided in much of the implementation and testing

phase as both JTAG and the Universal Asynchronous Recieve/Transmitter (UART) were made available over USB.

3.2. System Software Design

LM4F120XL development board was chosen for use with the project. This MCU comes with great libraries and toolchains called Texas Instrument Stellearis Peripheral Driver Library, Driverlib. The use of Driverlib greatly reduces development time and learning curve when attempting to control the peripherals of an unfamiliar architecture. Another advantage of using Driverlib is that a common API is made available for the entire family of Stellearis MCU. This will allow for the easy porting of any software written to other microcontrollers in the family, even what would otherwise be low-level, hardware-specific code. FreeRTOS was chosen as it is most of the source code is in C language, it is readable, portable and easily expandable and maintainable. Features like ease of use and understandability makes it very popular. More than 77,500 official downloads in 2009, and the survey result performed by professional engineers in 2010 puts the FreeRTOS at the top for the question "which kernel are you considering using this year" showing its increasing popularity. Intelligence part of this design is real heart of this project. Clustering was chosen for decisions which are not certain, that needs some intuitive approach. K-means is a rather simple but well known unsupervised learning algorithm that implements the clustering[4]. Simply put, k-means is a procedure that n observations partition into k groups and each observation only belongs to the group of data points which are nearest to the cluster center (centroids). Each of the generated clusters is compact inside its group, and independent of each other. High memory usage of this algorithm comes with a cost that makes to run this algorithm impossible on Cortex series MCUs. It is obvious that SRAM capacity is low which prohibits to use kmeans algorithm. Flash ram which is bigger in capacity than SRAM is used to save constant data. New observations also added to Flash ram. This design allow extra space on SRAM that allows our core to run other several tasks[6][7]. Forest fire prior knowledge, consist of 517 observations, which reside in flash memory so SRAM is available only for computations. Forest fire dataset classes divided two distinct decisions, fire exists or not. Decision variety can be incremented if dataset allows which makes thing devices give decisions like humans.

Model of learning that is inference from newly added observations highly resemble a master-apprentice teaching. Here FILID is apprentice who is to be trained by a master, for our solution center application. Center solution waits new decisions from FILID and feedback with the result of decision. With the help of this decision FILID can expand its own knowledge and get extra smartness. Communication with FILID and Center only needed for sending decisions. High energy consumption of Wi-Fi reduced with this design.

4. Learning Protocol

Proposed solution uses k-means algorithm which is a machine learning technique. Machine learning, a branch of articial intelligence, concerns the construction and study of systems that can learn from data.

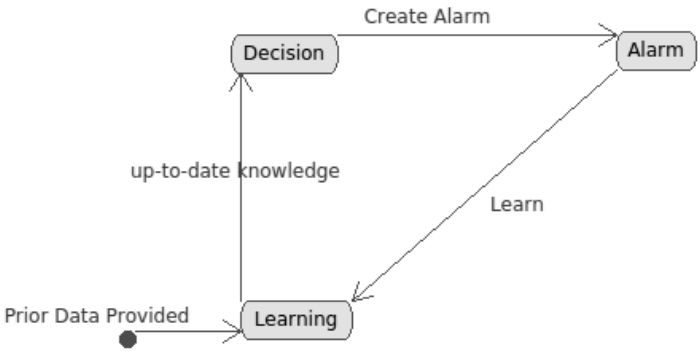

Fig. 1. Flow of alarm status and learning.

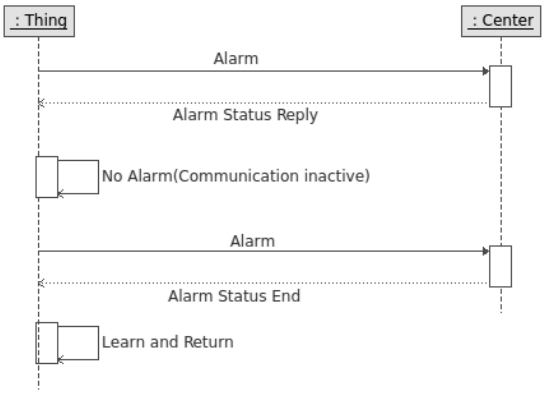

Fig. 2. Function calls between Thing and Center.

More formal definition: "A computer program is said to learn from expericence E with respect to some class of tasks T and performans measure P, if its performance at tasks in T, as measured by P, improves with experience E" [9]. Our model applies for situations where decisions limited to discrete number of status. Yes/No like situations with apropriate data provided prior to run, easily predicted by FILID. Learning evolves with the help of Center which could be run by a Human. Therefore, human intuition factor somehow added to FILID. FILID runs on predefined intervals and executes decision task. This task normally just checks data which come from sensors on that FILID. Decision task works without any intervention from another human or any other computer. Fully automated decision's results send to Center only if there exists an alarm situation. Communication needs lowered by this design so energy consumption. Self-living FILID core decisions based on prior data. Prior data and sensors identify the new FILID thing and task.

Alarm status that is sent to Center needs some feedback for continous learning. A human whose job is to monitor Center, feedback to alarms so learning evolves. Same alarm stuation may sent to Center more than one which is not an error but feedback for the alarm must be provided at least one by Center. With the help of feedback result and decision that created the alarm situation, new observation added to knowledge base. After that learning phase followed by normal run which listens environment and checks environment to give new decisions.

5. Experimental Evaluation

In order to show FILID concept , an early fire detection thing is proposed. Early fire detection expecially targeted to forests. Human can do this job very well but machines which educated by a human expert also does the job. A device is implemented on LM4F120XL platform which runs FreeRTOS as operating system. Also an efficient version of K-means algorithm developed. Learning method which explained in this paper also applied. Our learning protocol works between a thing and human operator. Human operator checks Center application for an emergency alert which may be initiated by a thing. Interaction between human and thing starts on first alert. The thing observes some measurements and predicts some behaviour with the help of prior knowledge. The thing sends alert if a new emergency status observed. The Center accepts alert and show information on UI. A Human whose job is to monitor Center application analysis the situation and feedback thing device about observation. The thing accepts observation and adds new knowledge to knowledge base with the decision of human operator. If human operator does not feedback in ten minutes interval, same decision would be sent to Center application. This will not affect the system, because learning is just performed when feedback returns from Center Application.

Other than core tasks, some helper tasks implemented to show that there exist extra space for future developments. Performans of the system seems very reasonable and works periodically as expected. FreeRTOS uses 30000 bytes of ram for heap usage, our solution uses only 12000 bytes of this heap space. This also shows extra power exists for more powerfull algorithms. Although, k-means like intelligence is enough for forest fire application, with this extra computing power some more degree of intelligence could be implemented for other kind of thing objects.

Power consumption is another strength of this design. Our solution uses little power which can be supplied by batteries which are long-life. Other solutions those are very populer nowadays consume more power so they are not sutiable for future internet of things.

Processor	***Energy Consumption (Avg)***	***Cost***
Rasperry pi B (ARM11)	500mA	35$
Arndale Exynos (Cortex A15)	750mA	259$
Ek-LM4F120XL (Cortex-m4)	100mA	5$

TABLE I. POWER CONSUMPTION AND COST COMPARISION

6. Conclusion

Future devices seem to have limited cpu power, low capacity storage and expected to be long-life, so new designs must be proposed. In this paper , it is shown that machine can learn new observations from experiments on low power devices. Human interaction is needed for this devices, but in near future machine teaching to machine concept seems possible. Some improvements on communication technology, and embedded programming techniques also required. Smart watches, intelligent home systems, intelligent shopping machines are some examples of our future.

7. References

[1] Luigi Atzori, Antonio Iera, Giacomo Morabito (2010). “The Internet of Things: A survey”, Computer Networks 54 (2010) 2787-2805

[2] Lu Tan, Neng Wang (2010). “Future Internet: The Internet of Things”. 3rd International Conference on Advanced Computer Theory And Engineering(ICACTE),2010.

[3] Adirana Wilde, Richard Oliver ve Ed Zaluska (2013). “Developing a Low-cost General-purpose Device for The Internet of Things”. In, The 7th International Conference on Sensing Technology (ICST 2013), Wellington, NZ, 03-05 Dec 2013.

[4] Rahul Shah, Shonali Krishnaswamy and Mohamed Medhat Gaber (2005). “Resource-Aware Very Fast K-Means for Ubiquitous Data Stream Mining”. In: Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams, held in conjunction with ECML PKDD 2005, 3-5 October 2005, Porto, Portugal.

[5] ARM. “Cortex-M4 devices generic user guide,” ARM Limited, 2010

[6] Texas Instruments: Stellaris LM4F120XL Evaluation Kit User’s Manual, Jan. 2010.

[7] Texas Instruments: Stellaris® Peripheral Driver Library Manual, Jan. 2010.

[8] FreeRtos Official Guide

[9] Mitchell, T. (1997). Machine Learning, McGraw Hill. ISBN 0-07-042307-7, p.2.

[10] Harnad, Stevan (2008), "The Annotation Game: On Turing (1950) on Computing, Machinery, and Intelligence",in Epstein, Robert; Peters, Grace, The Turing Test Sourcebook: Philophical and Methodological Issues in the Quest for the Thinking Computer, Kluwer.

Onur Akdemir recieved the BS degress from the department of Computer Engineering of Egean University,Turkey. He is currently working as a software engineer at Tubitak,Turkey as researcher. His research interest include real-time systems, low power computing and embedded systems.

Turgay Altilar recieved the BS and MS degrees from the department of Computer Science of Istanbul Technical University,Turkey. He recieved the PhD degree from QMW Department of Computer Science. He is currently an assistant professor in the Computer Engineering Department, Istanbul Technical University,Turkey. His research interest include paralel and distributed systems, grid computing, real-time systems, genetic algorithms.