IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 16, Issue 5, Ser. I (Sep. – Oct. 2021), PP 09-14
www.iosrjournals.org

A Review on Design and Development of Communication Protocols for Embedded System

Mhamane Sanjeev Chandrashekhar[1], Dr. Amol Kumbhare [2]
[1]*(Research Scholar, Electronics and Communication Engg. Dr.APJ Abdul Kalam University,Indore ,India)*
[2]*(Associate Professor, Electronics and Communication Engg. Dr.APJ Abdul Kalam University,Indore ,India)*

Abstract: *A communications protocol illustrates the rules for sending blocks of information [each referred to as a Protocol Data Unit (PDU)] from one node in a network to another node. Protocols are generally defined in a layered mode and supply all or possibly part of the services specified by a level of the OSI guide model. A protocol specification explains the operation of the process and could also propose the technique the protocol requires to be realized. In this paper, we discuss that a Methodology for the Design and Implementation of Communication Protocols for Embedded Wireless Systems. Communication protocol design involves for complementary domains: specification, verification, performance estimation, and implementation. Usually, these technologies are treated as separate, unrelated stages of the design: formal specification, formal verification, and implementation, in particular, are seldom approached from an integrated systems point of view. This paper presents a design methodology that employs a blend of formal and informal mappings to process a high-level specification into an implementation.*

Key Word: *- Protocol data, Embedded system, WPAN, WSN, Infopad, SDL, SMV*

Date of Submission: 06-09-2021 Date of Acceptance: 22-09-2021

I. Introduction

Data communications protocols administrate the mode, in which electronic systems swap over information with specifying a set of rules that, when followed, give a reliable, repeatable, and well-understood data transfer service. In designing communication protocols and the systems that implement them, one would like to guarantee that the protocol is accurate and competent. Accurateness means that the rules of exchange are internally consistent and unambiguously handle all possible events. Informally, we wish to know that the protocol is free from unwanted behavior, such as deadlock, and that it can indefinitely provide data transfer service under any input sequence. These correctness properties are only part of the design problem: it is equally important to guarantee that the protocol is efficient. Efficiency, used here to indicate how well a given protocol performs relative to an implementation with unconstrained complexity, is a much more difficult property to quantify. The measures of efficiency are largely dependent upon the context in which the protocol is to be used and upon the services that the protocol is supposed to provide. Throughput, delay, channel utilization, spectral efficiency, and end-to-end distortion are but a few of the measures commonly used to compare alternatives in protocol design. In practice, most new protocol designs are approached in an ad-hoc fashion that relies heavily on simulation to answer both the question of correctness and efficiency. Formal approaches such as formal specification and formal verification are usually relegated to the domain of theorists. This paper, in contrast, addresses the problem of integrating formal techniques within a comprehensive design flow.

The context for the protocol design methodology is link-level communication protocols for wireless networks that provide multimedia services to mobile users, such as the one described in example system. In particular, infrastructure-based networks that support mobile clients are considered; challenges specific to peer-to-peer communication between mobile hosts are not addressed). Portable devices, in this context, have severe constraints on the size, the power consumption, and the communications bandwidth available, and are required to handle many classes of data transfer service over a limited-bandwidth wireless connection, including delay sensitive, real-time traffic such as speech and video. This combination of limited bandwidth, high error rates, and delay-sensitive data requires tight integration of all subsystems in the device, including aggressive optimization of the communication protocols to suit the intended application. The protocols must be robust in the presence of errors; they must be able to differentiate between classes of data, giving each class the exact service, it requires; and they must have an implementation suitable for low-power portable electronic devices. In contrast to ap- application creation in general, what makes protocol design particularly for embedded systems? While applications with similar features as protocols can be found, there are many explanations why protocol design is treated as a special case. The behavior of a protocol is often governed by timer end, e.g. by limiting the time for a communication partner's response. The following segment will deal directly with the topic of real-time demands.

In addition, dynamic relationships may be interlaced between one or more protocol positions in contact partners. Compared to time dependence, this complexity creates a large number of potential protocol states and protocol runs. In any case, it is to be ensured that the protocol instance does not end up in a stalemate and that no live locks are reached again from each state reached. The robustness and correctness needed are particularly important in the context of embedded systems, which must run reliably without maintenance for months or years.

Embedded systems, such as car window openers, have replaced a lot of non-calculation systems over the last decades and have become all-round. During the mass production of digital hardware components their extensive acceptance has become economically feasible. Since embedded systems often belong to another product which can be sold in millions, they have to be extremely cheap. The price of an integrated circuit depends on the size of its die; the cheaper it is the lower the area a chip consumes. Embedded systems therefore have only a fraction of the computing and storage capabilities which a modern PC has incorporated for economic reasons. The need for low energy consumption is another significant feature for many embedded systems applications. As high clock and large chip size lead to more dynamic and static power dissipation, embedded device developers have to develop systems with the smallest clock frequency and the smaller chip size possible to perform the necessary tasks. There are emerging wireless personal area networks (WPAN) and wireless sensor networks (WSN). Domestic and safety automation, personal medical care, logistics, traffic monitoring, process and automation in manufacturing, agriculture are but not limited to applications.

Communication protocols describe interconnection rules for the endpoints of communication [15]. Protocols are thus the foundation for the implementation, in particular, of computer networks and wireless sensor networks. You may use it in hardware, software or as a combination of the two. The method of implementation affects effectiveness and other parameters, including the versatility of introducing subsequent protocol extensions or bug reparations. Protocols are however built on an abstract level without any particular method of implementation. In contrast to application creation in general, what makes protocol design particularly for embedded systems? While applications with similar features as protocols can be found, there are many explanations why protocol design is treated as a special case. The behavior of a protocol is often governed by timer end, e.g. by limiting the time for a communication partner's response. The following segment will deal directly with the topic of real-time demands. This paper is presented in five sections including the introduction. Section II gives a brief overview of literature. Section III presents Comparison of recent work carried, section IV describes a brief overview of implementation of communication protocol, and section V gives the conclusion.

II. Literature Review

Manoj Devare[1] in his paper presented the manufacturing IoT marching towards the digital twin as well as the wide spectrum of uses require the specific lower energy protocols for communication as well as information transfer. This offers an extensive discussion on protocols, platforms, use cases, opportunities, and the challenges for the deployment of energy protocols that are low of the context of IoT apps. Additionally, discussion extends to the different custom methods for energy saving in the communication of receptors to hardware, hardware to Cloud, and deferred information clicking in advantage computing. The conventional wireless data transfer and communication protocols are actually ideal in case of the hardware platforms associated with seamless power cord. Nevertheless, there's need of low energy protocols 6LoWPAN, LoRaWAN, Sub one, ZigBee, BLE, NFC, Neul, LTE M, NB IoT, Sigfox, Ingenu, Z Wave, EnOcean,Thread, Other telecommunication protocols and ec-gsm throughout many IoT uses. The SBCs as well as micro controllers are not necessarily built with these protocol enabled hardware. **Daryl Narakadan, et. al [2]** In this paper they used a design which is actually based on FPGA. Using FPGA instead of microcontroller has its many benefits such as for instance very low energy consumption, higher computation speed as well as much better effectiveness. Additionally for the wireless medium we research on different moderate and choose by far the most compatible one. **Stefan Marksteiner,et. al [3]** This particular paper presents an overview of IoT program domains and also discusses probably the most crucial wireless IoT protocols for smart house, which are actually Thread, Z-Wave, Zigbee, EnOcean, and KNX-RF. Lastly, it describes the security features of said protocols and compares them with one another, offering tips on whose protocols are more ideal for a protected smart home. **WojciechStaszewski, et. al [4] -** presented a latest development in the area of communication coming from the perspective of embedded machine condition monitoring system (CMS). Even though the common goal of CMS remains the same over many years, different electric innovations open brand new choices of enhancement, including lower price tag, smaller size, bigger bandwidth, smaller power usage or even larger distance for wireless transmission. **ShadiSarawi, et. al [5] -** This particular paper is going to be an effort to discuss various communication protocols in IoT. Internet of Things (IoT) consists of smart devices which speak with one another. It makes it possible for these units to gather as well as exchange information. Also, IoT has currently a broad range of living uses including business, transportation, logistics, healthcare, intelligent atmosphere and private, social gaming robot, and community info. Smart products are able to have wired or perhaps wireless connection. So far as the wireless IoT is actually the primary concern, a number of wireless communication technologies as well as protocols may be utilized to link the smart device like Internet Protocol Version six (IPv6), over Low power Wireless Personal Area Networks (6LoWPAN), ZigBee, Bluetooth Low Energy (BLE), Z Wave and Near Field Communication (NFC). They're short range regular network protocols, while SigFox and Cellular are actually Low Power Wide Area Network (LPWAN).standard protocols. **K. Iniewski [6]** In his paper has covered the substantial embedded computing technologies highlighting the uses of theirs in communication that is wireless as well as computing power. An embedded system is a pc system created for certain command operates inside a bigger system often with real time computing constraints. It's embedded as a part of an entire unit typically like mechanical components as well as hardware. Offered in 3 parts, Embedded Systems: Implementation, Design, and Hardware provides readers with an immersive introduction to this quickly growing part of the computer system business. To acknowledge the point that embedded programs control a lot of today's the majority of common products like smart phones, PC tablets, and also hardware embedded in automobiles, TVs, as well as refrigerators as well as heating systems, the guide begins with a standard introduction to embedded computing systems. It hones in on system-on-a-chip (SoC), multiprocessor system-on-chip (MPSoC), and network-on-chip (NoC). It then covers on chip integration of software program as well as customized hardware accelerators, in addition to fabric flexibility, custom architectures, as well as the numerous I/O requirements which facilitate PCB integration. Then, it concentrates on the technologies related to embedded computing systems, going with the fundamentals of field programmable gate array (FPGA), digital signal processing (DSP) and application specific integrated circuit (ASIC) technology, architectural assistance for on chip integration of custom accelerators with processors, along with O/S assistance for these methods. Lastly, it provides complete information on computer-aided design, testability, and architecture (CAD) assistance for embedded programs, smooth processors, heterogeneous online resources, and on chip storage before concluding with coverage of software support in specific, O/S Linux. Embedded Systems: Hardware, Design, and Implementation is actually a great guide for style engineers looking to enhance as well as minimize the size as well as price of embedded system products and improve their performance and reliability. **Sachin S Gade, et. al [7]** developed a protocol the liability of this particular protocol is actually lies in its 2 rules that are various for information transmission as well as reception. The great thing about this particular protocol is it decreases the code dimensions of embedded program and most crucial is no data loss or perhaps collision. This technique will help both half and full duplex mode. **P.H.W.Leong [8]** worked on methods based on FPGAs (Field Programmable Gate Arrays) offer numerous benefits over traditional implementations, including an FPGA to a style offers the flexibility to place a little performance in software and several in hardware. The range of this particular task is having wireless communication between embedded methods using FPGAs for management as well as information transmission demands. The communication must occur at range that is moderate along with average data rates.

III. Comparison of recent work carried

Table no 1: Shows the comparison of recent work carried.

Sr. No.	**Author**	**Description**
1	**Manoj Devare**	This particular chapter deals with the appropriate hardware as well as mixture with power choices that are low as per the spending budget, range, and specs.
2	**Daryl Narakadan, et. al**	The project is going to establish communication using UDP packets, which would facilitate wireless transfer of information of strong Wi-Fi engineering.
3	**Stefan Marksteiner,et. al**	This particular paper presents an overview of IoT program domains and also discusses probably the most crucial wireless IoT protocols for smart house, which are actually Thread, Z-Wave, Zigbee, EnOcean, and KNX-RF. Lastly, it describes the security features of said protocols and compares them with one another, offering tips on whose protocols are more ideal for a protected smart home
4	**Wojciech Staszewski, et. al**	The paper considers an overall notion of an embedded monitoring system, which consists of a sensor, processing unit and data acquisition, storage peripherals, along with a communication module for ultimate details destination. For every one of these defined components, a comprehensive explanation of potential details transfer protocols and interfaces is provided, which includes common, as well as experimental ones.
5	**Shadi Sarawi, et. al**	This particular paper is going to be an effort to discuss various communication protocols in IoT. Additionally, it is going to compare between generally IoT communication protocols, with a focus on the primary functions as well as actions of different metrics of

		power use security spreading information fee, along with other functions.

IV. A brief overview of implementation of communication protocol

The goals of designing embedded communication systems are to design reliable, accurate protocols, implement them, and incorporate them into the overall system. An integrated design approach can help all stages of the design process while still allowing for rapid growth.

SDL (Specification and Description Language) has grown-up in recognition as a tool for developing communication protocols. Telelogic TAU SDL Suite [9] is a development tool that permits you to plan, simulate, check, implement in software, and test protocols. To improve the design flow, wide research has been undertaken, for instance defining real-time constraints, designing more efficient implementation models, or generating hardware implementations. In the next part, we'll talk about this project.

In the literature, numerous protocol implementations based on formal specifications and employing SDL have been defined. Other methods to design that aren't based on SDL are also feasible. System C [10] has a theoretically simple refinement technique for deriving hardware and software implementations, Simulink [Mat07] has an integrated simulation environment with other protocol layers and physical channel models, and Unified Modeling Language (UML) is a general modeling practice. The high abstraction level and implementation-independence of SDL specifications, as well as its formal semantics that permit protocol verification, are the advantages of an SDL-based design flow.

System Design with SDL

The language has been employed in the standardization of communication protocols, like the IEEE standards 802.15.1 and 802.15.4, since its capability to formally define data types, such as protocol data units, and actions in an implementation-independent manner.

Synthesis of implementation We will present research on the transformation of abstract SDL specifications into implementations in concrete execution environments in the following sections. During this transformation progression the semantic similarity of both the specification and the implementation model must be preserved, or else all preceding theoretical examinations would become outdated. The fact that abstract SDL models have infinite queues and limitless memory space, conversely, contradicts physical reality and demonstrates that there are constraints when moving from a theoretical model to a practical implementation. [11] suggested an extension of the SDL semantics to permit for the utilize of bounded input ports.

SDL specifications can be realized as mixed hardware/software systems in common. A huge number of optimization methods have appeared as an outcome of comprehensive research into the subject of producing efficient implementations. Software deployment generation is now cutting-edge, and has been employed in commercial SDL tools for a few years. A numerous research groups researched automated translation into hardware depictions, i.e. VHDL code, in the 1990s and beyond. There were tools for creating prototypical hardware implementations that, to our information, did not make it into successful commercial products. Aside from code generation, embedding the SDL system into a run-time environment or operating system, as well as the design of hardware-software interfaces, is crucial. The so-called server model is an easy implementation model for SDL specifications that protects the language's semantics. Each SDL process is comprehended by an asynchronous server, which is a separate entity with its own signal queue that processes input signals one at a time and interacts with other processes by putting asynchronous signals in their input queues. This model is appropriate to both software and hardware implementations. In the first example, all servers run simultaneously under the running or run-time system's scheduling regime, whereas in the second case, real hardware parallelism can be developed. The Telelogic TAU code generator for software implementations and the COSMOS tool [12] for VHDL generation both employ the server model. The supplementary scheduling and context-switch overhead, in addition to the essential resources to handle signal queues, are drawbacks of this approach.

The operation thread model is an algorithm that removes the overhead connected with the server model. Communication among SDL processes is synchronous in this model, i.e. via procedure calls. Instead of sending a message to another process, the receiver process's corresponding transformation is named. There is no need for a context transfer or signal buffers because the execution stays in the same thread. All probable chains of transitions that are occurred by an external event and end with a transition without signal output or a signal to the environment must be identified in the SDL specification.

The operation thread model can be employed efficiently realize simple protocol stacks in which signals are swapped only from higher to lower layers as transmitting or in the opposite direction after a packet is received. More complex interaction patterns within the model may add overhead since transformations that are part of several operation threads, i.e. are performed in response to different external events, will be multiplied unless serialization is implemented [13] This refers to a hardware implementation. In order to preserve the model's semantics, special care must be taken when there are interdependencies among processes or multiple signal outputs within a change in software implementations employing the operation thread model [14]. [15] introduced such a dependency analysis and proposed optimizations to progress parallelism within an implementation.

Reduced the number of copy operations of large buffers, such as protocol data units, and a method called application-level framing have also enhanced performance. The first method makes use of quickly transferred references to buffers. Application-level framing is a protocol data unit handling strategy that works through protocol layers. Buffer space wide enough to handle lower-layer headers is already reserved at the highest protocol layer, the application layer. All layers simply add their control data to the appropriate location in the buffer. Each layer also accesses the related part of the frame buffer in the receive direction.

Since the 1990s, researchers have been focusing on deriving hardware implementations from SDL specifications. Owing to a lack of notions to express synchronous actions and data types for bit manipulation operations, attempts to use SDL as a definition language for synchronous digital systems, similar to how SystemC can be used today, have failed [16]. The work of Muth [17], who also contributed to the CORSAIR framework, will be highlighted as one of the many approaches to producing hardware implementations from SDL specifications. The [17] rapid prototyping design flow begins with an initial SDL specification and a collection of timing annotations that express the system's real-time constraints. The aim is to create a compatible hardware/software implementation on a rapid prototyping platform that comprises several high-performance general-purpose processors and a configurable I/O processor (CIOP). The CIOP is made up of programmable hardware (Xilinx FPGA) and is connected to the platform's other processors via a PCI bus.

The event stream model is employed in a real-time study of the SDL specification to model the potential occurrence of external events and timers. It is possible to set deadlines for the processing of events. During the real-time investigation, In addition, the worst-case message queue depth is estimated. This is essential in order to maintain the semantics of the original SDL model and efficiently distribute resources.

The target architecture's processing units are manually mapped to SDL processes. [17] The partitioning is based on their "timing criteria and computational complexity." Telelogic's C Advanced code generator is used for the software. The server model is used to build implementations for this C code generator. All of the required support functions, such as timers, inter-process communication, and interfacing with the environment, are implicit in the SDL specification and must be provided by a run-time device. The run-time framework is realized in the mentioned approach by the free real-time operating system RTEMS. Different implementation strategies for VHDL generation from SDL can be chosen, including the server model as well as serialized and parallel operation thread implementation. The process-specific extended finite state machine is modeled in VHDL for the server model, and pre-designed components for signal queues and output ports are added.

The chain of transitions originating from an initial external event is calculated in the case of an operation thread implementation. Every operation thread is implemented as a VHDL process on its own. Locks are inserted to protect shared variables from concurrent access in a parallel implementation, or activity threads are serialized, since multiple activity threads can access the same shared variables of a method. As previously stated, an execution environment for SDL specifications must provide facilities and mechanisms that are part of the language's semantic model. Timer handling, process scheduling, and asynchronous message queues are just a few of them. SDL `resources provide implementations of certain run-time environments that must adhere to the language's semantics. It is also possible to map the aforementioned services to operating system mechanisms that may already be part of the embedded system for performance purposes, eliminating the need for a run-time environment provided by the SDL tool vendor. As a result, certain broad concepts needed to be explored first. A proof of viability and reliability had to be provided as well. An integration library for an example operating system was created for this purpose. Reflex was chosen by the author because it is written in C++ and the implementation of our ideas can be seen clearly in the software design. At the same time, this implementation acts as an essential building block in the design flow. It can be used to create powerful target executables for platforms where Reflex has been ported, such as the Texas Instruments MSP430, Freescale HCS12, or Atmel ATMega 128 microcontrollers (see [18]).

We omitted automatic hardware synthesis from a high-level SDL specification from our proposed embedded systems design flow because of the increased hardware complexity as compared to a hand-optimized semi-custom design. Since the design time is greatly reduced, a hardware compiler is an excellent tool for rapid prototyping. Manual design, on the other hand, promises to produce better results for chips that must be very cheap and, in order to do this, the silicon area must be as small as possible. Mechanisms to buffer SDL signals as inputs to VHDL processes are one source of overhead, as described above in the definition of the COR- SAIR method, which uses automatic VHDL generation. Furthermore, hand-optimized architecture is likely to be more effective in translating finite state machines and SDL data types to hardware. Efficient behavioral compilers for high-level languages are still a hot topic in academia.

It is debatable if mapping complete SDL processes to either hardware or software, as done in the CORSAIR tool, is the best approach. Even if only one or two transformations, such as a time-critical cyclic redundancy check (CRC) measurement, present an actual bottleneck in the implementation, this coarse-grain partitioning will result in the mapping of a full process to hardware. Furthermore, communication through SDL signals between the software and hardware representations of SDL processes takes time because all signal parameters, including control information like the sender process address, must be transferred. As a result, we suggest a fine-grained, arbitrary partitioning of SDL processes that allows the designer to use an optimized interface to the hardware partition.

V. Conclusion

Embedded systems, such as wireless sensor nodes or microcontrollers embedded in another computer, have limited computing, memory, and energy capacity, as well as the need for months or years of trouble-free operation and the ability to connect with other electronic devices. As a result, applications and communication protocols designed for such platforms must maximize the use of available energy and avoid design flaws that could result in device failures or other unexpected actions

References

[1]. Devare, Manoj. (2018). Low Power Communication Protocols for IoT-Enabled Applications. 10.4018/978-1-5225-3805-9.ch003.

[2]. Daryl Narakadan, SadhanaPai, and Manita Rajput (2018) - Wireless Communication between Embedded Systems based on FPGA. International Journal of Engineering & Technology, 7 (4.41) (2018) 66-70

[3]. Stefan Marksteiner, V´ıctor Juan Exposito Jim ´ enez, HeribertVallant, HerwigZeiner (2018) An Overview of Wireless IoT Protocol Security in the Smart Home Domain. arXiv:1801.07090v1 [cs.CR] 22 Jan 2018

[4]. WojciechStaszewski, Adam JABŁOŃSKI, Kajetan DZIEDZIECH (2018) - A Survey Of Communication Protocols In Modern Embedded Condition Monitoring Systems. Diagnostyka. 2018;19(2):53-62.

[5]. Sarawi, Shadi& Anbar, Mohammed &Alieyan, Kamal &Alzubaidi, Mahmood. (2017). Internet of Things (IoT) Communication Protocols: Review. 10.1109/ICITECH.2017.8079928.

[6]. Iniewski, K. (2012). Embedded Systems: Hardware, Design, and Implementation. 10.1002/9781118468654.

[7]. Gade, Sachin&Kanase, AB &Shendge, SB &Uplane, Mahadev. (2010). Serial Communication Protocol for Embedded Application. International Journal of Information Technology. 2. 461-463.

[8]. P.H.W.Leong, "Recent trends in fpga architectures and applications",4th IEEE International Symposium on Electronic Design, Test and Applications 2008, Hong Kong, Jan., pp. 137–141, 2008." J. Serrano, "Introduction to FPGA design," 2008.

[9]. Telelogic AB. Telelogic TAU 4.6 User's Manual, 2006.

[10]. IEEE Computer Society. IEEE Standard SystemC Language Reference Manual. IEEE Std 1666TM-2005, 2005.

[11]. Reinhard Gotzhein, Rudiger Grammes, and Thomas Kuhn. Specify- ing Input Port Bounds in SDL. In Emmanuel Gaudin, Elie Najm, and Rick Reed, editors, SDL 2007: Design for Dependable Systems, 13th International SDL Forum, Proceedings, volume 4745 of Lecture Notes in Computer Science, pages 101–116. Springer, 2007.

[12]. D Dietterle ,Efficient Protocol Design Flow for Embedded Systems

[13]. Annette Muth and Georg F¨arber. SDL as a System Level Specification Language for Application-Specific Hardware in a Rapid Prototyping Environment. In Proceedings of the 13th International Symposium on System Synthesis (ISSS'00), pages 157–162. IEEE Computer Society, 2000.

[14].]. Ralf Henke, Andreas Mitschele-Thiel, Hartmut Konig, and Peter Langendorfer. Automated Derivation of Efficient Implementations from SDL Specifications. Technical report, FIN/IRB, Otto-von- Guericke-Universit¨at Magdeburg, 1996.

[15]. Leue, Stefan, Oechslin, Philippe, A method and tool for optimized parallel protocol implementation, journal of high speed networks 1996 ,2 pp.125-143.

[16]. C.A.M. Marcon, F. Hessel, A.M. Amory, L.H.L. Ries, F.G. Moraes, and N.L.V. Calazans. Prototyping of embedded digital systems from SDL language: a case study. In Proceedings of the Seventh IEEE International High-Level Design Validation and Test Workshop (HLDVT'02), pages 133–138. IEEE Computer Society, 2002.

[17]. Annette Muth. SDL-based Design of Application Specific Hardware for Hard Real-Time Systems. PhD thesis, Chair of Real-Time Computer Systems, Technical University Munich, 2002.

[18]. Jorg Nolte. Reflex - Realtime Event FLow EXecutive. http://wwwbs.informatik.tu-cottbus.de/index.php?id=38&L=2, 2009.

[19]. Bharat M.S ,Dr.Padmashree T. Design and development of embedded system using CAN Protocol. International Journal of Advance Research, Ideas and Innovations in Technology. 2021 (Volume 7, Issue 3) 463-466

[20]. T. Burd, and R. Brodersen, "Processor design for portable systems," *IEEE Journal of VLSI Signal Processing*, to appear 1997.

[21]. P. Bhagwat, P Bhattacharya, A. Krishna, and S. Tripathi, "Enhancing throughput over wireless LANs using Channel-State-Dependent Packet Scheduling," *Proc. of IEEE Infocom '96*. vol 3., pp. 1133-40.

[22]. Joakim Eriksson, Adam Dunkels, NiclasFinne, Fredrik O¨ sterlind, and Thiemo Voigt. MSPsim – an Extensible Simulator for MSP430- equipped Sensor Boards. In Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session, Delft, The Netherlands, January 2007.

[23]. Joachim Fischer, Eckhardt Holz, Martin von Lowis, and Andreas Prinz. SDL-2000: A Language with a Formal Semantics. In The Third Workshop on Rigorous Object-Oriented Methods (ROOM 2000), January 2000.

[24]. Stefan Fischer and Stefan Leue. Formal Methods for Broadband and Multimedia Systems. Computer Networks & ISDN Systems, 9-10(30):865–899, 1998.

[25]. Laura Marie Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Net- working Environment. In 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pages 1548– 1557, 2001.

[26]. Gaisler Research AB. LEON2 Processor User's Manual. http://www.gaisler.com, July 2005.

[27]. Thomas A. Henzinger and Joseph Sifakis. The Discipline of Embed- ded Systems Design. IEEE Computer, 40(10):32–40, October 2007.

[28]. Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System Architecture Directions for Net- worked Sensors. In Architectural Support for Programming Languages and Operating Systems, pages 93–104, 2000.