A systematic LEACH Gateway Design for Wireless Sensor Network

Arun Kumar Pradhan, Deba Narayana Pattanayak, Sushmita Satapathy

[1]*Dept. of ECE, Trident Academy of Technology, Bhubaneswar, Odisha, India*

***Abstract*— ARM Cortex Mx series has been academically suggested and industry opted core for the current embedded verticals. The 32-bit data processing capacity with extreme low power consumption features strive this as a viable asset for Wireless Sensor Network (WSN). This paper discusses about the embedded gateway design using ARM Cortex M4F based Kinetis MK66FN2M0VMD18 as a Micro Controller Unit (MCU). This gateway aims to interconnect the traditional TCP/IP over Ethernet with the deployed WSN using Low-Energy Adaptive Clustering Hierarchy (LEACH) over RF emulated in UART. The working principle of LEACH protocol, justifying its topology management for WSN and the architecture of K66 MCU are discussed in the first part of the paper. The implementation of UART drivers for handling LEACH frames, UDP socket design for IP packets processing is detailed in the next part of the paper. The user interface option with LCD driver implemented using SPI is presented. The integration of these modules in embOS RTOS along with the results and ongoing efforts were projected at the end of the paper.**

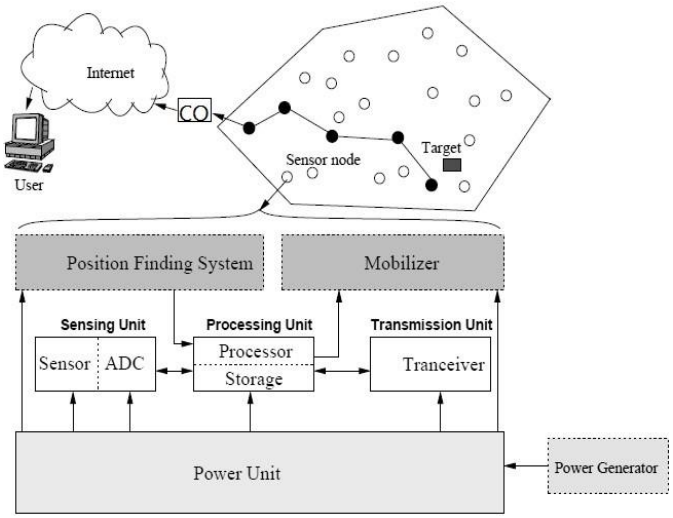
***Keywords*— Wireless Sensor Network; Embedded System; Device Driver; Real Time Kernel; Gateway.**

I. INTRODUCTION

Focus on research in Wireless Sensor Network (WSN) has been urging since the past decade. The three main components in a WSN topology are Coordinator, Router and end-device. Every sensor node is an entity within the WSN deployed network and will act as an end device. This has a tiny Micro Controller Unit (MCU) which is necessary for sensing, digitizing, processing and communicate over RF as indicated in Figure-1. These nodes are self organized among them, acting as router to participate in the multi-hop data communication to reach a centralized gateway or a coordinator. The centralized coordinator is often called as a fusion center or aggregator. This will provide interface to the user or internetworking with the traditional TCP/IP networking.

Though majority of the work is the areas of various routing protocols and topology management, this paper focuses of the issues of Gateway design, which is a crucial entity to continuously connect with the WSN network.

Low-energy adaptive clustering hierarchy (LEACH) is a TDMA-based MAC protocol which is integrated with clustering and a simple routing protocol in WSN. The goal of LEACH is to lower the energy consumption required to create and maintain clusters in order to improve the life time of a wireless sensor network. The working principle of LEACH as routing protocol is discussed in the first part of the paper. However, the main emphasis of the paper is on the gateway processing that has to handle the LEACH queries and response call-back-functions of the connected WSN network and integrate with TCP/IP call-back-functions.



Fig–1: Basic topology of a WSN and single node architecture.

The second part of the paper introduces and proposes the ARM Cortex M4F based K66 MK66FN2M0VMD18 as the gateway MCU. The architectural aspects, embedded firmware along with the user interface using this MCU is presented. The LCD device driver design, initial configuration of GPIO registers and implementation of it using the standard serial peripheral interface (SPI) protocol in Embedded Studio IDE for providing user interface is discussed in detail.

Since the gateway has to tackle multiple internetworking packet processing from Ethernet interface for traditional IP packets, UART interface for LEACH frames along with provision for user interface, the supporting hardware architectural modules and firmware driver interface is discussed.

As the gateway has to offer a reliable solution to be designed with multiple network interfaces along with user interface over LCD, there exists a need of porting a real-time kernel. The preemptive embOS micro kernel has been identified for this MCU and scheduling of multiple tasks in this micro kernel is discussed in detail. The debugging of handling the multiple tasks in RTOS environment is extremely tricky as we add multiple C/C++ modules in identified tasks. The debugging tools and methods for testing the designed modules were discussed in the last part of the paper.

II. LEACH PROTOCOL

The LEACH protocol (Low-energy Adaptive Clustering Hierarchy) presented by Heinzelman et al. assumes a dense sensor network of homogeneous, energy-constrained nodes, which shall report their data to a sink node. In LEACH, a TDMA-based MAC protocol is integrated with clustering and a simple "routing" protocol. LEACH partitions the nodes into clusters and in each cluster a dedicated node, the Cluster-Head (LEACH-CH), is responsible for creating and maintaining a TDMA schedule; all the other nodes of a cluster are member nodes.

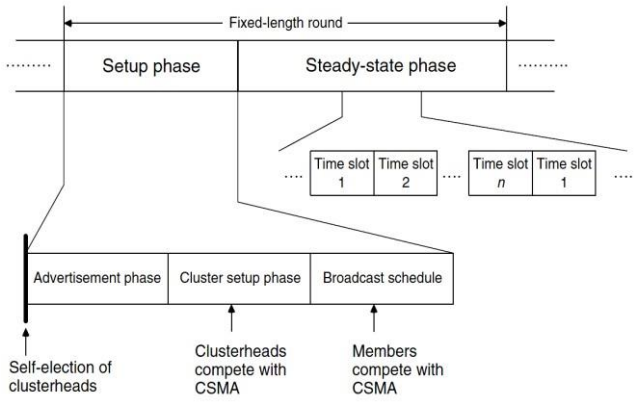


Figure – 2: LEACH – CH Election Process

To all member nodes, TDMA slots are assigned, which can be used to exchange data between the member and the CH; there is no peer-to-peer communication. With the exception of their time slots, the members can spend their time in sleep state.

The CH aggregates the data of its members and transmits it to the sink node or to other nodes for further relaying.

Since the Coordinator is often far away, the CH must spend significant energy for this transmission. For a member, it is typically much cheaper to reach the CH than to transmit directly to the sink. The CHs role is energy consuming since it is always switched on and is responsible for the long-range transmissions. If a fixed node has this role, it would burn its energy quickly, and after it died, all its members would be "head-less" and therefore useless. Therefore, this burden is rotated among the nodes. Specifically, each node decides independent of other nodes whether it becomes a CH, and therefore there is no signaling traffic related to CH election (although signaling traffic is needed for subsequent association of nodes to some CH). This decision takes into account when the node served as CH the last time, such that a node that has not been a CH for a long time is more likely to elect itself than a node serving just recently. The protocol is round based, that is, all nodes make their decisions whether to become a CH at the same time and the non-CH nodes have to associate to a CH subsequently. The non-CHs choose their CH based on received signal strengths. The network partitioning into clusters is time variable and the protocol assumes global time synchronization. After the clusters have been formed, each CH picks a random CDMA code for its cluster, which it broadcasts and which its member nodes have to use subsequently.

A critical network parameter is the percentage of nodes that are CHs. If there are only a few CHs, the expected distance between a member node and its CH becomes longer and therefore the member has to spend more energy to reach its CH while maintaining a given BER target. On the other hand, if there are many CHs, there will be more energy expensive transmissions from CHs to the Gateway and less aggregation. Therefore, there exists an optimum percentage of CHs, which for the scenario investigated in is ≈5%. If this optimum is chosen, LEACH can achieve a seven to eight times lower overall energy dissipation compared to the case where each node transmits its data directly to the fusion center, and between four and eight times lower energy than in a scenario where packets are relayed in a multi-hop fashion. In addition, since LEACH distributes the CH role fairly to all nodes, they tend to die at about the same time. The protocol is organized in rounds and each round is subdivided into a setup phase and a steady-state phase as shown in figure–2.

The setup phase starts with the self-election of nodes to CHs. In the following advertisement phase, the CHs inform their neighborhood with an advertisement packet. The CHs contend for the medium using a CSMA protocol with no further provision against the hidden-terminal problem. The non-CH nodes pick the advertisement packet with the strongest received signal strength. In the following cluster-setup phase, the members inform their CH ("join"), again using a CSMA protocol. After the cluster setup-phase, the CH knows the number of members and their identifiers. It constructs a TDMA schedule, picks a CDMA code randomly, and broadcasts this information in the broadcast schedule sub-phase. After this, the TDMA steady-state phase begins. Because of collisions of advertisement or join packets, the protocol cannot guarantee that each non-CH node belongs to a cluster. However, it can guarantee that nodes belong to at most one cluster. The CH is switched on during the whole round and the member nodes have to be switched on during the setup phase and occasionally in the steady-state phase, according to their position in the cluster's TDMA schedule. With the protocol described so far, LEACH would not be able to cover large geographical areas of some square miles or more, because a CH two miles away from the gateway likely does not have enough energy to reach it at all, not to mention achieving a low BER. If it can be arranged that a CH can use other CHs for forwarding, this limitation can be mitigated.

III. KINETICS ARM CORTEX M4F MK66 MCU

NXP's Kinetis MK66FN2M0VMD18 is operating at 180MHz 32-bit MCU core from ARM's Cortex-M class adding DSP instructions and single-precision floating point unit based on ARMv7 architecture targeting microcontroller cores focused on very cost sensitive, deterministic, interrupt driven environments. The Cortex M4 processor is based on the ARMv7 Architecture and Thumb®-2 ISA and is upward compatible with the Cortex M3, Cortex M1, and Cortex M0 architectures. Cortex M4 improvements include an ARMv7 Thumb-2 DSP providing
32-bit instructions with SIMD (single instruction multiple data) DSP style multiply-accumulates and saturating arithmetic.

As indicated in figure–3, this MCU has multiple low- power modes to provide power optimization based on application requirements, Hardware random-number generator with supports for DES, AES, SHA accelerator and Multiple levels of embedded flash security. It has 2
MB program flash memory with 256 KB RAM with a support of SRAM controller.

This MCU is equipped with rich set of communication interfaces such as hardware IEEE 1588 Ethernet, USB full-speed, On-the-Go (OTG) on-chip transceiver, RTC, Timers, Two CAN, three SPI and four I2C modules, Low Power Universal Asynchronous Receiver or Transmitter-0 (LPUART0) and five standard UARTs, GPIOs, Secure Digital Host Controller (SDHC) and I2S module with low- power hardware touch sensor interface (TSI) for Human- Machine Interface (HMI).

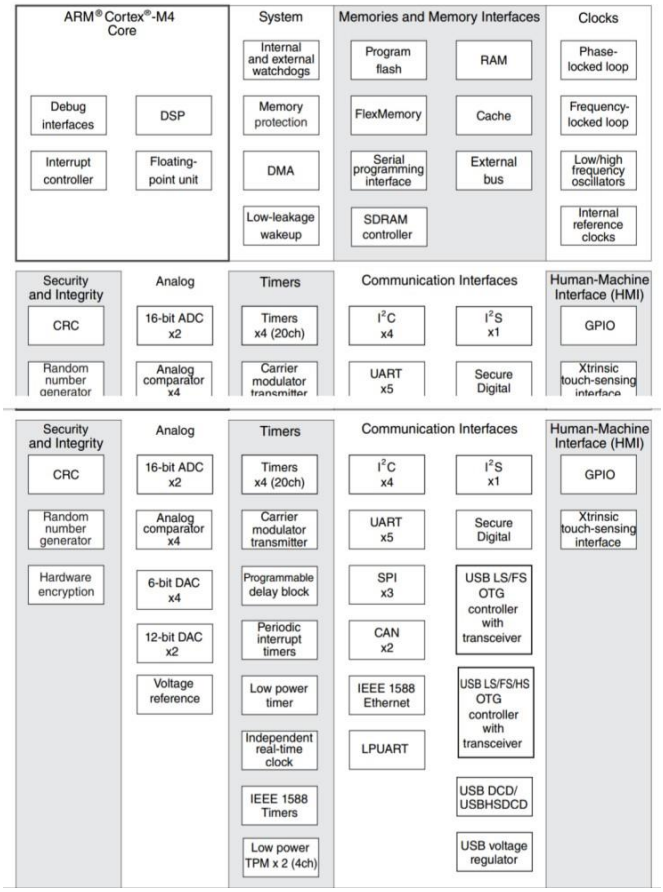


Figure – 3: Kinetis K66 Functional Block Diagram

IV. UART DRIVER FOR IMPLEMENTING LEACH

The UART module within the Kinetis MCU family of devices allows full duplex, asynchronous, non-return to zero serial communication between the CPU and remote devices.

The UART transmitter and receiver operate independently, although they use the same baud rate generator. Data formats can be programmed for 8- or 9-bits with programmable polarity and the ability to select MSB or LSB as the first bit; it can operate in interrupt, DMA or polled modes and it provides two wake-up methods. In addition, the UART also supports IrDA and ISO-7816 functionality. The initialization of UART peripheral is done using the BSP API which is implemented as shown in Pseudo Code listing–1.

```
void BSP_UART_Init(unsigned Unit, U32 Baudrate,
U8 DataBits, U8 Parity, U8 StopBits)
{


 U8 v; _UartTxIsBusy  = 0; // Enable clocks.
 SIM_SCGC5 |= UART_SCGC5_MASK;
 SIM_SCGC  |= SIM_SCGC_UART_MASK;


 // Setup port pins.
 UART_RX_PORT  =  PORT_PCR_MUX(0x03);
 UART_TX_PORT  =  PORT_PCR_MUX(0x03);
 if (StopBits == 2)
     UART->BDH |= (1 << 5);
 _SetBaudrate(Unit, Baudrate);
 //     Setup     UART     interrupts.
 NVIC_SetPriority (UART_RX_TX_IRQn,
(1<<__NVIC_PRIO_BITS)-2);
 NVIC_EnableIRQ (UART_RX_TX_IRQn);
 NVIC_SetPriority (UART_ERR_IRQn,
(1<<__NVIC_PRIO_BITS)-2);
 NVIC_EnableIRQ(UART_ERR_IRQn);
}
```

Listing – 1: UART driver Initialization routines

Once the UART is initialized in order to handle the LEACH traffic from the sensor nodes or from the nearest CH neighborhood, the following set of call-back functions as illustrated in source listing–2 were incorporated into the design so as to trigger the interrupt either during the UART read or write operations.

```
//The call-back-function for UART Rx isr
void BSP_UART_SetReadCallback(unsigned Unit,
    void (*pf)(unsigned Unit, unsigned char
    Data))
{
  BSP_UART_USE_PARA(Unit);
  _pfUartReadCallback = pf;
}
//The call-back-function for UART Tx Isr
void BSP_UART_SetWriteCallback(unsigned Unit,
int (*pf)(unsigned Unit))
{
  BSP_UART_USE_PARA(Unit);
  _pfUartWriteCallback = pf;
}
```

V. IP TRAFFIC OVER ETHERNET

For the validation of the Gateway, TCP/IP stack is ported with the static IP of "192.168.2.252" with Gateway IP of "192.168.2.1" and the same is also set as the DNS entry. For effective utilization of the bandwidth, the UDP socket is opted as the overhead is fairy minimum compared with the TCP Socket. With these settings for the existing Ethernet MAC interface, the set of call-back-functions for IP packet transmit and IP packet receive were implemented as shown in source listing – 3.

```
//Call back when the application sends UDP
static int _OnRx(IP_PACKET *pInPacket,

                 void *pContext)
{
  OS_U32 i;


  OS_USEPARA(pContext);
  // Get IP address from incoming packet
  IP_UDP_GetSrcAddr(pInPacket, &PeerAddr,
sizeof(PeerAddr));
  PeerPort = IP_UDP_GetFPort(pInPacket);
  // Handle received
  for (i = 0u; i < pInPacket->NumBytes; i++)
   {
    OS_OnRx(pInPacket->pData[i]);
   }
  return IP_OK;
}
//Call Back for incoming UDP Packet
static void _SetupCallback(void)
{

   IP_UDP_Open(0u, 0u, EMBOSVIEW_UDP_PORT,
_OnRx, 0u);
}
```

Listing – 3: UDP Socket IP Call-backs over Ethernet.

These functions are call backs when a client UDP called from stack whenever we get a UDP packet and whenever we need to send UDP packet from the application.

VI. SPI LCD DEVICE DRIVER

For MK66F18x, the LCD interface can be implemented using SPI. The LCD device initialization, LCD Reading process and LCD writing process are implemented with the preliminary LCD initialization macro functions. Upon the successful initialization of the SPI driver, SPI read & SPI Write functions were implemented in polling mode as shown in source listing–4. The data which needs to be displayed on the LCD is passed as an argument to the write function and the ASCII data is displayed.

Listing – 2: UART Call-back APIs for LEACH traffic

```
void LCD_X_SPI4_WriteM1(unsigned char * pData,
int NumBytes)
 {
  while ((SPI2_SR & SPI_SR_TCF_MASK)==0);
  LCD_SET_A0();
  for (; NumBytes; NumBytes--)
   {
    while((SPI2_SR & SPI_SR_TCF_MASK)==0);
    SPI2_SR = SPI_SR_TCF_MASK;
    SPI2_PUSHR=(*pData++ );
  }
}
```

Listing – 4: LCD ASCII data Write API using SPI.

VII. MULTI TASKING IN EMBOS KERNEL

Since the traffic from the WSN and from the traditional IP over Ethernet has to be processed without missing the packets, the need of multitasking and thereby need of a Real Time kernel is needed. The above mentioned independent drivers were integrated by creating three tasks namely TCPTask, WSNTask and UITask with 100, 99 and 98 as priority levels. The embOS micro kernel is RMS based fixed priority algorithm with the higher the priority number the highest is the priority. These 3 tasks were created and a template application has been designed as indicated in source listing–5.

```
OS_InitKern(); /*Initialize OS */
OS_InitHW(); /*Initialize Hardware for OS */
BSP_Init(); /* Initialize Single Board Com*/
  // Start creating the tasks.
OS_CREATETASK (&TCBTCP, "TCP_Task",
                TCPTask, 100, StackTCP);
OS_CREATETASK (&TCBWSN, "WSN_Task",
                WSNTask,  99, StackWSN);
OS_CREATETASK (&TCBUI, "UI_Task",
                UITask,  98, StackUI);
OS_Start();/* Start multitasking */
```

Listing – 5: embOS RTOS task creation and Scheduling.

VIII. RESULTS AND CONCLUSIONS

The paper proposes usage of ARM Cortex M4F based Kinetis K66 MK66FN2M0VMD18 as MCU in order to network the existing IPv4 as a Gateway to the deployed WSN network running on LEACH. The MCU is proved to be extremely cost sensitive for the need and the modules which are mandate to design a prototyped gateway are highlighted. The working principle of LEACH and election of the Cluster Head (CH) is represented in the first part of the paper. Many research results shows that LEACH may be widely accepted protocol for the deployment of dense WSNs because of its low power characteristics, robustness and scalability.

Figure – 4: MCU Gateway configured with static IP

The architectural considerations for K66 to be a gateway entity handling with LEACH traffic is highlighted in the next part of the paper. The configurability option available for UART transmit and receive modules for this MCU is discussed. The TCP/IP networking stack is ported with UDP socket and the MCU is configured with a static IP running a pre-assigned port in application layer. The display driver is implemented using SPI and the LCD module with template menus connected to this MCU is designed and implemented as shown in figure–4. The embOS as RTOS is selected with three tasks and integration of UART, TCP stack and SPI user interface is implemented. This scheduling demonstrates that the suggested MCU can be effectively utilized as a WSN gateway over existing IP network to manage the available LEACH nodes.

REFERENCES

[1] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy- efficient communication protocol for wireless microsensor networks", IEEE Proceedings of the Hawaii International Conference on System Sciences, pp. 1-10, January 2000.

[2] V.K. Arora, V. Sharmab, M. Sachdevaa, "A survey on LEACH and other's routing protocols in wireless sensor network", Optik - International Journal for Light and Electron Optics, vol. 127, no. 16, pp. 6590-6600, August 2016.

[3] N. A. Ali Khan and K. J. Sankar, "User interface design for LPC2138 to configure wireless sensor node parameters," 2014 International Conference on Smart Structures and Systems (ICSSS), Chennai, 2014, pp. 31-34. doi: 10.1109/ICSSS.2014.7006191

[4] A. Razaque, S. Mudigulam, K. Gavini, F. Amsaad, M. Abdulgader, G. S. Krishna, "H-LEACH: Hybrid-low energy adaptive clustering hierarchy for wireless sensor networks", 2016 IEEE Long Island Systems Applications and Technology Conference (LISAT), pp. 1-4, 2016.

[5] A. Razaque, M. Abdulgader, C. Joshi, F. Amsaad, M. Chauhan, "P-LEACH: Energy efficient routing protocol for Wireless Sensor Networks", 2016 IEEE Long Island Systems Applications and Technology Conference (LISAT), pp. 1-5, 2016.

[6] C. Bejaoui, A. Guitton, A. Kachouri, "Improved election of cluster heads in LEACH", 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), pp. 1-4, 2015.

[7] S.E.L. Khediri, N. Nasri, A. Wei, A. Kachouri, "A New Approach for Clustering in Wireless Sensors Networks Based on LEACH", The 5th International Conference on Ambient Systems Networks and Technologies (ANT-2014) Procedia Computer Science, vol. 32, pp. 1180-1185, 2014.

[8] L.M. Kamarudin, R.B. Ahmad, D.L. Ndzi, A. Zakaria, K. Kamarudin, M.E.E.S Ahmed, "Simulation and analysis of LEACH for wireless sensor networks in agriculture", International Journal of Sensor Networks, vol. 21, no. 1, 2016.

[9] A. Bouyer, A. Hatamlou, M. Masdari, "A new approach for decreasing energy in wireless sensor networks with hybrid LEACH protocol and fuzzy C-means algorithm", Int. J. of Communication Networks and Distributed Systems, vol. 14, no. 4, pp. 400-412, 2015.

[10] B. Wanga, H.B. Limb, D. Mab, "A coverage-aware clustering protocol for wireless sensor networks", Computer Networks, vol. 56, no. 5, pp. 1529-1611, March 2012.

[11] Freescale Semiconductor, Inc.; K66 Sub-Family Reference Manual Supports: MK66F/N2M0VMD18 X1M0VMD18, Doct Number: K66P144M180SF5RMV2 Rev. 2, May 2015.

[12] I F Akylidiz, W Su, Y Sankarasubramaniam, E Cayirci. Wireless sensor networks: a survey. Computer Networks 2012; 38 (4): 393-422, DOI: 10.1016/s1389- 1286(01)302-4.

[13] E Lule, T E Bulega. A scalable wireless sensor network (WSN) based architecture for fire disaster monitoring in the developing world. International Journal Computer Network and Information Security, 2015; 7 (2):40-49.

[14] W.B Heinzelman. Application-specific protocol architectures for wireless networks. 2000; Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge.

[15] O Younis, M Krunz, S Ramasubramaniam. Node clustering in wireless sensor networks: Recent developments and deployment challenges. IEEE Network 2006; 20(3), 20-25, DOI: 10.1109/MNET.2006.1637928.

[16] W Heinzelman, A Chandrakasan, H Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. 10.1109/HICSS.2000.926982. Proceedings of Hawaii International Conference on System Sciences 2000.

[17] W Heinzelman, a Chandrakasan, H Balakrishnan. Application-specific protocol architecture for wireless microsensor networks. IEEE Transactions on Wireless Communications 2002; 1(4): 660-670, DOI: 10.1109/TWC.2002.804190.

[18] M Tong, M Tang. LEACH-B: An improved LEACH protocol for wireless sensor network. Proceedings of International Conference on Wireless Communications Networking and Mobile Computing 2010; 1-4, DOI: 10.1109/WICOM.2010.5601113.

[19] W Xinhua, W Sheng. Performance comparison of LEACH and LEACH-C protocols by NS-2. Proceedings of International Symposium on Distributed Computing and Applications to Business Engineering and Science 2010; 254-258, DOI: 10.1109/DCABES.2010.58.