Wireless Body Area Network Studies for Telemedicine Applications Using IEEE 802.15.6 Standard

Hasan Yavuz Özderya[1] Hakan Erdöl[1] Temel Kayıkçıoğlu[1] Ali Özgür Yılmaz[2] İsmail Kaya[1]

[1] The Department of Electrical and Electronics Eng., Karadeniz Technical Univ., Turkey
[2] The Department of Electrical and Electronics Eng., Middle East Technical Univ., Turkey

Abstract. Wireless communication is becoming a part of our life at every step. But widespread use in medical applications is yet to come. We are developing a wireless communication system based on 802.15.6 MAC and 802.15.4 PHY for use in transmitting ECG data from a remote patient monitoring device which is used for home based telemedicine applications. The paper concentrates on explaining the stack program development phases of the standard IEEE 802.15.6 and its flexible access architecture/features. It is believed that the subjected standard is going to be medical data highway in its reserved bands.

1 Introduction

Developments in electronics, battery technology and wireless communication made wireless technologies a part of our daily life. Everyday a new wireless equipment surfaces be it a wireless version of an existing equipment or an equipment that is made possible by wireless connectivity. But wireless medical equipment is still uncommon since current state of wireless communication doesn't provide the reliability expected from a medical device.

We believe wireless, battery powered devices are essential for home based patient monitoring. Compared to traditional medical equipment wireless monitoring devices will offer more comfort and mobility for the patient.

A home based patient monitoring systems starts with the sensors that are placed on the patient. These sensors collect the medical data from patients body and transmits it to a nearby Hub device. Hub is connected to a host PC. Instead of a personal computer like laptop, a headless (no monitor, keyboard etc.) mini computer can also be used. Collected data is transmitted to the server over internet. Server is responsible for storing patient's medical data, and processing the data to generate alarm. Other side of the system is the client which is in this case the doctor. Doctor connects to the server via a special client software and monitors the patient in a realtime fashion or investigates the previously stored data. A key part of this system is the wireless communication protocol

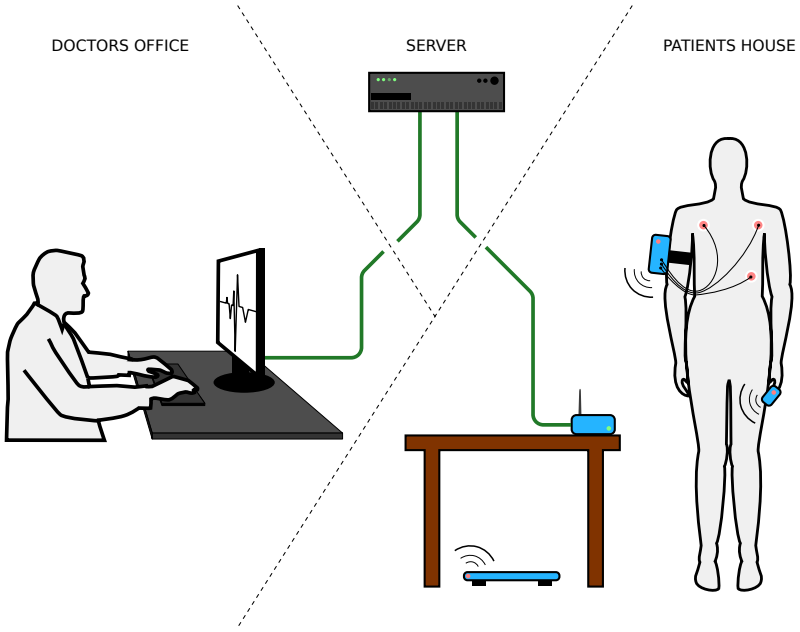

Fig. 1. Basic elements of a home based patient monitoring system

that facilitates the connection between body worn sensors and the Hub which we call "Body Area Network". In Fig. 1 main components of such a system is illustrated.

Existing wireless communication protocols that are in common use at the moment are not seen fit for medical purposes. 802.11 (Wi-Fi) is known for its high power consumption. 802.15.4 (ZigBee) is designed for low power sensors. But its primarily targeted at low throughput networks thus not fit for real time patient monitoring. Bluetooth is also designed for low throughput devices and doesn't offer reliability expected from a medical equipment.

Recognizing these problems, in 2007 an IEEE task group is established to develop a new wireless communication standard to address the wireless connectivity needs of medical applications. IEEE released 802.15.6 "Body Area Network" specification in 2012. [3] Aim of the specification is to define a communication protocol that will be used with sensors usually on a persons (patient) body and send data to a nearby central Hub such as PDA while providing cable equivalent reliability. [8]

IEEE 802.15.6 defines multiple radio bands for operation. But maybe most important of these is 2360-2400MHz band which is dedicated to medical ban applications by FCC in 2012. [1] While lower 30MHz portion of this band is to be used in health care facilities and requires registration, upper 10MHz portion of the band doesn't require a license which makes it suitable for home based telemedicine applications.

IEEE 802.15.6 defines a flexible PHY layer which supports multiple data rates. A node can increase and decrease its speed depending on its bandwidth requirement and or link quality. A forward error correction method based on BCH coding is also defined for increased reliability and lower power consumptiveness. Forward error correction allows a package to be salvaged if only a very small portion of it is damaged during transfer, otherwise sensor would have to re-transmit complete data package wasting valuable link resources and battery power.

When it comes to MAC (Medium Access Layer) 802.15.6 also offers multiple options. There are 3 main access methods.

1. Beacon mode with superframes
2. non-beacon mode with superframes
3. non-beacon mode without superframes

In Mode 1, timebase is divided into superframes that has a size determined by the Hub. Hub can allocate certain parts of superframe to specific nodes per its requirements. Some parts of a superframe can be allocated for contention access. At the beginning of each superframe Hub transmits a special frame called "beacon". This frame contains information on the network, such as Hub's address, superframe length and its structure. This allows a node looking for a new BAN to discover and connect to Hub. Beacon also provides a mean for nodes to continuously synchronize to Hub's timebase. In Mode 2, there are still superframes but no beacon is sent. In this mode, Hub should provide polling frames for facilitating new connections and synchronization (timed poll) of the nodes. In Mode 3, there are no superframes at all. In this case Hub should manage the communication by polling.

Having these options allows the implementation of different style of communication protocols based on 802.15.6 specification. For our application we have decided to implement the first access mode since TDMA based access methods are known for their reliability and power efficiency. [9] In Fig. 2 a simplified version of a superframe is shown. A superframe may consists of 4 different phases of variable sizes. First phase right after the beacon frame is the Emergency Access Phase. It's allocated for emergency transmission of highest priority frames. After that Random Access Phase follows. In random access phase, nodes can communicate with Hub freely using contention access methods. For narrow band PHY, 802.15.6 specifies that CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) should be used during RAP. After that, Managed Access Phase comes. This phase, as implied by the name is managed by the Hub which allocates it to the nodes according to their requests, taking into account their bandwidth requirements and priorities. MAP is divided into allocation slots of equal length. These slots can be allocated to nodes as uplink, downlink or bilink allocations. In an uplink allocation node transmits its frames to Hub, in downlink allocation Hub transmits its frames to node. In bilink allocation communication is controlled by the Hub using poll and post frames. With a poll frame Hub requests frames from node, with a post frame Hub transmits a frame to node.

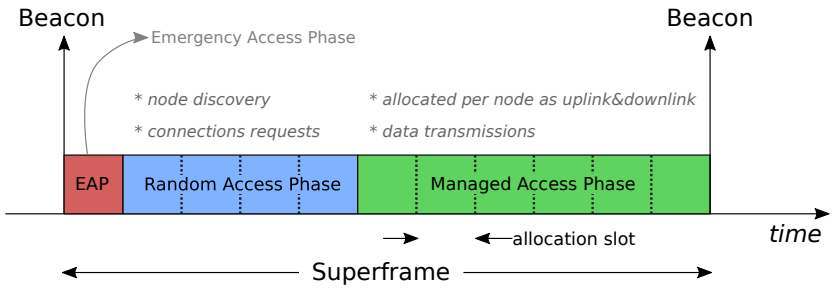

Fig. 2. Basic structure of a simplified 802.15.6 superframe showing beacon frame, EAP, RAP and MAP

1.1 Related Work

In [7] a home based remote patient monitoring system is implemented using 802.15.4 protocol with existing ZigBee modules. A custom FEC (Forward Error Correction) mechanism is implemented and systems efficiency is evaluated under 802.11 interference, it is shown that a reliable system is not possible in a band that has heavy 802.11 traffic present. In [6], a wireless wearable ECG and respiration recording system is implemented using Bluetooth protocol and a mobile phone as the gateway. [4] is another work that is based on Bluetooth protocol also uses the mobile phone of the patient as the Hub. Bluetooth SPP (Serial Port Profile) is used to transfer data. In [5] a simulation study on 802.15.6 access mode is done. CSMA/CA based access and polling based access is compeared in a scenario that contains multiple patient monitoring devices. It is shown that in case of a high througput sensor device (ECG) run time of the device is greatly improved when managed access phase is longer than the random access phase.

2 Design

2.1 Hardware

Hub Main processing unit is a STM32F4 MCU from St Micro which is based on Arm Cortex-M4 architecture. It has 1MB flash and 256 KB RAM. Such powerful MCU is selected since Hub has to serve multiple nodes in a timely fashion.

Other main part of the HUB unit is the transceiver. Due to lack of a commercially available 802.15.6 narrow band PHY, we have selected to use Atmel's AT86RF233 802.15.4 transceiver. Main criteria for its selection was that it supports operation in MBAN frequencies (2360-2400MHz) assigned by FCC. [1] It also provides some optional features for ZigBee MAC implementation but these features are not used since we are implementing our own MAC layer based on 802.15.6 specification.

Node In Fig. 3 a diagram of the node board is shown. To keep implementation at this stage of our work simple, we have used the same MCU & transceiver pair for the node. We plan to replace it with a smaller MCU such as a Cortex-M0 based one or even MSP430. Use of separate processing units for communication and sensor control is also considered to reduce complexity of operation. Node device has these sensor components:

- ECG frontend
- Motion sensor
- Humidity and Temperature sensor

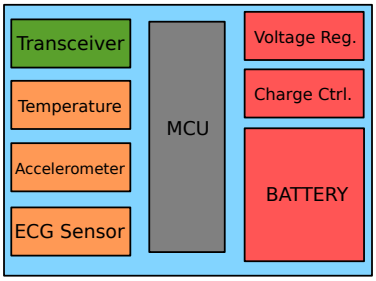

Fig. 3. Diagram that shows basic hardware components of a wireless node

2.2 Firmware

Firmware is developed in C programming language on top of ChibiOs RTOS. ChibiOs is a real time operation system which supports tickless operation. ChibiOs also provides a well integrated HAL for STM32 MCUs.

Firmware consists of several layers. These are illusturated in Fig. 4. "PHY Control" layer is the implementation of SPI interface between MCU and the transceiver IC. Its responsibility is to transfer frames to transceiver and receive frames and manage transceiver interrupts. MAC layer is our 802.15.6 implementation. Its main responsibility is the management of node connections, transmission and reception of frames at appropriate times. Service layer is described in detail in the next section.

2.3 Service Layer

Unlike some other communication protocols (Bluetooth), 802.15.6 doesn't define the full communication stack. 802.15.4 is in a similar situation in this regard. But there are already multiple communication stacks developed for 802.15.4 specification such as ZigBee and 6LoWPAN. We have developed our own stack for use in body area networks that is based on modular services.

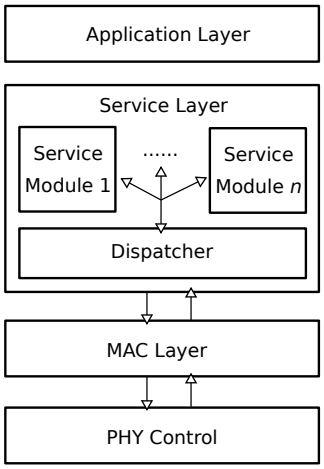

Fig. 4. Layers of the firmware running on the wireless node

Services are defined as modules in a special file in YAML format. Any part of the application that serves a distinct purpose can be represented as a separate module. For example for node device, we have modules such as "ECG", "Temperature", "Motion" for patient tracking, and modules such as "DeviceInfo", "Battery", "RTC" (Real Time Clock) for access to remote devices status information. Even our Hub device is implemented in this modular structure. But its structure is much simple. It only contains 3 modules; "DeviceInfo", "Hub", "Temperature".

"DeviceInfo" module is a special module that each device has to implement. It provides information about device such as its connection ID, name, unique ID, list of modules. Upon accepting a connection request from a node, first thing Hub does is to ask to node for a list of its modules. Further actions are taken according the service modules that device has.

Module structure is similar to classes in an object oriented programming language. Each module can define 3 type of members.

- attribute
- method
- signal

'Attributes' are for quick read/write access to module features that usually control module operation (start/stop, sensor configuration). An attribute can of the basic variable types such as "integer", "boolean", "string" etc. "Methods" are similar to member functions. They can be called from host application with parameters and they return some information in response. A method always sends a response, at the very least a success code. "Signal"s can be sent by a device at any time without a request from host. They are usually used to transfer

results of continuously running operations such as ECG data acquisition, or to notify the host about events, for example a new connection, battery alarm etc. This eliminates the need for host to continuously poll for information.

Here is an example module definition:

```
- name: ECG
  attributes:
    - {name: SampleRate , type: Integer}
    - {name: Gain       , type: Integer}
  methods:
    - name: Start
      params:
        - {name: numOfChannels, type: Integer}
    - {name: Stop}
  signals:
    - name: Data
      params:
        - {name: time, type: int64}
        - {name: data, type: int16, array: True}
```

This module is defined to control the operation of the ECG acquisition. It has two attributes that control sampling rate of ADC and amplitude gain. It also has methods to start and stop module operation. "Start" method requires a parameter which determines the number of channels. "Stop" method doesn't have any parameters. Module also defines a signal named "Data" which is used to send ECG data to host. Main difference between a method and signal is that signals are not controlled by host, they can be sent any time from the device.

Processing of the Module files Module files are processed by the generator program to generate source code to both implement the actual code on the embedded system and the driver for accessing the modules of the remote device.

For example, these function templates are automatically generated by the program but their implementation for the embedded system has to be done by the developer.

```
// Note: function parameters are omitted in this listing

// handlers for implementing attribute access functionality
MTL_STATUS mtl_module_ecg_get_attr_samplerate(...);
MTL_STATUS mtl_module_ecg_set_attr_samplerate(...);
MTL_STATUS mtl_module_ecg_get_attr_gain(...);
MTL_STATUS mtl_module_ecg_set_attr_gain(...);

// handlers for implementing methods
MTL_STATUS mtl_module_ecg_start(...);
MTL_STATUS mtl_module_ecg_stop(...);
```

Driver Driver library is developed in Python and C# programming languages. Module interface of the driver is auto generated from the module definition files. This allows us to easily add new modules to system or modify existing ones, without touching the driver code.

An example of access to a remote module using Python driver:

```
connection = MTL.Connection("/dev/ttyACM0")

# register a signal handler
connection.device(devid).ECG.Data.connect(dataHandlerFunc)
# call a method from the remote device to start ECG acquisition
connection.device(devid).ECG.start(3)  # number of channels 3

def dataHandlerFunc(signal):
    # process signal payload here
```

Similarities to Bluetooth Generic Attribute Profile Our system somewhat resembles the Bluetooth's GATT (Generic Attribute) profile. With GATT profile developers can implement their own profile. In GATT, there are services that consists of "characteristics". [2] These "characteristics" can be of any type Bluetooth specification defines such as "string", "integer" or special purpose such as "Weight", "Temperature" etc. User can also define custom characteristics that is distinguished with a unique ID. Characteristics can be read or written by a host. Device can also send the information provided by the characteristics to the host, with a notify event. One advantage over our system is that major mobile and desktop operating systems already have drivers for GATT profile in place that application developers can use.

3 Conclusions

In this work we have implemented a wireless communication protocol based on 802.15.6 MAC layer and using a 802.15.4 PHY transceiver. We have defined and implemented a modular service layer that simplifies the addition of modules to the firmware and automatically creates respective driver interface.

4 Acknowledgments

This research is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant 114E452.

References

1. FCC Dedicates Spectrum Enabling Medical Body Area Networks | Federal Communications Commission. https://www.fcc.gov/document/fcc-dedicates-spectrum-enabling-medical-body-area-networks

2. Generic Attribute Profile (GATT) Specification | Bluetooth Technology Website. https://www.bluetooth.com/specifications/generic-attributes-overview
3. IEEE Standard for Local and metropolitan area networks - Part 15.6: Wireless Body Area Networks. IEEE Std 802.15.6-2012 pp. 1–271 (Feb 2012)
4. Chan, C.C., Chou, W.C., Chen, C.W., Ho, Y.L., Lin, Y.H., Ma, H.P.: Energy efficient diagnostic grade mobile ECG monitoring. In: 10th IEEE International NEWCAS Conference. pp. 153–156 (Jun 2012)
5. Jacob, A.K., Jacob, L.: An investigation into the effectiveness of IEEE 802.15.6 MAC access methods. In: 2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES). pp. 1–5 (Feb 2015)
6. Jhuang, J.W., Ma, H.P.: A patch-sized wearable ECG/respiration recording platform with DSP capability. In: 2015 17th International Conference on E-Health Networking, Application Services (HealthCom). pp. 298–304 (Oct 2015)
7. Kritsotaki, A., Perakis, K., Koutsouris, D.: Evaluation of 802.15.4 WPAN for patient monitoring. In: Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine. pp. 1–4 (Nov 2010)
8. Martelli, F., Buratti, C., Verdone, R.: On the performance of an IEEE 802.15.6 Wireless Body Area Network. In: 17th European Wireless 2011 - Sustainable Wireless Technologies. pp. 1–6 (Apr 2011)
9. Ullah, S., Shen, B., Islam, S.R., Khan, P., Saleem, S., Kwak, K.S.: A Study of MAC Protocols for WBANs. Sensors (Basel, Switzerland) 10(1), 128–145 (Dec 2009)