Real-time performance evaluation of Bluetooth ARQ protocol

Fabrice PEYRARD

Abstract: **These research tasks present a measurement platform of Bluetooth asynchronous links in order to get the intrinsic time constraints of this network and communications protocols. These time measurements are necessary for the application we wish to implement for mobile robot control through Bluetooth link communication. We present the platform as well as the measurement protocol which we have carried out from real-time communicating operating systems. We have developed an application of radio and time data processing allowing a real-time evaluation of the global behavior of the communicating system.**

Keywords**: ARQ measurement, Bluetooth WPAN, real-time system**

I. INTRODUCTION

The objectives of our research work [5] [6] are focused on the implemention of a Bluetooth [1] communication platform in point-to-point mode to control and command a mobile robot with a remote RTAI[1] [12] real-time system. According to the properties of the Bluetooth [2] link communication in such a system, we can determine thanks to the measurement platform, the supported time constraints which will therefore be the entry parameters to the servo-control loop to control and command the robot through the Bluetooth WPAN[2]. Some research works on Bluetooth protocols [3] [11] and architectures [8] are done for communications but less frequently for command control systems [13]. The objectives of the work presented in this paper are to show the intrinsic properties of a DM1[3] point-to-point Bluetooth link in particular the ARQ[4] influence on the servo-control loop. We are emphasizing the implementation of measurement platform, the associated protocol and our results.

In this work we have identified the time constraints of the Bluetooth WPAN for applications requiring a high QoS level, such as remote process control or voice applications. The mobile robot control is an application with hard time contraints and whose imperfections of the wireless medium must be smoothed at the maximum by an efficient communication protocol. We determine in this paper the minimal time limit supported in the servo-control loop between real-time communicating systems.

Manuscript received January 05, 2007 and revised October 09, and December 23, 2007.

Author is with Toulouse University Research laboratory LATTIS-EA4155, France (e-mail: peyrard@iut-blagnac.fr)

[1] Real-Time Application Interface
[2] Wireless Personnal Area Network
[3] Data – Medium rate
[4] Automatic Repeat reQuest

II. BENCHMARK MEASUREMENT

A. Bluetooth piloting module

A1. HCI interface and packets

Thanks to specific commands, the HCI[5] interface allows us to pilot the Bluetooth module for its intrinsic control via the Baseband layer. The transmission of the useful data for ACL[6] links are done via the LMP[7] layer. These different commands and data are transmitted through specific packets identified in three categories:

- the command packets used by the host to control the Bluetooth module,
- the event packets used by the Bluetooth module to answer a command,
- the data packets to transport towards another remote Bluetooth module.

Fig. 1 presents the interaction of these three types of packets between the control station and the wireless Bluetooth module. Every HCI packet is identified by a specific header byte followed by its parameters as Fig. 1 presents. The ACL data packets sent on the HCI interface undergo a specific formatting from the Bluetooth module so that a new packet dedicated to transmitting on the ACL radio link is created.

A2. Bluetooth piloting

In order to optimize the Bluetooth module configuration in our real-time environment we have defined an initialization sequence. The sequence of following commands, illustrated in TABLE 1, represents the necessary and satisfactory command to initialize the Bluetooth module. This sequence is independent of the role played by the Bluetooth module in master or slave communication, which has allowed us to develop a unique `RT_init_BT.o` programmed module for the initialization. Every specific master or slave entity must execute this module and moreover only the master can establish the connection with the slave whose MAC address he knows beforehand. Once the connection is created the slave generates an event of connection establishment. We assessed the processesing times of each command of this initialization module, whose off-connection time is slightly over 23ms. We noted on the other hand that the time of connection establishment is in charge for about 2s. It is important to consider this time during the Bluetooth establishment connection stage in a multi-robot environment (Piconet).

[5] Host Controler Interface
[6] Asynchronous Connection Less
[7] Link Management Protocol

1845-6421/07/7023

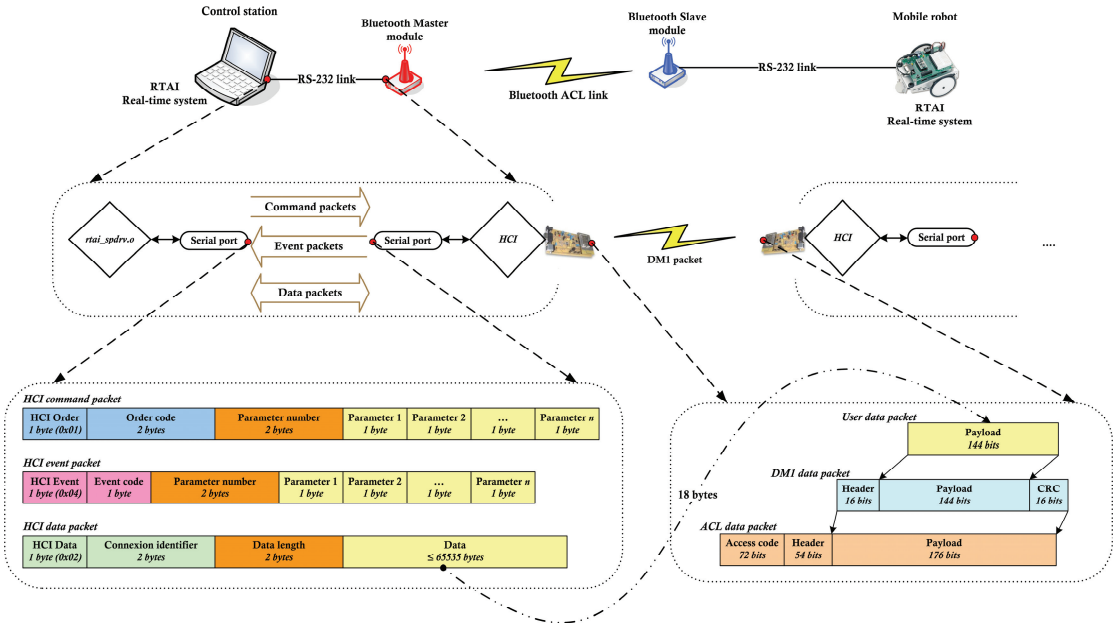

Fig. 1. Piloting a Bluetooth module via the HCI interface

TABLE 1 – MEASUREMENT TIMES OF THE INITIALIZATION SEQUENCE

Actions	*Bluetooth module initialization*	*Automatic detection stage*	*Activation of event control*	*Changing data rate to 115Kbits/s*	*New DM1 connexion on ACL link*
Applicative order	`> cmd reset`	`> cmd wse INQUIRY_PAGE`	`> cmd sef 0x02 0x00 0x02`	`> cmd esuartbr 115200`	`> cmd cc 0x111111111111 DM1 R0 MANDATORY 0x0000 0x00`
HCI order	`# COMMAND 01 03 0C 00`	`# COMMAND 01 1A 0C 01 03`	`# COMMAND 01 05 0C 03 02 00 02`	`# command 01 09 FC 01 02`	`# COMMAND 01 05 04 0D 11 11 11 11 11 11 08 00 00 00 00 00 00`
Running time (ms)	8,198023	4,063218	2,732338	8,054214	1905,595
Initialization time (ms)	**23,047793**				
Total time of connexion establishment (s)	**1,928642793**				

Following this connection stage between master and slave, we itemize, in the next paragraph, the data transmission stage between the two communicating entities in real-time environment.

B. Real-time communicating tasks

In order to validate our experimental platform, we have, for the first time on a real-time mono-system, defined two real-time tasks with the same priority on this system. These `RT_send_acl.o` and `RT_receive_acl.o` tasks process data transmission and reception respectively. The slave process (robot) must be initialized first, because it processes

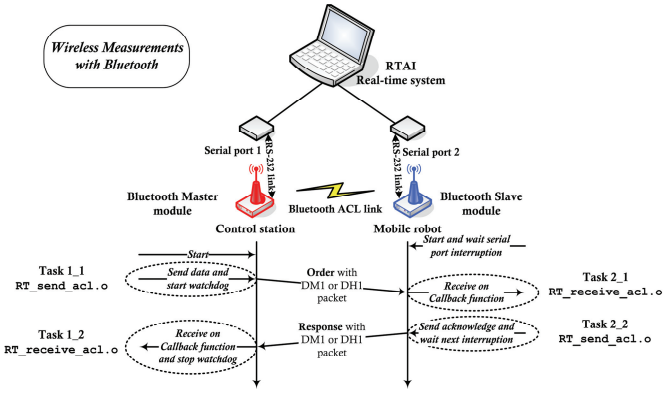

Fig. 2. Communication protocol on mono-system

the aperiodic data reception on serial port interruption. The master processing (control station) sends aperiodic data after defining a random time, then arms a watchdog in case of non reception of acknowledgment from the slave. Fig. 2 presents the communication protocol between the two serial ports of the same real-time system [4] in order to carry out the time measurement from a single clock.

The first essential stage for system validation is to estimate the global behavior of the real-time tasks by interconnecting the serial ports with a wire medium. The reliability of this system leads us then to carry out the same measurement after changing the wired transmission to a Bluetooth communication over the radio medium.

C. Benchmarcks

C1. Measurement on wire medium

Following the network and protocol architecture presented in Fig. 2, we have first estimated the behavior of the wired real-time mono-system by transmitting data packets of variable size, between 1 and 512 bytes. Indeed, we wished to check the intrinsic behavior of the `rtai_spdrv.o` module [10] whose software buffer maximal size is specified at 512 bytes in the RTAI kernel [9]. This platform allows us to carry out time measurements without the constraints of clock synchronization in a multisystem environment. The values

were measured in the wired environment whose serial port data rate is specified at 115200bits/s. We can note that the results obtained are superior to the theoretical value, since they include the transport of all the management bits of the serial interface (stop bit, parity,…) as well as the crossing time of the protocol layers of the real-time system. This additional load is characterized by an increase of between 18 and 73% as Fig. 3 presents.

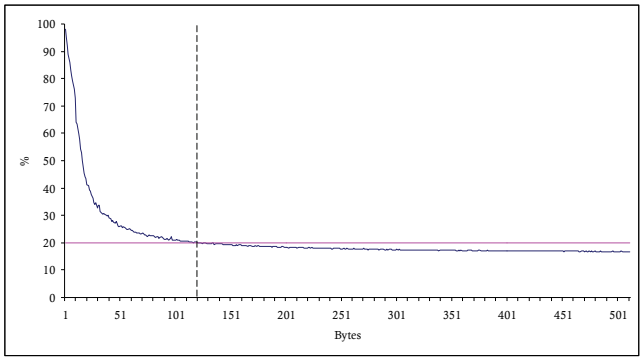

Fig. 3. Transmission time increases according to packet size

The smaller the packet size is, the bigger the global time necessary for processing and transmission. Indeed, this is mainly due to the software processing time by real-time tasks in comparison with the low transmission time. Nevertheless, in an application with strong time constraints, we could use packets over 120 bytes in order not to exceed 20% of increase in processing time.

C2. Measurement with Bluetooth

By using the same protocol presented in Fig. 2, we carried out the time measurements on the Bluetooth communicating mono-system whose results are presented in Fig. 4. We essentially note a weakness of Bluetooth's useful data rate because of the different bytes making up the frame envelope (header, CRC, …), of the management bits of the serial interface, as well as the serialization times and the crossing of Bluetooth modules.

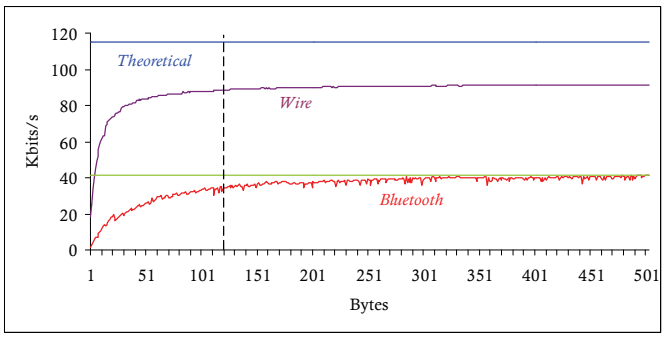

Fig. 4. Time measurements on Bluetooth real-time mono-system

The user data rate offered by Bluetooth on this real-time platform doesn't exceed 40Kbits/s via the HCI interface, which proves to be in conformity with the data rate of about 30Kbits/s [5] [13] measured on the software layers over HCI.

We present in the Fig. 5 a comparative analysis of an ACL Bluetooth link with DM1 and DH1 packets of 366 bits. They differ from one another by a better quality of service for the DM1 packets because they implement an additional ⅔ FEC redundancy code.

The difference in data rate between DH1 and DM1 is only 5Kbits/s at the maximum whereas in theory it is 65Kbits/s. This difference is due to the serial port rate limitation of 115.2Kbits/s.

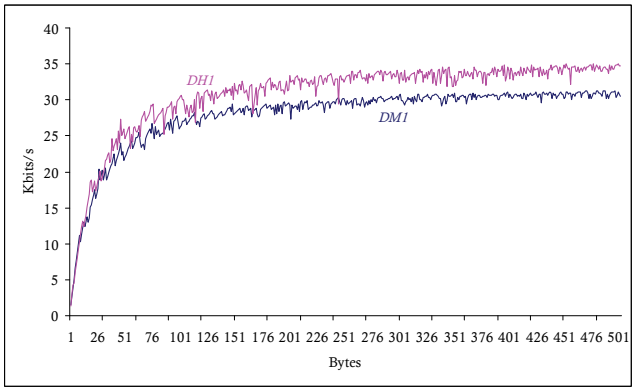

Fig. 5. Bluetooth data rate measured with DM1 and DH1 packets

Therefore, this limitation hasn't allowed us to estimate DM3/DH3 and DM5/DH5 packets explicitly. Indeed in this case, the serial ports are a bottleneck for the ACL Bluetooth channels beyond 115.2Kbits/s.

D. Conclusion

This mono-system experimental model has been designed with the goal of validating the interoperability between the RTAI real-time system and the Bluetooth wireless modules, for asynchronous communications via the system serial interface. We have characterized and developed the real-time transmission and reception tasks by using the RTAI kernel modules particularly for serial port interruption management. Then, we have carried out a whole set of measurements on the wired medium in order to estimate the global behavior of the system, before carrying out these same measurements in the Bluetooth wireless environment and we obtain like the Bluetooth wireless communication performances only 35% of the theorical data rate. We noted the impact of the crossing time of the RTAI layers and the Bluetooth modules, which we characterize more precisely in the following part. For 64% of payload increase between DM1 and DH1 packets, we note only 12% of throughput increase due to lack of FEC on DH1 packets. We have implemented a Bluetooth platform of communication between two communicating entities managed by distinct RTAI real-time systems.

E. Modelling and simulation of Bluetooth Baseband

The goal of this part is to show the advantage of the STPN[8] model in the analysis of the data transfer phase of the Bluetooth baseband layer. These is a dual function: first in terms of the modelling power which is well designed to represent the complex format of the packets at different redundancy levels; then in terms of performance which is

[8] Stochastic Timed Petri Net

assessed for the different types of data packets (DM1 and DH1) to take into account the different probabilities of error per bit.

E1. Receiver model

The receiver model on the slave site is illustrated by Fig. 6. This Petri Net represents the reception of the payload field and the sending of the NULL packet (either ACK or NACK) and is associated with probability density functions.

Probability of Detection of errors on an ERroneous Payload:

$$P_{DERP} = \frac{\sum_{k=1}^{3} C_n^k \, p_{EP}^k (1-p_{EP})^{n-k}}{\sum_{k=1}^{n} C_n^k \, p_{EP}^k (1-p_{EP})^{n-k}}$$

Probability of UnDetected errors on an ERroneous Payload:

$$P_{UDERP} = \frac{\sum_{k=4}^{n} C_n^k \, p_{EP}^k (1-p_{EP})^{n-k}}{\sum_{k=1}^{n} C_n^k \, p_{EP}^k (1-p_{EP})^{n-k}}$$

Probability to have a Correct Bloc of 15 bits:

$$P_{CB} = \sum_{k=0}^{1} C_{15}^k \, p_e^{\,k} (1-p_e)^{15-k}$$

Probability t affects the bits of each group of 10 bits.

$$p_{EP} = \frac{1-P_{CB}}{15}$$

Transitions	Associated probability densities
t_{117}	$f_{117}(x)=\delta(x-nTB)$
t_{118}	$f_{118}(x)=2P_{ERP}*\delta(x-n\,TB) + (1-2\,P_{ERP})$ with $n \le x \le n+1$
t_{120}	$f_{120}(x)=\delta(x)$
t_{121}	$f_{121}(x)=2P_{UDERP}*\delta(x-n\,TB) + (1-2\,P_{UDERP})$ with $0 \le x \le 1$
t_{122}	$f_{122}(x)=\delta(x)$
t_{123}	$f_{123}(x)=2P_{UDERP,2}*\delta(x-n\,TB) + (1-2\,P_{UDERP,2})$ with $0 \le x \le 1$

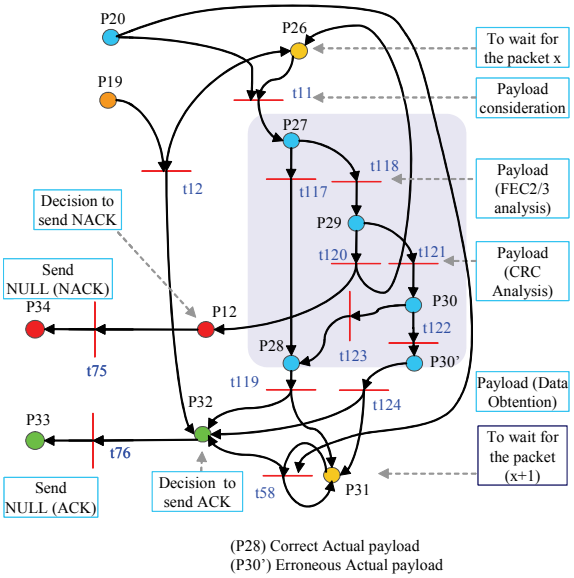

Fig. 6. Receive Petri Net and associated probabilities

E2. Analysis

On the basis of the randomized state graph, and by using the evaluations allowed by the STPN model and its associated tool, we, in particular, can obtain quantitative abstract views. These views allow the assessment of the incorrect payload probabilities p_{DERP} of DM1 and DH1 packets, depending on error probabilities p_e of the wireless medium as illustrated by Fig. 7. Modelling and simulation with STPN tools are explained in detail in publications [6] and [7], research in which the theoretical study is focused on the error probability of DM1 and DH1 packets. The errors on header or/and payload packets act directly upon the behavior of ARQ the mechanism. These theoretical results shown on Fig. 7 can be compared with previously obtained results for DM1 and DH1 data rates shown on Fig. 5, and the FEC used for DM1 confirming the QoS by an error probability not exceeding $6.09E^{-14}$ with $p_e = 10^{-3}$. This allows a reduction in the use of the ARQ protocol.

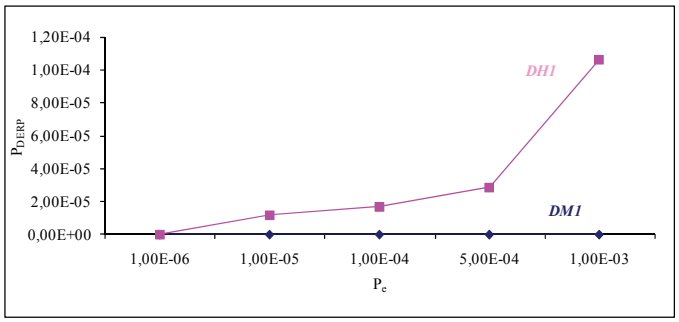

Fig. 7. Incorrect payload probabilities

III. Measurement Platform

In this part we present the whole measurement platform, using the RTAI real-time operating system and the Bluetooth communication system to collect data on time measurements for processing and transmission, radio signal levels and communication data rates.

A. Measurement architecture

The software architecture associated with each hardware entity, illustrated in Fig. 8, is composed of a Linux operating system of the Redhat-8 type, with an RTAI kernel and a real-time Bluetooth module piloting task. The real-time task for piloting the Bluetooth modules carries out an initialization sequence [5] thus allowing the definition of the module type (master or slave) to link the two modules, associating an asynchronous DM1 channel. A DM1 packet can carry up to 17 bytes of useful data.

The quality of service of the bit error rate is provided by the data redundancy implemented by the ⅔ FEC. Moreover data integrity is guaranteed by the CRC16. The maximum theoretical useful data rate is therefore 108.8Kbits/s. Once this initialization stage is carried out correctly, the transmission real-time task becomes periodic and activates the communication protocol we especially designed for the measurement of the Bluetooth communication link. This measurement protocol periodically sends a DM1 packet that must be positively acknowledged. The useful payloads of data and acknowledgement packets contain 17 bytes, corresponding to the maximum size supported by the norm for DM1 packets. The useful payload of acknowledged packets contains a value for the reception radio signal level from the slave module. This value corresponds to the RSSI[9] in dBm

[9] Received Signal Strength Indicator

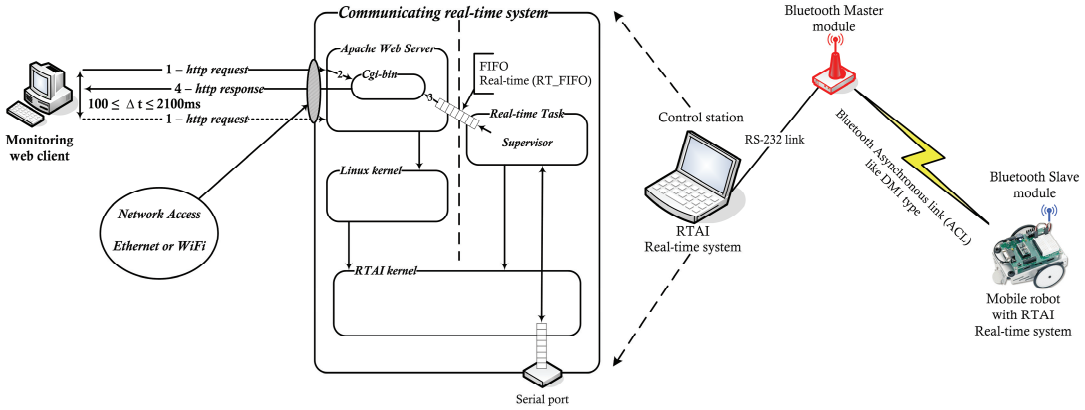

Fig. 8. Measurement platform and software architecture

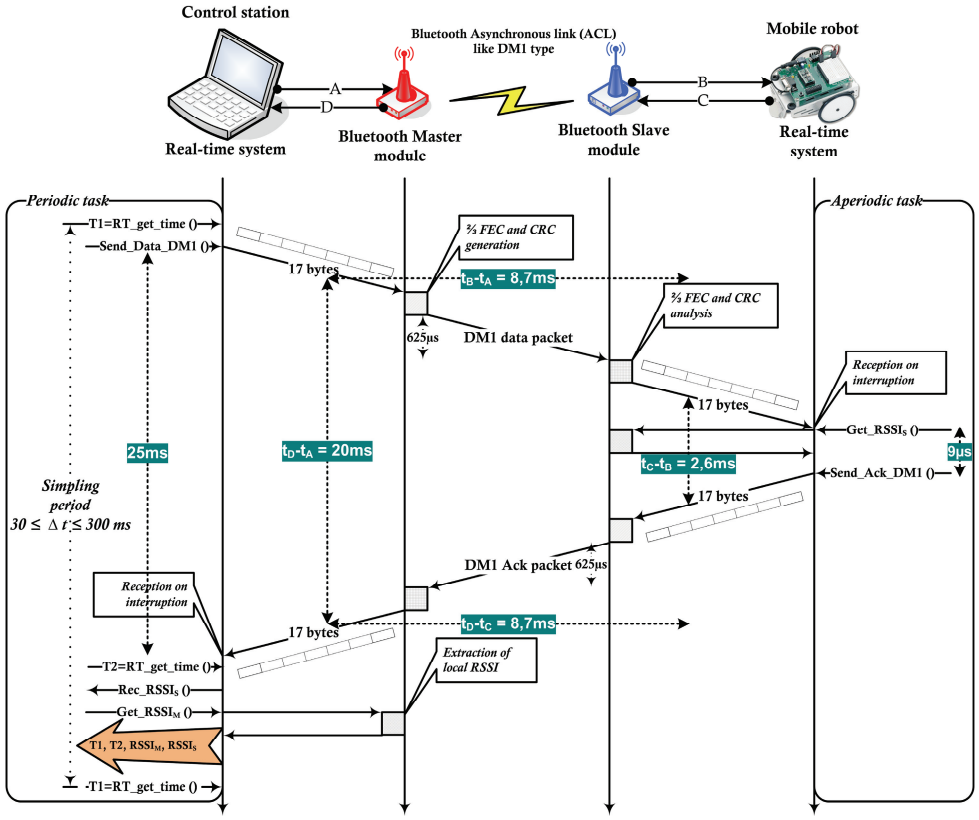

Fig. 9. Measurement protocol

measured on the Bluetooth module on reception of the data packet. Protocol behavior and radio measurements are illustrated by the chronogram in Fig. 9. This protocol is on the LLC3[10] layer. Let us bear in mind that the LLC3 layer corresponds to a constantly reliable and acknowledged transfer. Our protocol uses the Bluetooth layer 2 based on the TDD[11] type access method. We are not analysing here in detail this access method, but only explain briefly its principle for communication with DM1 packets. The time is slotted at intervals of 625µs. The master module uses 1 time slot to communicate with a slave module; the latter uses the following time slot to acknowledge the data from the master. Therefore, a master/slave exchange lasts 1.25ms at best. If the ARQ layer 2 protocol records non-reception of the packet after timer expiration or transmission errors thanks to the CRC, it will automatically undertake retransmissions on the appropriate following time slots. With the available modules for Bluetooth Ericsson, it is impossible to configure the TDD protocol for layer 2. Without going into detail on the intrinsic working of the TDD, we would like to give an indication for the PDU[12] layer 2:

- initially, the ARQ protocol parameters indicate the number of retransmissions desired in the event of error or packet loss,

[10] Logical Link Control
[11] Time Division Duplex
[12] Protocol Data Unit

- secondly, to transmit all the packets, correct or not, to the LLC layer from the LMP layer.

This would have allowed us to have better control over the behavior of the Bluetooth modules, in order to limit the resulting delays in the servo-control loop of the control system. On the other hand, as the error packets are sent up to the LLC layer, it becomes possible to analyze the radio link quality in real-time, and to anticipate a temporary halt of the robot before the radio link breaks and the robot goes out of control. With Bluetooth modules allowing such a parametrization of layer 2, we illustrate in Fig. 10 an example of a servo-control loop with and without automatic retransmission. In the second case we observe the communication length in the servo-control loop is of 2*625µs, whereas in the first case it is twice the value with the possibility that the communication might not be correct on time slots 3 and 4. This second case requires of course that the real-time communicating application undertakes the retransmissions of the bad or lost packets. These retransmissions can be calibrated according to the application, as for example when piloting a moving robot, it may be necessary to give an immobilization instruction if the previous one to turn to the left was not performed. However, the hardware architecture of the Bluetooth communicating modules proves to be limited in terms of real-time performance, because it must be used through the asynchronous RS-232 interface. We present in the conclusion of this paper future ideas for an integrated platform in order to reduce this limitation.

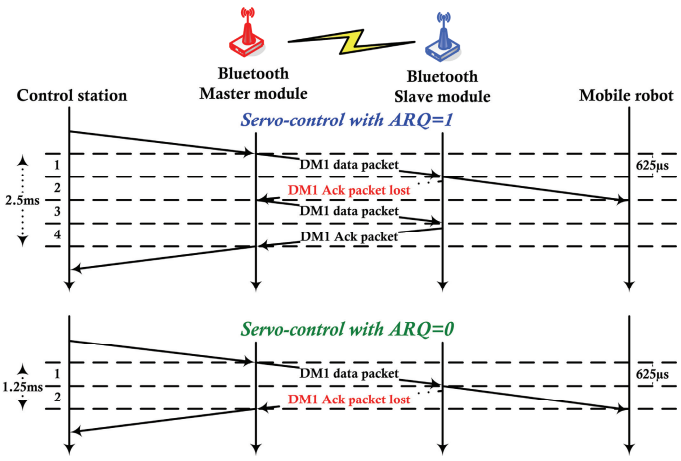

Fig. 10. ARQ servo-control and re-transmission

B. Measurements

After presenting the global environment of the communication platform, we dedicate this part to the explanation of the measurement protocol.

First of all, let us recall that according to the kind of real-time application, a classification of the measurement types is necessary:

- either in deferred time, that is to say in post-execution of the system, or with a time shift between the moment when data is generated and the moment when it is interpreted. This measurement type generally requires data storage and is suited for applications with weak time constraints, also named soft real-time applications.
- either in real-time, that is to say that at the exact moment when data is generated, when it is immediately interpreted, which is the case for applications with strong time constraints or in hard real-time as for the remote robot control application.

For our measurement platform, we used a real-time operating system piloting the master module to collect all the data measured during an exchange between master/slave. These data are then processed and interpreted on the go but never stored.

B1. Measurement protocol

The measurement protocol is based on periodic master/slave exchanges using Bluetooth DM1 packets. The master module is the coordinator for the global measurement system. Its part is therefore to collect its local radio measurements and to create a DM1 packet requesting the remote radio measurement from the slave module. The latter carries out the local radio measurement and creates a DM1 answer packet containing the radio measurement. Thanks to this protocol of radio data collecting, the master module can measure in real-time the performance of the LLC3 link. It deduces the data rate performances of the DM1 link at work. The radio measurements, known as RSSI, carried out on the master and slave modules, are coded in 1 byte each. The master module codes in 4 bytes each time stamp of the real-time system clock. This allows the time measurement of the exchanges right through. At the outcome, the master module deduces the useful throughput of the DM1 link and codes it in 4 bytes. Therefore, at every sampling period, the coordinator of the global measurement system generates 10 bytes of measured data. The problem of selecting a measurement system, among those previously presented, then arises. We didn't choose the measurement type in deferred time, because the real-time task of the master module rates the measuring protocol at a sampling frequency of 25ms, for the processing request and the radio measurements. At such a frequency, the data measured for 1 hour represent 1.44Mbytes to store before processing and analysis. Therefore, the deferred time measurement is not suited for our application, especially for extended observations. We therefore chose real-time measurement initially to carry out the measurements and estimate the communicating system benchmarks, and subsequently, to act on the servo-system from the moment the measurement protocol considers the system to be corrupt.

B2. Data acquisition

Collection of the measured data is carried out by the real-time tasks of the communicating systems and more especially by the measurement coordinators. These measurements must then be transferred to another task of the system, in order to unload the coordinators'. Two fundamental aspects are to be taken into account for the choice of the data transfer mode:

- data processing is asynchronous in relation to data generation,

- the measured data volume is 10 bytes per 25ms period.

We must therefore fix the appropriate transfer mode for these constraints. Among the existing communication methods (signals, file, pipe, message, shared memory, queue …) between the tasks of the real-time operating system, we chose to use a real-time queue (RT_FIFO[13]) because it offers the best compromise in terms of use and performance. It is a communication method ideally suited for inter-task data transfers with a time schedule in the exchanged data. We calibrated this FIFO size at 1024 bytes according to the sampling period of 25ms for data generation and according to the processing sampling period in the mean time of 100ms to 2.1s. In addition, we took into consideration the fact that the data collecting task is asynchronous to that of the coordinator via the FIFO. It is therefore the coordinator's task that manages the total monitoring of the FIFO to avoid overflow in the queue. According to Fig. 8, in order to give maximum flexibility to the graph production of the measured data, we introduce a client application (step 1) through a Web[14] server which is activated by the coordinator of the global system. A non-real-time task such as '*cgi-bin*' has been developed in order to recover the data (step 2) stored in the RT_FIFO (step 3) to send them through the *http* protocol (step 4) toward the client application: this is developed as a Java Applet offering total portability whatever the target client system. The collected data are formatted and finally displayed on a graph.

C. Benchmarks

Thanks to this real-time measurement protocol which is remotely analyzable on a computer with an Internet navigator, we carried out the intrinsic measurements of the Bluetooth communication protocols and those of the current modules. We carried out measurements on the Bluetooth DM1 link throughput by varying the packet size of the application layer.

We observed that data rates increase according to the packet size as in Fig. 11.

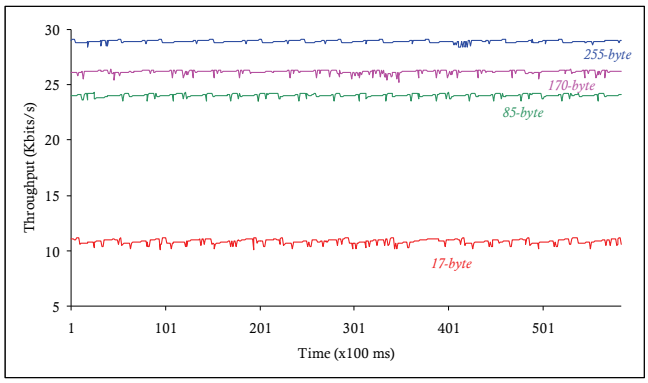

Fig. 11. Throughput with DM1 packets

We confirmed in a quiet radio environment that the minimal useful throughput is about 11Kbits/s, and the maximum is 29Kbits/s. These experimental measurements allowed us to note variable times in byte processing according to their volumes in the radio links. These processing times are composed of the processing time in the real-time tasks as well as the serialization time in the Bluetooth modules. We might observe that the available modules require between 0.13 and 0.32ms processing 1 byte. The processing times of real-time systems are 1000 times smaller and therefore considered as insignificant.

These time measurements are illustrated by Fig. 12. Paradoxically, the greater the load to process bytes, the lower the processing time is, but never under 0.11ms/byte.

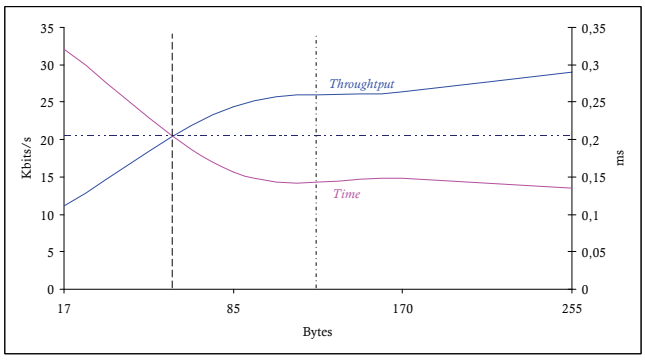

Fig. 12. Processing time by byte and throughput

This is because byte by byte data processing requires a routine execution of the Bluetooth module for every byte. This same routine is used for the processing of several bytes. Therefore we must determine a compromise between the volume of information to be transmitted in every packet and the minimum time to be adhered to validate the servo-control loop for control and command. According to the curves in Fig. 12, performance is optimal around 120 bytes. From the measurement protocol, Fig. 9, and the 17-byte data packets, the recorded measurements, Fig. 13, bring forward a minimum processing time of 24,327ms. That is the minimum time necessary for the master module to obtain an answer to a request, i.e the 34-byte transmission. In established communications and without disturbance, we notice that the automatic repetitions of the ARQ protocol are regularly in demand, for ARQ=1 and ARQ=2 values.

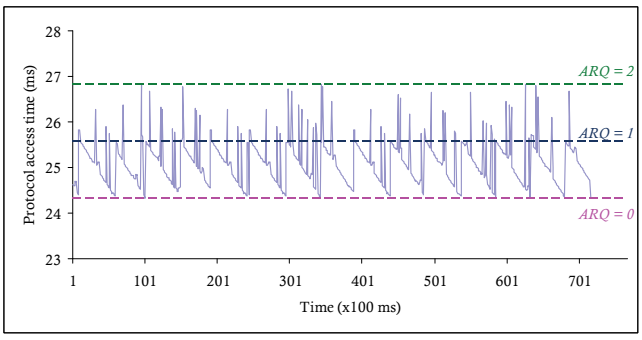

Fig. 13. Access time of the measurement protocol and ARQ influence

This leads to time increases of 2*625µs and 4*625µs respectively on the Bluetooth radio transmission. The ARQ protocol behavior is not suited for to control and command of real-time tasks, because it is a non-deterministic mechanism (in terms of repetitions) and induces considerably greater

[13] Real-Time – First In First Out
[14] A no real-time Apache process

times in the servo-control loop as presented in Fig. 10. The real behavior measured confirms our hypotheses, and would require total control and mastery of the ARQ protocol, which is not the case with our Bluetooth modules. In Fig. 9, we have specified the detailed transmission times, and we can deduce that the serialization time of each DM1 packet in each Bluetooth module is 4ms in agreement with the result in Fig. 3. The measurement platform interface is illustrated in Fig. 14, which represents the measurements carried out for a packet size of 255 bytes. The times measured by the coordinator are about 140,32ms for a maximum throughput of 29Kbits/s. With voluntary disruptions of the system, by antenna obstructions, we notice meaningful variations in the RSSI radio measurements, sometimes resulting in reductions in the data rate.

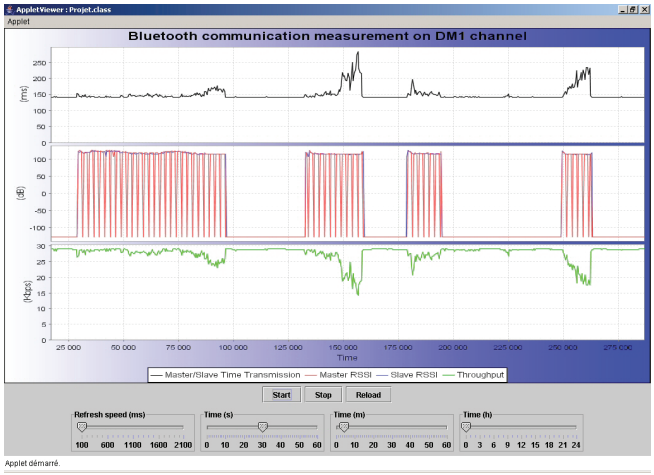

Fig. 14. Measurement platform interface

Multiple measurements have allowed us to notice that radio fading voluntarily generated near the antenna Bluetooth modules, immediately triggered variations in the RSSI before communication quality is affected. This observation is fundamental for the control and command of remote robot because it allows the prediction of a possible corruption of the radio link, and therefore the operator can take appropriate measurements. This phenomenon can be integrated into the servo-control loop of the system to anticipate delays generated by corruption of the communication link.

IV. Conclusion

We have presented our research work on the measurements of processing times of the DM1 asynchronous Bluetooth link managed by real-time systems. We have put forward the influence of packet size on the throughput and ARQ protocol. With regard to this phenomenon, we have estimated the processing time of one byte in the Bluetooth modules as well as the necessary crossing times of the software layers of RTAI real-time systems. The real-time radio measurements on the module master and slave are essential information for the operation of a control and command protocol of a robot integrating time delays. This represents a way of forecasting the communication state. The processing times of the DM1 packets by Bluetooth modules are distinctly superior to radio transmission times. This is due to the weak performance of the hardware architecture of the communicating modules. The technological developments of high data rate interfaces have progressed in terms of performance and integration offering more effective Bluetooth communicating systems (v2.0, v2.1, +EDR) and integrating instruction interfaces: this is particularly true for audio and video streaming transports between WAN access equipment and mobile phones with hard time contraints.

References

[1] Bluetooth SIG, « Specification of the Bluetooth system: Core V1.0 », 1999.

[2] Bluetooth SIG, « Specification of the Bluetooth system: Core V1.2 », 2001.

[3] Das A., Ghose A., Razdan A., Saran H., Shorey R., « Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network », Proceeding of IEEE INFOCOM'01, USA, 2001.

[4] Frederick M., William P. Shackleford, « Timing Studies of Real-Time Linux for Control », Real-Time Linux Workshop, October 2001, Milan – Italy.

[5] Khoutaif T., Peyrard F., « Performances evaluation of the asynchronous Bluetooth links in a real-time environment », IEEE PWC'05 (Personal Wireless Communications), ISBN 1-86094-582-1, pp. 235-243, Colmar, France. Août 2005.

[6] Khoutaif T., Juanole G., « Formal Modelling and Evaluation of the Data Transfer Phase of the ACL links on the WPAN Bluetooth », ETFA 2006, 11th IEEE International Conference on Emerging Technologies and Factory Automation, September 20-22, 2006, Prague, Czech Republic.

[7] Khoutaif T., Peyrard F., « Reliability stydy of a data transfert protocol with multi-redondancies packets », FET'07, 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems, november 7-9, 2007 ,Toulouse, France.

[8] Law C., Mehta A. K., Siu K-Y., « A New Bluetooth Scatternet Formation Protocol », Mobile Networks and Applications Journal, Volume 8, Number 5 / October, 2003.

[9] Mantegazza P., Bianchi E., Dozio L., Ghiringhelli G.L., « Complex control system, Applications of DIAPM-RTAI at DIAPM », Real-Time Workshop, Vienna, Austria, 1999.

[10] Mantegazza P., Renoldi G., « Real-Time Application Interface documentation of Serial Port Drivers », RTAI 2002.

[11] Peyrard F., Val T., « Simulation et analyse de performances de liens synchrones Bluetooth avec Opnet », Colloque Francophone sur l'Ingénierie des Protocoles CFIP'2002, Montréal, 27-30 mai, 2002.

[12] RTAI Programming Guide1.0, DIAPM, September 2000.

[13] Val T., Fraisse P., Andreu D., « Vers l'utilisation de Bluetooth pour la commande à distance de robots mobiles », Revue International JESA, Vol. 37 – n° 7-8/2003, pp. 859-892, January 2004.

Fabrice PEYRARD obtained his PhD in Computer Sciences in 1998. He is scientific researcher in the LATTIS laboratory of Toulouse University in the field of wireless local network communications. He is a professor assistant at the IUT of Blagnac (France) and lectures in Network and Telecommunications.