Trabalho

Título em Português:	Controladores para sistemas embarcados de átomos frios
Título em Inglês:	Controllers for cold atoms embedded systems
Autor:	Anderson Eduardo de Carvalho Filho
Instituição:	Universidade de São Paulo
Unidade:	Instituto de Física de São Carlos
Orientador:	Daniel Varela Magalhães
Área de Pesquisa / SubÁrea:	Física Atômica e Molecular
Agência Financiadora:	USP - Programa Unificado de Bolsas

Controladores para sistemas embarcados de átomos frios

Anderson Eduardo De Carvalho Filho

Prof. Dr. Daniel Varela Magalhães

Instituto de Física de São Carlos/Universidade de São Paulo

anderson.carvalho52@usp.br

Objetivos

O presente trabalho tem como foco o estudo de técnicas de sincronização entre dispositivos embarcados heterogêneos, tomando como referência a comunicação entre uma BeagleBone Black Rev C[1], um Arduino Nano[2] e um notebook que atua como supervisor. O objetivo é avaliar como diferentes protocolos de comunicação em especial o CAN, comparado ao UART e ao I2C se comportam em termos de latência, jitter e robustez, dentro de um cenário de testes plausível em laboratório. Essa análise busca não apenas compreender as limitações práticas de cada abordagem, mas também fornecer subsídios para a concepção de arquiteturas de controle distribuído aplicáveis a experimentos embarcados mais complexos, como os envolvidos em sistemas de átomos frios.

Métodos e Procedimentos

A BeagleBone Black foi configurada como unidade principal, responsável por gerar pulsos de sincronização e atuar como mestre em cada protocolo testado. O Arduino Nano desempenhou o papel de nó secundário, respondendo aos sinais enviados e reportando o tempo de processamento. O notebook, conectado simultaneamente a ambos, assumiu a função de supervisor, registrando dados de latência, analisando o jitter e contabilizando falhas de comunicação.

Foram considerados três cenários distintos. No primeiro, utilizou-se comunicação serial UART a 115200 bps, de forma ponto a ponto entre BeagleBone e Arduino. No segundo, adotou-se o barramento I2C a 100 kHz, no qual a BeagleBone atuou como mestre e o Arduino como escravo. No terceiro e principal cenário, implementou-se o protocolo CAN operando a 1 Mbps, com a BeagleBone utilizando o controlador DCAN interno e o Arduino Nano acoplado a um módulo MCP2515 com transceptor dedicado. Esse arranjo permitiu observar o comportamento de um barramento mais robusto, adequado a sistemas distribuídos.

As medições foram realizadas a partir de sinais de teste gerados pela BeagleBone e confirmados pelo Arduino, enquanto scripts em Python no notebook fizeram a coleta e análise estatística. Para cada protocolo foram extraídas séries temporais de latência, valores médios, desvios padrão e taxas de erro em aproximadamente sessenta amostras por caso.

Resultados

Os resultados mostraram diferenças claras no desempenho dos protocolos. O UART apresentou uma latência média próxima de 2,0 ms, com jitter em torno de 0,3 ms e uma taxa de erro inferior a 0,3%. Esse comportamento confirma a eficiência do enlace serial para comunicações diretas e relativamente estáveis, embora sem recursos adicionais de confiabilidade.

No caso do I2C, a latência média elevou-se para cerca de 4,6 ms, com jitter próximo de 0,7 ms e uma taxa de erro próxima de 1%. Esses números refletem as limitações inerentes ao clock mais baixo e ao overhead de endereçamento do protocolo, além de sua maior sensibilidade a ruídos e perdas ocasionais de sincronismo.

O CAN, por sua vez, apresentou o desempenho mais equilibrado. A latência média registrada foi de aproximadamente 1,2 ms, com jitter reduzido em torno de 0,2 ms e uma taxa de erro praticamente nula (0,05%). Esses resultados se explicam pela robustez do protocolo, que conta com arbitragem por prioridade, retransmissão automática em caso de falhas e mecanismos internos de detecção de erros. Mesmo com cerca de 25% de carga de barramento, o sistema manteve estabilidade temporal superior aos demais protocolos.

O gráfico gerado reforça essa conclusão: o CAN apresenta a série temporal mais estável, seguido pelo UART, enquanto o I2C evidencia maior dispersão.

Além dos aspectos quantitativos observados, é importante considerar o contexto prático de cada protocolo. A escolha não depende apenas de latência e erro, mas também da complexidade da aplicação, da escalabilidade do sistema e da integração com outros dispositivos. Protocolos simples, como o UART, são suficientes para sistemas de pequena escala, enquanto o I2C, apesar de maior latência, permite interligar múltiplos sensores com poucas linhas. O CAN, com sua robustez e baixa taxa de erro, é ideal para sistemas críticos e distribuídos, onde confiabilidade e previsibilidade temporal são essenciais. Assim, os resultados devem ser interpretados tanto em termos de desempenho quanto das demandas específicas de cada aplicação.

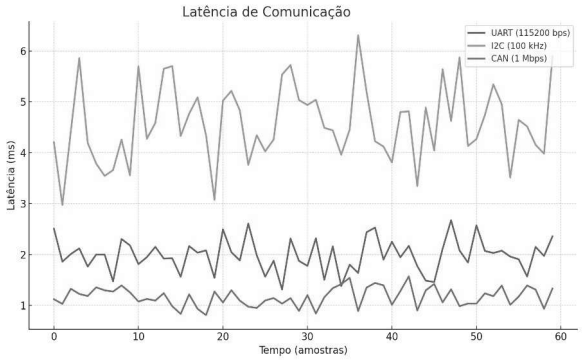

Figura 1: Gráfico de Comparação de desempenho

Conclusões

. A análise demonstrou que todos os três protocolos são capazes de realizar a sincronização entre BeagleBone, Arduino e notebook, mas com desempenhos bastante distintos. O CAN destacou-se como a solução mais robusta, oferecendo baixa latência, pequeno jitter e praticamente nenhuma perda, o que o torna ideal para sistemas embarcados que exigem previsibilidade e confiabilidade. O UART mostrou-se competitivo em termos de latência, sendo uma opção válida para comunicações ponto a ponto de menor complexidade, embora sem os recursos de priorização e redundância do CAN. Já o **I2C** revelou limitações marcantes, devendo ser reservado a tarefas secundárias que não exijam alta precisão temporal.

Esses resultados sugerem que uma arquitetura híbrida, combinando o CAN como barramento crítico de sincronização e protocolos como UART ou I2C para comunicação auxiliar, pode oferecer um equilíbrio adequado entre simplicidade, desempenho e confiabilidade. Esse arranjo é particularmente promissor para supervisão distribuída de subsistemas em experimentos de referência temporal, alinhando-se às necessidades previstas no plano de trabalho original.

Referências

[1] **BEAGLEBOARD.** BeagleBoard.org. Disponível em: https://www.beagleboard.org/boards/beaglebone-black. Acesso em: 10 ago. 2025.

[2] **ARDUINO.** Arduino.cc. Disponível em: https://www.arduino.cc/en/hardware/#nano-family. Acesso em: 10 ago. 2025.

Controllers for Embedded Cold Atom Systems

Anderson Eduardo de Carvalho Filho

Prof. Dr. Daniel Varela Magalhães

Institute of Physics of São Carlos / University of São Paulo

anderson.carvalho52@usp.br

Objectives

This work focuses on the study of synchronization techniques between heterogeneous embedded devices, taking as reference the communication among a BeagleBone Black Rev C [1], an Arduino Nano [2], and a notebook acting as a supervisor. The objective is to evaluate how different communication protocols particularly CAN, compared to UART and I2C behave in terms of latency, jitter, and robustness within a plausible laboratory testing scenario. This analysis seeks not only to understand the practical limitations of each approach but also to provide insights for the design of distributed control architectures applicable to more complex embedded experiments, such as those involved in cold atom systems.

Materials and Methods

The BeagleBone Black was configured as the main unit, responsible for generating synchronization pulses and acting as the master in each tested protocol. The Arduino Nano played the role of secondary node, responding to the signals sent and reporting processing time. The notebook, connected simultaneously to both, assumed the supervisory role, recording latency data, analyzing jitter, and logging communication failures.

Three distinct scenarios were considered. In the first, UART serial communication at 115200 bps was used, in a point-to-point connection between BeagleBone and Arduino. In the second, the I2C bus at 100 kHz was adopted, with the BeagleBone as master and the Arduino as slave. In the third and main scenario, the CAN protocol was implemented, operating at 1 Mbps, with the BeagleBone using its internal DCAN controller and the Arduino Nano connected to an MCP2515 module with a dedicated transceiver. This setup enabled observation of the behavior of a more robust bus, suitable for distributed systems.

Measurements were carried out from test signals generated by the BeagleBone and confirmed by the Arduino, while Python scripts on the notebook performed data collection and statistical analysis. For each protocol, time series of latency, mean values, standard deviations, and error rates were obtained from approximately sixty samples per case.

Results

The results showed clear differences in protocol performance. UART presented an average latency close to 2.0 ms, with jitter around 0.3 ms and an error rate below 0.3%. This behavior confirms the efficiency of the serial link for direct and relatively stable communications, although lacking additional reliability features.
For I2C, the average latency rose to about 4.6 ms, with jitter near 0.7 ms and an error rate

around 1%. These numbers reflect the inherent limitations of its lower clock speed and addressing overhead, as well as its greater sensitivity to noise and occasional synchronization losses.

CAN, in turn, showed the most balanced performance. The average latency recorded was approximately 1.2 ms, with reduced jitter around 0.2 ms and an almost negligible error rate (0.05%). These results are explained by the robustness of the protocol, which includes priority-based arbitration, automatic retransmission in case of failures, and internal error detection mechanisms. Even with about 25% bus load, the system maintained superior temporal stability compared to the other protocols.

The generated graph reinforces this conclusion: CAN shows the most stable time series, followed by UART, while I2C displays greater dispersion.

In addition to the quantitative aspects, it is important to consider the practical context of each protocol. The choice depends not only on latency and error rates but also on application complexity, system scalability, and integration with other devices. Simple protocols like UART are sufficient for small-scale systems, while I2C, despite higher latency, enables interconnection of multiple sensors with few lines. CAN, with its robustness and low error rate, is ideal for critical and distributed systems where reliability and temporal predictability are essential. Thus, results should be interpreted both in terms of performance and in relation to the specific demands of each application.

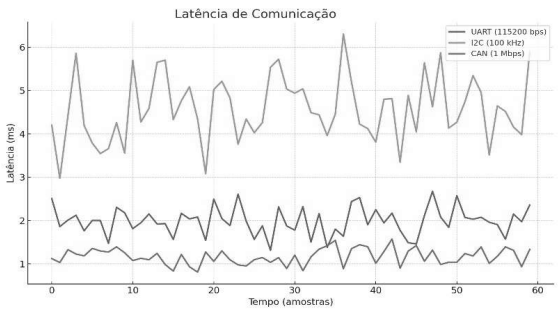

Figure 1: Performance Comparison Graph

Conclusions

The analysis demonstrated that all three protocols are capable of synchronizing BeagleBone, Arduino, and notebook, but with markedly different performances. CAN stood out as the most robust solution, offering low latency, minimal jitter, and virtually no loss, making it ideal for embedded systems that demand predictability and reliability. UART proved competitive in terms of latency, being a valid option for simpler point-to-point communications, though lacking the prioritization and redundancy features of CAN. I2C, on the other hand, revealed significant limitations and should be reserved for secondary tasks that do not require high temporal precision.

These results suggest that a hybrid architecture, combining CAN as the critical synchronization bus and protocols like UART or I2C for auxiliary communication, may provide an appropriate balance between simplicity, performance, and reliability. This arrangement is particularly promising for distributed supervision of subsystems in time-reference experiments, aligning with the needs outlined in the original work plan.

References

[1] **BEAGLEBOARD.** BeagleBoard.org. Available at: https://www.beagleboard.org/boards/beaglebone-black. Accessed on: Aug. 10, 2025.

[2] **ARDUINO.** Arduino.cc. Available at: https://www.arduino.cc/en/hardware/#nano-family. Accessed on: Aug. 10, 2025.