LoggerBIT: An Optimization of the OpenLog Board for Data Logging with Low Cost Hardware Platforms for Biomedical Applications

Margarida Reis[1] and Hugo Plácido da Silva[1,2]
[1]*IT - Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal*
[2]*Polytechnic Institute of Setúbal, Campus do IPS, Estefanilha, 2914-508 Setúbal, Portugal*

Keywords: Biomedical Devices, Data Logging, Biosignal Acquisition, Real-time Applications.

Abstract: Low cost hardware and open source software tools specialized in biomedical applications have recently seen a significant growth both in popularity and diversity. Within the current landscape of such tools, BITalino is increasingly used by students and professors for educational activities and prototyping, due to its modular design and flexibility. However, there is currently no user-friendly way of storing data locally without setting up a wireless connection to a receiver unit (e.g. a computer or smartphone). This poses a problem, especially in uses cases where ambulatory data acquisition is needed. In this paper we present the LoggerBIT, a data logging block to enable data recording onto a microSD card in biomedical development toolkits, without requiring a connection to a receiver nor any particular programming or electronics skills to setup. Our work builds upon the OpenLog serial data logger, to which multiple optimisations were made in terms of hardware and firmware, overcoming limitations preventing its use in high data throughput applications. Our LoggerBIT approach is suitable for high-speed data recording, with results showing that it can log data of up to 4 analog channels at 1000 Hz for 24 hour periods without loss of packets, and from all analog channels with minimal data loss. This work contributes to the state-of-the-art with a comprehensive description and validation of the block, and with a set of supporting tools released in open source, thus available for the biomedical engineering community at large.

1 INTRODUCTION

In the last decade, purpose-built biomedical hardware platforms to support learning and prototyping following a low cost and open source approach have become a staple in the field, as they are not limited by proprietary hardware or software. Examples of such platforms include the OpenBCI (Russomanno, 2014)(Frey, 2016) and BITalino (Silva et al., 2014), the latter of which has been reported as one of the most complete and versatile tools for biofeedback and quality of life research (even when compared with the latest wearables), hence being particularly popular (Nogueira et al., 2017).

However, existing platforms are mostly designed around the concept of real-time data streaming to a receiver, which can be quite restrictive. In particular, for data acquisition in ambulatory settings, streaming has a direct impact on the battery lifetime (both of the transmitter and of the receiver) due to the power required by the radio. It also requires an additional complexity on the receiver, for tasks such as managing reconnections when the device goes out of range. In cases where a memory card is available, this contributes to a higher energy consumption of the overall device, and has a direct impact on the achieavable form factor size when prototyping wearable devices.

In this work we present LoggerBIT, which consists of a hardware and firmware optimization to a commercially available data logger, devised to enable its use in high data throughput applications. This is targeted at eliminating the need for a radio transceiver and the dependency on a receiver in a user-friendly way, when working with low cost and open source biomedical platforms. We will use BITalino as a test bed for benchmarking, due to its widespread use. The remainder of the paper is organized as follows: Section 2 provides the background, Section 3 describes the proposed solution, Section 4 presents experimental results on the performance of LoggerBIT in different testing conditions, and finally Section 5 outlines the main conclusions and future work perspectives.

Reis, M. and Silva, H.
LoggerBIT: An Optimization of the OpenLog Board for Data Logging with Low Cost Hardware Platforms for Biomedical Applications.
DOI: 10.5220/0006872306000604
In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2018) - Volume 2*, pages 600-604
ISBN: 978-989-758-321-6

2 BACKGROUND

Data logging is a common problem in many applications, which has lead to the creation of several general-purpose accessories. However, these are primarily designed for use cases of relatively low data data rates, and where occasionally missing data is not troublesome.

The OpenLog is one such accessory, based on the ATmega328P 8-bit MCU and that can interfaced with via a standard Universal Asynchronous Receiver-Transmitter (UART) port. It supports an operating voltage between 3.3-5.0V and has a current consumption of approximately 5mAh while in idle state with the SD card inserted, and of approximately 20mAh when writing to the SD card. These specifications mean that it can be directly powered from the most common biomedical platforms, and interfaced with them via the UART.

A problem with this device is that there may be dropped bytes with baud rates equal or higher than 57600bps (Seidle, 2014), which is a severe limitation for its use with biomedical data acquisition hardware, where the combination of the number of channels and sampling rate can easily reach baud rates of 115200bps. The class of card can help reduce and/or eliminate this problem, although without guarantees of reliability using the existing OpenLog hardware and firmware options.

3 IMPLEMENTATION

Changes were made mainly to the method for configuring the acquisition settings and management of the read/write buffers to handle at least 115200 bps baud rates in continuous logging mode. These will be described throughout the following sections.

3.1 User-defined Configurations

The OpenLog is primarily designed to be controlled via commands sent through the UART from an MCU connected to it (e.g.). In LoggerBIT the user is able to define the acquisition settings (e.g., sampling rate, channels to be acquired, etc.), by placing a CSV configuration file named `config.txt` at the root directory of the SD card. The configuration file has two lines, one for the settings and the other for providing a human-readable description of their definitions, as can be seen in the following snippet.

```
1000,live,123456,00,00
sampling rate,mode,channels,trigger,
    digital IO
```

3.2 Recording Loop

The main task of the recording loop is the retrieval of a data stream from the UART, writing to the log file and occasionally synchronizing the SD card (i.e. flushing the buffer). We characterized the operational demand of each task by measuring their completion time, determining that, in average, writing approximately 128 bytes (the size of the read buffer) to the file takes approximately 50uS, whereas the synchronization lasts close to 5ms.

A major bottleneck is therefore having to periodically synchronize the card; however, this is a necessary step to ensure that any bytes written to the file are logically visible within the file system. In addition, it needs to be performed periodically rather than on the end of the recording, given that our goal for the LoggerBIT is that the user can stop the recording at any time, by simply powering off the device without prior announcement or action.

As such, we focused our efforts on improving this process. When the file is being synchronized to the SD card, LoggerBIT is blocked on that operation, meaning that it will not be able to receive any incoming data, potentially leading to a loss of packets. With the flow-control mechanism, LoggerBIT signals the transmitter to stop streaming and buffer the data internally before the synchronization, and to resume the streaming afterwards; in-between, LoggerBIT has enough time to synchronize the file. We also modified the synchronization cycle to have the synchronization done based on the number of bytes written instead of time and removed any calls that would put the MCU in sleep mode, both being the standard approach followed by the OpenLog.

3.3 Hardware Flow-control

Our proposed approach to mitigate the byte loss issue is based on implementing UART hardware flow-control, where LoggerBIT requests the transmitter to stop sending data while it is executing its critical, i.e., time consuming operations. By default, the OpenLog board does not have a `RTS` line, as the flow-control mechanism requires; as such, we changed both the hardware and firmware of the OpenLog to have one GPIO working as the `RTS` line (Figure 1).

Having the OpenLog performing the synchronizations based on the number of written bytes instead of time (as previously described), provides us with better control of the timming for card synchronization. When the synchronization is happening, the `RTS` line is set to the *HIGH* level, signaling the transmitter to stop sending data and start buffering it internally.

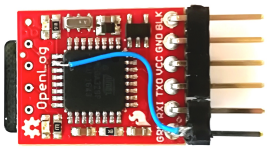

Figure 1: OpenLog with UART hardware flow-control line support.

Once the synchronization is completed, the `RTS` line is set to *LOW*, signaling the transmitter to resume the data streaming.

4 EXPERIMENTAL RESULTS

As previously mentioned, we used the BITalino platform as test bed for benchmarking, due to its characteristics and growing adoption. To evaluate the performance of the LoggerBIT we tested the loss of samples for different configurations of the memory card, number of channels to acquire and sampling rate, analyzed the acquisition reliability, and performed a signal distortion test against a common use case of streaming over standard Bluetooth.

4.1 Sample Loss

Our LoggerBIT approach was benchmarked against the standard OpenLog minimal firmware. To better understand the effects of the hardware flow-control mechanism we also evaluated a version of LoggerBIT that does not use hardware flow-control. The evaluated metrics were the total number of CRC fails and the total estimated number of lost packets associated with those fails, based on the sample sequence number.

We started by testing data acquisition sessions with 8h duration, for which the results are presented in Table 1. The tests were done until a zero loss configuration of the system was found for each version and for each card, starting from the most demanding use case (acquisition of 6 channels at 1kHz) and descending to the less demanding use case (which would be the acquisition of 1 channel at 1Hz). Analyzing first the Class 3 card, with the highest write speed (UHS-I U3 PRO), we can see that, for the original (OpenLog) version of the firmware and without hardware flow-control, there was loss of samples using the most demanding acquisition settings. For both versions, we observed that this firmware and card combination can be used reliably for 8h periods, to acquire 4 channels at 1kHz and 6 channels at 100Hz.

It is important to note that, even though the firmware without flow-control has registered sample loss, there was approximately a 50% reduction in the number of CRC fails and associated number of lost packets when compared with the original version. This confirms that the option to enter sleep mode when the SD card synchronization takes place (followed in the original firmware), introduces a significant overhead when dealing with high throughput data received via the UART. As for the LoggerBIT firmware, this test demonstrates how incorporating the hardware flow-control mechanism improves the performance, considering that it is capable of being used in the most demanding configuration possible for the BITalino device, without any CRC failure.

The card within the same class rating but with a lower write speed (UHS-I U3) only worked without any losses in the most demanding configuration with the LoggerBIT firmware. For the remaining versions, the card proved to be stable only for the same configurations previously mentioned (4 channels at 1kHz and 6 channels at 100Hz). For acquiring all channels at the maximum sampling rate, this card had more CRC fails than the one previously examined, which is likely associated with the lower writing speed.

As can be observed, the card with the lowest speed class rating (UHS-I U1) is also the less reliable. We started by testing it with the LoggerBIT firmware at the most demanding configuration, and progressively reduced the number of channels to 2 until no data loss events occurred while recording data at 1kHz. Because LoggerBIT is the most effective version of the firmware, we only started to test the remaining versions also from 2 channels, since other channel configurations would certainly exhibit higher data loss. For both firmwares, sampling 2 channels at 1kHz also appears to be a stable configuration with no associated CRC failures. As for the possibility to log the maximum number of channels, this card is capable of doing so at 100Hz for all versions of the firmware.

Once we confirmed that the LoggerBIT approach is the best performing one, we decided to test its reliability in long term recordings, by doing a 24-hour long test of 6 channels at 1kHz using the identified optimal SD card (UHS-I U3 PRO). We ran multiple iterations of this test and verified that there are always CRC fails (between 2 to 5), which is fairly acceptable, but led us to determine a more reliable configuration. As such, we tested the acquisition of 4 channels at 1kHz, consistently obtaining recordings for 24h without any data loss, which enabled us to conclude that this is the most stable configuration and the one to be used by default for long-term logging.

Table 1: Performance of the LoggerBIT while continuously recording data for 8h.

		Firmware Version / Card Type / # of Channels															
		Standard						Without hardware flow-control						With hardware flow-control			
	Sampling	UHS-I U1		UHS-I U3		UHS-I U3 PRO		UHS-I U1		UHS-I U3		UHS-I U3 PRO		UHS-I U1		UHS-I U3	UHS-I U3 PRO
	Rate [Hz]	2 CH	6 CH	4 CH	6 CH	4 CH	6 CH	2 CH	6 CH	4 CH	6 CH	4 CH	6 CH	2 CH	6 CH	6 CH	6 CH
Decoding	100	-	2.84	-	2.85	-	2.68	-	2.92	-	2.36	-	2.32	-	3.06	-	-
time [min.]	1000	21.44	-	22.59	40.31	24.57	39.07	22.63	-	23.41	35.13	23.09	32.38	20.19	38.67	24.02	26.97
# of CRC	100	-	0	-	0	-	0	-	0	-	0	-	0	-	0	-	-
fails	1000	0	-	0	13	0	6	0	-	0	4	0	2	0	34	0	0
# of lost	100	-	0	-	0	-	0	-	0	-	0	-	0	-	0	-	-
packets	1000	0	-	0	97	0	38	0	-	0	32	0	15	0	227	0	0

4.2 Acquisition Reliability

To test the acquisition reliability we used the most stable configuration previously determined, connected BITalino to a function generator and injected a square wave with a frequency of 1Hz, peak-to-peak amplitude of 3V and DC offset of 150mV into one of the analog channels. The goal of this test is to compare basic statistics of the parameters of the generated function against the data actually recorded to the SD card by the LoggerBIT. During an 8h recording, the average pulse width was approximately 502.2 ± 0.45ms, indexing a maximum of 504ms and a minimum of 498ms.

We repeated a similar test, but instead of using the function generator, we connected the `RTS` line of the LoggerBIT to one of the digital input pins on BITalino. This way, we can evaluate the effective time of writing to the SD card, i.e., measure the synchronization process. Results during an 8h test showed that the average synchronization takes approximately 5.2 ± 1.13ms, with a recorded maximum of 79ms and minimum of 4ms.

$$RMSE(x,y) = \sqrt{\frac{\sum_{k=1}^{n} (x[k] - y[k])^2}{n}} \tag{1}$$

$$SS_{xx} = \sum_{k=1}^{n} x[k]^2 - n\mu_x^2 \tag{2a}$$

$$SS_{yy} = \sum_{k=1}^{n} y[k]^2 - n\mu_y^2 \tag{2b}$$

$$SS_{xy} = \sum_{k=1}^{n} x[k]\,y[k] - n\mu_x\mu_y \tag{2c}$$

$$R^2 = \frac{SS_{xy}^2}{SS_{xx}SS_{yy}} \tag{2d}$$

4.3 Signal Distortion

Lastly, we tested LoggerBIT in a common usage scenario, that is, recording real-world physiological signals. For experimental comparison, we performed simultaneous acquisition, with optical synchronization, of an Electrocardiography (ECG) signal sampled at 1kHz while it was being recorded by the LoggerBIT from one BITalino, and also by a second BITalino with standard Bluetooth connectivity, streaming all data to a computer. We analized our data based on the methodology used in (Batista et al., 2017), where the suggested comparison metrics to quantify the similarity between acquisitions are the RMSE (Equation 1) and the coefficient of determination R^2 (Equation 2, where μ corresponds to the data mean).

We synchronized the ECG signals by removing the initial temporal offset, with the support of data collected by optical sensors in both devices. The synchronized ECG signals were normalized, filtered with a 300^{th} order bandpass FIR filter with cutoff frequencies of 3Hz and 45Hz, and segmented using the algorithm by Hamilton (Hamilton, 2002) as implemented in the BioSPPy[1] toolbox.

Afterwards, we aligned the segmented heartbeat waveforms by the R-peak between both ECG signals and, for each paired segment, we calculated the RMSE and the coefficient of determination R^2 of the corresponding templates. Figure 2 shows part of the time domain representation of the segmented heartbeat waveforms as extracted from the ECG signals acquired by the different devices.

We concluded this morphological comparison by aggregating the results per segment in an single value, in the form of the overall average and standard deviation across all segments, for each comparison metric. As such, for the RMSE the result was 0.00715 ± 0.00195 and for the coefficient of determi-

[1] https://github.com/PIA-Group/BioSPPy

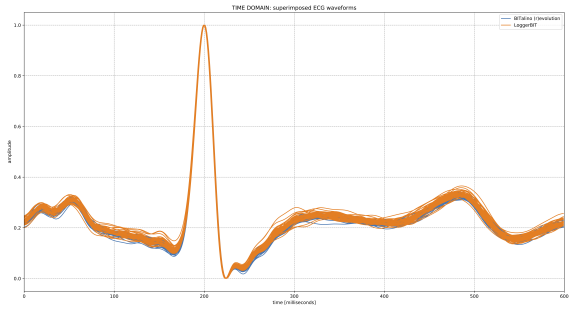

Figure 2: Partial time domain representation of the superimposed segmented heartbeat waveforms as they were recorded by the BITalino (r)evolution and the LoggerBIT.

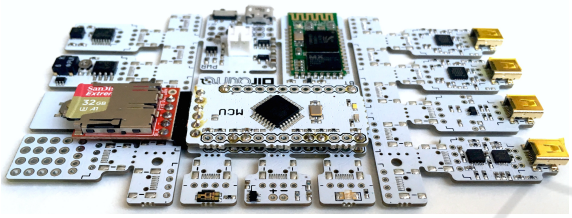

Figure 3: BITalino (r)evolution board in a dual MCU setup allowing simultaneous Bluetooth streaming and data logging with LoggerBIT.

nation R^2 it was of 0.984 ± 0.0194. As we can see, there is a high correlation between both ECG signals, as the value for the RMSE is low and the value for R^2 is high and close to 1. This allows us to demonstrate that the signals acquired through LoggerBIT are in agreement with those collected using the standard approach.

5 CONCLUSIONS

In this paper we presented LoggerBIT, a hardware and firmware optimization of the OpenLog serial data logger, with the goal of making it usable with low cost and open source biomedical data acquisition platforms with the best performance possible. We then evaluated the performance of our approach, using the BITalino platform as a test bed, for different variations of the data acquisition settings (i.e. channels and sampling rate) and SD cards with diverse speed class and write speed ratings. Figure 3 shows an example deployment of LoggerBIT on our test bed.

Results show that the LoggerBIT can operate reliably in the most demanding configuration of BITalino (6 channels at 1kHz) using Class 3 UHS-I U3 PRO cards. In 8h recordings no data loss events appear to occur, and for longer periods minimal data loss is detected (e.g. 2 to 5 packets loss over a 24h period), outperforming the other approaches under testing. In terms of battery life, our tests have shown that the LoggerBIT is able to operate for approximately 40h using a 3.7V 700mAh battery, which is a significant gain when compared with the approximately 12h of operation in the Bluetooth streaming approach.

Future work will focus on further characterizing the overall LoggerBIT architecture in order to prevent packet loss as the memory card has more space occupied (as shown by the 24h tests) and on benchmarking our approach using commonly used biomedical platforms other than BITalino as a test bed.

ACKNOWLEDGMENTS

This work is funded by FCT/MEC through national funds and when applicable co-funded by FEDER PT2020 partnership agreement under the project UID/EEA/50008/2013 “SmartHeart”. The developments herein described have been publicly released and made available to the community at large[2], in alignment with the open hardware and open source software movement.

REFERENCES

Batista, D., Silva, H., and Fred, A. (2017). Experimental characterization and analysis of the BITalino platforms against a reference device. In *Proc. IEEE Eng. in Medicine and Biology Society Annual Int’l Conf.*, pages 2418–2421.

Frey, J. (2016). Comparison of an open-hardware electroencephalography amplifier with medical grade device in brain-computer interface applications. In *Int’l Conf. on Physiological Comp. Syst.*, pages 105–114.

Hamilton, P. (2002). Open source ECG analysis. *Computers in Cardiology*, pages 101–104.

Nogueira, P., Urbano, J., Reis, L., Cardoso, H., Silva, D., and Rocha, A. (2017). A review between consumer and medical-grade biofeedback devices for quality of life studies. In *Recent Adv. in Inf. Syst. and Tech.*, volume 570, pages 275–285. Springer.

Russomanno, C. (2014). Open BCI: Rise of the brain-computer interface. *Make:*.

Seidle, N. (2014). OpenLog: Open source data logging. *GitHub*.

Silva, H., Fred, A., and Martins, R. (2014). Biosignals for everyone. *IEEE Pervasive Computing*, 13(4):64–71.

[2]https://github.com/BITalinoWorld/firmware-loggerbit