BMET2922 Major Report |

The Heart Beat Measurinator 3000 – a novel photoplethysmography solution to wearable pulse monitoring.

Sithma Gunawardena | 510510006

TABLE OF CONTENTS

I INTRODUCTION & BACKGROUND

1.1 Photoplethysmography (PPG) is an optical diagnostic and monitoring technique that is simple, inexpensive, non-invasive, wireless, and measures continuously, giving a broad range of clinical applications. Through the utilization of light absorption and transmission, PPG monitors pulsatile vascular dynamics, providing health professionals with valuable information concerning cardiovascular parameters hence detecting blood volume at the peripheral vascular level. With various adaptations in hardware or in signal processing, it can measure blood oxygen saturation, blood pressure, hydration, breathing rate, and detect arrhythmias and other cardiac health issues. PPG holds a significant advantage in its ability to be unobtrusive to the user and as in the design proposed in this report, it can be incorporated into a wireless wearable device.

1.2 The clinical need for monitoring and accessible diagnostic technology in cardiovascular diseases is very high, with heart attacks and strokes remaining leading global causes of mortality [1]. The necessity of timely intervention makes early detection and continuous monitoring invaluable, and wearable, cheap, wireless solutions will make this more accessible to people. This report will discuss the significance of a pulse monitoring device, Heart Beat Measurinator 3000 (HBM 3000) and its impact on telemedicine, enabling healthcare professionals to monitor patients' cardiovascular well-being remotely and cheaply in the context of understaffed hospitals and GP offices, COVID-19, long waiting times, and higher cost-of-living.

1.3 PPG technology uses a photodetector and a light emitting diode (LED) to measure light absorption changes by returning different voltages depending on how much light is transmitted back to the detector, giving a graph as in Figure 1. Whilst PPG is advantageous in many scenarios, this is not to ignore its clear limitations compared to other technologies such as the ECG. In key, it is far more susceptible to sensor inaccuracies than its cousins, requiring the HBM 3000 software to alleviate the ‘noise’ the hardware collects by inputting and removing recognisable patterns. Such patterns or challenges that the HBM 3000 is guaranteed to struggle with is the varying device-to-skin pressure, light threshold differences from person to person and day to day, motion artefacts, and limitations in capturing deep tissue data [4].

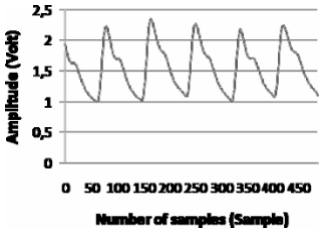

Figure 1 Waveform of the raw PPG signal [1]

1.4 The primary differentiation of PPG lies in its modality of measurement in comparison to other technologies such as the electrocardiogram (ECG). PPG employs an optical framework whereas the electrocardiogram derives its name from its utilization of electrical signals to ascertain hemodynamic parameters, a direct representation of a heartbeat, and hence an effective source of information about both heart rate and heart rate variability [5]. However, the added convenience of PPG being able to be detected by a wearable sensor, as well as the additional information beyond heart rate that PPG can give, make PPG more clinically valuable than ECG [5]. A downfall of PPG is that because of the signal being mapped peripherally and not at the heart (like ECG), individual physiological variation can affect the signals, with slight discrepancies in waveforms and intensity observed [7].

II SYSTEM DESCRIPTION

Figure 2 illustrates the operational framework of our system, delineated through a concise block diagram. The initial phase utilises data acquisition, facilitated by a wired connection between the

ESP32 Microcontroller and the host computer, operating on the Arduino platform. In this stage, three pivotal data streams are extracted from the microcontroller, demarcated by distinct blocks. I acquire the system status, the real-time pulse signal emanating from a sensor interfaced with the ESP32 microcontroller, and the instantaneous detection of the button presses. All pertinent information is then channelled through Arduino via a serial USB interface. The hardware is setup as in *Figure 3*.

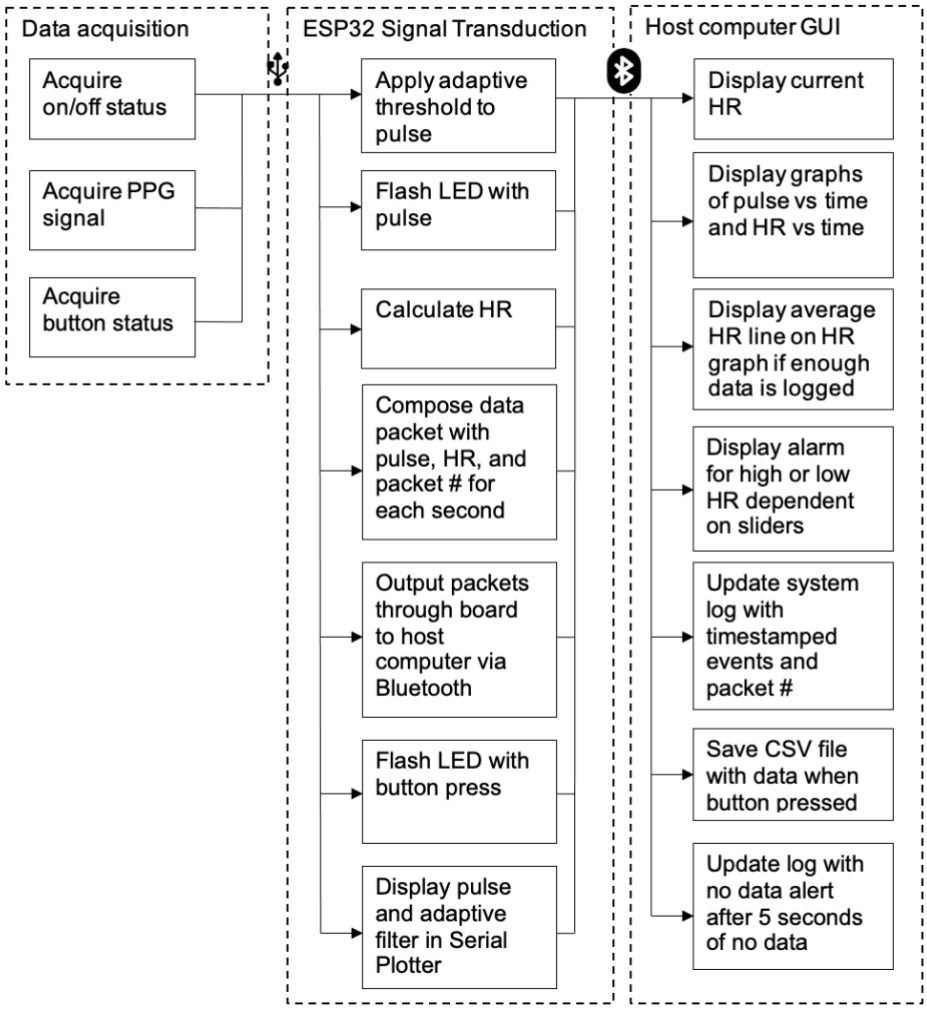

Figure 2: Block diagram outlining the basic function of the HBM 3000 system.

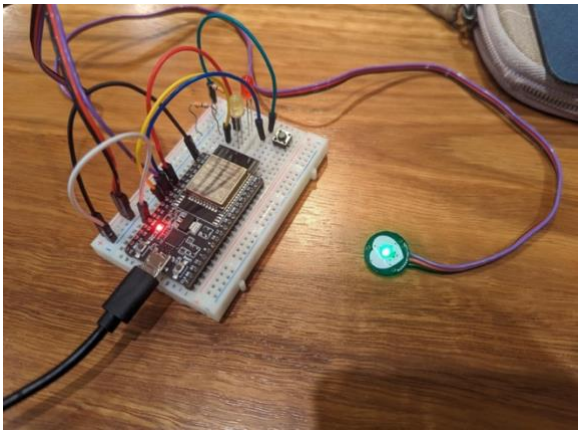

Figure 3: Hardware components.

The next component of the system is signal transduction, performed in Arduino, writing back to the board across the wired connection. The pulse signal is taken in and filtered (as described in *Chapter III Signal Processing*). The filtered pulse data is then used to calculate an average heart rate across each fifty filtered pulse datapoints. The actual pulse signal is graphed in the Arduino Serial Plotter, with a visual of the thresholds being used to filter the data. Data packets, composed of fifty filtered pulse values, an average heart rate (in bpm), a packet number and a button status binary variable, are compiled for every second (with pulse sampled every 20ms), and sent back to the board, to be output via Bluetooth. Finally, the actual hardware is controlled from Arduino, as one LED is flashed in rhythm with the pulse, and the other LED is flashed if a button press is registered from the board. All this data is handled and processed in Arduino and sent back to the ESP32 Microcontroller real-time.

The last component of the system is a graphic user interface (GUI) being generated using Python on the host computer. First, the program is run, creating a GUI with a blank system log, blank frame object for alarms and heart rate display and two blank graphs (one for heart rate, one for pulse readings). The data packets are read from the microcontroller using Bluetooth and unpacked for processing. From these data packets, the pulse readings, average heart rate, packet number and button status are all stored for every packet sent. Every second, the GUI is updated as follows: packet numbers are printed to the system log; the average heart rate for the most recent second of data is displayed prominently; and the graphs are updated with the most recent second of data. If more than 10 seconds of data are registered, a moving average line appears on the heart rate graph, showing the average heart rate across the most recent 10 seconds. If no data is registered for 5 consecutive seconds, a timestamped error message is printed in the log. There are two extra features encoded in Python in addition to these functions. The GUI contains two sliders entitled “Upper Threshold” and “Lower Threshold” to define the values above and below which the heart rate value is considered high or low respectively. When the upper threshold is exceeded, or lower threshold is fallen below, a corresponding alarm flashes prominently in the GUI. Furthermore, there is an inbuilt capability for storage of data, and when a button press is communicated (via the binary button status variable being written into data packets in “ESP32 Signal Transduction”), all of the pulse values and the average heart rates so far are written into and saved in an external Comma-Separated Values (CSV) file, separated by individual data packet. These files can also be generated by pressing a button in the GUI itself. Together, these features are constructed through Python and appear in the corresponding GUI created, for user interaction.

III SIGNAL PROCESSING

3.1 Adaptive Threshold

Our primary method of signal processing was implementing an adaptive threshold on the waveform calculation. The pulse calculation relies on determining each systolic peak and the period between successive peaks. A traditional fixed static threshold may correctly judge the cutoff value in some instances, but in others (where fingertip pressure changes, placement changes, a different patient uses the device, or blood vessel transmission changes due to exercise, stress, or breathing patterns) the static threshold would either overcount by including the lower diastolic peak, or miss the waveform entirely above it or below it.

A moving threshold, however, adapts in real-time to the patient's physiological dynamics which is essential in this context for collecting reliable results and data. It continuously monitors the level at which the light transmission oscillates and updates every three seconds to ensure it detects the heart rate correctly.

For instance, during physical activity or moments of heightened stress, the threshold may temporarily increase to accommodate expected fluctuations in heart rate. This ensures that the system remains highly sensitive to potential emergencies while avoiding unnecessary alarms in normal situations. As the patient's heart rate stabilizes, the moving threshold gradually adjusts downward, maintaining its responsiveness to the evolving physiological state.

3.2 Band Pass on BPM values

A secondary signal processing technique incorporated in our system involves the implementation of a bandpass filter, designed to effectively mitigate false pulse values. For instance, in scenarios where the waveform values all increase but the threshold hasn’t yet adapted, triggering pulse calculations at both the Systolic and Diastolic Peaks (as shown in Figure 4), the resultant BPM calculation could be erroneously inflated, potentially doubling the actual value. Similarly, instances where the sensor's light source is not stably positioned on the user's fingertip leads to waveform distortions primarily attributed to external noise and variations in light intensity rather than genuine blood vessel transmission, yielding substantially elevated, diminished, or inconsistent BPM readings.

Balancing the need to eliminate such erroneous pulse values while preserving accurate readings of extreme but genuine heart rates is a critical consideration. An overly restrictive bandpass, defined by narrow boundaries typically indicative of a healthy heart rate range (approximately 50 to 120 BPM), could inadvertently exclude critical data related to potentially life-threatening cardiac events. In our implementation, I widened the pass to encompass a range from 30 to 220 BPM. This adjustment accommodates a broader spectrum of heart rate variations, acknowledging that it may inadvertently include some spurious calculations. However, such outliers can be identified, scrutinized, and subsequently disregarded as hardware-related anomalies in the absence of corresponding cardiac events experienced by the user.

In accordance with established principles governing the design of medical diagnostic devices, it is worth emphasizing that in situations where diagnostic certainty is paramount, a false positive outcome is considered preferable to the potential consequences associated with a false negative result.

Figure 4: Normal function with the threshold adapting partway along (left), and a problematically low

threshold triggering at both the Systolic and Diastolic peaks, which would be corrected by the HR band pass (right)

IV DESIGN JUSTIFICATION

This system has multiple unique features, designed to maximise user-friendliness and optimise functionality. These features all have their own purposes and include an LED flashing in rhythm with the patient's pulse, sliders to enable customisation of high and low heart rate values, CSV data saving features and labelled Arduino Serial Plotter features.

The LED in our system has been set up to flash in rhythm with the pulse signal being received. This gives a very visual display of the patient's pulse with the peaks in signal corresponding to flashes of light. I decided to implement this feature both as a testing mechanism and as an extra feature to confirm to the user that a reading is being taken. It is useful for testing of the system, as the light flashing indicates when the sensor is outputting realistic pulse data. Given the highly volatile nature of the actual sensors themselves, this is particularly useful in debugging of code, as it can indicate when the problem with the program might be arising from a lack of stable data being registered by the sensor. This feature is also notable for potential patients using the system, as the flashing light representation of pulse can provide a simpler depiction of the signal than the graphs and can guide a user if they need to adjust the placement or pressure of their fingertip.

The sliders in our GUI (*Figure 5*) are also built to optimise the patient/client experience. Rather than fixed values for what is considered an unhealthily high or low heart rate, I decided to implement sliders to allow users to set their own thresholds. Heart rate norms tend to vary greatly based on existing health conditions, age, gender and race [8]. This means it would be presumptuous to set a single one-size-fits-all set of values to define what is unhealthy. By basing these values of sliders, I have allowed users to select the 'unhealthy values' themselves on recommendations from their practitioner. Based on the values set at any moment in time, our program will then output a prominent warning flag when the upper threshold is exceeded or lower threshold is fallen below, indicating that an unhealthy heart rate has been recorded. This will additionally be timestamped and indicated in the system log when saved.

A third unique feature of our system is the inbuilt CSV writing and saving capability. Given the established medical utility of PPG data and of monitoring heart rates (see *Chapter I Introduction and Background*), it is clearly of significant value to a patient's clinician to see a cohesive log of PPG and HR data across a relevant measuring period. Consequently, I decided that it would be useful to store the data being logged every second by our program in CSV format, separated by data packet for clarity. By storing in CSV format, I have ensured ease of future data processing and analysis. The stored data can be accessed during the running of the program by either pressing a "Save" button in the GUI (*Figure 5*), or pressing the button wired into the hardware. This action will save a new timestamped CSV file in the local computer for later analysis by the patient or clinician. The exact name of the file is printed in the system log when saved for ease of access. The added benefit of being able to keep a strict record of this data is extra utility for this pulse monitoring system for more complex experiments. Some medical studies tend to use heart rate as a proxy for sympathetic nervous arousal and hence monitor the effects of some external stimulus (e.g. a stressor [9]) on heart rate. For these sorts of studies, it would be key to store the heart rate data obtained to be able to properly analyse trends between the variables, further supporting our addition of this CSV feature into the system.

The final unique feature of our system is a secondary plot being output in Arduino using the Serial Plotter tool. The main reason for this feature being added is for testing and monitoring of the hardware. In the Serial Plotter, our Arduino program creates a continuous graph of the pulse signal, with superimposed lines representing the current values of the adaptive threshold. This allows a much clearer visual understanding of the data filtration ongoing in our system before the cleaned pulse signal reaches the GUI. This also can explain the data appearing in our GUI as when no data is detected in Python, I can map exactly what data is being received by Arduino, and how much of it is

being filtered out. This was very useful to us throughout the testing of our system, providing a sophisticated understanding of the exact data I were left with after cleaning and enabling us to more successfully put packets together and read and graph data in Python. This additionally could be helpful to a user, especially for a clinician, as it provides some background into the raw data behind the actual graphs being output in the GUI, helping provide some perspective into the error-prone nature of the hardware.

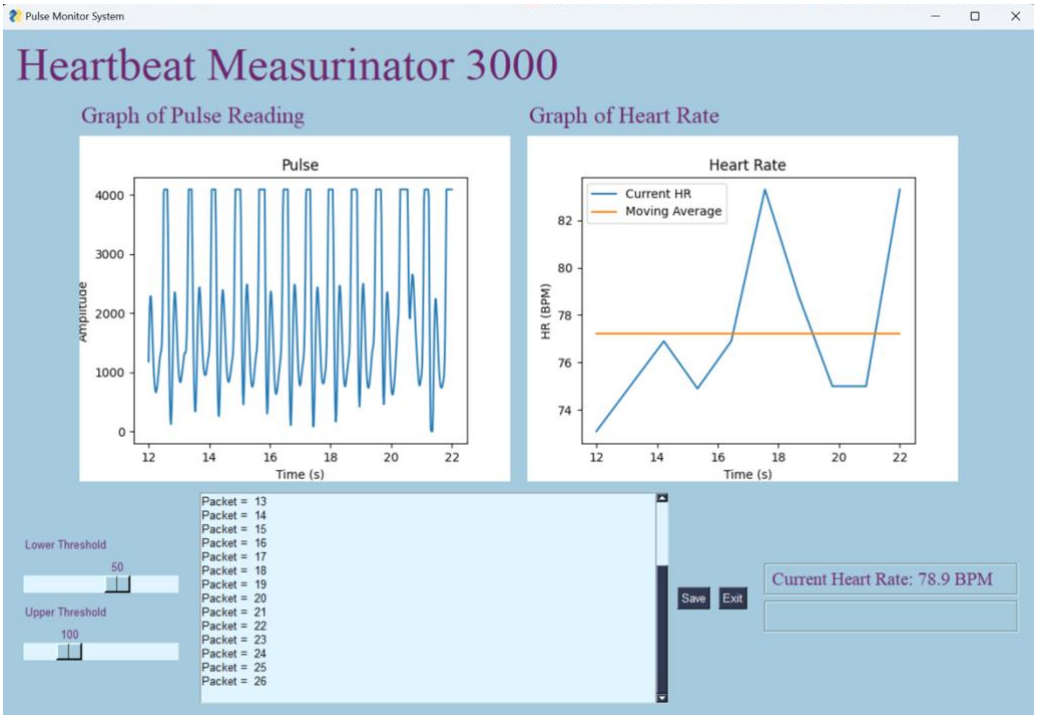

Figure 5: GUI for the HBM 3000.

V DEVELOPMENT PROCESS

I structured the project workflow by defining three distinct roles apparent in the design process of the pulse monitor system. Figure 6 illustrates the initial workflow plan that I developed, showing how tasks were separated into three roles, with certain components able to progress concurrently while others required prerequisite checkpoints to be completed.

For instance, I identified that data interpretation functions could only be validated once packet formatting and Bluetooth data transmission had been successfully implemented in Arduino. Similarly, graphing functions could only be meaningfully tested after the incoming data was being interpreted correctly. To mitigate these dependencies and prevent delays, I implemented sample input code that injected random data directly into the Python graphing functions. This allowed me to test and refine the graphing logic independently of the hardware and Bluetooth transmission, significantly reducing idle development time.

During development, I encountered challenges in constructing the graphical user interface (GUI) in Python, which resulted in a minor delay to the overall timeline. As a consequence, final system-level testing was deferred until Week 12, immediately prior to the laboratory demonstration.

I conducted system testing through a structured verification process to ensure that all functional requirements were met. Once Bluetooth connectivity was established, I executed the Python GUI program and performed iterative debugging to eliminate software errors. After achieving stable program operation, I proceeded with requirement-based testing of the system.

Heart rate calculations were validated by comparing the system output against manually measured pulse readings taken at the neck, as well as against reference values obtained during the BMET2901 cardiovascular physiology practical. I measured system response times using a stopwatch to confirm that Bluetooth connection was established within ten seconds of program execution and that graphical updates occurred at a minimum frequency of once per second. To assess robustness and fault detection, I removed a finger from the sensor and timed the delay before data loss was reflected in the graphs, as well as verifying that the no-data alarm was triggered after five seconds.

Finally, I implemented a print function within the Python data-reading routine to output the raw data packets in real time. This allowed me to directly verify that the data transmitted from Arduino matched the data received and interpreted in Python, confirming compliance with the specified packet format requirements.

VI CONCLUSION

The development of the Heart Beat Measurinator 3000 (HBM 3000) pulse monitoring system represents a significant advancement in the field of cardiovascular health monitoring. Leveraging the power of Photoplethysmography (PPG) technology, the HBM 3000 offers a simple, non-invasive, and cost-effective solution for continuous monitoring of vital cardiovascular parameters. Its wireless and wearable design further enhances its accessibility and usability for patients and healthcare professionals alike.

The system's adaptive thresholding and bandpass filtering techniques address critical challenges associated with PPG signal processing, ensuring accurate and reliable heart rate measurements. The incorporation of user-customizable high and low heart rate thresholds, LED pulse visualization, and CSV data saving features demonstrates a commitment to user-friendliness and versatility. As the HBM 3000 enters the next phase of development and implementation, several avenues for improvement and expansion present themselves. Further research and refinement of signal processing algorithms could enhance the system's accuracy in detecting abnormal heart rhythms and other cardiac anomalies. Additionally, exploring opportunities for real-time data analysis and decision support systems could provide valuable insights for both patients and healthcare providers.

Integration with telemedicine platforms and electronic health records (EHRs) could facilitate seamless remote monitoring and data sharing between patients and healthcare professionals. This could be particularly beneficial in the context of understaffed hospitals, the ongoing challenges posed by COVID-19, and the increasing demand for accessible and cost-effective healthcare solutions. Exploring additional sensor modalities, such as temperature and accelerometer data, could enable a more comprehensive assessment of cardiovascular health and overall well-being and research into the potential application of machine learning and artificial intelligence techniques could further enhance the system's capabilities in detecting and predicting cardiovascular events.

The Heart Beat Measurinator 3000 represents a significant step forward in the field of cardiovascular monitoring technology. Its user-friendly design, coupled with advanced signal processing techniques, holds great promise in revolutionizing how we monitor and manage cardiovascular health. With ongoing research and development, the HBM 3000 has the potential to make a substantial impact on healthcare accessibility and patient outcomes.

REFERENCES

[1] D. Castaneda, A. Esparza, M. Ghamari, C. Soltanpur, and H. Nazeran, "A review on wearable photoplethysmography sensors and their potential future applications in health care," (in eng), Int J Biosens Bioelectron, vol. 4, no. 4, pp. 195-202, 2018, doi: 10.15406/ijbsbe.2018.04.00125.

[2] Vulcan, R. S., André, S., & Bruyneel, M. (2021). Photoplethysmography in Normal and Pathological Sleep. Sensors (Basel, Switzerland), 21(9), 2928. https://doi.org/10.3390/s21092928

[3] Abay, T. Y., & Kyriacou, P. A. (2018). Photoplethysmography for blood volumes and oxygenation changes during intermittent vascular occlusions. Journal of clinical monitoring and computing, 32(3), 447–455. https://doi.org/10.1007/s10877-017-0030-2

[4] Zhang et al. A Noninvasive Blood Glucose Monitoring System Based on Smartphone PPG Signal Processing and Machine Learning. IEEE Transactions on Industrial Informatics 16(11), 7209-7218 (2020).

[5] G. Lu, F. Yang, J. A. Taylor, and J. F. Stein, "A comparison of photoplethysmography and ECG recording to analyse heart rate variability in healthy subjects," Journal of Medical Engineering & Technology, vol. 33, no. 8, pp. 634-641, 2009/11/01 2009, doi: 10.3109/03091900903150998.

[6] S. Blok, M. A. Piek, I. I. Tulevski, G. A. Somsen, and M. M. Winter, "The accuracy of heartbeat detection using photoplethysmography technology in cardiac patients," Journal of Electrocardiology, vol. 67, pp. 148-157, 2021/07/01/ 2021, doi: https://doi.org/10.1016/j.jelectrocard.2021.06.009.

[7] J. Fine et al., "Sources of Inaccuracy in Photoplethysmography for Continuous Cardiovascular Monitoring," (in eng), Biosensors (Basel), vol. 11, no. 4, Apr 16 2021, doi: 10.3390/bios11040126.

[8] R. Avram et al., "Real-world heart rate norms in the Health eHeart study," (in eng), NPJ Digit Med, vol. 2, p. 58, 2019, doi: 10.1038/s41746-019-0134-9.

[9] P. L. Dobkin and R. O. Pihl, "Measurement of psychological and heart rate reactivity to stress in the real world," (in eng), Psychother Psychosom, vol. 58, no. 3-4, pp. 208-14, 1992, doi: 10.1159/000288629.