Security of Pacemakers using Runtime Verification

Srinivas Pinisetty
School of Electrical Sciences
IIT Bhubaneswar
Bhubaneswar, India
Email: spinisetty@iitbbs.ac.in

Partha S Roop
Dept. of Electr. & Comput. Eng.
University of Auckland
Auckland, New Zealand
Email: p.roop@aucklanduni.ac.nz

Vidula Sawant
Dept. of Electr. & Comput. Eng.
University of Auckland
Auckland, New Zealand
Email: vsaw985@aucklanduni.ac.nz

Gerardo Schneider
Department of Comput. Sci. & Eng.
University of Gothenburg
Gothenburg, Sweden
gerardo@cse.gu.se

***Abstract*—The US Food and Drug Administration (FDA) recently recalled approximately 465,000 pacemakers that were vulnerable to hacking. It was reported that hackers could either pace the devices rapidly inducing arrhythmia or could drain the battery. Such actions would compromise the health and well being of the patient concerned. Considering this, techniques to ensure the security of implantable medical devices is an emerging area of research. To the best of our knowledge, existing techniques lack the formal rigour for ensuring the safety and security of such systems. While methods exist for formal verification of pacemaker software, these are not suitable to prevent security vulnerabilities. To this end we develop a run-time verification based approach. Our approach proposes a wearable device that non-invasively senses the familiar ECG signals in order to determine if a pacemaker has been compromised. We develop a set of timed policies to be monitored at run-time. We provide a methodology for the design of the wearable device and results demonstrate the technical feasibility of the developed concept.**

***Index Terms*—Monitoring, runtime verification, pacemaker security, timed automata**

I. Introduction

A *cardiac pacemaker* is an electrical device that manages irregular heart rhythms [1]. The pacemaker is implanted under the skin of the patient's chest, just under the collarbone, hooked up to the heart using a set of leads. Pacemakers are used to treat arrhythmia, which produces irregular heartbeat. During an arrhythmia, the heart can beat too fast, which is called *tachycardia*, or too slow, which is called *bradycardia*. The pacemaker senses intrinsic events (*atrial* and *ventricular events*) of the heart and gives electrical pacing pulses (either an atrial pulse or a ventricular pulse) whenever necessary. Figure 1 illustrates a dual chamber pacing system using a DDD-mode pacemaker. Here two leads are inserted in the right atrium and the right ventricle respectively. These leads act as both sensors and actuators, sensing the heart surface electrical activity as Electrogram (EGM) signals, and actuating by pacing the heart when the right signal is not detected at the right time.

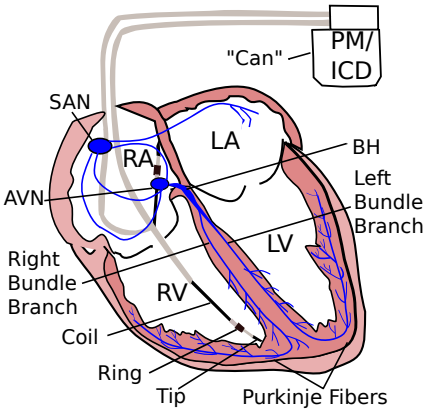


Fig. 1. The heart-pacemaker system showing the atrial and the ventricular chambers with pacing leads.

a) Motivation: Technical advances in biomedical engineering have resulted in a boom of wirelessly accessible pacemakers. Recent investigations on pacemakers revealed security vulnerabilities on existing pacemakers available commercially [2]. These artificial pacemakers are fitted with tiny radio components so they can be controlled and updated without having to cut them out and replace them each time. This means someone with the right technical knowledge and in the vicinity of the patient, could take control over the pacemaker, making it pace inappropriately. The hacker can make the pacemaker pace either too slow or too fast or may even drain the battery of the pacemaker. Many possible hacking avenues for pacemakers are reported in the detailed survey [2] and specific instances are reported in [3], [4].

b) Problem definition: A key problem related to pacemaker security is the trade-off of allowing emergency device access to health care professionals while ensuring that the device prevents all unauthorised access. Such trade-off for cyber physical systems (CPS), such as a pacemaker (a cyber com-

This research has been partially supported by the Swedish Research Council (*Vetenskapsrådet*) under grant Nr. 2015-04154 (*PolUser: Rich User-Controlled Privacy Policies*).

978-1-5386-6195-6/18/$31.00 ©2018 IEEE INDEXING & ABSTRACTING

ponent) controlling the rhythmic beating of the human heart (a physical system), is discussed in the detailed survey [5]. While cryptography is one of the best mechanisms for securing such CPS, a challenge is the problem of key distribution to legitimate parties, such as emergency health care workers. Considering this, alternative solutions based on devices that automatically detect and respond to anomalous behavior have been developed [6]. Here a medical security monitor, called MedMon has been developed. MedMon can detect and prevent anomalous events by snooping all wireless transactions. MedMon relies on a set of policies to be monitored to determine which events need to be jammed. The developed solutions has some limitations. First, the developed formulation requires the design of embedded devices mainly for snooping wireless channels. The efficacy of the device regarding cost, power consumption and certification needs to be studied. Importantly, MedMon's wireless snooping approach is as effective as the set of policies that are defined. As the policy framework is informal, there is no guarantee of soundness of the developed framework. Soundness guarantees are provided by techniques grounded in formal methods [7]. To the best of our knowledge, there is no formal solution for anomaly detection that can be implemented very efficiently and in a cost effective manner. To this end, we develop a solution using readily available Electrocardiogram (ECG) sensing technology combined with formal methods.

A. Typical ECG

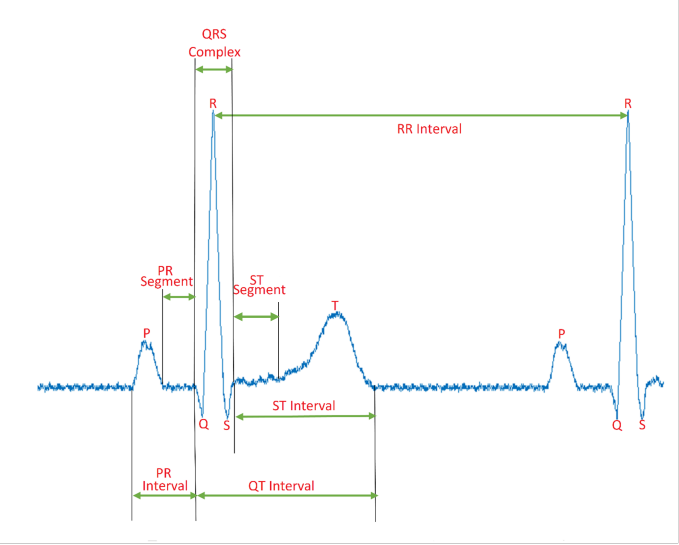


Fig. 2. Timing information in ECG signals (this is an adaptation from [8])

The Electrocardiogram (ECG) [9] [10] is the waveform produced by the heart showing the electrical activity of the heart over a period of time. Electrodes are attached to the skin on the chest, arms and legs which record the heart's electrical activity as waveforms from different angles. A typical ECG signal is illustrated in Figure 2. The *P-wave* symbolizes atrial depolarization (indicating an atrial event has occurred). P-waves precede QRS complex in sinus rhythm. The PR interval is the time interval between the beginning of the P-wave and the beginning of the *Q-wave*. It shows the time taken by the electrical impulses to travel between the atria and ventricles. The QRS complex denote the ventricular depolarisation (indicating ventricular event has occurred), atrial repolarization also happens during this interval. The ST-segment starts at the end of the S-wave and finishes at the start of the T-wave. It marks ventricular contraction i.e. time between depolarisation and repolarization of ventricles. The wave followed by QRS complex is the T-wave which represents ventricular depolarisation. The R-R interval indicates the time interval between two QRS complex. It begins at the peak of one R-wave and ends at the peak of the consecutive R-wave. The QT-interval is from the start of the QRS complex and finishes at the end of the T-wave. It represents the time taken for the ventricles to depolarise and then repolarise. Thus, a lot of information regarding the heart conditions can be extracted from the ECG signals of the patients, which is also valuable in the security context as highlighted by our methodology.

B. Overview of the proposed solution

Security risks in pacemakers are life threatening, which can make a life saving device a potential killer [3], [4]. Existing monitoring solutions for pacemakers require wireless communication with the pacemaker. This raises additional security challenges, especially when encryption and key distribution is a challenge. We propose an alternative monitoring mechanism that does not require any communication with the pacemaker or any external device. The monitor runs on an external wearable device, and uses person's ECG to identify events of interest. The device is programmed with critical pacemaker timing values by the cardiologist. We assume that this device is not connected to any other device, including the pacemaker, using any wireless protocol. Hence, we can make a reasonable assumption that our device is fairly robust and secure.

We adapt a runtime verification approach for timed automata [11] to create a monitor that identifies anomalous events at run-time. When any anomaly is detected, an alarm is sounded to alert the patient. *Runtime verification* (RV) [7], [12]–[17] approaches are concerned with monitoring and checking if a run of a system under inspection satisfies or violates a particular desired property (φ). RV is an ideal fit for pacemaker security due to the fact that it is only concerned with runs of the system, which is considered a black-box. This will thus require no modification to the existing pacemaker and hence will not require additional wireless protocols and associated key distribution. This will also not need any additional certification cost.

RV can be considered as a lightweight formally based verification approach, and one of the main emphasis of formally based RV approaches such as [13]–[15] is to generate RV monitor from a formal high-level specification of a set of properties. RV monitors do not modify the execution of the system. They are used to verify a stored execution of a

system (offline verification), or the current live execution of a system (online) with respect to a desired correctness property φ. The context of an RV monitor is illustrated in Figure 3, which takes a stream of events σ as input from the system being monitored, and emits a verdict that provides information whether σ violates or satisfies property φ. In this paper, we propose an externally wearable device which continuously monitors the ECG signals of the body for verifying critical safety properties defined for the heart-pacemaker activity using runtime verification techniques, giving additional layer of security as well as safety.

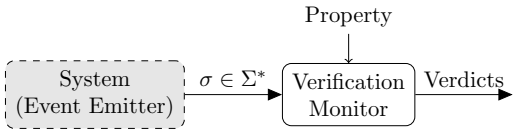


Fig. 3. Runtime Verification Monitor.

The key contributions of the paper are:

- We develop the first formal framework for tacking the emerging problem of pacemaker security using RV. We consider a set of safety properties and obtain RV monitors for them using techniques proposed in [13], [15],
- We develop the ECG sensing module for identification of key events and a run-time verification framework based on policies expressed as timed automata. These two modules operate online to verify these policies at run-time. Whenever a violation is detected, an alarm is triggered. Empirical results show that the overhead of such monitoring is minimal and hence such as wearable device can be implemented as an embedded system cost effectively.
- The developed solution requires no modifications to be made to pacemakers and their pacing logic. Also, there are no wireless communication modules, which minimizes the risk of attacks on the wearable device or the pacemaker using the developed formulation.

This paper is organised as follows: Section II discusses the key timers of a DDD mode pacemaker, their mapping to ECG intervals and finally the properties that should be monitored by the proposed wearable device. In Section III after presenting all the notations and preliminaries, formal definition of the verification monitor is given and its behavior is illustrated via an example. Overview of the proposed approach is given in Section IV, and later the two main modules of the system namely `ECG_Processing` and `RV_Monitor` are described in sections V, and VI respectively. Section VII discusses about implementation and results, and finally conclusions are drawn in Section VIII.

II. DDD mode Cardiac Pacemaker

In Section I, a DDD mode pacemaker has been briefly introduced via Figure 1. In this section we present various key timers for a DDD mode pacemaker, and discuss how these timers map to the ECG intervals discussed in Section I.

A timing diagram for a DDD mode pacemaker is shown in Figure 4. Unlike ECG, which are body surface signals, Electrograms (EGMs) record heart surface electrical activity based on the view provided by the leads (i.e. the pacemaker leads shown in Figure 1). At the top of the diagram EGMs for both an atrium and ventricle are shown. The status of various timers used for ensuring correct operation of the heart-pacemaker system are shown at the bottom. The traces at the top of the figure indicate three types of events: *natural events (Atrial Sense (AS) or Ventricular Sense (VS)), ignored natural event (Atrial Refractory Sense (AR) or Ventricular Refractory Sense (VR)), or artificial pacing from the pacemaker (Atrial Pace (AP) or Ventricular Pace (VP)).*

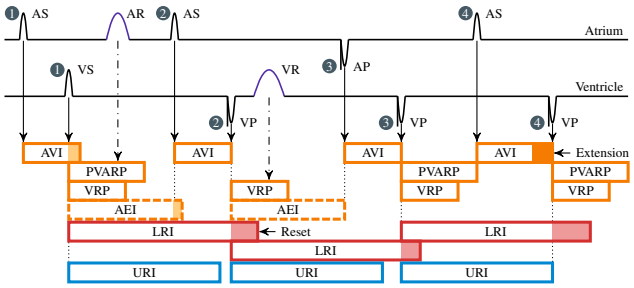


Fig. 4. Timing Diagram for a DDD mode pacemaker (adapted from [18]).

The Atrioventricular Interval (AVI) timer ensures the correct time delay between any atrial event and a subsequent ventricular event. When VS has not occurred before the AVI-interval the pacemaker delivers a paced variant i.e. VP. Examples of paced VP events are the beats 2 and 3. The Atrial Escape Interval (AEI) timer monitors the maximum time separation between any ventricular event and the corresponding atrial event. The pacing induced by the AEI-timer is illustrated by beat 3 in the diagram.

In addition, the Lower Rate Interval (LRI) and Upper Rate Interval (URI) timers maintain the overall heart rate within an upper and a lower bound. LRI ensures that the heart rate never falls below some minimum value. If this timer ever expires before another ventricular event has been detected, then the pacemaker must deliver a VP to force the heart rate within a normal range. URI performs the opposite task and ensures that the heart rate never goes *above* some maximum value. If this timer has not yet expired and the pacemaker attempts to deliver a VP signal, the signal will be delayed until the time that the URI timer has expired. This phenomenon can be seen in beat 4 where a VP from the AVI expiration has been delayed, and the timer extended, in order to not violate the URI. The Post-Ventricular Atrial Refractory Period ($PVARP$) timer acts as a blocking mechanism for spurious atrial events that occur shortly after ventricular events. If an atrial event occurs within this interval the event is marked as AR and is ignored, as shown between beats 1 and 2. Similarly, the Ventricular Refractory Period (VRP) timer blocks spurious ventricular events that also occur shortly after other ventricular

events. In this case, the events are marked as VR as shown between beats 2 and 3.

We consider the following set of critical safety properties from [19], where the timing intervals such as AVI, AEI and LRI are real values denoted as some constant C.

PM_1 AP and VP cannot happen simultaneously.

PM_2 VS or VP must be true within AVI after an atrial event AS or AP.

PM_3 AS or AP must be true within AEI after an ventricle event VS or VP.

PM_4 After a ventricle event, another ventricle event can happen only after URI.

PM_5 After a ventricle event, another ventricle event should happen within LRI.

a) From Pacemaker timers to ECG intervals: In the proposed monitoring approach, input events to be fed into the monitor that checks for the violation of properties of interest are extracted from the person's ECGs. Based on feature extraction of the ECGs, our monitoring device must detect whether properties $PM_1, \cdots, PM_5$ are violated or not. Since, the device has access only to surface ECGs, we need to map the Pacemaker timer values such as AVI and AEI to the ECG intervals described in Section I. The timers and their corresponding intervals are as follows.

- The AVI timer corresponds to the PR interval.
- The AEI timer corresponds to the time interval beginning from $R-wave$ till the subsequent $P-wave$.
- The LRI timer corresponds to the $R-R$ interval.
- The URI timer corresponds to the minimum time interval between two consecutive $R-waves$.

With the above mapping, properties $PM_1, \cdots, PM_5$ can be described in terms of ECG intervals as follows:

P_1 $P-wave$ and $R-wave$ cannot happen simultaneously.

P_2 $R-wave$ must arrive within PR interval after a $P-wave$.

P_3 $P-wave$ must be true within $R-P$ interval after an $R-wave$.

P_4 After an $R-wave$, another $R-wave$ can come only after $R-P$ interval.

P_5 After an $R-wave$, another $R-wave$ should come within $R-R$ interval.

In the remainder of this paper, $P-wave$, $Q-wave$ and $R-wave$ are denoted as P, Q and R respectively.

III. Preliminaries and Background

RV relies on monitoring of input streams σ and we start with a formalisation of such streams as words over an alphabet. Subsequently, we extend the setting to timed words. Given a finite *alphabet* Σ, a finite *word* over Σ is a finite sequence $\sigma = a_1 \cdot a_2 \cdots a_n$ of elements of Σ. The *length* of a finite word σ is denoted using $|\sigma|$, and ϵ denotes empty word over Σ. Σ^* denotes the set of all words over Σ, and any subset $\mathcal{L}$ of Σ^* is a *language* over alphabet Σ.

Given two words σ and σ', their *concatenation* is denoted as $\sigma \cdot \sigma'$. A word σ' is a prefix of a word σ, denoted as $\sigma' \preccurlyeq \sigma$, if $\exists$ a word σ'' s.t. $\sigma = \sigma' \cdot \sigma''$, and if additionally $\sigma' \neq \sigma$ then $\sigma' \prec \sigma$; conversely σ is called an *extension* of σ'.

The *set of prefixes* of σ is denoted as $\mathrm{pref}(\sigma)$ and subsequently, $\mathrm{pref}(\mathcal{L}) \stackrel{\text{def}}{=} \bigcup_{\sigma \in \mathcal{L}} \mathrm{pref}(\sigma)$ denotes the set of prefixes of words in $\mathcal{L}$. A language $\mathcal{L}$ is *extension-closed* if $\mathcal{L} \cdot A^* = \mathcal{L}$ and *prefix-closed* if $\mathrm{pref}(\mathcal{L}) = \mathcal{L}$.

A. Timed words and timed languages

When a timed framework is considered, the time instances of occurrence of actions are also important. Let the set of non-negative real numbers be denoted by $\mathbb{R}_{\geq 0}$, and let Σ be a finite set of actions. An event is a pair (t, a). The absolute time of the event is given by $\mathrm{date}((t,a)) \stackrel{\text{def}}{=} t \in \mathbb{R}_{\geq 0}$ and the action is given by $\mathrm{act}((t,a)) \stackrel{\text{def}}{=} a \in \Sigma$.

A timed word over the alphabet Σ is a finite sequence of events $\sigma = (t_1, a_1) \cdot (t_2, a_2) \cdots (t_n, a_n)$, with $(t_i)_{i \in [1,n]}$ being a non-decreasing sequence in $\mathbb{R}_{\geq 0}$. The starting date of σ is denote by $\mathrm{start}(\sigma) \stackrel{\text{def}}{=} t_1$ and $\mathrm{end}(\sigma) \stackrel{\text{def}}{=} t_n$ denotes its ending date. The starting and ending dates are null for ϵ (empty timed word).

Given alphabet Σ, $\mathrm{tw}(\Sigma)$ denotes the set of timed words over Σ, and any set $\mathcal{L} \subseteq \mathrm{tw}(\Sigma)$ is a *timed language*. Notations in the untimed setting related to length, prefix, etc extend to timed words, though the alphabet in the timed setting $(\mathbb{R}_{\geq 0} \times \Sigma)$ is infinite.

The *untimed projection* (i.e., ignore dates) of σ is $\Pi_\Sigma(\sigma) \stackrel{\text{def}}{=} a_1 \cdot a_2 \cdots a_n$ in Σ^*. When concatenating two timed words, the dates should be non-decreasing in the resulting timed word. This is ensured if the ending date of the first timed word is less than the starting date of the second timed word. Formally, consider $\sigma = (t_1, a_1) \cdots (t_n, a_n)$ and $\sigma' = (t'_1, a'_1) \cdots (t'_m, a'_m)$ be two timed words with $\mathrm{end}(\sigma) \leq \mathrm{start}(\sigma')$. Their concatenation is

$$\sigma \cdot \sigma' \stackrel{\text{def}}{=} (t_1, a_1) \cdots (t_n, a_n) \cdot (t'_1, a'_1) \cdots (t'_m, a'_m).$$

By convention $\sigma \cdot \epsilon \stackrel{\text{def}}{=} \epsilon \cdot \sigma \stackrel{\text{def}}{=} \sigma$. Concatenation is undefined otherwise.

B. Properties as Timed Automata (TA)

A *timed automaton* [11] is a finite automaton that is extended with a set of finite (real-valued) clocks $X = \{x_1, \ldots, x_k\}$. A *clock valuation* for X is a function from X to $\mathbb{R}_{\geq 0}$, which is an element of $\mathbb{R}^X_{\geq 0}$. Consider $\chi \in \mathbb{R}^X_{\geq 0}$ and $\delta \in \mathbb{R}_{\geq 0}$, $\chi + \delta$ denotes the valuation assigning $\chi(x) + \delta$ to each clock x of X. The clock valuation χ where all clocks in X' are assigned to 0 is denoted as $\chi[X' \leftarrow 0]$ for a given set of clocks $X' \subseteq X$. The set of *guards* denoted as $\mathcal{G}(X)$, clock constraints defined as conjunctions of constraints of the form $x \bowtie c$ with $x \in X$, $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. We denote $\chi \models g$ when g holds according to χ (for given $g \in \mathcal{G}(X)$ and $\chi \in \mathbb{R}^X_{\geq 0}$).

a) TA syntax and semantics: In this paper, properties to be verified are formalised as timed automata, from which RV monitors are synthesised.

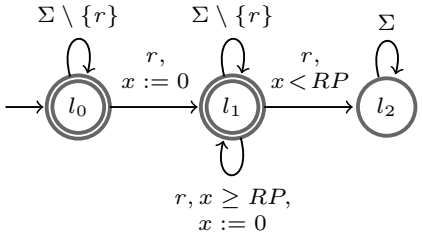


Fig. 5. Timed automaton defining property P_4 in Section II.

Definition 1 (Timed automata): A *timed automaton* $\mathcal{A} = (L, l_0, X, \Sigma, \Delta, F)$ is a tuple, s.t. L is a finite set of *locations* with the *initial location* $l_0 \in L$, a finite set of *clocks* X, Σ is a finite set of *actions*, $\Delta \subseteq L \times \mathcal{G}(X) \times \Sigma \times 2^X \times L$ is the *transition relation*. $F \subseteq L$ is a set of *accepting locations*.

Since the set of possible values for a clock is infinite, a TA has an infinite number of states. The semantics of a TA is defined as a transition system where each state is a tuple consisting of the current location and the current values of clocks. Its semantics is defined as follows.

Definition 2 (Semantics of TA): The *semantics* of a TA $[\![\mathcal{A}]\!] = (Q, q_0, \Gamma, \rightarrow, Q^F)$ is a *timed transition system* where $Q = L \times \mathbb{R}^X_{\geq 0}$ is the (infinite) set of *states*, $q_0 = (l_0, \chi_0)$ is the *initial state* where χ_0 is the valuation mapping every clock in X to 0, the set of *accepting states* is $Q^F = F \times \mathbb{R}^X_{\geq 0}$, and $\Gamma = \mathbb{R}_{\geq 0} \times \Sigma$ is the set of transition *labels*, that are pairs composed of a delay and an action. $\rightarrow \subseteq Q \times \Gamma \times Q$ is the *transition relation*, the set of transitions of the form $(l, \chi) \xrightarrow{(\delta, a)} (l', \chi')$ with $\chi' = (\chi + \delta)[Y \leftarrow 0]$ whenever $\exists (l, g, a, Y, l') \in \Delta$ s.t $\chi + \delta \models g$ for $\delta \in \mathbb{R}_{\geq 0}$.
A *run* of $\mathcal{A}$ from a state $q \in Q$ to a state $q' \in Q$ over a timed trace $\sigma = (t_1, a_1) \cdot (t_2, a_2) \cdots (t_n, a_n)$ is a sequence of transitions $q_0 \xrightarrow{(\delta_1, a_1)} q_1 \cdots q_{n-1} \xrightarrow{(\delta_n, a_n)} q_n$, where $q = q_0$, $q' = q_n$, $t_1 = \delta_1$, and $\forall i \in [2, n] : t_i = t_{i-1} + \delta_i$. If there exists a run from q to q' over σ, it is denoted by $q \xrightarrow{\sigma} q'$.

The *language of $\mathcal{A}$, starting in q and ending in K* (for $q \in Q$ and $K \subseteq L$) denoted as $\mathcal{L}(\mathcal{A}, q, K)$, defined by

$$\mathcal{L}(\mathcal{A}, q, K) = \{w \mid \exists q' \in K \times \mathbb{R}^X_{\geq 0} : q \xrightarrow{w} q'\}.$$

The language of $\mathcal{A}$, denoted as $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}, q_0, F)$ is the language of $\mathcal{A}$ starting from q_0 (initial state) and ending in a state in F.

Example 1 (Timed automaton): Let us now consider an example. The TA in Fig. 5 defines property P_4 presented in Section II i.e. *after an $R - wave$, another $R - wave$ can come only after $R - P$ interval*. The set of locations is $L = \{l_0, l_1, l_2\}$, and l_0 is the initial location. The set of actions is $\Sigma = \{p, q, r\}$. There are transitions between locations upon actions. The set $X = \{x\}$ is the set of real-valued clocks. On the transitions, there are guards with constraints on clock values such as $x < RP$ on the transition between l_1 and l_2 (where $RP \in \mathbb{N}$), and resets of clocks. Upon the first occurrence of action r, the automaton moves l_1 from l_0, and the clock x is reset to 0. If action r occurs in location l_1, and if $x \geq RP$, then the automaton remains in l_1, resetting the value of clock x to 0. It moves to location l_2 otherwise. For the property to be satisfied over runs, the state l_2 should never be reached i.e. it is a non-accepting location.

Remark 1 (Complete and Deterministic TA): In the sequel, a timed property is defined by a timed language $\varphi \subseteq \mathrm{tw}(\Sigma)$ that can be recognised by a deterministic and complete TA. A TA $\mathcal{A} = (L, l_0, X, \Sigma, \Delta, F)$ with its semantics $[\![\mathcal{A}]\!]$ is *complete* whenever $\forall l \in L, \forall a \in \Sigma : \bigvee_{(l,g,a,Y,l')} g = true$. That is, for any $l \in L$ and any $a \in \Sigma$, the disjunction of the guards of the transitions leaving l and labeled by a evaluates to *true* (i.e., it holds according to any valuation).

$\mathcal{A}$ is a *deterministic* TA if for any location l and any two distinct transitions $(l, g_1, a, Y_1, l'_1) \in \Delta$ and $(l, g_2, a, Y_2, l'_2) \in \Delta$ with same source l, the conjunction of guards $g_1 \wedge g_2$ is unsatisfiable.

C. Runtime Verification Monitor

In this section, for any given timed property φ, let us see the definition of the verification monitor.

Definition 3 (RV monitor): Consider a given property $\varphi \subseteq \mathrm{tw}(\Sigma)$ defining the property to monitor that is defined as TA $\mathcal{A}_\varphi$. Function $M_\varphi : \mathrm{tw}(\Sigma) \rightarrow \mathcal{D}$ is a verification monitor for φ, where $\mathcal{D} = \{\mathsf{true}, \mathsf{false}, \mathsf{c\_true}, \mathsf{c\_false}\}$ and is defined as follows, with $\sigma \in \mathrm{tw}(\Sigma)$ denoting the current observation (a finite timed word over the alphabet Σ):

$$M_\varphi(\sigma) \quad = \begin{cases} \mathsf{true} & \text{if } \forall \sigma' \in \mathrm{tw}(\Sigma) : \sigma \cdot \sigma' \in \varphi \\ \mathsf{false} & \text{if } \forall \sigma' \in \mathrm{tw}(\Sigma) : \sigma \cdot \sigma' \notin \varphi \\ \mathsf{c\_true} & \text{if } \ \sigma \in \varphi \wedge \exists \sigma' \in \mathrm{tw}(\Sigma) : \sigma \cdot \sigma' \notin \varphi \\ \mathsf{c\_false} & \text{if } \ \sigma \notin \varphi \wedge \exists \sigma' \in \mathrm{tw}(\Sigma) : \sigma \cdot \sigma' \in \varphi \end{cases}$$

In Definition 3, *true* (true) and *false* (false) are conclusive verdicts, and *currently true* (c_true), and *currently false* (c_false) are inconclusive verdicts, where an inconclusive verdict states an evaluation about the execution seen so far.

If for any continuation $\sigma' \in \mathrm{tw}(\Sigma)$, $\sigma \cdot \sigma'$ *satisfies* φ then $M_\varphi(\sigma)$ returns true. $M_\varphi(\sigma)$ returns false if for any continuation $\sigma' \in \mathrm{tw}(\Sigma)$, $\sigma \cdot \sigma'$ *falsifies* φ.

Monitor $M_\varphi(\sigma)$ returns inconclusive verdict c_true if σ satisfies φ, and if there is a continuation $\sigma' \in \mathrm{tw}(\Sigma)$ such that $\sigma \cdot \sigma'$ does not satisfy φ (i.e., not all continuations of σ satisfy φ). Inconclusive verdict c_false is returned if σ falsifies φ, and there is a continuation $\sigma' \in \mathrm{tw}(\Sigma)$ such that $\sigma \cdot \sigma'$ satisfies φ.

Remark 2 (Monitorability): A property $\varphi \subseteq \mathrm{tw}(\Sigma)$ expressed as a TA $\mathcal{A}_\varphi$ is *monitorable* [13], [14] if for any current observation $\sigma \in \mathrm{tw}(\Sigma)$, there exists a finite word $\sigma' \in \mathrm{tw}(\Sigma)$ such that the property φ can be evaluated to true or false for $\sigma \cdot \sigma'$. That is,

$$\forall \sigma \in \mathrm{tw}(\Sigma), \exists \sigma' \in \mathrm{tw}(\Sigma) : M_\varphi(\sigma \cdot \sigma') \in \{\mathsf{true}, \mathsf{false}\}.$$

Remark 3 (Impartiality and anticipation): For any given property $\varphi \subseteq \mathrm{tw}(\Sigma)$ (expressed as a TA $\mathcal{A}_\varphi$), monitor M_φ as per Definition 3 satisfies impartiality and anticipation constraints [13].

Impartiality means that for a finite trace $\sigma \in \mathrm{tw}(\Sigma)$, M_φ provides an inconclusive verdict (c_true or c_false) if and only

if there exists a continuation of σ leading to another verdict. That is, if σ itself satisfies φ, but there is some extension of σ which does not, or conversely, if σ does not satisfy φ but some extension it does satisfy, then the monitor must give an inconclusive verdict on σ.

Anticipation states that for a finite trace $\sigma \in \mathrm{tw}(\Sigma)$, the monitor $M_\varphi(\sigma)$ should provide a conclusive verdict true (resp. false) iff every continuation of σ satisfies (resp. violates) φ. Thus, if $M_\varphi(\sigma)$ is true (resp. false), then every continuation of σ also evaluates to true (resp. false).

TABLE I
EXAMPLE ILLUSTRATING BEHAVIOR OF RV MONITOR FOR PROPERTY P_4

σ	$M_\varphi(\sigma)$
$(50, p)$	c_true
$(50, p) \cdot (208, r)$	c_true
$(50, p) \cdot (208, r) \cdot (300, p)$	c_true
$(50, p) \cdot (208, r) \cdot (300, p) \cdot (451, r)$	false

Example 2 (Example illustrating behaviour of an RV monitor): Consider monitoring property P_4 from Section II, defined formally by the TA in Figure 5. Let RP be 900 time units. Table I illustrates how the monitor for P_4 behave when the input timed word $\sigma = (50, p) \cdot (208, r) \cdot (300, p) \cdot (451, r)$ is processed incrementally.

At t = 50, when the current observed input is $\sigma = (50, p)$, σ satisfies the property P_4 but there are some extensions $\sigma' \in \mathrm{tw}(\Sigma)$ such that $\sigma \cdot \sigma'$ falsify the property P_4. So the verdict provided by the monitor is c_true in the first step. Similarly, the verdict provided by the monitor is c_true in the next two steps at t = 208 and at t = 300. At t = 451, after observing the event $(451, r)$ (i.e., when the current observed input is $\sigma = (50, p) \cdot (208, r) \cdot (300, p) \cdot (451, r)$), the property φ is falsified by σ and for any extension $\sigma' \in \mathrm{tw}(\Sigma)$, $\sigma \cdot \sigma'$ falsifies the property φ. Thus, the monitor provides a conclusive verdict (false) immediately after observing $(451, r)$.

IV. OVERVIEW OF THE PROPOSED APPROACH

In Section II, we presented some example of safety properties that we consider for verifying via runtime monitoring. In this section, we provide a brief overview of the proposed approach, and outline the architecture of the heart-pacemaker-safety monitoring device. Let us recall that, in this work, we consider the DDD mode in which both atria and ventricles are sensed as well as paced (see Figure 1).

A. Overview of the approach

The externally wearable device (RV monitor) is assumed to have more power and computational resources than the pacemaker and is capable of measuring ECG signals. The pacemaker, once implanted, is expected to remain inside the body for an extended period of time. Before implanting it is programmed by the doctor with the help of a programming unit (outside controller) with direct connection. After implantation, if the pacemaker has to be reprogrammed, that should be done wirelessly. The programming unit provides doctors an interface to interact with the pacemaker through radio frequency transmission for adjusting running parameters (timers), changing operation modes, or retrieving stored data [20].

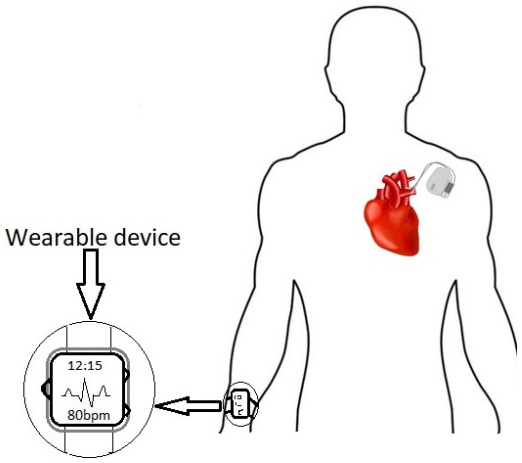


Fig. 6. Overview.

Whenever the pacemaker is implanted, the doctor programs the pacemaker's computer with an external programming unit. The doctor doesn't have to use needles or have direct contact with the pacemaker. The two main types of programming for pacemakers are demand pacing and rate-responsive pacing. The doctor will work with the patient to decide which type of pacemaker is best for the patient. While programming the doctor essentially sets the pacing mode (eg. DDD), sets the threshold voltage value of the pacing pulse, sensitivity of the pacemaker and most importantly the timers such as AVI, and AEI. If an attacker hacks the pacemaker he/she may try to increase or decrease any of these timers. Hence, the properties in Section II are formulated keeping in mind this security vulnerability.

a) External wearable device: In the approach that we propose in this work, the patient is given the wearable device once the pacemaker is implanted. This external wearable device could be any computing device with an ECG sensor and an accelerometer such as a smart watch as illustrated in Figure 6. The doctor also configures the external wearable device with timing values. Essentially, all the values of the set timers are stored inside the wearable device's memory. It also knows the normal heart rate at which the pacemaker is set to pace (eg. 60-120 BPM), and has information of the attributes of the pacing pulses like voltage, current and impedance. The device has a built-in accelerometer which monitors the activity of the body.

Via the ECG sensor, the device has access to the surface ECG signals, and it continuously monitors these signals. After filtering and processing the ECG data, it extracts all relevant actions of interest (the peaks of P, Q, R, S, T waves and the pacing pulses). Depending on the time instances at which the peaks (actions) are occurring, it checks for any violation of any of the safety properties using RV monitors. If any of the desired properties are violated, it generates an alarm for the

user. As illustrated in Figure 6, the device does not have any direct communication with the Pacemaker.

R waves detection helps in calculating the heartbeat every minute. Hence, the RV monitoring device can constantly check for the normal heart rate (beats per minute) too. Also the detection of pacing pulses, which have different morphology, provide crucial information about the amplitude of the pulses. Based on this information the condition of the leads can be found out, which will help in examining any malfunctions in the leads.

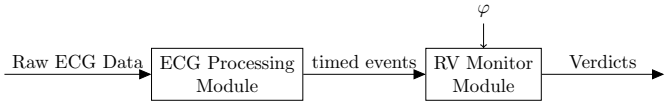


Fig. 7. Architecture of the monitoring system.

B. Architecture of the monitoring system

In this section, we present the architecture of the proposed monitoring system which is hosted on the wearable device. As illustrated in Figure 7, the monitoring system consists of two modules namely i) `ECG_Processing` module, and ii) `RV_Monitor` module. Both these modules work concurrently in an *on-line* manner.

The `ECG_Processing` module is responsible for filtering and processing the ECG data. For monitoring the considered set of properties ($P_1, \cdots, P_5$ introduced in Section II), we need to extract all the relevant actions of interest such as P, Q and R peaks from raw ECG data. The `ECG_Processing` is responsible for detecting these actions and feeding them as input with relevant timing information to the `RV_Monitor` module.

The `RV_Monitor` module takes the property to be verified (φ) and a stream of timed events (fed as input by the ECG processing module in an online manner) as input, and emits verdicts providing information whether the input event stream satisfies (resp. violates) property φ.

The `ECG_Processing` module performs real-time signal processing of human ECGs to detect the events of interest. These events are passed in the appropriate format (as timed events) to the online `RV_Monitor` module. We elaborate the `ECG_Processing` module in Section V, and the online `RV_Monitor` module in Section VI.

Remark 4: Note that the properties that we consider to monitor ($P_1, \cdots, P_5$) are timed safety properties, formally expressible as timed automata [11], [15], for which verification monitors can be synthesised using approaches such as [15]. However, monitoring of these properties, implicitly prevent attacks as follows. First, as the pacemaker has wireless programmability, an attacker may gain access to the device and maliciously change the programmed timing values, which may cause serious harm. However, the attacker is unable to access the wearable device as this is secure (through secure authentication mechanisms such as say the use of human biometric) and is not programmable wirelessly. In this event the attack will be detected within a short time due to the mismatch in timing values leading to violation of the properties.

V. ECG Signal Processing Module

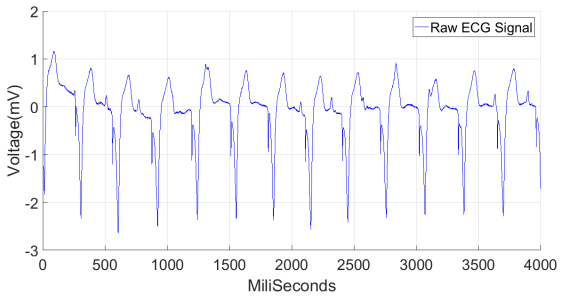
In this section, we will briefly discuss about the functioning of the `ECG_Processing` module. That is, we discuss how ECG signals are processed by the wearable device to extract all the relevant actions of interest for monitoring the considered set of properties. The `ECG_Processing` module is implemented in MATLAB. We have proposed the concept of the wearable device and have not actually designed it in this paper. Thus, this paper demonstrates the technical feasibility of the idea. Hence, we use prerecorded ECG data that have pacing artifacts and do not actually generate ECG signals. We then use them as if they are being generated in real time by feeding one ECG cycle at a time for processing. Thus, the `ECG_Processing` module runs in a loop, processing all ECG cycles in the recording one by one.

A. ECG Database

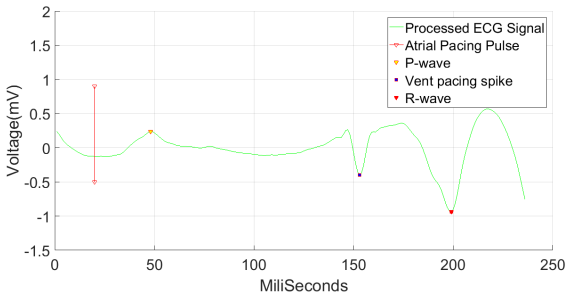
A huge collections of recorded physiologic signals are available from [21]. Consequently, the database of ECG signals for our experiments is also obtained from [21]. The ECG signal in use is from a 63 year old male patient. The patient is living with a pacemaker installed in his body. Hence, the ECG signals from this patient include pacing artifacts. He suffers from a complete heart block and the PVCs (Premature ventricular contractions) are multiform. Originally the recorded signal contains 30,000 samples and is in the form of a `.dat` file which is first read in MATLAB. We then break the recording in 6 segments (5000 samples each) for ease of performing experiments and convert in to a `.mat` file to be supplied to the code for further execution. Pacers distort the typical QRS complex because the depolarisation wave is not propagated normally in a paced heart, so the Q-wave and S-wave are not evidently present to be detected by the code in this recording. Hence, we only detect the P-wave, R-wave and the pacing pulses.

B. Pre-processing of ECG signals

Generally, a raw ECG signal (Figure 8(a)) is corrupted by Baseline wander, Power line interference (50 Hz or 60), Electromyographic (EMG) or muscle noise, and artifacts due to electrode motion and Electrode Contact Noise [22]. Hence, pre-processing of the signal is important to remove unwanted data and to be able to extract vital information from the signals. Because of the baseline shift, the signal is not present in it's true amplitude. Hence, appropriate detrending is required for correct detection of events. A lower order polynomial can be fitted to the signal and then used to detrend it, so that the signal with it's actual amplitude can be obtained. Since, the signal is corrupted with high frequency noise, Savitzky-Golay filtering is used to remove noise from the signal. MATLAB "filter design and analysis" application can also be used for filtering of the ECG signal if the sampling frequency of the signal is known. A comparison of the raw ECG signal and pre-processed signal is shown in Figure 9.



(a) Raw ECG signal.



(b) Processed ECG signal.

Fig. 8. Raw and processed ECG signals.

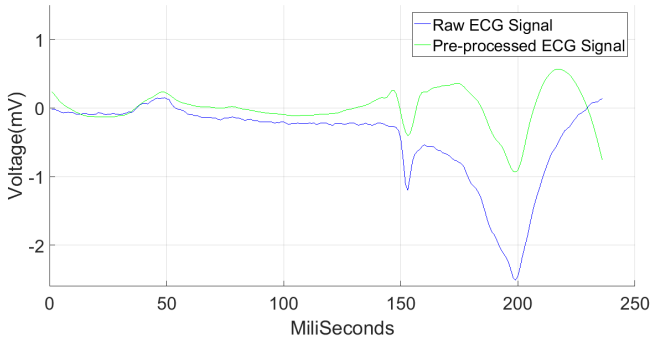


Fig. 9. Comparison between raw and pre-processed ECG.

C. Processing of ECG signal

Since we focus on online monitoring, we process one cycle of ECG signal from the entire recorded signal at a time. We divide the entire recording into ECG cycles and then apply pre-processing to remove unwanted data. After the filtering of the signal, the waves of our interest are detected using peak analysis.

The peaks of P, R waves and the pacing pulses are extracted using the function "findpeaks" in MATLAB (in-built MATLAB function). Originally, the ECG signal had fixed rate ventricular pacing only. We add synthetic atrial pacing pulse to the signal, to make it resemble to a DDD mode signal. The signal begins with a pacing pulse (atrial pacing pulse), followed by P-wave, then another pacing pulse (ventricular pacing pulse), followed by R-wave as shown in Figure 8(b). In Figure 8(b), X-axis denotes time in milliseconds and Y-axis denote amplitude in millivolts.

After processing the ECG data, the `ECG_Processing` module returns arrays of P, R waves and the pacing pulses (the time in milliseconds at which the peaks are occurring). Thus, a stream of timed events from the ECG data is obtained. The ECG signal was recorded when the patient was in resting position. The `ECG_Processing` module successfully detects all the relevant events for this recording. The module would not be able to detect events correctly when the patient is exercising or is under heavy physical activity. Since the module also detects pacing pulses, it can differentiate whether the ECG cycle under observation was paced or intrinsic. If a pacing spike is detected, the ECG cycle is paced otherwise it is intrinsic.

VI. Online Monitoring Algorithm

In this section, we will focus on the `RV_Monitor` module, which is implemented in Python. In Section III-C, we saw a functional description of a RV monitor for any given timed property φ. Let us now briefly see an *online* RV monitoring algorithm, which is similar to the existing RV monitoring algorithms such as [13], [15].

Consider $\mathcal{A}_\varphi = (L_\varphi, l^0_\varphi, X_\varphi, \Sigma, \Delta_\varphi, F_\varphi)$, the TA defining property φ to be verified. Let q^0_φ be the initial state of $[\![\mathcal{A}_\varphi]\!]$, and let and Q^F_φ be the sets of final states of $[\![\mathcal{A}_\varphi]\!]$. Each input event consists of the absolute time t and the action $a \in \Sigma$.

Algorithm RV Monitor (see Algorithm 1) is a *repeat until* loop that starts from the initial state of the automaton $\mathcal{A}_\varphi$. In the iteration, the algorithm checks the verdict for the current state. If the verdict is either true or false (i.e., if the verdict is conclusive), then the loop terminates. Otherwise, upon receiving an input event (t, a), it updates the current state (in accordance with the received event), and it does a new iteration.

In Algorithm 1, $\mathcal{A}_\varphi$ (the TA defining the property to monitor) is provided as input parameter. Variable q denotes state in the semantics of $\mathcal{A}_\varphi$, which is initialised with the

Algorithm 1 RV Monitor

1: $t_0 \leftarrow 0$
2: $F, F' \leftarrow F_\varphi, L_\varphi \setminus F_\varphi$
3: $q \leftarrow q^0_\varphi$
4: **repeat**
5: **if** $\mathcal{L}(\mathcal{A}, q, F') = \emptyset$ **then**
6: notify(true)
7: exit
8: **else if** $\mathcal{L}(\mathcal{A}, q, F) = \emptyset$ **then**
9: notify(false)
10: exit
11: **else if** $q \in Q^F_\varphi$ **then**
12: notify(c_true)
13: **else**
14: notify(c_false)
15: **end if**
16: $t, a \leftarrow$ await_event()
17: $t_0, \delta \leftarrow t, t - t_0$
18: $q \leftarrow \mathsf{move}(\mathcal{A}_\varphi, q, (\delta, a))$
19: **until** false

initial states of the automaton. Variable t_0 keeps track of the date of the last received event which is initialised with 0.

Primitive await_event is used to wait for a new input event, and primitive notify notifies the verdict (the result of the function M_φ) at every step. Primitive $\mathsf{move}(\mathcal{A}, r, (\delta, a))$ is a function that returns a new state r' reached in $[\![\mathcal{A}_\varphi]\!]$ from r with (δ, a).

Remark 5 (Complexity of Algorithm 1): For a TA $\mathcal{A} = (L, \ell^0, X, \Sigma, \Delta, F)$ with semantics $[\![\mathcal{A}]\!] = (Q, q_0, \Gamma, \rightarrow, Q^F)$, consider a state $q \in Q$ and $K \subseteq L$. The language $\mathcal{L}(\mathcal{A}, q, K)$ starting from state q and ending in K is empty if K is not reachable from q. From Algorithm 1, we can notice that the only computationally expensive steps are the emptiness tests in line 5 and 8. Since reachability problem is PSPACE-complete [11], checking emptiness is also PSPACE-complete.

Remark 6 (Computing symbolic states and verdicts off-line): As discussed in earlier works on monitoring for timed properties such as [15] based on timed automata, to improve the real-time performance of the monitor, all the computationally expensive steps can be pre-computed. In $[\![\mathcal{A}]\!]$, all reachable symbolic states in where a symbolic state is a pair consisting of a location and a zone can be computed off-line. Moreover, checking if the set of locations F (resp. F') is reachable from a given symbolic-state can be also computed off-line. Thus, the verdicts corresponding to all reachable symbolic states can be computed off-line.

Remark 7: Verifying multiple properties. When a set of properties $\varphi_1, \cdots \varphi_n$ are considered, where φ_i is defined as TA $\mathcal{A}_{\varphi_i}$, we first compute the product of all the TA's

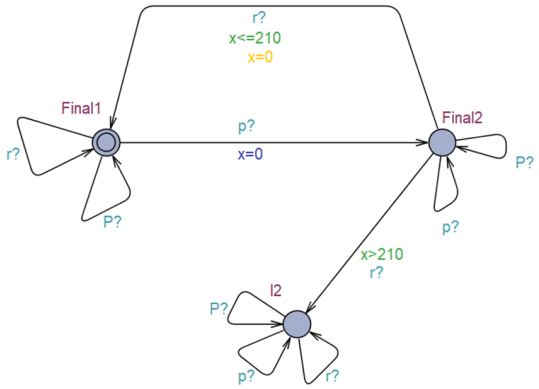


Fig. 10. Property P_2 in UPPAAL format.

$\mathcal{A}_\varphi = \mathcal{A}_{\varphi_1} \times \cdots \mathcal{A}_{\varphi_n}$, where $\mathcal{A}_\varphi$ recognises the language $\mathcal{A}_{\varphi_1} \cap \cdots \cap \mathcal{A}_{\varphi_n}$. The combined property $\mathcal{A}_\varphi$ is monitored. See for instance [11] for formal definition and details related to computing product of TA's.

VII. Implementation and Results

In order to demonstrate the practicality or the proposed approach, we have developed a prototype that shows how the monitor reacts when it receives a new event, and identifies violation(s) of the properties of interest. The prototype consists of different modules such as the `ECG_Processing` module (module to process ECG data and identify events of interest), and the `RV_Monitor` module (module that provides a verification monitor for a property given as input).

The `ECG_Processing` module has been implemented in MATLAB. The MATLAB version used for execution was R2017a. The input to this module is an real ECG signal recording taken from [21] which is about half an hour long. After the appropriate processing of the ECG signal, the module provides arrays of time instances at which the peaks (P, R) occur. P and R arrays are passed as arguments to the `RV_Monitor` module. The monitor is invoked every time P and R waves are detected. The monitor performs check for all the properties together.

The `RV_Monitor` module (implementation of Algorithm 1), is based on the implementation provided with [15], developed in Python 2.7 using UPPAAL DBM libraries [23]. The RV monitor module requires as input the set of all the events, the property to be verified and a sequence of input events (input event stream to be verified).

a) Input-output behavior of the monitor: Let us consider property P_2 in Section II as the property to be verified (φ). The TA defining Property $P2$ (in UPPAAL format) is illustrated in Figure 10. PR interval to be checked is set to 210 milliseconds for demonstration, and x is the clock. The time intervals differ from patient to patient. Here, we have taken the average between the PR intervals for all the ECG cycles and calculated it to be 210 milliseconds for the verification purpose. The RV Monitor is invoked passing the property to verify (Figure 10)

TABLE II
EVALUATION RESULTS.

Property	No. of ECG cycles	Time (ms) (ECG processing)	Time (ms) (RV monitor)	Total Time (ms)
P_1	2	207.80	3.460	211.26
P_2	2	207.80	3.507	211.307
P_3	2	207.80	3.238	211.038
P_4	2	207.80	4.030	211.83
P_5	2	207.80	4.022	211.82

and an input event sequence (input sequence to be checked against the property).

An example of an input event sequence could be $[(`p', 50), (`r', 250), (`p', 86), (`r', 300)]$ (where each event is associated with a delay, indicating the time elapsed after the previous event or the system initialization for the first event). p stands for P-wave, r stands for R-wave and p stands for paced pulse. Here, the monitor receives the first action p at t=50. Since, the property is not violated the monitor will output verdict c_true. The second action r comes at t=300, the monitor will again emit verdict c_true since r event has come before PR interval and the property is not violated. The third action p comes at t= 386, again the monitor will emit c_true. The fourth action r comes at t=686, here the property is violated because r event should have come before PR interval i.e. 210 but r event occurred after PR interval, the property is violated, therefore the monitor will output conclusive verdict false, indicating property is violated and the wearable monitoring device will display an alert message.

b) Performance analysis: We conducted the experiments on an Intel Core i5-7200 at 2.50GHz CPU, with installed RAM of 4 GB, running on Ubuntu 16.04, 64-bit operating system.

In order to evaluate the effect of runtime verification algorithm on the execution time, the runtime verification monitor was tested for all properties. Table II presents the results of the execution.

For each property, we consider events extracted from two consecutive ECG cycles, since we need maximum 2 ECG cycles to verify the properties. Likewise, the experiments are conducted on the entire recording, performed in a loop. For the considered ECG data, from table II we can observe that the total online time (sum of the online time taken by the `ECG_Processing` and the `RV_Monitor` modules) is about 212 ms. The `ECG_Processing` module takes approximately 130 ms to process one ECG cycle. The average resting heart rate for adults is between 60-100 bpms, for athletes it can go as low as 40 bpm [24]. Hence, one ECG cycle is approximately of 1000 milli-seconds. Thus, the wearable monitoring device is considerably fast to check for any anomalies in the heart activity.

VIII. CONCLUSION AND FUTURE WORK

Security vulnerabilities in implantable devices, such as pacemakers, is of increasing concern [2]. This paper develops a proposal to tackle this formally by combining run-time verification (RV) based monitoring of timed properties with a ECG processing module that identifies and forwards the timing events of interest to the monitor. A set of safety properties are used to continuously monitor a set of pre-programmed timing values into a secure wearable device. We assume that our monitoring method along with the ECG processing module is integrated into such a device, which is secured using human biometrics. Hence, the wearable device is considered tamper proof. In contrast, the pacemaker is considered insecure, due to the presence of wireless channels, which are used for programming the timing parameters of the pacemakers. This is a realistic assumption considering that attacks on pacemakers, where a malicious attacker can alter the timing parameters to induce dangerous situations has been already studied [3], [4]. Since the RV monitors timing values are tamper proof, any attack that modifies the timing parameters will be detected by the monitor and an alarm will be sounded. In order to demonstrate the efficacy of the proposed approach, we have demonstrated that the overhead of the RV monitor is minimal and an embedded device that implements such as system is feasible.

While this works paves the way for formal methods guided methods for pacemaker security, it is not devoid of limitations. First, this paper only develops the concept and its technical feasibility but this yet to be implemented on a real device. Second, the signal processing part is implemented currently in Matlab and its efficiency is yet to be carefully studied. Finally, we have only tested the system using ECG of a single patient. This may be extended to a larger sample size in the near future.

REFERENCES

[1] S. S. Barold, R. X. Stroobandt, and A. F. Sinnaeve, *Cardiac Pacemakers Step-by-Step: An Illustrated Guide*. John Wiley & Sons, 2008.

[2] J. Sametinger, J. Rozenblit, R. Lysecky, and P. Ott, "Security challenges for medical devices," *Communications of the ACM*, vol. 58, no. 4, pp. 74–82, 2015.

[3] J. Kirk, "Pacemaker hack can deliver deadly 830-volt jolt," *Computerworld*, vol. 17, 2012.

[4] D. Clery, "Could your pacemaker be hackable?" *Science*, vol. 347, no. 6221, pp. 499–499, 2015. [Online]. Available: http://science.sciencemag.org/content/347/6221/499

[5] R. AlTawy and A. M. Youssef, "Security tradeoffs in cyber physical systems: A case study survey on implantable medical devices," *IEEE Access*, vol. 4, pp. 959–979, 2016.

[6] M. Zhang, A. Raghunathan, and N. K. Jha, "Medmon: Securing medical devices through wireless monitoring and anomaly detection," *IEEE Transactions on Biomedical circuits and Systems*, vol. 7, no. 6, pp. 871–881, 2013.

[7] J. Olaf Blech, Y. Falcone, and K. Becker, "Towards Certified Runtime Verification," in *ICFEM 2012*, ser. LNCS, vol. 7635. Springer, 2012, pp. 494–509.

[8] I. Jekova, V. Tsibulko, and I. Iliev, "Ecg database applicable for development and testing of pace detection algorithms," *International journal bioautomation*, vol. 18, pp. 377–388, 12 2014.

[9] "ecgSignal," https://geekymedics.com/understanding-an-ecg/, accessed: 2018-05-14.
[10] M. K. Islam, N. Haque, G. Tangim, T. Ahammad, and M. R. H. Khondokar, "Study and analysis of ecg signal using matlab and labview as effective tools," *The International Journal of Computer and Electrical Engineering*, vol. 4, no. 3, pp. 404–408, 01 2012.
[11] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.
[12] K. Havelund and A. Goldberg, "Verify your runs," in *VSTTE 2005, Revised Selected Papers and Discussions*, ser. LNCS, vol. 4171. Springer, 2008, pp. 374–383.
[13] A. Bauer, M. Leucker, and C. Schallhart, "Runtime verification for LTL and TLTL," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 4, pp. 14:1–14:64, Sep. 2011.
[14] Y. Falcone, J. Fernandez, and L. Mounier, "Runtime verification of safety-progress properties," in *Runtime Verification 2009*, ser. LNCS, vol. 5779. Springer, 2009, pp. 40–59.
[15] S. Pinisetty, T. Jéron, S. Tripakis, Y. Falcone, H. Marchand, and V. Preoteasa, "Predictive runtime verification of timed properties," *Journal of Systems and Software*, vol. 132, pp. 353 – 365, 2017.
[16] D. Basin, F. Klaedtke, and E. Zălinescu, "Algorithms for monitoring real-time properties," in *Runtime Verification*. Berlin, Heidelberg: Springer, 2012, pp. 260–275.
[17] K. Havelund, "Rule-based runtime verification revisited," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 2, pp. 143–170, Apr 2015.
[18] S. Pinisetty, P. S. Roop, S. Smyth, N. Allen, S. Tripakis, and R. V. Hanxleden, "Runtime enforcement of cyber-physical systems," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 5s, Sep. 2017.
[19] Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam, "Modeling and verification of a dual chamber implantable pacemaker," in *TACAS*, ser. TACAS'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 188–203.
[20] C. Camara, P. Peris-Lopez, and J. E. Tapiador, "Security and privacy issues in implantable medical devices: A comprehensive survey," *Journal of Biomedical Informatics*, vol. 55, pp. 272 – 289, 2015.
[21] "PhysioNet," http://physionet.org/, accessed: 2018-05-14.
[22] http://www.ni.com/tutorial/6349/en/, accessed: 2018-05-14.
[23] "Uppaal DBM Library," http://people.cs.aau.dk/~adavid/UDBM/, accessed: 2018-05-14.
[24] https://www.livescience.com/42081-normal-heart-rate.html, accessed: 2018-05-14.