Title	ウェアラブルデバイスを用いた適切なメッセージ到着通知タイミングの検出
Sub Title	Detection of a suitable message arrived notification timing using a wearable device
Author	安藤, 亮介(Andō, Ryōsuke) 加藤, 朗(Katō, Akira)
Publisher	慶應義塾大学大学院メディアデザイン研究科
Publication year	2019
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2019年度メディアデザイン学 第746号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40001001-00002019-0746

修士論文 2019 年度

ウェアラブルデバイスを用いた
適切なメッセージ到着通知タイミングの検出

慶應義塾大学
大学院メディアデザイン研究科

安藤　亮介

本論文は慶應義塾大学大学院メディアデザイン研究科に
修士 (メディアデザイン学) 授与の要件として提出した修士論文である。

安藤　亮介

研究指導コミッティ：

加藤 朗 教授	(主指導教員)
前川 マルコス　貞夫 専任講師	(副指導教員)

論文審査委員会：

加藤　朗 教授	(主査)
前川 マルコス　貞夫 専任講師	(副査)
ムハマド ヤメン・サライジ 特任講師	(副査)

修士論文 2019 年度

ウェアラブルデバイスを用いた
適切なメッセージ到着通知タイミングの検出

カテゴリ：サイエンス / エンジニアリング

論文要旨

近年スマートフォンの普及に伴い、コミュニケーションに要する時間が短くなってきている。コミュニケーションを試みても返信が無い・メッセージを読まないなどのディスコミュニケーションが深刻な問題になってきている。ディスコミュニケーションは複数の要因によって生ずるが、特に通知のタイミングが悪かったり、再通知をする機能がない場合に、発生することが多い。

そこで本研究では、コミュニケーションを取りやすいタイミングを示す指標として、丁度良さ指数（Just In Timing index）を定義し、その値によってコミュニケーションがどう変わるのかについて論ずる。身体の動きや体位の変化、心の状態は脈拍数に大きく影響することが知られている。そこで、丁度良さ指数を脈拍数から推定し、ディスコミュニケーションの発生割合、コミュニケーションの成功までに要した時間との関係性について評価をおこなった。

その結果、丁度良さ指数の立ち上がりエッジにコミュニケーションが成功しやすいことが観測された。このタイミングでメッセージを送ると、返信率が高く、返信までにかかる時間が短くなることが確認された。

心拍センサーを誰しもが身につけている時代が到来しようとしている。本論文で定義した丁度良さ指数が通知の機能などに組み込まれれば、ディスコミュニケーションの軽減のみならず、効果的な広報・マーケティングへの活用や、様々な分野での活用が期待される。

キーワード：

ディスコミュニケーション, 脈拍数, メッセージングシステム, タスク割り込み, 割込み拒否度, 状況推定

慶應義塾大学大学院メディアデザイン研究科

安藤　亮介

Abstract of Master's Thesis of Academic Year 2019

Detection of a Suitable Message Arrived Notification Timing Using a Wearable Device

Category: Science / Engineering

Summary

It's been a while since smartphones became widespread, time required for communication has become shorter. Discommunication, such as no response or reading messages even attempting communications, has become a serious problem. Discommunication is caused one or more of the various different reasons, such as the timing of the message arrival was bad when the receiver was extremely busy, or no proper re-notification mechanism is implemented.

In this thesis, "Just In Timing" index is defined to indicate the degree of acceptance of the incoming messages, and studied how it affect the communication. It is known that the body movement and the state of mind affect the pulse rate. It is developed a way to estimate "Just In Timing" index based on the pulse rate and evaluated based on the discommunication occurrences and the duration to the responses.

As the result, it is observed that timely communication is likely conducted at the timing of rising edge of the "Just In Timing" index. When a message is sent or notified at one of these good timings, it is confirmed the increase of response rate and shorten duration to the responses.

Many users have wearable devices in these days and pulse rate is measured continuously. When the notion of "Just In Timing" index is built into the message notification system, discommunication may be mitigated. It is also expected

that “Just In Timing” index will be applied various situations including effective announcement and advertisement.

Keywords:

discommunication,heart pluse,messaging system,task interuption,state estimation

Keio University Graduate School of Media Design

Ryosuke Ando

目　　次

図　目　次

表　目　次

第 1 章
序　　論

1.1. 背景

近年スマートフォンの登場によりいつでも返信できるような新しいメッセージングの形が生まれた。e-mail などはコミュニケーションの返信に数時間から数日かかっていたものが、slack や LINE・facebook messenger では数分で返信をする、と言う文化を持った世代が存在する。コミュニケーションにかかる時間が短時間になり、意図しないディスコミュニケーションはとても深刻な問題になってきている。ディスコミュニケーションの発生の主な要因は通知の見逃しや無視・忘却などである。これを解決する一つの手段として、返信に対して適切な通知タイミングを調査し、コミュニケーションに与える影響を調べる。通知を遅らせたり、再通知の機能を強化すると言うアプローチでディスコミュニケーションを軽減できると考えている。

1.2. ディスコミュニケーションの発生

1.2.1 ディスコミュニケーションの定義

ディスコミュニケーションはコミュニケーションを取ろうとして、失敗したことをさす造語である。いくつかの論文にて、ディスコミュニケーションという単語が用いられているが、不明確であったために、本論文では以下のように定義した。コミュニケーションを試みたが、メッセージを読まなかったり、返信・返事をしなかったということをディスコミュニケーションとよぶ。似た言葉にミスコミュニケーション (Miscommunication) という言葉があるが、こちらは、ディスコ

ミュニケーションは発生しいないが、間違って伝わってしまうことや適切でない返信をしてしまったことであり、本論文で用いるディスコミュニケーションとは明確な違いがある。

ディスコミュニケーション発生の具体例

ディスコミュニケーションの具体例として、以下の事例を紹介する。ある教授に対してその教授の秘書が緊急性の高い確認事項があり、コミュニケーションを取ろうとしていた。秘書は教授が最も使いやすいコミュニケーションチャンネルを把握し切れておらず、facebook messenger や email をはじめとする様々なコミュニケーションチャンネルでコミュニケーションを試みた。しかし、教授はある作業に集中しており、返信をもらうことができなかった。そこで、研究室にいた生徒に尋ねると、教授とオンラインで繋がっていたため、その繋がっていたチャンネルにてコミュニケーションを試みると、コミュニケーションが成功した。適切なコミュニケーションチャンネルがあるとはいえ、新しくコミュニケーションチャンネルを増やすと、先ほど述べたようにディスコミュニケーションが発生する危険性が増加する。

ディスコミュニケーション発生要因

ディスコミュニケーションは複数の要因によって発生する。その中でも単純な要因としては以下のような要因に大別される。いくつかの要因では、適切なタイミング (適切な状態) で通知を送ることで解決できると考え、“JIT” を検知する。

- 返信するための時間を確保できなかった
- 返信するための端末を持っていなかった
- 返信することが面倒だった
- 返信を自分で判断できなかった
- 通知に気がつかない

- 通知を見たが後回しにしてしまう

1.2.2 メッセージングシステムに適切な通知タイミング “JIT”

本論文ではディスコミュニケーションを軽減する手法として、ウェアラブルデバイスなどのセンサを使ってメッセージの返信ができるタイミングやメッセージを読むことに嫌悪感を示さないタイミングを 丁度良いタイミング (“JIT”:Just In Timing) と定義し、ディスコミュニケーションの軽減を測った。また、ディスコミュニケーションを発生させないようなタイミングの指標として、“JIT” を定義する。

1.2.3 メッセージングシステムに適切な通知タイミングを示す指数 “JIT” index

本論文ではディスコミュニケーションを軽減する手法として、ウェアラブルデバイスなどのセンサを使ってメッセージの返信や読むことができる指数として、丁度良さ指数 ()“JIT” index:“Just In Timing” index) を定義する。メッセージを読むことに嫌悪感を示さない指数を “JIT” index と定義し、返信をしたかしていないかを基準として評価を行う。

1.3. ディスコミュニケーションを軽減させるための手段

ディスコミュニケーション（メッセージの返信や未読による問題）を軽減するための手段として、丁度良いタイミングやそのための指数を定義することが必要と考えた。ディスコミュニケーションが発生しずらい時に身体に現れる特徴や身体に現れる特徴を調べる。

コミュニケーションをおこう上では節 1.2.2 で述べたように、メッセージングシステムに適切な通知タイミング “JIT” が存在し、そのタイミングでのコミュニケーションは成功する確率は高いだろう。リアルタイムに “JIT” を推定し、それらを可視化や自動で制御するシステムが必要だ。

“JIT” の度合いとして “JIT” index を用い、評価を行う。コミュニケーションをいつ取ればコミュニケーションが取れるかや、通知を送るタイミングをシステムが制御しする際のタイミングの指標として用いることができる。“JIT” index を用いることで、通知のコントロールしディスコミュニケーションを軽減するのみならず、集中している作業を邪魔せずにコミュニケーションをとることで円滑なコミュニケーションができる。

第 2 章
関 連 研 究

2.1. ディスコミュニケーションの発生に関する研究

2.1.1 他研究でのディスコミュニケーション発生の定義

他の論文で用いられるディスコミュニケーションとは、コミュニケーションが成立しない状態のことをさす造語である [3] が、具体的に不明確な言葉であるため 1.2.1 章で述べたように定義した

2.1.2 コミュニケーションタイミングにおける拒否度

本研究の根幹となる丁度良いタイミングでコミュニケーションをとるタイミングに関する論文調査を行なった。一般事務のオフィスや研究開発現場の観察により、電子的なペーパーワークやプログラム開発などの個人作業は、窓口業務等を除く多くの従事者の業務の中核である。内容は、特許申請準備や論文の執筆のように集中を要する個人作業や、業務日誌の記入や郵便発送など比較的リラックスして遂行できるものまで様々であり図 2.1(藤田 欣也氏の論文 [4] より引用) のようにこれらの間を遷移しながら、作業を行なっている。さらに、時折、必要に応じて業務に関連した質問や相談連絡・調整といったローカルなコミュニケーションが発生する。すなわち、オフィスでは、個人作業と、完結的に発生する共同作業の間の遷移を繰り返している。

これらの作業の中間にコミュニケーションを発生させると、割り込みに影響を与えると考えられる。作業者からすると、割り込んで欲しくない度合い、つまり

割り込み拒否度が高い状況がある。平易な作業をしている時や集中していない時は拒否度が低くなると考えられ、作業が一段落したタイミングでは割り込み拒否度が低くなると考えられる。

さらに、作業状況以外にもオフィスワーカーの割り込み拒否の一因に、外部環境がある。コミュニケーション中や共同作業中などを通した他社との結合の程度は割り込み拒否度に影響すると考えられる。他にも、作業者を取り巻く周辺環境なども関係している [1] [4]。

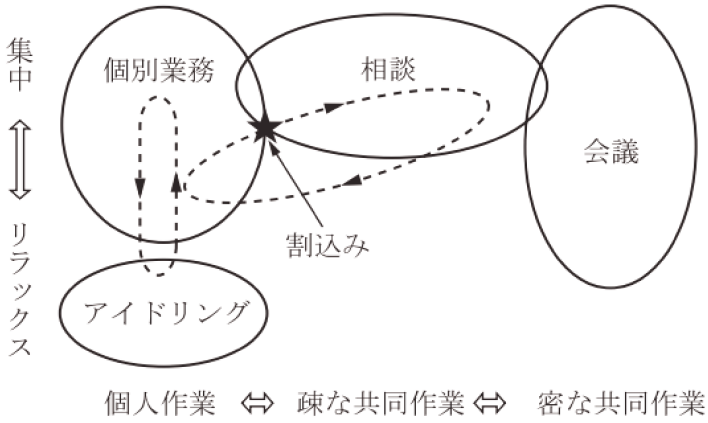


図 2.1 オフィスワーカーの状態モデル [1]

丁度良いタイミングは、オフィスワーカーに絞れば、状態モデルを定義できると考えた。

コミュニケーションをとるための適切なタイミング

コミュニケーションにおける受信に対して拒否を示さないタイミングを話しかけやすいタイミングを調査し割り込み拒否度という指数で評価を行う先行研究 [4]、図 2.2(大越 匡氏の論文 [5]Reducing Users’ Perceived Mental Effort Due to Interruptive Notifications in Multi-Device Mobile Environments より引用) に

示すようにメッセージングの通知に対して行動と行動の切れ目「Break Point」を通知に対するユーザー認知負荷を軽減するタイミングとしている先行研究 [6] [7] も存在する。通知に対する返信の有無、コミュニケーションの成功・失敗を心拍数から推定した指数は存在しないため、本論文では 1.2.3 章より “JIT” index と定義した。

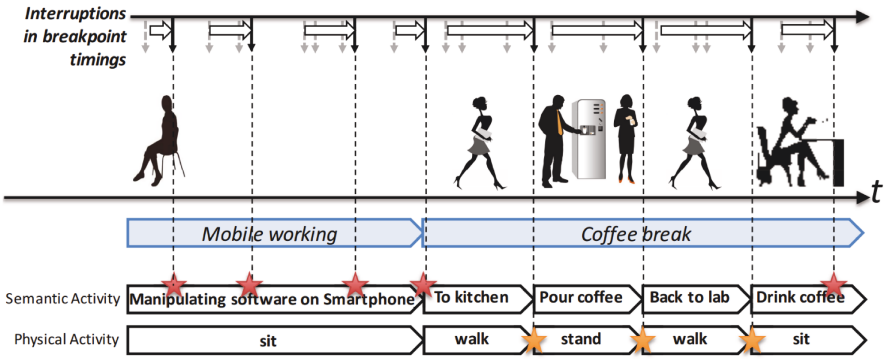


図 2.2 User ’ s Notification Experience Scenario with Attelia II

2.2. 生体情報と人間の行動の関連性

2.2.1 脈拍数と体勢の関係

図 2.3(古川 順光氏の論文 [8] 図 1 各体位における血圧と脈拍数の 2 分 30 秒から 30 秒間の平均値 (N=12) より引用) に示すように、体勢の変化は脈拍数と大きく関わることが知られている。健常者を対象とした時、背臥位、座位、立位などの能動的な体位変換時に脈拍を測定したときに、各体位における血圧と脈拍数には変化があり、退位変換直後の血圧と脈拍数に変化がある [9]。人間の体位変換時に脈拍数変動が現れることと、人間の集中している時時間との間（一段落したタイミング）に体位変化が生まれることに着目し、ディスコミュニケーションの減少ができると考えた。

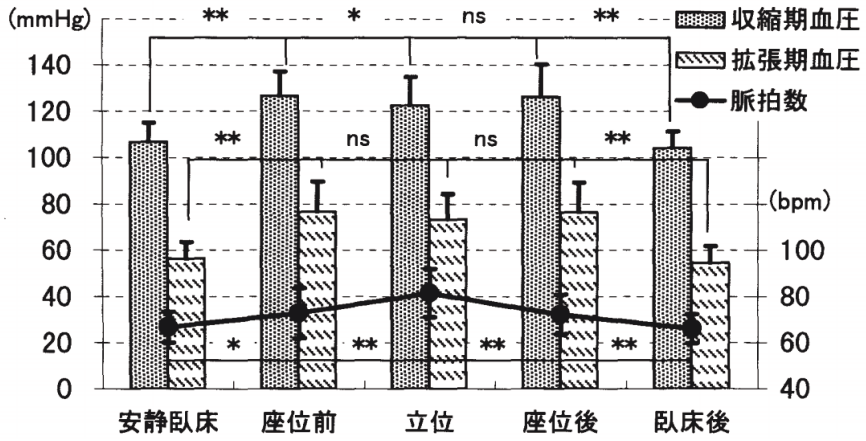


図 2.3 各体位における血圧と脈拍数の 2 分 30 秒から 30 秒間の平均値 (N=12)

2.2.2 生体情報のコミュニケーション活用

生体情報がコミュニケーションに与える影響を論じた論文がいくつか存在する。脳の血流量から感情を推定し、情動インターフェースとして活用し、コミュニケーションや情動体験に影響を与える [10] コミュニケーションの引き込み現象を脈拍数等により、評価を行う [11] など、コミュニケーションが人間の生体情報に与える影響は多くの論文で示されている。しかし、脈拍数などの生体情報から能動的にコミュニケーションへ影響を与えるようなシステムなどは存在しない。

2.3. コミュニケーションチャンネルの増加

コミュニケーションチャンネルやコミュニケーションのスタイルが増加している。大きく分けて表 2.3 のようになる。ディスコミュニケーションが発生しているチャンネルととディスコミュニケーションが発生し辛いチャンネルが存在し、リアルタイムにチャットをする必要がある場合に適しているチャットツールを選定し、合理的なチャンネルでコミュニケーションをとっていることがわかる。

表 2.1 Comparison of communication channel

-	call	e-mail	Individual chat	buisiness chat
Frequency of discommuniation	high	high	low	low
Certainty of read	high	low	high	high
Be able to transmit exactly mean	yes	no	no	no
Real time communication	good	bad	good	good
History of communication	no	yes	yes	yes
Interaption of other tasks	yes	no	no	no

2.3.1 e-mail

RFC822 [12] で記述された形式のテキストメッセージを RFC821 [13] で定義されている Simple Message Transfer Protocol で交換されるメッセージングシステムである。ユーザが使うインターフェースから、最寄りの MTA (Message Transfer Agent) に RFC822 形式のメッセージを送ると、MTA は受信者のメールアドレスに連携した MTA に対して、SMTP でメッセージを送信する。分散的に配信が行われるシステムを基本としている。到着したメッセージは、POP3 [14] や IMAP [15] などでアクセスすることができる。その後、メッセージに日本語など英語以外の言語での表記がサポートされたり [16]、SPAM 問題への解決の一助として、メールの送信時に認証をサポートする submission [17] などがサポートされている。相手のメールアドレスさえわかっていれば、文書だけでなく、データなどを添付することができる。電子メールには送信者の送付したタイミングに関わらず、受信者が任意のタイミングで読むことができる。

欠点を以下にあげる。

- 情報漏洩リスク
- ウィルスが添付される危険性
- 迷惑メールが多い
- 余分な文書（季語など）やテンプレート通りに送る文化が存在

- コミュニケーションの発信元に読んだことを通知する機能が無い
- 文字エンコードが統一されていない場合があり、環境依存が多い

2.3.2 個人間チャットシステム

個人間のチャットシステムには Facebook Messenger や LINE、WeChat をはじめとするチャットシステムが存在する。電子メールと同様に受信者が任意のタイミングで読むことができ、任意のタイミングで返信することが可能なシステムとなっている。Facebook Messenger のように Facebook の一機能としてローンチされたものもあれば、コミュニケーションを主体に開発されたものも存在する。いずれも、email に比べ、コミュニケーションのコストが掛からず、簡単にメッセージをやりとりすることが多い。

個人間チャットシステムには以下のような特徴がある。

- 一点集中管理
- スレッド型メッセージが多くを占める
- 承認制であることが多く、メールに比べ安全
- コミュニケーションの発信元に読んだことを通知する機能がある (既読機能)
- 既読機能があるので、未読のまま無視する文化が存在

Facebook Messenger

Facebook Messenger はインスタントメッセージサービスやテキスト通信を行うアプリケーションソフトウェアである。基本的には Web ベースのチャット機能を中心にスタートし、web ブラウザやスマートフォンアプリから、本サービスを利用することができる [18]。Facebook Messenger のアプリケーションインターフェースは図 2.4 図 2.5(App store ページ [4] より引用) のようになっている。

図 2.4 Chat Room of Facebook Messenger

図 2.5 Facebook Messenger chat select

LINE

LINE は、LINE 株式会社が開発したソーシャル・ネットワーキング・サービス (SNS) である。スマートフォンとパソコンに対応し、通話機能とテキストチャットの機能などを有し、スタンプ機能などが特徴的だ。

日本では個人間のコミュニケーションツールとして広く普及しており、若年層から中高年層にも慕われるシステムとなている。

図 2.6(line.me [19] より引用) のように、通常のコミュニケーションに加えて、スタンプを送信することができスタンプのみで会話する文化がある。既読機能やファイルを送信する機能も有しており、日本国内では LINE Pay などを運営するなど、日本の社会インフラとなっている [19]。

WeChat

WeChat は、中華人民共和国の IT 企業テンセントのリアルタイムコミュニケーションアプリである。グレート・ファイアウォールによる検閲の対象となっていないアプリの一つで、Facebook Messenger などを使えない中国国内のユーザーを

図 2.6 Screen Shot of LINE Application

中心にサービスを展開している。図 2.7(wechat.com [20] より引用) のように競合のアプリと同様の機能を有している [20]。

2.3.3 グループワークチャット

主にある組織に所属する個人がコミュニケーションをとるために用いられるものになっている。主に以下のような特徴がある。

- 一点集中管理
- 社内・グループ内などの許可された人のみ参加可能
- 迷惑メールが少ない
- コミュニケーションコストが低い
- チャット以外の機能が豊富

図 2.7 Screen Shot of WeChat Application

slack

slack は 2013 年にローンチされ、その直後から、急速に世界中で普及し続けるグループチャットである。2019 年現在も、利用者を増やし続けている。主に、オフィスを離れてやりとりする時に便利なビジネスチャットツールで、無料で始められることが大きな強みである。

他のチャットツールと同じように複数人でのグループトークを用いることができる。ワークスペースごとにメンバーとの会話ができ、チャンネルごとに会話することができ、その中にさらにコミュニケーションスレッドを立ち上げコミュニケーションをとることが特徴的だ。また、slack はプラグインや拡張機能が充実しており、様々なアプリケーションと連携できることが知られている [21]。

図 2.8(slack.com [21] より引用) のようなインターフェースになっている。

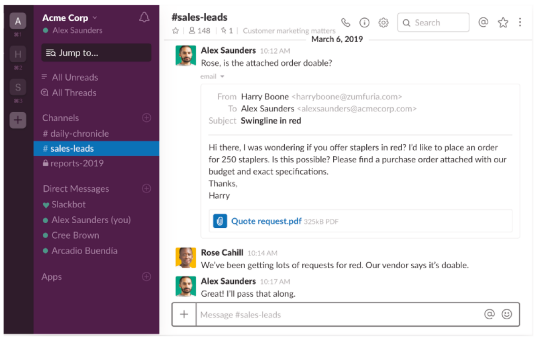


図 2.8 Slack Application

Microsoft teams

Microsoft teams は、コミュニケーションツールと同様に、インスタントメッセージや通話の機能を有している。他のコミュニケーションチャンネルと比べて、多くの機能を有しており、ファイル管理・メモの共有・スケジュール管理などのビジネスシーンで役立つ機能が標準搭載されている。また、wiki などを簡単に作ることができ、社内のコミュニケーションを円滑に進めることが簡単にできるようになっている [22]。図 2.9(products.office.com [22] より引用) のようなインターフェースになっている。

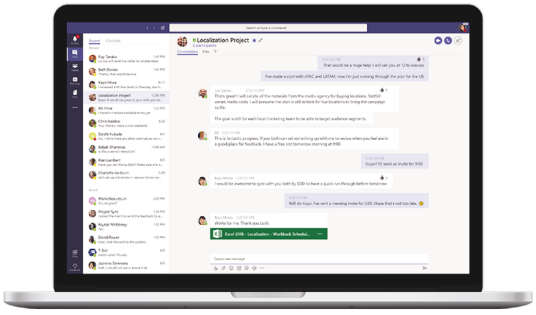


図 2.9 Microsoft teams

2.3.4 トラブルチケットシステム

トラブルチケットシステムは一つの障害に対して一つのチケットが発行される。内容は障害の発生日時、発生箇所、報告者、作業担当者、障害内容、対応優先度を含む。作業の状況や復旧までの作業履歴が書き込まれ、障害の発生から収束までが記録される。タスク一覧や作業の進行状況を可視化するタスク管理のような機能を有する。主にセキュリティや医療などの緊急対応が必要な分野で用いられる [23]。

2.4. コミュニケーションツールに於ける重要性緊急性機能

2.5. 心拍検出デバイス

2.5.1 脈拍数をトラッキングできるウェアラブルデバイス

fitbit charge3

Fitbit Charge 3 は、fitbit が発売しているフィットネストラッカーの一つである。心拍のトラッキングや GPS トラッキングが可能で、今回は脈拍の記録の機能単体として利用する。fitbit charge 3 は図 2.12(fitbit.com [24] より引用) に示すように、腕に装着可能。外観は図 2.13 のようになっている。

Garmin foreathlete 235j

手首に装着するウェアラブルウォッチである。光学式心拍計や VO2MAX センサ、加速度センサ、GPS などを搭載しており、ランニングや自転車等の運動に特化したウェアラブルデバイスである。図 2.11 のように手に装着可能である。

Apple watch

Apple Watch は、腕時計型端末の一つであり、光学式心拍センサーと電気式心拍センサー、マイク、スピーカー、加速度センサー、環境光センサーを搭載している。(世代により多少の差異は存在する。) ジャイロセンサーの搭載により、転倒・落下を検出することができ、1 分間動かないと、自動的に緊急通報を行う機能 (転倒検出) も搭載されている。図 2.10 のように手に装着可能である。

図 2.10 Apple Watch

図 2.11 Garmin foreathlete 235j

図 2.12 Fitbit Charge 3

Ant・ant plus

Ant は超低消費電力のワイヤレス通信プロトコルの一つである。個人が使用する機器同士を接続するネットワーク機器向けのプロトコルで、非常に近距離（数 [m] 程度）の通信に限られる。超低消費電力であるため、ウェアラブルウォッチなどの電力が限られた状況でも通信可能だ。また、ANT は 2.4GHz 帯を利用しており、無線免許所持なしに運用ができ、ネットワーク網を柔軟に構成できる。通信は Master と slave が存在し、複数 Master と通信することが可能であり、様々なネットワーク網を構築できる。

Ant plus は相互運用を保証するプロトコルとして、ant を採用したデバイスが相互に通信できるようにした共通使用だ。ANT+ Alliance に加盟した各社が販売

するデバイスを相互にテストし、互換性を保証している。プロトコルには心拍数や歩数などの生態情報、運動情報、フィットネス用データなどのやりとりが可能である [2]。

Ant plus のドングルやデバイスは図 2.13 があり、 本論文で使用する Garmin foreathlete 235j2.5.1 とも通信することができる。

外観は図 2.13(upthejunction.com [2] より引用) のようになっている。

図 2.13 Ant plus dongle [2]

第 3 章
提　　案

3.1. 現状の問題点

3.1.1 ディスコミュニケーションの発生タイミング

現在のシステムでは様々な問題があり、ディスコミュニケーションは集中している時や一人で休んでいるときに発生しやすいことがわかっている。そこで図 3.1 に示すように、一人で働いている時、グループで働いている時、休憩中、チャットコミュニケーションの 4 つの状態にわけ、その状態の遷移中であれば、ディスコミュニケーションが軽減されると考えた。

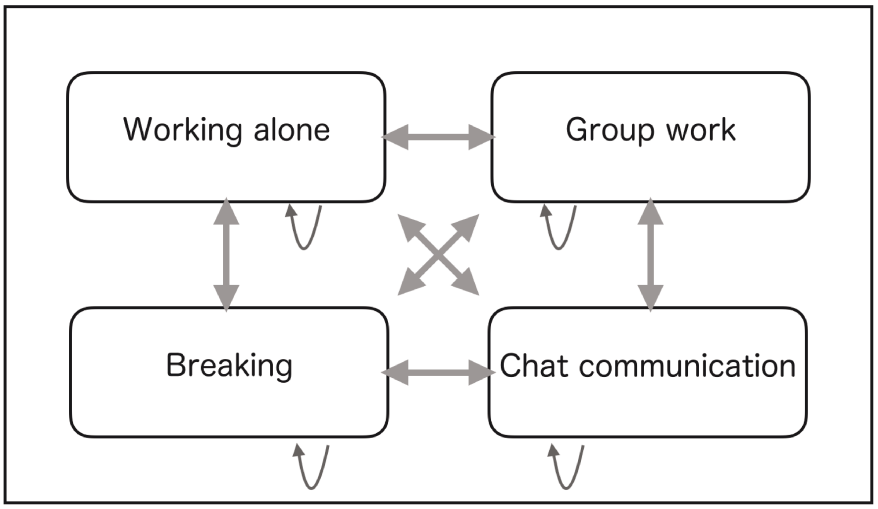


図 3.1　State transition

3.1.2 通知とディスコミュニケーションについての関係性

1. 通知が送られているにもかかわらず、通知を見逃す
2. 通知に気がついたが、読めないため後回しにしてしまう
3. 返信に対応できず、後回しにしてしまい忘れてしまう
4. コミュニケーション可能な端末を所持していない
5. 通信相手が忙しく、忖度してしまう
6. システム管理者が通知発行を設定忘れ
7. 読んでいないチャンネルがあるため、コミュニケーションを試みても、ディスコミュニケーションが発生してしまう
8. 相手がどのチャンネルを好んで利用しているか分からないため、コミュニケーションを複数チャンネルで試みて、コミュニケーションコストがかかる
9. チャットチャンネルの増加により、最も連絡の取りやすいチャンネルがわからない
10. チャットチャンネルの増加により、どのチャンネルで来た連絡なのかわからなくなってしまい、ディスコミュニケーションが発生
11. チャットチャンネルの User Exprience がそれぞれ違うため、誤送信などが発生
12. チャットチャンネルごとに文化があり、それぞれに学習コストが多くなる
13. PC やスマートフォンなどクライアントが多様化しており、PC で読むべき文書をスマホで受信してしまうなど、ディスコミュニケーションが発生
14. 業務時間外

などの問題点がある。その中でも、1. 2. 3. 4. 5. のディスコミュニケーションの発生に関し、タイミングを調整するなどのアプローチで、解決が可能と考えた。

3.2. 提案手法

3.2.1 脈拍情報から “JIT” を検知するアルゴリズム

メッセージを届けたい相手の生体情報から、“JIT” を検知し、通知のタイミングを制御したり、メッセージを受信可能な時間帯を事前に知ることができるシステムを提案する。そのために必要な脈拍数から “JIT” を検知するためのブラックボックスを図 3.2 のように定義し、ブラックボックスのフィルタを作成する。

また、フィルタ (1) および、フィルタ (2) はデジタルローパスフィルタを設計し、フィルタの係数と次数を変更可能とする。filter の係数はカットオフ周波数と密接に関係しており、また、カットオフ周波数はフィルタの周波数領域において input の値が通過域平坦部の $\frac{1}{\sqrt{2}}$ 倍となる部分と定義する。すると、式 3.1 のような式で、カットオフ周波数を求めることができる。
ただし、
F_{sample} サンプリング周波数、
r をフィルタの係数
とする。

$$\frac{F_{sample}}{2\pi} \arccos\left(\frac{2-2r-r^2}{2*(1-r)}\right) \tag{3.1}$$

図 3.2 脈拍数から “JIT” を検知するためのアルゴリズムの提案

そのために、“JIT” を図 3.3 のように作業中と作業中の間、休憩と休憩の間等の遷移時（エッジ部分）が最も良い部分であると考えた。生体情報から “JIT” を検知する。

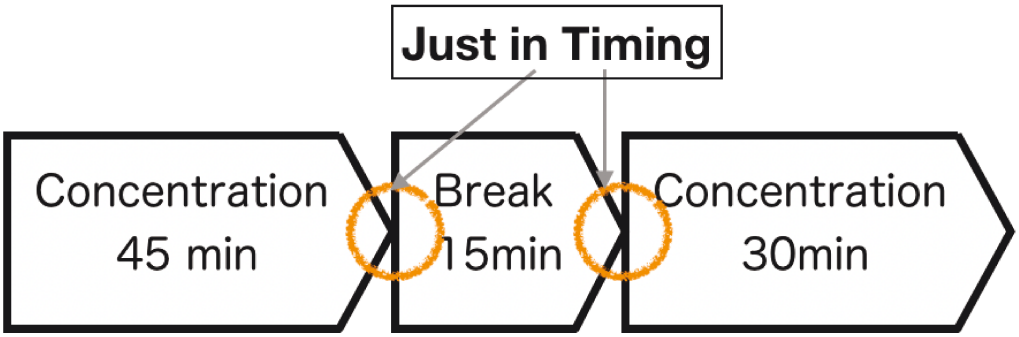


図 3.3 Relationship between “JIT” and concentration

3.2.2 “JIT” index により溜め込み・再通知するシステム

図 3.4 に示すように、トラッキングした “JIT” index から相手のメッセージシステムをスタックさせ、re-notify するシステムを提案する。通知を見逃したとしても、再通知の機能によって、ディスコミュニケーションの発生確率を減らすことができ、コミュニケーションを円滑に進めることへの一助になると考えている。

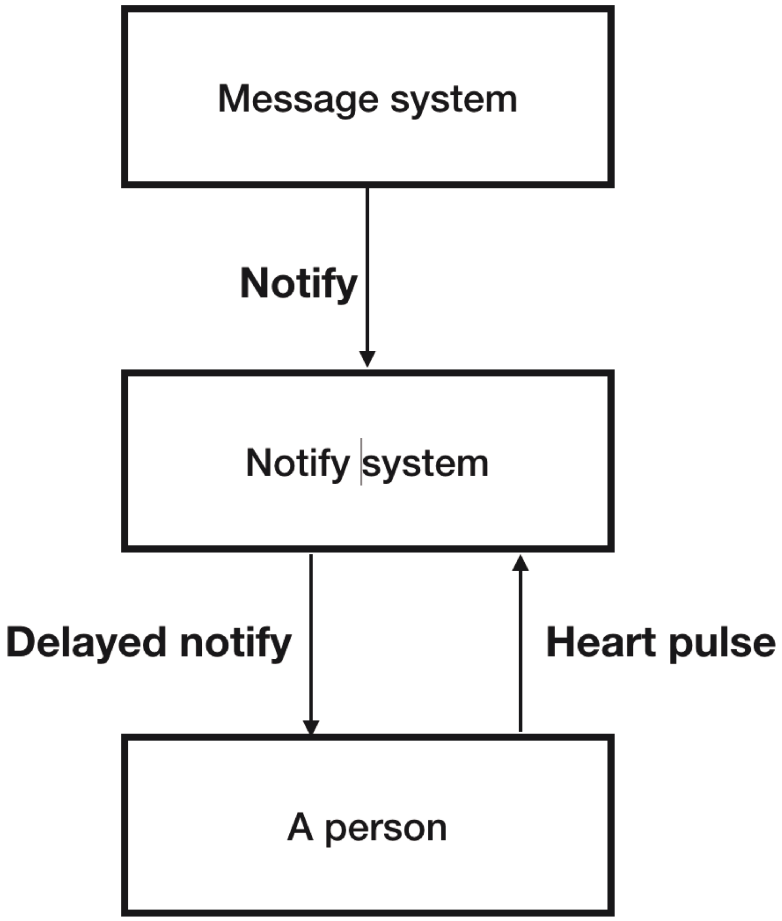


図 3.4 Re-notify system diagram

図 3.5 のようにメッセージの通知を通常通りに行い、届いた通知を再通知するシステムを提案する。また、ため込むことによって、現在進行しているタスクを中断する必要がなくなるため、タスクに対して作業の割り込みがなく、作業に集中できる利点がある。しかし、緊急を要する連絡には対応できないため、ディスコミュニケーションを誘発する可能性がある。緊急性を要するコミュニケーションに対しては別途注意する必要がある。

また、図 3.6 のように通知を行う前に MAC OS X に搭載されているおやすみ

モードのように、通知を一時的に止め丁度良いタイミング “JIT” に一斉に送るシステムを提案する。

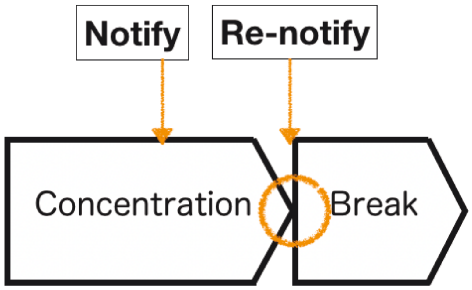


図 3.5 System of re-notify using “JIT” index

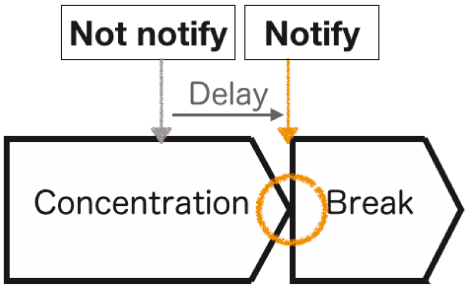


図 3.6 Delaied notify system using “JIT” index

3.2.3 “JIT” index 相手の状態を可視化するシステム

相手の状態を事前に知ることができれば、相手の “JIT” をある程度知ることができるので、忖度してメッセージすることの一助となり、ディスコミュニケーションを軽減できる。そのために、図 3.7に示すような、“JIT” index により相手の状

0 Icon made by freepik from www.flaticon.com

態を可視化するシステムを提案する。しかし、コミュニケーション相手の状態をリアルタイムに表示することは、トップレベルな個人情報である脈拍数を公開することになる。プライバシーの機能に関しては別途考察が必要である。

図 3.7 相手の “JIT” を事前に知ることができるシステム

3.3. 設計

“JIT” index を検知するプログラムを図 4.4 のように定義した。前処理として、LPF(ローパスフィルタ) を挿入し、その移動偏差の偏差値を求め、LPF された脈拍数と掛け合わせ、これ “JIT” index と定義した。安定性検知をするために移動分散のアルゴリズムや、微分した値の定積分値などを組み込むアルゴリズムなどを提案する。

また、再通知や通知保留型の機能に関しては “JIT” の情報はメッセージングシステムのユーザーインターフェースに組み込む。コミュニケーション相手に通知する機能は必要なく、自分の心拍情報を含む生体情報を外部に後悔しなくても良いので、プライバシーの保護にも繋がる。

第 4 章
実　　装

4.1. ディスコミュニケーションを軽減するシステムの設計

3 章で述べたように、通知の見逃しや、後回しにしてしまい忘却してしまった場合に関して、ディスコミュニケーションを軽減するために、再通知を行うようなシステムを設計する。主に通知タイミングはウェアラブル端末の心拍センサーから “JIT” を推定し、“JIT” index を定義し、ディスコミュニケーションの軽減を図る。通知を見逃した場合には再通知を行うことで解決する。

4.1.1 “JIT” を検知するための機能

“JIT” を検知するためにバイオロジカルデータを取得する。本実装では fitbit Charge3 を用いて実装し、リアルタイムに “JIT” ・“JIT” index の推定を行う。また、サンプリング周波数 0.2[Hz] で過不足なく、“JIT” index を推定することができるかも同時に検討する

4.1.2 通知タイミングによるディスコミュニケーション軽減システム

また、通知のタイミングが悪く対応できない場合に対する解決策として、コミュニケーションの “JIT” を検出し、その “JIT” で通知を行う必要がある。そこで、脈拍数から “JIT” を検出し、“JIT” で re-notify する機能を実装することにした。

4.1.3 受信拒否度の可視化システムによるディスコミュニケーション軽減システム

また、“JIT” をリアルタイムで外から可視化する機能を実装し、相手の状態を事前にしることで、相手の “JIT” にメッセージを送信するタイミングを検討する一助になるシステムを構築する。

4.2. 実装要件

今回の実装では以下のような機能を実装する。丁度良さ指数を定義するために、バイオロジカル情報をトラッキングし、脈拍数の安定度合いや高さを元に推定を行う。

- 脈拍数に時刻を付加し記録する機能
- 脈拍数安定度合いや脈拍数の高さを可視化する機能
- 自分が返信できるタイミングを記録する機能

4.3. 概略図

4.4. ウェアラブルデバイスを用いた心拍検出の解析アルゴリズム提案のための予備実験

4.4.1 目的

ウェアラブルウォッチを用い、脈拍などの情報を計測し、心拍と “JIT” にどのような相関が存在するのかを分析する。人間の行動と脈拍数の関係について考察する。

4.4.2 手法

図 4.1 に示すようにプロトタイプし、付録 2 に示す python スクリプトで解析を行う。スマートウォッチからスマートフォンやサーバーとの通信は既存の Fibbit から提供されているものを用いた。

脈拍データ記録手法

ウェアラブルウォッチ (fitbit Charge3) を用い、脈拍などの情報を計測し、fitbit 社のサーバーへためる。そのバイオロジカルデータをダウンロードするためのスクリプトを別途用意し、csv データとして保存した。また、“JIT” と “JIT” ではない時刻を csv データとして記録する。

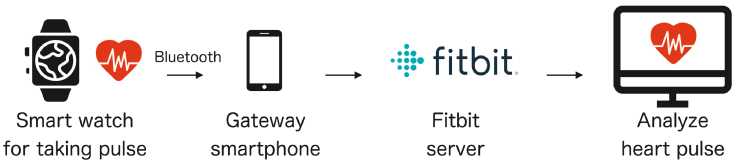


図 4.1　ウェアラブルデバイスを用いた心拍検出の解析プロトタイプ構成[1]

解析に用いる手法

解析には python スクリプト 2 を用い、グラフ化することで、解析を行なった。また、“JIT” と “JIT” ではない時刻をラベルとして使用した。

4.4.3 結果

実験を行った。脈拍数と時刻情報の関係を図 4.2 へ示す。また、行動の大まかなに 5 種類に分類したラベルを示す。

1　Icon made by Freepik from www.flaticon.com
Icon made by Smashicons from www.flaticon.com

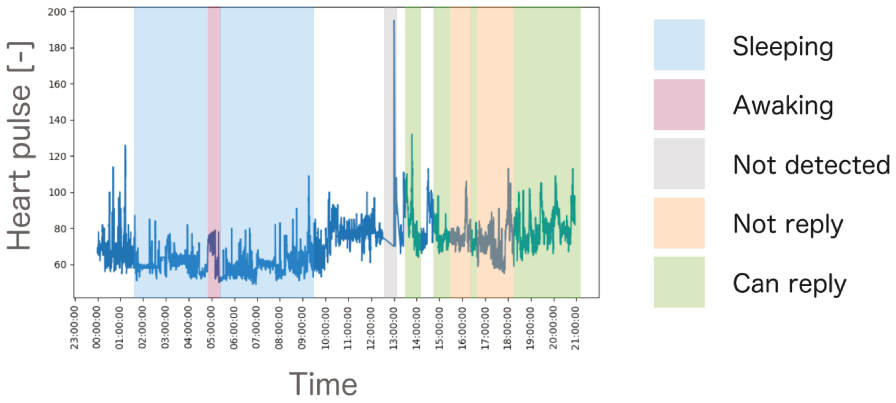


図 4.2　Data of heart pulse at 2019/10/14

4.4.4　考察・提案

実験結果より以下のことがわかった。

- 心拍が低い時は睡眠中・リラックス時で返信したくない・返信をしていない状態
- 脈拍数のばらつきが多いほど、返信をできる状態の割合が高い

そこで、脈拍数の高さ・不安定性を “JIT” の指数として組み込む。不安定性の検知には、過去 n 回のデータを母集団とする標準偏差を移動標準偏差と定義し、移動標準偏差アルゴリズムを用いることとした。また、脈拍数の高さや不安定性には個人差があるために、過去の平均値などから、偏差値をとり、人による変化少なくなるようにした。

4.5. ウェアラブルデバイスを用いたステータス検出に関する予備実験

4.5.1 目的

4.4 節で提案したアルゴリズムの設計とその検証をする。人間の行動と定義した丁度良さ指数にどの程度相関があるのか検討する。

4.5.2 手法

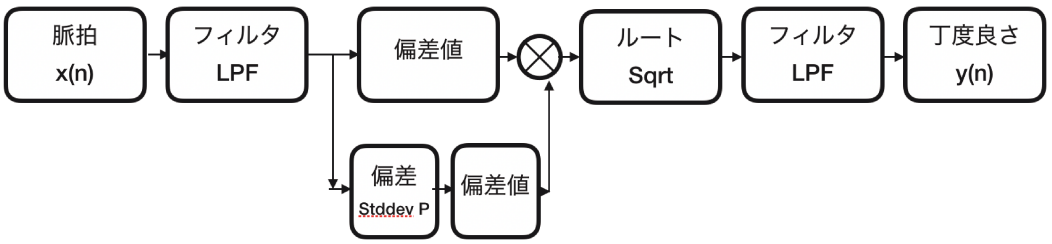
ウェアラブルウォッチ（fitbit）を用い、脈拍数を検知する。そこで、脈拍数の安定性と、脈拍数の高さを用いた、図 4.3 ようなシステムを提案する。丁度良さ指数は図 4.4 のように実装し検出する。

図 4.3　A system detecting heartpulse using a wearable device[2]

4.5.3 手順

スマートウォッチ、heart rate sensor、スマートフォン、返信できるタイミングを記録するボタンを用いて、時間を記録する。
被験者はスマートウォッチを装着し、普段通り生活をお願いし、脈拍データを数秒～数十秒に 1 度記録する。

2　Icon made by Freepik from www.flaticon.com
Icon made by Smashicons from www.flaticon.com



下記パラメータを変更可能

1. フィルタ(1)の次数
2. フィルタ(1)の係数
3. 分散の母体の個数
4. フィルタ(2)の次数
5. フィルタ(2)の係数

図 4.4　Prototype of estimation a “JIT” index

返信できるタイミングにスマートフォン上で動作するボタンを押してもらい、メッセージ通知が読まれるタイミングやメールや LINE メッセージに返信しやすいタイミングの教師データとして使用する。また、返信できると推定されるタイミングでコミュニケーションを試みて、コミュニケーションが成功するかどうかを確認し、これらを比較する。

実験場所：慶應義塾大学大学院および自宅、生活範囲全域

期間：2019 年 11 月から 2020 年 2 月

4.5.4 結果

実験結果を図 4.5 に示す。予備実験の生データから以下のことが読み取れた。

- 脈拍数が安定している時より不安定な時に返信できる割合が高い
- 脈拍数が低い時はリラックスしている、または休憩中であり、返信をする割合が低い
- 心拍数が極端に高い時は運動中などで、返信できない割合が多い
- 心拍数がある程度高く、脈拍数が乱れている時は返信できる割合が高い

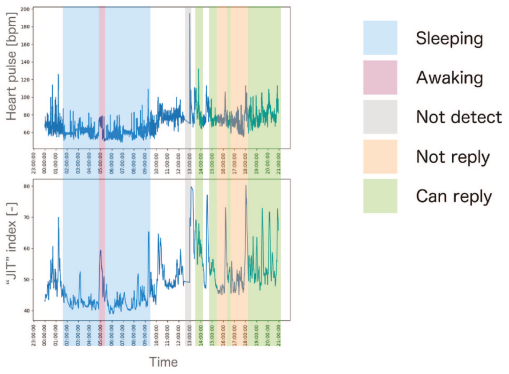


図 4.5　Result of preliminary experiment

睡眠中の “JIT” index

移動中の “JIT” index を図 4.6 に示す。睡眠中の “JIT” index は全体的に心拍数が低いことが読み取れる。リラックスして落ち着いているために心拍数が低く、感情の起伏などが少ないためと考えられる。

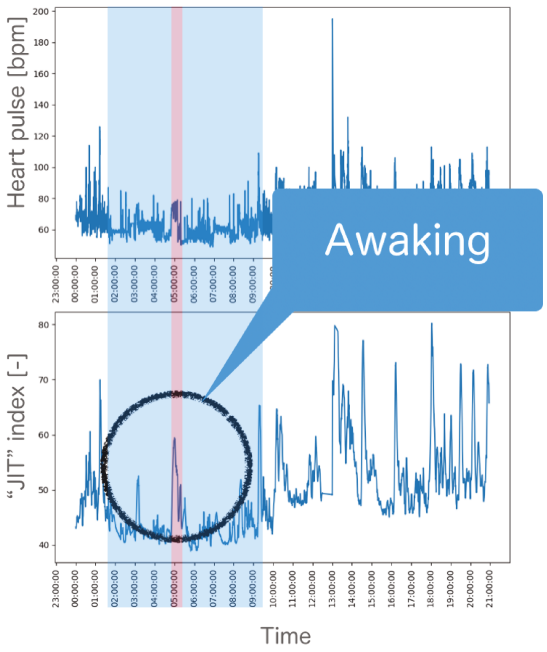


図 4.6　“JIT” during sleeping

移動中の “JIT” index

移動中の “JIT” index を図 4.7 に示す。移動中の “JIT” index は高いことがわかった。移動中は電車移動の間や待ち時間などが多くスマートフォンを見れるタイミングが多かった。その様子が “JIT” index にも現れていると言える。

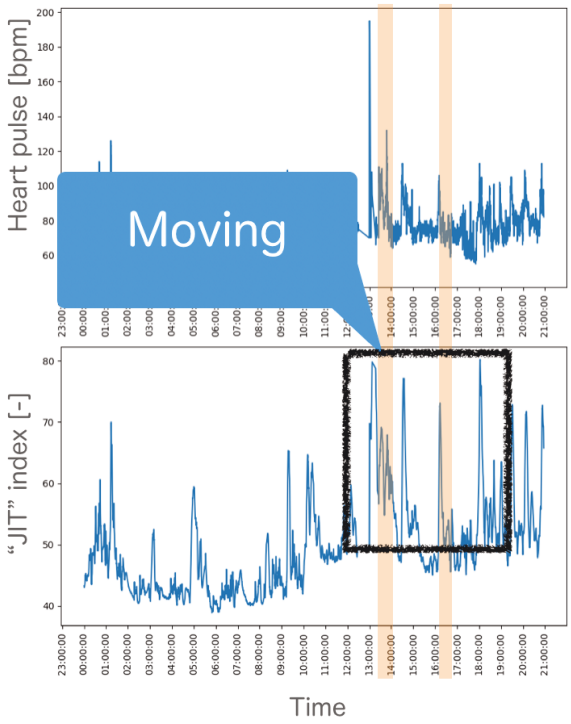


図 4.7 “JIT” during moving

仕事中の “JIT” index

仕事 (PC 作業) 中の “JIT” index を図 4.8 に示す。仕事中の環境では “JIT” index は低く、返信できないことが多かった。その様子が “JIT” index にも現れていると言える。

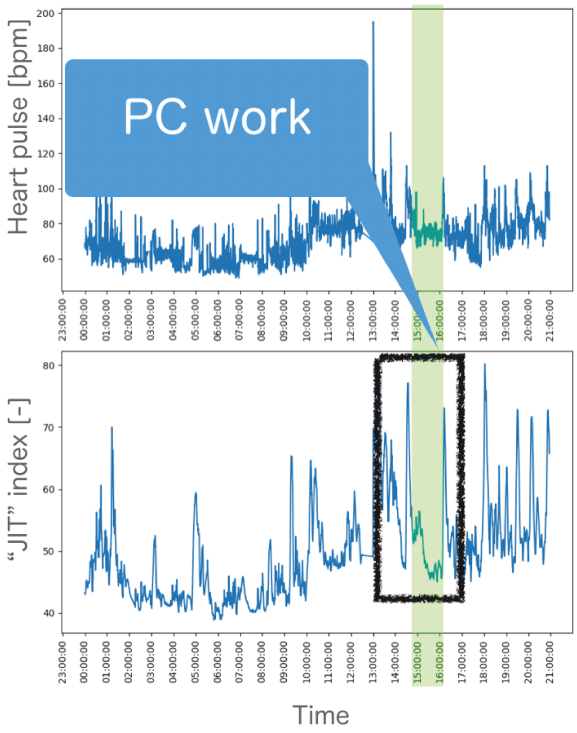


図 4.8　“JIT” during working with PC

研究活動中の “JIT” index

研究活動 (カフェでの PC 作業) 中の “JIT” index を図 4.9 に示す。カフェでの PC 作業中は “JIT” index は低く現れている。これは、カフェという集中している環境で、コミュニケーションその様子が “JIT” index にも現れていると言える。

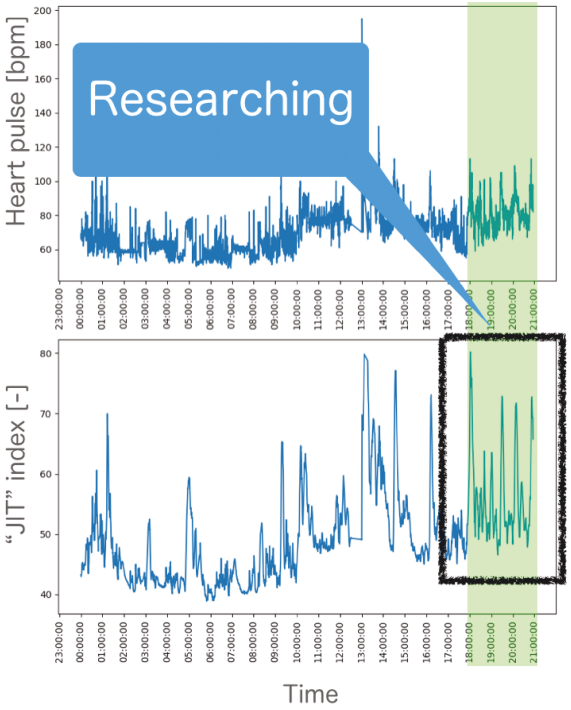


図 4.9 “JIT” during researching

“JIT” index プロトタイプの評価

実験の結果、被験者が “JIT” と評価したラベルと “JIT” ではないと評価したラベルで丁度良さ指数をヒストグラムにした。その結果を図 4.10 に示す。

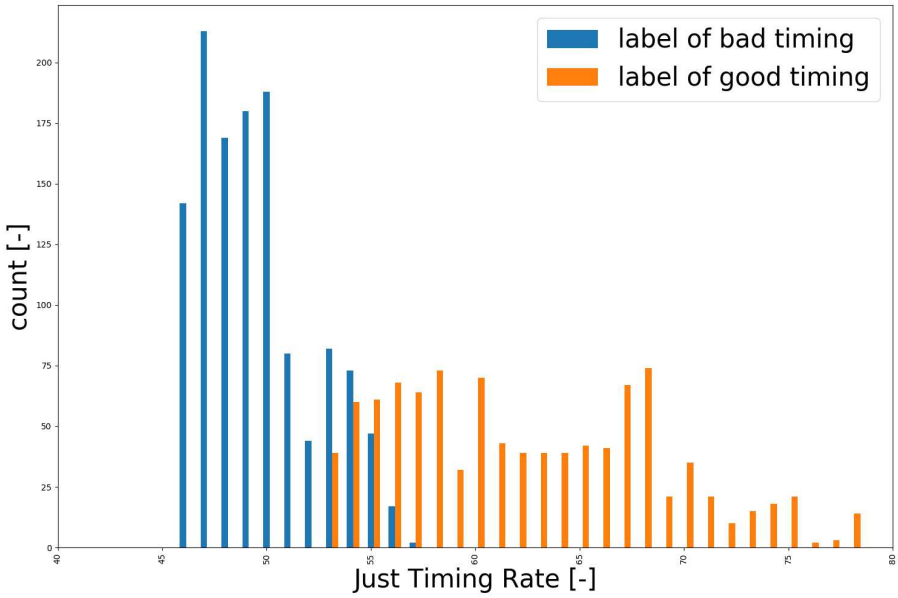


図 4.10　Histogram of “JIT” index

4.5.5　考察・提案

予備実験より、この被験者において、被験者が “JIT” と評価したラベルと “JIT” ではないと評価したラベルと丁度良さ指数に相関があることを確認した。別の被験者やサンプリングレートが 0.2[Hz] から変更された時、本フィルタや本アルゴリズムでどう変わるのかを比較検討するアルゴリズムを提案する必要がある。

そこで、より高速な微分アルゴリズムでエッジ検知を行い、丁度良さ指数を計算する。

fitbit Charge3 ではスマートフォンアプリの立ち上げと同時にデータを同期するためリアルタイム性に乏しく、ディスコミュニケーションの軽減の一助とはなら

ないことが判明した。そこで、Garmin Forethlete 235J および、 通信規格である ant plus を用い実装し、リアルタイム性を確保する。

4.6. 予備実験を踏まえたシステム設計

4.5 節より、リアルタイム性を確保できる Garmin Forethlete 235J を用いることとした。そこで、システムの設計を変更し、ゲートウェイとなっていたスマートフォンを ant plus ドングルの使える PC(Ubuntu OS) に変更した。また、mongoDB と node.js/express にて単純な CRUD を行うサーバーも用意した。丁度良さ指数はクライアントサイドで行い、サーバーサイドは最低限の計算処理能力のみを持たせた。

より様々なクライアントで利用できるよう、マルチプラットフォーム対応が簡単な Web アプリケーションとして実装した。実装には javascript/Nuxt.js にて実装し、グラフ処理や丁度良さ指数の計算などを行う。

“JIT” index はより高速な微分・絶対値・総和のアルゴリズムで構成し、計算速度の向上を計った。

コンテンツサーバーは単に nuxt で作成した Web ページを配信するのみとしている。インフラは docker docker-compose で構築した。

第 5 章
評価 および 考察

5.1. ウェアラブルデバイスを用いたステータス検出

5.1.1 目的

ウェアラブルウォッチを用い、脈拍数の測定を行う。リアルタイムにサーバーと通信し、行動と行動の間に存在する一段落を検知し、“JIT” に合わせて通知を送る。本研究で推定した “JIT” index と、返信された割合、および既読した割合の相関をとり、ディスコミュニケーションが軽減されることを評価する。

5.1.2 手法

ウェアラブル端末を用い、脈拍数を検知しこれを HR とする。事前調査として、下記の傾向が現れることがわかっている。

- 脈拍数が安定している時より不安定な時に返信できる割合が高い
- 脈拍数が低い時はリラックスしている、または休憩中であり、返信をする割合が低い。
- 脈拍数が極端に高い時は運動中などで、返信できない割合が多い。
- 脈拍数がある程度高く、脈拍数が乱れている時は返信できない
- 脈拍数が下がる時かつ、心拍数が低くなりすぎない場合、一段落している場合が多いため返信する割合が多い。

データの収集は図 5.2 のようなシステムを構築し利用し、脈拍数の安定性と脈拍数の高さを用いた丁度良さ指数を図 5.1 のように推定する。

リアルタイムにトラッキングを行えるよう、スマートウォッチを Garmin Forthlete 235J に変更した。それに伴いサンプリング周波数が 0.2[Hz] から 2[Hz] に上昇し、フィルタの周波数も変更した。

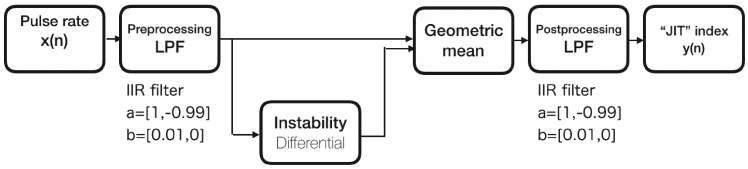


図 5.1　Algorithm of estimating "JIT" index from heart pulse

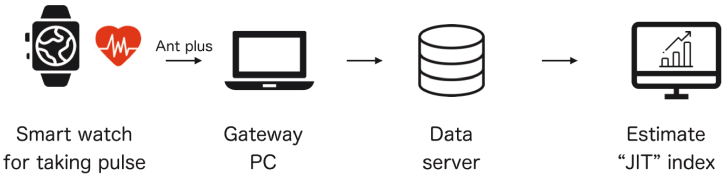


図 5.2　System diagram of collecting heart pulse using a wearable watch[1]

5.1.3　手順

スマートウォッチ、heart rate sensor、スマートフォン、返信できるタイミングを記録するボタンを用いて、時間を記録する。
被験者はスマートウォッチを装着し、普段通り生活をお願いし、脈拍データを数秒～数十秒に 1 度記録する。

1　Icon made by Freepik from www.flaticon.com
Icon made by Smashicons from www.flaticon.com

返信できるタイミングにスマートフォン上で動作するボタンを押してもらい、メッセージ通知が読まれるタイミングやメールメッセージに返信しやすいタイミングの教師データとして使用する。また、返信できると推定されるタイミングでコミュニケーションを試みて、コミュニケーションが成功するかどうかを確認し、これらを比較する。
実験場所：慶應義塾大学大学院および自宅、生活範囲全域
期間：2019 年 11 月から 2020 年 2 月
また、実験で用いるメッセージングシステムは slack を用いてメッセージを送り、適当なタイミングに通知を送る。slack のワークスペースは本実験のために新しく用意した。通知は “返信をください” という単純なメッセージを送信し、slack の標準的な通知を用いた。

5.1.4 結果

“JIT” index を推定する実験を 6 名に行なった。スマートウォッチを用い、“JIT” index を計算することができることがわかった。また、人に話しかけられ一度立ち上がって、着席すると言う状態遷移では、図 5.3 に示すように着席し、一段落ついたタイミングで丁度良さ指数が上昇していることがわかった。メッセージを送りコミュニケーションが成功し、返信が来たタイミングを赤の破線で、図 5.4 に示す。スマートウォッチから取れる脈拍データなどから、メッセージ通知が読まれるタイミングやメールや LINE メッセージに返信しやすいタイミングを数値化することができるようになる。

“JIT” index とメッセージの返信率

6 人に行った。被験者それぞれのデータを図 5.5 5.6 5.7 5.8 5.9 5.10 に示す。

中でも特徴が顕著に現れたデータを拡大して表 5.1.4 図 5.11 5.12 に示す。“JIT” index が低い時に送信された通知は “JIT” index が高くなっているときに返信をしていることが確認できる。ゲートウェイ PC から離れ、トラッキングが外れたタイミングでは返信をしなかった。

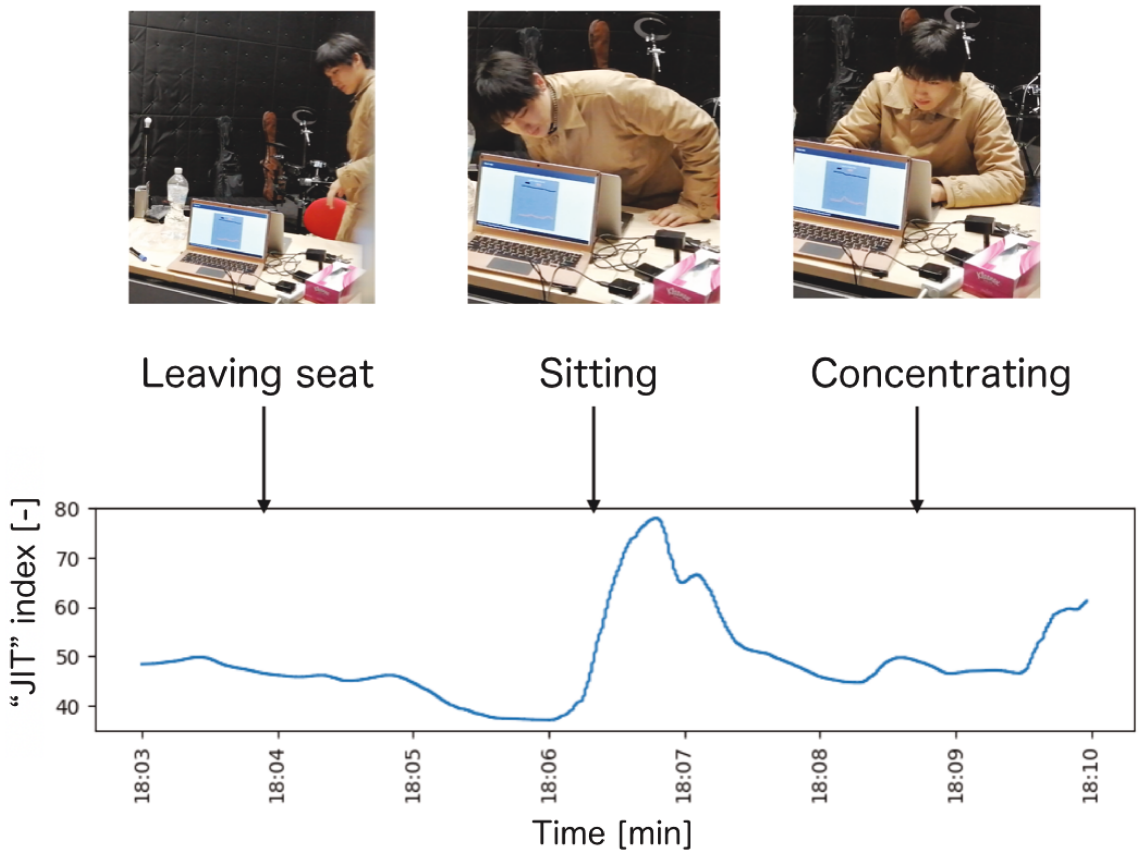


図 5.3 Relationship between activity and “JIT” index

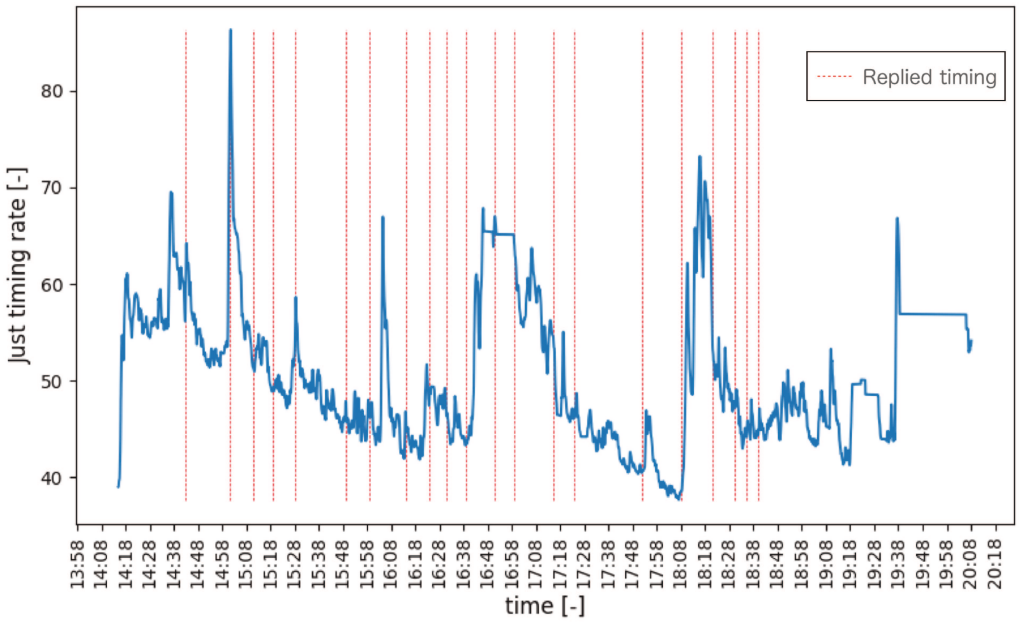


図 5.4 Relationship between reply timing and justtimingrate

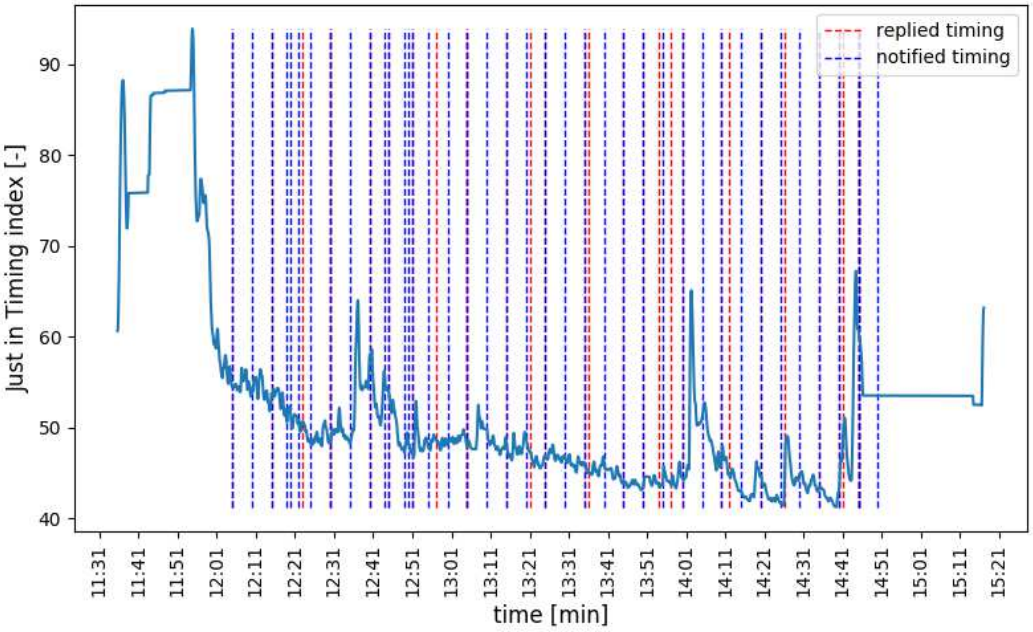


図 5.5 “JIT” index of participant A

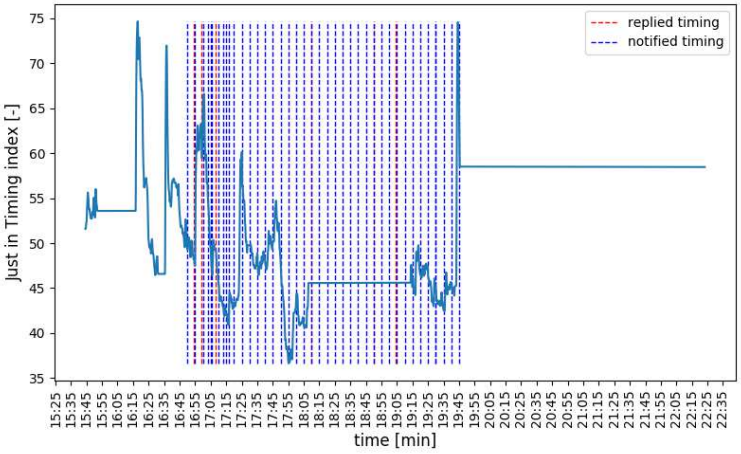


図 5.6 “JIT” index of participant B

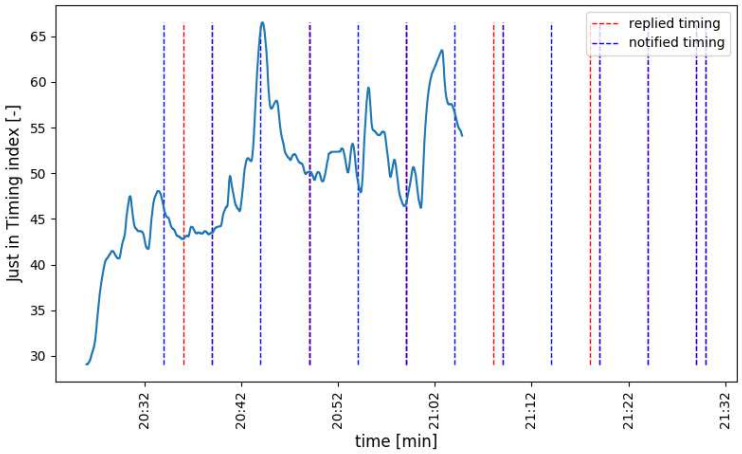


図 5.7 “JIT” index of participant C

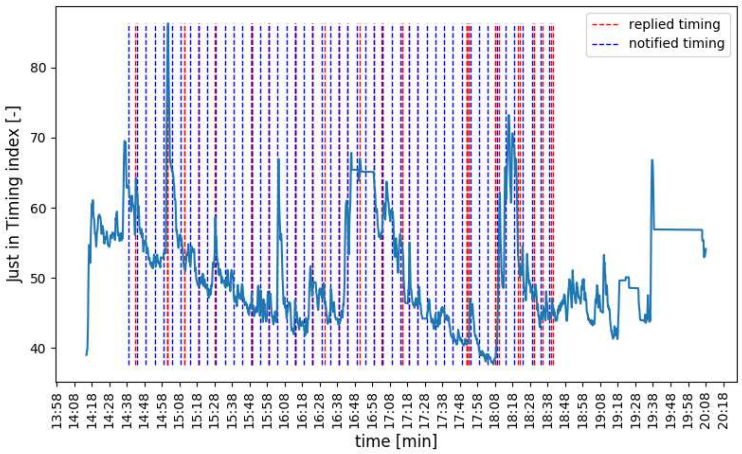


図 5.8 “JIT” index of participant D

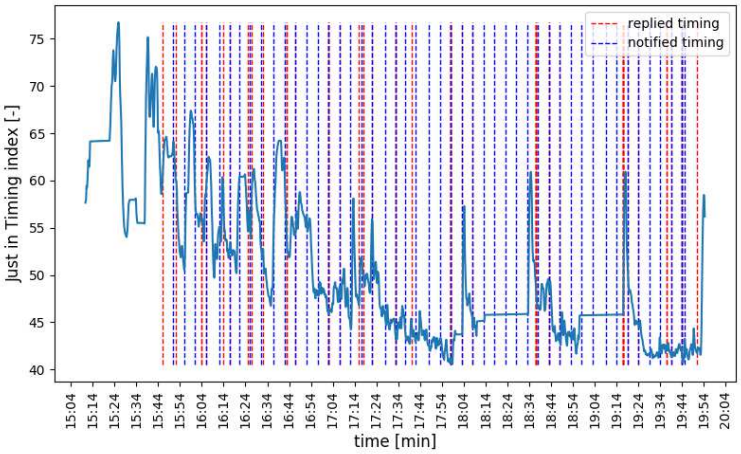


図 5.9 “JIT” index of participant E

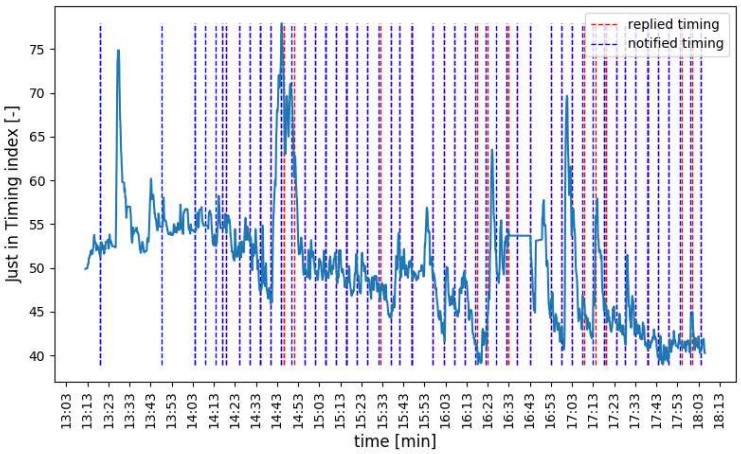


図 5.10 “JIT” index of participant F

表 5.1 Relationship between “JIT” index and time to replying

Notify timing	Respond timing	Time to respond[min]	“JIT” index at notify	“JIT” index at reply
14:39	14:43	4	53.5345012425	74.644531641
14:44	15:01	17	53.90780292582	87.5149941702
14:49	15:01	12	48.6736620647	87.5149941702
14:54	15:01	7	51.6823202499	87.5149941702
14:59	15:01	2	52.5911239901	87.5149941702

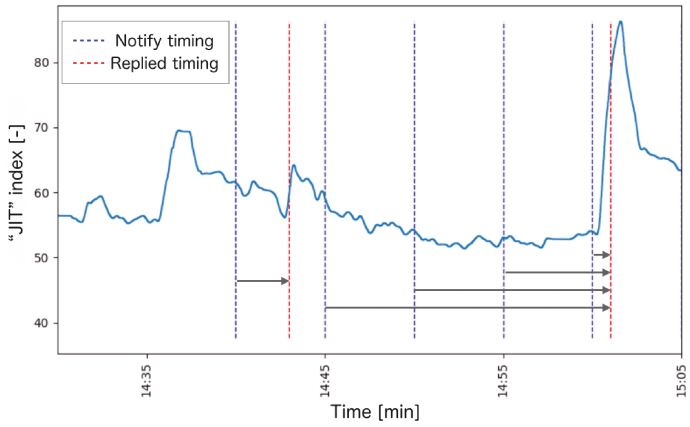


図 5.11 Relationship between “JIT” index and time to replying

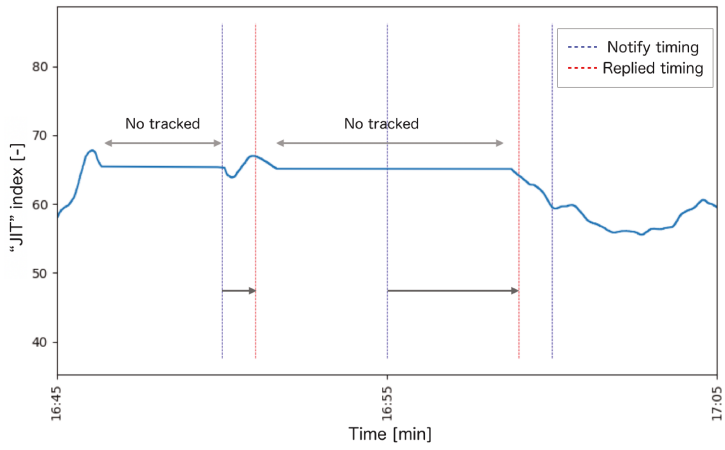


図 5.12 “JIT” index while heart pulse is not tracked

5.2. 考察

5.2.1 “JIT” index に関する評価考察

“JIT” index の立ち上がりエッジに関する評価

コミュニケーションに成功したタイミング (相手から返信があったタイミング) では “JIT” index が高く、true Positive が多いことが確認された。直前の “JIT” index より高くなっている部分（立ち上がりエッジ）で返信されることが多く、コミュニケーションに成功していることが分かった。“JIT” index を観測すればディスコミュニケーションの軽減が成立することがわかった。

また、脈拍のトラッキングが外れ、トラッキングが再開されたタイミングでも、返信されていた。これは被験者がゲートウェイドングルから離れてしまいトラッキングができなくなったからである。つまり、移動などのトランジションの終わりのエッジで返信されることができたと考えられる。返信が返ってこないタイミングでは相手の状態を知ることができなかったため、true negative false negative false positive に関しては、判別できないままとなってしまった。

“JIT” indexの立ち上がりエッジではディスコミュニケーションが軽減されることを確認した。“JIT” indexの高いところではすでにメッセージは返信しメッセージは溜まっていないため、“JIT” indexの高いところがディスコミュニケーションの発生の割合が低いかはわからない。

緊張時や高心拍時の “JIT” index に関する評価

離席、着席、集中までの流れがあり、離席 (休憩・移動時) から集中 (PC 操作) するまでの状態遷移時に “JIT” indexが高くなっていることを確認できた。また、楽器の演奏時や、緊張時、ランニング時など、場面変化が多い時は、ディスコミュニケーションが発生する割合にかかわらず、“JIT” indexが高くなっていることがわかった。心拍の変化や心拍数が高すぎるタイミングを検出し、“JIT” indexを低くする機能の検討が必要がある。

5.2.2 状態変化の検出

5.1 章より、5.3 に示すように、離席、着席、集中までの流れがあり、離席 (休憩・移動時) から集中 (PC 操作) するまでの状態遷移時に “JIT” indexが高くなっていることを確認できた。つまり、状態変化中に “JIT” indexが高く評価され、状態変化の遷移を検出できるのではないかと考えた。。今回の実験では、ウェアラブルウォッチを用い、脈拍の測定をリアルタイムに行い、行動と行動の間に存在する一段落 (状態変化) を検知し、タイミングに合わせて “JIT” で通知を送ることでメッセージ送信から返信までにかかる時間が短く通知に気がつき、返信までの時間を縮めることができると考える 242。

5.2.3 “JIT” と “JIT” index の関数評価・考察

脈拍数情報のみから、ディスコミュニケーションを軽減するための “JIT” indexを検出が可能である。特に、脈拍数のばらつき具合が身体の動きや心の状態と密

接に関わり、立ち上がる・座るなどのタイミング・その前後のタイミングでばらつき具合が大きくなることがわかった。

その検出アルゴリズムとして、安定性を示す移動偏差（一定時間の標準偏差）アルゴリズムおよび、微分した値の絶対値の総和を使ったアルゴリズムの２つを用いた。移動偏差アルゴリズムはある程度長い期間 (1 日程度) のまとまったデータを解析することに向いているが、リアルタイム性に欠いており、処理速度も遅い。微分絶対値の総和を使ったアルゴリズムでは心拍数のエッジをとることが得意であり、リアルタイムに “JIT” index を検出することに向いていた。エッジも急峻で、処理速度も早く、様々なプラットフォームで動かすことが可能だ。

5.2.4 “JIT” と “JIT” index の一般性

本論文では被験者 5 名に対して、“JIT” の評価を行ったが、一般化に対して実験の被験者数が少なく、年齢層や性別に偏りがある。
本論文のターゲットは、メッセージが極端に多く、メッセージングに問題意識を抱え、通知に対するアウェアネスの改善が必要な人に対して行うべきではあるが、実験の協力者として確保するのが難しかったため、メッセージの受信が 10-100 通程度の人にターゲットを変更した。今後はメッセージが特にたくさんくるような人にターゲットを変更して通知を行う必要があるだろう。

5.2.5 “JIT” index が行動に与える影響

現在のメッセージングシステムの多くは送信して相手の返信を待つ受動的な行動が多い。相手の “JIT” index を知ることができれば、能動的にコミュニケションを起こすことを促せるだろう。考えられる活用方法として、“JIT” index を用いた新しいチャットシステムを作れば、すぐに会話を始められる SNS チャットや会話しやすいタイミングを狙って送信するマーケティングなどへの応用が期待される。

5.2.6 “JIT” index のセキュリティについて

心拍数の生データから、集中しているか・話しかけても良いタイミングかなどがわかるため、現在の仕事を邪魔しない円滑なコミュニケーションや、返信をもらいやすいコミュニケーションへの活用が可能だ。

一方で、集中状態は周りが見えない度合いと関係があり、そこを狙った新しい犯罪が発生する危険性を孕んでいる。例えば、集中している人から、何かしら盗むという犯罪や、ある会社のセキュリティエンジニアの忙しさを悪用し、攻撃を仕掛けるような犯罪が考えられる。これにより、初動対応が遅れるなどセキュリティインシデントの発生を増加させる可能性がある。

本論文で論じた “JIT” index を悪用した攻撃やを防ぐために、公開範囲などに関して注意深く取り扱うべきだろう。

5.2.7 自覚した “JIT” 記録に関する考察

メッセージの返信は受動的に行うものであるのに対して、能動的にアクションをとり、返信ができる・できないタイミングできないとしてラベル付けを行った。被験者が自覚した “JIT” の能動的に記録したものであり、実際の “JIT” は受動的なアクションによって決まるため、“JIT” のラベル付けとして不適切であると考えられる。受動的なアクションを求められるディスコミュニケーションの軽減とは本質的な違いがあった。

第 6 章
結　　論

6.1. まとめ

丁度良さ指数 “JIT” index を脈拍数から推定し、ディスコミュニケーションの発生割合、コミュニケーションの成功までに要した時間との関係性について評価をおこなった。コミュニケーションの試みから丁度良さ指数の立ち上がりエッジに返信をもらうことができ、さらにメッセージを送ると 1 分以内に返信される傾向があると言うことが分かった。丁度良さ指数のタイミングでコミュニケーションを試みた時に、返信率が高く、返信までにかかる時間が短くなることが確認された。

スマートホンや PC の一方でのみ利用されるチャンネルは返信に要する時間や状況が変化するために、チャンネルごとの特性が出てきてしまうために、コミュニケーションチャンネルなどによって丁度良いタイミング “JIT” index は変化することが確認された。

6.1.1 丁度良いタイミングと脈拍数の関係

丁度良いタイミングは以下のような脈拍数などと以下のような関係があることが分かった。

- 脈拍数が安定しているより不安定な時に返信できる割合が高い
- 脈拍が低い時はリラックスしている、または休憩中であり、返信をする割合が低い

- 心拍数が極端に高い時は運動中などで、返信できない割合が多い
- 心拍数がある程度高く、脈拍数が乱れている時は返信できる割合が多い

6.1.2 ウェアラブルウォッチによる状態遷移推定

ディスコミュニケーション発生要因のうち、読まれない・返信されないことに対し、状態の遷移タイミングを推定するというアプローチで、通知を適切なタイミング “JIT” および、その指数 “JIT” index を定義した。返信にかけた時間指数には関係があることが確認できた。

6.2. 今後の展望

6.2.1 ディスコミュニケーション軽減への活用検討

本論文では返信に対して適切なタイミングの調査と実装、評価を行なった。返信に対する適切なタイミングの指数 “JIT” index を使うことで、返信してもらえる通知を発行することによってディスコミュニケーションの発生割合の改善が可能と考える。今後、様々なコミュニケーションチャンネルに導入し、チャンネルごとにどのような変化が現れるか観察をする。

6.2.2 ウェアラブルウォッチの普及による新たな活用方法

ウェアラブルウォッチの普及が目覚しく、心拍センサの性能やリアルタイム性上がり、様々なセンサが搭載されている。今後は普及とともに、一人 1 台のウェアラブル端末を装着する時代が来るだろう。複合的なデータを使い複雑な機能を持たせることでさらに詳細に、“JIT” index を改善できる。

そこで、本研究で提案した “JIT” index を用いて通知のタイミングを調整することで、ディスコミュニケーションの軽減だけでなく広い分野で活用されることが期待される。現在はメールで行われている EC サイトの広報・マーケティングなどに向けた “JIT” index にも活用することで、売り上げの向上や PV 数の増加などができるのでは無いかと考えている。

6.2.3 新しいコミュニケーションの形式と新しい働き方

相手のステータスが事前にわかることによって、コミュニケーションの通知に対して、向けるための意識が少なくなりアウェアネスの上昇が期待される。アウェアネスが改善され、より長く集中でき、効率の高い業務へと活用が期待される。

謝　　辞

本研究の指導教員であり、幅広い知見から的確な指導と暖かい励ましやご指摘をしていただきました慶應義塾大学大学院メディアデザイン研究科の加藤　朗教授に心から感謝いたします。

研究の方向性について様々な助言や指導をいただきました慶應義塾大学大学院メディアデザイン研究科の前川　マルコス　貞夫 専任講師に心から感謝いたします。

研究指導や論文執筆など数多くの助言を賜りました慶應義塾大学大学院メディアデザイン研究科のムハマド ヤメン・サライジ 特任講師に心から感謝いたします。

数多くの助言を賜りました慶應義塾大学大学院メディアデザイン研究科の砂原秀樹 教授に心から感謝いたします。

研究発表やポスター発表などの機会を与えてくださり、数多くの助言をいただきましたWIDE研究会の皆様に心から感謝いたします。

慶応義塾大学大学院メディアデザイン研究科の先輩、後輩、同期の皆様のご助力いただき、論文執筆を達成することが出来ました。2年間という短い期間でしたが、この時間を忘れることはありません。本当にありがとうございました。

参 考 文 献

[1] 貴紘田中. オフィスワーカーの状況推定 : 割込み拒否度を中心に. 電子情報通信学会誌 = The journal of the Institute of Electronics, Information and Communication Engineers, Vol. 95, No. 5, pp. 457–460, may 2012. URL: https://ci.nii.ac.jp/naid/110009445381/.

[2] What is ant+ ? - cake athletic. https://www.cakeathletic.com/what-is-ant-plus/. (Accessed on 12/18/2019).

[3] 菅原龍光, 片上大輔. コミュニケーションをしないコミュニケーションロボットの開発. 日本知能情報ファジィ学会 ファジィ システム シンポジウム 講演論文集, Vol. 29, pp. 118–118, 2013. doi:10.14864/fss.29.0_118.

[4] 欣也藤田. オフィスにおける集中とコミュニケーションの両立 (特集オフィスと働き方の未来). ヒューマンインタフェース学会誌 = Human interface = Journal of Human Interface Society, Vol. 20, No. 2, pp. 120–124, 2018. URL: https://ci.nii.ac.jp/naid/40021837174/.

[5] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K. Dey, and Hideyuki Tokuda. Reducing users' perceived mental effort due to interruptive notifications in multi-device mobile environments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, p. 475–486, New York, NY, USA, 2015. Association for Computing Machinery. URL: https://doi.org/10.1145/2750858.2807517, doi:10.1145/2750858.2807517.

[6] 貴裕岩田, 哲生山邉, 達夫中島. マルチタスク環境下における認知負荷の測定

と評価. Technical Report 8, 早稲田大学, 早稲田大学, 早稲田大学, may 2009.

[7] 匡大越, 大幹野崎, ラモスフリアン, 仁中澤, デイアニンド, 英幸徳田. ユーザの認知負荷を軽減する情報提供タイミングの検知. 情報処理学会論文誌, Vol. 56, No. 10, pp. 1944–1958, oct 2015.

[8] 古川順光. 姿勢保持・変換時における心拍数変化の分析. 理学療法学 Supplement, Vol. 2012, pp. 48102024–48102024, 2013. doi:10.14900/cjpt.2012.0.48102024.0.

[9] 将宏佐竹, 日出樹籾山, 佐知子上村. 健常者の能動的な体位変換に伴う血圧と脈拍数の変化. 秋田大学医療技術短期大学部紀要, Vol. 6, No. 2, pp. 169–174, jan 1998.

[10] 福嶋政期. 情動インタフェースのエンタテインメントとコミュニケーションへの応用. 日本ロボット学会誌, Vol. 32, No. 8, pp. 692–695, 2014. doi:10.7210/jrsj.32.692.

[11] 富夫渡辺, 雅史大久保. コミュニケーションにおける引き込み現象の生理的側面からの分析評価. 情報処理学会論文誌, Vol. 39, No. 5, pp. 1225–1231, may 1998.

[12] STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES. RFC 822, August 1982. URL: https://rfc-editor.org/rfc/rfc822.txt, doi:10.17487/RFC0822.

[13] Simple Mail Transfer Protocol. RFC 821, August 1982. URL: https://rfc-editor.org/rfc/rfc821.txt, doi:10.17487/RFC0821.

[14] Dr. Marshall T. Rose and John G. Myers. Post Office Protocol - Version 3. RFC 1939, May 1996. URL: https://rfc-editor.org/rfc/rfc1939.txt, doi:10.17487/RFC1939.

[15] Mark Crispin. INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501, March 2003. URL: https://rfc-editor.org/rfc/rfc3501.txt, doi:10.17487/RFC3501.

[16] Dr. Masataka Ohta and Ken'ichi Handa. ISO-2022-JP-2: Multilingual Extension of ISO-2022-JP. RFC 1554, December 1993. URL: https://rfc-editor.org/rfc/rfc1554.txt, doi:10.17487/RFC1554.

[17] Dr. John C. Klensin and Randall Gellens. Message Submission for Mail. RFC 6409, November 2011. URL: https://rfc-editor.org/rfc/rfc6409.txt, doi:10.17487/RFC6409.

[18] 「messenger」を app store で. https://apps.apple.com/jp/app/messenger/id454638411. (Accessed on 12/18/2019).

[19] コミュニケーションアプリ line（ライン）. https://line.me/ja/. (Accessed on 12/18/2019).

[20] 「wechat」を app store で. https://apps.apple.com/jp/app/wechat/id414478124. (Accessed on 12/18/2019).

[21] Slack でメールを受信する — slack. https://slack.com/intl/ja-jp/help/articles/206819278-Slack-%E3%81%A7%E3%83%A1%E3%83%BC%E3%83%AB%E3%82%92%E5%8F%97%E4%BF%A1%E3%81%99%E3%82%8B. (Accessed on 12/18/2019).

[22] Microsoft teams でインスタント メッセージングをオンラインで. https://products.office.com/ja-jp/microsoft-teams/instant-messaging. (Accessed on 12/18/2019).

[23] 裕泉, 哲太郎上原, 義敏國枝. 柔軟な管理情報の共有によるトラブルチケットシステムの構築. Technical Report 113(2000-DSM-020), 和歌山大学システム情報学センター, 和歌山大学システム工学部, 和歌山大学システム工学部, dec 2000.

[24] Fitbit charge 3 — 進化した健康づくり & フィットネス用トラッカー. https://www.fitbit.com/jp/charge3. (Accessed on 12/18/2019).

付　　録

A. 丁度良さを求めるpythonスクリプト

ソースコード 1 “JIT” を検知するスクリプト

```
1  import csv
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from matplotlib import dates as mdates
5  from datetime import datetime as dt
6  import math
7
8
9  def readCsv(fileName):
10     with open(fileName) as f:
11         reader = csv.reader(f)
12         header = next(reader)
13         return [[dt.strptime(row[0], '%H:%M:%S'), float(row[1])]
               for row in reader]
14
15
16 def readTiming(goodTimingFileName, badTimingFileName):
17     f = open(goodTimingFileName)
18     reader = csv.reader(f)
19     header = next(reader)
20     goodTiming = [[dt.strptime(row[0], '%H:%M:%S'), dt.strptime(
           row[1], '%H:%M:%S')] for row in reader]
21     f.close()
22
23     f = open(badTimingFileName)
24     reader = csv.reader(f)
25     header = next(reader)
```

```
26      badTiming = [[dt.strptime(row[0], '%H:%M:%S'), dt.strptime(
            row[1], '%H:%M:%S')] for row in reader]
27      f.close()
28      return (goodTiming, badTiming)
29
30
31  def drawData(label, data):
32      fig = plt.figure(figsize=(15.0, 10.0))
33
34      # 図の中にサブプロットを追加する
35      subPlots = []
36      for d in range(len(data)):
37          subPlot = fig.add_subplot(len(data), 1, d + 1)
38          subPlot.xaxis.set_major_locator(mdates.HourLocator())
39          subPlot.xaxis.set_major_formatter(mdates.DateFormatter("%
                H:%M:%S"))
40          subPlots.append(subPlot)
41
42      for i, subPlot in enumerate(subPlots):
43          fig1_a_1, = subPlot.plot(label, data[i])
44
45      # ラベルを縦向きに
46      for ax in fig.axes:
47          plt.sca(ax)
48          plt.xticks(rotation=90)
49
50
51  def drawHeartRate(label, heartRate, goodTiming):
52      fig = plt.figure(figsize=(15.0, 10.0))
53
54      # 図の中にサブプロットを追加する
55
56      subPlotHeartRate = fig.add_subplot(2, 1, 1)
57      subPlotHeartRate.xaxis.set_major_locator(mdates.HourLocator
            ())
58      subPlotHeartRate.xaxis.set_major_formatter(mdates.
            DateFormatter("%H:%M:%S"))
59      subPlotHeartRate_ax, = subPlotHeartRate.plot(label, heartRate
            )
60
61      subPlotHeartRate.set_xlabel("time [-]", fontsize=20)
```

```
62      subPlotHeartRate.set_ylabel("Heart pulse [bpm]", fontsize=20)
63
64      subPlotJustTiming = fig.add_subplot(2, 1, 2)
65      subPlotJustTiming.xaxis.set_major_locator(mdates.HourLocator
            ())
66      subPlotJustTiming.xaxis.set_major_formatter(mdates.
            DateFormatter("%H:%M:%S"))
67      subPlotJustTiming_ax, = subPlotJustTiming.plot(label,
            goodTiming)
68
69      subPlotJustTiming.set_xlabel("time [-]", fontsize=20)
70      subPlotJustTiming.set_ylabel("Just timing rate [-]", fontsize
            =20)
71
72      # ラベルを縦向きに
73      for ax in fig.axes:
74          plt.sca(ax)
75          plt.xticks(rotation=90)
76
77
78  def calcFulterCutoffFreq(Samplefreqency, filterCoefficient):
79      r = 1 - filterCoefficient
80      fc = Samplefreqency / (2 * math.pi) * math.acos((2 - (2 * r)
            - (r ** 2)) / (2 * (1 - r)))
81      return fc
82
83
84  def drawHistgram(bad, good):
85      width = 0.3
86      x_axis = np.arange(100)
87      fig = plt.figure(figsize=(15.0, 10.0))
88
89      subPlot = fig.add_subplot(1, 1, 1)
90      subPlot.set_xlabel("Just Timing Rate [-]", fontsize=30)
91      subPlot.set_ylabel("count [-]", fontsize=30)
92      subPlot.legend(fontsize=18)
93      # fig.xlim(40, 80)
94
95      subPlot.bar(x_axis, bad, width=width, align='center', label='
            label of bad timing')
```

```
96      subPlot.bar(x_axis + width, good, width=width, align='center
            ', label='label of good timing')
97
98      # ラベルを縦向きに
99      for ax in fig.axes:
100         ax.set_xlim(40, 80)
101         plt.legend(loc='upper right', fontsize=30)
102         plt.sca(ax)
103         plt.xticks(rotation=90)
104
105
106 def deviation(data):
107     mean = np.mean(data)
108     std = np.std(data)
109     return [((d - mean) / std * 10) + 50 for d in data]
110
111
112 def filter(data, coefficient):
113     filteredData = []
114     for HR in data:
115         filteredData.append(
116             HR if len(filteredData) == 0
117             else float(filteredData[-1]) * coefficient
118                  + float(HR) * (1.0 - coefficient)
119         )
120     return filteredData
121
122
123 def stddev(data, n):
124     stdData = []
125     for i, HR in enumerate(data):
126         stdData.append(np.std(data[0 if i < n else (i - n):i +
                1]))
127     return stdData
128
129
130 def combine(array1, array2):
131     return [x * y for (x, y) in zip(array1, array2)]
132
133
```

```
134 def calcJustTiming(data, filter1Coefficient, filter1Order,
        stdCount, filter2Coefficient, filter2Order):
135     # digital filter
136     for i in range(filter1Order):
137         data = filter(data, filter1Coefficient)
138 
139     deviationdHeartRate = deviation(data)
140 
141     stddevHeartRate = stddev(data, stdCount)
142 
143     deviationstddevHeartRate = deviation(stddevHeartRate)
144 
145     data = np.sqrt(combine(deviationdHeartRate,
            deviationstddevHeartRate))
146 
147     for i in range(filter2Order):
148         data = filter(data, filter2Coefficient)
149 
150     return data
151 
152 
153 def compareTiming(timing, Label):
154     for label in Label:
155         if (label[0] < timing <= label[1]):
156             return True
157     return False
158 
159 
160 def calcHistgram(time, justTiming, goodTiming, badTiming):
161     goodTimngList = [0] * 100
162     badTimngList = [0] * 100
163     for i, timing in enumerate(justTiming):
164         if compareTiming(time[i], goodTiming):
165             goodTimngList[int(timing) if 0 <= timing < 100 else
                    99] += 1
166         if compareTiming(time[i], badTiming):
167             badTimngList[int(timing) if 0 <= timing < 100 else
                    99] += 1
168     return [badTimngList, goodTimngList]
169 
170 
```

```
171 def main():
172     # sample rate
173     Fs = 2
174 
175     # CSV からデータを読み込む
176     data = readCsv("rawData20191014.csv")
177     goodTiming, badTiming = readTiming("goodtiming.csv", "
            badtiming.csv")
178 
179     # 転地してtime と heartrate に分割
180     (time, heartRate) = np.array(data).T
181 
182     filter1Coefficient = 0.99
183     filter1Order = 1
184     stdCount = 60
185     filter2Coefficient = 0.9
186     filter2Order = 1
187 
188     Fc1 = calcFulterCutoffFreq(Fs, filter1Coefficient)
189     Fc2 = calcFulterCutoffFreq(Fs, filter2Coefficient)
190 
191     print("filter 1 cutoff freqency is {filter1Cutoff}[Hz]\
            nfilter 2 cutoff freqency is {filter2Cutoff}[Hz]"
192           .format(filter1Cutoff=Fc1, filter2Cutoff=Fc2))
193 
194     justTiming = calcJustTiming(heartRate, filter1Coefficient,
            filter1Order, stdCount, filter2Coefficient, filter2Order)
195 
196     badhist, goodhist = calcHistgram(time, justTiming, goodTiming
            , badTiming)
197 
198     drawHeartRate(time, heartRate, justTiming)
199     drawHistgram(badhist, goodhist)
200 
201     plt.show()
202 
203 
204 if __name__ == '__main__':
205     main()
```

ソースコード 2 心拍数を解析するためのスクリプト

```
1  import csv
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from matplotlib import dates as mdates
5  from datetime import datetime as dt
6
7  def readCsv(fileName):
8      with open(fileName) as f:
9          reader = csv.reader(f)
10         header = next(reader)
11         return [[dt.strptime(row[0], '%H:%M:%S'), float(row[1])]
               for row in reader]
12
13 def drawHeartRate(label, heartRate):
14     fig = plt.figure(figsize=(15.0, 10.0))
15
16     # 図の中にサブプロットを追加する
17
18     subPlotHeartRate = fig.add_subplot(1, 1, 1)
19     subPlotHeartRate.xaxis.set_major_locator(mdates.HourLocator
           ())
20     subPlotHeartRate.xaxis.set_major_formatter(mdates.
           DateFormatter("%H:%M:%S"))
21     subPlotHeartRate_ax, = subPlotHeartRate.plot(label, heartRate
           )
22
23     subPlotHeartRate.set_xlabel("time [-]", fontsize=20)
24     subPlotHeartRate.set_ylabel("Heart pulse [bpm]", fontsize=20)
25
26     # ラベルを縦向きに
27     for ax in fig.axes:
28         plt.sca(ax)
29         plt.xticks(rotation=90)
30
31 def main():
32
33     # CSV からデータを読み込む
34     data = readCsv("rawData20191014.csv")
35
36     # 転地してtime と heartrate に分割
```

```
37      (time, heartRate) = np.array(data).T
38
39      drawHeartRate(time, heartRate)
40
41      plt.show()
42
43
44  if __name__ == '__main__':
45      main()
```