江端さんのDIY奮闘記 介護地獄に安らぎを与える“自力救済的IT”の作り方（最終回）：

走れ!ラズパイ 〜 迷走する自動車からあなたの親を救い出せ

https://eetimes.jp/ee/articles/2003/25/news029.html

認知症患者が、自動車を運転したままさまよい続ける、という問題は社会で大きくクローズアップされています。私も、父が存命だったころ、同じような出来事で人生最大級の恐怖を味わったことがあります。そこで今回、「ラズパイ」を自動車に積み込み、迷走する自動車から運転者（認知症患者）を救い出すためのシステムを構築してみました。

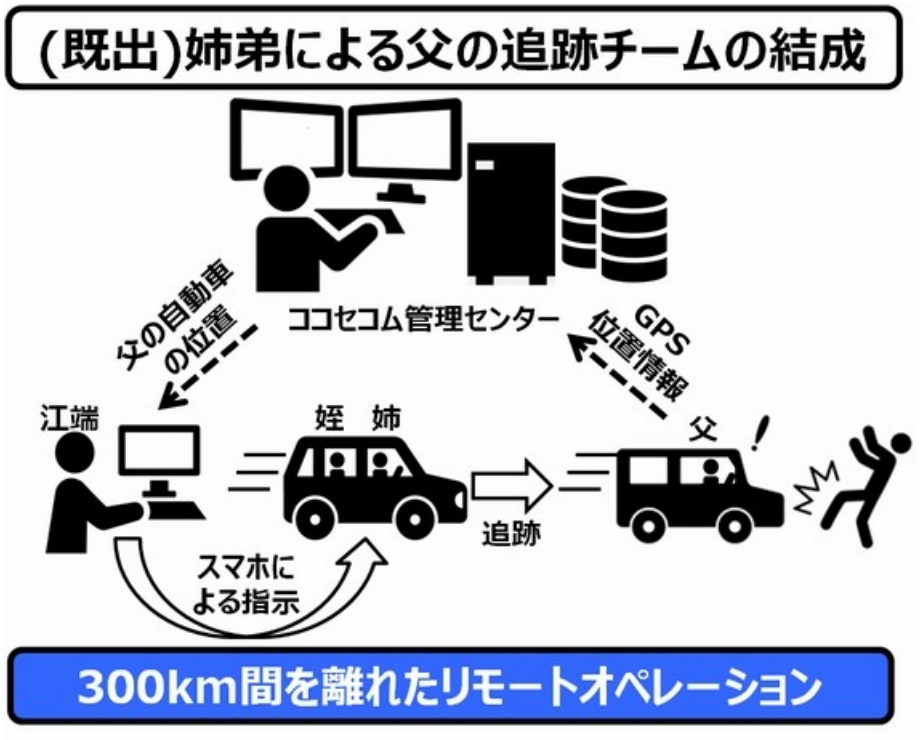
2020年03月30日 10時30分 更新

[江端智一, EE Times Japan]

人材不足が最も深刻な分野の一つでありながら、効率化に役立つ（はずの）IT化が最も進まない介護の世界。私の実体験をベースに、介護ITの“闇”に迫りつつ、その中から一筋の光明ともなり得る、“安らぎ”を得るための手段について考えたいと思います。⇒連載バックナンバーはこちらから。

「私にできること」は、もうないのか？

本シリーズの第1回で、自動車を運転したまま自宅に帰れなくなった父を遠隔から救出するための、姉との共同作戦およびその全概要をお伝えました。



その発生から身柄の確保に至る全記録を、一一 特に、私の人生最大級の恐怖（リアルタイムの高速道路への誤進入の確認）と、父が体験したであろう人生最大級の恐怖（発生から6時間以上の迷走のプロセス）を 一一 できるかぎり詳細に記載したのです。

そんな父も、今はもういません（2018年7月28日他界）。もう、私にできることは、何もないのです。

―― 私にできることは何もない？　本当に？

私が第1回で記載したような事件（自動車に乗ったまま広域で行方不明になる）は現在、リアルな社会問題として、クローズアップされています。本人に自覚がなかったり、報告されていなかったりするだけで、同様の事件は相当数あり、捜索に要したコストも一回当たり10万円を要した、という報告もあります（参考）。

無論、このような事件が事故に発展してしまえば、このコストは2桁以上跳ね上がりますが、さらに死傷者が出てしまうと、もはや金額に換えることすらできません。多くの人が、行政や立法による解決（免許有効期限の定年制の導入など）を望んでいることも知っていますが、これが「恐ろしく難しい」ことは、**既に説明しました**。

これを「（自動車運転による）"徘徊"」と呼んでしまったら「私たちの負け」だ ――と思うのです（ 参考）。

未来（老後）を、自分たちで閉ざしてしまう社会が、私たちの願いであるはずがありません。

私たちの向かう未来は、高齢者を自宅や施設に閉じ込める社会 ―― アニメ『PSYCHO-PASS サイコパス』のように、罪を犯していない者までも「潜在犯」として裁くような社会 ―― を、作ることではないはずです（参考：著者のブログ）。

運悪く行方不明になってしまった人さえも、社会全体で見守り、保護し「**いつでも、どこでも、誰もが、安心して外出できる国**」―― これが、私たちが目指すべき社会の方向であるべきです。

こんにちは、江端智一です。

「介護IT」シリーズの第5回（最終回）です ―― というか、最終回にしなければ、このシリーズが終わりそうにない、という直感が（自分も、恐らくはEE Times Japan編集部にも）あり、今回、「力で捻じ伏せる最終回」となります。

今回の前半は、今回のタイトル「走れ!ラズパイ」の通り、ラズパイを車に搭載しますが、単なるIoT（モノのインターネット）デバイスではなく、新しいコンセプト「移動サーバ（Movable Server）」を提唱します ―― つまり、デバイスを動かすという従来のIoTの考え方の真逆、「**『サーバ本体』を動かす**」の意義と、その具体例を詳細に説明します。

そして、後半では、前回の宿題となっていた、「苦痛に苦しむ私を、誰が殺してくれるのか」というテーゼに対して、現在、私ができることとして、自作の「尊厳死宣誓書――リビング・ウィル」の最終版を公開します。

さらに、本シリーズ第1回で掲げたテーゼ、「介護ITの究極の目的は『苦痛の定量化/見える化』であるとした上で、『「苦痛」が動かす社会の未来像』を明らかにする」についての私なりの検討結果をご報告致します。

ラズパイをクルマに持ち込んで「移動サーバ」にする

それでは、前半を始めます。

最終回では、ラズパイを「(Web等の)サーバ本体」として、自動車本体に搭載してしまいます —— イメージをしては、**「データセンター」の設備を自動車で持ち運ぶようなもの**です(ちょっと大げさですが)。

本連載第1回では、行方不明の父の捜索に「ココセコム」という、安価で使いやすい位置情報提供サービスを活用できたことを紹介しました。

また、IoT関連であれば、現在のスマホには100%、位置情報アプリがインストールされていますし、(私が**自分のサイト**でお世話になっている)さくらインターネットの「sakura.io」や、Amazon Web Service(以下AWSという)のように、開発者のカスタマイズを前提としたIoTプラットフォームを提供しているところもあります。

私が、ラズパイをクルマに持ち込んでやろうとしている「移動サーバ(Movable Server)」は、その真逆の発想です —— 普通に考えれば**ナンセンスの極み**です。

なぜ私が、たった一人でも、このようなやり方に固執しているかというと、汎用のIoTプラットフォームは、個人の週末エンジニアが簡単に試みるような仕組みになっていないからです。

そして、前述した通り、未来の私にとっての「いつでも、どこでも、誰もが、安心して外出できる国」を目指すための、**私の、私による、私が楽しめるアプリケーションが作れない**からです。

それと、当初私は、AWSを含めた介護ITシステムの構築を目指して、AWSの勉強を、それはもう死ぬほど真剣に、相当に膨大な時間(2019年11月から、ほぼ全部の週末)をつぎ込んで、勉強と試作を繰り返してきました(一定の成果*)もありましたが) —— **もうイヤだ**、とはっきり言い切れるレベルで、AWS(正確に言うとクラウドサーバの構築全般)が嫌いになりました。

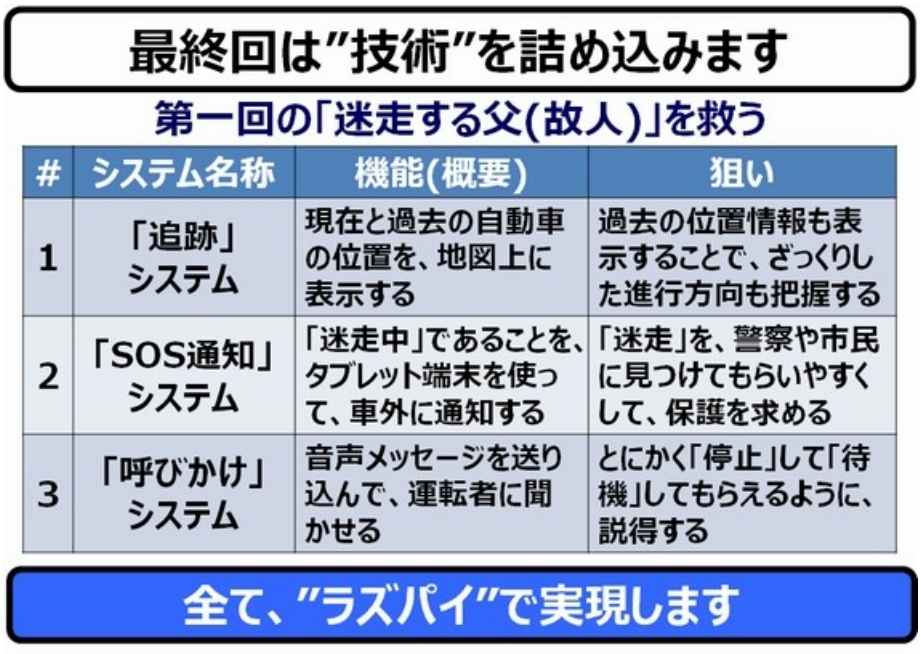
*)成果1、成果2、成果3

私の手に余る技術を、読者に説明できる訳がない —— そう考えた私は、最終回のここに至って、**AWSを介護ITのプラットフォームの構成から追放すること**を決意しました。

（AWSが「悪い技術」といっている訳ではありません。私が何を決意しようとも、もう私たちにはオンプレミスに戻る未来はありえませんし、実際に、私も実証実験（https://mory.mobi）等で使っています）

最終的には「私一人しか使わない介護ITを、どうして公のサーバで構築しなければならない？」と考えましたーー つまり、私の考え方の原点「**世界がどうなろうと知ったことか**」に立ち戻ったということです。

では、本最終回で展開するシステムを3つ纏めて説明します。一応、この記事の読者の皆さんが、ラズパイを使って実際に構築可能な程度には丁寧に（可能な限りコミカルに）説明しますが、退屈と感じた方は、一気に後半までスキップしてください。



以下の図は、江端家の自家用車を想定した、車載装置の配置を示す図と写真です。

このシステムの構築風景（の一部）を、20秒程度の動画で紹介致します。

新規に購入しなければならないデバイズ群

https://eetimes.jp/ee/articles/1912/26/news034_5.htmlの物に加えて、

申し訳ありませんが、以下のもの揃えて下さい

#	対象	機能(概要)	外観	価格
0	全システム	Anker PowerHouse (ポータブル電源 434Wh / 120,600mAh)		5万円
		BESTEK カーインバーター MRI3010BU		3千円
1	「追跡」システム	ＧＰＳ受信機キット　１ＰＰＳ出力付き「みちびき」３機受信対応(秋月通商)		450円
2	「SOS通知」システム	タブレット(WiFi対応)であればなんでも良い(スマホでも良い)		2万円
3	「よびかけ」システム	TPA2006使用超小型Ｄ級アンプキット(秋月通商)		100円
		ダイナミックスピーカ　50mmΦ　８Ω 0.4W		100円

「移動サーバ（Movable Server）」の最大の弱点は「電力」です。なぜなら、今回のラズパイは、自動車が動いていない状態でも、一定時間（少なくとも丸一日くらい）は動作し続けてもらう必要があるからです。『父が車の運転をやめて、車から離れるたびに、移動サーバがダウンする』を避けるため、バッファとなる電源が必要となります。

私はバッファ電源として、市販のポータブル電源を使うことにしました。ポータブル電源への充電は、シガーライターからカーインバーターを経由して、運転中のみ行います（つまり充電しながら放電する、ということです）。ラズパイは電力が低下すれば自動的にシャットダウンプロセスが走ります。また、車を動かし出せば、ポータブル電源に給電されて、ラズパイが自動的に再起動するので、ラズパイにとっても安全な運用ができます。

5万円のポータブル電源を購入するのは、痛い出費と思いますが、探せばもっと安いものもありますので、安心してください（でも、災害時用に持っておくと結構安心です）。ちなみに、車のバッテリーの電源をラズパイに直結するという手段もあります（今回は割愛します）。

ソフトウェアですが、今回はプログラム言語として"Go"を採用しました。インストールの方法は、ネット上に山ほど転がっていますので、お好きな方法で実施しておいてください（私のインスールメモはこちら）。

ちなみに、C/C++に固執していた私に一体何が起きたか、と問われれば、いろいろなことが言えますが、最も衝撃的だったのは**「40万円のゲーミングPCを使って、力づくでブン回していたデモ用のプログラム（python）が、Goに移植されたことで（ただし高度な並列処理を活用）、5000円のラズパイ上でサクッと動いたのを見てしまった時」**です。

そうです。御託（ごたく）はどーでもいいんです。この世界では**「動いているものだけが真実」**ですから。

では、最初に上記の3つのシステム、（1）「追跡」システム、（2）「SOS通知システム」、（3）「よびかけ」システムに共通の環境を記載します。

　まず、ラズパイは、これまでの連載で使ってきたものと同じものを使います。固定IPアドレスが付与されるイプシムのSIMを使い、IPアドレスは**211.158.177.150**とします（もちろん、デタラメなIPアドレス）。これによって、私（江端）は、車に搭載されたラズパイに直接SSHでログインできます（SSHログインは、TeraTerm等を使ってください）

　また、ラズパイに搭載されている無線LANをアクセスポイントに改造します（方法は後述します）。IPアドレスは**192.168.101.1**とします（こっちは本物のIPアドレス）

　ラズパイへのSSHログインのパスワードについては、ご自分で設定したものを使って下さい（デフォルトではpi/raspberryとなることが多いようですので、設定変更しておかないと、簡単にラズパイを乗っ取られます）。

「追跡システム」を作ってみる

　では最初に、（1）の「追跡」システムの作り方を説明します。

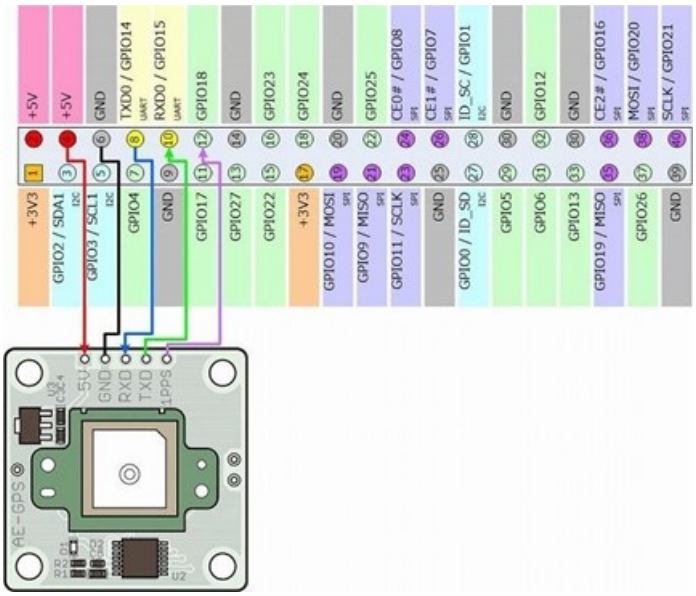
　と、その前に、このシステムが実現するイメージを示します。ブラウザから、このサイトにアクセスして見てください。これは、本シリーズ第1回の父の「高速道路進入」事件を再現してみたものです。

もし、このシステムがあれば、父の高速の道路進入は止められていた ―― それは無理だったと思いますが ―― 過去60分間分くらいのトレースが残れば、未来の進路を、ある程度絞ることくらいのことはできたかもしれません。

それでは、構築方法を説明します。

(1)GPS受信キットの接続

(a)ラズパイと「秋月のGPS受信機キット」を、ジャンパー線を使って、以下のように接続して下さい。



(b)GPS受信キットを稼働させます

```
$ sudo raspi-config
```

「5 Interfacing Options」を選択→「 P6 Serial 」を選択し、「Would you like a login shell to be accessible over serial?」に対してYesを選択、Finishし、リブートします。

```
$ sudo reboot
$ ls /dev/se*
```

で、/dev/serial0 ができている（はず）

(c)/boot/cmdline.txtの修正

既存の設定を#でコメントアウトした後、以下を追加して下さい。

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait quiet splash plymouth.ignore-serial-consoles
```

ここでもう一度リブートします。

```
$ sudo reboot
```

その後、Pythonのシリアルモジュールをインストールします。

```
$ pip install pyserial
```

その後、

```
$ cat /dev/serial0
```

で、b'$GPGSV,4,4,14,23,05,150,19,14,04,054,*74¥r¥n' のようなメッセージが出てくれば、成功しています（結構な頻度で失敗しますので、覚悟してとりかかって下さい。Web等で調べて、自分の環境に合わせて動くようにして下さい）

(2)ディレクトリの作成

どこにどういう名前でもいいですが、ディレクトリを作ります。（ここではmaptest01としました）

```
cd ~; mkdir maptest01; cd maptest01
```

(3)位置情報取得プログラム（python言語）の作成

「秋月のGPS受信機キット」から得られた（ラズパイがある場所（自動車）の位置情報を捕捉し、その情報を、data.csvに書き込むプログラムです。nmea2csv_line.py という名前で保存して下さい。

```
# coding: UTF-8
import time, serial, micropyGPS, csv
gps = micropyGPS.MicropyGPS（9, 'dd'）
def rungps（）:
    s = serial.Serial（'/dev/serial0', 9600, timeout = 10）
    s.readline（）
    while True:
        sentence = s.readline（）.decode （'utf-8'）
        if sentence[0] != '$':
            continue
        for x in sentence:
            gps.update（x）
            if gps.clean_sentences > 20: # データーが溜ったら出力
                break
    now = time.ctime（）
    cnvtime = time.strptime（now）
    print（'id,title,address,latitude,longitude'）
    print（',',end=""）
    print（time.strftime（"%H:%M,", cnvtime）, end=""）
    print （',%2.8f,%2.8f'% （gps.latitude [0], gps.longitude [0]））
    f = open（'data.csv'）
    # 1行スキップ
    line = f.readline（）
    # 60分間分のデータ
    for num in range（60）:
        line = f.readline（）
        print（line, end=""）
    f.close
rungps（）
```

　これと、micropyGPS.pyというプログラムを、https://github.com/inmcm/micropyGPSあたりからダウンロードして、home/pi/maptest01の中にセーブしておいて下さい（行頭に、# coding: UTF-8 を付け加える必要があるかもしれません）

```
$ echo " id,title,address,latitude,longitude" > data.csv
```

として、data.csvを作った後、

```
$ python3 nmea2csv_line.py > tmp; cp tmp data.csv
```

を実行してみて下さい。

```
$ more data.csv
id,title,address,latitude,longitude
,17:00,,35.216812, 137.133924
,16:59,,35.221106, 137.133951
,16:58,,35.228639, 137.137333
```

という感じで、data.csvファイルが1行増えていれば、成功しています。

　さて、このプログラムを、自動的に1分間に1度だけ起動させるように、設定しておきます。

```
$ /etc/crontab
```

に以下を追記して下さい

```
*/1 * * * * pi    python3 nmea2csv_line.py > tmp; cp tmp data.csv
```

（4）Webサーバプログラム（Go言語）の作成

以下のWebサーバプログラムを、main.goという名前で作成して下さい。

```
import （
   "log"
   "net/http"
）
func main（） {
    http.Handle（"/", http.FileServer（http.Dir（"/home/pi/maptest01"）））
   if err := http.ListenAndServe（":8787", nil）; err != nil {
      log.Fatal（"ListenAndServe: ", err）
   }
}
```

その後、go build main.go でビルドすると、main という実行ファイルができます。

これだけのプログラムで、Webブラウザから"http://211.158.177.150:8787"と入力すると、home/pi/maptest01にあるindex.htmlの内容が表示されるようになります（後述）。apache2やnginxのインストールも必要ありません。

さて、このWebサーバプログラムを、ラズパイ起動時に自動起動するようにしておきます。

/etc/systemd/system/maptest01.serviceという名前で、以下を保存して下さい。

```
[Unit]
Description = maptest01 daemon
[Service]
ExecStart = /home/pi/maptest01/main
Restart = always
Type = simple
[Install]
WantedBy = multi-user.target
```

そして、

```
$ sudo systemctl enable maptest01
```

でエントリして下さい。

```
$ sudo systemctl start maptest01
```

とすることで、手動で起動を確認することができます。

（4）index.htmlの作成

今回は、GoogleMapではなく、Mapboxという地図サービスプラットフォームを利用して、地図上に、自動車の移動軌跡（マップピン）を、最大60本立てて、過去1時間の移動経路を表示します。

ちょっと話はズレますが「なぜ、私は、今回Mapboxを選んだか？」ですが、これは、私のGISシステムの師匠とあがめている方の以下の分析結果に従った結果です。

GIS界隈は、10年ほど見ている感じでは、登場人物は次のような感じです。

［商用］

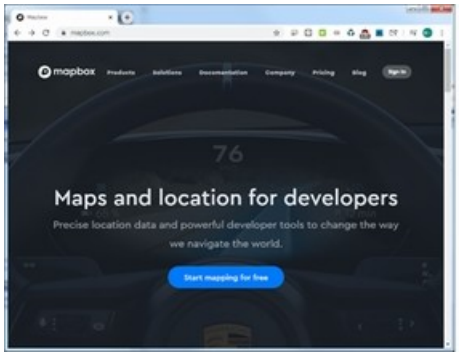
- エスタブリッシュメント ArcGIS
  →https://www.esrij.com/products/arcgis/
- 新興 mapbox→https://www.mapbox.com/
- 新興の対抗 here.com→https://www.here.com/ Uber→http://deck.gl/#/

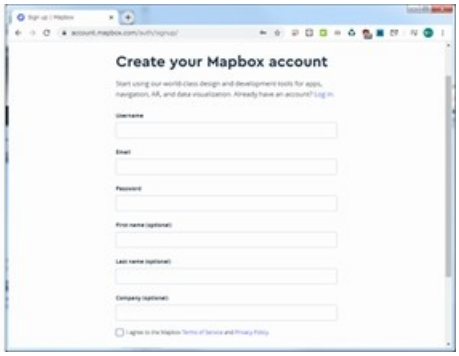
［オープンソース］

国土地理院、OpenStreetMap、PostGISなど各種の確立されたプロダクトやデータ

「無料」のため、やはりオープンソースが最大勢力です。こちらの地図情報については、今後も更新され続けていること、ノウハウの取得や維持更新の確実さから、さらに普及していくものと考えます。

　最初に、Mapboxのトークンを入手して下さい（トークンが何のことか、分からなくてもいいです。とりあえず、必要なのは、あなた個人に付与される文字列です）。https://www.mapbox.com/にアクセスして、気前よく、個人情報をバカスカ入力して下さい。





　すると、メールで、Mapboxを使うための、こんな感じの個人用のトークンが送られてきます。

```
pk.eyJ2IjoidG9tbXVycGh5IiwiYSI6ImNrNzF2b252NTA5czYzZW84ZTdnOWNmbjIifq._ezppDHNneEWX5BtYwAdzw
```

　以下を、index.htmlとして保存して下さい。

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>CSVファイルを地図に表示（mapbox）</title>
<meta name='viewport' content='initial-scale=1,maximum-scale=1,user-scalable=no¥
' />
<script src='https://api.tiles.mapbox.com/mapbox.js/v2.1.9/mapbox.js'></script>
<link href='https://api.tiles.mapbox.com/mapbox.js/v2.1.9/mapbox.css' rel='styl¥
esheet' />
<style>
  body { margin:0; padding:0; }
  #map { position:absolute; top:0; bottom:0; width:100%; }
</style>
</head>
<body>
<script src='https://api.tiles.mapbox.com/mapbox.js/plugins/leaflet-omnivore/v0¥
.2.0/leaflet-omnivore.min.js'></script>
<div id='map'></div>
<script>
// トークンはここで使う
L.mapbox.accessToken = 'pk.eyJ1IjoidG9tbXVycGh5IiwiYSI6ImNrNzF2b252NTA5czYzZW84¥
ZTdnOWNmbjIifQ._ezppDHNneEWX5btYwAdzw';
var map = L.mapbox.map（'map', 'mapbox.streets'）
    .setView（[35.057686, 136.974811], 11）;
omnivore.csv（'data.csv', null, L.mapbox.featureLayer（））.addTo（map）;
</script>
</body>
</html>
```

この後、

```
$ sudo reboot
```

で、

```
Webブラウザから"http://211.158.177.150:8787"と入力
```

して下さい。取りあえず、現在時刻から、過去1時間分（60ポイント）の位置情報が表示されるはずです。また、リロード（また

は、Ctrl+F5）で、最新情報に更新されます ―― 運が良ければ。

なお、最初からうまく動くなんて、甘いこと考えてはいけません。あなたと私のラズパイは完全に同じモノではないからです。

システムというのは、原則「動かないもの」で、その後、ジタバタ、ドタバタを繰り返しながら、なんとかして「動かすもの」です。

「SOS通知システム」を作る

さて、では、次に、（2）の「SOS通知」システムの作り方を説明します。

このシステムは、車内の無線LANシステムを前提としていますで、ラズパイをアクセスポイント（AP）化することが必要となります。

最初に、このAP化の方法について説明します。SIMカード（ここではイプシムSIM）が準備できない場合に備えて、ポケットWi-Fiやスマホのデザリングも使えるようにしておきます。

（1）ラズパイのAP化

ここでは説明用に、以下のMACアドレスとUSB Wi-Fiを想定した環境のラズパイを想定して説明します。

(A)wlan0, wlan1を固定する

USBのWi-Fiを刺すと、wlan0, wlan1が移動してしまうので、固定します(これ、本当に困りました)。事前にifconfigでMACアドレスを取得しておいて下さい。**絶対に内蔵Wi-FiとUSB Wi-Fi を混同しないように注意して下さい。**

/etc/udev/rules.d/70-persistent-net.rules

```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="b8:27:eb:ec:9c:0b", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="wlan*", NAME="wlan0"
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="d0:37:45:30:71:23", ATTR{dev_id}=="0x0",
ATTR{type}=="1", KERNEL=="wlan*", NAME="wlan1"
```

(B)wlan0のIPアドレスの設定

/etc/dhcpcd.conf

```
interface wlan0
static ip_address=192.168.101.1/24
static routers=192.168.101.1
static domain_name_servers=192.168.101.1
```

この後、sudo rebootで、再起動します

(C)isc-dhcp-serverのインストールと設定

/etc/dhcp/dhcpd.confを次の内容にして保存します。

```
default-lease-time 600;
max-lease-time 7200;
ddns-update-style none;
authoritative;
subnet 192.168.101.0 netmask 255.255.255.0 {
        range 192.168.101.10 192.168.101.200;
        option broadcast-address 192.168.101.255;
        option routers 192.168.101.1;
        default-lease-time 600;
        max-lease-time 7200;
        option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

/etc/default/isc-dhcp-serverを編集します。

```
INTERFACESv4=""の箇所をINTERFACESv4="wlan0"に変更して保存します。
```

sudo rebootを実行し、再び再起動します。

(D)hostapdのインストールと設定

hostapdをインストールします。

```
$ sudo apt install hostapd -y
```

/etc/hostapd/hostapd.confを作成し、次の内容を記入して保存します。

ssidとwpa_passphraseの内容は、必要に応じて変更してください。ここでは「rp1」というSSIDにします。

```
interface=wlan0
driver=nl80211
ssid=rp1
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=password
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_group_rekey=86400
ieee80211n=1
wme_enabled=1
```

/etc/default/hostapdを開き、# DAEMON_CONF=""の箇所を
DAEMON_CONF="/etc/hostapd/hostapd.conf"に変更して保存します。

hostapdを実行できるよう、次のコマンドを実施します。

```
$ sudo systemctl unmask hostapd
```

(E)DHCPの起動タイミングの変更

ネットワークよりも先にDHCPサーバが起動する問題を回避するため、待ち時間を設定します。

/etc/init.d/isc-dhcp-serverを編集します。

log_daemon_msg “Starting $DESC” “$NAME”の次の行にsleep 10を追加して保存します。

該当箇所は、次のようになります。

```
log_daemon_msg “Starting $DESC” “$NAME”
sleep 10
```

(F)インターフェースの設定

/etc/network/interfaces.d/wlan0を作成し、次の内容を記入して保存します。

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
     wpa-conf /etc/wpa_supplicant/wpa_supplicant_wlan0.conf
```

/etc/network/interfaces.d/wlan1を作成し、次の内容を記入して保存します。

```
auto wlan1
allow-hotplug wlan1
iface wlan1 inet dhcp
     wpa-conf /etc/wpa_supplicant/wpa_supplicant_wlan1.conf
wlan0用のWi-Fi接続設定を/etc/wpa_supplicant/wpa_supplicant_wlan0.confに作成します。
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=JP
network={
       ssid="ssid"
       psk="password"
       key_mgmt=WPA-PSK
}
```

wlan1用のWi-Fi接続設定を/etc/wpa_supplicant/wpa_supplicant_wlan1.confに作成します。

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=JP
network={
       ssid="iPhone"
       psk="9yniyg4p4cmh1" （これ私のiPhoneのデザリングのパスワード（データメ））
}
```

sudo rebootを実行し、OSを再起動します。

(G)稼働確認

ifconfigで、以下の(ような)内容が表示されればO.K.です。

---

wlan0: flags=4163 mtu 1500
inet 192.168.101.1 netmask 255.255.255.0 broadcast 192.168.101.255
inet6 fe80::1392:b0b4:b56a:836b prefixlen 64 scopeid 0x20
ether b8:27:eb:ec:9c:0b txqueuelen 1000（イーサネット）
RX packets 685 bytes 61030（59.5 KiB）

RX errors 0 dropped 0 overruns 0 frame 0
TX packets 148 bytes 23267（22.7 KiB）
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
wlan1: flags=4163 mtu 1500
inet 172.20.10.4 netmask 255.255.255.240 broadcast 172.20.10.15
inet6 fe80::e4a8:f21f:a6fa:111f prefixlen 64 scopeid 0x20
ether d0:37:45:30:71:23 txqueuelen 1000（イーサネット）
RX packets 28 bytes 4774（4.6 KiB）
RX errors 0 dropped 30 overruns 0 frame 0
TX packets 50 bytes 8415（8.2 KiB）
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

表示させたいタブレットで、「rp1」というSSIDのアクセスポイントが見えて、パスワード"password"で接続できれば成功です。

(H) wlan0(無線LAN) ←→wwan0(4GPi)でインターネットに出られるようにする

念のため、最初にリブートしておきます。

```
$ sudo reboot
```

以下、手動で以下のコマンドを入力して下さい。

```
$ sudo iptables -xnvL
$ sudo iptables -F
$ sudo iptables -X
$ sudo iptables -P INPUT ACCEPT
$ sudo iptables -P OUTPUT ACCEPT
$ sudo iptables -P FORWARD ACCEPT
$ sudo iptables -t nat -A POSTROUTING -o wwan0 -j MASQUERADE
```

iptables が起動時に設定されるようにしておきます。

```
$ sudo apt-get install iptables-persistent
```

インストール時に現在のiptablesをipv4,ipv6ともに保存するか聞かれるので、「YES」としておいて下さい。最期に $ sudo rebootして終了です。

さて、ここで、ようやく車内LANの準備が整いました。ここから、「SOS通知」システムを作ります。まず、実施イメージをこの動画で確認して下さい。

簡単に言うと、どのクライアントからでも表示ボタンを押すと、全部のクライアントが一斉に連動するというものです。実際には、私が自宅から表示内容を変えて、自動車の窓に張りつけたタブレットの表示を変更させます。

(2)(表示画面用)ディレクトリの作成

どこにどういう名前でもいいですが、ディレクトリを作ります。(ここではwstest4としました)

```
cd ~; mkdir wstest4; cd wstest4
```

(3)(表示画面用)htmlの作成

wstest4の中に、さらに、staticというディレクトリを作って、そこに表示画面のhtmlファイルを作ります。

```
cd ~/wstest4; mkdir staic; cd static
```

blue.html

```
<!DOCTYPE html>
<html lang="ja">
 <head>
  <link rel="stylesheet" type="text/css" href="blue-style.css">
  <meta charset="utf-8">
 </head>
 <body>
  <div class="central">
   <div class="subtitle">
    移動中
   </div>
  </div>
 </body>
</html>
```

上記のファイルの一部を変更して、あと2つファイルを作成して下さい。

ファイル名	変更点1	変更点2
yellow.html	blue-style.css → yellow-style.css	移動中→迷走中
red.html	blue-style.css → red-style.css	移動中→012-345-6789 に電話下さい

blue-style.css

```
body {
  background: blue;
  color: white;
  overflow: hidden;
}
.central {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate（-50%, -50%）;
  text-align: center;
  width: 100%;
  font-size: 220px;
  font-weight: bold;
  -webkit-animation:blink 1.5s ease-in-out infinite alternate;
  -moz-animation:blink 1.5s ease-in-out infinite alternate;
  animation:blink 1.5s ease-in-out infinite alternate;
}
@-webkit-keyframes blink{
  0% {opacity:0;}
  100% {opacity:1;}
}
@-moz-keyframes blink{
  0% {opacity:0;}
  100% {opacity:1;}
}
@keyframes blink{
  0% {opacity:0;}
  100% {opacity:1;
}
```

上記のファイルの一部を変更して、あと2つファイルを作成して下さい

ファイル名	変更点1
yellow-style.css	blue → yellow
red-style.css	blue → red

(4)（表示画面用）Webサーバプログラム（Go言語）の作成

```
$ cd ~/wstest4
```

とした後、以下のWebサーバプログラムを、main.goという名前で作成して下さい。

```
import （
   "log"
   "net/http"
）
func main（） {
    http.Handle（"/", http.FileServer（http.Dir（"/home/pi/wstest4/static"）））
   if err := http.ListenAndServe（":8686", nil）; err != nil {
      log.Fatal（"ListenAndServe: ", err）
   }
}
```

その後、go build main.go でビルドすると、main という実行ファイルができます。

(5)（連動表示制御用）ディレクトリの作成

どこにどういう名前でもいいですが、ディレクトリを作ります（ここではwstest33としました）

```
cd ~; mkdir wstest33; cd wstest33
```

(6)(連動表示制御用)Webサーバプログラム(Go言語)の作成

```
$ cd ~/wstest33
```

とした後、以下のWebサーバプログラムを、websocket-server.goという名前で作成して下さい。

```
import （
    "log"
    "net/http"
    "github.com/gorilla/websocket"
）
// WebSocket サーバーにつなぎにいくクライアント
var clients = make（map[*websocket.Conn]bool）
// クライアントから受け取るメッセージを格納
var broadcast = make（chan Message）
// WebSocket 更新用
var upgrader = websocket.Upgrader{}
// クライアントからは JSON 形式で受け取る
type Message struct {
    Message string `json:message`
}
// クライアントのハンドラ
func HandleClients（w http.ResponseWriter, r *http.Request） {
    // ゴルーチンで起動
    go broadcastMessagesToClients（）
    // websocket の状態を更新
    websocket, err := upgrader.Upgrade（w, r, nil）
    if err != nil {
        log.Fatal（"error upgrading GET request to a websocket::", err）
    }
    // websocket を閉じる
    defer websocket.Close（）
    clients[websocket] = true
    for {
        var message Message
        // メッセージ読み込み
        err := websocket.ReadJSON（&message）
        if err != nil {
            log.Printf（"error occurred while reading message: %v", err）
            delete（clients, websocket）
            break
        }
       // メッセージを受け取る
        broadcast <- message
    }
}
func main（） {
    http.HandleFunc（"/", func（w http.ResponseWriter, r *http.Request） {
        http.ServeFile（w, r, "/home/pi/wstest33/index.html"）
    }）
    http.HandleFunc（"/chat", HandleClients）
   err := http.ListenAndServe（":8080", nil）
    if err != nil {
        log.Fatal（"error starting http server::", err）
        return
    }
}
func broadcastMessagesToClients（） {
    for {
        // メッセージ受け取り
        message := <-broadcast
        // クライアントの数だけループ
        for client := range clients {
        //書き込む
            err := client.WriteJSON（message）
            if err != nil {
                log.Printf（"error occurred while writing message to client: %v",
 err）
                client.Close（）
                delete（clients, client）
            }
        }
    }
}
```

その後、$ go build websocket-server.go でビルドすると、websocket-server という実行ファイルができます。

(7)(連動表示制御用)htmlの作成

wstest4の中にindex.htmlを作成して下さい。

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>WebSocket Server</title>
 </head>
 <body>
  <input id="Button1" type="button" value="移動中" onclick="onButtonClic
k1（）;" />
  <input id="Button2" type="button" value="迷走中" onclick="onButtonClick2（）;"
 />
  <input id="Button3" type="button" value="緊急電話" onclick="onButtonClick3（）;"
 />
  <p id="input"></p>
  <p id="input1"></p>
  <p id="input2"></p>
  <p id="input3"></p>
  <pre id="output"></pre>
  <iframe id="image_place" src="http://192.168.101.1:8686/red.html" width="100
%" height="700"></iframe>
  <script>
   var input1 = document.getElementById（'input1'）;
   var input2 = document.getElementById（'input2'）;
   var input3 = document.getElementById（'input3'）;
   var img = document.getElementById（"image_place"）;
   var output = document.getElementById（'output'）;
   // socketの変数は、connect（）の外に出しておかないとボタンに届かない
   var socket;
   function connect（） {  // で、ここから connect（）を括る
   socket = new WebSocket（"ws://" + window.location.host + "/chat"）;
   socket.onopen = function（） {
   };
   socket.onmessage = function（e） {
   var obj = JSON.parse（e.data）; // JSONオブジェクトに変換
   switch（obj.Message）{
   case '1':
   img.src = "http://192.168.101.1:8686/red.html";
   break;
   case '2':
   img.src = "http://192.168.101.1:8686/yellow.html";
   break;
   case '3':
   img.src = "http://192.168.101.1:8686/blue.html";
   break;
   }
   }
   socket.onclose = function（e） {
   console.log（'Socket is closed. Reconnect will be attempted in 1 second.',
e.reason）;
   setTimeout（function（） {
    connect（）;
   }, 1000）;
   };
   }  // で、ここから connect（）を閉じる
   connect（）;  // で、ここでconnect（）を起動する
   function onButtonClick1（）{
    input1.value = "1";
    socket.send（JSON.stringify（
    {
     message: input1.value
    }
    ））;
    input1.value = "";
   };
   function onButtonClick2（）{
    input2.value = "2";
    socket.send（JSON.stringify（
    {
     message: input2.value
    }
    ））;
   input2.value = "";
   };
   function onButtonClick3（）{
    input3.value = "3";
```

```
      socket.send（JSON.stringify（
      {
        message: input3.value
      }
      ））;
      input3.value = "";
     };
    </script>
  </body>
</html>
```

(8)自動起動化

　上記の2つのGoで作ったサーバを自動起動するようにしておきます。

　正直、2つのWebサービスを作るのは美しくない（というか醜悪）ですが、もういろいろ考えるのが面倒なので、このまま強行します。

　以下を/etc/systemd/system/kanban.serviceとして、保存して下さい。

```
[Unit]
Description = kanban daemon
[Service]
ExecStart = /home/pi/wstest4/main
Restart = always
Type = simple
[Install]
WantedBy = multi-user.target
```

```
$ sudo systemctl enable kanban
```

でエントリして、

```
$ sudo systemctl start kanban
```

で、起動確認をして下さい。

　次に、以下を/etc/systemd/system/kanban-cont.serviceとして、保存して下さい。

```
[Unit]
Description = kanban-cont daemon
[Service]
ExecStart = /home/pi/wstest33/websocket-server
Restart = always
Type = simple
[Install]
WantedBy = multi-user.target
```

```
$ sudo systemctl enable kanban-cont
```

でエントリして、

```
$ sudo systemctl start kanban-cont
```

で、起動確認をして下さい。

　起動確認できていれば、sudo rebootで再起動して下さい。

　車載のタブレットおよび自宅のPCのWebブラウザから**"http://211.158.177.150:8080"**と入力して3つのボタンを押下してください。青、黄色、赤のSOSの画面が切り替わるはずです。

「呼びかけシステム」を作る

では最後に、(3)の「「呼びかけ」システム」の作り方を説明します。

このシステムは、自宅のPCから音声ファイルを起動して、運転手にメッセージを送付するものです。

(1)秋月電子通商で、TPA2006使用 超小型D級アンプキットを購入します

(2)音声ファイルを作って、ラズパイに放り込みます

私の場合は、「ボイスロイド「結月ゆかり」」で音声ファイルを作って(参考)、mp3ファイルに変換した後、SFTP(SSH File Transfer Protocol)で音声ファイルを/home/piに送り込みました。音声ファイルのサンプルはこちらをクリックしてください。

(3)mpg321とgpioコマンドをインストールします

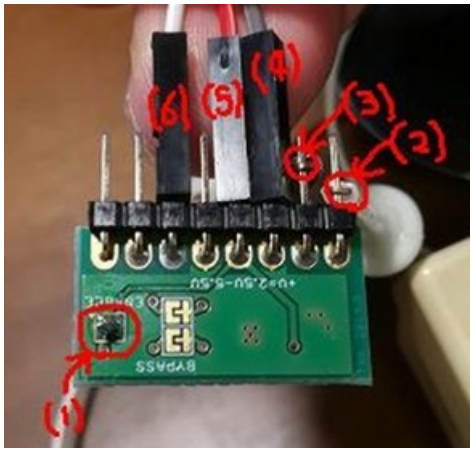
```
$ sudo apt-get install mpg321
$ sudo apt-get install wiringpi
```

(4)gpioの稼働を確認します

```
>gpio -v
```

```
gpio version: 2.50
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty
Raspberry Pi Details:
Type: Pi 3B+, Revision: 03, Memory: 1024MB, Maker: Sony
* Device tree is enabled.
*--> Raspberry Pi 3 Model B Plus Rev 1.3
* This Raspberry Pi supports user-level GPIO access.
```

(5)TPA2006の、はんだ付けと配線をします



(1)の部分をはんだ付けしてショートさせます(理由は知りません)

(2)と(3)のピンをスピーカー(8Ω 0.3W)に接続します

(4)のピンをラズパイのグランド(6番ピン)に接続します

(5)のピンをラズパイの3Vへ(1番ピン)へ接続します

(6)のピンをGPIO18(12番ピン)に接続します。

(6)簡単な試験

```
>sudo gpio -g mode 18 pwm
>mpg321 wait4me.mp3
```

これで音声がスピーカーから聞こえてきたら成功です。

```
>alsamixer
```

これで、ボリューム調整ができます。

(7)その他

GPIO18(12番ピン)は、みちびきの1PPS信号用に差してあるが、当面使う予定はないので、こっちを外して音源にして使うことにしました。

(8)自動起動化

(A)概要

コマンドから、"sudo gpio -g mode 18 pwm"をやるのと同じイメージです。

(B)設定

/home/pi/sounder.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import os
import time
os.system（"sudo gpio -g mode 18 pwm"）
```

```
$ sudo chmod +755 /home/pi/sounder.py
```

/usr/lib/systemd/system/sounder.service

```
Description=Sounder Daemon
[Service]
ExecStart=/home/pi/sounder.py
Restart=always
Type=simple
[Install]
WantedBy=multi-user.target
```

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable sounder
```

再起動した後、TeraTermから車載の"$ mpg321 wait4me.mp3"を押下して、音声が出力されれば成功です。

以上、3つのサービスの構築方法について説明しましたが、実際の運用では課題が山積です。

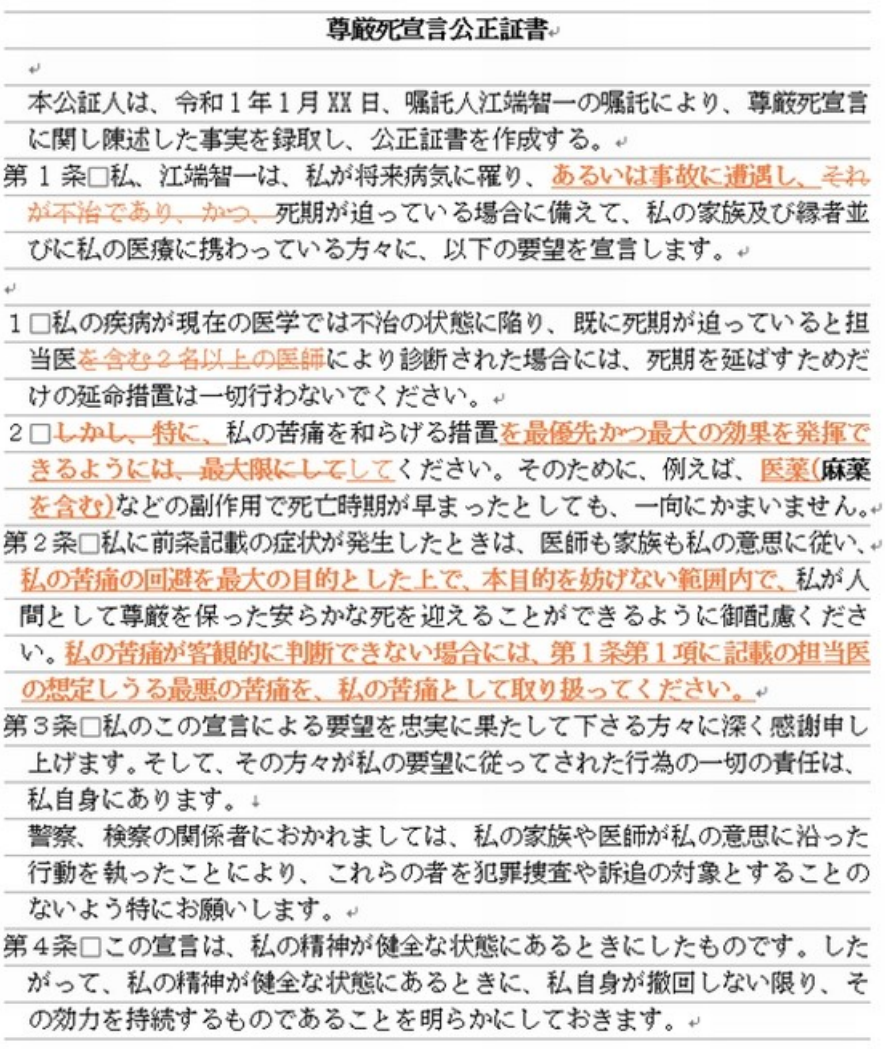
特に、「SOS通知システム」については、タブレットを自動起動させる方法が分かっていません。父（故人）が自分で「タブレットの電源入れて、無線LANの設定をして、ブラウザを起動する」なんてことは、そもそも想定がナンセンスです（そういう操作ができるくらいなら、行方不明になったりしない）。

一番望ましいことは、最初から、こういうサービスが最初からイントールされている自動車が販売されることでしょう ――そういう自動車が売れるかどうかは不明ですが、これからの日本においては、潜在ユーザー数が増えていくことは確実です。

私の「尊厳死宣言書」を見た現役医師のレビュー

さて、ここから後半です。

前回のコラムで、私は自作の「尊厳死宣言書」のドラフトの作成を行ったというお話をしました。このドラフトは「尊厳」よりも、「苦痛回避」を全面に押し出しました。



尊厳死宣言公正証書

本公証人は、令和１年１月 XX 日、嘱託人江端智一の嘱託により、尊厳死宣言に関し陳述した事実を録取し、公正証書を作成する。

第１条□私、江端智一は、私が将来病気に罹り、~~あるいは事故に遭遇し、それが不治であり、かつ、~~死期が迫っている場合に備えて、私の家族及び縁者並びに私の医療に携わっている方々に、以下の要望を宣言します。

１□私の疾病が現在の医学では不治の状態に陥り、既に死期が迫っていると担当医~~を含む２名以上の医師~~により診断された場合には、死期を延ばすためだけの延命措置は一切行わないでください。

２□~~しかし、特に、~~私の苦痛を和らげる措置~~を最優先かつ最大の効果を発揮できるようには、最大限にして~~してください。そのために、例えば、医薬(麻薬を含む)などの副作用で死亡時期が早まったとしても、一向にかまいません。

第２条□私に前条記載の症状が発生したときは、医師も家族も私の意思に従い、私の苦痛の回避を最大の目的とした上で、本目的を妨げない範囲内で、私が人間として尊厳を保った安らかな死を迎えることができるように御配慮ください。私の苦痛が客観的に判断できない場合には、第１条第１項に記載の担当医の想定しうる最悪の苦痛を、私の苦痛として取り扱ってください。

第３条□私のこの宣言による要望を忠実に果たして下さる方々に深く感謝申し上げます。そして、その方々が私の要望に従ってされた行為の一切の責任は、私自身にあります。

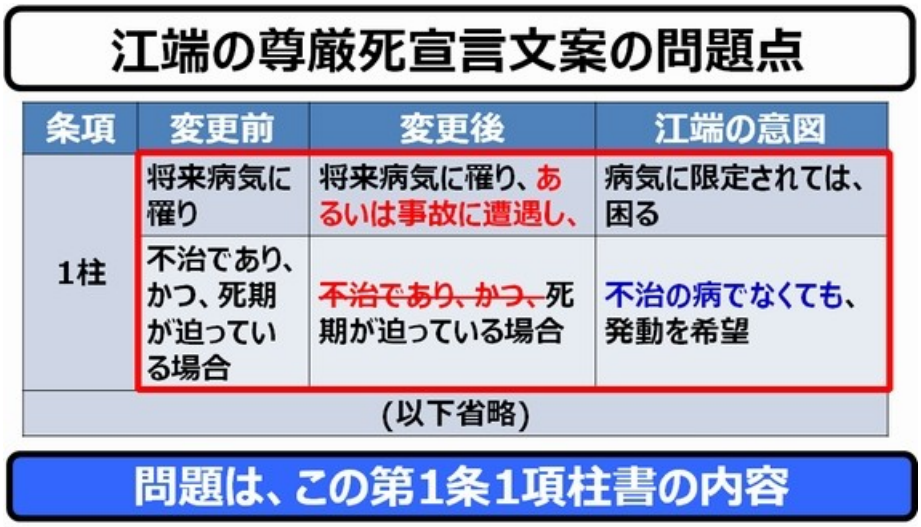
警察、検察の関係者におかれましては、私の家族や医師が私の意思に沿った行動を執ったことにより、これらの者を犯罪捜査や訴追の対象とすることのないよう特にお願いします。

第４条□この宣言は、私の精神が健全な状態にあるときにしたものです。したがって、私の精神が健全な状態にあるときに、私自身が撤回しない限り、その効力を持続するものであることを明らかにしておきます。

前回の原稿を、大学の医学部に所属の医師であり、かつ「1/100秒単位でシミュレーションした「飛び込み」は、想像を

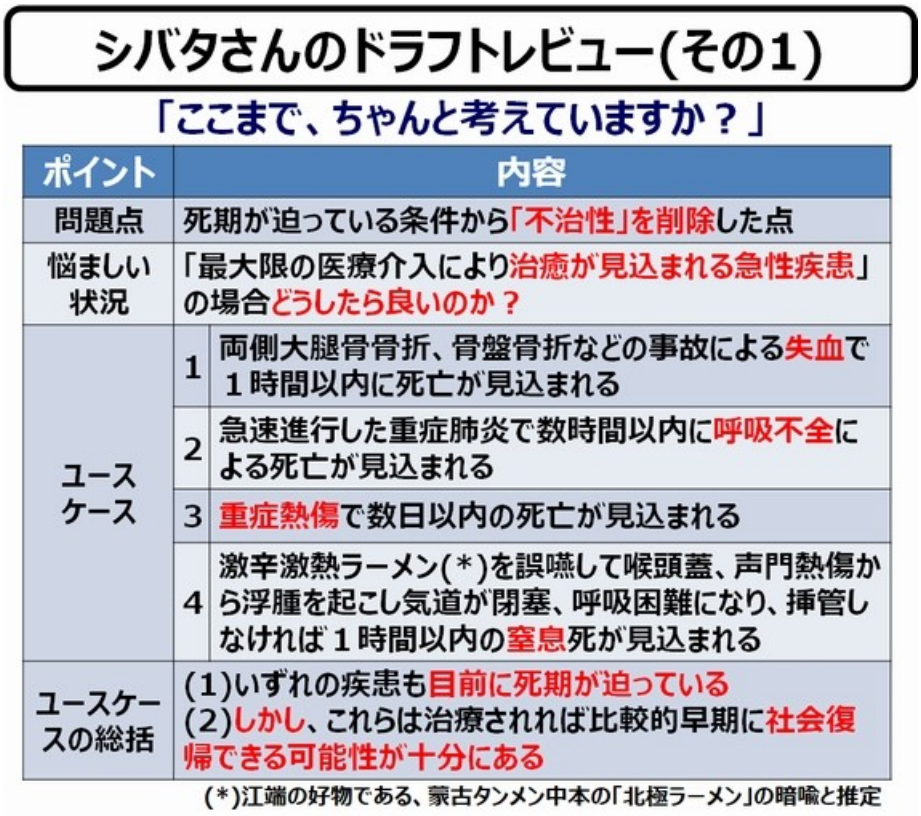
絶する苦痛と絶望に満ちていた」の解析の時に、大変お世話になった、ご自称「轢断のシバタ」さんにレビューして頂きました。

その結果、「現場の医師の業務を混乱させるダブルスタンダード」が含まれている旨の問題点を教えて頂きました。今回は、このお話から始めたいと思います。



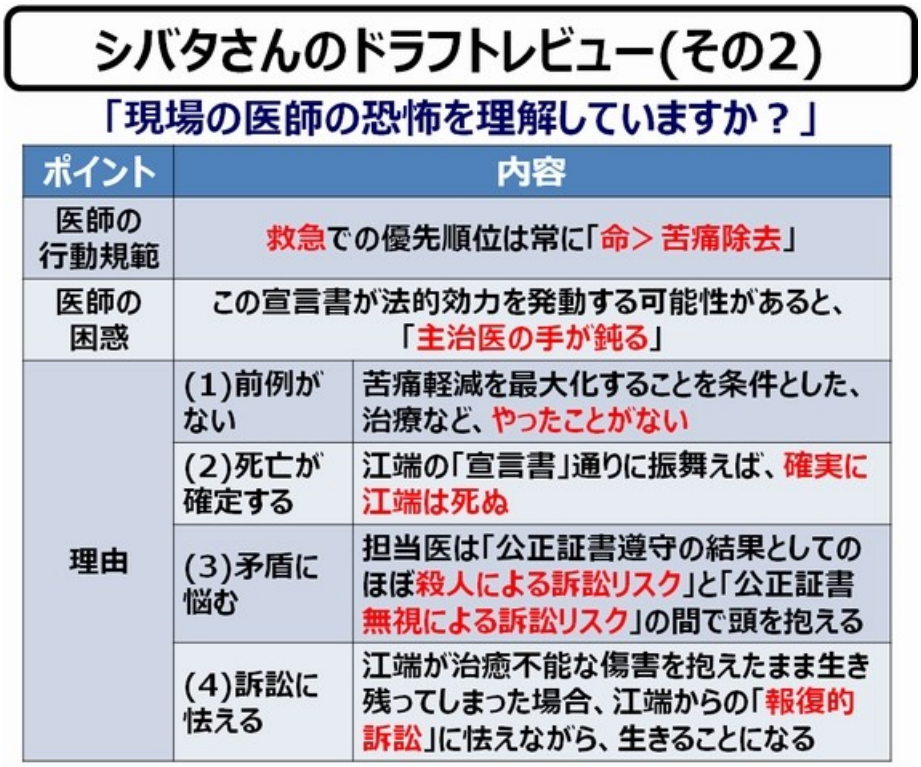
私は、「苦痛回避」を優先するため、「病気」に加えて「事故」を加えて、さらに「不治の病」の限定を外す旨の記載をしました。

これに対して、シバタさんは、以下の問題点を指摘されてきました。



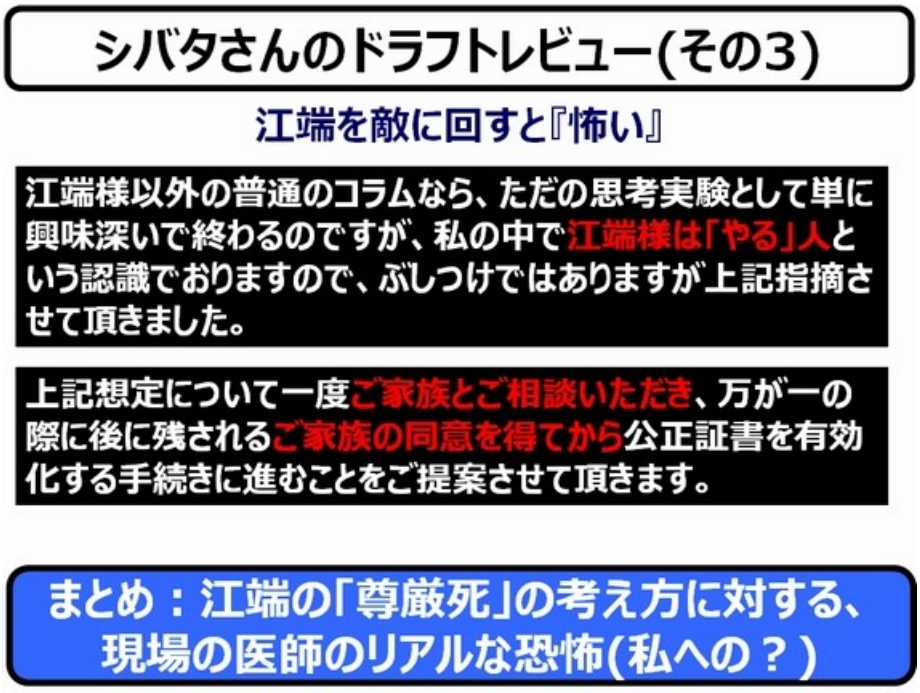
上記に記載した通り『「不治の病」の限定を外す』ことで、助かる可能性の高い命があることを4つのユースケース（大量失血、呼吸困難、大火傷、窒息）で具体的に示されていました。つまり「放置すれば死亡確実。処理すれば救命可能」状態です。

さらに、シバタさんは、通常の医師の行動規範を規定され、この宣言書が、「医師を追い詰めることになる」ことを、4つの理由から明示されました。



医師は、通常の行動規範と、宣言書の法的効力の2つの矛盾の間で、その能力を発揮できない状況に陥るという現実を、上記の4つの理由（前例なし、患者の死亡の看過、法的なダブルスタンダード、報復的訴訟の恐怖）から、具体的に説明されています。

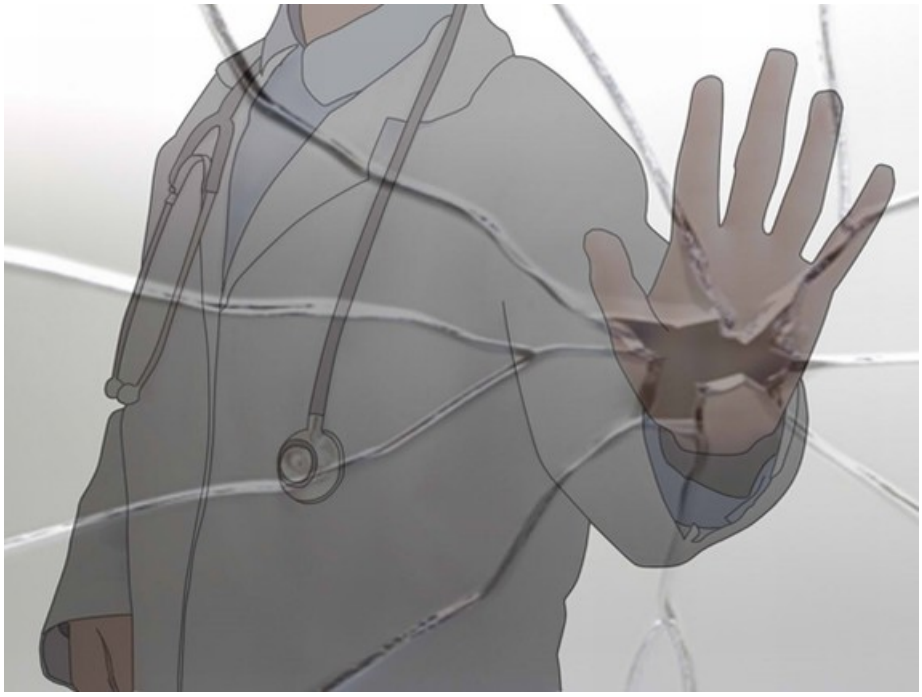
そして、最後は、江端個人（あるいは江端のような思想と動機を持つ誰か）に特化したメッセージとなっていました。



つまり、「江端みたいな奴が、世の中に一人でもいる」と考えると（まあ、実際に1人はいる訳ですが）医師は非常に困るのであって、そこのところも、ちゃんと考えてくれないと困る、という、現場の医師の切実な声が聞こえてくる、

―― 江端の「尊厳死」の考え方に対する、現場の医師のリアルな恐怖

が伝わってくる、シバタさんのレビューレポートでした。



　前回も記載しましたが、2014年5月23日に内閣府が、一般社団法人日本尊厳死協会の公益認定申請に対し、却下の処分をしました。しかし、その後の裁判によって不認定処分の取消（内閣総理大臣の負け）が確定しました（2019年11月14日）。

　内閣不の却下処分には、尊厳死に関する上記の「現場の医師のリアルな恐怖」が色濃く反映されていると思っています。また、悪意で考えれば、「終末治療というビジネスチャンスの損失」を恐れた団体からの圧力があったかもしれません（私の邪推です）。

　実際、シバタさんのレポートの中にも、現在の医療の一部が「"収益＞苦痛除去"になりがちなのは否定しきれません」とも記載があり、医療ビジネスが、患者の苦痛回避を最優先していないかもしれないことも、記載されていました。

　ともあれ、私個人としては、私は、このレポートを何度も読み直し、ドラフトの第1項柱書の記載を元の状態に戻すことにしました。現場の担当医を困らせることは、私の本意ではないからです。

　ところが、地元の公証役場に予約の電話を入れたところ、『申し訳ありませんが、新型コロナウイルス*)の影響で、4月まで受付を中止しております』と言われてしまいました。

*）2020年3月時点で有効な医薬、治療法がなく、致死率は低いものの、全世界で肺炎死亡者が確認されているウイルス性疾患。政府は、2020年2月27日、全国すべての小中高、特別支援学校を対象に3月2日から春休みまで臨時休校を要請する、という歴史的決定をした。

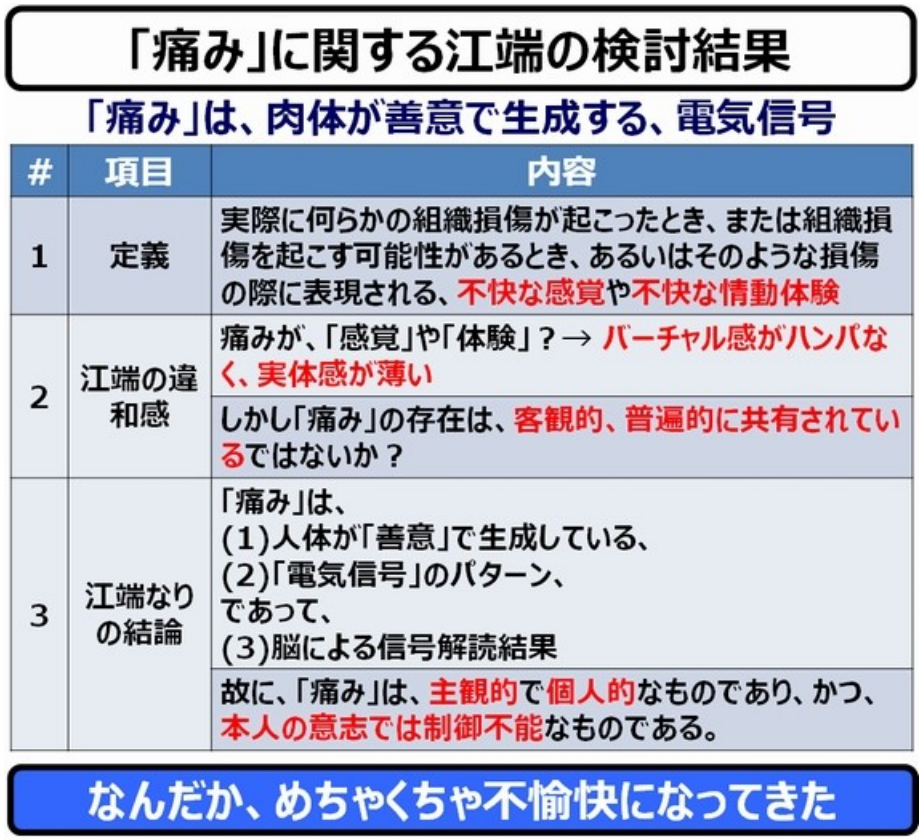
　――というわけで、残念ながら、現時点で、私の自作の「尊厳死宣言書」を、公証役場に届け出ることができません。このウイルス疾患の収束までは、「尊厳死宣言書」が不要となるように、健康と事故には十分に留意していきたいと思います。

「痛み」は何のため？

　さて、それでは、最後に、本シリーズ第1回で掲げたテーゼ、「介護ITの究極の目的は『苦痛の定量化／見える化』であるとした上で、『「苦痛」が動かす社会の未来像』を明らかにする」についての私なりの検討結果をご報告致します。

　まず私は、「苦痛」の中でも、特に肉体に発生するもの ―― これを、ここでは「痛み」と言うことにします ―― についての

定義から調べ始めました。その結果、**結構、不愉快な事実**を知るに至りました。



「痛み」は ーー あんなに痛くて、辛くて、苦しいのに ーー 「感覚」や「体験」であり、主観的で個人的なものである、という事実です。

「痛み」というのは、結局のところ「生体の電気信号の伝達」と「脳による信号の解釈の結果」ということを知り、不愉快が最大値に至りました。

つまり「痛み」は、「痛い」ことそのものでなく、**痛いと感じるように設計された体のシステムの「出力」にすぎない**のです。

誰が何のために、そんな厄介な仕組みを作りやがったのかと思い、私は「痛み」の意義について調べてみました。その結果、「「痛み」は、体の異常の発見のためにある」ということが分かりました。

ちょっと話はズレますが、現在、我が国では、鉄道や電力や道路や橋などの社会インフラシステムの老朽化が、深刻な社会問題となっています。これは、昨今の税収減や人材不足で、インフラに対してメンテナンス（保守）に十分な人材とお金を投入できないからです。

現在のインフラのメンテナンスは、TBM（Time Based Maintenance） —— 故障の有無に関係なく定期的にメンテナンスを実施する考え方で行われています。「予防保全」とも言われます。

ところが、これは大変なお金と時間がかかるので、現在、この業界では、CBM（Condition Based Maintenance）の研究が多いに流行っています。CBMは、老朽化や異常検知といった設備の状態を、「AI技術」等を用いて予知し、必要なインフラだけを保全するという素晴しい新しい保守方法です —— が、私が知る限り、芳しい成果を上げているCBMの実例はありません*)。

*）なにしろ、私、PCで1億行を超えるセンサーデータの分析を行ったエンジニアです。

つまるところ、**鉄道や電力や道路や橋が、自分から「痛い！痛い!!」と叫んでくれればいいのです** 。それなら、私たちは「痛がる道路」や「痛がる橋」の「痛い箇所」だけを修理すればいいのですから。

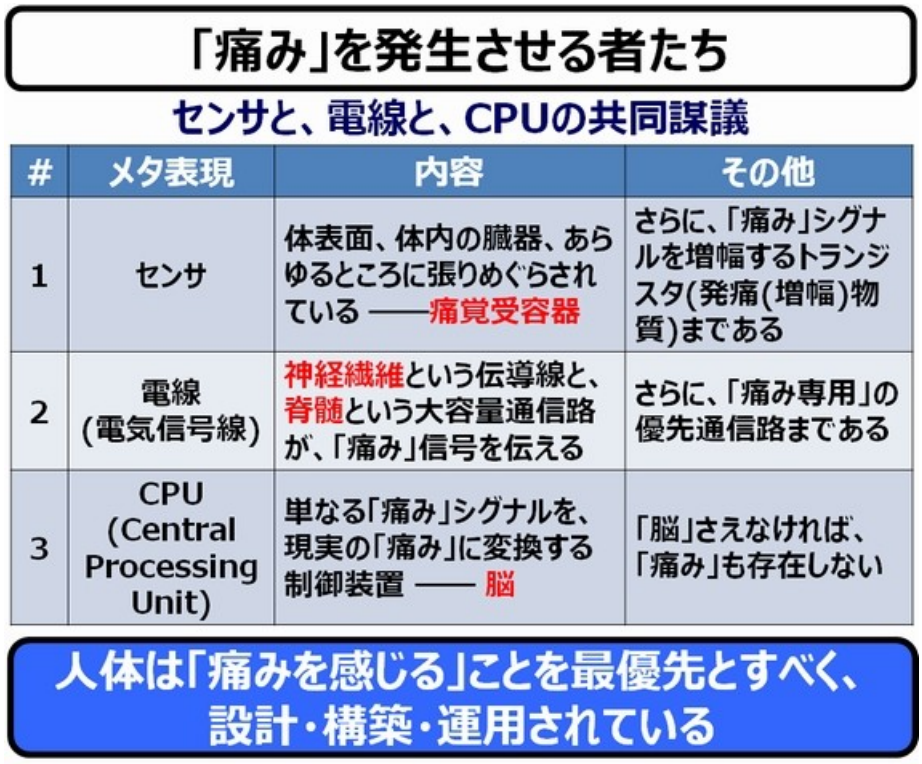
そこにくると、人間の「痛み」による検知システムは、究極のメンテナンスシステムと言えます。「痛み」は、身体の疾患や怪我の「原因」「場所」「程度」の全てがリアルタイムで分かるという、CBMの最終形です。

しかも、「耐えられないような激痛」という信号を作り出すことで、保守対象（私たち）を、修理工場（病院）に強制的に連

行するという、究極の保守システムです。

　不愉快なことですが、「痛い」は**人体が人工的に作り出した感覚**で、その「痛い」が、患部の状況と完全に一致している保証は、何もないのです。さらに、「痛い」が、必要以上に強く発動することで、私たちの日常生活の質を低下させているのも事実です。

　一体、どこの誰が、**こんな迷惑な「痛み」を作り出しているかというと、基本的には「センサ」と「電線」と「CPU」**なのです。基本はパソコンやIoTシステムと同じ仕組みです。



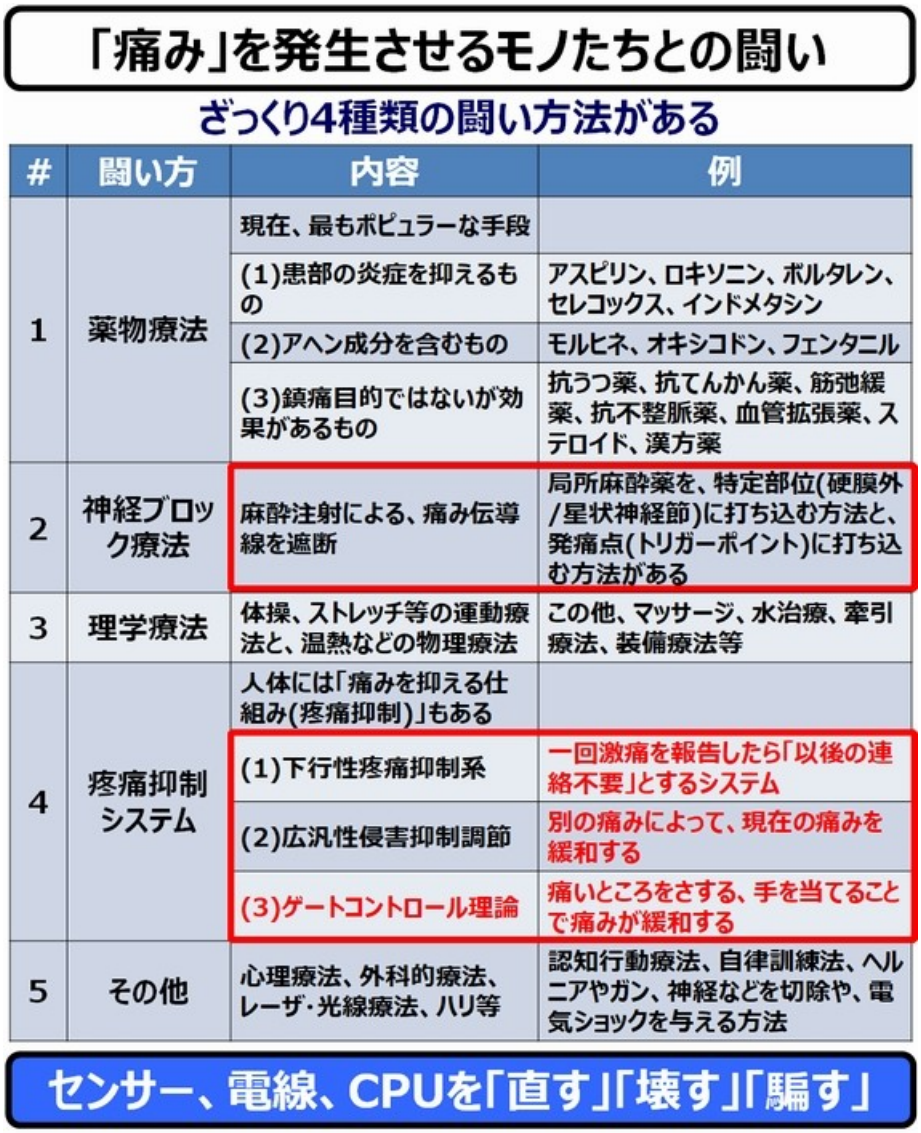
　私、今回のコラムを執筆するに際して、入門書から学会論文まで、さまざまな文献に目を通してきたのですが、その結果として「**肉体は『痛み』を作るためなら、どんな努力も惜しまない**」というように設計され、構築され、運用されていることを知りました。

―― 一体、どこの誰が、ここまで周到な「『痛み』システム」を作ったのか

　もちろん、この答えは明白です。「**種を保存するためであれば、なんだってやる。個体の一つや二つ、死ぬまで痛めつけたって構わん**」という、個人の人権を無視した種の生き残り戦略 ――「生存本能」です。

　私たちが生物であること、それ自体が、「痛み」を常に最大限発揮できる状態として常にスタンバイしているのです ―― 頼みもしていないのに。

　しかし、このような「『痛み』システム」に対して、私たち人類も座していた訳ではありません。以下は、その闘い方をまとめたものです。



「患部を治癒する」ということが、この「『痛み』システム」の狙いでしょうが、全てのケガや疾患が、一瞬で治癒できる訳ではありませんし、完治できないものだってあります。

それらに対して「痛み」を与え続ける、この低能システムは、反人道的なシステムと言えます ーー まあ「本能」に「人道」を説いても無駄ですが。

人類は、これらのシステムの一部を「壊す(例:神経ブロック療法)」や「だます(例:アヘン成分を含むモルヒネなど)」などの対抗措置を取ってきましたし、実際のところ、肉体には、過度な痛みを抑制する機能も(一応)用意されています(疼痛抑制システム)。

この疼痛抑制システムで、興味深い理論が「ゲートコントロール理論」です ーー『痛いの痛いの飛んで行けー』と、患部をさすることで痛みが緩和される「あれ」です。

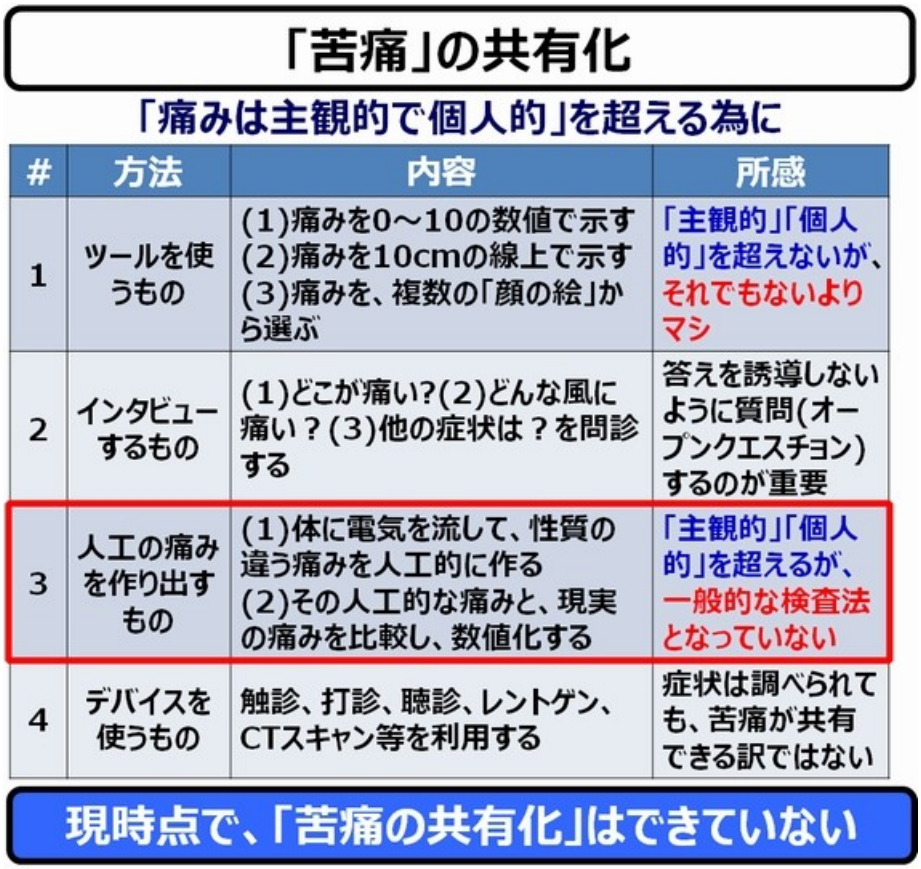
これ、理論としては簡単で、痛みの信号の通過地点(患部)をさすると、痛み信号のゲートが閉じる(信号線が遮断される)というものです(現在も検証が続けられています)。

さらに、この理論を応用して、別の部位に、通電デバイスを使って、痛み(と認識できる前のギリギリの痛み)を発生させることで、その痛みを緩和させる(忘れさせる)という治療法(経皮的電気神経刺激法)も開発されました。

「痛みの共有化」は可能なのか

では、『苦痛の定量化/見える化』さらには『苦痛の共有化』について、現状どうなっているかを、ざっくり調べてみまし

た。



#2のインタビュー（問診）や#4の装置（デバイス）を使う方法は、古くからの「他人の苦痛を知る手段」として一般的です。#1のツールを使う方法も、個人の主観を超えるものではありませんが、それでも数値化/離散化できる、という点においては、何もしないよりは、ずっとマシだと思います。

私としては#3の「人工の痛みを作り出すもの」が、現時点で最も有望な「苦痛の共有化」だと思いました。自分の感じている痛みと同程度の痛みを装置で作り出せれば、それを他人に体感してもらうことができるからです ―― 理屈の上では。

しかし、これは、現時点では「一般的な検査法」とはなっていないようです。これについての理由を探してみたのですが、私は見つけられませんでした。

そこで私は、2つの仮説を考えてみました。

（仮説1）この装置は痛みを数値化する目的でのみ使われており、第三者への痛みの共有としては機能を発揮できない

その装置で生成した痛みは、結局のところ、その患者本人の個体の状態（体格、年齢、性別、身体の弱り方、神経の反応等）に依存するため、結局のところ、完全に同一の痛みは共有できない

（仮説2）そもそも、他人の痛みなど共有したくない

他人の痛みをわざわざ自分の体で再現するなど、つらいに決まっている。そんなことは、やらずに済むに越したことはない

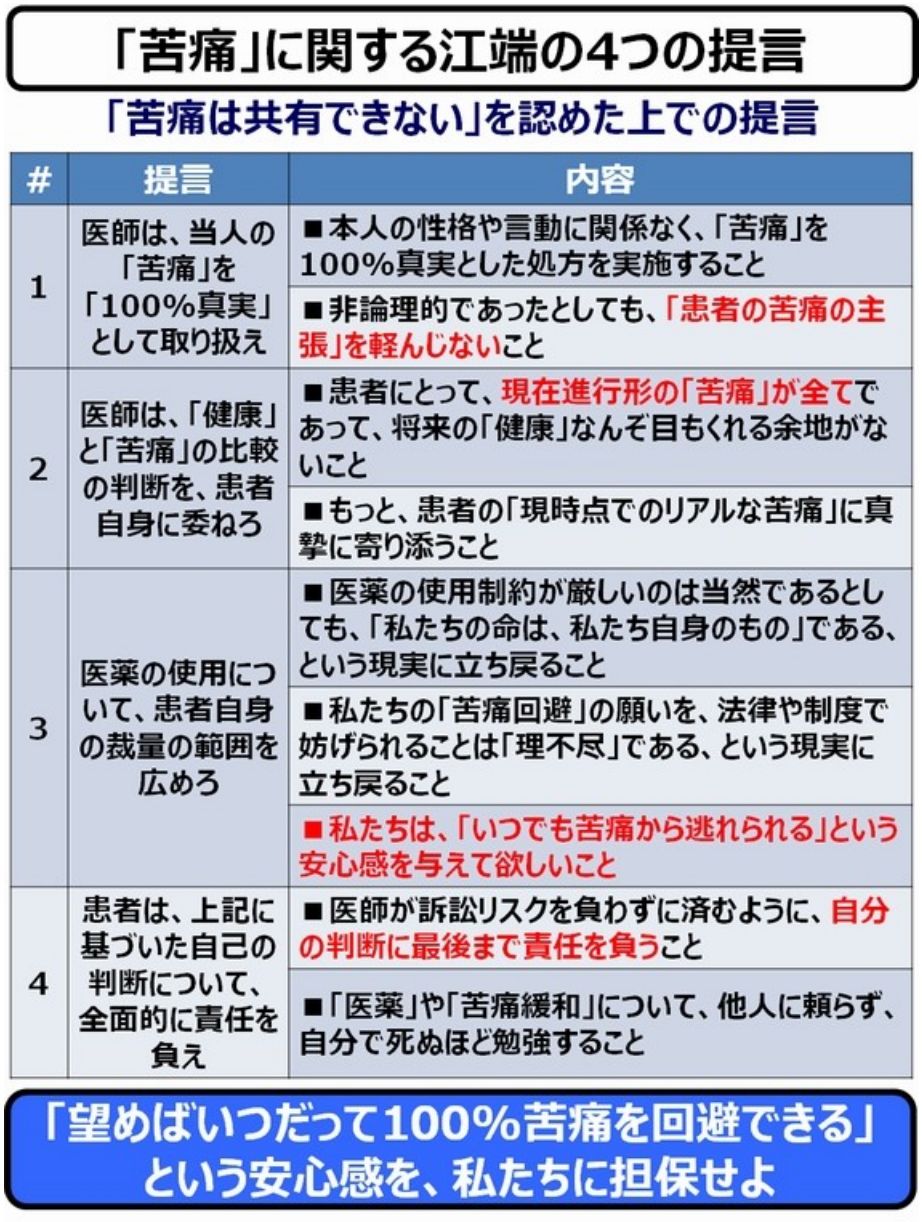
私たちは、「自分の痛み」には真剣ですが、「他人の痛み」については、本当のところ『どーでもいい』と思っているのではないでしょうか。「他人の痛みを感じられる人間になりましょう」という子ども向けのメッセージは、メッセージのままにしておくのが、誰にとってもラクです ―― その本人を除けば。

私は、あなたを責めている訳ではありません。私だってそう思うからです。毎日、他人の痛みを自分の体で検査する ―― そんな仕事は正直、ゴメンだと思います。

いずれにしても上記の私の2つの仮説は、検証方法がありませんし、これからも、仮説でとどまることになるでしょう。

「苦痛を100%回避できる逃げ道」が欲しいだけ

それでは最後に、「苦痛は共有できない」または「私たちは、他人の苦痛を共有したくない」ことを、『認めがたい事実』として認定した上で、『「苦痛」が動かす社会の未来像』について、私からの提言という形でまとめてみたいと思います。



上記の話は、私の体験に基づいています。

例えば ーー 患者（私）の主張する苦痛に対して、処方箋を出し渋る医者がいます。私が過去に処方された経験があり、その効果があり、そしてその薬の名称まで指定しているのに、「そのクスリは効果がない」と言い張る医者がいます ーー それは1人や2人ではありませんでした。

私、本当に訳が分からないのですが、医者の中には『患者の望む通りにクスリを出したら負け』と思っている者がいるのでしょうか？　医学的な知識や理論は知りませんが、被験者に効果の認められた処方を、あれほどムキになって否定するのは ーー『患者から指示されることは、プライドが許さない』と思っているとしか思えません。

私はエンジニアですが、自分の思い込みが間違っていたことなど日常茶飯事です。特にIT関係は、間違いがすぐに分かる ーー「**動けば正解、動かなければ不正解**」 ーー という性質がありますから、自分の見解には常に謙虚になります。

医者も、疾病や傷害を治癒する技術者という意味ではエンジニアであるはずで、当然100%の正解が出せる訳ではありません。実際のところ、セカンドオピニオンが真逆の診断をしたことなど普通に結構あります（というか、私の体験では、ほとんどそうだった）。

結局、ITであれ、医療であれ、サービスを提供する側は、顧客のリクエストを最優先すべきです。しかし、医療分野においては、その姿勢が著しく欠けているのではないか思えるのです。

―― 自分のメンツにこだわるアホな医者、そして、緩和ケアを簡単に実施できない法律や制度……

本当に、本当に、本当に、ばかばかしい。「苦痛」は、私たち個人の重要で深刻な問題です。医者や法律の都合なんぞ、なんで私が考慮してやらねばならない。私の苦痛に関しては、それを一番よく知っているのは、この私です。

ただ一点、医師は、患者の生命について、常に法的リスクを負っていることは考慮してあげたいとは思います。

前述した通り、医師は矛盾した状況で判断を強いられる、割の合わない職業です。ですから、彼らのリスクを患者である私が負うように法律と制度を変えて欲しいのです。もちろん、私自身も独力で努力しなければならないこともあります。

つまるところ、私は、**「望めばいつだって100%苦痛を回避できる」「望めばいつだって自分の手で人生を終えることができる」**という保証があればいいのです。

そのような保証 ―― 逃げ道 ――さえあれば、私は安心して「希望とともに苦痛と闘う」ことができるのです。

少なくとも、**「絶望とともに苦痛と闘わされる」**というような、人生の晩節を他人に踏み躙られるような最期だけは、ご勘弁頂きたいのです。

「介護」と「IT」は、恐ろしく相性が悪い

それでは、本シリーズ第5回（最終回）の内容をまとめます。

【1】本シリーズ第1回でご紹介した「自動車に乗ったまま自宅に帰れなくなった父（故人）」の事件を振返って、**「あの時、どんなITシステムを作っておけば良かったのだろう」**という観点に立ち戻り、3つの車載システムを考案しました。

【2】従来のIoTシステムとは真逆のアプローチ「移動サーバ（Movable Server）」をラズパイで実現させて、（1）父の自動車の現在と過去の位置を表示する**「「追跡」システム」**、（2）父が迷走中であることを車の外部に表示する**「「SOS通知」システム」**、そして、（3）迷走中の父に音声で指示を与えることのできる**「「よびかけ」システム」**の3つのシステムについて、その作り方と運用方法を説明しました。

【3】前回私が自作した「尊厳死宣言書」について、現職の医師の方から頂いたレビューを公開しました。江端の「尊厳死宣言書」は、現場の担当医としての救命の使命と、江端の宣言書の内容が矛盾するものとなり、**担当医がその職務を遂行できなくなる**という指摘を受け、さらに、**私の「尊厳死」の考え方に対する現場の医師のリアルな恐怖**が表明されていました。

【4】本シリーズ第1回で掲げたテーゼ、「介護ITの究極の目的は『苦痛の定量化/見える化』であるとした上で、人体の「「痛み」検知システム」が、**人間にとって理不尽なほど非道なもの**であることと、それに対する現在の医療による対抗策について紹介しました。

【5】さらに、私たちは実際のところ「わざわざ、他人の痛みを感じたいとは思わない」という仮説を立てた上で、『「苦痛」が動かす社会の未来像』として4つの提言を行いました。その目的は、**「望めばいつだって100%苦痛を回避できる」「望めばいつだって自分の手で人生を終えることができる」**という保証である、と主張しました。

以上です。

本シリーズは、「介護IT」をテーマとするコラムですが、実際のところは、私の父（故人）と母（寝たきり）と、そして姉と一緒にさまざまな事件に遭遇しながら、ここ十数年の間、私が考え続けてきたことを、一気に吐き出すコラムとなってしまいました。

最終回のここに来て、こんなことを書くのは気が引けますが、

―― 「介護」と「IT」は、恐ろしく相性が悪い

というのは、偽らざる事実です。

これは「介護」がアナログサービスの極みに立つものである一方で、「IT」がデジタルサービスの極みに立つものであるからです（この具体例については、これまで散々紹介してきたので、もう繰り返しません）。

この連載が成立したのは、たまたま、私が「介護」と「IT」の両方に手の届く位置に立っていたたからに過ぎません。**「介護IT」は、現時点においては、その兆しすら見えない漆黒の闇の中**にあります。

この記事をメインで読んで頂いている方の多くは、「IT」に手が届きやすい場所にいる方だと思います。一方、「介護」の方は、あなたが望まなくても、「親」または「自分自身」が主体となって、勝手にやってきます。

その時に、もう一度、この連載の記事を読み直してみてください。

そして、『ああ、当時の介護サービスは、目も当てられない絶望的なサービスだったのだなぁ』と振り返ることができる未来であればうれしいのですが、残念ながら私はネガティブです。

今の私が感じている**「介護IT」に関する深い絶望感**というその一点で、私たちは、時空を超えて共感することになると思います。

選ばれし預言者『江端』なのです

後輩:「今回の前半は、いわゆる昭和の最後から平成の最初あたりを思い出させますねえ。ノスタルジーですねえ」

江端:「ノスタルジー？」

後輩:「いや、UNIXワークステーションの奪い合い、始めてPCにインストールしたFreeBSD, そしてLinuxが世界に放たれた時の興奮……あの懐かしい時代を思い起こさせてくれましたよ」

江端:「はい？」

後輩:「電源投入時のサービスの自動起動とか、デバイスとの通信とか、随分苦労しましたよねえ ―― うん、決まりましたね。今回のコラムの題目は**『"おっさんホイホイ"としてのラズパイ』**、この一択です」

後輩:「いやいや、ラズパイは、基本的に教育用のボードコンピュータという位置付けであって……」

後輩:「江端さん。バカ言ってんじゃないですよ。今回前半のようなシステム構築を、そこらのガキやビギナーが、ホイホイできると本気で思っているんですか」

江端:「それは……」

後輩:「そもそも、"教育用コンピュータ"というフレーズって、変ですよ。その方向性を前面に出したいなら、「**個人で試して、何度でぶっ壊しても、やりなおせるコンピュータ**」というべきでしょう」

江端:「まあ、SDカードのバックアップだけで、簡単に故障前に戻せる安心感は大きいな。SDカード、Amazonで600円くらいで購入できるし」

後輩:「それはさておき ―― ところで、江端さんは、この連載の第1回目で、（1）介護の問題をITで解決する方法を考える、（2）介護の苦痛を、政治社会、国家、歴史の観点まで広げて論じる、と大風呂敷広げていましたよね」

江端:「……（グッ!）」

後輩:「今回の後半では、（1）物理的な痛みに問題を絞り込み、その対象を医師と患者に限定している。しかも個人的な観

点に終始している。(2)ITは介護は親和性が悪く、今後も悪くあり続けるだろう、というネガティブな見解でオチとしている」

江端:「……(グサッ!)」

後輩:「**ショボい結論ですね。江端さんの力量って、この程度ですか**」

江端:「いや、本当に私は頑張ったんだよ。だが、苦痛の問題に多様性があり、常に変化し続けるものだから、static(静的)なものとしてしか取り扱いできなかったし……」

後輩:「……」

江端:「さらに、介護に関しては、"介護者"から、"要介護者"にフェーズチェンジと時に、パラダイムの逆転現象が発生して、一貫性のあるロジックに落し込めなかったんだ」

後輩:「つまり、普遍的な介護哲学や介護ソリューションの登場は、今後も期待できない、と?」

江端:「その時代の社会通念や技術、そしてコスト(経済状況)に応じた『場当たり対応』を続けていくことで、精一杯だと思う」

後輩:「後ろ向きの総括ではありますが、その総括を言い切れる程度には、江端さんは、がんばったのでしょう。その点は褒めて上げましょう」

江端:「……なんでお前、そんなに『上から目線』なの?」

後輩:「あ、それはそうと、江端さん、先月、原稿落しましたね?」

江端:「そうじゃない。私は原稿を落さない主義だ。先月は、EE Times Japan編集担当者の都合と、私の仕事のタスク負荷が、うまいことコラボして『Win-Winの休載』となったんだよ」

後輩:「『Win-Winの休載』……あいからわらず、美辞麗句の発明家ですねえ、江端さんは」

江端:「先月は、屋外の実証実験で、ずっと『動かないシステム』におびえて、精神安定剤を握りしめながら生きていたんだーー 動かないシステムと共に、真冬の夜の屋外の現場でシステムチューンを続ける日々は、本当に辛かったぞ」

後輩:「まあ、江端さんの日記読んでいたから知っていましたけどね」

江端:「私、もうちょっとでリタイア予定のシニアエンジニアなんだけど、なんで、今もい、ガリガリとコーディングやってんだろう? それどころか、『江端さん、ここの部分のコード書けますか』と、なんで後輩から仕事を押しつけられているんだろう?」

後輩:「それが何か?」

江端:「え?　そもそも、コーディングは、若いエンジニアの仕事だろう?」

後輩:「江端さんともあろう人が、なんという情けない考え方をしているのですか。江端さんの言っていることは『男女差別』にも匹敵する、『老若差別』ですよ」

江端:「『老若差別』?」

後輩:「『ジュニアはフィールドで実装、シニアはマネジメント』ーー そんな惚けた考え方が、このご時世で通ると思っているんですか。道路工事や建設現場を見たことありませんか?　今は『シニアこそがフロント』です」

江端:「ええーー でも、もう、体力的になぁ……」

後輩:「**そもそも、江端さんは『自分の幸せ』に全然気がついていない**」

江端:「幸せ?　誰?　私?」

後輩:「江端さんは、新しい企画を『つぶす』ことしかできず1mmも役にも立たないあの町内会のジジイたちとは全く違うんですよ。江端さんは、真の意味での『知恵の泉としての敬愛の対象である村の大長老』なんです。その自覚あります?」

江端:「『村の大長老』?」

後輩:「古代の呪術言語(C/C++)を唱え、神秘の石版(ラズパイのボード)に魔方陣を切り、賢者の石(秋月通商のモジュールキット)を使った術法増幅を行い、神の言葉(SQL)を使って神託を得ることができる、**選ばれし預言者『江端』**なのです」

江端:「……は?」

後輩:「江端さんは、本当に頼られていて、必要とされているのです。どれだけ多くのシニアが、お飾りの職位を付けられ、誰からも頼られることもなく、自己の存在を肯定し得ず、毎日を寂しく生きているか、ご存じですか?」

江端:「いや、むしろ、私は、そんな生き方ができればと、願うことすらあるが……」

後輩:「ならば、この私から申し上げることがあるとすれば、このフレーズになりますね」

『ぜいたく言ってんじゃねーぞ!　この馬鹿者!』

Profile

江端智一(えばた ともいち)

日本の大手総合電機メーカーの主任研究員。1991年に入社。「サンマとサバ」を2種類のセンサーだけで判別するという電子レンジの食品自動判別アルゴリズムの発明を皮切りに、エンジン制御からネットワーク監視、無線ネットワーク、屋内GPS、鉄道システムまで幅広い分野の研究開発に携わる。

意外な視点から繰り出される特許発明には定評が高く、特許権に関して強いこだわりを持つ。特に熾烈(しれつ)を極めた海外特許庁との戦いにおいて、審査官を交代させるまで戦い抜いて特許査定を奪取した話は、今なお伝説として「本人」が語り継いでいる。共同研究のために赴任した米国での2年間の生活では、会話の1割の単語だけを拾って残りの9割を推測し、相手の言っている内容を理解しないで会話を強行するという希少な能力を獲得し、凱旋帰国。

私生活においては、辛辣(しんらつ)な切り口で語られるエッセイをWebサイト「こぼれネット」で発表し続け、カルト的なファンから圧倒的な支持を得ている。また週末には、LANを敷設するために自宅の庭に穴を掘り、侵入検知センサーを設置し、24時間体制のホームセキュリティシステムを構築することを趣味としている。このシステムは現在も拡張を続けており、その完成形態は「本人」も知らない。

本連載の内容は、個人の意見および見解であり、所属する組織を代表したものではありません。

関連記事

誰がために「介護IT」はある?
今回から、「介護のIT化(介護IT)」をテーマにした新しい連載を始めます。人材不足が最も深刻な分野の一つでありながら、効率化に役立つ(はずの)IT化が最も進まない介護の世界。私の実体験をベースに、介護ITの“闇”に迫ってみたいと思います。

装着から歩行まで1人でできる、歩行支援ロボット
外骨格ロボットの開発と販売を手掛けるFREE Bionics Japanは「CEATEC 2019」（2019年10月15〜18日、幕張メッセ）で、歩行支援ロボット「FREE Walk（フリーウォーク）」のデモを行った。

デジタル時代の敬老精神 〜シニア活用の心構えとは
今回は「シニアの活用」についてです。やたらとずっと働きたがるシニアに働いてもらうことは、労働力の点から見ればよい施策のはずです。ただし、そこにはどうしても乗り越えなくてはならない壁が存在します。シニアの「ITリテラシー」です。

介護サービス市場を正しく理解するための“悪魔の計算”
今回は、寿命と介護について、恐らくは“世界初”となるであろう「悪魔の計算」を試みました。この計算結果を見て、あなたはどう思いますか？そして私が見いだした、「働き方改革」を行う本当の理由とは……？

「シュタインズ・ゲート」に「BEATLESS」、アニメのAIの実現性を本気で検証する
前回で最終回を迎えた「Over the AI」ですが、番外編として、私がどうしても、どうしても、書きたかったコラムをお届けすることにしました。SFやアニメに登場するAI（人工知能）の実現性です。今回は、「シュタインズ・ゲート」や「BEATLESS」に登場するAIを取り上げ、エンジニアとして、それらの実現性を本気で検証してみました。

不幸な人工知能 〜尊敬と軽蔑の狭間で揺れるニューラルネットワーク
今回のテーマは、第3次AI（人工知能）ブームを支えているといっても過言ではない花形選手、ニューラルネットワークです。ただしこのニューラルネットワーク、幾度もダイナミックな“手のひら返し”をされてきた、かわいそうなAI技術でもあるのです。