卒業論文

小型無線タグを用いた
子供の行動計測システム
に関する研究

東北大学工学部電気情報物理工学科
張山研究室

斉藤 涼太

目 次

第1章　緒言

近年, 様々な分野でICTの活用が進められ, 医療分野における高度な医療の提供, 製造業における生産性向上などに貢献している. 一方, 保育の分野では, 教育の高度化の観点でICTの導入が進んでおらず, 保育園に預けた子供の様子を知る手立てが未だ保育士の主観による報告に限られているといった問題が発生している. そのため, 子供の行動解析データを介した保育分野におけるICTの導入が求められている. 行動解析のために用いるデータにはさまざまあるが, 移動場所の傾向を解析することで, 同様の移動をした子供のデータから交友関係を, 移動場所の傾向から子供の嗜好・個性をつかめる.

以上より, 本研究では, 子供の移動場所の傾向から行動解析を行うことを目指す. 移動場所の傾向解析を行うには子供がどこにいるのか計測する必要がある. 子供の位置情報を高精度で計測することができれば, 子供の移動場所を計測できるが, こちらは設備コストの問題から決定的な技術がまだ登場していない.[1] 他方, カメラによる記録を解析する手法もあるが, こちらは記録から解析可能なデータにするのが手作業で行われており, 解析コストの問題が生じる. そこで, 本稿では, 設備コスト, 消費電力の低い小型無線タグ (TWELITE) を利用して位置計測を自動化することを目的とする.

本論文は以下に示す5章より構成される.

第1章は緒言であり, 本研究の背景, 目的及び概要について述べたものである.

第2章では本研究で使用したモデルとして, 無線タグおよびMQTTプロトコルを用いた計測システムについてまとめた.

第3章では本研究で使用した計測機器であるTWELITEタグの計測精度および使用プロトコルであるMQTTプロトコルの通信の確実性やレイテンシをまとめ

た. 得られたデータの処理方法についてもここで述べた.

第 4 章では計測システムを用いて実験を行い, その評価を行った.

第 5 章は結言である.

以上, 本論文の企図するところを概説した.

第2章　無線タグおよびMQTTプロトコルを用いた計測システム

2.1　はじめに

この章では今回使用するシステムの構成についてまとめた.

2.2　無線タグ TWELITE

2.2.1　TWELITE の諸元

無線タグ TWELITE の通信は親機として MONOSTICK, 子機として TWELITE CUE という機器を使用して行われる. 各機器の諸元は表 2.1 の通りである.

親機・子機間の通信は 2.4GHz IEEE 802.15.4 に準拠して行われ, 変調方式は O-QPSK, DSSS である.[2] 親機 MONOSTICK は多数の子機からの情報を受信でき,PC にはシリアルポートとして認識される. そのため, 多くの OS で利用可能である. 子機には加速度センサーが搭載されていて, その測定値を送信することができる. 親機を接続した PC・マイコン等はその情報を確認できるほか, 子機から送信された電波がどのくらいの強度で届いたかを後述の LQ 値という指標を用いて表示する.[3] 子機から親機への送信頻度の設定は加速度サンプリング頻度と加速度送信サンプル数を変化させることにより行え, サンプリング頻度が 25Hz, 送信サンプル数が 16 の時の送信頻度は 25/16(≒1.56)Hz である. 本稿では, 電波強度が親機と子機との距離に応じて変化することに着目して実験を行う.

表 2.1: TWELITE 諸元

機器	W/D/H(mm)
TWELITE CUE(子機)基板部	25/25/10(最厚)
TWELITE CUE(子機)ケース	30/30/15
MONOSTICK(親機)	50/22/8(USB端子部除く)

2.2.2 LQ 値

LQ(Link Quality) 値とはモノワイヤレス社製品における通信品質を表す値で, LQ 値 150 以上は送信機の近傍にいること,50 未満は通信品質が著しく悪いこと (-80dBm) を示す. 以下の式で dBm と変換することができる.[4]

$$dBm = (7 * LQ - 1970)/20 \tag{2.1}$$

本研究では, 電波強度を評価するための指標としてこの LQ 値を用いる.

2.3 MQTT プロトコル

MQTT(MQ Telemetry Transport) プロトコルは, 小型デバイス間でのメッセージ交換用に設計された TCP/IP(ポート 1883) を使用するプロトコルである. メッセージを確実に送信することを目的としているため, プロトコルヘッダが小さいことが特徴で, 即時性, 軽量さが長所として挙げられる.[5] 通信環境は, 図 2.1 のように, メッセージを送信する側のパブリッシャと受信する側のサブスクライバに加え, 中継者としてのブローカーから構成される. 通信はパブリッシャがブローカーにメッセージを送信し, サブスクライバがブローカーから必要なメッセージを受信するという方式により行われる. 必要なメッセージのみを受信する方法として,TOPIC という識別子が用いられている. パブリッシャはメッセージに TOPIC を付加し, サブスクライバは受信する TOPIC を設定する. ブローカーはパブリッシャから受信したデータについて, 付加された TOPIC を受信する設定をしているサブスクライバのみにメッセージを送るという仕組みによりサブスクライバが不

必要なメッセージまで受信することを防いでいる. 本研究では,MONOSTICK を接続しているマイコン (Raspberry Pi) をパブリッシャ, マイコンと LAN を通して接続された PC をサブスクライバとする. 片方向, かつ集約的な通信を行う本研究において, ブローカーはサブスクライバの PC と兼ねて差し支えないため, サブスクライバ PC にブローカー (Eclipse Mosquitto^{TM}) を導入して通信を行う.

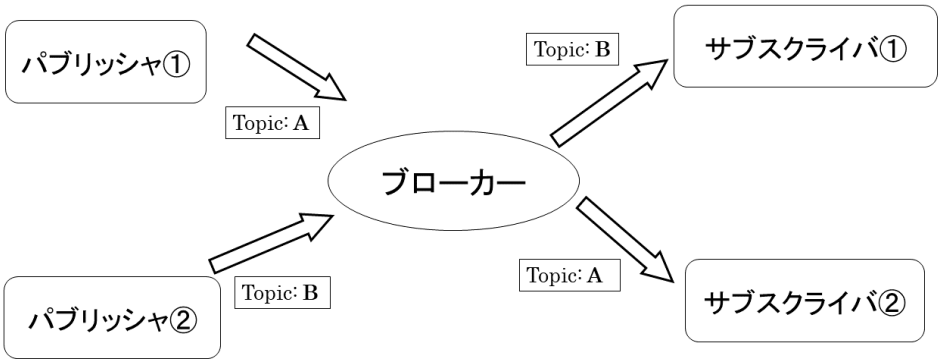

図 2.1: MQTT 構成の概要

2.4 計測システムの構成

本研究では, これまでに述べた機器・プロトコルを用いて, プレーパークや保育園において図2.2のようなシステムを構成する. 子供につけたTWELITE無線タグから発信されたデータは, 各遊具・遊び場にあるマイコンに接続されたMONOSTICK及びMQTTプロトコルを介して, 子供が”いつ”,”どの遊具・遊び場を使っていたか”がわかるデータとして集積される. そのうえで, 集積されたデータから子供の動きを解析し, 個性や交友関係について定量的な分析を行うことを可能にし, より高度な教育につなげることを目指す.

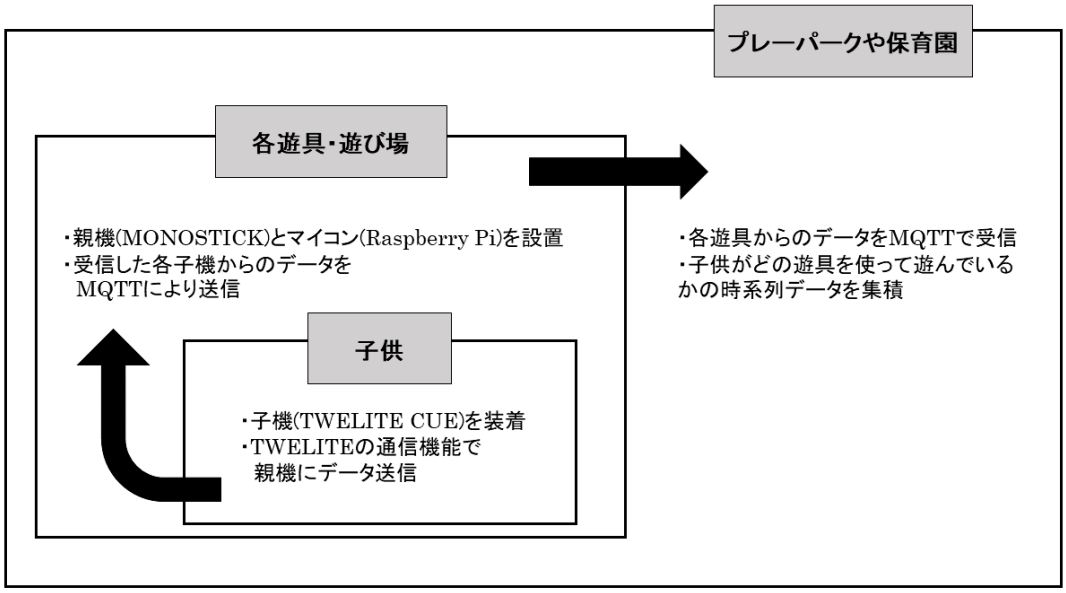

図 2.2: 計測システムの構成

第3章　システムの性能評価

3.1　はじめに

本章では2章で示したシステムの機器・プロトコルについて, 実環境での使用を目標とした性能評価を行った. また, 測定に伴い発生する欠測値の補間方法ならびにノイズの除去方法についても考察・実証のうえまとめた.

3.2　TWELITE 無線タグの性能評価

ここでは, システムのうち,TWELITE 無線タグについて, 実際の通信距離と電波強度の関係についてまとめた.

3.2.1　通信距離と電波強度の関係の定量的な測定

3.2.1.1　測定の条件

本測定では MONOSTICK を PC に接続し,MONOSTICK と TWELITE CUE を使用して行った. 周囲に何もない屋内環境（体育館）において,MONOSTICK と TWELITE CUE の距離を 1m,2m,3m,4m,5m,10m,15m,20m,25m に固定し 30 秒間測定した. 測定は 2 つの TWELITE CUE タグを使用して行い, 発信頻度は 1.56Hz(加速度サンプリング頻度 25Hz, 送信サンプル数 16) とした. 測定時間 30 秒間に送られてくる電波強度（LQ 値）を平均し, 測定結果とした.

3.2.1.2　測定結果

通信距離と電波強度の測定結果は図 3.1 のようになった. 電波強度は通信距離 5m までは一次関数的な減少を見せたが, その後は多少の上下こそあるもののほぼ

横ばいの推移を見せた.

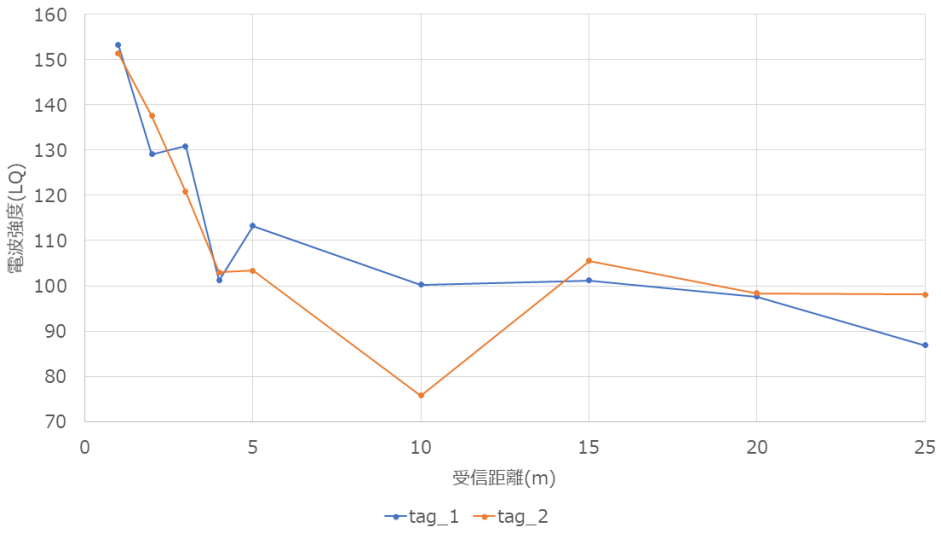

図 3.1: 通信距離と電波強度の測定結果

3.2.1.3　測定結果からの考察

以上の結果より, 特異的な結果は通信距離おおむね 5m 以内で現れる LQ 値 120 以上という値であることが分かった. よって, 以降の研究では通信距離が 5m 以内となる計測条件を設定して実験を行うこととする.

3.2.2　機器の評価を踏まえた TWELITE 無線システムの利用方法

以上の評価結果を踏まえ考えられる TWELITE 無線システムの利用方法は以下の通りである. 図 3.1 からわかるように,TWELITE 無線の電波強度は, 通信距離 5m 程度までは単調に減少し, それ以降は横ばいの推移をとる. このことを用いて,MONOSTICK と TWELITE CUE が 5m 以下の距離にあるような使い方をする.

3.2.2.1　網羅的配置

まず,MONOSTICK を取り付けたマイコンの配置について, 一つの MONOSTICK が受け持つ範囲は, 半径 5m 以下の円形の範囲とする. そのうえで, 図 3.2 のようにカバーしたい範囲を網羅するように配置することで, 子供がいずれかの MONOSTICK と 5m 以内に近接していることになるので, 高い電波強度を示す MONOSTICK が必ず現れる. それにより子供の位置を概ね特定することができる. この方法では, 値による閾値を示さずとも他との比較により位置を推定できるので簡単である.

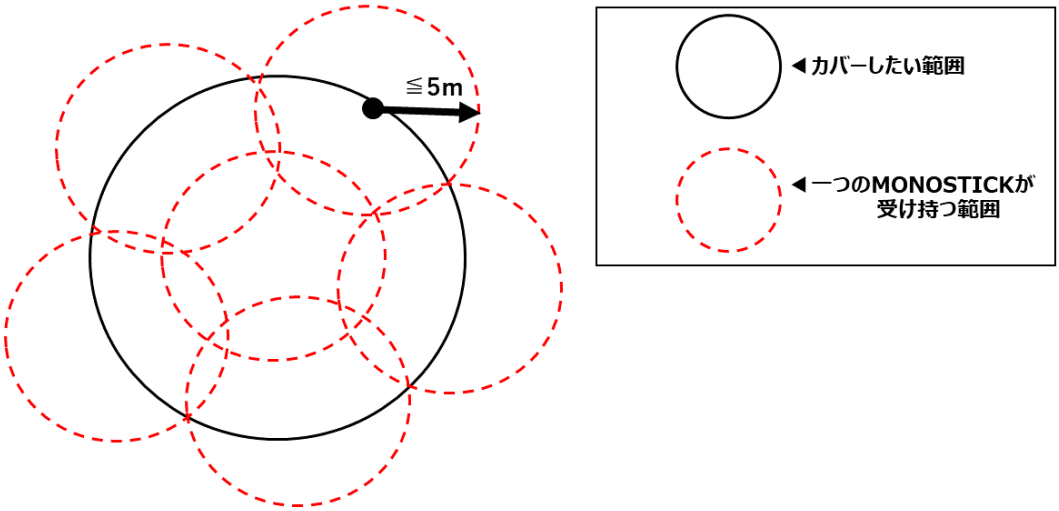

図 3.2: MONOSTICK の網羅的配置

3.2.2.2 チェックポイント的配置

MONOSTICK が受け持つ範囲は同様に半径 5m 以下とする．ここで,MONOSTICK が受け持つ範囲の中に, 特定の入口・出口ゲートや, 滑り台の降り口など, 特定の遊びをしたときに通るポイントを含めることにより, 子供が何をしていたかを概ねつかむことができる．使用する MONOSTICK, マイコンの量を減らせるので, リソースに限りがあるときに有用である．

3.3　MQTT プロトコルの性能評価

つづいて,MONOSTICK を取り付けた複数のマイコンからデータを集約するための MQTT プロトコルについて評価を行う. MQTT プロトコルは軽量かつ即時性に優れたプロトコルであるが, 実際の使用環境におけるデータの送受信に耐えるか確かめるべくデータ受信のレイテンシや送受信の成功率についてまとめた.

3.3.1　測定環境

MQTT のパブリッシャとしては MONOSTICK を取り付けたマイコン (Raspberry Pi) を使用し, ブローカーおよびサブスクライバ PC として Windows PC1 台を使用した. なお,MQTT はポート 1883 を使用した TCP/IP 通信によって行われる. LAN による接続はすべて Wi-Fi を用いた無線通信により行うこととし, それを実現するための Wi-Fi ルーター 1 台も用いた. Wi-Fi は屋外利用が想定されることおよび安定した通信を目指すことから 2.4GHz 帯を使用した. 実際に使用した MQTT の構成詳細を図 3.3 に示した.

なお, 本測定で使用したタグからのデータ取得及び MQTT データ転送のプログラムについて, ソースコードは最後に付録としてまとめてある.

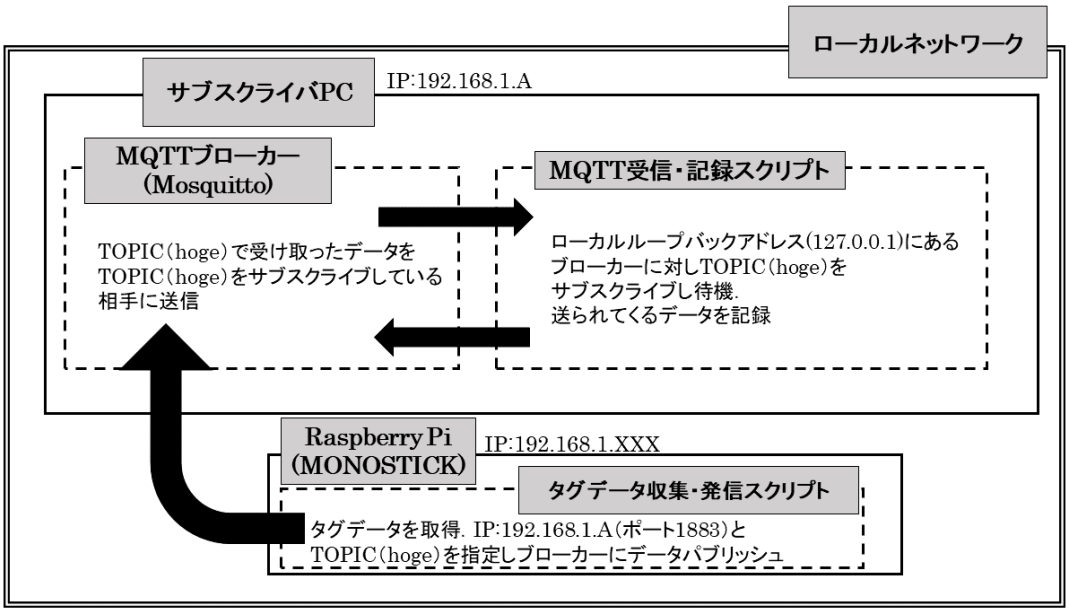

図 3.3: MQTT によるやり取りの詳細

3.3.2 MQTT の受信成功率測定

受信成功率の測定は MONOSTICK から取得した TWELITE CUE タグからの受信データ (34 Byte) を MQTT プロトコルによって送信する際, パブリッシャ側のマイコンでも同時にログをとることにより行った. MQTT を介してサブスクライバ側が受け取ったデータとパブリッシャ側に残るデータを照らし合わせ, データの欠測のうち MQTT プロトコルに起因する欠測の数を計測し受信成功率を求めた. 試行回数 6879 回のうち MQTT プロトコルに起因するデータの欠測は 0 件だったため,MQTT の受信成功率は 100%であることが確かめられた.

3.3.2.1　時刻の同期に関する問題の MQTT による解決

本研究では, 複数のマイコンを用いてデータの取得を行うが, 複数のマイコンの時刻がずれているとマイコンごとに TWELITE CUE からのデータ受信時刻が異なることになり, 正確な受信時刻を得ることができない. それを防ぐためには, マイコンの時間を同期する必要があった. しかし,MQTT が即時性に優れたプロトコルであることを利用して受信時刻をサブスクライバ側の PC の時刻に一括することができれば, マイコンの時間を同期する必要がなくなる. 本稿ではそのための条件を設定して実験を行った.

3.3.3　MQTT のレイテンシ測定

本研究で送信されているデータの送信頻度は 1.5Hz 程度である. よって, レイテンシを 0.5 秒以内に抑えることができれば前後のデータが入れ替わることなくマイコンの時刻同期をせずとも受信時間に正確さが出る. 本稿では, 様々な条件のもとで MQTT プロトコルのレイテンシを測定し, 実際に 0.5 秒以内に抑えられているか確かめた.

3.3.3.1　単一パブリッシャからの受信で測定したレイテンシ

受信成功率と同様にして MQTT のレイテンシ測定を行った. レイテンシ測定はパブリッシャのマイコンとサブスクライバの PC 双方に MONOSTICK を取り付けることによって行った. TWELITE CUE から発信されたデータは, パブリッシャのマイコンに取り付けられた MONOSTICK および MQTT を介してサブスクライバの PC に届く. サブスクライバ PC では,MQTT を介してデータが届いた時刻を記録した. 一方, サブスクライバの PC でも同じデータを MONOSTICK のみを介して取得し, その時間も記録した. その二つの受信時間を比較すれば MQTT の遅延時間を求めることができる. レイテンシの計測結果は以下の表 3.1 のようになった. 最大値こそ 4 秒台と大きいものの, 平均, 標準偏差ともに小さく抑えられていることからわかる通り, ほとんどの場合では非常に小さい値になっていることが特徴である.

表 3.1: 単一パブリッシャからの受信で測定したレイテンシ

試行回数	1720
平均	0.079(秒)
標準偏差	0.304(秒)
最大値	4.537(秒)

3.3.3.2 複数パブリッシャからの受信で最速をとったレイテンシ

本研究で想定されているシステムでは TWELITE CUE からのデータは複数のマイコン・MQTT パブリッシャを通して届く. そのため, 複数のパブリッシャから送られた同一時刻の情報のうち, 一番最初にサブスクライバに届いた時刻を代表として記録すればレイテンシを抑えることができる. 4 台のマイコンを用意し, それぞれに MONOSTICK を取り付け, 同一のサブスクライバに向けて MQTT で情報を送信した. 単一パブリッシャでの実験と同様に,MQTT を通してサブスクライバに届いた時刻を記録し, 同一データに関する時刻のうち一番早いものを比較のための代表時刻とした. サブスクライバの PC にも MONOSTICK を取り付け TWELITE CUE からのデータを記録し, 受信時間を比較しレイテンシとした. レイテンシの計測結果は以下の表 3.2 のようになった. 最大レイテンシは 0.5 秒以下に抑えられているため, 本研究ではマイコンの時刻の同期は行わず, 記録する時刻は MQTT 受信後のサブスクライバの時刻とする.

表 3.2: 複数パブリッシャからの受信で測定したレイテンシ

試行回数	4181
最大レイテンシ	0.149(秒)（＜0.5）

3.4　データの処理方法の決定

ここでは,TWELITE タグ及び MQTT プロトコルを用いて集約されたデータに発生するノイズの除去や欠測値の補間について述べる.

3.4.1　状態空間モデルを用いたフィルタの適用

一般に, あるシステムを運用する際に, 実際にはシステムの状態を直接確認できず, 環境や観測機器によるシステムノイズや観測ノイズが生じる. このノイズはある分散をもった正規分布に従うと仮定してフィルタリングすることができ, 計量や信号処理などの分野で活用されている. 本研究で得られる測定結果も, 電波干渉等に影響され得られるデータには振動が生じているため, 状態空間モデルを用いたフィルタを使用しノイズを除去する. フィルタリングは以下の式 3.1,3.2 に基づいて行われる.

$$x_n = Fx_{n-1} + Gv_n \tag{3.1}$$

$$y_n = Hx_n + w_n \tag{3.2}$$

ここで, F,G,H は適切な次元を持つ行列で, v_n,w_n は平均 0 でそれぞれ τ^2,σ^2 を分散とする正規分布に従うとする.[6] 状態空間モデルでは, 時間更新 (Predict) により x の一歩先の状態を予測し, 実際の観測値 y によって推定値の更新 (Correct) を行う.[7] 本稿では, これを利用し, 適切な τ^2 の値を用いることで x の状態を推定する. 図 3.4 は測定されたデータセットに対し $\tau^2 = 0.1$ としてフィルタリングした結果である. 得られた測定値の特徴を残しつつノイズの除去が行われている.

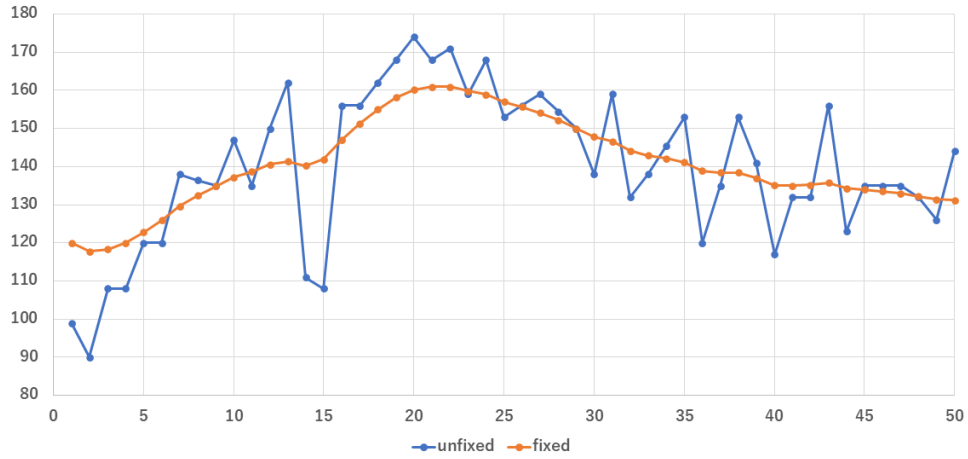

図 3.4: 状態空間モデルを用いたフィルタリング結果

3.4.2　欠測値の補間方法

前述した状態空間フィルタを適用するためにはデータを時系列データとする必要がある．しかし,TWELITE CUE・MONOSTICK を用いた実験では, データの発信頻度にかかわらずデータの欠測が生じる．複数の MONOSTICK を使用して同一のデータを複数経路で受信している場合, 受信しているうちの一部が欠測により不足していることがある．この場合, 正しい比較ができなくなるため適切な方法で欠測を補間する必要がある．ここでは, 適切に補間するための方法について検討する．

3.4.2.1　NN 補間

NN(Nearest Nearby) 補間は, 一番単純な補間方法で, 欠測が発生した部分の前後のデータのうち, 該当部に近いほうのデータを用いてそのまま補間する．

3.4.2.2　線形補間

線形 (linear) 補間は, 欠測が発生した部分の前後のデータ間で, 値が線形な変化をしたと仮定し, その値で補間する方法である.

3.4.2.3　スプライン補間

(三次) スプライン (spline) 補間は,N 個の測定点（データ）が与えられたデータセットを N-1 個の区間に分割し, その区間ごとに適切な三次曲線を求めてそれをもとに補間する方法である. 区間ごとの三次曲線は以下の条件を満たす.

- 各区間を補間する曲線はその両側の点を通る.
- N 個の各点において両側の区間の 1 次導関数は等しい.
- N 個の各点において両側の区間の 2 次導関数は等しい.

三次曲線の未知数 4 つに対して, 式が 4 つできるのでこの条件により各区間ごとに補間する曲線の式が求まることを利用し補間するのがスプライン補間である.

3.4.3　補間方法の比較

先に述べた三つの補間方法の比較のため, 実際に計測できたデータセットからランダムに一部を削除し, 欠測のある状態を作り出した. そのうえで実際の計測データ (unfixed) とそれぞれの補間方法で補間されるデータを示したのが図 3.5 である. まず, スプライン補間は, 各測定点の両側で 1 次導関数,2 次導関数が等しいという条件に影響され, データの上下が激しい区間で欠測した場合にオーバーシュートが発生するため, 補間方法として適切ではないことが分かった. よって, 線形補間と NN 補間に絞って比較した. この二つの補間方法による差は大きくないが, 実際の子供の移動により近いと考えられるのは線形補間であるため, 本稿では線形補間により補間を行うこととする.

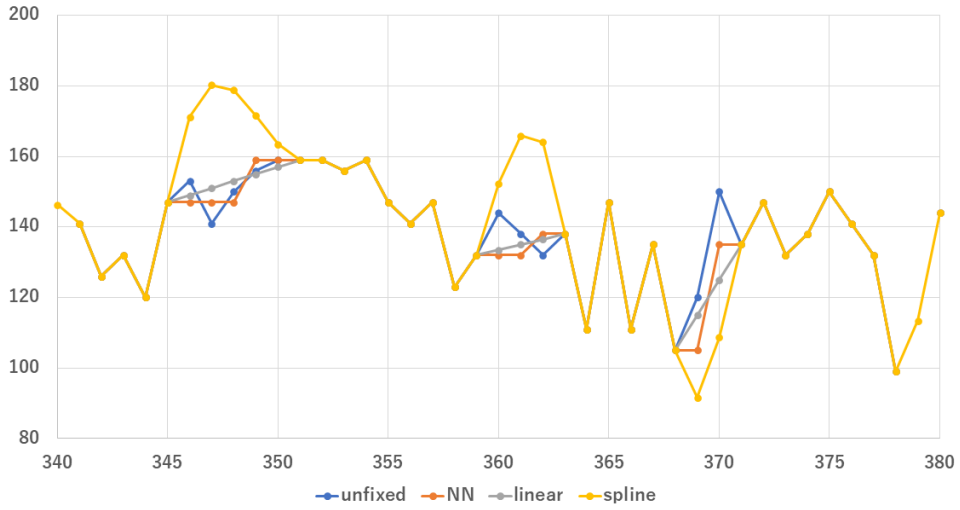
200
180
160
140
120
100
80
340
345
350
355
360
365
370
375
380
unfixed
NN
linear
spline

図 3.5: 補間方法の比較

第4章　システムを用いた実験, 評価

4.1　まえがき

ここでは, 観測者である TWELITE CUE タグの位置の推定を目指し, 第 3 章において評価したシステムを用いて簡単な実験を実施した. その実験の結果と評価についてまとめる.

4.2　実験の概要

4.2.1　測定機器

先述の TWELITE CUE,MONOSTICK のほかに, サブスクライバ PC として MQTT ブローカー Mosquitto を導入済みの Windows PC1 台, MQTT プロトコル (TCP/IP) を実現するための Wi-Fi ルーター 1 台を用いて行った.

4.2.2　実験環境

図 4.1 に実験環境の写真を示した. 室内で 180cm 四方の机を囲むような環境になっているが, 机を用いたのは MONOSTICK について, 床から離して設置するよう推奨されているためである. MONOSTICK を取り付けたマイコン (Raspberry Pi) は机の四隅に設置されている. 同様に床から離すよう推奨されている TWELITE CUE タグは, 自由に動くプラスチック製のワゴンに置かれ, そこから発信されたデータは 4 か所の MONOSTICK・MQTT プロトコルを介してサブスクライバ PC に集約されるようになっている.

図 4.1: 実験環境の写真

4.2.3 実験方法

1. 机の四隅に置かれた MONOSTICK・Raspberry Pi からサブスクライバ PC に MQTT でデータが送られるよう設定する.

2. 1.56Hz(加速度サンプリング頻度 25Hz, 送信サンプル数 16) でデータを送信するよう設定した TWELITE CUE タグを置いたプラスチックワゴンを机を回るように自由に移動させる. 移動の様子は同時にビデオ撮影を行って記録しておく.

3. サブスクライバ PC で, 集約されたデータに線形補間による補間と状態空間モデルによるフィルタリングを行ったのち比較を行い, 観測者から一番近い MONOSTICK が 4 台のうちどれであるか求める.

4. ビデオ撮影によって得られた情報と比較を行い, 今回のシステムによる推定の正解率を調べる.

4.3　実験結果

実験中 192 回のデータ送信があったうち, システムによって得られた推定の正解は 174 回であり, 正解率は 90.6%であった. 不正解となった約 10%のデータは約 5 秒程度の連続した不正解であった.

4.4　実験の評価

システムによる推定の正解率が高く, システムの近距離の判別性能は高いといえる. 不正解となった時間について, その時間はどの MONOSTICK にも極度に接近していない状態であった. そのため, 不正解の一因としては, 最も近いタグとそれ以外のタグとの距離比があまりないために電波強度に差が生まれず, システムが推定を誤ったと考えられる.

第5章　結言

本稿では, 子供の移動場所の傾向を解析して行動解析を行うという目標のもと, 小型無線タグ TWELITE を用いた位置情報推定に向け, 機器の性能評価と転送プロトコルの提案, 補間やフィルタリングを含めたシステムの構成と性能評価を行った. まず, 小型無線タグの性能評価においては, 通信距離と電波強度の関係について, ある一定の距離 (5m 程度まで) は通信距離の増加に伴い電波強度が減少するがそれ以降は電波強度が横ばいになるという結果が得られた. 続いて,MQTT プロトコルについて,0.5 秒以下の低レイテンシで確実性の高いプロトコルであることが確かめられ, 本稿における転送プロトコルとしては十分な性能を示した. 小規模な屋内で実装したシステムの性能評価では,90.6%という高い推定正解率を示し, 子供の移動場所の傾向を示すためには十分な結果と考えられる. 推定不正解となった部分についても, 距離比の差が小さいことに起因するため, 今後大きな規模に拡大することにより改善することも考えられる. 本稿ではデータ処理について, 状態空間モデルを用いたフィルタリングおよび線形補間を用いたが, フィルタや補間方法には今回検討したよりも多くの方法・ソースコードがあるため, その中から最適なものを検討していきたい. また, 今回は小規模な実験にとどまったが, 今後さらに実験環境を広げた場合の実験も行っていきたい. 本研究の最終目標はプレーパーク, 保育園等での子供の位置情報を推定し, その情報を集積・分析することである. 実際の測定環境はきれいな図形のような形をしているとは限らず, システムの性能評価を踏まえたカスタマイズが必要となるので, それも併せて取り組んでいきたい. そのうえで, 集積されたデータの解析を含めた解析をして評価可能なシステムとして提案したい.

参考文献

[1] 株式会社 NTT データ, 谷沢幹也. ”屋内測位のニーズと技術紹介”, https://inforium.nttdata.com/trend_keyword/264.html, 2017-07.

[2] モノワイヤレス株式会社 ”TWELITE CUE データシート”, https://twelite.gitbook.io/mw-psd-cue/

[3] 大澤文孝 ”TWELITEではじめる「センサー」電子工作” 工学社 2019-06

[4] モノワイヤレス株式会社 ”TWELITE CUE 製品情報”, https://mono-wireless.com/jp/products/twelite-cue/index.html

[5] IBM ”MQTT プロトコル”, https://www.ibm.com/docs/ja/ibm-mq/8.0?topic=ssfksj-8-0-0-com-ibm-mq-pro-doc-q002870--htm

[6] The Comprehensive R Archive Network ”Package TSSS”, https://cran.r-project.org/web/packages/TSSS/TSSS.pdf

[7] MathWorks ”カルマンフィルタ”, https://jp.mathworks.com/discovery/kalman-filter.html

[8] モノワイヤレス株式会社 ”パルスクリプト”, https://mono-wireless.com/jp/products/TWE-APPS/App_pal/palscript.html

謝辞

本研究を進めるにあたり, 多くの皆様にたくさんの協力, 助言をいただきました.

主指導教員である張山昌論教授には, 日々の討論などでデータの取り方, まとめ方から研究に対する姿勢に至るまで鋭い指摘や多くの助言をいただきました. また, 研究室において安心して, 集中して研究をできるよう気にかけていただきました. 心より感謝申し上げます.

副指導教員であるハシタムトゥマラウィッデヤスーリヤ准教授には, 研究活動だけでなく, 日々の研究室生活において充実した研究生活を送れるように多くの支援, 助言をしていただきました. 心より感謝申し上げます.

また, 小柴満美子准教授および小山高専の小林康浩先生にも, 実際のプレーパークでの経験や機器の使用経験からのありがたい助言, サンプルプログラムの提供など, 様々な面でサポートいただきました. 心より感謝申し上げます.

最後に, 研究会や日々の議論におきまして親睦を深めともに研鑽しあった同期をはじめ, 多くのご助言を頂きました張山研究室の皆様に感謝申し上げます.

令和 4 年 3 月 24 日

付 録A　付録

付録として，MQTTパブリッシャ側のプログラム（TWELITE CUEタグ）からのデータ取得プログラム (A.1)，MQTTサブスクライバ側のプログラム (A.2) のソースコードを記載した．実行環境はPython3.10.1である．なお，パブリッシャ側のプログラムはモノワイヤレス株式会社のパルスクリプトを改変したものである．[8]

ソースコード A.1: パブリッシャ側プログラム

```
1  #!/usr/bin/env python
2  # coding: UTF-8
3
4  ###############################################################
5  # Copyright (C) 2017 Mono Wireless Inc. All Rights Reserved. #
6  # Released under MW-SLA-*J,*E (MONO WIRELESS SOFTWARE LICENSE #
7  # AGREEMENT). #
8  ###############################################################
9
10 # ライブラリのインポート
11 import sys
12 import csv
13 import os
14 import serial
15 import paramiko
16 import time
17 import datetime
18 from optparse import *
19
20 from time import sleep
21 #mqtt用のライブラリのインポート (paho-mqttパッケージが必要)
22 import paho.mqtt.client as mqtt
23
24 # WONO WIRELESSのシリアル電文パーサなどのAPIのインポート (TWELITE製造元･モノワイヤレス社のパルスクリプト参照)
25 sys.path.append('./MNLib/')
26 from apppal import AppPAL
27
28 # ここより下はグローバル変数の宣言
29 # コマンドラインオプションで使用する変数
30 options = None
31 args = None
32
33 # 各種フラグ
34 bEnableLog = False
35 bEnableErrMsg = False
36
37 # プログラムバージョン
38 Ver = "1.1.0"
39
40 # mqtt 接続------------------------------------------------------------
41 sys.stderr.write("*** 開始 ***\n")
42 host = '999.999.99.9' #ブローカーのipアドレス(ipv4)を入れる
```

```
43 port = 1883 #mqttではこのポートを使う
44 topic = 'TWELITE' #サブスクライバプログラムと同一のtopicにすることで送受信に成功する
45
46 client = mqtt.Client(protocol=mqtt.MQTTv311)
47
48 #mqttブローカーに接続,keepalive設定は設定秒数なにもこちらからの発信がなかったとしたらping確認をして,
       その応答がなければ通信が切断されたとみなして良いという設定
49 client.connect(host, port=port, keepalive=60)
50 #上記のkeepalive設定により送られるping確認に応答する設定
51 client.loop_start()
52
53 header =['時間', '論理ID', 'タグID', '中継器ID', '送信番号', '電波強度', '電源電圧']
54 #送信データはこのように並ぶ(辞書データ)
55
56
57 # ----------------------------------------------------------------
58
59
60 #CSVを開く(送信側でのログ)
61
62 shortlog_name = str(datetime.datetime.now().strftime('%Y-%m-%d-%H-%M')) + '.csv'
63
64 print(shortlog_name)
65
66 #データを取得するメイン部分。newline=''は改行をなくすためのオプション
67
68 def writeX(note):
69     with open(shortlog_name,'a',encoding='shift_jis',newline='') as f:
70         csvout = csv.writer(f)
71         csvout.writerow(note)
72
73 #ここから先はパルスクリプト(TWELITE製造元のプログラム)を改変
74
75 def ParseArgs():
76     global options, args
77
78     parser = OptionParser()
79     parser.add_option('-t', '--target', type='string', help='target for connection', dest='t
       arget', default=None)
80     parser.add_option('-b', '--baud', dest='baud', type='int', help='baud rate for serial co
       nnection.', metavar='BAUD', default=115200)
81     parser.add_option('-s', '--serialmode', dest='format', type='string', help='serial data
       format type. (Ascii or Binary)', default='Ascii')
82     parser.add_option('-l', '--log', dest='log', action='store_true', help='output log.', de
       fault=False)
83     parser.add_option('-e', '--errormessage', dest='err', action='store_true', help='output
       error message.', default=False)
84     (options, args) = parser.parse_args()
85
86
87 if __name__ == '__main__':
88     print("*** MONOWIRELESS App_PAL_Viewer " + Ver + " ***")
89
90     writeX(header)
91
92     ParseArgs()
93
94     bEnableLog = options.log
95     bEnableErrMsg = options.err
96     try:
97         PAL = AppPAL(port=options.target, baud=options.baud, tout=0.05, sformat=options.form
           at, err=bEnableErrMsg)
98     except:
99         print("Cannot open \"AppPAL\" class...")
100        exit(1)
101
102    while True:
103        try:
104            # データがあるかどうかの確認
105            if PAL.ReadSensorData():
106                # あったら辞書を取得する
107                Data = PAL.GetDataDict()
```

```
108                 if Data['RouterSID'] == '80000000':
109                     RSID = 'No Relay'
110                 else:
111                     RSID = Data['RouterSID']
112                 print(Data)
113                 print(Data['ArriveTime'].strftime('%Y-%m-%d %H:%M:%S.%f')[:-3], end = ",")
114                 print(Data['LogicalID'], end = ",")
115                 print(Data['EndDeviceSID'], end = ",")
116                 print(RSID, end = ",")
117                 print(Data['SequenceNumber'], end = ",")
118                 print(Data['LQI'], end = ",")
119                 print(Data['Power'], end = "\n")
120
121                 # なにか処理を記述する場合はこの下に書く
122
123                 #辞書型で取られたシリアルポート(MONOSTICK)からのデータから必要なものを抜粋する
124                 datas_TAG = [str(Data['ArriveTime'].strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]),\
125                 Data['LogicalID'],Data['EndDeviceSID'],RSID,Data['SequenceNumber'],Data['LQ
                        I'],Data['Power']]
126
127                 #本稿で説明している方法では時間はブローカー,サブスクライバ側のものを使用するので
128                 #上記のデータからさらに取得時間のデータを除く
129                 send_datas_TAG = [\
130                        #str(Data['ArriveTime'].strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]),\
131                 Data['LogicalID'],Data['EndDeviceSID'],RSID,Data['SequenceNumber'],Data['LQ
                        I'],Data['Power']]
132
133                 writeX(datas_TAG) #送信側ログには時間データも必要なのでdatas_TAGを書き込む
134
135                 #パブリッシュ用データはsend_datas_TAG,こちらもカンマ区切りテキストにしてパブリッシュ
136                 client.publish(topic, ",".join(map(str, send_datas_TAG)))
137
138
139                 # ここまでに処理を書く
140
141                 # ログを出力するオプションが有効だったらログを出力する。
142                 if bEnableLog == True:
143                     PAL.OutputCSV() # CSVでログをとる
144             # Ctrl+C でこのスクリプトを抜ける
145         except KeyboardInterrupt:
146             break
147     del PAL
148
149
150     print("*** Exit App_PAL Viewer ***")
151 client.disconnect() #接続を終了する
```

ソースコード A.2: サブスクライバ側のプログラム

```
# -*- coding: utf-8 -*-
import paho.mqtt.client as mqtt
import datetime
import random
import json
import csv

#for MQTT broker
MQTT_HOST = '127.0.0.1'
MQTT_PORT = 1883
KEEP_ALIVE = 60
TOPIC = 'TWELITE'


"""
接続を試みたときに実行
def on_connect(client, userdata, flags, respons_code):

* client
Clientクラスのインスタンス

* userdata
任意のタイプのデータで新たなClientクラスののインスタンスを作成するときに>設定することができる

* flags
応答フラグが含まれる辞書
クリーンセッションを 0に設定しているユーザに有効。
セッションがまだ存在しているかどうかを判定する。
クリーンセッションが 0のときは以前に接続したユーザに再接続する。

0 : セッションが存在していない
1 : セッションが存在している

* respons_code
レスポンスコードは接続が成功しているかどうかを示す。
0: 接続成功
1: 接続失敗 - incorrect protocol version
2: 接続失敗 - invalid client identifier
3: 接続失敗 - server unavailable
4: 接続失敗 - bad username or password
5: 接続失敗 - not authorised
"""
mqttlog_name = 'mqtt' + str(datetime.datetime.now().strftime('%Y-%m-%d-%H-%M')) + '(' + st
    r(random.randint(100, 999)) + ')' + '.csv'

header =['時間', '論理ID', 'タグID', '中継器ID', '送信番号', '電波強度', '電源電圧']

#ヘッダを定義し書き込む
with open(mqttlog_name,'a',encoding='shift_jis',newline='') as fw:
    csvout = csv.writer(fw)
    csvout.writerow(header)

def on_connect(client, userdata, flags, respons_code):
    print('status {0}'.format(respons_code))
    client.subscribe(client.topic) #サブスクライブTOPICを指定

"""
def on_message(client, userdata, message):
topicを受信したときに実行する
"""
def on_message(client, userdata, message):
    nakami = message.payload
    message_json = nakami.decode('utf-8') #受信データはバイト列なのでそれを文字列に変換する
    print (message_json)
    print (type(message_json))
    #print(message.topic + ' ' + str(message.payload))
    #message_json=str(message.payload)
    #message_json=my_removeprefix(message_json, "b'")
    #message_json=my_removesuffix(message_json, "'")
    #print(message_json.decode('utf-8'))
```

```
70      with open(mqttlog_name, 'a', encoding='shift-jis',newline='') as f:
71          wdata = str(datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]) + ',' + m
                essage_json
72          f.write(wdata + '\n') #すでにコンマ区切りされたデータなので単にwriteするだけ+改行
73
74
75  #受信データはプレフィックス,サフィックスデータを含むので除く
76
77  def my_removeprefix(s, prefix):
78      if s.startswith(prefix):
79          return s[len(prefix):]
80      else:
81          return s
82
83  def my_removesuffix(s, suffix):
84      return s[:-len(suffix)] if s.endswith(suffix) else s
85
86
87  if __name__ == '__main__':
88
89      while 1:
90          try:
91
92              client = mqtt.Client(protocol=mqtt.MQTTv311)
93              client.topic = TOPIC
94
95              client.on_connect = on_connect
96              client.on_message = on_message
97
98              client.connect(MQTT_HOST, port=MQTT_PORT, keepalive=KEEP_ALIVE)
99
100             # ループ(接続を維持する)
101             client.loop_forever()
102         except KeyboardInterrupt:
103             print('end')
104             break
```