三鷹市立図書館・調布市立第一小学校
スマート都市農業プロジェクト
報告書

2024 年 2 月 22 日

国立大学法人電気通信大学

目次

1 概要・目的

これまでの農業は土地を利用した農作物の生産を目的としていが、今般のコロナ禍において家庭菜園や市民農園の人気が高まり、楽しみを目的とした都市農業の需要が高まっている。しかし都市の農地は減る一方で、その利用は限られた範囲でしかない。それに対して本研究室では、都市のビルの屋上やベランダ等の空きスペースを活用し、土を使わずに本格的な果菜類を育てることが可能な水耕栽培装置の研究を行っている。令和元年度は本研究事業において調布市立第一小学校の校舎屋上に試作の水耕栽培装置 3 台を設置し、トマトとイチゴの栽培を行った。また昨年度は三鷹市立図書館本館の児童書コーナーから見える庭に改良版の装置を設置し、本格運用に向けたトマトと花の試験栽培を行った。

本年度はコロナで 3 年間休止していた小学校施設での栽培を再開して食育・環境教育としての活用を進めると共に、図書館というオープンな場で多くの人たちに水耕栽培による本格的な果菜類の栽培を体験してもらい、IoT 技術を導入した新しい都市農業を広めていくことを目的とする。さらに昨年度、収穫した野菜等を用いた加工品やメニューの試作を行っており、本年度も 6 次産業化をさらに進める。

2 申請団体のプロフィール

電気通信大学は、1918（大正 7）年に無線通信技術者の養成機関として創設された社団法人電信協会管理無線電信講習所をその起源とする。その後、1949（昭和 24）年に国立学校設置法施行により新制大学として電気通信大学を開学し、1952（昭和 27）年には、現在の調布校舎を開校した。

2004（平成 16）年には、国立大学法人法の施行に伴い「国立大学法人電気通信大学」として新たに発足し、2018 年に創立 100 周年を迎えた。

3 期間

2023（令和 5）年 6 月 1 日～2024（令和 6）年 2 月 17 日

4 背景

都市農業は、消費地の近くでの生産・供給の向上、地産地消と 6 次産業化による地域経済の活性化、緑の空間の創出による都市景観・環境の改善、農業体験や農業プロジェクトを通じた食育・環境教育とコミュニティの形成など多くのメリットを有し、持続的な振興を図るための施策が進められている。しかし、都市における農地は宅地化により減少を続け、一旦宅地化された土地を農地に戻すことは困難である。そのため、これまでの都市農業は農地の保護と活用が議論の中心であり、新たな農地の創出という動きには至っていない。商業施設屋上の緑地化も一部では行われているものの、土は非常に重いため建物の耐荷重のほか、防水・防根工事、土の飛散防止など様々な問題がある。

そこで我々は、土を使わずに本格的な果菜類の栽培が可能な水耕栽培システムを開発し、図 4.1~4.2 のように屋上や住宅のベランダなどの都市の空きスペースを活用した栽培の研究を進めてきた。本水耕栽培システムは風による土の飛散や防水や防根対策が不要で、軽量なためどこにでも設置可能である。また頭上に枝を誘引するため、その下をイベントスペースとしても活用できる。収穫した新鮮な野菜や果物は、近隣の飲食店でのメニューや地域特産品として利用可能である。さらにビルの屋上の観光農園化やレンタルファーム化など、“都市部”での農業には高い収益性が期待される。

また農家の高齢化と後継者不足が大きな問題となっているが、都市の子供たちは動植物に触れる機会が極めて少なく、農業を身近に感じられないこともその大きな要因である。子供たちに栽培・収穫の楽しさを伝える農体験は後継者問題の解決だけでなく、コンピュータゲームのように一つの正解が存在しない「自然の摂理」を通じ、考える力を養う上でも大変有意義である。

そこで令和元年度の「民学産公」協働研究事業において、調布市立第一小学校の校舎屋上に水穀栽培施設を設置し、子供達への食育や環境教育に供することを目的に令和元年 10月から試験栽培を開始した。その後の

図 4.1 大学屋上の水耕栽培施設(左)とバーベキューイベント(右)

図 4.2 収穫イベント(左)とベランダでの栽培(中・右)

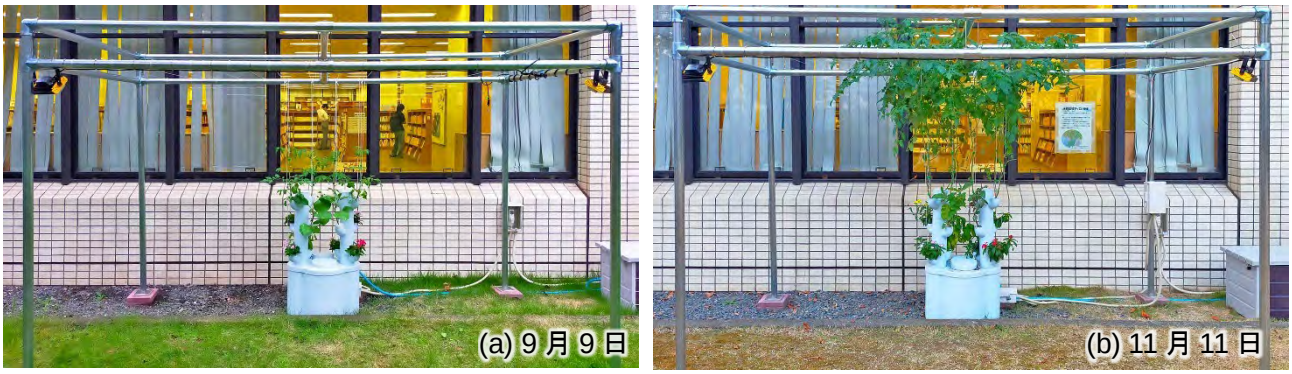


図 4.3 三鷹市立図書館に設置した水耕栽培装置

コロナ禍により中断していたが、本年 4 月より再開することとなった。また誰でも参加できるオープンスペースでの運用を目的に、昨年度の事業において図 4.3 のように三鷹市立図書館へ装置を設置して 9 月からミニトマトと草花の栽培を開始した。しかし開始時期が遅かったため青い実は沢山生ったものの、赤く熟れたものは数えるほどしかなかった。

そこで本年度は 4 月からトマト、きうり、ナス、パプリカ、いちご等、栽培品種を増やして本格栽培を行うと共に、図書館では野菜栽培の書籍コーナー設置や講演会を行い、小学校では苗定植から栽培まで授業を行う。その他にも、本事業を中心とする水耕栽培による都市農業普及に向けた講演活動を進める。

栽培のフィードバックを受けながら、装置の製品化を進めるため、図書館、小学校、大学に設置している装置の改良と評価を行う。特に電源のない場所でも設置可能として利用範囲を広げるため、ソーラパネル駆動のための開発も行う。小学校は施設のフレームや装置の筐体を再利用して、内部機構を全て新しいシステムに入れ替える。

三鷹での 6 次産業化を目指し、社会福祉法人「むうぷ」の協力の元で収穫した野菜の加工品の開発販売やランチメニューやお弁当への利用を進める。また昨年度再開され、今年は飲食物の提供が見込まれる図書館フェスタへ参加する。

5 栽培と授業・講演

本年度は、1月末から図 5.1 のように室内でトマト、パプリカ、カボチャ等の育苗を開始した。図 5.2 のように 3月末に大学屋上施設へ、4月には三鷹市立図書館でボランティア市民の方と、調布市立第一小学校屋上では 4 年生の 4 クラスの児童約 120 名と定植を行った。

図 5.1 室内での育苗

図 5.2 苗の定植

4月8日に本プロジェクトに関連して三鷹市市民参加でまちづくり協議会（マチコエ）での講演会を、図 5.3 のようにコロナ禍のため人数制限を行った会場とネットのハイブリッドで開催した。三鷹市立図書館の定植には、講演会に参加した地元のボランティアの方にも協力いただいた。また図 5.4 に示したように、三鷹市市民参加でまちづくり協議会（マチコエ）から三鷹市への政策提案書の中でも、本プロジェクトの水耕栽培が紹介されている。

図書館でも 6月3日に図 5.5 の講演会を実施し、参加者に水耕栽培装置の見学をしていただき、また 10月28日に開催された図 5.6 の図書館フェスタでも見学いただいた。


図 5.3　三鷹市市民参加でまちづくり協議会（マチコエ）での講演会

図 5.4　三鷹市市民参加でまちづくり協議会（マチコエ）から三鷹市への政策提言


図 5.5　三鷹市立図書館での講演と見学会

図 5.6　図書館フェスタ

令和 4 年度は季節外れの 9 月からの栽培だったのに対して、本年度は 4 月から開始した。図 5.7 は 4-6 月の栽培の様子で、前年度の試験栽培から日当たりは良いものと思われていたが、建物や木の陰の影響からか影になる時間が多くトマトは徒長して枝葉が日弱かった。一方カボチャの成長は良好であった。

図 5.7　4~6 月の図書館の栽培の様子

7 月上旬までは図 5.8 のように全体として比較的順調であったが、中旬以降は記録的な猛暑が始まり苗が枯れ始めた。8 月には枝葉を整理するとともに、比較的日当たりがよい東側の苗を西側に這わせて全体のバランスを

整えた。さらに緑を増やすためにサツマイモの苗を 8 月 22 日に、新しいトマトの苗を 8 月 29 日に定植した。9 月に入っても暑さは衰えることがなく、4 月に植えた苗は次第に弱っていった。新しく植えたトマトとサツマイモは順調であったため、10 月にカボチャを全て撤去して空いたところに唐辛子を定植した。

図 5.8　7~8 月の図書館の栽培の様子

図 5.9　8~10 月の図書館の栽培の様子

暑さが和らいだ 10 月になるとトマトはぐんぐん育ち、11 月にはたくさんの花や実をつけるようになった。成長が芳しくなかったパプリカも多くの赤い実を生らせるようになった。サツマイモもたくさんの葉を茂らせていたが、トマトの邪魔となるためつるをカットした。その後、12 月まで成長を続けたが、気温が低いためトマトの実は青いままで

終わってしまった。

図 5.10　11 月の図書館の栽培の様子

図 5.11 は三鷹市の 6~9 月の今年の最高および最低気温を棒グラフで、また平年の平均の最高と最低気温を線グラフで示している。6 月中旬までは最高・最低気温ともに、線の上下にあり平年並であることがわかる。しかしその後は 9月までで最高気温が平均を下回ったのはわずか 2 回、最低気温に至っては一日もなく平年の最高気温を上回る日もあったことがわかる。水耕栽培装置のセンサデータによると水温が夜間も 30 度以下にならず、このため根が弱っていたことも考えられる。

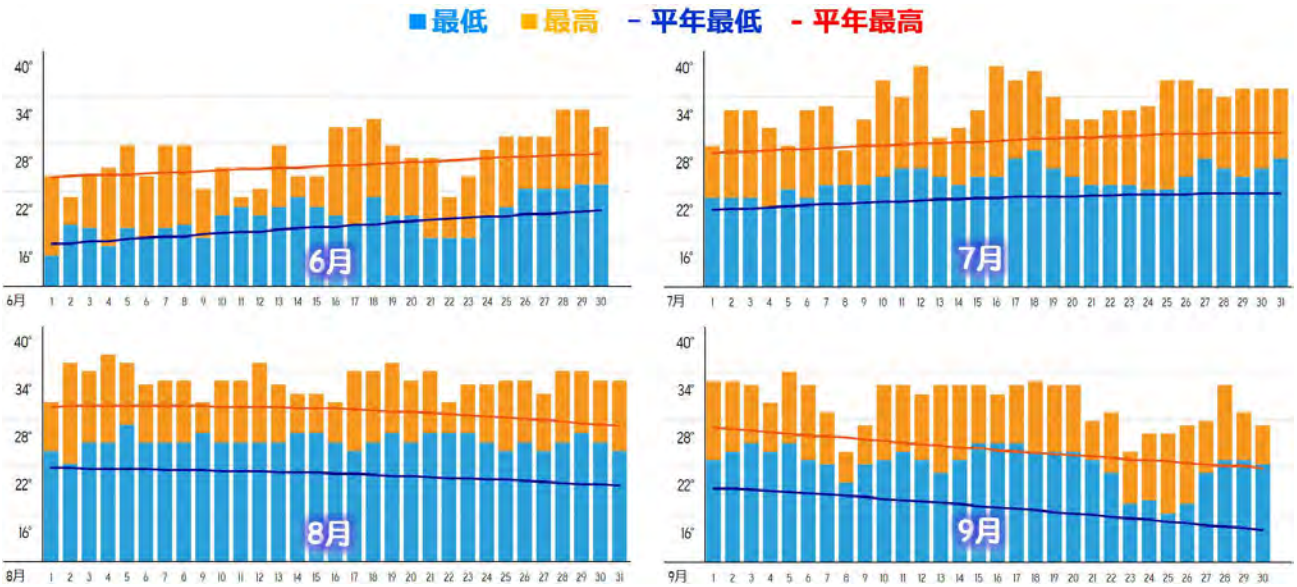


図 5.11　三鷹市の 6-9 月の最高および最低気温

調布市立第一小学校屋上での 3 台の装置による 4~6 月の栽培の様子を図 4.12 に示す。順調に育っていたものの 6月6日に季節外れの台風が襲来し、1 台の装置の上部のパイプが強風でずれてポンプで汲み上げた水が装置の外に流れ出るようになってしまった。さらに注水ホースも外れてしまったためタンクの水がなくなり苗が全て枯れてしまった。センサデータが水位低下を検出したことから警告メッセージが送信されており、ポンプの電流センサから負荷が低くなっていることもわかり、水がなくなっていることはわかっていた。しかしながら運悪く台風は金曜日の夜に関東に上陸し、土日は小学校に入ることができなかった。そのため週明けに対応した際には、中央の 1 台が図 5.12 右下のように完全に枯れてしまっていた。

そこで図 5.13 のようにバックアップ用に大学で育てていたトマトの脇芽と、ホームセンターで購入した花の苗を6月8日に植え直した。その後の好天にも恵まれ、一ヶ月後には実を生らすまでに成長した。

図 5.12　4~6 月の小学校の栽培の様子

図 5.13　植え替えた苗の生育の様子

小学校でも 5月に、屋上のトマトの脇芽を児童と切って、催場を行っている。7月初旬からは図 5.14 のように児童とトマトやきうりを収穫した。なお屋上での活動の前には、図 5.15 のように関連した内容の授業を約 20 分行っている。これらの授業活動は 4 回×4 クラスの計 16 回実施した。


図 5.14　児童との野菜の収穫

屋上は日当たりが良いため

成長がよく、大学でも 7 月には図 5.16 のように沢山のトマトやカボチャが実った。コロナも収束したため、7月16日のオープンキャンパスでは、図 5.17 のように午前中に親子を招待してトマトの収穫や採蜜イベントを行い、午後には収穫した野菜等を用いてバーベキューを開催した。

枝葉は頭上に誘引する栽培方式のため、その下の日陰のスペースはこのような様々なイベントに活用できる。図 5.18 はサーマルカメラで撮影した大学の施設である。屋上での水耕栽培は、ヒートアイランド対策にも大きな効果が期待できることがわかる。

図 5.15　小学校での授業の様子

図 5.16　7 月の大学屋上での栽培の様子

図 5.17　オープンキャンパスでの収穫・BBQ・採蜜会

小学校と大学とも 7月下旬以降は猛暑の影響で、根が弱ったり葉が焼けるなど状態が悪化した。図 5.19 のように遮光ネットを掛けて日差しを和らげたが、状態はあまり改善しなかった。小学校では 9月にカボチャ、きうり、ゴーヤの収穫を行い、トマトは 12月まで栽培を続けたがほとんど収穫できず、また大学施設は 10 月に終了となった。

12月2日には、東京農工大学小金井キャンパスにて、同校、東京外語大学、電気通信大学合同のオープンセミナーを開催し、本事業の取り組みについても講演を行った。

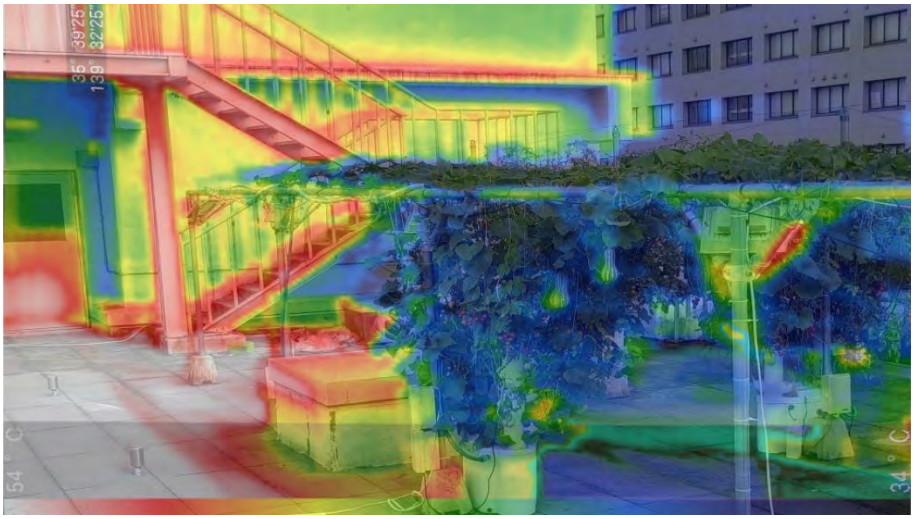


図 5.18　サーマルカメラで撮影した大学屋上の水耕栽培施設

図 5.19　小学校屋上施設への遮光ネット設置

図 5.20　東京農工大学でのオープンセミナー

6 水耕栽培システムの改良

6.1 追肥機構

図 6.1.1 は小型水耕栽培装置の追肥機構で、センサモジュールから Type B USB コネクタに 5V を供給してエアポンプを制御している。センサが EC 値の低下を検知すると、ボトルにポンプに空気を送り込んで肥料の原液を押し出して水耕栽培装置のタンクへ注入する。水耕栽培では通常、窒素・リン酸・カリウムを含む液と、石灰等のカルシウムを含む液を混合して使用する。この 2 つの原液を事前に混合すると、肥料が固まって沈殿してしまうので、別ボトルに入れて 1:1 の比率で注入できるように濃度を調整しておく。この 2 つの原液を 2 つのエアポンで注入してみたところ、ポンプの性能ばらつきやチューブのセッティング等の影響により、同量にすることが困難であった。そのため、水耕栽培の定番の肥料である OAT ハウス 1 号と 2 号の混合をあきらめ、本来は水耕栽培用でないが窒素・リン酸・カリウム・カルシウムを含有した OAT アグリオ OK-F-1、もしくは OK-F-3 を溶かした 1 液の原液で栽培を行った。生産量を比較評価してはいないものの、2 液の場合と大差はないようであった。

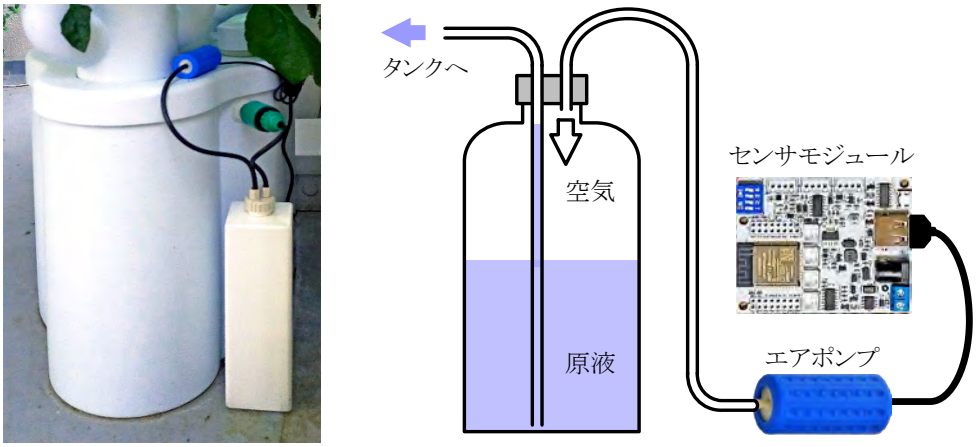


図 6.1.1 小型水耕栽培装置の追肥機構

図 6.1.2 施設園芸用水耕栽培装置(左)とシンガポールの施設での葉物栽培(右)

また佐藤研究室はシンガポールで集合住宅の駐車場屋上で、葉物野菜の栽培をメインに図 6.1.2 の装置を稼働している。14 個のポットを有したパイプを 6 本、または 12 本立てたものを 1 セットとして、1 つの大きなタンクを共有して液肥を循環させている。なお、シンガポールの一番大きな施設では、12 本×150 セット = 1,800 本のパイプ、総計 25,200 本の苗を育てることができる。この装置は生産性を重視しているため 2 液混合での栽培を行って

いる。エアポンプで押し出す方式は、注入口よりも原液ボトルを低くしないと液が全部タンクに入ってしまい、また注入口が高すぎると空気で押し上げることができない。シンガポールでは人手で液肥の追加を行っているが、数日に一回のため液肥濃度が大きく上下して生育に影響を与えるという課題がある。市販の施設園芸用の追肥装置は数十万円とかなり高価であり、また遠隔モニタや操作といった機能も有していない。

そこで、洗濯機や食洗器の洗剤の投入等にも使われる図 6.1.3 左の 蠕(ぜん)動ポンプを用い、一つのモータで 2 液を吸い上げるシステムを作製した。蠕動ポンプの動作原理は図 6.1.3 右のように、弾性のあるチューブをローラで押しつぶしながら回転させて内部の液体を押し出すのと同時に、押しつぶされたチューブが元に戻ることで次の液体が吸入される。小型水耕栽培装置のエアポンプによる追肥のように設置の制約がなく、原液タンクの場所が高くても低くてもモータの回転数に応じて安定した液肥の注入が可能となる。

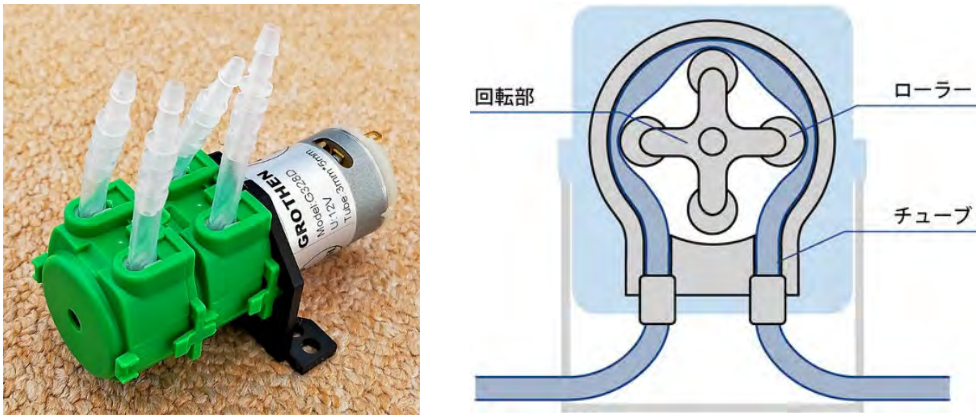
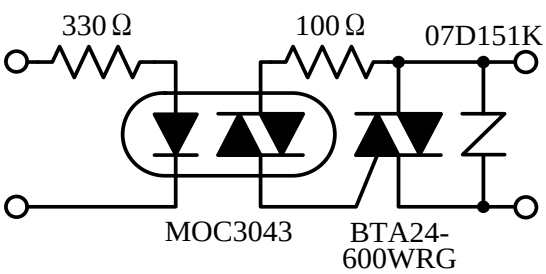


図 6.1.3 蠕動ポンプ

このポンプの電源は DC 12V 仕様であるが、5V~との表記があったため直接 5V を入力してみたが、ほとんど回転しなかった。そこで 12V 電源をリレーでポンプに供給することとし、図 6.1.4 のフォトトライアックによるソリッドステートリレーを用いた図 6.1.7 左の装置を試作した。トライアックの入力は、センサモジュールの ESP32 の I/O から 0V(オフ)/3.3V(オン)の信号で制御した。なお小型水耕栽培装置のエアポンプを制御する type-B USB コネクタの 5V 出力回路は、図 6.1.5 のように出力が常時 5V 電源につながっているため、これを制御信号として使用することはできない。このソリッドステートリレーでポンプを制御したところ、オフからオンへの動作には問題なかったが、なぜかリレーの制御信号を 0V に戻しても 12V が流れ続けてポンプが止まらなかった。もし不具合でポンプが回り続けて原液の注入が止まらないと、液肥が濃くなりすぎて植物が枯れてしまう恐れがある。この不具合が修正できても、別の原因で同様のトラブルが発生する心配がある。

図 6.1.4 ソリッドステートリレー



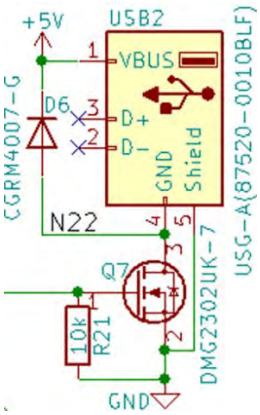


図 6.1.5 センサモジュールの
エアポンプ用 5V 出力回路

そこでスイッチを確実にメカニカル動作させる図 6.1.6 の電磁リレーJQC3F-03VDC を用いることにした。このリレーを用いた装置を図 6.1.7 右に示す。図 6.1.6 左の回路図で、モータは COM(Common)と NO(Normally Open)の端子に接続している。制御信号は IN に接続するが、その電圧が 3.3V のときモータはオフ、0V のときオンとなる点に注意が必要である。なお、制御信号の線が外れたときも IN は 0V となるが、フォトカプラ 817C は電流駆動なので、この時にリレーは動作しない。

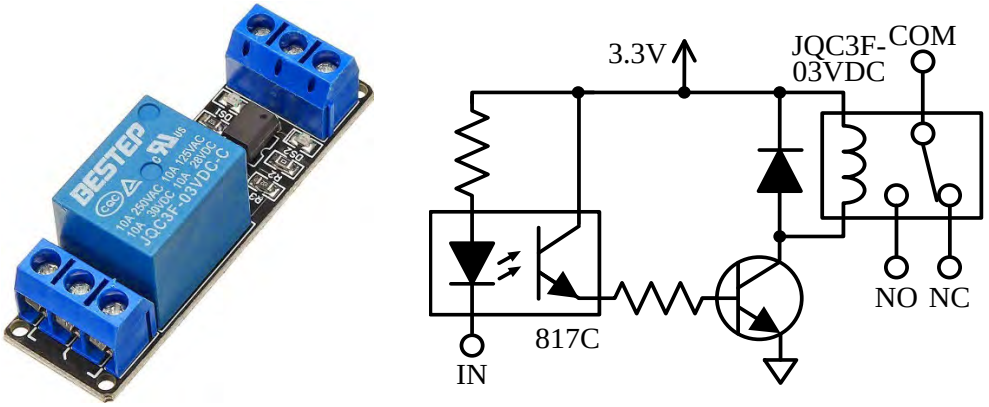


図 6.1.6　電磁リレーJQC3F-03VDC

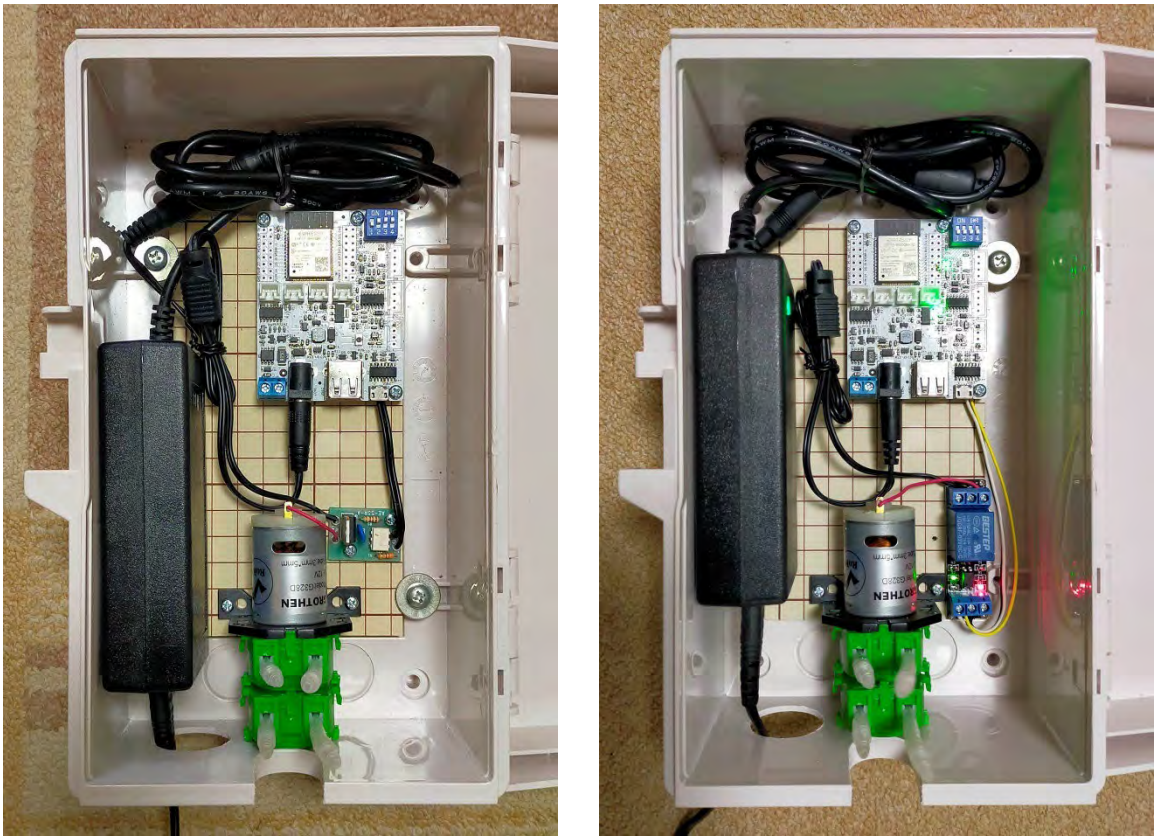


図 6.1.7　ソリッドステートリレー(左)と電磁リレー(右)による追肥ジュール

6.2　注水機構

図 6.1.1 の小型水耕栽培装置の追肥機構は全ての機構が一ヶ所にコンパクトにおさまっているが、施設園芸では広いハウス内に電源、水道、タンク、ポンプ等が要所に置かれており、一つのセンサモジュールで制御することは困難である。そこで複数のモジュールのデータをサーバで連動させる機構の開発を進めた。協力いただいている世田谷区の農家「そらまめ農園」のハウス内では、図 6.1.2 のタイプの水耕栽培装置を導入してトマトの栽培試験を行っている。液肥は深さ 2.5m のタンクに溜め、ポンプで汲み上げて循環させている。同ハウス内で以前に水を大量に消費する植物を栽培していた際は、タンク上部に設置したボールタップで満水に制御していた。これに対して現在栽培しているトマトでは、水位を40~50cmにして数日に一度肥料と水を人手で追加している。そのため肥料の濃度が安定せず、また水を切らせてしまう心配もある。

図 6.2.1　HC-SR04

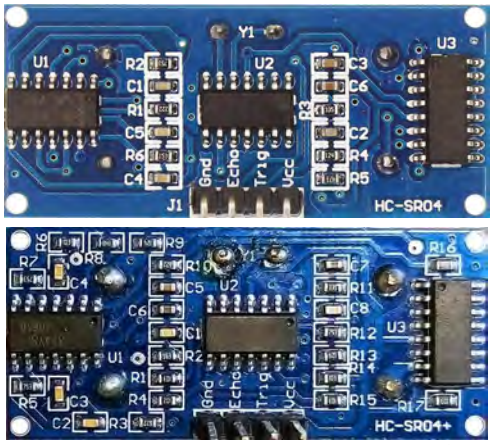


図 6.2.2　HC-SR04 と SR04+の基板裏面

そこで今回、タンク内に設置した水位センサのデータで、タンク外に設置した電磁弁を制御して注水する機構を作成した。なお、追肥機構は前節の追肥モジュールを利用する。

図 6.2.1 は超音波距離センサの定番 HC-SR04 で、500 円程度で入手可能である。一方の筒から照射した超音波が対象物に当たって戻って来る時間をもう一方の筒で測定して距離を算出する。測定範囲は 2~400cm で、最大精度は 3mm であるが、2 つのセンサで同時に測定したところ最大誤差は 3cm 程度であった。なお、対象物の測定範囲は中心から±15 度である。駆動電源は 5V であるが、HC-SR04+は 3.3V もサポートしている。HC-SR04 と HC-SR04+の超音波送受信側の基板には両者とも“HC-SR04”と記されているが、裏面は図 6.2.2 のように”HC-SR04”や“HC-SR04+”と記載されている。なお、HC-SR04 の基板は、この写真以外にもいくつかバリエーションが存在するが、HC-SR04+にバリエーションがあるかどうかは未確認である。

図 6.2.3 に ESP-32mini で 2 つの距離センサ制御するブレッドボード実装(左)、配線部分のみ(中央)、そして CAD ツール(Fritzing)による実態配線図(右)を示す。ESP-32mini は小型化のためにピンヘッダのスルーホールが両サイドに並んでいるが、これをブレッドボードに立てると 2 列のピン間で信号がショートしてしまうので、図 6.2.4 のように必要な信号の内側にだけにピンを立てた。またタンク内に設置するためのケースを、図 6.2.5 のように 3D プリンタで作製した。

リスト 6.2.1 に HC-SR04 を制御する ESP32 のスケッチを示す。距離センサでは注水のための水位を測定するが、誤動作によって注水されずに植物が枯れてしまったり、あるいは逆に水が止まらずにタンクから溢れてしまったりということを避けるため、2 つのセンサで同時に測定し（wlevel1, wlevel2）その差（wlevel）が 5cm 未満のときに平均値を MQTT サーバに送信するようにしている。また差が 5cm 以上の場合は 999 を返す。測定間隔の INTERVAL は 10 分（10*60000ms）としているが、トピック TOPIC_DCHECK（デバイス名/dcheck）を受信すると、そのときの値を直ちに Publish する。callback()は DCHECK が来た時に瞬時に実行される関数で、MQTT 接続が確立されたときには 1 回だけ mqtt.subscribe(TOPIC_DCHECK)を実行し、その後は mqtt.loop()を繰り返し実行

する。ここで mqtt.subscribe(TOPIC_DCHECK)を毎回実行してしまうと、新たな TOPIC_DCHECK が来ていなくても、同じものを何度も受信することになってしまうので注意が必要である。MQTT 接続が切れて再接続しなおしたのかどうかの判断には、変数 MQTTconnected を用いている。

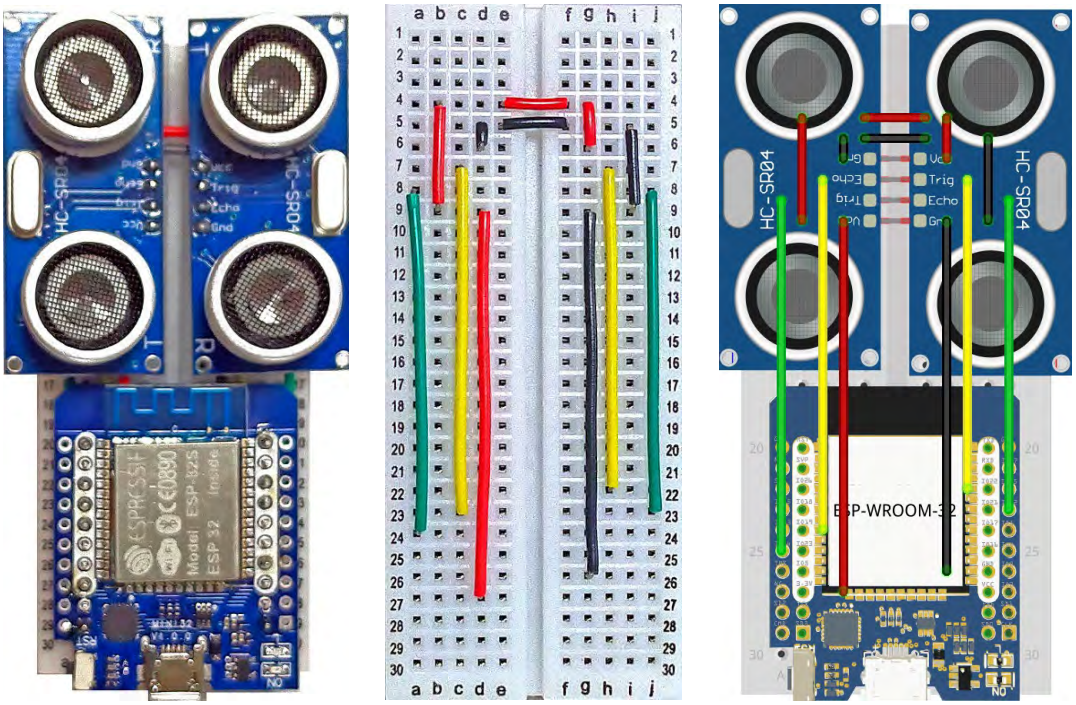


図 6.2.3　ESP-32mini と距離センサ HC-SR04 を実装したブレッドボード

図 6.2.4　ESP-32mini に立てたピンヘッダ

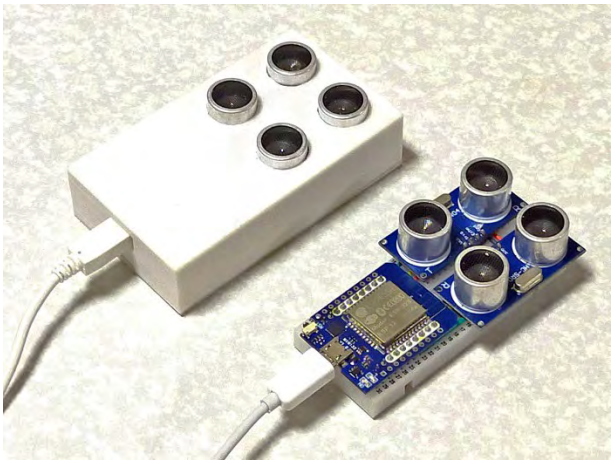


図 6.2.5　距離センサのケース

リスト 6.2.1　2 つの HC-S04 を制御するスケッチ

```
#include <PubSubClient.h>
#include "myMQTT.h"

#define TRIG1 19
#define ECHO1 18
#define TRIG2 22
#define ECHO2 21
#define MAX_DELAY 20000
#define SOUND_SPEED 0.034
#define APmode false                // true:APmode    false:STAmode
#define INTERVAL (10 * 60000)
```

```
char TOPIC_WLEVEL[128];
char TOPIC_DCHECK[128];
bool DCHECK;

/***********callback function work when message is arrived***********/
void callback(char* topic, byte* payload, unsigned int length) {
   payload[length] = '¥0';                  // The last published string data is atteched at the last of the subscribed string
data. So, insert EOL.
   Serial.printf("Msg arrived [%s]¥n", topic);
   if (String(topic) == String(TOPIC_DCHECK))
      DCHECK = true;
}

void setup() {
   Serial.begin(115200);
   pinMode(TRIG1, OUTPUT);
   pinMode(ECHO1, INPUT);
   pinMode(TRIG2, OUTPUT);
   pinMode(ECHO2, INPUT);
   digitalWrite(TRIG1, LOW);
   digitalWrite(TRIG2, LOW);

   if (!setup_MQTT(APmode))
      ESP.restart();

   while (APmode)
      AP_start();

   create_topic("wlevel", TOPIC_WLEVEL);
   create_topic("dcheck", TOPIC_DCHECK);
   DCHECK = false;
}

unsigned long last_time = 0;
bool MQTTconnected = false;

void loop() {
   char    str_buf[10];
   float wlevel, wlevel1, wlevel2;

   if (WiFiMQTT_reconnect()) {
      if (!MQTTconnected) {
         mqtt.subscribe(TOPIC_DCHECK);
         MQTTconnected = true;
      }
      mqtt.loop();

      if ((millis() - last_time > INTERVAL) || DCHECK) {
         for (int i = 0; i < 10; i++){
            digitalWrite(TRIG1, HIGH);
            delayMicroseconds(10);
            digitalWrite(TRIG1, LOW);
            float wlevel1 = pulseIn(ECHO1, HIGH, MAX_DELAY) * SOUND_SPEED/2;
            delay(2000);

            digitalWrite(TRIG2, HIGH);
```

```
            delayMicroseconds(10);
            digitalWrite(TRIG2, LOW);
            float wlevel2 = pulseIn(ECHO2, HIGH, MAX_DELAY) * SOUND_SPEED/2;

            Serial.printf("wlevel1: %5.1f¥n",      wlevel1);
            Serial.printf("wlevel2: %5.1f¥n¥n", wlevel2);

            if (abs(wlevel1 - wlevel2) < 5){
               wlevel = (wlevel1 + wlevel2) / 2;
               break;
            }
            else {
               wlevel = 999;
               Serial.printf("wlevel measurement error¥n");
            }
         }

         sprintf(str_buf, "%5.1f", wlevel);
         mqtt.publish(TOPIC_WLEVEL, str_buf);
         DCHECK = false;
         last_time = millis();
      }
   }
   else if (MQTTconnected)
      MQTTconnected = false;

   delay(100);
}
```

超音波距離センサで測定した水位に応じて注水を行うのに、図 6.2.6 のソレノイドバルブ（電磁弁）を用いる。これは通常の水道にも広く用いられている G/12 径のホースジョイントが接続でき、DC12V で制御される。二つの端子に電圧をかけるとバルブがオン（水が出る）またはオフ（水が止まる）する。電圧の向きは関係ないが、電圧をかけるとオフからオンになるもの（ノーマリーオフ）と、逆にオンからオフになるもの（ノーマリーオン）がある。注水装置の電源が落ちたときに水が出続けるとタンクから溢れてしまうので、ここではノーマリーオフのものを使用する。内部の構造は非常に簡単で、コイルの中心に前後に動く金属の棒が入れられており、バネで片方に押し付けられている。電源が入ると電磁石によってそれが押し付けられていた方向と逆に移動し、ゴムの弁をオンまたはオフする仕組みとなっている。

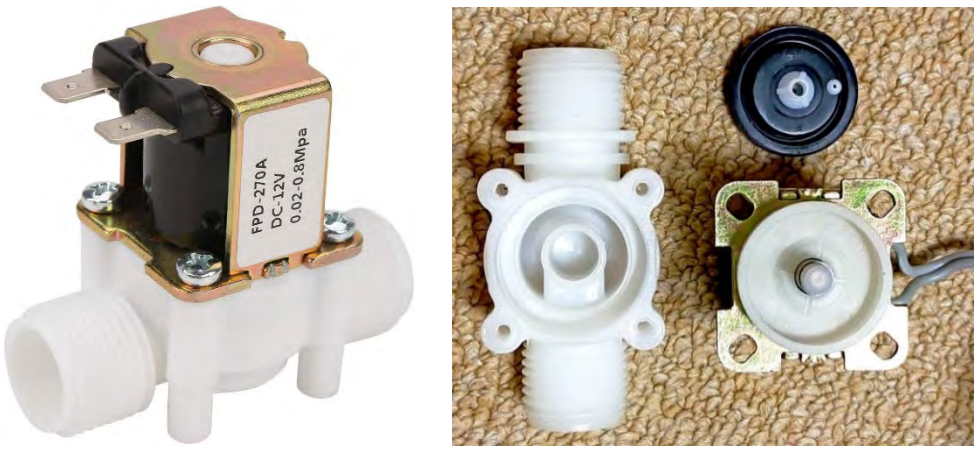


図 6.2.6　ソレノイドバルブとその内部構造

図 6.2.7 は小型水耕栽培装置用に開発したセンサモジュールでソレノイドバルブを制御するものである。小さなタッパーにバルブのコイル部分が入る穴を空け、小さな穴を 4 ヶ所に開けてピアノ線を通し、G1/2 ネジの部分を 2 ヶ所固定している。またネジにはホースをワンタッチで留められる金属のホースニップルを付けている。センサモジュールはタッパーの底に開けた 3 つの穴にネジ固定し、12V 電源と水温および液肥センサのコードを横に開けた穴から引き出している。このセンサモジュールでは循環ポンプの制御は行わないので、12V コネクタにソレノイドバルブの電源を接続した。

図 6.2.7 水耕栽培用センサによるソレノイドバルブの制御

図 6.2.8 に距離センサでソレノイドバルブを制御する注水システムの Node-RED のフロー図を、その JSON をリスト 6.2.5 に示す。また図 6.2.10 はスマートフォンのダッシュボード画面である。温湿度や水位、液肥濃度の部分は、小型水耕栽培装置とほぼ同じである。このフローはデバイス HP_SORA を対象にしている。制御はスイッチノードの「注水」がオンになっているときに、図 6.2.10 の numeric ノード「水位設定」の値によって行われる。水位は 10～100cm の範囲を 10cm 刻みで設定する。図 6.2.3 の超音波距離センサから入力ノード距離センサ「HP_SORA/#」に入力されたトピック「HP_SORA/wlevel」は、深さ 189cm のタンク上部から下の水面までの距離である。そこでリスト 6.2.3 の水位の「修正」function ノードの JavaScript によって、タンクの底からの距離に変換し、

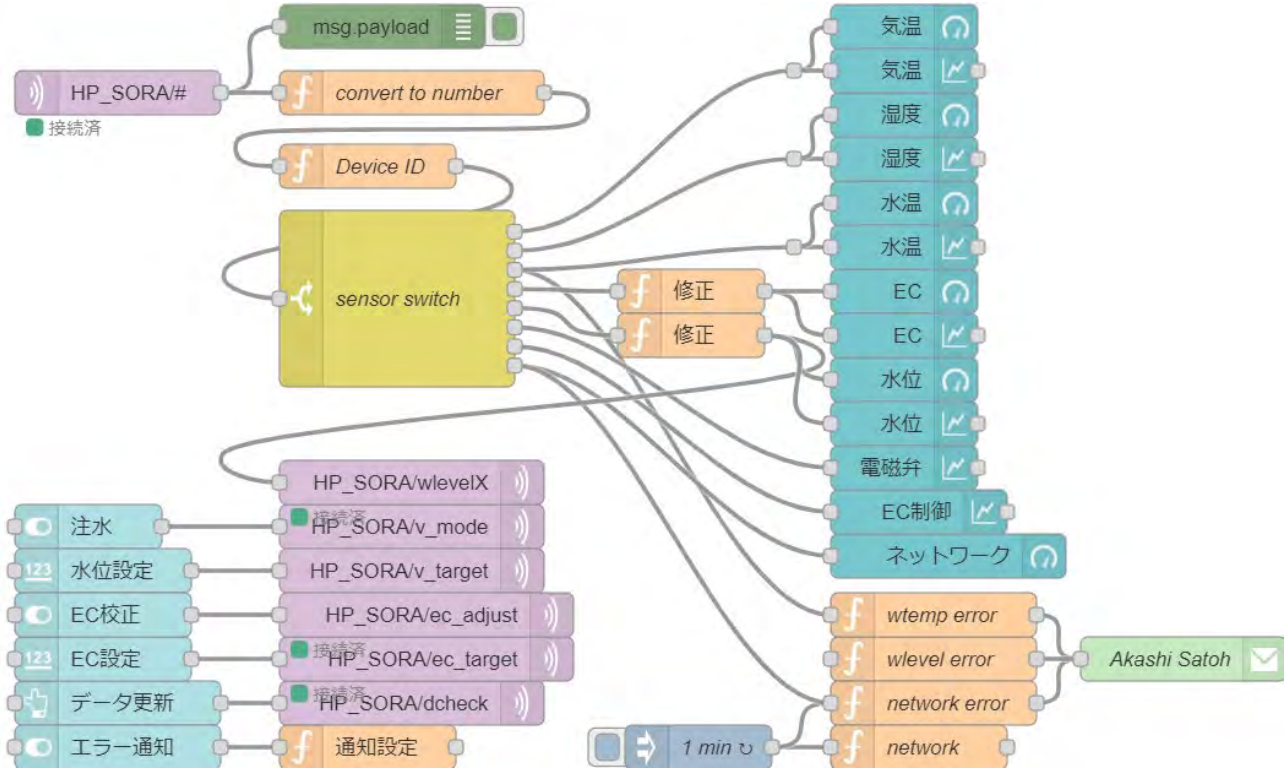


図 6.2.8 距離センサでソレノイドバルブを制御する注水システムのフロー図

小数点第 2 位までに丸めている。測定エラー時には 999 が返されるので、これよりも小さい値であることもチェックしている。求めた水位を図 6.2.7 のソレノイドバルブに送信するが、同じトピック名で Publish すると再び入力ノード「HP_SORA/#」を通ってこのフローを無限に巡回するので、「HP_SORA/wlevel」から「HP_SORA/wlevelX」に変えている。EC の「修正」ノードは、データが一定時間（10 分）に自動的に測定されたのではなく、「DCHECK」ボタンを人がタップして確認したときはこれを区別するために+100 しているので、これを減じている。なお、「6.3 水耕栽培施設への導入実験と改良」では距離センサによる水位測定の代わりに、あらかじめ決められた水位に設置したフロートスイッチのオン/オフでソレノイドバルブを制御するため、この入力ノードはフローから削除している。

図 6.2.9　numeric ノード「水位設定」の設定

リスト 6.2.2　EC の「修正」の JavaScript

```
if (msg.payload > 100)
   msg.payload = Math.round((msg.payload - 100) *
    100) / 100;
return msg;
```

リスト 6.2.3　水位の「修正」の JavaScript

```
if (msg.payload < 999) {
   msg.payload = Math.round((189 - msg.payload) *
100) / 100;
   return msg;
}
```

リスト 6.2.4 はソレノイドバルブを制御するスケッチである。関数 allback()で TOPIC_WLEVEL (= “HP_SORA/wlevelX”)を受け取ると、水位を wlevel に設定し（wlevel = atof((char *)payload);）、バルブをオンできるように“VULVE_state = true;”とする。そして、関数 vulve_control()の中の、“if (v_mode && VULVE_state && (wlevel < v_target))”でバル

図 6.2.10 注水システムのダッシュボード

ブのモード（v_mode ）が ture、先に設定した VULVE_state が true、そして測定した水位 wlevel が設定値 w_target よりも低ければ、“digitalWrite(VULVE_CTR, HIGH);”でバルブを開け、15 秒後（delay(15000);）に再びバルブを閉じる（digitalWrite(VULVE_CTR, LOW);）。またそれと同時に、バルブの電流を測定してトピック TOPIC_VULVE（=“HP_SORA/vulve”）として Publish し、動作状況をダッシュボードに状の「電磁弁」のグラフに表示する。なお、バルブが開くのは距離センサから水位が送信された直後だけで、距離センサは 10 分間隔で距離を送信させているが、図 6.2.10 では動作テストのために 3 分間隔で動いている様子が「電磁弁」に表示されている。また 10 分間隔でなくともダッシュボード上で「DCHECK」をタップすると距離センサからデータが送信されるので、水位が下がっていればバルブを開けることができる。リスト 6.1.1 の距離センサのスケッチで MQTT の再接続の判定には変数 MQTTconnected を用いていたが、リスト 6.2.4 のバルブ制御では MQTT 接続状態を基板上の LED で表示するための変数 LED を流用している。

センサモジュールは、温湿度と EC 値、水温も測定しているが、この部分は小型水耕栽培装置と同様である。

リスト 6.2.4 ソレノイドバルブを制御するスケッチ

```
// File Name  : Soramame_PEPINO_vulve.ino
// Last Update: 2023-9-1

#include "driver/adc.h"
#include "esp_adc_cal.h"
#include <PubSubClient.h>
#include <EEPROM.h>
#include <Wire.h>
#include <math.h>
#include "myMQTT.h"
#include <Adafruit_Sensor.h>
#include <Adafruit_SHTC3.h>    // Temp/Humid

#define SHTC3_SENSOR          // use SHTC30 (Temperature/Humidity)
Adafruit_SHTC3 shtc3;

esp_adc_cal_characteristics_t adcChar;
bool APmode = false;     // true:APmode  false:STAmode
bool NormalOP = true;

#define BETA 3950         // B parameter of thermistor. 3470 or 3950
#define ADC_POINTS 31      // ADC sampling points, better to be an even number.

// Pin Assign
#define VULVE_CTR    12   // vulve ON:1 / OFF:0
#define AIR_CTR      16   // air pump on:1 / off:0
#define STA_MODE      5   // AP mode:0 / STA mode:1   (SW1)
#define SW3          25   // SW3
#define SW4          26   // SW4
#define PIN_EC       34   // EC (ADC1_CH6)
#define PIN_PWM      15   // PWM (ADC1_CH6)
#define PIN_WLEVL    32   // Water Level (Analog)
#define PIN_WTEMP    35   // Water Temperature (Analog)
#define PIN_VULVE    33   // Water Vulve current (Analog)
#define POUT_WLSW    13   // Water Level mesurement activation switch
#define POUT_LED      2   // LED
#define SDA          21   // Temp/Humid (Digital) Si7006-A20-IM

/***************************** Topic *********************************/
char TOPIC_EC[128];       // Calcurated EC from ECADC and WTEMP
char TOPIC_WLEVEL[128];    // Water Level
char TOPIC_WTEMP[128];     // Water temperature
char TOPIC_TEMP[128];      // Temperature SI7006 or BME280
char TOPIC_HUMID[128];     // Temperature SI7006 or BME280
char TOPIC_VULVE[128];     // Vulve current
char TOPIC_V_MODE[128];    // Vulve Mode <on:1 / off:0>
```

```
char TOPIC_V_TARGET[128];  // Water level target fo Vulve control
char TOPIC_DCHECK[128];    // Data check
char TOPIC_EC_ADJUST[128]; // EC value adjust <on:1 / off:0>
char TOPIC_EC_MODE[128];   // EC control mode <on:1 / off:0>
char TOPIC_EC_TARGET[128]; // Target EC value for auto mode
char TOPIC_AIR[128];       // AIR air pump state  <on:1 / off:0>
char TOPIC_NETWORK[128];   // Network connection check data for every 60s

bool DCHECK       = false;
bool DCHECK_temp   = false; // 1 : check temperature if button pressed
bool DCHECK_humid  = false; // 1 : check humidity if button pressed
bool DCHECK_wtemp  = false; // 1 : check water temperature if button pressed
bool DCHECK_ec    = false; // 1 : check ec if button pressed
bool DCHECK_vulve  = false; // 1 : check vulve current if button pressed

float wlevel;       // Watar level
float vcurrent = 0;  // Water vulve current
float ec      = 0;  // Electro Conductivity
float wtemp    = 0;  // Watar temperature
float temp     = 0;  // Temperature
float humid    = 0;  // Humidity

bool  v_mode = false;    // ture : vulve control  false : vulve off
float v_target = 10.0;    // Watar level target for vulve control
bool  ec_adjust = false;  // 0: off  1: on EC adjust
bool  ec_mode   = false;  // 0: off  1: auto & adjust
float ec_target = 1.0;    // default EC target
float ec_magni  = 1.0;    // EC magnification

#define INTERVAL_EC     300    // At least 60s interval for measurement during fertilizer
addition woud be required
#define INTERVAL_WTEMP   300    // Water Temperature is not changed so fast
#define INTERVAL_TEMP    300    // Temperature on sensor board
#define INTERVAL_HUMID   300    // Humidity on sensor board
#define INTERVAL_VULVE   120    // Long interval for MQTT current data transfer even with
no data change.
#define INTERVAL_WIFI    120    // Connect to WiFi every 120s when they are disconnected.
#define INTERVAL_MQTT    120    // Connect to WiFi and MQTT every 120s when they are
disconnected.
#define INTERVAL_NETWORK  60

unsigned long last_WLEVEL  = 0;
unsigned long last_EC     = 0;
unsigned long last_WTEMP   = 0;
unsigned long last_TEMP    = 0;
unsigned long last_HUMID   = 0;
unsigned long last_VULVE   = 0;
unsigned long last_NETWORK = 0;
unsigned long last_AIR    = 0;     // Last air pump on/of time
bool        VULVE_state  = false;  // vulve enable   0:disable 1:enable
bool        AIR_state    = false;  // air pump state 0:off 1:on

/**** Cubic Curve Parameter for EC ****/
#define P00 -1.018e-01
#define P10  2.715e-03
#define P01 -6.933e-04
#define P20 -2.072e-07
#define P11 -6.209e-05
#define P02  1.071e-04
#define P30  6.804e-10
#define P21 -2.432e-08
#define P12  9.783e-07
#define P03 -3.372e-06

void sensor() {
  unsigned long now;
  char  str_buf[10];
  uint32_t sdata;
```

```
  uint32_t ADC_buf[ADC_POINTS];

  /**** network connection check ****/
  now = millis();
  if (now - last_NETWORK > INTERVAL_NETWORK * 1000) {
    mqtt.publish(TOPIC_NETWORK, "1");
    last_NETWORK = now;
  }

  /*********** Read sensor data ************/

#ifdef SHTC3_SENSOR
  /**** temp ****/
  now = millis();
  if (DCHECK_temp || (now - last_TEMP > INTERVAL_TEMP * 1000)) {
    sensors_event_t h, t;
    shtc3.getEvent(&h, &t);
    temp = t.temperature;

    if (0 < temp && temp < 100) {
      sprintf(str_buf, "%3.1f", temp);
      mqtt.publish(TOPIC_TEMP, str_buf);
    }
    Serial.printf("temp = %3.1f¥n", temp);
    DCHECK_temp = false;
    last_TEMP = now;
  }

  /**** humid ****/
  now = millis();
  if (DCHECK_humid || (now - last_HUMID > INTERVAL_HUMID * 1000)) {
    sensors_event_t h, t;
    shtc3.getEvent(&h, &t);
    humid = h.relative_humidity;

    if (0 < humid && humid < 100) {
      sprintf(str_buf, "%3.1f", humid);
      mqtt.publish(TOPIC_HUMID, str_buf);
    }
    Serial.printf("humid = %3.1f¥n", humid);
    DCHECK_humid = false;
    last_HUMID = now;
  }
#endif

  /**** wtemp ****/
  now = millis();
  if (DCHECK_wtemp || (now - last_WTEMP > INTERVAL_WTEMP * 1000)) {
    sdata = analogRead(PIN_WTEMP);
    if (sdata == 4095)
      Serial.println("Thermister is disconnected!");
    else {
      for (int i = 0; i < ADC_POINTS; i++) {
        esp_adc_cal_get_voltage(ADC_CHANNEL_7, &adcChar, &sdata);
        ADC_buf[i] = sdata;
      }

      for (int j = 0; j < ADC_POINTS - 1; j++)       // Select center value
        for (int i = 0; i < ADC_POINTS - j - 1; i++)
          if (ADC_buf[i] > ADC_buf[i + 1]) {
            sdata = ADC_buf[i];
            ADC_buf[i] = ADC_buf[i + 1];
            ADC_buf[i + 1] = sdata;
          }
      sdata = ADC_buf[ADC_POINTS / 2];

      wtemp = 10000.0 * ADC_buf[15] / (3346 - sdata);   // if sdata == 4095 then 0 division
error and WDT halt occurs.
```

```
    wtemp = (1 / (log(wtemp / 10000) / BETA + 1 / 298.15) - 273.15);
    sprintf(str_buf, "%3.1f", wtemp);
    Serial.printf("WTEMP = %s¥n", str_buf);  // Calcurated temperature WTEMP is printed
    mqtt.publish(TOPIC_WTEMP, str_buf);
    DCHECK_wtemp = false;
    last_WTEMP = now;
  }
 }

 /**** ec ****/
 now = millis();
 if (DCHECK_ec || (now - last_EC > INTERVAL_EC * 1000)) {
  ledcWrite(0, 128);    // duty 50%
  delay(200);          // wait for PWM stable

  for (int i = 0; i < ADC_POINTS; i++) {
   esp_adc_cal_get_voltage(ADC_CHANNEL_6, &adcChar, &sdata);
   ADC_buf[i] = sdata;
   //    buf[i] = analogRead(PIN_EC);
  }
  ledcWrite(0, 0);      // PWM off

  for (int j = 0; j < ADC_POINTS - 1; j++)       // Select center value
   for (int i = 0; i < ADC_POINTS - j - 1; i++)
    if (ADC_buf[i] > ADC_buf[i + 1]) {
     sdata = ADC_buf[i];
     ADC_buf[i] = ADC_buf[i + 1];
     ADC_buf[i + 1] = sdata;
    }
  sdata = ADC_buf[ADC_POINTS / 2];

  float y = wtemp, x = sdata;
  ec = P00
      + P10 * x + P01 * y
      + P20 * x*x + P11 * x*y + P02 * y*y
      + P30 * x*x*x + P21 * x*x*y + P12 * x*y*y + P03 * y*y*y;
  ec *= ec_magni;

  if (DCHECK_ec)
   sprintf(str_buf, "%.3f", ec + 100);  // ec (+ 100) will not be avareged in node-red
  else
   sprintf(str_buf, "%3.2f", ec);
  mqtt.publish(TOPIC_EC, str_buf);
  Serial.print("ec =");
  Serial.println(str_buf);
  if (DCHECK_ec && ec_adjust && (!ec_mode) && (ec > 0.1)) {
   ec_magni = ec_magni * ec_target / ec;
   sprintf(str_buf, "%3.2f", ec_target + 100);
   mqtt.publish(TOPIC_EC, str_buf);
   sprintf(str_buf, "%3.2f", ec_magni);
   Serial.printf("ec magni = %s¥n", str_buf);
   EEPROM.write(146, (char)(ec_magni * 100));  // EC adjustment value ec_magni * 100
   EEPROM.commit();
   Serial.println("EC magnification value write to EPROM");
  }
  DCHECK_ec = false;
  last_EC = now;
 }
}

bool MQTTconnectedX = false;

void vulve_control() {
 unsigned long now = millis();
 char  str_buf[10];
 uint32_t sdata;

 if (DCHECK_vulve || (now - last_VULVE > INTERVAL_VULVE * 1000)) {
```

```
    sdata = analogRead(PIN_VULVE);
    sprintf(str_buf, "%4u", sdata);
    mqtt.publish(TOPIC_VULVE, str_buf);
    last_VULVE = now;
    DCHECK_vulve = false;
  }

  if (v_mode && VULVE_state && (wlevel < v_target)) {
    mqtt.publish(TOPIC_VULVE, "0");
    digitalWrite(VULVE_CTR, HIGH);       // vulve on
    sdata = analogRead(PIN_VULVE);
    sprintf(str_buf, "%4u", sdata);
    mqtt.publish(TOPIC_VULVE, str_buf);
    Serial.printf("vulve on: %d¥n", sdata);

    delay(15000);

    digitalWrite(VULVE_CTR, LOW);       // vulve off
    sdata = analogRead(PIN_VULVE);
    sprintf(str_buf, "%4u", sdata);
    mqtt.publish(TOPIC_VULVE, str_buf);
    mqtt.publish(TOPIC_VULVE, "0");
    Serial.printf("vulve off: %d¥n", sdata);
    VULVE_state = false;
  }
}

void ec_control()
{
  unsigned long now = millis();

  // air vulve interval 15 minuites
  if (ec_mode && !AIR_state && ((now - last_AIR) > 15 * 60000) && (ec < ec_target) && (0.4
< ec) && mqtt.connected()) {
    digitalWrite(AIR_CTR, HIGH);       // AIR air pump on
    mqtt.publish(TOPIC_AIR, "0");
    mqtt.publish(TOPIC_AIR, "1");
    AIR_state = true;
    Serial.println("air pump on");
    last_AIR = now;
  }
  // air pump off after 8 seconds
  else if (AIR_state && (now - last_AIR > 8 * 1000)) {
    digitalWrite(AIR_CTR, LOW);        //air pump off
    mqtt.publish(TOPIC_AIR, "1");
    mqtt.publish(TOPIC_AIR, "0");
    AIR_state = 0;
    Serial.println("air pump off");
    last_AIR = now;
  }
}

/***********callback function work when message is arrived**********/
void callback(char* topic, byte* payload, unsigned int length) {
  payload[length] = '¥0';        // The last published string data is atteched at the
last of the subscribed string data. So, insert EOL.
  Serial.printf("Msg arrived [%s]¥n", topic);

  if (String(topic) == String(TOPIC_DCHECK)) {
    DCHECK_temp = DCHECK_humid = DCHECK_wtemp = DCHECK_ec = DCHECK_vulve = true;
    Serial.println("Data Check");
  }
  else if (String(topic) == String(TOPIC_WLEVEL)) {
    wlevel = atof((char *)payload);
    Serial.println(wlevel);
    VULVE_state = true;             // Water vulve control enable
  }
  else if (String(topic) == String(TOPIC_V_MODE)) {
```

```
    v_mode = (payload[0] == 't') ? true : false;
    Serial.println(v_mode);
  }
  else if (String(topic) == String(TOPIC_V_TARGET)) {
    v_target = atof((char *)payload);
    Serial.println(v_target);
  }
  else if (String(topic) == String(TOPIC_EC_ADJUST)) {
    ec_adjust = (payload[0] == 't') ? true : false;
    Serial.println(ec_adjust);
  }
  else if (String(topic) == String(TOPIC_EC_TARGET)) {
    ec_target = atof((char *)payload);
    Serial.println(ec_target);
  }
  else if (String(topic) == String(TOPIC_EC_MODE)) {
    ec_mode = (payload[0] == 't') ? true : false;
    Serial.println(ec_mode);
    if (ec_mode) {
      if ((ec < ec_target) && (0.4 < ec)) { // if EC value is too low (less than 0.4)
something wrong
       digitalWrite(AIR_CTR, HIGH);         // When ec mode is swiched from off -> on,
adjust immediatly
       mqtt.publish(TOPIC_AIR, "0");
       mqtt.publish(TOPIC_AIR, "1");
       AIR_state = 1;
       Serial.println("air pump on");
       last_AIR = millis();
      }
    }
  }
}

/****************************** Setup ******************************/
void setup() {
  Serial.begin(115200);

  adc_power_on();                                          // ADC enable
  adc_gpio_init(ADC_UNIT_1, ADC_CHANNEL_6);                 // ADC1_CH6 (GPIO34 )enable
  adc_gpio_init(ADC_UNIT_1, ADC_CHANNEL_7);                 // ADC1_CH7 (GPIO35 )enable
  adc1_config_width(ADC_WIDTH_BIT_12);     // ADC1 12bit (0-4095)
  adc1_config_channel_atten(ADC1_CHANNEL_6, ADC_ATTEN_DB_11);   // ADC1_CH6 -11dB
  adc1_config_channel_atten(ADC1_CHANNEL_7, ADC_ATTEN_DB_11);   // ADC1_CH7 -11dB
  esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,  ADC_WIDTH_BIT_12, 1100,
&adcChar); // set parameters to addChar

  pinMode(STA_MODE,  INPUT_PULLUP);
  pinMode(VULVE_CTR, OUTPUT);
  pinMode(AIR_CTR,   OUTPUT);
  pinMode(PIN_EC,    INPUT);
  pinMode(PIN_WTEMP, INPUT);
  pinMode(PIN_VULVE, INPUT);
  pinMode(PIN_PWM,   OUTPUT);
  pinMode(POUT_WLSW, OUTPUT);
  pinMode(POUT_LED,  OUTPUT);

  Wire.begin(21, 22);
  Wire.endTransmission();

  ledcSetup(0, 5000, 8);         // Set PWM frequiency 5KHz, 8bit
  ledcAttachPin(PIN_PWM, 0);     // Assing PWM pin to 15

#ifdef SHTC3_SENSOR
  shtc3 = Adafruit_SHTC3();
  if (! shtc3.begin())
    Serial.println("Couldn't find SHTC3");
#endif
```

```
  APmode   = !digitalRead(STA_MODE);
  setup_MQTT(APmode);

  digitalWrite(VULVE_CTR, LOW);  // vulve off
  digitalWrite(AIR_CTR, LOW);    // Stop AIR air pump for EC control

  ////// MQTT topic initialization
  create_topic("ec",       TOPIC_EC);
  create_topic("wlevelX",   TOPIC_WLEVEL);
  create_topic("wtemp",     TOPIC_WTEMP);
  create_topic("temp",      TOPIC_TEMP);
  create_topic("humid",     TOPIC_HUMID);
  create_topic("vulve",     TOPIC_VULVE);
  create_topic("v_mode",    TOPIC_V_MODE);
  create_topic("v_target",  TOPIC_V_TARGET);
  create_topic("air",       TOPIC_AIR);
  create_topic("ec_adjust", TOPIC_EC_ADJUST);
  create_topic("ec_mode",   TOPIC_EC_MODE);
  create_topic("ec_target", TOPIC_EC_TARGET);
  create_topic("dcheck",    TOPIC_DCHECK);
  create_topic("network",   TOPIC_NETWORK);

  ec_magni = (EEPROM.read(146)) / 100.0;   // read ec adjustment magnification value
  Serial.printf("EC magnification value: %3.2f¥n", ec_magni);
  if (ec_magni < 0.2)                   // magnification might be wrong
    ec_magni = 1.0;
}

bool LED = false;

void loop() {
  if (APmode)
    AP_start();
  else if (WiFiMQTT_reconnect()) {
    if (!LED) {
      digitalWrite(POUT_LED, HIGH);
      LED = true;
      mqtt.subscribe(TOPIC_WLEVEL);
      mqtt.subscribe(TOPIC_V_MODE);
      mqtt.subscribe(TOPIC_V_TARGET);
      mqtt.subscribe(TOPIC_EC_ADJUST);
      mqtt.subscribe(TOPIC_EC_MODE);
      mqtt.subscribe(TOPIC_EC_TARGET);
      mqtt.subscribe(TOPIC_DCHECK);
    }
    mqtt.loop();
    sensor();
  }
  else if (LED) {
    digitalWrite(POUT_LED, LOW);
    LED = false;
  }

  delay(100);
  vulve_control();
  ec_control();
}
```

6.3　水耕栽培施設への導入実験と改良

図 6.3.1 は千歳烏山のハウス内のトマトの水耕栽培施設である。そして図 6.3.2 は、ここにセンサ制御システムを構築するための実験を行っている様子である。地下のタンクから溶液をポンプで汲み上げ、図 6.3.1 の立てた塩ビパイプに上から潅水し、再びタンクに回収する。図 6.2.5 の距離センサを図 6.3.2 のように薄い金属板で上部

に固定し、水面からの距離を測定する。センサの位置から底面までは 189cm であった。ハウスに隣接する部屋の Wi-Fi に、距離センサと図 6.2.7 のセンサモジュールとソレノイドバルブによる注水装置を接続し、遠隔の Node-RED サーバを介してスマートフォンで制御を行った。距離センサと注水装置の連動もうまくいき、システムとしては想定どおりに動作した。しかしながら、距離センサの 2 つの値が大きく異なり、正しく測定できないという問題が生じた。原因を調べたるために距離センサの設置位置を変えながら測定を行ったところ、測定用の音波が深いつぼ型をしたタンクの壁面で反射することが原因と見られた。

図 6.3.1　ハウス内のトマトの水耕栽培システム

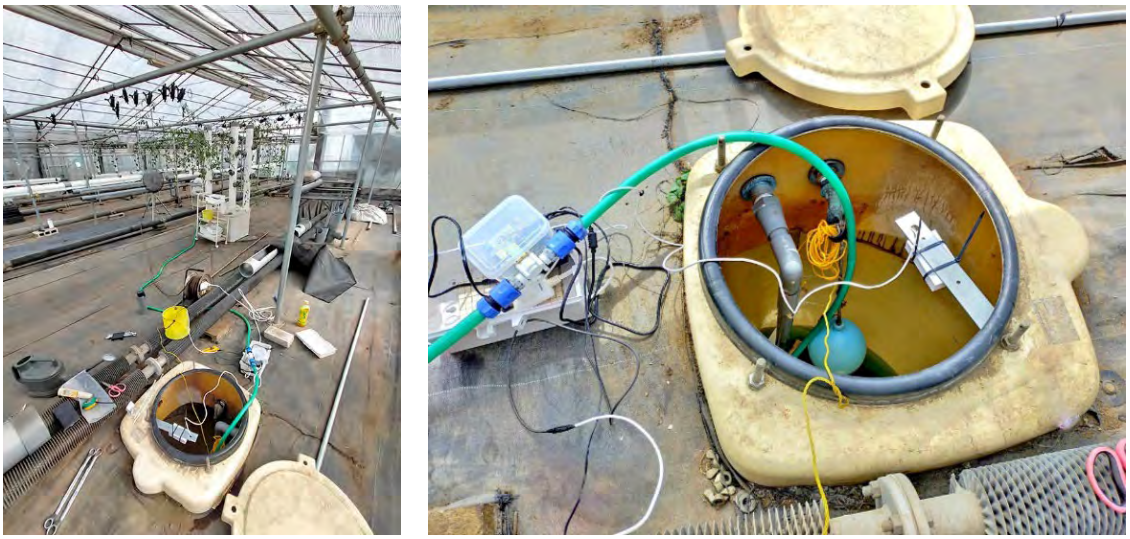


図 6.3.2　水耕栽培施設での距離センサとソレノイドバルブによる注水機構の設置実験の様子

距離センサを用いることができればタンクの水位を任意に制御できるが、レーザー距離センサでも同様の結果になると見られる。そこで任意の調整は断念し、図 6.3.3 のフロートスイッチによる固定水位調節を行うこととした。浮き輪が下にあるときは黄色い 2 本のケーブルは開放状態にあり、浮き輪が水で上に上がるとケーブルは短絡される。これを長さ約 2m、外径 18mm の塩ビ管（VP13）に固定してタンク

図 6.3.3　フロートスイッチ

内に立て、ケーブルをパイプ内に通してタンクの外に置いた図 6.2.7 のソレノイドバルブモジュールの白いセンサ基板に接続する。

図 6.3.2 の実装では距離センサと注水モジュールが分かれているため、距離センサは水位データを管理サーバに送信し、そのデータが設定水位以下かどうかをサーバが判断して、注水モジュールに制御信号を送信していた。これに対してフロートスイッチを用いた実装は、ソレノイドバブルを有した注水モジュールのセンサ基板に直接スイッチを接続している。そのためセンサ基板が管理サーバを介さずに直接注水を行う。このシステム構成を図 6.3.4 に示す。なお注水はフロートスイッチが開放(低水位)から短絡(高水位)になるまで連続して続けるのではなく、10 分サイクルのセンシングで低水位を検知したときに 15 秒だけ行うようにしている。これはフロートスイッチの故障等でバルブが開いたままの状態となり、水がタンクから溢れるのを防止するためである。水は短時間で大量に消費されるわけではないので少しずつ注水すればよいが、注水が一回も止まることなく長時間繰り返されるようであればなんらかの異常が生じていると判断することができる。これに対して一度に連続して注水する実装でフロートスイッチに異常が生じたときは、すぐに水があふれてしまうためそれに気付くのが遅れてしまうことになる。なお、ソレノイドバルブは 12V 電源が落ちたり、センサ基板の 3.3V 電源が落ちたときには閉じるようになっている。リスト 6.3.1 にフロートスイッチでソレノイドバルブを制御するためのスケッチを示しておく。

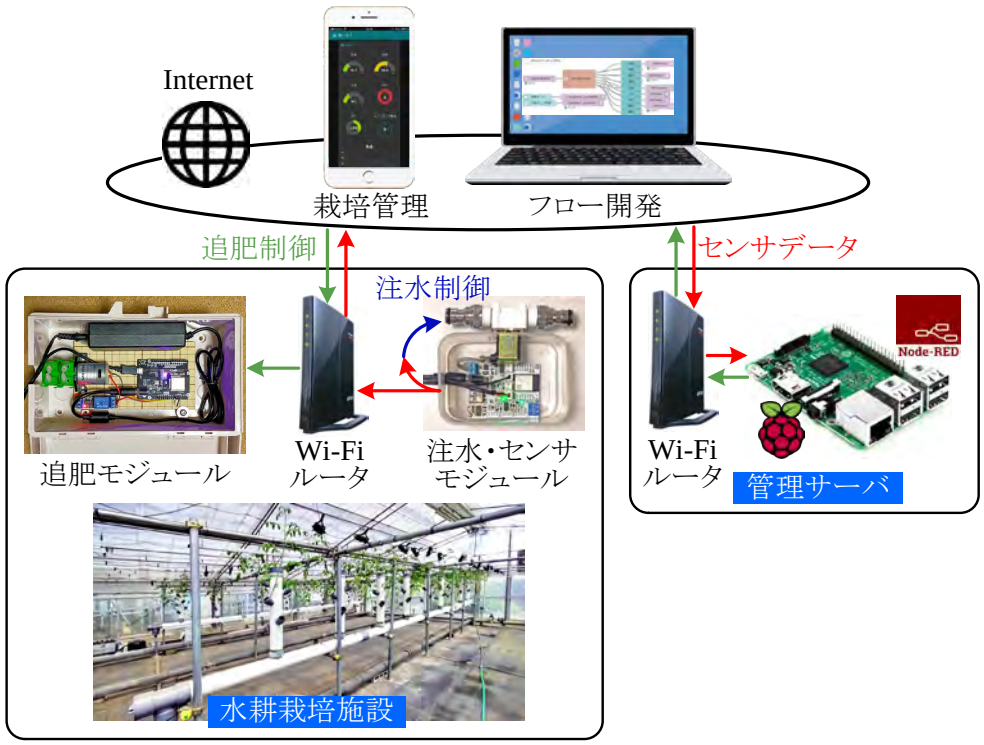


図 6.3.4　水耕栽培システムの構成図

図 6.3.5 は注水モジュールが測定した液肥濃度を受信した管理サーバによって制御される、追肥モジュールとその配線図である。図 6.1.7 の追肥モジュールと基本的に同じであるが、このモジュールはタンクから離して設置し、液肥濃度の測定は注水モジュールで行うためセンサ機能は不要となったため、センサ基板と同じ ESP32 を用いた市販の WeMos D1 R32 ボード(1,000 円以下)に置き換えている。なお追肥はタンクに直接行うのではなく、水耕栽培のパイプに流した液肥を回収する途中経路に入れることとした。蠕動ポンプを制御する電磁リレーは 3.3V 電源が落ちていたり、ESP32 の起動時に IO4 からの制御信号 0V の時には、ポンプへの 12V

電力を遮断して回らないようにしている。Wi-Fi を介して制御サーバと通信できるようになると、IO2 に接続されている青色の LED を点灯させるようにしている。また通信状況が切れた場合はリトライをかけ、繋がるまでは LED を消灯する。IO18 は内部でプルアップしており、このピンをジャンパで GND に接続して起動すると、ESP32 に Web サーバが立ち上がり、そこに接続することで Wi-Fi と制御サーバの設定を行うことができる。リスト 6.3.2 にこの追肥モジュールで蠕動ポンプを制御するためのスケッチを示す。

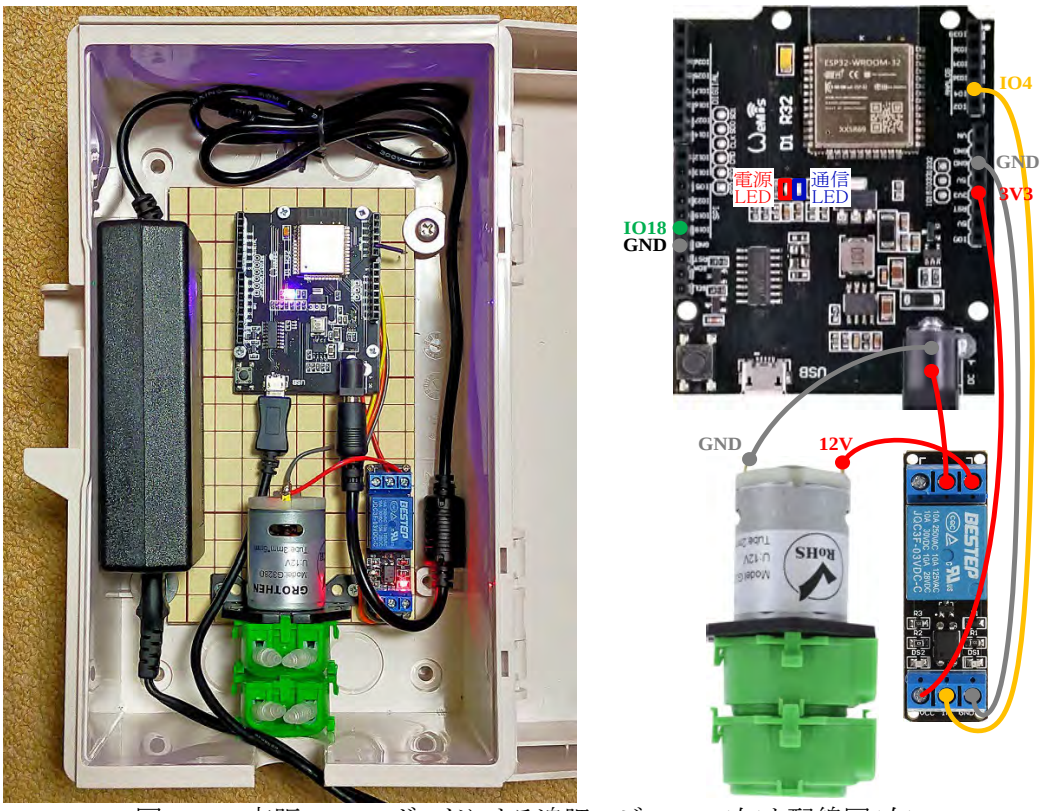


図 6.3.5　市販 ESP32 ボードによる追肥モジュール(左)と配線図(右)

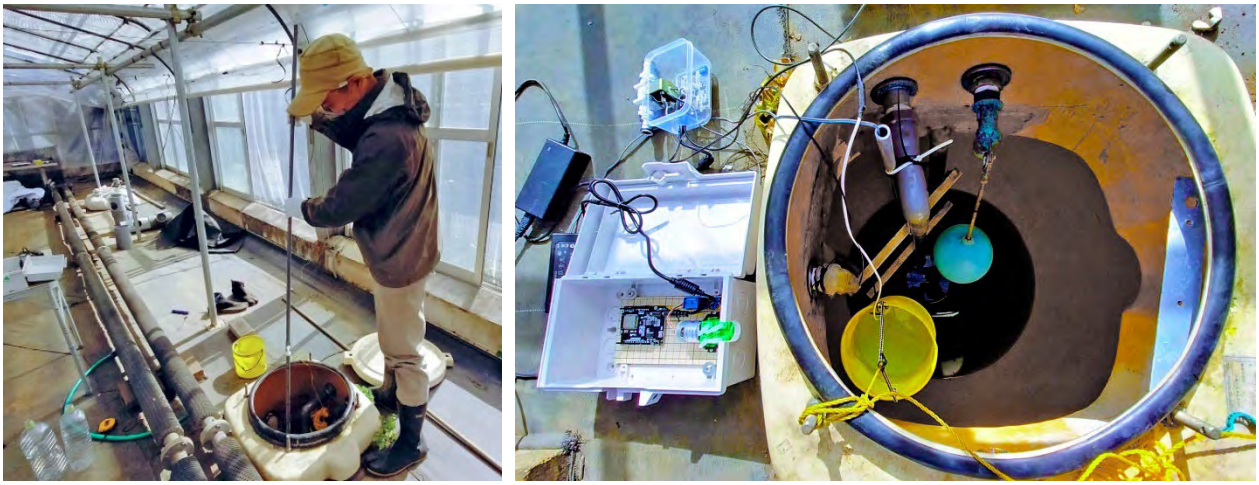


図 6.3.6　フロートスイッチを使った水位制御機構と追肥モジュールの設置

図 6.3.6 にフロートスイッチを使った水位制御機構と、追肥モジュールの設置の様子を示す。長さ約 2m の概形 18mm の塩ビパプ VP13 をタンクの中に立て、下から 50cm の場所にフロートを固定し、延長したコードをパイプの中を通して上まで引き出している。注水時にパイプが浮いてこないように、下 50cm にはドリルで 10 個の穴を空けて水がパイプ内に流入しやすいようにしている。また延長コードの接続部は熱収縮チューブと自己融着テー

プで防水した。図 6.3.7 は設置時の動作テスト中のダッシュボード画面である。注水用の電磁弁と追肥用の蠕動ポンプは動かしているが、ホースや液肥タンクはつないでいないので、水位は低位を示す 0、EC 値は初期値の約 0.9 から変化しない。リスト 6.3.1 で INTERVAL_VULVE を 300 秒にセットすることで、5 分に一回、注水の判断をして対応した動作を行わせている。また、追肥はリスト 6.3.2 の下記のコードで 15 分に一回、判断と動作を行わせている。

```
if (ec_mode && !AIR_state && ((now - last_AIR) > 15 * 60000) && (ec < ec_target) && (0.4 < ec) && mqtt.connected())
```

水位は 0 のまま、EC 値は設定の 1.0 よりも常に低い 0.9 なので、電磁弁は 5 分毎、液肥ポンプは 15 分に一回定期的に動作するはずである。しかし図 6.3.7 の左のグラフで、電磁弁は 30 分に一回しか動作していなかったり、液肥ポンプグラフが大きく乱れ、数分間隔で動作したりしている。これは通信障害が原因で、センサモジュールを設置したハウスからは Wi-Fi ルータを設置した部屋まで 30m 以上の離れていることによるものである。注水のための電磁弁用センサモジュールはタンクの近くに置く必要があるが、追肥はタンクに回収される液肥用のパイプのどこから注入してもよいため、さらに 10m ほど離れた装置の近くに設置している。このため液肥ポンプの通信状況は電磁弁に比べて悪い。通信が遮断されると、注水や追肥は行わないようにしているため間隔が広がる。また逆に感覚が短くなっているところがあるが、これは通信が復帰して MQTT で再接続を行う際に、callback() 関数内でインターバル用のタイマをリセットしているためである。なお図 6.3.7 左は、特に通信状態が悪かったときのグラフで、図 6.3.7 右はその後、状態が良くなった時の物である。

通信が長時間切れると注水が行われずに枯れてしまう心配もあるが、通信が正常であれば注水や追肥は数分間隔で行う必要はない。センサデータのチェックを頻繁に 5 分間隔で行っているのは、このように通信が不安定な環境下でにおいて繋がる確率を高めるためである。なお、ネットワークのチェックと遮断後の再接続は 1 分間隔で行っている。

図 6.3.7 動作実験中のダッシュボード画面

リスト 6.3.1　フロートスイッチでソレノイドバルブを制御するスケッチ

```
// File Name  : Soramame_PEPINO_vulve.ino
// Last Update: 2024-1-24
#include "driver/adc.h"
#include "esp_adc_cal.h"
#include <PubSubClient.h>
#include <EEPROM.h>
#include <Wire.h>
#include <math.h>
#include "myMQTT.h"
#include <Adafruit_Sensor.h>
#include <Adafruit_SHTC3.h> // Temp/Humid

#define intWL_SENSOR       // use internal water level sensor

// #define SHTC3_SENSOR    // use SHTC30 (Temperature/Humidity)
// Adafruit_SHTC3 shtc3;

esp_adc_cal_characteristics_t adcChar;
bool APmode = false;       // true:APmode  false:STAmode
bool NormalOP = true;

// #define RST_CYCLE 3710000 // ESP32 restart cycle (ms)
#define BETA 3470          // B parameter of thermistor. 3470 or 3950
#define ADC_POINTS 31      // ADC sampling points, better to be an even number.

// Pin Assign
#define VULVE_CTR    12    // vulve ON:1 / OFF:0
#define AIR_CTR      16    // air pump on:1 / off:0
#define STA_MODE      5    // AP mode:0 / STA mode:1   (SW1)
#define SW3          25    // SW3
#define SW4          26    // SW4
#define PIN_EC       34    // EC (ADC1_CH6)
#define PIN_PWM      15    // PWM (ADC1_CH6)
#define PIN_WLEVL    32    // Water Level (Analog)
#define PIN_WTEMP    35    // Water Temperature (Analog)
#define PIN_VULVE    33    // Water Vulve current (Analog)
#define POUT_WLSW    13    // Water Level mesurement activation switch
#define POUT_LED      2    // LED
#define SDA          21    // Temp/Humid (Digital) Si7006-A20-IM

/***************************** Topic *********************************/
char TOPIC_EC[128];       // Calcurated EC from ECADC and WTEMP
char TOPIC_WLEVEL[128];    // Water Level
char TOPIC_WTEMP[128];    // Water temperature
char TOPIC_TEMP[128];     // Temperature SI7006 or BME280
char TOPIC_HUMID[128];    // Temperature SI7006 or BME280
char TOPIC_VULVE[128];    // Vulve current
char TOPIC_V_MODE[128];   // Vulve Mode <on:1 / off:0>
char TOPIC_V_TARGET[128];  // Water level target fo Vulve control
char TOPIC_DCHECK[128];   // Data check
char TOPIC_EC_ADJUST[128]; // EC value adjust <on:1 / off:0>
char TOPIC_EC_MODE[128];  // EC control mode <on:1 / off:0>
char TOPIC_EC_TARGET[128]; // Target EC value for auto mode
char TOPIC_AIR[128];      // AIR air pump state  <on:1 / off:0>
char TOPIC_NETWORK[128];  // Network connection check data for every 60s

bool DCHECK       = false;
bool DCHECK_temp   = false; // 1 : check temperature if button pressed
bool DCHECK_humid  = false; // 1 : check humidity if button pressed
bool DCHECK_wlevel = false; // 1 : check water level if button pressed
bool DCHECK_wtemp  = false; // 1 : check water temperature if button pressed
bool DCHECK_ec    = false; // 1 : check ec if button pressed
bool DCHECK_vulve  = false; // 1 : check vulve current if button pressed

float wlevel;       // Watar level
float vcurrent = 0;  // Water vulve current
```

```
float ec      = 0;  // Electro Conductivity
float wtemp    = 0;  // Watar temperature
float temp     = 0;  // Temperature
float humid    = 0;  // Humidity

bool  v_mode = false;     // ture : vulve control  false : vulve off
float v_target = 1.0;     // Watar level target for vulve control
bool  ec_adjust = false;  // 0: off  1: on EC adjust
bool  ec_mode   = false;  // 0: off  1: auto & adjust
float ec_target = 1.0;    // default EC target
float ec_magni  = 1.0;    // EC magnification

#define INTERVAL_EC     300    // At least 60s interval for measurement during fertilizer
addition woud be required
#define INTERVAL_WLEVL   300    // Water Level will be controlled by a solenoid valve in
some implementation
#define INTERVAL_WTEMP   300    // Water Temperature is not changed so fast
#define INTERVAL_TEMP    300    // Temperature on sensor board
#define INTERVAL_HUMID   300    // Humidity on sensor board
#define INTERVAL_VULVE   300    // Long interval for MQTT current data transfer even with
no data change.
#define INTERVAL_WIFI    120    // Connect to WiFi every 120s when they are disconnected.
#define INTERVAL_MQTT    120    // Connect to WiFi and MQTT every 120s when they are
disconnected.
#define INTERVAL_NETWORK  60

unsigned long last_EC     = 0;
unsigned long last_WLEVL   = 0;
unsigned long last_WTEMP   = 0;
unsigned long last_TEMP    = 0;
unsigned long last_HUMID   = 0;
unsigned long last_VULVE   = 0;
unsigned long last_NETWORK = 0;
unsigned long last_AIR    = 0;     // Last air pump on/of time
bool        VULVE_state  = false;  // vulve enable   0:disable 1:enable
bool        AIR_state    = false;  // air pump state 0:off 1:on

/**** Cubic Curve Parameter for EC ****/
#define P00 -1.018e-01
#define P10  2.715e-03
#define P01 -6.933e-04
#define P20 -2.072e-07
#define P11 -6.209e-05
#define P02  1.071e-04
#define P30  6.804e-10
#define P21 -2.432e-08
#define P12  9.783e-07
#define P03 -3.372e-06

void sensor() {
  unsigned long now;
  char  str_buf[10];
  uint32_t sdata;
  uint32_t ADC_buf[ADC_POINTS];

  /**** network connection check ****/
  now = millis();
  if (now - last_NETWORK > INTERVAL_NETWORK * 1000) {
    mqtt.publish(TOPIC_NETWORK, "1");
    last_NETWORK = now;
  }

  /*********** Read sensor data ************/
#ifdef SHTC3_SENSOR
  /**** temp ****/
  now = millis();
  if (DCHECK_temp || (now - last_TEMP > INTERVAL_TEMP * 1000)) {
    sensors_event_t h, t;
```

```
    shtc3.getEvent(&h, &t);
    temp = t.temperature;

    if (0 < temp && temp < 100) {
      sprintf(str_buf, "%3.1f", temp);
      mqtt.publish(TOPIC_TEMP, str_buf);
    }
    Serial.printf("temp = %3.1f¥n", temp);
    DCHECK_temp = false;
    last_TEMP = now;
  }

  /**** humid ****/
  now = millis();
  if (DCHECK_humid || (now - last_HUMID > INTERVAL_HUMID * 1000)) {
    sensors_event_t h, t;
    shtc3.getEvent(&h, &t);
    humid = h.relative_humidity;

    if (0 < humid && humid < 100) {
      sprintf(str_buf, "%3.1f", humid);
      mqtt.publish(TOPIC_HUMID, str_buf);
    }
    Serial.printf("humid = %3.1f¥n", humid);
    DCHECK_humid = false;
    last_HUMID = now;
  }
#endif

  /**** wtemp ****/
  now = millis();
  if (DCHECK_wtemp || (now - last_WTEMP > INTERVAL_WTEMP * 1000)) {
    sdata = analogRead(PIN_WTEMP);
    if (sdata == 4095)
      Serial.println("Thermister is disconnected!");
    else {
      for (int i = 0; i < ADC_POINTS; i++) {
        esp_adc_cal_get_voltage(ADC_CHANNEL_7, &adcChar, &sdata);
        ADC_buf[i] = sdata;
      }

      for (int j = 0; j < ADC_POINTS - 1; j++)       // Select center value
        for (int i = 0; i < ADC_POINTS - j - 1; i++)
          if (ADC_buf[i] > ADC_buf[i + 1]) {
            sdata = ADC_buf[i];
            ADC_buf[i] = ADC_buf[i + 1];
            ADC_buf[i + 1] = sdata;
          }
      sdata = ADC_buf[ADC_POINTS / 2];

      wtemp = 10000.0 * ADC_buf[15] / (3346 - sdata);  // if sdata == 4095 then 0 division
error and WDT halt occurs.
      wtemp = (1 / (log(wtemp / 10000) / BETA + 1 / 298.15) - 273.15);
      sprintf(str_buf, "%3.1f", wtemp);
      Serial.printf("WTEMP = %s¥n", str_buf);  // Calcurated temperature WTEMP is printed
      mqtt.publish(TOPIC_WTEMP, str_buf);
      DCHECK_wtemp = false;
      last_WTEMP = now;
    }
  }

  /**** ec ****/
  now = millis();
  if (DCHECK_ec || (now - last_EC > INTERVAL_EC * 1000)) {
    ledcWrite(0, 128);    // duty 50%
    delay(200);          // wait for PWM stable

    for (int i = 0; i < ADC_POINTS; i++) {
```

```
     esp_adc_cal_get_voltage(ADC_CHANNEL_6, &adcChar, &sdata);
     ADC_buf[i] = sdata;
     //    buf[i] = analogRead(PIN_EC);
    }
    ledcWrite(0, 0);      // PWM off

    for (int j = 0; j < ADC_POINTS - 1; j++)       // Select center value
      for (int i = 0; i < ADC_POINTS - j - 1; i++)
        if (ADC_buf[i] > ADC_buf[i + 1]) {
          sdata = ADC_buf[i];
          ADC_buf[i] = ADC_buf[i + 1];
          ADC_buf[i + 1] = sdata;
        }
    sdata = ADC_buf[ADC_POINTS / 2];

    float y = wtemp, x = sdata;
    ec = P00
        + P10*x + P01*y
        + P20*x*x + P11*x*y + P02*y*y
        + P30*x*x*x + P21*x*x*y + P12*x*y*y + P03*y*y*y;
    ec *= ec_magni;

    if (DCHECK_ec)
      sprintf(str_buf, "%.3f", ec + 100);  // ec (+ 100) will not be avareged in node-red
    else
      sprintf(str_buf, "%3.2f", ec);
    mqtt.publish(TOPIC_EC, str_buf);
    Serial.print("ec =");
    Serial.println(str_buf);

    if (DCHECK_ec && ec_adjust && (!ec_mode) && (ec > 0.1)) {
      ec_magni = ec_magni * ec_target / ec;
      sprintf(str_buf, "%3.2f", ec_target + 100);
      mqtt.publish(TOPIC_EC, str_buf);
      sprintf(str_buf, "%3.2f", ec_magni);
      Serial.printf("ec magni = %s¥n", str_buf);
      EEPROM.write(146, (char)(ec_magni * 100));      // EC adjustment value ec_magni *
100
      EEPROM.commit();
      Serial.printf("EC magnification value (%3.2f) write to EPROM¥n", ec_magni);
    }
    DCHECK_ec = false;
    last_EC = now;
  }

#ifdef intWL_SENSOR
  /**** wlevel ****/
  now = millis();
  if (DCHECK_wlevel || (now - last_WLEVL > INTERVAL_WLEVL * 1000)) {
    digitalWrite(POUT_WLSW, HIGH);   // Start measurement
    delay(5);
    sdata = analogRead(PIN_WLEVL);
    digitalWrite(POUT_WLSW, LOW);    // Stop measurement
    Serial.printf("wlevel = %d¥n", sdata);
    wlevel = (sdata < 2000U)? 1: 0;
    sprintf(str_buf, "%3.0f", wlevel);
    mqtt.publish(TOPIC_WLEVEL, str_buf);
    DCHECK_wlevel = false;
    last_WLEVL = now;
    VULVE_state = true;
  }
#endif
}

bool MQTTconnectedX = false;

void vulve_control()
{
```

```
  unsigned long now = millis();
  char  str_buf[10];
  uint32_t sdata;

  if (DCHECK_vulve || (now - last_VULVE > INTERVAL_VULVE * 1000)) {
    sdata = analogRead(PIN_VULVE);
    sprintf(str_buf, "%4u", sdata);
    mqtt.publish(TOPIC_VULVE, str_buf);
    last_VULVE = now;
    DCHECK_vulve = false;
  }

#ifdef intWL_SENSOR
  v_target = 1;
#endif
  if (v_mode && VULVE_state && (wlevel < v_target)) {
    mqtt.publish(TOPIC_VULVE, "0");
    digitalWrite(VULVE_CTR, HIGH);       // vulve on
    sdata = analogRead(PIN_VULVE);
    sprintf(str_buf, "%4u", sdata);
    mqtt.publish(TOPIC_VULVE, str_buf);
    Serial.printf("vulve on: %d¥n", sdata);

    delay(15000);                     // vulbe open for 15sec

    digitalWrite(VULVE_CTR, LOW);       // vulve off
    sdata = analogRead(PIN_VULVE);
    sprintf(str_buf, "%4u", sdata);
    mqtt.publish(TOPIC_VULVE, str_buf);
    mqtt.publish(TOPIC_VULVE, "0");
    Serial.printf("vulve off: %d¥n", sdata);
    VULVE_state = false;
  }
}

void ec_control()
{
  unsigned long now = millis();

  // air vulve interval 15 minuites
  if (ec_mode && !AIR_state && ((now - last_AIR) > 15 * 60000) && (ec < ec_target) && (0.4
< ec) && mqtt.connected()) {
    digitalWrite(AIR_CTR, HIGH);       // AIR air pump on
    mqtt.publish(TOPIC_AIR, "0");
    mqtt.publish(TOPIC_AIR, "1");
    AIR_state = true;
    Serial.println("air pump on");
    last_AIR = now;
  }
  // air pump off after 8 seconds
  else if (AIR_state && (now - last_AIR > 8 * 1000)) {
    digitalWrite(AIR_CTR, LOW);        //air pump off
    mqtt.publish(TOPIC_AIR, "1");
    mqtt.publish(TOPIC_AIR, "0");
    AIR_state = 0;
    Serial.println("air pump off");
    last_AIR = now;
  }
}

/***********callback function work when message is arrived***********/
void callback(char* topic, byte* payload, unsigned int length) {
  payload[length] = '¥0';         // The last published string data is atteched at the
last of the subscribed string data. So, insert EOL.
  Serial.printf("Msg arrived [%s]¥n", topic);

  if (String(topic) == String(TOPIC_DCHECK)) {
```

```
    DCHECK_temp = DCHECK_humid = DCHECK_wtemp = DCHECK_ec = DCHECK_wlevel = DCHECK_vulve =
true;
    Serial.println("Data Check");
  }
#ifndef intWL_SENSOR
  else if (String(topic) == String(TOPIC_WLEVEL)) {
    wlevel = atof((char *)payload);
    Serial.println(wlevel);
    VULVE_state = true;             // Water vulve control enable
  }
#endif
  else if (String(topic) == String(TOPIC_V_MODE)) {
    v_mode = (payload[0] == 't') ? true : false;
    Serial.println(v_mode);
  }
  else if (String(topic) == String(TOPIC_V_TARGET)) {
    v_target = atof((char *)payload);
    Serial.println(v_target);
  }
  else if (String(topic) == String(TOPIC_EC_ADJUST)) {
    ec_adjust = (payload[0] == 't') ? true : false;
    Serial.println(ec_adjust);
  }
  else if (String(topic) == String(TOPIC_EC_TARGET)) {
    ec_target = atof((char *)payload);
    Serial.println(ec_target);
  }
  else if (String(topic) == String(TOPIC_EC_MODE)) {
    ec_mode = (payload[0] == 't') ? true : false;
    Serial.println(ec_mode);
    if (ec_mode) {
      if ((ec < ec_target) && (0.4 < ec)) { // if EC value is too low (less than 0.4)
something wrong
       digitalWrite(AIR_CTR, HIGH);         // When ec mode is swiched from off -> on,
adjust immediatly
       mqtt.publish(TOPIC_AIR, "0");
       mqtt.publish(TOPIC_AIR, "1");
       AIR_state = 1;
       Serial.println("air pump on");
       last_AIR = millis();
      }
    }
  }
}

/****************************** Setup ******************************/
void setup() {
  Serial.begin(115200);

  adc_power_on();                                          // ADC enable
  adc_gpio_init(ADC_UNIT_1, ADC_CHANNEL_6);                    // ADC1_CH6 (GPIO34 )enable
  adc_gpio_init(ADC_UNIT_1, ADC_CHANNEL_7);                    // ADC1_CH7 (GPIO35 )enable
  adc1_config_width(ADC_WIDTH_BIT_12);     // ADC1 12bit (0-4095)
  adc1_config_channel_atten(ADC1_CHANNEL_6, ADC_ATTEN_DB_11);   // ADC1_CH6 -11dB
  adc1_config_channel_atten(ADC1_CHANNEL_7, ADC_ATTEN_DB_11);   // ADC1_CH7 -11dB
  esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,  ADC_WIDTH_BIT_12, 1100,
&adcChar); // set parameters to addChar

  pinMode(STA_MODE,  INPUT_PULLUP);
  pinMode(VULVE_CTR, OUTPUT);
  pinMode(AIR_CTR,   OUTPUT);
  pinMode(PIN_EC,    INPUT);
  pinMode(PIN_WTEMP, INPUT);
  pinMode(PIN_VULVE, INPUT);
  pinMode(PIN_PWM,   OUTPUT);
  pinMode(POUT_WLSW, OUTPUT);
  pinMode(POUT_LED,  OUTPUT);
```

```
  Wire.begin(21, 22);
  Wire.endTransmission();

  ledcSetup(0, 5000, 8);         // Set PWM frequiency 5KHz, 8bit
  ledcAttachPin(PIN_PWM, 0);     // Assing PWM pin to 15

#ifdef SHTC3_SENSOR
  shtc3 = Adafruit_SHTC3();
  if (! shtc3.begin())
    Serial.println("Couldn't find SHTC3");
#endif

  APmode   = !digitalRead(STA_MODE);
  setup_MQTT(APmode);

  digitalWrite(VULVE_CTR, LOW);  // vulve off
  digitalWrite(AIR_CTR, LOW);    // Stop AIR air pump for EC control

  ////// MQTT topic initialization
  create_topic("ec",       TOPIC_EC);
#ifdef intWL_SENSOR
  create_topic("wlevel",    TOPIC_WLEVEL);
#else
  create_topic("wlevelX",   TOPIC_WLEVEL);
#endif
  create_topic("wtemp",     TOPIC_WTEMP);
  create_topic("temp",      TOPIC_TEMP);
  create_topic("humid",     TOPIC_HUMID);
  create_topic("vulve",     TOPIC_VULVE);
  create_topic("v_mode",    TOPIC_V_MODE);
  create_topic("v_target",  TOPIC_V_TARGET);
  create_topic("air",       TOPIC_AIR);
  create_topic("ec_adjust", TOPIC_EC_ADJUST);
  create_topic("ec_mode",   TOPIC_EC_MODE);
  create_topic("ec_target", TOPIC_EC_TARGET);
  create_topic("dcheck",    TOPIC_DCHECK);
  create_topic("network",   TOPIC_NETWORK);

  ec_magni = (EEPROM.read(146)) / 100.0;   // read ec adjustment magnification value
  Serial.printf("EC magnification vlue: %3.2f¥n", ec_magni);
  if (ec_magni < 0.2)                   // magnification might be wrong
    ec_magni = 1.0;
}

bool LED = false;

void loop() {
  if (APmode)
    AP_start();
  else if (WiFiMQTT_reconnect()) {
    if (!LED) {
      digitalWrite(POUT_LED, HIGH);
      LED = true;
#ifndef intWL_SENSOR
      mqtt.subscribe(TOPIC_WLEVEL);
#endif
      mqtt.subscribe(TOPIC_V_MODE);
      mqtt.subscribe(TOPIC_V_TARGET);
      mqtt.subscribe(TOPIC_EC_ADJUST);
      mqtt.subscribe(TOPIC_EC_MODE);
      mqtt.subscribe(TOPIC_EC_TARGET);
      mqtt.subscribe(TOPIC_DCHECK);
    }
    mqtt.loop();
    sensor();
  }
  else if (LED) {
    digitalWrite(POUT_LED, LOW);
```

```
    LED = false;
  }

  delay(100);
  vulve_control();
}
```

リスト 6.3.2　蠕動ポンプを制御するスケッチ

```
// File Name  : Soramame_PEPINO_PersitalicPump.ino
// Last Update: 2024-01-10
#include <PubSubClient.h>
#include <EEPROM.h>
#include "myMQTT.h"

bool APmode   = false;     // true:APmode  false:STAmode
bool NormalOP = true;
bool PowerUp  = true;      // turned to false after powerup initializiation

// Pin Assign
#define POUT_LED     2   // LED
#define AIR_CTR      4   // air pump on:1 / off:0
#define STA_MODE    18   // AP mode:0 / STA mode:1

/***************************** Topic *********************************/
char TOPIC_DCHECK[128];    // Data check
char TOPIC_EC_MODE[128];   // EC control mode <on:1 / off:0>
char TOPIC_EC_TARGET[128]; // Target EC value for auto mode
char TOPIC_EC[128];       // Calcurated EC from ECADC and WTEMP
char TOPIC_AIR[128];      // AIR air pump state  <on:1 / off:0>
char TOPIC_NETWORK[128];   // Network connection check data for every 60s

float ec       = 0;      // Electro Conductivity
bool  ec_mode   = false;   // 0: off  1: auto & adjust
float ec_target = 1.0;     // default EC target

#define INTERVAL_WIFI   120    // Connect to WiFi every 120s when they are disconnected.
#define INTERVAL_MQTT   120    // Connect to WiFi and MQTT every 120s when they are
disconnected.
#define INTERVAL_NETWORK  60

unsigned long last_NETWORK = 0;
unsigned long last_AIR    = 0;     // Last air pump on/of time
bool        AIR_state   = false;  // air pump state 0:off 1:on

bool MQTTconnectedX = false;

void sensor() {
  unsigned long now;

  /**** network connection check ****/
  now = millis();
  if (now - last_NETWORK > INTERVAL_NETWORK * 1000) {
    mqtt.publish(TOPIC_NETWORK, "1");
    last_NETWORK = now;
  }
}

void ec_control()
{
  unsigned long now = millis();

  // air vulve interval 15 minuites
  if (ec_mode && !AIR_state && ((now - last_AIR) > 15 * 60000) && (ec < ec_target) && (0.4
< ec) && mqtt.connected()) {
    digitalWrite(AIR_CTR, LOW);       // AIR air pump on
    mqtt.publish(TOPIC_AIR, "0");
    mqtt.publish(TOPIC_AIR, "1");
```

```
    AIR_state = true;
    Serial.println("air pump on");
    last_AIR = now;
  }
  // air pump off after 20 seconds
  else if (AIR_state && (now - last_AIR > 20 * 1000)) {
    digitalWrite(AIR_CTR, HIGH);       //air pump off
    mqtt.publish(TOPIC_AIR, "1");
    mqtt.publish(TOPIC_AIR, "0");
    AIR_state = 0;
    Serial.println("air pump off");
    last_AIR = now;
  }
}

/***********callback function work when message is arrived**********/
void callback(char* topic, byte* payload, unsigned int length) {
  payload[length] = '¥0';        // The last published string data is atteched at the
last of the subscribed string data. So, insert EOL.
  Serial.printf("Msg arrived [%s]¥n", topic);

  if (String(topic) == String(TOPIC_EC_MODE)) {
    ec_mode = (payload[0] == 't') ? true : false;
    Serial.println(ec_mode);
    if (ec_mode && !PowerUp) {
      if ((ec < ec_target) && (0.4 < ec)) { // if EC value is too low (less than 0.4)
something wrong
       digitalWrite(AIR_CTR, LOW);         // When ec mode is swiched from off -> on,
adjust immediatly
       mqtt.publish(TOPIC_AIR, "0");
       mqtt.publish(TOPIC_AIR, "1");
       AIR_state = 1;
       Serial.println("peristaltic pump on");
       last_AIR = millis();
      }
    }
    PowerUp = false;
  }
  else if (String(topic) == String(TOPIC_EC_TARGET)) {
    ec_target = atof((char *)payload);
    Serial.println(ec_target);
  }
  else if (String(topic) == String(TOPIC_EC)) {
    ec = atof((char *)payload);
    Serial.println(ec);
  }
}

/****************************** Setup ******************************/

void setup() {
  PowerUp = true;
  Serial.begin(115200);

  pinMode(POUT_LED, OUTPUT);
  pinMode(AIR_CTR,  OUTPUT);
  digitalWrite(AIR_CTR, HIGH);  // Stop AIR air pump for EC control

  pinMode(STA_MODE, INPUT_PULLUP);
  delay(50);
  APmode  = !digitalRead(STA_MODE);
  setup_MQTT(APmode);

  ////// MQTT topic initialization
  create_topic("ecX",      TOPIC_EC);
  create_topic("ec_mode",   TOPIC_EC_MODE);
  create_topic("ec_target", TOPIC_EC_TARGET);
  create_topic("air",      TOPIC_AIR);
```

```
  create_topic("dcheck",    TOPIC_DCHECK);
  create_topic("network",   TOPIC_NETWORK);
}

bool LED = false;

void loop() {
  if (APmode)
    AP_start();
  else if (WiFiMQTT_reconnect()) {
    if (!LED) {
      digitalWrite(POUT_LED, HIGH);
      LED = true;
      mqtt.subscribe(TOPIC_EC);
      mqtt.subscribe(TOPIC_EC_MODE);
      mqtt.subscribe(TOPIC_EC_TARGET);
      mqtt.subscribe(TOPIC_DCHECK);
    }
    mqtt.loop();
    sensor();
  }
  else if (LED)
    LED = false;

  delay(100);
  ec_control();
}
```

6.4　大規模栽培施設の検討

本事業は個人参加による地域コミュニティのレベルでの楽しむ農業をメインとして進めているが、農産物の生産を目的に、シンガポールで事業化している図 6.1.2 の装置の導入も当初検討していた。同国では大規模屋上施設を複数運営しており、図 6.4.1 は広さ 3,000 平米の施設の全景である。ここでは 10m の長さに 14 個のポットを備えたパイプを 12 本立てたものを基本とし、それを複数並べている。タイマによる自動潅水は行っているが、ほとんどは人手で運用している。そこでこれを 2 セット輸入してセンサを実装し、三鷹市内で生産のための試験運用を考えていたが、設置可能な場所を確保することができなかった。さらに本体は $1,700/セットであるがこれに関税や輸送コストがかかり、急激な円安により 30 万円を超えるに至り導入を断念した。しかしながら農作物生産の水耕栽培施設の将来的な運営も考慮し、上記システムの開発進めた。そして国内での大型装置の製造の検討も行ったので、それを以下に述べる。

図 6.4.1　シンガポールの水耕栽培施設全景

図 6.4.2 は国内で調達できる水耕栽培「装置本体の部材と、それを用いた装置を側面から見た図である。図 6.4.3 のように150cm の塩ビパイプ VU100 (外径 11.4cm)に 16 個のポットを挿せるよう、15cm 間隔で 90° ずらしながら 16 個の方受けエルボ CU45KL-50 を付けていく。長さ 4m のパイプ VU125(外径 14cm)二本を VU ソケット VU-DS125 でつないで 8m とし、そこに 40cm 間隔で直径 6.5cm の穴を 10 個開ける。そこにインクリーザ VU-IN100X50 を挿して、その上に図 6.4.3 のパイプを立てる。図面には記していないが、パイプ上部は図 6.3.4 のよ

うに雨樋用の金具で固定する。

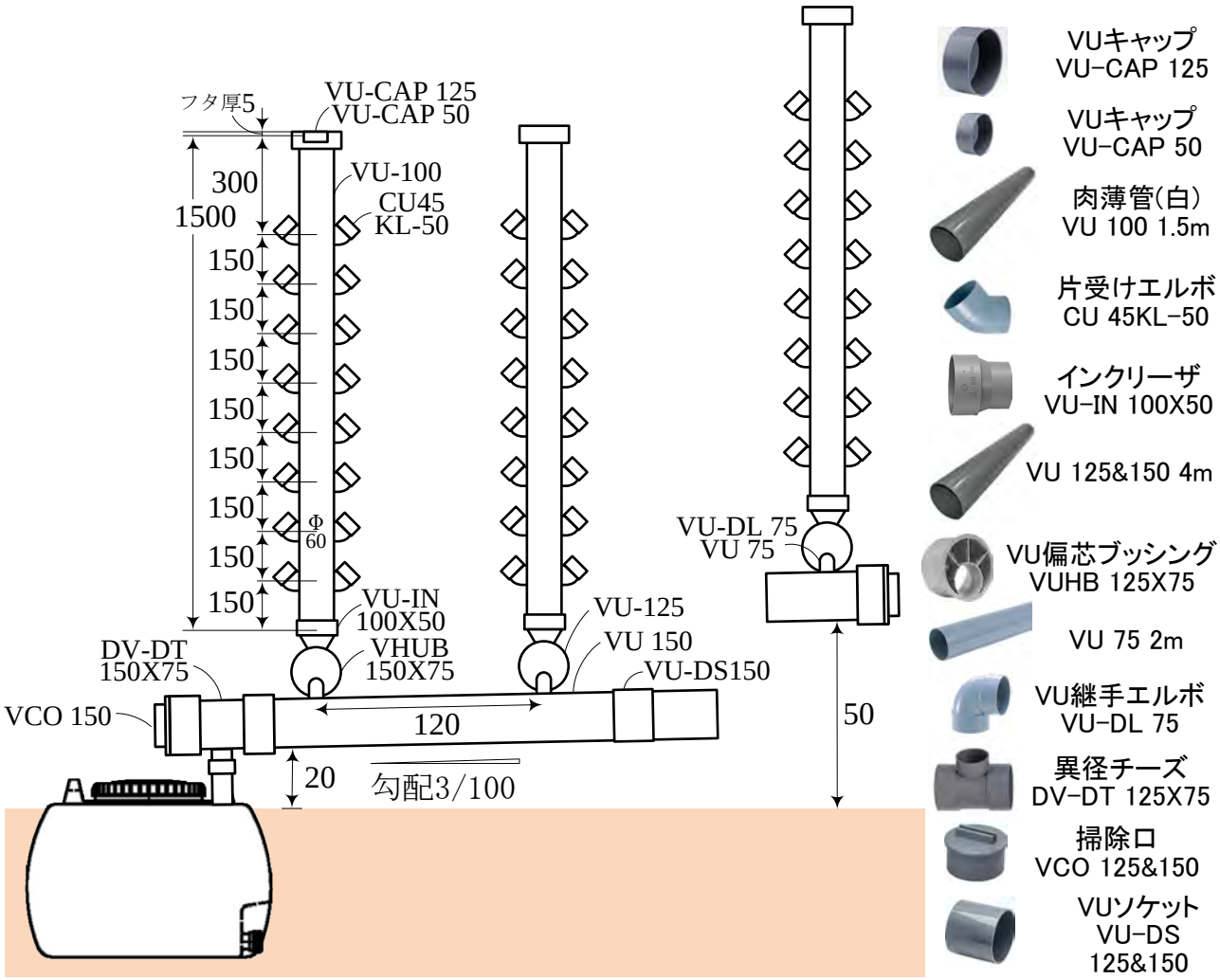


図 6.4.2　大規模生産用水耕栽培装置

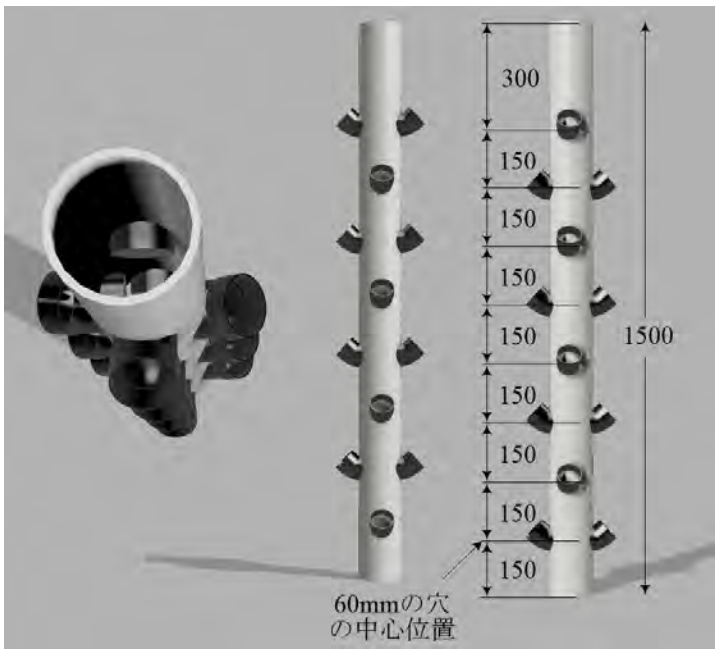


図 6.4.3　パイプの構造

図 6.4.4　パイプの固定

パイプ上部のキャップ VU-CAP125 には直径 7mm の二つの穴を空けて、図 6.4.5 左のように潅水用のチューブを挿入する. さらに図 6.4.5 右のように、6 個の穴を空けた VU-CAP50 をかぶせてボルトとナットで固定する。ここに注水された液肥はシャワーとなって、下のポットに挿った苗の根にかけられる。

上記 8m の塩ビパイプ VU100 の端には、図 6.4.6 のように偏芯ブッシング VUHB125×75 と塩ビ管 VU75、継手エルボ VU-DL75 等が接着されており、上から降ってきた液肥はここで回収されて塩ビ管 VU150（直径 16.5cm）に流れ込む。VU150 は 3/100 の勾配をつけており、最終的に液肥は地中に埋め込んだタンクに回収される。なお、VU100 を立てている 8m の塩ビ管 VU125 には勾配をつけない。これは上部に這わせる注水用の塩ビ管 VP20 が注水の向きとは逆の昇りとなってしまわないためである。注水された液肥は数本の VU125 に分かれて流れるがタンクに戻る VU150 は一本なので、ここに液肥が滞留すると溢れる恐れもあるため勾配を付した。

図 6.4.5　シャワー用キャップ

図 6.4.7 は 8m の塩ビパイプ VU125 に 10 本のパイプ VU100 を立てた構成を 1 セットとして、それを 120cm ピッチで 10 セット並べた施設の俯瞰図である。塩ビパイプを固定する単管パイプは薄い水色で示している。2m の単管 50 本を柱として、8m に繋いだ単管 10 本と、11m に繋いだ単管 5 本を直交クランプ 100 個で接続して全体のフレームを組んでいる。なお、図 6.4.7 では見やすさのために単管の位置をずらしている。例えば横に渡している 8m の単管は、立てたパイプ VU100 や図に示していないが注水用パイプ VP20 を固定するために、VU100 の上になるように配置する。

図 6.4.6　偏芯ブッシングの接着

50 本の柱の下部には、30cm にカットした単管 100 本を直交クランプや直交三連クランプで固定し、その上に液肥回収用の塩ビ管 VU125 と VU150 を載せる。VU125 と VU150 で液肥が流れない端には、掃除口 VCO125 や VCO150 を、VU ソケット VU-DS125 や VU-DS150 を介して接着する。

電気系統は示していないが、4 セットの電磁弁とタイマで注水の制御を行う。ポンプは一台とし、電磁弁が空いて水圧が下がったことを検知して動作するタイプを用いることで、タイマによる制御を不要とする。

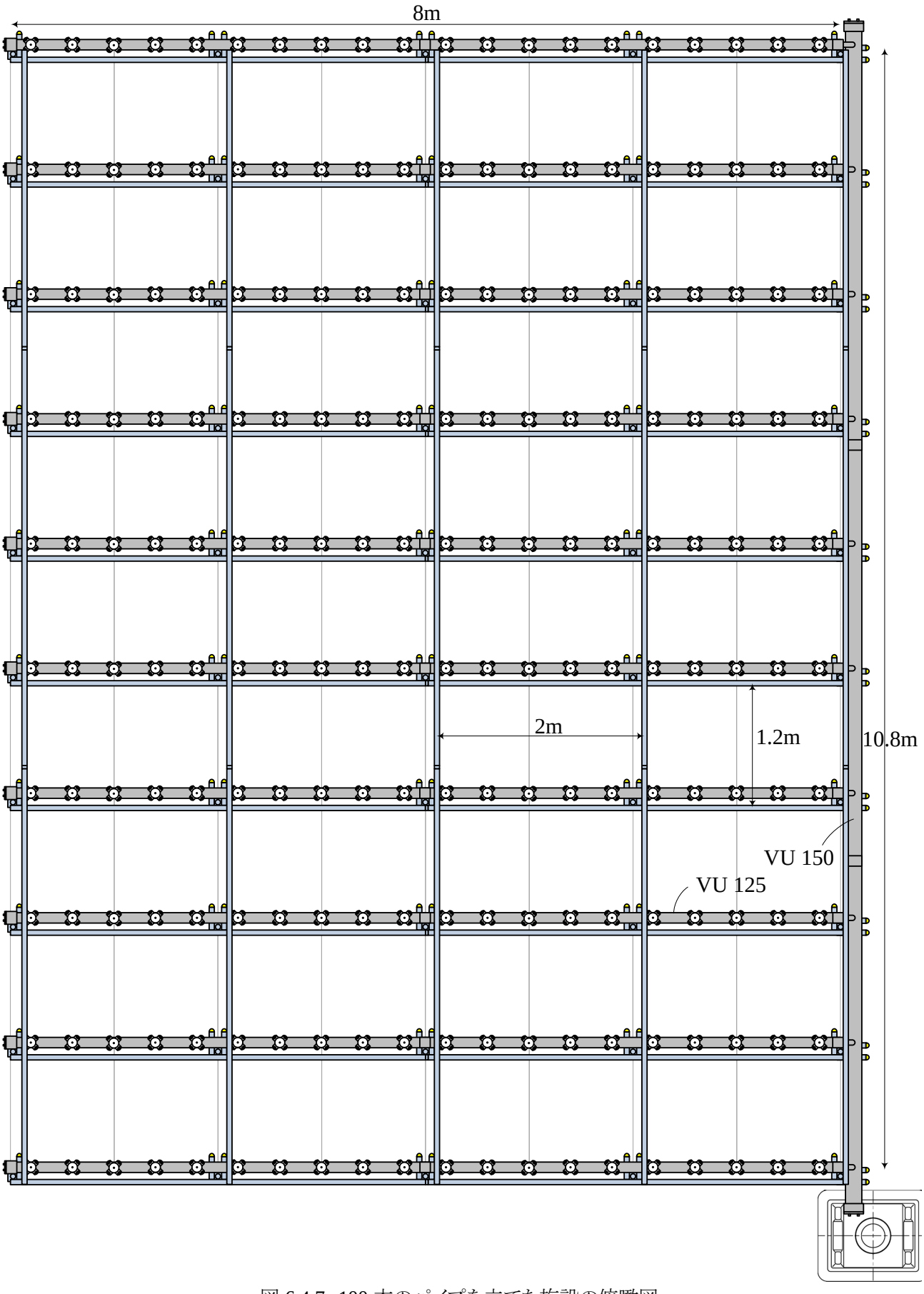


図 6.4.7　100 本のパイプを立てた施設の俯瞰図

表 6.4.1 に、図 6.4.7 の施設の主なパーツリストを示す。栽培できるポットの数は、16 個×100 本＝1,600 個である。他にも細かいパーツはあるが、このリストを基に製作のための人件費は除いた全体の部材費だけを見積もると、およそ 140 万円となる。シンガポールの栽培パイプ 12 本のシステムを 10 セット用意した場合、14 個×120 本＝1,680 個のポットを挿すことができ、図 6.4.7 の施設と同規模となる。そのコストは 30 万円×10 セット=300 万円で、10 セットを並べて固定するための単管フレームや、タンク、ポンプ、タイマも別途必要である。ただし穴あけや接着といった作業は一切不要で、フレームに 12 本の栽培パイプが固定された状態で納品される。全体としては人件費を含めても国内製造のほうが 2~3 割安価となる見込みであるが、為替レートが円高に進んだ場合、そのメリットは低下する。しかしシンガポールのシステムは、コンテナで大きなロット数を輸入して在庫を持たなくてはならないことや、ユーザのリクエストに応じて構成を柔軟に変更するのが難しいことなどから、国内製造に大きなメリットがあると言える。

表 6.4.1 大規模水耕栽培施設の主要パーツリスト

用途	品名	個数
本体	塩ビVUキャップ VU-CAP125	100
	塩ビVUキャップ VU-CAP50	100
	塩ビ管 VU100 2m	100
	片受エルボ CU45KL-50	1,600
	インクリーザ VU-IN 100X50	100
	塩ビ管 VU125 4m	20
	VU偏芯ブッシング VUHB 125X75	10
	塩ビ管 VU 75 4m	1
	VU継手エルボ VU-DL 75	10
	VUソケット VU-DS 125	20
	掃除口 VCO 125	10
フレーム	配管支持金物 野島角清 100A	100
	単管パイプ ライト管 4m	35
	単管パイプ ライト管 3m	10
	単管パイプ ライト管 2m	50
	固定ベース	50
	3連クランプ 直行	40
	直行クランプ	120
	単管C型ジョイント	317
	単管キャップ	200
電気系	プラボックス 日東工業 P12-1525A	4
	給水ポンプ エバラ 32HPE0.4S 100V単相	1
	電磁弁 ベン 桃太郎2 WS22-F 10A	4
	ツインタイマー OMRON H3CR-F	4
	安全ブレーカー パナソニックBS1112	4
	組端子台 極数4	4

用途	品名	個数
注水	塩ビ管 VU 150 4m	3
	VUソケット VU-DS 150	2
	掃除口 VCO 150	2
	異径チーズ DV-DT 150X75	1
	塩ビ管 VPW20 4M	45
	TSソケット TS-S 20	20
	TS継手 エルボ TS-L 20	20
	45°エルボ TS45L20	20
	HI継手 給水栓用ソケット HI-WS 20	20
	ホースニップル GHN(R)-0619	40
	塩ビ管 VPW50 4M	6
	VUソケット VU-DS 50	4
	TSキャップ TS-C-50	4
	タフダイン(青) 500G	1
	プッシュジョイント90° ホース径6mm	200
	ホース内径18mm外径23mm 30m	1
	ホースバンド	200
	ボルト・ナット	400
	スーパーローリータンク 800L SLT-800	1

6.5　ソーラパネルのサポート

大学の水耕栽培施設ではソーラ発電で 5 台の水耕栽培装置のセンサモジュールと DC12V ポンプを駆動し、夜間は余剰電力を蓄電したカーバッテリーを用いている。なお天候不良でバッテリーが上がってしまうトラブルに備えて、バックアップ用に商用 100V 電源で動作する AC ポンプをウィークリータイマも設置している。

令和元年の事業の小型水耕栽培装置には、図 6.5.1(左)の市販のマイコンボード WeMos D1 R32 を利用し、その上に各種センサを有するオリジナルの子基板をスタックした旧センサモジュールを用いていた。本事業では通信の信頼性向上、小型化、コスト低減等を目的に図 6.5.1(中)の新センサモジュールする PEPINO-HYDROPONICS を搭載した新型の水耕栽培装置を利用している。センサモジュールは図 6.5.1(右)に赤い楕円で示した新しい EC（液肥濃度）センサ回路も搭載した PEPINO-HYDROPONICS Rev.1 へと改良されている。

図 6.5.1　旧センサモジュール(左)と新モジュール PEPINO-HYDROPONICS (中)と Rev.1(右)

Rev.1 モジュールではセンサの電源を AC アダプタから供給しているときには特に問題なかったが、ソーラパネルを接続したカーバッテリーに接続すると、EC センサの出力電圧が研究室内の測定値よりも 1V 以上高く、その後、見る見る低下していくという不具合が生じた。その原因は、センサの電極間の電圧に影響を及ぼす液肥の電位であることが判明した。AC アダプタで商用電源から給電するときの GND レベルは自然界の GND レベルに一致し、基板の GND と液肥の間に電位差は見られなかった。しかしカーバッテリーの GND 電位は液肥よりも 5V 以上高くなっており、それに引っ張られた電極の電位を測定したため出力電圧が高くなっていた。また電極に見た目の変化がなくとも表面を拭くことで、電位が下がることが確認された。これは EC センサの停止中もセンサ電極と液肥間に大きな電位が生じているため、肥料が電極へ付着することが原因と考えられた。なお電極を拭かずに 1 日放置しておくと、肥料が白い結晶として析出していた。そこで基板（カーバッテリー）と液肥の GND 電位の差による直流電流が生じないように、電極と Rev.1 基板の間に図 6.5.2 のコンデンサを 2 つ挿入して測定実験を行ったところ、電極への肥料の析出が生じず、EC 測定が正常に行えるようになった。

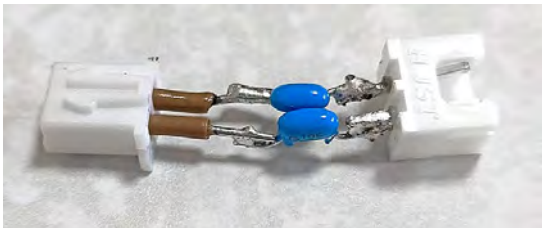


図 6.5.2　電極に接続したコンデンサ

この改良を行ったセンサモジュール Rev.2 の回路図を図 6.5.3 に示す。基板 GND と液肥間の電位差で測定値が引っ張られたり電流が流れたりしないように、図中に赤丸で示したように電極の出力と入力側の根元にそれぞれ 1μF のコンデンサ C23 と C24 を挿入した。

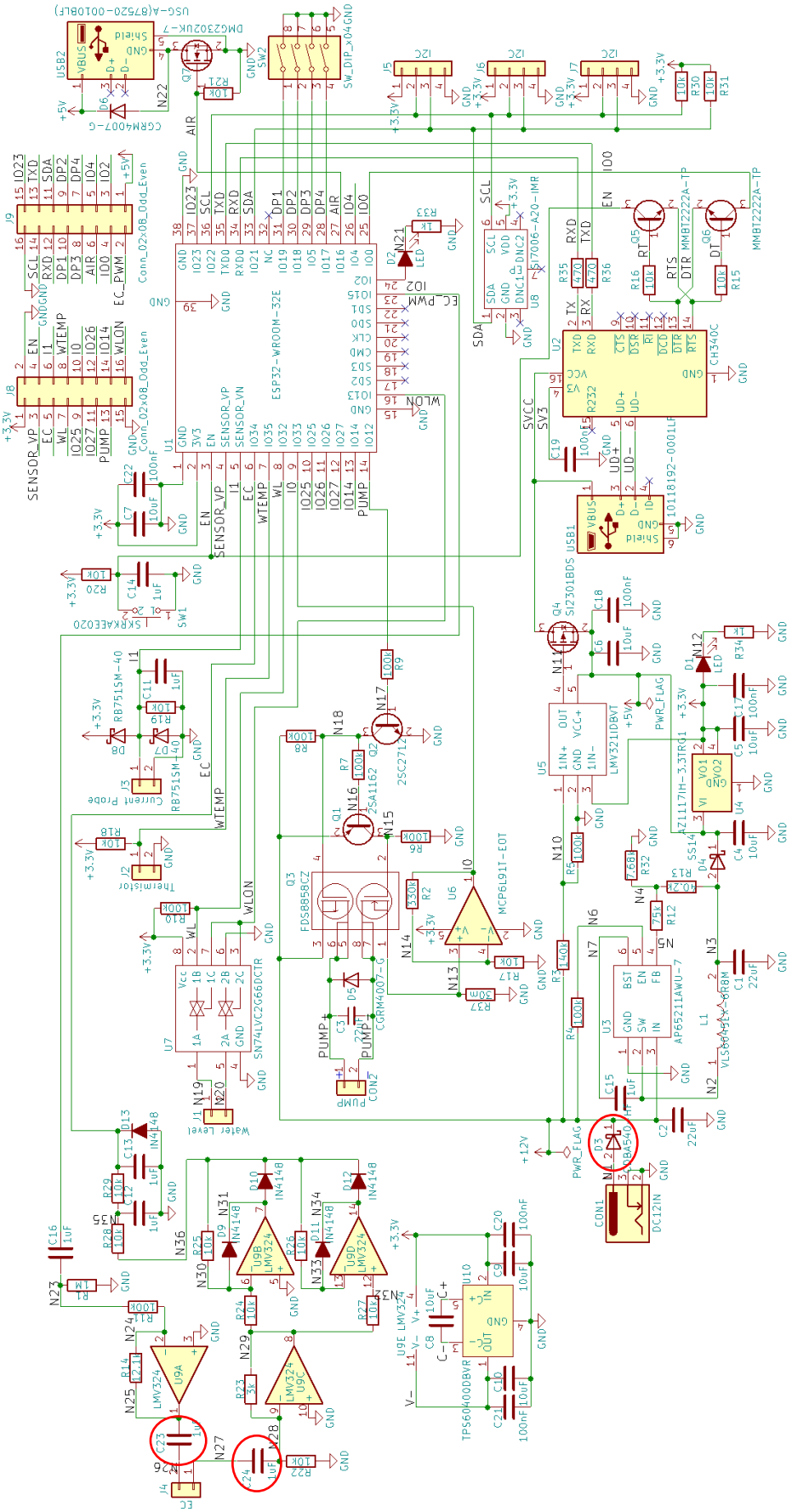


図 6.5.3　PEPINO-HYDROPONICS Rev.2 の回路図

旧型の装置で WeMos D1 R32 の子基板としてセンサモジュールを動かしていた時にはポンプ制御回路に問題は発生せず、3 年以上変更せずに同回路部品をもちいていた。しかし ESP32 マイコンとセンサ回路を 1 枚の電子基板に集約したモジュール PEPINO-HYDROPONICS を開発し、水耕栽培施設で動作テストを行ったところ、日が落ちるとポンプが次第に動かなくなり、最終的にはセンサも全て停止する不具合が生じた。昼間はソーラ発電で駆動され、夜間はカーバッテリーから給電されるため、バッテリーの劣化やソーラチャージコントローラの不具合が疑われた。しかし、それらを交換しても改善は見られなかった。チャージコントローラからの出力電圧は、晴天時のソーラパネル駆動時が 13.6V で、バッテリー駆動時は 12.6V と問題のない値であった。そこで、バッテリーだけを接続した状態で、ポンプの電源コードの電圧を測定したところ、ポンプを一台ずつ駆動する度に 0.5~1.0V 低下していき、4 台駆動時には 10V を切ってソーラチャージコントローラが動作を停止してしまった。

バッテリーの電流供給能力に問題があると考えられたため、2 台のバッテリーを並列に接続してみた。すると夜間にセンサモジュールが落ちることはなく、ポンプも日没後にすぐに止まることはなくなったが、5 時間ほどで全てのポンプが停止してしまった。そこで回路図を詳しく調べてみると、図 6.5.3 左に赤い丸で示した 12V 電源に挿入されたショットキーバリアダイオード D3 に 1A 品の SS14 を使用していたことがわかった。ポンプと基板を合わせて 3A 以上流す必要があるので、ここがまず問題であることがわかる。そこですべての基板のダイオードを 5A 品の CDBA540-HF に張替えて水耕栽培装置に再搭載した。

この改修により夜間にポンプが完全に止まることはなくなったが、やはりバッテリーの電力が消費されていくにしたがって停止時間が長くなっていった。図 6.5.4 はソーラパネル駆動時のポンプの制御信号とポンプの消費電流を、図 6.5.4 は夜間の信号と電流を示している。電流は 12 ビット AD コンバータの出力で、ここでは 2,000 がおよそ 1A である。図 6.5.4 左は 12V3A ポンプ SEAFLO 500GPH でおよそ 2A 前後（2A 以上は測定範囲を超えている）、図右は 5 台の水耕栽培装置のうち 1 台だけに使用している 12V1A ポンプ SEAFLO 350GPH (2A と記述をしているサイトもある)で1A弱流れていることがわかる。なお 3A ポンプと言っても常に 3A 流れるわけではなく、ポンプで汲み上げる水の量と揚程によって変動する。またポンプは 1 回に 15 秒間動かしているが測定は 5 秒間隔で行っているので、基本的に 2 回の測定のピーク値が示されることになる。これが夜になると図 6.5.5 のように、制御信号を出しても動かないことが生じ始める。また図 6.5.5 左の 3A ポンプは、ソーラパネルで駆動していたときよりも電流が少なくなっている。図右の 1A ポンプも、3A ポンプほど顕著ではないが低下している。この状況から、駆動電流もさることながら駆動電圧の影響が大きいように見える。

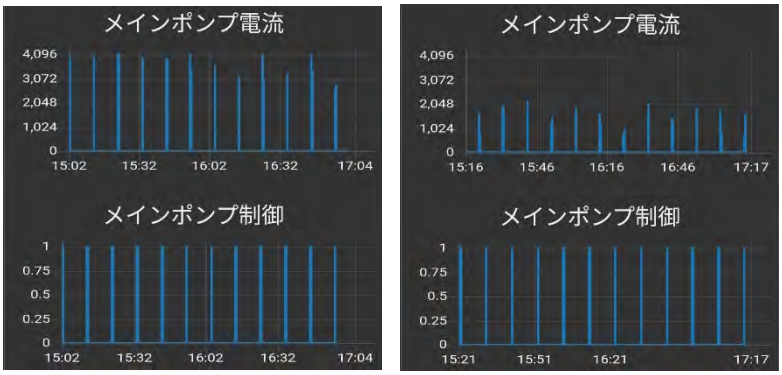


図 6.5.4 ソーラパネル駆動時のポンプの状態

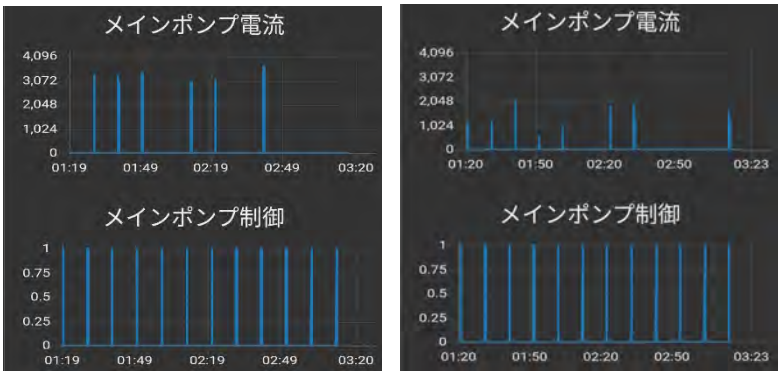


図 6.5.5 バッテリー駆動時のポンプの状態

マイコンボード WeMos D1 R32 とポンプ駆動用のドライバを実装したセンサ基板を分けていたときにはこのような不具合はなく、駆動回路に変更も加えていない。しか

し、WeMos D1 R32 とセンサ基板にそれぞれ 12V を供給するのに 2 分岐コネクタを用いており、センサ基板側には逆流防止用の図 6.5.3 のショットキーバリアダイオード D3 は入っていない。そこで、図 6.5.6 のように、PEPINO-HYDROPONICS Rev.1 のパターンをカットしてダイオードをバイパスするリワークを入れた。

図 6.5.6　Rev.1 基板のリワーク

リワークを施した Rev.1 基板を水耕栽培装置に実装したところ、夜間も問題なくポンプが動作するようになった。そこで PEPINO-HYDROPONICS Rev.3 の基板でも、図 6.5.7 のように同様の修正を行った。なお、ポンプの 12V 電源ラインからはダイオードをはずしたが、3.3V 系の電圧レギュレータへのラインは電圧低下の問題は生じないので、電源ノイズやポンプ停止時の逆起電力による逆流防止用としてダイオード D3 を入れている。また電源ライン強化のため、図 6.5.8 のようにレイアウト上のライン幅を太くした。

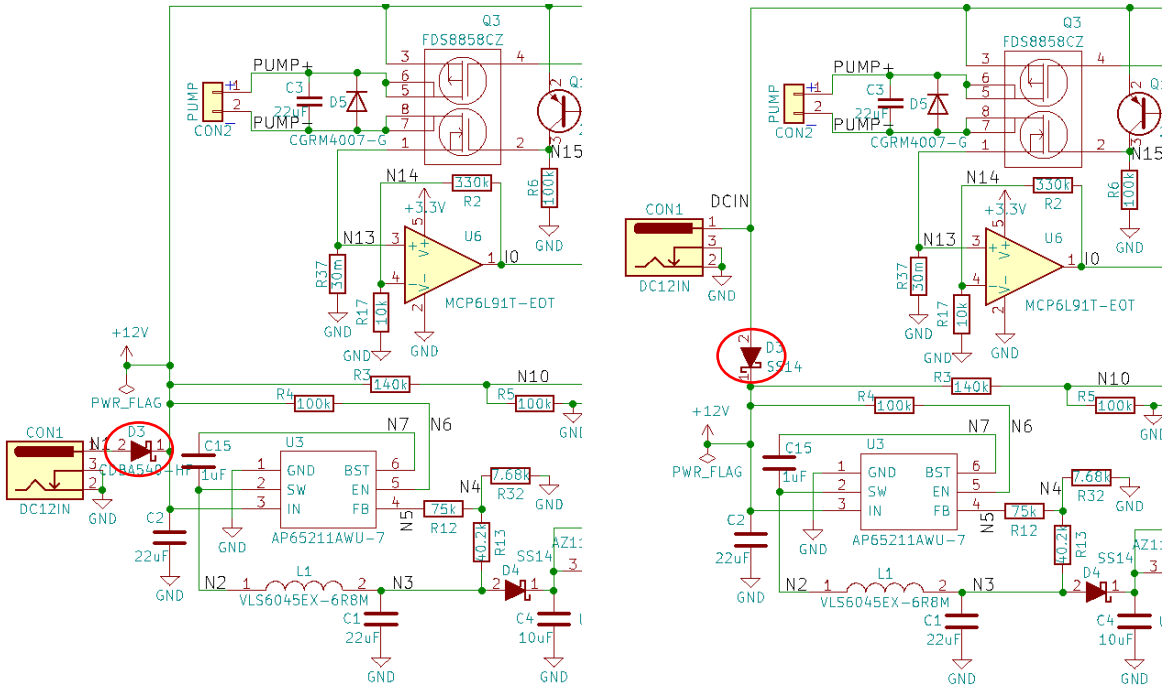


図 6.5.7　12 V 電源に対するショットキーバリアダイオード D3 の変更（左 Rev.2、右 Rev.3）

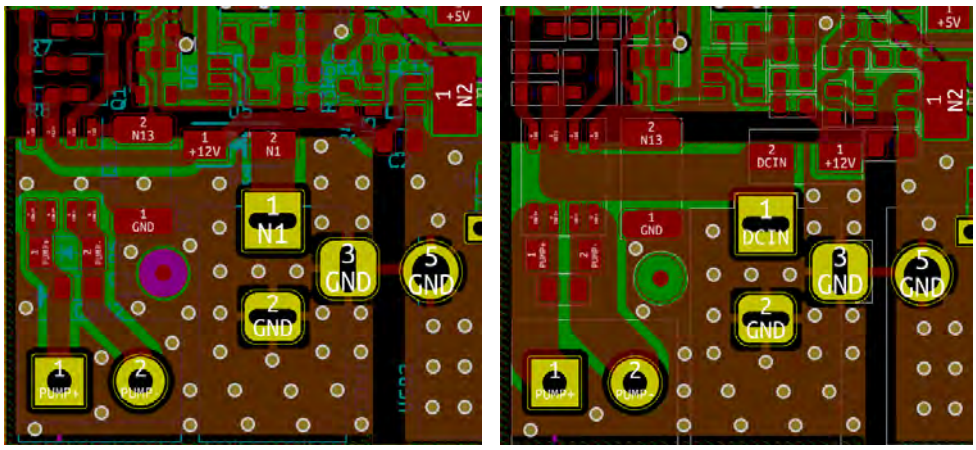


図 6.5.8　Rev.2(左)から Rev.3(右)への DC12V ラインの変更

その後、Rev.3 と基本的な回路構成は同じものの、部品調達の問題から複数の回路変更をおこなっている。図 6.5.1 中央の最初の PEPINO-HYDROPONICS では、12V→5V レギュレータに WeMos D1 R32 が使用している Monolithic Power Systems 社の MP1482DS-LF-Z を採用した。しかしこのチップが新規設計不適合品（近い将来に廃番予定）となったため、Rev.1 で Diodes Incorporated 社の AP65211AWU-7 に変更した。ところが突然このチップが製造中止となり互換品もなかったため、それに近い Diodes Incorporated 社の AP62300TWU-7 を用いることとした。AP62300TWU-7 は入力電圧 4.2V～18V、出力電圧 0.8V～7V（但し降圧のみ）、最大出力電流 3A で、AP65211AWU-7 の入力電圧 4.7V～18V、出力電圧 0.8V～6V（但し降圧のみ）、最大出力電流 2A に比べて性能が向上している。リファレンスデザインでは、出力電圧設定用の抵抗の数が AP65211AWU-7 の 3 個から AP62300TWU-7 は 2 個に減っているが、出力電流が 2A から 3A に増えたことで、そこに接続するコンデンサが 22µF×1 個から 22µF×2 個に増えている。しかし、このレギュレータは USB インタフェースの 5V と ESP32 の 3.3V 電源（Diodes Incorporated 社の 5V→3.3V レギュレータ AZ1117IH-3.3TRG1 で生成）に使用されるだけで、1A も消費されない。そこで、22µF のコンデンサは 1 個とした。リファレンスデザインでは入力側のコンデンサが AP65211AWU-7 の 22µF から、AP62300TWU-7 に 10µF 変更されているが、部品の種類を減らすために 22µF（耐圧 18V の 0805 パッケージ）を使用することとした。なお、10µF のコンデンサは他にも使用しているが、耐圧 6.3V の 0603 パッケージであり、耐圧 18V は 0805 パッケージとなる。10µF を 0603 から大きな 0805 に統一しようとするとレイアウトが大幅変更となり、また部品コストも増加するので好ましくない。22µF はもともと 0805 で 0603 パッケージはない。

その後、さらにレギュレータ AP62300TWU-7 も入手不可能となり、コロナ禍の半導体不足の影響でその他の部品も 1/3 が在庫切れとなり互換品や類似品を用いて再設計を行った。特に、レギュレータ周辺のレイアウトが大幅変更となっている。I2C インタフェースの温湿度センサ SI7006-A20-IMR（U2）は類似製品を含めてパーツがまったく見つからなかった。屋外での使用では湿度センサの耐久性に問題があり、そもそも外気の温湿度というよりも、センサモジュールを入れる水耕栽培装置のケースの中の温湿を計測することになるため、実装は行わないこととした。なお、フットプリントのパターンだけは基板上に残した。

この設計を製造発注する段階で、5V レギュレータ RT8299AZSP と-3.3V レギュレータ TPS60400DBVR が在庫不足を起こしてしまった。そこで設計時点で在庫が豊富なレギュレータ TPS562201DDCR と LM2776DBVR に変更した。これらの修正を行った Rev.4 の基板とアートワークを図 6.5.9 に、回路図を 6.4.10 に示す。

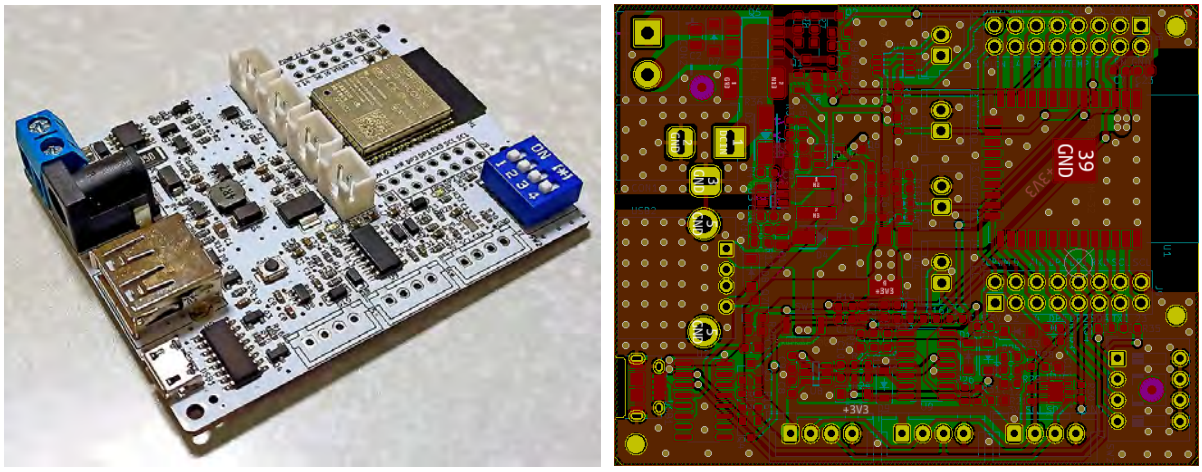


図 6.5.9 PEPINO-HYDROPONICS Rev.4 基板とアートワーク

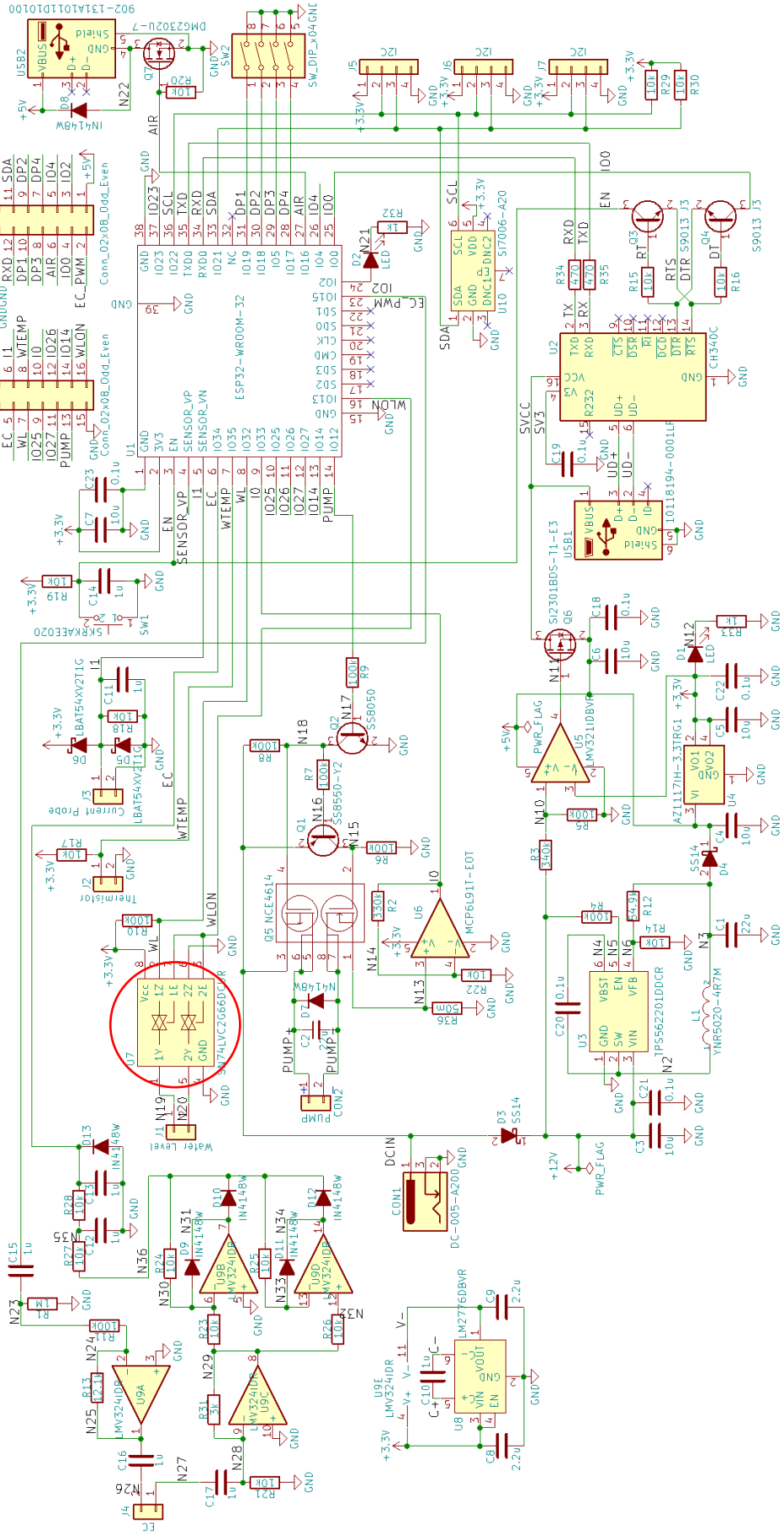


図 6.5.10　PEPINO-H YDROPONICS Rev.4 の回路図

Rev.4 の基板製造と部品実装は中国の JLCPCB 社で行った。ESP32 のアンテナ部を基板の上部エッジにそろえたところ、図 6.5.11 に示したようにアンテナ部がドリルカッターで削られてしまった。基板製造時に既にエッジカットが浅く入っているが、部品実装後のドリルカッターによる切断時には、このエッジよりも少し内側に入ってきている。それによってアンテナ部分が 7mm ほど削られ、それを検知したのかドリルが少し外に逃げてまた内側に入って来るということが行われている。これまではカットされた基板に後から部品を実装していたので問題なかったが、実装時の部品のずれもあることから、上面ぴったりではなく 0.5mm 程度内側に配置する必要がある。

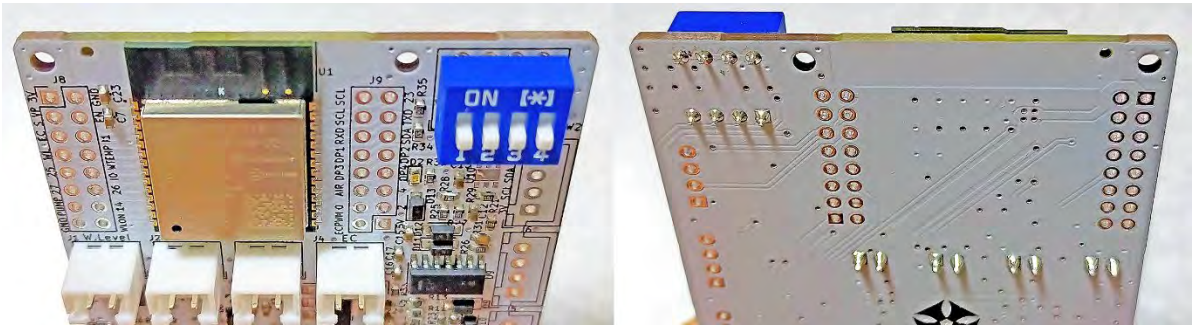


図 6.5.11　基板のエッジカッティングの問題

ところで水位測定には電極先端が液肥に浸かっているか否かを、電極に電流を流して調べているが、測定時以外に流れるのを避ける目的で、図 6.5.10 のように電極との間にトランスファゲート U7 を入れている。Rev.3 ではピンピッチ 0.65mm の SSOP-8 パッケージの SN74LVC2G66DCTR を用いていたが、JLCPCB で扱いがなかったため Rev.4 ではピンピッチ 0.5mm の VSSOP-8 パッケージの SN74LVC2G66DCUR を用いた。センサ基板の部品で 0.5mm ピッチは最小で、かつこの IC だけであるが、製造した 5 枚のうち 2 枚が図 6.5.12 のように下 4 本のピンがショートするトラブルがあった。基板のピン間にレジストがきちんと載っていない可能性もある。JLCPCB ではこれよりもピン間の広い 2 チャンネルの双方向スイッチを扱っていないため、Rev.5 では 4 チャンネルのピンピッチ 1.27mm の SOIC-14 パッケージの CD4066BM96 を用いた。

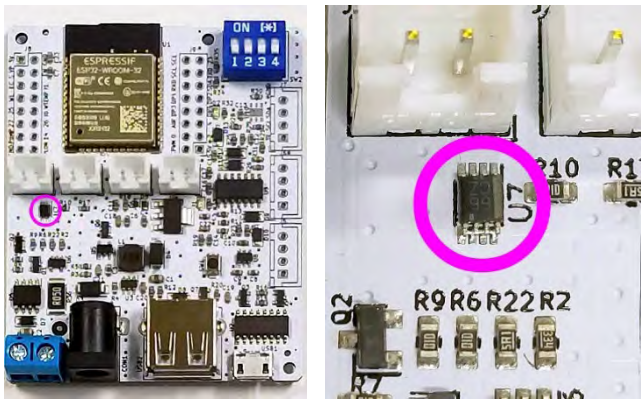


図 6.5.12　はんだのショート

図 6.5.13 に PEPINO-HYDROPONICS Rev.5 のイメージ図とアートワークを、図 6.5.14 に回路図を示す。Rev4 からの修正箇所は 4 点あり、まずショートを起こし

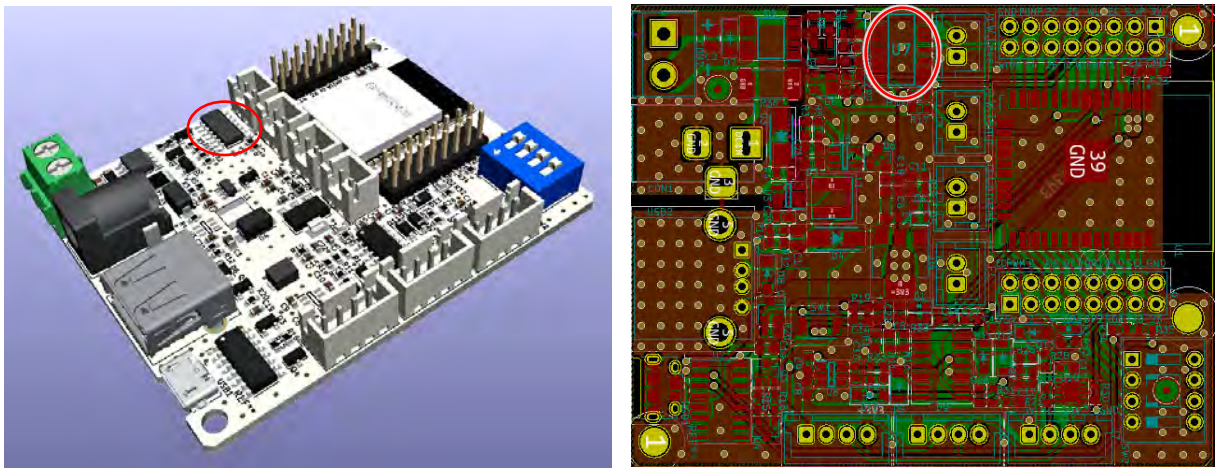


図 6.5.13　PEPINO-HYDROPONICS Rev.5 のイメージ図とアートワーク

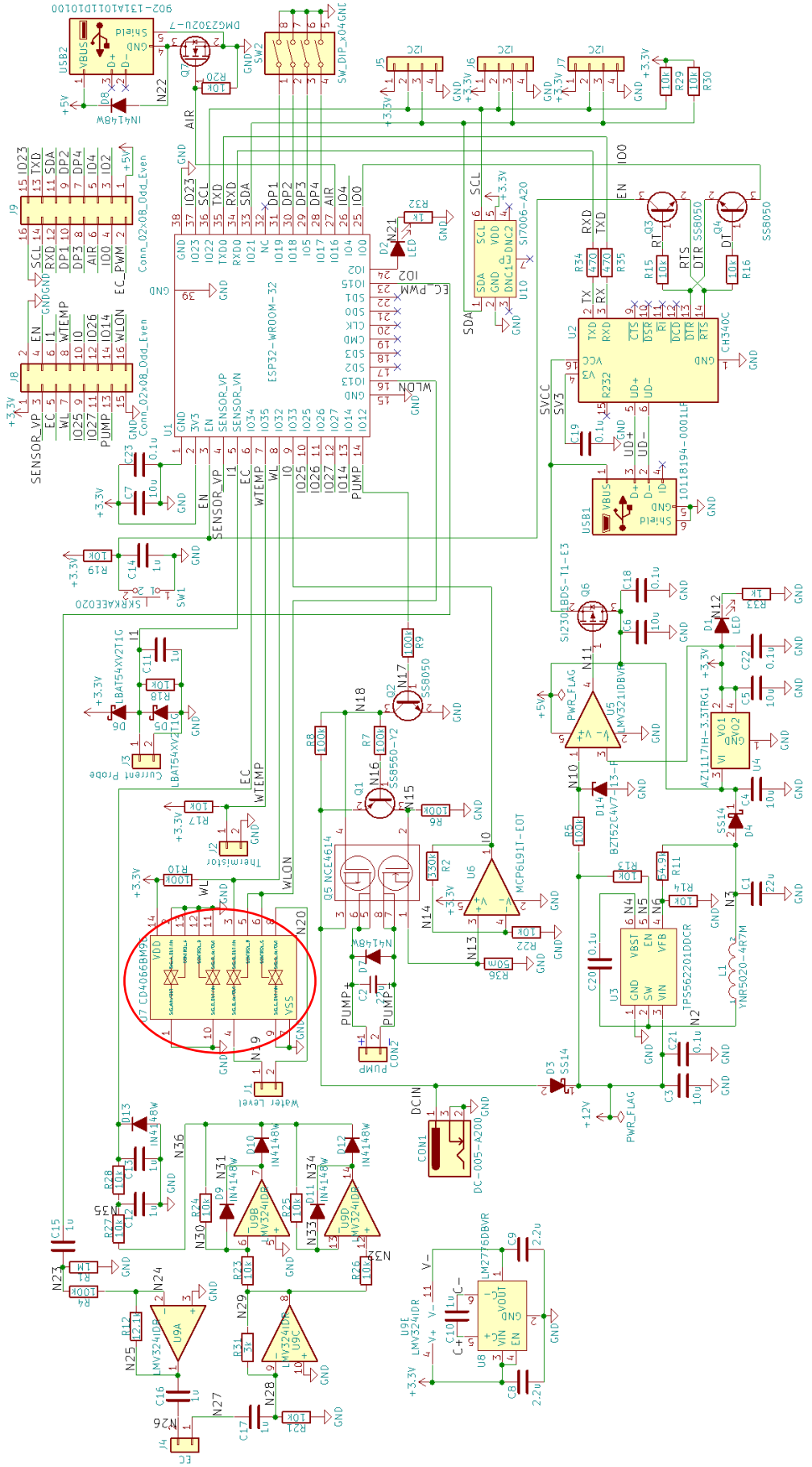


図 6.5.14　PEPINO-HYDROPONICS Rev.5 の回路図

たピッチ 0.5mm の SN74LVC2G66DCUR から 1.27mm ピッチの CD4066BM96 への変更である。図 6.5.13 および 6.4.14 では赤の枠で囲っている。4 チャンネルの内で必要なのは 2 つだけなので、残りの 2 チャンネルのピンは GND に落としている。

2 番目の変更点は、オペアンプ U5 への+入力へのツェナーダイオードの付加である。もともとは 12V 電源が入ったときに抵抗で分圧して 5V を作って入力していた。これによって 12V 電源が入った時にレギュレータ U3 で 5V が作られ、それに続いて U4 で 3.3V が作られると U5 の出力が 5V となり、PMOS Q6 をオフして USB 側からの 5V 電源供給が止まる。U5 の－入力を GND ではなく 3.3V から取っているのは、12V が供給されていないにもかかわらず 12V ラインにノイズが入って＋側に振れたときに、誤検知で PMOS オフにならないようにするためである。12V 入力はソーラコントローラから供給される場合があるが、安価なコントローラはバッテリーやソーラパネルの電圧をそのまま出力するものがある。そこでオペアンプの最大定格入力電圧を越えないように、ソーラパネルの最大出力電圧 22V が入った時に 5V となるように分圧の抵抗値を決めた。しかしこれでは 12V の入力時に 2.7V しか出力されず、PMOS がオフせずに 12V 側と USB 側の電源がぶつかってしまうことになる。そこでソーラパネル電圧を分圧して 5V を作るのをやめ、ショットキーダイオードの降伏電圧でオペアンプの+入力に 4.7V を供給するようにした。

3 番目の変更点は図 6.5.11 に示したように、基板のエッジにピッタリに配置した ESP32 のアンテナ部分がドリルで削られてしまわないように、ESP32 全体を 0.4mm 基板の内側に移動した。

4 番目は 3 ヶ所に開けたねじ止め用の穴である。水耕栽培装置のケースに固定しようとしたところ、精度の問題でケースのねじ穴をドリルで広げるなどの手間を要した。そこでねじ留めではなく、ケース側に突起を作ってそこにはめるという簡単な固定方法に変更し、穴を大きくした。

Rev.5 は正しく動作していたが、その後も複数のパーツが半導体不足で入手困難となり Rev.6 を作成した。図 6.5.15 に PEPINO-HYDROPONICS Rev.6 のイメージ図とアートワークを、図 6.5.16 に回路図を示す。回路図上の変更は、赤い枠でも示した 5 ヶ所である。1 つ目は EC センサのオペアンプに-3.3V を供給する Taxus Instrument 社の 200mA レギュレータ LM2776DBVR を、同社の 60mA レギュレータ TPS60403DBVR に変更した。なおピン互換ではない。2 つ目は在庫切れのために実装していなかった SILICON LABS 社の I2C 温湿度センサ SI7006-A20 の代わりに、SENSIRION 社の SHT-C3 を用いた。3 つ目は、江蘇沁恒股分有限公司の USB-シリアル変換チップ CH340C を CH340B に変更した。回路図上は CH340C のままであるが、BOM ファイルでは CH340B を指定している。ピンアサインがやや異なるが、本センサ基板の仕様ではそのまま載せることができる。

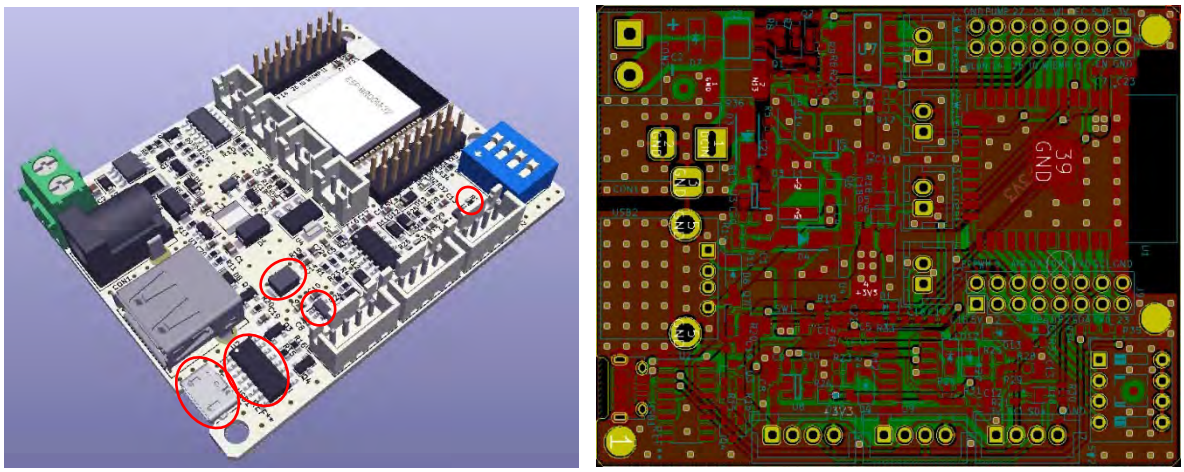


図 6.5.15　PEPINO-HYDROPONICS Rev.6 のイメージ図とアートワーク

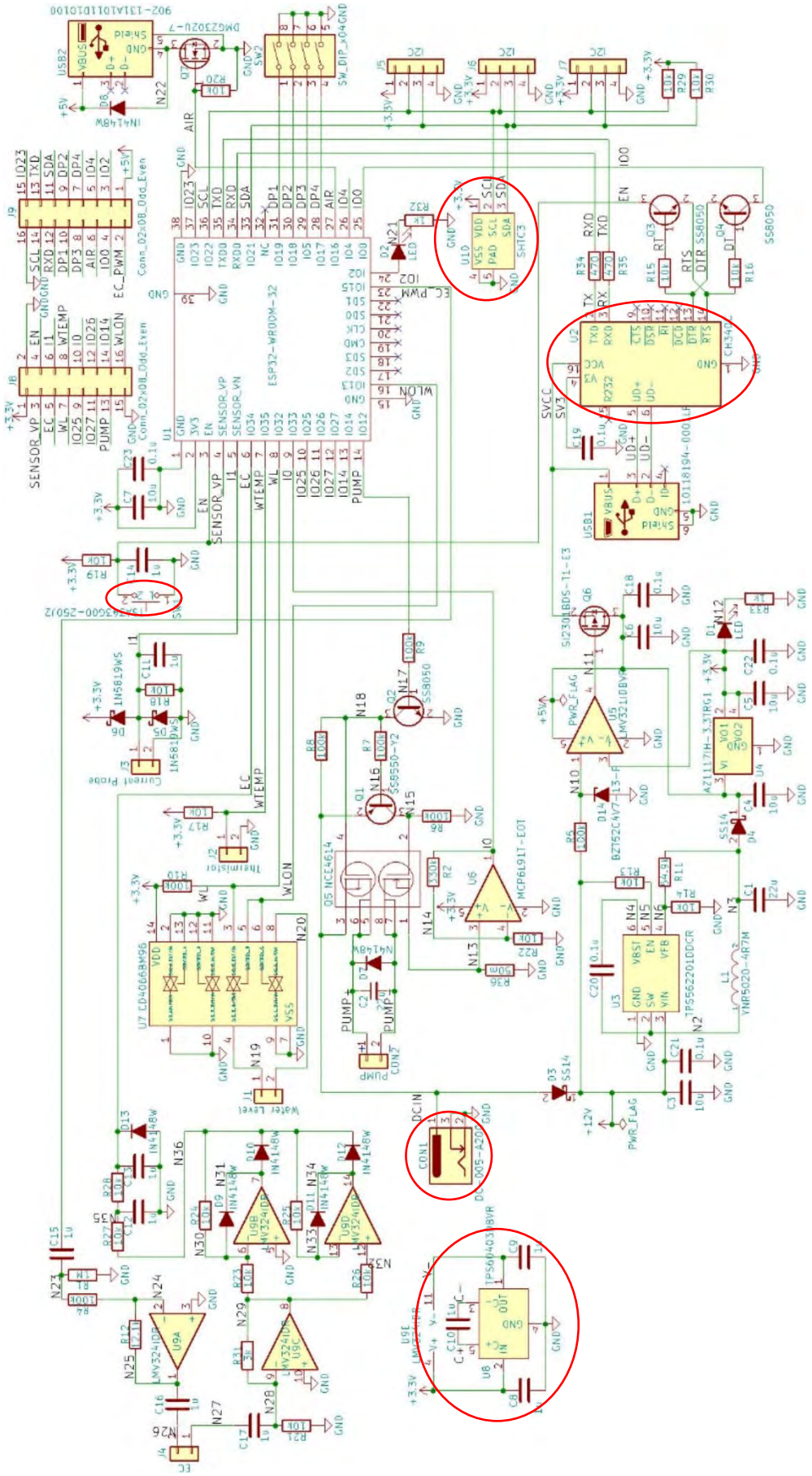


図 6.5.16　PEPINO-HYDROPONICS Rev.6 の回路図

4 つ目は、リセット用のタクトスイッチ SW1 を Alps Alpin 社の SKRKAEE020 から、形状とフットプリントが似ている BRIGHT 社の TSA343G00-250J2 に変更した。最後は Micro USB コネクタで、部品に変更はないが実装位置が基板外形にぴったりであったため、カットの邪魔にならないように少し内側に移動した。

Rev.6 を 12V AC アダプタと、5V USB の両方を接続して 2 時間ほど経ったときに、図 6.5.17 のように 12V-5V レギュレータ U3 が煙を出して焦げてしまった。回路を調べたところ、降伏電圧 Vz が 4.7V の D14 のツェナーダイオードが 12V を接続したときに 3.3V 前後の電圧しか出しておらず、USB 側の 5V→3.3V レギュレータ U4 の出力と比較して、どちらの電源を使うかを決めるコンパレータ U5 が誤動作していることが分かった。4.7V がちゃんと出ていれば、PMOS の Q6 がオフになるはずであるが、ダイオードの電圧が 3.3V を下回ると、Q6 がオフとなり、USB から供給される 5V と、12V-5V レギュレータ U3 が出力する 5V がショートすることになる。両方とも同じ 5V なのですぐには焦げなかったものの、2 時間の間、何かの拍子に大電流が流れたものと思われる。ダイオードの電圧が 3.3V にしかならなかったのは、スペックではツェナー試験電流定格 I_{ZT} が 5mA のところ、12V 電源との間に 100kΩ の抵抗を入れてしまったために、0.12mA しか流れていなかったためであった。そこで 2.2kΩ に変更しておよそ 5mA 流したところ、正しく 4.7V が出力され、電源ショートの問題が解決された。

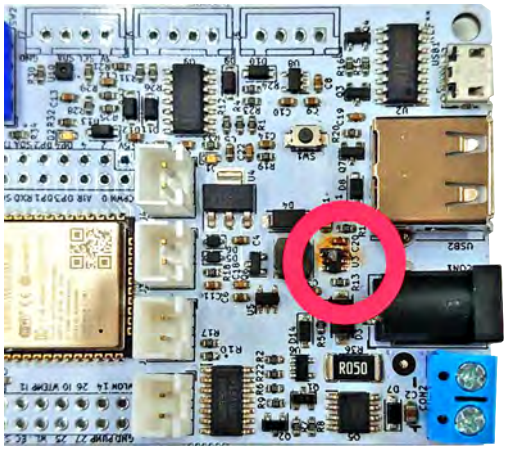


図 6.5.17 焼けた 12V-5V レギュレータ

Rev.6 の基板製造では図 6.5.18 のように 5mm×2mm の mouse bite により、1 枚の基板に 6 個のセンサの面付けを行った。センサ 1 枚のサイズは 56mm×71mm で、センサの周りの mouse bite のスペースは 2mm 左右の捨て基板の幅は 5mm、上下の捨て基板は 10mm、それらを合わせた全体の基板サイズは 168mm×186mm である。mouse bite は基板を折取るときにバリが出るが、並べた基板間にスペースを入れずに表裏からカッターで切り込みを入れる図 6.5.19 の V-cut はキレイに折り取ることができる。しかし JLCPCB 社は当初 V-cut 基板をサポートしていなかった。なお Rev.6 の mouse bite 基板製造時は実装費が無料で、製造するセンサ基板の数が同じであれば電子パーツのコストは同じであった。そのためセンサ基板 6 個付けの面付け基板を 5 枚製造したときと、センサ基板を単体で 30 個製造したときでは、基板 30-5=25 枚分の削減にしかならない。さらに面付けは追加費用がかかるので帳消しとなり、単体で製造した方が安くエッジにバリも生じない。しかし、単体では最大 50 個しか製造できないのに対して、面付けでは最大 6×50=300 個製造できるので量産時には有用である。Rev.6 注文時の送料を含んだ本センサ 1 個当たりの金額は、単体 30 個製造時で約 1,600 円、面付け 300 個製造時に約 1,000 円であった。

Rev.6 は当初 4MB の ESP32 を使用していたが、リモートでのファームウェア書き換え等の新しい機能を追加するにはメモリが不足していた。そこで 16MB の ESP32-WROOM-32E-N16 を用いることとした。この変更時には JLCPCB 社が V-cut をサポートしたので、センサ基板の 1 個の価格は多少高いものの面付けを試してみた。図 6.5.20 と図 4.6.21 に mouse bite と V-cut で製造された面付け基板を示す。V-cut 基板の裏面のロゴは図 6.5.22 のように、製品名に合わせて PEPINO から TOWER FARM に変更した。ガーバデータはそれ以外は変更しておらず、全ての機能が正しく動作していた。

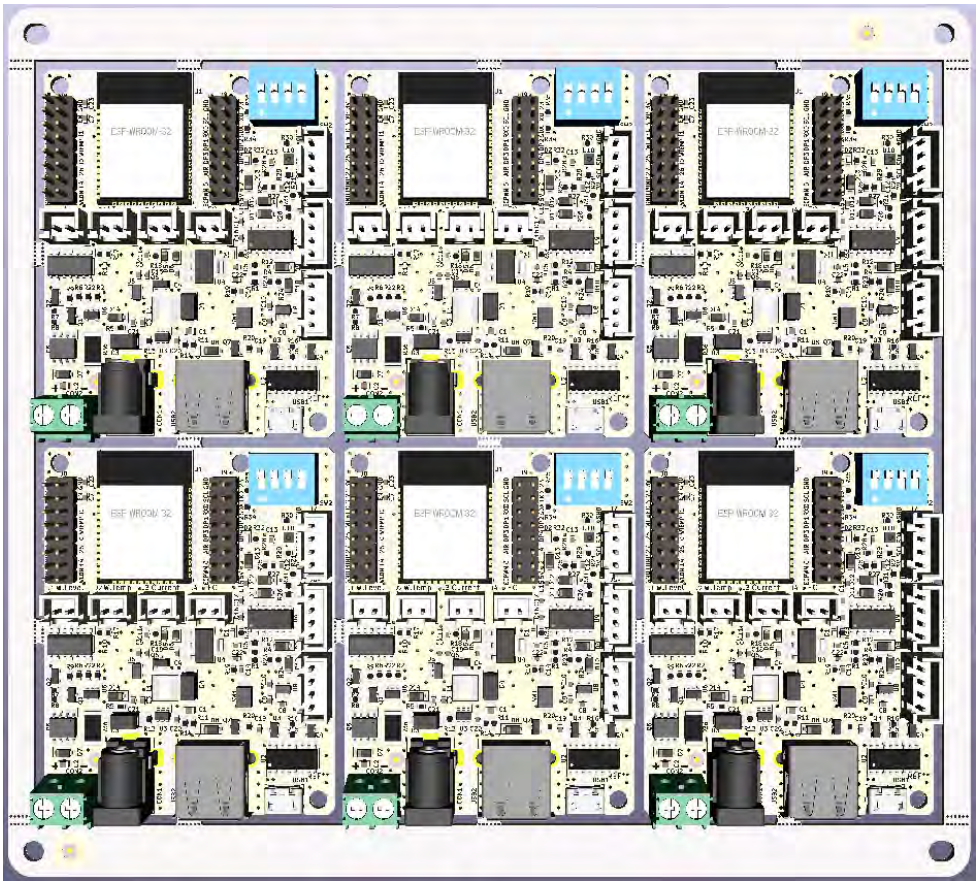


図 6.5.18　PEPINO-HYDROPONICS Rev.6 の mouse bite による面付け

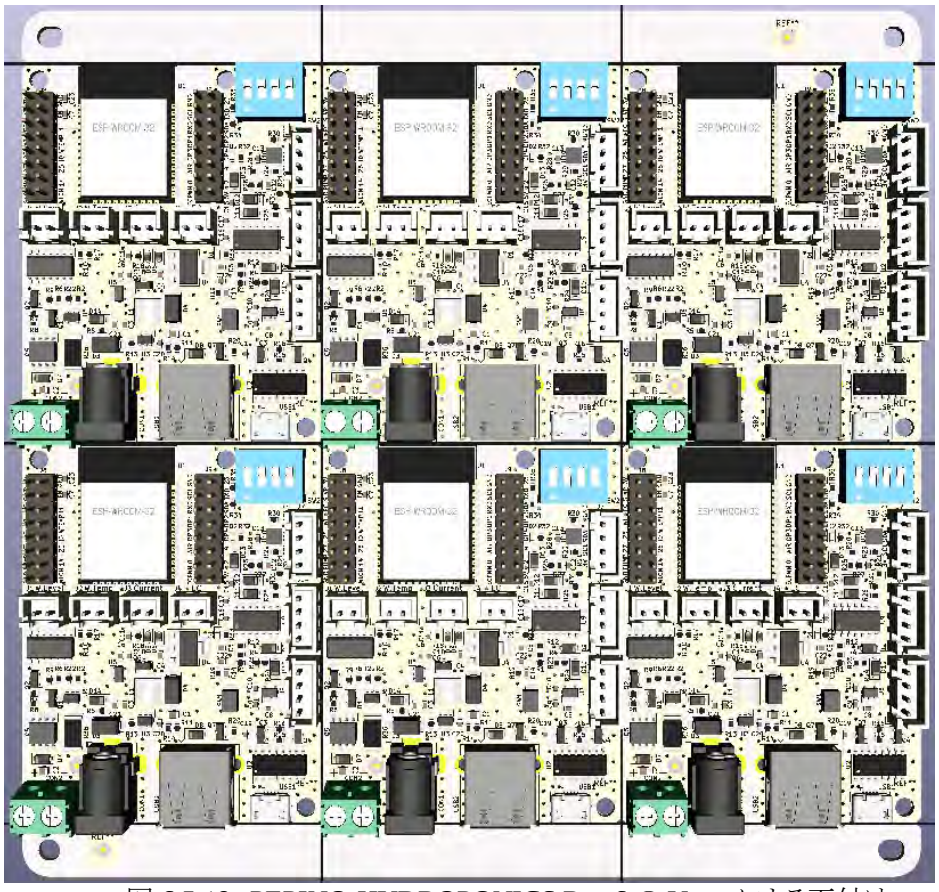


図 6.5.19　PEPINO-HYDROPONICS Rev.6 の V-cut による面付け

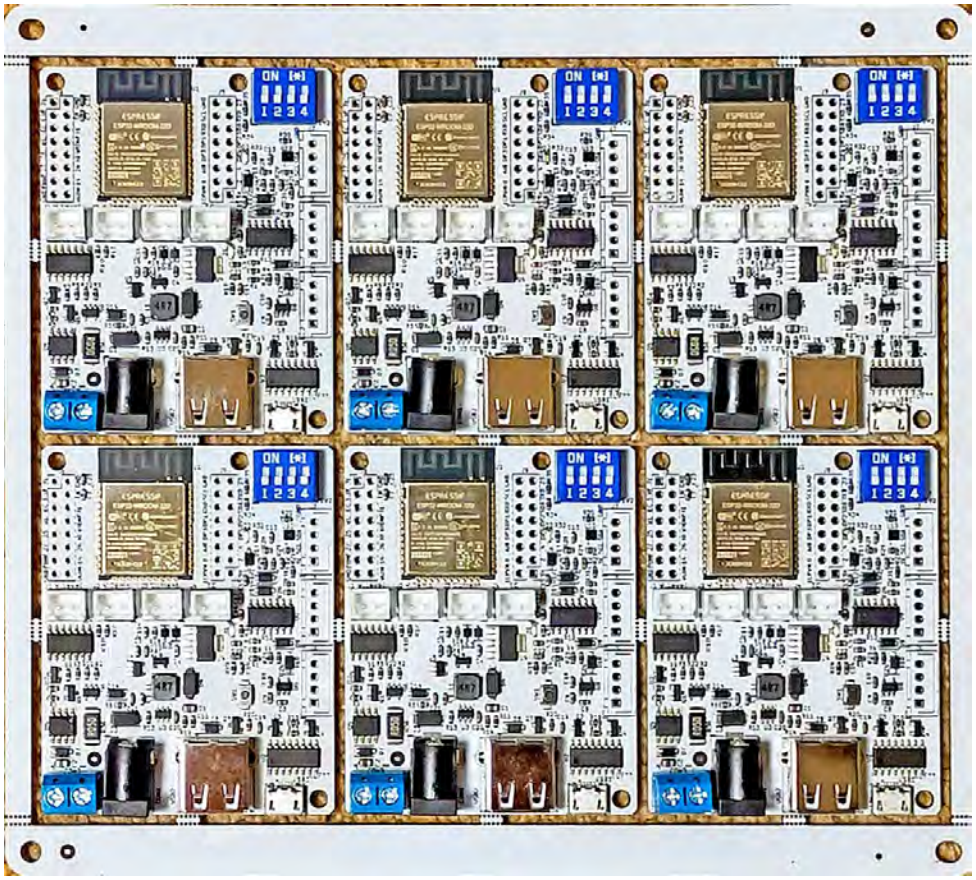


図 6.5.20　製造された mouse bite 面付け基板

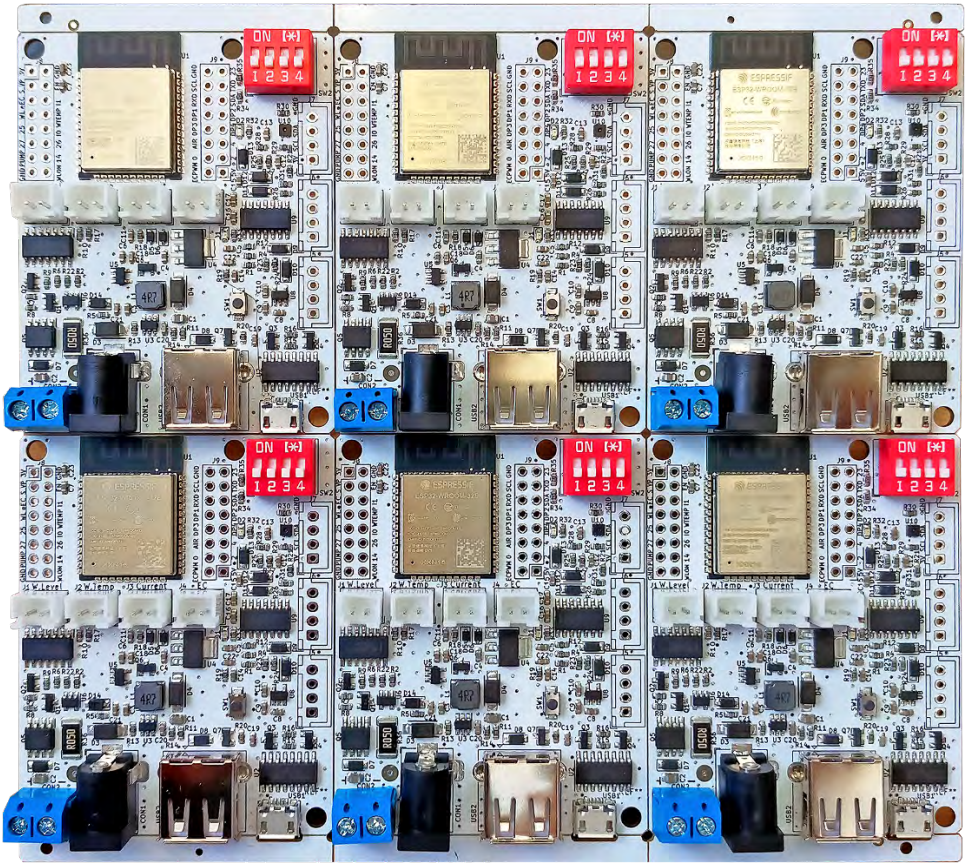


図 6.5.21　製造された V-cut 面付け基板

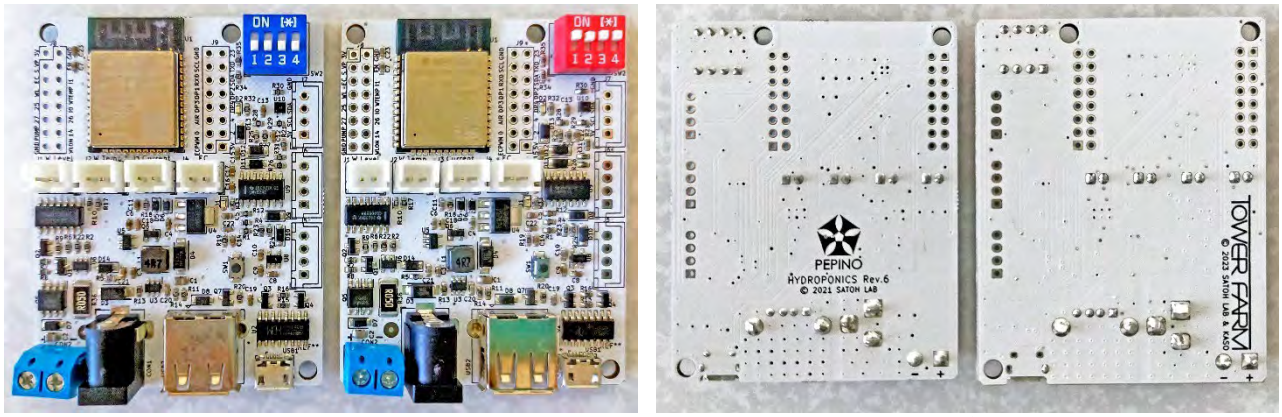


図 6.5.22　Rev.6 の Mouse bite と V-cut 基板の表面(左)と裏面(右)

7 管理アプリ

図 7.1 左にスマートフォンの管理アプリのホーム画面を示す。画面は縦にスクロールしたときの全てを表示している。画面は水温と EC の数値と、一週間の変動がグラフ表示される。水温は日中に上がり、夜間に下がるので、“過去一週間の水温”のグラフに一週間で 7 つの山が見えている。蛇口からホースを水耕栽培装置に接続しておけば、減った分だけ自動的に注水される。蛇口に接続せずに水位が下がっていく場合、満水の半分程度になると“高水位”→“低水位”と表示が変わる。アプリを使用していないときには、スマートフォンの待ち受け画面に低水位を伝えるメッセージが表示される。

“ポンプオフ”は潅水ポンプが動き出すと“ポンプオン”に表示が変わる。将来はポンプが長時間止まっている場合にエラー表示を行う予定である。

“オンライン”は AWS サーバと接続ができていることを意味し、繋がらないときは“オフライン”となる。

“信号良好”は水耕栽培装置のセンサモジュールと Wi-Fi ルータの間の電波強度を示し、弱くなるにしたがって“信号良好”→“通信可能”→“不明”に、接続が切れると“不明”と表示される。

“過去 3 時間のポンプ状態”は、直流 12V の潅水ポンプに流れる電流を計測して表示している。アナログ値なので高さが多少上下するが、半分以下の高さが続くような場合は、タンク内のホースジョイントが外れて、水がパイプ内で組み上げられていない可能性、あるいはポンプが劣化している可能性がある。その場合はパイプ上部の蓋を開けてシャワー部が濡れているか、あるいは図 8.2 の画面で強制的にポンプを回して動作を確認する。ポンプが止まって 30 分経つと、エラーメッセージが通知される。グラフが歯抜けになる場合は、ポンプの不具合ではなく、通信状態が悪い可能性もある。

“過去 3 時間の液肥ポンプ状態”のグラフは、EC 値が設定値を下回った時に、追肥用のエアポンプの On/Off 状態を表している。この画面での EC 値の設定は 1.0 で、これを下回るとエアポンプが動作して、ボトルから肥料の原液を装置のタンク内に数秒間注入される。このとき濃い原液はタンクの底に溜るが、EC センサの電極は底に横たわっているので、EC 値は急激に高くなる。そして潅水ポンプが動作するに従い、原液が拡散して EC 値は低下し、1.0 を切るとまた原液が追加されるといった動作が繰り返される。“過去一週間の EC 値”のグラフにこの原液注入→拡散のピークが 3 回見えている。なお、その前に急激に EC 値が

図 7.1 管理アプリのホーム画面

下がっているのは、液肥が減っていたタンクに水を大量に入れたためである。

潅水ポンプと追肥ポンプが動作するとそのデータはすぐに画面に反映されるが、その他のセンサデータは 10 分間隔で更新される。現在のデータをすぐに見たい場合は、画面右上のリロードボタン🔄をクリックする。その横の🏠は、水耕栽培装置を追加するときに、装置に付属の QR コードを読み取るのに使用する。画面左上のハンバーガーボタン☰をクリックすると、図 7.1 右の画面となり、現在登録されている装置の一覧が表示される。データの閲覧および装置の操作を行いたいときは、それぞれの装置名をクリックする。装置名の右にある⋮をクリックすると、登録している装置名の変更と装置の削除を行うことができる。

図 7.2 管理アプリの設定画面

画面右下の“装置設定”ボタンをクリックすると、図 7.2 左の画面に切り替わる。スライドボタンで“エラー通知”を ON にすると、水位低下、ポンプ停止、高水温、通信切断、等のエラー発生時にスマホの待ち受け画面にメッセージが通知される。自動モードを ON にしておくと、潅水ポンプは 10 分に 30 秒動作する。定植直後等にポンプを多く回したいときや、作業のためにポンプを止めたいときなどは、自動モードを OFF にする。すると“手動ポンプ時間”のスライドバーが現れる。スライドの最左が停止、最右が連続運転なので、所望のタイミング（バーの上に表示される）に設定する。設定が終わったならば「確定」ボタンをクリックする。

EC センサの測定値は電極の状態によって変化する。液肥に長時間浸けていると、なんらかの変化が起こって測定値がずれることがある。したがって市販の EC 計(2,000 円程度)の測定値と図 7.1 左に表示されるセンサの値にずれがないかを一ヶ月に一回程度チェックして、10%以上異なっている場合は校正を行う。これを行うのが“EC 校正”で、このボタンを ON にし、“EC 設定”に EC 計で測定した値を設定し、「確定」ボタンをクリックする。すると図 7.1 左に表示されている EC 値が設定した値に変わる。校正が終わったならば、“EC 校正”を OFF にしておく。

“EC 制御”を ON にすると、液肥濃度が下がると追肥を行う。目標の EC 値は“EC 校正”でも利用した“EC 設定”を用い、“確定”をクリックする。なお“EC 制御”と“EC 校正”は同時に行うことができないため、どちらかのスライドスイッチを ON にすると、もう一方は必ず OFF になる。また追肥を継続する場合は“EC 制御”を ON のままにしておく。液肥濃度が EC 設定の値よりも低いとき、“確定”を押すとすぐに液肥の原液が注入される。なお原液は少しずつしか注入されないため、真水を大量に入れた後、すぐに濃くしたい場合は“確定”ボタンを連続して押すか、直接タンクに手で追加する。

8 6次産業化

昨年度に引き続き、社会福祉法人「むうぷ」が経営している食茶房「むうぷ」で図 8.1 のように収穫した水耕栽培の野菜を利用して、ランチやお菓子を提供した。雇用している障がい者の方々に収穫もしてもらう予定であったが、残念ながら収穫のピークの 7 月下旬～8 月上旬にコロナにより「むうぷ」が休業となってしまい、実施することはできなかった。

図 8.1　収穫したカボチャを使った食茶房「むうぷ」のランチとお菓子

図 8.2　野崎果樹園の養蜂施設へのネットワークカメラとセンサの試験設置

本事業に関連して三鷹市生活環境部都市農業課から、三鷹市立図書館から徒歩 10 分ほどの「野崎果樹園」を7月に紹介いただいた。「野崎果樹園」では図 8.2 のように、農作物の授粉に不可欠な養蜂も営んでいる。佐藤

研究室では水耕栽培のセンサ技術を応用したスマート養蜂システムの研究開発にも取り組んでおり、原宿、赤坂、埼玉県坂戸市の養蜂施設で実験を進めてきた。そこで野崎果樹園にも 8 月からカメラとセンサを試験的に導入した。また 9 月に図 8.3 の「しみずファーム」で開かれた養蜂の勉強会に参加し、三鷹市の養蜂家と交流を深めた。その後 11 月に電気通信大学で養蜂サークルを立上げ、本館屋上で図 8.4 のように養蜂を開始した。

図 8.3 しみずファームの養蜂施設での勉強会)

図 8.4 電気通市大学の屋上養蜂施設とスマート養蜂システム

図 8.5 電気通市内愕生協での三鷹産のハチミツを使ったお菓子の販売

大学でハチミツが採れるのは 4 月以降であるが、三鷹産のハチミツを活用したお菓子を近隣の洋菓子店や調理製菓専門学校と進め、11 月末から図 8.5 のように大学生協および洋菓子店での販売を開始した。地元産のハチミツは大変好評で生産が追い付かず慢性的な品切れ状態となったうえ、三鷹の養蜂家の方から仕入れていたハチミツの在庫も底をつき、令和 6 年 1 月からは一時的に県外のハチミツを利用している。国産ハチミツは国内流通のおよそ 5%しかなく、特に地元産はスーパーやデパート等では販売されておらず、道の駅や直売所といった限られた場所でしか入手できない。そのような地元ハチミツを使ったスイーツは、地元野菜の料理やスイーツと比較して差別化と高付加価値化が狙える。また両者を組み合わせての提供も訴求力があると考えられ、今後も様々な商品の開発を進めていく予定である。

9　むすび

本事業では、都市の建物の屋上等を活用して様々な野菜の栽培を可能とする水耕栽培システムを活用し、図書館や学校と言った生活に身近な場所での栽培・収穫体験を通じ、自然や都市環境について学ぶ講演会や授業を行った。コロナ禍で昨年度はできなかった調布第一小学校での農業体験と合わせた授業は、4 年生の 4 クラス（約 120 人）へ各 4 回、計 16 回行うことができ、担任教員と児童から好評であった。そのため本事業が終了した来年度も実施の予定である。またそこで得られた教育の知見を、他にも広げていきたい。三鷹市立図書館の栽培は日照や猛暑の問題からうまくいかなかったものの、興味を持たれた市民の方にも協力いただきながら継続していきたい。本年度は楽しむ農業のための小型水耕栽培装置だけでなく、生産としての水耕栽培装システムの技術開発も進めた。しかしながら都市農業も高齢の生産者の方が多く、それまでのやり方とはまったく異なる新しい技術の導入は難しい。ここでも次世代を担う子供達に、IT・AI 等の新しい技術をそれらが用いられていない産業へ活用してもらうための教育の重要性を実感した。

6 次産業化においては地元野菜に加え、地元ハチミツの活用に高い可能性が見いだされた。国産ハチミツは国内流通の 5%程度で、スーパーやデパートでは大規模養蜂場のものしか購入することができない。外国産の多くは混ぜ物がされていると言われ、一般にハチミツには消費期限がないので古いものも輸入される。それらに対して地元の採れたてのハチミツの香りは格別で、またその地域に咲く花により、季節ごとにまったく異なる味や風味を楽しめることができる。しかし、小規模の養蜂では安定した生産や販売経路の確保、さらにはブランド化、商品化が困難なため、直売所等で安価に売られることも少なくない。蜜蜂はハチミツの生産だけでなく、農作物の授粉にも不可欠な生き物である。しかし日本の養蜂は世界に比べて非常に遅れており、国として養蜂を推進している韓国の 1/10 程度の規模でしかなく、蜜蜂は危険な生き物であるという誤解を持つ人も多い。

令和 5 年 11 月に、隔年開催のミツバチサミットがつくば市で開かれ、全国から専門家や研究者、農家、企業学生が一堂に会した。その中で開催された全国学生養蜂サミットでは図 9.1 のように、電気通信大学が小学校での授業、6 次産業、そして IT 化についての発表が優秀書を受賞した。養蜂は IT・AI 化がまったく進んでいない業種で、スマート養蜂の活動が評価されたものである。このことからも、都市での新しい養蜂の可能性は非常に高く、スマート農業と合わせて調布市と三鷹市で活動を広げていきたい。

図 9.1　全国学生養蜂サミットでの発表