ftドゥイーノ

fschertechnik互換のArduino

取扱説明書

Dr.-Ing。ハーバウムまで

2021年1月12~~日~~

タンジャ、マヤ、ファビアン、アイダの場合

プロジェクトホームページ： http://ftduino.de コン
タクト： mailto：// info@ftduino.de フォーラム：
https://forum.ftcommunity.de/ WEEE登録番号：
DE 25270264

目次

1 印心 7日
1.1 ftドゥイーノ-概念 7日
1.1.1Schertechnikモジュラーシステム。 7日
1.1.2Arduinoシステム。 8日
1.2 ftドゥイーノコントローラ 9
1.2.1マイクロコントローラー 10
1.2.2USBポート 11日
1.2.3リセットボタン。 11日
1.2.4内部LED。 11日
1.2.5電源。 11日
1.2.6接続。 14日
1.2.7内部OLEDディスプレイを備えたバリアント。 17日
1.2.8Arduinoの経験豊富なユーザー向けの注意事項。 18日
1.3問題の解決策。 18日
1.3.1の緑色のLED ftドゥイーノ 点灯しません。 18日
1.3.2 ftドゥイーノ としてPCに表示されません COM：-ポートオン。 18日
1.3.3 ftドゥイーノ 動作しますが、出力は動作しません。 18日
1.3.4 ftドゥイーノ 恥じることはできません。 19日
1.3.5 ftドゥイーノ レオナルドとして認識されています。 20日

2 インストール 22日
2.1ドライバー。 22日
2.1.1 Windows10。 22日
2.1.2 Windows8.0およびWindows8.1。 22日
2.1.3 Windows7およびWindowsVista。 23
2.1.4Linux。 24
2.2ArduinoIDE。 26日
2.2.1Ubuntuソフトウェアストアのlinux用ArduinoIDE。 26日
2.2.2ボード管理者によるインストール。 26日
2.2.3アップデート。 28

3 最初のステップ 30日
3.1最初のスケッチ。 30日
3.1.1まばたきスケッチをにダウンロードします ftドゥイーノ。 31
3.1.2スケッチのしくみ。 32
3.1.3機能 設定（） と ループ（） 32
3.1.4スケッチの調整。 32
3.2schertechnikコンポーネントの制御。 33
3.2.1スケッチ。 33
3.2.2入力。 34
3.3PCとの通信。 35
3.3.1シリアルモニター。 36
3.3.2スケッチの説明。 37
3.3.3USB接続の確立。 37

4位プログラミング 39
4.1テキストベースのプログラミング。 39
4.2プログラミング言語C++。 41
4.3基本。 41
4.3.1コメント。 42
4.3.2エラーメッセージ。 43
4.3.3機能。 44
4.3.4機能 設定（）とループ（） 45
4.3.5例。 46
4.4役立つライブラリ関数。 46
4.4.1 pinMode（ピン、モード） 47
4.4.2 digitalWrite（ピン、値） 47
4.4.3 遅延（ミリ秒） 47
4.4.4 Serial.begin（速度） 48
4.4.5 Serial.print（val） と Serial.println（val） 48
4.4.6 ftduino.input_get（）、ftduino.output_set（） と ftduino.motor_set（） 48
4.5変数。 50
4.5.1データ型 int。 50
4.6条件。 51
4.6.1 もしも-命令 51
4.7研削。 52
4.7.1 その間-リボン。 52
4.7.2 にとって-リボン。 53
4.8例。 54
4.8.1単純な信号機。 54
4.8.2バリア。 55
4.9警告 少しの記憶。 56
4.9.1影響。 56
4.9.2予防措置。 57
4.10詳細情報。 59

5 ftドゥイーノ 学校で 60
5.1スクラッチを使用したグラフィックプログラミング。 60
5.1.1スクラッチバージョン。 61
5.1.2 Arduino（S4A）のスクラッチ1.4。 61
5.1.3スクラッチ3.0。 62
5.2 Blockly / Bricklyを使用したグラフィックプログラミング。 64
5.2.1ブリックリー。 65
5.2.2Brickly-Lite。 66
5.3Minecraftでの遊び心のあるプログラミング。 67
5.4 ArduinoIDEを使用したテキストベースのプログラミング。 68
5.4.1Arduinoのアイデア。 68
5.4.2Arduinoと ftドゥイーノ 69
5.4.3 ftドゥイーノ エントリーレベルのArduinoとして。 69

6日実験 70
6.1ランプタイマー。 70
6.1.1スケッチ ランプタイマー。 70
6.2緊急停止。 72
6.2.1スケッチ 緊急停止。 72
6.3パルス幅変調。 75
6.3.1スケッチ Pwm。 75
6.4ステッピングモーター制御。 79
6.4.1フルステップ制御。 81
6.4.2ハーフステップ制御。 82
6.5サーボモーター制御。 84
6.5.1外部6ボルト電源。 85
6.6の入力 ftドゥイーノ 87

6.6.1電圧測定。 87
6.6.2抵抗測定。 87
6.6.3出力としての入力。 88
6.7温度測定。 88
6.7.1スケッチ 温度。 89
6.8出力はオン、オフ、またはどれも出力しませんか？。 91
6.8.1スケッチ OnOffTristate。 91
6.8.2漏れ電流。 92
6.9アクティブエンジンブレーキ。 92
6.10USBキーボード。 94
6.10.1スケッチ USB / KeyboardMessage。 94
6.11USBゲームパッド。 95
6.11.1スケッチ USB /ゲームパッド。 96
6.12デバウンス。 97
6.12.1スケッチ デバウンス。 97
6.13Iの使用2Cバス。 100
6.13.1スケッチ I2C / I2cScanner。 101
6.13.2MPU-6050センサー。 101
6.13.3OLEDディスプレイ。 102
6.13.4 VL53L0XLIDAR距離センサー。 103
6.13.5 ftドゥイーノ 私のように。2Cクライアントと2つのカップリング ftドゥイーノs。 104
6.13.6 ftドゥイーノ-私2Cエキスパンダー。 109
6.13.7schertechnik方向センサー。 109
6.13.8schertechnik環境センサー。 110
6.13.9 Mini-I2Cサーボアダプタ。 112
6.14WS2812Bフルカラー発光ダイオード。 114
6.14.1スケッチ WS2812FX。 115
6.15からの音楽 ftドゥイーノ 115
6.15.1スケッチ 音楽。 116
6.15.2スケッチ MusicPwm。 116
6.16 ftドゥイーノ MIDI楽器として。 117
6.16.1スケッチ MIDI楽器。 117
6.17 ftドゥイーノ Androidスマートフォンで。 118
6.18 WebUSB： ftドゥイーノ Webブラウザを介した制御。 119
6.18.1Chromeブラウザ。 119
6.18.2WebUSBスケッチ。 119
6.18.3コンソール。 120
6.18.4ブリックライト。 121
6.18.5スクラッチ3.0。 121
6.19Bluetooth。 122
6.19.1Bluetoothバリアント。 122
6.19.2への接続 ftドゥイーノ 124
6.19.3PCまたはスマートフォンでの使用。 129

7日モデル 132
7.1自動化ロボット：ハイベイ倉庫 132
7.2電空：ピンボール。 133
7.3 ROBOTICS TXT Explorer：ラインフォロワー。 134
7.4アイダの信号機。 135
7.4.1ステートマシン。 136
7.5クラシック2Dプロッタ。 138

8日コミュニティプロジェクト 140
8.1 ftduino_direct： ftドゥイーノ-USB経由でTXTおよびTX-Piに接続 140
8.2 ftDuinIO： ftドゥイーノ-TXTおよびTX-Piの制御アプリ。 141
8.3 レンガのプラグイン： グラッシュ ftドゥイーノ-Bricklyでのプログラミング。 142
8.4 startIDE： TX-PiまたはTXTで直接プログラミングします。 143
8.5フィートエクステンダー：I。2C拡張子。 144

8.6 Arduino（S4A）のスクラッチ。 144
8.6.1インストール。 145
8.6.2S4Aでのピン割り当ての表現。 146
8.7 マインクラフト と ftドゥイーノ：コンピュータゲームは現実の世界と出会う。 146
8.7.1のインストール ftドゥイーノ-モッド。 147
8.7.2の準備 ftドゥイーノ 147
8.7.3の使用 ftドゥイーノ Minecraftで。 148

9つのライブラリ 149
9.0.1ポートの定義と定数 149
9.1 FtduinoSimple。 150
9.1.1スケッチで使用します。 150
9.1.2 bool input_get（uint8_t ch） 151
9.1.3 bool counter_get_state（uint8_t ch） 151
9.1.4 void output_set（uint8_tポート、uint8_tモード） 152
9.1.5 void motor_set（uint8_tポート、uint8_tモード） 152
9.1.6スケッチの例。 152
9.2 Ftduino。 152
9.2.1入力 I1 それまで I8。 153
9.2.2 void input_set_mode（uint8_t ch、uint8_tモード） 153
9.2.3 uint16_t input_get（uint8_t ch） 153
9.2.4出力 O1 それまで O8 と M1 それまで M4。 154
9.2.5 void output_set（uint8_tポート、uint8_tモード、uint8_t pwm） 154
9.2.6 void motor_set（uint8_tポート、uint8_tモード、uint8_t pwm） 154
9.2.7 void motor_counter（uint8_tポート、uint8_tモード、uint8_t pwm、uint16_tカウンター） 155
9.2.8 bool motor_counter_active（uint8_tポート） 155
9.2.9 void motor_counter_set_brake（uint8_tポート、ブール値オン） 155
9.2.10カウンター入力 C1 それまで C4。 156
9.2.11 void counter_set_mode（uint8_t ch、uint8_tモード） 156
9.2.12 uint16_t counter_get（uint8_t ch） 156
9.2.13 void counter_clear（uint8_t ch） 157
9.2.14 bool counter_get_state（uint8_t ch） 157
9.2.15 void Ultrasonic_enable（bool ena） 157
9.2.16 int16_t超音波_get（） 157
9.3コマンドの概要。 158

10 DIY 159
10.1建設段階の電源 159
10.1.1コンポーネントの極性。 160
10.1.2制御測定。 160
10.2マイクロコントローラの第2ステージ。 161
10.2.1マイクロコントローラの機能テスト。 161
10.3第3建設段階の入り口 162
10.4第4建設段階の成果 163
10.4.15ボルトでの出力テスト。 163
10.4.2ボードの履歴。 164

A。回路図 165

B。回路基板のレイアウト 166

C。配置計画 167

D。寸法 168

E。ケーシング 169

第1章

印心

建設キット用の電子機器とコンピュータモジュールは、1980年代に個人使用の家庭用コンピュータが始まって以来存在しています。これらのモジュールは、それ自体のインテリジェンスがほとんどなく、主にホームコンピューターとモジュラーシステムのモーターおよびスイッチ間の信号調整を担当していました。そのため、これらのモジュールは通常、インターフェイス、つまりコンピューターとモデル。

何年にもわたって、家庭用コンピュータのパフォーマンスは向上し、電子モジュールも多くのことを学びました。とりわけ、インターフェースは時間の経過とともにコントローラーになりました。主にパッシブなインターフェースは、ホームコンピューターまたは後にPCがプログラミングにのみ必要とする独自のインテリジェンスを備えたモジュールになりました。プログラムされると、これらのコントローラーはモデルを独立して操作することもできます。この目的のために、PCで開発されたプログラムデータがコントローラにロードされ、そこで保存されました。

今日のLegoまたはSchertechnikコントローラーは、それ自体が強力なコンピューターです。エンドユーザーがその複雑さを利用できるようにするために、メーカーは、快適なユーザーインターフェイスの背後にあるデバイスで実行されている電子コンポーネントとソフトウェアの詳細を隠しています。残念ながら、このように、そのようなシステムは、そのようなコントローラーの構造と機能についての知識を与える機会を逃しています。メーカーは、文字通りの意味で複雑な機械歯車を理解できるようにすることに優れていますが、関連するコントローラーは、ユーザーにとって不透明な構成要素です。

同時に、いわゆるメーカー運動は、ミレニアムの変わり目から発展しました。それは、日曜大工のアイデアをエレクトロニクス開発の分野に持ち込みます。Raspberry PiやArduinoなどのシステムでは、これらの完全にアクセス可能で文書化されたコントローラーのすべての技術的な詳細を調査し、独自のコントローラーを開発することができます。大規模なコミュニティは、広範なノウハウを提供し、知識交換のためのプラットフォームを提供します。SchertechnikやLegoのコントローラーとは対照的に、ここではコントローラーの内部に焦点を当てています。ただし、これらのコントローラーを使用するには、電子機器自体を構築するとき、特にロボット工学プロジェクトで機械部品を実装するときに、ある程度の手動スキルが必要になることがよくあります。

1.1 ftドゥイーノ-概念

の背後にある考え方 ftドゥイーノ 2つの世界の間に架け橋を築くことです。一方では、Schertechnik構築キットのRoboticsシリーズに機械的および電気的にシームレスに統合されています。一方、組み込みシステムのソフトウェア開発用のArduinoエコシステムに完全に適合します。

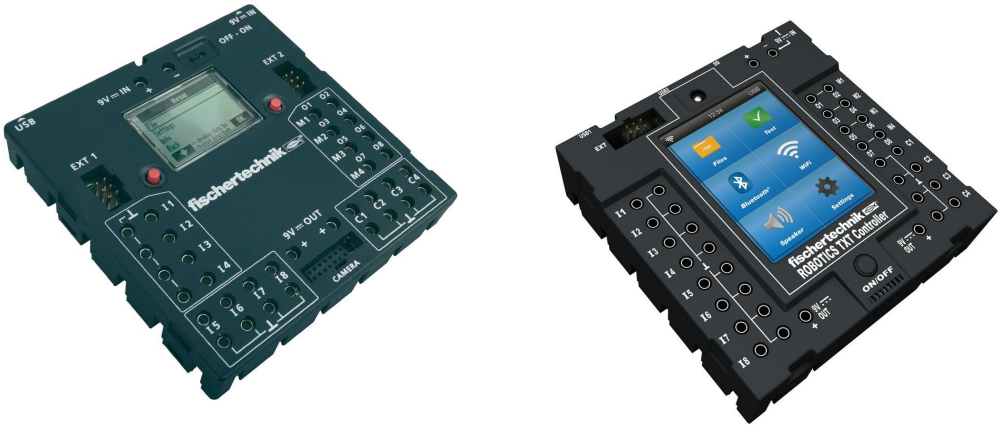
1.1.1Schertechnikモジュラーシステム

Fischertechnikは、テクノロジー指向の建設玩具です。焦点は、力学、電気機械、電子工学、そしてますますロボット工学と情報処理コンポーネントの必要な統合にあります。

Fischertechnikは、80年代初頭から電子モジュールを開発および販売してきました。これにより、コンピューターと機械モデル間の接続が可能になるか、独自のインテリジェンスが得られます。これに使用されるプラグインコネクタ、およびセンサー（ボタン、スイッチ、光センサーなど）およびアクチュエーター（ランプ、モーター、バルブなど）は、長年にわたって相互に互換性があり、それでも必要に応じて互いに組み合わせることができます。

図1.1： ftドゥイーノ

最後の2世代のコントローラー（Schertechnik TXおよびTXTコントローラー）は、同等の機械的サイズを持ち、モデルに接続するための同等の数とタイプの接続を備えています。現在のすべてのロボティクスキットのモデルは、これらの接続用に設計されており、互いに組み合わせることができます。



(a) TXコントローラー (b) TXTコントローラー

図1.2：元のコントローラー せん断技術

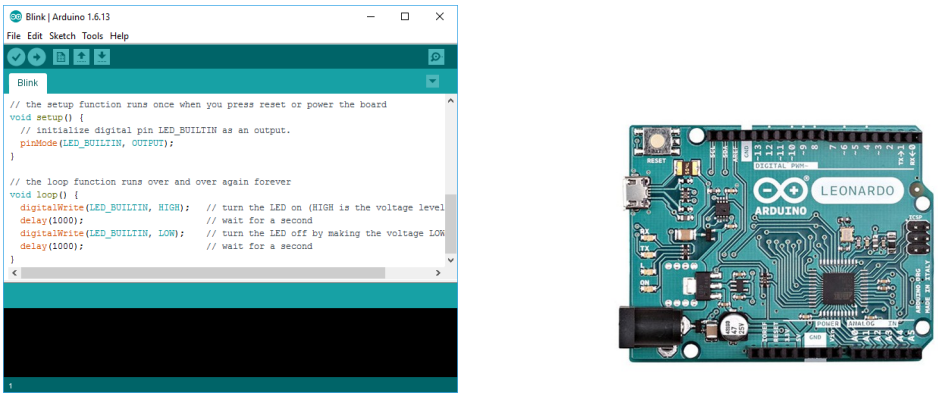
元のコントローラーは両方とも、8つのアナログ入力、8つのアナログ出力、4つの高速カウンター入力、およびIを備えています。2C拡張コネクタ。

Fischertechnik自体が、社内コントローラー用のビジュアルソフトウェア開発用のPCソフトウェアRoboProを販売しています。RoboProの使用を開始するのは比較的簡単で、すでに子供たちにアピールしています。中等学校、大学、職業訓練における実践的でコンテンツ関連のプロジェクトに関しては、RoboProの限界にすぐに到達します。Arduinoプラットフォームなどのシステムは、これらの分野で確立されています。

1.1.2Arduinoシステム

Arduinoエコシステムは、組み込みシステムのエントリーレベルおよびセミプロフェッショナルな開発とプログラミングの事実上の標準として、近年確立されています。組み込みシステムは通常、機械的に小さなコンピューターと情報処理モジュールであり、マシン内で制御と規制のタスクを引き受け、さらに頻繁に外界と通信します。

Arduino IDEは、Windows、Linuxで使用できる明確で使いやすいプログラミングインターフェイスです。



（a）Arduino開発環境（IDE）　（b）Arduino-Leonardoコントローラー

図1.3：Arduino開発環境とコントローラー

とApplePCを使用することができます。Arduino-Leonardoなど、プログラム対象のデバイスは、USB経由でPCに接続される小型で安価なボードです。それらは通常、ハウジングなしで提供され、プラグコネクタを介してセンサーとアクチュエータを接続するための多数の信号線を提供します。Arduinoプラットフォームで実行される典型的なタスクは、単純なデータ取得（温度ロギングなど）と制御タスク（ブラインド制御など）です。

Arduino IDEで作成されたプログラムは、Arduinoの世界ではスケッチと呼ばれています。Arduino IDEの助けを借りて、ftドゥイーノ USBケーブルを介してデバイスに直接ダウンロードします。

Arduinoプラットフォームは、単純なロボット実験にも最適です。最も単純なロボットプロジェクトでさえ、機械的に満足のいく実装は、多くの場合、より困難です。schertechnikシステムはこのギャップを埋めることができます。

1.2 ftドゥイーノコントローラ

the ftドゥイーノコントローラーは、意図的に機械的および電気的にTXおよびTXTコントローラーに基づいているため、現在のロボットボックスと直接組み合わせることができます。同時に、ソフトウェアはArduinoシステムとの互換性を維持しました。

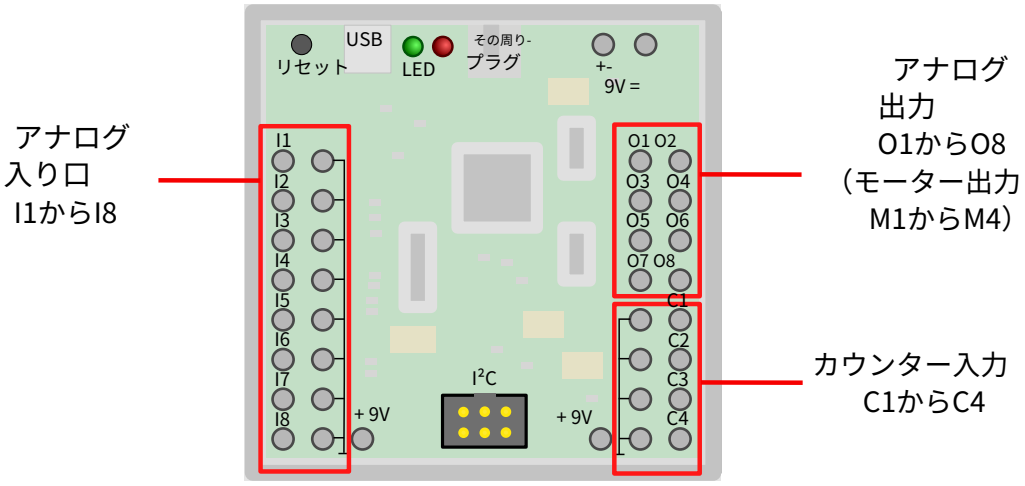


図1.4：の接続 ftドゥイーノ

1.2.1マイクロコントローラー

の中心 ftドゥイーノ タイプATmega32u4のマイクロコントローラです。このマイクロコントローラーはMicrochip（以前のAtmel）によって製造されており、Arduino-Leonardoでも使用されています。レオナルドのために翻訳されたスケッチは、多くの場合、直接ftドゥイーノ ぬるい。

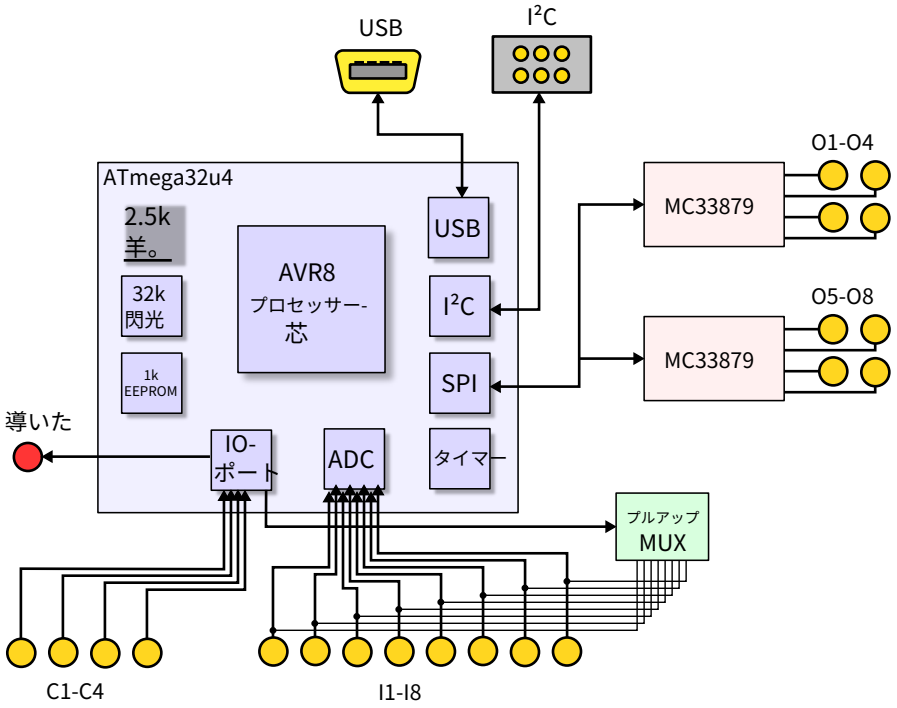


図1.5：のブロック図 ftドゥイーノ

ATmega32u4コントローラーは、ほとんどのArduinoボードのベースとなっているいわゆるAVRファミリーのメンバーです。AVRコントローラーは小型で安価であり、動作するために必要な追加コンポーネントはわずかです。それらのメモリと計算能力は、ロボティクスシリーズのすべてのSchertechnikモデルの操作に明らかに十分です。

ATmega32u4には、スケッチプログラムメモリとして使用される32キロバイトの不揮発性フラッシュメモリと、データストレージ用の2 .5キロバイトの内部RAMメモリがあります。プロセッサクロックは16メガヘルツです。一度に1つのスケッチをフラッシュメモリに永続的に保存でき、ftドゥイーノ 電源から切断されています。

ATmega32u4は、チップ上ですでに直接USBをサポートしているAVRファミリーの数少ないメンバーの1つです。このようにftドゥイーノ PCのUSBデバイスとして非常に柔軟に使用できます。

ブートローダー

the ftドゥイーノ ATmega32u4にプリインストールされたいわゆるCaterinaブートローダーで提供されます。このプログラムは、ATmega32u4の32キロバイトのフラッシュメモリのうち4つを恒久的に占有し、簡単に削除または変更することはできません。

ブートローダーはPCとの通信を可能にし、PCが28キロバイトのフラッシュメモリにプログラムデータを保存または交換できるようにします。このようにして、ブートローダーを使用してスケッチをにダウンロードできます。ftドゥイーノ。

内部LEDのステータスは、ブートローダーがアクティブであり、スケッチが現在実行されていないことを示しています（1.2.4を参照）。

1.2.2USBポート

プログラミングとデータ転送のためのPCへの接続は、USBを介して確立されます。theftドゥイーノ いわゆるミニUSBソケットがあり、標準のミニUSBケーブルを使用してPCに接続されます。

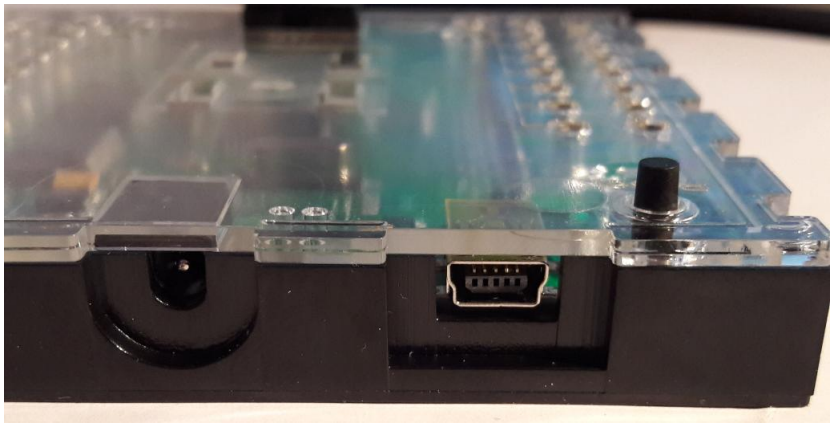


図1.6：電源とUSB接続 ftドゥイーノ

1.2.3リセットボタン

通常、ArduinoIDEはのブートローダーをロードできます ftドゥイーノ 新しいスケッチをダウンロードするには、アクティブにします。ただし、ダウンロードしたスケッチに通常のプログラム実行を妨げるエラーが含まれている場合、通常のプログラム実行中にUSB通信が機能しない可能性があり、ArduinoIDEは機能します。ftドゥイーノ もはや彼自身の合意に応じることはできません。

この場合、 ftドゥイーノ リセットボタンを介して。これを押すと、ブートローダーが強制的に起動し、それに応じてLEDがブートローダーの開始を示します。

スケッチに欠陥があるもの ftドゥイーノ したがって、ダウンロードの直前にリセットボタンを短く押すことで、修正されたスケッチを簡単に提供できます。詳細については、セクション1.3を参照してください。

1.2.4内部LED

the ftドゥイーノ 緑と赤の内部発光ダイオード（LED）があります。緑色の電源LEDは、内部の5ボルトの分岐に電力が供給されていることと、ftドゥイーノ 供給されます。

赤いLEDはあなた自身の使用のために利用可能であり、名前の下で彼自身のスケッチからユーザーが作成することができます LED_BUILTIN 対処済み（セクション3.1を参照）。

赤いLEDは、Caterinaブートローダーでも使用されています。 ftドゥイーノ 使用済み。ブートローダーがアクティブな場合、LEDは毎秒穏やかに明るくなり暗くなります（フェード）。

1.2.5電源

the ftドゥイーノ 次の4つの方法で電圧を供給することができます。

USB The ftドゥイーノ 他の電源が接続されていない場合は常に供給されます。USB
ただし、アナログ出力を動作させるには電源が 十分ではありません。USB電源で使用できるのは入力のみです。
さらに、アナログ入力での抵抗測定の精度が大幅に低下します（1.2.6を参照）。

図1.7： ftドゥイーノ 推奨されるオリジナルのSchertechnik電源

バレルコネクタ しますか ftドゥイーノ バレルコネクタを使用します。たとえば、schertechnik電源ユニット505287を使用します。[1] または電源
Schertechnikパワーセット505283から[2] 9ボルトで供給されるので、全体 ftドゥイーノ そこから供給され、USB接続がロードされていません。この場合、アナログ出力を使用することができ、アナログ入力での抵抗測定は完全な精度で実行されます。サードパーティのネットワークデバイスを使用する場合、schertechnikはアイテム番号134863で提供しています[3] 通常の5mm中空プラグからschertechnikが使用する3.45mmプラグへのアダプター。

9V =入力 の供給 ftドゥイーノ 例：バッテリーセットまたはバッテリーセットからのバッテリー34969[4位] と同等です
バレルコネクタを介した供給。しますかftドゥイーノ 9V =入力およびバレルコネクタを介して供給される場合、電源はより高い電圧を供給する電源から供給されます。バッテリーがシステムにフィードバックされていないか、バッテリーが充電されています。

私。[2]C。 私について[2]Cポートは ftドゥイーノ 主に小さなディスプレイや
センサー。ただし、この接続を介して自分で供給することも可能です。ここでは、USB経由の供給と同じ制限が適用されます。このように、例えば、2つの結合された供給 ftドゥイーノ■単一のソースから可能です（セクション6.13.5を参照）。

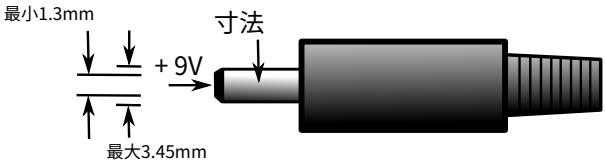
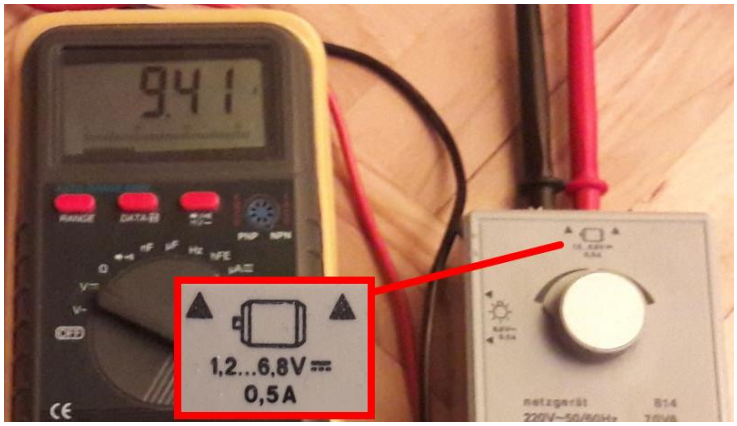


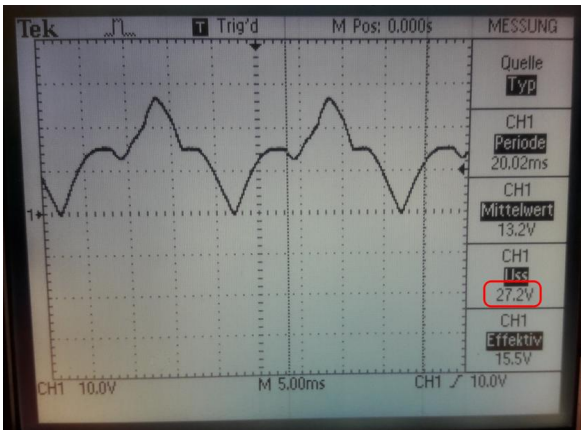
図1.8：3.45mmSchertechnikバレルコネクタ

注意 未知または非常に古い電圧源を使用する場合に推奨されます。多くの場合、印刷された公称電圧は現実から大きく外れています。図1.9に示す電源ユニットは、6.8ボルトの出力で27ボルトを超える開回路電圧を提供します。単純なマルチメータはまだ適度な9.4ボルトを示しており、この電源ユニットがftドゥイーノ 適切。オシロスコープだけが正確な電圧曲線を明らかにし、電圧が短時間でも27.2ボルトに達することを示しています。スイッチのオンとオフのプロセス中に、大幅に高い電圧が発生する可能性があり、ftドゥイーノ 損傷する。

[1] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=505287
[2] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=505283
[3] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=134863
[4位] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=34969



（a）マルチメータは良好な9ボルトを示しています



（b）オシロスコープは27ボルトを超えるピークを示します

図1.9：注意：古いSchertechnik変圧器は、6.8ボルトの出力で27V以上を供給し、 ftドゥイーノ 適切ではありません

このため、このような古いまたは未知の電圧源を使用する場合は十分に注意し、疑わしい場合は使用を控える必要があります。

図1.10：古いものと ftドゥイーノ 不適切な電源

疑わしい場合は、Schertechnikの最新の電源を使用する必要があります。現在のFischertechnik電源ユニット9V505287[5] 34969バッテリーセットのバッテリーと同様に、理想的に適しています[6日] またはバッテリーホルダーの9ボルトブロック。

サードパーティのデバイスを使用することがどうしても必要な場合は、安定した9ボルトの電圧を確保する必要があります。強力なモーターを操作できるようにするには、アンペア数が1.5アンペアを下回らないようにする必要があります。例として、図1.11に示すReichelt-Onlineversandのユニバーサルスイッチモード電源があります。[7日]。

[5] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=505287
[6日] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=34969
[7日] Reichelt MW 3R15GS https://www.reichelt.de/universal-schaltnetzteil-18-w-3-12-v-1500-ma-mw-3r15gs-p87340.html

図1.11： ftドゥイーノ オンライン配信からの適切なユニバーサル電源

1.2.6接続

の接続 ftドゥイーノ Schertechnikと互換性のある入力と出力に分けられます。これらは通常の2.6mmシングルプラグと、コンピュータセクターの通常のコネクタに適しています。schertechnikと互換性のある入力と出力は、schertechnikTXTコントローラーと同じように配置されています。したがって、TXTの配線方式は通常直接採用できます。

アナログ入力

the ftドゥイーノ 8つのアナログ入力があります I1 それまで I8、 0〜10ボルトの電圧と0〜10キロオームを超える抵抗を記録するために使用できます。

入力は、高値の直列抵抗を使用して、短絡や過電圧および低電圧から保護されています。

各入力は、ATmega32u4マイクロコントローラーの独自のアナログ入力に接続されています。アナログ値の取得は、最大10ビットの分解能（0〜1023の値の範囲に対応）で実行でき、マイクロコントローラーのハードウェアで直接実行されます。

分圧器は、マイクロコントローラーの入力電圧範囲を0〜5ボルトから、schertechnikが使用する0〜10ボルトの範囲に拡張します。Schertechnikモデルで発生するすべての応力を記録することができます。

任意の入力を使用して抵抗を測定できます ftドゥイーノ-5ボルトに対する抵抗器と相互接続されています。この抵抗器は外部抵抗器との分圧器として機能し、外部接続された抵抗器の値はマイクロコントローラーで測定された電圧から測定できます。Schertechnikモデルで一般的に使用されるすべての抵抗は、この方法で記録できます。

アナログ入力は外部の9ボルト電源に依存しませんが、PCのUSBポートを介して電源が供給されている場合にも機能します。ただし、この場合、抵抗測定の精度は大幅に低下します。
縁。

アナログ出力

the ftドゥイーノ 8つのアナログ出力があります O1 それまで O8。 これらの出力は、の2つの特別なドライバーブロックを介してアクティブ化されます。 ftドゥイーノ 制御されます。ドライバモジュールは、8つの出力のそれぞれを個別に制御できます。それらはそれらと同一です

せん断技術は、TXおよびTXTコントローラーで使用されます。したがって、出力は、TXおよびTXTコントローラーでも操作できるすべてのSchertechnikモーターおよびアクチュエーターと互換性があります。出力あたりの利用可能な最大電流は600mAから1.2Aです。

出力は、の純粋なUSB電源用です。 ftドゥイーノ 利用不可。

使用されるMC33879ドライバモジュールは、短絡防止であり、出力の過電圧および低電圧に対して堅牢です。

8つの出力はすべて、互いに独立して、グランド電圧または入力電圧、および高抵抗に切り替えることができます。2つの個別の出力を組み合わせて、モーター出力を形成できます。個々の出力O1 と O2 エンジン出力を形成する M1、O3 と O4 形 M2 などなど。

出力のアナログ値は、パルス幅変調（PWM）として知られているものによって生成されます。出力は連続的にすばやくオン/オフされるため、モーター、ランプ、およびその他の低迷する消費者は信号を追跡できず、平均値に従って動作します。この手順は、TXおよびTXTコントローラーでも同じように使用されます。

両方のMC33879は、 ftドゥイーノ いわゆるSPIインターフェースを介して内部接続されています。これは、マイクロコントローラの特別なPWM出力を使用してパルス幅変調を生成できないことを意味するため、PWM信号はスケッチまたは使用するソフトウェアライブラリによって生成する必要があります（第9章を参照）。いわゆるPWM周波数は、使用するスケッチによって決定され、必要に応じて変更できます。

PWMの詳細については、セクション6.3を参照してください。

カウンター入力

the ftドゥイーノ 4つの特別なカウンター入力があります C1 それまで C4。 これらの入力は、純粋なデジタル信号を記録し、イベントを高速で評価できます。スケッチに応じて、記録可能な最大信号レートは1秒あたり数10,000イベントです。

カウンター入力は、Schertechnikエンコーダーモーターのエンコーダーと互換性があり、特に回転角の評価と速度の決定に使用できます。

カウンター入力 C1 Schertechnik ROBOTX超音波距離センサー1330009を使用するオプションもあります[8日]
評価します。

注：ここで説明した3線式接続ケーブル付きの超音波センサーに加えて、4線式ケーブル付きの非常に古いセンサーがあります。長い間利用できなかったこの非常に古いセンサーは、ftドゥイーノ 互換性がありません（また、現在の元のSchertechnikコントローラーのいずれとも互換性がありません）。

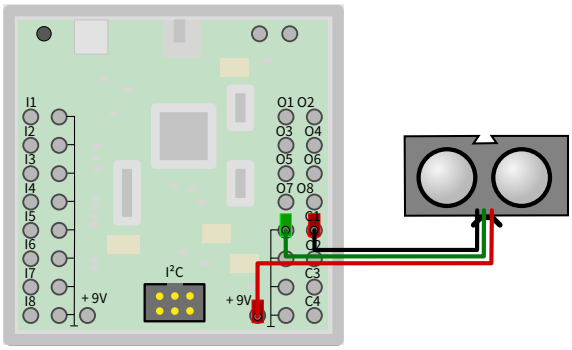


図1.12：超音波センサー133009の接続

[8日] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=133009

私。2Cコネクタ

私。2C接続は、SchertechnikTXコントローラーの接続と電気的および機械的に互換性があります。いわゆる私はそこのデバイスから導き出しました2C-BusはArduino環境でも頻繁に使用され、センサー、アナログ-デジタルコンバーター、ディスプレイなどの適切な電子部品の接続を可能にします。また、私は2C-Busいくつかのカップリング ftドゥイーノs可能なだけでなく、の結合 ftドゥイーノ セクション6.13で説明されているように、TXコントローラーとTXTコントローラーを使用します。

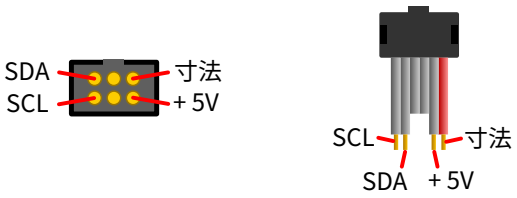


図1.13：Iのソケットとケーブルの割り当て2Cバス ftドゥイーノ

Iの信号2C接続は、TXコントローラーのように5ボルトレベルを使用します。また、の電源ftドゥイーノ 接続されたコンポーネントに供給するために5ボルトが提供されます。を達成するために、5ボルトの出力から最大100mAを引き出すことができます。ftドゥイーノ-内部電源に過負荷をかけないでください

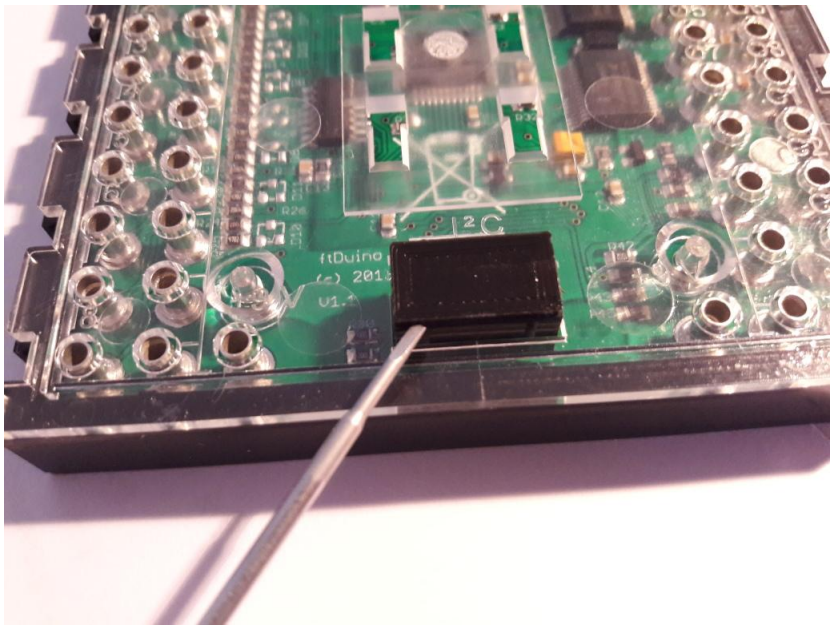


図1.14：Iから保護キャップを取り外します。2のCポート ftドゥイーノ

注意！ schertechnik TXTコントローラーおよびTXTでの操作を目的としたコンポーネントは、 ftドゥイーノ 互換性。TXTとの間の直接リンクftドゥイーノ TXTに損傷を与える可能性があります。TXTまたはTXTでの動作を目的としたコンポーネントをに接続する必要がありますftドゥイーノ 使用され、Iに一致します。2セクション6.13.5で説明されている中間の経営幹部レベルの調整。

注意！ 私のもの2C接続にある信号は直接であり、 ftドゥイーノ またはその電源に接続されています。この接続で短絡が発生した場合、または5Vを超える電圧が印加された場合、ftドゥイーノ 破壊されます。私。2したがって、C接続は経験豊富なユーザーのみが使用する必要があります。このため、ftドゥイーノ Iに保護キャップが付いています。2Cポートが分散されています。必要に応じて、このキャップはドライバーで慎重に取り外すことができます。

1.2.7内部OLEDディスプレイを備えたバリアント

のバリエーションがあります ftドゥイーノすでにOLEDディスプレイが組み込まれています。ディスプレイの解像度は128 * 32 ピクセルで、Iの内部にあります。2Cバス接続。

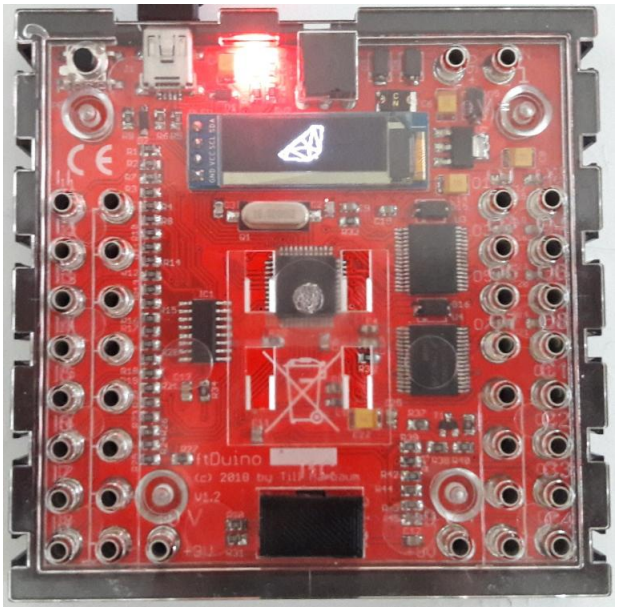


図1.15：内部OLEDの宇宙船アニメーション ftドゥイーノ

OLEDftドゥイーノ は、スイッチを入れた直後に表示を初期化し、ftDuinoのレタリングを表示する適応ブートローダーを備えています。ブートローダーはアクティブにディスプレイに対応するため、OLED-ftドゥイーノ いつも私。2Cバスマスター。Aftドゥイーノ したがって、ディスプレイが組み込まれている場合は、Iとしても、ごく限られた範囲でしか適していません。2仕事をするCクライアント、そして別の私からも2対処するCマスター。ただし、彼はマスターとして無制限に作業でき、たとえば、別のディスプレイレスで作業できます。ftドゥイーノ 1日2Cと話してください。追加の外部I。2Cデバイスは通常通り接続できます。

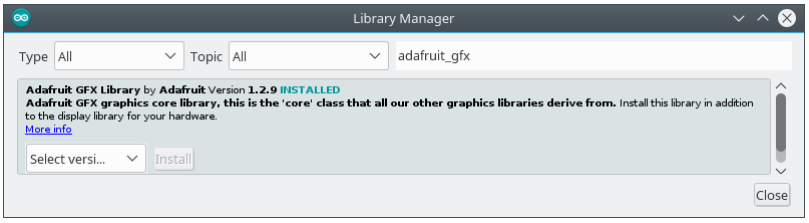


図1.16：ArduinoIDEのライブラリマネージャーにあるAdafruitGPXライブラリ

the ftドゥイーノ-インスタレーションハウス ファイル例 。Ftduino 。InternalOLED それぞれのいくつかの例の独自のバリアントを介して FtduinoDisplay.cpp-としょうかん。すべての例には、ArduinoIDEのライブラリマネージャーを使用して数回クリックするだけでインストールできるものも必要です。 AdafruitGFXライブラリ。

例 ファイル例 。Ftduino 。InternalOLED 。Ship3D これの迅速かつ簡単なバリアントをもたらしますとのライブラリ。他にIがない場合は常に使用する必要があります。2上のCデバイス ftドゥイーノ 操作され、それは速い画像の構築に依存します。

例 ファイル例 。Ftduino 。InternalOLED 。Ship3DWire 一方、Arduinoワイヤーに基づいていますここでは、ライブラリとイメージの構築が大幅に遅くなります。Iにアクセスするには。2C-Busは、ワイヤーライブラリと、ワイヤーライブラリによって制御される他の操作と一緒に使用します。2Cセンサーが可能です。この変種はFtduinoDisplay.cpp-ライブラリは、OLEDの場合にも使用されますftドゥイーノ さらに遠く ftドゥイーノs私について2Cは、追加の入力と出力を提供するために接続されています。

内部OLEDディスプレイがアドレスを占有します 0x3C（10進数の60）であるため、このアドレスで他のOLEDディスプレイと並行して簡単に操作することはできません。

1.2.8Arduinoの経験豊富なユーザー向けの注意事項

古典的なArduinoと ftドゥイーノ。まず第一に、これらはで使用される保護およびドライバー回路ですftドゥイーノ 接続がせん断技術と互換性があることを保証します。これらの回路は、の入力と出力の理由ですftドゥイーノ 通常のArduinoではありません pinMode（） と digitalWrite（）-機能は魅力的です。これらの機能は、ATmega32u4マイクロコントローラーの接続を直接制御するように設計されており、ftドゥイーノ 追加の回路が含まれています。

このため、 ftドゥイーノ 第9章で説明されているように、独自のライブラリを介して制御されます。

経験豊富なユーザーは、引き続き、 ftドゥイーノ 伝える。付録Aの回路図は、これに必要なすべての情報を提供します。

1.3問題の解決策

1.3.1の緑色のLED ftドゥイーノ 点灯しない

まず第一に、 ftドゥイーノ すべての接続から切断され、USBポートを介してのみPCに接続されます。の緑色発光ダイオードftドゥイーノ すぐに点灯する必要があります。そうでない場合は、別のPCまたは別のUSBポートを試す必要があります。

それでも問題が解決しない場合は、最初にUSBケーブルを確認してください。他のデバイスはこのケーブルで動作しますか？ケーブルの交換が必要になる場合があります。

1.3.2 ftドゥイーノ としてPCに表示されません COM：-ポートオン

the ftドゥイーノ PCによって認識されなくなり、 COM：-ポートが作成されました。

ライトの緑色のLEDが点灯します ftドゥイーノ？そうでない場合は、1.3.1の手順に従ってください。

緑色のLEDが点灯した場合は、リセットボタン（1.2.3を参照）を短く押すと、のブートローダーがアクティブになります。 ftドゥイーノ 数秒間アクティブにします。これは、セクション1.2.4で説明されているように、赤色LEDのゆっくりとしたフェードインおよびフェードアウトによって認識できます。この間、ftドゥイーノ PCによって認識されます。これは、セクション2.1.3で説明されているように、Windowsのデバイスマネージャに表示されます。

しますか ftドゥイーノ リセット後に検出されたが、デバイスマネージャのビューから数秒後に消えるか、不明なデバイスとして表示された場合は、スケッチに欠陥がある可能性があります。 ftドゥイーノ ロードされ、ArduinoIDEは ftドゥイーノ に接続します。この場合、Arduino IDEでまばたきの例（セクション3.1を参照）を開く必要があります。ftドゥイーノ リセットボタンを短く押してブートローダーモードにし、その直後にArduinoIDEのダウンロードボタンを押します。作業スケッチがロードされるとすぐに、ftドゥイーノ リセットボタンと対応するボタンを手動で押さなくてもPCによって認識されます COM：-ポートが再表示されます。

the ftドゥイーノ ブートローダーに数秒間留まり、その後通常のスケッチモードに戻ります。したがって、リセットボタンを押してからArduinoIDEからダウンロードを開始するまでの時間はできるだけ短くする必要があります。

1.3.3 ftドゥイーノ 動作しますが、出力は動作しません

出力を使用するには、 ftドゥイーノ バレルコネクタまたは通常のSchertechnikコネクタを介して9ボルトの電圧源に接続します。持っていますftドゥイーノ 9ボルトの供給が不十分な場合、出力は動作できません。以来ftドゥイーノ それがより低い電圧で動作しているとしても、その機能は十分な9ボルトの供給が利用可能であることを確実に示すものではありません。

それは ftドゥイーノ USB経由でPCに接続され、9ボルト電源が不足または不足した場合に、そこから電源を供給します。行きますftドゥイーノ 完全にオフ、USB接続を切断するとすぐに、9ボルトの電源がなくなります

また、極性と電圧が正しいか十分であることを確認する必要があります。必要に応じて、バッテリーを交換するか、使用したバッテリーを充電する必要があります。

1.3.4 ftドゥイーノ 恥じることはできません

たまたま ftドゥイーノ PCによって（一時的に）認識されますが、単に恥じることはできません。これには多くの理由が考えられますが、通常は次の2つの原因のいずれかです。

?? USB接続が電気的または機械的に不安定です

?? コンピュータ上の別のプログラムも通信しています ftドゥイーノ

機械的または電気的に不安定な接続は、通常、USBケーブルの緩みが原因で発生します。この場合、通常はUSBケーブルを少し揺すってftドゥイーノ 少しの間緊張を失い、それを再開させます。スケッチのアップロード中にこれが発生した場合、スケッチはぬるま湯ではなく、ftドゥイーノ セクション1.3.2で説明されているように、リセットボタンを押した後にのみPCによって認識されます。

原則として、電源LEDも少し光ります。図1.17では、フラッシュ中にUSBコネクタを引き抜くことにより、対応するエラーが発生しました。通常、別のUSBケーブルを使用すると役立ちます。

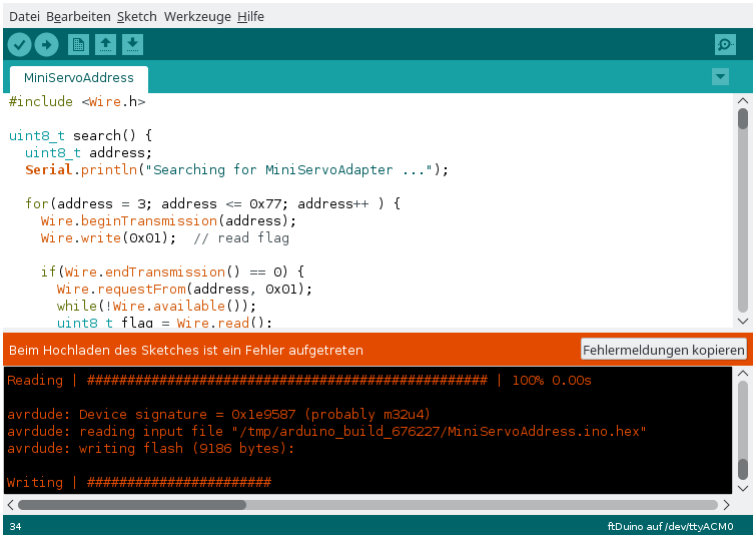


図1.17：アップロードの失敗

USBポートに緩い接続がない場合、詳細出力はアップロード中に追加情報を提供する場合があります。これらの出力は、設定でアクティブになります。

可能な出力は次のようになります。

書く| ################## avrdude：エラー：プログラマーがコマンドに応答しませんでした：set addr avrdude：エラー：プログラマーがコマンドに応答しませんでした：書き込みブロック
＃＃＊＊＊失敗した;
＊＊＊失敗した;
＊＊＊失敗した;
＊＊＊失敗した;
＊＊＊失敗した;
＊＊＊失敗した;
＊＊＊失敗した;
＊＊＊失敗した;
＊＊＊失敗した;
avrdude：エラー：バタフライプログラマーはavr_write_page（）を使用しますが、cmd（）メソッドを提供しません。

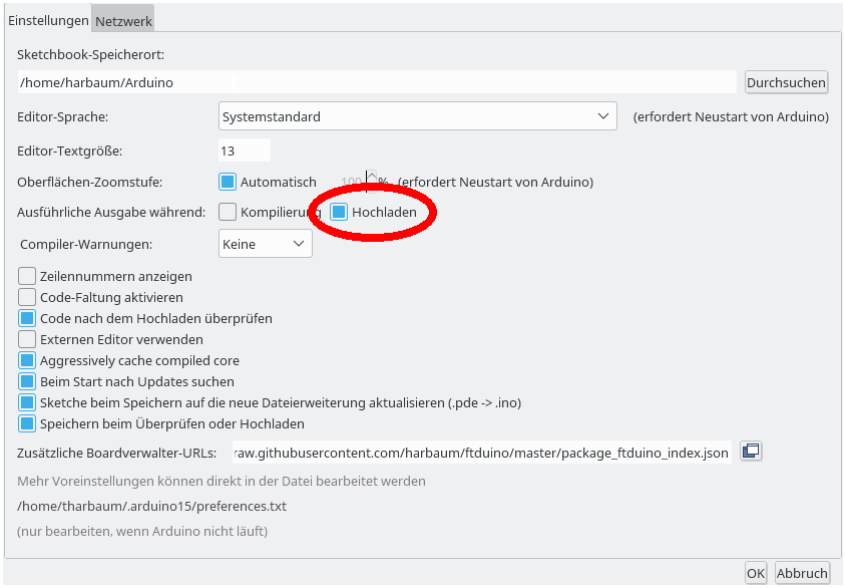


図1.18：アップロード中に詳細なデバッグ出力を有効にする

* * * 127ページ（アドレス0x0000-0x007f）の書き込みに失敗しました
* * * 失敗した;
* * * 失敗した;
* * * 失敗した;
* * * 失敗した;

ここでは、データ転送の途中でエラーが発生します。これはLinuxPCで最も一般的です。原因は通常、いわゆるモデムマネージャであり、モデムを管理し、ftドゥイーノ 接続して、モデムかどうかを確認します。これに関するいくつかのヒントは、セクション2.1.4にすでに記載されています。

ファイルのインストールに加えて/etc / udev / rules.d / 99-ftduino.rules 多くの場合、モデムマネージャサービスを使用すると役立ちます systemctrl-まず、テストとしてコマンドを停止します。

```
sudo systemctl stop ModemManager.service
```

または成功した場合は永続的に：

```
sudo systemctl disable ModemManager.service
```

この問題はそうではありません ftドゥイーノ-固有ですが、たとえば、Arduino-Leonardoにも影響します。したがって、レオナルドの問題に対応するインターネット上の解決策は、多くの場合、ftドゥイーノ 移行。

1.3.5 ftドゥイーノ レオナルドとして認識されています

可能性があります ftドゥイーノ 彼の身元とIDEではなく ftDuino Linuxシステムの例として図1.19に示すように、は Leonardoとして表示されます。

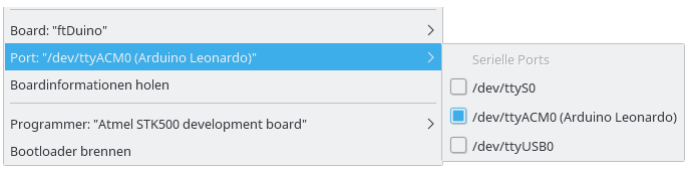


図1.19：IDEは ftドゥイーノ レオナルドとして

これは、スケッチをアップロードする前にIDEを使用しない場合に発生します ftDuino ボードとしてですが、ArduinoLeonardoです。

この状態は完全に重要ではありません。セクション2.2.2で説明されているようにボードを入手したらすぐにftDuino 次にスケッチをアップロードすると、次のように設定されます。 ftDuino 認識された。

第2章

インストール

を使用するためのソフトウェアのインストール ftドゥイーノ いくつかのステップで行われます。まず第一に、ftドゥイーノ 適切なドライバーにより、コンピューターはコンピューターがどのように動作するかを学習します。 ftドゥイーノ 通信する必要があります。

2番目のステップでは、いわゆるArduino IDEがインストールされます。つまり、実際のプログラミング環境とArduinoIDEがインストールされます。 ftドゥイーノ 接続されています。

インストールおよび第3章の最初のステップでは、 ftドゥイーノ USB経由でPCに接続します。電源ユニットまたはバッテリーを介した追加の電源は、ftドゥイーノ 使用すべきです。

2.1ドライバー

ほとんどのオペレーティングシステムは ftドゥイーノ プラグを差し込むとすぐにコンピュータによって認識されます。これは、Linux、MacOS X、Windows 10などに適用されますが、Windows7には適用されません。

2.1.1 Windows 10

Windows 10でftDuinoを使用する場合、ユーザーがドライバーをインストールする必要はありません。

一度 ftドゥイーノ Windows 10を実行しているPCに接続されている場合、適切なドライバーが自動的にインストールされます。Windows 10は、これを初めて表示しますftドゥイーノ 画面の右下に対応するメッセージが表示されます。数秒後、インストールが完了し、ftドゥイーノ 使用可能。

さらにプラグを抜き差ししても、それ以上のメッセージは生成されませんが、 ftドゥイーノ Windows 10では、ハードウェアを検出したときにWindowsPCが発する典型的なメロディーによっていつでも認識されます。

2.1.2 Windows8.0およびWindows8.1

Windows 8でのドライバーのインストールは、非常にありがたいものです。また、可能であれば、Windows10を使用する必要があります。

Windows 8の問題は、.inf-下のファイル https://harbaum.github.io/ftduino/ftduino/driver/ftduino.inf はMicrosoftによって署名されておらず、Windows 8は、たとえば、署名されていないドライバーをインストールすることをWindows7よりもユーザーにとってはるかに困難にします。この場合、実際のドライバーはMicrosoftとによってすでに提供されているコンポーネントであるため、これは特に厄介です。inf-Windows 8は、ファイルにそれを使用するように指示するだけです。実際のドライバーはMicrosoftから提供され、それに応じて署名されています。

ダウンロードするとき。inf-ファイルがとして保存されていないことを確認してください。txt-ファイルが保存されます。一部のWindowsおよびブラウザのバージョンは、ファイルのダウンロード時に目に見えない形でハングすることがよくあります。txt-で終わります。

署名されていないドライバーをインストールするためのインターネット上のさまざまな手順は、検索用語windows8で署名されていないドライバーをインストールすることで簡単に見つけることができます。主な手順を以下に簡単に要約します。

1.チャームメニューの設定領域を次のように開きます Windowsキー+「I」キー。 クリック 消す。

2.Shiftキーを押しながら 再起動 選ぶ。次に、メニューが次々に表示されます トラブルシューティング、 それから 詳細オプション、スタートアップ設定 そして最後に 再起動 選択する必要があります。

3.起動プロセスの後、キーを入力できる新しいメニューが表示されます。7 '（F7 また 7日 オンディジットブロック）は、署名されていないドライバーを許可するために、ドライバーの署名をオフにすることができます。

2.1.3 Windows7およびWindowsVista

Windows7とWindowsVistaにも適切なドライバーが付属しています。ただし、適切なものでなければなりません。inf-Windowsがこのドライバーを使用できることを確認するためのファイル ftドゥイーノ を使用します。

ドライバーがロードされていないことは、 ftドゥイーノ デバイスマネージャの[その他のデバイス]の下に表示されます。

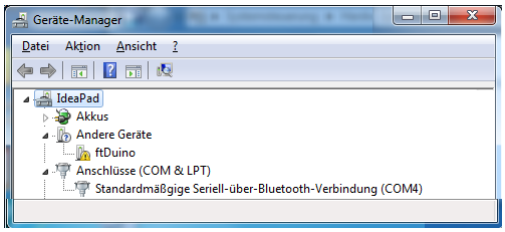


図2.1： ftドゥイーノ Windows7では適切なドライバーなし

。inf-ファイルは下にあります https://harbaum.github.io/ftduino/ftduino/driver/ftduino.inf 見つけられる。

ダウンロードするとき。inf-ファイルがとして保存されていないことを確認してください。txt-ファイルが保存されます。一部のWindowsおよびブラウザのバージョンは、ファイルのダウンロード時に目に見えない形でハングすることがよくあります。txt-で終わります。

ダウンロード後、ファイルを右クリックして、次のメニューで[インストール]を選択するだけです。

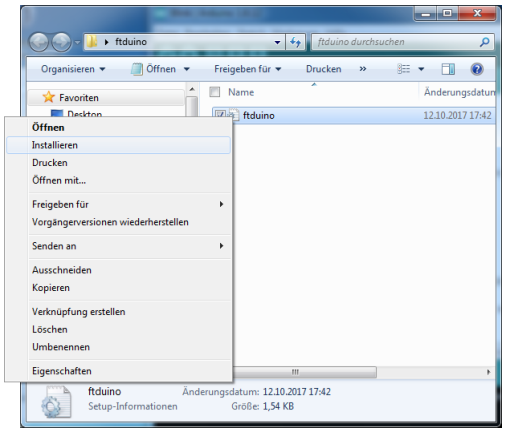
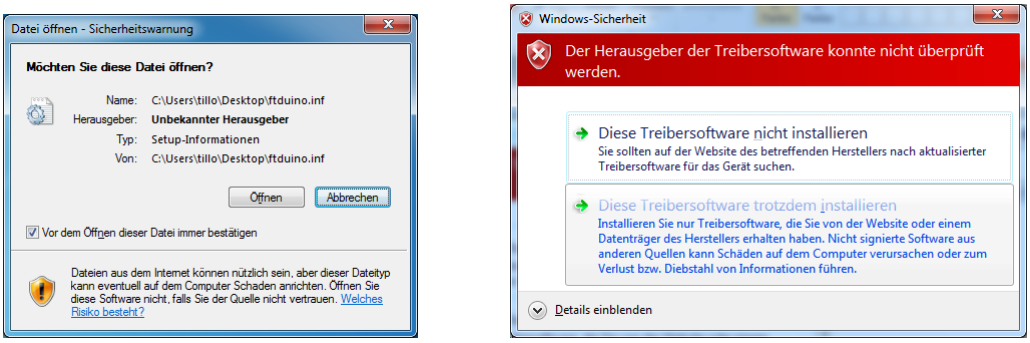


図2.2：右クリック ftduino.inf

その後、Windowsはドライバーのインストールを提案します。

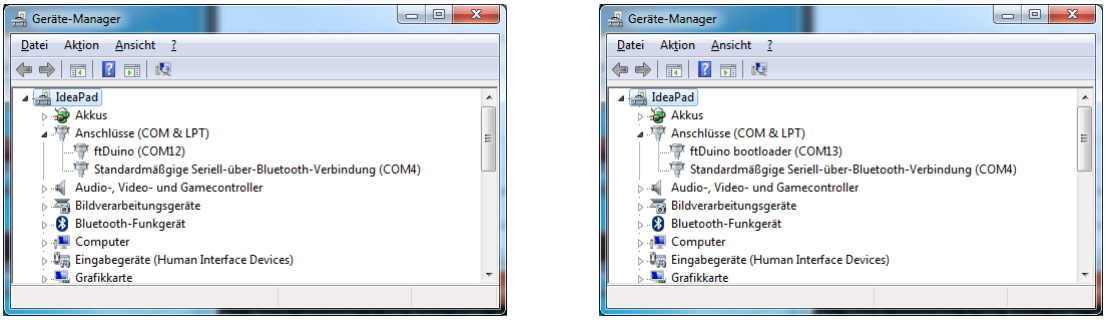
必要に応じて、セキュリティクエリがあります。実際のドライバーはすでにWindows7またはWindowsVistaの一部であるため、この質問に自信を持って同意できます。theftduino.inf-このファイルは、Windowsにそれを使用するように要求するだけです。

インストールが成功するとすぐに、 ftドゥイーノ いわゆる COM：-ポート統合。



(a) 確認クエリ (b) 該当する場合、次のセキュリティクエリ

図2.3：ドライバーのインストール



(a) アプリケーションモード (b) ブートローダー

図2.4： ftドゥイーノ Windows7で適切なドライバーを使用する

の動作モードに応じて ftドゥイーノ インストールされているアプリケーションに応じて ftドゥイーノ アプリケーションモードまたはブートローダーの場合。Windowsは2つの状態を区別し、2つの異なる状態がありますCOM：-ポートも。これは意図的なものであり、それ以上の苛立ちとなるべきではありません。ほとんどの場合、ユーザーにはアプリケーションモードのみが表示されます。

2.1.4 Linux

the ftドゥイーノ 手動による介入なしで、標準のLinuxPCによって認識されます。いわゆるAbstractControl Model（ACM）を実装しているため、Linuxシステムの/の下に表示されます。dev / ttyACMX ここで、Xは連番です。他のACMデバイスが接続されていない場合、ftドゥイーノ なので /dev / ttyACM0 関与。

たとえば、プラグを差し込んだ直後に詳細を確認できます。 ftドゥイーノ とともに dmesg-指示：

```
$ dmesg
。 。 。
[15822.397956] usb 3-1：xhci_hcdを使用した新しいフルスピードUSBデバイス番号9 [15822.540331]
usb 3-1：新しいUSBデバイスが見つかりました、idVendor = 1c40、idProduct = 0538 [15822.540334]
usb 3-1：新しいUSBデバイス文字列：製造元= 1、製品= 2、シリアル番号= 3 [15822.540336] usb 3-1：
製品：ftDuino
[15822.540337] usb 3-1：メーカー：Till Harbaum [15822.541084]
cdc_acm 3-1：1.0：ttyACM0：USBACMデバイス
```

正確なメッセージはシステムごとに異なりますが、一般的な内容は同等です。

は、認識されたUSBデバイスの詳細を提供します lsusb-指示：

```
$ lsusb -vd 1c40：0538
バス003デバイス009：ID 1c40：0538 EZプロトタイプ
デバイス記述子：
```

```
  bLength                18日
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass          239その他のデバイス
  bDeviceSubClass         2？
  bDeviceProtocol         1インターフェイスアソシエーション
  bMaxPacketSize0         64
  idVendor           0x1c40 EZPrototypes
  idProduct          0x0538
。。。
```

これらの出力は、の拡張USB機能を見ると特に興味深いものです。 ftドゥイーノ を使用します。

アップロードエラーまたはデバイスまたはリソースがビジーです

Linuxがすでに実際のデバイスドライバーを持っている場合でも、システム構成を調整する必要がある場合があります。症状は、あなたが打つことを試みるときですftドゥイーノ アクセスすると、ArduinoIDEに次のエラーメッセージが表示されます。

図2.5：ModemManagerがインストールされている場合のエラーメッセージ

考えられる原因：モデムマネージャ

この場合、最も可能性の高い原因は ModemManager、 モデムを操作するためのプログラムがインストールされ、 ftドゥイーノ 接続されています。ModemManagerが使用を試みないようにするには ftドゥイーノ 接続するには、次のコマンドを入力する必要があります。

```
sudo wget -P /etc/udev/rules.d https://raw.githubusercontent.com/harbaum/ftduino/master/ftduino/driver/99-ftduino.rules
```

ファイル /etc / udev / rules.d / 99-ftduino.rules その場合、正確に次のコンテンツが含まれている必要があります。

ATTRS {idVendor} == "1c40" ATTRS {idProduct} == "0537"、ENV {ID_MM_DEVICE_IGNORE} = "1" ATTRS {idVendor} == "1c40" ATTRS {idProduct} == "0538"、ENV {ID_MM_DEVICE_IGNORE} = "1"、MODE = "0666"

そうして ftドゥイーノ PCから一時的に切断してから再度接続すると、問題なく使用できるようになります。

このコマンドは、/というファイルを作成しますetc / udev / rules.d / 99-ftduino.rules の上。このファイルには、Linuxカーネルが特定のイベントを処理する方法に関するルールが含まれています。この場合、メーカーIDのUSBデバイスを挿入する場合1c40 およびデバイスの識別 0537 と 0538 これはModemManagerによって無視されます。さらに、セクション6.18.1で説明されているWebブラウザからのアクセスが機能するように、USBデバイスへのアクセス権がいくらか拡張されています。

一部のLinuxセットアップでは、ファイルをインストールするだけでは、ModemManagerの誤動作を防ぐのに十分ではありません。この場合、ModemManagerを非アクティブ化すると役立ちます。これは通常、問題なく省略できます。次のコマンドは、次のシステムが起動するまで、多くの一般的なLinuxインストールでModemManagerを停止します。

```
sudo systemctl stop ModemManager.service
```

コマンドに目的の効果がある場合は、次のコマンドを使用して設定を永続的にすることができます。

```
sudo systemctl disable ModemManager.service
```

その後、ModemManagerは、システムの起動後も再起動されません。

考えられる原因：グループメンバーシップの欠落

Linuxでのもう1つの一般的なハードルは、のUSBインターフェイスへのアクセスに使用されるハードウェアへのアクセス権です。 ftドゥイーノ 必要です。

それは ftドゥイーノ Linux PCに接続すると、セクション2.1.4で説明したように、デバイスが割り当てられます。 ttyACM 割り当てられました。これは例えばttyACM0 だからあなたは ls-コマンドの詳細情報：

$ ls -l / dev / ttyACM0
crw-rw ---- 1ルートダイヤルアウト166、0 Jul 27 23:06 / dev / ttyACM0

ここで重要な言葉 ダイヤルアウト、 これは、このインターフェイスにアクセスできるグループの名前だからです。あなたがそれに属しているかどうかはあなたにそれを伝えますグループ-指示：

$グループ
harbaumadmダイヤルアウトcdromsudo dip plugindev lpadmin sambashare

この場合、それは現れます ダイヤルアウト リストにあり、ユーザーは対応する権限を持っています。ダイブダイヤルアウト オンではありません。たとえば、ユーザーは次のことができます。 ハーバウム 次のコマンドでグループに参加します。

$ sudo adduserharbaumダイヤルアウト

次に、システムからログアウトして再度ログインし、新しい権限を使用できるようにする必要があります。

2.2 Arduino IDE

Arduinoの統合開発環境（IDE）は、最も一般的なオペレーティングシステムで無料で利用できます。 https://www.arduino.cc/en/Main/Software。独自のインストーラーを備えたWindowsバージョンは、たとえば、リンクのすぐ下にありますhttps://www.arduino.cc/download_handler.php 利用可能。このArduinoIDEが最初にインストールされます。

2.2.1Ubuntuソフトウェアストアのlinux用ArduinoIDE

Linuxでは、IDEをからダウンロードする必要があります http://arduino.cc 多くの場合、対応するLinuxディストリビューションのリポジトリまたはストアからIDEを直接取得する可能性もあります。理論的には、これにはインストールが迅速かつ簡単であるという利点があります。たとえば、Ubuntuストアは図2.6に示すバージョンを提供しています。写真の右上にある星が3つしかないというユーザー評価には、正当な理由があります。

このバージョンのArduinoIDEは、いわゆるSNAP形式に基づく比較的新しいパッケージ管理に基づいて構築されました。SNAPの新機能は、プログラムの権利がスマートフォンアプリから知っているのと同じ方法で管理されることです。これで、ArduinoIDEはUSBポートにアクセスする必要があります。ftドゥイーノ 接続されています。Ubuntu Linuxは現在、そのような権利の付与をサポートしていません（2019年7月28日現在）。したがって、Arduino IDEはストアからインストールできますが、ftドゥイーノ （および他のArduinoと同様に）は不可能です。この問題には2つの解決策があります。

1. ストアのプログラムは、グラフィックインターフェイスではなく、コマンドラインに次のように入力してインストールされます。 スナップインストール--devmodearduino-mhall119。 詳細については、たとえば、 [1]。
2. アーカイブからArduinoIDEをインストールします https://www.arduino.cc/en/main/software

どちらの場合も、セクション2.1.4で説明されているようにUbuntuでの権限を調整し、セクション2.1.4で説明されているようにモデムマネージャーをアンインストールする必要があります。

2.2.2ボード管理者によるインストール

に ftドゥイーノ Arduino IDEで使用できるようにするには、対応する構成を行う必要があります。Arduino IDEを使用すると、このプロセスを大幅に自動化できます。

[1] Arduino IDE用のアナウンススナップパッケージ： https://groups.google.com/a/arduino.cc/forum/#!topic/developers/Qp-G910Mt9c

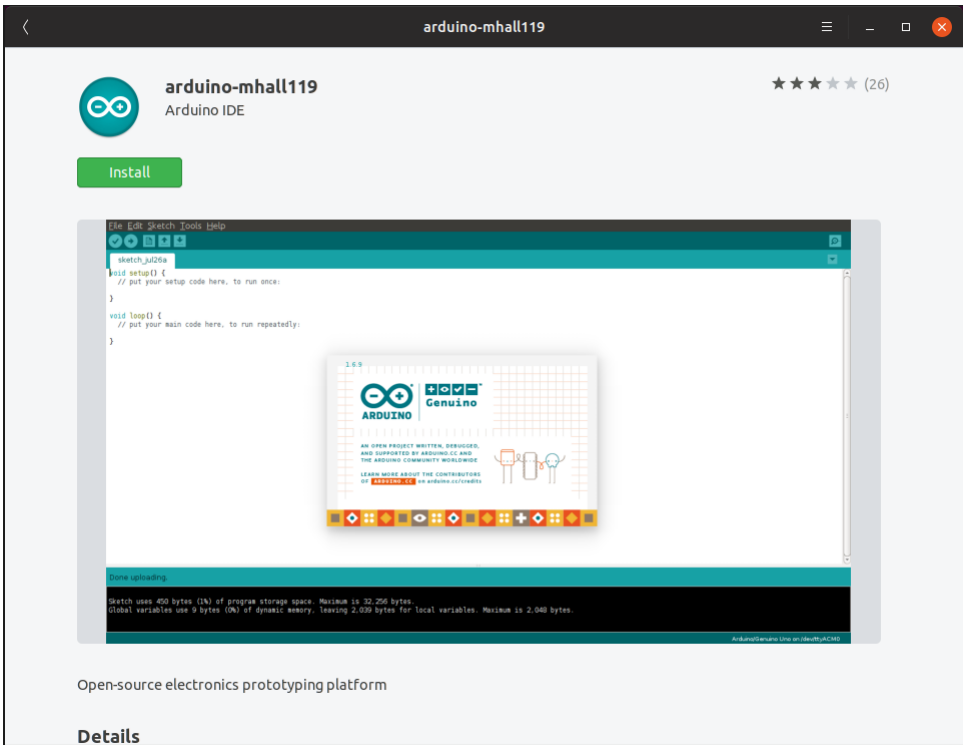


図2.6：UbuntuソフトウェアストアのArduinoIDEには落とし穴があります

追加のボードを簡単にインストールするために、ArduinoIDEにはいわゆるボードマネージャーが付属しています。まず、ボード管理者は、Arduinoの設定で通知を受ける必要があります。ftドゥイーノ-構成を見つけることができます。

あなたはそれを着ます https://harbaum.github.io/ftduino/package_ftduino_index.json 次のように設定で。対応する行を入力するときは、URLにアンダースコア（_）が含まれていることを確認してください。アンダースコアは、このPDFドキュメントからURLをコピー（コピーアンドペースト）すると失われる可能性があります。この場合、URLは手動で入力する必要があります。

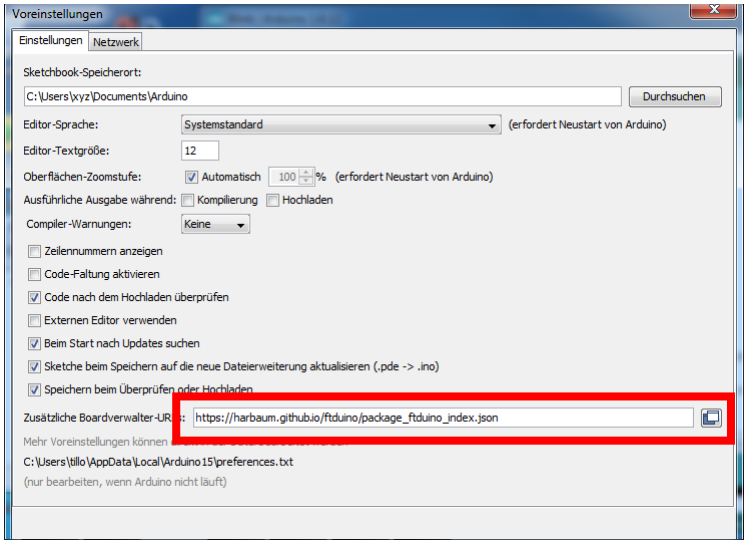


図2.7：のURL ftドゥイーノ-Arduinoプリファレンスの構成

実際のボードマネージャーには、IDEメニューから直接アクセスできます。 ツール 。ボード：... 。取締役会管理者..。 。

後に JSONファイルが設定に入力されている場合、ボード管理者は自動的に ftドゥイーノ-構成。

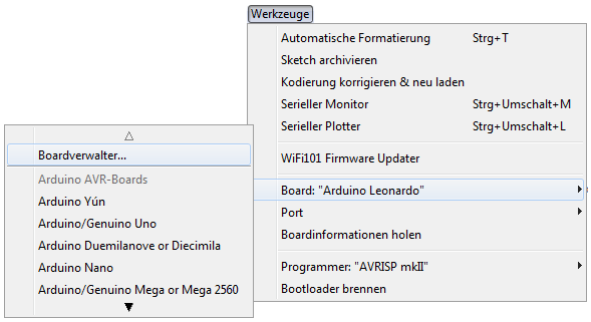


図2.8：ボードマネージャーはメニューから開始されます

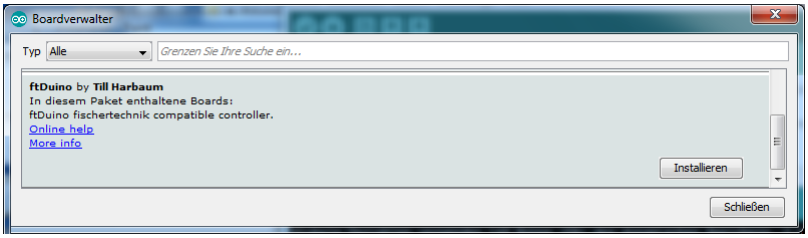


図2.9：これはボードマネージャーで実行できます ftドゥイーノ設置するボード

をクリックして インストールするには... すべてのためになります ftドゥイーノ 必要なファイルが自動的にダウンロードおよびインストールされます。

インストールが正常に完了すると、 ftドゥイーノ ボードの中から選択できます。

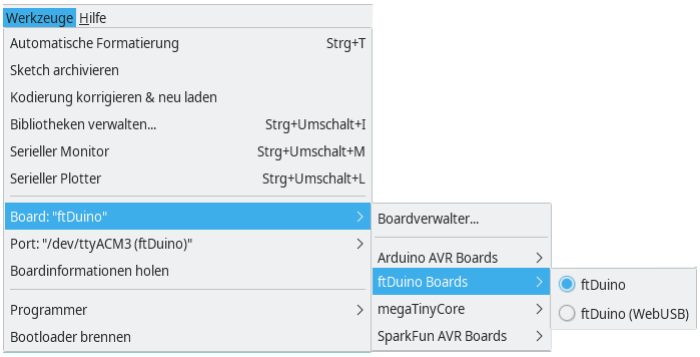


図2.10：選択 ftドゥイーノボード

注意：新しいインストールでは、ボードftDuinoとftDuino（WebUSB）のどちらかを選択できます。通常の作業では、ftDuino設定を選択してください。設定ftDuino（WebUSB）は、セクション6.18で説明されている特別なWebUSBスケッチを対象としています。この設定は、WebUSBスケッチを実際にロードする場合にのみ使用してください。

すでに ftドゥイーノ 接続され、必要なドライバがインストールされている、 ftドゥイーノ 下を選択します。 ポート

これでインストールは完了です。一部のサンプルプログラムは、インストール中にすでにインストールされています。この 下のメニューにあります ファイル例 。ftDuinoの例 。

これらの例を直接ロードして、 ftドゥイーノ ダウンロードする。

2.2.3アップデート

Arduino IDEは、ソフトウェアアップデートを自動的に通知します。 ftドゥイーノ-構成。少しの努力で、常に最新の状態に保つことができます。

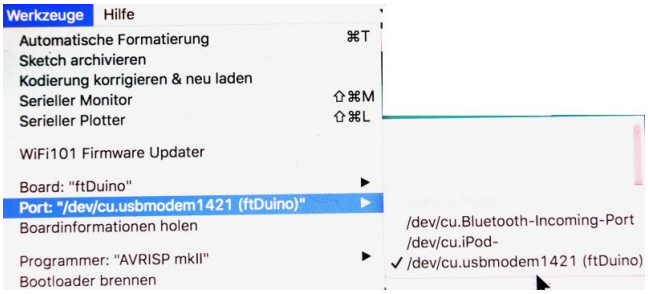


図2.11：MacOSでのポートの選択

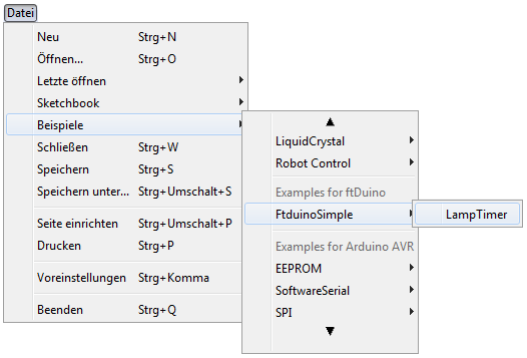


図2.12：の例 ftドゥイーノ-ボード

第3章

最初のステップ

この章では、 ftドゥイーノ ArduinoIDEを収集します。前提条件は、ftドゥイーノ PC上の適切なドライバーによってサポートされており、Arduino IDEが第2章で説明されているようにインストールされていること、および ftドゥイーノsが用意されています。

に加えて ftドゥイーノ たとえば、Schertechnik TXやTXTでも使用されているように、標準のミニUSBケーブルが必要です。

3.1最初のスケッチ

最初の試みでは ftドゥイーノ 個別の電源はありません。PCからUSB経由で供給されれば十分です。schertechnikの入力と出力は当分の間未使用のままです。

まず、ArduinoIDEに次のスケッチを直接入力できます。この例は、既成の例としてで見つけることができるため、必ずしも手動で入力する必要はありません。 ファイル下のArduinoIDEのメニュー ファイル例 。FtduinoSimple 。点滅 。

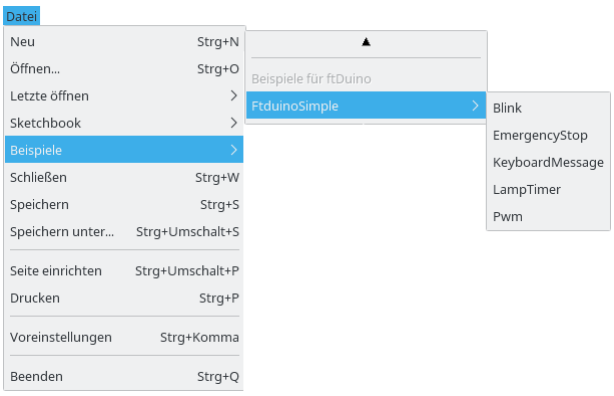


図3.1： ftドゥイーノ-ArduinoIDEの例

プレインストールされているすべてのサンプルをワンクリックでロードでき、選択したサンプルで新しいウィンドウが開きます。

```
1   /*
2     点滅
3
4位   ftDuinoの内部の赤いLEDを1秒間オンにし、1秒間オフにして、これを際限なく繰り返しま
5     す。
6日
7日   元の：
8日   http：//www.arduino.cc/en/Tutorial/Blink
9   */
```

```
10
11日 //セットアップ関数は起動時に1回呼び出されます 空所 設定 （） {
12日
13日    //内部LEDが接続されているピンを出力として設定します pinMode （（LED_BUILTIN 、 出力）；
14日
15日 }
16
17日 //ループ関数が何度も呼び出されます 空所 ループ （） {
18日
19日    digitalWrite （（LED_BUILTIN 、 高い）；遅れ        // LEDをオンにします（HIGHは高電圧レベルです）// 1000ミリ秒（1秒）待ちます
20日    （1000）；
21      digitalWrite （（LED_BUILTIN 、 低い）；            //電圧を//低レベル（LOW）に切り替えてLEDをオフにします// 1秒待
22日                                                        ちます
23      遅れ （1000）；
24   }
```

3.1.1まばたきスケッチをにダウンロードします ftドゥイーノ

まばたきスケッチが開いているはずです。theftドゥイーノ PCに接続し、メニューの下にある必要があります ツール 。ボード the ftドゥイーノ 選択されたものと正しいもの COM：-下のポート ツール 。ポート選択されます。

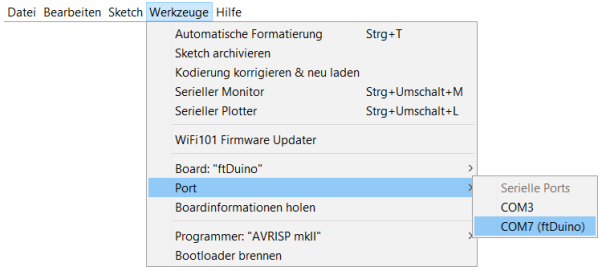


図3.2：WindowsでのArduinoIDEのポートの選択

Arduino IDEは、ステータスバーの右下にもこれを表示します。

（a）Linux （b）ウィンドウ

図3.3：選択された ftドゥイーノ ステータスバーに表示されます

上のスケッチのダウンロード ftドゥイーノ 左上のArduinoIDEのダウンロード矢印ボタンをクリックするだけです。

図3.4：ArduinoIDEのダウンロードボタン

IDEは、最初にスケッチをマシンコードに変換します。変換が成功した場合、マシンコードはに転送されますftドゥイーノ フラッシュメモリに保存されます。

ダウンロード中、内部の赤いLED ftドゥイーノ セクション1.2.4で説明されているように、ブートローダーがアクティブ化され、ダウンロードが実行されます。

ダウンロードが成功すると、スケッチがすぐに開始され、内部の赤いLEDがゆっくり点滅します。

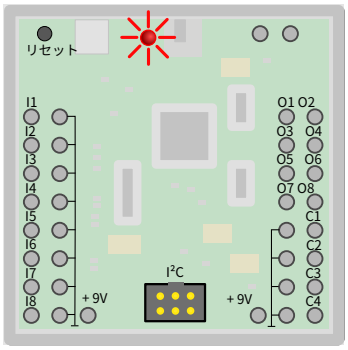


図3.5：内部の赤いLEDの点滅 ftドゥイーノ

3.1.2スケッチのしくみ

スケッチコードは、説明コメントの大部分で構成されています。これは、スケッチの機能にとってはまったく重要ではなく、人間の読者の理解に役立つだけです。コメント行は二重スラッシュ（//）で始まります。複数行コメントは/ *および* /で囲まれています。このドキュメントとArduinoIDEでは、コメントは明るい灰色で簡単に認識できます。実際のコードは12〜15行目と18 〜24行目のみです。

3.1.3機能 設定（） と ループ（）

すべてのArduinoスケッチには、少なくとも2つの機能が含まれています 設定（） （施設の英語）および ループ（） （英語のforループ）。実際の角かっこ（{および}）はセミコロンで区切られ、ftドゥイーノ 実行するコマンド。関数内のコマンド設定（） スケッチ開始時に1回実行されます。これらは通常、初期設定を行うため、または入力と出力をパラメーター化するために使用されます。のコマンドループ（）-一方、機能は、 ftドゥイーノ スイッチがオンのままになるか、再プログラムされるまで。これは実際のスケッチ機能が行われる場所であり、センサーが反応し、アクチュエーターが制御されます。

それも 点滅例は次のように機能します。の中に設定（）-の赤色発光ダイオードに接続された内部接続 ftドゥイーノ 出口を宣言しました。

の中に ループ（）-機能、赤色LEDの内部接続がライン19でオンになっています（電圧レベルが高い、 高い） 21行目ではオフになっています（電圧レベルが低い、 低い）。 その間、20行目と23行目は1000ミリ秒または1秒待機します。発光ダイオードがオンになり、1秒間待機し、オフになり、さらに1秒間待機します。これは何度も繰り返されるため、発光ダイオードは0.5ヘルツの周波数で点滅します。

3.1.4スケッチの調整

開始するには、多くの場合、既製のスケッチから始めて、独自の変更を加えることが理にかなっています。Arduino IDEの例は、PCのすべてのユーザーが利用できるため、最初は変更できません。サンプルスケッチに変更を加えて保存しようとすると、ArduinoIDEは独自のコピーを作成する必要があることを通知します。これを行うには、Arduino IDEでファイルダイアログが開き、保存する前にスケッチの名前を変更するオプションがあります。FastBlink。

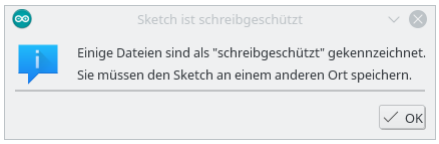


図3.6：ArduinoIDEはあなた自身のコピーを保存するように促します

この方法で独自のコピーを作成したら、必要に応じて変更できます。自分のコピーは、ArduinoIDEのメニューの下にあります ファイル スケッチブック 挿入され、後でいつでもそこからリロードできます。

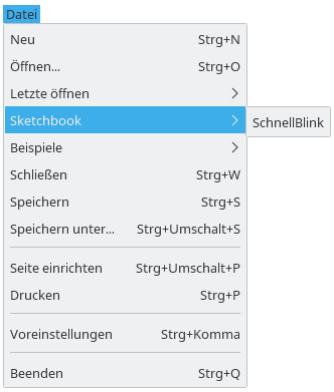


図3.7：コピー FastBlink ArduinoIDEのスケッチブックで

たとえば、スケッチでは、20行目と23行目の1000ミリ秒をそれぞれ500ミリ秒に変換できるようになりました。

```
18日 空所 ループ （） {
19日    digitalWrite （（LED_BUILTIN 、 高い）;遅れ       // LEDをオンにします（HIGHは高電圧レベルです）// 500ミリ秒（0.5秒）待ちます
20日     （500）;
21      digitalWrite （（LED_BUILTIN 、 低い）;          //電圧を//低レベル（LOW）に切り替えてLEDをオフにします//0.5秒
22日                                                    待ちます
23      遅れ （500）;
24   }
```

ダウンロードが成功すると、LEDが0.5秒間オンとオフに切り替わり、点滅周波数が2倍の1ヘルツになります。

3.2schertechnikコンポーネントの制御

の内部発光ダイオードを使用するには ftドゥイーノ 点滅はありません ftドゥイーノ 必要です。すべてのArduinoにはそのような内部発光ダイオードがあり、最初の例に使用できたはずです。

彼は彼の特別なスキルを果たしています ftドゥイーノ 通常のせん断技術センサーとアクチュエーターを扱うことになると。したがって、まばたきのスケッチは、発光ダイオードに加えて、出力に1つあるように拡張する必要があります。O1 接続されているランプが点滅します。

通常のSchertechnikランプはプラグで出力に接続されています O1 の ftドゥイーノ そして、2番目のプラグをのアース接続の1つに接続します ftドゥイーノ。アース接続は、図3.8にアース記号が付いた12の接続です。⊥ 接続されている。

9ボルトで動作するSchertechnik出力を使用する必要があるため、 ftドゥイーノ 9ボルトで供給することもできます。これは、たとえば、標準のSchertechnikパワーパックまたはバッテリーホルダーを使用して行うことができます。両方の接続は極性の反転から保護されているため、特にバッテリーを接続するときに損傷を与えることはありません。

3.2.1スケッチ

次のスケッチ例 BlinkO1 でも見つけることができます ファイル下のArduinoIDEのメニュー ファイル例 。 FtduinoSimple 。BlinkO1 。

```
1   //   BlinkO1.ino
2   //
3   //出力O1でのラモーの点滅//
4位
5   //（c）2018 by Till Harbaum < till@harbaum.org >
6日
```

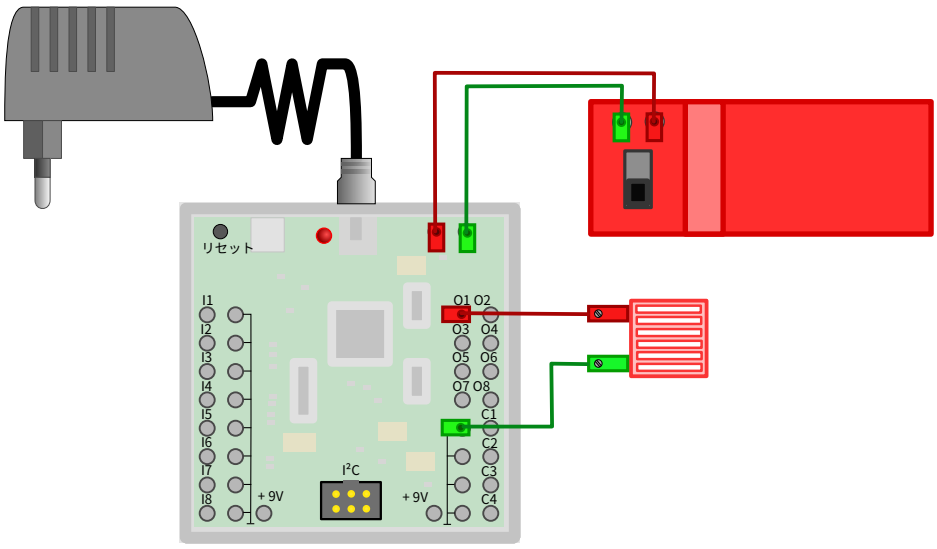


図3.8：schertechnikランプの点滅 ftドゥイーノ

```
 7日 # 含む       <FtduinoSimple.h>
 8日
 9   空所   設定 () {
10      //   LEDを初期化します
11日    pinMode ( (LED_BUILTIN 、    出力) ;
12日 }
13日
14日 空所 ループ () {
15日    //内部LEDをオンにして、O1（HIGHまたはHI）を出力します digitalWrite ( (LED_BUILTIN 、
16      高い) ; ftduino。output_set ( (Ftduino ： ： ：O1 、
17日                                          Ftduino ： ： ：こんにちは) ;
18日
19日    遅れ (1000) ;                              // 1000ミリ秒（1秒）待つ
20日
21      //内部LEDと出力O1（LOWまたはLO）をオフにします digitalWrite ( (LED_BUILTIN 、 低い
22日    ) ; ftduino。output_set ( (Ftduino ： ： ：O1 、
23                                          Ftduino ： ： ：LO) ;
24
25日    遅れ (1000) ;                              // 一瞬待って
26日 }
```

スケッチは、いくつかの詳細だけが元のまばたきスケッチと異なります。7行目、17行目、23行目が追加されました。7行目には、ftドゥイーノ が供給され、の入力と出力へのアクセス ftドゥイーノ 簡略化。17行目と23行目で出力O1 オン（こんにちは） またはスイッチをオフにしました（LO）。 このライブラリの詳細については、第9章を参照してください。

内部発光ダイオードのオンとオフを切り替えるコマンドは引き続き使用できるため、内部発光ダイオードは外部接続されたランプと並列に点滅します。

独自のスケッチで入力と出力を使用するためのさらに簡単な例と説明は、セクション9.1.1にあります。

3.2.2入力

The ftドゥイーノ 8つの入り口から I1 それまで I8 およびカウンター入力 C1 それまで C4。

これらの入力には、セクション9に示すように適切なライブラリを介してアクセスします。関してFtduinoSimple- ボタンの切り替えステータスは、ライブラリで照会できます。

```
1   # 含む <FtduinoSimple.h>
```

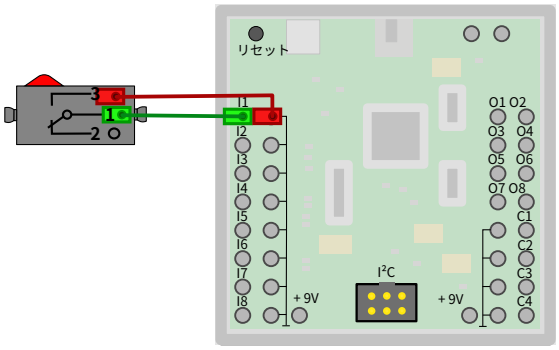


図3.9：入力のボタン I1 の ftドゥイーノ

```
 2
 3   空所 設定 （） {
 4位    //初期化は必要ありません
 5   }
 6日
 7日 空所 ループ （） {
 8日    //入力I1のキーの状態を読み取ります もしも （（ftduino。
 9      input_get （（Ftduino ： ： ： I1））） {
10         / * ...何かをする... * /
11日    }
12日 }
```

電圧や抵抗などのアナログ値を読み取るために、 Ftduinoライブラリが必要です。それらを使用する場合、抵抗値を読み取る前に、最初に入力測定モードを設定する必要があります：

```
 1   # 含む <Ftduino.h>
 2
 3   空所 設定 （） {
 4位    // Ftduinoライブラリの初期化 ftduino。初期化 （） ;
 5
 6日
 7日    //抵抗測定用に入力I1を準備します ftduino。input_set_mode （（Ftduino ： ： ： I1
 8日    、 Ftduino ： ： ： 抵抗）;
 9   }
10
11日 空所 ループ （） {
12日    //入力I1での抵抗の評価 uint16_t抵抗 = ftduino。input_get （（Ftduino ： ： ：
13日    I1）; / * ...何かをする... * /
14日
15日 }
```

第6章の実験では、さまざまな例があります。 ftドゥイーノ セクション6.7の温度センサーなどの特別なセンサーを含めて評価されます。

3.3PCとの通信

the ftドゥイーノ 主にモデルを自律的に制御することを目的としており、操作中にPCの助けに頼る必要はありません。それでも、操作中にPCとのデータ交換が望ましい場合があるのには理由があります。

特にスケッチの開発とトラブルシューティングの際に、たとえば、特定の値をPCに表示できる場合や、エラーメッセージをプレーンテキストでPCに送信できる場合は、多くの場合、非常に役立ちます。ただし、測定値をPCに出力して、さらに評価したり保存したりすることは、多くの場合役立ちます。

スケッチはこれを行うことができます COM：-間のポート ftドゥイーノ データ交換にはPCを使用します。theftドゥイーノ-サンプルスケッチ ComPort たとえば、 COM：-PC上でいくつかの簡単なテキスト出力を生成するためのポート。TheComPort

例はで見つけることができます ファイル下のArduinoIDEのメニュー ファイル例 。FtduinoSimple 。USB。ComPort 。
また、 ComPortSketchは出力を使用しないため、USB接続以外の電源は必要ありません。

```
1   /*
2       ComPort-COM：ポートを介したPCとの通信
3
4位  */
5
6日 int カウンター = 0;
7日
8日 空所 設定 () {
9     //ポートを初期化し、USB接続を待ちます シリアル。始める (9600); その間
10      (! シリアル);
11日                                  // USB接続を待ちます
12日
13日   シリアル。println ( ("ftDuino COM：ポート      テスト");
14日 }
15日
16  空所 ループ () {
17日   シリアル。印刷 ( ("カウンター： "); シリアル。      //「counter：」を出力します
18日   println ( (カウンター、 12月);                    //カウンタを10進数として出力します
19日
20日   カウンター = カウンター +1;                       // 1ずつカウントアップ
21
22日   遅れ (1000);                                    // 1秒待つ（1000ミリ秒）
23  }
```

3.3.1シリアルモニター

PC上のいわゆるターミナルプログラムを使用して、 ftドゥイーノ 経由 COM：-受信するポート。Arduino IDEは、このような端末を便利に備えています。メニューにあります
ツール 。シリアルモニター 。

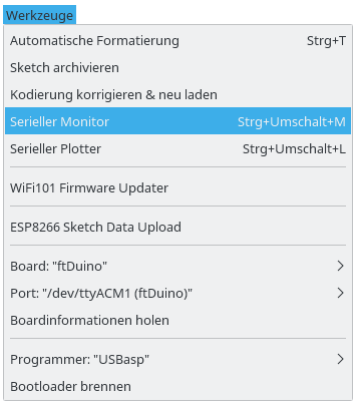


図3.10：シリアルモニターは ツール -メニュー

必ず COM：-ポートを個別に設定しないでください。 COM：-スケッチダウンロード用に設定済みのポートを採用しています。

メニューで選択すると、シリアルモニターはPC上に独自の追加ウィンドウを開きます。になったComPort すでにスケッチ ftドゥイーノ ロードされると、対応する出力がシリアルモニターが開かれるとすぐに表示されます。

行末

シリアルモニターには、ウィンドウの下部に行の終わりを示すための目立たないオプションがあります。このオプションは、ftDuinoからPCへの単純なテキスト出力には意味がありません。しかし、PCからの入力が期待されるとすぐに

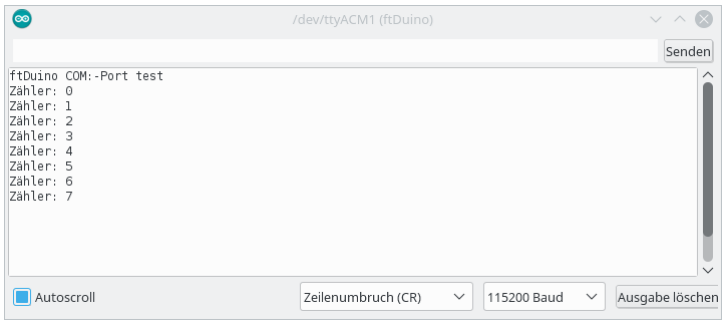


図3.11：シリアルモニター

たとえば、セクション7.1のハイベイ倉庫のモデルでは、このオプションは重要です。オプションがオンになっている必要があります 改行 また 改行（CR） コマンド入力が機能するように立ってください。

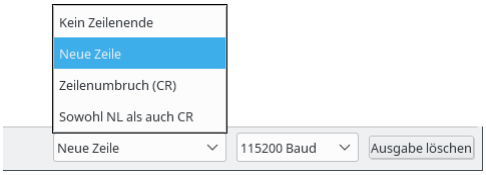


図3.12：行末の設定

の使用 COM：-ポートは、SketchDownloadとシリアルモニターの間で自動的に共有されます。シリアルモニターを開いた状態で、スケッチエディタを前面に表示し、いつでもダウンロードを開始できます。その後、ダウンロード中、シリアルモニターは自動的に非アクティブ化されます。

USBの特徴COM：-ポート ftドゥイーノ Arduino Leonardoから継承されたのは、いわゆるビットレート（ボーレートとも呼ばれる）には意味がないということです。の中にComPortたとえば、10行目では9600ビット/秒のビットレートが設定されていますが、図3.11では右下に115200ビット/秒のビットレートが設定されています。USB接続はこれらの設定を無視し、エラーなしで通信します。ただし、Arduino Unoなど、Arduinoファミリーの他のメンバーには、ここで同じ設定が必要です。

3.3.2スケッチの説明

スケッチは、操作するための実際の呼び出しにすぎません。 COM：-ポート。以来COM：-ポートもしばしばシリアル ポート、シリアルと呼ばれる英語はすべて対応する関数呼び出しをキャッチします シリアル。 の上。

10行目では、スケッチの開始時にすぐに呼び出されたもの 設定（）-機能は最初は COM：-通信用にポートが開いています。次に、11行目はCOM：-ポートはPC側で利用可能であり、最初の通信を行うことができます。次に、最初の開始メッセージが13行目でPCに送信されます。

繰り返しトラバースされた中で ループ（）-関数、テキストCounter：が17行目と18 行目に出力され、その後に変数の内容が続きます カウンター 10進表記で。theprintln（）-関数は、出力後に改行を実行します。したがって、次の出力は画面の次の行の先頭にあります。

最後に20行目で カウンター-変数が1つ増え、1秒（1000ミリ秒）待機しました。

3.3.3USB接続の確立

Arduino UnoなどのUSB通信用に別のUSB通信モジュールを使用するデバイスの場合、デバイスがPCに接続されるとすぐにUSB接続が継続されます。でftドゥイーノ Arduino Leonardoと同様に、マイクロコントローラーはUSB通信自体の重要な側面を引き継ぎます。つまり、新しいスケッチがマイクロコントローラーに転送されるたびに、論理USB接続が切断されて再確立されます。

しますか ftドゥイーノ ダウンロード直後にテキスト出力が表示されます COM：-ポート、USB接続がまだ再確立されていないため、最初のメッセージは失われます。したがって、11行目のスケッチは、ftドゥイーノ そして最初の費用が発生する前に再びPC。

この行は、テストとして削除またはコメントアウトできます（//でコメントに変換されたプログラムコードは実行されません）。

```
 8日 空所 設定（）{
 9       //ポートを初期化し、USB接続を待ちます シリアル。始める （9600）；
10
11日 //     while（！シリアル）；            // USB接続を待ちます
12日
13日    シリアル。println（（"ftDuino      COM：ポートテスト "）；
14日 }
```

このスケッチをにロードする場合 ftドゥイーノ、シリアルモニターに出力されるテキストは、その行からのみ開始されます カウンター：2。 前の2行は日付が付けられています ftドゥイーノ USB接続が再度確立される前にPCに送信されるため、失われます。USB経由の出力が追加で行われるだけで、ftドゥイーノ PCが接続されていなくても動作するはずです。

第4章

プログラミング

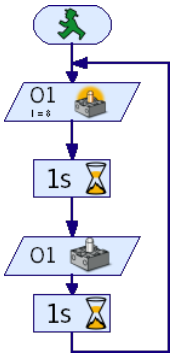
この章では、独自のプログラムまたはスケッチをプログラムする方法について説明します。 ftドゥイーノ。最初は独自のプログラムを作成したくなく、プログラムコードを理解せずに既成の例や実験を実行したい場合は、第6章に直接進んでください。

the ftドゥイーノ 常に増え続ける既製のサンプルスケッチが付属しています。したがって、多くのモデルや実験では、前の章で説明したようにこれらの例を使用するだけで十分です。ftドゥイーノ ロードする。しかし、せん断技術を使用して構築する楽しみが、構築手順の所定のパスを残したところから始まるのと同じように、ftドゥイーノ 本当の利点は、独自のスケッチでプログラムできることです。自己設計モデルと一緒に、印象的な可能性があります。ちなみに、あなたは機械の基礎を知るだけでなく、マイクロコントローラープログラミングの世界への現実的な洞察を得ることができます。

この章は、プログラミングの最初の洞察を提供することを目的としています。 ftドゥイーノ 与える。の基本的な言語構成ftドゥイーノ 使用されるプログラミング言語は、次の章でプログラムを理解するために必要な範囲で説明されています。説明は意図的に最小限に制限されています。それにもかかわらず、このかなり短い章の終わりまでに、あなたの最初の自分のスケッチに必要なすべての基本が利用可能になります。

4.1テキストベースのプログラミング

建設玩具の世界のほとんどのプログラミング環境とは対照的に、Arduinoはグラフィカルではなくテキストベースでプログラミングされています。SchertechnikのRoboProやLegosEV3プログラミングアプリなどのグラフィック環境は簡単に習得できるように設計されていますが、ArduinoIDEは実用的な関連性に重点を置いています。Arduinoのプログラミングとは、プロの商用製品開発者が行うのと同じ方法でプログラムを作成することを意味します。



(a) gra sch

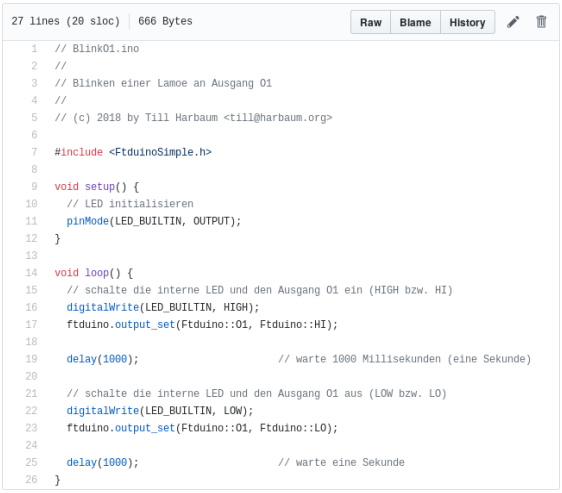
```
#include <FtduinoSimple.h>

void setup() {
}

void loop() {
  ftduino.output_set(Ftduino::O1, Ftduino::HI);
  delay(1000);
  ftduino.output_set(Ftduino::O1, Ftduino::LO);
  delay(1000);
}
```

(b) テキストベース

図4.1：出力O1でのランプの点滅

実際、テキスト表現にはいくつかの重要な利点があります。したがって、特に大規模なプロジェクトの場合、適切にフォーマットされたテキスト表現は、グラフィックよりもはるかに理解しやすくなります。

（a）ArduinoIDEで



（b）Githubで

図4.2：同じArduinoスケッチの表現

テキストベースのプログラムは、プレーンテキストで構成されています。それにもかかわらず、Arduino IDEなどの一部のプログラミング環境では、色付きの強調表示や異なるフォントサイズを使用したり、プログラムテキストに行番号を表示したりできます。ワードプロセッサからのテキストとは対照的に、このフォーマットは作成されたプログラムの一部ではありません。代わりに、それらはプログラミング環境自体の一部であり、たとえばプログラムコードが渡されるとフォーマットは失われます。したがって、同じプログラムコードは、異なる環境では完全に異なって見える可能性があります。これは、プログラムの機能には関係ありません。プログラムテキストがどのように表示されるかは関係ありません。実際のプログラム機能は、表示されるテキストの外観ではなく、表示されるテキストの内容に単独で責任を負います。

図4.2に示されているように、Arduino IDEと、たとえばWebサービスGithubとの間の同一のプログラムコードの表現は大きく異なります。

このマニュアルも独自の表現を使用しており、たとえば行番号が示されている場合があります。これらの行番号は、個々の行を参照しやすくするためにのみ使用され、プログラムの一部として明示的に入力しないでください。

```
 1   //   BlinkO1.ino
 2   //
 3   //出力O1でのランプの点滅//
 4位
 5   //（c）2018 by Till Harbaum < till@harbaum.org >
 6日
 7日 # 含む       <FtduinoSimple.h>
 8日
 9   空所   設定 () {
10      //   LEDを初期化します
11日    pinMode（（LED_BUILTIN、    出力）；
12日 }
13日
14日 空所 ループ () {
15日    //内部LEDをオンにして、O1（HIGHまたはHI）を出力します digitalWrite（（LED_BUILTIN、
16      高い）；ftduino。output_set（（Ftduino：：：O1、
17日                                          Ftduino：：：こんにちは）；
18日
19日    遅れ （1000）；                              // 1000ミリ秒（1秒）待つ
20日
21      //内部LEDと出力O1（LOWまたはLO）をオフにします digitalWrite（（LED_BUILTIN、低い
22日    ）；ftduino。output_set（（Ftduino：：：O1、
23                                          Ftduino：：：LO）；
24
25日    遅れ （1000）；                              // 一瞬待って
26日 }
```

Arduino IDEにも行番号を表示したい場合は、図4.3に示すように、ArduinoIDEのデフォルト設定で行番号をアクティブにすることができます。

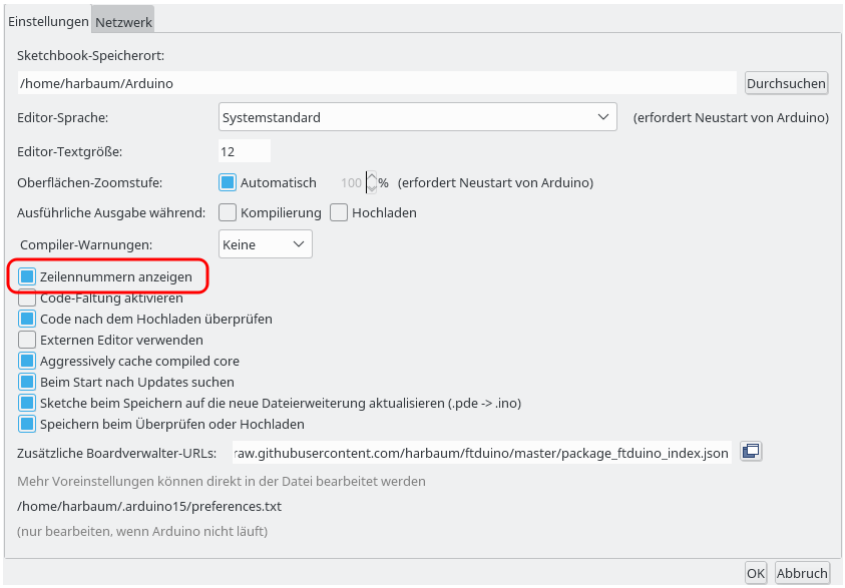


図4.3：ArduinoIDEでの行番号のアクティブ化

4.2プログラミング言語C++

Arduinosと ftドゥイーノ プログラミング言語C++、より正確には標準C++11プログラム。Arduinoスケッチは、いくつかの基本的な点で従来のCとは異なります++-プログラムは、それを除けば、プログラミングは標準に対応しています。プログラミング言語C++ プロのソフトウェア開発で広く使用されています。Windows、Linux、MacOSなどのすべての一般的なオペレーティングシステムの大部分はC言語です++ またはCに密接に関連するもの++ 関連するプログラミング言語。Arduinoプログラミングは、専門分野への現実的な洞察を提供します。

Cと呼ばれるスケッチ++-Arduinoの世界のプログラムは、Arduinoのマイクロコントローラーが応答するコンパイラーと呼ばれるプログラムによって、いわゆるマシンまたはバイナリコードに変換されます。 ftドゥイーノ 理解されています。このバイナリコードは、ftドゥイーノ ダウンロードで転送できます。

ArduinoIDEで使用されるコンパイラはいわゆるGCCです[1]。このコンパイラは業界でも使用されており、特に、すべてのAndroidスマートフォンで使用されているLinuxカーネルを変換するために使用されます。Arduino IDEとその内部で使用されるツールは、決して純粋な趣味のテクノロジーではありません。それどころか、初心者に適した表面は、強力でプロフェッショナルなコンポーネントを隠します。

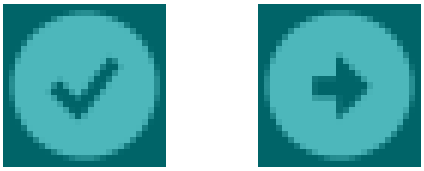
4.3基本

Arduinoスケッチはテキストによる説明で構成されています。有効なスケッチを形成する最小限のテキストは次のようになります。

```
空所 設定 () {
}

空所 ループ () {
}
```

[1] GNUコンパイラコレクションGCC： https://gcc.gnu.org/

このスケッチを入力するには、最初にArduino IDEを開き、メニューで選択します ファイル新しい開く
上記の行がすでに正確に含まれている新しいウィンドウ。//で始まる2行もあります。これらを削除して、最後のテキストが上記の例に正確に対応するようにすることができます。



（a）翻訳 （b）ダウンロード

図4.4：ArduinoIDEボタン

この最小限のスケッチは、Arduino IDEで正常に翻訳され、 ftドゥイーノ ロード。この目的のために、ArduinoIDEには図4.4に示す2つのボタンがあります。サーフェスはマシンコードへの変換を開始します。この機能は接続する必要はありませんftドゥイーノ また、スケッチに翻訳を妨げる基本的なエラーが含まれていないかどうかをすばやく判断するために使用できます。2番目のボタンは、へのダウンロードを開始しますftドゥイーノ。スケッチがまだ翻訳されていない場合は、ダウンロードボタンをクリックすると自動的に翻訳が開始されます。この翻訳が成功した場合にのみ、ダウンロードが開始されます。

経由でスケッチを持っていましたか ファイル新し新しく作成されたArduinoIDEは、スケッチを使用できるかどうかを尋ねる場合があります
保存したい。保存することに同意した場合は、後でいつでもスケッチの作業を続けることができます。

ダウンロードが成功した後、 ftドゥイーノ 目に見える反応は見られません。スケッチは、2つのいわゆる関数で構成されています。設定（） と名前の1つ ループ（）。 これらの関数は、すべてのArduinoスケッチのスケルトンを形成します。それぞれに、スケッチの実際の機能を説明する中括弧{および}のペアに埋め込まれた命令が含まれています。連続するステートメントはセミコロン（;）で区切られます。

ただし、この単純な例のように、括弧の間に指示がまったくない場合、スケッチは結果として、認識可能な反応をトリガーしません。 ftドゥイーノ アウト。

4.3.1コメント

//で始まる最初に削除された2つの線も、スケッチに機能を追加しません。もう一度確認できます
ファイル新し今回はスケッチを変更せずに残します。

```
空所 設定 () {
  //セットアップコードをここに配置して、1回実行します。

}

空所 ループ () {
  //メインコードをここに配置して、繰り返し実行します。

}
```

ダウンロードボタンをもう一度クリックすると、このスケッチが翻訳され、 ftドゥイーノ ロード。再び行われますftドゥイーノ 認識できる機能はありません。これは、追加の2行が純粋なコメント行であるためです。それらは、人間の読者に追加の説明を提供することを目的としています。これらの行は、マシンコードへの変換には意味がありません。マシンコードが生成されるときに、二重スラッシュ//の後に一列に並んでいるものはすべて無視されます。

/ *および* /にコメントを含めることもできます。マシンコードの生成は、要素を閉じる* /の後でのみ再開されます。さらに、この種のコメントは数行にまたがることがあります。

```
空所  設定 () {
  / *  複数行コメント
       可能です* /

}
```

Arduino IDEの配色は、コメントのヒントを提供します。コメントは常に薄い灰色で表示されるため、実際のプログラムコードと簡単に区別できます。

4.3.2エラーメッセージ

次に、2つのプログラムステートメントを中括弧の間に配置する必要があります。 設定（）-挿入する関数：

```
1  空所 設定 () {
2    pinnMode ( (LED_BUILTIN 、 出力) ;
3    digitalWrite ( (LED_BUILTIN 、 高い) ;
4位 }
5
6日 空所 ループ () {
7日 }
```

これらの2つの不可解な行が正確に何を意味するかについては、後で説明します。まず第一に、スケッチは再度翻訳され、に転送されるだけです。ftドゥイーノ 課金されます。プログラムをエラーなしでコンパイルできる場合は、次のように更新されます。ftドゥイーノ 移行。

残念ながら、この場合、エラーが発生することはありません。プログラムが示されているとおりに入力された場合、プログラムのコンパイルは図4.5に示されているエラーメッセージで中止されます。

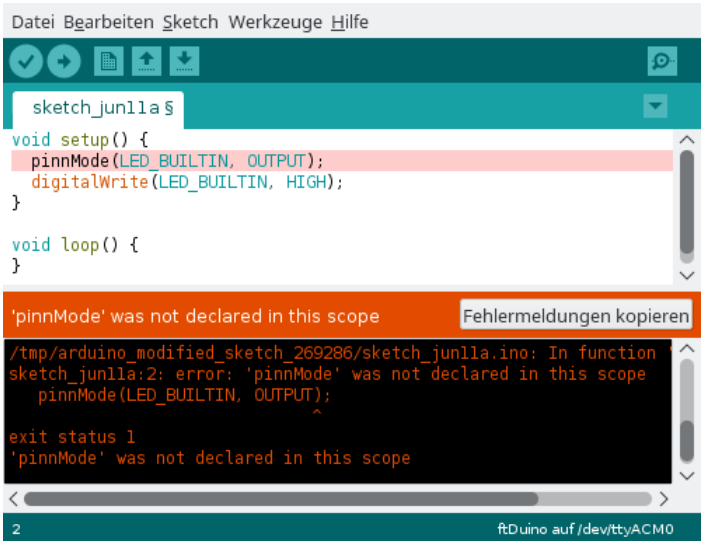


図4.5：変換エラーの表示

画面の上部で、エラーのあるコード行が強調表示されます。GCCコンパイラの出力は、ウィンドウの下部に表示されます。実際のエラーメッセージは 'pinnMode 'はこのスコープで宣言されていません。

これにより、コンパイラは、用語で始まったことを通知します pinnMode 何もできない。スケッチに入ると、この問題の兆候がありました。pinnMode たとえば、2行目の単語がIDEに入力されたときにのみ黒で表示されました digitalWrite オレンジ色で強調表示されます。Arduino IDEは、認識しているほとんどの命令に色を付けており、色の付いていない部分はエラーを示している可能性があります。

実際、ここに間違いが忍び込んで起こった pinnMode それを持っているだろう pinMode 呼び出す必要があります。それに応じて単語を変更すると、IDEは予想どおりオレンジ色になり、翻訳は成功し、ダウンロードが可能になります。修正されたスケッチは次のようになります。

```
1  空所 設定 () {
2    pinMode ( (LED_BUILTIN 、 出力) ;
3    digitalWrite ( (LED_BUILTIN 、 高い) ;
4位 }
5
6日 空所 ループ () {
7日 }
```

ダウンロード後、 ftドゥイーノ 輝く、それはまさに2つの不可解な線が行うことであり、これについては次のセクションで詳しく説明します。

4.3.3機能

最も重要なC++-言語要素は、いわゆる関数です。それらは、意味のある単純なプログラムスケッチを書くために使用できます。

関数定義は命令を要約します。次のプログラムフラグメントは、と呼ばれる関数の定義を示しています。姓、 機能本体の2つの指示 命令1 と 命令2 が含まれています。指示の後にセミコロンが続きます。互いに分離しました。

```
空所 姓 () {
  命令1;
  命令2;
}
```

関数は通常、複雑なタスクに必要なすべての命令を要約し、それに適切な名前を付けるために使用されます。関数本体のステートメントは、上から下に次々に実行されます。したがって、この場合のみ命令1 実行されてから 命令2。

個々の命令の実行には数マイクロ秒しかかからないため、その複雑さにもよりますが、同時に起こっているという印象を与えることがよくあります。実際、すべての命令は順番に実行されます。したがって、矛盾する指示も互いに矛盾する可能性があり、例えば、ランプをオンにしてからすぐに再びオフにすることができる。これらのイベントは次々に発生するため、ユーザーは、たとえば、発光ダイオードが数マイクロ秒オンになったことを確認できません。

命令の実行は直接的な影響を与える可能性があり、たとえば、発光ダイオードは ftドゥイーノ 点灯させてください。たとえば、発光ダイオードをオンにするためのすべての命令を要約する関数は、次のようになります。

```
空所 LEDをオンにします () {
  pinMode (LED_BUILTIN、出力);          //ピンを出力に切り替えます//出力ピンを
  digitalWrite (LED_BUILTIN、高い);      「high」に設定します
}
```

C言語の関数++ は数学関数に非常に密接に基づいており、これらのように、結果を提供できます。関数が返すかどうか、およびどのような結果が返されるかは、関数名の前にあります。この場合、結果は返されません。そのため、関数名がその前に配置されます。空所、 何のための英語。さらに、関数は、丸括弧（ () ）で指定された、処理される1つ以上の入力値を受け取ることができます。現在の関数は入力値を必要としないため、角かっこは空のままです。

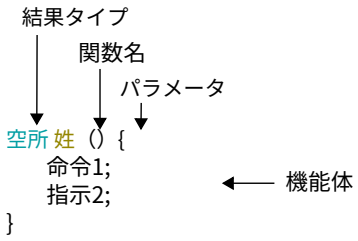


図4.6：関数の定義

関数の定義には、コンパイラが対応する命令を変換し、必要なマシンコマンドを次々にマシンコードに格納するという効果があります。ただし、これらの指示が実際にいつ実行されるかについては何も述べていません。

関数の命令は、関数が呼び出されるとすぐに実行されます。あなたがしなければならないのは、命令としてパラメータとともに関数名を入力することです。機能があるのでLEDをオンにします パラメータは予期されていません。丸括弧の間の領域は、呼び出されたときに空のままです。関数呼び出し自体は、関数定義の関数本体に含まれている必要があります。

```
空所 設定 () {
  LEDをオンにします ();
}
```

これにより、指示が明確になります pinMode（LED_BUILTIN、OUTPUT）; と digitalWrite（LED_BUILTIN、HIGH）; 関数呼び出しもありました。両方の関数には、コンマで区切られた2つのパラメーターが与えられました。

4.3.4機能 設定（） と ループ（）

https://www.arduino.cc/reference/en/language/structure/sketch/setup/
https://www.arduino.cc/reference/en/language/structure/sketch/loop/

関数自体が他の関数からしか呼び出せない場合、最初の関数呼び出しがどこから行われるかという問題に関しては、鶏が先か卵が先かという問題が発生します。

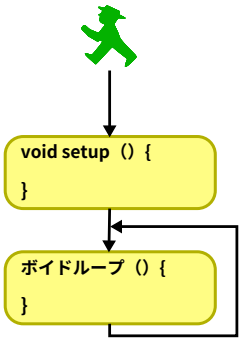


図4.7：スケッチのコース

これが2つの関数定義の出番です 設定（） と ループ（） すべてのスケッチに少なくとも含まれている必要があります。一方または両方の定義が欠落している場合、ArduinoIDEはエラーメッセージを表示して変換を中止します。

両方の関数を明示的に呼び出す必要はありません。代わりに、スケッチの変換中に、ArduinoIDEによって生成されたマシンコードにそれらの呼び出しが自動的に挿入されます。関数設定（） スケッチの開始時に1回呼び出され、関数 ループ（） 図4.7に示すように、は何度も呼び出されます。

の場合 ftドゥイーノ 次の3つの状況のいずれかを意味します。

1.翻訳されたスケッチをダウンロードした直後に、そのコードが開始されます。

2.電源を供給した後 ftドゥイーノ 以前にダウンロードしてインストールしたスケッチコードを起動します。

3.のブートローダーがあります ftドゥイーノ リセットボタンを押すことで開始し、8秒間アクティブのままになります。この間にブートローダーがPCによってアドレス指定されない場合、ブートローダーは終了し、代わりに最後にダウンロードされたスケッチコードが開始されます。

> 経験豊富なユーザーへの注意
>
> Cを始めたことがある人++ PCなどでは、この時点で驚かれるかもしれません。その場合、特別な機能がありました設定（） と ループ（） いいえ。代わりに、という関数がありました主要（）、 これは、プログラムの開始時に自動的に呼び出されました。
>
> Arduino IDEの開発者は、この時点で通常の標準から逸脱することを決定しました。 主要（）-関数は、複雑なコンピューター上で一時的にのみ実行され、ハードウェアプログラミングの世界に限られた範囲でしか適合しないプログラムのために開発されました。

ライブラリ関数

の場合のように 姓（）-関数を実行すると、独自の関数を作成できます。ArduinoIDEにはすでに独自の関数コレクションがあります。コレクションは、主に頻繁に使用されるユニバーサル関数で構成されています。

それらは最初から名前でシステムに認識されており、自分で定義しなくても、独自のプログラムで呼び出すことができます。

機能 pinMode（） と digitalWrite（） そのようなライブラリ関数です。言語Cが++ はユニバーサルであり、PCまたはArduinoで使用しても違いはありません。ライブラリ関数は通常、プラットフォーム固有です。関数pinMode（） はArduinoのプログラマーのみが利用でき、Windows PC用のプログラムを開発する場合、この関数を呼び出すとコンパイルエラーが発生します。

これが多くのCが++-例やチュートリアルをインターネットから直接Arduinoに転送しないでください。これらのプログラムがまだArduinoプログラムではなくPCプログラムである場合は、そこで関数ライブラリが使用され、Arduinoでは使用できません。これは主に、さまざまなプラットフォームのハードウェアが大幅に異なるためです。PCプログラムは主に画面上のウィンドウを開き、ユーザー入力を処理しますが、Arduinoプログラムはハードウェア入力とスイッチ出力を評価する可能性が高くなります。

> 経験豊富なユーザーへの注意
>
> ライブラリを使用する場合の通常のCとの違いもあります++-PCでのプログラミング。Arduino IDEは、基本的なArduino固有のライブラリをスケッチで自動的に認識し、次のような機能を実現します。pinMode（） 直接使ってみましょう。
>
> 共通C++-コンパイラは、ライブラリを単独で統合しません。いわゆる＃を使用する必要があります含む- 指示は明示的に知られています。これは、少なくとも提供されているライブラリのほとんどでは、Arduinoでは必要ありません。

4.3.5例

これにより、基本的な基本事項が説明され、次の例がわかりやすくなります。

```
 1  / * LEDをオンにする機能* / 空所 LEDをオンにします （）
 2  {
 3     pinMode（（LED_BUILTIN、 出力）；
 4位   digitalWrite（（LED_BUILTIN、      高い）；
 5  }
 6日
 7日 空所 設定（）{
 8日    LEDをオンにします （）；
 9  }
10
11日 空所 ループ（）{
12日 }
```

システムが起動すると、 設定（）-関数が呼び出されました。関数設定（） 関数の呼び出しの形式で単一の命令が含まれています LEDをオンにします。 この関数は、スケッチの2行目から5行目で定義されており、2つのライブラリ関数を呼び出して、 ftドゥイーノ 点灯させてください。

the ループ（）-関数は、システムの起動後に常に再度呼び出されます。ただし、空であるため、システムの起動時にLEDが直接オンになっている場合、スケッチにはそれ以上の機能は表示されません。

1行目のコメントは説明のみを目的としており、スケッチの翻訳やその機能には影響しません。

4.4役立つライブラリ関数

前のセクションで示したように、Arduino IDEには、Arduinoの特別なプロパティを使用するための関数を提供するいくつかのライブラリが付属しています。加えてftドゥイーノ アクセスするための独自のライブラリのインストール
Schertechnikの入力と出力 ftドゥイーノ アクセスするために。

最も一般的な7つの機能を以下に説明します。あなたはすでに彼らと一緒に様々なスケッチを書くことができます。

Arduino IDEの他の機能の説明は、Arduino言語リファレンスでオンラインで見つけることができます https://www.arduino.cc/reference/en/#functions さらに ftドゥイーノ-特定の機能については、第9章で説明しています。

4.4.1 pinMode（ピン、モード）

https://www.arduino.cc/reference/en/language/functions/digital-io/pinMode/

この関数は、ATmega32u4マイクロコントローラーのピンを入力として構成します（モード=入力） または終了（モード=出力）。 この機能は、マイクロコントローラーの接続を直接操作するため、Arduinoで非常に頻繁に使用されます。にftドゥイーノ に接続するためのほとんどの接続に追加の回路があります
の直接使用を可能にするschertechnikコンポーネント pinMode（）-機能を不要にします。代わりに、彼らはもたらす ftドゥイーノ-せん断技術に従って入力と出力を操作するためのすべての機能を備えたライブラリ。

主な例外は、内部の赤色発光ダイオードの接続です。にとってピンコード 場合に必要 LED_BUILTIN 利用される。

```
//発光ダイオードの内部接続ピンを出力として宣言します pinMode（（LED_BUILTIN、 出
力）；
```

4.4.2 digitalWrite（ピン、値）

https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

関数 digitalWrite（） 機能を通してあなたを操縦します pinMode（） ピンが出力として宣言されました。したがって、ftドゥイーノ 通常、発光ダイオードに使用され、まれにIの2つを制御するために使用されます。2Cコネクタの利用可能な信号。

にとって ピンコード と同じ値が適用されます pinMode（）-関数。の値価値 できる 高い また 低い 対応するピンをオンまたはオフにするもの。

```
//発光ダイオードの内部接続ピンを出力として宣言します pinMode（（LED_BUILTIN、 出
力）；//発光ダイオードをオンにします digitalWrite（（LED_BUILTIN、 高い）；
```

4.4.3 遅延（ミリ秒）

https://www.arduino.cc/reference/en/language/functions/time/delay/

ATmega32u4のような単純なマイクロコントローラーでさえ、人間が知覚できるよりも速くほとんどのタスクを実行します。プロセスを適切なレベルに減らすために遅れ （） -役立つ機能。パラメータとしてミリ秒単位の待機時間を想定しています。

セクション4.3.4で説明されているように、関数 設定（） 1回だけ呼び出される関数 ループ（） しかし、何度も何度も。したがって、次のスケッチでは、発光ダイオードが連続的に点滅します。

```
空所 設定（）{
  //発光ダイオードの内部接続ピンを出力として宣言します pinMode（（LED_BUILTIN、 出
  力）；
}

空所  ループ（）{
  //  LEDをオンにします
  digitalWrite（（LED_BUILTIN、 //1  高い）；
  秒待ちます 遅れ （1000）；

  //発光ダイオードをオフにします
  digitalWrite（（LED_BUILTIN、 //1  低い）；
  秒待ちます
```

```
    遅れ（1000）；
}
```

4.4.4 Serial.begin（速度）

https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/

の通信オプション ftドゥイーノ 基本的に、デバイス上で直接せん断技術の入力と出力に制限されます。より複雑なプロジェクトでは、ftドゥイーノ プログラムをフォローします。スケッチ内からユーザーに情報を出力する簡単な方法は、シリアルとしょうかん。

PC側では、このライブラリの出力は、いわゆるシリアルモニターを介して表示されます。その使用法については、セクション3.3.1で詳しく説明しました。関数Serial.begin（） シリアルモニターを使用するためのスケッチを準備します。したがって、スケッチの最初または設定（）-関数を呼び出すことができます。

関数 Serial.begin（） 次の名前のパラメータが必要です 速度。 この値は、 ftドゥイーノ 無関係であり、たとえば、115200の値に設定する必要があります。

```
空所 設定 （）{
    //シリアル接続を準備します シリアル。始める
     （115200）；
}

空所 ループ （）{
}
```

4.4.5 Serial.print（val） と Serial.println（val）

https://www.arduino.cc/reference/en/language/functions/communication/serial/print/
https://www.arduino.cc/reference/en/language/functions/communication/serial/println/

経由のシリアル接続です Serial.begin（） 関数が機能するように設定します Serial.print（） と Serial.println（） シリアルモニターにメッセージを出力するために使用できます。の違い Serial.print（） と Serial.println（） を介して出力した後という事実にあります Serial.println（） 新しい出力行が開始され、その後にさらに出力が開始されます Serial.print（） 同じ行で直接実行します。関数 Serial.println（） したがって、より複雑な問題を解決するためによく使用されます。

にとって val とりわけ、文字列と数字を使用できます。文字列は二重引用符（）で囲む必要があります。

```
空所 設定 （）{
    //シリアル接続を準備します シリアル。始める
     （115200）；
    // PCに接続を受け入れる時間を与えるための2秒の遅延//

    遅れ （2000）；
    //いくつかの問題
    シリアル。印刷（（「答えは：」）；シリアル。println
     （42）；
}

空所 ループ （）{
}
```

4.4.6 ftduino.input_get（）、ftduino.output_set（） と ftduino.motor_set（）

もちろん、Schertechnikモデルのコントローラーは、モデルから入力を受け取り、モデル内の反応をトリガーする必要があります。theftドゥイーノ-ライブラリについては、第9章で詳しく説明しています。したがって、この段落では最低限のことだけを説明します。

セクション4.3.4で、ライブラリ関数が追加の前提条件なしで使用可能であり、スケッチで使用できることが特別な機能であると説明されました。これは、前の例で使用したArduino自体のライブラリにのみ適用されます。theftドゥイーノ-ライブラリは自動的に使用可能ではありませんが、スケッチの先頭に＃を付ける必要があります含む-公表される指示。

```
# 含む <FtduinoSimple.h>

空所 設定 () {
}

空所 ループ () {
}
```

そうして初めて、このライブラリの関数をスケッチで使用できます。

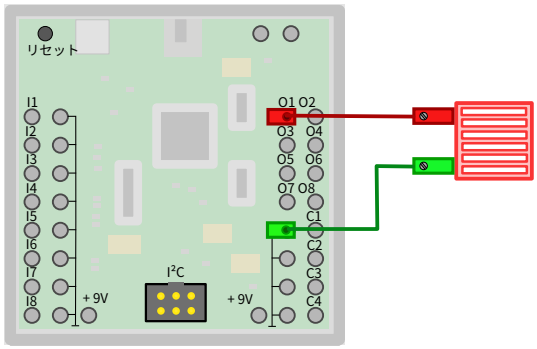


図4.8：出力O1のランプ

このための最も重要な機能は次のとおりです ftduino.input_get（ポート） 入力を照会し、 ftduino.output_set（ポート、モード） 出力を切り替えます。でftduino.input_get（ポート） のためです ポート の値 Ftduino :: I1 それまで Ftduino :: I8 許可されます。出力を切り替える機能があります ftduino.output_set（ポート、モード）、 それによって ポート の値 Ftduino :: O1 それまで Ftduino :: O8 受け入れるかもしれません ファッション の上 Ftduino :: HI 出力をオンにしてに切り替えるときに設定されます Ftduino ::オフ 出力をオフに切り替えるように設定されています。

```
# 含む <FtduinoSimple.h>

空所 設定 () {
   //入力I1を読み取ります
   ftduino。input_get （（Ftduino ：：：I1）；//
   出力O1をオンにします ftduino。output_set
    （（Ftduino ：：：O1、                    Ftduino ：：：こんにちは）；
}

空所 ループ () {
}
```

このスケッチは、図4.8に示すように接続されたランプを点灯します。注意：これを行うには、 ftドゥイーノ 9ボルトで供給されます。

関数 ftduino.motor_set（ポート、モード） は ftduino.output_set（ポート、モード） 非常に似ていますが、ここにあるのはエンジン出力だけです M1 それまで M4 切り替えることができます。の値ファッション その後できます Ftduino ::左、Ftduino ::右 また Ftduino ::オフ モーターを反時計回りに回すか、時計回りに回すかによって異なります。

```
# 含む <FtduinoSimple.h>

空所 設定 () {
   //入力I1を読み取ります
   ftduino。input_get （（Ftduino ：：：I1）；
   // M1でモーターを反時計回りにオンにします ftduino。motor_set
    （（Ftduino ：：：M1、 Ftduino ：：：左）；
}

空所 ループ () {
}
```

4.5変数

https://www.arduino.cc/reference/en/#variables

多くの場合、スケッチに何かを記憶したり保存したりする必要があります。たとえば、特定の頻度で何かが発生した場合、それがすでに発生した頻度または発生しなければならない頻度を途中で記録する必要があります。もう1つの例は、ユーザーがボタンを押したなどの1回限りのイベントです。このイベントが通過し、ユーザーがボタンを離した場合でも、必要に応じてアクションを続行する必要があります。これを行うには、ボタンが最近押されたことに注意する必要があります。

この目的のためのいわゆる変数があります。これらは、コンパイラにメモリ内のスペースを割り当てるように指示するために使用されます。ftドゥイーノ 注目すべきことのために予約する。スケッチがいわゆるフラッシュメモリに保存されている間ftドゥイーノ が保存されるため、スイッチをオフにすることによっても ftドゥイーノ も保存されます。の大容量RAMメモリ内の変数のストレージスペースの場合ftドゥイーノ 提出した。変数の保存された値は、ftドゥイーノ 失った。

変数は、スケッチの関数のように定義されます。関数のような名前が付けられています。さらに、変数に格納するデータのタイプを指定する必要があります。次の例では、 variableName タイプのデータの場合 int 作成した。

```
int variableName;

空所 設定（）{
}

空所 ループ（）{
}
```

4.5.1データ型 int

https://www.arduino.cc/reference/en/language/variables/data-types/int/

データ型 int 標準のデータ型です。でftドゥイーノ このデータ型では、-32768〜 +32767の範囲の整数値を格納できます。これはほとんどの用途に十分です。

変数は、データを格納し、後で再度呼び出すために使用されます。値の保存は、変数に等号（=）の値を割り当てることによって行われます。割り当ての右側の式には、複雑な数学関数や関数呼び出しを含めることができます。たとえば、入力のステータスを判別する場合などです。ftドゥイーノ 保存する。

```
//単純な割り当て variableName = 42;

//複雑な式 variableName =（4 * 8 * 8 +
38）/ 7; //入力の状態I1

variableName = ftduino。input_get（（Ftduino ：：：I1）;
```

式には変数を含めることもできます。左側の変数は右側にも表示される可能性があり、方程式の数学的理解と矛盾します。

```
//別の変数から値を取得します variableName = 他の変数名;

//変数の値を変更し、それを変数に再割り当てします variableName = variableName + 1;
```

表現がそれを示唆しているとしても、このタイプの割り当てを数学的比較として読むべきではありません。代わりに、右側の式が最初に計算され、次に結果が左側の変数に割り当てられます。このタイミングは、次のような割り当てにつながります

```
variableName = variableName + 1;
```

理にかなっています。

変数は、関数呼び出しのパラメーターとしても使用できます。

```
int variableName;

空所 設定（）{
  シリアル。始める （115200）；
  variableName = 192/4;

  シリアル。印刷（（「可変コンテンツ： シリ 「」）；
  アル。println（（variableName）；
}

空所 ループ（）{
}
```

ここでは、引用符（）を正しく使用することが重要です。引用符で囲まれた単語またはテキストは、テキスト自体を表し、それ以上解釈されません。引用符のない単語は、命令、関数名、または変数名を表すことができ、コンパイラーはこの単語に意味を割り当てようとします。

```
//単語variableNameを出力します シリアル。
println（（「variableName」）；
// variablenNameという名前の変数の内容を出力します シリアル。println（（
variableName）；
```

4.6条件

これまでのところ、すべての例は固定された順序の命令で構成されています。すべての指示は常に同じ方法で実行され、たとえば、LEDが点滅したり、メッセージが発行されたりしました。しかし、他のイベントに応じて何も起こりませんでした。

ただし、ロボット制御および一般的なプログラミングでは、プログラムがイベントに反応する必要があることがよくあります。C++-このメカニズムは、条件に反応できる命令です。

4.6.1 もしも-命令

https://www.arduino.cc/reference/en/language/structure/control-structure/if/

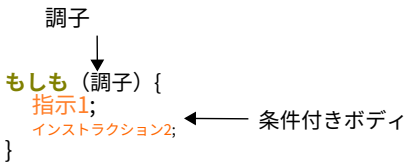


図4.9： もしも-命令

the もしも-命令はそのような命令です。彼女は括弧内の条件と上の条件を期待していますもしも-ステートメント条件本体の次のステートメントは、条件が真であることが判明した場合にのみ実行されます。

```
もしも（12> 5 + 3）
  シリアル。println（（「12は5と3の合計よりも大きい」）；
```

条件では、さまざまな比較演算子を使用できます。最も重要なものは次のとおりです。

C。++-表記	比較演算
>>	より大きい
<	未満
==	同じ
！=	等しくない

条件には、関数呼び出しを含めることもできます。の場合にはftduino.input_get（）-関数、関数呼び出しの結果はすでに真理値（trueまたはfalse）であり、関数呼び出しを直接使用できます。からの複数のステートメントが必要ですもしも-命令は影響を受けます。中括弧（{および}）を使用して要約できます。

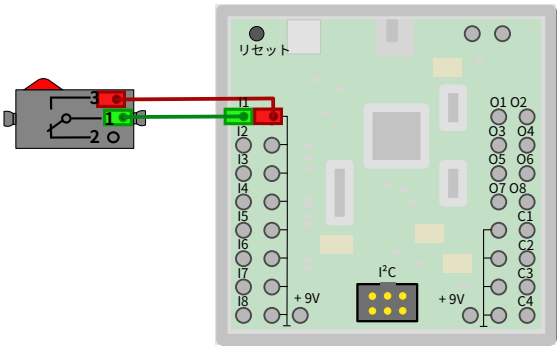


図4.10：入力I1のボタン

```
# 含む <FtduinoSimple.h>

空所 設定 () {
   //シリアル接続を準備します シリアル。始める
    (115200);
}

空所 ループ () {
   // I1のボタンが押されているかどうかをテストします もしも
    ( (ftduino。input_get ( (Ftduino ： ： I1) ) {
      シリアル。印刷 ( ( 「ボタンが押されました」 ) ; // 1/4秒待
      ちます 遅れ (250);

   }
}
```

4.7研削

プログラムループも非常に基本的な概念です。それらを通してのみ、プログラムの一部を数回実行することが可能です。これがないと、プログラムのすべての命令が次々に処理されます。

ただし、前の例では、何かを数回実行するプログラムがすでに1つまたは他にありました。これはArduino固有のものによるものですループ（）-関数。図4.7に示すように、プログラムシーケンス中に何度も自動的に呼び出されます。これは、対応するCを使用せずにArduinoスケッチも使用できることを意味します++-プログラム部分で指示を繰り返します。

それにもかかわらず、Cも見ると役に立ちます++-ループの独自のメカニズムにフォールバックできるようにする。そのうちの2つを以下に説明します。

4.7.1 その間-リボン

https://www.arduino.cc/reference/en/language/structure/control-structure/while/

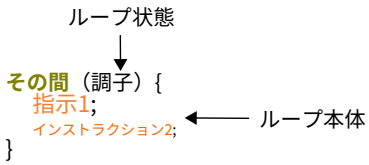


図4.11： その間-リボン

the その間-ループを使用すると、丸括弧の間の条件が満たされている限り、いわゆるループ本体で次のコマンドを繰り返すことができます。と同じようにもしも-ステートメントには、次のステートメントのいくつかを含めることができます

中括弧を使用して要約できます。theその間-次に、ループはステートメントブロック全体に適用されます。最初から条件が満たされない場合、ループ本体は実行されません。

```
その間（（variableName <12）{
  シリアル。println（（「変数は12未満です」）; variableName
  = variableName + 1;
}
```

状態も同じです もしも-ステートメントであり、同じ演算子を含めることができます。次の例は、のセクションから例を拡張したものです。もしも-キーが解放されるのを待つための指示。

```
# 含む <FtduinoSimple.h>

空所 設定（）{
  //シリアル接続を準備します シリアル。始める
   （115200）;
}

空所 ループ（）{
  // I1のボタンが押されているかどうかをテストします もしも
   （（ftduino。input_get（（Ftduino：：：I1））{
    シリアル。印刷（（「ボタンが押されました」）; // 1/4秒待
    ちます 遅れ （250）;


    //キーが解放されるのを待ちます その間（（ftduino。input_get（（
    Ftduino：：：I1））{
      //中括弧の間のスペースは、//中の場合、完全に空のままにすることもできま
      す
      //繰り返しこれ以上の命令は実行されません//

    }
  }
}
```

4.7.2 にとって-リボン

https://www.arduino.cc/reference/en/language/structure/control-structure/for/

それより少し複雑 その間-ループは にとって-リボン。丸括弧で囲まれたセミコロンで区切られた3つのステートメントが含まれています。1つ目はループが始まる前に一度実行され、2つ目はループの実行中に評価され、その間-ループが実行される頻度をループします。3番目のステートメントは最終的に後 ループ本体のすべての実行を実行しました。最初から条件が満たされない場合、ループ本体は実行されません。事前申告は常に実行されます。

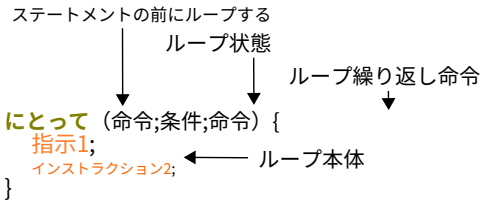


図4.12： にとって-リボン

非常に複雑に聞こえるものは、通常のアプリケーションを見ると理解できるようになります。 にとって-ループは次のことを確認します。特定の頻度でのコマンドの繰り返し。

```
にとって（（variableName = 0; variableName <12; variableName = variableName + 1）
  シリアル。println（（「このテキストは12回出力されます」））;
```

括弧内の3つのステートメントまたは条件は次のとおりです。

```
variableName       =  0;
variableName       <  12;
variableName = variableName + 1;
```

最初のステートメントは、ループの開始時に1回だけ実行されます。この場合、値0を変数に書き込みます variableName 2番目のステートメントは、ループが実行される頻度を決定する条件です。この場合、変数の内容がvariableName 12未満です。そして、3番目のステートメントは最終的に各実行の終わりに与えられますにとって-ループが実行されました。この場合、変数の内容がそこに表示されますvariableName 1つ増えました。したがって、この例では：

1.変数の内容 variableName 0に設定されます

2.変数の内容が variableName 12未満は...

（a）...ループ本体の命令が実行されます...

（b）...そして変数の内容 variableName 1つ増えました

ループ本体は正確に12回実行されます。

4.8例

C++-言語構成、およびArduinoとの選択された機能 ftドゥイーノ-ライブラリはごく一部にすぎません。しかし、この小さな情報でさえ、いくつかの興味深いスケッチを自分で書くのに十分です。

4.8.1単純な信号機

この例は、単純なデマンド信号機を示しています。スイッチをオンにすると、最初は赤で表示されます。ボタンを押すとすぐに10秒間緑色にジャンプし、その後初期状態に戻ります。

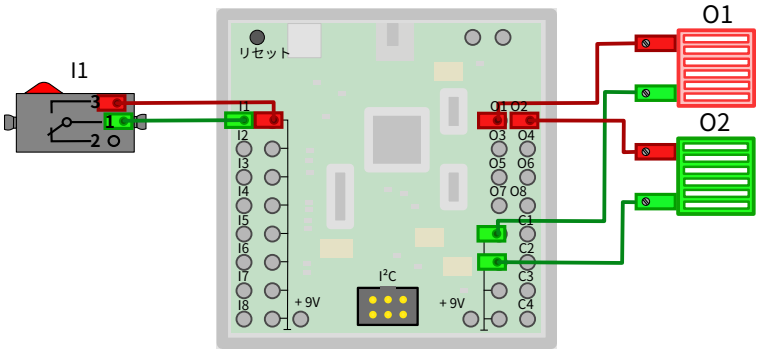


図4.13：単純な信号機

添付のスケッチは非常に単純です。の中に設定（）-機能、5行目で赤いランプが点灯します。

の中に ループ（）-ボタンが押されているかどうかにかかわらず、機能は10行目で永続的にテストされます。押すと、11行目から20行目の条件付き本文全体が実行されます。

そこで、12行目で赤いランプがオフになり、14行目で緑のランプがオンになります。16行目は10,000ミリ秒、つまり10秒待機してから、18行目と20行目で緑色のランプが最初にオフになり、次に赤色のランプがオンになります。

これは、マイクロプロセッサの速度のために同時であると認識されるものの例です。12行目と14行目または18行目と20行目のランプは次々にオンとオフが切り替えられますが、時間的な違いは見られません。数マイクロ秒の距離は見えません。

```
1   # 含む <FtduinoSimple.h>
2
3   空所 設定 () {
4位    //信号が始まると、赤いランプが点灯します ftduino。output_set（（
5       Ftduino：：：O1、 Ftduino：：：こんにちは）；
6日 }
7日
```

```
 8日 空所 ループ （） {
 9       // I1のボタンが押されているかどうかをテストします もしも （（
10       ftduino。input_get （（Ftduino ：：：I1））            {{
11日        //赤いランプを消します ftduino。output_set
12日         （（Ftduino ：：：O1 、 //緑色のランプをオ    Ftduino ：：：オフ）；
13日        ンにします ftduino。output_set （（Ftduino
14日        ：：：O2 、 // 10秒待ちます 遅れ （10000）；   Ftduino ：：：こんにちは）；
15日
16
17日        //緑色のランプをオフにします ftduino。
18日        output_set （（Ftduino ：：：O2 、 //赤いラ    Ftduino ：：：オフ）；
19日        ンプをオンにします ftduino。output_set （（
20日        Ftduino ：：：O1 、                          Ftduino ：：：こんにちは）；
21       }
22日 }
```

4.8.2バリア

バリアの例はもう少し複雑です。これは、それぞれ2つのリミットスイッチを備えた電動バリアで構成されています。上のボタンI2 バリアが完全に開いたときにアクティブになります。 I3 バリアが完全に閉じられるとアクティブになります。

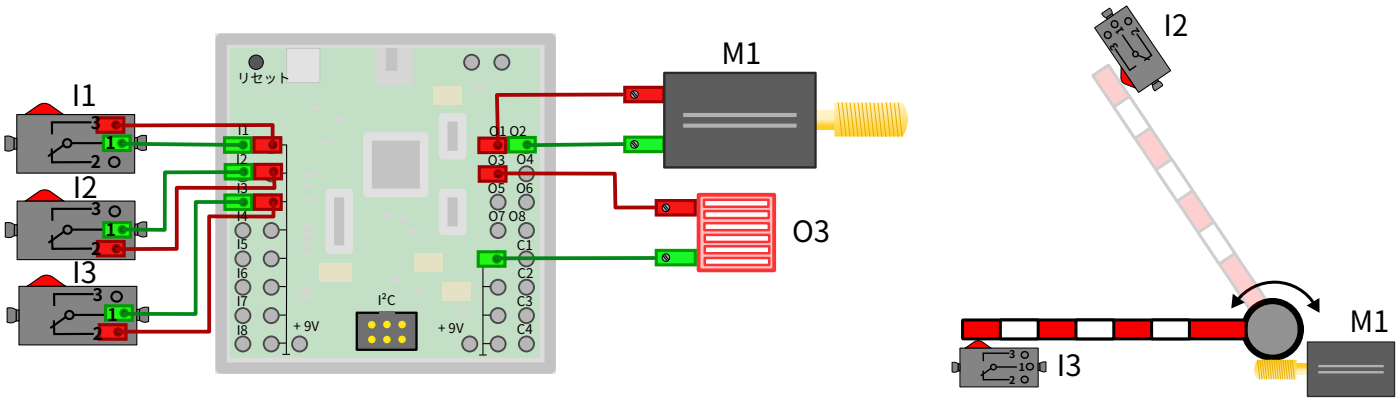


図4.14：バリア

注意：ボタン I2 と I3 作動していないときに接点が閉じるように配線されています。ボタンを押すとすぐに開きます。つまり、バリアが完全に開いているか閉じています。

スケッチが始まると、ボタンの接点がオンになっている限り、10行目のモーターが最初に左に回転します。 I2 15行目で閉じていると認識されます。つまり、バリアが完全に開いていない場合に限ります。ボタンが押されるとすぐに、モーターは18行目で停止します。

ボタンがオンになったら I1 23行目で押されると、今度はボタンがオンになるまで時計回りに26行目でモーターが始動します。 I3 が動作します。その後、モーターは28行目で停止します。

31〜40行目でランプが点灯します O3 毎回500ミリ秒の休止で5回オンとオフを切り替えました。

その後、43行目から45行目でバリアが最終的に再び閉じられます。

```
 1   # 含む <FtduinoSimple.h>
 2
 3   //点滅するカウンター変数 int
 4位        カウンター;
 5
 6日 空所   設定 （） {
 7日    //開始時にバリアが開きます
 8日
 9       //モーターを反時計回りに回します
10       ftduino。motor_set （（Ftduino ：：：M1 、 Ftduino ：：：左）；
11日
12日    //バリアが閉じて開くまで待ちます。ボタンはとして利用できるので
```

```
13日    // NC接点が配線されており、ボタンが閉じている限りモーターは//動作します
14日
15日    その間（（ftduino。input_get（（Ftduino：：：I2））{}
16
17日    //ボタンが閉じなくなったら、モーターを停止します ftduino。motor_set（（Ftduino
18日    ：：：M1、 Ftduino：：：オフ）；
19日 }
20日
21   空所   ループ（）{
22日    // I1のボタンが押されているかどうかをテストします もしも
23       （（ftduino。input_get（（Ftduino：：：I1））{
24
25日       //ボタンI3が開くまでモーターを時計回りに回します ftduino。motor_set（（
26日       Ftduino：：：M1、 Ftduino：：：正しい）；その間（（ftduino。input_get（（
27         Ftduino：：：I3））{} ftduino。motor_set（（Ftduino：：：M1、 Ftduino
28         ：：：オフ）；
29
30日       //ランプを5回点滅させます
31         にとって（（カウンター= 0; カウンター<5; カウンター=カウンター+1）        {{
32            //ランプが点灯
33            ftduino。output_set（（Ftduino：：：O3、   Ftduino：：：こんにちは）；
34            // 500ミリ秒待つ
35            遅れ（500）；
36            //ランプオフ
37            ftduino。output_set（（Ftduino：：：O3、   Ftduino：：：オフ）；
38            // 500ミリ秒待つ
39            遅れ（500）；
40         }
41
42         //ボタンI2が開くまでモーターを反時計回りに回します ftduino。motor_set（（
43         Ftduino：：：M1、 Ftduino：：：左）；その間（（ftduino。input_get（（
44         Ftduino：：：I2））{} ftduino。motor_set（（Ftduino：：：M1、 Ftduino
45         ：：：オフ）；
46      }
47   }
```

4.9警告 少しの記憶

大規模なプロジェクトをプログラミングする場合、特にライブラリを多用する場合、図4.15に示す警告メッセージが簡単に表示される可能性があります。

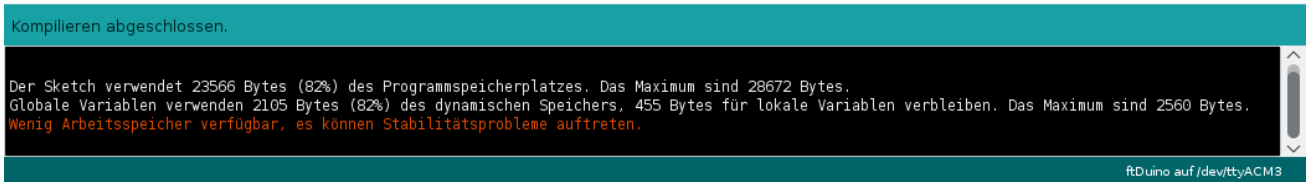


図4.15：メモリ不足に関するArduinoIDEの警告

以来 ftドゥイーノ 2560バイトの動的メモリ（RAMメモリ）しかないため、この不足しているリソースを処理するのは簡単ではありません。

ワーキングメモリの不足に関する警告は、ダイナミックRAMメモリと いいえ メッセージにも記載されているプログラムメモリスペース（フラッシュメモリとも呼ばれます）。プログラムメモリは、ためらうことなく100%まで満たすことができます。

4.9.1影響

のフラッシュメモリ内 ftドゥイーノ 2つのプログラムがいつでも保存されます。

ブートローダー ブートローダー（セクション1.2.1を参照）は、フラッシュメモリの保護された部分に保存されます。彼はスケッチをArduinoIDEからUSB経由でフラッシュメモリの残りの部分にロードするために使用されます。

スケッチ スケッチは、Arduino IDEとブートローダーの助けを借りてユーザーがインストールし、すべてを行うことができます
ブートローダーが使用していないフラッシュメモリを使用してください。の中にftドゥイーノ ブートローダーが占める領域に加えて、28672バイトのフラッシュメモリを自分のスキットに使用できます。

メッセージは、両方のプログラムで（異なる）意味を持ちます。

スケッチへの影響

スケッチに対するRAMの不足の正確な影響を予測することは困難です。動的メモリの不足は、スケッチの実行中に追加の動的メモリが何度も必要になるため、問題になります。たとえば、計算の中間結果を保存したり、サブ機能が実行された後にスケッチの実行を続行する必要がある場所を保存したりするためです。と呼ばれる。中間結果または継続ポイントは、メモリの不足によって改ざんされます。プログラム全体の実行は、完全に予期しない無意味な反応につながる可能性があります。

このような欠陥のあるスケッチは、プログラムの正しい実行を復元するために、いつでも修正されたスケッチに置き換えることができます。

疑わしい場合は、最初に単純で既知の機能スケッチをインストールする必要があります。下のまばたきスケッチ
ファイル例 。01.基本 。点滅 これに役立ちます。

ブートローダーへの影響

ブートローダー自体は、Flashの独立したプログラムです。このメモリをスケッチと共有する必要がないため、動的メモリのボトルネックの影響を受けません。ただし、各スケッチ内にはブートローダーに属する小さな機能ブロックがあり、ArduinoIDEによってユーザーのスケッチのコードに目に見えない形で統合されています。

この部分は、スケッチの実行中にPCとのUSB通信を実現します。とりわけ、この部分は、スケッチのアップロードの準備としてブートローダーを開始するためにArduinoIDEによって送信されたコマンドに反応します。実行中のスケッチのこの部分で、Arduino IDEが新しいスケッチを転送したいと判断した場合、ブートローダーがアクティブになります。次に、ブートローダーが新しいスケッチの実際の受信を処理します。

動的メモリが不足している場合、スケッチのこの非表示部分が正しく機能しない可能性があります。Arduino IDEはブートローダー自体の起動を開始できなくなり、ArduinoIDEによるアップロードの試行は失敗します。PCへのUSB通信が非常に損なわれている可能性がありますftドゥイーノ スケッチの実行中に、PCによって正しく認識されないか、まったく認識されません。

この場合、 ftドゥイーノ リセットボタンを介して（セクション1.2.3を参照）。ArduinoIDEが使用できなくなった場合ftドゥイーノ USB経由でアドレス指定し、ブートローダーをアクティブ化する場合でも、セクション1.3.2で説明されているように、リセットボタンを使用してブートローダーを手動でアクティブ化することができます。リセットボタンの機能は常に利用可能であり、誤ったスケッチの影響を受けることはありません。その助けを借りて、上に新しいスケッチを作成することは常に可能ですftドゥイーノ ロードする。

4.9.2予防措置

上記の警告が表示された場合、疑わしい場合は、上のスケッチを使用しないでください。 ftドゥイーノ ロードする。リセットボタンを使用して障害のあるスケッチをいつでも置き換えることができる場合でも、この手順には適切なタイミングが必要であり、頑固なスケッチが正常に置き換えられるまで数回の試行が必要になる場合があります。

一定のデータのためのフラッシュメモリの使用

多くのスケッチでは、テキスト出力は、シリアルモニターを介して、またはたとえば小さなディスプレイで行われます。したがって、スケッチには通常、次のような線が含まれます。

```
シリアル。println（（"こんにちは世界！"）；
```

12バイトの貴重なダイナミックRAMメモリが不必要に使用されていることは明らかではありません。これは、最小限のスケッチで簡単にテストできます。

```
1   空所 設定 （） {
2       シリアル。始める （9600）；
3   }
4位 空所 ループ （） {
5       シリアル。println （（"こんにちは世界！"）；遅
6日     れ （1000）；
7日 }
```

Arduino IDEによると、このスケッチは163バイトまたは動的メモリの6％を占めます（正確な値はArduino IDEのバージョンによってわずかに異なる場合があります）。5行目に接頭辞//を付けてコメントアウトすると、メモリ消費量が149バイトに削減されます。つまり、14バイト全体の動的メモリが節約されます。

その理由は、ArduinoIDEが文字列HelloWorld！を想定しているためです。さらに処理する必要があります。したがって、ArduinoIDEは文字列HelloWorld！を作成します。スケッチによって変更できるダイナミックメモリ内。ただし、スケッチの実行中にこの文字列を変更する予定はないため、フラッシュメモリに残しておくこともできます。これはまさにArduinoIDEがヘルプ機能に提供するものです。シンプルなもの F（...） 文字列の周りはまさにそれを行います。これを行うには、前のhelloworld出力を次の構文に置き換えます。

```
シリアル。println （（F。 （（"こんにちは世界！"））；
```

このスケッチは前のバージョンと同じように動作しますが、163バイトではなく151バイトしか占有しません。違いは、文字列Hello World！の長さに正確に対応します。さらに、文字列を終了する別のバイト。多くのスケッチでは、これによりすでにかなりの量の動的メモリが節約されています。

ただし、フラッシュメモリの取り扱いはそれほど簡単ではありません。次の例では、別のヘルパーメカニズムを使用しています。プログラム 文字列をフラッシュに保存します。残念ながら、これは単純に行うことはできません println（） 出力。

```
// ftDuinoはRAMを読み取るべきかFlashを読み取るべきかを知らないため、//以下は機能しません

static const char st[] プログラム = "こんにちは世界！"; シリアル。
println （（st））；
```

問題はそれです println（） かどうかわからない st フラッシュまたはRAMメモリを指します。考えられる解決策の1つは、次の例のように、特殊機能を使用してフラッシュメモリから文字を個別に読み取り、出力することです。

```
static const char st[] プログラム = 「Helloworld！\ N」；にとって （（char
c = 0; c < strlen_P （（st））；c++）
    シリアル。印刷 （（（char）。pgm_read_byte_near （（st + c））；
```

Arduino IDEのドキュメントには、フラッシュメモリを使用するためのさらに多くの例が含まれています。キーワードの下で プログラム さらなる情報は、とりわけ、の下で見つけることができます

https://www.arduino.cc/reference/en/language/variables/utilities/progmem/

と

http://playground.arduino.cc/Main/PROGMEM。

この手法は、文字列だけでなく、トーンや値の表など、あらゆる種類の静的データに適用できます。

代替ライブラリの使用

ライブラリは実用的なものであり、問 題ではありません。そして、機能豊富なライブラリは、特定の問題に必要なすべてのものを見つける可能性が高くなります。

ただし、多くの場合、特にリソースを大量に消費し、大量のメモリを占有するのは、まさに特に大規模なライブラリです。セクション6.13.3amのOLEDディスプレイを使用する場合ftドゥイーノ たとえば、Adafruitのグラフィックライブラリは、複雑なグラフィックのすべての機能を備えていますが、同時に大量のメモリを消費します。

ほんの数行のコードで、次の例はメッセージをもたらします "こんにちは世界" OLED画面で。

```
1  # 含む <Adafruit_GFX.h>
2  # 含む <Adafruit_SSD1306.h>
3  Adafruit_SSD1306 画面 (-1);
4位
5  空所 設定 () {
6日   画面。始める ((SSD1306_SWITCHCAPVCC、 画面 0x3C);
7日   。clearDisplay ();画面。setTextColor ((白い);
8日   画面。println (("こんにちは世界！");画面。画面
9      ();
10
11日 }
12日
13日 空所 ループ () {}
```

この単純な例でさえ、1504バイトまたは動的メモリの58%を占めています。これは主に、ライブラリが複雑な描画操作のために画面コンテンツのコピーをダイナミックメモリに保持しているという事実によるものです。128 x 64ピクセルでは、それはすでに128*64/8日 = 1024 バイト。

ただし、グラフィックスキルはまったく必要ないが、16 x 8文字の表示でうまくいく場合は、経済的な代替手段があります。

それらの1つは U8g2-ライブラリ、特にそこに提供されているもの U8x8-としょうかん。ArduinoIDEのライブラリマネージャーに直接インストールできます。The "こんにちは世界" -この場合の例は次のようになります。

```
1  # 含む <U8x8lib.h>
2  U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8 (U8X8_PIN_NONE);
3
4位 空所 設定 ((空所){
5
6日   u8x8。始める ();u8x8。
7日   setPowerSave (0);
8日   u8x8。setFont ((u8x8_font_chroma48medium8_r);u8x8
9      。drawString (0.0、"こんにちは世界！");
10  }
11日
12日 空所 ループ () {}
```

ここでは、動的メモリの22%に相当する578バイトのみが使用されており、他のライブラリや独自のコードのためにさらに多くのバイトが残っています。

これは多くのライブラリと似ており、最小限の労力とリソースの使用で、どのライブラリがこれらの要件を満たすことができるかを詳しく調べる価値があります。多くの場合、わずかな制限で大きな利益を得ることができます。

4.10詳細情報

両方のCに多くのチュートリアルがあります++-プログラミング[2] Arduinoプログラミングと同様に[3]。

これらおよび同様のチュートリアルでは、他の多くの言語構造とライブラリ関数について説明しています。そのようなチュートリアルはにありませんftドゥイーノ 注文仕立て。ただし、この章では、他のチュートリアルでの作業を継続できるようにするために必要な基本事項を提供しました。PCプログラミングやArduinoプログラミングの世界のすべてをに適用できるわけではありませんftドゥイーノ 移行。次の章の実験とモデル、および提供されているサンプルプログラムとともに、プロフェッショナルCのより深い紹介++-プログラミングが可能です。

[2] C。++-チュートリアル http://www.online-tutorials.net/c-c++-c/c++-tutorial-teil-1/tutorials-t-1-58.html
[3] Arduinoチュートリアル http://www.arduino-tutorial.de

第5章

ftドゥイーノ 学校で

the ftドゥイーノ Arduinoファミリーのすべてのメンバーと同様に、主にC言語でのプログラミングに使用されました++ 第4章で説明されているように設計されています。しますかftドゥイーノ 対応するスケッチが事前に提供されているため、はるかに少ない需要で使用できますが、プログラミングの知識がなくても使用できます。theftドゥイーノ そんなに多くのことができます
さまざまなグレードレベルで柔軟に使用できます。

図5.1： ftドゥイーノ-難易度の異なるプログラミング環境

すべての場合において ftドゥイーノ 確立されたプロジェクト（Scratch、Blockly、...）。その結果、ftドゥイーノ 孤立したソリューションではありませんが、他のプロジェクト、たとえば、Arduinoベースのプロジェクトと同等の立場で使用できます。Arduinoの使用からの知識とツールはに適用することができますftドゥイーノ 送信され、その逆も同様です。多くの場合、ftドゥイーノ せん断技術モデルの助けを借りてこのような簡単なスタートを切ることができ、その後、古典的なArduinoに基づいた電気的および機械的により洗練された構造が続きます。

5.1スクラッチを使用したグラフィックプログラミング

ウィキペディアはScratchプログラミング言語について次のように書いています。

目標の設定

その目的は、プログラミングの基本的な概念を初心者、特に子供や若者に理解させることです。想像する、プログラムする、共有する（考え、開発する、共有する）というモットーの下で、相互交換と組み合わせた独自のゲームやマルチメディアアプリケーションの創造的かつ探索的な作成が動機として使用されます。結果は、スクラッチプレーヤーを使用して、広告なしで国際的なオンラインコミュニティで再生、議論、さらに発展させることができます。初心者向けのアイデアを作成し、プログラミングの原則を近づける例もいくつかあります。

https://de.wikipedia.org/wiki/Scratch_(Programmierssprache)

オリジナルでは、Scratchは純粋なシミュレーション環境として設計されていました。プログラミングは、PC上のマウスを使用してグラフィカルに実行されます。PCは、プログラムの実行と結果の表示も担当します。Scratchは、最初はそのような実際のハードウェアを想定していませんftドゥイーノ プログラム開発に関与する。

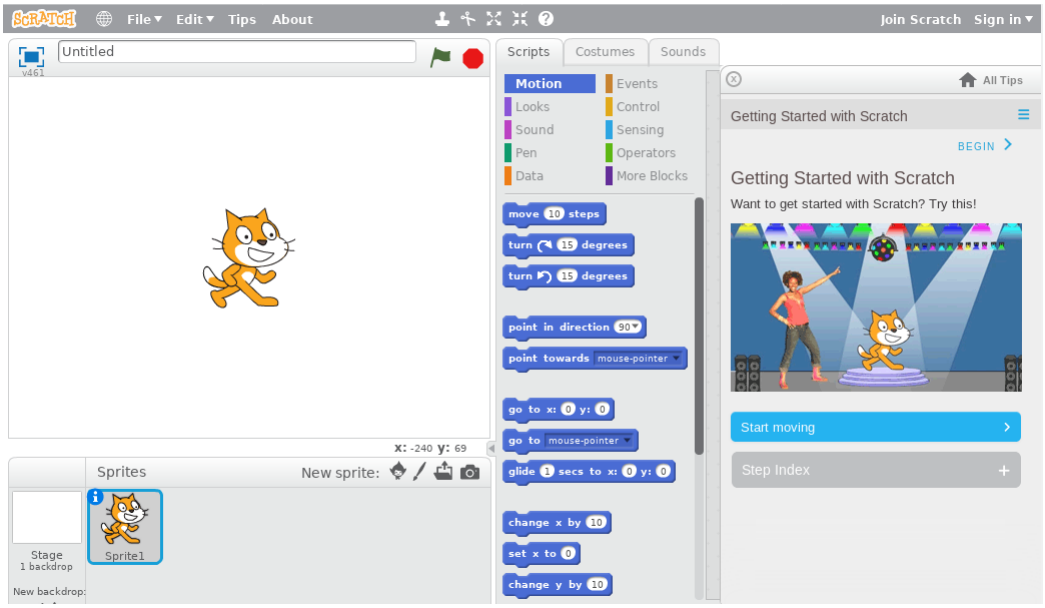


図5.2：Scratch2.0プログラミング環境

スクラッチは英語を話す環境から来ており、元の https://scratch.mit.edu/ 見つけられる。しかし、そのようなドイツ語のポータルもありますDACH Scratch Wiki 下 https://scratch-dach.info と INFスクール 下 https://www.inf-schule.de/programmierung/scratchこれは特にドイツ語圏の教師と生徒を対象としており、学校へのスクラッチエントリーをサポートしています。

5.1.1スクラッチバージョン

Scratchは、3つの独立したバージョンで登場しました。

スクラッチ1.x 2007年にリリースされ、スタンドアロンPCプログラムとして提供されました。このシリーズの最後のバージョン1.4ベースのScratch-for-Arduino（S4A）は、 ftドゥイーノ セクション5.1.2で説明されているように使用できます。Scratch 1は、2009年以降開発されていません。

スクラッチ2.0 2013年に登場し、まったく新しい開発でした。Scratch 2.0はブラウザベースであり、FlashFrameworkと呼ばれます。Flashは、主にセキュリティ上の理由から、最近の多くのブラウザではサポートされなくなりました。したがって、Scratch2.0の使用はますます困難になっています。外部デバイスの使用については、ScratchXバリアント（を参照）https://scratchx.org/） 派生。—ftドゥイーノ-接続はScratch2.0またはScratchX用に開発されていません（これまでのところ）。

スクラッチ3.0 2019年にリリースされ、再び完全に新しい開発です。Scratch3.0はブラウザでもありますベースですが、Flashを省略し、代わりにHTML5を使用します。したがって、最新のすべてのブラウザで使用できます。—ftドゥイーノ接続については、セクション5.1.3で説明しています。

新しいプロジェクトにはScratch3.0を使用する必要があります。Scratch for Arduino（S4A）の使用は、既存のS4Aインストールに基づいて構築する場合にのみ意味があります。S4Aは、通常のArduinoを使用する多くの学校で使用されており、ftドゥイーノ 適用されます。

一方、Scratch 3.0は活発に開発されており、 ftドゥイーノ-接続は積極的に開発されています。

5.1.2 Arduino（S4A）のスクラッチ1.4

プロジェクト Arduinoのスクラッチ、 略してS4AはScratch1.4に基づいており、PC上の仮想ScratchワールドとPCに接続された物理ハードウェアとの間に相互作用を作成するという目標を設定しています。S4A

これを行うために、プロジェクトはArduinosを使用し、それらを仮想スクラッチ環境に統合します。PCのスクラッチプログラムは、接続されたArduinoのセンサー入力（キーストロークなど）に反応したり、Arduinoのアクチュエーター（ランプなど）のアクションをトリガーしたりできます。

S4Aは、人気のあるArduinoのほとんどと互換性があります。これらには、対応するスケッチを事前に提供し、USB経由でPCに接続する必要があります。対応するスケッチは以下のとおりですhttp://s4a.cat/ 利用可能。

Scratchとは対照的に、S4Aはブラウザでは実行されませんが、PCに別のプログラムをインストールする必要があります。対応するダウンロードはで見つけることができますhttp://s4a.cat/。

S4Aはスペイン語圏から来ており、元の http://s4a.cat/ 利用可能。ドイツ語の情報は、たとえば、https://scratch-dach.info/wiki/S4A。

スクラッチ ftドゥイーノ

S4Aの ftドゥイーノ 入力するだけ ftドゥイーノ それ自体とそこにインストールされるスケッチ。PC側では、S4AとS4Aでの使用に違いはありません。ftドゥイーノ。Arduinosと ftドゥイーノ 共有。

Arduinoと ftドゥイーノ 異なる接続がある場合、ユーザーはどの指定の下での接続を知っている必要があります ftドゥイーノ S4Aで対処できます。次の図は、対応する割り当てを示しています。

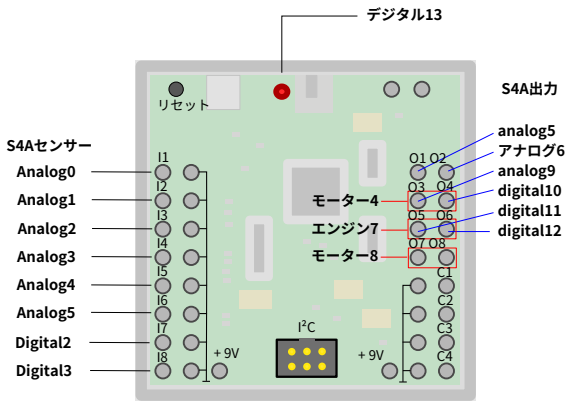


図5.3：の割り当て ftドゥイーノ-S4Aのピン

でS4Aを使用するための詳細情報 ftドゥイーノ セクション8.6にあります。

5.1.3スクラッチ3.0

Scratch 3.0はオンラインアプリケーションであり、一般的なブラウザから直接アクセスできます。 https://scratch.mit.edu 開かれます。

完全に新しい開発と完全に異なる基盤技術にもかかわらず、Scratch3.0はScratch2.0に非常に基づいており、切り替えはまったく問題がない可能性があります。

せん断技術または同様のシステムを使用していない学校での引っかき傷の広がりは、通常、簡単で迅速な開始を保証します。学生は最初、追加のハードウェアなしでスクラッチ環境で直接経験を積み、基本的な取り扱いに慣れることができます。の使用ftドゥイーノ その後、効果的に実行でき、実際のハードウェアの使用をスクラッチに切り替える場合は、ソフトウェア環境を変更する必要はありません。

の使用 ftドゥイーノ Scratch3.0の下で

の使用について ftドゥイーノ さらに2つのことが必要です。一方では、ftドゥイーノ 適切なスケッチをインストールする必要があります。次に、使用するScratch 3.0バリアントには、適切ないわゆる拡張機能が含まれている必要があります。

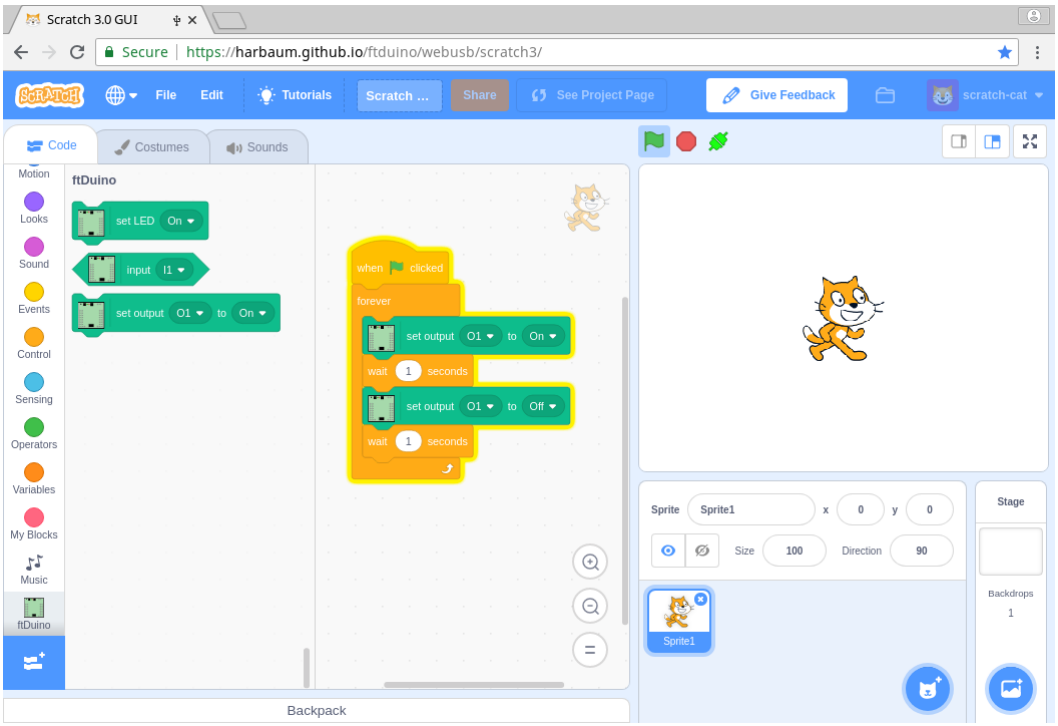


図5.4：Scratch3のメイン画面 ftドゥイーノ-拡大

Scratch 3.0を使用するには、次の手順が必要です。

?? のインストール ファイル例 。WebUSB 。IoServer -スケッチ ftドゥイーノ。

?? の接続 ftドゥイーノ USB経由でPCまたはスマートフォンに

?? ScratchWebサイトを開きます https://harbaum.github.io/ftduino/webusb/scratch3/ Chromeブラウザで

WebUSBの詳細と適切なスケッチのインストール方法については、セクション6.18を参照してください。開始後、最初は拡張機能は統合されていません。内線番号には、選択画面からアクセスできます。

図5.5：Scratch3.0で拡張機能を選択するためのボタン

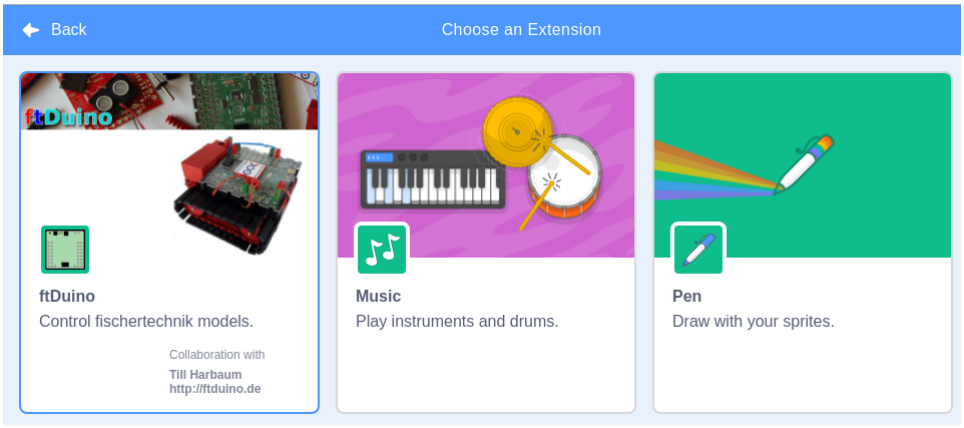


図5.6： ftドゥイーノ-選択画面の拡張

拡張機能をクリックすると、Scratchにインストールされます。接続状態ftドゥイーノ 3分の1までです

緑の開始フラグと赤の停止記号の横にある記号。

シンボルは3つの状態を区別します。赤い十字は、USB接続に一般的な問題があることを示しています。オレンジ色の接続されていないアイコンは、ftドゥイーノ 選択できます。この記号をクリックすると、接続されているデバイスのリストが開きます。この場合は、ftドゥイーノ。しますかftドゥイーノ を選択すると、記号が緑色の接続記号に変わり、 ftドゥイーノ に使える。

図5.7： ftドゥイーノスタートストップシンボルの横にあるステータスシンボル

通常、選択は1回だけ必要です。theftドゥイーノ 使用中にプラグを抜き差しすることができ、その後自動的に統合する必要があります。

Scratch3.0の使用

Scratch 3.0はブラウザでの使用を目的としているため、最初はオンラインで使用することを目的としています。インターネットに接続せずにScratch3.0を使用できるようにするために、必要なすべてのデータをPCにローカルに保存することができます。

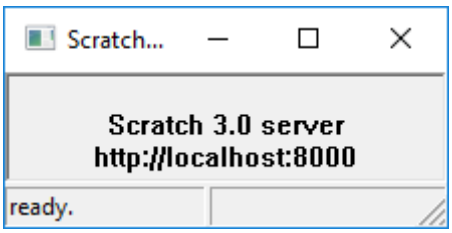


図5.8：ローカルのScratch3.0サーバー

この目的については、を参照してください。 https://github.com/harbaum/ftduino/releases と呼ばれるzipアーカイブs30srv.zip。 これには、通常インターネットから取得されるScratch3.0インストールのすべてのデータが含まれています。さらに、と呼ばれる小さなWindowsアプリケーションがありますs30srv.exe 含む。これは、インストールや特別な権限なしで開始でき、ローカルデータをWebブラウザで利用できるようにします。プログラムはそれ以上の注意を必要としませんが、Scratchを使用する限り開いたままにしておく必要があります。

Oine-Scratchはブラウザの下にあります http：//ローカルホスト：8000 到達すること。

すべてのデータはPCにローカルに保存されるため、もちろん自動更新はありません。更新が発生した場合、ユーザーはZIPアーカイブを再度ダウンロードする必要があります。

5.2 Blockly / Brickly を使用したグラフィックプログラミング

Googleは、スクラッチのアイデアの可能性と、それが現在受けている技術的な制限の両方を認識しました。

Blocklyは、Googleがスクラッチ哲学を現代の技術ベースに置く試みです。BlocklyはHTML5に基づいているため、すべての一般的なWebブラウザーおよびすべての一般的なプラットフォーム（Windows、Apple MacOS、Linux、Android、およびIOSを含む）で、当面はぬるいです。

Scratchは完全な環境を形成し、グラフィックコードの表示に加えて、作成されたプログラムを実行してその結果を表示するオプションも含まれています。一方、Blockly自体は、純粋なコードエディタです。したがって、Blocklyで生成されたコードは、Blockly自体では実行できず、Blocklyは何ももたらしません。

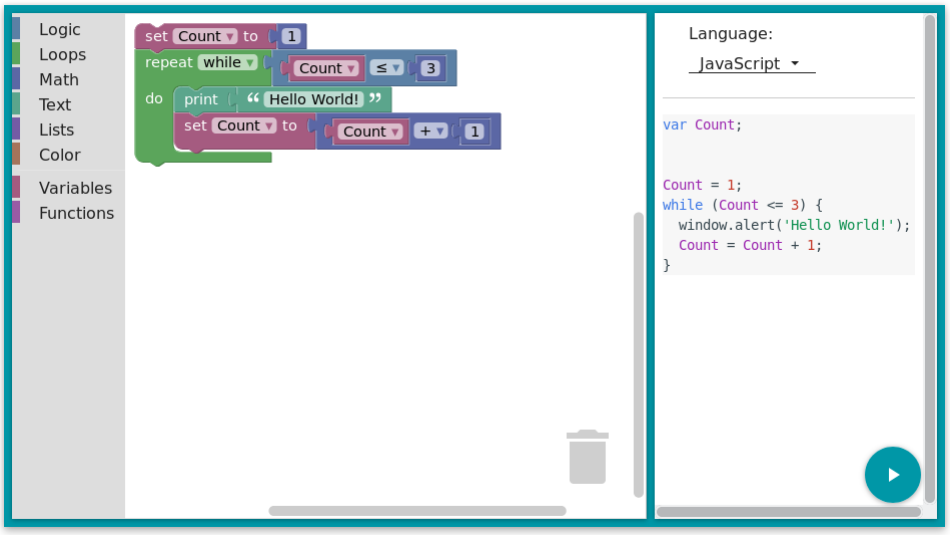


図5.9：Blocklyユーザーインターフェイス

プログラム出力を表すためのメカニズム。エンドユーザーが使用できる環境を作成するには、最初にソフトウェア開発者が両方を追加する必要があります。

Brickly（セクション8.3を参照）とBrickly-Lite（セクション6.18.4を参照）は、schertechnik-TXTコントローラーと ftドゥイーノ 開発されました。両方の環境が意図的に類似していて、ユーザーに非常に類似している場合でも、技術的な機能は大きく異なります。

5.2.1ブリックリー

Bricklyの場合、BlocklyはTXTコントローラーのコードを開発するために使用されます。Bricklyは、ブラウザのいわゆるPythonコードで生成され、TXTに転送されてそこで実行されます。一度作成されたレンガのようなプログラムは、TXTに永続的に存在し、後でいつでもTXTで手動で開始することもできます。Webブラウザは、新しいプログラムを作成するためにのみ必要です。TXTでプログラムを実行している間、TXTの画面で出力を行うことができます。ユーザーがWebブラウザーでTXTに接続している場合、これらの出力もTXTからブラウザーに返送され、そこで出力されます。

ブラウザに表示されるすべての情報は、TXTからブラウザに送信されます。インターネットの他の部分とデータが交換されることはありません。

Bricklyは、主にTXTの入力と出力を操作するために開発されました。したがって、BricklyのBlocklyベースのグラフィックエディタは、TXTの接続を操作するのに適したブロックを含むように拡張されました。TXTにいわゆるBricklyプラグインをインストールする（を参照）https://github.com/harbaum/brickly-plugins）ブリックリーに拡張できます。このようなプラグインにより、ftドゥイーノ TXT自体に加えて、USBを介してTXTおよびBricklyプログラムから結合され、接続された接続も ftドゥイーノ 使用します。

Bricklyは、若い学生でも使いやすく、Scratchよりもシンプルで使いやすいユーザーインターフェイスを備えています。ただし、Bricklyを使用するには、事前にTXTにBricklyソフトウェアをインストールする必要があります。これには、TXTでいわゆるコミュニティファームウェアを使用する必要があります（を参照）。 https://cfw.ftcommunity.de/ftcommunity-TXT/de/）。を使用するにはftドゥイーノ TXTには、 ftドゥイーノ-プラグインが必要です。経験豊富なユーザーは、TXTコントローラーの代わりにはるかに安価なRaspberryPiを使用することもできます。エンドユーザーにとって違いはありません。この設定は、学生にとっても使いやすいものです。

後でBricklyを使用するには、次の準備手順が必要です。

?? TXTへのコミュニティファームウェアのインストール（または の上 に ラズベリーパイ、 ご参照ください https://github.com/harbaum/tx-pi）。

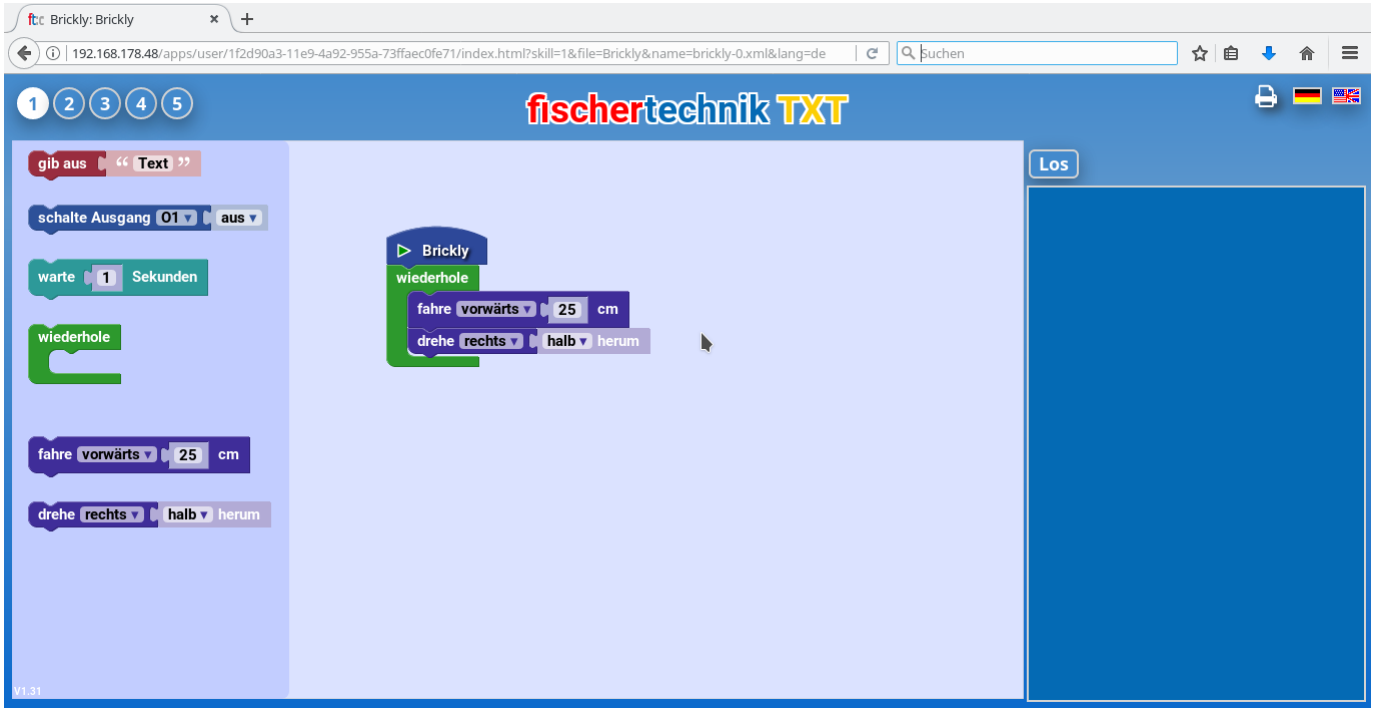


図5.10：Bricklyユーザーインターフェイス

?? インストール から ブリックリー の上 に txt （ご参照ください https://cfw.ftcommunity.de/ftcommunity-TXT / de /プログラミング/ブリック/）。

?? 任意のブラウザからのTXTへのアクセス

する必要があります ftドゥイーノ TXTによって対処されている場合でも、次の手順が必要です。

?? のインストール ftduino_direct-上のスケッチ ftドゥイーノ （ご参照ください https://github.com/PeterDHabermehl/ftduino_direct）。

?? ftduino-Brickly-Pluginのインストール（を参照） https://github.com/harbaum/brickly-plugins）。

5.2.2Brickly-Lite

Brickly-Liteを使用する場合の管理作業は、Bricklyを使用する場合よりもはるかに少なくなります。学生の観点からは、BricklyとBrickly-Liteの違いはごくわずかです。Brickly-Liteは簡略化された表現を提供します。

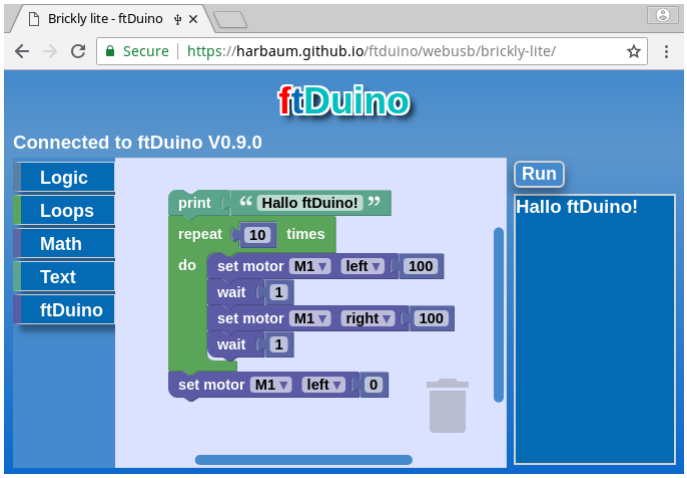


図5.11：BricklyLiteのユーザーインターフェイス

Bricklyとは対照的に、Brickly-Liteはインターネットからすべてのデータを取得します。したがって、Brickly-Liteを使用するにはインターネット接続が必要です。また、Bricklyでは、ブラウザーはコードの入力と結果の出力にのみ使用されますが、Brickly-Liteでは、実際のプログラムの実行を含むほとんどすべてのタスクを引き継ぎます。theftドゥイーノ したがって、その入力と出力をBrickly-Liteを使用してブラウザーで使用できるようにしますが、実際のBricklyプログラムは実行しません。このため、ブラウザは常に後でプログラムを実行する必要があります。Brickly-Liteで作成されたプログラムは、ブラウザなしでは実行できません。

学生はこれらの違いに気づきません。ただし、教師にとっては、準備の労力はブリックリーに比べてはるかに少なくなります。ftドゥイーノ およびブラウザには、これ以上のコンポーネントは必要ありません。最大の制限は、Chromeブラウザの定義です（を参照）https://www.google.com/intl/de_ALL/chrome/）。この場合のように、USB経由でローカルに接続されたデバイスを備えたこのブラウザタイプのみが必要であるため、これが必要です。ftドゥイーノ 通信できます。

Brickly-Liteを使用するには、次の手順が必要です。

?? のインストール ファイル例 。WebUSB 。IoServer -スケッチ ftドゥイーノ。

?? の接続 ftドゥイーノ USB経由でPCまたはスマートフォンに

?? BricklyLiteのWebサイトを開きます https://harbaum.github.io/ftduino/webusb/brickly-lite/ Chromeブラウザで

5.3Minecraftでの遊び心のあるプログラミング

Minecraftは非常に人気のある建築ゲームです。とりわけ、それはいわゆるレッドストーンの形で一種の電気回路シミュレーションを含みます。この導電性の仮想素材を使用すると、対応するブロックをゲーム内で接続でき、ボタンやスイッチなどのゲーム内部センサーをランプや圧力シリンダーなどのアクチュエーターに接続できます。レッドストーントーチ（インバーター、ネゲーター）、コンパレーター、遅延エレメントなどの追加のロジックエレメントにより、複雑な回路が可能になります。

とともに ftドゥイーノ Minecraftは物理的な世界に接続でき、Minecraftの仮想センサーは物理的な世界に接続できます
schertechnikアクチュエータをトリガーします。その逆も同様です。

図5.12：Minecraft ftドゥイーノ-インターフェース

教師側の準備時間は、適切なスケッチを再生することに限定されています。 ftドゥイーノ 対応するのインストールと同様に ftドゥイーノ-セクション8.7で説明されているように、既存のMinecraftインストールに変更します。Minecraftは商用ソフトウェアであり、ライセンスを別途購入する必要があります。のような拡張によるゲームの変更ftドゥイーノ -Modはメーカーによって明示的に意図されたものであり、ライセンス条件に違反していません。

学生の側では、習熟は理論的にはかなり時間がかかります。実際には、学生の大部分はMinecraftの概念に精通しており、追加のロジックを理解しています。ftドゥイーノすぐにブロックします。原則として、Minecraftの経験を積んだ学生は、ライトなどの単純な回路を実装して、わずか数分で物理モデルに接続することができます。

Minecraftの使用は、すでにMinecraft-nerである学生をやる気にさせるのに特に適しています。教師として、あなたは非常に高いレベルの専門知識と驚くべき能力を持っている学生を考慮しなければなりません。Minecraftは、非正統的なソリューションを作成し、たとえば、仮想生物または車両（大型トラック）を信号処理に統合することを勧めます。

Minecraftを使用するには、セクション8.7で詳しく説明されている次の手順が必要です。

?? ライセンスされたMinecraftバージョン1.12.2のインストール

?? ForgeModシステムのインストール

?? のインストール ftドゥイーノ-モッド

?? のインストール ファイル例 。WebUSB 。IoServer -スケッチ ftドゥイーノ。

?? の接続 ftドゥイーノ USB経由でPCに

5.4 ArduinoIDEを使用したテキストベースのプログラミング

ScratchとBricklyは、それ自体が複雑な構造です。たとえば、PC上のプログラムとして、またはモバイルデバイスのブラウザ内のアクティブなWebサイトとして実行されます。theftドゥイーノ どちらの場合も、比較的パッシブなインターフェイスモジュールとしてのみ機能します。USB接続を介してPCからコマンドを受信し、結果を送り返します。

PCを使用する場合の慣例として、プログラム内のほとんどのテクノロジはユーザーには隠されたままです。このようにしてプログラミングの基本を理解することはできますが、初心者が基本的なコンポーネントと手順のすべてを理解することはほとんど不可能です。

5.4.1Arduinoのアイデア

Arduinosの元の概念は、基本的な理解を提供することを目的としています。Arduinoとそれに由来するものftドゥイーノ技術的に最も単純なプログラマブルデバイスに属しています。それらが基づいている技術は、コンピューターのマウス、co eeマシン、飲み物のマシンなど、多くの日常のデバイスでほとんど目に見えず、気付かれずに使用されているテクノロジーに対応しています。

Arduinoで使用されるコンポーネントの数と複雑さは可能な限り低く抑えられているため、関心のある一般の人々でもデバイス全体を大部分見ることができます。それにもかかわらず、このタイプの使用に必要なスキルは、ScratchまたはBricklyを使用するためのスキルよりも大幅に高くなっています。Arduinoを直接使用するには、生徒と教師からのより多くのノウハウが必要であり、7年生から8年生までが賢明です。

学校でのArduino

Arduinoはすでに学校で広く使用されています。特に、低価格と個人世帯の普及は、参入障壁の低下につながります。

ただし、学習曲線は比較的急であり、必要なノウハウはタスクの複雑さとともに急速に増加します。Arduinoはベアボードとして販売されており、その電子機器と接続は外部の影響からほとんど保護されていません。何よりも、誤って破壊された部品を低コストで交換できるため、低価格はこれらの欠点を考慮に入れています。最も単純なArduinoは3ユーロ未満で入手でき、最初はそれ以上の費用はかかりません。

Arduino IDEプログラミング環境は、次のURLから無料でダウンロードできます。 https://www.arduino.cc/en/Main/Software すべての一般的なPCオペレーティングシステムで利用できます。多くのドキュメントと例がインターネット上で無料で入手できます。Arduinoのソフトウェアとハ ードウェアは一般的なオープンソースライセンスの下で利用可能であり、ライセンスなしで共有できます。したがって、学校でためらうことなく使用でき、ほとんどのドキュメントの無料交換は、学校環境に関連する制限の対象にはなりません。

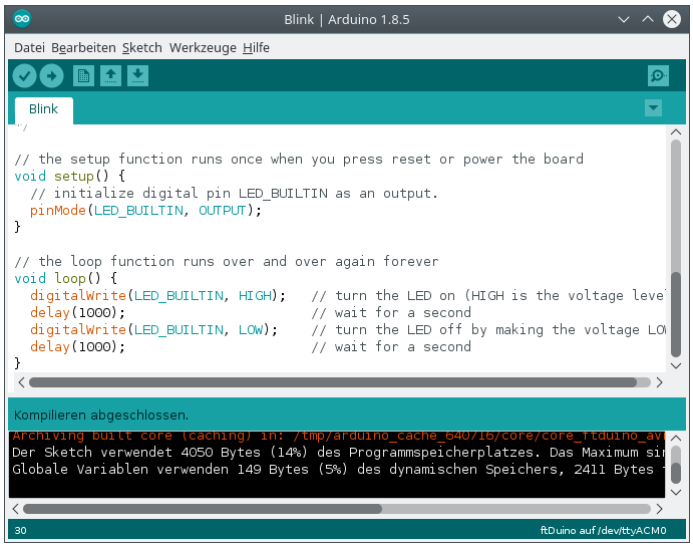


図5.13：Arduino IDE

5.4.2Arduinoと ftドゥイーノ

the ftドゥイーノ せん断技術に向けたArduinoの概念の拡張です。theftドゥイーノ Arduinoのようにプログラムされていますが、電気的および機械的なせん断技術コンポーネントを使用して、複雑なモデルの構築を可能にします。

The ftドゥイーノ Arduinoと比較して、エントリのハードルを下げるいくつかの簡略化。theftドゥイーノ ユーザーの観点から、Arduinoの概念の継続として必ずしも理解される必要はありません。むしろ、Arduinoの世界へのより簡単な参入を可能にします。

the ftドゥイーノ ArduinoLeonardoの近くに寄りかかっています。次の機能によってレオナルドを拡張します。

の堅牢性 ftドゥイーノ 堅牢です。Arduinoの接続とは異なり、その接続は短絡防止であり、電子機器は閉じたハウジングで保護されています。

の供給電圧 ftドゥイーノ せん断技術では通常の9ボルトで動作し、これらの電圧を供給することもできます Arduinoは最大5ボルト用に設計されており、多くのschertechnikエレメントに直接接続されていない可能性がありますが、すべての入力と出力を使用してください。

の入り口 ftドゥイーノ 8つの専用アナログ入力があります I1 それまで I8 および4つのカウンター入力 C1 それまで C4、 すでに電圧、抵抗、またはイベント測定用に準備されています。一方、Arduinoには普遍的に使用可能な入力と出力がありますが、追加の配線によって特定の用途に適合させることができます。

の出力 ftドゥイーノ 8つの専用アナログ出力があります O1 それまで O8、 制御するのに十分強力です ランプとモーターがあります。ユニバーサル入力および出力は、強力な消費者を制御できるようにするために追加の配線を必要とします。

5.4.3 ftドゥイーノ エントリーレベルのArduinoとして

the ftドゥイーノ 学校でのArduinoの使用に簡単かつ問題なく入ることができます。日常の学校生活に十分な堅牢性を備えており、Schertechnikシステムとともに、電気的、電子的、または機械的な要件をさらに必要としません。すべての機械的および電気的接続が接続されており、工具は必要ありません。このマニュアルを含め、配信に含まれている多くの例により、簡単で安全な開始が可能になります。

the ftドゥイーノ 古典的なArduinoの使用を準備することができます。からプロトタイプを構築するftドゥイーノ そしてArduinoは安全なスタートを提供します。その後、古典的なArduino、追加の電子機器、および特別に構築されたメカニズムを使用した後の実装が、ftドゥイーノ 得られた知識。

第6章

実験

この章の実験は、 ftドゥイーノ。最小限の外部コンポーネントを使用して、効果または概念を説明します。実験自体は完全なモデルを表すものではありませんが、多くの場合、それらの基礎として役立ちます。

複雑なモデルの例は第7章にあります。

6.1ランプタイマー

難易度： ★☆☆☆☆

この非常に単純なモデルは、ボタンとランプのみで構成され、典型的な階段の吹き抜けの照明の機能をシミュレートします。エネルギーを節約するために、トグルスイッチは単にライトを切り替えるために使用されるのではありません。代わりに、ボタンが使用され、ボタンを押すたびに、たとえば10秒間ライトがオンになります。この間にボタンをもう一度押すと、残り時間が10秒に延長されます。10秒が経過すると、ライトが消え、ゲームが最初からやり直します。

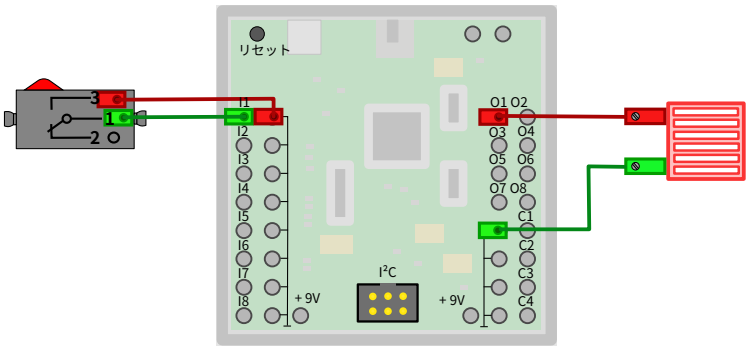


図6.1：ランプタイマー

6.1.1スケッチ ランプタイマー

次のスケッチは、 ftドゥイーノ-ArduinoIDEのメニューでのサポート ファイル 例 。FtduinoSimple 。ランプタイマー 。

```
1  /*
2     LampTimer-ランプタイマー
3
4位    (c) 2017 by Till Harbaum < till@harbaum.org >
5
```

```
 6日    入力I1のボタンが押されるとすぐに、出力O1のランプが10秒間点灯します。
 7日
 8日 * /
 9
10   # 含む       <FtduinoSimple.h>
11日
12日 uint32_t開始時間 = 0;
13日
14日 //セットアップ機能は、リセットが押されたとき、または//ボードが起動したときに1回実行されます。
15日
16   空所 設定 () {}
17日
18日 //ループ関数は何度も何度も何度も実行されます 空所 ループ () {
19日
20日    // I1のボタンが押されているかどうかをテストします もしも ( (
21      ftduino。input_get ( (Ftduino ： ： ：I1) )          {{
22日       // 知らせ      始まる時間
23         始まる時間    = ミリス () ;
24
25日       //ランプをオンにします（出力HI）
26日       ftduino。output_set ( (Ftduino ： ： ：O1、 Ftduino ： ： ：こんにちは) ;
27      }
28
29      //有効な開始時刻で、それ以降10秒以上//（10,000ミリ秒）経過しましたか？
30日
31      もしも ( ( (始まる時間 ! = 0) &&
32            ( (ミリス () > 始まる時間 + 10000) ) { // 忘れ
33          る              始まる時間
34         始まる時間    = 0;
35         //ランプをオフにします（出力をオフにします）
36         ftduino。output_set ( (Ftduino ： ： ：O1、 Ftduino ： ： ：オフ) ;
37      }
38   }
```

スケッチの説明

の機能 ftドゥイーノ 非常にシンプルで、シンプルで使用できます FtduinoSimple-ライブラリをカバーします（セクション9.1を参照）。

Arduinoスケッチには空のスケッチが含まれています 設定（）-初期化が不要なため、機能します。すべての機能はループ（）-関数。

上のボタン I1 についてです input_get（） 永続的に照会されます。押すと、デバイスが起動してからの現在の時刻がミリ秒単位で表示されます。ミリス（） クエリされ、変数内 始まる時間 保存され、ランプがオンになります。ランプがすでにオンになっている場合、この追加のスイッチオンは何もしませんが、時間値はすでに設定されています始まる時間 現在のものに置き換えられます。

これに関係なく、それは常にテストされています 始まる時間 有効な値と、現在のシステム時刻がそこに保存されている値からすでに10秒（10,000ミリ秒）を超えているかどうかが含まれます。この場合、最後にボタンを押してランプをオフにし、の値が 始まる時間 ゼロに設定すると、無効としてマークされます。

タスク1：20秒

各キーを押した後、ランプが20秒間点灯していることを確認してください。

解決策1：

32行目では、ランプが20000ミリ秒、つまり20秒間オンのままになるように、値10000を値20000に置き換える必要があります。

```
31      もしも ( ( (始まる時間 ! = 0) &&
32            ( (ミリス () > 始まる時間 + 20000) ) {
```

演習2：延長なし

ランプがすでに点灯しているときにボタンをもう一度押すと、残り時間が10秒に戻らないことを確認してください。

解決策2：解決策2：

23行目の割り当ての前に、追加のクエリを挿入する必要があります。これは、以前に何も設定されていない場合にのみ新しい値を設定します。両方の行を合わせると、次のようになります。

```
23      もしも（（始まる時間    == 0）
24        始まる時間    =  ミリス  （）；
```

専門家の仕事：

ランプの代わりに発光ダイオードを接続する場合（発光ダイオードの赤い接続を出力に接続する必要があります O1）、次に、ライトが実際にオフになっているはずのときに、何か奇妙なことに気付くでしょう。出力はしますが、発光ダイオードはまだ非常に弱く輝いています。 オフ は。どうして

説明

2つの接続の間に電圧差がある場合、電流はランプまたは発光ダイオードを流れます。一方の接続をアースまたは0ボルトにしっかりと接続し、もう一方の接続は開いており、ftDuinoのコンポーネントから電圧が供給されていません。機械式スイッチとは異なり、imftドゥイーノ いわゆる出力ドライバとして使用される半導体部品は完全ではありません。非常に小さないわゆるリーク電流が9V電源電圧に流れ込みます。この小さな電流は、ランプを光らせるのに十分ではありません。しかし、はるかに効率的な発光ダイオードを非常に簡単に点灯させるには十分です。

それについて何か変更できますか？はい！出力を完全に接続しないままにする代わりに、ftDuinoの出力ドライバーに出力を永続的にグランド（0ボルト）に切り替える必要があることを伝えることができます。その後、発光ダイオードの両方の接続がしっかりと接地され、漏れ電流の影響がなくなります。これを行うには、定数が36行目にある必要がありますオフ 終えた LO 交換してください。 LO 低、英語低を表し、この場合は0ボルトを意味します。時間が経過すると、LEDが完全に消灯します。

```
36        ftduino。output_set（（Ftduino：：：O1、Ftduino：：：LO）；
```

ただし、ftDuinoをオンにした直後は、LEDが点灯し続けます。おそらく、それをどのように変更できるかを自分で理解するでしょう。ヒント：以前は使用されていなかった設定（）-関数が役立つ可能性があります。

これについてはセクション6.8で詳しく説明しています。

6.2緊急停止

難易度：

緊急停止スイッチは人命を救うことができ、簡単なことのようです。ボタンを押すと、問題のマシンのスイッチがすぐにオフになります。モデルには、出力にファンを備えたXSモーターがありますM1 マシン入力のボタン I1 緊急停止ボタンを形成します。

6.2.1スケッチ 緊急停止

```
1   / *
2       EmergencyStop-緊急停止
3
4位     （c）2017 by Till Harbaum < till@harbaum.org >
5
6日     非常停止ボタンが押されるとすぐにファンをオフにします
```

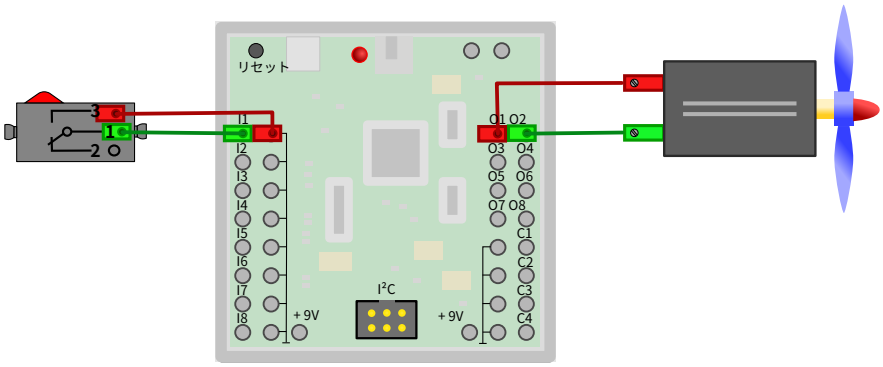


図6.2：緊急停止

```
 7日    が動作します。
 8日 * /
 9
10   # 含む <FtduinoSimple.h>
11日
12日 //システムの起動時にセットアップ機能が1回実行されます 空所 設定 （） {
13日
14日    //システムの起動時にファンをオンにします ftduino。motor_set （（
15日    Ftduino ：：：M1 、 Ftduino ：：：左）；
16
17日    //内部の赤いLEDの出力をアクティブにします pinMode （（
18日    LED_BUILTIN 、 出力）；//そしてLEDをオフにします digitalWrite
19日     （（LED_BUILTIN 、 低い）；
20日
21   }
22日
23   //ループ関数は何度も何度も何度も実行されます 空所 ループ （） {
24
25日    // I1のボタンが押されているかどうかをテストします もしも
26日     （（ftduino。input_get （（Ftduino ：：：I1））{
27          //モーターにブレーキをかけます
28          ftduino。motor_set （（Ftduino ：：：M1、 Ftduino ：：：ブレーキ
29          ）；//内部の赤いLEDをオンにします digitalWrite （（LED_BUILTIN 、
30日        高い）；
31       }
32   }
```

スケッチの説明

スケッチは非常に短くシンプルです。の中に設定（）-スケッチが開始されると、モーターは15行目で開始されます。さらに、の赤い内部発光ダイオードftドゥイーノ 後で使用できるように有効にしましたが、今は省略しました。

の中に ループ（）-緊急停止ボタンが閉じられているかどうかは、26行目で機能が永続的に照会されます。この場合、モーターは28行目ですぐに停止し、30行目で赤色発光ダイオードがオンになります。エンジンは意識的にブレーキ 代わりに停止しました オフ。 このように、モーターは短絡され、積極的にブレーキがかけられますが、そうでない場合、モーターはゆっくりと惰走し、緊急時に危険になります。

タスク1：ケーブルの断線

緊急ボタンは多くのマシンで利用できます。幸いなことに、それらが実際に必要になることはめったにありません。これには、非常停止ボタンに問題がある場合に誰も気付かないという欠点があります。多くの場合、ケーブルはボタン自体よりも脆弱であり、日常の作業でケーブルが損傷する可能性があります。多くの場合、ケーブルを見てもわかりません。たとえば、シースに損傷がないように見えても、過度のストレスのために内部の銅導体が遮断されています。その結果、いわゆるケーブルが断線します。

ケーブルを引き裂く必要はありません。ボタンを接続するケーブルのプラグの1つを接続すれば十分です。ftドゥイーノ接続し、引き出します。非常停止ボタンが機能しなくなり、機械を停止できなくなります。危険な状況。

解決策1：

この問題の解決策は驚くほど簡単です。非常停止ボタンをクローザーとして接続しました。これは、ボタンが押されたときに接点が閉じられることを意味します。Schertechnikボタンをオープナーとして使用することもできます。その後、接点はアイドル状態で閉じられ、ボタンが押されると開きます。

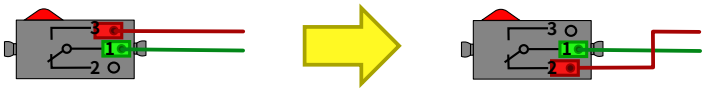


図6.3：ケーブル破損防止緊急停止オープナー

現在のスケッチでは、ボタンがすぐに閉じていると認識されるため、ケーブルが断線した場合、マシンはすぐに緊急状態になります。そのため、スケッチでもロジックを逆にする必要があります。これは、26行目の次の変更で発生します。

```
26日    もしも（！ftduino。input_get（（Ftduino：：：I1））{
```

違いを確認するには注意深く見る必要があります。開き括弧の後ろに感嘆符があります。感嘆符は、Cプログラミング言語では式の論理否定を表します。ボタンが押されたときに条件が実行されるようになりましたいいえ 閉まっています。この変更後、マシンは元の動作とまったく同じように動作するはずです。小さな変更が1つあります。緊急停止ボタンのプラグの1つを抜くと、マシンはすぐに停止します。これは、ケーブルが断線した場合にも発生します。

このような回路は英語ではフェイルセーフと呼ばれます。何かが壊れると、回路は安全な状態に変わります。schertechnik 3Dプリンターは、たとえば、この回路をリミットスイッチに使用します。ここでケーブルが引き裂かれた場合、プリンターはモーターを軸のエンドストップに押し付けません。代わりに、リミットスイッチへの接続が中断されるとすぐにプリンタは完全に動作を拒否します。

専門家の仕事：

ケーブルを単に遮断することはできません。また、ケーブルがひどく絞られて、内部導体が互いに接触することもあります。これはそれほど頻繁には発生しませんが、現実的なリスクでもあります。

この場合、改良された非常用回路は保護されず、非常停止ボタンは再び機能しません。したがって、接続の閉じた状態も開いた状態も良好として認識されないバリアントが必要です。

解決：

この場合、解決策はもう少し複雑です。ここで、少なくとも3つの状態（正常、中断、および短絡）を区別する必要があります。純粋なスイッチング入力は、閉じた状態と開いた状態の2つを区別することしかできません。

解決策は、入力のアナログ機能を使用することです。これは、たとえば、ボタンで直接行うことができます100 Ω- ラインに抵抗を統合します。

通常、ボタンは閉じており、入り口にあります I1 測定する抵抗は 100 Ω。 回線が遮断されると、抵抗は無限に高くなります。そして、ラインが短絡している場合、抵抗は近いです 0 Ω。 機械は、抵抗がに近い場合にのみ実行が許可されます 100 Ω は。使用される抵抗の正確な値は製造公差の影響を受け、閉じたボタンとその接続ケーブルも非常に低い抵抗を持っているため、ある程度の公差が必要です。これは、測定される総抵抗に影響します。

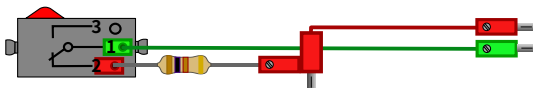


図6.4：ケーブルの断線や短絡に対する安全な緊急停止

なぜ抵抗器をボタンの近くに取り付ける必要があるのですか？彼が近くにいる場合はどうなりますかftドゥイーノ を使用すると、抵抗器とボタンの間のケーブルで短絡が発生しますか？

6.3パルス幅変調

難易度： ★★☆☆☆

さまざまな明るさでランプを輝かせたい場合、または制御可能な速度でモーターを実行したい場合は、ランプまたはモーターのエネルギー消費に影響を与える方法が必要です。これを行う最も簡単な方法は、調整可能な電圧源を使用することです。電圧が高いとランプのエネルギー消費量が増加し、ランプが明るく点灯してモーターが速く回転します。電圧が低いとランプが暗くなり、モーターの回転が遅くなります。のアナログ出力の場合ftドゥイーノ これは、ランプとモーターを完全な暗闇または停止状態から最大の明るさまたは速度まで操作できるようにするために、0〜9ボルト（アナログ）の間で連続的に調整可能な電圧を出力できる必要があることを意味します。

可変電圧の生成は、技術的に比較的複雑です。ただし、同様の結果を得る簡単な方法があります。電圧を下げる代わりに、電圧は非常に短い間だけ定期的にオンになります。たとえば、電圧がオンになっている時間は50%で、オフになっている時間は50%の場合、エネルギーの半分だけが合計時間に転送されます。結果をランプの点滅として認識するか、モーターの途切れとして認識するか、またはランプが単に半分の明るさで点灯し、モーターが半分の速度で回転するかどうかは、電圧のオンとオフを切り替える速度によって異なります。

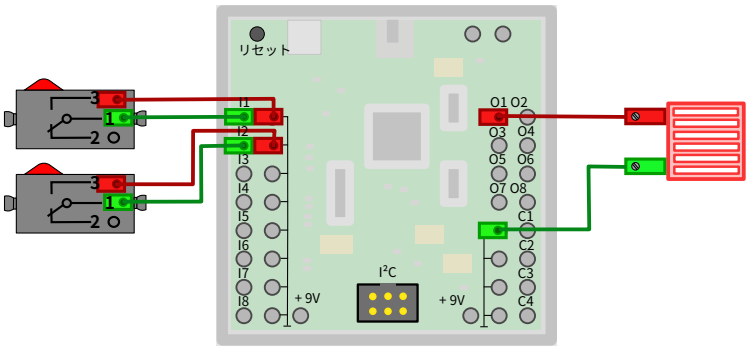


図6.5：パルス幅変調

6.3.1スケッチ Pwm

```
1   / *
2       Pwm-パルス幅変調
3
4位     （c）2017 by Till Harbaum < till@harbaum.org >
5   * /
6日
7日 # 含む <FtduinoSimple.h>
8日
9   uint16_tスイッチング時間 = 8192;              // 8192は1/2秒のオンとオフに対 応します
10
11日 //セットアップ機能は、リセットが押されたとき、または//ボードが起動したときに1回実行されます。
12日
```

```
13日 空所 設定 （） {}
14日
15日 //指定された時間待機します。「時間」値8192は、0.5秒//に対応する必要があります。したがって、500000/8192マイクロ
16   秒の「時間」を待つ必要があります 空所 待つ （（uint16_t時間）{
17日
18日    その間 （（時間 -）
19日        _delay_us （500000/8192）;
20日 }
21
22日 //ループ関数は何度も何度も何度も実行されます 空所 ループ （） {
23
24      静的 uint8_t on_off = false; 静的 uint8_t i1=false        //現在の出力のオン/オフ状態// I1およびI2のボタンの最後の状
25日    、 i2=false;                                               態
26日
27      // I1のボタンが押されていますか？ もしも （（
28      ftduino。input_get （（Ftduino ：：：I1））          {{
29          //以前はボタンが押されていませんでしたが、//現在の切り替え時間は8192未満
30日        ですか？
31          もしも （！i1 &&（切り替え時間 <8192））{
32              //次に切り替え時間を2倍にします 切り替え時間
33              * = 2;
34              //キーがバウンスする場合に備えてミリ秒待機します _delay_ms （1）;
35
36          }
37          // I1のキーが現在押されていることに注意してください i1 =
38                  true;
39      }   それ以外
40          // I1のキーが現在押されていないことに注意してください i1 = false;
41
42
43      // I2のボタンが押されていますか？ もしも （（
44      ftduino。input_get （（Ftduino ：：：I2））{
45          //以前はボタンが押されていなかったので、//現在の切り替え時間は1より大きい
46          ですか？
47          もしも （！i2 &&（切り替え時間 > 1））{
48              //次に切り替え時間を半分にします 切り替え時
49              間 / = 2;
50              //キーがバウンスする場合に備えてミリ秒待機します _delay_ms （1）;
51
52          }
53          // I2のキーが現在押されていることに注意してください i2 =
54                  true;
55      }   それ以外
56          // I2のキーが現在押されていないことに注意してください i2 = false;
57
58
59      // on_off変数の状態に応じて、出力O2をオンまたはオフに切り替えます もしも （（オンオフ））。
60
61          //現在のon_off状態がtrueの場合、出力をオンにします ftduino。output_set （（Ftduino ：：：O1 、 Ftduino
62          ：：：こんにちは）; それ以外
63
64          //現在のon_off状態がfalseの場合、出力をオフにします ftduino。output_set （（Ftduino ：：：O1 、 Ftduino
65          ：：：オフ）;
66
67      //現在の切り替え時間を待つ 待つ （（切り替え
68      時間）;
69
70      // on_off状態を変更します オンオフ = ！オ
71      ンオフ;
72  }
```

スケッチの説明

スケッチは出力を切り替えます O1 の中に ループ（）-60行目から71行目で連続してオンとオフを切り替えます。変数の値に応じてオンオフ 62行目の出力は9ボルトに設定されています（こんにちは） 65行目でスイッチまたはオフになっています（電源から切断されています）。71行目では、ループ（）-変数の状態の関数 オンオフ 各サイクルで出力が交互にオンとオフに切り替わるように変更されました。

オン/オフを変更するたびに、68行目に待機があります。変数の待機時間はどのくらいですか切り替え時間 彼女はで待ち時間を与えます 1/8192 0.5秒。これは関数で行われます待つ （） 19行目 500000/8192

変数のようにマイクロ秒待機 切り替え時間 指定。なぜ0.5秒？サイクルごとに2回待機するため、1回は出力がオンになっているとき、もう1回はオフになっているときです。0.5秒待つと、サイクル全体が1秒間続き、出力は1秒に1回0.5秒間オンになります。出力は次の周波数で変化します1/Sec。またはヘルツ。

ボタンを押すことで I1（28行目）は変数の値にすることができます 切り替え時間 2倍になり（33行目）、ボタンを1回押すだけで I2（44行目）半分（49行目）。スイッチング時間の値は、1（47行目）から8192（31行目）の範囲に制限されています。ここで、この奇妙に曲がった値8192が選択された理由が明らかになります。8192は2の累乗であるため（2_{13}日）は、値を丸め誤差なしで1に分割してから、再度乗算することができます。

ボタンはオンとオフを切り替えるときにのみ照会されるため、点滅する周波数が変わるまで、低周波数でボタンをしばらく押し続ける必要があります。

スケッチが始まると、ランプが1秒に1回、0.5秒間点灯します。ボタンを（長押し）押すI2 待ち時間が半分になり、ランプが1秒間に2回点滅します。ボタンを2回押すと、4回点滅します。6回押すと、1秒間に32回点滅しますが、これはわずかなちらつきとしてのみ認識され、7回押すと64回点滅します。人間の目は約50ヘルツを超える周波数を解決できなくなり、ランプは半分の明るさで光っているように見えます。周波数をさらに上げても、認識できる効果はありません。

演習1：遅いランプ

この構造では、動きが鈍いのは人間の目だけではありません。ランプも遅いです。フィラメントが熱くなり、ランプが点灯するまでに時間がかかります。また、フィラメントが十分に冷えてランプの点灯が止まるまでにも時間がかかります。

発光ダイオードは白熱灯よりもはるかに高速です。それらの中の何も加熱または冷却する必要はありません;代わりに、光は発光ダイオードの半導体材料の光電気効果によって直接生成されます。ランプの代わりに発光ダイオードを接続した場合（出力への赤いマークの付いた接続O1）、 その後、動作は最初は同じように見えますが、64ヘルツの周波数から、発光ダイオードは半分の明るさで均一に輝いているように見えます。ただし、多くの人は依然として64 Hzをわずかなちらつきとして認識しており、100Hzからのみ本当に常に無料のディスプレイについて話します。

発光ダイオードが動いているとき、発光ダイオードのちらつきはこれらの周波数でまだ観察することができます。少し長いケーブルを使用して、発光ダイオードを自由に動かし、少し暗い環境ですばやく前後に動かすと、一連の中断されたライトストリップの印象が生まれます。

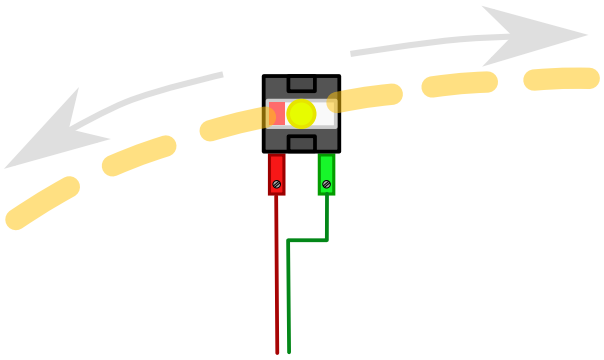


図6.6：耕起発光ダイオードの急速な動きのパターン

PWM周波数が高いほど、可視光ストリップは短くなります。

この実験は、ランプを使用して繰り返すこともできます。ランプの慣性により、連続した光の帯しか見ることができません。ただし、光るランプの繊細なフィラメントは振ると壊れやすいので、あまり乱暴に進めないでください。この点でも、発光ダイオードは頑丈であり、強い振動によっても感動することはできません。

タスク2：エンジンからの音

モーターも低速で、どの速度でもオン/オフ信号を追跡できません。非常に低いPWM周波数でも、モーターは半分の速度で連続的に回転します。聞こえる主な騒音はエンジンの運転騒音です。

ただし、モーターを手で持つなどして機械的にブロックすると、ランニングノイズが抑制され、別の効果が聞こえるようになります。モーターのコイルがスピーカーのように機能し、モーターがブロックされたときにPWM周波数を聞くことができます。 。PWM周波数を変更すると、はっきりと聞こえるピッチの違いが生じます。

PWM周波数が高いほど、ブロックされたモーターで聞こえる音が大きくなります。

演習3：高いPWM周波数の欠点

ランプの場合、周波数が高くなるとフリッカーが減少するため、PWM周波数が高くなることは非常に有利であるように思われます。ただし、モーターに悪影響が見られる場合があります。

モーターが自由に回転している場合、モーターの回転音のピッチは回転速度に関係しますが、前のタスクのPWMノイズは後部座席になります。モーターの回転が速いほど、ランニングノイズの周波数が高くなり、その逆も同様です。

PWM周波数を上げると、モーターから放出されるトーンの周波数がわずかに低下します。明らかに、PWM周波数が高くなると遅くなります。この効果は、モーターがいわゆる誘導負荷を表すという事実によって説明できます。それは本質的にコイル、いわゆるインダクターで構成されています。誘導性負荷の抵抗は、印加される交流電圧の周波数に依存します。そして、PWMによって生成されるオン/オフ信号は他に何もありません。周波数が高いほど、コイルの抵抗が高くなり、コイルを流れる電流が少なくなります。

出力電圧を平滑化し、この影響を軽減することは技術的に可能です。このタイプの平滑化は、使用されるPWM周波数とモーターの消費電流によって異なります。また、出力の一般的なスイッチング動作にも影響します。対応する平滑化の使用ftドゥイーノ したがって、これは出力のユニバーサルユーザビリティを制限するため、問題外です。

したがって、PWM周波数を選択する際の目的は、ランプのプラウやモーターのスタッターを防ぐのに十分高い周波数ですが、モーター巻線の誘導抵抗を最小限に抑えるために可能な限り低い周波数です。100〜200HzのPWM周波数はこれらの条件を満たす。

PWM比に応じたモーター速度

モーター速度は、パルス幅変調中のフェーズインとフェーズアウトの比率によって影響を受ける可能性があります。以前の試みでは、オンフェーズとオフフェーズはそれぞれ同じ長さでした。2相の比率を変えることで、ランプの明るさやモーターの速度を制御することができます。PWM周波数は一定に保つことができます。

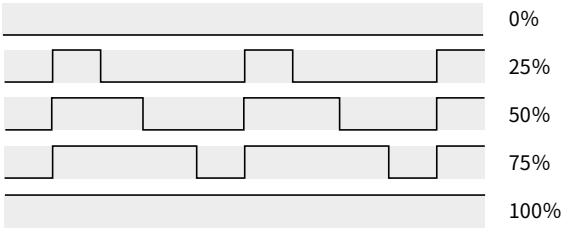


図6.7：0から100％までの選択されたPWM比

スイッチオフフェーズと比較してスイッチオンフェーズが長いほど、ランプが明るく輝き、モーターの回転が速くなります。適切な測定オプションがないため、ランプの明るさとPWM比の正確な関係を判断するのは簡単ではありません。ただし、いわゆるエンコーダモーターには、速度測定用のオプションが組み込まれています。TXTエンコーダーモーターの場合、これらのエンコーダーは1回転あたり63の信号パルスを生成します。

軸。のカウンタ入力でエンコーダ信号を評価することによってftドゥイーノ モーターの速度を決定します。

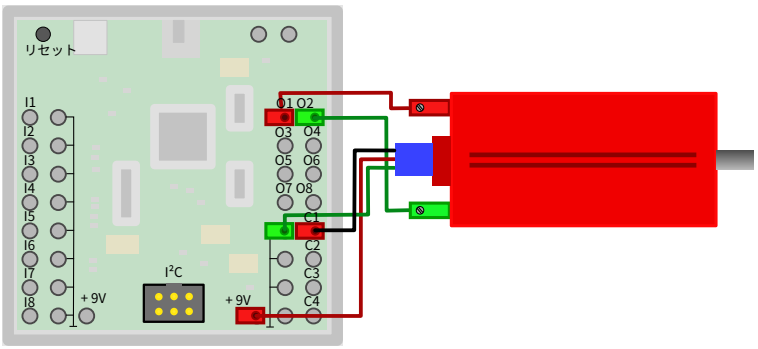


図6.8：PWM依存速度測定用のTXTエンコーダモーターの接続

例 ファイル例 。Ftduino 。PwmSpeed PWMのオン/オフ比を0からゆっくりと調整します 100%の高さで、入り口の1つを一度に1秒間継続的に測定します C1 適用されたインパルス。これらは変換され、1分あたりの回転数で出力されます。これが完全なライブラリの出番ですFtduino これは、PWM信号の実際の生成に使用されます。PWM信号の生成は完全にバックグラウンドで行われるため、スケッチ自体はモーターを始動してから1秒間待機するだけです。

この方法で取得したデータを、ArduinoIDEのメニューにあるいわゆるシリアルプロッタにフィードすると、 ツール 。シリアルプロッタ 最後に、測定結果を明確に視覚化することができます。

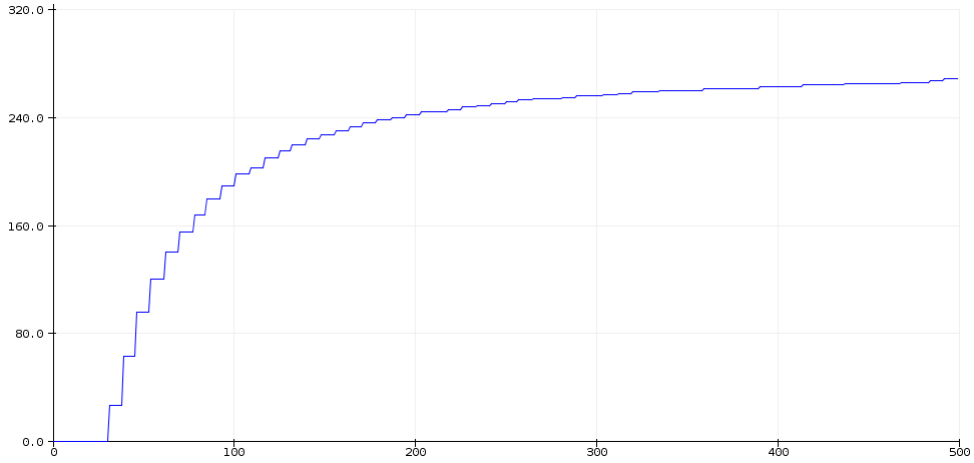


図6.9：PWM比に応じたTXTエンコーダモーターのアイドル速度

PWM比は横軸にプロットされ、左端から永久に右端から永久に始まります。測定された速度は、縦軸に毎分回転数で表示されます。アイドリング時に関係が線形ではないことがわかります。信号が約25%の時間だけオンになっている場合、最大エンジン速度の90%に達します。

たとえば、モーターに一定の負荷をかけることで、この曲線と負荷がどのように変化するかを確認できます。

6.4ステッピングモーター制御

難易度： ★★★★★

一般的な電気モーターは、通常おもちゃで使用されるため、いわゆる非同期モーターです。これらのモーターは通常DC電圧が供給されますが、電圧が印加されるとすぐに作動するという特徴があります。

回転を開始します。速度は外部の影響に間接的にのみ依存し、モーターは可能な限り速く回転します。このタイプのモーターは、多くのアプリケーションに非常に適しています。モデルカーは、それ以上の制御なしでモーターを駆動し、これらのモーターで可能な限り迅速に運転することができます。Schertechnikもほとんどの場合これらのモーターを使用し、ftドゥイーノ それらをそれぞれ1つのモーター出力に直接接続することができます M1 それまで M4 接続する。

しかし、このモーターファミリーが非常に大きな弱点を示すアプリケーションがあります。単純な非同期モーターでは、正確な速度を達成したり、正確な位置に移動したりすることは非常に困難です。schertechnikのエンコーダモーターは、追加のハードウェアでこれを可能にしようとします。ただし、この手順には制限もあります。特に、回転方向を検出できないSchertechnikエンコーダーモーターの場合はそうです。

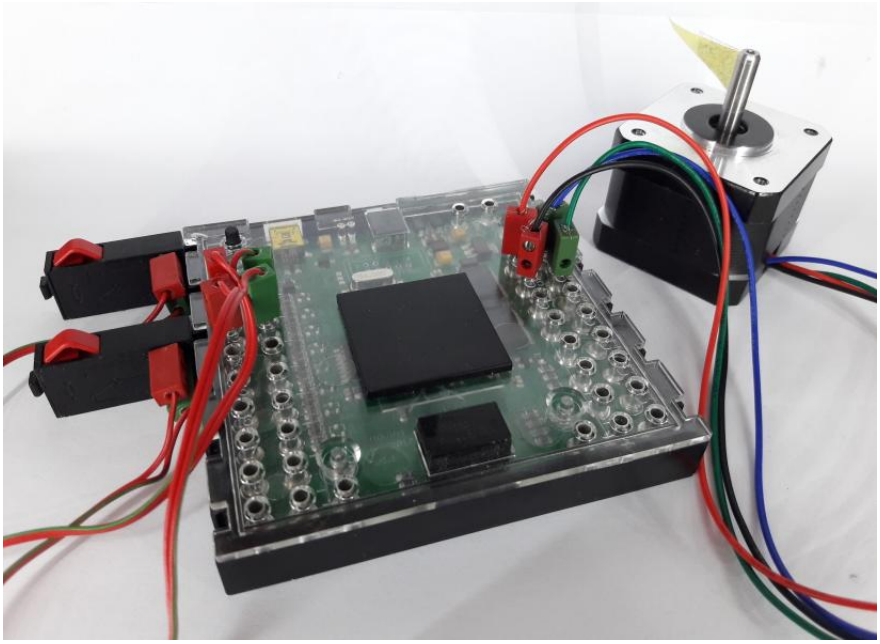


図6.10：17HS13ステッピングモーター ftドゥイーノ

モーターの正確で再現性のある動作が重要なタスクには、いわゆるステッピングモーターなど、同期して動作するモーターがあります。この今日の一般的なアプリケーションは、スキャナーと3Dプリンターですが、過去に使用されたフロッピーディスクドライブと初期のハードドライブもあります。ステッピングモーターの使用は、エンドユーザーがそのようなモーターで音楽を作成するために使用する特徴的な動作ノイズによっても認識できます。[1]。

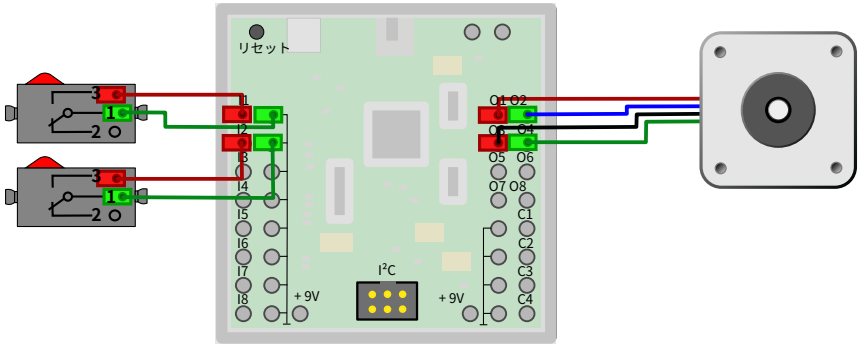


図6.11：ステッピングモーターの ftドゥイーノ

一般的なステッピングモーターは、電磁石に囲まれた永久磁石で作られた回転可能な電機子で構成されています。永久磁石は周囲の磁場に応じて整列します。緊張のない中で

[1] フロッピー音楽というキーワードでインターネット上で簡単に見つけることができます

状態、モーターの軸は比較的簡単に回転させることができます。顕著な抵抗は、電圧がない場合でも電磁石の鉄心に永久磁石が引き付けられるという事実に起因します。

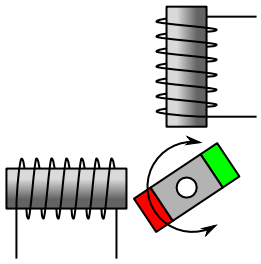


図6.12：ステッピングモーターの簡略化されたスキーム

図に示す簡略化されたステッピングモーターには、永久磁石と2つの電磁石があります。実際のステッピングモーターには通常2つ以上のコイルがあり、電機子も2つ以上の磁極を形成します。この単純化は、機能原理に影響を与えません。

従来のいわゆるバイポーラステッピングモーターには、2つの電磁石のそれぞれに2つずつ、合計4つの接続があります。電磁石は電圧を印加することによって磁化されます。その結果、アンカーはそれに応じて整列します。印加電圧の極性により、電磁石の磁場の方向が決まります。

6.4.1フルステップ制御

両方のコイルが常に通電されている場合、2つの電磁界には4つの異なる方向があり、アーマチュアは4つの異なる位置になります。対応する反復信号シーケンスが電磁石に適用されると、電機子は信号に追従して回転します。変化する磁場に正確に追従し、印加された信号パターンと同期して回転します。このようにして、モーターの速度と位置を正確に予測することができます。すべてのコイルが常に通電されているため、サイクルが正確に4つの状態を通過する場合、1つはフルステップ制御について説明します。

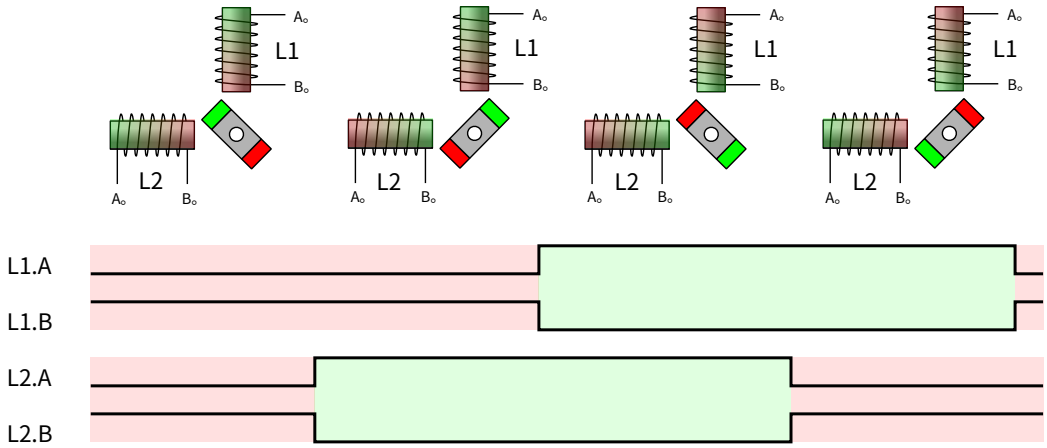


図6.13：ステッピングモーターのフルステップ制御

対応する信号パターンを継続的に生成するスケッチは次のようになります。

```
その間 (1) {
  ftduino。motor_set (（Ftduino：：：M1、    Ftduino：：：左）；
  遅れ (5)；
  ftduino。motor_set (（Ftduino：：：M2、    Ftduino：：：左）；
  遅れ (5)；
  ftduino。motor_set (（Ftduino：：：M1、    Ftduino：：：正しい）；
  遅れ (5)；
  ftduino。motor_set (（Ftduino：：：M2、    Ftduino：：：正しい）；
```

```
    遅れ（5）；
}
```

完全な例はArduinoIDEにあります ファイル例 。FtduinoSimple 。ステッピングモーター 。

示されている簡略化されたステッピングモーターは90変化しますが°° 回転し、したがって4つのステップの後に完全な回転を完了しました、実際のステッピングモーターはより高い解像度を持っています。1.8のステップ角が一般的です°。 そのようなエンジンが完全に回転したのは、200ステップ後になってからです。示されているリストでは、各ステップの後に5ミリ秒待機しているため、1秒あたり正確に200ステップが生成されます。一般的な1.8°-モーターは毎秒1回正確に回転します。

の実験のために ftドゥイーノ の9V出力に接続されているモーターを選択する必要があります ftドゥイーノ 互換性があります。ここで使用されている17HS13は、12Vの動作電圧用に設計されていますが、せん断技術で一般的な9ボルトでも確実に動作します。schertechnikプロッタ30571のモーター[2] 1985年から6ボルト用に設計されました。これらのエンジンをオンにする必要がありますftドゥイーノ 動作する場合は、通常の9ボルトではなく6ボルトを供給する必要があります。

プロセスの図の下半分には、モーターの4つの接続での信号プロセスが含まれています。信号曲線は、結果として得られるメグネットフィールドの方向に応じて色で強調表示されます。磁石の2つの接続が常にまったく逆の方法で制御され、信号が変化すると磁場が変化することがわかります。示されている色は、アーマチュアに面している電磁石の側面の極性に対応しています。

6.4.2ハーフステップ制御

ステッピングモーターをいわゆるハーフステップモードで制御すると、より高い角度分解能を実現できます。この場合、信号サイクルは4つではなく、8つのステップで構成されます。2番目のステップごとに、2つの電磁石の1つがオフになり、アーマチュアが残りの磁石とのみ位置合わせされます。結果として得られる4つの中間状態は、結果として得られる角度から、フルステップ制御の4つの状態の間に正確に配置されます。したがって、モーターは2倍の角度を制御し、それに応じてより正確に配置することができます。

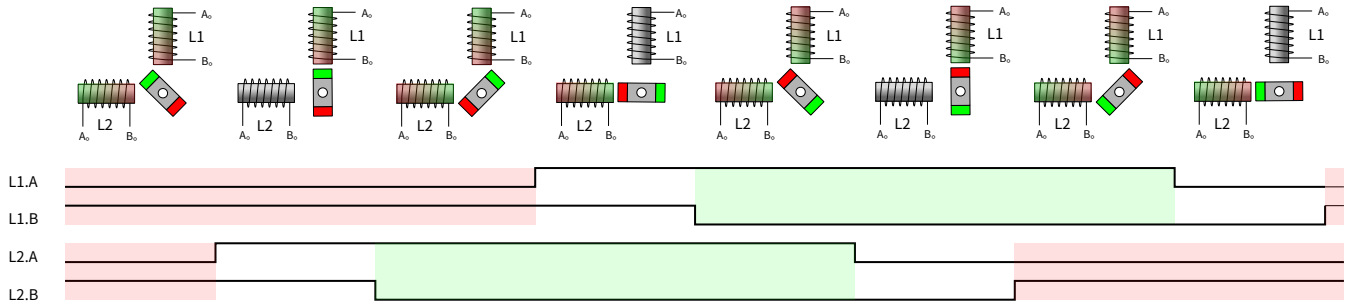


図6.14：ステッピングモーターのハーフステップ制御

この図は、電磁石を制御するための2つの信号が同時に変化しなくなったことを示していますが、両方の信号が同じレベルにあるオフセットがあります。このとき、磁石には電圧がかかっておらず、磁場もありません。したがって、現時点では信号曲線はカラーで強調表示されていません。

ハーフステップ制御の欠点は、1つの電磁石のみがアクティブな場合にモーターの電力が低下することです。

タイマー割り込み作動ステッピングモーター

前のセクションで使用したステッピングモーター制御のスケッチには、1つの大きな利点があります。それは明らかです。ただし、問題は、モーターの制御には永続的な信号変更が必要であるため、モーターが回転するように、スケッチ内のモーター機能を永続的にアクティブにする必要があることです。スケッチは、ほとんどすべての時間をさまざまな場面で費やしています。遅れ （） -次の信号変更を待つ関数呼び出し。最大

[2] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=30571

短所：スケッチがモーターを操作している間は、他のことはほとんどできません。この単純なスケッチでは、おそらく異なる速度で2番目のモーターを同時に実行することはほとんど不可能です。

同じ問題が開発中に発生しましたftドゥイーノ の他のコンポーネントとftドゥイーノ アナログ入力の評価、モーター出力のPWM速度制御、およびカウンターの評価にも、マイクロコントローラーからの絶え間ない積極的な協力が必要です。それにもかかわらず、ユーザーは自分のスケッチでこのための関数を提供する必要はありません。これらのことはすべて、バックグラウンドでほとんど気付かれずに発生します。このようなバックグラウンド機能は、ステッピングモーターの動作にも望ましいでしょう。

ATmega32u4のようなマイクロコントローラーftドゥイーノ マイクロプロセッサ（実際の処理装置）と、USBインターフェイス機能などのさまざまな追加ハードウェアコンポーネントで構成されています。特に、ATmega32u4にはいわゆるタイマーがいくつかあります。タイマーは、実際のプロセッサとは独立して動作するクロックと考えることができます。ソフトウェアを使用して、時計の実行速度と特定の時間に特定のことが発生するかどうかを判断できますが、実際の時間の進行は自動的に行われ、プロセッサで実行されたスケッチによる追加のアクションはありません。このようなタイマーによって定期的にトリガーできるものの1つは、いわゆる割り込み要求です。プロセッサに割り込みを発生させ、

このタイプの割り込みは、たとえばステッピングモーターの制御に最適です。ステッピングモーターを移動する場合、たとえば1秒あたり200ステップの場合、プロセッサーが5ミリ秒ごとに中断されるようにタイマーをプログラムできます。この中断の間、プロセッサはモーターの磁場をさらに1ステップ回転させてから、通常のタスクに戻る必要があります。

次のコードセグメントは、ATmega32u4のタイマー1をプログラムして、いわゆる割り込みサービスルーチンが正確に5ミリ秒ごとに実行されるようにします。モーターは、このルーチン内の適切なプログラムコードを使用して、スケッチのメインプログラムとは完全に独立して動作させることができます。

```
1   # 含む <FtduinoSimple.h>
2
3   //いわゆる割り込みサービスルーチン（ISR）は//スケッチ自体では実行されませんが、//
4位 ATmega32uのハードウェアは//タイマーイベントに基づいて実行をトリガーします
5
6日
7日 ISR（（TIMER1_COMPA_vect）{
8日     //この関数は5msごとに実行されます。//たとえば、ステッピングモー
9       ターの磁場が//さらに回転する可能性があります。
10
11日    //   。。。
12日 }
13日
14日 空所   設定（）{
15日    // ATmega32u4のタイマー1の構成、//レジスタの正確な説明は第14章にありま
16      す//
17日          データシートの：
18日    //    http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel -7766-8-bit -AVR -ATmega16U4 -32
              U4_Datasheet.pdf
19日
20日    //タイマー1は、OCR1Aを上限として//いわゆるCTCモードに切り替える必要があります。タイ
21      マーは1 / 256CPUサイクルで動作します。//これは16MHzであるため、タイマーは62.5kHzで動
22日    作します。//モーターを毎秒200ステップ回転させるには、//タイマー312（62500/200）がカ
23      ウントステップを実行したときに、モーターは常にステップを実行する必要があります。
24
25日
26日    TCCR1A = 0;
27      TCCR1B =（1 <<WGM12）|（1 <<CS12）; // 1/256 F_CPU = 62.5kHzでタイマー1を開始します TCCR1C = 0;
28
29
30日    // 6240/200カウンターステップに達したときにイベントをトリガーします TCNT1
31              =  0;
32      OCR1A   =  62500/200;
33
34      //ターゲットステップに達したときにイベント生成をトリガーします TIMSK1 =
35                      (1 <<OCIE1A）;
36  }
37
38  空所 ループ（）    {{
39      //メインルーチンは必要に応じて使用でき、//タイマー1割り込みは//独立して
40      定期的に実行されます
41
```

```
42 }
```

完全な例は以下にあります ファイル例 。FtduinoSimple 。StepperMotorSrv 。それは等しい主にタイマー1の提案された使用法。たとえば、2番目のモーターは、バックグラウンドでタイマー3によって制御され、既存のモーターとは独立して、スケッチのメインプログラムとは独立して制御されます。たとえば、いわゆるプロッタは、セクション7.5に示すように、2つのステッピングモーターを使用して非常にエレガントに実装できます。

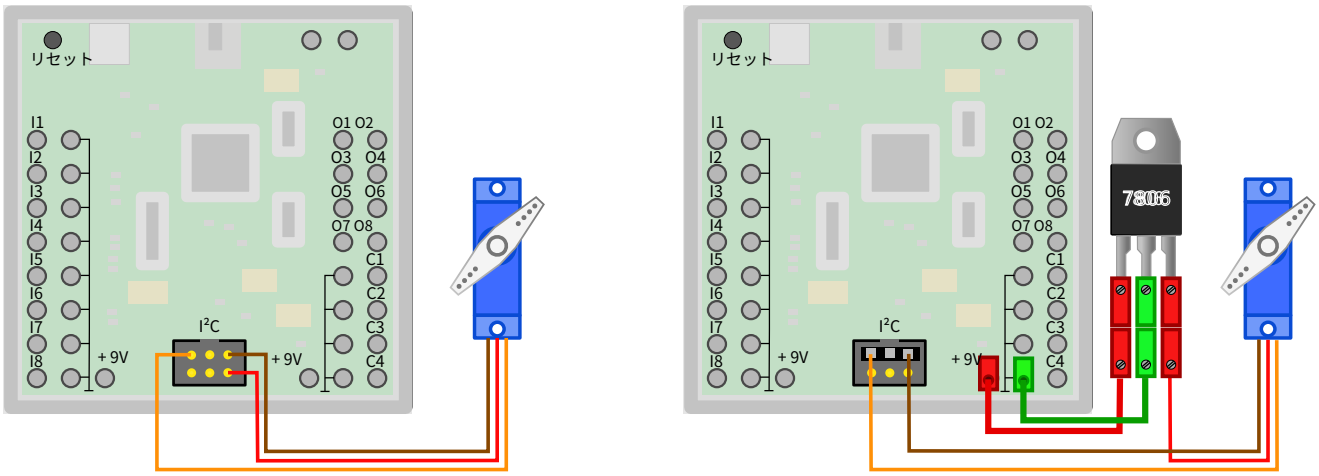
6.5サーボモーター制御

難易度： ★★★☆☆

セクション6.4の通常のDCモーターとステッピングモーターに加えて、第3のタイプのモーター、いわゆるサーボがあり、これは主にモデル構築で使用されます。schertechnikはアイテム番号132292でサーボを販売しています[3]。

技術的には、サーボは単純なDCモーターと単純な電子機器で構成されています。測定機構は、サーボの電流制御値（角度）をこれらの電子機器に常に報告します。電子機器はこれを外部設定値と比較し、必要に応じてモーターを再調整します。サーボは、回転角で外部設定値に従います。

したがって、サーボには3芯接続ケーブルがあります。電圧供給には2本のワイヤーが使用され（赤= 6ボルト、茶色= アース）、3本目のワイヤー（オレンジ）が設定値を送信します。市販のサーボを使用できますが、Schertechnikサーボ132292も使用できます。[4位] 540585から[5]-PLUSBluetoothコントロールセット。また、Schertechnik RC-Servo 30275[6日] 1983年からこのように使用できるはずです。ただし、これはまだ確認されていません。



（a）による供給 ftドゥイーノ （b）7806コントローラーからの供給

図6.15：サーボの接続 ftドゥイーノ

サーボは6ボルトの動作電圧用に設計されており、多くの場合5ボルトで動作するため、内部の5ボルトからの供給 ftドゥイーノ 私の上に。[2]図6.15（a）に示すようにC接続が可能です。サーボの内部電源を供給するためにサーボの消費電流が100mAを超えてはならないため、この電源には注意が必要です。ftドゥイーノ 過負荷にならないように。ほとんどのサーボは明らかにこの値を超えているため、ftドゥイーノ 供給されます。

[3] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=132292
[4位] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=132292
[5] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=540585
[6日] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=30275

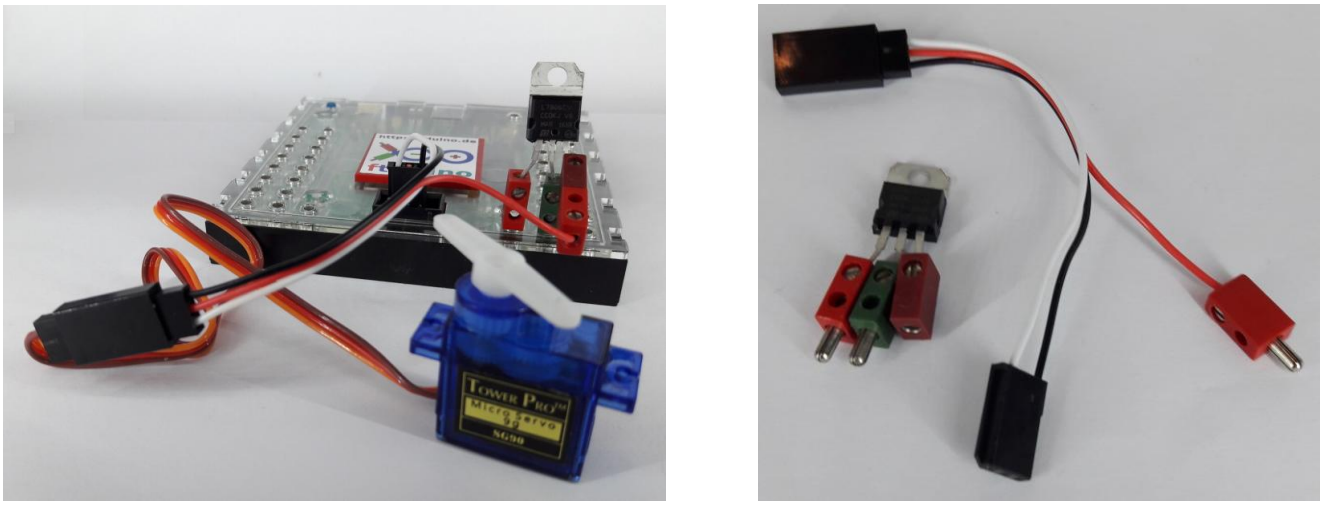
6.5.1外部6ボルト電源

オンライン小売店でこの名前で簡単に見つけることができる、たとえば7806タイプの外部電圧レギュレータを介した供給は、はるかに堅牢で、ほとんど複雑ではありません。このいわゆるシリーズレギュレータは、の9ボルト出力の1つに直接接続できます。ftドゥイーノ 接続され、その出力で6ボルトに低減された電圧を提供します。

必要な部品：

1x 電圧レギュレータ7806
1x サーボ延長ケーブルJRプラグ赤
2倍 フィートプラグ
1x 緑のフィートプラグ
1x 赤いフィートのソケット

7806は Schertechnik標準プラグを差し込んでから、 ftドゥイーノ プラグを差し込む。the
制御信号の接続は、IのSDA接続と一致する必要があります。2Cコネクタ。SCL接続を追加で使用すると、2番目のサーボを接続できます。図6.15（b）に示すように、サーボのコネクタで赤いケーブルを直接切断し、schertechnikコネクタを提供すると、残りの2本のワイヤを備えたサーボのコネクタをIに直接接続できます。2C接続を接続します。サーボの接続ケーブルを切断したくない場合は、図6.5.1に示すように、標準のJRサーボ延長ケーブルを切断することもできます。切断された真ん中の赤い接続は、schertechnikプラグを使用して7806に接続されます。7806にはschertechnikスリーブも付属しています。



（a）コントローラーとアダプターケーブルのサーボ （b）電圧レギュレーターとアダプターケーブル

図6.16：オンのサーボモーター ftドゥイーノ

サーボの制御信号はIに対応していません。2C標準。代わりに、サーボは20ミリ秒ごとに繰り返される単純なパルス幅信号を使用します。パルス自体の長さは1〜2ミリ秒で、サーボがとる角度を決定します。1ミリ秒は最小値を表し、2ミリ秒は最大値を表します。サーボモーターを中間位置で動作させる場合は、それに応じて1.5ミリ秒のパルスが必要です。

サーボにはそれ以上のインテリジェンスがなく、指定された制御信号はチェックされません。サーボはまた、1ミリ秒未満または2ミリ秒を超えるパルス長を対応する角度に変換しようとします。サーボの動きには機械的な制限があり、サーボが通常の動作範囲外の位置に移動しようとすると、サーボが損傷する可能性があることに注意してください。したがって、1〜2ミリ秒の範囲を残すことはお勧めできません。

私のために実際に必要なパルス信号を取得します。2C信号用に提供されたピンを生成するには、ソフトウェアを使用する必要があります。サーボを中央位置に移動するプログラムフラグメントは、次のようになります。

```
1  空所 設定 () {
2      //ポートD.1（SDA接続）を出力に切り替えます
```

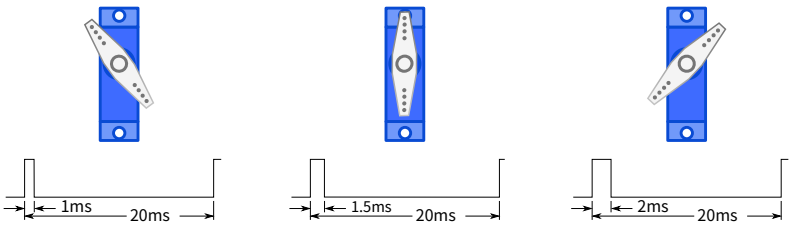


図6.17：制御信号の関数としてのサーボ角度

```
 3      bitSet（（DDRD、1）；
 4位 }
 5
 6日 空所 ループ（）{
 7日    //ポートD.1をハイレベル（5V）に設定します bitSet
 8日     （（移植、1）；
 9
10      // 1500us（1.5ms）待つ _delay_us
11日     （1500）；
12日
13日    //ポートD.1をローレベル（GND）に設定します
14日    bitClear（（移植、1）；
15日
16      // 18500 us（18.5ms）待つ _delay_us
17日     （20000-1500）；
18日 }
```

ここでIのSDA接続2C最初に 設定（）-独立して使用できる出力に構成された関数。その結果、ループ（）-機能レジスタの適切なビットを設定することにより、出力をHi（5ボルト）またはグランド（GND）に切り替えることができます。 移植 設定または削除されます。出力をオンにした後、システムは18.5ミリ秒をオフにした後、1500マイクロ秒（1.5ミリ秒）待機するため、合計20ミリ秒のサイクル時間が達成されます。

ステッピングモーターと同様に、 ftドゥイーノ この単純なタイプのプログラミングでは、信号生成で常に忙しく、他のタスクを実行することはできません。

ステッピングモーターと同様に、解決策は、ハードウェアタイマーを使用してバックグラウンドで信号を生成させることです。例 ファイル例 。FtduinoSimple 。ServoDemo サーボ制御に簡単なクラスをもたらします と。実際のメインプログラムは次のようになります。

```
 1   //
 2   //   Servo.ino
 3   //
 4位
 5   # 含む      「Servo.h」
 6日
 7日 空所 設定（）      {{
 8日    サーボ。始める（）；
 9   }
10
11日 空所 ループ（）{
12日    静的 uint8_t値 = サーボ ：：： VALUE_MAX / 2;
13日
14日    もしも（（価値 < サーボ：：： VALUE_MAX）。 価値 ++; それ以
15日    外                                          価値 = 0;
16      サーボ。セットする（（価値）；
17日
18日    遅れ （10）；
19日 }
```

サーボの制御は、呼び出しに制限されています Servo.begin（）-バックグラウンドで必要なタイマーを設定する機能。サーボの角度は、次のように調整できます。サーボセット（）-0（最小角度）から サーボ:: VALUE_MAX（最大角度）。真ん中の位置は、例えばです。 Servo.set（サーボ:: VALUE_MAX / 2） 近づいた。

6.6の入力 ftドゥイーノ

難易度： ★★☆☆☆

マイクロコントローラーのすべてのピンの方向を切り替えることができるため、Arduinoを扱ったことのある人なら誰でも、入力または出力として使用する接続を比較的自由に決定できることを知っています。でftドゥイーノ Schertechnikのランプとモーターを操作できるようにするために、入力で過電圧と短絡に対する追加の保護回路が使用され、信号出力が増幅されるため、この機能を取得できませんでした。

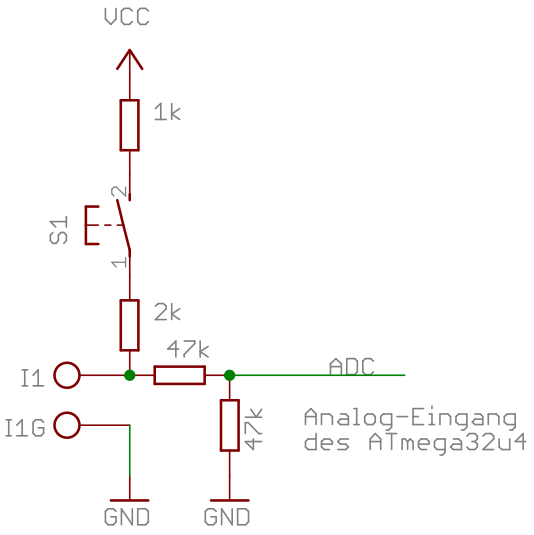


図6.18：入力の内部配線 I1 それまで I8 の ftドゥイーノ

8つの入り口のそれぞれ I1 それまで I8 の ftドゥイーノ ATmega32u4マイクロコントローラの独自のアナログ入力につながり、これとは独立して評価できます。これを行うために、マイクロコントローラーは対応する入力の電圧を測定できます。

6.6.1電圧測定

入力信号がマイクロコントローラに到達する前に、2つの47キロオームの抵抗で構成される分圧器を通過します。これらの抵抗器には2つの目的があります。まず、マイクロコントローラーに到達する前に、印加された信号の電圧を半分にします。マイクロコントローラ以来ftドゥイーノ-5ボルトで内部的に動作し、0〜5ボルトの範囲の信号のみを処理できます。電圧を半分にすると、測定可能な入力電圧範囲が0〜10ボルトに拡張されます。これは、せん断技術で一般的な電圧を最大9ボルトまで処理できることを意味します。次に、これらの抵抗は、マイクロコントローラー内部の保護ダイオードと組み合わせて、マイクロコントローラーを許容できる0〜5ボルトの電圧範囲外の電圧からマイクロコントローラーを保護します。したがって、入力で最大47ボルトの電圧がマイクロコントローラに損傷を与えることはありません。

6.6.2抵抗測定

the 1 kΩ- と 2。2 kΩ-スイッチが開いている限り、抵抗とスイッチは関係ありません。これらのスイッチのうち8つ（入力ごとに1つ）は、ftドゥイーノ 付録Aの回路図に示されているように、CD4051（IC1）と指定されたモジュール内。マイクロコントローラは、いつでも8つのスイッチの1つを正確に閉じることができ、このようにして総抵抗を閉じることができます。3。2 kΩ（1 kΩ プラス 2。2 kΩ） それぞれの入力から5ボルトに向かってアクティブにします。

抵抗測定を行う場合は、スイッチを閉じて抵抗をアクティブにします。the3。2 kΩ- 次に、抵抗は入力とグランドの間に接続された外部抵抗と分圧器を形成します。次に、測定された電圧から未知の外部抵抗を決定できます。

抵抗測定は、 Ftduinoライブラリはバックグラウンドで実行されています。抵抗測定のスケッチで現在使用されているすべての入力への抵抗測定の自動切り替えもあります。プログラマーは詳細を気にする必要はなく、いつでもこの関数を使用して抵抗値を決定できますftduino.input_get（） （9.2.2を参照）。

6.6.3出力としての入力

抵抗測定の場合、抵抗が5ボルトに切り替えられるという事実は、測定対象の外部接続された抵抗を介して回路が閉じられることを意味します。この回路を流れる電流は比較的小さいです。入力が直接グランドに接続されている場合、総抵抗は次のようになります。3。2 kΩ そしてそれはの流れを流します 私。 = 5*V*/3。2 kΩ= 1、 5625*mA*。

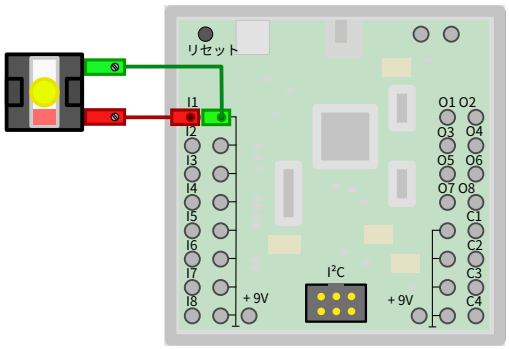


図6.19：入力へのLEDの接続 I1 の ftドゥイーノ

この電流は、ランプやモーターを動作させるのに十分ではありません。ただし、発光ダイオードを弱く光らせることができます。発光ダイオードを入力とアースの間に直接接続し、入力を抵抗測定に切り替えると、LEDがごくわずかに点灯します。

実際の電流は、予測された1.5mAをはるかに下回ります。これは、一方で、約0.7Vのいわゆる順方向電圧が発光ダイオードに直接降下するため、両端に4ボルトをわずかに超える電圧しかないためです。抵抗器。

一方、彼女は尋ねます Ftduino8つの入力すべてをバックグラウンドでライブラリし、各入力のみをアクティブにします 1⁄8日 現在。したがって、それは平均して食べるだけです1⁄8日 ストリームの。

the FtduinoSimple-ライブラリは、最後にアクティブ化された入力の抵抗もオンにします。この抵抗は、別の入力が要求されるまで永続的にアクティブになります。次のコードフラグメントは、入力にLEDを残しますI1 毎秒点滅します。

```
1    # 含む <FtduinoSimple.h>
2
3    空所 ループ （） {
4位     //入力I1の値を読み取り、I1の抵抗をアクティブにします ftduino。input_get （（
5       Ftduino ： ： I1）；遅れ （1000）；
6日
7日     //入力I2の値を読み取り、I1の抵抗を非アクティブ化します//（そしてI2でアクティブ化
8日     します）
9       ftduino。input_get （（Ftduino ： ： I2）；
10      遅れ （1000）；
11日 }
```

6.7温度測定

難易度： ★☆☆☆☆

Fischertechnikはアイテム番号36437で販売されています[7日] いわゆるNTC。この目立たないコンポーネントは、一部のロボットキットに含まれています。

NTCは、周囲温度に応じて値が変化する電気抵抗です。したがって、温度測定に適しています。NTCは負の温度係数の略で、温度の上昇とともにオーム抵抗が減少することを意味します。NTCは、導電率が温度とともに増加するため、ドイツ語では高温導体としても知られています。

公称抵抗 *R*。ɴNTCは通常25の温度にあります °°C（298.15 K）が述べた。のためのもの
schertechnikセンサーは指定値です 1。5 kΩ。 オーム抵抗は25です °°Cそう 1。5 kΩ。

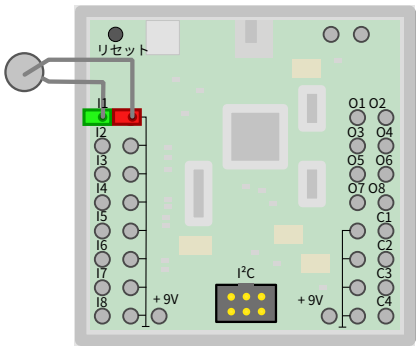


図6.20：入力への温度センサーの接続 I1

6.7.1スケッチ 温度

次のスケッチは、 ftドゥイーノ-ArduinoIDEのメニューでのサポート ファイル
例 。Ftduino 。温度 。

```
1   //
2   //   Temperaure.ino
3   //
4位 //入力I1でfischertechnikの温度抵抗をクエリします//
5
6日 //（c）2018 by Till Harbaum < till@harbaum.org > //
7日
8日
9   # 含む      <Ftduino.h>
10  # 含む      <math.h>                  //浮動小数点演算の場合
11日
12日 # 定義 K2C 273.15                    //ケルビンを摂氏にオフセット//センサーのいわゆ
13日 # 定義 B 3900.0                      るB値
14日 # 定義 R_N 1500.0                    //摂氏25度の基準温度での抵抗//ケルビン単位の基準温度
15日 # 定義 T_N（K2C + 25.0）
16
17日 浮く r2deg（（uint16_t r）{
18日    もしも（（r == 0） 戻る NAN;       // 0オームの抵抗は意味のある温度を与えません
19日
20日    //抵抗をケルビンに変換します
21      浮く t = T_N * B。 /（B。 + T_N * ログ（（r / R_N））；
22日
23      //ケルビンを摂氏に変換します 戻る t - K2C;
24
25日
26日    //または、ケルビンを華氏に変換します
27  //     t * 9/5を返す-459.67;
28  }
29
30日 空所   設定（）{
31      //   LEDを初期化します
32      pinMode（（LED_BUILTIN、    出力）；
```

[7日] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=36437

```
33      digitalWrite（（LED_BUILTIN、 低い）；
34
35      シリアル。始める （115200）；そ
36      の間（！ シリアル）；
37
38      ftduino。初期化 （）；
39
40      //温度は
41      ftduino。input_set_mode（（Ftduino：：：I1、 Ftduino：：： 抵抗）；
42  }
43
44  空所   ループ （）{
45      uint16_t r = ftduino。input_get（（Ftduino：：：I1）；
46
47      シリアル。印刷（（「I1：」）；シリアル。印刷
48       （（r2deg（（r））；シリアル。println（（「摂
49      氏」）；
50
51      遅れ （1000）；
52  }
```

スケッチの説明

温度スケッチは、温度を格納するために、いくつかの場所でいわゆる浮動小数点数を使用します。これは、コンピューター技術で使用される非整数値（小数）の内部表現に付けられた名前です。この目的のために、スケッチは10行目のファイルをバインドしますmath.h スケッチに浮動小数点関数へのアクセスを提供します。データ型は、浮動小数点数を格納するために使用されます浮く 例：21行目で使用されています。

温度測定に使用するセンサーは抵抗器であるため、 設定（）-41行目の関数は入力です I1 の ftドゥイーノ 抵抗測定に設定します。

実際の抵抗値は入力の45行目に示されています I1 整数変数で読み取ります r 提出した。48行目の値が出力されている間、関数r2deg（） と呼ばれる。この関数は、17行目から28行目にあります。オーム単位の整数の抵抗値を受け入れ、浮動小数点値として摂氏温度を返します。

まず、21行目で抵抗がケルビンに変換されます。これは抵抗に加えて行われます$R_{。N}$25で °°Cいわゆるいわゆる $B_{。}$-必要なセンサーの値。この値は、25日以降のセンサーの動作を表します°°Cポイントと抵抗が温度変化にどれだけ強く反応するか。この値は、によって販売されているセンサーの3900です。 せん断技術

NTCの場合、以下がおおよそ適用されます[8日]：

$$\frac{1}{T} = \frac{1}{T_N} + \frac{1}{B_{。}} \ln\left(\frac{R_{。T}}{R_{。N}}\right) \Leftrightarrow T = \frac{T_N * B_{。}}{B_{。} + T_N * \ln\left(\frac{R_T}{R_N}\right)}$$

と

?? T。。。現在の温度

?? T_N。。。公称温度（通常25 °°C）

?? $B_{。}$ 。。。B値

?? $R_{。T}$。。。現在の温度での抵抗

?? $R_{。N}$。。。公称温度での抵抗

変換後、温度はケルビンで利用可能になります。摂氏に変換するには、24行目を定数から引くだけです。華氏への変換はもう少し複雑で、例として27行目に示されています。

温度測定の精度は抵抗測定の精度に直接依存し、セクション1.2.5で説明したように、これは電源に依存します。温度を測定するには、ftドゥイーノ したがって、9V電源から電圧を供給することができます。USBインターフェースのみでの供給では不十分です。

8日https://de.wikipedia.org/wiki/Hei%C3%9Fleiter

6.8出力はオン、オフ、またはどれも出力しませんか？

難易度： ★★☆☆☆

出力はオンまたはオフに切り替えることができます。これが一般的なビューです。しかし、一見したところ、別の条件があることは明らかではないかもしれません。

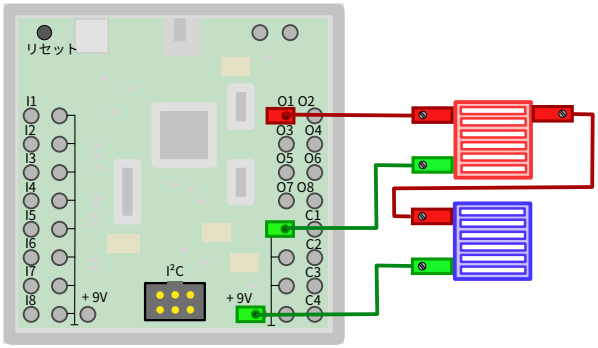


図6.21：出口にある2つのランプ O1

の出力 ftドゥイーノ 次の3つの状態で切り替えることができます。 Ftduino :: HI、Ftduino :: LO と Ftduino ::オフ。

最も明白なのは状態です Ftduino ::こんにちは。 この状態では、対応する出力 ftドゥイーノ-9ボルトの供給電圧に内部接続されています。2番目の接続がグランドに接続されるようにランプまたはモーターがこの出力に接続されている場合、電流は9Vソースから出力を介してランプまたはモーターを介してグランドに流れます。モーターが回転し、ランプが点灯します。示されている例では、赤いランプが点灯しています。

状態で Ftduino :: LO 対応する出力はグランドに接続されています。ランプの両方の接続がアースに接続されているため、2つの接続間の電圧が0ボルトであるため、アースへの2番目の接続に再び接続されているランプは点灯しなくなります。ランプの2番目の接続を9ボルトに接続すると、ランプが点灯します。電流はの電源から流れますftドゥイーノ 9 V接続を介して、ランプを介して、最後に接地された出力を介して。示されている例では、青いランプが点灯しています。

最後に、3番目の状態は状態です Ftduino ::オフ。 この場合、出力は完全に開いています。アースまたは9ボルトに接続されておらず、電流が流れません。その結果、出力からの電流が両方のランプの影響をまったく受けないため、両方のランプが半分の明るさで点灯します。 ieÿt。この状態は、多くの場合、英語の用語 トライステートおよびトライステート対応としての半導体上の対応する出力。ドイツの高抵抗で用語を説明します は、この3番目の状態は非常に良好です。

次のスケッチは、3つの状態の間で毎秒変化します。この効果は、たとえば、出力を節約するために、1つの出力で2つのモーターまたはランプを独立して制御するために使用できます。ただし、この接続では、両方のランプを同時にオフにすることはできません。

6.8.1スケッチ OnOffTristate

```
1   /*
2      OnOffTristate-3番目の状態
3   */
4位
5   # 含む <FtduinoSimple.h>
6日
7日 空所 設定 () {}
8日
9   //ループ関数は何度も何度も何度も実行されます 空所 ループ () {
10
11日    //出力O1を9Vに切り替えます ftduino。
12日    output_set ((Ftduino ：：：O1 、 遅れ       Ftduino ：：：こんにちは) ;
13日     (1000) ;
14日    //出力O1をグランドに切り替えます ftduino。
15日    output_set ((Ftduino ：：：O1 、            Ftduino ：：：LO) ;
```

```
16      遅れ（1000）;
17日    //出力O1を未接続のままにします
18日    ftduino。output_set（（Ftduino：：：O1、 Ftduino：：：オフ）;遅
19日    れ（1000）;
20日 }
```

6.8.2漏れ電流

高抵抗またはトライステート状態では電流が流れないという記述は完全に正しいです。多くの場合、低電流が電力出力段とその内部保護回路を流れます。場合によっては、これは意図的に行われることもあります。たとえば、この低電流の流れを利用して、接続された消費者の存在を判断できるようにするためです。これらのいわゆるリーク電流は、セクション6.1.1ですでに観察されています。

現在のモデルの2つのランプを2つの発光ダイオードに置き換えると、対応する出力が高抵抗に切り替えられたときに、出力からグランドに接続されたLEDが常にわずかに点灯することがわかります。出力がグランドに切り替えられた場合にのみ、LEDは点灯しません。したがって、LED上で3つの状態を直接区別できます。

6.9アクティブエンジンブレーキ

難易度： ★★☆☆☆

エンジンのスイッチを切ることは、純粋に電気的な観点からは些細なことのようです。モーターが電源から切断されるとすぐに停止します。本質的に、それは本当です。

物理的には、電源からの分離は、モーターにそれ以上のエネルギーが供給されないことを意味します。これが最終的にモーターの停止につながるのは、モーターの回転に蓄えられたエネルギーが、たとえばモーターシャフトのベアリングなどの摩擦によってゆっくりと失われているためです。このようにモーターが停止するまでにかかる時間は、モーターの設計とベアリングの品質に大きく依存します。

さらに、せん断技術で使用されるものなど、多くの直流電気モーターが発電機として機能します。それらが回転すると、内部の電磁石に電圧が誘導されます。

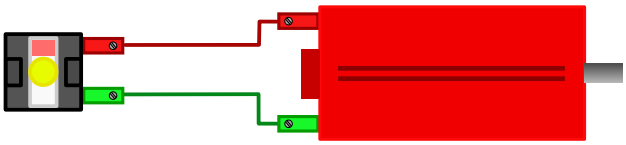


図6.22：発電機としてのTXTエンコーダーモーター

この効果は、発光ダイオードで簡単に理解できます。発光ダイオードを直接モーターに接続し、モーター軸を手動で回すと、モーターを正しい方向に回すと発光ダイオードが点灯し、発光と一致する極性の電圧が発生します。ダイオード。この実験では、白熱灯や2番目のモーターを使用することもできます。ただし、発光ダイオードと比較してエネルギー消費量が多いため、多少強力な回転が必要になる場合があります。

発電機が供給する負荷が大きくなり、からより多くのエネルギーが引き出されるほど、発電機を回転させるために必要な機械的な力が大きくなります。この場合、負荷が高いほど電気抵抗が低くなります。負荷が比較的低い発光ダイオードは電気抵抗が高く、白熱灯などがあり、モーターの電気抵抗が低く、発電機に大きな負荷やブレーキをかけます。この場合に考えられる最大の負荷は短絡です。電気抵抗が最小で、最大の電流が流れ、発電機に最大の電気負荷がかかります。ブレーキ効果も最大です。

この効果は、電気モーターにブレーキをかけるために使用できます。モーターの両方の接続が相互に接続されている場合、ブレーキ効果を発生させる電流が生成されます。これは、セクション6.2の緊急停止モデルですでに使用されており、緊急時にエンジンをすばやく停止します。一方、モーターの接続の1つが開いている場合、それは

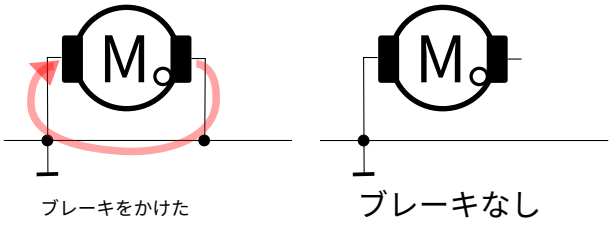


図6.23：電気的にブレーキがかけられた電気モーターとブレーキがかけられていない電気モーター

閉回路も電流も流れず、ブレーキ効果もありません。この効果はどれくらいの大きさですが？

schertechnikエンコーダモーターには、セクション6.3のPWM実験ですでに使用されている速度測定の可能性が含まれています。したがって、このモーターのブレーキ動作は、実験的に簡単に追跡できます。

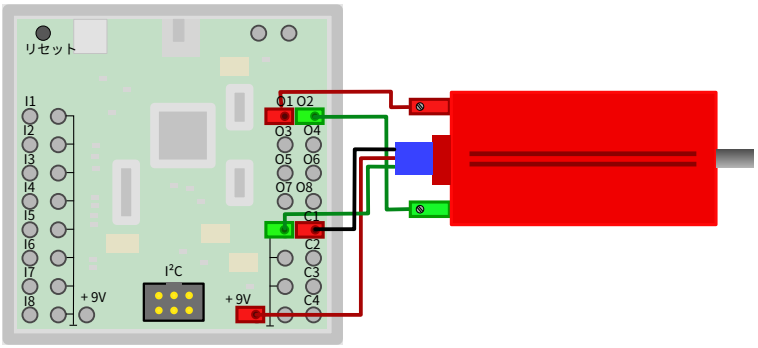


図6.24：TXTエンコーダモーターの接続への接続 M1 と C1

例 ファイル例 。Ftduino 。モーターブレーキ モーターを出力に残します M1 5秒ごとに3回回転が実行され、エンコーダが入力にあるパルスの数をさらに1秒間測定します C1 3回転を完了し、スイッチをオフにした後に配信します。

関数 motor_counter_set_brake（）（セクション9.2.9を参照）は、モーターが自由に惰走するように、またはモーターがアクティブにブレーキをかけられるように、交互に呼び出されます。

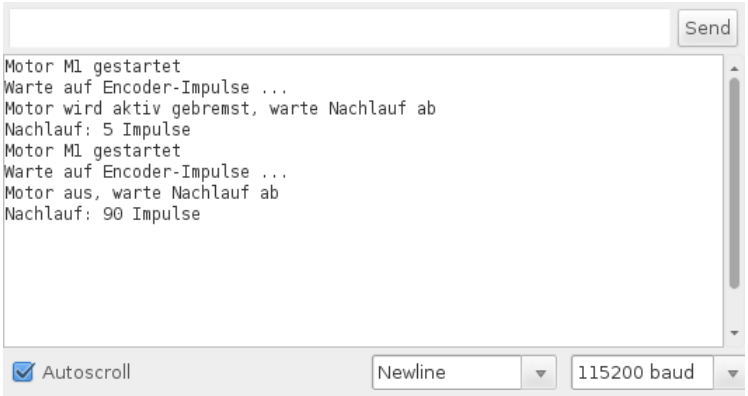


図6.25：TXTエンコーダモーター使用時の出力

図6.25に示すように、アクティブブレーキは明確な違いをもたらします。エンコーダモーターはブレーキなしでさらに90インパルス、つまりほぼ1.5回転し続けますが、アクティブブレーキを使用すると、さらに5回インパルスすると停止します。それはほぼ同じです1/13日 革命。

6.10USBキーボード

難易度： ★★☆☆☆

the ftドゥイーノ 古典的なArduinoUnoからではなく、ArduinoLeonardoから派生しています。2つのArduinoの主な技術的な違いは、Arduino UnoがPCとのUSB通信に別のチップを使用するのに対し、ArduinoLeonardoではこのタスクはATmega32u4マイクロコントローラーのみに任されているという事実にあります。

ほとんどの場合、違いはなく、ほとんどのスケッチは両方のArduinoで同じように実行されます。ただし、両方のArduinoがUSB接続に提供する可能性には非常に大きな違いがあります。宇野のUSBチップが作成中COM：-レオナルドは自分自身が制限されていることを示しています。 ftドゥイーノ はるかに柔軟で ftドゥイーノ とりわけ、PCのUSBキーボードのふりをすることができます。

の出力以来 ftドゥイーノ このモデルを使用できない場合は、USB経由の電源で十分であり、バッテリーや電源を介した追加の電源は必要ありません。

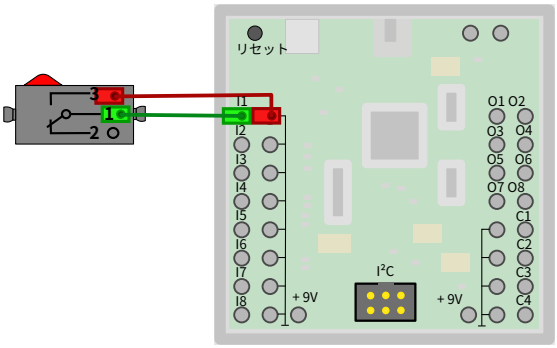


図6.26：キーボードメッセージ

6.10.1スケッチ USB / KeyboardMessage

```
1   / *
2      KeyboardMessage-USBキーボード
3
4位    ftDuinoはUSBキーボードのふりをして、入力I1のボタンが少なくとも10ミリ秒押されるとすぐに、メッセージを
5      「入力」します。
6日
7日    スケッチに基づく：
8日    http：//www.arduino.cc/en/Tutorial/KeyboardMessage
9
10     このサンプルコードはパブリックドメインです。
11日 * /
12日
13日 # 含む       <FtduinoSimple.h>
14日 # 含む       <Keyboard.h>
15日
16   unsigned long lastButtonEvent = 0; uint16_t
17日 previousButtonState = Ftduino ：：：オフ;                    //プッシュボタンの状態を確認するため
18日
19日 空所 設定 （） {
20日    //キーボードの制御を初期化します： キーボード。始める
21       （）；
22日 }
23
24   空所 ループ （） {
25日    //入力I1でキーを読み取ります
26日    uint16_t buttonState = ftduino。input_get （（Ftduino ：：：I1）；
27
28      //キーの状態は変更されましたか？ もしも （（buttonState ！=
29      previousButtonState）{
30日       //はい、変更の時間を覚えておいてください
```

```
31          lastButtonEvent = ミリス （）;
32          //さらに変更を認識できるように、//新しいステータスをメモします
33
34          previousButtonState= buttonState;
35      }
36
37      //未処理のイベントがあり、キーの状態が//それから10ミリ秒以上変更されていませんか？
38
39      もしも （（lastButtonEvent &&（（ミリス （）-lastButtonEvent）> 10））{
40          //このイベントの時間を忘れる lastButtonEvent = 0;
41
42
43          //ボタンが押されました もしも
44           （（buttonState）。   {{
45             // ニュース          「タップ」
46              キーボード。println （（"こんにちは     ftDuinoから！」）;
47          }
48      }
49  }
```

スケッチの説明

Arduino IDEには、マウスやキーボードなどのUSBデバイスを実装するためのライブラリがすでに付属しています。実際のスケッチは非常に単純なままであり、複雑なUSBの詳細はライブラリに隠されたままです。このスケッチはそれに応じて短いです。

の中に設定（）-メソッドのみが機能する必要がありますKeyboard.begin（）の開始時に呼び出されますftドゥイーノ USB側ですべての予防策を講じてftドゥイーノ PCによってUSBキーボードとして認識されます。ただし、このキーボードには最初はキーがないため、PCが追加のキーボードを備えていると見なしていることに気付くことはほとんどありません。

キーボードを生命で満たすためには、キーボードが ループ（）-機能は、必要に応じて、対応するキー信号を生成してPCに送信できます。スケッチの25行目から35行目には、入力にボタンがありますI1 照会され、10msより長いキーストロークのみがそのように認識されることを確認しました（このいわゆるデバウンスの詳細については、セクション6.12を参照してください）。

ボタンがオンになっているときはいつでも I1 を押すと、スケッチ線４５以降が実行される。これは、関数がここにありますKeyboard.println（） Arduinoキーボードライブラリから呼び出され、PCにテキストを送信しました。PCの場合、ユーザーがキーボードでテキストを入力しているように見えます[9]。

キーボードエントリとしてメッセージを直接送信する可能性は、PCでさらにプログラミングしなくても、測定値をテーブルなどに自動的に入力できるため、非常に実用的です。もちろん、この能力は、を使用してあらゆる種類のジョークに使用することもできますftドゥイーノ 時間管理された、または他のイベントに反応することは、予期しないテキスト入力で驚いたユーザーを苛立たせます。このような夜の外出では、間違った時間に間違ったキーを押すとデータが簡単に失われる可能性があるため、常に十分な注意を払う必要があります。

6.11USBゲームパッド

難易度：

PCの観点からは、USBキーボードとUSBジョイスティックまたはゲームパッドの違いはごくわずかです。どちらもいわゆるUSB-HIDプロトコルを使用します（HID =ヒューマンインターフェイスデバイス、つまり人のためのインターフェイスデバイス）。ただし、Arduino側では、Arduino環境にキーボード用のライブラリ関数が事前に作成されているが、ゲームパッドやジョイスティック用にはないという根本的な違いがあります。とにかくUSBゲームパッドを実装するには、スケッチでさらに多くの努力を払う必要があります。

[9]マウスとキーボードのシミュレーション用のArduinoライブラリの詳細と詳細な説明は、次のURLにあります。 https://www.arduino.cc/en/Reference/MouseKeyboard

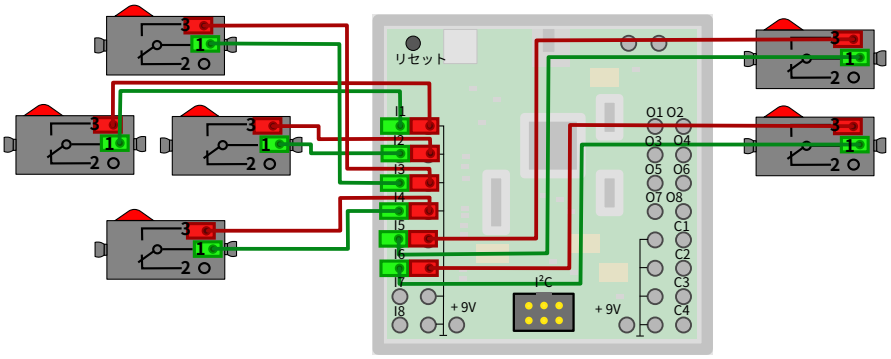


図6.27：4つの方向キーと2つの発射ボタンを備えたゲームパッド

6.11.1スケッチ USB /ゲームパッド

対応する例は以下にあります ファイル例 。FtduinoSimple 。USB。ゲームパッド 。このスケッチは3つのファイルで構成されています。その間GamePad.ino 実際のスケッチを実装する HidGamePad.cpp と HidGamePad.h ArduinoIDEが提供していないゲームパッドサポートのその部分。特におもしろいもの _hidReportDescriptor-ファイル内の構造 HidGamePad.cpp。

```
 9  静的定数 uint8_t _hidReportDescriptor [] プログラム = {
10      0x05 、0x01 、                  // USAGE_PAGE（汎用デスクトップ）//
11日    0x09 、   0x05 、                   使用法（ゲームパッド）
12日    0x85 、   REPORT_ID 、          //   REPORT_ID（3）
13日    0xa1 、   0x01 、               //   コレクション（アプリケーション）
14日    0x09 、   0x01 、               //      使用法（ポインタ）
15日    0xa1 、   0x00 、               //      コレクション     （物理的）
16      0x09 、   0x30 、               //         利用方法（パツ）
17日    0x09 、   0x31 、               //         利用方法（Y）
18日    0x15 、   0x00 、               //         LOGICAL_MINIMUM（0）
19日    0x26 、   0xff 、   0x00 、     //         LOGICAL_MAXIMUM（255）
20日    0x35 、   0x00 、               //         PHYSICAL_MINIMUM（0）
21      0x46 、   0xff 、   0x00 、     //         PHYSICAL_MAXIMUM（255）
22日    0x75 、   0x08 、               //         REPORT_SIZE（8）
23      0x95 、   0x02 、               //         REPORT_COUNT（2）
24      0x81 、   0x02 、               //         INPUT（Data、Var、Abs）
25日    0xc0 、                         //         END_COLLECTION
26日    0x05 、   0x09 、               //      USAGE_PAGE（ボタン）
27      0x19 、   0x01 、               //      USAGE_MINIMUM    （ボタン   1）
28      0x29 、   0x02 、               //      USAGE_MAXIMUM    （ボタン   2）
29      0x15 、   0x00 、               //      LOGICAL_MINIMUM（0）
30日    0x25 、   0x01 、               //      LOGICAL_MAXIMUM（1）
31      0x95 、   0x02 、               //      REPORT_COUNT（2）
32      0x75 、   0x01 、               //      REPORT_SIZE（1）
33      0x81 、   0x02 、               //      INPUT（Data、Var、Abs）
34      0x95 、   0x06 、               //      REPORT_COUNT（6）
35      0x81 、   0x03 、               //      INPUT（Const、Var、Abs）
36      0xc0                            //      END_COLLECTION
37  };
```

この比較的不可解な構造は、USBHIDデバイスの機能を説明しています[10]。それはそれがどんなタイプのデバイスであるか、そしてジョイスティックの場合にはそれがどんな軸とボタンを持っているかを説明します。

この場合、デバイスは2つの軸XとYを持ち、それぞれが0から255の値の範囲をカバーしていることを報告します。オンとオフのステータスのみを認識する2つのボタンもあります。この説明は、単純なジョイスティックには十分です。ただし、説明を拡張して、追加の軸とボタンを提供することは簡単に可能です。合計8つのアナログ入力と4つのデジタル入力で、ftドゥイーノ 複雑な入力デバイスに十分な接続オプション。

一般的なHIDデバイスは、キーボード、マウス、ジョイスティックまたはゲームパッドです。しかし、いわゆるの仕様

[10]詳細については、 http://www.usb.org/developers/hidpage/

10個のHID使用状況テーブル[11日] さまざまなスポーツ、VR、シミュレーション、医療機器などに、はるかに多くのオリジナルの入力デバイスを提供します。そしてもちろん、ftドゥイーノ たとえば、力のフィードバックの形でランプまたはモーターを介してフィードバックを実装することも可能です。

6.12デバウンス

難易度： ★★★☆☆

以前のスケッチのいくつかでは、ボタンを照会するために予想外の量の努力が払われました。の中にPwmセクション6.3.1のスケッチでは、35行目と51行目、および KeyboardMessage- セクション6.10のスケッチでは、時間も31行目と39-41行目に記録され、キー押下の評価に挿入されました。なぜこれが必要なのかという問題は、もう少し詳しく検討する必要があります。

個々のキーストロークを評価するときにこの時間を使用する理由は、バウンスと呼ばれるものです。メカニカルボタンは、分離されているか接触している2つの金属接点で構成されています。静止時には、接点が分離され、ボタンが押されると、メカニズムによって2つの金属接点が確実に接触し、接点が閉じます。

次のスケッチは、入力I1のボタンを継続的に照会し、 COM：-ステータスが変化すると、ポートはメッセージを発行します。さらに、状態がすでに全体的に変化した頻度をカウントします。

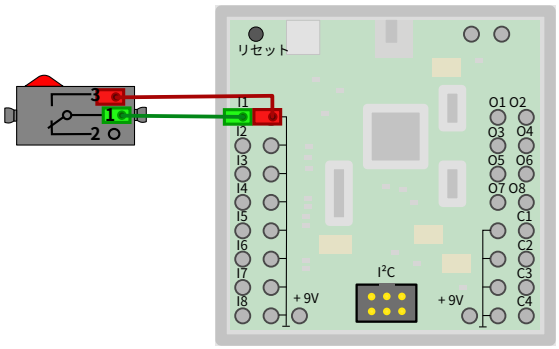


図6.28：デバウンス

6.12.1スケッチ デバウンス

```
/ *
   デバウンス

4位   デモンストレーション   キーバウンス
* /

# 含む <FtduinoSimple.h>

//セットアップ関数は起動時に1回呼び出されます 空所 設定 () {

   シリアル。始める (9600);

   その間 (! シリアル);            // USB接続を待ちます

   シリアル。println ( "ftDuinoキーバウンスの例" );
}

uint8_t last_status = false; uint8_tカウンター
の変更 = 0;

//ループ関数が何度も呼び出されます
```

11日http://www.usb.org/developers/hidpage/Hut1_12v2.pdf

```
22日 空所 ループ（）{
23      uint8_t状態 = ftduino。input_get（（Ftduino：：：I1）；                    //読み上げボタン
24
25日    もしも（（状態 ！= last_status）{                                  //状態は変わりましたか？//はい、カウンターを1
26日       カウンターを変更する = カウンターを変更する + 1;                  つ増やします
27
28          シリアル。印刷（（「I1」）；シリアル。印刷                      //そしてメッセージを印刷します
29           （（カウンターを変更する）；シリアル。println
30日         （（「時代が変わった」）；last_status = 状態;
31                                                                    //新しい状態を最後に記憶します
32      }
33  }
```

スケッチの説明

10行目から16行目は、 ComPort3.3節の例では、PCへの出力を準備し、シリアルモニターのメッセージを出力しました。

23行目では、入力は継続的に行われています I1 照会されました。変数の状態と比較した状態last_status 変更された場合、これは25行目で決定されます。その結果、変数 カウンターを変更する が増加し、28行目から30行目に新しい値が出力されます。

タスク1：それはあまりにも重要です

スケッチをクリックすると何か奇妙なことが起こります ftドゥイーノ ロードして試してみました：ボタンを押すとすぐに、入力のステータスの変化に関するいくつかのメッセージが表示され、カウンターは予想よりも大幅にカウントされます。何が起こっていますか？

問題は、スイッチを切り替えた瞬間に、接点がすぐに完全に閉じないことです。代わりに、金属表面が短時間接触し、数マイクロ秒の間跳ね返り、非常に短時間の間再び開きます。数回のばね操作の後でのみ、接点が停止し、完全に閉じられます。

解決策1：

この問題の最も簡単な解決策は、切り替えイベントの後で別の問題を受け入れる前に少し待つことです。これは、たとえば、31行目以降で何かを待つことで実現できます。Pwmセクション6.3.1からのスケッチが行われました。

```
31          last_status = 状態; 遅れ （10）；                              //新しい状態を最後として記憶します// 10ミリ秒待
32                                                                        ちます
33      }
```

この変更後、スケッチは実際には1回のキーストロークのみをカウントします。ただし、この単純なソリューションには1つの欠点があります。ボタンが押されるたびに、スケッチ全体の実行が10ミリ秒一時停止されます。スケッチに実行する他のタスクがある場合、これらのタスクの処理もこれらの10ミリ秒の間中断されます。使用するボタンによっては、時間をミリ秒未満に短縮することができます。ただし、待ち時間が短すぎると、間違ったイベントが再度認識されます。

したがって、この機能を備えたすべてのイベントでよりエレガントになります ミリス（） システムタイムカウンターからタイムスタンプを取得し、最後のイベントが10ミリ秒以上前の場合にのみイベントを有効として認識します。the KeyboardMessage-セクション6.10のスケッチは、まさにこの方法で問題を解決します。

演習2：今何が起こっているのですか？

ボタンがバウンスする時間とその動作は、推測することしかできませんでした。できますかftドゥイーノ ボタンの切り替え動作を詳しく見てみましょう。

解決策2：解決策2：

Arduino IDEには、信号プロセスを説明するための非常にシンプルですが興味深いツールがあります。いわゆるシリアルプロッタは、下のメニューにあります。 ツール 。シリアルプロッタ シリアルのように開きます独自のウィンドウを監視します。しかし、代わりにCOM：-受信したテキストをポートで直接表示するために、シリアルプロッターは入力データを1行ずつ、曲線でグラフィカルに表示（プロット）される値として解釈します。

次の例を以下に示します ファイル例 。FtduinoSimple 。BounceVisu 見つけられる。

```
 1  / *
 2     BounceVisu
 3
 4位   視覚化              キーバウンス
 5   * /
 6日
 7日 # 含む       <FtduinoSimple.h>
 8日
 9   # 定義      EVENT_TIME 480            //   480us
10   uint8_t     イベント[EVENT_TIME ];
11日
12日 //セットアップ関数は起動時に1回呼び出されます 空所 設定 () {
13日
14日    シリアル。始める (9600) ; そ
15日    の間 (！ シリアル) ;              // USB接続を待ちます
16   }
17日
18日 //ループ関数が何度も呼び出されます 空所 ループ () {
19日
20日
21      //ボタンが押されるまで待ちます
22日    もしも ( (ftduino。input_get ( (Ftduino ： ： I1) ) {
23
24          //マイクロ秒ごとに480マイクロ秒の入力値をフェッチします にとって ( (uint16_t
25日                          私= 0;私<EVENT_TIME;私++) { ftduino。
26日          イベント[私] =    input_get ( (Ftduino ： ： I1) ;
27          _delay_us (1) ;
28        }
29
30日      //最初に20個のゼロを出力します にとって
31          ( (uint16_t i= 0;私<20;私++)
32            シリアル。println (0) ;
33
34          //読み取った480の値を出力します にとって ( (
35          uint16_t i= 0;私<EVENT_TIME;私++)
36            シリアル。println ( (イベント[私]) ;
37
38          // 一瞬待って 遅れ (1000) ;
39
40      }
41   }
```

スケッチは22行目でキーが入力されるのを待ちます I1 が押されました。それから彼は少しの間入り口の状態を描きます I1 の上。9行目は、480個の値が記録されることを指定しています。行27は、2つの記録の間に1マイクロ秒待機するため、合計480マイクロ秒が記録されます。記録が完了すると、最初の20行のゼロが出力され、次に以前に記録された480の値が出力されるため、合計500の値が出力されます。最初の20の値は、ボタンがまだ押されていないときの録音前の状態を表します。

シリアルプロッタは、合計500個の値を曲線として表示します。値は、接点が開いていると認識されるとゼロになり、接点が閉じられるとすぐに1になります。グラフィックでは、ボタンが約40マイクロ秒の間数回開閉し、その後、信号が100マイクロ秒以上安定してから、接点がさらに数回開き、最後に合計200マイクロ秒後に安定して閉じることがわかります。したがって、ソリューション1で使用される一時停止は、バウンスが影響を与えることなく、200マイクロ秒に短縮できます。

PCへの文字の送信には比較的長い時間がかかるため、出力する前に値を完全に記録して保存する必要があります。値がすぐにPCに送信された場合、データ送信自体に時間がかかるため、マイクロ秒の解像度は達成できません。実際、それも持続します

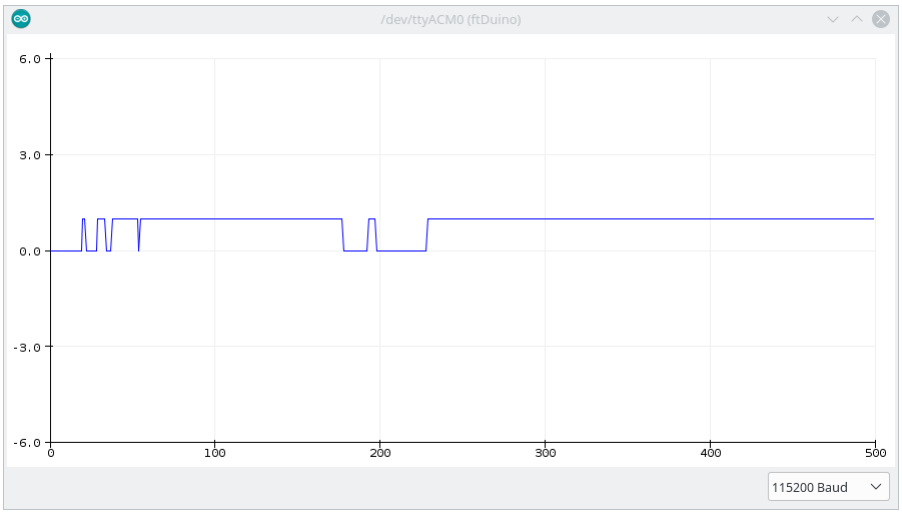


図6.29：シリアルプロッタでのバウンスのコース

入力を読み取る I1 しばらくの間、測定のタイミングはあまり正確ではありません。ただし、基本的なプロセスを示すだけで十分です。

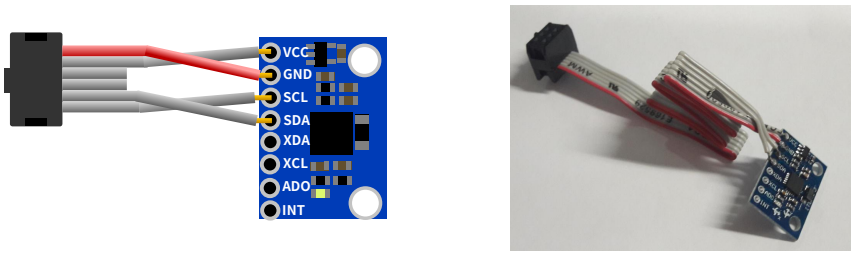
6.13Iの使用2Cバス

難易度： ☆☆☆☆☆

セクション1.2.6で説明されているように、 ftドゥイーノ 私について。2Cコネクタ。Arduinoの世界では、私は2C-Busは、多数の安価な拡張モジュールを簡単に接続できるため、非常に人気があります。

the ftドゥイーノ Iには保護キャップが付いています。2セクション1.2.6に示すように配信されるC接続。このキャップは、Iを使用する前に使用する必要があります。2Cコネクタを取り外す必要があります。

少しの努力で、ほとんどのセンサーをに接続できます ftドゥイーノ 間違い。図6.30は、オンラインで安価に入手できる一般的なMPU6050センサーボードのケーブル接続の例を示しています。したがって、センサーは直接接続されますftドゥイーノ 接続可能。



（a）配線図 （b）完成したセンサー

図6.30：接続ケーブル付きのMPU6050センサー ftドゥイーノ

独自のプロジェクトでそれぞれのセンサーを使用するには、通常、追加のコードルーチンまたはライブラリが必要です。Arduinoプラットフォームが広く使用されているということは、ほとんど検索することなく、ほぼすべての一般的なセンサーに適した例とコードライブラリを見つけることができることを意味します。12日。

12日センサーライブラリの大規模なコレクションは、以下にあります。 https://github.com/ControlEverythingCommunity。

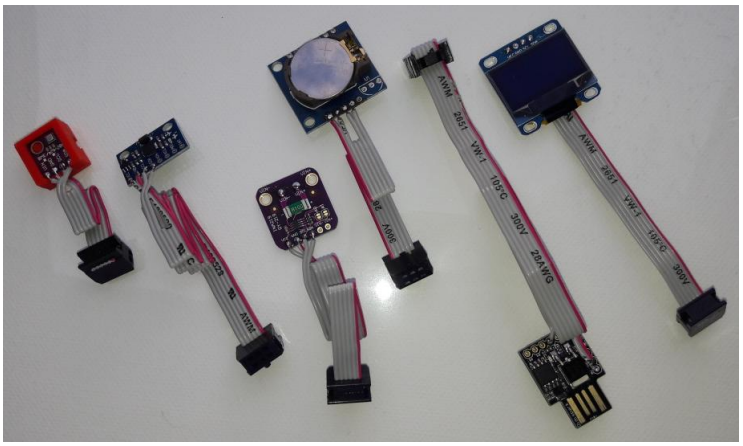


図6.31：さまざまなI2に適切な接続ケーブルを備えたCセンサー ftドゥイーノ

6.13.1スケッチ I2C / I2cScanner

センサーへの電気的接続が正しいかどうかをすばやくテストするには、通常、Iの簡単なテストで十分です。2C通信オフ。下 ファイル例 。FtduinoSimple 。I2C 。I2cScanner シンプルなものです私。2私から始まったCテストプログラム2C-Busは接続されたセンサーを検索し、それらのアドレスを出力します。センサーのそれぞれのアドレスは、通常、センサーの製造元によって永続的に割り当てられます。MPU-6050の場合、これはアドレスです0x68。 このアドレスは、センサーが正しく接続されている場合に表示されます。

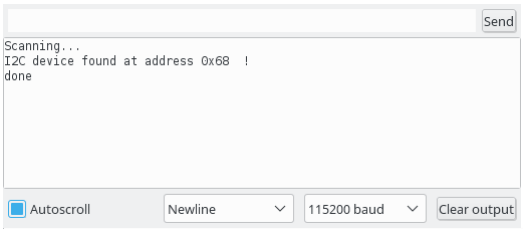


図6.32：の出力 I2cScanner MPU-6050を接続した状態

6.13.2MPU-6050センサー

MPU6050の場合、 ftドゥイーノ-あなた自身の例のある環境。以下のスケッチ例 ファイル例 。FtduinoSimple 。I2C 。MPU6050テスト MPU-6050から加速度値を読み取り、それらをに出力しますシリアルモニターがオフです。

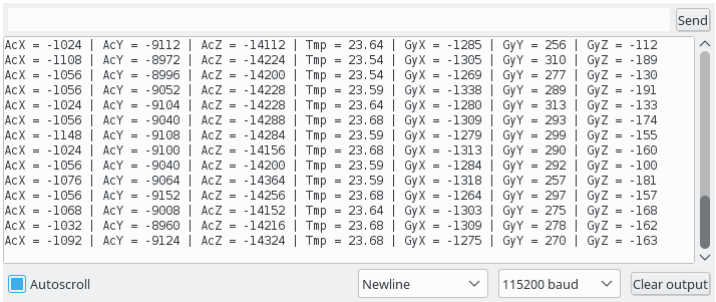
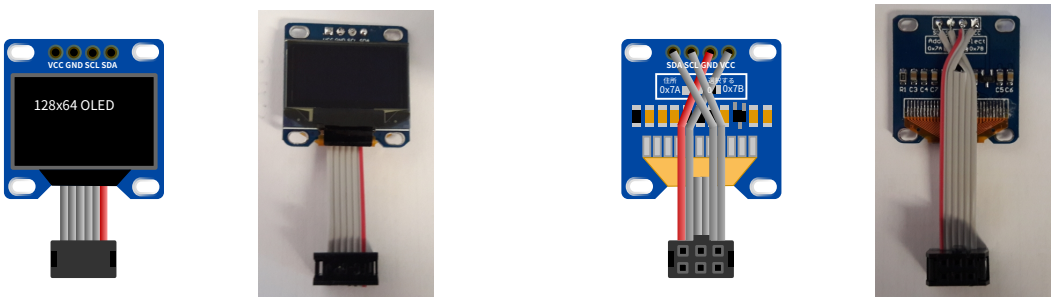


図6.33：の出力 MPU6050テスト

6.13.3OLEDディスプレイ

Iのもう1つの明らかなアプリケーション。2C接続は、小さなディスプレイの接続であり、たとえば、 ftドゥイーノ 測定値を出力できます。

わずかなお金でオンライン小売りに0.96インチの画面対角線を持つOLEDディスプレイがあります。サイズは3弱*3cm2 これらのディスプレイは、せん断技術と互換性のある対応するハウジングへの設置にも非常に適しています 13日。



（a）フロント （b）戻る

（c）住宅内

図6.34：接続ケーブル付きのOLEDディスプレイ ftドゥイーノ

ケーブルをはんだ付けするときは、他の点では同一のディスプレイ間でピン割り当てに違いがあるため、ここでのスケッチではなく、ディスプレイボード上のインプリントを使用することが重要です。

このディスプレイは、SSD1306をディスプレイコントローラーコンポーネントとして使用します14日 SolomonSystechによる。このタイプのディスプレイはArduino環境で非常に人気があり、適切なライブラリがインターネットで入手できます。1516。

重要： 他の多くの私と同じように。2OLEDディスプレイもCセンサーではありません ftドゥイーノ-固有ですが、他のArduinoプロジェクトでも使用されます。したがって、そのサポートはの一部ではありませんftドゥイーノ-インストール。ただし、上記のAdafruitライブラリは個別にインストールする必要があります。それ以外の場合、スケッチの翻訳はアートのメッセージとともに送信されます致命的なエラー：Adafruit_GFX.h：そのようなファイルまたはディレクトリはありません または同様のキャンセル。

Adafruit_SSD1306ライブラリは、表示テストに対応しています ファイル例 。Adafruit SSD1306 。ssd1306_128x64_i2c 例。

注：このマニュアルの以前のバージョンでは、ライブラリへの手動変更がここで説明されていました。これは、128x64ディスプレイに必要でした。Adafruit SSD1306ライブラリの現在のバージョンでは、この調整は不要になりました。例もそれに応じて適合されています。

スケッチ自体では、私は2のCアドレス 0x3D 後 0x3C 調整する：

13日https://www.thingiverse.com/thing:2542260
14日SSD1306のデータシート： https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf
15日Adafruit SSD1306ライブラリ： https://github.com/adafruit/Adafruit_SSD1306
16Adafruit GFXライブラリ： https://github.com/adafruit/Adafruit-GFX-Library

```
60      //デフォルトでは、3.3vラインから内部で高電圧を生成します！（きちんとした！） 画面。始める（（
61      SSD1306_SWITCHCAPVCC、0x3C）；                    // I2Cアドレス0x3Dで初期化します（
            128 x64）
62      //初期化が完了しました
```

the ftドゥイーノ-インストール自体も、この表示を使用する例をもたらします。Shootduinoゲームは以下にあります ファイル例 。FtduinoSimple 。Shootduino 。ゲームは入り口に3つのボタンを期待していますI1、I2 と I3 宇宙船とおそらくランプを制御するために O1。

6.13.4 VL53L0XLIDAR距離センサー

Fischertechnikは、距離測定用の超音波センサーを提供しています。これは、セクション1.2.6に示すように、 ftドゥイーノ 操作することができます。この超音波センサーは超音波パルスを送信し、音波が障害物に到達してセンサーに反射するまでの通過時間を測定します。距離は、通過時間と既知の音速から決定できます。

Arduinoの世界では、VL53L0Xレーザー距離センサーの形で興味深い代替手段があります。機能原理は超音波センサーと同じですが、音の代わりにレーザー光を使用しています。VL53L0XはI経由で簡単にアクセスできます。2Cと ftドゥイーノ 仲間、同僚。

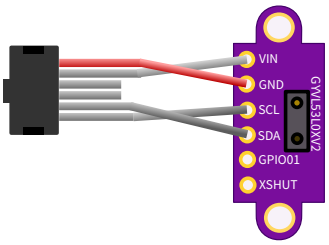


図6.35：VL53L0Xの ftドゥイーノ

セルフプリントに適したハウジングは、 ftドゥイーノリポジトリ17日。

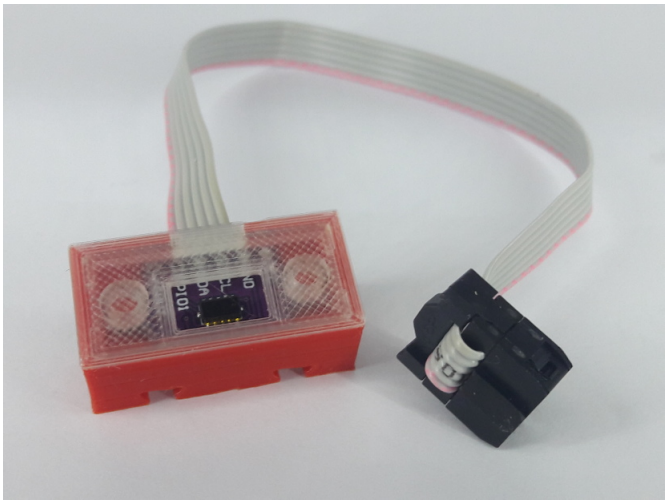


図6.36：3Dプリントされたハウジング内のVL53L0X

ほぼすべての通常の私は。2Cセンサーは、VL53L0X用の既製のArduinoライブラリとスケッチを使用してインターネット上で見つけることもできます18日。

17日VL53L0Xハウジング： https://github.com/harbaum/ftduino/tree/master/addons/vl53l0x
18日VL53L0X用のAdafruitライブラリ： https://github.com/adafruit/Adafruit_VL53L0X

6.13.5 ftドゥイーノ 私のように。2Cクライアントと2つのカップリング ftドゥイーノs

the ftドゥイーノ Iを介して他のデバイスを使用できるだけではありません。2Cバスをアドレス指定します。別のデバイスでアドレス指定するために、バス上にパッシブデバイスとして出力することもできます。

このオプションを使用する最も簡単な方法は、2つの場合です。 ftドゥイーノsIの真上。2結合するC。

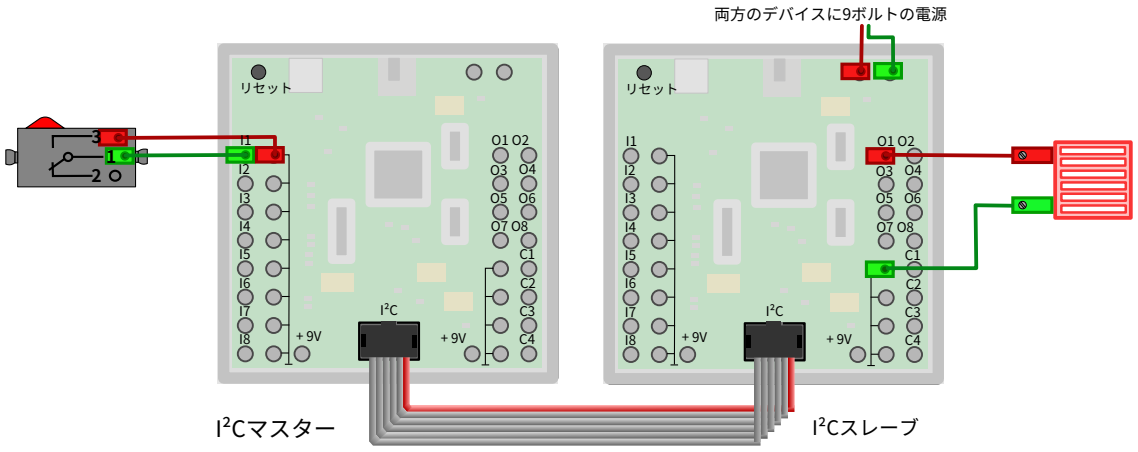


図6.37：2つの結合 ftドゥイーノs私について2C。

2つのIの間には1：1の接続があります。2関連するコントローラーのC接続。この構造では、1つのコントローラーをマスターとして、もう1つをスレーブとして構成する必要があります。対応するスケッチ例は以下にありますファイル 。例 。FtduinoSimple 。I2C 。I2cMaster と ファイル例 。FtduinoSimple 。I2C 。I2cSlave 。

マスターは継続的に入力を求めます I1 ボタンを接続し、Iを介してボタンのステータスを送信します。2スレーブとして構成された2番目のC ftドゥイーノ。次に、ランプを出力に切り替えますO1 それに応じてオンまたはオフにします。

マスターの電源供給は、Iを介して行うことができます2C接続。出力でランプを制御できるようにするには、スレーブにのみ9ボルトを直接供給する必要があります。私を介した供給2Cは、セクション1.2.5で説明されている既知の制限付きのUSB経由の電源に対応します。

高度 I2cSlave

簡単な例に加えて ファイル例 。FtduinoSimple 。 I2C 。 I2cSlave、機能を減らすことについて th FtduinoSimple-ライブラリのビルドは下にあります ファイル 。 例 。 Ftduino 。I2C 。I2cSlave 1つの入力および出力機能のほとんどをカバーする本格的なライブラリ構築の例 ftドゥイーノ 私について。2Cが公開します。

この拡張スケッチは、複雑なモデルの基礎として適しています。ここに示す例では、3つを使用していますftドゥイーノsマスターを拡張するftドゥイーノ さらに24の出力と36の入力によって。

最初 ftドゥイーノ 左端の写真では、マスターが形成され、その後に3つのスレーブが続きます。変更されていない例は、最初のスレーブで実行されます ファイル例 。Ftduino 。I2C 。I2cSlave 。他の奴隷のために、私は2のCアドレス スケッチはカスタマイズできます。これは、16行目をI2cSlave-他の2つのスレーブのアドレス43が44または45に置き換えられていることをスケッチします。

```
13日 空所 設定 () {
14日   pinMode ( (LED_BUILTIN 、 出力) ; // LEDを初期化します
15日
16     ワイヤー。始める (43) ;                    //アドレス＃43で「スレーブ」としてI2Cバスに参加します
17日   ワイヤー。onReceive ( (receiveEvent) ; //書き込みイベントに登録します ワイヤー。要求に応じて
18日    ( (requestEvent) ; //読み取りイベントに登録します
19日
20日   ftduino。初期化 () ;
21
```

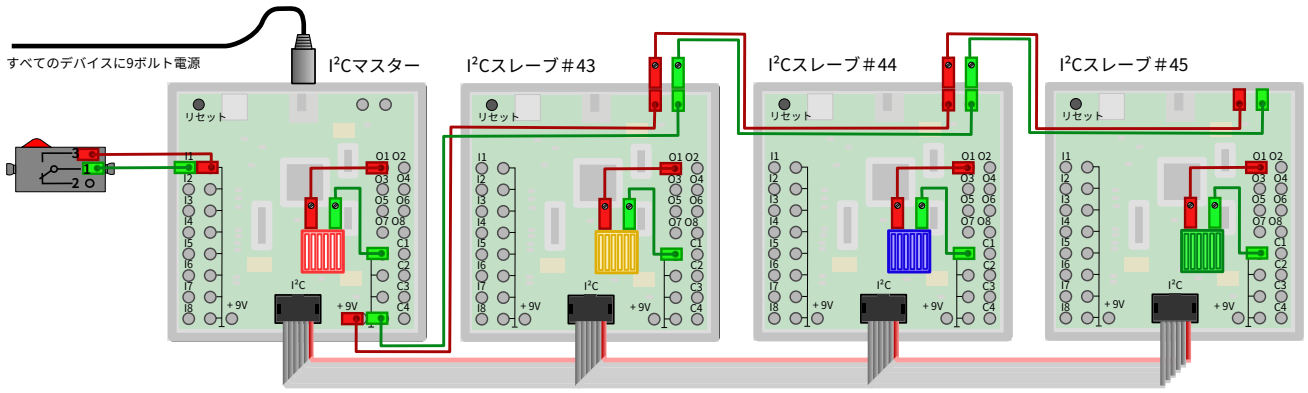


図6.38：4つのカップリング ftドゥイーノs私について2C。

```
22日    //すべての出力は高インピーダンスです
23      memset（（output_mode、0、 のサイズ（（output_mode））；
24   }
```

この場合、マスターには実際のプログラムロジック全体が含まれ、スレーブはIを介して基本的な制御コマンドを受け取ります。2C反対。マスタースケッチは以下にあります ファイル例 。Ftduino 。I2C 。I2cMaster 。12行目3つのスレーブのアドレス43、44、および45はすでにそこに設定されています。より多くのまたはより少ないスレーブを使用する場合は、それに応じて12行目を調整する必要があります。

```
 9   // 0（マスター自体）で始まり、//終了マーカーとして-1で始まる、制御されるI2cクライアントのリスト。この場合、3つ
10   のクライアントがアドレス// 43、44、および45で接続されています
11日
12日 静的定数 int8_tクライアント [] = {0、43、44、45、-1};
```

4つすべての電源 ftドゥイーノ■マスターに接続されているschertechnik電源ユニットから作成できます。次に、スレーブは2極Schertechnikケーブルを介してマスターの9ボルト出力から供給されます。

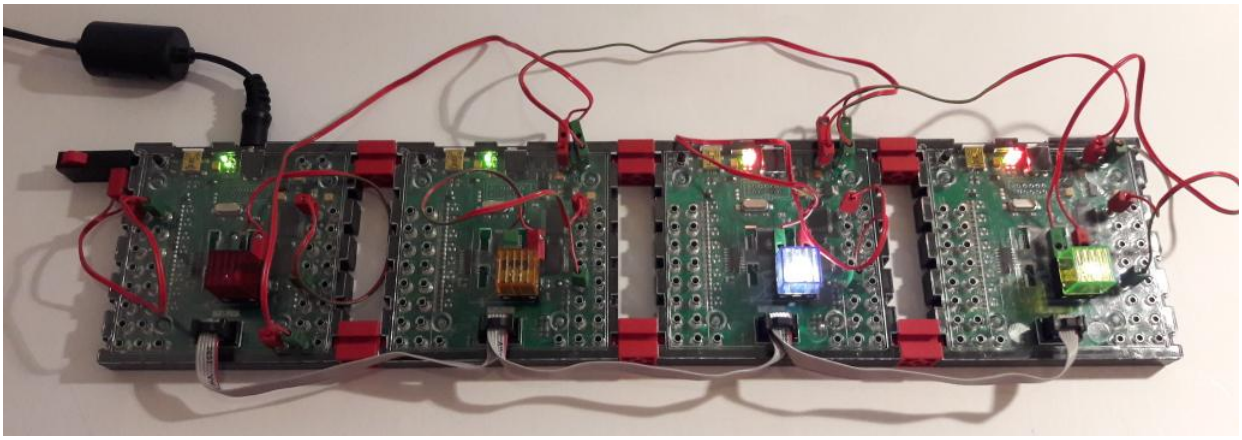


図6.39：4 ftドゥイーノs私について2C結合

例 ファイル例 。Ftduino 。I2C 。I2cMaster 複雑なモデルのテンプレートとして適しています。応じてただし、スレーブを変更および拡張する方がより実用的である可能性があります。ほぼ独立したサブユニット（組立ラインなど）を備えたモデルのデバイスは、個々のステーションを1つから簡単に分離できます。ftドゥイーノ 主に自律的な制御と私を持つために2C通信を最小限に抑えるため。たとえば、ステーションに新しいコンポーネントが到着したことをアナウンスするため。

次の表にIを示します2Cは日付として登録します I2cSlave.ino のために ftドゥイーノ 実装されました。前に付けられた値0バツ16進表記で表示されます。

出力O1からO8またはM1からM4のレジスタ割り当て

登録	説明
0ﾊﾞｲﾄ00	出力モードO1 / M1 0ﾊﾞｲﾄ0ﾊﾞｲﾄ- 単一出力としての操作 0ﾊﾞｲﾄ00- シングル出力オープン/高抵抗（トライステート） 0ﾊﾞｲﾄ01- + 9V（高）に切り替えられた単一出力 0ﾊﾞｲﾄ02- 単一出力がグランドに切り替えられました（低） 0ﾊﾞｲﾄ1ﾊﾞｲﾄ- モーター出力M1へのO2と結合された出力 0ﾊﾞｲﾄ10- ブレーキなしのモーター出力（o） 0ﾊﾞｲﾄ11- モーター出力がブレーキオフ（ブレーキ） 0ﾊﾞｲﾄ12- モーター出力が左に曲がる 0ﾊﾞｲﾄ13- モーター出力が右に曲がる
0ﾊﾞｲﾄ01	出力値（PWM）O1 / M1 0（オフ）から255（100%オン）このレジスタが書き込まれると、ハードウェア出力のステータスが更新されます。
0ﾊﾞｲﾄ02	出力モードO2 出力モードO1 / M1（レジスタ 0ﾊﾞｲﾄ00） 値 0ﾊﾞｲﾄ1ﾊﾞｲﾄ含む 0ﾊﾞｲﾄ00- シングル出力オープン/高抵抗（トライステート） 0ﾊﾞｲﾄ01- + 9V（高）に切り替えられた単一出力 0ﾊﾞｲﾄ02- 単一出力がグランドに切り替えられました（低）
0ﾊﾞｲﾄ03	0（オフ）から255（100%オン）までの出力値（PWM）O2 出力モードO1 / M1（レジスタ 0ﾊﾞｲﾄ00） 値 0ﾊﾞｲﾄ1ﾊﾞｲﾄ含む このレジスタが書き込まれると、ハードウェア出力のステータスが更新されます。
0ﾊﾞｲﾄ04	出力モードO3 / M2、出力モードO1（レジスタ 0ﾊﾞｲﾄ00）
0ﾊﾞｲﾄ05	出力値O3 / M2、出力値O1（レジスタ 0ﾊﾞｲﾄ01）
0ﾊﾞｲﾄ06	出力モードO4、出力モードO2（レジスタ 0ﾊﾞｲﾄ02）
0ﾊﾞｲﾄ07	出力値O4、出力値O2（レジスタ 0ﾊﾞｲﾄ03）
0ﾊﾞｲﾄ08	出力モードO5 / M3、出力モードO1（レジスタ 0ﾊﾞｲﾄ00）
0ﾊﾞｲﾄ09	出力値O5 / M3、出力値O1（レジスタ 0ﾊﾞｲﾄ01）
0ﾊﾞｲﾄ0*a*	出力モードO6、出力モードO2（レジスタ 0ﾊﾞｲﾄ02）
0ﾊﾞｲﾄ0*b*	出力値O6、出力値O2（レジスタ 0ﾊﾞｲﾄ03）
0ﾊﾞｲﾄ0*c*	出力モードO7 / M4、出力モードO1（レジスタ 0ﾊﾞｲﾄ00）
0ﾊﾞｲﾄ0*d*	出力値O7 / M4、出力値O1（レジスタ 0ﾊﾞｲﾄ01）
0ﾊﾞｲﾄ0*e*	出力モードO8、出力モードO2（レジスタ 0ﾊﾞｲﾄ02）
0ﾊﾞｲﾄ0*f*	出力値O8、出力値O2（レジスタ 0ﾊﾞｲﾄ03）

入力I1からI8の割り当てを登録します

一般的なI2C転送では、入力の最大2バイトを読み取ることができます。各入力は個別に読み取る必要があります。

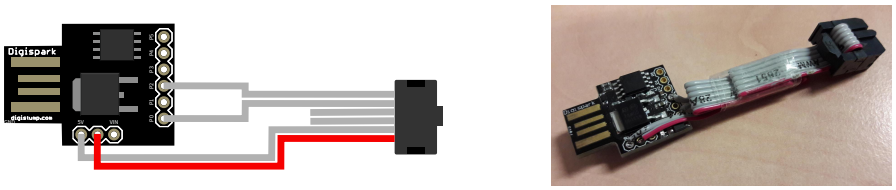
登録	説明
0バツ10	書き込み：入力モードI1 0バツ00- 電圧 0バツ01- 抵抗 0バツ02- カウンター 読み取り：入力値I1、下位バイト（LSB）
0バツ11日	読み取り：入力値I1、上位バイト（MSB）
0バツ12日	入力モード/入力値I2、入力モードI1（レジスタ 0バツ10）
0バツ13日	入力値I2、入力値I1（レジスタ 0バツ11）
0バツ14日	入力モード/入力値I3、入力モードI1（レジスタ 0バツ10）
0バツ15日	入力値I3、入力値I1（レジスタ 0バツ11）
0バツ16	入力モード/入力値I4、入力モードI1（レジスタ 0バツ10）
0バツ17日	入力値I4、入力値I1（レジスタ 0バツ11）
0バツ18日	入力モード/入力値I5、入力モードI1（レジスタ 0バツ10）
0バツ19日	入力値I5、入力値I1（レジスタ 0バツ11）
0バツ1*a*	入力モード/入力値I6、入力モードI1（レジスタ 0バツ10）
0バツ1*b*	入力値I6、入力値I1（レジスタ 0バツ11）
0バツ1*c*	入力モード/入力値I7、入力モードI1（レジスタ 0バツ10）
0バツ1*d*	入力値I7、入力値I1（レジスタ 0バツ11）
0バツ1*e*	入力モード/入力値I8、入力モードI1（レジスタ 0バツ10）
0バツ1*f*	入力値I8、入力値I1（レジスタ 0バツ11）

カウンタ入力C1～C4のレジスタ割り当て

登録	説明
0バツ20日	書き込み：カウンターモードC1 0バツ00- アウト 0バツ01- 立ち上がりエッジ 0バツ02- 立ち下がりエッジ 0バツ03- 両方の側面 0バツ04- 超音波センサー読み取りをアクティブにします：入力ステータスC1
0バツ21	書き込み：カウンターC1 0バツ00- カウンターを変更しないでください。それ以外の場合は、カウンターをクリアしてください 読み取り：カウンター読み取りC1 /超音波距離、下位バイト（LSB）
0バツ22日	カウンター読み取りC1 /超音波距離、上位バイト（MSB）
0バツ24	書き込み：カウンターモードC2 0バツ00- アウト 0バツ01- 立ち上がりエッジ 0バツ02- 立ち下がりエッジ 0バツ03- 両端を読み取る：入力ステータスC2
0バツ25日	書き込み：カウンターC2 0バツ00- カウンターを変更しないでください。それ以外の場合は、カウンターをクリアしてください 読み取り：カウンター読み取りC2、下位バイト（LSB）
0バツ26日	カウンタ読み取りC2、上位バイト（MSB）
0バツ28	カウンターモードC3、カウンターモードC2（レジスタ 0バツ24）
0バツ29	カウンタ読み取りC3、下位バイト（LSB）、カウンタ読み取りC2（レジスタ 0バツ25）
0バツ2*a*	カウンタ読み取りC3、上位バイト（MSB）、カウンタ読み取りC2（レジスタ 0バツ26）
0バツ2*c*	カウンターモードC4、カウンターモードC2（レジスタ 0バツ24）
0バツ2*d*	カウンタ読み取りC4、下位バイト（LSB）、カウンタ読み取りC2（レジスタ 0バツ25）
0バツ2*e*	カウンタ読み取りC4、上位バイト（MSB）、カウンタ読み取りC2（レジスタ 0バツ26）

ftドゥイーノ 私のように。2PC上のCスレーブ

もちろんできます ftドゥイーノ 私のように。2他人だけでなくC奴隷 ftドゥイーノsだけでなく、適切なIが装備されている場合はPCやその他のデバイスでも。2Cインターフェースを搭載。PCの場合、必要なI2USB経由のシンプルなアダプターを使用してCインターフェースを改造します。そのようなアダプターの1つはi2c_tiny_usb19日。



（a）配線図 （b）完成したアダプター

図6.40：Digispark /i2c_tiny_usb Iへの接続用。2C des ftドゥイーノ

LinuxPCの場合20日 プログラムはできますか i2c_detect 利用される。パラメータ付き-l あなたは最初にすべての私のリストを得ることができます2Cバスを出力します。

```
$ i2cdetect -l
i2c-3   わからない      i915 gmbus dpc                  N / A
i2c-1   わからない      i915 gmbus vga                  N / A
i2c-8   i2c             em2860＃0                       I2Cアダプター
i2c-6   わからない      DPDDC-B                         N / A
i2c-4   わからない      i915 gmbus dpb                  N / A
i2c-2   わからない      i915gmbusパネル                 N / A
i2c-0   わからない      i915 gmbus ssc                  N / A
i2c-9   i2c             バス001デバイス023DPDDC-Cのi2c-tiny-  I2Cアダプター
i2c-7   わからない      usb                             N / A
i2c-5   わからない      i915 gmbus dpd                  N / A
```

Digispark /i2c_tiny_usb この場合、次のように表示されます i2c-9。 私は2のCバス i2c_tiny_usb デバイスを検索します。

```
$ i2cdetect -y 9
     0 1 2 3 4 5 6 7 8 9 abcdef
00：                         --
10：---------------- ---- ---- ---- 40：---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
---- ---- ---- ---- 70：---- ---- ---- ---- ----
```

この場合、 FtduinoSimple-ライブラリベースのシンプル I2cSlave の ftドゥイーノ 認識された。

リポジトリ内21 アクセスに使用できるPythonの例があります ftドゥイーノ アクセスすることができます。より広範なについてはFtduinoライブラリベースのバリアント。リポジトリにはPythonの例もあります22日。

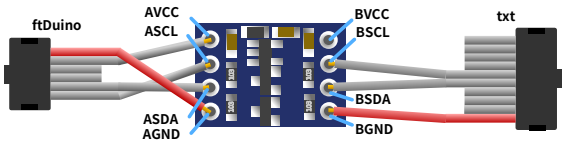
ftドゥイーノ 私のように。2TXTのCスレーブ

セクション6.13で述べたように、私は3.3ボルトで動作しました2schertechnik TXTコントローラーのC接続は、5ボルトで動作するIと電気的に互換性がありません2のCポート ftドゥイーノ。

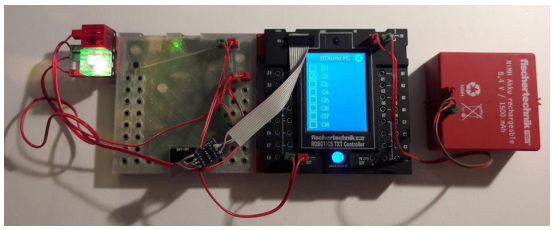
19日に関する詳細情報 i2c_tiny_usb 下にあります https://github.com/harbaum/I2C-Tiny-USB
20日また、ラズベリーパイまたは schertechnikTXTはLinuxPCです
21 https://harbaum.github.io/ftduino/ftduino/libraries/FtduinoSimple/examples/I2C/I2cSlave/master.py
22日 https://harbaum.github.io/ftduino/ftduino/libraries/Ftduino/examples/I2C/I2cSlave/master.py

シンプルなレベルシフター

ただし、適切なレベルコンバータを使用すると、必要な信号調整を簡単に行うことができます。このための電子機器は、オンラインショップで安価に入手できます。TXT自体は3.3ボルトを供給しないため、電子機器が5ボルト側の電源から3.3ボルト側の電圧供給を生成することを確認する必要があります。



（a）ケーブル図

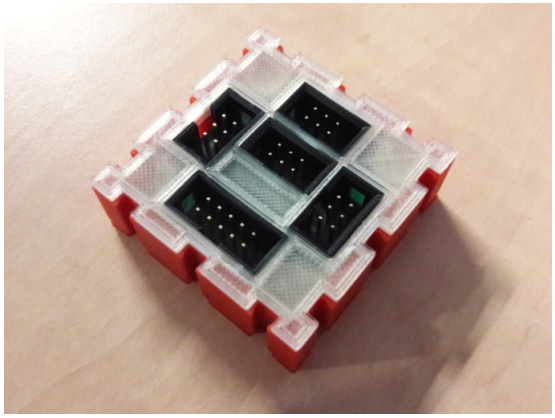


（b）TXTへの接続

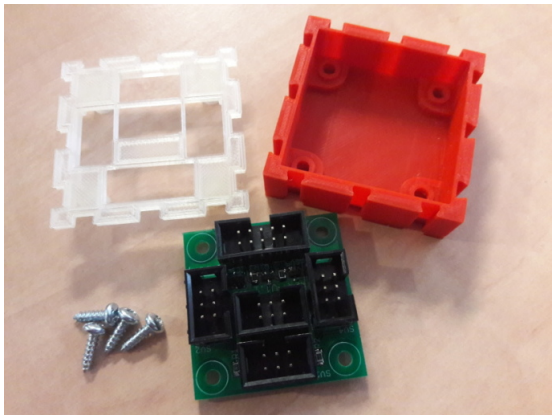
図6.41：TXTとを接続するためのレベルシフター ftドゥイーノ

6.13.6 ftドゥイーノ-私2Cエキスパンダー

いわゆるIはレベルシフターの機能も果たします2Cエキスパンダー23。このデバイスはで使用されましたftドゥイーノ 設計されていますが、TXTまたはTXで操作することもできます。



（a）完成したデバイス



（b）個々の部品

図6.42：I2のためのCエキスパンダー ftドゥイーノ

私。2C-Expanderは、TXT互換の10ピンIを提供します。2Cコネクタ対応と4つの6ピンTXそれぞれ ftドゥイーノ-互換性。4つのftドゥイーノ-互換性のあるポートは1対1で接続され、複数のIを接続するために使用できます。2Cデバイスから ftドゥイーノ 利用される。TXTまたはそのセンサーへの接続を可能にするレベルシフターも含まれています。レベルシフターには、ftドゥイーノ。

schertechnik TXTコントローラーのコミュニティファームウェアに適したアプリは、cfwアプリリポジトリにあります。24。

私以来2TXT上のCデバイスは、RoboProおよび元のrmwareである ftドゥイーノ このように、TXTの拡張としてRoboProでも使用できます。

6.13.7schertechnik方向センサー

を接続するためのケーブル ftドゥイーノ TXTとIに2Cエキスパンダーは固定方向に固定されていません。したがって、TXT用に設計されたセンサーをftドゥイーノ 接続する。

23the ftドゥイーノ-私2Cエキスパンダー： https://github.com/harbaum/ftduino/tree/master/addons/i2c-expander
24 https://github.com/harbaum/cfw-apps/tree/master/packages/ftDuinoI2C

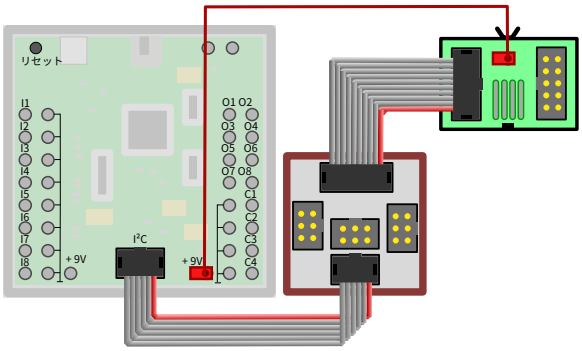
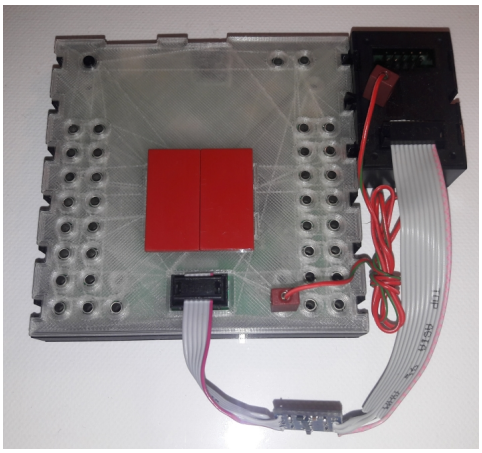
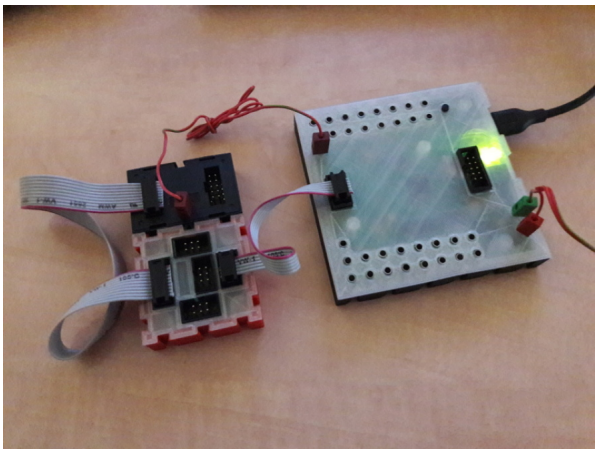


図6.43：Iを介したschertechnikセンサーの接続2Cエキスパンダー

これは、たとえば、組み合わせセンサー158402でテストされました。25日 3-in-1方向センサー 26日 既製のArduinoスケッチもあるBoschBMX055に基づいています27。9ボルトの供給電圧は、ftドゥイーノ。



(a) レベルシフター



(b) 1日2Cエキスパンダー

図6.44：からの方向センサー　　せん断技術 ftドゥイーノ

6.13.8schertechnik環境センサー

ROBOTICSTXTスマートホームキット544624の環境センサー28 BME680に基づく29 ボッシュセンサーテックから。温度、気圧、湿度、空気の質のセンサーが含まれています。

図6.45に示すように、Arduino環境にはこのセンサー用のさまざまなライブラリもあり、ArduinoIDEのライブラリマネージャーを使用して簡単にインストールできます。Adafruitライブラリは最初の実験を歓迎します。

私は2BME680のCアドレスは、適切な配線を使用して調整できます。Fischertechnikがアドレスを設定します0x76 しばらくの間、Adafruitライブラリにはデフォルトでアドレスの下にセンサーがあります 0x77 期待される。下の例では ファイル例 。AdafruitBME680ライブラリ したがって、schertechnikが使用するアドレスは、いずれの場合も明示的である必要があります 指定できます。関数を呼び出すbme.begin（） 例では次のように変更する必要があります。

```
もしも （！bme。始める （0x76））{
  シリアル。println （（「有効なBME680センサーが見つかりませんでした。配線を確認してください！」）；その
  間 （1）；
```

25日 schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=158402
26日https://content.ug scher.com/cb les / scher / Zulassungen / ft / 158402-Kombisensor-Kurzanleitung-BMX055-2017-06-09.pdf
27https://github.com/ControlEverythingCommunity/BMX055
28 schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=544624
29ボッシュBME680： https://www.bosch-sensortec.com/bst/products/all_products/bme680

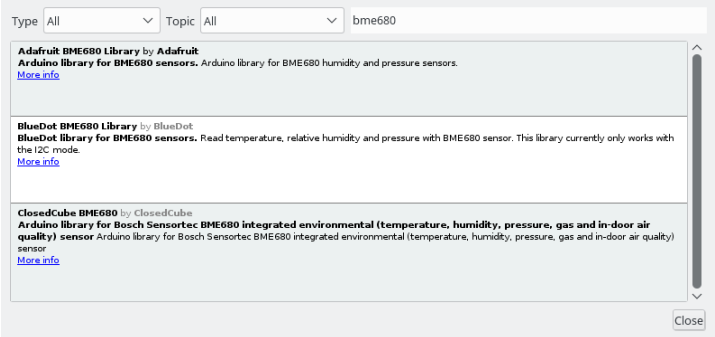
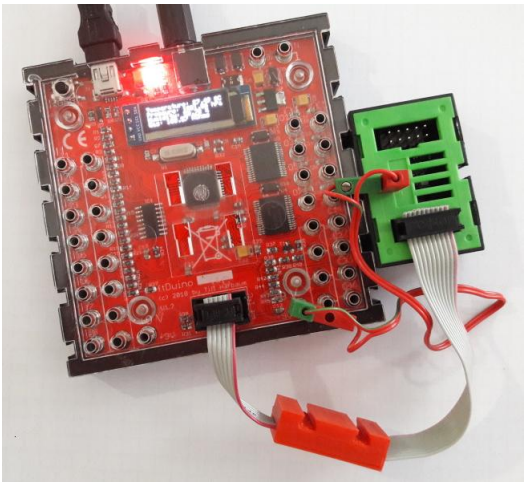


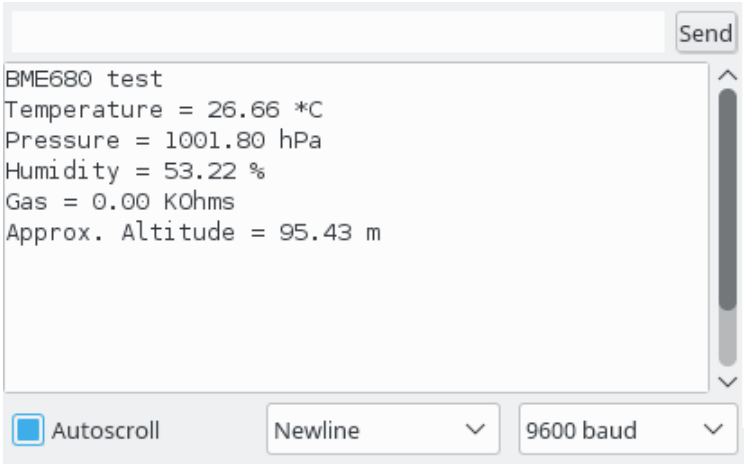
図6.45：BME680のArduinoライブラリ

}

センサーはレベルシフターとIで調整できます。2Cエキスパンダーとftエクステンダーを操作します（セクション8.5を参照）。



（a）レベルシフターを介した接続



（b）の発行 ファイル例 。AdafruitBME680ライブラリ 。bme680test

図6.46： schertechnik環境センサー ftドゥイーノ

BME680は、加熱と抵抗の測定を通じて空気の質を評価します。Adafruitライブラリは、ここではあまり意味のない抵抗値のみを提供します。ボッシュ自体は、値をACQ品質値に変換するためのクローズドソースライブラリを提供しています。これは、schertechnik自体が構築キットで使用しています。このライブラリはArduinoでも利用できますが、ライブラリが大きすぎてフラッシュメモリに対応できませんftドゥイーノ。

抽象空気質値の代替計算 airq このように見えます。

```
gas_baseline      =  200000.0;
hum_baseline      =  40.0;
hum_weighting     =  0.25;
gas_offset        = gas_baseline - gas_resistance; =湿度 -
hum_offset        hum_baseline;
hum_score         = (100- hum_baseline - hum_offset) /
                      (100- hum_baseline) * (hum_weighting * 100) ;
gas_score         = (gas_resistance / gas_baseline) * (100- (hum_weighting * 100) ) ; =hum_score + gas_score
airq              ;
```

6.13.9 Mini-I2Cサーボアダプター

特に ftドゥイーノ Mini-Iになりました2設計されたCサーボアダプタ。サイズは小さいですが、電源とサーボ制御を1つのハウジングにまとめています。

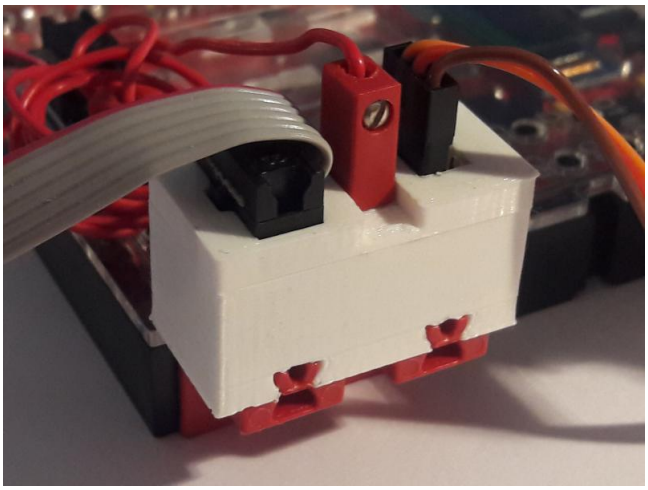


図6.47：Mini-I2Cアダプターオン ftドゥイーノ

接続と試運転

最初の起動では、Iのみ2間のC接続 ftドゥイーノ とミニI2Cアダプターが作られています。一般的に、を使用することをお勧めしますftドゥイーノ 電圧とUSB電源から切断し、2C接続が確立され、切断されます。接続が確立されるとすぐに、ftドゥイーノ スイッチを入れます。Mini-Iの発光ダイオード2Cアダプターは、約1秒間点灯してから、消灯します。

に ftドゥイーノ 最初のテストに使用する必要があります ファイル例 。FtduinoSimple 。I2C 。I2cScanner -スケッチイン-インストールされます。最初のテストでは、9ボルトもサーボ接続も必要ありません。I2cScannerスケッチがアドレスの下にあるはずです0バツ11日 アダプターを見つけます。

I2Cスキャナー
スキャン..。
アドレス0x11でI2Cデバイスが見つかりまし ！
た

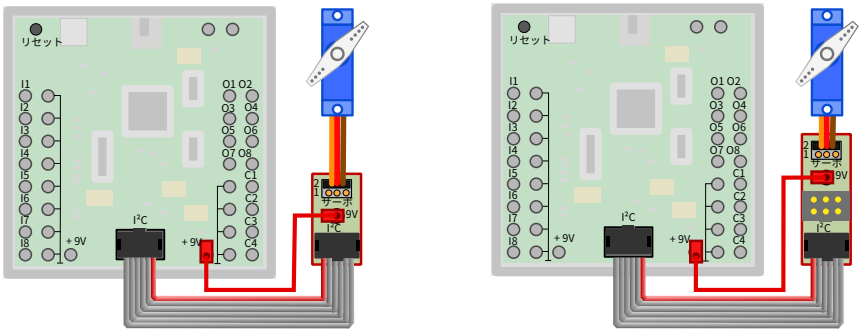
アダプターが ftドゥイーノ に対処すると、アダプタの発光ダイオードが短時間点灯するはずです。このテストが機能しない場合は、2Cケーブルを確認します（両方のコネクタが正しい向きになっていますか？）。他のデバイスはIで動作していますか。2のCポート ftドゥイーノ ？

このテストが成功した場合、図6.48に示すように、サーボと9ボルト電源を接続できます。ここでも、ftドゥイーノ 再配線中は電源とUSB電源から切断してください。

Mini-I2Iの間にCサーボアダプタを挿入します。2のC出力 ftドゥイーノ サーボの接続が接続されています。サーボに電力を供給するためにも9ボルトの接続が必要です。Mini-I2Cアダプターは
Schertechnik-通常9ボルトからサーボ互換の5ボルト。

注意！ 私は2のCポート ftドゥイーノ、Mini-Iの2Cサーボアダプタとサーボは過電圧から保護されておらず、9Vと接触してはなりません。を使用することをお勧めしますftドゥイーノケーブルの電源を切る。

すべての接続が確立されたら、下のスケッチ ファイル例 。FtduinoSimple 。I2C 。MiniServoAdapter 利用される。両方のサーボ出力を反対方向に連続的に制御します。



（a）個人I。2Cコネクタ （b）ダブルI。2Cコネクタ

図6.48：Mini-Iの接続2へのCアダプター ftドゥイーノ

プログラミング

Mini-Iのプログラミング2C-Servo-Adapterは非常にシンプルです。簡単な例を以下に示します。 ファイル 例 。FtduinoSimple 。I2C 。MiniServoAdapter 。出力を値に設定するには、通常、

次のコードをリッチにします。

```
ワイヤー。beginTransmission（（住所）；ワイヤー。
書きます （0x00）；ワイヤー。書きます （94）；ワイ      // 0 =サーボ1、1 =サーボ2
ヤー。endTransmission （）；                          //中央、63（左端）から125（右端）
```

レジスタ割り当て

Mini-Iの標準アドレス2Cサーボアダプターは16進です 0バツ11日 または10進数 17日 このアドレスは永続的に変更できます（以下を参照）。住所を忘れた場合は、彼がお手伝いします ファイル例 。FtduinoSimple 。I2C 。I2cScanner -スケッチしてアダプタのアドレスを決定します。

登録	説明
0バツ00/0バツ01	サーボ出力1または2のパルス幅を16刻みで設定しますμ標準設定を参照してください。63（1ms）から125（2ms）までのサーボに適した範囲の値のみが受け入れられます。この範囲外の値には上限があります。制限は変更できます（以下を参照）。サーボ1またはサーボ2のパルスモードの下限を16で設定するμsステップ。デフォルト値は63です。レジスタに入力されたこの制限より大きい値0バツ00 と 0バツ01 この下限に制限されています。
0バツ02/0バツ03	
0バツ04/0バツ05	サーボ1またはサーボ2のパルスモードの上限を16で設定するμsステップ。デフォルト値は125です。レジスタに入力されたこの制限よりも小さい値0バツ00 と 0バツ01 この上限に制限されています。
0バツ06/0バツ07	このレジスタを書き込むと、サーボ1または2のoセットが設定されます。デフォルト値は 0。oセットはレジスタによるものになります 0バツ00 また 0バツ01 パルスごとに追加 後 境界がチェックされました。このようにして、例えば、中心位置をサーボの個々の特性に適合させることができる。
0バツ08	私を入れて2Cアドレス。新しいアドレスは、次の範囲内にある必要があります0バツ03 それまで 0バツ77 を除いて嘘をつく 0バツ3*c* （（このアドレスは、内部OLEDディスプレイからより適切です ftドゥイーノs使用）。
0バツ09	の執筆 0*xa*5 レジスターで 0バツ09 転送はビアレジスタを保存します 0バツ02 それまで 0バツ08 EEPROMで永続的に行われる設定。この時点から、再起動後も電源が入っていなくても設定は保持されます。
0バツ00 読む	レジスタの読み取り 0バツ00 ファームウェアバージョンをBCDコーディングで返します。たとえば、バージョン1.0の場合は0x10
0バツ01+ 読む	他のすべてのレジスタは、読み取り時に供給します 0バツ5*a*

私の変化2Cデバイスアドレス

前のセクションで説明したように、I2Mini-IのCアドレス2Cサーボアダプタを交換してください。下 ファイル 例 。FtduinoSimple 。I2C 。MiniServoAddress 完成したスケッチもあります。

このスケッチがに適用される場合 ftドゥイーノ ロードすると、アドレスをインタラクティブに変更できます。通常、これは必要ありません。複数のMini-Iの場合のみ。2C-Servo-Adapterを同時に操作する場合は、各アダプタに個別のアドレスを指定する必要があります。これを行うには、調整するアダプターを最初に接続します。ftドゥイーノ 図6.49に示すように、接続してアドレスを変更しました。

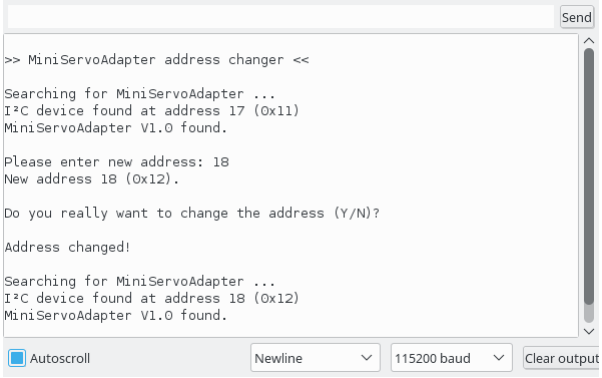


図6.49：Iの変更2Mini-IのCアドレス2Cアダプター

6.14WS2812Bフルカラー発光ダイオード

難易度： ★★★★★

でのWS2812B発光ダイオードの使用 ftドゥイーノ いくつかのはんだ付けと補助ライブラリのインストールが必要です。したがって、このプロジェクトは上級ユーザーにのみお勧めします。

私。2のCポート ftドゥイーノ 主にIを接続するために使用されます2Cデバイス。そこに接続されている信号のためSDA と SCL ただし、ATmega32u4マイクロコントローラーの自由に使用できる接続では、他のタイプの信号にも使用できます。そのような信号の1つは、WS2812B発光ダイオードで使用されるシリアル同期データストリームです。これらの発光ダイオードは、メーターによってさまざまなオンラインプロバイダーからわずかなお金で入手できます。

の内部電源を使用するには ftドゥイーノ 過負荷を回避するには、最大2つのWS2812B発光ダイオードをIの5ボルト電源に接続する必要があります。2のCポート ftドゥイーノ 動作します。より多くの発光ダイオードを使用する場合は、別の外部5ボルト電源を用意する必要があります。

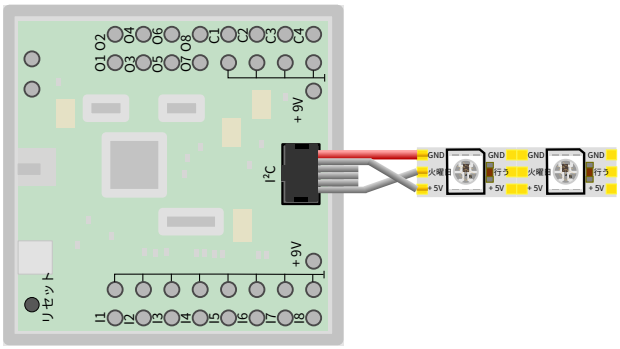


図6.50：2つのWS2812Bフルカラー LEDの接続

各WS2812B発光ダイオードストリップには、次の3つの入力信号があります。 グラウンド、+ 5V と 火曜日。 供給信号 寸法 と + 5V Iの対応するものに直接関連しています2Cポートが接続されています。シグナル火曜日 のデータを表し、

発光ダイオードのデータ信号入力。データ出力は、発光ダイオードストリップのもう一方の端にもあります （行う） 使用しないでください。彼はそれを渡します火曜日 受信信号は、追加の発光ダイオードに渡される場合があります。The DI-信号はオプションで使用できます SCL また SDAIのピン。2Cポート。対応する信号名は、後でスケッチに入力する必要があります。

LEDは注意して接続する必要があります。短絡または誤った接続は、LEDおよびftドゥイーノ 損傷する。

6.14.1スケッチ WS2812FX

WS2812B LEDと必要なコードライブラリを制御するための例は、たとえばWS2812BFXライブラリにあります。30日 削除することができます。WS2812B発光ダイオードを制御するための他のライブラリも使用できるはずです。

ライブラリのインストールには、ArduinoIDEの経験が必要です。ライブラリが正しくインストールされている場合は、 さまざまな例を以下に示します ファイル例 。WS2812FX 。例 auto_mode_cycle 適しています 発光ダイオードの機能をチェックします。

スケッチの最初に、使用する発光ダイオードの数と接続を調整するために、2つの小さな変更を加えるだけで済みます。

```
1   # 含む      <WS2812FX.h>
2
3   # 定義    LED_COUNT   2
4位 # 定義    LED_PIN    SCL
5
6日 # 定義    TIMER_MS   5000
```

6.15からの音楽 ftドゥイーノ

難易度： ★★☆☆☆

the ftドゥイーノ スピーカーが内蔵されていないため、補助なしでは音を出力できません。ただし、スピーカーを出力の1つに接続することは可能です。ftドゥイーノ 接続する。Schertechnikスピーカーカセット36936はもちろんこれに特に適しています。

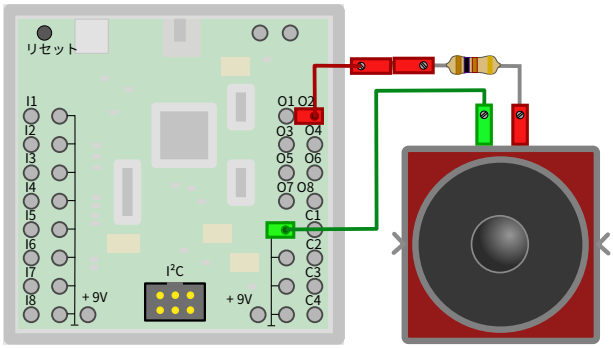


図6.51：スピーカーカセットの ftドゥイーノ

少なくともの直列抵抗が重要です 100 Ω スピーカーと ftドゥイーノ 切り替えられます。の9ボルトはftドゥイーノ スピーカーに直接置くと、スピーカーが非常に簡単に損傷し、生成される音が非常に大きくなる可能性があります。直列抵抗は最大電流を制限し、スピーカーと聴覚を保護します。

ここで、50%のPWM比で出力O2をオンにすると、PWM信号を直接聞くことができます。

30日https://github.com/kitesurfer1404/WS2812FX

```
ftduino。output_set（（Ftduino：：：O2、Ftduino：：：こんにちは、Ftduino：：：MAX / 2）；
```

50%PWM信号により、出力は永続的に こんにちは と オフ 切り替えられました（セクション6.3も参照）。出口のときだけこんにちは これは、スピーカーを介した出力からアース接続への電流です。のPWM周波数以来Ftduinoライブラリは約200ヘルツで、この周波数でトーンを聞くことができます。

セクション6.3で説明したように、PWM生成は、この目的のために提供されたATmega32u4マイクロコントローラーのPWM出力を介して生成されるのではなく、マイクロコントローラーがSPIバスを介して出力ドライバーに継続的に送信する信号を介して生成されます。この手順は非常に柔軟性があり、8つの出力すべてを独立した信号で制御できますが、マイクロコントローラーからの絶え間ない協力が必要であり、計算時間の一定の割合を永続的に消費します。PWM周波数が高いほど、マイクロコントローラーが信号を変更しなければならない頻度が高くなり、実際のスケッチでは利用できない計算時間の必要性が高くなります。200ヘルツでは、この影響はごくわずかです。ただし、音を発生させるには、キロヘルツの範囲の周波数が必要です。デジタルのオン/オフ方形波信号だけでなく、たとえばアナログ正弦波信号を生成する場合は、メガヘルツ範囲のPWM信号も必要です。ATmega32u4は、SPIバスを介してこれを行うことはできません。

MC33879出力ドライバにはそれぞれ2つの入力があります[31]SPIバスをバイパスして直接制御できます。の中にftドゥイーノ これらの入力のほとんどは接地されていますが、出力用のものです O2 責任あるインプット EN6 出力ドライバの U3 ピン付きです PB7 ATmega32u4の。これにより、出力ドライバの出力トランジスタの1つをATmega32u4のこのピンを介して直接切り替えることができます。Arduinoの世界では、このピンの番号は11で、通常のArduino機能と切り替えることができます。

```
//出力O2のハイサイドドライバーを1秒間アクティブにします pinMode （11、 出力）；
digitalWrite （11、
                    高い）；
遅れ （1000）；
digitalWrite （11、     低い）；
```

出口で効果を確認するには、 FtduinoSimple-まだ必要な出力ドライバーの初期化はライブラリによって行われるため、ライブラリをスケッチに含める必要があります。

6.15.1スケッチ 音楽

難易度： ★☆☆☆☆

または、このピンでサウンド生成用のArduinoコマンドを使用することもできます。下のスケッチ例 ファイル例 。FtduinoSimple 。音楽 これを使って。

```
//コンサートピッチAを1秒間再生します 調子（11、440、
1000）；
```

6.15.2スケッチ MusicPwm

難易度： ★☆☆☆☆

使用したピン PB7 はATmega32u4内部タイマー1の一部です。これは、ATmega32u4の特別なハードウェアを使用して信号を生成できることを意味します。これは、計算能力を使用せずに高周波信号を生成できることを意味します。

スケッチ例 ファイル例 。FtduinoSimple MusicPwmはそれを使用して、前のスケッチを作成します。コードはかなり不可解で、より複雑に見えます。Arduinoが調子 （） - 機能しますが、いわゆるタイマー割り込みのバックグラウンドでトーンを生成するための計算能力が必要です。この例のトーンは、ATmega32u4プロセッサのいわゆるタイマーハードウェアからのみ生成されます。プロセッサの実際のコンピューティング部分は、ピッチの変更またはサウンド出力の停止のみを担当します。

このような単純な音楽の例では、バックグラウンドの計算能力要件はごくわずかです。ただし、超音波範囲以上などの非常に高い周波数を生成する場合は、周波数が高くなるほど信号を頻繁に変更する必要があるため、バックグラウンドでの計算能力の必要性が高まります。タイマーハードウェアを使用しても、この問題は発生しません。音の生成に必要な信号の変化、つまり、

[31] EN5 と EN6、 ご参照ください http://cache.freescale.com/ les / analog / doc / data_sheet / MC33879.pdf

希望のトーン周波数での出力は自動的に行われるため、高周波数であってもマイクロコントローラーからの計算時間を必要としません。

6.16 ftドゥイーノ MIDI楽器として

スピーカーを接続することは、音を出す唯一の方法です。Schertechnikなどのモジュラーシステムは、たとえばダイナミックモジュラーシリーズのサウンドチューブの助けを借りて、電気機械的な方法で音を発するように自然に誘います。

その柔軟なUSBインターフェースで、ftドゥイーノ そのようなスキルを活用するエレガントな方法。いわゆるMIDIインターフェース[32] コンピュータと電子楽器を接続するように設計されています。電気的な非常に特殊なMIDI接続に加えて、USB経由のMIDIのバリエーションがあります。Arduino環境はMIDIUSBIDEのライブラリ管理を介して直接インストールできるライブラリ。

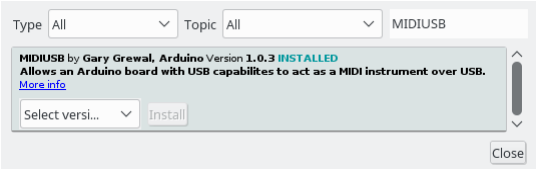


図6.52：ArduinoIDEへのMIDIUSBライブラリのインストール

6.16.1スケッチ MIDI楽器

the ftドゥイーノ-以下のサンプルスケッチ ファイル例 。FtduinoSimple 。MIDI楽器 このライブラリを使用して the ftドゥイーノ PC用のMIDIデバイスとして使用できるようにします。一般的なオペレーティングシステムには、対応するデバイス用のドライバーが既にあり、Windows、Linux、およびMacOSは、MIDIデバイスに構成されたデバイスを認識します。ftドゥイーノ USBオーディオデバイスとして追加のドライバをインストールする必要はありません。

the ftドゥイーノ いわゆるUSB複合デバイスです。これは、同時に複数のUSB機能を実装できることを意味します。MIDIの場合、これはPCにMIDIデバイスとして表示され、同時に USB経由で続行COM：-インターフェイス。これは、特にスケッチの開発中に非常に役立ちます。

それと1つ MIDI楽器提供されるスケッチ ftドゥイーノ たとえば、一般的なMIDIツールを備えたLinuxPCによって認識されます。

```
$ aplaymidi -l
 ポート    クライアント名                  ポート名
 14-0      ミディスルー                    ミディスルーポート-0
 24：0     ftDuino                         ftDuino MIDI 1

$ aplaymidi -p 24：
0demosong.mid。。。
```

the ftドゥイーノ 片方の歌の声になります O2 接続されたスピーカーとで再生 COM：-受信したコマンドに関する出力ポート情報は、電気機械機器の基礎として機能します。

この単純な例はモノフォニックであり、一度に1つのノートしか再生できないため、この単純なセットアップで十分に再生できるMIDIファイルはほとんどありません。ポリフォニックソングは再生できません。もちろん、この制限はポリフォニックメカニカルミュージックモデルで解除でき、ここで使用されている非常に単純なタイプのサウンド生成にのみ起因します。

サンプルモノフォニックファイル song.mid スケッチのディレクトリにあります。

[32]MIDI、楽器デジタルインターフェース、 https://en.wikipedia.org/wiki/MIDI

6.17 ftドゥイーノ Androidスマートフォンで

難易度： ★★☆☆☆

最新のAndroidスマートフォンおよびタブレットのほとんどは、 ftドゥイーノ 適切なケーブルを使用して直接接続します。これには、いわゆるUSB-On-the-Go（USB-OTG）ケーブルが必要です。のためにftドゥイーノ Lindy31717が適しています。両端に適切なプラグがあり、50cmでモデルに収まるほど短いです。

(a) スクリーンショット (b) スマートフォンをオンにするftドゥイーノ

図6.53：を使用するためのAndroidデモ ftドゥイーノ スマートフォンで

USB-OTGを使用すると、スマートフォンがPCの役割を果たします。対応するケーブルが接続されると、電圧がUSBポートで利用可能になり、アクティブないわゆるUSBホストとしてUSBバスの制御を引き継ぎます。の出力がftドゥイーノ 必須ではありません。このようにして、独自の電源で実行することもできます。 ftドゥイーノ 免除されます。theftドゥイーノ その後、携帯電話から電力が供給されます。

スマートフォンとの間のすべての通信 ftドゥイーノ シリアルUSB接続（COMポート）を介して実行され、 ftドゥイーノ -PCとの通常の通信と同じように扱われます。場合によっては、それが理にかなっているかもしれませんftドゥイーノ スマートフォンへのUSB接続が存在するかどうかがわかります。たとえば、スマートフォンが接続されている状態で手動操作が必要なときに、スマートフォンなしで自動機能を実行する場合などです。これは、USB電源に問い合わせることでスケッチで可能になります。

```
// USB電源のクエリを準備します（例：setup（）関数で） USBCON |=（1 <<OTGPADE）；


// USB電源を照会します もしも （（USBSTA &（1
<<VBUS））{
   // USBが接続されていま
   す//...。
}  それ以外 {{
   // USBが接続されていません//...。

}
```

完全な簡単なデモスケッチは以下にあります ファイル例 。FtduinoSimple 。USB。Androidデモ 。USB経由で1秒に1回メッセージを送信し（COUNTER：XX）、メッセージのオンまたはオフを予期して出力を切り替えます O1 によると。

の中に ftドゥイーノ対応するAndroidアプリはリポジトリにあります[33]。AndroidStudioで翻訳して、

[33] ftドゥイーノ-Androidアプリ： https://github.com/harbaum/ftduino/tree/master/android

あなた自身の開発のための基礎を使用してください。AndroidアプリはUsbSerialライブラリに基づいています[34] と以外の最小限の調整後 ftドゥイーノ また、ほとんどのArduinoを接続します。

それ以上の構成を必要としないため、他の方法で使用されるBluetoothまたはWLAN接続よりもはるかに簡単に使用できます。スマートフォンとftドゥイーノ connectedは対応するアプリを起動し、接続が完了します。

6.18 WebUSB： ftドゥイーノ Webブラウザによる制御

難易度： ★★☆☆☆

のようなUSBデバイスでの通常の方法 ftドゥイーノ アクセスは固定スキームに従います。PCにデバイスとの通信方法を指示する適切なドライバーがPCにインストールされます。Arduino IDEのようなPCアプリケーションは、このドライバーを使用してデバイスにアクセスします。

WebUSB[35] は、ドライバーとソフトウェアのインストールを完全に省くことにより、USBデバイスの使用を大幅に簡素化する試みです。代わりに、特別なWebサイトには、適切に適合されたUSBデバイスと直接通信するために必要なすべてのものがあります。USBデバイスは、制御WebサイトがどのURLで見つかるかをPCに通知できます。これにより、ユーザーにとって実際のプラグアンドプレイソリューションが実現します。これまで知られていなかったWebUSBデバイスを接続すると、ブラウザは適切なWebサイトを開き、ドライバやソフトウェアをインストールしなくてもデバイスをすぐに使用できます。

6.18.1Chromeブラウザ

WebUSBは公式のWeb標準ではないため、GoogleのChromeブラウザでのみ使用されます[36] サポートします。この実験では、ChromeをPCにインストールする必要はありません。いわゆるポータブルバージョンで十分です。[37] たとえば、USBスティックから開始します。セキュリティ上の理由から、Googleは、現在のバージョンでデバイスによって報告されたURLを自動的に開始するオプションを禁止しています。URLhttps://harbaum.github.io/ftduino/webusb/console したがって、ブラウザに手動で入力する必要があります。

Chromeブラウザは、Googleがanrdoidスマートフォンおよびタブレットで提供しているブラウザです。このバージョンは、WebUSB実験にも適しています。接続するにはftドゥイーノ スマートフォンには、USBホストケーブル（例：Lindy 31717、セクション6.17を参照）が必要です。

図6.54：ChromeブラウザはほとんどのAndroidデバイスにプリインストールされています

Chromiumブラウザは、LinuxPCでも同じ目的を果たします。通常、パッケージマネージャーを使用して後でインストールできます。Windowsバージョンとは対照的に、図6.55に示すように、USBデバイスによって提供されるURLへの自動転送もここで機能します。

Linux PCでは、通常、WebブラウザにはUSBデバイスを直接アドレス指定するための十分な権限がありません。のインストールudev-セクション2.1.4で説明されているルールは、アクセスを有効にします。

6.18.2WebUSBスケッチ

いくつかのWebUSBのサンプルスケッチは、の例として示されています。 ftドゥイーノ-ArduinoIDEのインストール ファイル 例 。WebUSB 含まれています。

[34]Android用UsbSerial： https://github.com/felHR85/UsbSerial
[35]WebUSB： https://developers.google.com/web/updates/2016/03/access-usb-devices-on-the-web
[36]Google Chromeブラウザ： https://www.google.com/intl/de_ALL/chrome/
[37]Chromeポータブル： https://portableapps.com/apps/internet/google_chrome_portable

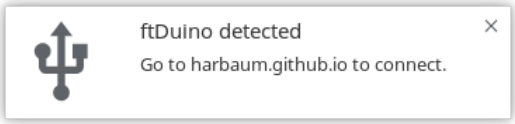


図6.55：メッセージ ftドゥイーノ Chromiumブラウザを実行している

WebUSBプロジェクトのURLはスケッチで直接指定され、それに応じて独自のスケッチに適合させることができます。

```
WebUSB WebUSBSerial （1 / * https：// * /、 "harbaum.github.io/ftduino/webusb/console"）；
```

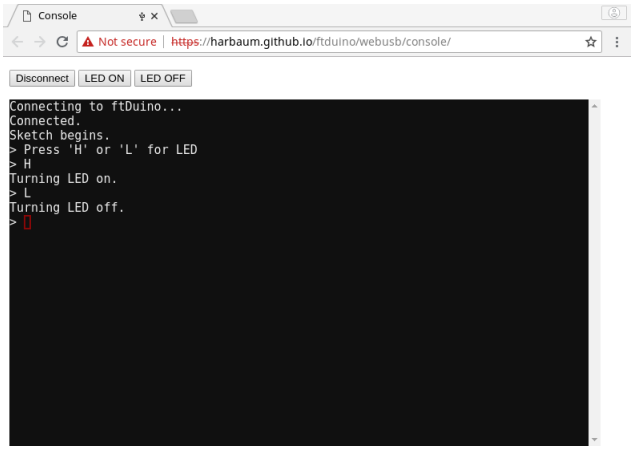
適切なスケッチに加えて、WebUSB仕様では、デバイスのUSB構成をさらに調整する必要があります。Arduino IDEは、ボード選択でボードタイプを選択することにより、必要な変更を行いますftDuino（WebUSB） 図6.56に示すように。適応はWebUSBライブラリに関連してのみ完了し、Windowsはそれ以外の場合はデバイスを認識しないため、この設定はWebUSBにのみ使用する必要があります。

図6.56：WebUSB構成の選択

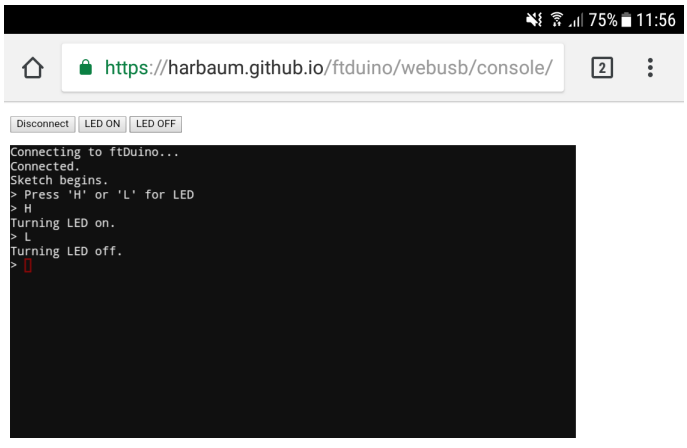
6.18.3コンソール

the ファイル例 。WebUSB 。コンソール --Sketchは、USB経由のシンプルなコマンドインターフェイスを提供します。内蔵の発光ダイオードはオンとオフを切り替えることができます。

上のスケッチは ftドゥイーノ それをインストールしました ftドゥイーノ PCに接続し、ChromeまたはChromiumブラウザを開いて、ワンクリックで十分です 接続、 ブラウザからデバイスへの接続を確立します。発光ダイオードは、コンソールに直接入力するコマンドを介して制御するか、ボタンをクリックして切り替えることができます。



（a）Linux上のChromiumブラウザ

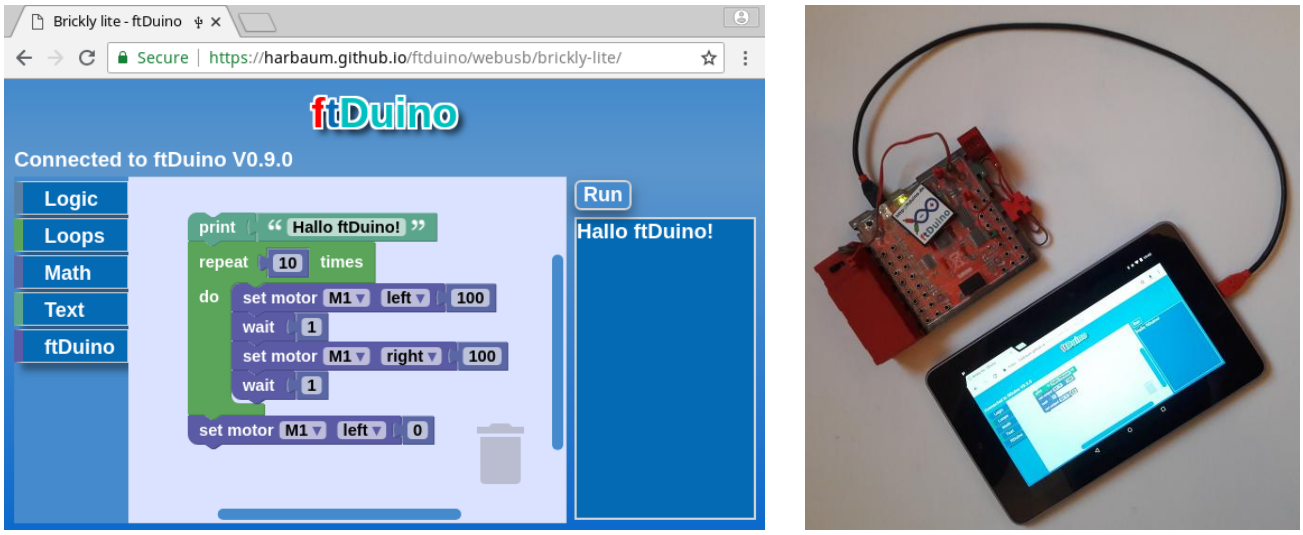


（b）AndroidのChromeブラウザ

図6.57：のWebUSBコンソール ftドゥイーノ

6.18.4ブリックライト

セクション8.3で説明されているBricklyプロジェクト[38] TXTコントローラーのコミュニティファームウェアから開始します。BricklyはBlocklyに基づいています[39]、Webブラウザを介して使用するためのグラフィックプログラミング環境。TXTの場合、プログラマーはWebブラウザーを使用してスマートフォン、PC、またはタブレットで作成し、TXTで実行できます。



（a）PCのChromeブラウザで

（b）Lindy31717ケーブルを搭載したAndroidタブレット

図6.58：ブリックライト ftドゥイーノ

Brickly-liteは ftドゥイーノ 完全に再設計されました。でブロック的に生成されたコード実行ftドゥイーノ 放棄されました。代わりに、Brickly-liteプログラムはブラウザーで作成され、ブラウザーでも実行されます。ブラウザは、接続されたUSBポートにのみアクセスして、Schertechnikセンサーとアクチュエーターを操作します ftドゥイーノ 戻る。にftドゥイーノ スケッチをしなければならない ファイル例 。WebUSB 。IoServer 走る。これから内部OLEDディスプレイ（セクション1.2.7を参照）を処理できるSketchは、AdafruitGFXライブラリです。[40] スケッチを翻訳する必要がありました。

Brickly-liteのWebサイトは次の場所にあります。 https://harbaum.github.io/ftduino/webusb/brickly-lite/。を使用するにはftドゥイーノ インストール済み ファイル例 。WebUSB 。IoServer -PC、スマートフォン、またはタブレットで直接スケッチする接続されます。その後、Chromeブラウザはに接続しますftドゥイーノ モデルで入力または出力が行われるとすぐに、ブラウザは対応するコマンドを ftドゥイーノ。

残念ながら、実際のプラグアンドプレイエクスペリエンスの要件は、Chromeブラウザーの定義と、Windowsでデバイスによって送信されるURLの自動開始の欠如によって満たすことができません。USBスティックに適切なブラウザをポータブルにインストールする準備ができていれば、インストールなしで初心者が使用でき、たとえば学校のPCプールで使用できるシナリオを簡単に作成できます。以下の例 ファイル例 。WebUSB 。コンソール ブラウザ操作の単純なモデルの開始点として機能できます。

同様のグラフィックプログラミングシステムは、Arduino用のScratchとScatch3.0です。詳細については、セクション8.6および6.18.5を参照してください。

6.18.5スクラッチ3.0

Bricklyと同様に、Scratch3.0はGoogleのBlocklyプロジェクトに基づいています。したがって、どちらもブラウザからWebUSBに接続するための非常によく似た方法を使用しますftドゥイーノ アクセスするために。

[38] TXTのブリック： https://cfw.ftcommunity.de/ftcommunity-TXT/de/programming/brickly/
[39] ブロック的に： https://developers.google.com/blockly/
[40] Adafruit GFXライブラリ： https://github.com/adafruit/Adafruit-GFX-Library

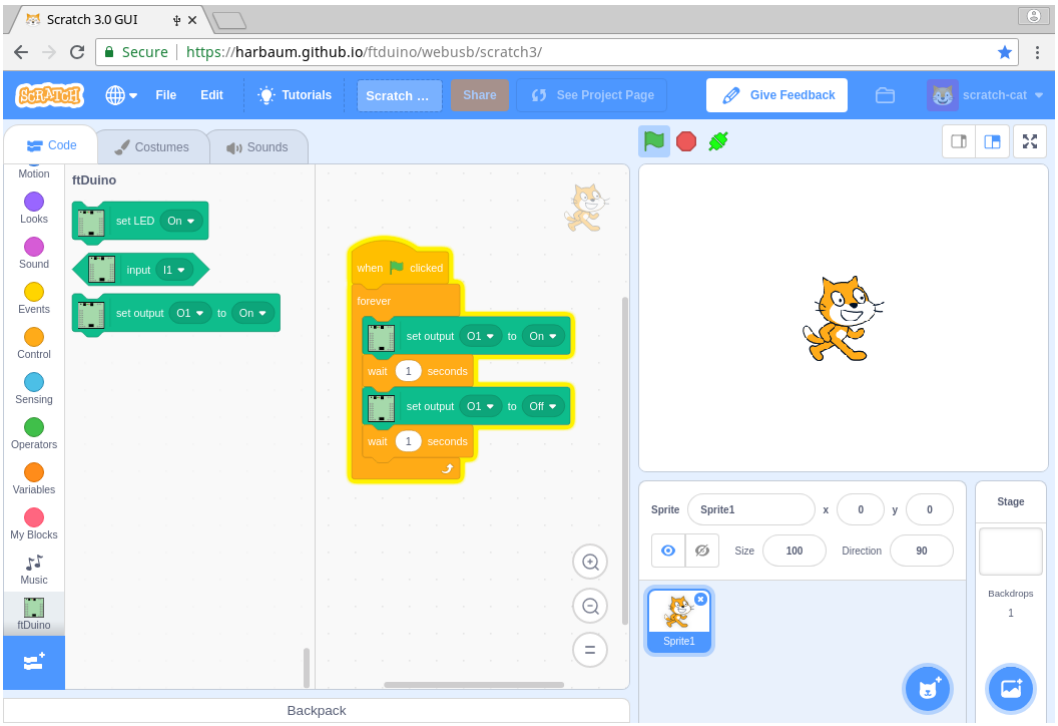


図6.59：ChromeブラウザでのScratch 3.0 ftドゥイーノ-拡大

ブリックリーは完全ですが ftドゥイーノ-特定の開発、Scratch3.0は独立したプロジェクトです。アンダーhttps://harbaum.github.io/ftduino/webusb/scratch3/ Scratch 3.0の有限バージョンには、 ftドゥイーノ。

the ftドゥイーノ-拡張機能は ftドゥイーノ したがって、たとえば、レゴの拡張機能とは対照的に、PC上に追加のソフトウェアは必要ありません。これは、拡張機能が主にWebUSBをサポートするすべてのプラットフォームで使用できることを意味します。通常のWindows、Apple、Linux PCに加えて、これにはほとんどのスマートフォンとAndroidタブレットも含まれます。

Scratch3.0との使用に関する詳細情報 ftドゥイーノ セクション5.1.3にあります。

6.19 Bluetooth

難易度： ★★★★★

Bluetoothは、短距離用のワイヤレス伝送テクノロジーです。ほとんどのPC、タブレット、スマートフォンはBluetoothを使用できるため、エンドデバイスにワイヤレスで接続できます。多くのおもちゃもBluetoothをサポートしており、ほとんどのLegoまたはSchertechnikコントローラーもこのテクノロジーに精通しています。

6.19.1Bluetoothバリアント

Bluetoothには、データレート、範囲、エネルギー要件などが異なる多くのバリエーションがあります。違いのほとんどは、上のアプリケーションのためのものですftドゥイーノ 無関係。ただし、いわゆるクラシックBluetoothとBluetooth Low Energyには、次の用途での使用を目的とした基本的な違いがあります。ftドゥイーノ 重要です。

クラシックBluetooth

元のバリアントは通常、クラシックBluetoothと呼ばれます。これは主にPCとモバイルデバイス間のデータの自発的な交換のために設計されており、たとえばBluetoothを使用する場合に使用されます

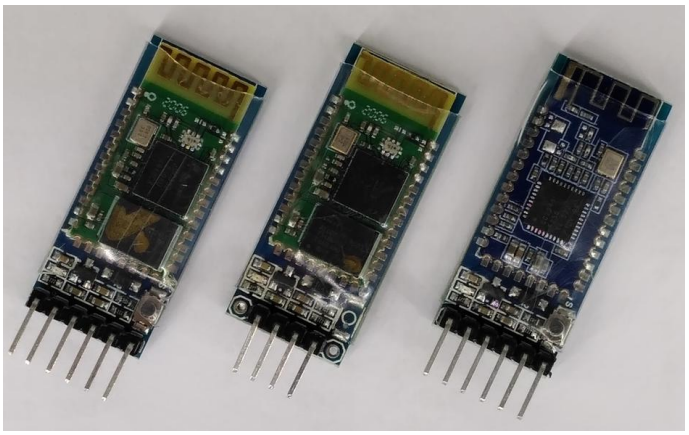


図6.60：Arduino用のBluetoothモジュールHC-05、HC-06、HM-10

ファイル、写真、連絡先などを交換することができます。Bluetoothキーボードとマウス、およびBluetoothヘッドフォンとスピーカーも、従来のBluetoothを使用しています。従来のBluetoothは、このためのいわゆるプロファイルを実装しています。これらのプロファイルの1つは、たとえば、ファイルの交換に使用されるObject Exchangeプロファイル（OBEX）です。

従来のBluetoothのもう1つのプロファイルは、いわゆるシリアルポートプロファイル（SPP）です。このプロファイルは、ポイントツーポイントのデータ接続を作成します。SPPは、主に有線シリアルRS232接続を置き換えることを目的としており、たとえば、個別のGPS受信機や同様のデバイスでよく使用されていました。現在、SPPは小さな役割を果たしていますが、従来のBluetoothを使用するほとんどのオペレーティングシステムで引き続きサポートされています。これらには、Windows、MacOS、Linuxなどの現在のPCオペレーティングシステム、および一部のモバイルオペレーティングシステムが含まれます。

PCオペレーティングシステムは常にシリアルRS232接続を処理できるため、これらの接続を使用できるソフトウェアはそれに応じて大量にあります。このサポートは、シリアル接続にマッピングできるSPP、USB、およびその他の最新テクノロジーに引き継がれます。ユーザーにとって、SPP Bluetooth接続は他のシリアル接続と同じように見え、たとえばWindowsではいわゆるCOMポートで表されます。COMポートを使用できるWindowsソフトウェアは、SPPBluetooth接続も使用できます。これには、Arduinoに接続するためのCOMポートを備えたArduinoIDE自体が含まれます。ftドゥイーノ を使用するため、SPPBluetooth接続も使用できます。

シリアル接続の使用は、モバイルオペレーティングシステムではあまり一般的ではありません。SPPはAndroidで使用できますが、IOSではサポートされなくなりました。

TXTコントローラーやRaspberryPiなどのデバイスも、従来のBluetoothとSPPをサポートしています。

Bluetooth Low Energy（BLE）

従来のBluetoothとは対照的に、Bluetooth Low Energy（BLE）は非常に若いテクノロジーです。これはBluetoothファミリに非常に遅れて追加され、まったく新しい伝送方法とプロトコルに基づいています。BLEデバイスとの通信には、比較的最新のハードウェアと完全に異なるソフトウェアが必要です。BLEは主に低エネルギー消費を対象としていました。これは、低いデータレートと転送される少量のデータによって実現されます。さらに、BLEは、接続の確立と切断という複雑で時間のかかるプロセスを不要にします。代わりに、小さなデータパケットがBLEで送信されます。

BLEがPCセクターで使用されることはめったになく、最新のWindowsオペレーティングシステムバージョンでのサポートには問題があることがよくあります。[41]。しかし、特にモバイルデバイスをスマートウォッチやパルスセンサーなどのデバイスと接続する場合、BLEはその地位を確立しています。AndroidおよびIOSのモバイルオペレーティングシステムでのBLEのサポートは成熟しており、信頼性があります。

TXTコントローラーやRaspberryPiなどのデバイスは、BLEだけでなく従来のBluetoothもサポートしています。

[41] schertechnikエラーメッセージWin-10： https：// www。schertechnik.de/-/media/ schertechnik / te / service /ダウンロード/ robotics / robo-prolight /ドキュメント/hinweis-win-10-Fehler-rpl436-de.ashx

クラシックBluetoothまたはBLE？

以前の説明はすでに示唆しています。従来のBluetoothは主にPCで使用されていますが、BLEはモバイル環境で選択されるテクノロジーです。

それは、それが一般的に逆に機能しないという意味ではありません。BLEは特定の条件下でWindowsで使用でき、Androidは従来のBluetoothに対して広範なSPPサポートを提供します。ただし、疑わしい場合は、PC環境の従来のBluetoothがより簡単なソリューションであり、BLEをモバイルデバイスと組み合わせて使用 できます。

Arduino BluetoothモジュールHC-05、HC-06およびHM-10

シンプルなArduinoは、Bluetoothサポートがほとんど機能していません。 ftドゥイーノ。したがって、Bluetoothを使用するには追加のハードウェアが必要です。図6.60に示すように、HC-05、HC-06、およびHM-10シリーズの小型モジュールは、非常に安価で広く普及しています。

HC-05およびHC-06モジュールは従来のBluetoothをサポートしますが、HM-10モジュールはBluetooth LowEnergyを使用します。HC-05とHC-06の違いは小さいです。HC-05はもう少し強力で、たとえば、他のデバイス（それ自体の種類を含む）への接続を確立できますが、HC-06は他のデバイスによるアドレス指定に制限されています。通常、PCまたはモバイルデバイスを使用して接続するため、これはほとんどの目的に完全に十分です。ftドゥイーノ 製造したい。

HC-05およびHC-06は、WindowsPCへの接続に特に適しています。一方、HM-10モジュールは、IOSベースのモバイルデバイスで使用できる3つのモジュールのうちの1つだけです。

	ウィンドウズ	Linux	マックOS	アンドロイド	IOS
HC-05	わかった	わかったわかった	わかった	使えない	
HC-06	わかった	わかったわかった	わかった	使えない	
HM-10	問題がある	わかったわかった	わかった	わかった	

6.19.2への接続 ftドゥイーノ

電気的および機械的に、3つのモジュールはわずかに異なるだけなので、 ftドゥイーノ は3つすべてで同じであり、モジュールはいつでも交換できます。

Arduinoへの接続/ftドゥイーノ 4つの信号を介して3つのモジュールすべてで発生します。

図6.61：HC-06の接続

VCC	グランド5ボルトの供給電圧
GND	
TXD	モジュールのシリアルUARTデータ出力モ
RXD	ジュールへのシリアルUARTデータ入力

Arduinoとマイクロコントローラーの両方に適切なUARTインターフェースがあります ftドゥイーノ。従来のArduinoUNOでは、このインターフェースはすでにPCとのUSB通信に使用されています。でftドゥイーノ カウンタ入力のこれらの信号になります C1 と C2 使用済み。どちらの場合も、UARTインターフェイスをBluetoothモジュールの接続に直接使用することはできません。

さらに、モジュールには5ボルトが供給されますが、UART信号は3.3ボルトしか使用しないことに注意してください。これにより、モジュールからArduino /へのTXD接続が可能になります。ftドゥイーノ これらはモジュールのより低い電圧レベルを処理できるため、違いはありません。Arduinoからの接続/ftドゥイーノ モジュールのRXD入力への接続は、ArduinoのTX出力からの5ボルト全体ではないように設計する必要があります。ftドゥイーノ モジュールに来てください。インターネット上のさまざまな指示はこの規則を無視します。これは、モジュールがこの点でより堅牢であることを示唆しています。

下側のプリントが示唆するよりも。それでも、3.3ボルトの制限内にとどまることは確かに害はありません。

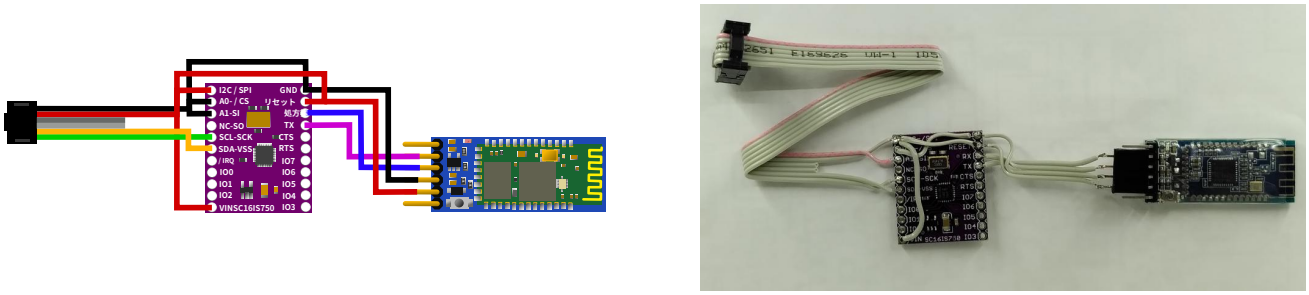
SoftwareSerial

ArduinoのマイクロコントローラーのハードウェアUARTへのアクセスがありません ftドゥイーノ Arduinoの世界では、通常、SoftwareSerialライブラリを使用して回避できます[42]。彼らの助けを借りて、マイクロコントローラーの他のピンも限られた範囲でUARTタスクを引き受けることができます。でftドゥイーノ マイクロコントローラの2つの接続のみが外部から直接アクセスできます。これらは、IのSCLおよびSDA信号です。2のCポート ftドゥイーノ。残念ながら、これら2つの信号は SoftwareSerialを介して使用することはできません。したがって、SoftwareSerialはftドゥイーノ Arduinosで通常行われているように、Bluetoothモジュールとの通信には使用できません。

私。2C-UART SC16IS750

エレガントなオプションは、Iを分離するモジュールを使用することです2のCポート ftドゥイーノ Bluetoothモジュールに必要なUARTプロトコル。

そのようなコンポーネントの1つがSC16IS750です。[43] NXPによる。特に、シンプルなアダプタボードでCJMCU 750という名前でオンラインで入手でき、Iに直接接続できます。2のCポート ftドゥイーノ BluetoothモジュールのUART信号と同様に。このチップのもう1つの利点は、3.3ボルトの信号で動作するため、Bluetoothモジュールに危険を及ぼすことがないことです。一方、彼自身はIで5ボルトに耐性があります。2C側 ftドゥイーノ。



（a）接続スキーム （b）実装にははんだ付けが必要です

図6.62：CJMCU750-Iを介したBluetooth2C-UARTアダプター

図6.19.2は、I間の必要な接続を示しています。2のCポート ftドゥイーノ そして私。2CJMCU 750のC接続、およびCJMCU750とBluetoothモジュールのUART接続を確立する必要があります。

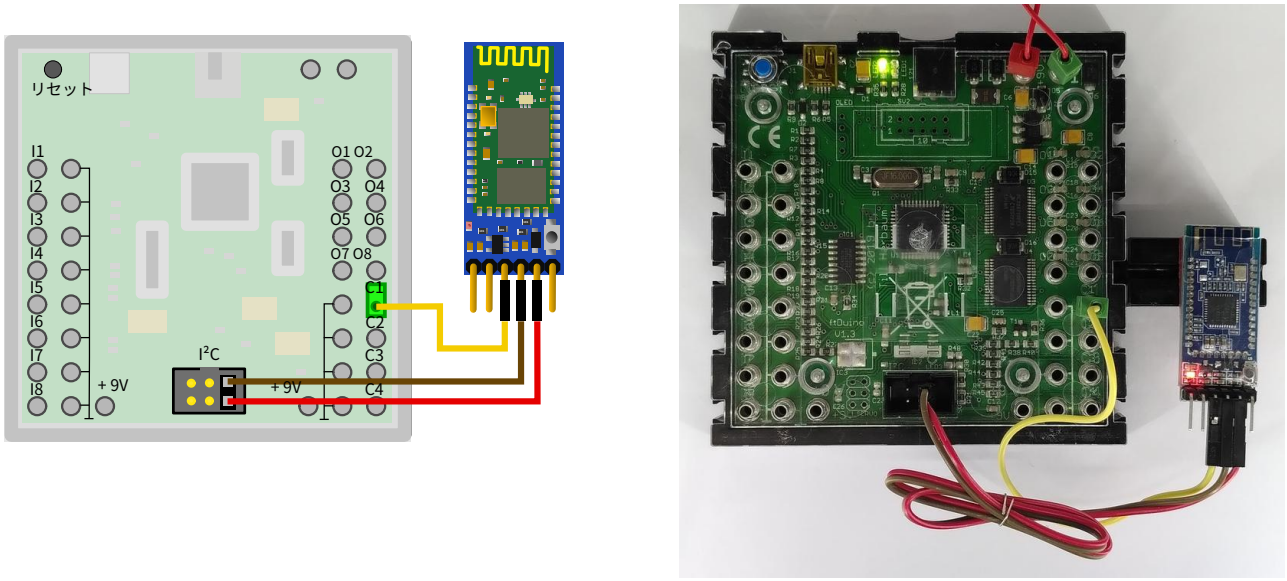
接続 A0-CS と A1-SI この例では、CJMCU750の一部が接地されています。したがって、SC16IS750はアドレスに反応します0x4e 1日2Cバス。これは、次のスケッチ例で考慮されています。複数のSC16IS750を接続する場合ftドゥイーノ それに応じて接続することで接続できます A0-CS と A1-SI 別のアドレスを割り当てることができます。

簡単なスケッチ例を以下に示します。 ファイル例 。Ftduino 。ブルートゥース 。CJMCU_750 。橋 。彼のUSBポート間で両方向にブリッジ ftドゥイーノ PCへの接続とSC16IS750のBluetoothモジュールへのUART接続。PCから（たとえば、シリアルモニターを使用して、セクション3.3.1を参照）に送信されるすべての文字ftドゥイーノ 送信元 ftドゥイーノ Bluetoothモジュールに転送されます。その逆も同様です。Bluetoothモジュールと直接通信し、Bluetooth経由でシリアルモニターからデータを送信できます。次に、Bluetooth経由で受信したデータがシリアルモニターに直接表示されます。

[42] SoftwareSerialライブラリ： https://www.arduino.cc/en/Reference/SoftwareSerial
[43] SC16IS750データシート： https://www.nxp.com/docs/en/data-sheet/SC16IS740_750_760.pdf

に簡単に接続 C1

すでに述べたように、マイクロコントローラのUART ftドゥイーノ すでに内部で使用されています。schertechnik超音波センサー（セクション1.2.6を参照）もUARTを使用しているため、UARTはftドゥイーノ カウンター入力 C1 アクセス可能になりました。の保護回路ftドゥイーノ BluetoothモジュールのTXD接続がカウンター入力に直接接続されるように、着信信号への影響が少ない C1 の ftドゥイーノ マイクロコントローラのUARTを接続し、その恩恵を受けることができます。



（a）スキーム （b）最も単純な実装

図6.63：受信専用：Bluetoothオン C1

モジュールに必要な5ボルトの供給電圧はIから見ることができます。2C接続をタップオフする必要があります。これを行う最も簡単な方法は、図に示すように、いわゆるソケット間ジャンパー線を使用することです。これらのケーブルは、のピンに直接接続できます。ftドゥイーノ-私2CコネクタもBluetoothモジュールのピンに接続します。

重要：この方法で接続されたBluetoothモジュールは、 ftドゥイーノ 受け取るが、彼に送るだけです。たとえば、PCへのBluetooth接続の場合、これはデータをPCからPCに転送できることを意味します。 ftドゥイーノ 送信できますが、からは送信できません ftドゥイーノ PCに。ただし、多くの単純なリモートコントロールアプリケーションでは、これで十分であり、たとえば、ArduinoBlue（セクション6.19.3を参照）を完全に使用できます。

PCからのデータの受信は、たとえば、Sketchを使用して行うことができます。 ファイル例 。Ftduino 。ブルートゥース 。CounterPort 。橋 シリアルモニターで観察できます。

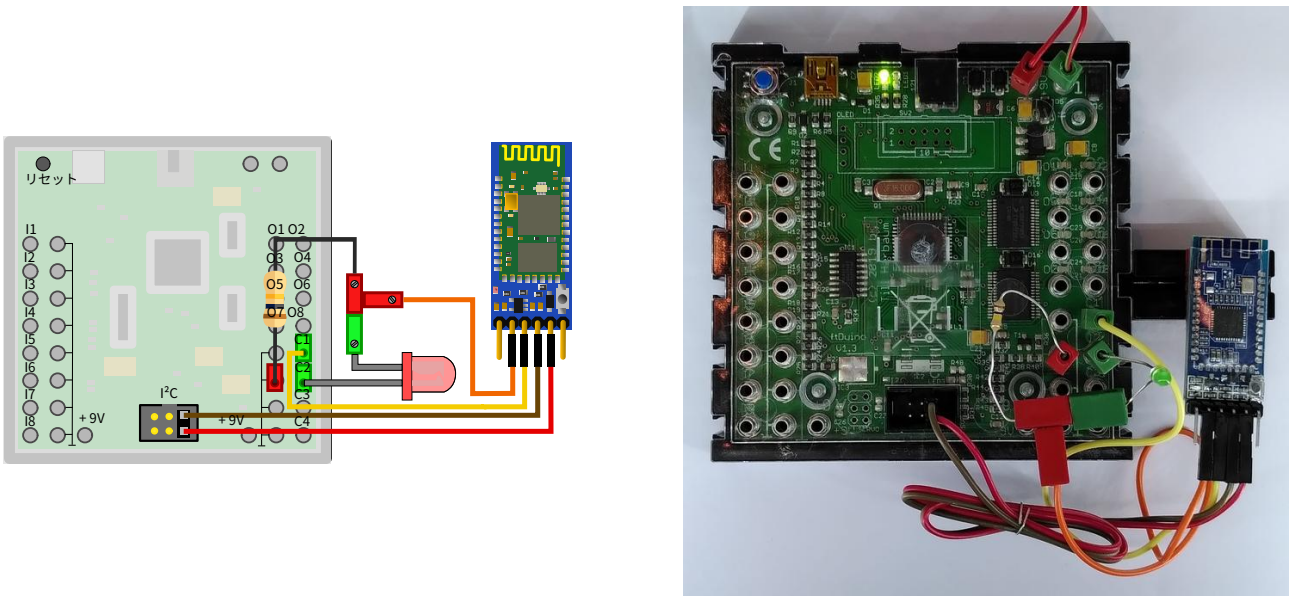
この形式でのUARTの使用は、 Ftduino私が言ったように、UARTは超音波センサーにも使用されているため、ライブラリ。スケッチ例 ファイル例 。Ftduino 。ブルートゥース 。CounterPort 。ArduinoBlue したがって、の修正バージョンが含まれています Ftduinoサポートしていないライブラリ超音波センサー用。このバージョンは、Bluetooth接続に接続せずに使用できますC1 利用される。

への双方向接続 C1＆C2

からデータを送信するために使用されるマイクロコントローラのUART伝送ライン ftドゥイーノ Bluetoothモジュールの場合、メーター接続には内部的に必要です C2 使用済み。この接続は、通常の構成になっていますftドゥイーノ 入力と適切な保護回路を備えています。

この接続を出力として使用する場合、これらの保護回路により、出力デジタル信号の電圧が低レベル（0）として2.5ボルト、高レベル（1）として5ボルトになります。BluetoothモジュールのRXD入力は、低信号の場合は0.8ボルト未満、高信号の場合は2〜3.3ボルトの電圧を想定しています。

単純なシリコン発光ダイオードは、それに応じて電圧を下げる最も簡単な方法の1つです。図6.64に示すように、それらの電圧降下を使用して、信号レベルを約2ボルト下げることができます。結果として生じる0.5ボルトと3ボルトは、Bluetoothモジュールによって確実に検出され、3.3ボルト未満のBluetoothモジュールの許容範囲内にあります。発光ダイオードを接続するときは、極性を守る必要があります。発光ダイオード（アノード）の長い方の接続線は、C2-の接続ftドゥイーノ。発光ダイオードと抵抗器が正しく接続されている場合、発光ダイオードは非常に弱く点灯するはずです。100kの抵抗は、3ボルトから0.5ボルトに変更するときに電圧が十分に速く降下することを保証します。



（a）スキーム （b）簡単な実装

図6.64：送受信：Bluetoothオン C1＆C2

この回路により、双方向動作が可能であり、通信は両方からです。 ftドゥイーノ PCに、PCから反対方向に ftドゥイーノ 可能。

スケッチ ファイル例 。Ftduino 。ブルートゥース 。CounterPort 。橋 とりわけ、
いわゆるATコマンドを使用したBluetoothモジュールの構成。図6.65は、ブリッジスケッチを使用したコマンドに対するHM-10の反応を示しています。AT + NAME、AT + VERSION と AT +ヘルプ。 HC-05とHC-06は、コマンドに対する反応が少し異なる場合があります。または、最初にボタンを押してコマンドモードにする必要があります。これらのコマンドは、最初は単純な操作では意味がありません。これらは、双方向でのコミュニケーションの成功例にすぎません。

への双方向接続 C1、C2 および電圧レギュレータ

残りの欠点は、Iの使用です。2Bluetoothモジュールの電源用のC接続。一方では、これは接続を機械的にブロックし、Iを接続するために使用できなくなります2Cセンサーが使用できます。一方、の内部電源ftドゥイーノ 不必要に負担。

したがって、図6.66は、単純な7805電圧レギュレータが9ボルトの ftドゥイーノ を有効にします。このソリューションの唯一の欠点は、ftドゥイーノ 現在、実際には9ボルトを供給する必要があります。私の使用のために2Cコネクタは ftドゥイーノ USB経由で十分です。

図6.67は、必要な個々の部品と、それらをどのように使用できるかを詳細に示しています。 ftドゥイーノ 接続する必要があります。必要な個々の部品は、例えばReicheltにあります[44]利用可能。リンクリストにはHC-05Bluetoothモジュールも含まれており、必要に応じてHM-10（残念ながらReicheltからは入手できません）に置き換えることができます。

[44]ReicheltコンポーネントリストftDuinoBluetooth HC05：https://www.reichelt.de/my/1733294

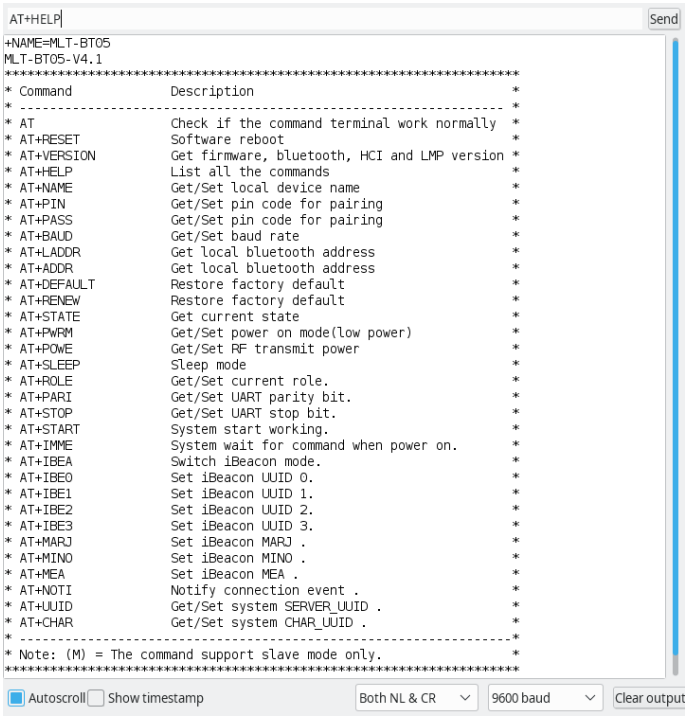
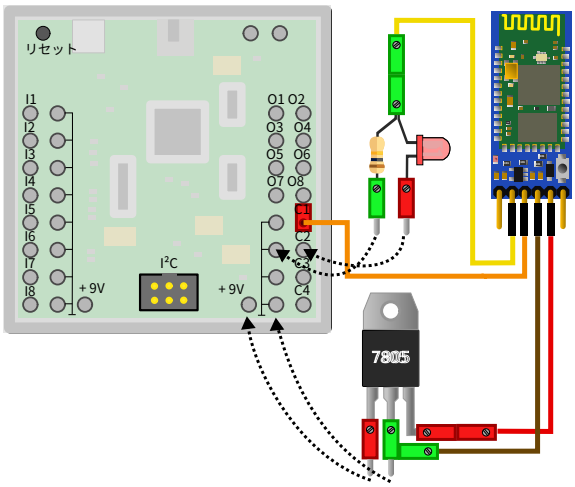
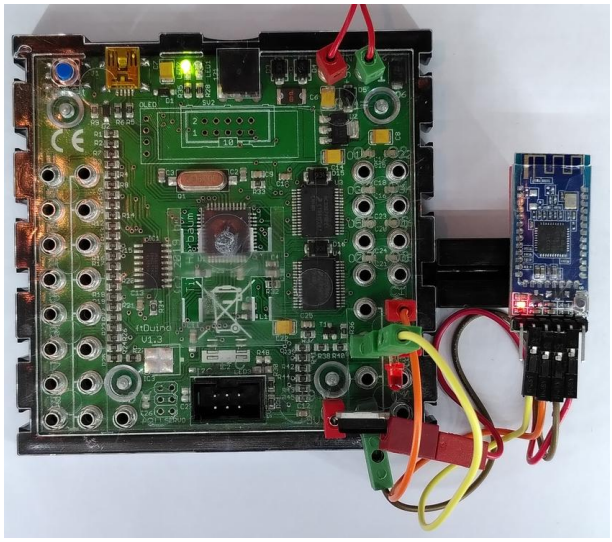


図6.65：HM-10を使用したATコマンド



（a）スキーム



（b）簡単な実装

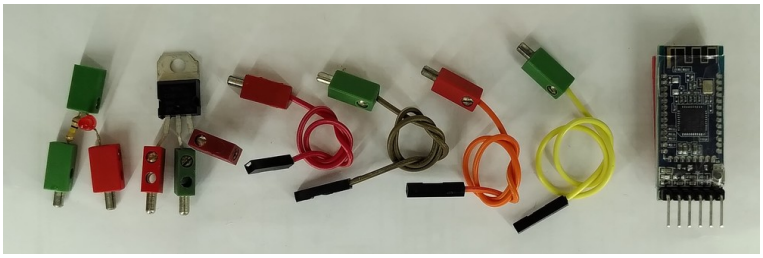
図6.66：求婚者I。2C接続：Bluetoothから9V、 C1＆C2

まとめ

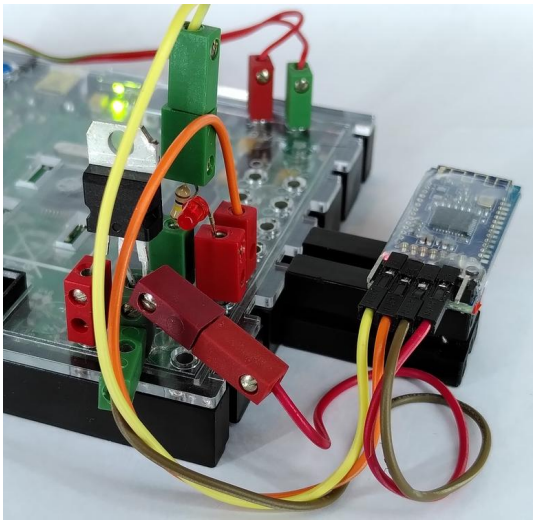
これを行うにはいくつかの方法があります ftドゥイーノ Bluetoothモジュールで拡張可能。これらのソリューションの主な長所と短所は次のとおりです。

SoftwareSerial この人気のある亜種は ftドゥイーノ 使用できません。

私。2C-UART SC16IS750 私。2C-UARTは最も高価なソリューションであり、はんだ付けが必要です。このソリューションは電気的なものです最もクリーンで技術的にエレガントです。



（a）必要な部品



（b）アセンブリ

図6.67：9V電源とデータ接続

経由の接続 C1 ただ C1 使用すると、最も単純な構造が可能になります。ただし、このソリューションではデータ受信のみが可能です。
キャッチは可能ですが、送信はできません。

経由の接続 C1＆C2 の追加使用 C2 セットアップには少し余分な労力が必要です
しかし、双方向通信。

経由の接続 C1、C2 および電圧レギュレータ この解決策はもう少し面倒です。このために、私は2のCポート
ftドゥイーノ 他の使用は無料です。

最初の試みは、唯一の接続を介して最も簡単です C1 可能。純粋なリモートコントロールアプリケーションはすでに可能です。の追加使用C2 Bluetooth通信のすべての可能性を開きます。Iを使用します。2C-UARTまたは9ボルト電源はオプションであり、経験とニーズに応じて実行できます。

Bluetoothをschertechnik超音波センサーと一緒に使用する必要があります ftドゥイーノ 使用される、の使用 C1 オフとそれは私でなければなりません。2C-UARTを使用できます。

6.19.3PCまたはスマートフォンでの使用

Bluetoothオン ftドゥイーノ PCまたはスマートフォンで適切なソフトウェアの形で適切なリモートステーションを使用する必要があります。

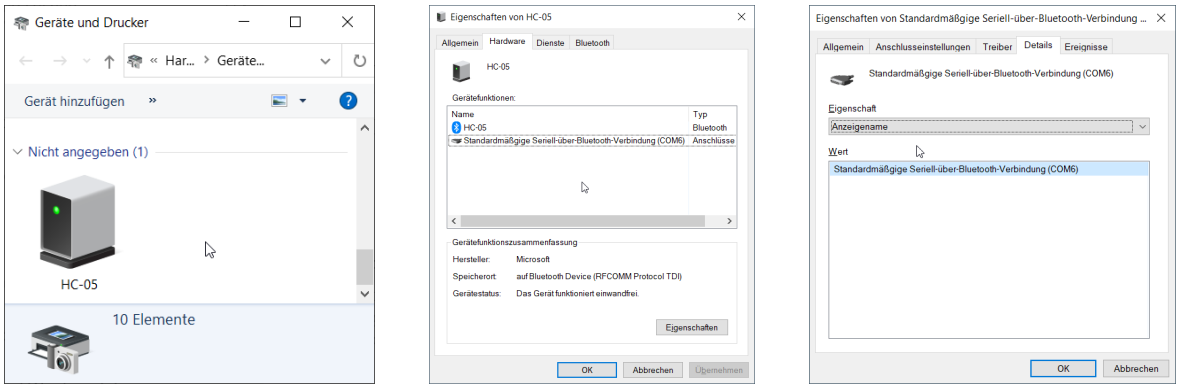
Windows上のHC-05

説明したように、HC-05はWindowsPCでの使用に最適です。電圧が供給され、赤色発光ダイオードが激しく点滅すると（1秒間に約5回）、PCからの接続を受け入れる準備が整います。

PCでは、右下のツールバーにあるBluetoothアイコンを使用して、HC-05を新しいデバイスとして統合します。結合中にセキュリティピンが要求されます。ここに示されているすべてのモジュールについて、これは元の作品です1234。

ペアリングが成功すると、図6.68に示すように、HC-05がWindowsの[デバイスとプリンタ]ウィンドウに表示されます。次に、実際の通信はCOMポートを介して行われます。COMポートは次のように確認できます。この場合、HC-05モジュールはCOM6 統合されており、たとえば、シリアルモニターを使用してArduinoIDEからアドレス指定できます。

注：Bluetooth COMポートは、Bluetoothを介したデータ交換専用に使用されます。たとえば、新しいスケッチをに追加することはできませんftドゥイーノ Bluetooth経由で再生します。



（a）デバイスおよびプリンターのHC-05　（b）HC-05の特性　（c）COMポートのプロパティ

図6.68：WindowsでのBluetoothCOMポートの決定

Arduino IDEに加えて、Tera-Termなどの他のいわゆるターミナルプログラムが適しています[45] COMポートを介した通信用。Arduino IDEは一度に1つのシリアルモニターウィンドウしか開くことができないため、テストデータが必要な場合は、2番目のターミナルプログラムを使用する必要があります。ftドゥイーノ-USBCOMポートとBluetoothCOMポートを交換したい。

Windowsで使用するためのサンプルスケッチは次のとおりです。 ファイル 。 例 。 Ftduino 。ブルートゥース 。CounterPort 。 橋 と ファイル例 。Ftduino 。ブルートゥース 。CJMCU_750 。 橋。

スマートフォンのHM-10

主要なモバイルオペレーティングシステムのアプリストアでは、HM-10モジュールにアクセスしたり、Androidの場合はHC-05またはHC-06にアクセスしたりするためのさまざまなアプリを提供しています。

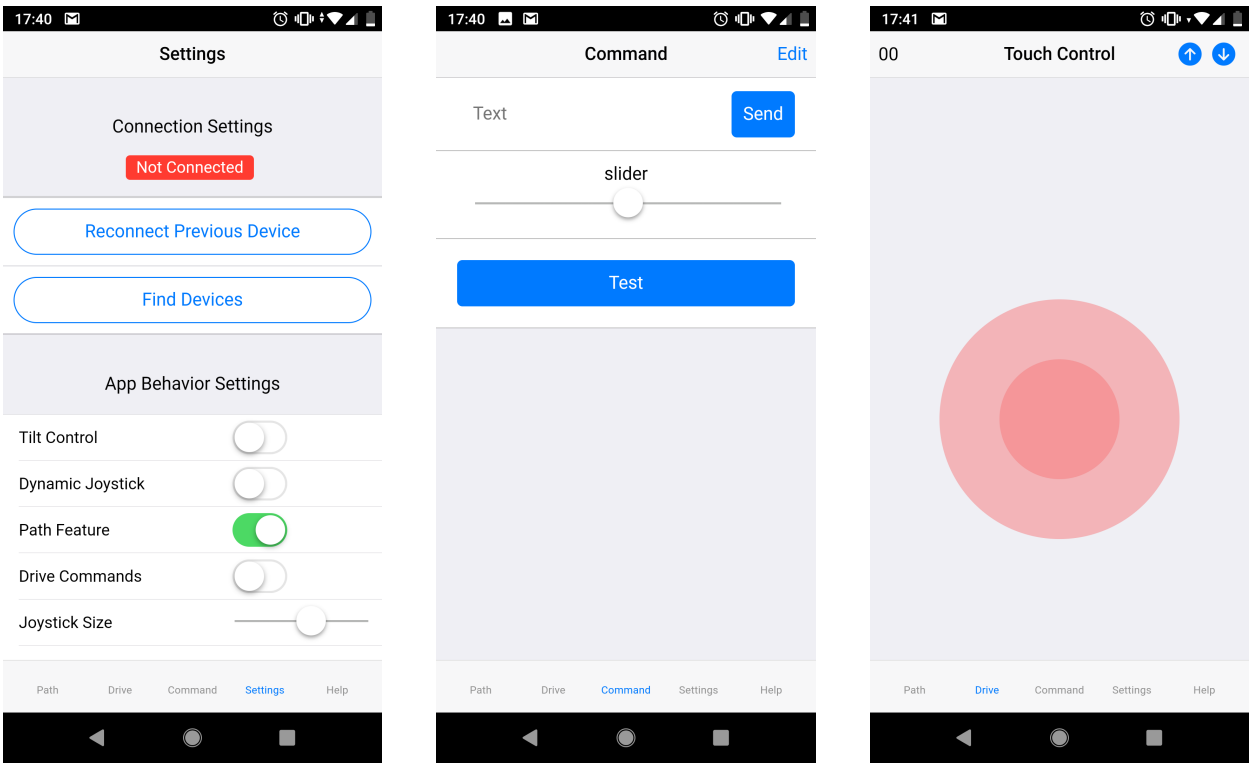
無料および広告なしのArduinoBlueアプリはHM-10で際立っています[46] 群衆の中から。

図6.69に示すように、ArduinoBlueインターフェースはクリアに保たれ、モデルのジョイスティック制御と調整可能なボタンとスライダーを使用した制御の両方を可能にします。

ArduinoBlueはAndroidとIOSで利用できます。

ArduinoBlueで使用するサンプルスケッチは次のとおりです。 ファイル 。 例 。 Ftduino 。ブルートゥース 。CounterPort 。 ArduinoBlue と ファイル例 。Ftduino 。ブルートゥース 。CJMCU_750 。 ArduinoBlue 。前者は再び含まれています これは、のカスタマイズされたバージョンです ftduino-ライブラリとモーターがどのように動作するかの例を示しています M1-の接続 ftドゥイーノ ArduinoBlueのジョイスティックで制御できます。

[45] テラターム： http://ttssh2.osdn.jp
[46] ArduinoBlue： https://sites.google.com/stonybrook.edu/arduinoble/



（a）設定　（b）コマンド　(c) ジョイスティック

図6.69：AndroidでのArduinoBlue

第7章

モデル

の第6章からの実験中に ftドゥイーノコントローラが焦点であり、使用された外部コンポーネントはごくわずかでした。この章では、より複雑なモデルについて説明します。theftドゥイーノ ここで従属的な役割を果たします。

すべてのモデルは現在の建設キットからのものであるか、モデルに密接に基づいているため、対応するキットとのレプリカが可能です。

7.1自動化ロボット：ハイベイ倉庫

ハイベイウェアハウスモデルは、AutomationRobotsキットに含まれています。TXコントローラーの使用法は、元の手順で説明されています。追加のシートでは、TXTコントローラーについて説明しています。

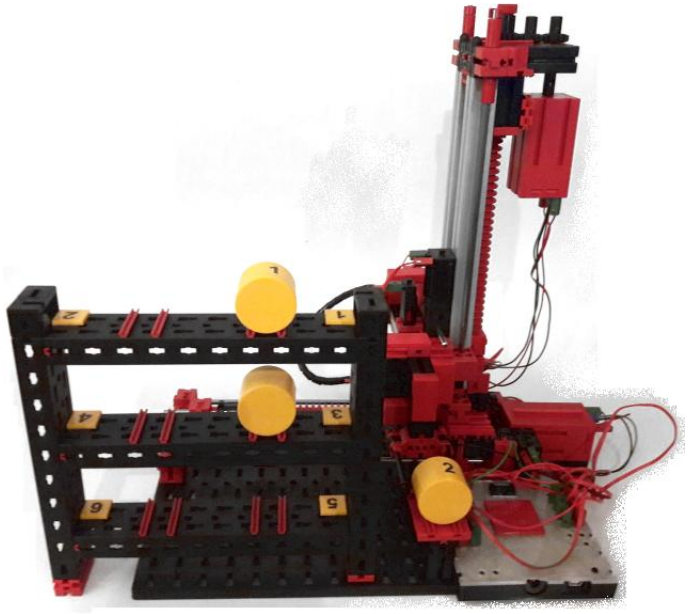


図7.1：ハイラック ftドゥイーノ

スケッチ例 ファイル例 。Ftduino 。HighLevelRack 建物からハイベイ倉庫モデルを制御しますボックス511933ROBOTX自動化ロボット。の接続ftドゥイーノ モデルのは、TXTの回路図に正確に対応しています。

PCのシリアルモニターから操作します[1]。

[1]ハイラックビデオ https://www.youtube.com/watch?v=Sjgv9RnBAbg

重要： コマンド入力が機能するためには、行の端がシリアルモニターで開く必要があります 改行 また 改行（CR） セクション3.3.1で説明されているように設定されています。

図7.2：ハイラックとのシリアル通信

7.2電空：ピンボール

の入力 ftドゥイーノ スイッチモードのSchertechnikフォトトランジスタとも互換性があります。次に、点灯しているトランジスタは真理値を真に、消灯しているトランジスタは真の値を提供します。

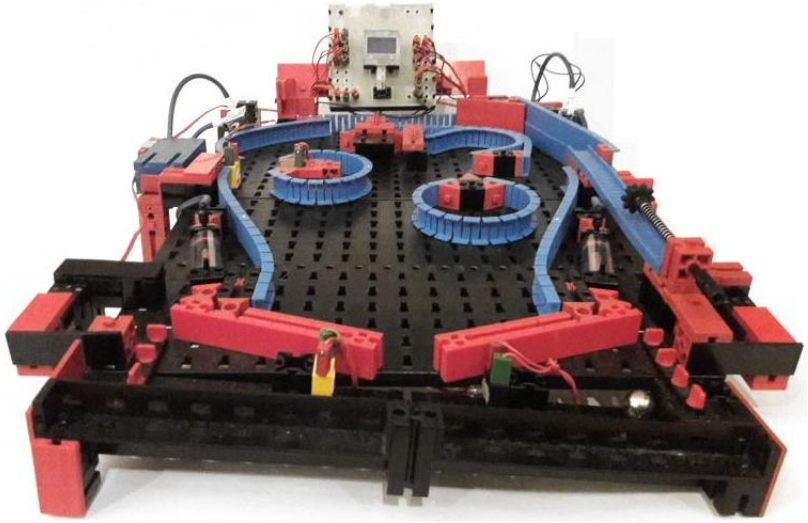


図7.3：フリッパーオン ftドゥイーノ-ベース

電空セットからのピンボールマシンのスケッチ例は以下にあります ファイル例 。Ftduino 。ピンボール 。彼は、光バリアのスイッチ入力としてフォトトランジスタを使用しています。ボールによって中断されたもの次に、ライトバリアは値falseを返します。

```
もしも（！ftduino。input_get（（Ftduino：：：I4））{
  もしも（（ミリス（）- Loose_timer > 1000）{
    //..。
  }
  Loose_timer = ミリス（）；
}
```

タイマーが実行されます。この場合、たとえば、イベント後の最も早い1秒（1000ミリ秒）で別のイベントが認識されるようになります。

このスケッチは、OLEDディスプレイを使用して、残りのゲームボールとスコアを表示します[2]。以来ftドゥイーノ それでも出力が空いている場合は、代わりにランプまたは発光ダイオードを使用できます。

[2]ピンボールビデオ https://www.youtube.com/watch?v=-zmuOhcHRbY

7.3 ROBOTICS TXT Explorer：ラインフォロワー

モバイルラインフォロワーは、ROBOTICS TXT Explorerセットのモデルに基づいており、このセットのIRレーンセンサーを使用します。

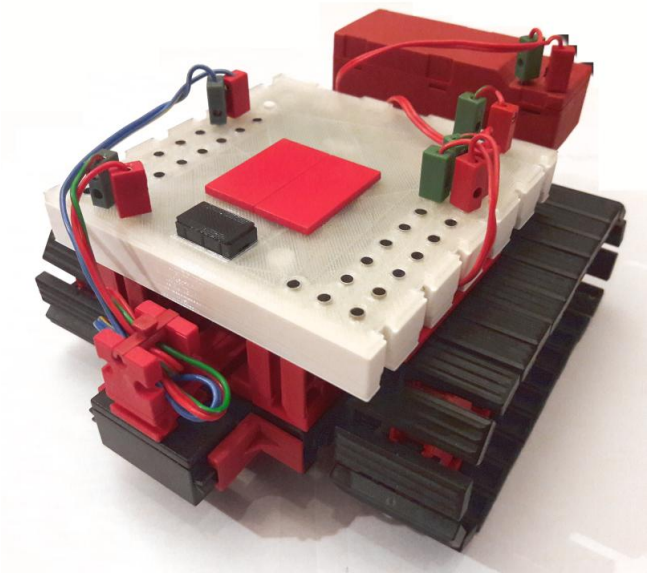


図7.4：上のラインフォロワー ftドゥイーノ-ベース

適切なスケッチ例を以下に示します ファイル例 。Ftduino 。LineSensor 見つけられる。このスケッチは評価します黒い線をたどるためにラインセンサーを継続的にオフにします[3]。

ラインセンサーは、黄色と青色のケーブルで任意の2つの入力に接続されています I1 それまで I8 接続されています。さらに、電力は赤と緑のケーブルを介して供給されますftドゥイーノ。

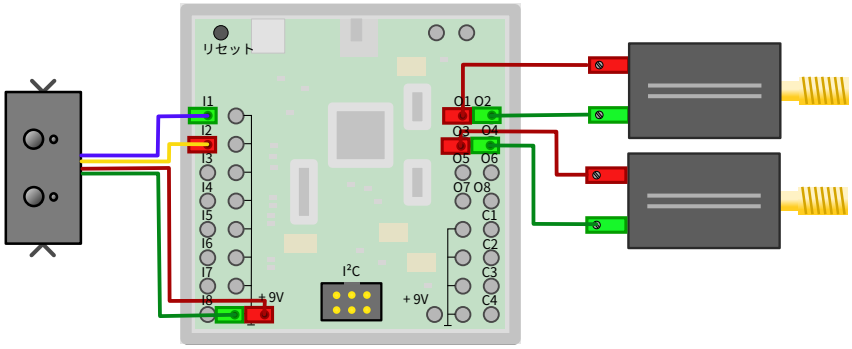


図7.5：ラインフォロアの配線図

この場合、トラックセンサーは入力に接続されています I1 と I2 接続されています。センサーは、白い領域が検出されたときにほぼ最大の電圧（約9ボルト）を供給し、黒い線が検出されたときにわずか数ミリボルトを供給します。

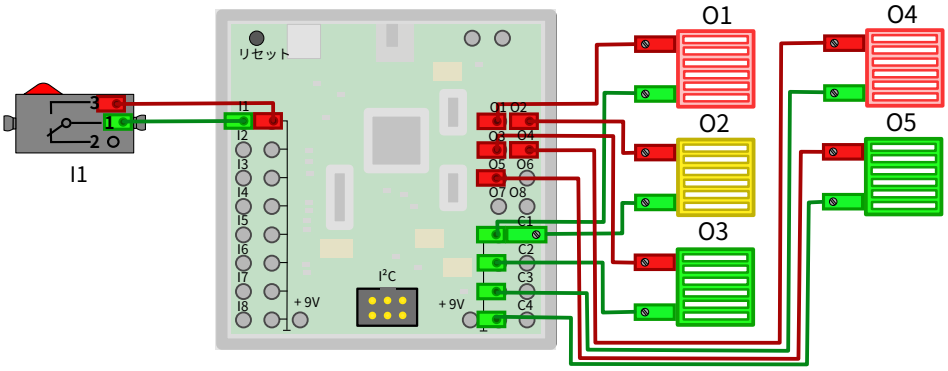
```
//両方の入力を電圧測定に設定します ftduino。input_set_mode （（
Ftduino ： ： ：I1、 ftduino。input_set_mode （（Ftduino ： ： ：I2、電圧）；
                                                Ftduino ： ： ：電圧）；

//両方の電圧を読み取ります
uint16_t left_value = ftduino。input_get （（Ftduino ： ： ：I1）；uint16_t
right_value = ftduino。input_get （（Ftduino ： ： ：I2）；

// 1ボルト（1000mV）未満の電圧は、「ラインが検出された」ことを意味します
```

[3]ラインフォロワービデオ https://www.youtube.com/watch?v=JQ8TLt5MC9k

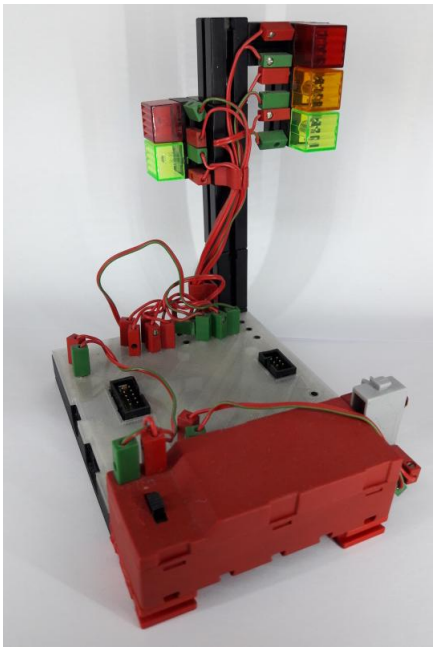
```
もしも（（（left_value <1000）&&（right_value <1000））{
    //両方のセンサーがラインを認識しました//..。

}
```

7.4アイダの信号機

古典的なモデルは、信号機または歩行者用ライトです。このモデルは、車用に3つ、歩行者用に2つのランプを備えた信号機を示しています。



（a）配線図



（b）モデル

図7.6：アイダの信号

適切なスケッチ例を以下に示します ファイル例 。FtduinoSimple 。PedestrianLight 見つけられる。彼は実装します-
ステートマシンの形で信号機に言及します。ライトシーケンスは、8つのステップの通常のプロセスに対応します。車は緑、歩行者は赤、歩行者の緑のフェーズを経て、最終的に車が再び運転できるようになります。

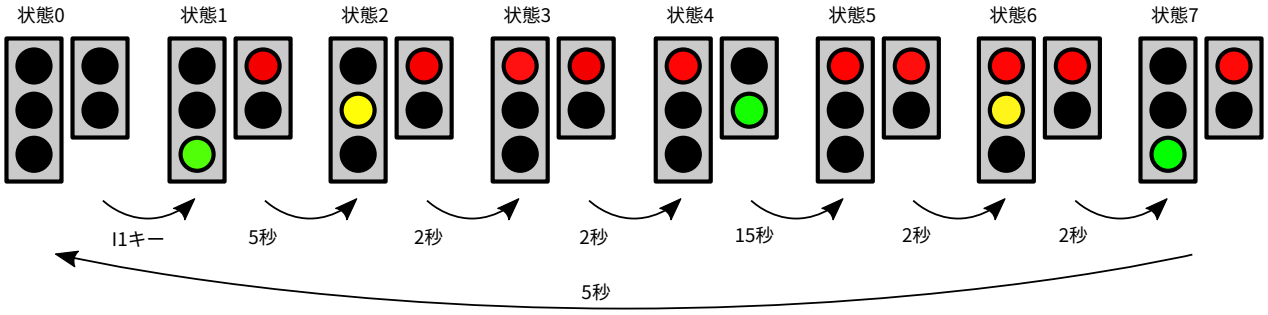


図7.7：信号機の状態

7.4.1ステートマシン

ソフトウェアでの信号機制御の単純で明白な実装は、信号機の状態の経過を直接追跡するプログラムで構成されます。プログラムシーケンスは、ユーザーがキーが押されるのを待つか、時間が経過すると停止し、対応するイベントが発生すると続行します。これは、信号機の最初の2つの状態の例として以下に示されています。

```
空所 ループ （） {
   //キーが押されるのを待ちます
   その間 （！ ftduino。input_get （（ボタン））{};


   //信号が点灯し、車は緑、歩行者は赤 cars_green （）；

   pedestrians_red （）；
   遅れ （（CARS_GREEN_PHASE）；

   //車が黄色くなる cars_yellow
    （）；
   遅れ （（YELLOW_PHASE）；

   //..。
}
```

このプログラムは短くて理解しやすいです。これは実際には良いことですが、大きな欠点が1つあります。イベントを待機している間、プログラムシーケンス全体が停止し、他のことを並行して実行できなくなります。

the PedestrianLightたとえば、Sketchは、の組み込みLEDも使用する必要があります。 ftドゥイーノ 閃光。これは、実際の信号機の状態に関係なく、中断することなく発生するはずです。

解決策はステートマシンです。

```
//ループ関数が何度も呼び出されます 空所 ループ （） {

   //次のライトチェンジイベントの時間 static unsigned long next_event = 0;
   // Ameplの現在のステータス 静的文字 州 = 0;



   //内部発光ダイオードは1秒に1回点滅する必要があります static unsigned long
   flash_timer = 0; もしも （（ミリス （） > flash_timer + 10）

      digitalWrite （（LED_BUILTIN 、 低い）; もしも
    （（ミリス （） > flash_timer + 1000）{
      digitalWrite （（LED_BUILTIN 、 高い）；
      flash_timer = ミリス （）；
   }

   //状態0（信号がオフ）の歩行者が//ボタンを押したかどうかをテストします

   もしも （（（州 == 0）&&（ftduino。input_get （（ボタン））））
```

```
        州 = 1;                //はい->状態1に変更

    もしも （（州 > 0）{

        //設定時間が経過したかどうかをテストします もしも （（ミリス （） >
        next_event）{
            スイッチ （（州）{

                //信号が状態1に変わります：車は緑、歩行者は赤 場合
                        1：{
                    //   ランプをつける
                    cars_green （）;
                    pedestrians_red （）;
                    //次のイベントの時間を設定します next_event = ミリス （）
                    + CARS_GREEN_PHASE; //次のイベントの状態を設定します

                    州 ++;              //「state = state +1」の短い表記 壊す;

                }

                //信号が状態2に変わります：車は黄色、歩行者は赤です 場合 2：{

                    cars_yellow （）;
                    next_event = ミリス （） + YELLOW_PHASE; 州 ++;

                    壊す;
                }

                //信号が状態3に変わります：車は赤、歩行者は赤 場合
                        3：{
                    //   。。。
                    壊す;
                }

                //..。
            }
        }
    }
}
```

このリストはもっと複雑です。ただし、どの時点でもアクティブなメンテナンスが行われないという大きな利点があります。代わりに、プログラムは実行を継続します。個々の信号機フェーズを引き続き実行できるようにするために、2つの変数がメモリとして作成されます（next_event と 州）。 ここでは、信号機のステータスと、このステータスを維持する期間が永続的に記録されます。

このようにして、LEDを完全に独立して点滅させ、必要に応じて他の制御タスクを実行することができます。

大きな出費は、 ftドゥイーノ たとえば、PCやスマートフォンで一般的なように、複数のプログラム部分（いわゆるプロセスまたはスレッド）を同時に提供できる独自のオペレーティングシステムはありません。

シンプルなものの大きなメリット ftドゥイーノアプローチは、その正確な予測可能性にあります。オペレーティングシステムがバックグラウンドで予期せずビジー状態になり、プログラムの実行が停止した場合、PCまたはスマートフォンから誰もが知っています。たとえば、モーターが特定の位置に到達したときにモーターが十分に速く停止しない場合、ユーザーインターフェイスでのみ煩わしいことが、開ループおよび閉ループの制御タスクで簡単に問題になる可能性があります。このため、より単純なものができますftドゥイーノ Linuxオペレーティングシステムによって駆動されるTXTコントローラーやRaspberryPiなどよりも、多くのことに迅速かつ予測どおりに反応します。存在しないオペレーティングシステムのもう1つのプラスの効果は、システムの高速起動です。Aftドゥイーノ 電源を入れた直後に完全に機能し、デバイスがタスクを実行する前にオペレーティングシステムが起動するのを待つ必要はありません。

商業環境においてさえ、このような単純なシステムが常に必要とされる主な理由は、システムの起動が速く、動作が簡単に予測できることです。 ftドゥイーノ ハードウェアの価格が下がる複雑なオペレーティングシステムベースのソリューションを使用することも、ますます単純化するデバイスで可能です。

7.5クラシック2Dプロッタ

Schertechnikプロッタ30571[4位]1985年からは、schertechnikの最初のコンピューティングモデルの1つでした。X軸とY軸の動きは、6ボルトのステッピングモーターによって実現され、ペンは磁石によって上下されました。当時のコンピューティングインターフェースの8つの出力を介して制御されていたため、ペンをモーターの1つと巧みに組み合わせて、合計8つの出力を処理しました。

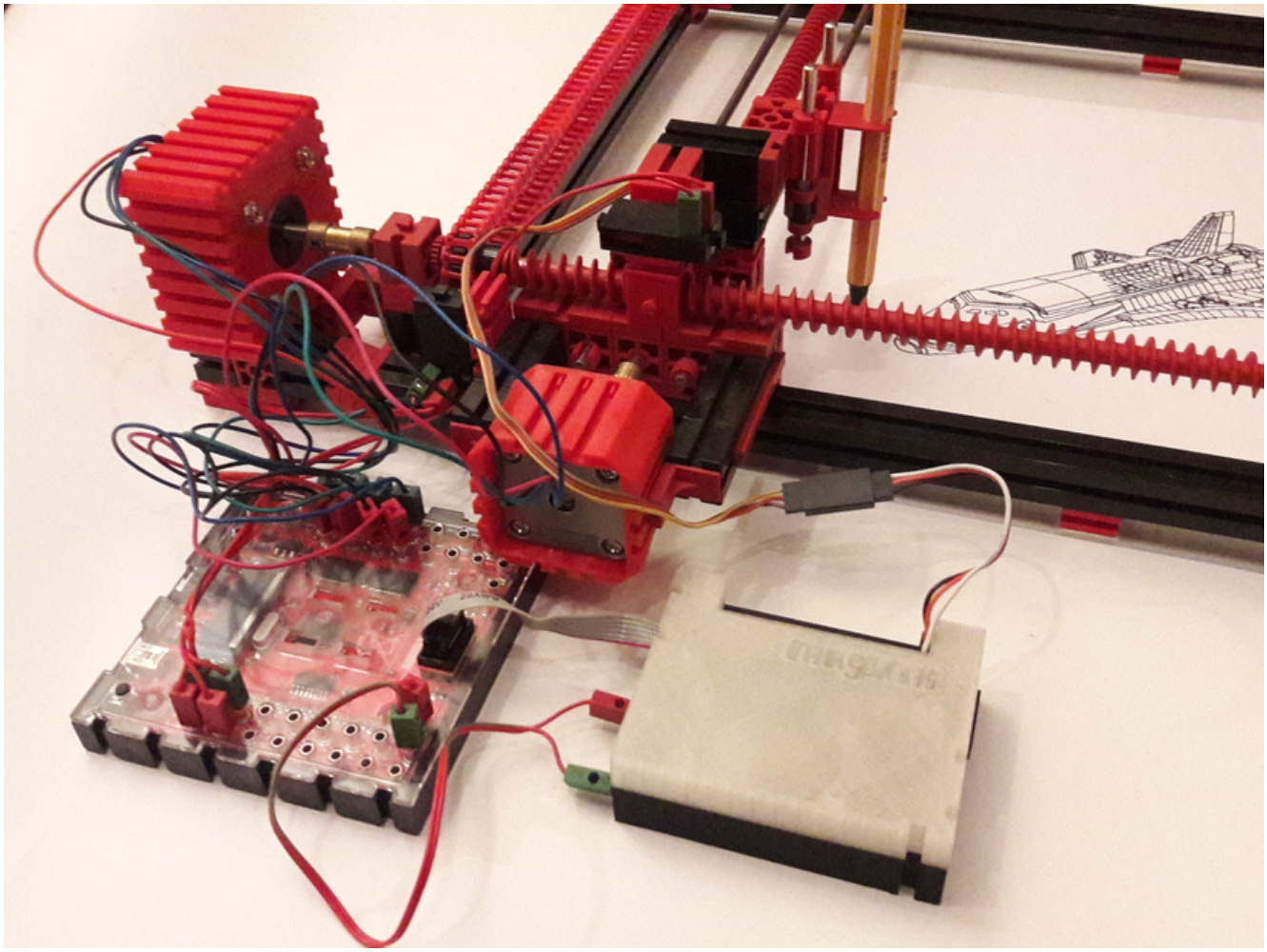


図7.8：ピンはschertechnikサーボによって持ち上げられます

ここで紹介するモデルは同じ寸法で、非常によく似た基本的な仕組みを使用しています。2つのステッピングモーターも使用されますが、9ボルトでもプロッター動作に十分な電力を発生する12ボルトバージョンです。ステッピングモーターは、セクション6.4で説明されているように制御および接続されます。

このモデルでは、Schertechnikサーボによってペンが上下します。ここでは、セクション6.5のアプローチを使用できます。この場合、私は。2Cサーボシールド[5]使用済み。それはもう少し扱いに くいですが、本当の私を許可します。2C操作とOLEDでそうですftドゥイーノ 互換性。

下の一致するスケッチ ファイル例 。FtduinoSimple 。プロッタ 非常にシンプルに保たれ、優れた機能を備えています 改善の可能性。USB / COMインターフェースで標準のHP-GLコマンドを受け入れます[6日] したがって、多くのPCプログラムから直接アドレス指定できます。

[4位] schertechnikデータベース： https://ft-datenbank.de/tickets?fulltext=30571
[5] Thingiverseのサーボシールド： https://www.thingiverse.com/thing:3316758
[6日] ウィキペディアのHP-GL： https://de.wikipedia.org/wiki/Hewlett-Packard_Graphics_Language

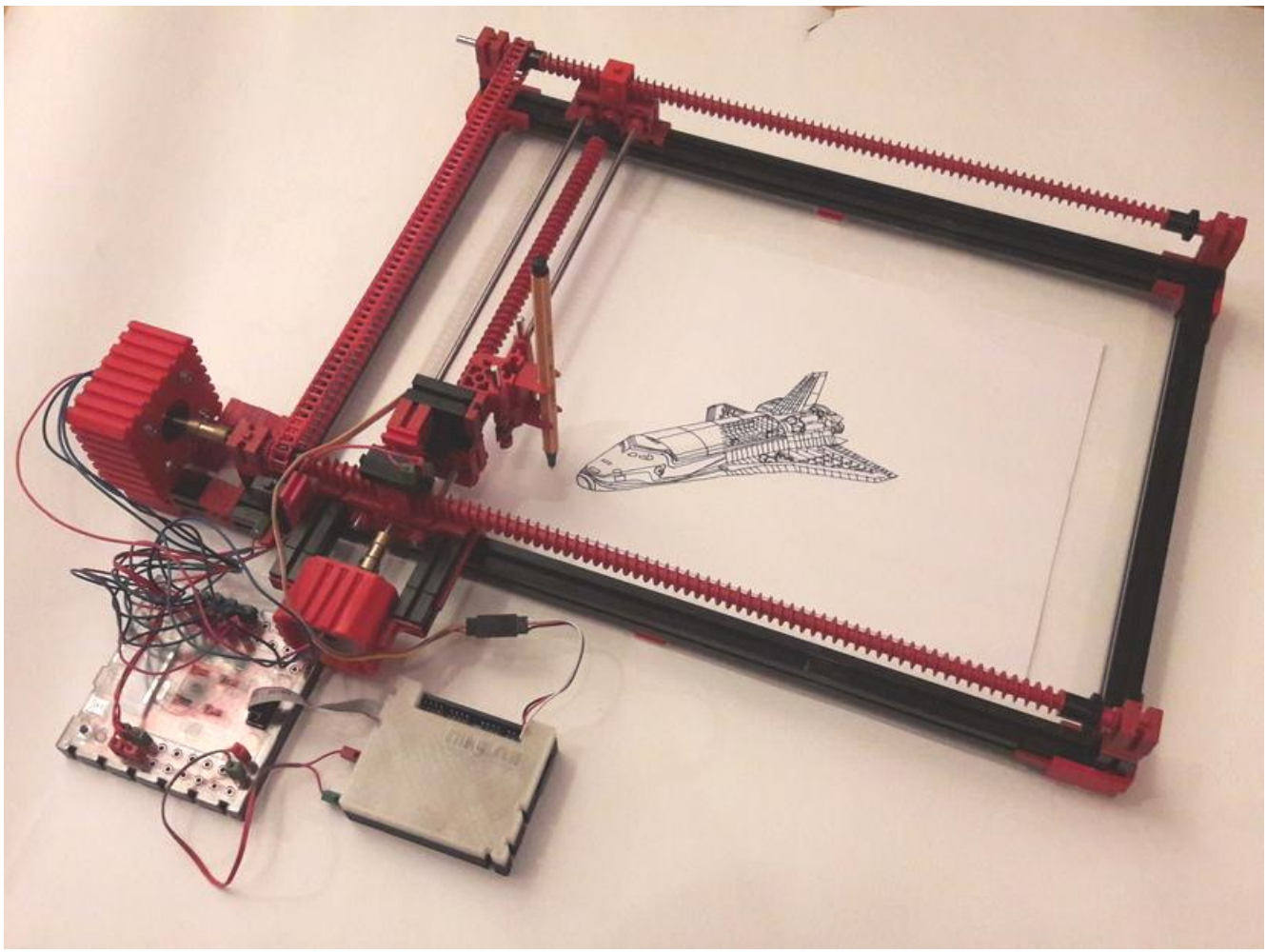


図7.9：設計は1985年のプロッタに大きく基づいています

第8章

コミュニティプロジェクト

the ftドゥイーノ 本当のコミュニティプロジェクトです。これは、schertechnikコミュニティからのアイデアに基づいているため、既存のコミュニティプロジェクトにうまく統合されます。商用製品は多くの場合、前任者と競合しており、顧客には何よりも新しいものを提供する必要がありますが、コミュニティプロジェクトでは、古い技術的に競合するシステムをより簡単に統合できます。

the ftドゥイーノ セクション6.13.5で説明されているように、Iを介してschertechnikTXTコントローラーで使用できます。2ペアC。いわゆるコミュニティファームウェアはTXTに付属しています[1]使用するため。接続するための対応するプログラム ftドゥイーノ 私あたり。2Cもそこにあります[2]。

8.1 ftduino_direct： ftドゥイーノ-USB経由でTXTおよびTX-Piに接続

USBを介したPC、TXT、またはRaspberry-Piへの接続は、よりシンプルで堅牢です。コミュニティは、この目的のためにスケッチと適切なPythonライブラリを提供します[3]。

スケッチの助けを借りて、 ftドゥイーノ その接続は、USB経由で接続された上位レベルのデバイスで利用できます。Pythonライブラリを上位デバイスで使用して、の入力と出力にアクセスできます。ftドゥイーノ アクセスするために。

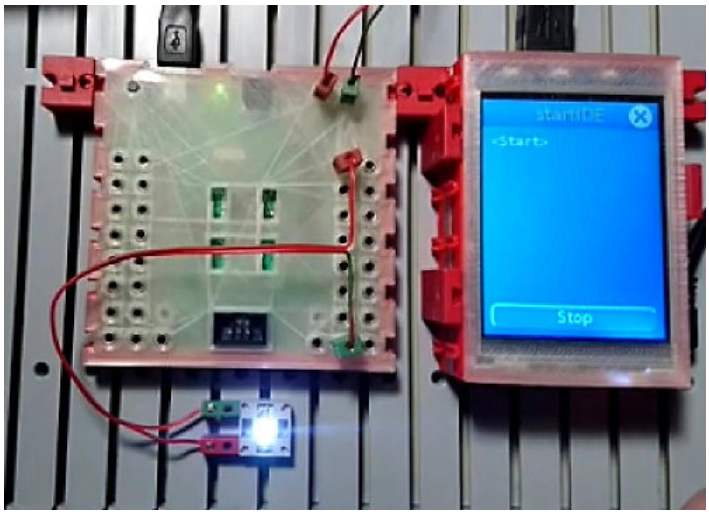


図8.1： ftドゥイーノ RaspberryPiのUSB経由

1 schertechnikTXTコミュニティファームウェア http://cfw.ftcommunity.de/ftcommunity-TXT
2ftDuino I2CFWの場合はC https://github.com/harbaum/cfw-apps/tree/master/packages/ftDuinoI2C
3ftduino_direct-スケッチ https://github.com/PeterDHabermehl/ftduino_direct

このライブラリの助けを借りて、コミュニティファームウェア用の既存のPythonプログラムは、 ftドゥイーノ 拡大。これは、Raspberry Piなどのデバイスで特に興味深いものです。これらのデバイスには、せん断技術センサーやアクチュエーターへの固有のインターフェイスがないためです。

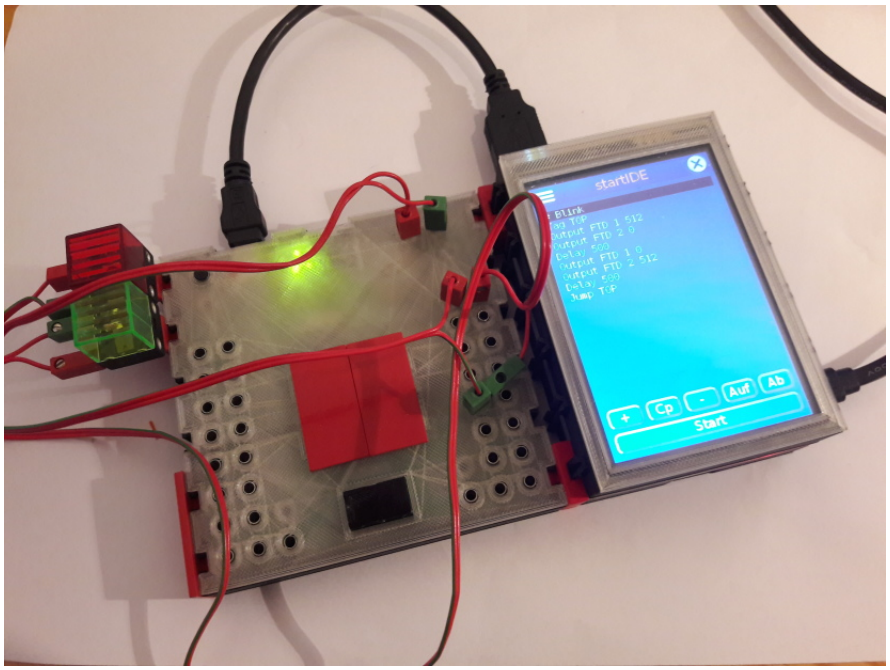


図8.2： startIDE TX-Piでアクセス可能 ftドゥイーノ

のようなプログラム startIDE[4位] と ブリックリー[5] の助けを借りてすることができます ftduino_direct に ftドゥイーノ の Schertechnik互換接続にアクセスして使用する ftドゥイーノ 使用します。

8.2 ftDuinIO： ftドゥイーノ-TXTおよびTX-Piの制御アプリ

のインストール ftduino_direct-上のスケッチ ftドゥイーノ Arduino IDEを介して通常どおり実行できますが、従来のPCが必要です。PCから完全に独立するために、ftDuinIO-コミュニティファームウェア用に設計されたアプリ。

図8.3： ftDuinIO-アプリ

このアプリは、schertechnik-TXTだけでなく、TX-Piに構成されたRaspberry-Piでも操作でき、スケッチを実行できます。 ftドゥイーノ ロードするだけでなく、 ftduino_direct-機能をテストします。

[4]位startIDE： https://forum.ftcommunity.de/viewtopic.php?f=33&t=4297
[5]レンガ： https://cfw.ftcommunity.de/ftcommunity-TXT/de/programming/brickly/

8.3 レンガのプラグイン： グラッシュ ftドゥイーノ-Bricklyでのプログラミング

ブリックリー[6日] GoogleのBlocklyにあるものです[7日] ベースのグラフィックプログラミング環境。

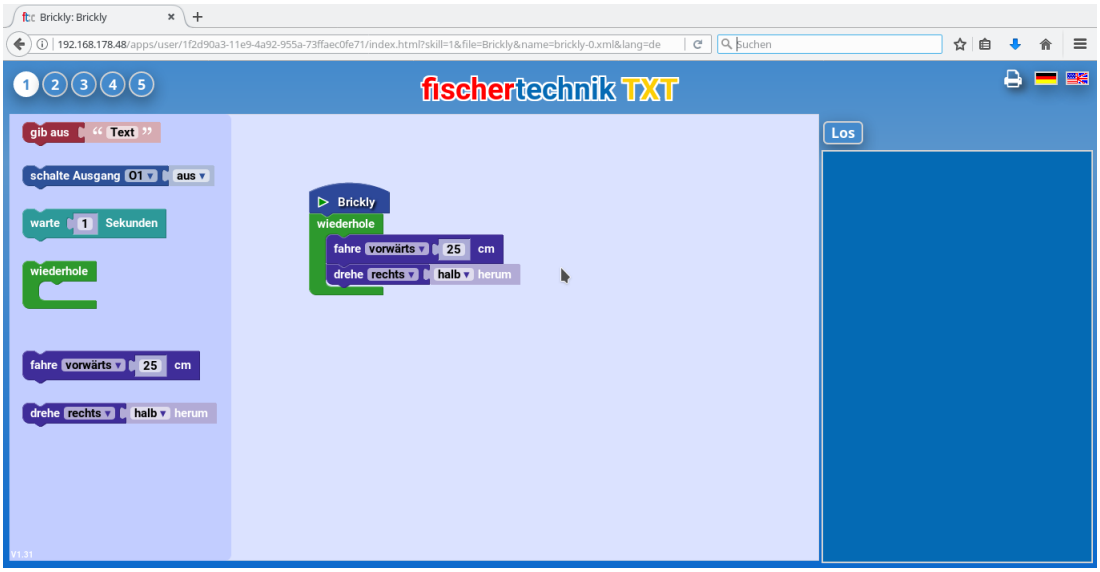


図8.4：Bricklyプログラミングインターフェイス

BricklyはschertechnikTXTコントローラー用に作成されており、入力と出力を使用するためのすべてが備わっています。 Brickly自体を実行するには、強力なコントローラーが必要です。これは、schertechnikTXTコントローラーまたはRaspberryPiの場合があります。操作とプログラミングは、WLAN経由で接続されたスマートフォンまたはPCを使用してWebブラウザで実行されます。

the ftドゥイーノ それ自体は、Bricklyを実行するのに十分なほど強力ではありません。また、必要なWiFi接続を提供します ftドゥイーノ ではない。

代わりに、Bricklyは1つをschertechnik-TXTコントローラーまたはRaspberry-Pi（TX-Pi）に接続できます。 ftドゥイーノ 向かう。 Bricklyは、このためにセクション8.1で説明されているものを使用しますftduino_direct-繋がり。

Theftドゥイーノ-Bricklyのプラグイン[8日]ブラウザインターフェイスで直接既存のBricklyインストールにインストールできます。

図8.5：Bricklyプラグインのインストール

その後、 ftドゥイーノ 入力と出力は、Bricklyプログラムで直接使用できます。Raspberry Piはschertechnikの世界を開き、TXTコントローラーは20の追加入力と8つの出力で拡張できます。

Bricklyは、セクション6.18.4で説明されているliteバリアントにも存在します。これにより、ブラウザからUSB経由で接続されたブラウザに直接アクセスできます。ftドゥイーノ 追加のTXTコントローラーやRaspberryPiは必要ありません。

6日レンガのような指示 https://github.com/EstherMi/ft-brickly-userguide/blob/master/de/brickly/index.md
7日Google-Blockly https://developers.google.com/blockly/
8日ブリックリーftドゥイーノ-プラグイン https://github.com/harbaum/brickly-plugins#ftduino-io ftduinoxml

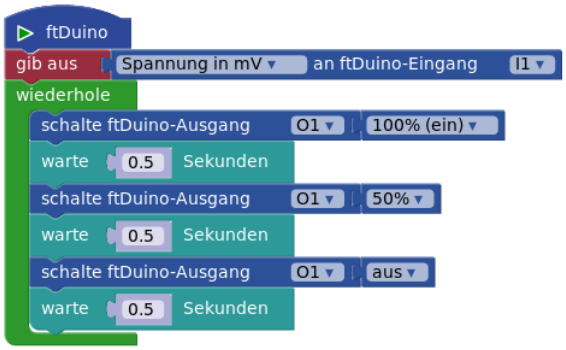
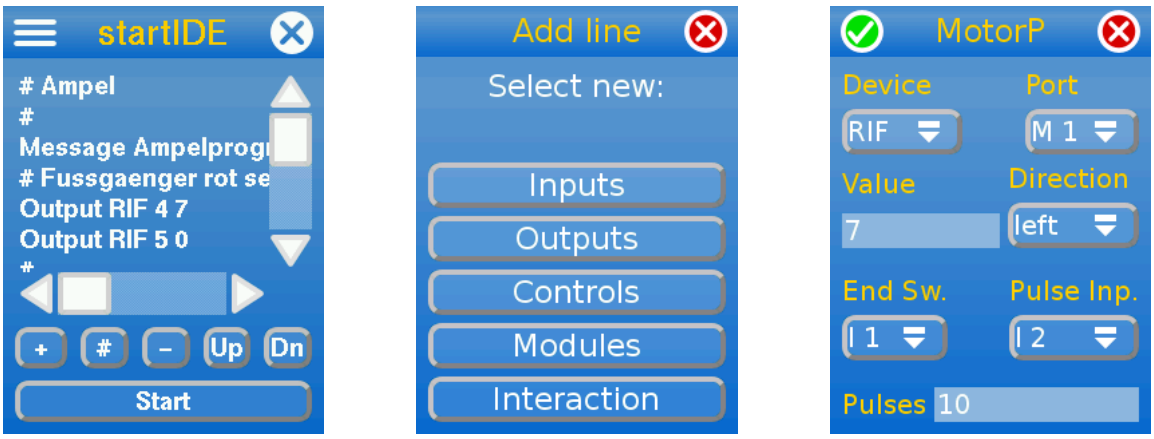


図8.6：を制御するためのBricklyプログラム ftドゥイーノ

8.4 startIDE： TX-PiまたはTXTで直接プログラミング

schertechnik TXTおよび-TXコントローラーの通常のRoboProプログラミング環境では、プログラミングにWindowsPCが必要です。より現代的なBrickly（セクション8.3を参照）でさえ、プログラム開発のために外部デバイスに依存しています。

StartIDE[9]一方、schertechnikTXTコントローラーまたはRaspberryPiの小さなタッチスクリーンでも、デバイス上で直接プログラムを作成できるように設計されています。



（a）メイン画面　（b）機能の選択　（c）IO構成

図8.7：StartIDEの表面

the startIDE TXTの独自の接続だけでなく、USBを介して外部に接続される一連のインターフェイス全体の接続もサポートします。 ftドゥイーノ。これを行うには、それ自体を利用しますstartIDE セクション8.1ですでに提示されているものftduino_direct-繋がり。以来startIDE また、TX-PiまたはRaspberry-Piで実行され、Schertechnikモデルでの使用に必要な接続を提供します。

TX-Pi（またはRaspberry-Pi）チーム、 ftドゥイーノ と startIDE これにより、SchertechnikモデルをPCやスマートフォンなしでプログラムできます。

詳細な取扱説明書[10] the startIDE 使用方法の詳細が含まれています ftドゥイーノ。

[9] startIDE-ホームページ： https://github.com/PeterDHabermehl/startIDE/
[10] startIDE-マニュアル： https://github.com/PeterDHabermehl/startIDE/blob/master/ddoc/Manual_160_de.pdf

8.5フィートエクステンダー：I。2C拡張子

ft-extenderはIのようなものです。2セクション6.13.6からのC-Expanderは、Iを許可するデバイスです。2のCバス ftドゥイーノ 拡張して、さまざまなデバイスを同時に接続および結合します。

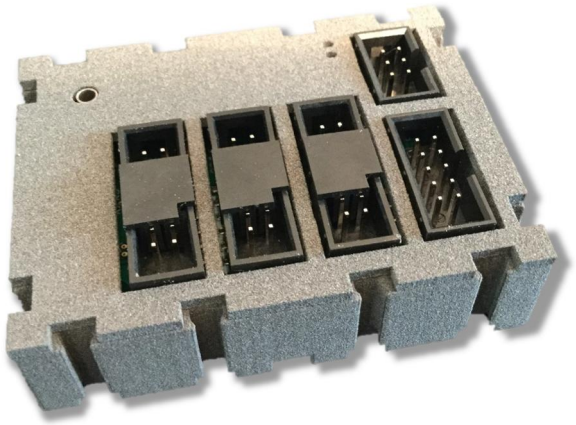


図8.8：フィートエクステンダー

Iの機能について2C-Expanderに加えて、ft-Extenderは接続されたデバイスに追加の電源を提供します。ftエクステンダーの助けを借りてftドゥイーノ 接続されているセンサーは、Iとは異なります。2C-からではないエキスパンダー ftドゥイーノ ft-Extender自体によって供給されます。一方で、これはわずかに高い電流が利用可能であることを意味します。しかし何よりも、5ボルトの隣にはftドゥイーノ 電源、3.3ボルトの電源もご利用いただけます。したがって、ft-ExtenderはIよりも柔軟性があります。2Cエキスパンダー。

ftエクステンダーの詳細については、次のURLをご覧ください。 https://github.com/elektrofuzzis/ftExtender、具体的には https://github.com/elektrofuzzis/ftExtender/blob/master/Handbuch_ft-Extender.pdf。

8.6 Arduino（S4A）のスクラッチ

傷11日 BlocklyおよびBricklyに匹敵するグラフィックプログラミング環境です（セクション6.18.4を参照）。

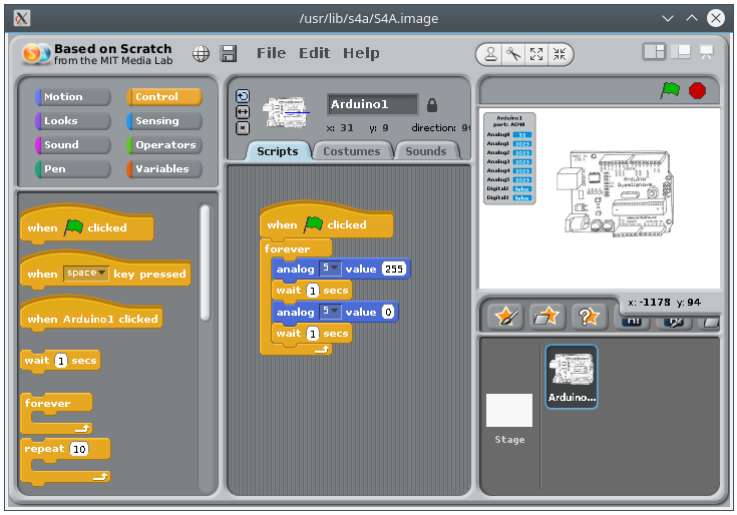


図8.9：S4Aのメイン画面

11日傷： https://scratch.mit.edu/

Scratchは、純粋なシミュレーション環境として開発されました。実際のハードウェアとの相互作用は意図されていませんでした。Arduinoのスクラッチ12日 Arduinosに対処するScratchimの機能を拡張します。この目的のために、Arduinoに特別なスケッチがインストールされています。このように準備されたArduinoは、S4Aによって自動的に認識され、統合されます。

以来 ftドゥイーノ 特別な入力と出力があるため、S4Aスケッチで直接制御することはできません。これの代わりに
で見つけることができます ftドゥイーノ-互換性のあるスケッチを下にインストールします ファイル例 。Ftduino 。S4AFirmware16 。A
このスケッチを装備 ftドゥイーノ S4Aによって自動的に認識され、統合されます。

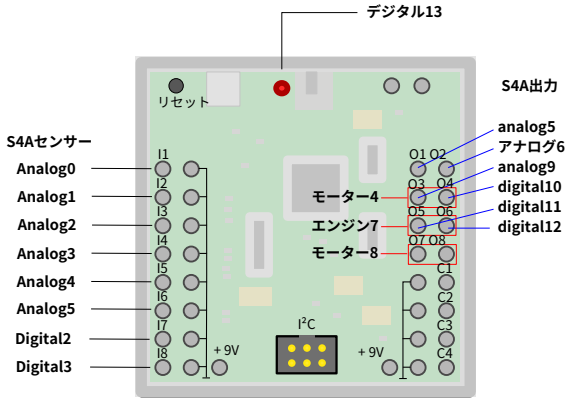


図8.10： ftドゥイーノ-S4Aでの接続

S4Aは、入力と出力の名前を持つ通常のArduino名に密接に基づいているため、 ftドゥイーノ-通常の指定が使用されます。図8.10は、ftドゥイーノ S4Aにどのような名前で表示されますか。

8.6.1インストール

上のS4A ftドゥイーノ S4Aの通常のインストールが必要です。インストールはS4Aホームページの下にあります http://s4a.cat/ 説明された。

そこで指定された元のスケッチの代わりに、スケッチは単に上にある必要 ファイル例 。Ftduino 。S4AFirmware16 -
があります ftドゥイーノ 課金されます。

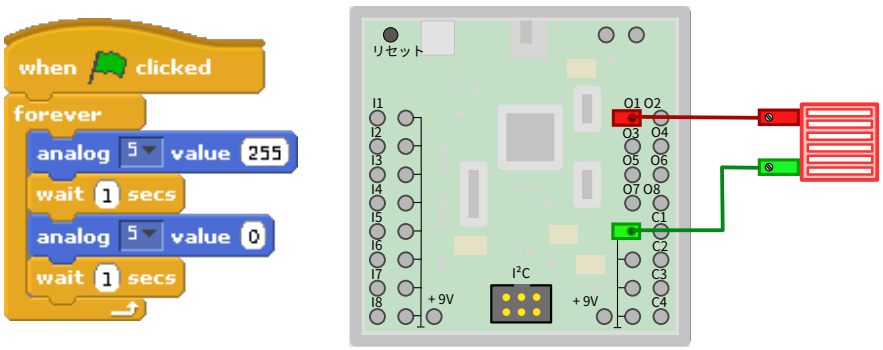


図8.11：ランプを点滅させる単純なS4Aの例

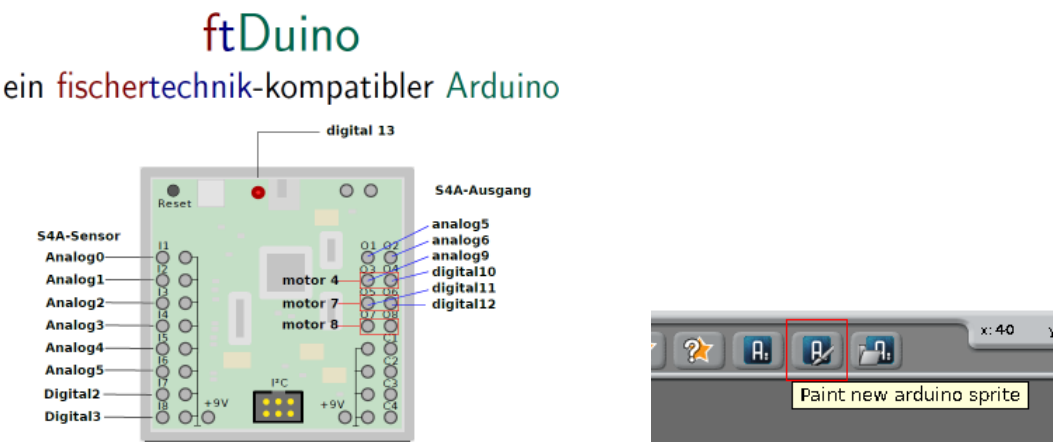
図8.11に、プログラム例と対応するケーブルを示します。アナログ入力の場合、ftドゥイーノ 0から0までの値の範囲 65 。535 kΩ Scratchが0〜1023の値で動作している間に測定できます。S4Aの1023の値はこれに対応します65。535 kΩ または上のオープンエントランス ftドゥイーノ。S4Aの値0は、に対応します。0 Ω または閉じた入り口。

12日S4A： http://s4a.cat/

8.6.2S4Aでのピン割り当ての表現

S4AはArduinoで使用するように設計されているため、プログラムは最初に右上の領域にArduinoの割り当てを示します。それはそれでの作業を容易にしますftドゥイーノ S4Aの下で、 ftドゥイーノ 示されています。

対応するいわゆるスプライトファイルは、 https://harbaum.github.io/ftduino/www/images/s4a_sprite.png 利用可能。Arduino画像のディスプレイのすぐ下に、新しいスプライトを選択するためのアイコンがあります。



(a) ftドゥイーノ-S4Aのスプライト (b) 新しいスプライトの選択

図8.12：新しいスプライトのインストール

このアイコンをクリックすると、次のペイントダイアログで新しいスプライトを描画できます。上記のファイルを使用するには、最初に自分のPCにファイルをロードする必要があります。の中に ペイントダイアログ。[OK]の[インポート]ボタンで確認できます。
S3Aをインポートできます。新しいスプライト画像はで終わる必要があります

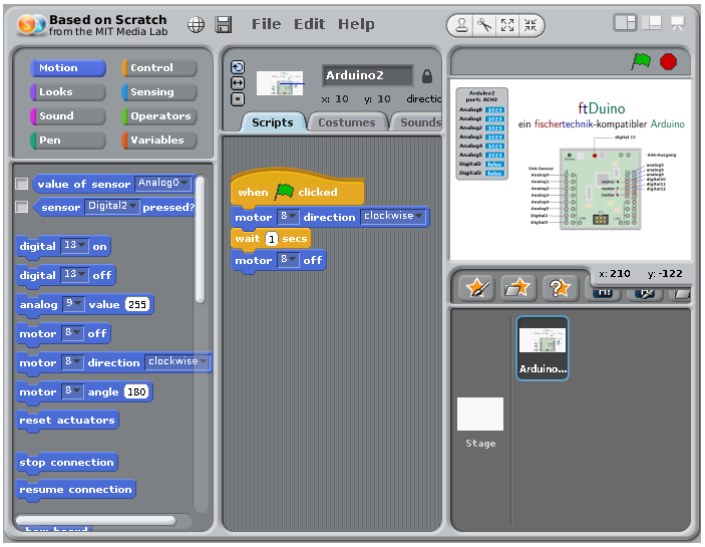


図8.13：の表示 ftドゥイーノ-S4Aのスプライト

の接続割り当て ftドゥイーノ S4Aの右上に現在のプロジェクトの永続的に表示されるようになりました。

8.7 マインクラフト と ftドゥイーノ：コンピュータゲームと現実の世界が出会う

教訓分野に端を発するScratchなどのグラフィックモジュラープログラムキットに加えて、一見予想外の方向から来るグラフィックプログラミングシステムもあります。

この人気のある例はMinecraftです。Redstoneと対応するコンポーネントを使用すると、このゲームは仮想Minecraftの世界にケーブルを敷設し、内部のゲームアクチュエータとセンサー、およびロジック要素を接続する可能性を提供します。

図8.14： ftドゥイーノ-MinecraftのMod

Minecraftmodの ftドゥイーノ 接続されたPCの接続でMinecraftの内部世界を接続するブロックでMinecraftを拡張します ftドゥイーノ 接続します。

8.7.1のインストール ftドゥイーノ-モッド

いわゆる ftドゥイーノ-MinecraftのModはForge1.12.2に基づいています [13日] これらのMinecraftバージョンとForgeバージョンをインストールする必要があります。のインストールftドゥイーノ-Modは他のModと何ら変わりはありません。

the ftドゥイーノ-Mod自体は https://harbaum.github.io/ftduino/minecraft/ftduino-0.0.3.jar ダウンロードしました。Windows、Linux、MacOSPCに適しています。通常の手順に従って、関連するMinecraft modディレクトリにコピーされ、次にMinecraftが起動されたときに自動的にロードされます。

8.7.2の準備 ftドゥイーノ

the ftドゥイーノ と一緒に行かなければなりません IoServerスケッチを提供することができます。これはArduinoIDEの下にあります ファイル例 。 WebUSB 。IoServer 見つけられる。この場合、通常のボードを使用する必要がありますftDuino 選択され、 いいえ ftDuino （WebUSB）、 MinecraftはWebアプリケーションではないためです。

IoServerSketchが適切に装備されているので ftドゥイーノ■内部OLEDディスプレイも使用するため、AdafruitGFXライブラリをインストールする必要があります。必要に応じて、ArduinoIDEのライブラリメニューから数回クリックするだけでこれを実行できます。内部表示は絶対に必要というわけではなく、スケッチはすべての人に機能しますftドゥイーノ 表示なし。

[13日] Forge 1.12.2： http：// les.minecraftforge.net/maven/net/minecraftforge/forge/index_1.12.2.html

8.7.3の使用 ftドゥイーノ Minecraftで

だった ftドゥイーノ-Modが正常に統合されると、図8.15に示す追加のブロックが、レッドストーン領域のMinecraftのクリエイティブモードで使用できるようになります。

現時点では、入力用の純粋なデジタルブロックがあります I1 それまで I8、 出口 O1 それまで O8 およびの内部発光ダイオード ftドゥイーノ 処分する。

（a）入口 I1-I8 （b）終了 O1-O8 （c）内部LED

図8.15： ftドゥイーノ-Minecraftのブロック

入力ブロック（の） レッドストーンを入り口の1つに接続します I1 それまで I8 の ftドゥイーノ 処分する。ブロックの前面を右クリックすると、入力を選択できます。これは、8つのソケットの1つに差し込まれている緑色のプラグによって象徴されています。その右側の黄色の点滅は、選択した入力がアクティブになったことを示します。たとえば、ftドゥイーノ 接続されているスイッチが閉じています。図に示す入力I1 現在アクティブではありません。の同じ入力を持つ複数の入力ブロックftドゥイーノ が関連付けられると、すべてのブロックが対応する入力の信号を同時に受信します ftドゥイーノ。

出力ブロック（アウト） Redstoneを出力の1つに接続します O1 それまで O8 の ftドゥイーノ ここ。この場合、それは赤いプラグで識別されます。図に示されているブロックは、アクティブなレッドストーンによって制御されています。したがって、これはアクティブであり、この場合は出力を制御しますO2 そしてそれを状態に切り替えます こんにちは 出口の間に1つ O2 アース（-）接続ランプが点灯します。これは、稲妻の記号で示されます。一度に存在できるブロックは1つだけですftドゥイーノ-出力が割り当てられます。そうしないと、矛盾する信号が同じ出力に送信される可能性があります。

LEDブロックは、の内部発光ダイオードを制御します ftドゥイーノ の上。このブロックは、制御の矛盾を防ぐために、Minecraftの世界に1回だけ存在できます。

たとえば、図8.14の例の世界では、単純な交互のフラッシャーが右奥に表示されています。出口のあるもののレッドストーントーチO2 関連する ftドゥイーノ-出力ブロックは、このブロックの前にある信号を無効にするインバーターを形成します。信号はその前にあるレッドストーンリピーターにルーティングされます。これにより、信号がわずかに遅延し、出力のあるリピーターに戻ります。O2 関連する ftドゥイーノ-出力ブロックを転送します。信号の継続的な遅延と否定により、約1秒ごとに点滅します。別の出力ブロック、今回はO1 関連付けられているは、反転信号に接続されています。出口へO1 と O2 したがって、接続されたランプは交互に点滅します。写真のさらに左側には、上の入り口がありますftドゥイーノ-入力の上にあるレッドストーンランプを含む、実装された入力ブロック I7 の ftドゥイーノ 向かう。

第9章

ライブラリ

のすべての接続 ftドゥイーノ Arduinoスケッチからの制御。ただし、さまざまな入力と出力を制御するためのすべてのコードをスケッチ自体に実装する必要があるため、この手順には多くの経験が必要であり、比較的複雑なArduinoスケッチになります。

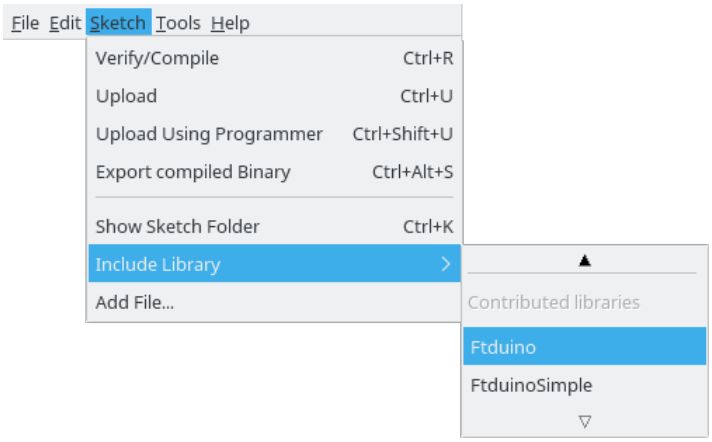


図9.1： ftドゥイーノArduinoIDEのライブラリ

the ftドゥイーノ したがって、いわゆるライブラリを持ってきます。これらは、の入力と出力を制御するための既製のルーチンを含むコードコレクションです。ftドゥイーノ含まれています。これにより、実際のArduinoスケッチがはるかに単純で短くなり、何よりも、プログラマーはいくつかの使いやすいルーチンにアクセスするだけでよいため、ハードウェアの側面を完全に理解する必要はありません。ftドゥイーノ-ハードウェア使用。ライブラリは、の一部として含まれていますftドゥイーノ -ArduinoIDEへのインストールは自動的にインストールされます。これらのライブラリの更新は、Arduino IDEによって自動的に検出され、更新のために提供されます。

それでも、もちろん直接アクセスは可能です。上級プログラマーは、すべてのライブラリをバイパスしてハードウェアに直接アクセスできます。ライブラリのコードも無料で入手できます[1]ユーザー自身が必要に応じて拡張や改善を行えるようにします。

利用可能な2つのライブラリがあります ftドゥイーノ 向かう。theFtduinoSimple-非常にシンプルに保たれ、非常に基本的な関数のセットのみを提供するライブラリ Ftduino非常に複雑で、信号の入出力に幅広い機能を提供するライブラリ。

9.0.1ポートの定義と定数

のさまざまなポートの場合 ftドゥイーノ 次の2つのライブラリでは同じ定数が使用されています。たとえば、8つの入力は定数で表されますFtduino :: I1 それまで Ftduino :: I8 説明された。もしそれでも

[1] https://github.com/harbaum/ftduino/tree/master/ftduino/libraries

これらの定数の後ろに値0〜7を非表示にします。可能であれば、常に定数を使用する必要があります。定数の後ろに隠されている値を変更する必要が生じた場合でも、対応するスキットは引き続き機能します。

プログラマーは、これらの定数が常に昇順で連続しているという事実に頼ることができます。たとえば、すべてのポートでカウントできます。

```
// ftDuinoのすべての入力をループします
にとって（（uint8_tポート=Ftduino：：：I1;ポート<=Ftduino：：：I8;ポート++）{
  /*ポートは次々にすべてのポートI1からI8を参照します*/do_something（（ポート））;

}
```

同等のバージョンは次のとおりです。

```
にとって（（uint8_t i=0;私<8;私++）{
  do_something（（Ftduino：：：I1+私））;
}
```

9.1 FtduinoSimple

the FtduinoSimple-ライブラリは非常に単純なライブラリです。単純なデジタル値（オン/オフ）のクエリと出力のオンとオフの切り替えのみが可能です。アナログの電圧と抵抗を読み込んだり、出力を可変値に切り替えたりすることはできません。

の利点 FtduinoSimple-ライブラリは次のとおりです。

シンプルさ ライブラリには、非常に使いやすい関数がいくつかあります。間違いはほとんどありません。

低メモリ要件 ライブラリは、フラッシュまたはRAMメモリをほとんど使用しません。 ftドゥイーノ。ほとんどすべてのメモリ
実際のスケッチに使用できます。

副作用はありません ライブラリは、ATmega32u4コントローラーの他の内部ハードウェアを使用せず、実装します
たとえば、独自の割り込みハンドラを管理しません。内部ハードウェア全体（タイマー、カウンター、割り込みなど）は、独自のスケッチに使用でき、ATmega32u4ハードウェアに直接アクセスするときに予期しない影響を考慮する必要はありません。

に FtduinoSimple-ライブラリを使用するには、対応するインクルードラインをスケッチの先頭に挿入する必要があります。

```
# 含む <FtduinoSimple.h>
```

9.1.1スケッチでの使用

the FtduinoSimple-ライブラリは、センサーとアクチュエーターの制御の多くの詳細をユーザーから隠し、入力と出力を数行の簡単なコードでスケッチで使用できるようにします。

次の例は、を使用した出力の定期的な切り替えを示しています。 FtduinoSimple-としょうかん：

```
1  # 含む <FtduinoSimple.h>
2
3  空所 設定 () {
4位   //初期化は必要ありません
5  }
6日
7日 空所 ループ () {
8日   //出力O1をオンにします ftduino。output_set
9      （（Ftduino：：：O1、遅れ（1000）；      Ftduino：：：こんにちは）；
10
11日   //出力O1をオフにします ftduino。output_set
12日   （（Ftduino：：：O1、遅れ（1000）；      Ftduino：：：LO）；
13日
14日 }
```

9行目と12行目で出力 O1 の ftドゥイーノ オンまたはオフに切り替えました。の最初のパラメータoutput_set（）- 関数は、切り替えられる出力を指定します。これにより、出力をオンにするかオフにするかが決定されます。

注意： 出力を使用できるようにするために、 ftドゥイーノ もちろん9ボルトで供給することができます。

の入力 ftドゥイーノ で非常に使いやすいです FtduinoSimple-クエリライブラリ：

```
1   # 含む <FtduinoSimple.h>
2
3   空所 設定 () {
4位    //初期化は必要ありません
5   }
6日
7日 空所 ループ () {
8日    //入力I1を読み込みます
9      もしも ( (ftduino。input_get ( (Ftduino ：：：I1) )        {{
10        //出力O1をオンにします ftduino。output_set
11日       ( (Ftduino ：：：O1 、 それ以外 {{          Ftduino ：：：こんにちは) ;
12日   }
13日      //出力O1をオフにします ftduino。output_set
14日       ( (Ftduino ：：：O1 、                    Ftduino ：：：LO) ;
15日   }
16  }
```

9行目では、aの状態が入力に送信されます I1 接続されたスイッチ。押されている（閉じている）か開いているかに応じて、出力は11行目または14行目に表示されます。O1 オンまたはオフ。

9.1.2 bool input_get（uint8_t ch）

この関数は、入力のステータスを読み取ります ch 1。許容値ch それは Ftduino :: I1 それまで Ftduino :: I8。 戻り値は 本当、 入力がアースに接続されている場合 false、 そうでもなければ。このようにして、たとえば、それぞれの入力とその隣の対応するアース接続の間に接続されているボタンを簡単に照会できます。

入力の評価はバックグラウンドでは行われませんが、正確にその瞬間に行われます。 input_get（）-関数が呼び出されます。特に、直前の呼び出しとは異なるポートが要求された場合input_get（） これにより、数マイクロ秒の遅延が発生します。 ftドゥイーノ-変更された入力への内部切り替えを実行する必要があります。

例

```
//入力I1のキーの状態を読み取ります もしも ( (ftduino。
input_get ( (Ftduino ：：：I1) ) {
   /* ...何かをする... */
}
```

9.1.3 bool counter_get_state（uint8_t ch）

この機能は、その動作モードに対応しています input_get（）。 しかし、 counter_get_state（） カウンター入力に適用されます。の値の範囲ch したがって、 Ftduino :: C1 それまで Ftduino :: C4。

戻り値は 本当、 入力がアースに接続されている場合 false、 そうでもなければ。

例

```
//カウンタ入力C1のキーの状態を読み取ります もしも ( (ftduino。
counter_get_state ( (Ftduino ：：：C1) ) {
   /* ...何かをする... */
}
```

9.1.4 void output_set（uint8_tポート、uint8_tモード）

機能付き output_set（） 出力を使用できます O1 それまで O8 制御されています。の値の範囲ポート したがって、 Ftduino :: O1 それまで Ftduino :: O8。

パラメータ ファッション 出力がもたらされる状態を記述します。可能な値ファッション それは Ftduino ::オフ、 出力を完全に接続解除する必要がある場合は、 Ftduino :: LO、 出力をグランドに切り替える場合、 Ftduino ::こんにちは、 出力を9ボルトに切り替える場合。

例

```
//出力O1とアースの間でランプを点灯させます ftduino。output_set（（Ftduino
：：：O1、 Ftduino ：：：こんにちは）；
```

注：出力は、次の場合にのみ使用できます。 ftドゥイーノ 9ボルト電源に接続されています（セクション1.2.5を参照）。

9.1.5 void motor_set（uint8_tポート、uint8_tモード）

関数 motor_set（） モーター出力を操作します M1 それまで M4。 モーター出力は、2つの出力を組み合わせて形成されます（M1 = O1 と O2、M2 = O3 と O4、。。。）。の値 ポート したがって、 Ftduino :: M1 それまで Ftduino :: M4。

パラメータ ファッション モーター出力がどの状態を想定するかを示します。可能な値ファッション それは Ftduino ::オフ、 エンジンを停止する必要がある場合は、 Ftduino ::左、 モーターを左に回す必要がある場合は、 Ftduino ::右、 モーターが右に曲がる必要がある場合 Ftduino ::ブレーキ、 モーターにブレーキをかけるとき。

の違い Ftduino ::オフ と Ftduino ::ブレーキ エンジンがまだ回転しているということです Ftduino ::ブレーキ 2つの接続を相互接続することにより、モーターはでアクティブにブレーキがかけられます Ftduino ::オフ オフになっているだけで、ゆっくりと期限切れになります。

例

```
//出力M1のモーターを反時計回りに動かします ftduino。motor_set
（（Ftduino ：：：M1、 Ftduino ：：：左）；
```

注：出力は、次の場合にのみ使用できます。 ftドゥイーノ 9ボルト電源に接続されています（セクション1.2.5を参照）。

9.1.6スケッチの例

を使用するためのコード例 FtduinoSimple-ライブラリはArduinoIDEのメニューにあります ファイル 例 。FtduinoSimple 。

9.2 Ftduino

the Ftduinoライブラリは、のすべての機能をカプセル化します ftドゥイーノ-ユーザーが特定の技術的な実装について心配することなく、すべての入力と出力に簡単にアクセスできるようにするハードウェア。

the Ftduinoライブラリ自体には、フラッシュメモリ、RAMメモリ、およびバックグラウンドコンピューティング能力が必要であるため、アプリケーションスケッチですべてのリソースを完全に利用できるわけではありません。さらに、以下の機能説明で説明するように、タイマーや割り込みなどのATmega32u4の内部リソースを利用します。

に Ftduinoライブラリを使用するには、対応するインクルードラインをスケッチの先頭に挿入する必要があります。

```
# 含む <Ftduino.h>
```

加えて 初期化 （） -関数を呼び出すことができます。これは理にかなっている早い段階で起こります設定（）-関数。

```
//セットアップ関数は起動時に1回呼び出されます 空所 設定（）{

  // Ftduinoライブラリを使用する準備をします ftduino。初期化（）；

}
```

9.2.1入力 I1 それまで I8

入り口 I1 それまで I8 ATmega32u4マイクロコントローラのアナログ入力に接続されています ftドゥイーノ 接続されています。これらのアナログ入力は、ftドゥイーノ-アナログ変換には一定の時間がかかるため、ライブラリはバックグラウンドで永続的に評価されます。これにより、入力のクエリでの望ましくない遅延を回避できます。

the ftドゥイーノライブラリは、いわゆる ADC_vect-割り込み。アナログ-デジタル変換器（ADC）は、1秒あたり約8900回の測定速度に設定されています。安定した2番目の測定値を取得するために、各入力が2回照会されるため、8つの入力に対して合計16の測定値が必要になります。これにより、入力ごとに1秒あたり約560回の測定の変換率が得られ、バックグラウンドで自動的に実行されます。したがって、読み取り時の測定値は最大で約2ミリ秒前であり、値は約2ミリ秒ごとに更新されます。

the ftドゥイーノ 各入力でいわゆるプルアップ抵抗をアクティブにできるため、電圧測定を抵抗測定にすることができます。それはまたによってサポートされていますftドゥイーノ-ライブラリはバックグラウンドで管理され、スイッチオーバーは測定前に自動的に行われます。これは、チャネルごとに2つの測定が行われる理由でもあります。これにより、スイッチオーバー後、2回目の測定前に信号が安定します。

9.2.2 void input_set_mode（uint8_t ch、uint8_tモード）

関数 input_set_mode（） 入力の測定モードを設定します ch。 の有効な値 ch から Ftduino :: I1 それまで Ftduino :: I8。

値 ファッション することができます Ftduino ::抵抗、Ftduino ::電圧 また Ftduino :: SWITCH 悩ませる。関数input_get（） その後、抵抗値（オーム）、電圧値（ミリボルト）、またはスイッチのスイッチングステータスを真理値として提供します。

9.2.3 uint16_t input_get（uint8_t ch）

この関数は、入力の現在の測定値を読み取ります ch アウト。の有効な値ch から Ftduino :: I1 それまで Ftduino :: I8。

返される測定値は16ビット値です。電圧測定の場合、0〜100ボルトの電圧に対応する0〜10,000の値が返されます。抵抗測定の場合、0〜65535オームの抵抗値が返されます。これにより、65キロオームを超えるすべての抵抗に対して値65535も返されます。測定原理により、約10キロオームを超える値はますます不正確になります。スイッチを測定する場合のみtrue また false 入力が100オーム未満（スイッチが閉じている）でグランドに接続されているかどうかに応じて、返送されます。

この関数は通常、バックグラウンドで決定された最後の測定値をすぐに返します。入力の測定モードが事前に直接変更されている場合、関数が有効な測定値を返すまでに最大2ミリ秒しかかかりません。この場合、関数はプログラムの実行を長時間ブロックします。

例

```
// I1での抵抗を評価します
ftduino。input_set_mode（（Ftduino：：：I1、 Ftduino：：： 抵抗）；uint16_t抵
抗= ftduino。input_get（（Ftduino：：：I1）；
```

9.2.4出力 O1 それまで O8 と M1 それまで M4

出口 O1 それまで O8 従来の9ボルトアクチュエータを直接接続するための8つの独立した増幅出力です。
ランプ、バルブ、モーターなどのせん断技術。

タイプのドライバーブロック MC33879A[2] 制御されます。このモジュールには、8つの独立して制御可能なパワートランジスタが含まれています。2つのトランジスタはそれぞれ、出力をグランドに切り替えるか、9ボルトに切り替えるか、完全に未接続のままにすることができます。これにより、各出力の3つの可能な状態が発生しますLO（アースに切り替え）、 こんにちは （9ボルトに切り替え）または オフ （接続されていません）。

それぞれ2つの出口 牛 エンジン出力につながる可能性があります Mx 組み合わせる。出力O1 と O2 エンジン出力になります M1、O3 と O4 エンジン出力 M2 などなど。組み合わされたエンジン出力は、4つの可能な状態を持つことができますオフ、左、右 と ブレーキ 受け入れる。州では左 と 正しい 接続されたモーターは、接続の極性に応じて反時計回りまたは時計回りに回転します。状態でオフ 両方の出力が接続されておらず、モーターは接続されていないかのように動作し、たとえば比較的簡単に回転させることができます。状態でブレーキ たとえば、両方の出力がアースに接続されており、接続されているモーターにブレーキがかかっているため、回転が困難です。

モータードライバーは、ATmega32u4のいわゆるSPIインターフェースを介して接続されます。両方のモータードライバーは直列に接続されており、すべてのSPIデータ転送でデータが提供されます。出力での信号変化、特に出力でアナログ信号を生成するためのPWM信号生成（セクション6.3を参照）では、バックグラウンドでSPIバス上で継続的に通信する必要があります。これを行うには、Ftduinoバックグラウンドで永続的に実行され、モータードライバーのステータスを永続的に更新する、いわゆるSPI割り込みハンドラーをライブラリ化します。

注：出力は、次の場合にのみ使用できます。 ftドゥイーノ 9ボルトの電圧源に接続されています。

9.2.5 void output_set（uint8_tポート、uint8_tモード、uint8_t pwm）

この関数は、単一の出力を切り替えます。の有効な値ポート の範囲内にあります Ftduino :: O1 それまで Ftduino :: O8。

パラメータ ファッション 出力を切り替える出力モードを示します。許容値ファッション それは Ftduino ::オフ（出力がオフになっています）、 Ftduino :: LO（出力がグランドに切り替えられます）および Ftduino :: HI（出力は9ボルトに切り替えられます）。

the pwm-パラメータは、アナログ信号を生成するためのパルス幅変調の値を指定します。値は0から可能です（ Ftduino :: OFF） 最大64（Ftduino :: MAX また Ftduino :: ON） 十分です。ここで、0は十分を表し、64はオンを表します。出力に接続されたランプは0で点灯せず、64で明るく点灯します。中間値は対応する中間値を生成し、ランプは1で点灯します。pwm-低輝度でのみ32の値。可能であれば定数を使用する必要がありますFtduino ::オフ、Ftduino ::オン と Ftduino :: MAX これらはPWM値の範囲が変更された場合に使用されるため、たとえば、以降のバージョンで使用できます。 Ftduinoライブラリは簡単にカスタマイズできます。中間値は定数から導出できます（例：Ftduino :: MAX/2）。

例

```
//出力O2を50%に切り替えます
ftduino。output_set（（Ftduino：：：O2、 Ftduino：：：こんにちは、 Ftduino：：：MAX / 2）；
```

9.2.6 void motor_set（uint8_tポート、uint8_tモード、uint8_t pwm）

関数 motor_set（） 結合されたモーター出力を切り替えます。の有効な値ポート の範囲内にあります Ftduino :: M1 それまで Ftduino :: M4。

パラメータ ファッション モーター出力を切り替える出力モードを示します。許容値ファッション それは Ftduino ::オフ（出力がオフになっています）、 Ftduino :: LEFT（モーターが左に曲がる）、 Ftduino ::右（モーターは時計回りに回転します）そして Ftduino :: ブレーキ（モーターは、両方の個別の出力をグランドに切り替えることによってアクティブにブレーキがかけられます）。

the pwm-パラメータは、アナログ信号を生成するためのパルス幅変調の値を指定します。値は0から可能です（Ftduino :: OFF） 最大64（Ftduino :: MAX また Ftduino :: ON） 十分です。ここで、0は十分を表し、64はオンを表します。家畜

[2]下のデータシート http://cache.freescale.com/ les / analog / doc / data_sheet / MC33879.pdf。

出力接続モーターはモードで回転します Ftduino ::左 と Ftduino ::右 0ではなく、64で最大速度。中間値は対応する中間値を生成し、モーターは1つで回転しますpwm-低速の場合のみ32の値（モーター速度とPWM値の関係の詳細については、セクション6.3を参照してください）。可能であれば定数を使用する必要がありますFtduino ::オフ、Ftduino ::オンと Ftduino :: MAX これらはPWM値の範囲が変更された場合に使用されるため、たとえば、以降のバージョンで使用できます。 Ftduinoライブラリは簡単にカスタマイズできます。中間値は定数から導出できます（例：Ftduino :: MAX/2）。モードで Ftduino ::ブレーキ を決定します pwm-モーターのブレーキの強さの値。モードでFtduino ::オフ 持っている pwm-値は関係ありません。

例

```
// M3のモーターを1/3の速度で左に回します ftduino。motor_set（（Ftduino ：：：M3、
Ftduino ：：：左 、 Ftduino ：：：MAX / 3）；
```

9.2.7 void motor_counter（uint8_tポート、uint8_tモード、uint8_t pwm、uint16_tカウンター）

この関数はエンコーダモーターを制御するために使用され、最初の3つのパラメーターはと同じです。 motor_set（）-関数。これらのパラメータの意味は同じです。

追加の4番目のパラメーターは、エンコーダーモーターが実行するパルス数を指定します。ステップは、対応するカウンター入力、つまりカウンター入力で測定されますC1 エンジン出力用 M1、C2 にとって M2 などなど。指定されたインパルスが経過すると、モーターが停止します（を参照）。void motor_counter_set_brake（））。

インパルスのカウントとモーターの停止は、スケッチのその後の実行とは関係なく、バックグラウンドで行われます。モーター1回転あたりに検出されるパルス数は、モーターの種類によって異なります。TXTディスカバリーセットからモーターをお届けします63 1/3 TXコントローラー用に最初に販売されたセットからモーターによって供給されるモーター軸の1回転あたりのパルス 75 回転あたりのパルス。

9.2.8 bool motor_counter_active（uint8_tポート）

関数 motor_counter_active（） スルーのパルスカウントかどうかを返します ポート 指定されたモーター出力がアクティブです。の有効な値ポート の範囲内にあります Ftduino :: M1 それまで Ftduino :: M4。

アクティブとは、対応するモーターが呼び出しによってアクティブ化されることを意味します motor_counter（） が開始されており、パルスカウンタはまだ有効期限が切れていません。この機能は、特に、パルスカウントが停止し、モーターが停止するのを待つために使用できます。

例

```
// M4でTXTエンコーダモーターを3回転始動します ftduino。motor_counter（（Ftduino
：：：M4、 Ftduino ：：：左 、 Ftduino ：：：MAX 、 //モーターが停止するまで待ちます      190）；

その間（（ftduino。motor_counter_active（（Ftduino ：：：M4））；
//エンジンが停止しました
```

9.2.9 void motor_counter_set_brake（uint8_t port、bool on）

この機能は、出力でのモーターのブレーキ動作を決定します ポート、 彼が機能を通過しているとき motor_counter（） が開始されます。

パラメータの場合 の上 真に（true） が設定されている場合、時間が経過した後、モーターはアクティブにブレーキをかけられます。彼は真実ではありませんか （false）、 モーターはオフになっているだけで、ブレーキをかけずに惰性で停止します。ライブラリを初期化した後のデフォルト設定はtrueです。つまり、アクティブブレーキがアクティブになっています。

どちらの場合も、エンジンは引き続き実行されます。ブレーキがかかると、TXTディスカバリーセットのエンコーダーモーターは、約5インパルス（約1/10 革命または28.5°）。 ブレーキがかかっていない場合、同じモーターが約90パルス（約 1 1/2 革命）。

エンコーダが停止した後もカウンタは実行を継続するため、オーバーランはプログラムで測定することもできます。

例

```
//出力M4のブレーキをオフにします
ftduino。motor_counter_set_brake（（Ftduino：：：M4、 false）；
// M4でTXTエンコーダモーターを3回転始動します ftduino。motor_counter（（Ftduino
：：：M4、 Ftduino：：：左、 Ftduino：：：MAX、 //モーターが停止するまで待ちます      190）；

その間（（ftduino。motor_counter_active（（Ftduino：：：M4））；
//モーターが動作する時間を与えるためにもう少し待ちます 遅れ （500）；

//カウンタの読み取り値を出力します
シリアル。println（（ftduino。counter_get（（Ftduino：：：C4））；
```

9.2.10カウンター入力 C1 それまで C4

アナログ入力とは対照的に、カウンタ入力は純粋にデジタルで機能します。それらは、それぞれの入力がグランドに接続されているかどうかを区別するだけです。これは通常、カウンタ入力とそれに対応するアース接続の間に接続されたテスター、またはエンコーダ出力がカウンタ入力に接続されているエンコーダモーターを使用して行われます。カウンタ入力には内部プルアップ抵抗があります。それは彼らが日付を記入されていることを意味します ftドゥイーノ 接続されたボタンが押されていないなどの理由で信号がない場合は、高または高信号レベルとして認識されます。ボタンを閉じると、入力がグランドに切り替わります。ftドゥイーノ 低として認識されます。

4つのカウンタ入力は、ATmega32u4の割り込み対応入力に直接接続されています。技術的には、毎秒数十万のカウントパルスの範囲での反応が可能です。たとえば、キーストロークがカウントされる場合、避けられないバウンス（セクション6.12を参照）は誤った結果につながります。このため、Ftduinoバックグラウンドでライブラリを作成し、イベントの最小長を1ミリ秒に制限します。短いイベントはカウントされません。

システムの起動後、4つのカウンターすべてがゼロに設定され、非アクティブ化されます。入力でのイベントはカウンターを変更しません。

さらに、カウンター入力により C1 セクション1.2.6に示すように、schertechnik ROBOTX超音波距離センサー1330009の接続。

9.2.11 void counter_set_mode（uint8_t ch、uint8_tモード）

この関数は、カウンタ入力の動作モードを設定します。の有効な値ch から Ftduino :: C1 それまで Ftduino :: C4。

しますか ファッション-の価値 Ftduino :: C_EDGE_NONE 設定すると、信号の変化はカウントされず、カウンタは非アクティブになります。これが開始状態です。

意思 ファッション の上 Ftduino :: C_EDGE_RISING 設定すると、立ち上がり信号がカウントされます。つまり、入力信号がグランドからより高い電圧に変化します。これは、たとえば、接続されているボタンが離された（開いた）ときに発生します。

A ファッション-の値 Ftduino :: C_EDGE_FALLING 立ち下がり信号がカウントされます。つまり、入力信号が高電圧からグランドに変化します。これは、たとえば、接続されたボタンが押された（閉じた）ときに発生します。

しますか ファッション-最後に、価値 Ftduino :: C_EDGE_ANY が設定されている場合、両方の信号変更方向により、カウンタがインクリメントされます。次に、たとえば、テスターを押すことと離すことの両方がカウントされます。

9.2.12 uint16_t counter_get（uint8_t ch）

この関数は、現在のカウンターステータスを返します。の有効な値ch の範囲内にあります Ftduino :: C1 と Ftduino :: C4。

返される最大値は65535です。この値を超えると、カウンターは0に戻ります。

9.2.13 void counter_clear（uint8_t ch）

機能の助けを借りて counter_clear（） カウンタの読み取り値はゼロに設定できます。の有効な値ch の範囲内にあります Ftduino :: C1 と Ftduino :: C4。

例

```
//入力C1での立ち上がり（低から高）パルスを1秒間カウントします ftduino。counter_set_mode（（Ftduino
：：：C1、 Ftduino：：：C_EDGE_RISING）；ftduino。counter_clear（（Ftduino：：：C1）；遅れ
（1000）；

uint16_tインパルス = ftduino。counter_get（（Ftduino：：：C1）；
```

9.2.14 bool counter_get_state（uint8_t ch）

カウンタ入力の状態は、機能で直接確認することもできます。 counter_get_state（） 照会されます。の値ch の範囲内である必要があります Ftduino :: C1 それまで Ftduino :: C4 横たわる。

この関数はtrueを返します（true） 入力がグランドに接続されていてfalseの場合は戻る（false） それが開いているとき。

この機能にはフィルタリングがないため、たとえば、キーのバウンスは抑制されません。このようにして、非常に高い周波数のデジタル信号を記録することができます。

9.2.15 void Ultrasonic_enable（bool ena）

カウンタ入力時 C1 あるいは、Schertechnik ROBO TX超音波距離センサー1330009は、セクション1.2.6に示すように操作できます。関数超音波有効化（） パラメータが次の場合にセンサーのサポートをアクティブにします エナ 真に（ true） が設定され、falseに設定されると非アクティブ化されます（false） が設定されています。

超音波センサーのサポートがアクティブになっている場合、入力のカウント機能 C1 自動的に非アクティブ化されます。

超音波センサーが作動すると、バックグラウンドで1秒に約2回継続的に評価されます。したがって、現在の測定値は最大500ミリ秒前のものです。

9.2.16 int16_t超音波_get（）

関数 超音波取得（） カウンタ入力の測定値を供給します C1 センチメートル単位の接続距離センサー。起動後にセンサーが有効な測定値を受信して いない場合は、距離として-1が返されます。これは、センサーが接続されていない場合にも発生します。

センサー自体は0〜1023センチメートルの範囲で動作します。

例

```
//入力C1で距離センサーを照会します ftduino。超音波有
効化（（true）；
遅れ（1000）；        //最初の測定に1秒与えます int16_t距離 = ftduino。超音波_get
                        （）；
```

9.3コマンドの概要

コマンドの概要 FtduinoSimple

指図	説明
bool input_get（uint8_t ch）bool counter_get_state（uint8_t ch） void output_set（uint8_tポート、uint8_tモード）void motor_set（uint8_tポート、uint8_tモード）	デジタル入力での読み取り I1-I8 カウンタ入力状態の読み込み C1-C4 単一の出力を切り替える O1-O8 モーター出力の切り替え M1-M4

コマンドの概要 Ftduino

指図	説明
void input_set_mode（uint8_t ch、uint8_t mode） uint16_t input_get（uint8_t ch） void output_set（uint8_t port、uint8_t mode、uint8_t pwm） void motor_set（uint8_t port、uint8_t mode、uint8_t pwm）void motor_counter（uint8_t port、uint8_t mode、 uint8_t pwm、uint16_tカウンター）bool motor_counter_active（uint8_tポート） void motor_counter_set_brake（uint8_t port、bool on） void counter_set_mode（uint8_t ch、uint8_t mode）bool counter_get_state（uint8_t ch） void超音波有効化（bool ena）int16_t 超音波_get（）	入力モードの設定 I1-I8 入力の読み取り I1-I8 出力の切り替え O1-O8 モーター出力の切り替え M1-M4 エンコーダモーターの切り替え M1-M4 エンコーダモーターカウンタの評価 C1-C4 エンコーダモーターブレーキの設定 M1-M4 カウンターモードの設定 C1-C4 カウンタ入力状態の読み込み C1-C4 超音波センサーのアクティブ化超音波センサーの読み取り

第10章

DIY

1つを持つことが可能です ftドゥイーノ 手動でビルドします。最初のプロトタイプはこの方法で作成されました。手動で組み立てる場合は、機能グループに従って進め、機能を段階的に確認することをお勧めします。

自己構築の基礎は、付録Aに従った回路図に基づいて、付録BおよびCから工業的に製造された回路基板です。

10.1建設段階の電源

最初のステップでは、電源装置をセットアップします。コンデンサーダイオードで構成され エーター C6 それまで C11 そのようなていますD1 と D3 それまで D5 だけでなく、電圧レギュレータ U2 とバックアップ F1。 クラ C14、 ngsversorungs-LED（eコイルンプ部1.2.4） LED2 関連する直列抵抗器と R35 マウントされています。火曜日 C1 すでにイ を参照） L1 コンデンサと同様にンストールすることもできます。

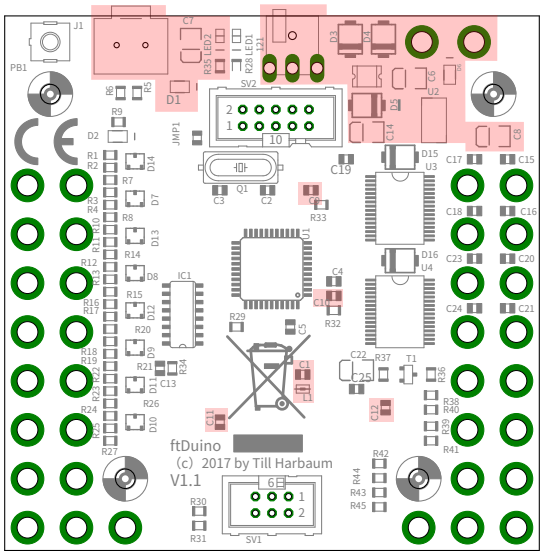


図10.1：電源のコンポーネント

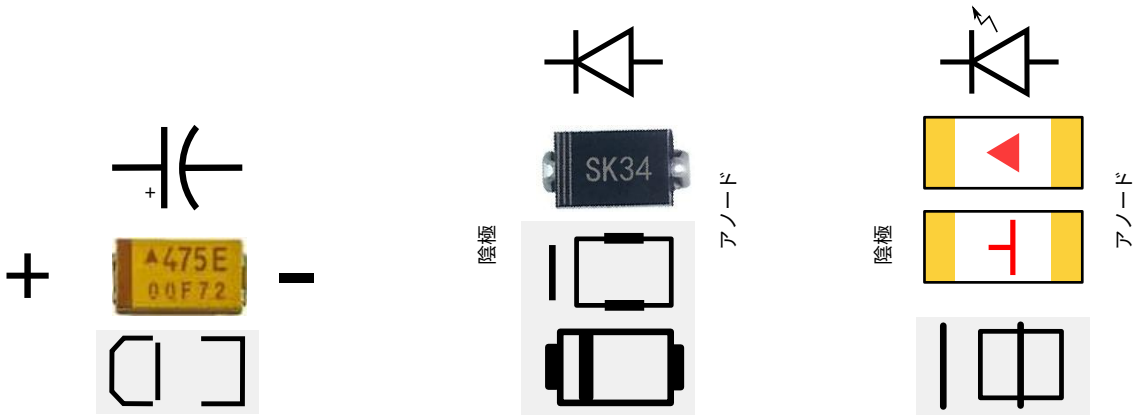
9Vソケット 121 2つの襟の袖と同様に 9VIN + と 9VIN- USBはすでにロードされていま　ライフル J1 今もす。

10.1.1コンポーネントの極性

次のコンポーネントの極性に注意してください。 D1 と D3 それまで D5、C6 それまで C8 と C14 そのような LED2。

コンデンサの正の接続は、印刷されたバーまたはストリップで示されます。この接続は、組み立て計画でもバーと角度の付いたコーナーでマークされています。

のさまざまなダイオード ftドゥイーノ アセンブリ印刷には異なる記号を使用してください。しかし、ここでも、ダイオード自体とアセンブリシンボルの両方のカソード側にバーがあります。



（a）タンタルコンデンサ (b) ダイオード (c）発光ダイオード

図10.2：回路図の記号、コンポーネントの極性、およびアセンブリの記号

発光ダイオード（LED）を使用すると、正しい極性を判別するのが少し難しくなります。三角形またはTは通常、LEDの下側に印刷され、それぞれがカソードの方向を指しています。

発光ダイオードの極性は、ダイオードテストモードの簡単なマルチメータで確認できます。LEDのアノードを赤いマルチメータケーブルに接触させ、カソードを黒いケーブルに接触させると、LEDが弱く点灯します。

電源のすべてのコンポーネントが装備されている場合、USBコネクタが接続されるとすぐに、またはカラースリーブまたは9V丸型プラグを介して9ボルトが供給されるとすぐに緑色のLEDが点灯します。

少なくとも1つのケースでLEDが点灯しない場合は、マルチメータを使用して、電圧がまだ存在している場所と存在していない場所を簡単に追跡できます。最も可能性の高いエラーは、ダイオードまたはLEDの極性にあります。

10.1.2制御測定

Iのピン1と2の間に9ボルトの電源があります。2組み立てられていないコネクタのC接続 SV1 5ボルトの電圧（±0.4ボルト）を測定することができます。ここで測定された電圧が明らかに高すぎる場合は、いかなる状況でもマイクロコントローラーの組み立てを続行しないでください。

また、9ボルトの電源からの電源では、以前は組み立てられていなかった2つの低い9ボルトの出力でほぼ9ボルトを測定する必要があります。ソースからの電圧損失は、ダイオードを介して発生しますD3 また。 D4 とダイオード D5 の上。

以前は装着されていなかったパッド14と7の間 IC1 純粋なUSB電源で5ボルトをわずかに下回る電圧を測定できる必要があります。そうでない場合、電圧レギュレータはU2 いわゆるボディダイオード（推奨されるMCP 1755Sにはこれがあります）はなく、このダイオードの機能はスルーでなければなりません D6 外部から後付けすることができます。推奨されるMCP1755Sを使用する場合、D6 交換せずに省略。

ダイオード D5 ボディダイオードを介しての出力ドライバへの電流を防ぎます ftドゥイーノ 得た。それ以外の場合、ftドゥイーノ また、USB 5ボルト電源から給電されるため、USBやボディダイオードが過負荷になる可能性があります。

10.2第2建設段階　　マイクロコントローラー

電圧供給は確保されており、何よりも、Iで9ボルトで動作していますか。2C接続は5ボルトで安定し、マイクロコントローラーと接続できます U1 つづく。

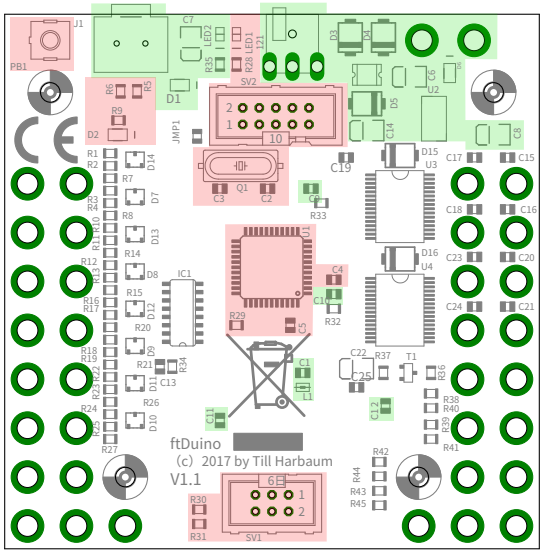


図10.3：マイクロコントローラーのコンポーネント

マイクロコントローラの極性も逆にしないでください。この場合、正しい向きではんだ付けする必要があります。回路基板とチップハウジングの両方に、1つのコーナーに丸いマーキングまたはくぼみがあります。
このマーキングは、マイクロコントローラのピン1を参照し、そのすぐ隣にある正しいOを決定します。 オリエンテーション。はU1 はんだ付けする C5 と R29 組み立てられました。

この段階で組み立てられる他のコンポーネントには、リセットロジックダイオードが含 ボタンで構成 PB1、 USB回路をまれています D2 と抵抗 R9。 コンデンサ C4 と抵抗 R5 と R6 水晶は16MHzのシステムク変更するだけでなく。The ロックを完了します Q1 コンデンサー付き C2 と C3。

私。2Cコネクタ SV1 プルアップ抵抗付き R30 と R31 今もできます　　在庫があります。

発光ダイオード LED1 それらの直列抵抗器で R28 マイクロコントローラから直接マウン制御され、現在もトされます。発光ダイオードの極性を再度確認する必要があります。

いわゆる ISPコネクタ SV2 Bの1回限りの取り付けにのみ必要であり、標準のハウジングootloaders（ハウジングに対応にこのコネクタの切り欠きがない場合は、必ずしも恒久的に取り付ける必要はありませ する切り欠きがあるため、セクん。自由に利用できる印刷テンプレート1 恒久的にインストールされている SV2 利用さ ション1.2.1を参照）れる。

10.2.1マイクロコントローラの機能テスト

マイクロコントローラーは、USBブートローダーを備えたAtmel（またはMicrochip）製です2 したがって、PCからは、配信されます。enに接続する場マイクロコントローラはPCによってデバイスとして指定される必要があります ATm32U4DFU 重要なコンポーネントは合。その場合、帽子は無視できる機能していると認識されています。そのWindowsにはこのデバイス用のドライバーがありません　　ます。

DFUブートローダーはArduinoIDEと互換性がなく、ArduinoIDEはコネクタを介してプ 独自のブートローダー。これは燃ログラミングデバイスを提供します SV2 記録された（ge　　える）。

Arduinoブートローダーを書き込むために、Arduino IDEは、USBaspなどの単純なバリアン nプログラミングデバイス。10トの全範囲をサポートします3。USBaspは、USBおよび経由でPCに接続されます　　ピンリボンケーブルで十分です

1ケース印刷テンプレート https://github.com/harbaum/ftduino/tree/master/case
2DFUブートローダー、 http://www.atmel.com/Images/doc7618.pdf
3USBasp-Atmel AVRコントローラー用のUSBプログラマー、 http：// www。schl.de/usbasp/

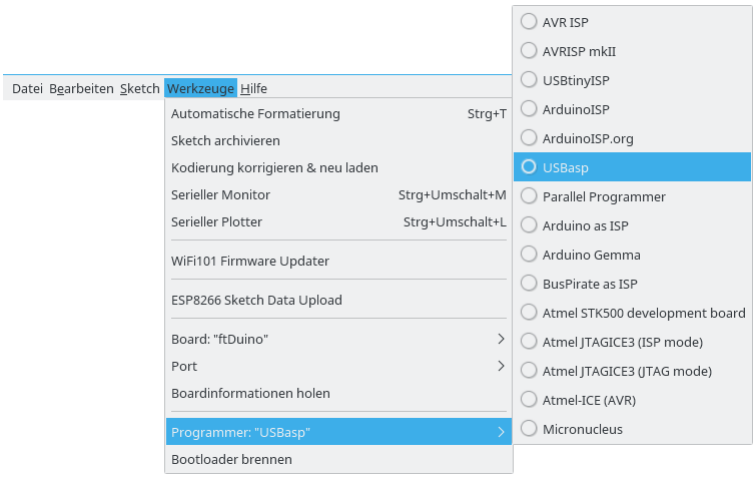


図10.4：ArduinoIDEを介したブートローダーの書き込み

SV2 の ftドゥイーノ 接続されています。はSV2 装備されていない場合、対応するプラグがボードに緩く差し込まれ、すべての接触が行われるように簡単に誤解される可能性があります。実際のフラッシュプロセスは数秒しかかからず、わずかにずれている間、プラグを非常に長い間簡単に保持できます。

燃やした後、 ftドゥイーノ この名前でPCのオペレーティングシステムによって認識できます。Arduino IDEで対処し、スケッチでプログラムする必要があります。以下のまばたきスケッチを最初のテストに使用できます
ファイル例 。FtduinoSimple 。点滅 必要だから LED1 インストールされたばかりです。

10.3第3建設段階の入り口

3番目の構築段階ははんだ付けが非常に簡単で、主に抵抗器で構成されます。 m使用される入力保護

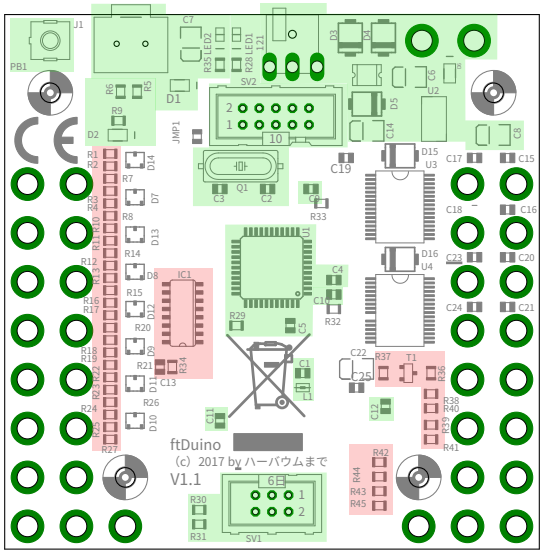


図10.5：入力のコンポーネント

アナログ入力のコンポーネント I1 それまで I8 カウンター入力と同様に C1 それまで C4 誰 同時にマウントされます。

組み立ては、左端の抵抗器から始まります。 IC1 それらの組み立て後d。 すなわち入力 I1 それまで I8 完全に

2番目のステップは抵抗器です R36 それまで R46 トランジスタだけでなく T1 装備され、カウンター入力を完了します。これで、超音波距離計のトリガー回路（セクション1.2.6を参照）が完成しました。

適切なテストプログラムを使用して、入力間の短絡も検出するために、各入力を個別にテストする必要があります。入力が期待どおりに機能しない場合は、構築ステージ2のマイクロコントローラーもエラーの原因と見なすことができます。

10.4第4建設段階の成果

第4の最終建設段階では、出力を操作するために必要なコンポーネントがインストールされます。

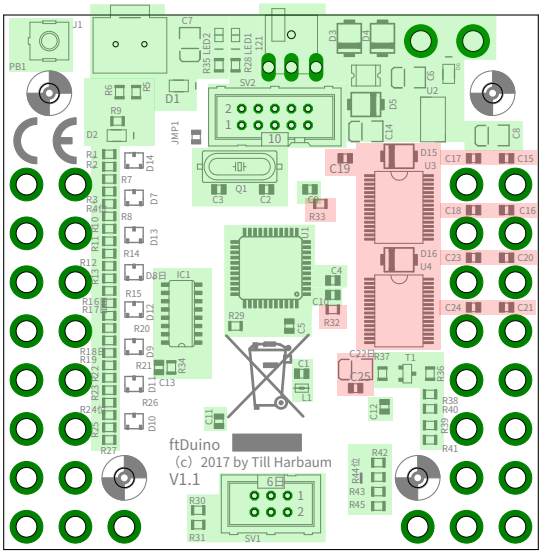


図10.6：出力のコンポーネント

パフォーマンスドライバー U3 と U4 少数の抵抗とコンデンサを介してドライバの接続 uははんだ付けしています。何よりも、r パッドにアクセスできるため、カラースリーブを簡単に取り付けることができないた はそれを難し くするはずです。The め、最初に組み立てる必要があります。

10.4.15ボルトでの出力テスト

ドライバは9ボルト電源に接続されているため、はんだ付けエラーが発生した場合、9V間の短絡が oltおよび5ボルトの先行信号 発生する可能性があります。

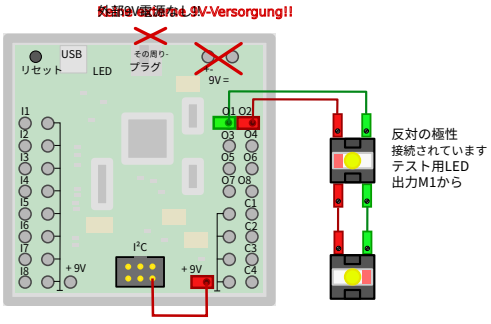


図10.7：5Vと9Vの間のテストブリッジ

はんだ付けポイントを注意深くチェックすることに加えて、初期機能テストのために5ボルトの分岐から9ボルトの分岐を供給することは理にかなっています。5ボルトを運ぶ信号間の短絡は一般に損傷を引き起こしませんが、より高い電圧の5ボルトのコンポーネント間の短絡は非常に簡単に大きな損傷を引き起こす可能性があります。

注意：もちろん、外部の9ボルト電源を接続することはできません。接続しないと、 ftドゥイーノ すぐにダメージを受ける！

ブリッジが設定されている場合、 ftドゥイーノ の内部5ボルトで ftドゥイーノ 出力と他の部分との間の短絡 ftドゥイーノ 危険性は低くなります。これで、発光ダイオードを使用して出力が正しく機能するかどうかをテストできます。内部の5ボルト電源はそのような負荷を駆動するように設計されていないため、このテストではより大きな負荷（モーターまたは電球）を使用しないでください。

すべての出力が5ボルトで正しく機能する場合にのみ、ブリッジを取り外して実際の9ボルト電源を接続できます。

10.4.2取締役会の歴史

the ftドゥイーノ 継続的に開発されています。したがって、回路基板もわずかに変化します。これまでのところ、次のバリアントが存在します。

0.1 の試用版 ftドゥイーノ まだAtmega328に基づいており、入力と出力にも使用されていました
: 他のハードウェア。このバージョンから8つのプロトタイプが作成されました。

1.0 近直列 回路基板の最初のバージョンは、設計エラーのために10個のプロトタイプにのみ使用されました。

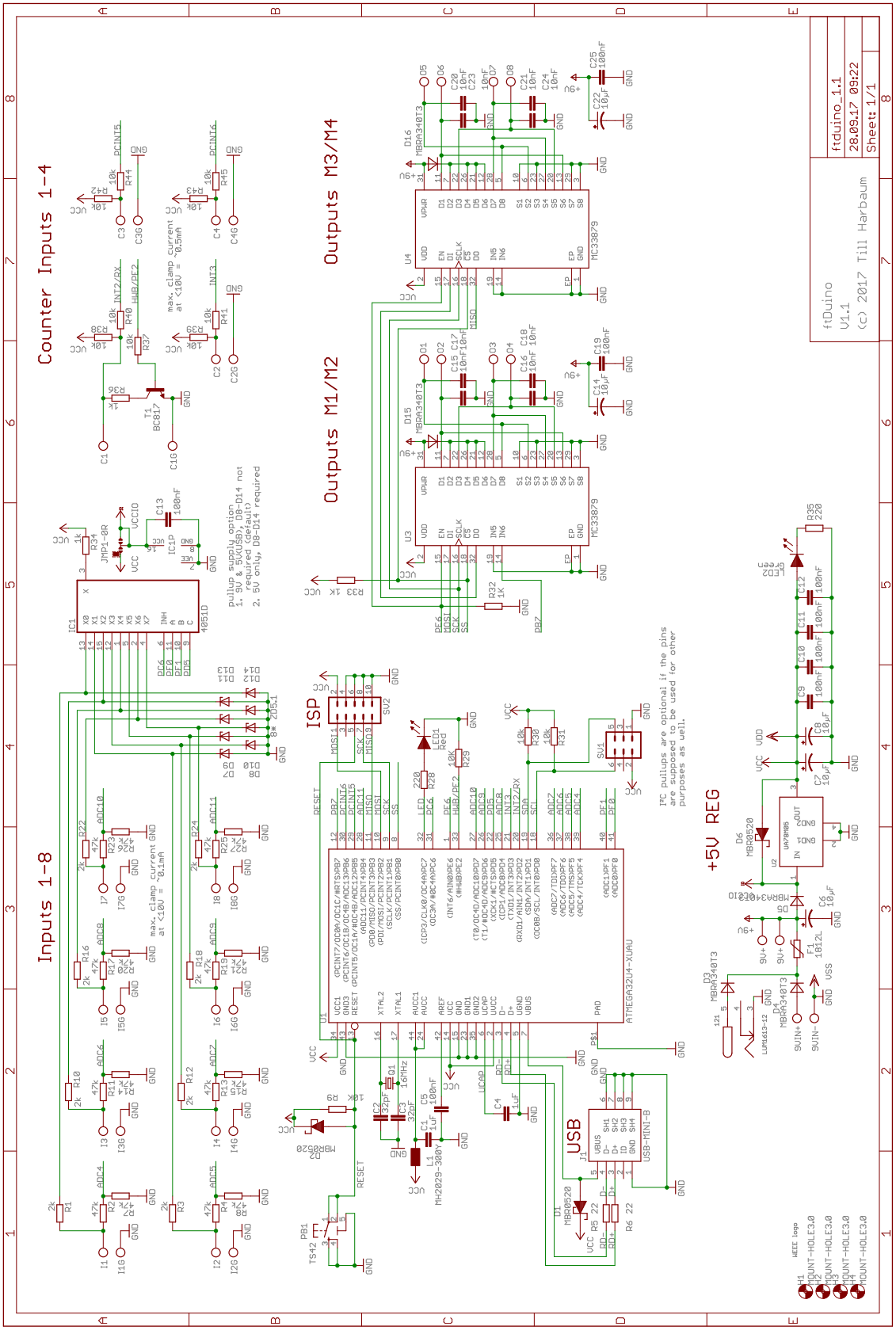
1.1 この章では最初のシリーズバージョンを示し、最初のシリーズデバイスはこのバージョンに基づいていました。このバージョンは、オプションで入力にツェナーダイオードD10〜D17を装備できますが、これはシリーズでは使用されませんでした。

1.2 2番目のシリーズバージョンでは、シリーズで使用されていないオプションのツェナーダイオードが不要です。代わりに、このバージョンには、セクション1.2.7で説明されているように、内部OLEDディスプレイ用の追加の接点があります。

1.3 3番目のシリーズバージョンには、セクション6.13.9で説明されているように、オプションで内部サーボアダプタを提供できます。このバージョンには、電源装置に追加のESD保護も含まれています。

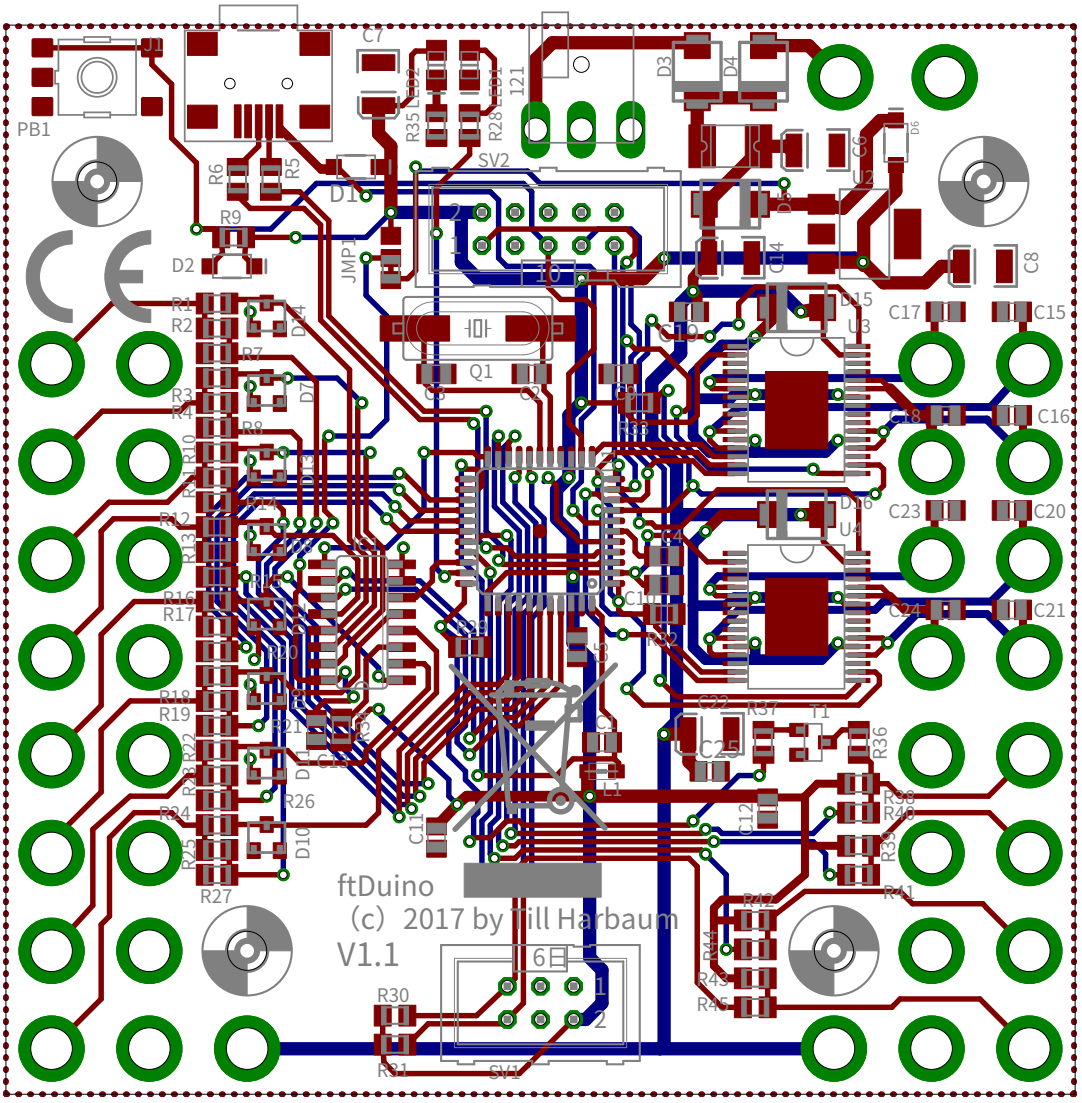
バージョン1.0のすべてのボードバージョンは、論理的および電気的に互換性があり、ユーザーの観点からは同じように動作します。

付録A：回路図



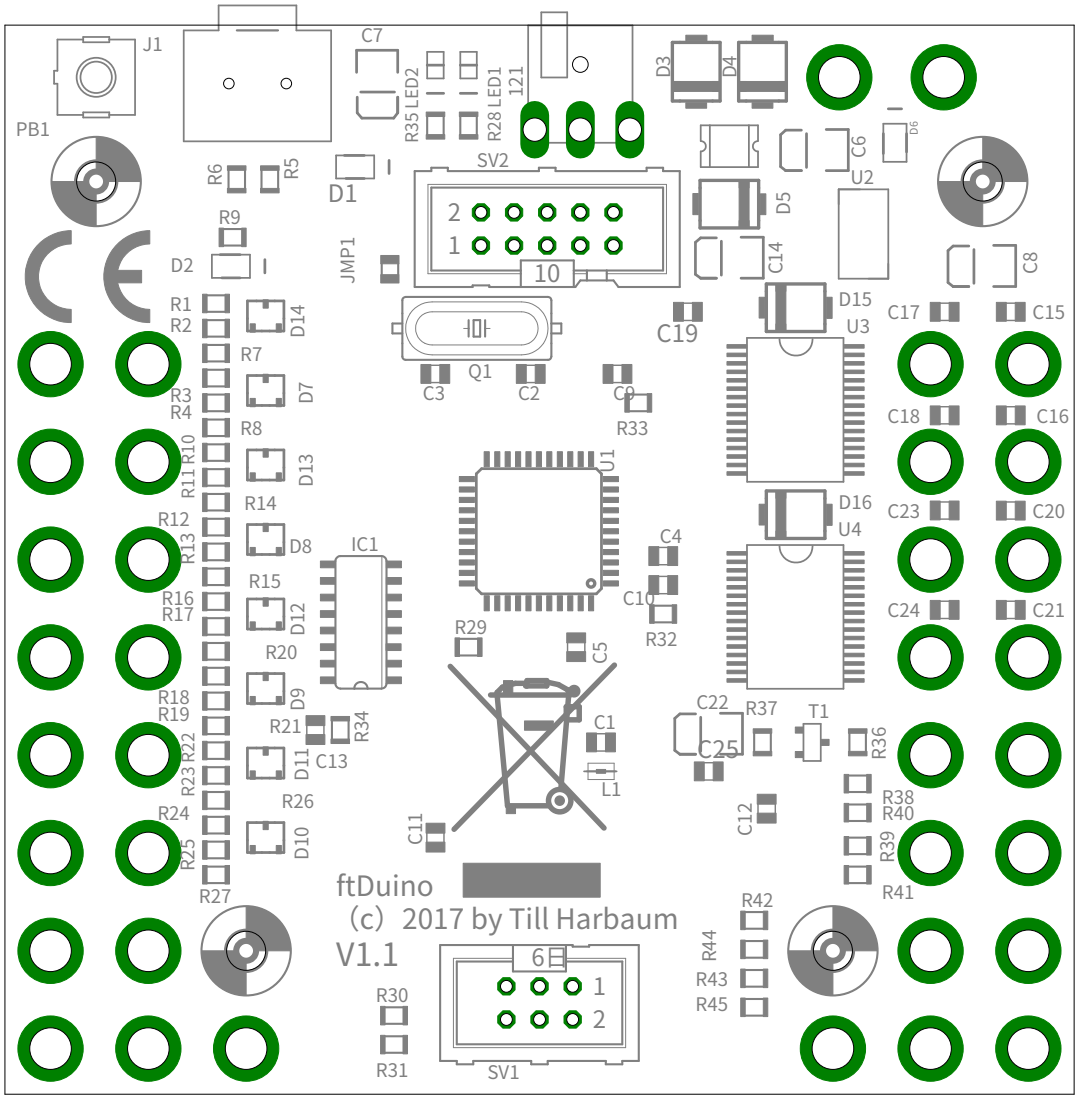
図A.1：回路図 ftドゥイーノ バージョン1.1

付録B：ボードレイアウト



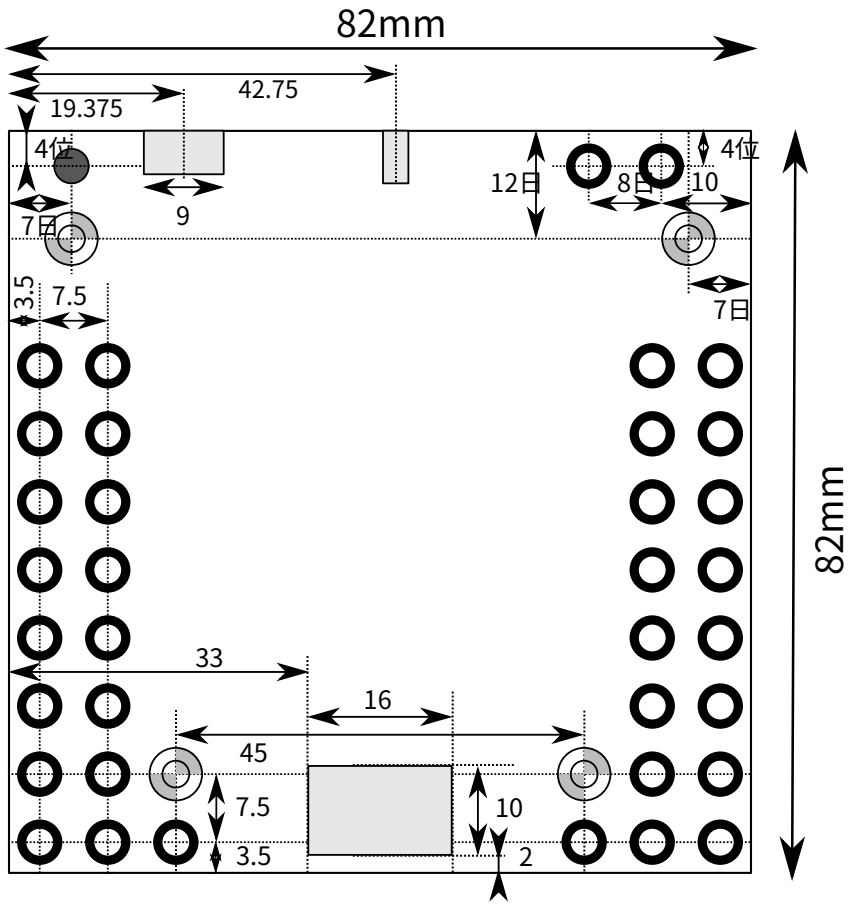
図B.1：ボードのレイアウト ftドゥイーノ バージョン1.1

付録C：コンポーネントのレイアウト

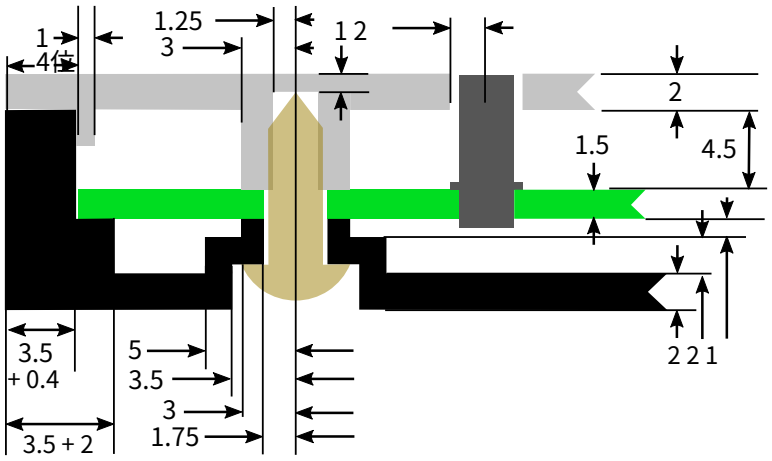


図C.1：機器計画 ftドゥイーノ バージョン1.1

付録D：寸法

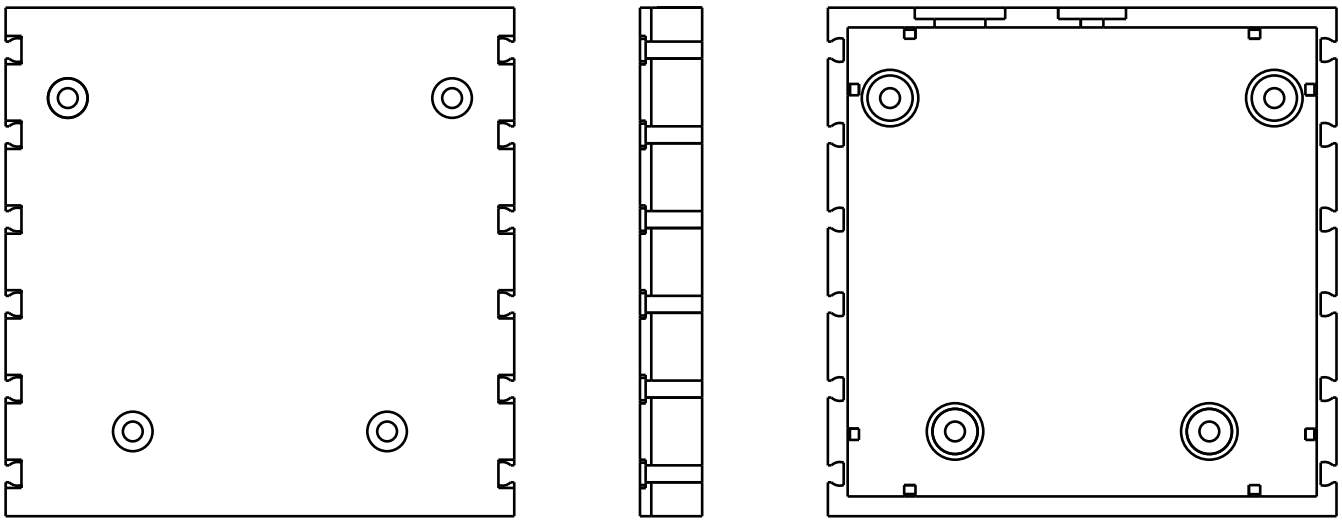


図D.1：寸法A ftドゥイーノ バージョン1.1

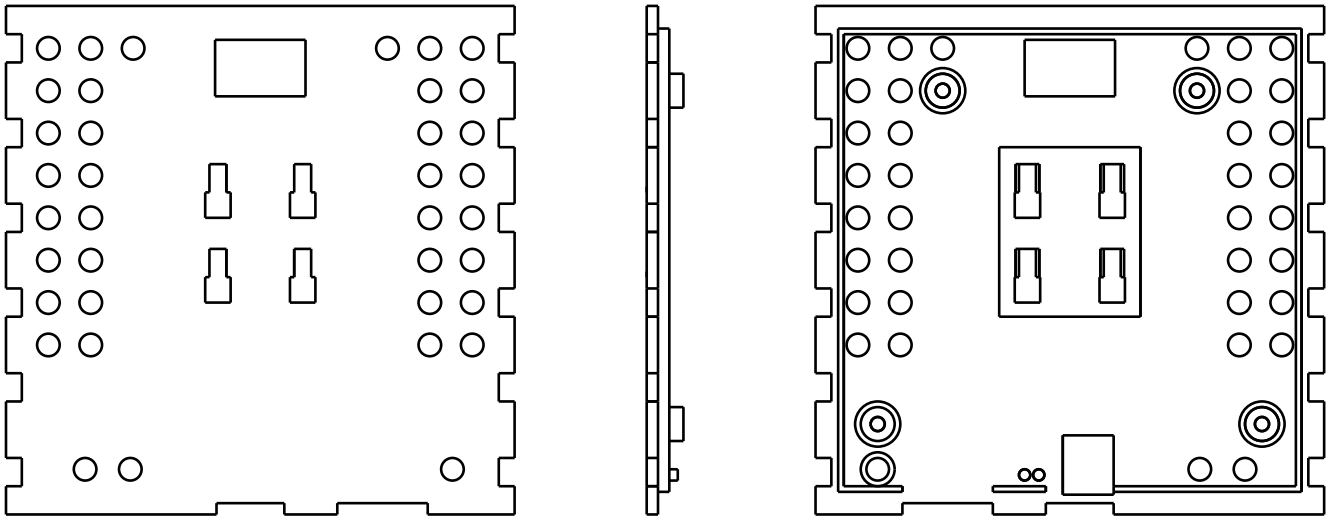


図D.2：寸法B ftドゥイーノ バージョン1.1

付録E：住宅



図E.1：下部ハウジングシェル



図E.2：上部ハウジングシェル