### **ARDUINOワークショップ** センシングネット社 小林 靖 2019年7月18日

## 小林プロフィール

- 旭化成エレクトロニクス株式会社にて、半導体回路設計に従事
- 主な設計分野

アナログーオーディオ、通信、センサー、パワーマネジメント デジタルーインターフェース、信号処理

• 趣味での電気いじり

オーディオアンプ、パソコン製作

### ワークショップ概要

- 本日のワークショップは、チーム毎、個人毎に各自のパソコンで、 Arduinoキットを使って、電子回路製作を実践する。
  - C言語でのプログラミング作成
  - センサーやLEDを使ったArduinoハードウエア製作
  - 合わせて、センサーインターフェース(SPI、I2C、UART)の理解

インターフェース毎に、上記内容を繰り返してワークショップを行う予定。 Arduinoを初めて使う方にも、分かり易く進める予定。

## Arduinoワークショップの目的

- センサーからのセンシングデータをPCなどに送るシステムを開発する目的に、Arduinoボード(及びその互換ボード)を用いる事を想定。
- この目的の為のマイコンボードは、Arduino以外にも多くあるが、Arduinoは、比較的容易に、また、互換ボードを使えば、低コストで、システム開発可能。
- また、Arduinoを使った電子工作例は、WEB上に大変多く、 自分の開発したいものに、近い例が、容易に検索可能。
   (PCへのデータ転送は、ワークショップではUSBによる有線
   互換ボードには、WiFiやBLEなど無線でも可能。)

## ARDUINOとは

ウィキペディアを参照

https://ja.wikipedia.org/wiki/Arduino

本日のファイルのダウンロード先やURL

http://supersensingforum.com/meeting/190718.html

### 1. Arduinoのインストール (1) ネットワークに接続 → SSID P-Comp\_Wifi (2) Arduinoのインストール → <u>https://www.arduino.cc/</u> アクセスして、ダウンロード後、インストール



### 2. Arduinoの環境設定

ファイルメニューの環境設定をクリック

Ctrl+O				
開く	<mark>&gt;</mark>			
	>			^
	>to run once:			
Ctrl+W				
Ctrl+S				
Ctrl+Shift+S	run reneatedly.			
Ctrl+Shift+P	, ran repeatedry.			
Ctrl+P				
Ctrl+カンマ				
Ctrl+Q				
	Ctrl+W Ctrl+S Ctrl+Shift+S Ctrl+Shift+P Ctrl+P Ctrl+カンマ Ctrl+カンマ	ctrl+W Ctrl+S Ctrl+Shift+S Ctrl+Shift+P Ctrl+P Ctrl+Q Ctrl+Q	ctrl+W Ctrl+S Ctrl+Shift+S Ctrl+Shift+P Ctrl+P Ctrl+D Ctrl+Q	Ctrl+W Ctrl+S Ctrl+Shift+S Ctrl+Shift+P Ctrl+P Ctrl+Q

短 ネッワーク スケッチブックの保存場所: CMUsersWinnerWDocumentsWarduina 正子がの文学の大きだ: I2 インダフェースのスケール: 「日日 100 ** 変更の反映にはArduino IDEの再起動が必要 ティア: デフォルトのテーン。変更の反映にはArduino IDEの再起動が必要 大野雑報な情報を表示する: コンパイル - ***********************************	厚堤認足			^		
A by 5 J by 01 (名材紙 The Second Seco	設定 ネットワーク					
C4UserskinnelWDocumentskArduino       ●標         言語説定:       System Default       文更の反映にはArduino IDEの再起動が必要         エディの文字の大き:       12         インタフェースのスケール:       公目動       100 * K       変更の反映にはArduino IDEの再起動が必要         テマ:       デフォルトのテーーン       変更の反映にはArduino IDEの再起動が必要         ン切けるの響告:       なし       日分のドキュメントフォル ダーのArduinon         ○「作場香を表示する □ つトの折ぶしたすった後に用する       日分のドキュメントフォル ダーのArduinon         ②者参込みを検掘する       ロin no         ○「常参な見の友快に対応に、放展子を力かからいこ変更する       マリンパイルたれたコアを積極的にたヤッシュする         ②ためかすを指律する際に、北球長子を力かからいこ変更する       マリンパイルたれたコアを積極的にたやッシュする         ○日動的がに発明した。       レン         「たちかったりなき行うかにスケッサを指称する       レントフナル ダーのArduinon         ○フパイルなたれたコアを積極的にキャッシュする       レンパルシッシの有無をチェッシウする         ○兄校のURL:       https://dlaspressif.com/dl/package_esp82_index.json       正         UF のファイルを直接編集まれば、よりがらの設定を行うたとができます。       C         CVISerset       Arduino IDEの参析 Toとせておいてくだきい。         編集する際には、Arduino IDEを始てきせておいてくだきい。       ー	スケッチブックの保存場所:					
語説定:	C:¥Users¥ <b>user</b> s¥ <b>use</b> /¥Do	ocuments¥Arduino		参照		
IF 7490文字の大参: 12 1. クレタフェースのスケーい: 「自動 100℃ ※ 変更の反映には Arduino IDEの再起動が必要 2. 切ド400丁	言語設定:	System Default 🗸 🗸 🗸	変更の反映にはArdu	iino IDEの再起動が必要		
インタフェースのスケーパ:       「自動」」00** 変更の反映には Arduino IDEの再起動が必要         テーマ:       「フォルトのテー」、変更の反映には Arduino IDEの再起動が必要         は助理編な情報を表示する:       コンパイル   書を込み         コンドクがり返しを有効に:       本レー         日春号を表示する:       コンドクがり返しを有効に         コードのがり返しを有効に       クックム r d u i n o         営き込みを検証する       ー         ウ脂のエディグを使用する       ー         コンドイルたコアを積極的にたやッシュする       ー         シスシッチ ち体存する際に、北部長・マシロする       ー         シスシッチ ち体存する際に、北部長・マシロする       ー         シスシッチ ち体存する際に、たいション       「抽動」に変更する         「検証または書き込みを行う前にスケッチ ち保存する       ー         追加のボードマネージンのURI:       Intro: //dlepressif.com/dl/package.gs92_index.json         レドのケイルを直接編集すれば、よりやくの酸定を行うことができます。       ー         Callsersk:       ムー         編集する際には、Arduino IDEを終了 させておいてください。	エディタの文字の大きさ:	12				
F.Y: 「アクル MDF-」 ▼ 変更の反映(cl Arduino IDEの再起動が必要 は)取締ぬた情報を表示する □ コードの折り返しを有効に □ コードの近けのさを使用する □ コードのがり返しを有効に □ コードの折り返しを有効に □ コードのがり返しを有効に □ コードのがり返しを有効に □ コードのがり返しを有効に □ コードのがり返しを有効に □ コードのがり返しを有効に □ コードのがり返しを有効に □ コードのがり返しを有効に □ コードのがり返しを有効に □ コードのが □ コードの □ コードのが □ コードの □ コードのが □ コードのが □ コードのが □ ロードのが □ コードのが □ コードので □ コードのが □ コードので □ コードのが □ コードので □ コードのが □ コードので □ コードのが □ コードのが □ コードので □ コードのが □ コードの □ コードのが □ コードのが □ コードの □ コー □ コー □ コードの □ コードの □ コー □ コー □ コー □ コードの □	インタフェースのスケール:	🗹 自動 🛛 100 😂 % 変更の反映には Arduino IDEの再起	記動が必要			
は野細な 情報を表示する: コンパイル   書き込み ンパイラの警告: なし 、 日行番号を表示する コードの折り返しを有効に 全書き込みを検証する 分部のDTディタを使用する コンパイルだれたコアを積極的にキャッシュする。 乏起動時に最新バージョンの有無をチェックする スカッチを保存する際に、拡張子をpdeがら.noに変更する 没たりまたは書き込みを行う前にスカッチを保存する 追加のボードマネージャのURL: https://dlespressif.com/dl/package_esp82_index.json 以下のファイルを直接編集すれば、より多くの設定を行うことができます。 CYUsers Human Map DataHocalHArduino ISPreferences.txt 編集する際には、Arduino IDEを終了させておいてください。	テーマ:	デフォルトのテー > 変更の反映にはArduino IDEの再起	動が必要			
□ンパイラの警告: なし 、 □ 1-ドの折り返しを有効に □ 1-ドの折り返しを有効に ○ 書参込みを検証する □ 分部のIIディタを使用する ○ コンパイルされたコアを積極的にキャッシュする。 ○ 主起動時に最新バージョンの有無をチェックする。 ○ スケッチを保存する際に、拡張子をpdeからinolC変更する。 ○ えケッチを保存する際に、拡張子をpdeからinolC変更する。 ○ 検証または書き込みを行う前にスケッチを保存する。 注助のポードマネージャのURL: https://dlespressif.com/dl/package_esp32_index.json II F0ファイルを直接編集すれば、より多くの設定を行うことができます。 CYUsers ¥int of the YApp Data¥Local #Arduino IB*preferences.txt 編集する弊能には、Arduino IDEを終了させておいてください。	より詳細な情報を表示する	5: 🗌 コンパイル 🗌 書き込み				
○ 行番号を表示する □ コードの折り返しを有効に ④ 書き込みを検証する □ 分部のエディダを使用する ④ コンパイルされたコアを積極的にキャッシュする ④ 起動的に最新パージョンの有無をチェックする ② スケッチを保存する際に、拡張子をpdeから.inoに変更する ② スケッチを保存する際に、広想長子をpdeから.inoに変更する ② 検証または書き込みを行う前にスケッチを保存する 追加のポードマネージャのURL: https://dlespressif.com/dl/package_esp32_index.json 以下のファイルを直接編集すれば、より多くの設定を行うことができます。 C*USers*i + 1000 ** AppData¥Local¥Arduino IDFopreferences.txt 編集する際には、Arduino IDEを終了させておいてください。	コンパイラの警告:	なし ~		白分のドキ	コメントフ	ォル
<ul> <li>□ ¬⊢ŸŎĬŶŶĬŶĹĊĂŢŶĹ</li> <li>□ ¬ĖŸŶŶŢŶŶŶŶŢŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶŶ</li></ul>	□ 行番号を表示する					.]
<ul> <li>▲ 書き込みを探診する         <ul> <li>外部のエディタを使用する</li> <li>ゴンパイルされたコアを積極的にキャッシュする</li> <li>ジ 起動時に最新パージョンの有無をチェックする</li> <li>スケッチを保存する際に、拡張子をpdeから.inoに変更する</li> <li>ダ 検証または書き込みを行う前にスケッチを保存する</li> <li>道加のボードマネージャのURL: https://dlespressif.com/dl/package_esp32_index.json</li> <li>以下のファイルを直接編集すれば、より多くの設定を行うことができます。</li> <li>C¥Users¥=1000000000000000000000000000000000000</li></ul></li></ul>	□ コードの折り返しを有効	助に		ッーのAr	auino	
<ul> <li>□ コンパイルされたコアを積極的にキャッシュする</li> <li>□ 起動時に最新バージョンの有無をチェックする</li> <li>□ スケッチを保存する際に、拡張子をpdeから inoに変更する</li> <li>□ 検証または書き込みを行う前にスケッチを保存する</li> <li>道加のボードマネージャのURL: https://dlespressif.com/dl/package_esp82_index.json</li> <li>□ 以下のファイルを直接編集すれば、より多くの設定を行うことができます。</li> <li>C¥Users¥・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・</li></ul>	□ 外部のエディカを使用	オろ				
<ul> <li>✓ 起動時に最新バージョンの有無をチェックする</li> <li>✓ スケッチを保存する際に、拡張子をpdeから.inoに変更する</li> <li>✓ 検証または書き込みを行う前にスケッチを保存する</li> <li>追加のボードマネージャのURL: https://dlespressif.com/dl/package_esp32_index.json</li> <li>レ下のファイルを直接編集すれば、より多くの設定を行うことができます。</li> <li>C×Users¥→ → → × AppData¥Local¥Arduino 15¥preferences.txt</li> <li>編集する際には、Arduino IDEを終了させておいてください。</li> </ul>	□ 11001 15 2000 H	ッツ 積極的にキャッシュする				
<ul> <li>○ スケッチを保存する際に、拡張子をpdeから.inoに変更する</li> <li>○ 検証または書き込みを行う前にスケッチを保存する</li> <li>追加のボードマネージャのURL: https://dlespressif.com/dl/package_esp32_index.json</li> <li>以下のファイルを直接編集すれば、より多くの設定を行うことができます。</li> <li>C:¥Users¥L 1000000000000000000000000000000000000</li></ul>	□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□	aンの有無をチェックする				
☑ 検証または書き込みを行う前にスケッチを保存する 追加のボードマネージャのURL: https://dlespressif.com/dl/package_esp32_index.json 以下のファイルを直接編集すれば、より多くの設定を行うことができます。 C*Users¥ind AppData¥Local¥Arduino 15¥preferences.txt 編集する際には、Arduino IDEを終了させておいてください。	🗹 スケッチを保存する際(	に、拡張子をpdeから.inoに変更する				
道加のボードマネージャのURL: https://dlespressif.com/dl/package_esp82_index.json 以下のファイルを直接編集すれば、より多くの設定を行うことができます。 C:¥Users¥+ Lagente¥AppData¥Local¥Arduino15¥preferences.txt 編集する際には、Arduino IDEを終了させておいてください。	☑ 検証または書き込みる	を行う前にスケッチを保存する				
以下のファイルを直接編集すれば、より多くの設定を行うことができます。 C¥Users¥ Languta¥AppData¥Local¥Arduino 15¥preferences.txt 編集する際には、Arduino IDEを終了させておいてください。	追加のボードマネージャのい	JRL: https://dlespressif.com/dl/package_esp32_index.json				
C.¥Users¥LLagets¥AppData¥Local¥Arduino 15¥preferences.txt 編集する際には、Arduino IDEを終了させておいてください。	以下のファイルを直接編集	すれば、より多くの設定を行うことができます。				
	C:¥Users¥ <b>A</b> LA Arduino	pData¥Local¥Arduino15¥preferences.txt TDF54終アさせてたいてください				
	WHERE & WRITE (97, LILOUIDO	IDEGIN 1 CECUVICIACON				
$OK = \frac{1}{2} \frac{1}{2}$				OK ++++++++1		

# 2. Arduinの環境設定(参考) ・会社でプロキシ経由でネットワークを使われている場合 (但し、本日のワークショップでは、設定の必要なし)

環境設定	x	
設定ネットワーク		
<ul> <li>○ プロキジ無し</li> <li>● プロキジ設定を自動検出</li> </ul>		
ユーザ名: abc		
ホスト名:		
术		
ユーザ名:		
パスワード:	プロ	キシのURL、ユーザー名、パスワードを設定
- 4 - 4	OK キャンセル	

Arduinoボードの接続 3 (1) USBケーブルで、ArduinoボードとPCを接続 (2) ツールメニューでボードを選択 Arduino/Genuino Unoを選択 💿 sketch jul09a l Arduino 1.8.9 X ファイル 編集 スケッチ ツール ヘルプ 自動整形 Ctrl+T スケッチをアーカイブする (3) シリアルポートを選択 sketch\_jul09a エンコーディングを修正 void setup() { ライブラリを管理... Ctrl+Shift+I put your se シリアルモニタ Ctrl+Shift+M シリアルプロッタ Ctrl+Shift+L Fi101 / WiFiNINA Firmware Updater void loop() { 💿 sketch\_jul09a | Arduino 1.8.9 // put your ma ボード: "Arduino, Senuino Micro" ボードマネージャ... ファイル 編集 スケッチ ツール ヘルプ 自動整形 シリアルポート スケッチをアーカイブする ボード情報を取得 sketch\_jul09a Arduino AVRボード エンコーディングを修正 void setup() { ライブラリを管理... Arduino Yún 書込装置: "AVRISP mkll" // put your se シリアルモニタ Arduino/Genuino Uno ブートローダを書き込む シリアルプロッタ Arduino Duemilanove or Diecimila WiFi101 / WiFiNINA Firmware Updater void loop() { Arduino Nano // put your ma ボード: "Arduino/Genuino Uno" Arduino/Genuino Mega or Mega 2560 シリアルポート リアルボート ボード情報を取得 COM45 (Arduino/Genuino Uno) Arduino Mega ADK 書込装置: "AVRISP mkll" Arduino Leonardo ブートローダを書き込む Arduino Leonardo ETH Arduino/Genuino Micro Arduino Esplora Arduino Mini Arduino Ethernet - ド情報を取得するにはポートを選択してください Arduino Fio COM33のArduino/Genu

## 4. プログラミングスタート

 次のURLにアクセス (Arduino 日本語リファレンス) <u>http://www.musashinodenpa.com/arduino/ref/</u> Arduino言語を参照
 <Arduino言語>
 setup()
 Loop()

を参照

### ARDUINOの起動

🥺 sketch_jul09a   Arduino 1.8.9 ファイル 編集 スケッチ ツール ヘルプ	-8	×
		ø
sketch_jul09a		
<pre>void setup() {     // put your setup code here, to run once:</pre>		^
}		
<pre>void loop() {     // put your main code here, to run repeatedly:</pre>		
}		
		~
ボード情報を取得するにはボートを選択してください		

COM33のArduino/Genuino Uno

## 基本的なプログラミング言語内容

- 基本的な文法を参照。→ ;、 {}、 コメント
- データ型  $\rightarrow$  byte、int、unsigned int、float
- ●算術演算子 → 四則演算
- 制御文 → if、if else、for、while

### 5. LEDを光らせよう1

- デジタル入出力関数 pinModeとdigitalWriteを参照
- 参考として、次のURLにアクセス(Arduinoチュートリアル基礎編) <u>http://www.musashinodenpa.com/arduino/ref/index.php?f=2</u>
- <Arduinoボードの信号入出力について>
- Arduinoボードを手に取って見てみる
- ●デジタル信号の入出力 → 0番から13番
- アナログ入力 → A0番からA5番 (デジタル入出力としても可)
- アナログ出力(PWM信号) → デジタル信号の~の付いたピン



## 発光ダイオード駆動回路

		The second of the second secon		E V	
		デジタル信号		V C	
ARDUINO	digitalWir	e(ピン番号,HIGH)	0.1		
	digitalWir	e(ピン番号,LOW)			
デジタル出力端子	出力電圧 V o u t				
13番					
	►	抵抗の両端子間電圧	Δ V = 出力電圧	きい電圧=Vout-Vf	
	流れる電流Ⅰ		<発光ダイオードの	明るさ>は、流れる電流丨で	決まる
	アノード、発光ダ	イオード 个	電流   = 打	抵抗に流れる電流!=抵抗の	両端子間電圧ΔV/抵抗値R
	I I <del>`</del> }			(Vout−Vf)∕R	
	<i><b>DY</b>+F</i>	発光ダイ:	オードのしきい値電圧	V f	
		$\checkmark$			
	<u> </u>		Ĩ	20 m A	
/		·····	V o u t	5 V	
ーフォーラムのWo 発光ダイオードの	rkShopのへー )データシート	シからダワシロート	V f	2 V	
(内容確認)			R	?	

### 回路の設計手順

<目的 ダイオードを光らせる>
(1) 仕様を決める → 色、明るさ、方向、点滅間隔
(2) 回路概要を決める → ダイオード、抵抗
(3) 使用する部品をデータシートを見て決める → ダイオード
(4) 部品の定数を算出する → 抵抗値
(5) 詳細な回路を作成する

場合によっては、(1)から(5)の間でループを繰り返す データシートは、部品の選定時のネット注文会社(DIGIKEY、MOUSER、RSコ ンポーネンツ、秋月電子など)の部品ページある

回路が決まったら、Arduinoでプログラミング

### ブレットボードの配線

• 必ず、USB接続を外してから配線作業を行う事。



極性に注意 逆に接続すると定格の5Vになる。 (USBからの給電電圧に依存) 場合によっては、故障。







## 発光ダイオードのプログラム

- (1) pinMode()とdigitalWrite()のピン番号は、後で変更を容易にす る為、最初に変数定義で、int(整数型)で定義。
  - (2) Setup()で、pinModeで指定ピンを出力に設定。

(3) loop()で、digitalWriteでHIGH出力とLOW出力を1秒毎 (delay) に設定した時間間隔で、トグルさせる。

(4) ファイルメニューの下の検証(コンパイル)とマイコンボードに





### ワーストケースの考え方

- 発光ダイオードのデーターシートの絶対最大定格から、ダイオードに 流す事の出来る最大電流は、30mA。
- 仮に、USB電源が+5%が上限、発光ダイオードのVfのmin値は、1.8V、抵抗値が、-20%下限になると
  - I = (5. 25 1. 8) / 120 = 28. 75 mA
- → 仕様に対して、最悪条件(ワーストケース)を考える事は重要 <対策>
- •抵抗値の誤差を5%以内とする。
- •抵抗値自体を少し大きい値にする。→ 逆のケースで、暗くなる

目標となる仕様には、ある一定値ではなく、ある範囲という事になる

### 6. デジタル信号とは

デジタル信号出力のLOWを'O'、
 HIGHを'1'として、2進数で考える。

- 10進数と2進数(16進数)の
   対応表 →
- デジタル信号が、伝わるのは、
   1本の信号ライン(電線)に、
   0、1の信号のみ → 1ビット
- 8ビットには、信号ライン8本
   16ビットには、信号ライン16本

10進数	2 進数	べき乗	16進数			
0	0000	2^0	0			
1	0001		1			
2	0010	2^1	2			
3	0011		3			
4	0100	2^2	4			
5	0101		5			
6	0110		6			
7	0111		7			
8	1000	2^3	8			
9	1001		9			
10	1010		а			
11	1011		b			
12	1100		С			
13	1101		d			
14	1110		е			
15	1111	2^4-1	f			
16	10000	2^4	10			
255	11111111	2^8-1	ff			

### ビット演算について

- Arduino日本語リファレンスのビット演算子を参照のこと。
   ビットで演算した方が簡単な場合がある。
- C言語では、次の複合演算子もよく用いる。

## 7.アナログ信号とは

•時間的に信号ラインの中で、電圧(アナログ量)が変動していくもの



但し、ビット演算などの処理はできない

### アナログ→デジタル変換について

<アナログ信号からデジタル信号への変換器(ADコンバータ)> ある一定の時間間隔で、アナログ信号からデジタル信号へ変換する その時間間隔をサンプリング周期(逆数をサンプリング周波数) また、AD変換器が動作できる時間間隔以上の時間であれば、不定期な時間間隔で、デジタル 信号に変換しても良し。



### 8. LEDを光らせよう その2



# ブレッドボードの配線 その2

• 各自で配線してみよう。





## アナログ入力のプログラム

- Arduino日本語リファレンスのアナログ入出力関数を参照。
- ADコンバータは、10ビット。従って、0から1023までの値を 出力する。アナログ入力範囲は、0から電源(5V)まで。
- ●抵抗ボリュームは、0から電源(5V)まで、出力電圧が、変化。

## プログラムの仕様

ボリュームの出力電圧をAD変換した10ビットのデータの内、
 上位4ビットのデータを、そのまま10番から13番までの
 発光ダイオードで表す。また、大体1秒毎にAD変換を行う。

<使うプログラム> アナログ入力、 ビット演算子、 制御文(ifなど)

ADコンパータ		Ţ						
10 ビ	ע א ר	101	1001	₽Ø			1011	の場合
	1	最上位	1の時 1	3番L E	Dを点	灯	点灯	
		次のビッ	ト1の時	12番L	EDを	点灯	消灯	
		次のビッ	ト1の時	11番L	EDを	点灯	点灯	
		次のビッ	ト1の時	10番L	EDを	点灯	点灯	

プログラム作成してみよう

### 体験できた事

- AD変換のレートが早い(サンプリング周波数が高い)→ アナログの変動に対する追従性が良好。
- ・ボリュームがある一定以上変化しないと発光ダイオードが変わらない。
   → ビット分解能が低い

AD変換に重要な項目  $\rightarrow$  ビット分解能、変換レート 但し、ビット分解能が細かくても、 ビット間が等間隔  $\rightarrow$  INL、DNL、ひずみ率 (Distortion) ノイズの影響  $\rightarrow$  SN比

### 9. センサーのデジタルインターフェース

- 最近のセンサーデバイスは、AD変換を内臓して、出力がすでにデジタル信号になっているものが多い。
- 出力が、10ビットだと、10本のデジタル信号線が必要となり、効率的でない。
- 従って、デジタル出力信号を時間的に分けて送るシリアル伝送(イン ターフェース)が主流になっている。
- センサーなどの用途とマイコン通信用途には、次の3種類のインター フェースが使われている。

SPI, UART, I2C

#### < S P I >



<1個のマスターチップに複数のスレープチップを接続する事が可能>



チップセレクトが、例えば、'し'であるチップのみシリアルーパラエル変換する

### SPIインターフェースの波形例



### 11. 温度センサーを動かそう

 SPIインターフェースを使って温度センサーモジュール(ADT7 310)を動かす。

• ボード配線を行う。

#### ■基板外観と端子配置







### データシートについて

デバイスの特性をまとめたもので、世界共通語
 <構成>

1. 用途、概要、目次

2. ブロック図、回路構成

- 2. ピン仕様(ピン配置図、ピン内容、パッケージ図)
- 3.特性説明(絶対最大定格、動作条件、電気的特性、 デジタル特性(AC、DC、タイミングチャート)

4. 機能詳細説明

- 5. レジスタ説明(レジスタマップ全体、各レジスタ項目)
- 6. パッケージ寸法、リフロー条件

7. 推奨外部接続図例

8. 注意事項(損害、軍事利用、用途外使用、法令)

### ADT7310のデーターシート

ADT7310のデーターシートの内容確認

#### レジスタマップ

Table 6. ADT7310 Registers

Register Address	Description	Power-On Default
0x00	Status	0x80
0x01	Configuration	0x00
0x02	Temperature value	0x0000
0x03	ID	0xCX
0x04	T <sub>CRIT</sub> setpoint	0x4980 (147°C)
0x05	T <sub>HYST</sub> setpoint	0x05 (5°C)
0x06	Thigh setpoint	0x2000 (64°C)
0x07	TLOW setpoint	0x0500 (10°C)

#### 温度と出力コードの関係

Table 5. 13-Bit Temperature Data Format

Temperature	Digital Output (Binary) Bits[15:3]	Digital Output (Hex)
–55°C	1 1100 1001 0000	0x1C90
–50°C	1 1100 1110 0000	0x1CE0
–25°C	1 1110 0111 0000	0x1E70
-0.0625℃	1 1111 1111 1111	0x1FFF
0°C	0 0000 0000 0000	0x000
+0.0625°C	0 0000 0000 0001	0x001
+25°C	0 0001 1001 0000	0x190
+50°C	0 0011 0010 0000	0x320
+125°C	0 0111 1101 0000	0x7D0
+150°C	0 1001 0110 0000	0x960

#### 温度の連続読みだしモードを使うので、その部分のデーターシートを参照

### ADT7310のSPIレジスタ

#### Table 6. ADT7310 Registers

Register Address	Description	Power-On Default
0x00	Status	0x80
0x01	Configuration	0x00
0x02	Temperature value	0x0000
0x03	ID	0xCX
0x04	T <sub>CRIT</sub> setpoint	0x4980 (147°C)
0x05	T <sub>HYST</sub> setpoint	0x05 (5°C)
0x06	Thigh setpoint	0x2000 (64°C)
0x07	T <sub>LOW</sub> setpoint	0x0500 (10°C)



#### Table 15. Command Byte

C7	C6	C5	C4	C3	C2	<b>C1</b>	CO
0	R/W	Reg	gister ac	ldress	Continuous read	0	0



Figure 20. Read from an 8-Bit Register

### 温度センサーのプログラミング

- 今回は、連続で温度データを読み取れるようにする。
- (1) #includeで<SPI.h>を定義
- <setup>
- (2) 10番ピンを出力に定義して、まず、HIGHを出力
- (3) SPI. beginでスタート
- (4) SPIモード、MSBファースト、クロック分周を定義
- (5) 10番ピンをLOW出力(チップセレクト状態)
- (6) SPI. transfer(0x54)で、連続読みだしの命令をレジスタに設定
- (7) 240mS待つ

#### $<\!$ loop>

- (8) int val; float temp;を定義
- (9) val=SPI. transfer(0)で8ビットを読み出して、valの上位1バイトとする(8ビットシフト)
- (10) val=SPI. transfer(0)で同様に読み出して、valの下位1バイトとする(ORと取る)
- (11) 浮動小数点変数tempにvalを16.0で割った値を入れる
- (12) 1秒待つ

### 12. UARTインターフェース

#### SPIと異なり、対等な1対1のシリアル通信





### 13. UARTを使ってみよう

- BLEなどの通信モジュールのインターフェースに使われている
- また、USB接続を介したPCとARDUINOのデータのやり取り にも使われている
- Arduino日本語リファレンスのシリアル通信を参照のこと。
- 今回は、Serial.printを使う。

## 温度センサーのプログラミング 追加

- 今回は、連続で温度データを読み取れるようにする。
- (1) #includeで<SPI.h>を定義
- <setup>
- (2) 10番ピンを出力に定義して、まず、HIGHを出力
- (3) SPI. beginでスタート
- (4) SPIモード、MSBファースト、クロック分周を定義
- (5) 10番ピンをLOW出力(チップセレクト状態)
- (6) SPI. transfer(0x54)で、連続読みだしの命令をレジスタに設定
- (7) 240mS待つ

#### $<\!$ loop>

- (8) int val; float temp;を定義
- (9) val=SPI. transfer(0)で8ビットを読み出して、valの上位1バイトとする(8ビットシフト)
- (10) val=SPI. transfer(0)で同様に読み出して、valの下位1バイトとする(ORと取る)
- (11)浮動小数点変数tempにvalを16. 0で割った値を入れる
- (12) 1秒待つ

Serial.print("temp = ");
Serial.println(temp,2);

Serial.begin(115200)

### ARDUINOでのシリアル通信モニター

COM46

sketch_Temp   Arduino 1.8.9 レ 編集 スケッチ ツール ヘルプ 自動整形 スケッチをアーカイブする Etch_Temp Super Sensing マクリアルモニタ	Ctrl+T ବ : Ctrl+Shift+I	Temp = 16.50 Temp = 16.00 Temp = 16.00 Temp = 16.50 Temp = 16.50 Temp = 16.50	
集 スケッチ ツール ヘルプ 自動整形 スケッチをアーカイブする Temp Temp Temp Fr Sensing Vino Work Vino Work	Ctrl+T ବ : Ctrl+Shift+I	Temp = 16.50 Temp = 16.00 Temp = 16.00 Temp = 16.50 Temp = 16.50 Temp = 16.50	
自動整形 スケッチをアーカイブする エンコーディングを修正 ライブラリを管理 Sensing シリアルモニタ	Ctrl+T ବ : Ctrl+Shift+I	Temp = 16.00 Temp = 16.00 Temp = 16.50 Temp = 16.50 Temp = 16.50	
-Temp エンコーディングを修正 ライブラリを管理 r Sensing シリアルモニタ	Ctrl+Shift+I	Temp = 16.50 Temp = 16.50	
シリアルモニタ		Temp = 16.50	
A REPORT OF A REPO	Ctrl+Shift+M	Temp = 16.50	
/18 Y.K シリアルプロッタ	Ctrl+Shift+L	Temp = 16.50 Temp = 16.00	
<spi.h> WiFi101 / WiFiNINA</spi.h>	A Firmware Updater	Temp = 16.00 Temp = 16.50	
10; ボード: "Arduino/Ger シリアルポート	nuino Uno" >		
() { <mark>ポード情報を取得</mark> CSB, OU			
rite(C 書込装置: "AVRISP r	mkll"		
/in(); ブートローダを書き込む	3		

### 14. I2Cインターフェース

< I 2 C >			
	スレーブチップは、複数接続可能		
マスターチップ	スレープチップ		
デジタル入出力シリアルクロックSCL ◆ SC	CL デジタル入出力シリアルクロックSCL	スタート スレーブアドレス ロンディション 0xA0 アク //havi	データ 0x5A アク 川から
デジタル入田JJ ジリアルテーダSDA -	「 デジダル人田」」 ジリアルテーダSDA	V	
		SDA 1 0 1 0 0 0 0 0	
電源	スレープチップ		
─────────────────────────────────────	→ デジタル入出カシリアルクロックSCL		
$\nabla$	● デジタル入出力 シリアルデータSDA	sci in n n n n n n n	
抵抗 <			
▶抵抗 <			1 1 1
	スレープチップ		
	→ デジタル入出カシリアルクロックSCL		
	◆───→ デジタル入出力 シリアルデータSDA		
SCL、SDAともに、プルアップ抵抗が必要			
SPIインターフェースのように、チップセレクト(	言号がないが、スレーブチップを区別する必要がある		
マスターナップか送信するシリアルテータに、スレー	-フチップ毎の区別テータ(スレーフアドレス)を最初に	入れる	

### I 2 C インターフェース



#### I2CのArduino int i2c\_ int di

#include <Wire.h>

```
#define i2c_add 0x23
```

```
void i2c_write(int adr,int wdata) {
    int st;
    Wire.beginTransmission(i2c_add);
    Wire.write(adr);
    Wire.write(wdata);
    st = Wire.endTransmission();
    while (st != 0) {
        st = Wire.endTransmission();
        delay(10);
    }
```

```
int i2c_read(int adr) {
    int dread;
    int st;
    Wire.beginTransmission(i2c_add);
    Wire.write(adr);
    st = Wire.endTransmission();
    while (st != 0) {
        st = Wire.endTransmission();
        delay(10);
    }
}
```

```
Wire.requestFrom(i2c_add,1);
dread = Wire.read();
st = Wire.available();
while (st !=0) {
   st = Wire.available();
   delay(10);
}
return dread;
```

```
void setup() {
    int rdata;
```

```
i2c_write(0x00,0x11);
```

```
rdata = i2c_read(0x22);
```

### 15. 光センサーを動かしてみよう

- ・光センサーモジュール I2Cインターフェースで動かす
- センサーは、BH1750(ローム社製) データーシートを参照の事
  モジュールの回路図は以下の通り





![](_page_51_Picture_1.jpeg)

![](_page_52_Picture_0.jpeg)

## GitHubライブラリーを使用する方法

- 世の中にあるプログラムのソースファイルは、GitHubというサイトに 公開されている。活用すると簡単にプログラミングする事ができる。
- •検索すると、大抵のセンサーに対するプログラム例を見つけられる。
- 基本的に、GitHubからライブラリーとしてインクルードする。
- 以下のURLからライブラリーをダウンロード

https://github.com/claws/BH1750

今回は、フォーラムからZIPファイルをダウンロード

			oo :	sketch_jul17a   Arduino 1.8.9		– 🗆 X	
			774)	ル 編集 スケッチ ツール ヘルス	1	na	zor <b>a</b> Amazor 🕀 GY-
				後証・コンパイル	Ctrl+R	₽ P	
			sk	ketch_jul 書込装置を使。	吉さ込む Ctrl+U って書き込む Ctrl+Shift+U	🖬 🧧	hub.com/claws/BH1750
_			#in	nclude < Jy/(イルした/	イナリを出力 Ctrl+Alt+S	^	
		<u></u>		id setur スケッチのフォル	ダを表示 Ctrl+K	plo	ore – Marketplace P
( -1 +		$ \neg  =b$		// put y ライブラリをイン	クルード -	1	2
UIU		י עע ע	. 以了,	ファイルを追加		ライブラリを管理	Ctrl+Shift+I
						.ZIP形式のライブラリをインスト	-Jl
			voi	id loop() { // put your main code h	ere, to run repeated]	Arduino ライブラリ	
					8 Å	Bridge	
sketch_jul17	a   Arduino 1.8.9	– 🗆 🗙	}			EEPROM	
ファイル 編集 スク	アッチ ツール ヘルプ	SitHub? \	Enterprise			Esplora	
	検証・コンパイル Ctrl+R	Sitt tub.	Enterprise			Ethernet	
JO L		1921 - Landard				Firmata	
sketch jul						GSM	
	音込装直を使う(音き込む Ctrl+Sniπ+0 この。	A 11750				Keyboard	
// put v	コンハイルしたハイナリを出力 Ctrl+Alt+S					LiquidCrystal	
// pat 1	スケッチのフォルダを表示 Ctrl+K					Mouse	
	ライブラリをインクルード		Chul - Shift - I			Robot Control	
	ファイルを追加	フィノフラを皆注	Ctri+Snint+I			Robot IR Remote	
roid loop(		.ZIP形式のライブラリをインストール				Robot Motor	
// put you	r main code nere, to run repeated.					SD	
}		Arduino ライブラリ				SPI	
		Bridge				Servo	
		EEPROM				SpacebrewVup	
		Esplora	51	「ブラリが追加されました。」	「ライブラリをインクルート	Stepper	
		Ethernet				TFT	
		Firmata				Temboo	
		GSM				WiFi	
					cc	Wire	
				nt other	examples	提供されたライブラリ	
				Contraction of the second	resources	Adafruit_VL53L0X	
				Note that is a		BH1750	

ALS CONTRACT

E) aitianore

## プログラミング

sketch\_Light

\* Super Sensing Forum \* Arduino Work Shop \* 2019/7/18 Y.K \*/

#include <Wire.h>
#include <BH1750.h>

BH1750 lightMeter;

void setup() {

Serial.begin(9600);
Wire.begin();
lightMeter.begin();

8

void loop() {
 float lux;

lux = lightMeter.readLightLevel(); Serial.print("Light = "); Serial.print(lux); Serial.println(" lx"); delay(1000);

## 明るさLUXについて

- 照度の単位として、
- 1m<sup>2</sup>当たり1ルーメンであれば、1ルクス
- ・明るさの目安としては、

照度(ルクス)	明るさの目安	(ルクス)
100,000	<ul> <li>・雪山・真夏の海岸</li> <li>・晴天昼太陽光</li> <li>・晴天午前10時太陽光</li> <li>・晴天午後3時太陽光</li> <li>・曇天昼太陽光</li> </ul>	$\begin{array}{c} > 1 \ 0 \ 0, \ 0 \ 0 \ 0 \\ 1 \ 0 \ 0, \ 0 \ 0 \ 0 \\ 6 \ 5, \ 0 \ 0 \ 0 \\ 3 \ 5, \ 0 \ 0 \ 0 \\ 3 \ 2, \ 0 \ 0 \ 0 \end{array}$
10,000	25,000	
1,000	<ul> <li>・晴天日入1時間前太陽光</li> <li>・パチンコ店内</li> <li>・百貨店売場</li> <li>・蛍光灯照明事務所</li> <li>・日出入時</li> <li>・30W蛍光灯2灯使用八畳間</li> <li>・夜のアーケード</li> </ul>	$ \begin{array}{c} 1, 000\\ 1, 000\\ 500 \sim 700\\ 400 \sim 500\\ 300\\ 300\\ 150 \sim 200 \end{array} $
100	・街灯下 ・ライター@30cm	50~100 15
1 0	・ロウソク@20cm ・市民薄明(太陽天頂距離96度)	1 0~1 5 5
1	<ul> <li>・月明り</li> <li>・航海薄明(太陽天頂距離102度)</li> <li>・天文薄明(太陽天頂距離108度)</li> </ul>	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$

### まとめ

- LEDを用いてライティング(アナログ的にコントロール)
- 部品には、データーシートがあり、それを用いて設計する
- アナログ信号について サンプリングと分解能
- 大抵の場合は、相反する仕様項目がある
- センサーのデジタルインターフェース SPI、UART、I2C
- それらを使ってプログラミング GitHubの活用

### 今後の活用

- 温度センサーで、いろいろなものを測定
   部屋の場所による違い、ペット等の居心地
- ・光センサーで、いろいろなものを測定
   ベランダ(洗濯)の日光照射量把握
- 有線でなく、無線を使う
   無線モジュールとしては、UARTインターフェースが多い