Armadillo-640
開発基礎セミナー

第6部 拡張I/Fを使った開発の流れ（その1）
（ハードウェア、カーネル編）

株式会社アットマークテクノ
http://www.atmark-techno.com/

目次

Armadilloにおける開発の要素

- **ハードウェア**
  - 拡張I/Fを使った拡張基板 → **GPIO接続のSWとLED**
    I2C接続の加速度センサー
- **カーネル/デバイスツリー**
  - デバイスドライバの適用 → **I2Cドライバの有効化確認**
  - 拡張I/Fの端子機能の設定 → **I2C端子の有効化**
- **アプリケーション**
  - 既存のパッケージのインストール → **C言語ビルド環境**
    Pythonパッケージ
  - 独自のアプリケーションの作成
    → **SWからデータを取得してLEDへ出力するプログラム**
    照度センサーからデータを取得するプログラム

この章の概要

- **ハードウェア**
  - 拡張I/Fを使った拡張基板 → **GPIO接続のSWとLED**
    I2C接続の加速度センサー
- **カーネル/デバイスツリー**
  - デバイスドライバの適用 → **I2Cドライバの有効化確認**
  - 拡張I/Fの端子機能の設定 → **I2C端子の有効化**
- **アプリケーション**
  - 既存のパッケージのインストール → **C言語ビルド環境**
    Pythonパッケージ
  - 独自のアプリケーションの作成
    → **SWからデータを取得してLEDへ出力するプログラム**
    照度センサーからデータを取得するプログラム

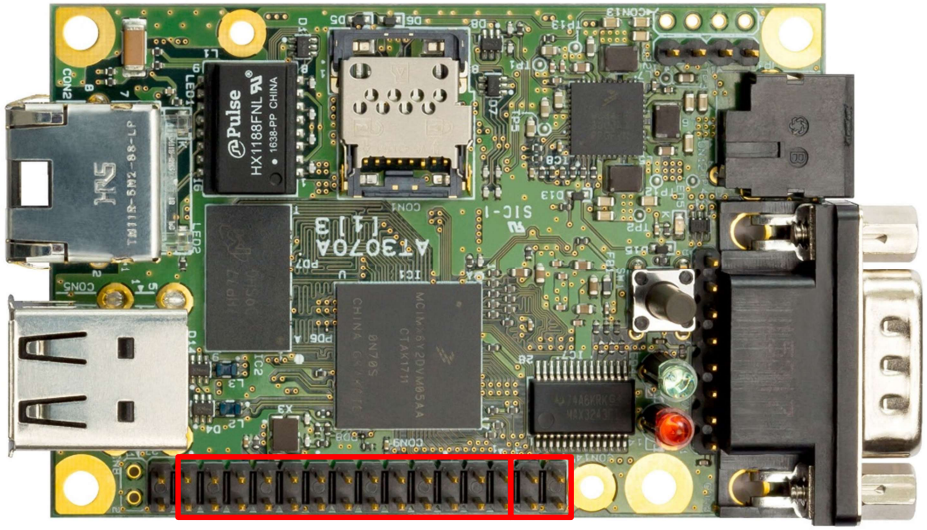
ハードウェア

トグルスイッチとLEDの接続

I2C接続の加速度センサーとの接続

Armadillo-640の拡張I/F

- Armadilloの拡張I/Fは様々な機能を持っています。



CON9　　CON14

CON9：GPIO、I2C、SPI、UART、SD等
CON14：GPIO、I2C、UART等

- 各機能のドライバはカーネルコンフィギュレーションで適用します。
- 各機能の端子設定はデバイスツリーで行います。

端子の確認方法

- **Armadillo-640マルチプレクス表（※）で端子を確認します。**
  ※）Webサイトよりダウンロードできます。

GPIOに使用する端子

- **Armadillo-640マルチプレクス表では、下記のような表記です。**

  CON9_13ピン：GPIO3_IO23
  CON9_28ピン：GPIO4_IO09

GPIO

部品番号	ピン番号	信号名	ピン名	電圧グループ	own	GPIO
CON9	12	PWRON	PWRON	VDD_SNVS_IN		
	13	GPIO3_IO23	LCD_DATA18	VCC_3.3V		GPIO3_IO23
	14	GPIO3_IO24	LCD_DATA19	VCC_3.3V		GPIO3_IO24
	15	GPIO3_IO25	LCD_DATA20	VCC_3.3V		GPIO3_IO25
	16	GPIO3_IO26	LCD_DATA21	VCC_3.3V		GPIO3_IO26
	17	GPIO3_IO27	LCD_DATA22	VCC_3.3V		GPIO3_IO27
	18	GPIO3_IO28	LCD_DATA23	VCC_3.3V		GPIO3_IO28
	19	GND				
	20	VCC_3.3V				
	21	USB2_DN	USB_OTG2_DN	-		
	22	USB2_DP	USB_OTG2_DP	-		
	23	USB2_VBUS	USB_OTG2_VBUS	USB_VBUS		
	24	EXT_USB_EN	-	VCC_3.3V		
	25	GPIO4_IO06	NAND_DATA04	VCC_3.3V		GPIO4_IO06
	26	GPIO4_IO07	NAND_DATA05	VCC_3.3V		GPIO4_IO07
	27	GPIO4_IO08	NAND_DATA06	VCC_3.3V		GPIO4_IO08
	28	GPIO4_IO09	NAND_DATA07	VCC_3.3V		GPIO4_IO09

I2Cに使用する端子2

- **Armadillo-640マルチプレクス表では、下記のような表記です。**

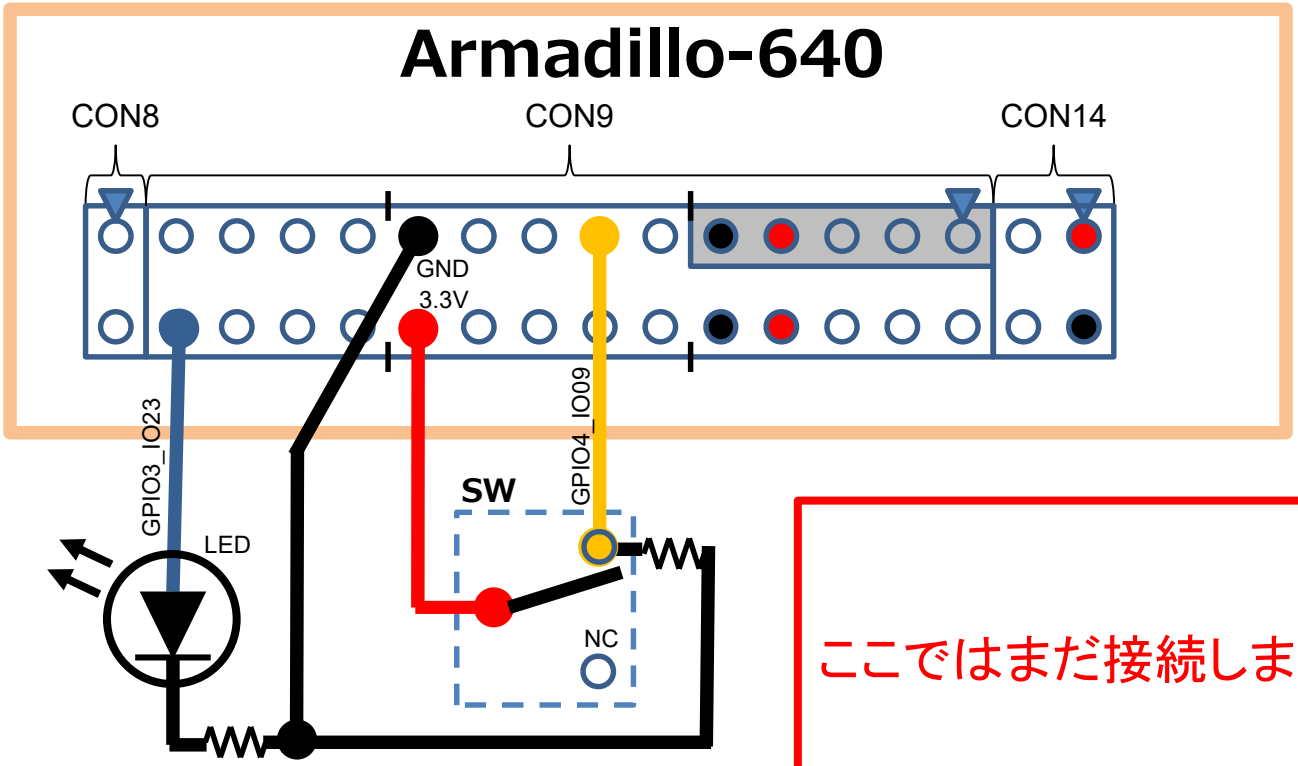
CON9_4ピン：I2C2_SDA
CON9_6ピン：I2C2_SCL

I2C2

部品番号	ピン番号	信号名	ピン名	電圧グループ	マルチプレクス機能 I2C2	I2C3
	4	GPIO1_IO21	UART2_RX_D	VCC_3.3V		
CON9	1	GPIO1_IO22	UART2_CTS	VCC_3.3V		
	2	GPIO1_IO23	UART2_RTS	VCC_3.3V		
	3	GPIO1_IO17	UART1_RX_D	VCC_3.3V		I2C3_
	4	GPIO1_IO31	UART5_RX_D	VCC_3.3V	I2C2_SDA	
	5	GPIO1_IO16	UART1_TX_D	VCC_3.3V		I2C3_
	6	GPIO1_IO30	UART5_TX_D	VCC_3.3V	I2C2_SCL	
	7	VCC_3.3V				
	8	VCC_3.3V				
	9	GND				
	10	GND				

Armadillo-640
トグルスイッチとLEDの接続

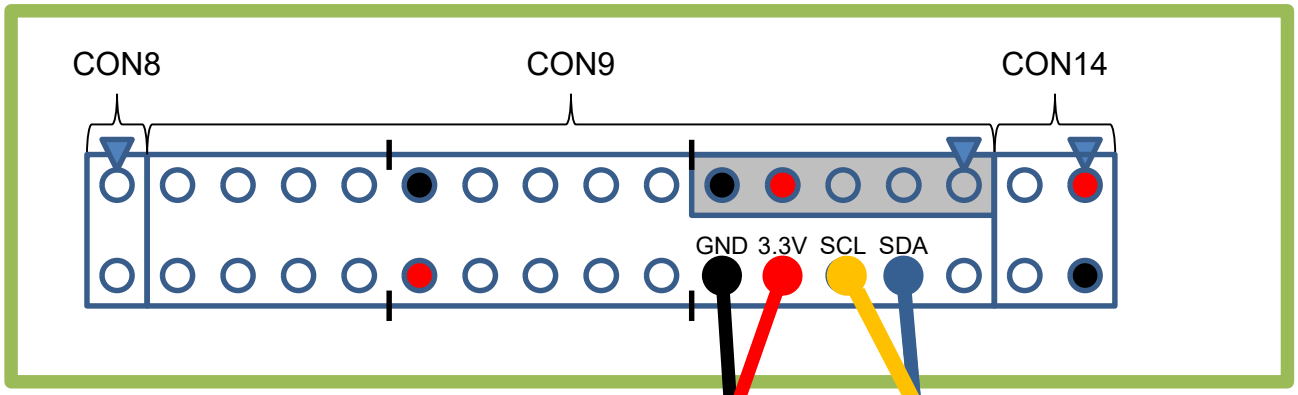
- Armadillo-640とトグルSWとLEDを下記のように接続します。



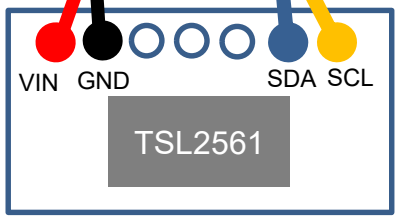
ここではまだ接続しません。

Armadillo-640と照度センサー(TSL2561)の接続

- Armadillo-640と照度センサー（TSL2561）搭載モジュールを下記のように接続することになります。



ここではまだ接続しません。



デジタル光センサ
搭載モジュール

カーネル/デバイスツリー

I2Cドライバの設定

カーネルのカスタマイズについて

- Armadillo-640のカーネルをカスタマイズする場合は、ATDE上に展開したカーネル（第4章で実施）のソースコードで行います。

- ATDEで、カーネルのソースコードのディレクトリに移動します。

```
atmark@atde7:~$ cd linux-v4.14-at5
```

- カーネルの設定を「デフォルト」にするには、下記のコマンドを実行します。

```
atmark@atde7:~/linux-v4.14-at5$ make ARCH=arm armadillo-640_defconfig
```

このコマンドは、初めてカーネルをビルドする場合や、意図的にデフォルトに戻す場合に実行します。
カスタマイズ後は、このコマンドは実行する必要はありません。

カーネルコンフィギュレーション
I2Cドライバの設定1

- ここでは、カーネルコンフィギュレーションの用法として、
  I2Cドライバの設定を確認してみます。

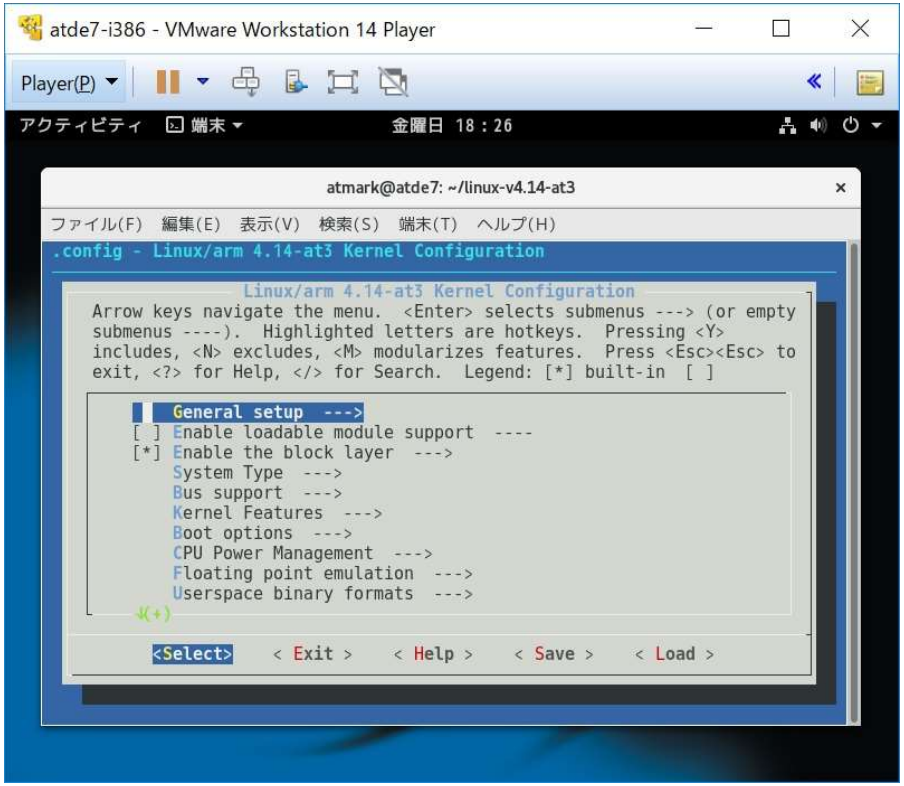
- なお、Armadillo-640のカーネルコンフィギュレーションの
  デフォルトでは、I2Cドライバは有効になっています。
  よって、ここでは、設定変更は行わず、設定確認のみを行います。

カーネルコンフィギュレーション
I2Cドライバの設定2

- 下記コマンド（make menuconfig）を実行すると、

```
atmark@atde7:~/linux-v4.14-at5$ make ARCH=arm menuconfig
```

下記のようなカーネルコンフィギュレーションの画面が開きます。

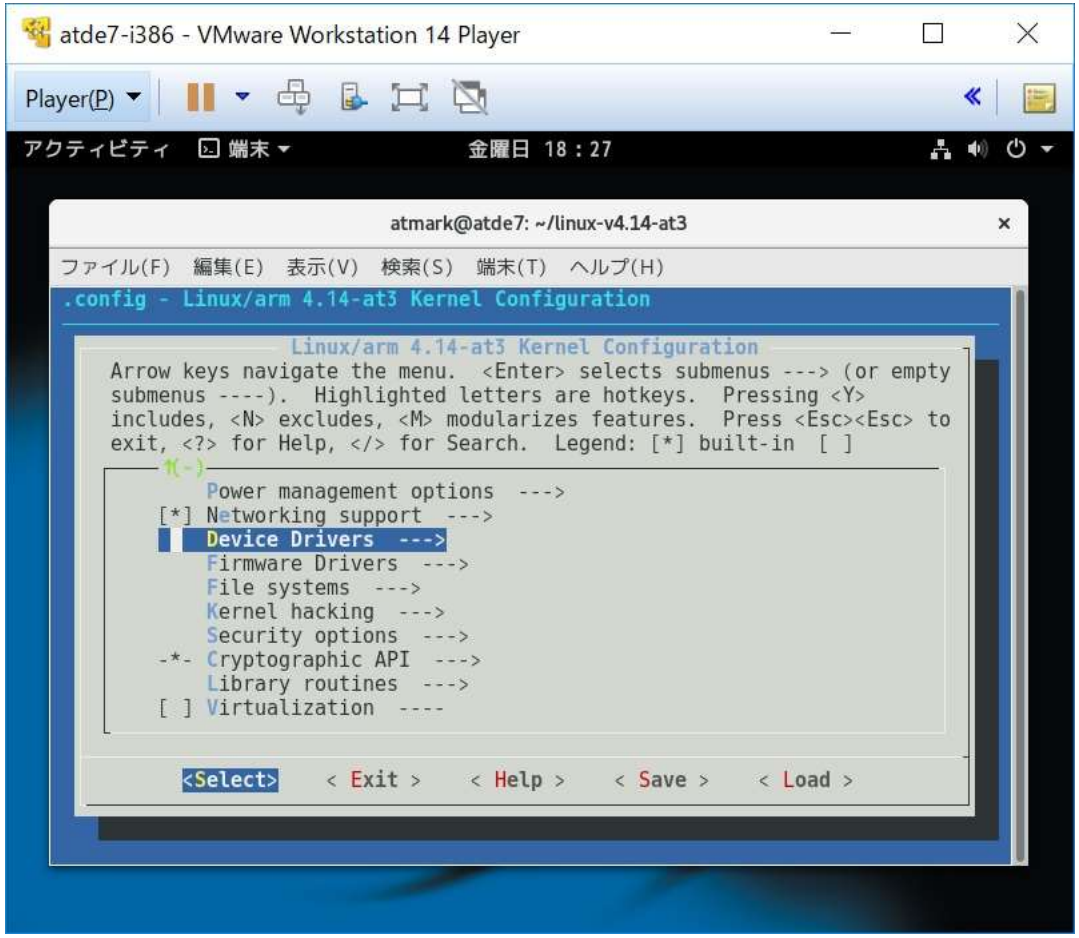


参考）カーネルコンフィギュレーション操作方法

- **Menuconfigでの操作方法**
  - 上下キー: メニューの選択
  - 左右キー: 動作の選択
  - スペースキー: オプションの選択
  - Enterキー: 動作の決定
  - yキー：*を付ける。（選択中の設定を有効にする）
  - nキー：*を外す。（選択中の設定を無効にする）

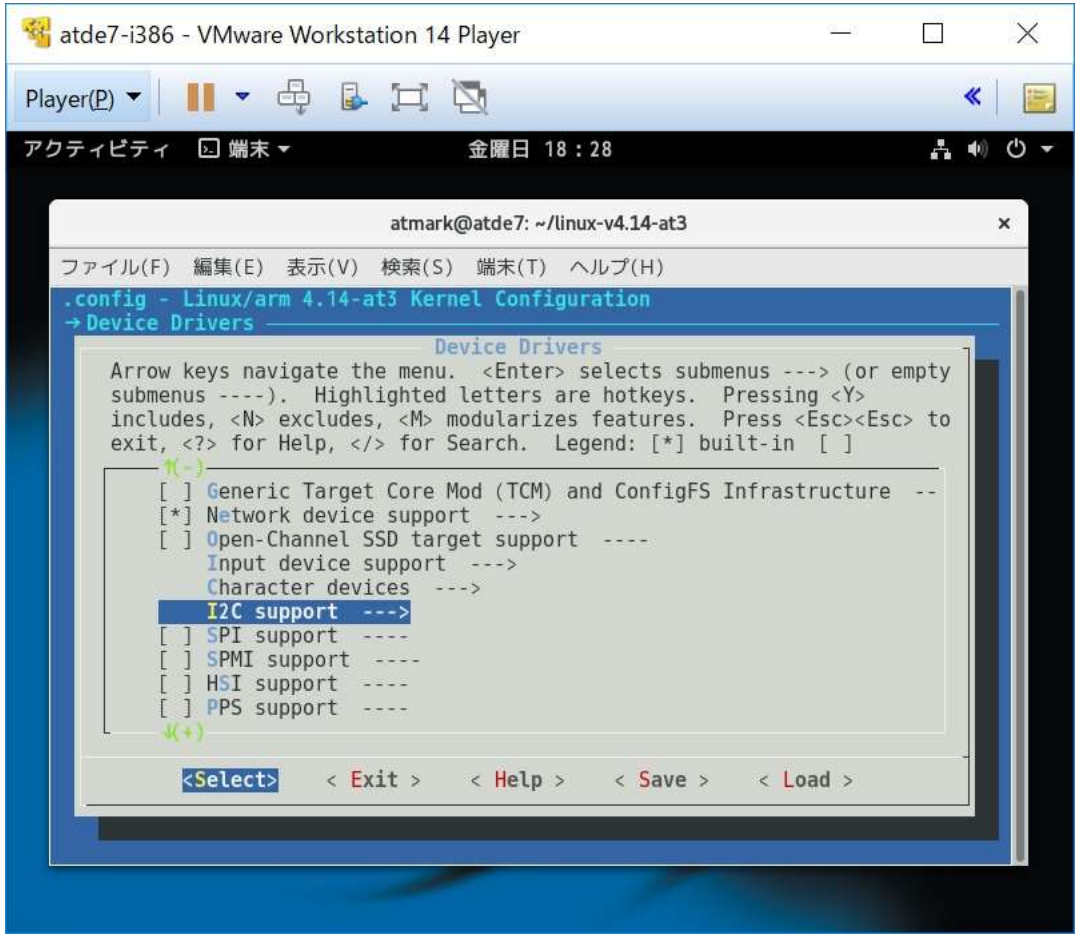
カーネルコンフィギュレーション
I2Cドライバの設定3

- Device Driversにカーソルを合わせて<Select>でEnterします。



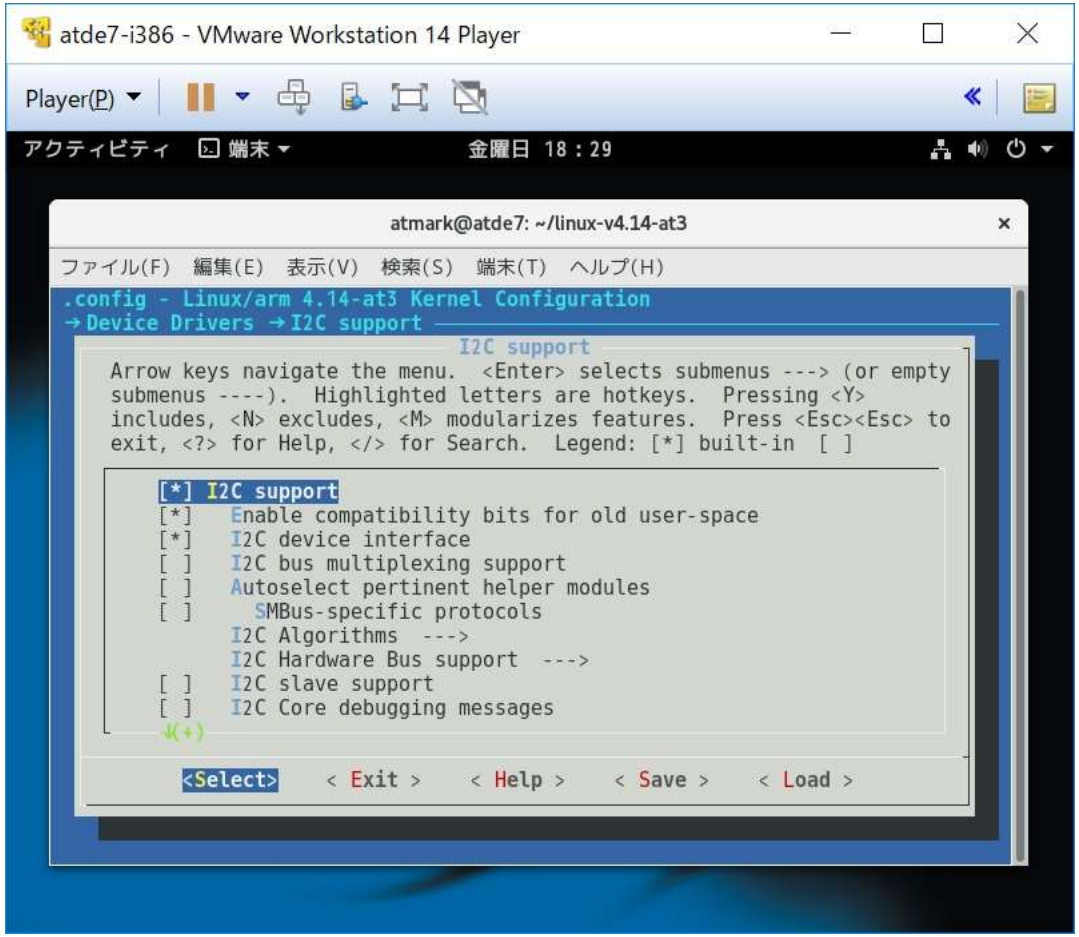
カーネルコンフィギュレーション
I2Cドライバの設定4

- I2C supportにカーソルを合わせて<Select>でEnterします。



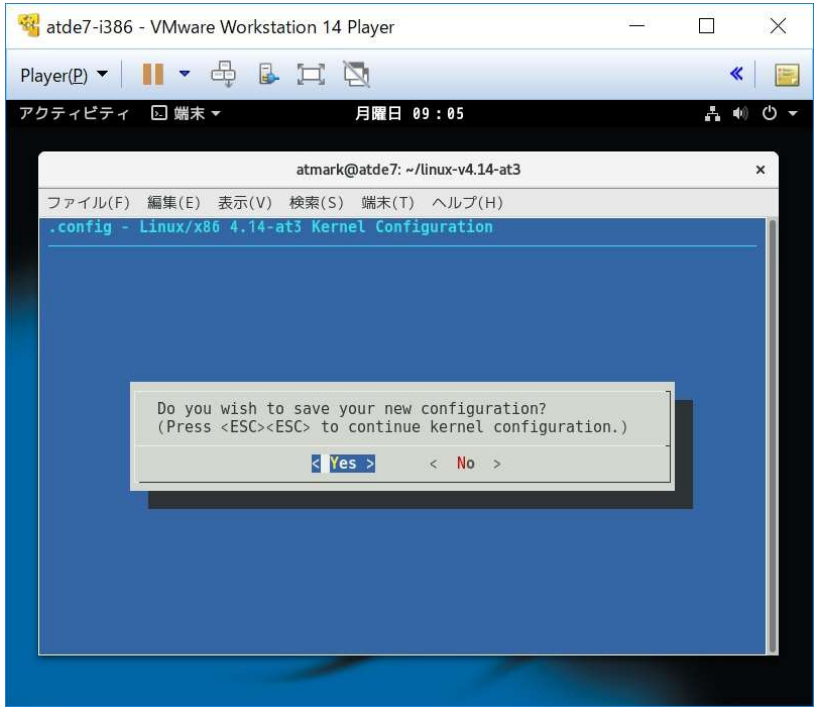
カーネルコンフィギュレーション
I2Cドライバの設定5

- I2C support内の設定が[*]（有効）になっていることを確認します。



カーネルコンフィギュレーション
I2Cドライバの設定6

- カーネルコンフィギュレーションが終了するまで、<Exit>でEnterします。
- ここでは、設定変更しませんでしたが、設定変更した場合は、下記のように、設定保存の確認があります。



参考）カーネルコンフィギュレーション 製品マニュアル等の記述

- **カーネルコンフィギュレーションの設定は、製品マニュアル等では、下記のように表記されます。**

UARTの場合

```
Device Drivers --->
  Character devices --->
    [*] Enable TTY <TTY>
      Serial drivers --->
        [*] IMX serial port support              <SERIAL_IMX>
        [*] Console on IMX serial port           <SERIAL_IMX_CONSOLE>
```

カーネル/デバイスツリー

I2C端子の設定

デバイスツリーの設定（at-dtweb）

- Armadillo-640 製品マニュアル
  20.3. Device Treeをカスタマイズする
  https://manual.atmark-techno.com/armadillo-640/armadillo-640_product_manual_ja-1.9.0/ch20.html#sect.customize-dts
  に記載されているat-dtwebをATDE上にインストールすることで、容易に端子設定できます。

- 基本的な機能は、at-dtwebで端子設定できますが、GPIOなどを独自に設定する場合は、ソースコードの設定が必要な場合があります。

- 本セミナーでは、 at-dtwebを使用してデバイスツリーを設定する別の方法を説明します。

デバイスツリーの設定
使用する端子にI2Cの機能を選択1

- Armadillo-640の拡張I/Fの端子は、複数の機能がマルチプレクスされています。
- 端子をどの機能で使用するかを設定するのが、デバイスツリーです。

Armadillo-640 マルチプレクス表

I2C2

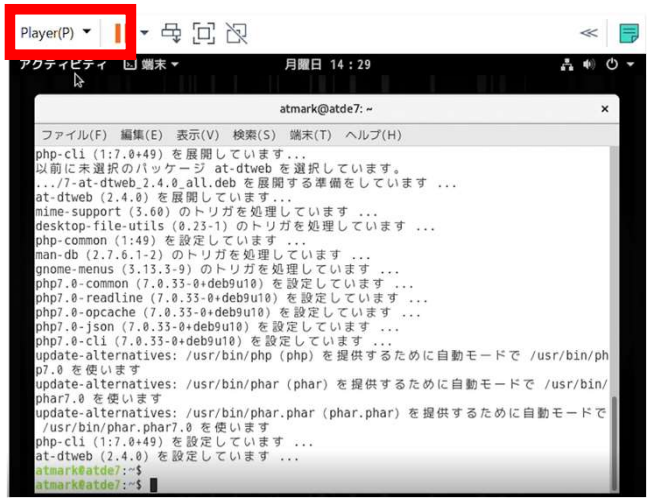
部品番号	ピン番号	信号名	ピン名	電圧グループ	マルチプレクス機能(i.MX6ULLの信号						
					ECSPI					I2C	
					ECSPI1	ECSPI2	ECSPI3	ECSPI4	I2C2	I2C3	I2C4
CON14	1	VCC_3.3V									
	2	GND									
	3	GPIO1_IO20	UART2_TX_DATA	VCC_3.3V			ECSPI3_SS0				I2C4_SCL
	4	GPIO1_IO21	UART2_RX_DATA	VCC_3.3V			ECSPI3_SCLK				I2C4_SDA
CON9	1	GPIO1_IO22	UART2_CTS_B	VCC_3.3V			ECSPI3_MOSI				
	2	GPIO1_IO23	UART2_RTS_B	VCC_3.3V			ECSPI3_MISO				
	3	GPIO1_IO17	UART1_RX_DATA	VCC_3.3V						I2C3_SDA	
	4	GPIO1_IO31	UART5_RX_DATA	VCC_3.3V		ECSPI2_MISO			I2C2_SDA		
	5	GPIO1_IO16	UART1_TX_DATA	VCC_3.3V						I2C3_SCL	
	6	GPIO1_IO30	UART5_TX_DATA	VCC_3.3V		ECSPI2_MOSI			I2C2_SCL		
	7	VCC_3.3V									
	8	VCC_3.3V									
	9	GND									
	10	GND									

at-dtwebの起動

- 下記コマンドを実行して、
  ATDE7 に at-dtweb パッケージをインストールします。

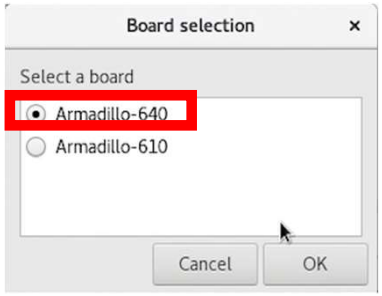
```
atmark@atde7:~$ sudo apt-get update
atmark@atde7:~$ sudo apt-get install at-dtweb
```

- アクティビティをクリックし、「at-dtweb」と入力してat-dtweb のアイコンをクリックして起動します。

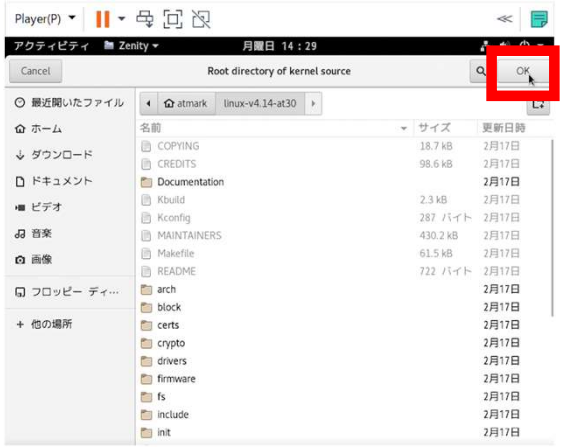


at-dtwebの起動

- ボードの選択画面が表示されるので、Armadillo-640を選択します。



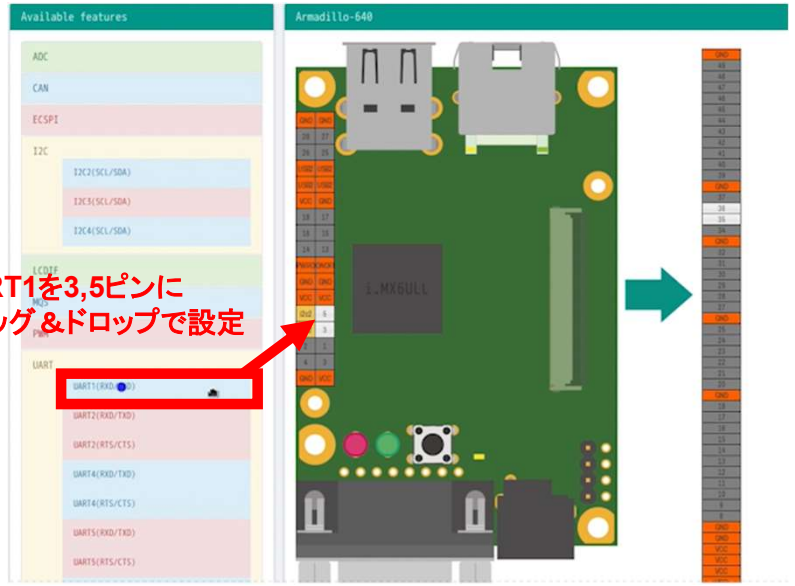
- コンフィギュレーション済みの Linux カーネルディレクトリを選択して、「OK」をクリックします。



デバイスツリーの設定
（at-dtwebの画面）

①

②

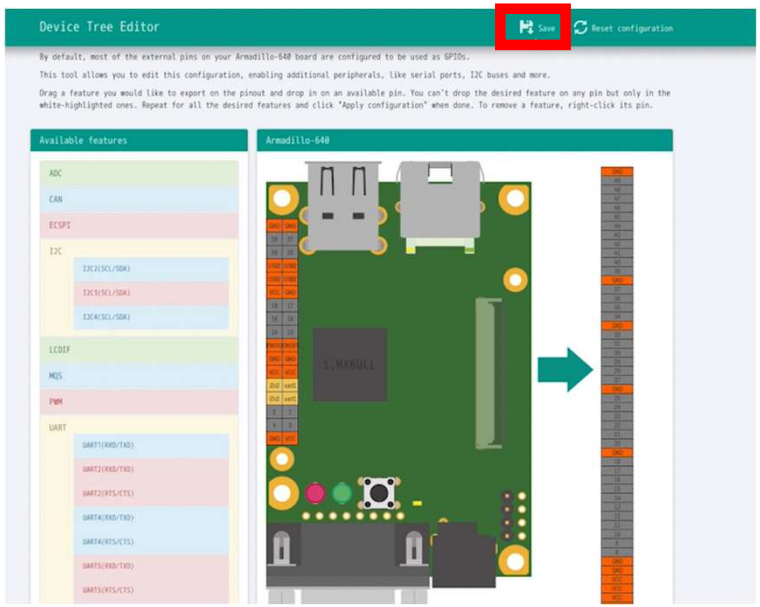


※コンソール出力用

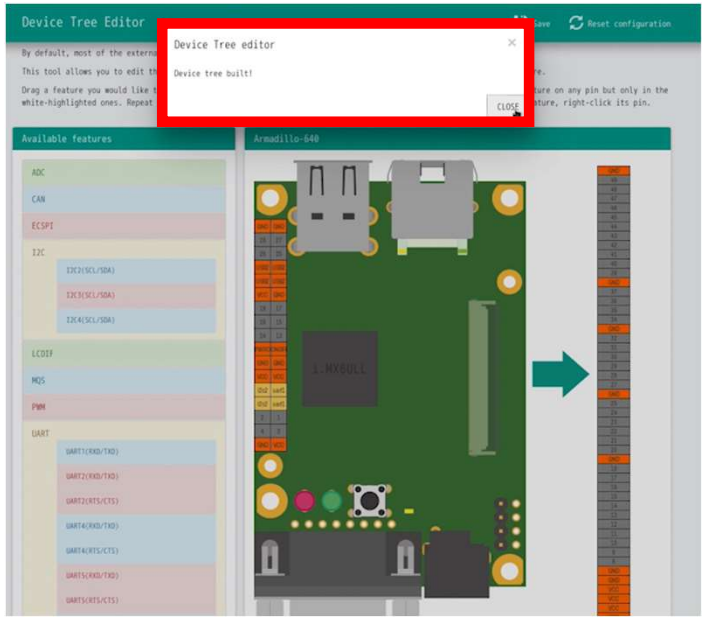
デバイスツリーの設定
（at-dtwebの画面）

③

「Save」をクリックしてビルド



④



「Device Tree build!」が表示されたらビルド完了

デバイスツリーの設定
使用する端子にI2Cの機能を選択2

- ビルドが終了すると、arch/arm/boot/dts/以下に DTS/DTB が作成されています。

- arch/arm/boot/dts/armadillo-640-expansion-interface.dtsi
- arch/arm/boot/dts/armadillo-640-at-dtweb.dtb

カスタマイズしたデバイスツリー

Armadillo-640への書き込み

Armadillo-640に
カーネル/デバイスツリーを書き込む1

- これまでの手順で生成したデバイスツリー

  arch/arm/boot/dts/armadillo-640-at-dtweb.dtb

  をArmadillo-640に書き込むことにって、I2C端子

  CON9_4ピン：I2C2_SDA
  CON9_6ピン：I2C2_SCL

  が有効になります。

Armadillo-640に デバイスツリーを書き込む2

- カスタマイズしたDTBイメージをArmadillo上に用意しておきます。

```
root@armadillo:~# ls
armadillo-640-at-dtweb.dtb
```

- eMMCの第2パーティションを/mnt/ディレクトリにマウントします。

```
root@armadillo:~# mount /dev/mmcblk0p2 /mnt
```

- DTB を a640.dtb にリネームして/mnt/boot/ディレクトリにコピーします。

```
root@armadillo:~# cp armadillo-640-at-dtweb.dtb /mnt/boot/a640.dtb
```

- マウントした eMMC の第 2 パーティションをアンマウントします。

```
root@armadillo:~# umount /mnt
```

I2Cが有効になっているかを確認1

- I2Cが有効になると
  /dev/i2c-1　　（末尾の-1は、I2Cのポートによって変わります。）
  のようにデバイスファイルとして見えるようになります。
- 現時点では、I2Cを有効にしたカーネルイメージ/デバイスツリーで起動していないので、/dev/i2c-1は見えません。

```
root@armadillo:~# ls /dev/i2c*
/dev/i2c-4
```

I2Cが有効になっているかを確認2

- I2Cを有効にしたカーネルイメージ/デバイスツリーで再起動します。
- haltコマンドを実行して、System haltedのログがでるまで待ちます。

```
root@armadillo:~# halt
（略）
[  891.779603] reboot: System halted
```

- Armadillo-640の電源をOFFにします。
- Armadillo-640の電源をONにしてログインします。

=>というプロンプトが出るところでは、 boot と入力してEnterを押します。
ログイン名、パスワード共に、root でログインします。

- I2Cのデバイスファイルが存在することを確認します。

```
root@armadillo:~# ls /dev/i2c*
/dev/i2c-1
```

※I2C2のデバイスファイル名は「i2c-1」になります。

Armadillo-640とSW/LED、照度センサーの接続

Armadillo-640と SW/LED、照度センサーの接続

- カーネルとデバイスツリーでI2C端子を有効化したことにより、I2C接続の照度センサー（TSL2561）と接続可能になりました。

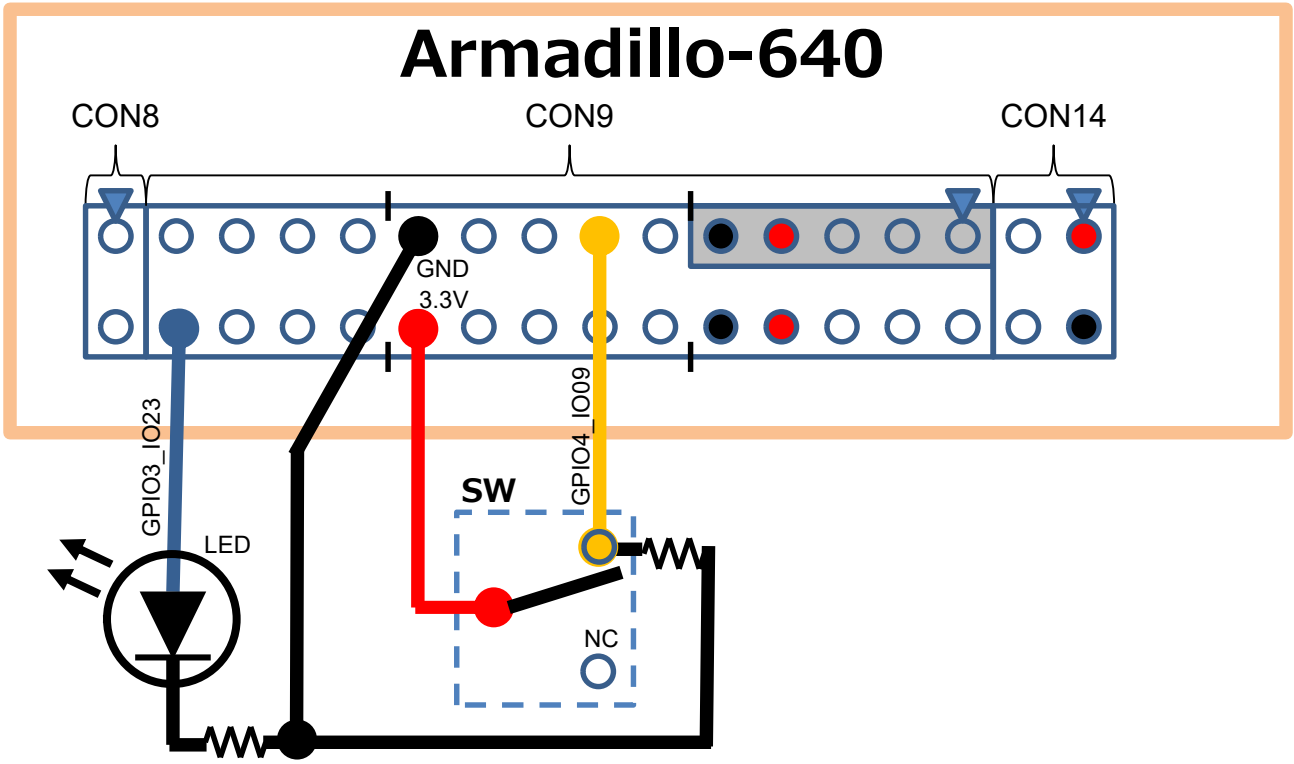
- haltコマンドを実行して、System haltedのログがでるまで待ちます。

```
root@armadillo:~# halt
（略）
[  891.779603] reboot: System halted
```

- Armadillo-640の電源をOFFにします。

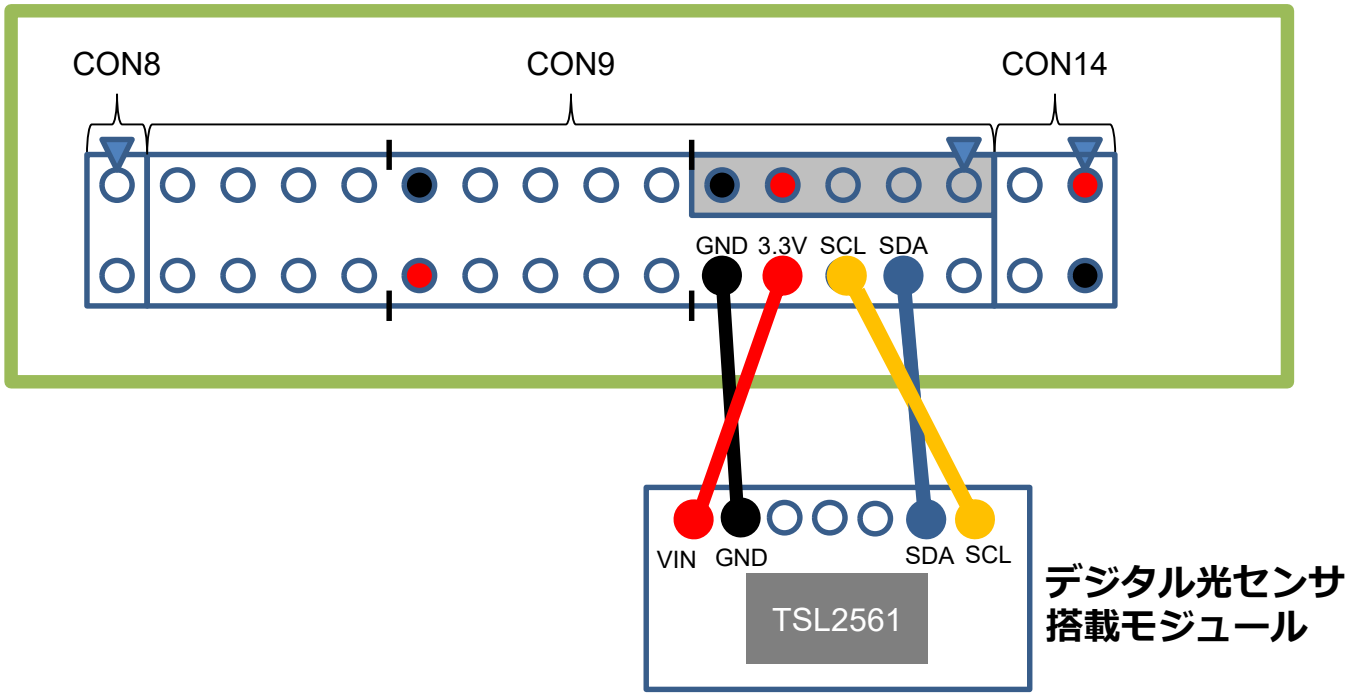
Armadillo-640
トグルスイッチとLEDの接続

- Armadillo-640とトグルSWとLEDを下記のように接続します。



Armadillo-640と照度センサー(TSL2561)の接続

- Armadillo-640と照度センサー（TSL2561）搭載モジュールを下記のように接続します。



Armadillo-640と SW/LED、照度センサーの接続

- **Armadillo-640の電源をONにしてログインします。**

  =>というプロンプトが出るところでは、 boot と入力してEnterを押します。
  ログイン名、パスワード共に、root でログインします。

- **以上で、 Armadillo-640と照度センサー（ TSL2561 ）が I2Cで通信できるようになりました。**