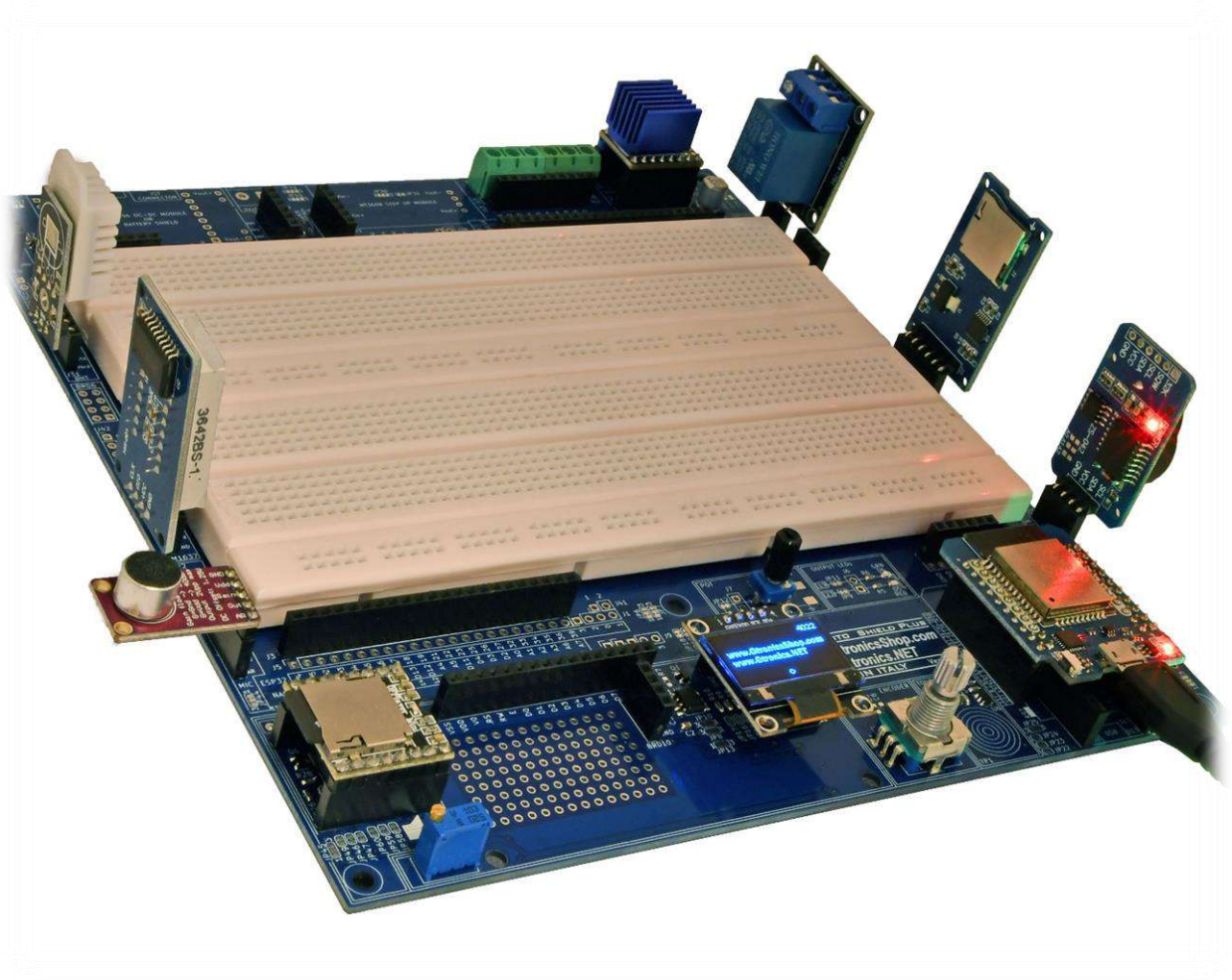
IoT Proto Shield Plus
(Ver. 1.0)
User Manual



INDEX

NOTE:

Some pictures contained in this manual, may be different from the board revision in your hands (at time of writing the current version is rev. 1.0), this does not affect the functionality of the board described in this manual.

1. Introduction and technical specifications

The **IoT Proto Shield Plus** is designed to let you easily prototype with ESP32, ESP8266 and ARDUINO NANO boards.

It works with:

- Arduino NANO form factor boards
  - Arduino NANO 33 IoT
  - Arduino NANO 33 BLE
  - Arduino Nano RP2040 Connect
  - Arduino NANO
  - Arduino NANO EVERY
- ESP32 MINI form factor boards
- ESP8266 MINI form factor boards

Features:

- Wide breadboarding area
  - 2x solderless breadboarding areas 740 contact points each (total 1480 points)
- Breakout headers to replicate each I/O pin close to the breadboarding area
- Breakout headers to replicate 3V3, 5V and GND close to the breadboarding area
- Different working voltages:
  - 3V3
  - 5V
  - external power supply by means of an additional LM2596 DC-DC module of widespread use
  - external power supply by means of an additional Battery Shield of widespread use
- Onboard, ready to use, rotary encoder with switch
- Onboard, ready to use, touch switch
- Two onboard, ready to use, OUTPUT LEDs
- Onboard, ready to use, potentiometer with knob
- Onboard speaker to play MP3 or tones
- Onboard, ready to use, PCF8574 I/O expander to drive 16x2 LCD (I2C communication with LCD with just 2 pins)
- Free soldering pads for additional prototyping parts
- Four onboard, ready to use, level shifter
- RST button, for resetting program
- Plug and play predisposition for the following modules, sensors and actuators of widespread use:
  - Display LCD 16x2
  - Display OLED SSD1306 128x64
  - Display OLED SSD1306 128x32
  - TM1637 seven-segment display
  - RTC (Real Time Clock)
  - SD Card
  - Servo
  - LDR (Light Dependant Resistor)
  - PTC or NTC (Positive or Negative Temperature Coefficient thermistor)
  - RELAY
  - HC-SR04 (ultrasonic distance sensor)
  - DTH11 / DHT22 module or DHT11 / DHT22 bare sensor (humidity sensor)

  - MAX9814 or MAX4466 module (microphone module)
  - DFR0299 MP3 Mini Player (MP3 file music player)
- Board dimensions in mm: 188x196x30[1] (HxWxH)

[1] 188x196x30 without any other additional board plugged into

2. Getting the IoT Proto Shield Plus board out of the box

The IoT Proto Shield Plus consists of:

- 1x IoT Proto Shield Plus board
- 5x adhesive rubber feet
- 1x Potentiometer knob

All you need to do is just to insert the knob in the potentiometer (Figure 1) and stick the adhesive feet (Figure 2) in case that they are not already sticked (it depends on the production process flow).

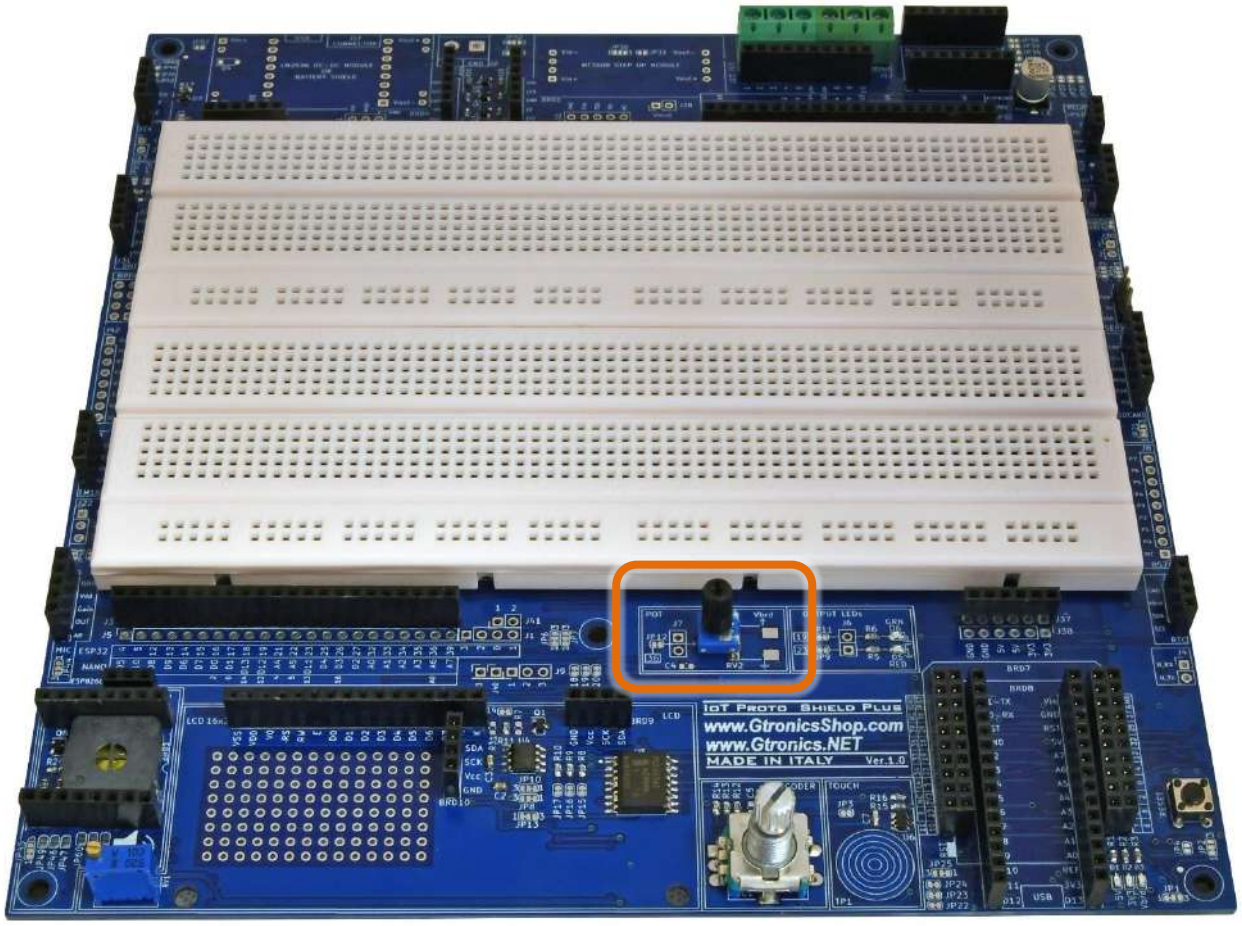


Figure 1 – Placing the potentiometer knob

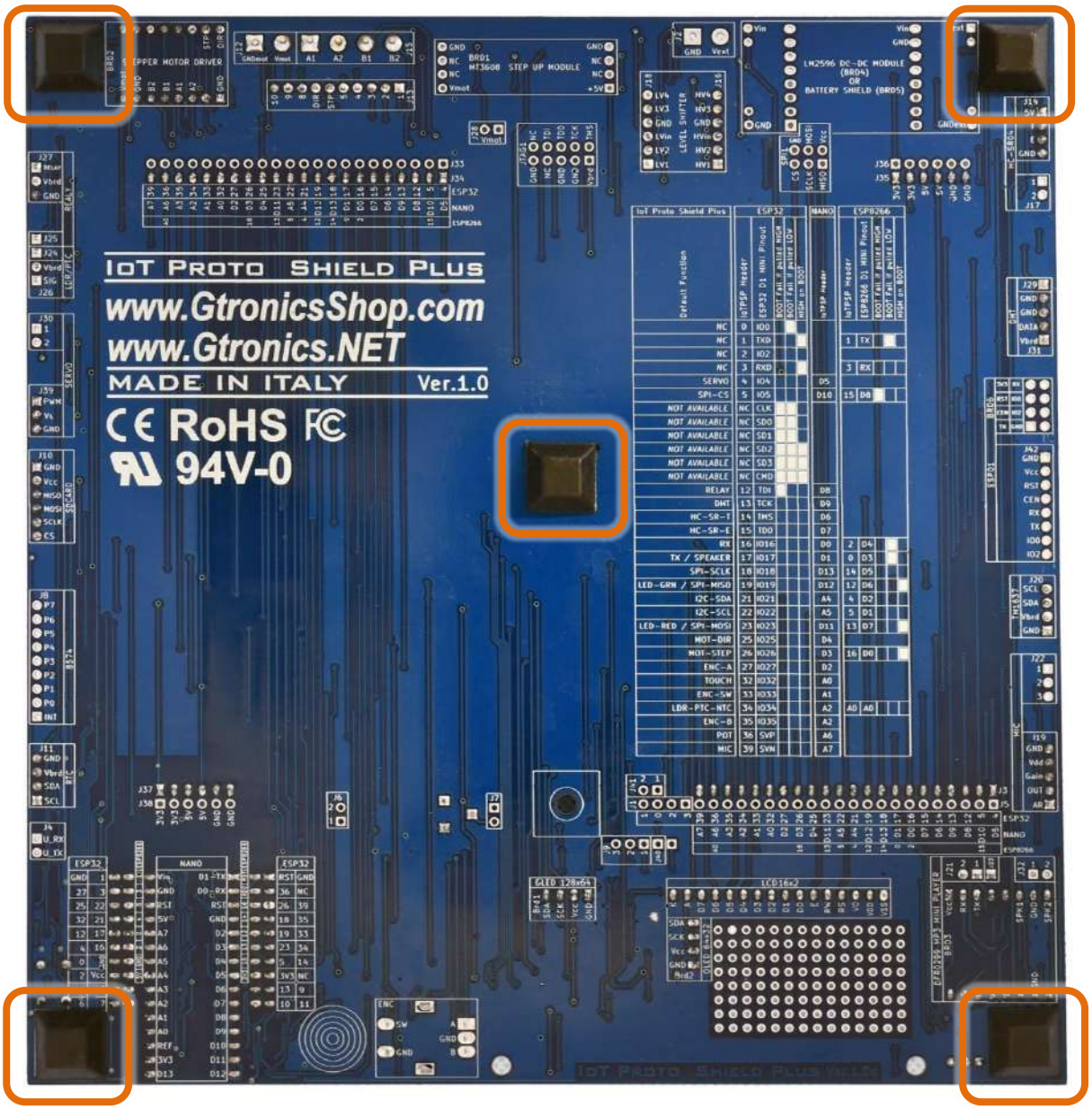


Figure 2 – Sticking the five adhesive rubber feet

3. Layout

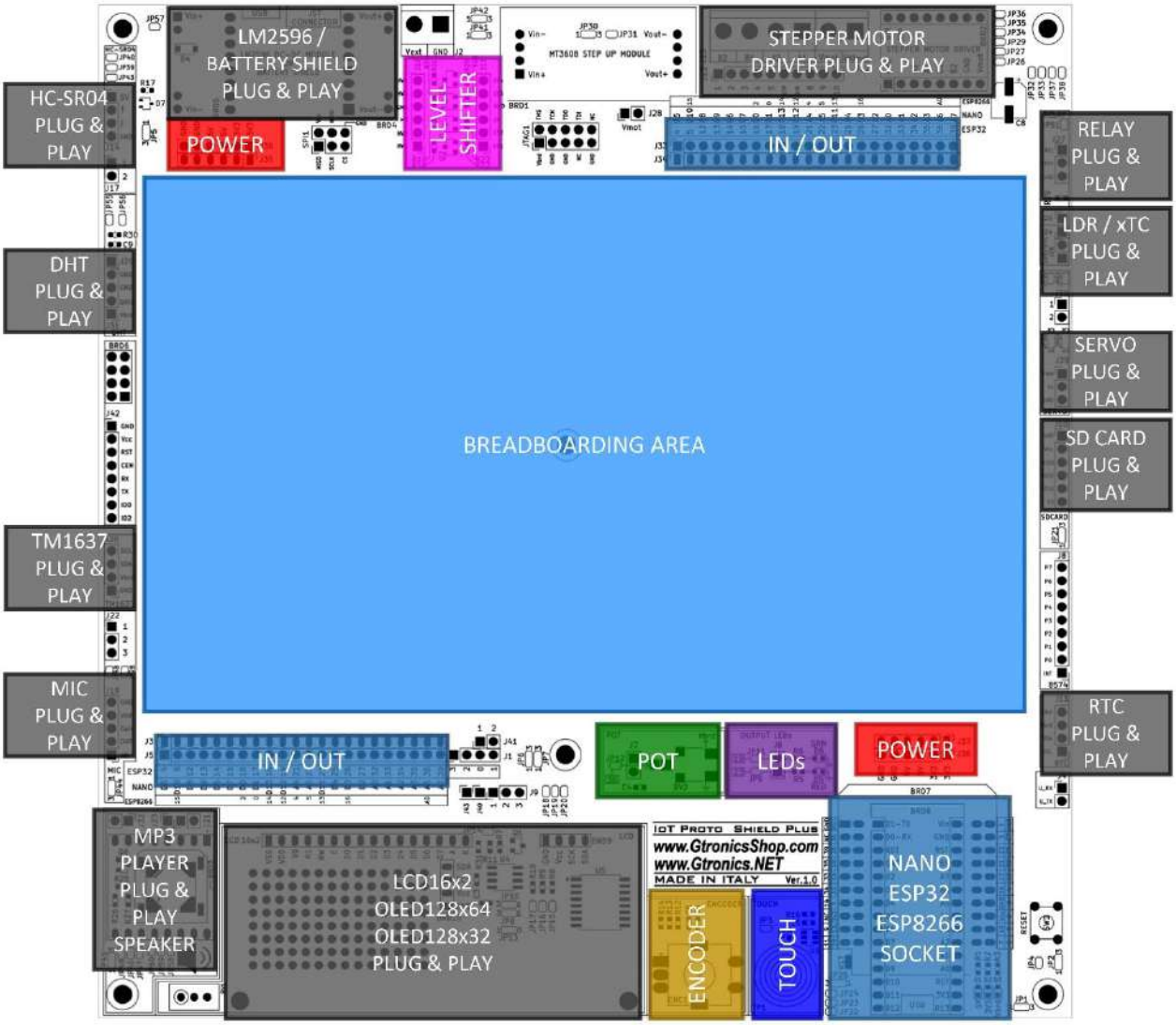


Figure 3 – Layout of the IoT Proto Shield Plus

Please note that the **IoT Proto Shield Plus** works with just one Controller Board at time (i.e. you can't plug the ESP32 and the NANO at the same time).

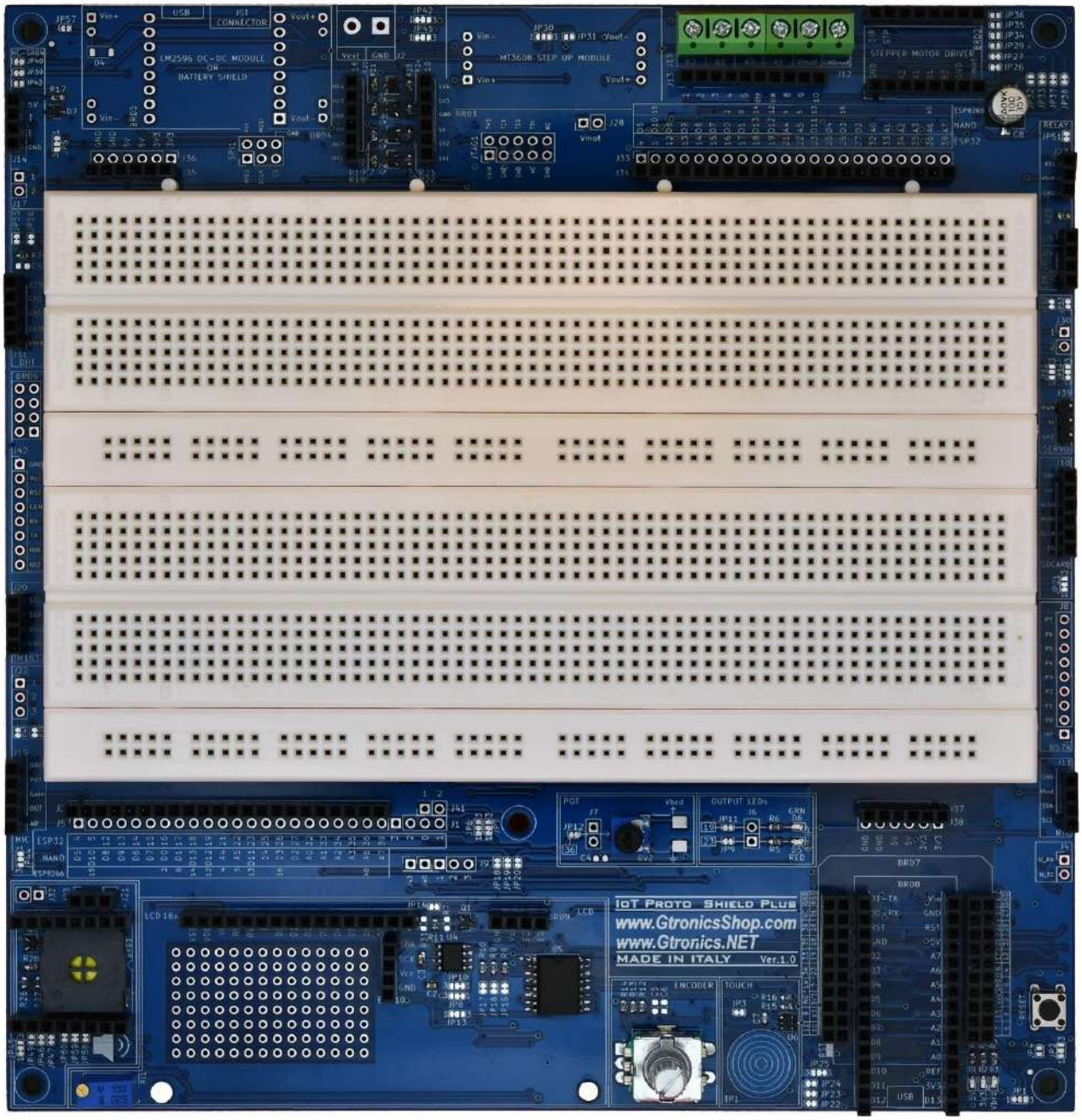


Figure 4 – Top view of the IoT Proto Shield Plus

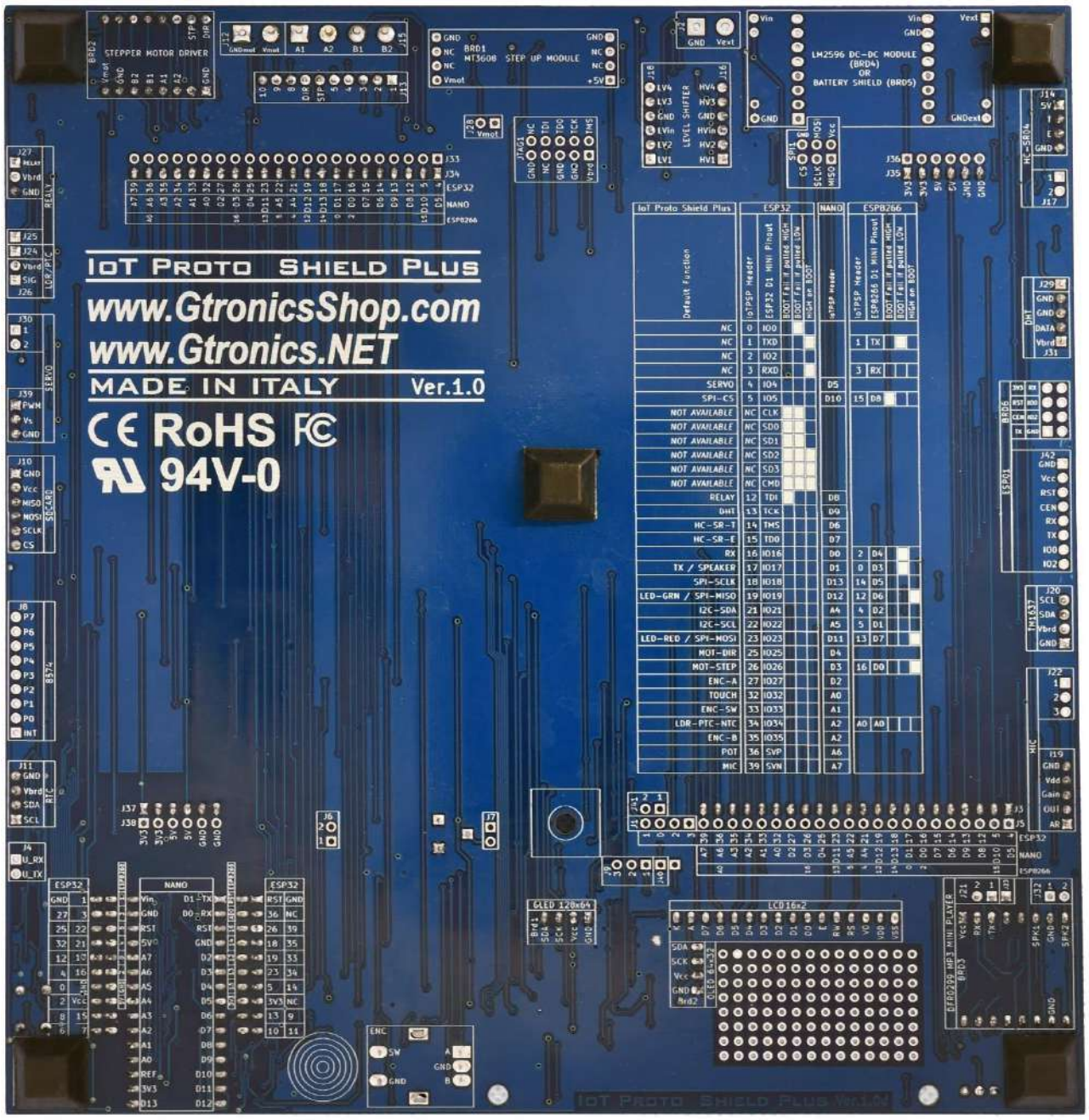


Figure 5 – Bottom view of the IoT Proto Shield Plus

4. Important considerations about 5V and 3V3 before starting to work with the IoT Proto Shield Plus board

ESP32, ESP8266, NANO and NANO33 boards have an operating voltage of 5V or 3,3V (from now on 3V3).
A 5V operating voltage device can handle a 3V3 input signal on its pins, while a 3V3 operating voltage device may be damaged if connected to a 5V input signal.
That's why the IoT Proto Shield Plus routes 3V3 voltage as the default voltage to power all the devices that you can plug in the dedicated headers.
Vbrd sets and routes the operating voltage of the IoT Proto Shield Plus.

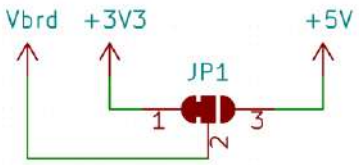


Figure 6 – Schematic of the routing of 3V3 or 5V to Vbrd

4.1. Switching Vbrd to 5V

Be very careful if you switch Vbrd to 5V because you can damage some 3V3 operating voltage devices.

To switch Vbrd to 5V: OPEN (cut in the middle) the connection between JP1 pad 1 and pad 2 and CLOSE (connect with a little drop of tin) JP1 pad 2 to pad 3 (see **Figure 9**).

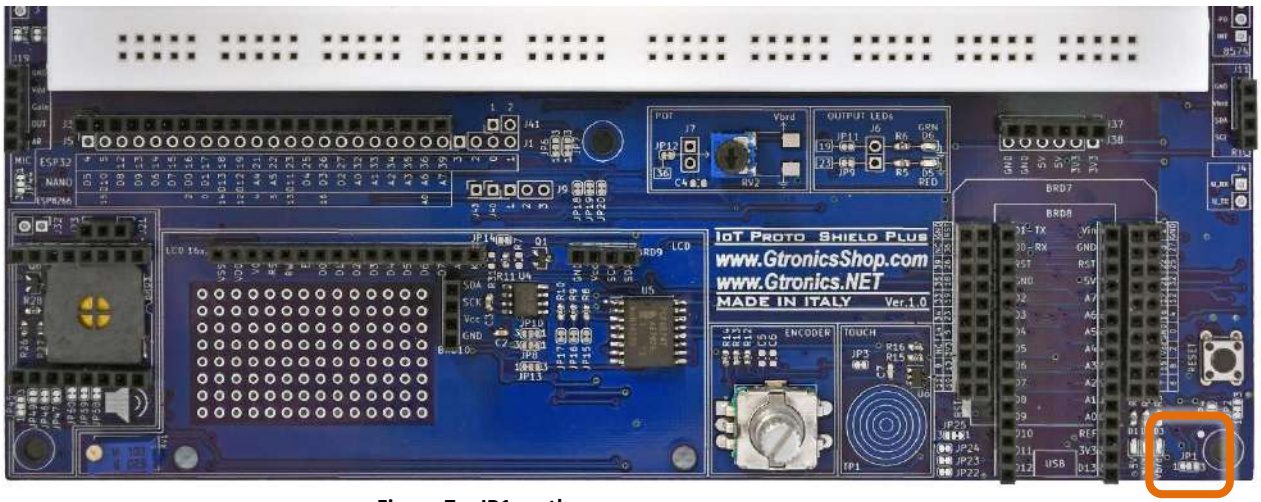


Figure 7 – JP1 on the IoT Proto Shield Plus

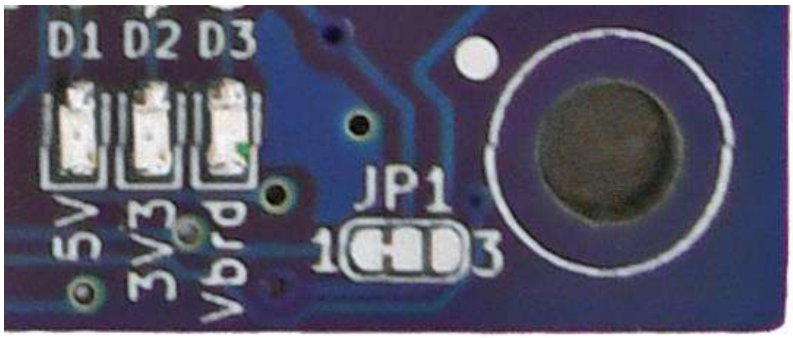


Figure 8 – Setting Vbrd to 3V3 (default)

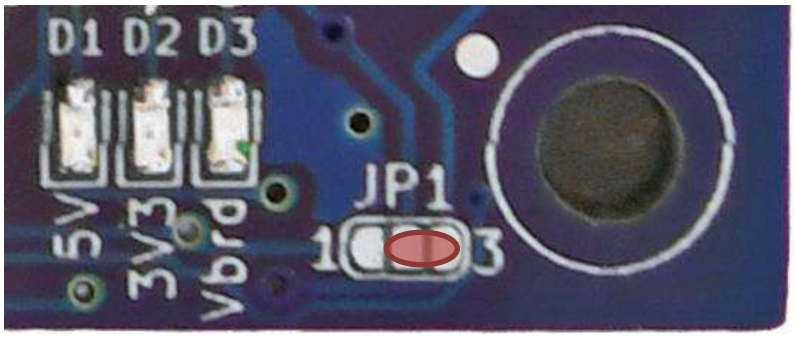


Figure 9 – Setting Vbrd to 5V

5. Using the IoT Proto Shield Plus with Arduino NANO and NANO EVERY boards

Plug the NANO board in the Arduino NANO SOCKET paying attention to pin alignment and orientation.
Connect the NANO board by means of a USB cable to your PC and start using it.
Consider that NANO and NANO EVERY boards operate at 5V, while NANO33 family boards operate at 3V3.

It is strongly suggested to read the technical specification of the board you are using to become familiar with it.

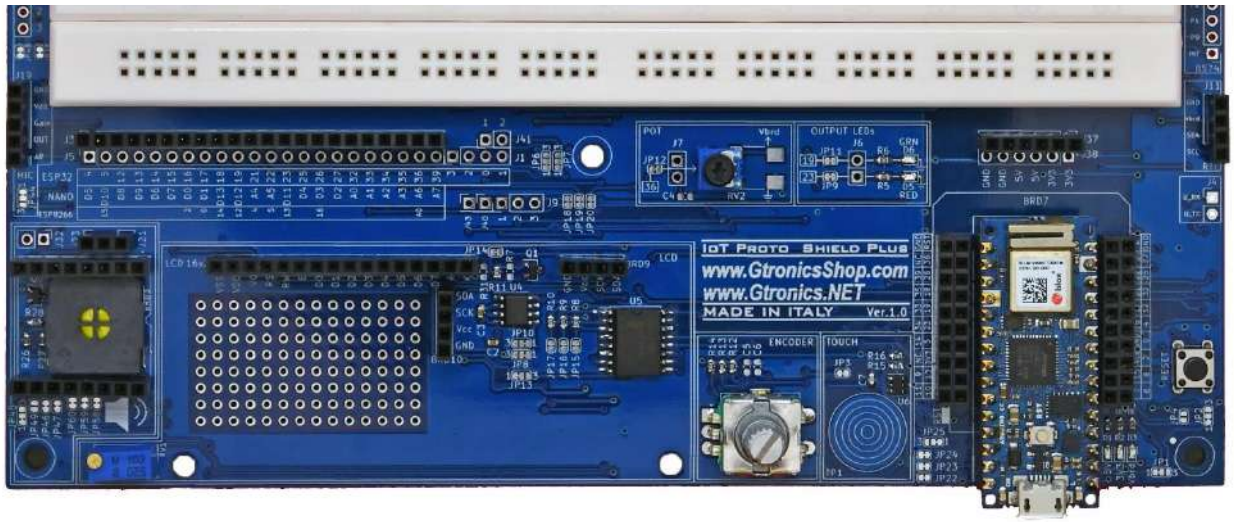


Figure 10 – An ARDUINO NANO (NANO33 IoT) plugged into the IoT Proto Shield Plus

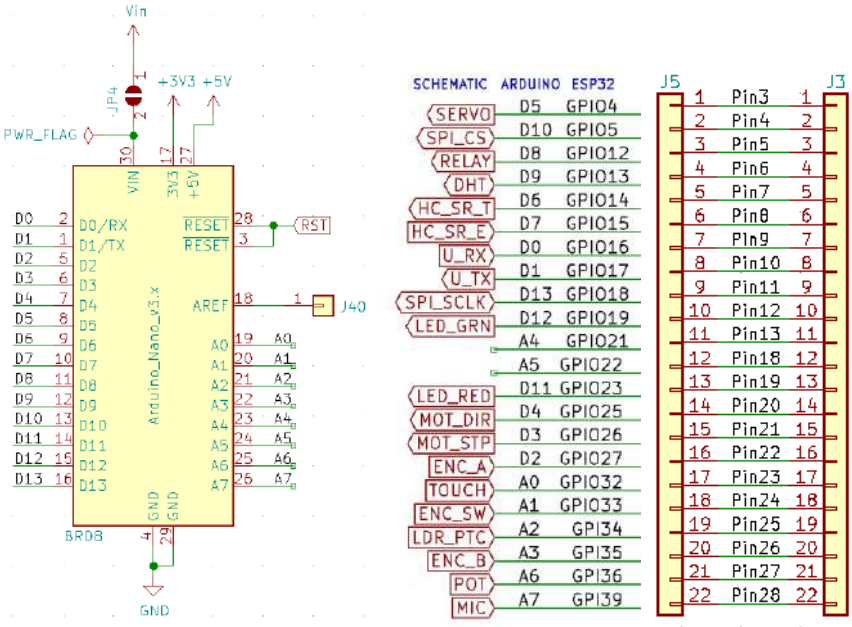


Figure 11 – ARDUINO NANO signals routing on the IoT Proto Shield Plus

6. Using the IoT Proto Shield Plus with ESP32 Mini boards

Plug the ESP32 MINI board in the ESP SOCKET paying attention to pin alignment and orientation.
Connect the ESP32 MINI board by means of a USB cable to your PC and start using it.
Consider that ESP32 boards operate at 3V3.

It is strongly suggested to read the technical specification of the board you are using to become familiar with it.

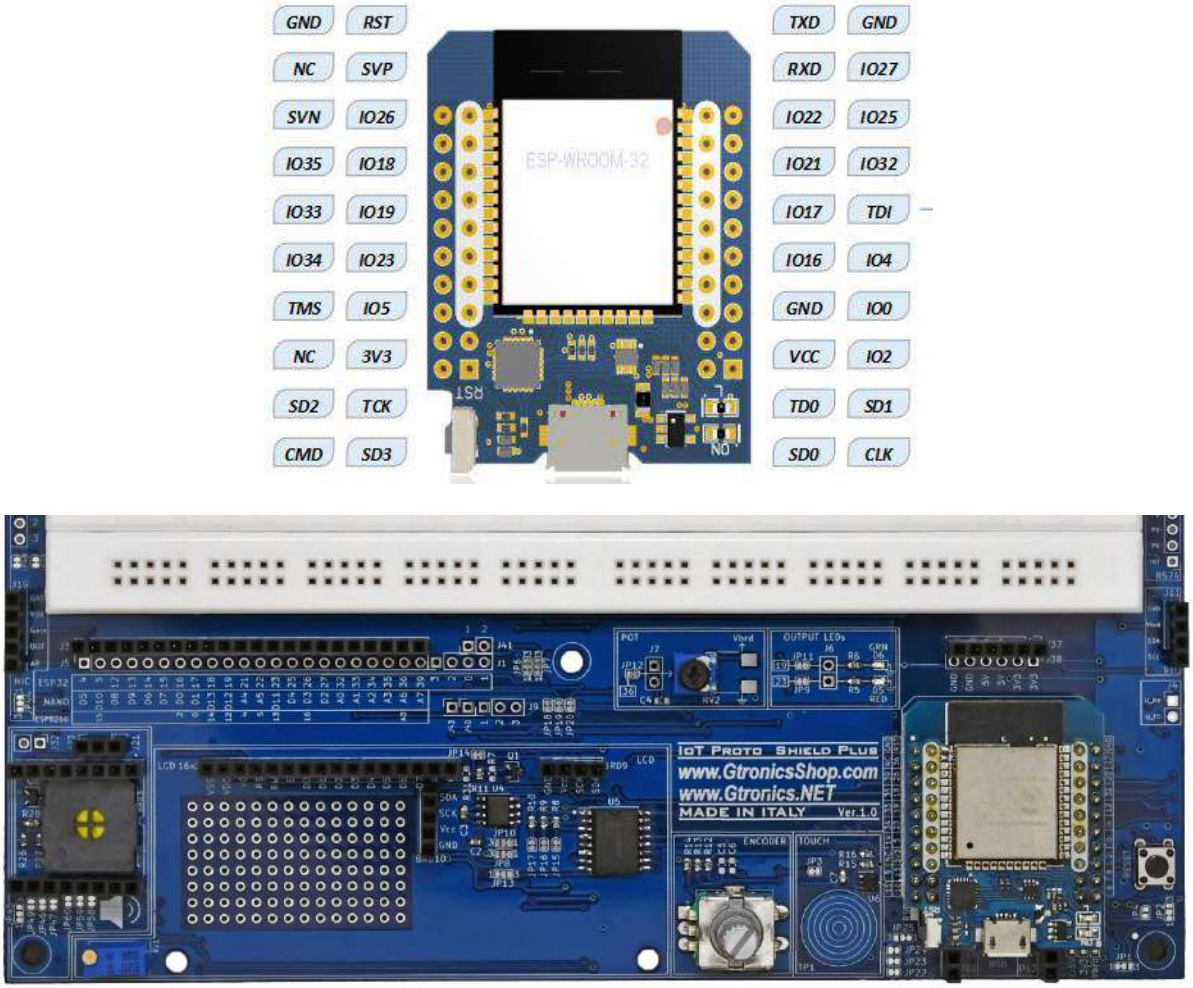


Figure 12 – The ESP32 mini D1 plugged into the IoT Proto Shield Plus

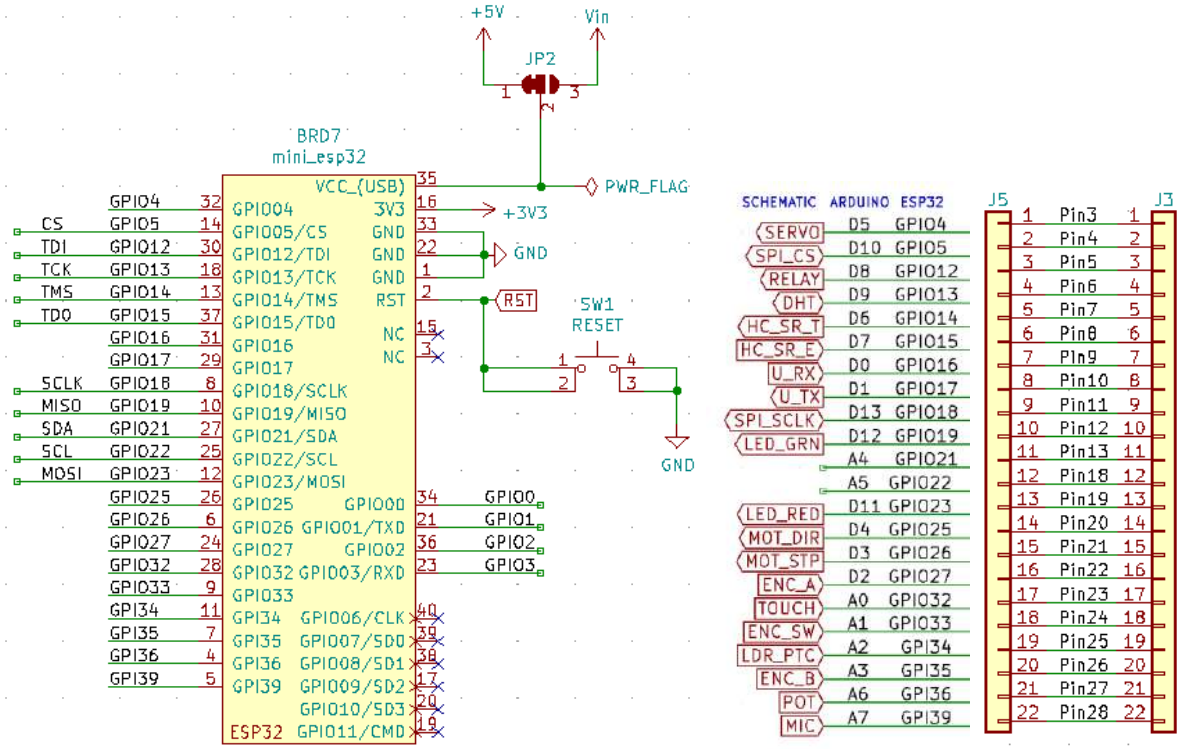
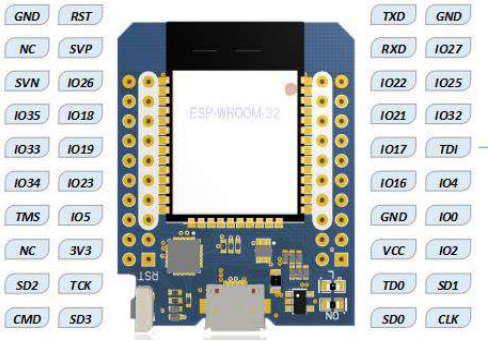


Figure 13 – ESP32 signals routing on the IoT Proto Shield Plus

7. Using the IoT Proto Shield Plus with ESP8266 Mini boards

Consider that the footprint of the ESP8266 mini board is a "shrinked" version of the ESP32 footprint (or the ESP32 is an "extended" version of the ESP8266 footprint....) and they are compatible since they share the pinout on some pins.
Plug the ESP8266 MINI board in the ESP SOCKET paying attention to pin alignment and orientation.
Connect the ESP8266 MINI board by means of a USB cable to your PC and start using it.
Consider that ESP8266 boards operate at 3V3.

It is strongly suggested to read the technical specification of the board you are using to become familiar with it.



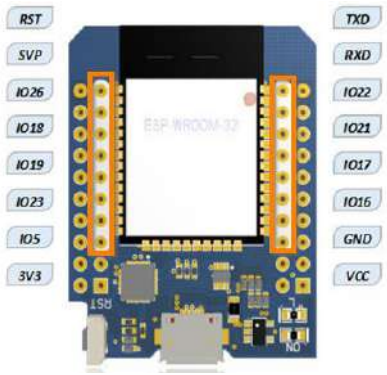


Figure 14 – **The ESP8266 D1 mini plugged into the IoT Proto Shield Plus**

8. Default connections on the IoT Proto Shield Plus

BOARDS PINOUT			IoT Proto Shield Plus		
ESP 8266	ESP32 Mini	Arduino NANO	Primary Function	Secondary Function	Reference JP
	GPIO00 / IO0				
GPIO1 / TX	GPIO01 / TXD				
	GPIO02 / IO2				
GPIO3 / RX	GPIO03 / RXD				
m	GPIO04 / IO4	D5	Servo		JP53
GPIO15 / D8	GPIO05 / IO5	D10	SPI_CS		
	GPIO12 / TDI	D8	RELAY	JTAG_TDI	JP51
	GPIO13 / TCK	D9	DHT	JTAG_TCK	JP56
	GPIO14 / TMS	D6	HC_SR_T	JTAG_TMS	JP39
	GPIO15 / TD0	D7	HC_SR_E	JTAG_TD0	JP40
GPIO2 / D4	GPIO16 / IO16	D0	RX		JP46
GPIO0 / D3	GPIO17 / IO17	D1	TX	SPEAKER	JP47
GPIO14 / D5	GPIO18 / IO18	D13	SPI_SCLK		
GPIO12 / D6	GPIO19 / IO19	D12	LED-GRN	SPI_MISO	JP11
GPIO4 / D2	GPIO21 / IO21	A4	I2C_SDA		JP7
GPIO5 / D1	GPIO22 / IO22	A5	I2C_SCL		JP6
GPIO13 / D7	GPIO23 / IO23	D11	LED_RED	SPI_MOSI	JP9
	GPIO25 / IO25	D4	MOT_DIR		JP37
GPIO16 / D0	GPIO26 / IO26	D3	MOT_STP		JP38
	GPIO27 / IO27	D2	ENC-A		JP20
	GPIO32 / IO32	A0	TOUCH		JP3
	GPIO33 / IO33	A1	ENC-SW		JP18
	GPIO34 / IO34	A2	LDR / xTC		JP50
	GPIO35 / IO35	A3	ENC-B		JP19
ADC0	GPIO36 / SVP	A6	POT		JP12
	GPIO39 / SVN	A7	MIC		JP28

HIGH ON BOOT

BOOT FAILURE IF PULLED LOW

BOOT FAILURE IF PULLED HIGH

CONNECTED TO ONBOARD DEVICES BY DEFAULT

FREE TO BE USED BY DEFAULT

HIGH ON BOOT

You have to consider that during the device booting process, the pin will be in high state.
i.e.
If you are using an ESP8266 module and you want to use its pin GPIO16/D0, you have to consider that the pin GPIO16/D0 will be in high state during the booting process of the ESP8266 module.

BOOT FAILURE IF PULLED LOW

You have to consider that, if you connect the pin to a PULL DOWN resistor, the device will not boot (it will not work).
i.e.
If you are using an ESP8266 module and you connect pin GPIO0/D3 to a pull down resistor, the ESP8266 module will not boot, thus it will not work.

BOOT FAILURE IF PULLED HIGH

You have to consider that, if you connect the pin to a PULL UP resistor the device will not boot (it will not work).
i.e.
If you are using an ESP32 module and you connect pin GPIO12/TDI to a pull up resistor, the ESP32 module will not boot, thus it will not work.

CONNECTED TO ONBOARD DEVICES BY DEFAULT

You have to consider that the pin is connected by default to one of the onboard devices of the **IoT Proto Shield Plus**.
i.e.
If you are using an Arduino NANO and you want to use A6 to acquire an analogue signal, you have to consider that it connects by default to the onboard POT (RV2).
To use A6 you need to disconnect it from POT cutting JP12 on the middle (see further in this manual detailed description for each onboard device).

FREE TO BE USED BY DEFAULT

If its primary function device or module is not inserted in the **IoT Proto Shield Plus**, you can use the pin for your own purposes without any trouble.
i.e.
If you are using an Arduino NANO and you want to use A7 to acquire an analogue signal, and the MIC module is not plugged into the MIC header, you can connect your signal to A7 without any trouble.

CONCLUSION

- If you need to use one device or module, and the **IoT Proto Shield Plus** has the plug and play predisposition for that module, just plug it into the header (see further in this manual the detailed description for each plug and play predisposition) and it will be ready to work.
- If you need to use a device or module, and the **IoT Proto Shield Plus** does not have the plug and play predisposition for that module, plug it into the breadboarding area and connect it to one of the “FREE TO BE USED BY DEFAULT” pin. Remember to consider its possible behavior during boot.
- If you absolutely need to use one of the “CONNECTED TO ONBOARD DEVICES BY DEFAULT” pin, you must disconnect it from the onboard device before using it for other purposes. Remember to consider its possible behavior during boot.

9. Using the IoT Proto Shield Plus with other form factor boards

If you need to use the **IoT Proto Shield Plus** with a board with a form factor different than NANO (see §5) or the ESP32 D1 MINI (see §6) or the ESP8266 D1 MINI (see §7), you need to use an adapter.

9.1. UNO to IoT Proto Shield Plus Adapter

If you need to use the **IoT Proto Shield Plus** with an Arduino UNO form factor board (i.e. Arduino UNO, Leonardo, Zero, Yun, etc.), you can use the **UNO to IoT Proto Shield Plus Adapter** (https://www.gtronicsshop.com/en/proto-shield-plus/37-uno-to-iot-proto-shield-plus-adapter.html).

The adapter works with any board compatible with Arduino UNO in terms of form factor, pin-out and electrical specifications.

Anyway, before using the **Uno To IoT Proto Shield Plus Adapter**, please take a look at the user manual.

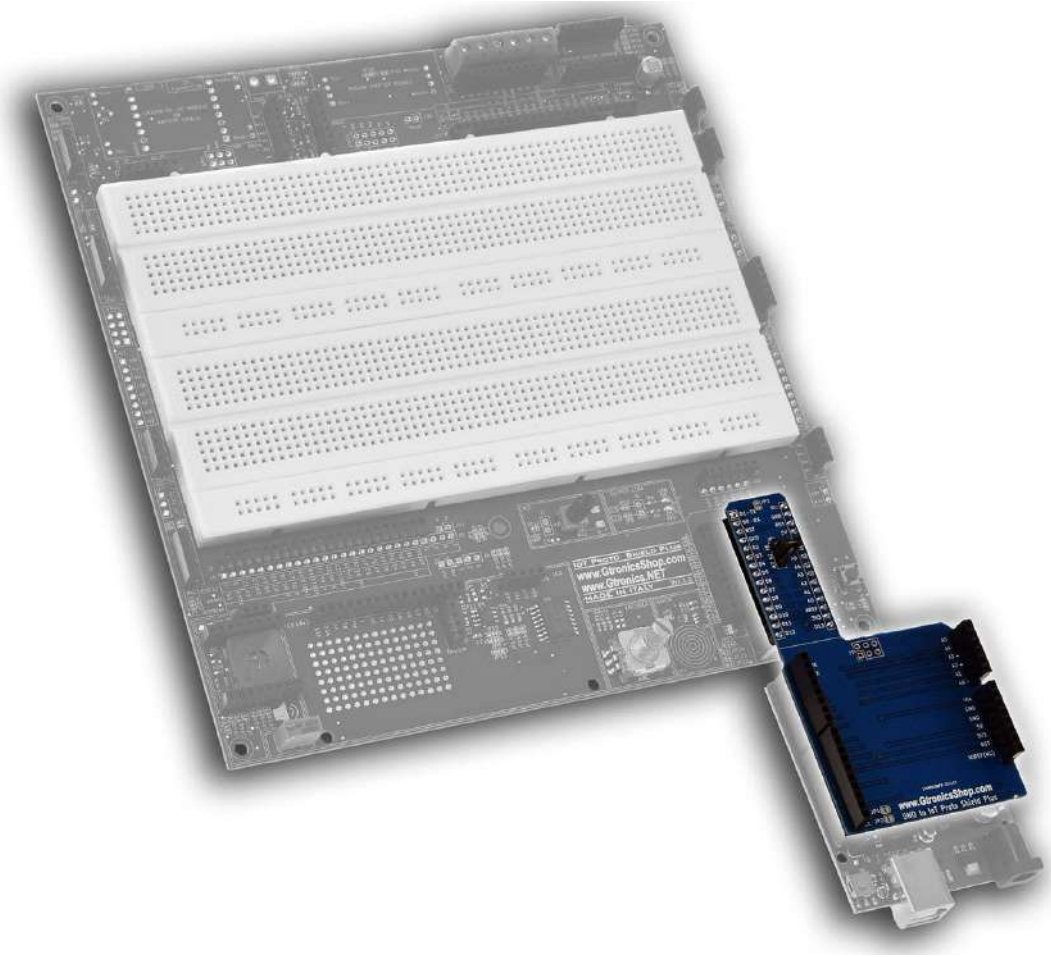


Figure 15 – The UNO to IoT Proto Shield Plus Adapter

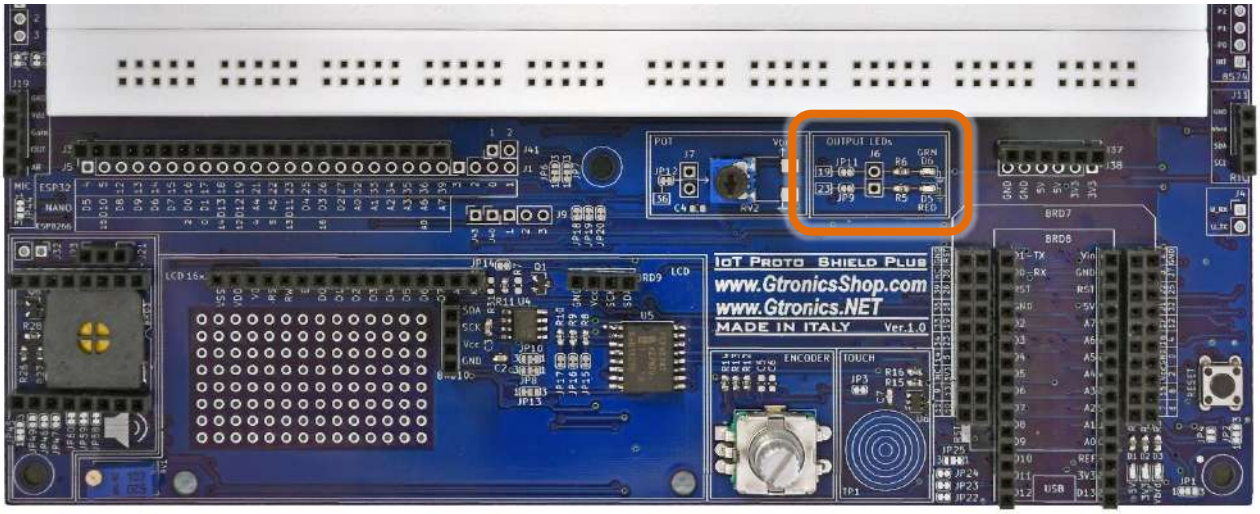
9.2. ESP32 or ESP8266 boards with different form factor

If you need to use the **IoT Proto Shield Plus** with an ESP32 or ESP8266 board with a different form factor (i.e. the ones shown in Figure 16) you can request us if an adapter is available or design your own adapter taking as an example the UNO to IoT Proto Shield Plus adapter shown in the previous paragraph.

Figure 16 – Examples of ESP32 and ESP8266 boards

10. LEDs

Working with the two onboard LEDs of the **IoT Proto Shield Plus**.



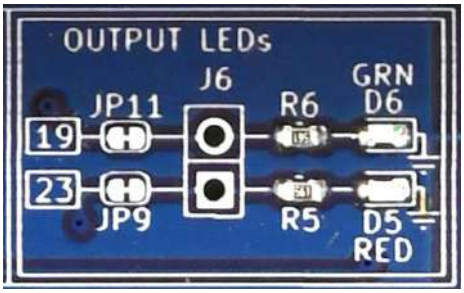


Figure 17 – LEDs Section of the IoT Proto Shield Plus

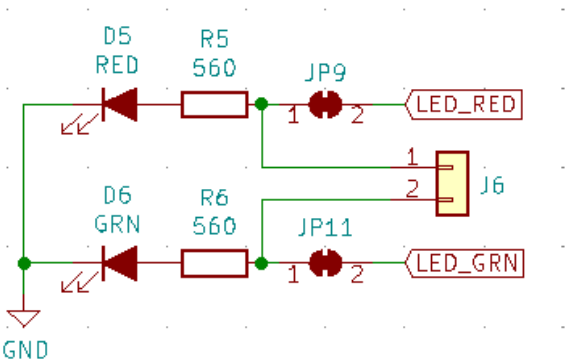


Figure 18 – Schematic of the IoT Proto Shield Plus LEDs Section

10.1. Default connections

	ESP32	ARDUINO	ESP8266
GRN LED	GPIO19	D12	GPIO12 (D6)
RED LED	GIOP23	D11	GPIO13 (D7)

10.2. Sample Sketches

Location: IoTPSP_Sample_Sketches\IoTPSP_LEDs

According to the board you are using and the signals routing, the sketches make the green (GRN) and the red (RED) LEDs blink.

Use the IoTPSP_LEDs_NANO sketch to test the LEDs with ARDUINO NANO boards.
Use the IoTPSP_LEDs_ESP32 sketch to test the LEDs with ESP32 boards.
Use the IoTPSP_LEDs_ESP8266 sketch to test the LEDs with ESP8266 boards.

10.3. Using the LEDs with different GPIO pins

According to Figure 18, GRN and RED LEDs are routed to the default GPIO pins by means of JP11 and JP9.
To disconnect GRN LED from its default GPIO, you must open JP11 (cut it in the middle).
To disconnect RED LED from its default GPIO, you must open JP9 (cut it in the middle).
Solder a header on J6 and, by means of some jumper wires, connect GRN and RED LEDs to other GPIO pins according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** (see §8) to avoid conflict with other default connections.

11. POT

Working with the onboard Potentiometer of the **IoT Proto Shield Plus**.

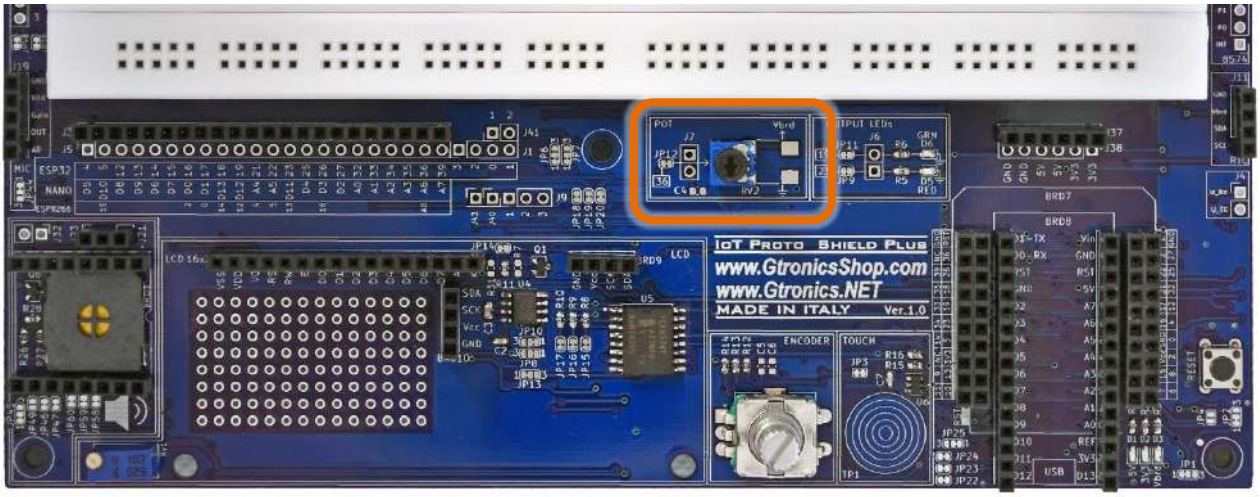


Figure 19 – POT section of the IoT Proto Shield Plus

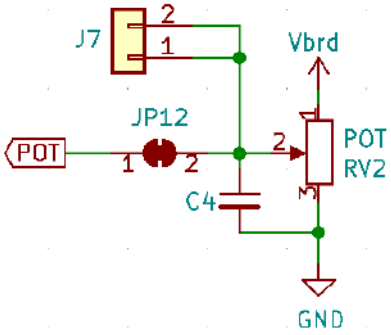


Figure 20 – Schematic of the IoT Proto Shield Plus POT section

11.1. Default connections

	ESP32	ARDUINO	ESP8266
POT	GPIO36	A6	A0

11.2. Sample Sketches

Location: IoTPSP_Sample_Sketches\IoTPSP_POT

According to the board you are using and the signals routing, the sketches read the analog value of the POT and print it to the serial monitor.

Use the IoTPSP_POT_NANO sketch to test the POT with ARDUINO NANO boards.
Use the IoTPSP_POT_ESP32 sketch to test the POT with ESP32 boards.
Use the IoTPSP_POT_ESP8266 sketch to test the POT with ESP8266 boards.

11.3. Using the POT with a different GPIO pin

According to Figure 20, POT is routed to the default GPIO pin by means of JP12.
To disconnect POT from its default GPIO pin, you must open JP12 (cut it in the middle).
Solder a header on J7 and, by means of some jumper wires, connect it to another GPIO pin according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** (see §8) to avoid conflict with other default connections.

12. ENCODER

Working with the onboard ENCODER of the **IoT Proto Shield Plus**.

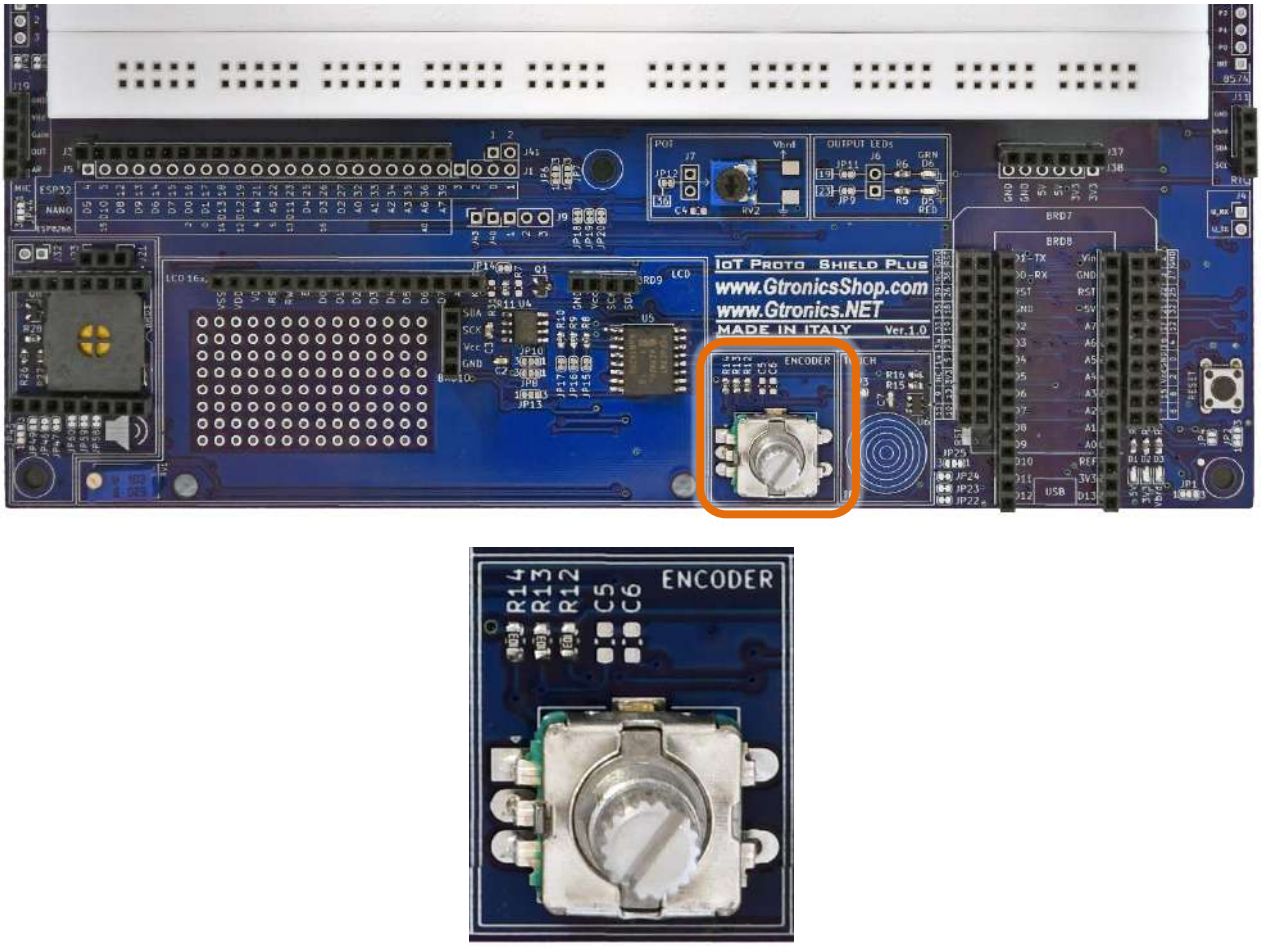


Figure 21 – ENCODER section of the IoT Proto Shield Plus

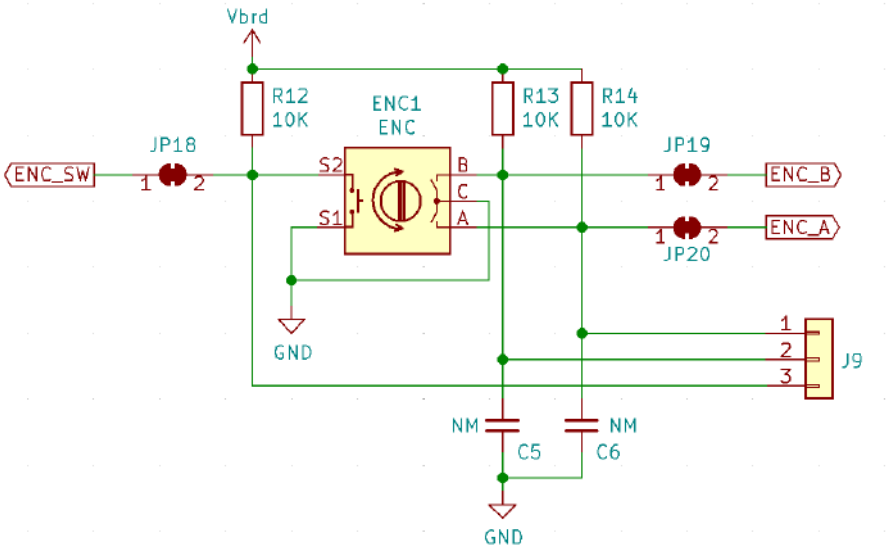


Figure 22 – Schematic of the IoT Proto Shield Plus ENCODER section

12.1. Default connections

	ESP32	ARDUINO	ESP8266
ENC A	GPIO27	D2	*NC*
ENC B	GPIO35	A3	*NC*
ENC SW	GPIO33	A1	*NC*

NC = Not Connected

Please note that there are no default connections for the ESP8266. If you need to use the ENCODER with ESP8266 boards, you need to route the signals with jumper wires.

12.2. Suggested ESP8266 connections

The sample sketches are made to work with these connections:
ESP8622 GPIO 14 (D5) connected to header J3 ESP32 GPIO33 (ENC_SW)
ESP8622 GPIO 5 (D1) connected to header J3 ESP32 GPIO27 (ENC_A)
ESP8622 GPIO 4 (D2) connected to header J3 ESP32 GPIO35 (ENC_B)

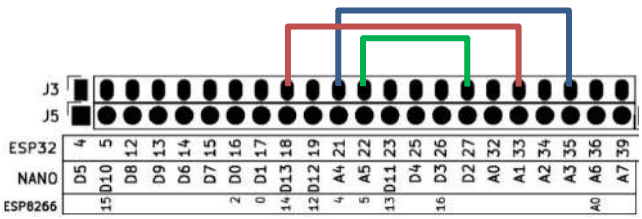


Figure 23 – Suggested connections to work with ENCODER and ESP8266

12.3. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_ENCODER`

Use the `IoTPSP_ENCODER_Simple_NANO` sketch to test the ENCODER with ARDUINO NANO boards.
Use the `IoTPSP_ENCODER_Simple_ESP32` sketch to test the ENCODER with ESP32 boards.
Use the `IoTPSP_ENCODER_Simple_ESP8266` sketch to test the ENCODER with ESP8266 boards.

Open the serial monitor and rotate the encoder knob to see its incremental position changing or press the encoder knob to see the GRN led changing its status.

A more robust way to manage the A and B signals of the ENCODER is using the interrupts.

Use the `IoTPSP_ENCODER_Interrupt_NANO` sketch, to test the ENCODER with ARDUINO NANO boards using interrupts.
Use the `IoTPSP_ENCODER_Interrupt_ESP32` sketch, to test the ENCODER with ESP32 boards using interrupts.
Use the `IoTPSP_ENCODER_Interrupt_ESP8266` sketch, to test the ENCODER with ESP8266 boards using interrupts.

12.4. Using the ENCODER with different GPIO pins

According to Figure 22, ENCODER signals are routed to the default GPIO pins by means of JP18, JP19 and JP20.
To disconnect ENCODER signals from their default GPIO pins, you must open JP 18, JP19 and JP20 (cut them in the middle).
Solder a header on J9 and, by means of some jumper wires, connect other GPIO pins according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** (see §8) to avoid conflict with other default connections.

13. TOUCH

The touch sensor on the **IoT Proto Shield Plus** uses the TTP223E chip, which is the same used on the touch boards of widespread use.

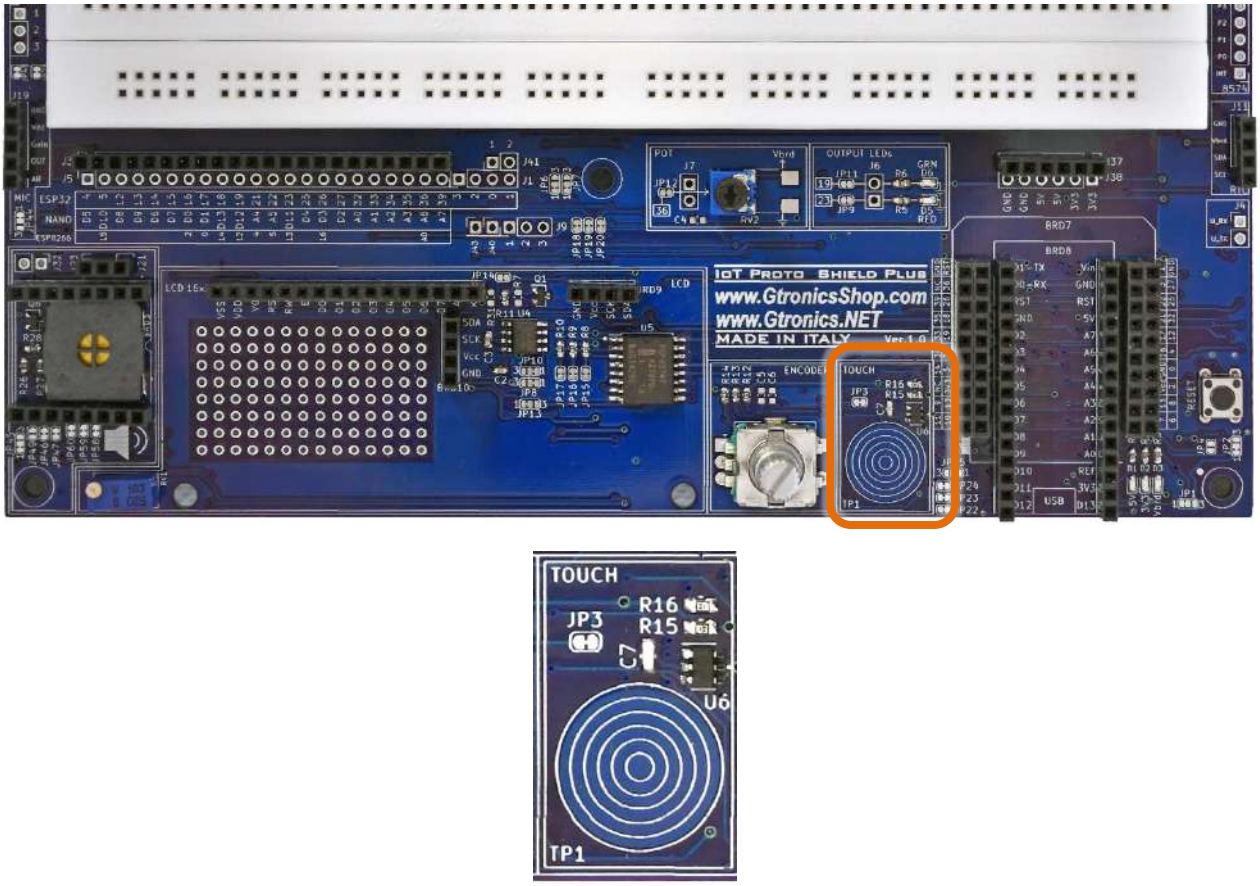


Figure 24 – TOUCH section of the IoT Proto Shield Plus

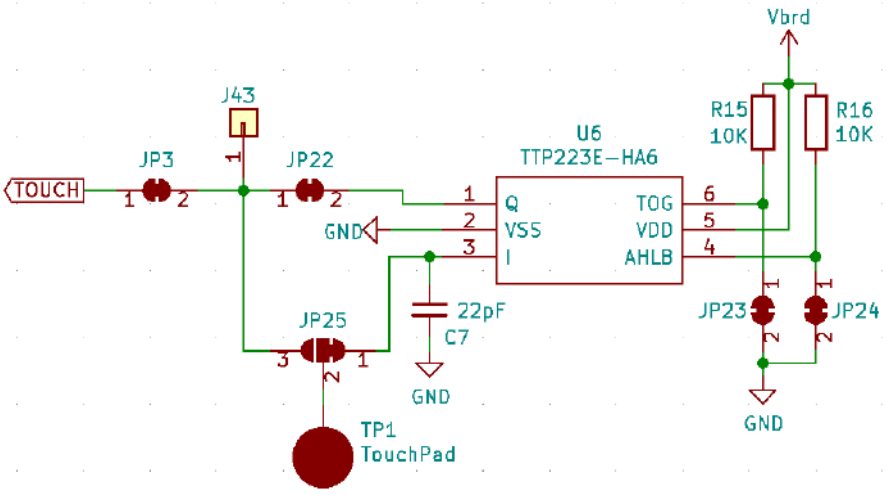


Figure 25 – Schematic of the IoT Proto Shield Plus TOUCH section

13.1. Default connections

	ESP32	ARDUINO	ESP8266
TOUCH	GPIO32	A0	*NC*

NC = Not Connected

Please note that there are no default connections for the ESP8266. If you need to use the TOUCH with ESP8266 boards, you need to route the signals with jumper wires.

13.2. Suggested ESP8266 connections

The sample sketch is made to work with this connection:
ESP8622 GPIO 14 (D5) connected to header J3 ESP32 GPIO32 (TOUCH)

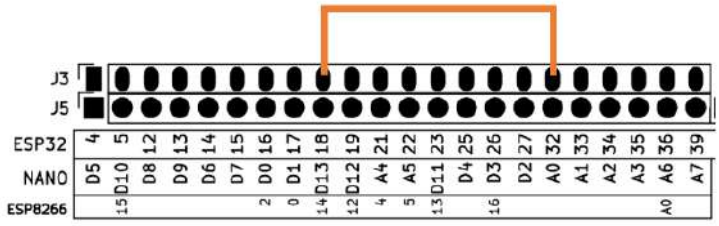


Figure 26 – Suggested connections to work with TOUCH and ESP8266

13.3. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_TOUCH`

Use the `IoTPSP_TOUCH_NANO` sketch to test the TOUCH section with ARDUINO NANO boards.
Use the `IoTPSP_TOUCH_ESP32_EXTERNAL` sketch to test the TOUCH section with ESP32 boards.
Use the `IoTPSP_TOUCH_ESP8266` sketch to test the TOUCH section with ESP8266 boards.

Put your finger on the touch circles and the GRN led will turn on according to your touch.

ESP32 also have internal touch sensors, so with ESP32 boards, you can bypass the TTP223E chip and work directly with the internal sensor connected to GPIO32 (TOUCH9).

To setup the board to work with ESP32 internal touch sensor, you need to OPEN JP22 (cut it in the middle), CLOSE JP25 pad 3 and pad 2 (connect them with a little drop of tin) and OPEN pad 1 and 2 (cut between the pads).
Use the `IoTPSP_TOUCH_ESP32_INTERNAL` sketch to test the TOUCH section with ESP32 boards.
Use serial monitor to check the internal touch sensor value.

13.4. Using TOUCH with a different GPIO pin

According to Figure 25, TOUCH is routed to the default GPIO pin by means of JP3.

To disconnect TOUCH from its default GPIO pin, you must OPEN JP3 (cut it in the middle).
Solder a header on J43 and, by means of some jumper wires, connect it to another GPIO pin according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** (see §8) to avoid conflict with other default connections.

14. I²C on the IoT Proto Shield Plus

I²C (Inter-Integrated Circuit, eye-squared-C) is widely used for attaching peripheral ICs to processors and microcontrollers (check out more about I²C Wikipedia or other online documentation).

I²C on the **IoT Proto Shield Plus** is used to communicate with devices such as displays, RTC clock, etc.

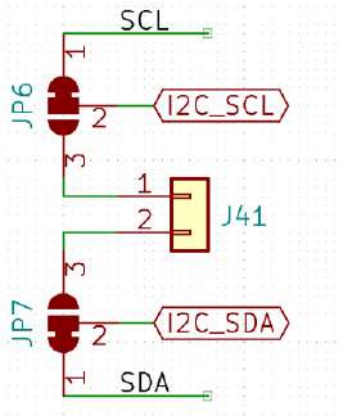


Figure 27 – I²C signals routing on the IoT Proto Shield Plus

14.1. Default connections

I²C signals on the **IoT Proto Shield Plus** are routed by default according to the following connections:

	ESP32	ARDUINO	ESP8266
I2C_SCL	GPIO22	A5	GPIO5 (D2)
I2C_SDA	GPIO21	A4	GPIO4 (D1)

14.2. Scanning for I2C devices

You can use the sample sketch `IoTPSP_I2C_scanner` (location: `IoTPSP_Sample_Sketches\IoTPSP_I2C_scanner`) to scan I²C.
The scanner shows connected I²C devices addresses. With no external devices (such as OLED displays etc.) the scanner shows the address of the PCF8574 onboard chip (usually 0x20 or 0x38).

14.3. Using the I²C with different GPIO pins

Most of the libraries used to communicate to I²C devices, use the default I²C GPIO pins (see §14.1), thus **it is NOT recommended to reroute I2C signals**.
Anyway, if you want or need, follow the instructions below.
According to Figure 27, I²C signals are routed to the default GPIO pins by means of JP6 and JP7.

To disconnect I²C signals from their default GPIO pins, you must OPEN JP6 and JP7 pads 1-2 (cut in the middle of pad 1 and pad 2) and CLOSE pads 2 with pad 3 (by means of a little drop of tin).
Solder a header on J41 and, by means of some jumper wires, connect other GPIO pins according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** (see §8) to avoid conflict with other default connections.

15. SSD1306 128x64 OLED DISPLAY (I²C type)

The **IoT Proto Shield Plus** is ready to work with 4 pins I²C SSD1306 128x64 OLED displays of widespread use.
To work with a I²C SSD1306 128x64 OLED display, just plug the module into the BRD9 header according to Figure 30.

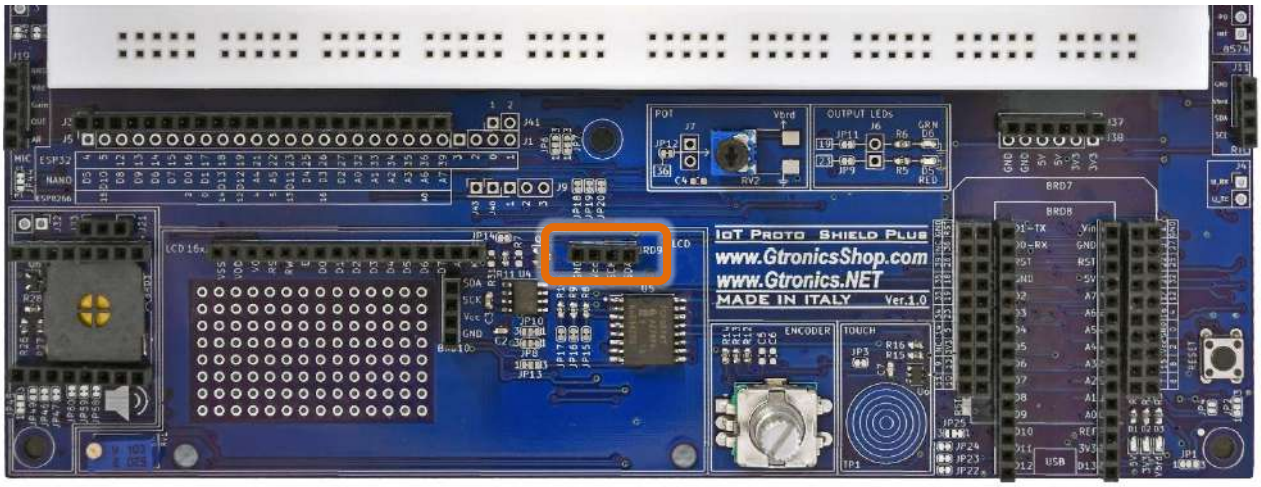


Figure 28 – BRD9 OLED DISPLAY 128x64 header of the IoT Proto Shield Plus

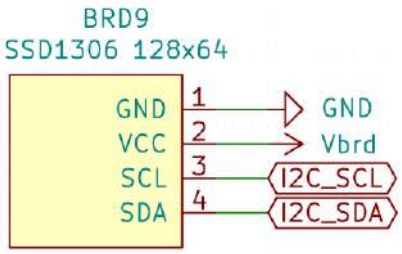


Figure 29 – Schematic of the IoT Proto Shield Plus BRD9 OLED DISPLAY 128x64 header

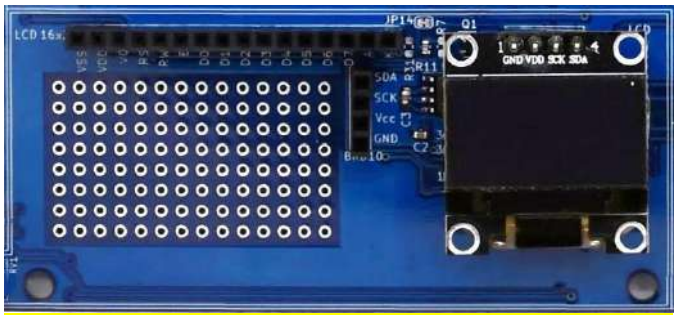


Figure 30 – An I²C SSD1306 128x64 OLED display module plugged into BRD9 header of the IoT Proto Shield Plus

15.1. Default connections

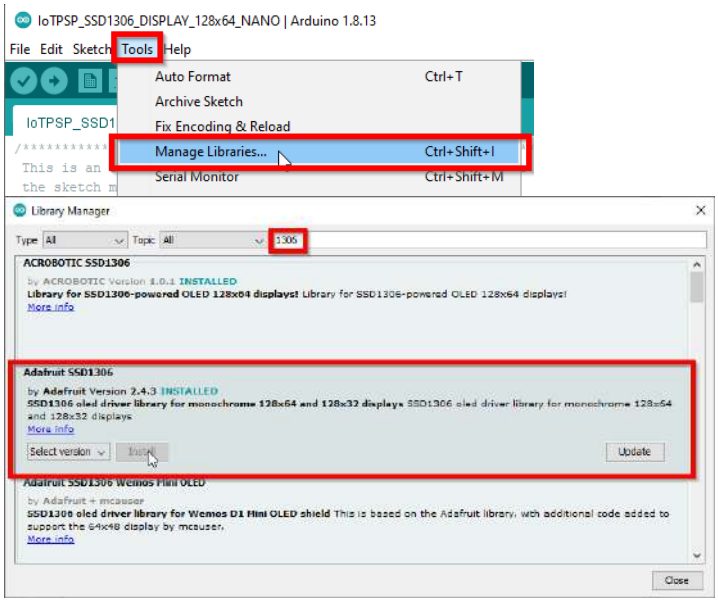
Refer to § 14.1 to see default routing of I²C signals on the **IoT Proto Shield Plus.**

15.2. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_SSD1306_DISPLAY_128x64`

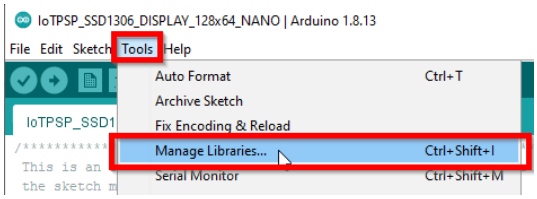
15.3. IoTPSP_SSD1306_DISPLAY_128x64_NANO

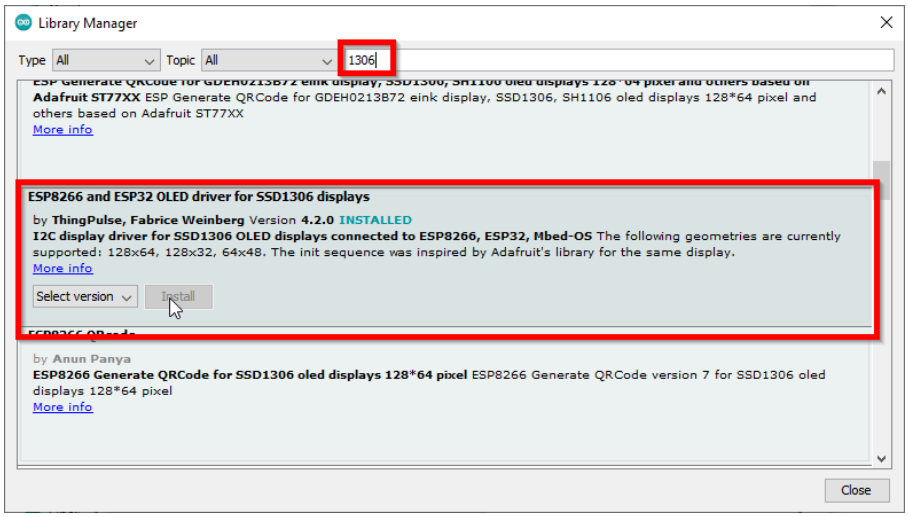
The `IoTPSP_SSD1306_DISPLAY_128x64_NANO` sketch, used to test the **IoT Proto Shield Plus** with NANO boards, uses the *Adafruit SSD1306* library.
Be sure to install it before using the sketch.



15.4. IoTPSP_SSD1306_DISPLAY_128x64_ESP32

The `IoTPSP_SSD1306_DISPLAY_128x64_ESP32` sketch, used to test the **IoT Proto Shield Plus** with ESP32 boards, uses the *ESP8266 and ESP32 OLED driver for SSD1306 displays* library.
Be sure to install it before using the sketch.





15.5. IoTPSP_SSD1306_DISPLAY_128x64_ESP8266

The `IoTPSP_SSD1306_DISPLAY_128x64_ESP8266` sketch, used to test the **IoT Proto Shield Plus** with ESP8266 boards, uses the *ESP8266 and ESP32 OLED driver for SSD1306 displays* library.
Be sure to install it before using the sketch (see § 15.4).

15.6. Using the SSD1306 128x64 OLED display with different GPIO pins

Refer to § 14.3 to see how to use I^2C with different GPIO pins on the **IoT Proto Shield Plus.**

16. SSD1306 128x32 OLED DISPLAY (I²C type)

The **IoT Proto Shield Plus** is ready to work with 4 pins I²C SSD1306 128x32 OLED displays of widespread use.
To work with a I²C SSD1306 128x32 OLED display, just plug the module into the BRD10 header according to Figure 33.

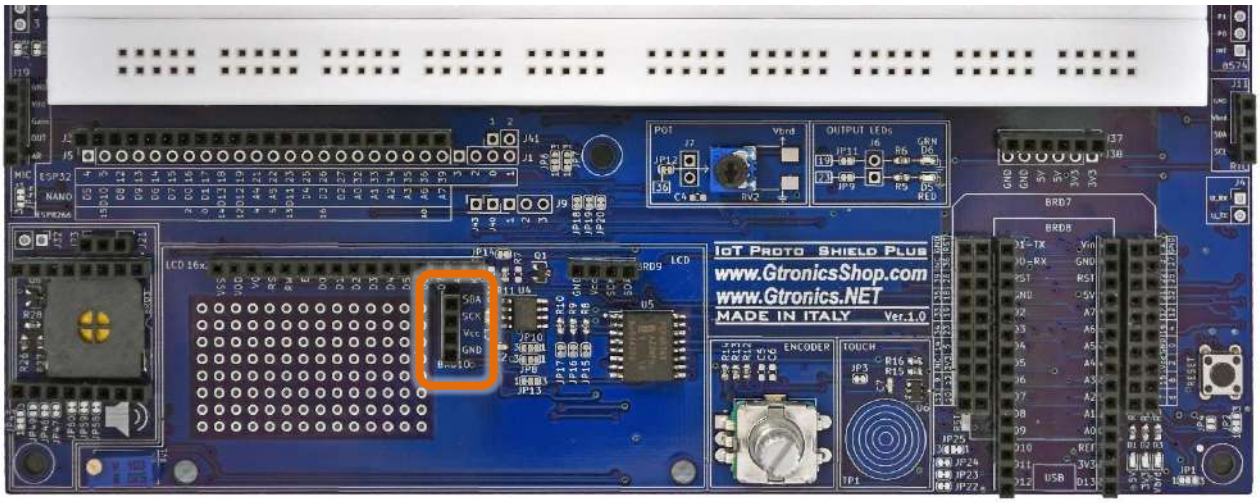


Figure 31 – BRD10 OLED DISPLAY 128x32 header of the IoT Proto Shield Plus

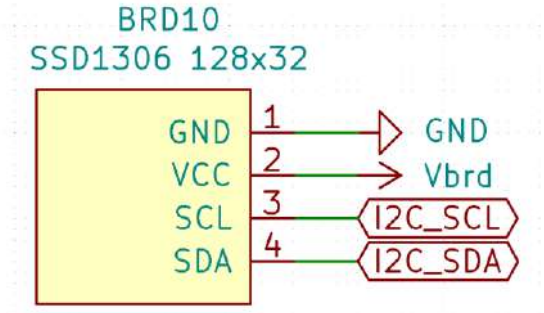


Figure 32 – Schematic of the IoT Proto Shield Plus BRD10 OLED DISPLAY 128x32 header

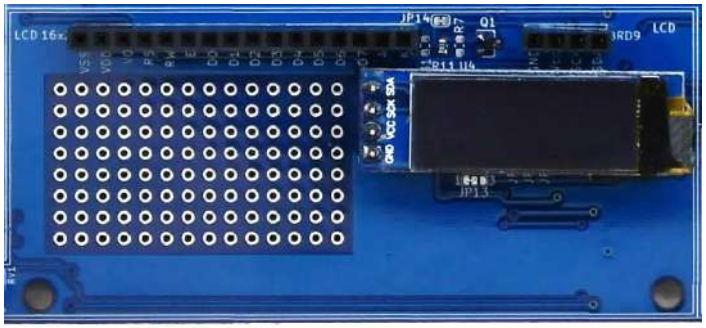


Figure 33 – An I²C SSD1306 128x32 OLED display module plugged into BRD10 header of the IoT Proto Shield Plus

16.1. Default connections

Refer to § 14.1 to see default routing of I²C signals on the **IoT Proto Shield Plus.**

16.2. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_SSD1306_DISPLAY_128x32`

16.3. IoTPSP_SSD1306_DISPLAY_128x32_NANO

The `IoTPSP_SSD1306_DISPLAY_128x32_NANO` sketch, used to test the **IoT Proto Shield Plus** with NANO boards, uses the *Adafruit SSD1306* library.
Be sure to install it before using the sketch (see § 15.3).

16.4. IoTPSP_SSD1306_DISPLAY_128x32_ESP32

The `IoTPSP_SSD1306_DISPLAY_128x32_ESP32` sketch, used to test the **IoT Proto Shield Plus** with ESP32 boards, uses the *ESP8266 and ESP32 OLED driver for SSD1306 displays* library.
Be sure to install it before using the sketch (see § 15.4).

16.5. IoTPSP_SSD1306_DISPLAY_128x32_ESP8266

The `IoTPSP_SSD1306_DISPLAY_128x32_ESP8266` sketch, used to test the **IoT Proto Shield Plus** with ESP8266 boards, uses the *ESP8266 and ESP32 OLED driver for SSD1306 displays* library.
Be sure to install it before using the sketch (see § 15.4).

16.6. Using the SSD1306 128x32 OLED display with different GPIO pins

Refer to § 14.3 to see how to use I²C with different GPIO pins on the **IoT Proto Shield Plus.**

17. LCD 1602 (Liquid Crystal Display)

The **IoT Proto Shield Plus** is ready to work with 16 pins LCD 1602 displays of widespread use.
To work with an LCD 1602 display, just plug the module into the LCD16x2 header according to Figure 35.
Use RV1 to adjust the LCD contrast.
The LCD 1602 is driven by I²C by means of the onboard PCF8574 (I²C I/O expander) and it does not require any additional wiring.
Signal connections are the same used in the LCD I²C adapter of widespread use.
Since ESP32, ESP8266 and NANO33 boards works at 3.3V, to have a sufficient contrast on the display, ICL7660 is used to ensure a sufficient voltage to the V0 pin of the LCD 1602 module.
Depending on component availability, the I/O expander 8574 could be the PCF8574 or the PCF8574A.
Both versions perform the same, the only difference is the base address of the I²C communication.
Since A0, A1 and A2 of the PCF8574 connect to ground by default (by means of JP17, JP16 and JP15), the default address is set to 0x20 (PCF8574) or 0x38 (PCF8574A).
Use the sample sketches (see § 17.2) to find out which version is your one.

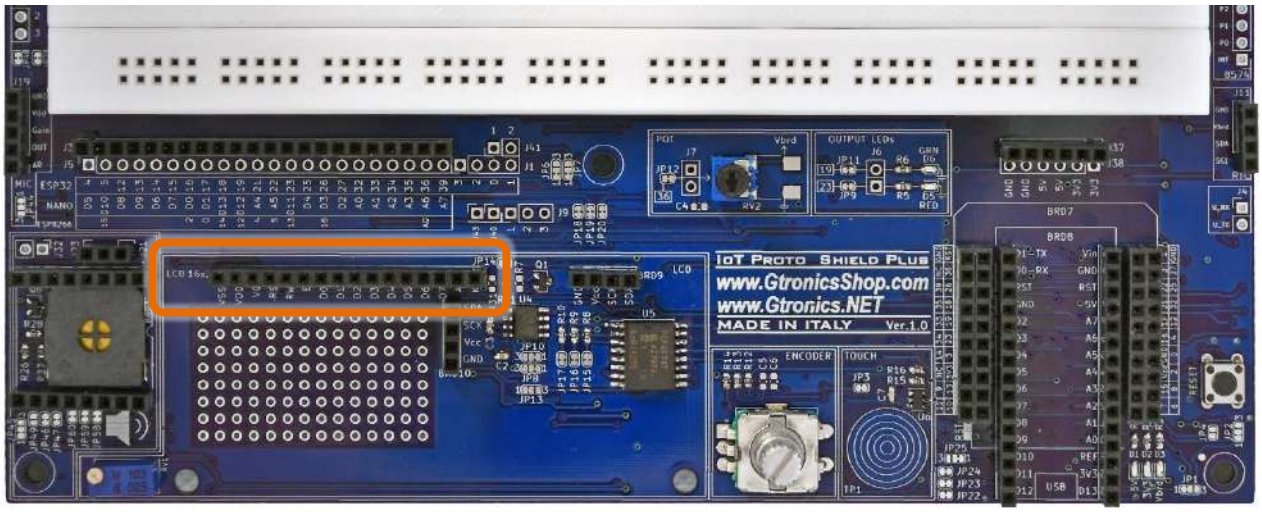


Figure 34 – LCD 16x2 header of the IoT Proto Shield Plus

Figure 35 – An LCD 1602 display module plugged into the 16 pins header of the IoT Proto Shield Plus

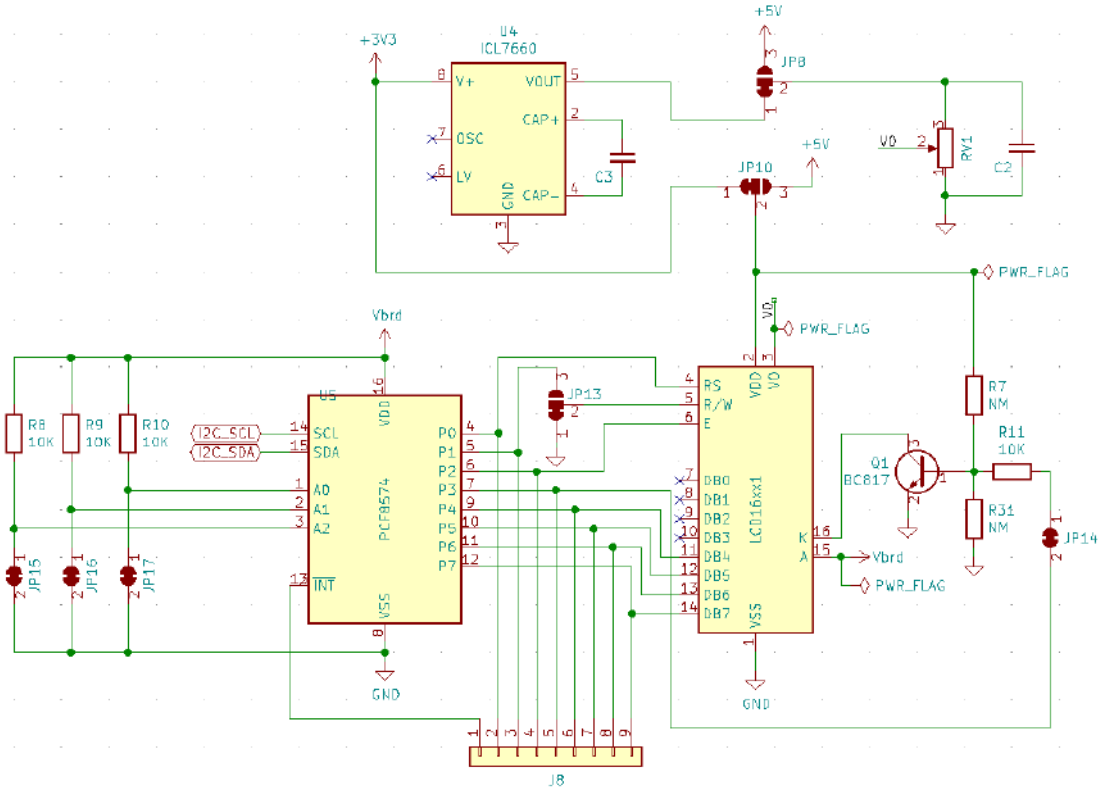


Figure 36 – Schematic of the IoT Proto Shield Plus LCD 16x2 section

17.1. Default connections

Refer to § 14.1 to see default routing of I²C signals on the **IoT Proto Shield Plus.**

17.2. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_LCD16x2`

These sketches make use of the *LiquidCrystal_I2C library*, you will find the library in the `IoTPSP_Sample_Sketches\IoTPSP_LCD16x2` path.

These sketches use by default the I2C default pins shown in § 14.1.

Use the `IoTPSP_LCD16x2_NANO` sketch to test the LCD1602 with ARDUINO NANO boards.
Use the `IoTPSP_LCD16x2_ESP32` sketch to test the LCD1602 with ESP32 boards.
Use the `IoTPSP_LCD16x2_ESP8266` sketch to test the LCD1602 with ESP8266 boards.

17.3. Changing the I²C address of the PCF8574

It is not very usual, but in case that you want / need to change the address of the 8574 chip, you have to OPEN or CLOSE JP17, JP16 and JP15 according to your needs and the PCF8574 datasheet.
To OPEN JP17, JP16 or JP15 cut them in the middle.
To CLOSE them solder a little drop of tin.

17.4. Using the PCF8574 with different I²C GPIO pins

Refer to § 14.3 to see how to use I²C with different GPIO pins on the **IoT Proto Shield Plus.**

18. PCF8574 I2C I/O expander

If you do not need to use the LCD 1602, you can use the on-board 8574 as a general-purpose I/O expander. All the I/O pins of the 8574 are available on the right side of the **IoT Proto Shield Plus** (see Figure 37).

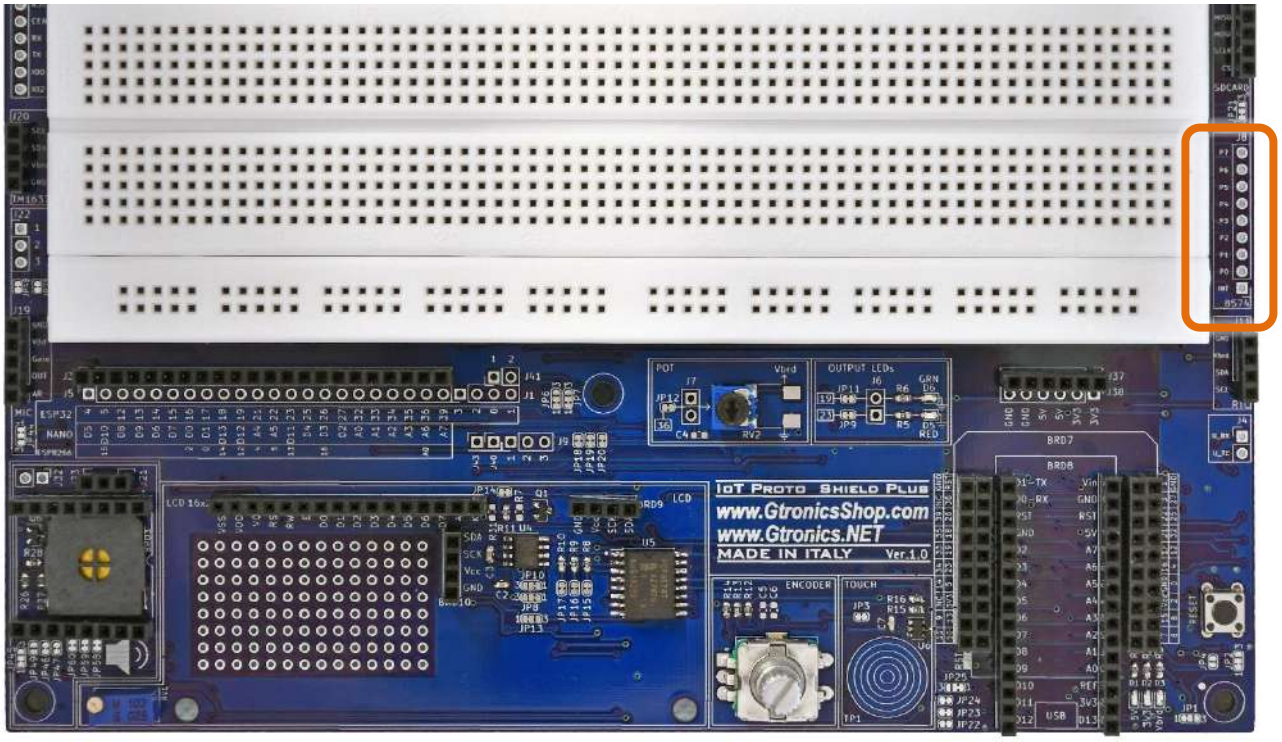
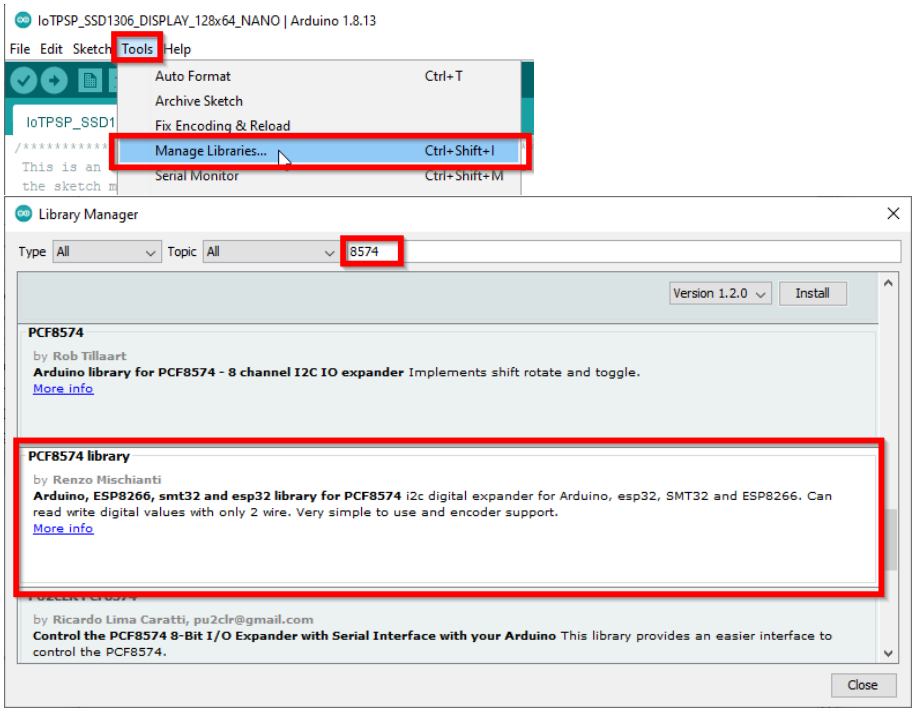


Figure 37 – The 8574 pins used for general-purpose I/O

18.1. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_PCF8574`

These sketches use by default the I2C default pins shown in § 14.1.
These sketches use the *PCF8574 library*, be sure to install it before using the sketches.



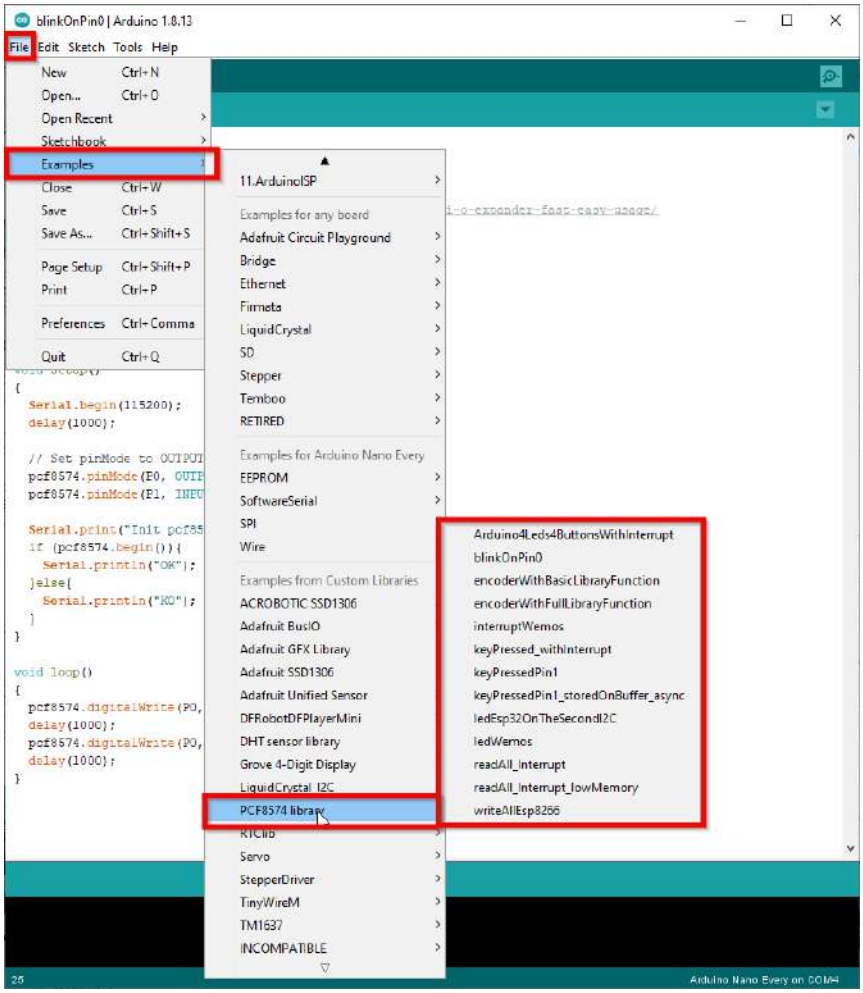
18.2. IoTPSP_PCF8574_GET_ADDRESS

As said before, depending on component availability, the I/O expander 8574 could be the PCF8574 or the PCF8574A.
Both versions perform the same, the only difference is the base address of the I^2C communication.
Since A0, A1 and A2 of the PCF8574 connect to ground by default (by means of JP17, JP16 and JP15), the default address is set to 0x20 (PCF8574) or 0x38 (PCF8574A).

If you have an LCD 1602 you can get the address of the PCF8574 using sketches shown in § 17 (the address of your PCF8574 is shown on the display).

If you do not have an LCD 1602, then you can use the `IoTPSP_PCF8574_GET_ADDRESS` sketch.
The sketch works with ESP32, ESP8266 and NANO boards.
The sketch looks for addresses between 0x20 to 0x27 (PCF8574) or between 0x38 to 0x3F (PCF8574A) and print them on the serial monitor.

Experiment with the examples provided with the *PCF8574 library*.


blinkOnPin0 | Arduino 1.8.13
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
11.ArduinoISP
Examples for any board
Adafruit Circuit Playground
Bridge
Ethernet
Firmata
LiquidCrystal
SD
Stepper
Temboo
RETIRED
Examples for Arduino Nano Every
EEPROM
SoftwareSerial
SPI
Wire
Examples from Custom Libraries
ACROBOTIC SSD1306
Adafruit BusIO
Adafruit GFX Library
Adafruit SSD1306
Adafruit Unified Sensor
DFRobotDFPlayerMini
DHT sensor library
Grove 4-Digit Display
LiquidCrystal_I2C
PCF8574 library
RTClib
Servo
StepperDriver
TinyWireM
TM1637
INCOMPATIBLE
Arduino4Leds4ButtonsWithInterrupt
blinkOnPin0
encoderWithBasicLibraryFunction
encoderWithFullLibraryFunction
interruptWemos
keyPressed_withInterrupt
keyPressedPin1
keyPressedPin1_storedOnBuffer_async
ledEsp32OnTheSecondI2C
ledWemos
readAll_Interrupt
readAll_Interrupt_lowMemory
writeAllEsp8266
Arduino Nano Every on COM4

19. TM1637 Digital display module

The **IoT Proto Shield Plus** is ready to work with 4 pins TM1367 seven segment digital display modules of widespread use.
To work with an TM1367 seven segment digital display, just plug the module into the J20 header according to Figure 40.

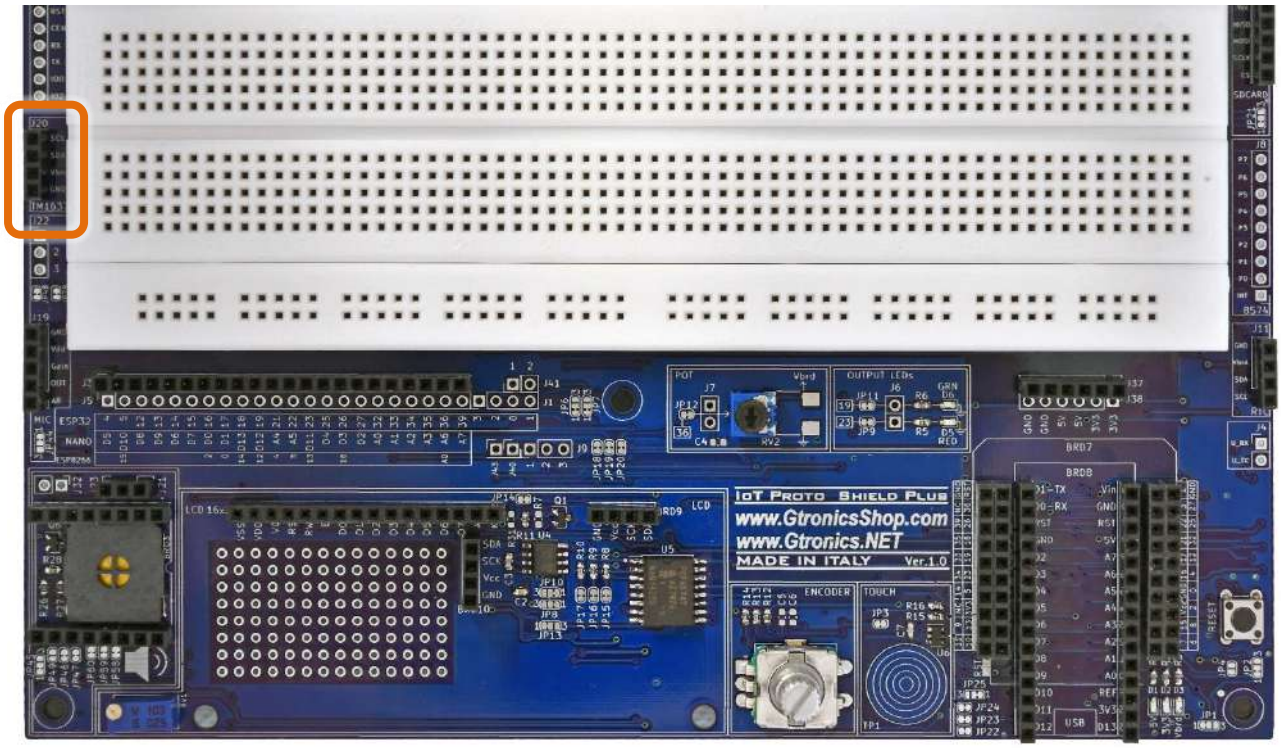


Figure 38 – J20 TM1367 digital display header of the IoT Proto Shield Plus

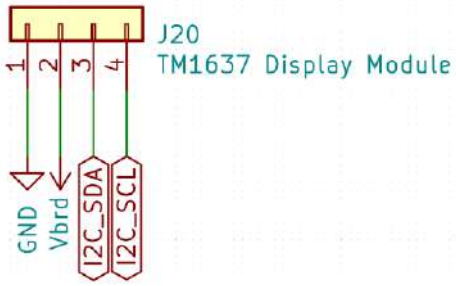


Figure 39 – Schematic of the IoT Proto Shield Plus J20 TM1367 digital display header

Figure 40 – An TM1367 digital display module plugged into J20 header of the IoT Proto Shield Plus

19.1. Default connections

As shown in Figure 39, the TM1637 section of the **IoT Proto Shield Plus** uses the I^2C routing to drive the TM1637 chip.
In some cases, this crate a conflict if you are using other I^2C devices combined with the TM1637.
Refer to § 14.1 to see default routing of I^2C signals on the **IoT Proto Shield Plus.**

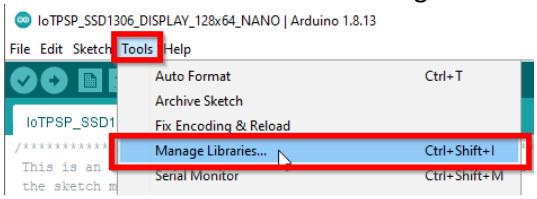
19.2. Sample Sketches

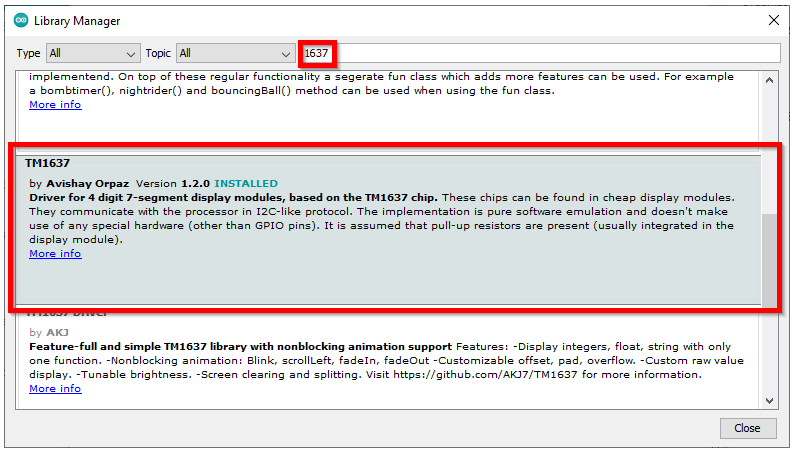
Location: `IoTPSP_Sample_Sketches\IoTPSP_TM1637_Display`

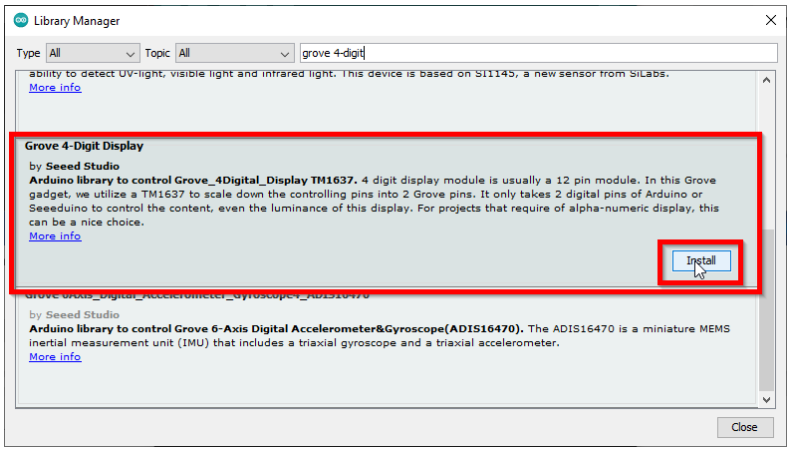
These sketches are based on the tutorial at the following link:
https://create.arduino.cc/projecthub/ryanchan/tm1637-digit-display-arduino-quick-tutorial-ca8a93

The tutorial has been modified in order to match the **IoT Proto Shield Plus** default routing of the I2C signals shown in § 14.1.

The sketches make use of the *TM1637* and the *Grove 4-Digit Display* libraries.
Be sure to install them before using the sketches.







Use the `IoTPSP_TM1637_Display_NANO` sketch to test the TM1637 display with ARDUINO NANO boards.
Use the `IoTPSP_TM1637_Display_ESP32` sketch to test the TM1637 display with ESP32 boards.
Use the `IoTPSP_TM1637_Display_ESP8266` sketch to test the TM1637 display with ESP8266 boards.

19.3. Using the TM1637 display with different GPIO pins

Refer to § 14.3 to see how to use I^2C with different GPIO pins on the **IoT Proto Shield Plus.**

20. RELAY

The **IoT Proto Shield Plus** is ready to work with 3 pins relay modules of widespread use.
To work with a relay module, just plug the module into the J27 header.

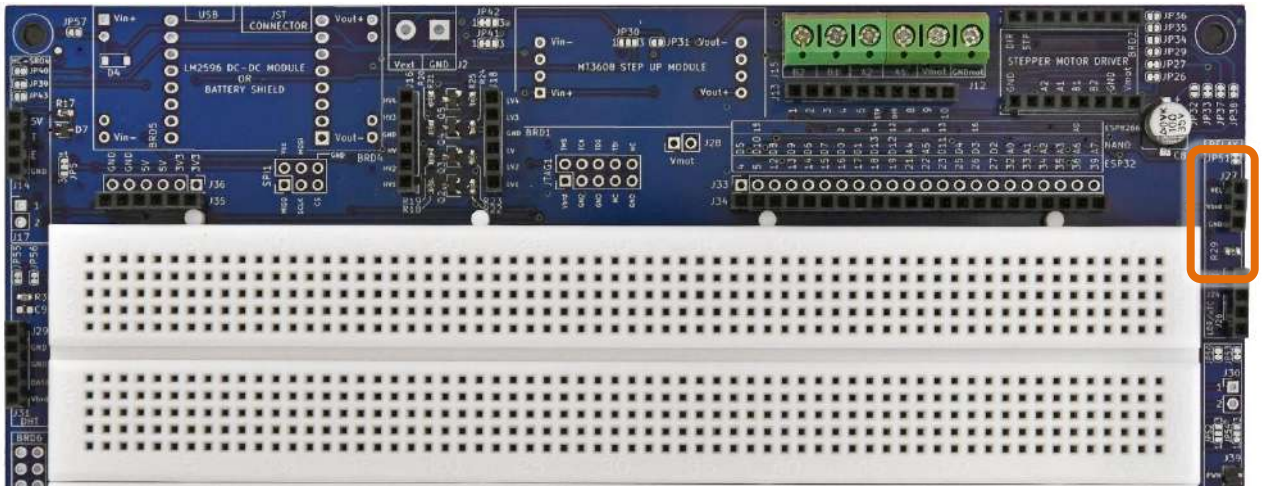


Figure 41 – RELAY section of the IoT Proto Shield Plus

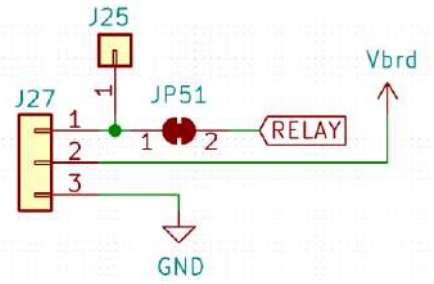


Figure 42 – Schematic of the IoT Proto Shield Plus RELAY section

Figure 43 – A Relay module plugged into J27 header of the IoT Proto Shield Plus

20.1. Default connections

	ESP32	ARDUINO	ESP8266
RELAY	GPIO12	D8	*NC*

NC = Not Connected

Please note that there are no default connections for the ESP8266. If you need to use the RELAY module with ESP8266 boards, you need to route the signals with jumper wires.

20.2. Suggested ESP8266 connections

The sample sketch is made to work with this connection:
ESP8622 GPIO 0 (D3) connected to header J3 ESP32 GPIO12 (RELAY).
ESP8622 GPIO 14 (D5) connected to header J3 ESP32 GPIO32 (TOUCH).

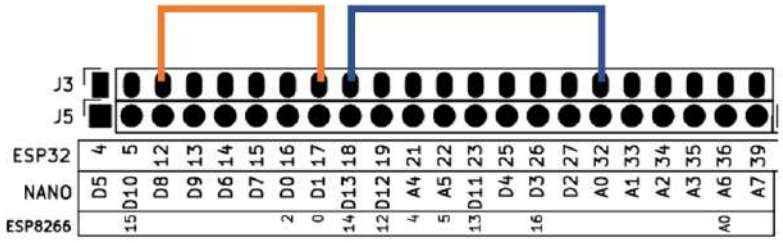


Figure 44 – Suggested connections to work with RELAY and ESP8266

20.3. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_RELAY`

Use the `IoTPSP_RELAY_NANO` sketch to test the RELAY with ARDUINO NANO boards.
Use the `IoTPSP_RELAY_ESP32` sketch to test the RELAY with ESP32 boards.
Use the `IoTPSP_RELAY_ESP8266` sketch to test the RELAY with ESP8266 boards.

These sketches use the touch sensor to switch the RELAY on.
Put your finger on the touch circles, the RELAY will act, and the GRN LED will turn on according to your touch.

See § 13 for more details about TOUCH sensor on the **IoT Proto Shield Plus.**

20.4. Using RELAY with a different GPIO pin

According to Figure 42, RELAY connects to the default GPIO pin by means of JP51.
To disconnect RELAY from its default GPIO pin, you must OPEN JP51 (cut it in the middle).
By means of a jumper wire, connect J25 to another GPIO pin according to your needs.

Please check the default signal routing of the **IoT Proto Shield Plus** (see §8) to avoid conflict with other default connections.

21. DHTxx (humidity and temperature sensor)

The **IoT Proto Shield Plus** is ready to work with three pins DHT22 or DHT11 modules or "naked" DHT22 or DHT11 sensors of widespread use (see Figure 45).
To work with a DHT module or a naked DHT sensor, just plug it into the J31 header (see Figure 46, Figure 48 and Figure 49).
The DHT naked sensor has four pins. Pin #3 is Not Connected, but if you connect pin#3 to GND it works as well.
So, if you are working with a naked DHT sensor, you can plug it in two different ways: with pin #3 connected to GND or with bent pin #3 (so it will be Not Connected).

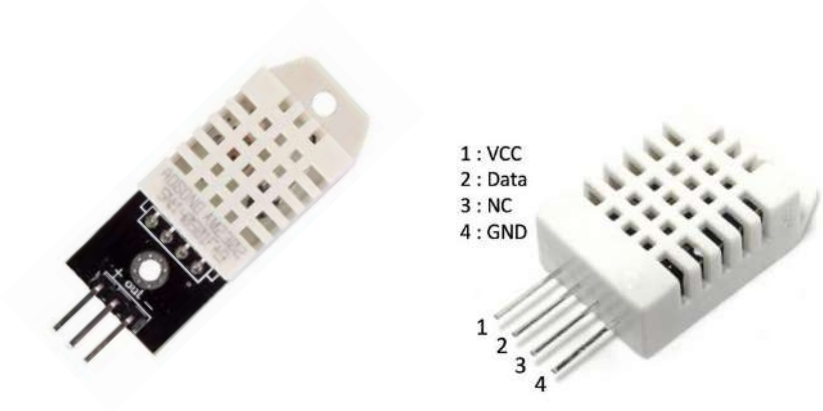


Figure 45 – DHT22 three pins module (on the left), Naked DHT22 sensor (on the right)

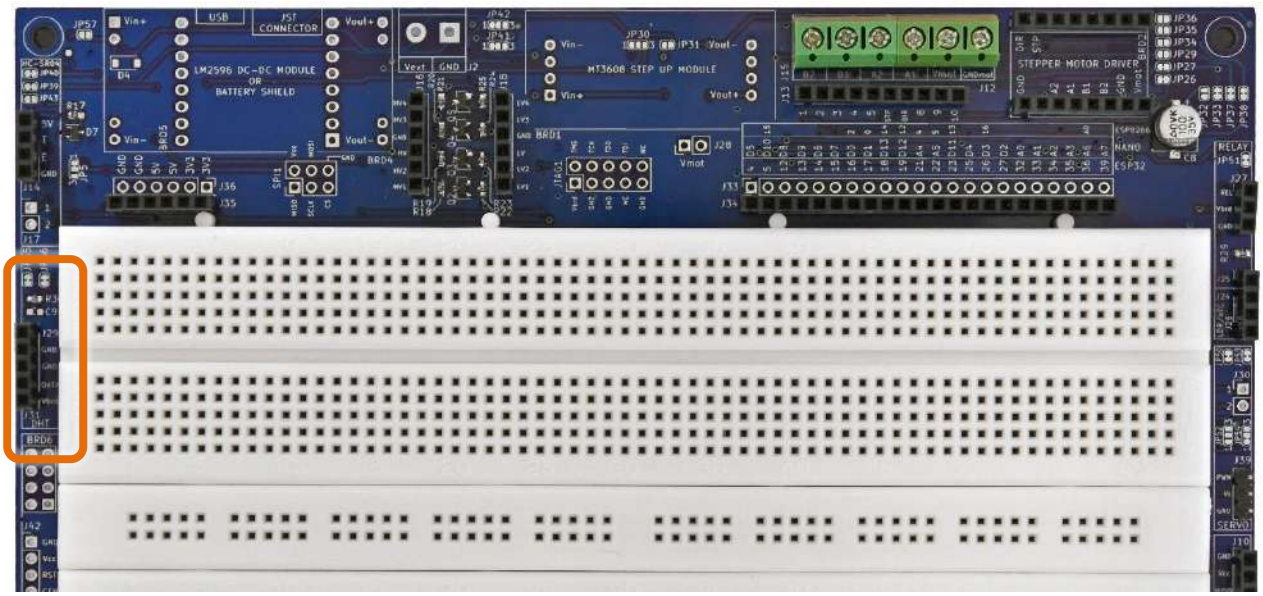


Figure 46 – DHT section of the IoT Proto Shield Plus

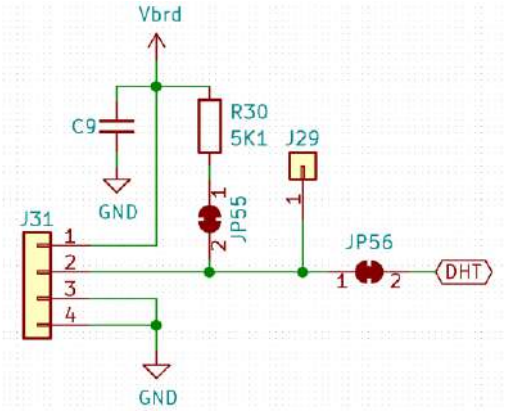


Figure 47 – Schematic of the IoT Proto Shield Plus DHT section

Figure 48 – A DTH22 module plugged into J31 of the IoT Proto Shield Plus

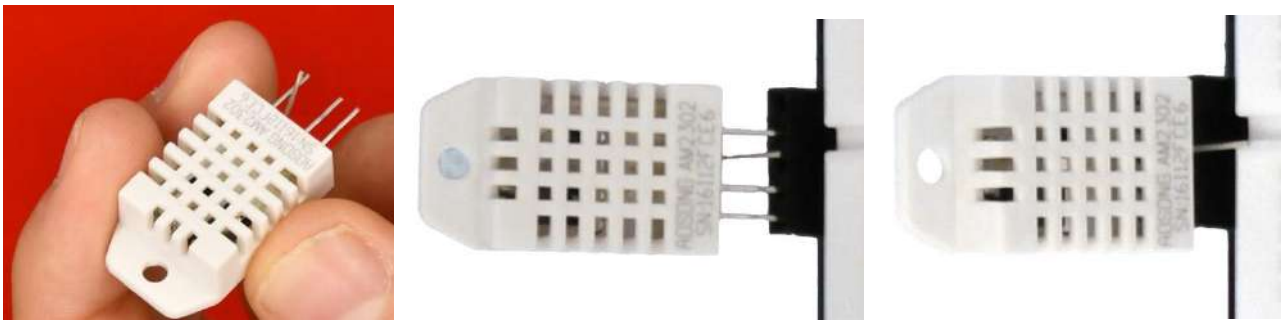


Figure 49 – A Naked DTH22 module plugged into J31 of the IoT Proto Shield Plus

21.1. Default connections

	ESP32	ARDUINO	ESP8266
DHT	GPIO13	D9	*NC*

NC = Not Connected

Please note that there are no default connections for the ESP8266. If you need to use the DHT with ESP8266 boards, you need to route the signals with jumper wires.

21.2. Suggested ESP8266 connections

The sample sketch is made to work with this connection:
ESP8622 GPIO 0 (D3) connected to header J3 ESP32 GPIO13 (DHT).

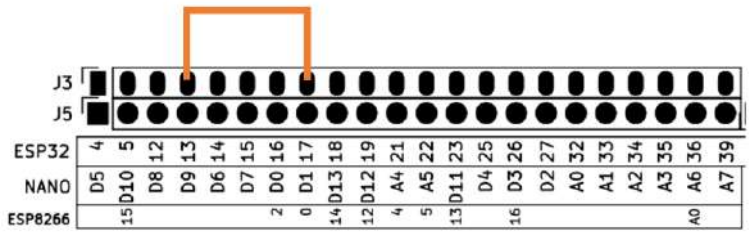


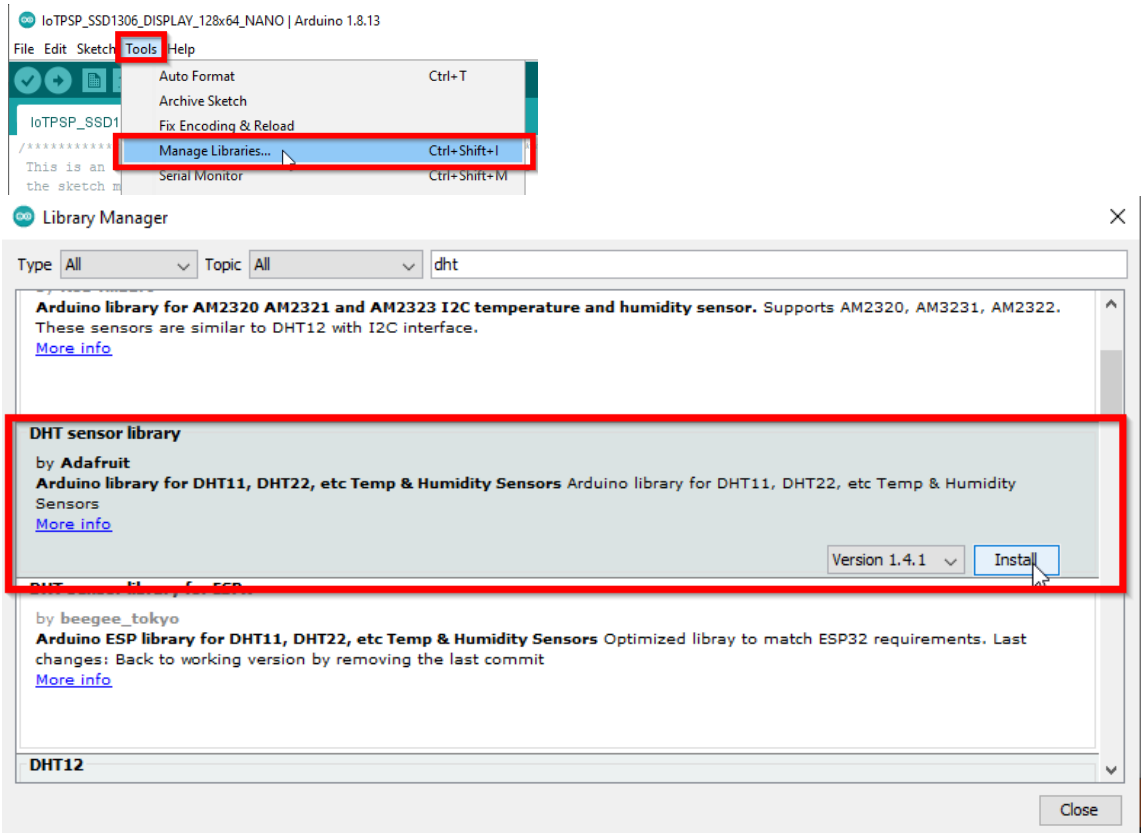
Figure 50 – Suggested connections to work with DHT and ESP8266

21.3. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_DHT`
These sketches work out of the box with a DHT22 module or DHT22 naked sensor.

21.4. IoTPSP_DHT_NANO

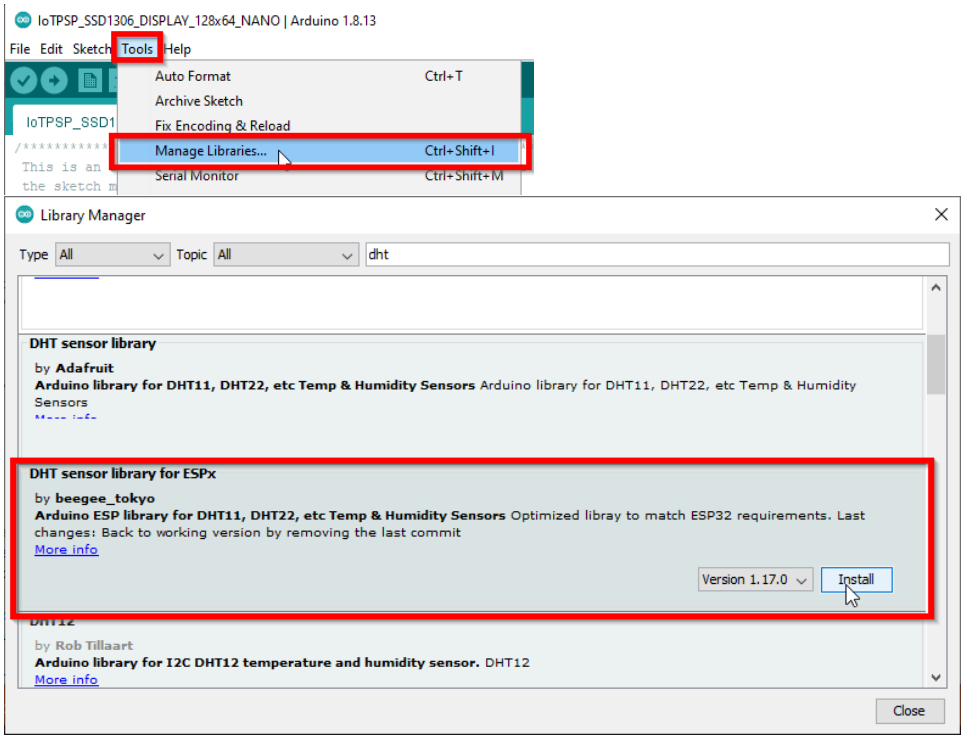
The `IoTPSP_DHT_NANO` uses the *DHT sensor library*, be sure to download it before using the sketch.



Use the `IoTPSP_DHT_NANO` sketch to test the DHT with ARDUINO NANO boards.
Open the serial monitor to see the Humidity and Temperature data coming from the DHT sensor.

21.5. IoTPSP_DHT_ESP32

The `IoTPSP_DHT_ESP32` uses the *DHT sensor library for ESPx*, be sure to download it before using the sketch.



Use the `IoTPSP_DHT_ESP32` sketch to test the DHT with ESP32 boards.
Open the serial monitor to see the Humidity and Temperature data coming from the DHT sensor.

21.6. IoTPSP_DHT_ESP8266

The `IoTPSP_DHT_ESP32` uses the *DHT sensor library for ESPx*, be sure to download it before using the sketch (see § 21.5).
Use the `IoTPSP_DHT_ESP8266` sketch to test the DHT with ESP8266 boards.
Open the serial monitor to see the Humidity and Temperature data coming from the DHT sensor.

21.7. Using DHT with a different GPIO pin

According to Figure 47, DHT connects to the default GPIO pin by means of JP56.
To disconnect DHT from its default GPIO pin, you must OPEN JP56 (cut it in the middle).
By means of a jumper wire, connect J29 to another GPIO pin according to your needs.

Please check the default signal routing of the **IoT Proto Shield Plus** (see §8) to avoid conflict with other default connections.

22. RTC (Real Time Clock)

The **IoT Proto Shield Plus** is ready to work with I²C RTC modules of widespread use (see Figure 51).
The board can work with both RTC DS3231 or DS1307, we suggest DS3231 which is a lot more stable: the time drift is really low even on long period of time.
To work with an RTC module, just plug it into the J11 header.
Since J11 is a four pins header, just check signals on the DS module you are using to be sure to match the J11 pinout (GND, Vcc, SDA and SCL).
Some boards (such as Arduino NANO33 and ESP32) have their own internal RTC, this section is about working with DS3231 or DS1307 RCT modules.

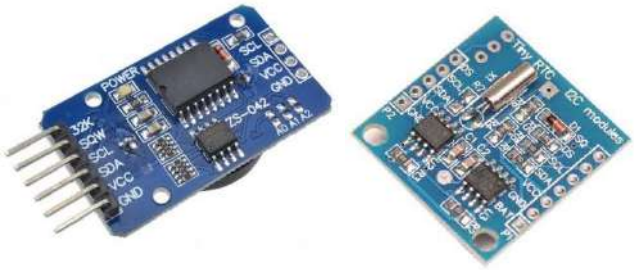


Figure 51 – RTC modules of widespread use, DS3231 (on the left), DS1307 (on the right)

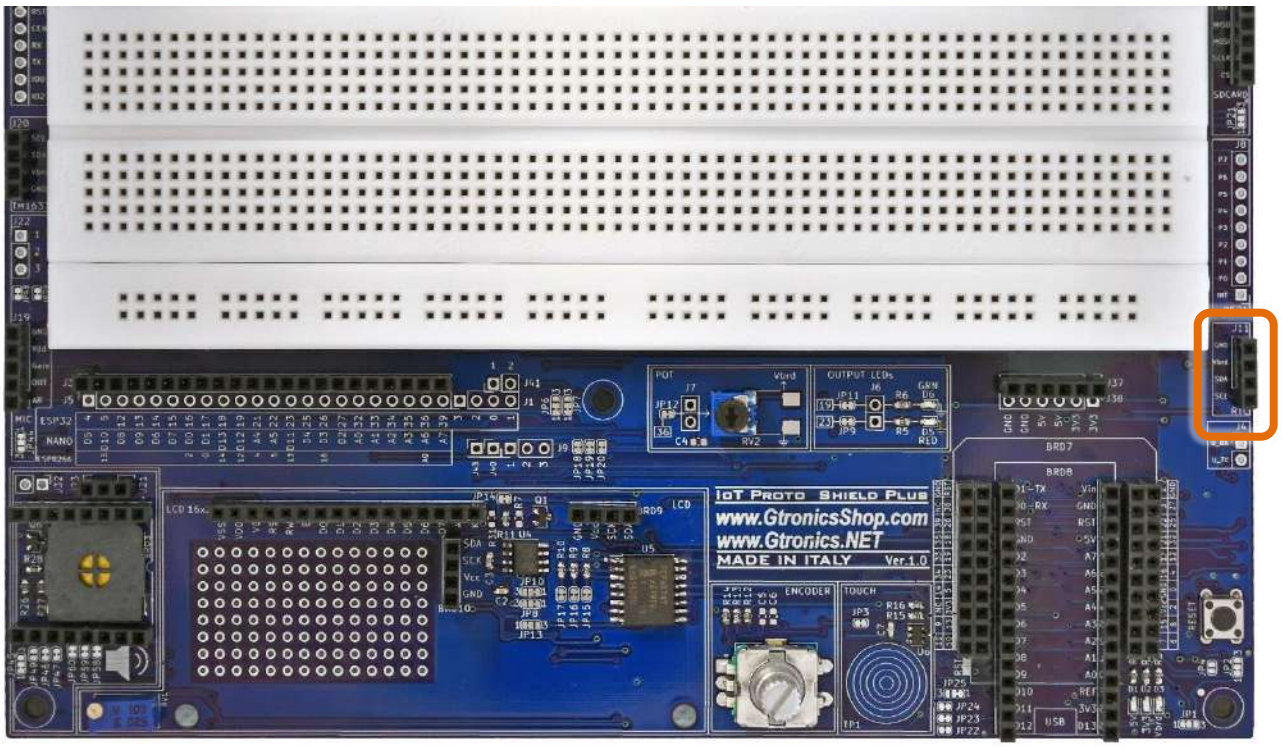


Figure 52 – J11 RTC header of the IoT Proto Shield Plus

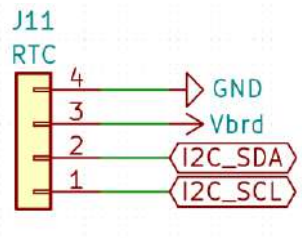


Figure 53 – Schematic of the IoT Proto Shield Plus J11 RTC header

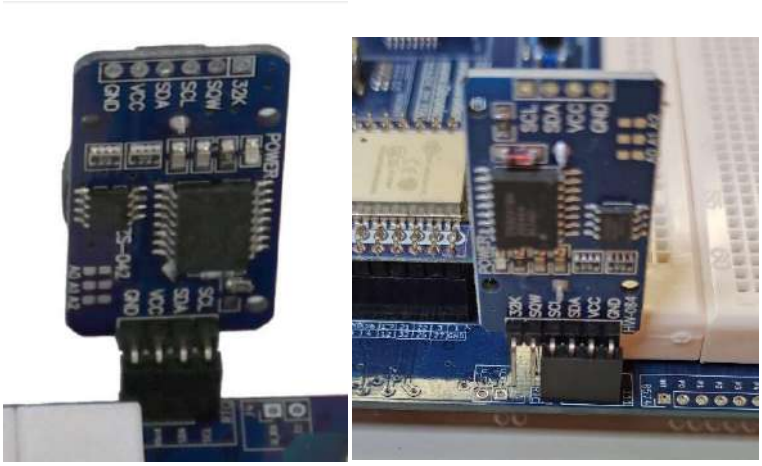


Figure 54 – An RTC DS3231 module plugged into J11 header of the IoT Proto Shield Plus

Usually, these modules have a battery holder, and the module is provided with a battery charger circuit. If you are using a rechargeable battery, everything would be ok, but if you are using a NOT rechargeable battery it is suggested to "disable" the charging circuit in order not to damage your battery.
The easiest way to "disable" it, is to remove the diode of the charging circuit (see Figure 55).

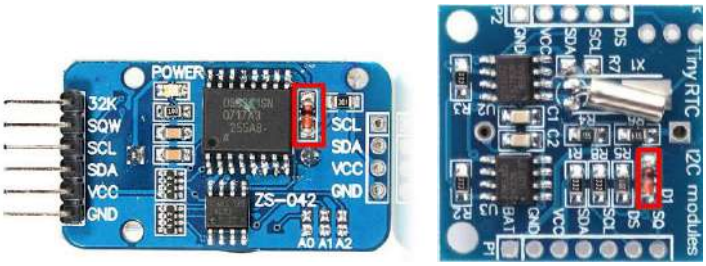


Figure 55 – The diode of the charging circuit on a DS3231 (on the left), and on a DS1307 (on the right)

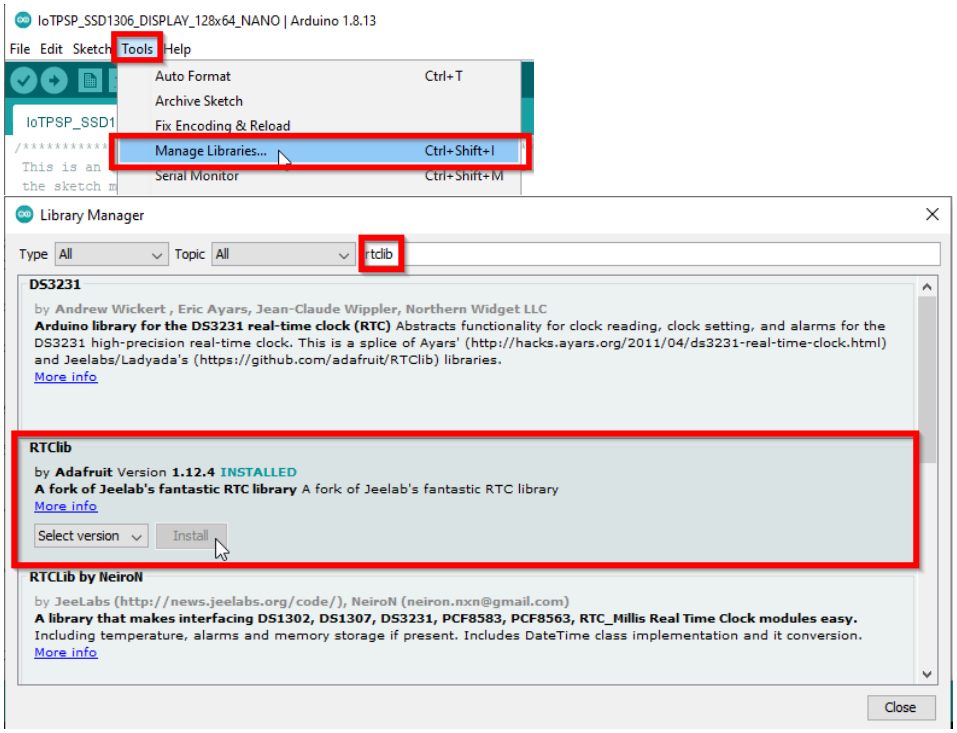
22.1. Default connections

Refer to § 14.1 to see default routing of I^2C signals on the **IoT Proto Shield Plus.**

22.2. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_RTC`

The sketches make use of the *RTClib* library by Adafruit.
Be sure to install it before using the sketches.



Use the `IoTPSP_RTC_NANO` sketch to test the RTC with ARDUINO NANO boards.
Use the `IoTPSP_RTC_ESP32` sketch to test the RTC with ESP32 boards.
Use the `IoTPSP_RTC_ESP8266` sketch to test the RTC with ESP8266 boards.

Open the serial monitor to see date and time red from the RTC module.
If you are using a DS3231 module you will get the temperature of the module too, if you are using a DS1307 the temperature data red will be meaningless.

22.3. Using the RTC with different GPIO pins

Refer to § 14.3 to see how to use I^2C with different GPIO pins on the **IoT Proto Shield Plus.**

23. SPI on the IoT Proto Shield Plus

SPI (Serial Peripheral Interface) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems (check out more about SPI Wikipedia or other online documentation).

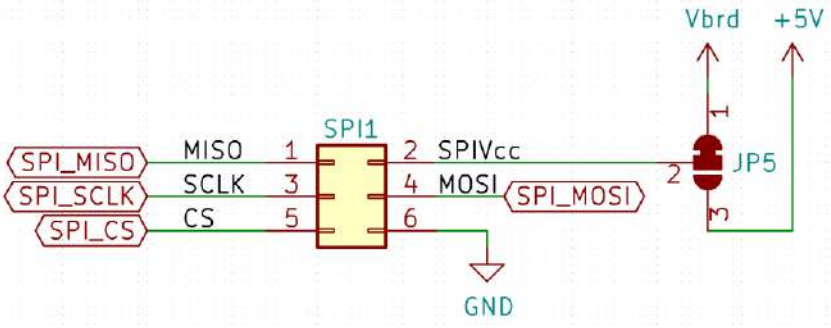


Figure 56 – SPI signals routing on the IoT Proto Shield Plus

23.1. Default connections

SPI signals on the **IoT Proto Shield Plus** are routed by default according to the following connections:

	ESP32	ARDUINO	ESP8266
SPI_MISO	GPIO19	D12	GPIO12 (D6)
SPI_MOSI	GPIO23	D11	GPIO13 (D7)
SPI_CS	GPIO5	D10	GPIO15 (D8)
SPI_SCLK	GPIO18	D13	GPIO14 (D5)

SPI on the **IoT Proto Shield Plus** is used to work with SD CARD modules (see § 24), or to any other device by means of the SPI1 socket (see Figure 57).

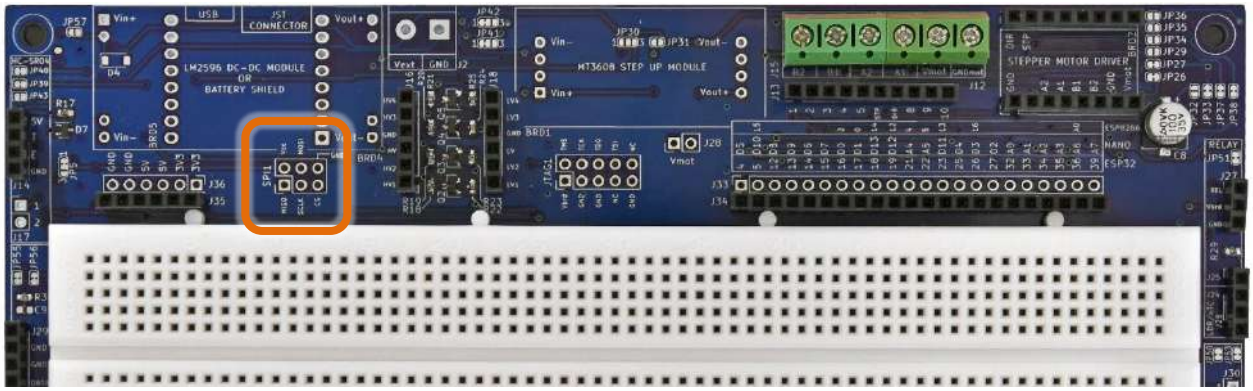


Figure 57 – SPI1 socket on the IoT Proto Shield Plus

24. SD CARD MODULE

The **IoT Proto Shield Plus** is ready to work with 6 pins SPI SD card modules of widespread use (see Figure 60).
To work with an SPI SD card module, just plug the module into the J10 header (see Figure 60).
Of course, you will need a microSD card too.

You may find useful information and knowledges about SD card at the following links:
https://www.arduino.cc/en/reference/SD
https://www.arduino.cc/en/Reference/SDCardNotes
https://create.arduino.cc/projecthub/electropeak/sd-card-module-with-arduino-how-to-read-write-data-37f390

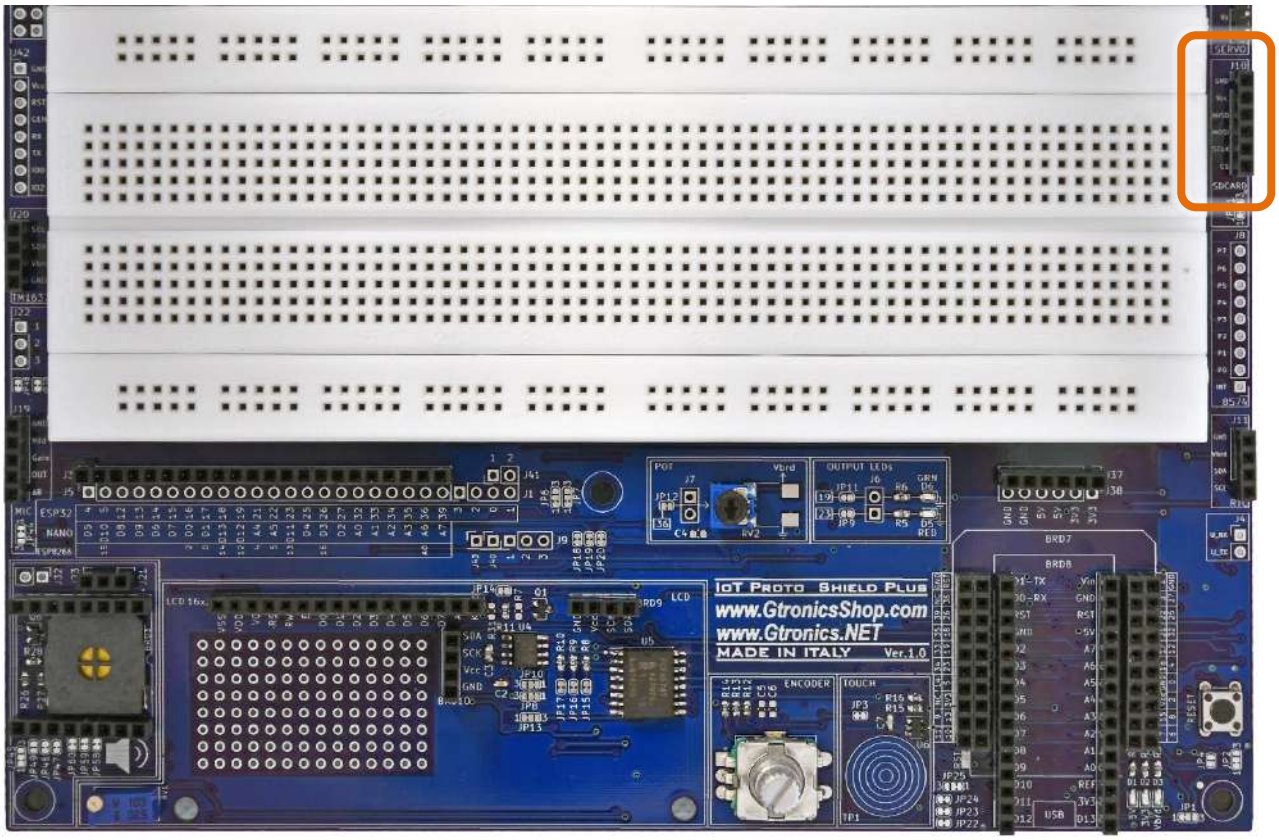


Figure 58 – J10 SD CARD header of the IoT Proto Shield Plus

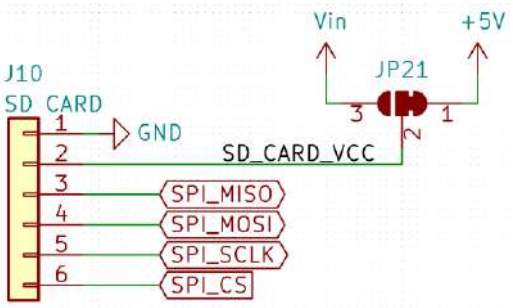


Figure 59 – Schematic of the IoT Proto Shield Plus J10 SD CARD header

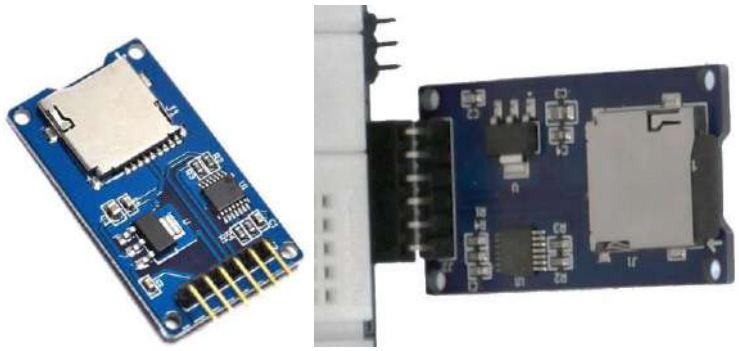


Figure 60 – An SPI SD CARD module (on the left)
and the module plugged into J10 header of the IoT Proto Shield Plus (on the right)

24.1. Default connections

Refer to § 23.1 to see default routing of SPI signals on the **IoT Proto Shield Plus.**

24.2. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_SD_CARD`

These sketches make use of the SD library, included in the Arduino IDE.
These sketches use by default SPI routing shown in § 23.1.

24.3. Testing the SD CARD

These sketches are very useful for testing a card when you're not sure if it's working or not.

Use the `IoTPSP_SD_CARD_INFO_NANO` sketch to get SD CARD info with ARDUINO NANO boards.
Use the `IoTPSP_SD_CARD_INFO_ESP32` sketch to get SD CARD info with ESP32 boards.
Use the `IoTPSP_SD_CARD_INFO_ESP8266` sketch to get SD CARD info with ESP8266 boards.

Open the serial monitor to see info form your SD CARD.
If your serial monitor is blank, press the reset button.

24.4. Reading and writing from and to the SD CARD

Once you tested your SD CARD and you are sure it is working (see § 24.3), you can read and write to your SD CARD with the following sketches.

Use the `IoTPSP_SD_CARD_RW_NANO` sketch to test writing data to SD CARD and read them back on the serial monitor with ARDUINO NANO boards.

Use the `IoTPSP_SD_CARD_RW_ESP32` sketch to test writing data to SD CARD and read them back on the serial monitor with ESP32 boards.
Use the `IoTPSP_SD_CARD_RW_ESP8266` sketch to test writing data to SD CARD and read them back on the serial monitor with ESP8266 boards.

Open the serial monitor to see data red from your SD CARD.
If your serial monitor is blank, press the reset button.

25. LDR/xTC

The **IoT Proto Shield Plus** is ready to work with LDR, PTC or NTC sensors.
LDR (light dependent resistor, also known as photoresistor) are light sensitive devices to measure the light intensity (more info on Wikipedia).
NTC and PTC (also knowns as thermistor) are resistors whose resistance is strongly dependent on temperature.
NTC are negative temperature coefficient sensors, while PTC are positive temperature coefficient sensors (more info on Wikipedia).
To work with one of these sensors, just plug it into J26 header.
Since these sensors act like a variable resistor the principle is the same you have with a voltage divider (POT).

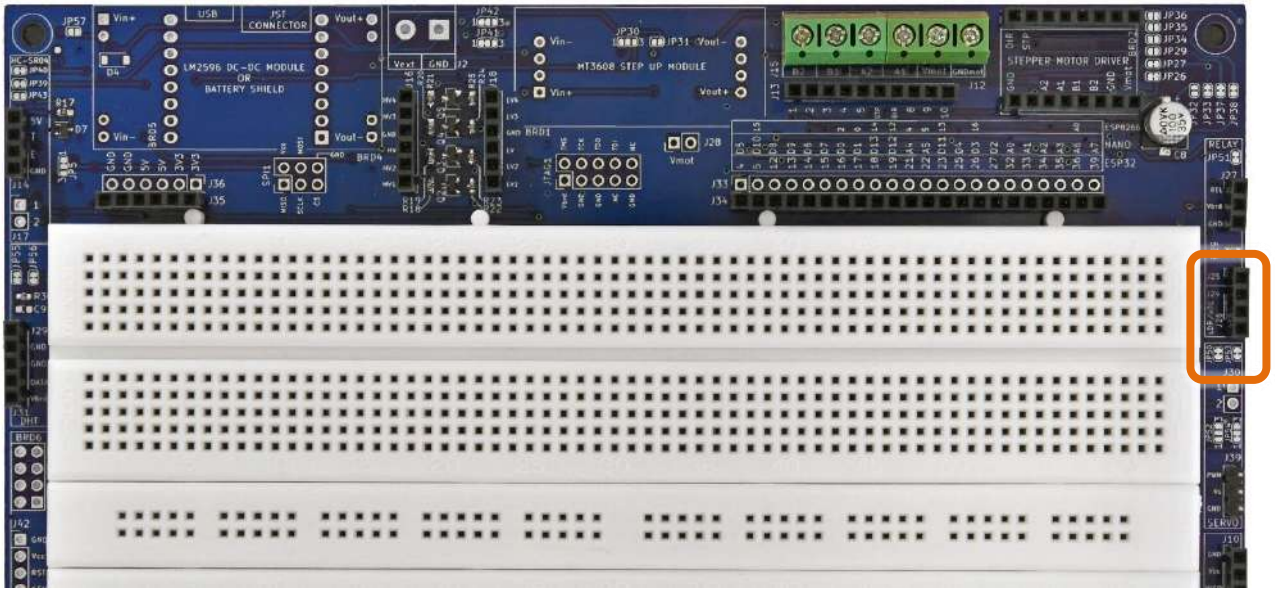


Figure 61 – LDR/xTC section of the IoT Proto Shield Plus

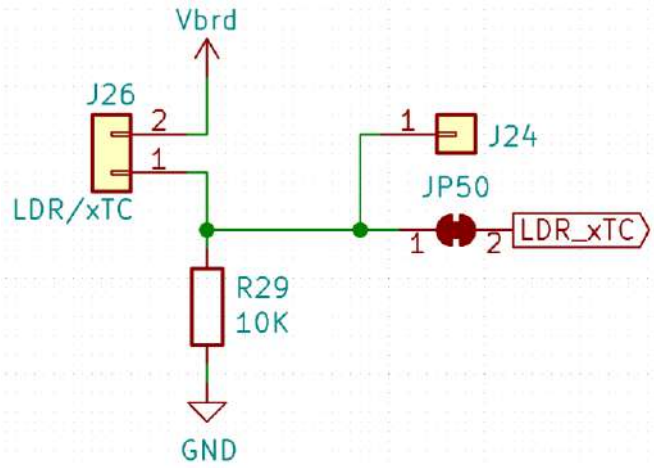


Figure 62 – Schematic of the IoT Proto Shield Plus LDR/xTC Section

25.1. Default connections

	ESP32	ARDUINO	ESP8266
LDR/xTC	GPIO34	A2	*NC*

NC = Not Connected

Please note that there are no default connections for the ESP8266. If you need to use the LDR/xTC with ESP8266 boards, you need to route the signals with jumper wires.

25.2. Suggested ESP8266 connections

The sample sketches are made to work with these connections:
ESP8622 A0 connected to header J3 ESP32 GPIO34 (LDR/xTC).
Remember that ESP8266 A0 pin connects by default to POT (see §11.1). You must open JP12 (see Figure 20) to disconnect ESP8266 A0 pin from POT and let LDR/xTC works properly.

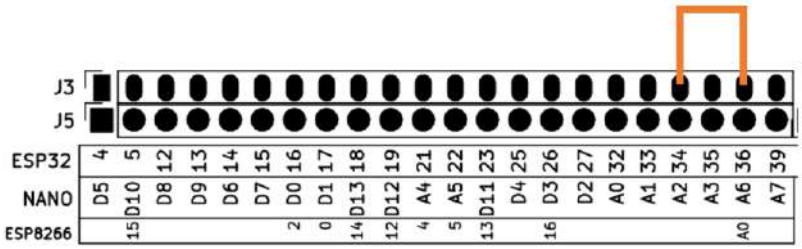


Figure 63 – Suggested connections to work with LDR/xTC and ESP8266

25.3. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_LDR_xTC`

According to the board you are using and the signals routing, the sketches read the analog value of the LDR/xTC and print it to the serial monitor.

Use the `IoTPSP_LDR_xTC_NANO` sketch to test LDR/xTC with ARDUINO NANO boards.
Use the `IoTPSP_LDR_xTC_ESP32` sketch to test LDR/xTC with ESP32 boards.
Use the `IoTPSP_LDR_xTC_ESP8266` sketch to test LDR/xTC with ESP8266 boards.

25.4. Using the LDR/xTC with a different GPIO pin

According to Figure 62, LDR/xTC is routed to the default GPIO pin by means of JP50.
To disconnect LDR/xTC from its default GPIO pin, you must OPEN JP50 (cut it in the middle).
By means of some jumper wires, connect J24 to another GPIO pin according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** to avoid conflict with other default connections.

26. Serial communication on the IoT Proto Shield Plus

Additional serial communication can be used to communicate with other devices.

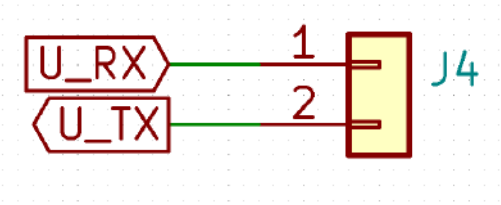


Figure 64 – Additional serial communication routing on the IoT Proto Shield Plus

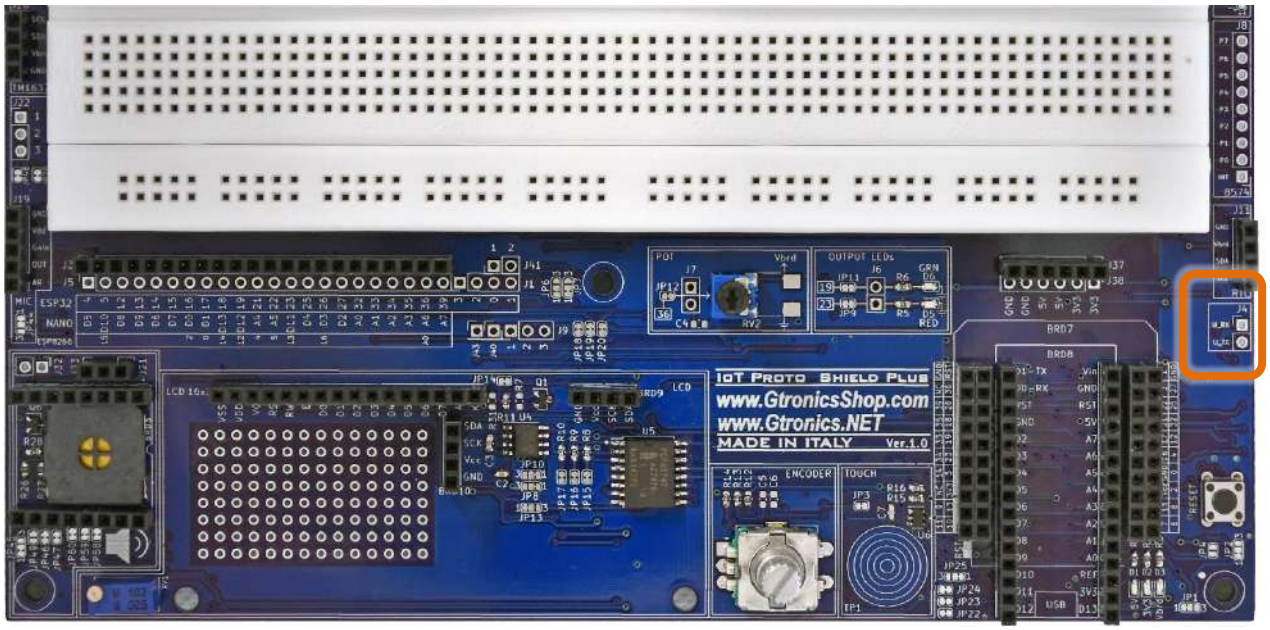


Figure 65 – J4 header on the IoT Proto Shield Plus

26.1. Default connections

Additional RX and TX signals on the **IoT Proto Shield Plus** are routed by default according to the following connections:

	ESP32	ARDUINO	ESP8266
U_RX	GPIO16	D0	GPIO2 (D4)
U_TX	GPIO17	D1	GPIO0 (D3)

On the **IoT Proto Shield Plus,** this additional serial communication is used by default to communicate with the DFPlayer Mini MP3 (see §27), or to any other device by means of J4 header (see Figure 65).

27. DFPlayer Mini MP3 Player

The **IoT Proto Shield Plus** is ready to work with 8 pins DFPlayer Mini MP3 modules of widespread use (see Figure 66).
To work with a DFPlayer Mini MP3 module, just plug the module into the BRD3 socket (see Figure 66).
You can find additional information about the module at the following link:
https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299
To work with the DFPlayer Mni MP3, you need a microSD card too (with FAT32 file system). The MP3 files need to be into a folder named “mp3”.

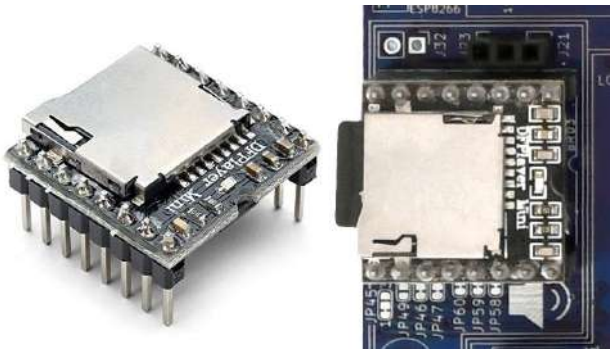


Figure 66 – A DFPlayer Mini MP3 module (on the left)
and the module plugged into BRD3 socket of the IoT Proto Shield Plus (on the right)

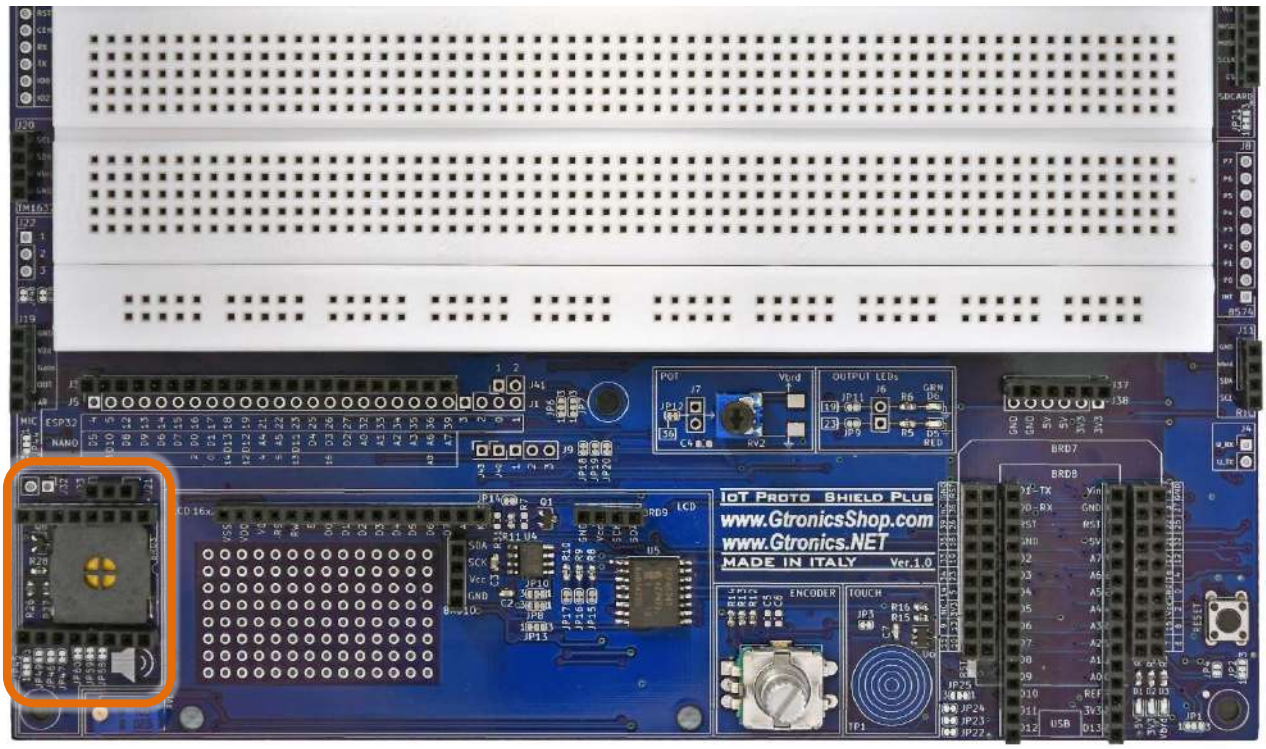


Figure 67 – BRD3 socket of the IoT Proto Shield Plus

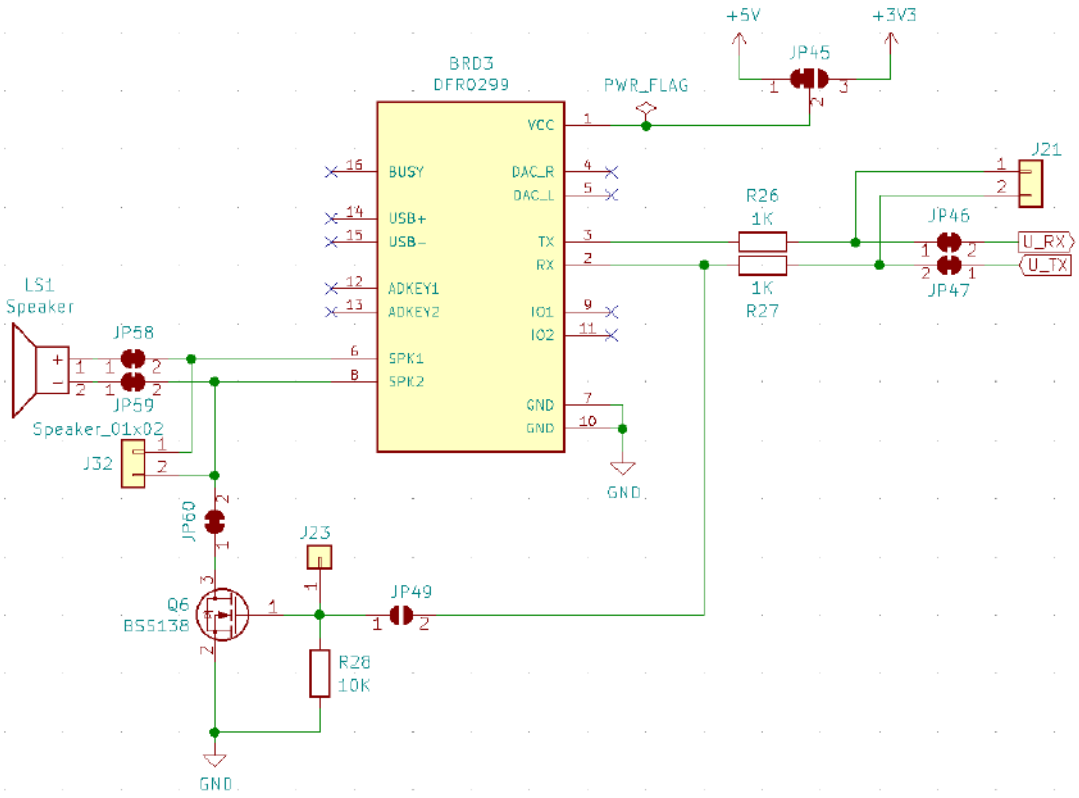


Figure 68 – Schematic of the IoT Proto Shield Plus BRD3 socket header

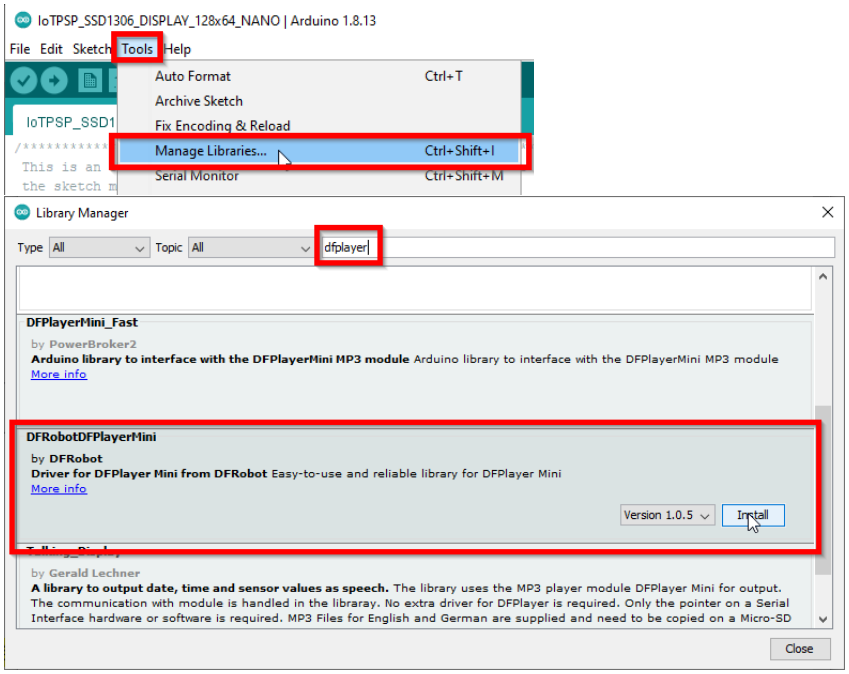
27.1. Default connections

On the **IoT Proto Shield Plus,** the communication with the DFPlayer Mini MP3 is performed with serial communication.
Refer to §26.1 to see default routing of additional serial communication signals on the **IoT Proto Shield Plus.**
The DFPlayer Mini MP3 can work @3V3 as well, but in most cases the 3V3 generated by the board in use (ESP32, NANO, etc.) cannot provide enough current to let the MP3 player works correctly.
Thus, as shown in Figure 68, *Vcc* pin of the DFPlayer Mini MP3 connects by default to 5V (which are usually supplied by the USB port of your computer connected to the board in use).

27.2. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_MP3_Player`

These sketches make use of the *DFRobotDFPlayerMini library* by DFRobot.
Be sure to install it before using the sketches.



The `IoTPSP_Sample_Sketches\IoTPSP_MP3_Player` folder contains also a folder with 3 sample MP3 files. Copy this folder on your micro-SD card to test the **IoT Proto Shield Plus** with this samples.

These sketches use by default additional serial communication routing shown in §26.1

27.2.1. IoTPSP_MP3_NANO

The `IoTPSP_MP3_NANO` uses the standard serial communication of Arduino NANO boards (routed on D0, D1) to communicate with the DFPlayer Mini MP3.
Of course, since the serial port is used to communicate with the MP3 Player it cannot be used to work with the Serial Monitor.

27.2.2. IoTPSP_MP3_NANO_EVERY_33

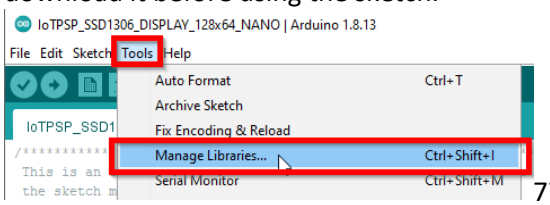
The `IoTPSP_MP3_NANO_EVERY_33` uses the secondary serial communication port on the NANO EVERY or NANO33 boards (routed on D0, D1) to communicate with the DFPlayer Mini MP3.

27.2.3. IoTPSP_MP3_ESP32

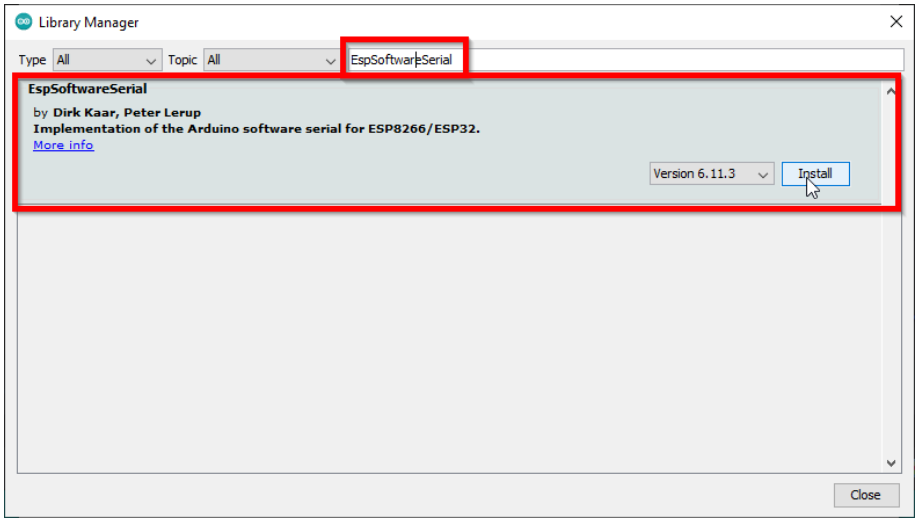
The `IoTPSP_MP3_ESP32` uses the secondary serial communication port on the ESP32 boards (routed on GPIO16, GPIO17) to communicate with the DFPlayer Mini MP3.

27.2.4. IoTPSP_MP3_ESP8266

The `IoTPSP_MP3_ESP8266_SoftwareSerial` uses the *EspSoftwareSerial library*, be sure to download it before using the sketch.



77



The `IoTPSP_MP3_ESP8266_SoftwareSerial` create an additional software serial port on GPIO0 and GPIO2 to communicate with the DFPlayer Mini MP3.

The ESP8266 has a secondary serial communication port routed on GPIO1 and GPIO3, in case you want to use this second serial port, you need to reroute the signals and experimenting by yourself.

27.3. Using the DFPlayer Mini MP3 module with a different GPIO pins

It is not recommended to use different pins to establish a serial communication with the DFPlayer Mini MP3 module.
Anyway, if you are using an "old" Arduino NANO board you may want to use the Serial Monitor, thus you need to free D0 and D1.
In this case we will use the *SoftwareSerial library* to create a "virtual" serial port.
According to Figure 68, U_RX and U_TX connects by default to TX and RX pins of the DFPlayer Mini MP3 module by means of JP46 and JP47.
You can experiment by OPENING JP46 and JP47 (cut them in the middle), rerouting RX and TX signals via J21 and then using the sketch described in § 27.4.

27.4. IoTPSP_MP3_NANO_SoftwareSerial

The `IoTPSP_MP3_NANO_SoftwareSerial` uses the *SoftwareSerial library*, included in the Arduino IDE.

To use this sketch, refer to § 27.3.
The sketches uses D10 as RX pin and D11 as TX pin.
Additional information about Software Serial can be found at this link
https://www.arduino.cc/en/Reference/softwareSerial.

28. SPEAKER

The SPEAKER can be used also as a buzzer or to create frequency modulated tones.
It does not come "out of the box" and two jumper wires are required.
To let the SPEAKER work as a tone buzzer you have to:

- Remove the DFPlayer Mini MP3 module from the BRD3 socket
- Use a jumper wire to connect BRD3 pin #1 (*Vcc*) to BRD3 pin #6 (*SPK1*)
- Use a wire to connect J21 pin #2 to J23

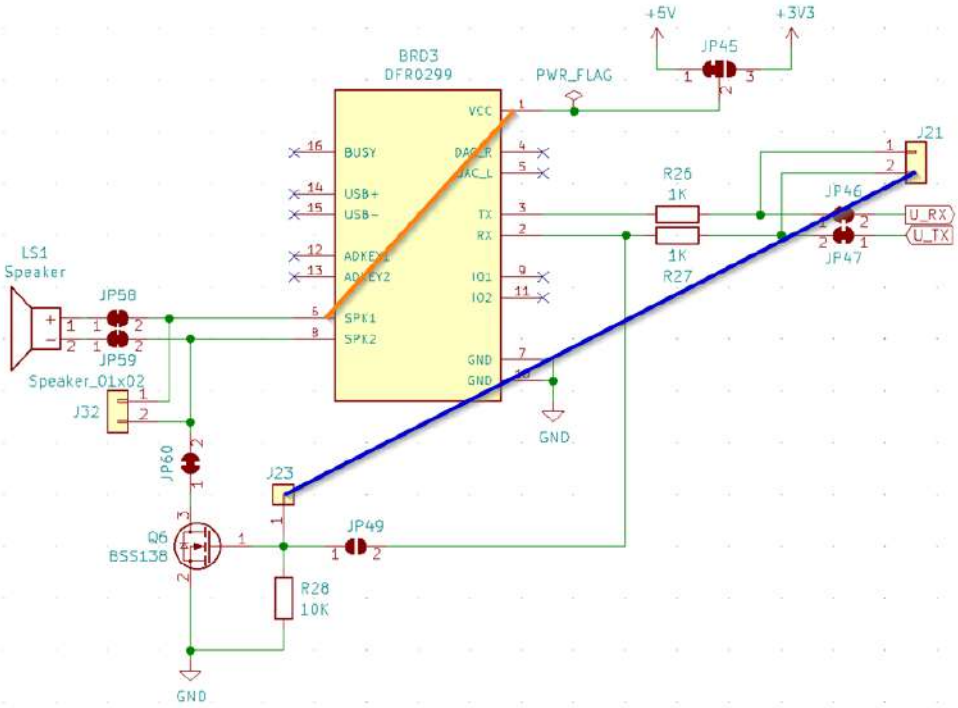


Figure 69 – Schematic of the IoT Proto Shield Plus SPEAKER buzzer configuration

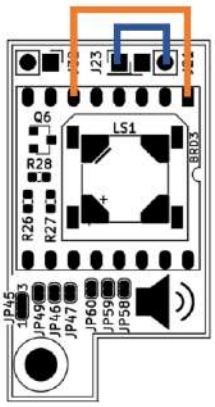


Figure 70 – Suggested wiring to work with SPEAKER as a tone BUZZER

28.1. Default connections

According to Figure 69, U_TX will drive the buzzer tone. As shown in § 26.1, U_TX connects by default to the following pins:

	ESP32	ARDUINO	ESP8266
U_TX	GPIO17	D1	GPIO0 (D3)

28.2. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_SPEAKER`

28.3. IoTPSP_SPEAKER_NANO

The `IoTPSP_SPEAKER_NANO` is based on the *tone()* function of the Arduino base library. Information about the Arduino *tone()* function can be found at this link:
https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/
The function does not work on NANO33 BLE (at least at the moment of writing this manual).

28.4. IoTPSP_SPEAKER_ESP32

The `IoTPSP_SPEAKER_ESP32` uses the *Tone32* library. The library is downloadable at https://github.com/lbernstone/Tone32. A copy of the library is also contained in the sketch folder.

28.5. IoTPSP_SPEAKER_ESP8266

The `IoTPSP_SPEAKER_ESP8266` is based on the *tone()* function of the Arduino base library. Information about the Arduino *tone()* function can be found at this link:
https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

29. MIC

The **IoT Proto Shield Plus** is ready to work with MAX9814 or MAX4466 microphone modules of widespread use.

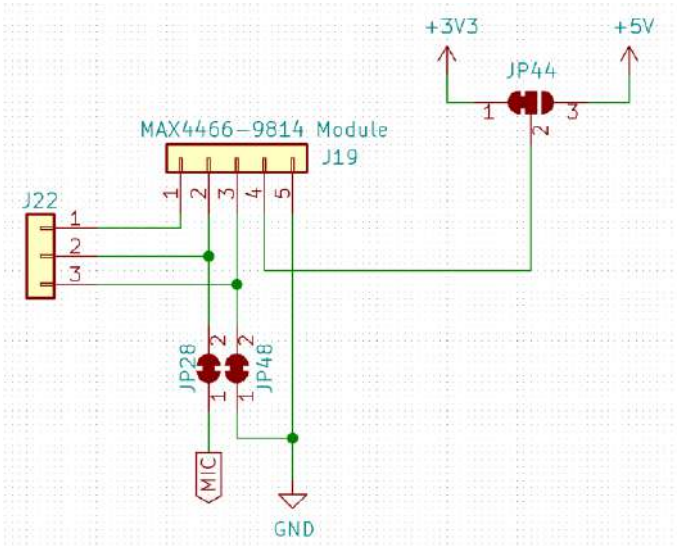


Figure 71 – Schematic of the IoT Proto Shield Plus MIC Section

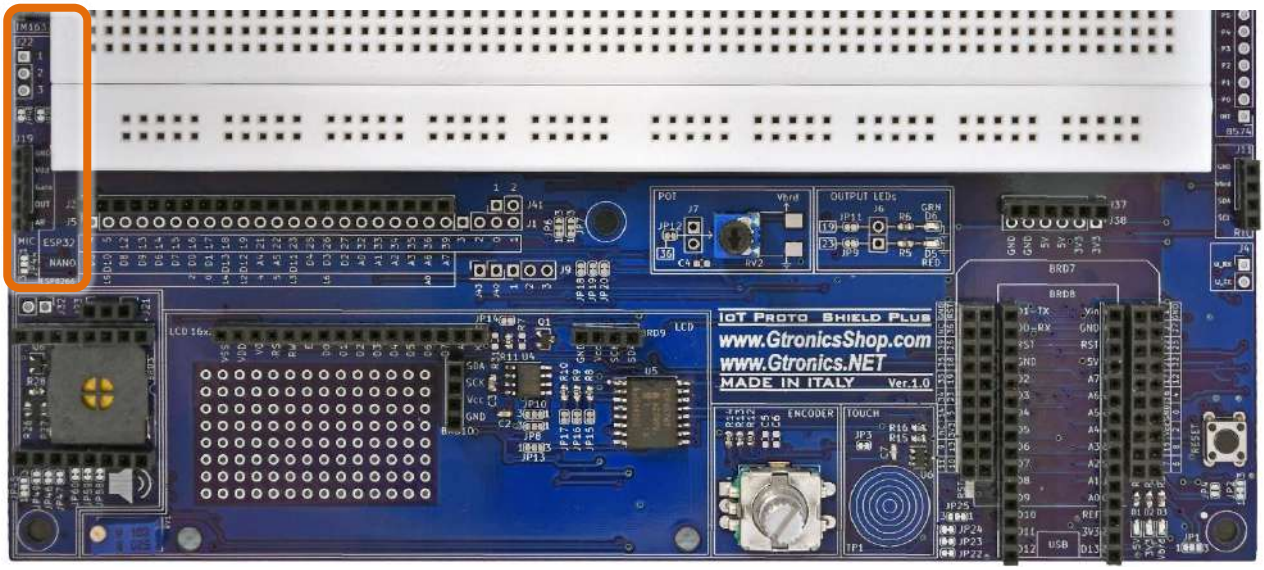


Figure 72 – MIC section of the IoT Proto Shield Plus

29.1. MAX9814 microphone module

This microphone module is a 5 pins module based on MAX9814 chip.
More details on the MAX9814 chip at this link:
https://datasheets.maximintegrated.com/en/ds/MAX9814.pdf

This module has 5 pins:
Vdd: Module power supply – 2.7-5.5 V
GND: Ground
OUT: Analog data output to the microcontroller
Gain: Adjusting maximum module output*
AR: Adjusting module accuracy*

*	Float	Vcc	GND
Gain	60dB	50dB	40dB
AR	1:4000	1:2000	1:500

The module's output is not zero in silent mode and has an offset 1.25V.

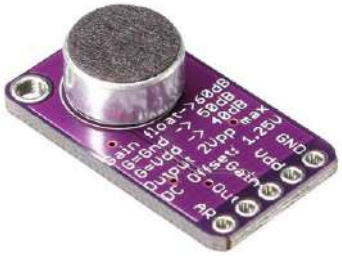


Figure 73 – A MAX9814 microphone module of widespread use

To work with a MAX 9814 microphone module just plug it into J19 as shown in the following picture.

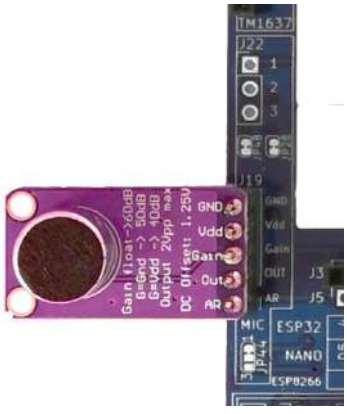


Figure 74 – A MAX9814 microphone plugged into J19

29.2. Setting the Gain

By default, Gain is set to 40dB: J19 pin #3 connects to GND by means of JP48.
Opening JP48 (cut it in the middle) left J19 pin #3 floating, thus gain is set to 60dB.
With JP48 open and J22 pin 3# connected to Vcc, gain is set to 50dB.

29.3. Setting the module accuracy

By default, AR (adjusting module accuracy) is set to 1:4000: J19 pin #1 is floating.
Connecting J22 pin #1 to Vcc sets AR to 1:2000.
Connecting J22 pin #1 to GND sets AR to 1:500.

29.4. MAX4466 microphone module

This module is a 3 pins module based on MAX4466 chip.
More details on the MAX4466 chip at this link:
https://www.maximintegrated.com/en/products/analog/audio/MAX4466.html

Figure 75 – A MAX4466 microphone module of widespread use

To work with a MAX4466 microphone module plug it into J19 as shown in the following picture.
Be sure that JP48 is closed: JP19 pin #3 must connect to GND.

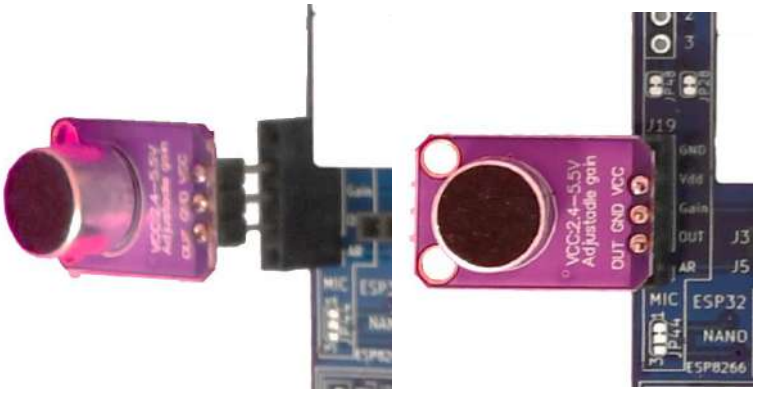


Figure 76 – A MAX4466 microphone plugged into J19

29.5. Default connections

	ESP32	ARDUINO	ESP8266
MIC	GPI39	A7	*NC*

NC = Not Connected

Please note that there are no default connections for the ESP8266. If you need to use the MIC with ESP8266 boards, you need to route the signals with jumper wires.

29.6. Suggested ESP8266 connections

The sample sketches are made to work with these connections:
ESP8622 A0 connected to header J3 ESP32 GPIO39 (MIC).
Remember that ESP8266 pin A0 connects by default to POT (see §11.1). You must OPEN JP12 (see Figure 20) to disconnect ESP8266 pin A0 from POT and let the MIC works properly.

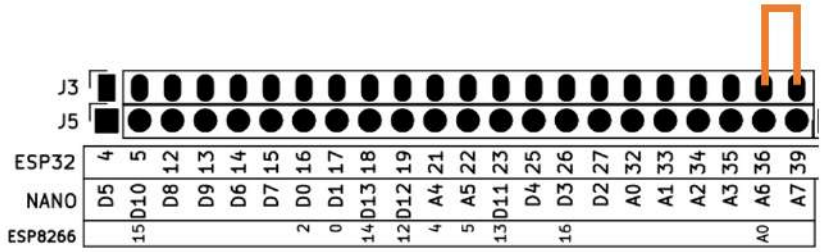


Figure 77 – Suggested connections to work with MIC and ESP8266

29.7. Sample Sketches

Location: IoTPSP_Sample_Sketches\IoTPSP_MIC

According to the board you are using and the signals routing, the sketches read the analog value of the MIC and print it to the serial monitor.

Use the IoTPSP_MIC_NANO sketch to test the MIC with ARDUINO NANO boards.
Use the IoTPSP_MIC_ESP32 sketch to test the MIC with ESP32 boards.
Use the IoTPSP_MIC_ESP8266 sketch to test the MIC with ESP8266 boards.

29.8. Using the MIC with a different GPIO pin

According to Figure 71, MIC connects to the default GPIO pin by means of JP28.
To disconnect MIC from its default GPIO pin, you must OPEN JP28 (cut it in the middle).
By means of some jumper wires, connect J22 pin #2 to another GPIO pin according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** to avoid conflict with other default connections.

30. SERVO

The **IoT Proto Shield Plus** is ready to work with a standard **5V** hobby servo of widespread use. To connect a servo, you need to solder a standard 2,54mm 3 pins single row male header (not provided with the board) to J39.

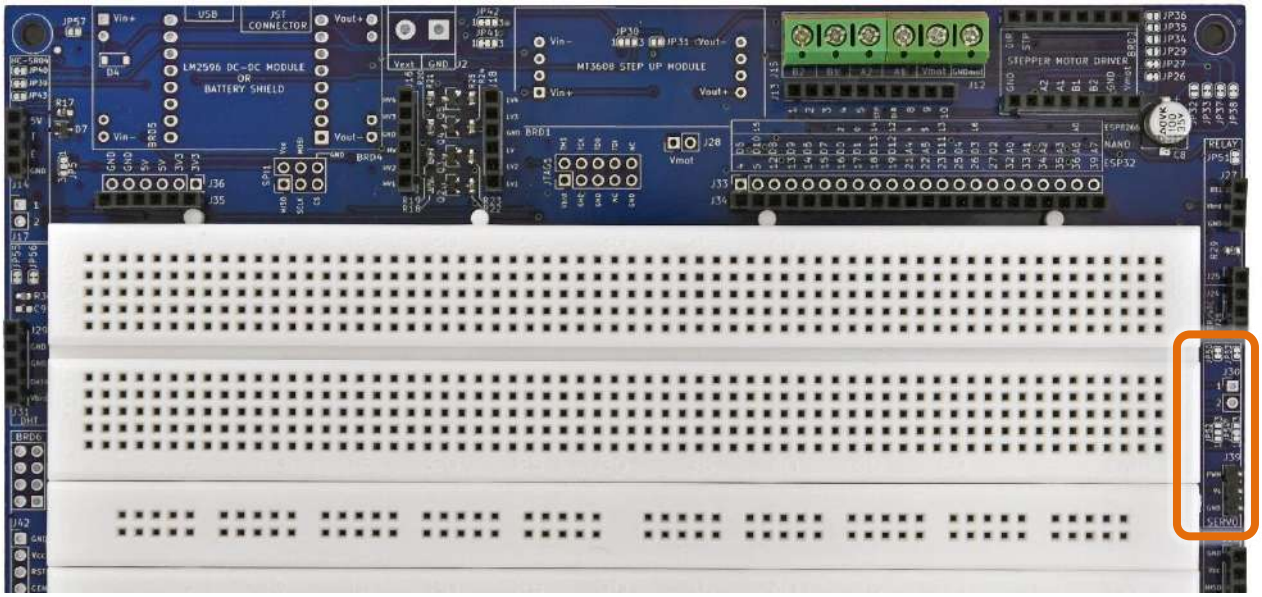


Figure 78 – SERVO section of the IoT Proto Shield Plus

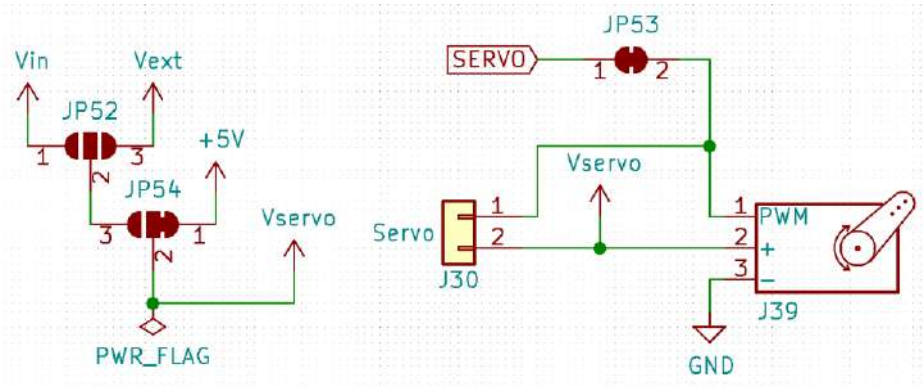


Figure 79 – Schematic of the IoT Proto Shield Plus SERVO Section

30.1. Default connections

	ESP32	ARDUINO	ESP8266
SERVO	GPIO04	D5	*NC*

NC = Not Connected

Servo Power Supply connects by default to 5V.

Please note that there are no default connections for the ESP8266. If you need to use the SERVO with ESP8266 boards, you need to route the signals with jumper wires.

30.2. Suggested ESP8266 connections

The sample sketch is made to work with this connection:
ESP8622 GPIO2 (D4) connected to header J3 ESP32 GPIO04 (SERVO).

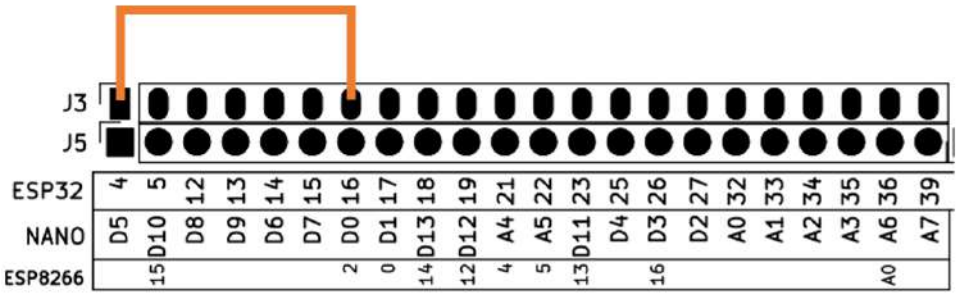


Figure 80 – Suggested connections to work with SERVO and ESP8266

30.3. Sample Sketches

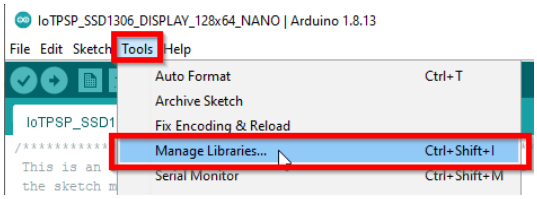
Location: `IoTPSP_Sample_Sketches\IoTPSP_SERVO`

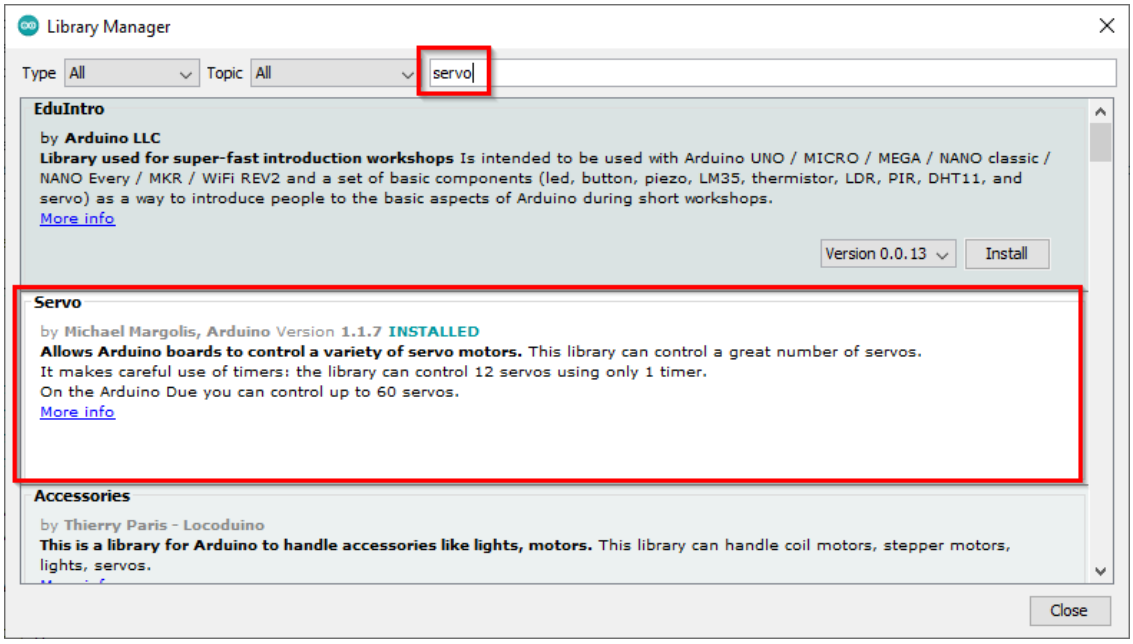
These sketches use the POT to move the servo position.
Rotate the POT knob to change the angular servo position.

See §11 for more details about POT on the **IoT Proto Shield Plus.**

30.4. IoTPSP_SERVO_NANO

The `IoTPSP_SERVO_NANO` sketch, uses the *Servo* library.
Be sure to install it before using the sketch.
Be sure to use the latest version, if you are using NANO 33 boards or EVERY (old releases of the library will not work).
More details here https://www.arduino.cc/reference/en/libraries/servo/

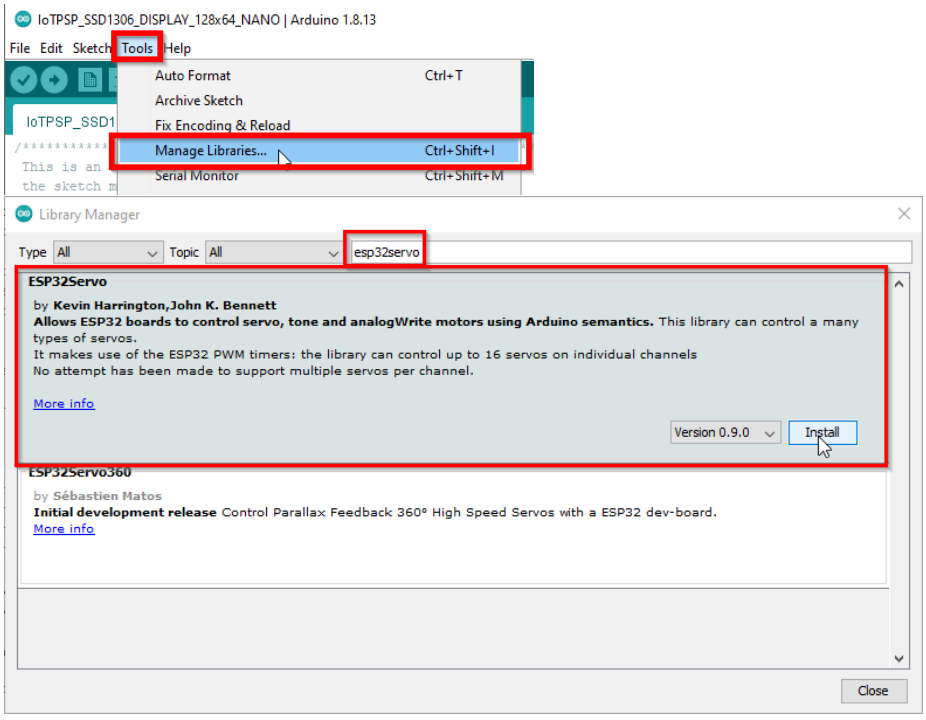




Use the `IoTPSP_SERVO_NANO` sketch to test the SERVO with ARDUINO NANO boards.

30.5. IoTPSP_SERVO_ESP32

The `IoTPSP_SERVO_ESP32` sketch, uses the *Esp32Servo* library.
Be sure to install it before using the sketch.
More details here https://www.arduino.cc/reference/en/libraries/esp32servo/



Use the `IoTPSP_SERVO_ESP32` sketch to test the SERVO with ESP32 boards.

30.6. IoTPSP_SERVO_ESP8266

The `IoTPSP_SERVO_ESP8266` sketch, uses the *Servo* library (see § 30.4).
Be sure to install it before using the sketch.
Servo signal must be routed as shown in Figure 80.

Use the `IoTPSP_SERVO_ESP8266` sketch to test the SERVO with ESP8266 boards.

30.7. Using SERVO with a different GPIO pin

According to Figure 79, SERVO connects to the default GPIO pin by means of JP53.
To disconnect SERVO from its default GPIO pin, you must OPEN JP53 (cut it in the middle).
By means of a jumper wire, connect J30 pin #1 to another GPIO pin according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** to avoid conflict with other default connections.

30.8. Powering the SERVO with a different power supply

According to Figure 79, *Vservo* connects by default to 5V by means of JP54.
First thing first you must disconnect *Vservo* from 5V (cut JP54 between pads 1 and 2).
Be sure that JP54 pads 1-2 connection is OPEN to avoid damage on the **IoT Proto Shield Plus**.
At this point you can reroute *Vservo* according to JP52 and JP54 shown in Figure 79, or you can use J30 pin #2 to connect an external power supply (in case do not forget that GND must be in common).

31. HC-SR04 (distance sensor)

The **IoT Proto Shield Plus** is ready to work with 4 pins HC-SR04 distance sensor modules of widespread use (see Figure 81).
The power supply of the HC-SR04 must be 5V.
The TRIGGER signal (HC_SR_T) works @3V3 as well, so it doesn't matter if you are working with a 5V board (i.e. NANO or EVERY).
Since ECHO is a 5V signal, the **IoT Proto Shield Plus** provides a clipping circuit on this signal in order to let the HC-SR04 works with 3V3 tolerant pin as well (i.e. ESP32, NANo33, etc.)
To work with a HC-SR04 module, just plug the module into the J14 header.

Figure 81 – HC-SR04 module of widespread use

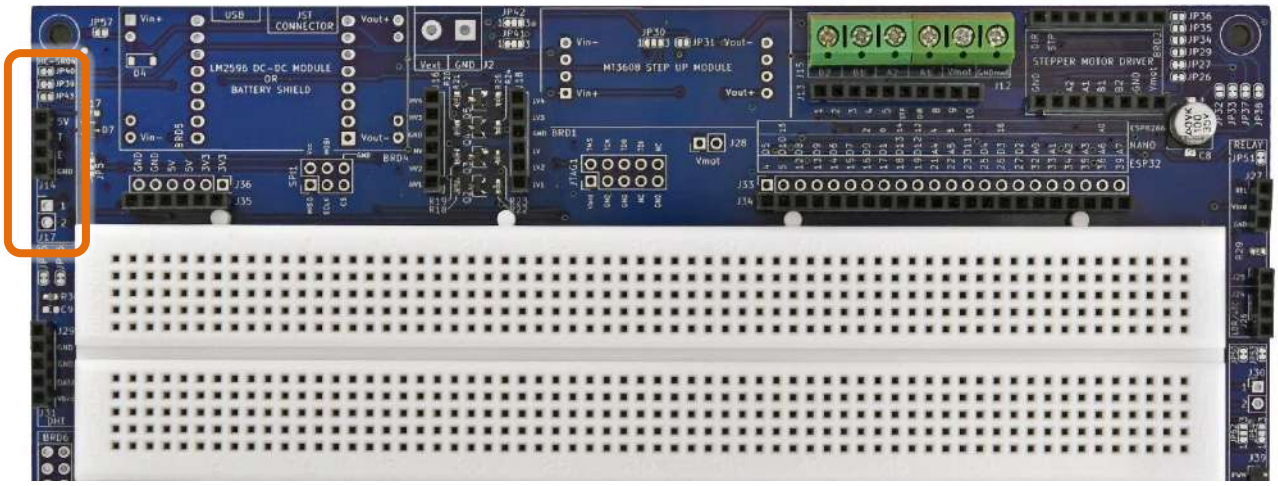


Figure 82 – HC-SR04 section of the IoT Proto Shield Plus

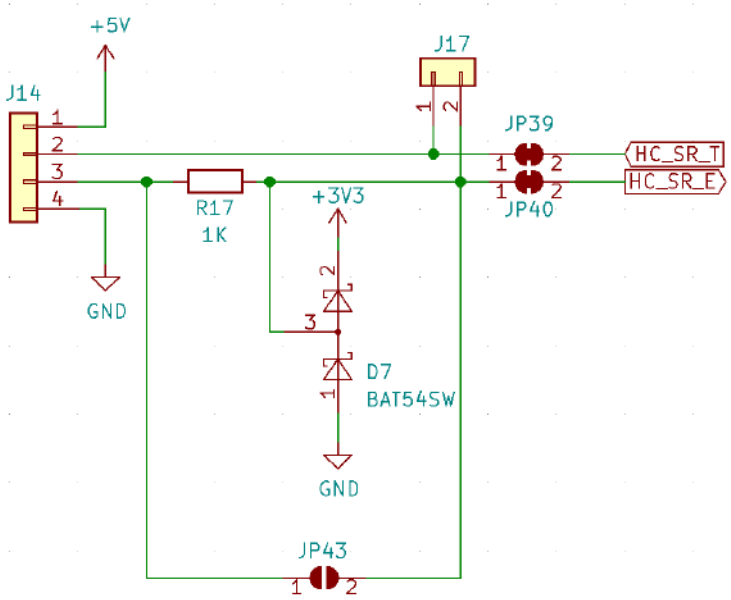


Figure 83 – Schematic of the IoT Proto Shield Plus HC-SR04 Section

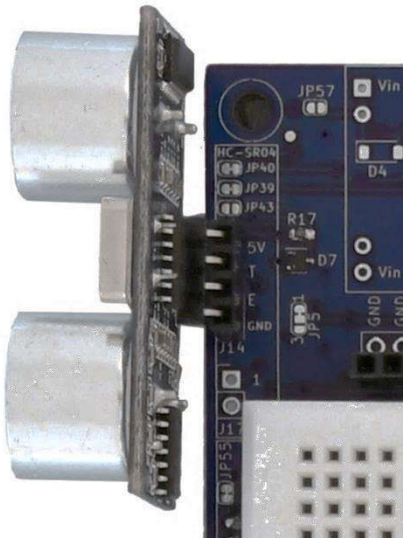


Figure 84 – An HC-SR04 module plugged into J14 header of the IoT Proto Shield Plus

31.1. Default connections

	ESP32	ARDUINO	ESP8266
HC_SR_T	GPIO14	D6	*NC*
HC_SR_E	GPIO15	D7	*NC*

NC = Not Connected

Please note that there are no default connections for the ESP8266. If you need to use the HC-SR04 with ESP8266 boards, you need to route the signals with jumper wires.

31.2. Suggested ESP8266 connections

The sample sketch is made to work with this connection:
ESP8622 GPIO 4 (D2) connected to header J3 ESP32 GPIO14 (HC_SR_T).
ESP8622 GPIO 5 (D1) connected to header J3 ESP32 GPIO15 (HC_SR_E).

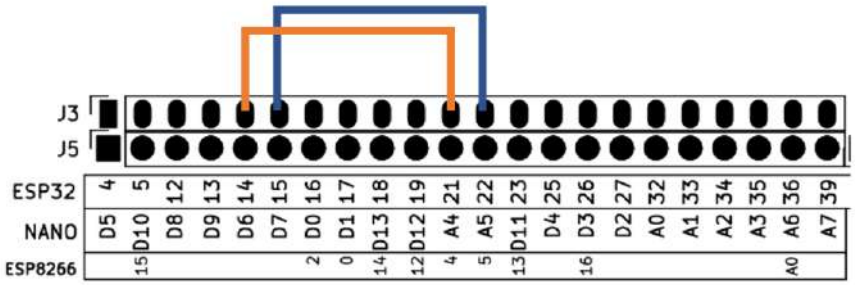


Figure 85 – Suggested connections to work with HC-SR04 and ESP8266

31.3. Sample Sketches

Location: `IoTPSP_Sample_Sketches\IoTPSP_HC_SR04`

Use the `IoTPSP_HC_SR04_NANO` sketch to test the HC-SR04 with ARDUINO NANO boards.
Use the `IoTPSP_HC_SR04_ESP32` sketch to test the HC-SR04 with ESP32 boards.
Use the `IoTPSP_HC_SR04_ESP8266` sketch to test the HC-SR04 with ESP8266 boards.

Open the serial monitor to see the distance measured by the SR-HC04 sensor.

More details at this link: https://create.arduino.cc/projecthub/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-036380

31.4. Using HC-SR04 with different GPIO pins

According to Figure 83, HC_ST_T and HC_SR_E connect to their default GPIO pins by means of JP39 and JP40.
To disconnect HC_ST_T and HC_SR_E from their default GPIO pins, you must OPEN JP39 and JP40 (cut them in the middle).
Solder a 2 pins header on J17.
By means of two jumper wires, connect J17 to other GPIO pins according to your needs.
Please check the default signal routing of the **IoT Proto Shield Plus** to avoid conflict with other default connections.

32. STEPPER MOTOR

The **IoT Proto Shield Plus** is ready to work with DRV8825 form factor stepper driver modules.
These modules are used to drive 2 phases 4 wires (A, /A, B, /B) stepper motors.
The wiring diagram slightly changes according to the driver you are using (see further in this manual), anyway there are some common minimum wiring requirements whose apply to all drivers:

- Motor power supply
- Logic power supply
- Sleep and (and Reset) wiring
- Step signal wiring
- Direction signal wiring

According to Figure 86 and §32.8, the **IoT Proto Shield Plus** is ready to work with a DRV8825 without any additional connection (or disconnection).
Consider that *Vmot* is disconnected by default. You have to provide the right motor power supply voltage by means of J12 or according to signal routing shown in Figure 87.

Please read carefully this entire chapter (§32) before using a stepper motor: you must consider all the aspects and all the options provided by the IoT Proto Shield Plus, to correctly drive a stepper motor.

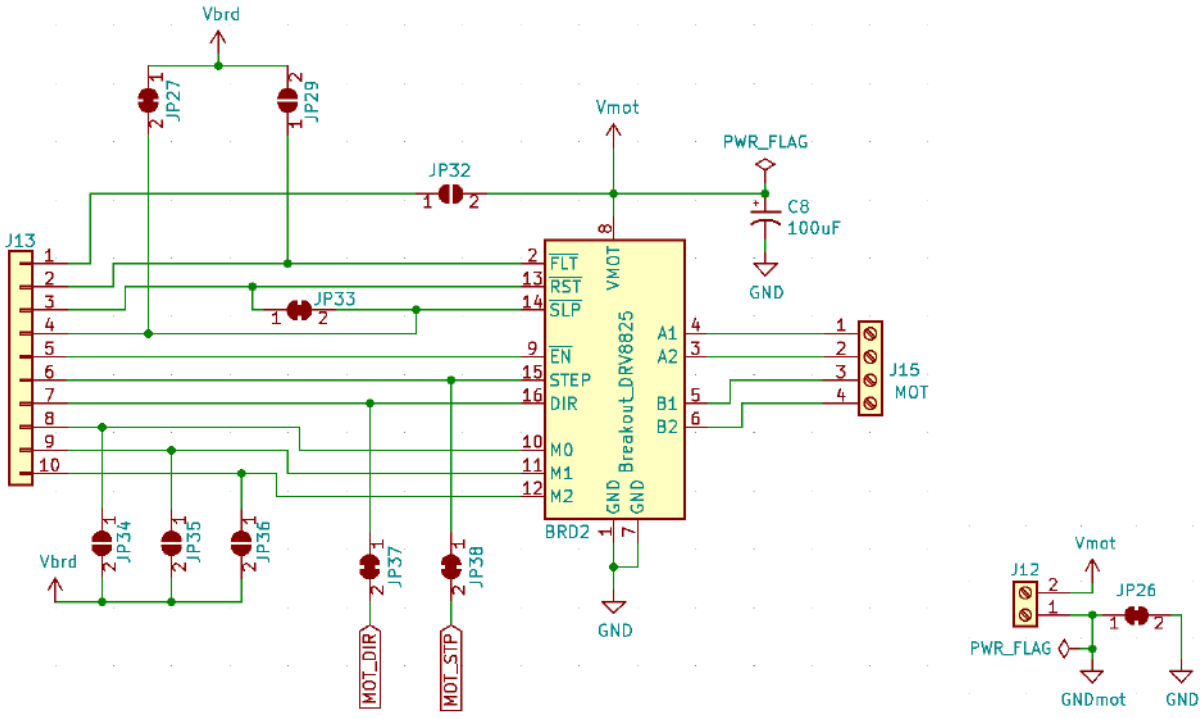


Figure 86 – Schematic of the IoT Proto Shield Plus Motor Stepper Section

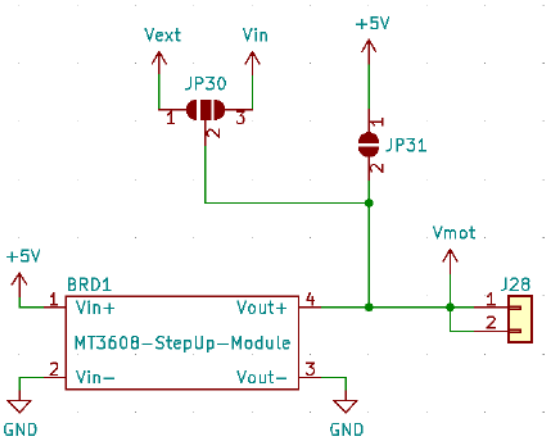


Figure 87 – Schematic of the Vmot routing section of the IoT Proto Shield Plus

BRD1 (MT3608 step-up-Module) is a predisposition for possible future features, do not consider it now.

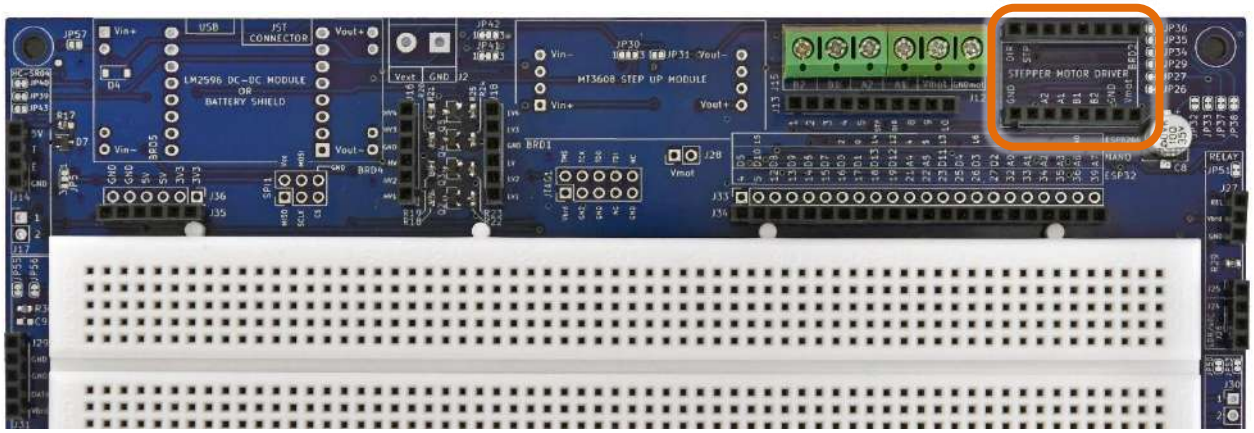


Figure 88 – BRD2 socket of the IoT Proto Shield Plus

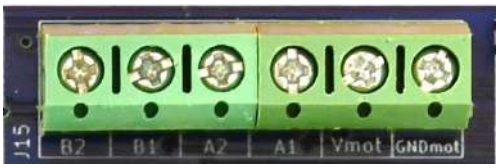


Figure 89 – J15 screw connector of the IoT Proto Shield Plus

32.1. Default connections

	ESP32	ARDUINO	ESP8266
MOT_DIR	GPIO25	D4	*NC*
MOT_STP	GPIO26	D3	GPIO16 (D0)

NC = Not Connected

Please note that there are no default connections for the MOT_DIR signal when working with ESP8266. If you need to use a stepper motor with ESP8266 boards, you need to route MOT_DIR signal with a jumper wire.

32.2. Suggested ESP8266 connections

The sample sketch is made to work with this connection:
ESP8622 GPIO14 (D5) connected to header J3 ESP32 GPIO25 (MOT_DIR).

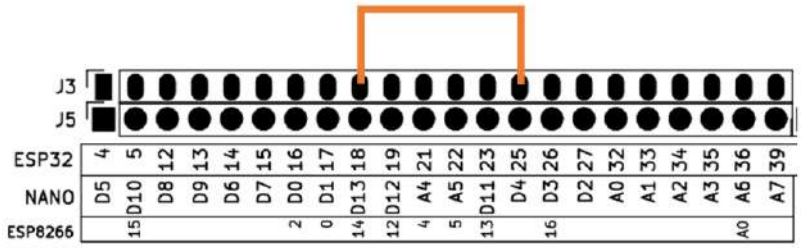


Figure 90 – Suggested connections to work with a stepper motor and ESP8266

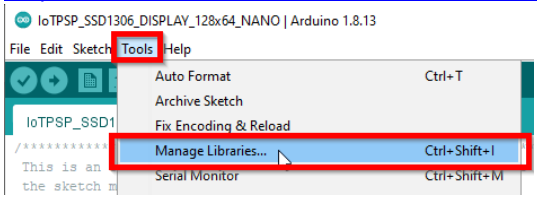
32.3. Sample Sketches

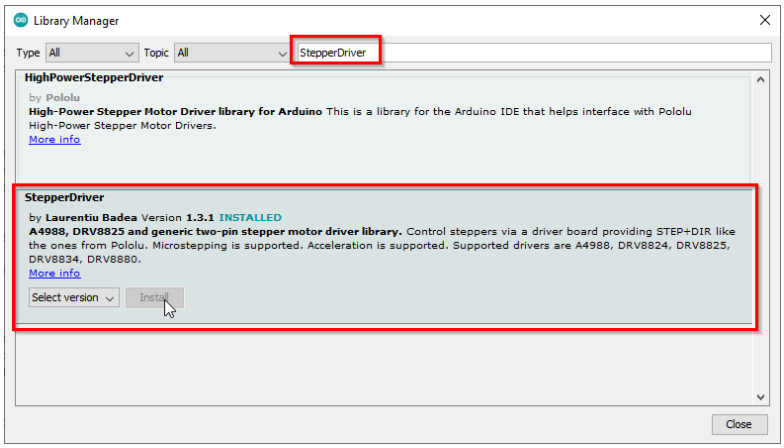
Location: `IoTPSP_Sample_Sketches\IoTPSP_STEPPER_MOTOR`

According to step per revolution of your motor, these sketches make rotating the motor clockwise and counterclockwise in a loop.

32.4. IoTPSP_STEPPER_MOTOR_NANO

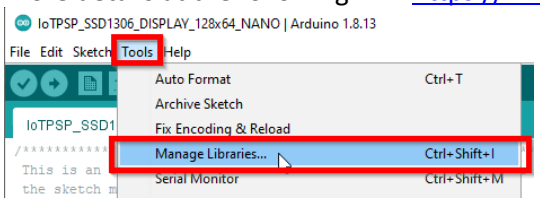
The `IoTPSP_STEPPER_MOTOR_NANO` is based on the *StepperDriver* library.
Be sure to install it before using the sketch.
Information about the Arduino *StepperDriver* can be found at this link:
https://www.arduino.cc/reference/en/libraries/stepperdriver/

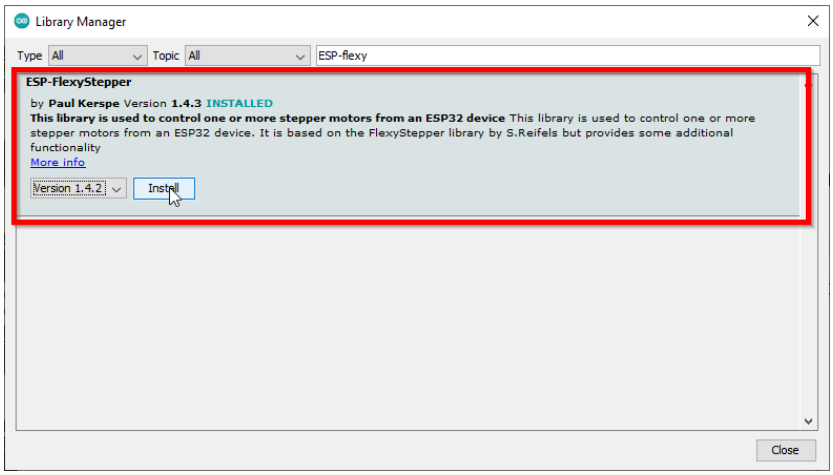




32.5. IoTPSP_STEPPER_MOTOR_ESP32

The `IoTPSP_STEPPER_MOTOR_ESP32` uses the *ESP-FlexyStepper* library.
Be sure to install it before using the sketch.
More details at the following link: https://www.arduino.cc/reference/en/libraries/esp-flexystepper/





32.6. IoTPSP_STEPPER_MOTOR_ESP8266

The `IoTPSP_STEPPER_MOTOR_ESP8266` is based on the "Connecting all control pins from DRV8825 to NodeMCU" from the link below
https://hackaday.io/project/160569-nodemcu-and-drv8825

32.7. Choosing the Stepper Driver Module

As mentioned above in this manual, there are different motor drivers, they are slightly different and each one has its pros and cons.
Regardless of the motor driver model, the first thing to do when working with these drivers, is to set current per phase according to the motor you are using.
A detailed explanation is shown at this link: https://www.youtube.com/watch?v=89BHS9hfSUk

32.8. DRV8825 module

This module is based on the DRV8825 Stepper Motor Controller IC.
It is recommended to read the datasheet (https://www.ti.com/lit/ds/symlink/drv8825.pdf) before using this module.
A more detailed explanation of this module (and in general DRV8825 based modules) can be found here: https://www.pololu.com/product/2133

- DRV8825 module PROS
  - Reasonably cheap
  - Of widespread use
  - Easy to find on the market
  - The IoT Proto Shield Plus is ready to work with this module without changing any JP configuration (see Figure 86 and Figure 91)
- DRV8825 module CONS
  - It needs 8.2-45V motor power supply, which means that you will need an external power supply to let the motor work

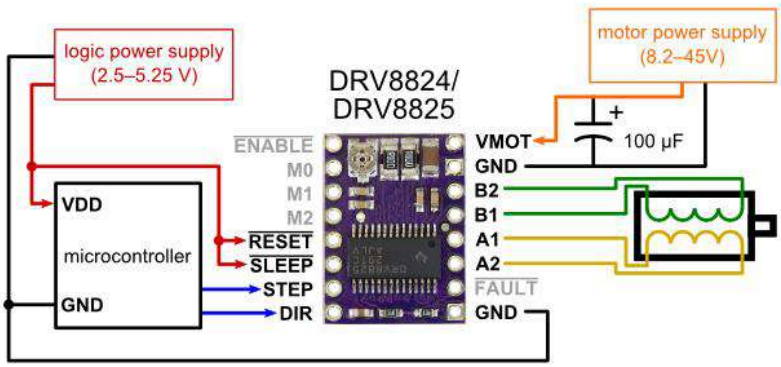


Figure 91 – Minimal wiring diagram drive a stepper motor with a DRV8825 module (full-step mode)

To work with a DRV8825 module on the IoT Proto Shield Plus:

- According to Figure 95 and Figure 86, JP27 and JP33 must be CLOSED (default configuration), in order to connect RESET and SLEEP pins of the DRV8834 module to Vbrd of the IoT Proto Shield Plus
- plug it into BRD2 socket according to the pinout (see Figure 92)
- connect the motor A, /A, B, /B according to motor wiring and Figure 89
- connect motor power supply according to motor specs and Figure 89

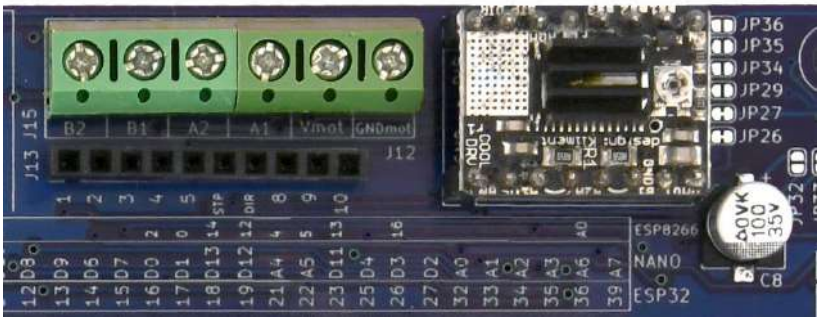


Figure 92 – A DRV8825 module plugged into BRD2 socket

32.9. TMC2xxx module

This module is based on the TMC21xx to TMC26xx Stepper Motor Controller IC.
It is recommended to read the explanation and datasheet (https://www.trinamic.com) before using this module.

- TMC2xxx module PROS
  - Noiseless current control
  - Becoming of widespread use (at the moment while writing this manual)
  - Easy to find on the market
  - A minimal modification of the **IoT Proto Shield Plus** configuration is required to work with this module (see below)
  - It works with 3V to 5V logic power supply
  - It works with 5V to 36V motor power supply (which means that with current per phase lower than 0.5A you can use the 5V provided by the USB of your PC to let the motor work)
- TMC2xxx module CONS
  - A little more expansive compared to a DRV8825

Considering the above-mentioned PROS and CONS, the modules based on TMC chips are the more versatile choice to experiment with stepper motor driver.

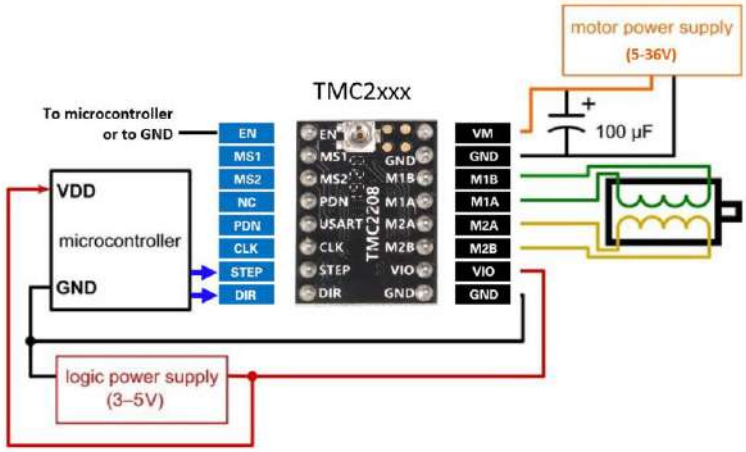


Figure 93 – Minimal wiring diagram drive a stepper motor with a TMC2xxx module (full-step mode)

To work with an TMC2xxx module on the IoT Proto Shield Plus:

- according to Figure 95 and Figure 86, JP29 must be CLOSED in order to connect *VIO* pin of the TMC2xxx module to Vbrd
- plug the TMC2xxx module into BRD2 socket according to the pinout (see Figure 94)
- connect the motor A, /A, B, /B according to motor wiring and Figure 89
- according to the motor specs, Figure 87 and Figure 89, provide the required motor power supply
    - if you want to use 5V coming from the USB of your PC (or laptop) you need to CLOSE JP31 (with a drop of tin)
- this module requires EN pin to be LOW to enable the motion (connect J13 Pin#5 to GND)

Figure 94 – A TMC2xxx module plugged into BRD2 socket

32.10. DRV8834 module

This module is based on the DRV8834 Stepper Motor Controller IC.
It is recommended to read the datasheet (https://www.ti.com/lit/ds/symlink/drv8834.pdf) before using this module.
A more detailed explanation of this module (and in general DRV8834 based modules) can be found here: https://www.pololu.com/product/2134

- DRV8834 module PROS
    - Works with a motor power supply range between 2.5 and 10.8V (which means that with current per phase lower than 0.5A you can use the 5V provided by the USB of your PC to let the motor work)
    - A minimal modification of the IoT Proto Shield Plus configuration is required to work with this module (see below)
- DRV8834 module CONS
    - Quite expensive if compared with other modules
    - More difficult to find on the market
    - Not suitable to drive motors whose require a voltage greater than 10.2V (anyway a NEMA17 o similar works fine)

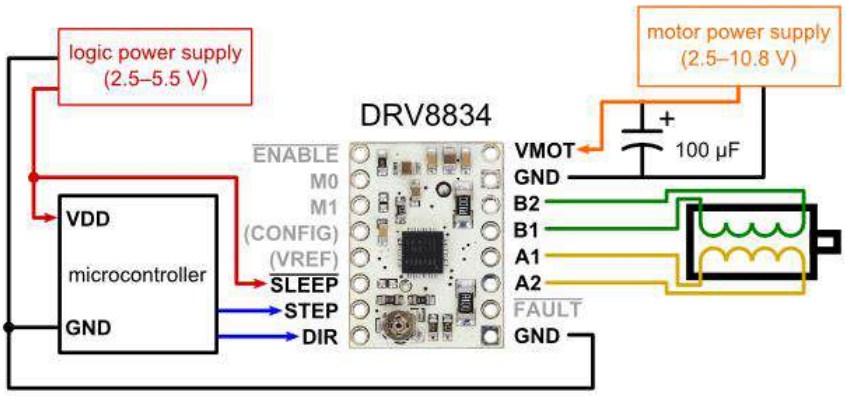


Figure 95 – Minimal wiring diagram drive a stepper motor with a DRV8834 module (full-step mode)

To work with a DRV8834 module on the **IoT Proto Shield Plus**:

- according to Figure 95 and Figure 86 JP27 must be CLOSED (default configuration), in order to connect SLEEP pin of the DRV8834 module to Vbrd of the **IoT Proto Shield Plus**
- according to Figure 95 and Figure 86 JP33 must be OPEN, in order to disconnect *VREF* pin of the DRV8834 module from Vbrd of the **IoT Proto Shield Plus**
- plug the DRV8834 module into BRD2 socket according to the pinout (see Figure 96)
- connect the motor A, /A, B, /B according to motor wiring and Figure 89
- according to the motor specs, Figure 87 and Figure 89, provide the required motor power supply
    - if you want to use 5V coming from the USB of your PC you need to CLOSE JP31 (with a drop of tin)

Figure 96 – A DRV8834 module plugged into BRD2 socket

32.11. A4988 module

This module is based on the A4988 Stepper Motor Controller IC.
It is recommended to read the datasheet (https://www.pololu.com/file/0J450/A4988.pdf) before using this module.
A more detailed explanation of this module (and in general A4988 based modules) can be found here: https://www.pololu.com/product/1182

- A4988 module PROS
    - Reasonably cheap
    - Of widespread use

- Easy to find on the market
- A minimal modification of the **IoT Proto Shield Plus** configuration is required to work with this module (see below)

- A4988 module CONS
  - It needs 8-35V motor power supply, which means that you will need an external power supply to let the motor work

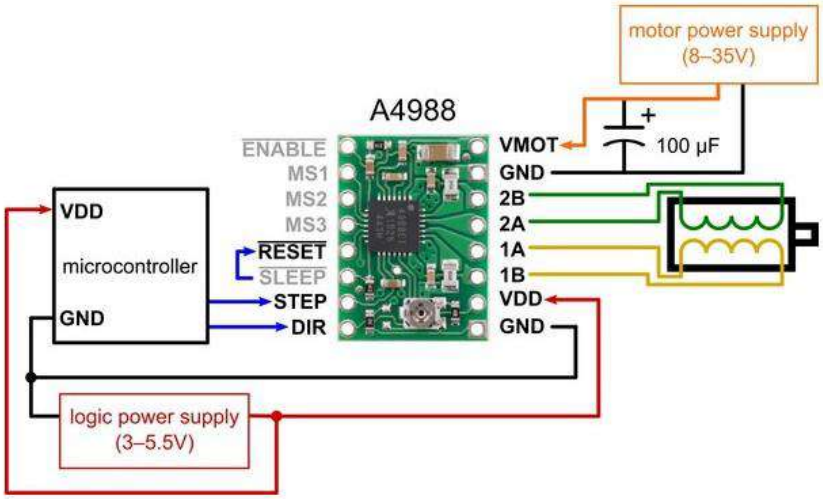


Figure 97 – Minimal wiring diagram drive a stepper motor with a A4988 module (full-step mode)

To work with an A4988 module on the **IoT Proto Shield Plus**:

- according to Figure 95 and Figure 86, JP27 must be OPEN in order to disconnect RESET and SLEEP pins of the A4988 module from Vbrd of the **IoT Proto Shield Plus**
- according to Figure 95 and Figure 86, JP33 must be CLOSED in order to connect RESET and SLEEP pin of the A4988 module
- according to Figure 95 and Figure 86, JP29 must be CLOSED in order to connect VDD pin of the A4988 module to Vbrd
- Plug the A4988 module into BRD2 socket according to the pinout (see Figure 98)
- connect the motor A, /A, B, /B according to motor wiring and Figure 89
- according to the motor specs, Figure 87 and Figure 89, provide the required motor power supply

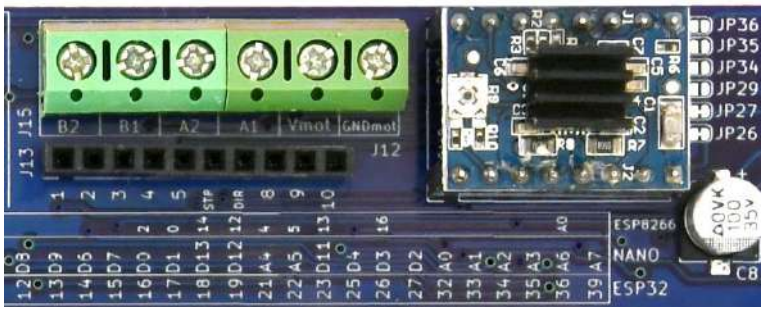


Figure 98 – An A4988 module plugged into BRD2 socket

32.12. Choosing and connecting the motor

Starting from your application you will choose a motor that better fits your needs.
According to the motor specifications, you can choose the stepper driver module that better fits your application.

Following are just two examples of different stepper motors for your reference:

- NEMA17 motor of widespread use (see Figure 99)
- micro motor (see Figure 100).

Figure 99 – NEMA 17 stepper motor of widespread use

Figure 100 – Micro stepper motor

32.13. Using STEPPER MOTOR with different GPIO pins or use additional features

According to Figure 86, MOT_DIR and MOT_STP connect to their default GPIO pins by means of JP37 and JP38.
To disconnect MOT_DIR and MOT_STP from their default GPIO pins, you must OPEN JP37 and JP38 (cut them in the middle).
By means of two jumper wires, connect J13 pin #6 and pin #7 to other GPIO pins according to your needs.
Looking at Figure 91, Figure 95 and Figure 97 you can see that some pins (such as the ones used to set microsteps) are not connected by default, you can access to the functions provided by those pins by means of J13 referring to the schematic sown in Figure 86.
Please check the default signal routing of the IoT Proto Shield Plus to avoid conflict with other default connections.

33. LEVEL SHIFTER

Since ESP NANO33 boards works with 3V3 operating voltage (see §4), you may need to convert 5V to 3V3 or vice versa.
You can do this level shifting using the 4 onboard level shifter provided on the **IoT Proto Shield Plus**.
The onboard level shifter is a copy of the level shifter module of widespread use.
The level shifter provides HV (HighVoltage) and LV (LowVoltage) connections (see Figure 102).
The level shifter converts an HV voltage signal into LVin voltage signal and, vice versa, it converts a LV voltage signal into an HVin voltage signal.
By default, HVin connects to 5V (by means of JP41) and LVin connects to 3V3 (by means of JP42).

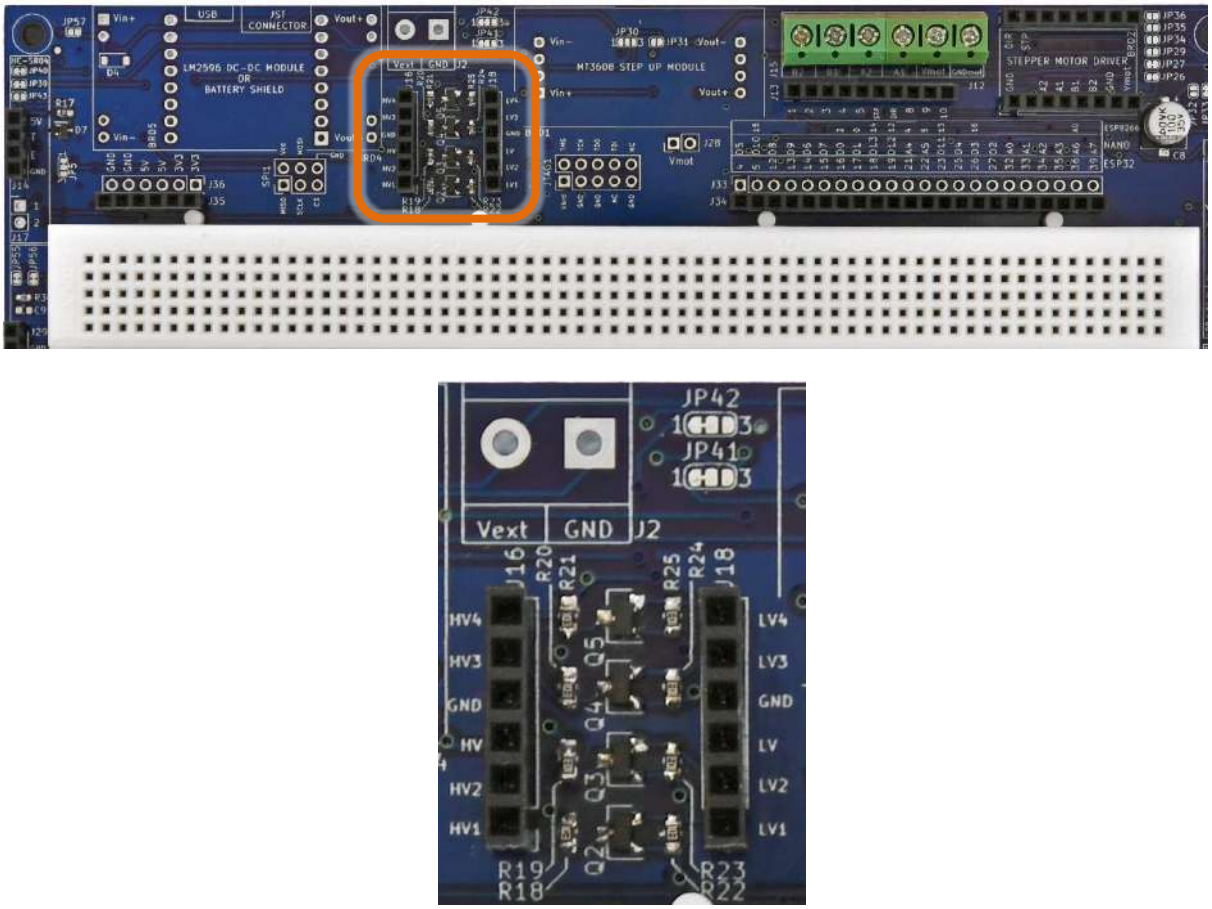


Figure 101 – The Level Shifter section of the IoT Proto Shield Plus

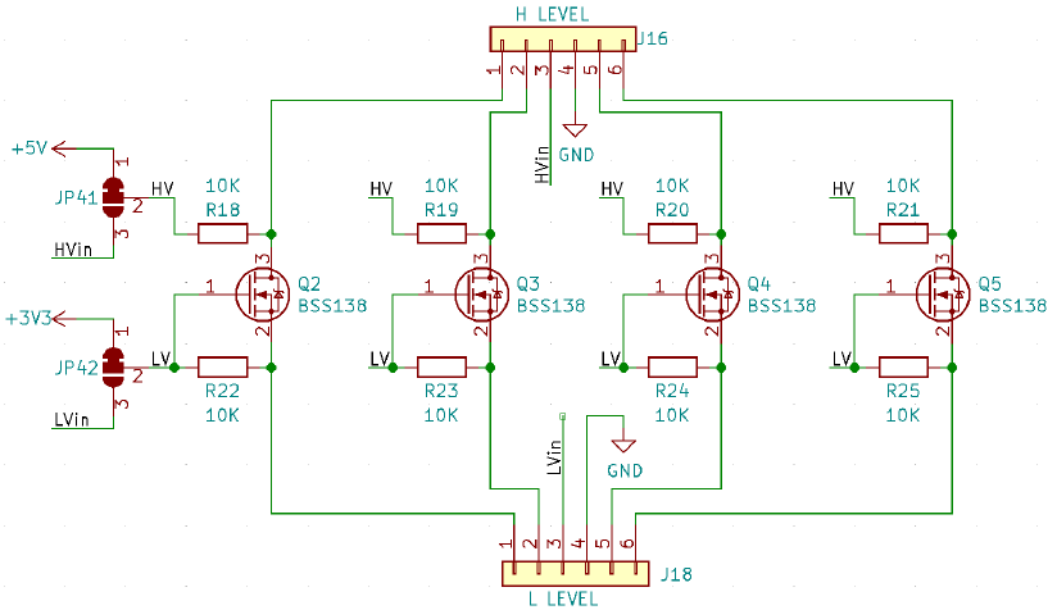


Figure 102 – Schematic of the Level Shifter section of the IoT Proto Shield Plus

33.1. Using the onboard level shifter with different HVin and / or LVin

If you need to use the onboard level shifter with different HVin, you must disconnect JP41 pad #1 from pad #2 (cut in the middle) and connect JP41 pad #2 to pad #3 (with a little drop of tin).
Now you must provide your desired HVin to J16 pin #3 header.

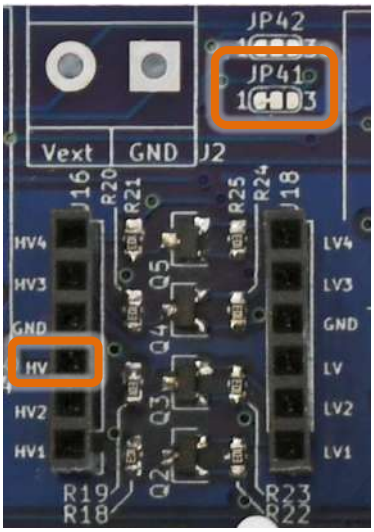


Figure 103 – JP41 and J16 pin#3 of the Level Shifter section of the IoT Proto Shield Plus

If you need to use the onboard level shifter with different LVin you must disconnect JP42 pad #1 from pad #2 (cut in the middle) and connect JP42 pad #2 to pad #3 (with a little drop of tin).
Now you must provide your desired LVin to J18 pin #3 header.

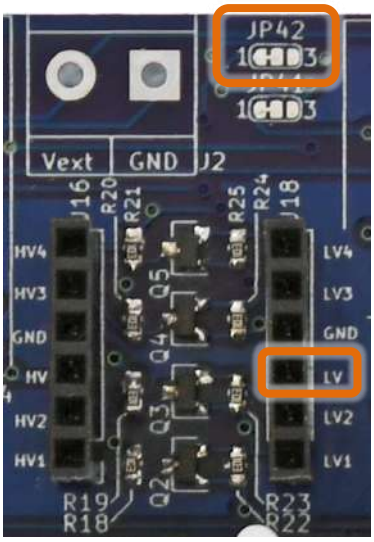


Figure 104 – JP42 and J18 pin#3 of the Level Shifter section of the IoT Proto Shield Plus

34. ESP01 WiFi module

If you want to experiment with an ESP01WiFi module (ancestor of the newer ESP8266 modules) you can solder a 2x4 header to BRD6 pads and solder a 1x8 header to J42 (see Figure 105).
J42 lets you access ESP01 pins easier than with its 2x4 native footprint.

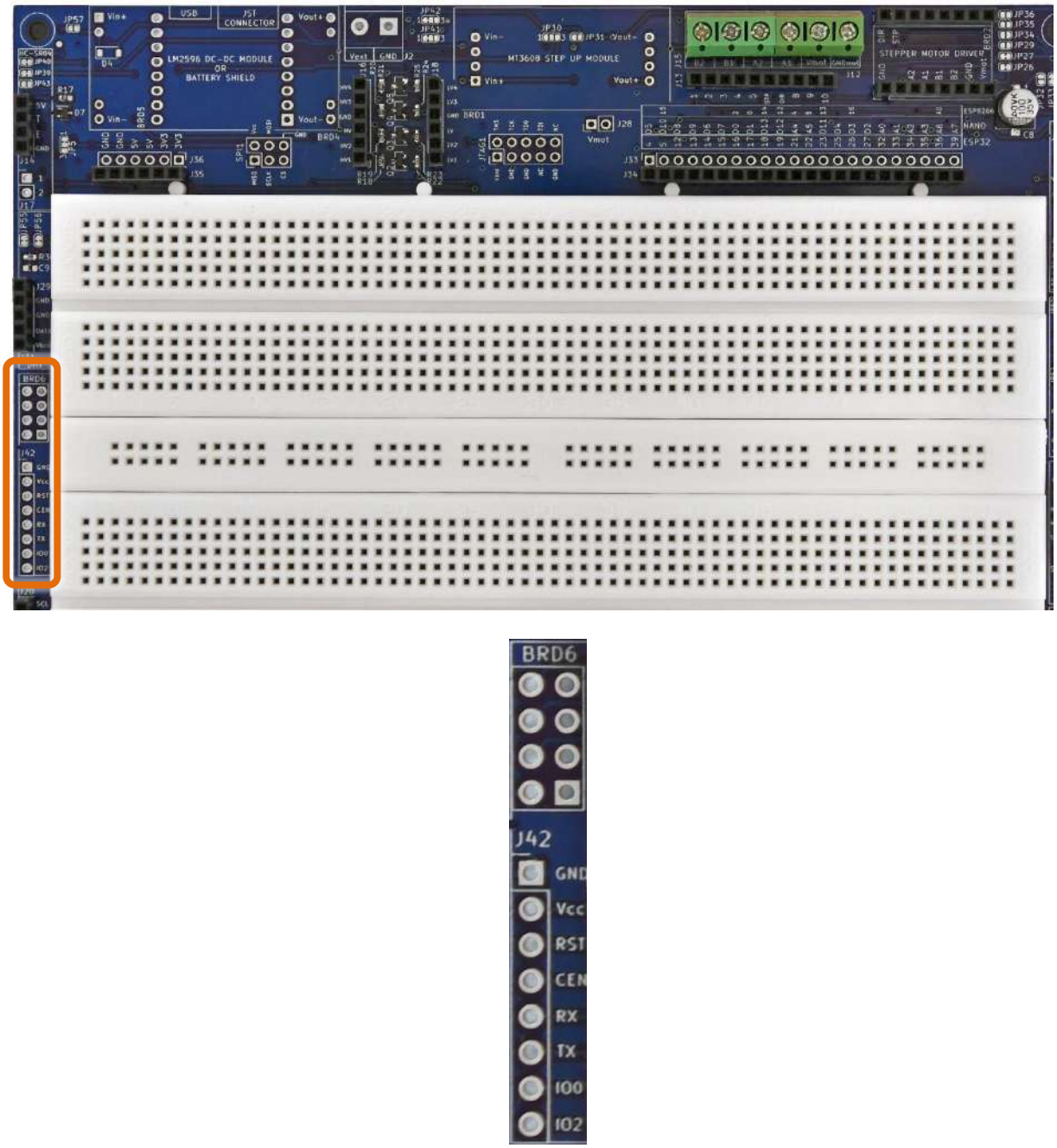


Figure 105 – The ESP01 section of the IoT Proto Shield Plus

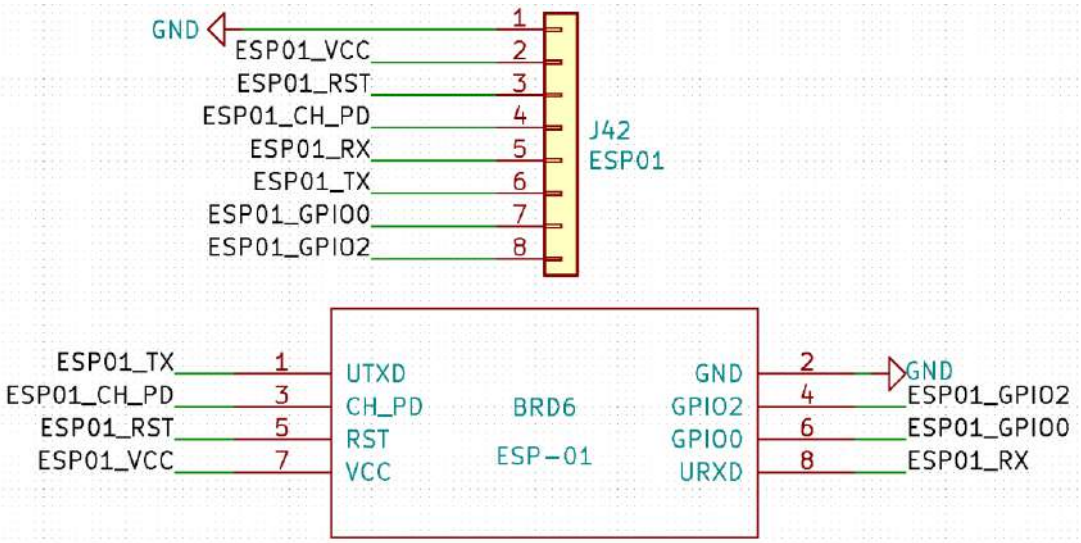


Figure 106 – Schematic of the ESP01 section of the IoT Proto Shield Plus

35. EXTERNAL POWER SUPPLY

DISCLAIMER

This section is recommended for advanced user, that's why this part is not populated on the board.
Refer to the **IoT Proto Shield Plus** schematics (see §36) and to the technical specifications of the board you are using to correctly use an external power supply combined with an LM2596 module.
Choosing a wrong voltage regulator or configure a wrong routing of Vin power rail (see schematics on §36) may damage the ESP or NANO board you are using.

The **IoT Proto Shield Plus** can accept typical LM2596 DC-DC step down modules of widespread use (see Figure 107).
The LM2596 module must be plugged into the **IoT Proto Shield Plus** according to Figure 109.
To plug the LM2596 module you need four 2 pins header (not included in the **IoT Proto Shield Plus**) to be soldered to Vin+, Vin-, Vout+ and Vout- pads of the **IoT Proto Shield Plus**.
The external power supply can be provided by means of JP2 screw terminals connector (see Figure 108 and Figure 110).
If you do not have a 2 screw terminals connector, you can use the two pads of the board and solder two wires directly to them.
The 2 pads screw terminals connector is not provided with the **IoT Proto Shield Plus** and it must be purchased and soldered separately.
The external power supply connects to IN+ (Vin+ on the silkscreen) of the LM2596 module by means of JP57, which must be CLOSED (with a little drop of tin).
If you want to add a reverse polarity protection you can add D4 (i.e., 1N5819, see Figure 110) but in this case JP57 must be OPEN.
The OUT+ (Vout+ on the silkscreen) regulated voltage of the LM2596 module connects to Vin of the **IoT Proto Shield Plus**.

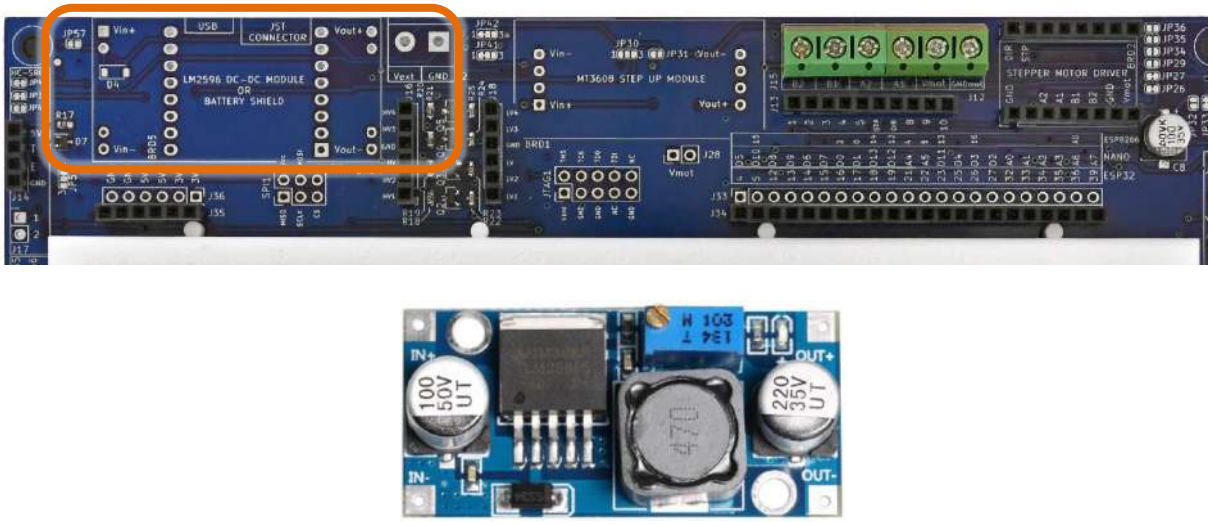


Figure 107 – A typical LM2596 DC-DC module of widespread use

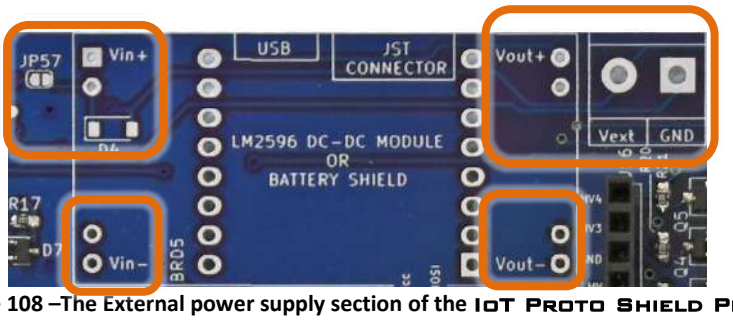


Figure 108 –The External power supply section of the IoT Proto Shield Plus

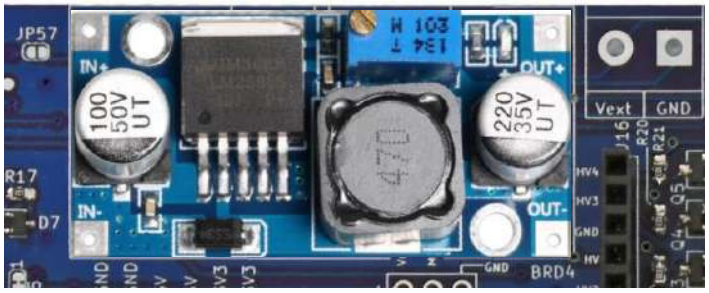


Figure 109 – A typical LM2596 DC-DC module of widespread use plugged into the IoT Proto Shield Plus

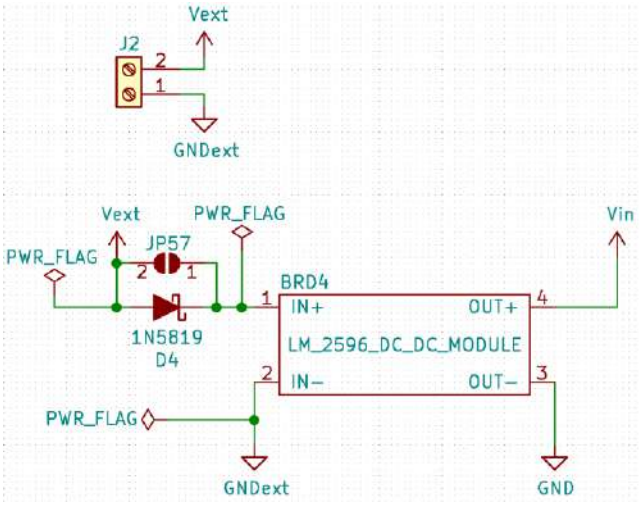


Figure 110 – Schematic of the External power supply section of the IoT Proto Shield Plus

35.1. Battery Shield

Alternatively, to the LM2596 module, you can use a typical battery shield module of widespread use (see Figure 111).
The battery shield module is used to boost the 3.7Vdc of a one element Lipo battery, up to 5V in order to supply it to the 5V pin of the ESP32 or Vin pin of the NANO boards (see Figure 114).
You connect the Lipo battery to the battery shield module by means of its JST connector.
To plug the battery shield module, you need two 8 pins header (not included in the **IoT Proto Shield Plus**) and solder them to the pads showed in Figure 112.
Then you can plug the battery shield module according to Figure 113.

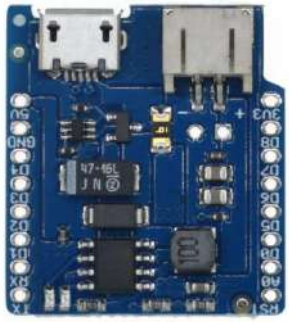


Figure 111 – Typical battery shield of widespread use

Figure 112 – BRD5 socket of the IoT Proto Shield Plus

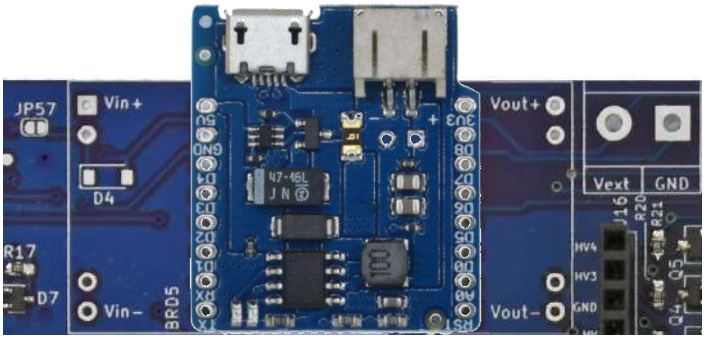


Figure 113 – Typical battery shield of widespread plugged into BRD5 socket of the IoT Proto Shield Plus

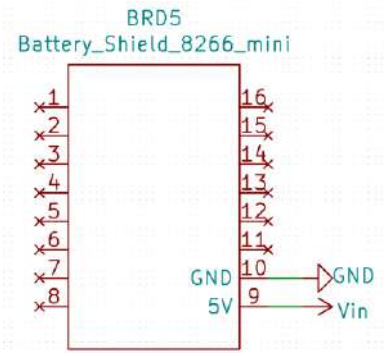
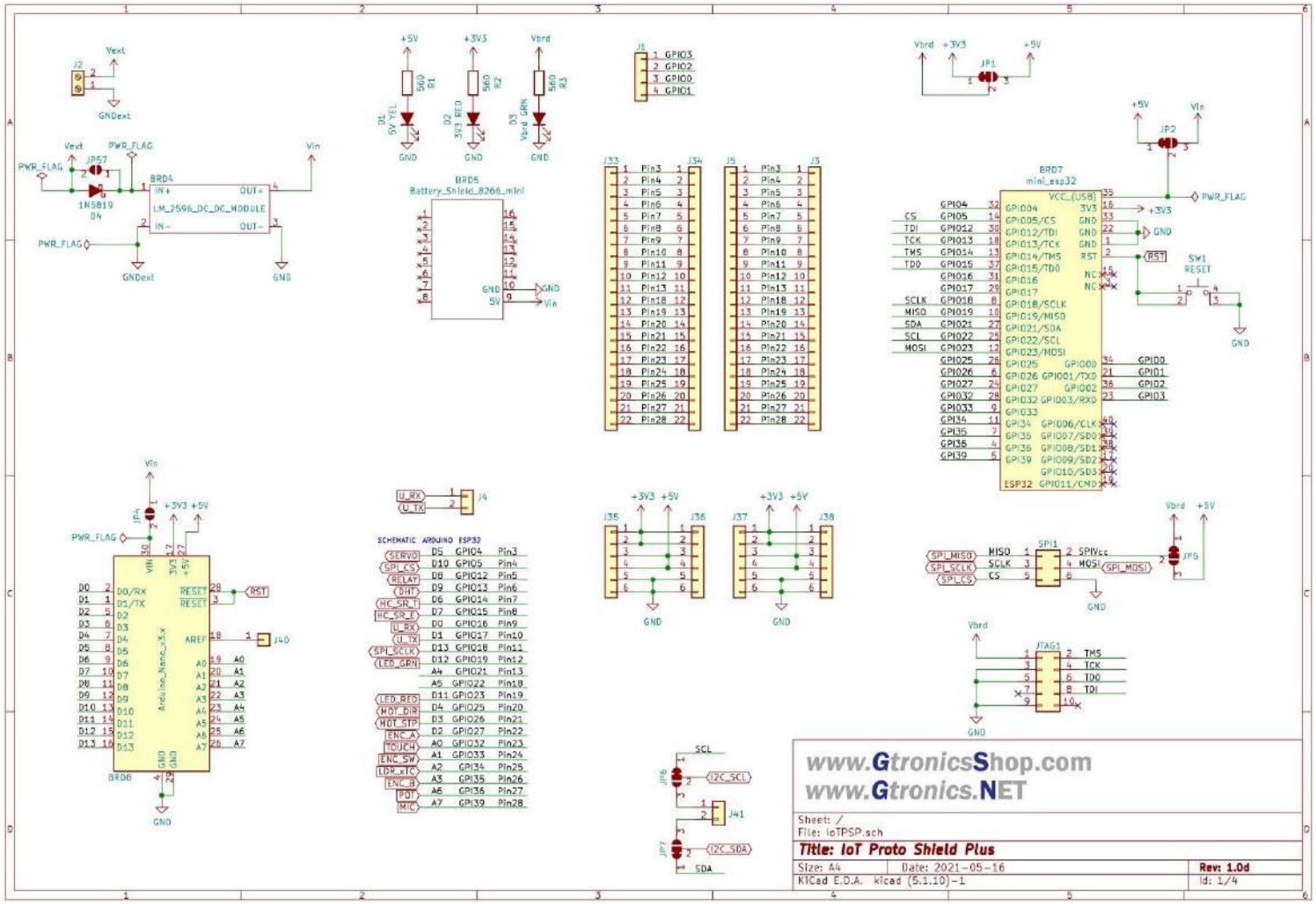
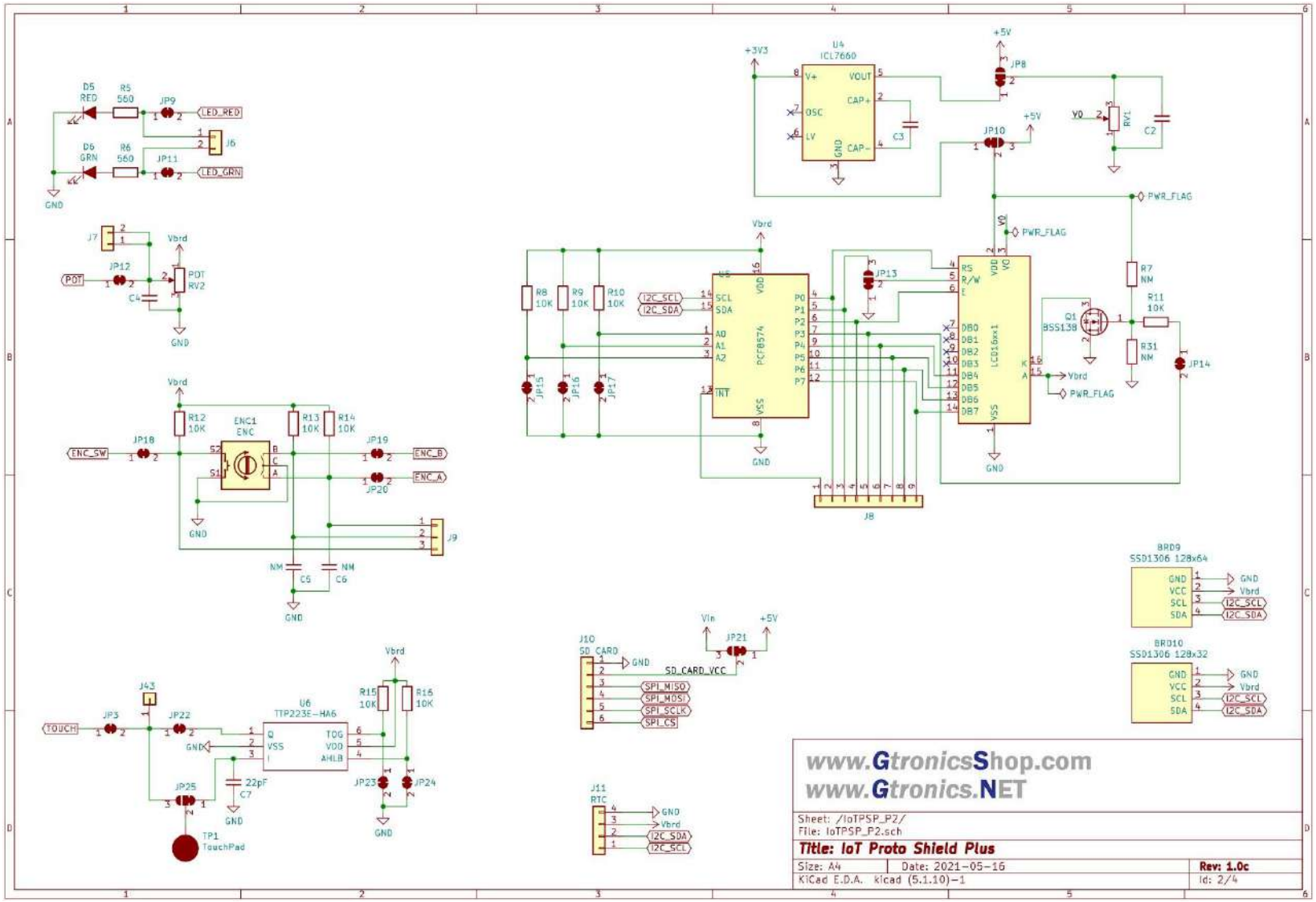
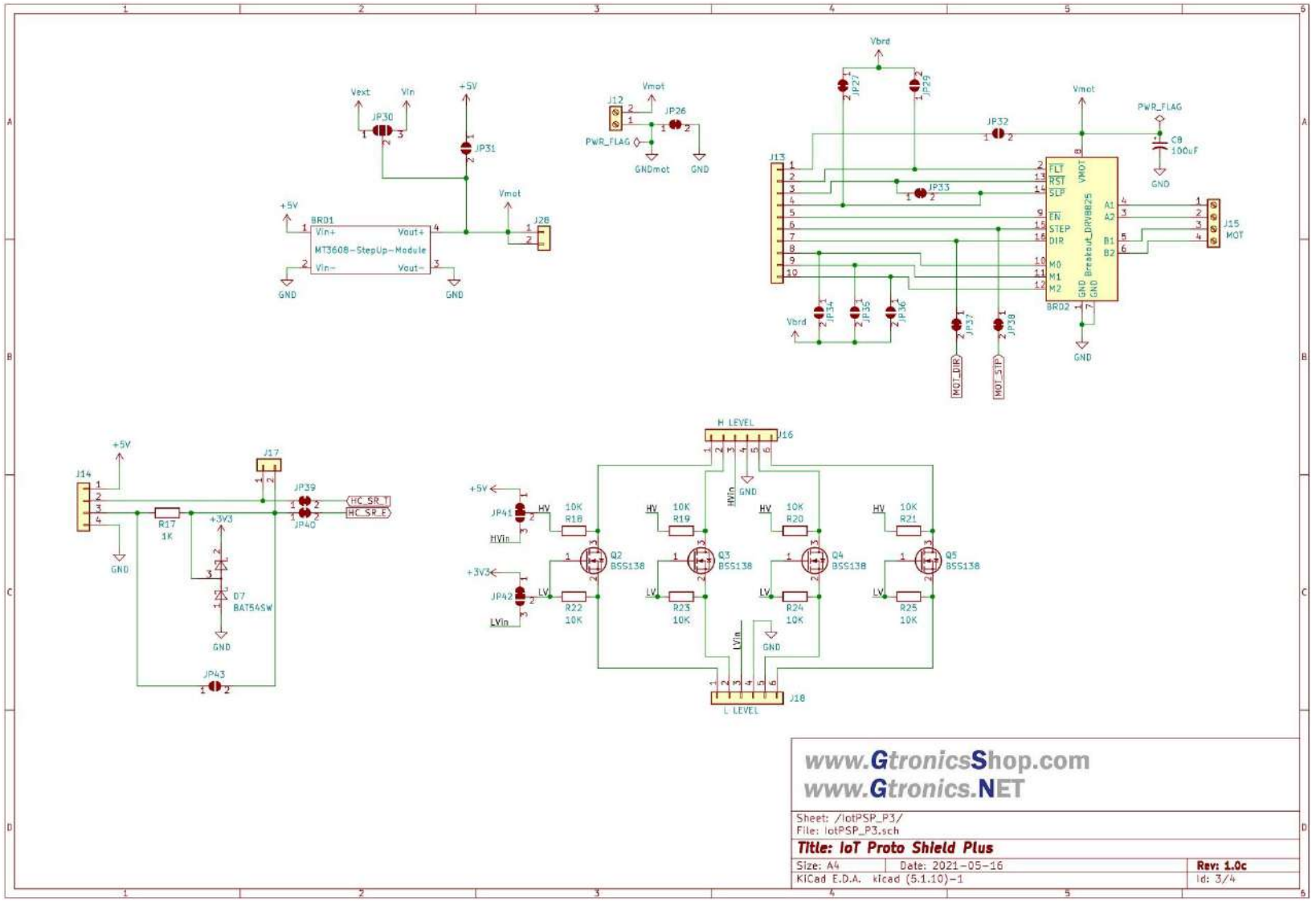


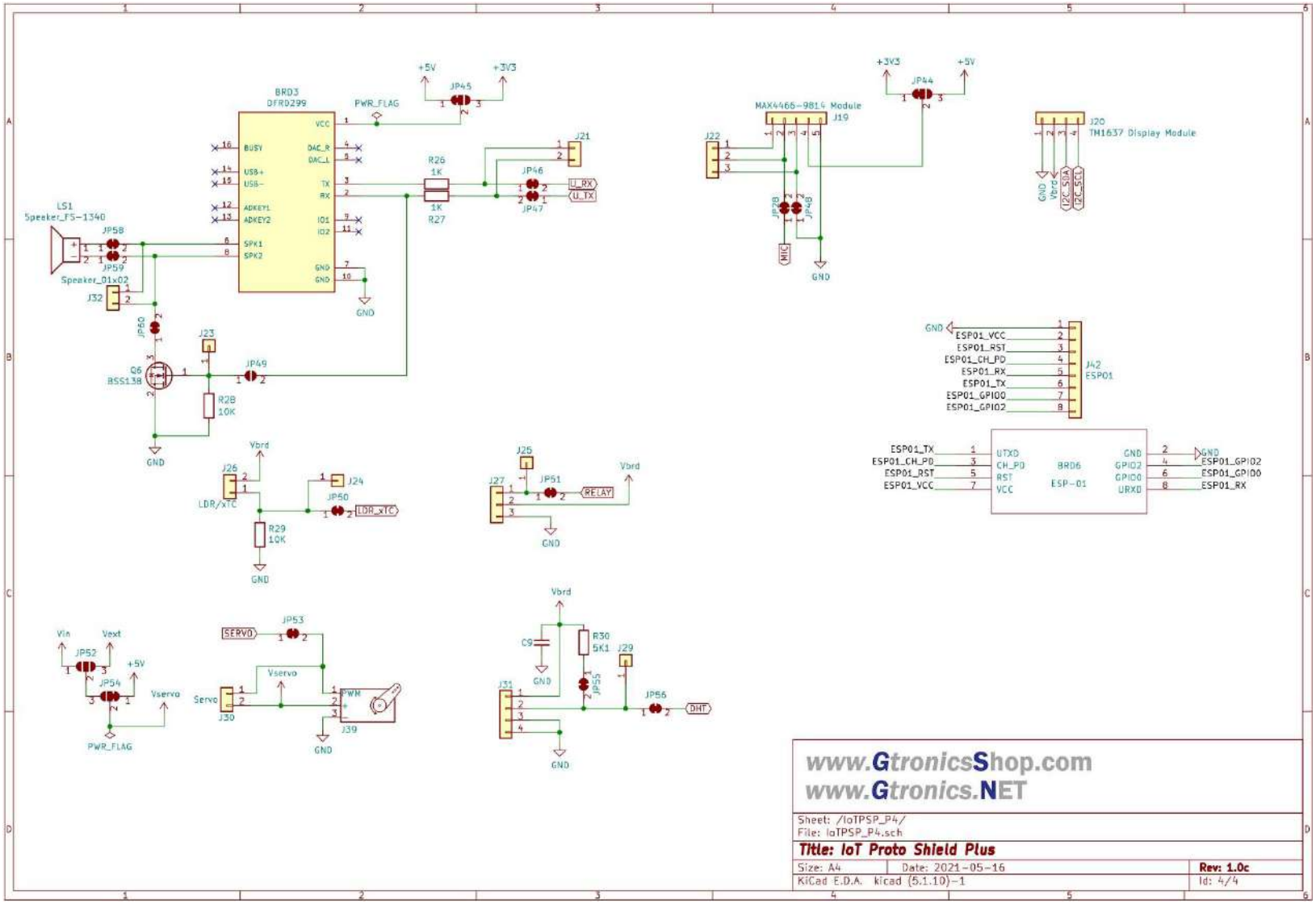
Figure 114 – Schematic of the battery shield section of the IoT Proto Shield Plus

36. The IoT Proto Shield Plus schematic design




www.GtronicsShop.com
www.Gtronics.NET
Sheet: /IoTPSP_P2/
File: IoTPSP_P2.sch
Title: IoT Proto Shield Plus
Size: A4
Date: 2021-05-16
Rev: 1.0c
KiCad E.D.A. kicad (5.1.10)-1
Id: 2/4


www.GtronicsShop.com
www.Gtronics.NET
Sheet: /IotPSP_P3/
File: IotPSP_P3.sch
Title: IoT Proto Shield Plus
Size: A4
Date: 2021-05-16
Rev: 1.0c
KiCad E.D.A. kicad (5.1.10)-1
Id: 3/4


BRD3
DFRD299
PWR_FLAG
+5V
+3V3
JP45
VCC
BUSY
DAC_R
DAC_L
USB+
USB-
TX
RX
ADKEY1
ADKEY2
IO1
IO2
SPK1
SPK2
GND
J21
R26
1K
R27
1K
JP46
JP47
U_RX
U_TX
LS1
Speaker_FS-1340
JP58
JP59
Speaker_01x02
J32
JP60
J23
Q6
BSS138
JP49
R28
10K
GND
MAX4466-9814 Module
J19
J22
JP44
JP8
JP48
MIC
J20
TM1637 Display Module
GND
Vbrd
I2C_SDA
I2C_SCL
ESP01_VCC
ESP01_RST
ESP01_CH_PD
ESP01_RX
ESP01_TX
ESP01_GPIO0
ESP01_GPIO2
J42
ESP01
BRD6
ESP-01
UTXD
CH_PD
RST
VCC
GND
GPIO2
GPIO0
URXD
J26
LDR/xTC
JP50
LDR_xTC
R29
10K
J24
J25
J27
JP51
RELAY
Vin
Vext
JP52
JP54
Vservo
PWR_FLAG
SERVO
JP53
Servo
J30
PWM
J39
C9
R30
5K1
JP55
J31
J29
JP56
DHT
www.GtronicsShop.com
www.Gtronics.NET
Sheet: /IoTPSP_P4/
File: IoTPSP_P4.sch
Title: IoT Proto Shield Plus
Size: A4
Date: 2021-05-16
Rev: 1.0c
KiCad E.D.A. kicad (5.1.10)-1
Id: 4/4