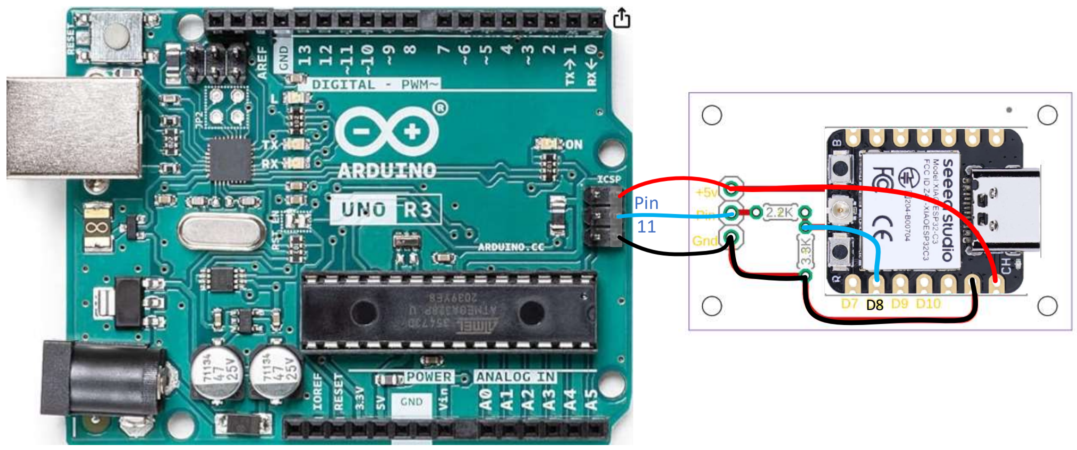
Arduino Tasmota Gateway

Problem definition:
Available Arduino solutions for controlling mains powered devices use solid state or electromechanical relays that could expose users to mains voltage. This solution uses wireless connectivity to control CE approved sealed Tasmota smartplugs.

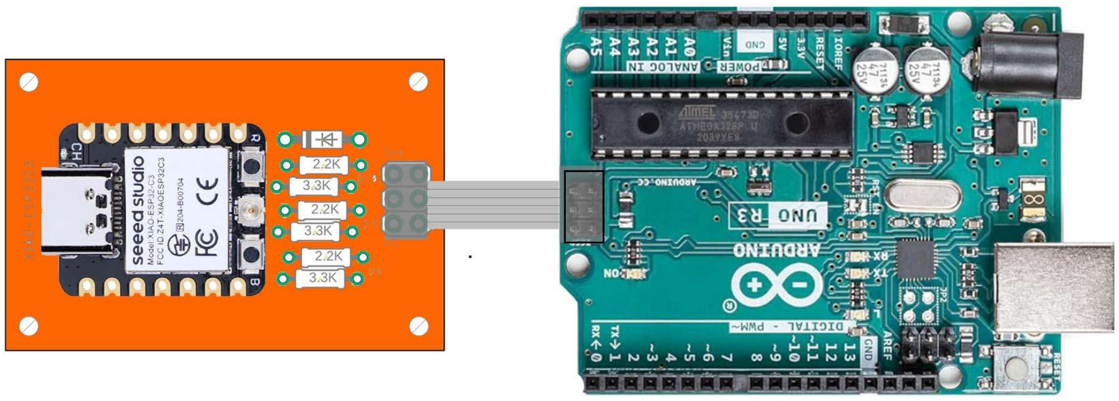
The basic circuit for controlling a single smartplug is an ESP32 with a voltage level shifter to drop a 5v Arduino to the 3.3v ESP32 logic level. In the example below, the ESP32 is powered using +5 volt and ground from the six pin ICSP connector. The middle ICSP pin is Arduino digital pin 11 connected via a logic level voltage divider to ESP32 pin D8.



Pairing and configuration of the ESP32 is covered in a separate document, here we assume the ESP has been paired to a smartplug and configured to respond on ESP pin D8.

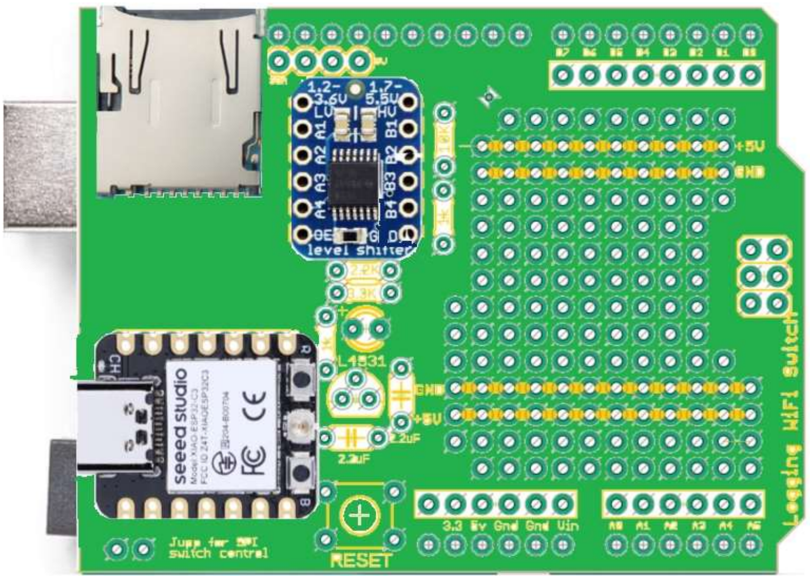
A sketch running on the Uno turns the smartplug on by setting pin 11 HIGH, and off by setting Pin11 low. Any free Arduino pin could be wired but pin 11 was chosen to enable connection via a three pin female jumper.

An example for controller up to three independent smartplugs is similar to the above circuit with ICSP pins 12 and 13 also connected using voltage dividers.



Here a 6 pin IDC ribbon cable is used to connect the PCB to the Arduino ICSP connector. Note the orientation shown ensures that the 5V and ground connections are correct.
With this wiring, Arduino pins 11, 12 and 13 are connected to ESP pins 10,9,8 respectively. See the documentation on configuration to check the config file matches this pin mapping.
As described in the previous example, setting Arduino pins HIGH and LOW sets the corresponding Tasmota smartplug on and off.

Arduino applications requiring data logging can use a custom logging shield, shown below.

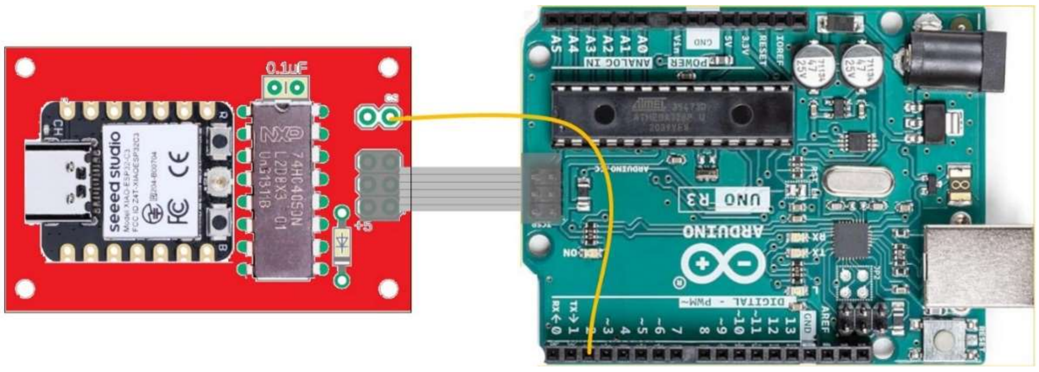


Arduino pin 8 should be used to control the smartplug. This is connected to ESP32 pin 7, check that pin 7 is set in the config file.

ESP32 Pin	Arduino Pin
7 (CS)	8

Note pins 10-13 are used for SD card data logging and are not available for use directly by the sketch if logging is utilized.

SPI control - future development not yet supported
With the ESP32 wired to the Arduino SPI pins, with the logging shield above or the mini SPI board shown below, a Tasmota library can be used to control smartplugs.



The SPI interface allows control of as many smartplugs as are configured on the gateway, without tying up any pins other than those required for SPI. It also enables reading of power and voltage statistics from connected smartplugs. It can also provide wireless signal quality between the gateway and connected smartplugs.
SPI mode is enabled by setting: "SPI_controlMode": true in the config.json file. With this enabled, pin change control is disabled. (setting: "SPI_controlMode": false restores pin control.

Library instantiated with the gateway chip select (pin 2 in the picture above, pin 8 on the logging shield)
const Int gatewayCS = 2; // pick a pin not otherwise used by the sketch
Smartplug plug(gatewayCS); //arduino pin 2 is chip select for the gateway.

Example calling convention with a single smartplug:

  Calling `powerOn()` turns on the smartplug
  Calling `powerOff()` turns off the smartplug
  Calling `readPower()` returns a structure with ...
  Calling `readRSSI()` returns wifi signal strength percent((0-100)

If more than one smartplug is configured, the above calls operate on the first configured smartplug. To operate on other configured smartplugs, an argument must be given to the above commands indicating the index into the list of configured smartplugs.

For example, assuming the following config.json fiile:

```
{ "SPI_controlMode": false,
  "esp_pin_map": [-1, -1, -1],
  "plug_ip": [13,12],
  "plug_mac4": ["EA2D", "E946"]}
```

`powerOn(0)` turns on plug at IP address 192.168.4.13 (same as calling powerOn())
`powerOn(1)` turns on the plug at IP address 192.168.4.12

notes:

- all plugs have routing prefix of 192.168.4
- esp pin map values are ignored in SPI control mode