EE 1301: IoT

Quick Lesson

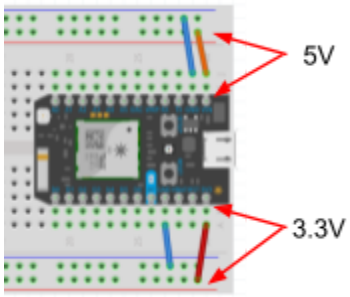
"Getting to know your pins"

Power Supplies, Analog, and Digital Pins

In this quick lesson, we will review the different types of pins utilized in your Particle Photon. It is assumed you have reviewed "Quick Lesson - Electrical Circuits" or are familiar with basic electronics theory.

Power Supplies

Circuits that do work, compute, or change a signal in some manner generally need a voltage source. Most integrated circuits (ICs) we use in this course require a single power supply pin and a ground pin to operate. A power supply is a type of voltage source (discussed in Quick Lesson - Electrical Circuits.) The power supply for our Particle Photons is provided by the USB connector (nominally 5V). The pins labeled "3V3" and "VIN" are power supplies provided by the Particle Photon to operate other circuits.



Power supplies are specified by their nominal voltage. For example, the pin "3V3" is a power supply with a nominal output of 3.3 Volts. The pin "VIN" is a power supply with a nominal output of 4.5 Volts. The Photon has some protection circuits between the USB provided power and the "VIN" pin, for most purposes you may treat it as a 5V power supply. It is almost always assumed that a given power supply is designed to return current through the ground (GND) pin. If you're having problems with your circuits be sure to check that both the power supply and ground pins are well connected.

WARNING:
Never connect a 5V supply to a 3.3V circuit!

Circuits are designed to work with certain power supplies. Connecting the wrong power supply can damage your circuit (especially connecting a higher voltage supply to a low voltage circuit.)

Power supplies also have limits on the amount of current they can supply. If a power supply is asked to supply too much current the power supply could fail and become damaged, though it is more common that the supply just reduces its output voltage. The Particle Photon has a current limit of 100mA from the 3.3V supply. The VIN supply is limited by the USB power supply (usually 500mA, but it can be much lower in older computers.)

Signals

When two circuits need to communicate information there are two broad categories of signals to accomplish this, **analog** and **digital** signals. In addition, there is a hybrid category of signal called **PWM**.

Analog Signals

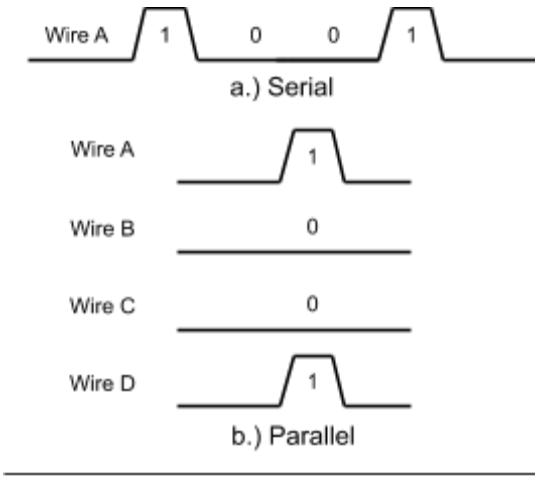
In analog signaling, the information is passed along based on the voltage of the wire. For example, 1.0 Volts might correspond to 25℃ and 1.2 Volts might correspond to 30℃.

Analog signals are generally specified by a minimum voltage, a maximum voltage, and often a slope (ie., volts/℃). Analog signals can nearly instantaneously transmit data, require only one pin, and are very simple to operate but they suffer from sensitivity to noise and loss of information when copied/reproduced.

The TMP36 temperature sensor is a good example of a sensor with an analog signal output. The "AnalogRead()" function can read analog signals on the A0-A5 pins. It returns a value between 0 and 4095. The AnalogWrite() function can output an analog voltage on the DAC and A3 pins. PWM (described below) can also be utilized to output an analog signal with some limitations.

Digital Signals

In digital signaling, the information is passed along based on a series of high and low pulses. For example, "0011" might correspond to 25℃ and "1000" might correspond to 30℃.



Examples of serial digital signals of the decimal number 9

Digital signals come in two varieties. With **parallel** digital signals, every bit of information requires a signal wire. Since our microcontrollers have a limited number of pins (around 16), this severely limits the number of devices you can connect.

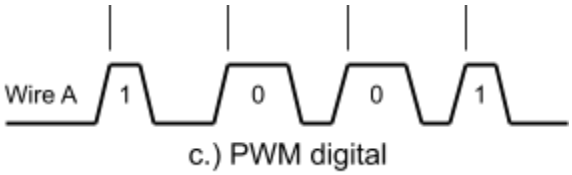
With **serial** digital signals, the bits are sent sequentially down a signal wire over time. This requires a complex dedicated circuit on both ends to encode and decode the information. Often these encode/decode circuits can be directly integrated into an existing component (like the iLED, sensor IC, or the Photon microcontroller.)

Since serial digital signals are encoded, each device sending or receiving information must be designed using the same serial digital signaling standard. Some common standards available for communication with your Photon include Universal Serial Bus (USB), Inter-Integrated Circuit

(I^2C), Controller Area Network (CAN), and Serial Peripheral Interface (SPI or Wire). Each device that utilizes these standards should give you an idea of what is needed to set up communication. If you have further questions try Google first then talk to your TA.

Common digital signalling pins on the Particle are D0,D1 (I2C); A2-A5 (SPI); D1,D2 (CAN). Notice that some of them overlap, when implementing a project with multiple communication standards you will want to carefully review the Photon datasheet to plan your I/O (link at bottom of this document.)

PWM



c.) PWM digital

In Pulse-Width Modulation (PWM) signaling, a digital pulse is continuously transmitted on a wire. The width of this pulse confers the information being transmitted. The signaling can be either digital or analog in nature. In the case of an analog PWM signal, the width of the pulse is continuously variable. For example, in an ideal servo motor, a pulse width of 1500µs corresponds to a position of 0° and 2300µs corresponds to a position of +90°, 700µs corresponds to a position of -90°. Any pulse width between 700µs and 2300µs will result in the tracking of the servo to a unique position within the capabilities of the servo.

The “AnalogWrite()” function creates PWM output on pins following pins: D0-D3, A4, A5, WKP, RX, TX. Since the PWM signal toggle on and off very quickly, if you are driving a “slow” element, like a motor or an LED, PWM can be used to control the average speed or brightness in an analog fashion.

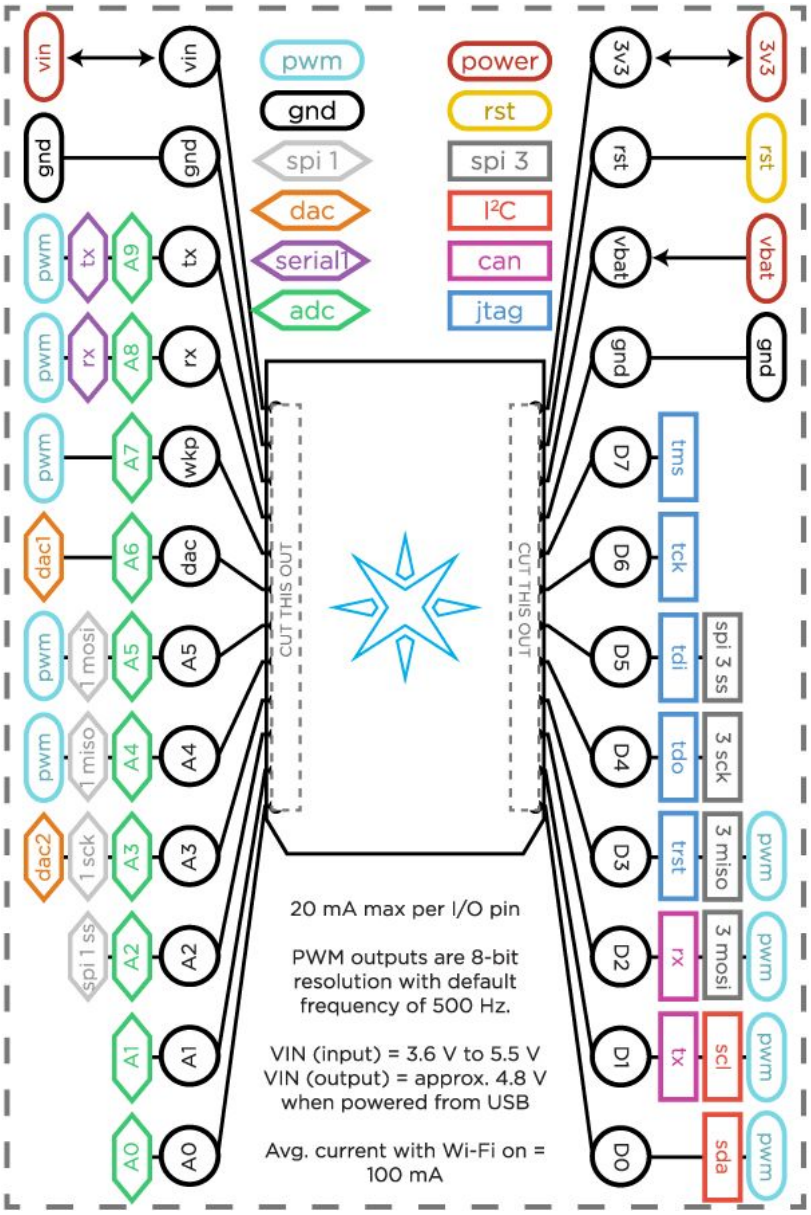
Serial Port

A serial port is typically a 4 wire digital interface but can be used with just 3 wires. The wires are transmit (TX), receive(RX), ground (GND), and power (VCC, optional). A serial port is bidirectional, which means it can both send and receive data at the same time. Unlike a clocked interface, both sides of a serial port must know, in advance, the transmit/receive rate (or BAUD rate). RS-232 is a standard for serial communication and is often used to refer to a serial port. See: https://en.wikipedia.org/wiki/RS-232

Serial ports communicate 8 bits of data at a time (usually encoded with a start and a stop bit.) This byte is often referred to as a character. Using the ASCII standard these 8-bits can be decoded into a character and displayed on the screen for you to read. When connected with PuTTY or CoolTerm, each character you see on the screen represents one byte transmitted from your Photon. Every key you depress will send 8-bits to your photon (even though you don’t see it appear on your screen by default.) There is a local_echo feature that will show, on your screen, every character you transmit to the Photon.

The Photon has two serial ports. One is available through the USB port with the appropriate driver from Particle (we installed this in Lab 1). This is accessible via the Serial.* functions and can be externally accessed by the TX and RX pins. The other is available via the D0(receive) and D1(transmit) pins (plus GND and 3V3 as needed.). The second serial port is accessible with the Serial2.* functions.

Pinout Diagram



Further Reading

https://docs.particle.io/datasheets/wi-fi/photon-datasheet/