Digital Communications in Microcontroller Systems

Rick Sellens

analogRead(A0)

- Reading analog data monitors voltages from sensors
  - directly, or after signal conditioning, e.g. amplification
- One measurement, one wire, one input, no communication
- Analog all the way and subject to noise, voltage drop in long cables, etc.
- If we want to change that, we need another way to talk to the sensors

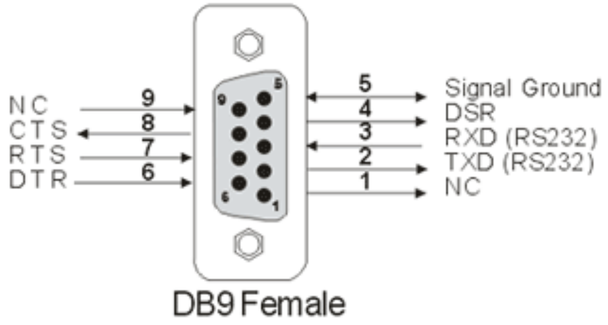
Digital Data Paths

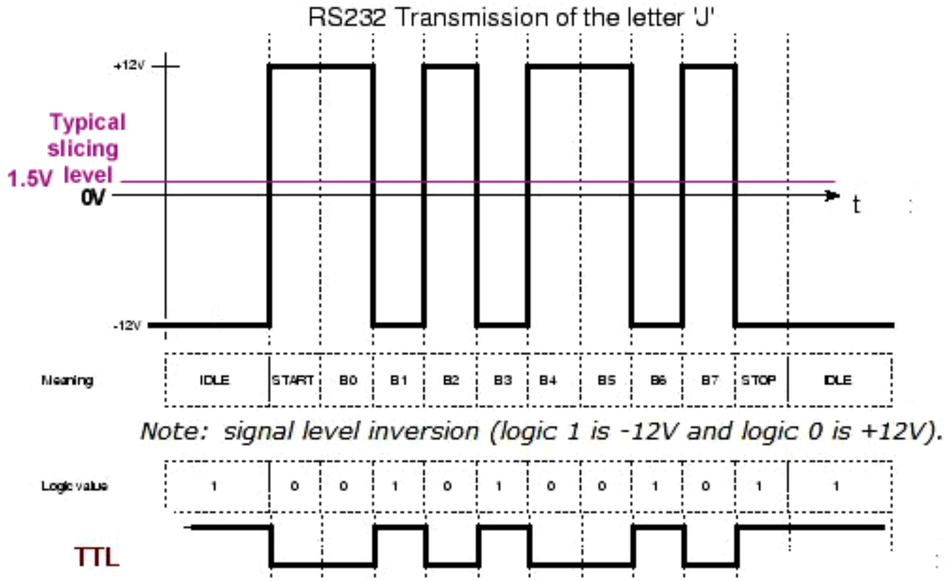
- Analog outputs are converted at the sensor, or very close by — reduces analog noise, etc.
- Can transmit multiple values on a single wire
- Can share a single wire between multiple sensors
- Can be wireless via various RF protocols, e.g. Bluetooth
- Can include error checking, etc. for lossless data transmission

Serial v Parallel

- Parallel transmission allows multiple bits to be sent simultaneously over multiple wires.
  - 8 bit parallel transmission is 8 times as fast as serial at the same switching speed, which was really important 30+ years ago.
  - Computer memory paths still parallel inside the box and on the microcontroller for speed
- Most external data transmission now serial over a single wire per data path, allowing multiple bits in time series

RS232 Serial Communication





- Minimum 2 wires plus ground
  - RX for receiving data signals
  - TX for sending data signals
  - GND for a common 0 voltage reference
- Standard says +/-12 volts
- Common to use the same encoding at TTL voltage levels (0, +5V)

Basic Serial (232 or TTL)

- Simple if both ends are set up the same e.g. 9600 baud, no parity, 8 data bits, 1 stop bit. Almost all are “none, 8, 1” in current devices
- Standard speeds of 110, 300, 1200, 2400, 4800, 9600…57600, 115200 baud for bits per second, but unlikely to see anything under 4800
- Asynchronous, so both ends create their own timing clock, need to resynchronize with each character (stop bits)
- Can be software controlled by switching DIO on and off at just the right timing, but hard to do anything else at the same time.
- Better to use a UART (Universal Asynchronous Receiver/Transmitter) for hardware control, so the microcontroller can be doing other things

Serial.begin(9600)

- Start the Arduino serial port running at 9600 baud to talk to the USB on the computer, or another device attached to pins 0 and 1.
- Serial1.begin(115200) to start another serial port (if the controller has one)
- Serial2.begin, etc. and the microcontroller will need enough serial ports for all the devices you want to talk to. **One port each.**

Choose a baud rate

- 9600 is the Arduino sketch standard, slow, but will work on anything
- 57600 is the fastest for some hardware, including some FTDI serial to USB hardware
- 115200 is fast enough that you probably don’t have to care most of the time, and still one of the standard speeds.

NDI 35 YEARS Medical

RS232 is old but not obsolete

State of the art magnetic field tracker for image guided surgery research communicates by RS232 serial, and we just wrote Arduino code to talk to it

Aurora

Track multiple disposable tools or sensors situated inside or outside the body with this customizable, real-time electromagnetic tracking system that delivers sub-millimetric, sub-degree accuracy.

Send with >
Receive marked with <

17:11:33> INIT:E3A5

17:11:33< OKAYA896

17:11:33> VER:4A6EF

17:11:33< Aurora Control Firmware
NDI S/N: A4-00215
Characterization Date: 09-20-2007
Freeze Tag: AURORA Rev 007.000
Freeze Date: 2007-09-21
(C) Northern Digital Inc.
D23A

17:11:33> SFLIST:10C14E

17:11:33< 04D715

17:11:33> SFLIST:1200CF

17:11:33< 1144D4

17:11:33> VER:7A7AF

17:11:33< Aurora Field Generator S/N: 250
Aurora Field Generator Model: Planar 1.0

17:11:47> TSTART:5423

17:11:49< OKAYA896

17:11:49> TX:0001031A

17:11:49< 010AMISSING0000003F00000000
0000581F

17:11:49> TX:0001031A

17:11:49< 010A+02014-08176-00889+05319+001360-002713-009688+00826
0000797F

Send with >
Receive marked with <

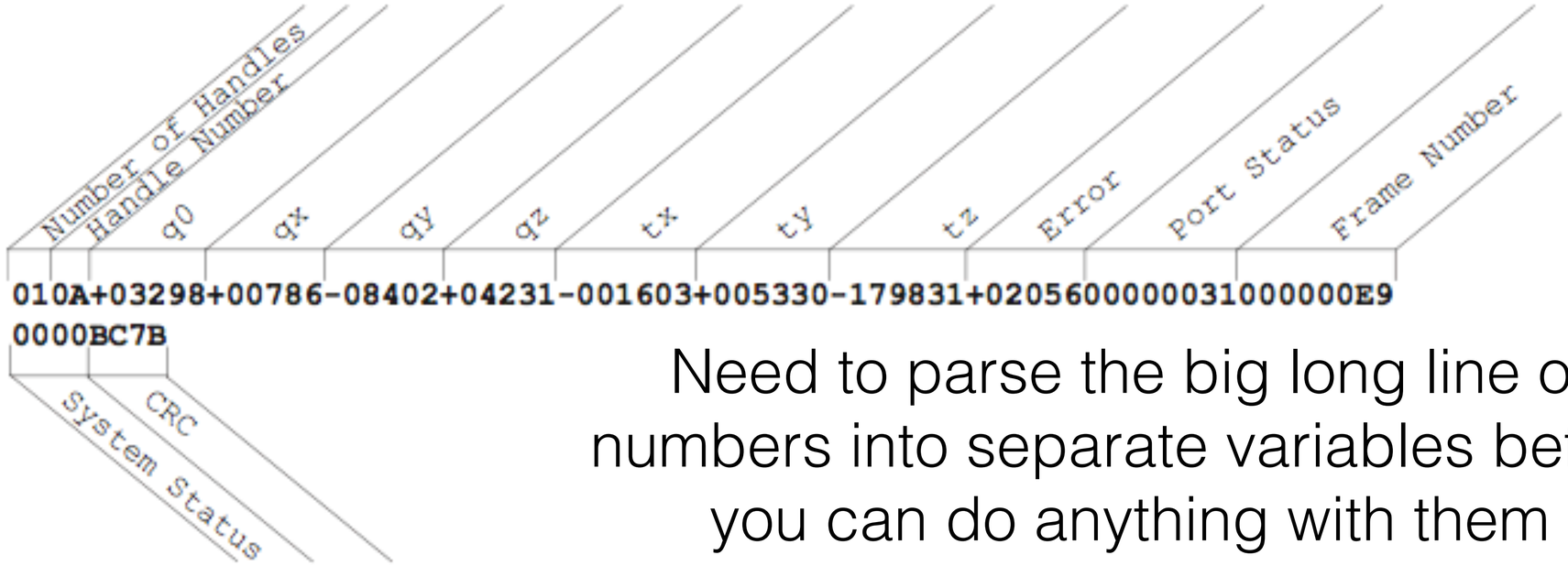
17:11:49> TX:0001031A

17:11:49< 010A+02014-08176-00889+05319+001360-002713-009688+008260000003F00000070
0000797F

Command:

TX

Reply:



Need to parse the big long line of numbers into separate variables before you can do anything with them

The system returned transformation data for one tool.

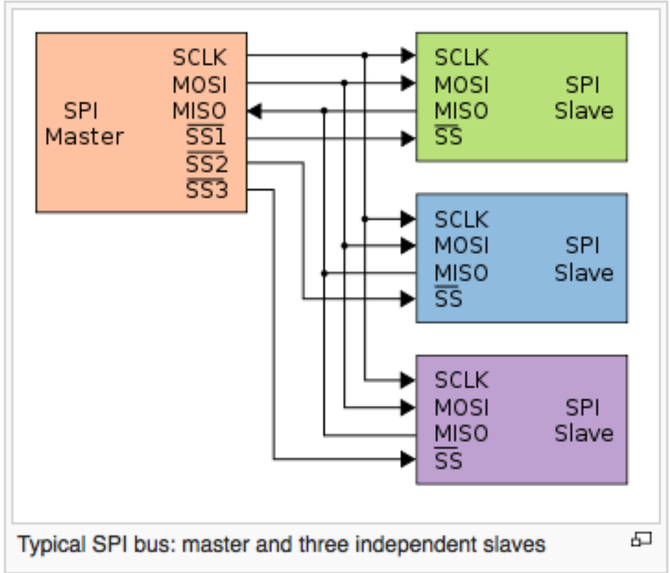
USB (Universal Serial Bus)

- Much faster, much more complicated to program, communicates to self configure on a plug&play basis (which was a big deal 20 years ago)
- dedicated USB chips drive up cost/complexity
- Device registration, so you know what is plugged in
- **Provides up to 500 mA of 5 volt power (4.75-5.25)**
- Can talk to simple serial devices through an FTDI cable or similar adapter, but usually takes some configuration.

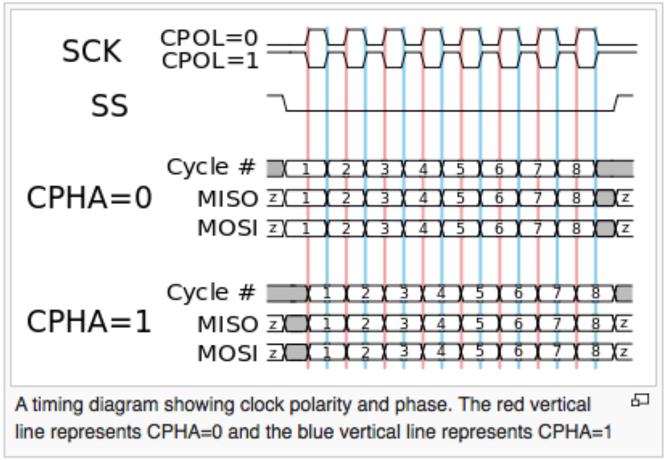
SPI (Serial Peripheral Interface Bus)

- Four wires (in addition to ground), digitally modulated, master/slave, full duplex with clock and one transmission line in each direction
- Additional slave devices require individual select lines
- Speeds of 10 Mbps +, but often slower
- Common for SD cards and displays using Arduino SPI library

- https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi



Typical SPI bus: master and three independent slaves

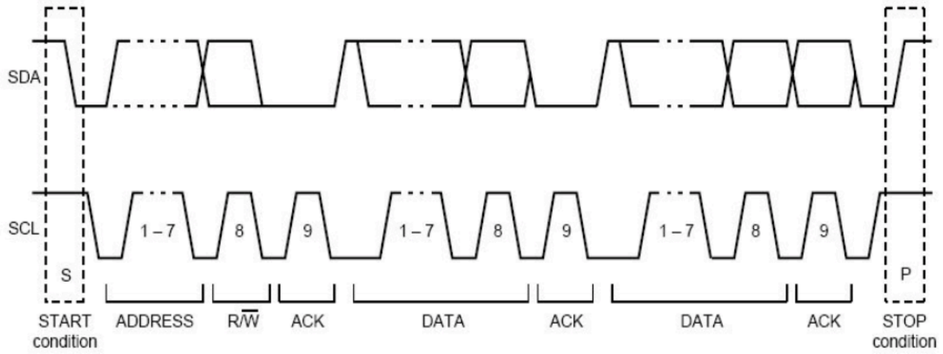


A timing diagram showing clock polarity and phase. The red vertical line represents CPHA=0 and the blue vertical line represents CPHA=1

I2C (Inter-Integrated Circuit)

- Two wires, SCL and SDA, digitally modulated, bi directional (half duplex, meaning transmission only goes one way at a time)
- 7 bit address allows master (usually a microcontroller) to communicate with multiple sensors (or other devices) on the same wires
- speeds up to 3.2Mbps, but often slower (100, 400, 1000, 3200 kbps), data sent in Bytes
- Arduino Wire library allows you to use I2C, similar to Serial

```
Wire.begin()
Wire.beginTransmission(address)
Wire.write()
Wire.read()
```



More data to and fro

- Sensor can store its own calibration information
- Sensor can have different operating modes, set by micro-controller sending codes
- One sensor can communicate multiple measurements on the same line
- BMP 180 pressure results are sensitive to temperature, need compensation, provides an example you will use in MECH 217

Restricted data sheet

BMP180
Digital pressure sensor

Bosch Sensortec

BOSCH
Invented for life

About $2 per unit

Key features

Pressure range:	300 ... 1100hPa (+9000m ... -500m relating to sea level)
Supply voltage:	1.8 ... 3.6V (V_{DD}) 1.62V ... 3.6V (V_{DDIO})
Low power:	5µA at 1 sample / sec. in standard mode
Low noise:	0.06hPa (0.5m) in ultra low power mode 0.02hPa (0.17m) advanced resolution mode

- Temperature measurement included
- I^2C interface
- Fully calibrated

1 hPa = 1 millibar = 100 Pa

Supersedes BMP085(2011), now superseded by BMP280(2015) with both I2C and SPI

Power consumption

- BMP180: 5 uA average current in low power mode at 3.3 volts supply voltage is
  - Power = VI = 3.3 * 5x10^-6 = 16.5 uW
- 100 ohm bridge of strain gauges at 3.3 V
  - I = V/R = 0.033 A or 33 mA (about 6000x more)
  - P = VI = 0.1089 W (so 1/4 W resistors still OK)
- 2 AA batteries (~2000 mA hrs) will last about 61 hours, just powering the bridge, or 45 years just powering the BMP180 (phone batteries similar)

Start by reading calibration coefficients

Table 5: Calibration coefficients

	BMP180 reg adr	
Parameter	**MSB**	**LSB**
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

```
174  /* read calibration data */
175  ac1 = read16(BMP085_CAL_AC1);
176  ac2 = read16(BMP085_CAL_AC2);
177  ac3 = read16(BMP085_CAL_AC3);
178  ac4 = read16(BMP085_CAL_AC4);
179  ac5 = read16(BMP085_CAL_AC5);
180  ac6 = read16(BMP085_CAL_AC6);
181
182  b1 = read16(BMP085_CAL_B1);
183  b2 = read16(BMP085_CAL_B2);
184
185  mb = read16(BMP085_CAL_MB);
186  mc = read16(BMP085_CAL_MC);
187  md = read16(BMP085_CAL_MD);
```

- Factory set during calibration of each sensor
- Only need to be read once, since they won't change

Read Raw Data, then Correct by Coefficients

- Write a request to the sensor
- Wait a little while, then read back the 16 bit result
- Convert the raw result into temperature in C
- Similar conversions for pressure in hPa, millibars or Pascals

```
213 uint16_t Adafruit_BMP085::readRawTemperature(void) {
214   write8(BMP085_CONTROL, BMP085_READTEMPCMD);
215   delay(5);
216   return read16(BMP085_TEMPDATA);
217 }
```

```
340 float Adafruit_BMP085::readTemperature(void) {
341   int32_t UT, B5;     // following ds convention
342   float temp;
343
344   UT = readRawTemperature();
345   B5 = computeB5(UT);
346   temp = (B5+8) >> 4;
347   temp /= 10;
348
349   return temp;
350 }
```

Some funny C operations

A >> 3 means shift the bits in variable A by 3 positions to the right, LSBs fall off, equivalent to dividing by 8, round down

107 in binary is 01101011 shifted right 00110101 = 53
shifted right again 00011010 = 26
shifted right again 00001101 = 13

Each shift to the left << is multiplying by 2

B += 5 is the same as B = B + 5, but more compact
Same with *=, /=, -=, |= (bitwise or) &= (bitwise and)

Decide how you want to sample the pressure

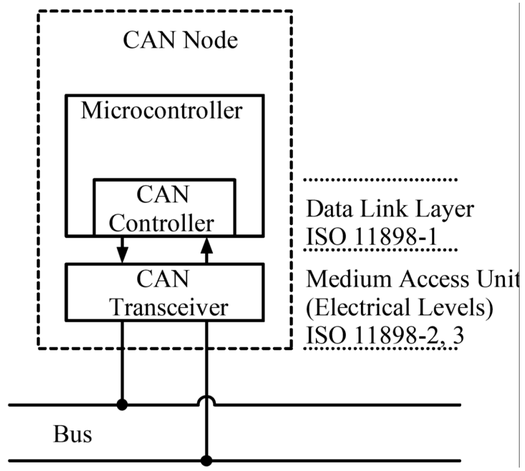
- Oversampling will average over multiple samples right on the sensor BMP180 will take 1, 2, 4, or 8 samples internally and sum them
- Noise is reduced by sqrt(n) when you average, just as if you did it in software on the micro controller
- The micro controller could be away doing something else for 26 ms while the sensor gets a better estimate

```
222  write8(BMP085_CONTROL, BMP085_READPRESSURECMD + (oversampling << 6));
223
224  if (oversampling == BMP085_ULTRALOWPOWER)
225    delay(5);
226  else if (oversampling == BMP085_STANDARD)
227    delay(8);
228  else if (oversampling == BMP085_HIGHRES)
229    delay(14);
230  else
231    delay(26);
232
233  raw = read16(BMP085_PRESSUREDATA);
```

Need to wait longer if taking more samples, and we hate waiting with delay()

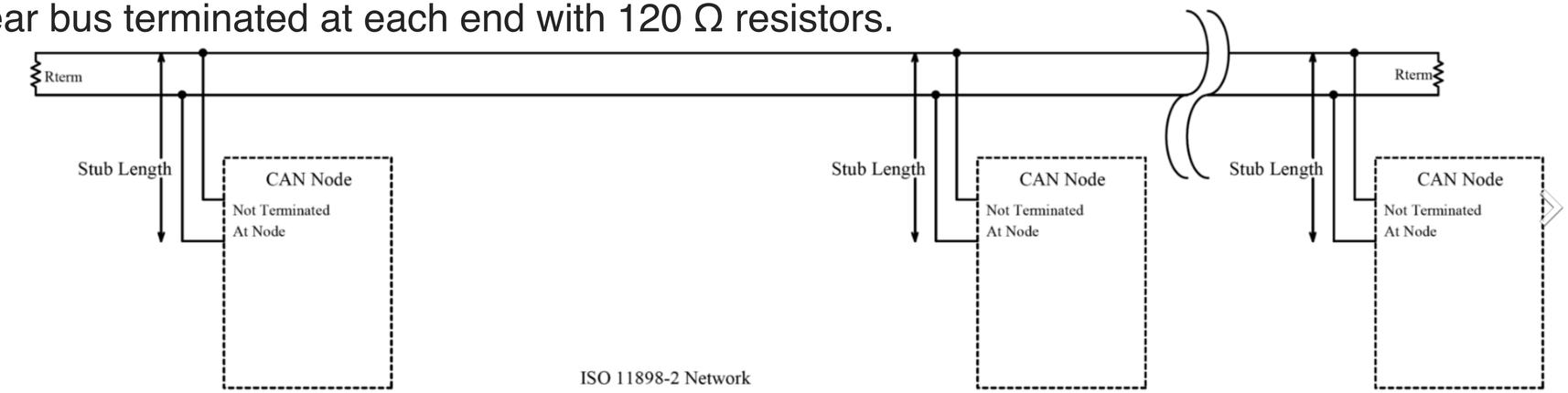
CAN bus

A **Controller Area Network** (**CAN bus**) is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based protocol, designed originally for multiplex electrical wiring within automobiles, but is also used in many other contexts. -Wikipedia



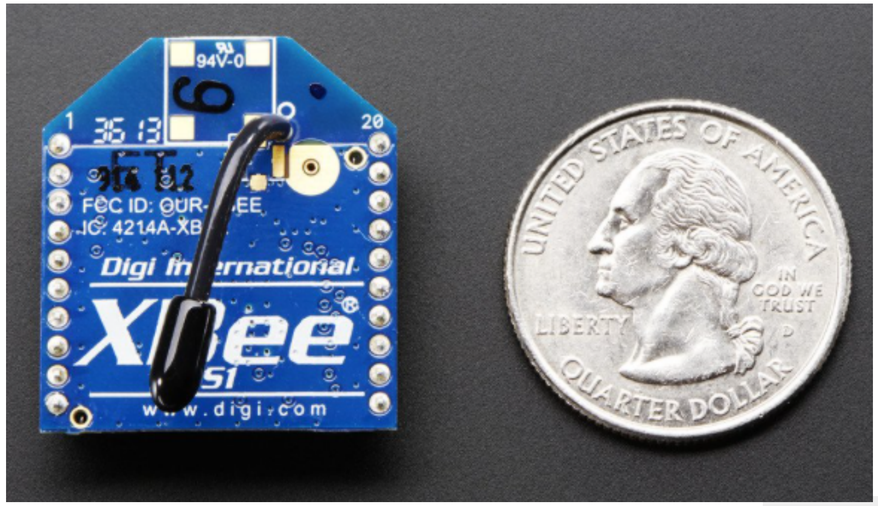
CAN bus is one of five protocols used in the on-board diagnostics (OBD)-II vehicle diagnostics standard. The OBD-II standard has been mandatory for all cars and light trucks sold in the United States since 1996, and the EOBD standard has been mandatory for all petrol vehicles sold in the European Union since 2001 and all diesel vehicles since 2004.[3]

CAN is a multi-master serial bus standard for connecting Electronic Control Units [ECUs] also known as nodes. ISO 11898-2, also called high speed CAN, uses a linear bus terminated at each end with 120 Ω resistors.



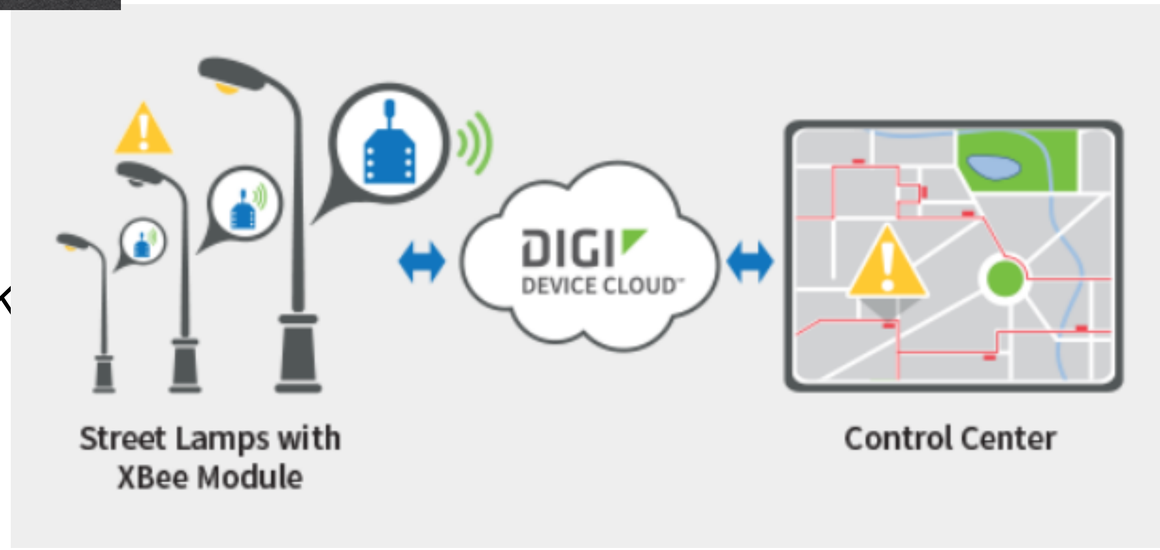
0 wires?

- Radio Frequency (RF) in various flavours
  - Bluetooth, Bluetooth LE, Wifi, etc.
  - Garage door openers
- IR Optical in various flavours
  - most TV remotes

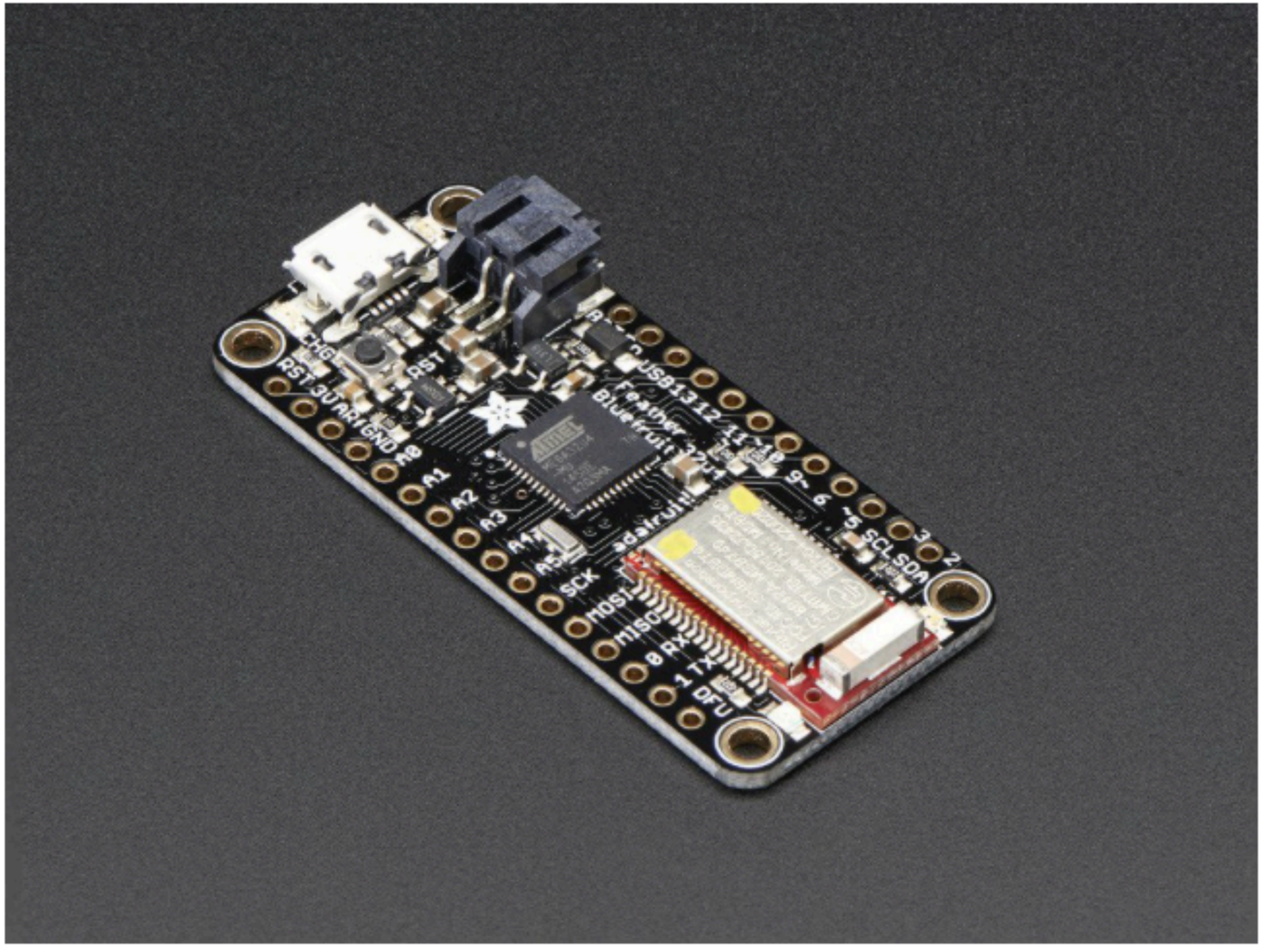


Digi XBee

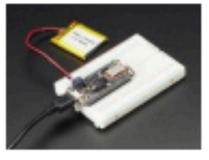
- Wireless Serial port is simplest
- More complex network possible
- Starting at $25/node



	Frequency	Form Factor	Protocol	Multipoint	Mesh	Programmable	Gateway	Modem
XBee® ZigBee	2.4 GHz	TH/SMT	ZigBee		✓	✓	✓	✓
XBee® 802.15.4	2.4 GHz	Through-Hole	802.15.4	✓			✓	✓
XBee® DigiMesh® 2.4	2.4 GHz	Through-Hole	DigiMesh		✓		✓	✓
XBee® Wi-Fi	2.4 GHz	TH/SMT	802.11 bgn	✓				
XBee® SX	900 MHz	Surface Mount	DigiMesh	✓	✓			





Adafruit Feather 32u4 Bluefruit LE

PRODUCT ID: 2829

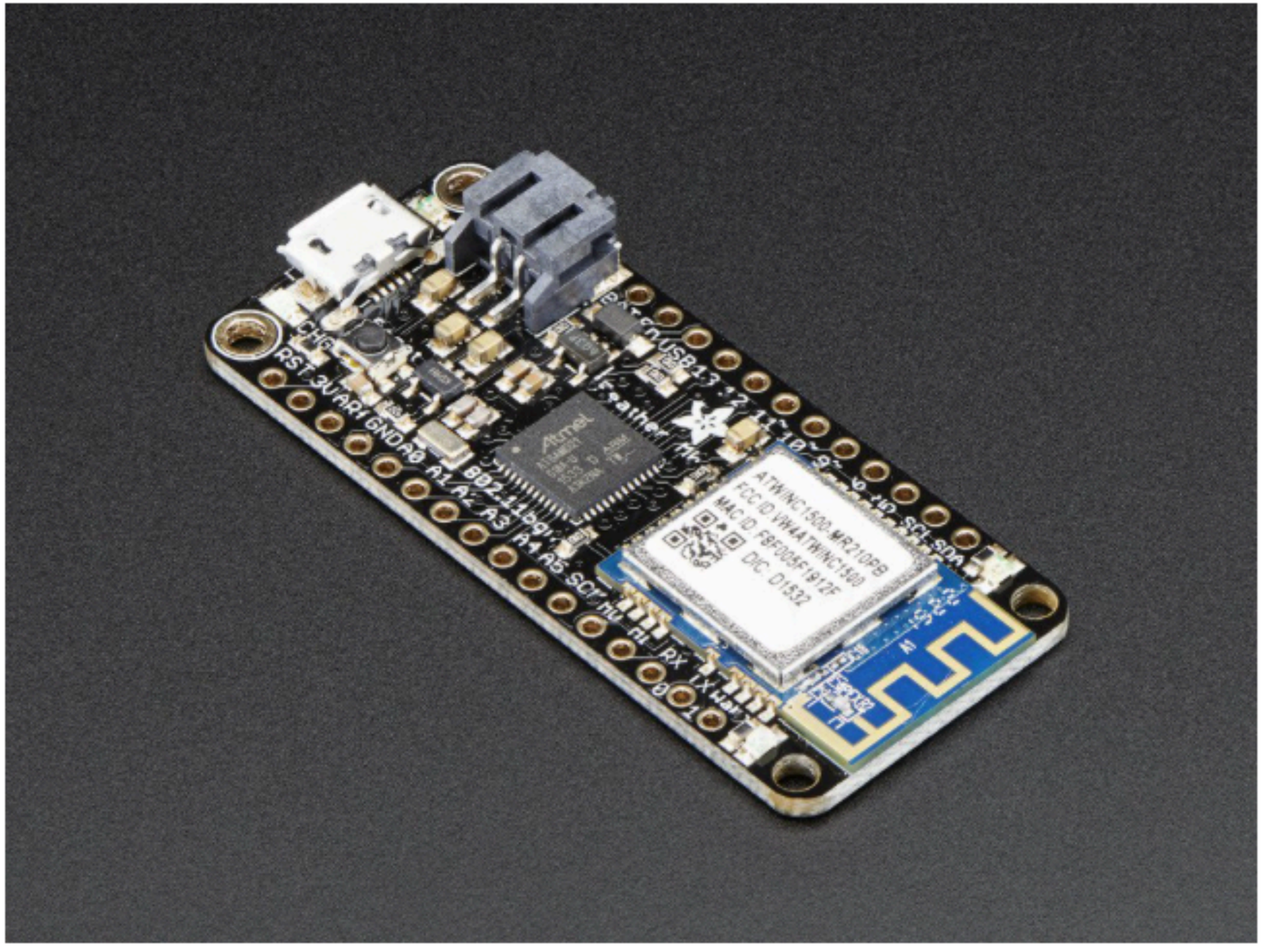
$29.95

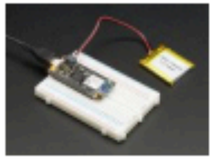
IN STOCK

1 ADD TO CART

- [ ] **Also include 1 x Feather Header Kit - 12-pin and 16-pin Female Header Set ($0.95)**
- [ ] **Also include 1 x Feather Stacking Headers - 12-pin and 16-pin female headers ($1.25)**
- [ ] **Also include 1 x Lithium Ion Polymer Battery - 3.7v 100mAh ($5.95)**
- [ ] **Also include 1 x Lithium Ion Polymer Battery - 3.7v 150mAh ($5.95)**
- [ ] **Also include 1 x Lithium Ion Polymer Battery - 3.7v 350mAh ($6.95)**
- [ ] **Also include 1 x Lithium Ion Polymer Battery - 3.7v 500mAh ($7.95)**
- [ ] **Also include 1 x Short Feather Headers Kit - 12-pin and 16-pin Female Header Set ($1.50)**

QTY	DISCOUNT
1-9	$29.95





Adafruit Feather M0 WiFi - ATSAMD21 + ATWINC1500

PRODUCT ID: 3010

$34.95

IN STOCK

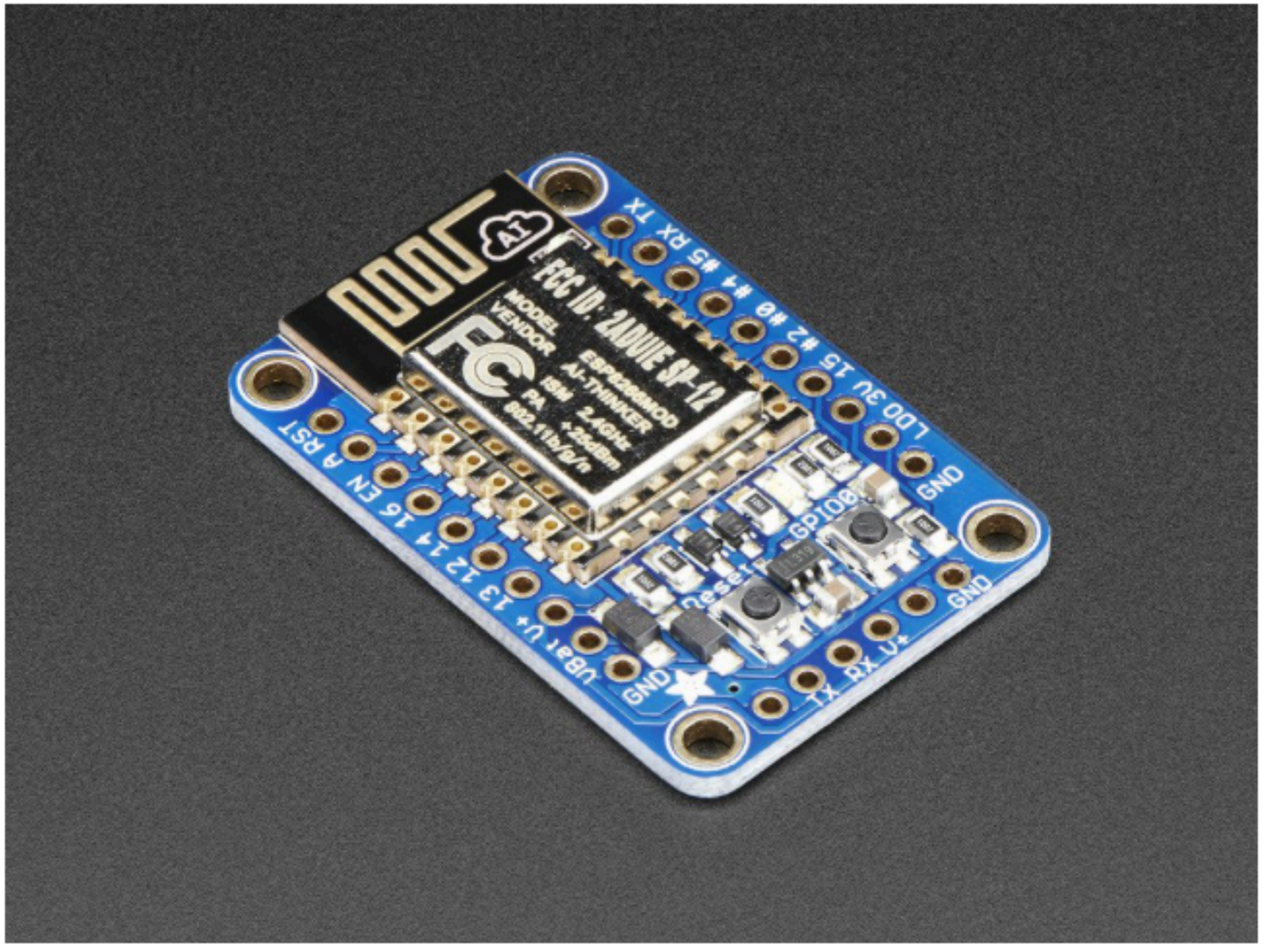
MAX PER CUSTOMER: 5

1 ADD TO CART

- ☐ **Also include 1 x Feather Header Kit - 12-pin and 16-pin Female Header Set ($0.95)**
- ☐ **Also include 1 x Feather Stacking Headers - 12-pin and 16-pin female headers ($1.25)**
- ☐ **Also include 1 x Lithium Ion Polymer Battery - 3.7v 350mAh ($6.95)**
- ☐ **Also include 1 x Lithium Ion Polymer Battery - 3.7v 500mAh ($7.95)**
- ☐ **Also include 1 x Short Feather Headers Kit - 12-pin and 16-pin Female Header Set ($1.50)**

QTY DISCOUNT

1-5 $34.95








Adafruit HUZZAH ESP8266 Breakout

PRODUCT ID: 2471

$9.95

IN STOCK

1 ADD TO CART

☐ **Also include 1 x FTDI Serial TTL-232 USB Cable ($17.95)**

☐ **Also include 1 x USB to TTL Serial Cable - Debug / Console Cable for Raspberry Pi ($9.95)**

QTY	DISCOUNT
1-9	$9.95
10-99	$8.96
100+	$7.96

ADD TO WISHLIST

DESCRIPTION

TECHNICAL DETAILS

LEARN