Agenda

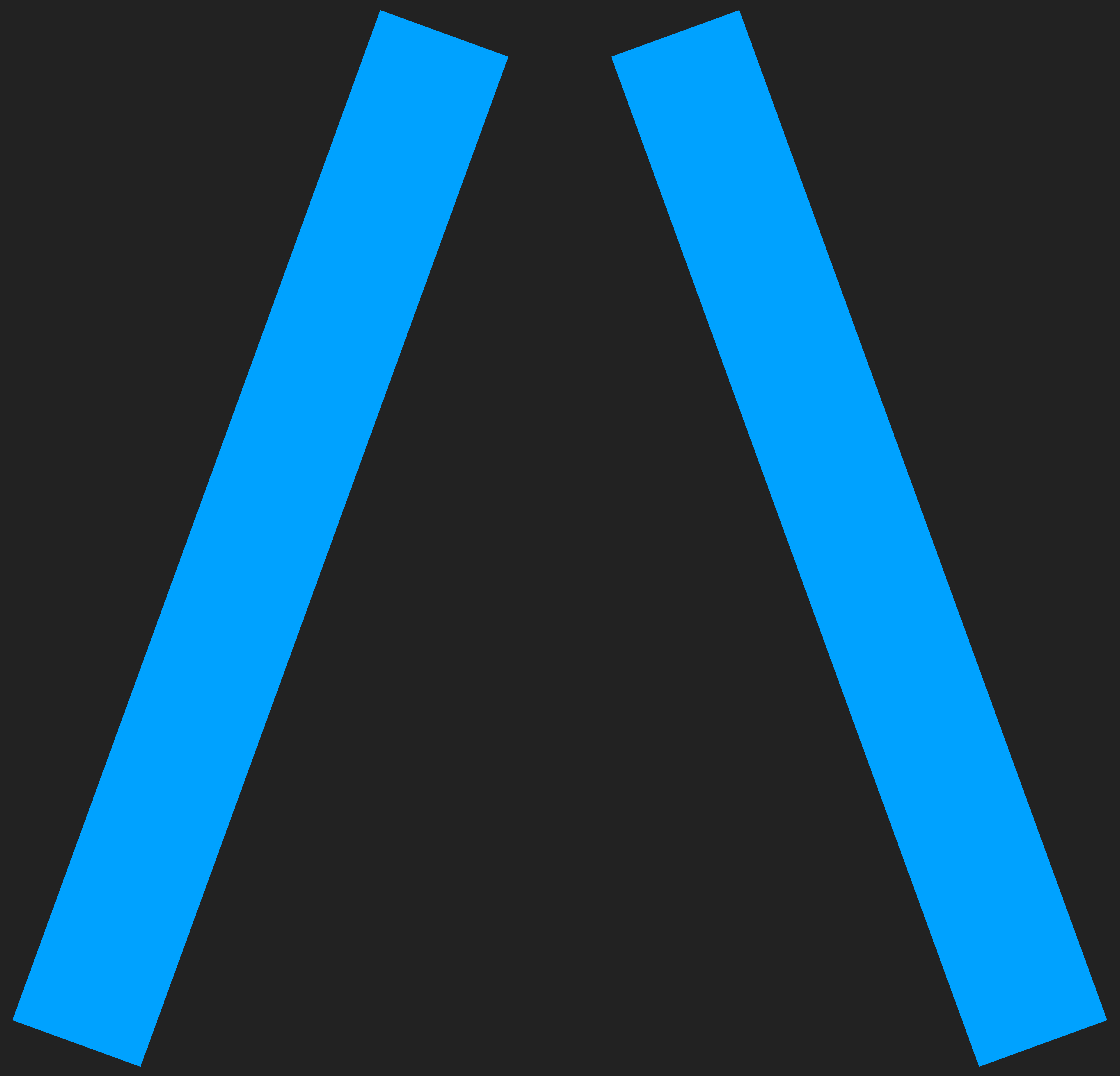
Midterm Review

Semester Lookahead

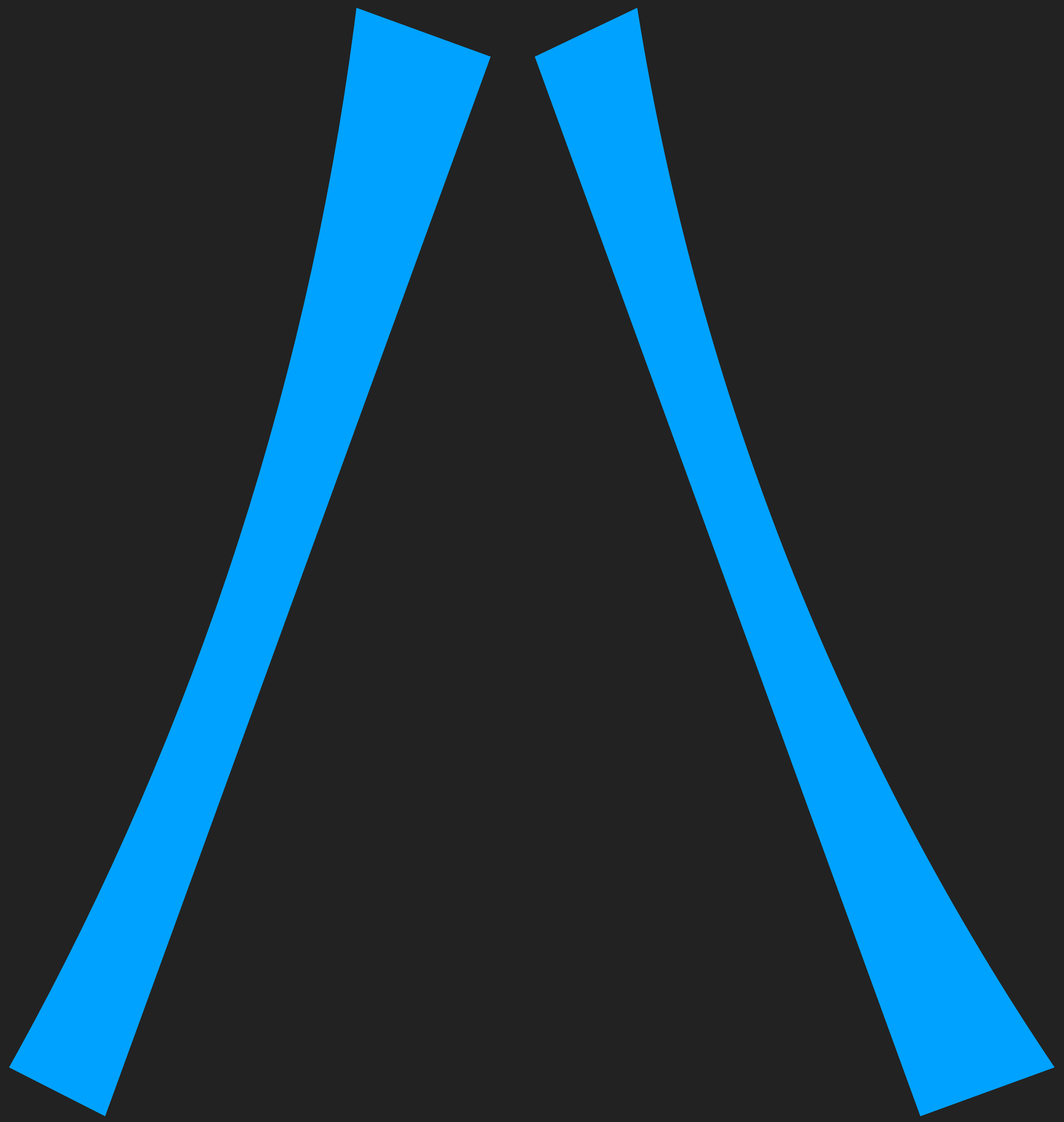
Serial Intro

Serial Workshop

What does this mean?

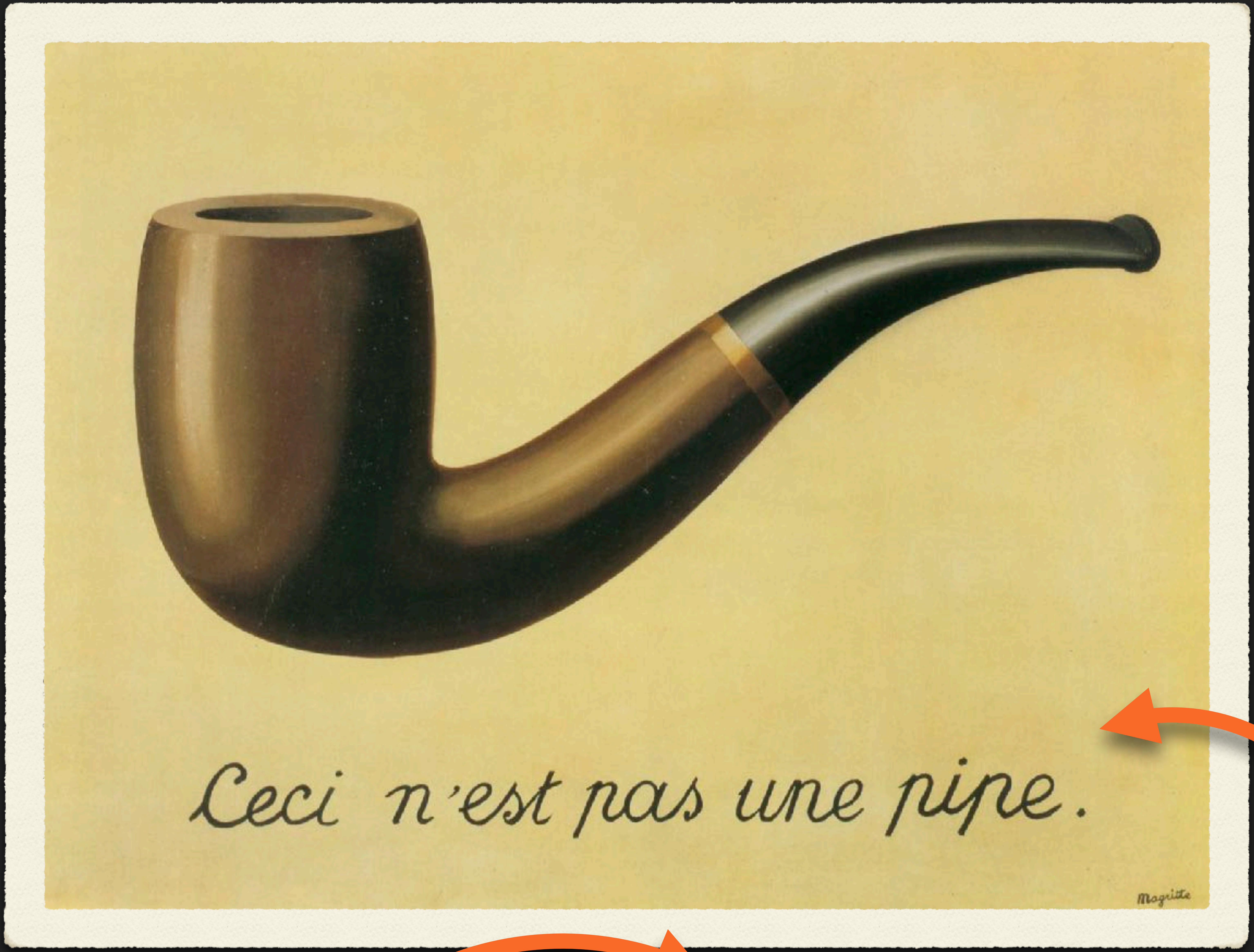


What does this mean?



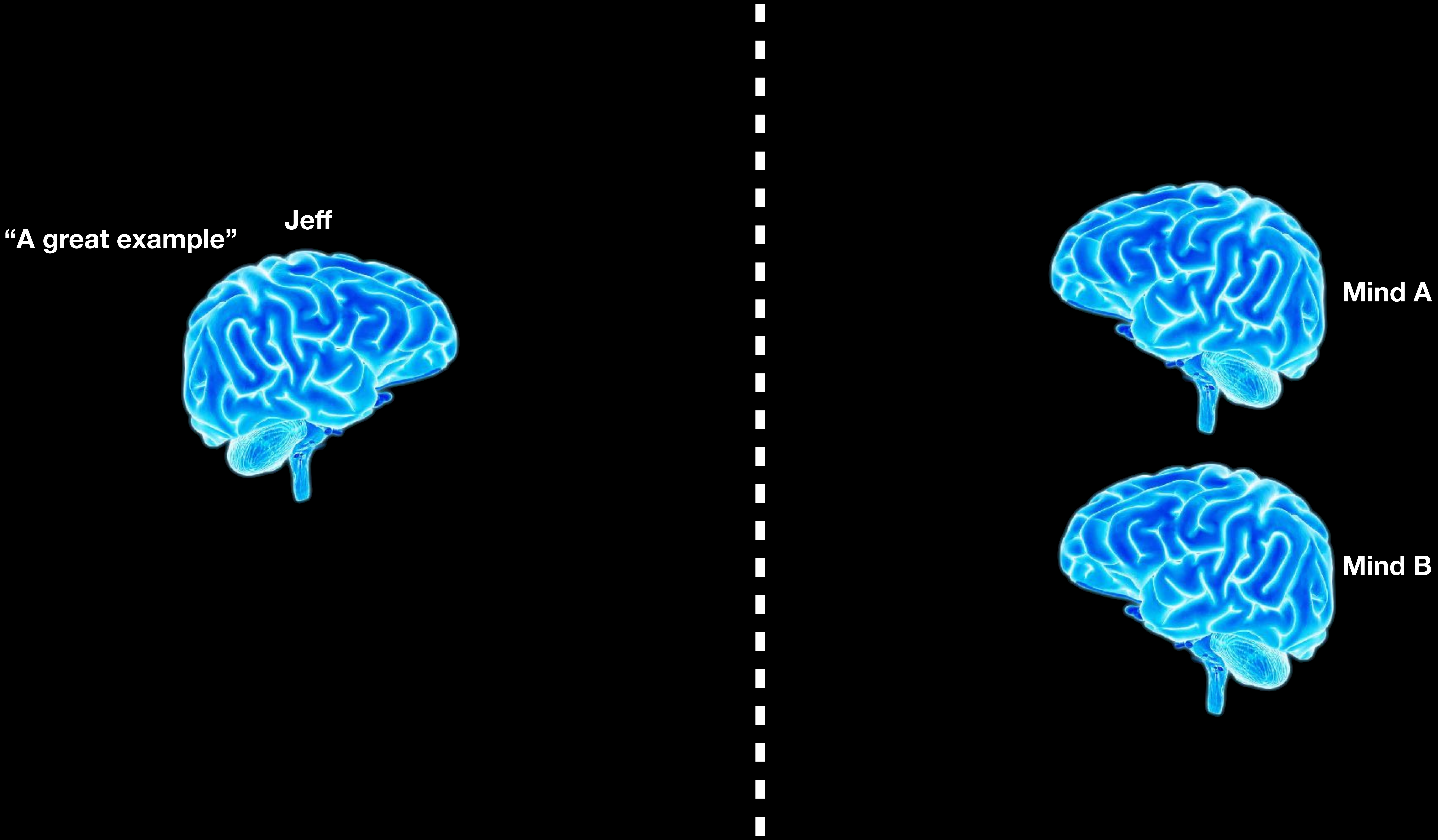
What does this mean?

人



This is not a painting by Magritte

This is not the idea that this is not a painting by Magritte…

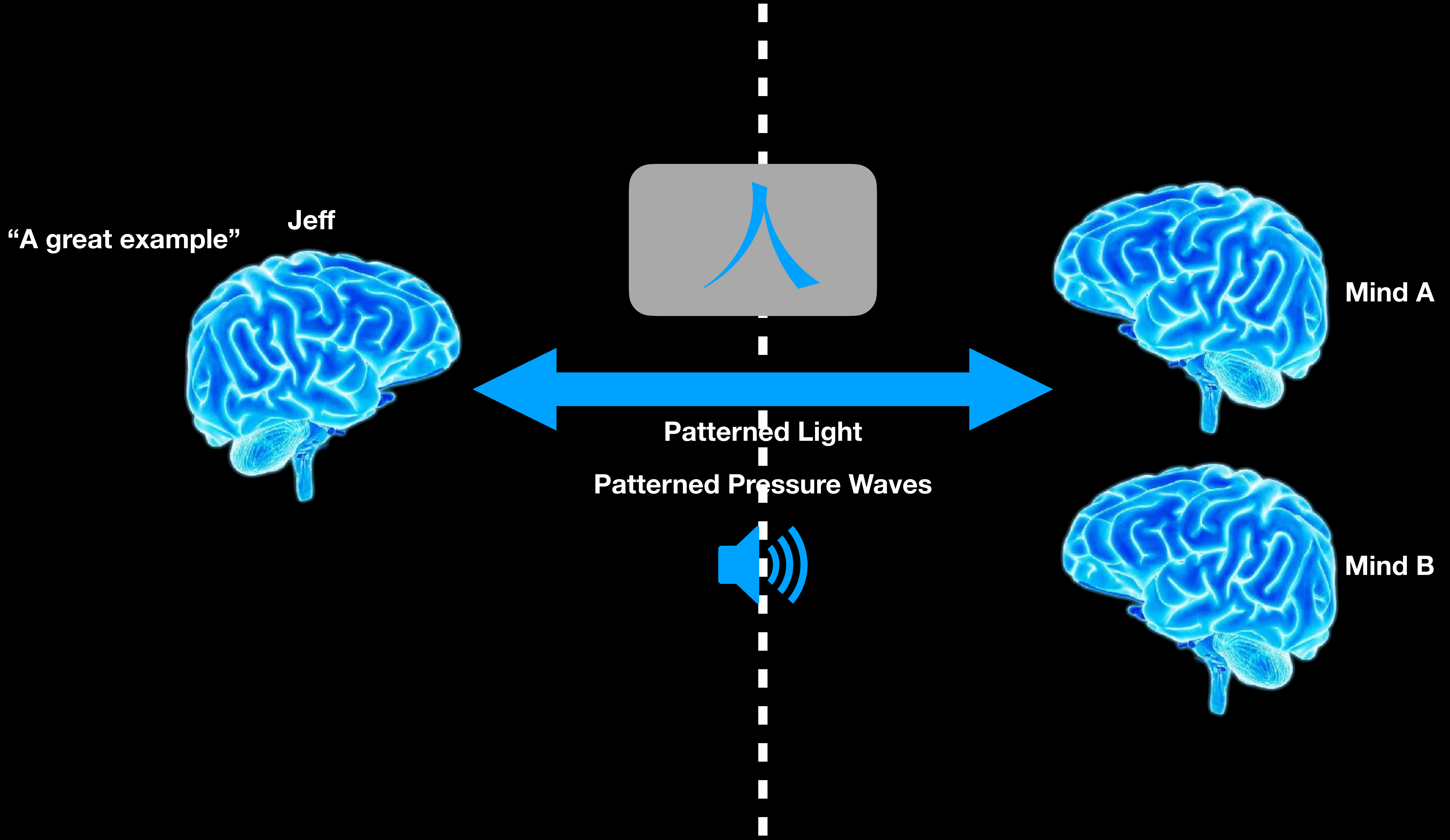


“A great example”

Jeff

Mind A

Mind B



“A great example”

Jeff

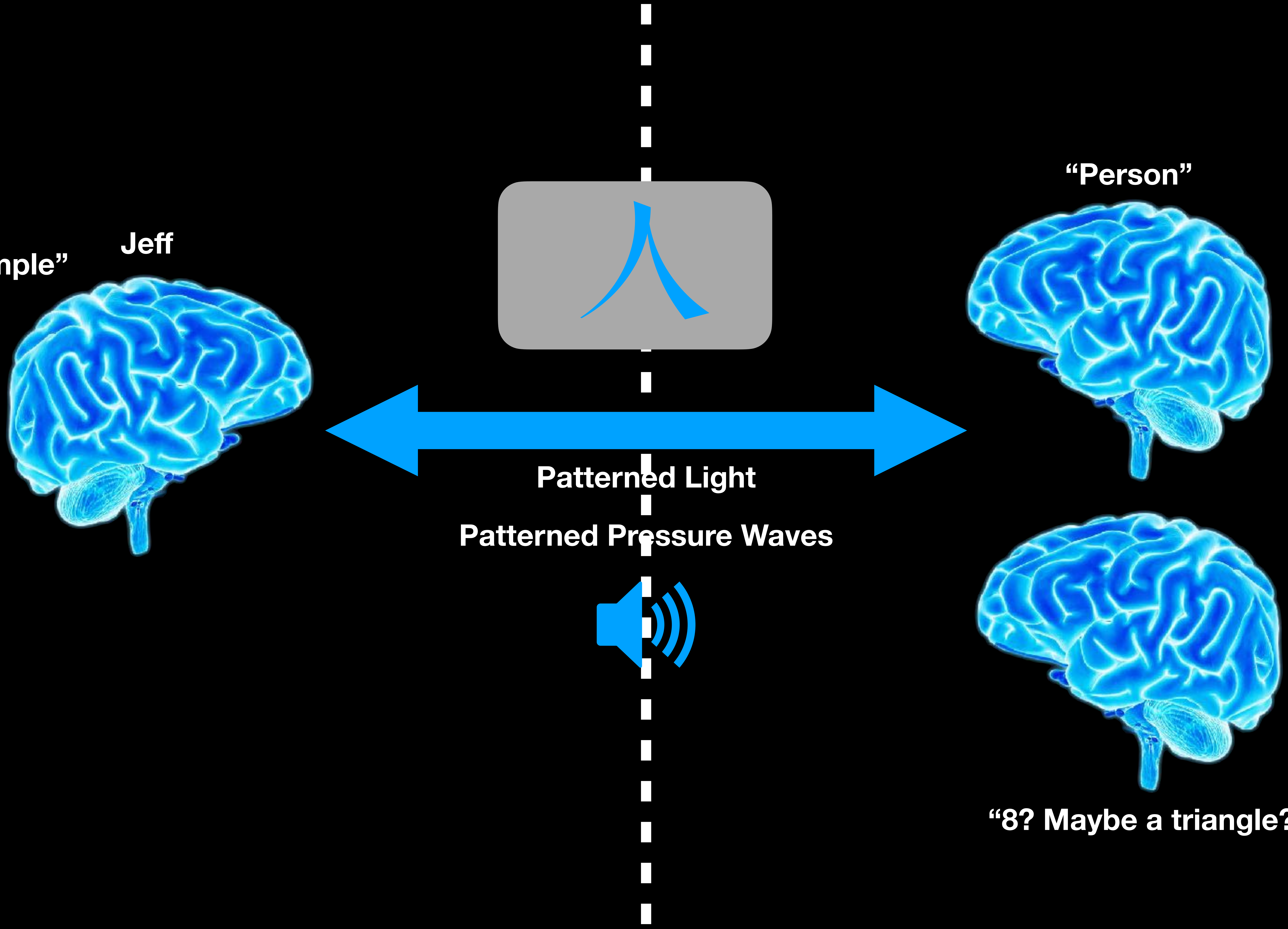
人

Patterned Light

Patterned Pressure Waves

Mind A

Mind B



"A great example"

Jeff

人

Patterned Light

Patterned Pressure Waves

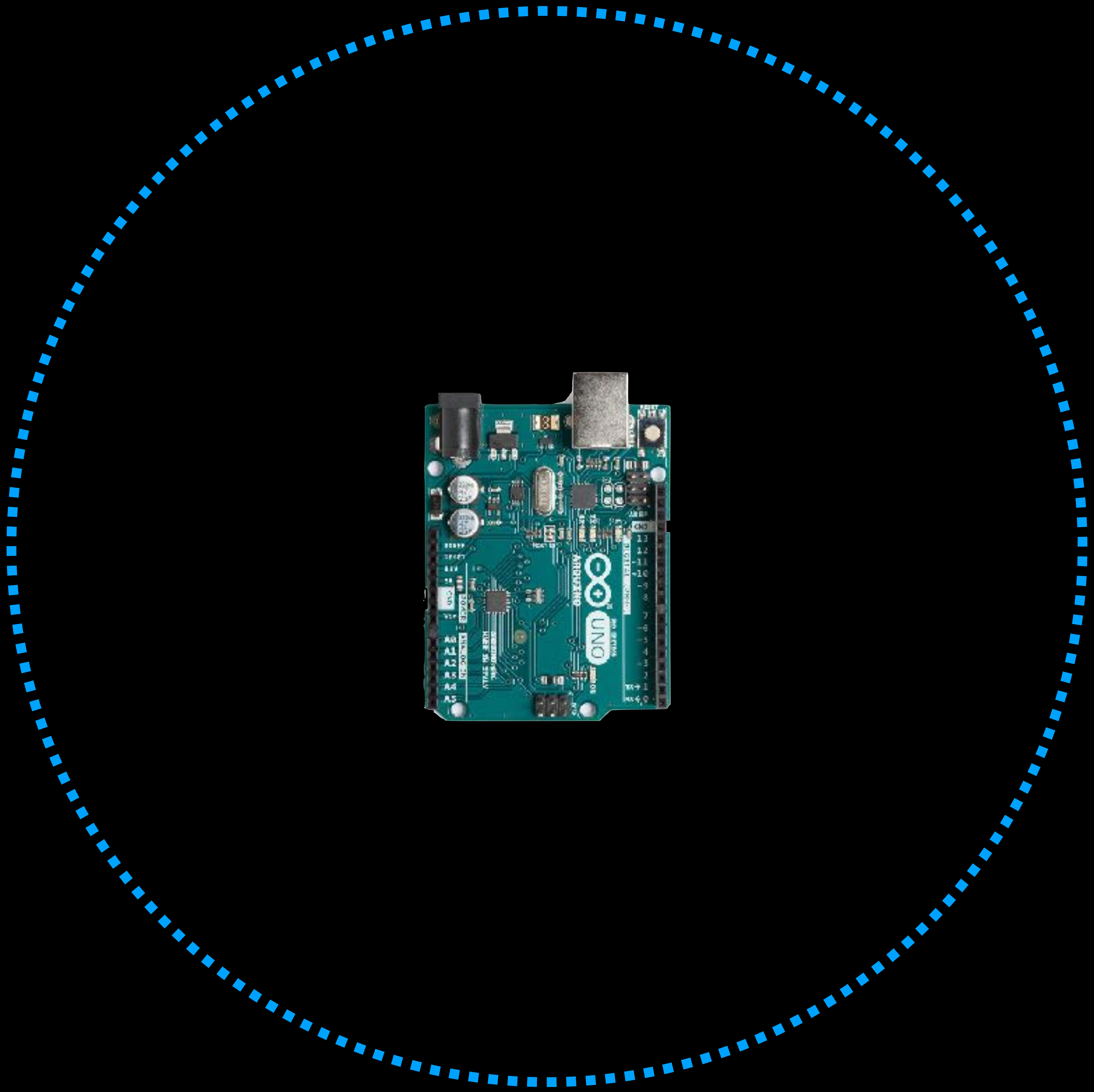
"Person"

Mind A

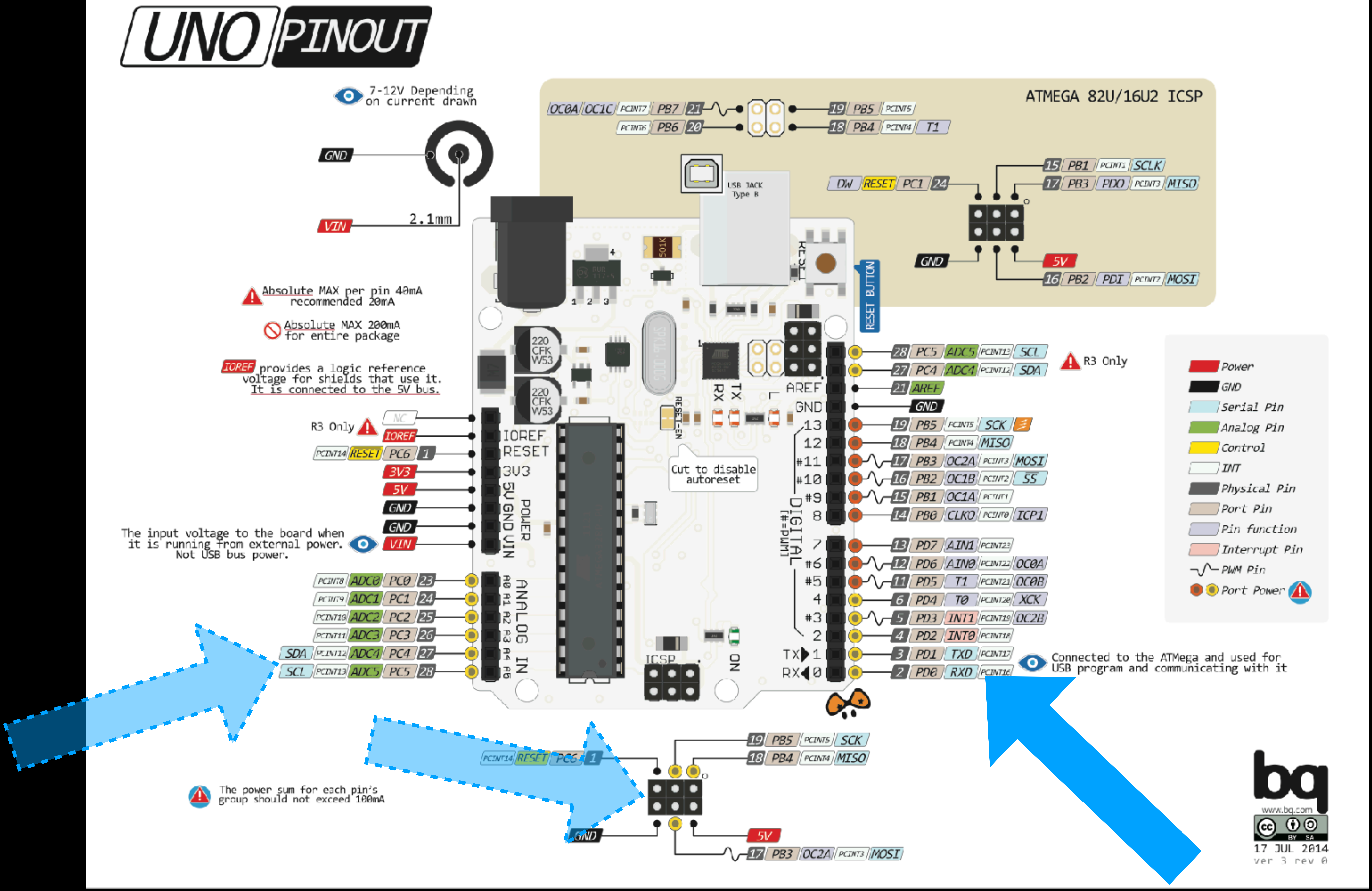
Mind B

"8? Maybe a triangle?"

Why Serial?



Arduino is lonely

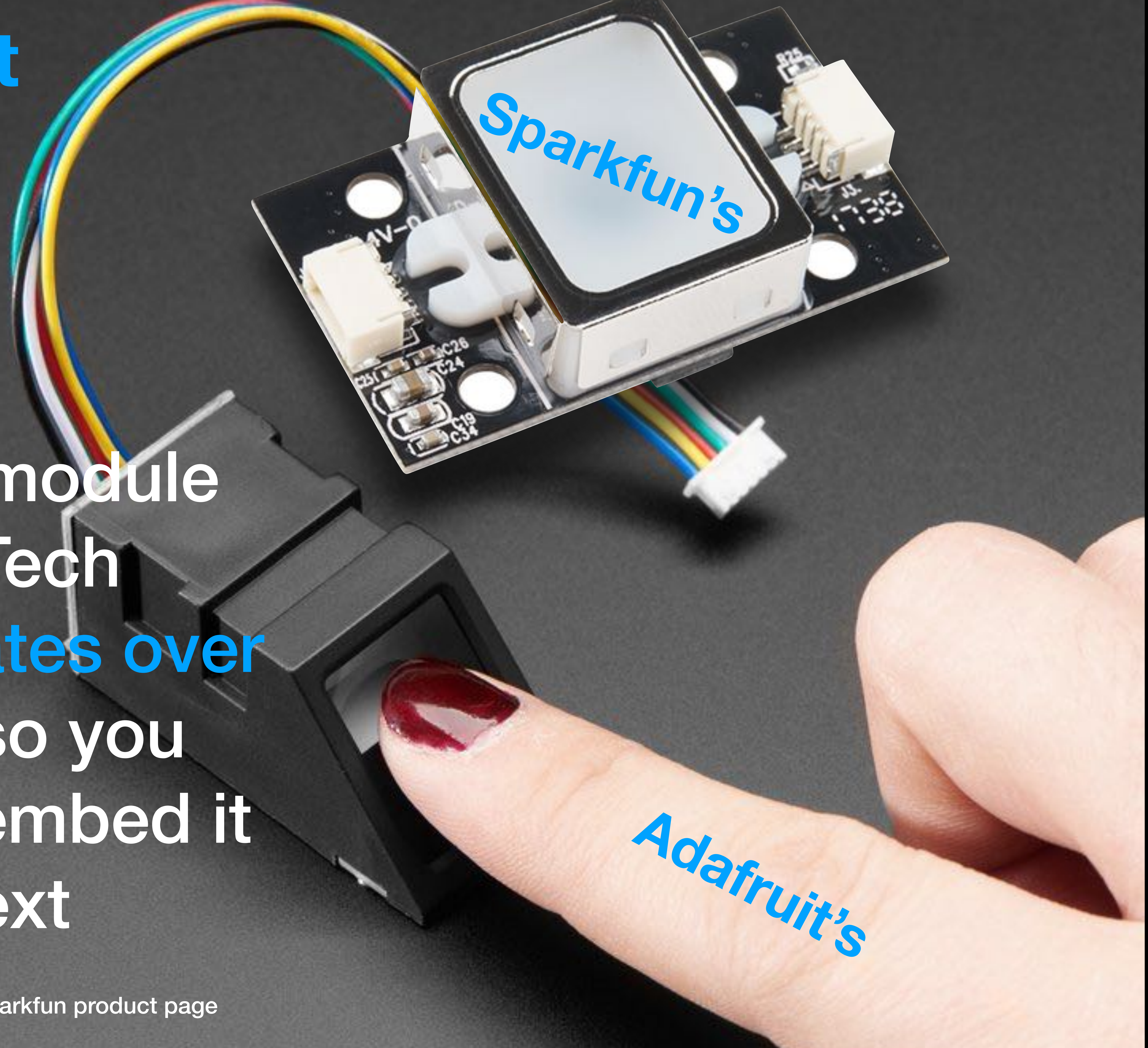


Arduino can talk to the world three ways p5*js

p5*js
node JS®
C++
Java™

Fingerprint Sensors

“This great GT-521F52 fingerprint module from ADH-Tech communicates over TTL Serial so you can easily embed it into your next project.” - Sparkfun product page



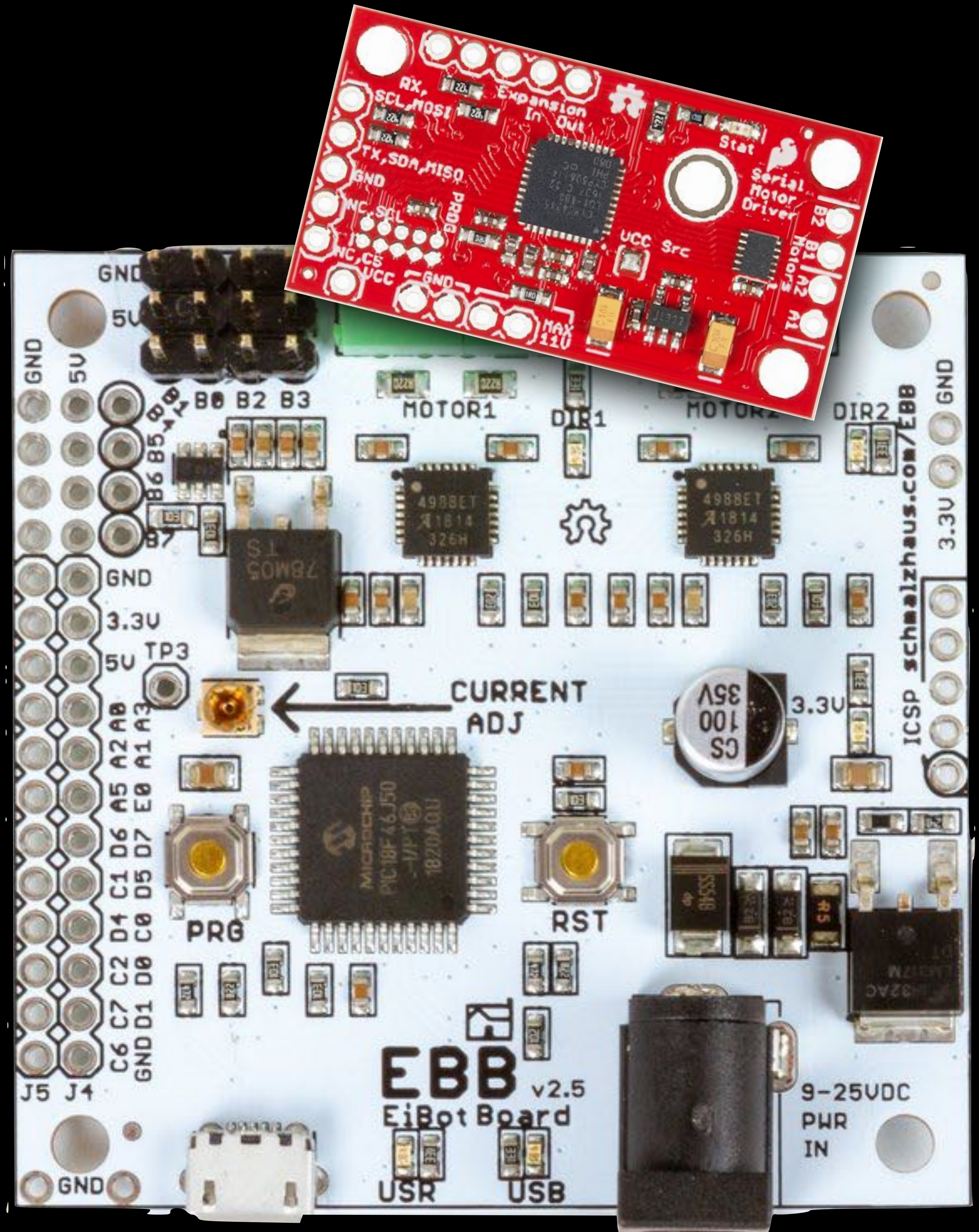
GSM + GPS

“On its own, this module can't do anything. It requires a microcontroller to drive it! We suggest and use an Arduino but any 3-5V microcontroller with a UART can send and receive commands over the RX/TX pins.”

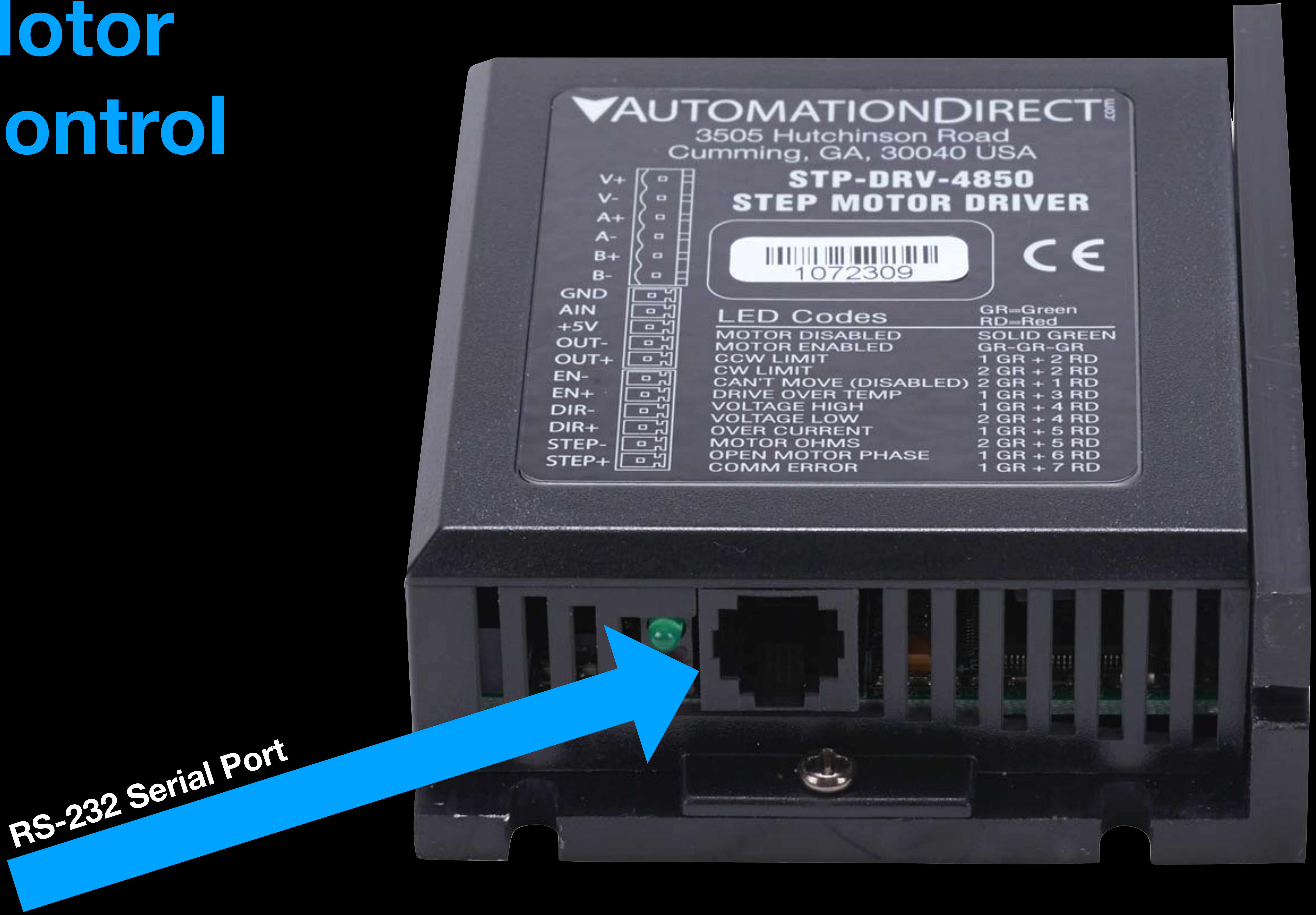
- Adafruit product page

Cheap Motor Control

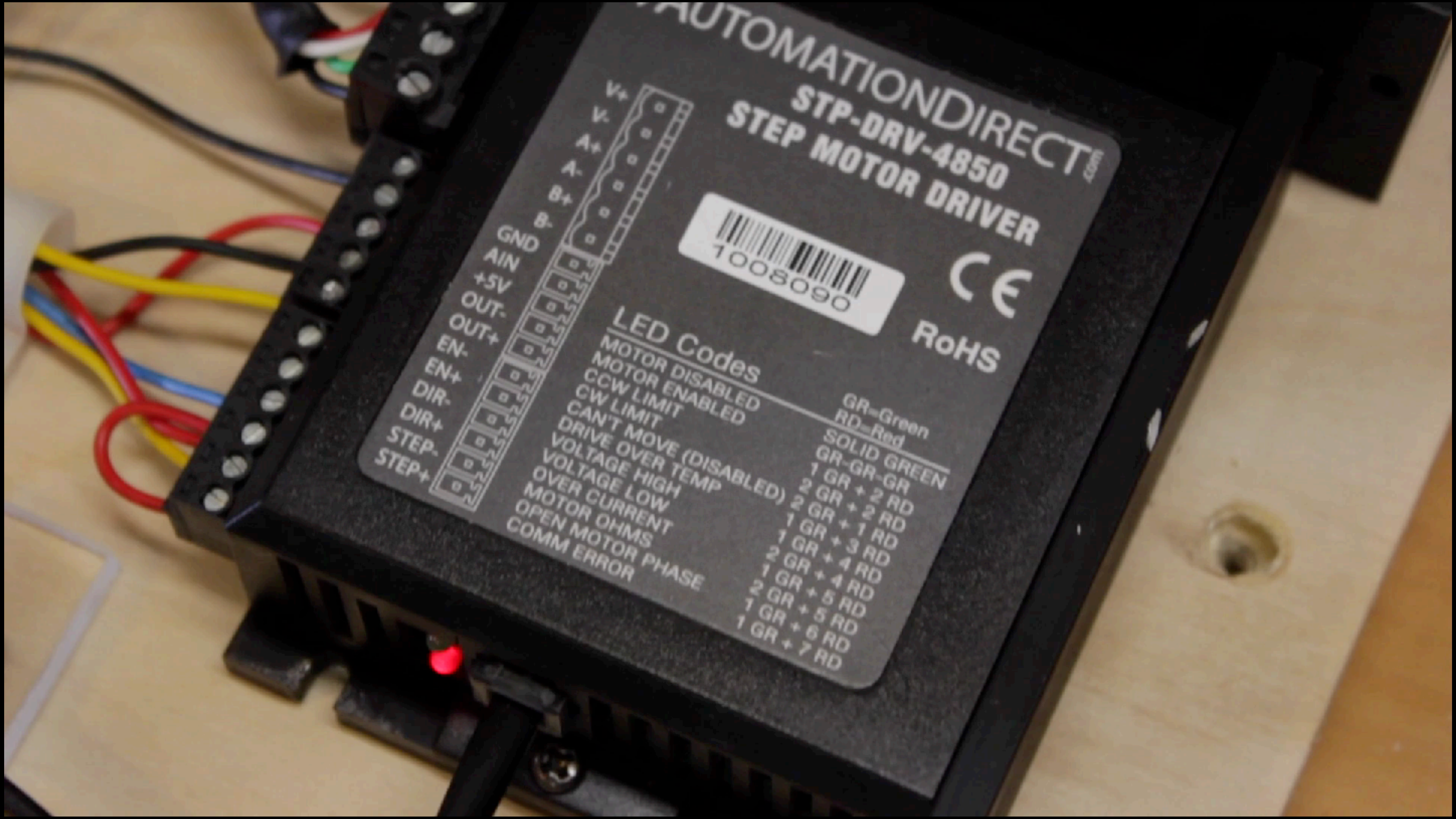
“You can easily type commands to it using a terminal emulator, or write your own application to send commands for moving the stepper motors.” - EEB Product page



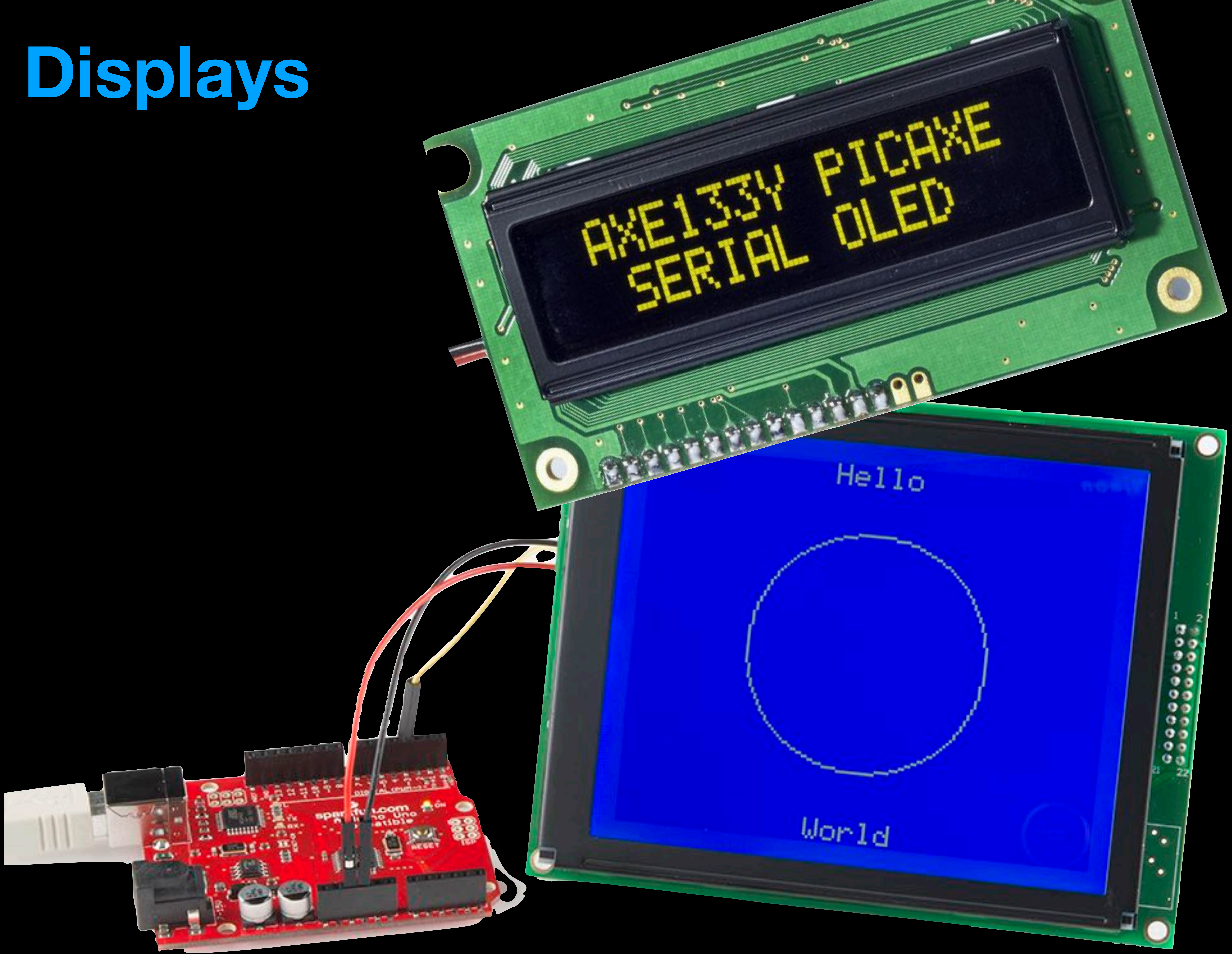
Mid-priced Motor Control



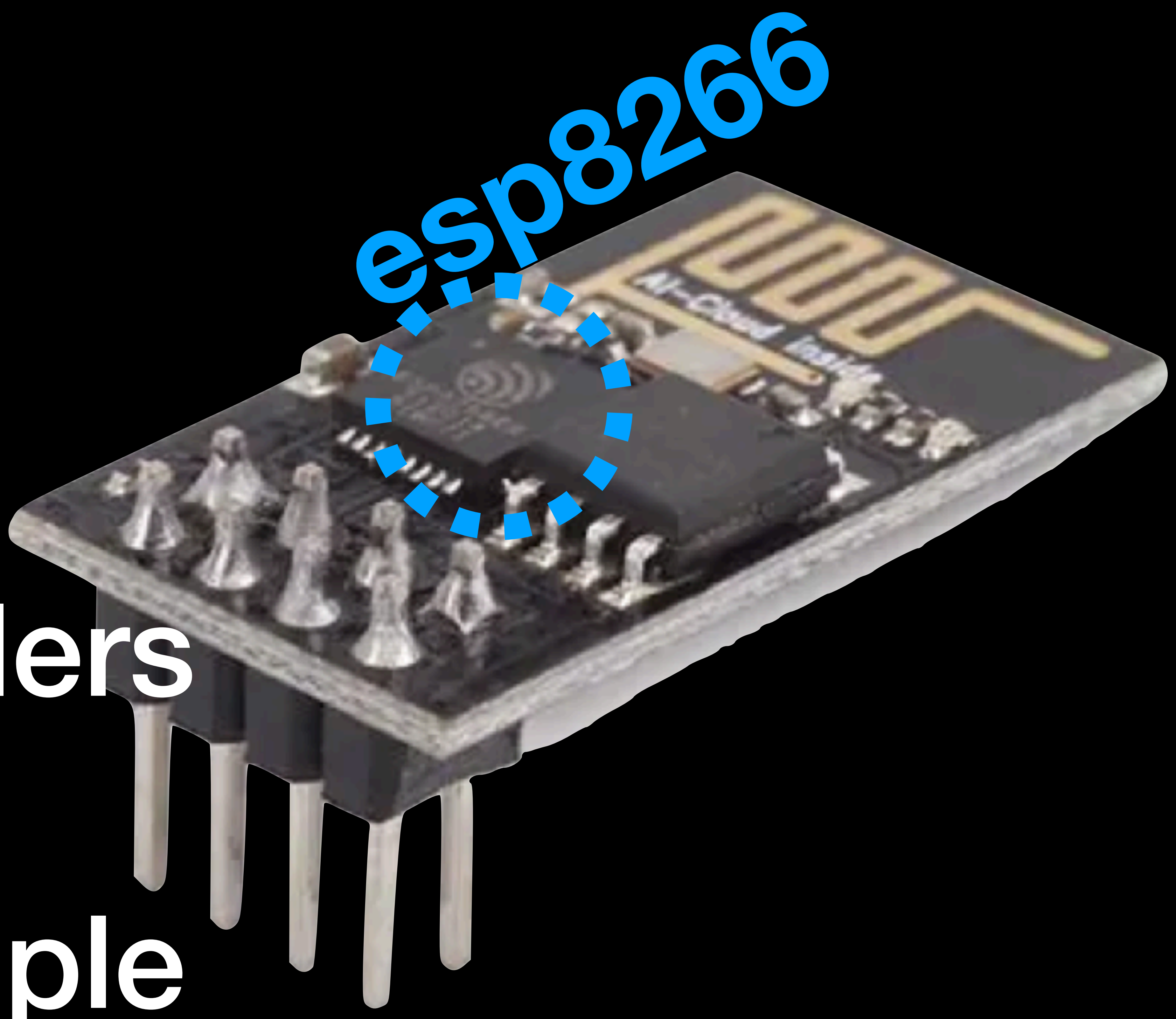
RS-232 Serial Port


AUTOMATIONDIRECT.com
STP-DRV-4850
STEP MOTOR DRIVER
1008090
CE
RoHS
V+
V-
A+
A-
B+
B-
GND
AIN
+5V
OUT-
OUT+
EN-
EN+
DIR-
DIR+
STEP-
STEP+
LED Codes
GR=Green
RD=Red
MOTOR DISABLED
MOTOR ENABLED
CCW LIMIT
CW LIMIT
CAN'T MOVE (DISABLED)
DRIVE OVER TEMP
VOLTAGE HIGH
VOLTAGE LOW
OVER CURRENT
MOTOR OHMS
OPEN MOTOR PHASE
COMM ERROR
SOLID GREEN
GR-GR-GREEN
1 GR-GR-GR
2 GR + 2 RD
2 GR + 2 RD
1 GR + 1 RD
1 GR + 3 RD
2 GR + 4 RD
1 GR + 4 RD
2 GR + 5 RD
1 GR + 5 RD
1 GR + 6 RD
1 GR + 7 RD

Displays



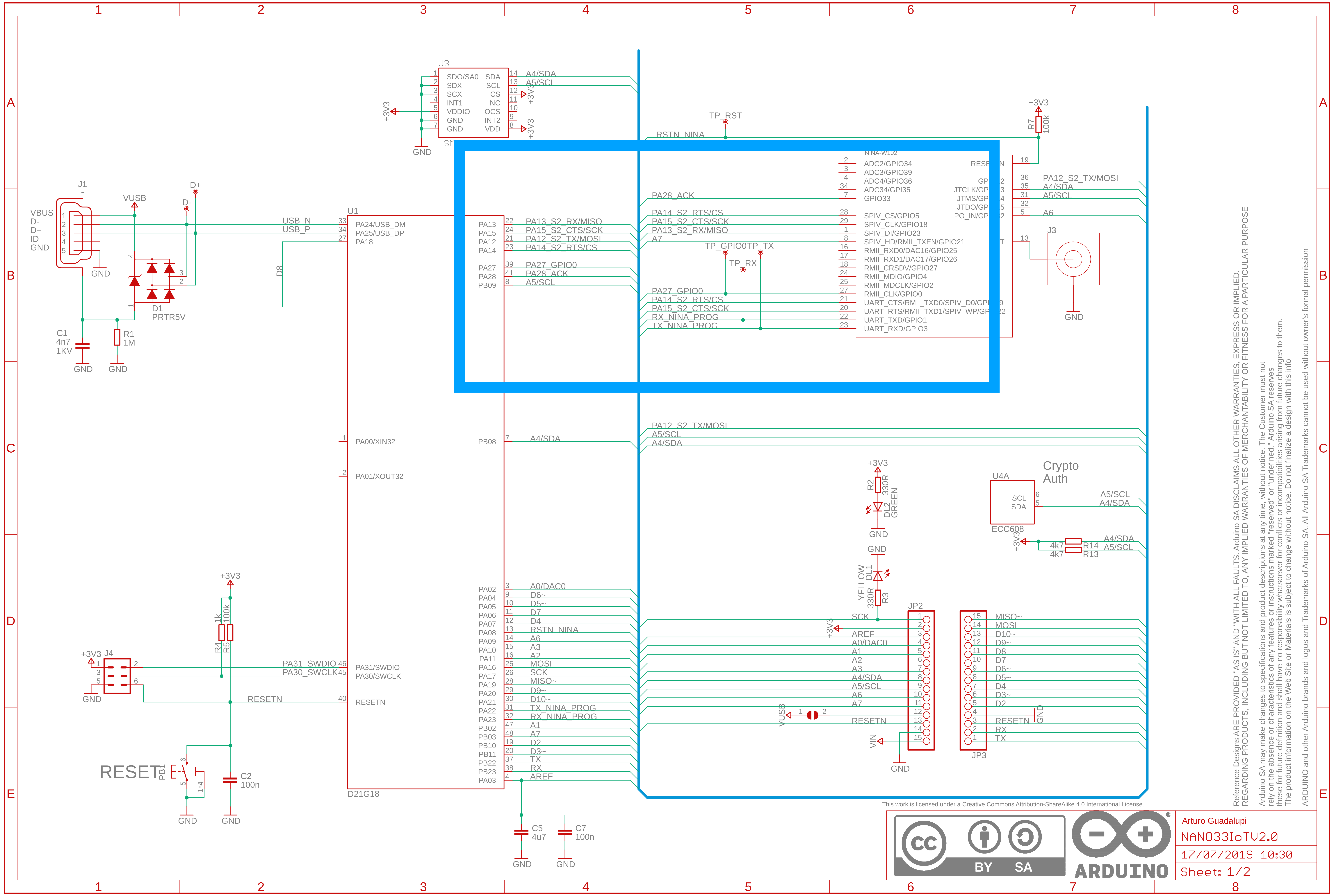
Wifi-serial circa 2014



“This small module allow[ed] microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style [serial] commands.” - Wikipedia

Wifi post 2018

“In late October 2014, Espressif Systems released a software development kit (SDK) for programming the chip directly, which removed the need for a separate microcontroller.” - Wikipedia



This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

Arturo Guadalupi

NANO33IoTV2.0

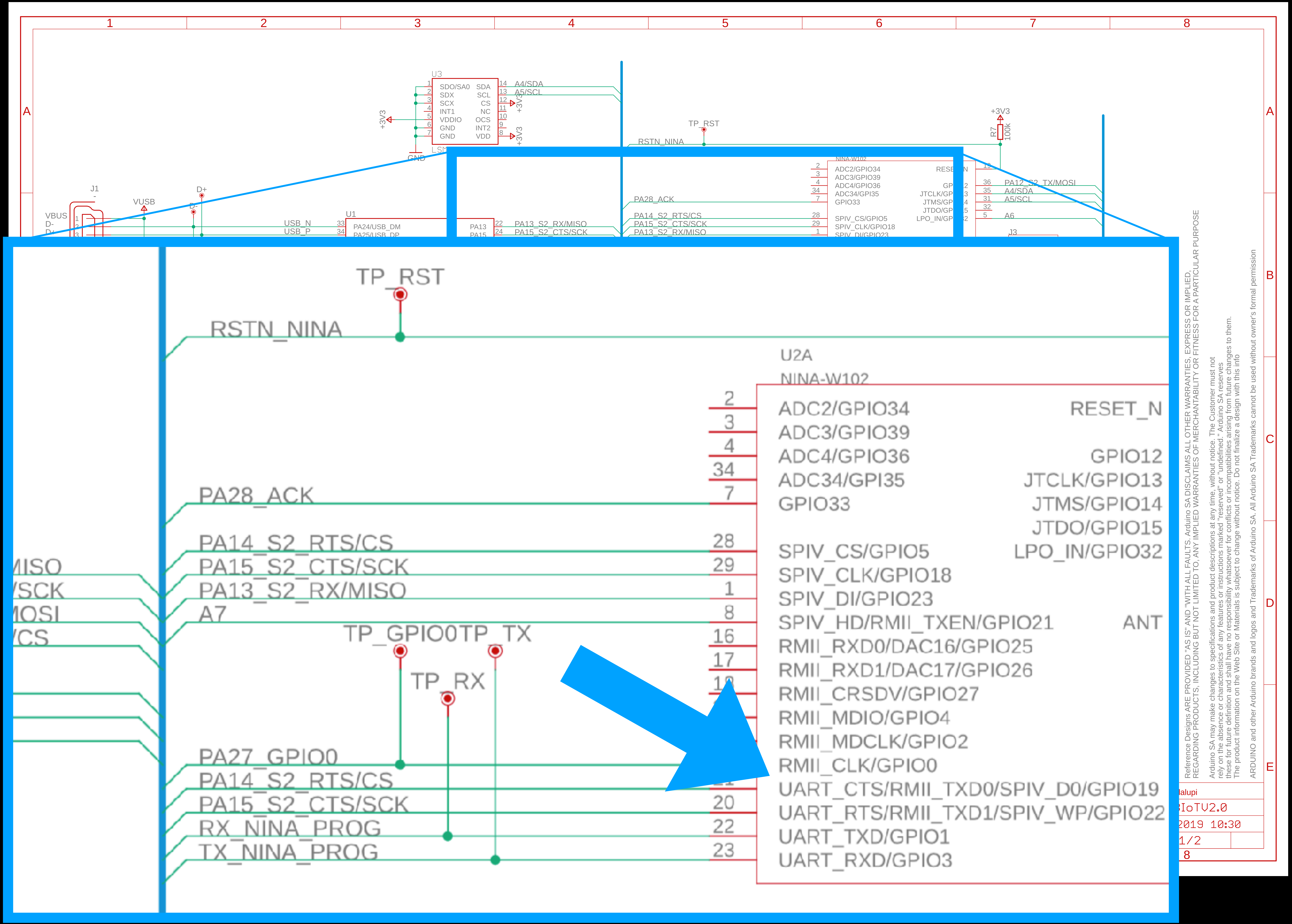
17/07/2019 10:30

Sheet: 1/2

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino SA DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

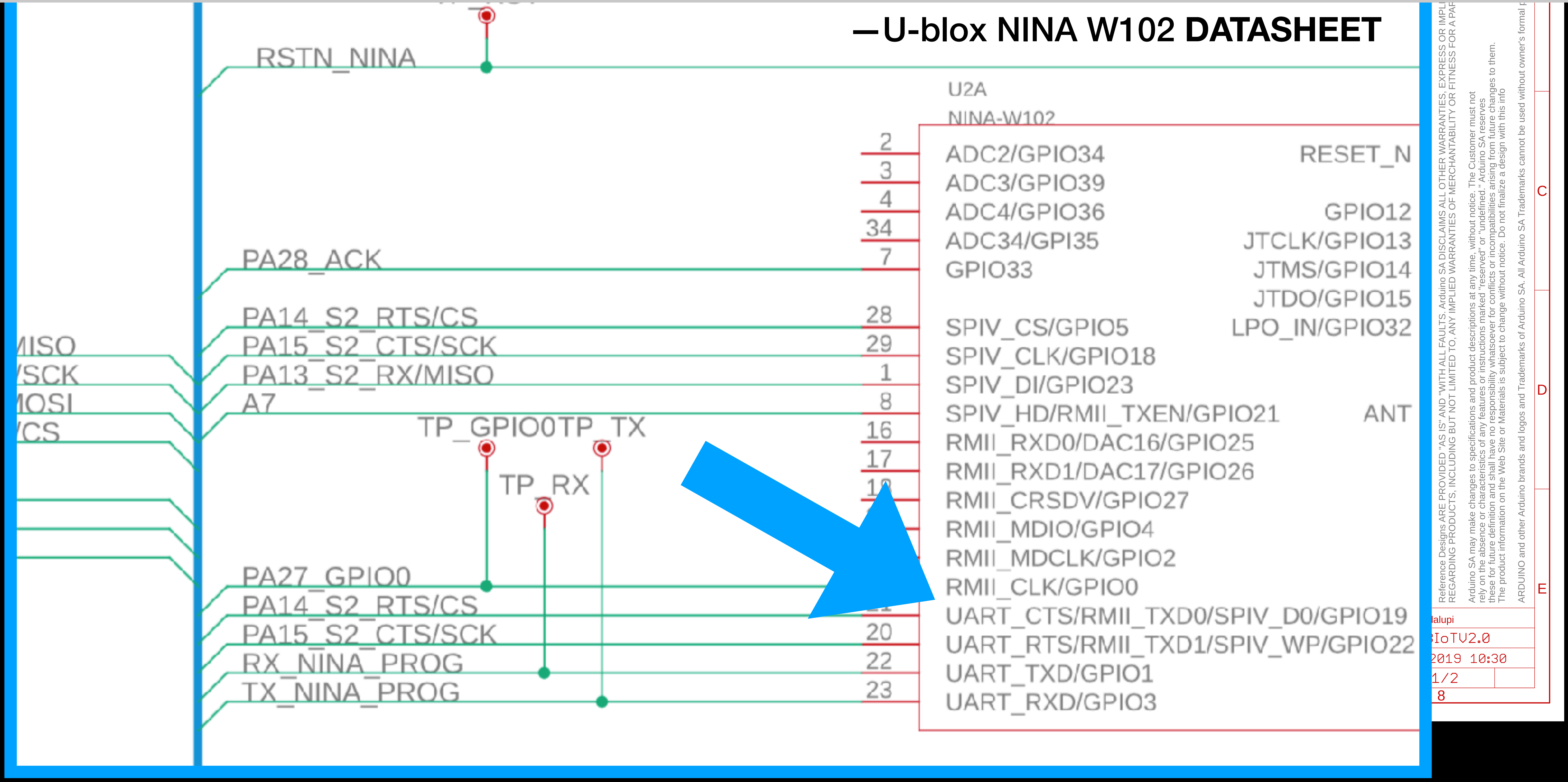
Arduino SA may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino SA reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this info

ARDUINO and other Arduino brands and logos and Trademarks of Arduino SA. All Arduino SA Trademarks cannot be used without owner's formal permission

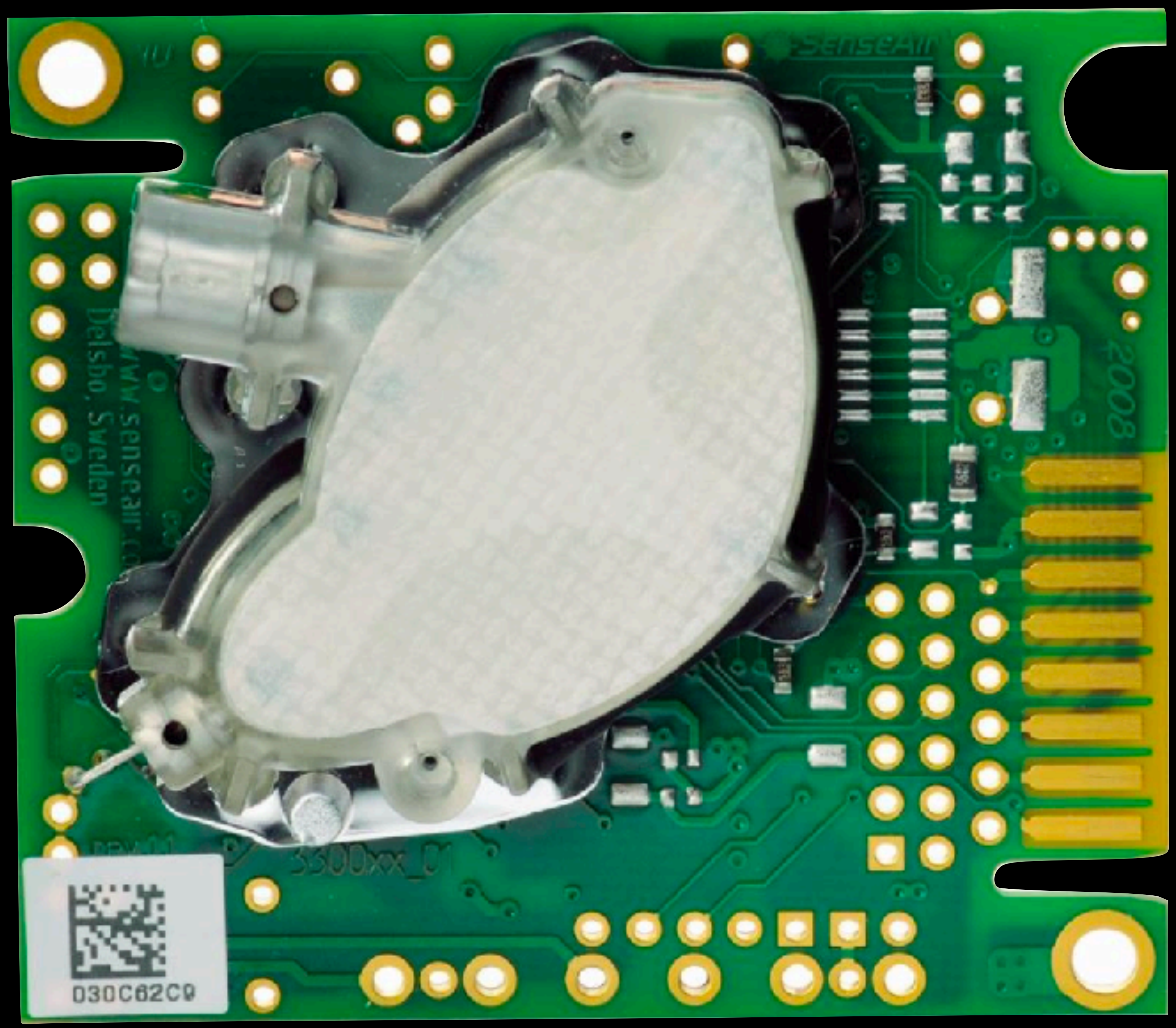

TP_RST
RSTN_NINA
U2A
NINA-W102
2 ADC2/GPIO34
3 ADC3/GPIO39
4 ADC4/GPIO36
34 ADC34/GPI35
7 GPIO33
RESET_N
GPIO12
JTCLK/GPIO13
JTMS/GPIO14
JTDO/GPIO15
LPO_IN/GPIO32
PA28_ACK
PA14_S2_RTS/CS
PA15_S2_CTS/SCK
PA13_S2_RX/MISO
A7
28 SPIV_CS/GPIO5
29 SPIV_CLK/GPIO18
1 SPIV_DI/GPIO23
8 SPIV_HD/RMII_TXEN/GPIO21
ANT
16 RMII_RXD0/DAC16/GPIO25
17 RMII_RXD1/DAC17/GPIO26
RMII_CRSDV/GPIO27
RMII_MDIO/GPIO4
RMII_MDCLK/GPIO2
RMII_CLK/GPIO0
UART_CTS/RMII_TXD0/SPIV_D0/GPIO19
20 UART_RTS/RMII_TXD1/SPIV_WP/GPIO22
22 UART_TXD/GPIO1
23 UART_RXD/GPIO3
TP_GPIO0
TP_TX
TP_RX
PA27_GPIO0
PA14_S2_RTS/CS
PA15_S2_CTS/SCK
RX_NINA_PROG
TX_NINA_PROG
MISO
SCK
MOSI
CS
U3
SDO/SA0
SDX
SCX
INT1
VDDIO
GND
SDA
SCL
CS
NC
OCS
INT2
VDD
A4/SDA
A5/SCL
+3V3
GND
J1
VBUS
D-
D+
VUSB
U1
USB_N
USB_P
PA24/USB_DM
PA25/USB_DP
PA13
PA15
PA13_S2_RX/MISO
PA15_S2_CTS/SCK
R7
100k
PA12_S2_TX/MOSI
A6
J3
1 2 3 4 5 6 7 8
A B C D E
IoTV2.0
2019 10:30
1/2
Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino SA DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE
Arduino SA may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino SA reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this info
ARDUINO and other Arduino brands and logos and Trademarks of Arduino SA. All Arduino SA Trademarks cannot be used without owner's formal permission

2.7.1 UARTs NINA-W10 modules have three UART interfaces, UART0 to UART2. Each interface provides asynchronous communication support for RS232, RS485, and IrDA standards (with external drivers). UART0 serves as the primary interface port. The maximum speed for all UART interfaces is 5 Mbit/s.

—U-blox NINA W102 **DATASHEET**



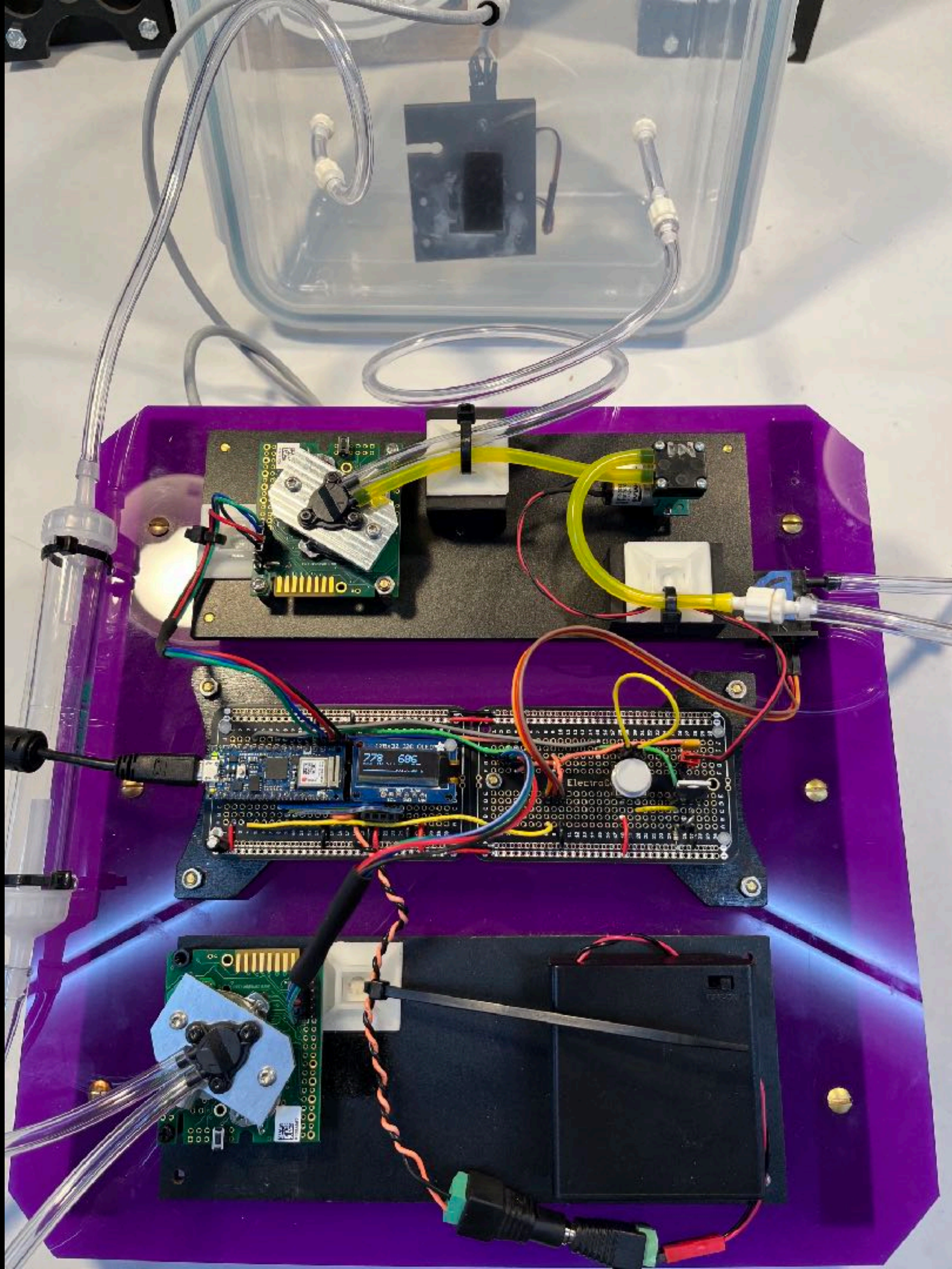
CO2 Sensor

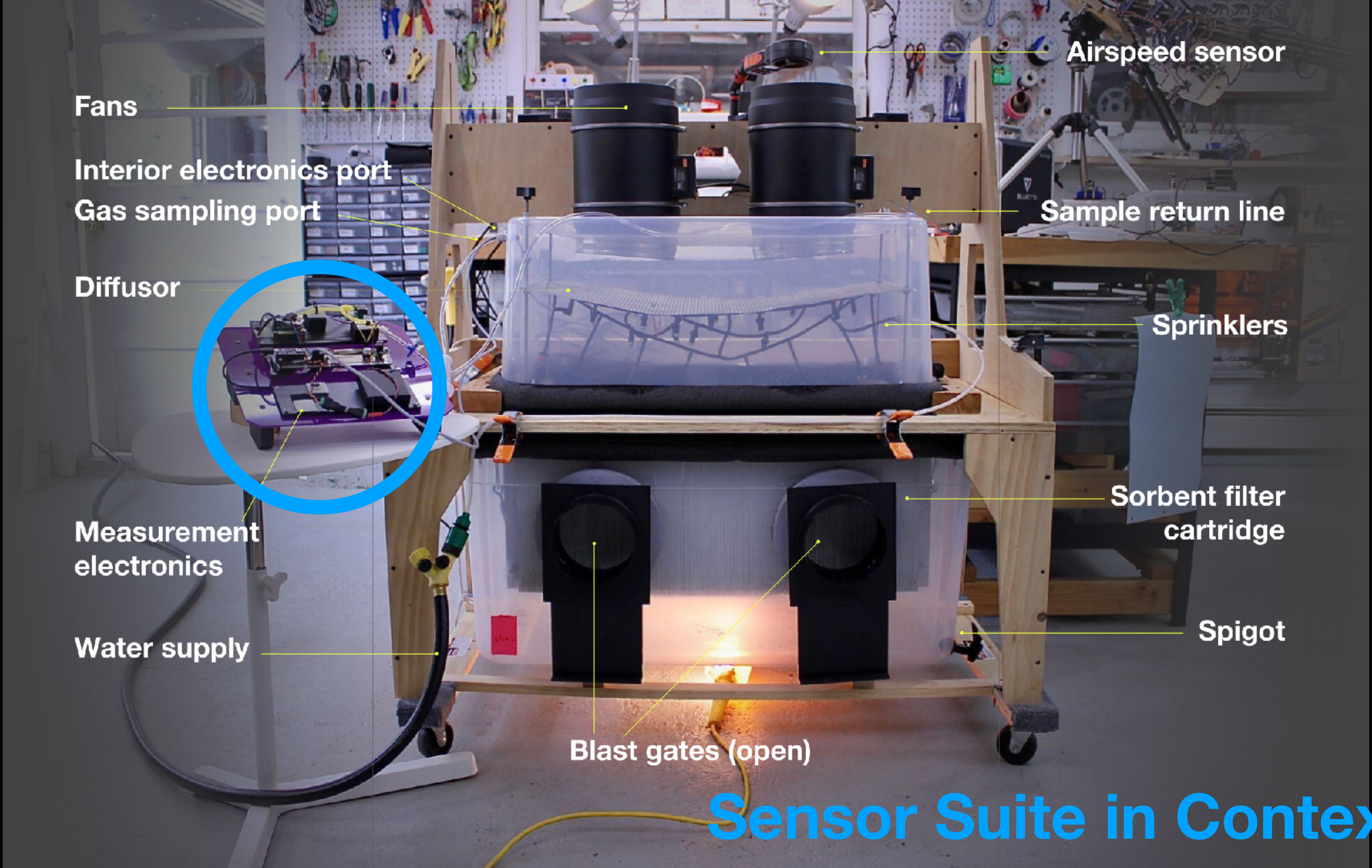


UART Serial, I2C *and* analog outputs

CO2 Sensor Suite:

2x serial to K30
1x serial to computer
WIFI to cloud (option)
I2C to sensors and display





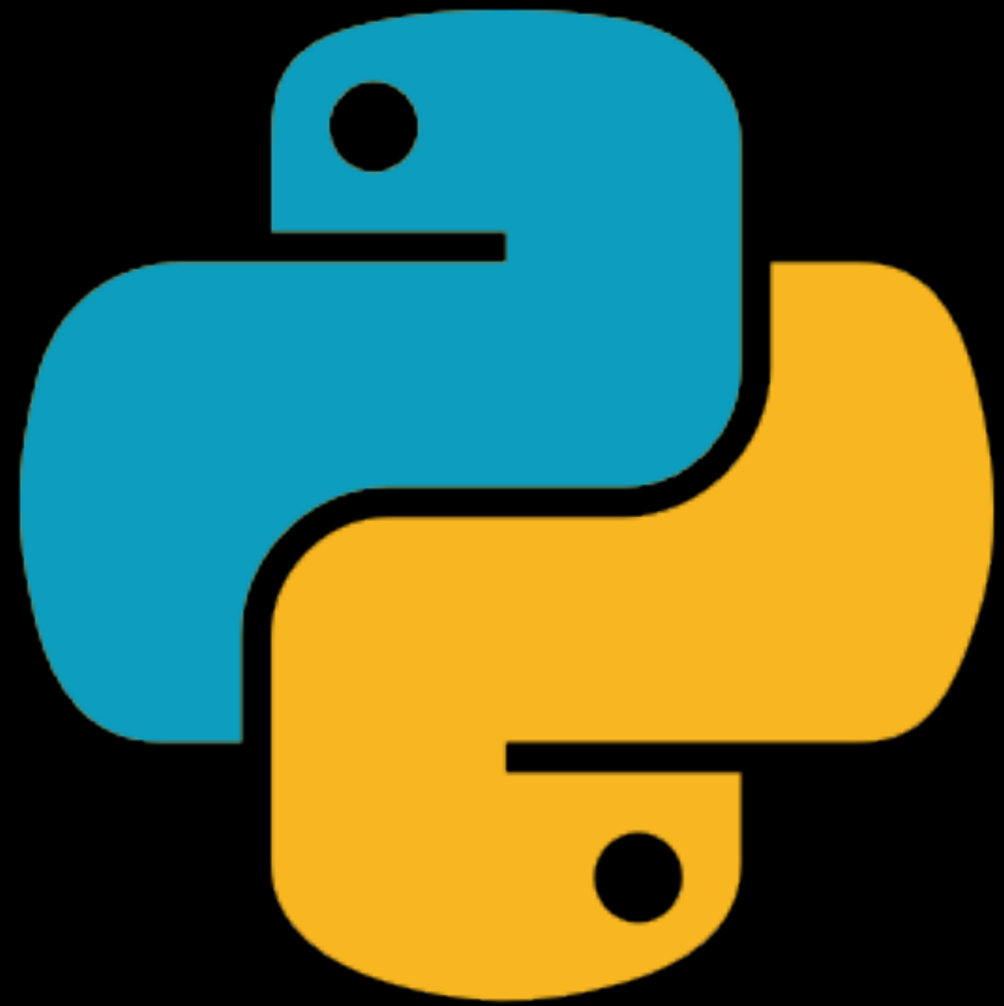
Sensor Suite in Context

p5*js

node
JS®

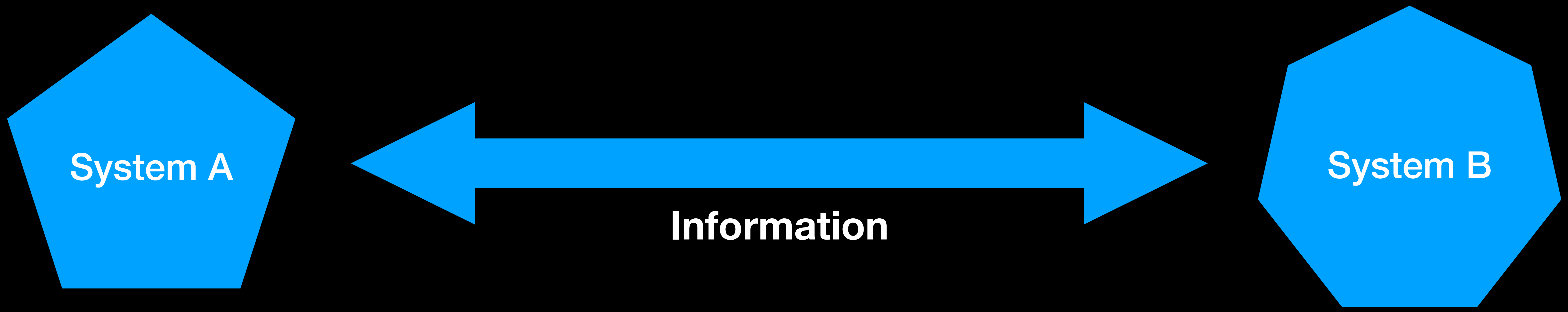
C++

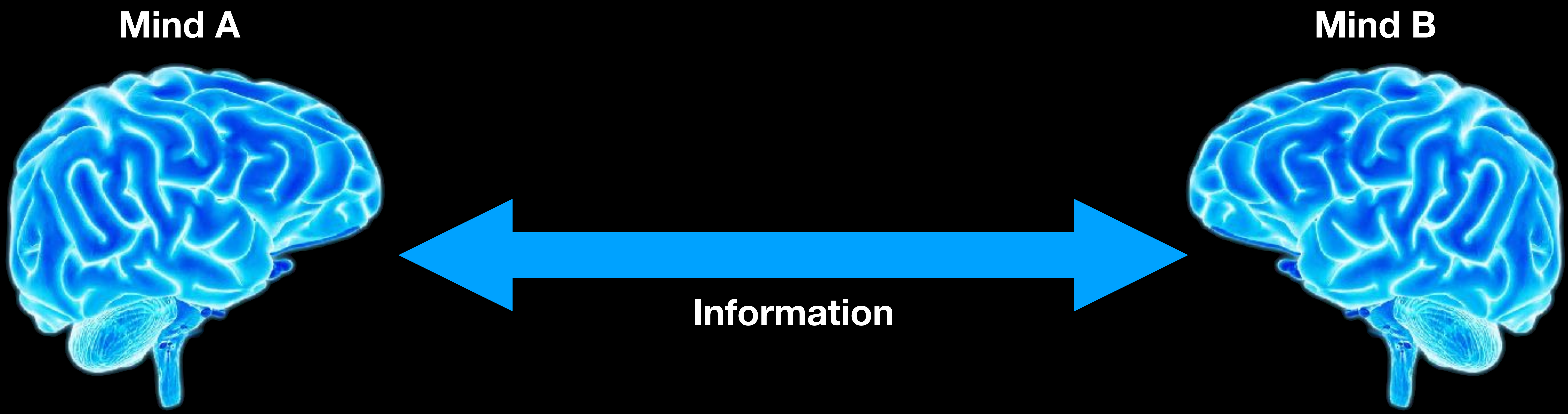
Java™



etc...

etc...

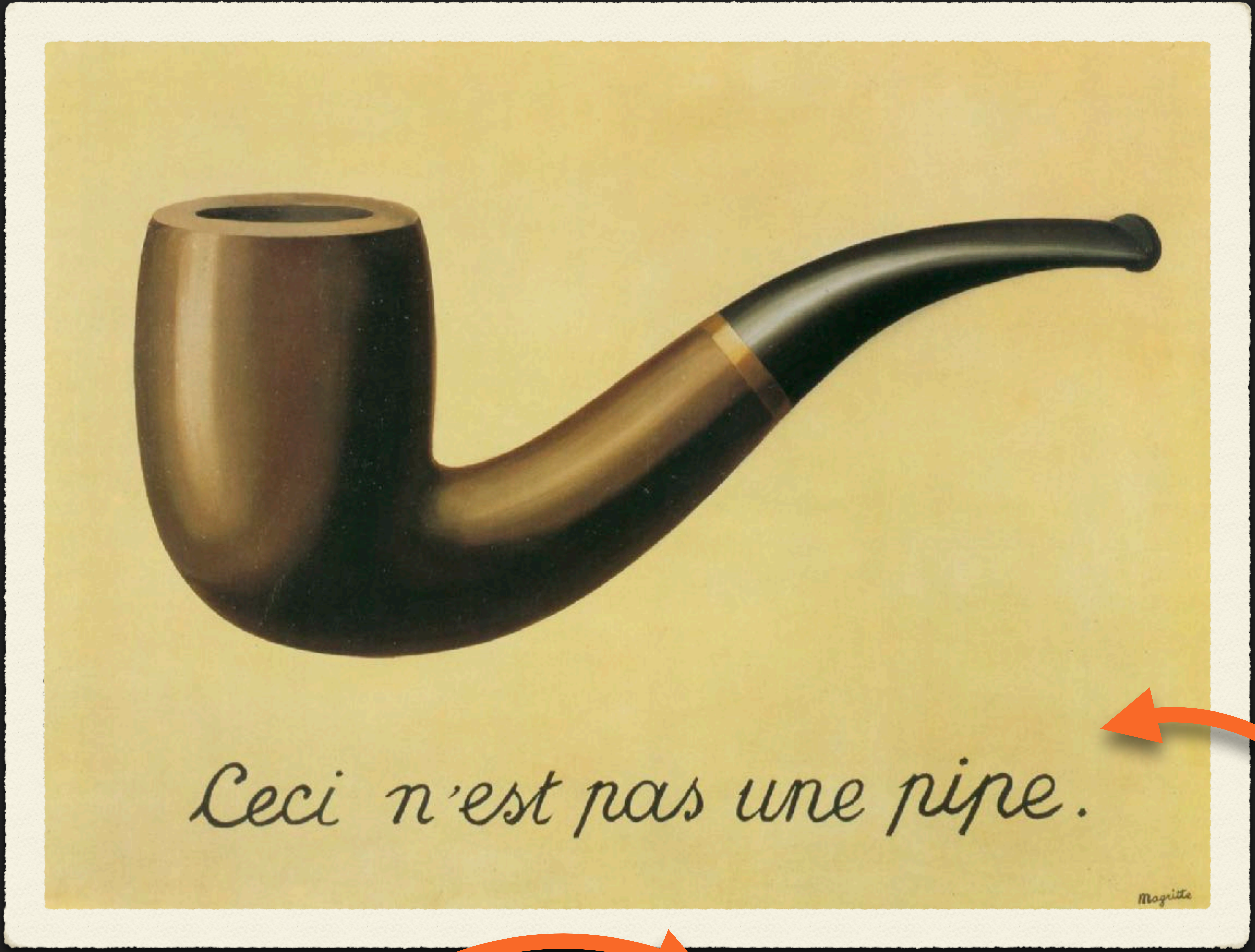

System A
Information
System B



Mind A

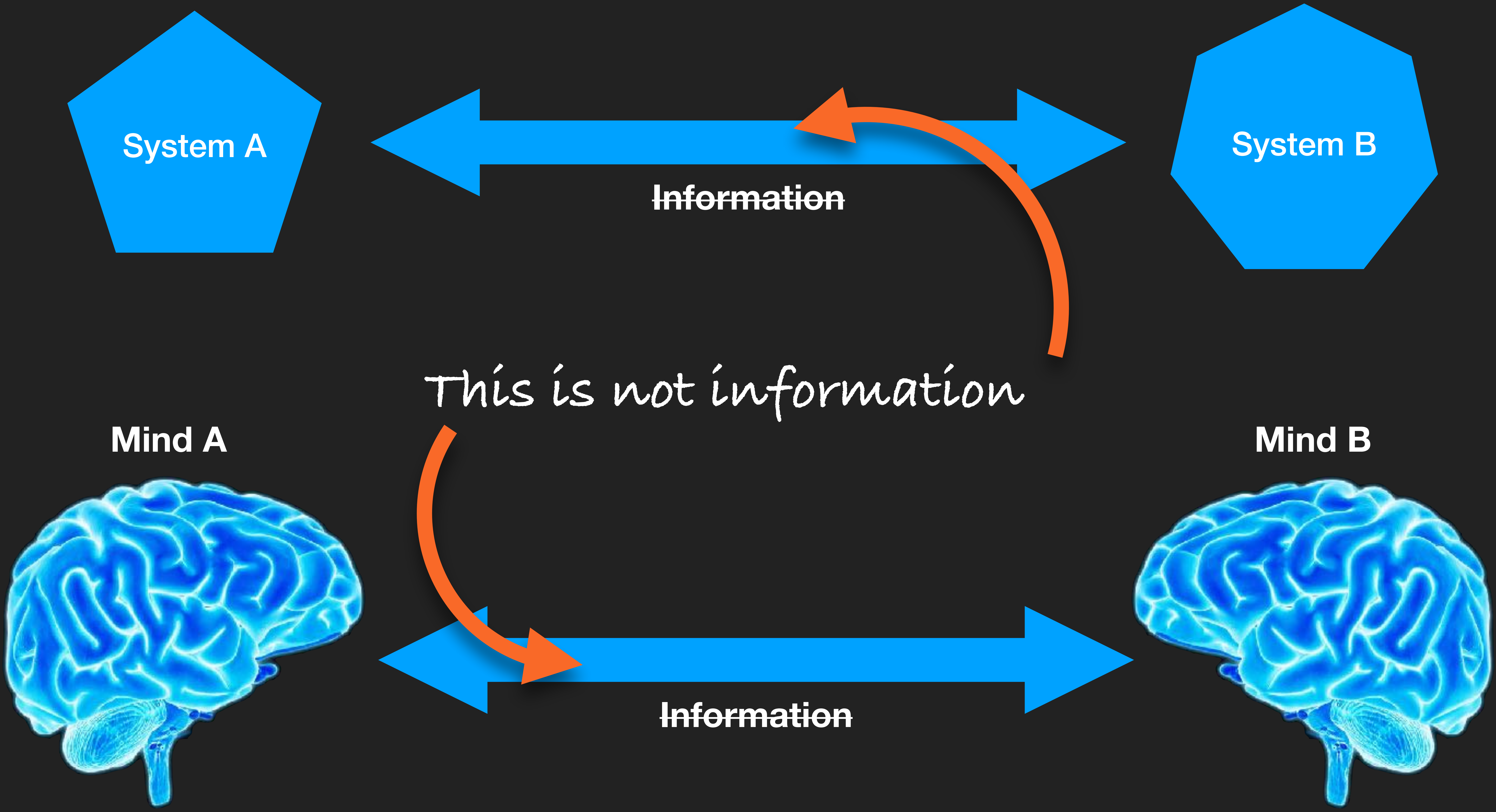
Mind B

Information



This is not a painting by Magritte

This is not the idea that this is not a painting by Magritte…



System A

System B

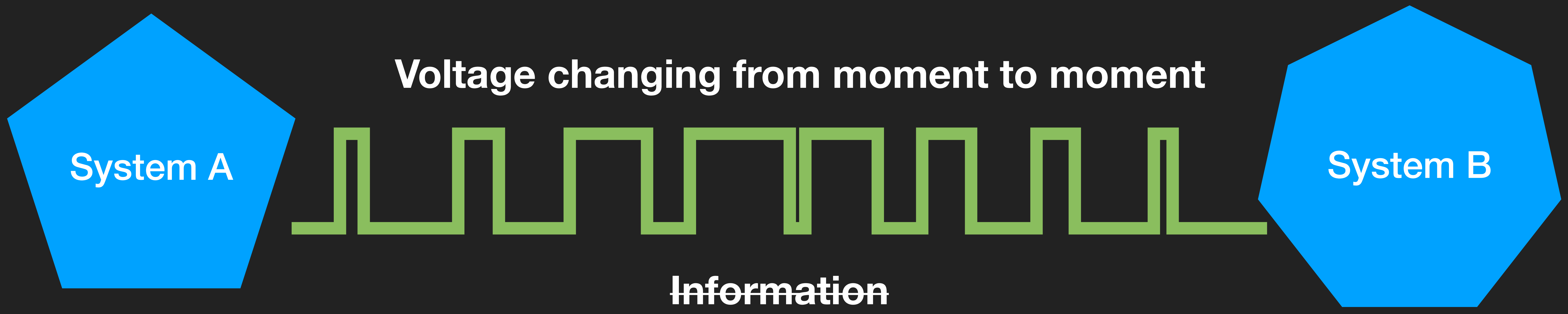
~~Information~~

This is not information

Mind A

Mind B

~~Information~~



System A

Voltage changing from moment to moment

~~Information~~

System B

Variable declaration in code:

```
byte b = 64;
```

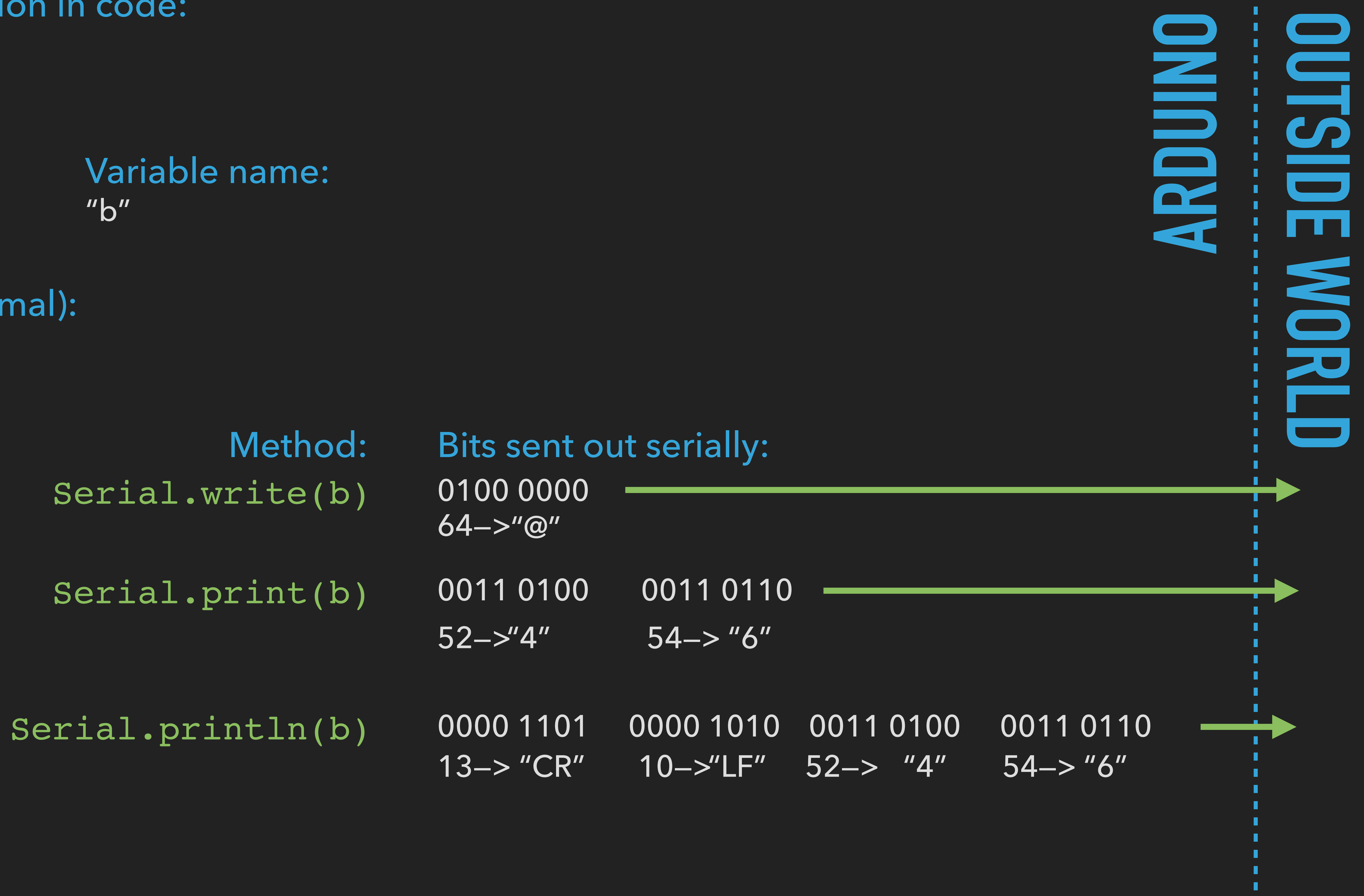
Bits in memory:
0100 0000

Variable name:
“b”

Value to us (decimal):

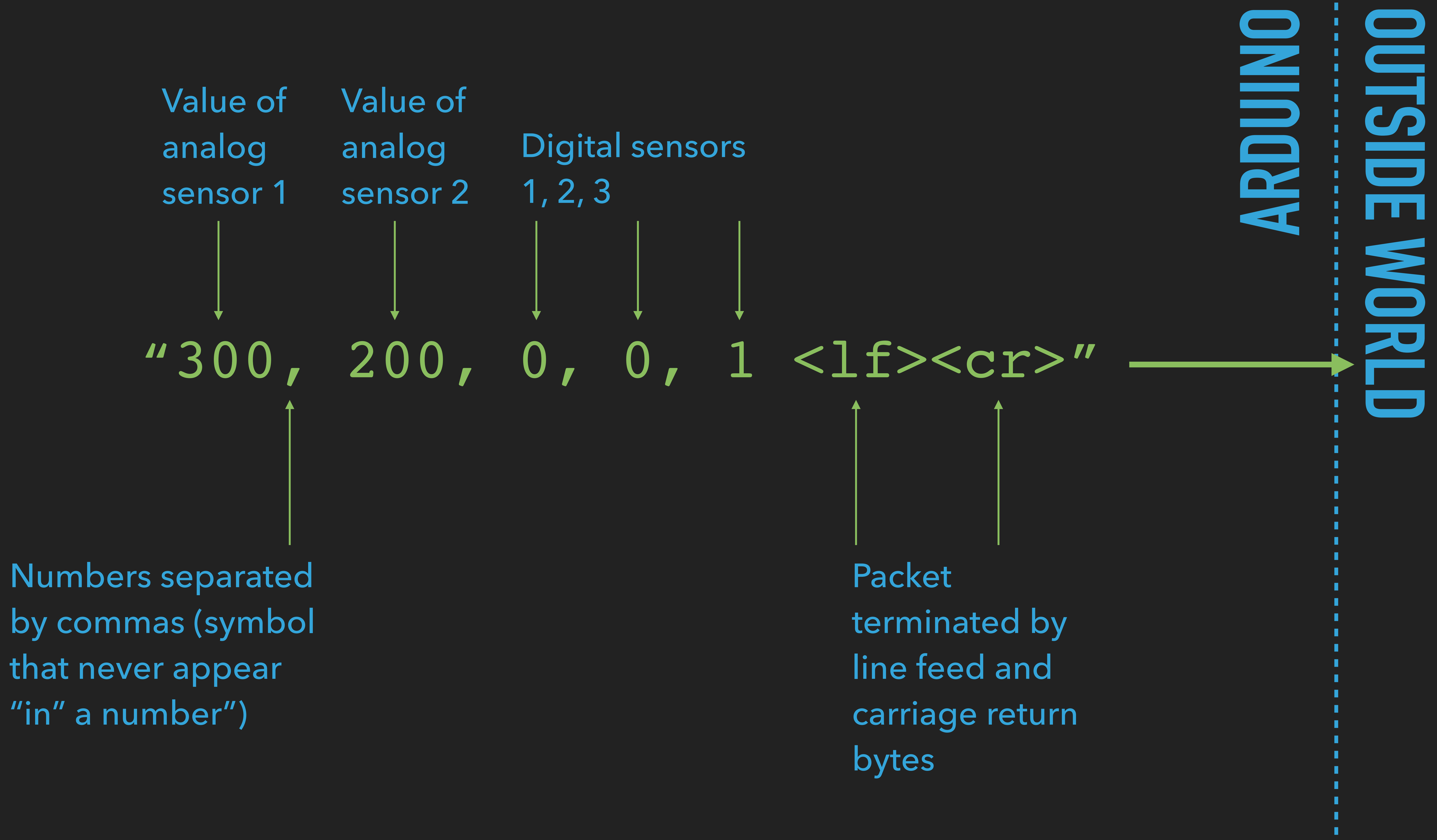
“64”

ASCII symbol
“@”



Method:	Bits sent out serially:			
`Serial.write(b)`	0100 0000 64–>“@”			
`Serial.print(b)`	0011 0100 52–>“4”	0011 0110 54–> “6”		
`Serial.println(b)`	0000 1101 13–> “CR”	0000 1010 10–>“LF”	0011 0100 52–> “4”	0011 0110 54–> “6”

Sender generates data “packet” as a string



Value of analog sensor 1

Value of analog sensor 2

Digital sensors 1, 2, 3

```
“300, 200, 0, 0, 1 <lf><cr>”
```

Numbers separated by commas (symbol that never appear “in” a number”)

Packet terminated by line feed and carriage return bytes

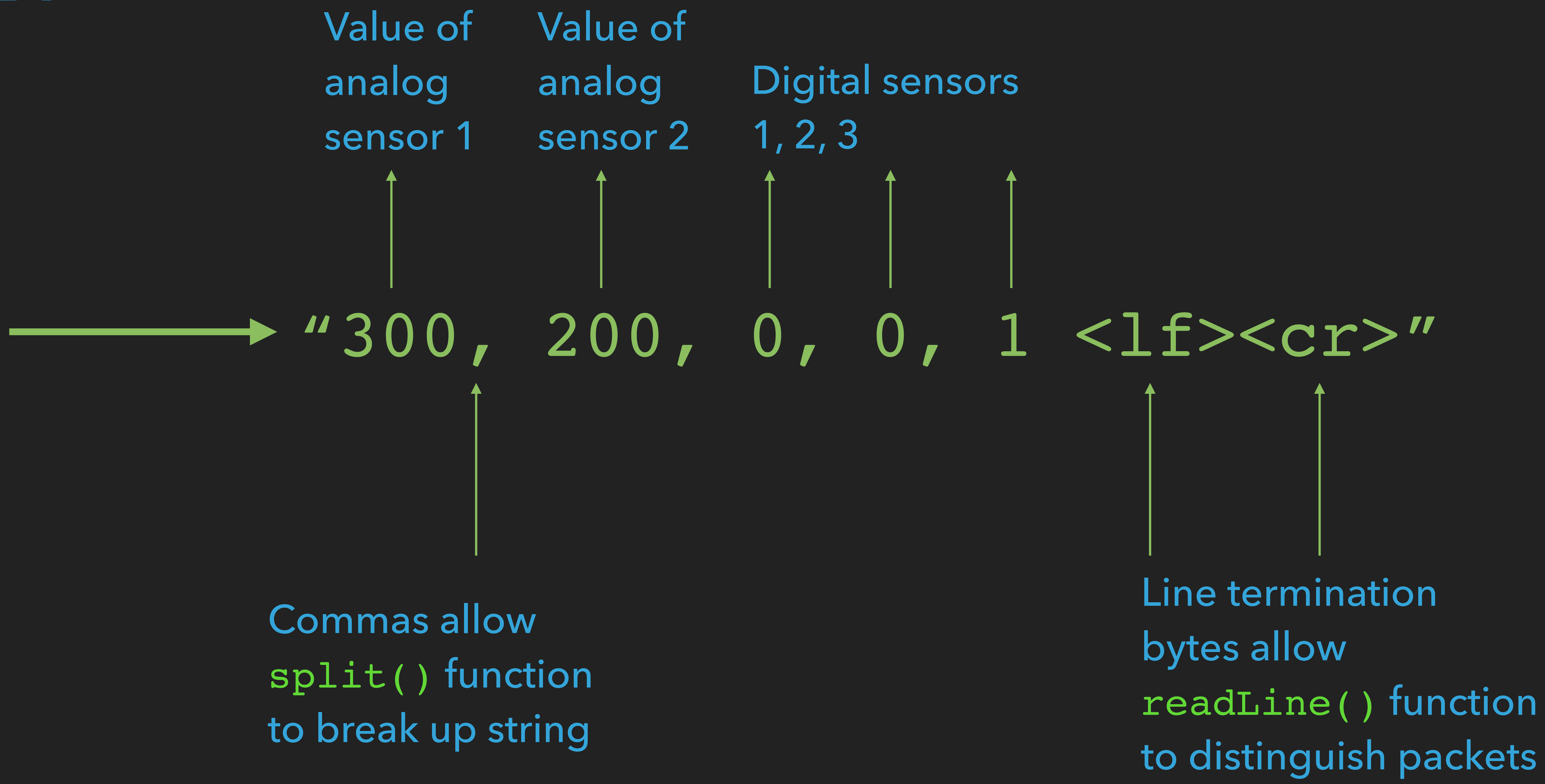
ARDUINO

OUTSIDE WORLD

Receiver breaks up packet into parts

OUTSIDE WORLD

P5



Value of analog sensor 1

Value of analog sensor 2

Digital sensors 1, 2, 3

`“300, 200, 0, 0, 1 <lf><cr>”`

Commas allow `split()` function to break up string

Line termination bytes allow `readLine()` function to distinguish packets

Why “Handshaking”?

Because: ASYNCHRONOUS

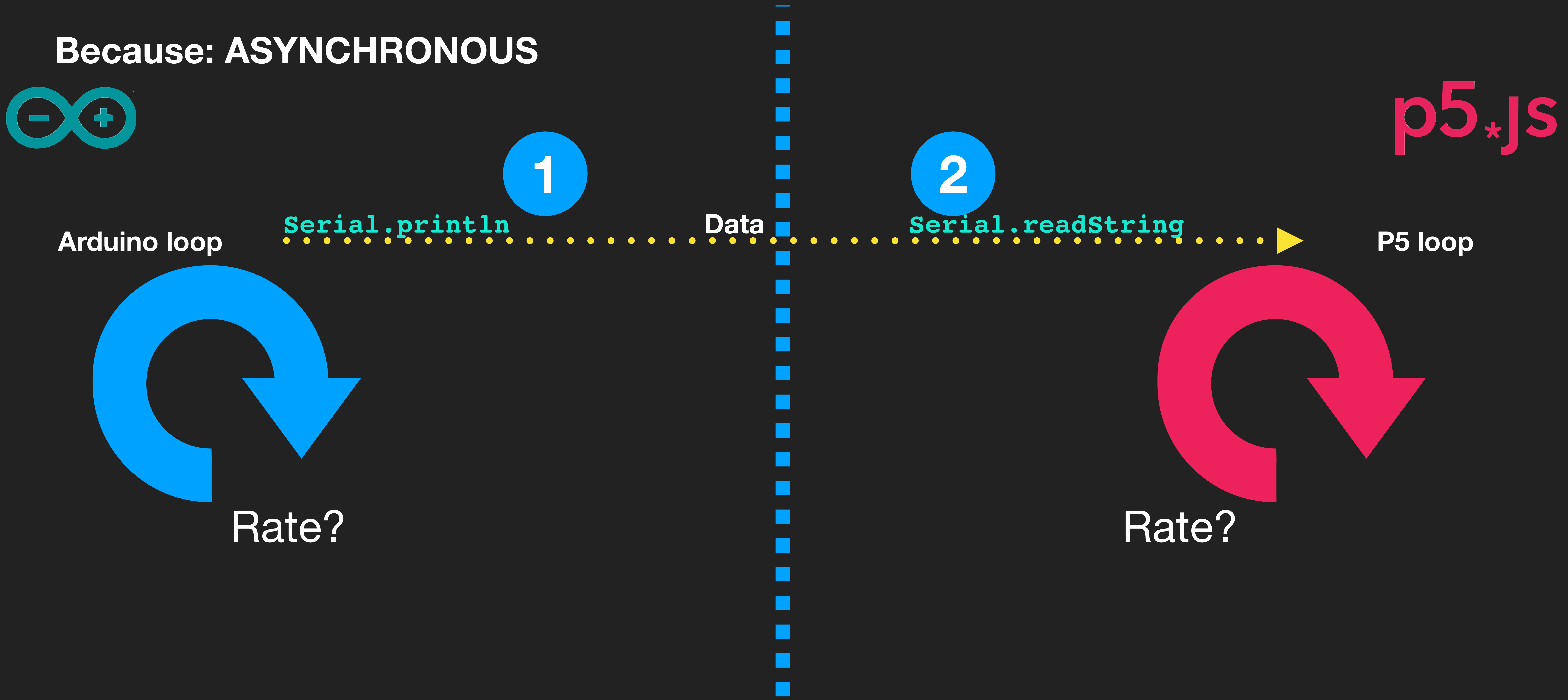
There is no coordination between then LOOP in Arduino and the (draw) LOOP in P5. Each runs at their own speed. Handshaking, call-and-response, etc. are methods to COORDINATE SENDER AND RECEIVER

This might not always be necessary, and different platforms can behave differently.

It helps to **visualize** what’s happening with data transfer in more detail.

Why “Handshaking”?

Because: ASYNCHRONOUS



Why “Handshaking”?

Because: ASYNCHRONOUS

Arduino loop

1

`Serial.println`

Bytes **added** to buffer each time through loop

Buffer

Data

Bytes **sent** from buffer based on baud rate

2

3

Bytes **added** to buffer “below” P5, triggering SerialEvent

Buffer

4

`Serial.readString`

Bytes **removed** from buffer when Serial.readString sees line terminator

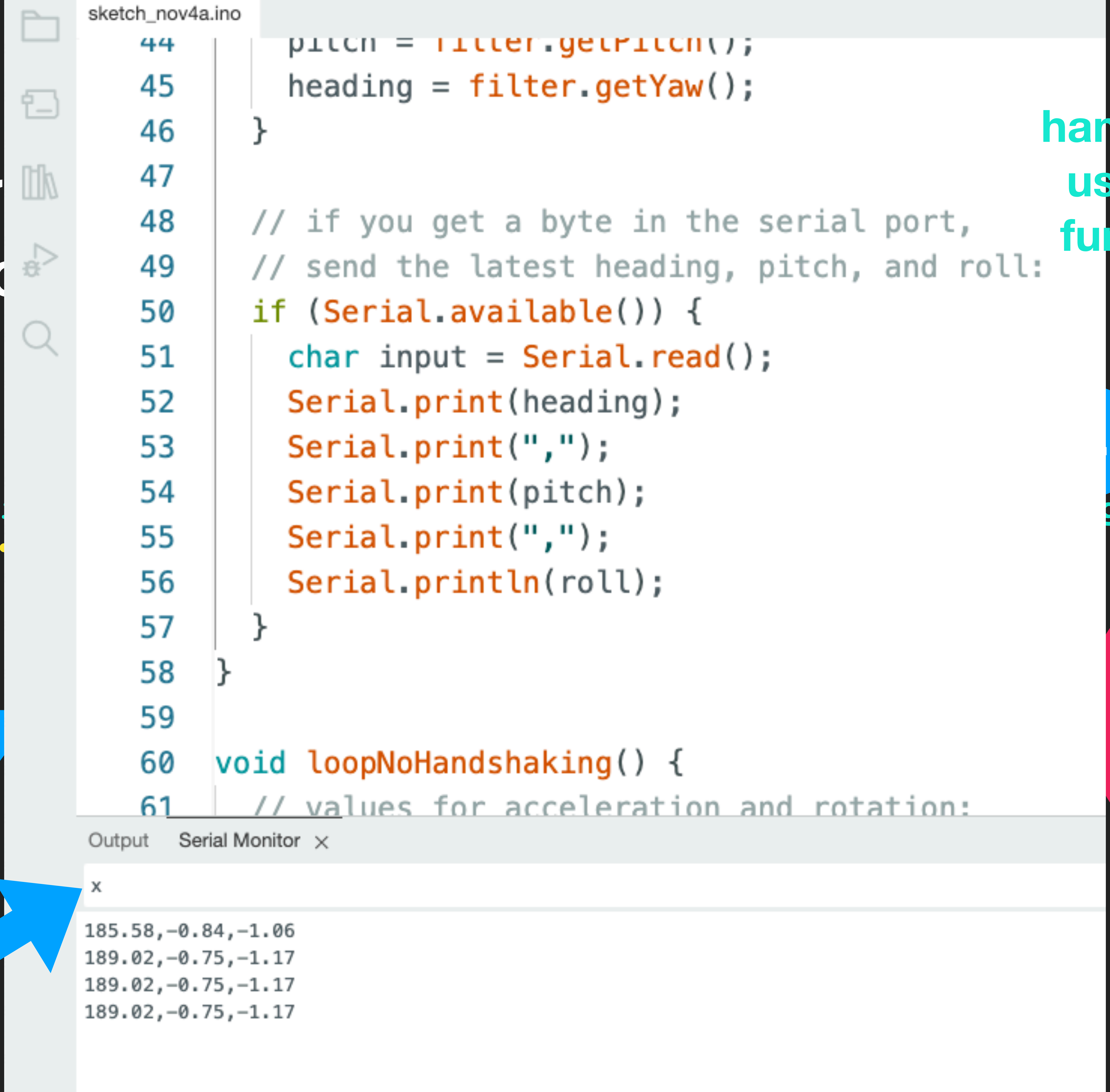
P5 loop

Why “Ha

Because: ASYNC

Arduino loop

Ser



```
    pitch = filter.getPitch();
    heading = filter.getYaw();
  }

  // if you get a byte in the serial port,
  // send the latest heading, pitch, and roll:
  if (Serial.available()) {
    char input = Serial.read();
    Serial.print(heading);
    Serial.print(",");
    Serial.print(pitch);
    Serial.print(",");
    Serial.println(roll);
  }
}

void loopNoHandshaking() {
  // values for acceleration and rotation:
```

```
x
```

```
185.58,-0.84,-1.06
189.02,-0.75,-1.17
189.02,-0.75,-1.17
189.02,-0.75,-1.17
```

You can debug handshaking in the IDE using the serial send function to send data back to Arduino.

P5 loop