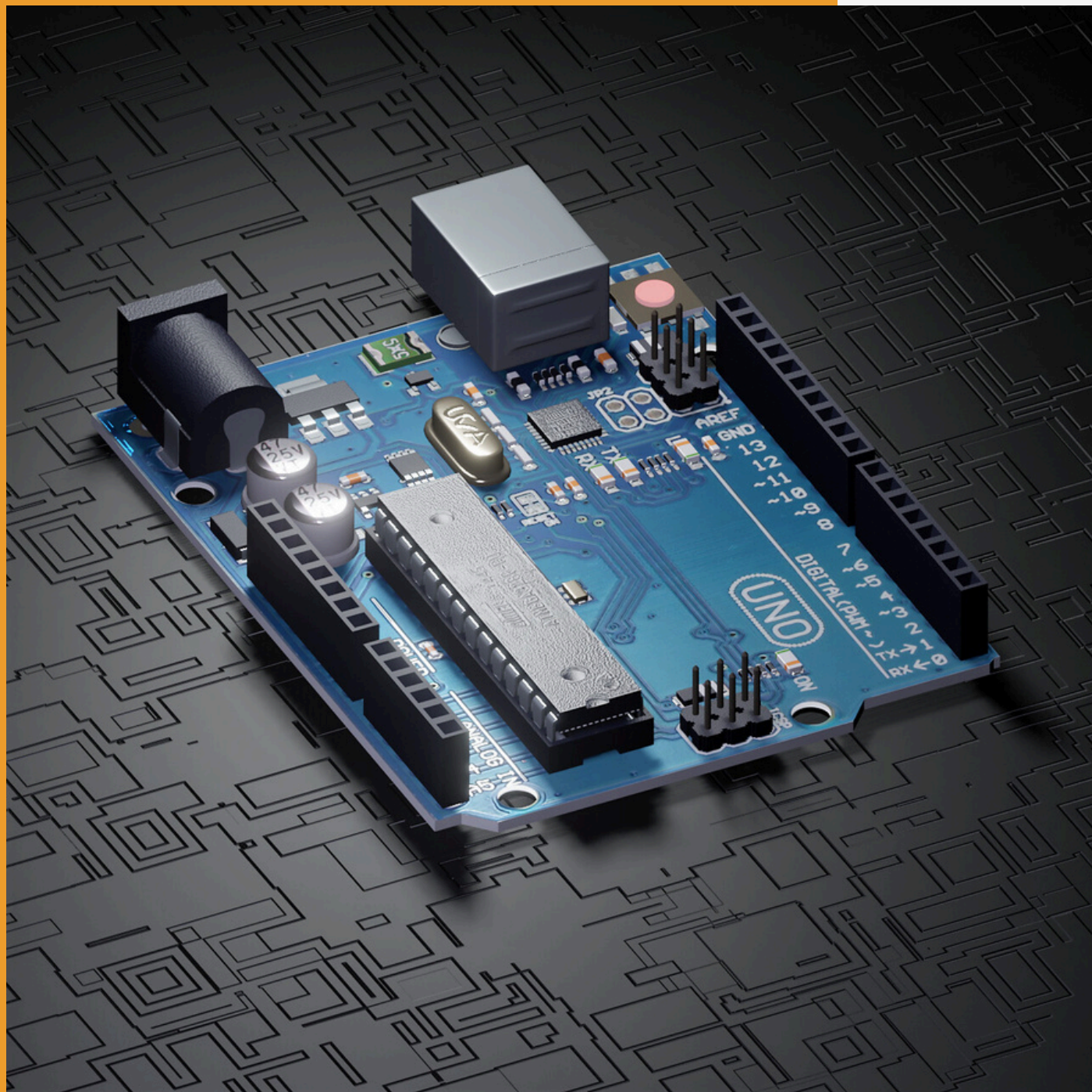




ARDUINO WORKSHOP

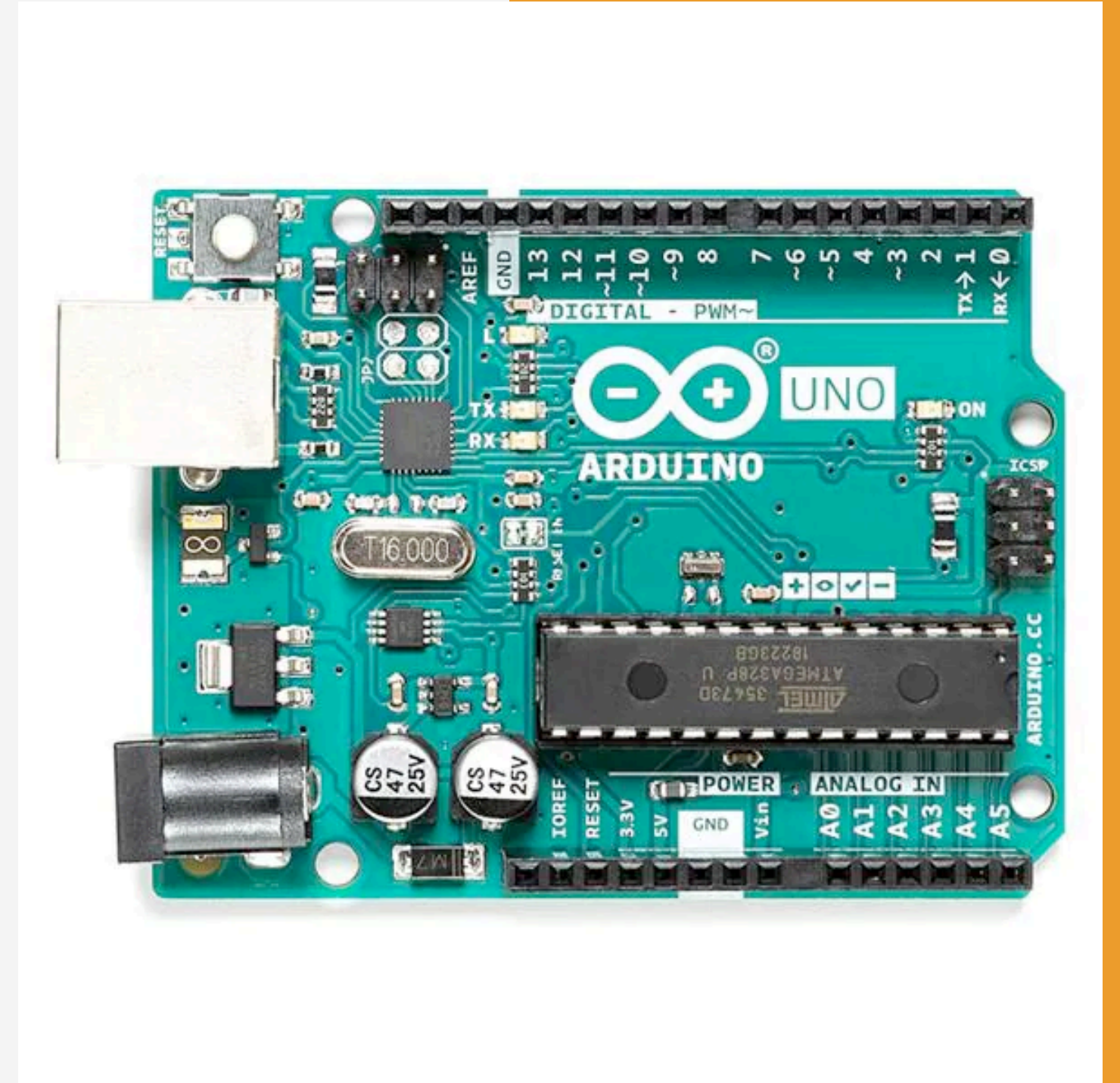
An Introduction to
Programming and Arduino

Sam Knox
Keith Anderson



WHAT IS AN ARDUINO?

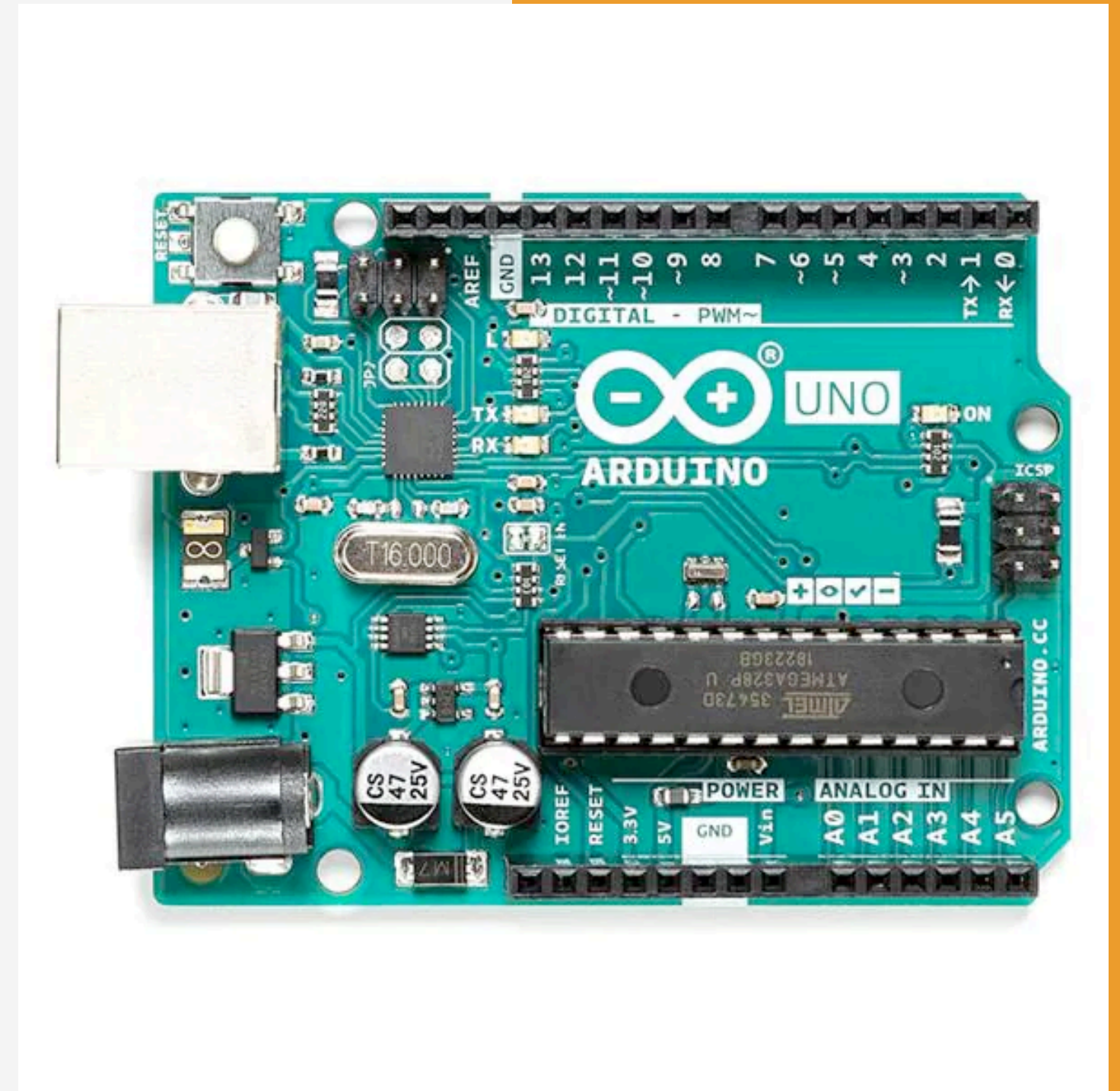
It's a computer.



WHAT IS AN ARDUINO?

Like a desktop/laptop, it has:

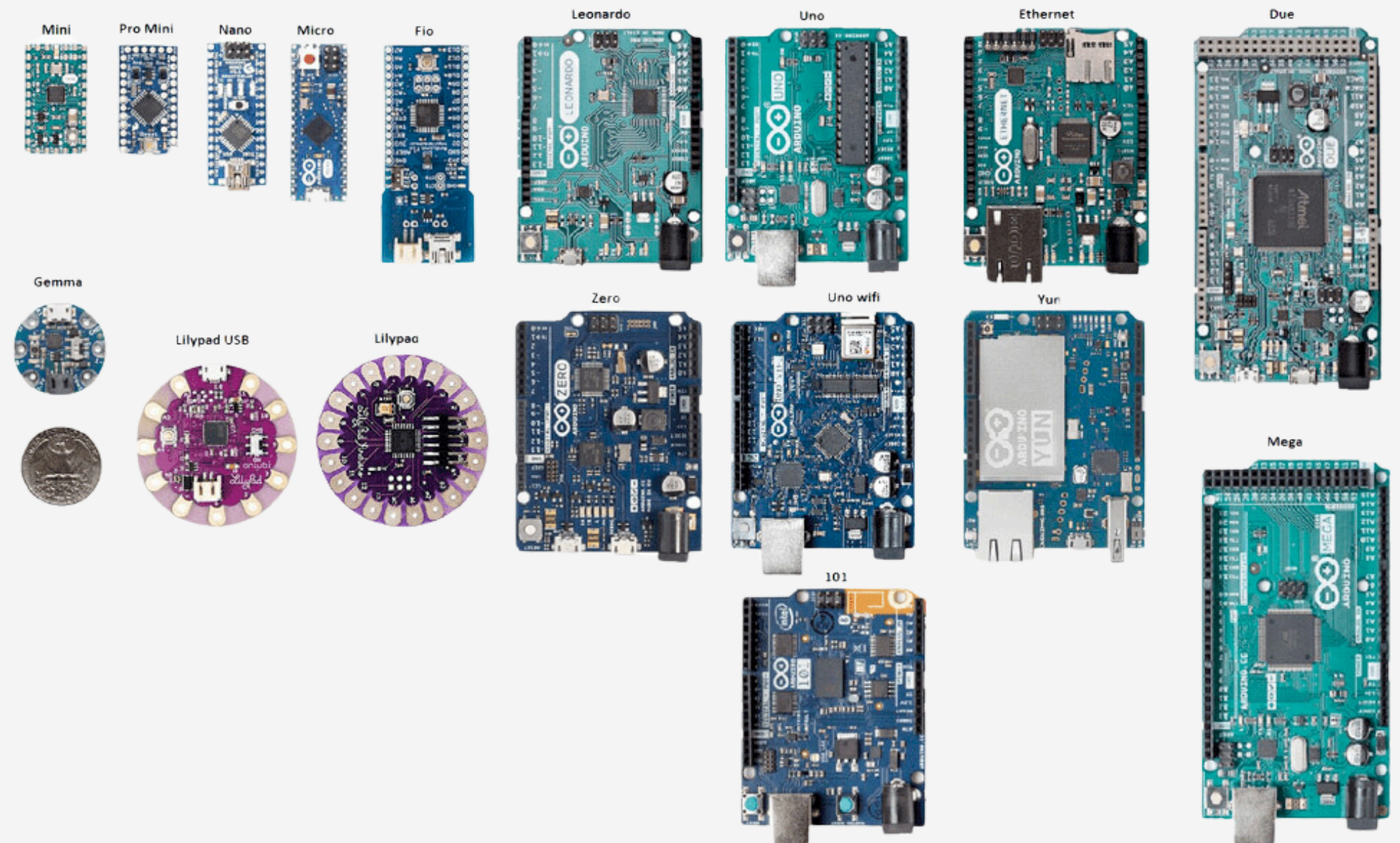
- A CPU (Central Processing Unit)
- Plugs to connect components
- Ability to run programs

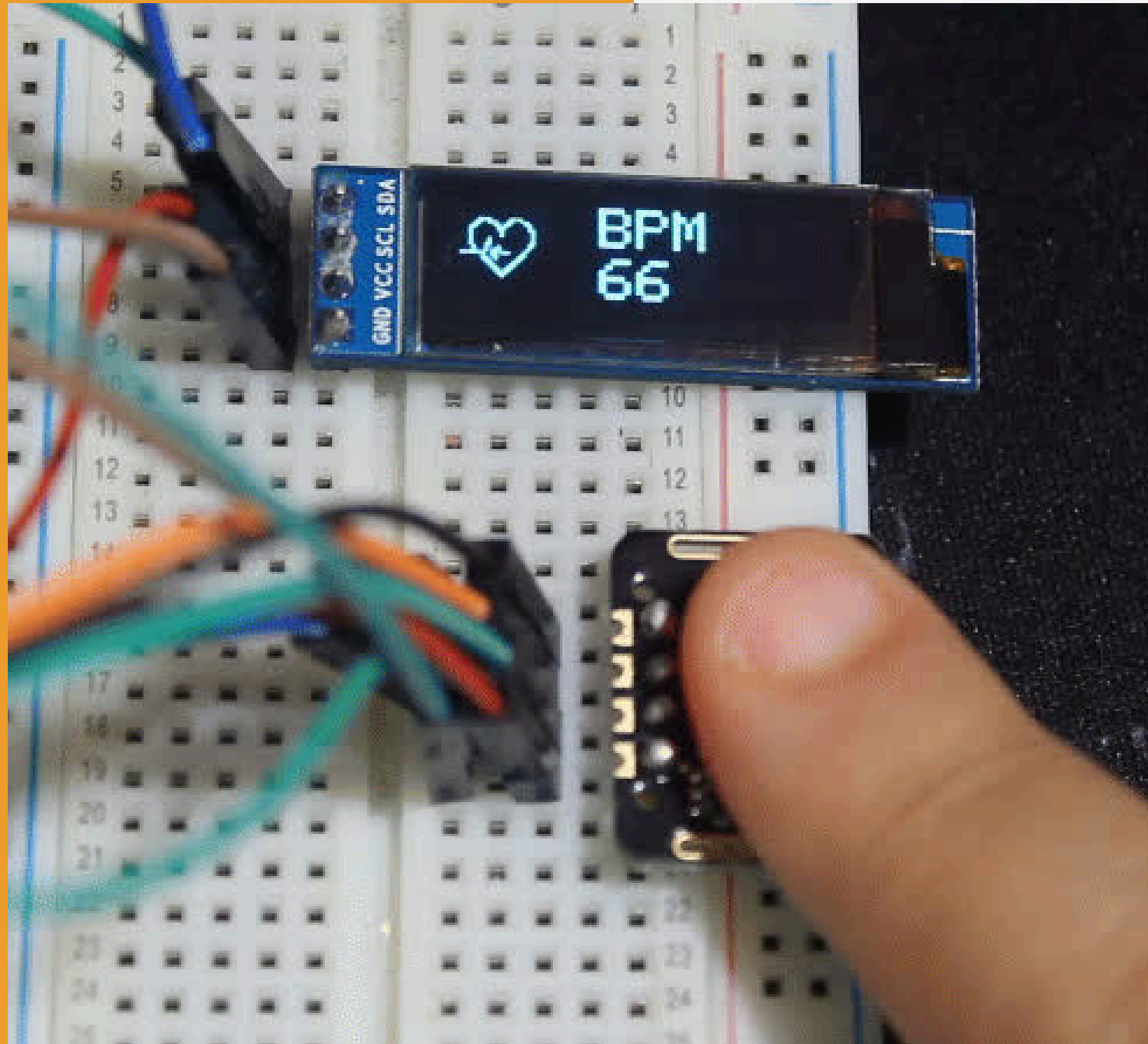




The Arduino Ecosystem

- Arduino is the brand name of a family of microcontroller boards
- They come in all shapes and sizes
- There's a large community of hobbyists and professionals using the Arduino platform
- There's a large number of compatible attachments and components

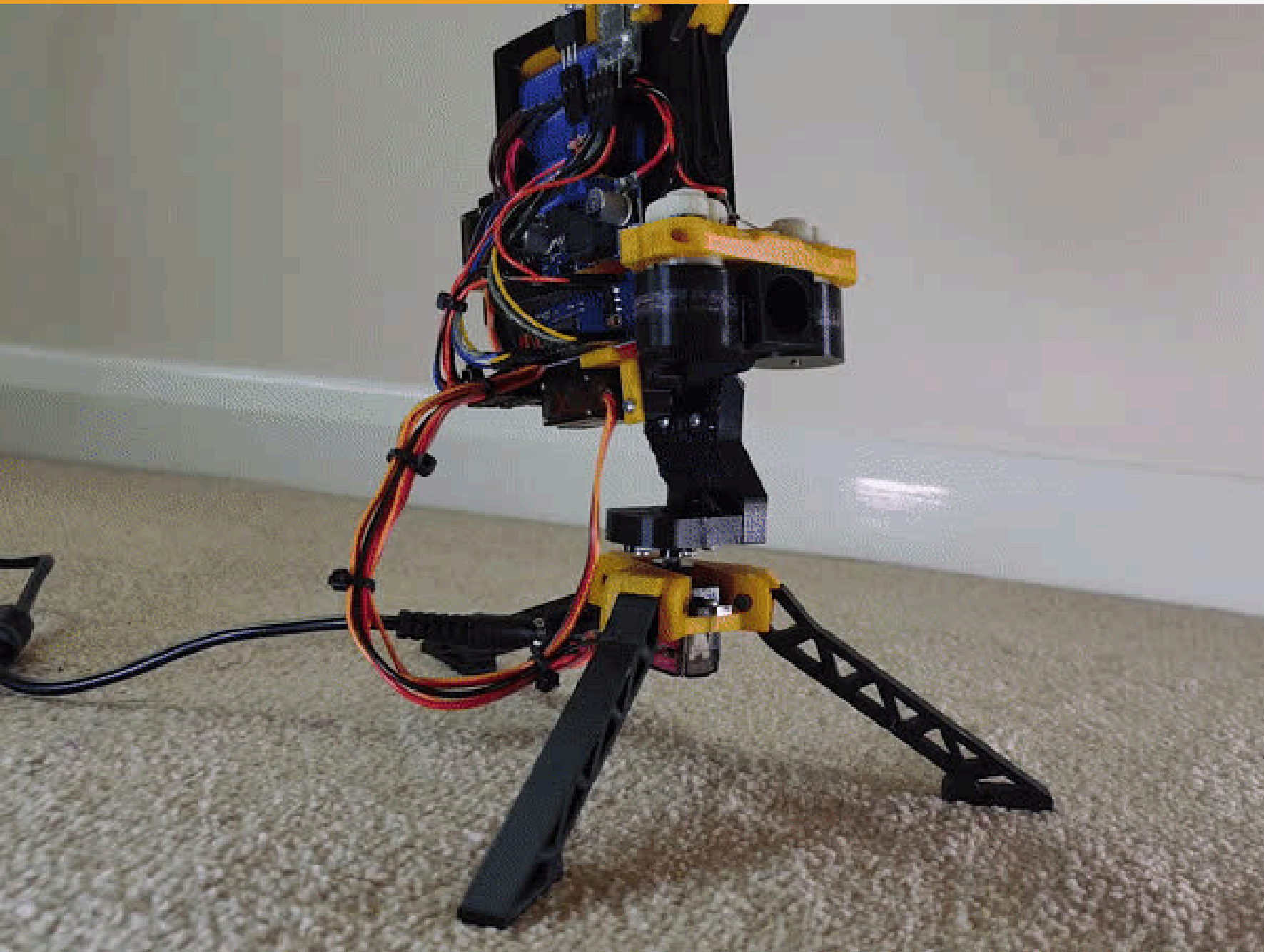




WHAT CAN WE DO WITH ONE?

- Make a heartbeat sensor
- Control RGB lights
- Make a flight controller for a drone





WHAT CAN WE DO WITH ONE?

- Build a Bluetooth-controlled Nerf turret



Build a system monitor for your vehicle



What are some useful things that we could measure and display?



Build a system monitor for your vehicle



- Speed
- Battery Temperature
- Motor Temperature
- GPS Location
- Battery Voltage
- Battery Current
- Lap Timing
- Efficiency





EVelocity Rules

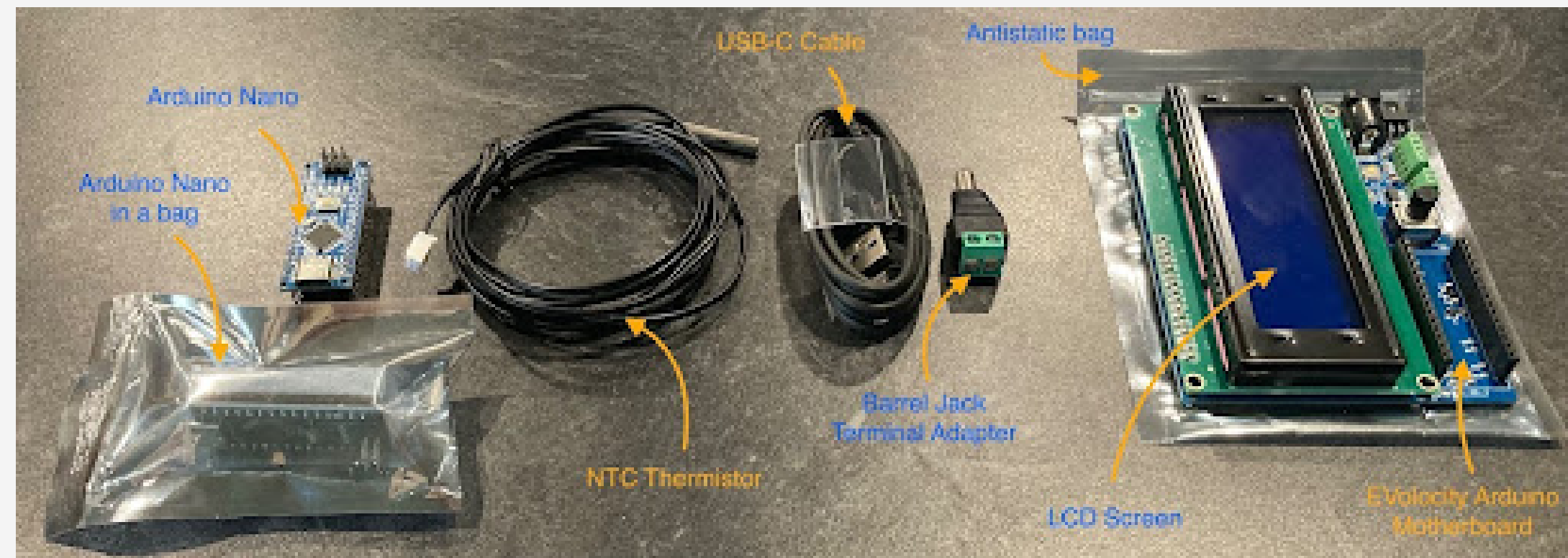
OPEN	10	Lithium Batteries (skip if lead-acid used)
P	F	10.1 BMS present or balance charger used. Cells balanced to within 0.1V (highest charged to lowest charged cell).
P	F	10.2 Low voltage indicator present and in working condition. Team is aware of the individual cell and total pack lowest voltage and has a multimeter to measure it in the pits.
P	F	10.3 Battery temperature monitoring present and indicator light working. Measurement probe has a good thermal connection with the cell's alert trigger set at <60°C.
P	F	10.4 An adequate firewall is present to protect the driver incase of battery thermal runaway or fire.



Your Arduino Kit

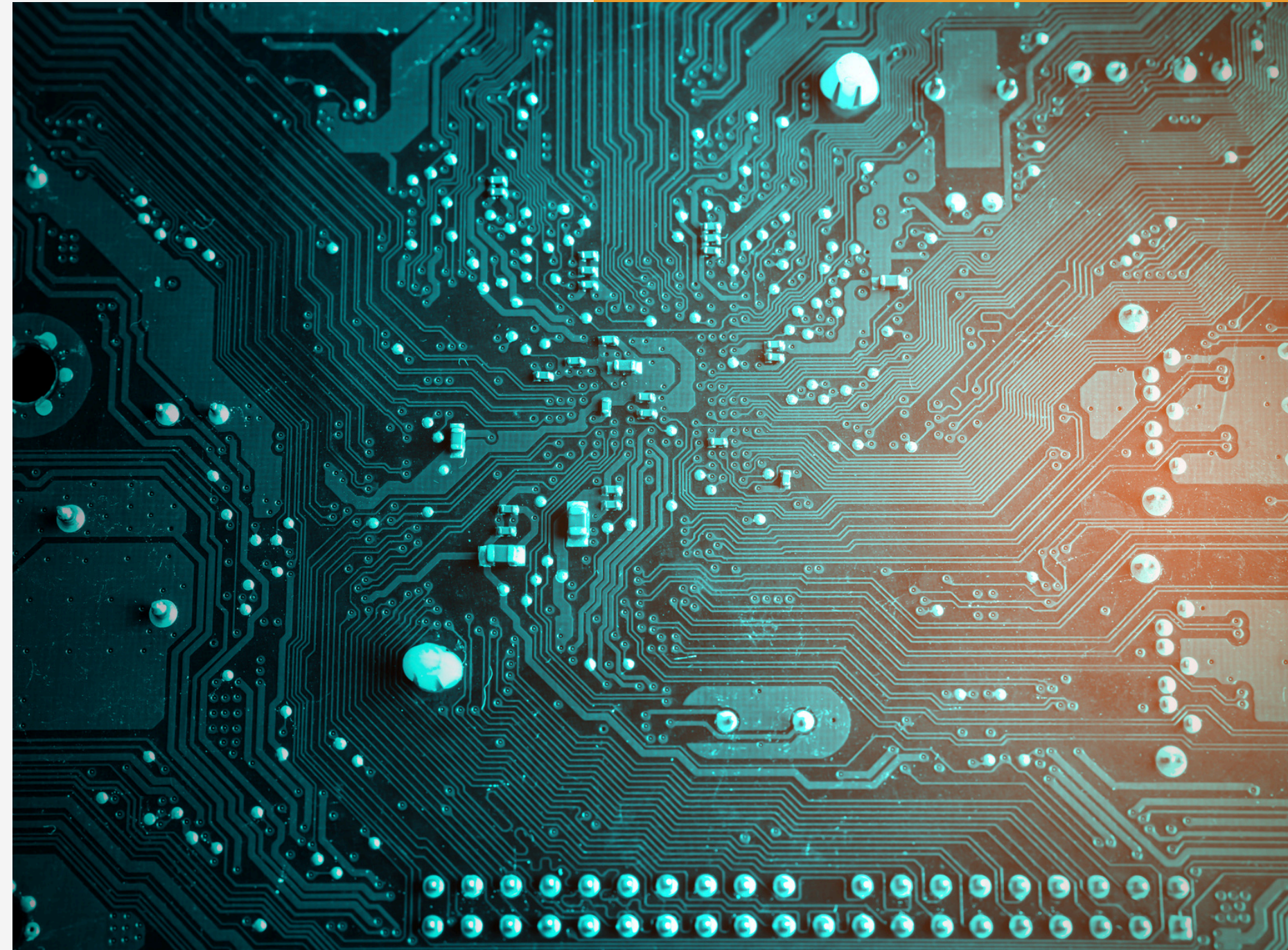
Check that your kit has these items:

- Arduino Nano (In a bag)
- NTC Thermistor
- USB C Cable
- Barrel Jack Terminal Adapter
- LCD Screen
- EVolocity Arduino Motherboard (AMB)



ELECTRONICS FUNDAMENTALS

- Electricity
- Ohm's Law

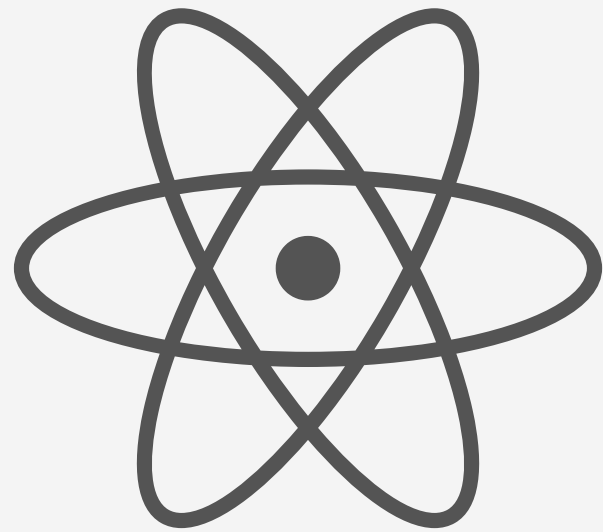


What is electricity?





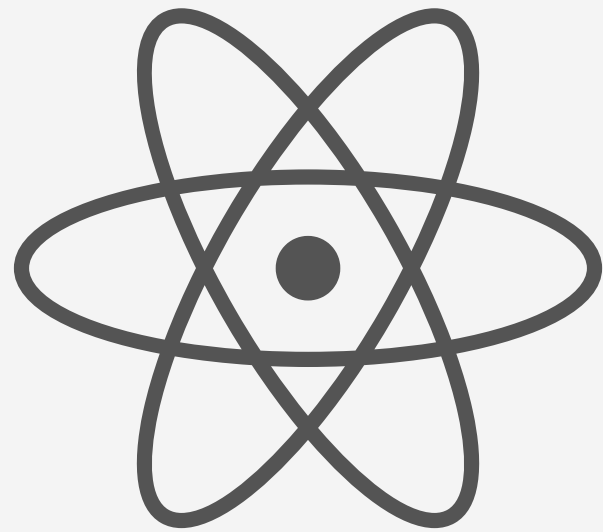
What is electricity?



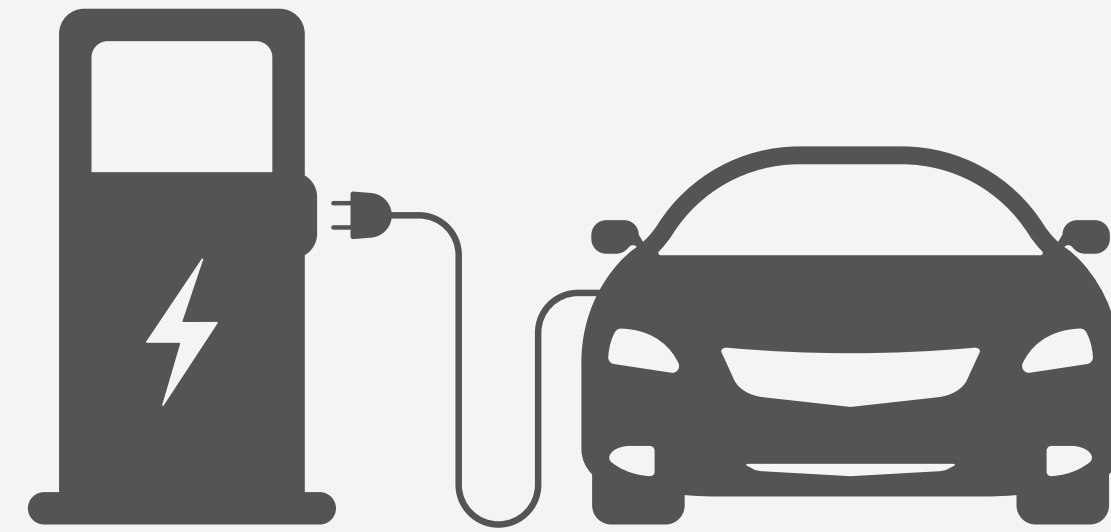
Flow of electrons



What is electricity?



Flow of electrons



A form of energy

Electricity is like a water hose

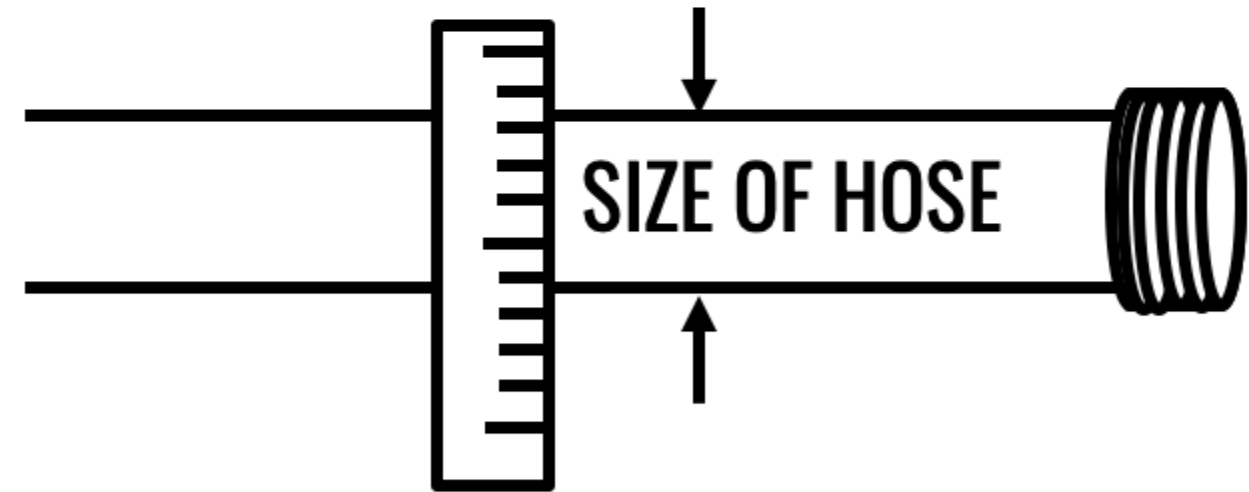
Voltage

Volts (V)



Current

Amps (A or I)



Resistance

Ohms (R or Ω)

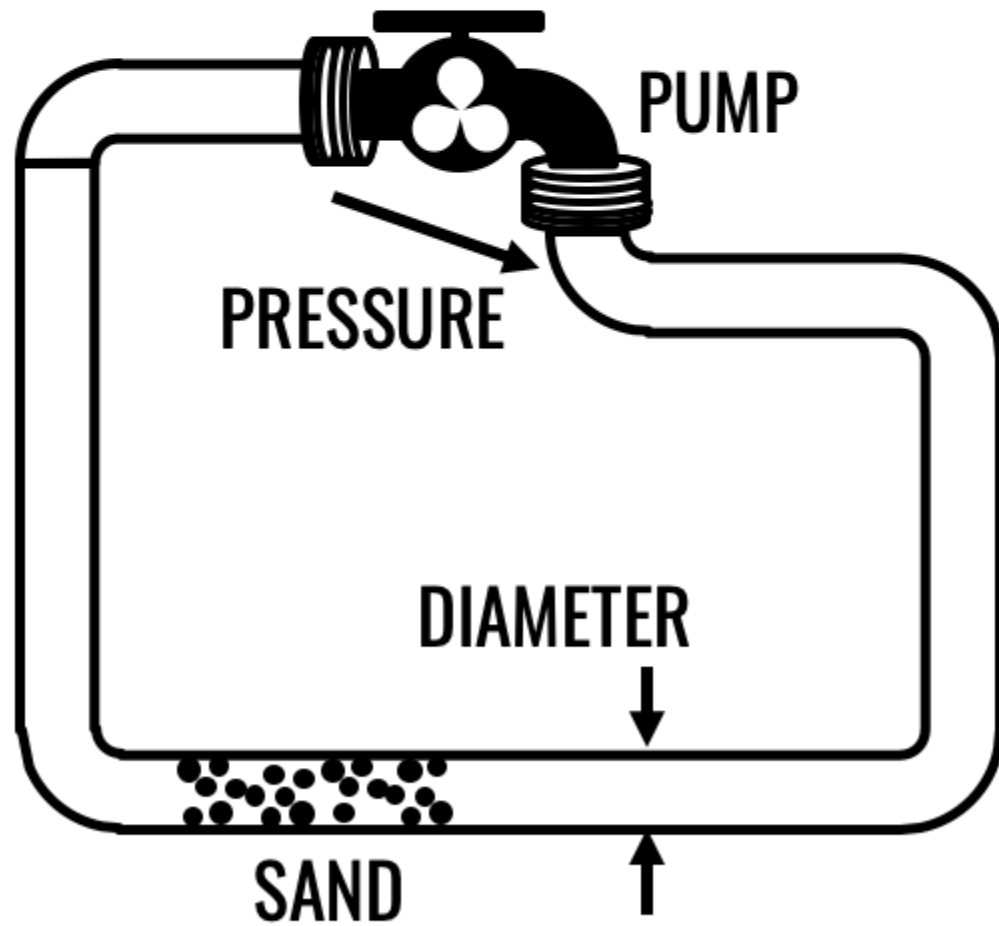


FREEING
ENERGY

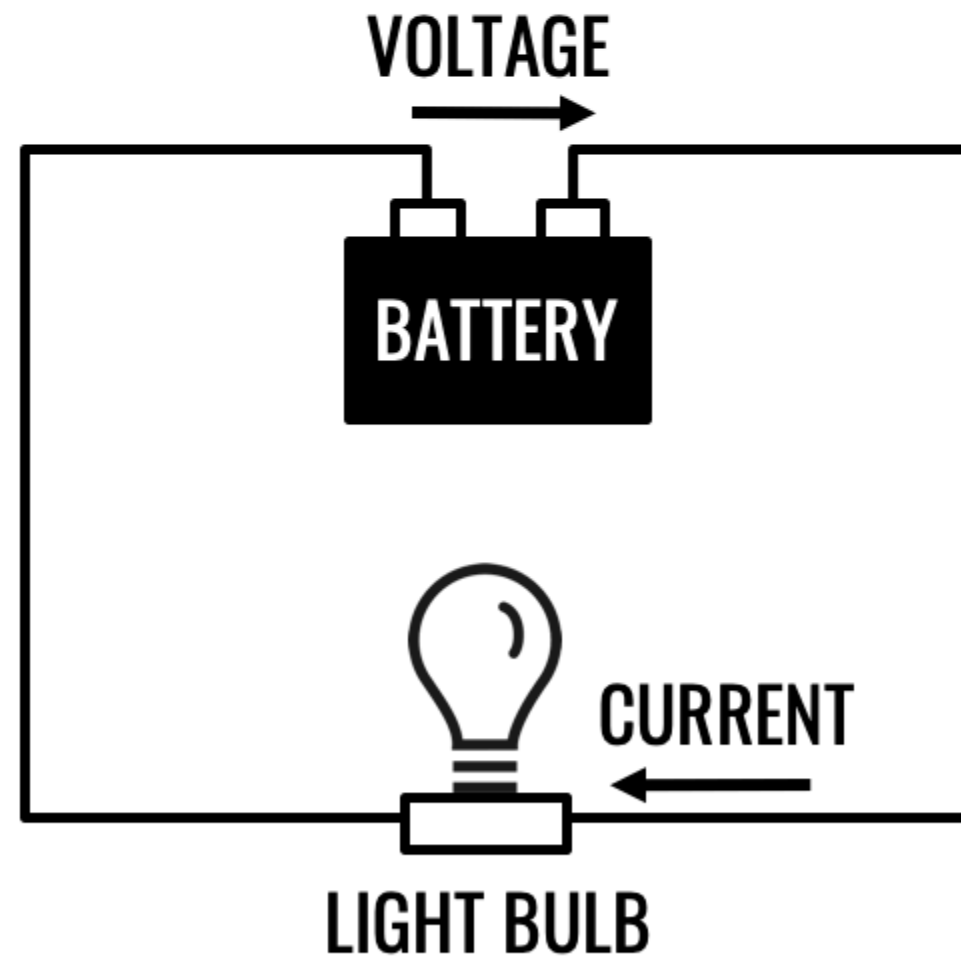
Voltage = Current x Resistance

$$(V = I \times R)$$

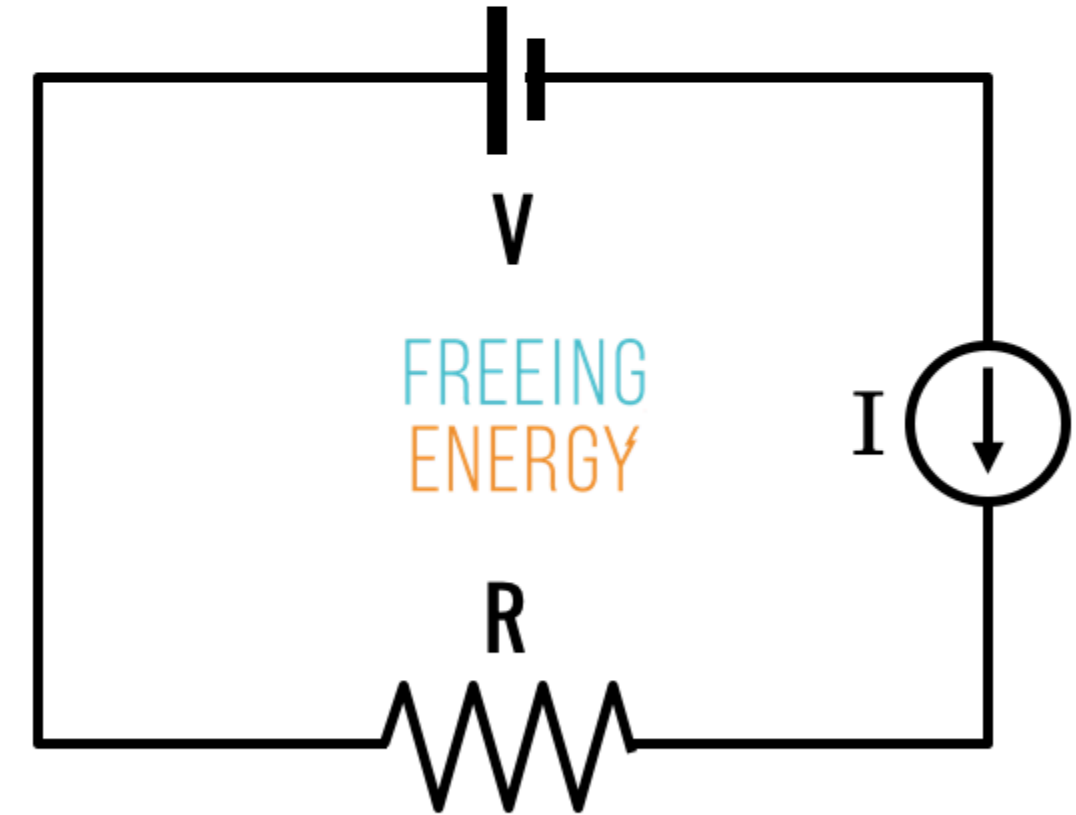
Water



Electricity



Circuit Diagram



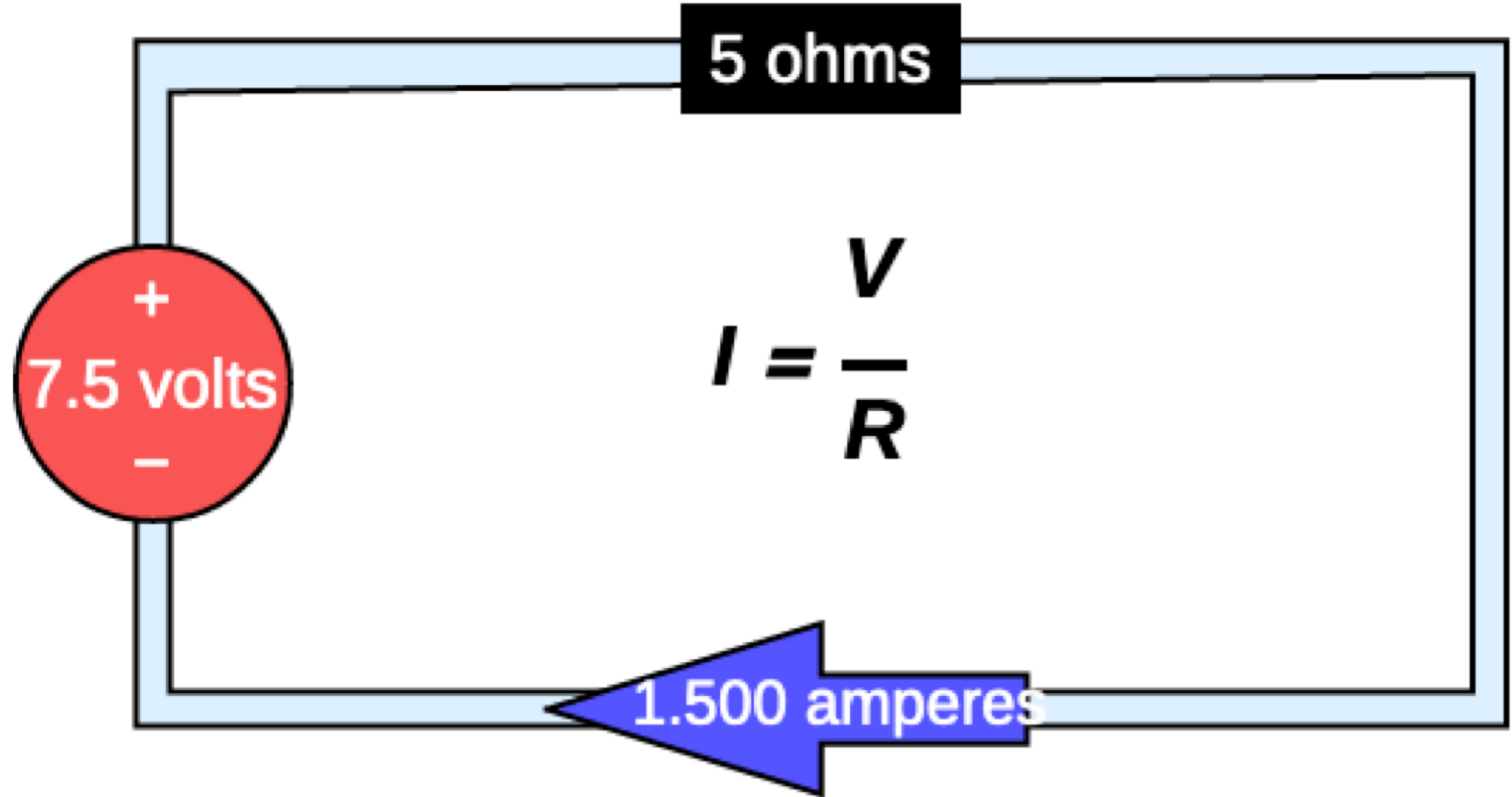
OHM'S LAW

Voltage(V) = Current(I) x Resistance(R)

More simply **$V = I \times R$**

Describes the relationship between electrical properties

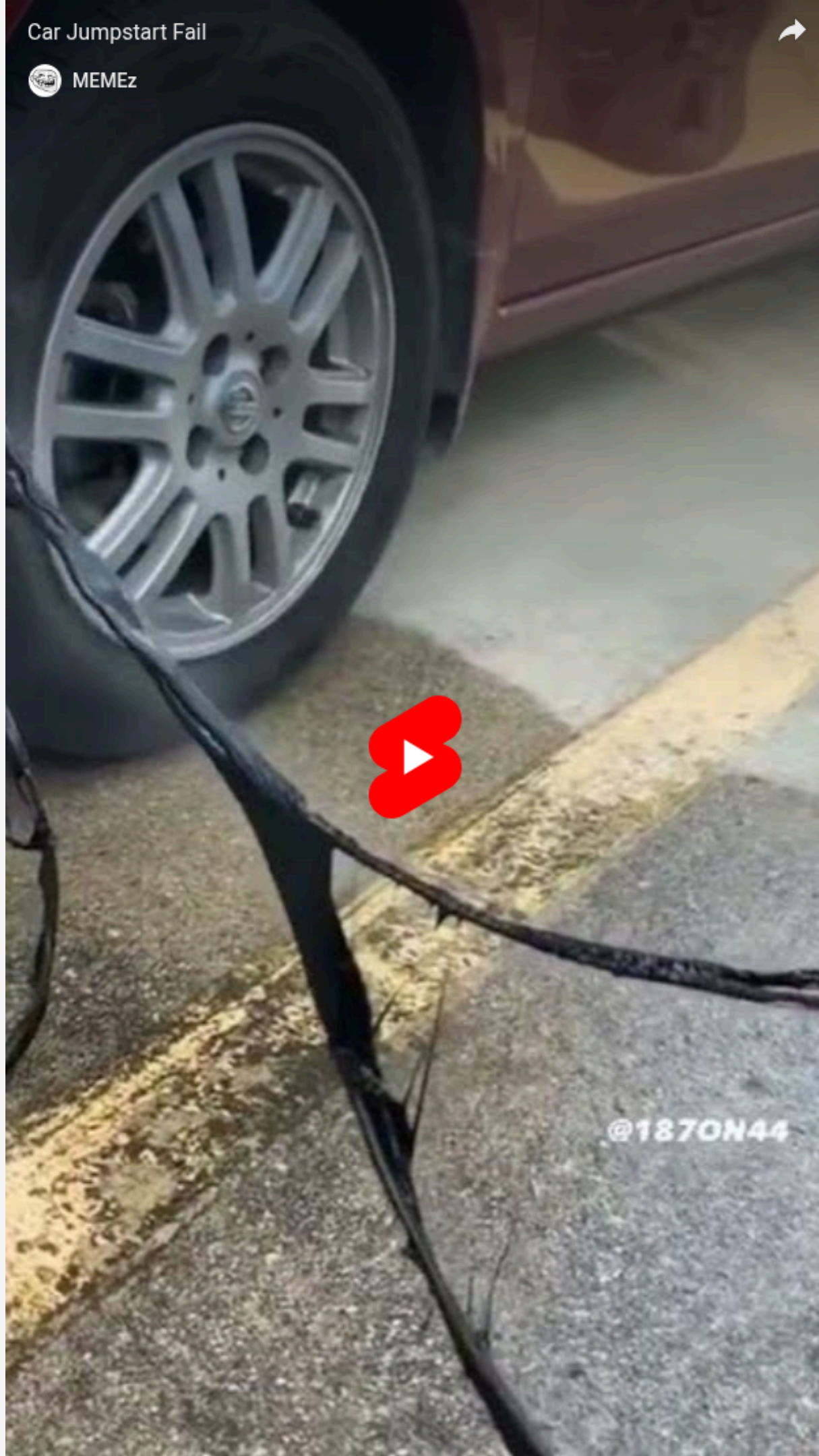




Car Jumpstart Fail



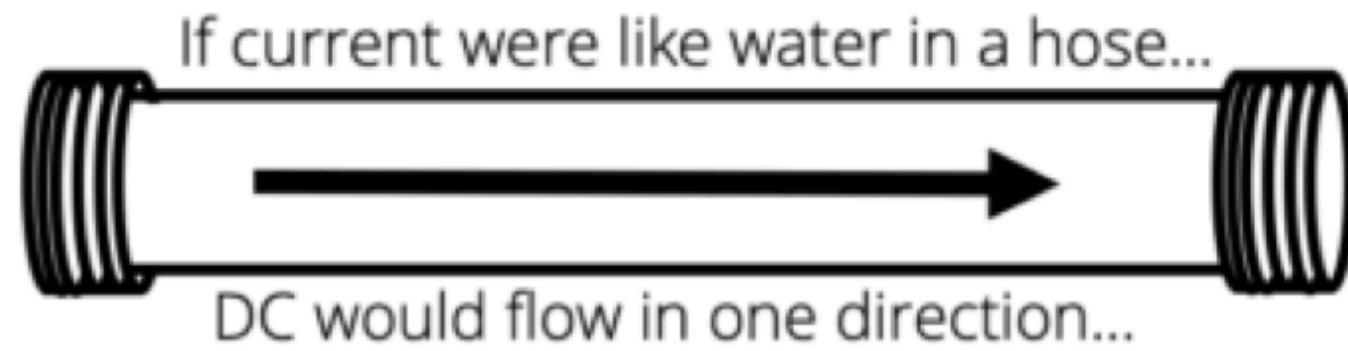
MEMEz



@187ON44

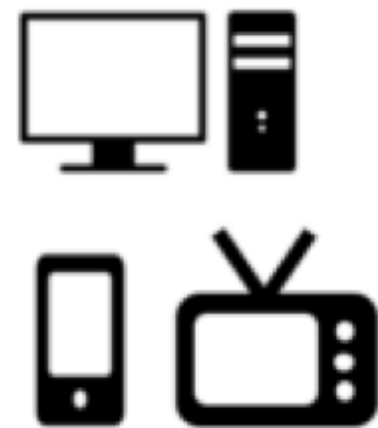
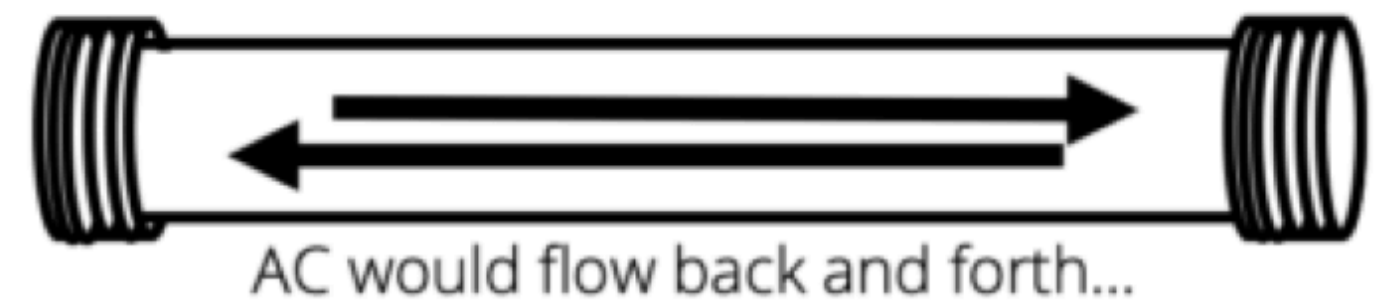
Alternating Current vs Direct Current

DC

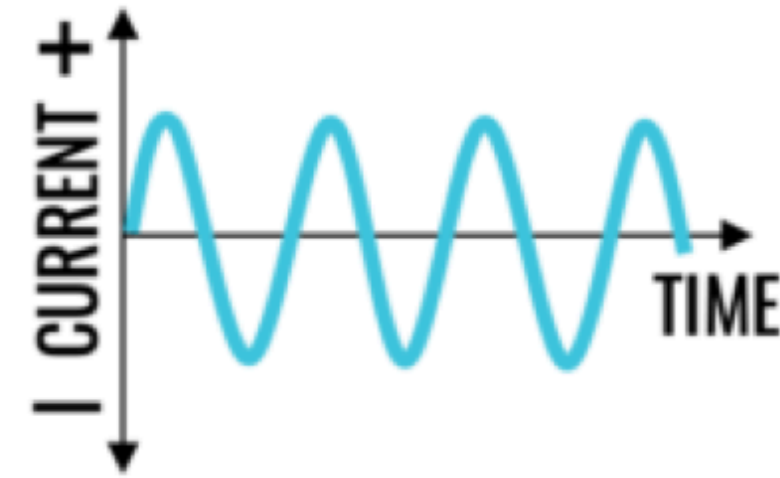


FREEING
ENERGY

AC



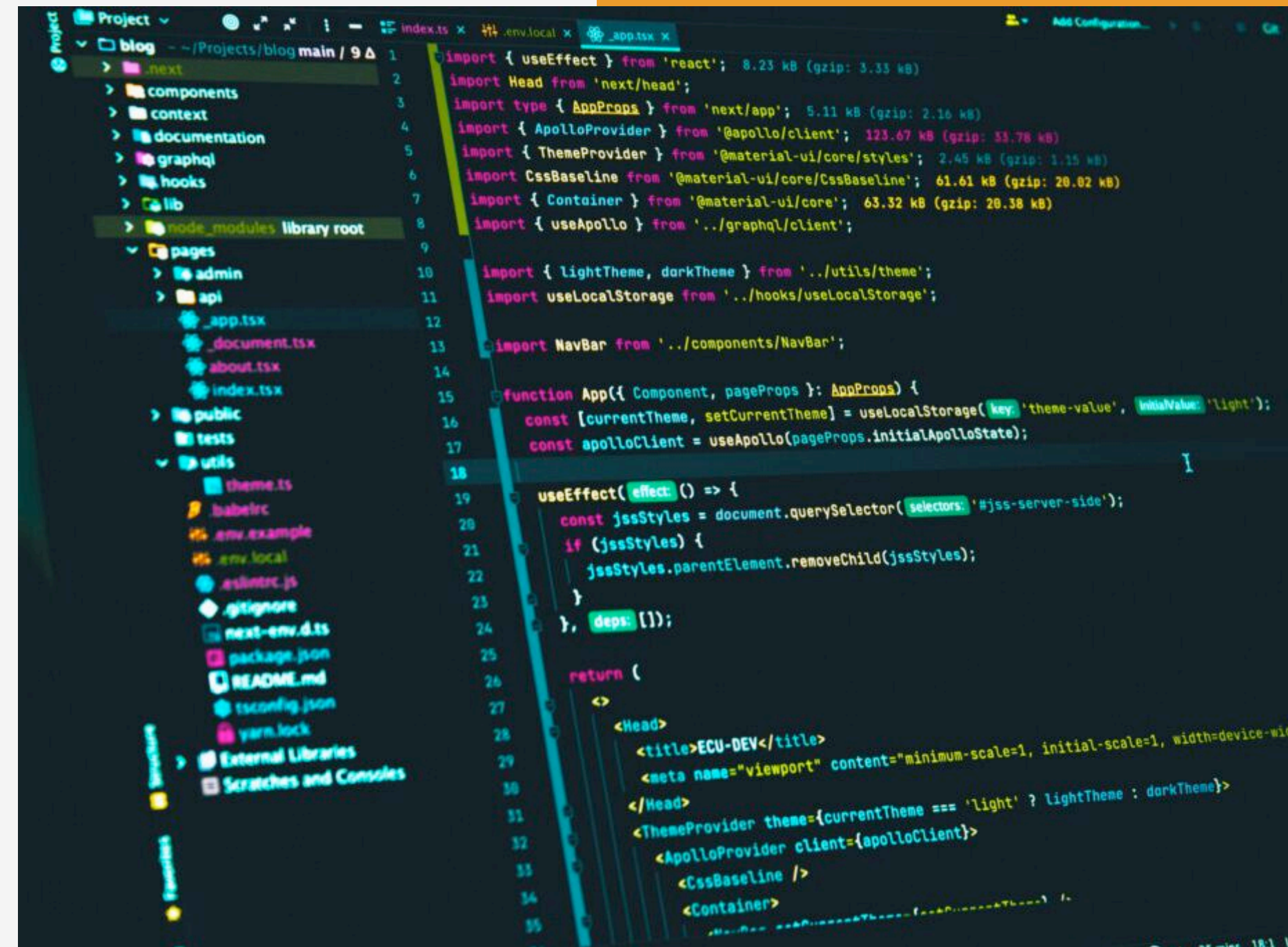
Things that use DC



Things that use AC

PROGRAMMING FUNDAMENTALS

- Analog vs Digital
- Variables & Data Types
- Conditionals
- Functions

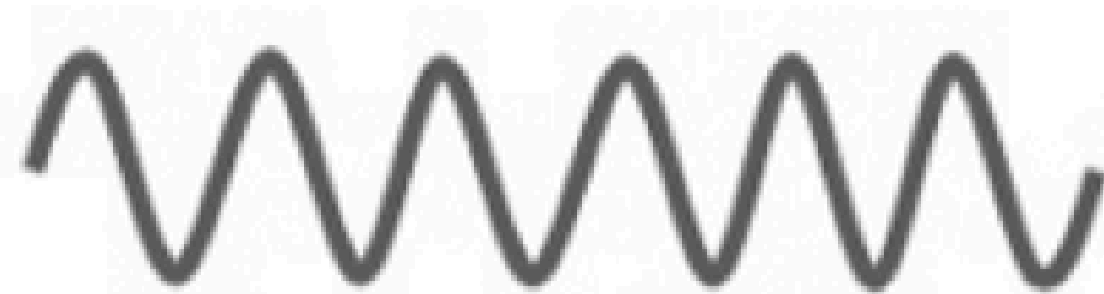


```
1 import { useEffect } from 'react'; 8.23 kB (gzip: 3.33 kB)
2 import Head from 'next/head';
3 import type { AppProps } from 'next/app'; 5.11 kB (gzip: 2.16 kB)
4 import { ApolloProvider } from '@apollo/client'; 123.67 kB (gzip: 33.78 kB)
5 import { ThemeProvider } from '@material-ui/core/styles'; 2.45 kB (gzip: 1.15 kB)
6 import CssBaseline from '@material-ui/core/CssBaseline'; 61.61 kB (gzip: 20.02 kB)
7 import { Container } from '@material-ui/core'; 63.32 kB (gzip: 20.38 kB)
8 import { useApollo } from '../graphql/client';
9
10 import { lightTheme, darkTheme } from '../utils/theme';
11 import useLocalStorage from '../hooks/useLocalStorage';
12
13 import NavBar from '../components/NavBar';
14
15 function App({ Component, pageProps }: AppProps) {
16   const [currentTheme, setCurrentTheme] = useLocalStorage({ key: 'theme-value', initialValue: 'light' });
17   const apolloClient = useApollo(pageProps.initialApolloState);
18
19   useEffect(() => {
20     const jssStyles = document.querySelector(selectors: '#jss-server-side');
21     if (jssStyles) {
22       jssStyles.parentElement.removeChild(jssStyles);
23     }
24   }, []);
25
26   return (
27     <>
28     <Head>
29       <title>ECU-DEV</title>
30       <meta name="viewport" content="minimum-scale=1, initial-scale=1, width=device-width" />
31     </Head>
32     <ThemeProvider theme={currentTheme === 'light' ? lightTheme : darkTheme}>
33       <ApolloProvider client={apolloClient}>
34         <CssBaseline />
35         <Container>
```

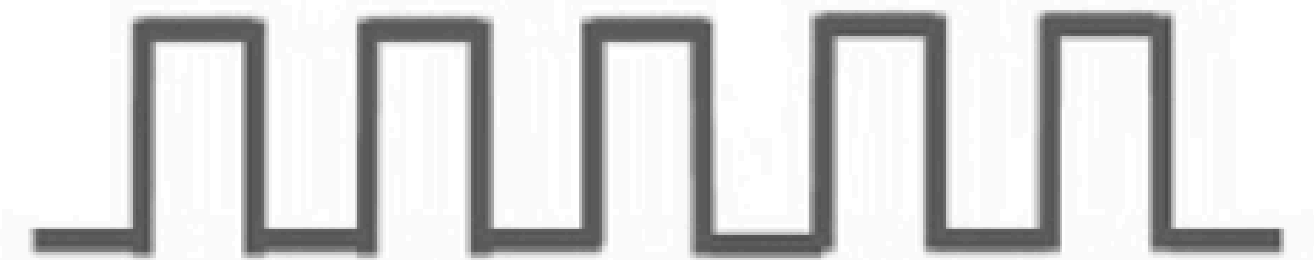


Analog vs Digital

- We can measure the voltage at any point of an analog signal and have an infinite number of possible values (depending on our measuring equipment)
- A digital signal will only ever measure one of two values, AKA binary



Analog Signal



Digital Signal



Digital Signals

We can read:

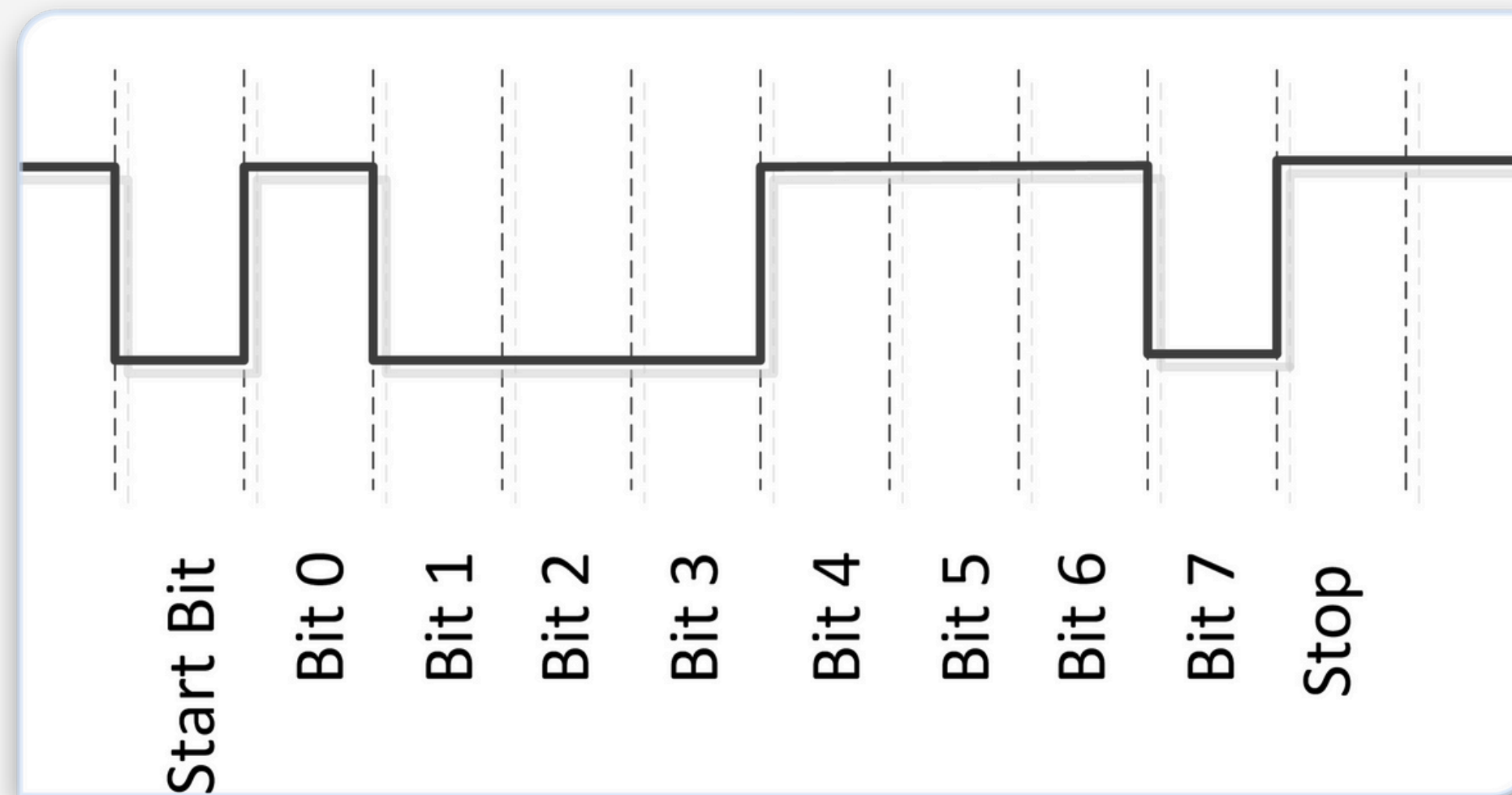
- On/Off
- High/Low
- 1/0



Digital Signals

We can use this principle to transmit messages:

- We communicate with binary digits or bits
- 8 **bits** make a **byte**



How many **bits**
in a **gigabyte**?



Digital Signals

So if we have two devices programmed to understand one another's bit sequences, then they can talk to each other.

For example, we tell both devices that:

- 01101011 = launch
- 01101010 = land

Then we send the byte from device 1 to device 2 and have it react accordingly.





Variables & Data Types

Variables are numbers that we identify using a name.

They can change, hence **variable**.

We declare (or initialise) them like so:

```
int my_integer = 1;
```

Now the variable "my_integer" can be used throughout the program by typing its name.



Variables & Data Types

Variables always have a “data type”, which tells the Arduino what kind of data it should contain.

int - holds values between -32,768 and 32,767

unsigned int - holds values between 0 and 65,535

char - holds a letter

float - holds a decimal

bool - holds either a **true** or a **false** (a 1 or a 0)

An Arduino Program

Variable

```
int x = 0; // initialise the variable "x"
```

Comments

```
void setup() {
```

```
    // nothing to set up in this example
```

```
}
```

Program
functions

```
void loop() {
```

```
    if(x == 1) {
```

```
        // do something
```

```
    }
```

```
    else {
```

```
        // do something else
```

```
    }
```

```
    x = x + 1;
```

```
}
```

Conditional
statement

Equation



Conditional Statements

We use logic operators to make decisions.

- Equal ($A == B$)
- Greater than ($A > B$)
- AND ($A \&\& B$)
- Not equal ($A != B$)
- Less than ($A < B$)
- OR ($A || B$)



Conditional Statements

"if" statement

```
void loop() {  
  
    if(x == 1) {  
        // do something  
    }  
    else {  
        // do something else  
    }  
  
    x = x + 1;  
}
```

What will this do?



Conditional Statements

"while" loop

```
void loop() {  
  
    while(x < 100) {  
        x = x + 1;  
    }  
    Serial.print(x) ;  
}
```

And this?



Conditional Statements

"for" loop

```
void loop() {  
  
  // we write a for loop with the following syntax:  
  //   for(variable; condition; increment){}  
  // and we can declare our variable inside the statement if we like  
  
  for(int x = 0; x <= 10; x++){  
    Serial.print(x);  
  }  
}
```

What about this?



Functions

A way for us to tidy up and reuse our code.

```
void loop() {  
  // put your main code here, to run repeatedly:  
  blink();  
}  
  
void blink() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(500);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(500);  
}
```

Functions are called from within our main loop (or from within other functions).



Libraries

Another way for us to tidy up our code.

Provides us with functions and tools to operate hardware without us having to know exactly how the hardware works.

```
#include <LiquidCrystal.h>
```

This operation adds code to our program behind the scenes, i.e. we don't see it but we can still call its functions.

We read the library documentation to find out what those functions are.





The Arduino IDE

If you are using your own computer, and you haven't already, you should download and install the IDE now:

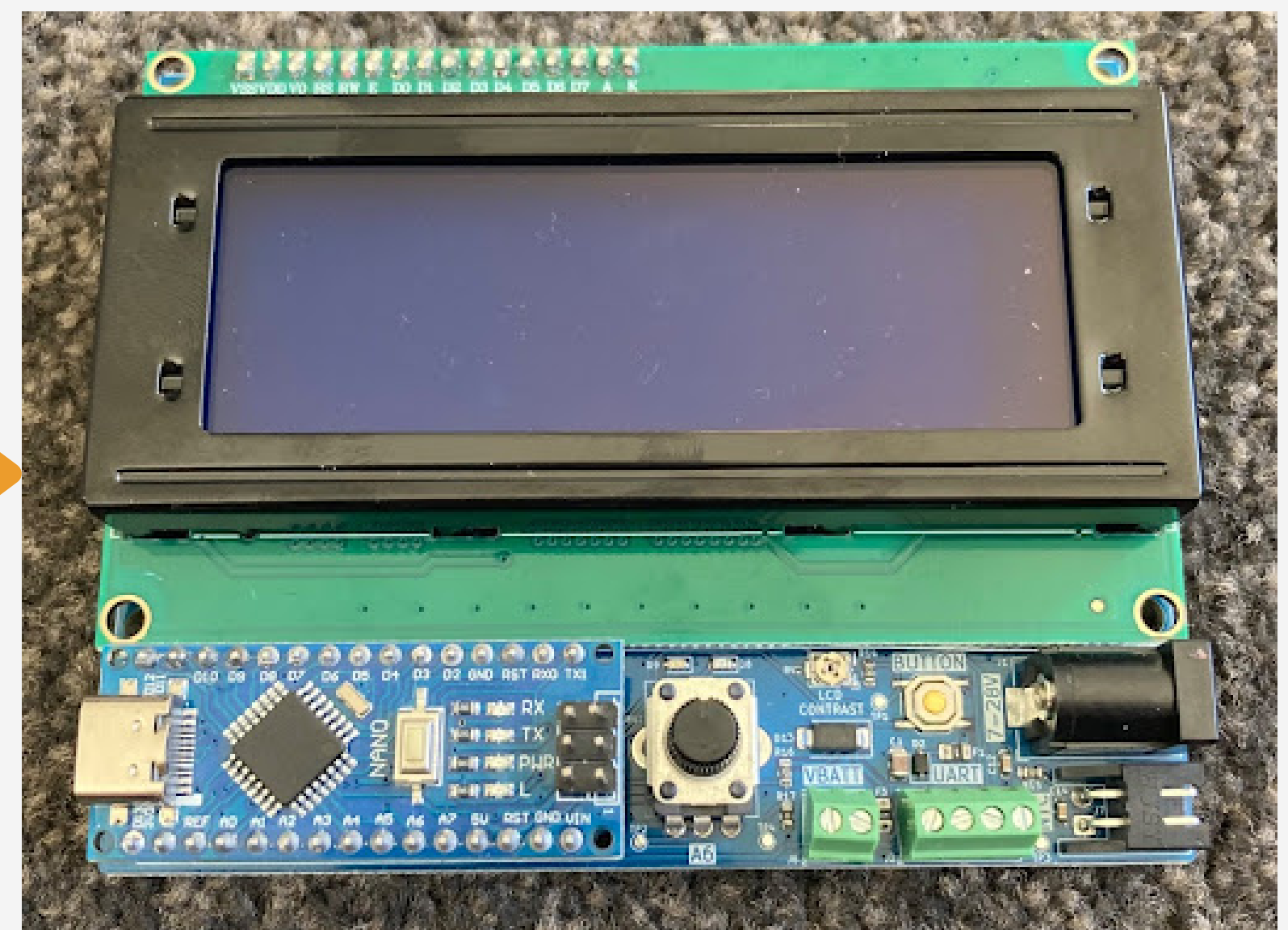
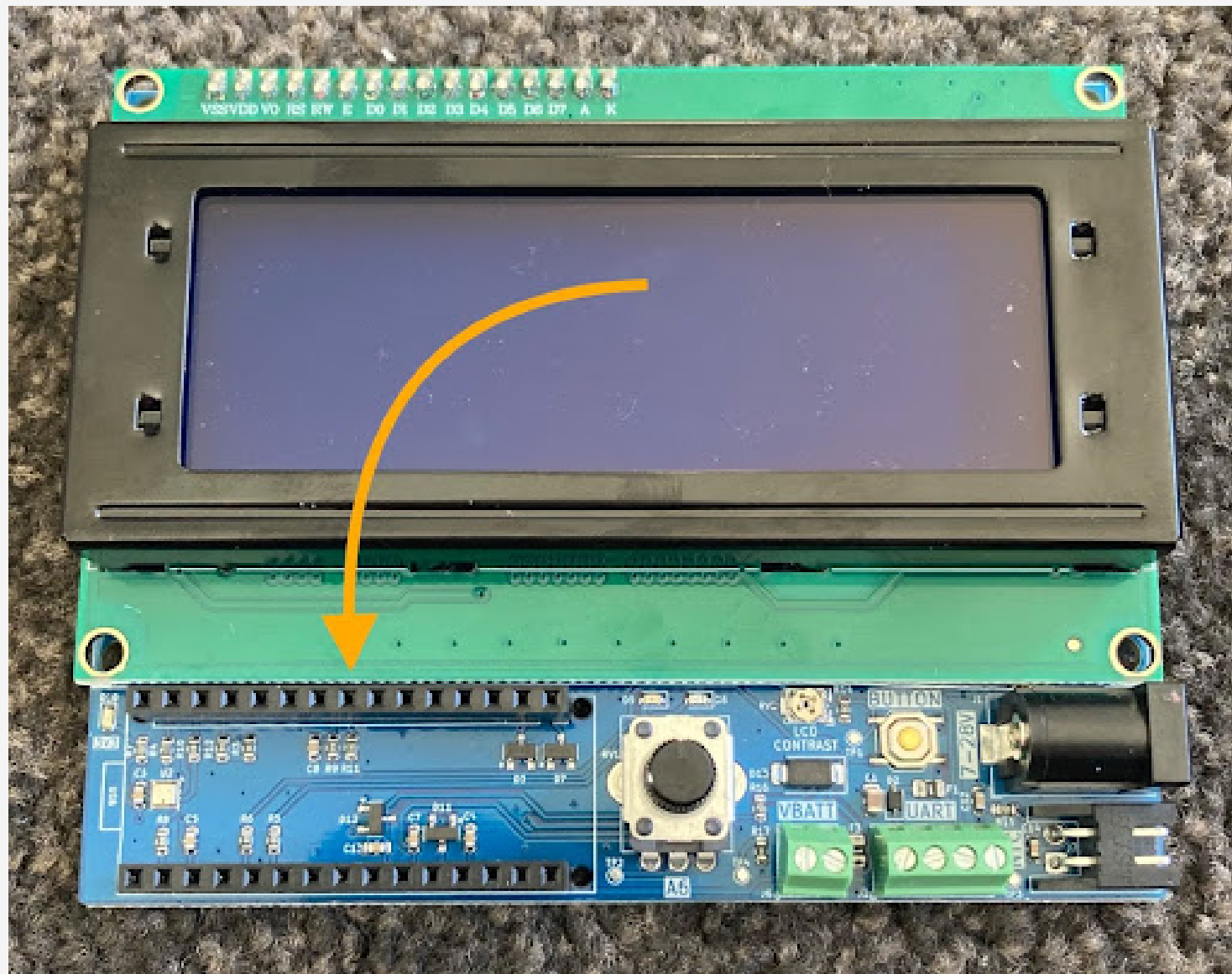
<https://www.arduino.cc/en/software/>

TASKS

- Open Arduino IDE on your computer.
- Go to evolocity.co.nz > Workshops > Introduction to Arduino.
- Download the Arduino Workshop Guidebook.
- Turn to the Getting Started section.
- Start working through the exercises.



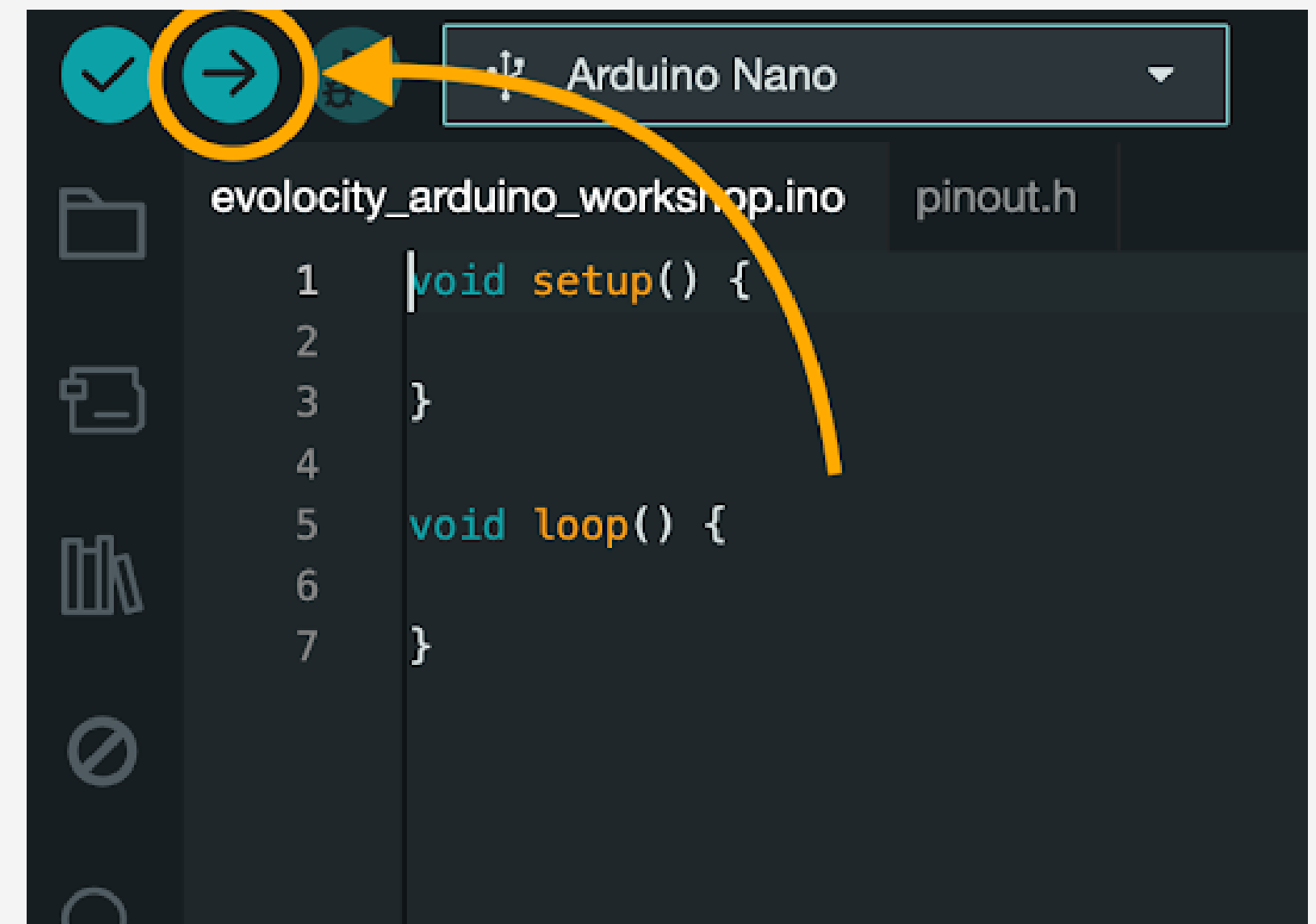
Attach the Arduino





Test your kit

1. Connect the Arduino to your computer
2. Load the example sketch “Blink”
(File>Examples>01.Basics>Blink)
3. Click “Select Board” and search for “Arduino Nano”
4. Click the right arrow (the “Upload” button)
5. Watch the LED on the Arduino





1.1 Blinking an LED

1. Run example 1.1
2. Try to explain what you think the code is doing to a partner
3. Do the “Try This” section

```
1  #define LED_Y A0 // Define our LED pins
2  #define LED_R A1
3
4  void setup() {
5      pinMode(LED_Y, OUTPUT); // Initialise our pins as outputs
6      pinMode(LED_R, OUTPUT);
7  }
8
9  void loop() {
10     digitalWrite(LED_Y, HIGH); // Turn the yellow LED on
11     digitalWrite(LED_R, LOW);  // Turn the red LED off
12     delay(1000);               // Wait 1000ms
13     digitalWrite(LED_Y, LOW);  // Turn the yellow LED off
14     digitalWrite(LED_R, HIGH); // Turn the red LED on
15     delay(1000);               // Wait again
16 }
```



1.2 Reading a Button

1. Run example 1.2
2. Try to explain what you think the code is doing to a partner
3. Do the “Try This” section

```
1  #define LED_ONBOARD 13 // Define our pins
2  #define BTN 6
3
4  void setup() {
5      pinMode(LED_ONBOARD, OUTPUT); // Initialise our pins
6      pinMode(BTN, INPUT);
7  }
8
9  void loop() {
10     int buttonState = digitalRead(BTN); // Read the state of the button
11
12     if (buttonState == HIGH) { // If our button is pressed...
13         digitalWrite(LED_ONBOARD, HIGH); // Turn the LED on
14     } else { // Otherwise...
15         digitalWrite(LED_ONBOARD, LOW); // Turn it off
16     }
17 }
```



2.1 Sending a Serial Message

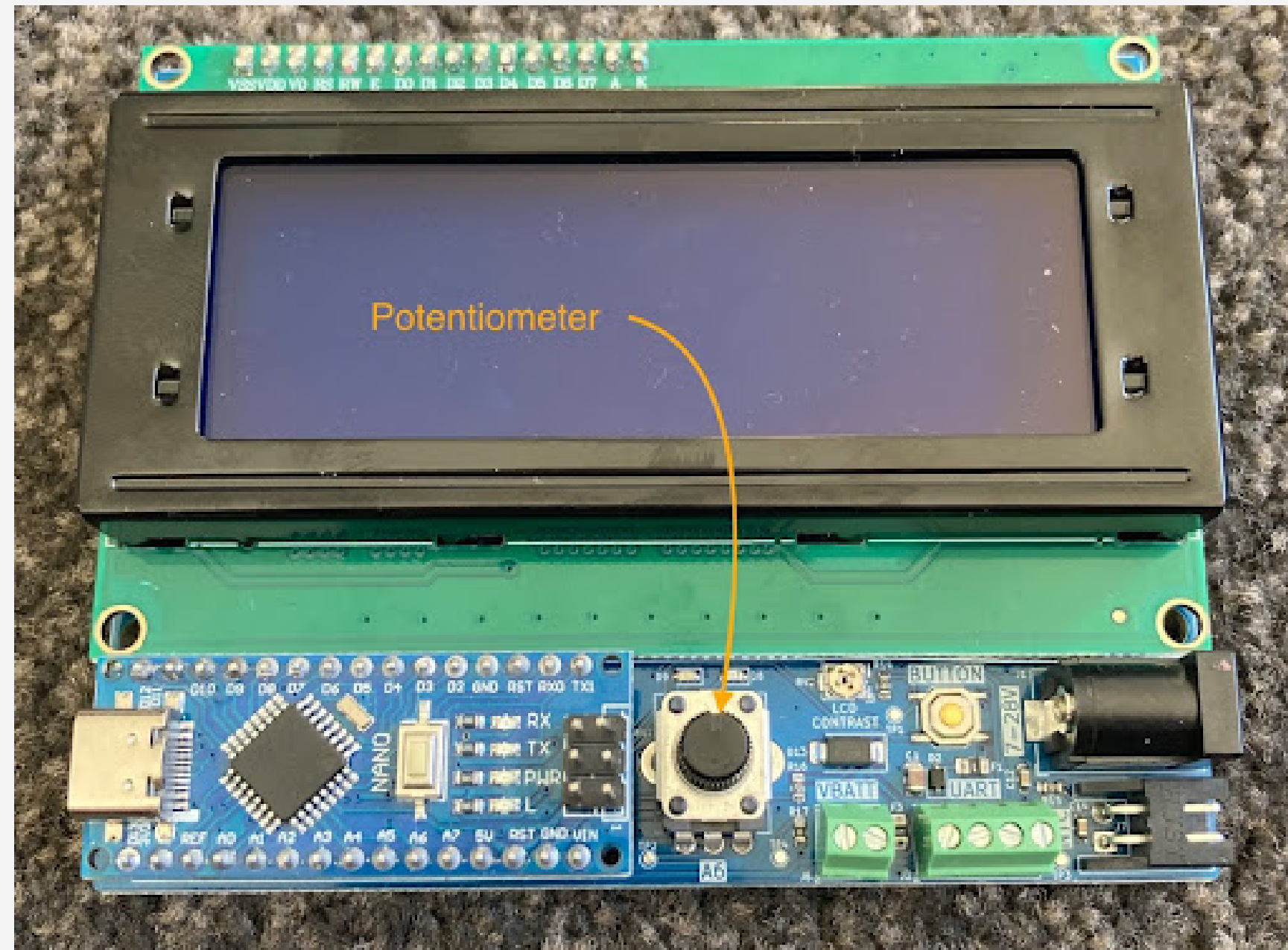




2.1 Sending a Serial Message

```
1 void setup() {  
2     Serial.begin(9600); // Start the Serial communication at 9600 baud  
3 }  
4 void loop() {  
5     Serial.println("Hello, world!"); // Print "Hello, world!" to the Serial Monitor  
6     delay(1000); // Wait for one second  
7 }
```

3.1 Reading the Dial





3.1 Reading the Dial

```
1  #define DIAL A6 // Define our pin
2
3  void setup() {
4      Serial.begin(9600); // Initialise Serial comms
5      pinMode(DIAL, INPUT); // Initialise our potentiometer pin
6  }
7
8  void loop() {
9      int dial_value = analogRead(DIAL); // Read the pin voltage
10     Serial.println(dial_value); // Print the result
11 }
```

3.2 Using a Potentiometer to Control LEDs



```
1  #define LED_Y A0
2  #define LED_R A1
3  #define DIAL A6
4
5  int dial_value;
6
7  void setup() {
8      pinMode(LED_Y, OUTPUT); // Initialise the yellow LED.
9      pinMode(LED_R, OUTPUT); // Initialise the red LED.
10     pinMode(DIAL, INPUT); // Initialise the potentiometer.
11 }
12
13 void loop() {
14     dial_value = analogRead(DIAL); // Read the potentiometer voltage.
15
16     if(dial_value < 400) { // If the dial is pointing left...
17         digitalWrite(LED_Y, HIGH); // Turn on the yellow LED.
18         digitalWrite(LED_R, LOW); // And make sure the red LED is off.
19     } else if (dial_value < 600) { // If the dial is about midway...
20         digitalWrite(LED_Y, LOW); // Make sure the yellow LED is off.
21         digitalWrite(LED_R, LOW); // Make sure the red LED is also off.
22     } else { // If neither of the other cases are true...
23         digitalWrite(LED_Y, LOW); // Make sure the yellow LED is off.
24         digitalWrite(LED_R, HIGH); // Turn on the red LED.
25     }
26 }
```



3.3 Measuring Battery Voltage

Pretend that your dial is the battery voltage sensor.

Have a go at programming this yourself, using this equation to calculate the battery voltage.

```
float battery_voltage = analogRead(VBATT) * (120.0 + 10.0) / 10.0;
```

Use one of the previous examples to help you!



3.3 Measuring Battery Voltage

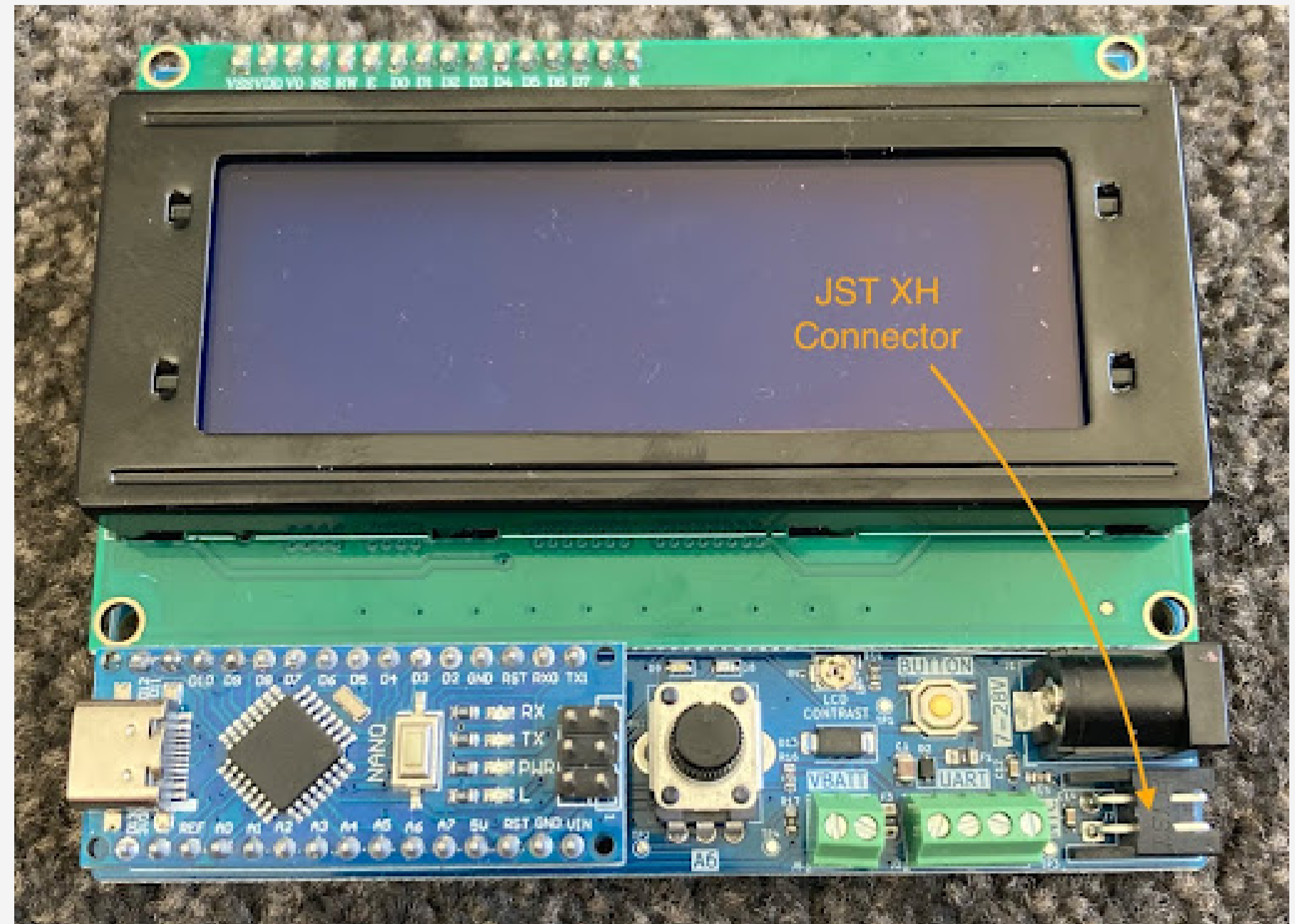
```
1  #define VBATT A5 // If we are simulating a voltage with the potentiometer, use A6 instead
2
3  float Vout; // Initialise variables
4  float Vin;
5
6  void setup() {
7      pinMode(VBATT, INPUT); // Initialise our VBATT pin.
8      Serial.begin(9600);
9  }
10
11 void loop() {
12     Vout = analogRead(VBATT) * (5.0 / 1023.0); // Convert the analog reading to a voltage.
13     Vin = Vout * (120.0 + 10.0) / 10.0; // Plug our voltage in to the voltage divider equation.
14     Serial.print("Voltage: "); // Print it all to our Serial Monitor
15     Serial.print(Vin);
16     Serial.println("V");
17     delay(1000);
18 }
```

3.4 Measuring Battery Temperature

Connect your thermistor to the JST XH connector on your motherboard.

Have a go at programming this yourself.

Use the guidebook for the required maths.





3.4 Measuring Battery Temperature

```
1  #define NTC_PIN A7
2
3  void setup() {
4      pinMode(NTC_PIN, INPUT);
5      Serial.begin(9600);
6  }
7
8  void loop() {
9      int analogReading = 1023 - analogRead(NTC_PIN); // Get the raw reading and invert it.
10     // The inversion is required because of onboard circuitry
11
12     float voltage = analogReading * (5.0 / 1023.0); // Convert analog signal to voltage.
13
14     float resistance = 5100 * ((5.0 / voltage) - 1); // Convert voltage to resistance.
15
16     float temp = 1 / ((1 / 298.15) + (1 / 3950.0) * log(resistance / 10000)) - 273.15;
17     // Using the Beta Parameter equation, calculate the temperature in degC.
18
19     Serial.print("Temp: ");
20     Serial.print(temp);
21     Serial.println(" C");
22     delay(1000);
23 }
```

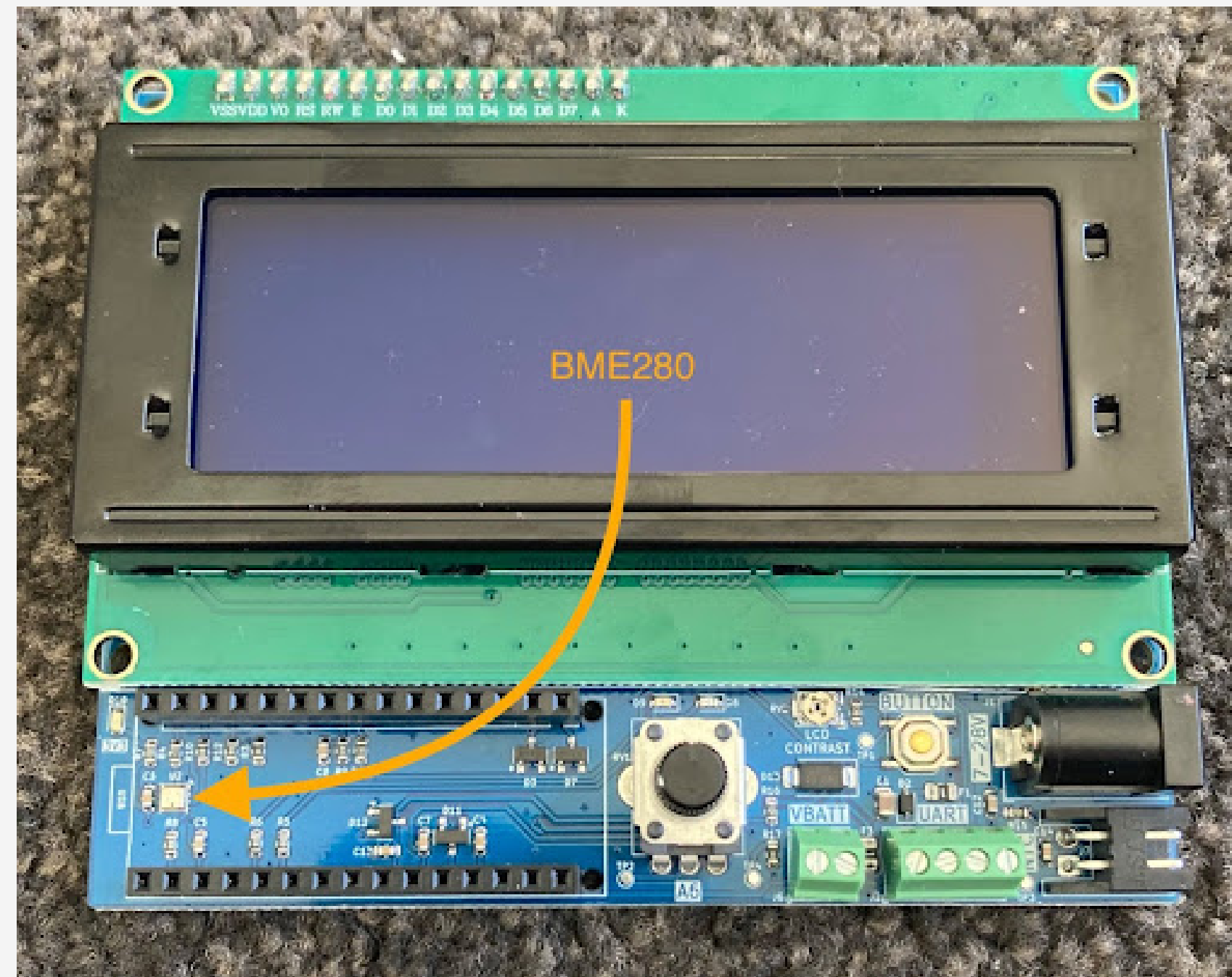


4.1 Communicating with an LCD

```
1  #include <LiquidCrystal.h>
2
3  #define LCD_RS 9
4  #define LCD_E 8
5  #define LCD_D4 5
6  #define LCD_D5 4
7  #define LCD_D6 3
8  #define LCD_D7 2
9  #define LCD_BACKLIGHT 7
10 #define LCD_WIDTH 20
11 #define LCD_HEIGHT 4
12
13 LiquidCrystal lcd(LCD_RS, LCD_E, LCD_D4, LCD_D5, LCD_D6, LCD_D7);
14
```

```
14
15  ✓ void setup() {
16      pinMode(LCD_BACKLIGHT, OUTPUT);
17      digitalWrite(LCD_BACKLIGHT, HIGH);
18      lcd.begin(LCD_WIDTH, LCD_HEIGHT);
19
20      lcd.clear();
21      lcd.setCursor(0, 0);
22      lcd.print("Hello, world!");
23  }
24
25  void loop() {
26
27  }
```

4.2 Measuring Data from the BME280 Sensor

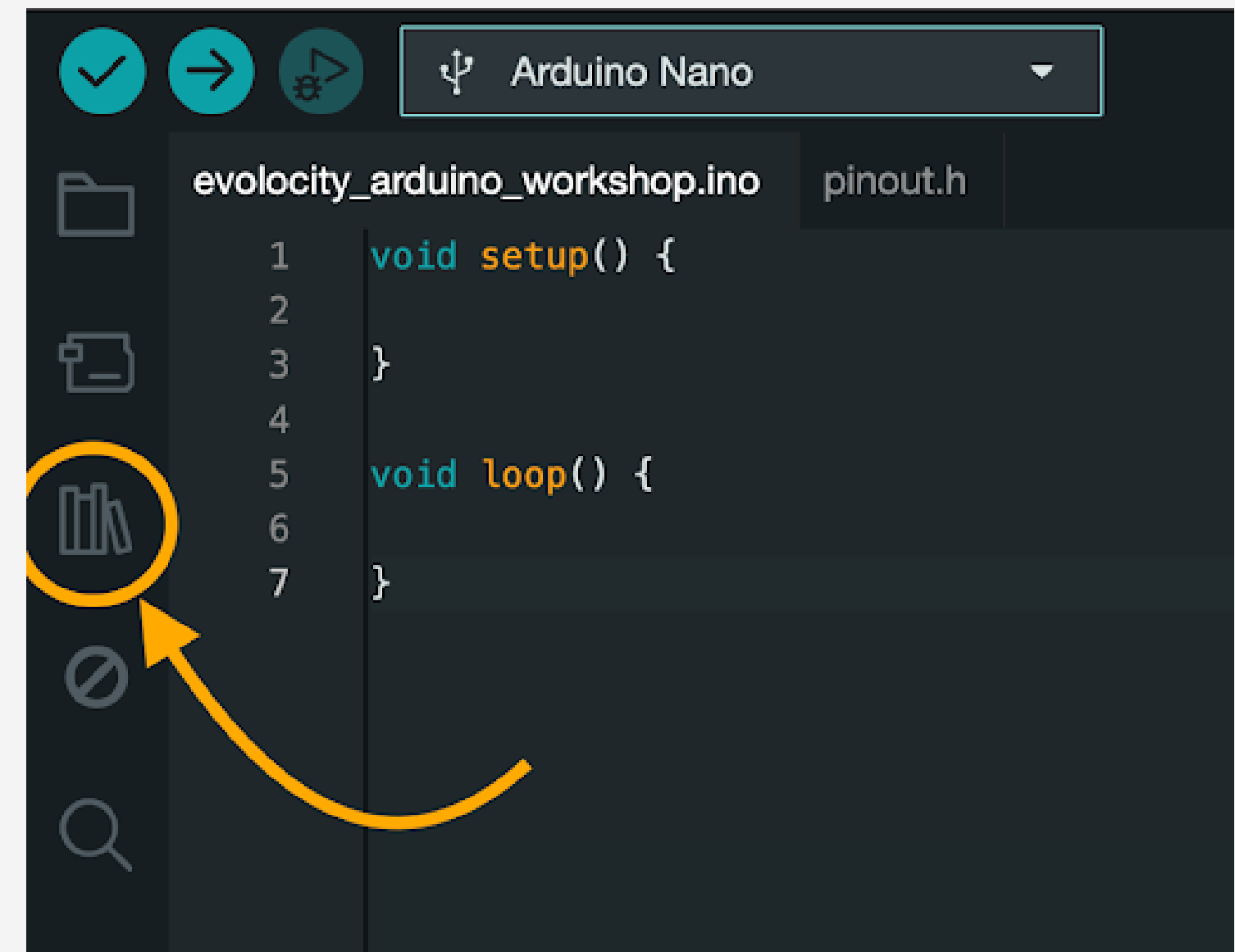




4.2 Measuring Data from the BME280 Sensor

1. Click the books icon on the left of your screen
2. Search for and install the “Adafruit BME280” library

Make sure to use version 2.3.0 to avoid compatibility issues during the workshop.





4.2 Measuring Data from the BME280 Sensor

Find the documentation.

Use the link in the guidebook, or google “Adafruit BME280 library Arduino”.

Note that we are using SPI, not I2C.

See if you can get a temperature reading from the BME280 sensor.



4.2 Measuring Data from the BME280 Sensor

```
1  #include <Adafruit_BME280.h>
2
3  #define BME_SCK 13
4  #define BME_MISO 12
5  #define BME_MOSI 11
6  #define BME_CS 10
7
8  Adafruit_BME280 bme(BME_CS);
9
10 void setup() {
11     bme.begin();
12     Serial.begin(9600);
13 }
```

```
14
15 void loop() {
16     float temperature = bme.readTemperature();
17     Serial.print("Motherboard Temp: ");
18     Serial.print(temperature);
19     Serial.println(" C");
20     delay(1000);
21 }
```



5 Putting It All Together

You can now:

- Measure battery voltage
- Measure battery temperature
- Measure ambient (BME280) temperature
- Display it all on-screen

Try to combine this into one program.

THANK YOU FOR COMING

EVOLOCITY

