ARDUINO/PROJECT

What is Arduino?

- A hardware and software company
- Originated in 2005 at Ivrea Interaction Design Institute (northern Italy) as an easy tool for fast prototyping aimed at students without a background in electronics and programming.
- Designs and sells singleboard microcontrollers and kits – the design of many of them is open-source.
- License other companies to do the same: Adafruit, Sparkfun
- Open source electronics platform – both hardware and software.

Microprocessor

- CPU is standalone. Memory, IO, timers, etc. are separate
- System designer can add memory & IO
- High processing power
- High power consumption
- 32/64 bit
- General purpose

Microcontroller

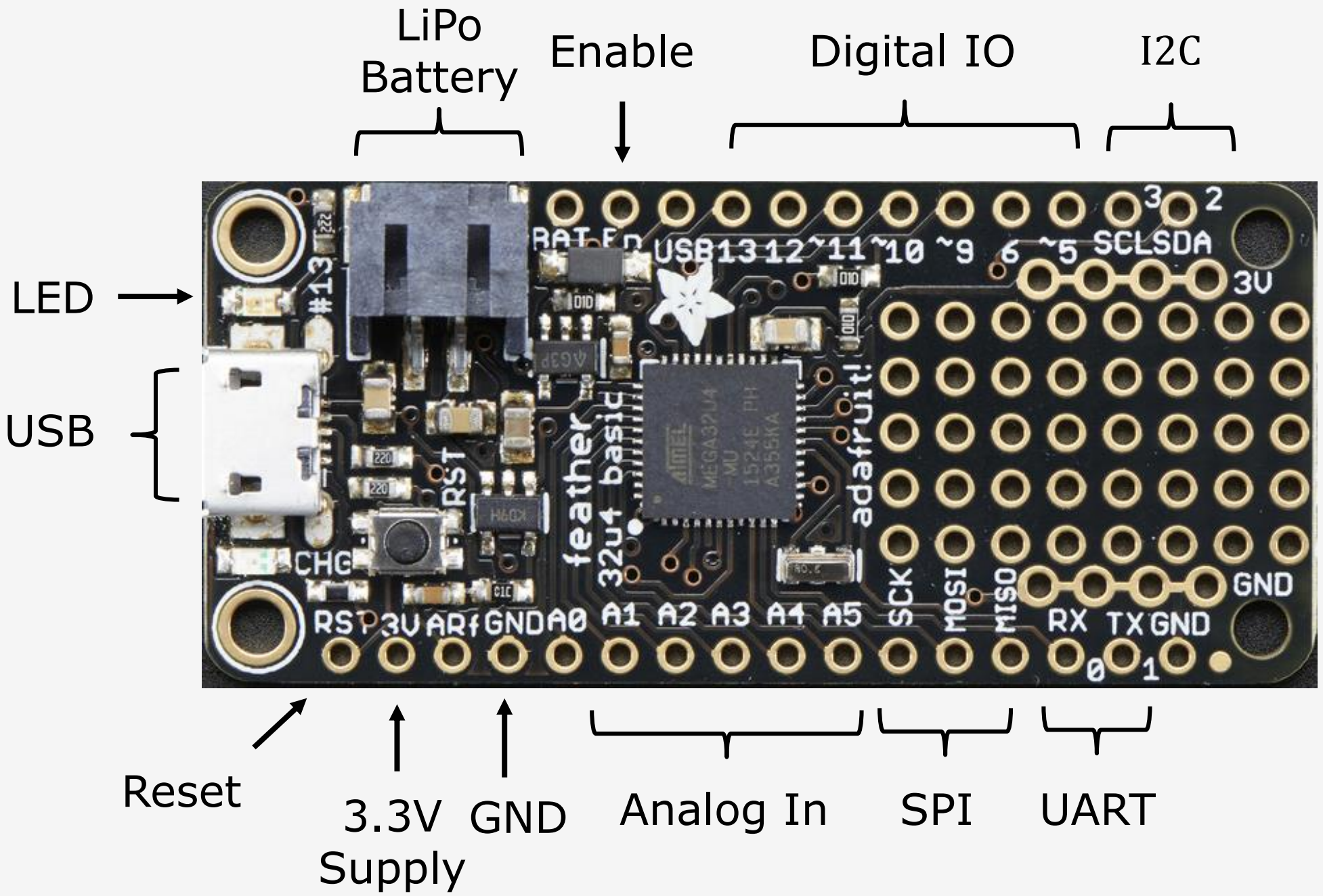
- CPU, memory, IO, timers, etc. are integrated on a single chip
- Memory & IO are fixed
- Low processing power
- Low power consumption
- 8/16 bit
- Single purpose (embedded)
- Within the capabilities of DIY

Embedded Systems - definition

An “embedded system” is the computing part of a larger mechanical or electrical system.

- digital watch
- vending machine
- MP3 player
- traffic light controller
- automobile electronics
- medical imaging
- avionics
- graphic controllers

The increasing use of embedded systems (computing replacing mechanical and electrical functionality) provided a market for microcontrollers that were relatively simple to program and use.


LiPo Battery
Enable
Digital IO
I2C
LED
USB
Reset
3.3V Supply
GND
Analog In
SPI
UART
#13
RST
CHG
feather
32u4 basic
adafruit!
RST 3V AREF GND A0 A1 A2 A3 A4 A5 SCK MOSI MISO RX TX GND
USB 13 12 ~11 ~10 ~9 6 ~5 SCL SDA 3 2
3V
GND
0 1

Sketch - Arduino's name for a computer program

```
//include libraries
#include <math.h>

//declare and initialize global variables and contants
int var1 = 0;
#define baudrate  9600

void setup() {
    //code runs once
    Serial.begin(baudrate);
}

void loop() {
    //main code here, to run repeatedly
    String greeting = "Hello World";   //local variable
    Serial.println(greeting);
}
```

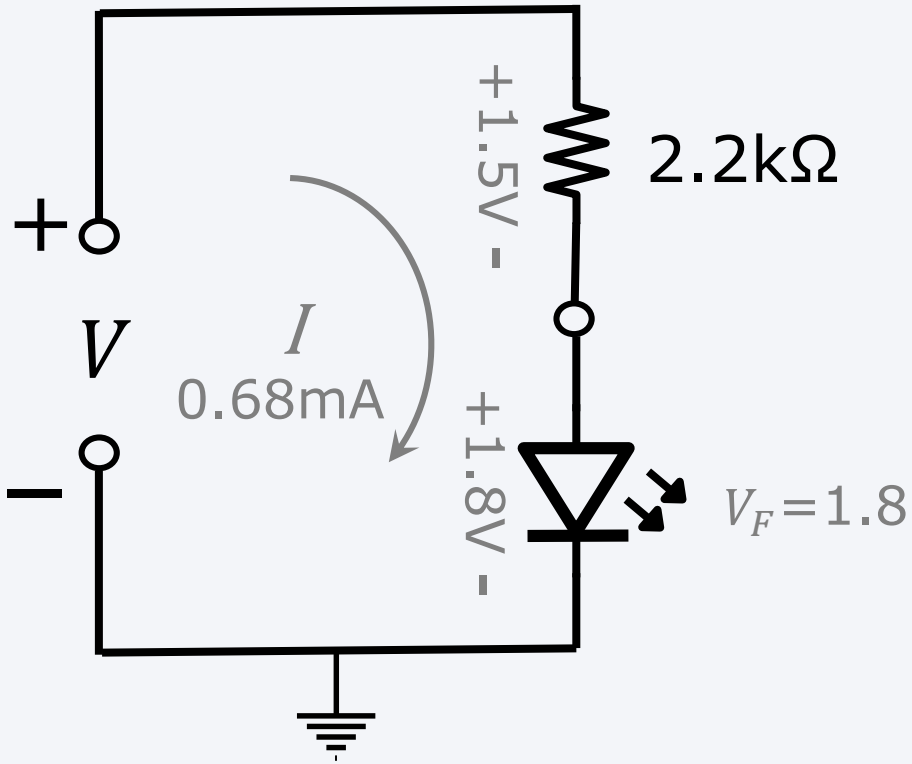
Functions – reduce redundant code

```
void loop() {
    int result;
    result = addTwoNumber(63,8);
    Serial.println(result);
}

int addTwoNumbers(int x, int y) {
    int z;
    z = x + y;
    return z;
}
```

PROJECT

Blink

```
// the setup function runs once when you power the board
void setup() {
      // initialize digital pin 13 as an output.
      pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
      digitalWrite(13, HIGH);   // set D13 HIGH – LED is turned on
      delay(1000);              // wait for a second
      digitalWrite(13, LOW);    // set D13 LOW – LED is turned off
      delay(1000);              // wait for a second
}
```



pinMode(pin, mode) mode: INPUT, OUTPUT, INPUT_PULLUP

digitalWrite(pin, value) value: HIGH, LOW

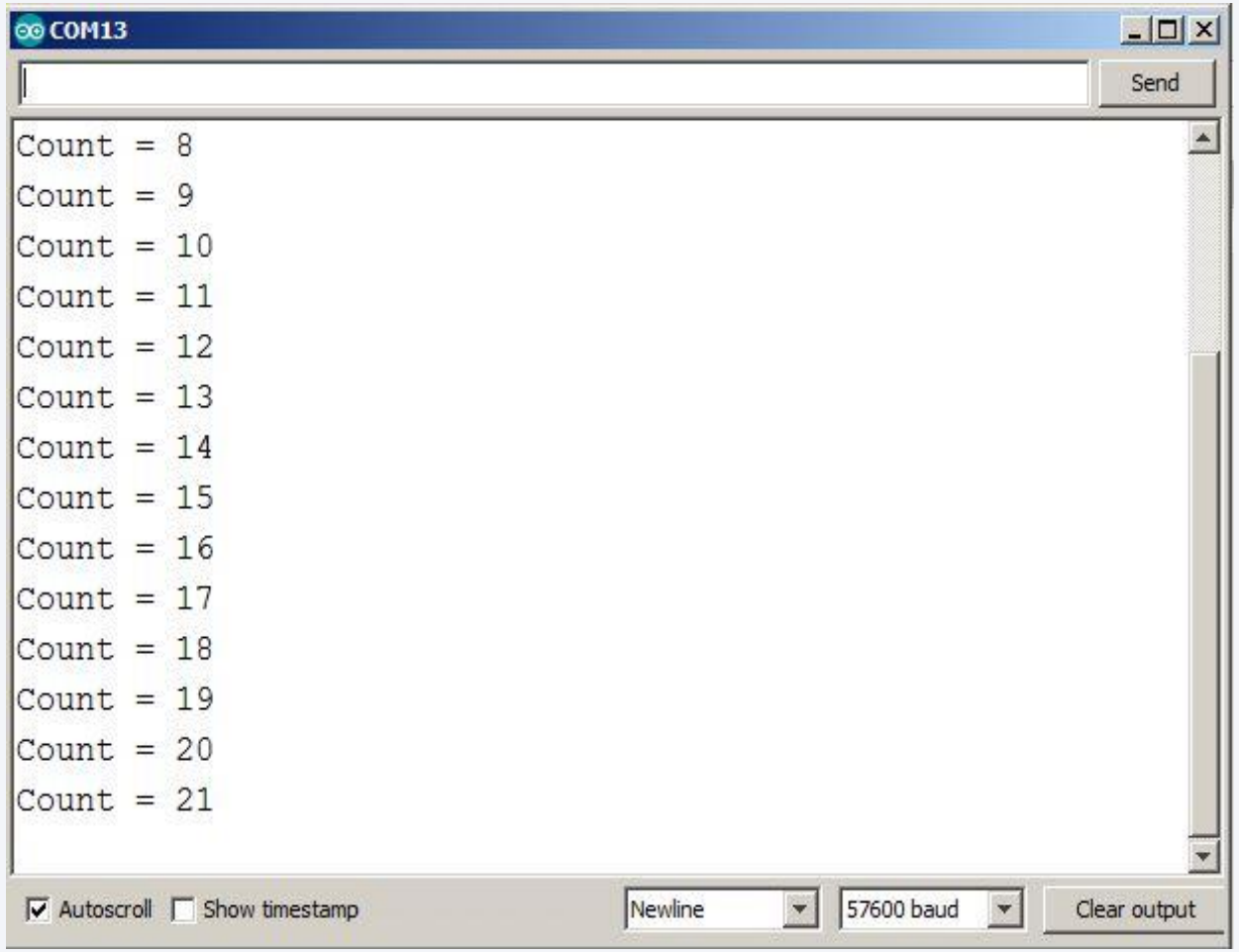
delay(ms) simply a pause in milliseconds

D13 High: V=3.3V
D13 Low: V=0V

Serial Monitor – comms between Arduino and your computer

```
//declare and initialize integer variable 'count'
int count = 0;

void setup() {
    Serial.begin(9600);
}

void loop() {
    Serial.print("Count = ");
    Serial.println(count);    //print the variable
    delay(1000);
    count++;    //increment count by 1
}
```

Tools -> Serial Monitor



RTC – set and use the real time clock

```
#include <RTClib.h>  //add the RTC library
RTC_PCF8523  rtc;   // Link to RTC


void setup() {
      Serial.begin(9600);
      rtc.begin(); //start up the real-time clock

      //set the clock
     rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}
void loop() {
     DateTime now = rtc.now();
     String timeStamp = formatTimestamp(now);
     Serial.println(timeStamp);  //print the variable
}
```

```
String formatTimestamp(DateTime now) {
      String result = "";
      result += now.year();
      result += "-" + now.month() + "-";
      result += now.day() + " ";
      result += now.hour() + ":";
      result += now.minute() + ":";
      result += now.second();
      return (result);
}
```

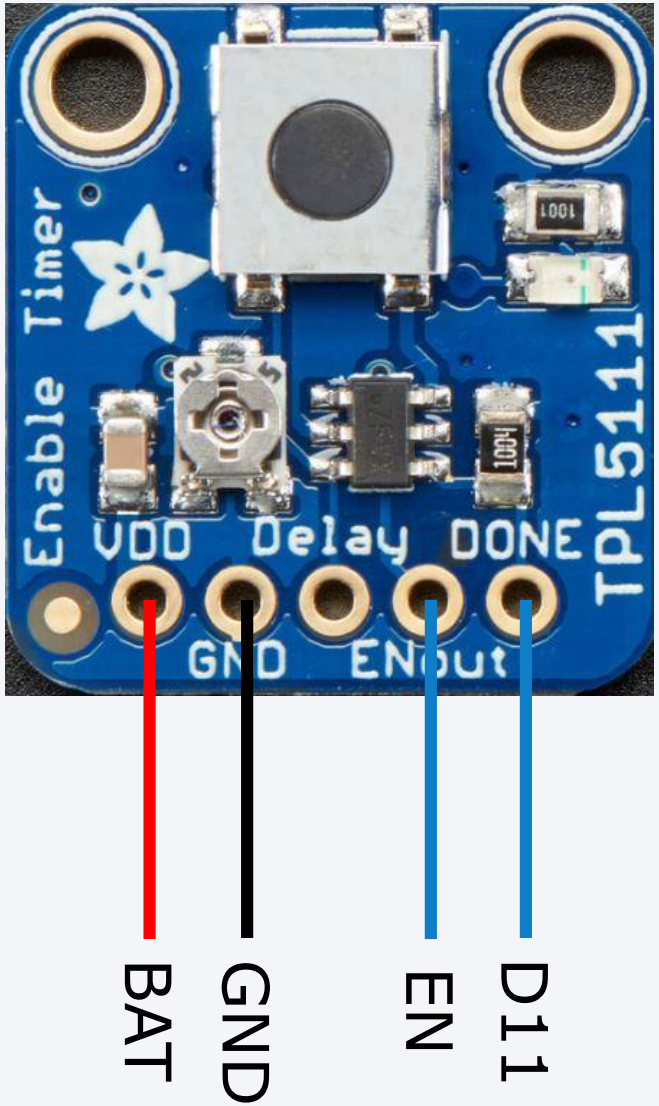
2019-9-23 13:47:0

SD – use the SD card

```
#include <SD.h>  //add the RTC library
      // The SD uses pin 10 on the Feather
const int SDchipSelect = 10;

void setup() {
      // Initialize the SD card
      SD.begin(); //start up the real-time clock
}

void loop() {
       // Open the output file for writing/appending
      File dataFile = SD.open("testfile.txt", FILE_WRITE);
      dataFile.println("Some text on the SD card");
      dataFile.close(); //close the CSV file
}
```

Timer

```
// Timer
#define DONEPIN 11

void setup() {
    //Configure the done pin
    pinMode(DONEPIN, OUTPUT);
    digitalWrite(DONEPIN, LOW);
}

void loop() {
    ...
     // When it's time to shut down...
    digitalWrite(DONEPIN, HIGH);
}
```



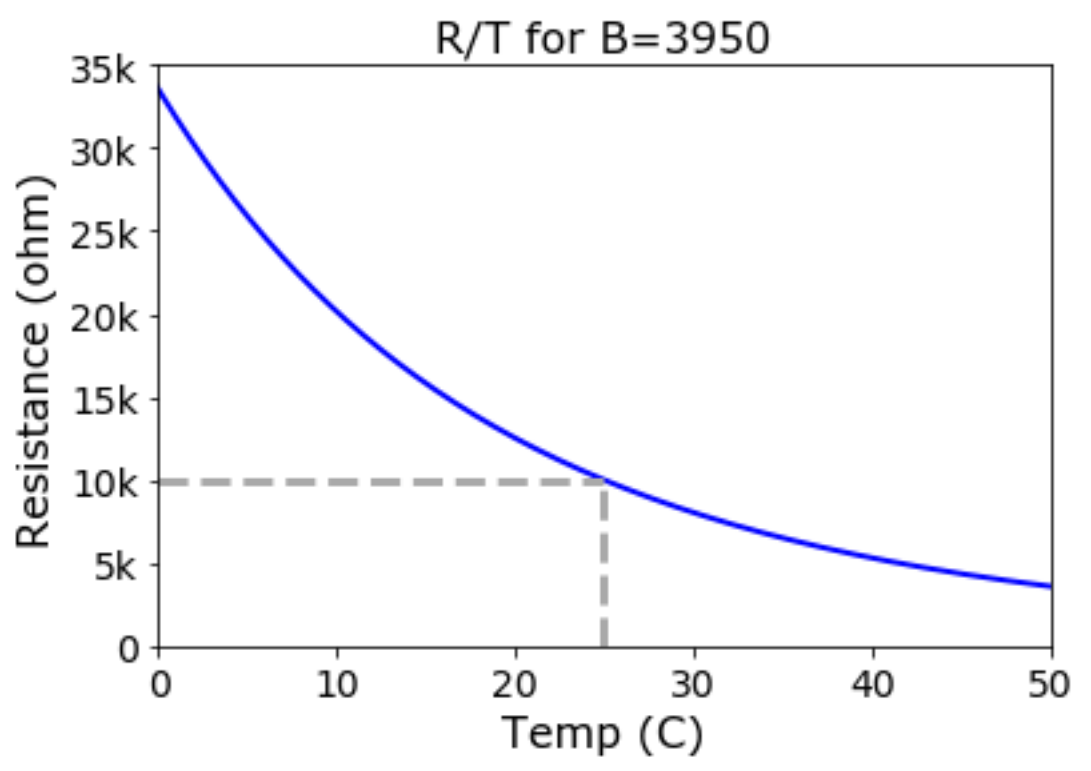
Timer

Table 3. Most Common Time Intervals Between 1s to 2h

t_{IP}	CALCULATED RESISTANCE (kΩ)	CLOSEST REAL VALUE (kΩ)	PARALLEL of TWO 1% TOLERANCE RESISTORS,(kΩ)
10s	11.20	11.199	15.0 // 44.2
20s	14.41	14.405	16.9 // 97.6
30s	16.78	16.778	32.4 // 34.8
40s	18.75	18.748	22.6 // 110.0
50s	20.047	20.047	28.7 // 66.5
1min	22.02	22.021	40.2 // 48.7
2min	29.35	29.349	35.7 // 165.0
3min	34.73	34.729	63.4 // 76.8
4min	39.11	39.097	63.4 // 102.0
5min	42.90	42.887	54.9 // 196.0
6min	46.29	46.301	75.0 // 121.0
7min	49.38	49.392	97.6 // 100.0
8min	52.24	52.224	88.7 // 127.0
9min	54.92	54.902	86.6 // 150.0
10min	57.44	57.437	107.0 // 124.0
20min	77.57	77.579	140.0 // 174.0
30min	92.43	92.233	182.0 // 187.0
40min	104.67	104.625	130.0 // 536.00

Thermistor

- Resistance at **25°C = 10kΩ**
- **B** 25/50 3950 ±1%
- Time constant ~ 15s



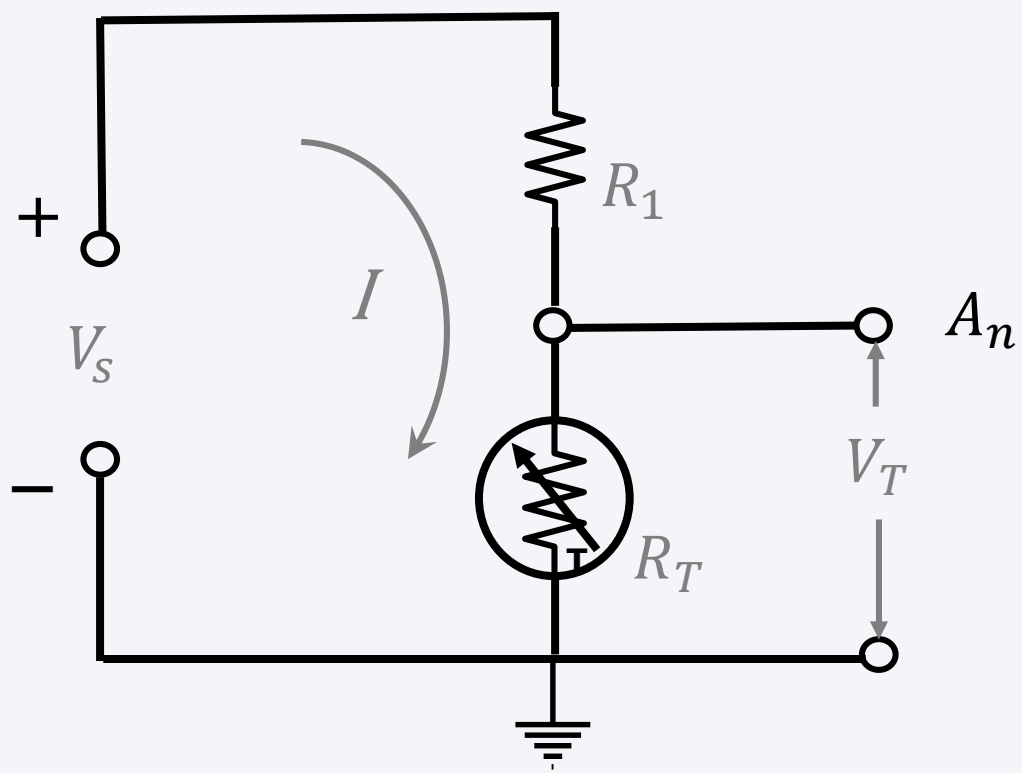
Adafruit.com

$$\frac{1}{T_K} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R_T}{R_0}\right)$$

$T_0 = (\mathbf{25} + 273.15)°K$

$R_0 = 10000$

Thermistor



$$V_T = IR_T = \frac{V_S \cdot R_T}{(R_1 + R_T)}$$

$$ADC\ value = \frac{V_{input}}{V_{ref}} \cdot 1023 = \frac{V_T}{V_S} \cdot 1023$$

$$V_T = \frac{ADC \cdot V_S}{1023} = \frac{V_S \cdot R_T}{(R_1 + R_T)}$$

$$\frac{ADC \cdot V_S}{1023} = \frac{V_S \cdot R_T}{(R_1 + R_T)}$$

$$R_T = \frac{R_1 \cdot ADC}{1023 - ADC}$$

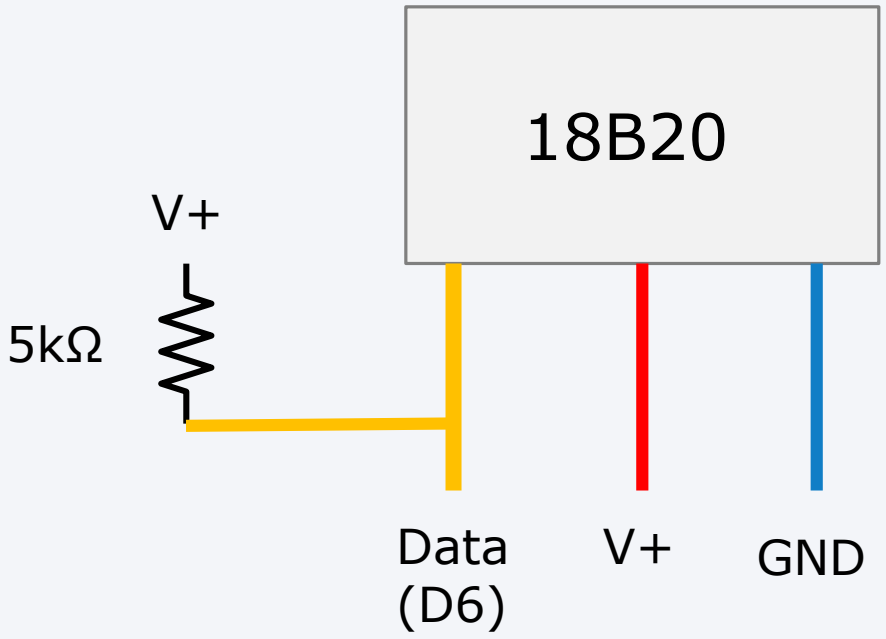
Thermistor

```
//set analog pin
#define thermPin A5

void setup() {
}

void loop() {
    //get the value from the ADC
    int thermADC = analogRead(thermPin);
    //calculate thermistor resistance
    float rTherm = 10000.*thermADC/(1023.-thermADC);
    //calculate T inverse in Kelvin
    float Tinv = (1/298.15) + (1/3950.)*log(rTherm/10000.);
    //convert to centigrade
    float thermTemp = (1/Tinv) – 273.15;
}
```

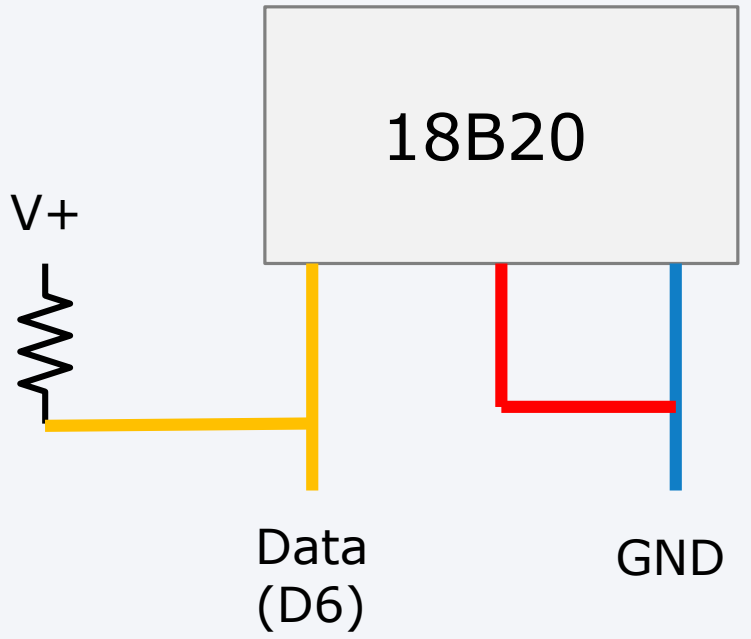
$$R_T = \frac{R_1 \cdot ADC}{1023 - ADC}$$

$$\frac{1}{T_K} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right)$$

Temp – DS18B20



Direct Power Mode



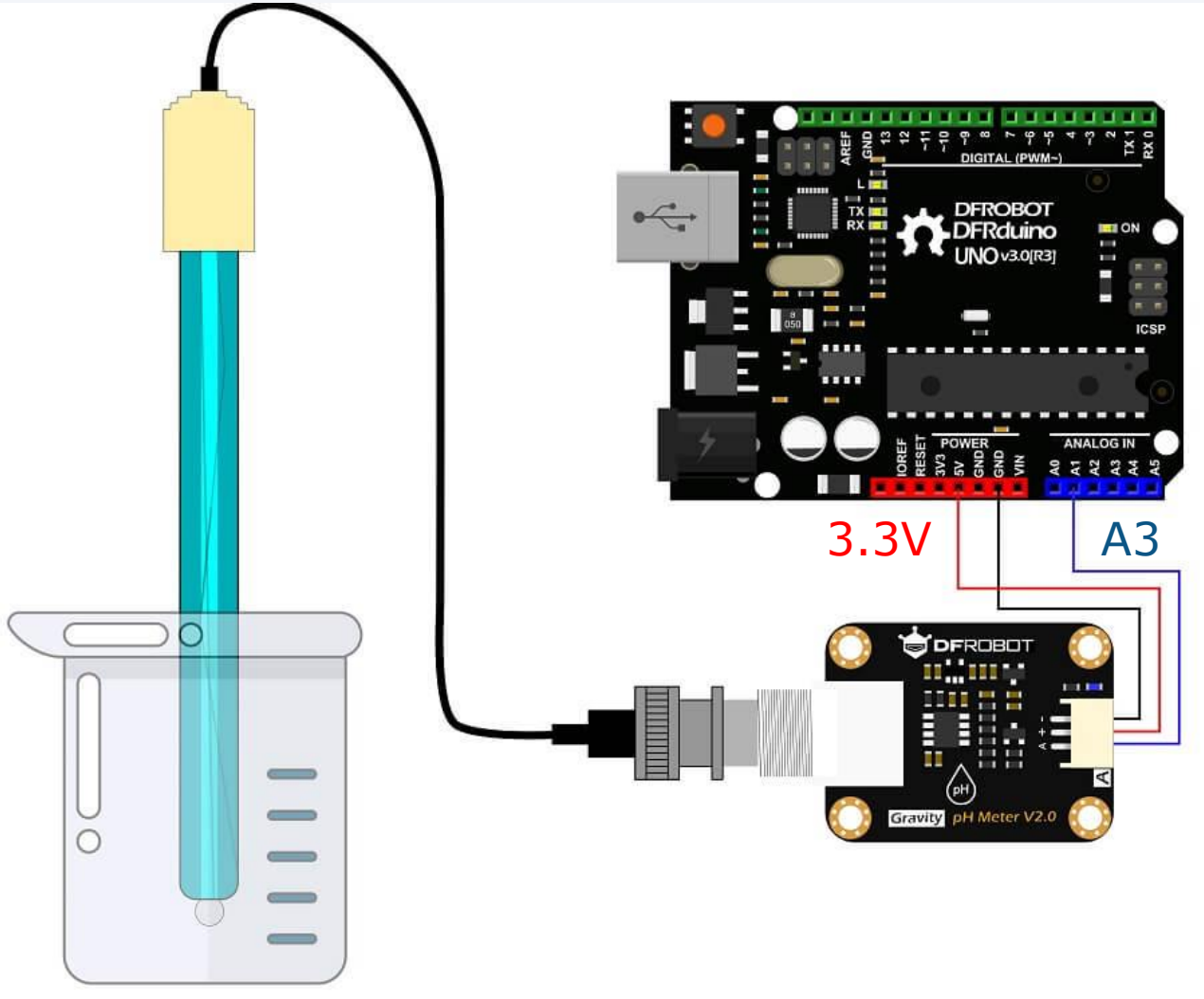
Parasitic Power Mode

Temp – DS18B20 using Dallas 1-Wire

```
#include <DallasTemperature.h>
//The One Wire bus is on D6
#define ONE_WIRE_BUS 6
// Link to One Wire
OneWire oneWire(ONE_WIRE_BUS);
// Link temp sensor to One Wire bus
DallasTemperature dallasTemp(&oneWire);
//Let's pretend it was calibrated; Temp reading at 0C
#define T0 1.06

void setup() {
    dallasTemp.begin()
}
```

```
void loop() {
    // Get the dallas temperature data
    dallasTemp.requestTemperatures();
    float tempRaw= dallasTemp.getTempCByIndex(0);
    // Calibration correction
    float tempCorr = tempRaw – T0;
}
```

pH



Interpolation with 2-point cal

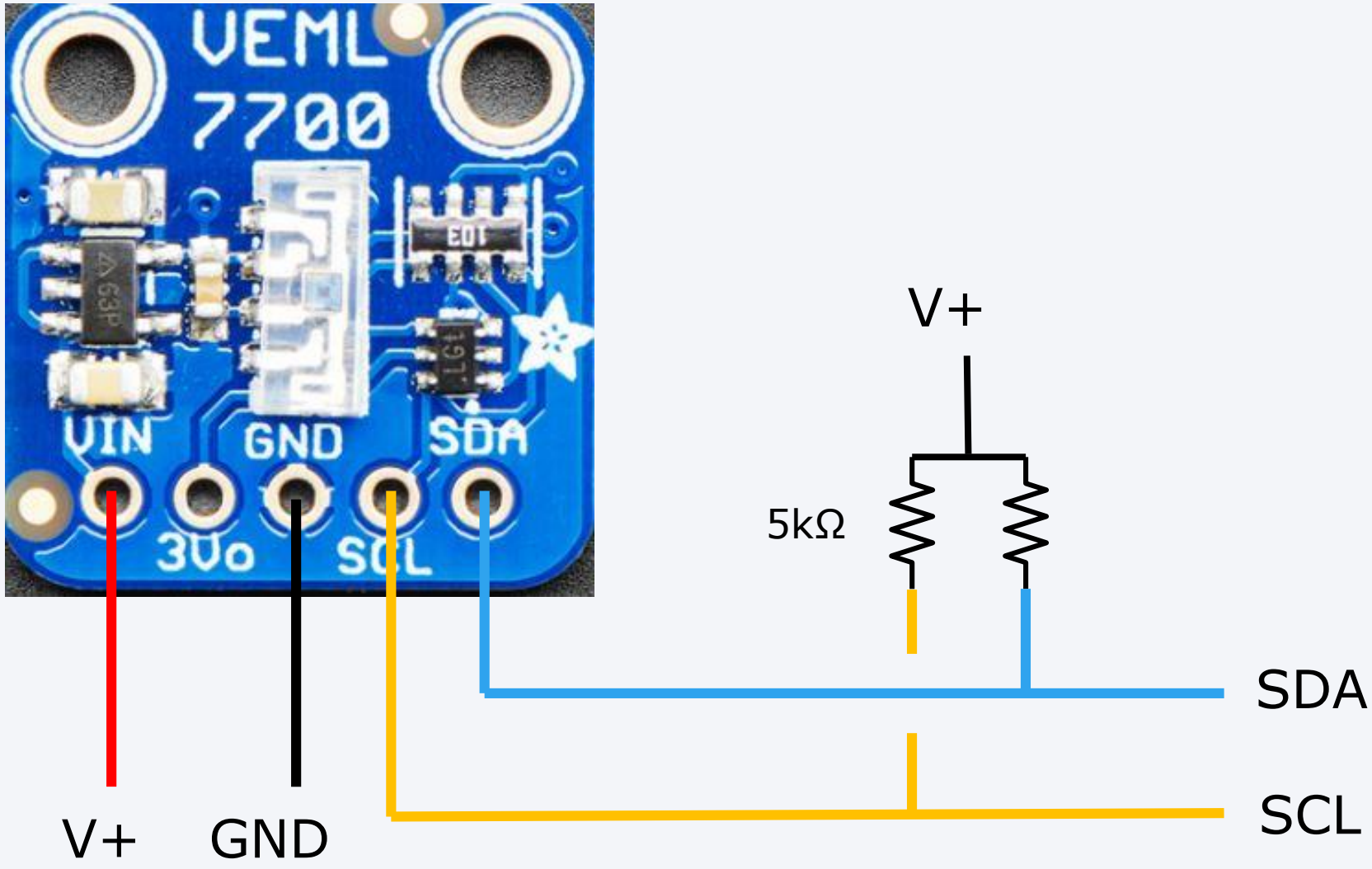
$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

y_i are pH values
x_i are ADC outputs

pH

```
#define pHpin A3
//Calibration results – values from 10-bit ADC
#define pH4 620
#define pH7 463

void setup() {
}

void loop() {
    // Get the pH ADC output from A5 (pHpin)
    float pHread = analogRead(pHpin); //0-1024 digital
    // Convert ADC output to corrected pH using two-point calibration results
    //corrVal =  (((rawValue-rawLow)*referenceRange)/(rawRange)) + referenceLow
    //where referenceRange = -3 and rawRange= pH4-pH7
    float pHcorr = (((pHread-pH7)*(-3))/(pH4-pH7)) + 7;
}
```

Lux



Lux – VEML7700

```
#include <Wire.h>
#include <Adafruit_VEML7700.h>
//Also install “Adafruit_BusIO” library

// Lux Sensor
Adafruit_VEML7700 veml = Adafruit_VEML7700();


void setup() {
    veml.begin();
}

void loop() {
    //get lux value
    float lux = veml.readLux();
}
```