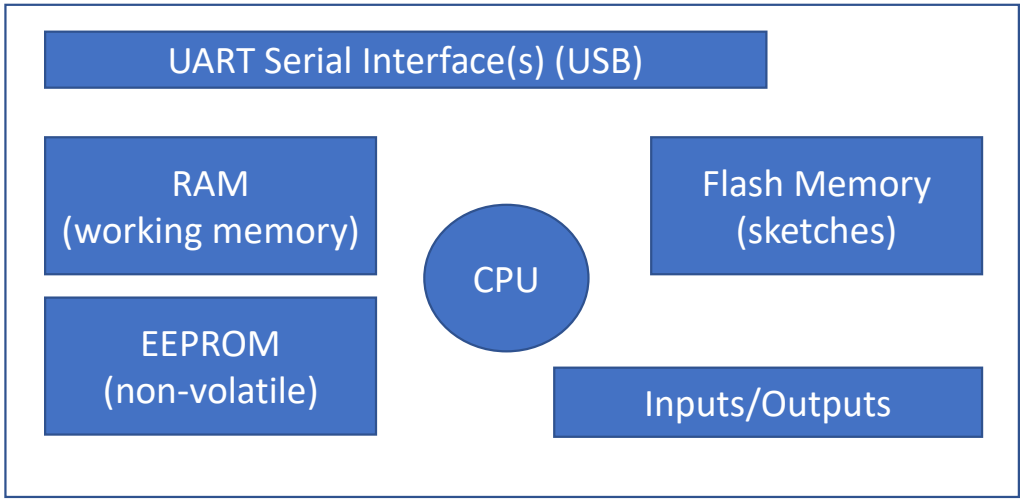
Milton Keynes Amateur Radio Society

Getting Started with Arduino Micro-Controllers

Tim Cowell G6GEI & Laszlo Sanduly M0BOY

What is Arduino?

- A Micro-controller with on-board Inputs/Outputs to interface things.

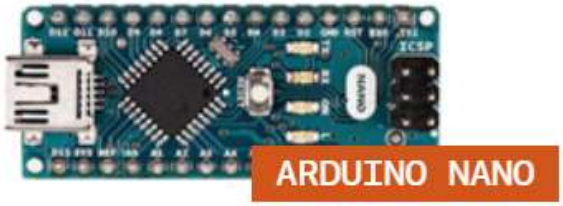


Open-source design = Low cost + lots of variants / manufacturers (and quality)

https://www.arduino.cc/

Different Models

ATmega328P processor.
14 digital I/O pins (6 PWM out) 6 analog in.
2KB RAM, 1KB EEPROM, 32KB FLASH



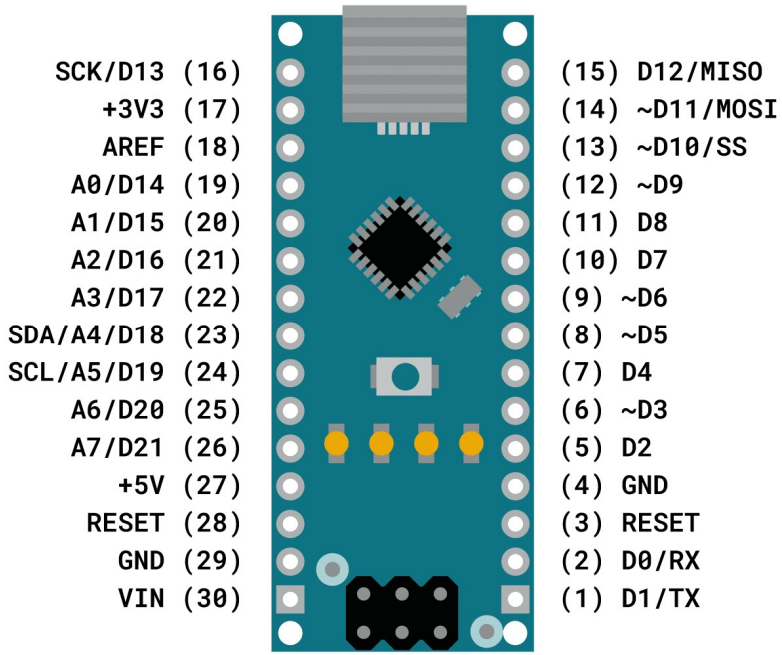
ATmega2560 processor.
54 digital I/O pins (15 PWM out) 16 analog in, 4 UARTS.
8KB RAM, 4KB EEPROM, 256KB FLASH

ATmega32u4 processor.
20 digital I/O pins (7 PWM out) 12 analog in.
2.5KB RAM, 1KB EEPROM, 32KB FLASH

And plenty more…

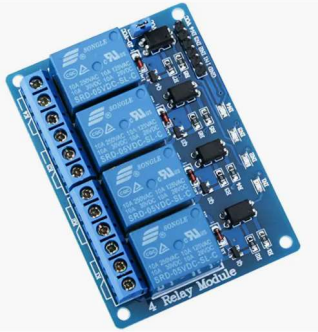
Determine pinout of your board

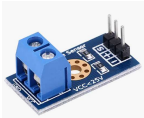
- Digital pins D0..D21…
- Analog pins A0..A7…
- Power – either via USB or Vin
- PWM pins ~ (analog out)
- Interrupt (D2/D3)
- UART (USB) D0/D1
- I2C bus (A4/A5) (addressable bus)
- SPI (SCK/MISO/MISI/SS)
- AREF (Analog Ref) more later…

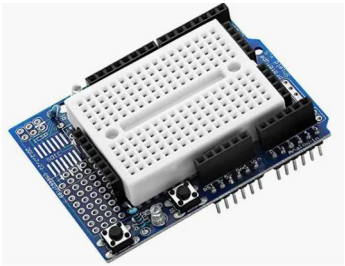


An Interface for every purpose

- Communication
  - Ethernet, WiFi, BlueTooth, Serial
- Displays
  - LCD/LED/Touchpanel
- Inputs (digital/analog)
  - Voltage, Current, Temperature
  - Movement, Compass, Humidity
- Outputs (digital/PWM)
  - Motor control, lighting, relay…
- Note interface type and pins used…
  - I2C / SPI etc.
- Note the voltage 3.3v or 5v
  - Can damage device if not careful





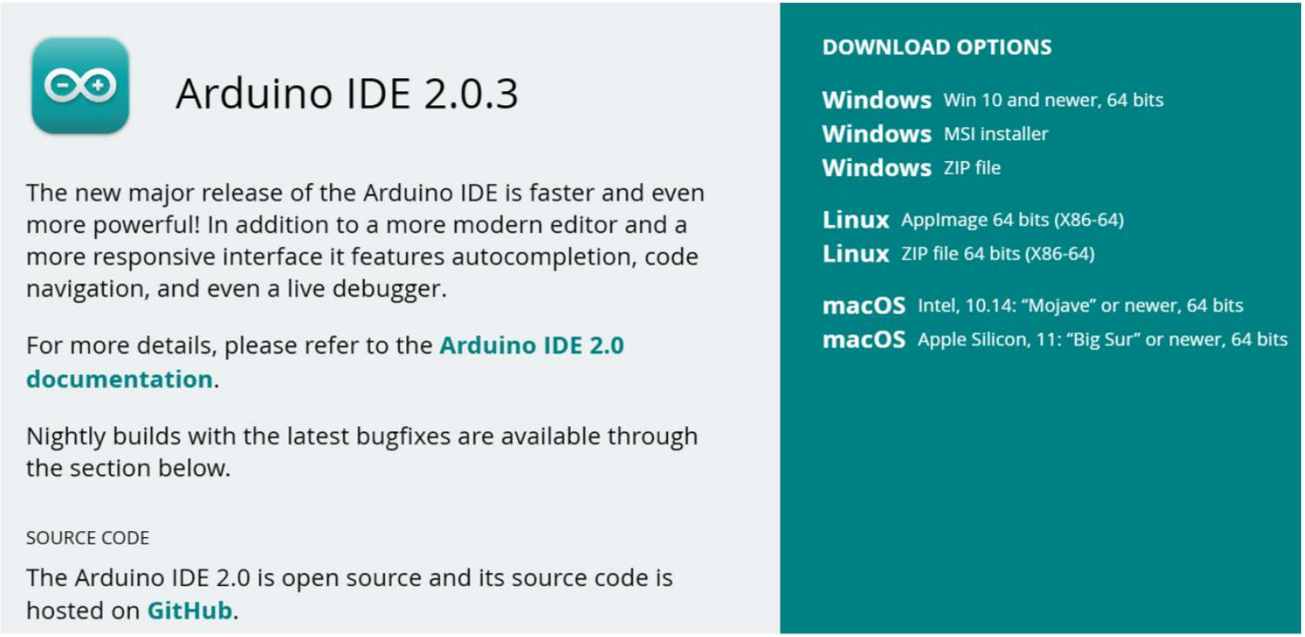


Easy to program.

- Arduino IDE (Integrated Development Environment)
  - Comes with loads of example programs (Sketches)
  - Windows / Linux / macOS
- C++ Programming language (MicroPython also avail)
  - easy to read/learn for simple tasks
  - powerful for more complex tasks
- HUGE set of open-source examples to crib from or just use.
- Compile your code
- Connect to PC via USB -> Submit compiled code
- Disconnect from PC = Turnkey system

Installing the IDE

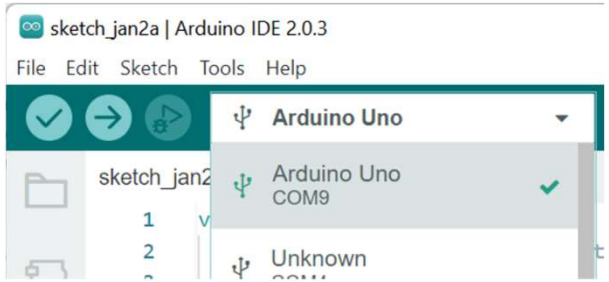
- Visit https://www.arduino.cc/en/software
- Download version for your PC
- Run the installer
- Run the IDE and wait
- Installs drivers etc.



Configuring the IDE

- AFTER installing IDE, connect Arduino USB
- Run the IDE

- The IDE should detect your board type and port number, if not select it now. Different Arduinos have different CPU and resources, so it's critical to get this right.



(On the laptops provided.. Login as Mkars M5mk!dx the Arduinos are the Nano type)

Your first program

- Let's make the LED Flash

- setup()
  - Code runs once at startup
  - Configure pins etc.
- loop()
  - Runs forever – repeating
- void is a datatype that has 'no result/type'
- All statements end with ;
- LED_BUILTIN, OUTPUT, HIGH, LOW are pre-defined Constants

Test your code compiles

- Compile your code.. Click the top left tick on the toolbar 

```
Output
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.
```

- Or if you've made a mistake…

```
C:\Users\reneg\AppData\Local\Temp\.arduinoIDE-unsaved202302-17712-2z6088.bmzjn\sketch_jan2a\sketch_jan2a.ino: In function 'void loop()':
C:\Users\reneg\AppData\Local\Temp\.arduinoIDE-unsaved202302-17712-2z6088.bmzjn\sketch_jan2a\sketch_jan2a.ino:10:3: error: expected ';' before 'digitalWrite'
  digitalWrite(LED_BUILTIN,LOW);    //turn the LED Off
  ^~~~~~~~~~~~

exit status 1

Compilation error: expected ';' before 'digitalWrite'
```

Compilation error: expected ';' before 'digitalWrite'

COPY ERROR MESSAGES

- If I missed the ; on the line preceding digitalWrite(

Send it to the Arduino

- Click the Upload button on the toolbar 
- Note it will also compile before uploading
- The code should now run (flash led)

- Try changing your code (Sketch) to make the LED blink slower or faster or flash some morse code…..

- Save your sketch File/Save As ‘Presentation_1’
- Creates a Presentation_1 folder in MyDocuments\Arduino and in the folder is a file Presentation_1.ino

Debugging / Tracing

- With Version 2 of the IDE there is a Debug mode that allow interactive debugging. But not supported on all Arduino models.
  https://docs.arduino.cc/software/ide-v2/tutorials/ide-v2-debugger
- Instead can use Serial port to send Debug messages to IDE
- Serial.begin(9600)
- Serial.println("setup")

- IDE Tools\Serial Monitor to display the output

```
void setup() {
  // put your setup code here, to run once:
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("setup");
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED_BUILTIN, HIGH);  //turn the LED On
  delay(500);                      //wait 500 milliseconds
  digitalWrite(LED_BUILTIN,LOW);    //turn the LED Off
  delay(500);                       //wait 500 milliseconds
  Serial.println("loop");
}
```

The USB Serial port

- The USB port can be used for general IO
  - Sending / Receiving commands e.g. Rotator Control commands etc.

  - char input
    Variable of type char (character)
  - Serial.available()
    True if a character has been received
  - Serial.read()
    Gets the most oldest character from the receive buffer.
- Try adding the input code to echo what was sent back to the output

```
char input;

void setup() {
  // put your setup code here, to run once:
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
  Serial.println("setup");
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED_BUILTIN, HIGH);  //turn the LED On
  delay(500);                      //wait 500 milliseconds
  digitalWrite(LED_BUILTIN,LOW);    //turn the LED Off
  delay(500);                       //wait 500 milliseconds

  if(Serial.available()){
    input = Serial.read();
    Serial.print("You typed: " );
    Serial.println(input);
  }
}
```

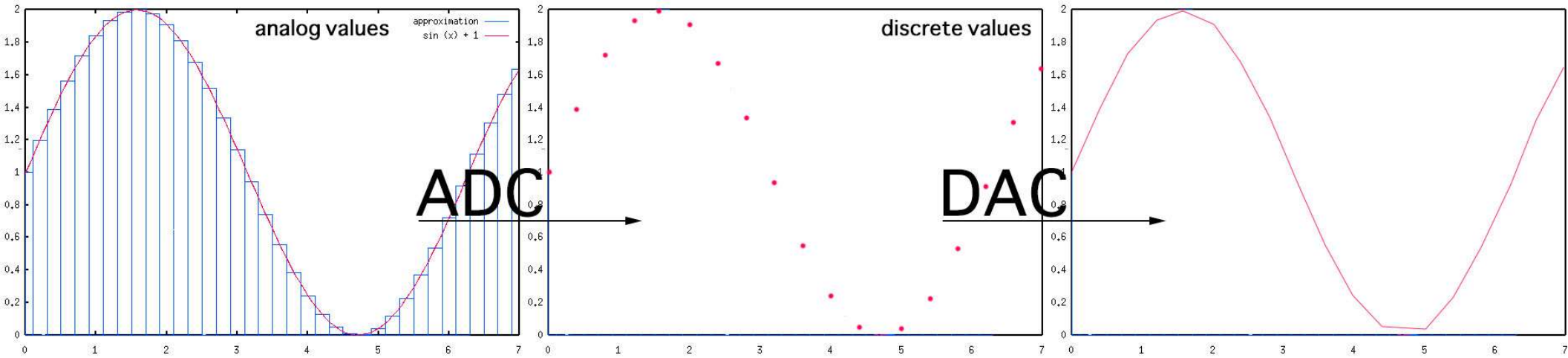
Analog Inputs

- The analog pins are inputs are connected to a 10bit A to D
- Reading returns a value between 0 and 1023
- The value represents a scale between 0v and the Reference Voltage.
- By default the Reference voltage is 5v.
- So an input of:
  - 0v will return 0
  - 5v will return 1023
  - 2.5v will return 512

- Simple!

Hmm… What is an ADC?

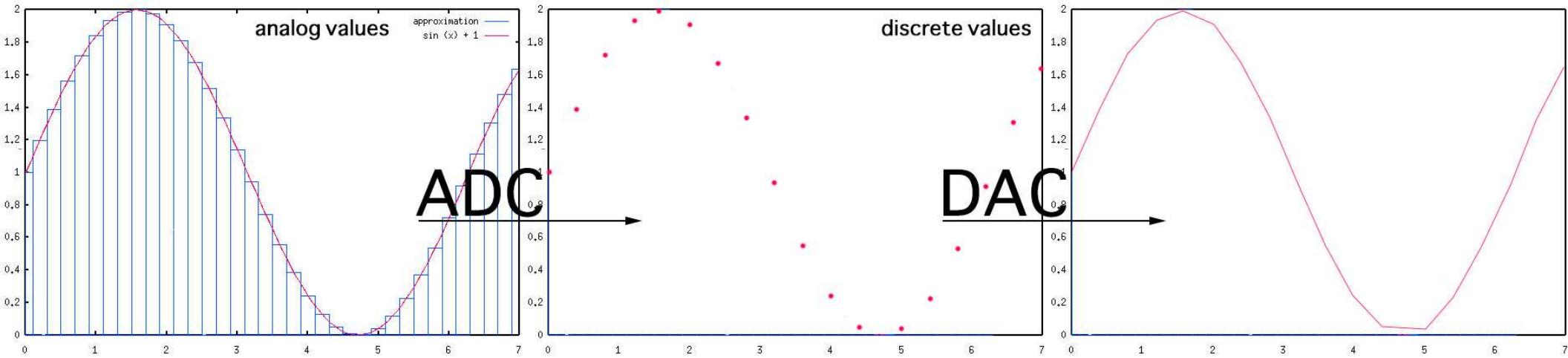
- A circuit which converts a continuous analog signal into discrete samples (approximation of the value at a given time)
- Sampling at a constant and stable ratio the discrete dots can be stored as values. In Arduino IDE the command to read an analog signal is **analogRead();**
- These dots can be used to restore the signal shape and value.



A set of input values consisting of all numbers in an interval. A set of input values consisting of only certain numbers in an interval. An approximation of the original analog values.

Hmm… What is an ADC?

- The “quality” of an ADC is given by the resolution and sampling rate.
  - The resolution is given by the minimum voltage change it can read as different numbers where the sampling rate is defined by the number of readings it takes in a given timeframe (usually defined as 1 second).
  - We will often hear terms as “samples per second” or “Megasamples per second”.



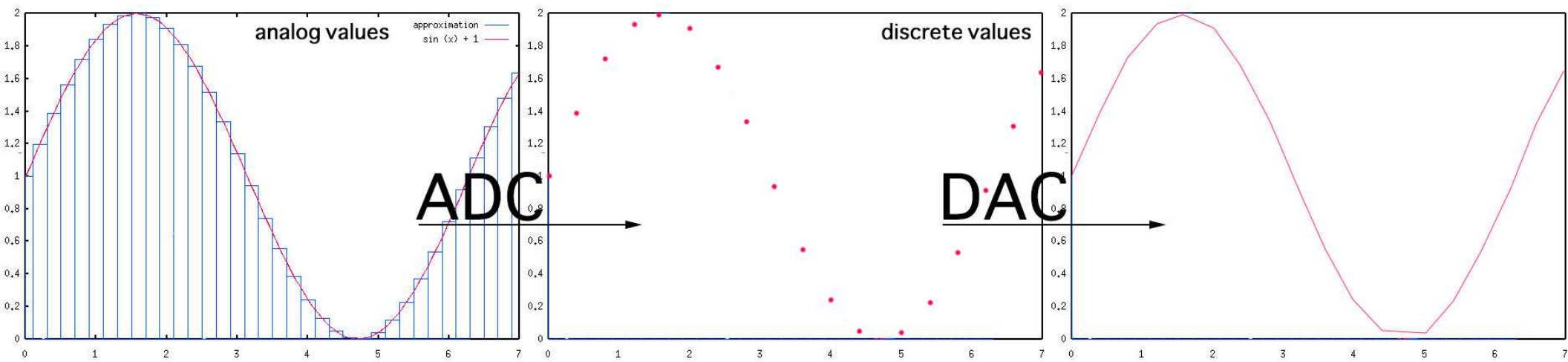
A set of input values consisting of all numbers in an interval. A set of input values consisting of only certain numbers in an interval. An approximation of the original analog values.

Hmm… ADC Resolution

$$\text{ADC Voltage Resolution} = \frac{(V_H - V_L)}{2^n}$$

where n is the number of bits of resolution of the ADC

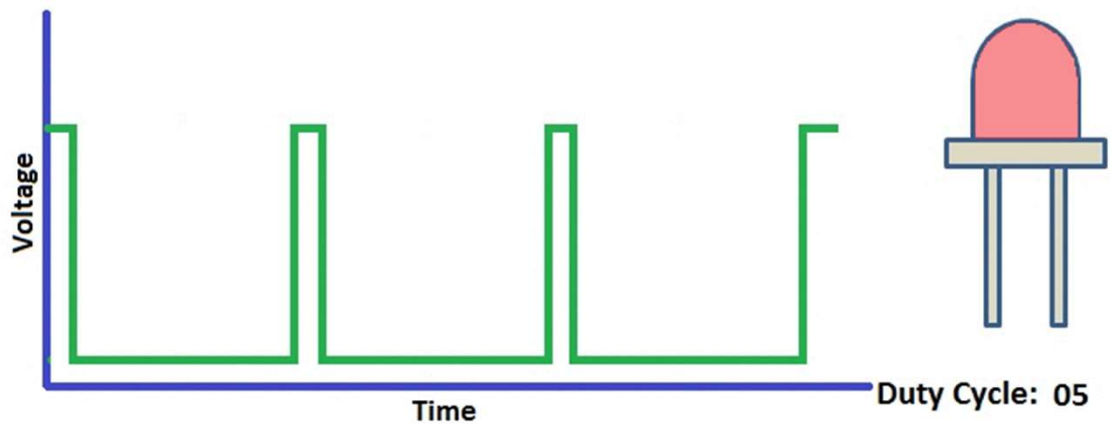
- If we convert a Voltage range of 0-5V :
- 2 bit ADC:
  - Will read a change when voltage varies by at least 1.25V
- 10 bit ADC (Arduino):
  - Will read a change when voltage varies by at least 0.0048V (1023 changes)
- It can be also defined as the value of voltage for one LSB (least significant bit)



A set of input values consisting of all numbers in an interval. A set of input values consisting of only certain numbers in an interval. An approximation of the original analog values.

Hmm… OK, what about PWM?

- Pulse width modulation
  - It is a square wave signal at a fixed or variable frequency.
  - By varying the on and off time (duty cycle) we can control peripherals such as the brightness of LED's or the speed and even direction of motors.
- In Arduino IDE the command to write to a PWM enabled pin is **analogWrite();**



Simple Analog input code – let’s write it.

```
const int LED_PIN = 3;
// We will attach the LED to the D3 PWM enabled pin.
// Cannot use internal LED because that is not on a PWM enabled pin (13).
const int POT_PIN = A0;
// We will attach the POT wiper to the A0 analog input (10bit ADC)

void setup() {
// the setup routine runs once, on power up or when you press the reset button

  Serial.begin(9600);
// initialize serial communication at 9600 bps so we can see what's happening (debug)
   pinMode(LED_PIN, OUTPUT);
// declare the above defined LED pin to be act as an output
}

// the loop routine runs over and over, forever
void loop() {

}
```

Let’s go to the next page!

Simple Analog input code – let’s write it.

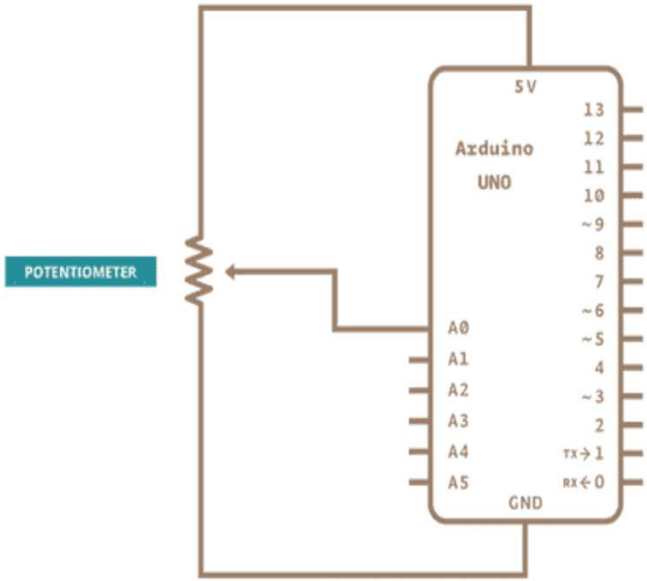
```
// the loop routine runs over and over, forever; stay within brackets { }
void loop() {

int PotReadValue = analogRead(POT_PIN);
// reads the input on analog pin A0 (value 0 and 1024 because is 10bit (2^10) ADC)

int brightness = map(PotReadValue, 0, 1024, 0, 255);
// scales the analog value to match the scale of PWM (value in the range of 0 to 255)

float PotVoltage = PotReadValue * (5.0 / 1024);
// Let's scale it so we also see the pot ADC counts as a voltage 0-5V

analogWrite(LED_PIN, brightness);
// sets the brightness of the LED connected to the above defined PWM enabled pin D3

// See what's happening
Serial.print("Potentiometer feedback value:  ");
Serial.print(PotReadValue);
Serial.print(" Counts");
Serial.print(" Voltage on the wiper:  ");
Serial.print(PotVoltage);
Serial.print(" V");
Serial.print(", Brightness: ");
Serial.println(brightness);
delay(100);
}
```

The Potentiometer and ADC part…

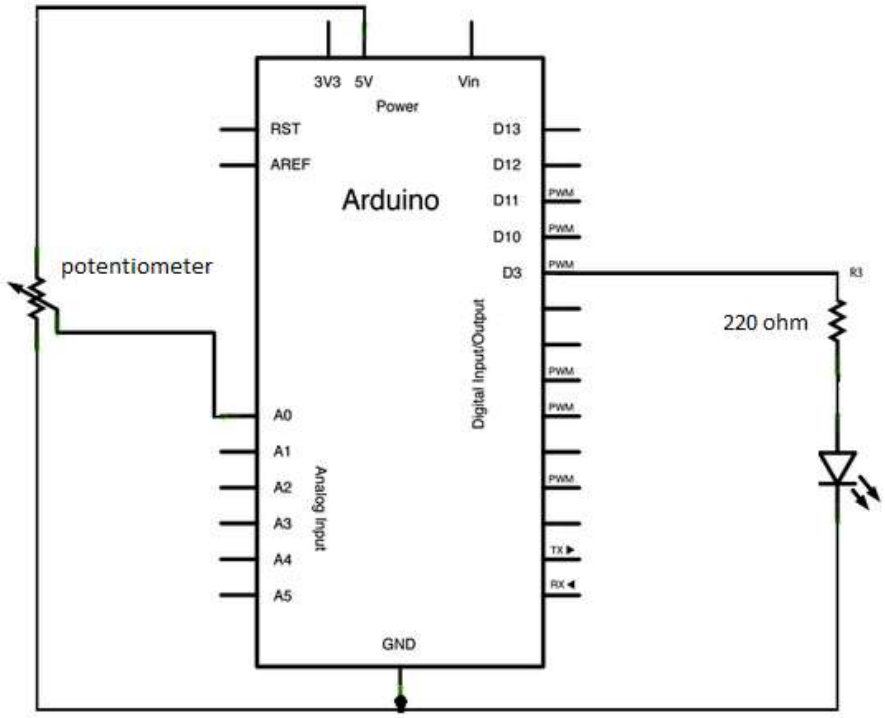
- Connect a 10k Potentiometer
  - Wiper to A0
  - One end to 5v and the other end to GND

The potentiometer will act as a divider with variable ratio. The A0 pin will be connected to a voltage ranging from 0 to 5V as you rotate the potentiometer shaft. This will cause the ADC to read from 0 to full counts which is 1024 for a 10-bit ADC as in the Arduino Nano.

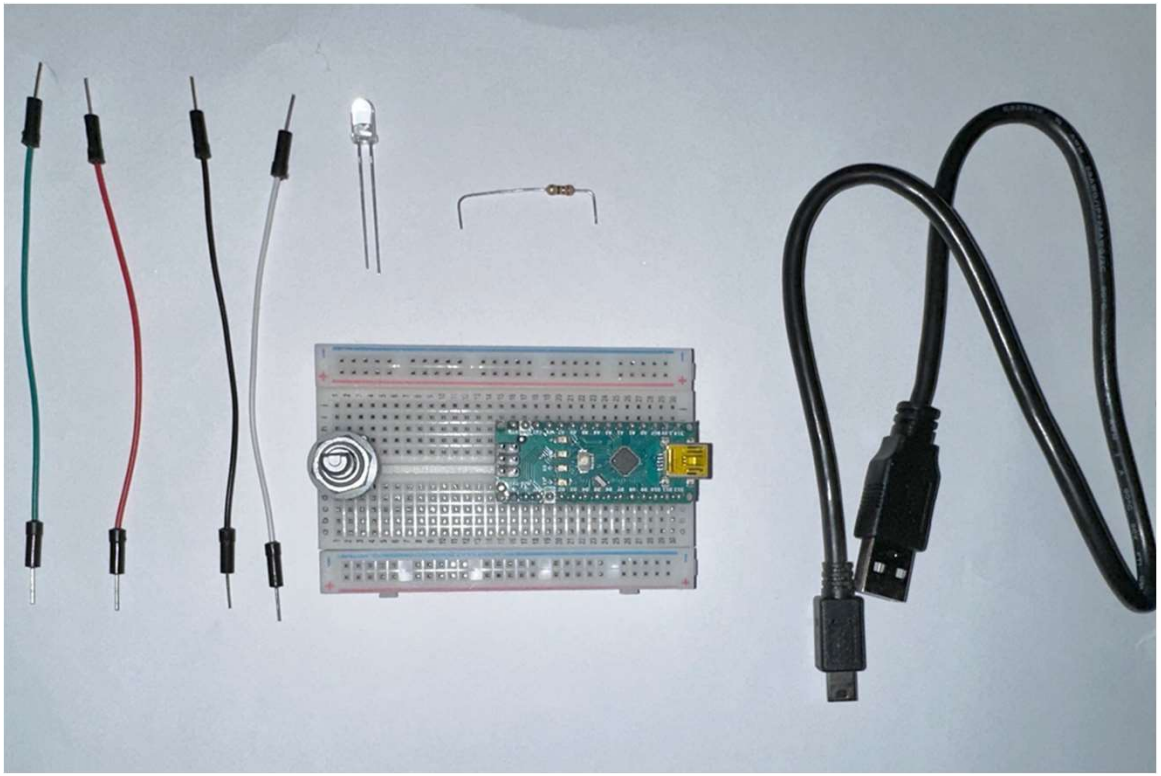
$2^{10} = 1024$

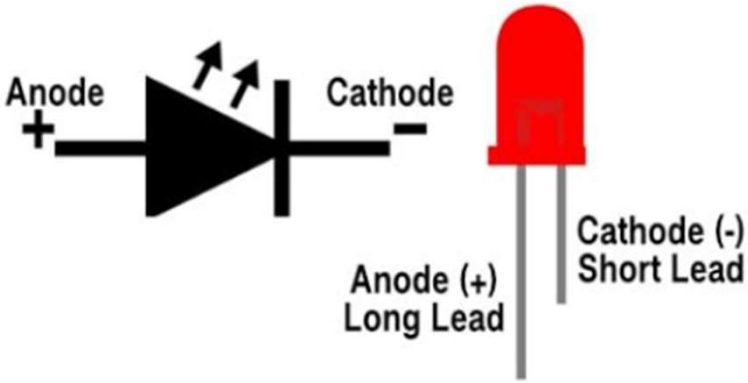


Use the potentiometer to dim an LED



Let’s build it!



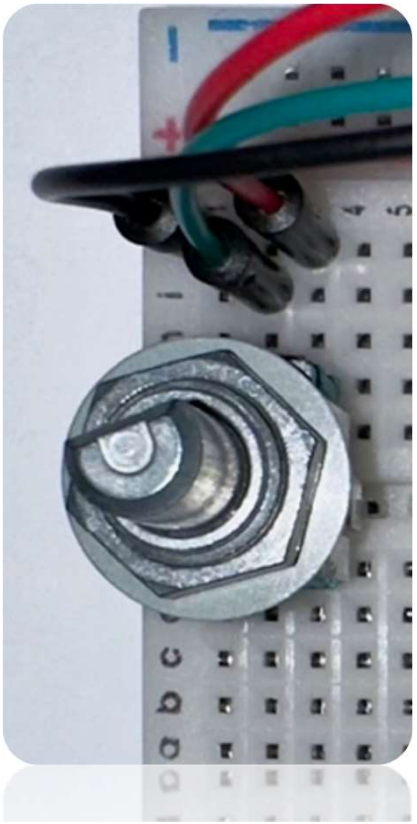


Your BOM:

- Breadboard with fitted Arduino Nano
- USB Mini cable
- 4 jumper wires
    - Black
    - Red
    - Green
    - White
- Teal LED
- 220ohm resistor with preformed leads

Let’s build it!

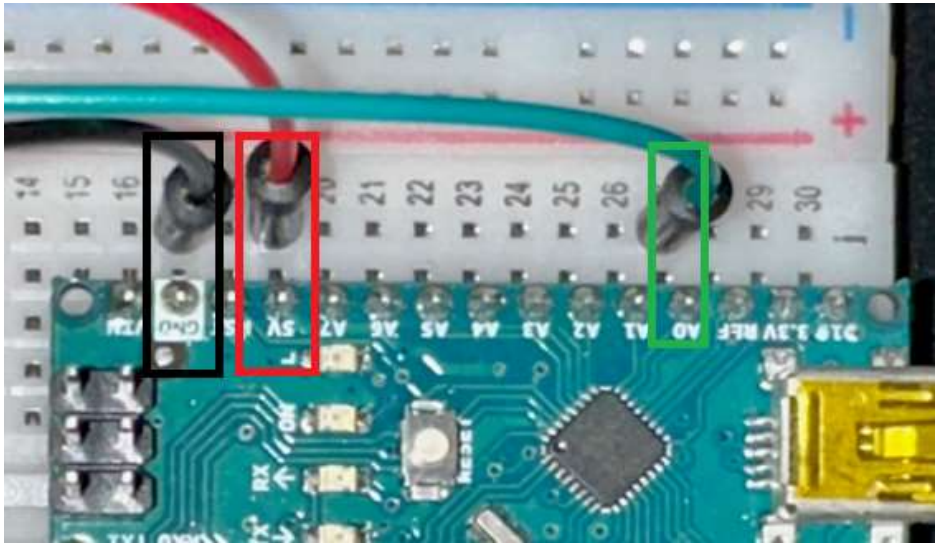
Connect the black wire to one end of the POT (either side)
Connect the red wire to the opposite end of the POT
Connect the green wire to middle pin (wiper) of the POT



Let’s build it!

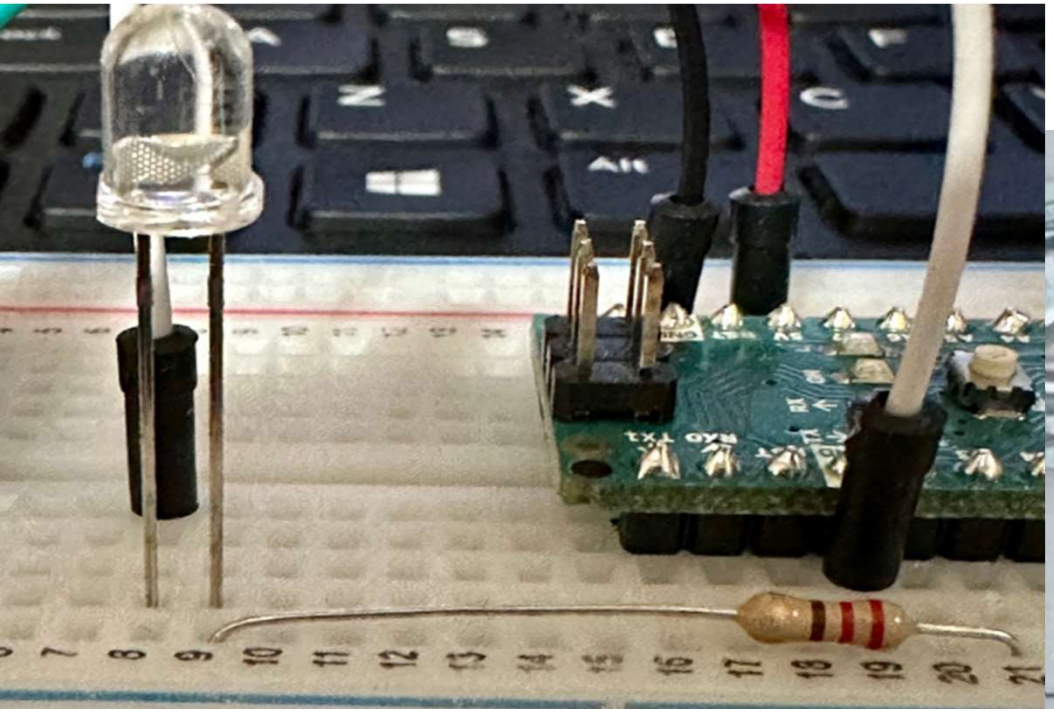
Connect the opposite end of the same jumper wires as below:

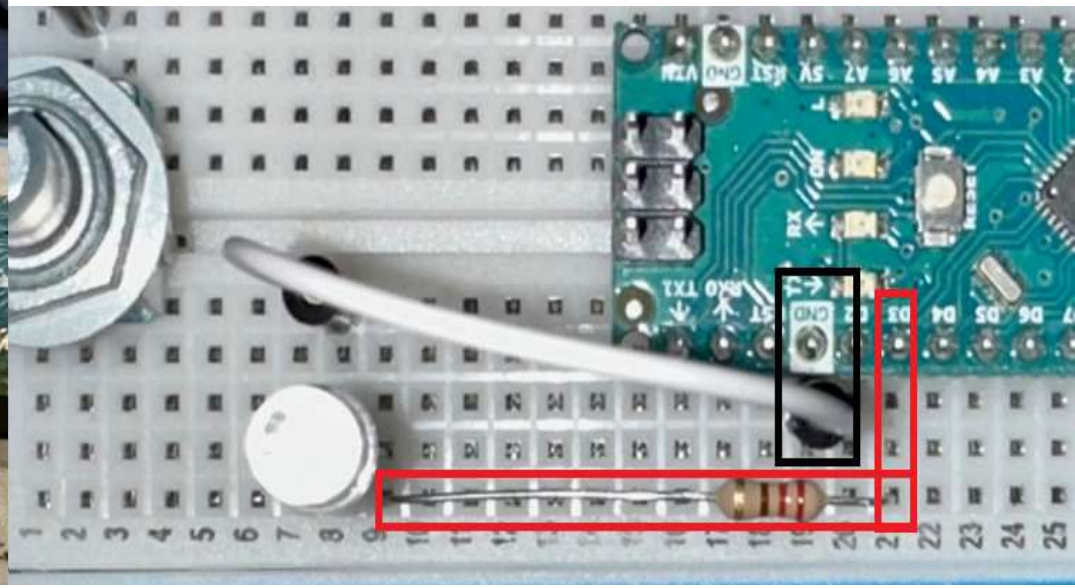
- Red wire to 5V pin line on ARDUINO
- Green wire to A0 pin line on ARDUINO
- Black wire to GND pin line on ARDUINO



Let’s build it!

- Connect the resistor to the D3 pin and either available strip of contacts
- Connect the LED’s positive (longer lead) to the other end of the resistor
- Connect the white wire to the LED’s shortest lead
- Connect the other end of the white wire to ARDUINO GND line





Read values > 5V. But how?

- ADC inputs levels are limited
  - Any ADC can read up to the reference voltage or to that of power supply rail, 5V in our case
  - While is easy to math out higher values, we still need to scale the inputs to less than 5V.
- Voltage dividers:
  - Using two resistors we can scale the input voltage to match our requirement

V_{out} needs to be max 5V

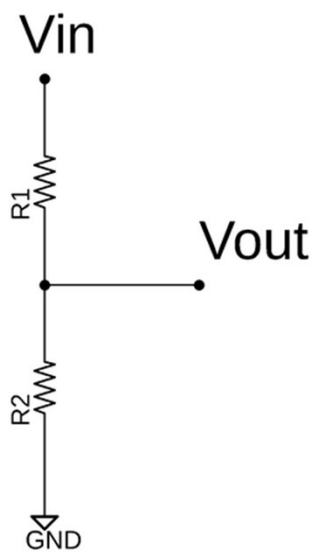
V_{in} is the voltage to be measured

R_2 (bottom resistor)

R_1 (top resistor)

Vout = ADCValue * 5 / 1023;

Vmeasured = Vout / (R2 / (R1 +R2));



Voltage/Potential Divider

$$V_{out} = \frac{R2}{R1 + R2} V_{in}$$

The code…

```
const int VinPin = A0;  // define ADC input as previously seen
float Vout = 0.0;
float Vmeasured = 0.0;
const float R1 = 10000.0;
const float R2 = 1000.0;  // (Ratio of 1/11 will give us a range of 0-55V for 0-5V input);

void setup() {
  Serial.begin(9600); // get eyes on what is happening with our values
}

void loop() {

int ADCValue = analogRead(VinPin);
int Vout = ADCValue * 5 / 1024; // 1024 is the ADC resolution (10 bit)
Vmeasured = Vout / (R2 / (R1 + R2)); // scales 0-5V measured to match with the ratio of the divider
Serial.print("Vout:"); // let's see what we have on the console
Serial.print(Vout);
Serial.print("V, Vmeasured:");
Serial.print(Vmeasured);
Serial.print("V, Divider ratio: ");
Serial.println(Vout / Vmeasured);
}
```

Protecting the Arduino

- Do not exceed voltage on the pins
  - Common mistake is when people use series resistors relying on the ADC pin internal pulldown to do the job for the divider
  - Always use a divider and allow between 1 and 5mA to flow through the divider.
  - I = Vin / R1 + R2
- Do not exceed load on the pins
  - The chips usually do up to 40mA load per IO pin, never get to the limits, stay at max. 25-30mA. Should you need more, use a buffer or simply a bipolar transistor.
- Don’t play with the AREF pin unless you stay within limits and always use a stable, precision voltage supply. (search for *analogreference* on arduino.cc)
- Always use the lowest voltage possible for your ARDUINO circuit.
  - Don’t try to use a 24V supply when you have 12V available. The Linear regulator on the ARDUINO board will convert the “difference” into heat.

What can go wrong?

- Values flutter, e.g. 60,61,60,60,60,61 due to noise on ADC or reference
  - Let's assume the counts value is between 60 and 61
  - There is noise on the input signal or reference
  - The USB supply voltage isn't steady, causes the reference to vary
  - Average multiple readings e.g. 60.4 (use rolling average, software filter)
  - At hardware level this achieves the same as an RC filter.
  - Disregard massive spikes e.g. 60,61,~~75~~,60,61,61,~~30~~,60
- We want to input/read voltages > 5v
  - Use a separate/external psu and use voltage dividers on the inputs
  - Use the inbuilt voltage references
  - Use an external voltage reference

Useful References

- Arduino Tutorial
  - https://www.tutorialspoint.com/arduino/arduino_functions_examples.htm
- The 1023 vs 1024 debate and how ADC works.
  - https://www.best-microcontroller-projects.com/arduino-adc.html