Arduino programming

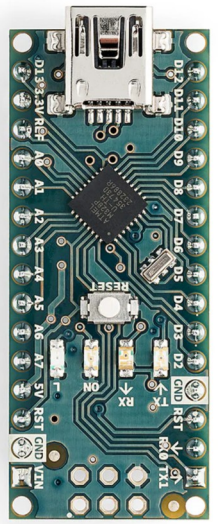
Some advanced techniques

By

Roger Walker

Why use an Arduino

(Instead of a Pi nano/Pi Zero/Full Pi/ESP32/Whatever)

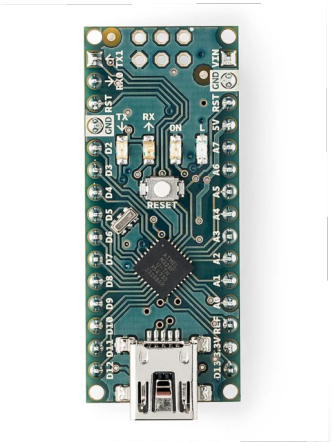


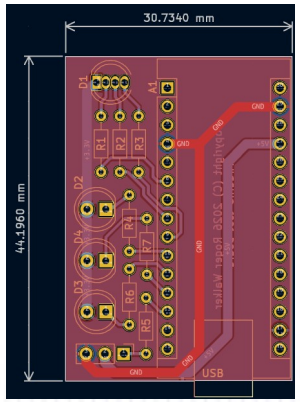
- Cheap – Nano clones are £2...3 from Aliexpress, ~£5 in UK
- Low power – Can be run from a few AA batteries / USB
- Fast boot time - can be under 1 second
- Modules are easy to use with prototyping boards – Easy to wire
- Lots of example wiring and code on-line – Lots of starting points
- Widely used - 50M+ downloads of the IDE, so lots of forums
- Open source hardware and software, that is well supported – Not locked in
- Lots of libraries available to interface to a wide range of devices – Easy to use
- Robust – Pretty tolerant of voltage errors, overloads, and other basic mistakes

Lets start!

What should you have

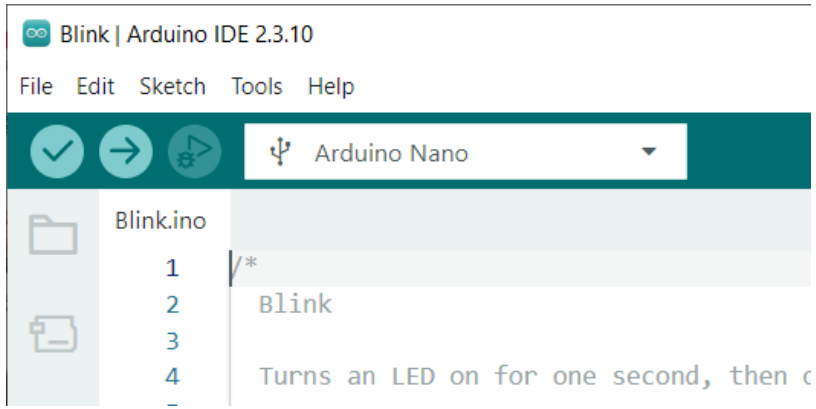
- Windows laptop with the IDE pre-installed
- An Arduino Nano – The brain
- A simple test board connecting some of the output pins to simple devices
  - 3 of the pins to a Red+Green+Blue LED so we can mix colours
  - 3 of the pins to separate LEDs so we can animate them
  - 1 of the pins to a servo header, so we can connect a radio control servo
- A USB cable to connect them



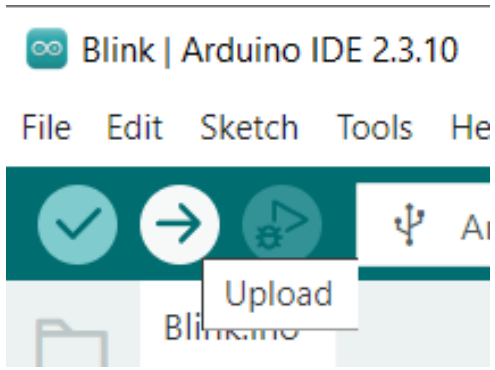


Starting things

- We start by launching the IDE, double click on this icon
- The IDE should start up, looking like this

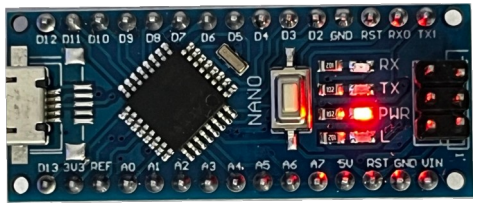


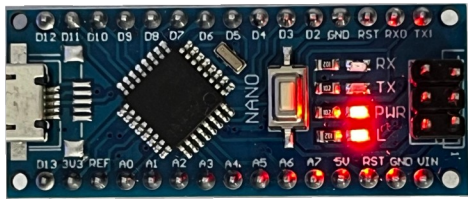
- Click on the “Upload” button
- You should see the following messages:
  - Compiling sketch
  - Uploading Sketch
  - Upload done



First result

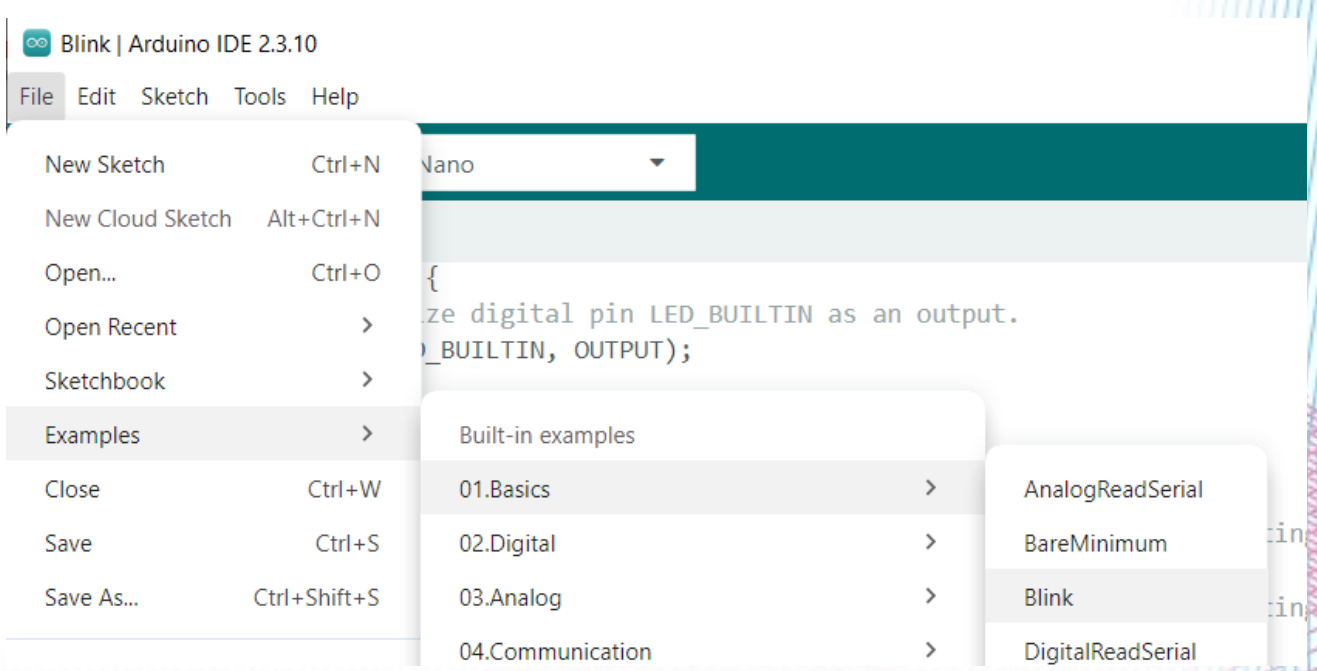
- The on board LED marked “L” should be flashing on-and-off with a 1000 milli-second (=1 second), delay between the on and off events.





- This program is is in the built in examples

- It’s a simple test to make sure everything is connected before we try anything more complex.



Some simple programs

Common LED board setup

To save time we will load this from a file

Use: File / Open / LED-Board-3 / LED-board-3.ino

```
// These are the pin definitions
#define LED_ON  HIGH
#define LED_OFF LOW

#define LED_1 7
#define LED_2 8
#define LED_3 9

#define LED_R 6
#define LED_G 5
#define LED_B 3

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize LED pins as outputs
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(LED_1, OUTPUT);
  pinMode(LED_2, OUTPUT);
  pinMode(LED_3, OUTPUT);
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);
}
```

Simple LED loop

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_1, LED_ON);
  delay(500);
  digitalWrite(LED_1, LED_OFF);

  digitalWrite(LED_2, LED_ON);
  delay(500);
  digitalWrite(LED_2, LED_OFF);

  digitalWrite(LED_3, LED_ON);
  delay(500);
  digitalWrite(LED_3, LED_OFF);

  digitalWrite(LED_2, LED_ON);
  delay(500);
  digitalWrite(LED_2, LED_OFF)
}
```

- Use the upload button, to program the board + run the program
- The LED sequence is 1,2,3,2, repeating as 1,2,3,2,1,2,3,2
- This zig-zag sequence is often known as the “knight rider” sequence (internet search “Kitt lights”)

Flickering LEDs

To save time we will load this from a file
Use: File / Open / LED-Board-4 / LED-board-4.ino

```
// the loop function runs over and over again forever
void loop() {
  analogWrite(LED_R, random(0, 255));
  analogWrite(LED_G, random(0, 255));   // Copy the LED_R line for LED_G
  analogWrite(LED_B, random(0, 255));   // Copy the LED_R line for LED_B
  delay(50);                            // Change the 500 to 50
}
```

- Use the upload button, to program the board + run the program
- If we change the brightness levels quickly then they appear to flicker
- This changes the 3 LEDs to different random levels every 50 milli-seconds (0.05 seconds), so 20 times per second.

How can we run both of these at the same time?

State based functions - 1

To save time we will load the basics from a file
Use: File / Open / LED-Board-5 / LED-board-5.ino

```
// Pick a reasonable time
#define POLL_TIME 10     // milli-seconds (so 100 times per second)

// Flickering LEDs
static unsigned guFlickerCount=0;
void led_flicker() {
   if (guFlickerCount==0) {
       analogWrite(LED_R, random(0, 255));
       analogWrite(LED_G, random(0, 255));
       analogWrite(LED_B, random(0, 255));
   }
   guFlickerCount++;
   if (guFlickerCount>=(50/POLL_TIME)) guFlickerCount=0;
}
```

We assume we are called at POLL_TIME intervals (10ms in this case), and we want to change at 50ms (like before). We change the LEDs at 0 (so the first time we're called we set them), then count until we reach 50ms, then reset to the count to 0, this triggers a change next time we are called.

State based functions - 2

```
// Zig Zag LEDs aka Knight Rider
static unsigned guZigZag=0;
void led_knight_rider() {
   if (guZigZag==0) {                    // Step 1
       digitalWrite(LED_2, LED_OFF);
       digitalWrite(LED_1, LED_ON);
   } else
   if (guZigZag==(500/POLL_TIME)) {      // Step 2 after 500 ms
       digitalWrite(LED_1, LED_OFF);
       digitalWrite(LED_2, LED_ON);
   } else
   if (guZigZag==(1000/POLL_TIME)) {     // Step 3 after 1000 ms
       digitalWrite(LED_2, LED_OFF);
       digitalWrite(LED_3, LED_ON);
   } else
   if (guZigZag==(1500/POLL_TIME)) {     // Step 4 after 1500 ms
       digitalWrite(LED_3, LED_OFF);
       digitalWrite(LED_2, LED_ON);
   }
   guZigZag++;
   if (guZigZag==(2000/POLL_TIME)) {     // reset to step 0 after 2000ms
       guZigZag=0;
   }
}
```

This will be called at POLL_TIME intervals, and we want to change state at these points.

Simple calling loop

```
// the loop function runs over and over again forever
void loop() {
   led_flicker();
   led_knight_rider();
   delay(POLL_TIME);
}
```

- Use the upload button, to program the board + run the program
- The led_flicker() will set the RGB values
- The led_knight_rider() will sequence the LEDs
- delay(POLL_TIME) will set the timing
- The timing is not 100% accurate, because the program has to run the loop every POLL_TIME milli-seconds, and call the functions. But all of that might take a 100 micro seconds, so if you keep the POLL_TIME big enough, the error is small (100us vs 10ms is about 1%).

State based functions - 3

```
// Flash the LEDs at different rates
static unsigned guLedFlash1=0;
void led_flash1() {
   if (guLedFlash1==0)                          digitalWrite(LED_1, LED_ON);
   else if (guLedFlash1==(100/POLL_TIME))       digitalWrite(LED_1, LED_OFF);
   guLedFlash1++;
   if (guLedFlash1==(200/POLL_TIME))    guLedFlash1=0;
}
static unsigned guLedFlash2=0;
void led_flash2() {
   if (guLedFlash2==0)                          digitalWrite(LED_2, LED_ON);
   else if (guLedFlash2==(50/POLL_TIME))        digitalWrite(LED_2, LED_OFF);
   guLedFlash2++;
   if (guLedFlash2==(100/POLL_TIME))    guLedFlash2=0;
}
static unsigned guLedFlash3=0;
void led_flash3() {
   if (guLedFlash3==0)                          digitalWrite(LED_3, LED_ON);
   else if (guLedFlash3==(90/POLL_TIME))        digitalWrite(LED_3, LED_OFF);
   guLedFlash3++;
   if (guLedFlash3==(180/POLL_TIME))    guLedFlash3=0;
}
```

These will be called at POLL_TIME intervals, and we want to change state at these times.

Modified calling loop

```
// the loop function runs over and over again forever
void loop() {
   led_flicker();
   //led_knight_rider(); // Make this line a comment (add // to the start)
   led_flash1();         // Add this line
   led_flash2();         // Add this line
   led_flash3();         // Add this line
   delay(POLL_TIME);
}
```

- Use the upload button, to program the board + run the program
- The led_flicker() will set the RGB values
- The led_flash1() will flash LED1 (and 2 for 2, 3 for 3)
- delay(POLL_TIME) will set the timing

- So 4 separate LED operations appear to happen at once.

State based functions - 4

```
static unsigned guLedFlash2=0;
void led_flash2() {
   if (guLedFlash2==0)                        digitalWrite(LED_2, LED_ON);
   else if (guLedFlash2==(50/POLL_TIME))    digitalWrite(LED_2, LED_OFF);
   guLedFlash2++;
   if (guLedFlash2==(150/POLL_TIME))        guLedFlash2=0; // Change to 150
}


static unsigned guLedFlash3=0;
void led_flash3() {
   if (guLedFlash3==0)                         digitalWrite(LED_3, LED_ON);
   else if (guLedFlash3==(90/POLL_TIME))     digitalWrite(LED_3, LED_OFF);
   guLedFlash3++;
   if (guLedFlash3==(120/POLL_TIME))         guLedFlash3=0; // Change to 120
}
```

- Use the upload button, to program the board + run the program
- This shows that the LED flash rate doesn't have to be equal
- LED2 is on for 50ms, off for 100ms
- LED3 is on for 90ms, off for 30ms

State based functions - 5

To save time we will load the basics from a file
Use: File / Open / LED-Board-6 / LED-board-6.ino

```
// Red Green Blue fade up
static unsigned guRGBTime=0;
static unsigned guRGBChan=0;
static unsigned guRGBLevel=0;
void led_rgb_fade() {
  guRGBTime++;
  if (guRGBTime>=(20/POLL_TIME)) {
    guRGBLevel++;
    if (guRGBLevel==255) {
      // Reach max
      analogWrite(LED_R, 0);
      analogWrite(LED_G, 0);
      analogWrite(LED_B, 0);
      guRGBChan=(guRGBChan+1)%3;
      guRGBLevel=0;
    }
  analogWrite((guRGBChan==0) ? LED_R :
              (guRGBChan==1) ? LED_G : LED_B, guRGBLevel);
  }
}
```

The guRGBTime counts the time between changes (20ms x 255 = 5.1 secs)
guRGBChan is 0 for R(Red), 1 for G(Green), 2 for B(Blue)
guRGBLevel is 0...255 for all off to all on

Questions & Answers