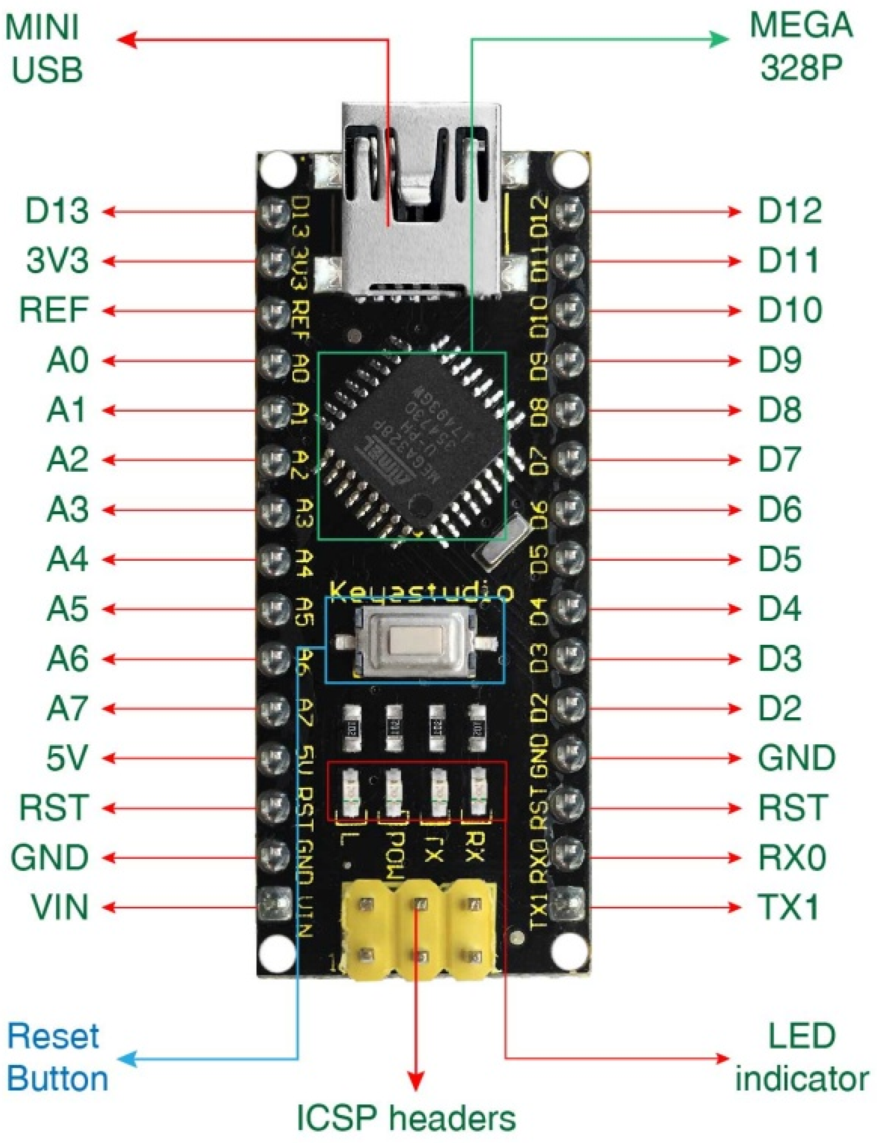
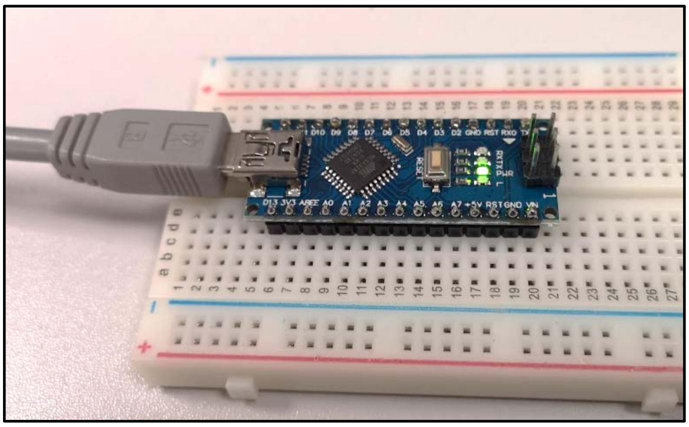
1 Arduino NANO Pin Diagram



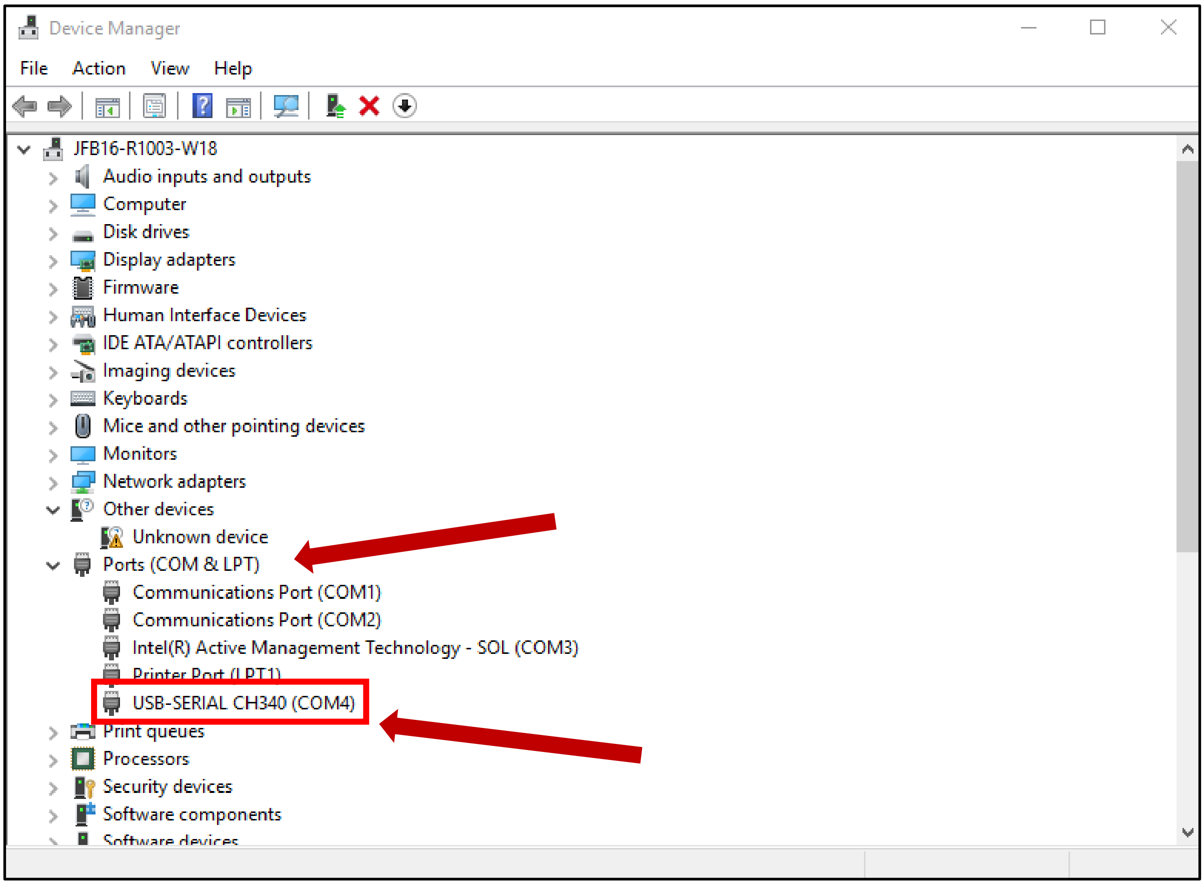
2 Connecting NANO board to PC



The boards provided for the embedded system challenge of the MicroHack are Arduino-compatible “Nano” boards.

They can be programmed from within the Adruino IDE over USB.

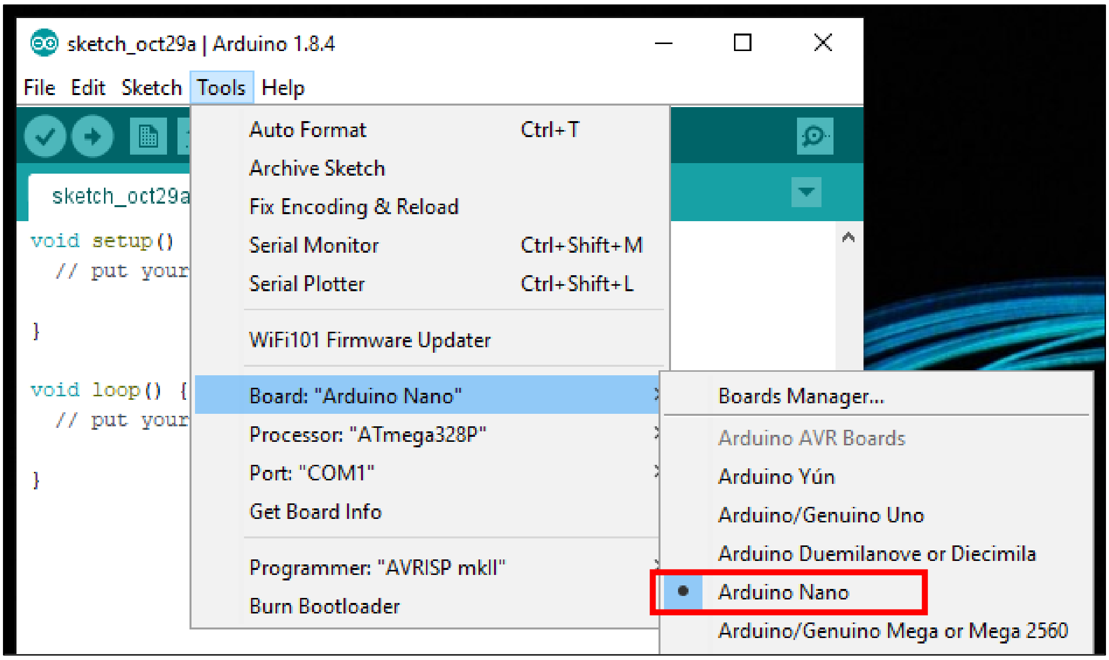
First you need to determine which COM port the Nano is connected to. To do this open the start menu and open “Device Manager”



```
JFB16-R1003-W18
  Audio inputs and outputs
  Computer
  Disk drives
  Display adapters
  Firmware
  Human Interface Devices
  IDE ATA/ATAPI controllers
  Imaging devices
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  Other devices
    Unknown device
  Ports (COM & LPT)
    Communications Port (COM1)
    Communications Port (COM2)
    Intel(R) Active Management Technology - SOL (COM3)
    Printer Port (LPT1)
    USB-SERIAL CH340 (COM4)
  Print queues
  Processors
  Security devices
  Software components
  Software devices
```

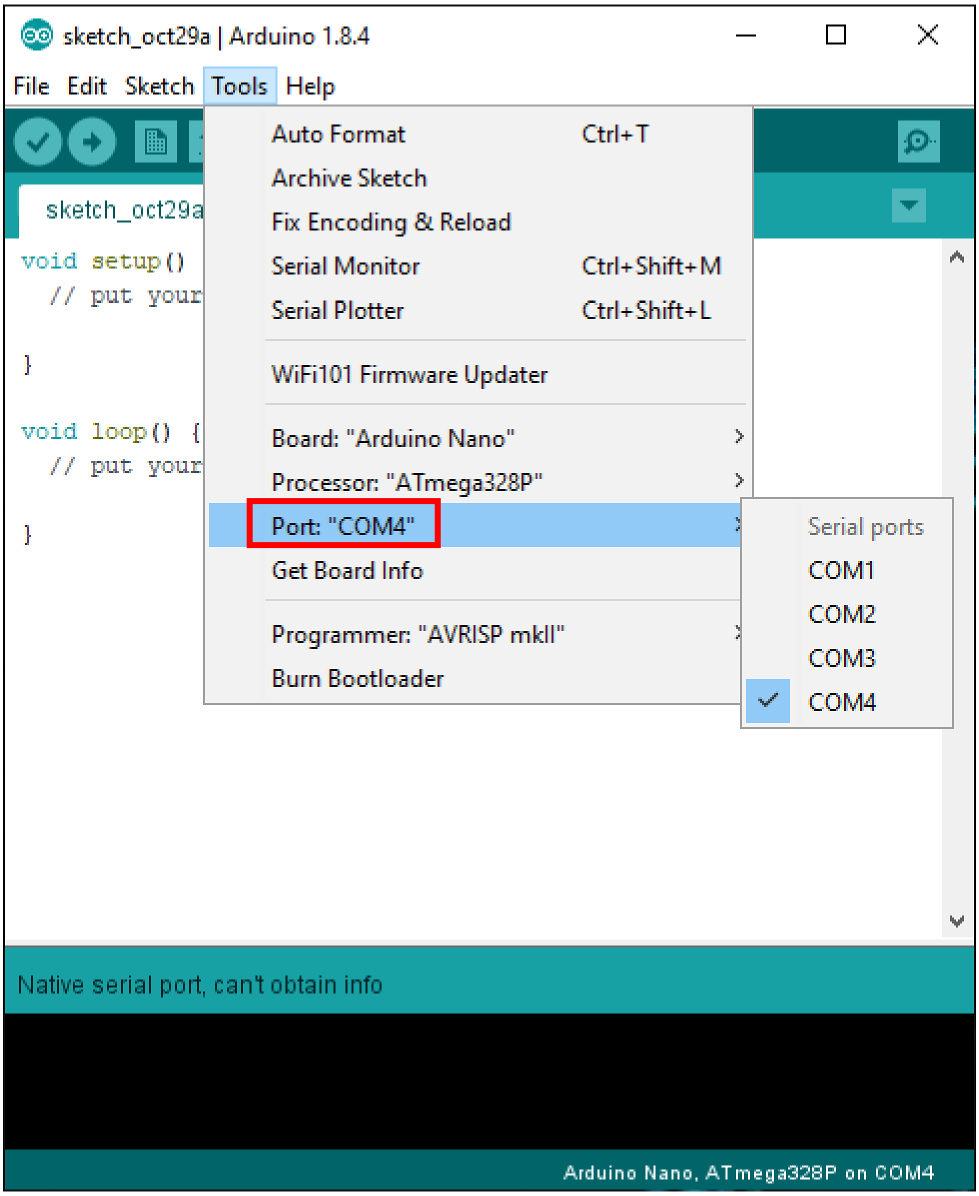
Under the "Ports (COM & LPT)" section you should see "**USB-SERIAL CH340**" and beside it the COM port it linked to. In this case it is "**COM4**" but this won't always be the case. If you do not see USB-SERIAL CH340 with the board connected, ask one of the ECSS committee to help before messing around with any drivers on lab machines.

Now we have the COM port we can launch the **Arduino IDE** which should be installed on all lab PCs.



Under "Tools" you will need to set the board to "Arduino Nano", the processor should already be selected as "ATmega328P".

Now change the Port to whichever COM port was found in the previous step, in this case COM4 is selected.



```
void setup()
  // put your

}

void loop() {
  // put your

}
```

Native serial port, can't obtain info

Arduino Nano, ATmega328P on COM4

The Arduino IDE should now be setup for programming the Nano boards.

3 Beginners Introduction to Arduino C Language

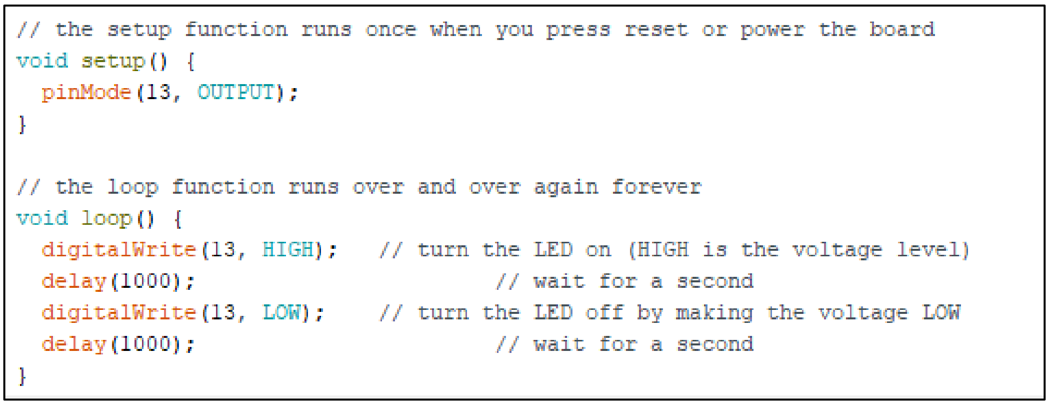
Arduinos are programmed in C, however if you have never used an Arduino before you will notice it is slightly different and there are custom functions for controlling the hardware. This will briefly explain how to use some basic functionality of an Arduino for somebody who has never used one before. Further information of how to program on an Arduino can be found at: https://www.arduino.cc/reference/en/

Structure

Normally in C we write all our code within the `main()` function, however on an Arduino we have `setup()` and `loop()` functions to write code in.

`setup()` is called once when the Arduino is powered on and then loop() runs indefinitely. This means we put setup code such as pin assignments in `setup()` and then we do the majority of the program within the infinite `loop()` function.

A good example is the LED blink program - or as they are called within the Arduino IDE: “sketches”.



```
// the setup function runs once when you press reset or power the board
void setup() {
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Basic Digital Pin Arduino Functions

`pinMode(PIN, MODE)` – Set a pin as an Input or Output

PIN – Pin number on Adruino board

MODE – Can be INPUT, OUTPUT or INPUT_PULLUP

`digitalWrite(PIN, VALUE)` – Set a output pin as high or low

PIN – Pin number on Adruino board

VALUE – Can be HIGH (1) or LOW (0)

`digitalRead(PIN)` – Read if a input pin in High or Low

PIN – Pin number on Adruino board

Arduino Serial Communications

The Arduino can communicate with the PC using Serial however it is important to note that serial uses pins D0 and D1 so these cannot be used for regular pin functionality.

In order to use serial communications, we initialise it with `Serial.begin(9600)` - this starts serial communications with a speed (baud rate) of 9600 baud.

We can then print over serial using `Serial.print(VALUE)` or `Serial.println(VALUE).`

Similarly, we can read data from serial using `Serial.read()` for a single byte, `Serial.readBytes(BUFFER, LENGTH)` for multiple bytes and then `Serial.readString()` for a string of text.

For additional functions and information, I recommend reading from:

https://www.arduino.cc/reference/en/language/functions/communication/serial/