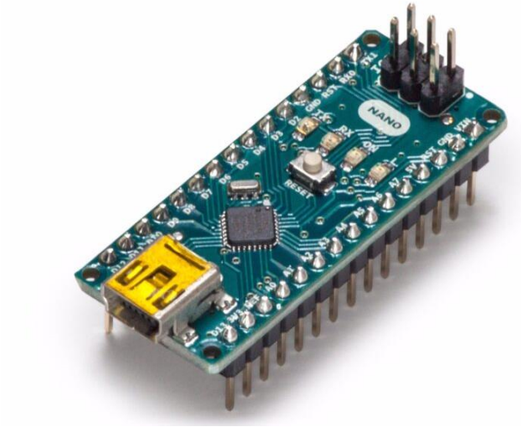
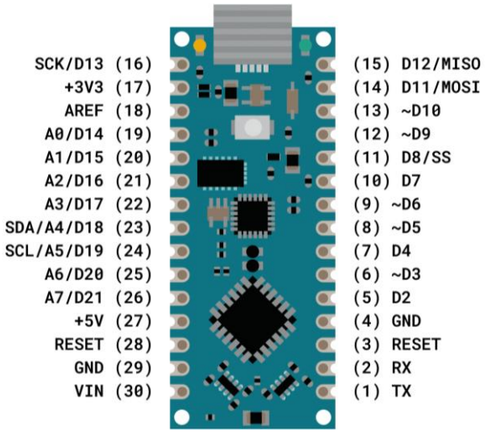
Introduction to microcontroller programming

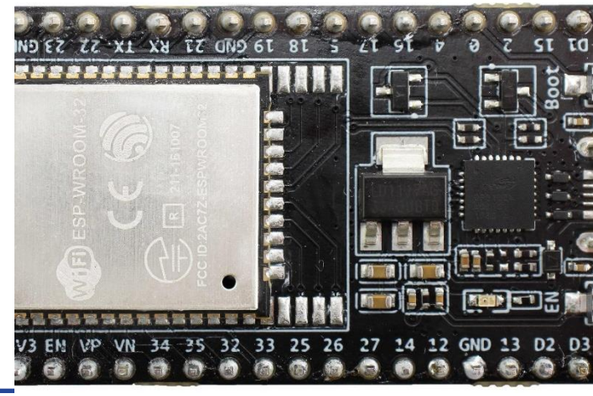
Mark Jones, G0MGX

What is an MCU?

- Generic term used for a small computer in a single package
- An MCU is a CPU plus memory and other peripherals
- Designed specifically for embedded applications – not a home computer
- Unlike a PC which needs other bits and bobs

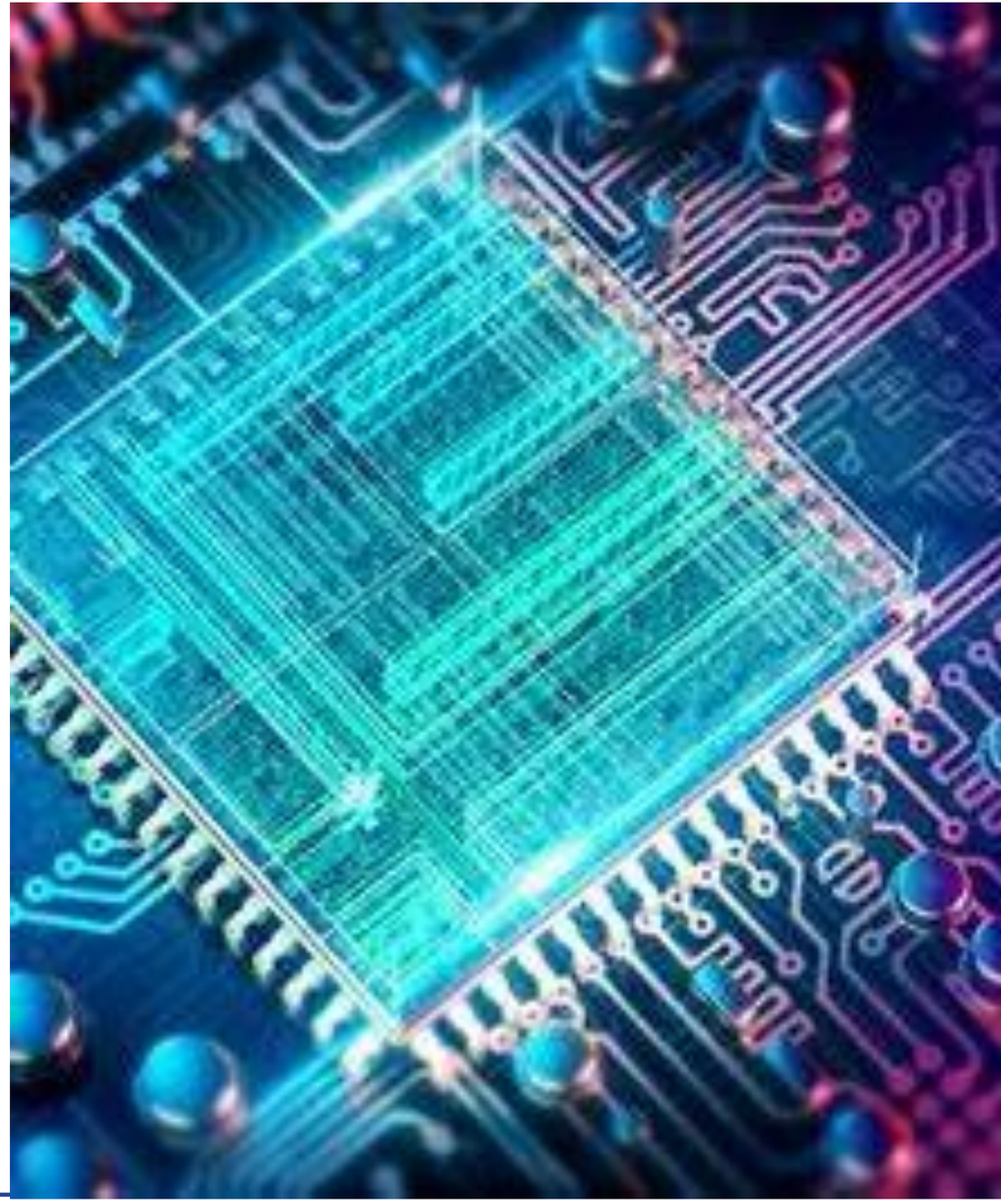






What is an MCU?

- Different MCUs contain varied architecture depending on target application
- Many manufacturers and types
- Many “clones” from the land where Copyright means copy it right.
- Aimed at embedded applications

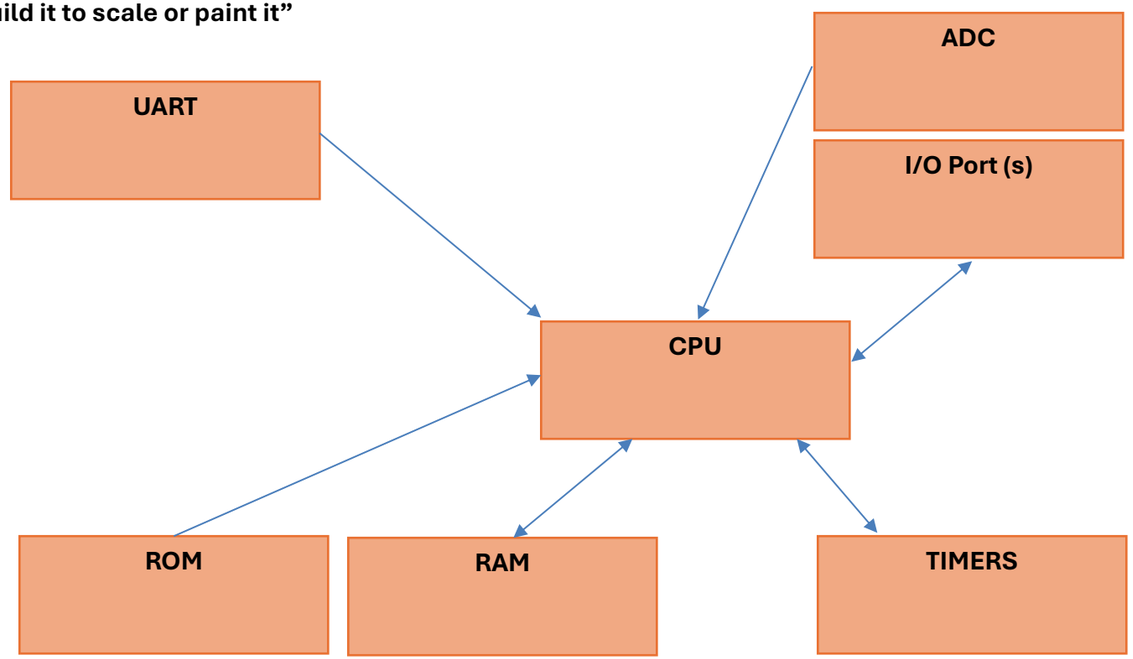


What is an MCU?

- Typically contain:
  - Memory
  - Configurable GPIO
  - ADC
  - DAC
  - PWM
  - Interfaces like UART, I2C, SPI
  - Timers/Counters

What is an MCU?

“Please excuse the crudity of this model. I didn't have time to build it to scale or paint it”



What are we going to play with today?

- Arduino Nano:
  - ATmega328P
  - On Board Clock 16MHz
  - 32Kb Flash (ROM) 2Kb (RAM)
  - USB SPI I2C UART interfaces
  - 14 digital I/O 6 PWM 8 Analogue
  - It’s only 18 x 45mm!

What are we going to play with today?

- ATMega 328:
  - 8-bit AVR® RISC-based microcontroller
  - AVR® CPU
  - 32kB Flash, 2kB RAM, EEPROM 1024B
  - 48 Pins
  - 3 counter/timers
  - UART, SPI, GPIO pins
  - ADC
  - Chip costs £1.47 (DigiKey 12 Sept 2024)

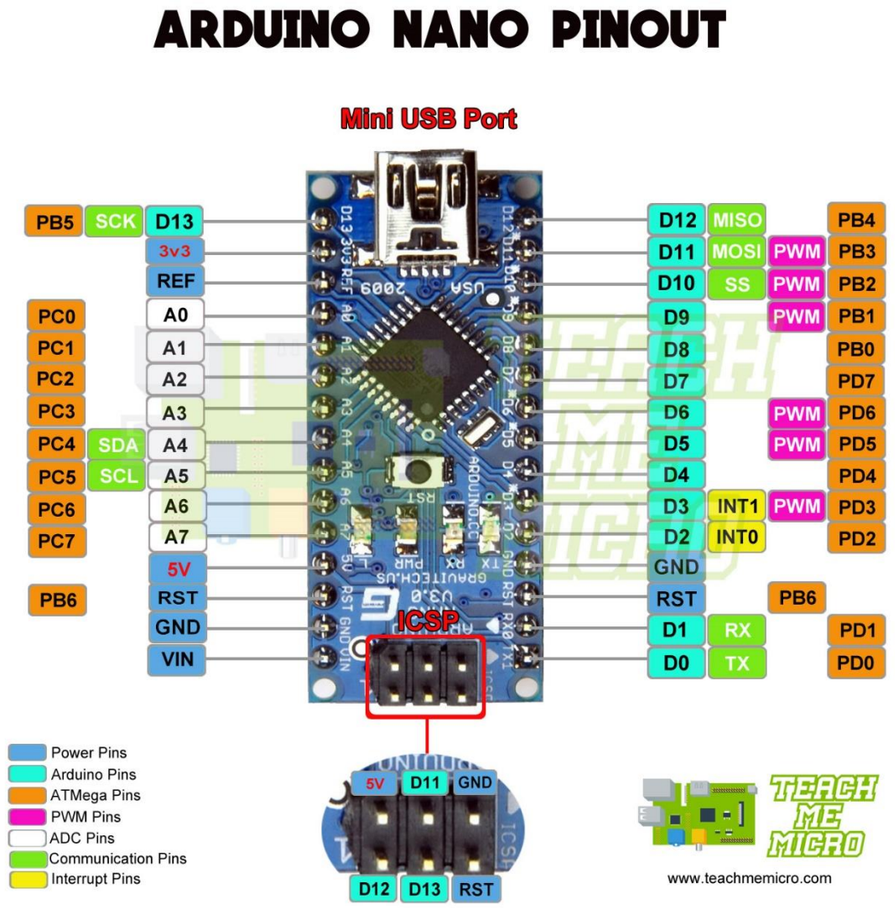
Just for nostalgia

- In 1984 I had a BBC B Microcomputer:
  - 6502A microprocessor (DIL package bigger than the board)
  - 2MHz clock
  - 32 KB RAM
  - Cassette tape for storage
  - Cost £399 (£1,259 today)

Hello World!

- In the world of Software Engineering the first program you would write would be “hello world”
- In the embedded software engineering world we flash an LED
- So let’s get started
- There are 2 main parts to any embedded piece of software:
    - Initialisation (runs once)
    - Loop (runs forever)

Our Board


ARDUINO NANO PINOUT
Mini USB Port
PB5 SCK D13
3v3
REF
PC0 A0
PC1 A1
PC2 A2
PC3 A3
PC4 SDA A4
PC5 SCL A5
PC6 A6
PC7 A7
5V
PB6 RST
GND
VIN
D12 MISO PB4
D11 MOSI PWM PB3
D10 SS PWM PB2
D9 PWM PB1
D8 PB0
D7 PD7
D6 PWM PD6
D5 PWM PD5
D4 PD4
D3 INT1 PWM PD3
D2 INT0 PD2
GND
RST PB6
D1 RX PD1
D0 TX PD0
ICSP
Power Pins
Arduino Pins
ATMega Pins
PWM Pins
ADC Pins
Communication Pins
Interrupt Pins
5V D11 GND
D12 D13 RST
TEACH ME MICRO
www.teachmemicro.com

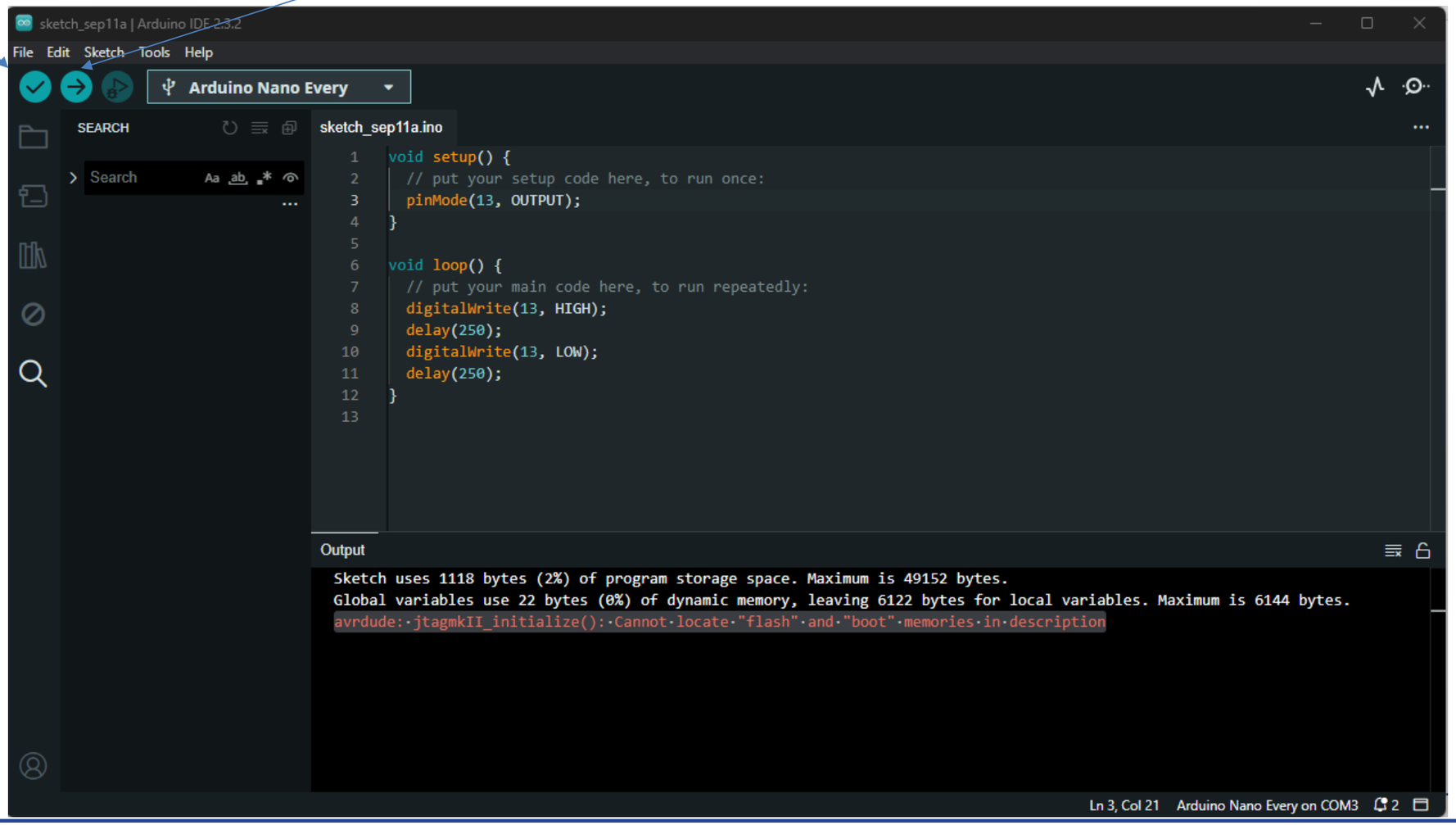
Hello World!

- Connect the board to the USB port on your PC
- Open the Arduino IDE (which you preinstalled)
- You may need to install board configuration files (instal on demand)
- Wi-Fi details: Kents Hill

Hello World!

This Compiles your code:

This Uploads your code:



Hello World! – Whats happening here?

```ino
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  delay(250);
  digitalWrite(13, LOW);
  delay(250);
}
```

A word about libraries

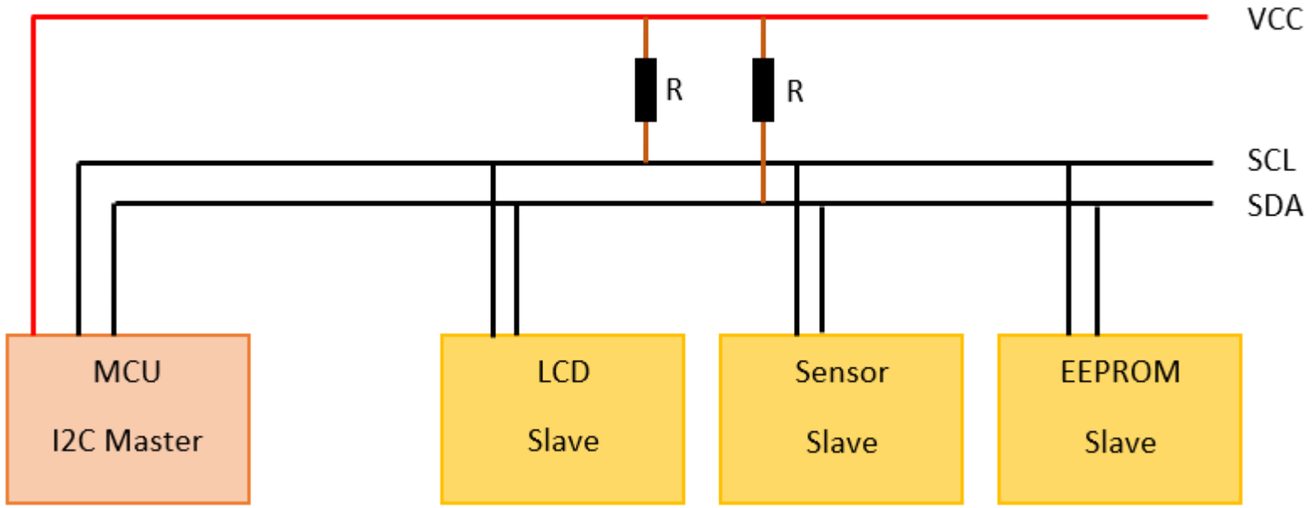
- We often need to interface to external bits and bobs
- You:
  - Start with the datasheet and figure out exactly how the interface works, then write the code to provide the interface
- Or:
  - Let someone else do it for you.

A word about libraries

- Arduino IDE contains a “Library Manager” where we can find a host of other peoples code
- We are going to use “LiquidCrystal I2C”
- Tools -> Manage Libraries
- Search “LiquidCrystal I2C”
- Install

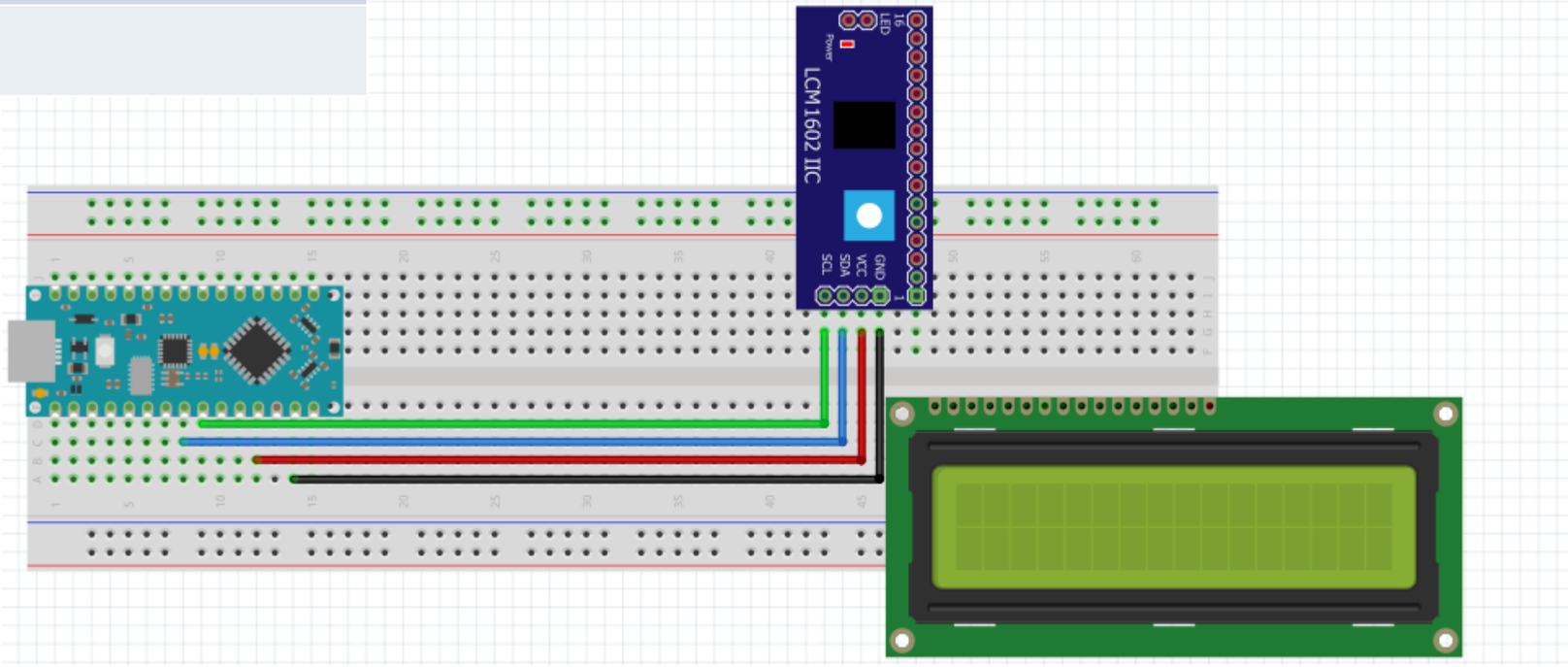
A word on I2C or IIC

- Our LCD module uses I2C (pronounced I squared C) sometimes called I2C (I two C) Inter-Integrated circuit serial Communications for the MCU to talk to the LCD
- Developed in 1982 by Phillips Semiconductors® it is a mechanism designed for short distance serial communications between low-speed peripheral ICs and processors or microcontrollers.
- The protocol uses a serial clock line (SCL) and a serial data line (SDA) and has the concept of a slave address, meaning that multiple devices with different addresses can reside on the same bus.
- In our example the address of the LCD interface is hex 27
- Most I2C compatible hardware devices will have a default address, but usually include a mechanism of changing the address in the event of a clash with another peripheral.
- In our example, there are spaces on the I2C LCD interface board for links labelled A0, A1, A2 which alter the device address.



Wiring The LCD

I2C LCD	Every
GND	GND
VCC	5V
SDA	A4
SCL	A5



Create a new sketch…..

LCDExample1.ino

```ino
#include "Wire.h"
#include "LiquidCrystal_I2C.h"

#define SCREEN_ADDRESS 0x27 // this is the address of the LCD I2C Interface

LiquidCrystal_I2C lcd(SCREEN_ADDRESS, 20, 4);

void setup() {
  // put your setup code here, to run once:
  lcd.init();           //initializs the lcd
  lcd.backlight();      //switch on the backlight
  lcd.setCursor(0, 0);  // column, row);
  lcd.print("Hello World");
  lcd.setCursor(0,3);
  lcd.print("Easy, instn't it?");
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Try something fancy…..

LCDExample2.ino

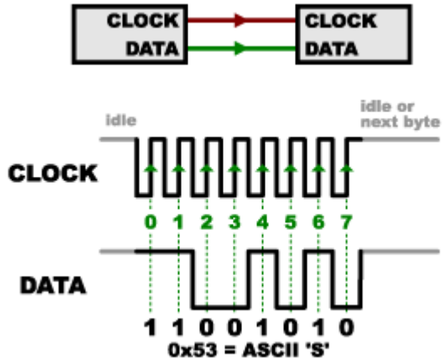
```
#include "Wire.h"
#include "LiquidCrystal_I2C.h"

#define SCREEN_ADDRESS 0x27 // this is the address of the LCD I2C Interface

LiquidCrystal_I2C lcd(SCREEN_ADDRESS, 20, 4);

void setup() {
  // put your setup code here, to run once:
  lcd.init();           //initializs the lcd
  lcd.backlight();      //switch on the backlight
  lcd.setCursor(0, 0);  // column, row);
  lcd.print("Hello World");
  lcd.setCursor(0,3);
  lcd.print("Easy, instn't it?");
  delay(5000);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("MCU with Arduino");

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

The AD9833

- Analog Devices make some amazing gizmos
- The AD9833 is a very simple signal generator
- Capable of Sine, Square and Triangular waveforms up to about 12.5MHz
- It uses the simple SPI interface

A word on SPI

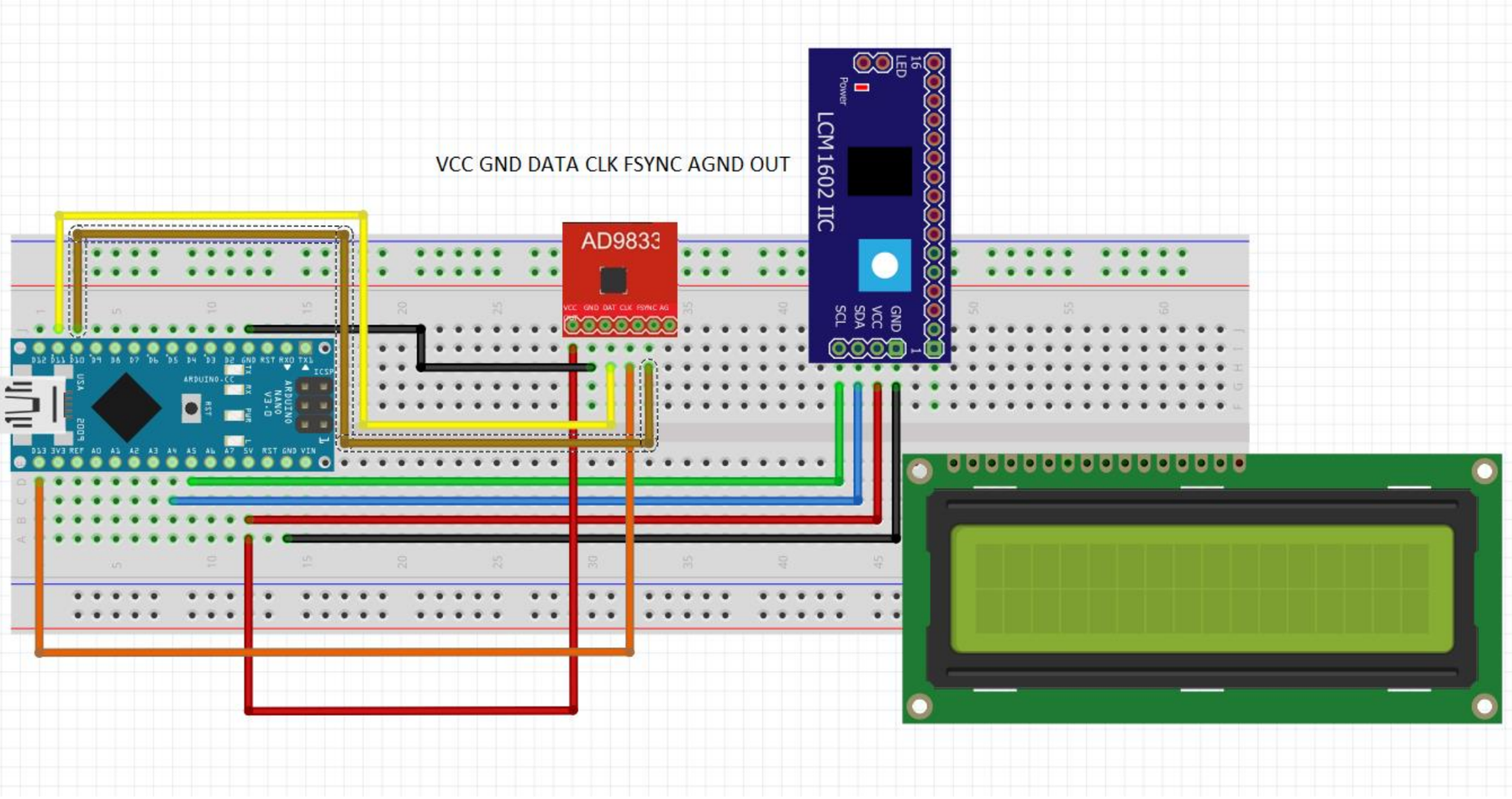
- SPI is a simple Serial Peripheral Interface – it is a synchronous serial communication protocol using a master and slave architecture.
- Its popularity is due to its simplicity – it can use a simple shift register as a receiver rather than a more complex Universal Asynchronous Receiver / Transmitter – hence synchronous
- The peripheral is selected using a CS line, hence 3 wires are needed for more than one device



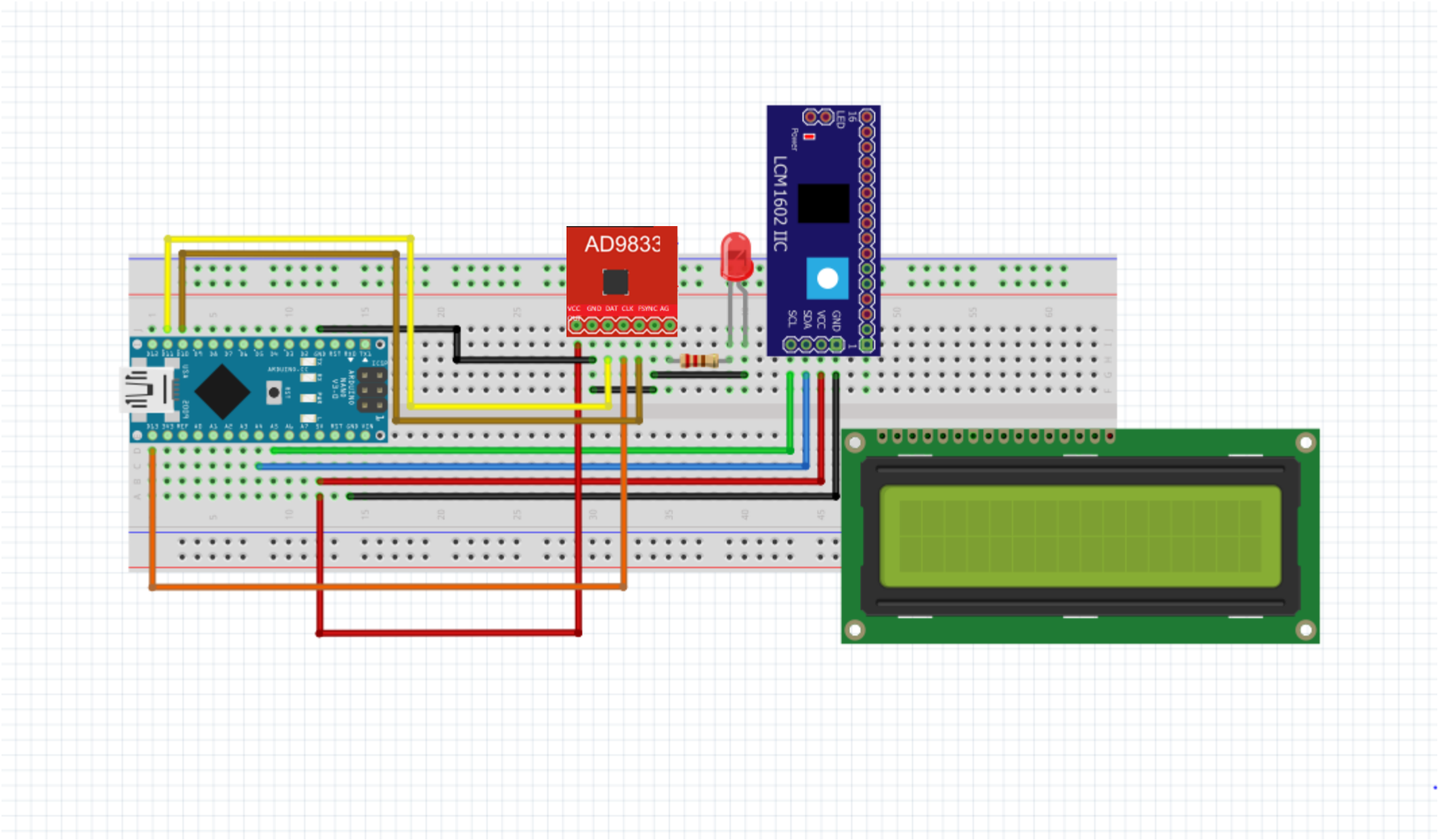
Wiring The AD9833

AD9833	Arduino
VCC	5V
DGND	GND
SDATA	D11
SCLK	D13
FSYNC	D10

“Please excuse the crudity of this model,
I didn’t have time to make it to scale or paint it”

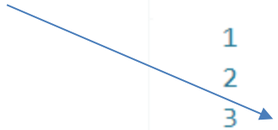


Wiring the AD9833



Expand on our sketch…..

New Library!
AD9833



AD9833.ino

```
1   #include "Wire.h"
2   #include "LiquidCrystal_I2C.h"
3   #include "AD9833.h"
4
5
6   #define SCREEN_ADDRESS 0x27 // this is the address of the LCD I2C Interface
7
8   LiquidCrystal_I2C lcd(SCREEN_ADDRESS, 20, 4);
9
10  //Pins for our AD9833
11  //const int DATA      = 11;
12  //const int CLK       = 13;
13  const int FSYNC     = 10;
14
15  AD9833 SigGen(FSYNC);
16
```

Expand on our sketch…..

```
void setup() {
  // put your setup code here, to run once:
  lcd.init();           //initialise the lcd
  lcd.backlight();      //switch on the backlight

  SPI.begin();          //initialise SPI interface
  SigGen.begin();       //initialise AD9833
  SigGen.reset();       //reset for safety

  lcd.setCursor(0, 0);  // column, row
  lcd.print("AD9833");
  lcd.setCursor(0,1);
  lcd.print("Signal Gen");
}
```

Expand on our sketch…..

```
void loop() {
  // put your main code here, to run repeatedly:
  SigGen.setFrequency(1000);   //  1000 Hz.
  SigGen.setWave(AD9833_SQUARE1);
  lcd.setCursor(0, 3);
  lcd.print("1000Hz Square      ");
  delay(10000);
  SigGen.setFrequency(1000);   //  1000 Hz.
  SigGen.setWave(AD9833_TRIANGLE);
  lcd.setCursor(0, 3);
  lcd.print("1000Hz Triangle     ");
  delay(10000);
  SigGen.setFrequency(1000);   //  1000 Hz.
  SigGen.setWave(AD9833_SINE);
  lcd.setCursor(0, 3);
  lcd.print("1000Hz Sine         ");
  delay(10000);
}
```

And finally

- Some silly bonkers maths code to calculate Pi (ish)

```
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Starting...");
  StartTime = millis();
  for (float k = 0.0; k <= numofTerms; k++)
  {
    pi += 4.0 * ( pow((-1.0), k) ) * ( 1.0 / (2.0 * k + 1) );
  }
  EndTime = millis();
  ElapsedTime = EndTime - StartTime;
  Serial.print("Pi calculated as ");
  Serial.println(pi);
  Serial.print("It took ");
  Serial.print(ElapsedTime);
  Serial.println(" Milliseconds");
  delay(2000);
}
```

```
Starting...
Pi calculated as 3.14
It took 29601.00 Milliseconds
Starting...
```

And finally

- Using 100,000 as the number of terms:

30.14999961853027343750	Arduino Mega
29.60099983215332031250	Arduino Uno/Nano
7.46299982070922851562	Arduino Zero
7.45300006866455078125	SAMD21 Mini
3.29900002479553222656	STM32F303
3.07200002670288085937	STM32F103C (Blue Pill)
2.96900010108947753906	Arduino Due
2.89599990844726562500	STM32L432KCU6
1.87699997425079345703	STM32F411
0.66799998283386230468	ESP8266

- Quantum computer…. Instant (‘cos it can do them all at once, right?)