Embedded Systems Fundamentals

ENGD2103

Dr M A Oliver

michael.oliver@dmu.ac.uk

Lecture 1: Introduction

Contents

- Staff
- Module Organization
- Assessments
- Introduction to Embedded Systems
- Introduction to Arduino
- General Purpose I/O.
- LED Blinking

Staff

- **Dr Mike Oliver** michael.oliver@dmu.ac.uk
  Module Leader, Lectures, Laboratory Sessions, Marking and Moderation.

- **Dr Tanvir Allidina** tanvir.allidina@dmu.ac.uk
  Laboratory Sessions, Marking and Moderation.

- **Dr John Gow** jgow@dmu.ac.uk
  Guest Lectures, Marking and Moderation

Module Organization

- **Lectures**

  One 1-hour lecture per week (face-to-face)

- **Laboratory Sessions**

  One 2-hour laboratory session per week in Q1.01.

- **Self-directed Study**

  You are expected to work on your own on the coursework and use the lab sessions for fault-finding, problem-solving, Q&A, etc.

Outline Module Content

- Introduction
- General Purpose I/O
- Hardware Adaption Layer (HAL)
- 7-Segment Display & Shift Register
- Accelerometer & I^2C Communications
- “Bare-Metal” Programming
- Concurrency
- Finite State Machines
- Schedulers
- Coding techniques (in both C and C++)

Assessments

Formative Assignment

- Counts towards 0% of the module.
- Demos in Week 5.
- Feedback only
- Relevant to the summative assignment

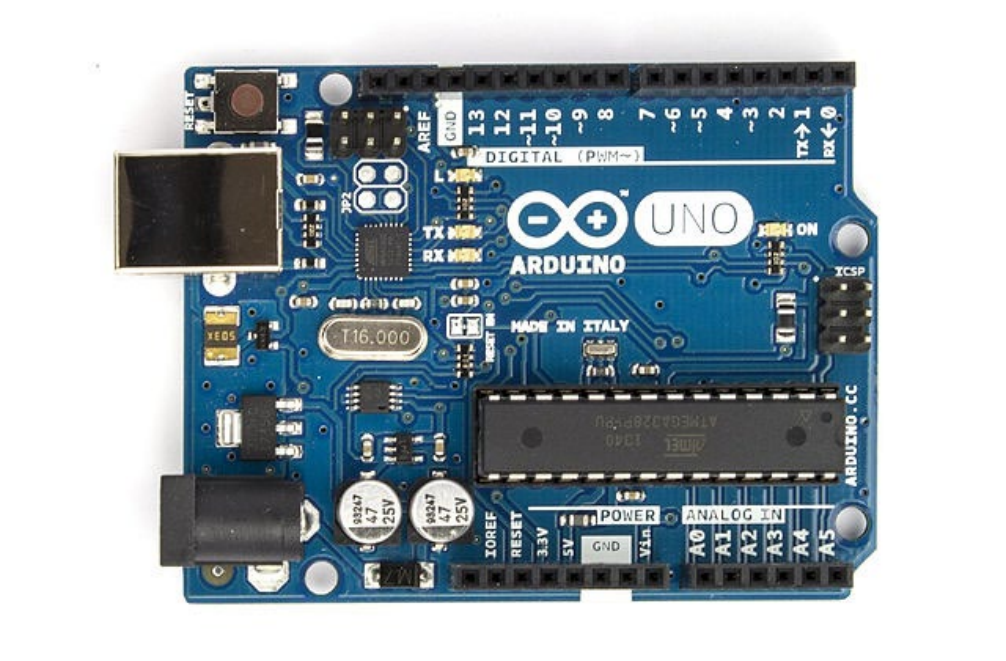
Summative Assignment

- Counts towards 100% of the 15-credit module.
- Submission in Week 11, Demos in Week 15.

What is an Embedded System?

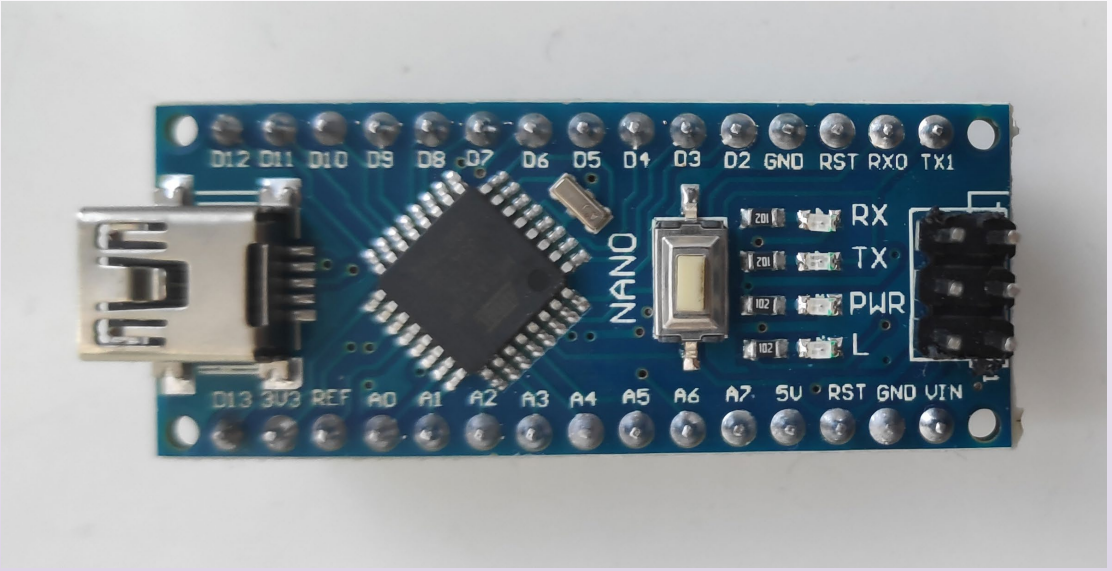
- Computer System having dedicated function within a larger system.
- A combination of computer hardware and software (firmware) for a particular function.
- Often based on microcontrollers
  - Integrated circuits with peripherals that interface with the outside world.
  - Computers in their own right.

Introduction to Arduino



- Open-source prototyping platform. The Arduino Uno is the widely-known member of the range.
- Overview, tech specs, documentation (including schematics):
  https://store.arduino.cc/products/arduino-uno-rev3/
- Photograph Acknowledgement: Creative Commons, oomlout, 2014

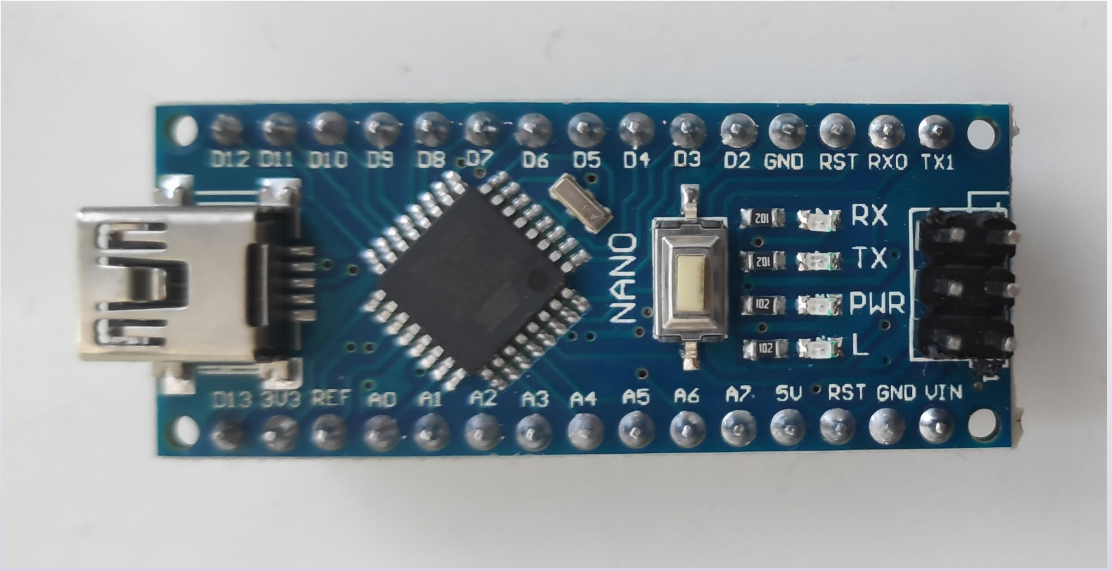
Arduino Nano Hardware



- The Arduino Nano can be considered to be a space-saving Uno.
- Overview, tech specs, documentation (including schematics):
  https://store.arduino.cc/products/arduino-nano

- Photograph Acknowledgement: Creative Commons, MegaMegaAndOmega, 2021

Arduino Nano Hardware



- Based on ATMEL ATmega328p microcontroller.
- 14 digital I/O (GPIO) pins D0 – D13
- 6 PWM outputs: pins D3, D5, D6, D9, D10, D11
- SPI supported on pins D10, D11, D12, D13
- I2C supported on pins A4, A5
- Interrupt pins: D2, D3
- 8 Analog Inputs: A0-A7 (Pins A0-A5 can be configured as digital I/O)
- 10 kHz 10-bit ADC.
- 16 MHz clock.

Arduino IDE

Verify (Compile)

Verify (Compile) and Upload

Arduino Software

- The cross-platform IDE (Integrated Development Environment) is open-source and freely available from:
- www.arduino.cc

- An Arduino program is a C (strictly C++) program.

- An Arduino program *does not* have an explicit `main()` function.

- The name **main** is still reserved and you cannot use it in an Arduino program because a **`main()`** function is created by the compiler when an executable is being built.

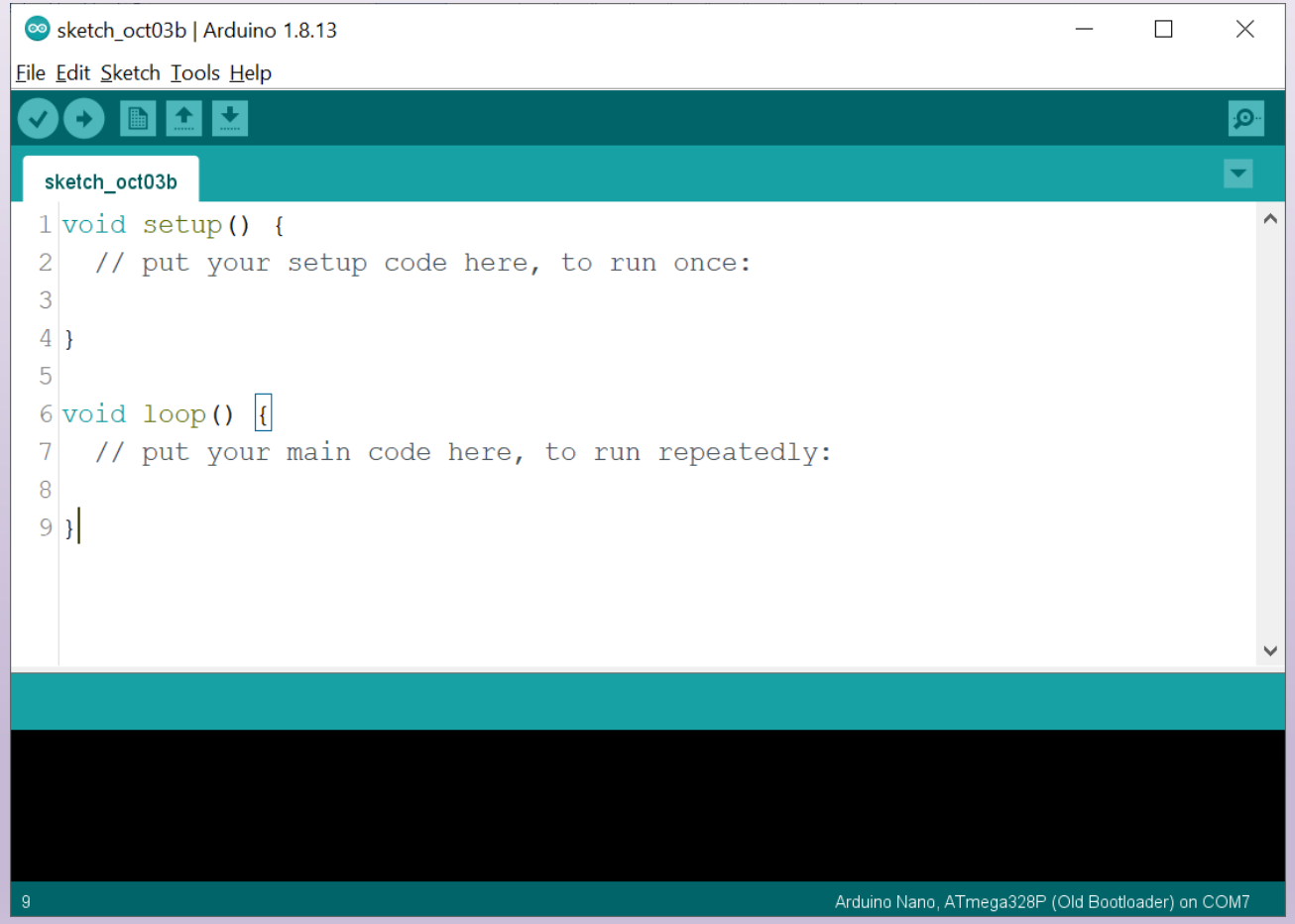
Arduino Software

All Arduino code must contain two C functions:-

`setup()`
Primarily for initialization code. This code is run before anything else.

and

`loop()`
For code that is run forever after `setup()` is called.



Arduino Software – The hidden `main()` function

The `main()` function is hidden. In simplified form:-

```
int main(void)
{
    init();
    setup();

    for(;;)
    {
        loop();
        if (serialEventRun) serialEventRun();
    }
}
```

General Purpose Inputs / Outputs (GPIO) / Digital I/O

- Digital signals can have two states:-
  - High (voltage around V_{DD})
  - Low (voltage around 0V)

- For digital logic, supply voltages of 5V, 3.3V or 1.8V are typical.

- For 5V TTL Logic
  - A voltage > 2V is considered a ‘1’ (high)
  - A voltage < 0.6 is considered a ‘0’ (low)

General Purpose Inputs / Outputs (GPIO) / Digital I/O

- The Arduino Nano (strictly the ATmega328P) has 14 dedicated I/O pins and 6 analog pins (that can be configured as digital lines)

- Each Digital I/O (GPIO) line can be configured:
  - As an output,
  - As an input, or
  - As an input with internal pull-up resistor

- All Digital I/O lines default to input on a power-on reset.

- This is done using register access – something that will be encountered in a few weeks' time……

(GPIO) / Digital I/O Pin Configuration

- Meanwhile the `pinMode()` library function can be used to configure a pin. It takes two arguments:-
  - The first is the pin number
  - The second is the mode.

- Example: Configuring digital pin D5 as an output:-

```
pinMode(5, OUTPUT);
```

- Example: Configuring digital pin D6 as an input:-

```
pinMode(6, INPUT);
```

- Example: Configuring digital pin A3 as an input with pullup resistor:-

```
pinMode(A3, INPUT_PULLUP);
```

(GPIO) / Digital I/O
Digital Writes

- The `digitalWrite()` library function can be used to set the state of a digital output. It also has two arguments:-
  - The first is the pin number
  - The second is the state (`HIGH`, `true`, `1` or `LOW`, `false`, `0`).

- Example: Pulling digital pin D5 to logic HIGH (i.e. logic 1):-

```
digitalWrite(5, HIGH);
```

- Example: Pulling digital pin D5 to logic LOW (i.e. logic 0):-

```
digitalWrite(5, LOW);
```

(GPIO) / Digital I/O
Digital Reads

- The `digitalRead()` library function can be used to read and return the state of a digital line. It has a single argument:-
  - The pin number

- It returns 1 if the state of the line is HIGH, or 0 if the state of the line is LOW.

- Example: Reading the state of line D6 and storing it in the variable: `status`

```
bool status = digitalRead(6);
```

- Example: Reading the state of line A3 (configured as a digital input) and storing it in the variable: `status`

```
bool status = digitalRead(A3);
```

Blocking Delays

- The `delay()` library function creates a blocking delay
- Its argument is the delay time in milliseconds.
- When a `delay()` function is executed, the processor cannot do any other task until completed. In other words it blocks any other operation

- Example: Create a blocking delay of 750 ms.

```
delay(750);
```

- Caution: When developing code for concurrency, some good advice regarding blocking delays. Avoid, avoid, avoid!

Blink An LED (Hardware)

- An Arduino Nano will be used to blink an LED.
- Require knowledge of the physical characteristics of the LED.
- Need to know maximum current values for the pins on Arduino.
- Series resistor needs to be used to protect LED and Arduino.
- At 5V, maximum DC current per I/O pin is 40mA
  - 20mA recommended maximum.
- Check the following URLs:-

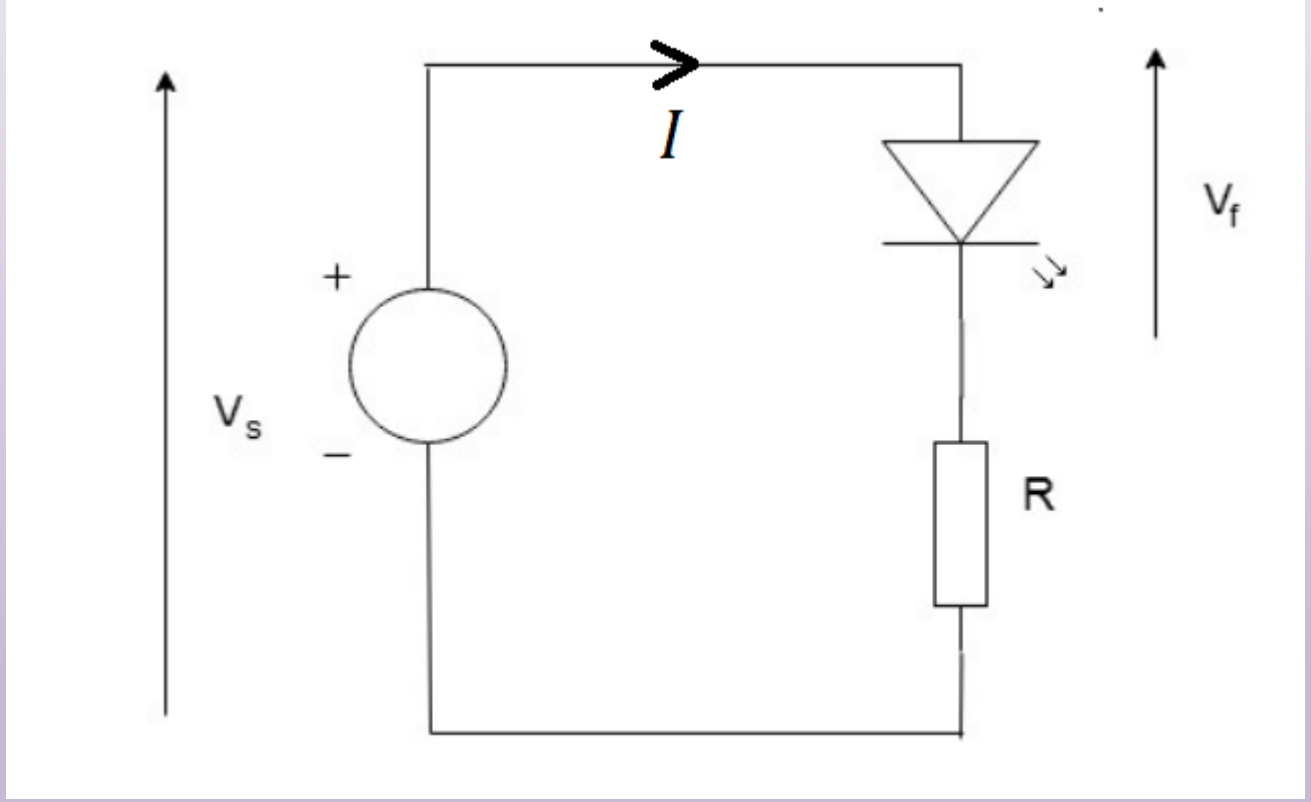
https://playground.arduino.cc/Main/ArduinoPinCurrentLimitations/

https://arduinoinfo.mywikis.net/wiki/ArduinoPinCurrent

Blink An LED (Hardware)

- V_s is logic 1 voltage from digital pin
- V_f is forward voltage for LED (from LED datasheet)
- I should be less than the peak current (from LED datasheet)
- Calculate R (Ohm's law)

$$R = \frac{V_s - V_f}{I}$$

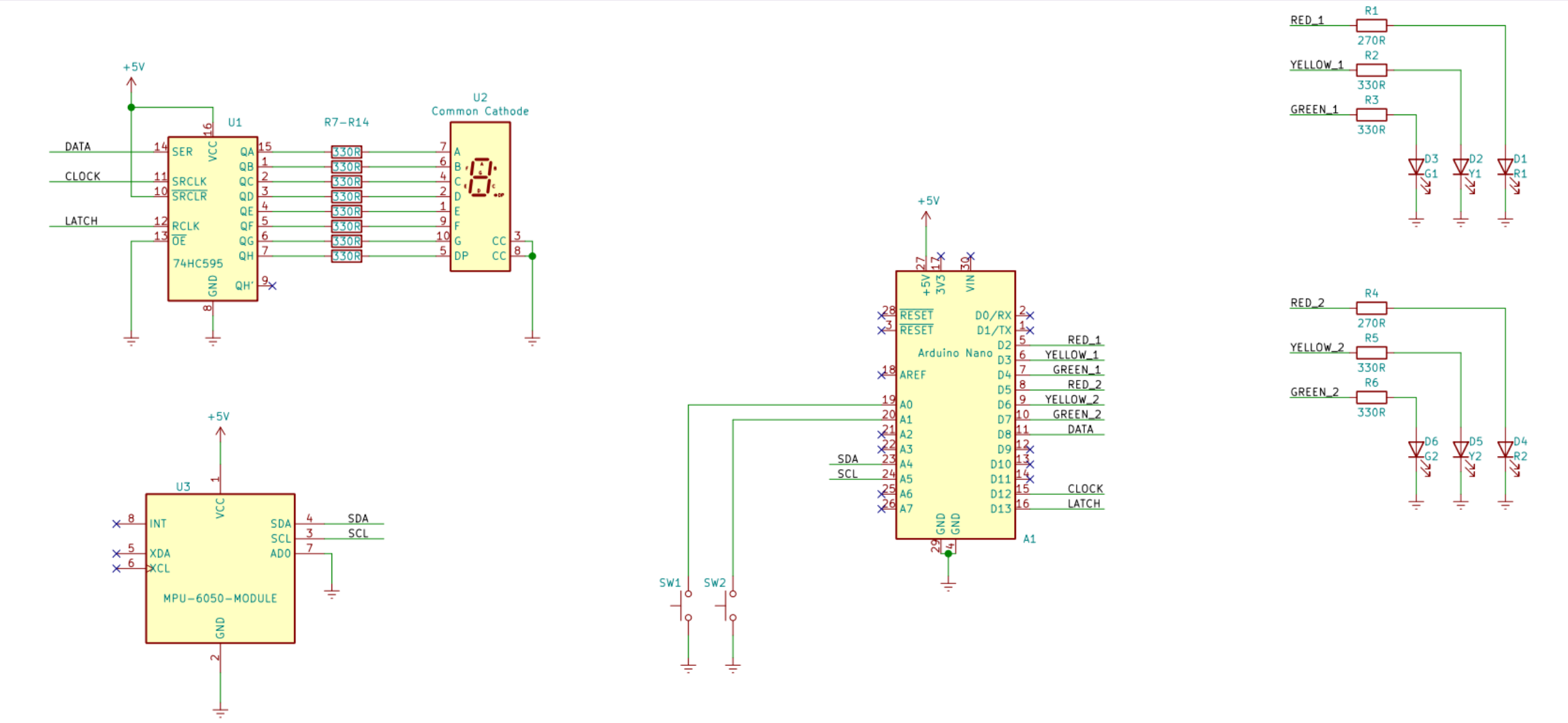


Blink An LED (Code)

- Using the built-in LED on digital pin 13.

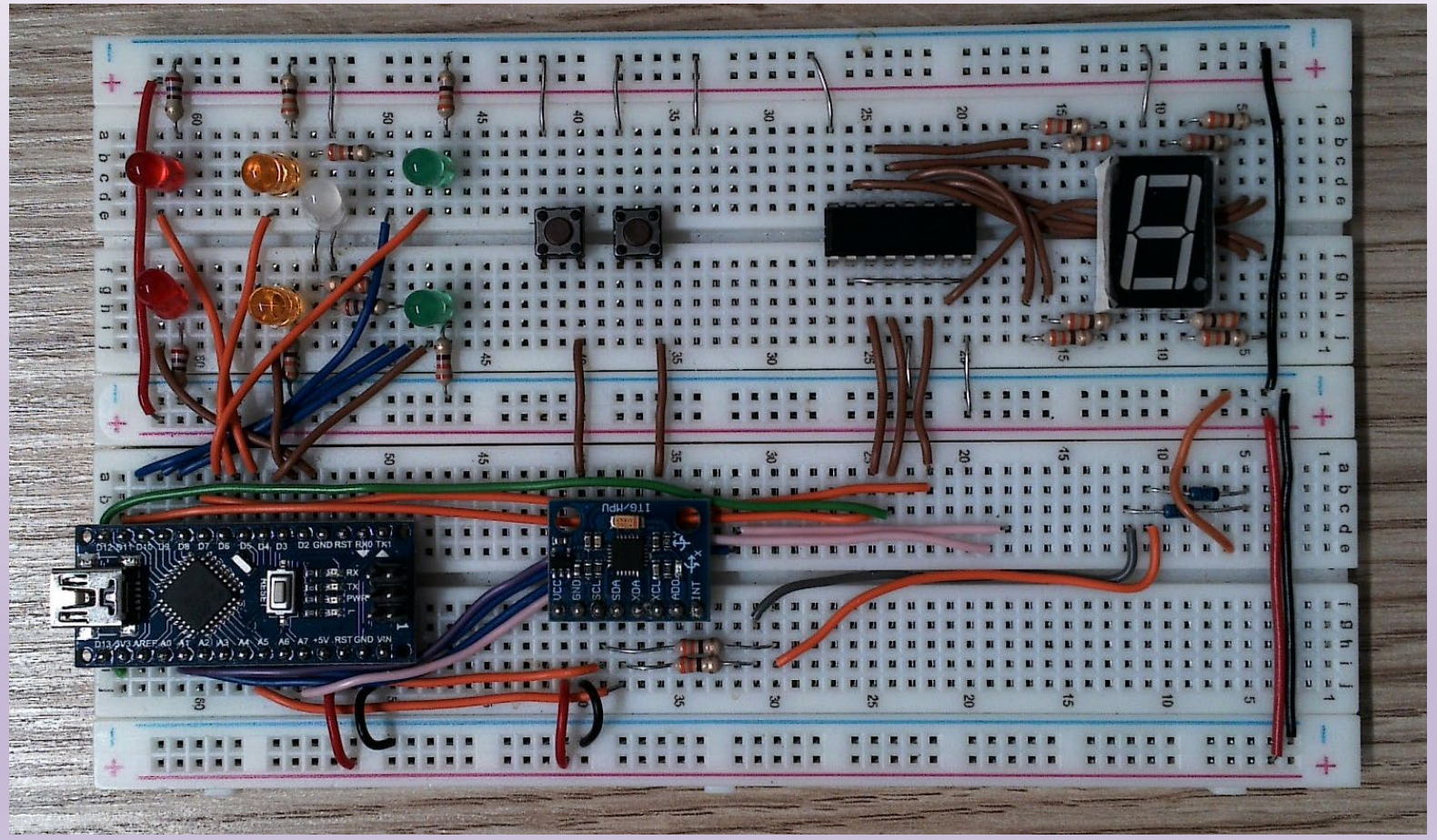
```
#define led      13                // Using the built-in LED.

void setup()
{
    pinMode(led, OUTPUT);          // Make the LED pin an output
}

void loop()
{
    digitalWrite(led, HIGH);       // Turn the LED on
    delay(500);                    // Wait for 500 ms
    digitalWrite(led, LOW);        // Turn the LED off
    delay(200);                    // Wait for 200ms
}
```

Coursework Schematic



Breadboard Layout

- You will all receive a kit of components.
- This kit is <u>your responsibility</u>.
- Tidy robust construction encouraged.



Blink An LED (Revisited)

- Once you have finished your construction, you will have 6 LEDs on the breadboard.
  - Red 1 on Digital Pin 2.
  - Amber 1 on Digital Pin 3.
  - Green 1 on Digital Pin 4.
  - Red 2 on Digital Pin 5.
  - Amber 2 on Digital Pin 6.
  - Green 2 on Digital Pin 7.

- You have seen how to blink an LED.
- Now go ahead and test these 6 LEDs…...

Summary

- `pinMode()`, `digitalWrite()` and `digitalRead()` are Arduino-specific library functions. They are good for people just starting out. Beyond Week 5, they will be forsaken in favour of low-level register access (bare metal).
- Blocking delays prevent the processor doing anything else whilst the delay is being executed. Ultimately need a different approach for timing.
- Next week:-
  - Hardware Adaption Layer (HAL)
  - Seven-segment displays and shift register.