LED DESIGN

1 Summary

The calculator requires a visual display that allows the user to interact with the calculator and interpret its answers. To achieve this a simple 7 segment display will be used. The design illustrated in ***Error! Reference source not found.*** considers the multiple requirements and restrictions that come with this project and addresses them through design choices. All the major design choices are motivated in this report.

A common anode LED will be used. It has a maximum brightness of 15mcd, however it will be operated at 6.3mcd as required. The common anode will be connected to the 5-volt power pin on the Arduino. Each LED segment will be connected in series with a 150Ω resistor to attain the required 20mA current. Each of the eight cathodes are then grounded at digital I/O pins (3, 5-11). These pins will be connected to the Arduino's internal pullup resistors to tie them to a logical high and ensure no unwanted power is consumed while each segment is not in use. All components are operating below their maximum ratings to ensure the reliability and longevity of the calculator.

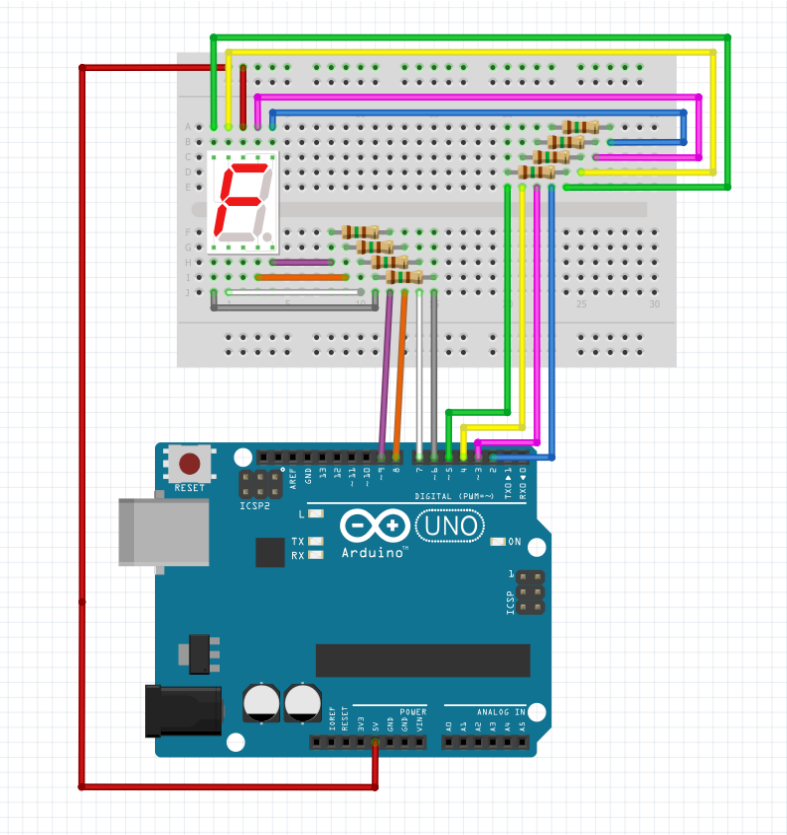

Figure 1: Fritzing wiring schematic for LED circuit design.

The LED's brightness is going to be controlled using a PWM signal with a 42% duty cycle to achieve the required 6.3mcd brightness. There are only 6 hardware PWM pins available, so the remaining two LED's will be supplied with a software PWM signal.

Table 1 Bill of Materials

2 Bill of Materials

Component	Digikey Product Code	QTY	Price (AUD)
Arduino Uno	N/A	1	Provided
7 Segment LED Display	1830-1151-ND	1	1.49
Solderable Breadboard	1778-PIM528-ND	1	1.32

150 Ω Resistor	RNF14FTD150RCT-ND	8	0.13
Connections	N/A	N/A	Negligible for prototype
		TOTAL	**4.114**

3 Requirements

- Single digit seven segment LED display to be controlled by an Arduino Uno.
- $4.5 AUD budget for all components excluding Arduino Uno.
- Outputted results larger than a single digit will cycle through display on a 0.5 Hz cycle.
- Minimum LED brightness of 6.3 mcd [1].
- LED circuit maximum average power consumption of 320 mW.

4 Assumptions/Specifications

- Users are not professional typists, hence the 0.5 Hz display cycle.
- Calculator will not be mounted directly in front of a light source including direct sunlight.
- Calculator will be viewed directly from less than 1m away.
- Office background light intensity is no greater than 750 Lux [2].
- Calculator will be mounted in normal room temperature conditions (10°C – 35°C).

5 Arduino Uno

Figure shows the pinout for the Arduino Uno. The Arduino Uno has 14 digital I/O pins. 6 of these have inbuilt hardware to produce PWM signals. These pins are 3, 5, 6, 9, 10, and 11. There are also a further 6 analog I/O pins. These can be operated as digital pins, however this comes at a cost of increased power consumption through the ADC.

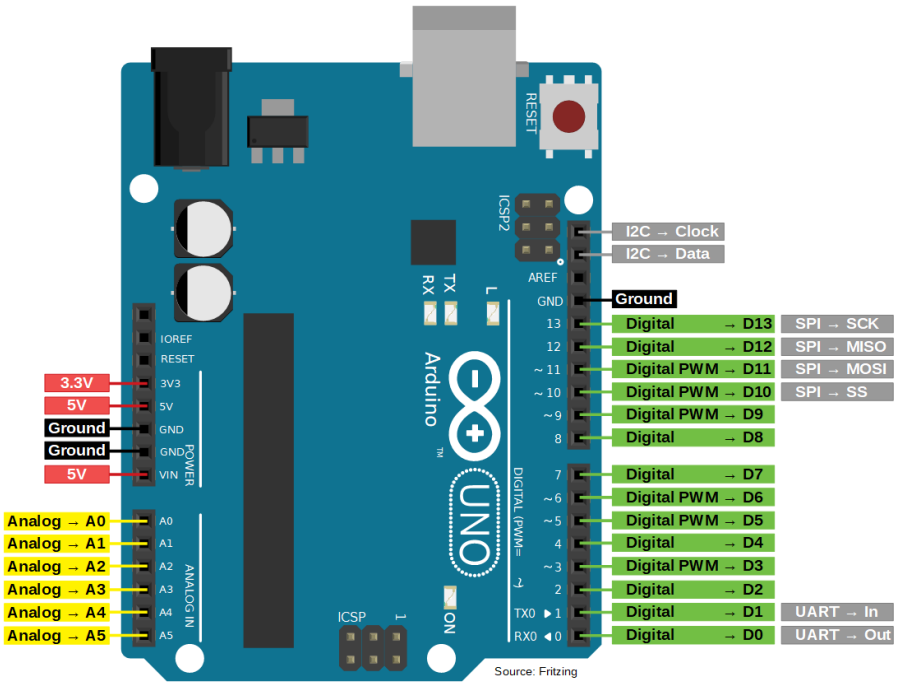

Figure 2: Arduino Uno pin map.

Table 2: Arduino Uno major specifications [3].

Microcontroller	**ATmega328**
Input Voltage	7V – 12V
Number of Digital I/O Pins	14 (6 of which can provide hardware PWM signal)
Number of Analog I/O Pins	6
Maximum Current Per I/O Pin	40mA
Maximum Current on 3.3V Supply Pin	50mA
Maximum Current on 5V Supply Pin	500mA
Maximum Combined I/O Current Draw	200mA
Maximum Current Available to Arduino Through USB Supply	500mA
Maximum Current Available to Arduino Through Barrel Power Connector Supply	1A
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
I/O Internal Pull-Up Resistors (Does not have internal pull-down resistors)	20 $K\Omega$ – 50 $K\Omega$ (automatically disconnected and disconnect on reset/boot).
I/O Bootup State	Set to input.
Boot Time (progress from boot loader to installed code)	6 – 8 seconds.
ADC Current Consumption	9.2mA

6 7 Segment LED Display Selection

The display specified in Table 3 will be used.

Table 3: Major specifications of selected LED [4].

INOLUX 7-SEGMENT LED DISPLAY		
Specification	**Details**	**Meets Specification**
Construction	Common Anode	Yes
Colour	Yellow – Green	Yes
Peak Wavelength	572nm	Yes
Forward Voltage	2V	This is well below the operating voltage of the Arduino, meaning it will be easily driven.
Typical Current	20mA	This is half of the capability of the Arduino's digital I/O pins (maximum 40mA). This will prolong the lifespan of Arduino.
Power Consumption (MAX)	70mW	Yes
Brightness	15mcd	The LED will be run at 6.3mcd. Over specifying allows room for future brightness increase if required. Also prolongs LED lifespan.
Decimal Point LED		Yes

6.1 Brightness

Must be 6.3 mcd minimum.

According to international lighting manufacturer Airfal, the background light intensity in office settings ranges from 150 Lux (waiting rooms and low intensity areas) to 750 Lux (technical drawing studios) [2]. From a report written by Jerry Wachtel, president of the Veridian Group Inc, the recommended brightness for a display in a 750 Lux environment is approximately 1000 cd/m^2 [1]. Taking the average surface area of a single segment on a display as 6.3×10^{-6} m^2, the minimum required candela rating of the display is 6.3 mcd.

6.2 Colour

Must be yellow, red, or orange.

The colour for the LED should bring the maximum amount of attention towards the digits being displayed. In a study comparing non-coloured and coloured multimedia presentations it was found that the coloured presentations resulted in better attention than the non-coloured presentation [5]. Furthermore, another study showed "that warm colours such as yellow, red and orange have been found to have a greater effect on attention compared to the cool types of colours like brown and grey" [6].

6.3 Internal Construction

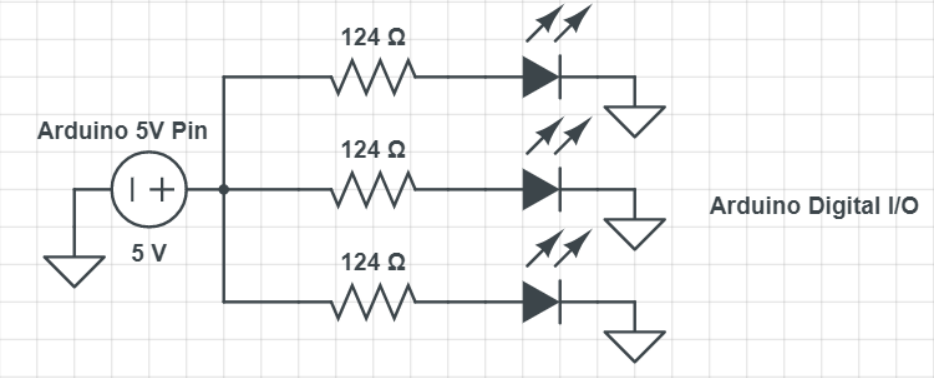

Figure 3: Common cathode seven segment LED display diagram.

Must be common anode as explained in section 7.1.

7 Segment displays have two setup types, common cathode, and common anode. In a common anode display power will be applied to a common voltage source [7]. Then if an individual segment is grounded the LED will turn on. For a common Cathode display, the LED's all share a common ground. Each will have a unique voltage source, which when turned on will power the respective LED [7].

6.4 Electrical Characteristics

The Arduino must be capable of driving the selected LED. This means that the forward voltage must be below 5V and have a current draw less than 40mA (Table 2).

6.5 Decimal point

A decimal point is not a requirement for the agreed upon specifications. However, for future expandability an LED display with a dedicated decimal point LED will be selected. Software modifications will be all that is required to integrate decimal point, making it a faster and cheaper process than having to redesign the hardware as well.

7 Protection

7.1 Floating Voltage

Arduino internal pull-up resistors will be used.

While not in use the Arduino I/O pins do not default to a set state, meaning that any components connected to these will see a floating voltage. Electromagnetic radiation, temperature and other factors can cause the voltage of the pin to drift anywhere in the available 5V range. This may result in any connected devices consuming unwanted power or providing unwanted inputs. To prevent this a high value resistor pull-up or pull-down resistor is used to tie the pin to a logical high (5V) or low (0V) without drawing significant current.

A common anode display must use pull-up resistors. To prevent current flowing the digital pins must be set to the same potential as the common 5V source, hence they must be set to a logical high while not in use. Contrarily a common cathode must use pull down resistors to prevent current flowing when each LED segment is not in use. This sets the digital pin voltages to the same potential as the common ground, preventing current flow.

The budget does not have room for the inclusion of a further 8 resistors (1 for each led) to implement external pull-up or pull-down resistors (the lowest single purchase price on Digikey is $0.13). This means that the Arduino's internal pull-up resistors (Table 2) which can be implemented using the pinMode(*pin*, INPUT_PULLUP) function must be used. This in turn means a common anode display must be used.

7.2 Boot Time

As stated in Table 2, the default state of the Arduino's internal pull up resistors is disconnected and must be connected using the pinMode(*pin*, INPUT_PULLUP) function. The Arduino has a 6-8 second boot time before the installed program is run and these pull-up resistors are connected. During this time, the connected LEDs will see a floating voltage. This will likely cause them to draw power and dimly illuminate during this time. This will not damage the LEDs, as they and the respective support circuitry are designed to handle constant 20mA hour current at up to 5V (maximum the Arduino can supply).

This is an unavoidable tradeoff in performance. The solution to this issue is the implementation of external pull-up resistors which the budget does not allow for, hence the unwanted power consumption must be accepted.

7.3 Current Limiting Resistors

A 150 Ω resistor will be placed in series with each LED. Resistors of a $\pm 1\%$ tolerance will be used to ensure minimal variation from the desired value.

The LED's must be connected in series with a resistor to limit the forward current the LED experiences. While a large forward current can result in a brighter LED it can also permanent damage to the LED. Omitting a resistor could also seriously damage the I/O port, the current outputted by the pin will be limited to the maximum current that the pin is capable of sinking.

In accordance with Table 3, the forward voltage of the LED is 2V and a 20mA current draw is required. Applying Ohm's law, the required series resistance to draw 20mA is:

$$R = \frac{V}{I}$$

$$R = \frac{5V - 2V}{20mA}$$

$$R = 150\Omega$$

7.4 Current Sinking Limitations for I/O Pins

The Arduino Uno's digital I/O pins are equally capable at both current sinking and sourcing. Each individual pin can handle up to 40mA at 5V, any more will permanently damage the pin [8]. The specified seven segment display is a common anode and will draw 20mA. This means that each digital I/O pins will be required to sink 20mA. The maximum current sink from all the digital I/O pins combined is 200mA. The maximum number of simultaneously operated LEDs will be 7, as the decimal point will only ever be operated individually. This gives a maximum of $20mA \times 7LEDs = 140mA$ to be sunk across 7 digital I/O pins. Operating these pins well under maximum capacity will protect them and prolong their lifespan.

8 Software

8.1 Controlling Displayed Number

As discussed above the display will be driven by the digital I/O pins of the Arduino. These pins will be controlled by software uploaded to the Arduino so the calculator can be operated in a workplace environment. This code will be split up into multiple header files with one header file dedicated to operating the seven-segment display. This header file will allow the code to call multiple functions that run the display. For example, we will require a setup function that declares all digital pins used for the display to be inputs, a function to display zero etc.

To set each pin to be an input to sink the current the *pinMode()* function will be used. If the pins are not set to be an input, the LEDs may appear dim when toggled [9]. The pins will be individually set to HIGH or LOW by the *digitalWrite()* function. This inbuilt function sets the pin to either 5V or 0V. Each digit will be displayed for 0.5 seconds, to achieve this the *delay()* function will be used. *delay()* pauses the program for a specified time. An example of these functions being used is shown in *appendix 1.*

8.2 Controlling Brightness

The brightness of the LEDs in the display will be controlled using a PWM signal. This is a more suitable control method than adjusting the value of the series resistor with each LED as only the software must be updated should brightness require changing. A PWM signal effectively switches the pin between high and low in accordance with its respective duty cycle. 100% duty cycle means always on, 50% means on half the time [10]. There are 8 pins that require PWM control. Only pins 3, 5, 6, 9, 10, and 11 have hardware PWM registers, meaning the other two pins, 7 and 8, will require a software implemented PWM signal. The duty cycle of the hardware PWM registers is easily manipulated using the *analogWrite*(pin,value) function. A software PWM cycle can be manually developed in a process known as bit banging, in which the selected pin is manually switched between high and low repeatedly in accordance with the required duty cycle as exemplified below [12].

```
digitalWrite(pin, HIGH);

delayMicroseconds(425);
```

digitalWrite (pin , low);

delayMicroseconds(575);

The selected Display has a maximum brightness of 15mcd. Only 6.3mcd is required, meaning the required light percentage can be found to be 42.5%. This translates directly to a 42.5% duty cycle. Assuming the frequency of the duty cycle is at 60Hz or above, the LED will appear to produce solid illumination to the human eye at 6.3mcd [11]. Should greater brightness's be required, up to 15mcd, the duty cycle must be increased.

9 Power Consumption

The maximum power drawn by the display circuit will be $20mA \times 5V \times 7 = 700mW$. However as we are operating the LEDs on a 42% duty cycle the maximum average power consumption for the circuit will only be $700mW \times 0.42 = 294mW$, this is below our maximum average power requirement of 320 mW.

10 Incorporating Keypad

As the keypad circuit will also draw/sink current from the I/O pins the design must consider the combined current draw/sinking between the keypad and the LED does not exceed the 200mA maximum. Therefore, to minimise power consumption and stress on the Arduino I/O pins the two circuits will never be operating at the same time.

When a key is pressed the Arduino will store the value and then disable the keypad circuit, so it's not using any power. The LED circuit can then be enabled and the user's input will be shown on the display. Once the digit has been displayed for long enough the LED circuit will again be disabled and the keypad circuit will be enabled, with the system ready to repeat the process.

11 Construction

To construct the LED support circuit and connect this to the LED and Arduino a PERF board will be used in accordance with a series of jumper wires. A PERF board has been selected as they are the cheapest available development board, and budget is the highest priority requirement. Soldered joints will also provide a more robust prototype. The LED display and resistors will be mounted upon the PERF board with solder bridges to make the required connections. Connections between the Arduino and the PERF board will be made using male to male jumper wires.

12 Future Developments

The Arduino's I/O pins contain internal pull up resistors, these pull up resistors are utilized in the providing the HIGH signal to our common anode 7 segment led display. On startup though theses pull up resistors are not instantly initialized as the Arduino requires 6-8 seconds to check for updates to the code being provided. Therefore, during this period the voltage at the I/O pins will be floating. This is not an ideal situation as with the common anode setup the pins may appear at a high to the 7-segment display, causing the LEDS to become dimly illuminated for a short this short period of time. If this is a major issue there is the possibility of the 24 mcd, common cathode configures INND-TS40RCB 7 segment display. Using this 7-sgement display would require the use of external pull-down resistors to ensure the state of the voltage

from the I/O pins. This is a more expensive method but will prevent the floating voltages on startup and therefore is a possibility to be implemented as the project develops.

A PERF board is an effective development board for prototyping electronics. However, should the prototyping phase be successful and bulk quantities of the calculator be required a custom printed PCB should be produced. An Arduino Uno hat style PCB in which the PCB has header pins that line up with the pin rails of the Arduino Uno board would likely be the best solution for this. Using a custom PCB also opens up the possibility of using surface mount components, which are generally cheaper.

13 Appendix

```
void setup()
{
  for(int i=2; i++; i<10){ //configures pins 2 through 8 to be outputs and writes them at off(LEDs are off)
pinMode(i, INPUT);
digitalWrite(i, off);
}
}

void displayZero(){
  digitalWrite(segmentA, on);
  digitalWrite(segmentB, on);
  digitalWrite(segmentC, on);
  digitalWrite(segmentD, on);
  digitalWrite(segmentE, on);
  digitalWrite(segmentF, on);
  digitalWrite(segmentG, off);
  delay(displayNumberTime);
}

void clearDisplay(){
  digitalWrite(segmentA, off);
  digitalWrite(segmentB, off);
  digitalWrite(segmentC, off);
  digitalWrite(segmentD, off);
  digitalWrite(segmentE, off);
  digitalWrite(segmentF, off);
  digitalWrite(segmentG, off);
  delay(clearTime);
}
```

Appendix 1 example display code utilizing the pinMode(), digitalWrite() and delay() functions

14 References

[1] J. Wachtel, "Report on Digital Sign Brightness," The Veridian Group, November 2014. [Online]. Available: https://scenicnevada.org/wp-content/uploads/Condensed-Verstion-Part-I-Wachtel-Study-1.pdf. [Accessed 27 March 2021].

[2] Airfal International, "Recommended lighting levels for offices," Airfal International, [Online]. Available: https://www.airfal.com/en/residential-lighting-news/recommended-lighting-levels-for-offices-4265/. [Accessed 27 March 2021].

[3] Atmel, "ATmega328P DATASHEET," Atmel, 2015. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. [Accessed 29 March 2021].

[4] Inolux, "INND-TS40 Series 0.4” Through Hole Single Digit Display," Inolux, [Online]. Available: http://www.inolux-corp.com/datasheet/Display/Through-Hole-Display/SingleDigit/INND-TS40%20Series_V1.0.pdf. [Accessed 30 March 2021].

[5] F. FH, "Grant AP. Arousal and cognition: Memory for color versus black and white multimedia presentation.," 1976. [Online]. Available: https://scholar.google.com/scholar_lookup?journal=J+Psychol&title=Arousal+and+cognition:+Memory+for+color+versus+black+and+white+multimedia+presentation&author=FH+Farley&author=AP+Grant&volume=94&issue=1&publication_year=1976&pages=147-150&. [Accessed 30 March 2021].

[6] J. Gombak, "The Influence of Colour on Memory Performance: A Review," 20 March 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3743993/#b13-mjms-20-2-003. [Accessed 30 March 2021].

[7] Engineers Garage, "Difference between Common Anode and Cathode seven segment display," Engineers Garge, 9 August 2019. [Online]. Available: https://www.engineersgarage.com/knowledge_share/common-anode-and-cathode-7-segment-display/. [Accessed 31 March 2021].

[8] ARDUINO, "ARDUINO UNO REV3," ARDUINO, 2021. [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed 30 03 2021].

[9] ARDUINO, "digitalWrite()," ARDUINO, 25 06 2019. [Online]. Available: https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/. [Accessed 30 03 2021].

[10] M. Raynsford, "Arduino Hardware PWM for Stepper Motor Drives," Instructable Circuits, [Online]. Available: https://www.instructables.com/Arduino-Hardware-PWM-for-stepper-motor-drives/. [Accessed 30 March 2021].

[11] Canadian Centre for Occupation Health and Saftey, "Lightin Ergonomics," 31 August 2018. [Online]. Available: https://www.ccohs.ca/oshanswers/ergonomics/lighting_flicker.html#:~:text=People%20can%20see%20lights%20flashing,which%20it%20starts%20to%20decrease. [Accessed 29 March 2021].

[12] stevenvh, "Do I really need resistors when controlling LEDs with Arduino?," Stack Exchange, 31 May 2012. [Online]. Available: https://electronics.stackexchange.com/questions/32990/do-i-really-need-resistors-when-controlling-leds-with-arduino#:~:text=The%20short%20answer%20is%2C%20yes,is%20pretty%20high%2C%20see%20this.. [Accessed 3 March 2021].

[13] T. Hirzel, "PWM," Arduino, 5 February 2018. [Online]. Available: https://www.arduino.cc/en/Tutorial/Foundations/PWM. [Accessed 29 March 2021].