NSN: 0129972568

OS

Introduction
In my report I will research a variety of complex concepts that I have used in developing my project. My project is an electronic dice that allows users to play a variety of minigames. My project contains a variety of components, such as an, dot matrix, microphone, accelerometer and a touch sensor that are all used together to create independent learning games for young children. Although my project will consist of a variety of independent games, the main use of my project is to be used as an electronic dice. The dice function of my project will use an accelerometer to know when to randomly generate a random number (between 1-6 in a dice pattern) and display it on the dot matrix. The dice function will also be activated via voice (using a microphone) as well as being shaken. All these components will work together using an Arduino microcontroller.

Educational Dice
My stakeholder works at an early childhood centre as doesn't have any games that teach children how to user technology. As the games that are used at the centre need to follow a philosophy, they don't have any electronic learning games. As my stakeholder doesn't have any games that for fill this need, I will be creating an educational dice that will teach young children how to use technology. My device consists of a variety of education games that teaches the end-users.

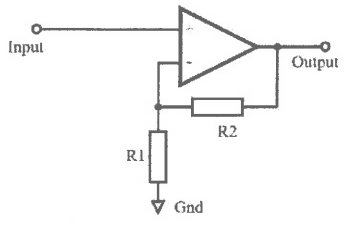
Specifications
- Needs to be small and compact
- My project must teach, and my targeted users must learn and understand.
- Must meet the early childhood centre teaching requirements.
- Must meet the early childhood centres philosophy requirements.

Requirements
- Education games (e.g. dice roll, clap to a sound pattern etc).
- Device must not be dangerous, meet health and safety requirements.
- Targeted users must be able to operate my device (users aged between 3 and 5).
- Device should automatically turn on after a period not being used.

Operational Amplifier
An op amp (operational amplifier) is a voltage amplifier that has two inputs and an output. It is designed to be used with feedback components such as a resistor or a capacitor between the input and output. The feedback components decide on the overall functionality of the op amp. An op amp can perform a variety of different operations such as rectifiers and a current to current converter. Op amps are a three terminal component that has two high independence inputs. This means that theoretically no current flows into the inputs. One of the inputs is an inverting input and the other is a non-inverting input. The third terminal is the output terminal which outputs voltage or current. Op amps finds the difference between the two inputs and changes the output by a gain factor that theoretically approximates to infinity.

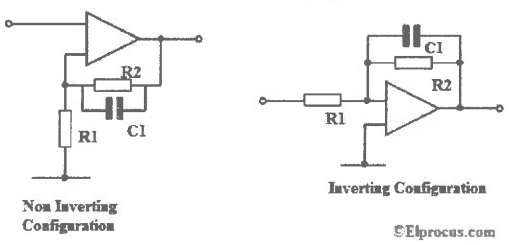
<u>Non-inverting amplifier</u>
A non-inverting amplifier is one of the most commonly used op amps. A non-inverting amplifier provides a high independence (no current flowing through the inputs) as well as the features used in an operational amplifier. The feedback that a

non-inverting amplifier receives is taken from the output of an op amp via a resistor to the inverting input of the op amp where another resistor is taken to ground. It is applied to the inverting input as it is negative feedback. The values of the two resistors, determine the gain of the op amp circuit that decides the level of feedback outputted from the circuit (diagram shown below). The gain of a non-inverting amplifier is easy to determine and can be calculated using the formula shown below. As the input of the op amp takes no current, this means that the current going through both resistors is the same. The voltage for the inverting input is formed from the potential divider consisting of both resistors.



Basic non-inverting operational amplifier circuit

$$Av = 1 + \frac{R2}{R1}$$



The left diagram above is a low pass filter. It is a filter than only allows low frequency signals to pass through. This is because low frequency signals go through much easier with less resistance compared to high frequency signals. If a high frequency signal is present, it will be blocked and unable to pass through. Low pass filters are achieved by using capacitors and resistors in a circuit (shown above).

<u>Impendence</u>
The impendence for a non-inverting circuit is high. The input impendence can be greater than 10^7. This is good as it will ensure that the non-inverting amp causes no loading in the circuit. The lower the impendence, the more current that an op amp draws. The higher the impendence, the lower the current that an op amp will draw. The idea output impendence for an op amp is 0. When an op amp produces its output, it is better to have a zero voltage so that the maximum voltage will be transferred to the output load. The voltage is decided in a circuit depending to the amount of impendence in a circuit. Voltage drops across a component of higher impendence in a circuit. For a voltage to drop cross the output load, that load must be greater impendence than the output of the op amp. Therefore, it is better to have the output impendence of an op amp to be 0.

I will be using op amps in my project to amplify the signal from the microphone to a level that the Arduino microcontroller can detect. As the input level the Arduino receives is quite low, I will need to amplify using an op amp so that the microphone can detect more sensitive sounds that the Arduino microcontroller can detect and act accordingly.

Suitability of a microcontroller
In my project I will be using an Arduino microcontroller to measure and provide the overall function of my project. An Arduino is a programmable component that can be used to measure inputs and outputs. An Arduino microcontroller is based on the programming language C. I have chosen to use an Arduino microcontroller compared to a microcontroller for my project as it is like the Raspberry Pi and less complex. An Arduino microcontroller can achieve my purpose easier with less complex components such as an Arduino microcontroller. Because of this, an Arduino microcontroller suits my project the best.

An Arduino microcontroller is based on the ATmega3280. The main features of this microcontroller that are the most useful to me are its 14-digital input/output pins, 6 analogue inputs, a 16 MHZ quartz crystal and a USB connection. These are important and useful for my project as it enables me to be able to read inputs and outputs, and act when necessarily, achieving my purpose effectively and efficiently.

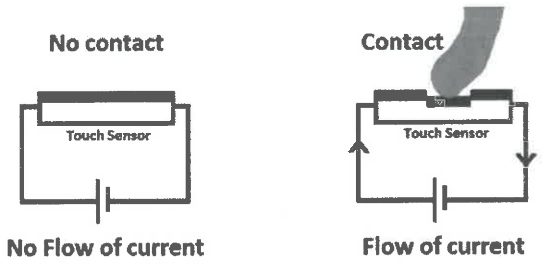
Multiple sensors

Accelerometer
An accelerometer measures the acceleration/movement of itself. It's a device that works by sensing the acceleration of gravity. It measures three linear axes (x, y and z) in the unit of g-force (1g = 9.8ms-2). The accelerometer figures out the acceleration by following a set of formulas the calculate the acceleration by sensing the acceleration of gravity. The acceleration of gravity is figured out by using a system called MEMS (Micro Electro-Mechanical Systems). MEMS is a tiny electronic circuit that has a mechanical nature that creates small mechanical structures that interface to electronics. MEMS consists of silicon and rows of resistors, creating a resistor path. This is attached to a compactor can move around by vibrations, gravity or movement. When it moves it causes the structure containing the capacitor to move around. When it moves around, the distance between the fingers of the capacitor structure changes the capacitance, sensing movement. These readings are then given to the device which then works out the acceleration from calculations.

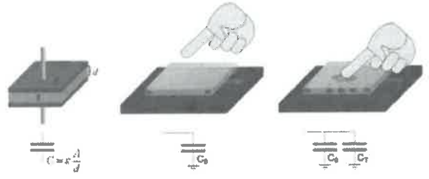
I will be using this in my project to sense when the dice has been shaken. When the dice is shaken the accelerometer will measure the amount of g-force. When a g-force amount has been exceeded (dice is shaken hard enough) then more processes will begin.

Capacitive touch sensor
A touch sensor is a type of sensor that detects touch and enables a device to use this input as a switch that activates an actuator.

Touch sensors are activated upon touch from a human. A touch sensor works by enabling the flow of current when contact is made to the sensor. When touch is present, it will act as a closed switch and will allow current to flow through it. When contact is released it acts like an opened switch and will not allow current to flow through



A capacitive touch sensor is a popular touch sensor as they are robust and easy to use. A compactor touch sensor is made up of two conductors (tends to be metal plates) with an insulator in between.



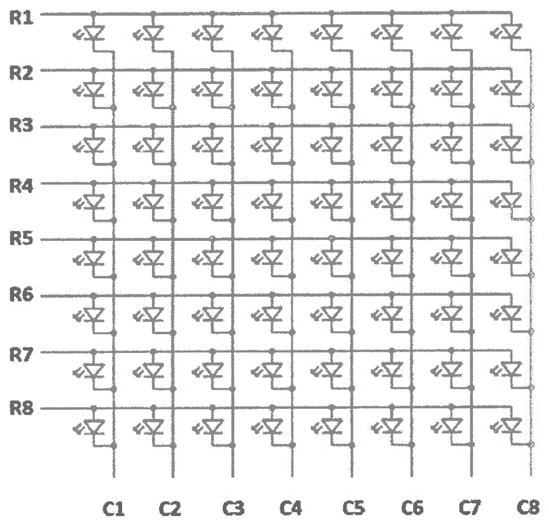
If the conductor plates have a larger area than the insulator material, then there will be a greater value of the touch capacitance. This means that by controlling the size and thickness of the conductor plates, we can control the sensitivity of the of the touch sensor.

I will use these sensors in my project as they are simple and easy to use. As the age demographic of my users is quite young, my inputs cannot be complicated. If they are too hard to use then my targeted users will not be able to operate my device, therefore my device will not meet its intended purpose.

Dot Matrix
A dot matrix is a digital display that has set resolution. Dot matrix's are commonly found on watches, clocks, information on machines and public transport indicators. The display consists of lights in a dot matrix format. Dot matrix's can be configured to represent letters and numbers by turning on a specific light to be on. An 8x8 dot matrix consists of 64 individual lights that can be selectively lit to display a number or an alphabetic character.

In a dot matrix, multiple LED's are wired together in rows or columns to allow an efficient way of enabling and disabling LEDS, minimising the number of pins required. For example, an 8x8 LED dot matrix would require 64 pins, 1 per LED. But as all the LEDS are wired together in rows and columns, it only requires 16 pins. Because we will then be using less pins than theoretically needed, each LED can only be addressed/targeted by using its row and column number. Every LED in a dot matrix is assigned by a row and column number (as shown below).

NSN: 0129972568



If you wanted to turn the 3rd led across from the left, in the 4th row from the top, we would select the row id (which in this case is R4 and column id C3). We would select the column id first, disabling every other column by disabling their ground. As the 3rd column is now only active, we will turn the LED in the 4th of this column by applying a voltage to this row. This will enable the led in the 3rd column and 4th row.

I will be using a dot matrix in my project is it is the simplest display that I can use that my intended users might be familiar with. As my targeted user demographic is quite young, this would be the perfect display to use that shouldn't be too hard to learn or understand.

Interrupts
Interrupts is a function that when triggered 'interrupts' the current process and executes code designed to react. The point of an interrupt is to make sure that the processor responds to important events. Interrupts is a good timing function that is like delay but is better depending on what you're doing. The difference is that a delay delays a particular point in the program, where as an interrupt will execute a set amount of code at a particular point of the code. Interrupts also make sure that the processor responds quickly to events. I will use interrupts in my project to execute an amount of code when a condition is met.

An example of this is below:

NSN: 0129972568

```
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
volatile int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // Attach an interrupt to the ISR vector
  attachInterrupt(0, pin_ISR, CHANGE);
}

void loop() {
  // Nothing here!
}

void pin_ISR() {
  buttonState = digitalRead(buttonPin);
  digitalWrite(ledPin, buttonState);
}
```

Instead of setting LEDS to high, we can use ISR (Interrupt Service Routine) to set the LED high. An ISR is designed to run quickly and handle the interrupt to allow the processor to get back to main program. The 'Interrupt Vector' determines which pin causes the interrupt. An interrupt pin references where in the Arduino memory it must look to see if an interrupt occurs. The function name 'interrupt service routine' is the set code that is execute when the interrupt condition is met.

The example above is an example of a button interrupt. In this code, when the button is pressed, an ISR is triggered and a block of code is executing, as the condition is met (button press).

Flags
Flags are a state variable that signalises something. For example, a flag could be 'yes' or 'no', 'true' or 'false' which enables a part of your code when the condition is changed or met. Flags are used as a Boolean value. These are very useful if you want to delay your code from carrying on for a period. For example, if you wanted to blink a light but have the light on for 2 seconds then off 2 seconds you can easily achieve this with delays.

In my project I will be using flags to determine the mode that my project is in. When a mode is chosen (e.g. dice, Tetris, snake etc) it will change the state variable of a flag. While this happens, my code will be constantly looking for a change in the flag, to determine what set of code to activate. Each mode will change the state variable of the flag differently, allowing the code to easily figure out when the mode has been changed. A flag variable will determine the behaviour of my projector.

Analogue to digital and digital to analogue conversion
Analogue to digital and digital to analogue conversion is the conversion of an analogue signal into a digital signal (vice versa). Microcontrollers detect and respond to binary values. 0 volts is represented as the binary value of 0, and 5v is

represented as the binary value of 1. Between these values there is a threshold value that represents a value between the voltages. These values are called digital values. Although, as a microcontroller can receive voltages that are in between (e.g. 2.8v) and voltages greater than 5v, one of these values are called an analogue voltage/value. Arduino ADC is the pins (A0 -> A5) on an Arduino. These are the only pins on an Arduino microcontroller that can read an analogue input. A microcontroller would then need to convert the voltage into a digital value, for it to understand, as microcontrollers only respond to digital values.

For an Arduino microcontroller, we can read an analogue value by loop serial printing the analogue () value from an analogue pin (A1 -> A5). This will continuously print a value between 0 -1023 as the Arduino ADC is 10 bits. If we were to add a thermistor component to the circuit, then it would constantly read a value that reflects on its current temperature.

I will be using Arduino ADC in my project get the digital values of an accelerometer. I will get a range of values from an accelerometer that represent an enough shake of the device, which will activate the dice function of my device. By having analogue values as a range, I can get accurate shakes each time.

Structuring complex programs logically
Structuring your code is very important when coding. By coding 'neatly' and efficiently it allows other people to understand your code from their perspective. I will be structuring my code logically so that I can easily make changes in the future where I might forget what I have coded. I will also do this so that other people can understand my code and makes efficient changes.

Here are some examples of structuring programs logically:

- Declare variables at the top of your code - I will do this so that they are easy to find and change
- Give variables appropriate names - So that I can easily understand and revisit my code after a period (in which I have forgotten).
- Declare Arduino pins as variables - So I can make efficient and effective easy changes while also having the best understanding of variables.
- Use white spaces when appropriate - I will do this for better separation of code leading to easier understanding, although I will use this appropriately when necessarily.
- Use functions - I will do this for easy separation of code and activating it when required.
- Group functions together at top of page - I will do this so that they are easy to find and change

Conclusion
I will use all the complex concepts (listed from above) appropriately and effectively in my project. If I apply my software and hardware concepts in an effective manner, my project will function as intended.

NSN: 0129972568

Bibliography
https://www.allaboutcircuits.com/technical-articles/using-interrupts-on-arduino/
https://www.electronics-notes.com/articles/analogue_circuits/operational-amplifier-op-amp/non-inverting-amplifier.php
http://www.learningaboutelectronics.com/Articles/What-is-an-ideal-op-amp.php
https://chrisgammell.com/how-does-an-op-amp-work-part-1/
https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
https://en.wikipedia.org/wiki/Dot-matrix_display
https://hobbycomponents.com/opto-electronics/190-max7219-serial-dot-matrix-display-module
https://1sheeld.com/accelerometer-module/
http://embedded-lab.com/blog/lab-12-basics-of-led-dot-matrix-display/
https://eeeproject.com/touch-sensor/
http://www.learningaboutelectronics.com/Articles/Low-pass-filter.php