WACL Electronics Course

William Burbidge

August 11, 2022

Contents

This document was produced using LATEX and a range of linked packages. All images have been drawn by William Burbidge using InkScape, other than those that reference software such as Falstad or Fritzing, plus screenshots of other programs.

The document has been designed to best be read as a PDF. It uses a range of links to both internal information and external websites. Terms found in the glossary and links are in bold, so you know that you can click on them to lead to their definition or the website.

Remember that there are risks when dealing with electronics. Ensure to take necessary precautions when using this manual as a guide.

0.1 Introduction

Understanding the electronics behind processes or **Sensor**s used in the field can help to optimise time, reduce costs and find tailored solutions to problems within research. Whether you are repairing equipment, designing a system to automate data measurement or need help explaining why data differs from what's expected.

This course will take you through the basics while linking it to examples important for chemistry and atmospheric research. By the end of the course, you should understand basic electronics theory and practice, have a knowledge of the terminology used, and should be able to build a basic system with a **Sensor** and **Control Loop**.

The course touches on other useful areas to be aware of when dealing with electronics, such as enclosures, environmental factors, sourcing parts and safety. We will start with the basics of electronics. There are multiple ways to represent this. Starting with the physics, using analogies where helpful.

0.2 Electronics Fundamentals

Electronics involves a flow of electrons, for a transfer of energy over time (i.e. power). We represent this through **Voltage** and **Current**, seen as key units in electronics. If represented as letters within a mail network, **Voltage** is the number of letters each mail person carries, where **Current** is the rate of new batch deliveries. A component or drop-off point along the network has **Resistance**, which is the ratio of letters they pick up from each mail person.

From a physics standpoint, electronics work through transferring electrical energy, which goes through a conversion process in components. A **Resistance** causes a heat conversion/heat loss. There is also **Capacitance** and **Inductance**, discussed later on, due to conversion to electric and magnetic potential.

Focusing just on **Resistance** and power, for now, we can represent this as several key physics equations, important for physics-based electronic theory and practice.

$$V = IR, P = IV, P = I^2R, P = E/T$$

It is possible to derive $P = I^2R$ by rearranging and putting $V = IR$ into $P = IV$. We can see this below:

$$V = IR \Longleftrightarrow P = IV$$

$$P = I(IR)$$

$$P = I^2R$$

Voltage is represented with V, with the unit **Volt Current** is represented with I, with the unit **Ampere** (Amp). **Resistance** is represented with R, with the unit Ohm (with the Greek letter Ω). Power is represented with P, with the unit **Watt**. Time is represented with T in seconds.

Quiz

1. What does **Resistance** demonstrate?

a) The rate of flow of electrons.

b) The energy in each "packet" or electron.

c) Conversion of electrical to thermal energy/ heat.

2. Demonstrate how you would go from the equation $P = I^2R$ to $P = IV$.

3. A circuit has a 3V battery source, with 0.5A of **Current** flow. What is the **Resistance** seen in the circuit, and the overall power draw?

4. Use an analogy to explain **Current**, **Voltage**, and **Resistance**.

0.2.1 Resistors

What are Resistors?

Resistors are key components in electronics, only requiring an understanding of equations already discussed.

Resistors are components which are rated to have a specific **Resistance**. Therefore, depending on the circuit, losses due to them will vary (depending on **Current** and **Voltage**). Useful for when parts need a specific **Voltage** close to the input **Voltage**, or as a way of limiting **Current** flow.

This is an example of two main **Resistor** representations.

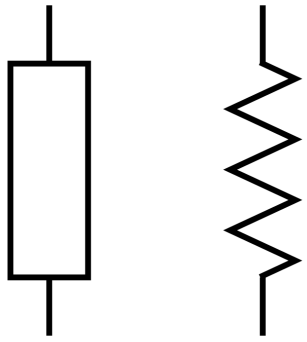


Figure 1: **Resistor** representions, European and North American representation

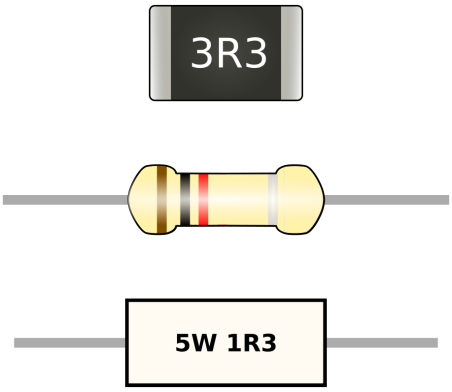


Figure 2: Common **Resistor** Packages

As can be seen from the middle **Resistor** image, bands are frequently used on through hole **Resistor** s, so that from any angle their **Resistance** can be read. Below is a reference guide for this, which is worthwhile remembering. Although, there are plenty of online calculators, where you can enter the bands to see the **Resistance**. Alternatively, a multimeter should provide a quick and accurate **Resistance** measurement, which is discussed later in the document.

Example circuits etc

This is a basic circuit, with wires, a **Resistor** and a battery, to demonstrate V=IR. **Series Link**

In the specific example, we have a **Resistor** of 1 Ohms and a **Voltage** of 5V. Therefore, using $V = IR$, we can rearrange to find a **Current** of $I = 5/1 = 5A$. Then we can find a power of $P = IV = 5 * 5 = 25W$. The simulation shows each of these. Feel free to play around with it and alter values. Instructions for using the simulation program are found (here).

There aren't many example circuits using the above knowledge alone. One example would be a heater project. Given you know the available input **Voltage**, and the needed thermal load, you can calculate the required heating element **Resistance**.

There may be the case where a constant heat power output is required. A **Resistor** is purely resistive, so all the electrical power it uses should be dissipated as heat. Lets say there is a supply of 240V (to simply represent mains **Voltage**), and a 2000W supply of heat is required (of a similar magnitude to an oven). Therefore, using $V = IR$ and $P = IV$. Starting with $P = IV$, we can find a **Current** of the

below:

$$I = \frac{P}{V}, I = \frac{2000}{240}, I = \frac{25}{3}A$$

We can now use this in:

$$V = IR, R = \frac{V}{I}, R = \frac{240}{\frac{25}{3}}, R = 28.8\Omega$$

In reality, most heater systems will have a **Control Loop**, with a focus to achieve and then sustain a given temperature, so there would be an element of sensing and turning the heater on and off, which we will get to later in the document.

Quiz

1. True or false, power draw is always constant for a given **Resistor**, whatever circuit it is in?

2. True or false, **Resistor**s are available in a wide range of **Resistance** values, but multiple can be used together to achieve more specific values?

3. True or false, **Resistor**s convert electrical energy to thermal energy?

0.2.2 Potentiometers

A **Potentiometer** is a variable **Resistor** acting as a **Potential Divider**, using a dial to alter its output **Resistance**. There are many cases where this is helpful, like altering speaker volume, or where **Voltage** needs aren't clear or certain. They typically have 3 pins, for the input, ground and output. The input and output together act as a **Resistor**, with a value depending on how far the dial is turned.

Digital Potentiometers are an example of an integrated circuit, which are programmed to act as a set **Resistance**, using steps, meaning the same resolution isn't usually attainable.

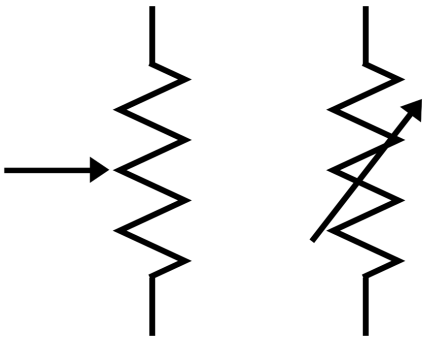


Figure 3: **Potentiometer** on the left, Variable **Resistor** on the right

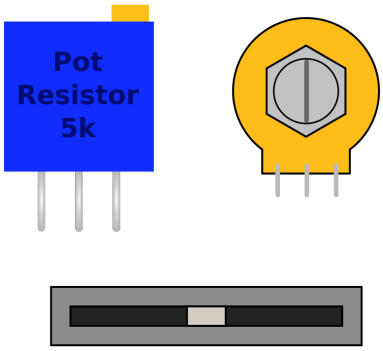


Figure 4: Common **Potentiometer** Packages

0.2.3 Series and Parallel

You can wire a circuit with **Parallel** and **Series** elements. A **Series** circuit has a single loop, with components connected one after another, sharing a single **Voltage**. A **Parallel** circuit can have more than one loop branching out. Each loop has a **Voltage** equivalent to the input, with the **Current** split between them, based on their **Resistance**.

Therefore, the **Voltage** in a **Series** loop will be distributed based on the **Voltage** ratios. For example, there is a 10Ω and 5Ω **Resistor**, with a 9V supply. There would be a **Voltage** drop of 6V through the 10Ω **Resistor** and a 3V drop through the 5Ω **Resistor**. With a total **Resistance** of 15Ω, we can calculate the **Current** to be $V = IR$, $I = \frac{V}{R}$, $I = \frac{9}{15} = 0.6A$.

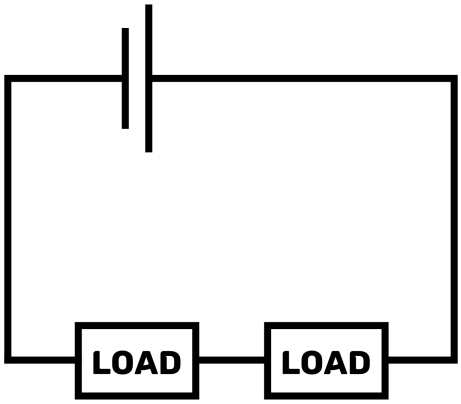


Figure 5: **Series** base example circuit

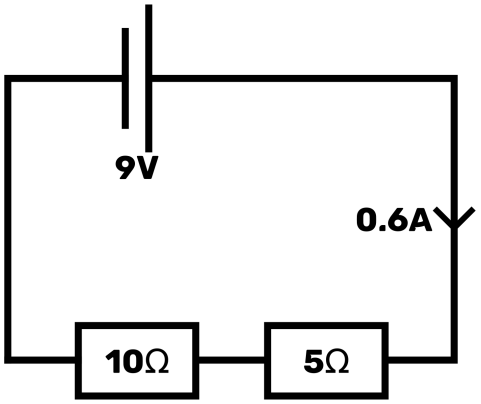


Figure 6: **Series** example circuit 1

A **Parallel** circuit has equal **Voltage** across each path, with the **Current** being split. Because of this, **Resistor**s in **Parallel** have a total **Resistance** lower than the lowest **Resistance** path. We can calculate this using the below equation:

$$\frac{1}{R_{total}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \cdots$$

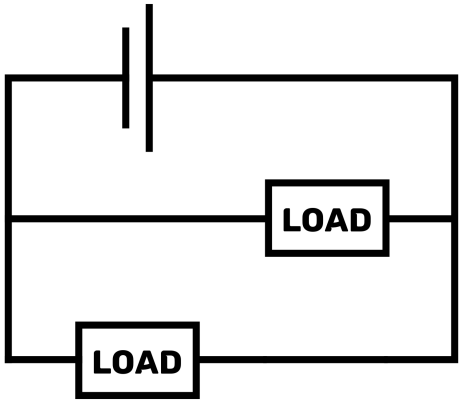


Figure 7: **Parallel** base example circuit

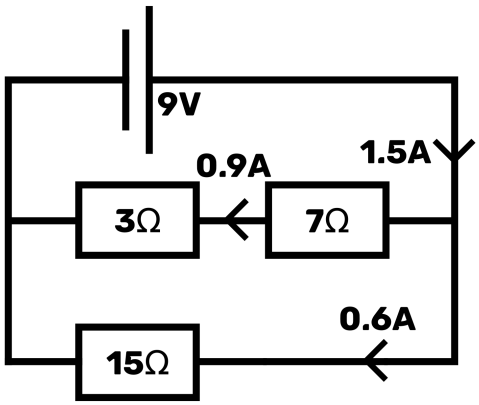


Figure 8: **Parallel** example circuit 1

Therefore, an example with the following circuit with 2 **Resistor**s of 3Ω and 7Ω, in **Parallel** with a single **Resistor** of 15Ω. You first add the **Series Resistor**s, therefore an equivalent **Resistor** R_1 is 10Ω and R_2 is 15Ω. Putting it into the equation gives us the below:

$$\frac{1}{R_{total}} = \frac{1}{10} + \frac{1}{15}$$

$$\frac{1}{R_{total}} = \frac{1}{6}$$

$$R_{total} = 6\Omega$$

If we were to think again of the mail network analogy, a **Series** circuit has the mailpeople take multiple stops on their journey, handing a certain amount of letters to each, depending on their need/**Resistance**. With a **Parallel** circuit, some mailpeople will focus on different paths, so will still have the same total

parcels for their path (the **Voltage** stays the same for each path). The **Current** decreases, since the mailpeople split to go down different routes.

Example circuits etc

A basic example would be a basic circuit to "divide" a **Voltage**. **Series Link**

Quiz

1. Which of the following statements are true?

a) **Current** is equal down each path from a junction.

b) **Current** splits at a path junction, based on each paths **Resistance**.

c) **Voltage** is equal down each path from a junction.

d) **Voltage** splits at a path junction, based on each path's **Resistance**.

2. What is the **Voltage** drop of each component? Also, calculate the power of the circuit.

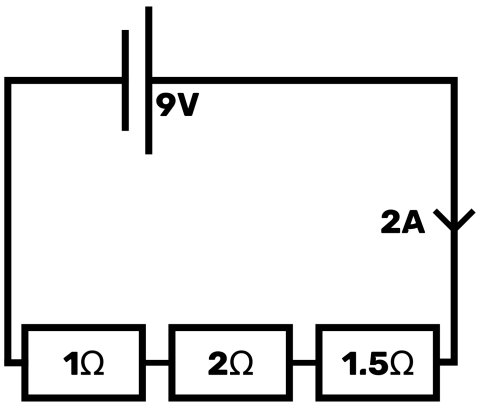


Figure 9: **Series** example circuit 2

3. What is the equivalent **Resistance** of a 20Ω, 100Ω and 42Ω component, all in **Parallel**?

4. True or false, the power draw through two 20 Ω **Resistor**s is higher than through a single 10 ohm **Resistor**?

0.2.4 Potential Dividers

Potential Dividers use knowledge of both **Parallel** circuits and **Resistance**. As the name suggests, they help to divide **Voltage** potential based on the use of **Series Resistor**s. As stated before, within a **Series** path, the **Voltage** will be divided between the components based on their **Resistance**.

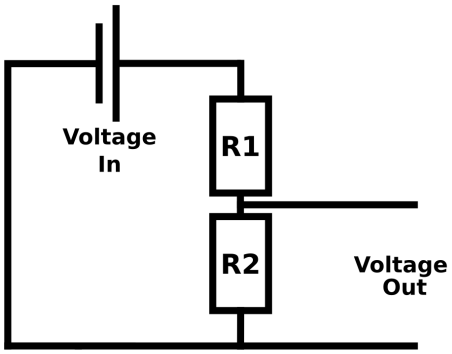


Figure 10: **Potential Divider** base example circuit

Therefore, if there was an input **Voltage** of 5V, and R1 has a **Resistance** of 10Ω and R2 has a **Resistance** of 3Ω, then the **Voltage** seen through the **Potential Divider**'s output is the below:

$$V_{out} = \frac{(R_1 + R_2) - R_2}{R_1 + R_2} * V_{in} = \frac{10}{13} * 5 = \frac{50}{13} V$$

0.2.5 AC vs DC

AC stands for alternating **Current**, whereas **DC** stands for direct **Current**. This is shown on a graph with the direction of flow on one axis and magnitude of **Voltage** on the other. **DC** has a constant magnitude and direction of flow. This compares to a sinusoidal **Voltage** magnitude with **AC**, where the flow of **Current** varies in direction. **AC** is typically given as an **RMS Voltage** and a frequency. The frequency is the number of **Period**s that happen per second, typically given in Hz, where the **Voltage** is an average value, taking the root mean square. Root mean square (**RMS**) represents the **DC Voltage** that has the same power/ heating effect as the **AC** circuit. This is seen with power from the **Grid**, with **RMS Voltage**s of 230V and frequency of 50Hz in the UK. Being in **AC** allows for high **Voltage** conversion through long distance cables using **Transformer**s, to reduce thermal losses due to heat from high **Current**. This is particularly important, as using $P = I^2R$, we can see that power

compared to **Current** is a quadratic, so with a fixed cable **Resistance**, the higher the **Current**, the larger the increase of heat loss.

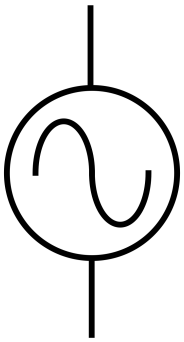


Figure 11: **AC** Source circuit icon

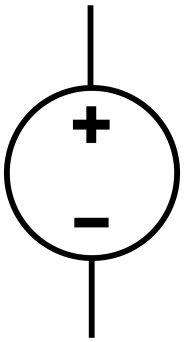


Figure 12: **DC** Source circuit icon

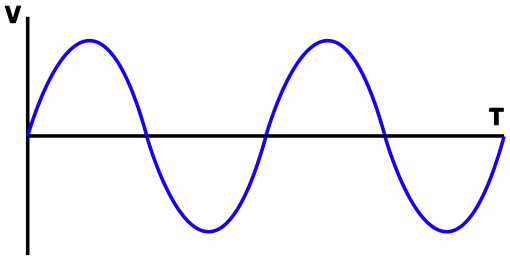


Figure 13: **AC** Wave example graph

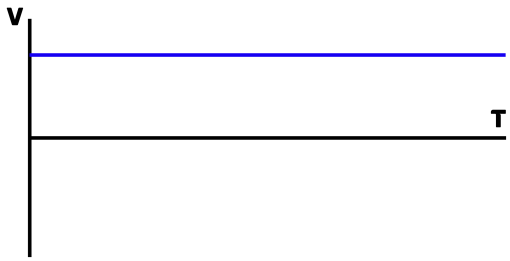


Figure 14: **DC** Wave example graph

Digital electronics are within the **DC** realm, but many devices still require **AC**. It is the standard for energy generation and for how it is sent across the **Grid**. **AC** is widely used for **Motor**s, and where large **Voltage** transformations are required, such as microwaves. **Semiconductor** electronics primarily work within **DC**.

The course will touch upon ways to convert between **AC-DC** and the benefits of each for different applications.

Quiz

1. What is the frequency of **AC** Electricity from the **Grid** in the UK?
2. What does **RMS** stand for?
3. Why is **AC** used within the **Grid**?

0.2.6 Capacitors

What are Capacitors?

Capacitors are another key component. They store energy as Electric Potential, with their **Voltage** increasing towards the input **Voltage**, as they charge. Therefore, the charge and discharge curve is nonlinear, more similar to a natural log curve.

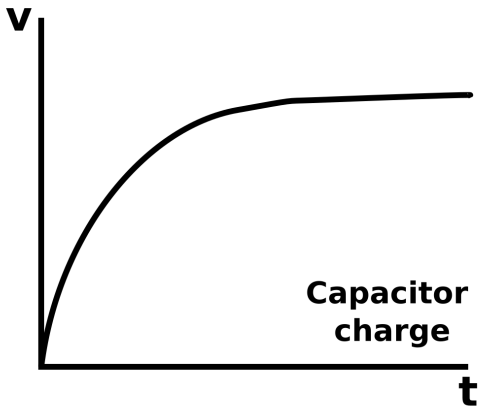


Figure 15: **Capacitor** charge **Voltage** graph

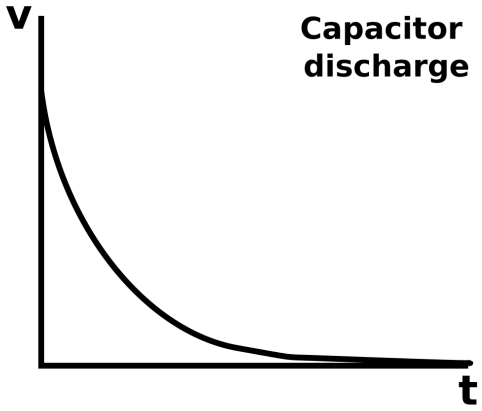


Figure 16: **Capacitor** discharge **Voltage** graph

The unit for **Capacitance** is **Farad**, with typical values depending on **Capacitor** type, but usually in the uF units or below.

A range of physics based equations will be touched upon now.

Capacitors have a charge relating to their input **Voltage** with the following basic equation: $q = CV$. C is **Capacitance**, q is charge, and V is **Voltage**. There is also the formula for energy stored within a **Capacitor**, with the following equation: $E = 1/2 * C * V^2$, where E is energy in **Joules**.

A **Resistor** is usually placed in **Series** with a **Capacitor** to limit the **Capacitor** inlet **Current**. As it charges, the ratio between its **Voltage** and the supply **Voltage** decreases, along with the charge **Current**, in a curve of reverse direction to the **Voltage**.

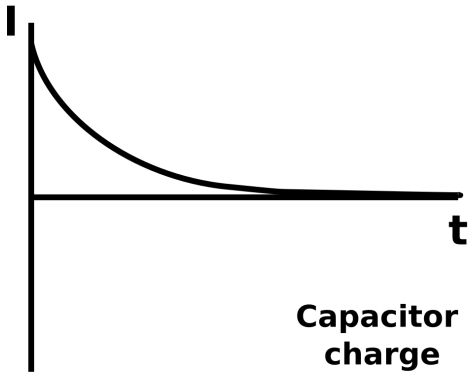


Figure 17: **Capacitor** charge **Current** graph

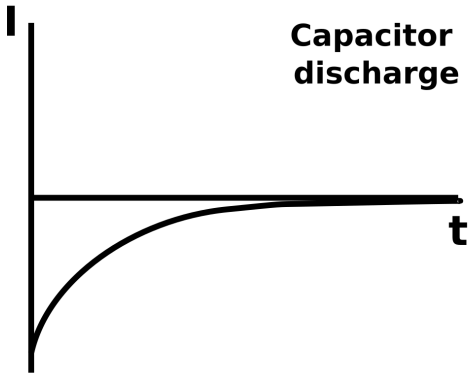


Figure 18: **Capacitor** discharge **Current** graph

One of the built-in simulation circuits from Falstad is helpful for showcasing the charge and discharge characteristics of a **Capacitor**, by flicking a switch. **Series Link**

You should be aware of what happens with **Series** and **Parallel Capacitor**s. The **Capacitance** of **Capacitor**s add when in **Parallel**, where in **Series**, it is like **Resistor**s in **Parallel**. You can think of it having inverse rules to the **Resistor**. Therefore, **Capacitor**s are often used in **Parallel** but less so in **Series**, unless a higher **Voltage** is required than their individual rated value. Shown by the equations below:

Series Capacitors-

$$C_{total} = \frac{1}{\frac{1}{C_1} + \frac{1}{C_2} + \cdots}$$

Parallel Capacitors-

$$C_{total} = C_1 + C_2 + \cdots$$

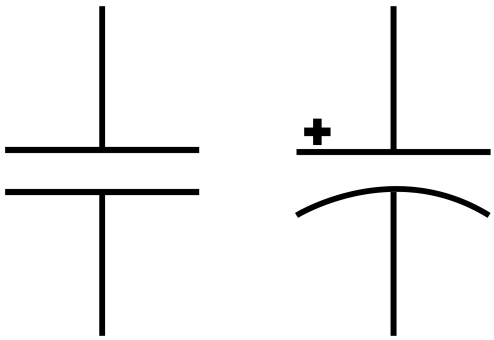


Figure 19: Bipolar **Capacitor** on left, Polar **Capacitor** on right

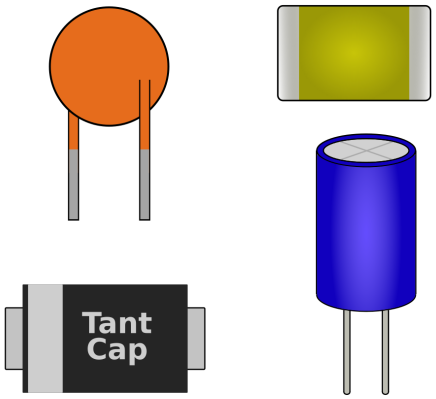


Figure 20: Common **Capacitor** Packages

Example circuits etc

Quiz

1. How do **Capacitor**s store energy?

a) In the magnetic field

b) In the electric field

c) As thermal energy

d) They do not store energy

2. True or false, a **Resistor** is typically used in **Series** with a **Capacitor** to limit **Current** flow through the **Capacitor**.

3. True or false, 1 **Farad Capacitor**s are commonly available.

4. Design a circuit for charging a **Capacitor** at a maximum **Current** of 1A, assuming a source **Voltage** of 3V.

0.2.7 Inductors

What are Inductors?

Inductors are another key passive energy storage component. These store energy within the magnetic field, and have a range of uses, often in power electronics.

The unit for **Inductance** (L) is **Henry**, with typical values depending on **Inductor** type, but usually in the uH or mH units or below.

A range of physics based equations are used to show **Inductance**.

There is the following basic equation: $L = \frac{\Phi(i)}{i}$. There is also the energy formula for calculating the energy stored within an **Inductor**, with the following basic equation: $E = 1/2 * L * I^2$.

It is also useful to be aware of the rules for **Series** and **Parallel Inductor**s. It is like the rules for **Resistor**s, although this time being **Inductance**.

Series Inductors-

$$L_{total} = L_1 + L_2 + \cdots$$

Parallel Inductors-

$$L_{total} = \frac{1}{\frac{1}{L_1} + \frac{1}{L_2}}$$

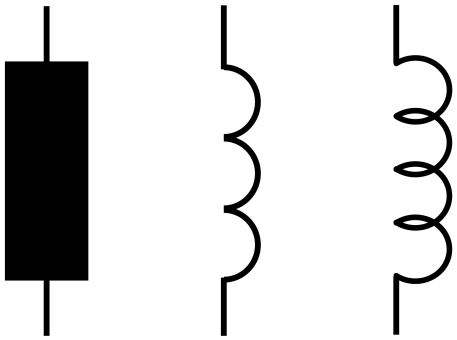


Figure 21: **Inductor** representations

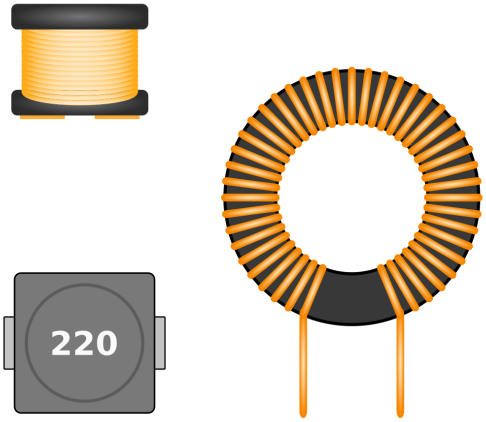


Figure 22: Common **Inductor** Packages

One of the built-in simulation circuits from Falstad is helpful for showcasing the charge and discharge characteristics of an **Inductor**, by flicking a switch. **Series Link**

Example circuits etc

Quiz

1. What units is **Inductance** given in?

0.2.8 Diodes

Diodes are widely used in electronics. The basic principle of a **Diode** is a device/component which limits how **Current** can flow within a circuit, such as restricting **Current** flow to one direction. This makes them beneficial for a range of circuits, such as protection circuits, preventing unwanted reverse **Current** flow, which could damage other components or cause unwanted side effects. They are a simpler examples of a **Semiconductor**, where usually silicon is **Doped** with other atoms to cause a surplus or shortage of electrons, changing the polarity of elements of the material, affecting how **Current** flows through the **Diode**.

There are also Light Emitting **Diode**s (**LED**s), which work through similar principles, but emit light of a certain spectrum depending upon **LED** selection. There is a **Voltage** drop associated with this, but with considerably lower heat output, compared to filament bulbs, which heat filament until it glows. Therefore

There is a common formula for calculating the **Resistor** needed for an **LED**, given you know its **Voltage** drop and the **Current** it runs at. This is the following:

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

Where V_s is the supply **Voltage**, V_f is the forward **Voltage** (**Voltage** drop of the **LED**) and I_f is the forward **Current**.

Different colour **LED**s work at different **Voltage**s, so **Resistor**s are commonly used to drop the input **Voltage** to a level suitable for the **LED**. The **LED**s also help to limit **Current** through the **LED**. This **Voltage** usually is between 1.8-3.3v, and is because of how the **Semiconductor** is **Doped**, as different dopants will cause different **Wavelength**s of light to be emitted.

It is important to consider that **Diode**s have an inherent **Voltage** drop, so this needs to be considered, especially when using them with low-**Voltage** circuits, as a typical **Diode** can have **Voltage** drops between around 0.4-0.7v.

There are many subcategories of **Diode**s. The most common type mentioned above is known as a junction **Diode**. **Zener Diode**s allow reverse **Current** after a certain threshold **Voltage** is met. A **Schottky Diode** has low **Voltage** drop and fast switching. There are also **DIAC**s and **TRIAC**s for **AC**, allowing some **Current** flow in both directions.

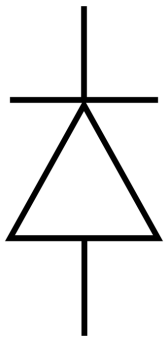


Figure 23: **Diode** representation

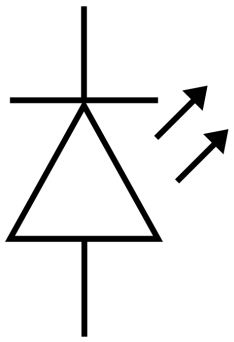


Figure 24: Light Emitting **Diode** representation

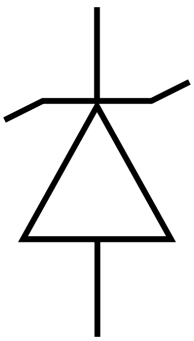


Figure 25: Zener **Diode** representation

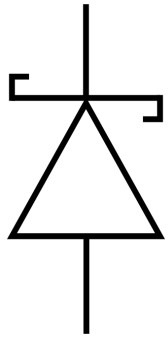


Figure 26: Schottky **Diode** representation

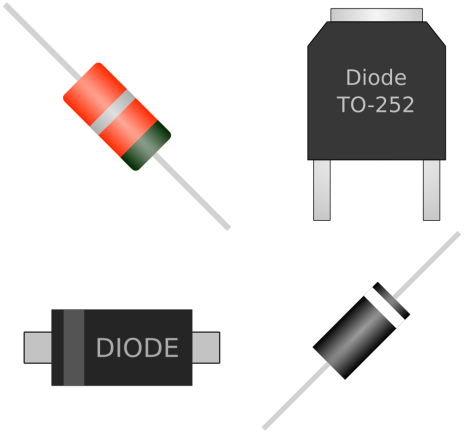


Figure 27: Common **Diode** Packages

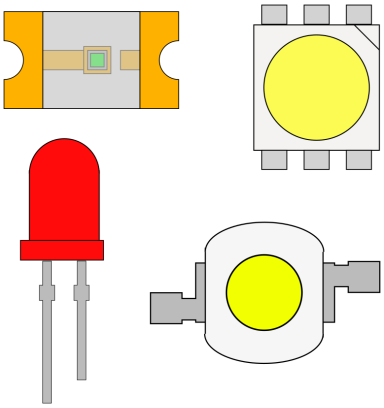


Figure 28: Common **LED** Packages

Quiz

1. Name 4 examples of **Diode**s, and what they do?

2. True or false, **Diode**s always restrict **Current** flow to one direction?

3. You are building a low **Voltage** circuit, and need to limit **Current** flow to one direction, but with as low losses as possible. What **Diode** would you recommend?

4. True or false, **LED**s can be found in a range of colours, each with differing **Voltage** drops?

5. Design a **LED** circuit, with an input **Voltage** of 3V and the **LED** has a **Voltage** drop of 1.8V running at its recommended **Current** of 25mA. What is the total power consumption of this circuit?

0.2.9 Transformers

What are Transformers?

Transformers use the fact that a **Current** carrying coil of wire will generate a magnetic field, which induces a **Current** in a coil to allow for **Voltage**/ **Current** conversion. Think of it as two **Inductor**s, **Coupled** through a single **Core**. This is typically a **Ferritic Core** to provide a magnetic path to reduce losses. These are often laminated to limit eddy **Current**s, which are loops of **Current** within the **Core**, induced from the magnetic field. **Lamination** increases the **Core Resistance**, reducing the eddy **Current**.

A lot of **Transformer** design is out of the courses scope. We will assume that the **Transformer** is ideal, although in reality, there are losses, leading to efficiencies of upwards of 95%. This depends on how well suited a **Transformer** is for a design.

Due to **Current** only being induced under a changing magnetic field, **Transformer**s only work with an **AC** input.

This is the basic set of equations for **Transformer**s.

$$\frac{V_p}{V_s} = \frac{I_s}{I_p} = \frac{N_p}{N_s}$$

Where p represents the input/ primary coil and s represents the output/ secondary coil. V represents

Voltage, I represents **Current** and N represents the number of turns/ coils.

There is an inverse relationship between **Voltage** and **Current** when there are different coil turns.

There are 1:1 **Transformer**s, where no **Voltage** or **Current** conversion takes place, but it isolates the input from the output, so is a safety feature often used in products. These are known as isolation **Transformer**s.

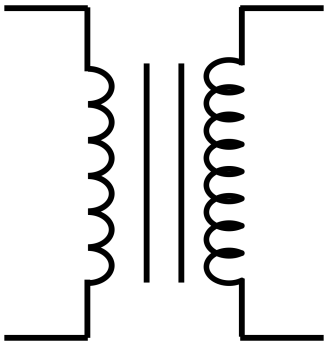


Figure 29: Transformer representation

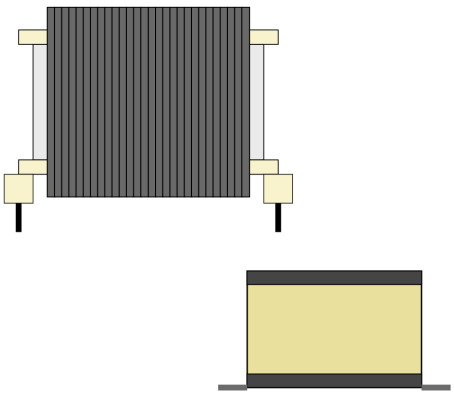


Figure 30: Example Transformer Packages

Quiz

1. Why are **Transformer** coils often laminated?

2. A **Transformer** with a 240V **RMS** input with a peak **Current** capability of 0.1A, with 100 turns on its primary side, needs to supply a 12V **RMS** circuit. How many secondary turns are required, and

what is the peak **Current** that could be supplied to the output circuit?

3. Why are isolation **Transformer**s used?

0.2.10 Analogue vs Digital

It is important to be aware of differences between **Analogue** and **Digital**. **Analogue** refers to a wave, with the fundamental characteristics of **Wavelength**, frequency and magnitude. This value can vary considerably within a **Period**, which is particularly useful for some applications.

Below are a couple of examples of Analogue graphs.

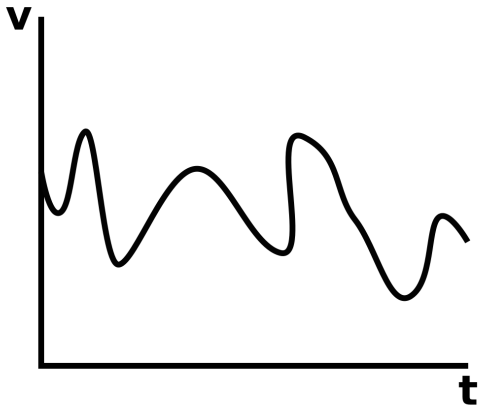


Figure 31: Example **DC** Analogue Waveform

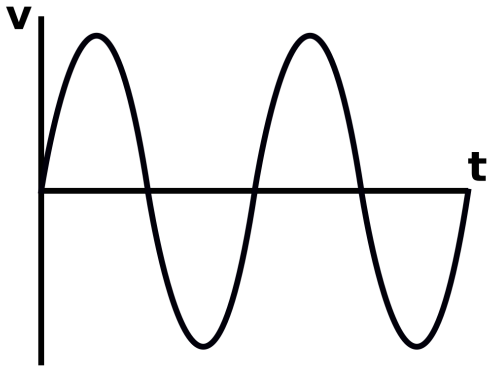


Figure 32: Example **AC** Analogue Waveform

On the contrary, **Digital** is represented with either 1 or 0 (off or on). As already touched upon, the entire field of modern **Digital** electronics is based around **Semiconductor**s, which form switches.

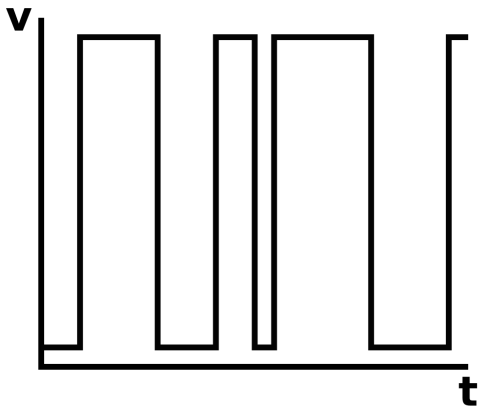


Figure 33: Example Digital Waveform

Signals can be sent through both methods, with a range of benefits and negatives. A lot of data can be shown in a single **Analogue** signal, although it can be difficult to accurately read. Whereas, **DC** deals with two distinct boundaries, so although many more "pulses" are required to output the same amount of data, it is more reliable and easier to use/ read within logic based processing/ **Boolean Algebra**, which is touched upon later on in the course.

Sensors are often read using **Analogue** methods, where there is an induced or dropped **Voltage** which can vary in magnitude. This would need to be converted to a **Digital** value for **Microcontroller** use. A combination of **Digital** 1/0 can be **Binary**, which means a **Voltage** value can be represented as a **Binary** combination of 1s and 0s. An A**DC** is an IC which does this conversion.

Quiz

1. For a wire communicating over a long distance, would you recommend that an **Analogue** or **Digital** signal be used?

2. Can equivalent numbers be represented with **Analogue** and **Digital**, along with some processing/ data conversion?

0.2.11 Mechanical switches

Switches are used as input devices, where depending on being open or closed, will allow power transmission or not, by opening or closing a circuit. These can be split into many types, such as typically open

switches unless pressed, typically closed switches unless pressed, or those which switch between open and closed when pressed, and switches which you can switch to close several sets of pins/ connections.

This section will quickly talk through a few different switches/ buttons, their names and uses.

There are different switches that fit into different classifications, which will briefly be discussed.

Single Pole Single Throw (SPST) switches are ones which connect or break a connection between two single wires/ points/ terminals.

Single Pole Double Throw has a single input but two outputs which can be switched between.

There is then also Double Pole Single Throw, which is a combination of two SPSTs and Double Pole Double Throw, a combination of two SPDTs.

Now onto the different types of switch. They can mainly be split into several types, being the below:

Push button- A typical button, some of which will stay closed/ pressed until pressed again, when others such as momentary push switches will only be closed as they are pressed.

Slide switches are flicked into set position, which can be two positions or multiple.

Toggle is a slight adaptation of slide switches, flicking a centred switch from one angle to another, otherwise working through similar means.

Rotary switches can twist to connect a pin with a range of other pins depending where it is twisted to.

You may have a range of switches, such as a few slide switches packaged together in **DIP** format.

0.2.12 Digital switches/ Transistors

Switches are widely used within electronics and are what led to the field of **Digital** electronics. **Microcontroller**s may use millions of switches on a single silicon **Die**, where discrete switches are mainly used for power electronics.

Switches are important to have control of a circuit. These are discussed in greater depth later, but they usually have an externally triggered **Gate**, which closes a circuit. Think of a switch as a tap that allows or restricts water flow. The vast majority of these are **Semiconductor** based, due to the low losses involved, although they are limited to **DC** function.

This document quickly talks over the main types in detail, so you can understand why each are used, and to help understand surrounding circuitry.

There are **Current** controlled switches and **Voltage** controlled. **BJT**s are an example of a **Current** controlled switch, where a flow between the **Gate** pin and the emitter allows for a higher **Current** to flow between the emitter to the collector. There is a **Voltage** drop, which can mean losses are fairly significant at lower **Voltage**s. They are easy to use in a circuit after calculating values.

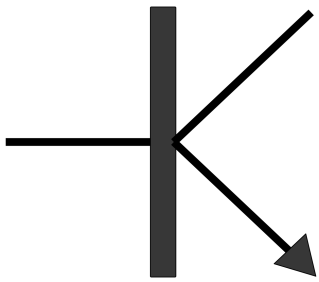


Figure 34: Bipolar Junction Transistor (BJT) representation

Then there are **Voltage** controlled switches, the main example of this being **MOSFET**s. **Current** can flow after the **Voltage** is higher than a threshold is reached in its **Gate**. They are more complex to use, since the **Gate** acts as a **Capacitor**, so has to be charged, so may not function well without a properly considered **Gate** charge circuit. Losses are related to the **MOSFET**'s internal **Resistance**, which can get very low, so they are typically fairly efficient.

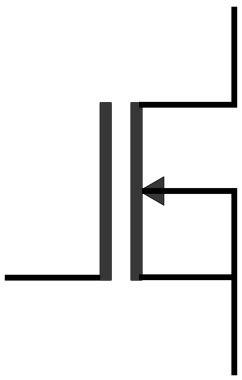


Figure 35: Metal-Oxide-**Semiconductor** Field Effect Transistor (MOSFET) representation

There are other switches, mainly for high power electronics, which don't require as much depth.

Relays are an alternate form of switch, which uses an **Electromagnet** to cause a switch to open or close. Most switches work with **DC**, where relays are useful for **AC** applications. They are another form of switch that provides electrical isolation between parts, as it is an **Electromagnet** which pushes the switch closed.

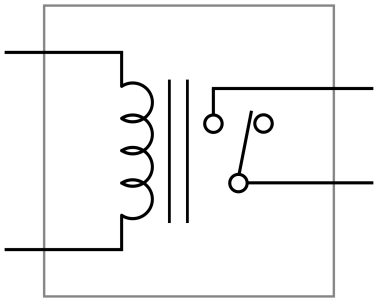


Figure 36: Relay representation

Darlington **Transistor**s are also commonly known as a **Darlington Pair**, made of a couple **BJT Transistor**s, allowing higher levels of **Current** amplification/ high **Current** gain.

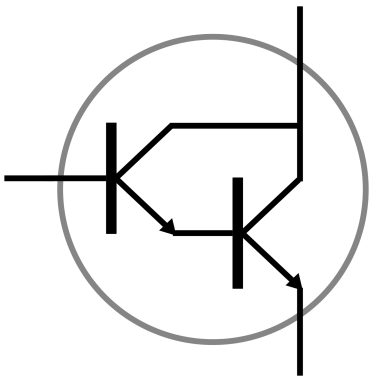


Figure 37: Darlington Transistor representation

You can get arrays of **Transistor**s, such as Darlington Arrays, which is a chip with multiple Darlington **Transistor**s with common emitters. These are useful for high power or inductive loads being controlled by a **Microcontroller**.

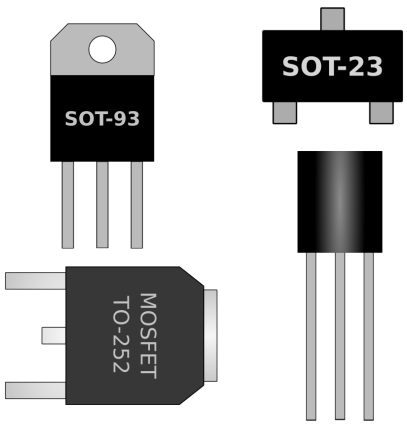


Figure 38: Common Transistor Packages

Quiz

1. What type of **Digital** switch could you use for turning an **AC Motor** on and off?

2. If you needed a simple **Transistor** circuit to turn a high power **LED** on and off from a **Microcontroller**, what type of **Transistor** would you use and why?

3. What is the purpose of a **Transistor Gate**? Feel free to use an analogy to help you explain?

0.2.13 Sensors

What are Sensors?

Sensors are likely to be a key component or **Module** you use in your research, so understanding how they work and your options is important. Key aspects of **Sensor** theory are discussed here, with more in the further reading section.

This is split into several sections, covering some of the major **Sensor**s, with more detail on **Sensor**s likely to be used within chemistry related projects.

Sensors typically use **Resistance**, **Capacitance** or **Inductance** to take a vast range of measurements, often using a MicroElectroMechanical system (**MEMS**) to allow for this.

Temperature sensing can be done with a range of different methods, and is likely to be important. The

most common **Sensor**s are **Analogue** ones, where the components **Resistance** changes along with a temperature change, although there are **Digital Sensor**s for this too. There are 4 major categories of temperature **Sensor**, which we will quickly touch upon each. There are **Thermocouple**s, **RTD**s, **Thermistor**s and specific **Digital** ICs.

Thermistors are common, and are divided into NTC and PTC, with NTC more common. NTCs have a decreasing **Resistance** compared to temperature, and will have a **Resistance** rating for 25°C typically along with a value for how much their **Resistance** changes per degree. Therefore, utilising a circuit to measure this **Resistance** allows for a relatively accurate temperature to be read. In comparison, PTCs have a positive temperature coefficient.

RTDs are similar in their operation to **Thermistor**s, but are meant for a wide range of temperatures, upwards of around 600C. Many consist of a length of thin wire wrapped around a ceramic or glass **Core**.

A **Thermocouple** consists of two differing conductors which generate a temperature dependant **Voltage**. They have the highest temperature range of temperature **Sensor**s, upwards of around 1800C.

Digital temperature ICs will measure a temperature and report the data through a specific **Protocol**, which is discussed further (here).

Humidity **Sensor**s may be used along with temperature **Sensor**s, in a single package. The output can be **Digital** or **Analogue**. A couple of common **Digital** examples include the DHT11, DHT22 and AM2320.

There are many **Sensor**s for detecting compounds in the air, important for atmospheric research. Usually these are **Doped** so that a higher concentration of a certain gas, such as carbon dioxide, induces a **Voltage**. Because of the calibration involved and comparison with data, they are often complex ICs, which communicate via a **Protocol** such as **I2C** or **SPI** to a **Microcontroller**, discussed in the **Protocol**s section. Although certain **Sensor**s may leave the conversion to the user. More **Module**s are appearing in the market, helpful for taking more reliable measurements.

Photo**Resistor**s use a similar method for light, where a **Resistance** is associated with pitch black, and separately for light. Therefore, an estimated light intensity will be attained, with a given **Voltage**. There are also photo**Diode**s and photo**Resistor**s, each with their own benefits, like higher response rate, or lower costs.

There is a range of movement based **Sensor**s, which will quickly be touched upon. You are unlikely to

need to use these much.

One of the main cheap and widely used range/ movement **Sensor**s is an **Ultrasonic Sensor**, which has an **Ultrasonic** transmitter and receiver. This uses knowledge of the speed of sound, to measure the time it takes for sound to hit a surface and reflect, to then calculate a distance value.

There are also **LiDAR Sensor**s, which are similar but known to have higher precision, which use a singular beam of light, and calculate its time of travel. These are also known as Time of Flight (ToF) **Sensor**s.

The motion based **Sensor**s discussed above are ones that are contactless, but there are also contact based **Sensor**s, which a **Potentiometer** could be an example of, using its **Resistance** based on a linear or rotational movement as a measurement of movement.

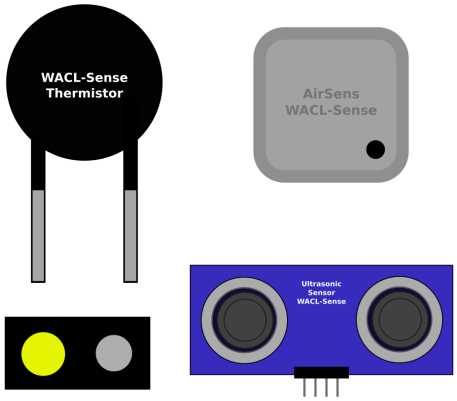


Figure 39: Example **Sensor** Packages

Examples

At this point you have seen a good proportion of components, to have more complex examples. Coming back to the heater suggestion mentioned earlier in the document, but this time there could be a **Control Loop** factored in. We will assume a the processing is all sorted by a **Microcontroller**. We can use the same **Resistor** / heating element of 2000W, but will use a **Transistor** to turn it on and off. For simplicity, and since a small **Voltage** drop should not have a massive effect, a **BJT** could be selected here. The **BJT** could be triggered to turn off when the value from an **Analogue Thermistor** correlates to a higher temperature than what has been selected as the required temperature. It could be a **Potentiometer** which is used as a temperature knob.

Quiz

1. What is the difference between a NTC and a PTC **Thermistor**?

2. What type of temperature **Sensor** would you choose for a furnace, with temperatures as high as 900°C.

3. If you needed to precisely measure a distance, what would be a good **Sensor** choice?

4. Name a couple of **Protocol**s that **Sensor**s may communicate with?

0.2.14 Optoisolators

Optoisolators, otherwise known as optocouplers, are components that use light to transfer electrical signals between two isolated circuits. This is important if the two circuits need differing **Voltage**s, or need isolating.

They use an **LED** and a photo**Sensor**/ photo**Transistor**, so it is light that transmits the signals, which keeps them electrically isolated. They are useful for measuring **AC** using **DC** electronics, so have uses within **AC-DC** power conversion/ supplies.

It may be worth reading the sections on photo-**Sensor**s/ photo-**Transistor**s within **Sensor**s, as well as **Transistor**s and **Diode**s, to fully understand this.

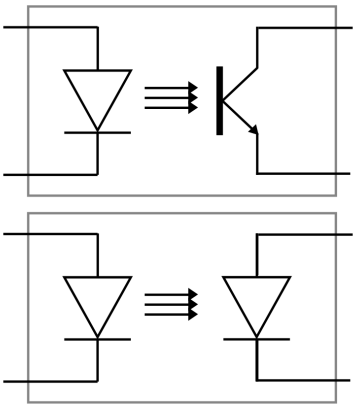


Figure 40: Optoisolator representations, phototransistor varient on top, photo**Diode** varient below

Quiz

1. Name an application where an **Optoisolator** would be helpful?
2. What components are within an **Optoisolator**?
3. Are **Optoisolator**s suitable for transferring a lot of power?

0.2.15 Power Sources

It is important to consider what power source you should use for a given project.

Batteries come in many different types, some being rechargeable chemistries and others not. Many batteries use lithium, a very common material for rechargeable batteries, although it requires careful consideration and the use of a charge IC, to ensure that both its environmental and operating conditions are suitable. If not used wisely, it can be a dangerous chemistry with a range of risks stemming from mishandeling, shorts, over and undercharging and more.

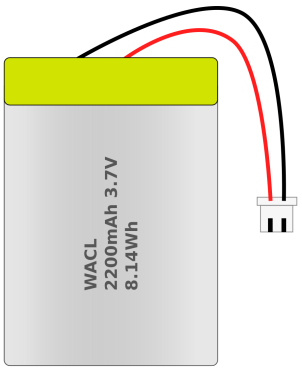


Figure 41: Example Lithium Polymer (LiPo) battery

There are a few more forgiving rechargeable battery chemistries, one of the main ones being NiMH, which has characteristics in a similar magnitude to lithium cells, although generally worse in both energy and power density. Energy density is the amount of energy it can store either per litre or kg, so will affect how long a product will last, where power density is the magnitude of power it can continuously supply, again either per litre or kg. Energy density is given in Wh, which is the number of **Watt**s of energy it could supply for an hour.

NiCD and Lead Acid are two other battery chemistries which still get some use, but are decreasing in relevance due to them having worse characteristics in a range of aspects, and poor carbon footprint. Lead acid is still used for certain applications due to how easy it is to charge and discharge, and its low cost per Wh.

Common non-rechargeable battery chemistries include alkaline and zinc carbon. Alkaline tends to provide more energy, and tends to use lower pollutants, so should be the better choice than zinc carbon unless a very low budget is a key constraint. So other than costs, I would say use of rechargeable battery chemistries would be best in most cases.

Along with battery sources, considering wired methods are also useful. **USB** is one of the more common methods, with most **USB** methods transmitting 5V. **USB** C for example allows for devices to "negotiate" their **Voltage** draw, between 5-20V, with a 5V default. Micro **USB** B, which used to be a standard for portable devices in some cases can handle a max of 3A, where **USB** C can handle up to 5A, making it suitable to power many projects. The connector has a higher durability, and can be inserted either way. Other data transfer can be done through it as well, making it versatile.

DC Jacks are useful for power only, and can vary in dimensions and power output. It is fairly common to get **DC Jack** based power supplies with 6V, 12V and 20V **Voltage** outputs, with a range of **Current** outputs.

Adjustable **DC** power supplies are helpful for testing stages, or if specific **Voltage**s are needed, and will power a device with **Voltage**s between around 0-30V and sometimes 0-60V or more. 5A and 10A are both fairly common for max **Current** output capabilities.

Supplementory power methods include the use of a **Motor** as a **Generator** or a solar panel, spoken about later. Alow energy density battery alternative with quick charging capability is a **SuperCapacitor**, suitable for very low power applications, potentially topped up with a solar panel.

Quiz

1. A temperature **Sensor** for remote use needs to be made, where it takes a measurement once an hour, with someone manually retrieving data once a week. Name a power source which might be suitable?

2. Why are lead-acid batteries still used, and why are they not very goof for a lot of cases now?

3. A device for lab use needs a specific **Voltage** of 27.6V to run. What do you think a good choice

would be to power it?

4. What is commonly known as quite a safe battery chemistry?

5. What is good about **USB** C?

0.2.16 Ideal vs Non-ideal Components

Until now, we have talked about components as if they are ideal. This means they have no losses other than expected. In reality, components have internal **Resistance**, **Capacitance** and **Inductance**. For low-speed circuit design, the internal **Resistance** is what you mainly need to be aware of. This includes the source, with certain batteries/ sources having a high internal **Resistance**, which means at higher **Current** draws, there can be a fairly significant **Voltage** drop before it even gets to the rest of the circuit. Any of these factors can have a significant effect upon a more high-speed circuit, with **Capacitance** and **Inductance** altering over different frequencies. In high speed circuit boards, there are a range of techniques used to limit the potential for unwanted **Capacitance** and **Inductance** to affect a circuit. Issues such as this could be what causes circuit problems, especially for breadboard designs. These are often known as **Parasitics**, which are undesirable characteristics affecting components or the circuit.

Because of factors such as the above, component functions don't always happen instantaneously, such as there being a turn on/ off or switch on/ off time for a given component.

A battery/ power source would usually have an ESR value, which stands for equivalent **Series Resistance**. This is important to consider what losses, heat build up and **Voltage** drop could be seen at particular loads.

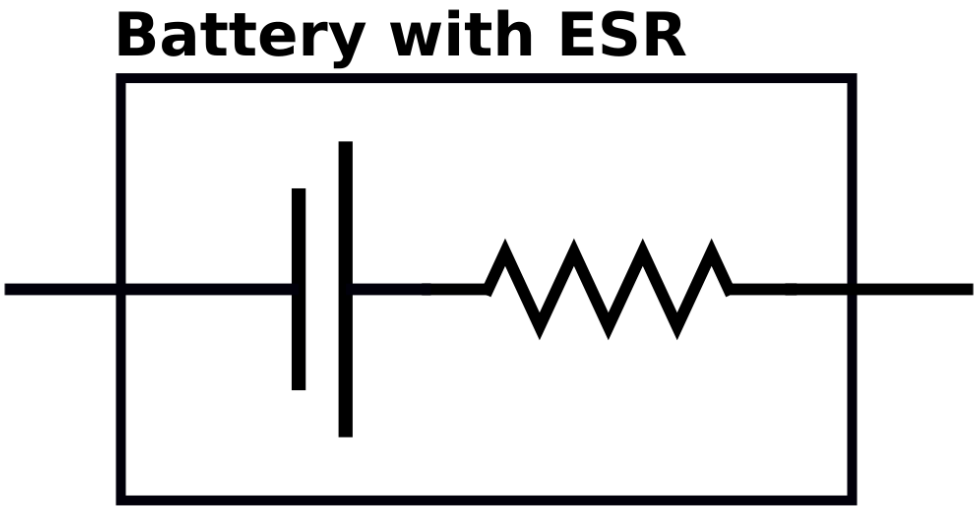


Figure 42: Battery with ESR Representation

It is also beneficial to know the term **Impedance**, which is the effective **Resistance** of an **AC** circuit, being a combination of both **Resistance** and **Reactance**. As mentioned above, circuits will have a level of **Inductance** and **Capacitance**, which causes **Reactance** in an **AC** circuit.

Quiz

1. True or false, real world components can be thought of as non-ideal?

2. What characteristics do non-ideal components have, which may cause issues for certain circuits?

3. True or false, non-ideal components tend to be more of a problem for low-speed circuits.

0.3 Power conversion

Power conversion could be its own course, so this lightly touches upon conversion for a base understanding and why it is important. Within the further reading section, there is information on **Rectification**, combining theory above, so is worth checking for more of a challenge, or if **AC-DC** conversion knowledge is necessary.

The main types of power conversion are active and linear. Active utilises both **Capacitor**s and **Inductor**s, as well as switches for efficient conversion of **Voltage**, either to a higher or lower level, with efficiencies into the 90% possible. Whereas linear power conversion is cheaper, requires a lower number of components and is specifically for decreasing the **Voltage** at the output. It acts as a variable **Resistor**, to dissipate the unwanted **Voltage** as heat, so depending on the circuit's needs, can be particularly inefficient. Therefore, if efficiency is less important than costs, or the **Voltage** decrease is only very minimal, linear **Voltage** converters can be an okay option. Given how easy it is to find power conversion **Module**s (either known as step-up or step-down, whether **Voltage** is increased or decreased), these are usually better than custom designs, as usually only a **Potentiometer** needs altering to get necessary output **Voltage**, with minimal losses.

For a quick active power converter insight, a switch is turned on at set ratios to charge an **Inductor**, which then discharges to charge a **Capacitor** for a smoothed **Voltage** at the intended level. The ratio between the switch on and off times will affect the output **Voltage**.

The above mainly discussed **DC** to **DC** power conversion. To do **AC** to **AC**, a **Transformer** is commonly used. **AC** to **DC** may use a similar method to the above but using a **Transformer** rather than an **Inductor**, and the use of a **Rectifier** to convert the **AC** to **DC**.

Quiz

1. Name three other types of conversion if the example is **DC** to **DC**.

2. True or false, active power conversion is always more efficient than linear power conversion?

3. True or false, it is typically recommended to use a **Module** for active power conversion due to their design complexity?

4. What would the efficiency be of a linear power converter, converting an input of 5V down to 3.3V?

5. For a variable load circuit, which runs off 3V and has a 3.3V supply, would a linear regulator be a good option, and why?

0.4 Communication Protocols

0.4.1 Wired Communication Protocols

Communication **Protocol**s are important to interface with different devices and equipment, so a base understanding of some of the major ones would be important.

Communication **Protocol**s are a set of rules which devices follow to transfer information, allowing coherent data transfer at set speeds, depending on the use case.

It will first be useful to distinguish between **Serial** and **Parallel**, since these are widely referenced regarding communication **Protocol**s.

Serial means that there is a single data line/ wire used for communication in a given direction, where **Parallel** means that there are several data lines or wires with data travelling through. Therefore, **Serial** is often an easier method to use or implement.

They can first be split into two main kinds, with inter system **Protocol**s being those used to have communication between devices, where intra system communication **Protocol**s are meant for between component communication.

Therefore, you may use a combination of the two, depending on the project.

It is useful to know the term Port. Ports are points where communication can be done from, to a specific device.

USB and **UART** are the main examples of inter system **Protocol**s, where **SPI**, **I2C** and **CAN** are some of the main intra system **Protocol**s. Although some of the main **Protocol**s, it is not a definitive list, with niche or proprietary **Protocol**s being around. These may have specific benefits, or are proprietary to limit use of other companies' chips for a given application.

Master and Slave are still common terms within communication **Protocol**s, that you need to be aware of, although there is a shift to alternate terms such as controller and responder, although not yet standardised. Other alternatives could include primary and secondary, leader and follower and source and sink, although the latter of which already is widely used within electronics. Because of the negative connotations of master and slave, this course uses the suggested terms of controller and responder. I2C is an example interface where the Master and Slave terms have officially been replaced with Controller and Target, although it may take some time for this to be fully recognized.

USB is very widely used and is how many devices are connected to computers. It is high speed, and quite simple to implement, although requires the use of specific **Driver**s and a powerful controller. **USB** 1 and 2 have 4 pins, one for power, one for ground and two for data, with data being sent **Serial**ly in each direction.

UART is physical circuitry for conversion between **Serial** and **Parallel** data, where two **UART**s may communicate with each other, communicating in **Serial**, meaning that only two wires/ data lines are necessary for communication between two devices.

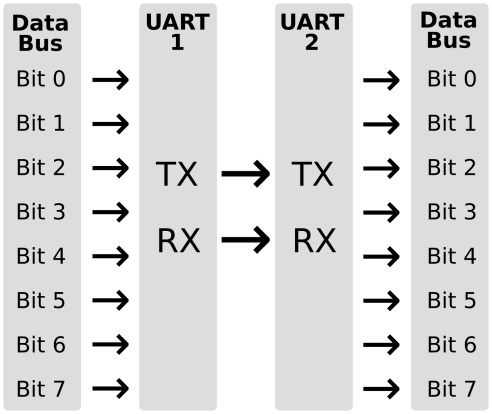


Figure 43: UART Protocol diagram

RS232 is another **Protocol** (Recommended standard 232), which is often used within telecommunications. In general, it is often used for connecting devices such as data acquisition or modems, the prior of which are likely to be particularly important. On creation, **RS232** allowed for "handshaking", a method for checking that data is transmitting correctly, although many devices no longer require the use of this. **RS232** can directly connect to the **Serial Port** of a computer.

ModBus is a **Protocol** which was created for communication for programmable logic controllers, often for industrial use. One reason for its wide usage is that it has no royalties and is open.

SPI is a commonly used **Protocol**, which has a single controller and multiple responders connected to it. **SPI** stands for **Serial** Peripheral Interface, with there being 4 main lines used, two pins for data transfer, a pin for selecting which responder the controller is communicating with and a line used as a clock. Therefore, for each responder that the controller wants to communicate with, a separate select pin is required. It is a relatively fast communication **Protocol**, but doesn't scale the best due to needing a new **GPIO** pin for each new device.

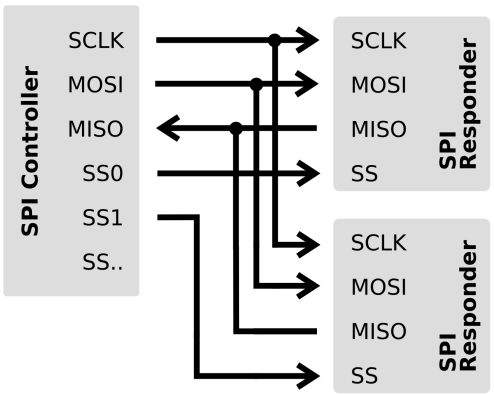


Figure 44: SPI Protocol diagram

I2C is a similar **Protocol**, which is also **Synchronous**, like **SPI**, controlled with a clock from the controller. It only requires two wires for communication, the clock and the **Serial** data line. It can communicate with many devices/ responders with just two wires, but is limited in speed and has a combined transmitting and receiving data line.

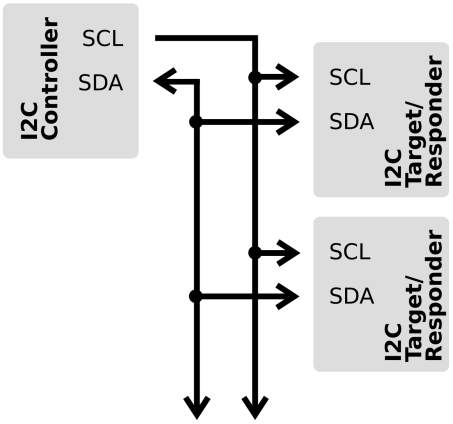


Figure 45: I2C Protocol diagram

CAN stands for Controlled Area Network, and is a **Protocol** meant for the transmission and receipt of data within a network of devices, especially for harsher environments. Due to this, it is mainly used in industrial and automotive environments, so may be used by certain research equipment. It is fast, secure and low cost, but is complex and automotive orientated.

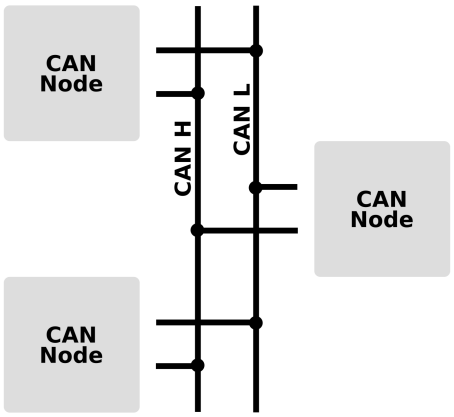


Figure 46: CAN Protocol diagram

Since not all devices have **Protocol**s that are usable together, there are chips for conversion of **Protocol**s, often seen as **Module**s which are often used to interface between a device and a computer. As previously mentioned, **USB** often requires **Driver**s to be installed, so you should be aware that you may need to download a **Driver** for interfacing with it.

This section will quickly go over **Module**s/ chips you may come across for communication with these different **Protocol**s.

USB to **UART Module**s are very common, often having a **USB Port** on one side and 4 pins on the other. These may use a range of chips and are likely to use TTL level, which is **Transistor-Transistor** logic between the limits of 0 and VCC, which is often either 3.3V or 5V. A supplier will mention whether it works at TTL **Voltage**s, or others such as RS-232, a standard meant for telecommunications.

I3C is a recent protocol which may start to get some more usage as time goes on, designed to get the benefits from both I2C and SPI, while allowing some backwards compatibility with I2C. This uses the official terms Controller and Target, which is part of what led to the change in name with I2C. It is not yet widely used, but may end up replacing, or being used in conjunction with I2C, depending on constraints such as performance needs or budget.

Quiz

1. What is the difference between intra system **Protocol**s and inter system **Protocol**s?

2. Name two intra system **Protocol**s and two inter system **Protocol**s.

3. You have a system which uses a range of seperate individual but connected industrial **Sensor** units, name an appropriate **Protocol** that could be used?

0.4.2 Wireless Communication Protocols

Although we won't go into too much detail, it is helpful to be aware of some wireless communication **Protocol**s and their benefits.

Wi-Fi- This is a family of wireless **Protocol**s meant for local area networking, and is one of the most widely used.

Bluetooth- Bluetooth is a short range communication **Protocol**, meant for communication between a fixed and mobile device. It has been through several iterations, with Bluetooth 5.0 having distances upwards of 400m, but it is typically below 10m.

ZigBee- This is meant for creating local area networks of devices, with communication range between 10-100m.

LoRa- This is a long range radio **Protocol** meant for unlicenced radio bands, with ranges in the Km range being viable, depending on the landscape and antenna used.

GPRS- This is the **Protocol** which is the mobile standard for 2G and 3G mobile networks.

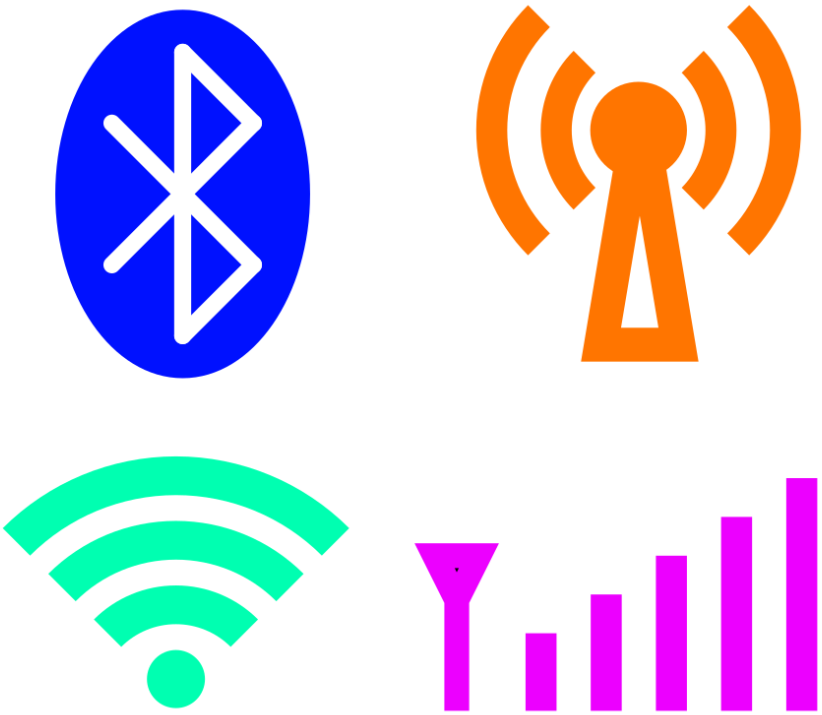


Figure 47: Wireless Communication Icons

Quiz

1. What would be a suitable communication **Protocol** for a sensing unit 600m away, given an appropriate antenna is used?

2. Name an example **Protocol** meant for local area networking?

0.5 Integrated Circuits (ICs) and Microcontrollers

Along with the discrete components mentioned above, there are many integrated circuits, components with a circuit built-in for specific functions. A complex example of this, which is widely used, is **Microcontroller**s.

Integrated circuits may do a wide range of things, such as amplification, timing and power conversion. These help to decrease the time, complexity, power consumption and often costs of a design by packaging all the discrete components necessary for its function. Usually a few passives or discrete components are required, which are referenced in the parts **Datasheet**.

This section will quickly talk through some of the more important integrated circuits and their uses.

Amplifiers- A range of chips meant for the amplification of signals, usually analog signals.

Circuit Protection- A range of components meant to protect other aspects of the circuit, such as **Digital** fuses or **ESD** protection.

Timing- A range of chips to provide the timing of other chips.

Data Conversion- Chips used to convert one type of communication **Protocol** to another, or for analog-to-**Digital** conversion.

Display/ **LED Driver**s- Chips meant for the driving of displays or **LED**s.

Interface ICs- Chips meant for communication over different **Protocol**s, or to interface with different parts/ devices.

Communication ICs- A range of ICs and **Module**s for wireless communication, such as Wi-Fi, Bluetooth, LoRa, ZigBee.

Logic ICs- Chips that work through low level boolean logic, for building logic based circuits.

Memory- Necessary for storing data.

Power Management- A range of chips for power management, whether this is active or linear power conversion, battery management, power monitoring or other driving chips.

Sensors- A range of chips for interfacing with analog sensing elements or combining both into a single package.

Quiz

1. What is an integrated circuit?

2. Are integrated circuits always better than using discrete components?

3. Name two advantages of using an integrated circuit?

0.5.1 Microcontrollers

Microcontrollers are low power processing units, with built-in memory, that can be programmed to do a range of functions. These may typically be put together as a single complete unit, such as an Arduino board. An Arduino is a board with a **Microcontroller**, passives and discrete components to interface with basic electronics components.

There are a range of **Protocol**s by which **Microcontroller**s may communicate by. They also may contain a range of integrated circuits, to help interface it with different components, such as with an **Analogue Sensor**. A**DC**sare frequently used (often internal to a **Microcontroller** but are also available as an IC). An A**DC** is an **Analogue** to **Digital** converter, which will represent an **Analogue** value as a bit value, depending on its resolution. This allows a **Voltage** reading to be associated with a **Digital** value, for use within the program.

The pins of a **Microcontroller** that enable data transfer are known as **GPIO** (General Purpose Input Output) pins. This term is widely used, but sometimes may just be written as IO pins. These have limitations, such as what **Protocol**s they can be used for, along with electrical limitations like max **Voltage Tolerance**, or **Current** output.

These have a built-in processor, which operates at a set clock speed, as well as **RAM** , **ROM** and sometimes **Flash** memory. A processor is what computes instructions, based on what is in memory. These operate at a set rate given in Hz, with some instructions possibly taking multiple cycles. They are commonly in the MHz region. **RAM** is fast access memory, to store data which is needed by the program, such as variables. Then memory such as **Flash** is slower access, used for storage of whole programs, which elements of which can be moved into **RAM** when it needs to run.

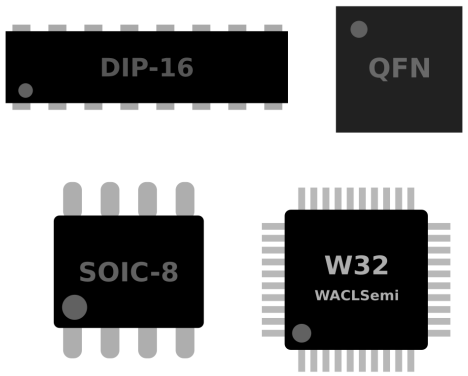


Figure 48: Example IC and Microcontroller Packages

Quiz

1. True or false, a **Microcontroller** doesn't always have **RAM** ?

2. True or false, a **Microcontroller**s output pin could be used to directly power a filament bulb?

0.5.2 Amplifiers

Amplifiers are there to increase the amplitude of an **Analogue** signal, often useful for use alongside **Analogue Sensor**s, to take it to a **Voltage** range readable by a **Microcontroller**. A common component for amplification is an **OpAmp**, which, when wired differently, can amplify a **Voltage** difference, or amplify a signal by a specific amount. This can be negative or positive amplification.

An understanding of how amplifiers work internally is beyond the course scope, although understanding of their uses is helpful. We will touch upon the types mentioned above in some further detail. Each of the below uses an **OpAmp** as the principal component.

The primary focus will be on the use of **OpAmp**s to create amplification circuits.

Although, not an amplifier, it is important to separate two circuit elements. This is known as a **Voltage** follower, which isolates the **Resistance**s at either end of the part, particularly important where you may try to read a **Resistance** value from a **Sensor**.

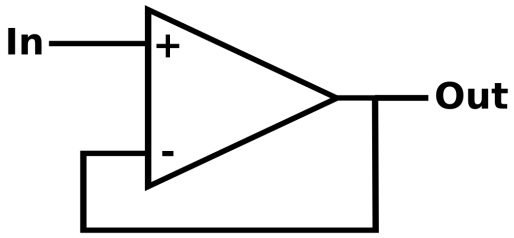


Figure 49: **Voltage** Follower representation

There are non-inverting and inverting amplifiers, which the first of which amplifies a **Voltage** in its **Current** direction/ magnitude, and the latter amplifies a **Voltage** in the negative/ opposite direction or magnitude.

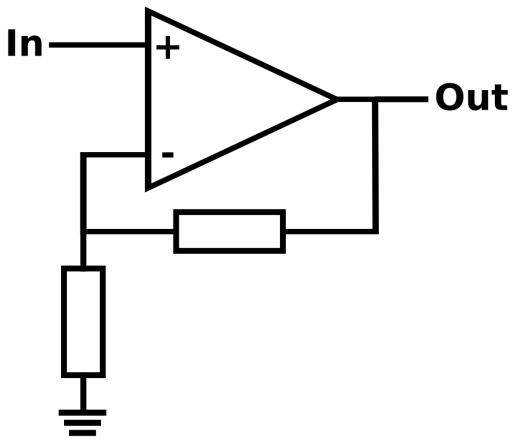


Figure 50: Non-Inverting Amplifier representation

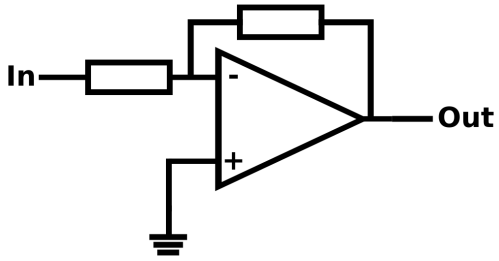


Figure 51: Inverting Amplifier representation

There are summing amplifiers which can add two **Voltage**s together.

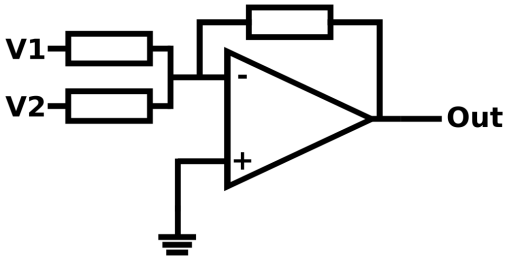


Figure 52: Summing Amplifier representation

Differential amplifiers output the difference between two **Voltage** inputs, with a certain level of amplification.

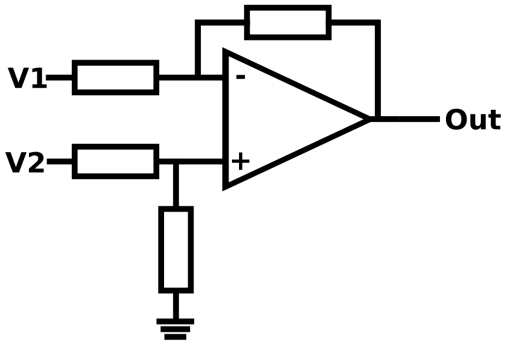


Figure 53: Differential Amplifier representation

Quiz

1. True or false, **OpAmp**s can be used in a range of configurations to act as different amplifiers?

2. True or false, amplifiers are suitable for providing power for high power components?

0.6 Boolean Logic

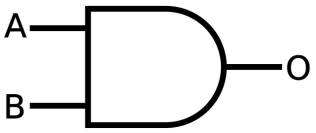
Boolean Algebra is a branch of mathematical algebra, where variables are true or false, and it is centred around three main operators "And", "Or" and "Not", and several sub-operators.

For this section, the practical elements use DDSIM, another online simulator, developed by a lecturer at the University. Although similar functionality can be done through Falstad circuit simulator, DDSIM is more streamlined and intuitive for **Digital**/ boolean electronics. Feel free to see which you prefer for which task, as they are not completely different. **Series Link**

(There is also DASIM, for **Analogue** electronics, similar to Falstad, where it can be easier to draw up circuits, it isn't quite as easy to visualise, **Series Link**).

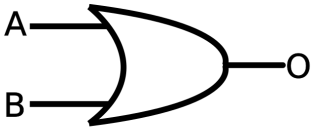
First, we will describe what the different operations mean. Regard it as a comparison of usually two inputs, either true or false, on or off, a **Voltage** present or not, or 1 or 0. We will use true and false, being the main terminology in **Boolean Algebra**. First, "And", is where both inputs are true, the output will be true, with any other combination having a false output. Then there is "Or", which states if either or both input is true, then the output is true. The last key type is "Not", which has a single input, where the input is the reverse of the output, true-false, false-true. Within electronics/ circuitry, **Gate**s are used to allow this logic, so you have And **Gate**s, Or **Gate**s and Not **Gate**s. There are several more complex examples, such as NAnd and NOr, which are a combination of an And followed by a Not and an Or followed by a Not. Another more complex example is an Exclusive Or (XOr), which has an output which is only true if only one input is true. A combination of **Transistor**s can make these **Gate**s, with a combination of NAnd **Gate**s creating any **Gate**. This has become the normal, since NAnd **Gate**s are simple to produce from **Transistor**s.

In the examples, we have used A and B as the inputs and O as output.



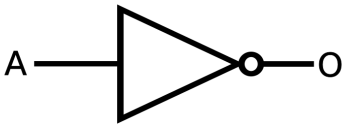
A	B	O
0	0	0
1	0	0
0	1	0
1	1	1

Figure 54: AND Gate representation and Truth Table



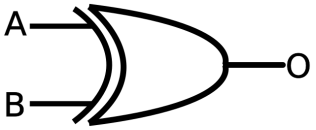
A	B	O
0	0	0
1	0	1
0	1	1
1	1	1

Figure 55: OR Gate representation and Truth Table



A	O
0	1
1	0

Figure 56: NOT Gate representation and Truth Table



A	B	O
0	0	0
1	0	1
0	1	1
1	1	0

Figure 57: XOR Gate representation and Truth Table

Gates such as NOr and NAnd usually are represented by their sign with a circle at their output, like a Not Gate has. This can be seen below:

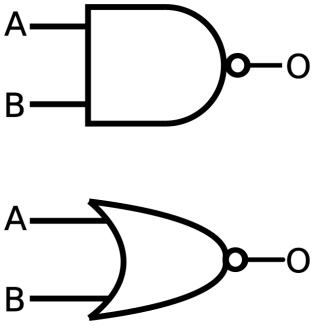


Figure 58: NAnd Gate and Nor Gate representation

Boolean logic has led to modern computing, as it is a combination of logical operations done with **Gate**s. There are fairly standard circuits you can try to make, to ensure you fully understand boolean logic.

A famous theorem within Boolean logic is De Morgan's Theorem/ Law, which shows how AND logic can be represented through OR. We will use the below to represent this-

A & B are our inputs or "sets".

$\bar{A}$ is the compliment of A, equivalent to A through a Not **Gate**.

$\cap$ is the intersection, otherwise known as an AND **Gate**.

$\cup$ is union, otherwise known as an OR **Gate**.

With the terminology explained, De Morgan's Law can be shown as the below:

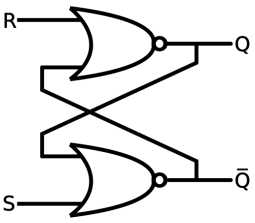
$\overline{A \cup B} = \bar{A} \cap \bar{B}$

$\overline{A \cap B} = \bar{A} \cup \bar{B}$

This also helps to explain why NAND **Gate**s are typically used within circuitry to represent other **Gate**s.

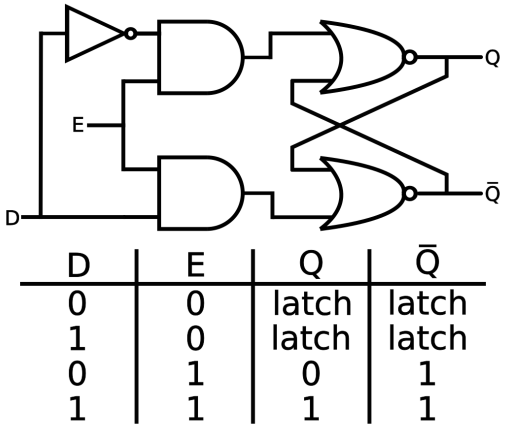
There are a couple of advanced components made with logic gates which would be important to be aware of, especially for when you are doing the practical elements of this course in LabVIEW.

The first of these is a Latch (also known as a flip-flop), which can be thought of as a basic memory element. It is an element made of just a few gates, which can latch a value while the "D" input is false, and will output "E's" value when "D" is true. That is an example of a D latch, but the most basic form is an S-R latch, formed from only two NOR Gates.



R	S	Q	$\bar{Q}$
0	0	latch	latch
1	0	0	1
0	1	1	0
1	1	0	0

Figure 59: S-R Latch Gate Representation



D	E	Q	Q̄
0	0	latch	latch
1	0	latch	latch
0	1	0	1
1	1	1	1

Figure 60: D Latch Gate Representation

Shift registers have a chain of latches, where the output of one is connected to the input to the next. This theoretically allows multiple output pins to be controlled from a clock and GPIO pin. How shift registers are used in circuits will be discussed in the further reading section.

If this is an area that particularly interests you, I would recommend looking at the site, **Series Nandgame**, which takes you step by step from placing **Gate**s to building a basic **Gate** computer.

Quiz

1. What is De Morgan's law, and why is it important?
2. What discrete components allow us to make logic gates?
3. Name three gates and what they do.
4. Draw a logic table for the XOR gate.
5. What could be thought of as a basic memory element, made out of a few logic gates?

0.7 Electronics equipment

There are a range of equipment that aids working with electronics, some may seem initially overwhelming. Understanding what they do, why and if they are needed, is important!

0.7.1 Multimeters

A very common piece of equipment is a **Multimeter** . These are used to measure **Current**, **Voltage**, and **Resistance**. When measuring **Current**, the **Multimeter** needs connecting in **Series** with what is being measured. In contrast, to measure **Voltage** or **Resistance**, you place the probes in **Series** with the component or circuit segment needing measuring.

There are analog and **Digital Multimeter** s, the latter more common, and what this course focuses upon.

We will talk through the main **Multimeter** functions now.

You can measure **Resistance**, **Current**, **Voltage** and continuity on most, with some measuring hFE (**Current** gain) of **BJT Transistor**s. Some may also measure **Capacitance**.

Depending on the **Multimeter** , some will split each of the sections into specific scales, representing the max value it can show. This is so the resolution shown is close to the measurement magnitude. If, when trying to measure a value, nothing seems to be read, you may need to alter this scale.

The continuity tester is used to check whether two points within a circuit are connected, or whether **Current** can flow between them, so is helpful for checking **Soldering**.

For most functions, connecting the cables to the V/Ω/mA section would be suitable as well as COM/ ground, unless it is a high **Current** circuit, if so there is the third connection usually for **Current**s at a maximum of 5A or sometimes 10A.

A more complex piece of equipment for complex diagnosis is an Oscilloscope. These are used for measuring **Voltage** and **Current**, compared against time, in graphical format. The software sometimes used within this course Falstad, uses scopes which are a representation of what you would view through an Oscilloscope probing a specific component or circuit element. They are particularly helpful for troubleshooting signals or **Inductor**s or **Capacitor**s.

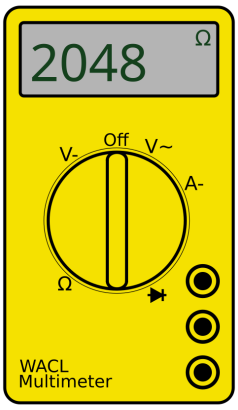


Figure 61: Example Multimeter diagram

0.7.2 Breadboards

For electronics prototyping, breadboards are seen to be important, as they allow you to connect **DIP** components quickly for prototyping. They have many rows of 5 pins which are interconnected, so you can create an array of complex circuits with them, to test out ideas. Remember that they may not produce the most stable circuits, and would not be recommended for the field due to loose connections.

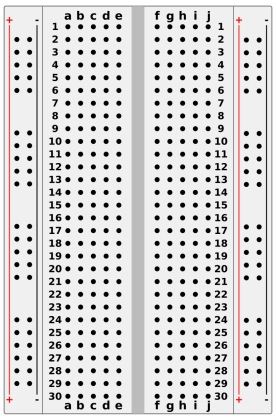


Figure 62: Example breadboard diagram

Internally, it is just rows of metal, which connect/ bridge pins together. As seen, there are two sets of 30 rows of 5 pins, with each pin on rows connecting. Two connecting parts should be on the same row. There are also power rails, which allow for 2 different **Voltage**s, and two sets of ground pins. These are convenient to access, but not the only suitable pins for providing power. Breadboards are specifically meant for **DIP** components, with the gap in the centre allowing for **DIP** chips to be placed, separating

each of their pins. Components can be connected with male to male **Jump** wires. As mentioned, it is worth only attaching power when you are sure that everything is firmly connected, with no potential for shorts. There are limits to the suitability of a breadboard circuit, which are meant for lower power and lower speed circuitry.

Examples

Here are a range of example circuits to help you understand how a breadboard circuit should be wired, with increasing complexity.

The first example is an **LED** with a **Resistor**, a button, and a battery.

Challenges

1. Connect a circuit with a blue 5mm led, 2 AA batteries, a 50Ω **Resistor** and a momentary push button.

2. Connect a circuit with 2 AA batteries, with a blue **LED**, **Resistor** and button, in **Parallel** with a red **LED**, **Resistor** and button. You will need to calculate the **Resistor** required.

Here are a range of challenges, to build up your confidence. You may need to read other sections of this course if you are struggling.

The next stage from here is a proto-board design, using vero or strip board, design of a custom circuit board, or using existing **Module**s suitable for the design. The cost of custom printed circuit boards has decreased considerably in the past few years, through sites such as JLCPCB. This has made custom **PCB** design a more viable option for field use. In the further reading section, there is detail on **PCB** design and related resources, but it is not a key course focus.

0.7.3 Soldering

Soldering is the process of creating a metal joint between a component and a circuit board, to allow for a considerably improved electrical, mechanical and thermal connection. Solder is typically in wire form, but can also be in paste form, and it is a metal with a low melting point, allowing for a quick

connection to be made with a reduced risk of damaging components.

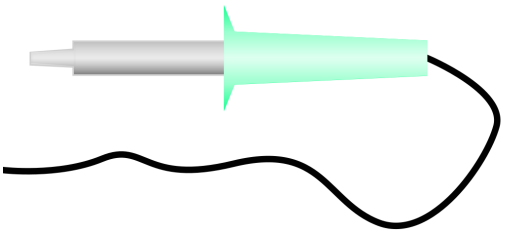


Figure 63: Example Soldering Iron diagram

The most basic form of **Soldering** uses a **Soldering** iron and solder wire. It is recommended to use a holder, to hold the parts in place, to free up hands for **Soldering**. The **Soldering** iron should be placed, so it heats the metal of the component and the interface metal, with solder wire placed at the other side so the solder can bridge them, making a clean connection. A clean connection should have a clean flow between the parts, rather than a ball of solder or poor joint.

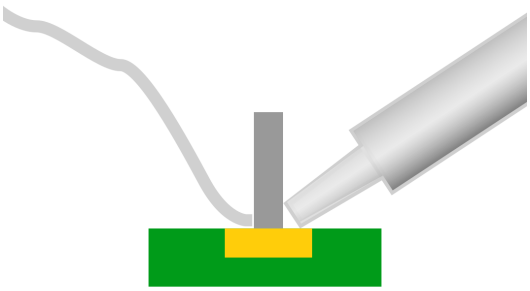


Figure 64: Soldering representation

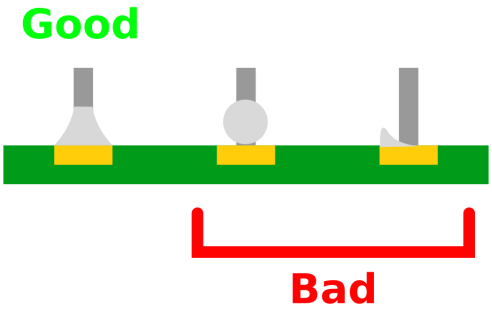


Figure 65: Example Solder Joints

Sometimes too much solder may be added, which can be difficult to remove. A solder sucker can help, where you remelt the area with excess solder and quickly press the solder sucker just above, which uses suction to remove excess solder. Solder wick is another option, which I prefer, as you place it between the excess solder and your **Soldering** iron, where it "wicks" up the additional solder. This tends to be copper, with high heat transfer, so keep hands further from the iron.

Soldering temperature is important. It is worth looking at the melting point of the solder you are using, with lead-free solder typically having a slightly higher melting temperature of around 217C, depending on composition.

Although lead-free solder is known to be slightly more difficult to use, being better for the environment and for your health for me makes it the better option. I haven't found too high a difficulty jump, especially if good quality equipment is used. Also it would be more beneficial learning what's used in commercial/public products, and shouldn't be phased out.

Being able to solder is an important skill to have. Usually, someone would first make a circuit using **DIP** components before attempting more challenging **SMD** components.

0.8 Electronics safety

Safety is a key consideration when doing anything with electronics. If you are unsure, it is always best to ask rather than to assume.

As discussed, **Current** is related to **Voltage** and **Resistance** through $V = IR$. The body has a **Resistance** which can vary depending on length etc, but is relatively high (this can decrease significantly from water/ high humidity). Therefore, as **Voltage** increases, so will **Current** if the **Resistance** stays the same. You should always take sufficient precautions when using electronics, but especially when dealing with higher **Voltage**s, where it can be fatal. The in-person practical elements use low **Voltage**s, which are much safer. It is important to know when you don't have the knowledge or resources for a given piece of electronics and when it may be the better option to have someone else repair it. You should also consider that sometimes the best option is to replace something rather than repair it, such as when it may be dangerous, could end up costing you more if it isn't reliable, or takes a considerable amount of time to fix.

Component ratings are often not initially considered but are very important. Knowledge of ratings ensure that a selected part can take the expected power loss without damage. Therefore, it is important when selecting a **Resistor** to calculate its estimated power drop, and select a **Resistor** with a power rating clearly above the max predicted power losses. It is important to add a safety buffer, as circuits in reality are not ideal, so may react slightly differently than expected, such as a component having a **Resistance** value that differs from its value. **Tolerance** is the percentage at which a value can differ. This is usually under 20% (it can be considerably smaller). If you are unsure, it is better though to add a safety buffer, double the calculated value is fairly common, as it helps to account for how the **Tolerance**s of other components affect that one.

All components have losses, including wire, so this needs to be considered, with a wire or connector of suitable ratings selected. With these losses, also comes a **Voltage** drop, which may also be a factor, as if high enough, can cause issues with circuit functionality.

For wires, you have different wire **Gauge**s, and sometimes different materials. Typically wires are copper, but sometimes are aluminium, with a **Resistance** around 64% higher than copper, so looking at wire **Gauge** alone isn't enough. In general, there are standardised values for what **Current** flow is suitable through a given **Gauge** of wire. Again, it is helpful to keep the chosen wire a fair bit above the power rating you require, considering that only just meeting those ratings will lead to considerable

losses in the wire. And if there are cost constraints, it is important to consider that a thinner wire, which is initially cheaper, may end up costing more due to electrical losses.

There are a range of connectors, and depending on the application, will have their own ratings. Mechanically connected, such as screw terminals, are likely to have a **Resistance** higher than directly soldered, but can give more flexibility to make alterations.

You should be careful when handling components, as built up static can cause damage. Grounding yourself, such as touching a metal surface, can help to reduce this static potential. Anti-static wrist straps are something that can be sourced cheaply to limit the potential of causing damage.

Soldering may be something that is required for a project. Health risks are associated with all forms of **Soldering**, so you should always solder in a well-ventilated area with suitable equipment. If possible, although slightly harder to work with, it would be recommended to use lead-free solder, reducing health risks and allowing for safer use in the field. **Flux** is used in solder, to help it flow, but its evaporation produces a vapour which can be harmful if inhaled. Many **Soldering** iron setups have extraction next to the iron, to minimise this chance of inhaling it.

Before connecting wires, particularly at higher **Voltage** levels, running your hand along a cable to check for potential damage is important, replacing a cable if this is the case. It is also good practice to ensure both sides of equipment being connected are turned off before and during wiring, to prevent potential sparks, damage or electric shock.

Certain components can still be dangerous regardless of whether they are powered. **Capacitor**s are an example, which still hold power after a circuit has been turned off, for a **Period** of time. Therefore, ensure not to touch or short one in a circuit. It is good practice to have bleeder/ discharge **Resistor**s connected at either end of high **Voltage Capacitor**s so that when not powered, over time they will discharge. This will add some inefficiencies to a system, but improves safety.

As previously mentioned, it is important to ensure components aren't shorted, which can cause damage quickly as the source will pump as much **Current** as it can into the part, causing fast heat up. Storage of components, such as batteries in a proper case, is important to ensure that they cannot get shorted. Checking soldered circuits to ensure nothing is **Bridged**, such as using a combination of a microscope/ magnifier, and using a **Multimeter**scontinuity checker, as mentioned previously, is a worthwhile exercise.

0.9 Important considerations

0.9.1 Component footprints

Components can be found in many footprints, which needs consideration in design or repair. Imperial measurement is commonly used for surface mount **Resistor**s, **Capacitor**s and **Inductor**s, with values in mils, representing a 1000th of an inch. In many cases surface mount components are used in commercial products, so may be required in product repair. Understanding their size, and your ability to solder them is important. You may be able to access a **Soldering** oven, which can help make this process easier, considering that as solder melts, it can somewhat help to pull a component into place, if it is marginally off-centred.

Components are only rated for a certain amount of heat, so care should be taken to limit thermal damage. If a component with multiple pads is being done, leaving time between **Soldering** pins helps to prevent heat build up.

You should keep in mind that footprint alone may not tell you what a part is, as some footprints are used across a range of components.

0.9.2 Supply Chain issues

Since the Pandemic, there has been many supply shortages and stock issues. This has been worsened by some companies/ institutions beginning to purchase components in excess to ensure they had what they needed. Being aware of this is key in electronics design, especially when a certain project may need replicating in the future.

You should consider stock at the beginning of a design project, and ordering parts early if you know they are needed, rather than waiting for design completion, as by that point it could be out of stock. The main issue is with specialised ICs and **Microcontroller**s, less so with passives/ discrete components, which can likely be ordered once you know specific requirements. A similar mindset should be followed for **Module**s though, as they often have parts with low stock.

The parts shortages started in 2020 are some of the worst that have been seen, mainly affecting silicon components. There was a combination of factors which kick-started it. These included increase in demand for electronics, pandemic related shipping delays, natural disasters/ fires, vast increases in the

price of silicon and trade disputes. The market naturally goes through **Period**s of higher and lower supplies, with early 2020 meaning to be a **Period** of lower supply, further worsening things. The situation has slowly been improving.

Tips could include to use discrete or widely available parts over specific or niche ones where necessary, or **Module**s where there are similar alternatives available.

Digi-Key has a small **Lead Time** trends page. It is useful to be aware of **Lead Time**s for a given part, and how they might vary. **Series Link**

TTI also has a similar tool, with viewable graphs too. **Series Link**

0.9.3 Environmental considerations

There are two sides to environmental considerations. First, ensuring a product is suitable to the environment it is placed in, and second ensuring the design of the project has a limited carbon footprint or use of scarce, conflict, or toxic materials.

You should aim to use all **RoHS** certified components, luckily being most modern components. **RoHS** restricts the use of harmful materials such as lead within electronics. Solder is not always **RoHS** certified, as lead-based solder is still commonly used, especially with prototypes. Lead-based solder tends to be a mix of tin and lead, whereas lead-free substitutes the lead for some of the following: copper, silver, nickel and zinc, among others.

One way to somewhat reduce the carbon footprint could be choosing a smaller footprint, although for silicon products, this would only make a minor difference since it is likely that the silicon wafer fabrication is a large proportion of the overall carbon footprint.

There are limited steps in reducing carbon footprint within sourcing. Buying local will have an impact, but it's likely that they sourced the parts from long distances, anyway. Therefore, it is worth focusing on the longevity and reusability of parts. If designing a circuit board, using pin headers for **Module**s to slot into means that the **Module**s can be reused, or replaced if there are faults, rather than the whole unit.

There is a range of aspects regarding placement, such as whether waterproofing or cooling is needed. You should consider that rated temperature values are usually in ambient temperatures of around 25°C, so an enclosed part with poor thermal path, close to its rating, could fail over time. You may consider

whether active or passive cooling is needed, such as a copper or aluminium **Heatsink**, along with a fan if actively cooled. If a fan is used, the circuit enclosure should have air intake and outtake. Some component footprints may have **Heatsink**s specifically built for them. Thermal design/ thermal relief can be factored into **PCB** design too if needs be.

There are online calculators for figuring aspects such as thermal design, that are more time effective than learning thermo and fluid dynamics.

0.9.4 CAD and Material Design

You often require an enclosure for use of electronics in the field. Therefore, understanding different types of processes for their design would be useful.

Some of the main techniques for prototyping include 3D printing, laser cutting, and CNC milling, techniques widely used for quick prototyping of an enclosure.

3D printing mainly uses plastics, depending on the application. The most common is PLA (Polylactic Acid), a plastic derived from plants such as Corn. It is biodegradable and compared to most other plastics, has lower contamination risk. A problem is its low glass transition temperature (60°C), which means at lower temperatures it warps, so may not be suitable in direct sunlight for sustained **Period**s.

Lasercutting is suitable for some types of enclosure design. Depending on laser type and power, a range of different materials, from plastics like acrylic to wood can be cut. It is a good approach when quick and less complex parts are needed.

CAD Milling is a subtractive process where material is removed rather than added. Therefore, it is a wasteful process but allows a strong piece to be produced in a wide range of materials, including metals. Depending on the machine, there are limits to possible geometries.

There is a range of project enclosures meant for electronics projects, which can be sourced relatively cheaply, so are also worth considering to save time. These are typically seen in materials such as metals like steel or plastics like ABS. Some of these are**IP** rated, so can withstand different levels of water.

0.9.5 How to find a part

This is an important skill, especially in a changing market with parts sourcing causing many problems. You should first understand what you need, to then figure whats available. It is useful to remember that a part with your specific requirements may not be available, so consider bounds a part could work between, or whether circuit alterations are required to find parts. Remember, multiple parts can be used in **Series** or **Parallel** to get the equivalent value, ensuring other factors are considered.

You need to make sure to consider what restrictions there are, such as what you feel capable of soldering. Kits can be sourced to test out soldering of surface mount parts. 0603 tends to be on the lower limit of what is hand solderable, when tweezers are used, with 1206 also being quite common. DIP/ through hole parts can be considerably easier, and often do not require tweezers. You should keep in mind that costs and space can be higher with DIP/ through hole components, but can be easier and quicker to solder.

I will quickly talk through the processes needed for part selection. This is done in detail for a particular part, and in less detail with other parts, as different components have different characteristics which are important. For this I will use Digi-Key, as it a major components supplier, although it may not always be the cheapest, it is known to have one of the better "search engines". Therefore, once you have found the parts you could use, you can then check other suppliers or comparison tools such as Octopart.

I will give an example part, being a **Resistor**. Let's say I am designing a basic **LED** circuit, where when a button is pressed, the light turns on. It needs to use a 3.6V LIR2032 (a small rechargeable coin cell), and it needs to be a red **LED**. I have been told it should have a **Current** consumption of 25mA, with a 1610 footprint. I have been told it needs to be in the colour white. I have been told it should be 1206 size (considering that this represents a size of around 0.12*0.06 inches or 3*1.4mm). Therefore, the first step you can do is open up the Digi-Key homepage- **Series Link**

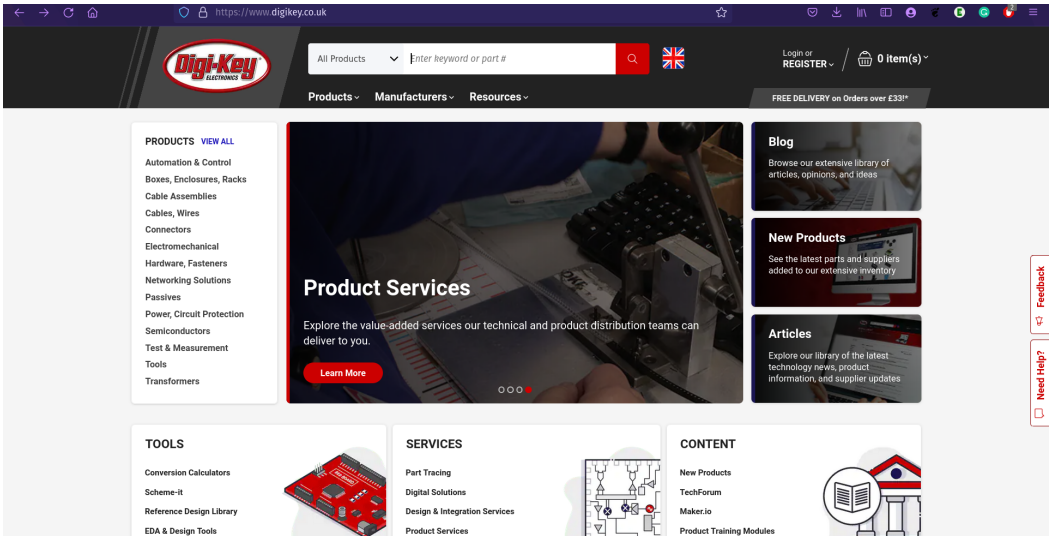


Figure 66: Digi-Key Home Page

There are many categories with subcategories within the products section, on the left-hand side. This is useful when we know what category is needed, but takes some time to learn, particularly when different sites categorise things differently. Therefore, you can instead use the search bar at the top centre, type **LED**, and press the search button.

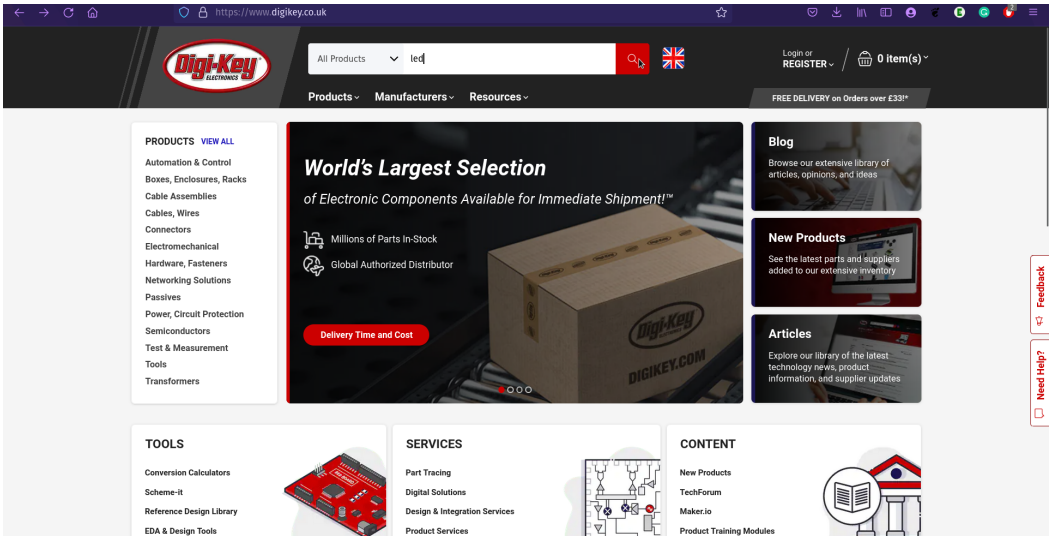


Figure 67: Digi-Key Home **LED** Search

It usually shows top results, each with a link and image, as well as listing the number of suitable parts. As seen, there is a white **LED** subsection, which is ideal.

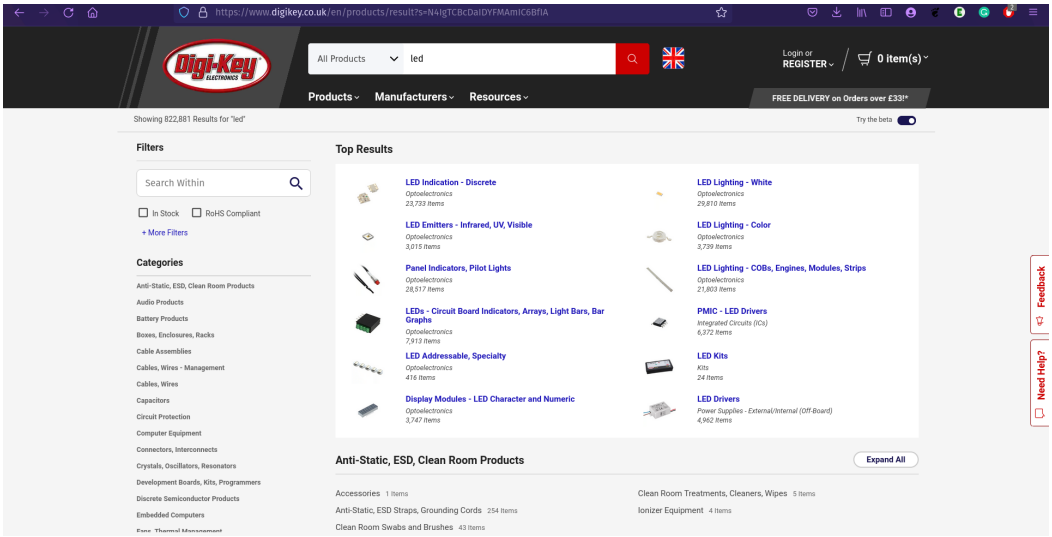


Figure 68: Digi-Key **LED** searched

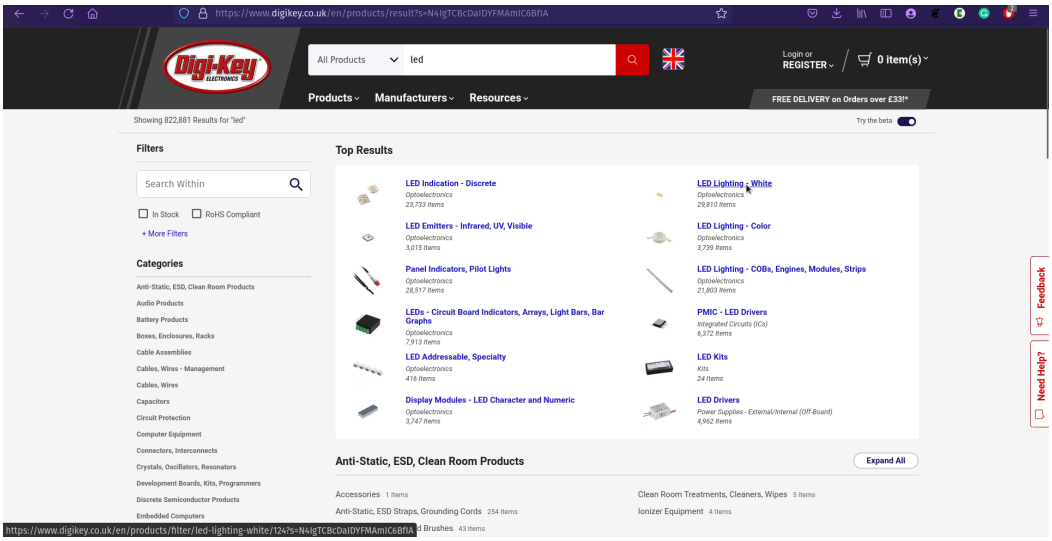


Figure 69: Digi-Key **LED** Section found

This opens a page with a lot of information, so it may initially be overwhelming. At the top, there are scrollable filters to see whats available specifically for our requirements. Below, there are checkboxes for specific but often important search criteria, with the parts shown below this.

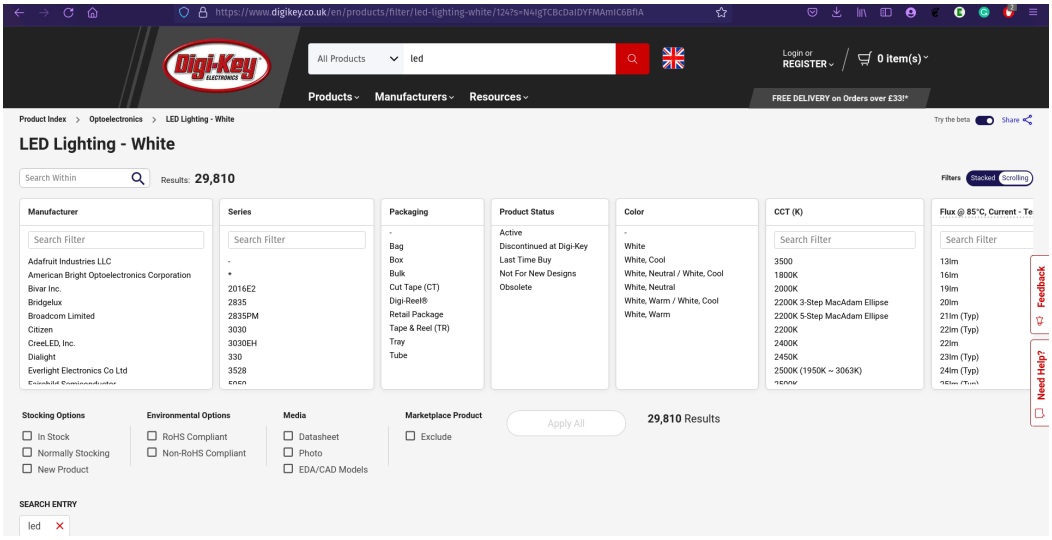


Figure 70: Digi-Key **LED** Page

On the top right of the criteria, I would recommend setting the filters to stacked, to see all filters at once, to go through each methodically.

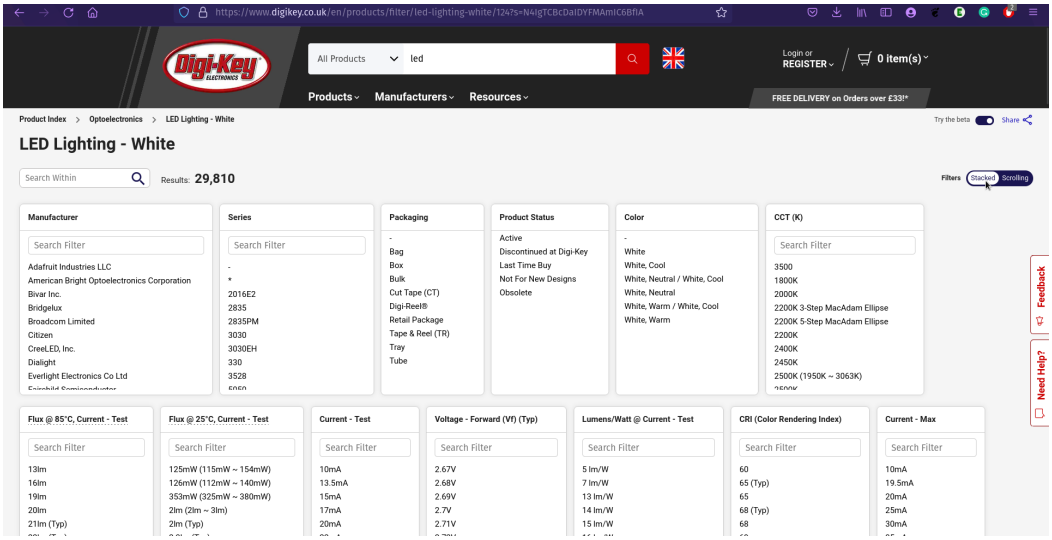


Figure 71: Digi-Key **LED** stacked filters

Many of these you will not need or may not understand, which is okay. You just need to scope out the specific characteristics which are important. Since some chips may have multiple **LED**s on one **Die**, firstly, it would be beneficial to find "**Voltage**- Forward", and select between the lowest number until 3.6V as a maximum, you can hold click to select a range, or click on the lowest boundary, and then scroll to find the highest suitable boundary and shift-click on it.

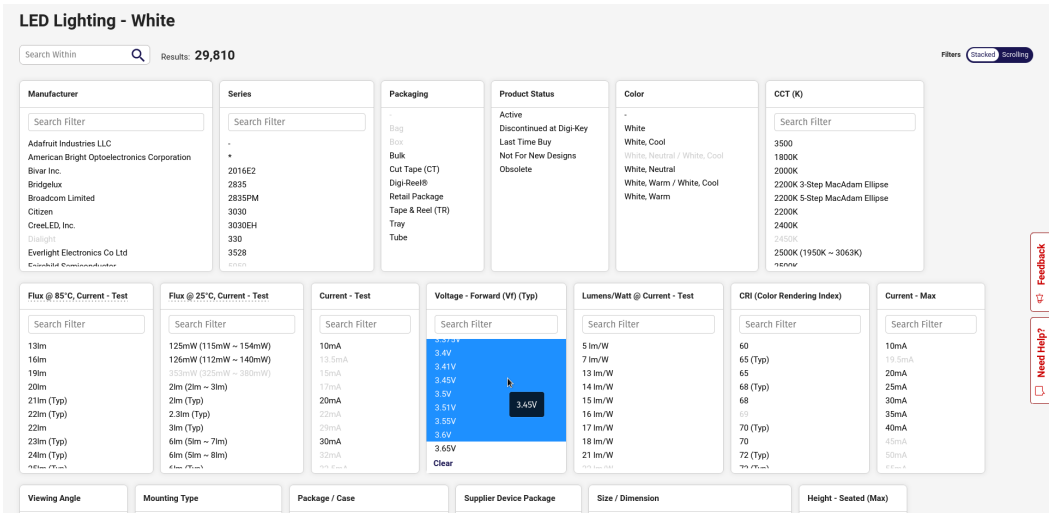


Figure 72: Digi-Key **LED Voltage** filter

Then the next suitable section looks to be maximum **Current**, since we will likely use a **Resistor** to limit **Current** anyway, the maximum could be higher than 25mA, since running a component at its max rated power can decrease its longevity. Therefore, we could select between the minimum value and 40mA.

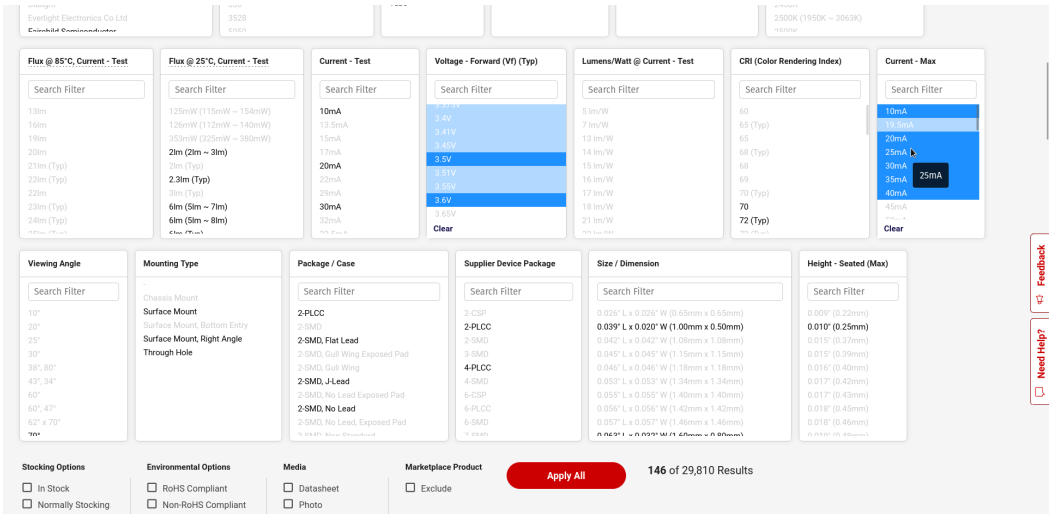


Figure 73: Digi-Key **LED Current** filter

We can then use "Package/Case", to look for the 1206 package size. You can then pick Apply Now. With the components shortages, it may be useful to select "In Stock", and also select **RoHS** compliant for environmental reasons.

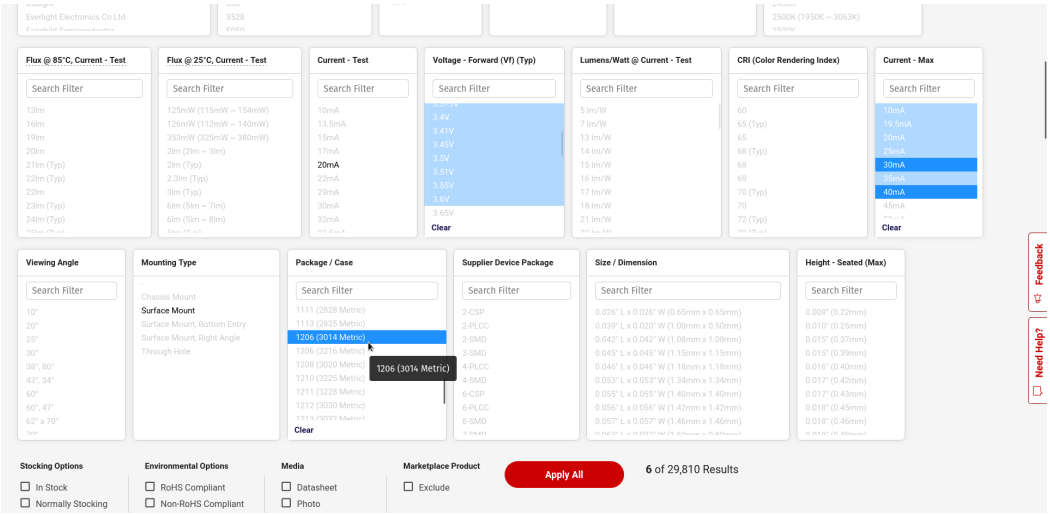


Figure 74: Digi-Key **LED** Package filter

We can now see the available options. You should see available stock, and cost in different quantities, along with other data, such as the CCT value for the colour, depending on the shade of white needed.

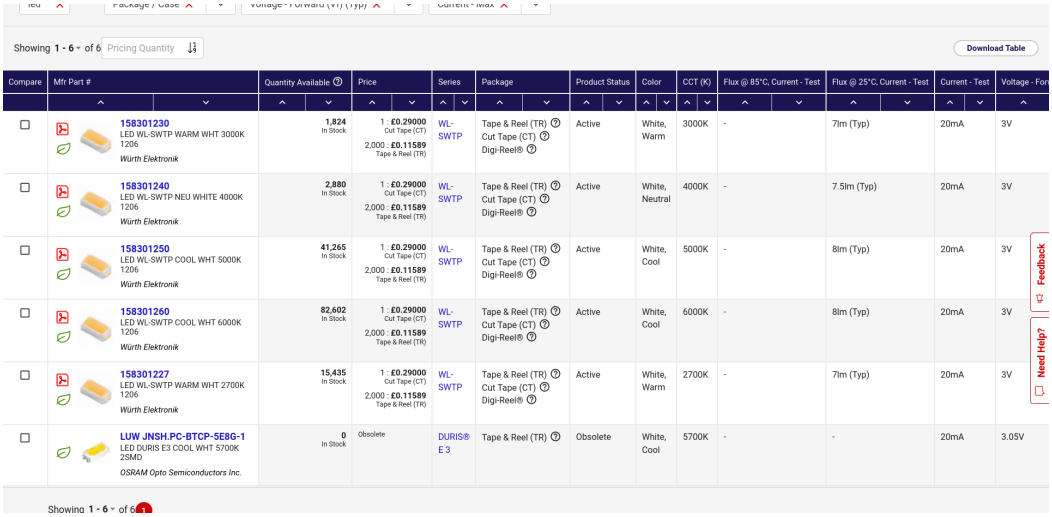


Figure 75: Digi-Key **LED** additional options

Although the stock number may seem a lot, for industrial customers producing many products, this might be a tiny amount, so selecting a part with plenty in stock if you feel you may need more in the future is important. I have now selected the final part.

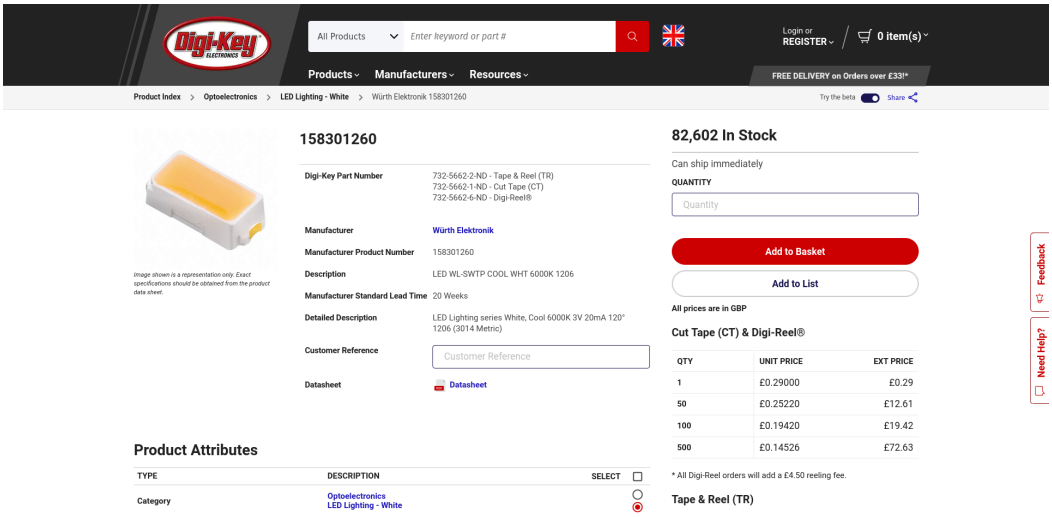


Figure 76: Digi-Key final **LED** found

To know what **Resistor** we may need to select, we should check the **Datasheet**. For an **LED**, you can find the forward **Voltage** vs forward **Current** graph. We can see at 25mA, the forward **Voltage** is around 3.25V. Therefore, we need to drop a **Voltage** of 0.35V in a **Resistor**. The ratio between the forward **Voltage** and **Resistor Voltage** is 9.286. Given that the **Resistance** of the **LED** will be $R = V/I = 3.25/0.025 = 130\Omega$. With this ratio, we need a **Resistor** of around 14Ω. Sometimes an exact **Resistance** value isn't available, so an alternate value may be required. You should ensure that it is as close as possible and if there isn't something close enough, you can use the **Parallel Resistor** formula/ use multiple **Resistor**s in **Series**.

As we have talked about needing a **Resistor** above, we can try to find one. Rather than having predetermined specifics, we can use the data derived from the **LED** requirements. There will be around 25mA through the **Resistor**, meaning power of around 0.00875W. As previously mentioned, it can be worth doubling this value to be clear of its max ratings, so we are looking for a **Resistor** of at least 0.0175W ratings. Here, I am going to say that we don't mind what size it is, but we want a cheap option. We want a **Resistance** as close 14Ω as possible, in this case a **Tolerance** of a maximum of 1% (Often 5% is used, for non precision applications). We have been told it must be a surface mount part. Luckily, in this case, **Resistance**, **Tolerance** and Power are next to each other.

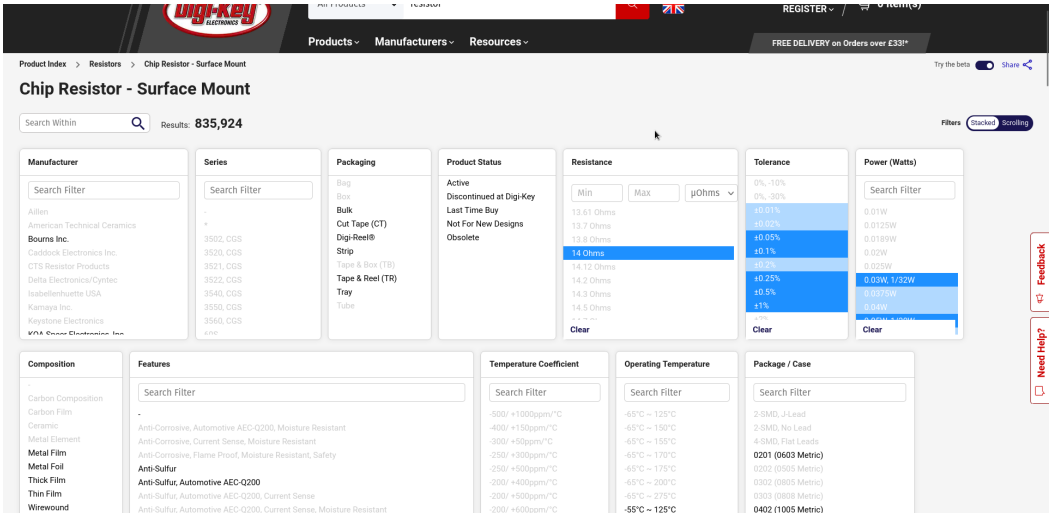


Figure 77: Digi-Key **Resistor** filters selected

We want to ensure its available and that it is at least 0603 for hand **Soldering**. A part is selected with plenty in stock, with all the ratings we would like, while being a cheaper part.

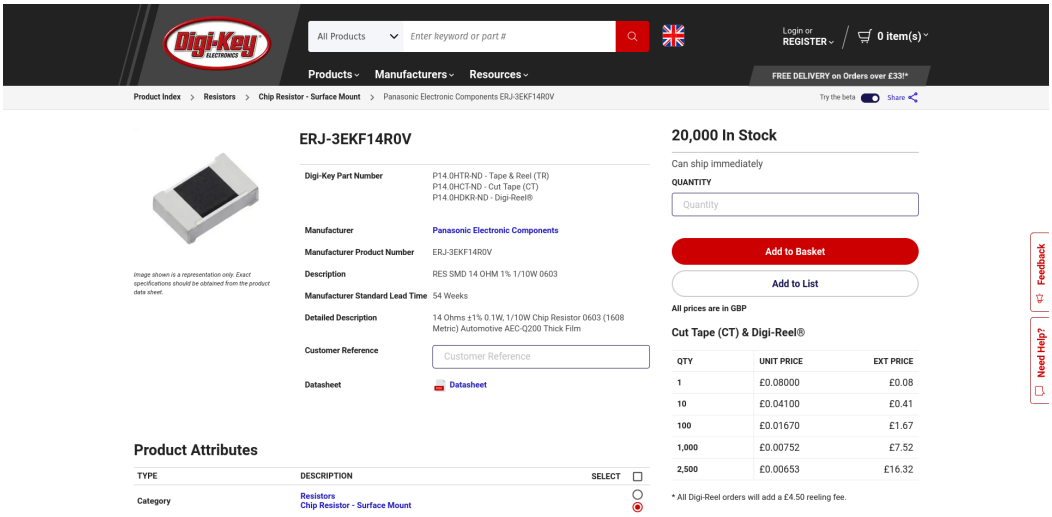


Figure 78: Digi-Key final **Resistor** found

You should remember that sites such as Digi-Key are not always the cheapest, with this **Resistor** costing £0.08. Although this sounds cheap, when hundreds of parts are required, the price quickly adds up. Sites such as LCSC have parts such as **Resistor**s for considerably cheaper, but has a worse search engine and filters, along with typically slower delivery times. You need to factor shipping costs into the overall costs, as well as import charges, often around 20%.

Now, a button suitable for the task should be found. It needs to take a **Current** of at least 25mA, and a **Voltage** of at least 3V. In this case, we are budget-constrained to under £0.20, so a tactile switch

would be the best option. A switch has been found, rated for 50mA, (again using a times two safety buffer), to account for plenty of unexpected deviance.

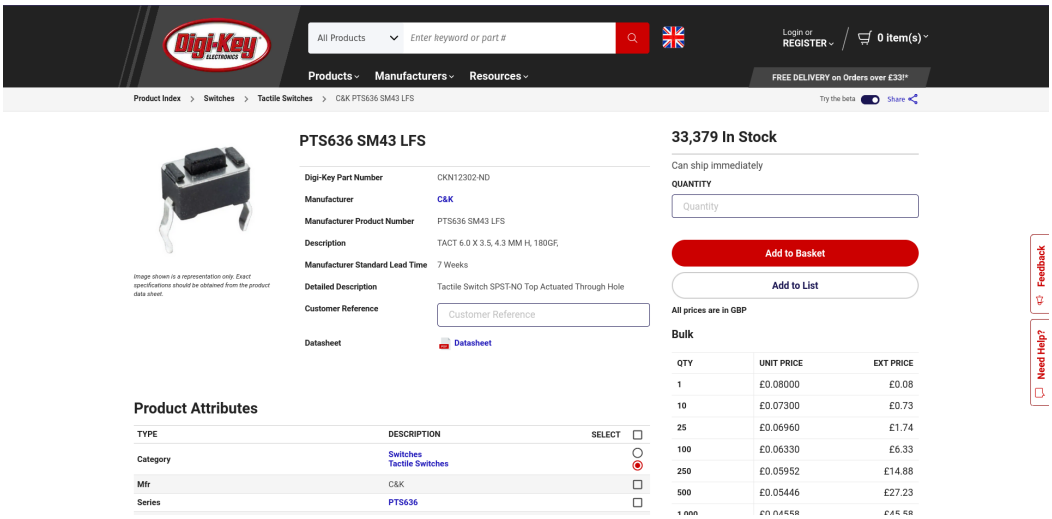


Figure 79: Digi-Key final Button found

We have now selected several parts, using just the criteria required. There are many other filters that can be selected. As already mentioned, Digi-Key has plenty of components on offer, and a good search function, but is certainly not the cheapest option. It can be used to get a better idea of what you are searching for, to then search elsewhere, such as Octopart using a specific part number.

This section quickly discusses some of the key factors and variables you should consider for a range of parts.

You should consider that power and **Current** ratings sometimes regards the peak power seen but sometimes the average. **Voltage** ratings usually relate to peak **Voltage**s.

Resistors- **Resistor**s are basic components, but still have many filters, with key ones touched upon now. **Resistance** is obviously key. **Tolerance** is the max percentage deviance with the **Resistance**, so lower is better. Power is its max power rating usually at ambient temperatures. The temperature coefficient is how much its **Resistance** can change due to a temperature change. Package or footprint is regarding its size and the specific footprint as mentioned in the **Datasheet**.

Capacitors- **Capacitor**s have a few similar characteristics such as **Tolerance** etc. Factors include ESR similar to **Impedance**, which is the internal **Resistance** at a given frequency, **Voltage** rating which is the max **Voltage** it can be charged to, ripple **Current** is the **Current** it can withstand at a given frequency and polarity, as some **Capacitor**s are restricted to a single direction for **Current** flow.

Inductors- These have a few of the same criteria already mentioned, a key criteria being its **Current** rating, which is the maximum **Current** the **Inductor** wire can take (constantly/ on average) and saturation **Current**, which is where the magnetic field is no longer proportional to **Current**.

0.9.6 Reading Datasheets

Sometimes going off filters or parameters of parts search engines isn't enough, particularly for integrated circuits where extra passive or active components are required for their operation. Therefore, this section quickly goes through looking through **Datasheet**s, with some specific part examples.

Like the section above, you should remember that there will be a lot of information in **Datasheet**s, a lot of which you might not understand, with the likelihood being you don't need to understand the vast majority of it. There is a skill though to be able to pick through all this, to find out what you do need to know.

Firstly, datasheets are not standardised, so you unfortunately can't just learn the layout for different parts. For a set part though, there should usually have similar information mentioned, which you just have to look for. The level of depth and focus of datasheets can vary quite significantly too. There tends to be quite a difference between ICs, discrete **Semiconductor**s and passives, so you can almost split them into several categories.

We will first though, go through things that should look similar in datasheets, and important bits of info to look for. Most datasheets would have max ratings, which will show the conditions that they will work in at max, these are not the recommended operating boundaries, but the point at which damage or instability/problems will happen if surpassed. You may not need to pay too much attention to the more specific sections, as some may link to others anyway. Ensure the max **Current**, **Voltage**s or power are surpassed, and that the temperatures and cooling would be sufficient. You should remember that data is usually not linear in these cases, so should not be extrapolated, as stated before, if you are unsure, using a significant safety buffer would be recommended, but certainly not that alone. You should find similar parts to have an idea of what max values there are for your specific operating conditions. A lot of datasheets have operating graphs which should help to visualise things.

ICs often have a fairly similar structure, with some of the following elements. These could include a general description, list of features, an example circuit, component pinout and pin descriptions, data signaling/ low level programming resources, footprint information, along with the maximum ratings.

The datasheets for passives can be quite large, often covering a whole range of parts, so searching for specific keyterms using Ctrl-F would likely to be beneficial. Realistically, for most passives, the supplier such as Digi-Key or LCSC should give you most the info you need to know through the parameters, without having to dig into the datasheet.

Discrete **Semiconductor**s often have information somewhere between a passive datasheet and an IC, such as more specific characteristics and graphs, but typically only a reference to their uses. Graphs can be particularly helpful for components like **LED**s, to figure out what **Voltage** and **Resistor** is required for sufficient operation at the intended brightness.

It would take a long time going through many datasheets to show you what to look for. It is handy to know that there are usually a range of in-depth guides for reading datasheets of specific components, often made by producers of those components.

0.9.7 Circuit Design Process

Now that you have seen many components, this section discusses some techniques, tips, and processes involved in circuit design.

It is important to first consider what your requirements are. What function does the circuit need to have, what environment should it be suitable for, what budget should it adhere to, what interaction does a user need to have with it, how easy will it be to make, can parts actually be sourced for it, and what restrictions are there regarding input power, how much processing power is required, how easy should it be to program, are there any niche **Libraries** needed for it. These factors will be split and discussed individually in some detail. Remember that it is hard to get a "perfect" circuit. There are many variables and factors at play. It is a balancing act of figuring out which factors are most important for the project.

As a first tip, I would recommend writing answers to the above questions, and then writing "nice to haves" for that factor, and what is needed for a "minimal viable product".

First, figuring out what function is required, will dictate what type of components are needed. It is worth seeing if there are existing circuits which do what you need or similar. An example could be checking for **Open-Source** designs which could be adapted for your specific use case.

Regarding what environment it needs to be suitable for, there are multiple factors. These include areas

such as does the enclosure need to be waterproof? Does there need to be a **Heatsink** or fan due to high expected temperatures? Does it need all **SMD** components to be low profile? Does a specialist circuit board need producing to adhere to certain standards? What fail-safes need to be integrated into the design?

Considering the budget, it is worth leaving a large safety buffer, as there needs to be extra in case there are issues in the first design. You need to factor in that multiple parts may need purchasing if there is a minimum order quantity. You should also consider whether the budget would be restrictive for what needs to be done. It is important to figure out whether saving money or saving time is more important, such as whether using **Module**s or custom designs would be better. There should be some wiggle room in the budget, to account for unseen factors.

You need to figure out what interface it needs for someone to use it. This could include considering whether it needs to transmit data through a wired or wireless **Protocol**? Does it store data on an SD card or inbuilt storage that needs connecting to a computer? Does it have a small display and buttons for setting it up, or is a computer or external device required?

Depending on its complexity, you need to consider whether an external service should solder the board, whether you do it by hand or whether a heat station should be used. This may make you focus on a specific design, such as using **DIP** components or focusing on **SMD** components.

A more important question than usual is, can components actually be sourced? There are tips elsewhere in this course on the matter, being a key parameter. Sometimes more novel approaches may need to be taken, if the typical route is not viable with parts shortages. Having a quick feel on stock levels for necessary parts is a worthwhile exercise before getting too far into a design.

It is important to be aware of how power needs may restrict the design, whether it is for mobile uses, with a restriction on how much power components can use, while ensuring that high-efficiency parts are selected, or whether there is only a specific **Voltage** input, so deciding what power conversion circuitry is required and viable.

Selecting what **Microcontroller** to use is important. Certain platforms, such as those compatible with the Arduino IDE can help to speed up design. This is due to a wide availability of **Libraries** and easier-to-use code. Within different platforms, there are **Microcontroller**s of varying performance, so understanding your needs regarding this is important. This could relate to whether an 8 bit, 16 bit or 32 bit **Microcontroller** is required, the latter of which should work for most end cases.

0.10 Further reading

This section provides reading into advanced topics. Although you may not make a circuit with the below, understanding them may be beneficial. The below ideas may help you to solidify your understanding of the key theory mentioned above.

0.10.1 Rectification

What is Rectification?

Rectification is the process of converting **AC** to **DC**, commonly used in power supplies, such as a phone or laptop charger. This is because **Digital** electronics require **DC** to function. Although diagnosis of **AC** circuits should not be done without the necessary training, understanding of how **AC** circuits function is important for safety and to better understand the use of components.

This circuit uses components which have already been discussed earlier in this document, so for further explanation, refer back to the appropriate sections.

For this **AC-DC** conversion, the alternating **Current** needs to be converted to flow in just one direction. Therefore, a key component in this process is a **Diode**.

The most basic form of **Rectifier** is a half-bridge (otherwise known as half-wave). This uses a single **Diode** in **Series** with the load, to limit **Current** flow to one direction. Therefore, the output wave is the positive side of the sinusoidal wave, with the average output **Voltage** around half of the **RMS Voltage**, although in a form unsuitable for most **DC** circuits.

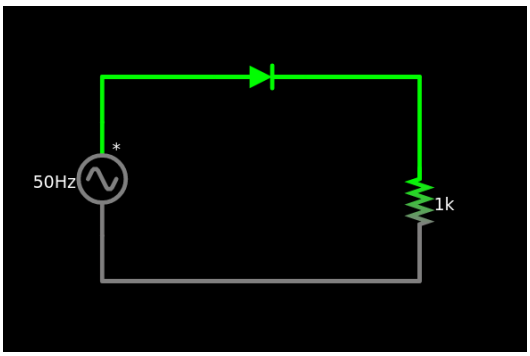


Figure 80: Example Half Bridge circuit

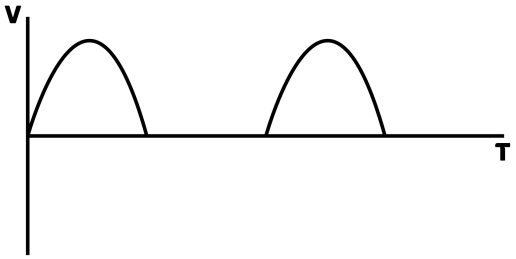


Figure 81: Example Half Bridge wave

A half-bridge circuit can be improved through use of filtering, with the addition of a **Capacitor** to smooth the output **Voltage**. This **Capacitor** is added in **Parallel**, where it is charged by the input, which it then outputs when the **Voltage** drops below its own **Voltage**. This enables an output **Voltage** closer to a smooth **DC Voltage**. As previously mentioned, even if perfectly smoothed, the max **Voltage** that could be attained is half the **RMS Voltage**.

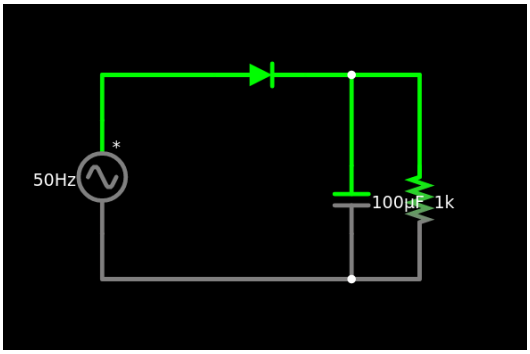


Figure 82: Example Half Bridge circuit smoothed

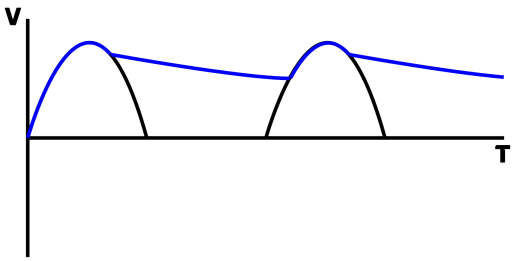


Figure 83: Example Half Bridge wave smoothed

A full-bridge **Rectifier** is a further improvement. These use four **Diode**s, meaning that whichever way

the **Current** flows at the input, it will be fixed at the output. Therefore, there will be a sinusoidal **Voltage** magnitude, but only in the positive axis. This can again be smoothed with a **Capacitor**, and if efficiently smoothed, will have an output close to **RMS**.

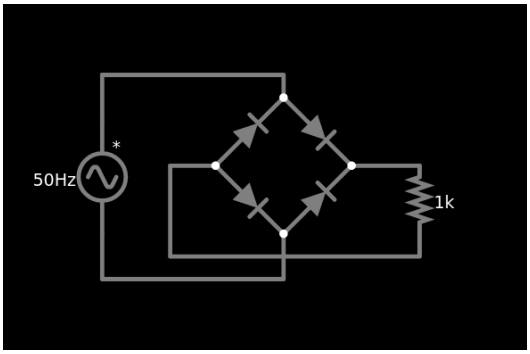


Figure 84: Example Full Bridge circuit

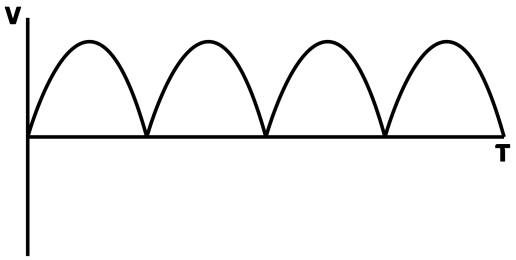


Figure 85: Example Full Bridge wave

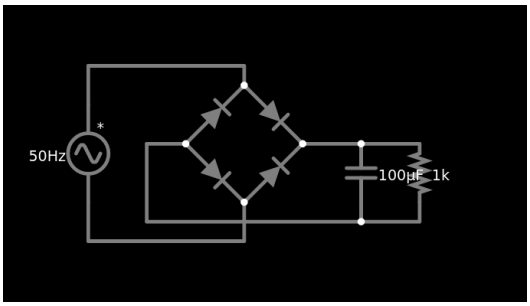


Figure 86: Example Full Bridge circuit smoothed

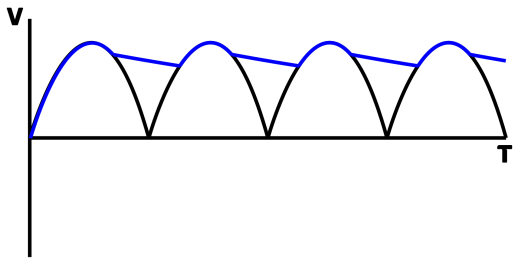


Figure 87: Example Full Bridge wave smoothed

Example circuits, applications and considerations

This section will quickly look at applications of the above mentioned circuits.

Rectification circuits are widely used, especially for **AC-DC** power supplies, as an initial stage to convert **AC** to **DC**. Therefore, their main use is within mains connected power supplies. Another example is a dynamo torch, where the **AC** power from the **Generator** needs rectified to power an **LED**.

It is likely that in most cases a full-bridge **Rectifier** would be used rather than a half-bridge **Rectifier**, unless initial costs were a major consideration.

Because of variance in the manufacturing process, even within a single product, it is commonly considered best practice to use a full-bridge **Diode** package, where there are four equal **Diode**s within one part.

Rectifiers are generally used as part of a more complex power system, usually as an earlier stage.

Simulations

Below are several simulations to have an experiment with to help your understanding of **Rectification**.

Half-Bridge **Rectifier** at 230V **RMS**, 50Hz- **Series Link**

Half-Bridge **Rectifier** at 230V **RMS**, 50hz with capacitive smoothing- **Series Link**

Full-bridge **Rectifier** at 230V **RMS**, 50Hz without smoothing- **Series Link**

Full-bridge **Rectifier** at 230V **RMS**, 50Hz with capacitive smoothing- **Series Link**

Quiz

1. What is the main benefit of a full-bridge **Rectifier** over a half-bridge **Rectifier**.

a) It is cheaper to build.

b) It allows for full utilisation of the input **Voltage**.

c) It uses fewer components.

2. Why are smoothing **Capacitor**s useful? (There may be more than one right answer)

a) They allow for a more constant output **Voltage**.

b) They increase average power output.

c) They reduce output ripple.

3. True or False, it is known to be better to use several **Diode**s individually rather than packaged together as a full bridge **Rectifier** package.

4. True or False, **Rectifier**s can be used to also convert **DC** to **AC**.

Challenge- Using Falstad or hand drawn, design a full-bridge **Rectifier** with **Voltage** variance/ ripple lower than 10%, working with an **AC Voltage** of a peak value of 326V, at 50Hz.

0.10.2 Gate Driving

What is Gate Driving?

Gate driving is necessary for using **MOSFET**s in high-speed circuits. As mentioned earlier, **MOSFET Gate**s have an inherent **Capacitance** which requires charging in order for the **Gate** to "open". Therefore, for fast switching, it may require large amounts of **Current**, which a **Microcontroller** usually can't supply. Therefore, there are integrated circuits with common methods to allow for these to be charged safely. A common simpler methods is called a Totem Pole **Gate Driver**. This uses two **BJT Transistor**s, one used to charge and the other to discharge the **Gate**. **BJT**s are used because of their simplicity and ability to amplify the input **Current** using an external source. Even then, it is important to use a **Resistor** to limit the **Gate**s charge **Current**.

This section references a range of components and knowledge from the main sections of the course, to help consolidate past theory.

Here is a circuit of it in action, feel free to play around with it- **Series Link**

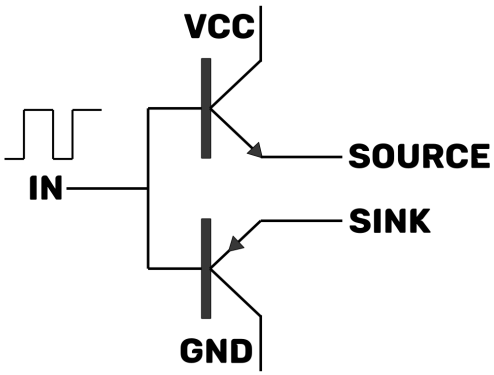


Figure 88: Totem Pole Gate **Driver** representation

The method shown above isn't the only viable option, but is one which is widely used and one of the simpler methods to understand.

0.10.3 555 Timers

555 timers are adaptable ICs, for a range of timing techniques, that may not require a whole **Microcontroller**. Although understanding the internal structure of a 555 timer is outside the course scope, understanding how and why they are used could be beneficial. Depending on how it is connected to, it can output waveforms of different frequencies, or as a timer.

0.10.4 Motors and Generators

Motors and **Generator**s use a set of physics principles in their operation. **Motor**s convert electrical energy to kinetic (typically rotational), where **Generator**s do the reverse.

They operate due to the **Motor** effect which is the following. A **Current** carrying conductor will induce a magnetic force, which will interact with other magnetic forces. If the conductor is free to move, it will. Therefore **Motor**s are designed using both **Electromagnet**s and permenant magnets with interacting magnetic fields to cause rotational movement.

The reverse is also true, when a conductor moves through a magnetic field, a **Current** is induced. A large proportion of electricity is generated using a **Generator**, where usually steam from either a combustion process of oil, gas or coal, or from nuclear fission, causes the evaporation of steam, the movement of which causes a **Generator** to turn, thus generating electricity.

0.10.5 Renewable Electricity

Renewable Electricity is of increasing importance, as we look to reduce our carbon footprint. This section quickly looks into different methods, and how they work.

It is important to consider that all renewable energy sources have a carbon footprint, and this can vary significantly from manufacturer to manufacturer. As shown **Series here**, how sustainable a manufacturer is can vary considerably, so it would be better to select panels from a manufacturer at the top of a list such as this.

Solar panels are **Diode**s, working like a **Generator** (with the photovoltaic effect), where solar radiation induces a **Current** and therefore electrical energy. These are produced using semiconducting material, such as **Doped** silicon.

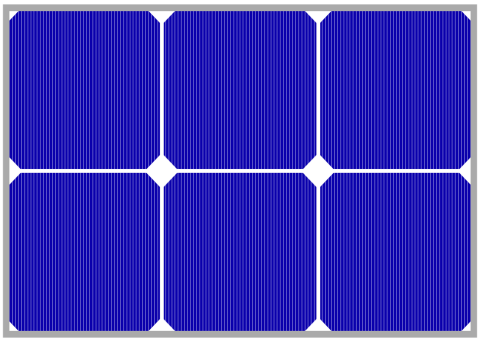


Figure 89: Example Solar Panel diagram

Most other renewable methods involve a **Motor** acting as a **Generator**. Whether it is wind power through wind turbines, where blades are connected to a **Motor** to generate wind power, or hydroelectric, where flow of **Current** against the sea or through a dam spins rotors. Hydroelectric, especially using dams has a mixed record, because of its fairly significant impact upon the environment and nature. Geothermal uses heat from within the ground to heat up steam, to push rotors of a **Generator**.

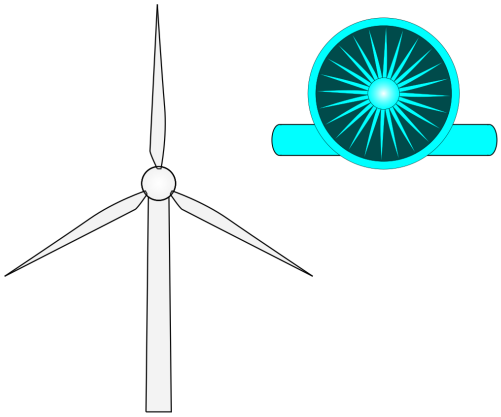


Figure 90: Example Wind and Water Turbines

0.10.6 PCB Design Basics

Although this isn't in depth **PCB** design, it is a few useful to know tips.

The cost of custom **PCB** production has reduced considerably due to the increase in prototype **PCB** services. JLCPCB was one of the first low cost options. The reason their model is viable is that they use excess board space, left from larger customers, so most costs are due to the set-up, rather than the material costs. It also means that it is reducing waste, and possibly carbon footprint.

A couple of free **PCB** design tools include EasyEDA and KiCAD. If you prefer **Open-Source** software, KiCAD is a good option, otherwise, EasyEDA is pretty easy to get started on, is free for even its advanced features, and has links to LCSC and JLCPCB to get you going quicker. Lots of time can be spent finding suitable package footprints or designing footprints. EasyEDA simplifies things, with most parts in LCSC's library being available to search for and drop into the design. Even if it is an external part, there are many community part footprints and **Schematic** designs. For community derived parts, it is worth reading the **Datasheet** to ensure it matches up with that design. Most **Datasheet**s should provide information on suitable **PCB** footprint design.

0.11 Resources

There are many electronics resources for further learning. Particularly useful resources are referenced below, with their purpose.

0.11.1 Manufacturers

It is handy to know reputable manufacturers of components and **Module**s. This is not definitive, but provides a trustworthy list. This focuses on larger manufacturers, but there are many more trusted manufacturers. Keep in mind that there are fake components in the market, so ensure you source parts from a trusted supplier, talked about in the section below! As touched upon in the finding components section, most parts through a site such as Digi-Key should be reliable.

Resistors- Yageo, BOURNS, Panasonic, ROHM, Samsung, TE Connectivity, Vishay, Walsin.

Capacitors- Rubycon, Panasonic, Hitachi, Sanyo, Vishay, Nichicon, NCC, Wurth, Illinois **Capacitor**, Cornell Dubilier.

Inductors- Wurth, Walsin, Vishay, TDK, TE Connectivity, Pulse Elec, Murata Electronics, BOURNS, EATON.

Transistors- Vishay, TOSHIBA, Texas Instruments, STMicroelectronics, ROHM, RENESAS, onsemi, NXP Semicon, Nexperia, Infineon, **Diode**s Incorporated.

Diodes- Vishay, TOSHIBA, STMicroelectronics, ROHM, onsemi, Nexperia, Microchip Tech, Infineon, **Diode**s Incorporated.

Sensors- Infineon, Analog Devices, OMRON, Texas Instruments, TE Connectivity, NXP, Microchip Technology, Murata, BOSCH, STMicroelectronics.

Microcontrollers- Microchip, ATMEL/AVR, NXP, STMicroelectronics, Espressif Systems, Infineon, Altera, Raspberry Pi, RENESAS.

ICs- Texas Instruments

0.11.2 Suppliers

Starting with several trusted parts suppliers.

Series Arrow Electronics- Europe based electronics components shop.

Series RS Components- UK based electrical and industrial components shop.

Series Mouser- US based electronics components shop.

Series Farnell/Series CPC- Electronics components shop based in several regions.

Series LCSC- Electronics components shop based in China, with links to JLCPCB and EasyEDA, offering lower costs than most other suppliers for larger orders.

Series Digi-Key

Series Rapid- UK based electrical and industrial components shop.

Series EuroCircuits- Lower cost **PCB** manufacturer based in Europe.

Series JLCPCB- Low cost **PCB** manufacturer based in China

Series Aisler- Lower cost **PCB** manufacturer based in Europe.

Most the above focus on singular components, which may not be what you are looking for. If you are looking more for **Module**s or DIY components, here are a few trusted retailers.

Series HobbyTronics- UK Based hobbyist electronics shop.

Series Bitsbox- UK Based hobbyist electronics shop.

Series Adafruit- US Based company producing a range of boards/ **Module**s.

Series Sparkfun- US Based company producing a range of boards/ **Module**s.

Series Pimoroni- UK Based **Module** maker and hobbyist electronics shop.

There are a couple of sites used to find stock of components across suppliers, while comparing their costs. Within these tools, there are multiple suppliers mentioned, a few of which being resellers, where you may pay more, but they often have parts in tight supply.

These could be thought of Electronic Part Search Engines.

Series Octopart- Components stock checking and Datasheet search engine with BOM tool

Series Findchips- Components stock checking and comparison tool

Series OEMSecrets- Components stock checking and Datasheet search engine with BOM tool

Series EasyBom- Components stock checking and comparison tool

0.11.3 Electronics Software

There are several bits of electronics software mentioned below, whether it is for simulation or **PCB** design.

Series Falstad- Free online electronics simulation program, which is easy to visualise what is happening in a circuit.

Series LTSpice- Free and powerful electronics simulator, although slightly dated presentation, produced by LT.

Series TinkerCAD- Simple online software for basic 3D design and electronics design/ simulation.

Series MicroCap 12- Although it can be seen as abandonware, it is now free, relatively recent and a powerful piece of software.

Series KiCAD- Free and **Open-Source PCB** design package

Series EasyEDA- Free and easy-to-use **PCB** Design Package with links to both LCSC and JLCPCB.

subsectionFurther learning sites

Another helpful resource could be additional places to go to learn further.

Series Allaboutcircuits

Series Hackaday

Series Hackster.io

There are a range of helpful YouTube channels, such as the below:

Series GreatScott- Page with many tutorials and projects, in plenty of detail, although it can be complex for beginners. If you like learning through a project-based ideas, it's worth checking out.

Series Electronoobs- Another project based channel, teaching you a range of things.

Series EEVBlog- One of the biggest electronics video blogs, covering a range of areas, from an Electronics Engineer. Known to be unscripted, and "off-the-cuff", but informative.

Series Adafruit Industries- Channel set up by Adafruit, discussing electronics and their **Module**s.

Series Element 14 Presents- A channel with a wide range of presenters doing electronics and engineering projects

0.12 Necessary software

In this document, you are encouraged to check simulations through Falstad. Falstad is a web-based, simple but powerful circuit simulation program. It was designed by Paul Falstad and is free to use.

Selected because of its ease of use and ability to represent circuit information clearly and helpfully.

This can be found through the following link- **Series Link**

It provides a visual and easy way to represent circuits, with the ability to plot **Voltage** vs **Current** for set positions or components. There is a vast array of sample circuits to look at for reference. The simulations mentioned in this course use this software, so a brief understanding of how it works would be useful. Each component has nodes/ squares which other components connect to. Components are found in the draw section, and you can click and drag to alter their size.

LabVIEW is required for the practical part of this course. LabVIEW is a visual programming platform, intended for test and measurement, selected as a suitable program for testing out skills developed through the course. This need to be installed, using the Universities licence, the steps of which can be found here- **Series Link**

LabVIEW will be used to interface with an Arduino, rather than using a programming environment such as the Arduino IDE. It is useful to mention that platforms such as this are often used to program **Microcontroller**s, with Arduino IDE based on C++. The package for interfacing requires Windows specific **Libraries** to run, although has been tested to work on a Windows 10 virtual machine.

We will discuss the steps to install LabVIEW, Arduino IDE and the LabVIEW LINX package. Although we will mostly be graphically programming through LabVIEW, it is useful to know Arduino programming basics, especially if you need to use use specific parts or scripts which the LINX package doesn't integrate.

Step 1- Go onto **Series Link** Go onto the "Obtaining the software" tab, and follow the first link. As shown, you first need to create an account with National Instruments. Ensure that the email used is your University email.

Once this has been done, follow the next link to download the software. You want to be installing the most recent version, at the time of writing this was 2021 SP1. It is the "Base, Full, Professional" included editions version you require, as it is a Professional licence included from the University. If possible, you should install the 64-bit version, with **Driver** software included. Then on the right, you

can click on the download button. This will install the package manager which allows you to select what packages are needed. The default ticked packages are all you should need, at least initially, such as LabVIEW under programming environments, some of the ticked **Driver**s are necessary for the LINX package, so leave them as is. You then need to agree to install them and then follow the steps. Once it is all installed, you will need to activate it, which is where the third link from the university page comes in. After clicking that link, you may first be prompted to sign in with your university details. Scroll down to the bottom of the page, where it says "Licence keys". You should select a link depending on whether you are using a University-owned computer or your own. The code shown is a **Serial** code so you need to use that option. You can copy and paste the code into all of the below. Once you press continue, it will state that 4 should have been verified, which is all that needs to be, as it is the Professional version, not base or full etc which the university use. You can then press done, and the software is installed! The only specific package you need to install on top is the LINX package, which allows an interface between an Arduino (as well as other **Microcontroller**s and small board computers) with LabVIEW.

It is pretty simple to get LINX installed, you have to use the VI package manager. It will have to install a few things first, which you have to wait for. Once these have been done, you can use the search bar in the top right, to search for LINX. It should be the top option, which you can then click and press install, accepting the stuff that comes up. Now you have the main things needed for running the software!

We will quickly go over the steps to install Arduino IDE now. You should go to this **Series Link**, and click on whichever download option on the right suits best. You should then run and install it, accepting any **Driver** installs which are necessary. Once it is installed, you can plug the Arduino in and see if it can be read. If you go onto the toolbar, onto tools, you should be able to go onto board, and then scroll to Arduino AVR boards and then to Arduino Uno. Just below board, there should be Port, which there should be an option available either with the name Arduino Uno or COM followed by a number. This shows that it has successfully been read by the computer.

0.13 Practical work

This section puts the theory which has been learnt into practice, focusing on using LabVIEW along with the LINX package, to interface with an Arduino and any components connected to it.

0.13.1 LabVIEW Examples

Setting up LabVIEW with an Arduino

We will first go through the initial steps to set up an Arduino with LabVIEW. LabVIEW uses a graphical interface/ code which you design to create code which runs on an Arduino. This needs firmware to be set up on it first.

If you open LabVIEW and create a new project, under the name Blank VI.

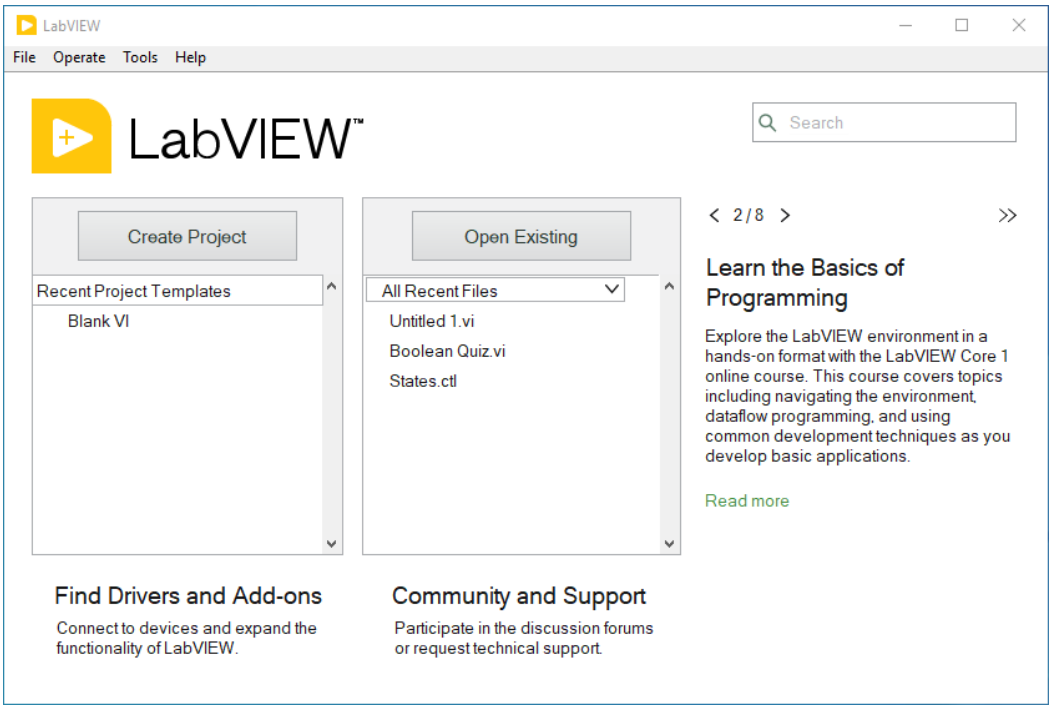


Figure 91: LabVIEW Start Up Page

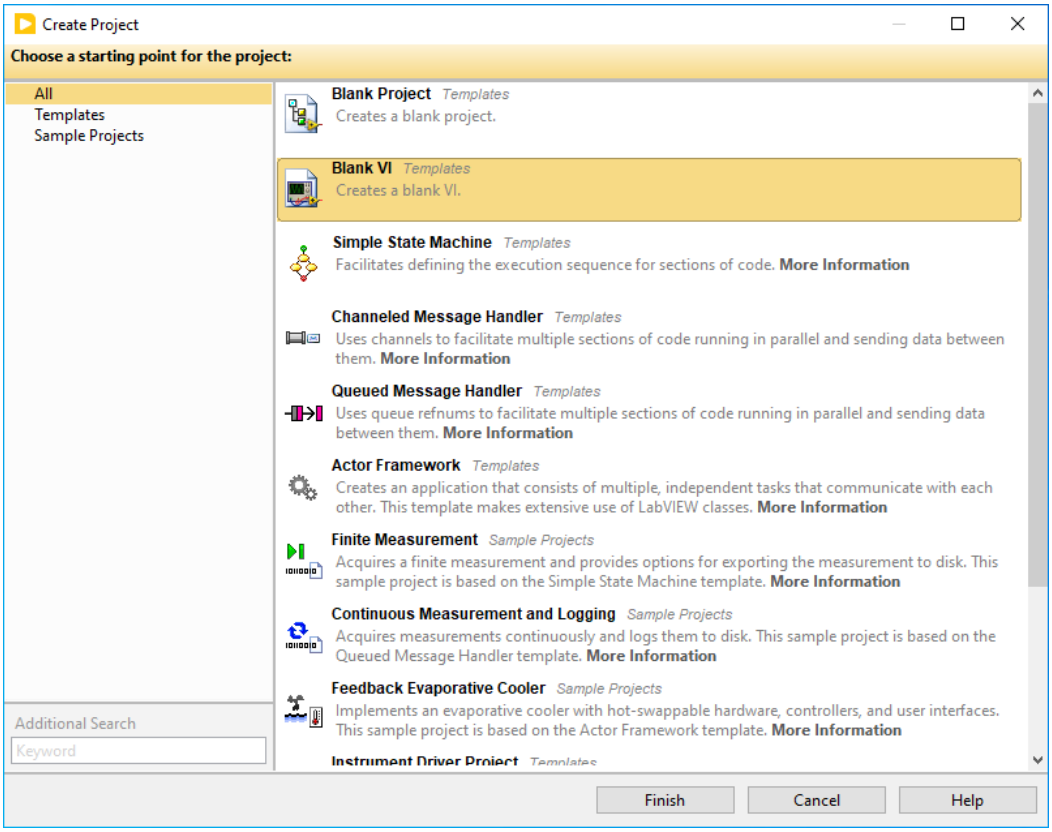


Figure 92: LabVIEW New Project Select

Once this is open, there should be a front panel and block diagram which opens (You can use either CTRL+E or the toolbar, then "Window", then "Show Block Diagram" if the block diagram doesn't show).

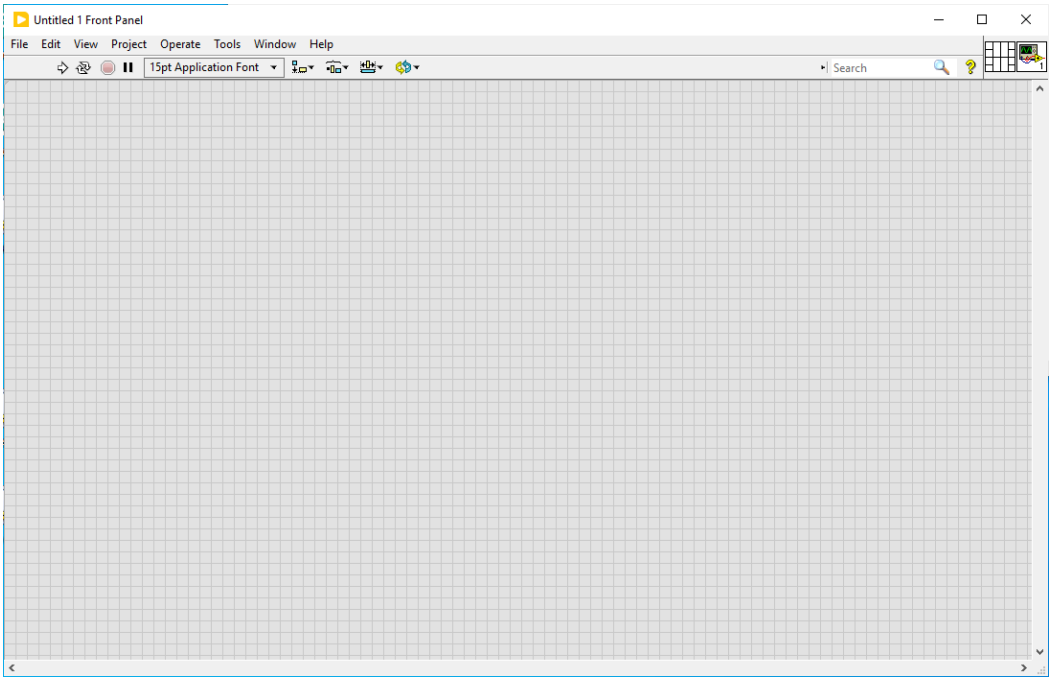


Figure 93: LabVIEW empty Front Panel

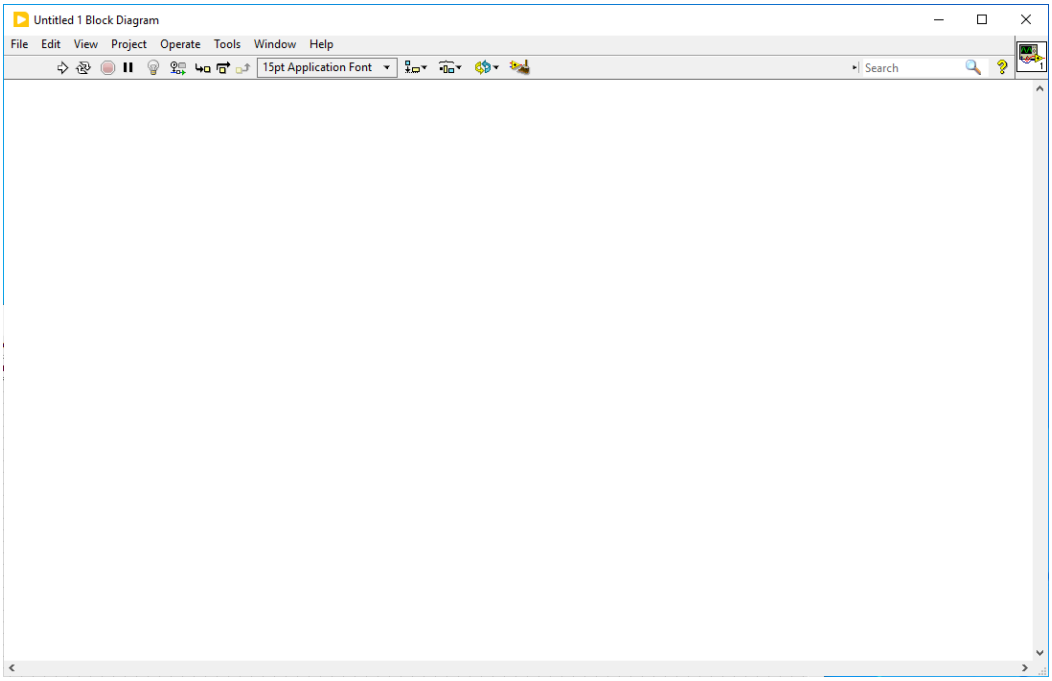


Figure 94: LabVIEW empty Block Diagram

The front panel is what a user interfaces with, in the final program, such as buttons to turn on and off a heater, or a dial to choose the temperature of a heater. The block diagram is where the visual programming takes place. When you add objects to the front panel, they also appear on the block diagram to use.

With the very basics discussed, we will now use the LINX firmware wizard to generate the firmware for the Arduino. Go to the "Tools" section of the toolbar, and then move your mouse to MakerHub. After this, go to "LINX" and then "LINX Firmware Wizard".

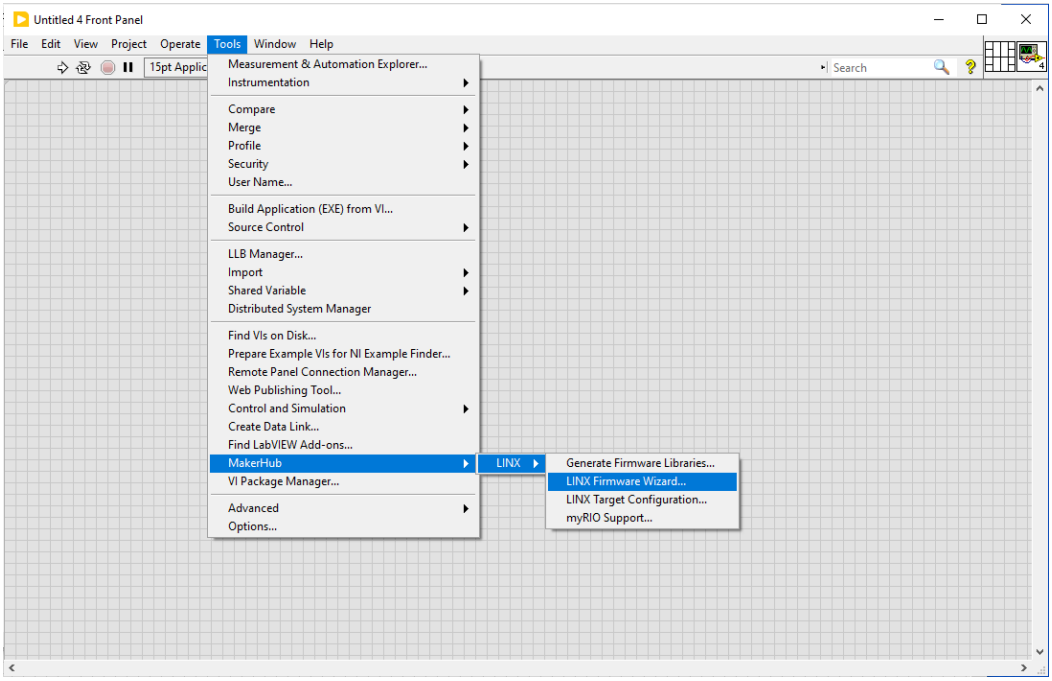


Figure 95: LabVIEW LINX Firmware Setup Find

There should then be a pop-up. You should ensure that "Arduino" is selected as the "Device Family", then for "Device Type", you want to select "Arduino Uno" and the "**Serial / USB**" "Firmware Upload Method".

Figure 96: LabVIEW LINX Firmware Setup Device

You can then click on "Next". You should next select the COM port, in my case it is COM3.

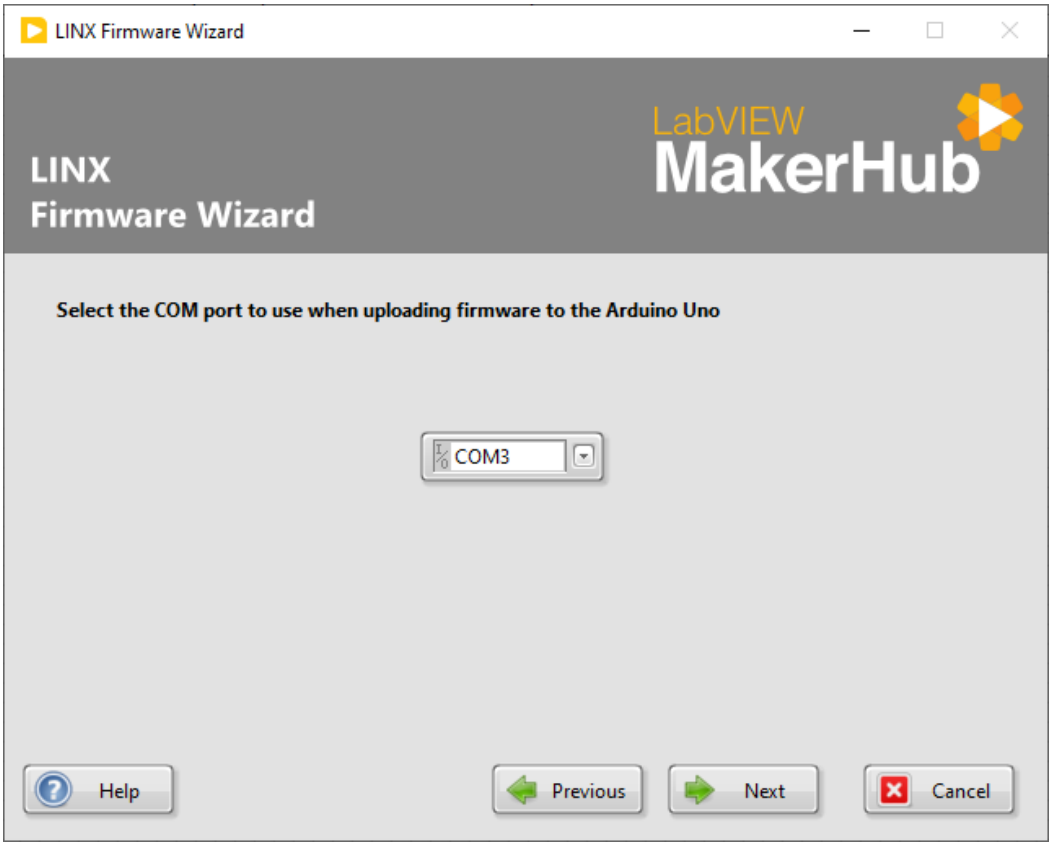


Figure 97: LabVIEW LINX Firmware Port

The "Firmware Version" should be "LINX - **Serial / USB**", and you can select "Pre-Built Firmware" for "Upload Type".

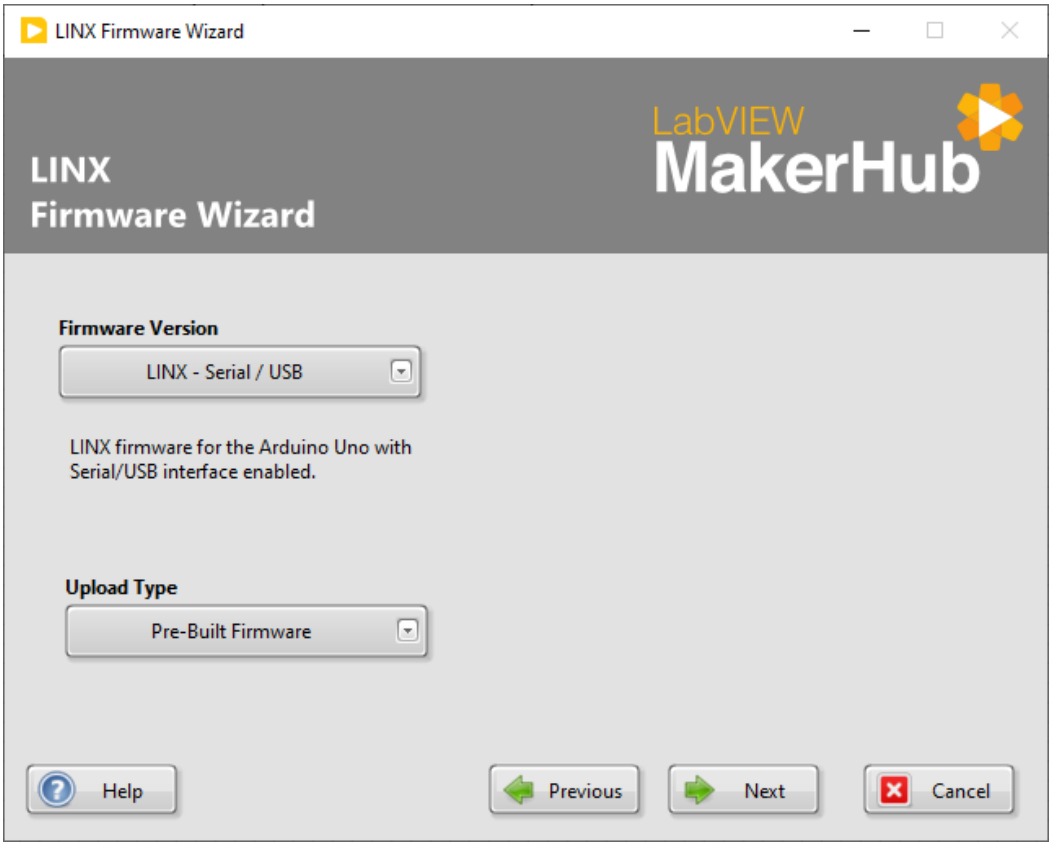


Figure 98: LabVIEW Linx Firmware Upload Type

It should then go through the stages of uploading this firmware.

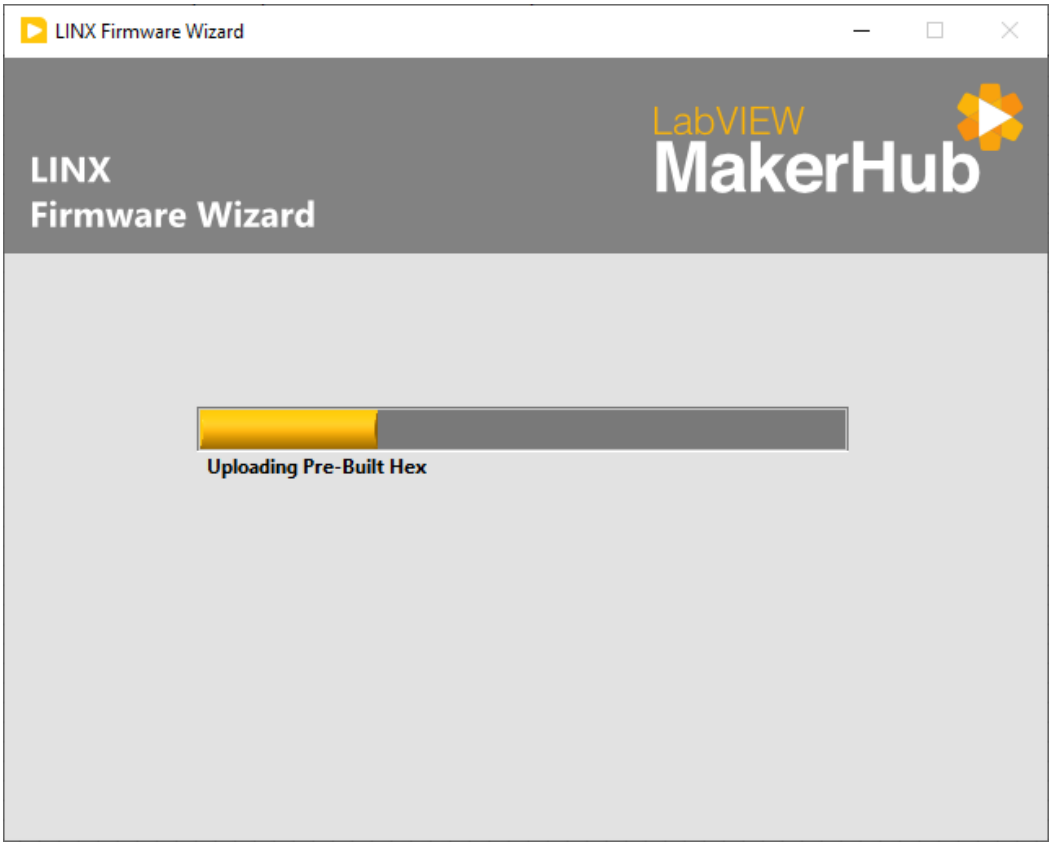


Figure 99: LabVIEW LINX Firmware Uploading

You can then press finish, or Launch Example. We will view this example, as it would be good to explain some of the basics of LabVIEW and the extras that the LINX package provides.

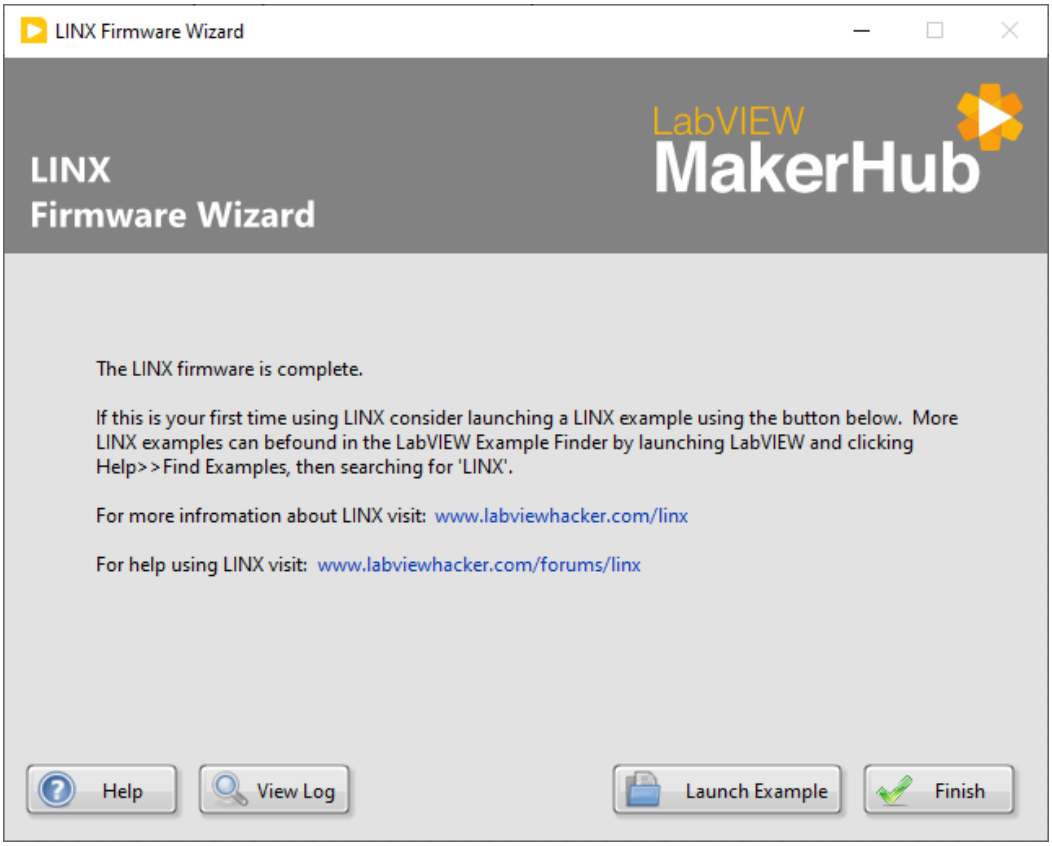


Figure 100: LabVIEW LINX Firmware successful Upload

The example is a manual blink, where you can click on the button on the right of the front panel to control the turn on and off of an **LED**. We can run this without using an external **LED**, by using the built-in **LED**, linked with Pin 13. Therefore, for **Digital** output channel, you can type in 13. You should also select the **Serial Port** to which the Arduino is connected to. Then press the run arrow in the top left. You should make sure to press the stop button before closing it, as otherwise there can be issues regarding reconnecting to the Arduino.

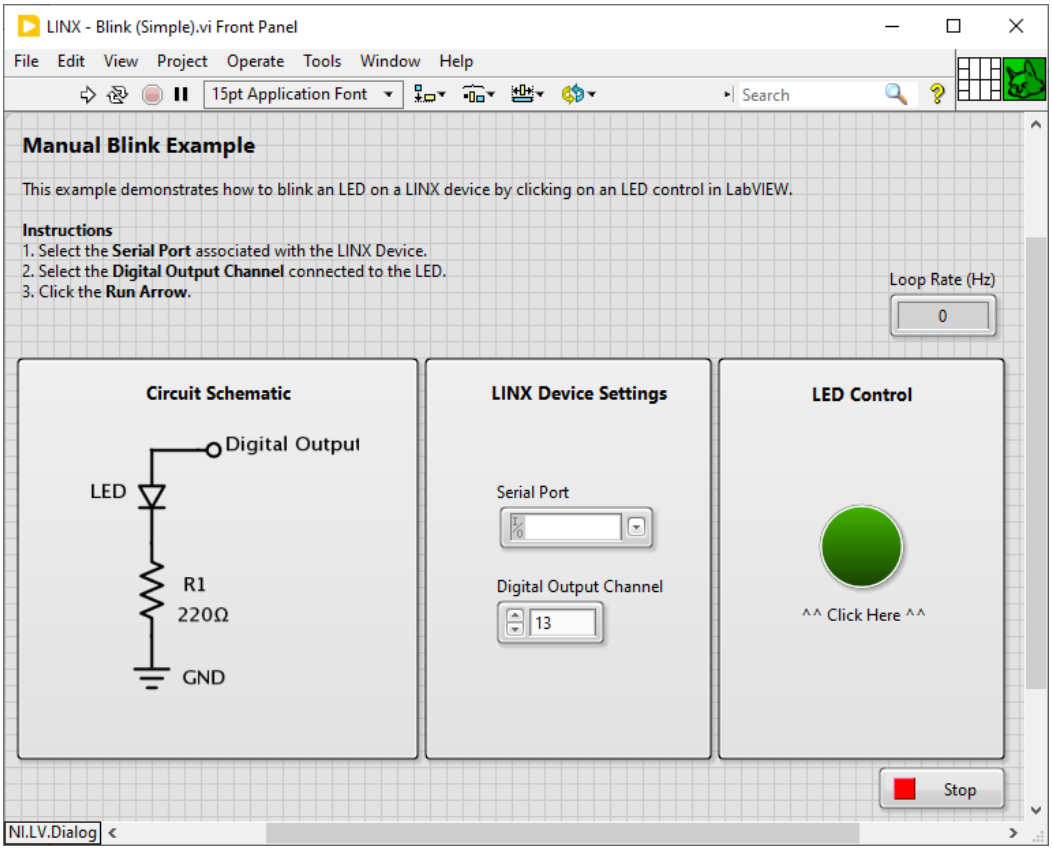


Figure 101: LabVIEW LINX Example Program

You should see the RX and TX **LED**s light up, which indicates that the **Serial** bus is being used to transfer data both ways. After a few seconds, you should be able to turn on and off the **LED** with button presses.

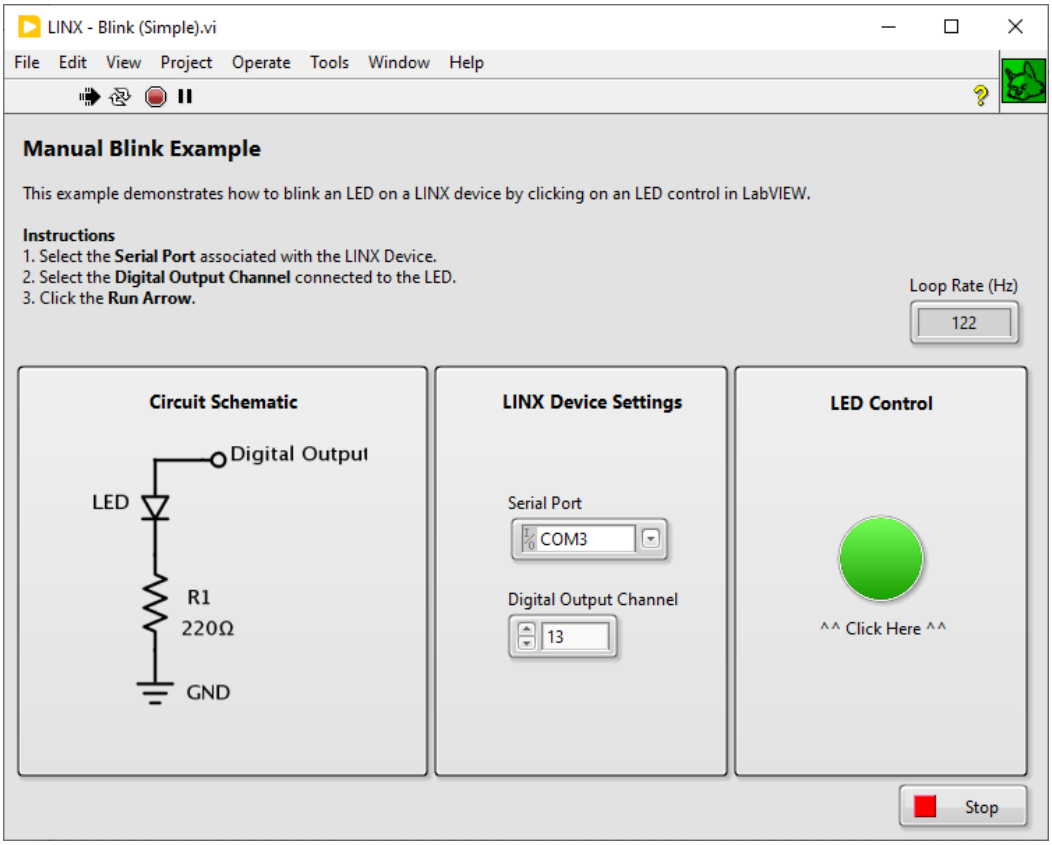


Figure 102: LabVIEW LINX Example Program in use

We will now talk through some steps in the design of a LabVIEW design. As stated, LabVIEW provides a graphical programming interface through the block diagram, as well as a front panel meant for input and output, both of which will be used. We will do this by taking you through an example, to run yourself. A good first exercise is to interface with an **LED**, similar to the above but using a couple more advanced features.

We can use the new project which was created for the purpose of setting up the Arduino.

It is worth first setting up the basic back-end elements, that you know your project requires, along with the necessary inputs and outputs. For a project which interfaces with an Arduino, LINX start and close blocks are needed.

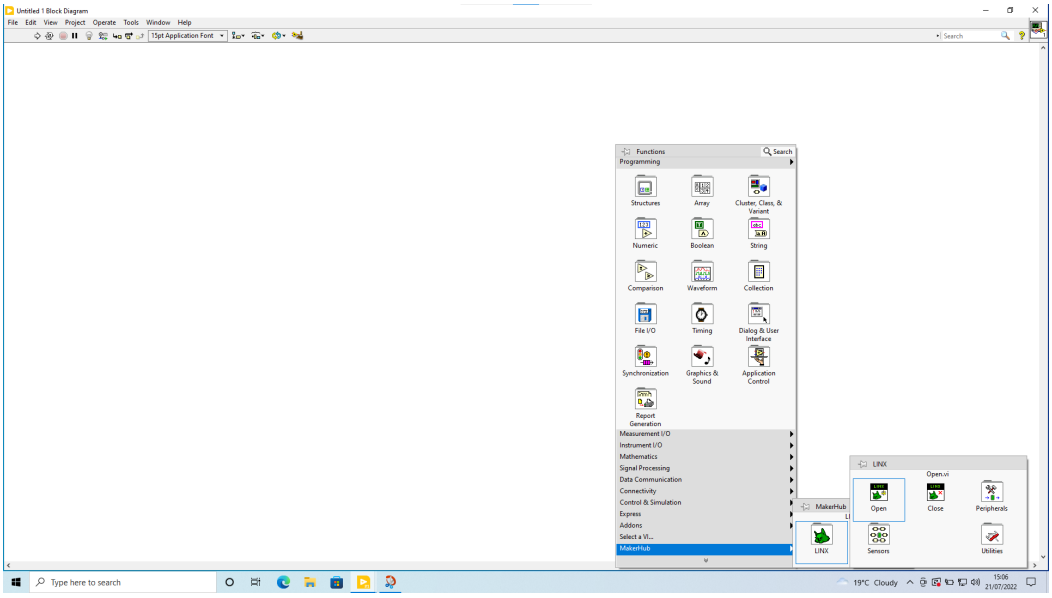


Figure 103: LabVIEW LINX Block select

The Start Block requires a **Serial** input, corresponding to the **Port** used for communication between the computer and the Arduino. The Stop Block's close function should run to ensure the program stops correctly and should be an endpoint after all other potential processes.

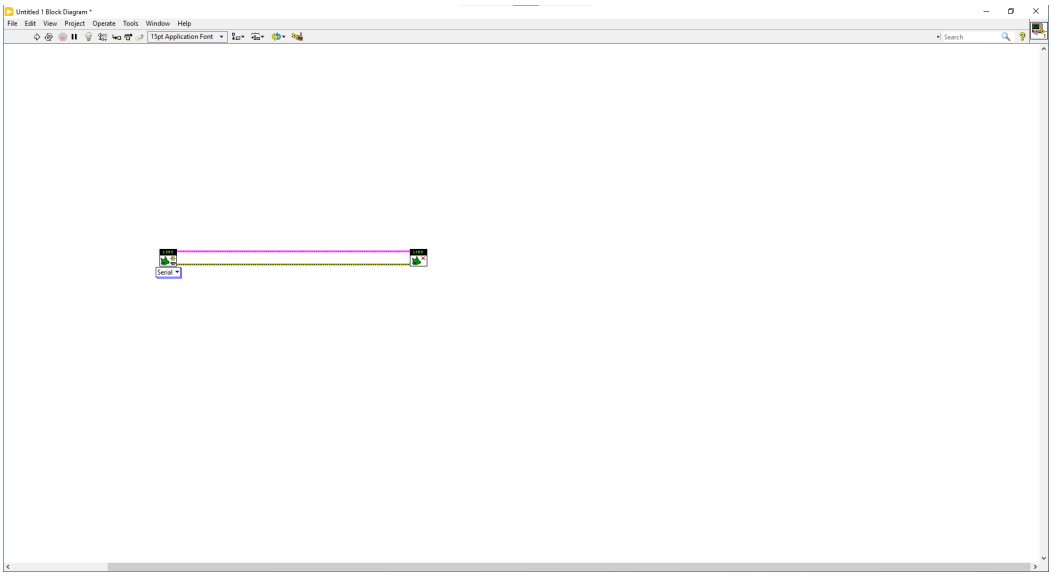


Figure 104: LabVIEW LINX Start and Stop placed

You would likely want to encapsulate most processes within a while loop, which would continue to run until triggered to stop, then allowing the close function to run.

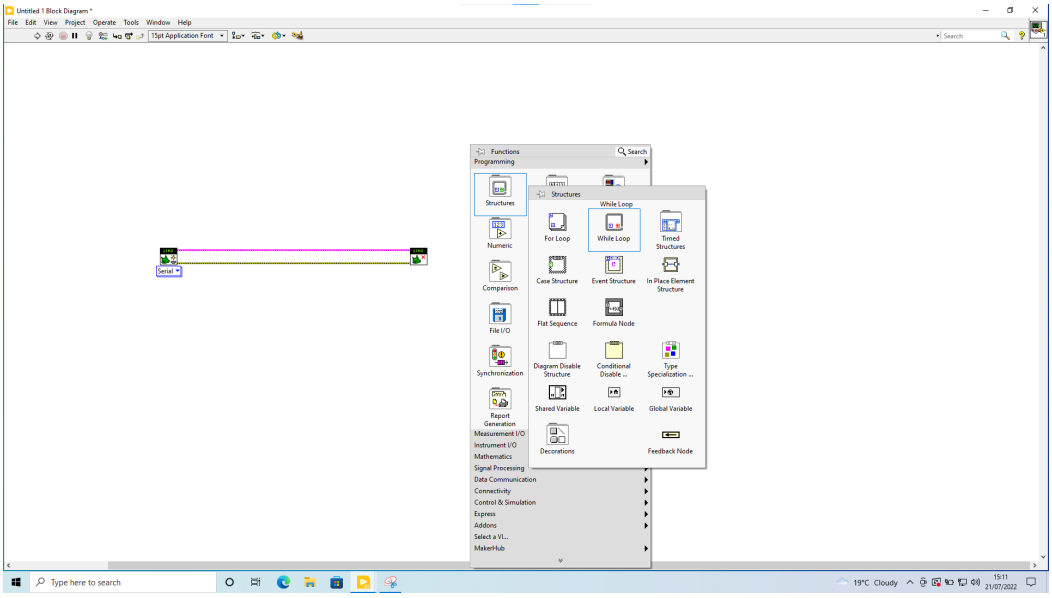


Figure 105: LabVIEW While Loop select

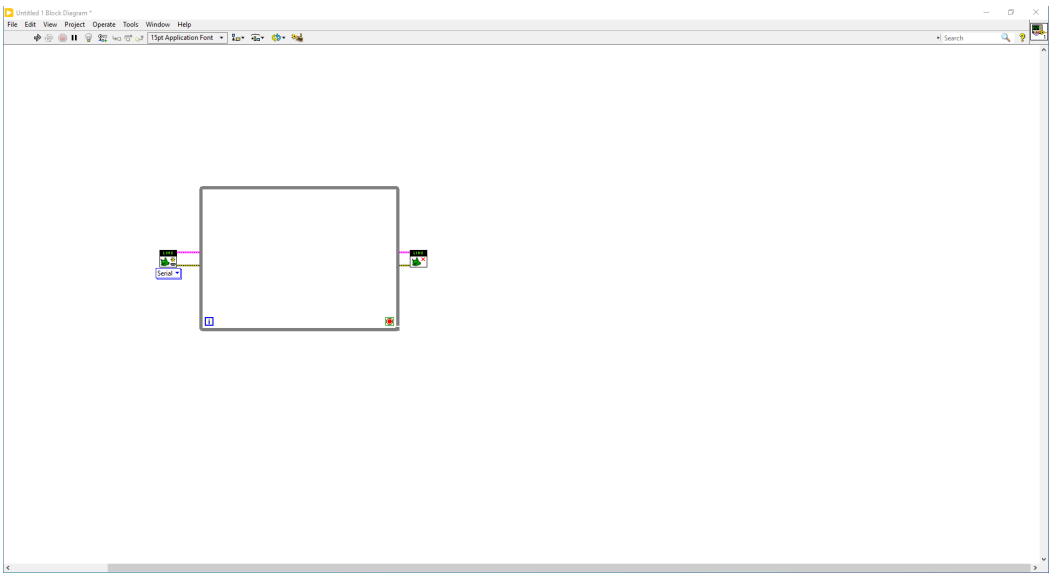


Figure 106: LabVIEW While Loop placement

There some processes are on the computer side and some on the Arduino side. Functions that may be regularly used is the digital and analogue write and read. Each of the LINX elements/ blocks are connected through the LINX resource line, indicating the order in which they are done (from left to right). Each LINX resource should be connected to the error out, which can be connected to the loop condition. This means if there is an error, it exists out the loop, so that the Arduino can safely close with the LINX close block. A DO channel and DI channel is used used to select a pin to either write or read to, to be passed to the LINX digital/ analogue read or write block. There are a range of other

elements not part of the LINX package which may be necessary. These include a range of numeric, boolean and comparison functions. These can connect to items placed within the front panel, which also appear in the block diagram.

So a combination of boolean elements, buttons, and LINX elements, along with the while loop could make up a program. They could be used together, for a system where an **LED** comes on when one of two switches are pressed, but while a pause button is pressed, pressing either of the switches will not affect the **LED**'s **Current** state. A front panel output could represent the **LED** too. We can imagine this **LED** to be a certain process needed for an experiment, which may need to run only when one of two variables is true, unless manually overridden.

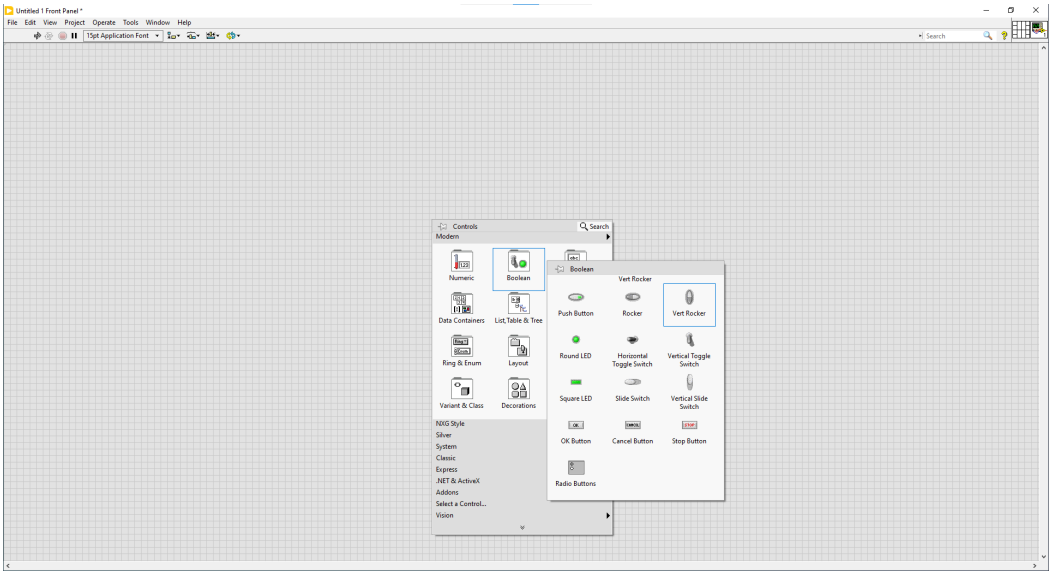


Figure 107: LabVIEW Front Panel Button select

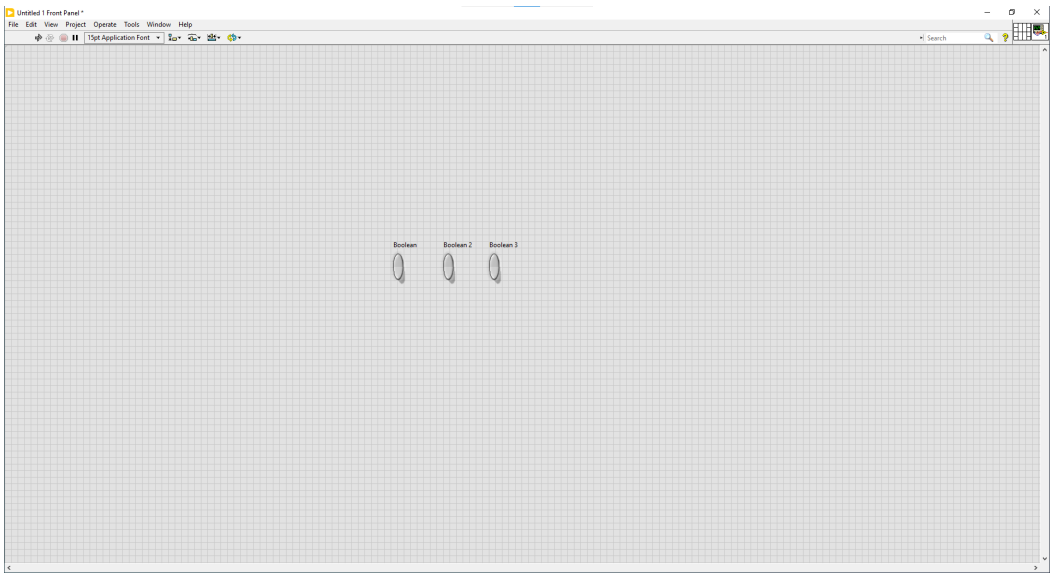


Figure 108: LabVIEW Front Panel Button placement

Three toggle switches have been selected, although there is a range of different switch types. You should consider that the Front Panel is there to provide an easy and convenient interface, so you should select inputs which make it more intuitive.

Therefore, I am going to choose to delete one of these switches, replacing it with a pause button. To delete an object, you can simply click on it, and if you see a blue box surrounding it and its corresponding text, then you can press the "Delete" key.

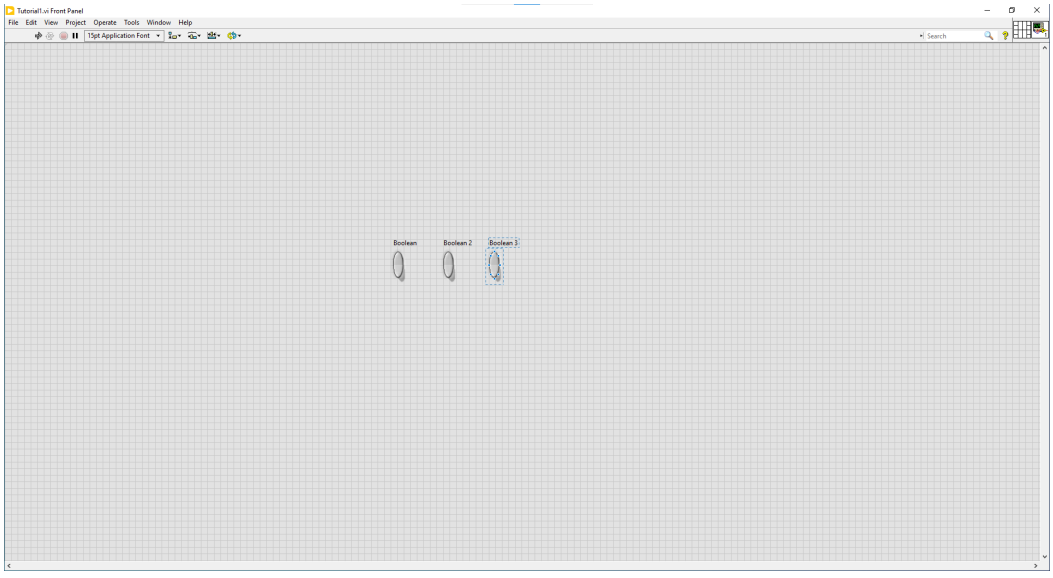


Figure 109: LabVIEW Front Panel Button delete

As we can see, first of all I have selected a cancel button, this is to show that text as well as names can

be altered simply by clicking on things. We could have the button having a different name to what is on the button if we really wanted to. I will show this by writing "Press" on the button, with the name of "Pause". You can do this simply by double clicking on the text and then typing.

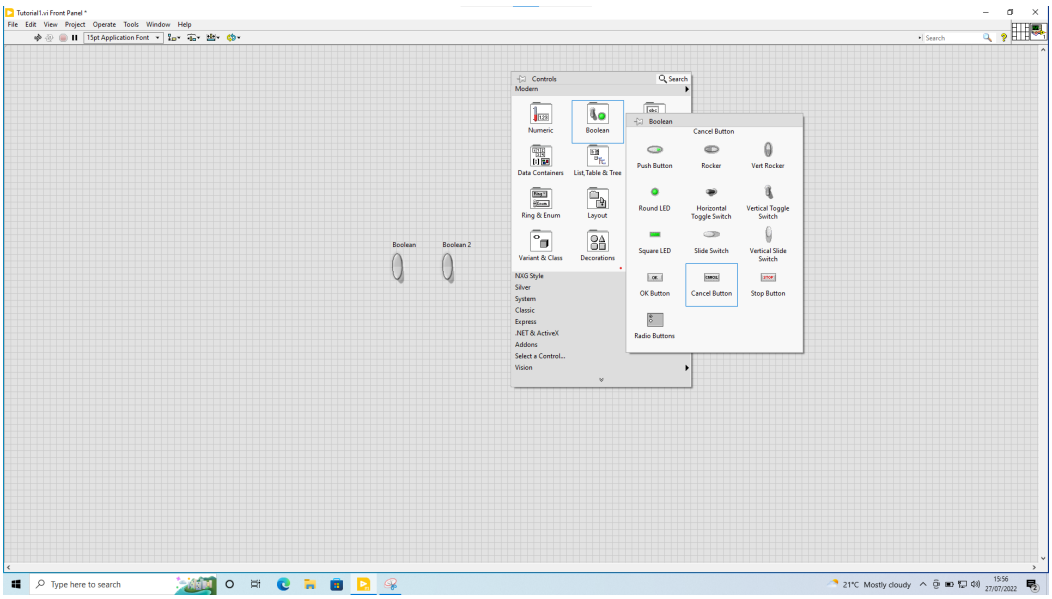


Figure 110: LabVIEW Front Panel New Button

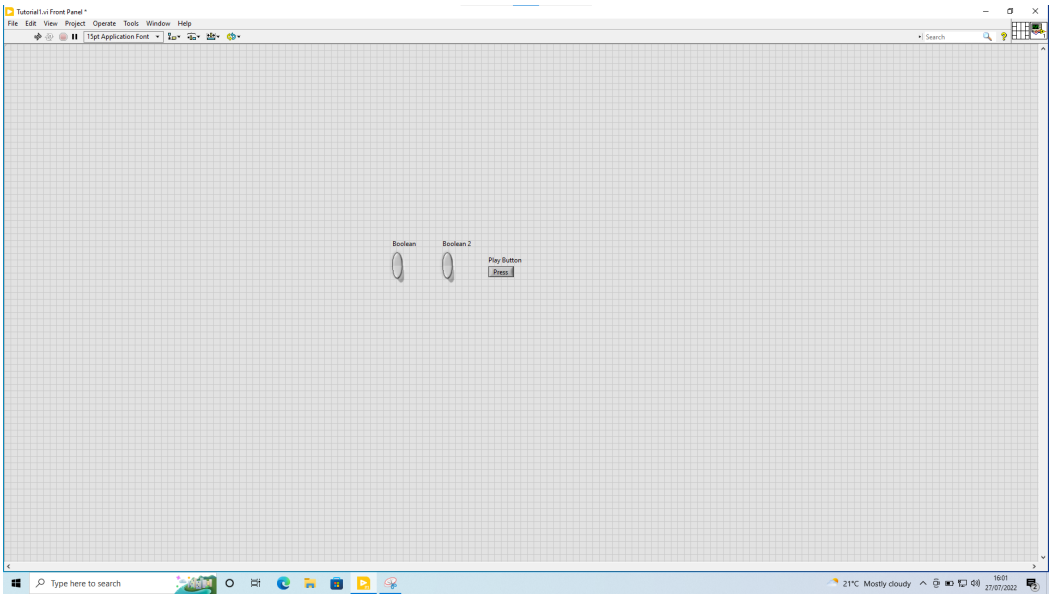


Figure 111: LabVIEW Front Panel Pause button rename

We will leave it like this for now, and will come back later to improve their positioning.

As seen, upon opening the Block Diagram, the new buttons are there, and are moveable. We will now get more blocks set up. The buttons want to be within the while loop, so they are checked only once

the Arduino setup has started and during the main operation. If you click and drag on blocks, you can move them to where you would like them to go.

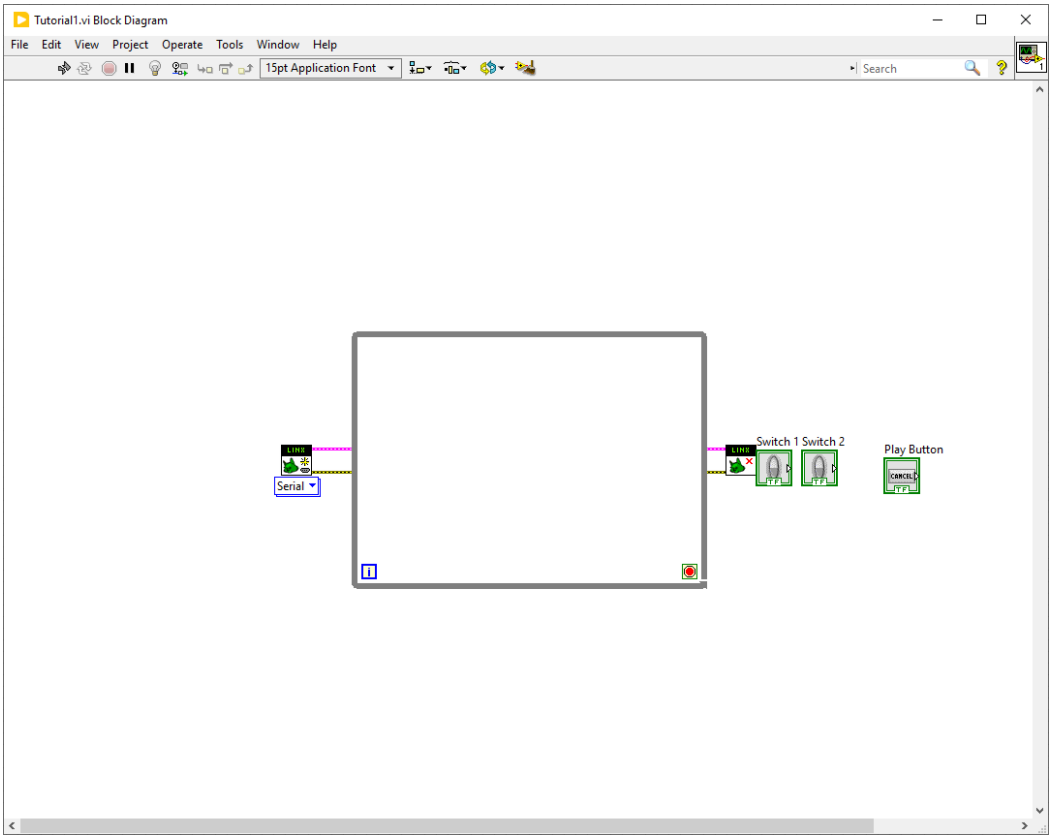


Figure 112: LabVIEW Block View Buttons

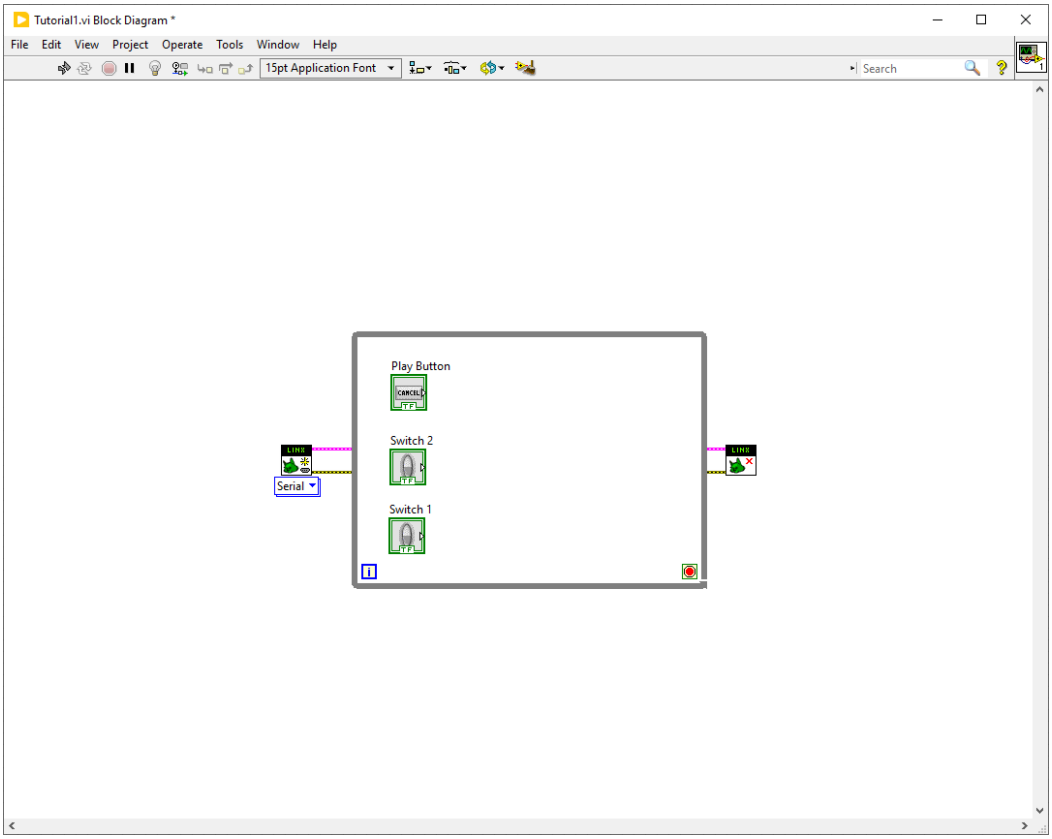


Figure 113: LabVIEW Block View Button Movement

We need to set up a way of manually selecting which Serial Port is used before the program runs. If you right click on the top left side of the LINX start block, there should be a purple node with the name "Serial Port". You should right click and then select "Create Control", which creates a front panel element to select the port before the program runs.

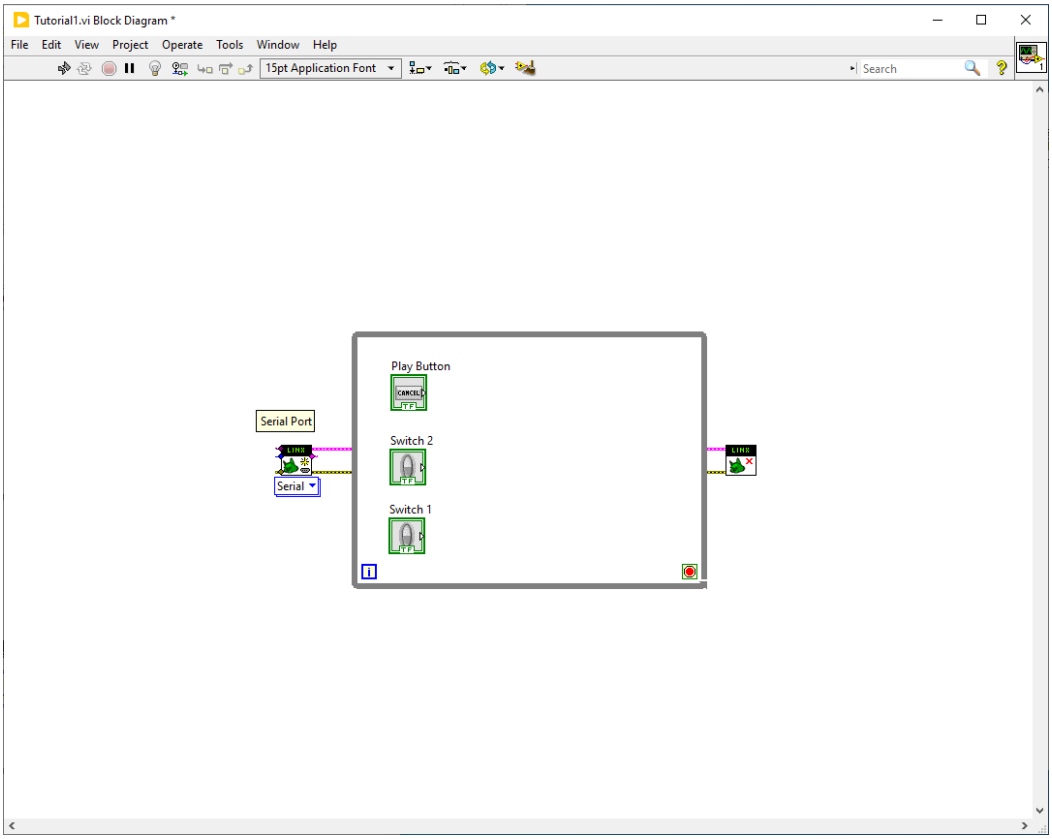


Figure 114: LabVIEW Block View Serial Port

We need to use a Digital Write function to interface with the **LED**. Upon an input we will set up later, it will send an output of 1 to a Digital Out (DO) Channel, set up soon. It needs to react in real-time to buttons being pressed, so should be placed within the while loop.

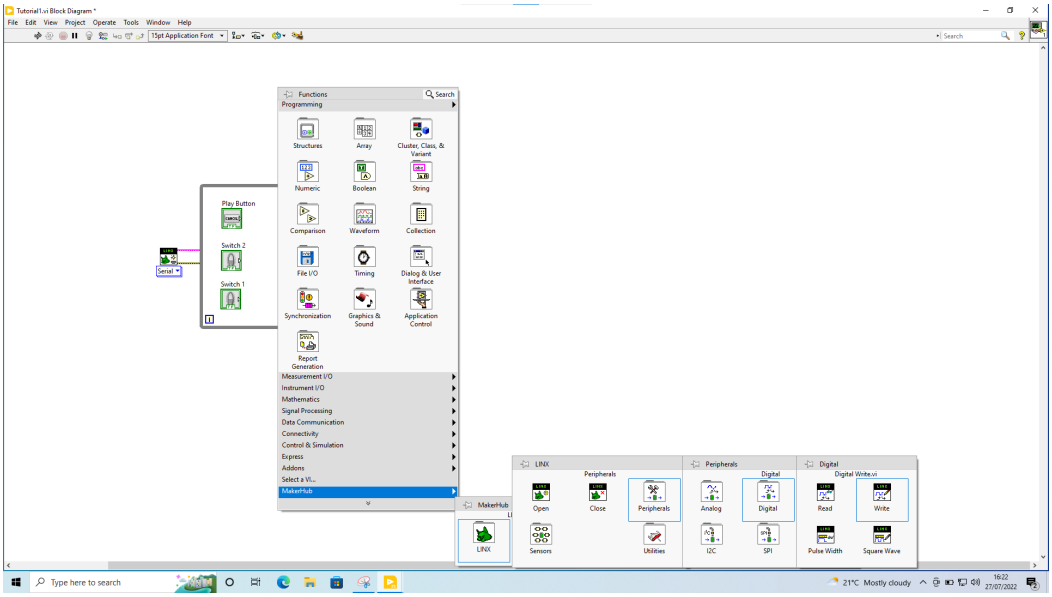


Figure 115: LabVIEW Block View Digital Write Find

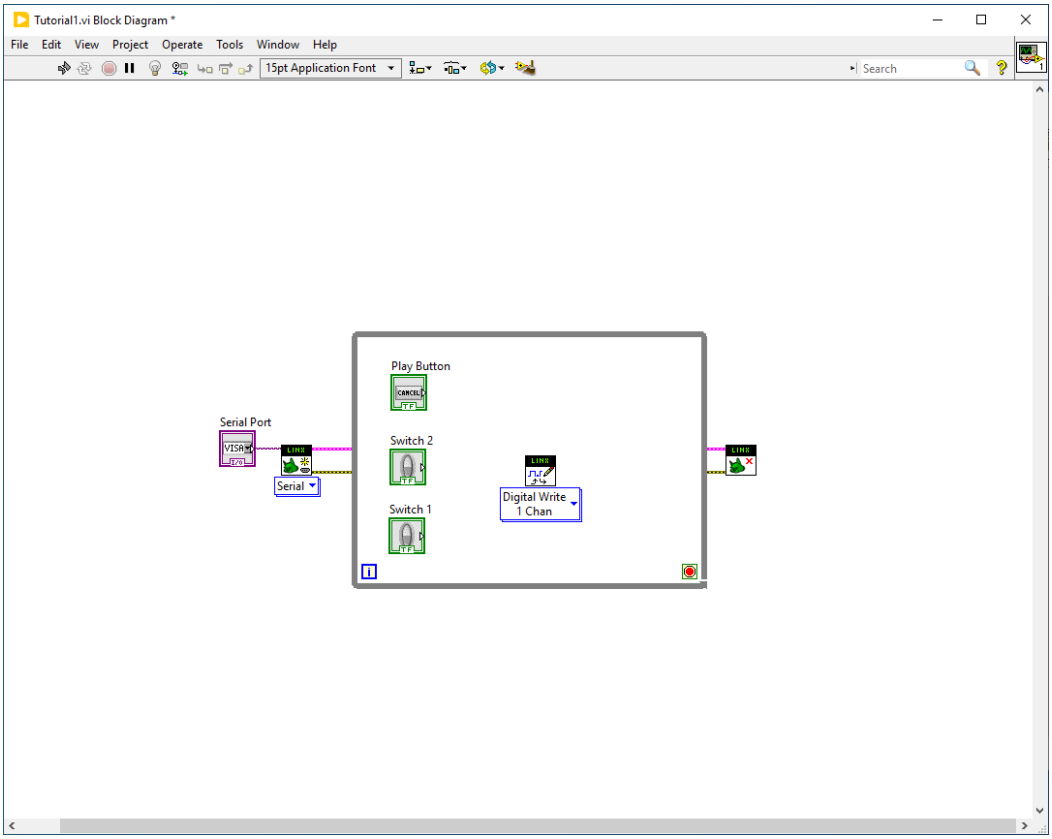


Figure 116: LabVIEW Block View Digital Write Placement

We will now sort the DO channel. This needs to be placed outside of the while loop, as the channel should only be selected upon the Arduino setup, and is something we want to keep constant. We are setting the **LED** to either be on or off, so don't need a range of output values. We should look for where it says DO Channel, which is a blue node on the left side of the Digital Write block, and again right click and click on Create Control". As mentioned, this should be moved outside the while loop.

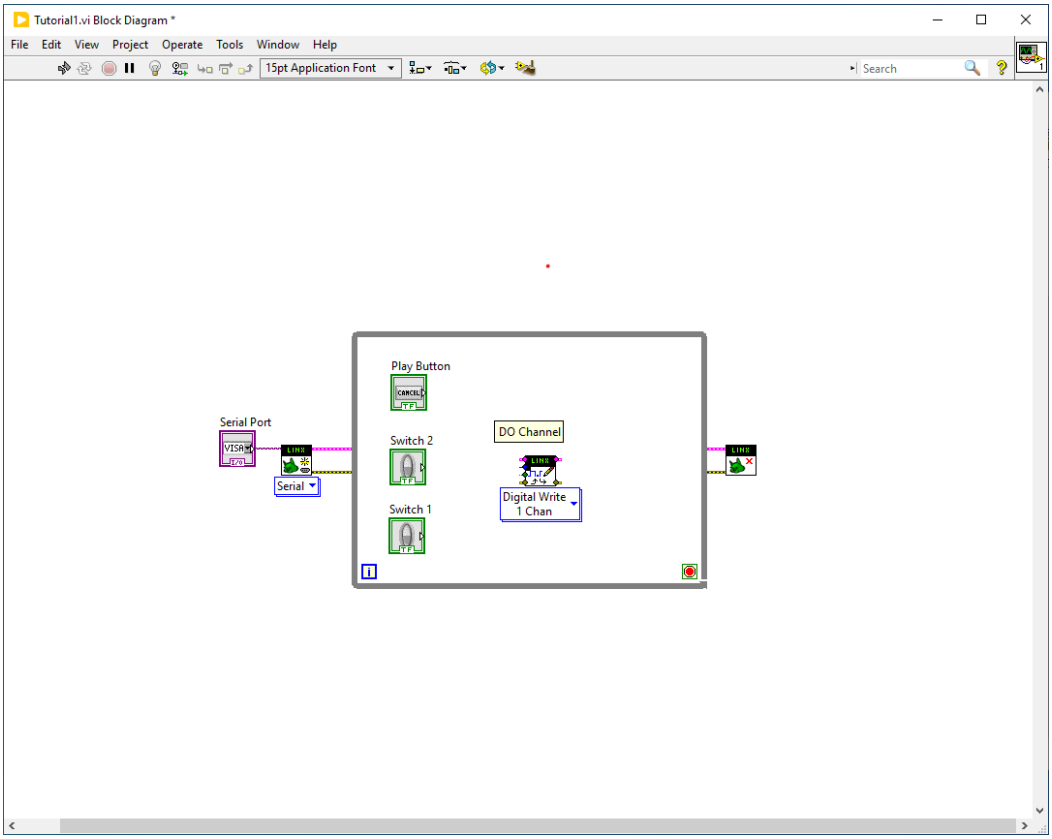


Figure 117: LabVIEW Block View Digital Output setup

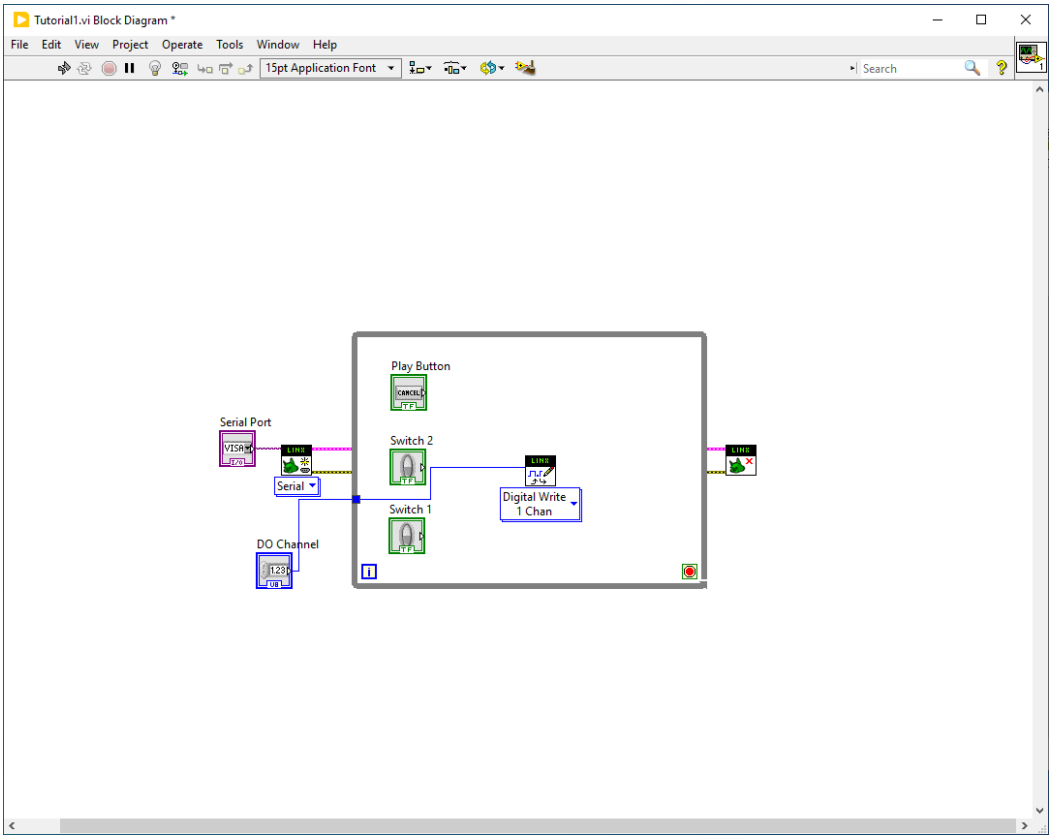


Figure 118: LabVIEW Block View Digital Output placement

As seen, the LINX start and stop blocks don't **Current**ly connect to what is in the while loop, so we can now connect both the error line and the LINX resource line to the Digital Write block. We also want to delete the existing lines that connect them directly using the "Delete key", which were just used to show how blocks connect.

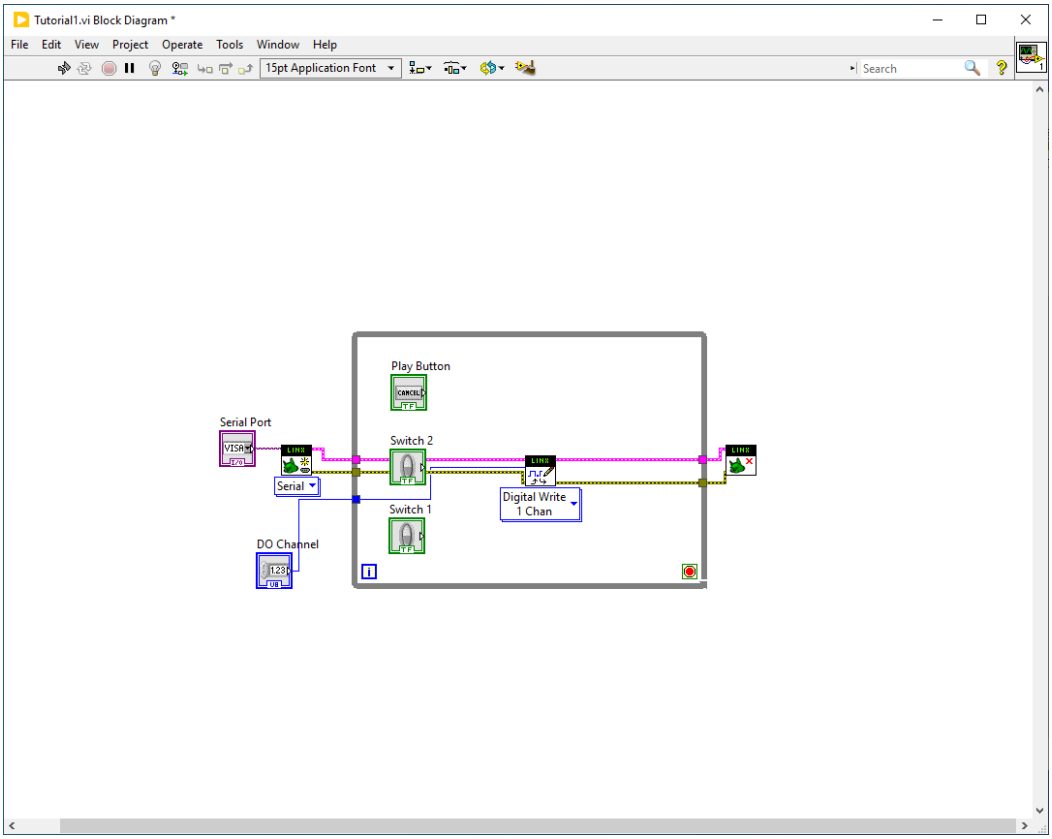


Figure 119: LabVIEW Block View LINX Block Connection

Now, these have been connected, it is good practice to rearrange things, as in the **Current** state, it is a little unclear. This can be done simply, by dragging blocks, wires and nodes around. Sometimes you may need to delete wires either through clicking and dragging to delete items in an area or through right-clicking on them. As seen, the while loop box size has been increased to give more space for neat and clear placement.

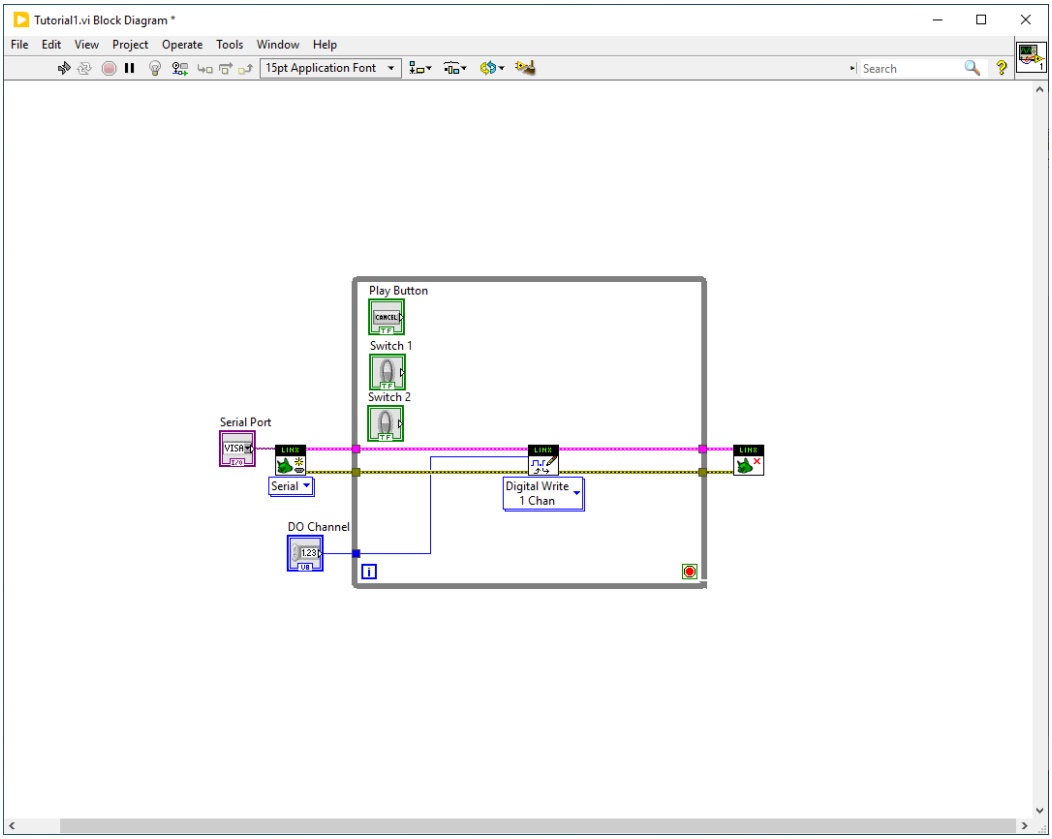


Figure 120: LabVIEW Block View Neatening things

As mentioned at the start, we wanted to have the **LED** turn on if either button is pressed, but only when it isn't paused. This can be done pretty easily with Boolean Logic, using an OR gate for the two Switches, and an XOR gate between the output of that OR gate and the Play/Pause button.

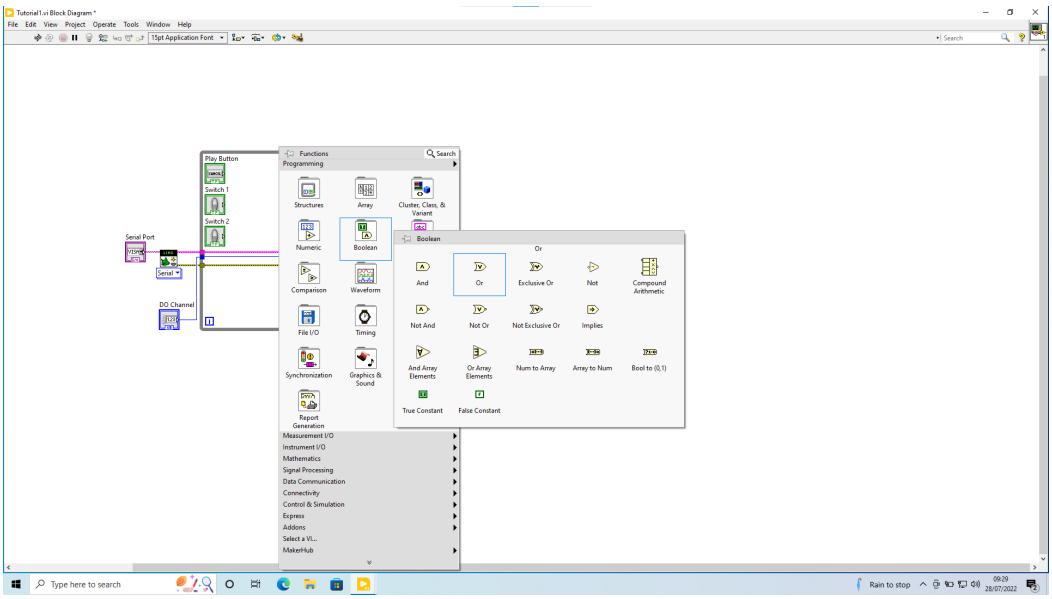


Figure 121: LabVIEW Block View Finding Button Logic

Like we have already done, its as simple as connecting the blocks together.

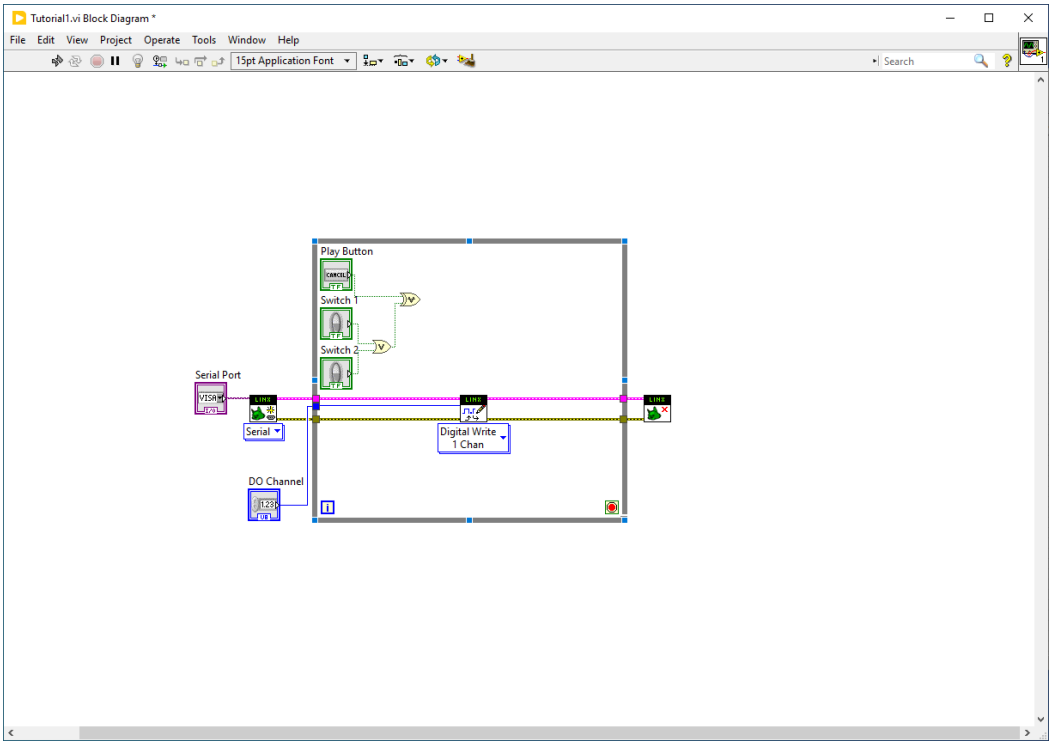


Figure 122: LabVIEW Block View Adding Button Logic

This can now be connected to the Output Value node of the Digital Write Block, which means that the value (either 1 or 0) from the switches is what is "written" to the **LED**.

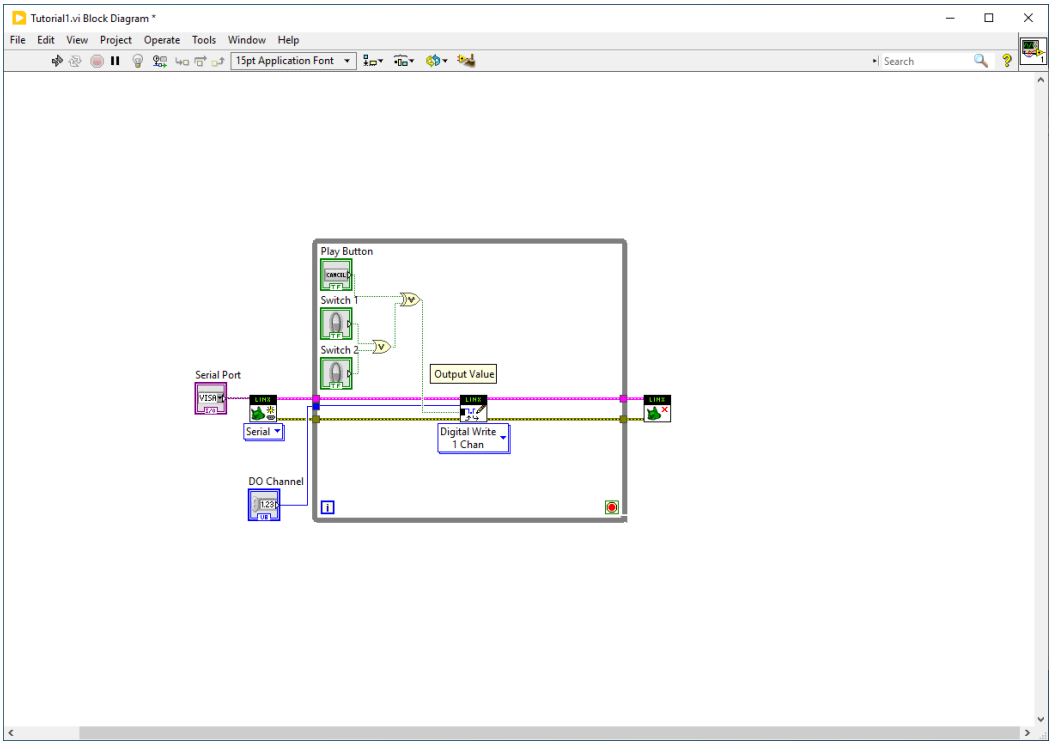


Figure 123: LabVIEW Block View Connecting Button Logic to Write Block

Something that hasn't been done yet, is adding in a Stop button. It is always good practice to have a button to properly stop the While loop so that the Arduino can safely stop using the LINX Stop Block.

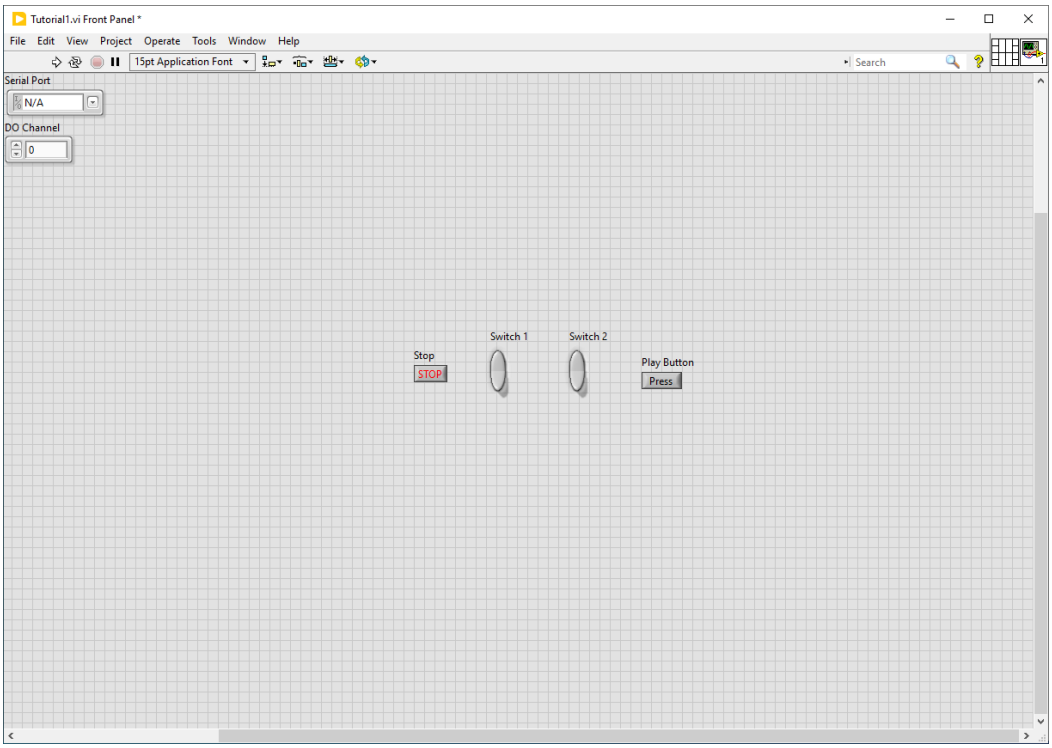


Figure 124: LabVIEW Front Panel Adding Stop Button

This stop button block can be connected to the Stop "Loop Condition". Since we would also want the loop to stop if there are any LINX based errors, we will use an OR gate, so that in either case, the loop would stop.

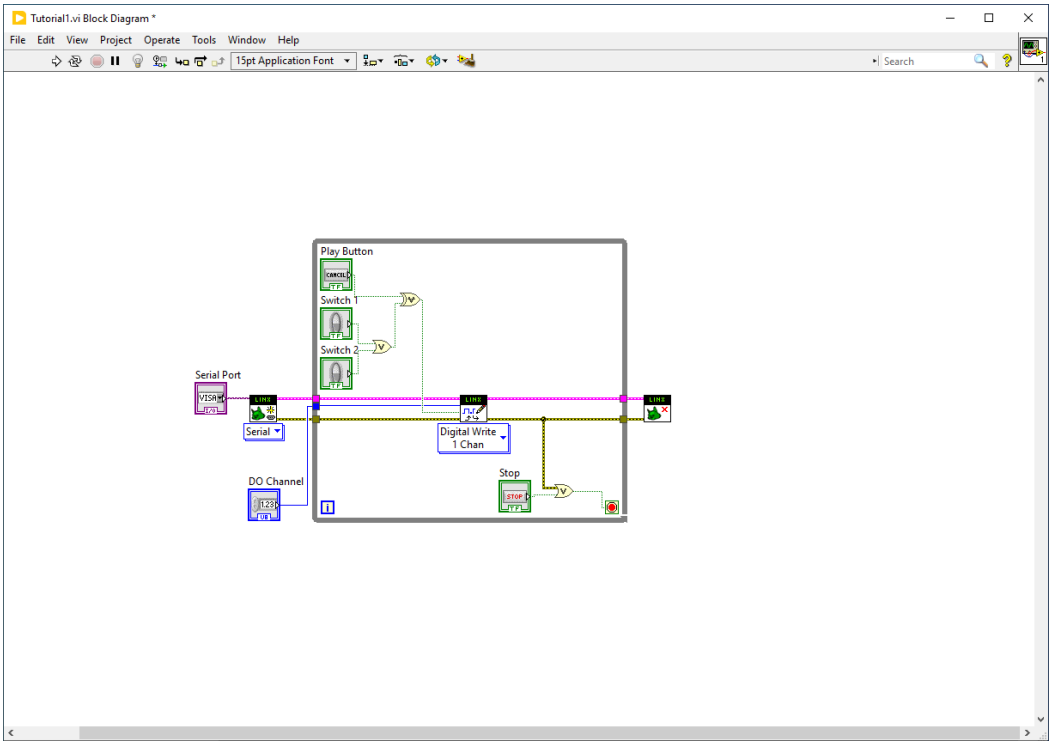


Figure 125: LabVIEW Block View Final Layout

It is now at the point where it can be tested with the Arduino, making sure to first save the program. Initial testing is worthwhile before investing time into making the Front Panel easier to use.

You should plug the Arduino in, if you have followed prior steps, it should work when needed.

Before pressing run, you need to use the control boxes for the Serial Port and the DO Channel. The correct COM port should come up for the Arduino, and if using the Arduino's inbuilt **LED** you should select channel/ pin 13.

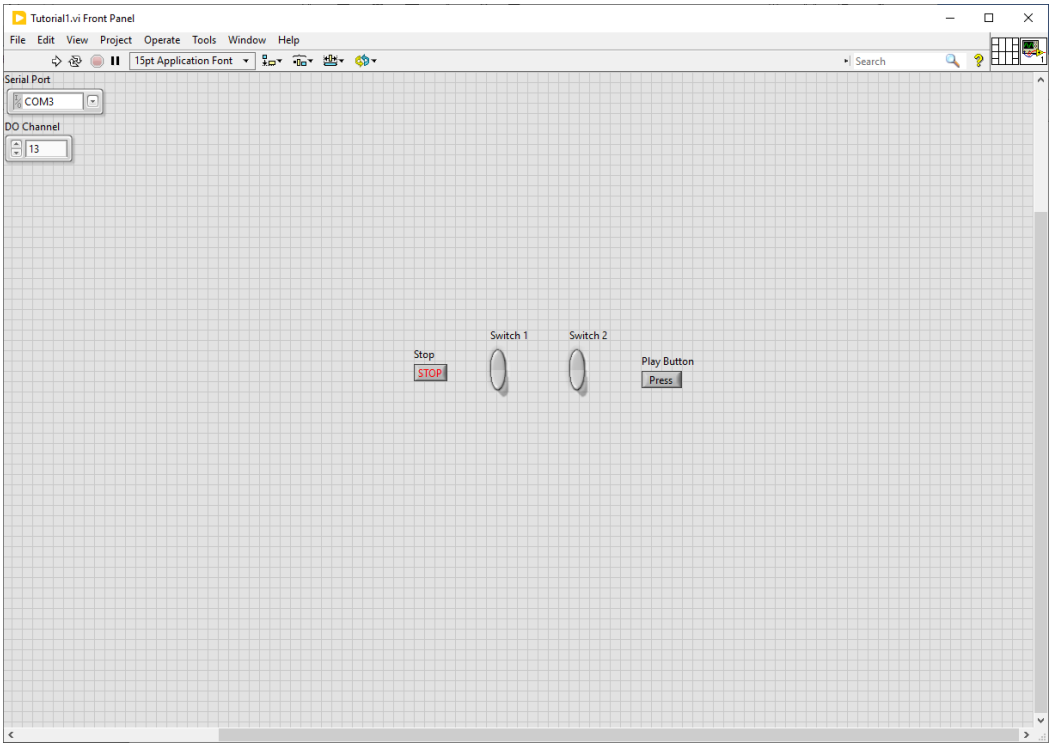


Figure 126: LabVIEW Front Panel Port and Pin Config

On pressing play, it will take a few seconds for the Arduino to set up, once set up, we should see both the TX and RX **LED**s light up, again indicating data transfer, and thus a successful connection.

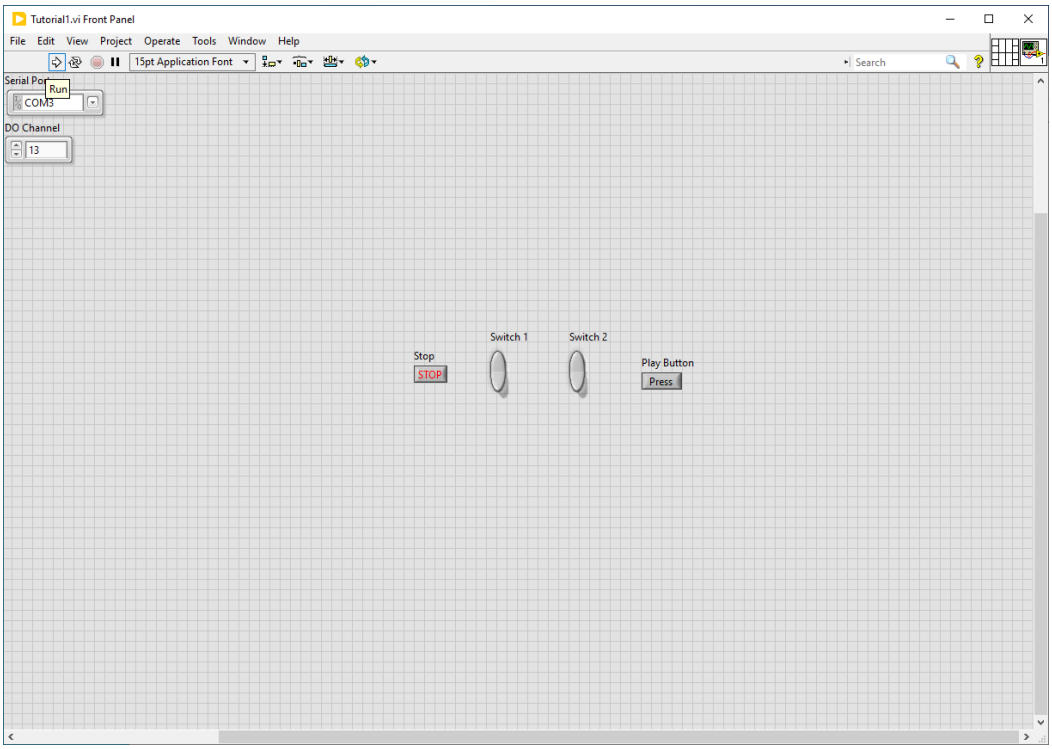


Figure 127: LabVIEW Program Run

You should now test each possible button combination to see whether it works as expected. Once finished, press the stop button you added to safely stop the program, rather than the built-in stop button. The built-in LabVIEW stop button forces the program to stop, without giving time for the stop sequence of the LINX Stop Block to do its thing.

As can be seen upon testing, there is an issue. The play button does not hold in place, which we should fix. We could use the simple step of swapping out the button for a fixed position switch. It is best to delete and place a new one, although, there is the "Replace" option. In this case, we are changing it to a vertical toggle switch and changing the name to Pause button.

We will also decide to change the logic, so that only one switch press at a time will cause the **LED** to light, using a XOR gate. This has been added just to show that changes can be made to the existing system quite easily, and is frequently part of the process.

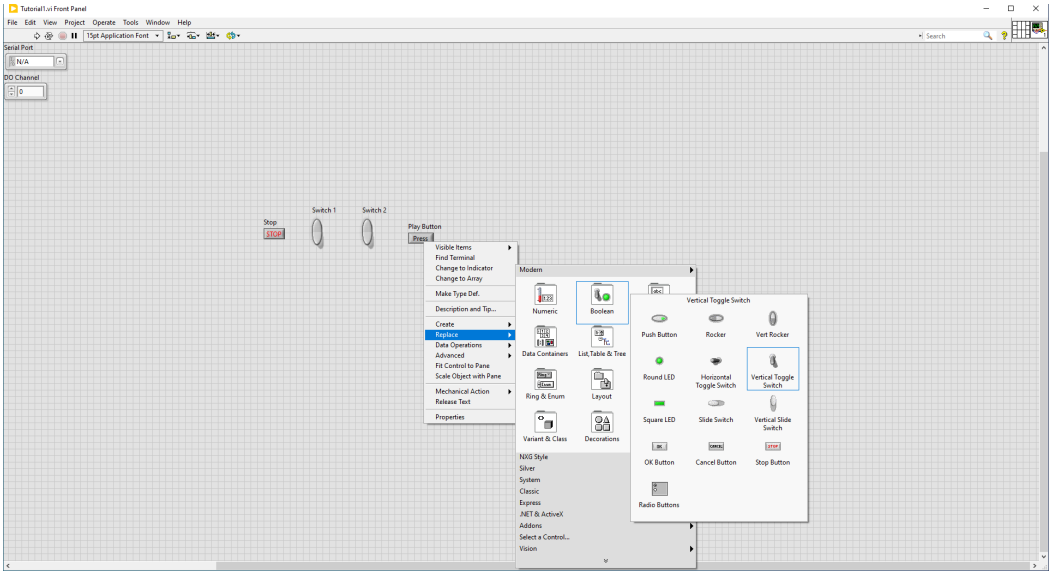


Figure 128: LabVIEW Block View Final Layout

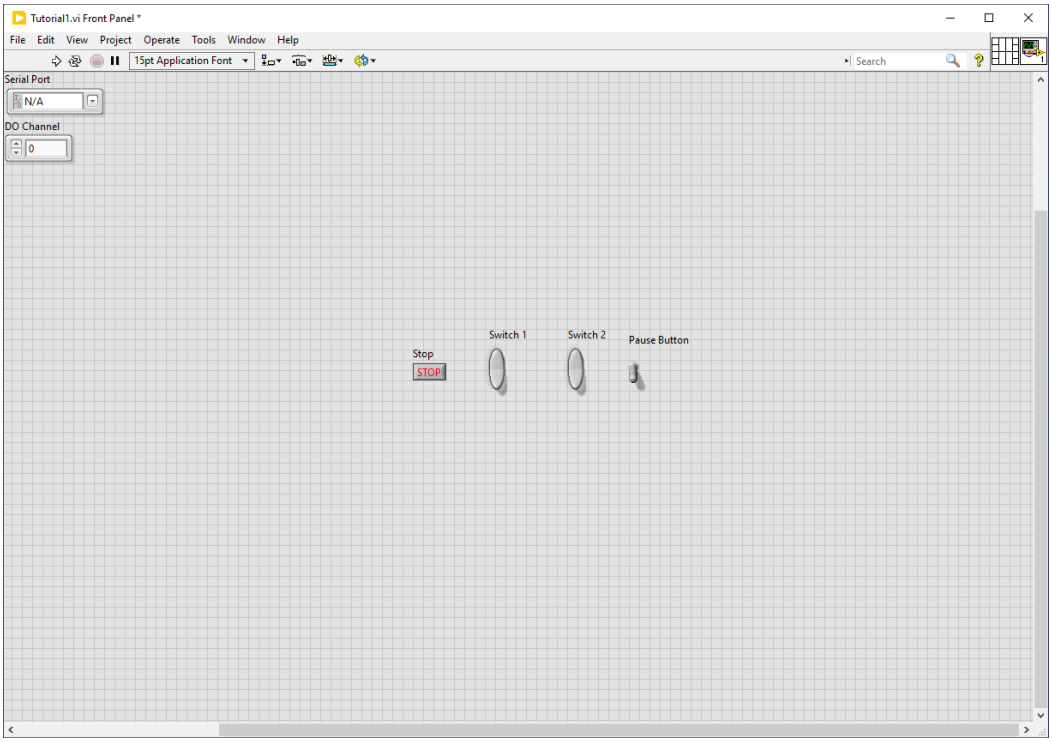


Figure 129: LabVIEW Block View Final Layout

Before testing again, ensure that the Serial Port and Digital Out are set correctly.

We need to add a latch to hold a **Current** value until pause is pressed again.

There isn't a set component for a latch in LabVIEW, but it can be produced from our gate knowledge earlier on in the course. Below is the example circuit we would like to achieve, with an XOR Gate for the two inputs, and then when the pause button is pressed, the latch will hold the value until the button is pressed again.

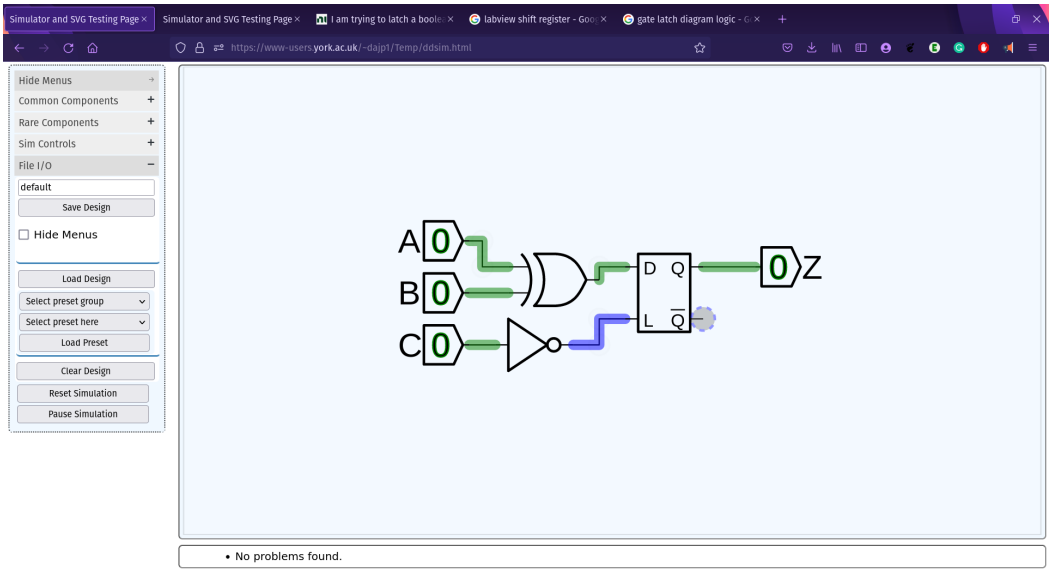


Figure 130: Pause Circuit Logic Diagram

Here is the below, when broken down into the necessary Gates to product the Latch.

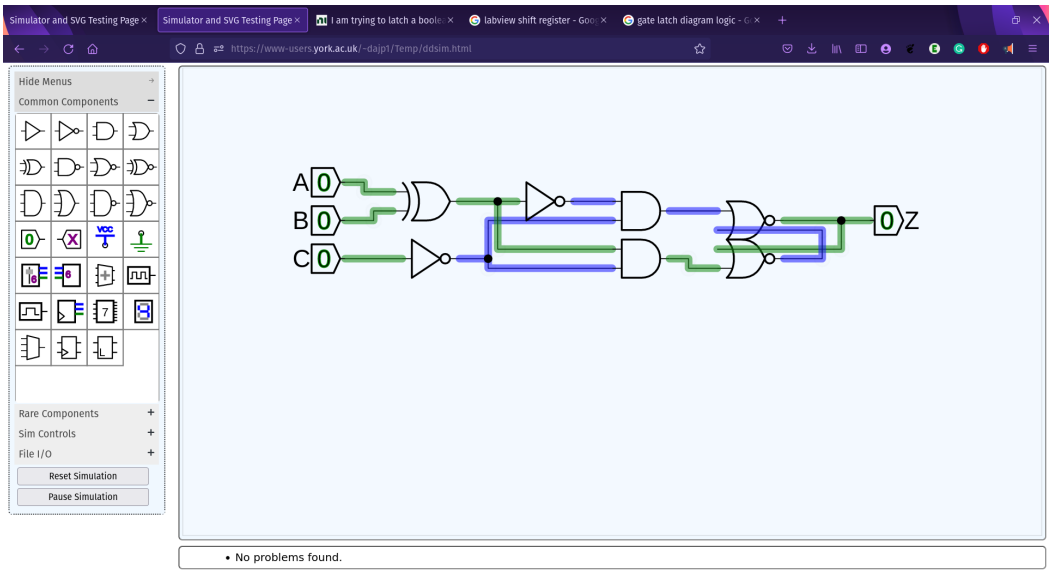


Figure 131: Pause Circuit Gate Logic Diagram

This works with a switch which is held in place until it is next pressed, rather than a button.

This can be drawn into the Block Diagram now. It may add a feedback node automatically, due to how some nodes connect together, but you don't need to worry about that.

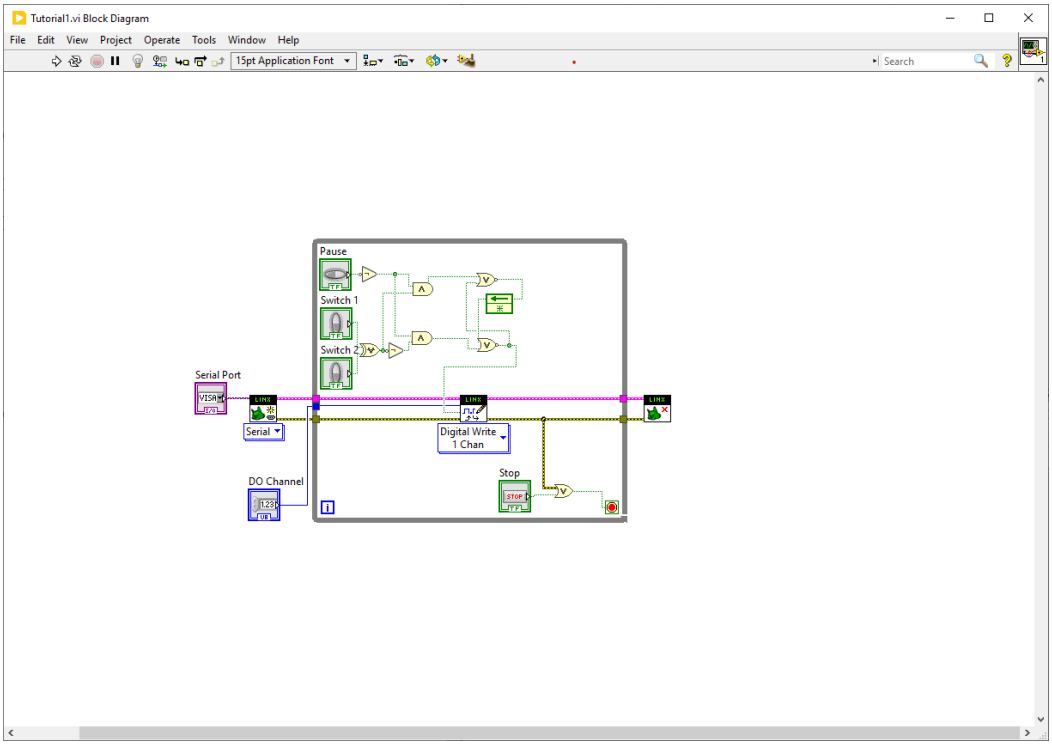


Figure 132: LabVIEW Block View Final Layout with Pause functionality

Now we have finally got the main functionality which is needed, the front panel can be edited to be more helpful. Assuming what the buttons trigger is more advanced, having an **LED** indicator on the front panel can be helpful, and in our case, its a good way to check whether the Arduino **LED** is working correctly. This front panel **LED** is an output, so connects to the same wire as the "Output Value" pin/ node.

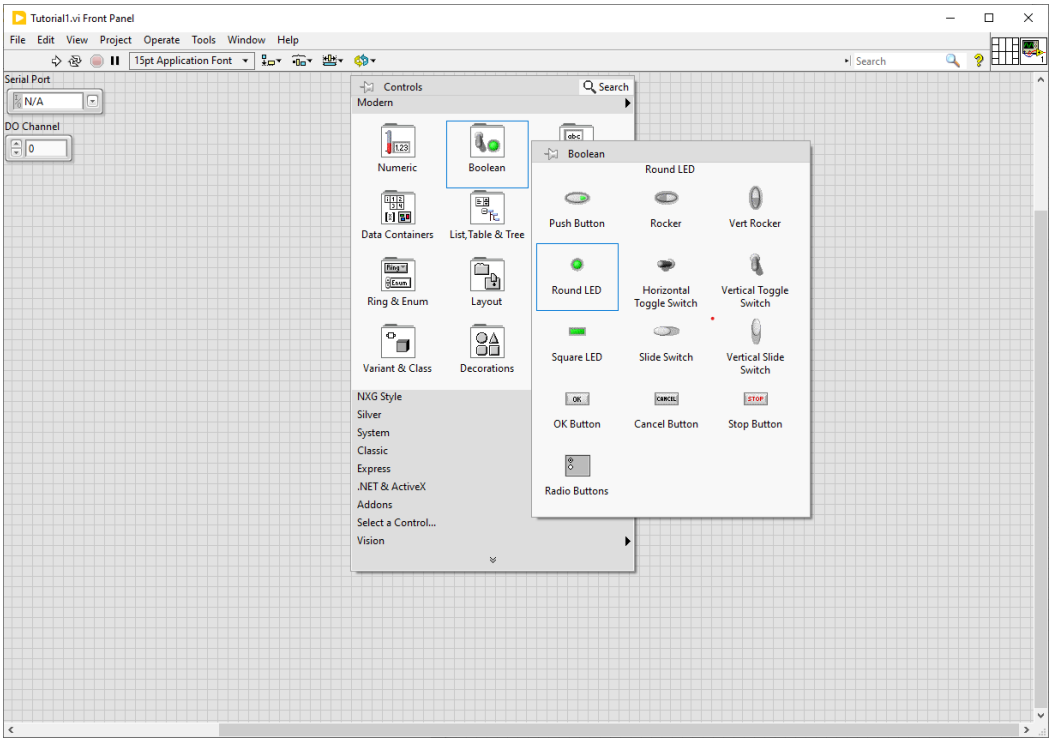


Figure 133: LabVIEW Front Panel **LED**

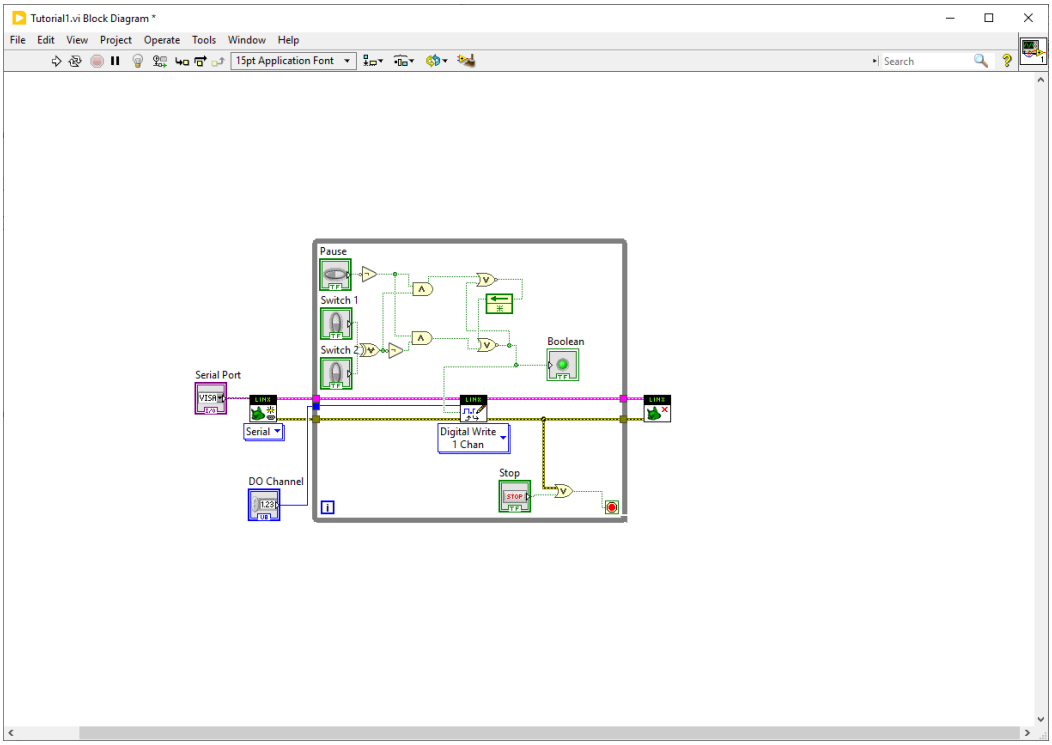


Figure 134: LabVIEW Block View **LED** connection

Assuming the Front Panel is how you like it, we can get ready to compile the document into a program. You may want to open the properties and alter some characteristics.

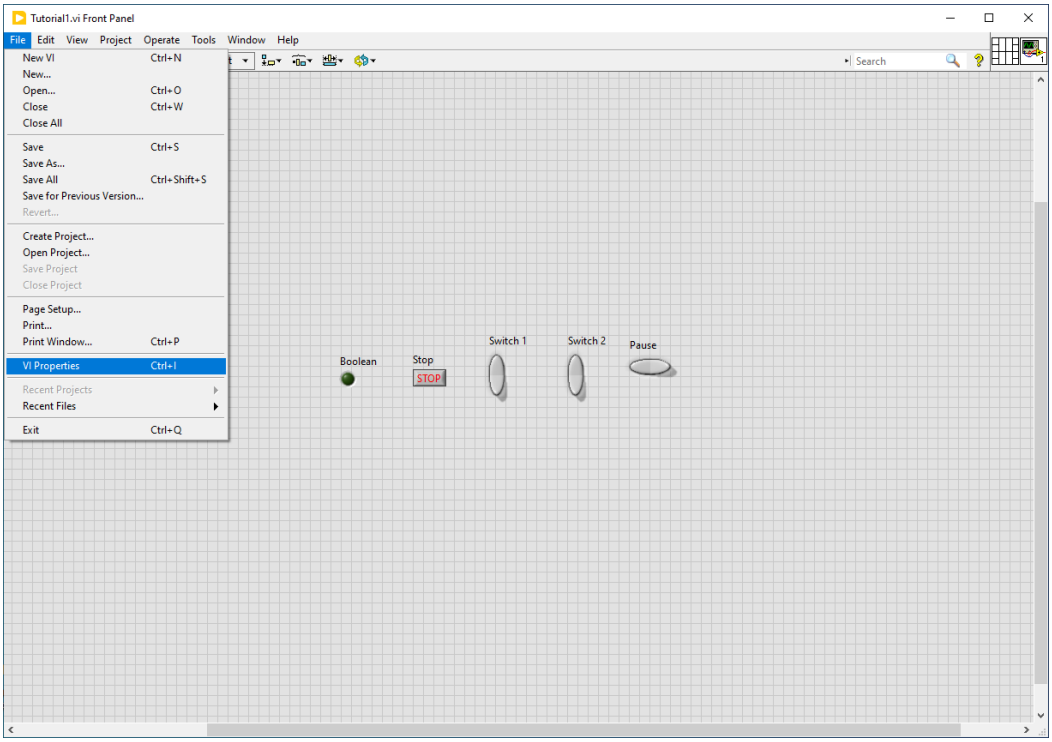


Figure 135: LabVIEW VI Properties

Given that the panels objects are positioned as intended, but only for a set screen size, having the front panel elements scale to whatever resolution is used for a display could be helpful. This is done through switching "Category" to "Window Size", and then once on that page, check the box, "Scale all objects on front panel as the window resizes".

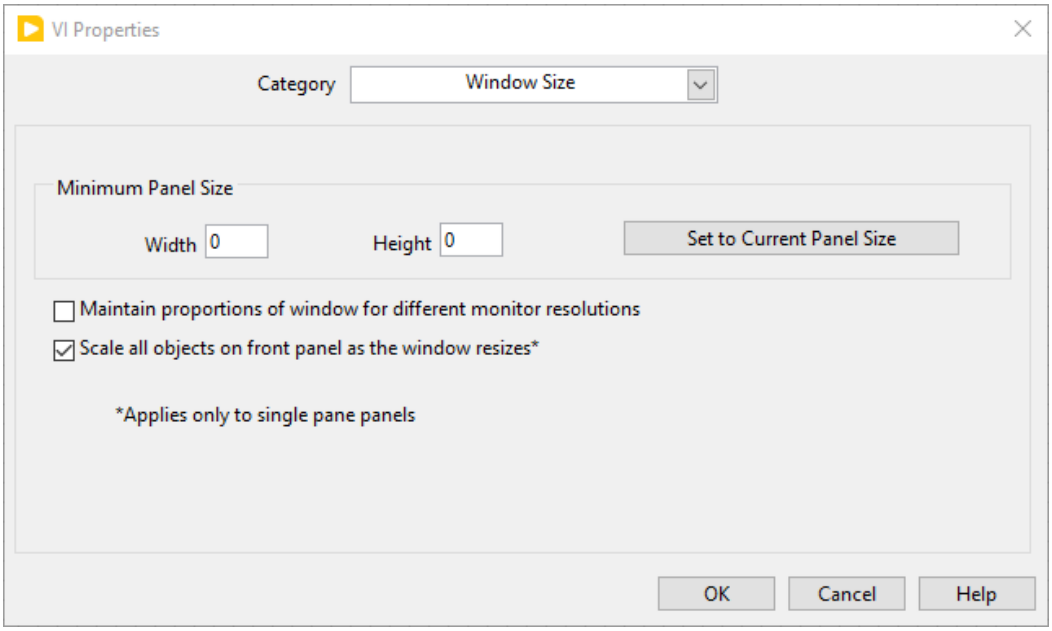


Figure 136: LabVIEW VI Properties Window Scaling

It may also be worth providing a small description for the program, if a user feels they need to check it. This can be written within the "Documentation" Category. This is meant for documentation on the workings of a VI, so think of it as a space to put comments about the general program function.

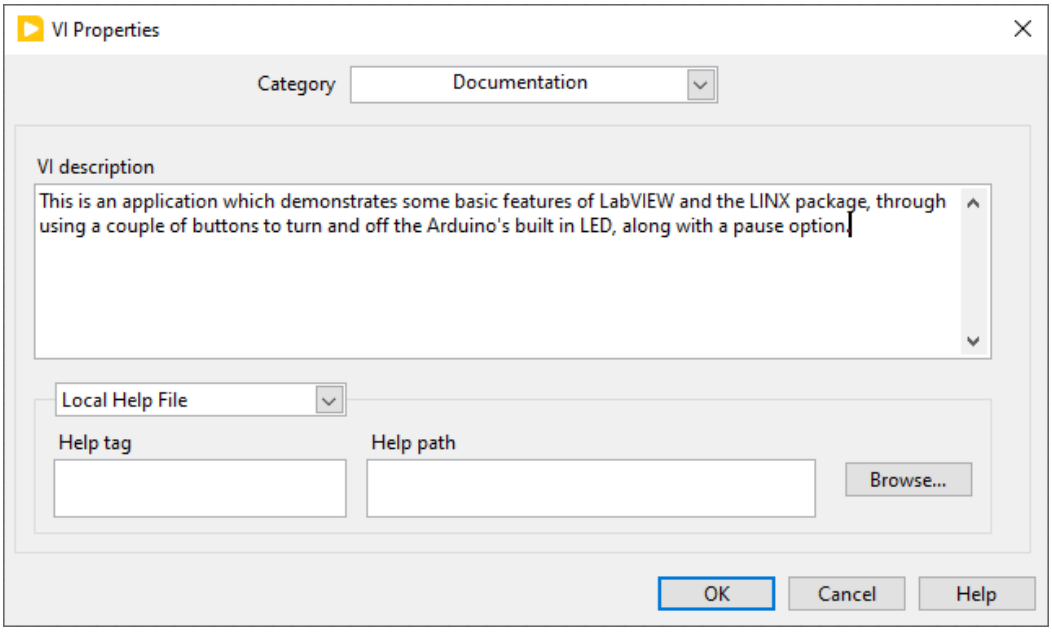


Figure 137: LabVIEW VI Properties Window Scaling

Now a couple of properties have been assigned, we can use a tool to convert the program into an executable, allowing it to be ran without opening in LabVIEW, meaning it could be shared for others to use. It allows for a "Read only" version of the program, less risky for field use.

To do this, you should click on "Tools" on the toolbar, and then "Build Application (EXE) from VI...".

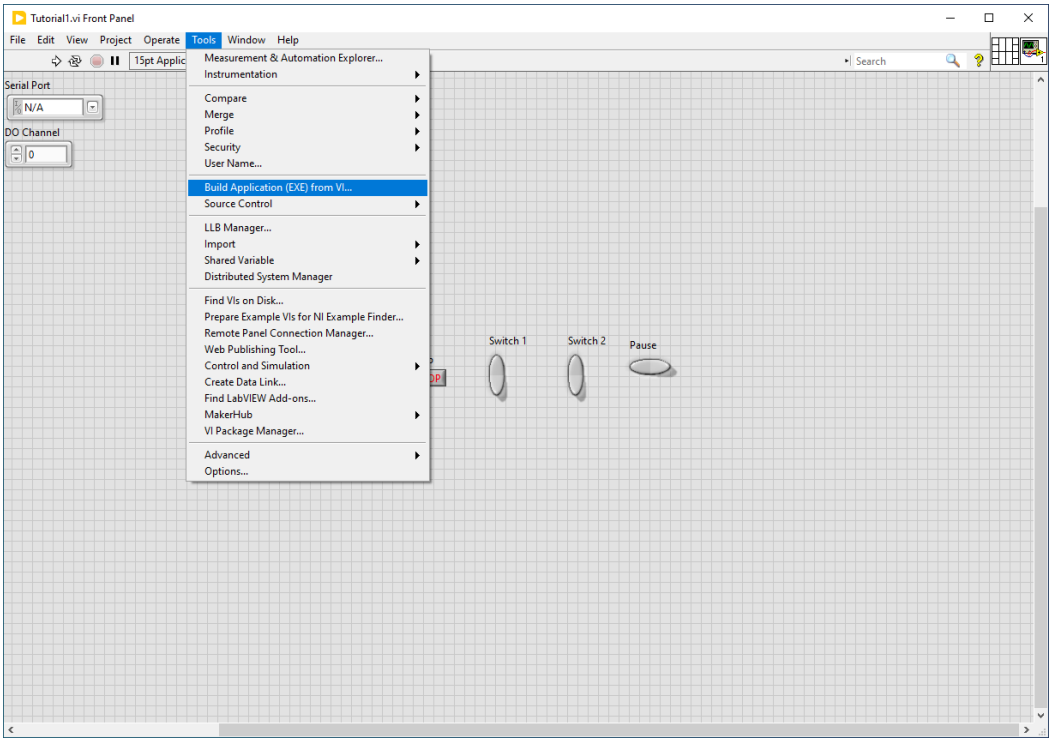


Figure 138: LabVIEW Build Application Tool find

Once you have clicked on this, you need to create a new file, which stores all the configuration information for building the program, with the file type "lvproj". Either use the default location, or choose a more suitable one and then press "Continue", to get onto selecting important properties.

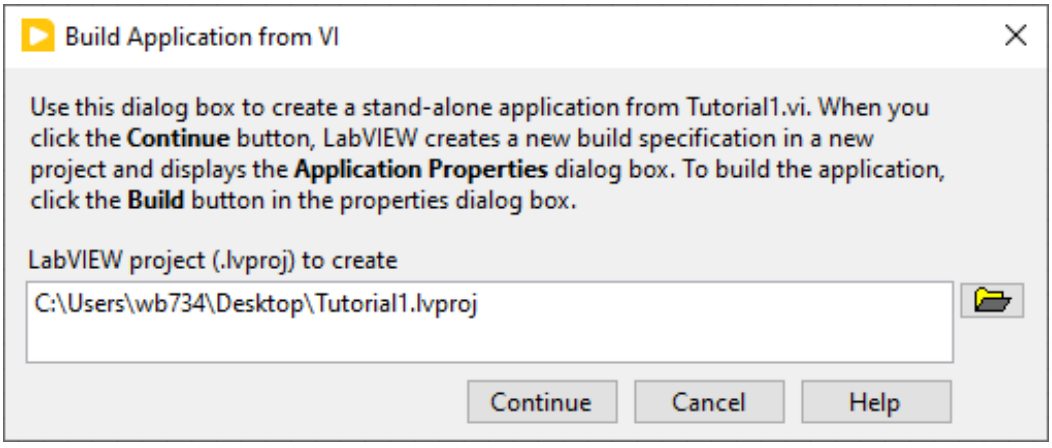


Figure 139: LabVIEW EXE Configuration Naming

The first main page is for "Information", such as the file name and the directory it will be saved to, as well as a description, which can be the same as the description you chose earlier.

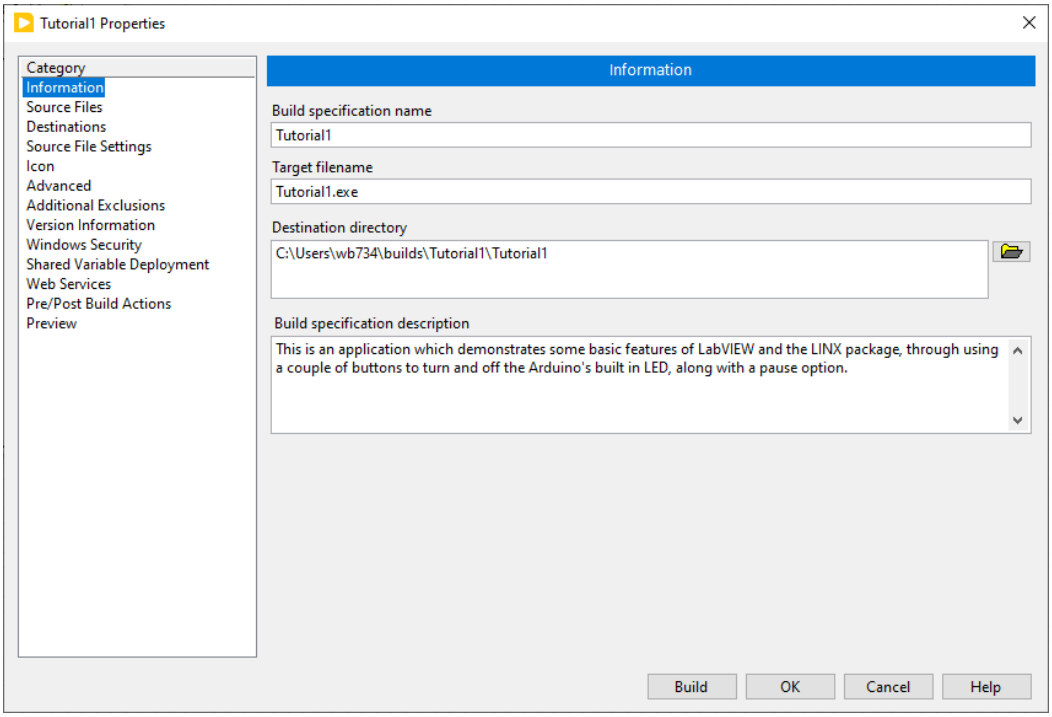


Figure 140: LabVIEW EXE Information Properties

There are quite a few parameters which can be altered, such as password protection through ”security”, although we will focus on just the key parameters.

Version information is pretty important, such as if the program has gone through multiple iterations, the description could be used to detail the differences to prior editions. Copyright information and Company name are other options.

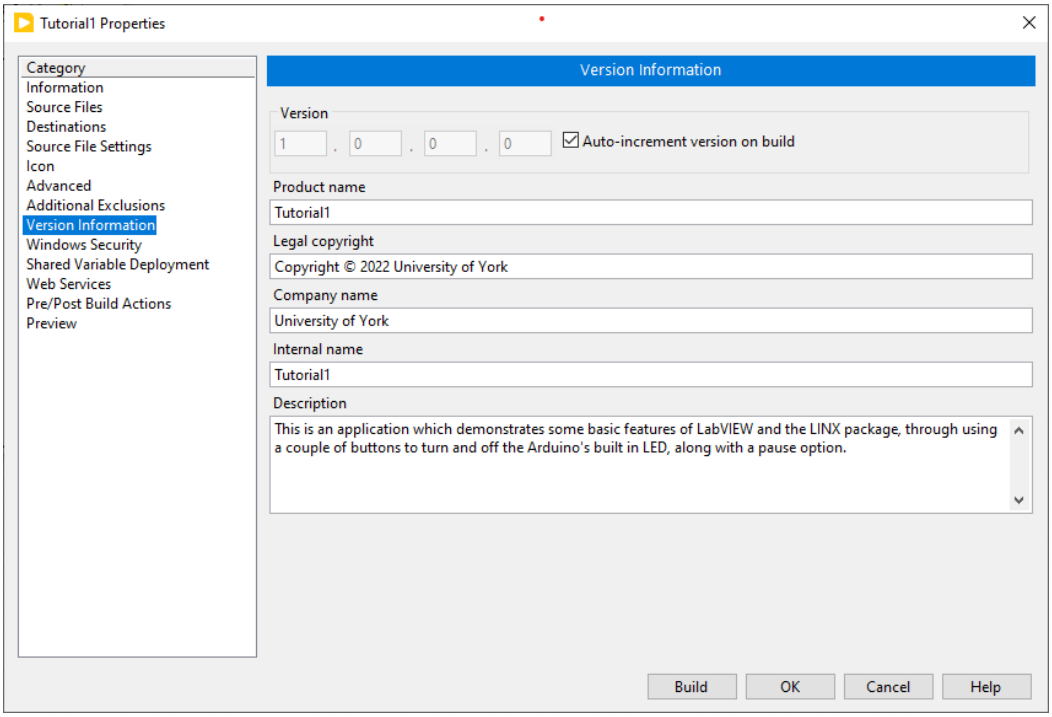


Figure 141: LabVIEW EXE Version Information

The last main properties that are important to check, are preview related ones, to ensure the program files are produced as expected.

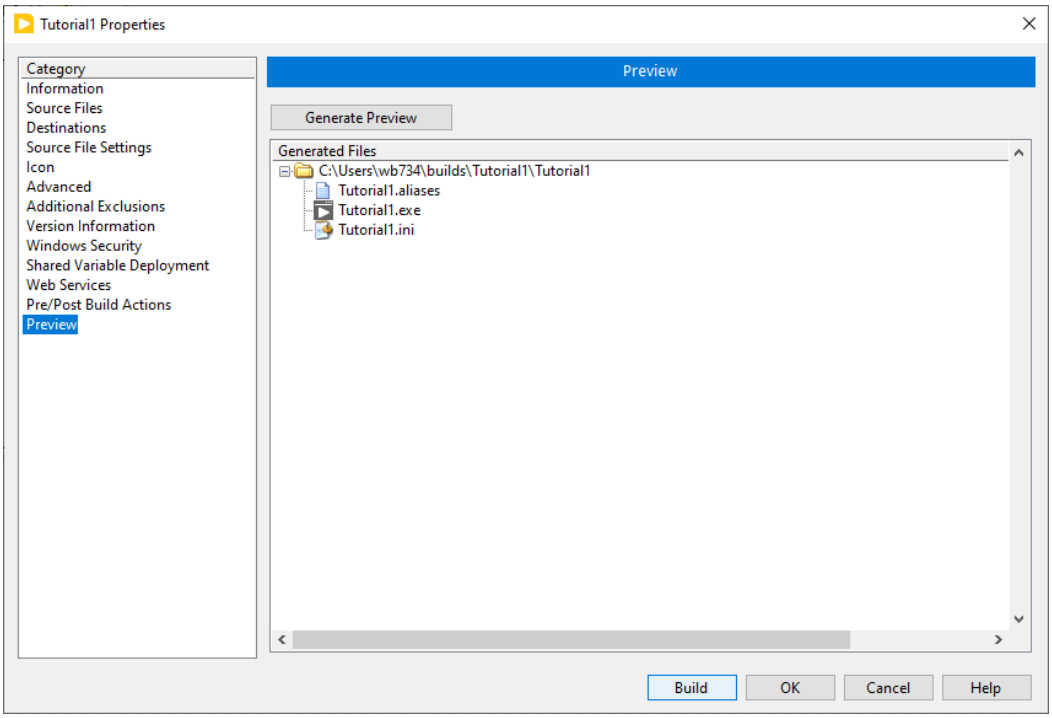


Figure 142: LabVIEW EXE Preview Page

You can then press "Build", with a popup showing its status as it compiles the program into an EXE executable. This is now something you should be able to run on any machine with the LabVIEW Runtime program, which is free to install with or without licensing.

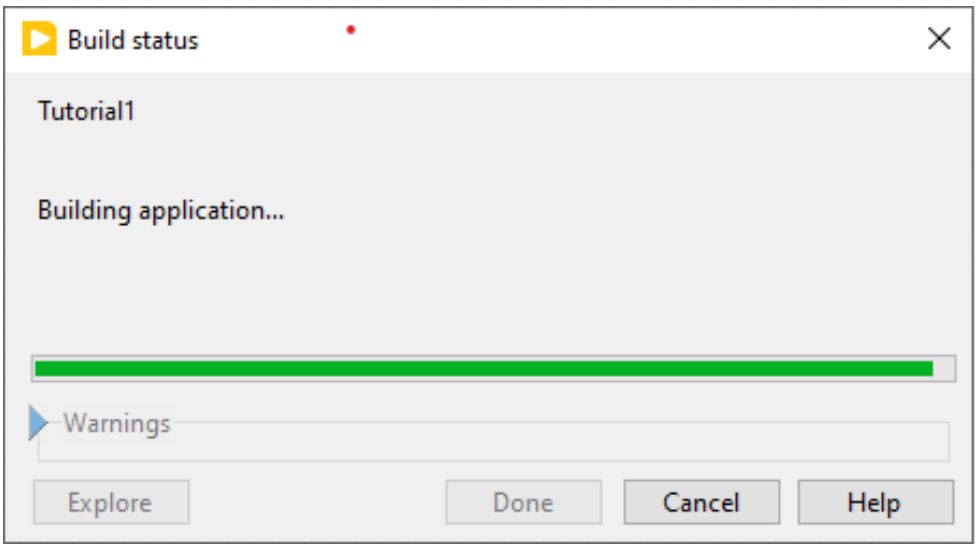


Figure 143: LabVIEW EXE Build Progression

Well done! You have successfully built your first LabVIEW program, which works with an Arduino. This is just the tip of the iceberg when it comes to what can be done on LabVIEW as well as Arduino. We will next go through some more complex examples, both looking how some of the simulation circuits could be turned into LabVIEW Programs, as well as a more complex example which you could base your main project from.

Arduino Capacitor LabVIEW Example

We will adapt the basic **Capacitor** circuit from the main document into a LabVIEW project. Something to do first, is to consider the **Current** limitations of the Arduino. The IO pins have a safety maximum of 40mA, but this is the point where they could get damaged. It is recommended to keep **Current** in pins under 20mA, but again using our safety buffer, we will keep it at 5mA max, incase of any faults/ mistakes. Therefore, at a 5V output a 1000Ω**Resistor** should be used. This means it may take a bit of time to charge. This may make it hard to see the **Capacitor** charge, so a lower **Current**, and higher **Resistance** will be used. A 5000Ω, therefore 1mA. So a 200uF **Capacitor** and a 1mA **Resistor** . In this case I used Falstad to figure what value I wanted, although its always worth doing the theoretical values too, to double check.

We will use a similar **Voltage** method as before, but this time measuring the **Capacitors****Voltage**, like

a **Potential Divider** circuit, but we can just measure its **Voltage** directly.

Like before, the first step is to place down the while loop,along with the start and stop blocks. The while loop can be found within "Structures" and the start and stop blocks within "MakerHub", then "LINX".

We also want to add a "Control" serial port, so the LINX block can find the Arduino. This can again be done by right clicking the purple node at the top left edge of the start block.

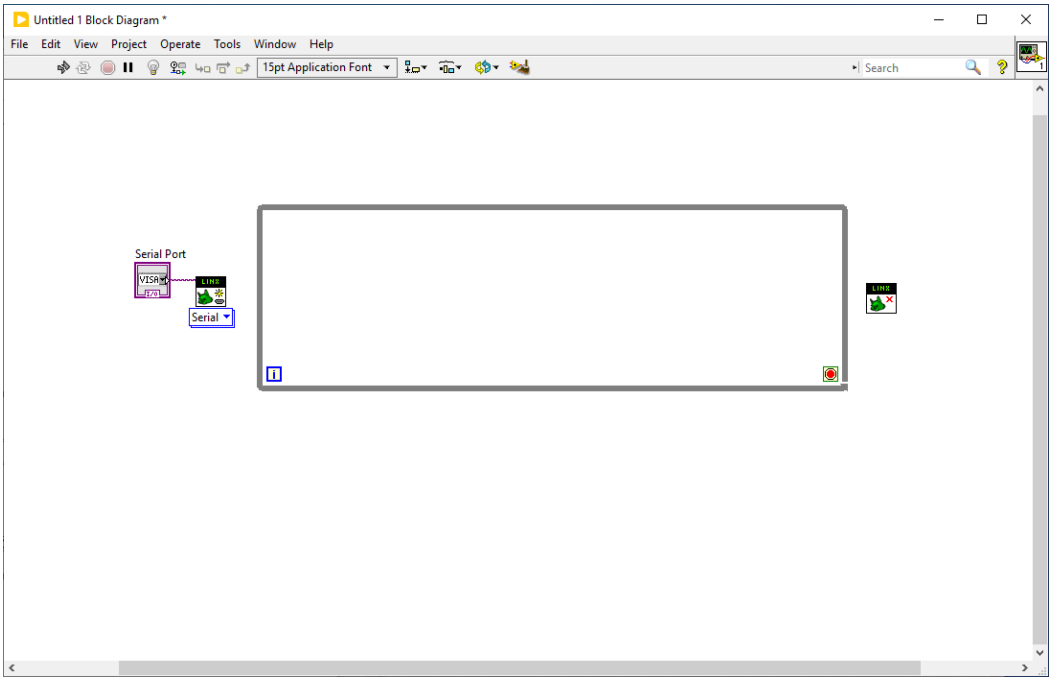


Figure 144: LabVIEW **Capacitor** Circuit Block Diagram Initial Setup

We know that the main function of the circuit is to charge and discharge a **Capacitor** reading its **Voltage**, therefore you need a way of connecting to the **Capacitor** , and then a way to read its **Voltage**. We can use an "Analog Read" block for the **Voltage** reading, and a "Digital Write" block for switching the **Capacitor** .

In this case, their order shouldn't matter too much, but we will do reading first, since it is best to understand the condition components are in before making changes. We can then simply link them up using the "LINX Resource" nodes, as well as the error lines using the correct nodes.

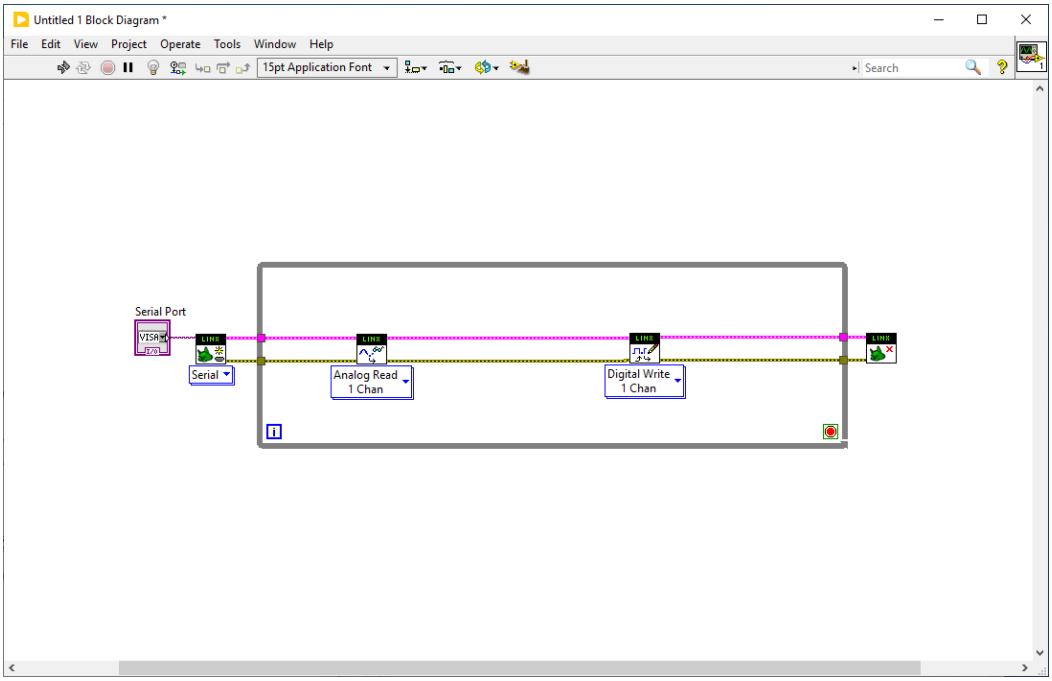


Figure 145: LabVIEW **Capacitor** Circuit Block Diagram Read and Write Blocks

Like before, the read and write functions can be found in "MakerHub", then LINX, then "Peripherals", then pick the correct section from there. Otherwise there is the search function, if you are struggling to find a block.

Now we have the blocks to make changes, we need to connect them witth the right channels to do this. We will use the "Analog Channel" and "DO Channel" nodes, right clicking to make a Control, which we place outside the loop.

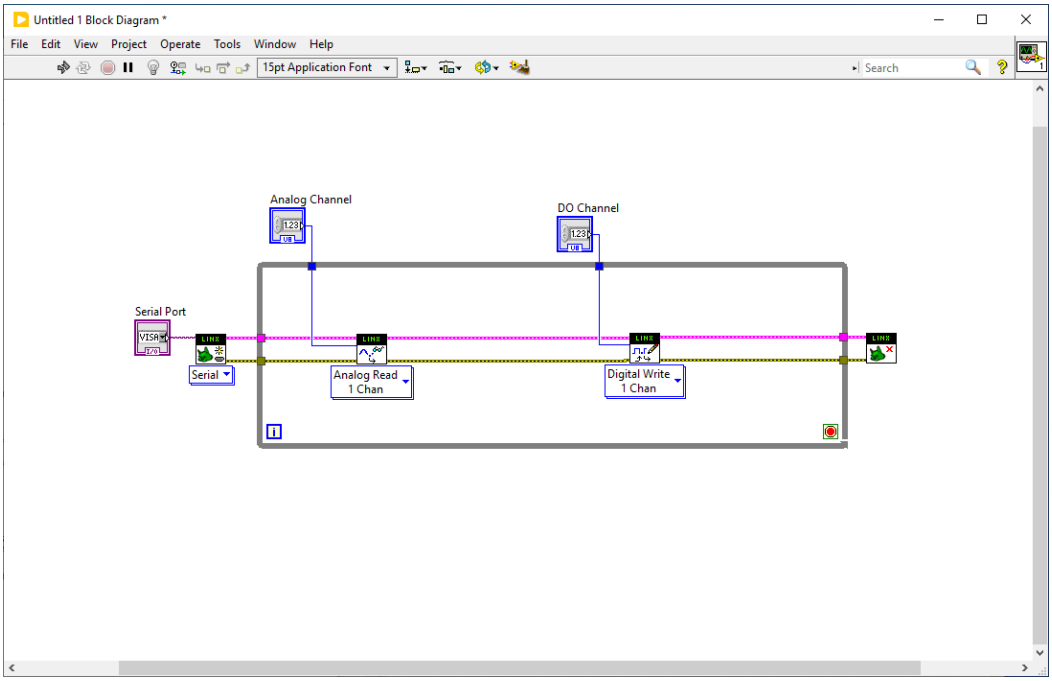


Figure 146: LabVIEW **Capacitor** Circuit Block Diagram Input Output Channels

Now the backend things have mostly been done, we should provde ways to interact. As stated, it is important to have a way to stop the program properly, so that the LINX stop function can correctly disconnect the Arduino. As stated before, we would want the while loop to stop if either that button is pressed, or if there is a LINX error. So we can use an OR gate, found within "Boolean". You should switch over to the front panel in order to add the stop button. This can be found under "Boolean". Upon switching back to the block view, this should now appear, so we can now connect these to the stop "Loop Condition". We should now have a block diagram and front panel which look like the below.

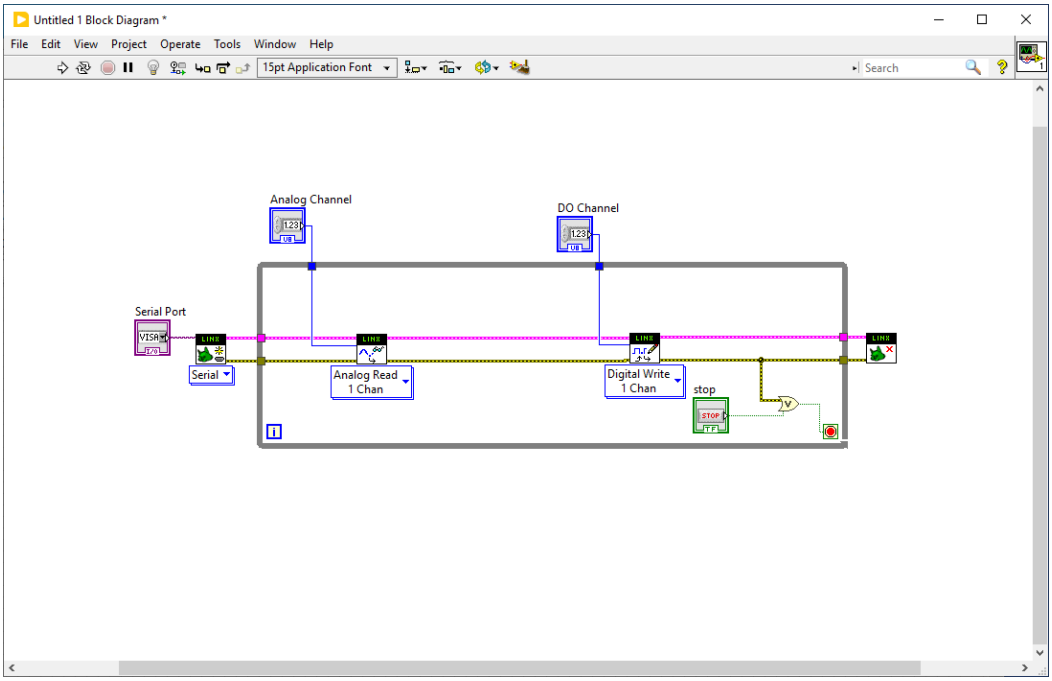


Figure 147: LabVIEW **Capacitor** Circuit Block Diagram Error/Stop Handling

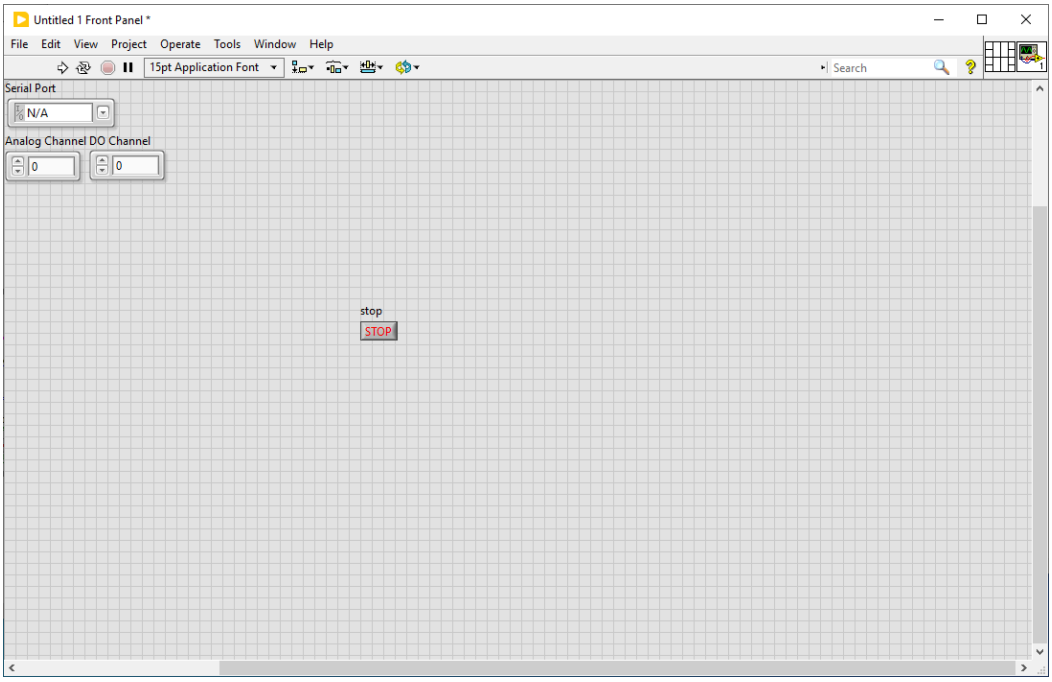


Figure 148: LabVIEW **Capacitor** Circuit Stop Button Front Panel

Now all that simply need to be added, is an output to view the **Voltage**, and a switch to switch between charging and discharging the **Capacitor** . We will add these from the front panel, and then connect them on the block diagram, keeping in mind that you can rename things. I have selected a **Voltage** display from "Numeric", the "Numeric Indicator". Switching back to the block diagram, the switch can

connect to the "Output Value" node of the "DO Channel" and the numeric indicator can connect to the "**Voltage**" node of the "Analog Read" block.

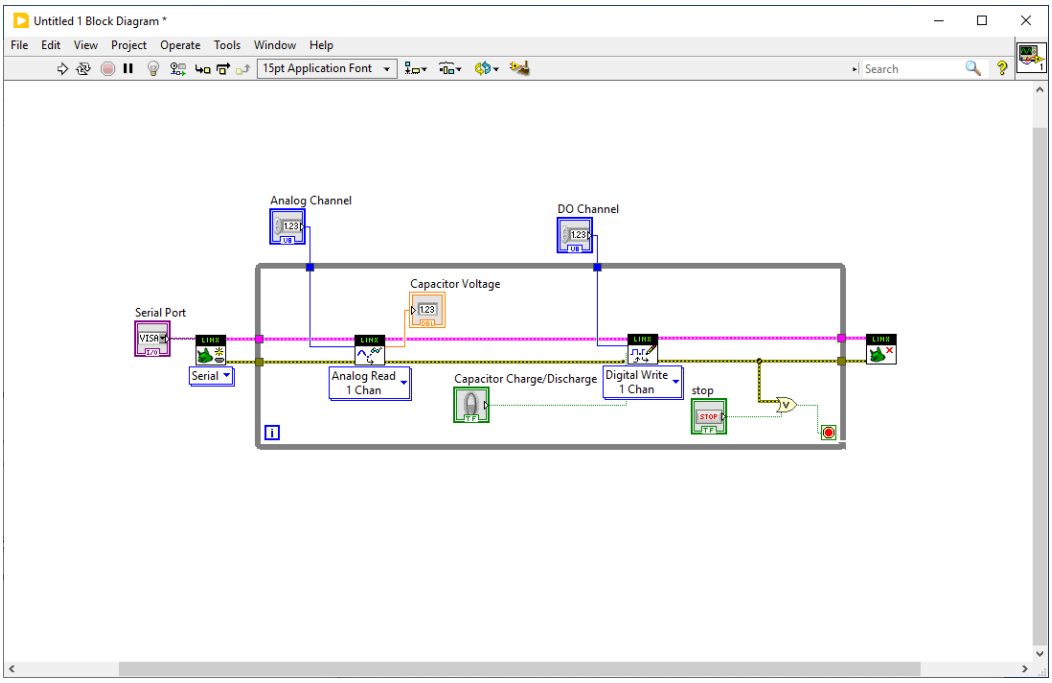


Figure 149: LabVIEW **Capacitor** Circuit Block Diagram Final

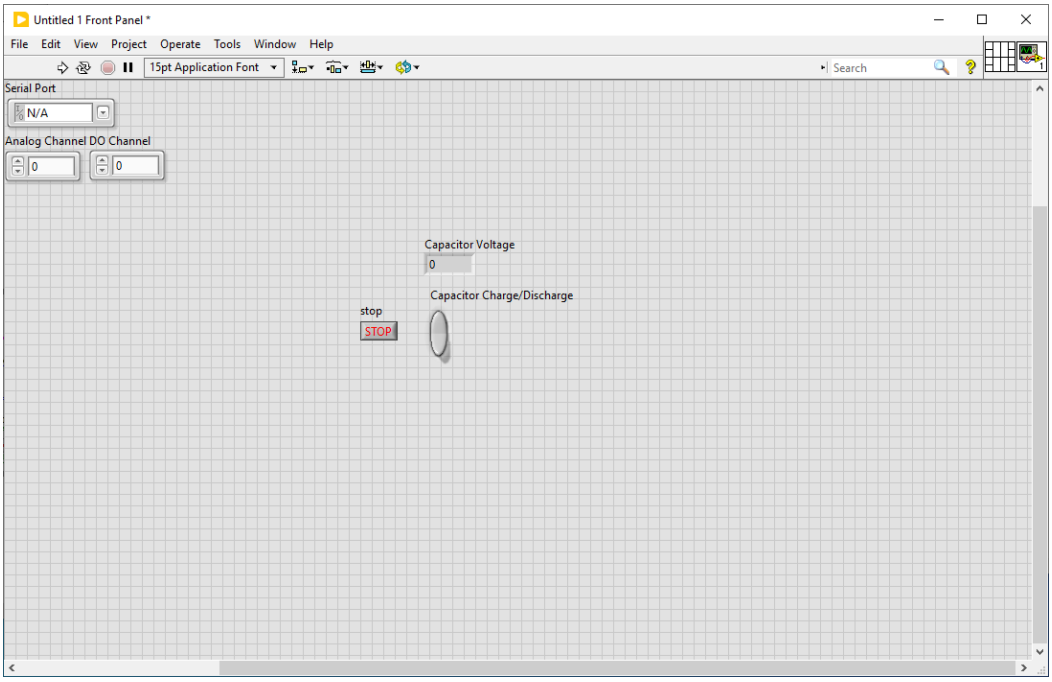


Figure 150: LabVIEW **Capacitor** Circuit Front Panel Final

This is quite a basic example, but allows you to see in reality how a **Capacitor** charges and discharges. It could be a more advanced tool, which tries to predict its **Capacitance**, if you know the **Resistor**

used.

It is based off the simulation from Falstad Circuit Simulator, found here: **Series Link**.

Arduino Heater LabVIEW Example

A more complex example, which uses a range of knowledge from the rest of this document would be a heater which sustains its temperature for a given time. It will use nested loops/ events to go through several stages.

First of all, it is worth talking through what parts may be necessary. For the heater, we will simply use a resistive heater. Given that it could take hundreds of **Watt**sto heat the air, parts would be connected to a heatsink, where the heatsink represents the area needing heated. We will also use a **Sensor** which can read this temperature, there are options to go both a digital as well as analogue route. In this case, we will go down the analogue route, using a Thermistor. With a Thermistor, we need to know its **Resistance** at a given temperature, along with how its **Resistance** changes due to temperature, to then read an analogue **Voltage** value to work out its temperature. The best way to do this is to make a **Voltage** divider, where one **Resistor** is fixed, and then a thermistor which varies.

It is important to consider that there is a limit to the accuracy of the temperature reading, due to the resolution of the Arduino's A**DC** . An Arduino has a 10 bit A**DC** , meaning that the value can be somewhere between 0 and 1023.

Due to the low **Current** output capability of the Arduino, using an external power source along with a transistor would be best to power the **Resistor** . Due to using a small heatsink, we will source no more than 2.5W, using a **Resistor** rated to at least 5W, in case a poor thermal connection is made with the heatsink. The **Resistor** used could be a 10K **Resistor** , along with a thermistor which is around 10K at 25°C. Therefore, the below equation could be used to calculate a the **Voltage** read.

$$V = R/(R + 10000) * Vcc$$

Therefore, if we use the 5V rail, and there is an ambient temperature of 25, the **Voltage** read is the below:

$$V = 10000/(10000 + 10000) * 5 = 2.5V$$

To convert this to the integer value read by the A**DC** , we can use the following equation:

$$A\mathbf{DC}Val = 1023 * (\frac{V}{Vcc})$$

$$A\mathbf{DC}Val = 1023 * (\frac{2.5}{5}) = 511.5$$

Since it is an integer, this would likely round up to 512.

Below shows how the above circuit could be connected, made using Fritzing.

We need to know the B parameter of the thermistor, to use an equation to correlate an A**DC** value with a temperature. The below is known as the B parameter equation, otherwise known as the Steinhart-Hart Formula.

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}ln(\frac{R}{R_0})$$

The above equation should use Kelvin, with T being the temperature output, T_0 being room temperature (at which it is rated at), R being its measured **Resistance**, R_0 **Resistance** at room temp, and B being the thermistor coefficient.

In this case, the Thermistor used has a **Resistance** of 10K, at 25°C, with a coefficient B of 3380K, quite a common value for such a Thermistor. Kelvin should be used though, to match up with the coefficient (25°C = 298.15°K).

Therefore, in the code we need a way to convert the A**DC** value to a **Resistance**, which can then be put into the temperature equation.

As a way of controlling temperature, the recorded temperature should be compared to the intended temperature, preventing **Current** flow to the resistive heating element using the transistor when this measured temperature is above the intended temperature.

The Analog Read helps us out, by giving us the **Voltage** value, cutting out a conversion step. We need to use this to find the **Resistance**. We have the equation from earlier below:

$$V = R/(R + 10000) * Vcc$$

This can be rearranged to find the below:

$$R = -\frac{V10000}{V - Vcc}$$

We can assume that the **Voltage** is fixed at 5V, therefore the only variable we need is the **Voltage**, which we already have. So we can use constants and numeric blocks.

We have now found a way to get the **Resistance**, which can then be put into the temperature equation.

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} ln(\frac{R}{R_0})$$

We can convert this to the below:

$$\frac{1}{T} = \frac{1}{298.15} + \frac{1}{3380} ln(\frac{R}{10000})$$
$$T = \frac{1}{\frac{1}{298.15} + \frac{1}{3380} ln(\frac{R}{10000})}$$

We can convert the fractions into decimals:

$$T = \frac{1}{0.003354016 + 0.000295858 ln(\frac{R}{10000})}$$

This has now been put into LabVIEW. It could be improved as a tool though, by allowing the user to enter these values themselves, by making what are **Current**ly constants control variables. This could allow for a system where any **Resistor** thermistor combo can be used and read to get a relatively accurate temperature measurement. It can be thought of as a closed loop feedback system.

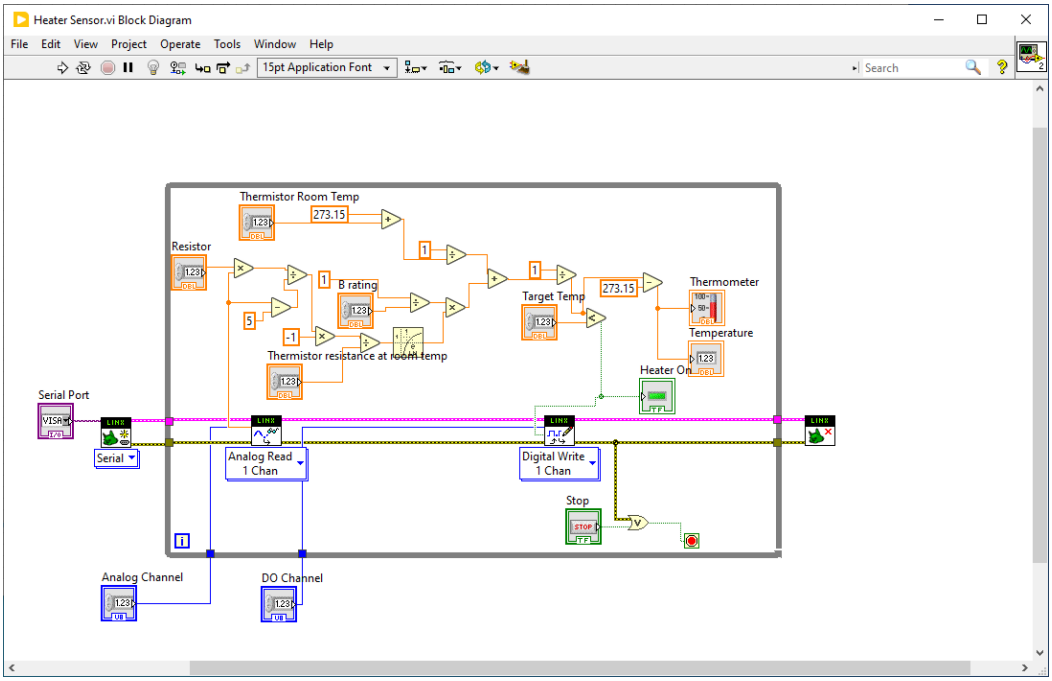


Figure 151: LabVIEW Temperature Controlled Heater Block Diagram

This may look quite overwhelming, but it is a range of numerical comparison functions being used to put both of the above equations into place. The difference is, that this circuit allows any of the characterisitcs to be altered, so it should work with any **Resistor** , and any thermistor.

This can be made into a nice panel which makes it really simple to measure temperatures from thermistors using an Arduino.

We could also add in elements to display the **Current** predicted temperature.

We can also add an **LED** which indicates if the heater is being turned on.

As seen below, you can very quickly position things for a basic Front Panel. More time could be spent to improve its visuals, and make it clearer, if this seemed useful.

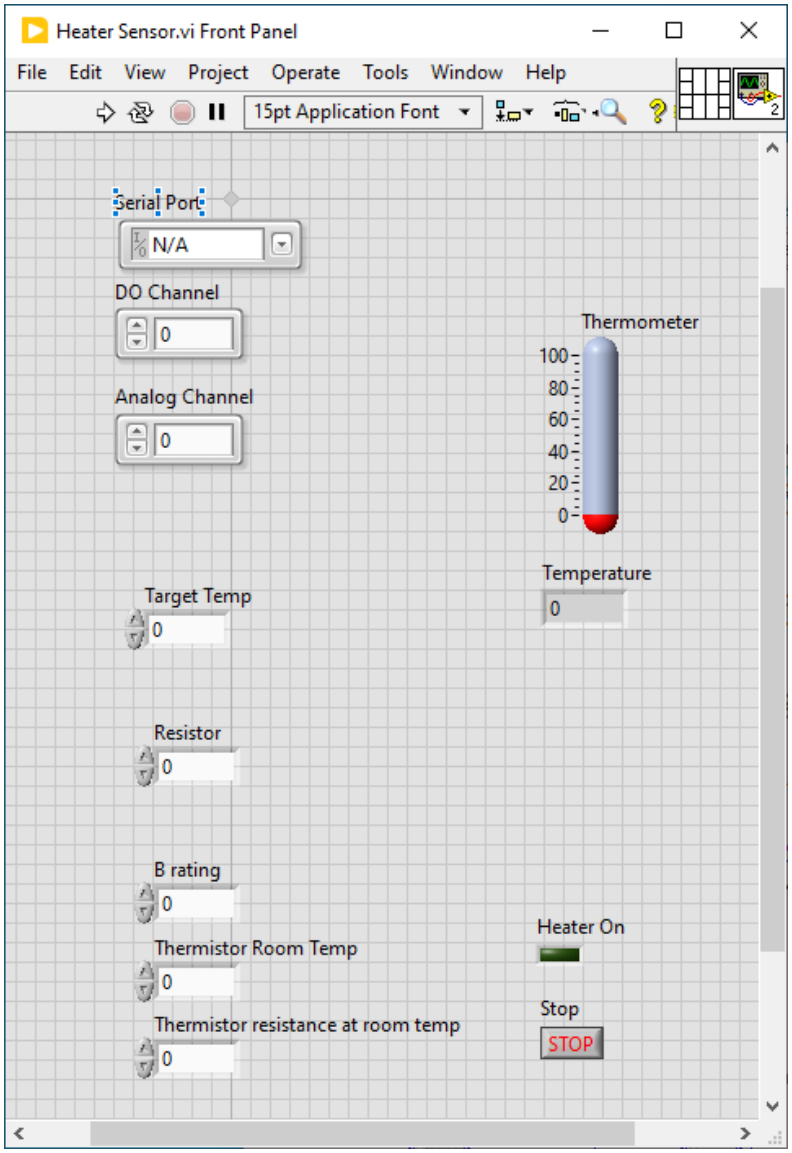


Figure 152: LabVIEW Temperature Controlled Heater Basic Front Panel

That is another project done! We haven't done much with events yet, which is going to be important for multiprocess examples, which we will look at now. This will be the final main example before the main project.

LabVIEW Case Based Example

It is likely that many projects would benefit from a case/event based systems, and this is what really allows a tool like LabVIEW to be beneficial. As it allows a multistage process to be done automatically, whether this is waiting for inputs, or waiting for set temperatures. The example you will be shown in person is events based, and uses a piece of the lab equipment, so shows a case where Arduinp LabVIEW interfacing has come in useful in WACL.

We will start with a simple case based system, which rather than running code continually, it is based upon cases changing. This could be only executing code once a certain temperature is met, or waiting upon a user to press a button.

We will start with a very basic example of turning the **LED** on and off, this time using a case structure. In this case, we will have two different states for events, with one while the application is paused, where the previous value is sent to it, until the pause button is pressed again. We can use the built in "shift registers" to pass values, rather than using more complex logic gates.

We will add the same indicator **LED** onto the front panel, as it is a useful way to ensure the **LED** is working as it should.

The final layout looks like the below. As seen, shift registers are linked to the While structure.

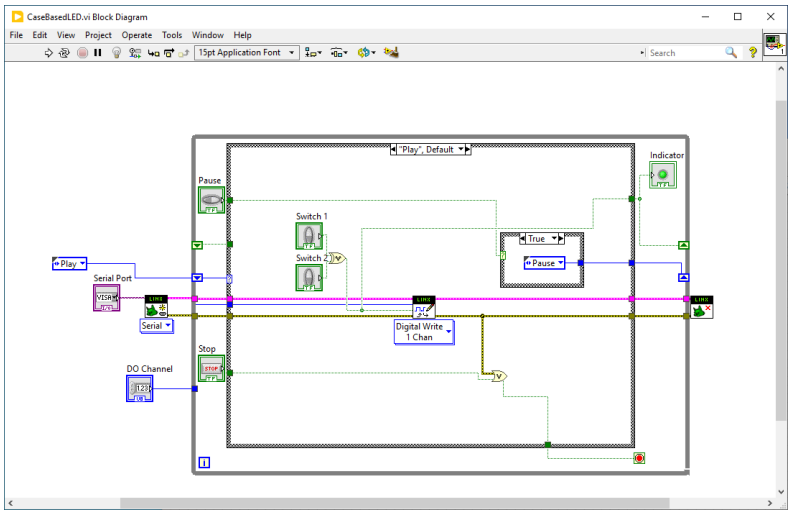


Figure 153: LabVIEW Case Based **LED** Final Block Diagram

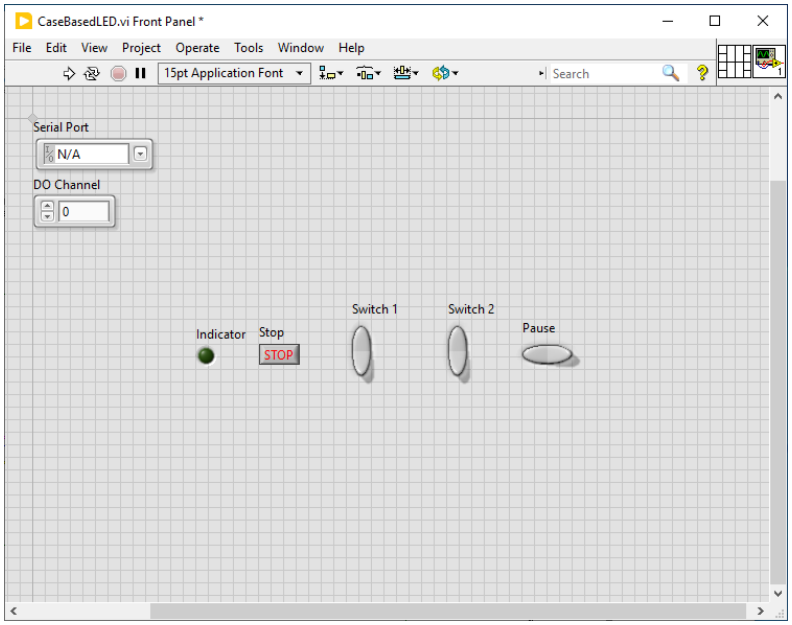


Figure 154: LabVIEW Case Based **LED** Final Front Panel

Now this very basic form has been done, we will look at a State Machine example.

State Machines are useful ways of organizing a several stage process, as when a certain event or condition is met, then the state can change, to allow for different processes to take place. This is useful for processes which have elements of linear flow to them.

This again uses a similar system to above, where it stays on a certain case until criteria is met, which will change what case it is on. Within this main case, events could be used, triggered by a set action, rather than running as soon as the specific case is switched to, which in more complex systems can reduce the layout complexity.

A case will run as long as its case is true, whereas with an events based system, you have a queue of events along with a timeout, which is the period of time of nothing happening for it to time out. Therefore there are cases where one might be more useful and others where the other is. Due to this, events are meant for shorter pieces of code.

The example state machine example will use events too, so you should know how to use a range of the main types within your programming. In this example, there are several different states, from which different events can be triggered within, particularly useful for an experiment where at different stages processes need altering somewhat.

As seen below, we have a similar case structure as before, where it is the second case structure used to

pass the next state on. Then within the main case, we have an event structure, which means a set of events could trigger for a particular stage. The example in this case will be a basic representation of a several stage experiment. Imagine that it is an experiment to look at simulating air quality throughout a day, where each stage emulates a certain time of day, altering both the temperature and lighting conditions. Then within a time of day, there can be different chemicals released into this chamber, all of which is recorded throughout the experiment.

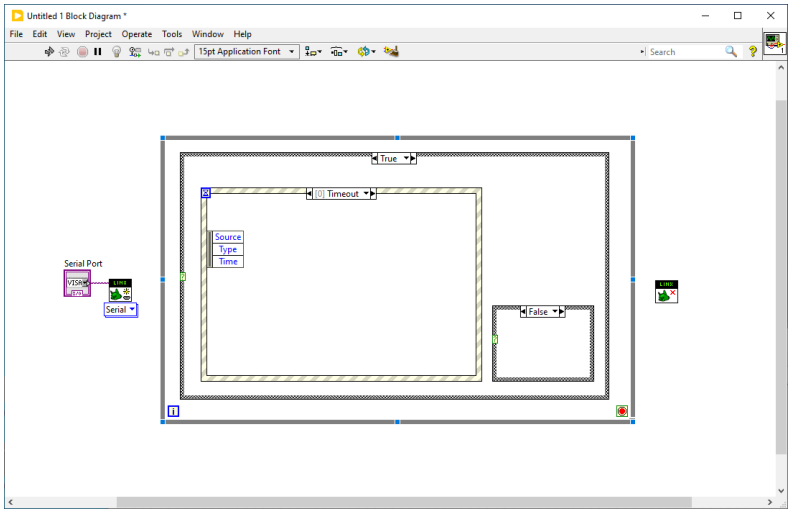


Figure 155: LabVIEW Case Based **LED** Final Front Panel

We want to cut down on the amount of blocks that have to be repeated, so some blocks will be placed in different sections, depending on whether they need to be used. So we need a heater and temperature **Sensor**, an **LED** for lighting, gas measurement **Sensor**s and several distinct machanisms which releases gas. Most of these elements are used throughout, it is the conditions which change, which will be based upon user input. In this case we are going to use the "PWM" "Duty Cycle" block for the **LED**. As a reminder, PWM is where you can have the input turn on and off at a set percentage, given by the duty, allowing an **LED** to be dimmed. This means a single **LED** can emulate a range of lighting conditions! A digital write block is used for a heater like before, where the pin could be connected to a BJT transistor to power the heating element. We can assume a similar kind of trigger is used for the chemical release, for two seperate chemicals. This time, we have decided to use a digital temperature **Sensor**. Luckily, LINX has a couple of examples, in this case we will use the "TCN75A" found within "MakerHub", then "LINX", then "**Sensor**s", then "Temp", then "TCN75A". We will need both the Open and Read block, the first of which used in the initialization steps, and the second used every runthrough. We will assume that we are using two analogue gas **Sensor**s, which give a readable **Voltage**, for two seperate gases.

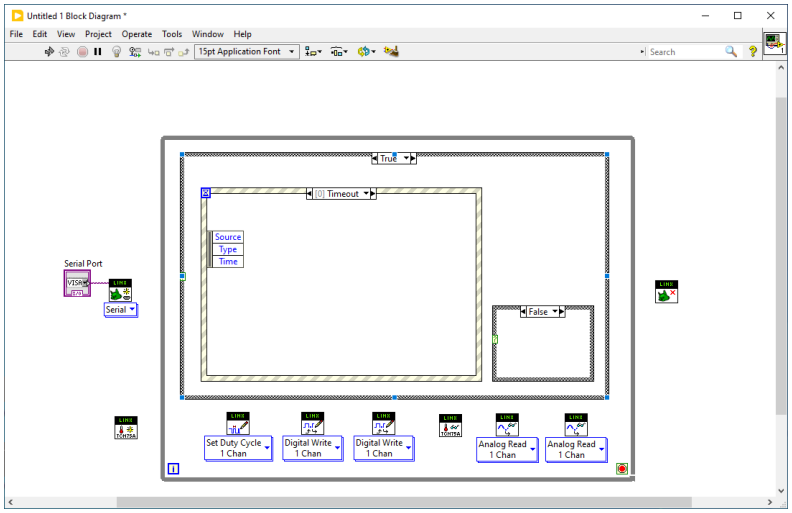


Figure 156: LabVIEW Case Based **LED** Final Front Panel

Now the inital blocks have been placed, we think more about the structure. It is important to have some sort of structure together, before you start connecting blocks. We need to have the different time phases, which as an example, I will call these "Morning", "Midday", "Evening" and "Night". We can use the same process as earlier, where an "enum constant" is used for these four cases. Like before, we can make it a type def, and then open this, to create the four case names. This can then be saved, and we now have those 4 cases. Remember for the main case structure, we need to "Add case for every value". Within each of these, there needs to be the subcase, and the event structure.

We should ensure this enum is passed through a shift register at the while loop, so the value can be passed for the next iteration, which wasn't initally done.

We should now add in the user inputs that allow all of this to happen. There should be inputs for target temperatures, and lighting at each of the phases, as well as a button to switch phase, and buttons for the chemical release. The temperature inputs are "Numeric Inputs", the lighting has to be a duty cycle value, between 0 and 1, with 1 being on all the time, therefore we could use "Vertical Pointer Slides" found within "Numeric". These should have their scales altered to between 0 and 1 in properties. Then we can simply add in a time/ case switch button and chemical release buttons, using the "Ok Button" in "Boolean", which we can rename.

Switching back to the block view, we need to place these, again within the while block, so they can be altered on the fly if needs be. It would be good to have an initial stage, where these values are set though, with a "Values ready" button. We will use this as a chance to go back into the type def, and add an additional value, called "Start".

We can edit this, add the new case in and save it, ensuring to again do "Add case for every value" with the main case. We want it to start on this case, so we can switch to it then right click, and "Make This The Default Case".

Ensure all the user input are in the main while loop, then we can add in a start button, specifically for the start stage, which connects to its subcase, to move to the morning phase.

Now this has been done, we can start to move around everything else!

First we will connect all the blocks together in the correct order. We will connect the Error Lines and LINX Resource lines, as well as connecting the channel nodes to control blocks, where you can select what pins of the Arduino to use. It would be awkward for the user to have to enter these every time they start the application, so we can instead use constants for these.

For each time of day, the right temp and lighting input can go into it, and then connect to the output node of the case structure, to then connect to the blocks for the lighting and the heating.

The value which is passed to the inputs of the blocks can be done through shift registers. There will be a comparison for the temperature, once the temperature **Sensor** read block is connected.

LabVIEW Cheatsheet

This will quickly run through some of the main block sections etc in LabVIEW, touching on a couple we have not used too. Although this doesn't go into much depth, it will help you to understand why each may be used, so you can research into them further if you feel it will aid your work.

Structures

While loop- Continues to run while "true", it will only stop when a true value is passed to its stop "loop condition".

For loop- Will run for a certain number of times, based on the loop iteration, which can be referenced.

Timed Structures- There are a few timed structures which will iterate based on a certain clock rate or timing, which may be pretty key for some processes.

Case Structure- Will run code based on what case has been passed to it, whether that is a variable name, or true or false value.

Event Structure- Will run the code within when a certain event takes place, and has a timeout clause, based on time, although can continuously run.

There are other more specific structures, although these are less likely to be needed.

Numeric

Numeric operations deal with number variables, and operations such as adding, subtracting or multiplication, as well as a range of other advanced ones. Within this, there is data type conversion blocks, mathematical constants and complex maths etc. Most are fairly self explanatory, and you can hover over the nodes to figure out what needs to be connected to which node.

Boolean

This deals with Boolean Algebra, therefore true and false. There are a range of gates, most of which have been discussed, plus some more conversion, as well as boolean constants.

Comparison

Comparison is somewhere between Boolean and Numeric, as it could theoretically be used for either, although is likely to be more useful for Numeric cases.

File I/O

Although we haven't looked into this, it could be particularly useful to automate your work, as blocks can be used to save data as files, such as data measurements or timings recorded from running the program. It also allows files to be opened, so theoretically the program could be used for some basic data analysis, for example for a program that may benefit from comparison to other data while it is running, potentially to improve the efficiency of its process.

Timing

Timing allows you to use more advanced timing functions in your program, such as converting to and from dates/ time, which may work hand in hand with the above data recording.

Arrays

Arrays can be used to store a range of values which can then be used by the program.

There are a vast range of other blocks for more specific functions, some may come in useful for specific cases, where others may never need to be touched.

0.13.2 Finding Arduino Libraries

Arduino IDE is a widely used development platform, particularly for DIY projects. This makes it considerably easier to find **Libraries** for a specific chip or **Module**.

Often, going onto the page for a **Module** should give details on its usage, as well as how to install **Libraries**.

The Arduino IDE has a built-in **Libraries** search tool, which should have many of the more common ICs/ **Module**s. This may be needed along with some custom commands for more specific modules or components, as although LINX has a few components it will interface with natively, it certainly isn't a comprehensive list.

Lets say you are wanting to use a Bosch BME688, a 4 in 1 **Sensor** capable of temp, humidity, pressure and gas measurement **Sensor**. Because it is quite a commonly used **Sensor**, you can go straight onto Arduino.

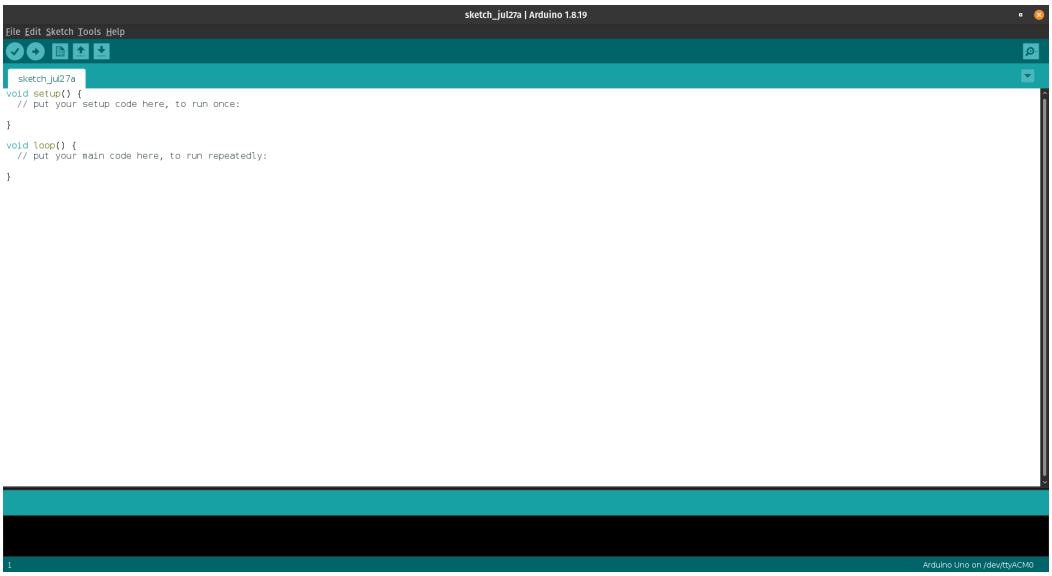


Figure 157: Arduino IDE New Program

As we can see, an empty script has opened up. We want to first go to the toolbar over to "Sketch", then from there we want to go to "Include Library" and then to "Manage Libraries".

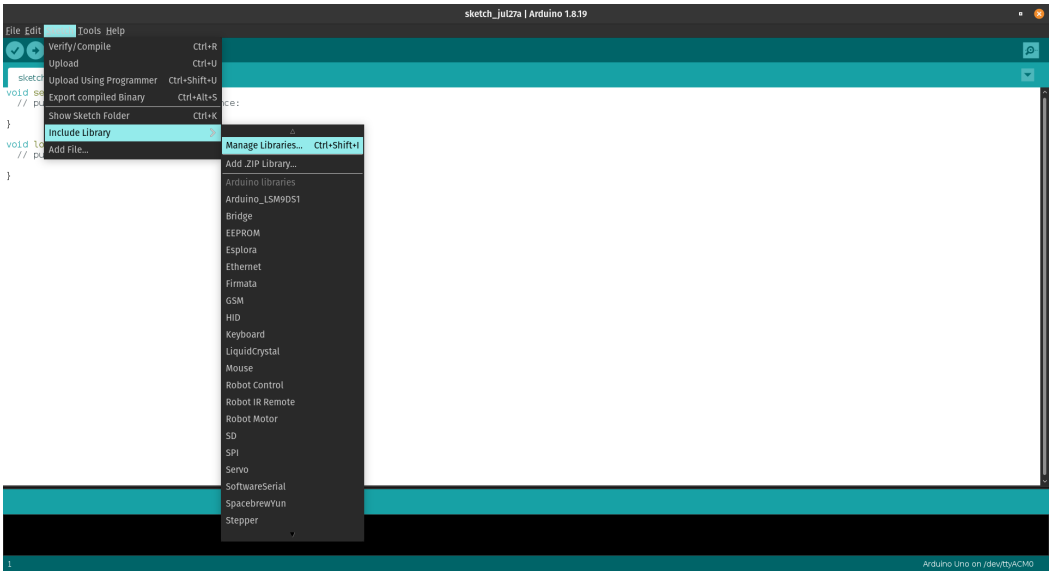


Figure 158: Arduino Manage Libraries Selection Setting

It may take a little bit of time updating the preinstalled libraries before it allows you to search for a new one.

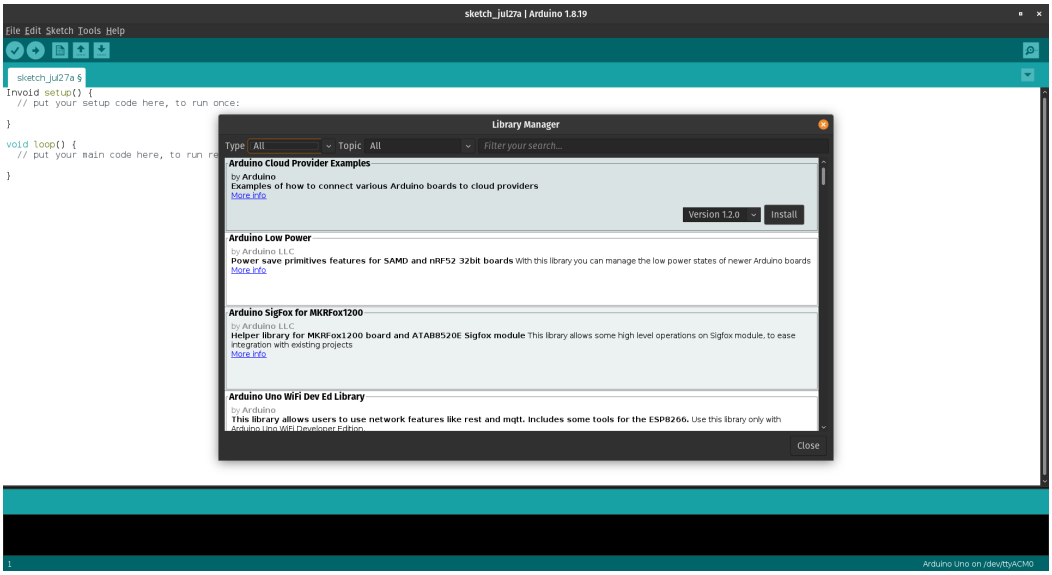


Figure 159: Default Arduino Libraries Page

From here, it is as simple as using the search bar, and searching for the module/ IC you want to interface with. As we can see, there is a package available from Adafruit, due to them designing a suitable module which either uses BME680 or BME688 ICs. Install can be clicked on, or if you want to read more about the product, you can click on "More Info" on the left side. After clicking install, you may get a pop-up that the library has other library dependencies, which means aspects of the library relies upon code

from another library not installed, so you should click "Install All".

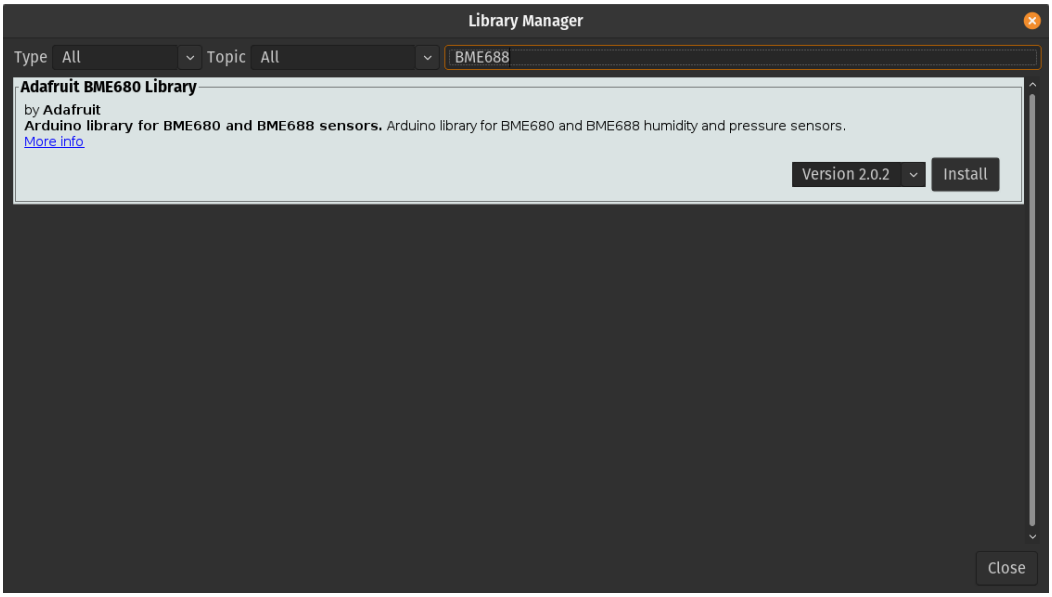


Figure 160: Arduino Libraries BME688 Search

Once installed, the library can be used within projects through "including it". There are often example scripts which are useful to check out to see how it works, and to ensure the circuit is working before its used as a wider project. To check some of these example scripts, you can go to "File", then "Examples" and then under "Examples from Custom Libraries", there may be example programs for the library you just installed. In the case of the BME688, there are three different scripts, with the test script possibly being a good example to check out.

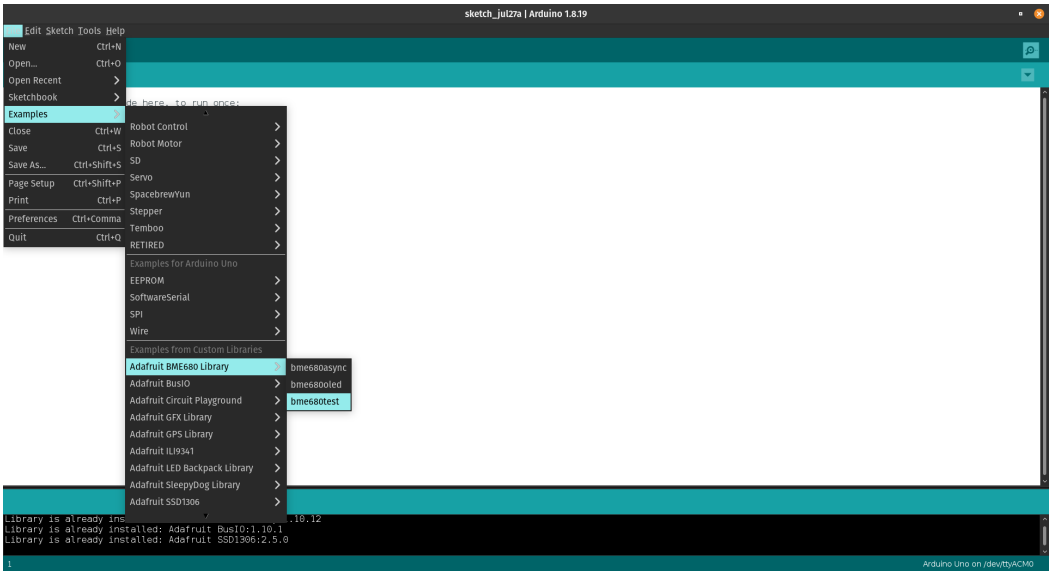


Figure 161: Arduino Finding BME688 Example Script

On clicking it, it should open a new script, configured to work when uploading onto the Arduino directly from Arduino IDE. Although there could be a few more comments, we can see where pins need to be connected, and the scripts needed to do all the main functions. Typically checking the "More Info" button mentioned before should have more documentation on the example script or using the library as a whole, often linking to a GitHub page.

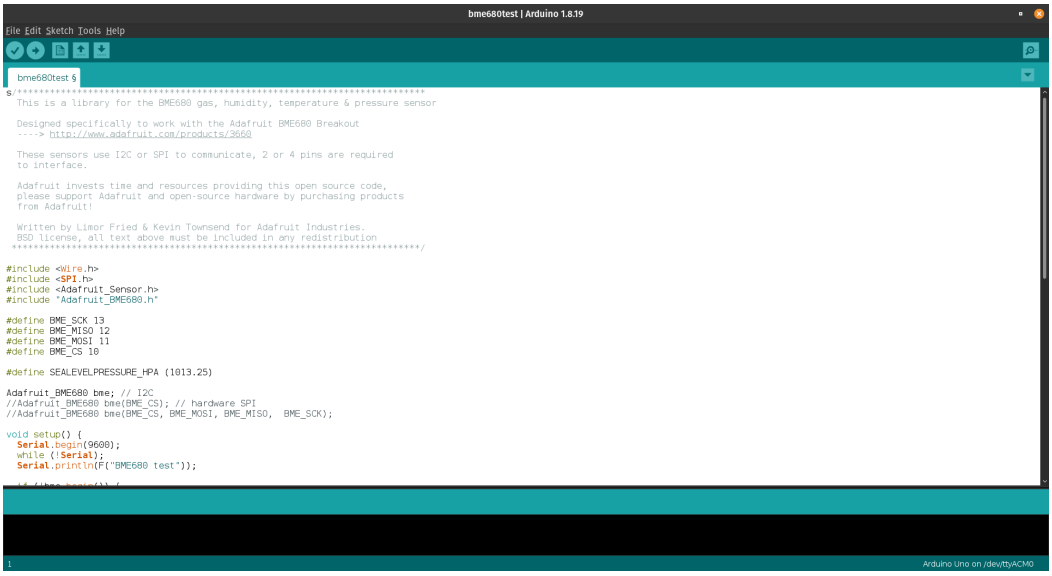


Figure 162: Arduino BME688 Example Script

We will now go through one of the steps that can be taken if you cannot find a library through the Arduino search. Since it is already installed, I will quickly show the steps to delete a library, which is handy to do if you know you are not going to use a package, and it is using up space/ creating clutter.

Libraries are usually stored in "Sketchbook", which we can find the location of which by checking preferences.

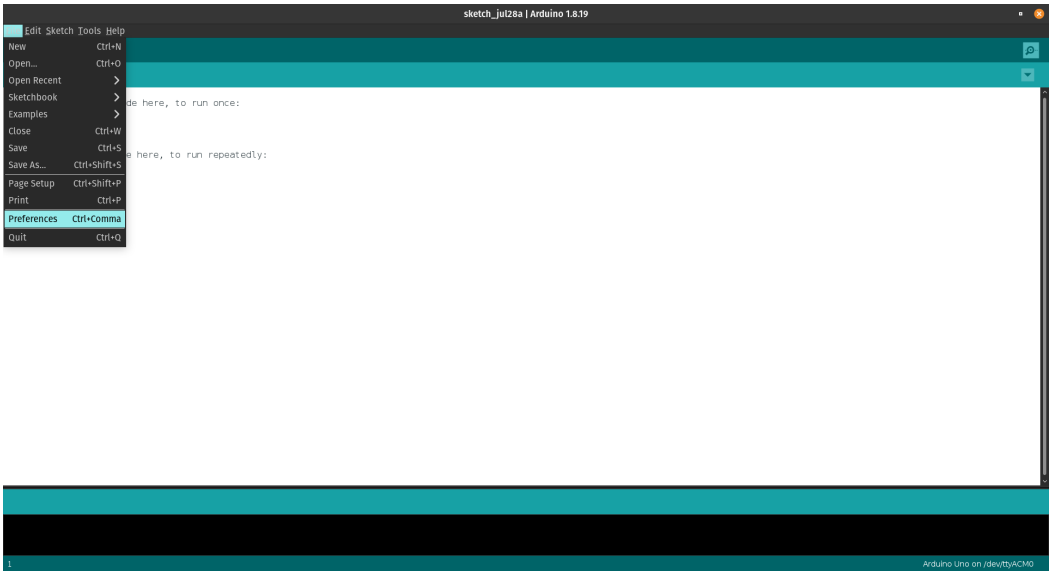


Figure 163: Arduino Find Preferences

After clicking on this, we can find the Sketchbook location found at the top. You should then open your file explorer to navigate to this.

Within this location, I found a folder called "libraries", which contained any installed libraries. I have now found the folder associated with the BME688, called "Adafruit_BME680_Library", so I have deleted that. You should remember that there may also be the dependencies as separate libraries. In this case I will just delete the specific BME688 one, given I am reinstalling it anyway.

If you search for the module or part along with Arduino library, you should find what you are looking for. It turns out that the manufacturer of the **Sensor** has written their own library for using the BME688, so I will show you the steps to install that. It is on GitHub directly.

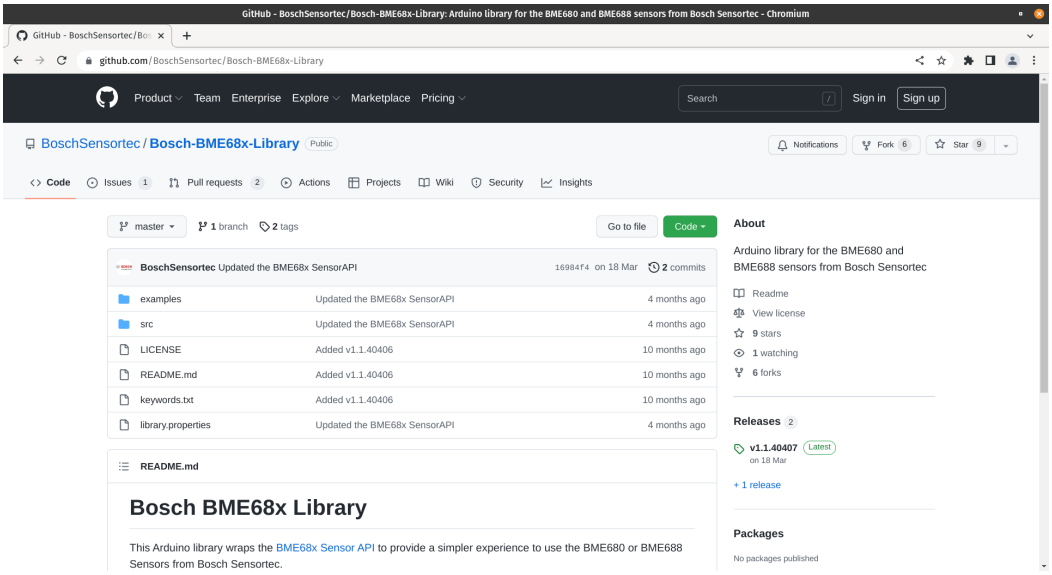


Figure 164: BME688 Bosch Arduino Library Github

To allow Arduino IDE to install this library, we should download it as a ZIP file, and provide Arduino IDE with its location. We can do this by clicking on "Code" and then "Download ZIP".

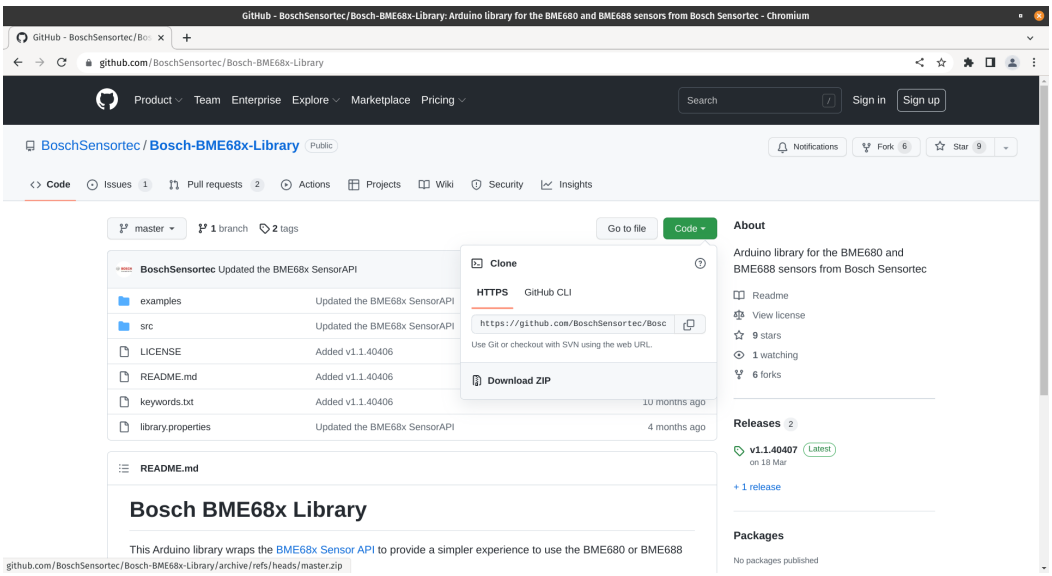


Figure 165: BME688 Bosch Arduino Library Github Download

We can then go back onto Arduino IDE, then go to the "Sketch" button on the toolbar, then "Include Library" and then "Add .ZIP Library...". You should then find where it was installed. It should now install, and you can check within "Sketch" and "Include Library" to make sure it has installed, which we can see in this case it has.

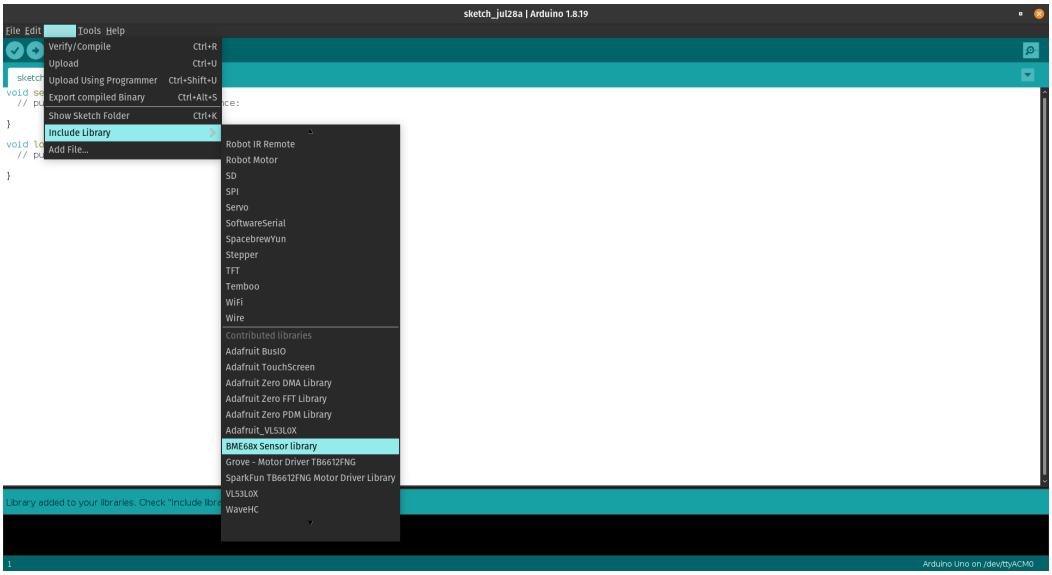


Figure 166: Arduino Library check

You can take similar steps as before to check if there are example programs for the library. In this case, I have seen that although the examples are a bit longer, more comments mean that its easier to understand what is going on!

0.13.3 Additional Steps for Linux + Mac Users

There are some additional steps that need to be taken to get the program to properly work. Both Arduino IDE as well as LabVIEW themselves work on both operating systems, but unfortunately LINX requires some Windows specific libraries, which means it is best to run both LabVIEW and Arduino IDE. This document was produced on a Linux machine (running PopOS, a Ubuntu derivative), so I can confirm these steps have been tested successfully for a Linux machine.

We first need to ensure that it is doable from the device you are using. Although it may work with devices that don't meet these specs, I would personally recommend at least a quad core processor, along with 8GB of RAM and at least around 50GB of storage free. If all of these hold true, we should first download a Windows 10 Iso, which is what is needed to install Windows 10. This can be found here: **Series Link**.

We will also need to install Oracle VM VirtualBox, which is the software required to run a Windows 10 Virtual Machine. This can be found here: **Series Link**

You should go through the steps of installing VirtualBox, which may vary somewhat between operating systems.

Now we should open and configure VirtualBox. As stated, I have actively used a Windows 10 Virtual Machine on this system, which is why there is one there when I open VirtualBox.

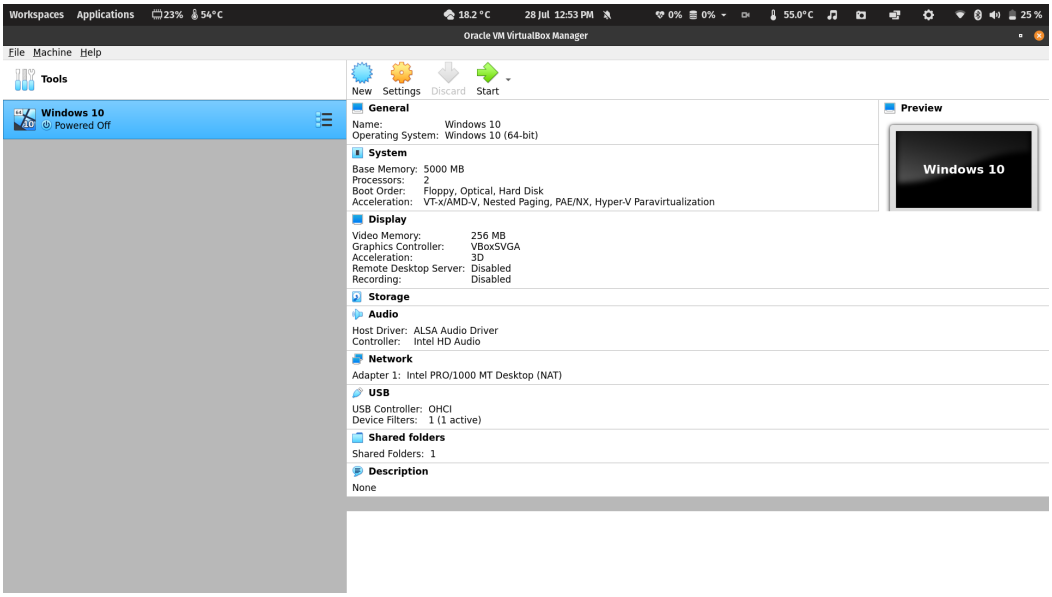


Figure 167: Virtual Box

But you should select "New", the name can be whatever you like, in this case I will call it LabVIEW Windows.

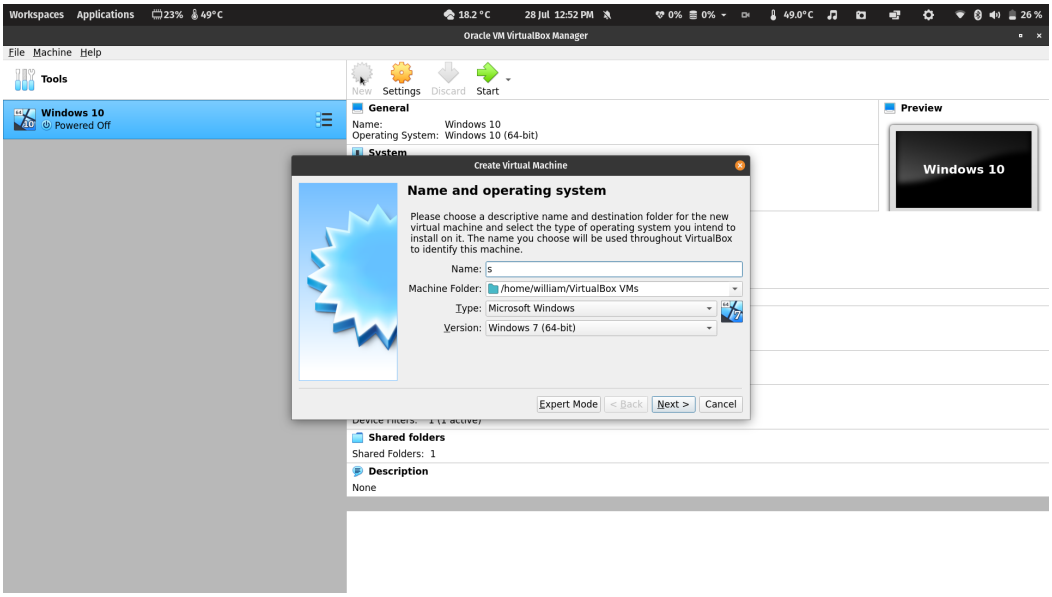


Figure 168: Virtual Box New Machine

We want the "Type" to be "Microsoft Windows" and the "Version" to be "Windows 10 (64-bit)".

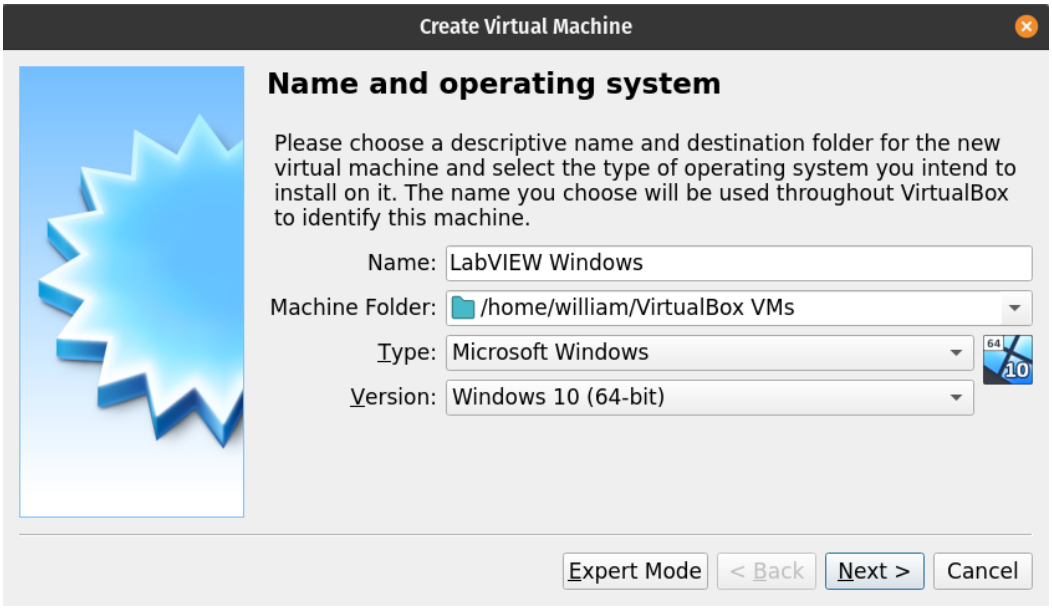


Figure 169: Virtual Box New Machine Initial Details

The next screen is Memory Size, which is the amount of your RAM which is configured for the Virtual Machine while it is running. You need to ensure to use no more than your desktop uses anyway. You should set aside a minimum of around 4096mb, as that is the recommended minimum for LabVIEW. The coloured bars give you an idea of how much space might be too much.

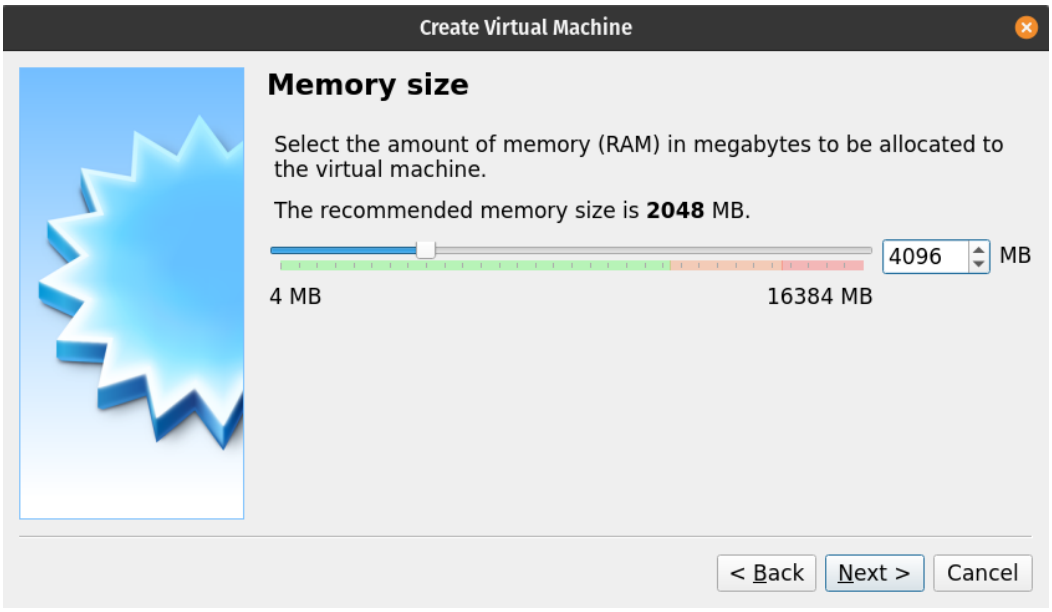


Figure 170: Virtual Box New Machine RAM Allocation

You then go onto the page for creating a Virtual Hard Disk. As seen, the recommended amount is 50GB, although you may get away with 40GB including the LabVIEW installation if you really needed

to. You should click on "Create a virtual hard disk now" and then press "Create".

Figure 171: Virtual Box New Machine Hard Disk Allocation

A VirtualBox Disk Image should be suitable, and then dynamically allocated is helpful, as space is only used up if needed, it might be somewhat slower to use though.

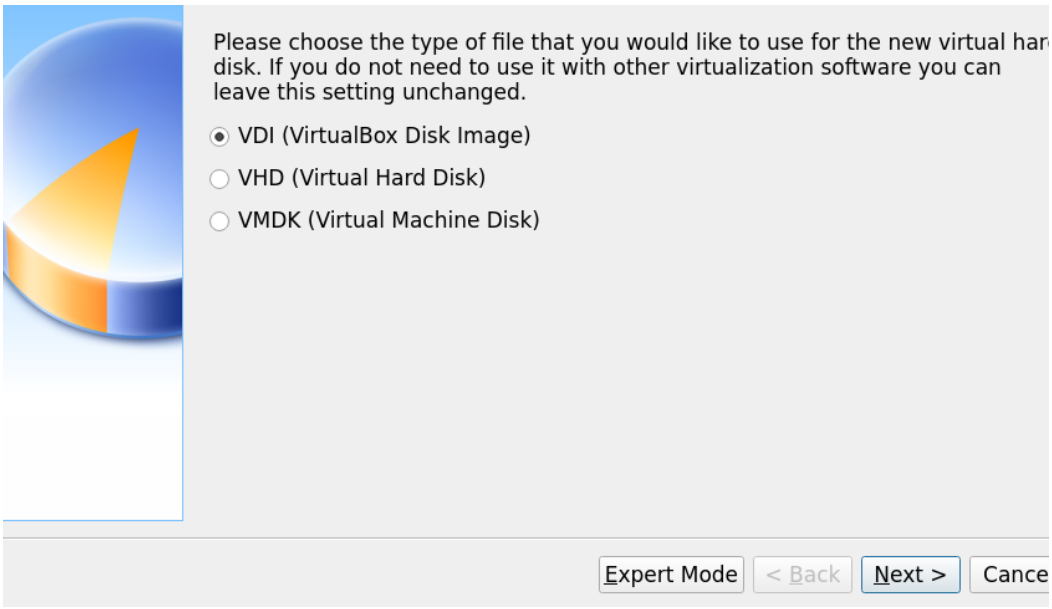


Figure 172: Virtual Box New Machine Hard Disk Type

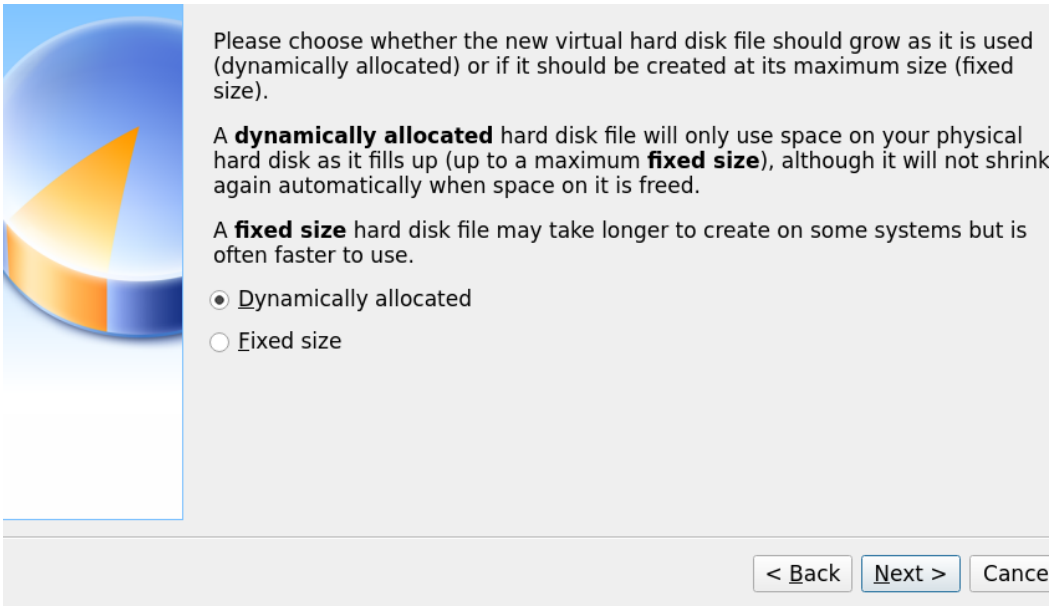


Figure 173: Virtual Box New Machine Hard Disk Dynamic/ Fixed

You should then select both the location and the size, I will stick with the recommended size of 50GB, which should leave around 15GB free on the Virtual Drive after the required installations, plenty for using LabVIEW with Arduino. You can then click "Create".

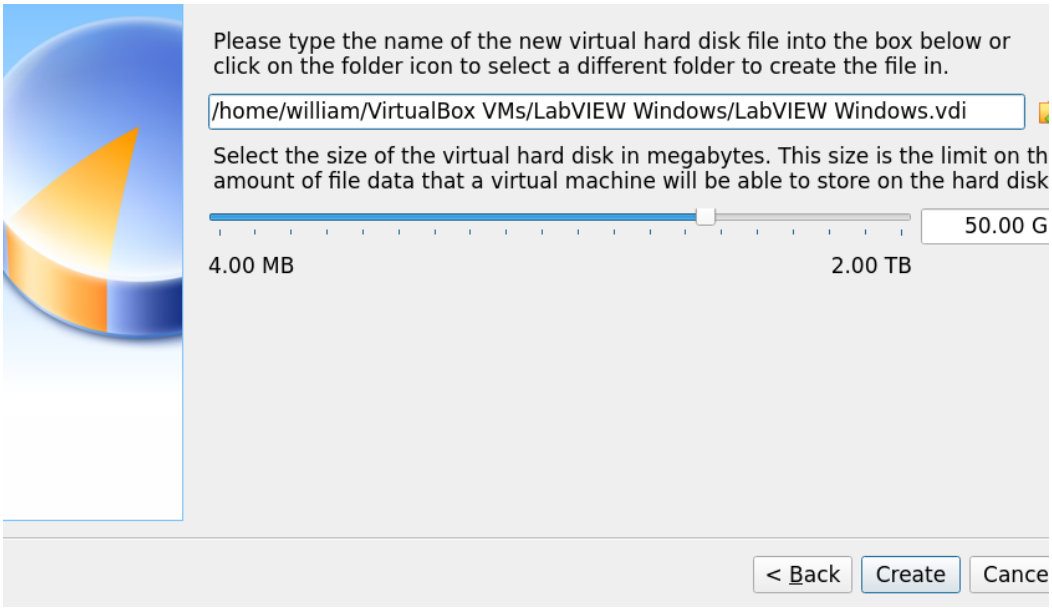


Figure 174: Virtual Box New Machine Hard Disk Dynamic/ Fixed

Now the initial setup should have been done, but there is still some things that need doing. This includes actually installing the Windows 10 ISO, and allowing an Arduino to be used with the Virtual Machine. We need to go onto the settings for our newly created VM to do this.

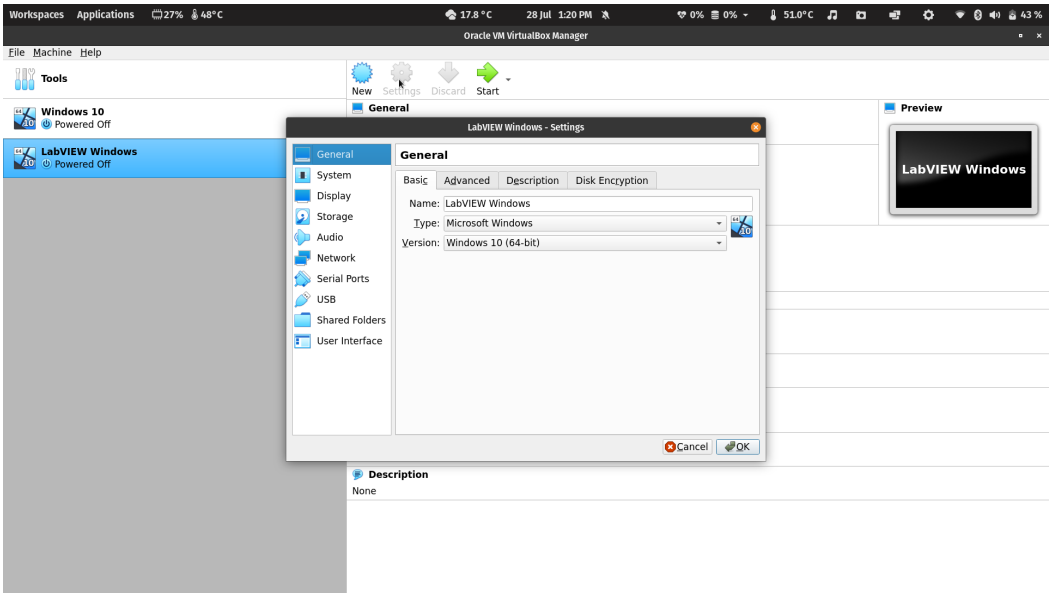


Figure 175: Virtual Box New Machine Settings

We first want to go onto Display, to "Enable 3D Acceleration" to then increase the available "Video Memory" to 256mb, as 128mb could be very restrictive.

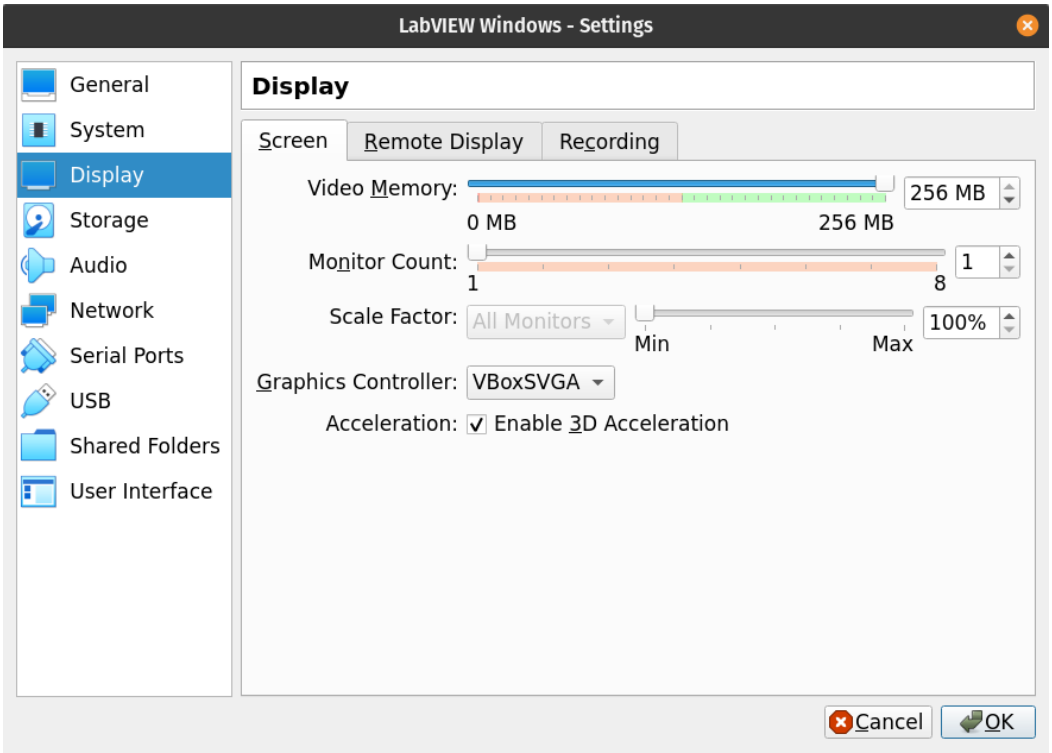


Figure 176: Virtual Box New Machine Display Settings

Then you should next go onto "Storage", click on the "Empty" disc, and then the disc icon on the right, to then click on "Choose a disk file" and look for the Windows ISO file.

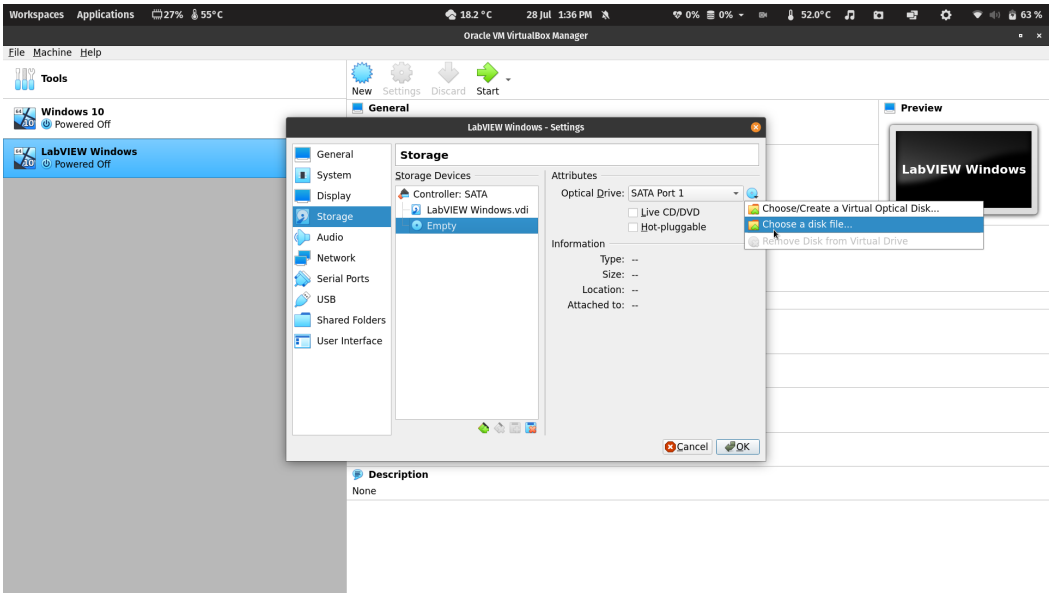


Figure 177: Virtual Box New Machine Storage Settings

The main last thing to do in settings is ensuring that the Arduino can be interfaced with through the VM. You should first make sure that Arduino IDE has been installed on your own operating system, so that the Arduino drivers are installed. Then you should plug the Arduino in. You should go onto the USB Page, click on the button to the right with a USB plug and plus, and then select Arduino.

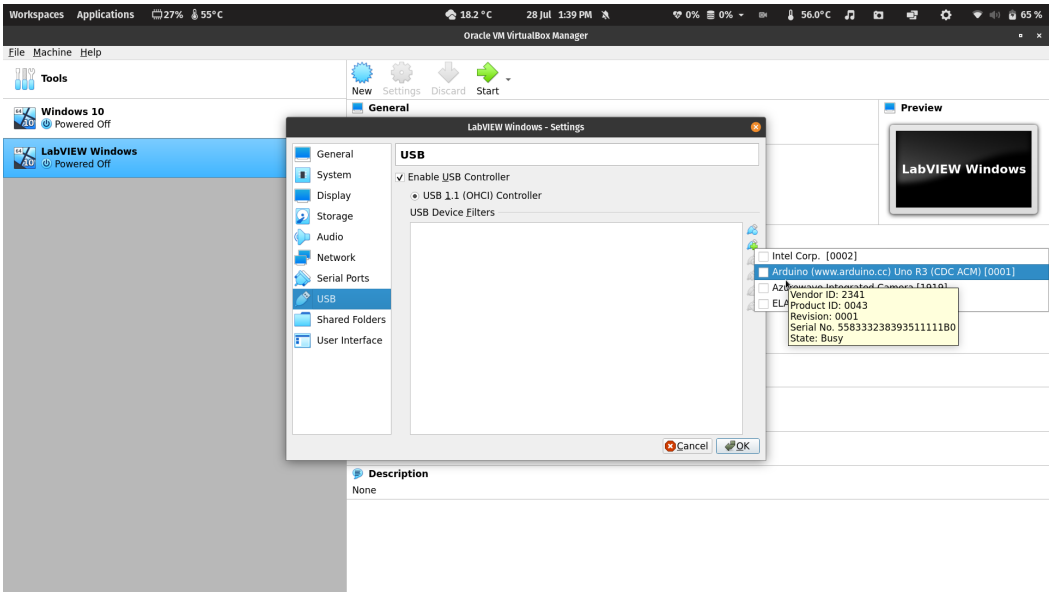


Figure 178: Virtual Box New Machine Arduino USB Setup

You should now be able to click start, and then follow the Windows 10 installation. Once it is correctly

installed, you can follow the above steps for setting LabVIEW and Arduino IDE in Windows.

Electronics Glossary

AC Alternating **Current** is a **Current** which continually changes direction as it flows. 1, 12, 13, 14, 20, 23, 25, 29, 30, 33, 37, 38, 79, 82, 83, 164, 167

Ampere The unit for **Current**, often shortened to Amp. 4

Analogue It is the representation of information like **Voltage** as a continuous and variable value. 25, 26, 31, 32, 47, 48, 52

Binary A mathematical system which uses 2 rather than 10 as a base, often used in computing, due to its compatibility with boolean algebra/ digital electronics. 26

BJT It is short for Bipolar Junction **Transistor**, where the **Current** flow between the collector and emitter is the **Current** flow through the base amplified. 28, 29, 32, 57, 83

Boolean Algebra A field of mathematics only using true or false comparisons. 26, 52

Bridged A term used commonly for when components are soldered together, often accidentally, which could cause a short. 63

CAN This stands for Controller Area Network, and is meant for communication between embedded devices for a larger system or process. 40, 42

Capacitance The ability to store an electrical charge. 4, 15, 16, 30, 36, 37, 57, 83, 136, 164, 166, 167

Capacitor A component which stores energy as electrical charge. 1, 14, 15, 16, 17, 18, 28, 35, 38, 57, 63, 64, 75, 80, 81, 83, 87, 131, 132, 133, 134, 135, 136, 165, 168

Control Loop Uses a combination of components and processes to adjust a value or process. 3, 7, 32

Core A component which transfers power from one source to another through Electromagnetic induction, which are typically highly magnetic. Sometimes a term for a central or key part of a component. 23, 31

Coupled Components which are connected electromagnetically through induction, electrostatically or optically. 23

Current An electrical flow due to directional movement of electrons. 4, 5, 6, 8, 9, 11, 12, 13, 15, 16, 18, 20, 23, 24, 25, 28, 29, 35, 36, 47, 49, 57, 62, 63, 67, 71, 73, 74, 75, 76, 79, 81, 83, 84, 85, 91, 106, 114, 115, 122, 131, 137, 138, 139, 140, 162, 163, 164, 165, 166, 167, 168, 169

Darlington Pair A **Transistor** which internally uses a pair of BJT **Transistor**s to allow for higher **Current** amplification than a single BJT. 29

Datasheet A document which provides the specifications for a product or component, often including how it can be used within a circuit. 46, 73, 75, 76, 86

DC Direct **Current** is **Current** with a constant direction of flow. 1, 12, 13, 14, 25, 26, 27, 29, 33, 35, 38, 47, 79, 80, 82, 83, 137, 138, 167

DIAC A type of **Diode** which conducts **Current** after a certain **Voltage** (breakover **Voltage**) is momentarily reached. 20

Die A piece of semiconducting material on which a functional circuit such as a **Microcontroller** or IC is fabricated. 27, 70

Digital It is where data or signals are expressed either as 0 or 1, enabled by transistors. 7, 14, 25, 26, 27, 30, 31, 46, 47, 52, 57, 79, 101

Diode A component which limits how **Current** can flow. 1, 20, 21, 22, 23, 31, 33, 79, 80, 82, 83, 85, 87, 163, 165, 167, 168, 169

DIP This stands for Dual In-line Package, but has become a common term for components with pins that go through a PCB, or into a breadboard. 27, 58, 61, 78

Doped The process of adding an impurity to a material to alter its characterisitcs. Frequently used to alter the electrical/ conduction properties. 20, 31, 85, 165

Driver Software necessary for controlling or interfacing with a device external to the computer. It is also a term used for a part which provides power for components, such as motors. 41, 43, 46, 83, 84, 91, 92

Electromagnet This is typically a coil of wire which is magnetised only when **Current** flows through it. 29, 84

ESD This stands for Electrostatic Discharge, and is the sudden flow of **Current** between two electrically charged objects, typically through contact. Steps should be taken to avoid this, which can cause damage. 46

Farad The unit for **Capacitance**. 15, 18

Ferritic Metallic materials which contain a portion of iron. 23

Flash This is a memory type which retains its data without power, which can be read from and written to. 47

Flux A substance added to solder to improve its wetting ability, by removing oxides. 63

Gate It can be thought of as a trigger to allow **Current** flow in a transistor. Where in boolean algebra, it is an idealized device implementing a Boolean function. 27, 28, 30, 52, 54, 55, 56, 83

Gauge A standardised measurement for the diameter of wire. 62

Generator A device which converts mechanical energy to electrical energy. 35, 82, 84, 85

GPIO This stands for General Purpose Input Output and is typically used to describe microcontroller pins that can be used for a range of purposes, and sometimes a range of protocols too. 41, 47

Grid A network transferring energy as either gas or electric. 12, 14

Heatsink An object used to absorb and dissipate unwanted heat, typically made out of aluminium or copper. Can be passively cooled through natural air **Current**s, or actively cooled through a liquid cooling loop or fan. 66, 78

Henry The unit for **Inductance**. 18

I2C This stands for Inter-Integrated Circuit, and is a protocol for serial communication between two devices/ ICs. There are seperate lines for data to be written and read. 31, 40, 42

Impedance This is the effective **Resistance** of an **AC** circuit, being a combination of both **Resistance** and reactance. 37, 75

Inductance The ability to generate an electromotive force due to changing **Current** flow. 4, 18, 19, 30, 36, 37, 164, 166, 167

Inductor A component which stores energy in the magnetic field. 1, 18, 19, 23, 38, 57, 64, 76, 87

IP This stands for Ingress Protection, and it is an objects ability to block foreign bodies, often used as a rating for devices for harsher environments. Also an acronym for Intellectual Property. 66

Jack A term widely used in audio and video as well as power, for a typically rounded **Port** or socket. 35

Joules The unit for energy. 15

Jump A wire meant to temporarily connect or short components, regularly used with breadboards. 59

Lamination It is the process of using stacked sheets of core material rather than a single "block" to limit eddy **Current**. 23

Lead Time A term used to describe the time it takes between a supplier making an order and recieving it from a manufacturer, commonly used in the supply chain. 65

LED This stands for Light Emitting **Diode**, which are **Diode**s **Doped** so that they emit light, typically in the visible spectrum, depening on use case. 20, 22, 23, 30, 33, 46, 59, 67, 68, 69, 70, 71, 72, 73, 77, 82, 101, 102, 103, 106, 111, 112, 116, 117, 119, 120, 121, 124, 125, 140, 142, 143, 144, 145

Libraries A collection of scripts/ programs for a specific use, such as interfacing with a certain **Sensor**. 77, 78, 91, 148

LiDAR This stands for light detection and ranging, using a laser to measure distance by the time it takes to reflect back to the **Sensor**. This method can be known as ToF (Time of Flight). 32

MEMS This stands for MicroElectroMechanical Systems, and is a combination of micromechanical components such as springs or levers, along with microelectronic circuitry, such as **Capacitor** s, widely used for **Sensor**s. 30

Microcontroller An integrated circuit with built-in microprocessor, memory and other IC elements, which can be thought of as a small computer for a specific or embedded task. 26, 27, 29, 30, 31, 32, 46, 47, 48, 64, 78, 83, 84, 87, 91, 92, 163

ModBus The most common protocol for communication between industrial devices. 41

Module A Module is a distinct assembly of parts for easier use in a system. 30, 31, 38, 43, 46, 59, 64, 65, 78, 87, 88, 89, 148

MOSFET This stands for Metal-Oxide **Semiconductor** Field-Effect **Transistor**, and is a **Voltage** controlled switch, where a certain **Voltage** on its gate will cause **Current** to flow between its

drain and source. 28, 83

Motor A device which outputs a motive force, typically converting electrical energy first to magnetic and then to mechanical. 14, 30, 35, 84, 85

Multimeter A common piece of test equipment typically for measuring **Current**, **Voltage** and **Resistance**, sometimes with extra functions such as **Capacitance** or transistor **Current** gain measurement. 57, 63

OpAmp This stands for Operational Amplifier, and is an IC for **Voltage** amplification, which can be used in a range of configurations, depending on intended output. 48, 51

Open-Source Software or hardware with its original source code or files made freely available to use, contribute to or modify. 77, 86, 89

Optoisolator A component which transfers an electrical signal using light, between two isolated circuits. 33, 34

Parallel It is a circuit configuration where a circuit board/node splits into several branches, with **Current** being divided inversely by a ratio of each branches **Resistance**. It can also stand for a communication bus of multiple con**Current** data lines. 1, 8, 9, 10, 11, 12, 16, 17, 18, 40, 41, 59, 67, 73, 80, 169

Parasitics These undesirable characteristics affecting components or the circuit, typically being **Inductance** or **Capacitance**. These tend to have more of an effect with high speed electronics. 36

PCB This stands for printed circuit board, and is a manufactured board with the necessary connections and solder points for a given circuit. 59, 66, 86, 88, 89

Period An interval of time regarding successive occurances, such as with a wave. 12, 25, 63, 65, 66

Port This is a point where communication can be done from, usually with a select number of protocols. 41, 43, 101, 104, 165

Potential Divider A **Resistor** configuration used to get an output which is a fraction of a **Voltage**. Widely used in sensing. 1, 7, 12, 132, 166

Potentiometer A variable **Resistor** acting as a **Potential Divider**. 1, 7, 8, 32, 38

Protocol A set of rules regarding data communication between devices. 31, 33, 40, 41, 42, 43, 44, 45,

46, 47, 78

RAM This stands for Random Access Memory, which is fast to access, short term memory, not saved on power off. 47, 48

Reactance This is due to a combination of **Inductance** and **Capacitance**, and makes up the non-reactive part of impedance, therefore **AC** specific. 37

Rectification This is the process of converting between **AC** and **DC**. 38, 79, 82

Rectifier A component which allows **AC** to **DC** conversion. 38, 79, 80, 82, 83

Resistance The degree to which a component opposes **Current** flow. 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 23, 28, 30, 31, 32, 36, 37, 48, 57, 62, 63, 73, 75, 131, 137, 138, 139, 164, 166, 167, 168

Resistor A component which opposes **Current** flow, causing loss as heat. 1, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 18, 20, 31, 32, 38, 59, 62, 63, 64, 67, 71, 73, 74, 75, 77, 83, 87, 131, 136, 137, 139, 140, 166, 168

RMS This is an acronym for Root Mean Squared. This is the effective value of a waveform, for example the **DC** magnitude equivlane energy wise. 12, 14, 24, 79, 80, 81, 82

RoHS This stands for Restriction of Hazardous Substances, and was initially adopted by the EU to put limits on the use of toxic materials such as lead in products, now widely adopted. 65, 71

ROM This stands for Read Only Memory, which is long term memory, which cannot be written to by the device itself, but is done externally on setup, so often contains firmware necessary for a device to function. 47

RS232 This stands for Recommended Standard 232, and is a serial communication protocol meant for communication between a computer and other devices. 41

RTD This stands for **Resistance** Temperature Detector and is a **Sensor** which has its **Resistance** vary significantly with temperature. Also referred to as a **Resistance** thermometer. 31

Schematic A representation of a circuit, commonly the first step to connect up a circuit before designing the final circuit board. 86

Schottky A type of **Diode** which has low **Voltage** drop and fast switching. 20

Semiconductor A substance with conductivity between a metal and an insulator. 14, 20, 25, 27, 28, 76, 77, 165, 168

Sensor A device which measures and responds to physical properties, outputting this as an electrical signal. 1, 3, 26, 30, 31, 32, 33, 35, 44, 46, 47, 48, 87, 137, 144, 146, 148, 152, 165, 167, 168

Serial Refers to data communication through a single data line. 40, 41, 42, 92, 96, 98, 101, 102, 104

Series It is a circuit configuration where **Current** passes through each component one after another. 1, 6, 8, 9, 10, 11, 12, 15, 16, 18, 19, 36, 52, 56, 57, 65, 67, 73, 79, 82, 84, 85, 88, 89, 90, 91, 92, 137, 154

SMD This stands for Surface-Mounted Device, and are components whos pins are soldered directly ontop of a PCB, so are generally unsuitable for breadboard or veroboard design. 61, 78

Soldering This is the process of using molten metal/ solder to bond parts together, for improved electrical, mechanical and thermal properties. 57, 59, 60, 61, 63, 64, 74

SPI This stands for Serial Peripheral Interface, and is a protocol for serial communication between two devices/ ICs. 31, 40, 41, 42

SuperCapacitor A form of **Capacitor** , with far higher energy density, not far off standard batteries, ideal for high power scenarios, due to their low **Resistance**. 35

Synchronous A type of communication where a specific clock rate is required, usually from the controller. 42

Thermistor A **Resistor** whose **Resistance** alters with temperature change, commonly used as a **Sensor**. 31, 32, 33

Thermocouple A **Sensor** for measuring temperature, based on the **Voltage** developed between two junctions, proportional to temperature difference. 31

Tolerance In electronics, Tolerance stands for the percentage at which a value can deviate from its rated value. 47, 62, 73, 75

Transformer A device which will increase or decrease the **Voltage** of an alternating **Current** source, having the reverse effect upon **Current**. 12, 23, 24, 25, 38

Transistor A **Semiconductor** meant for electronics control, signal generation or amplification. 29, 30, 32, 33, 43, 52, 57, 83, 87, 162, 163, 165

TRIAC A type of **Diode** which will conduct **Current** in either direction after triggered by a signal at its Gate. 20

UART A component for converting between **Parallel** and serial data, for better communication. UART stands for Universal Asynchronous Reciever-Transmitter. 40, 41, 43

Ultrasonic Sound waves above the human audible hearing range, typically above 20000Hz, often used in sensing. 32

USB This stands for Universal Serial Bus, with a collection of linked protocols and connectors. 35, 36, 40, 41, 43, 96, 98

Volt The unit for **Voltage**. 4

Voltage A Key electronics unit, otherwise known as Electromotive Force or Potential Difference, representing the energy provided to each Coulomb of Charge. 4, 5, 6, 7, 8, 9, 11, 12, 14, 15, 16, 18, 20, 23, 24, 26, 28, 31, 32, 33, 35, 36, 38, 43, 47, 48, 49, 50, 52, 57, 58, 62, 63, 70, 71, 73, 74, 75, 76, 77, 78, 79, 80, 81, 83, 91, 131, 132, 135, 136, 137, 138, 139, 144, 162, 163, 165, 166, 167, 168, 169

Watt The unit for power. 4, 34, 137

Wavelength It is the distance between successive peaks of a wave. 20, 25

Zener A type of **Diode** which allows reverse **Current** flow after a certain threshold **Voltage** is met. 20