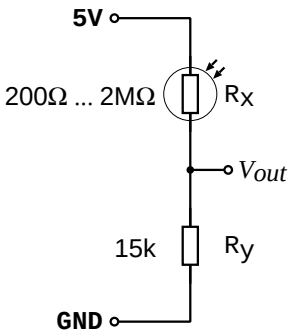
# Microcontrollers Systems and Interfacing

**week 6 experimental lab**

## 1 Measure the ambient light level

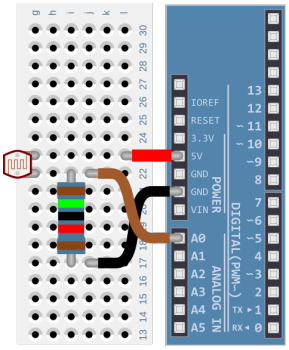



Your hardware kit includes a light-dependent resistor (LDR, or 'photocell').

► Display the ambient light level on the serial monitor or serial plotter.

Begin by creating a *potential divider* using the LDR and a fixed resistor. (A fixed resistor of 15 kΩ is a good choice.)
Read the analogue input voltage and use the serial monitor or serial plotter to display the values. Verify that changing the amount of light reaching the LDR causes a corresponding change to the analogue reading.

► Make your light readings be *self-calibrating*.



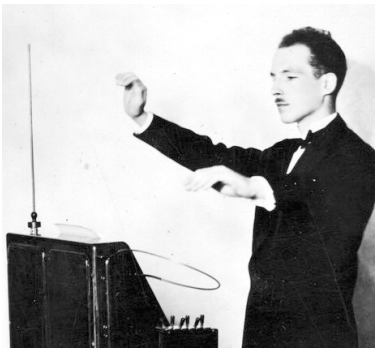
Create two `int`eger variables that will 'remember' the minimum and maximum values that have been returned from `analogRead()`. These values will be the lower and upper limits, respectively, of the 'input range' of a mapping using `map()`.

? What values should you use to initialise these two variables? Hint: the remembered minimum value must never increase. What initial value is *guaranteed* to be larger than any possible `analogRead()` value? Similarly for the remembered maximum value (except that it must never decrease).

Use these remembered values to specify the input range for the `map()` function to make your readings *self-calibrating*. Display the light level as a percentage of maximum, between `0` and `100`.

## 2 Challenges and mini-projects

### 2.1 Make a 'light theremin'



Use ambient light to control the frequency of an audio tone played on a loudspeaker. The dimmer the ambient light, the higher the frequency of the audio tone. This will create a musical instrument similar to a *theremin* that can be controlled (for example) by your hand moving over the light sensor.

To make your instrument more expressive, create a 'null zone' where the tone will be switched off to make the instrument become silent. There are at least three ways to set where the null zone begins: (1) by using a potentiometer to set a threshold value above which you use `noTone()` instead of `tone()`, or (2) by choosing a fixed threshold (maybe 95% of maximum brightness or higher) where the instrument becomes silent, or (3) by combining both approaches and making a potentiometer control the threshold of silence as a percentage of maximum light level. Try all three. From the perspective of a performing artist, which one works best?

If you make such an instrument, record yourself playing it and upload the recording to our Teams 'File' section.

### 2.2 Make a 'gentle' nightlight

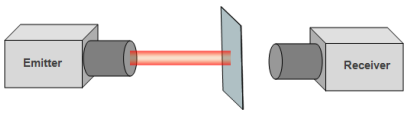
Use ambient light to control the brightness of an LED using pulse width modulation. As the light level falls, the LED should illuminate more brightly. A real-world application of this could be a lighting system that reacts gently to falling light levels, gradually coming to full brightness as night falls and then gradually coming back to off as daylight begins.

Improve your nightlight by making it react slowly to changes in ambient light level. If the room suddenly becomes dark, make your nightlight gradually change from fully off to fully on (and the opposite for a room that suddenly becomes bright).

Adapt your system to control the 'auto' setting of car headlights. Car headlights do not react immediately to changes in ambient light level. This prevents the headlights from turning on briefly when you drive under a bridge or through a short tunnel during the day, and prevents them from turning off briefly if you drive under a bright light at night. When you detect a change in light level (from dark to light, or light to dark) make your system wait 5 seconds before starting to react.

## 2.3 Make a 'light beam counter'

Use the ambient light level to count objects that move past the photocell. (A source of light pointing at the photocell, such as a torch or desk lamp, might help.) Every time an object moves past the sensor, increase the count by 1 and display it on the serial monitor.




### 2.3.1 Add frequency measurement to the light beam counter

A more sophisticated counter would also report how many objects are passing every minute. The function `millis()` returns the number of milliseconds that have elapsed since the program started to run. Maybe a more convenient measure would be the number of seconds, as a floating-point number:

```
float seconds() { return millis() / 1000.0; }
```

Use this function to calculate the amount of time that elapses between objects arriving. Display the average number of objects per minute (or per second) on the serial monitor every time a new one is counted.

There are two easy ways to calculate the average. The first uses the total elapsed time since the program started and the total number of objects seen so far. The second uses only the time that elapsed between the most recent two objects to calculate an 'instantaneous arrival rate'. Calculate both and print both. Perform experiments by simulating objects arriving with your hand. Which kind of average is more useful?

### 2.3.2 [DIFFICULT] Calculate a moving average for a fixed number of objects

Neither of the averages calculated in the previous challenge are perfect. After a while, the 'global average' stops reacting to local changes in arrival rate; the 'instantaneous average' reacts too quickly when the arrival rate increases suddenly and too slowly when objects suddenly stop arriving. (You can often see both behaviours in [badly designed] 'progress bars' when transferring large files over the Internet.)

A *moving average* is calculated from a subset of the full data set. Given a series of numbers and a fixed subset size, a moving average is calculated by using only the values in the subset. For example, the series might be 'elapsed time between objects being detected' and the fixed subset size might be 'only the most-recent ten objects detected'. This creates a kind of 'sliding window' onto the data, ten objects 'wide', through which measurements such as 'average' are made.

Modify your counter to print the average arrival rate, in objects per minute, for the last ten objects only.

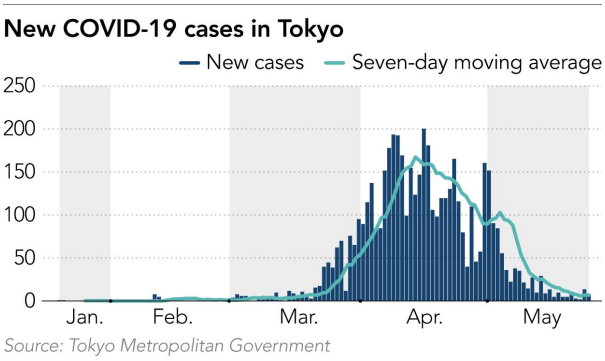
For example, if you have ten objects arriving per minute then after one minute the arrival rate will be printed as '10 per minute'. If you then increase that to twenty objects per minute, the arrival rate will increase slowly for 30 seconds until it finally reaches '20 per minute'. (From a digital signal processing point of view, this type of moving average is a kind of 'low-pass' filter.)

### 2.3.3 [VERY DIFFICULT] Calculate a moving average for a fixed time interval

Modify your counter to print the average arrival rate, in objects per minute, for the last minute only. Your sliding window is now 60 seconds 'wide'. You will have to figure out how to deal with an unpredictable number of objects being detected during that window. (It might be zero objects, it might be a hundred objects.)

To make your information truly useful, you should report the average arrivals per minute at least once every 10 seconds — even if no new objects have been detected during that time.

Of all the different averages you calculated and displayed, which is the most useful? Does it depend on the application? (What about transferring large files over the Internet? Which kind of average would you find most useful, personally?)



# Microcontrollers Systems and Interfacing

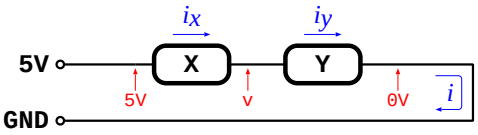
**week 6 reference material**

## A Series and parallel circuits

A *series* circuit is one in which all the components are arranged in a line, connected one after the other.

- *The **current** through **every component** in a **series** circuit is the **same**.*
- *The **voltage** across a **series** circuit is the **sum** of the voltages across the individual components.*

In the following circuit, X and Y represent two sub-circuits (containing one or more components) connected in series. Let's call their resitances $r_x$ and $r_y$.

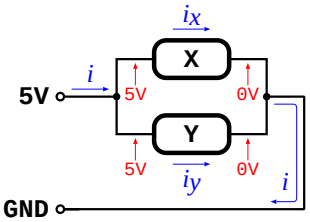



$$i_x = i_y = i = V/R = 5/(r_x + r_y)$$
$$v = IR = i \times r_y = 5 - (i \times r_x)$$

A *parallel* circuit is one in which all the components are connected between the same two points.

- *The **voltage** across **every component** in a **parallel** circuit is the **same**.*
- *The **current** flowing through a **parallel** circuit is the **sum** of the currents flowing in the individual components.*

In the following circuit, the sub-circuits X and Y are connected in parallel.




$$i_x = V/R = 5/r_x$$
$$i_y = V/R = 5/r_y$$
$$i = i_x + i_y$$

These two rules, called 'Kirchoff's laws', are always obeyed in any circuit. They are used often in circuit analysis, even in the simplest of circuits containing just two components.

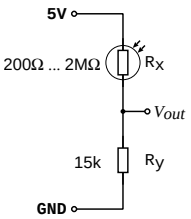
(If you are having trouble seeing intuitively why these laws *must* be true, use the water analogy of electrical current where water pressure is analogous to voltage and water flow rate is analogous to electrical current.)

## B Potential dividers

Potentiometers work by dividing a resistor into two halves. As the wiper moves from one end of the potentiometer to the other, the resistances of the two halves vary inversely to each other. At one extreme, the wiper is effectively connected to one end of the resistor. At one extreme, the wiper is effectively connected to the other end of the resistor. In the middle, the wiper appears to be connected to two resistors, each having half the total resitance of the potentiometer.

The effect is to vary the voltage at the wiper smoothly between the voltages at each end of the resistor, as the wiper is moved from one to the other. A potentiometer is an example of a *potential divider*; the potential (voltage) across the entire component is divided by the two resistances either side of the wiper position, to create an intermediate voltage at the wiper terminal.

In a series cirtcuit, the current flowing through all components is always the same. The total resistance of all components in series controls the overall current through the circuit. The individual resistance of each component, combined with the current flowing through the circuit, lets us calculate the voltage across that particular component.




We can create a potential divider using one fixed resistor and one variable resistor. An example of a variable resistor is a light dependent resitor (LDR) or 'photocell'. Assume the fixed resistor is 15 kΩ.

If the incident light on the LDR is low, the resistance is high; say 1 MΩ. The total resistance is 1.015 MΩ, and so with a 5 V supply the current through both resistors is 5 V÷1.015 MΩ=0.985 $\mu$A. The voltage across the fixed resistor is therefore .985 $\mu$A×15 kΩ≈1.5 mV.

If the incident light on the LDR is high, the resistance is low; say 200 Ω. The total resistance is 15.2 kΩ, and so with a 5 V supply the current through both resistors is 5 V÷15.2 kΩ=.329 mA. The voltage across the fixed resistor is therefore .329 mA×15 kΩ=4.93 V.