What is an Arduino?

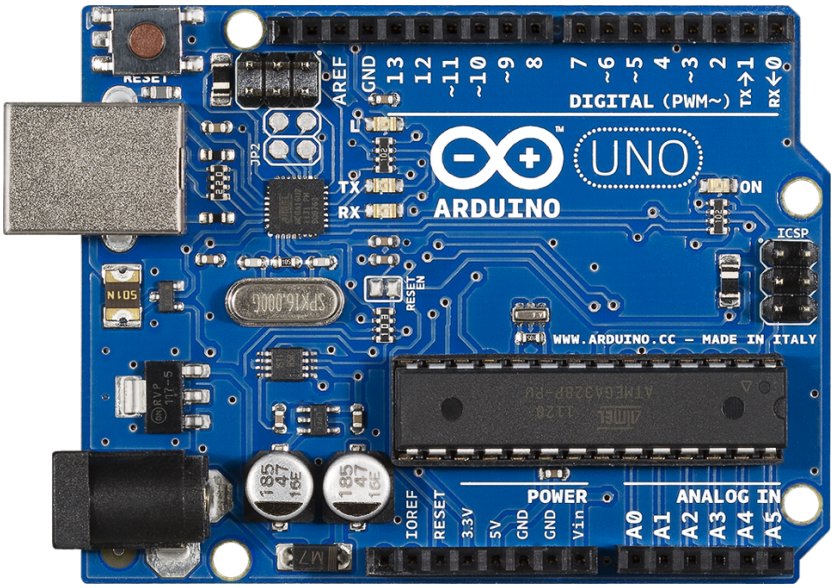
Arduino is an open-source hardware platform for developing embedded computer systems. Arduino targets physical computing – the hardware, the wires, the sensors and the actuators - and getting the job done. The aspects of Arduino that make it brilliant for beginners and developers are the speed at which results are achieved, the amount of support (on-line videos tutorials) on offer and the very low cost of getting started.

Arduino UNO is available from Amazon for around £18. The other bits-and-pieces breadboards wires and components are also readily available at very low cost, typically around £10.

Arduino code development is based on a dialect of C that very much simplifies programming. Extensive use is made of libraries to hide the code detail and allow beginners to achieve results quickly. Arduino is not about formal approaches to computer system development: it is about getting non-technical people engaged quickly and keeping them engaged through hands-on experimentation and having fun.

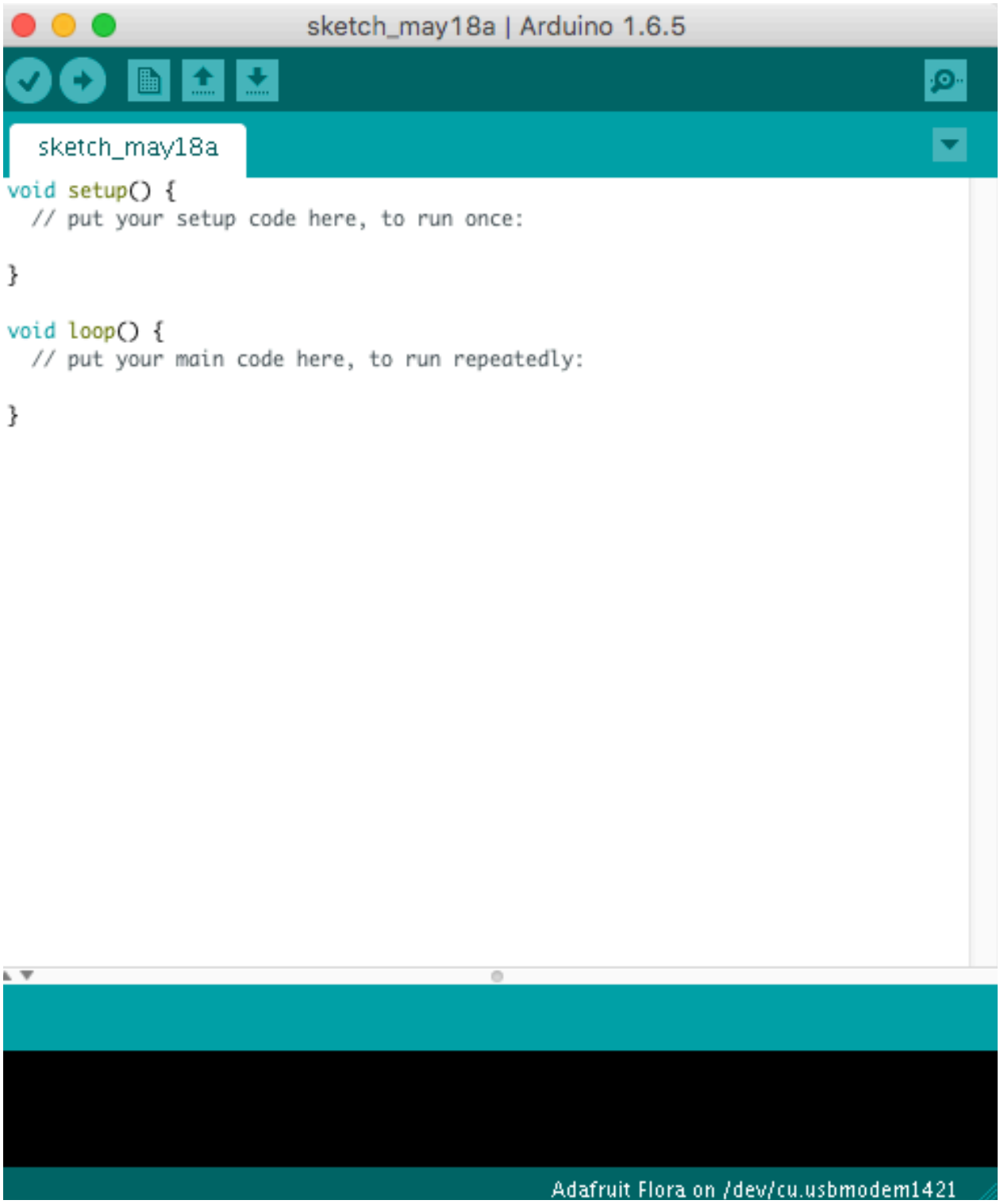
If you run into problems, there is a vast army of people developing applications with Arduino so on-line support and help is always available. The official Arduino website at is a good starting point : http://www.arduino.cc/ and many others are available on-line.

Arduino hardware comes in many forms with Arduino UNO being most suitable for beginners. The Arduino UNO, shown in below, comes ready to use and you should be up-and-running your first 'blink-the-LED' programme in a matter of minutes.



Let's get going, first plug your Arduino into the USB. We will then open the programming environment. Double click this icon:

This is what the Arduino programming environment looks like.



The next thing we're going to do is upload some software. This is called Blink. First find File in the Arduino menu.

Select

File>Examples>Basics>Blink.

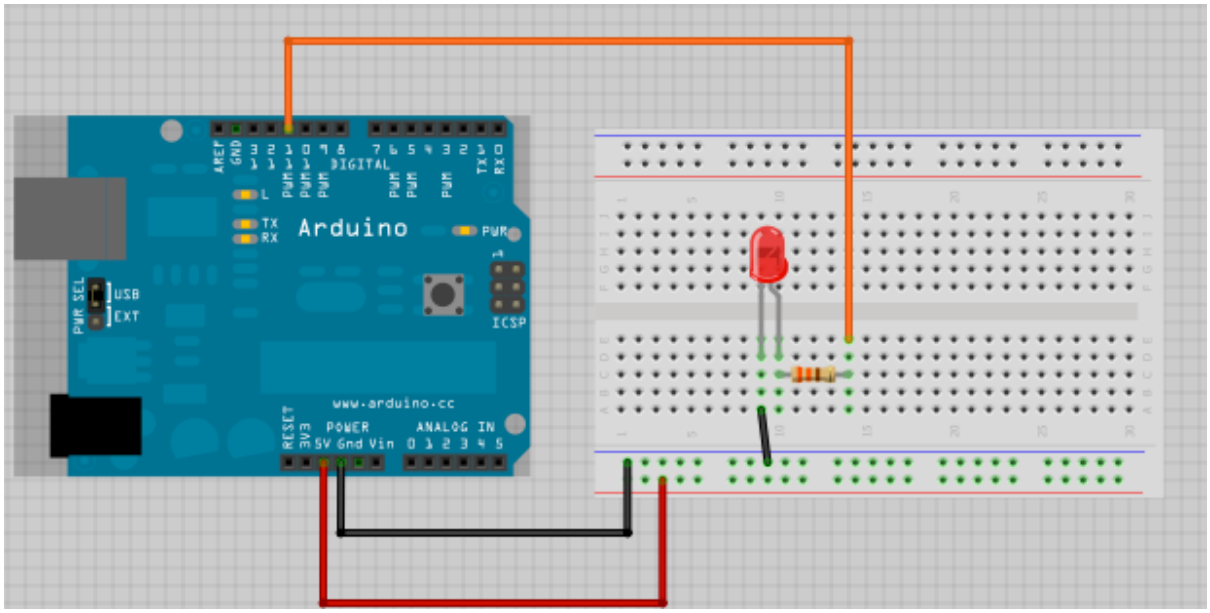
Your Turn

Modify the code to increase the speed that the LED flashes, say 2 flashes per second.
Modify the code to decrease the speed that the LED flashes, say one short flash every two seconds.

USE A BREADBOARD TO CONNECT AND FLASH A RED LED

This task introduces a the breadboard and a simple LED circuit. The picture below shows the UNO board connected to a RED LED on digital output pin11.

ARDUINO COMPONENTS & CONNECTIONS



When the digital output pin 11 is set 'HIGH', pin 11 actually outputs a voltage - 5 volts to be exact. This voltage drives a current through the RED LED and when that happens the LED turns ON. The resistor (220Ohms) is connected in series with the diode to limit the amount of current in the circuit to approximately 15mA. A 'HIGH' output turns the diode ON and a 'LOW output turns the diode 'OFF'.

You should also be aware that a diode only conducts current in one direction. The long leg is positive and connects to the resistor.

ARDUINO CODE

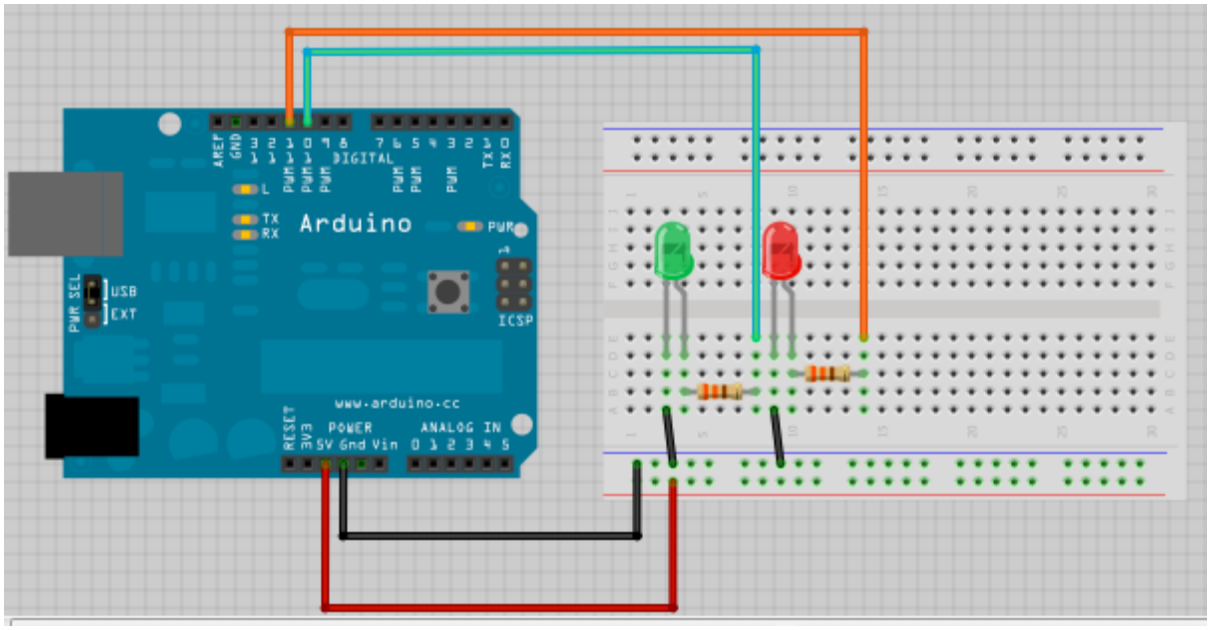
The code that drives the RED LED is shown below. When digital pin 11 is set 'HIGH' the RED LED turns ON. Conversely, when pin 11 is turned 'LOW' the RED LED turns OFF. The rate at which the LED flashes is controlled b the simple delay blocks. Simple really!

```
#define RedLED 11 // create a meaningful name for pin 11 - RedLED communicates much more than plain old 11.

void setup() {
pinMode(RedLED, OUTPUT);
} // end of setup
void loop() {
digitalWrite(RedLED, HIGH); // Turn the RED LED ON
delay(1000); // wait a second - 1000 milliseconds before switching the LED OFF
digitalWrite(RedLED, LOW); // Turn the RED LED OFF
delay(1000); // wait a second - 1000 milliseconds before switching the LED OFF
} // end of loop
```

TWO LEDS CIRCUIT

Add a second, GREEN LED to pin 10 using the connections shown.



YOUR TURN

T1.4. - Modify the code from Task 1B so that both LEDs blink at 0.5 second intervals

.

T1.5. - Modify the code so that the RED and GREEN LEDs flash alternately.

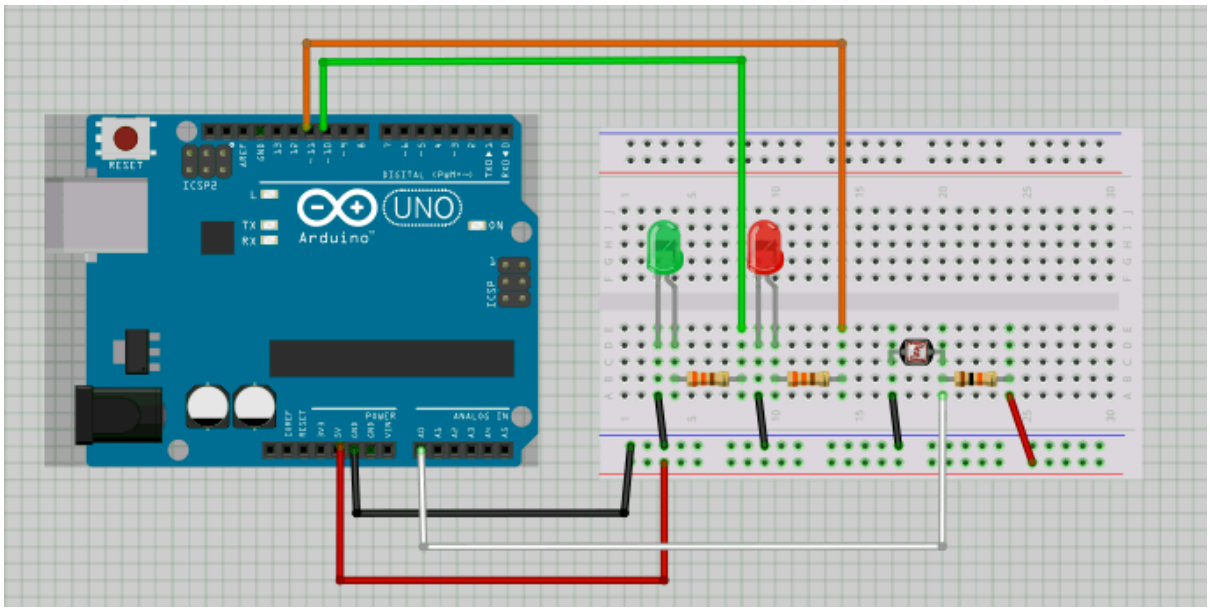
MANAGING YOUR CODE

If you have some that works save it, so that you can return to it when things go wrong **- and things will go wrong!** From the file menu use SAVE to save the current version and use SAVE AS to create a new version that you can experiment with. in the above example save your original code as BLINK1a, your RED LED version (Task 1B) as REDLED1 and save Task 1C as TWOLEDs1.

Another quick and useful way to experiment with code is to cut-and-paste existing lines and then comment out the lines that are not in use. New code can then be created by modifying the pasted code. The benefit of this approach is that the new and the old can be directly compared and it is relatively easy to switch back to the original version when things go wrong.

SENSOR INPUT - A SIMPLE PHOTOCELL

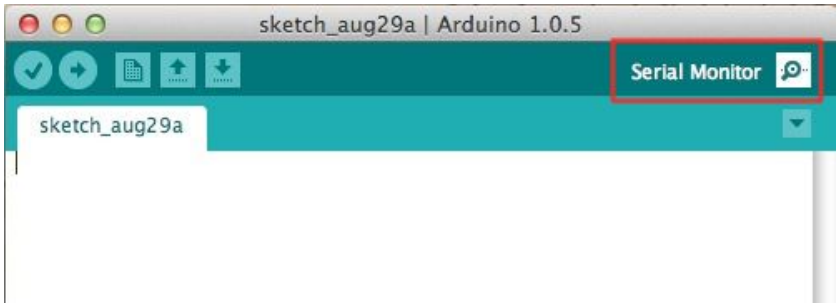
Now let's add a photocell so that we can detect the amount of light in the room. Add the Photocell circuit so that your breadboard looks like the figure below.



Breadboard wiring diagram for LEDs and Photocell sensor.

Let's take a closer look now and explain ANALOG IN. A photocell is just a resistor that changes its value depending on how much light works it detects. In the circuit the photocell is placed in series with another fixed resistor (10000 Ohms). The current flowing through the photocell and resistor creates a voltage at the white wire and the amount of voltage depends on the photocell's resistance which in turn depends upon the amount of light that the photocell 'sees'. The voltage at the ANALOG IN 0 pin is read by the microcontroller and made available as a number (0.. 1023).

```
int PhotoCellValue = 0; // create a place holder in RAM memory for the sensor value
void setup()
{
Serial.begin(57600);
} // end of setup
void loop()
{
PhotoCellValue = analogRead(0);
Serial.print("Photocell reading is >> " );
Serial.println(PhotoCellValue);
delay(1000); // pause for a second between readings
} // end of loop
```



Open the serial monitor and set the baud rate to 57600.

You should see numbers; these are the values from the photo cell. Note the values for the darkest and lightest inputs.

CODE ANALYSIS

// comments - can be placed anywhere in your code to help explain what your code is doing.
#define - is not code: it is just a mechanism for making code more readable. Pin 13 has very little meaning but ledPin or REDLed or AlarmLED gives us a better understanding of what this pin is used for in the program. When the code is compiled andwhere the name ledPIN is used replaced with the value 13.

; - semi colons - are used to mark the end of program statements. Statements are usually placed on separate lines so semi-colons are normally found on each line, The statements in the loop block is a typical example.

{} - 'curly brackets' - are used to mark blocks of code. Every programme statement within a block is executed in sequence. The two programme statements in this setup block are executed in sequence: the LED pin is set as an output so that the pin can drive the LED to turn it ON and OFF; the serial communications interface is configured to send sample values to the PC so that the we can read the sensed photocell values. The value - 57600 - specifies the speed at which the data is transferred.
void - newcomers to programming in 'C' are scared of 'void' but there is really nothing to be scared about. A void is an empty space so ignore it. It is included here because C sometimes requires it: as a beginner you can just ignore it.

DECISIONS - DECISIONS

T2.2 - Now combine the Photocell with the RED LED. Using the dark and light values recorded in T1 set a threshold value so that when the photocell detects a dark level the RED LED is ON and at the higher light levels the RED LED is OFF. You need some help with this because you need the code to make a decision.

IF-ELSE - A DECISION MAKING PROGRAM STATEMENT

It looks like this :

```
if (PhotoCellValue > 500)
{
digitalWrite(ledPin, HIGH); // Turn the alarm LED ON
Serial.println ("Light Level HIGH" );
}
else
{
digitalWrite(ledPin, LOW); // Turn the alarm LED OFF
Serial.println ("Photocell reading LOW" );
}
```

and it works like this: the value ***PhotoCellValue*** is compared with 300 and a TRUE or FALSE decision is made. If ***PhotoCellValue*** is greater than 300 the alarm LED is turned ON and the message " Light Level HIGH" is sent to the PC over the USB serial communication link. Conversely, if the photocell value is 300 or less the alarm LED is turned OFF and the message " Light Level LOW" is sent to the PC.

You have just let your program make a decision. The - if-else - program statement lets microcontroller control the LED output on the basis of the light level that it senses. You are now in the world of using sensors to control external devices. Welcome to the world of computer controlled devices!

MORE COMPLEX DECISIONS

T2.3. - Now combine the Photocell with the RED and GREEN LEDs. Use the dark and light values recorded in T1 set threshold values so that at dark the RED LED is ON and the GREEN LED is OFF, in mid-light the both LEDs are OFF and at higher light levels the GREEN LED is ON and the RED LED is OFF.

You need some more help here because a more complex decision is required. For the middle range we need a decision that includes two conditions greater than 300 (>300) and less than 600 (<600).

It looks like this :

```
if ( (PhotoCellValue >= 300) && (PhotoCellValue <= 600))
{
Serial.println ("Light Level OK " );
}
```

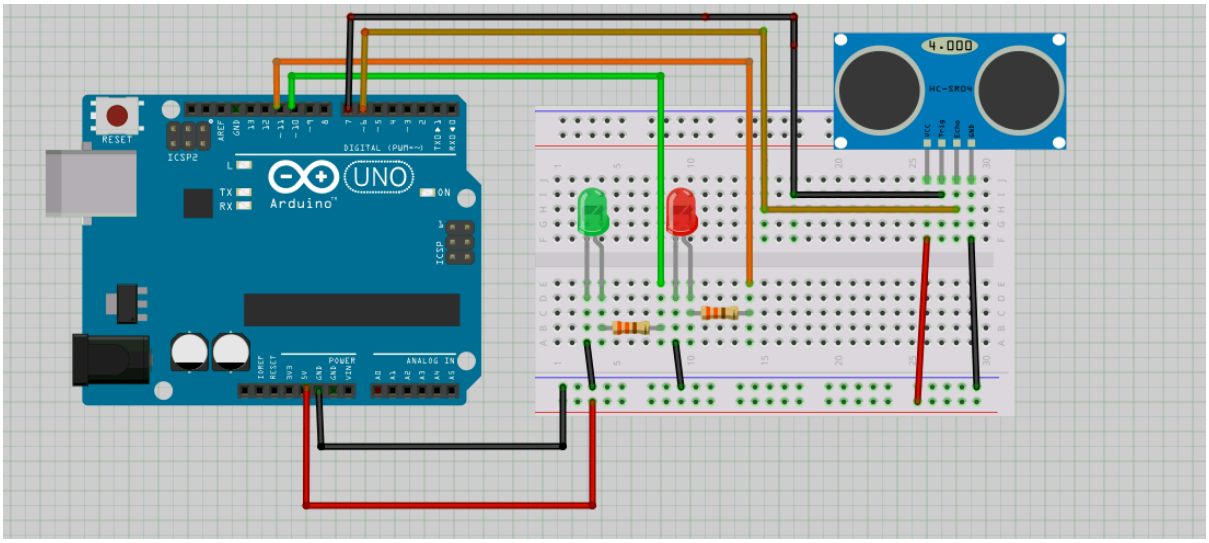
and it works like this: the value ***PhotoCellValue*** is compared with 300 and a TRUE or FALSE decision is made. The photocell value is then compared with 600 and a TRUE or FALSE decision is made. The two decisons are now combined into a single outcome : if both decisions are TRUE then the overall decision is TRUE (TRUE AND TRUE = TRUE). In code the && means AND. If the outcome is TRUE the message "Light Level is OK " is sent to the PC.

Notice all the brackets () - they are necessary to emphasise the order in which the decisions are made. First (**(PhotoCellValue > 300)** then (**PhotoCellValue < 600)** finally the two are combined using a logical AND - the && as in (() && ()) - if both of these decisions are TRUE the overall decision is TRUE. Looks complex, perhaps sounds complex but its fairly straightforward really.

T2.4.- Now make a panic meter as the light level changes from light to dark the RED LED flashes faster and faster.

SENSOR INPUT - A RANGE SENSOR

Range sensors are most often used to warn things that they are getting to close to other things. Robots use range sensors to stop them bumping into walls and doors and people detectors prevent people from bumping into glass doors. In this picture the photocell has been replaced with a range sensor.



An ultrasonic range sensor works like a simple radar by transmitting a series of short ultrasonic pulses and listening for the returned echo pulse as it is reflected from an object and back to the sensor. The distance to the object is measured by the time difference between the original pulse and the returned echo -simple really!

UAnother form of range sensor uses pulses of infra-red light to achieve the same result.

```
int SR04_Echo = 6; // Echo returned from range detection
int SR04_Trigger = 7; // Trigger a pulse-echo range detection
long distance;
long cm;
void setup(){
pinMode(SR04_Echo, INPUT);
pinMode(SR04_Trigger, OUTPUT);
Serial.begin(57600); // Set the USB serial port to 57600 bits per second
} // end of setup
void loop()
{
digitalWrite(SR04_Trigger, LOW); // Send a trigger pulse
delayMicroseconds(2);
digitalWrite(SR04_Trigger, HIGH);
delayMicroseconds(10);
digitalWrite(SR04_Trigger, LOW);
distance = pulseIn(SR04_Echo, HIGH); // the distance is proportional to the time interval of the
ECHO pulse
cm= distance/58; // Convert time to distance - 58 from data sheet
Serial.print("Distance to object (cm) >> ");
Serial.println(cm);
} // end of loop
```

CODE ANALYSIS

This is a good example of a how Arduino simplifies code development. The ***pulseIn()*** function returns the time interval of a digital pulse and it is built-in to the Arduino library. There is no need to get involved in the low-level coding detail of how this achieved -simply use it! The delayMicroseconds(10); is another example. This function provides a much shorter delay period, this time in millionths of a second.

YOUR TURN

T3.1. - Now repeat the photocell task T2.3 but modify it so that it detects three distances TOO FAR, TOO CLOSE and OK. How could this interface be applied in the a real situation ?

EXTENDING YOUR SENSED WORLD

It is not difficult to see how these simple interfaces could be extend to more complex application. Here are some possibilites to set you thinking:

CONTROLLING THINGS

By adding a small motor we could control it's speed using distance or light

Adding a temperature sensor would allow us to control the speed of a fan.

By adding a string of coloured LEDs we could change the colour or brightness depending on the light in the room.

SIMPLE INTERACTIVE GAMES

By installing a programme called Scratch4Arduino on your PC you can very easily build interactive games using sensors. For example you could bounce a ball based on the data from a photocell. A Range Sensor would work equally well.

To raise funds at our local school we connect a wire coat hanger across a battery. The wire is bent into a maze and the game is to trace the path along the maze using a wire hoop. If the hoop touches the maze a buzzer sounded and a light flashed - GAME OVER! At 10p for two attempts and 20p for five you can quickly add to the school funds.

GETTING MOBILE

Can you control a motor or a strip of coloured LEDs from a smart phone. The answer is 'YES' and its easier than you think. By adding a Bluetooth modem (costing around £7.00 from Amazon) and downloading a free app from the APP store you can get up and running easily and quickly.

SUMMARY

We have covered a lot of ground in a short time but we have only scratched the surface. Hopefully we have done enough to get you started and to show you that learning about computers and electronics is both interesting and fun. It is also not 'rocket science': although several people have built rockets that are guided by an Arduino microcontroller. Robotics, flying drones - quadcopters etc and home automation systems are common areas for Arduino projects and there is a vast range of project ideas available. Go on-line browse 'Arduino projects and tutorials', then spend a while reading through some of the 2 million hits on Google.

You should also be aware that we are currently taking our first steps towards the 'Internet of Things' (IoT) and Arduino type devices will play their part. There will be many interesting and rewarding careers in developing intelligent sensor based applications.