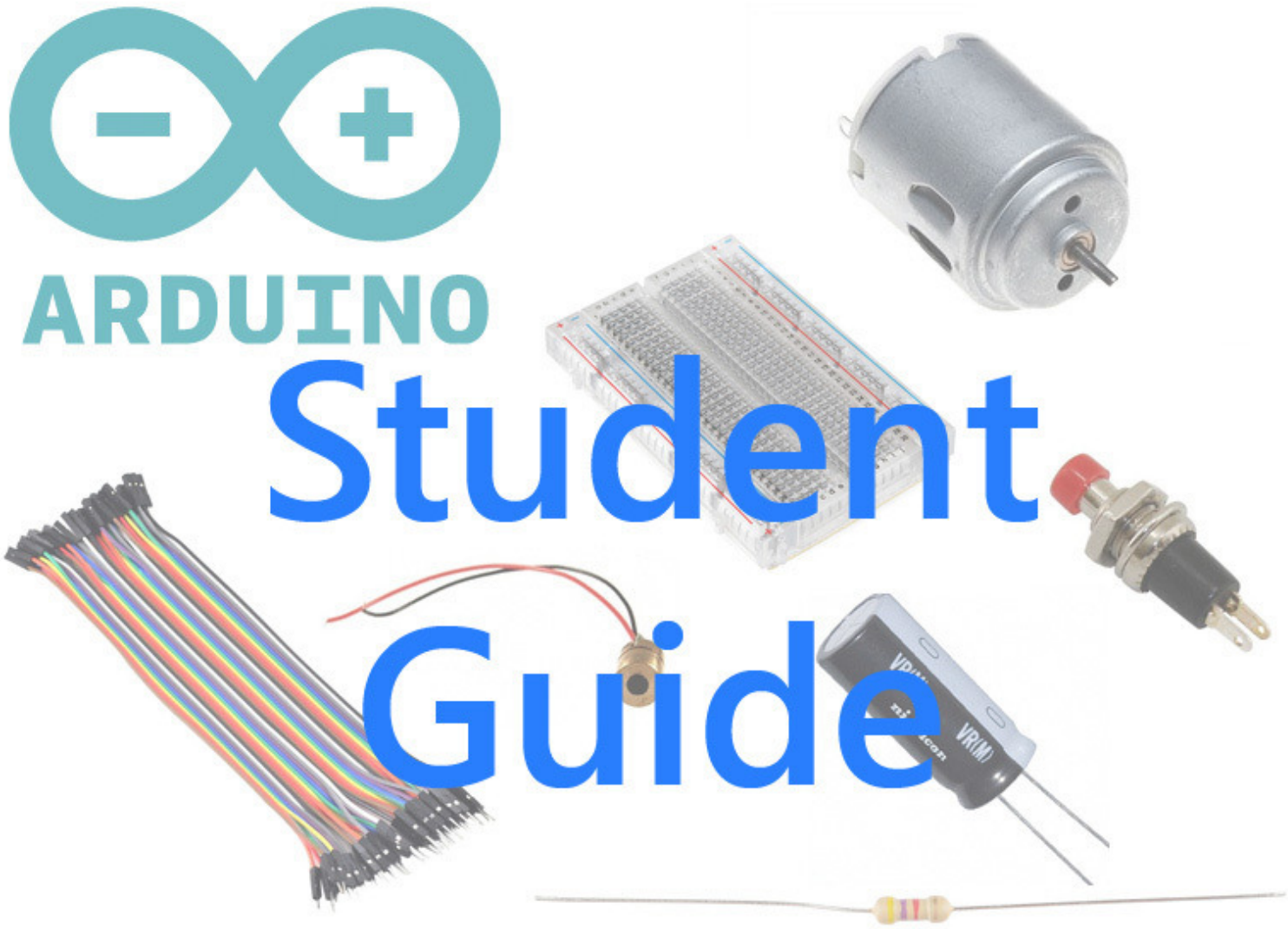
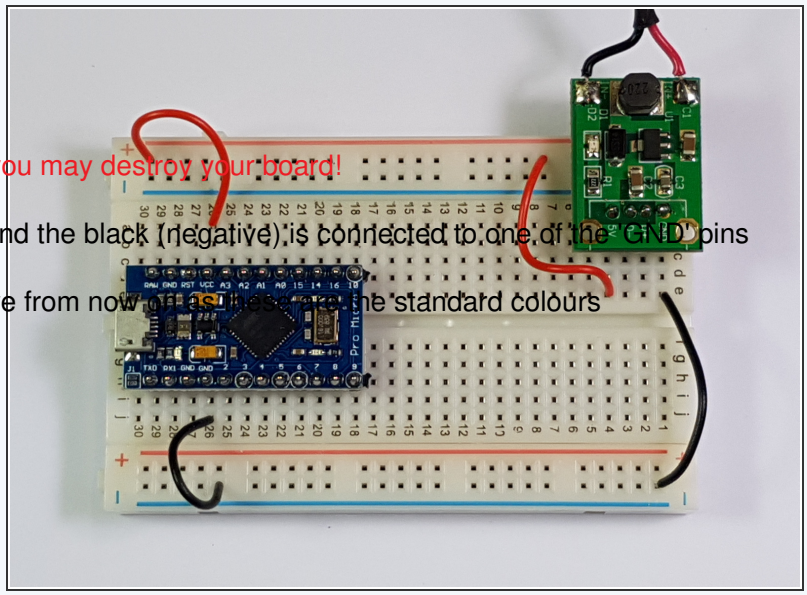
Inventor School Session 1 - Buzzers, LEDs and pushbuttons



Step 1

Step 1 — Connecting the microcontroller



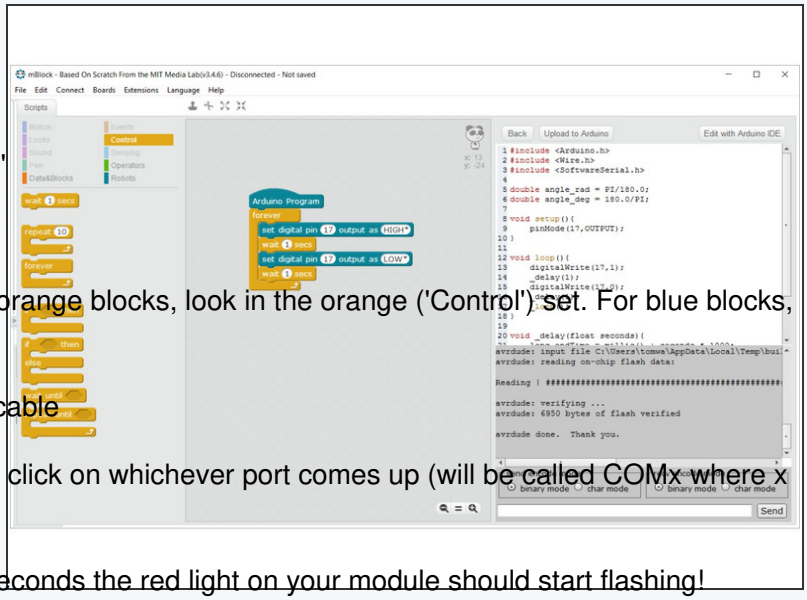
- Wire up the microcontroller as shown

⚠ Make sure you get the wiring **exactly** right otherwise you may destroy your board!

ⓘ N.B. The red (positive) is connected to the 'VCC' pin and the black (negative) is connected to one of the 'GND' pins

- You'll always use red for positive and black for negative from now on as it is best to use the standard colours

Step 2 — Your first programme (M)

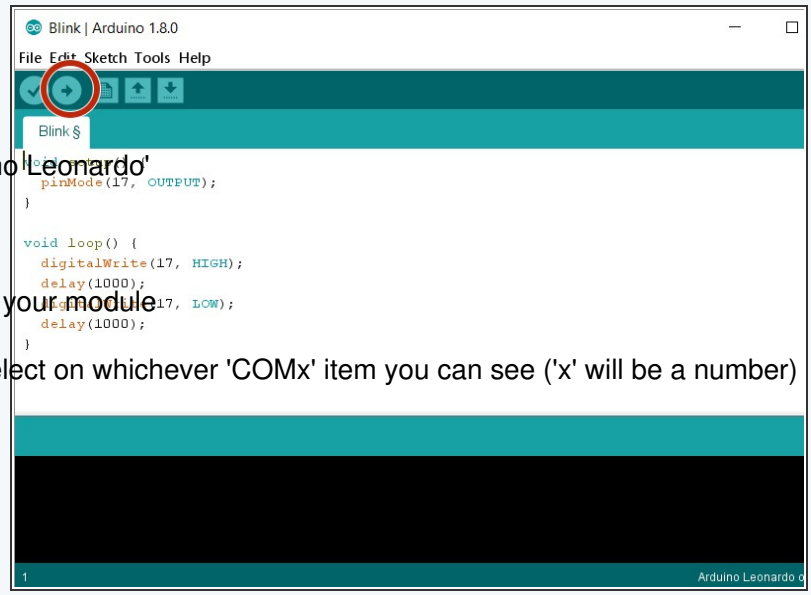


- Start the mBlock software
- Go to the 'Boards' menu and select 'Arduino Leonardo'
- Go to the 'Edit' menu and select 'Arduino Mode'
- Create the programme shown on the screen (hint: for orange blocks, look in the orange ('Control') set. For blue blocks, look in the blue ('Robots') set
- Connect the module to your computer using the USB cable
- Go to the 'Connect' menu, select 'Serial Port' and then click on whichever port comes up (will be called COMx where x is a digit)
- Press the 'Upload to Arduino' button - after about 10 seconds the red light on your module should start flashing!

Step 3

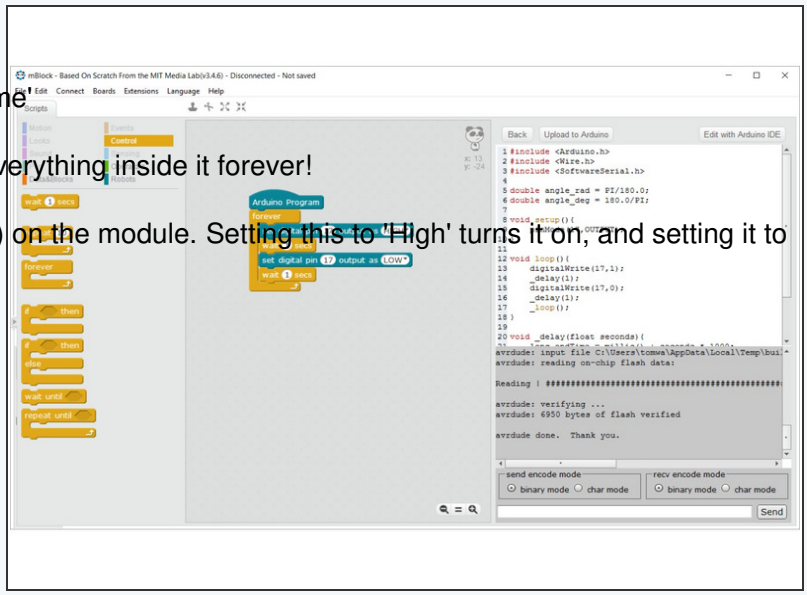
Step 3 — Your first programme (A)

- Start the Arduino software
- From the 'Tools' menu, select 'Board' and then 'Arduino Leonardo'
- Type the programme **exactly** as shown on the screen
- Connect your USB cable between your computer and your module
- Go to the 'Tools' menu, select the 'Port' option, and select on whichever 'COMx' item you can see ('x' will be a number)
- Press the 'upload' button (looks like a right arrow)



Step 4 — How it works (M)

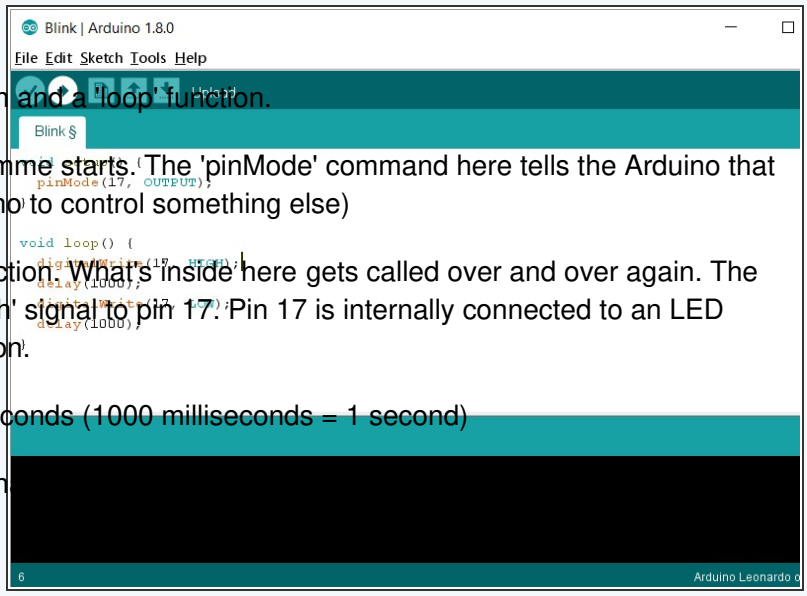
- Every programme must start with a 'Arduino Programme'
- The 'forever' block does just what it says - it repeats everything inside it forever!
- Digital pin '17' is connected internally to the LED (light) on the module. Setting this to 'High' turns it on, and setting it to 'Low' turns it off



Step 5

Step 5

— How it works (A)



- Every Arduino programme must have a 'setup' function and a 'loop' function.
- The 'setup' function gets called once when the programme starts. The 'pinMode' command here tells the Arduino that pin 17 is an output (a signal is sent out from the Arduino to control something else)
- Every Arduino programme must also have a 'loop' function. What's inside here gets called over and over again. The 'digitalWrite' command tells the Arduino to send a 'High' signal to pin 17. Pin 17 is internally connected to an LED (light) on the module. 'High' means to turn something on.
- The 'delay' command waits a certain number of milliseconds (1000 milliseconds = 1 second)
- The command after the first 'delay' outputs a 'Low' sign

Step 6

— Look Ma - no wires!

- Now try removing the USB cable
- If your battery pack isn't turned on, turn it on now
- Your LED should flash just like before
- Your programme is running inside the little Arduino module all by itself - it doesn't need to be connected to a computer to work
- This is really useful as you can make things like gadgets and robots that can run all by themselves once you've written your programme!

Step 7

Step 7 — Faster flashing lights ...

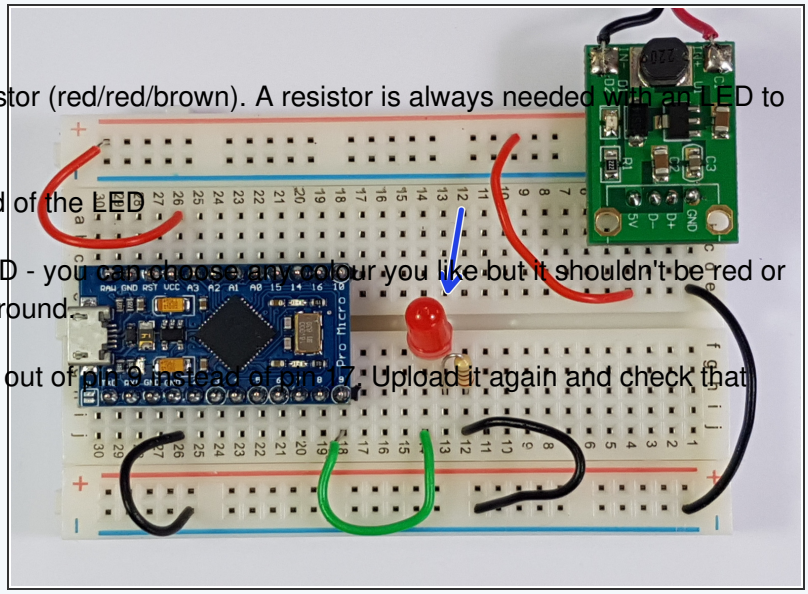
- Now change your programme to make your LED flash twice as quickly!
- Remember to show your tutor and get your challenge stamped off as you do it!

Step 8 — Adding an LED

- Wire up the circuit with a red LED and a 220 ohm resistor (red/red/brown). A resistor is always needed with an LED to limit the current - without it, the LED will burn out.

ⓘ Note the blue arrow shows the flat side (negative) lead of the LED

- We have a green wire here connecting pin 9 to the LED - you can choose any colour you like but it shouldn't be red or black as they are reserved for the power supply and ground.
- Now change your programme so it is sending a signal out of pin 9 instead of pin 17. Upload it again and check that your LED flashes.



Step 9

Step 9

— Flasher with a limp

...

- Now change your flashing light so that it has a 'limp'!
- Do this by making the 'on' time longer than the off time - you can choose your times, as long as it is obvious the flashing light is now 'uneven'
- Remember to show your tutor every time you complete one of these challenges!

Step 10

— SOS flasher

- Now make your flashing light into a beacon - it should signal 'SOS' in Morse Code over and over again!

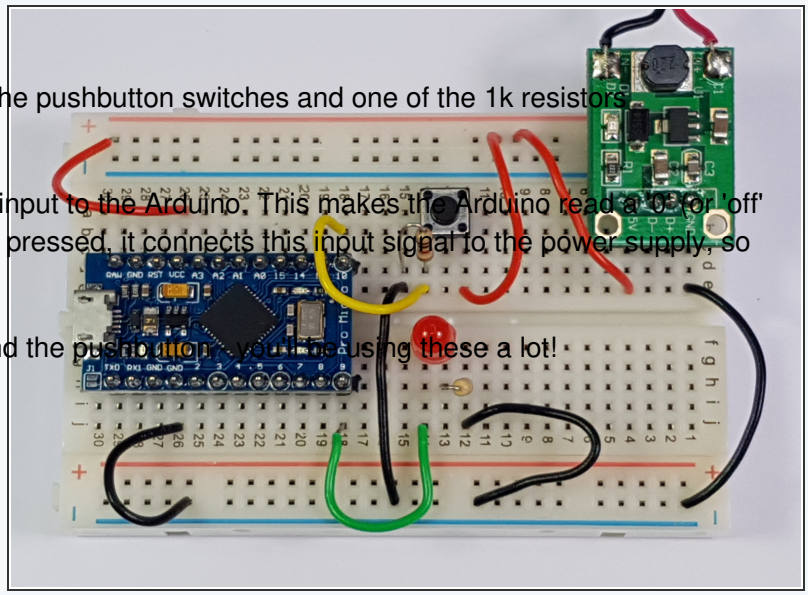
Step 11

Step 11 — Switches ... the hardware

- Now wire up the circuit as shown - you'll need one of the pushbutton switches and one of the 1k resistors (brown/black/red)

ⓘ The 1k resistor is connected between ground and the input to the Arduino. This makes the Arduino read a '0' (or 'off' signal) if the button is not pressed. When the button is pressed, it connects this input signal to the power supply, so will instead make the Arduino read an 'on' signal.

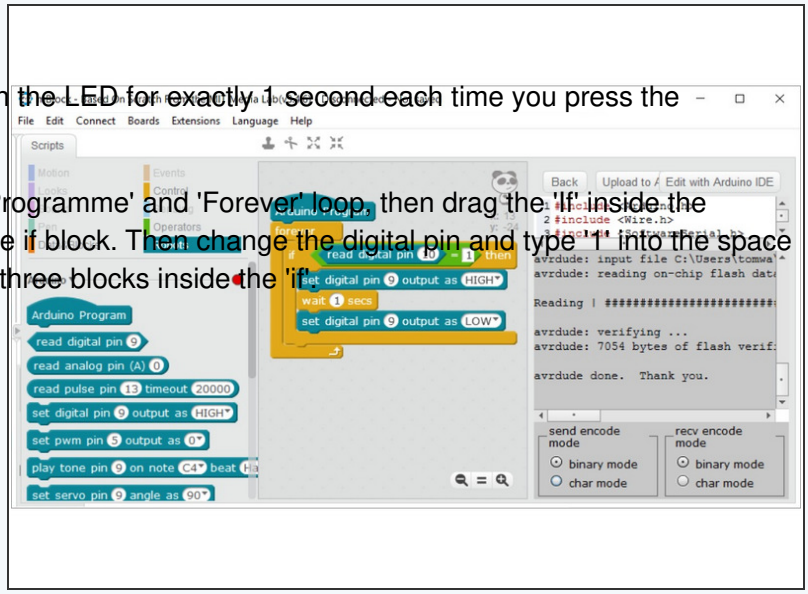
- Try to remember how you have wired both the LED and the pushbutton - you'll be using these a lot!



Step 12 — Switches ... the software (M)

- Now create the software and try it out! It should turn on the LED for exactly 1 second each time you press the pushbutton.

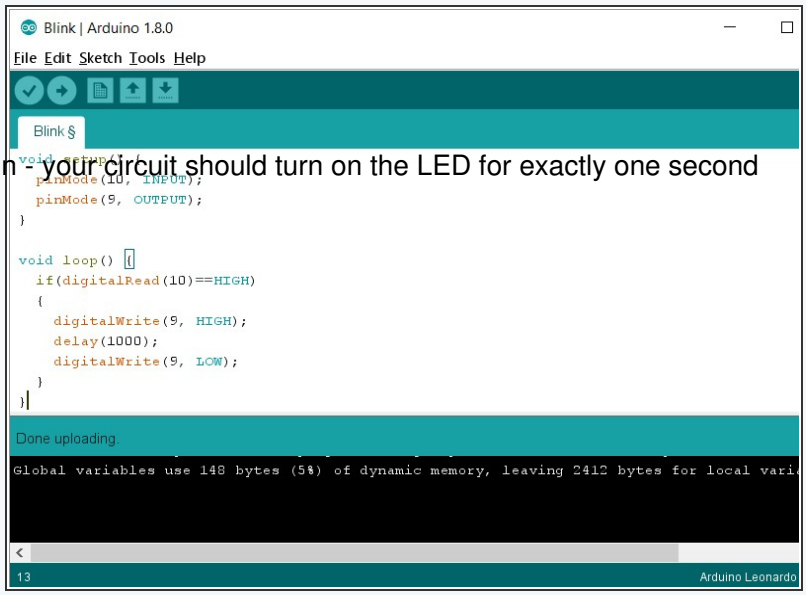
ⓘ Hint - drag the blocks in this order - first the 'Arduino Programme' and 'Forever' loop, then drag the 'If' inside the forever. Now drag a green '=' block inside the top of the if block. Then change the digital pin and type '1' into the space on the right of the equal sign. Finally drag in the other three blocks inside the 'if'.



Step 13

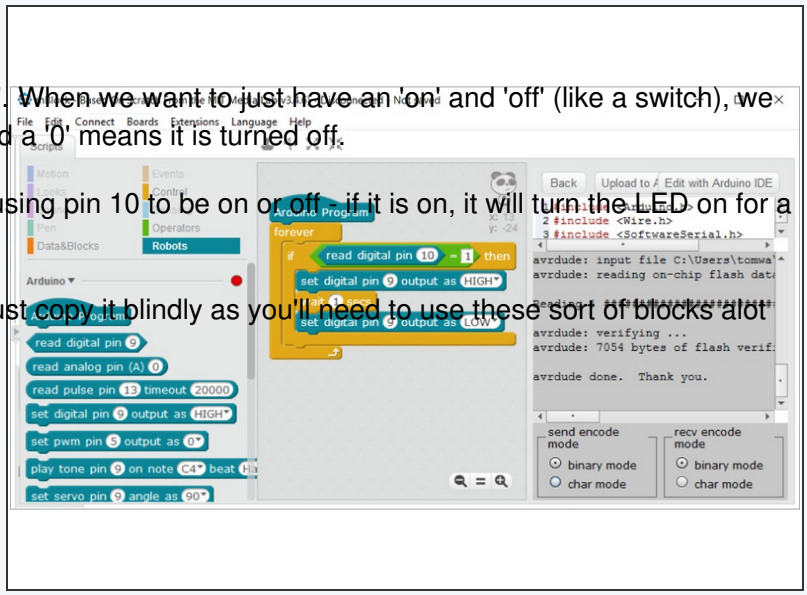
Step 13 — Switches ... the software (A)

- Now create the programme as shown.
- When you're ready to go, press the upload button again - your circuit should turn on the LED for exactly one second each time you press the button.



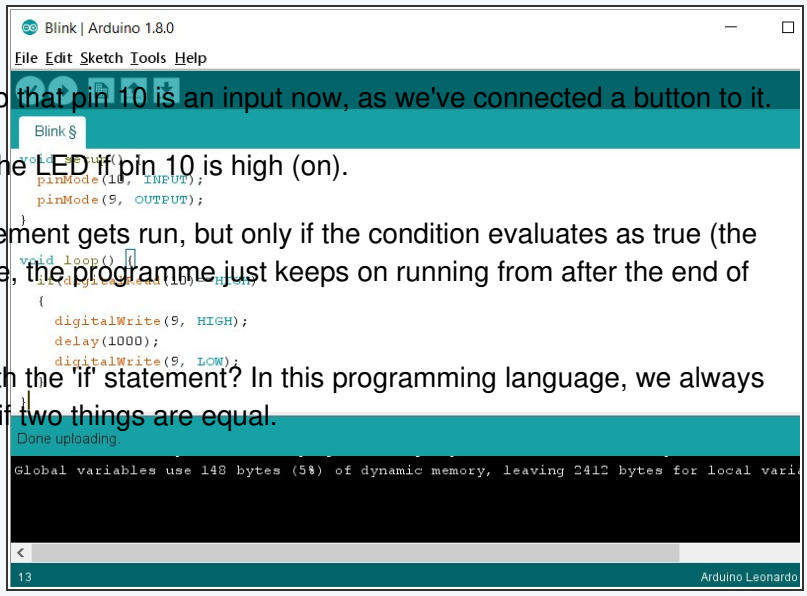
Step 14 — How it works (M)

- Some pins can be used as both 'digital' and 'analogue'. When we want to just have an 'on' and 'off' (like a switch), we use a digital one. A digital '1' means it is turned on, and a '0' means it is turned off.
- So the if statement just checks if the pushbutton is causing pin 10 to be on or off - if it is on, it will turn the LED on for a second, and then turn it off.
- Try to understand how this programme works - don't just copy it blindly as you'll need to use these sort of blocks alot in your projects!



Step 15

Step 15 — How it works (A)



- You'll see in the setup() function we've told the Arduino that pin 10 is an input now, as we've connected a button to it.
- We've now also used an 'if' statement to only turn on the LED if pin 10 is high (on).
- Anything within the two curly brackets after the 'if' statement gets run, but only if the condition evaluates as true (the button is pressed in this case). If the condition isn't true, the programme just keeps on running from after the end of these two curly brackets.
- Can you see that there is a double equal sign used with the 'if' statement? In this programming language, we always need to use a double-equal whenever we are **testing** if two things are equal.

Step 16 — Two-button Morse Code sender

- Can you add another button to your circuit? You'll need to choose another pin to connect it to, and don't forget you'll need another 1k resistor. Try to copy the wiring of the other switch, except just connect it to another pin.

⚠ Make sure you don't connect two components that aren't meant to be connected by mistake - remember that any wires (or component leads) inserted into the same **column** will be joined. To fit new components in you might sometimes need to remove previous ones and put them in somewhere else.

Step 17

Step 17

— Egg Timer

- Can you make an egg timer? It should start timing when you press the first button. After the set time (you can decide how long), the LED should turn on. When you press the second button, the timer should reset and the LED should go out.

Hint: an egg timer would normally time perhaps 3 or 4 minutes. You might be waiting a long time to test yours, so feel to make your times much shorter - perhaps just a few seconds!