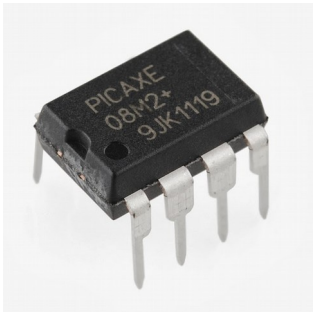
Working with Micro Controllers

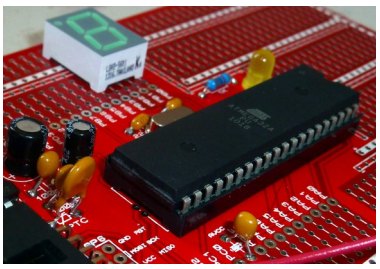
PICAXE

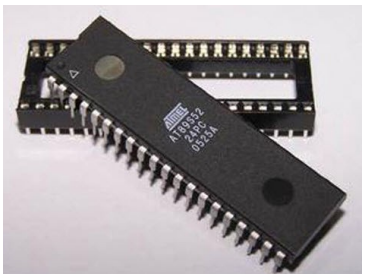


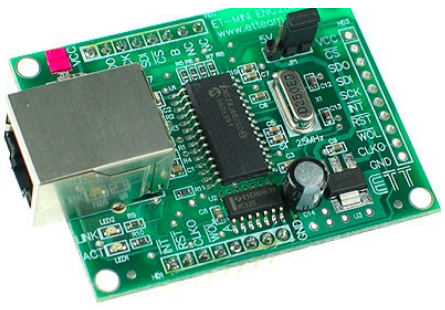
What is a micro controller?

- Also called a ‘computer-on-a-chip’.
- It is a low-cost integrated circuit (IC) that contains memory, processing units, and input/output circuitry in a single unit.

Micro controllers are purchased ‘blank’ and then programmed with a specific control program.







Once programmed, the microcontroller is built into a product to make the product more intelligent and easier to use.

Example:

A microwave

- It uses a single microcontroller to process information from the keypad, display user information on the seven segment display, and control the output devices (turntable motor, light, bell and magnetron).

One microcontroller can often replace a number of separate parts, or even a complete electronic circuit.

advantages of using microcontrollers in a product design are:

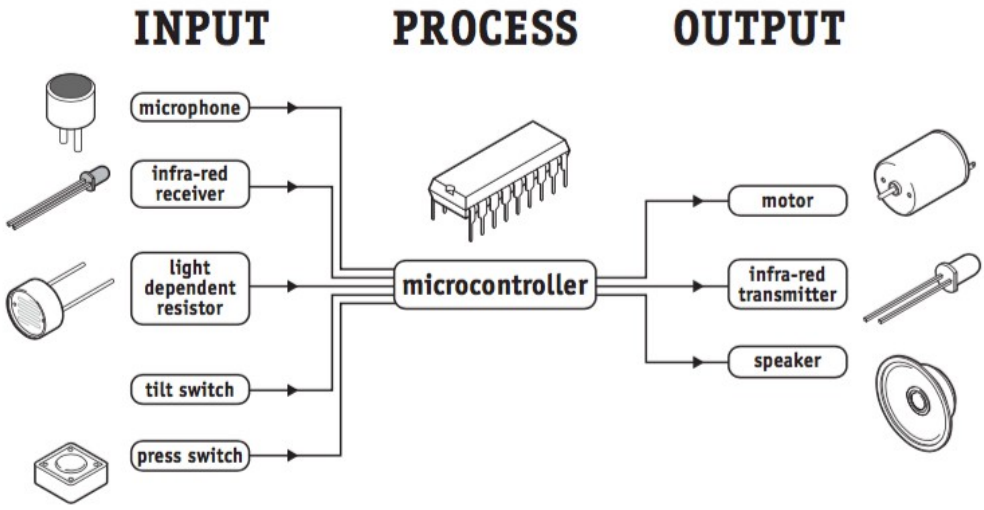
- To increase reliability through a smaller part.
- To reduce stock levels, as one micro controller replaces several parts.
- To simplify product assembly and smaller end products.
- Greater product flexibility and adaptability since features are programmed into the micro controller and not built into the electronic hardware
- rapid product changes or development by changing the program and not the electronic hardware.

Applications of microcontrollers:

- household appliances, alarm systems, medical equipment, vehicle subsystems, and electronic instrumentation.

- Some modern cars contain over thirty microcontrollers - used in a range of subsystems from engine management to remote locking!

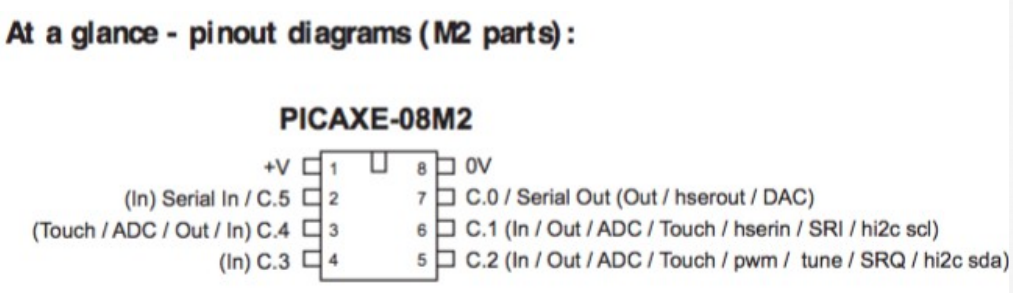
Micro controllers input and output

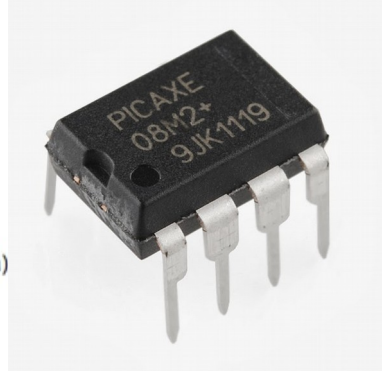


What is the PICAXE system?

The PICAXE system has the similar characteristics of the new generation of low-cost 'FLASH' memory based microcontrollers. These microcontrollers can be programmed over and over again (typically 100 000 times) without the need for an expensive programmer.

PICAXE layout:

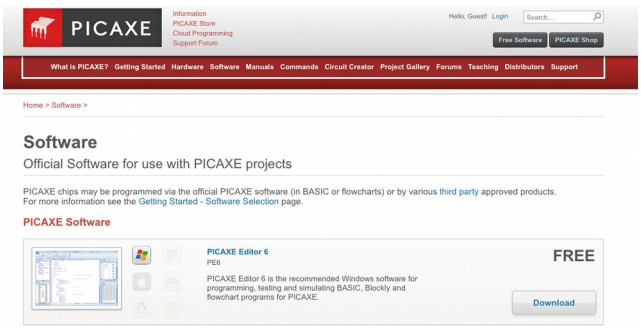




- More about PICAXE visit: http://www.picaxe.com/
- PICAXE manuals and help visit: http://www.picaxe.com/Getting-Started/PICAXE-Manuals/

Software download:

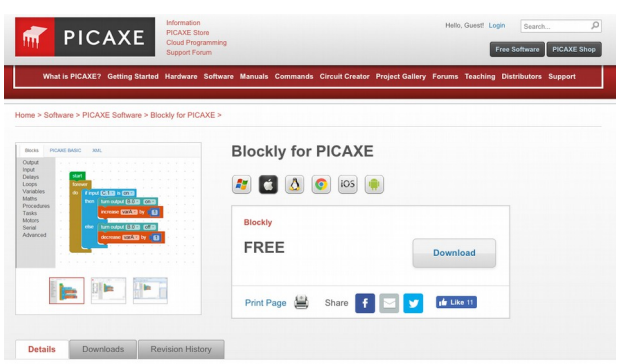
- For **Windows, Mac and Linux** operating systems visit: http://www.picaxe.com/Software



- For **Chromebooks and Tablet** users:

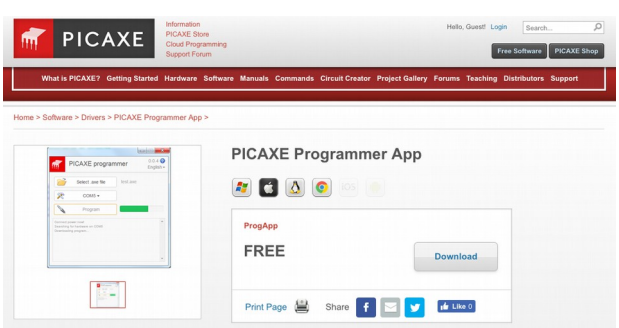
Install the following programs on your Google account :-

1. **PICAXE Blockly** (this allows you to write code for the PICAXE) https://chrome.google.com/webstore/detail/picaxe-blockly/hhdlapnjifkkcpghcapopejopnbpapnb



2. **PICAXE programmer** (this allow you to program your PICAXE)

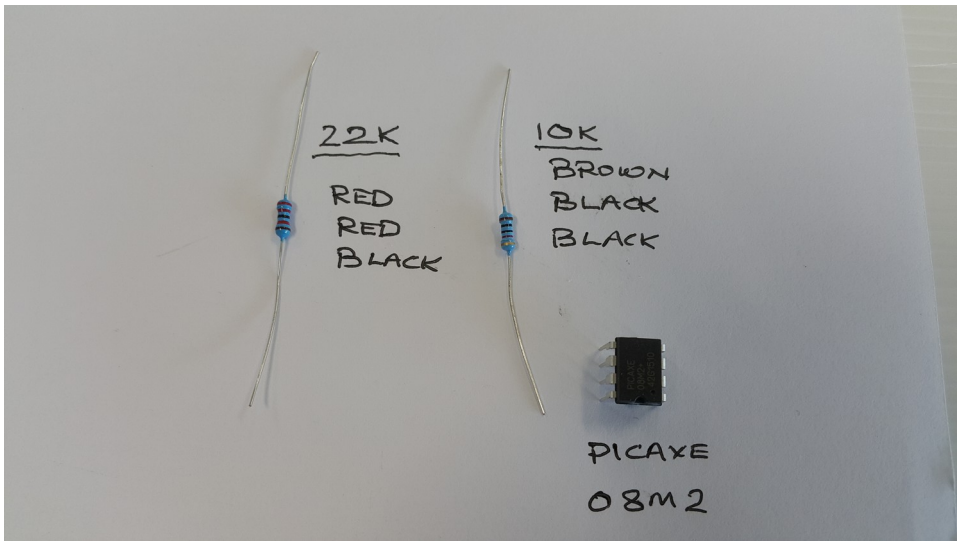
https://chrome.google.com/webstore/detail/picaxe-programmer/mcakegabnfookegpcpdmhgpdajgkjncp?hl=en-GB



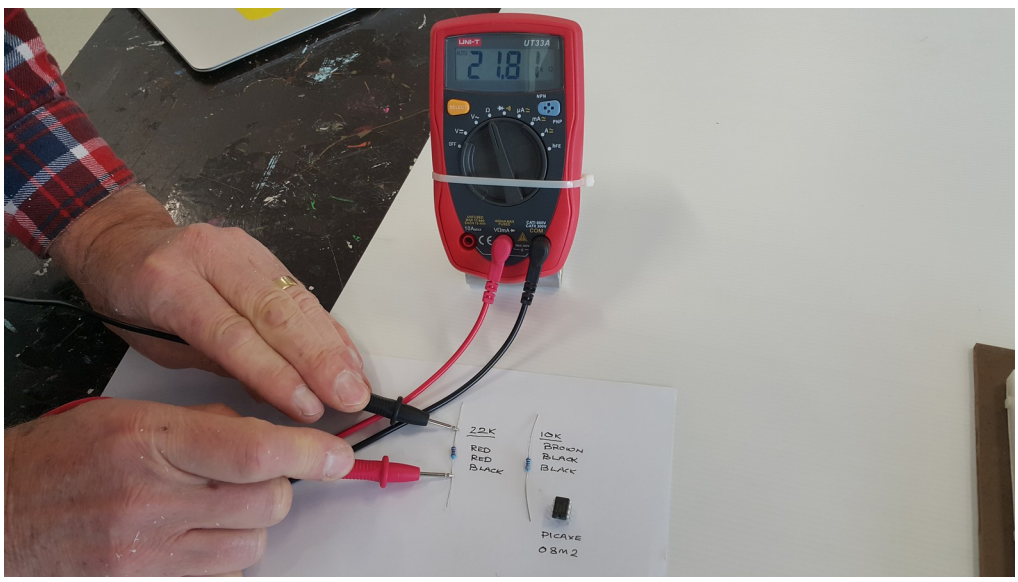
To code on Chromebook click on the following link: https://drive.google.com/a/mhjc.school.nz/file/d/0B9hRsiioOJuaM2hzVlFMLVZqZ1E/view?usp=sharing

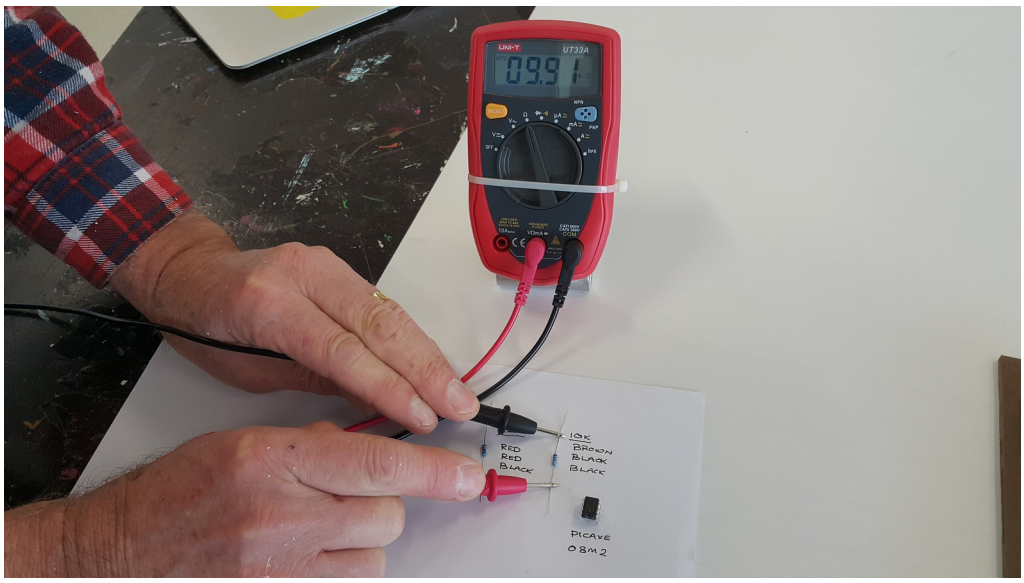
Lets get started with breadboard and components preparations:

1- Electronic components you need:

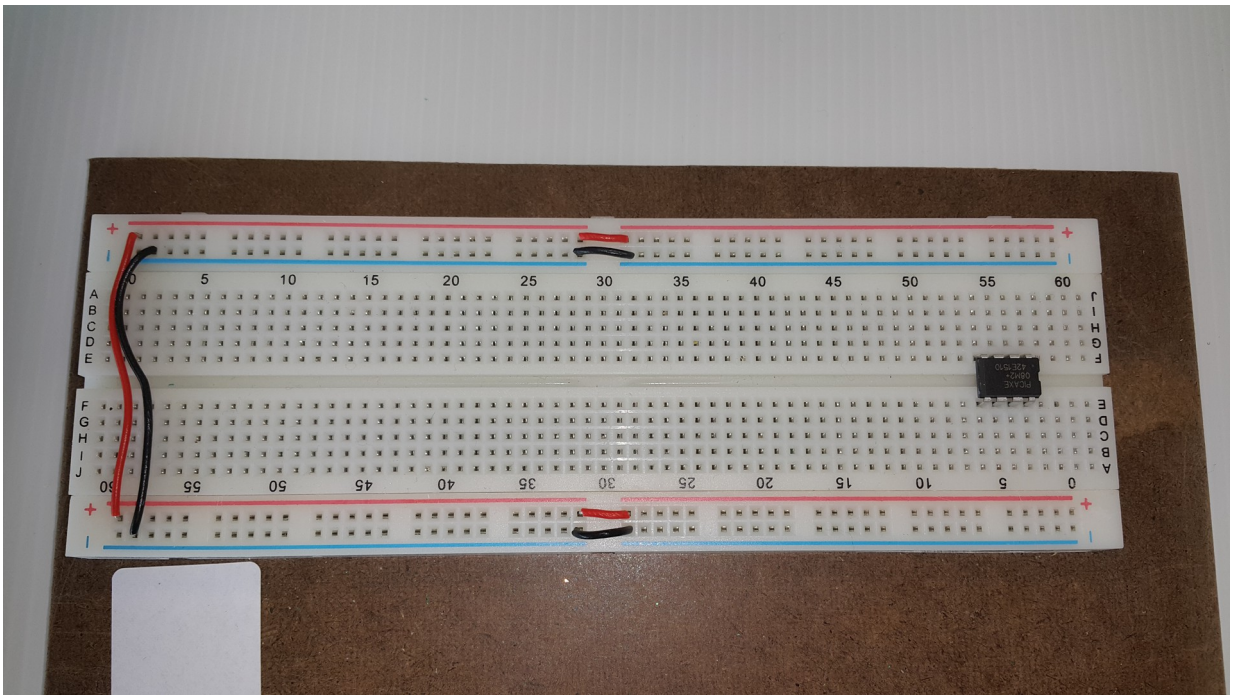


2- Test the resistance of the 2 Resistors using the multi-meter on Ω

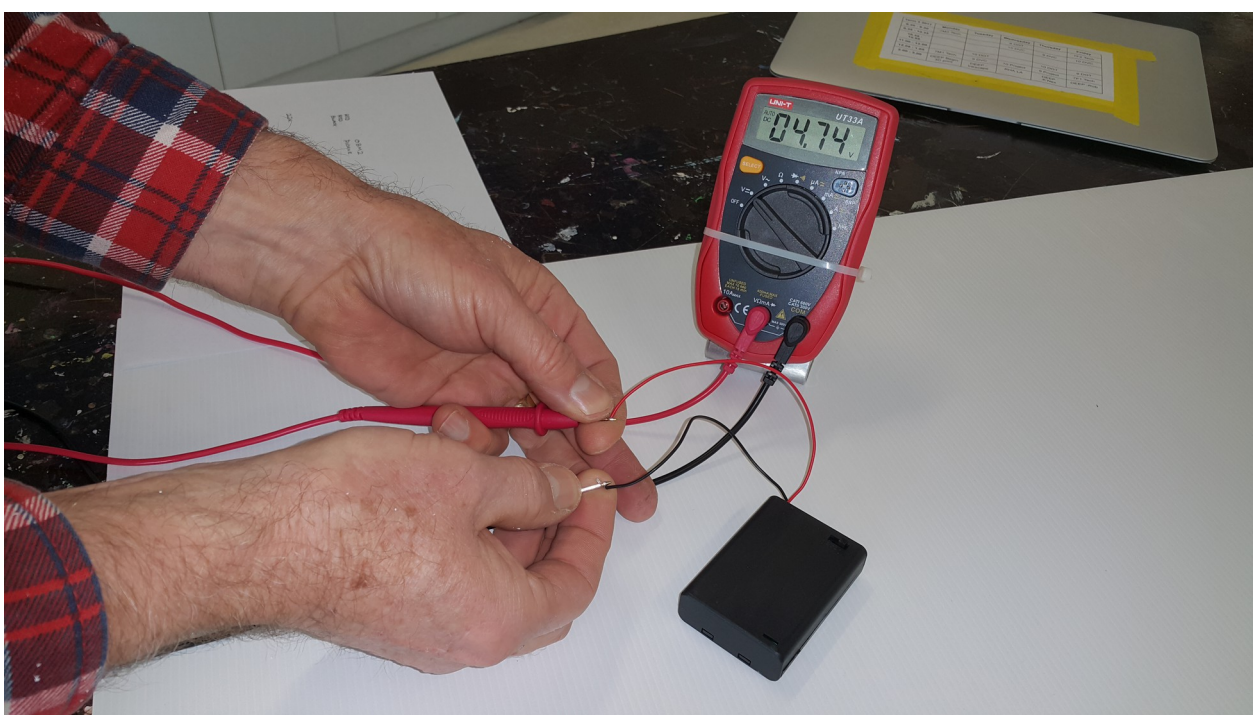




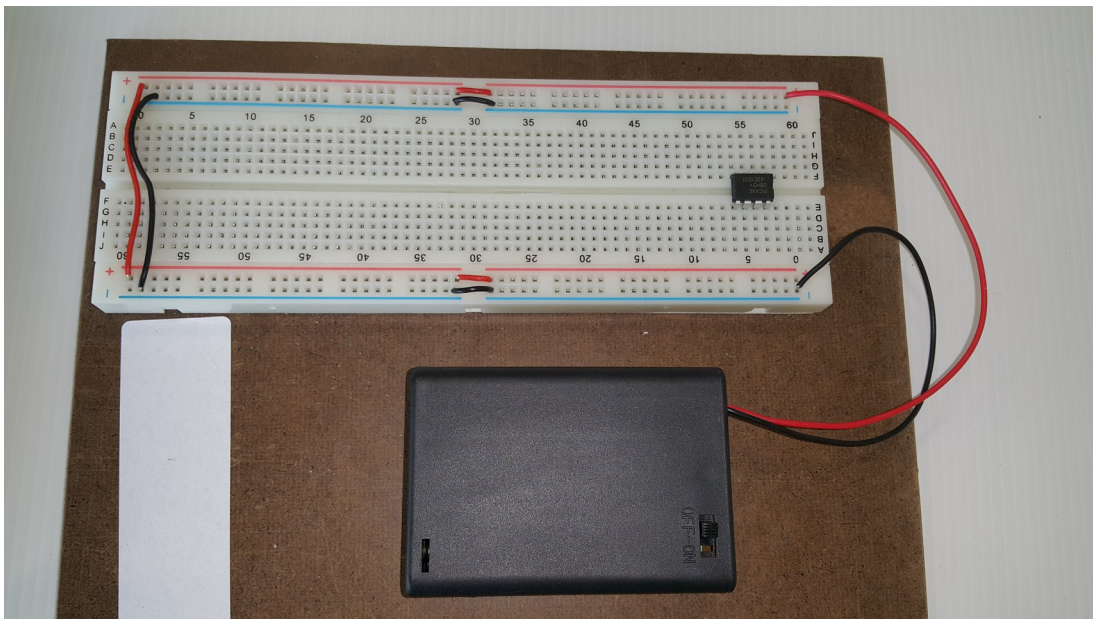
3- Prepare your Breadboard, connect the positive and negative lines and then insert the PICAXE as shown:



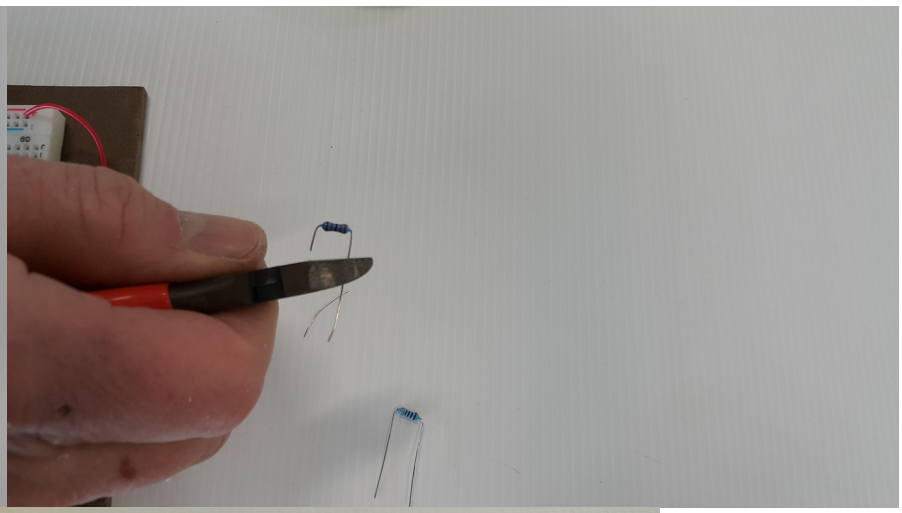
4- Prepare the battery box with 3 AA batteries in it. Test the voltage using the multi meter on V-. Any reading over 3.6V will be enough for the PICAXE to operate otherwise, change the batteries.

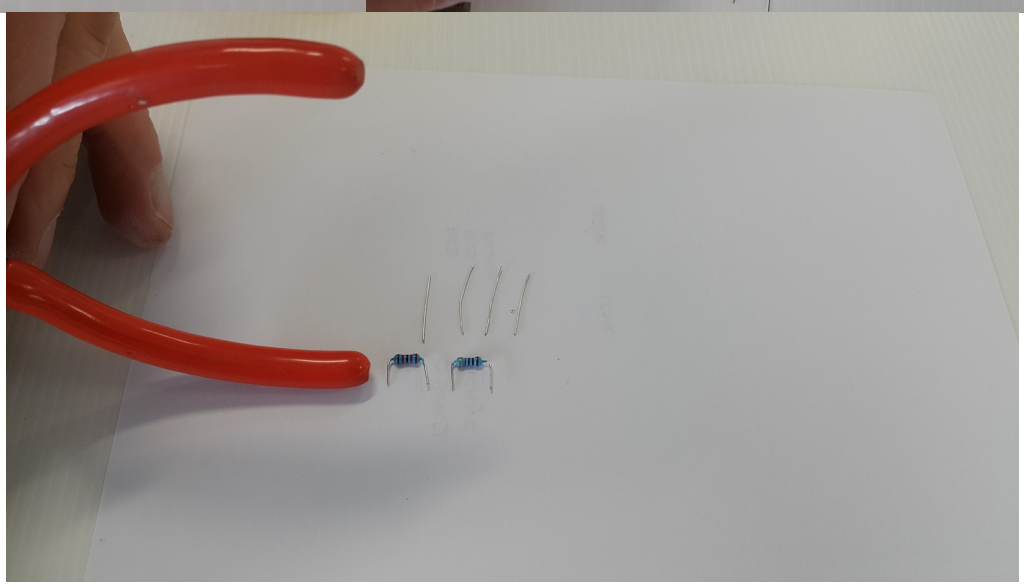


5- Connect the battery box wires to the positive and negative terminals on the breadboard.

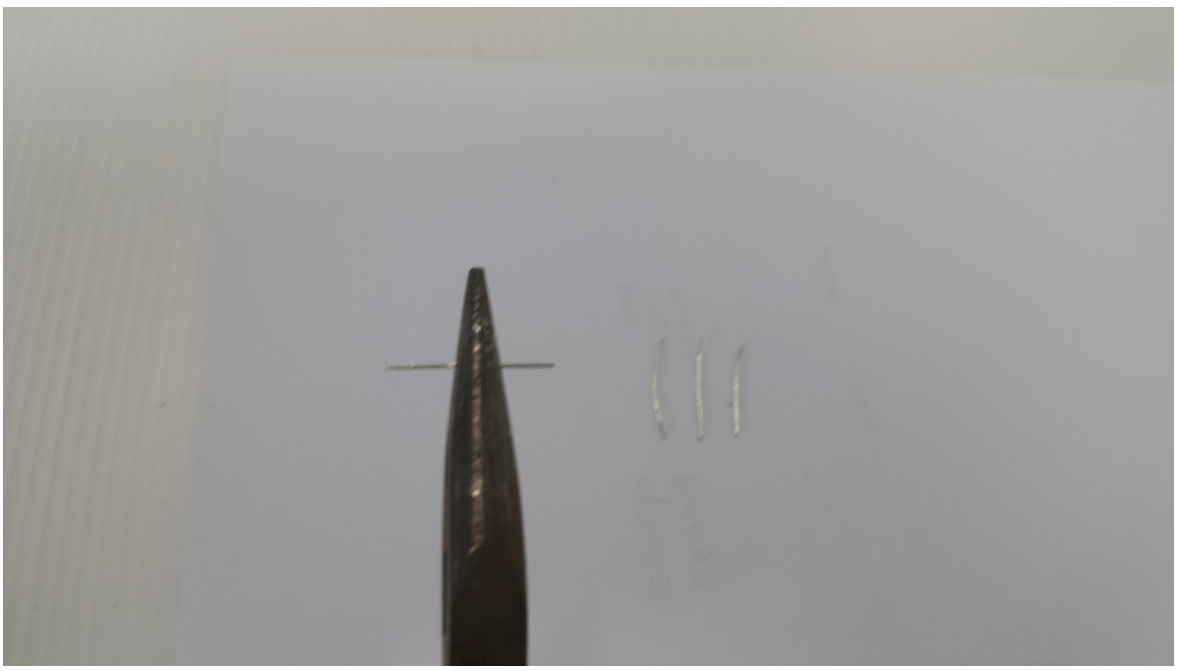


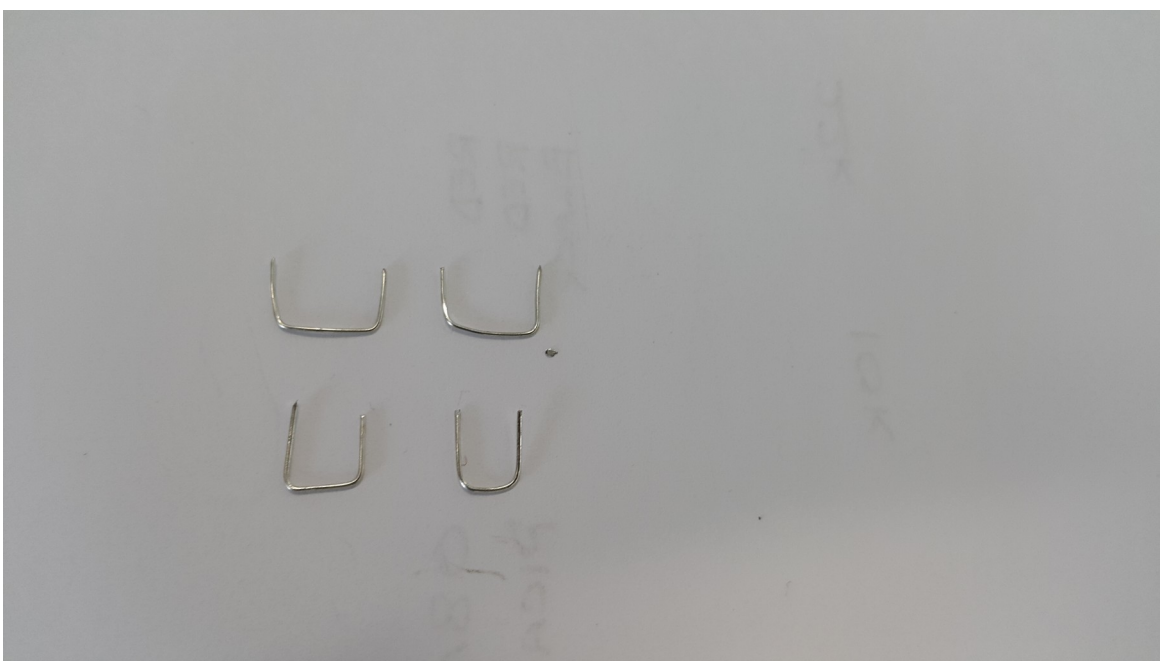
6- Trim the resistors legs to about 8mm length as shown. Keep the waist cuts to use them as connectors. They are hard enough to insert in in the breadboard.



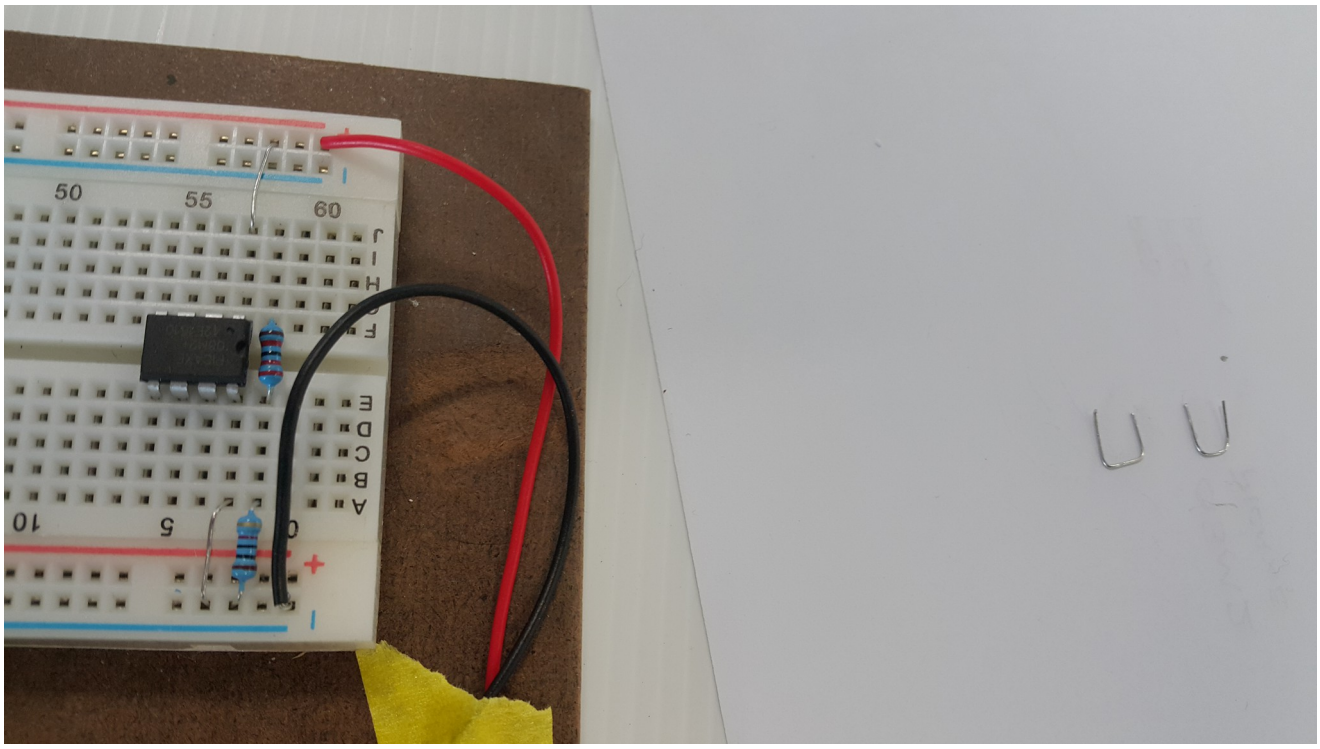


7- Bend the wires as shown using pliers. You should have 2 pairs pent at different lengths:

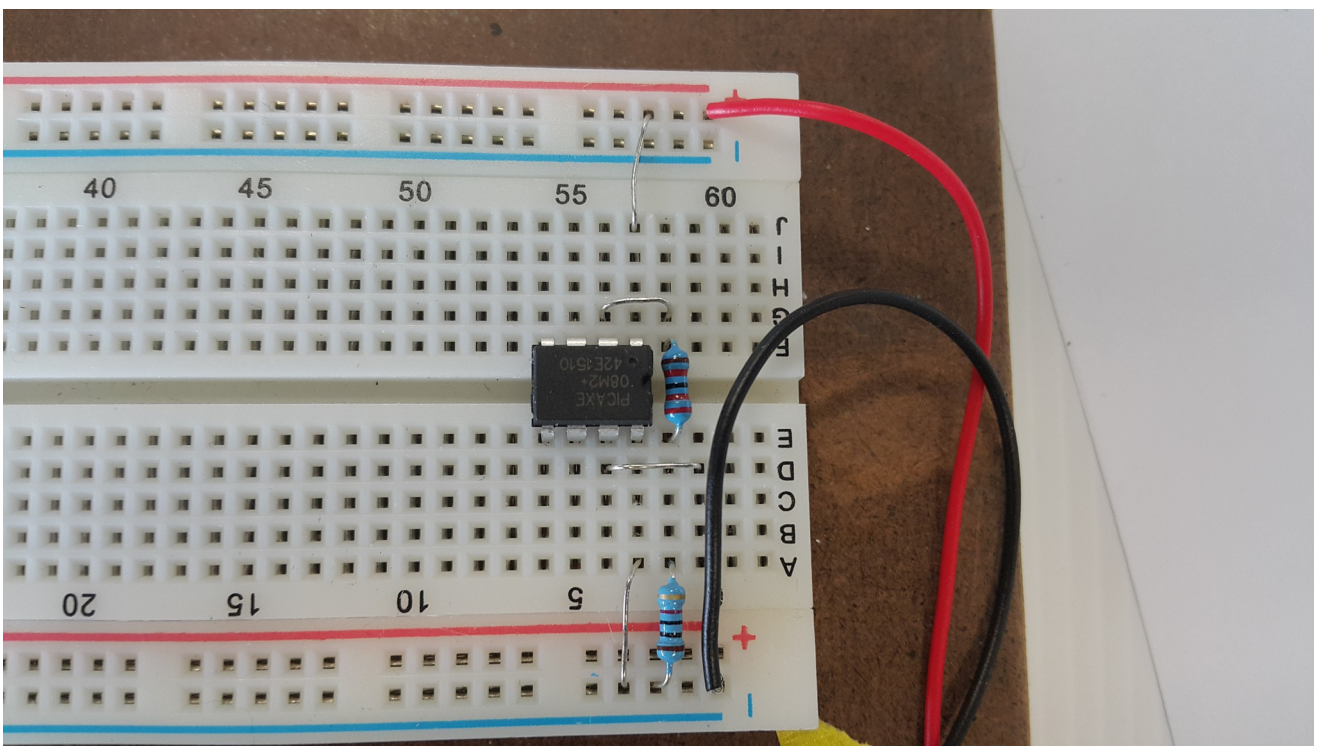




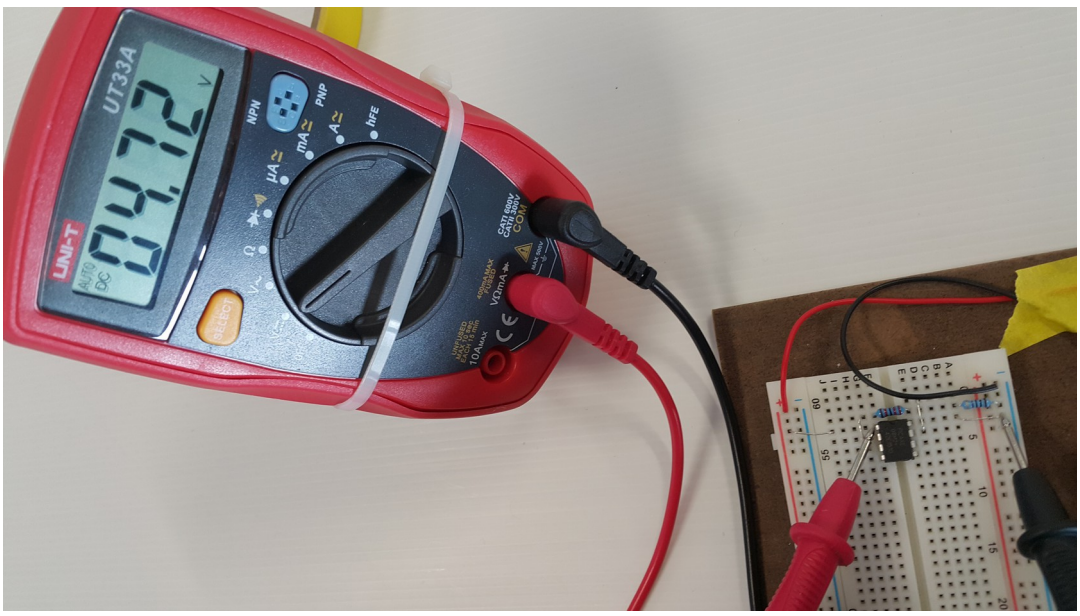
8- Insert the 2 resistors and the bigger pair of wires into the breadboard as shown:



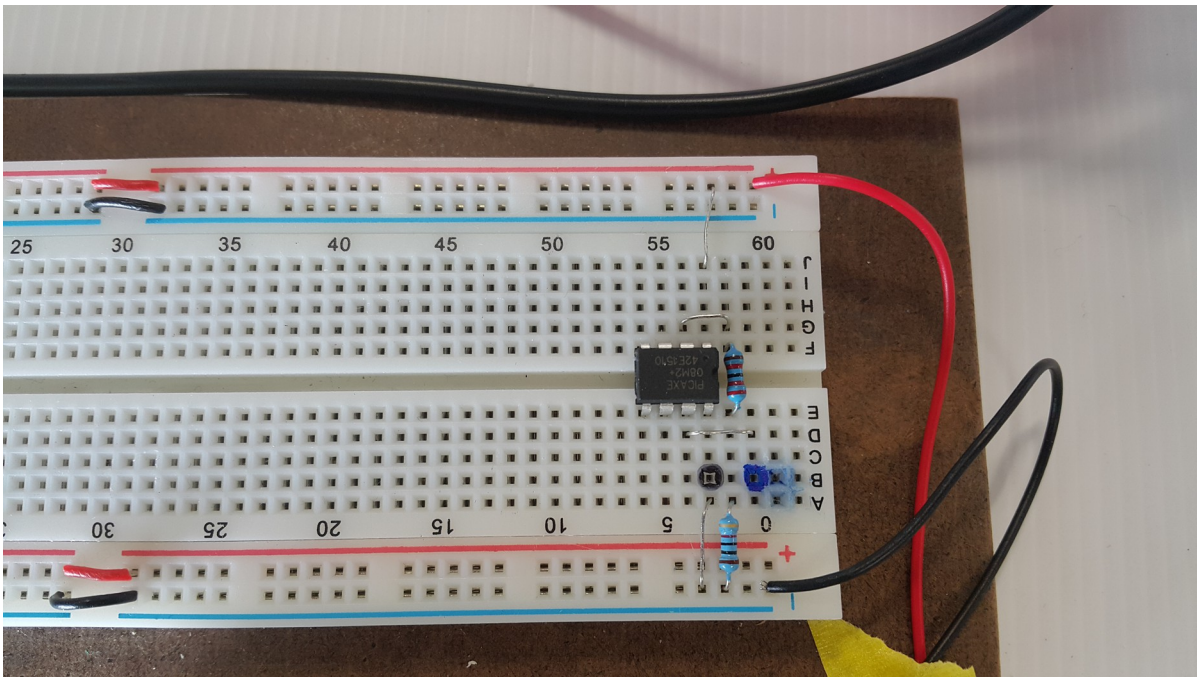
Then the smaller pair of wires:



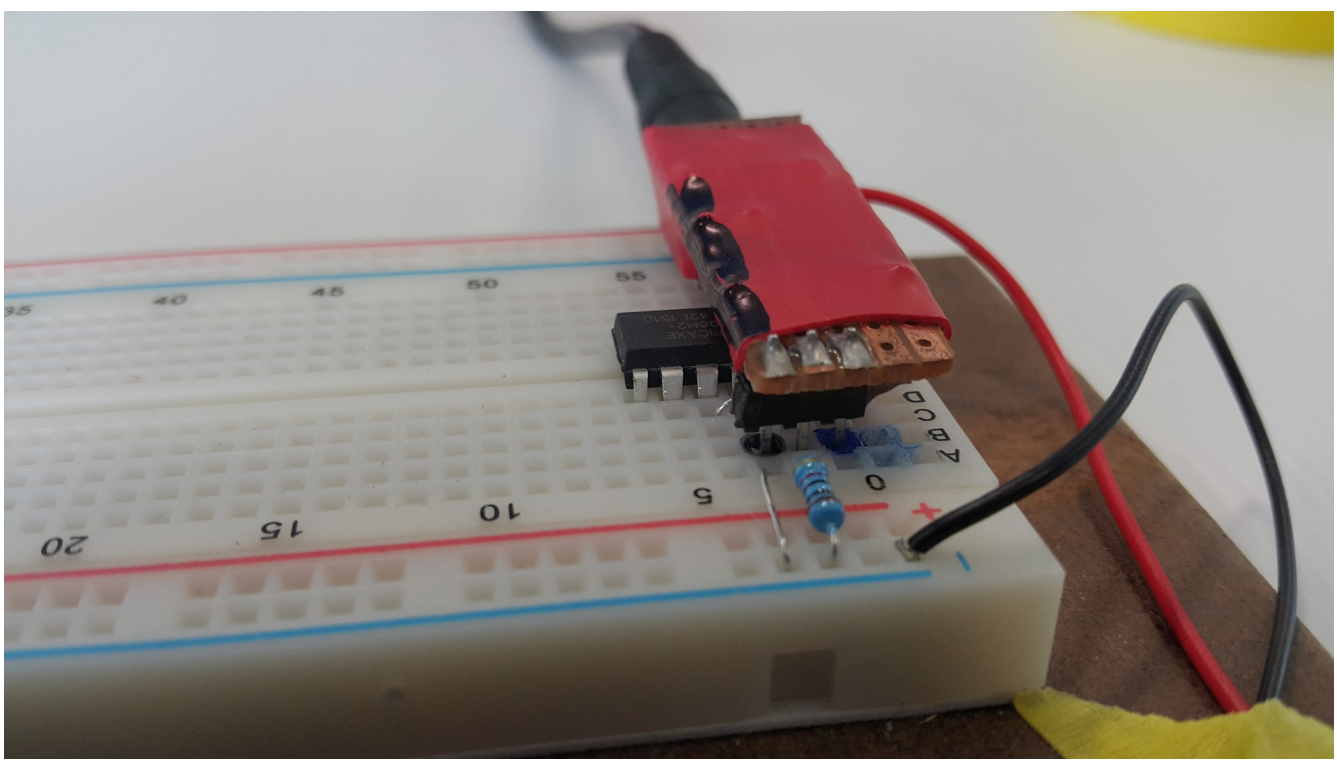
9- Lets test all the connections to make sure they are all well connected.
Start with the voltage across the PICAXE and then across the components lines.
You should have the same voltage of the batteries.



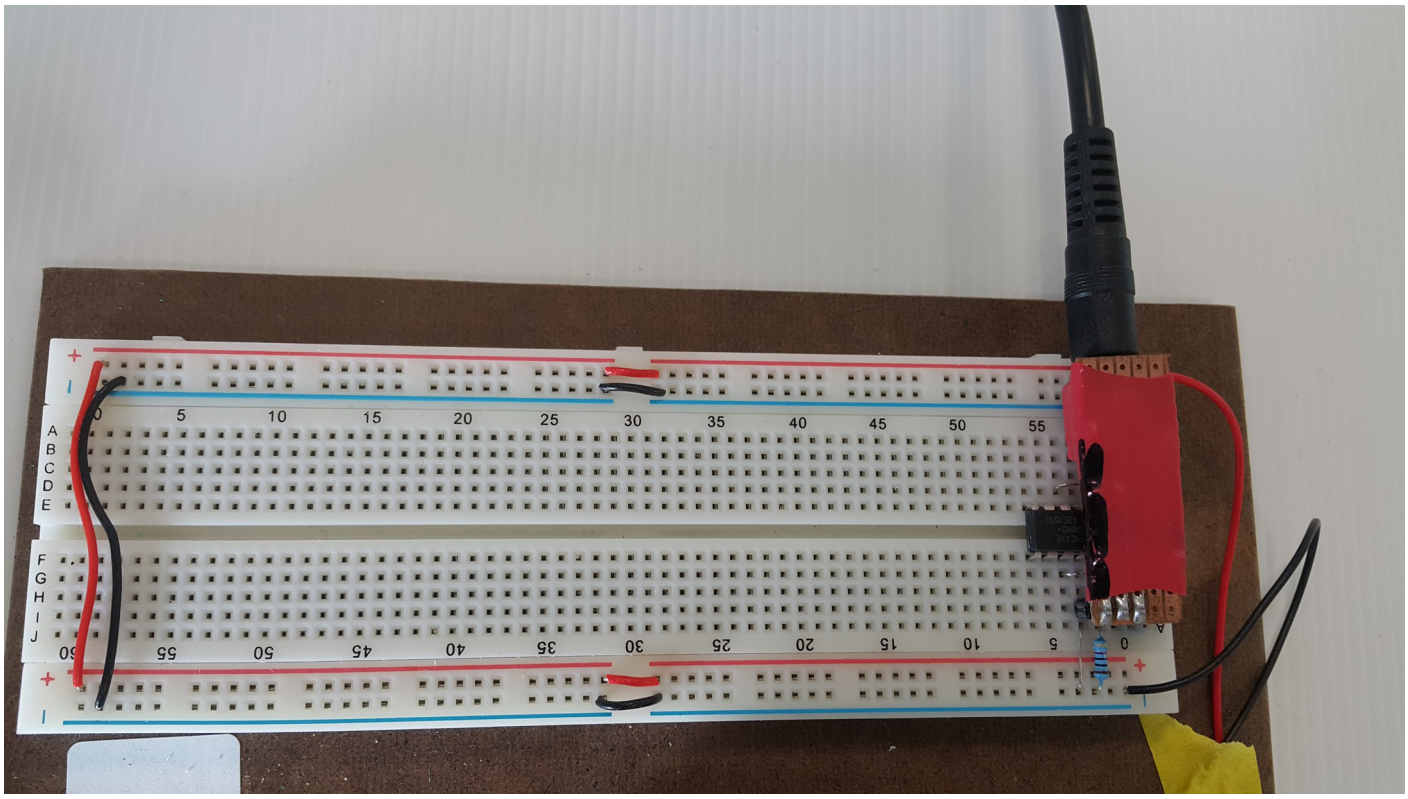
10- Using marker or vivid pen. Mark 2 dots (one blue and one black) on the breadboard as shown:

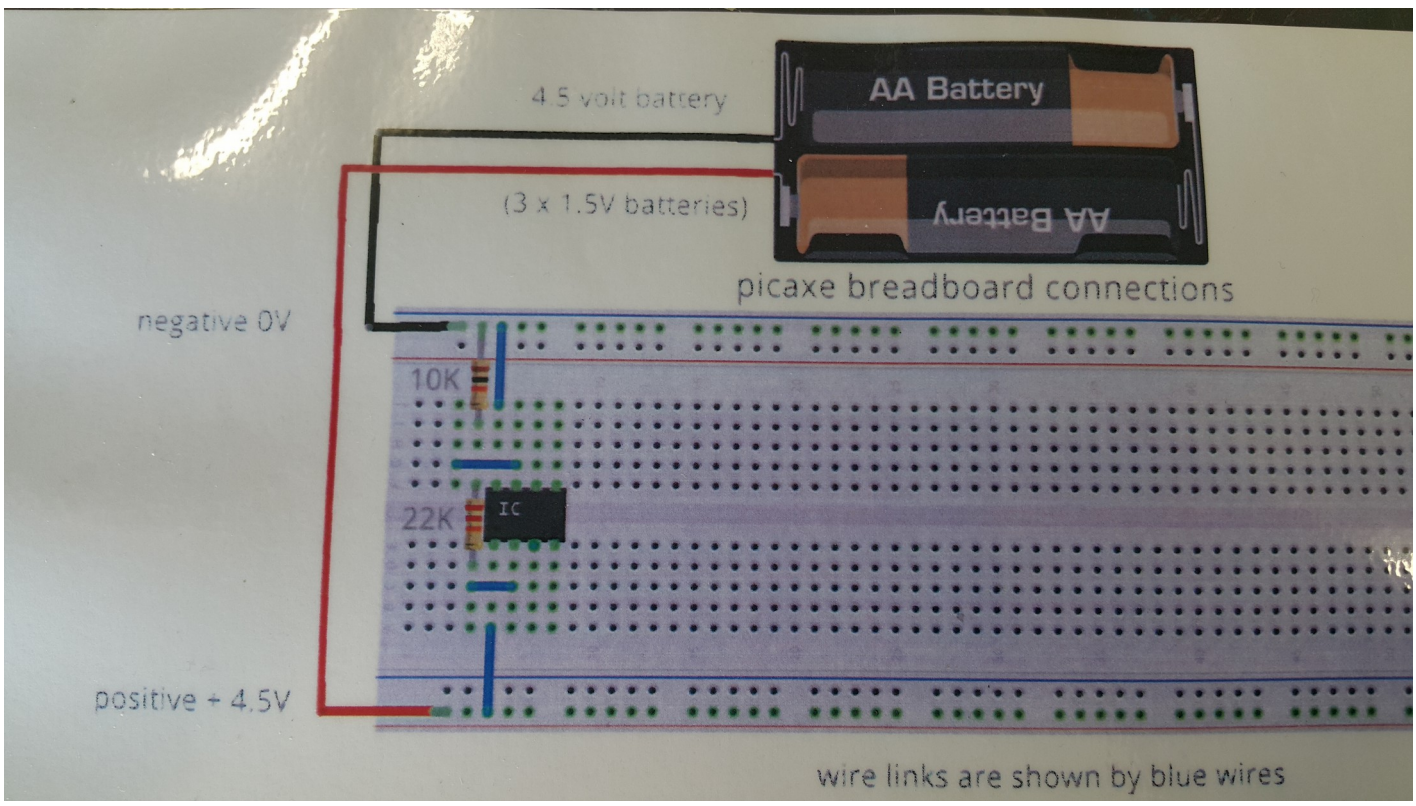


11- Finally insert the (3pins) of the data cable (programming cable) into the bread board. The pin near the black marking on the cable chip goes in the black dot on the breadboard.



Final circuit layout and data cable:





Lets start programming – Coding:

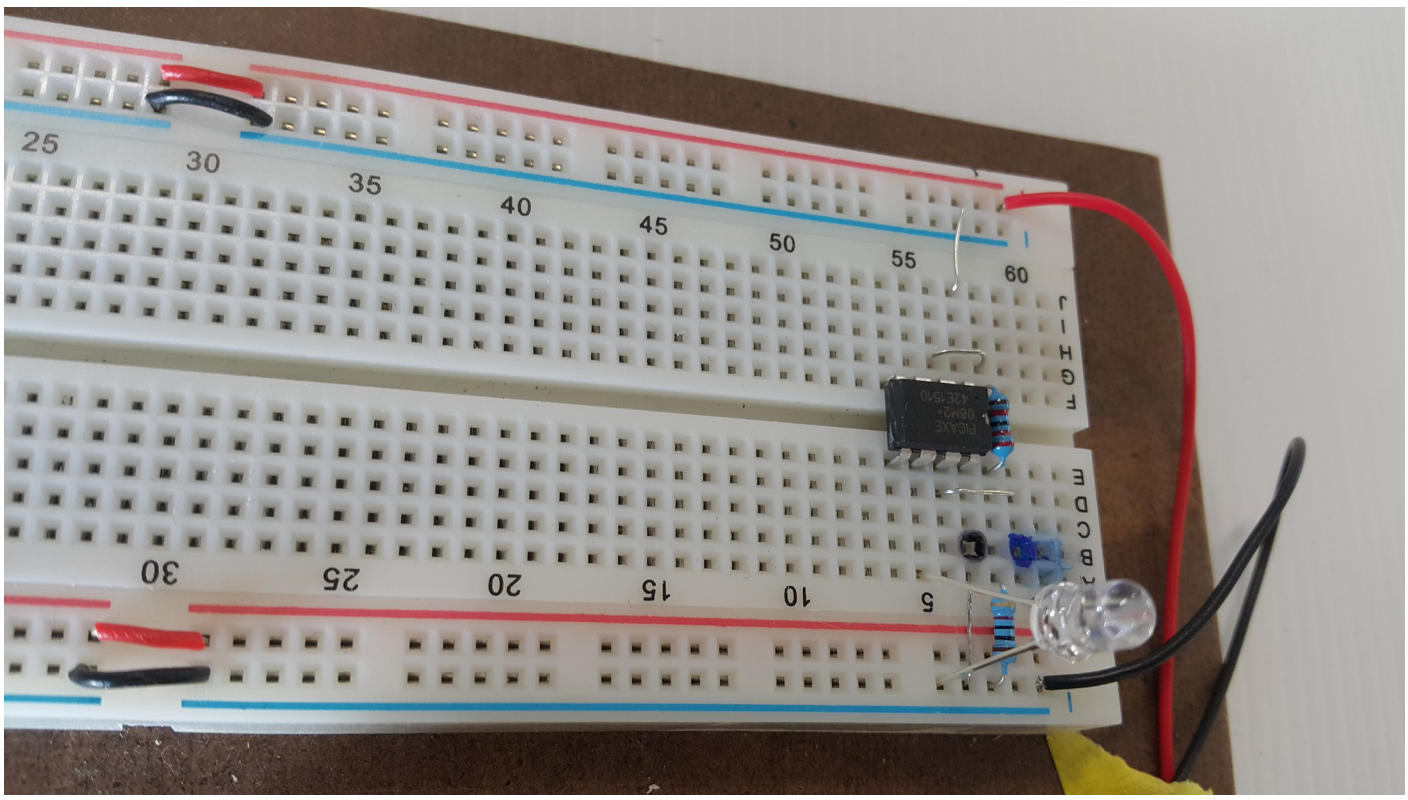
You need to create a Picaxe folder then write the code and save it in that folder.

Remember: Any script you write after the **apostrophe '** will not be part of your code (as shown in the green script). You can use it to write instructions or descriptions.

Always start with program title, written by, version and date. (Notice the use of the apostrophe ' at the beginning of the script).

1- **Test download with one LED.**

Circuit layout:



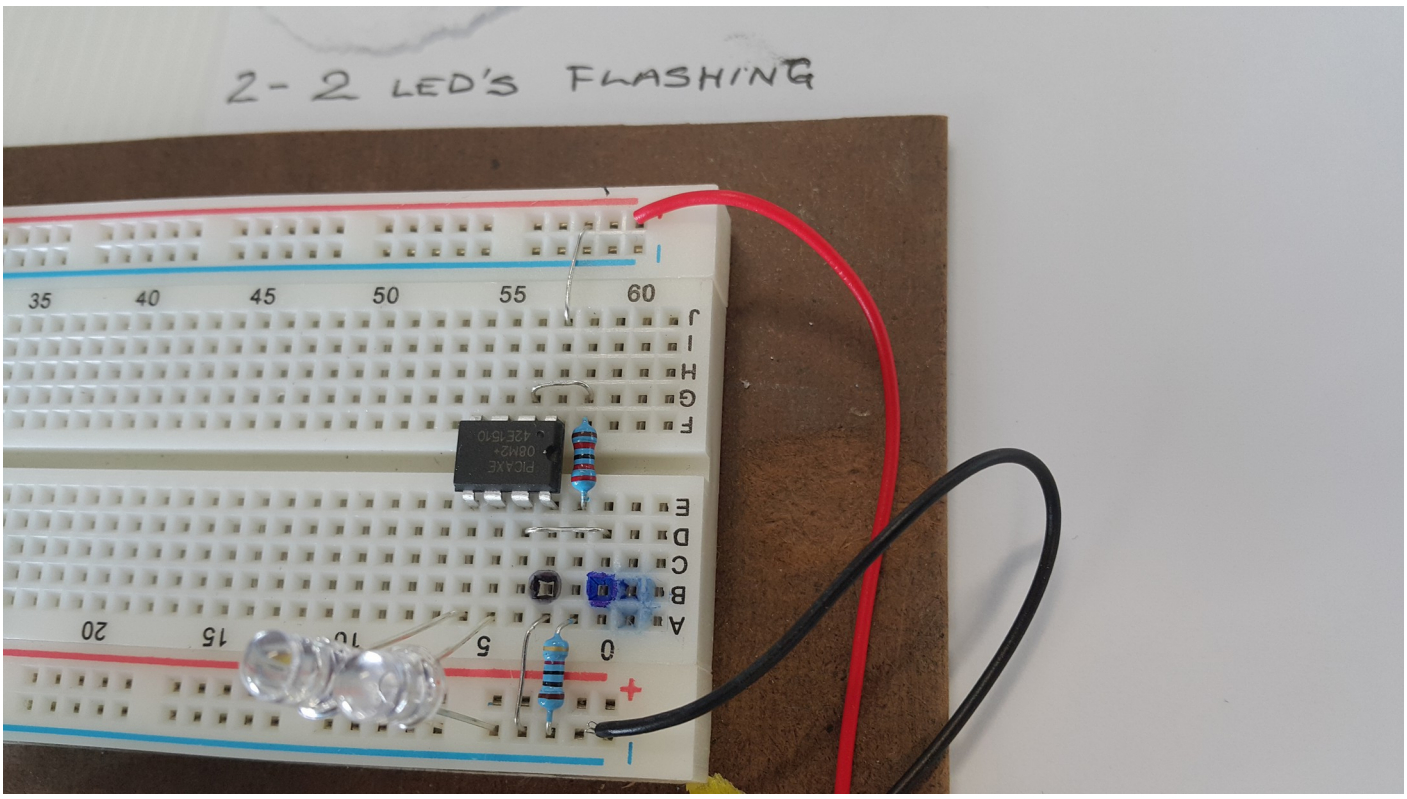
The code:

1:0 Use Check Syntax to create label list

```
'programme title – test download function
'written by – R Hargreaves
'version number – V1
'date – 17.7.2017

start:                    'a label for the start of the programme
      high c.1            'turn on pin c.1
      pause 1000          'pause for 1000 milliseconds (1 second)
      low c.1             'turn off pin c.1
      pause 500           'pause for 500 milliseconds (1/2 second)
      goto start          'go to the start label
```

2- 2 LED flashing:

Circuit layout:

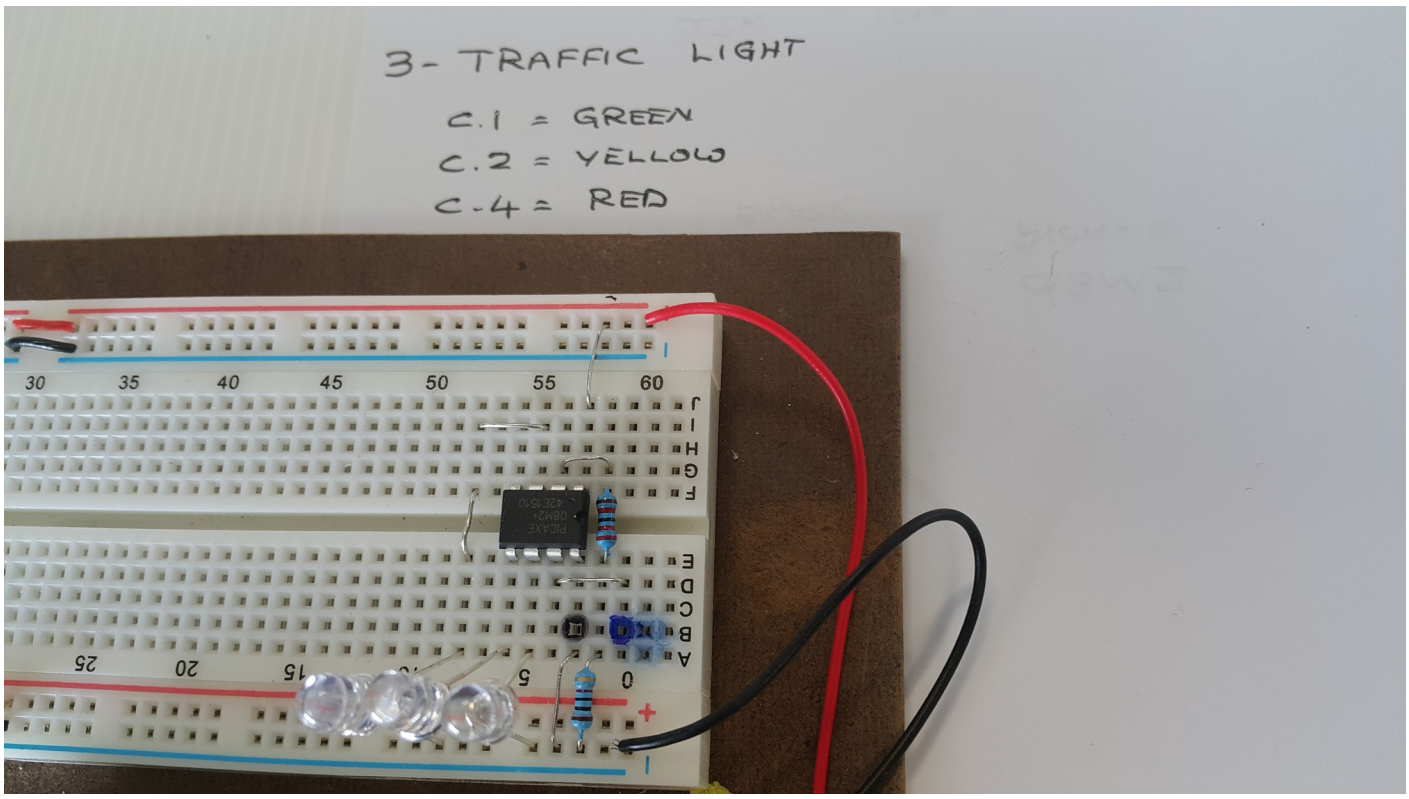


The code:

```
1:0  Use Check Syntax to create label list
1   'programme title - Flash LED's
2   'programme function - two led's flashing on/off
3   'written by - R Hargreaves
4   'version number - V1.1
5   'date - 17.7.2017
6
7
8   start:                  'a label address at the beginning of the programme
9   high c.1               'turn on pin c.1
10  low c.2                 'turn off pin c.4
11  pause 1000             'pause for 1000 milliseconds (1 second)
12  low c.1                 'turn off pin c.1
13  high c.2                'turn on pin c.4
14  pause 1000             'pause for 1000 milliseconds (1 second)
15  goto start             'go to start and repreat the programme
16
17
```

3- Traffic light:

Circuit layout:



The code

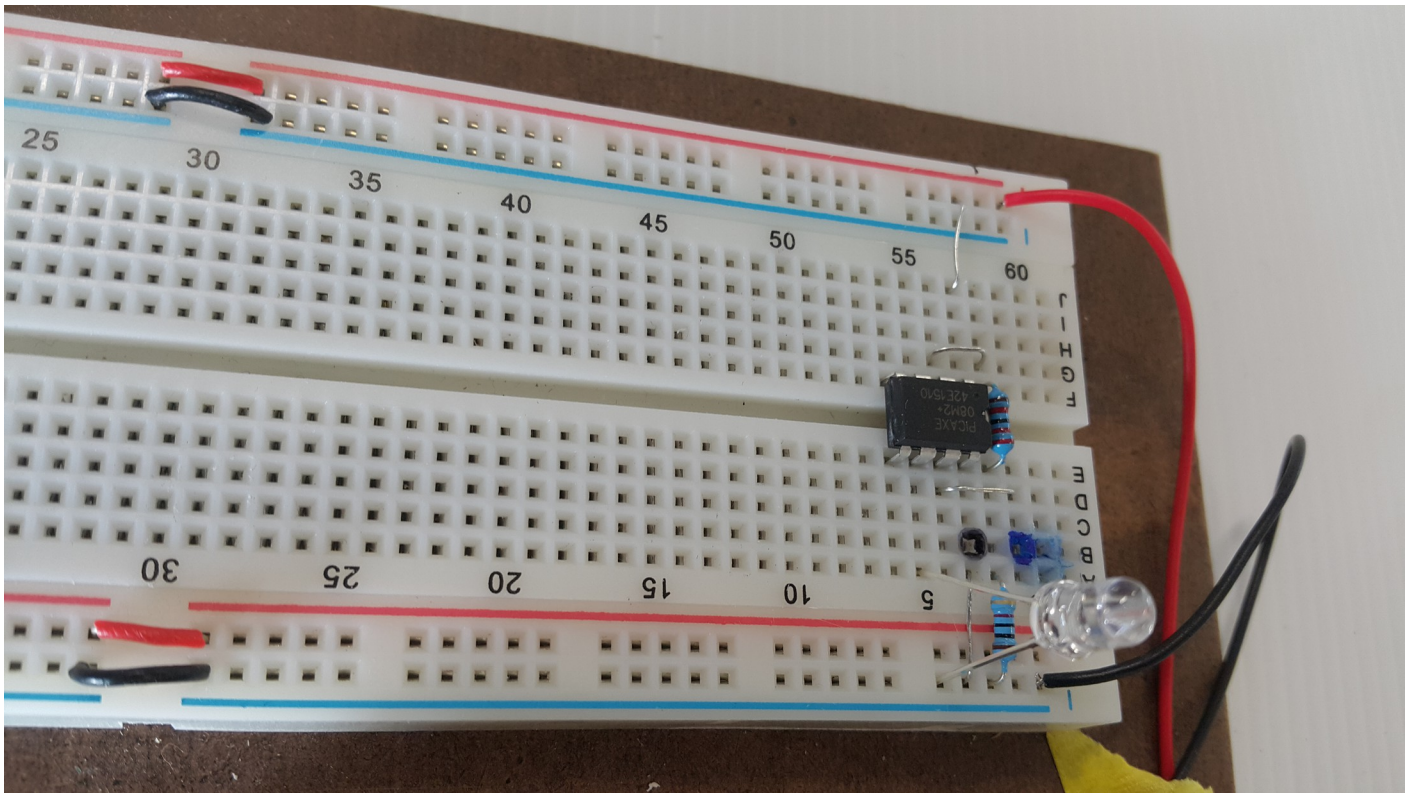
3- traffic lights.bas - AXEpad

1:0 Use Check Syntax to create label list

```
'programme title - traffic lights using led's
'written by - R Hargreaves
'version number - V1
'date - 17.7.2017

start:                  'start of the programme
wait 2                   'a 2 second delay at the start of the programme
high c.1                   'turn on pin c.1 green led
pause 1000                      'wait for 1 second
low c.1                    'turn off pin c.1 green led
pause 500                   'wait for 1/2 second
high c.4                  'switch on pin c.4 yellow led
pause 1000                      'wait for 1second
low c.4                    'turn off pin c.4 yellow led
pause 500                   'wait for 1/2 second
high c.2                  'turn on pin c.2 red led
pause 1000                      'wait 1 second
low c.2                    'turn off pin c.2 red led
pause 1000                      'wait for 1second
goto start                 'go to the start of the programme and repeat
```

4- Flash an LED 4 times

Circuit layout:



The code:

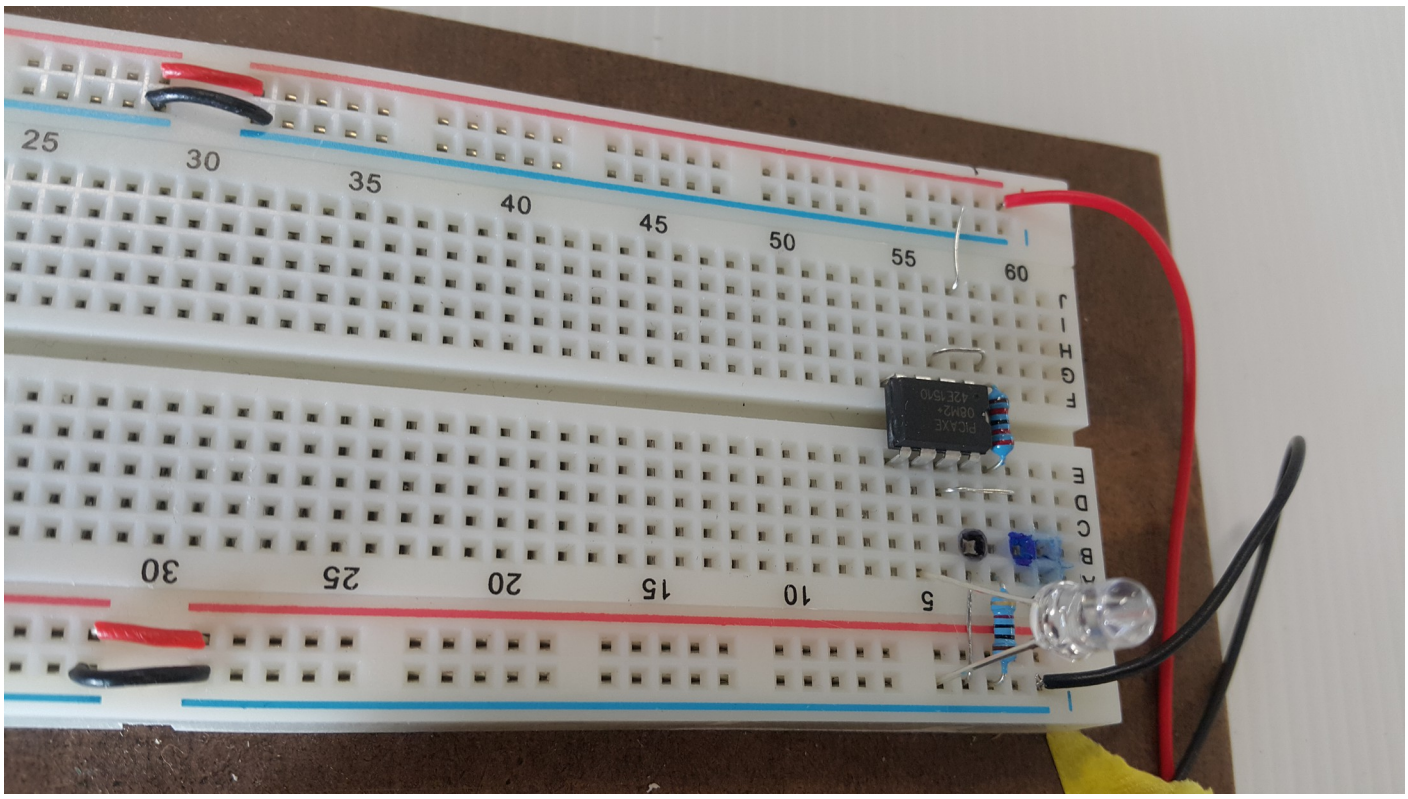
Flash an led four times.bas - AXEpad

19:7 Use Check Syntax to create label list

```
1  'programme title - Flash an LED four times
2  'written by - R Hargreaves
3  'version number - 1.1
4  'date - 24.7.2017
5
6  #no_data            'a command that cuts the programme download time by 1/2
7  start:
8  wait 2
9  high c.1
10 pause 500
11 low c.1
12 pause 500
13 high c.1
14 pause 500
15 low c.1
16 pause 500
17 high c.1
18 pause 500
19 low c.1
20 pause 500
21 high c.1
22 pause 500
23 low c.1
24 pause 500
25 goto start
26
```

5- Flash LED using for next loop:

Circuit layout:



The code:

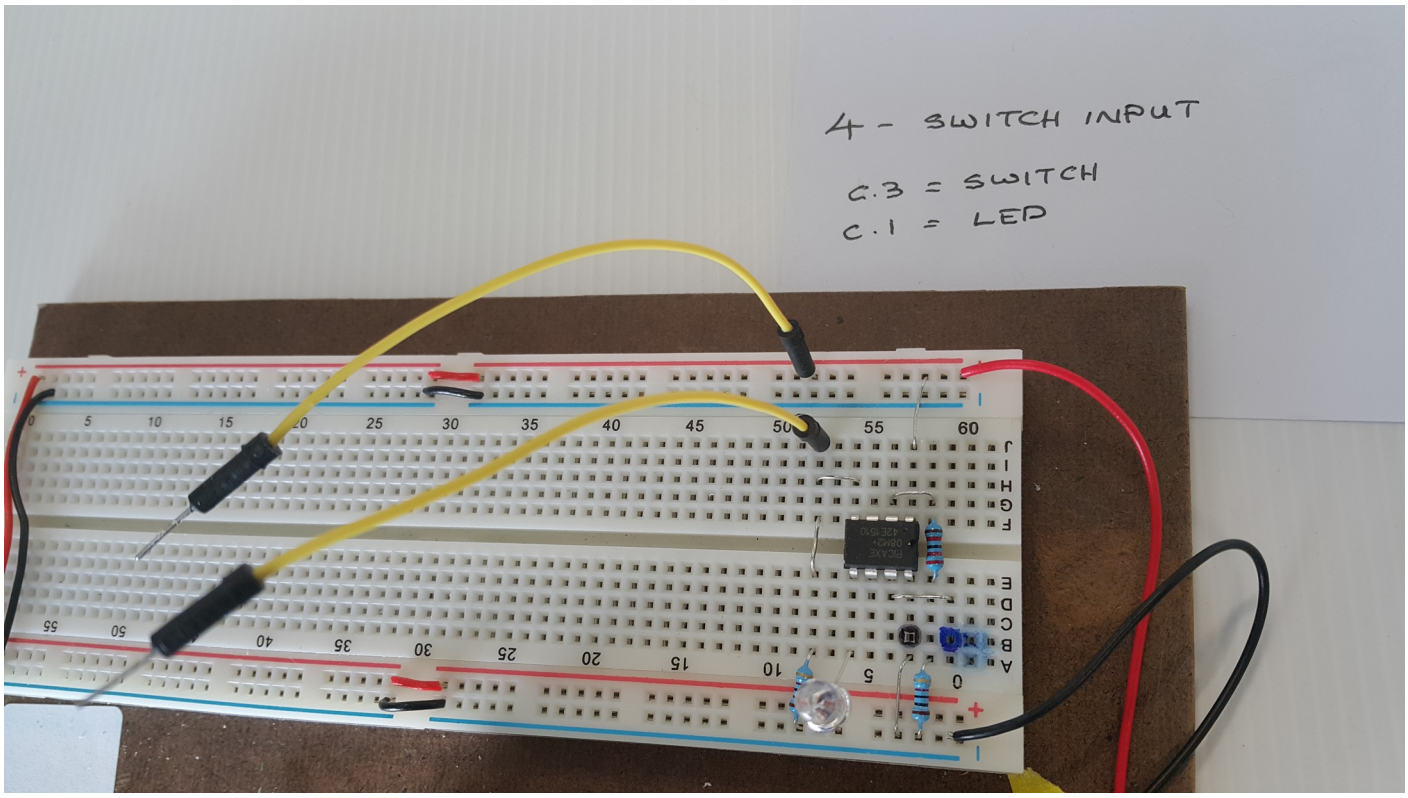
5- flash led using a for next loop.bas - AXEpad

1:0 Use Check Syntax to create label list

```
1  'programme title - flash LED with a for-next loop
2  'written by - R Hargreaves
3  'version number - V1.1
4  'date - 17.7.2017
5
6  #no_data            'a command to reduce programme download time by 50%
7  start:              'start of the programme
8  wait 2              'wait for 2 seconds so you can see when the loop starts again
9  for b1=1 to 4       'count up to 4 places - b1 is the place where the counts are stored, 1 to 4 means count to 4 places
10 high c.1            'turn on pin c.1
11 pause 500           'wait for 500 milliseconds
12 low c.1             'turn off pin c.1
13 pause 500           'wait for 500 milliseconds
14 next                'go to the next part of the count sequence
15 goto start          'go to the start of the programme and repeat
16
```

6- Switch input with LED:

Circuit layout:



The code:

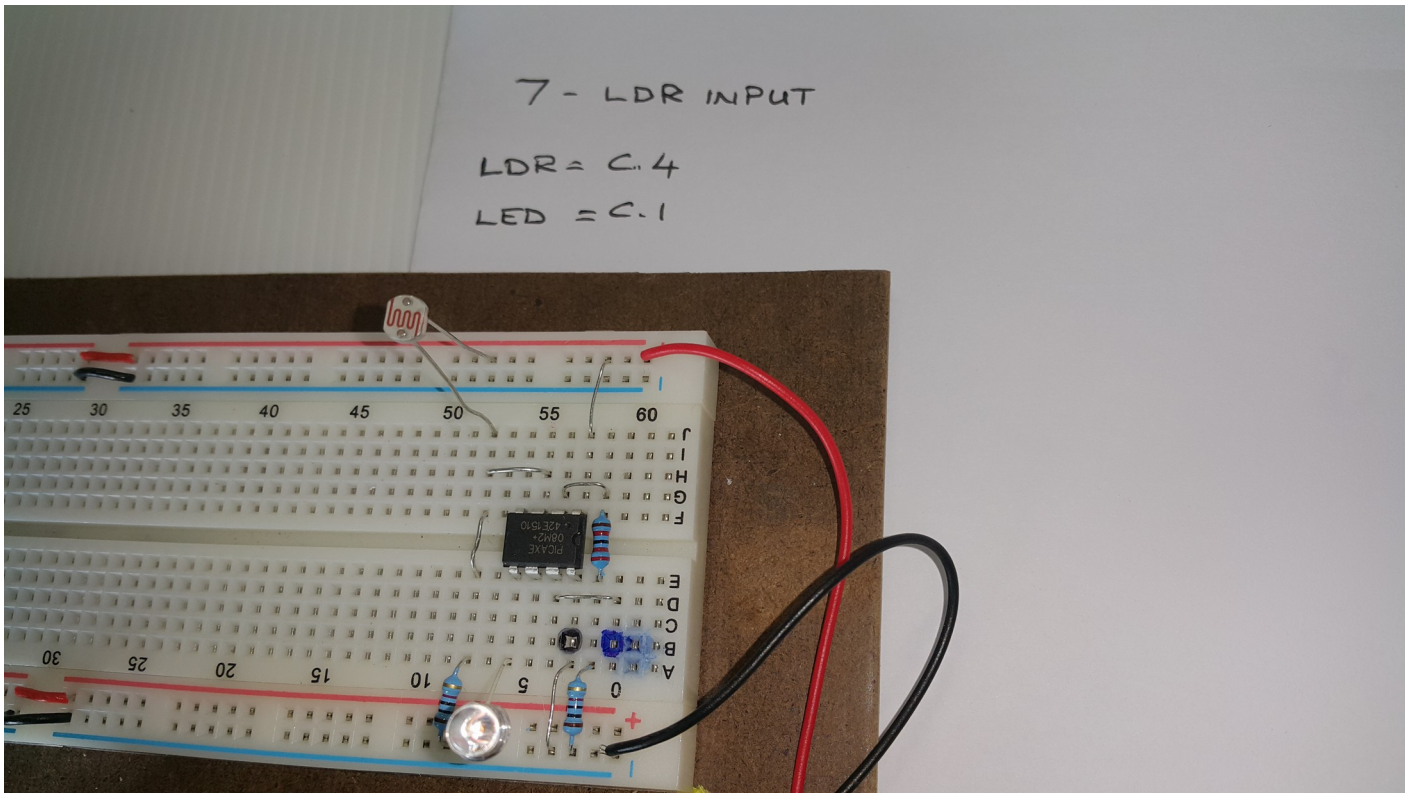
6- switch input with led.bas - AXEpad

1:0 Use Check Syntax to create label list

```
'programme title - switch input
'written by - R Hargreaves
'version number - V!
'date - 17.7.2017

start:                          'a label for the start of the programme
      if pinc.3 =1 then ledon     'if the switch on pin c.3 is pressed on then jump to subroutine named ledon
      if pinc.3 =0 then ledoff    'if the switch on pin c.3 is not pressed on then jump to subroutine named ledoff

ledon:                          'label for subroutine named ledon
      high c.2                    'turn on pin c.2
      pause 2000                  'wait for 2 seconds
      goto start                'go to the start label

ledoff:                         'label for subroutine named ledoff
      low c.2                     'turn pin c.2 off
      goto start                'go back to start and repeat programme
```

7- LDR (Light Dependent Resistor) input. Read ADC command.

Circuit layout:



The code:

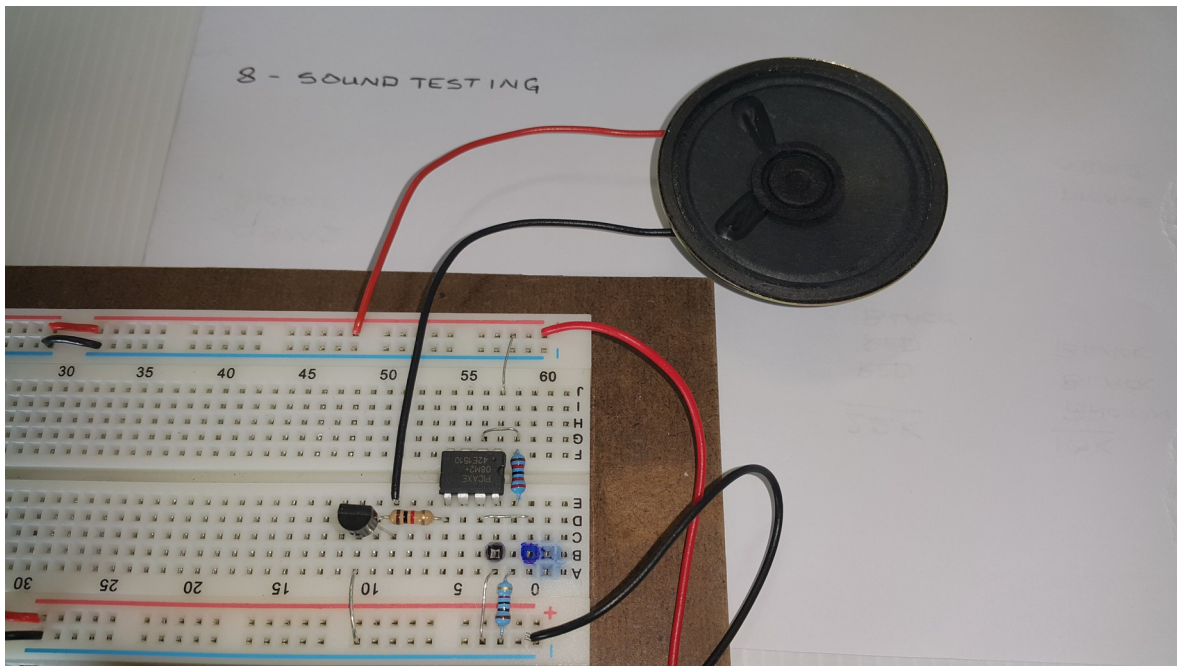
7- LDR input readadc command.bas - AXEpad

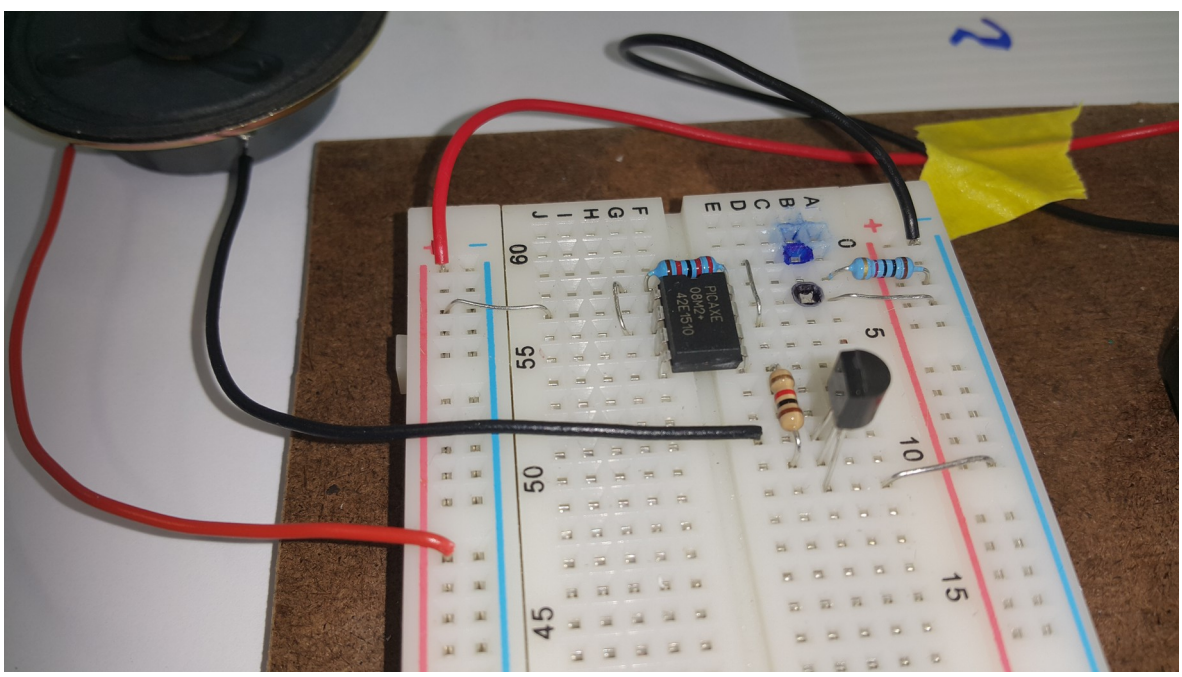
8:9 Use Check Syntax to create label list

```
'programme title - LDR input
'written by - R Hargreaves
'version number - V1
'date - 17.7.2017


start:                  'a label begining the programme
readadc 4,  b1          'Read the analogue voltage (scaled 0 to 255) on pin c.1 and save the digital value in storage place b1
debug b1                'Display the variables in b1 on the screen (press F6 to display the screen)

if b1>150 then ledon    'if the value in b1 is greater than 150 then goto label ledon
if b1<150 then ledoff   'if the value in b1 is less than 150 then goto label ledoff
goto start              'Go back to the beginning again to check the values

ledon:                  'subroutine label  for ledon
high c.2                'turn on pin c.2
goto start              'Go back to the beginning again

ledoff:                 'subroutine label  for ledoff
low c.2                 'turn off pin c.2
goto start              'Go back to the start of the programme
```

8- O8M Sound testing:

Circuit layout:





The code:

8- O8M sound testing.bas - AXEpad

1:0 Use Check Syntax to create label list

```
'programme title - test the sound output
'written by - R Hargreaves
'version number - V1
'date - 17.7.2017

start:                  'start of the programme
sound c.2,(65,100)      'play sound using pin c.2, 65 is the tone (from 1 to 120), 100 is the length of time to play in milliseconds
sound c.2,(78,100)      'same as above line except tone has changed to 78
sound c.2,(88,100)      'same as above line except tone has changed to 88
sound c.2,(119,100)      'same as above line except tone has changed to 119
goto start               '
```

9- Sound playing a tune:

Same circuit layout above.

The code is:

9- O8M sound playing a tune.bas - AXEpad

10:0 Use Check Syntax to create label list

```
'programme title - play a sound using a speaker
'written by - R Hargreaves
'version number - V1
'date - 17.7.2017

start:                  'start of the programme
                        'play TheSimpsons tune
sound c.2, 4, ($46,$66,$66,$4A,$66,$50,$66,$53,$51,$66,$66,$4A,$66,$46,$66,$43,$40,$40,
$40,$41,$66,$66,$60,$61,$40,$40,$40,$41,$66,$66,$60,$61,$46,$66,$66,$4A,$66,$50,$66,$53,
$51,$66,$66,$4A,$66,$46,$66,$43,$40,$40,$40,$41,$66,$66,$60,$61,$40,$40,$40,$41,$66,$66,$60,$61,$26)
'sound with speaker on pin c.2, 4 is the speed at which to play the sound, $46,$66 ..... is the code for each note of the sound
wait 2              'wait for 2 seconds
goto start      'go to the start and repeat the sound
```

To connect your breadboard to your device and transfer the coding data into PICAXE editor check the link below:

https://drive.google.com/a/mhjc.school.nz/file/d/0B9hRsiioOJuaZXJHSHJucmZYMXM/view?usp=sharing