

# FINAL PROJECT

## ENGDUINO RACE (STEP COUNTER)

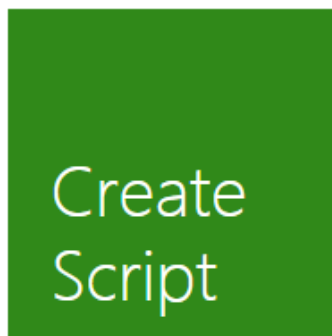
In this tutorial, we will teach you to write the code to make an Engduino race meter! Let's call our little Engduino race meter a Pedometer.

Our Pedometer will count the number of steps the user take and light up the Engduino's LED according to the number of steps he/she takes! So at the end of the race you and your friends can take a healthy run then compare to each other who wins the race by counting the total of LEDs lights that lit!

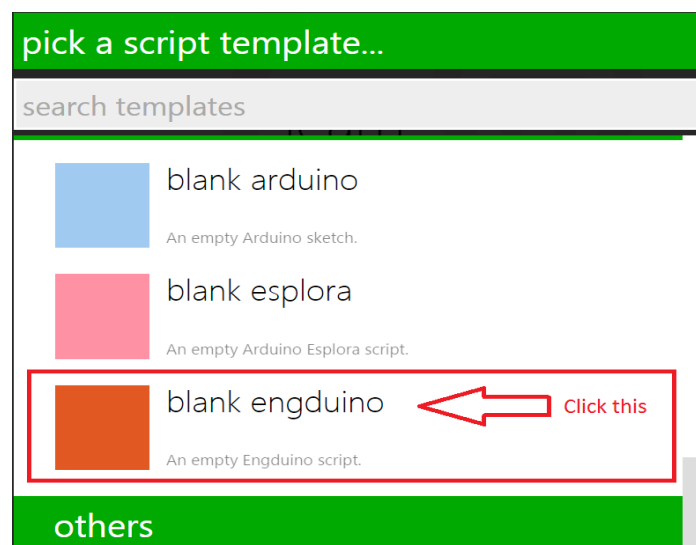
We are able to use the Engduino to create a simple Pedometer by using the accelerometer on the Engduino. The Pedometer counts the number of steps the user takes by calculating the acceleration from the Engduino.

## GETTING STARTED!

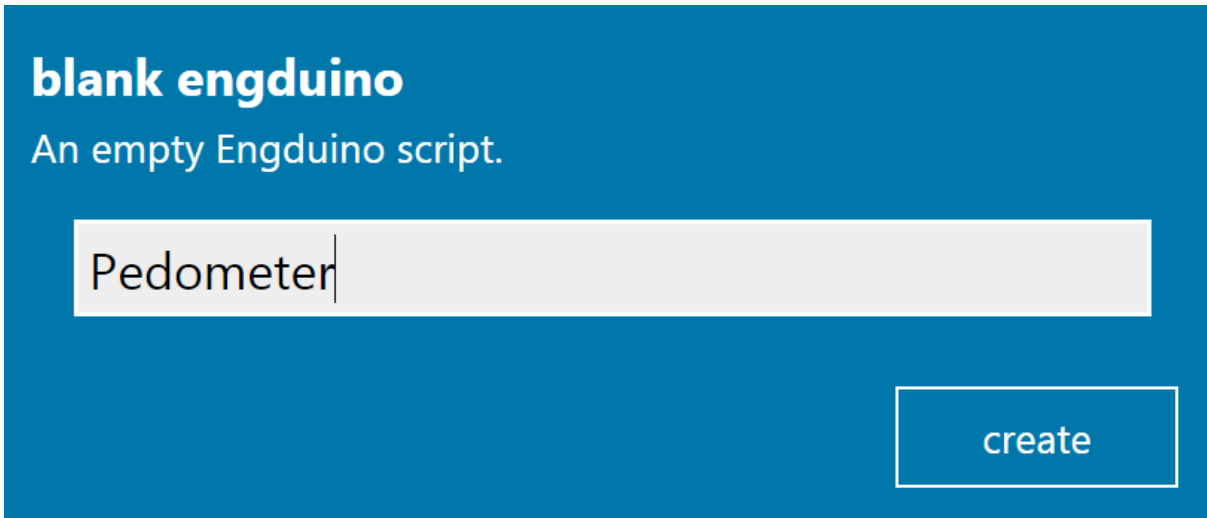
Before we can create our Pedometer, we have to set up the script first!



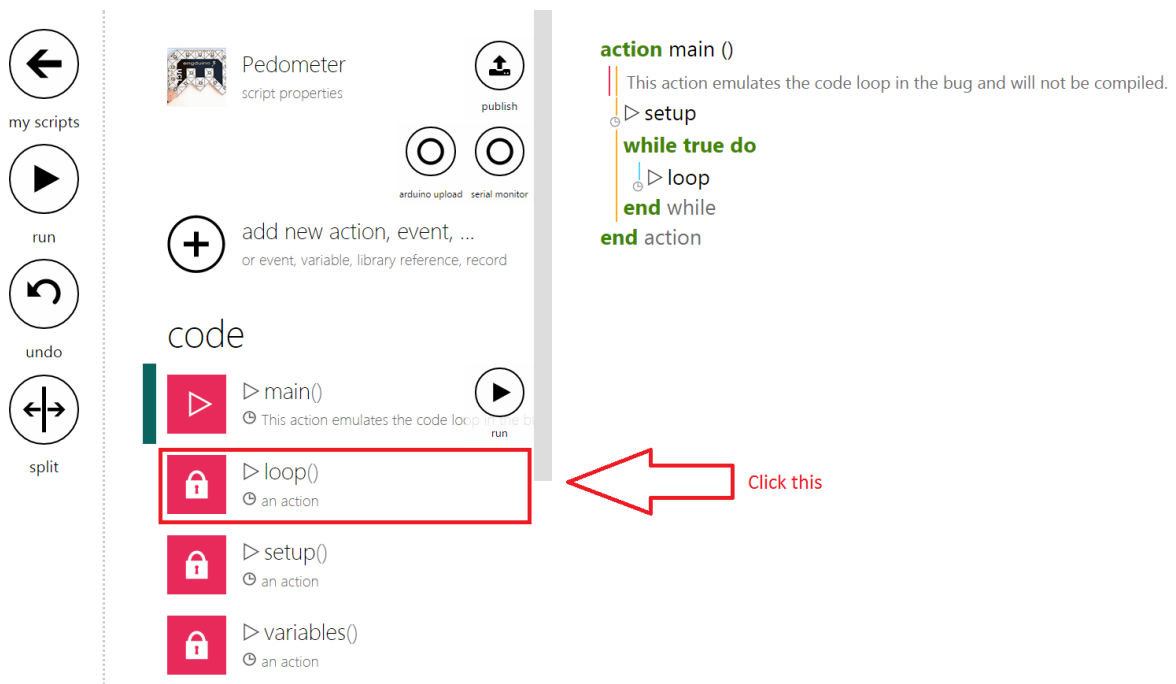
First, click on "Create Script"



Next, click on "blank engduino"



Name your new script as "Pedometer" and click create



This is the main page of your script! But before we start coding, we need to delete some codes from the template!

my scripts

run

undo

split

Pedometer  
script properties

publish

arduino upload serial monitor

add new action, event, ...  
or event, variable, library reference, record

code

- main()
  - This action emulates the code loop in the Arduino IDE
- loop()
  - an action
- setup()
  - an action
- variables()
  - an action

private action loop ()

engduino → set all LEDs(colors → random)

engduino → delay(200)

engduino → set all LEDs(colors → black)

engduino → delay(200)

end action

Click on the "private action loop()" to bring up the menu to delete the loop function.

dismiss

run

undo

split

action

loop

private action

Private actions do not get a run button.

'atomic' action [more info](#)

+ add input parameter  
create a new parameter

+ add output parameter  
create a new parameter

cut

copy

delete

Ready for more options? [Change skill level!](#)

private action loop ()

engduino → set all LEDs(colors → random)

engduino → delay(200)

engduino → set all LEDs(colors → black)

engduino → delay(200)

end action

Click on "delete" to delete the function.

The screenshot shows the Arduino IDE interface. On the left, there are navigation icons: 'my scripts', 'run', 'undo', and 'split'. The main workspace is divided into two panes. The top pane shows the 'Pedometer' script properties with buttons for 'publish', 'arduino upload', and 'serial monitor'. Below this is a '+ add new action, event, ... or event, variable, library reference, record' button. The bottom pane is titled 'code' and contains three actions: 'main()', 'setup()', and 'variables()'. The 'main()' action is selected and expanded, showing its code: 'action main ()', 'This action emulates the code loop in the bug and will not be compiled.', '▷ setup', 'while true do', '▷ → loop', 'Ⓢ i cannot find property 'loop' on code [TD112]', 'end while', and 'end action'. A red error message is visible next to the 'loop' line.

You will be redirected to the Main function. Click on loop statement and delete it from the codes.

## LET'S START CODING OUR PEDOMETER!

**action** main ()

▷ setup

**var** counter := 0

We declare a counter variable to count the number of steps the user has taken in total

Create a variable and rename it to "counter" and initialise it to 0.

**action** main ()

▷ setup

**var** counter := 0

We declare a counter variable to count the number of steps the user has taken in total

**while true do**

▷ **var** p := engduino → acceleration

Next we create a variable "p" and assign it with the Engduino's accelerometer!

**action** main ()

▷ setup

**var** counter := 0

We declare a counter variable to count the number of steps the user has taken in total

**while true do**

▷ **var** p := engduino → acceleration

**var** x := p → x

**var** y := p → y

**var** z := p → z

We assign 3 different variables to store our X,Y and Z values of the Engduino's Accelerometer

Next, we store the accelerometer's X,Y and Z readings to 3 variables.

**action** main ()

▷ setup

**var** counter := 0

We declare a counter variable to count the number of steps the user has taken in total

**while true do**

▷ **var** p := engduino → acceleration

**var** x := p → x

**var** y := p → y

**var** z := p → z

We assign 3 different variables to store our X,Y and Z values of the Engduino's Accelerometer

**var** acceleration := x \* x + y \* y + z \* z

acceleration := math → sqrt(acceleration)

In order to calculate the acceleration from the Engduino, we have to sum the square of x,y,z and square root the summation

We create a new variable "acceleration" which sums the square of X,Y and Z.

We then square root the summation to get the real acceleration.

The acceleration formula is given as :

$$\text{Acceleration} = \sqrt{x^2 + y^2 + z^2}$$

The acceleration formula calculates the user's acceleration by using all 3 axis of the accelerometer to determine if the user has taken a step

\*Note that this calculation is not the real way of how an actual pedometer calculates a step!

**action** main ()

▷ setup

**var** counter := 0

We declare a counter variable to count the number of steps the user has taken in total

**while true do**

▷ **var** p := Engduino → acceleration

**var** x := p → x

**var** y := p → y

**var** z := p → z

We assign 3 different variables to store our X,Y and Z values of the Engduino's Accelerometer

**var** acceleration := x \* x + y \* y + z \* z

acceleration := math → sqrt(acceleration)

In order to calculate the acceleration from the Engduino, we have to sum the square of x,y,z and square root the summation

**if** acceleration ≥ 1.1 **then**

counter := counter + 1

For us to register a step, we have to make sure the acceleration from the user is at least of acceleration 1.1

In order for us to know if a user has taken a step, we have to check if the acceleration of the user is above 1.1g. If the acceleration is above 1.1g, we consider it as a step and we add the step to the counter!

## CREATING LED OUTPUT ACCORDING TO NO. OF STEPS TAKEN.

The image shows the Arduino IDE interface. On the left, the 'code' editor displays the code from the previous block. A red box highlights the 'add new action, event, ... or event, variable, library reference, record' button. A red arrow points to this button with the text 'Click this'. On the right, the 'create a new ...' dialog box is open, showing a search bar and a list of options. The 'action()' option is highlighted with a red box, and a red arrow points to it with the text 'Click this'. Other options in the list include 'page()', 'library', 'data', 'picture resource', and 'sound resource'.

We will add a new action to show the output of the counter on the Engduino's LED

  
dismiss

  
run

  
undo

  
split

## action

StepCounter

**private action** StepCounter ()  
 | do nothing  
**end** action

private action 

Private actions do not get a run button.

'atomic' action [more info](#)

+

add input parameter 

create a new parameter

+

add output parameter

create a new parameter

cut

copy

delete

We will rename our action to "StepCounter" and make it a private action and also add an input parameter.

```

private action StepCounter (
  | step : Number)
do
  | for 0 ≤ i < 17 do
    | if i = 0 then
      | i + 1
      | '+' returns a 'Number'; insert 'post to wall' if you want to display it
    | else do nothing end if
    | engduino → set LED(0, colors → black)
  
```

We rename our input parameter to "step" (This input will be the *counter* from the main function!) and include a "for" loop with a range of 0 to 16 because we want to light up all 16 LEDs on the Engduino.

We also include a statement to make  $i=1$  as the LED output starts from 1 and not 0.

```

private action StepCounter (
  step : Number)
do
  for 0 ≤ i < 17 do
    if i = 0 then
      i + 1
      '+' returns a 'Number'; insert 'post to wall' if you want to display it
    else do nothing end if
    engduino → set LED(0, colors → black)
    var count := i * 20
    if step > count then
      engduino → set LED(i, colors → green)
      i + 1
      '+' returns a 'Number'; insert 'post to wall' if you want to display it

```

Next, we create a new variable called 'count' and initialize it to be the number of steps we want the user to take so that a LED will light up. In this case, we want the user to walk 20 steps before lighting one LED up.

In order for all 16 LEDs to be lighted up on the Engduino, the user has to walk :

$$16 * \text{No. of steps you want the user to take}^*$$

So in the case of our code, the user has to walk 320 steps so that all the LED would light up.

\*You can assign any value you want the user to walk before lighting up an LED!



```

private action StepCounter (
  step : Number)
do
  for 0 ≤ i < 17 do
    if i = 0 then
      i + 1
      '+' returns a 'Number'; insert 'post to wall' if you want to display it
    else do nothing end if
    engduino → set LED(0, colors → black)
    var count := i * 20
    if step > count then
      engduino → set LED(i, colors → green)
      i + 1
      '+' returns a 'Number'; insert 'post to wall' if you want to display it
      if i = 16 then
        engduino → set all LEDs(colors → black)
        engduino → delay(500)
        engduino → set all LEDs(colors → green)
        engduino → delay(500)
      else do nothing end if
    else do nothing end if
  end for
end action

```

beta 80345 © 2015 Microsoft

In the code above, we add a conditional statement to check if the user have completed the required amount of steps to light up all the LED on the Engduino. If the user have completed the requirement, the Engduino will start flashing to indicate that he/she is done.

## COMPLETING OUR PEDOMETER!


```
action main ()
```

```
  ▷ setup
```

```
    var counter := 0
```

We declare a counter variable to count the number of steps the user has taken in total

```
  while true do
```

```
    var p :=  engduino → acceleration
```

```
    var x := p → x
```

```
    var y := p → y
```

```
    var z := p → z
```

We assign 3 different variables to store our X,Y and Z values of the Engduino's Accelerometer

```
    var acceleration := x * x + y * y + z * z
```

```
    acceleration := math → sqrt(acceleration)
```

In order to calculate the acceleration from the Engduino, we have to sum the square of x,y,z and square root the summation

```
    if acceleration ≥ 1.1 then
```

```
      counter := counter + 1
```

For us to register a step, we have to make sure the acceleration from the user is at least of acceleration 1.1

```
    else do nothing end if
```

```
  ▷ StepCounter(counter)
```

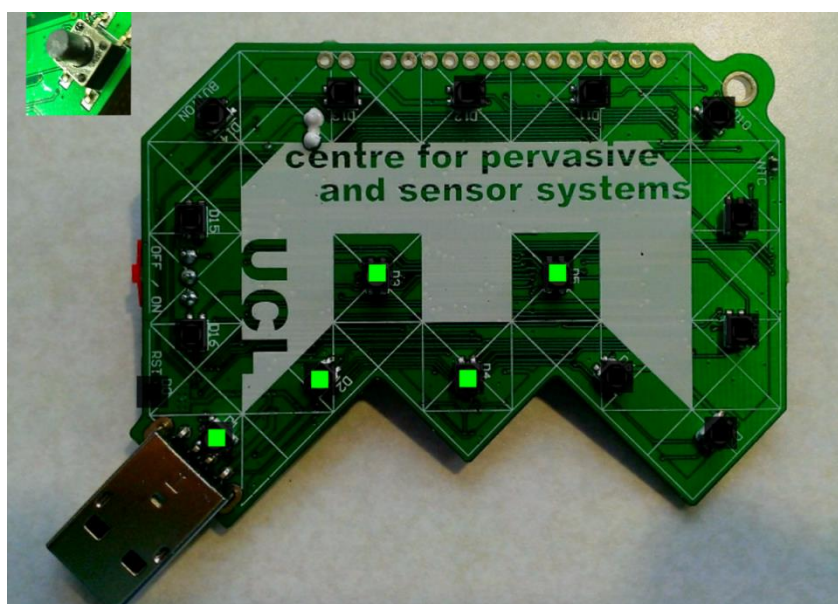
We create a new function to show the number of steps taken

```
end while
```

```
end action
```

In order to finish our Pedometer, we have to add the "StepCounter" function in our main function and pass in the "counter" variable as the input parameter!

### Final Product!



## CHALLENGE PROBLEM

For every 4 completed sets of steps taken, the LED will light up from Red to Orange to Yellow and lastly Green. The first 4 set of LED should be Red and once the user complete the 5th set, all the lighted LED will change to Orange instead. The table below shows the full problem that you should solve!

Challenge your friends to see who is able to solve this problem the fastest and show the result to your teacher!

Steps taken (Steps needed for one set: 20)	LED color	LED
0 - 4 * Steps needed ( 0 - 80 steps)	Red	1,2,3,4
5 * Steps needed - 8 * Steps Needed ( 100 - 160 steps )	Orange	1,2,3,4,5,6,7,8
9 * Steps needed - 12 * Steps needed (180 steps - 240 steps )	Yellow	1,2,3,4,5,6,7,8,9,10,11,12
13 * Steps needed - 16 * Steps needed (260 steps - 320 steps )	Green	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
Above 16 * Steps needed (Above 320 steps)	Green	All LED blinking

## REFERENCES

Full working code of tutorial : [TouchDevelop Pedometer](#)